Интерактивная цифровая книга для профессионалов

Анатолий Постолит

Мастер web разработки на Python и Django



Postolit Press

Постолит А.В.

Мастер веб-разработки на Python и Django (интерактивная цифровая книга). — М.: Postolit Press, 2024. — 615 с.: 294 ил.

Интерактивная цифровая книга предназначена для практикующих программистов, имеющих опыт работы с Django. В книге раскрываются некоторые особенности Django и Bootstrap, которые позволяют создать набор сервисов, обеспечивающих модификацию дизайна и содержания веб-страниц готового сайта без изменения программного кода. Когда сайт уже выложен на публичный сервер, администратор сайта (и даже пользователи с правами администратора) через личный кабинет будут иметь возможность добавлять или удалять элементы интерфейса, менять дизайн страниц сайта, структуру древовидных каталогов продуктов и услуг, содержимое базы данных, не затрагивая программный код сайта.

В книге читатель найдет множество примеров программного кода, который можно не только просмотреть и скачать, но и выполнить в рамках самой интерактивной цифровой книги, так как она содержит весь необходимый инструментарий для выполнения программных модулей. Все примеры программного кода взяты из разработанных сайтов, находящихся в промышленной эксплуатации, поэтому они гарантировано не содержат ошибок, их можно скачать и использовать в своих проектах.

Книга предназначена для программистов, имеющих опыт работы с Python, Django и Bootstrap. Она может быть использована как лабораторный практикум для студентов вузов и слушателей компьютерных курсов при изучении практических приемов программирования веб-приложений.

Оглавление

ПРЕДИСЛОВИЕ 8
Структура книги 9
О примерах кода… 10
ГЛАВА 1. МАКЕТИРОВАНИЕ HTML-СТРАНИЦ С ФРЕЙМВОРКОМ BOOTSTRAP 5 11
1.1. Что нового в Bootstrap 5? 11
1.1.1. Наиболее важные изменения 12
1.1.2. Наиболее важные дополнения 13
1.1.3. Новые компоненты 15
1.1.4. Прочие улучшения и изменения 15
1.2. Получение файлов фреймворка Bootstrap 16
1.2.1. Использование сети CDN 16
1.2.2. Скачивание файлов Bootstrap 18
1.3. Контейнеры и сетка Bootstrap 18
1.3.1. Адаптивные контейнеры 19
1.3.2. Ряды, или строки (*row*) 21
1.3.3. Адаптивные блоки (*col*) 21
1.3.4. Адаптивные блоки без указания числа колонок 22
1.3.5. Расположение адаптивных блоков 22
1.4. Верстка макета HTML-страниц 23
1.4.1. Выравнивание адаптивных блоков 23
1.4.2. Смещение адаптивных блоков 24
1.5. Простейшая HTML-страница с подключением фреймворка Bootstrap 28
1.6. Задание цвета элементам HTML-страниц 29
1.7. Задание отступов элементам макета HTML-страниц 31
1.8. Выравнивание содержимого в адаптивных блоках HTML-страниц 33
1.9. Обозначение границ элементов макета HTML-страниц 34
1.10. Пример использования адаптивных контейнеров 38
1.11. Таблицы Bootstrap 39
1.12. Адаптивные таблицы в Bootstrap (с полосами прокрутки) 42
1.13. Краткие итоги 44
ГЛАВА 2. СТАТИЧНЫЕ ЭЛЕМЕНТЫ BOOTSTRAP В ШАБЛОНАХ DJANGO 46
2.1. Задание произвольного цвета элементам Bootstrap 46
2.2. Тег *include* для встраивания шаблонов Django 48
2.3. Подключение значков к элементам Bootstrap 49
2.4. Разделение страницы на вкладки компонентом *Tabs* 52
2.4.1. Традиционные горизонтальные вкладки *Tabs* фреймворка Bootstrap 52
2.4.2. Традиционные вертикальные вкладки *Tabs* фреймворка Bootstrap 53
2.4.3. Горизонтальные вкладки *Tabs* с использованием тега Django для организации циклов 55
2.4.4. Вертикальные кнопки для навигации по вкладкам *Tabs* фреймворка Bootstrap 56
2.5. Компонент Bootstrap *Ofcanvas* 57
2.6. Компонент для выдачи подсказок *Tooltip* 60
2.7. Использование атрибута *href* в шаблонах Django для создания ссылок на URL-адреса 60
2.7.1. Ссылки на внешние URL-адреса 60
2.7.2. Ссылки на внутренние URL-адреса 62
2.7.3. Ссылки на внутренние динамические URL-адреса 63
2.7.4. Использование атрибута *href* для обмена параметрами между шаблонами и представлениями 65

2.8. Всплывающие модальные окна 69
2.8.1. Простое модальное окно 72
2.8.2. Модальное окно с полосой прокрутки 73
2.8.3. Широкое модальное окно 74
2.9. Компонент Bootstrap *Accordion* «Аккордеон» 75
2.9.1. Класс *accordion* 76
2.9.2. Класс *accordion-flush* 78
2.10. Компонент Bootstrap *card* «Карточки» 79
2.10.1. Карточка с параметрами по умолчанию 80
2.10.2. Карточка с параметрами цвета 80
2.10.3. Группы карточек 81
2.11. Компонент Bootstrap *Carousel* «Карусель» 82
2.11.1. Статичная карусель с фиксированным контентом 82
2.12. Краткие итоги 84
ГЛАВА 3. ДИНАМИЧНЫЕ ЭЛЕМЕНТЫ BOOTSTRAP В ШАБЛОНАХ DJANGO 85
3.1. Динамичный «Аккордеон» с заполнением вкладок через административную панель Django 85
3.2. Динамичный «Аккордеон» с заполнением вкладок через интерфейс пользователя 88
3.2.1. Программные модули для создания динамичных элементов «Аккордеон» 90
3.2.2. Модули для вывода динамичных аккордеонов на страницы сайта 93
3.2.3. Пример создания динамичного аккордеона через интерфейс пользователя 95
3.2.4. Пример вывода динамичного аккордеона на страницу сайта 97
3.3. Динамичные карточки с заполнением контента через административную панель 100
3.4. Динамичные карточки с заполнением контента через интерфейс пользователя 103
3.4.1. Программные модули для создания динамичных карточек 104
3.4.2. Пример формирования динамичных карточек через интерфейс пользователя 106
3.4.3. Модули для вывода динамичных карточек на страницы сайта 108
3.4.4. Пример отображения динамичных карточек на страницах сайта 109
3.5. Динамичная карусель с заполнением контента через административную панель 111
3.6. Динамичная карусель с заполнением контента через интерфейс пользователя 113
3.6.1. Программные модули для создания слайдов динамичной карусели 114
3.6.2. Пример формирования слайдов динамичной карусели через интерфейс пользователя 116
3.6.3. Модули для вывода динамичных каруселей на страницы сайта 118
3.6.4. Примеры отображения динамичных каруселей на страницах сайта 119
3.7. Краткие итоги 119
ГЛАВА 4. ДРЕВОВИДНЫЕ СТРУКТУРЫ В DJANGO 120
4.1. Использование Django-mptt для создания древовидных структур 120
4.2. Формирования дерева категорий продуктов с использованием административной панели Django 123
4.2.1. Отображение статичного дерева категорий продуктов в шаблонах Django 124
4.2.2. Отображение динамичного дерева категорий продуктов в шаблонах Django 126
4.3. Формирования дерева категорий продуктов с использованием интерфейса пользователя 127
4.3.1. Модель для категорий продуктов 127
4.3.2. Модули для отображения дерева категорий продуктов 128
4.3.3. Модули для добавления корня к дереву категорий продуктов 129
4.3.4. Модули для добавления дочерних категорий продуктов 132
4.3.5. Модули для обновления категорий продуктов 136
4.3.6. Модули для удаления категорий продуктов 136
4.4. Создание списков объектов с фильтрацией через древовидные каталоги 137
4.4.1. Модель для хранения категорий продуктов 138

4.4.2. Модель для хранения сведений о продуктах 139
4.4.3. Форма для ввода сведений о продуктах 140
4.4.4. Редактор продуктов 140
4.4.5. Модули для добавления в БД продуктов (товара) 142
4.4.6. Модули для обновления сведений о продуктах 144
4.4.7. Модули для удаления продукта из БД 145
4.4.8. Модули для показа подробностей о продуктах 146
4.5. Фильтрация сведений о продуктах с использованием статичного древовидного каталога категорий продуктов 147
4.6. Фильтрация сведений о продуктах с использованием динамичного древовидного каталога категорий продуктов 150
4.7. Создание древовидных меню с использованием плагина megamenu-js 154
4.7.1. Плагин megamenu-js: подготовка к работе 154
4.7.2. Создание мегаменю на основе плагина megamenu-js 156
4.8. Фильтрация продуктов с использованием панели инструментов и древовидного каталога категорий 159
4.9. Краткие итоги 168
ГЛАВА 5. БИБЛИОТЕКА HTMX: МОДАЛЬНЫЕ ОКНА ДЛЯ ВВОДА И КОРРЕКТИРОВКИ ДАННЫХ 169
5.1. Библиотека HTMX 169
5.2. Ввод и редактирования сведений о клиентах сайта с модальными окнами на основе библиотеки HTMX 171
5.2.1. Модель и форма для ввода данных 171
5.2.2. Модули для создания редактора списка клиентов 171
5.2.3. Модули для добавления клиентов в БД 173
5.2.4. Программируем кнопку *Изменить* — для корректировки в БД сведений о клиентах 174
5.2.5. Программируем кнопку *Удалить* — для удаления из БД сведений о клиентах 175
5.2.6. Программируем модули для показа подробных сведений о клиентах 176
5.2.7. Испытание программных модулей для редактирования сведений о клиентах в модальных окнах 176
5.3. Краткие итоги 182
ГЛАВА 6. ИНТЕРНЕТ-МАГАЗИН 183
6.1. Обмен данными между представлениями и шаблонами 183
6.1.1. Обмен данными через сессии 183
6.1.2. Обмен данными через контекстный процессор 184
6.2. Модели для корзины и заказов клиентов магазина электронной торговли 185
6.3. Контекстный процессор для магазина электронной торговли 186
6.4. Создаем магазин пользователя 187
6.4.1. Вывод категорий продуктов в левой боковой панели магазина (дерево свернуто) 188
6.4.2. Проверка работы модулей магазина пользователя с фильтрацией данных по свернутому дереву категорий продуктов 190
6.4.3. Вывод категорий продуктов в левой боковой панели магазина (дерево раскрыто) 193
6.4.4. Проверка работы модулей магазина пользователя с фильтрацией данных по раскрытому дереву категорий продуктов 195
6.4.5. Вывод категорий продуктов в центральной панели в виде мегаменю 198
6.4.6. Проверка работы модулей магазина пользователя с фильтрацией данных по дереву категорий продуктов в виде мегаменю 199
6.5. Модули формирования корзины (предварительного заказа) .202
6.5.1. Добавление продуктов (товаров) в корзину 202
6.5.2. Вывод содержимого корзины (предварительного заказа) 204
6.5.3. Корректировка количества товара в корзине 205
6.5.4. Удаление товара из корзины 209
6.5.5. Формирование заказа для оплаты на основе набора продктов, помещенных в корзину 210
6.5.6. Удаление корзины или активного заказа 212
6.5.7. Проверка работы модулей для перевода корзины в предварительный заказ 213
6.5.8. Интеграция интернет-магазина с платежной платформой Robokassa 217
6.5.9. Обработка сбоя оплаты заказа пластиковой картой 219
6.5.10. Обработка успешной оплаты заказа банковской картой 219

6.5.11. Обработка реальной оплаты заказа банковской картой на платежной платформе Robokassa ... 220
6.6. Проверка работы модулей оплаты предварительного заказа 221
6.7. Модули для показа оплаченных заказов (покупок) ... 225
6.8. Проверка работы модулей для показа оплаченных заказов (покупок) ... 226
6.9. Краткие итоги ... 228
ГЛАВА 7. ПОЛЕЗНЫЕ СОВЕТЫ ... 229
7.1. Показываем на сайте документ PDF ... 229
7.1.1. Создание документа в формате PDF ... 229
7.1.2. Вывод документа формата PDF из представления ... 231
7.1.3. Вывод документа формата PDF из шаблона Django ... 233
7.1.4. Вывод PDF-документа в формате флеш-анимации ... 235
7.2. Переводим страницы сайта на другие языки ... 239
7.3. Отправляем с сайта почтовые сообщения ... 242
7.3.1. Регистрация в почтовом сервисе Timeweb ... 242
7.3.2. Программные модули для отправки почтовых сообщений ... 244
7.3.3. Проверка работы модулей отправки почтовых сообщений ... 246
7.4. Используем скрипты в шаблонах Django ... 248
7.4.1. Скрипты для придания страницам сайта новогоднего стиля ... 248
7.4.2. Скрипты для анимации кнопок ... 251
7.5. Используем фреймворк PyScript в шаблонах Django ... 253
7.5.1. PyScript: отображение в шаблоне Django содержимого символьной переменной и списка ... 253
7.5.2. PyScript: импорт дополнительных библиотек, создание функции и выполнение обращений к ним ... 255
7.5.3. PyScript: работа с пакетами NumPy и pandas ... 256
7.5.4. PyScript: работа с внешними библиотеками для обработки и визуализации данных 258
7.6. Используем пакет Brython для создания программ на Python для HTML-страниц ... 260
7.6.1. Brython: вывод на HTML-страницу обычной символьной строки ... 260
7.6.2. Brython: игра «Змейка» (создание окон, функций и обращений к ним) ... 261
7.7. Откатываем миграцию ... 263
7.8. Краткие итоги ... 264
ГЛАВА 8. ВИЗУАЛИЗАЦИЯ ДАННЫХ В DJANGO ... 265
8.1. Визуализация данных с библиотекой plotly.py: двумерная графика (2D) ... 265
8.1.1. Создание статичных графиков (одна линия) ... 268
8.1.2. Создание статичных графиков (несколько линий) ... 270
8.1.3. Отображение двух графиков в одном окне макета ... 272
8.1.4. Отображение четырех графиков в одном окне макета ... 273
8.1.5. Создание 2D-диаграммы рассеяния ... 275
8.1.6. Создание круговой диаграммы ... 277
8.1.7. Создание столбчатой гистограммы ... 278
8.1.8. Создание столбчатой гистограммы и сглаживающей кривой ... 280
8.1.9. Создание графиков с тепловой картой ... 281
8.1.10. Создание пузырьковой диаграммы ... 283
8.1.11. Пузырьковая диаграмма с цветными маркерами ... 284
8.1.12. Пузырьковая диаграмма с масштабируемыми маркерами ... 286
8.1.13. Пузырьковая диаграмма с поясняющим текстом на маркерах ... 288
8.1.14. Пузырьковая диаграмма с тепловой картой ... 289
8.1.15. Отображение статичных элементов на географических картах ... 291
8.2. Визуализация данных с библиотекой plotly.py: трехмерная графика (3D) ... 293
8.2.1. Создание 3D-графика линии по точкам ... 293
8.2.2. Создание 3D-графика с указанием параметров линии ... 295
8.2.3. Создание 3D-графика плавной витой линии ... 297
8.2.4. Создание 3D-графика линии с ключевыми точками ... 298
8.2.5. Создание 3D-графика плоской поверхности ... 300
8.2.6. Создание 3D-графика изогнутой поверхности с использованием функций ... 302
8.2.7. Создание 3D-графика изогнутой поверхности с контурами по вертикальной оси ... 304
8.2.8. Создание 3D-графика поверхности с контурами по всем осям ... 305

8.2.9. Создание 3D-графика поверхности с указанием параметров сцены и положения камеры ... 307
8.2.10. Создание 3D-графика поверхности с контурной сеткой ... 310
8.2.11. Создание 3D-диаграммы рассеяния точек ... 312
8.3. Визуализация интерактивных графиков с библиотекой Plotly.py ... 313
8.3.1. Создание интерактивного 2D-графика со слайдером ... 314
8.3.2. Комбинирование интерактивного графика с анимированными изображениями ... 316
8.3.3. Вывод двух интерактивных графиков в одну координатную сетку ... 319
8.3.4. Вывод статичных и интерактивных графиков в одну координатную сетку ... 322
8.3.5. Анимация графиков с исчезновением элементов ... 324
8.3.6. Анимация графиков с появлением элементов ... 327
8.3.7. Анимация графиков с использованием фреймов и кнопок ... 330
8.3.8. Анимация графиков с использованием фреймов, кнопок и слайдера ... 332
8.3.9. Анимация пузырьковой диаграммы ... 334
8.3.10. Анимация объемных 3D-срезов на примере снимков МРТ ... 337
8.3.11. Географические карты с анимацией данных пользовательских форм ... 339
8.3.12. Географические карты с анимацией данных кнопками и слайдером ... 342
8.4. Краткие итоги ... 344
ГЛАВА 9. ВИЗУАЛИЗАЦИЯ ДАННЫХ В DJANGO С АНИМАЦИЕЙ НА JAVASCRIPT ... 345
9.1. Включение JavaScript в шаблон Django ... 345
9.1.1. Анимация диаграмм формата 2D ... 346
9.1.2. Анимация трехмерной диаграммы формата 3D ... 350
9.2. Краткие итоги ... 355
ПОСЛЕСЛОВИЕ ... 356
СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ ... 357
ПРИЛОЖЕНИЕ 1. ПОРЯДОК РАБОТЫ С ИНТЕРАКТИВНОЙ ЦИФРОВОЙ ПЛАТФОРМОЙ ... 359
Просмотр карточек с загруженными книгами ... 360
Просмотр вкладок цифровой книги ... 363
Вкладка *Текст PDF* ... 364
Вкладка *Листинги* ... 364
Вкладка *Показ примеров* ... 366
ПРИЛОЖЕНИЕ 2. РЕГИСТРАЦИЯ ИНТЕРНЕТ-МАГАЗИНА НА САЙТЕ ПЛАТФОРМЫ ROBOKASSA ... 368

Предисловие

Специалисты IT-технологий уже много лет занимают первые позиции в рейтингах кадровых агентств, как представители наиболее востребованных профессий на отечественном и мировом рынке труда. Постоянная нехватка квалифицированных программистов дает высокий шанс молодым и целеустремленным новичкам для входа в эту профессию. В ближайшие несколько лет дефицит IT-специалистов будет только нарастать, так как они нужны повсеместно. Если раньше ими интересовались сугубо профильные компании (разработчики софта и онлайн-платформ), то сейчас квалифицированные кадры IT-профилей требуются практически везде. И речь здесь идет не о дефиците, а о катастрофическом дефиците. Уже много лет подряд сервис по поиску работы hh.ru фиксирует огромный неудовлетворенный спрос на IT-специалистов.

Рассматриваемый в этой книге язык программирования Python является сам популярным и востребованы языком программирования в мире по индексу TIOBE, и области его применения только расширяются. Используя Python, можно решать множество научных проблем и задач в области бизнеса. К нему обращаются ученые (математики, физики, биологи), так как изучить его не слишком сложно. Он используется при реализации нейронных сетей, для написания веб-сайтов и мобильных приложений. В целом это универсальный язык, входящий в тройку языков для анализа больших данных. У Python много обширных библиотек, которые постоянно обновляют разработчики из сообщества. Благодаря этому, писать код не нужно с нуля, всегда можно найти готовые фрагменты и отредактировать их под конкретные задачи. Еще одна причина, по которой следует обратить внимание на язык Python, — это его использование для веб-разработки совместно с фреймворками Django и Bootstrap. Здесь Django обеспечивает разработку серверной части, а Bootstrap — клиентской части веб-приложений. На Django созданы такие сайты, как YouTube, Facebook, Instagram, а Bootstrap — это детище компании Twitter.

Перед теми, кто решил освоить специальность веб-программиста, встает непростой выбор: с чего же правильно начать? Конечно, всегда существует возможность получить IT-образование в одном из ведущих технических вузов ранга МГУ, МГТУ им. Н. Баумана, СПбГУ, МФТИ, ИТМО. Но обучение в них обойдется в круглую сумму от 60 до 350 тыс. рублей в год. Существует и более быстрый вариант стать веб-разработчиком «с нуля», пройдя краткосрочные онлайн-курсы, которые тоже стоят немалых денег. Однако практикующие программисты уверяют, что самый правильный, а главное, практически бесплатный способ обучиться веб-программированию, — это освоить его самостоятельно. Так можно не только избежать серьезных расходов, но и получить только те практические навыки, которые пригодятся в будущей работе.

Эта книга предназначена не для начинающих программистов, а для специалистов с опытом, которые уже занимались разработкой веб-приложений с использованием Python и Django. Начинающим программистам сначала нужно получить базовые знания, изучив материалы такой, например, книги, как «Python, Django и Bootstrap для начинающих»[1]. Сразу следует отметить, что веб-программирование требует от разработчиков бо̲льших знаний, умений и усилий, чем программирование традиционных приложений. Здесь, кроме основного языка, который реализует логику приложения, необходимо знание структуры HTML-документов, основ работы с базами данных, принципов взаимодействия удаленных пользователей с веб-приложением.

Книга, как уже отмечено, предназначена для программистов, имеющих практику создания приложений на Python, Django и Bootstrap. В ней раскрываются некоторые особенности Django и Bootstrap, которые позволяют создать набор сервисов, обеспечивающих модификацию дизайна и содержания веб-страниц готового сайта без изменения его программного кода. Когда сайт уже выложен на публичный сервер, администратор сайта (и даже пользователи с правами администратора) через личный кабинет будут иметь возможность добавлять или удалять элементы интерфейса, менять дизайн страниц сайта, структуру древовидных каталогов продуктов и услуг, содержимое базы данных, не затрагивая программный код сайта. При разработке такого типа сайтов вам придется затратить больше усилий на написание программного кода, однако сопровождать работу готового сайта будет в десятки раз проще.

---

[1] См. **https://bhv.ru/product/python-django-i-bootstrap-dlya-nachinayushhih/**.

Структура книги

Первая глава книги посвящена макетированию HTML-страниц с использованием новых элементов фреймворка Bootstrap 5. В ней рассказывается о наиболее важных и новых компонентах Bootstrap 5. Эти материалы станут основой для разработки программных модулей, обеспечивающих удаленное управление дизайном шаблонов Django без изменения программного кода.

Во *второй главе* книги описываются параметры и способы создания статичных элементов Bootstrap для шаблонов Django: управление цветом элементов Bootstrap, применение тега `include` для встраивания одних шаблонов в другие, использование значков и вкладок в шаблонах Django, а также кнопок и панелей для навигации по сайту. Рассмотрены особенности использование атрибута `href` в шаблонах Django для создания ссылок на интернет-адреса (URL), создание всплывающих модальных окон. Приведены программные коды для создания таких статичных компонентов, как аккордеоны, карточки и карусели.

В *третьей главе* представлены технологии программирования динамичных элементов Bootstrap для шаблонов Django. Дизайн и наполнение таких элементов можно менять либо через административную панель Django, либо через специально разработанный интерфейс личного кабинета административного персонала сайта. В качестве примеров взяты следующие динамические элементы: аккордеон, карточки, карусели. Эти элементы можно удаленно добавлять (удалять) на различные страницы сайта, менять содержимое элементов, их дизайн (размер, цвет фона, цвета и размеры шрифта, размеры и цвета рамок и т. п.). Все эти операции допускается делать удаленно через личный кабинет административного персонала сайта, не затрагивая программный код.

Четвертая глава посвящена использованию древовидных структур в шаблонах Django на основе специализированного пакета Django-mptt. Древовидные структуры рассматриваются на примере дерева категорий продуктов (товаров) —наиболее распространенного элемента, широко применяемого на сайтах интернет-торговли. В главе описываются соответствующие модели и возможности формирования дерева категорий продуктов с использованием административной панели Django, приводятся программные коды для формирования дерева категорий продуктов на основе возможностей интерфейса пользователя, рассматриваются технология и программные модули создания списков объектов, а также показаны программные модули для фильтрации сведений о продуктах с применением статичных и динамичных древовидных каталогов категорий продуктов. Приведены примеры создания древовидных меню с использованием плагина megamenu js.

П*ятая глава* содержит примеры создания модальных окон при вводе и корректировке данных с использованием библиотеки HTMX. В ней приведены примеры добавления, обновления и удаления данных из БД с интерфейсными формами в виде модальных окон. Программный код для создания модальных окон более сложен, чем создание традиционных шаблонов Django. Однако интерфейс пользователя, построенный на модальных окнах, выглядит более привлекательно, а главное, более удобен для конечного пользователя.

В *шестой главе* приведен базовый набор программных модулей для создания интернет-магазина. Здесь описывается технология обмена данными между представлениями и шаблонами на основе переменных сессии и контекстными процессорами, показана структура моделей для категорий продуктов, продуктов, корзины, заказов и покупок, приводятся примеры программных кодов для формирования разного вида шаблонов с категориями продуктов и с карточками продуктов, рассмотрены примеры программных кодов для формирования и корректировки содержания корзины, предварительного заказа, создания заказа и его оплаты банковскими картами через платежную платформу Robokassa.

В седьмой главе приводятся некоторые полезные советы, которые пригодятся при разработке собственных проектов. Эти советы получены на основе обзора и использования опыта сообщества программистов на Python и Django. В частности, на примерах показано, как в шаблонах Django можно отобразить PDF-документы и флеш-анимации, как реализовать динамичный перевод страниц сайта на сотни других языков, как с сайта отправлять почтовые сообщения, как использовать в шаблонах Django Java-скрипты, и скрипты, написанные на Python.

В восьмой главе представлены материалы, связанные с выводом данных в шаблоны Django в виде статичных и интерактивных 2D-графиков, а также в виде трехмерных 3D-диаграмм точек, линий и поверхностей с элементами анимации. Рассмотрены возможности отображения данных на географических картах, анимации элементов географических карт, сделан

разбор функциональной сущности наиболее значимых фрагментов программного кода, построенного с использованием библиотеки Plotly.py.

Девятая глава посвящена использованию библиотеки Plotly.js для визуализации данных в шаблонах Django в виде статичных и интерактивных 2D- и 3D-диаграмм, с учетом того, что анимация диаграмм будет выполняться с помощью Java-скриптов, встроенных в шаблоны Django. Такой подход позволяет сократить программный код в представлениях (view) и перенести его в шаблоны. Рассмотрены возможности создания «живых» интерактивных диаграмм на основе Java-скриптов, встроенных в шаблоны Django, а также на примерах показана технология взаимодействия шаблонов Django с Java-скриптами. Этот подход оправдан для той группы программистов, которые имеют навыки и опыт работы с языком программирования Java.

О примерах кода…

На протяжении всей книги раскрываемые вопросы сопровождаются по возможности упрощенными, но полностью законченными примерами. Ход решения той или иной задачи показан на большом количестве иллюстративного материала. Желательно изучение тех или иных разделов осуществлять, сидя непосредственно за компьютером, — тогда вы сможете последовательно повторять выполнение тех шагов, которые описаны в примерах, и тут же видеть результаты своих действий, что в значительной степени облегчит восприятие материала книги.

Некоторые фрагменты программного кода этой книги могут показаться сложными, но они взяты из реально работающих сайтов, развернутых на публичных серверах. В них гарантированно нет ошибок и использованы разработанные сообществом действующих программистов приемы, о которых ничего не сказано в официальной документации. Абсолютно все примеры сопровождающей книгу интерактивной цифровой платформы (см. *приложение*) можно запустить на выполнение в рамках самой книги, так как она содержит весь необходимый инструментарий. Когда вы увидите результаты работы программного кода, появится ясность — как работает тот или иной элемент изучаемого фреймворка. Мало того, зарегистрированные пользователи платформы могут скачать листинги ее программных кодов и использовать их в своих проектах.

> ***Интерактивная цифровая платформа***
> Работать с интерактивной цифровой платформой можно, зайдя в нее по адресу **rumnik.ru**. Демо-версию платформы можно найти в опции **Каталог** ее главного меню. Полная версия платформы будет доступна в разделе **Мои книги** личного кабинета зарегистрированного пользователя после ее приобретения в магазине сайта.
>
> Более того, использование полной версии платформы избавляет вас от прибретения «бумажной» версии книги — если вам это покажется более удобным…

Наилучший способ обучения — это практика. Все листинги программ приведены на языке Python с элементами Java-скриптов, а шаблоны веб-страниц — в виде HTML-файлов. Любой пример, приведенный в интерактивной цифровой платформе, можно запустить на выполнение и посмотреть, как выглядит конечный результат.

Итак, если вас заинтересовали вопросы создания эффективных веб-приложений с использованием Python, Django и Bootstrap в среде PyCharm, то самое время перейти к изучению материалов этой книги и ее цифровой интерактивной платформы.

Глава 1. Макетирование HTML-страниц с фреймворком Bootstrap 5

Что такое фреймворк? Можно сказать, что фреймворк — это некий конструктор или набор готовых компонентов и правил, с помощью которых страницы сайта можно создать проще и быстрее, чем весь код писать с нуля. Bootstrap — это фреймворк для создания видимого интерфейса страниц сайтов. Главное, за что программисты любят Bootstrap, — это простота использования и адаптивность разработанных страниц сайтов к экранам разного разрешения. На нем можно создать сайт, который будет выглядеть одинаково хорошо на экранах компьютеров, планшетов и смартфонов. Кроме того, это бесплатный фреймворк, который подключается к страницам сайта всего несколькими строчками кода.

Bootstrap 5 (текущая основная версия, выпущенная в мае 2021 года) принес с собой массу изменений и улучшений, включая добавление новых компонентов, новых классов, новый стиль для старых компонентов, обновленную поддержку браузеров, удаление некоторых старых компонентов и многое другое. Можно с уверенностью сказать, что это самый популярный в мире фреймворк с открытым исходным кодом, который разработан командой Twitter. Он упрощает реализацию сложных многостраничных сайтов и предоставляет множество компонентов, которые, вероятно, понадобятся вам в различных проектах.

Из материалов этой главы вы узнаете:

- о наиболее важных изменениях и новых компонентах, которые появились в Bootstrap 5;
- как можно получить файлы фреймворка Bootstrap;
- что такое контейнеры и сетка Bootstrap;
- как можно сверстать макет HTML-страниц;
- как подключить файлы фреймворка Bootstrap к проекту;
- как задать цвет элементам HTML-страниц;
- как задать отступы в элементах макета HTML-страниц;
- как выровнять содержимое в адаптивных блоках HTML-страниц;
- как можно обозначить границы элементов в макете HTML-страниц;
- как использовать адаптивные контейнеры при макетировании HTML-страниц;
- как создать таблицы в Bootstrap и подключить к ним полосы прокрутки.

1.1. Что нового в Bootstrap 5?

Bootstrap — одна из самых популярных библиотек CSS. Она позволяет разработчикам легко применять красивые стили и компоненты и создавать адаптивные веб-сайты. Использование Bootstrap может сэкономить время разработчиков, особенно, в работе с компонентами, которые присутствуют практически в каждом проекте.

Bootstrap позиционирует себя как «самая популярная в мире платформа для создания адаптивных сайтов, ориентированных на мобильные устройства», и его использование — это отличный способ начать создавать современные и понятные веб-сайты, особенно, для новичков. Он упрощает реализацию сложных проектов, ориентированных на мобильные устройства, и предоставляет множество компонентов, которые, вероятно, понадобятся вам в нескольких проектах.

Bootstrap не предполагает сложного обучения и хорошо подходит для статических веб-сайтов, не требующих этапа сборки, поскольку вы можете просто сослаться на библиотеку из CDN Bootstrap. В этом его отличие от некоторых других популярных фреймворков CSS, которые можно оптимизировать для использования с упаковщиком или средством запуска задач.

В версии Bootstrap 5 имеется масса изменений и улучшений, включая добавление новых компонентов, новых классов, новых стилей для старых компонентов, обновленную поддержку браузеров, удаление некоторых старых компонентов и многое другое.

Выполнить обновление с Bootstrap 4 до версии 5, как правило, довольно просто. Большинство компонентов, которые были доступны в Bootstrap 4, по-прежнему доступны и в

Bootstrap 5. Главное, на чем вам следует сосредоточиться при миграции, — уточнить, были ли удалены используемые вами ранее классы или компоненты. Если они были удалены, то существуют замены или способы достижения того же результата с помощью служебных классов. Второе, на что вам следует обратить внимание, — это переключение с атрибутов `data-*` на компоненты `data-bs-*`, для работы которых требуется JavaScript.

1.1.1. Наиболее важные изменения

Библиотека jQuery более не является зависимостью

jQuery — это бесплатная JavaScript-библиотека с открытым исходным кодом. Она заметно упрощает веб-разработку, позволяя писать меньше кода, чем на «чистом» JavaScript.

Начиная с версии Bootstrap 5, библиотеку jQuery можно не подключать к Bootstrap. Теперь вы можете использовать Bootstrap во всей красе и без этой библиотеки, хотя Popper.js вам всё равно понадобится. Это изменение упрощает применение Bootstrap в проектах, которые не требуют или не задействуют jQuery, — например, при использовании Bootstrap с React.

Программисты по-прежнему могут применять библиотеку jQuery в работе с Bootstrap, если она является частью веб-сайта. Когда Bootstrap обнаружит jQuery в window-объекте, он автоматически добавит все компоненты в систему плагинов jQuery. Итак, если вы переходите с версии 4 на версию 5, вам не нужно беспокоиться об этом изменении, и вы все равно можете задействовать jQuery с Bootstrap при необходимости.

Но что, если вы применяете jQuery на своем веб-сайте, однако не хотите, чтобы Bootstrap использовал jQuery? Вы можете сделать это, добавив атрибут `data-bs-no-jquery` в элемент `body` документа:

```
<body data-bs-no-jquery="true">
</body>
```

Изменение поддержки браузеров

Вплоть до версии 4, Bootstrap поддерживал Internet Explorer (IE) 10 и 11. Начиная с версии Bootstrap 5, поддержка IE полностью прекращена. И если вам нужна поддержка IE на вашем веб-сайте, то следует воздержаться от перехода на версию 5. Другие изменения в поддержке браузеров включают:

- поддержку Firefox и Chrome теперь — начиная с версии 60;
- поддержку Safari и iOS теперь — начиная с версии 12;
- поддержка браузера Android и WebView начинается с версии 6.

Изменение атрибутов данных

Bootstrap 5 изменил названия атрибутов данных, которые обычно используются его компонентами с подключением JavaScript (как части своей функциональности). Раньше большинство компонентов, задействующих некоторые функции JavaScript, имели атрибуты данных, начинающиеся с `data-`. Например, чтобы создать компонент всплывающей подсказки, в Bootstrap 4 применялся следующий код:

```
<button
  type="button"
  class="btn btn-secondary"
  data-toggle="tooltip"
  data-placement="top"
  title="Tooltip"
>
  Tooltip
</button>
```

Обратите внимание на использование `data-toggle` и `data-placement`. В Bootstrap 5 атрибуты данных для этих компонентов теперь начинаются с атрибутов `data-bs` пространства имен Bootstrap. Например, чтобы создать компонент всплывающей подсказки в Bootstrap 5, нужно применить следующий код:

```
<button
```

```
  type="button"
  class="btn btn-secondary"
  data-bs-toggle="tooltip"
  data-bs-placement="top"
  title="Tooltip"
>
  Tooltip
</button>
```

То есть, вместо `data-toggle` и `data-placement` теперь надо писать `data-bs-toggle` и `data-bs-placement`. Если для создания компонентов в Bootstrap вы используете JavaScript, возможно, и нет необходимости вносить какие-либо изменения, однако если для функционирования ваших компонентов задействуются только атрибуты данных, вам необходимо изменить все атрибуты данных, которые начинаются с `data` на `data-bs`.

Что было удалено?

Изменения могут стоить дорого. При выходе новой версии, вносящей изменения и улучшения, из нее удаляются некоторые старые функции, которые поддерживались ранее. Так и в этом, обновленном выпуске Bootstrap, некоторые его компоненты, служебные классы и ряд других функций были удалены.

Причиной исключения из Bootstrap 5 большинства таких элементов явилась возможность достичь тех же результатов, используя служебные классы вместо того, чтобы делать их отдельным компонентом.

Вот список некоторых элементов, удаленных из Bootstrap 5:

1. Полностью прекращена поддержка IE.
2. Цветовые классы значков (например, `.badge-primary`) удалены в пользу служебных классов цветов (например, `.bg-primary`).
3. Класс `.badge-pill` значка, который придавал значку стиль таблетки, удален в пользу класса `.rounded-pill`, который дает тот же результат.
4. Класс `.btn-block` кнопок удален, поскольку того же результата можно достичь, используя служебные классы отображения `.d-block`.
5. Ранее доступный макет карты `.card-columns` исключен, поскольку он имел множество побочных эффектов.
6. Опция `flip` для раскрывающихся компонентов удалена в пользу конфигураций Поппера.
7. Компонент `Jumbotron` удален, поскольку того же результата можно добиться, используя служебные классы.
8. Некоторые из порядковых классов полезности (`.order-*`) исключены, поскольку раньше они находились в диапазоне от 1 до 12. Теперь они варьируются от 1 до 5.
9. Медиа-компонент удален, поскольку того же результата можно достичь с помощью служебных классов.
10. Классы `.thead-light` и `.thead-dark` удалены, поскольку варианты классов `.table-*` можно применять ко всем элементам таблицы.
11. Класс `.pre-scrollable` исключен, так как он мало используется.
12. Класс `.text-justify` удален из-за проблем с реагированием, как и сам класс `.text-hide`, поскольку его метод устарел и не должен использоваться. И классы `.text-*` больше не добавляют к ссылкам состояния наведения или фокуса, — эти классы `.text-*` следует заменить классами `.link-*`.

1.1.2. Наиболее важные дополнения

В версии 5 Bootstrap вносит в свою библиотеку множество интересных дополнений. Некоторые из них включают новые функции, новые поддерживаемые концепции, новые компоненты, новые служебные классы и многое другое.

Адаптивные размеры шрифтов теперь включены по умолчанию

Адаптивные размеры шрифтов (Responsive Font Size, RFS) присутствовали в предыдущих версиях Bootstrap, но по умолчанию были отключены. Bootstrap 5 по умолчанию включает RFS, что делает шрифты в Bootstrap более отзывчивыми.

RFS — это побочный проект, созданный Bootstrap для первоначального оперативного масштабирования и изменения размеров шрифтов. Теперь он способен делать то же самое с такими свойствами, как поля, отступы, тень блока и т. д. И если вы используете для Bootstrap файлы Sass, то можете задействовать параметры, созданные RFS, включая `font-size`, `margin`, `padding` и другие, — те, которые позволяют вам убедиться, что ваши компоненты и стили адаптивны.

Поддержка написания справа налево

Bootstrap 5 добавляет поддержку написания текста как «слева направо», так и «справа налево». Поскольку Bootstrap используется во всем мире — это большой и важный шаг: сделать такую поддержку доступной «из коробки». По этой причине Bootstrap 5 отказался от конкретных имен для направлений (например, использование `left` и `right`) в пользу `start` и `end`. Это делает Bootstrap достаточно гибким для работы с веб-сайтами с письмом в разных направлениях: LTR и RTL.

Новая точка останова

Bootstrap 5 обеспечивает поддержку новых XXL-устройств, нацеленных на точки останова, ширина которых больше или равна 1400 пикселов (px). Раньше самой высокой точкой останова была `xl`, которая предназначалась для устройств с шириной экрана больше или равной 1200 px. Это изменение позволяет улучшить реагирование на устройствах с большими экранами, поскольку точки останова `xl` стало недостаточно.

Использование новой точки останова аналогично применению всех других точек останова. Например, чтобы задействовать отступы для устройств, ширина которых больше или равна 1400 px, вы можете сделать это с помощью класса `.p-xxl-2`.

Темная и светлая тема

В версии Bootstrap 5.3 появилась новая функция — возможность создавать для проекта светлую и темную темы. Более не нужно писать кастомные CSS-стили для создания нескольких тем — всё настраивается через переменные.

В Bootstrap 5.3 светлая и темная темы включены по умолчанию — для них всё готово, и нужно лишь научиться их переключать. Это можно сделать с помощью HTML-атрибута `data-bs-theme`, который принимает одно из двух значений:

- `light` — светлая тема (используется по умолчанию);
- `dark` — темная тема.

Задать атрибут можно на любом теге HTML, но чаще всего его ставят на весь `<body>`, — это позволяет применить тему сразу для всего проекта.

Создание модальных око через атрибуты данных

В Bootstrap 5 через атрибуты данных можно активировать модальное окно, установив для элемента — например, кнопки, атрибут `data-bs-toggle="modal"`. Мы также можем переключить определенное модальное окно, используя `data-bs-target="#foo"` или `href="#foo"`. Примеры:

- `data-bs-toggle="modal"` — сообщает кнопке, что мы будем активировать модальное окно при нажатии на элемент;
- `data-bs-target="#id"` — сообщает кнопке идентификатор модального окна, которое необходимо активировать.

1.1.3. Новые компоненты

Аккордеон

Раньше Accordion был частью компонента `Collapsible`, а не отдельным компонентом. В Bootstrap 5 `Accordion` — новый компонент. Вместе с `Accordion` появился новый класс — `.accordion`, который содержит внутри себя список элементов аккордеона. Кроме того, у аккордеонов теперь новый стиль, отличный от предыдущего стиля в Bootstrap 4.

Offcanvas

Еще один новый компонент — `Offcanvas`, который позволяет создавать всплывающую панель, которая обычно организуется на веб-сайтах для отображения меню на небольших устройствах. Вы можете поместить такую панель на любой стороне страницы, применив атрибут `.offcanvas-start` слева в LTR, `.offcanvas-top` сверху, `.offcanvas-end` справа в LTR и `.offcanvas-bottom` снизу. Вы также можете указать, должен `Offcanvas` иметь фон или нет.

Плавающая этикетка

Плавающая этикетка (метка) — это новая составная часть компонентов формы. Она позволяет создать поле ввода, метка которого сначала выглядит как заполнитель поля, а затем, когда ввод получает фокус, метка перемещается в верхнюю часть ввода над значением. Этот компонент также работает с элементами `<select>` элементами и `<textarea>`.

1.1.4. Прочие улучшения и изменения

Помимо новых компонентов, добавлены новые классы для существующих компонентов, новые служебные классы, новые вспомогательные классы и многое другое. Например:

- добавлен класс `.row-cols-auto`, который позволяет столбцам принимать естественную ширину;
- с помощью новой утилиты «желоб» теперь можно устанавливать промежутки между строками и столбцами макета страницы. Для установки горизонтального желоба используется класс `.gx-*`, для установки вертикального — `.gy-*`. Чтобы установить желоб как по горизонтали, так и по вертикали, служит класс `.g-*`;
- добавлен новый ввод (выбор) цвета с использованием класса `.form-control-color`. Здесь пользователь может визуально подобрать нужный цвет из палитры RGB:

```
<label for="color" class="form-label">Color</label>
<input type="color" class="form-control form-control-color" id="color" val-
ue="#563d7c" title="Choose your color">
```

- добавлен новый тип ввода списков данных, который имитирует поле выбора с опережением ввода;
- добавлены новые классы для удобного центрирования абсолютных элементов по горизонтали (`.translate-middle-x`) и вертикали (`.translate-middle-y`);
- в раскрывающемся списке появилась новая опция автоматического закрытия, которая может изменить поведение по умолчанию при закрытии раскрывающегося меню. По умолчанию раскрывающиеся меню закрываются при щелчке за пределами раскрывающегося списка или одного из его элементов. Вы можете изменить это поведение, установив для атрибута данных `data-bs-auto-close` значение `inside`, при котором раскрывающийся список закрывается при щелчке на одном из элементов раскрывающегося списка, но не при щелчке за его пределами. Вы также можете установить для этого атрибута значение `outside`, при котором раскрывающийся список будет закрываться только при щелчке за пределами раскрывающегося списка. Наконец, вы можете настроить его как `false` — чтобы он закрывался только при повторном нажатии кнопки раскрывающегося списка. Если же вы инициализируете раскрывающийся список с помощью JavaScript, то можете использовать опцию `autoClose` вместо атрибута данных.

* * *

Bootstrap является одной из ведущих библиотек CSS на протяжении большей части последнего десятилетия. Это позволяет разработчикам легко создавать адаптивные веб-сайты и

обеспечивать удобство работы . Благодаря дополнениям и изменениям Bootstrap 5, этот процесс станет еще проще и позволит улучшить дизайн веб-сайтов и обеспечить для их посетителей дополнительные удобства.

Если вы используете Bootstrap 4 и подумываете о переходе на Bootstrap 5, миграция будет простой, и вам, вероятно, не потребуется вносить много изменений.

1.2. Получение файлов фреймворка Bootstrap

Есть несколько способов использовать фреймворк Bootstrap в своих приложениях:

- подключить свое приложение (сайт) к коду фреймворка через сеть доставки контента CDN (Content Delivery Networks);
- скачать готовые (скомпилированные) файлы и подключить их к своему приложению;
- скачать исходные файлы и скомпилировать (собрать) свой собственный пакет, который будет содержать только необходимые компоненты CSS и плагины JS.

Мы рассмотрим первые два варианта, как наиболее простые и доступные.

1.2.1. Использование сети CDN

Наиболее простой способ — подключение приложения (сайта) к фреймворку через сеть доставки контента (CDN). CDN — это географически распределенная сетевая инфраструктура, обеспечивающая быструю доставку контента пользователям веб-сервисов и сайтов. Входящие в состав CDN серверы географически располагаются таким образом, чтобы сделать время ответа для пользователей сайта (сервиса) минимальным. Для такого подключения Bootstrap, страницы сайта в заголовке должны содержать ссылку на CSS- и JS-файлы. Далее приведен код HTML-страницы, которая содержит эти ссылки на файлы Bootstrap 5.3.2:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Bootstrap 5</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
<!-- Подключение Bootstrap из интернета через CDN -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-
T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"
    crossorigin="anonymous">

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.
js"
    integrity="sha384-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
    crossorigin="anonymous"></script>
 </head>
<body>
</body>
/html>
```

Здесь в теге `<head>` заголовка сайта присутствует пять строчек. Первые три строки не имеют отношения в Bootstrap, но для начинающих изучать основы веб-программирования поясним их назначение:

- тег `<title>` служит для вывода информации, которая появится на самой верхней вкладке интернет-браузера. Далее следуют две строки с метатегами — разберемся, что это за теги `<meta>`, и для чего они нужны;

Тег <meta>

Тег <meta> (от англ. meta information, метаинформация) определяет данные, которые предназначены для браузеров и поисковых систем. Например, механизмы поисковых систем обращаются к метатегам для получения описания сайта, ключевых слов и других данных. Разрешается использовать более чем один метатег, все они размещаются в заголовке HTML-страницы — в теге <head>.

- метатег <meta charset="UTF-8"> определяет кодировку символов, которая будет принята на сайте. В приведенном коде этот метатег означает, что на сайте будет использоваться 8-битовый формат преобразования Unicode, который является стандартной кодировкой символов в последней версии HTML 5. Такая строка должна быть включена в каждую созданную веб-страницу, поскольку она обеспечивает правильное отображение символов любого языка в любом браузере. Указанная здесь кодировка UTF-8 гарантирует, что символы нелатинских языков искажаться не будут. Браузер Google Chrome автоматически установил кодировку UTF-8, поэтому вам не придется беспокоиться об этом при разработке дизайна для этого браузера. Но вам все равно нужно включать <meta charset="UTF-8"> в каждый файл HTML на случай, если эта функция не поддерживается другими браузерами;
- метатег <name="viewport"> определяет, как нужно настроить параметры окна браузера для конкретного устройства, — главным образом это актуально для мобильных смартфонов. В настоящее время важно, чтобы веб-страницы хорошо отображались на всех устройствах: на настольных компьютерах, ноутбуках, планшетах и — особенно — на мобильных смартфонах. Для этого и нужно включить метатег <name="viewport"> в каждый файл HTML. Этот тег определяет, как страницы сайта станут отображаться на экранах разного размера, и сколько визуальной области браузера будет доступно пользователю. Каждое устройство имеет разные размеры и разрешение экрана. Как правило, у мобильных устройств видимая область экрана меньше, а у настольных компьютеров — больше. Использование параметра content="width=device-width — первый шаг к тому, чтобы страницы сайтов хорошо выглядели на мобильных устройствах. Он не позволяет страницам сайта, который просматривается на мобильном устройстве, выглядеть так, как если бы он отображался на ноутбуке, — то есть сильно уменьшенным, с мелкими и плохо читаемыми символами. Это гарантирует, что HTML-страницы будут автоматически адаптированы под размеры экрана устройства. Параметр initial-scale=1.0 устанавливает начальный масштаб при первой загрузке страницы браузером — то есть разворачивает его на полный экран.

Затем следуют два тега: <link …> и <script …> — которые и обеспечивают подключение приложения к файлам CSS фреймворка Bootstrap через сеть доставки контента CDN:

- <link href="https://cdn... — подключение приложения к файлам CSS;
- <script src="https://cdn... — подключение приложения к файлам JS.

В дополнение к метатегам, приведенным в этом примере, на странице можно разместить еще четыре полезных метатега:

- метатег <X-UA-Compatible> — отвечает за браузеры Microsoft IE8, IE9, IE10, IE11. Он говорит о том, страница будет адаптирована к последней версии Internet Explorer. Всегда выбирайте последнюю версию Internet Explorer — то есть IE=edge. Вот пример строки с этим метатегом:

  ```
  <meta http-equiv="X-UA-Compatible"conent="IE=edge">
  ```

- метатег <meta name="description">. Наличие этого метатега помогает поисковым системам ранжировать ваш сайт по сравнению с другими сайтами. Он предназначен в основном для целей SEO (поисковой оптимизации). Метатег описания позволяет кратко и лаконично объяснить, о чем ваш сайт, и может выглядеть примерно так:

  ```
  <meta name="description" content="Разработка web приложений: Python, Django, Bootstrap, HTML. Программирование на Python.">
  ```

  При описании вашего сайта в этом метатеге необходимо учитывать следующие требования:

  - описание должно быть коротким, не превышающим 160 символов;
  - в описании должны присутствовать ключевые слова, по которым люди ищут услуги, предоставляемые вашим сайтом;
  - оно должно четко указывать на ваши преимущества перед конкурентами;

- метатег `<meta name="keywords"` — служит для задания ключевых слов, которые используются поисковыми серверами для ранжирования вашего сайта в списке других сайтов. Далее приведен пример задания набора ключевых слов:

  ```
  <meta name="keywords" content="Python, Django, Bootstrap, web приложение">
  ```

- метатег `<meta name="author">` — ще один полезный метатег, который следует включить в описание сайта, — это имя автора:

  ```
  <meta name="author" content="Семенов Алексей">
  ```

  Эта информация показывает, кто является владельцем веб-сайта или кому принадлежат авторские права.

1.2.2. Скачивание файлов Bootstrap

Второй путь использования фреймворка Bootstrap — скачать готовые файлы и подключить их к своему приложению. Этот путь тоже не представляет особых трудностей. Скачать готовые файлы можно с официального сайта фреймворка[1].

После скачивания вы получите архивированный zip-файл, в имени которого будет отражена текущая версия фреймворка, — например, bootstrap-5.3.2-dist.zip. После распаковки архива на вашем компьютере будут созданы две папки: css — с файлами стилей, и js — с Java-скриптами. Эти две папки нужно перенести в ваш проект.

1.3. Контейнеры и сетка Bootstrap

Сначала разберемся с тем, что такое адаптивный макет сайта. *Адаптивный макет* — это такой макет, вид которого может изменяться в зависимости от того, какую ширину (`viewport`) имеет браузер. Это означает, что при одних значениях ширины `viewport` адаптивный макет может выглядеть одним образом, а при иных — совершенно по-другому.

В Bootstrap изменение вида макета реализовано через медиа-запросы. Каждый медиа-запрос в Bootstrap строится на основании минимальной ширины окна браузера. В медиа-запросе Bootstrap ключевое значение ширины `viewport` называется `breakpoint` (контрольной точкой, или классом устройства). На рис. 1.1 приведены основные контрольные точки, которые в Bootstrap 5 назначены по умолчанию.

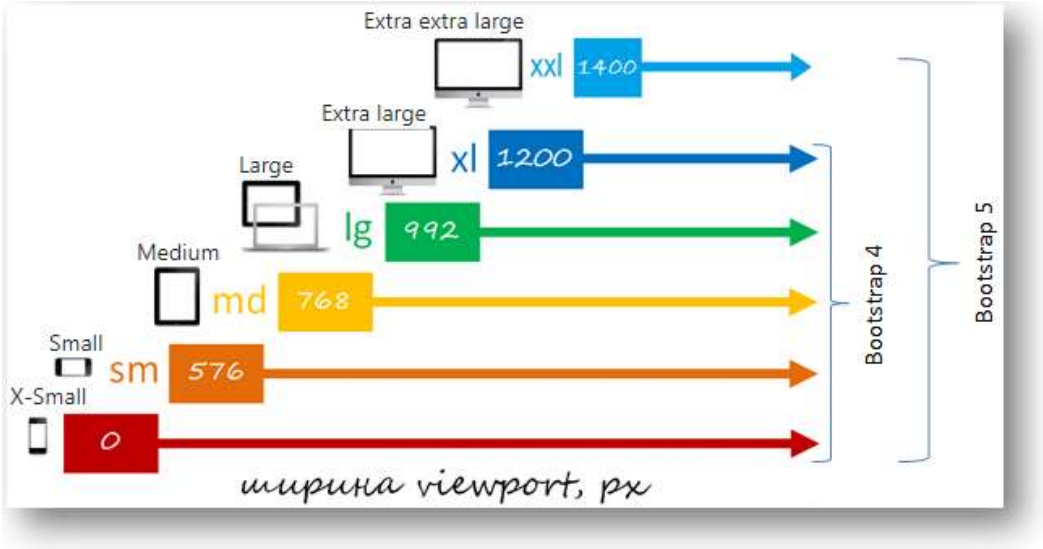


Рис. 1.1. Контрольные точки для Bootstrap4 и Bootstrap5

Цифры на этом рисунке означают, что при ширине видимой области браузера до 576 px макет сайта будет отображаться одним образом, от 576 px до 768 px — другим, и т. д. Иными словами, можно создать макет, который на каждом из этих участков видимой области браузера станет выглядеть по-разному.

---

[1] См. **https://getbootstrap.com/docs/5.2/getting-started/download/**.

Контрольные точки (breakpoint) обозначаются следующим образом. Первая контрольная точка не имеет обозначения (т. е. xs не указывается), вторая обозначается sm, третья — md, четвертая — lg, пятая — xl и шестая — xxl. Эти обозначения необходимо запомнить, поскольку они используются в классах, которые мы будем добавлять к элементам. Такие обозначения в имени класса будут указывать на то, с какой ширины viewport стили, определенные в нем, станут применяться к вложенным элементам. При этом контрольные точки задают только минимальную ширину. Например, если вы определили макет, указав в нем классы без обозначения контрольной точки (xs) и с использованием параметра md, то он на устройстве sm страница будет иметь такую же структуру, как и на устройстве xs, а на устройствах с большим экраном lg, xl и xll — как на md. Пример использования контрольных точек при макетировании страниц сайта будет приведен немного позже, после разделов с описанием сетки HTML-страниц.

Теперь разберемся с сеткой HTML-страницы в Bootstrap — она состоит из следующих элементов:

- оберточных контейнеров (элементов с классом container, container-fluid или container-{breakpoint});
- рядов или строк (элементов с классом row);
- адаптивных колонок или блоков, имеющих один или несколько классов col.

Все части сетки — это обычные элементы HTML, к которым просто добавлены определенные классы.

1.3.1. Адаптивные контейнеры

Адаптивный (или оберточный) контейнер — это элемент сетки Bootstrap, с которого обычно начинается создание макета страницы или ее части. Другими словами, это базовый элемент, в котором необходимо размещать все другие элементы сетки (ряды и адаптивные блоки). Его основная цель — установить шаблону ширину и выровнять его по центру страницы (рис. 1.2).

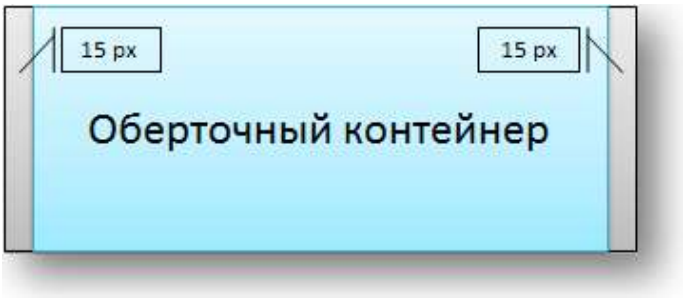


Рис. 1.2. Оберточный контейнер

Контейнеры в Bootstrap являются основным элементом макетирования страниц — они служат для вкладывания в них других элементов интерфейса, формирования отступов и выравнивания или центрирования их содержимого. Хотя контейнеры могут быть вложенными, для большинства макетов вложенный контейнер не требуется.

Bootstrap поставляется с двумя базовыми контейнерами:

- адаптивно-фиксированный контейнер (класс container) — устанавливает максимальную ширину (max-width) для каждой контрольной точки;
- адаптивно-подвижный («резиновый») контейнер (класс container-fluid), который устанавливает максимальную ширину (width: 100%) во всех контрольных точках.

Для адаптивно-фиксированного контейнера можно указать контрольную точку (container-{breakpoint}), при этом будет установлена максимальная ширина (width: 100%) до указанной контрольной точки.

Адаптивно-фиксированный контейнер

Адаптивно-фиксированный контейнер предназначен для создания контейнера с постоянной шириной, но которая будет оставаться постоянной только в пределах действия определенной контрольной точки. Например, на контрольной точке sm до действия контрольной точ-

ки md он будет иметь одну фиксированную ширину, а на md до действия lg — другую фиксированную ширину. Единственная контрольная точка, на которой этот контейнер не будет иметь фиксированную ширину — это breakpoint без обозначения. Здесь контейнер будет занимать 100%-ную ширину видимой области браузера. Пример расположения адаптивно-фиксированного контейнера в окне браузера представлен на рис. 1.3.



Рис. 1.3. Расположения адаптивно-фиксированного контейнера в окне браузера

Адаптивно-фиксированный контейнер будет иметь следующие значения ширины:

- 100%-ную ширину при ширине viewport до 576 px;
- 540 px при ширине viewport от 576 до 768 px;
- 720 px при ширине viewport от 768 до 992 px и т. д.

В горизонтальном направлении контейнер располагается по центру окна браузера. Определяется это посредством установки ему CSS-свойств margin-left:auto и margin-right:auto в файле bootstrap.css. Создать адаптивно-фиксированный контейнер можно так:

```
<div class="container">...</div>
```

Адаптивно-подвижный, или «резиновый» контейнер

Второй вид контейнера — это *адаптивно-подвижный контейнер*, или «резиновый». Он применяется тогда, когда необходимо создать полностью гибкий макет целой страницы или ее части. Этот контейнер на любых контрольных точках занимает полную (100%-ную) ширину видимой области браузера (рис. 1.4).

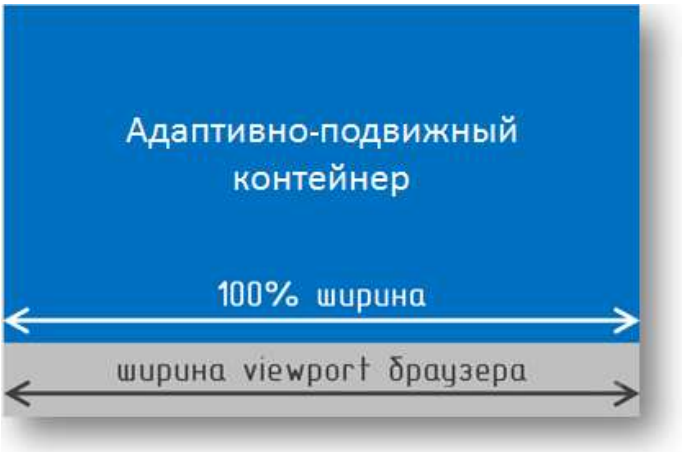


Рис. 1.4. Расположения адаптивно-подвижного контейнера в окне браузера

Создать адаптивно-подвижный контейнер можно так:

```
<div class="container-fluid">...</div>
```

* * *

Контейнеры класса container и container-fluid имеют внутренние отступы слева и справа по 15 px. Установка внутренних отступов контейнеров задается в файле bootstrap.css

посредством CSS-свойств `padding-left:15px` и `padding-right:15px`. При верстке макета страницы не следует одни оберточные контейнеры помещать внутрь других.

1.3.2. Ряды, или строки (*row*)

Ряд, или *строка* — это элемент сетки страницы, который выступает в роли контейнера для адаптивных блоков (колонок). Он представляет собой дочерний элемент контейнера и родительский элемент для адаптивных блоков (рис. 1.5).



Рис. 1.5. Расположения ряда (строки) в теле адаптивно-фиксированного контейнера

Для создния ряда используется следующий код:

```
<div class="row">...</div>
```

В Bootstrap ряды играют очень важную роль. Это связано с тем, что сетка HTML-страниц построена на CSS Flexbox — адаптивных блоках. В этой сетке ряд выступает в роли flex-контейнера для flex-элементов (адаптивных блоков). Если адаптивные блоки окажутся вне ряда, то они работать не будут, поэтому в Bootstrap адаптивные блоки должны обязательно находиться в блоке с классом `row`. Компенсация внутренних отступов (`padding`) осуществляется за счет отрицательных левых и правых внешних отступов, равных 15 px (`margin-left:-15px` и `margin-right:-15px`).

1.3.3. Адаптивные блоки (*col*)

Адаптивные блоки — это основные строительные элементы сетки. Именно от них зависит то, как макет веб-страницы будет отображаться в видимой области браузера на разных контрольных точках. Адаптивные блоки располагаются внутри ряда — то есть ряд является контейнером для адаптивных блоков (рис. 1.6).

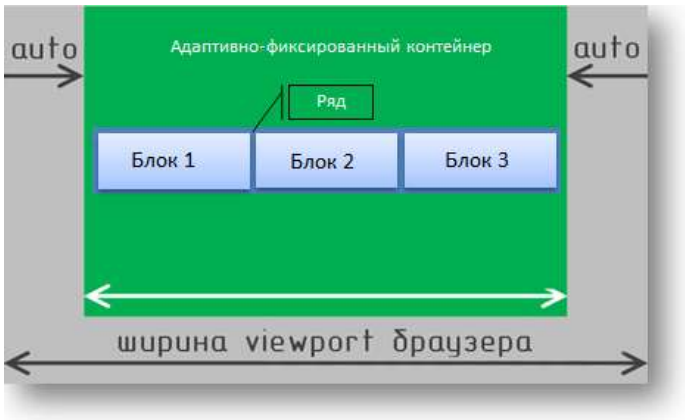


Рис. 1.6. Расположение адаптивных блоков в контейнере ряд (в строке)

Ширина адаптивных блоков на разных контрольных точках (breakpoint) может быть различной (в процентном отношении от родительского контейнера). Например, адаптивный блок в браузере смартфона (`sm`) может иметь ширину, равную 50% от родительского элемента, а на планшете (`md`) — 25%.

Адаптивный блок достаточно просто создать посредством добавления одного или нескольких классов `col-?-?` к HTML-элементу, расположенному в ряду. В классе `col-?-?` вместо первого знака вопроса указывается название контрольной точки («пусто», `sm`, `md`, `lg`, `xl` или `xxl`). Вместо второго знака вопроса задается ширина адаптивного блока, которую он должен иметь на указанной контрольной точке. Ширина адаптивного блока назначается в относительной форме либо по умолчанию, либо с помощью числа от 1 до 12 (по количеству колонок).

Число, которое стоит при инициализации класса `col`, определяет, какую часть ширины родительского контейнера (т. е. ряда) будет занимать адаптивный блок, начиная с указанной контрольной точки. При этом ширина ряда в числовом выражении (в колонках Bootstrap) по умолчанию равна 12. Например, блок с классом `col-md-4`, начиная с контрольной точки `md`, будет занимать $^{4}/_{12}$ ширины ряда, т. е. 4/12·100% = 33,3%.

Адаптивные блоки, как и оберточные контейнеры, имеют слева и справа внутренние отступы по 15 px. Эти отступы адаптивным блокам в фреймворке Bootstrap устанавливаются с помощью CSS-свойств `padding-left:15px` и `padding-right:15px`. Размещать адаптивные блоки необходимо в ряду (в строке) — то есть у любого адаптивного блока в качестве родителя должен быть обязательно элемент `row`.

Рассмотрим на примерах, какую ширину будут иметь следующие адаптивные блоки.

1. `class= "col-12"` задает ширину блока по умолчанию. Ширина блока будет равна 12 колонкам Bootstrap (т. е. 12/12·100% = 100% от ширины ряда), эту ширину блок получит, начиная с контрольной точки `xs`.
2. `class="col-sm-9"` задает ширину блока с контрольной точки `sm`. Ширина блока будет равна 9 колонкам Bootstrap (т. е. 9/12·100% = 75% от ширины ряда), эту ширину блок получит, начиная с контрольной точки `sm`.
3. `class="col-md-7"` задает ширину блока с контрольной точки `md`. Ширина блока будет равна 7 колонкам Bootstrap (т. е. 7/12·100% = 58,3% от ширины ряда), начиная с контрольной точки `md`.
4. `class="col-lg-5"` задает ширину блока с контрольной точки `lg`. Ширина блока будет равна 5 колонкам Bootstrap (т. е. 5/12·100% = 41,6% от ширины ряда), начиная с контрольной точки `lg`.
5. `class="col-xl-3"` задает ширину блока с контрольной точки `xl`. Ширина блока будет равна 3 колонкам Bootstrap (т. е. 3/12·100% = 25% от ширины ряда), начиная с контрольной точки `xl`.
6. `class="col-xxl-6"` задает ширину блока с контрольной точки больше `xxl`. Ширина блока будет равна 2 колонкам Bootstrap (т. е. 6/12·100% = 50% от ширины ряда).

При задании ширины адаптивному блоку мы указываем класс, содержащий контрольную точку, начиная с которой эта ширина станет действовать. Такую ширину этот блок будет иметь до тех пор, пока она не окажется переопределена с помощью другого класса, действие которого начинается с наибольшей ширины `viewport`.

1.3.4. Адаптивные блоки без указания числа колонок

В сетке Bootstrap имеются специальные классы: `col`, `col-sm`, `col-md`, `col-lg`, `col-xl`, `col-xxl` `col-auto`, `col-sm-auto`, `col-md-auto`, `col-lg-auto`, `col-xl-auto` и `col-xxl-auto`.

Первая группа классов (`col`, `col-sm`, `col-md`, `col-lg`, `col-xl`, `col-xxl`) предназначена для создания адаптивных блоков, ширина которых будет зависеть от свободного пространства в ряду (строке). Незанятая ширина (свободное пространство) в ряду между всеми такими блоками распределяется равномерно. Кроме того, эти адаптивные блоки перед распределением свободного пространства ряда (по умолчанию) имеют нулевую ширину.

1.3.5. Расположение адаптивных блоков

Адаптивные блоки по умолчанию располагаются в ряду горизонтальными линиями, при этом они выстраиваются последовательно слева направо. В одну горизонтальную линию могут поместиться адаптивные блоки с суммарным числом колонок не более 12. Адаптивные блоки, которые не помещаются в текущую строку, переходят на следующую строку.

Колонки можно объединять в адаптивные блоки произвольным образом. Пример различных вариантов расположения колонок в адаптивных блоках в ряду приведен на рис. 1.7.

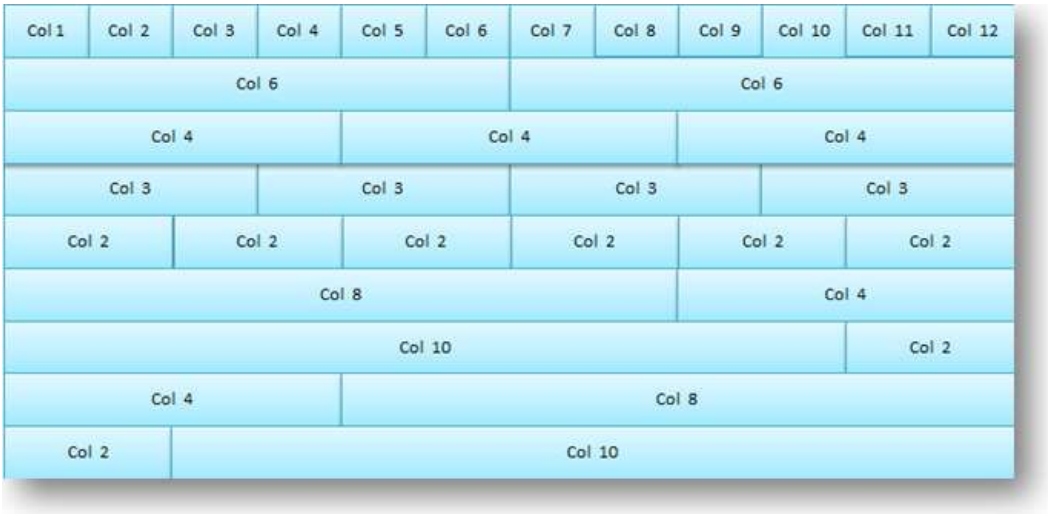


Рис. 1.7. Возможные варианты формирования адаптивных блоков из 12 колонок

1.4. Верстка макета HTML-страниц

Основной принцип верстки макета веб-страницы на сетке Bootstrap заключается во вкладывании адаптивных блоков в ряды, а рядов — в контейнеры. При этом ширина адаптивных блоков — это всегда относительная величина, которая задается в колонках Bootstrap и зависит только от ширины родителя, — то есть от ряда. Размещать контент веб-страницы следует только в адаптивных блоках. Вот пример формирования макета страницы:

```
<div class="container">
    <div class="row">
        <div class=col-8>Блок 1</div>
        <div class=col-4>Блок 2</div>
    </div>
</div>
```

В этом примере создан контейнер (`class="container"`), в него вложена строка (`class="row"`), и уже в эту строку вложены два адаптивных блока: шириной 8 колонок (`class=col-8`) и шириной 4 колонки (`class=col-4`).

Если предполагается, что блоки будут одинаковой ширины, то число колонок для них можно не указывать, например:

```
<div class="container">
    <div class="row">
        <div class="col"> Блок 1 </div>
        <div class="col"> Блок 2 </div>
        <div class="col"> Блок 3 </div>
        <div class="col"> Блок 4 </div>
        <div class="col"> Блок 5 </div>
    </div>
</div>
```

1.4.1. Выравнивание адаптивных блоков

Выравнивание адаптивных блоков в горизонтальном и вертикальном направлении осуществляется с помощью служебных flex-классов.

В пределах ряда по горизонтали адаптивные блоки выравнивают посредством одного из следующих классов, который необходимо дополнительно добавить к `row`:

- `align-items-start` — выравнивание относительно начала ряда (по левому краю);
- `align-items-center` – по центру ряда;

- `align-items-end` — относительно конца ряда (по правому краю).

Например, в следующем коде адаптивные блоки выровнены по центру ряда:

```
<div class="row align-items-center">
    <div class="col">Блок 1</div>
    <div class="col">Блок 2 </div>
</div>
```

Вертикальное выравнивание адаптивных блоков

По умолчанию адаптивные блоки занимают всю высоту ряда, в котором они расположены. Выравнивание какого-то определенного адаптивного блока по вертикали в пределах ряда может осуществляться одним из следующих классов:

- `align-self-start` — выравнивание относительно начала ряда (по верху ряда);
- `align-self-center` — по центру ряда;
- `align-self-end` — относительно конца ряда (по низу ряда).

Эти классы необходимо добавлять к адаптивным блокам, а не к ряду. Например, так мы выравниваем адаптивный блок 2 по нижнему краю ряда:

```
<div class="row align-items-center">
    <div class="col">Блок 1</div>
    <div class="col align-self-end">Блок 2</div>
</div>
```

Горизонтальное выравнивание адаптивных блоков

Для выравнивания адаптивных блоков в горизонтальном направлении можно также использовать следующие классы:

- `justify-content-start` — относительно начала ряда (по левому краю), установлено по умолчанию;
- `justify-content-center` — по центру ряда;
- `justify-content-end` — относительно конца ряда (по правому краю);
- `justify-content-around` — равномерно, с учетом пространства перед первым и последним адаптивными блокамм;
- `justify-content-between` — равномерно, с одинаковым пространством между адаптивными блоками.

Например, распределим адаптивные блоки в горизонтальном направлении равномерно:

```
<div class="row justify-content-around">
    <div class="col-4">Блок 1</div>
    <div class="col-4">Блок 2</div>
</div>
```

1.4.2. Смещение адаптивных блоков

Смещение адаптивных блоков в Bootstrap можно выполнить с помощью следующих классов:

- классов `offset` — на определенное число колонок;
- служебных (утилитных) классов `margin`.

Смещение с помощью классов *offset*

Классы `offset` предназначены для смещения адаптивных блоков вправо на определенное число колонок. Эти классы имеют следующий синтаксис:

- `offset-{1}`;
- `offset-{breakpoint}-{1}`.

Здесь `{breakpoint}` — контрольная точка, начиная с которой к блоку будет применено смещение. Если она не указана, то смещение применяется, начиная с самых крохотных

устройств (смартфоны). Второй параметр `{1}` — это величина смещения (число колонок). Рассмотрим следующий код:

```
<div class="row">
    <div class="col-4">Блок 1</div>
    <div class="col-4 offset-4">Блок 1</div>
</div>
<div class="row">
    <div class="col-3 offset-3">Блок 3 </div>
    <div class="col-3 offset-3">Блок 4</div>
</div>
<div class="row">
    <div class="col-6 offset-3">Блок 5</div>
</div>
```

В этом случае смещение адаптивных блоков будет выглядеть так, как показано на рис. 1.8.

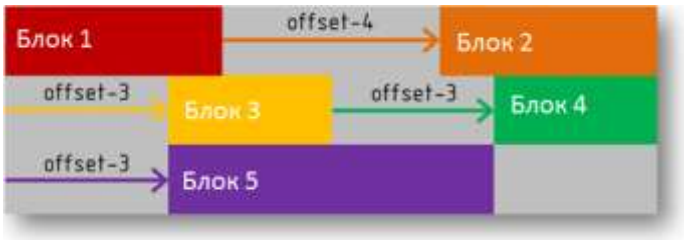


Рис. 1.8. Пример смещения адаптивных блоков с использованием класса `offset`

Смещение с помощью классов *margin*

В Bootstrap устанавливать смещение адаптивным блокам также можно с помощью отступов `margin` (`margin-left:auto` и `margin-right auto`). Этот вариант смещения появился благодаря тому, что сетка в новых версиях Bootstrap основывается на CSS Flexbox. Такой вариант удобен, когда блоки необходимо сместить относительно друг друга на некоторую переменную величину. Для более удобного задания блокам отступов `margin` можно указать сокращенное обозначение классов: `ml-auto`, `mr-auto`, `ml-{breakpoint}-auto` и `mr-{breakpoint}-auto`. Рассмотрим следующий код:

```
<div class="row">
    <div class="col-4">Блок 1</div>
    <div class="col-4 ml-auto">Блок 2</div>
</div>
<div class="row">
    <div class="col-3">Блок 3</div>
    <div class="col-3 ml-auto mr-auto">Блок 4</div>
    <div class="col-3">Блок 5</div>
</div>
<div class="row">
    <div class="col-4 ml-auto mr-auto">Блок 6</div>
    <div class="col-4 ml-auto mr-auto">Блок 7</div>
</div>
```

В этом случае смещение адаптивных блоков будет выглядеть так, как показано на рис. 1.9.

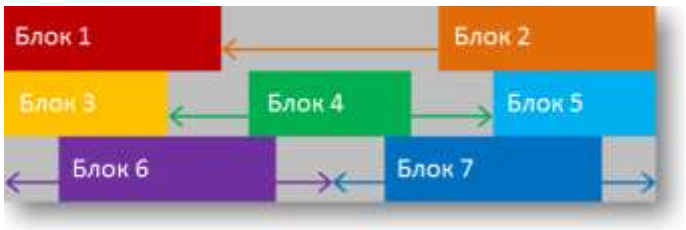


Рис. 1.9. Пример смещения адаптивных блоков с использованием класса `margin`

* * *

Итак, мы познакомились с основными «строительными блоками», из которых создаются макеты HTML-страниц. Рассмотренные нами адаптивные блоки являются конечными элементами макета, в которые и размещается основной контент страницы.

Разберемся с использованием контрольных точек, рядов и колонок на простом примере и создадим для этого страницу с двумя панелями: левой боковой панелью (на которой может располагаться боковое меню), и центральной панелью с основным контентом. Назовем эту страницу boot_point.html (листинг 1.1).

Листинг 1.1. Программный код страницы boot_point.html

```
{% extends "shop/base.html" %}

<div class="container-fluid text-center">
    {% block header %}
        <div class="row text-center">
            <h4>Контрольные точки в Bootstrap 5</h4>
        </div>
    {% endblock header %}

    <div class="row text-center">
        <!-- Это начало боковой панели -->
        <div class="col-12 col-sm-6 col-md-6 col-lg-4 col-xl-3 col-xxl-2">
            {% block sidebar %}
                <h5>Боковая панель</h5>
                 <div class="row my-2 text-white">
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 1</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 2</div>
                </div>
            {% endblock sidebar %}
        </div>
        <!-- Конец боковой панели -->

        <!-- Это начало центральной панели -->
        <div class="col-12 col-sm-6 col-md-6 col-lg-8 col-xl-7 col-xxl-10"
             style="background-color:rgb(224, 224, 224);">
            {% block content%}
                <h5>Центральная панель</h5>
                <div class="row my-2 text-white">
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 3</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 4</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 5</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 6</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 7</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 8</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 9</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 10</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 11</div>
                    <div class="col mx-2 my-2 bg-primary text-center">Блок 12</div>
                </div>
            {% endblock content %}
        </div>
        <!-- Конец центральной панели -->
    </div>
</div>
```

Рассмотрим структуру этого программного кода. Здесь создан «резиновый» контейнер (`container-fluid`) и в него вложена строка (`row`). Эта строка с использованием контрольных точек разбита на два блока, или на две панели:

- левую боковую панель с адаптивной шириной:

```
<div class="col-12 col-sm-6 col-md-6 col-lg-4 col-xl-3 col-xxl-2">
```

- центральную панель с адаптивной шириной:

```
<div class="col-12 col-sm-6 col-md-6 col-lg-8 col-xl-7 col-xxl-10"
```

При минимальном размере экрана обе панели будут занимать все 12 колонок сетки, так как для них заданы контрольные точки `col-12` (то есть одна панель будет располагаться под другой). При ширине экрана в контрольных точках `sm` и `md` обе панели будут занимать по 6 колонок сетки и располагаться рядом друг с другом (для обеих панелей заданы контрольные точки `col-sm-6` и `col-md-6`). А вот при больших разрешениях экрана левая боковая панель будет занимать либо 4, либо 3, либо 2 колонки сетки (`col-lg-4 col-xl-3 col-xxl-2`). При этом центральная панель будет занимать либо 8, либо 7, либо 10 колонок сетки (`col-lg-8 col-xl-7 col-xxl-10`).

В левую боковую панель вложено 2 адаптивных блока (`col`), а в центральную панель 10 адаптивных блоков (`col`). То есть адаптивные блоки будут занимать весь ряд, но в зависимости от разрешения экрана их расположение и размеры будут меняться.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница boot_point.html**.

Если открыть эту страницу (boot_point.html) в рамках интерактивной цифровой платформы и менять ширину окна браузера (имитировать устройства с разным размером и разрешением экрана), то мы получим следующие результаты (рис. 1.10).



Рис. 1.10. Адаптация HTML-страницы к разным разрешениям экрана

Как можно здесь видеть, при любом размере и разрешении экрана в левой боковой панели присутствуют два адаптивных блока, а на центральной панели — 10 адаптивных блоков. По мере уменьшения ширины экрана размеры и положение встроенных в панели адаптивных блоков меняется, при этом информация в каждом блоке отображается корректно и без искажений. Такая страница будет хорошо читаться как на экране компьютера, так и на экране смартфона.

1.5. Простейшая HTML-страница с подключением фреймворка Bootstrap

Чтобы процесс создания макетов страниц был более понятен, рассмотрим несколько простых примеров. В этих примерах к шаблонам страниц подключаются локальные файлы Bootstrap — то есть в текущем проекте была создана папка static и в нее перенесены две папки со скачанными файлами Bootstrap: css и js.

Для начала создадим контейнер фиксированной ширины с одним рядом (строкой), в котором находится один адаптивный блок из 12 колонок, — то есть он будет занимать всю ширину ряда (листинг 1.2, страница start_1.html).

Листинг 1.2. Программный код страницы start_1.html

```
{% extends "shop/base.html" %}
{% block header %}
        <div class="row text-center">
            <h4>Простейшая страница в Bootstrap 5</h4>
        </div>
{% endblock header %}

{% block content%}
<div class="container">
    <div class="row">
        <div class="col">
            <h1>Это страница Bootstrap</h1>
            <p>На странице создан контейнер фиксированной ширины class="container."</p>
            <p>В контейнер вложена одна строка class="row".</p>
            <p>В строке находится один адаптивный блок class="col".</p>
            <p>С изменением размера окна браузера ширина элементов контейнера будет
                меняться в разных контрольных точках.</p>
        </div>
    </div>
</div>
{% endblock content %}
```

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница start_1.html**.

При минимальном размере окна браузера эта страница будет выглядеть, как показано на рис. 1.11.



Рис. 1.11. Простейшая HTML-страница, созданная на основе Bootstrap

Как здесь можно видеть, с параметрами по умолчанию текст на странице показан черным цветом, при этом отсутствуют какие-либо признаки, по которым можно определить размер и положение адаптивного блока, — например, рамки или выделение блока цветом. Для того чтобы наглядно продемонстрировать возможности макетирования страниц, необходимо явно показать границы блока. Это можно сделать либо с помощью цветного фона, либо созданием ограничительной рамки.

1.6. Задание цвета элементам HTML-страниц

Для задания цвета в Bootstrap предусмотрено несколько полезных классов. Класс `bg` позволяет задать 9 цветов для фона:

- `class="bg-primary"`;
- `class="bg-secondary"`;
- `class="bg-success"`;
- `class="bg-danger"`;
- `class="bg-warning"`;
- `class="bg-info"`;
- `class="bg-light"`;

- `class="bg-dark"`;
- `class="bg-white"`.

Пример кода HTML-страницы, в которой показано использование класса `bg` для задания цвета фону, приведен в листинге 1.3 (страница color_bg.html).

Листинг 1.3. Страница color_bg.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница color_bg.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 1.12.

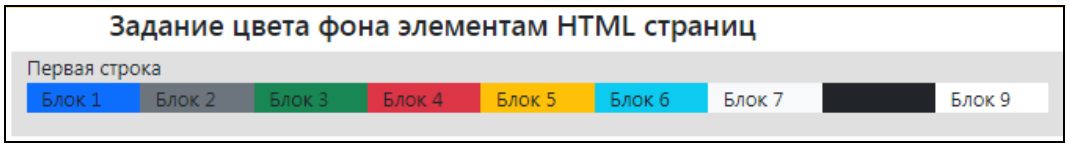


Рис. 1.12. Образцы цвета фона в Bootstrap

Текст **Блок 8** здесь не виден, так как и шрифт, и фон имеют одинаковый (черный) цвет. Для изменения цвета текста служит класс `text`, который позволяет задать 10 цветов:

- `class="text-primary"`;
- `class="text-secondary"`;
- `class="text-success"`;
- `class="text-danger"`;
- `class="text-warning"`;
- `class="text-info"`;
- `class="text-light"`;
- `class="text-dark"`;
- `class="text-muted"`;
- `class="text-white"`.

Пример кода HTML-страницы, в которой показано использование класса `text` для задания цвета шрифту, приведен в листинге 1.4 (страница color-text.html).

Листинг 1.4. Страница color-text.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница color-text.html**.

В окне браузера эта страница будет будет выглядеть так, как показано на рис. 1.13, — текст **Блок 7** и **Блок 10** здесь почти не виден, поскольку и шрифт, и фон имеют похожий цвет.

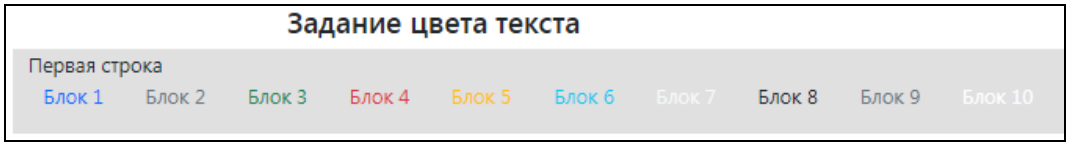


Рис. 1.13. Образцы цвета текста в Bootstrap

Для того чтобы текст элемента всегда был виден, он должен иметь цвет, отличный от цвета фона. Пример HTML-кода страницы, в которой показано использование двух классов: `text` и `bg` — для одновременного задания цвета для шрифта и для фона, приведен в листинге 1.5 (страница color_text_bg.html).

Листинг 1.5. Страница color_text_bg.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница color_text_bg.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 1.14. Теперь текст во всех блоках стал видимым.

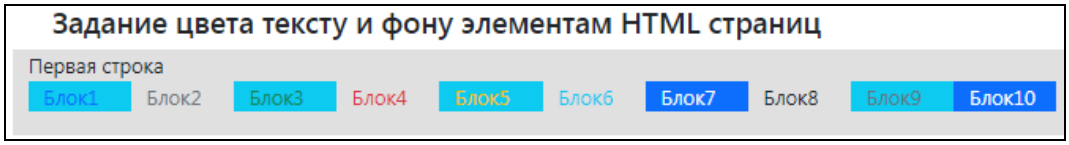


Рис. 1.14. Использование классов `text` и `bg` для задания цвета тексту и фону

1.7. Задание отступов элементам макета HTML-страниц

По умолчанию все адаптивные блоки «прилипают» друг к другу, однако достаточно часто требуется отделить их друг от друга дополнительным пространством. Особенно это актуально в тех случаях, когда они имеют рамки обрамления различных типов.

Для задания отступов (разрывов) между элементами HTML-страниц и их содержимым в Bootstrap предусмотрены два элемента, прдставляющие собой уникальные свойства CSS: `margin` (имеет сокращение `m`) и `padding` (имеет сокращение `p`). Свойство `margin` добавляет отступы *за пределами элемента*, а `padding` — *внутри элемента* (например, контейнера). Синтаксис для задания отступов:

```
{property}{sides}-{size}
```

Здесь приняты следующие обозначения:

- `property` — свойство;
- `sides` — сторона;
- `size` — размер.

Для свойства (`property`) в коде программ допустимы следующие сокращения:

- `m` — для классов `margin`, которые устанавливают отступы *за пределами элемента*;
- `p` — для классов `padding`, которые устанавливают отступы *внутри элемента*.

Параметр сторона (`sides`) может принимать сокращенные значения, которые устанавливают следующие отступы:

- `t` — отступ в верхней части элемента (`margin-top` или `padding-top`);
- `b` — в нижней части элемента (`margin-bottom` или `padding-bottom`);
- `s` — в стартовой части элемента (`start`), т. е. слева (`margin-left` или `padding-left`);
- `e` — в конечной части элемента (end), т. е. справа (`margin-right` или `padding-right`);
- `x` — с двух сторон элемента по горизонтали (`*-left` и `*-right`);
- `y` — с двух сторон элемента по вертикали (`*-top` и `*-bottom`);
- `blank` — отступ со всех четырех сторон элемента.

Параметр размер (`size`) может принимать следующие значения:

- `0` — нет отступов;
- `1` — отступ, равный `$spacer * .25`;
- `2` — отступ, равный `$spacer * .5`;
- `3` — отступ, равный `$spacer`;
- `4` — отступ, равный `$spacer * 1.5`;
- `5` — отступ, равный `$spacer * 3`;
- auto — отступ, равный авто.

> ***Пояснение***
> Здесь `$spacer = 1rem`. `Rem` — это единица типографики, равная корневому (базовому) значению размера шрифта (`font-size`). Это значит, что `1rem` всегда будет равен значению `font-size`, которое определено в HTML. Для большинства браузеров размер шрифта по умолчанию 16 px до тех пор, пока его не поменяют в настройках браузера (редко кто подобным занимается). Поэтому, как правило, `$spacer = 1rem = 16 px`.

Пример HTML-кода страницы, в которой показано использование отступов для разделения адаптивных блоков, приведен в листинге 1.6 (страница space_1.html).

Листинг 1.6. Страница space_1.html

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница space_1.html**.

Здесь для адаптивных блоков вставлены горизонтальные отступы в две позиции (`mx-2`). В окне браузера эта страница будет выглядеть так, как показано на рис. 1.15.

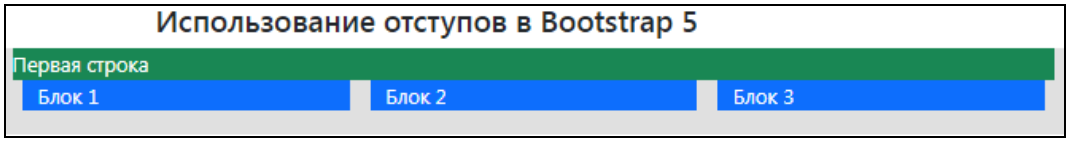


Рис. 1.15. Использование горизонтальных отступов между адаптивными блоками

Как здесь можно видеть, между адаптивными блоками появились горизонтальные отступы, однако между рядами (строками) нет разрывов и они вплотную примыкают друг к другу. Исправим это и вставим отступы между рядами (листинг 1.7, страница space_2.html).

Листинг 1.7. Страница space_2.html

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница space_2.html**.

Здесь для второго ряда вставлен вертикальный отступ в две позиции (`my-2`). В окне браузера эта страница будет выглядеть так, как показано на рис. 1.16, — между рядами появился вертикальный отступ.

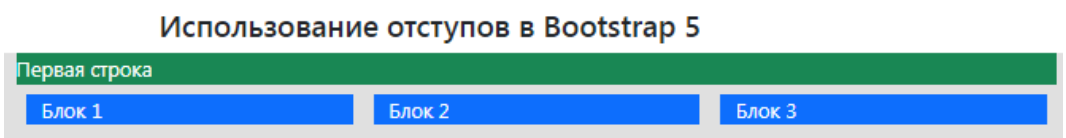


Рис. 1.16. Использование вертикальных отступов между рядами

Теперь посмотрим, как можно создать отступы внутри элемента. Пример HTML-кода страницы, в которой показано использование отступов в теле адаптивных блоков, приведен в листинге 1.8 (страница space_3.html).

Листинг 1.8. Страница space_3.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница space_3.html**.

Здесь для адаптивных блоков вставлены горизонтальные отступы в две позиции (`mx-2`) и внутренний вертикальный отступ (между текстом и внешней границей) в пять позиций (`py-5`). В окне браузера эта страница будет выглядеть так, как показано на рис. 1.17, — между текстом внутри адаптивных блоков и их внешней границей появились отступы.

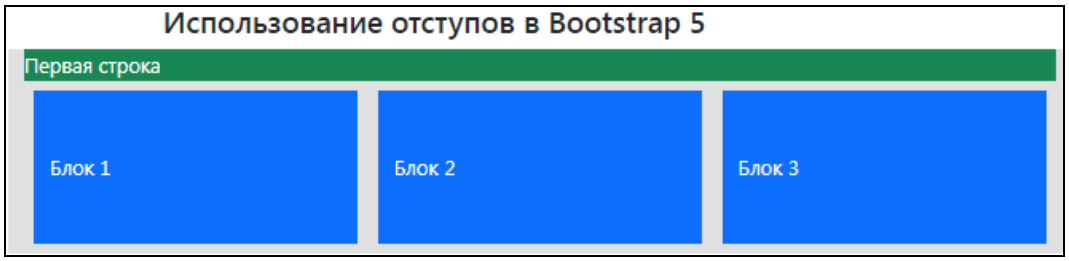


Рис. 1.17. Использование внутренних отступов в теле адаптивных блоков

Обратите внимание, что на всех этих HTML-страницах текст внутри адаптивных блоков по умолчанию прижимается к левому краю, — то есть к началу блока. Разберемся теперь, как можно выравнивать содержимое блоков HTML-страниц.

1.8. Выравнивание содержимого в адаптивных блоках HTML-страниц

Начнем с самого простого — выравнивания текста по горизонтальной оси (*x*). Для этого нужно использовать класс `text`, который обеспечивает следующие варианты выравнивания:

- `start` (умолчание браузера) — по левой стороне;
- `end` — по правой стороне;
- `center` — по центру.

В программном коде горизонтальное выравнивание текста реализуется так:

- `class="text-start"`;
- `class="text-center"`;
- `class="text-end"`.

Пример HTML-кода страницы, в которой показано выравнивание текста в теле адаптивных блоков, приведен в листинге 1.9 (страница alignment_1.html).

Листинг 1.9. Страница alignment_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница alignment_1.html**.

Здесь выравнивание текста задано в родительском классе `container` (`text-center`). В этом случае во всех дочерних элементах текст будет выровнен по центру. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.18, — во всех дочерних элементах контейнера текст выровнен по центру адаптивного блока.

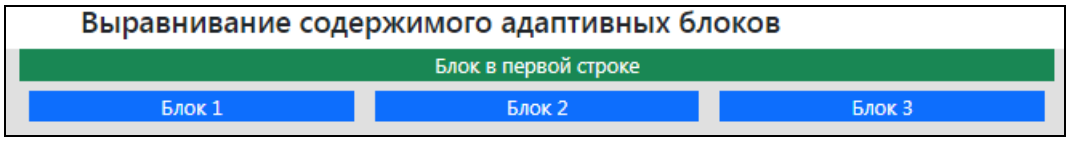


Рис. 1.18. Центрирование текста во всех адаптивных блоках

Немного изменим код этого примера и во втором ряду поменяем параметры выравнивания текста (листинг 1.10, страница alignment_2.html).

Листинг 1.10. Страница alignment_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница alignment_2.html**.

Здесь выравнивание текста по центру определено в родительском классе `container` (`text-center`). Однако во второй строке для первого блока задано выравнивание по левому краю (`text-start`), а для третьего блока — по правому краю (`text-end`). В окне браузера данная страница будет выглядеть так, как показано на рис. 1.19, — в адаптивных блоках текст выровнен по-разному: по центру, по левому и по правому краю.

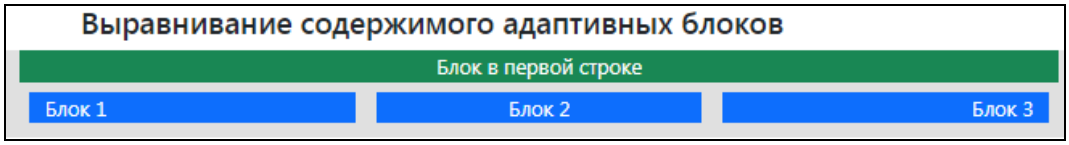


Рис. 1.19. Варианты выравнивания текста в адаптивных блоках

1.9. Обозначение границ элементов макета HTML-страниц

В Bootstrap есть утилиты для формирования границ элементов макета, которые позволяют создать рамку, радиус закругления рамки, ее толщину и цвет. Рамки отлично подходят для явного выделения блоков, для изображений, кнопок или любых других элементов интерфейса. Рамка может обрамлять элемент полностью или только одну из его сторон (рис. 1.20).



Рис. 1.20. Различные варианты обозначения границы элемента

Для добавления границ применяются следующие классы:

- `class="border"` — граница с четырех сторон;
- `class="border-top"` — граница сверху;
- `class="border-end"` — граница справа;
- `class="border-bottom"` — граница снизу;
- `class="border-start"` — граница слева.

Ширину рамки можно задать с помощью следующих классов:

- `class="border border-1"`;
- `class="border border-2"`;
- `class="border border-3"`;
- `class="border border-4"`;
- `class="border border-5"`.

Пример HTML-кода страницы, в которой показано создание границ вокруг адаптивных блоков, приведен в листинге 1.11 (страница border_1.html).

Листинг 1.11. Страница border_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **border_1.html**.

Здесь созданы границы вокруг адаптивных блоков с разных сторон, цвет линии задан по умолчанию, а толщина — `border-5`. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.21, — адаптивные блоки имеют ограничительные линии, расположенные с разных сторон.

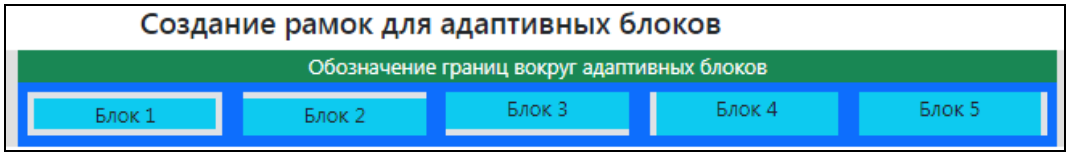


Рис. 1.21. Различные варианты обозначения границы вокруг адаптивного блока

Есть еще один способ обрамления адаптивных блоков, позволяющий убирать границу с одной из сторон. Для этих целей используется следующий класс:

- `class="border-0"` — убирается ограничительная линия с четырех сторон;
- `class="border-top-0"` — убирается ограничительная линия сверху;
- `class="border-end-0"` — убирается ограничительная линия справа;
- `class="border-bottom-0"` — убирается ограничительная линия снизу;
- `class="border-start-0"` — убирается ограничительная линия слева.

Пример HTML-кода страницы, в которой показано создание границ вокруг адаптивных блоков при удалении ограничительной линии с одной из сторон, приведен в листинге 1.12 (страница border_2.html).

Листинг 1.12. Страница border_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница border_2.html**.

Здесь заданы границы вокруг адаптивных блоков с разных сторон. Цвет линии установлен по умолчанию, а толщина — `border-5`. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.22, — ограничительную линию можно удалить либо со всех сторон адаптивного блока, либо с одной из его сторон.

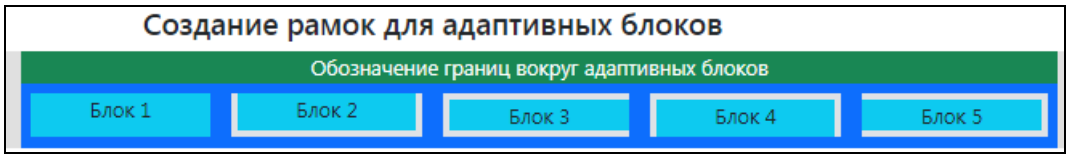


Рис. 1.22. Различные варианты обозначения границы вокруг адаптивного блока путем удаления ограничительной линии с одной из сторон

Изменить цвет границы вокруг адаптивного блока можно с помощью следующих классов:

- `class="border border-primary"`;
- `class="border border-secondary"`;
- `class="border border-success"`;
- `class="border border-danger"`;
- `class="border border-warning"`;
- `class="border border-info"`;
- `class="border border-light"`;
- `class="border border-dark"`;
- `class="border border-muted"`;
- `class="border border-white"`.

Пример HTML-кода страницы, в которой показано создание границ вокруг адаптивных блоков разного цвета, приведен в листинге 1.13 (страница border_color.html).

Листинг 1.13. Страница border_color.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница border_color.html**.

Здесь заданы цвета рамок границы вокруг адаптивных блоков с разных сторон толщиной линии `border-5`. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.23, — для ограничительных рамок можно задавать разные цвета.



Рис. 1.23. Различные варианты цвета границы вокруг адаптивных блоков

Обратите внимание на **Блок 9**, где почти не видно ограничительных рамок. Это связано с тем, что ограничительная рамка имеет цвет, близкий к цвету фона.

Углы ограничительных рамок можно скруглить — для этого предусмотрены следующие классы:

- `class="rounded"`;
- `class="rounded-top"`;
- `class="rounded-end"`;
- `class="rounded-bottom"`;
- `class="rounded-start"`;
- `class="rounded-circle"`;
- `class="rounded-pill"`.

Пример HTML-кода страницы, в которой показано создание рамок со скругленными углами, приведен в листинге 1.14 (страница border_radius.html).

Листинг 1.14. Страница border_radius.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница border_radius.html**.

Здесь заданы рамки вокруг адаптивных блоков с толщиной линии `border-3`, при этом реализованы разные варианты скругленных углов. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.24.

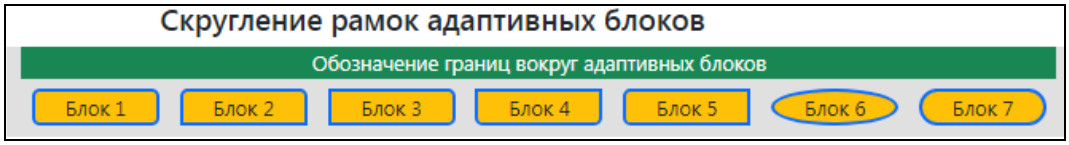


Рис. 1.24. Различные варианты скругленных углов рамок вокруг адаптивного блока

Для задания радиуса скругления углов ограничительных рамок используются следующие классы:

- `class="rounded-0"`;
- `class="rounded-1"`;
- `class="rounded-2"`;
- `class="rounded-3"`;
- `class="rounded-4"`;
- `class="rounded-5"`.

Пример HTML-кода страницы, в которой показано создание рамок с разными радиусами скругления углов, приведен в листинге 1.15 (страница border_radius_1.html).

Листинг 1.15. Страница border_radius_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница border_radius_1.html**.

Здесь созданы рамки вокруг адаптивных блоков с толщиной линии `border-3`, при этом заданы разные радиусы скругленных углов. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.25.

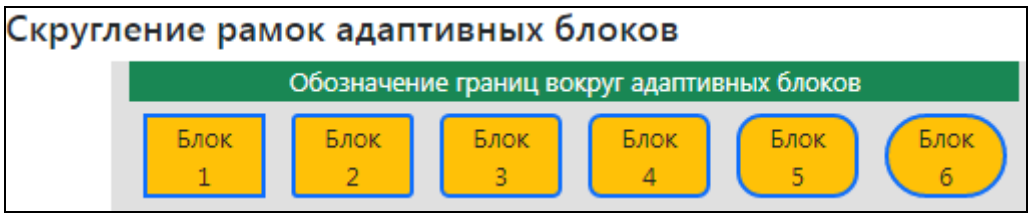


Рис. 1.25. Различные варианты радиусов скругления углов рамок вокруг адаптивного блока

1.10. Пример использования адаптивных контейнеров

На HTML-странице могут присутствовать несколько адаптивных контейнеров. Для создания адаптивного контейнера с фиксированной шириной предусмотрен класс `container`. В качестве примера создадим HTML-страницу с двумя адаптивными контейнерами с фиксированной шириной (листинг 1.16, страница container_1.html).

Листинг 1.16. Страница container_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница container_1.html**.

На этой странице имеются два контейнера класса `container`. В первом контейнере находится текст, во втором — создан один ряд, в котором содержатся три адаптивных блока с вложенным в них текстом. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.26.

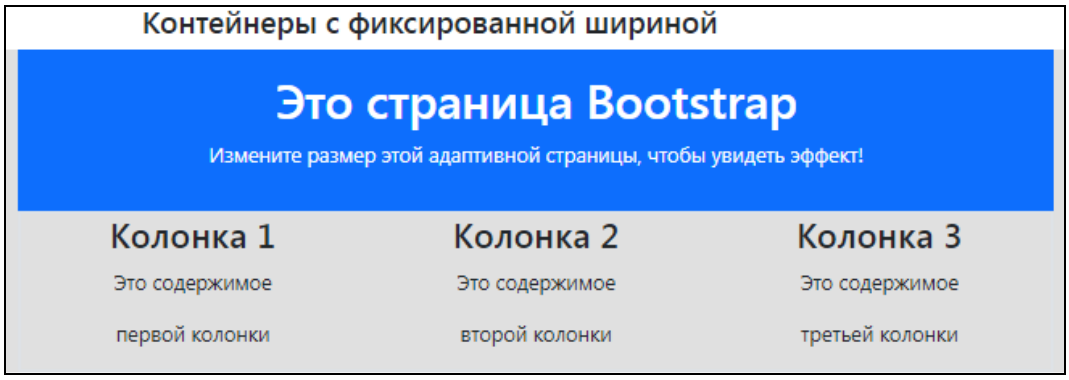


Рис. 1.26. Страница с двумя контейнерами с фиксированной шириной класса `container`

Обратите внимание, что при изменении ширины окна браузера меняется ширина контейнеров и их содержимого, при этом окно браузера всегда шире, чем ширина контейнеров (слева и справа остается свободное пространство).

Теперь создадим HTML-страницу с двумя адаптивными подвижными, или «резиновыми», контейнерами на основе класса `container-fluid` (листинг 1.17, страница container_2.html).

Листинг 1.17. Страница container_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница container_2.html**.

На этой странице имеются два контейнера класса `container-fluid`. В первом контейнере находится текст, во втором — создан один ряд, в котором содержатся три адаптивных блока с вложенным в них текстом. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.27.

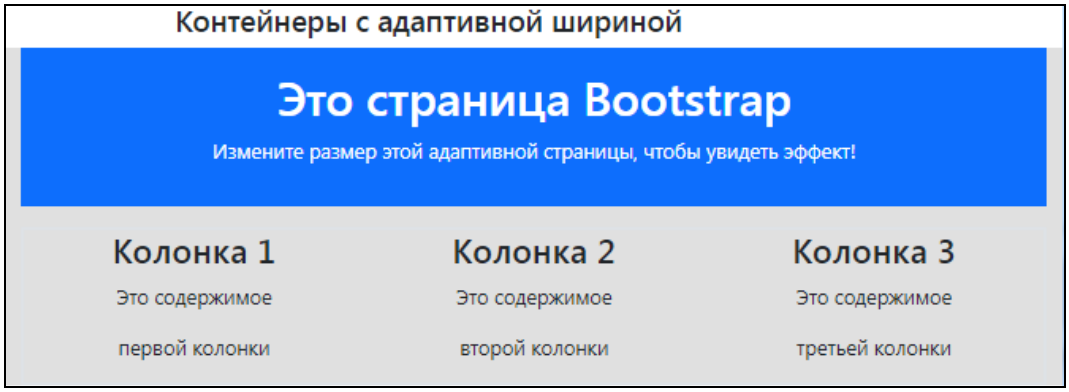


Рис. 1.27. Страница с двумя адаптивными подвижными контейнерами класса `container-fluid`

Обратите внимание, что при изменении ширины браузера меняется ширина контейнеров и их содержимого, при этом ширина контейнеров всегда соответствует ширине окна браузера (слева и справа нет свободного пространства). При минимальной ширине браузера колонки трансформируются в три строки.

* * *

На этом мы закончим знакомство с макетированием HTML-страниц на основе фреймворка Bootstrap. Конечно, в Bootstrap имеется множество различных визуальных элементов для создания привлекательного веб-интерфейса, о которых вы можете узнать в специальной литературе. Некоторые из этих элементов мы будем применять в шаблонах HTML-страниц и более детально познакомимся с ними в следующих главах на конкретных примерах.

1.11. Таблицы Bootstrap

Таблицы Bootstrap можно использовать как для вывода данных, так и для макетирования страниц. В базовом варианте таблицы имеют небольшие отступы и горизонтальные разделители. Для того чтобы создать таблицу, понадобятся следующие теги:

```
<table class="table">          ← Таблица
    <thead>                    ← Заголовок таблицы
        <tr>                   ← Строка заголовка
            <th>…</th>                 ← Колонка заголовка
                …              ← Колонка заголовка
            <th>…</th>                 ← Колонка заголовка
        </tr>
    </thead>

    <tbody>                    ← Тело таблицы
        <tr>                   ← Строка в теле таблицы
            <td>…</td>                 ← Колонка в теле таблицы
                …              ← Колонка в теле таблицы
            <td>…/td>          ← Колонка в теле таблицы
        </tr>
     </tbody>

</table>
```

Как здесь можно видеть, таблицу создают с помощью тегов:

- `<table>` — создает собственно таблицу;
- `<thead>` — заголовок таблицы;
- `<tbody>` — тело таблицы;
- `<tr>` — строку;
- `<td>` — колонку.

В качестве примера создадим HTML-страницу с таблицей, состоящей из трех строк и трех колонок (листинг 1.18, страница table.html).

Листинг 1.18. Страница table.html

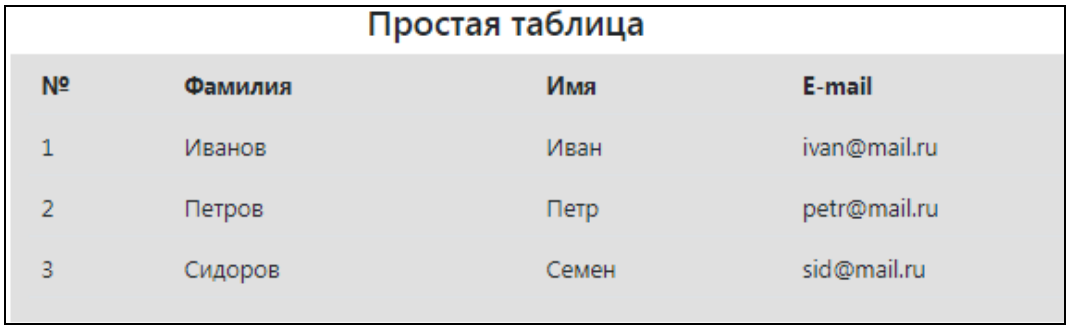
Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница table.html**.

Здесь нет контейнеров, и таблица находится непосредственно в теле HTML-страницы. В окне браузера эта страница будет выглядеть так, как показано на рис. 1.28.



Простая таблица

№	Фамилия	Имя	E-mail
1	Иванов	Иван	ivan@mail.ru
2	Петров	Петр	petr@mail.ru
3	Сидоров	Семен	sid@mail.ru

Рис. 1.28. Таблица Bootstrap

Как здесь можно видеть, с параметрами по умолчанию таблица имеет выделенный заголовок (жирный шрифт), строки и колонки таблица явно не выделены, цвет фона таблицы соответствует цвету фона основной страницы.

Считается хорошим тоном, когда четные и нечетные строки таблицы выделены альтернативными цветами. В Bootstrap это можно сделать с помощью класса `table-striped`. В качестве примера создадим HTML-страницу с таблицей, состоящей из трех строк и трех колонок с выделением цветом четных и нечетных строк (листинг 1.19, страница table_1.html).

Листинг 1.19. Страница table_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница table_1.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 1.29, — с классом `table-striped` таблица имеет выделенный жирным шрифтом заголовок, а четные и нечетные строки выделены разными цветами.



Простая таблица с выделением строк

№	Фамилия	Имя	E-mail
1	Иванов	Иван	ivan@mail.ru
2	Петров	Петр	petr@mail.ru
3	Сидоров	Семен	sid@mail.ru

Рис. 1.29. Таблица Bootstrap, где строки выделены альтернативными цветами

Для того чтобы создать для таблицы границы ячеек, используется класс `table-bordered`, — при этом цвет границ задается с помощью класса `bordered <цвет>`. В качестве примера создадим HTML-страницу с таблицей, состоящей из трех строк и трех колонок с рамками для выделения ячеек цвета `<primary>` (листинг 1.20, страница table_2.html).

Листинг 1.20. Страница table_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница table_2.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 1.30.

Простая таблица с выделением строк и с границами ячеек

№	Фамилия	Имя	E-mail
1	Иванов	Иван	ivan@mail.ru
2	Петров	Петр	petr@mail.ru
3	Сидоров	Семен	sid@mail.ru

Рис. 1.30. Таблица Bootstrap с границами ячеек

Если таблица используется для макетирования страницы, то все границы можно явно удалить, для чего предусмотрен класс `<table class="table table-borderless">`. В качестве примера создадим HTML-страницу с таблицей, состоящей из трех строк и трех колонок с явно удаленными границами ячеек (листинг 1.21, страница table_3.html).

Листинг 1.21. Страница table_3.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница table_3.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 1.31.

Простая таблица с удаленными границами ячеек

№	Фамилия	Имя	E-mail
1	Иванов	Иван	ivan@mail.ru
2	Петров	Петр	petr@mail.ru
3	Сидоров	Семен	sid@mail.ru

Рис. 1.31. Таблица Bootstrap без границ ячеек

В таблицах можно использовать контекстные классы для окрашивания всей таблицы (`<table>`), строк таблицы (`<tr>`) или ячеек таблицы (`<td>`):

- `table-primary` — синий, указывает на важное действие;
- `table-success` — зеленый, указывает на успешное или положительное действие;

- `table-danger` — красный, указывает на опасное или потенциально негативное действие;
- `table-info` — голубой, указывает на нейтральное информативное изменение или действие;
- `table-warning` — оранжевый, указывает на предупреждение;
- `table-active` — серый, применяется при наведении курсора мыши на строку или ячейку таблицы;
- `table-secondary` — серый, указывает на менее важное действие;
- `table-light` — светло-серый фон таблицы или строки таблицы;
- `table-dark` — черный фон таблицы или строки таблицы.

В качестве примера создадим HTML-страницу с таблицей, в которой показаны классы с основными цветами (листинг 1.22, страница table_4.html).

Листинг 1.22. Страница table_4.html

Примечание.
В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница table_4.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 1.32.



Рис. 1.32. Таблица Bootstrap со строками разного цвета

1.12. Адаптивные таблицы в Bootstrap (с полосами прокрутки)

На странице сайта может присутствовать таблица с большим количеством колонок или строк. На больших экранах компьютера такая таблица может вписываться в размер страницы, и все ее ячейки находятся в пределах видимости. Но на устройствах с меньшим разрешением, — например, на планшетах и смартфонах, часть ячеек выйдет за пределы видимо-

сти. В этом случае для больших таблиц необходимо использовать полосы прокрутки. Для того, чтобы таблица имела полосы прокрутки, ее нужно поместить в тег с указанием стиля `overflow:scroll`:

```
<div style="overflow:scroll;">
.....
</div>
```

При этом полосы прокрутки рядом с таблицей будут видны постоянно. В качестве примера создадим HTML-страницу с таблицей, состоящей из трех строк и четырех колонок с полосами прокрутки (листинг 1.23, страница table_5.html).

Листинг 1.23. Страница table_5.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница table_5.html**.

Если таблица сможет поместиться в границы страницы, то полосы прокрутки будут видны, но не будут активными (рис. 1.33).

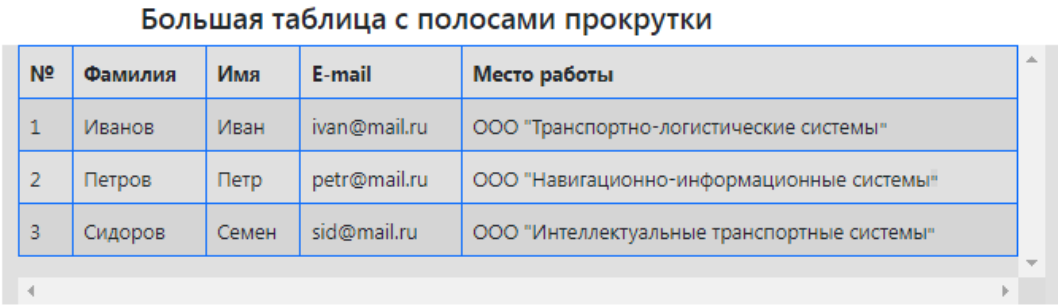


Рис. 1.33. Таблица Bootstrap с не активными полосами прокрутки

Если же таблица не сможет поместиться в границы страницы, то полосы прокрутки станут активными, — то есть на них будет виден бегунок для скроллинга таблицы (рис. 1.34).



Рис. 1.34. Таблица Bootstrap с активной горизонтальной полосой прокрутки

Для привязки к таблице полос прокрутки можно использовать специальный класс `<div class="table-responsive">`:

```
<div class="table-responsive">
.....
</div>
```

При этом, если таблица полностью помещается в пределы страницы, то полосы прокрутки рядом с таблицей вовсе не будут видны. Если же таблица не будет вписываться в размеры страницы, то полосы прокрутки автоматически появятся. В качестве примера создадим HTML-страницу с таблицей, состоящей из трех строк и четырех колонок с полосами прокрутки на основе класса `table-responsive` (листинг 1.24, страница table_6.html).

Листинг 1.24. Страница table_6.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 1** и перейти по ссылке **Страница table_6.html**.

Если таблица сможет поместиться в границы страницы, то полосы прокрутки не будут видны (рис. 1.35).

Большая таблица с полосами прокрутки

№	Фамилия	Имя	E-mail	Место работы
1	Иванов	Иван	ivan@mail.ru	ООО "Транспортно-логистические системы"
2	Петров	Петр	petr@mail.ru	ООО "Навигационно-информационные системы"
3	Сидоров	Семен	sid@mail.ru	ООО "Интеллектуальные транспортные системы"

Рис. 1.35. Таблица Bootstrap с невидимыми полосами прокрутки

Если же таблица не сможет поместиться в границы страницы, то полосы прокрутки станут видимыми (рис. 1.36).

№	Фамилия	Имя	E-mail	Мест
1	Иванов	Иван	ivan@mail.ru	ООО "Тран логис систе
2	Петров	Петр	petr@mail.ru	ООО "Нави инфо систе
3	Сидоров	Семен	sid@mail.ru	ООО "Инте транс систе

Рис. 1.36. Таблица Bootstrap с видимой горизонтальной полосой прокрутки

1.13. Краткие итоги

Итак, мы познакомились с некоторыми базовыми элементами фреймворка Bootstrap, которые нам пригодятся при формировании шаблонов HTML-страниц. Но прежде чем приступить к их использованию в веб-приложениях, нужно узнать о некоторых важных парамет-

рах элементов Bootstrap и способах их подключения к шаблонам Django. В частности нам предстоит разобраться с тем:

- как можно менять цветовое оформление элементов Bootstrap;
- как можно встраивать одни шаблоны Django в другие;
- как работает механизм подключения значков к элементам Bootstrap;
- как можно сделать разбивку HTML-страницы на горизонтальные и вертикальные вкладки и организовать навигацию по ним;
- какие есть варианты использования атрибута `href` в шаблонах Django для создания ссылок на интернет-адреса (URL).
- как создавать и использовать всплывающие модальные окна;
- как отобразить в шаблонах Django статические компоненты Bootstrap (на примере аккордеона, карточек и карусели).

Рассмотрению этих вопросов и посвящена следующая глава.

Глава 2. Статичные элементы Bootstrap в шаблонах Django

Дизайн практически всех современных сайтов строится с использованием элементов Bootstrap. Обращение к этому фреймворку обеспечивает адаптивность страниц сайта к экранам разного разрешения, а главное — сокращает время, затрачиваемое программистами, на разработку программного кода. Статичные элементы Bootstrap достаточно просто встраиваются в шаблоны Django и имеют множество параметров для настройки внешнего вида. Однако эти элементы статичны, и если возникнет потребность изменить их содержание или внешний вид (размер, цвет, форму), то приходится менять программный код и обновлять его на публичных серверах. Из материалов этой главы вы узнаете:

- как изменить цветовое оформление элементов Bootstrap;
- как использовать тег `include` для встраивания одних шаблонов Django в другие;
- как подключать значки к элементам Bootstrap;
- как выполнять разделение HTML-страницы на горизонтальные и вертикальные вкладки компонентом `Tabs`;
- как с элементами Bootstrap связать подсказки `Tooltip`;
- какие есть варианты использование атрибута `href` в шаблонах Django для создания ссылок на на интернет-адреса (URL);
- как на HTML-страницы вывести всплывающие модальные окна;
- как создать программные модули для отображения в шаблонах Django статичных элементов Bootstrap (на примере аккордеона, карточек и карусели).

2.1. Задание произвольного цвета элементам Bootstrap

Для задания цвета элементам Bootstrap используются 8 базовых классов (рис. 2.1):

- `text-primary` — синий;
- `text-secondary` — серый;
- `text-success` — зеленый;
- `text-danger` — красный;
- `text-warning` — оранжевый;
- `text-info` — голубой;
- `text-light` — светло-серый ;
- `text-dark` — черный.

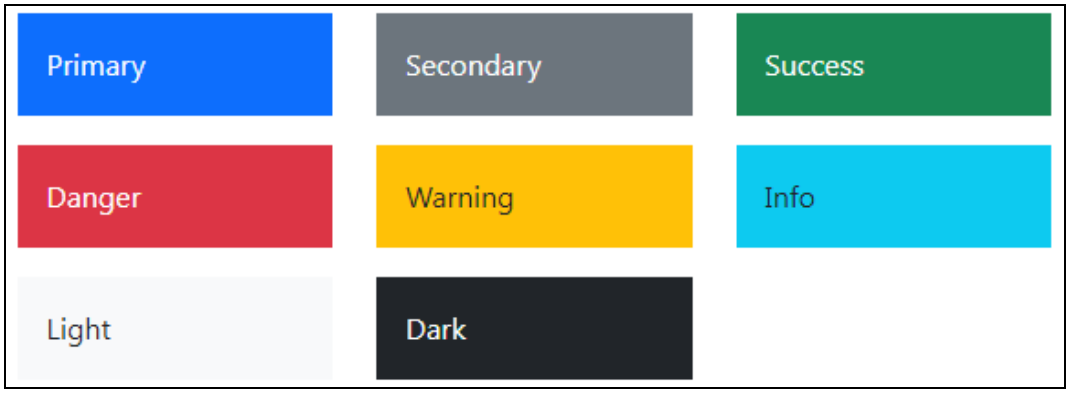


Рис. 2.1. Образцы базовых цветов в Bootstrap

Пример HTML-кода страницы, в которой показано использование базовых цветов для одновременного задания цвета для шрифта и для фона, приведен в листинге 2.1 (страница base_color.html).

Листинг 2.1. Страница base_color.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница base_color.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 2.2.



Рис. 2.2. Образцы базовых цветов для фона и текста в Bootstrap

При создании страниц сайтов можно не ограничиваться базовыми цветами Bootstrap, а использовать полную палитру цветов RGB. Для этого элементу Bootstrap надо изменить стиль. Пример HTML-кода страницы, в которой показано использование стилей для задания цвета фона адаптивным блокам, приведен в листинге 2.2 (страница style_color.html).

Листинг 2.2. Страница style_color.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница style_color.html**.

Как показано в этом листинге, цвет можно задать с помощью стиля либо в десятичном виде — например: `rgb(160, 82, 45)`, либо в шестнадцатеричном — например: `#FFFF00`. В окне браузера эта страница будет выглядеть так, как показано на рис. 2.3.

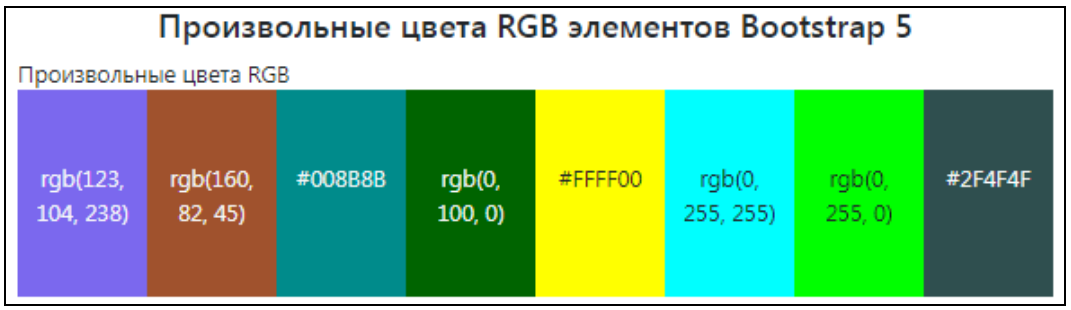


Рис. 2.3. Образцы произвольных цветов RGB для фона адаптивных блоков

В Bootstrap 5 есть возможность менять оттенки базового цвета за счет использования такого параметра, как прозрачность, а вернее, — непрозрачность: `opacity`. Параметр непрозрачности может принимать значения: 100% (по умолчанию), 75%, 50% и 10%. Пример HTML-кода страницы, в которой показано использование параметра непрозрачности для задания оттенка базового цвета фона адаптивных блоков, приведен в листинге 2.3 (страница opacity.html).

Листинг 2.3. Страница opacity.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница opacity.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 2.4.

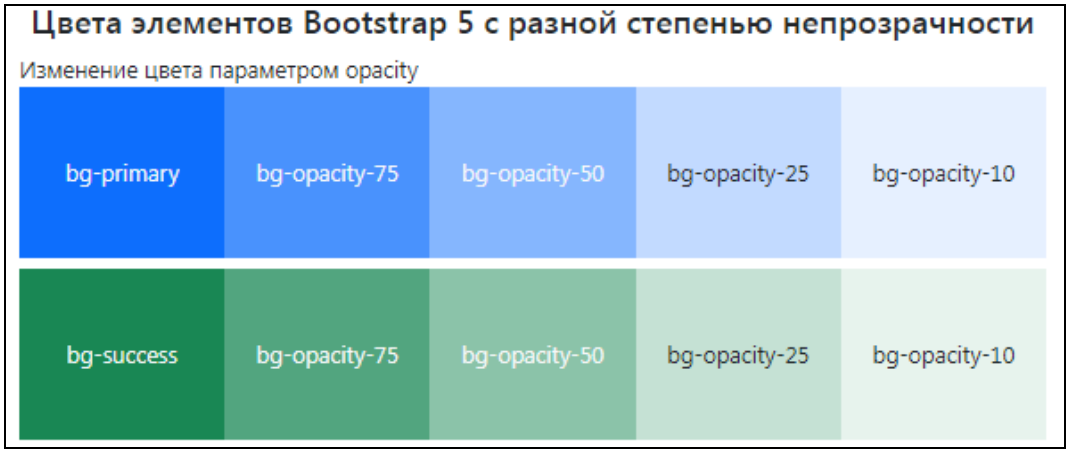


Рис. 2.4. Изменение цвета фона адаптивных блоков с использованием параметра `opasity`

2.2. Тег *include* для встраивания шаблонов Django

В элементах Bootstrap (в строках и адаптивных блоках) можно писать любой код HTML. Однако при большом количестве строк и высокой степени вложенности элементов программный код страницы плохо читается и его трудно отлаживать. Поэтому целесообразно сложные HTML-страницы разбивать на более простые, и вкладывать одну страницу в другую. Кроме того, такой подход позволяет содержимое одной HTML-страницы использовать в качестве фрагмента во множестве других HTML-страниц. Для этого в Django есть специальный тег `include`, который имеет следующий формат:

```
{% include "my_page.html" %},
```

Здесь `my_page.html` — имя встраиваемой страницы. При этом в то место сетки Bootstrap, где находится тег `include`, будет вставлен полный код страницы `my_page.html`.

Пример HTML-кода страницы, в которой показано использование тега `include` для встраивания шаблонов Django в другие страницы сайта, приведен в листинге 2.4 (страница include.html).

Листинг 2.4. Страница include.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница include.html**.

В этом программном модуле тег `include` находится в адаптивном блоке `col`, а в самом этом блоке указан путь к странице opacity.html (ее URL-адрес):

```
<div class="col text-center">
    {% include "master/glava_2/inc_page.html" %}
</div>
```

Программный код страницы inc_page.html приведен в листинге 2.5.

Листинг 2.5. Страница inc_page.html

Примечание.

В демо-версии книги удалены листинги программ.

Теперь при вызове шаблона include.html Django на место тега `include` поместит код из другого шаблона с именем inc_page.html. В итоге в окне браузера страница include.html будет выглядеть так, как показано на рис. 2.5.

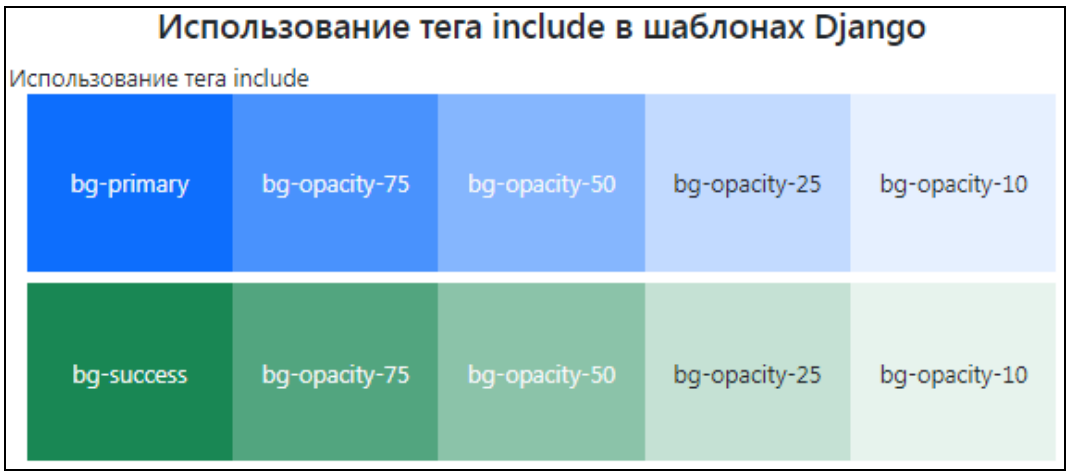


Рис. 2.5. Результаты встраивания шаблона opacity.html в шаблон include.html

Таким образом, с использованием тега `include`, можно многострочный код громоздкого шаблона разбить на простые фрагменты, отладить работу каждого фрагмента, а итоговую страницу показывать как единое целое.

2.3. Подключение значков к элементам Bootstrap

Фреймворк Bootstrap предлагает коллекцию, содержащую порядка двух тысяч значков (иногда называемых «иконками») в формате SVG, которые можно использовать в своих проектах. В зависимости от необходимости, допускается менять цвет этих значков — через свойство CSS `color` и их размер — через свойство CSS `font-size`. Подключить значки к проекту можно по следующей ссылке:

```
<link  href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.10.5/font/bootstrap-icons.css" rel="stylesheet">
```

Эту ссылку рекомендуется поместить в базовый шаблон сайта — тогда значки можно будет использовать на всех дочерних шаблонах. Как выглядят значки, и какой код надо применять для их подключения к элементу страницы, можно посмотреть на официальном сайте коллекции[1]. Фрагмент страницы этого сайта представлен на рис. 2.6.

[1] См. **https://icons.getbootstrap.com**.

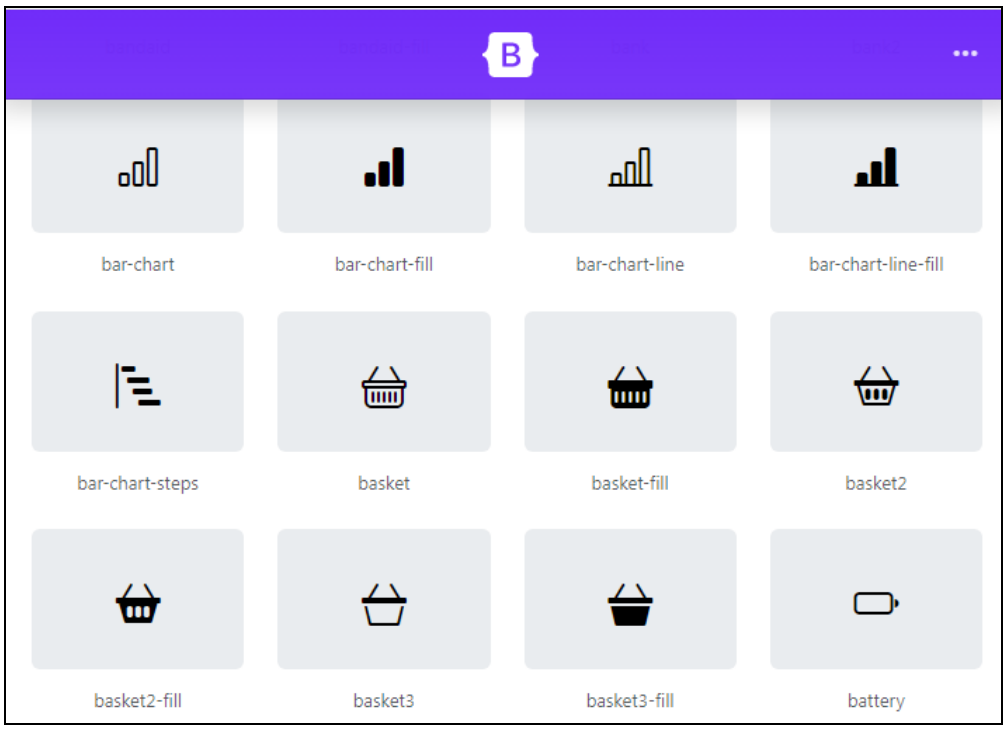


Рис. 2.6. Фрагмент официального сайта коллекции значков Bootstrap

Чтобы узнать код для подключения значка к элементу сайта, нужно щелкнуть левой кнопкой мыши на изображении значка, — откроется новая страница, на которой будут показаны параметры подключения. Например, если щелкнуть мышью на значке с именем **battery**, мы увидим страницу со следующими параметрами подключения (рис. 2.7).

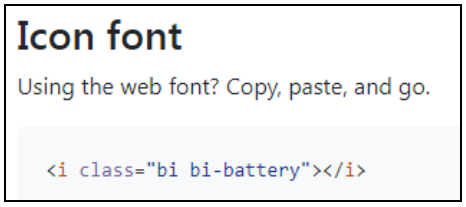


Рис. 2.7. Фрагмент кода для подключения значка к элементу Bootstrap

Пример HTML-кода страницы, в которой показано использование значков на кнопках Bootstrap, приведен в листинге 2.6 (страница icon_1.html).

Листинг 2.6. Страница icon_1.html

Примечание.
В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница icon_1.html**.

В этом шаблоне созданы три кнопки разного цвета с расположением значков (иконок) слева и справа от текстовой надписи. В итоге в окне браузера страница icon_1.html будет будет выглядеть так, как показано на рис. рис. 2.8.

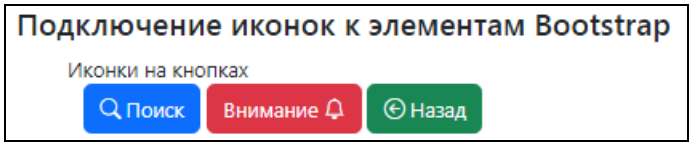


Рис. 2.8. Значки на кнопках Bootstrap

Значки можно поместить в любое место страницы сайта, указав их нужный цвет и размер. Пример HTML-кода страницы, в которой показано отображение значков разного цвета и размера в адаптивных блоках, приведен в листинге 2.7 (страница icon_2.html).

Листинг 2.7. Страница icon_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница icon_2.html**.

В этом шаблоне созданы три значка (иконки) разного цвета (цвет задан классами `text-primary`, `text-danger` и `text-success`) и разного размера (размер задан стилями `font-size: 3rem`, `font-size: 5rem` и `font-size: 7rem`). В итоге в окне браузера страница icon_2.html выглядеть так, как показано на рис. 2.9.



Рис. 2.9. Значки разного цвета и размера на странице сайта

Нужный цвет и размер значкам можно задать и с помощью стилей. Пример HTML-кода страницы, в которой показано задание цвета и размера значков с помощью стилей, приведен в листинге 2.8 (страница icon_3.html).

Листинг 2.8. Страница icon_3.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница icon_3.html**.

В этом шаблоне созданы три значка (иконки) разного размера (размер задан стилями `font-size: 48px`, `font-size: 64px` и `font-size: 76px`) и разного цвета (цвет задан стилями `color: red`, `color: blue` и `color: green`). В итоге в окне браузера страница icon_3.html будет выглядеть так, как показано на рис. 2.10.

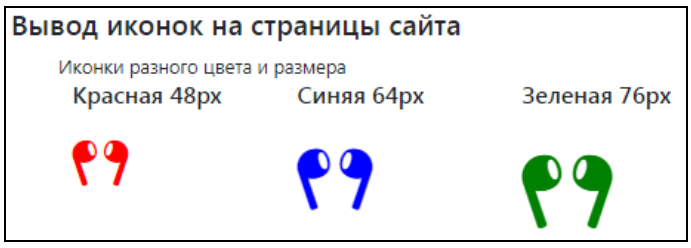


Рис. 2.10.Значки разного цвета и размера с параметрами, заданными через стили

2.4. Разделение страницы на вкладки компонентом *Tabs*

Компонент `Tabs` фреймворка Bootstrap предназначен для добавления на страницу вкладок. Вкладки используются тогда, когда контент нужно показывать не сразу весь целиком, а определенными порциями. Переключение между этими порциями и осуществляется с помощью вкладок.

Какие преимущества дает использование вкладок? Во-первых, вкладки улучшают организацию данных на странице и ее внешний вид: доступ к информации становится более простым и понятным. Во-вторых, занимая определенную часть страницы, они позволяют разместить на ней намного больше информации: страница становится более компактной и в ней намного проще ориентироваться. Мы рассмотрим несколько вариантов использования вкладок:

- традиционные горизонтальные вкладки;
- вкладки, созданные с использованием циклов Django;
- вертикальные вкладки с использованием кнопок.

2.4.1. Традиционные горизонтальные вкладки *Tabs* фреймворка Bootstrap

Реализация вкладок начинается с создания списка `ul` с классами `nav` и `nav-tabs`. Затем в `ul` нужно поместить определенное количество элементов `li` с классом `nav-item` — это количество должно соответствовать количеству вкладок, которые мы хотим создать. Пример HTML-кода страницы, в которой показано создание трех горизонтальных вкладок, приведен в листинге 2.9 (страница tabs_1.html).

Листинг 2.9. Страница tabs_1.html

> ***Примечание.***
> В демо-версии книги удалены листинги программ.
>
> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница tabs_1.html**.

В этом программном модуле созданы три вкладки с заголовками: `Вкладка 1`, `Вкладка 2` и `Вкладка 3`. Для создания группы заголовков вкладок использовался класс `class="nav nav-tabs`. Каждый заголовок сформирован на основе класса `class="nav-item"` и оформлен в виде ссылки с указанием идентификатора вкладки: `href="#tab_1`, `href="#tab_2"` и `href="#tab_3"`. Для первой вкладки указано, что при открытии блока с вкладками она будет активной: `class="nav-link active"`.

Для создания группы с содержимым вкладок использовался класс `class="tab-content"`. Формирование содержимого каждой отдельной вкладки реализовано на основе класса `class="tab-pane fade"`. Для содержимого каждой вкладки указывается тот же идентификатор, который был задан в соответствующем заголовке: `id="tab_1"`, `id="tab_2"` и `id="tab_3"`. При этом при открытии группы вкладок будет отображаться содержимое первой (активной) вкладки:

```
class="tab-pane fade show active" id="tab_1".
```

В качестве содержимого каждая вкладка содержит ссылку на другую страницу сайта с использованием тега Django `include`. Эти страницы находятся по следующим URL-адресам (в папках, вложенных в папку templates сайта):

- `{% include "master/glava_2/tab_1.html" %}`;
- `{% include "master/glava_2/tab_2.html" %}`;
- `{% include "master/glava_2/tab_3.html" %}`.

Указанные страницы содержат код, приведенный в листингах 2.9.1, 2.9.2 и 2.9.3.

Листинг 2.9.1. Страница tab_1.html

Примечание.
В демо-версии книги удалены листинги программ.

Листинг 2.9.2. Страница tab_2.html

Примечание.
В демо-версии книги удалены листинги программ.

Листинг 2.9.3. Страница tab_3.html

Примечание.
В демо-версии книги удалены листинги программ.

В итоге в окне браузера страница tabs_1.html будет выглядеть так, как представлено на рис. 2.11, рис. 2.12 и рис. 2.13.



Рис. 2.11. Страница tabs_1.html с открытой вкладкой **Вкладка 1**

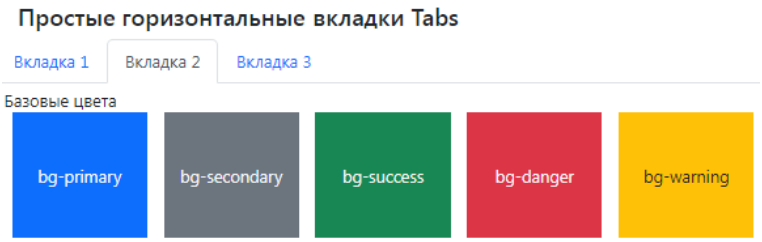


Рис. 2.12. Страница tabs_1.html с открытой вкладкой **Вкладка 2**

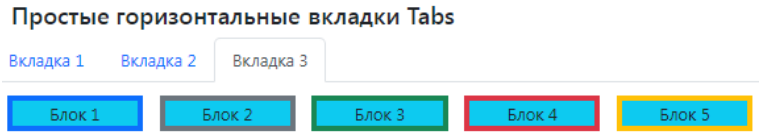


Рис. 2.13. Страница tabs_1.html с открытой вкладкой **Вкладка 3**

Как показано на этих рисунках, в одной и той же области экрана на одной странице за счет использования вкладок можно отображать совершенно разную информацию.

2.4.2. Традиционные вертикальные вкладки *Tabs* фреймворка Bootstrap

Реализация вертикальных вкладок практически не отличается от реализации горизонтальных — нужно просто создать два адаптивных блока `row`, при этом в левый блок поместить заголовки вкладок, а в правом блоке отобразить их контент. Пример HTML-кода страницы,

в которой показано создание трех вертикальных вкладок, приведен в листинге 2.10 (страница tabs_2.html).

Листинг 2.10. Страница tabs_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница tabs_2.html**. На больших экранах для того чтобы вертикальные кнопки отобразились с левой стороны экрана, необходимо уменьшить размер окна браузера.

Шаблон этой HTML-страницы имеет следующую структуру:

```
<div class="row">
    <div class="col-md-2">
        <!-- Заголовки вкладок -->
    </div>
    <div class="col-md-10">
        <!-- Содержимое вкладок -->
    </div>
</div>
```

В этом шаблоне создана строка `row`, в которую вложены два адаптивных блока: первый блок размером в две колонки (`<div class="col-md-2">`) и второй блок размером 10 колонок (`<div class="col-md-10">`). В первом блоке находится класс для создания заголовков вкладок, во втором блоке — класс для создания контента вкладок. В итоге в окне браузера страница tabs_2.html будет выглядеть так, как показано на рис. 2.14, рис. 2.15 и рис. 2.16.

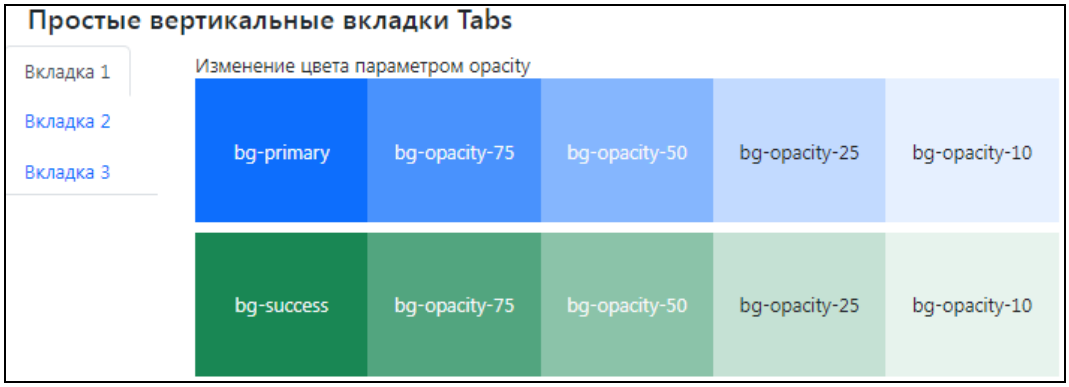


Рис. 2.14. Страница tabs_2.html с открытой вкладкой **Вкладка 1**

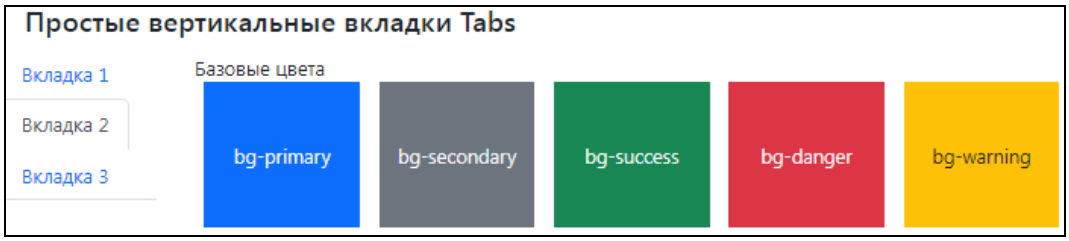


Рис. 2.15. Страница tabs_2.html с открытой вкладкой **Вкладка 2**



Рис. 2.16. Страница tabs_2.html с открытой вкладкой **Вкладка 3**

2.4.3. Горизонтальные вкладки *Tabs* с использованием тега Django для организации циклов

Достаточно часто при разработке сайтов возникает необходимость создания динамических вкладок. Например, когда загружаются несколько записей из БД, каждую из которых необходимо показать в отдельной вкладке. При этом могут различаться количество вкладок, их заголовки и содержимое контента. Такие вкладки можно создавать динамически с использованием тега Django для организации циклов.

Для создания динамических вкладок потребуется согласованная работа сразу нескольких модулей: модуля с определением имени URL-маршрута, самого URL-маршрута с указанием имени представления (`view`), представления, в котором загружаются динамические данные и указывается имя шаблона и, наконец, сам шаблон для отображения данных.

В следующем примере приведены коды этих модулей, которые задействованы в рамках нашей интерактивной цифровой платформы. В ваших проектах имена файлов и пути их размещения будут совершенно другими, но код итогового шаблона можно взять за основу для подключения к собственным проектам.

Итак, начнем с модуля, в котором определяется имя URL-маршрута (ссылка на URL маршрут):

```
<a href="{% url 'master:tabs_3' %}" target ="_blank">Страница tabs_3.html</a>
```

По этой ссылке из приложения `master` вызывается маршрут с именем `tabs_3`.

Сам маршрут в файле urls.py имеет следующий код:

```
path('glava_2/tabs_3/', views.tabs_3, name='tabs_3')
```

По этому маршруту будет активировано представление с именем `tabs_3`.

Представление с именем `tabs_3` имеет следующий упрощенный код:

```
def tabs_3(request):
    tab = ['Вкладка 1', 'Вкладка 2', 'Вкладка 3']
    content = {'tab': tab}
    return render(request, 'master/glava_2/tabs_3.html', content)
```

В этом представлении наименования вкладок `Tabs` сформированы в виде простого списка `tab`. В реальных проектах имена вкладок (как и их содержимое) могут быть подгружены из БД, и все эти данные упакованы в списки или кортежи.

На следующем шаге полученные динамические данные упаковываются в словарь с именем `content`, и этот словарь передается в шаблон с именем tabs_3.html.

Упрощенный пример HTML-кода страницы, в которой показано создание трех динамических горизонтальных вкладок, приведен в листинге 2.11 (страница tabs_3.html).

Листинг 2.11. Страница tabs_3.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница tabs_3.html**.

Рассмотрим более детально код этого шаблона, в котором, собственно, и будут сформированы динамические вкладки.

Сразу следует отметить, что при использовании циклов программный код значительно короче, чем коды предыдущих примеров, где они также формируют три вкладки.

Формирование заголовков вкладок здесь выполняется с использованием тегов Django для создания циклов со следующей структурой:

```
{% for obj in tab %}
    . . .
{% endfor %}
```

В этом цикле идентификаторы вкладок формируются на основе счетчика циклов `href="#v{{ forloop.counter }}"`. При первоначальной загрузке страницы активной будет первая вкладка, что обеспечивает следующий класс:

```
class="nav-link {% if forloop.first %}active{% endif %}"
```

Имена заголовков вкладок создаются из объекта `obj`, который получает значения из элементов списка `tab` (в представлении этот список имеет следующие значения: `tab = ['Вкладка 1', 'Вкладка 2', 'Вкладка 3']`) — за это отвечает параметр `data-bs-toggle="tab">{{obj}}`.

Содержимое вкладок (контент) также отображается в цикле со следующей структурой:

```
{% for obj in tab %}
    {% if forloop.counter == 1 %}
        {% include "<имя файла>.html" %}
    {% elif forloop.counter == 2 %}
        {% include "<имя файла>.html" %}
    {% elif forloop.counter == 3 %}
        {% include "<имя файла>.html" %}
    {% endif %}
{% endfor %}
```

В теле цикла для активации содержимого той или иной вкладки используется следующий класс:

```
<div class="tab-pane fade {% if forloop.first %} show active{% endif %}"
id="v{{ forloop.counter }}">
```

В итоге в окне браузера страница tabs_3.html будет выглядеть так же, как показано ранее на рис. 2.14, рис. 2.15 и рис. 2.16.

2.4.4. Вертикальные кнопки для навигации по вкладкам *Tabs* фреймворка Bootstrap

Реализация вертикальных вкладок с кнопками достаточно сильно отличается от реализации обычных горизонтальных и вертикальных вкладок. Для их формирования используется класс `pill` (таблетка, капсула). Пример HTML-кода страницы, в которой показано создание трех вертикальных вкладок с кнопками, приведен в листинге 2.12 (страница tabs_4.html).

Листинг 2.12. Страница tabs_4.html

> ***Примечание.***
> В демо-версии книги удалены листинги программ.
>
> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница tabs_4.html**.

В этом программном коде созданы три кнопки с текстовой надписью: `Кнопка 1`, `Кнопка 2` и `Кнопка 3`. Каждая кнопка имеет свой идентификатор: `id="v-1-tab"`, `id="v-2-tab"`, `id="v-2-tab"`, а также связанный с кнопкой идентификатор вкладки: `data-bs-target="#v-1"`, `data-bs-target="#v-2"`, `data-bs-target="#v-3"`. Первая кнопка является активной: `class="nav-link active"`.

Каждая вкладка также имеет свой идентификатор для связи с кнопкой: `id="v-1"`, `id="v-2"`, `id="v-3"`. Первая вкладка является активной: `class="tab-pane fade show`

active". В каждую вкладку встроена другая HTML-страница с использованием тега include.

В итоге в окне браузера страница tabs_4.html будет будет выглядеть так, как показано на рис. 2.17, рис. 2.18 и рис. 2.19.

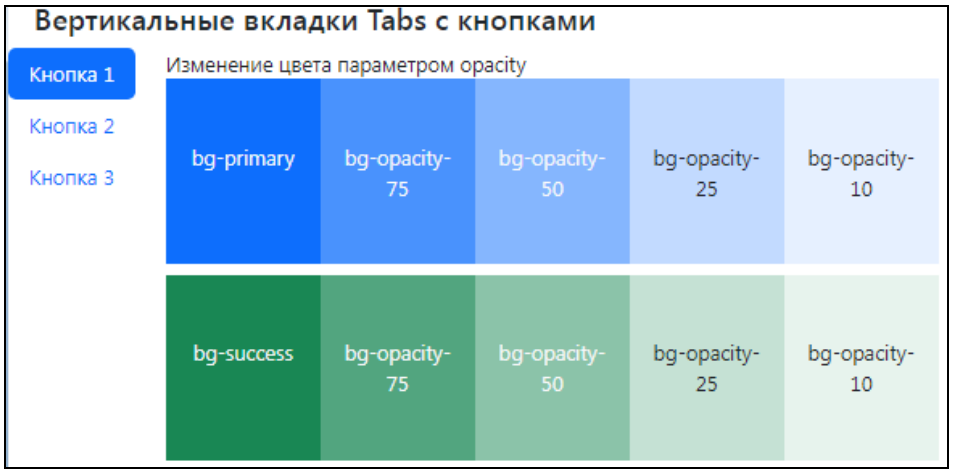


Рис. 2.17. Страница tabs_4.html с открытой по кнопке **Кнопка 1** вкладкой



Рис. 2.18. Страница tabs_4.html с открытой по кнопке **Кнопка 2** вкладкой

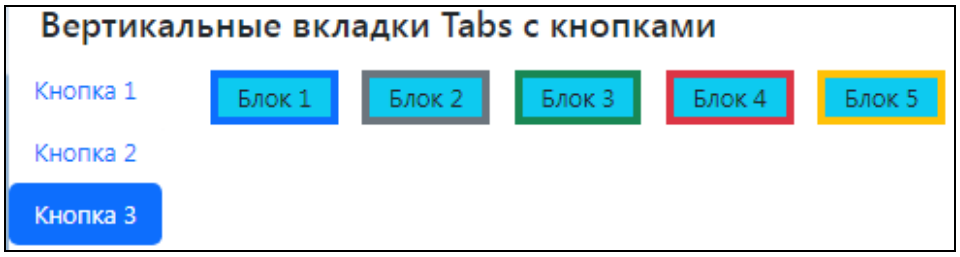


Рис. 2.19. Страница tabs_4.html с открытой по кнопке **Кнопка 3** вкладкой

2.5. Компонент Bootstrap *Ofcanvas*

В Bootstrap 5 появился еще один новый компонент — Offcanvas, который позволяет создавать выплывающую панель. Такая панель обычно используется на веб-сайтах для отображения меню на небольших устройствах. Выплывающую панель можно поместить на любую сторону страницы, используя следующие параметры:

- offcanvas-start — панель выплывает слева;
- offcanvas-top — панель выплывает сверху;
- offcanvas-end — панель выплывает справа;
- offcanvas-bottom — панель выплывает снизу.

В программном коде также можно указать, должен элемент offcanvas иметь фон или нет. Пример HTML-кода страницы, в которой показано создание элемента offcanvas, приведен в листинге 2.13 (страница offcanvas_1.html).

Листинг 2.13. Страница offcanvas_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница offcanvas_1.html**.

В этом программном коде создаns четыре выплывающие панели:

- `class="offcanvas offcanvas-start"` — левая панель с размером по умолчанию;
- `class="offcanvas offcanvas-top h-50"` — верхняя панель с высотой 50% экрана;
- `class="offcanvas offcanvas-end w-50"` — правая панель с шириной 50% экрана;
- `class="offcanvas offcanvas-bottom h-50"` — нижняя панель с высотой 50% экрана.

Каждая панель имеет кнопку для ее открытия (`button`). Сама панель состоит из двух блоков: блока заголовка панели (`class="offcanvas-header"`) с меткой или текстовой надписью (`class="offcanvas-title"`) и кнопкой закрытия панели в виде крестика (`class="btn-close text-reset"`) и блока тела панели, в который может быть вставлен любой контент. В приведеннном примере в качестве контента использовано изображение AP_400.png. Для того чтобы изображение адаптировалось к размеру экрана, к нему добавлен класс `class="img-fluid" alt="GFG"`.

В итоге в окне браузера страница offcanvas_1.html будет будет выглядеть так, как показано на рис. 2.20–2.24.

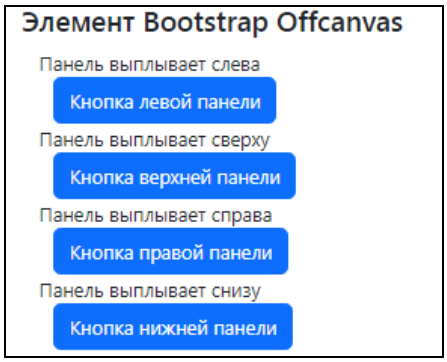


Рис. 2.20. Страница offcanvas_1.html с кнопками для открытия выплывающих панелей

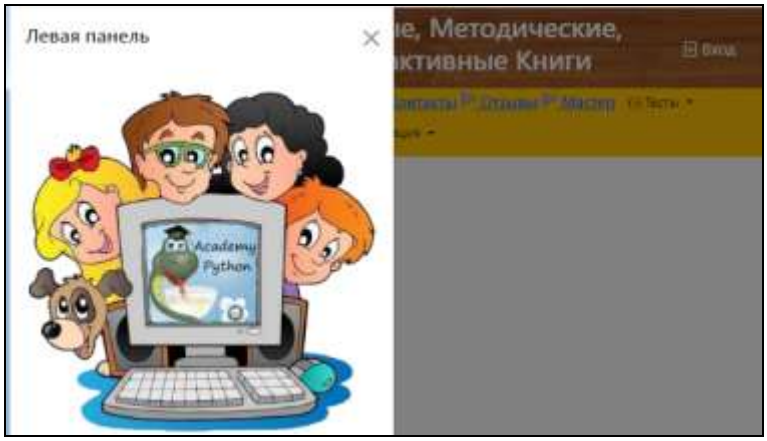


Рис. 2.21. Страница offcanvas_1.html с открытой левой боковой панелью

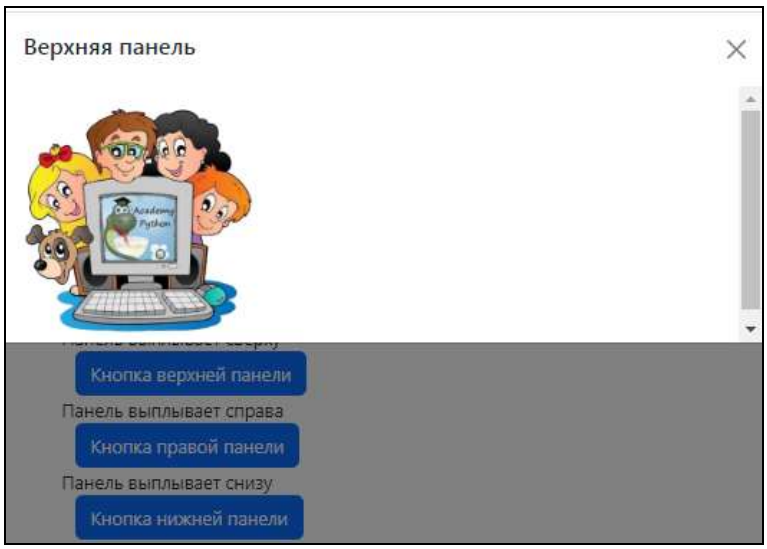


Рис. 2.22. Страница offcanvas_1.html с открытой верхней панелью

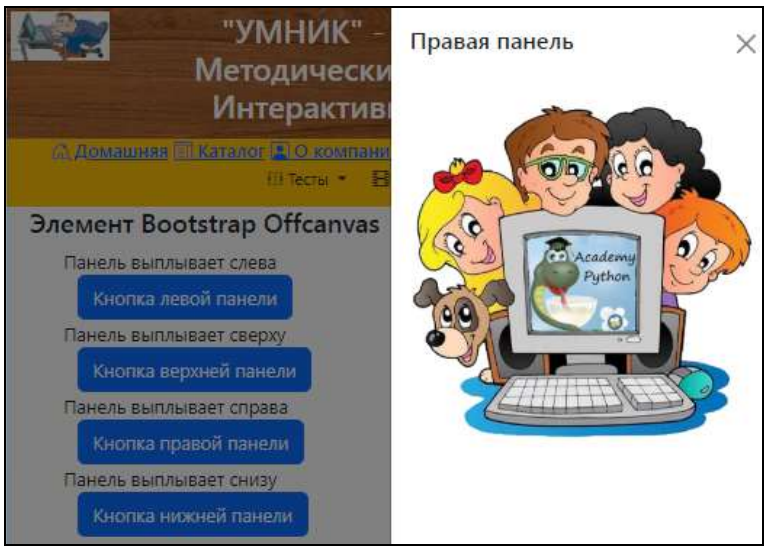


Рис. 2.23. Страница offcanvas_1.html с открытой правой панелью



Рис. 2.24. Страница offcanvas_1.html с открытой нижней панелью

Для закрытия выплывающей панели необходимо нажать на кнопку с крестиком в правом верхнем углу панели либо щелкнуть левой кнопкой мыши в любом месте основной страницы.

2.6. Компонент для выдачи подсказок *Tooltip*

Подсказки `Tooltip` отображают текстовые подсказки рядом с рассматриваемым элементом, когда пользователи наводят на элемент курсор, фокусируются на нем или нажимают его. Один из способов инициализировать всплывающие подсказки для элемента на странице — добавить к нему атрибут `data-bs-toggle`.

Пример HTML-кода страницы, в которой показано создание подсказок `Tooltip`, приведен в листинге 2.14 (страница tooltip.html).

Листинг 2.14. Страница tooltip.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница tooltip.html**.

В этом программном модуле для элементов `кнопка` и `ссылка` добавлен атрибут `data-toggle="tooltip"`, а текст подсказки задан атрибутом `title`. В итоге в окне браузера страница tooltip.html будет выглядеть так, как показано на рис. 2.25.

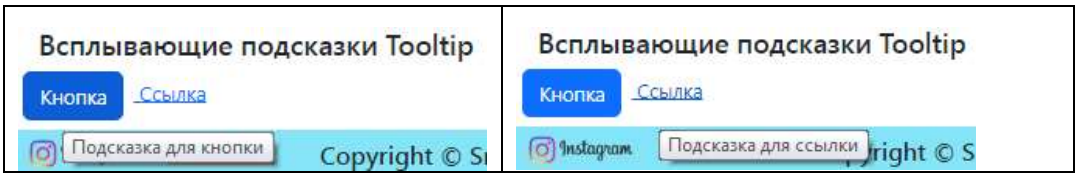


Рис. 2.25. Кнопка и ссылка с всплывающими подсказками при наведении на них курсора мыши

2.7. Использование атрибута *href* в шаблонах Django для создания ссылок на URL-адреса

Атрибут `href` обычно используется в HTML для определения гиперссылки на другую веб-страницу (в сочетании с тегом `<a></a>`). Его можно задействовать для предоставления навигации по веб-сайту или для ссылки на внешние источники (для создания ссылок перехода на страницы других сайтов). Однако применение атрибута `href` в шаблонах Django может привести к проблемам, если с ним не обращаться должным образом. В этом разделе мы рассмотрим несколько методов использования атрибута `href` в шаблонах Django (с текстовыми ссылками и кнопками Bootstrap).

2.7.1. Ссылки на внешние URL-адреса

Кнопка Bootstrap или текстовая ссылка могут ссылаться на внешние URL-адреса, предоставляя пользователям возможность переходить на другие веб-сайты или ресурсы за пределами вашего приложения. Для этого нужно использовать абсолютный URL-адрес внешнего ресурса в качестве значения `href` в теге `<a>`, который обертывает вашу кнопку или текстовую ссылку. Пример HTML-кода страницы, в которой показано создание ссылок на внешние URL-адреса, приведен в листинге 2.15 (страница link_1.html).

Листинг 2.15. Страница link_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница link_1.html**.

Внимание! Новые страницы откроются на той же вкладке браузера, поэтому для возвращения к цифровой книге воспользуйтесь стрелкой возврата на прежнюю страницу.

В этом программном модуле элементы `кнопка` и `ссылка` содержат ссылку на внешний сайт прогноза погоды, который имеет абсолютный URL-адрес **https://www.gismeteo.ru/**. В итоге в окне браузера страница link_1.html будет выглядеть так, как показано на рис. 2.26.

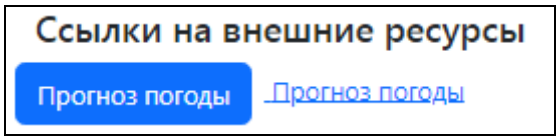


Рис. 2.26. Элементы интерфейса со ссылками на внешние ресурсы

По нажатию на кнопку **Прогноз погоды** или соответствующую текстовую ссылку пользователь будет перенаправлен на внешний сайт с абсолютным адресом **https://www.gismeteo.ru/**, при этом браузер отобразит страницу этого сайта, фрагмент которой представлен на рис. 2.27.



Рис. 2.27. Фрагмент страницы сайта **www.gismeteo.ru**

Достаточно часто требуется, чтобы внешний ресурс открывался не в текущей, а в новой вкладке браузера. Для этого нужно использовать ссылку с атрибутом `target="_blank"`. Пример HTML-кода страницы, в которой показано создание ссылок на внешние URL-адреса с переходом на новую вкладку браузера, приведен в листинге 2.16 (страница link_2.html).

Листинг 2.16. Страница link_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница link_2.html**.

По нажатию на кнопку **Прогноз погоды** или соответствующую текстовую ссылку пользователь будет перенаправлен на внешний сайт с абсолютным адресом **https://www.gismeteo.ru/**, при этом при этом браузер отобразит страницу этого сайта в новой вкладке.

2.7.2. Ссылки на внутренние URL-адреса

Кнопка Bootstrap или текстовая ссылка могут ссылаться и на внутренние URL-адреса сайта, предоставляя пользователям возможность переходить на другие веб-страницы или ресурсы в пределах вашего приложения. Для этого нужно использовать URL-маршрут внутреннего ресурса в качестве значения `href` в теге `<a>`, который обертывает вашу кнопку или текстовую ссылку. Пример HTML-кода страницы, в которой показано создание ссылок на внутренние URL-адреса, приведен в листинге 2.17 (страница link_3.html).

Листинг 2.17. Страница link_3.html

Примечание.

В демо-версии книги удалены листинги программ.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница link_3.html**.

В этом программном модуле элементы `кнопка` и `ссылка` содержат ссылку на маршрут с именем `icon_1` приложения `master`. В приложении `master` в файле urls.py нужно создать этот маршрут `icon_1` с использованием кода, приведенного в листинге 2.18.

Листинг 2.18. Маршрут icon_1 в файле urls.py приложения master

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `view` с именем `icon_1` — в листинге 2.19 приведен его код для простейшего случая.

Листинг 2.19. Представление icon_1 в файле view.py приложения master

Примечание.

В демо-версии книги удалены листинги программ.

Из этого представления будет отображен шаблон с именем icon_1.html. Мы уже использовали этот шаблон в предыдущих разделах — он имеет следующий код (листинг 2.20).

Листинг 2.20. Шаблон icon_1.html приложения master

Примечание.

В демо-версии книги удалены листинги программ.

В итоге в окне браузера страница link_3.html будет выглядеть так, как показано на рис. 2.28.

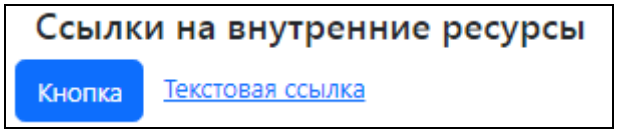


Рис. 2.28. Элементы интерфейса со ссылками на внутренние ресурсы

По нажатию на кнопку или текстовую ссылку браузер отобразит страницу icon_1.html этого сайта, представленную на рис. 2.29.

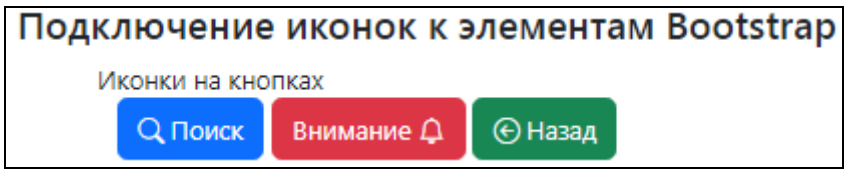


Рис. 2.29. Страница текущего сайта icon_1.html

2.7.3. Ссылки на внутренние динамические URL-адреса

В приложениях Django достаточно часто необходимо использовать динамические URL-адреса — например, для показа на сайте интернет-магазина информации о товарах, сведения о которых хранятся в базе данных. Количество товаров в БД может насчитывать десятки тысяч, и это количество может постоянно меняться. В этом случае невозможно обойтись без динамических URL-адресов.

Динамический URL-адрес состоит из двух элементов: постоянного адреса и переменного параметра. В качестве переменного параметра может выступать идентификатор товара в БД.

Пример HTML-кода страницы, в которой показано создание ссылок на внутренние динамические URL-адреса, приведен в листинге 2.21 (страница link_4.html).

Листинг 2.21. Страница link_4.html приложения master

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница link_4.html**.

В этом программном модуле созданы два элемента `кнопка` и один элемент `ссылка`, которые содержат атрибут `href` со ссылкой на маршрут с именем `book_web` приложения `shop`. Здесь после имени маршрута указан еще один параметр — идентификатор книги в базе данных (`5`), причем для кнопок этот параметр задан двумя синтаксическими вариантами:

```
href="{% url 'shop:book_web' id=5 %}"
href="/book_web/5"
```

> ***Совет***
> Если в вашем проекте несколько приложений, то для адресации надежней использовать синтаксис `{% url '<имя_приложения>:<имя_маршрута>' id=<id_объекта> %}`. В этом случае гарантированно не будет коллизий, и пользователь будет направлен на нужную страницу.

В приложении `shop` в файле urls.py нужно создать этот маршрут с использованием следующего кода (листинг 2.22).

Листинг 2.22. Маршрут book_web в файле urls.py приложения shop

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

Этот маршрут принимает параметр «идентификатор книги в БД» как целое число (`<int:id>`), которое будет передано в представление `views.book_web`.

В приложении `shop` в файле view.py нужно создать представление с использованием следующего кода (листинг 2.23).

Листинг 2.23. Представление book_web в файле view.py приложения shop

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно большой программный модуль, и мы не станем рассматривать его структуру, так как цель раздела не в этом, а остановимся только на деталях, связанных с динамической адресацией.

Созданное представление принимает идентификатор книги (`id`) в базе данных, переданный из маршрута `def book_web(request, id)`. Затем в объект `book` из БД загружается информация о книге: `book = get_object_or_404(Products, id=id)`. Объект `book` через словарь `content` вместе с другими параметрами передается в шаблон `book/book_web.htm`. В этом шаблоне и будет выведена информация о книге, которая в БД имеет идентификатор `id=5`.

Примечание

Здесь не приводится код этого шаблона, поскольку он достаточно большой (это один из элементов сайта **rumnik.ru**), однако будет показан внешний вид этого шаблона в окне браузера, полученный при выполнении этого примера.

Итак, мы привели код цепочки всех модулей, которые участвуют в динамической адресации. В итоге в окне браузера страница link_4.html будет выглядеть так, как показано на рис. 2.30.

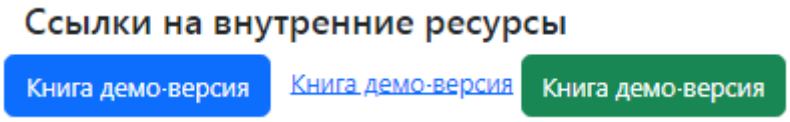


Рис. 2.30. Страница текущего сайта link_4.html

На этой странице имеются две кнопки и текстовая ссылка, которые содержат динамический адрес внутреннего ресурса текущего приложения. По нажатию на эти кнопки или при переходе по текстовой ссылке пользователь будет перенаправлен на страницу сайта `book_web.htm/5`, которая откроется в новой вкладке браузера (рис.2.31).

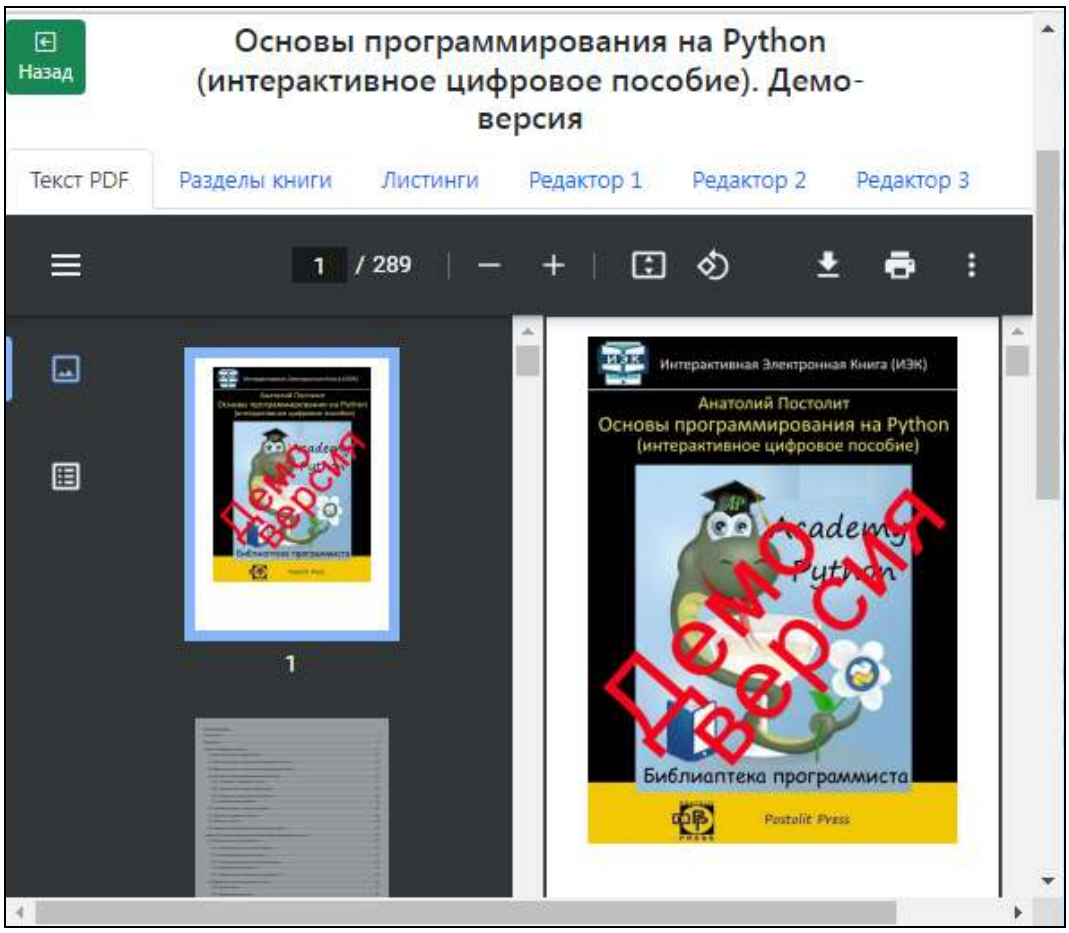


Рис. 2.31. Страница текущего сайта с динамическим адресом `book_web.htm/5`

Если по всей цепочке представленных здесь модулей задать другой идентификатор объекта в базе данных, то на конечной HTML-странице будет выдана информация именно о нем. В этом и заключается сущность динамической адресации, когда с использованием неизменного программного кода пользователь может получить информацию о совершенно разных объектах.

2.7.4. Использование атрибута *href* для обмена параметрами между шаблонами и представлениями

Достаточно часто возникают ситуации, когда из шаблона в представление надо передать идентификатор нажатой кнопки, а из представления в шаблон вернуть некое сообщение. Например, в шаблоне удаления информации из БД имеются две кнопки: **Удалить** и **Отмена**. В представлении нужно определить — какая кнопка была нажата, выполнить соответствующие действия и в шаблон вернуть сообщение о результатах этих действий. Попробуем проиграть эту ситуацию на упрощенном примере: создадим модули для вывода информации из БД и для удаления одной из записей. Для этого понадобится создать несколько URL-маршрутов, представлений и шаблонов.

Для начала на одной из HTML-страниц создадим ссылку для обращения к маршруту, который обеспечит вызов модуля для редактирования объектов в БД. А в приложении `master` в одном из шаблонов создадим ссылку на маршрут с именем `edit_products` с использованием следующего кода (листинг 2.24):

Листинг 2.24. Ссылка на маршрут edit_products в шаблоне приложения master

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке в файле urls.py приложения `master` будет активирован следующий URL-маршрут (листинг 2.25).

Листинг 2.25. Маршрут edit_products в файле urls.py приложения master

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `edit_products` в приложении `master`, имеющее следующий код (листинг 2.26):

Листинг 2.26. Представление edit_products в файле view.py приложения master

Примечание.
В демо-версии книги удалены листинги программ.

В этом модуле в объект `products` из БД загружаются все записи о продуктах (в нашем случае продуктами являются цифровые книги). Для работы этого модуля и модуля удаления продуктов из БД нужно импортировать в файл view.py некоторые библиотеки:

```
from shop.models import Products
from django.shortcuts import render, get_object_or_404, redirect
from django.urls import reverse
from django.contrib import messages
import time
```

Через словарь `context` информация о продуктах передается в шаблон `edit_products.html`, имеющий следующий код (листинг 2.27).

Листинг 2.27. Шаблон edit_products.html приложения master

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне создана таблица с данными о цифровых книгах, взятыми из БД. Здесь следует обратить внимание на блок кода в теге `{% if messages %}`, который обеспечивает вывод сообщений, переданных из представления.

В последней колонке таблицы находится кнопка `Удалить`, по нажатию на которую будет активирован маршрут с именем `product_del` приложения `master`. Этому маршруту будет передан идентификатор удаляемой записи (`id=object.id`). Сам маршрут имеет следующий код:

```
path('glava_2/del_product' /<int:id>', views.del_product, name=del_product'),
```

По этому маршруту будет активировано представление `product_del` в приложении `master`, имеющее следующий код (листинг 2.28):

Листинг 2.28. Представление product_del в файле view.py приложения master

Примечание.
В демо-версии книги удалены листинги программ.

Примечание
В реальном приложении нужно снять комментарий со строки:

```
Products.objects.get(id=id).delete()
```

Это представление принимает идентификатор удаляемой записи (`id`) и загружает в объект `obj` информацию из БД об удаляемом объекте. Имя удаляемого объекта (`del_prod = obj.name`) через словарь `content` передается в шаблон master/del_yesno.html, имеющий следующий код (листинг 2.29).

Листинг 2.29. Шаблон del_yesno.html приложения master

Примечание.

В демо-версии книги удалены листинги программ.

Этот шаблон представляет собой форму `<form action="" method="post">`, в которой выводится имя удаляемого объекта `{{ del_prod }}`, и имеются три кнопки:

- `Отмена` — кнопка с именем `name="no"` для отмены удаления (типовая кнопка Bootstrap);
- `Удалить` — простая кнопка для удаления типа `submit` с именем `name="yes"` для подтверждения удаления;
- `Удалить` — кнопка для удаления с крутящимся спином типа `submit` с именем `name="yes"` для подтверждения удаления.

Последняя кнопка приведена в качестве примера того, как можно отобразить ожидание завершения длительного процесса. К этой кнопке добавлен скрипт, который покажет на кнопке в процессе удаления записи из БД вращающийся спин.

По нажатию на любую из кнопок выполняется возврат к представлению `product_del` с методом `POST`, а в представление передается имя нажатой кнопки. В этом представлении в теле блока `if request.method == "POST"` анализируется имя нажатой кнопки. Если из формы получено имя нажатой кнопки `yes` (`if 'yes' in request.POST`), значит, была нажата кнопка `Удалить`, и будет выполнена процедура удаления записи из БД:

```
# Products.objects.get(id=id).delete()
```

В приведенном здесь примере эта строка закомментирована для предотвращения удаления информации из БД сайта **rumnik.ru**. При этом добавлена строка `time.leep(3)` (засыпание приложения на 3 секунды) — для имитации длительного удаления записи из БД, чтобы показать, как работает кнопка с вращающимся спином. Здесь же формируется текст сообщения об успешном удалении записи, которое будет передано в шаблон `edit_product`:

```
messages.success(request, 'Продукт удален из БД')
```

Если из формы получено имя нажатой кнопки `no`, значит, была нажата кнопка `Отмена`, и команда удаления записи из БД будет пропущена. Здесь же формируется текст сообщения об отмене удаления записи, которое будет передано в шаблон `edit_product`:

```
messages.success(request, 'Удаление продукта отменено')
```

Эти сообщения будут возвращены в шаблон `edit_products.html` и отображены там в теге `{% if messages %}`. В этом теге имеется кнопка с именем `Скрыть сообщение`, по нажатии на которую произойдет обновление шаблона `edit_products`, и текст сообщения исчезнет с экрана.

Итак, мы рассмотрели код группы модулей, в которых реализован обмен параметрами между шаблонами и представлениями. Для запуска в работу этой группы модулей надо в рамках интерактивной цифровой платформы активировать следующую ссылку:

```
<a href="{% url 'master:edit_products' %}" target ="_blank">Представление ed-
it_products</a>
```

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Представление edit_products**.

По этой ссылке будет показан шаблон edit_products.html (рис. 2.32).



Рис. 2.32. Внешний вид шаблона edit_products.html

По нажатию на кнопку **Удалить** в одной из строк таблицы с данными о цифровых книгах (например, в строке с номером 1), будет выполнен переход к шаблону product_del.html с передачей в него идентификатора текущей записи (рис. 2.33).

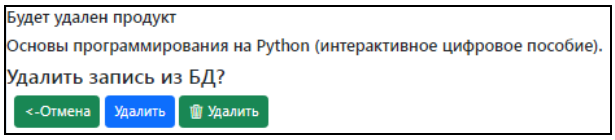


Рис. 2.33. Внешний вид шаблона product_del.html

Здесь мы видим имя удаляемого объекта — цифровая книга «Основы программирования на Python (интерактивное цифровое пособие)» и три кнопки: **Отмена** (зеленого цвета), **Удалить** (синего цвета) и **Удалить** (зеленого цвета).

По нажатию на кнопку **Отмена** в представление `product_del` будет передан идентификатор удаляемой записи и имя нажатой кнопки `no`, после чего система инструкцию удаления записи пропустит и сформирует сообщение с текстом **Удаление продукта отменено**, которое через представление `edit_products` будет передано в шаблон edit_products.html и показано в нижней части этого шаблона (рис. 2.34). По нажатию на кнопку **Скрыть сообщение** оно будет удалено с экрана.

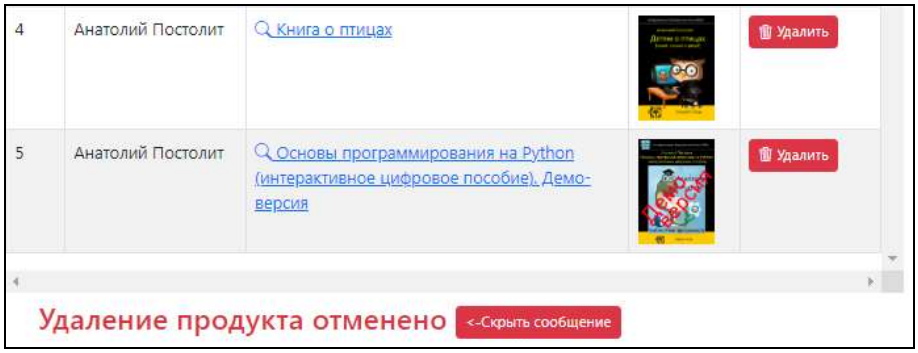


Рис. 2.34. Внешний вид фрагмента шаблона edit_ products.html с тестом сообщения

По нажатию на кнопку **Удалить** (синего цвета) в представление `product_del` будет передан идентификатор текущей записи и имя нажатой кнопки `yes`, после чего выполнится инструкция удаления текущей записи и сформируется сообщение с текстом **Продукт удален из БД**, которое через представление `edit_products` будет передано в шаблон edit_products.html и показано в нижней части этого шаблона (рис. 2.35).

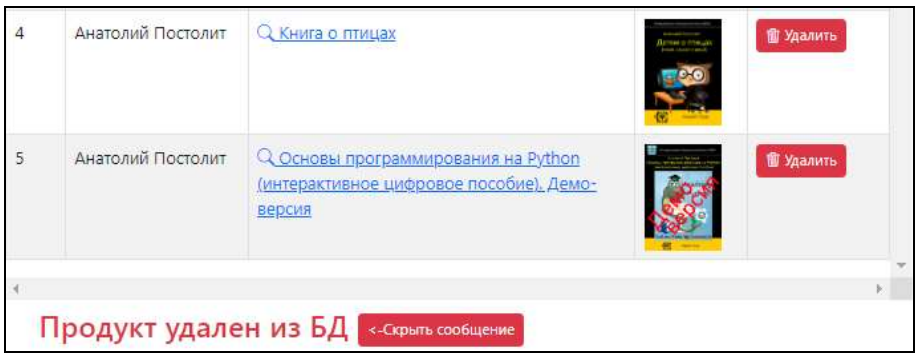


Рис. 2.35. Внешний вид фрагмента шаблона edit_ products.html с тестом сообщения

Эта страница появится с задержкой 3 сек., которая искусственно создана в инструкции `time.sleep(3)`. По нажатию на кнопку **Скрыть сообщение** оно будет удалено с экрана.

По нажатию на кнопку **Удалить** (зеленого цвета) в представление `product_del` будет передан идентификатор текущей записи и имя нажатой кнопки `yes`, после чего выполнится инструкция удаления текущей записи и сформируется сообщение с текстом **Продукт удален из БД**, которое через представление `edit_products` будет передано в шаблон edit_products.html. При этом в процессе удаления записи из БД текст на кнопке **Удалить** будет заменен вращающимся индикатором (рис. 2.36).

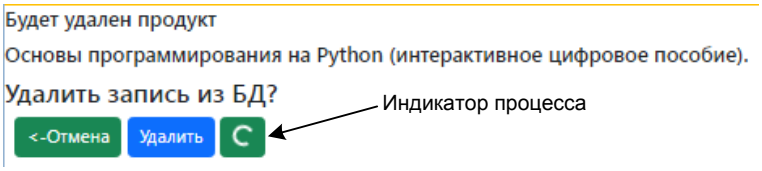


Рис. 2.36. Вращающийся спин для индикации процесса удаления записи из БД

Индикатор на кнопке будет отображаться до тех пор, пока выполняется процесс удаления записи, причем появляться он станет лишь в том случае, если база данных достаточно большая, и процесс удаления записи из нее занимает продолжительное время (как вы знаете, в нашем примере для показа этого индикатора введена искусственная задержка процесса удаления записи на 3 сек. с помощью инструкции `time.sleep(3)`). После удаления текущей записи и сформируется сообщение с текстом **Продукт удален из БД**, которое через представление `edit_products` будет передано в шаблон edit_products.html и показано в нижней части этого шаблона.

* * *

Итак, мы с вами рассмотрели возможности Django по передаче из шаблонов в представления (через параметры) имен нажатых кнопок и идентификаторов объектов, а также по передаче сообщений из представлений в шаблоны.

2.8. Всплывающие модальные окна

Всплывающее *модальное окно* является частью текущей страницы. Когда происходит вызов модального окна, то текущая страница остается на экране, а модальное окно появляется поверх текущей страницы, при этом текущая страница частично затеняется, и все ее управляющие элементы (меню, кнопки и т. п.) блокируются. Текущее окно остается заблокированным до тех пор, пока пользователь не выполнит определенные действия в модальном окне или не закроет его. Модальное окно создается с помощью Bootstrap, причем ему не нужны дополнительные скрипты. Принципиальная схема взаимодействия между представлением `view` (`boot_modal`), загруженным из представления шаблоном (страница boot_modal.html) и модальными окнами (modal_simple.html, modal_scroll.html) приведена на рис. 2.37.

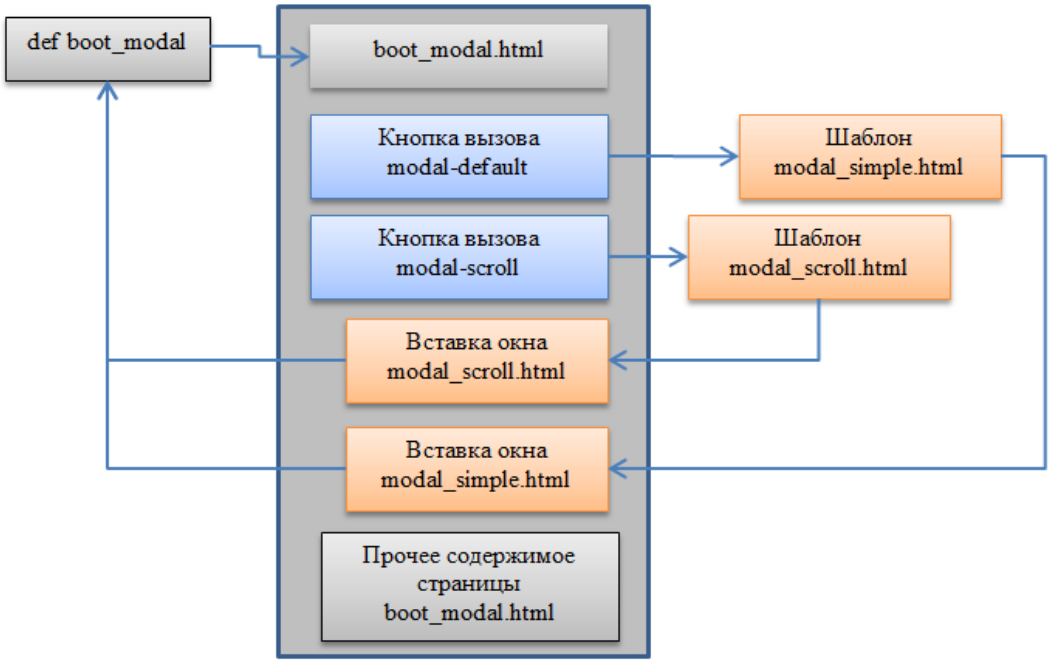


Рис. 2.37. Всплывающие модальные окна Bootstrap

Как здесь можно видеть, представление `def boot_modal` вызывает шаблон boot_modal.html. В этом шаблоне, кроме основного содержания, имеются две кнопки для вызова модальных окон, и два тега, предусматривающие всплытие модальных окон поверх основной страницы. Когда пользователь нажимает одну из кнопок, то на фоне основной страницы всплывает модальное окно. При этом в то место, где предусмотрена вставка модальных окон, производится вставка содержимого одного из шаблонов (либо modal_simple.html, либо modal_scroll.html). После закрытия модального окна происходит возврат к содержимому базовой страницы.

Программный код модальных окон может быть расположен непосредственно в теле шаблона boot_modal.html, либо вынесен в отдельные файлы. Вынос в отдельные файлы предпочтительней, так как, во-первых, упрощается код основной страницы, во-вторых, шаблоны модальных окон могут быть многократно использованы другими страницами сайта.

Для модальных окон Bootstrap можно задать ширину окна —это определяется классом-модификатором в элементе `modal-dialog`. Значения этого параметра классов-модификаторов `modal-dialog` приведены в табл. 2.1.

Таблица 2.1. *Значения параметра классов-модификаторов* `modal-dialog` *для модальных окон разной ширины*

Размер	**Класс-модификатор**	**Максимальная ширина (max-width)**
Маленький (Small)	`.modal-sm`	300px
Стандартный (Default)	None	500px
Большой (Large)	`.modal-lg`	800px
Очень большой (Extra large)	`.modal-xl`	1140px

Для демонстрации работы модальных окон создадим группу модулей. Ссылка, которая обеспечит открытие модальных окон, может находиться в одном из шаблонов или в одном из элементов меню сайта и иметь следующий код (листинг 2.30).

Листинг 2.30. Ссылка на доступ к представлению boot_modal приложения master

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке активируется следующий URL-маршрут (листинг 2.31).

Листинг 2.31. Маршрут с именем boot_modal приложения master

Примечание.
В демо-версии книги удалены листинги программ.

Этот маршрут обращается к представлению `boot_modal`, которое имеет следующий код (листинг 2.32).

Листинг 2.32. Представление (view) boot_modal приложения master

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении сформировано текстовое сообщение, которое через словарь `content` передается в шаблон boot_modal.html, имеющий следующий код (листинг 2.33).

Листинг 2.33. Шаблон boot_modal.html приложения master

Примечание.
В демо-версии книги удалены листинги программ.

Основная страница boot_modal.html, в которой предусмотрена возможность показа модальных окон, будет выглядеть так, как показано на рис. 2.38.

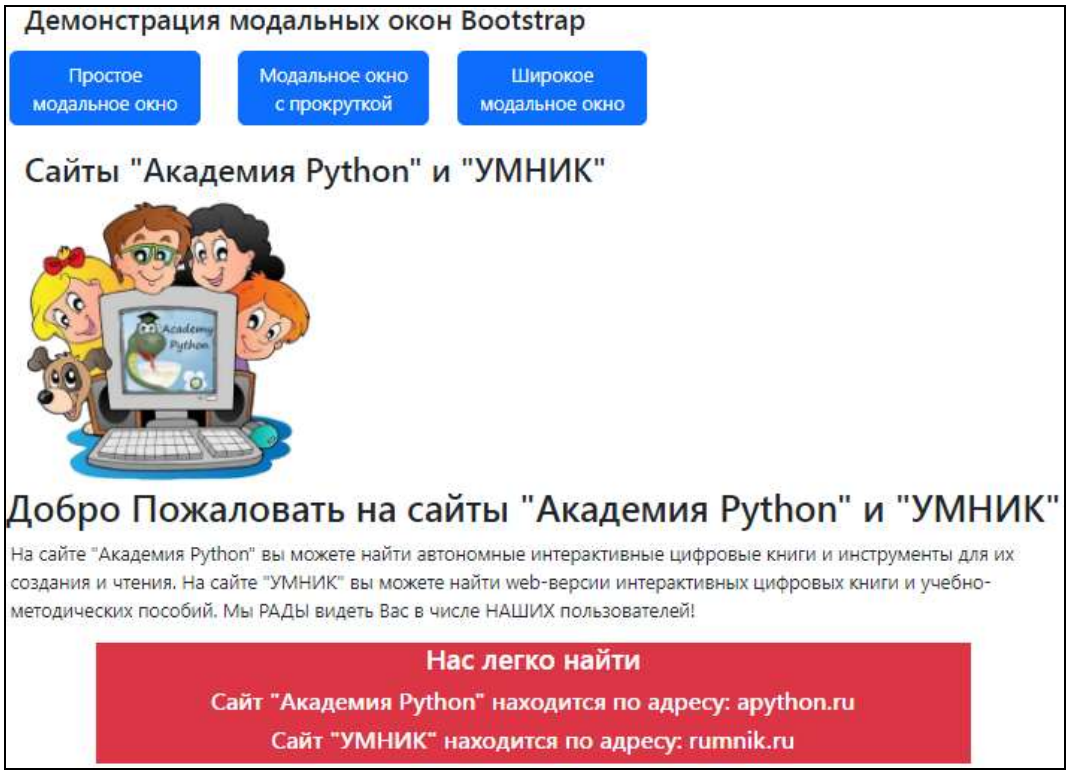


Рис. 2.38. Внешний вид шаблона boot_modal.html

В шаблоне boot_modal.html имеются три кнопки:

- для вызова простого модального окна;
- для вызова модального окна с полосой прокрутки;

- для вызова широкого модального окна.

2.8.1. Простое модальное окно

Простое модальное окно вызывается по нажатию на соответствующую кнопку с использованием следующего кода:

```
<button type="button" class="btn btn-block btn-primary mb-3"    data-bs-
toggle="modal"    data-bs-target="#modal-default">Простое модальное
окно</button>
```

Здесь целевое модальное окно идентифицируется с использованием следующего фрагмента кода:

```
data-bs-target="#modal-default"
```

Место вставки модального окна резервируется с использованием следующего фрагмента кода:

```
<!-- Это встроенное простое модальное окно -->{% include "mas-
ter/glava_2/modal_simple.html" %}
```

Само модальное окно находится в шаблоне modal_simpl.html, который имеет следующий код (листинг 2.34).

Листинг 2.34. Шаблон modal_simpl.html приложения master

Примечание.

В демо-версии книги удалены листинги программ.

Здесь в первой строке присутствует идентификатор модального окна (`id="modal-default"`), который совпадает с целевым именем окна в кнопке вызова (`data-bs-target="#modal-default"`).

Таким образом, по нажатию на кнопку **Простое модальное окно** соответствующее модальное окно будет отображено поверх текущей страницы (рис. 2.39).

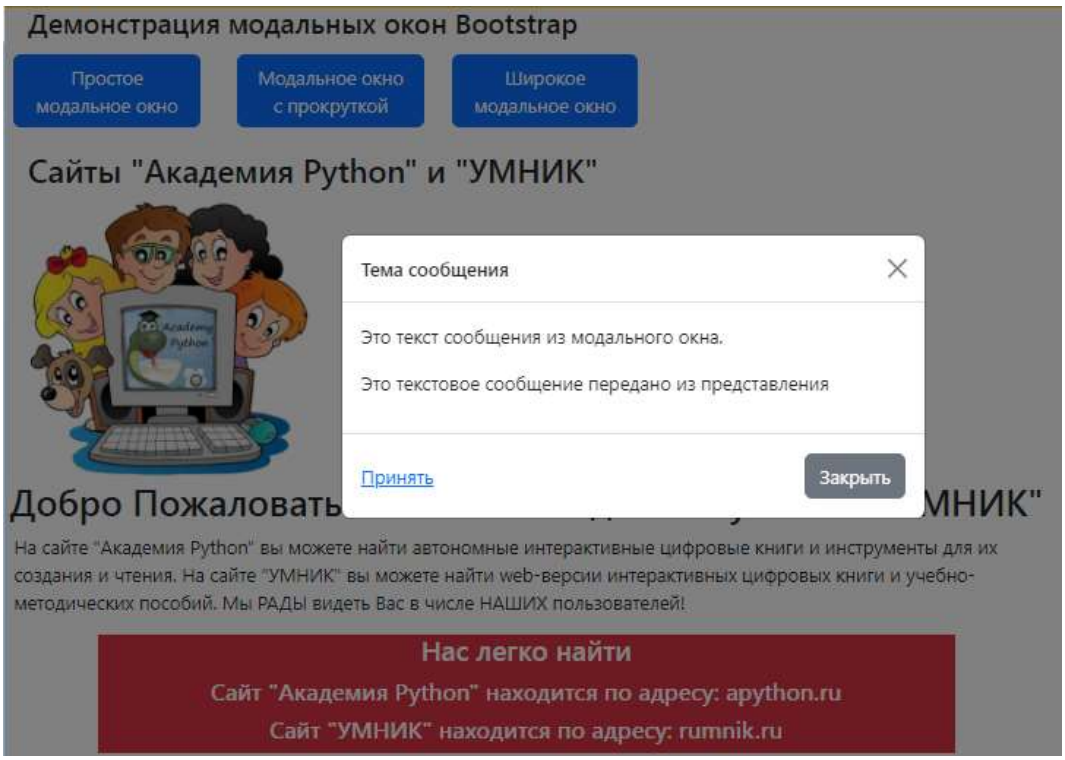


Рис. 2.39. Простое модальное окно на фоне шаблона boot_modal.html

Для закрытия модального окна нужно нажать на кнопку **Закрыть**, на кнопку с крестиком в правом верхнем углу модального окна или щелкнуть левой кнопкой мыши в любом месте родительского окна.

2.8.2. Модальное окно с полосой прокрутки

Модальное окно с полосой прокрутки вызывается по нажатию на соответствующую кнопку с использованием следующего кода:

```
<button type="button" class="btn btn-block btn-primary mb-3"    data-bs-
toggle="modal"    data-bs-target="#modal-scroll">Модальное окно с
прокруткой</button>
```

Здесь целевое модальное окно идентифицируется с использованием следующего фрагмента кода:

```
data-bs-target="#modal-scroll"
```

Место вставки модального окна резервируется с использованием следующего фрагмента кода:

```
<!-- Это встроенное модальное окно с прокруткой -->
{% include "master/glava_2/modal_scroll.html" %}
```

Само модальное окно находится в шаблоне modal_scroll.html, который имеет следующий код (листинг 2.35).

Листинг 2.35. Шаблон modal_scroll.html приложения master

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

Здесь в первой строке присутствует идентификатор модального окна (`id="modal-scroll"`), который совпадает с целевым именем окна в кнопке вызова (`data-bs-target="#modal-scroll "`).

Таким образом, по нажатию на кнопку **Модальное окно с полосой прокрутки** соответствующее модальное окно будет отображено поверх текущей страницы (рис. 2.40).

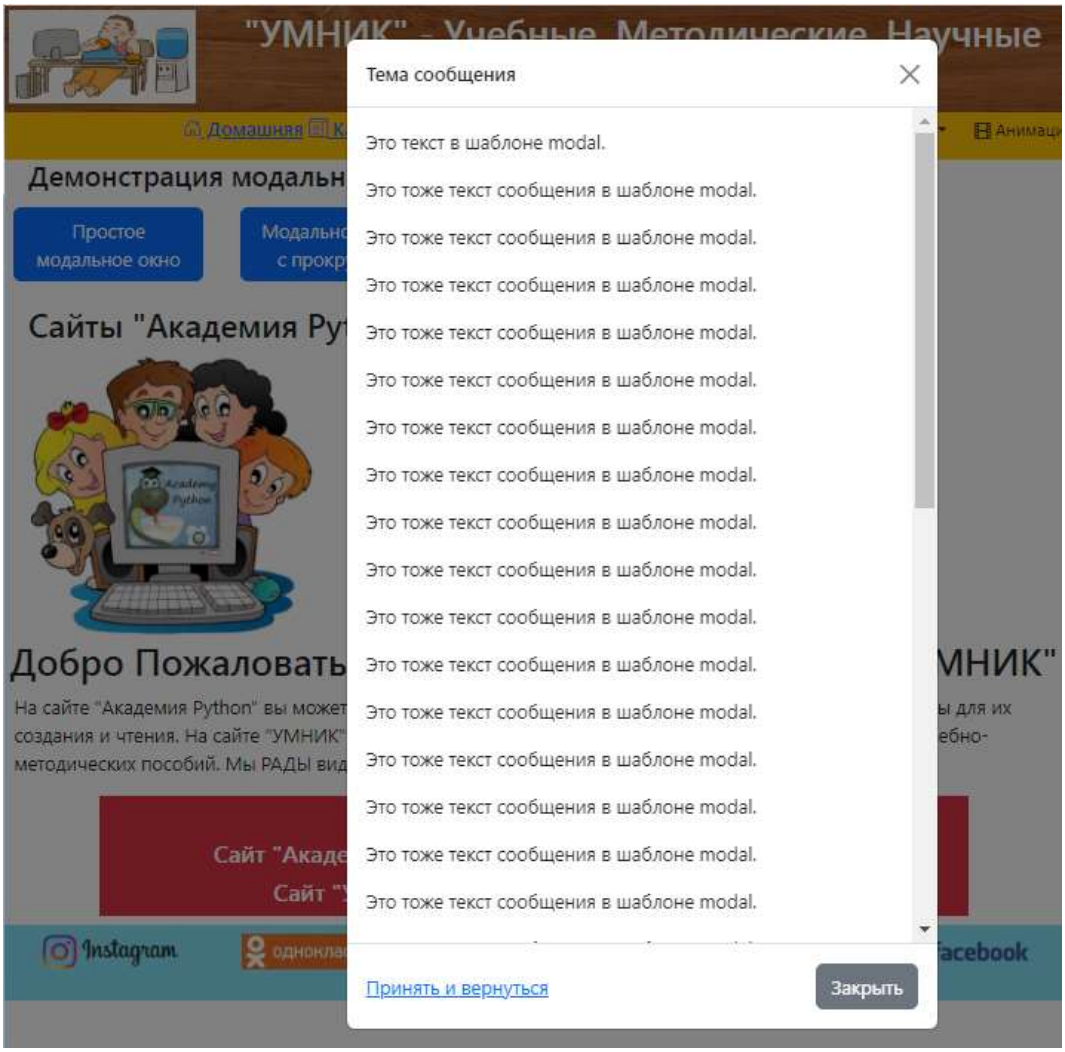


Рис. 2.40. Модальное окно с полосой прокрутки на фоне шаблона boot_modal.html

Для закрытия модального окна нужно нажать на кнопку **Закрыть**, на кнопку с крестиком в правом верхнем углу модального окна или щелкнуть левой кнопкой мыши в любом месте родительского окна.

2.8.3. Широкое модальное окно

Широкое модальное окно вызывается по нажатию на соответствующую кнопку с использованием следующего кода:

```
<button type="button" class="btn btn-block btn-primary mb-3"
              data-bs-toggle="modal"
              data-bs-target="#modal-large">Широкое модальное окно
</button>
```

Здесь целевое модальное окно идентифицируется с использованием следующего фрагмента кода:

```
data-bs-target="#modal-large"
```

Место вставки модального окна резервируется с использованием следующего фрагмента кода:

```
<!-- Это широкое модальное окно -->
{% include "master/glava_2/modal_large.html" %}
```

Само модальное окно находится в шаблоне modal_large.html, который имеет следующий код (листинг 2.36).

Листинг 2.36. Шаблон modal_large.html приложения master

Примечание.

В демо-версии книги удалены листинги программ.

Здесь в первой строке присутствует идентификатор модального окна (`id="modal-large"`), который совпадает с целевым именем окна в кнопке вызова (`data-bs-target="#modal-large"`). Размер модального окна определяется с использованием следующего фрагмента кода:

```
<div class="modal-dialog modal-dialog-centered modal-xl" role="document"
```

Таким образом, по нажатию на кнопку **Широкое модальное окно** соответствующее модальное окно будет отображено поверх текущей страницы (рис. 2.41).

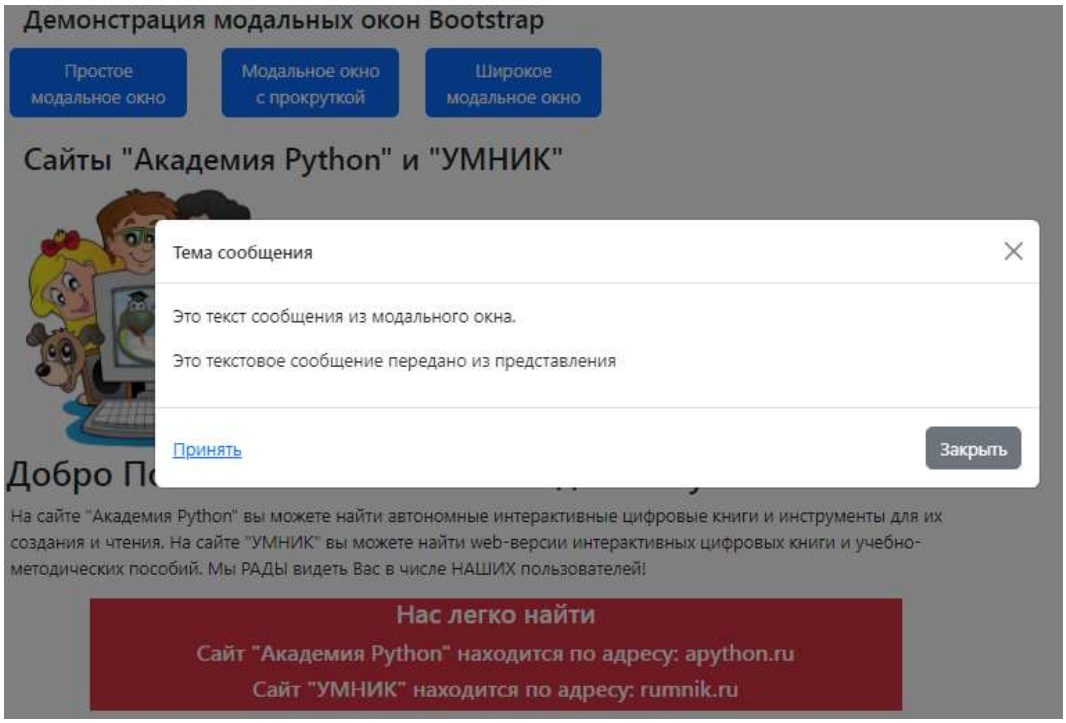


Рис. 2.41. Широкое модальное окно с полосой прокрутки на фоне шаблона boot_modal.html

Для закрытия модального окна нужно нажать на кнопку **Закрыть**, на кнопку с крестиком в правом верхнем углу модального окна или щелкнуть левой кнопкой мыши в любом месте родительского окна.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Представление boot_modal**.

2.9. Компонент Bootstrap *Accordion* «Аккордеон»

Компонент `Accordion` формирует содержимое фрагмента HTML-страницы в виде складных панелей, обеспечивая компактное представление информации в ограниченном пространстве экрана. В результате пользователи могут оперативно переключаться между панелями, открывая различный контент в одном и том же месте экрана.

Ранее (в Bootstrap 4) `Accordion` был частью компонента `Collapsible`, а не отдельным компонентом. В Bootstrap 5 `Accordion` — новый компонент. Вместе с `Accordion` появился новый класс `accordion`, который содержит внутри себя список элементов аккордеона. Кроме того, для аккордеонов предусмотрен новый стиль, отличный от предыдущего стиля в Bootstrap 4.

2.9.1. Класс *accordion*

В качестве примера сформируем шаблон accordion_1.html, в котором будет создан компонент `Accordion` на основе класса `accordion` (листинг 2.37).

Листинг 2.37. Шаблон accordion_1.html

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В этом шаблоне с использованием класса `accordion` созданы три вкладки аккордеона:

```
<!-- Начало аккордеона -->
<div class="accordion" id="accordionExample">
```

Каждая вкладка представляет элемент аккордеона, который создается на основе класса `accordion-item`:

```
<div class=»accordion-item»>
```

При этом каждая вкладка имеет заголовок с идентификатором (`id`) и с кнопкой для открытия (закрытия) вкладки:

```
<button class="accordion-button" type="button"
    data-bs-toggle="collapse"
    data-bs-target="#collapseOne"
    aria-expanded="true" aria-controls="collapseOne">
    Вкладка 1
</button>
```

Тело каждой вкладки создается на основе класса `class="accordion-collapse collapse"`. Если требуется, чтобы вкладка при загрузке страницы была открыта, то ее нужно создавать с использованием класса `accordion-collapse collapse show`:

```
class="accordion-collapse collapse show"
```

Содержимое тела вкладки (контент) создается на основе класса `accordion-body`:

```
<div class="accordion-body">
```

В теле вкладки можно отобразить совершенно разный контент. В приведенном примере вкладки отображают следующий контент:

- Вкладка 1 — обычный текст;
- Вкладка 2 — на основе тега `include` демонстрирует другую страницу сайта;
- Вкладка 3 — представляет рисунок.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница accordion_1.html**.

После загрузки шаблона accordion_1.html в окне браузера эта страница будет выглядеть так, как показано на рис. 2.42.

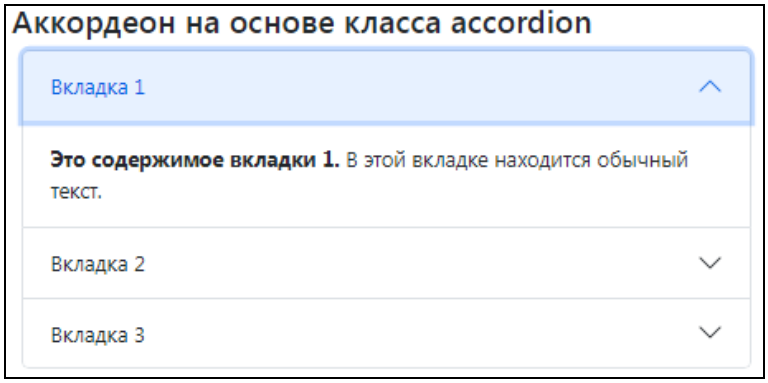


Рис. 2.42. Внешний вид шаблона accordion_1.html с открытой вкладкой 1

Как здесь можно видеть, при первоначальном показе шаблона в окне браузера вкладка 1 открыта, и в ней показана текстовая информация.

Если щелкнуть левой кнопкой мыши в любом месте заголовка вкладки 2, то она откроется (вкладка 1 при этом автоматически закроется), и в ней будет показана информация из другого шаблона сайта — inc_page.html (рис. 2.43).

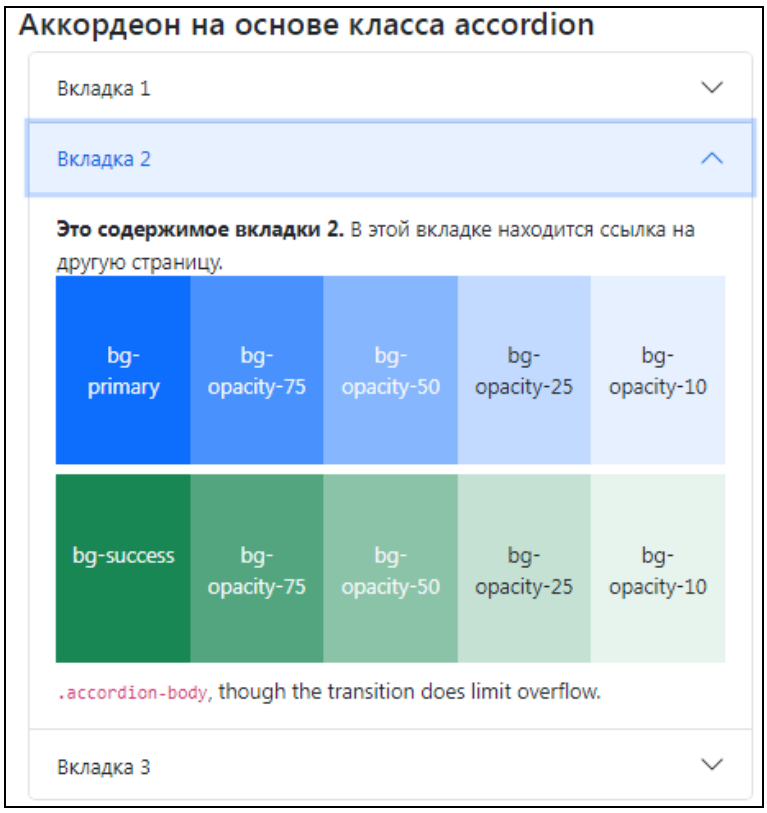


Рис. 2.43. Внешний вид шаблона accordion_1.html с открытой вкладкой 2

Если щелкнуть левой кнопкой мыши в любом месте заголовка вкладки 3, то она откроется (ранее открытая вкладка при этом автоматически закроется), и в ней будет показан рисунок (рис. 2.44).

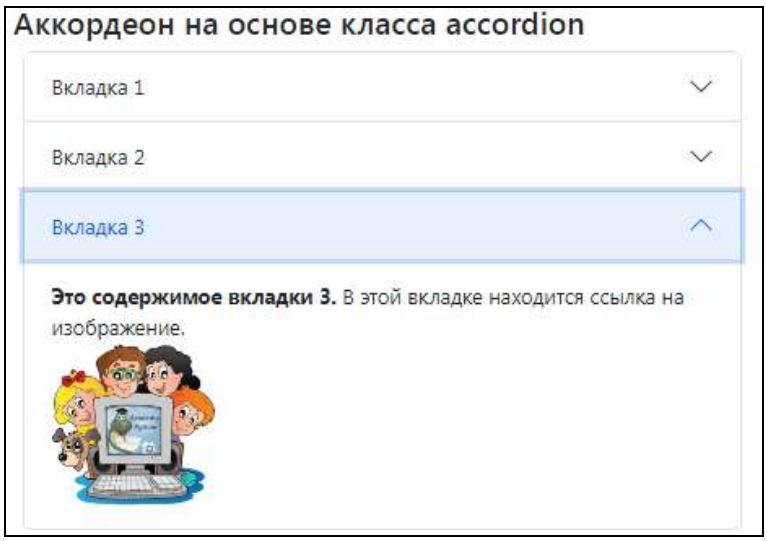


Рис. 2.44. Внешний вид шаблона accordion_1.html с открытой вкладкой 3

Если щелкнуть левой кнопкой мыши на заголовке открытой вкладки, то она закроется, и аккордеон станет занимать минимальную площадь экрана (рис.2.45).

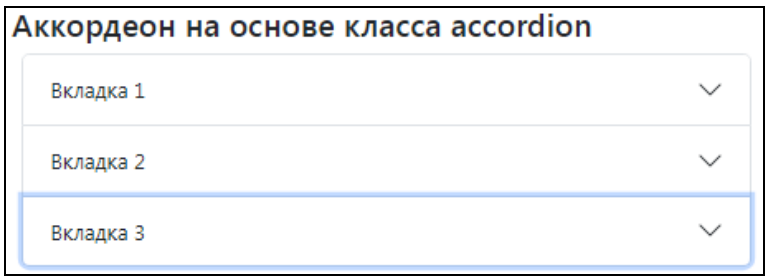


Рис. 2.45. Внешний вид шаблона accordion_1.html с закрытыми вкладками

2.9.2. Класс *accordion-flush*

Класс `accordion-flush` создает элемент «аккордеон», у которого удаляется цвет фона и границы с закругленными углами, что делает этот элемент похожим на родительский контейнер. В качестве примера сформируем шаблон accordion_2.html, в котором будет создан компонент «Аккордеон» на основе класса `accordion-flush`, добавив при этом возможность держать открытыми сразу несколько вкладок (листинг 2.38).

Листинг 2.38. Шаблон accordion_2.html

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В этом шаблоне созданы две вкладки аккордеона. Для создания компонента «Аккордеон» использован класс accordion-flush:

```
<!-- Начало аккордеона -->
<div class="accordion accordion-flush" id="ao-accordion">
```

Каждая вкладка представляет элемент аккордеона, который создается на основе класса `accordion-item`:

```
<div class=»accordion-item»>
```

При этом каждая вкладка имеет заголовок с идентификатором (`id`) и с кнопкой для открытия (закрытия) вкладки:

```
<button class="accordion-button collapsed"
    type="button"
    data-bs-toggle="collapse"
    data-bs-target="#panelsStayOpen-collapseTwo"
    aria-expanded="false"
    aria-controls="panelsStayOpen-collapseTwo">
    Вкладка 2
 </button>
```

В параметрах кнопки задана возможность панели оставаться открытой: `panelsStayOpen`.

Тело каждой вкладки создается на основе класса `class="accordion-collapse collapse"`. При этом параметры тела элемента аккордеона также содержат свойство, позволяющее вкладке оставаться открытой: `panelsStayOpen`.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница accordion_2.html**.

После загрузки шаблона accordion_2.html в окне браузера эта страница будет выглядеть так, как показано на рис. 2.46.

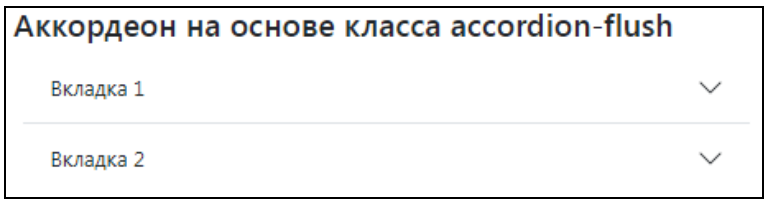


Рис. 2.46. Внешний вид шаблона accordion_2.html с закрытыми вкладками

Как здесь можно видеть, компонент «Аккордеон», созданный на основе класса `accordion-flush`, не имеет внешней рамки, а цвет его фона аналогичен цвету фона родительской страницы. Пользователю также предоставлена возможность держать открытыми сразу несколько вкладок.

Шаблон accordion_2.html при всех открытых вкладках будет выглядеть так, как показано на рис. 2.47).

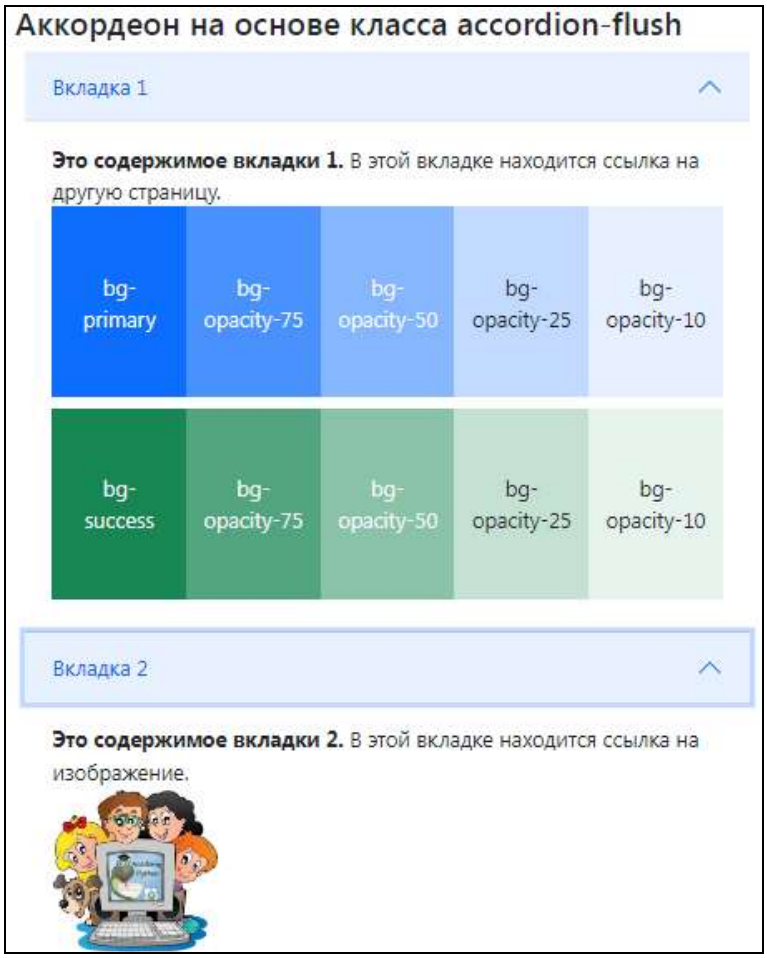


Рис. 2.47. Внешний вид шаблона accordion_2.html с открытыми вкладками

2.10. Компонент Bootstrap *card* «Карточки»

Карточка — это гибкий и расширяемый контейнер для размещения в нем различного контента. Карточка состоит из заголовка, подвала и контента. Для карточек доступно множество вариантов оформления — таких как выравнивание элементов, цвета элементов и многое другое. Структуру элементов карточки можно представить в следующем виде:

```
div class="card">
    <div class="card-header">Шапка</div>
    <div class="card-title">Заголовок</div>
    <div class="card-subtitle">Подзаголовок</div>
    <div class="card-body">Контент</div>
```

```
    <div class="card-text">Текст</div>
    <div class="card-footer">Подвал</div>
</div>
```

2.10.1. Карточка с параметрами по умолчанию

В качестве примера сформируем шаблон card_1.html, в котором будет создана простая карточка (листинг 2.39).

Листинг 2.39. Шаблон card_1.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом коде создана простая карточка Bootstrap со всеми элементами: шапка, заголовок, подзаголовок, тело, текст, подвал. В стиле задана ширина карточки (`style="width: 18rem;`). В тело карточки включены две ссылки на другие ресурсы на основе класса `card-link`.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница card_1.html**.

Шаблон card_1.html в окне браузера будет выглядеть так, как показано на рис. 2.48.



Рис. 2.48. Внешний вид шаблона card_1.html

Здесь можно видеть, что по умолчанию карточка обрамлена серой рамкой со скругленными краями и с цветом фона таким же, что и у родительской страницы. В этом примере ссылки не рабочие, а чисто декоративные, так как для нах указан пустой URL-адрес (`href="#"`).

Пояснение

Прием указания пустого адреса (`href="#"`) достаточно часто используется при отладке работы программных модулей в тех случаях, когда страница, на которую предполагается сослаться, еще не создана.

2.10.2. Карточка с параметрами цвета

В качестве примера создания простой карточки с заданием цвета для ее элементов сформируем шаблон card_2.html. В частности, зададим цвет фона, цвет текста, цвет рамки и толщину рамки (листинг 2.40).

Листинг 2.40. Шаблон card_2.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом коде созданы четыре карточки с разными параметрами цвета:

- цвет текста задан в параметрах `text-white` и `text-dark`;
- цвет фона задан в параметрах `bg-primary`, `bg-secondary`, `bg-success` и `bg-warning`;
- цвет рамок задан в параметрах `border-danger`, `border-dark`, `border-warning` и `border-primary`;
- толщина рамок задана параметром `border border-5` (5 пикселов).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница card_2.html**.

Шаблон card_2.html в окне браузера будет выглядеть так, как показано на рис. 2.49.



Рис. 2.49. Внешний вид шаблона card_1.html

2.10.3. Группы карточек

Если на странице много карточек, то их можно отобразить как единую группу с одинаковой шириной и высотой столбцов. В качестве примера сформируем шаблон card_3.html, в котором создадим группу карточек, которые, кроме текстовой информации, будут содержать и изображения. Код этого шаблона приведен в листинге 2.41.

Листинг 2.41. Шаблон card_3.html

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В этом коде создан блок из трех карточек. Каждая карточка содержит изображение, которое встроено в разные фрагменты карточек: перед заголовком, в теле карточки, в подвале.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница card_3.html**.

Шаблон card_3.html в окне браузера будет выглядеть так, как показано на рис. 2.50.

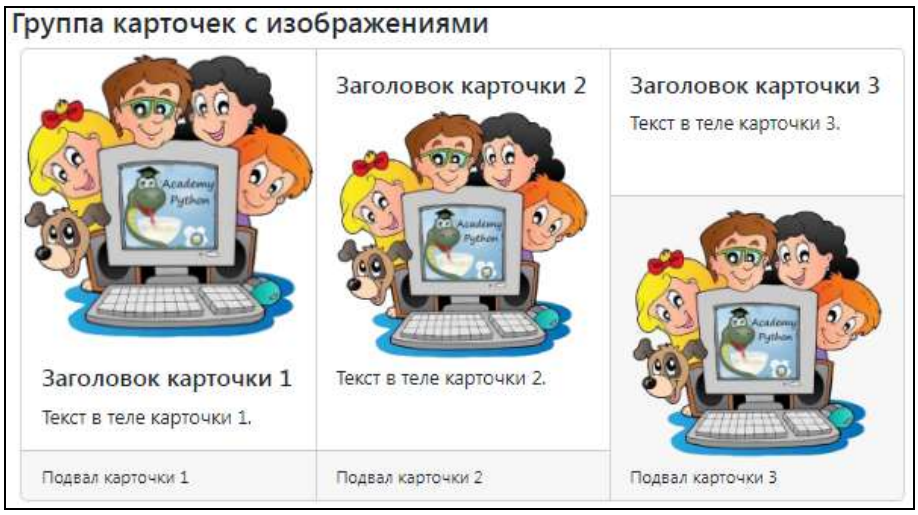


Рис. 2.50. Внешний вид шаблона card_3.html (группа карточек с изображениями)

Как здесь можно видеть, группа карточек имеет одинаковые размеры по высоте и ширине, а изображение встроено в разные фрагменты карточек.

2.11. Компонент Bootstrap *Carousel* «Карусель»

Карусель — это компонент Bootstrap, предназначенный для реализации слайд-шоу, то есть для создания окна, в котором происходит циклическая смена контента. Карусель работает с серией изображений, блоков текста или с изображениями с наложенным на них текстом. Карусель также имеет элементы управления принудительной смены слайдов.

В шаблоне HTML-страницы можно создать *статичную карусель* — с фиксированным программным кодом и с фиксированным контентом, заранее загруженным на сервер. Однако с использованием возможностей Django можно создать и *динамическую карусель*. В этом случае фиксированный программный код даст пользователю возможность удаленно загружать в БД сайта различный контент, который станет отображаться в окне карусели. При этом пользователь сможет менять как количество сладов карусели, так и их содержание.

В этой главе мы рассмотрим только первый вариант — «статичную карусель», а созданием динамической карусели займемся уже в следующей.

2.11.1. Статичная карусель с фиксированным контентом

Статичная карусель имеет фиксированное число слайдов и фиксированный контент, который отображается на каждом слайде. Создадим элемент «Карусель» в виде слайд-шоу, которое обычно располагается в верхней части страницы сайта, — например, интернет-магазина.

Пример HTML-кода страницы, в которой показано создание фиксированной карусели из трех слайдов, приведен в листинге 2.42 (страница carousel_1.html).

Листинг 2.42. Страница carousel_1.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном модуле создана карусель (`class="carousel slide"`), в которой имеются три слайда (`class="carousel-inner"`). Каждый слайд содержит только изображения (Kolbasa.jpg, Moloko.jpg, Krupa.jpg), которые находятся в папке static/images. Для изображений в стилях задано ограничение по высоте в 200 пикселов (`style="height: 200px;"`). Для принудительной смены слайдов предусмотрены две кнопки: для возврата к предыдущему слайду (`Previous`) и для перехода к следующему (`Next`).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница carousel_1.html**.

В окне эта страница будет выглядеть так, как показано на рис. 2.51–2.53.



Рис. 2.51. Слайд карусели с изображением Kolbasa.jpg



Рис. 2.52. Слайд карусели с изображением Moloko.jpg



Рис. 2.53. Слайд карусели с изображением Krupa.jpg

Как здесь можно видеть, на слайдах имеются по две кнопки в виде стрелок: стрелка «<» обеспечивает возврат к предыдущему слайду, а стрелка «>» позволяет выполнить переход к следующему слайду. Если эти кнопки не трогать, то по умолчанию принудительная смена слайдов будет происходить через каждве 5000 миллисекунд.

А теперь создадим элемент «Карусель» в виде слайд-шоу, на котором, кроме изображений, будет еще и поясняющий текст. Пример HTML-кода страницы, в которой показано создание фиксированной карусели из трех слайдов с изображениями и поясняющим текстом, приведен в листинге 2.43 (страница carousel_2.html).

Листинг 2.43. Страница carousel_2.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном модуле создан элемент «Карусель» (`class="carousel slide"`), к которому добавлен параметр, задающий принудительное время смены слайдов (`data-bs-interval="2000"`), то есть автоматическая смена слайдов будет происходить через каждые 2000 миллисекунд.

Карусель имеет три слайда (`class="carousel-inner"`). Каждый слайд содержит не только изображения, но и поясняющий текст (`class="carousel-caption d-none d-md-block"`). Первый слайд несет одну строку текста, второй — две, а третий — три текстовые строки.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 2** и перейти по ссылке **Страница carousel_2.html**.

В окне браузера эта страница будет выглядеть так, как показано на рис. 2.54–2.56.



Рис. 2.54. Слайд карусели с изображением Kolbasa.jpg и с одной строкой текста

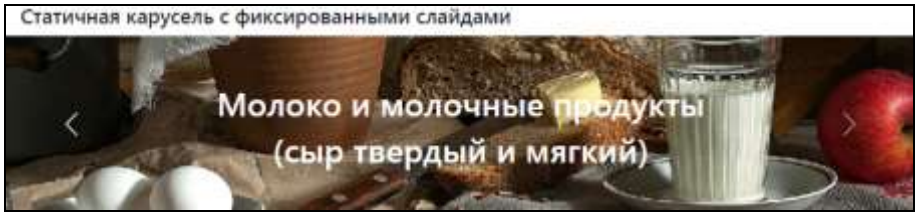


Рис. 2.55. Слайд карусели с изображением Moloko.jpg и с двумя строками текста

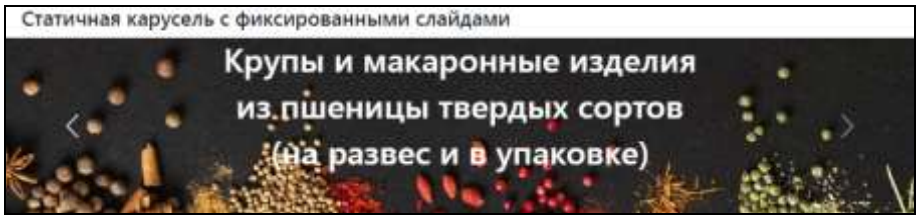


Рис. 2.56. Слайд карусели с изображением Krupa.jpg и с тремя строками текста

2.12. Краткие итоги

В этой главе мы познакомились с базовыми сведениями о статичных элементах Bootstrap, их параметрах и способах отображения в шаблонах Django. Они достаточно просто программируются и размещаются в этих шаблонах. Статичные элементы Bootstrap чаще используются на тех страницах сайта, которые относительно постоянны и не требуют частого обновления. Но если страница сайта «живая» — например, новостной сайт, то статичные элементы — это головная боль для его административного персонала. Ведь любое изменение элемента интерфейса (внешнего вида или содержания) потребует модификации программного кода с его переносом на публичный сервер.

Однако можно значительно упростить обновление элементов сайта, если использовать динамичные элементы Bootstrap. Для их создания придется потрудиться, так как программный код в этом случае будет и сложнее, и длиннее. Но вы получите колоссальный выигрыш при обновлении страниц сайта, поскольку и обновление контента, и изменение дизайна элементов сайта, а также добавление новых элементов и удаление старых, можно будет выполнять удаленно без переписывания программного кода. С этими возможностями вы и познакомитесь в следующей главе.

Глава 3. Динамичные элементы Bootstrap в шаблонах Django

В предыдущей главе мы получили базовые сведения о статичных элементах Bootstrap, их параметрах и способах отображения в шаблонах Django. Теперь можно перейти к изучению методов создания динамичных элементов Bootstrap. В этой главе показаны программные коды на примере трех элементов (аккордеон, карточки и карусель), но аналогичный подход можно использовать и для других элементов Bootstrap.

Из материалов этой главы вы узнаете:

- как создать динамичный «Аккордеон» с заполнением вкладок через административную панель Django;
- как создать динамичный «Аккордеон» с заполнением вкладок через интерфейс пользователя;
- как создать динамичные карточки с заполнением контента через административную панель;
- как создать динамичные карточки с заполнением контента через интерфейс пользователя;
- как создать динамичную карусель с заполнением контента через административную панель;
- как создать динамичную карусель с заполнением контента через интерфейс пользователя.

3.1. Динамичный «Аккордеон» с заполнением вкладок через административную панель Django

В *главе 2* было рассказано, как на основе фреймворка Bootstrap создать компонент «Аккордеон». Если использовать традиционный подход к созданию этого элемента, то на странице сайта будет показан статичный элемент «Аккордеон». Статичный в том смысле, что для изменения количества вкладок и их содержимого понадобится перепрограммировать этот элемент и обновить несколько программных модулей на публичном сервере.

Однако при проектировании сайта можно создать аккордеон с динамической загрузкой данных в раскрывающиеся вкладки. В этом случае администраторы сайта через административную панель смогут изменить как количество вкладок аккордеона, так и их содержание. При этом в программный код модулей сайта не придется вносить никаких изменений. Чтобы этого добиться, необходимо все элементы аккордеона хранить в базе данных. Посмотрим на упрощенном примере, как это можно сделать.

Сначала в приложении в файле models.py создадим модель `Accordion` со следующими полями (листинг 3.1).

Листинг 3.1. Модель Accordion

Примечание.

В демо-версии книги удалены листинги программ.

Здесь поле `title` предназначено для хранения надписи на кнопке аккордеона, а поля `strong_text` и `text` — для хранения содержания тела аккордеона (заголовок тела выделенным шрифтом и содержание тела обычным шрифтом). При этом нужно не забыть выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

Для того чтобы персонал мог удаленно менять содержимое аккордеона через административную панель, модель `Accordion` надо зарегистрировать в файле admin.py (листинг 3.2).

Листинг 3.2. Регистрация модели Accordion в файле admin.py

Примечание.

В демо-версии книги удалены листинги программ.

Для демонстрации динамичного аккордеона в меню или в теле одного из шаблонов нужно создать ссылку для вызова страницы, содержащей аккордеон (листинг 3.3).

Листинг 3.3. Ссылка для вызова динамичного аккордеона

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет инициирован URL-маршрут с именем `d_accordion_1` (листинг 3.4).

Листинг 3.4. URL-маршрут для вызова динамичного аккордеона

Примечание.

В демо-версии книги удалены листинги программ.

При адресации к этому маршруту будет инициировано представление `d_accordion_l` (листинг 3.5).

Листинг 3.5. Представление (view) accordion_1

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создается объект `tabs`, в который загружаются все записи из модели `Accordion` (то есть все вкладки аккордеона). Затем загруженные вкладки через словарь передаются в шаблон с именем d_accordion_1.html (листинг 3.6).

Листинг 3.6. Шаблон d_accordion_1.html

Примечание.

В демо-версии книги удалены листинги программ.

Здесь на основе тега `{% for tab in tabs %}` создан цикл, в тело которого вложены элементы аккордеона. Каждый элемент аккордеона имеет идентификатор (`id`). Для того чтобы в теле цикла каждый идентификатор аккордеона имел уникальное значение, к его текстовому содержанию добавлен счетчик циклов: `{{ forloop.counter }}`. То же самое сделано с другими уникальными атрибутами аккордеона, которые теперь имеют следующий вид:

- `class="accordion-header" id="headingOne{{ forloop.counter }}"`;
- `data-bs-target="#collapseOne{{ forloop.counter }}"`;
- `aria-labelledby="headingOne{{ forloop.counter }}"`.

При этом содержимое каждого элемента аккордеона выбирается из БД:

- заголовок кнопки: `{{ tab.title }}`;
- заголовок вкладки (жирный шрифт): `{{ tab.strong_text }}`;
- содержание вкладки (обычный шрифт): `{{ tab.text }}`.

Теперь заполнить вкладки аккордеона текстовой информацией можно в административной панели Django (рис. 3.1)..

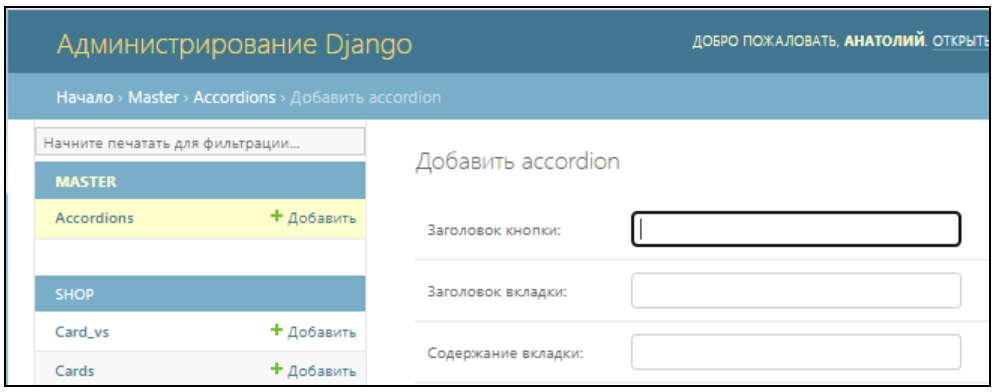


Рис. 3.1. Заполнение таблицы `Accordion` в административной панели Django

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Страница d_accordion_1.html**.

После загрузки шаблона d_accordion_1.html элемент «Аккордеон» в закрытом виде будет выглядеть, как показано на рис. 3.2, а с открытой вкладкой — как показано на рис. 3.3.



Рис. 3.2. Вкладки динамичного аккордеона в закрытом виде

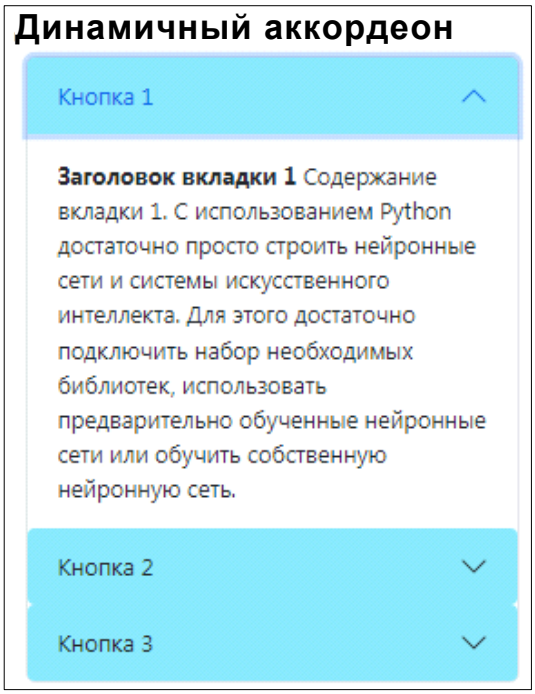


Рис. 3.3. Вкладка динамичного аккордеона в открытом виде

3.2. Динамичный «Аккордеон» с заполнением вкладок через интерфейс пользователя

В предыдущем разделе было показано, как с использованием фреймворка Bootstrap создать динамичный компонент «Аккордеон». Для динамичного аккордеона не нужно менять код страницы, однако недостатком этого метода является то, что такой аккордеон обычно привязан к одной странице сайта.

Тем не менее, при проектировании сайта можно создать такую модель для элемента «Аккордеон», которая обеспечит создание неограниченного количества этих элементов, позволит добавить их на любую страницу сайта или удалить оттуда, разместить в заданном месте страницы и изменять как содержимое, так и внешний интерфейс элемента. При этом все эти действия можно будет выполнять не только через административную панель сайта, но и через его интерфейс в личном кабинете администраторов сайта. Правда, для этого понадобится создать на сайте личный кабинет для административного персонала и набор необходимых программных модулей, необходимых для редактирования динамичных элементов «Аккордеон».

Посмотрим на упрощенном примере, как это можно сделать, опустив для простоты изложения реализацию личного кабинета, а показав только формирование модели динамичного аккордеона и модулей для ее редактирования. Для этого в приложении в файле models.py создадим несколько моделей.

Чтобы элементы аккордеона можно было разместить на разных страницах сайта, необходимо создать модель `Pages`, которая будет хранить наименования страниц сайта (листинг 3.7).

Листинг 3.7. Модель Pages

Примечание.

В демо-версии книги удалены листинги программ.

В этой модели создано всего одно поле: `name` — для хранения имени страницы сайта.

Чтобы на одной странице сайта можно было создать несколько элементов «Аккордеон», надо создать модель `Accordion_1`, — она и будет хранить эти элементы (листинг 3.8).

Листинг 3.8. Модель Accordion_1

Примечание.

В демо-версии книги удалены листинги программ.

В этой модели созданы два поля:

- `page` — для связи по первичному ключу с моделью `Pages` (со страницами сайта);
- `box` — для указания блока (раздела) страницы, в котором будет размещен аккордеон.

В нашем примере поле `box` будет принимать только три значения: `L` — левый блок (левая панель) страницы, `C` — центральный блок (центральная панель) страницы, `R` — правый блок (правая панель) страницы.

С использованием модели `Accordion_1` в любой панели на любой странице сайта можно будет создавать неограниченное количество элементов «Аккордеон».

Чтобы элементы «Аккордеон» могли иметь несколько вкладок, нужно создать модель `Tabs_accord` — для хранения вкладок аккордеонов и функцию `rename_tab` — для изменения имени изображения, загружаемого на вкладку.

Пояснение

Поскольку публичные серверы не принимают имена изображений на кириллице, наша модель сопровождается функцией автоматического изменения имени файла загружаемого изображения, которая создает на сайте удаленного сервера папку с именем *tabs*, в которую будут сохраняться загружаемые файлы изображений с именами *tab_<id записи в БД>.<расширение>*.

Программный код функции `rename_tab` и модели `Tabs_accord` приведен в листинге 3.9.

Листинг 3.9. Функция rename_tab и модель Tabs_accord

Примечание.

В демо-версии книги удалены листинги программ.

В модели `Tabs_acord` созданы следующие поля:

- `accordion` — поле для связи по первичному ключу с моделью `Accordion_1`;
- `title_btn` — текст заголовка кнопки аккордеона;
- `title_tab` — текст заголовка вкладки аккордеона;
- `text_tab` — текст вкладки аккордеона;
- `image` — изображение на вкладке аккордеона;
- `set_btn` — параметры внешнего вида кнопки;
- `set_text` — параметры внешнего вида текста;
- `link` — ссылка для обращения к внешним ресурсам Интернета.

Обратите внимание на импорт дополнительных модулей `os` и `uuid4` — они необходимы для работы функции `rename_tab`.

В примере нашей модели мы покажем на вкладке аккордеона текст, изображение и ссылку на внешние ресурсы Интернета. Надо также иметь в виду, что у этой модели — в зависимости от характера разрабатываемого приложения — может быть и совершенно другой набор полей.

Не забудьте выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

Чтобы созданные модели можно было заполнять данными через административную панель Django, их нужно зарегистрировать в файле admin.py текущего приложения:

```
from django.contrib import admin
from .models import Accordion_1, Pages, Tabs_acord

# Регистрация модели Accordion_1
admin.site.register(Pages)
# Регистрация модели Accordion_1
admin.site.register(Accordion_1)
# Регистрация модели Tabs_acord
admin.site.register(Tabs_acord)
```

Впрочем, если планируется заполнять эти модели данными только через интерфейс пользователя, то их можно и не регистрировать в файле admin.py.

Чтобы заполнять аккордеон данными через интерфейс пользователя, нам понадобятся две формы: для заполнения сведений об аккордеоне и для заполнения данными вкладок аккордеона. Для этого в файле forms.py создадим формы `AccordForm` и `Tab_acord_Form` (листинг 3.10).

Листинг 3.10. Формы AccordForm и Tab_acord_Form

Примечание.

В демо-версии книги удалены листинги программ.

Эти формы используют модели `Accordion_1` и `Tabs_acord`, при этом поля `page` и `accordion` исключаются из полей, доступных для ввода и корректировки.

Откроем административную панель и для текущего приложения создадим страницу с именем **Примеры** — на ней мы станем размещать динамичные элементы интерфейса «Аккордеон» (рис. 3.4).

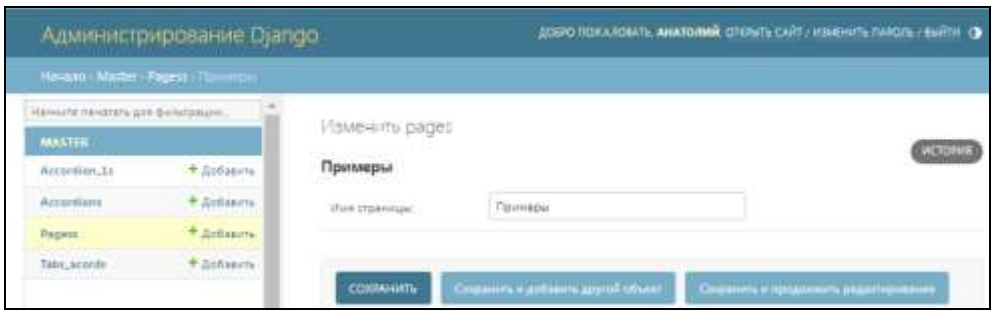


Рис. 3.4. Создание страницы **Примеры** в административной панели

3.2.1. Программные модули для создания динамичных элементов «Аккордеон»

Чтобы получить возможность в интерфейсе сайта создавать элементы «Аккордеон», необходимо сформировать следующий набор модулей:

- для редактирования динамичного аккордеона;
- для добавления динамичного аккордеона;
- для обновления динамичного аккордеона;
- для удаления динамичного аккордеона.

Код представления (view) `accord_edit` для редактирования динамичного аккордеона приведен в листинге 3.11.

Листинг 3.11. Представление (view) accord_edit

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении в объект `tek_page` загружена текущая страница на основе ее идентификатора в БД (`id_page`), а в объект `accords` из БД загружены все аккордеоны текущей страницы. Эти объекты через словарь `context` передаются в шаблон accord_edit.html, который имеет следующий код (листинг 3.12).

Листинг 3.12. Шаблон accord_edit.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне создана кнопка `Добавить аккордеон`, которая активирует представление с именем `accord_add` по следующей ссылке:

```
<a href="{% url 'master:accord_add' %}" class="btn btn-primary">
    <i class="bi bi-database-add fs-5"></i> Добавить аккордеон</a>
```

После этой кнопки идет таблица, в которой отображается информация о созданных аккордеонах, — в каждой ее строке имеются три кнопки:

- `Изменить` — для изменения сведений об аккордеоне;
- `Удалить` — для удаления аккордеона;
- `Редактор вкладок` — для редактирования вкладок аккордеона (добавления, изменения, удаления).

Эти кнопки активируют представления с именами `accord_upd`, `accord_del`, `accord_upd` и `ac_tab_edit` по следующим ссылкам:

```
<a href="{% url 'master:accord_upd' id=obj.id %}"
    class="btn btn-success btn-sm my-1">
    <i class="bi bi-repeat"></i> Изменить</a>
```

```
<a href="{% url 'master:accord_del' id=obj.id %}"
    class="btn btn-danger btn-sm my-1">
     <i class="bi-trash3"></i> Удалить</a>

 <a href="{% url 'master:ac_tabs_edit' id=obj.id %}"
    class="btn btn-primary btn-sm my-1">
     <i class="bi bi-repeat"></i> Редактор вкладок</a>
```

Код представления `accord_add` приведен в листинге 3.13.

Листинг 3.13. Представление (view) accord_add

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создается объект `form`, в который загружается форма `AccordForm()`. Эта форма через словарь `context` передается в универсальный шаблон add_upd_form.html, который имеет следующий код (листинг 3.14).

Листинг 3.14. Универсальный шаблон add_upd_form.html

Примечание.

В демо-версии книги удалены листинги программ.

Этот шаблон представляет форму, в которой в виде таблицы отображаются поля для ввода данных: `{ form.as_table }`. Здесь же имеются две кнопки: `Отмена` — для отмены сохранения введенной информации и возврата к предыдущему окну и `Сохранить` — для сохранения введенной информации и возврата к предыдущему окну.

При возврате из этой формы в представление `accord_add` там происходит сохранение в БД сведений о динамичном аккордеоне.

Код представления `accord_upd` приведен в листинге 3.15.

Листинг 3.15. Представление (view) accord_upd

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор редактируемого аккордеона (`id`), на основе которого в объект `obj` из БД загружаются сведения об этом аккордеоне. После чего создается объект `form`, в который загружается форма `AccordForm()` с информацией о текущем аккордеоне. Эта форма через словарь `context` передается в универсальный шаблон add_upd_form.html. При возврате из этого универсального шаблона-формы в представление `accord_upd` там происходит сохранение в БД сведений о динамичном аккордеоне.

Код представления `accord_del` приведен в листинге 3.16.

Листинг 3.16. Представление (view) accord_del

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор удаляемого аккордеона (`id`), на основе которого в объект `obj` из БД загружаются сведения об этом аккордеоне. После чего в объект `del_prod` загружается идентификатор удаляемого аккордеона, который через словарь `content` передается в универсальный шаблон del_yesno.html.

Код шаблона del_yesno.html приведен в листинге 3.17.

Листинг 3.17. Шаблон del_yesno.html

Примечание.

В демо-версии книги удалены листинги программ.

Этот шаблон представляет форму, в которой отображается идентификатор удаляемой записи и имеются две кнопки:

- `Отмена` — для отмены удаления аккордеона;
- `Удалить` — для подтверждения удаления аккордеона.

Кроме того, указанный шаблон содержит скрипт для отображения на кнопке вращающегося спина в том случае, если процесс удаления записи из БД занимает продолжительное время.

По нажатию на эти кнопки происходит возврат в представление `accord_del`, где в случае нажатия на кнопку `Удалить` выполняется удаление информации об аккордеоне из БД.

Для того чтобы можно было активировать описанные здесь представления, нужно в файле urls.py создать соответствующие URL-маршруты (листинг 3.18).

Листинг 3.18. Маршруты для редактирования динамичных аккордеонов

Примечание.

В демо-версии книги удалены листинги программ.

Код представления для редактирования вкладок аккордеона `ac_tabs_edit` приведен в листинге 3.19.

Листинг 3.19. Представление ac_tabs_edit

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор родительского аккордеона (`id`), затем в нем создаются объект `id_page` (идентификатор текущей страницы) и объект `tek_page`, в который из БД загружается сама текущая страница. В объект `tabs` из БД загружаются вкладки родительского аккордеона. Все созданные объекты через словарь `context` передаются в шаблон ac_tabs_edit.html, который имеет следующий код (листинг 3.20).

Листинг 3.20. Шаблон ac_tabs_edit.html

Примечание.

В демо-версии книги удалены листинги программ.

Этот шаблон содержит кнопку `Добавить вкладку` и таблицу для отображения параметров вкладок. Кнопка `Добавить вкладку` активирует маршрут для вызова представления tab_accord_add, которое имеет следующий код (листинг 3.21).

Листинг 3.21. Представление tab_accord_add

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении на основе формы `Tab_acord_Form()` создается объект `form`, который через словарь `context` передается в универсальный шаблон-форму add_upd_form.html. При возврате из этой формы в БД в таблице `Tabs_acord` создается новая запись с параметрами вкладки аккордеона.

В в каждой строке таблицы шаблона ac_tabs_edit.html имеются две кнопки:

- `Изменить` — для изменения (обновления) информации о вкладке аккордеона;

- `Удалить` — для удаления вкладки аккордеона.

Эти кнопки активируют маршруты с именами: `tab_accord_upd` — вызывает представление для обновления информации на вкладке аккордеона и `tab_accord_del` — вызывает представление для удаления вкладки аккордеона.

Код представления `tab_accord_upd` приведен в листинге 3.22.

Листинг 3.22. Представление tab_accord_upd

Примечание.
В демо-версии книги удалены листинги программ.

Это представление получает идентификатор текущей вкладки аккордеона (`id`), загружает в объект `obj` из БД все параметры текущей вкладки и через форму `Tab_acord_Form` загружает параметры текущей вкладки в объект `form`. Затем форма `form` через словарь `context` передается в универсальный шаблон-форму add_upd_form.html. При возврате из этой формы в БД в таблице `Tabs_acord` обновляется запись с параметрами вкладки аккордеона.

Код представления `tab_accord_del` приведен в листинге 3.23.

Листинг 3.23. Представление tab_accord_del

Примечание.
В демо-версии книги удалены листинги программ.

Это представление получает идентификатор текущей вкладки аккордеона (`id`) и загружает в объект `obj` из БД все параметры текущей вкладки. Затем в объект `del_prod` загружается имя кнопки вкладки аккордеона, и оно через словарь `context` передается в универсальную форму-шаблон del_yesno.html (см. листинг 3.17). Если в этом шаблоне была нажата кнопка `Удалить`, то текущая вкладка будет удалена из БД.

Для активации всех этих представлений, обеспечивающих редактирование вкладок аккордеона, нужно в файле urls.py создать соответствующие маршруты (листинг 3.24).

Листинг 3.24. Маршруты для редактирования вкладок динамичных аккордеонов

Примечание.
В демо-версии книги удалены листинги программ.

3.2.2. Модули для вывода динамичных аккордеонов на страницы сайта

Итак, мы создали все модули, которые позволяют сформировать как динамичный аккордеон, так и его вкладки. Теперь покажем, как динамичный аккордеон можно вывести на страницу сайта. Для этого создадим тестовую страницу (шаблон) examples_1.html и соответствующие маршрут и представление.

Код представления `examples_1` приведен в листинге 3.25.

Листинг 3.25. Представление examples_1

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении создаются три объекта: `tabs`, `tabs2` и `tabs3` — в которые загружаются вкладки аккордеонов соответственно левого блока (левой панели), центрального блока (центральной панели) и правого блока (правой панели). Все эти объекты через словарь `context` передаются в шаблон examples_1.html.

Для активации представления `examples_1` в файле urls.py нужно создать URL-маршрут (листинг 3.26).

Листинг 3.26. URL-маршрут с именем examples_1

Примечание.
В демо-версии книги удалены листинги программ.

Код шаблона examples_1.html приведен в листинге 3.27.

Листинг 3.27. Шаблон examples_1.html

Примечание.
В демо-версии книги удалены листинги программ.

Шаблон examples_1.html представляет собой демонстрационную страницу, состоящую из трех панелей: левой боковой, центральной и правой боковой. В каждую панель с использованием тега `include` вставлен другой шаблон:

- в левую боковую панель — шаблон accordion_show.html;
- в центральную панель — шаблон accordion_c_show.html;
- в правую боковую панель — шаблон accordion_r_show.html.

Коды шаблонов accordion_show.html, accordion_c_show.html и accordion_r_show.html представлены в листингах 3.28–3.30.

Листинг 3.28. Шаблон accordion_show.html

Примечание.
В демо-версии книги удалены листинги программ.

Листинг 3.29. Шаблон accordion_c_show.html

Примечание.
В демо-версии книги удалены листинги программ.

Листинг 3.30. Шаблон accordion_r_show.html

Примечание.
В демо-версии книги удалены листинги программ.

Все три эти шаблона имеют одинаковую структуру. В каждом шаблоне с использованием тега Django `for` организован цикл по вкладкам аккордеона (`tabs`, `tabs2`, `tabs3`). В качестве идентификатора элементов вкладок задействуется счетчик циклов (например, `id="{{ forloop.counter }}"`). В теле элементов аккордеона применяется тег Django `if`, в котором проверяется наличие или отсутствие в БД сведений об элементе. Если сведения об элементе в БД присутствуют, он будет отображен в шаблоне.

* * *

Итак, мы рассмотрели код всех модулей и шаблонов, которые обеспечивают формирование и вывод динамичных аккордеонов. Имея такой набор, администраторы сайта могут, не меняя программный код, добавить аккордеон на любую страницу сайта (и даже в любую панель страницы), изменить количество вкладок и их содержание, поменять дизайн элементов вкладок. При удалении аккордеона из БД он автоматически исчезнет со страниц сайта. Рас-

смотрим теперь, как работает наш программный код, и как выглядят динамичные аккордеоны на страницах сайта.

> ***Примечание***
> Для того чтобы читатели интерактивной цифровой книги не испортили содержание примеров этого раздела, на публичном сайте в модулях добавления, корректировки и удаления данных из БД закомментированы программные строки, связанные с физическим изменением записей в БД.

3.2.3. Пример создания динамичного аккордеона через интерфейс пользователя

Проверим работу созданных нами модулей.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Представление accord_edit.html**.

При вызове модуля для создания и редактирования динамичного аккордеона шаблон accord_edit.html будет выглядеть так, как показано на рис. 3.5).

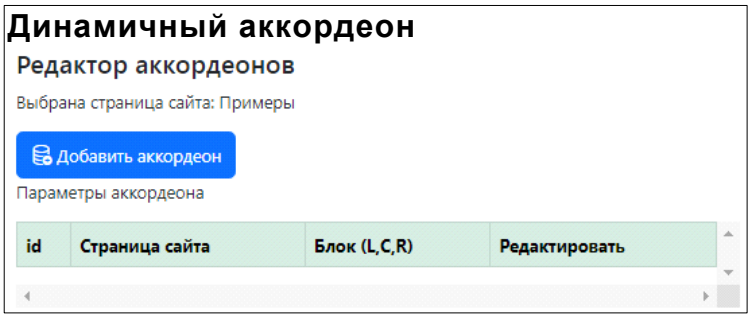


Рис. 3.5. Внешний вид редактора динамичного аккордеона (шаблона accord_edit.html) при первой загрузке

> ***Примечание***
> В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

По нажатию на кнопку **Добавить аккордеон** откроется шаблон-форма accord_add.html, в которой нужно указать, на какой панели страницы: левой (`L`), центральной (`C`) или правой (`R`) — будет размещен динамичный аккордеон (рис. 3.6).

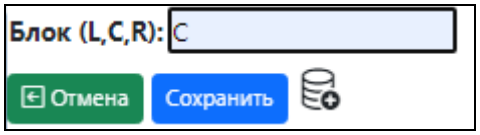


Рис. 3.6. Выбор панели страницы для размещения динамичного аккордеона

Для примера из выпадающего списка выбираем элемент `C` и нажимаем кнопку **Сохранить**. После этого пользователь будет возвращен в шаблон accord_edit.html, где появится запись с созданным динамичным аккордеоном (рис. 3.7).

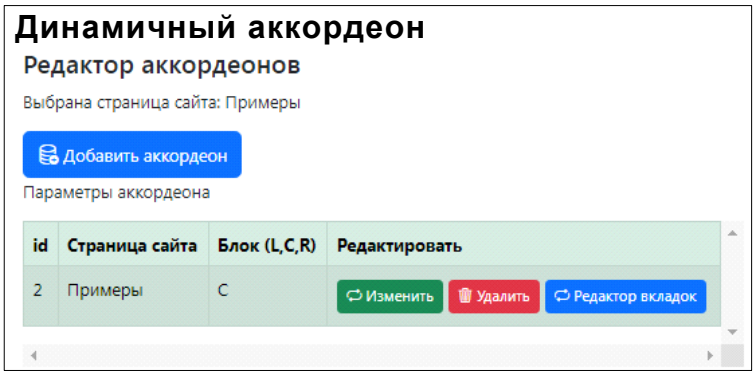


Рис. 3.7. Внешний вид редактора динамичного аккордеона (шаблона accord_edit.html) при созданном аккордеоне

Как здесь можно видеть, на страницу сайта **Примеры** в центральной панели будет помещен динамичный аккордеон, который в БД имеет идентификатор `id=2`. В колонке таблицы с именем **Редактировать** появятся три кнопки: **Изменить**, **Удалить** и **Редактор вкладок**.

На следующем шаге добавим к нашему динамичному аккордеону несколько вкладок — нажмите для этого кнопку **Редактор вкладок**. В результате откроется шаблон для создания и редактирования вкладок текущего аккордеона ac_tab_edit.html. При первой загрузке этого шаблона таблица будет пустой, так как в БД еще нет ни одной записи (рис. 3.8).



Рис. 3.8. Внешний вид редактора вкладок аккордеона (шаблона ac_tab_edit.html) при первой загрузке

Для создания новой вкладки нажмите кнопку **Добавить вкладку**, после чего откроется универсальный шаблон-форма add_upd_form.html, в котором можно заполнить данными поля базы данных вкладки динамичного аккордеона (рис. 3.9).

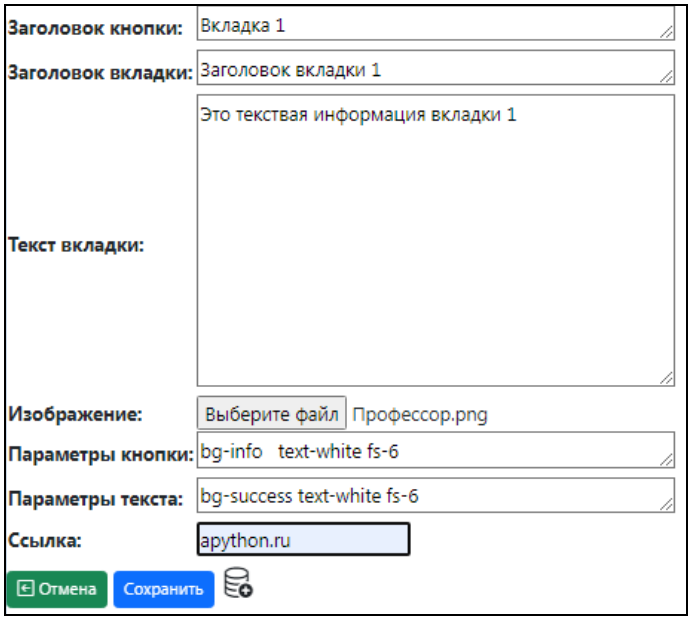


Рис. 3.9. Форма для заполнения данными вкладки аккордеона (шаблон add_upd_form.html)

По нажатию на кнопку **Сохранить** пользователь будет возвращен в шаблон ac_tab_edit.html, где появится запись БД с заполненной информацией первой вкладки (рис 3.10).



Рис. 3.10. Внешний вид редактора вкладок аккордеона (шаблона ac_tab_edit.html) с введенными параметрами вкладки

Создадим еще две вкладки в текущем динамичном аккордеоне и заполним их разной информацией (рис. 3.11).



Рис. 3.11. Внешний вид редактора вкладок аккордеона (шаблона ac_tab_edit.html) с введенными параметрами для трех вкладок

Таким образом, в базе данных сформирован динамичный аккордеон с тремя вкладками. Теперь его нужно добавить на страницу, которую мы назвали **Примеры**. Вспомним, что в *разд. 3.2.2* мы уже создали такую страницу (с именем examples_1.html) и программные модули для ее отображения.

3.2.4. Пример вывода динамичного аккордеона на страницу сайта

Итак, динамичный аккордеон создан и добавлен на страницу с именем **Примеры**. Теперь можно посмотреть, как он выглядит на этой странице.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Страница examples_1.html**.

При вызове модуля отображения динамичного аккордеона шаблон examples_1.html будет выглядеть так, как показано на рис. 3.12.

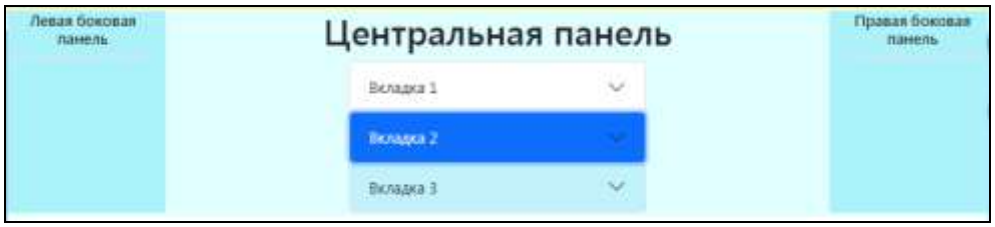


Рис. 3.12. Внешний вид шаблона examples_1.html при закрытых вкладках

Как здесь можно видеть, после занесения в БД информации о динамичном аккордеоне, он появился в центральной панели нашей тестовой страницы, при этом кнопки вкладок получили разное цветовое оформление.

Если мы теперь откроем вкладку 1, то на ней будет показана текстовая информация, изображение и кнопка для перехода к другому интернет-ресурсу — к сайту **apython.ru** (рис. 3.13). Вкладка 1 здесь имеет фон зеленого цвета и текстовую информацию белого цвета.

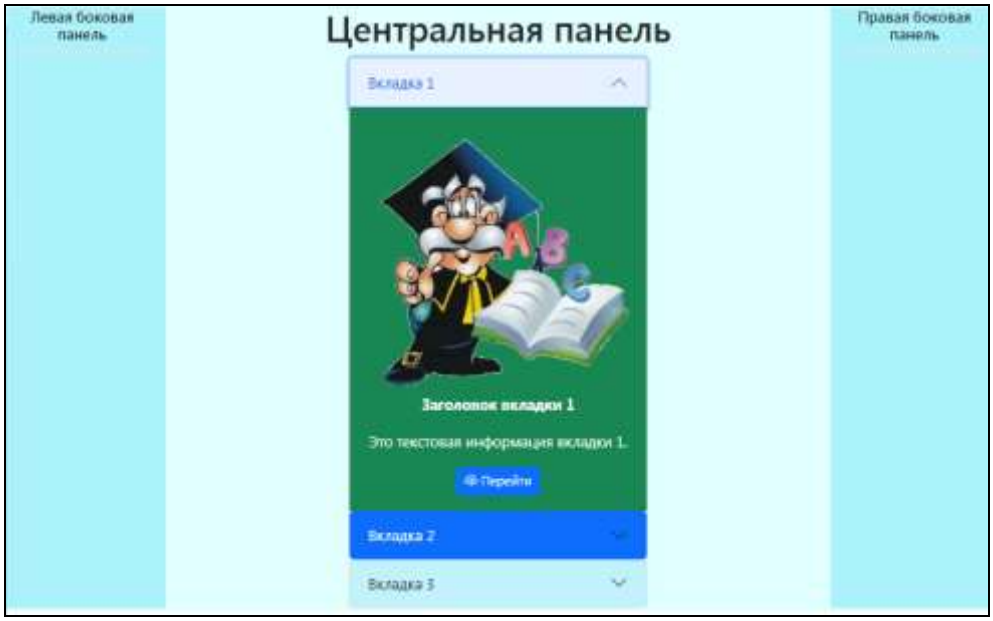


Рис. 3.13. Внешний вид шаблона examples_1.html при открытой вкладке 1

Открыв вкладку 3, мы увидим, что она имеет другое цветовое оформление: фон светло-зеленого цвета и текстовую информацию черного цвета (рис. 3.14).

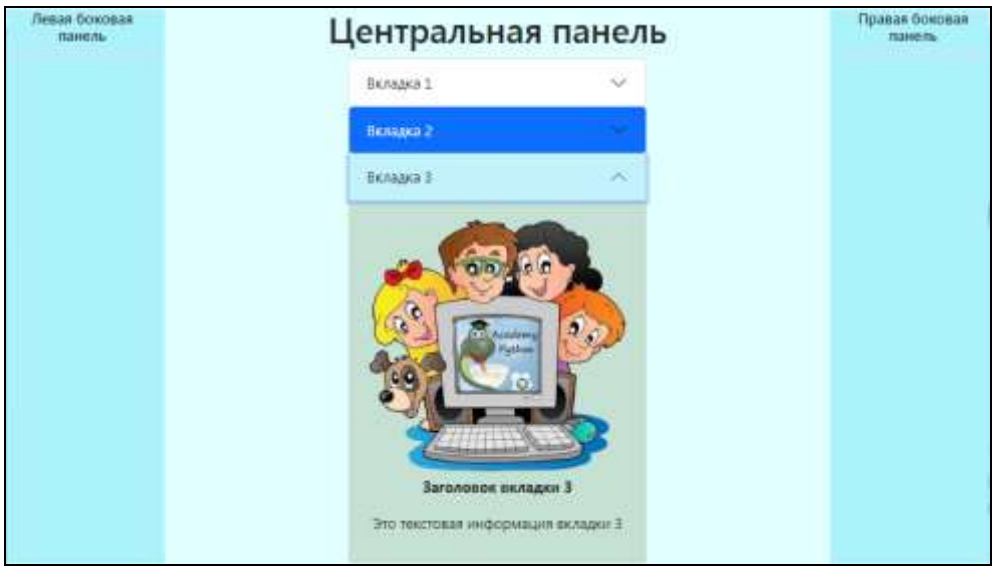


Рис. 3.14. Внешний вид шаблона examples_1.html при открытой вкладке 3

Теперь отобразим динамичный аккордеон в левой боковой панели. Для этого нужно сначала открыть редактор аккордеонов (рис. 3.15) и нажать там на кнопку **Изменить**.

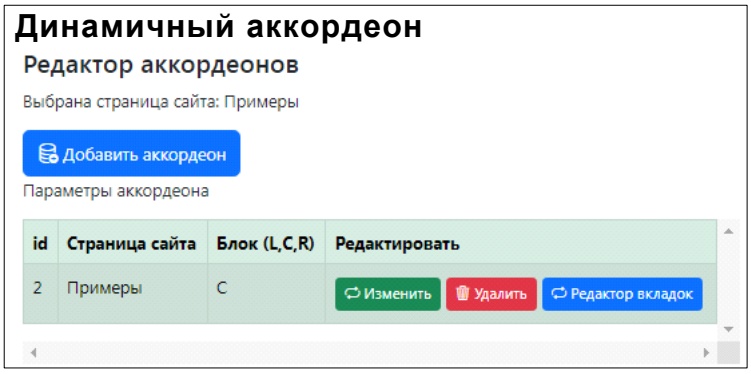


Рис. 3.15. Шаблон accord_edit.html для редактирования динамичного аккордеона

В открывшейся форме указываем расположение динамичного аккордеона в левой панели — `L` (рис. 3.16).

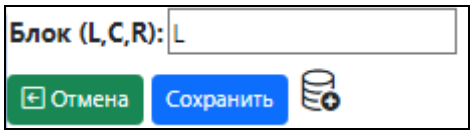


Рис. 3.16. Перенос динамичного аккордеона на левую боковую панель

Сохраняем измененный параметр и возвращаемся к тестовой странице examples_1.html для вывода динамичного аккордеона (рис. 3.17).

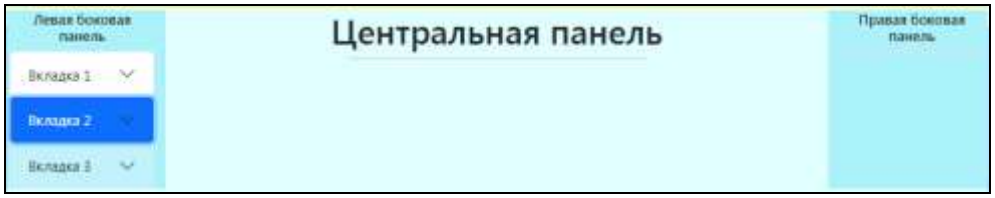


Рис. 3.17. Динамичный аккордеон в левой боковой панели

Как здесь можно видеть, левая боковая панель имеет меньшую ширину, чем центральная панель, однако вкладки аккордеона адаптировались к этой ширине. Если теперь открыть любую вкладку, то будет видно, что и содержимое вкладки адаптировалось к новому размеру панели (рис. 3.18).



Рис. 3.18. Динамичный аккордеон на левой боковой панели с открытой вкладкой 2

Аналогичным образом можно перенести динамичный аккордеон в правую боковую панель (рис. 3.19).

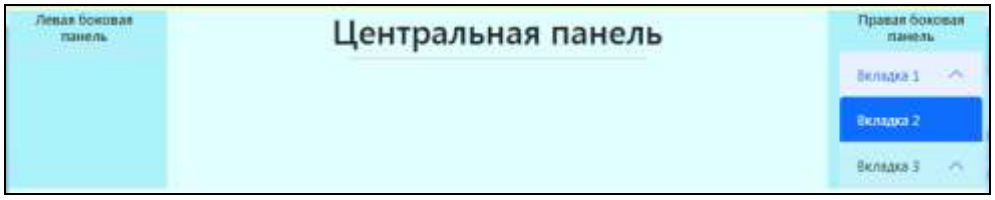


Рис. 3.19. Динамичный аккордеон в правой боковой панели

* * *

Подводя итоги этого раздела, можно сделать следующие выводы. Написание программного кода для формирования динамичного аккордеона — достаточно трудоемкий процесс, однако эти трудозатраты окупаются при сопровождении сайта. Администраторы сайта получают возможность добавлять или удалять элемент «динамичный аккордеон» в любые фрагменты любой страницы сайта, изменять количество вкладок аккордеона и их содержимое, менять оформление внешнего вида аккордеона. При этом им не требуется вносить какие-либо изменения в код программных модулей сайта.

3.3. Динамичные карточки с заполнением контента через административную панель

При проектировании сайта также можно создавать и карточки с динамической загрузкой текста, изображений, кнопок и других элементов. В этом случае администраторы сайта через административную панель смогут изменять как количество карточек, так и их содержание (изображения и поясняющие надписи).

Карточка имеет следующую структуру: изображение, заголовок, подзаголовок, текст, подвал. Кроме того, она обладает еще и такими характеристиками как размер, цвет фона, цвет текста, цвет границы, ширина границы. При этом каждая карточка может иметь свой набор стилей (цвет, размер), изображение и текстовую информацию.

При размещении на страницах сайта динамичных карточек необходимо все их элементы хранить в базе данных — для этого мы сейчас создадим модель `Card` и функцию `rename_image_card`, позволяющую изменять имя изображения, загруженного в карточку (листинг 3.31).

Пояснение

Поскольку публичные серверы не принимают имена изображений на кириллице, наша модель сопровождается функцией автоматического изменения имени файла загружаемого изображения, которая создает на сайте удаленного сервера папку с именем *image_card*, в которую будут сохраняться загружаемые файлы изображений с именами *img_*<id записи в БД>.<расширение>.

Листинг 3.31. Модель Card и функция rename_image_card

Примечание.
В демо-версии книги удалены листинги программ.

В модели `Card` созданы следующие поля:

- `image` — изображение, загруженное в карточку;
- `title` — заголовок карточки;
- `subtitle` — подзаголовок карточки;
- `text` — текст карточки;
- `footer` — подвал карточки;
- `style_size` — размер карточки;
- `style_fon` — цвет фона карточки;
- `style_text` — цвет текста карточки;
- `style_border` — цвет рамки карточки;
- `style_border_size` — толщина рамки карточки.

Надо также не забыть выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

Чтобы персонал мог удаленно менять содержимое карточек через административную панель, модель `Card` нужно зарегистрировать в файле admin.py (листинг 3.32).

Листинг 3.32. Регистрация модели Card в файле admin.py

Примечание.
В демо-версии книги удалены листинги программ.

Для демонстрации динамичной карточки в меню или в теле одного из шаблонов надо создать ссылку для вызова страницы, содержащей карточку. Эта ссылка может иметь следующий код (листинг 3.33).

Листинг 3.33. Ссылка для вызова динамичной карточки

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет инициирован URL-маршрут (листинг 3.34).

Листинг 3.34. URL-маршрут d_card_1

Примечание.
В демо-версии книги удалены листинги программ.

При адресации к этому маршруту будет инициировано представление `d_card_1` (листинг 3.35).

Листинг 3.35. Представление d_card_1

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении создается объект `cards`, в который загружаются все записи из модели `Card` (то есть все параметры карточки). Затем загруженные карточки через словарь передаются в шаблон с именем d_card_1.html (листинг 3.36).

Листинг 3.36. Шаблон d_card_1.html

Примечание.
В демо-версии книги удалены листинги программ.

Здесь организован цикл перебора всех карточек, которые имеются в БД. С использованием параметра `mx-auto` все карточки (независимо от их количества) будут автоматически размещаться в центре экрана. Параметры карточки задают стили, которые подгружены из БД. Поскольку некоторые элементы карточки могут быть не заполнены, они выводятся в теге `if` (проверяется условие наличия данных в элементе). При выводе текста в теле карточки применяется фильтр, который позволяет разделять фрагменты текста на абзацы (начинать абзац с новой строки). В этом случае при вводе текста пользователь, закончив абзац, нажимает клавишу <Enter> для перехода к новой строке.

Теперь в административной панели можно сформировать содержание карточки или нескольких карточек. Пример интерфейса административной панели при формировании содержания карточки приведен на рис. 3.20, а страница d_card_1.html с данными динамической карточки, полученными из БД, показана на рис. 3.21.

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Страница d_card_1.html**.

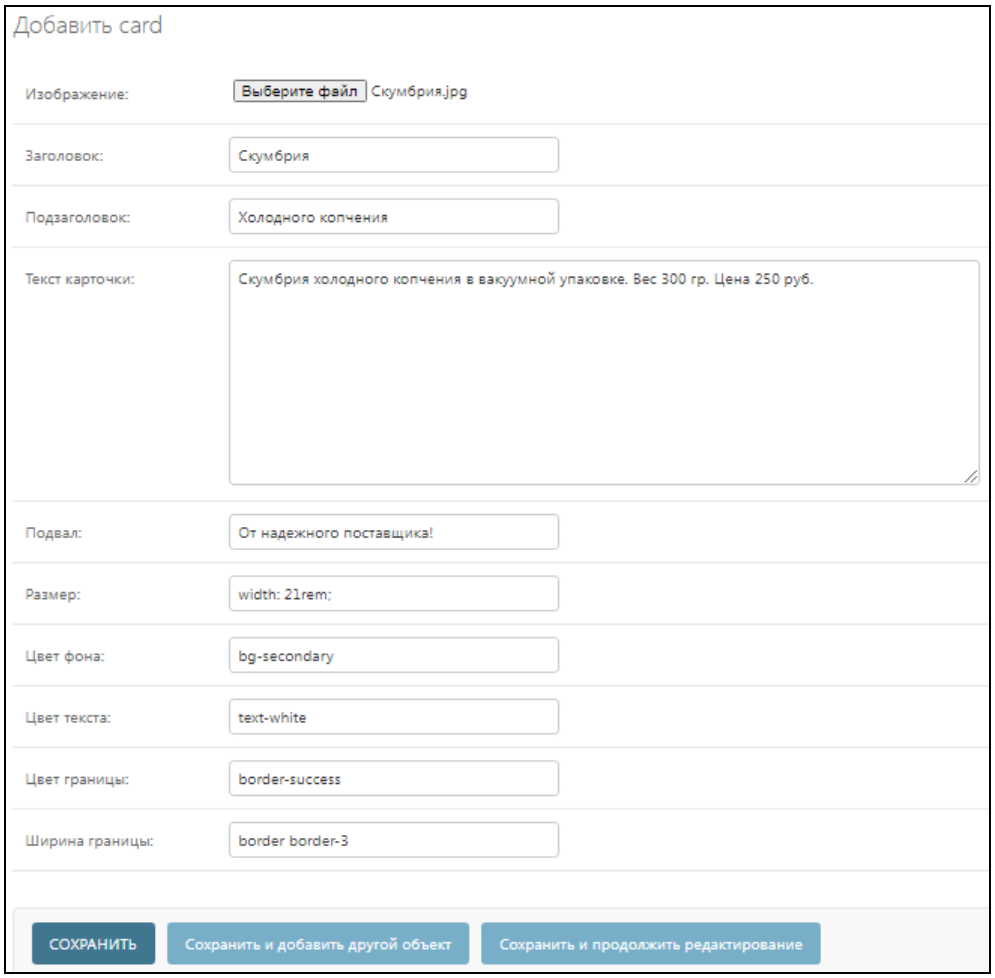


Рис. 3.20. Формирование динамичной карточки в административной панели Django



Рис. 3.21. Внешний вид динамичной карточки (шаблон d_card_1.html)

3.4. Динамичные карточки с заполнением контента через интерфейс пользователя

В предыдущем разделе было показано, как с использованием фреймворка Bootstrap создать динамичную карточку. Для динамичной карточки не нужно менять код страницы, однако недостатком этого метода является то, что карточка обычно привязана к конкретной странице сайта.

Тем не менее, при проектировании сайта можно создать такую модель для динамичных карточек, которая обеспечит создание неограниченного количества этих элементов, позволит добавить их на любую страницу сайта или удалить оттуда, разместить в заданном месте страницы и изменять как содержимое, так и внешний интерфейс элемента. При этом все эти действия можно будет выполнять не только через административную панель сайта, но и через его интерфейс в личном кабинете администраторов сайта. Правда, для этого понадобится создать на сайте личный кабинет для административного персонала и набор необходимых программных модулей для редактирования динамичных карточек.

Посмотрим на упрощенном примере, как это можно сделать, опустив для простоты изложения реализацию личного кабинета, а показав только формирование модели динамичной карточки и модулей для ее редактирования.

Чтобы на одной странице сайта можно было выводить несколько карточек, создим модель `Card_1` — для хранения карточек и функцию `rename_file` — для автоматического изменения имен загружаемых изображений (листинг 3.37).

> ***Пояснение***
> Поскольку публичные серверы не принимают имена изображений на кириллице, наша модель сопровождается функцией автоматического изменения имени файла загружаемого изображения, которая создает на сайте удаленного сервера папку с именем *file*, в которую будут сохраняться загружаемые файлы изображений с именами *file_<id записи в БД>.<расширение>*.

Листинг 3.37. Функция rename_file и модель Card_1

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В модели `Card_1` созданы следующие поля:

- `page` — страница сайта, на которой будет показана карточка;
- `image` — изображение, загруженное на карточку;
- `title` — текст заголовка карточки;

- `subtitle` — текст подзаголовка карточки;
- `text` — текст тела карточки;
- `footer` — текст подвала карточки;
- `card_size` — размер карточки;
- `bg_rgb` — цвет фона карточки;
- `set_text` — параметры текста карточки;
- `border_color` — цвет рамки карточки;
- `border_size` — ширина рамки карточки;
- `group` — группа, в которой будет выведена карточки;
- `link` — ссылка на внешний ресурс Интернета;
- `file` — файл (документ) формата PDF, загруженный на карточку.

Обратите внимание на импорт дополнительных модулей `os` и `uuid4` — они необходимы для работы функции `rename_file`.

В примере нашей модели мы покажем на карточке текст, изображение и ссылку на внешние ресурсы Интернета. Надо также иметь в виду, что у этой модели — в зависимости от характера разрабатываемого приложения — может быть и совершенно другой набор полей.

Не забудьте выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

Чтобы можно было заполнять данными созданную модель, ее надо зарегистрировать в файле admin.py текущего приложения:

```
from django.contrib import admin
from .models import, Card_1
# Регистрация модели Card_1
admin.site.register(Card_1)
```

Для заполнения карточки данными через интерфейс пользователя нам понадобится форма, поэтому в файле forms.py мы создадим форму `CardForm` (листинг 3.38).

Листинг 3.38. Форма CardForm

Примечание.

В демо-версии книги удалены листинги программ.

В этой форме из списка полей для ввода исключено поле `page`, а для некоторых полей изменен формат поля ввода текстовых данных.

3.4.1. Программные модули для создания динамичных карточек

Чтобы получить возможность в интерфейсе сайта создавать динамичные карточки, необходимо сформировать следующий набор модулей:

- для редактирования динамичных карточек;
- для добавления динамичных карточек;
- для обновления динамичных карточек;
- для удаления динамичных карточек.

Код представления (view) `card_edit` для редактирования динамичных карточек приведен в листинге 3.39.

Листинг 3.39. Представление (view) card_edit

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении в объект `tek_page` загружена текущая страница на основе ее идентификатора в БД (`id_page=1`), а в объект `card` из БД загружены все карточки текущей страницы. Эти объекты через словарь `context` передаются в шаблон card_edit.html, который имеет следующий код (листинг 3.40).

Листинг 3.40. Шаблон card_edit.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне создана кнопка Добавить карточку, которая активирует представление с именем card_add по следующей ссылке:

```
<a href="{% url 'master:card_add' %}" class="btn btn-primary">
    <i class="bi bi-database-add fs-5"></i> Добавить карточку</a>
```

После этой кнопки идет таблица, в которой отображается информация о созданных карточках, — в каждой ее строке имеются две кнопки:

- `Изменить` — для изменения сведений о карточке;
- `Удалить` — для удаления для удаления карточки.

Эти кнопки активируют представления с именами `card_upd` и `card_del` по следующим ссылкам:

```
<a href="{% url 'master:card_upd' id=obj.id %}"
   class="btn btn-primary btn-sm my-1">
    <i class="bi bi-repeat"></i>Изменить</a>
<a href="{% url 'mastercard_del' id=obj.id %}"
   class="btn btn-danger btn-sm my-1">
    <i class="bi-trash3"></i> Удалить</a>
```

Код представления `card_add` приведен в листинге 3.41.

Листинг 3.41. Представление (view) card_add

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создается объект `form`, в который загружается форма `CardForm()`. Эта форма через словарь `context` передается в шаблон card_form.html, который имеет следующий код (листинг 3.42).

Листинг 3.42. Шаблон card_form.html

Примечание.

В демо-версии книги удалены листинги программ.

Этот шаблон представляет форму, в которой в виде таблицы отображаются поля для ввода данных: `{ form.as_table }`. Здесь же имеются две кнопки: `Отмена` — для отмены сохранения введенной информации и возврата к предыдущему окну и `Сохранить` — для сохранения введенной информации и возврата к предыдущему окну.

При возврате из этой формы в представление `card_add` там происходит сохранение в БД сведений о динамичной карточке.

Код представления `card_upd` приведен в листинге 3.43.

Листинг 3.43. Представление (view) card _upd

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор редактируемой карточки (id), на основе которого в объект obj из БД загружаются сведения об этой карточке. После чего создается объект form, в который загружается форма CardForm() с информацией о текущей карточке. Эта форма через словарь context передается в шаблон card_form.html. При возврате из этого шаблона-формы в представление card_upd там происходит сохранение в БД сведений о динамичной карточке.

Код представления card_del приведен в листинге 3.44.

Листинг 3.44. Представление (view) card_del

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор удаляемой карточки (id), на основе которого в объект obj из БД загружаются сведения об этой карточке. После чего в объект del_prod загружается заголовок удаляемой карточки, который через словарь content передается в универсальный шаблон del_yesno.html (см. листинг 3.17). По нажатию кнопок в этом шаблоне происходит возврат в представление card_del, где в случае нажатия на кнопку Удалить выполняется удаление информации о карточке из БД.

Для активации всех этих представлений нужно в файле urls.py создать соответствующие URL-маршруты (листинг 3.45).

Листинг 3.45. Маршруты для редактирования динамичных карточек

Примечание.

В демо-версии книги удалены листинги программ.

Для вызова представления card_edit надо в меню одной из страниц сайта создать следующую ссылку (листинг 3.46).

Листинг 3.46. Ссылка для редактирования динамичных карточек

Примечание.

В демо-версии книги удалены листинги программ.

3.4.2. Пример формирования динамичных карточек через интерфейс пользователя

Проверим работу созданных нами модулей.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Представление card_edit**.

При первоначальной загрузке шаблон card_edit.html будет выглядеть так, как показано на рис. 3.22). Поскольку в БД нет ни одной записи о динамичных карточках, то таблица с параметрами карточек пустая.



Рис. 3.22. Внешний вид шаблона card_edit1.html при первоначальной загрузке

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

По нажатию на кнопку **Добавить карточку** откроется шаблон-форма card_form.html (рис. 3.23).

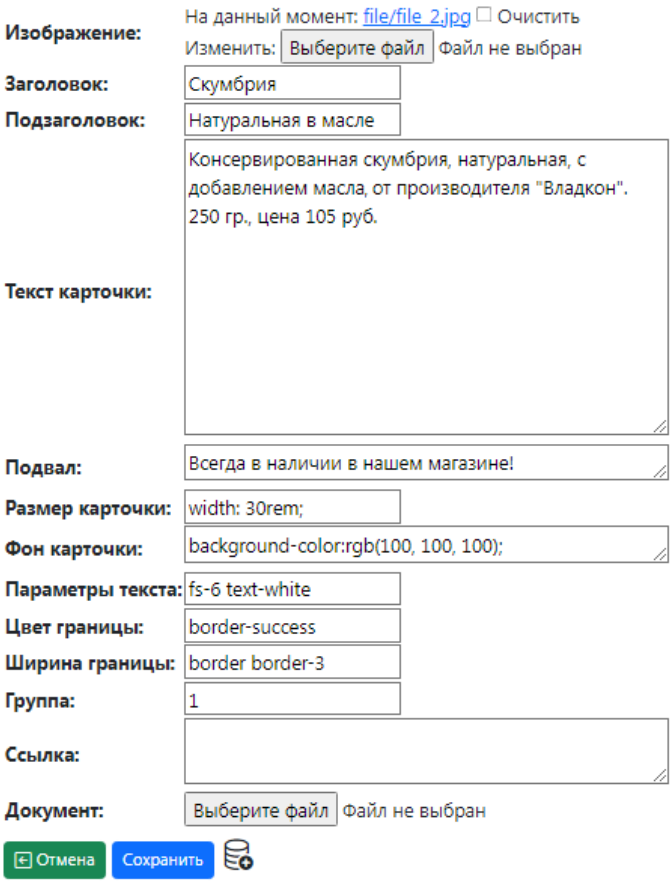


Рис. 3.23. Шаблон-форма card_add.html для формирования содержимого динамичной карточки

По нажатию на кнопку **Сохранить** параметры этой карточки будут сохранены в БД, а пользователь будет возвращен в шаблон card_edit.html, где в таблице отразятся введенные данные (рис. 3.24).



Рис. 3.24. Шаблон card_edit.html после ввода содержимого динамичной карточки

По нажатию на кнопку **Изменить** откроется форма для редактирования выбранной карточки, а по нажатию на кнопку **Удалить** карточка будет удалена из БД.

Обратите внимание: при создании карточки в поле **Группа** стояла цифра 1. Это поле служит для группировки карточек с выводом этой группы в заданную панель страницы сайта, — если, например, на странице сайта имеются три панели: левая боковая, центральная и правая боковая. При этом поле **Группа** служит для идентификации этих панелей. В качестве идентификатора можно выбрать любые числа или символы. В нашем примере в качестве идентификаторов панелей выбраны числа: 0 — левая боковая панель, 1 — центральная панель, 2 — правая боковая панель. В рассматриваемом случае карточке была назначена группа 1 — значит, она будет отображаться в центральной панели.

3.4.3. Модули для вывода динамичных карточек на страницы сайта

Итак, мы создали все модули, которые позволяют формировать динамичные карточки. Теперь покажем, как динамичные карточки можно вывести на страницу сайта. Для этого создадим тестовую страницу (шаблон) с именем examples_2.html, а также соответствующие маршрут и представление.

Код представления `examples_2` приведен в листинге 3.47.

Листинг 3.47. Представление examples_2

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создаются три объекта: `cards_l`, `cards` и `cards_r` — в которые загружаются карточки соответственно левого блока (левой панели), центрального блока (центральной панели), правого блока (правой панели). Все эти объекты через словарь `context` передаются в шаблон examples_2.html.

Для активации представления `examples_2` в файле urls.py нужно создать URL-маршрут (листинг 3.48).

Листинг 3.48. URL-маршрут с именем examples_2

Примечание.

В демо-версии книги удалены листинги программ.

Код шаблона examples_2.html приведен в листинге 3.49.

Листинг 3.49. Шаблон examples_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Шаблон examples_2.html представляет собой демонстрационную страницу, состоящую из трех панелей: левой боковой, центральной и правой боковой. В каждую панель с использованием тега `include` вставлен другой шаблон:

- в левую боковую панель — шаблон card_show_l.html;
- в центральную панель — шаблон card_show.html;
- в правую боковую панель — шаблон card_show_r.html.

Коды шаблонов card_show_l.html, card_show.html и card_show_r.html представлены в листингах 3.50–3.52.

Листинг 3.50. Шаблон card_show_l.html

Примечание.
В демо-версии книги удалены листинги программ.

Листинг 3.51. Шаблон card_show.html

Примечание.
В демо-версии книги удалены листинги программ.

Листинг 3.52. Шаблон card_show_r.html

Примечание.
В демо-версии книги удалены листинги программ.

Все три эти шаблона имеют одинаковую структуру. В каждом шаблоне с использованием тега Django `for` организован цикл по карточкам (`card_l`, `card`, `card_r`). В теле элементов карточек задействуется тег Django `if`, в котором проверяется наличие или отсутствие в БД сведений об элементе карточки. Если сведения об элементе в БД присутствуют, то он будет отображен в шаблоне.

* * *

Итак, мы рассмотрели код всех модулей и шаблонов, которые обеспечивают формирование и вывод динамичных карточек. Имея такой набор, администраторы сайта могут, не меняя программный код, добавить любое количество карточек на любую страницу сайта (и даже в любую панель страницы), изменять содержание карточек и менять дизайн элементов карточек. При удалении карточки из БД она автоматически исчезнет со страниц сайта. Рассмотрим теперь, как работает наш программный код, и как динамичные карточки выглядят на страницах сайта.

3.4.4. Пример отображения динамичных карточек на страницах сайта

Проверим работу созданных нами модулей.

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Страница examples_2.html**.

При вызове модуля отображения динамичных карточек шаблон examples_2.html (при наличии в БД сведений только об одной карточке) будет выглядеть так, как показано на рис. 3.25).



Рис. 3.25. Внешний вид шаблона examples_2.html с динамичной карточкой в центральной панели

Теперь с использованием модулей ввода и корректировки динамичных карточек добавим на эту страницу еще две карточки — в левую и в правую панель (рис. 3.26).



Рис. 3.26. Формирование содержимого динамичных карточек для левой и правой панелей страницы сайта

После этого страница сайта будет выглядеть следующим образом (рис.3.27).



Рис. 3.27. Внешний вид шаблона examples_2.html с динамичными карточками в разных панелях страницы

* * *

Подводя итоги этого раздела, можно сделать следующие выводы. Написание программного кода для формирования динамичных карточек — достаточно трудоемкий процесс, однако эти трудозатраты окупаются при сопровождении сайта. Администраторы сайта получают возможность добавлять любое количество динамичных карточек в любые фрагменты любой страницы сайта, а также изменять содержимое карточек и их оформление. При этом им не требуется вносить какие-либо изменения в код программных модулей сайта.

3.5. Динамичная карусель с заполнением контента через административную панель

При проектировании сайта также можно создать и карусель с динамичной загрузкой слайдов, содержащих изображения и текстовую информациею. В этом случае администраторы сайта через административную панель смогут изменять как количество слайдов, так и их содержание (изображения и поясняющие надписи).

При размещении на страницах сайта динамичной карусели необходимо все ее элементы хранить в базе данных — для этого мы сейчас создадим модель `Carousel` и функцию `rename_image`, позволяющую изменять имена изображений, загруженных на слайды карусели (листинг 3.53).

> ***Пояснение***
> Поскольку публичные серверы не принимают имена изображений на кириллице, наша модель сопровождается функцией автоматического изменения имени загружаемого изображения, которая создает на сайте удаленного сервера папку с именем *image_carousel*, в которую будут сохраняться загружаемые изображения с именами *img_<id записи в БД>.<расширение>*.

Листинг 3.53. Модель Carousel и функция rename_image

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В модели `Carousel` созданы следующие поля:

- `title_1` — верхняя надпись на изображении карусели;
- `title_2` — средняя надпись на изображении карусели;
- `title_3` — нижняя надпись на изображении карусели;
- `image` — изображение на слайде карусели.

Надо не забыть выполнить и процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

Чтобы персонал мог удаленно менять содержимое слайдов карусели через административную панель, модель `Carousel` нужно зарегистрировать в файле admin.py (листинг 3.54).

Листинг 3.54. Регистрация модели Carousel в файле admin.py

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

Для демонстрации динамической карусели в меню или в теле одного из шаблонов надо создать ссылку для вызова страницы, содержащей карусель. Эта ссылка может иметь следующий код (листинг 3.55).

Листинг 3.55. Ссылка для вызова динамической карусели

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этой ссылке будет инициирован URL-маршрут (листинг 3.56).

Листинг 3.56. URL-маршрут carousel

Примечание.
В демо-версии книги удалены листинги программ.

При адресации к этому маршруту будет инициировано представление `carousel` (листинг 3.57).

Листинг 3.57. Представление carousel

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении создается объект `images`, в который загружаются все записи из модели `Carousel` (то есть все слайды). Затем загруженные слайды карусели через словарь передаются в шаблон с именем carousel.html (листинг 3.58).

Листинг 3.58. Шаблон carousel.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне выделены два блока: блок, содержащий карусель, и блок с кнопками прокрутки слайдов. С использованием тега `{% if forloop.first %}active{% endif %}` первый слайд (независимо от их количества) будет активным и показан первым. Поскольку некоторые элементы слайда могут быть не заполнены, то они выводятся в теге `if` (проверяется условие наличия данных в элементе). В теге цикла загрузки слайдов карусели содержатся ссылки на фоновые изображения слайдов и на надписи, которые будут показаны над фоновыми изображениями.

Теперь в административной панели можно сформировать содержание слайдов карусели. Пример интерфейса административной панели при формировании содержания слайдов карусели приведен на рис. 3.28, а страница carousel.html с данными динамичной карусели, полученными из БД, показана на рис. 3.29).

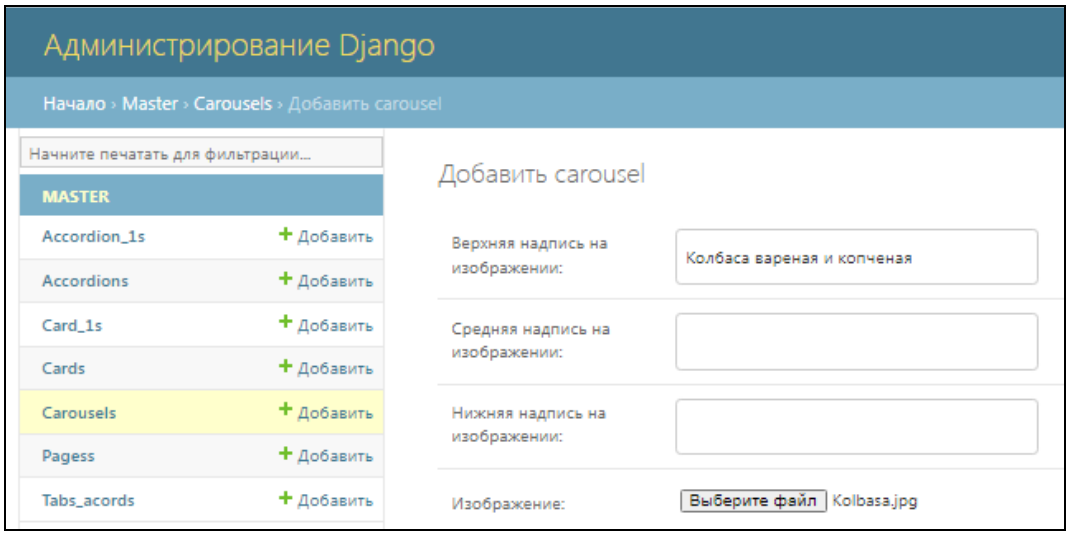


Рис. 3.28. Формирование слайда динамичной карусели в административной панели Django

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Страница carousel.html**.

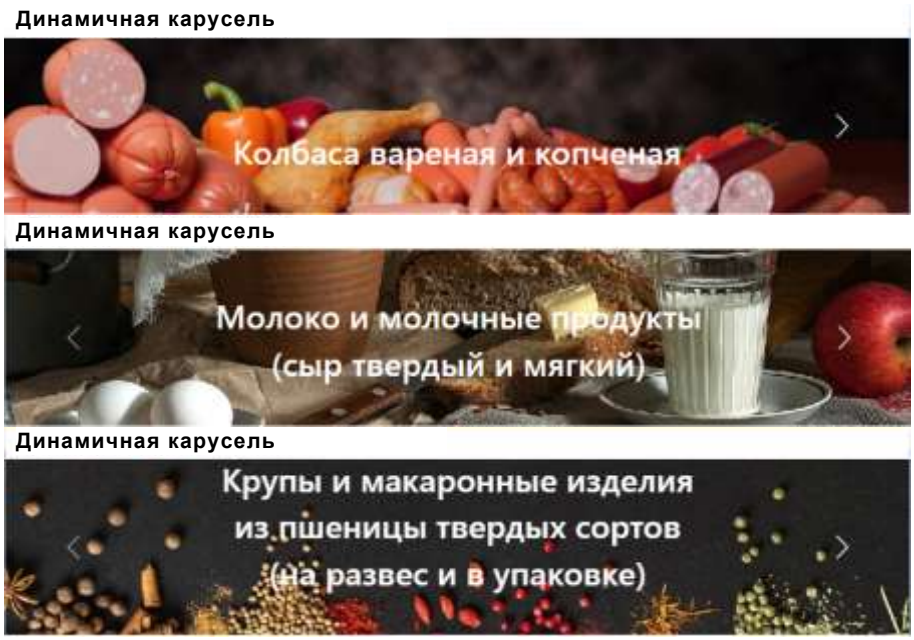


Рис. 3.29. Внешний вид слайдов динамической карусели

3.6. Динамичная карусель с заполнением контента через интерфейс пользователя

В предыдущем разделе было показано, как с использованием фреймворка Bootstrap создать динамичную карусель. Для динамичной карусели не нужно менять код страницы, однако недостатком этого метода является то, что динамичная карусель обычно привязана к конкретной странице сайта.

Тем не менее, при проектировании сайта можно создать такую модель для динамичных каруселей, которая обеспечит создание неограниченного количества этих элементов, позволит добавить их на любую страницу сайта или удалить оттуда и изменять как содержимое, так и внешний интерфейс элемента. При этом все эти действия можно будет выполнять не только через административную панель сайта, но и через его интерфейс в личном кабинете администраторов сайта. Правда, для этого понадобится создать на сайте личный кабинет для административного персонала и набор необходимых программных модулей для редактирования динамичных каруселей.

Посмотрим на упрощенном примере, как это можно сделать, опустив для простоты изложения реализацию личного кабинета, а показав только формирование модели динамичной карусели и модулей для ее редактирования.

Чтобы на одной странице сайта можно было выводить несколько несколько каруселей, создадим модель `Carousel_1` — для хранения каруселей и функцию `rename_carousel` — для автоматического изменения имен изображений, загруженных на слайды карусели (листинг 3.59).

Пояснение

Поскольку публичные серверы не принимают имена изображений на кириллице, наша модель сопровождается функцией автоматического изменения имени файла загружаемого изображения, которая создает на сайте удаленного сервера папку с именем *carousel*, в которую будут сохраняться загружаемые файлы изображений с именами *car_<id записи в БД>.<расширение>*.

Листинг 3.59. Функция rename_carousel и модель Carousel_1

Примечание.

В демо-версии книги удалены листинги программ.

В модели Carousel_1 созданы следующие поля:

- `page` — страница сайта, на которой будет показана карусель;
- `title_1` — верхняя надпись на изображении;
- `title_2` — средняя надпись на изображении;
- `title_3` — нижняя надпись на изображении;
- `image` — изображение, загруженное на слайд карусели;
- `set_text` — параметры текста слайда карусели (цвет и размер шрифта).

Обратите внимание на импорт дополнительных модулей `os` и `uuid4` — они необходимы для работы функции `rename_carousel`.

Надо также иметь в виду, что у этой модели — в зависимости от характера разрабатываемого приложения — может быть и совершенно другой набор полей.

Не забудьте выполнить процедуру миграции

```
python manage.py makemigrations
python manage.py migrate
```

Чтобы можно было заполнять данными созданную модель, ее нужно зарегистрировать в файле admin.py текущего приложения:

```
from django.contrib import admin
from .models import, Carousel_1
# Регистрация модели Carousel_1
admin.site.register(Carousel_1)
```

Для заполнения слайдов карусели данными через интерфейс пользователя нам понадобится форма, поэтому в файле forms.py мы создадим форму `CarouselForm` (листинг 3.60).

Листинг 3.60. Форма CarouselForm

Примечание.

В демо-версии книги удалены листинги программ.

В этой форме из списка полей для ввода исключено поле `page`, для некоторых полей изменен формат поля ввода текстовых данных.

3.6.1. Программные модули для создания слайдов динамичной карусели

Чтобы получить возможность в интерфейсе сайта создавать динамичные карусели, необходимо сформировать следующий набор модулей:

- для редактирования динамичных каруселей;
- для добавления динамичных каруселей;
- для обновления динамичных каруселей;
- для удаления динамичных каруселей.

Код представления (view) `carousel_edit` для редактирования динамичных каруселей приведен в листинге 3.61.

Листинг 3.61. Представление (view) carousel_edit

Примечание.

В демо-версии книги удалены листинги программ.

```
В этом представлении в объект tek_page загружена текущая страница на основе ее
идентификатора в БД (id_page=1), а в объект carousel из БД загружены все кару-
```

сели текущей страницы. Эти объекты через словарь context передаются в шаблон carousel_edit.html, который имеет следующий код (листинг 3.62).

Листинг 3.62. Шаблон carousel_edit.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне создана кнопка Добавить слайд, которая активирует представление с именем carousel_add по следующей ссылке:

```
<a href="{% url 'master:carousel_add' %}"
           class="btn btn-primary">
            <i class="bi bi-database-add fs-5"></i> Добавить слайд</a>
```

После этой кнопки идет таблица, в которой отображается информация о созданных слайдах карусели, — в каждой ее строке имеются две кнопки:

- Изменить — для изменения сведений о слайде;
- Удалить — для удаления слайда.

Эти кнопки активируют представления с именами: carousel_upd и carousel_del по следующим ссылкам:

```
<a href="{% url 'master:carousel_upd' id=obj.id %}"
                       class="btn btn-primary btn-sm my-1">
                        <i class="bi bi-repeat"></i> Изменить</a>
                    <a href="{% url 'master:carousel_del' id=obj.id %}"
                       class="btn btn-danger btn-sm my-1">
                        <i class="bi-trash3"></i> Удалить</a>
```

Код представления carousel_add приведен в листинге 3.63.

Листинг 3.63. Представление (view) carousel_add

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создается объект form, в который загружается форма CarouselForm(). Эта форма через словарь context передается в шаблон carousel_form.html, который имеет следующий код (листинг 3.64).

Листинг 3.64. Шаблон carousel_form.html

Примечание.

В демо-версии книги удалены листинги программ.

Этот шаблон представляет форму, в которой в виде таблицы отображаются поля для ввода данных: { form.as_table }. Здесь же имеются две кнопки: Отмена — для отмены сохранения введенной информации и возврата к предыдущему окну и «Сохранить — для сохранения введенной информации и возврата к предыдущему окну.

При возврате из этой формы в представление carousel_add там происходит сохранение в БД сведений о слайде динамичной карусели.

Код представления carousel_upd приведен в листинге 3.65.

Листинг 3.65. Представление (view) carousel_upd

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор редактируемого слайда карусели (id), на основе которого в объект slide из БД загружаются сведения об этом слайде. После чего создается объект form, в который загружается форма CarouselForm() с информацией о текущем слайде карусели. Эта форма через словарь context передается в шаблон carousel_form.html. При возврате из этого шаблона-формы в представление carousel_upd там происходит сохранение в БД сведений о слайде динамичной карусели.

Код представления carousel_del приведен в листинге 3.66.

Листинг 3.66. Представление (view) carousel_del

Примечание.

В демо-версии книги удалены листинги программ.

Данное представление получает идентификатор удаляемого слайда карусели (id), на основе которого в объект obj из БД загружаются сведения об этом слайде. После чего в объект del_prod загружается надпись верхней строки удаляемого слайда, которая через словарь content передается в универсальный шаблон del_yesno.html (см. листинг 3.17). По нажатию кнопок в этом шаблоне происходит возврат в представление carousel_del, где в случае нажатия на кнопку Удалить выполняется удаление информации о текущем сладе карусели из БД.

Для активации всех этих представлений нужно в файле urls.py создать соответствующие URL-маршруты (листинг 3.67).

Листинг 3.67. Маршруты для редактирования динамичных карточек

Примечание.

В демо-версии книги удалены листинги программ.

Для вызова представления carousel_edit надо в меню одной из страниц сайта создать следующую ссылку (листинг 3.68).

Листинг 3.68. Ссылка для редактирования динамичных каруселей

Примечание.

В демо-версии книги удалены листинги программ.

3.6.2. Пример формирования слайдов динамичной карусели через интерфейс пользователя

Проверимработу созданных нами модулей.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Представление carousel_edit**.

При первоначальной загрузке шаблон carousel_edit.html будет выглядеть так, как показано на рис. 3.30. Поскольку в БД нет ни одной записи о слайдах динамичной карусели, то таблица с параметрами слайдов карусели пустая.



Рис. 3.30. Внешний вид шаблона carousel _edit.html при первоначальной загрузке

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы..

По нажатию на кнопку **Добавить слайд** откроется шаблон-форма carousel_form.html (рис. 3.31).

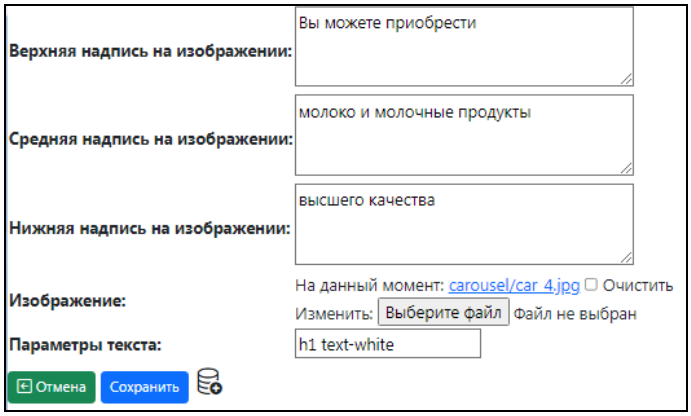


Рис. 3.31. Шаблон-форма carousel_form.html для формирования слайда динамичной карусели

По нажатию на кнопку **Сохранить** параметры этого слайда будут сохранены в БД, а пользователь будет возвращен в шаблон card_edit.html, где в таблице отразятся введенные данные. Таблица с введенными данными для трех слайдов карусели приведена на рис. 3.32.



Рис. 3.32. Шаблон carousel_edit.html после ввода содержимого трех слайдов динамичной карусели

По нажатию на кнопку **Изменить** откроется форма для редактирования выбранного слайда, а по нажатию на кнопку **Удалить** слайд будет удален из БД.

3.6.3. Модули для вывода динамичных каруселей на страницы сайта

Итак, мы создали все модули, которые позволяют создать динамичную карусель. Теперь покажем, как динамичную карусель можно вывести на страницу сайта. Для этого создадим тестовую страницу (шаблон) с именем examples_3.html, а также соответствующие маршрут и представление.

Код представления `examples_3` приведен в листинге 3.69.

Листинг 3.69. Представление examples_3

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создается объект `carousel`, в который загружаются слайды карусели страницы сайта с идентификатором `id_page=1`. Все эти слайды через словарь `context` передаются в шаблон examples_3.html.

Для активации представления `examples_3` в файле urls.py нужно создать URL-маршрут (листинг 3.70).

Листинг 3.70. URL-маршрут с именем examples_3

Примечание.

В демо-версии книги удалены листинги программ.

Код шаблона examples_3.html приведен в листинге 3.71.

Листинг 3.71. Шаблон examples_3.html

Примечание.

В демо-версии книги удалены листинги программ.

Шаблон examples_3.html представляет собой демонстрационную страницу, состоящую из трех панелей: левой боковой, центральной и правой боковой. В центральную панель с использованием тега `include` вставлен другой шаблон: carousel_show.html. Левая и правая боковые панели пустые. Код шаблона carousel_show.html представлен в листинге 3.72.

Листинг 3.72. Шаблон carousel_show.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне с использованием тега Django for организован цикл по слайдам карусели. В теле этого цикла задействован тег Django if, в котором проверяется наличие или отсутствие в БД сведений об элементе слайда. Если сведения об элементе в БД присутствуют, то он будет отображен в шаблоне.

* * *

Итак, мы рассмотрели код всех модулей и шаблонов, которые обеспечивают формирование и вывод динамичных каруселей. Имея такой набор, администраторы сайта могут, не меняя программный код, добавить элемент «карусель» на любую страницу сайта, изменять содержание слайдов карусели и менять дизайн выводимого текста. При удалении карусели из БД она автоматически исчезнет со страниц сайта. Рассмотрим теперь, как работает наш программный код, и как динамичная карусель выглядят на страницах сайта.

3.6.4. Примеры отображения динамичных каруселей на страницах сайта

Проверим работу созданных нами модулей.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 3** и перейти по ссылке **Страница examples_3.html**.

При вызове модуля отображения динамичной карусели шаблон examples_3.html будет выглядеть так, как показано на рис. 3.33. Как здесь можно видеть, текстовая информация на каждом слайде имеет разное цветовое оформление.

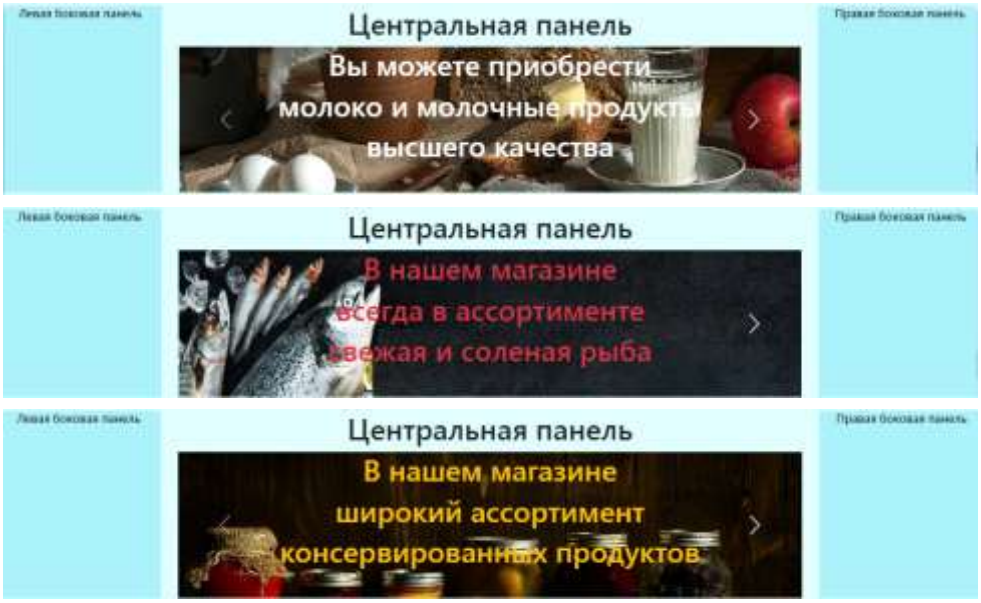


Рис. 3.33. Внешний вид шаблона examples_3.html с тремя слайдами динамичной карусели в центральной панели

Подводя итого этого раздела, можно сделать следующие выводы. Написание программного кода для формирования динамичных каруселей — достаточно трудоемкий процесс, однако эти трудозатраты окупаются при сопровождении сайта. Администраторы сайта получают возможность добавлять динамичные карусели на любую страницу сайта, а также изменять содержимое слайдов и их оформление. При этом им не требуется вносить какие-либо изменения в код программных модулей сайта.

3.7. Краткие итоги

В этой главе мы познакомились с методами и программными модулями создания динамичных элементов Bootstrap. Параметры настройки интерфейса и контент таких элементов хранятся в БД, и они могут обновляться удаленно без изменения программного кода. Здесь была показана реализация только трех элементов Bootstrap (аккордеон, карточки и карусель), но аналогичный подход вы можете использовать и для других элементов.

В интерфейсе многих сайтов используются древовидные структуры, которые позволяют строить многоуровневые меню и каталоги продуктов (товаров). Такие элементы интерфейса могут отображаться либо в развернутом виде, либо в виде свернутого дерева. Пользователю при этом предоставляется возможность по своему усмотрению раскрыть любую ветвь дерева, выбрать нужный элемент и отфильтровать большой массив данных по выбранному признаку. Методам создания древовидных структур в Django и посвящена следующая глава.

Глава 4. Древовидные структуры в Django

Древовидные структуры для представления данных в Интернете весьма популярны — для конечных пользователей это удобный и привлекательный элемент. Однако программирование модулей для создания древовидных структур — процесс достаточно сложный и трудоемкий. Впрочем, в Django для этих целей имеется специализированный пакет Django-mptt, который значительно упрощает реализацию древовидных структур.

В процессе изучения материалов этой главы вы узнаете:

- в каком виде пакет Django-mptt формирует таблицу БД для хранения древовидных данных;
- как создать модель для хранения древовидных данных и заполнить ее информацией через административную панель Django (на примере категорий продуктов интернет-магазина);
- как отобразить статичное дерево категорий продуктов в шаблонах Django;
- как отобразить динамичное дерево категорий продуктов в шаблонах Django;
- как разработать программные модули для формирования дерева категорий продуктов через интерфейс пользователя;
- как создать список объектов с возможностью их фильтрации через древовидные каталоги (на примере списка продуктов интернет-магазина);
- как создать интерфейс для фильтрации сведений о продуктах с использованием статичного и динамичного древовидного каталога категорий продуктов;
- как создавать древовидные меню с использованием плагина megamenu js;
- как фильтровать данные в БД с использованием панели инструментов и древовидного каталога категорий.

Итак, приступим к знакомству с древовидными структурами данных в Django.

4.1. Использование Django-mptt для создания древовидных структур

Каждый разработчик рано или поздно сталкивается с проблемой хранения и выборки данных в виде дерева. Самый распространенный вариант поиска той или иной ветки дерева — запрос наличия потомков или родителя у текущего элемента. У этого метода есть свой плюс — простота изменения дерева (удаление, добавление, перенос веток). Однако есть и минусы — это ресурсоемкие и медленные алгоритмы выборки. Как правило, они представляют собой рекурсивные функции с множеством мелких выборок из БД, либо с одной выборкой, но с последующей циклической и рекурсивной обработкой данных и их упорядочиванием. Эти процессы при сложной структуре дерева могут занимать много времени.

Но давайте рассмотрим еще один метод реализации древовидной структуры данных в БД — MPTT (Modified Preorder Tree Traversal, модифицированный предварительный обход дерева). Этот метод очень похож на Nested sets (вложенные множества), с той лишь разницей, что тут не хранится уровень вложенности. Применяя этот метод, можно выбрать любую ветку или любую часть дерева, выполнив всего лишь один SQL-запрос. Так же просто можно выбрать и путь к любому элементу дерева.

Многие интернет-приложения используют иерархические структуры данных. Например, на сайтах электронной коммерции представление категорий различных объектов в виде дерева является полезным инструментом для организации структуры товаров и услуг. Это позволяет группировать объекты (товары, услуги) по категориям и подкатегориям, упрощая навигацию и их поиск для пользователей. Многоуровневые меню часто содержат иерархические данные, которыми необходимо эффективно управлять, чтобы обеспечить удобную навигацию по сайту.

Один из способов реализации деревьев категорий в Django, который дает возможность упростить работу с деревьями объектов, — использование специализированного пакета Django-mptt. Этот пакет для организации структуры деревьев задействует алгоритм MPTT, позволяющий эффективно хранить и извлекать информацию из деревьев, упрощая написание запросов и ускоряя выполнение операций с записями в БД.

Для начала работы с Django-mptt необходимо установить этот пакет при помощи менеджера пакетов pip. Выполните команду:

```
pip install django-mptt
pip install django-mptt-admin
```

Здесь `django-mptt` — основной пакет для работы с древовидными структурами, а `django-mptt-admin` — дополнительный пакет, который обеспечивает удобное заполнение дерева в административной панели Django.

После установки пакета необходимо добавить его в раздел `INSTALLED_APPS` файла settings.py с помощью следующего кода:

```
INSTALLED_APPS = [
    ...
    'mptt',
         'django_mptt_admin',
    ...
]
```

Для создания модели дерева категорий в Django-mptt необходимо использовать абстрактный базовый класс `MPTTModel` — он предоставляет необходимые поля и методы для хранения и организации структуры дерева:

```
from django.db import models
from mptt.models import MPTTModel, TreeForeignKey


class Category(MPTTModel):
    name = models.CharField(max_length=50, unique=True)
    parent = TreeForeignKey('self', on_delete=models.CASCADE,
    null=True, blank=True,
    related_name='children')

    class MPTTMeta:
        order_insertion_by = ['name']
```

В этом примере кода создается модель `Category`, использующая базовый класс `MPTTModel`. Модель содержит поля `name` и `parent`, которые определяют название категории и родительскую категорию соответственно. Поле `parent` является внешним ключом на ту же модель `Category`, что позволяет строить древовидную структуру. Параметр `related_name='children'` определяет имя обратной связи для доступа к дочерним элементам категории.

Для того чтобы Django-mptt мог правильно организовать дерево категорий, необходимо в модели `Category` указать, по какому полю сортировать элементы дерева. Для этого в класс `Category` добавлен вложенный класс `MPTTMeta` с полем `order_insertion_by`, которое определяет сортировку по полю `name`.

Создав модель `Category`, можно наполнить данными иерархию категорий либо в административной панели Django, либо создать для этого собственный программный код.

Рассмотрим древовидную структуру данных на примере категорий продуктов `Category` интернет-магазина (рис. 4.1).

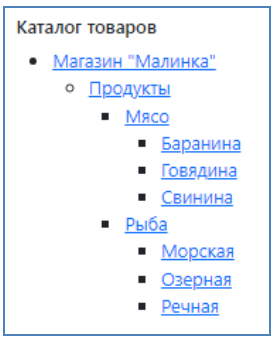


Рис. 4.1. Пример древовидной структуры каталога товаров (категории продуктов) интернет-магазина

В модели `Category` пакета Django-mptt каждая ветка в дереве категорий продуктов в базе данных будет иметь несколько идентификационных атрибутов:

- `id` — идентификатор записи;
- `name` — имя ветки дерева;
- `lft` — номер левой части ветки (при обходе дерева сверху вниз);
- `rght` — номер правой части ветки (при обходе дерева снизу вверх);
- `tree_id` — идентификатор дерева;
- `level` — номер уровня, на котором находится ветка дерева;
- `parent_id` — идентификатор родительской ветки.

Принцип нумерации левой и правой части веток дерева представлен на рис. 4.2.

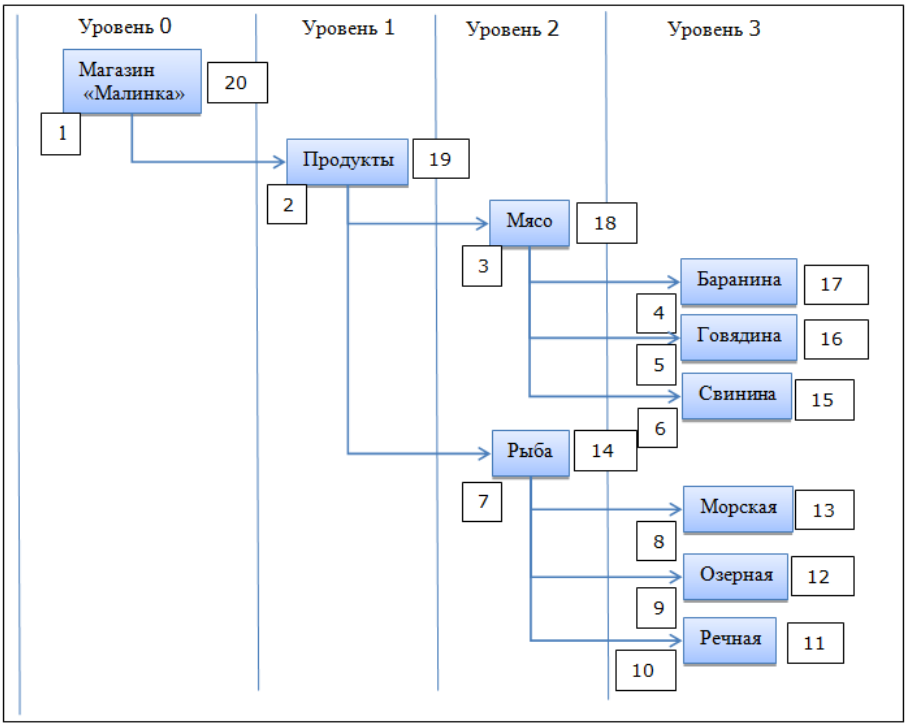


Рис. 4.2. Пример древовидной структуры каталога товаров интернет-магазина (категории продуктов `Category`)

В этом примере дерево категорий продуктов имеет 3 уровня. Каждая ветка дерева несет *слева* уникальный номер от 1 до 10 (при обходе дерева сверху вниз) и *справа* — уникальный номер от 11 до 20 (при обходе дерева снизу вверх).

Посмотрим, как такое дерево будет представлено в БД в таблице `Category` (табл. 4.1).

Таблица 4.1. *Содержание таблицы категорий товаров* `Category` *в базе данных*

id	name	lft	rght	tree_id	level	parent_id
1	Магазин «Малинка»	1	20	1	0	NULL

2	Продукты	2	19	1	1	1
3	Мясо	3	18	1	2	2
4	Баранина	4	17	1	3	3
5	Говядина	5	16	1	3	3
6	Свинина	6	15	1	3	3
7	Рыба	7	14	1	2	2
8	Морская	8	13	1	3	7
9	Озерная	9	12	1	3	7
10	Речная	10	11	1	3	7

В этой таблице ветки дерева отсортированы по полю `id`. При заполнении структуры дерева конкретными данными программисту нужно ввести значение только одного поля (`name`) и указать имя родительской ветки, — значения всех остальных полей Django сформирует самостоятельно.

При заполнении данными дерева категорий продуктов надо сначала создать корневой элемент (узел). У корневого элемента дерева нет родителя, поэтому при формировании значения для корня дерева родительская категория не задается (для корня дерева поле `parent_id` будет иметь значение `NULL`). Сформировать корневую категорию для дерева можно с использованием следующего метода класса `Category`:

```
root_category = Category.objects.create(name="Магазин «Малинка»")
```

Затем, когда создаются все дочерние категории, для них нужно задать имя категории и указать имя родительской категории — например:

```
Category.objects.create(name="Продукты", parent=root_category)
```

Примечание

В древовидной структуре данных может быть не один, а несколько корневых узлов.

Когда программист заполняет структуру дерева для своего приложения самостоятельно, ему нет необходимости писать собственный программный код, — для этих целей можно воспользоваться тем инструментарием, который встроен в административную панель Django. Однако если требуется предоставить возможность формирования дерева категорий конечным пользователям приложения, то для этого необходимо написать соответствующий программный код.

4.2. Формирования дерева категорий продуктов с использованием административной панели Django

Для начала необходимо создать модель для хранения данных о категориях продуктов. Программный код такой модели (`Category_pr`) приведен в листинге 4.1.

Листинг 4.1. Модель категорий продуктов Category_pr

Примечание.

В демо-версии книги удалены листинги программ.

Обратите внимание, что для работы с древовидными структурами нужно в файле models.py обязательно выполнить импорт библиотек `MPTTModel` и `TreeForeignKey`.

В этом программном модуле на основе базового класса `MPTTModel` создан класс `Categiry_pr` (категории продуктов). Модель `Categiry_pr` содержит два поля: `name` — имя категории и `parent` — имя родительской категории. В административной панели Django для этой модели будет отображаться только одно поле — имя категории (`self.name`), а в интерфейсе пользователя дерево категорий будет отсортировано по имени категории (`order_insertion_by = ['name']`).

Чтобы обеспечить возможность наполнения данными дерева категорий в административной панели, этоу модель нужно зарегистрировать в модуле admin.py (листинг 4.2).

Листинг 4.2. Регистрация модели Category_pr в файле admin.py

Примечание.

В демо-версии книги удалены листинги программ.

Обратите внимание, что для регистрации древовидных моделей потребовался импорт дополнительной библиотеки (`DjangoMpttAdmin`) и создание промежуточного класса (`CategoryAdmin`).

Создав код модели `Categiry_pr`, не забудьте выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

Реализовав миграцию, можно открыть административную панель Django и заполнить данными древовидную структуру категорий продуктов (рис. 4.3).

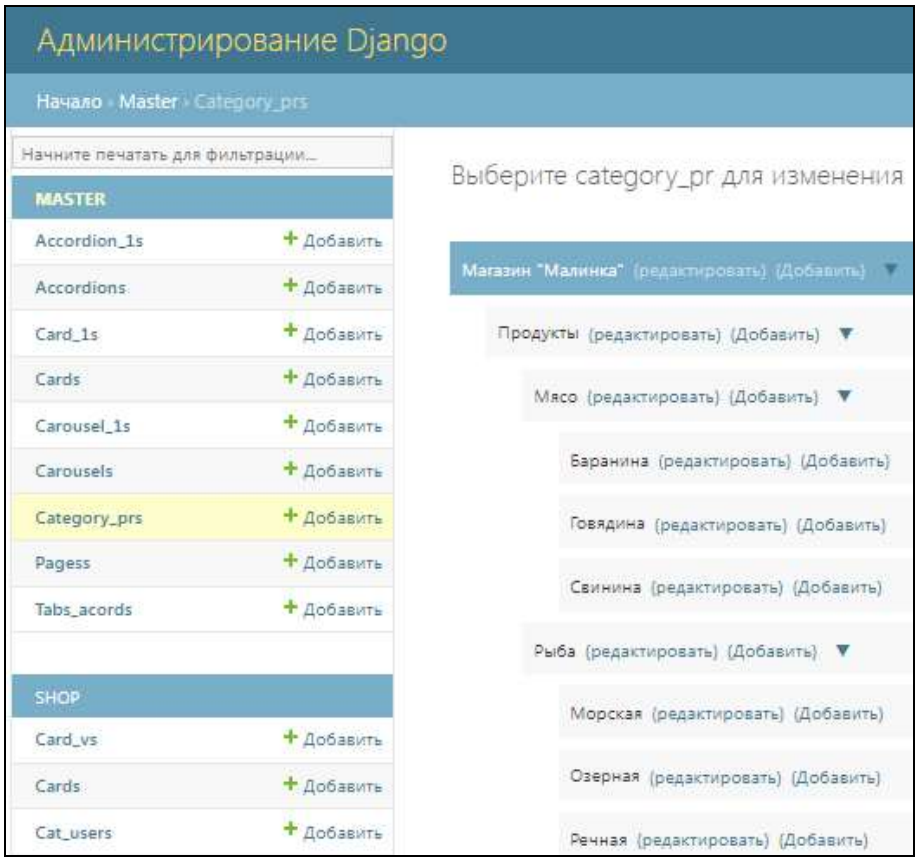


Рис. 4.3. Заполненные категории продуктов (модель `Category`) в административной панели Django

4.2.1. Отображение статичного дерева категорий продуктов в шаблонах Django

Создадим набор модулей, которые обеспечат вывод древовидных структур в шаблоны Django. Для начала нужно в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут (листинг 4.3).

Листинг 4.3. Ссылка активации маршрута для вывода данных модели Categiry_pr

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `category_1` (листинг 4.4).

Листинг 4.4. URL-маршрут category_1

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `category_1` (листинг 4.5).

Листинг 4.5. Представление (view) category_1

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении создан объект `category`, в который загружаются данные из модели `Category_pr`. Затем через словарь `content` сведения о категориях продуктов передаются в шаблон category_1.html, код которого приведен в листинге 4.6.

Листинг 4.6. Шаблон category_1.html

Примечание.
В демо-версии книги удалены листинги программ.

Это достаточно упрощенный шаблон страницы сайта, в котором имеются две панели: левая боковая и центральная. В левой боковой панели отображаются категории продуктов, а в центральной — должны отображаться сведения о продуктах той или иной категории (пока для упрощения примера эта панель пустая). В блоке вывода дерева категорий продуктов имеется ссылка: `<a href = "#">{{node.name}}</a>`. Пока эта ссылка тоже пустая — в ней нужно будет выполнить адресацию к тем модулям, которые обеспечат фильтрацию сведений о продуктах той или иной категории. Как это делается, будет подробно показано в следующих разделах, а пока посмотрим, как выглядит дерево категорий продуктов, для чего запустим на выполнение модули этого примера.

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Страница category_1.html**.

Внешний вид шаблона category_1.html представлен на рис. 4.4.

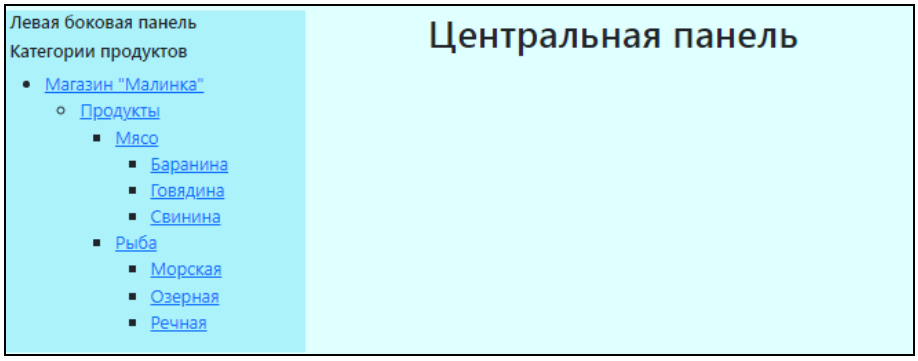


Рис. 4.4. Внешний вид категории продуктов в шаблоне category_1.html

Как здесь можно видеть, дерево категорий продуктов отображается в раскрытом виде, а ветки каждого нового уровня имеют некоторое смещение вправо. Каждый элемент в дереве категорий продуктов является ссылкой, и при переходе по ней будет происходить фильтра-

ция таблицы с информацией о продуктах, отображение которой предусмотрено в центральной панели.

По умолчанию дерево категорий продуктов всегда полностью раскрыто, что не совсем удобно при большом количестве записей в модели `Categiry_pr` и большом количестве уровней вложенности. Посмотрим теперь, как можно реализовать раскрывающееся дерево категорий продуктов.

4.2.2. Отображение динамичного дерева категорий продуктов в шаблонах Django

Создадим набор модулей, которые обеспечат вывод древовидных структур в шаблоны Django в виде раскрывающегося дерева. Для начала нужно в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут (листинг 4.7).

Листинг 4.7. Ссылка активации маршрута для вывода данных модели Category_pr

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `category_2` (листинг 4.8).

Листинг 4.8. URL-маршрут category_2

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление `category_2` (листинг 4.9).

Листинг 4.9. Представление (view) category_2

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создан объект `category`, в который загружаются данные из модели `Category_pr`. Затем через словарь `content` сведения о категориях продуктов передаются в шаблон category_2.html (листинг 4.10).

Листинг 4.10. Шаблон category_2.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно упрощенный шаблон страницы сайта, в котором имеются две панели: левая боковая и центральная. В левой боковой панели отображаются категории продуктов в виде свернутого дерева, а в центральной — должны отображаться сведения о продуктах той или иной категории (пока для упрощения примера эта панель пустая). В блоке вывода дерева категорий продуктов имеется ссылка: `<a href = "#">{{node.name}}</a>`. Пока эта ссылка тоже пустая — в ней нужно будет выполнить адресацию к тем модулям, которые обеспечат фильтрацию сведений о продуктах той или иной категории. Как это делается, будет подробно показано в следующих разделах.

Принципиальное отличие шаблона category_2.html от шаблона category_1.html заключается в том, что в шаблон category_2.html добавлены стили и Java-скрипт, которые обеспечивают раскрытие и сворачивание веток дерева категорий продуктов.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Страница category_2.html**.

Внешний вид шаблона category_2.html при первоначальной загрузке представлен на рис. 4.5.

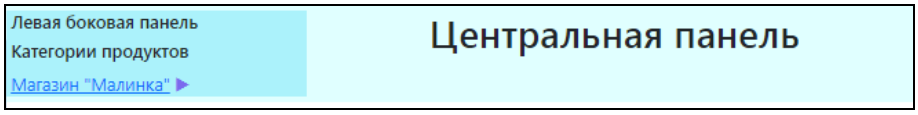


Рис. 4.5. Внешний вид категории продуктов в шаблоне category_2.html

Как здесь можно видеть, дерево категорий продуктов свернуто, и на экране присутствует только корень дерева — **Магазин «Малинка»**. С правой стороны от имени ветки дерева расположена кнопка ▶ — когда пользователь щелкнет на ней левой кнопкой мыши, эта ветка дерева (если у нее есть дочерние элементы) раскроется, а кнопка приобретет вид ▼ (рис 4.6).

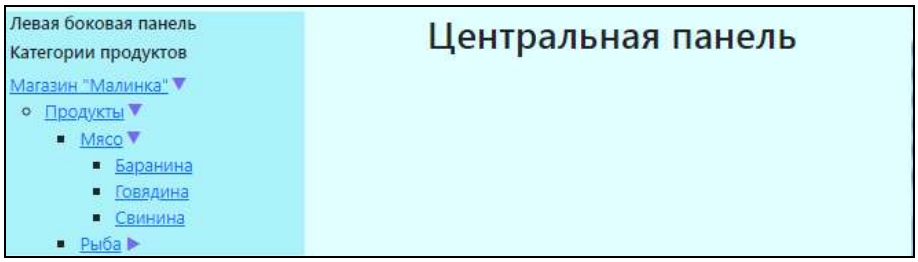


Рис. 4.6. Внешний вид категории продуктов в шаблоне category_2.html с частично раскрытым деревом

При повторном нажатии на эту кнопку дочерние ветки дерева свернутся. Таким способом пользователь может раскрывать только нужные ему ветки дерева, что очень удобно в тех случаях, когда дерево имеет много веток с большим количеством уровней их вложенности. Внешний вид дерева категорий продуктов при свернутой ветке **Мясо** и раскрытой ветке **Рыба** приведен на рис. 4.7.

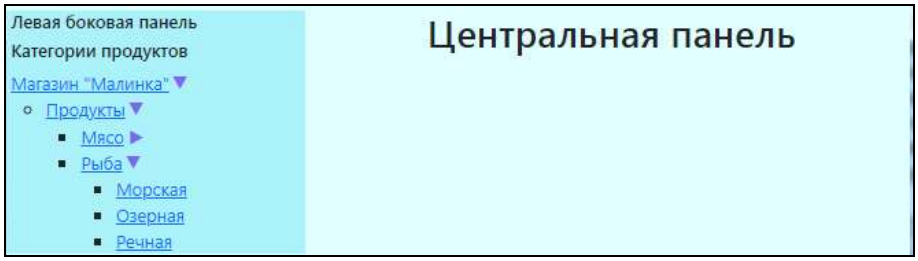


Рис. 4.7. Внешний вид категории продуктов в шаблоне category_2.html с раскрытой веткой **Рыба**

4.3. Формирования дерева категорий продуктов с использованием интерфейса пользователя

При создании сайтов для магазинов интернет-торговли достаточно часто требуется, чтобы функция формирования дерева категорий товаров была возложена не на разработчиков (администраторов) сайта, а на менеджеров магазинов. В этом случае логичнее не подпускать менеджеров магазинов к административной панели Django, а реализовать для них возможность формировать и менять структуру каталога категорий продуктов через пользовательский интерфейс. В этом разделе мы рассмотрим пример реализации такого интерфейса.

4.3.1. Модель для категорий продуктов

Для начала создадим новую модель для хранения данных о категориях продуктов, не содержащую данных. Программный код такой модели (`Category`) приведен в листинге 4.11.

Листинг 4.11. Модель категорий продуктов Category

Примечание.

В демо-версии книги удалены листинги программ.

В этом программном модуле на основе базового класса `MPTTModel` создан класс `Category` (категории продуктов). Модель `Category` содержит два поля: `name` — имя категории и `parent` — имя родительской категории. В административной панели Django для этой модели будет отображаться только одно поле — имя категории (`self.name`), а в интерфейсе пользователя дерево категорий будет отсортировано по имени категории (`order_insertion_by = ['name']`).

Создав код модели `Category`, не забудьте выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

Поскольку формирование данных дерева категорий продуктов будет выполняться через интерфейс пользователя, то эту модель можно не регистрировать в модуле admin.py (хотя для страховки действий менеджеров магазинов со стороны администраторов сайта это надо бы сделать).

Чтобы пользователи могли заполнять данными дерево категорий продуктов, необходимо создать соответствующую форму (листинг 4.12).

Листинг 4.12. Форма ввода категорий продуктов Category

Примечание.

В демо-версии книги удалены листинги программ.

Форма `Category_Form` построена на основе базового класса `forms.ModelForm`. В ней будет отображаться только одно поле — `name`. Объект `fields` принимает данные в виде списка. Чтобы поле `name` имело тип `<список>` (а не тип `<символьная_переменная>`), после имени поля нужно *обязательно поставить запятую*. В этой форме пользователю для ввода данных будет предоставлено всего одно поле — `name` (имя категории).

4.3.2. Модули для отображения дерева категорий продуктов

Для доступа к интерфейсу наполнения данными дерева категорий (редактору категорий) нужно создать следующую ссылку (листинг 4.13).

Листинг 4.13. Ссылка на редактор категорий продуктов

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будут активированы URL-маршруты с именем `show_category` (листинг 4.14).

Листинг 4.14. URL-маршруты для вызова редактора категорий продуктов

Примечание.

В демо-версии книги удалены листинги программ.

Следует обязательно создать два маршрута с именем `show_category` — один маршрут без передачи параметров в представление, а второй — с передачей в представление идентификатора категорий продукта (`id`).

По этим маршрутам будет вызываться представление `show_category`, которое может принимать идентификатор категории продукта (`id`). Код представления `show_category` приведен в листинге 4.15.

Листинг 4.15. Представление show_category

Примечание.

В демо-версии книги удалены листинги программ.

На первом шаге в объект `category_pr` загружаются все записи о категориях продуктов из БД и определяется количество таких записей — количество категорий (`kol_cat`). Если `kol_cat = 0`, то корень дерева не создан. В этом случае для переменной сессии с идентификатором категорий задается значение 1: `request.session['id_cat'] = 1` (идентификатор корневого элемента в дереве каталога). Поскольку корень дерева не создан, то у корня нет потомков, поэтому объекту `potomok` присваивается значение 0 (`potomok=0`). После этого формируется объект `context`, который передается в шаблон отображения дерева с категориями продуктов show_category.html (листинг 4.16).

Листинг 4.16. Шаблон show_category.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне имеются три блока:

- блок для отображения структуры дерева категорий продуктов (левая панель);
- блок для показа кнопки создания корневого элемента в дереве категорий продуктов (центральная панель);
- блок для показа таблицы, в которой можно наполнять данными дерево категорий продуктов (центральная панель).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Редактор категорий**.

Когда дерево категорий продуктов еще не создано (в таблице БД нет данных), то шаблон show_category.html будет выглядеть так, как показано на рис. 4.8.



Рис. 4.8. Внешний вид шаблона show_category.html при отсутствии в БД корневого узла дерева категорий продуктов

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы..

4.3.3. Модули для добавления корня к дереву категорий продуктов

Как можно видеть на рис. 4.8, левая панель с категориями продуктов пустая, а на центральной панели имеется только одна кнопка: **Добавить корень**. С этой кнопкой связана следующая ссылка (листинг 4.17).

Листинг 4.17. Ссылка для активации URL-маршрута category_root_add

Примечание.
В демо-версии книги удалены листинги программ.

По нажатию на эту кнопку будет выполнена адресация к URL-маршруту `category_root_add`, код которого приведен в листинге 4.18.

Листинг 4.18. URL-маршрут category_root_add

Примечание.
В демо-версии книги удалены листинги программ.

Этот URL-маршрут активирует представление `category_root_add` (листинг 4.19).

Листинг 4.19. Представление category_root_add

Примечание.
В демо-версии книги удалены листинги программ.

При обращении к этому представлению по умолчанию поступает GET-запрос. При этом на основе класса `Category_Form()` создается объект `form` с пустыми данными, и этот объект передается в шаблон category_root_form, код которого приведен в листинге 4.20.

Листинг 4.20. Шаблон category_root_form.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне (рис. 4.9) для пользователя отображается только одно поле, в которое он может ввести имя корневого узла. Здесь также имеются две кнопки: **Отмена** — для отмены записи в БД введенной информации и **Сохранить** — для записи этой информации в БД.

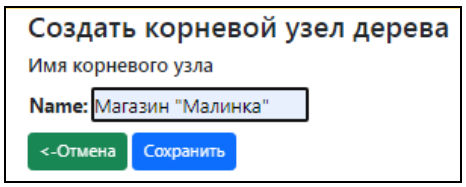


Рис. 4.9. Внешний вид шаблона category_root_form

По нажатию на кнопку **Сохранить** будет выполнен возврат в представление `category_root_add`. Поскольку в этом случае поступит POST-запрос, то будут выполнены инструкции в теле блока `if request.method == "POST"`. Здесь объект `form` получит значение имени корневой категории, и будет выполнена проверка валидности формы. Если в форме нет ошибок, то в БД будет создана запись с данными корневого элемента в дереве категорий продуктов с использованием следующей инструкции:

```
root_cat = Category.objects.create(name=form.cleaned_data.get('name'))
```

а информация о созданной корневой категории — сохранена в объекте `root_cat`. Затем выполнится возврат к представлению `show_category`, в который будет передан идентификатор корневого элемента в дереве категорий продуктов (`root_cat.id`). После этого шаблон show_category.html примет вид, показанный на рис. 4.10.

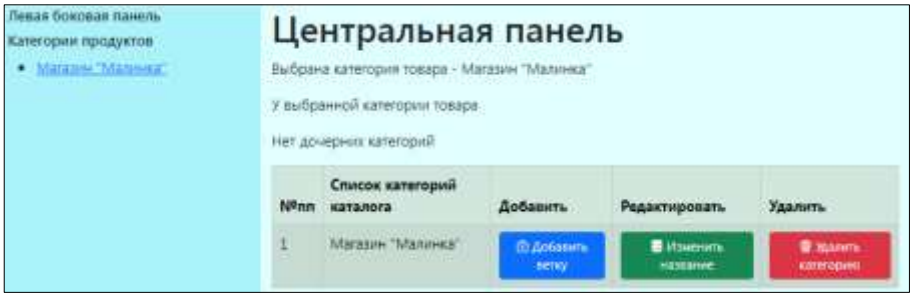


Рис. 4.10. Внешний вид шаблона show_category.html после создания корневого элемента категорий продуктов

В левой панели шаблона появится корневой элемент дерева категорий продуктов, а в центральной панели — таблица, в которую из БД будет загружено имя корневой категории и откроются три кнопки:

- **Добавить ветку** — для добавления дочерней ветки к текущей категории продуктов;
- **Изменить название** — сменить название текущей ветки дерева категорий продуктов:
- **Удалить категорию** — удалить текущую категорию (при этом будут удалены и все дочерние категории продуктов).

Обратите внимание, что кнопка **Добавить корень** исчезла, поскольку в нашем алгоритме предусмотрено создание только одного корневого элемента. На самом деле корневых элементов может быть несколько, и как их создать будет показано в одном из следующих разделов.

В результате создания корневого элемента категорий продуктов в БД появится запись с параметрами этой категории (рис. 4.11).

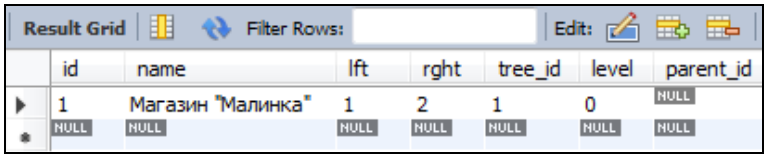


Рис. 4.11. Запись в БД с информацией о корневом элементе категорий продуктов

Как здесь можно видеть, корневой узел имеет идентификатор **id** = **1**, находится на нулевом уровне (**level** = **0**) и не имеет родительской категории (**parent_id** = **NULL**). Левая (**lft** = **1**) и правая (**rght** = **2**) части ветки корневого элемента получили значения 1 и 2.

Теперь пользователь с использованием кнопки шаблона **Добавить ветку** может по своему усмотрению формировать дерево категорий произвольным образом. Рассмотрим, как при этом выполняются процедуры в представлении `show_category`.

Это представление в параметре `id` получает идентификатор категории продукта. После ввода корневого элемента дерева категорий параметр `id` будет иметь значение `id=1`. Объект `category_pr` получит полный список элементов дерева категорий, а в объект `kol_cat` будет внесено количества веток в дереве категорий. Так как количество категорий не будет равно нулю, то в работу вступают процедуры блока `if id=0`. Здесь проверяется условие — откуда было обращение к этому представлению. Если обращение произошло из блока с каталогом продуктов (левая панель шаблона show_category.html), то в переменной сессии `id_cat` запоминается идентификатор категории, которую выбрал пользователь. Если обращение произошло из модулей ввода и корректировки данных, то в объект `id_cat` загружается значение идентификатора категорий, которое было сохранено в сессии.

Затем создаются два объекта:

- `select_cat` — категория, выбранная в каталоге;
- `id_rod_cat` — родительская категория для категории, выбранной в каталоге.

Если в каталоге выбрана корневая категория (а у нее нет родителя), то для нее в качестве идентификатора родителя указываем собственный идентификатор (`id`).

Далее работают процедуры, в которых в объект `categories` загружаем все записи с выбранной категорией продуктов, включая все дочерние категории. Для этого используется соответствующий метод класса `MpttModel`:

```
categories = select_cat.get_descendants(include_self=True)
```

На следующем шаге определяется количество потомков (дочерних элементов) у выбранной категории (объект `potomok`). После этого формируется объект `context`, который передается в шаблон show_category.html.

4.3.4. Модули для добавления дочерних категорий продуктов

Рассмотрим, как работают программные модули при нажатии на кнопку **Добавить ветку**. С этой кнопкой связана следующая ссылка в шаблоне show_category.html:

```
<a href="{% url 'master:category_child_add' kn=object.name %}"
   class="btn btn-primary btn-sm">
    <i class="bi bi-bag-plus"></i> Добавить ветку</a>
```

По этой ссылке вызывается URL-маршрут с именем `category_child_add`, и этому маршруту передается имя категории, к которой нужно добавить ветку (листинг 4.21).

Листинг 4.21. URL-маршрут category_child_add

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `category_child_add`, в которое в качестве аргумента передается имя родительской категории (`kn`) в виде текстовой строки (листинг 4.22).

Листинг 4.22. Представление category_child_add

Примечание.

В демо-версии книги удалены листинги программ.

Это представление имеет параметр `kn`, который принимает имя родительской категории. При поступлении GET-запроса создается форма для ввода данных (`form`) на основе класса `Category_Form()`. Затем формируется объект `context`, который передается в шаблон category_child_form.html (листинг 4.23).

Листинг 4.23. Шаблон category_child_form.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне пользователь заполняет только одно поле — имя дочерней категории. По нажатию на кнопку **Сохранить** происходит возврат в представление `category_child_add` уже в виде POST-запроса. Здесь в объект `form` загружаются введенные пользователем данные, и выполняется проверка их валидности. Если в форме нет ошибок, то формируются два объекта:

- `child_name` — имя дочерней категории (выгружается из формы);
- `parent` — имя родительской категории (берется из параметра `kn` представления).

После этого в БД создается новая запись об объекте с использованием метода класса `MpttModel`:

```
Category_pr.objects.create(name=child_name, parent=parent)
```

После этого выполняется возврат к представлению `show_category`, где происходит обновление шаблона show_category.html.

Итак, у нас есть корневой элемент категорий товаров с именем **Магазин «Малинка»**. Начнем формирование дерева категорий продуктов с использованием кнопки **Добавить ветку**. Для начала создадим дочернюю категорию **Продукты**. Для этого нужно нажать на кнопку **Добавить ветку** и в открывшейся форме ввести имя дочерней ветки — в нашем случае: **Продукты** (рис. 4.12).

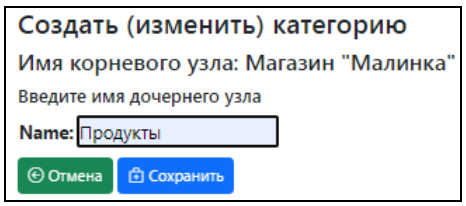


Рис. 4.12. Добавление дочернего элемента к корневому узлу категорий продуктов

После нажатия на кнопку **Сохранить** шаблон show_category.html обновится и приобретет следующий вид (рис. 4.13).

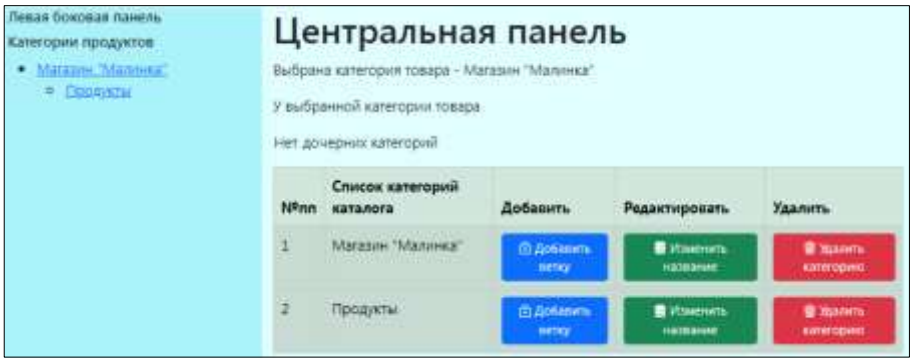


Рис. 4.13. Родительский и дочерний элементы категорий продуктов в шаблоне show_category.html

Здесь в левой боковой панели будет показано обновленное дерево категорий продуктов, а в центральной панели появится обновленная таблица с добавленным элементом. В базе данных будет создана новая запись, при этом значения полей **lft** и **rght** обновятся автоматически (рис. 4.14).

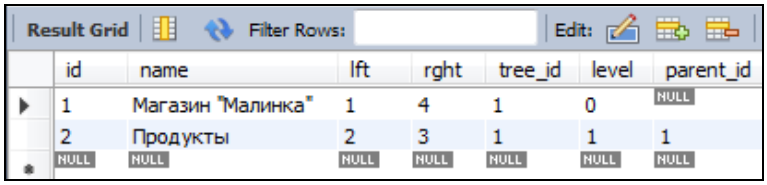


Рис. 4.14. Родительский и дочерний элементы категорий продуктов в базе данных

Продолжим заполнение данными дерева категорий продуктов, создав дочерние категории, приведенные в табл. 4.1. При этом дочерняя категория будет добавлена к тому элементу, в строке которого в шаблоне show_category.html была нажата кнопка **Добавить ветку**. В итоге дерево категорий продуктов примет вид, показанный на рис. 4.15, а в БД будут сформированы следующие данные (рис. 4.16).

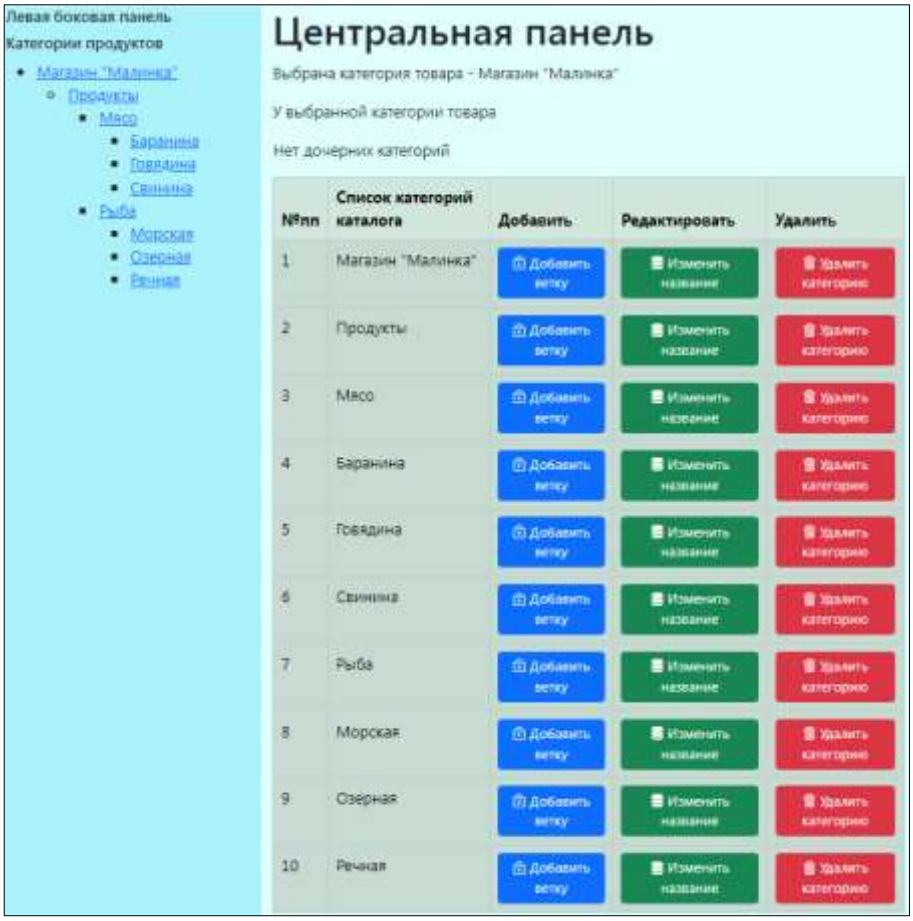
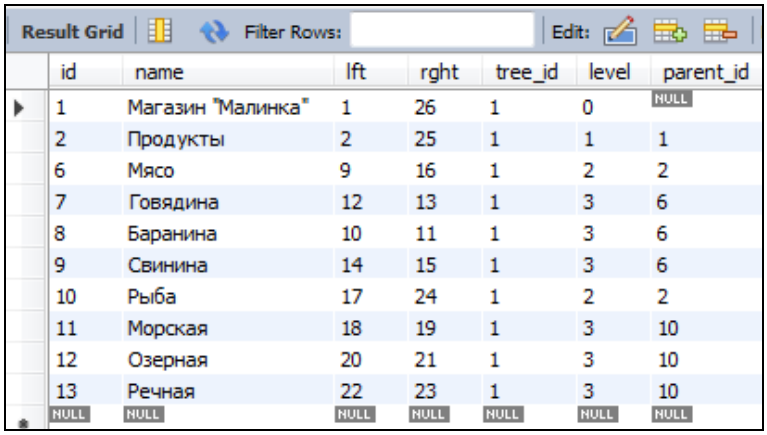


Рис. 4.15. Заполненное дерево категорий продуктов в шаблоне show_category.html

Интерактивная цифровая книга

Именно в таком виде будет показана таблица с категориями продуктов в интерактивной цифровой книге, если вы на вкладке **Показ примеров** нажмете на кнопку **Примеры главы 4** и перйдете по ссылке **Редактор категорий**.



id	name	lft	rght	tree_id	level	parent_id
1	Магазин "Малинка"	1	26	1	0	NULL
2	Продукты	2	25	1	1	1
6	Мясо	9	16	1	2	2
7	Говядина	12	13	1	3	6
8	Баранина	10	11	1	3	6
9	Свинина	14	15	1	3	6
10	Рыба	17	24	1	2	2
11	Морская	18	19	1	3	10
12	Озерная	20	21	1	3	10
13	Речная	22	23	1	3	10
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рис. 4.16. Заполненное дерево категорий продуктов в базе данных

Обратите внимание, что в процессе ввода данных пользователь в формах вводил только наименования категорий продуктов, при этом Django автоматически сконфигурировал значения остальных ключевых полей.

Если дерево категорий достаточно большое и имеет сложную структуру, то пользователь может в левой части страницы выбрать в дереве одну из веток, после чего таблица в центральной части страницы перестроится, и в ней будет отображена только выбранная категория и все ее дочерние категории. Например, при выборе в дереве категорий элемента **Мясо**

в таблице останутся только дочерние элементы этой категории: **Баранина**, **Говядина** и **Свинина** (рис. 4.17).

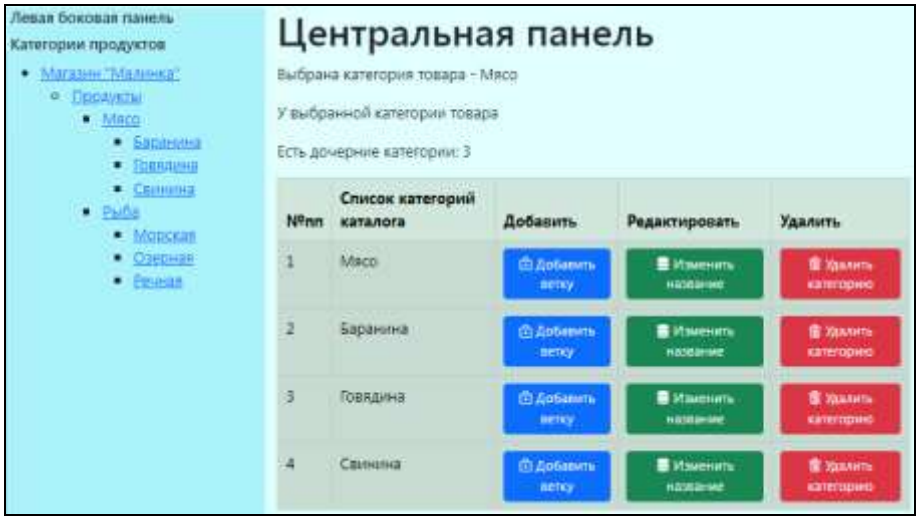


Рис. 4.17. Заполненное дерево категорий продуктов в базе данных

В этом режиме пользователь может продолжать редактировать дерево категорий продуктов. Рассмотрим, как работает такая фильтрация данных. Каждая ветка дерева категорий в левой части окна показана как ссылка. Эта ссылка в шаблоне show_category.html записана следующим образом:

```
<a href="/show_category/{{node.id}}">{{node.name}}</a>
```

По этой ссылке активируется маршрут `show_category`, в который передается идентификатор (`id`) выбранной категории:

```
path('show_category/<int:id>/', views.show_category, name='show_category'),
```

По этому маршруту вызывается представление `show_category` с аргументом выбранной категории (`id`).

В представлении этот идентификатор выбранной категории (`id`) принимается в качестве параметра, и на его основе данные о категориях продуктов фильтруются в следующих инструкциях:

```
# выбранная категория в каталоге
select_cat = Category.objects.get(id=id_cat)
 . . .
# категории, в которые входит выбранная категория и все дочерние категории
categories = select_cat.get_descendants(include_self=True)
```

Объект `categories` передается в шаблон show_category.html, где используется для отображения таблицы с выбранными категориями товаров. Для этого строки таблицы отображаются в центральной панели в следующем цикле:

```
{% for object in categories %}
        . . .
{% endif %}
```

Содержимой же левой панели шаблона, где показано само дерево категорий, не фильтруется и показывается в развернутом виде. Для этого используется следующий блок инструкций:

```
<h6>Категории продуктов</h6>
{% load mptt_tags %}
<ul id="myUL">
    {% recursetree category_pr %}
    <li>
        <a href="{% url 'master:show_category' id=node.id %}">{{ node.name
}}</a>
            {% if node.is_leaf_node %}
            <span class="leaf">{{ node.code }}</span>
            {% else %}
            <span class="caret parent">{{ node.code }}</span>
```

```
            <ul class="nested">{{ children }}</ul>
            {% endif %}
    </li>
    {% endrecursetree %}
</ul>
```

Здесь для показа дерева категорий задействован следующий тег:

```
{% recursetree category_pr %}
       . . .
{% endrecursetree %}
```

4.3.5. Модули для обновления категорий продуктов

Рассмотрим, как работают программные модули при нажатии в шаблоне show_category.html на кнопку **Изменить название**. С этой кнопкой связана следующая ссылка:

```
<a href="{% url 'master:category_child_upd' id=object.id %}"
   class="btn btn-success btn-sm">
    <i class="bi bi-server"></i> Изменить название</a>
```

По этой ссылке вызывается URL-маршрут с именем `category_child_upd`, и ему передается идентификатор категории, у которой нужно сменить название (листинг 4.24).

Листинг 4.24. Маршрут category_child_upd

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `category_child_upd`, в которое передается идентификатор текущей категории (`id`) в виде целого числа (листинг 4.25).

Листинг 4.25. Представление category_child_upd

Примечание.

В демо-версии книги удалены листинги программ.

Это представление имеет параметр `id`, который принимает идентификатор категории продукта. При поступлении GET-запроса на основе класса `Category_Form()` создается форма (`form`) для корректировки данных. Затем формируется объект `context`, который передается в шаблон category_child_form.html.

В шаблоне category_child_form.html пользователь корректирует только одно поле — имя категории. По нажатию на кнопку **Сохранить** происходит возврат в представление `category_child_upd` уже в виде POST-запроса. Здесь в объект `form` загружаются введенные пользователем данные, и выполняется проверка их валидности. Если в форме нет ошибок, то обновленное имя категории записывается в БД. После этого выполняется возврат к представлению `show_category`, где происходит обновление шаблона show_category.html.

4.3.6. Модули для удаления категорий продуктов

Рассмотрим, как работают программные модули при нажатии в шаблоне show_category.html на кнопку **Удалить категорию**. С этой кнопкой связана следующая ссылка:

```
<a href="{% url 'master:category_del' id=object.id %}"
   class="btn  btn-danger btn-sm">
   <i class="bi-trash3"></i> Удалить категорию</a>
```

По этой ссылке вызывается URL-маршрут с именем `category_del` и ему передается идентификатор удаляемой категории (листинг 4.26).

Листинг 4.26. Маршрут category_del

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `category_del`, в которое передается идентификатор удаляемой категории (`id`) в виде целого числа (листинг 4.27).

Листинг 4.27. Представление category_del

Примечание.

В демо-версии книги удалены листинги программ.

Это представление имеет параметр `id`, который принимает идентификатор удаляемой категории продукта. Из этого представления выполняется обращение к шаблону, в котором будет сформирована форма. В этой форме пользователь получит имя удаляемой категории и сможет либо подтвердить, либо отменить удаление.

При поступлении GET-запроса в представлении создается объект `category`, текстовое сообщение и имя категории, которые загружаются в объект `context`, и этот объект передается в шаблон del_yesno.html (рис. 4.18). В этом шаблоне в теле формы пользователю отображается имя удаляемой категории и две кнопки: **Отмена** — отменить удаление и **Удалить** — подтвердить удаление. Эти кнопки именованные: кнопка **Отмена** имеет имя `name='no'`, кнопка **Удалить** — имя `name='yes'`. Имя нажатой кнопки будет передано в представление `category_del` в POST-запросе.

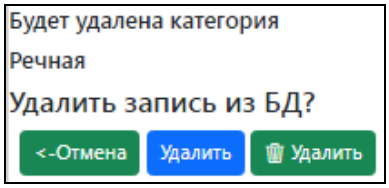


Рис. 4.18. Внешний вид шаблона del_yesno.html

Напомним, что здесь для примера показаны два варианта кнопки **Удалить**: с синим фоном — простая кнопка для удаления, с зеленым фоном — кнопка для удаления с крутящимся спином.

По нажатию на любую кнопку выполняется возврат в представление `category_del`, где в инструкции `if 'yes' in request.POST` производится идентификация нажатой кнопки. Если пользователь нажал кнопку **Удалить**, будет выполнена инструкции удаления записи из БД, а если кнопку **Отменить**, эта инструкция будет пропущена, и в обоих случаях произойдет возврат к шаблону отображения категорий продуктов show_category.html. Поскольку текущая категория может быть удалена, то в шаблон передается идентификатор корневой категории, при этом удаленная категория удаляется и из дерева категорий, и из таблицы со списком категорий.

* * *

Реализация древовидных структур Django-mptt предлагает эффективный метод обработки иерархических данных. Разработчики могут быстро создавать и запрашивать иерархические структуры и манипулировать ими, используя библиотеку Django-mptt. Недостатки традиционных методов выборки иерархической информации из БД преодолеваются за счет использования в модели MPTT левых и правых значений для структурных элементов (веток) дерева, что позволяет быстро находить предков, потомков, братьев и сестер. Таким образом, с помощью Django-mptt разработчики могут добиться повышения производительности, упрощения управлением взаимоотношениями и гибкости при обработке информации. Это достаточно мощный инструмент для эффективного управления иерархическими данными, будь то платформы электронной коммерции, системы управления контентом или любые приложения, работающие с иерархическими данными.

4.4. Создание списков объектов с фильтрацией через древовидные каталоги

В предыдущих разделах мы создавали древовидные структуры на примере категорий продуктов. Создание древовидных структур приобретает смысл лишь в том случае, если они

используются для фильтрации и выборки информации из больших массивов данных, — например, когда мы имеем дело с интернет-магазином, в номенклатуре которого не одна тысяча различных товаров. Тогда всю номенклатуру товаров можно разбить на категории и предоставить пользователям удобный интерфейс для поиска нужного товара через древовидный справочник категорий товаров. Для реализации такой возможности надо обеспечить связь древовидного каталога категорий товаров с таблицами БД, в которых хранятся данные о самих товарах. Или, например, когда сайт поддерживает блог со множеством статей на различные темы. В этом случае нужно создать справочник категории статей, а затем массив статей блога связать с древовидным справочником этих категорий, — таких примеров можно привести бесконечное множество. Мы далее рассмотрим набор программных модулей, которые позволяют реализовать создание списка продуктов интернет-магазина и связанного с ним справочника категорий продуктов. Это наиболее распространенная задача, которая востребована при создании сайтов интернет-торговли. Для ее решения мы создадим две новые модели: для хранения категорий продуктов (`Genre`) и для хранения данных о самих продуктах (`Products`).

4.4.1. Модель для хранения категорий продуктов

В предыдущих разделах была создана модель категорий продуктов `Category`, которая имела один корневой элемент. Здесь же мы для примера создадим другую модель категорий продуктов — `Genre_m`, которая получит несколько корневых элементов. Код модуля для формирования модели `Genre_m` приведен в листинге 4.28.

Листинг 4.28. Модель для хранения категорий продуктов Genre_m

Примечание.

В демо-версии книги удалены листинги программ.

Для заполнения этой модели данными через административную панель Django зарегистрируем ее в модуле admin.py (листинг 4.29).

Листинг 4.29. Регистрация модели Genre_m в модуле admin.py

Примечание.

В демо-версии книги удалены листинги программ.

Заполним эту модель данными через административную панель Django, создав несколько корневых категорий (рис. 4.19).

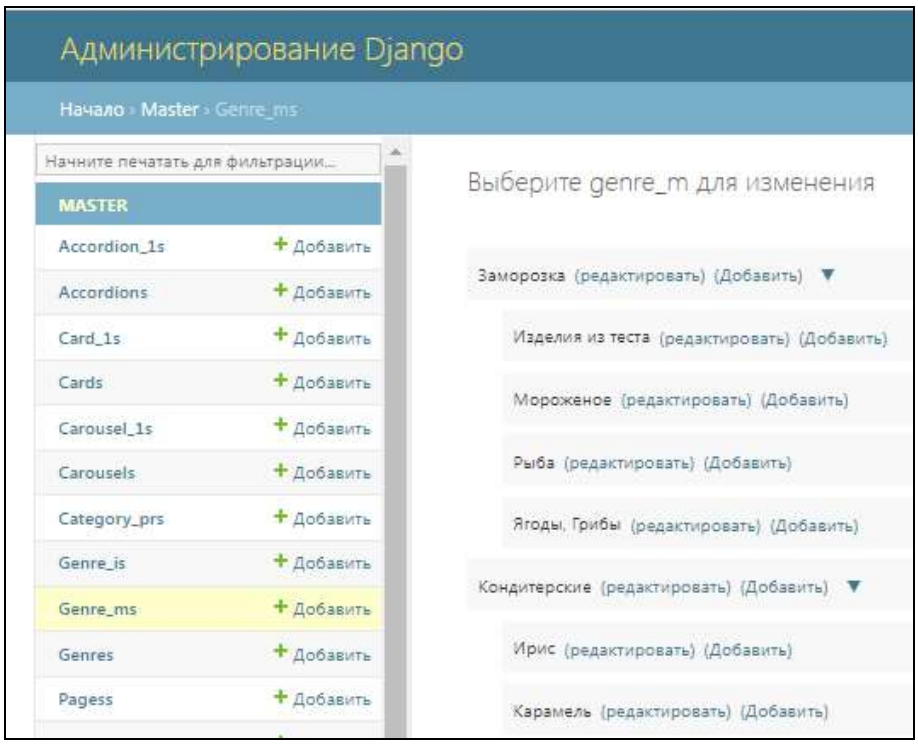


Рис. 4.19. Заполненный справочник категорий продуктов `Genre_m`

4.4.2. Модель для хранения сведений о продуктах

Теперь, когда мы реализовали модель для хранения категорий продуктов `Genre_m`, необходимо создать модель для хранения данных о продуктах `Product`, а также функцию, которая будет автоматически формировать имена файлов для изображений (фотографий) продуктов (листинг 4.30).

Листинг 4.30. Код функции rename_prod и модель для хранения данных о продуктах Product

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В этой модели для класса `Product` созданы следующие поля:

- `genre` — категория продукта;
- `name` — наименование продукта;
- `description` — описание (характеристики) продукта;
- `price` — цена продукта;
- `image` — изображение (фотография) продукта;
- `ostat` — остаток на складе.

Поле `genre` предназначено для связи с моделью `Geng_m` (категория продукта) по первичному ключу. Здесь используется тип связи «один ко многим», где «один» — категория продукта, а «многие» — сведения о конкретных продуктах. По умолчанию при выводе сведений о продуктах они будут отсортированы по наименованию продукта (поле `name`).

Чтобы обеспечить автоматическое формирование имен загруженных изображений продуктов, эта модель задействует функцию `rename_prod`. Поскольку публичные серверы, работающие на Linux, не принимают файлы с именами на кириллице, использование этой функции позволит автоматически переименовывать такие файлы.

Для реализации возможности наполнения данными сведений о продуктах в административной панели, эту модель нужно зарегистрировать в модуле admin.py (листинг 4.31).

Листинг 4.31. Регистрация модели Product в файле admin.py

Примечание.
В демо-версии книги удалены листинги программ.

Создав коды моделей `Genre` и `Product`, не забудьте выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

4.4.3. Форма для ввода сведений о продуктах

Чтобы пользователь мог заполнять данными сведения о продуктах, необходимо создать соответствующую форму (листинг 4.32).

Листинг 4.32. Форма для ввода сведений о продуктах ProductForm

Примечание.
В демо-версии книги удалены листинги программ.

Форма `ProductForm` построена на основе базового класса `forms.ModelForm`, и в ней будут отображаться все поля для ввода данных, кроме поля `genre`.

4.4.4. Редактор продуктов

Для доступа к интерфейсу формирования данных о продуктах нужно в личном кабинете пользователя или в меню любой страницы сайта создать следующую ссылку для вызова редактора продуктов (листинг 4.33).

Листинг 4.33. Ссылка для вызова редактора продуктов

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будут активированы URL-маршруты с именем `edit_product` (листинг 4.34).

Листинг 4.34. URL-маршруты edit_product

Примечание.
В демо-версии книги удалены листинги программ.

По этим маршрутам будет вызываться представление `edit_product`, в которое может передаваться идентификатор категории (листинг 4.35).

Листинг 4.35. Представление edit_product

Примечание.
В демо-версии книги удалены листинги программ.

Это представление имеет параметр id, который принимает идентификатор категории продукта. Если такой идентификатор не передан, то по умолчанию он принимает значение id=0 (пользователь попал в представление из модулей добавления или обновления продукта). Важно не забыть импортировать модели Product, Genre_m и пагинатор Paginator.

Если пользователь попал в представление из каталога продуктов (`id` не равно 0), то в переменной сессии `id_cat` запоминается идентификатор выбранной пользователем категории. Если пользователь попал в представление из модулей добавления или обновления продукта

(`id = 0`), то из переменной сессии `id_cat` в объект `id_cat` загружается идентификатор категории товара, которую пользователь выбрал на предыдущем шаге.

После этого из БД в объект `genre` загружаются все категории товаров, в объект `select_cat` — та категория товара, которая выбрана пользователем, а в объект `id_rod_cat` — `id` родительской категории для выбранной категории товара.

Если в дереве каталога создан только главный корень, у которого нет родителя, то для него делается присвоение `id_rod_cat=1` (главный корень ссылается сам на себя).

На следующем шаге объект `select_rod_cat` получает запись с параметрами родительской категории, а объект `genres` — записи, в которые входит выбранная категория и все дочерние категории.

После этого объект `n_genres` получает количество выбранных категорий, включая дочерние. Это нужно для того чтобы блокировать кнопку **Добавить продукт**, поскольку продукт может быть добавлен только к конечной ветке дерева, у которой нет дочерних категорий, — то есть для этой ветки количество категорий `n_genres=1`.

На следующем шаге объект `products` (продукты), получает из БД записи обо всех продуктах, которые входят в выбранную категорию и во все дочерние.

На заключительном этапе создается пагинатор (объект `paginator`), который в нашем примере позволяет выводить по 10 единиц товара на одной HTML-странице.

После этого формируется объект `context`, который передается в шаблон отображения и редактирования продуктов edit_product.html (листинг 4.36).

Листинг 4.36. Шаблон edit_product.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне имеются несколько блоков:

- блок для отображения структуры дерева категорий продуктов (левая панель);
- блок для показа кнопки создания нового продукта (центральная панель)
- блок для показа таблицы, в которой можно отображать/редактировать данные о продуктах (центральная панель);
- пагинатор для постраничного вывода сведений о продуктах.

В колонках таблицы этого шаблона отображается следующая информация:

- номер по порядку (счетчик строк таблицы);
- наименование продукта;
- цена продукта;
- фото продукта (в виде миниатюры);
- категория продукта;
- кнопки для редактирования продукта;
- кнопка для показа крупного изображения продукта.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Редактор продуктов**.

Когда сведения о продуктах выбранной категории еще не сформированы (в таблице БД нет данных), то шаблон edit_product.html будет выглядеть так, как показано на рис. 4.20.

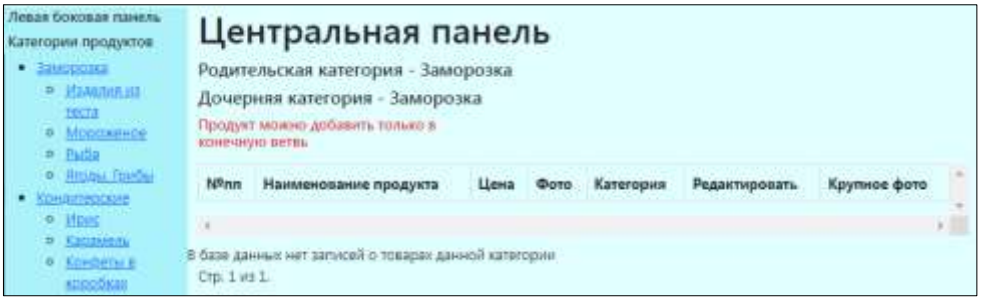


Рис. 4.20. Шаблон edit_product.html при выборе корневой категории продукта

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

Как здесь можно видеть, на центральной панели показано сообщение: **Продукт можно добавить только в конечную ветвь**. Этим алгоритмом предусмотрен запрет добавления продуктов к корневым и промежуточным категориям продуктов. Но если в левой боковой панели пользователь выберет конечную ветку категорий продуктов, то на центральной панели появится кнопка **Добавить товар** (рис. 4.21).

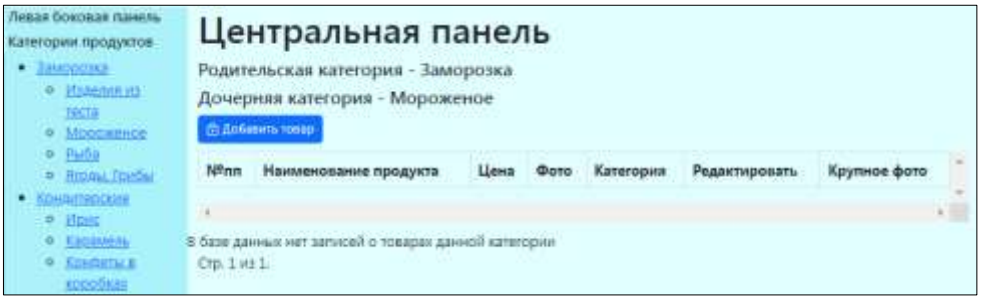


Рис. 4.21. Шаблон edit_product.html при выборе конечной ветки категории продукта

4.4.5. Модули для добавления в БД продуктов (товара)

Рассмотрим, как работают программные модули при нажатии на кнопку **Добавить товар** шаблона edit_product.html. С этой кнопкой в шаблоне связана следующая ссылка:

```
<a href="{% url 'master:product_add' %}"
   class="btn btn-primary btn-sm">
    <i class="bi bi-bag-plus"></i> Добавить товар</a>
```

По этой ссылке вызывается URL-маршрут с именем `product_add` (листинг 4.37).

Листинг 4.37. URL-маршрут product_add

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `product_add` (листинг 4.38).

Листинг 4.38. Представление product_add

Примечание.

В демо-версии книги удалены листинги программ.

При поступлении GET-запроса на основе класса `Product_Form()` создается форма для ввода данных (`form`). Затем формируется объект `context`, который передается в универсальный шаблон add_upd_form.html. Здесь важно не забыть выполнить импорт формы `Product_Form()`. В универсальном шаблоне add_upd_form.html эта форма имеет вид, показанный на рис. 4.22.

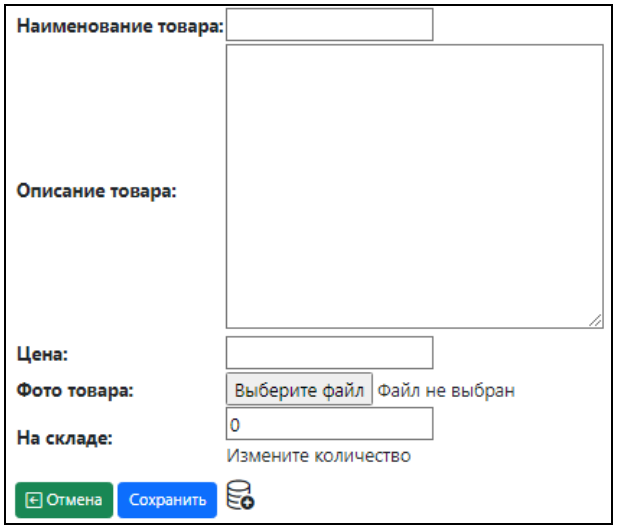


Рис. 4.22. Внешний вид формы `Product_Form()` в универсальном шаблоне add_upd_form.html

В этой форме пользователь заполняет нужные поля и нажимает на кнопку **Сохранить**, после чего происходит возврат в представление `product_add` уже в виде POST-запроса. Здесь в объект `form` загружаются введенные пользователем данные и выполняется проверка их валидность. Если в форме нет ошибок, то в БД создается новая запись об объекте с использованием метода `create` и выполняется возврат к представлению `edit_product`, где происходит обновление шаблона edit_product.html. После этого в таблице шаблона edit_product.html появится запись о добавленном товаре (рис. 4.23), а пользователь может продолжать ввод сведений о продуктах разных категорий.



Рис. 4.23. Внешний вид шаблона edit_product.html после ввода данных о продукте

Рассмотрим теперь, как в этом представлении работает фильтрация данных. Каждая ветка дерева категорий в левой панели страницы шаблона edit_product.html показана как ссылка:

```
<a href="{% url 'master:edit_product' id=node.id %}">{{ node.name }}</a>
```

По этой ссылке активируется маршрут `edit_product`, в который передается идентификатор (`id`) выбранной категории:

```
path('edit_product/<int:id>/', views.edit_product, name='edit_product'),
```

По этому маршруту вызывается представление `edit_product` с аргументом выбранной категории (`id`). В представлении в качестве параметра принимается идентификатор выбранной категории (`id`). Так как в этом случае `id` не равно нулю, то идентификатор категории принимает значение: `id_cat = id`, и на основе этого значения данные о категориях товара фильтруются в следующих инструкциях:

```
# Выбранная категория продукта в каталоге
select_cat = Genre.objects.get(id=id_cat)
```

```
  . . .
# категории, в которые входит выбранная категория и все дочерние категории
genres = select_cat.get_descendants(include_self=True)
```

Далее на основе объекта `genres` фильтруются данные о продуктах:

```
# продукты, которые входят в выбранную категорию, и во все дочерние
products = Product.objects.filter(genre__in=genres).distinct().order_by('name')
```

В таблице центральной панели фото продукта отображается в виде миниатюры. Для показа полноразмерной фотографии продукта в таблице имеется колонка **Крупное фото** с кнопкой **Показать**. С этой кнопкой связана следующая ссылка:

```
<a href="{{ object.image.url}}"
       class="btn btn-primary btn-sm" target="_blank">
       <i class="bi-search"></i> Показать</a>
```

По нажатию на кнопку **Показать** по этой ссылке в новой вкладке браузера будет выведено полноразмерное фото продукта (рис. 4.24).



Рис. 4.24. Полноразмерное фото продукта

4.4.6. Модули для обновления сведений о продуктах

Теперь рассмотрим, как работают программные модули при нажатии в шаблоне edit_product.html на кнопку **Изменить**. С этой кнопкой связана следующая ссылка:

```
<a href="{% url 'master:product_upd' id=object.id %}"
   class="btn btn-success btn-sm">
    <i class="bi bi-server"></i>Изменить</a>
```

По этой ссылке вызывается URL-маршрут с именем `product_upd` (листинг 4.39), в который передается идентификатор текущей записи (`id=object.id`).

Листинг 4.39. Маршрут URL product_upd

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `product_upd` (листинг 4.40).

Листинг 4.40. Представление product_upd

Примечание.

В демо-версии книги удалены листинги программ.

Это представление в качестве параметра принимает идентификатор (`id`) обновляемого продукта, и при поступлении GET-запроса на основе класса `Product_Form()` создается форма для ввода данных (`form`). Затем формируется объект `context`, который передается в универсальный шаблон add_upd_form.html, содержащий информацию о продукте из текущей записи в БД (рис. 4.25).

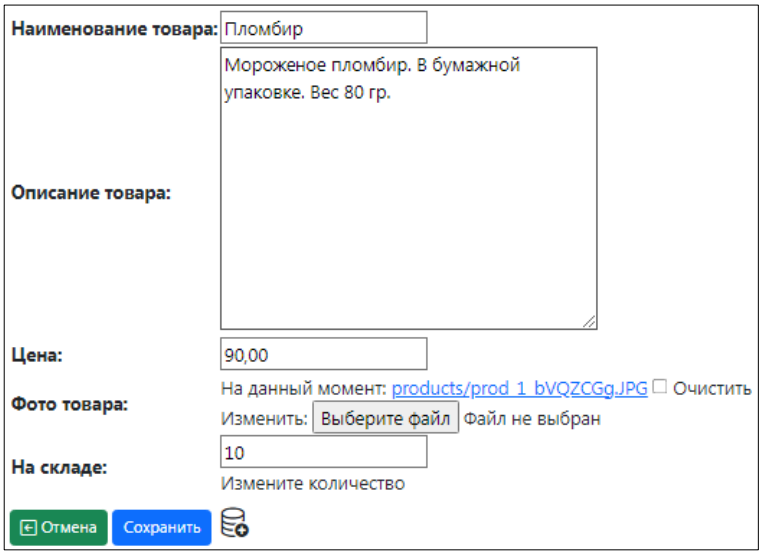


Рис. 4.25. Форма `Product_Form()` в универсальном шаблоне add_upd_form.html с заполненными данными

В этом шаблоне пользователь корректирует нужные поля и нажимает на кнопку **Сохранить**, после чего происходит возврат в представление `product_upd` уже в виде POST-запроса, где в объект `form` загружаются скорректированные пользователем данные и выполняется проверка их валидности. Если в форме нет ошибок, то в БД обновляется текущая запись об объекте с использованием метода `save`. Затем выполняется возврат к представлению `edit_product`, где происходит обновление шаблона edit_product.html. После этого в таблице шаблона edit_products.html появится обновленная запись о товаре.

4.4.7. Модули для удаления продукта из БД

Рассмотрим, как работают программные модули при нажатии в шаблоне edit_produt.html на кнопку **Удалить**. С этой кнопкой связана следующая ссылка:

```
<a href="{% url 'master:product_del' id=object.id %}"
   class="btn  btn-danger btn-sm">
    <i class="bi-trash3"></i> Удалить</a>
```

По этой ссылке вызывается URL-маршрут с именем `product_del` (листинг 4.41), и этому маршруту передается идентификатор удаляемого продукта.

Листинг 4.41. Маршрут URL product_del

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление product_del (листинг 4.42), в которое в качестве аргумента передается идентификатор удаляемого продукта (id) в виде целого числа.

Листинг 4.42. Представление product_del

Примечание.

В демо-версии книги удалены листинги программ.

Это представление имеет параметр `id`, который принимает идентификатор удаляемого продукта. После этого в представлении создается объект `del_prod` и текстовое сообщение —

они загружаются в объект `context` и этот объект передается в шаблон del_yesno.html, где будет сформирована форма (рис 4.26).

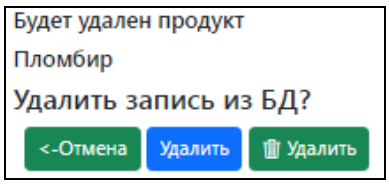


Рис. 4.26. Универсальный шаблон del_yesno.html с именем удаляемого продукта

В теле этой формы шаблона del_yesno.html пользователю отображаются имя удаляемого товара и две кнопки: **Отмена** — отменить удаление и **Удалить** — подтвердить удаление. Это именованные кнопки: кнопка **Отмена** имеет имя `name='no'`, кнопка **Удалить** — имя `name='yes'`. Имя нажатой кнопки будет передано в представление `product_del` в POST-запросе.

Напомним, что здесь для примера показаны два варианта кнопки **Удалить**: с синим фоном — простая кнопка для удаления, с зеленым фоном — кнопка для удаления с крутящимся спином.

По нажатию на любую кнопку выполняется возврат в представление `product_del`, где в инструкции `if 'yes' in request.POST` производится идентификация нажатой кнопки. Если пользователь нажал кнопку **Удалить**, будет выполнена инструкции удаления записи из БД, а если кнопку **Отменить**, эта инструкция будет пропущена, и в обоих случаях произойдет возврат к шаблону отображения сведений о продуктах edit_product.html.

4.4.8. Модули для показа подробностей о продуктах

Название товара в шаблоне edit_product.html представлено в виде ссылки:

```
<a href="{% url 'master:product_detail' pk=object.id %}">
    <i class="bi-search"></i> {{object.name}}</a>
```

Эта ссылка адресует к URL-маршруту `product_detail` (листинг 4.43) и передает в него первичный ключ текущей записи (`pk`).

Листинг 4.43. Маршрут URL product_detail

Примечание.
В демо-версии книги удалены листинги программ.

Этот маршрут принимает первичный ключ текущей записи (pk) и адресует к классу Django — представлению ProductDetail (листинг 4.44).

Листинг 4.44. Представление — класс ProductDetail

Примечание.
В демо-версии книги удалены листинги программ.

Фактически, это представление является классом, производным от класса Django с именем `DetailView`. В этом классе в объект `model` загружается имя модели (в нашем случае — модель `Product`), после чего класс по умолчанию вызывает шаблон product_detail.html (имя модели + `_detail`), код которого приведен в листинге 4.45.

Листинг 4.45. Шаблон product_detail.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне имеются кнопка **Назад** — для возврата к предыдущей странице сайта, и строка **Описание продукта**, в которую выводятся подробные данные о продукте (рис 4.27).



Рис. 4.27. Шаблон product_detail.html с детальной информацией о продукте

4.5. Фильтрация сведений о продуктах с использованием статичного древовидного каталога категорий продуктов

В предыдущих разделах (на примерах категорий продуктов) было показано, как создать древовидные структуры и списки самих продуктов. Здесь же мы на основе моделей `Product` (продукты) и `Genre_m` (категории продуктов) разработаем набор необходимых модулей, позволяющих фильтровать сведения о продуктах с использованием связанных с ними древовидных категорий.

Создадим страницу сайта, имеющую две панели: левую боковую — со статичным деревом категорий продуктов и центральную — с карточками продуктов. Для доступа к такой странице нужно в главном меню или на любой странице сайта создать следующую ссылку (листинг 4.46).

Листинг 4.46. Ссылка на страницу со статичным деревом категорий продуктов

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке вызывается URL-маршрут `catalog_stat` (листинг 4.47).

Листинг 4.47. Маршрут URL с именем catalog_stat

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `catalog_stat` (листинг 4.48).

Листинг 4.48. Представление catalog_stat

Примечание.

В демо-версии книги удалены листинги программ.

Здесь из БД в объект `genre` загружается каталог (категории продуктов) и формируется объект `context`, который передается в шаблон catalog_stat.html (листинг 4.49).

Листинг 4.49. Шаблон catalog_stat.html

Примечание.

В демо-версии книги удалены листинги программ.

В левой панели этого шаблона (рис. 4.28) открывается статичное дерево с категориями продуктов, а в центральной панели пользователю в виде текстового сообщения предлагается сделать выбор одной из категорий продукта из статичного древовидного каталога. Кроме того, в левой панели имеется кнопка **Все продукты**, с которой связана следующая ссылка:

```
<a href="{% url 'master:product_stat' id=0 %}" class="btn btn-success btn-sm">
                    <i class="bi bi-cart4"></i> Все продукты</a>
```

По этой ссылке в конечном итоге вызывается представление `product_stat`, в которое передается нулевой идентификатор категории продуктов (`id=0`). В этом случае в центральной панели должны быть показаны все продукты из БД.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Статичный каталог продуктов**.

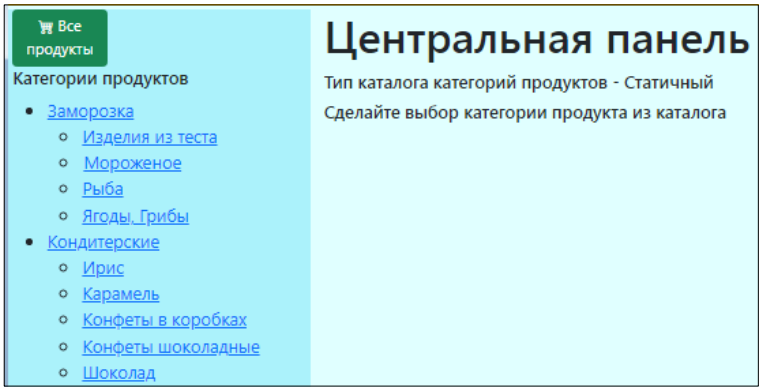


Рис. 4.28. Внешний вид шаблона catalog_stat.html

Рассмотрим, как в этом шаблоне и соответствующем представлении работает фильтрация данных. Каждая ветка дерева категорий продуктов в левой части окна шаблона catalog_stat.html показана как ссылка:

```
<a href="{% url 'master:product_stat' id=node.id %}">{{ node.name }}</a>
```

По этой ссылке активируются URL-маршрут `product_stat` (листинг 4.50), в который передается идентификатор (`id`) выбранной категории продукта.

Листинг 4.50. URL-маршрут product_stat

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `product_stat` (листинг 4.51) с аргументом выбранной категории (`id`).

Листинг 4.51. Представление product_stat

Примечание.

В демо-версии книги удалены листинги программ.

Это представление в качестве параметра принимает идентификатор выбранной категории (`id`). Поскольку в этом случае `id` не равно нулю, то идентификатор категории принимает значение: `id_cat = id`, и на основе этого значения данные о категориях товара фильтруются в следующих инструкциях:

```
# выбранная категория в каталоге
select_cat = Genre_m.objects.get(id=id_cat)
  . . .
# категории, в которые входит выбранная категория и все дочерние категории
categories = select_cat.get_descendants(include_self=True)
```

Затем на основе объекта `categories` фильтруются данные о продуктах:

```
products = Product.objects.filter(genre__in=categories).distinct().order_by('name')
```

Если категория продукта не выбрана (идентификатор категории `id=0`), то из БД будут выбраны записи обо всех продуктах:

```
products = Product.objects.all()
```

После фильтрации данных о продуктах создается объект `paginator`, обеспечивающий постраничный вывод данных о продуктах, и все полученные параметры передаются в объект `context`, который загружается в шаблон product_stat.html (листинг 4.52).

Листинг 4.52. Шаблон product_stat.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне имеется тег организации цикла `{% for mb in page_obj %}`, в котором в виде карточек выводятся сведения о продуктах, а также присутствует пагинатор для постраничного вывода данных.

Если в левой боковой панели шаблона catalog_stat.html с помощью мыши выбрать категорию продукта, то в центральной панели будут показаны карточки с информацией о продуктах выбранной категории (рис. 4.29).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Статичный каталог продуктов**.

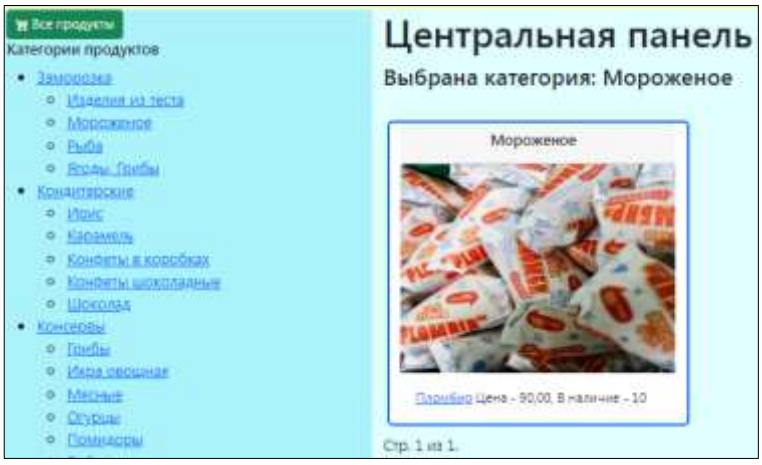


Рис. 4.29. Внешний вид шаблона catalog_stat.html с информацией о продуктах выбранной категории

Если в шаблоне catalog_stat.html нажать на кнопку **Все продукты**, то в центральной панели будут показаны карточки с информацией обо всех продуктах (рис. 4.30).

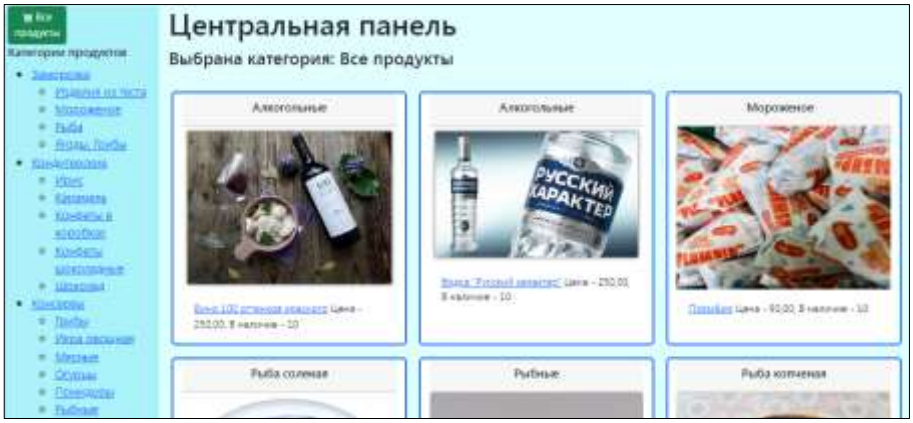


Рис. 4.30. Внешний вид шаблона catalog_stat.html с информацией о всех продуктах

Как здесь можно видеть, в карточках наименование продукта представлено в виде ссылки, и если щелкнуть на ней левой кнопкой мыши, то откроется страница с подробными сведениями о выбранном продукте (рис. 4.31).



Рис. 4.31. Страница с подробной информацией о продукте

4.6. Фильтрация сведений о продуктах с использованием динамичного древовидного каталога категорий продуктов

В предыдущем разделе было показано, как можно фильтровать сведения о продуктах с использованием связанных с ними древовидных категорий. Для этого мы создали страницу сайта, на которой в левой боковой панели присутствовало полностью раскрытое статичное дерево категорий продуктов. Работать с таким деревом удобно при небольшом количестве категорий продуктов и при небольшом количестве уровней вложенности. Однако на практике часто возникают ситуации, когда структура категорий продуктов имеет сложную многоуровневую организацию с большим количеством записей. В этом случае пользователю удобнее работать с динамичным (раскрывающимся) деревом категорий продуктов.

Создадим страницу сайта, которая будет иметь две панели: левую боковую — с динамичным (раскрывающимся) деревом категорий продуктов и центральную — с карточки про-

дуктов. Для доступа к такой странице нужно в главном меню или на любой странице сайта создать следующую ссылку (листинг 4.53).

Листинг 4.53. Ссылка на страницу с динамичным деревом категорий продуктов

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет вызваться URL-маршрут `catalog_tree` (листинг 4.54).

Листинг 4.54. URL=маршрут catalog_tree

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `catalog_tree` (листинг 4.55).

Листинг 4.55. Представление catalog_tree

Примечание.

В демо-версии книги удалены листинги программ.

Здесь из БД в объект `genre` загружается каталог (категории продуктов) и формируется объект `context`, который передается в шаблон catalog_tree.html (листинг 4.56).

Листинг 4.56. Шаблон catalog_tree.html

Примечание.

В демо-версии книги удалены листинги программ.

В левой панели этого шаблона открывается динамичное (раскрывающееся) дерево с категориями продуктов, а в центральной панели пользователю в текстовой строке предлагается сделать выбор одной из категорий продукта из динамичного древовидного каталога (рис. 4.32). Кроме того, в левой панели имеется кнопка **Все продукты**, с которой связана следующая ссылка:

```
<a href="{% url 'master:product_tree' id=0 %}" class="btn btn-success btn-sm">
                    <i class="bi bi-cart4"></i> Все продукты</a>
```

По этой ссылке в конечном итоге вызывается представление `product_tree`, в которое передается нулевой идентификатор категории продуктов (`id=0`). В этом случае в центральной панели должны быть показаны все продукты из БД.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Динамичный каталог продуктов**.

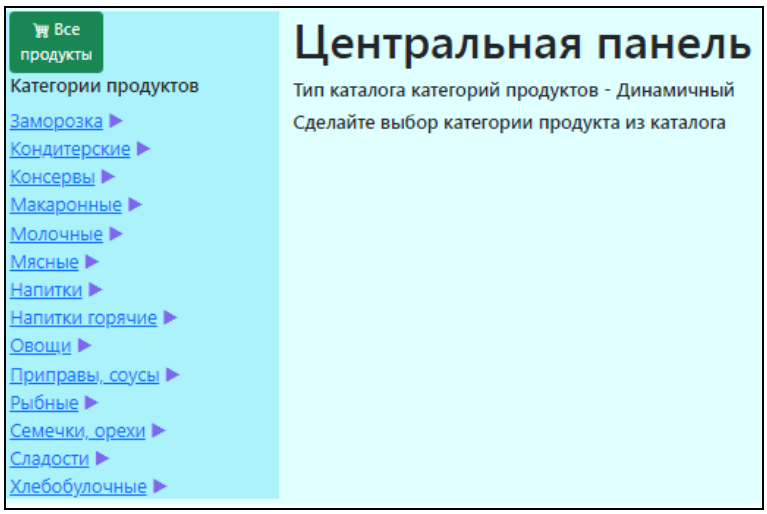


Рис. 4.32. Внешний вид шаблона catalog_tree.html при первоначальной загрузке

Как можно увидеть на рис. 4.32, в левой боковой панели страницы дерево категорий продуктов находится в свернутом виде (показаны только корневые узлы — в нашем примере их несколько). Рядом с каждым корневым узлом находится кнопка ▶, по нажатию на которую раскроются дочерние узлы нижнего уровня (рис. 4.33). При повторном нажатии на кнопку раскрытого узла ▼ дочерние уровни закроются.

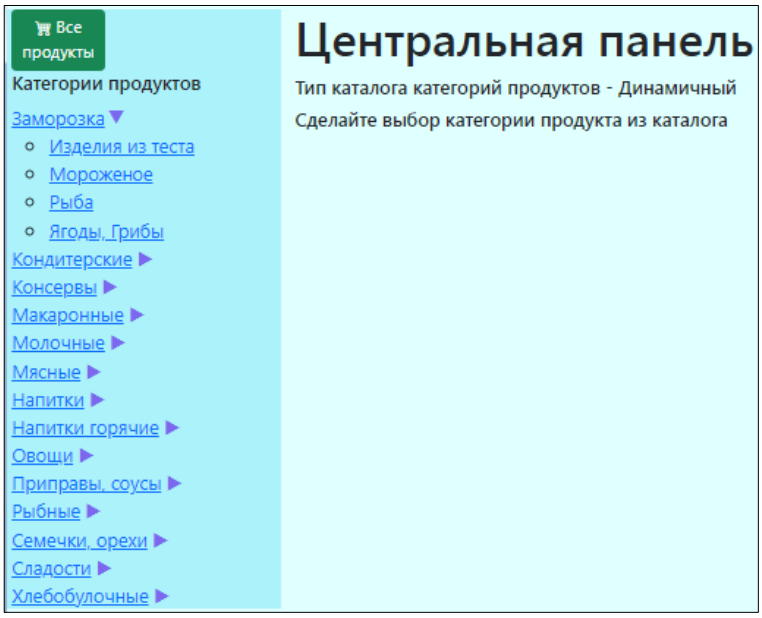


Рис. 4.33. Внешний вид шаблона catalog_tree.html с раскрытым узлом **Заморозка**

Рассмотрим, как в этом шаблоне и соответствующем представлении работает фильтрация данных. Обратите внимание, что каждая ветка дерева категорий продуктов в левой панели окна шаблона catalog_tree.html показана как ссылка:

```
<a href="{% url 'master:product_tree' id=node.id %}">{{ node.name }}</a>
```

По этой ссылке активируются URL-маршрут `product_tree` (листинг 4.57), в который передается идентификатор (`id`) выбранной категории продукта.

Листинг 4.57. Маршрут URL product_tree

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `product_tree` (листинг 4.58) с аргументом выбранной категории (`id`).

Листинг 4.58. Представление product_tree

Примечание.

В демо-версии книги удалены листинги программ.

Это представление в качестве параметра принимает идентификатор выбранной категории продукта (`id`). Поскольку в этом случае `id` не равно нулю, то идентификатор категории принимает значение: `id_cat = id`, и на основе этого значения данные о категориях товара фильтруются в следующих инструкциях:

```
# выбранная категория в каталоге
select_cat = Genre.objects.get(id=id_cat)
  . . .
# категории, в которые входит выбранная категория и все дочерние категории
categories = select_cat.get_descendants(include_self=True)
```

Затем на основе объекта `categories` фильтруются данные о продуктах:

```
products = Prod-
uct.objects.filter(genre__in=categories).distinct().order_by('name')
```

Если категория продукта не выбрана (идентификатор категории `id=0`), то из БД будут выбраны записи обо всех продуктах:

```
products = Product.objects.all()
```

После фильтрации данных о продуктах создается объект `paginator`, обеспечивающий постраничный вывод данных о продуктах, и все полученные параметры передаются в объект `context`, который загружается в шаблон product_tree.html (листинг 4.59).

Листинг 4.59. Шаблон product_tree.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне имеется тег организации цикла `{% for mb in page_obj %}`, в котором в виде карточек выводятся сведения о продуктах, а также присутствует пагинатор для постраничного вывода данных.

Если в левой боковой панели шаблона catalog_stat.html с помощью мыши выбрать категорию продукта, то в центральной панели будут показаны карточки с информацией о продуктах выбранной категории (рис. 4.34).



Рис. 4.34. Внешний вид шаблона catalog_tree.html с информацией о продуктах выбранной категории

Если в шаблоне catalog_tree.html нажать на кнопку **Все продукты**, то в центральной панели будут показаны карточки с информацией обо всех продуктах (рис. 4.35).

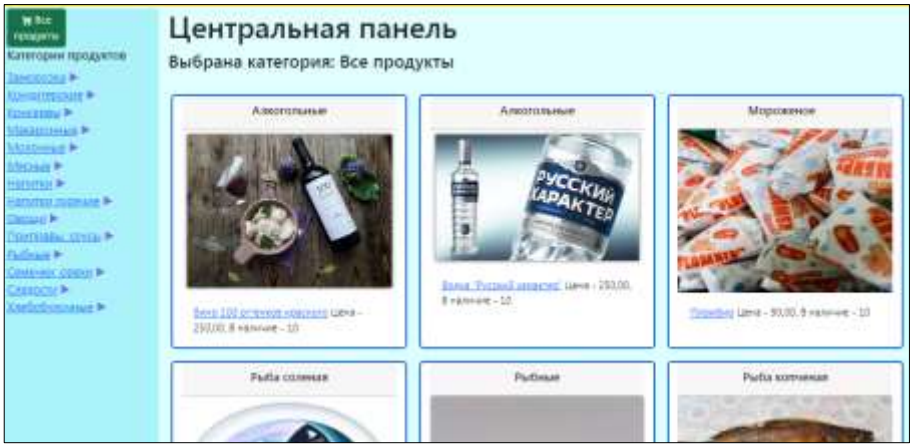


Рис. 4.35. Внешний вид шаблона catalog_stat.html с информацией обо всех продуктах

Как здесь можно видеть, в карточках наименование продукта представлено в виде ссылки. Если на ней щелкнуть левой кнопкой мыши, то откроется страница с подробными сведениями о выбранном продукте (рис. 4.36).



Рис. 4.36. Подробные сведения о продукте

4.7. Создание древовидных меню с использованием плагина megamenu-js

В предыдущих разделах на примере категорий продуктов было показано, как создавать древовидные структуры. Здесь мы рассмотрим набор модулей, которые позволяют отображать на страницах сайта категории продуктов в виде многоуровневого горизонтального меню. Для этого будет задействован специальный плагин — megamenu-js.

4.7.1. Плагин megamenu-js: подготовка к работе

Плагины

Плагины — это небольшие программные модули, которые добавляют новую функциональность к основной программе. Они помогают улучшить ее производительность, обогатить дизайн, расширить возможности пользователей.

Скачайте плагин megamenu-js[1], выбрав на странице его сайта опции: **Code | Download ZIP** (рис. 4.37).

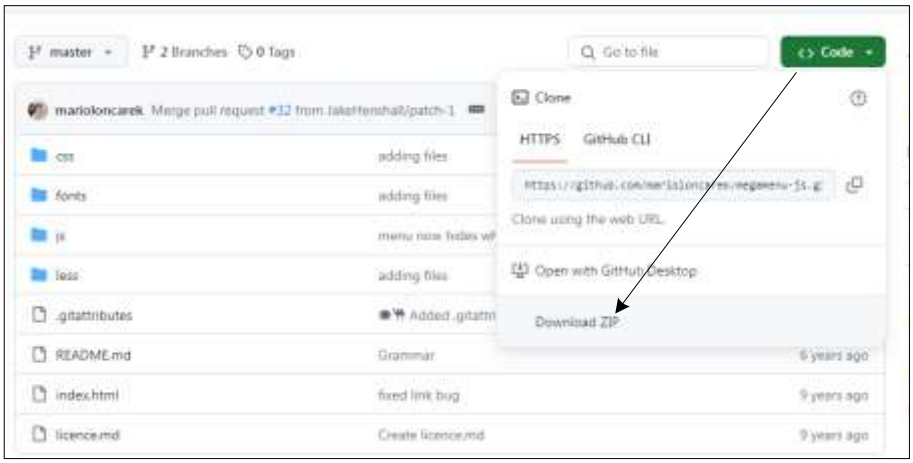


Рис. 4.37. Страница для скачивания архива с плагином megamenu-js

Скачанный файл с плагином megamenu-js будет иметь имя megamenu-js-master.zip. После распаковки мы получим каталог megamenu-js-master, содержащий следующие подкаталоги и файлы (рис. 4.38).



Рис. 4.38. Содержимое папки megamenu-js-master после распаковки архива

Из этого каталога надо взять содержимое трех подкаталогов: css, fonfs и js — и поместить находящиеся там файлы в соответствующие каталоги static вашего проекта.

Так, в каталог static/css вашего проекта нужно скопировать из подкаталога css плагина следующие файлы (рис. 4.39).

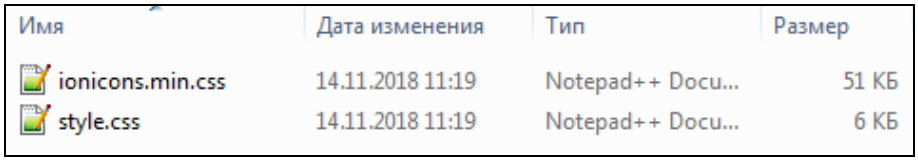


Рис. 4.39. Содержимое подкаталога css плагина megamenu-js

Подкаталог плагина fonts со всеми файлами полностью переносится в каталог static/fonts вашего проекта (рис. 4.40).

[1] См. **https://github.com/marioloncarek/megamenu-js**.

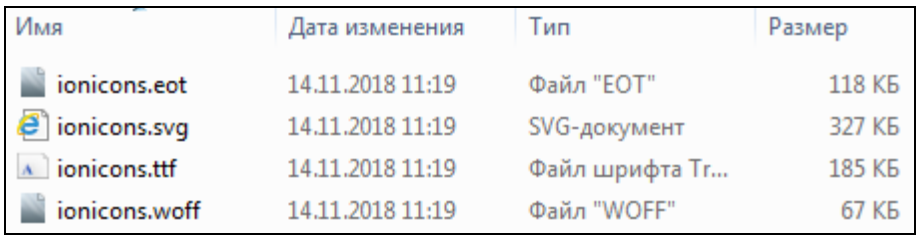


Рис. 4.40. Содержимое подкаталога fonts плагина megamenu-js

В каталог static/js вашего проекта переносятся следующие элементы из подкаталога js плагина (рис. 4.41).

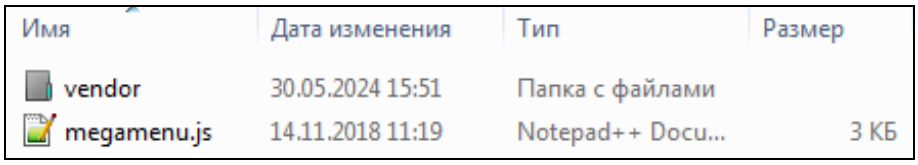


Рис. 4.41. Содержимое подкаталога js плагина megamenu-js

Для использования мегаменю в шаблонах сайта нужно к проекту подключить несколько скриптов и обернуть меню в следующие контейнеры:

```
<div class="menu-container">
    <div class="menu">
        <ul>
            <li>Элемент меню</li>
            <li>Элемент меню </li>
        </ul>
     </div>
</div>
```

Выполнив указанные действия, можно будет использовать плагин megamenu-js в своем проекте.

4.7.2. Создание мегаменю на основе плагина megamenu-js

Создадим мегаменю для выбора категорий продуктов. Чтобы подгрузить категории продуктов с использованием мегаменю, нужно либо в главном меню, либо на любой другой странице сайта создать следующую ссылку (листинг 4.60).

Листинг 4.60. Ссылка для вызова мегаменю

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке активируется URL-маршрут `mega_menu` (листинг 4.61).

Листинг 4.61. URL-маршрут mega_menu

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `mega_menu` (листинг 4.62).

Листинг 4.62. Представление mega_menu

Примечание.

В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон mega_menu.html (листинг 4.63).

Листинг 4.63. Шаблон mega_menu.html

Примечание.

В демо-версии книги удалены листинги программ.

Здесь подключаются элементы плагина мегаменю из каталога static:

```
<link rel="stylesheet", href="{% static 'css/ionicons.min.css' %}">
<link rel="stylesheet", href="{% static 'css/style.css' %}">
<script src="{% static 'js/megamenu.js' %}"></script>
```

Затем строится горизонтальное меню в виде свернутого дерева:

```
<div class="menu-container">
    <div class="menu">
        {% load mptt_tags %}
        <ul>
            {% recursetree categories %}
            <li>
                <a href="{% url 'master:get_category' id=node.id %}">{{ node.name
}}</a>
                {% if not node.is_leaf_node %}
                <ul class="children">
                    {{ children }}
                </ul>
                {% endif %}
            </li>
            {% endrecursetree %}
        </ul>
    </div>
</div>
```

Наконец создается блок, в котором выводится наименование выбранной пользователем категории продукта:

```
<div class="row text-start mx-2">
    <h3>{{ category.name }}</h3>
</div>
```

В шаблоне mega_menu.html для выбора категории продукта используется следующая ссылка:

```
<a href="{% url 'master:get_category' id=node.id %}">{{ node.name }}</a>
```

По этой ссылке активируется URL-маршрут `get_category` (листинг 4.64), в который передается идентификатор выбранной категории (`id=node.id`).

Листинг 4.64. Маршрут URL get_category

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `get_category` (листинг 4.65).

Листинг 4.65. Представление get_category

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении в объект `categories` из БД загружается полный список категорий продуктов, а в объект `category` — выбранная пользователем в мегаменю категория продуктов.

По умолчанию мегаменю занимает 80% ширины контейнера. Для того чтобы меню было представлено в виде строки на всю ширину контейнера, необходимо отредактировать файл style.css — откройте его в любом текстовом редакторе, найдите в нем фрагмент кода:

```
.menu-container {
    width: 80%;
    margin: 0 auto;
    background: #e9e9e9;
}
```

и измените этот код следующим образом, указав новое значение для ширины контейнера для меню — `100%`:

```
.menu-container {
    width: 100%;
    margin: 0 auto;
    background: #e9e9e9;
```

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Пример мегаменю**».

Запустив шаблон с мегаменю на выполнение, мы получим горизонтальное меню с категориями продуктов (рис. 4.42).

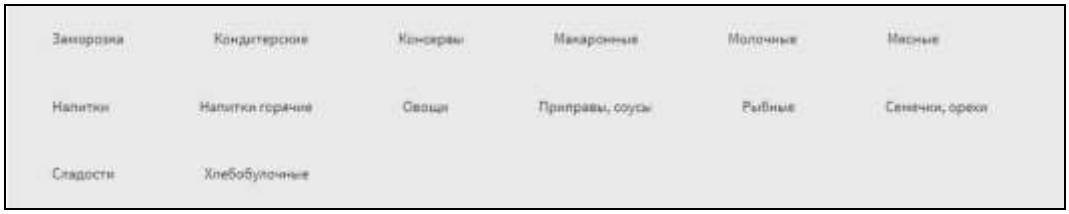


Рис. 4.42. Горизонтальное меню категорий продуктов на основе плагина megamenu-js

Как здесь можно видеть, в строке меню показаны только корневые элементы дерева категорий продуктов. Но при наведении курсора мыши на любой из элементов этого меню появляется выпадающий список с его дочерними элементами (рис. 4.43).



Рис. 4.43. Мегаменю с открытой вкладкой категорий продуктов **Мясные**

По щелчку левой кнопкой мыши на любом элементе мегаменю выбранный элемент будет отображен в нижней части шаблона (рис. 4.44).

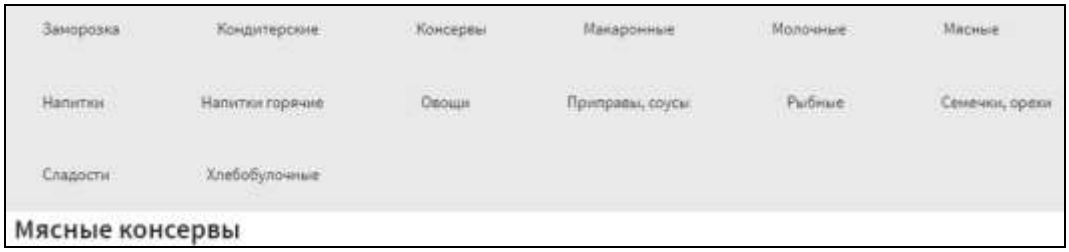


Рис. 4.44. Мегаменю с выбранной категорией продуктов **Мясные консервы**

Мегаменю, показанные на рис. 4.42–4.44, выглядят не очень привлекательно: между их элементами присутствуют слишком большие интервалы, да и выделенные элементы отображаются не очень четко. Немного изменим оформление мегаменю, для чего внесем некоторые изменения в файл стилей css. Найдите в нем следующий блок:

```
.menu > ul > li a {
    text-decoration: none;
    padding: 1.5em 3em;
    display: block;
}
```

и поменяйте величину `padding` — чтобы меню поместилось в одну строку (задаем в строке `padding` значение `1em 1em`):

```
.menu > ul > li a {
    text-decoration: none;
    padding: 1em 1em;
    display: block;
}
```

Затем найдите в файле css блок:

```
.menu > ul > li > ul.normal-sub > li a {
    border: 0;
    padding: 1em 0;
}
```

и уменьшите вертикальные отступы (задаем в строке `padding` значение `0.1em`).

```
.menu > ul > li > ul.normal-sub > li a {
    border: 0;
    padding: 0.1em 0;
}
```

Теперь добавьте после этого блока новый блок — чтобы в подменю выделить синим цветом опцию, над которой находится мышь:

```
.menu > ul > li > ul.normal-sub a:hover{
background: #0078BF;
color: #fff;
}
```

После проведенных изменений мегаменю становится компактнее, может вписаться в одну строку окна браузера, а опция, над которой находится мышь, явно выделяется синим цветом (рис. 4.45).

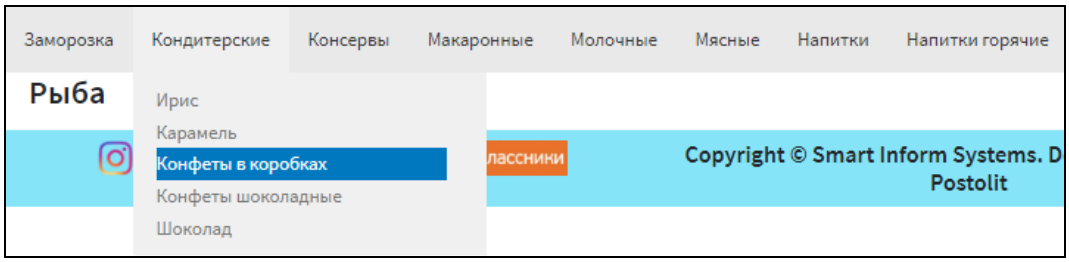


Рис. 4.45. Мегаменю с измененным интерфейсом

4.8. Фильтрация продуктов с использованием панели инструментов и древовидного каталога категорий

В предыдущих разделах на примере категорий продуктов было показано, как создавать древовидные структуры, как организовать список продуктов с привязкой к категориям и как выводить категории продуктов в виде горизонтального многоуровневого меню.

Рассмотрим теперь набор модулей, которые позволят отображать на страницах сайта сведения о продуктах, а также выбирать и фильтровать эти данные с учетом категорий. Кроме того, мы научимся создавать панель инструментов (типа toolbar) для представления элементов мегаменю в виде значков со всплывающими подсказками. И в этом нам поможет уже знакомый по предыдущему разделу плагин megamenu-js —понадобится лишь немного изменить модель категорий продуктов, добавив в нее поле `image`. Однако чтобы в цифровой интерактивной книге сохранить функциональность модулей предыдущих разделов, мы создадим новые модели для категорий продуктов и для самих продуктов.

В модели со значками категорий продуктов, в отличие от аналогичных моделей из примеров предыдущих разделов, предусмотрено дополнительное поле `image`, обеспечивающее возможность показа значков в панели инструментов (toolbar). Для загрузки файлов значков

нам также понадобится создать дополнительную функцию изменения имен файлов с изображениями (`rename_genre`). Код этой функции и модели для категорий продуктов `Genre_i` приведен в листинге 4.66.

Листинг 4.66. Функция rename_genre и модель для категорий продуктов Genre_i

Примечание.
В демо-версии книги удалены листинги программ.

Как уже отмечено ранее, в этой модели, в отличие от ранее рассмотренных моделей для древовидных структур, имеется дополнительное поле `image`, в которое будут загружаться значки для панели инструментов (toolbar). Чтобы исключить ошибки при загрузке изображений с именами файлов на кириллице, эту модель сопровождает функция автоматического переименования файлов загружаемых изображений.

Пояснение
Поскольку публичные серверы, работающие на Linux, не принимают файлы с именами на кириллице, использование этой функции позволит автоматически переименовывать такие файлы.

Создадим также новую модель для продуктов `Product_i`, связанную с моделью категорий продуктов `Genre_i` (листинг 4.67).

Листинг 4.67. Модель Product_i

Примечание.
В демо-версии книги удалены листинги программ.

Не забудьте теперь выполнить миграцию:

```
python manage.py makemigrations
python manage.py migrate
```

Для реализации возможности ввода данных о продуктах и категориях продуктов через административную панель Django зарегистрируем в файле admin.py модели `Product_i` и `Genre_i` (листинг 4.68).

Листинг 4.68. Регистрация моделей Product_i и Genre_i в файле admin.py

Примечание.
В демо-версии книги удалены листинги программ.

Заполним модель `Genre_i` несколькими категориями продуктов с использованием административной панели Django (рис. 4.46).

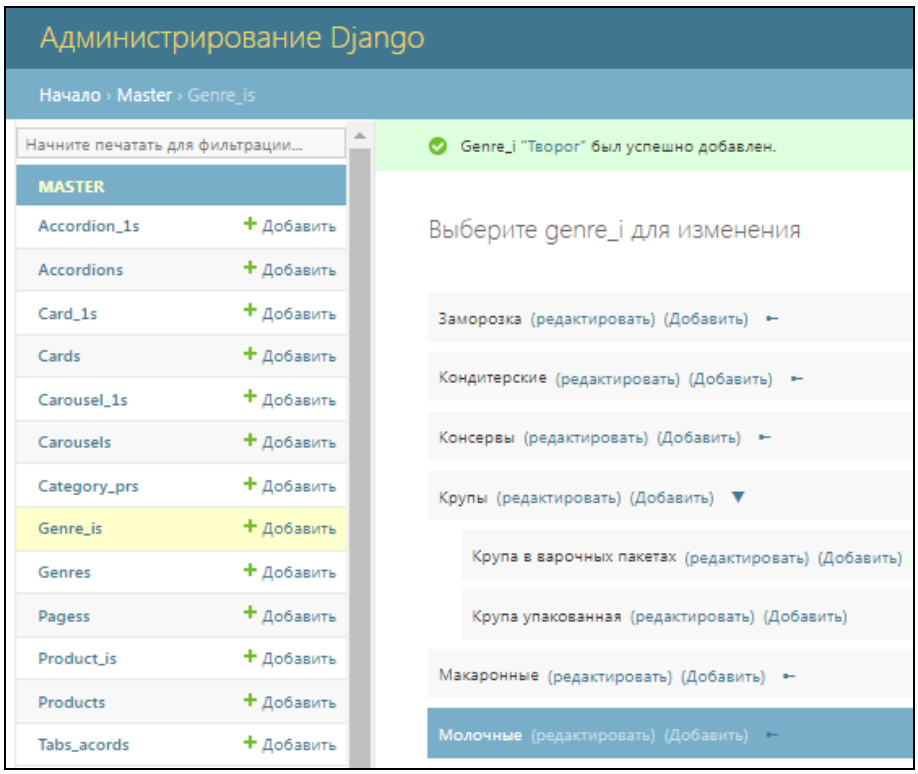


Рис. 4.46. Модель Genre_i с заполненными категориями продуктов

Заполним для нашего примера также с использованием административной панели Django модель `Product_i` несколькими продуктами разных категорий (рис. 4.47).

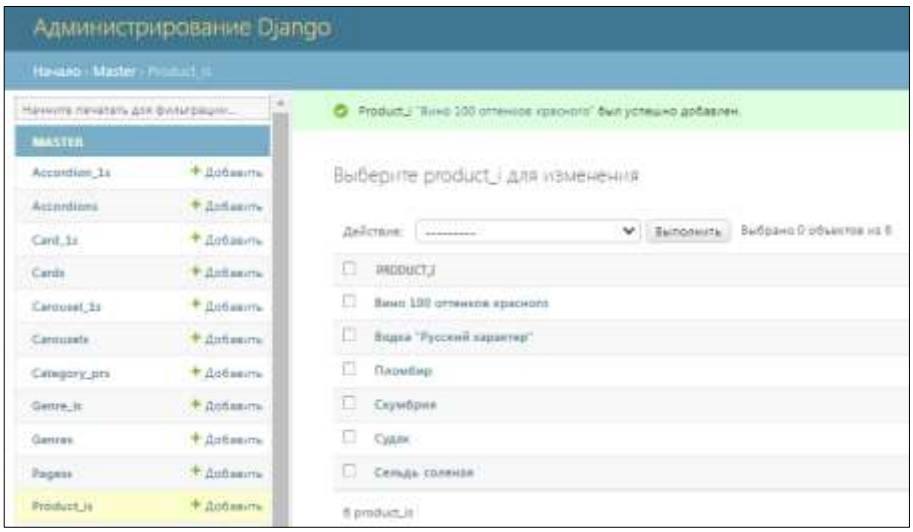


Рис. 4.47. Модель `Product_i` с заполненными продуктами разных категорий

Для отображения страницы с карточками продуктов и с возможностью выбора категории продукта из мегаменю или из панели инструментов (toolbar) нужно в главном меню или на любой другой странице сайта создать следующую ссылку (листинг 4.69).

Листинг 4.69. Ссылка на страницу вывода данных о продуктах и панели инструментов toolbar

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке вызывается URL-маршрут `toolbar_up` (листинг 4.70).

Листинг 4.70. URL-маршрут toolbar_up

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `toolbar_up` (листинг 4.71).

Листинг 4.71. Представление toolbar_up

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении в объект `products` из БД загружаются все продукты, а в объект `num_products` — количество записей обо всех продуктах. Затем в объект `categories` загружаются все категории продуктов из БД, а в объект `genre_toolbar` — категории продуктов только для панели инструментов (toolbar). Здесь на категории продуктов наложен фильтр `level=0` — то есть в этом объекте будет находиться список категорий продуктов самого верхнего уровня, используемый для формирования значков панели инструментов (toolbar). В завеершение создается пагинатор для постраничного вывода продуктов (30 элементов на странице). Всё это через словарь `context` передается в шаблон toolbar_up.html (листинг 4.72).

Листинг 4.72. Шаблон toolbar_up.html

Примечание.

В демо-версии книги удалены листинги программ.

Шаблон toolbar_up.html имеет следующую структуру:

```
{% block content %}
    Кнопка для показа всех продуктов
    Строка мегаменю с категориями продуктов
    {% include "…/toolbar.html" %}                ← Это tollbar со значками
    {% block products %}
        {% include "…/products_all.html" %}       ← Это карточки с продуктами
    {% endblock products %}
{% endblock conten %}
```

В верхней части шаблона toolbar_up.html показывается количество продуктов БД и представлена кнопка для загрузки из БД всех продуктов. В средней части выводится горизонтальное мегаменю с категориями продуктов, имеющее раскрывающиеся подкатегории. Ниже следует панель инструментов (toolbar) со значками категорий продуктов верхнего уровня. Сама панель инструментов находится в подключаемом шаблоне toolbar.html (листинг 4.73). После панели инструментов идет блок `products`, в который вложен шаблон show_products.html (листинг 4.74), демонстрирующий карточки с продуктами. В нем при первой загрузке будут показаны карточки всех продуктов, которые внесены в БД, то есть фильтрация по категориям продуктов в этом случае не производится.

Листинг 4.73. Шаблон toolbar.html

Примечание.

В демо-версии книги удалены листинги программ.

В шаблоне toolbar.html в цикле по полю `genre_toolbar` отображаются карточки, содержащие только значки корневых категорий продуктов размером 200×200 пикселов. Вывод каждого значка реализуется по ссылке на страницу с продуктами с фильтром по выбранной категории: `href="{% url 'master:product_up' id=mb.id %}"`. Кроме того, такой зна-

чок сопровождается всплывающей подсказкой: `data-toggle="tooltip" data-placement="right" title="{{ mb.name|safe }}"`, в которой отображается имя корневой категории продукта. Вывод всплывающей подсказки обеспечивает Java-скрипт:

```
<script>
    $(document).ready(function(){
        $('[data-toggle="tooltip"]').tooltip();
    });
</script>
```

Листинг 4.74. Шаблон show_products.html

Примечание.

В демо-версии книги удалены листинги программ.

В шаблоне show_products.html показываются карточки продуктов из БД с постраничным выводом по 30 карточек на странице. Карточки отображаются в цикле, который создан с использованием тега `{% for mb in page_obj %}`. Обратите внимание, что наименование продукта `name` представляет собой ссылку на URL-маршрут `product_detail_i` (листинг 4.75), в который передается первичный ключ `pk` (идентификатор) продукта:

```
<a href="{% url 'master:product_detail_i' pk=mb.id %}">{{mb.name}}</a>
```

Листинг 4.75. URL-маршрут product_detail_i

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление на основе класса `ProductDetail_i` (листинг 4.76).

Листинг 4.76. Представление на основе класса ProductDetail_i

Примечание.

В демо-версии книги удалены листинги программ.

В этом классе создается объект `model` на основе модели `Product_i`. По умолчанию класс использует шаблон product_i_detail.html (листинг 4.77), показывающий полноразмерную фотографию и подробную информацию о продукте.

Листинг 4.77. Шаблон product_i_detail.html

Примечание.

В демо-версии книги удалены листинги программ.

Вернемся к шаблону toolbar_up.html (см. листинг 4.73). В нем для фильтрации продуктов по категориям используется следующая ссылка:

```
<a href="{% url 'master:product_up' id=node.id %}">{{ node.name }}</a>
```

По этой ссылке активируется URL-маршрут `product_up` (листинг 4.78), в который передается идентификатор выбранной категории продукта (`id=node.id`).

Листинг 4.78. URL-маршрут product_up

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `product_up` (листинг 4.79), которое принимает идентификатор выбранной категории продукта (`id`).

Листинг 4.79. Представление product_up

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор категории продуктов (`id`), а в объект `genre_toolbar` загружаются только корневые категории продуктов — для панели инструментов (toolbar).

После этого в объект `genre` загружается полный список категорий продуктов, а объект `categories` получает из БД категории, в которые входит выбранная категория и все ее дочерние категории. В завершение в объект `products` загружаются из БД все продукты полученной категории и всех дочерних категорий (если дочерних категорий нет, то только полученной категории). Все эти данные через словарь `context` передаются в шаблон toolbar_up.html.

В шаблоне toolbar_up.html есть кнопка **Показать все продукты**, которая связана со следующей ссылкой:

```
<a href="{% url 'master:toolbar_up' %}">
```

По этой ссылке в конечном итоге будет вызвано представление `toolbar_up`, в котором снимаются фильтры на категории продуктов и отображается шаблон toolbar_up.html с карточками всех продуктов из БД.

Итак, мы разобрались с содержанием модулей, которые обеспечивают отображение как всех карточек продуктов из БД, так и отфильтрованных по дереву категорий продуктов. Посмотрим теперь, как выглядят наши шаблоны в окне браузера.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 4** и перейти по ссылке **Мегаменю и toolbar**».

Общий вид шаблона toolbar_up.html с карточками всех продуктов из БД показан на рис. 4.48, а на рис. 4.49 — демонстрируется подсказка с категорией продуктов, всплывающая, когда на значок наводится курсор мыши (см. листинг 4.73).

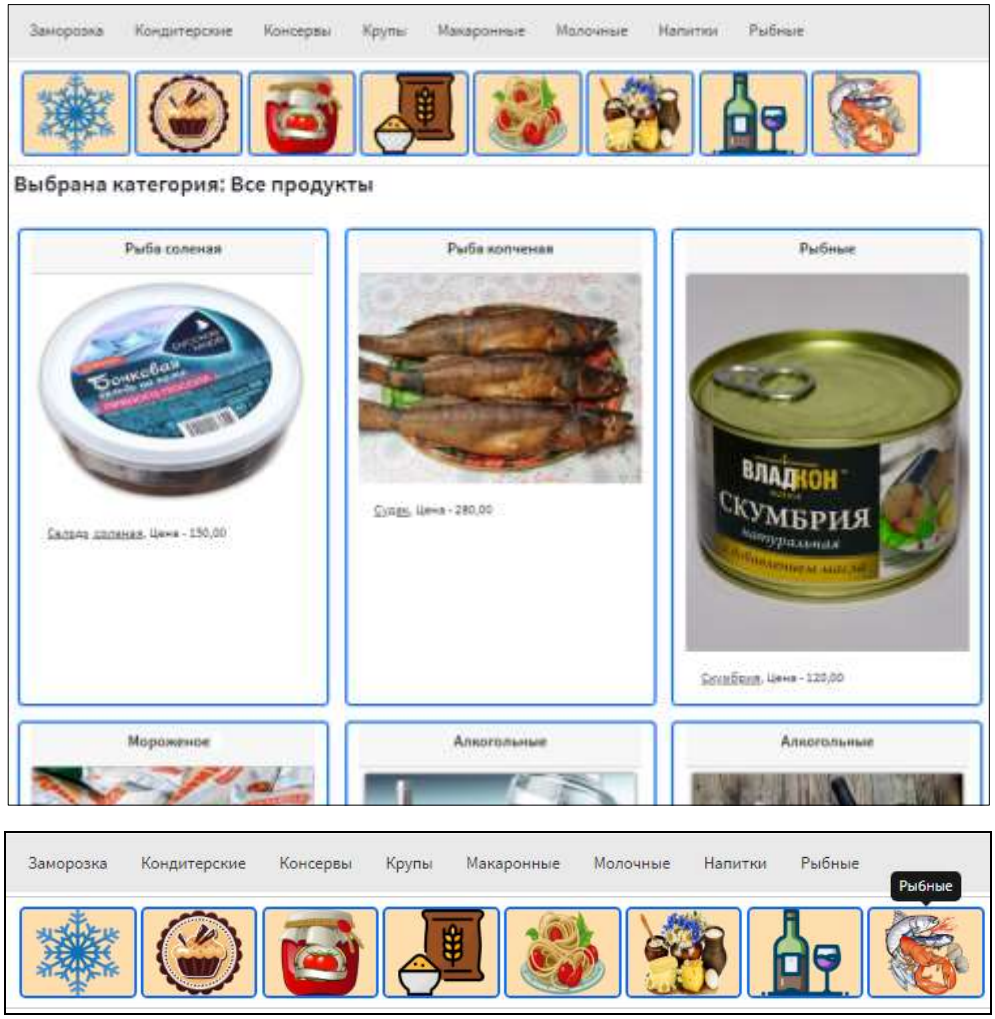


Рис. 4.49. Всплывающая подсказка категории продукта панели инструментов (toolbar)

Если щелкнуть левой кнопкой мыши на каком-либо значке панели инструментов (например, на значке **Рыбные**), то карточки продуктов будут отфильтрованы по этой категории продуктов с учетом всех дочерних категорий (рис. 4.50).

Аналогичную картину мы получим и в том случае, если будет выбрана категория **Рыбные** в строке мегаменю. При этом, когда пользователь перемещает указатель мыши над опциями строки мегаменю, то они автоматически открываются, отображая дочерние категории (рис. 4.51), а выбранная опция (здесь **Макаронные**) меняет цвет на синий. Когда же пользователь перемещает указатель мыши над опциями дочерних категорий мегаменю (рис 4.52), то активная опция (здесь **Рыба соленая**) отображается на синем фоне, а щелчок мышью на этой опции выводит карточку выбранного продукта (рис. 4.53).

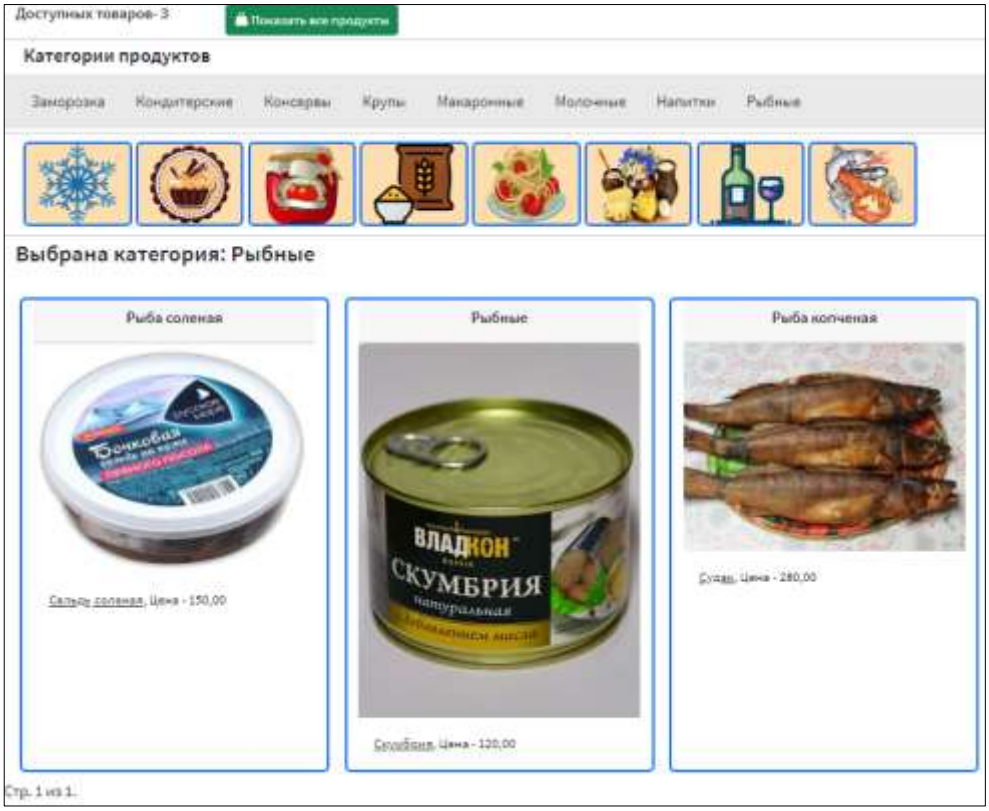


Рис. 4.50. Шаблон toolbar_up.html с карточками продуктов категории **Рыбные**

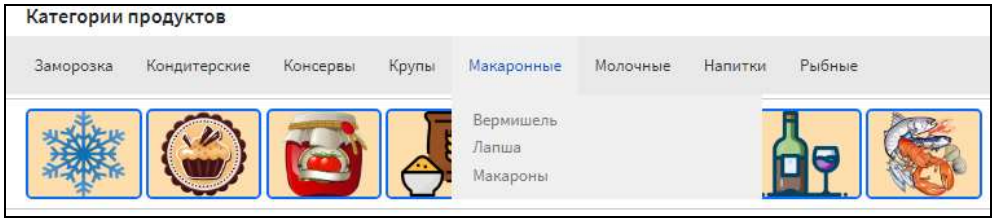


Рис. 4.51. Выбранная опция строки мегаменю с открытой вкладкой дочерних категорий



Рис. 4.52. Дочерняя активная опция мегаменю с подсветкой синим фоном

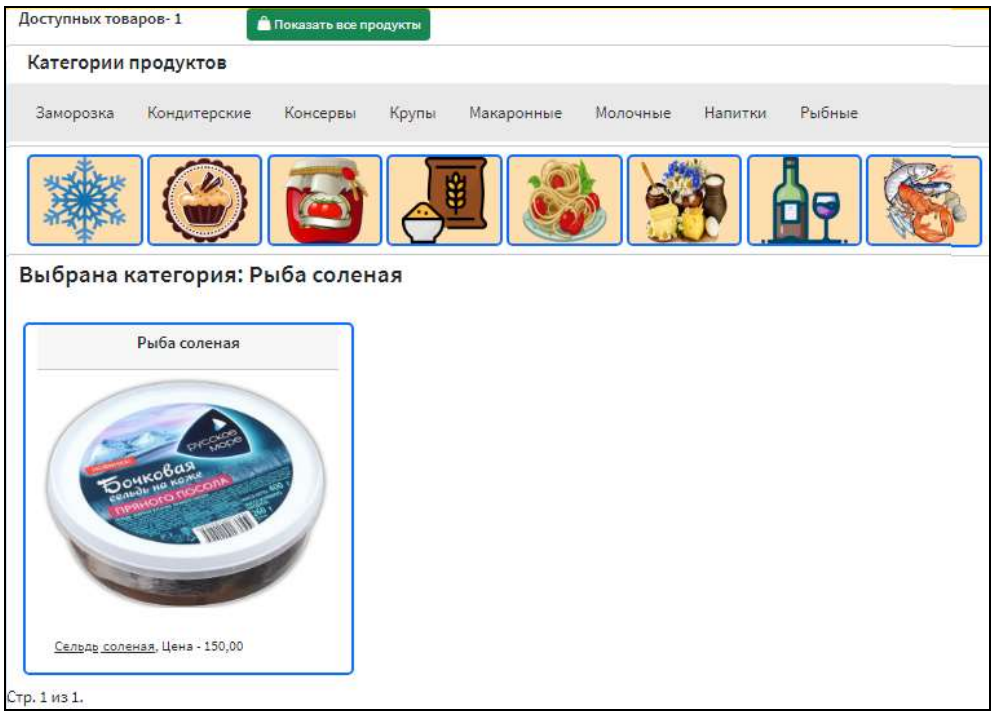


Рис. 4.53. Шаблон toolbar_up.html с карточкой продукта категории **Рыба соленая**

Если в шаблоне toolbar_up.html нажать на кнопку **Показать все продукты**, то фильтры на категории продуктов будут сняты, и на странице показаны карточки всех продуктов из БД (рис. 4.54).

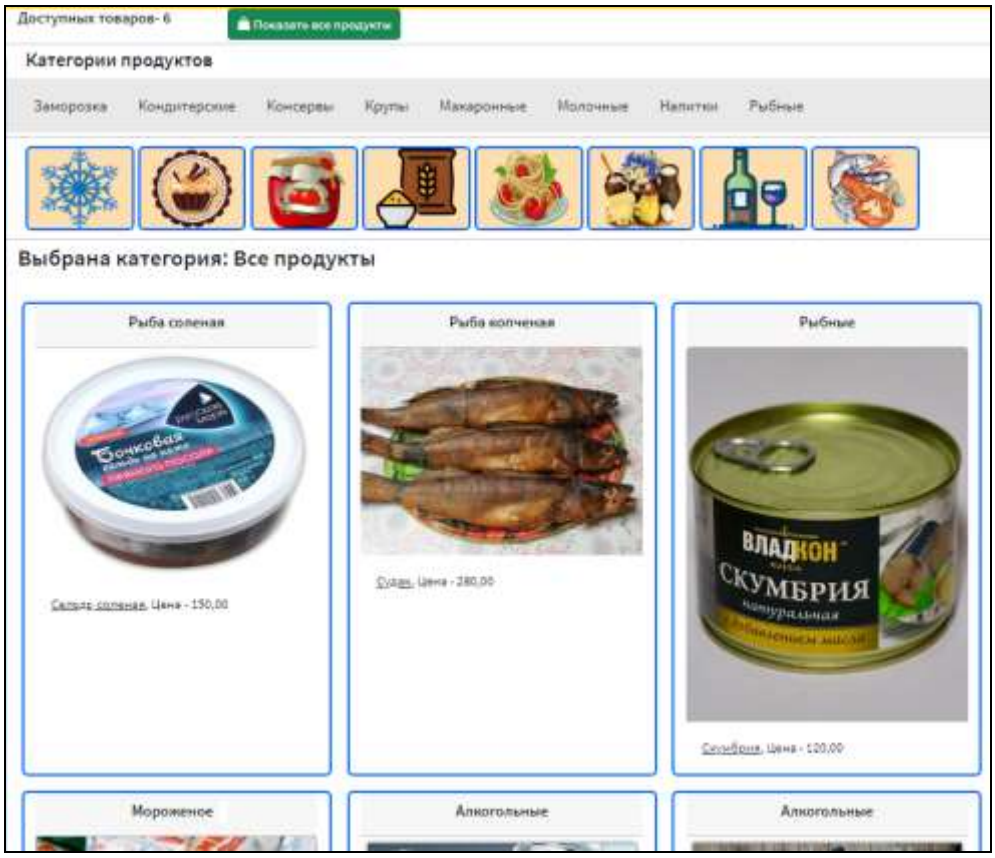


Рис. 4.54. Шаблон toolbar_up.html после нажатия на кнопку **Показать все продукты**

Если в шаблоне toolbar_up.html в карточке продукта щелкнуть левой кнопкой мыши на наименовании продукта, то откроется шаблон product_detail_i.html, в котором будет показана полноразмерная фотография продукта и его полное описание (рис. 4.55).



Рис. 4.55. Шаблон product_detail_i.html с фотографией и полным описанием продукта

4.9. Краткие итоги

В этой главе приведена общая информация о древовидном представлении данных в Django и показаны примеры программного кода для создания древовидных структур —эти программные модули вы можете встраивать в свои проекты. В следующей главе мы рассмотрим не менее важные вопросы использования модальных окон при разработке интерфейса взаимодействия пользователей с удаленными базами данных.

Глава 5. Библиотека HTMX: модальные окна для ввода и корректировки данных

Модальные (всплывающие) окна — это очень популярный элемент интерфейса современных сайтов. Он может использоваться для вывода различного контента веб-страниц — такого, например, как формы (обратной связи, регистрации, авторизации), блоки рекламной информации, изображения, уведомления и др. В большинстве случаев модальные окна создаются на JavaScript. Но их можно создать не только с помощью JavaScript, но и с использованием специальной библиотеки HTMX. Использованию этой библиотеки в шаблонах Django и посвящена эта глава, из материала которой вы узнаете:

- о преимуществах использования библиотеки HTMX в шаблонах Django;
- о том, как с ее помощью создаются программные модули для ввода и корректировки данных в модальных окнах.

Для многих код примеров этой главы с первого прочтения может показаться трудным и запутанным. Однако не стоит отчаиваться — надо просто сесть за компьютер и последовательно набрать и выполнить все приведенные здесь фрагменты программного кода. Да действительно, программировать интерфейс на основе модальных окон сложнее, чем на базе традиционных шаблонов Django. Но итоговый результат будет более удобен и привлекателен для конечных пользователей, а ведь именно для них создаются сайты. Если же вам удастся осмыслить изложенный здесь материал и разобраться, каким образом формируются модальные окна, то можно считать, что перед вами откроются дополнительные двери в сложный, но весьма увлекательный мир веб-программирования.

5.1. Библиотека HTMX

При вводе и корректировке данных пользователь чувствует себя более комфортно, когда основная страница с информацией об объектах не закрывается, а форма для ввода и корректировки данных появляется в виде модального окна, расположенного над основной страницей. Это можно реализовать с использованием специальных скриптов, задействовав в шаблонах элементы представленной в 2020 году программистом Карсоном Гроссом JavaScript-библиотеки HTML, позволяющей создавать современные веб-интерфейсы с помощью обычного кода HTML.

Существующие веб-приложения можно условно поделить на две большие группы:

- *многостраничные приложения (Multi-Page Application, MPA)*.

  MPA-приложения обновляют всю страницу при каждом взаимодействии пользователя с ней. При этом подходе состояние страницы обычно контролируется сервером. Однако перезагрузка страницы при каждом изменении может замедлять работу приложения;

- *одностраничные приложения (Single-Page Application, SPA)*.

  В этих приложениях управление состоянием страниц осуществляется с помощью кода JavaScript, выполняющегося в браузере. SPA-приложения взаимодействуют с сервером посредством вызовов API, возвращающих данные, обычно в формате JSON. После чего приложение обновляет интерфейс, не перезагружая страницу целиком.

Несмотря на очевидные плюсы SPA, у этой модели есть недостаток, с которым не все готовы мириться, — высокая сложность. Чтобы SPA работало как надо, браузер должен загрузить и выполнить несколько файлов JavaScript. Из-за этого SPA могут «тормозить» и задействовать много памяти, особенно, если программные модули плохо оптимизированы и каждую секунду забрасывают сервер десятками запросов.

Библиотека HTMX — это попытка уйти от указанных проблем, сохранив при этом современную функциональность приложений. Она позволяет достичь тех же целей, что и популярные фреймворки, но более простым путем, поскольку предоставляет возможность создавать динамичные веб-интерфейсы, ориентируясь только на HTML — с минимальным использованием кода JavaScript. Плюсы этого подхода заключаются в том, что мы можем запускать интерактивные элементы, написанные с помощью JAX, CSS Transitions, WebSockets и Server Sent Events, прямо из HTML-кода.

Библиотека HTMX сохраняет преимущества SPA, не требуя полной перезагрузки страницы, и в то же время по простоте она ближе к MPA. В предлагаемой этой библиотекой модели рендеринг страницы происходит на сервере, а клиенту передается уже готовый HTML-код, который затем встраивается в нужное место сайта. Браузеру при этом не нужно ничего интерпретировать и визуализировать — все эти вычисления берет на себя сервер. Такой подход облегчает процесс разработки за счет упрощения кода на стороне клиента.

Пример схемы взаимодействия представлений (view) с шаблонами при использовании библиотеки HTMX для реализации модальных окон при вводе и корректировке данных в процессе формирования списка объектов представлен на рис. 5.1.

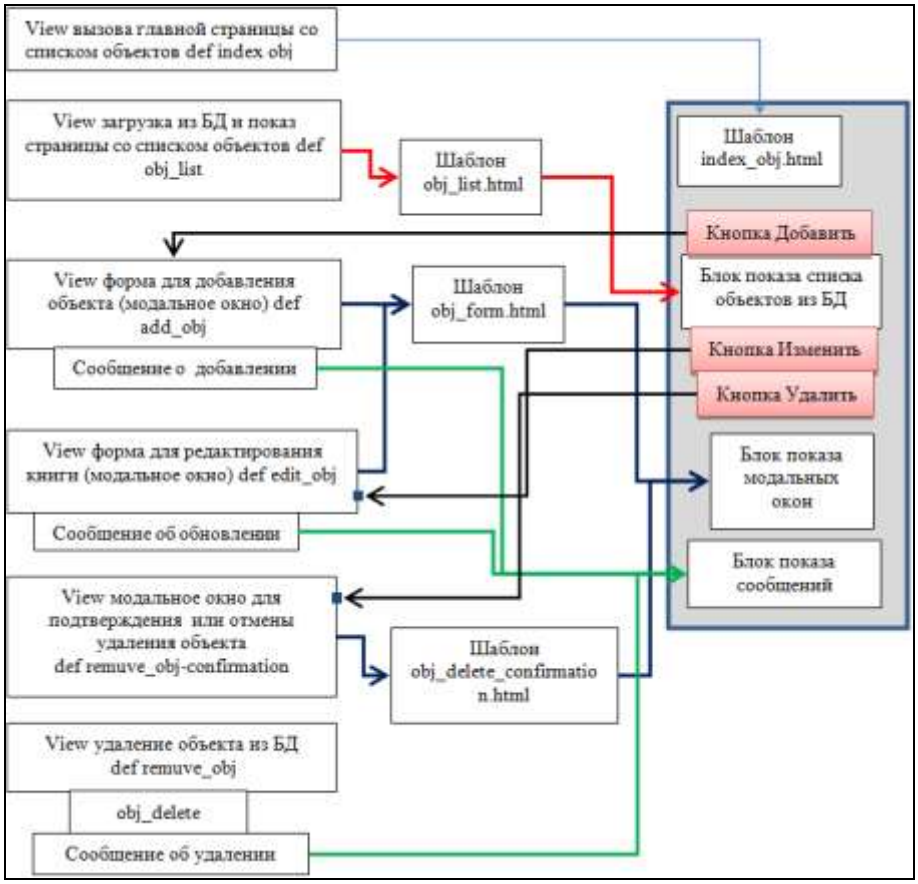


Рис. 5.1. Схема взаимодействия представлений (view) с шаблонами при использовании библиотеки HTMX

При такой реализации интерфейса взаимодействия пользователя с приложением в окне браузера пользователю всегда отображается только одна страница — index_obj.html. В эту страницу встроено несколько блоков:

- блок показа списка объектов из БД (виден постоянно);
- блок показа модальных окон (скрыт);
- блок показа сообщений (скрыт).

По мере взаимодействия пользователя с приложением страница index_obj.html всегда находится в окне браузера. А поверх нее появляются модальные окна в блоках показа модальных окон и показа сообщений. Сами тексты сообщений формируются в соответствующих представлениях: добавления объекта в БД (`def add_obj`), редактирования (обновления) сведений об объекте (`def edit_obj`), подтверждения или отмены удаления объекта из БД (`def remuve_obj_confirmation`), удаления объекта из БД (def remuve_obj).

Рассмотрим далее использование библиотеки HTMX с модулями ввода и редактирования сведений о клиентах сайта (это могут быть клиенты интернет-магазина, онлайн-библиотеки, социальной сети и т. п.).

5.2. Ввод и редактирования сведений о клиентах сайта с модальными окнами на основе библиотеки HTMX

5.2.1. Модель и форма для ввода данных

Создадим для нашего примера функцию `rename_foto_client`, обеспечивающую изменение имени файла с фотографией клиента, и модель для хранения данных о клиентах `Clients` (листинг 5.1).

> ***Пояснение***
> Поскольку публичные серверы, работающие на Linux, не принимают файлы с именами на кириллице, использование этой функции позволит автоматически переименовывать такие файлы. В результате имя файла с фотографиями клиентов получит следующую структуру: *client<id клиента в БД>.jpg*.

Листинг 5.1. Код функции rename_foto_client и модели Clients

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В этой модели для класса `Clients` созданы следующие поля:

- `name` — имя клиента (имя и фамилия);
- `adge` — возраст клиента;
- `about` — сведения о клиенте;
- `image` — изображение (фото клиента).

Мы не предполагаем заполнять данные о клиентах через административную панель Django, поэтому не станем регистрировать модель `Clients` в модуле admin.py.

Не забудьте теперь выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

Для ввода и корректировки данных о клиентах необходимо на основе модели `Clients` создать форму, код которого приведен в листинге 5.2.

Листинг 5.2. Форма ClientsForm

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В этой форме при вводе данных будут отображаться все поля модели `Clients`, при этом полю `about` (сведения о клиентах) назначен специальный виджет — `Textarea`.

5.2.2. Модули для создания редактора списка клиентов

Чтобы показать страницу с редактором списка клиентов, необходимо в главном меню сайта или на любой его странице создать соответствующую ссылку (листинг 5.3).

Листинг 5.3. Ссылка для вызова редактора списка клиентов

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этой ссылке активируется URL-маршрут `edit_clients` (листинг 5.4).

Листинг 5.4. URL-маршрут edit_clients

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `edit_clients` (листинг 5.5).

Листинг 5.5. Представление edit_clients

Примечание.

В демо-версии книги удалены листинги программ.

Это очень простое представление, которое активирует отображение шаблона edit_clients.html (листинг 5.6).

Листинг 5.6. Шаблон edit_clients.html

Примечание.

В демо-версии книги удалены листинги программ.

В программном коде шаблона edit_clients.html присутствуют несколько блоков:

- кнопка `Добавить клиента`, обеспечивающая вызов модального окна для добавления в БД нового объекта. Эта кнопка является элементом библиотеки HTMX и имеет следующие атрибуты:
  - `hx-get="{% url 'master:add_client' %}"` — отправляет GET-запрос на указанный URL-маршрут (для вызова представления добавления нового клиента);
  - `hx-target="#dialog"` — идентификатор модального окна, в котором будет показана форма для ввода данных о новом клиенте;
  - `class="btn btn-primary"` — класс для задания цвета кнопки;
  - `<i class="bi bi-bag-plus"></i>` — класс для задания значка для кнопки;
- заголовок таблицы, в которую будет помещен список объектов, вызванных из БД;
- блок загрузки списка объектов с помощью HTMX со следующими атрибутами:
  - `hx-trigger="load, bookListChanged from:body"` — триггер, запускающий загрузку списка клиентов;
  - `hx-get="{% url 'master:client fill' %}"` — отправляет GET-запрос на указанный адрес (URL) для заполнения таблицы со списком клиентов (маршрут для вызова представления загрузки списка клиентов из БД);
  - `hx-target="this"` — ключевое слово, указывающее, что элемент, на котором находится атрибут, является целевым;
- блок для показа модального окна с идентификатором `id="dialog"` (форма для ввода и корректировки данных об объекте);
- блок для показа модального окна с сообщениями о завершении операций ввода и редактирования данных (в центре текущей страницы);
- скрипты HTMX и JavaScript, обеспечивающие отображение и скрытие модальных окон.

На странице шаблона edit_clients.html предусмотрена таблица, в которой отображается список клиентов. Для этой таблицы сформирован статичный заголовок, но тело таблицы — ее строки — формируются динамически. Заполнение таблицы данными из БД обеспечивает следующий фрагмент программного кода:

```
<!-- Начало тела таблицы с загрузкой списка объектов с помощью HTMX -->
<tbody hx-trigger="load, bookListChanged from:body"
       hx-get="{% url 'master:client_fill' %}"
       hx-target="this">
```

```
  <tr>
    <td class="spinner-border" role="status">
      <span class="visually-hidden">Загрузка...</span>
    </td>
  </tr>
</tbody>  <!-- Конец тела таблицы -->
```

Здесь с использованием библиотеки HTMX вызывается URL-маршрут `master:client_fill` (листинг 5.7), вызывающий представление для заполнения таблицы данными из БД.

Листинг 5.7. URL-маршрут client_fill

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `client_fil` (листинг 5.8).

Листинг 5.8. Представление client_fill

Примечание.

В демо-версии книги удалены листинги программ.

Здесь в объект `clients` загружается полный список клиентов из БД, после чего вызывается шаблон client_fill.html (листинг 5.9), в который в качестве контекста передается словарь со списком клиентов.

Листинг 5.9. Шаблон client_fill.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне создан цикл, где в тело таблицы вставляются строки с информацией о каждом объекте, — в нашем случае о клиенте. Таким образом, два шаблона: edit_clients.html и client_fill.html — работают в связке: первый отображает список клиентов в виде таблицы, а второй заполняет таблицу данными из БД.

5.2.3. Модули для добавления клиентов в БД

В шаблоне edit_clients.html имеется кнопка **Добавить клиента**, с которой связана следующая ссылка:

```
hx-get="{% url 'master:add_client' %}"
```

По нажатию на эту кнопку активируется URL-маршрут add_client (листинг 5.10).

Листинг 5.10. URL-маршрут add_client

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `add_client` (листинг 5.11).

Листинг 5.11. Представление add_client

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении на основе класса ClientForm создается объект-форма form, после чего вызывается созданный с использованием элементов библиотеки HTMX шаблон-форма client_form.html (листинг 5.12), в который передается пустая форма form в виде словаря. Обратите внимание, что кроме импорта модели Clients и формы ClientForm здесь импортировано еще три библиотеки: json, HttpResponse и require_POST.

Листинг 5.12. Шаблон-форма client_form.html

Примечание.

В демо-версии книги удалены листинги программ.

Форма этого шаблона имеет атрибут hx-encoding='multipart/form-data', который обеспечивает передачу в представление файлов (фото клиентов), и атрибут class="modal-content" — то есть класс, задающий отображение формы в модальном окне.

Модальная форма содержит три блока:

- заголовок формы;
- тело формы;
- подвал формы.

В заголовке формы присутствует кнопка "Close" для закрытия модального окна. Тело формы представляет собой таблицу, состоящую из одной колонки, в которой отображаются поля модели для ввода данных. В подвале формы присутствуют две кнопки: Выход — для закрытия окна без сохранения данных и Сохранить — для закрытия окна с сохранением данных.

На странице client_fill.html должны также присутствовать две кнопки: Изменить и Удалить, и ссылка ФИО клиента. Эти кнопки и ссылка обеспечивают выполнение следующих действий:

- кнопка Изменить — вызывает модальное окно для редактирования сведений о клиенте;
- кнопка Удалить — вызывает модальное окно для подтверждения удаления сведения о клиенте из БД;
- ссылка в колонке ФИО клиента — показывает модальное окно с полными сведениями о клиенте.

Для выполнения действий по этим кнопкам и ссылке необходимо написать программный код — в противном случае шаблон edit_clients.html не будет работать корректно.

5.2.4. Программируем кнопку *Изменить* — для корректировки в БД сведений о клиентах

В шаблоне client_fill.html с кнопкой Изменить связана следующая ссылка:

```
hx-get="{% url 'master:edit_client' id=client.id %}"
```

По нажатию на эту кнопку активируется URL-маршрут edit_client (листинг 5.13), в который передается идентификатор текущего клиента (id=client.id).

Листинг 5.13. URL-маршрут edit_client

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление edit_client (листинг 5.14).

Листинг 5.14. Представление edit_client

Примечание.

В демо-версии книги удалены листинги программ.

В это представление передается идентификатор клиента (id) и создается объект client, в который из БД загружаются сведения об этом клиенте. После чего на основе класса ClientForm создается объект-форма (forma), в которую через объект client передаются все сведения о клиенте. Оба объекта: forma и client — через словарь передаются в шаблон client_form.html, который уже был использован нами для ввода данных о новом клиенте. После загрузки этого шаблона в нем отображаются все текущие сведения о клиенте.

В этом окне можно скорректировать данные о клиенте — например, изменить его возраст. По нажатию на кнопку Сохранить пользователю будет выдано модальное окно с информацией об успешном завершении корректировки данных.

5.2.5. Программируем кнопку *Удалить* — для удаления из БД сведений о клиентах

В шаблоне client_fill.html с кнопкой Удалить связана следующая ссылка:

```
hx-get="{% url 'master:del_client_yesno' id=client.id %}"
```

По нажатию на эту кнопку активируется URL-маршрут del_client_yesno (листинг 5.15), в который передается идентификатор текущего клиента (id=client.id).

Листинг 5.15. URL-маршрут del_client_yesno

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление del_client_yesno (листинг 5.16).

Листинг 5.16. Представление del_client_yesno

Примечание.
В демо-версии книги удалены листинги программ.

Это представление получает идентификатор клиента (id), и в объект client загружаются из БД сведения о нем. Эти сведения в виде словаря передаются в шаблон del_client_yesno.html (листинг 5.17).

Листинг 5.17. Шаблон del_client_yesno.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне отображаются две кнопки: Отмена — для отмены удаления объекта и Удалить — для подтверждения удаления объекта.

С кнопкой Удалить связана следующая ссылка:

```
hx-post = "{% url 'master:del_client' id=client.id %}"
```

По этой ссылке будет активирован URL-маршрут del_client (листинг 5.18), в который передается идентификатор удаляемого клиента (id=client.id).

Листинг 5.18. URL-маршрут del_client

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту будет активировано представление del_client (листинг 5.19).

Листинг 5.19. Представление del_client

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор удаляемого клиента (`id`) и создает объект `client`, в который из БД загружаются сведения об удаляемом клиенте, после чего объект `client` удаляется из БД с использованием инструкции `client.delete()`. Затем срабатывает триггер библиотеки HTMX, который в модальном окне выдает сообщение об успешном удалении записи из БД. Таким образом, после нажатия на кнопку `Удалить`, сведения о клиенте будут удалены из БД, а пользователю выдано модальное окно с информацией об успешном завершении процедуры удаления данных.

5.2.6. Программируем модули для показа подробных сведений о клиентах

В шаблоне client_fill.html тег показа `ФИО клиента` представлен в виде ссылки:

```
hx-get="{% url 'master:show_client' id=client.id %}"
```

По этой ссылке инициируется URL-маршрут `show_client` (листинг 5.20), в который передается идентификатор клиента (`id=client.id`).

Листинг 5.20. URL-маршрут show_client

Примечание.

В демо-версии книги удалены листинги программ.

Этот маршрут инициирует представление `show_client` (листинг 5.21).

Листинг 5.21. Представление show_client

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор клиента (`id`) и создает объект `client`, в который загружает все сведения о клиенте, после чего на основе класса `ClientForm` создается объект-форма `form`, и в эту форму загружаются сведения о клиенте. Затем формируется словарь `context`, который передается в шаблон-форму show_clien_form.html (листинг 5.22).

Листинг 5.22. Шаблон-форма show_clien_form.html

Примечание.

В демо-версии книги удалены листинги программ.

Здесь с использованием атрибутов библиотеки HTMX создается модальная форма, включающая три блока:

- заготовок формы с кнопкой закрытия модального окна "`Close`";
- тело формы с отображением данных о клиенте;
- подвал формы с кнопкой закрытия модального окна.

Поля с данными о клиенте отображаются внутри тега `<p></p>`, который обеспечивает защиту данных от редактирования.

5.2.7. Испытание программных модулей для редактирования сведений о клиентах в модальных окнах

Итак, мы реализовали набор модулей и шаблонов для ввода и редактирования данных о клиентах. Теперь посмотрим, как они работают.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 5** и перейти по ссылке **Редактор списка клиентов**.

При первоначальной загрузке шаблон edit_clients.html в окне браузера будет выглядеть так, как показано на рис. 5.2.

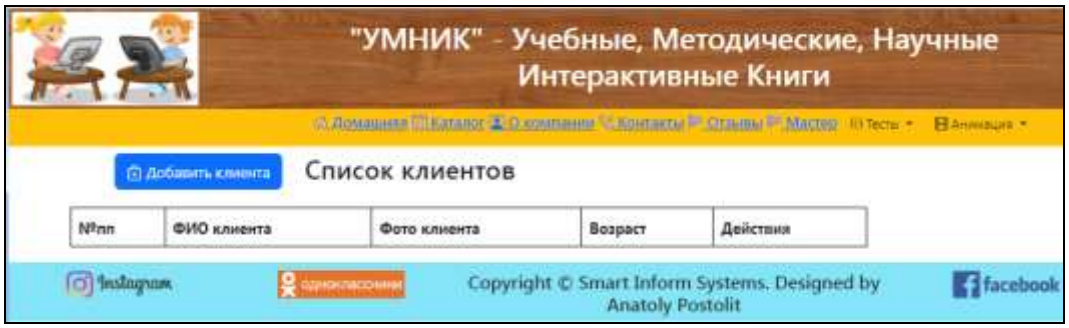


Рис. 5.2. Внешний вид шаблона edit_clients.html при первоначальной загрузке

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

Таблица со списком клиентов пока пустая — так как в БД еще нет ни одной записи о клиентах. В верхней строке окна имеется кнопка **Добавить клиента** — по нажатию на эту кнопку будет показано модальное окно на базе шаблона-формы client_form.html (рис. 5.3), где пользователь может ввести необходимые данные о себе (рис. 5.4). При этом основное окно не исчезло с экрана, а показано с некоторым затемнением.

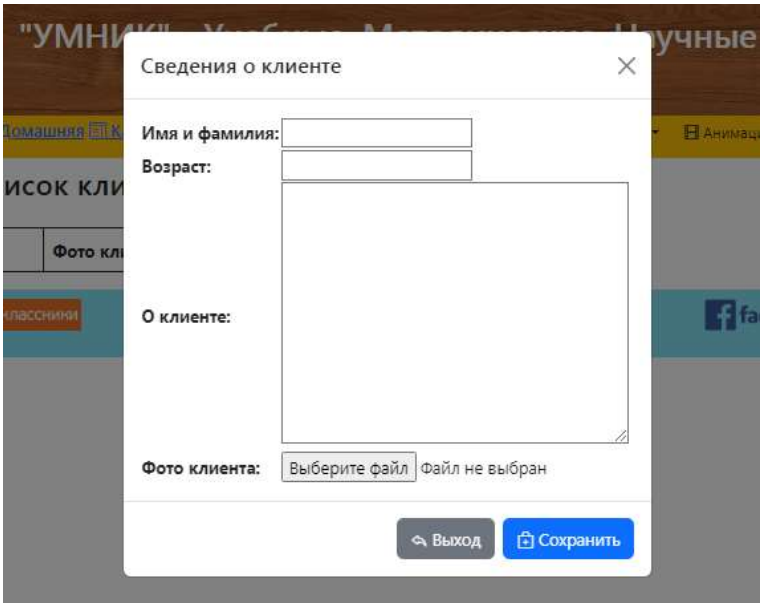


Рис. 5.3. Модальное окно на базе шаблона-формы client_form.html с пустыми полями

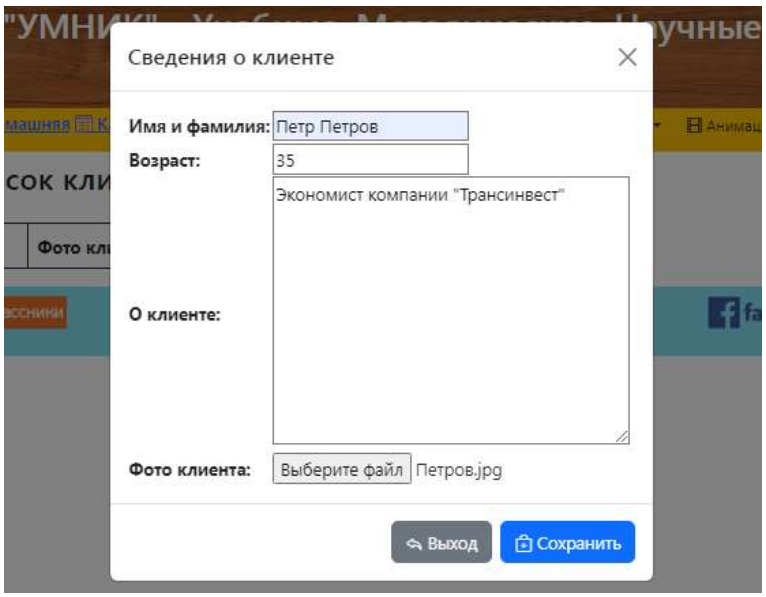


Рис. 5.4. Модальное окно на базе шаблона-формы client_form.html с введенными данными

По нажатию на кнопку **Сохранить** модальное окно закроется, и пользователь в новом модальном окне (рис. 5.5) получит сообщение о том, что он успешно добавлен в БД.



Рис. 5.5. Модальное окно с сообщением об успешном добавлении клиента в БД

Модальное окно с сообщением можно принудительно закрыть, нажав в нем на значок . Впрочем, если никаких действий не предпринимать, то через 1–2 секунды это окно само закроется, и введенные данные будут показаны в таблице шаблона edit_clients.html (рис. 5.6).



Рис. 5.6. Внешний вид шаблона edit_clients.html с введенными данными о клиенте

Если теперь в шаблоне edit_clients.html нажать на кнопку **Изменить**, то откроется модальное окно на базе шаблона-формы client_form.html с полными сведениями о клиенте, доступными для редактирования (рис. 5.7).

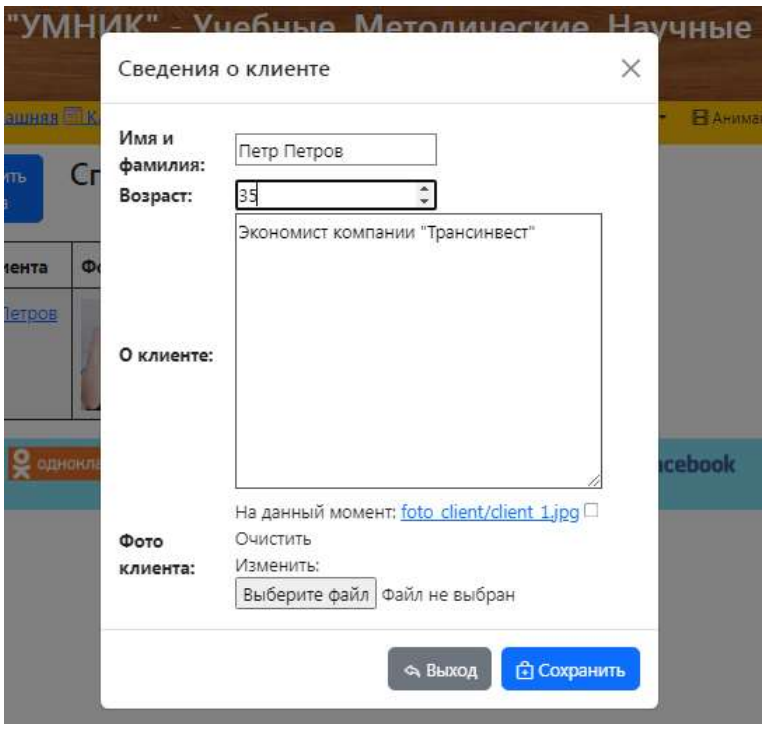


Рис. 5.7. Модальное окно на базе шаблона-формы client_form.html с редактируемыми данными

Изменим в этом шаблоне возраст клиента (заменим 35 на 37) и нажмем кнопку **Сохранить** — появится модальное окно с сообщением об успешном изменении данных о клиенте (рис. 5.8).

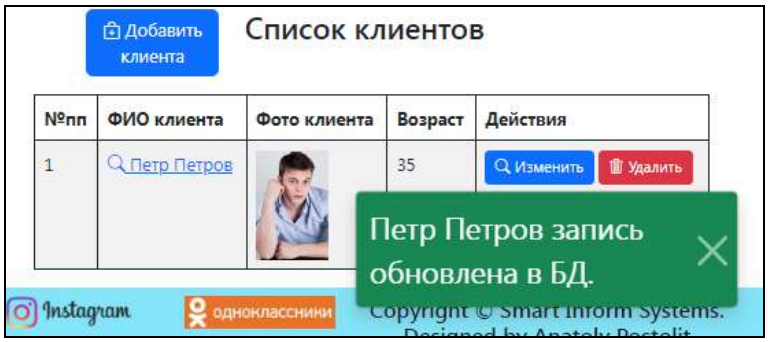


Рис. 5.8. Модальное окно на базе шаблона edit_clients.html с сообщением об обновлении данных

Когда всплывающее окно с сообщением закроется, мы увидим, что в таблице данные о клиенте действительно обновились — его возраст теперь 37 лет (рис. 5.9).

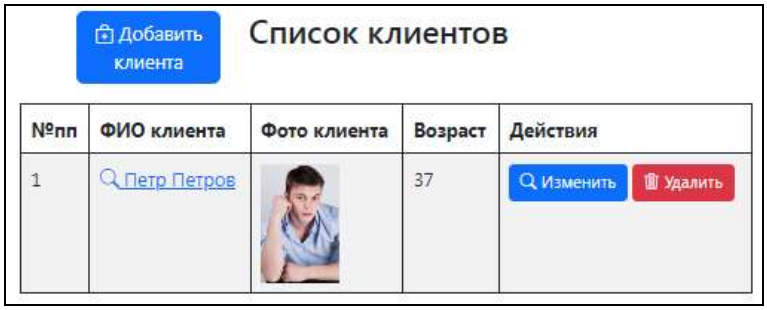


Рис. 5.9. Внешний вид шаблона edit_clients.html после обновления данных

Дополним список клиентов еще несколькими записями (рис. 5.10).

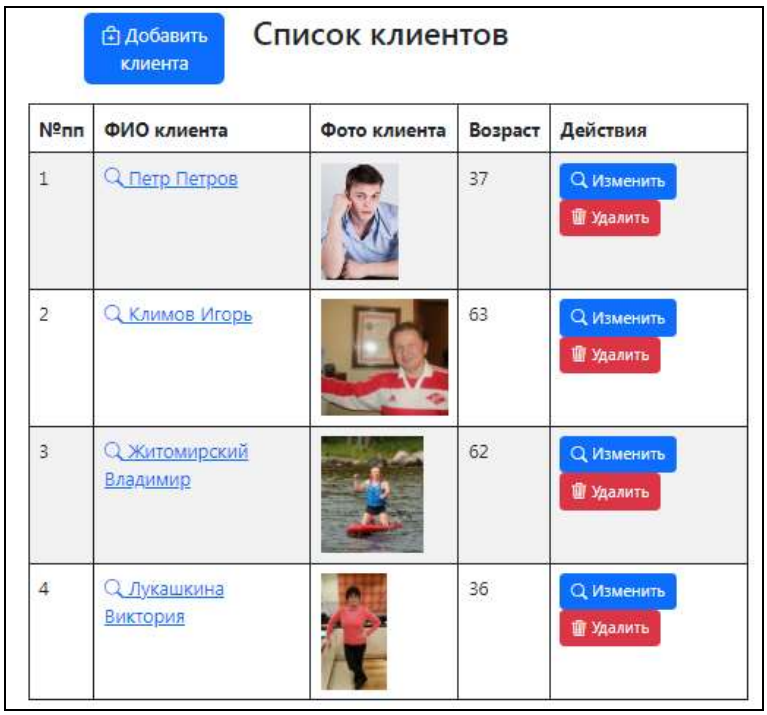


Рис. 5.10. Внешний вид шаблона edit_clients.html после добавления нескольких записей

Как здесь можно видеть, в столбце **ФИО клиента** фамилия и имя клиента представлены в виде ссылки, и если щелкнуть левой кнопкой мыши на этой ссылке, то откроется модальное окно на базе шаблона-формы show_clien_form.html (рис. 5.11).

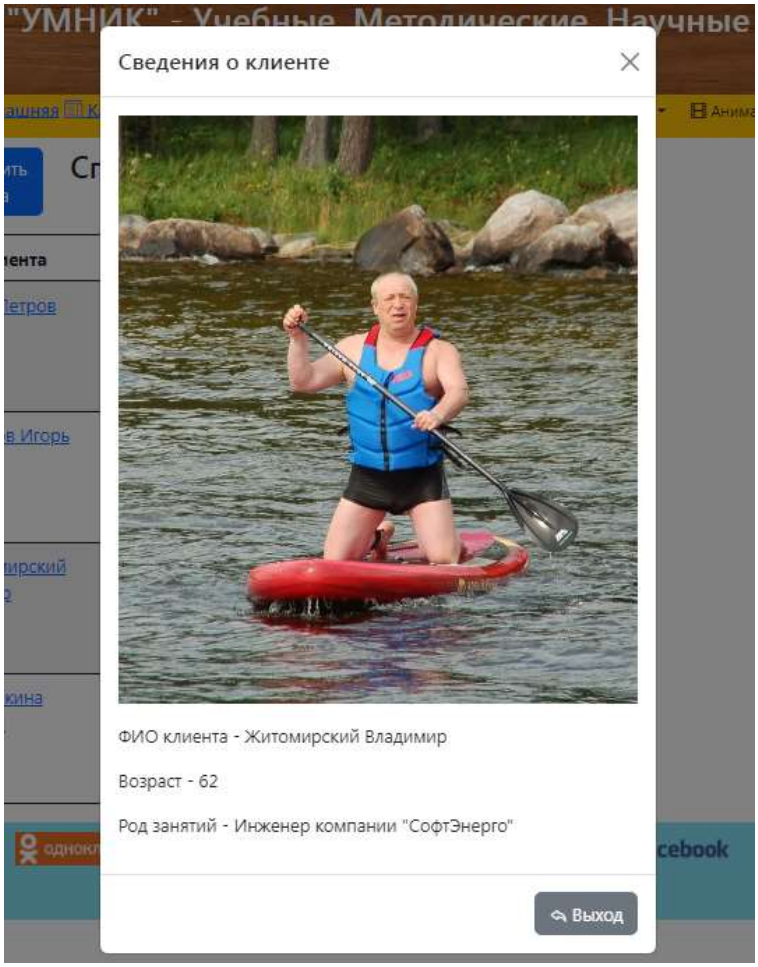


Рис. 5.11. Модальное окно на базе шаблона-формы show_clien_form.html с полными данными о клиенте

Если в шаблоне edit_clients.html (см. рис. 5.10) нажать на кнопку **Удалить**, то откроется модальное окно на основе шаблона del_client_yesno.html с просьбой о подтверждении удаления сведений о клиенте (рис. 5.12).

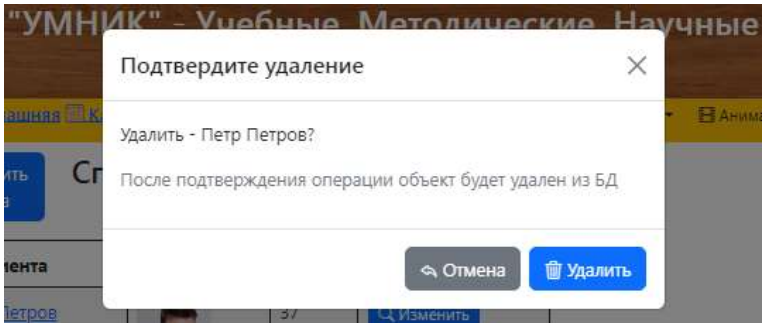


Рис. 5.12. Модальное окно на базе шаблона формы del_client_yesno.html с просьбой о подтверждении удаления записи

Если тепрь в этом окне нажать на кнопку **Удалить**, то сведения о клиенте будут из БД удалены, и пользователю выдано модальное окно с информацией об успешном завершении удаления данных (рис. 5.13).

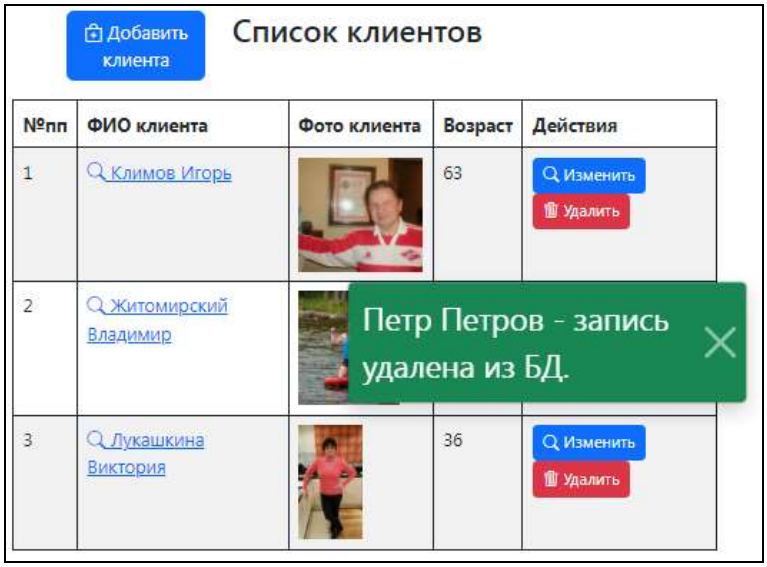


Рис. 5.13. Модальное окно с сообщением об успешном удалении записи из БД

Когда модальное окно с сообщением об удалении записи закроется, в списке клиентов строки с информацией об этом клиенте больше не будет (рис. 5.14).

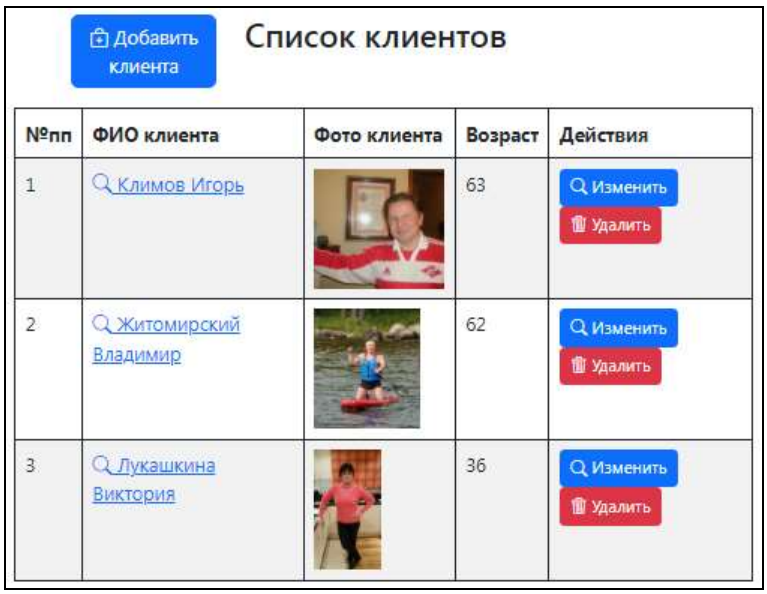


Рис. 5.14. Список клиентов после удаления записи о клиенте **Петр Петров**

5.3. Краткие итоги

В этой главе приведены сведения о модальных окнах. Вы можете скачать коды представленных здесь примеров и использовать их в своих проектах, где пользователи взаимодействуют с удаленной базой данных (новостные сайты, блоги, интернет-магазины и т. п.).

Говоря о сайтах для магазинов интернет-торговли, надо отметить, что это на сегодня является, пожалуй, самой распространенными задачей, которую приходится решать веб-разработчикам. В Интернете вы можете найти множество примеров кода для онлайн-магазинов, но они, как правило, представляют собой лишь отдельные решения для некоторых не связанных между собой фрагментов (создание БД для хранения сведений о товарах, создание корзины и заказов, оплата заказа пластиковыми картами). Исходя из этого, в следующей главе приведены примеры связанных модулей — от создания базы данных с продаваемыми товарами до оплаты заказ пластиковыми картами.

Глава 6. Интернет-магазин

Для конечного пользователя важно иметь такой интернет-магазин, в котором он сам бы смог заниматься его администрированием: изменять структуру справочника с категориями продуктов, корректировать список товаров, менять фотографии, цены, остатки на складе и т. п. — вплоть до изменения интерфейса страниц сайта. И выполнять все эти действия через интерфейс личного кабинета сайта, не затрагивая программный код. Именно такой магазин вы можете создать для своих клиентов, используя материалы предыдущих глав и этой, из материалов станет понятно:

- как выполняется обмен данными между представлениями и шаблонами с использованием переменных сессии;
- как выполняется обмен данными между представлениями и шаблонами с использованием контекстных процессоров (без них не будут работать модули формирования корзины и заказов);
- как можно в одной модели хранить информацию корзины, предварительного заказа и покупок клиентов магазина;
- как создать индивидуальный магазин для каждого клиента сайта электронной торговли;
- как отобразить категории продуктов в боковой панели страницы в виде свернутого и раскрытого дерева;
- как отобразить категории продуктов в центральной панели страницы в виде мегаменю;
- как создать программные модули для формирования корзины (предварительного заказа);
- как создать программные модули для перевода корзины в предварительный заказ;
- как можно интегрировать интернет-магазин с платежной платформой Robokassa;
- как можно отладить взаимодействие сайта с ответами платежной платформы на успех или ошибку оплаты заказа;
- как конечному пользователю показать информацию об оплаченных заказах и приобретенных товарах.

На первый взгляд — это простой набор не связанных модулей. Однако всё здесь не так просто, как кажется, указанные опции — это фрагменты программных кодов реально работающих в Интернете магазинов. Фактически, они представляют собой кирпичики, которые стыкуются между собой, и из них можно собирать полноценные сайты электронной коммерции.

6.1. Обмен данными между представлениями и шаблонами

В Django по умолчанию выборка данных из БД выполняется в представлениях, а затем эти данные передаются в шаблон через контекстный словарь. Например:

```
def client(request):
    clients = Clients.objects.all()
    context = {'clients': clients}
    return render(request, 'client_show.html', context)
```

Здесь в объект `clients` из БД загружается список всех клиентов, после чего список клиентов загружается в словарь `context`, который передается в шаблон с именем client_show.html, где эти данные и выводятся пользователю. Однако достаточно часто возникают ситуации, когда данные, сформированные в одном представлении, нужно получить в других представлениях. Или когда данные, полученные в представлении, надо сделать доступными во всех шаблонах приложения. Рассмотрим, какими способами в Django реализован такой обмен данными.

6.1.1. Обмен данными через сессии

Все взаимодействия между браузерами клиентов и публичными серверами осуществляются при помощи протокола HTTP, который не сохраняет свое состояние. Этот факт означает,

что сообщения между клиентом и сервером являются полностью независимыми один от другого, — то есть не существует какого-либо представления «последовательности» или поведения в зависимости от предыдущих сообщений. В результате, если вы хотите создать сайт, который будет отслеживать взаимодействие с клиентом (браузером), вам нужно реализовать это самостоятельно.

Механизмом, который использует Django для отслеживания «состояния» между сайтом и каким-либо конкретным браузером пользователя, являются *сессии*. Они позволяют хранить произвольные данные браузера и получать их в тот момент, когда между этим браузером и сайтом устанавливается соединение. Данные получаются и сохраняются в сессии при помощи соответствующего «ключа».

Django использует куки (cookie), содержащие специальный идентификатор сессии, который выделяет среди остальных каждый браузер и соответствующую сессию. Реальные данные сессии по умолчанию хранятся в базе данных сайта (это более безопасно, чем сохранять данные в куки, где они могут быть уязвимы для злоумышленников). Однако у вас есть возможность настроить Django так, чтобы сохранять данные сессий в других местах (кеше, файлах, «безопасных» куки). Но все же хранение по умолчанию является хорошей и безопасной возможностью.

Сессии становятся доступны в модулях приложения автоматически в тот момент, когда было создано приложение. В разделах `INSTALLED_APPS` и `MIDDLEWARE` конфигурационного файла проекта settings.py имеется следующий код:

```
INSTALLED_APPS = [
    ...
    'django.contrib.sessions',
    ....

MIDDLEWARE = [
    ...
    'django.contrib.sessions.middleware.SessionMiddleware',
    ....
```

Доступ к переменной сессии (`session`) в любом представлении можно получить через параметр `request` (`HttpRequest` передается как первый аргумент в каждое представление). Переменная сессии всегда связана с определенным пользователем (или, если быть более точным, с определенным браузером).

Переменная сессии (`session`) является объектом-словарем, который служит для чтения и записи неограниченное число раз. С ним вы можете выполнять любые стандартные операции, включая очистку всех данных, проверку наличия ключа, циклы по данным и пр. Далее представлены фрагменты кода, которые показывают, как задавать, получать и удалять данные при помощи ключа, связанного с текущей сессией (с текущим браузером):

```
# Запомнить категорию товара в переменной сессии
request.session['id_cat'] = id_cat
# Получить категорию товара из переменной сессии
id_cat = request.session['id_cat']
# Очистить переменную сессии
del request.session['id_cat']
```

6.1.2. Обмен данными через контекстный процессор

Теперь давайте рассмотрим способ доступа к некоторому блоку данных во всех шаблонах приложения с использованием *контекстных процессоров*.

> ***Контекстные процессоры***
> Контекстные процессоры — это функции, которые позволяют добавлять определенные данные в контекст каждого шаблона. Они используются, когда нужно сделать некоторые данные доступными на всех страницах сайта.

Для создания контекстных процессоров нужно выполнить следующие действия:

- в каталоге приложения создать файл context_processors.py;

- написать простую функцию, которая принимает только `request`;
- в теле функции получить нужный набор данных и вернуть их в виде словаря;
- добавить созданный контекстный процессор в файл settings.py.

Как правило, контекстные процессоры создаются для хранения данных, которые необходимо использовать на разных страницах сайта. Этими данными могут быть: сведения о зарегистрированных пользователях сайта, категории товаров, корзина с предварительными заказами товаров или сформированный заказ клиента в интернет-магазине и т. п. Более детально создание контекстных процессоров будет далее показано на примере создания модулей электронной торговли интернет-магазина.

6.2. Модели для корзины и заказов клиентов магазина электронной торговли

В магазине электронной торговли каждый пользователь должен иметь личный кабинет с такими индивидуальными данными, как номер телефона, адрес электронной почты, почтовый адрес и т. п. Адрес электронной почты нужен для того, чтобы, например, подтвердить клиенту почтовым сообщением факт оплаты товара банковской картой, почтовый адрес — для доставки товара, номер телефона — для связи с курьером. Кроме того, покупки в магазине электронной торговли одновременно делают множество пользователей, и каждый из них должен иметь свою корзину и свои заказы. Исходя из этого, каждый клиент магазина должен иметь личный кабинет, который, как правило, создается при регистрации. В этой книге мы не рассматриваем вопросы регистрации и идентификации пользователей — как опытный программист, вы с ними должны быть уже знакомы. Так что будем условно считать, что в системе есть один зарегистрированный пользователь с известным идентификатором (это может быть id=1, или id=2, или id=3 и т. п.).

В предыдущих главах книги были созданы модели для хранения категорий продуктов (`Genre_m`) и самих продуктов (`Product`) — их мы и используем в этой главе. Для магазина электронной торговли нужно иметь еще несколько моделей: для корзины, для заказов клиентов, для оплаченных покупок. Эти модели можно объединить в одну модель — тогда данные не понадобится переносить из одной таблицы в другую, достаточно будет изменять признак отнесения данных к корзине, к заказу или к покупке. Соответствующий код модели для хранения корзины (заказа, покупки) `Product_Order` приведен в листинге 6.1.

Листинг 6.1. Программный код модели Product_Order для корзины (заказа, покупки)

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно упрощенная модель — она имеет минимальный набор следующих полей:

- `user` — пользователь (владелец личного кабинета магазина);
- `num_order` — номер заказа;
- `status` — статус заказа;
- `amount` — стоимость заказа;
- `creation_date` — дата создания заказа;
- `date_pay` — дата оплаты заказа.

В этой модели поле `user` ссылается на первичный ключ модели `User` (зарегистрированный пользователь приложения). Полю `status` по умолчанию будет присвоено значение `cart` — корзина, то есть с этим статусом модель будет представлять корзину, в которой пользователь может формировать предварительный заказ. Когда корзина будет сформирована, ее можно будет перевести в статус «к оплате» (поле получит значение `for_payment`). При такой смене статуса корзина исчезнет и сформируется предварительный заказ. После оплаты заказа предварительный заказ можно перевести в статус оплаченного заказа — для этого полю `status` надо будет присвоить значение `paid` (оплачен). Таким образом, для манипулирования корзиной, заказами и покупками в этой модели потребуются изменения значения всего одного поля — `status`. То есть оно может принимать всего три значения: `cart` — корзина, `for_payment` — предварительный заказ, `paid` — оплаченный заказ (покупка).

Модель `Product_Order` содержит только общие сведения о заказе — в ней нет данных о деталях заказа, то есть о самих продуктах. Данные о продуктах, которые входят в заказ, будут содержаться в другой связанной модели — `Product_OrderItem` (листинг 6.2).

Листинг 6.2. Программный код модели Product_OrderItem для продуктов в корзине (в заказе)

Примечание.
В демо-версии книги удалены листинги программ.

Эта модель также имеет минимальный набор следующих полей:

- `order` — номер заказа (связь с моделью `Products_Order` по первичному ключу);
- `product` — продукт (связь с моделью `Product` по первичному ключу);
- `quantity` — количество продуктов (товара) в заказе;
- `price` — цена продуктов (товара) в заказе;
- `user_id` — идентификатор пользователя магазина (он понадобится при формировании корзины и заказа).

Создав указанные модели, не забудьте выполнить процедуру миграции:

```
python manage.py makemigrations
python manage.py migrate
```

6.3. Контекстный процессор для магазина электронной торговли

В типовом магазине электронной торговли необходимо создать контекстные процессоры для следующих типов данных:

- категории продуктов (товаров);
- количество продуктов (товаров) в БД;
- количество продуктов (товаров), положенных пользователем в корзину;
- стоимость продуктов (товаров), находящихся в предварительном заказе;
- стоимость продуктов (товаров) в оплаченном заказе.

Создадим в нашем тестовом приложении `master` файл с именем context_processor.py и напишем в нем следующий код (листинг 6.3).

Листинг 6.3. Контекстный процессор для магазина электронной торговли

Примечание.
В демо-версии книги удалены листинги программ.

В этом контекстном процессоре выполнен импорт необходимых пакетов и созданы следующие функции:

- `def genre_m` — полный список категорий продуктов (товаров) в БД;
- `def num_products_m` — количество продуктов (товаров), находящихся в БД;
- `def cart_kol_m` — количество продуктов (товаров), которые покупатель положил в корзину;
- `def cart_sum_zak_m` — общая стоимость продуктов (товаров) в предварительном заказе;
- `sum_buy_order_m` — общая стоимость продуктов (товаров) в оплаченном заказе.

Каждая из этих функций возвращает контекстный словарь с данными (`context`), которые будут доступны во всех шаблонах проекта (в нашем случае — это проект `master`). Но для этого все функции контекстного процессора необходимого зарегистрировать в разделе `TEMPLATES` файла settings.py проекта Django (листинг 6.4).

Листинг 6.4. Регистрация контекстных процессоров в файле settings.py

Примечание.
В демо-версии книги удалены листинги программ.

Как здесь можно видеть, строка регистрации контекстного процессора состоит из имени приложения, ключевого слова `context_processors` и имени функции, которая определена в файле context_processor.py.

Внимание!
Несмотря на то, что в файле *settings.py* каждая функция контекстного процессора имеет префикс в виде имени приложения, имена функций и возвращаемых ими параметров должны быть уникальными для всего проекта. Если в контекстном процессоре разные приложения будут иметь функции или возвращаемые функциями значения с одинаковыми идентификаторами, то возникнет коллизия, и программа будет работать с ошибками.

6.4. Создаем магазин пользователя

Напишем программный код модулей, которые обеспечивают создание индивидуального магазина пользователя. Ссылка для вызова магазина пользователя, скорее всего, должна находиться в его личном кабинете, — ее код приведен в листинге 6.5.

Листинг 6.5. Ссылка для вызова личного магазина пользователя

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке для показа магазина будет вызван URL-маршрут `user_shop_master` (листинг 6.6).

Листинг 6.6. URL-маршрут user_shop_master

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `user_shop_master` (листинг 6.7).

Листинг 6.7. Представление user_shop_master

Примечание.
В демо-версии книги удалены листинги программ.

Это достаточно простое представление, где в объект `user` загружаются данные о пользователе, — они затем заносятся в контекстный словарь `context`, который передается в шаблон user_shop_master.html. В этом тестовом примере в объект `user` загружается текстовая константа, а в реальном приложении нужно будет снять комментарий в строке `user = request.user`.

Пояснение
Этот код работает в рамках интерактивной цифровой книги. В реальном проекте, как уже было сказано ранее, надо снять комментарий со строки `# user = request.user` и удалить или закомментировать строку `user = User.objects.get(id=2)`.

Код шаблона user_shop_master.html приведен в листинге 6.8.

Листинг 6.8. Шаблон user_shop_master.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне создана кнопка `Выбор типа каталога продуктов` с выпадающим списком опций:

- `левая панель (свернута)` — показать дерево категорий продуктов в левой боковой панели в виде свернутого дерева;
- `левая панель (раскрыта)` — показать дерево категорий продуктов в левой боковой панели в виде раскрытого дерева;
- `верхняя панель` — показать дерево категорий продуктов в верхней части центральной панели в виде мегаменю.

Обратите внимание на переменную шаблона, в которой выводится информация о количестве доступных товаров: `{{num_products_m}}`. В представлении `user_shop` эта переменная не получает никаких значений и не передается в шаблон. Где же задается значение для этой переменной? Она получает значение в контекстном процессоре, и это значение может быть показано в любом шаблоне приложения. По аналогии переменные шаблона: `{{cart_kol_m}}`, `id_cart_m`, `{{sum_cart_m}}`, `id_zak_m` и `{{sum_my_buy_m}}` — также получают значения из контекстного процессора.

Внимание!

Несмотря на то, что в файле *settings.py* каждая функция контекстного процессора имеет префикс в виде имени приложения, имена функций и возвращаемых ими параметров должны быть уникальными для всего проекта. Если в контекстном процессоре разные приложения будут иметь функции или возвращаемые функциями значения с одинаковыми идентификаторами, то возникнет коллизия, и программа будет работать с ошибками.

Теперь рассмотрим программные модули, которые обеспечивают отображение категорий продуктов в различных видах и в различных частях шаблона user_shop_master.html.

6.4.1. Вывод категорий продуктов в левой боковой панели магазина (дерево свернуто)

В шаблоне user_shop_master.html при выборе типа каталога продуктов `Левая панель (свернута)` активируется следующая ссылка:

```
<a class="dropdown-item" href="master:user_catalog_tree">Левая панель
(свернута)</a>
```

По этой ссылке активируется URL-маршрут `user_catalog_tree` (листинг 6.9).

Листинг 6.9. URL-маршрут user_catalog_tree

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `user_catalog_tree` (листинг 6.10).

Листинг 6.10. Представление user_catalog_tree

Примечание.

В демо-версии книги удалены листинги программ.

Здесь переменная `name_cat` получает значение `Все продукты`. Затем из БД подгружается объект `products` — все продукты из БД, и создается пагинатор для отображения 30 карточек продуктов на странице. Всё это через объект `context` передается в шаблон user_catalog_tree.html.

Пояснение

Этот код работает в рамках интерактивной цифровой книги. В реальном проекте надо снять комментарий со строки `# user = request.user` и удалить или закомментировать строку `user = User.objects.get(id=2)`.

Код шаблона user_catalog_tree.html приведен в листинге 6.11.

Листинг 6.11. Шаблон user_catalog_tree.html

Примечание.

В демо-версии книги удалены листинги программ.

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне созданы две панели: левая боковая панель — для отображения свернутого дерева категорий продуктов и центральная панель — для отображения карточек с информацией о продуктах. Чтобы можно было развернуть (свернуть) дерево категорий продуктов в шаблон добавлены стили и Java-скрипты.

В центральной панели присутствует блок показа карточек с продуктами, в который вставлено подключение к шаблону user_product.html. Код этого шаблона приведен в листинге 6.12.

Листинг 6.12. Шаблон user_product.html

Примечание.

В демо-версии книги удалены листинги программ.

Шаблон user_product.html начинается с тега `{% if messages %}`, в котором выводится сообщение о том, что выбранный пользователем товар уже добавлен в корзину. Тег будет задействован в модуле формирования корзины.

В шаблоне также выводится подсказка о категории, выбранной в дереве категорий продуктов: `{{name_cat}}`. А далее организован цикл, в котором выводятся карточки с информацией о продуктах.

Следует отметить две особенности этих карточек. Первая — к фотографиям продуктов, которые отображаются в карточках, добавлен стиль. Этот стиль обеспечивает приведение фотографий во всех карточках к одному размеру, вне зависимости от размера исходного фото. Вторая особенность — кнопка с надписью `В корзину` сопровождается вращающимся спином (код традиционной кнопки закомментирован). Для запуска анимации вращающегося спина в шаблоне предусмотрен Java-скрипт. Когда пользователь нажимает на эту кнопку, то надпись на кнопке меняется на вращающийся спин, который отображается до тех пор, пока не завершится процедура добавления продукта в корзину.

В первой строке левой боковой панели шаблона user_catalog_tree.html (см. листинг 6.11) имеется кнопка `Все продукты`. С этой кнопкой связана следующая ссылка:

```
<a href="{% url 'master:user_catalog_tree' %}" class="btn btn-success btn-sm">
                <i class="bi bi-cart4"></i> Все продукты</a>
```

По этой ссылке в конечном итоге будет вызвано представление `user_catalog_tree`, в котором будут сняты фильтры на категории продуктов, и в центральной панели отобразятся все карточки продуктов из БД.

В левой боковой панели шаблона user_catalog_tree.html с наименованием категории продукта связана следующая ссылка:

```
<a href="{% url 'master:user_product_tree' id=node.id %}">{{ node.name }}</a>
```

По этой ссылке активируется URL-маршрут `user_product_tree` (листинг 6.13), в который передается идентификатор выбранной категории (`id=node.id`).

Листинг 6.13. URL-маршрут user_product_tree

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `user_product_tree` (листинг 6.14).

Листинг 6.14. Представление user_product_tree

Примечание.
В демо-версии книги удалены листинги программ.

Это представление получает выбранную категорию продукта, и в объект `products` загружаются все продукты выбранной категории и всех дочерних категорий (если они есть). Иными словами, на базу данных с продуктами накладывается фильтр по выбранной категории продукта из древовидного каталога. Затем все полученные параметры загружаются в словарь `context`, который передается в шаблон user_catalog_tree.html (см. листинг 6.11), где и отображаются карточки продуктов с учетом фильтра.

Пояснение
Этот код работает в рамках интерактивной цифровой книги. В реальном проекте надо снять комментарий со строки `# user = request.user` и удалить или закомментировать строку `user = User.objects.get(id=2)`.

Итак, к этому моменту создан набор модулей, обеспечивающих вывод страницы магазина пользователя с выборкой и фильтрацией продуктов по их категориям. Теперь можно проверить работу этих модулей.

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 6** и перейти по ссылке **Мой магазин**.

6.4.2. Проверка работы модулей магазина пользователя с фильтрацией данных по свернутому дереву категорий продуктов

При первоначальной загрузке шаблона user_shop_master.html (магазин пользователя) в окне браузера он будет выглядеть так, как показано на рис. 6.1.



Рис. 6.1. Шаблон user_shop_master.html (магазин пользователя) при первоначальной загрузке

Примечание
В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

Как здесь можно видеть, левая боковая панель пока пустая (ее, собственно, еще вовсе нет), а в центральной панели показаны некоторые элементы персонального магазина пользователя:

- кнопка **Выбор типа каталога продуктов**;
- имя владельца персонального магазина;
- количество доступных товаров в БД;
- ссылка на корзину покупателя;
- ссылка на активный заказ покупателя (не оплаченный);
- ссылка на покупки клиента (оплаченные заказы);

- баннер платежной системы, с которой работает магазин пользователя.

Ссылки на корзину, заказ и покупки мы пока не рассматриваем (еще не описаны соответствующие программные модули), а если нажать на кнопку **Выбор типа каталога продуктов**, появятся следующие опции выпадающего меню (рис. 6.2).

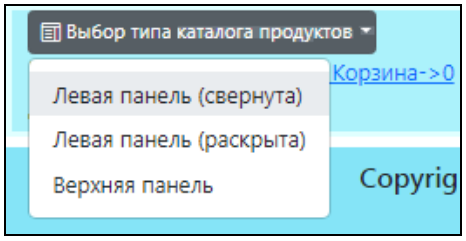


Рис. 6.2. Опции выбора типа каталога продуктов

Если выбрать здесь опцию **Левая панель (свернута)** (показать каталог категорий продуктов в виде свернутого дерева), откроется шаблон user_catalog_tree.html (рис. 6.3).

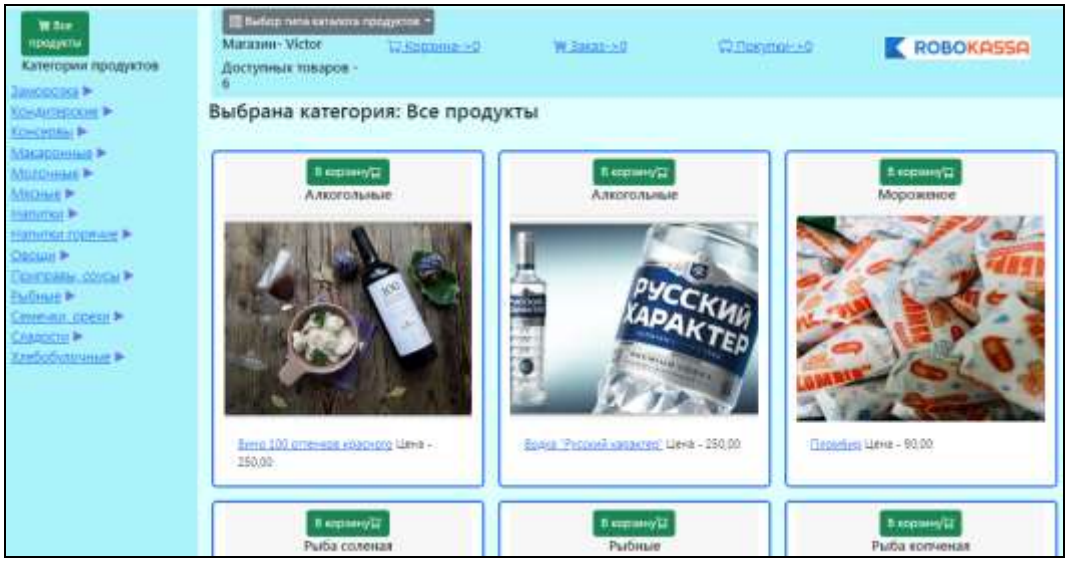


Рис. 6.3. Шаблон user_catalog_tree.html при первоначальной загрузке

Как здесь можно видеть, в левой боковой панели появились категории продуктов в виде свернутого дерева, а в центральной панели — карточки с информацией о продуктах. При первоначальной загрузке этого шаблона на него будут выведены карточки всех продуктов из БД (с учетом постраничной пагинации). Если же в каталоге категорий продуктов выбрать одну из категорий — например, **Рыбные**, то содержимое центральной панели изменится, и в ней будут показаны продукты выбранной категории, включая дочерние (рис. 6.4).

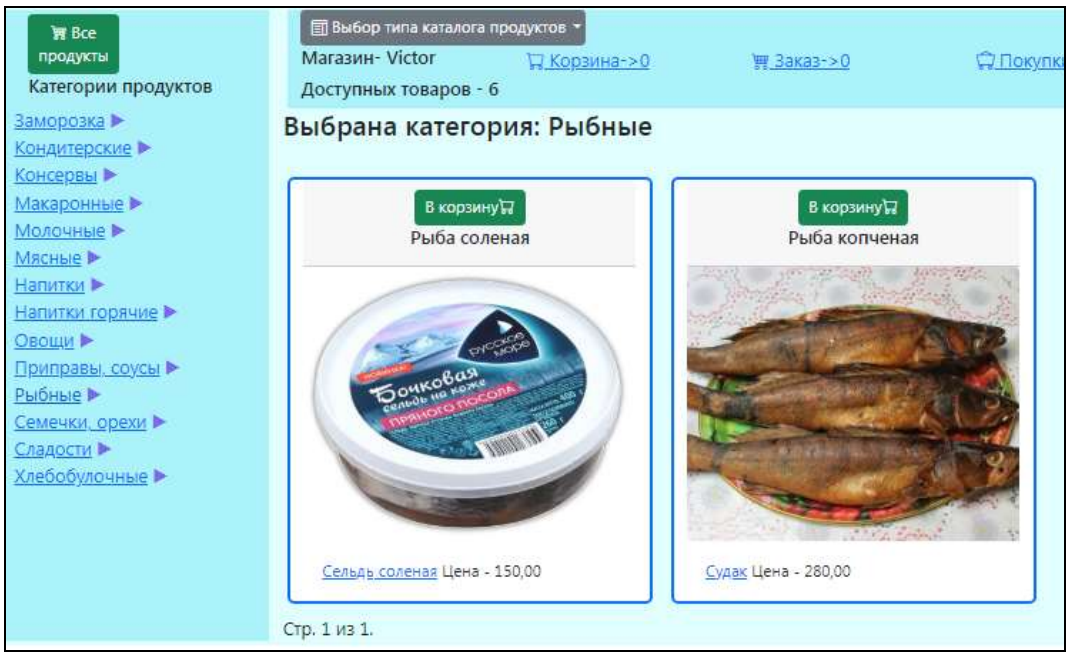


Рис. 6.4. Шаблон user_catalog_tree.html при выборе категории продуктов **Рыбные**

Если в каталоге категорий продуктов нажать на треугольную кнопку ▶ справа от имени категории, текущая ветка дерева раскроется, и будут показаны дочерние категории (рис. 6.5).

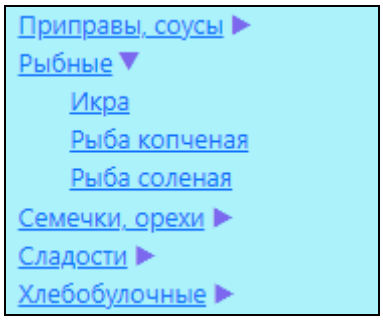


Рис. 6.5. Категория продуктов **Рыбные** с раскрытыми ветками дочерних категорий

Если теперь в дочерних категориях выбрать конечную ветку, — например, **Рыба копченая**, в центральной панели останутся карточки с продуктами только этой категории (рис. 6.6).

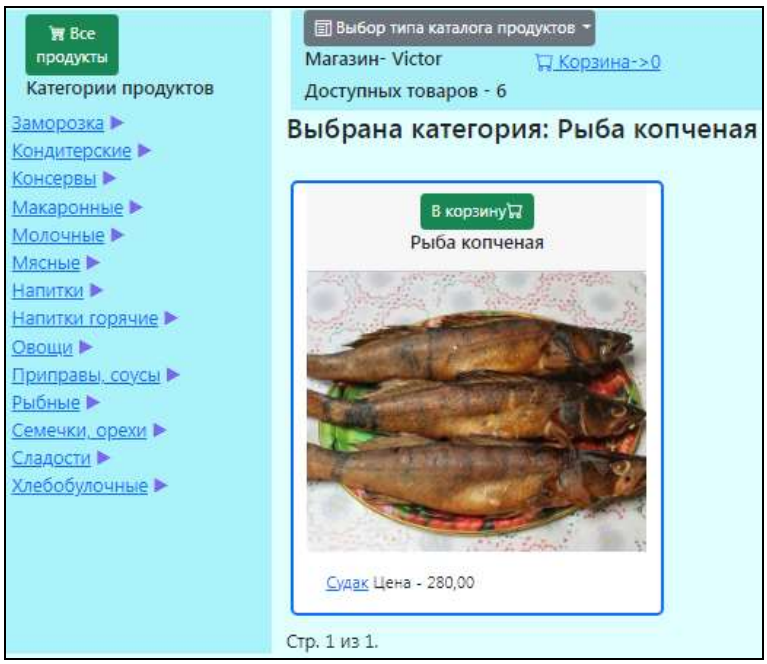


Рис. 6.6. Шаблон user_catalog_tree.html при выборе категории продуктов **Рыба копченая**

Обратите внимание, что наименование продукта представляет собой ссылку, при переходе по которой откроется страница с полноразмерной фотографией и полной информацией об этом продукте (рис. 6.7).



Рис. 6.7. Страница с подробными сведениями о продукте

На приведенных здесь рисунках в карточках продуктов имеется кнопка **В корзину** — она предназначена для того, чтобы пользователь положил этот товар в корзину (создал предварительный заказ). Пока эта кнопка не рабочая — для ее функционирования необходимо создать соответствующий программный код, что и будет сделано в одном из следующих разделов. А сейчас создадим набор модулей, которые позволяют отобразить в левой боковой панели дерево категорий продуктов в раскрытом виде.

6.4.3. Вывод категорий продуктов в левой боковой панели магазина (дерево раскрыто)

В шаблоне user_shop_master.html при выборе типа каталога продуктов **Левая панель (раскрыта)** активируется следующая ссылка:

```
<a class="dropdown-item" href="/user_catalog_stat/">Левая панель (раскрыта)</a>
```

По этой ссылке активируется URL-маршрут `user_catalog_stat` (листинг 6.15).

Листинг 6.15. URL-маршрут user_catalog_stat

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `user_catalog_stat` (листинг 6.16).

Листинг 6.16. Представление user_catalog_stat

Примечание.

В демо-версии книги удалены листинги программ.

Здесь переменная `name_cat` получает значение `Все продукты`. Затем из БД подгружается объект `products` — все продукты из БД, и создается пагинатор для отображения 30 карточек продуктов на странице. Всё это через объект `context` передается в шаблон user_catalog_stat.html.

Пояснение

Этот код работает в рамках интерактивной цифровой книги. В реальном проекте надо снять комментарий со строки `# user = request.user` и удалить или закомментировать строку `user = User.objects.get(id=2)`.

Код шаблона user_catalog_stat.html приведен в листинге 6.17.

Листинг 6.17. Шаблон user_catalog_stat.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне созданы две панели: левая боковая панель — для отображения раскрытого дерева категорий продуктов и центральная панель — для отображения карточек с информацией о продуктах.

В центральной панели также присутствует блок для показа карточек с продуктами, в который вставлено подключение к шаблону user_product.html (см. листинг 6.12).

В первой строке левой боковой панели шаблона user_catalog_stat.html имеется кнопка `Все продукты`. С этой кнопкой связана следующая ссылка:

```
<a href="{% url 'master:user_catalog_stat' %}" class="btn btn-success btn-sm">
                <i class="bi bi-cart4"></i> Все продукты</a>
```

По этой ссылке в конечном итоге будет вызвано представление `user_catalog_stat`, в котором будут сняты фильтры на категории продуктов, и в центральной панели отобразятся все карточки продуктов из БД.

В левой боковой панели шаблона `user_catalog_stat.html` с наименованием категории продукта связана следующая ссылка:

```
<a href="{% url 'master:user_product_stat' id=node.id %}">{{ node.name }}</a>
```

По этой ссылке активируется URL-маршрут `user_product_stat` (листинг 6.18), в который передается идентификатор выбранной категории (`id=node.id`).

Листинг 6.18. URL-маршрут user_product_stat

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `user_product_stat` (листинг 6.19).

Листинг 6.19. Представление user_product_stat

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает выбранную категорию продукта, и в объект `products` загружаются все продукты выбранной категории и всех дочерних категорий (если они есть). Иными словами, на базу данных с продуктами накладывается фильтр по выбранной категории продукта из древовидного каталога. Затем все полученные параметры загружаются в словарь `context`, который передается в шаблон user_catalog_stat.html (см. листинг 6.17), где и отображаются карточки продуктов с учетом фильтра.

Пояснение

Этот код работает в рамках интерактивной цифровой книги. В реальном проекте надо снять комментарий со строки `# user = request.user` и удалить или закомментировать строку `user = User.objects.get(id=2)`.

Итак, к этому моменту создан набор модулей, которые обеспечивают вывод страницы магазина пользователя с выборкой и фильтрацией продуктов по их категориям. Теперь можно проверить работу этих модулей.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 6** и перейти по ссылке **Мой магазин**.

6.4.4. Проверка работы модулей магазина пользователя с фильтрацией данных по раскрытому дереву категорий продуктов

При первоначальной загрузке шаблона user_shop_master.html (магазин пользователя) в окне браузера он будет выглядеть так, как показано на рис. 6.8.



Рис. 6.8. Шаблон user_shop_master.html (магазин пользователя) при первоначальной загрузке

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

Как здесь можно видеть, левая боковая панель пока пустая (ее, собственно, еще вовсе нет), а в центральной панели показаны некоторые элементы персонального магазина пользователя:

- кнопка **Выбор типа каталога продуктов**;
- имя владельца персонального магазина;
- количество доступных товаров в БД;
- ссылка на корзину покупателя;
- ссылка на активный заказ покупателя (не оплаченный);
- ссылка на покупки клиента (оплаченные заказы);
- баннер платежной системы, с которой работает магазин пользователя.

Ссылки на корзину, заказ и покупки мы пока не рассматриваем (еще не описаны соответствующие программные модули), а если нажать на кнопку **Выбор типа каталога продуктов**, появятся следующие опции выпадающего меню (рис. 6.9).

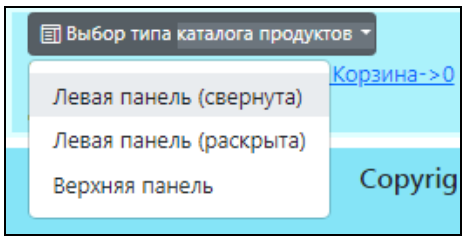


Рис. 6.9. Опции выбора типа каталога продуктов

Если выбрать здесь опцию **Левая панель (свернута)** (показать каталог категорий продуктов в виде свернутого дерева), откроется шаблон user_catalog_stat.html (рис. 6.10).

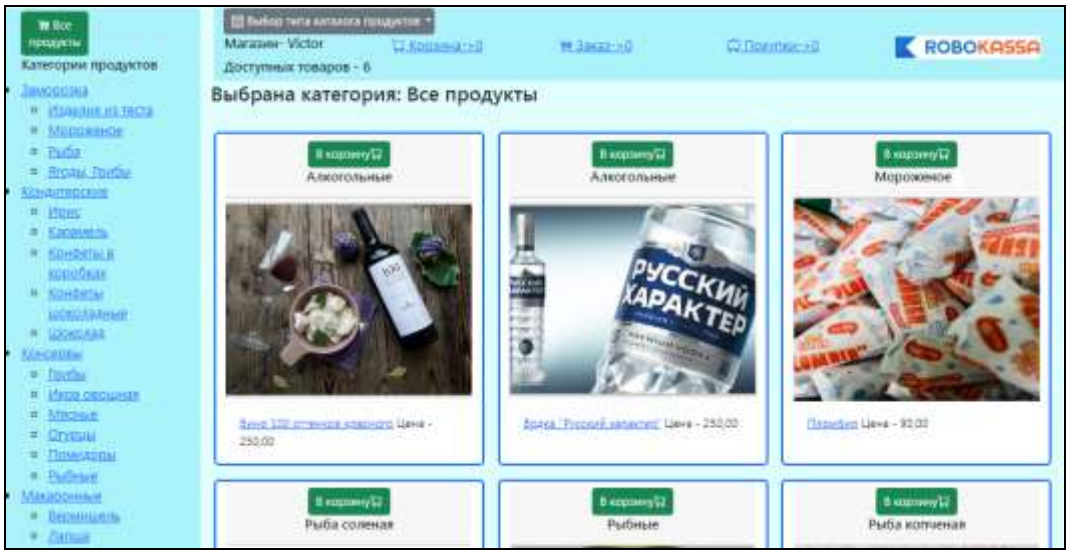


Рис. 6.10. Шаблон user_catalog_stat.html при первоначальной загрузке

Как здесь можно видеть, в левой боковой панели появились категории продуктов в виде раскрытого дерева, а в центральной панели — карточки с информацией о продуктах. При первоначальной загрузке этого шаблона на него будут выведены карточки всех продуктов из БД (с учетом постраничной пагинации). Если же в каталоге категорий продуктов выбрать одну из категорий — например, **Рыбные**, то содержимое центральной панели изменится, и в ней будут показаны продукты выбранной категории, включая дочерние (рис. 6.11).

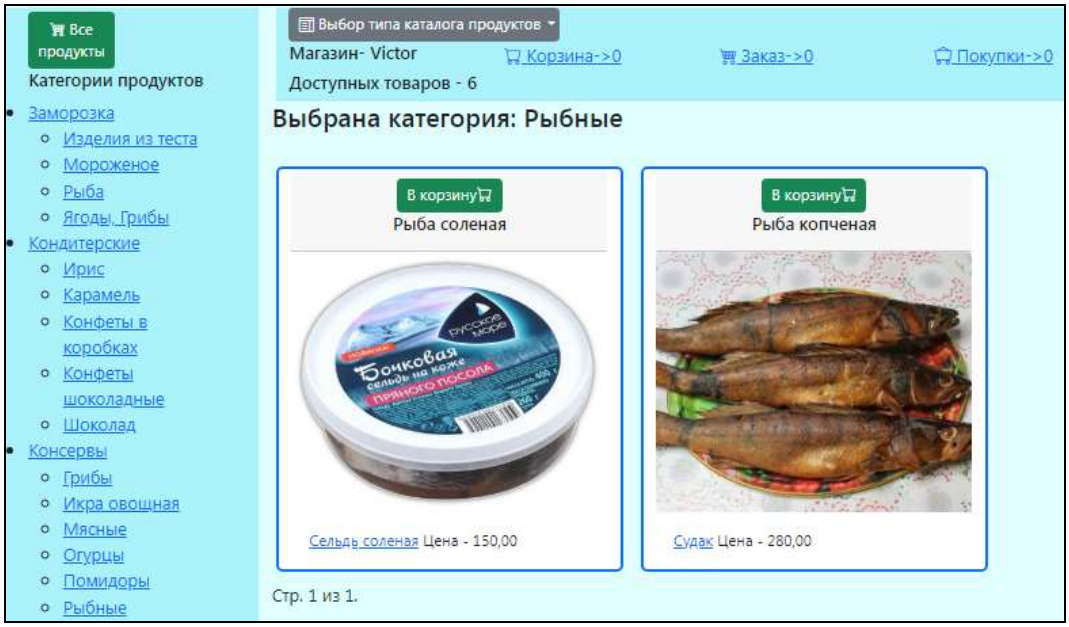


Рис. 6.11. Шаблон user_catalog_stat.html при выборе категории продуктов **Рыбные**

Обратите внимание, что дерево категорий продуктов в левой панели все время отображается в раскрытом виде, и пользователь всегда видит как категории продуктов, так и все дочерние подкатегории.

Если теперь в дочерних категориях выбрать конечную ветку, — например, **Рыба соленая**, то в центральной панели останутся карточки с продуктами только этой категории (рис. 6.12).



Рис. 6.12. Шаблон user_catalog_stat.html при выборе категории продуктов **Рыба соленая**

Обратите внимание, что наименование продукта представляет собой ссылку, при переходе по которой откроется страница с полноразмерной фотографией и полной информацией об этом продукте (рис. 6.13).



Рис. 6.13. Страница с подробными сведениями о продукте

6.4.5. Вывод категорий продуктов в центральной панели в виде мегаменю

В шаблоне user_shop_master.html при выборе типа каталога продуктов **Верхняя панель** активируется следующая ссылка:

```
<a class="dropdown-item" href="{% url 'master:user_catalog_up'%}">Верхняя
панель</a>
```

По этой ссылке активируется URL-маршрут `user_catalog_up` (листинг 6.20).

Листинг 6.20. URL-маршрут user_catalog_up

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `user_catalog_up` (листинг 6.21).

Листинг 6.21. Представление user_catalog_up

Примечание.

В демо-версии книги удалены листинги программ.

Здесь переменная `name_cat` получает значение `Все продукты`. Затем из БД подгружается объект `products` — все продукты из БД, и создается пагинатор для отображения 30 карточек продуктов на странице. Всё это через объект `context` передается в шаблон user_catalog_up.html.

Пояснение

Этот код работает в рамках интерактивной цифровой книги. В реальном проекте надо снять комментарий со строки `# user = request.user` и удалить или закомментировать строку `user = User.objects.get(id=2)`.

Код шаблона user_catalog_up.html приведен в листинге 6.22.

Листинг 6.22. Шаблон user_catalog_up.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне создана центральная панель, в которой отображено мегаменю категорий продуктов. В центральной панели также присутствует блок для показа карточек с продуктами, в который вставлено подключение к шаблону user_product.html (см. листинг 6.12).

В первой строке центральной панели шаблона user_catalog_up.html имеется кнопка `Все продукты`. С этой кнопкой связана следующая ссылка:

```
<a href="{% url 'master:user_catalog_up' %}" class="btn btn-success btn-sm">
       <i class="bi bi-cart4"></i> Все продукты</a>
```

По этой ссылке в конечном итоге будет вызвано представление `user_catalog_up`, в котором будут сняты фильтры на категории продуктов, и в центральной панели отобразятся все карточки продуктов из БД.

В центральной панели шаблона user_catalog_up.html в мегаменю с наименованием категории продукта связана следующая ссылка:

```
<a href="{% url 'master:user_product_up' id=node.id %}">{{ node.name }}</a>
```

По этой ссылке активируется URL-маршрут `user_product_up` (листинг 6.23), в который передается идентификатор выбранной категории (`id=node.id`).

Листинг 6.23. URL-маршрут user_product_up

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `user_product_up` (листинг 6.24).

Листинг 6.24. Представление user_product_up

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает выбранную категорию продукта, и в объект `products` загружаются все продукты выбранной категории и всех дочерних категорий (если они есть). Иными словами, на базу данных с продуктами накладывается фильтр по выбранной категории продукта из древовидного каталога в виде мегаменю. Затем все полученные параметры загружаются в словарь `context`, который передается в шаблон user_catalog_up.html (см. листинг 6.22), где и отображаются карточки продуктов с учетом фильтра.

Пояснение

Этот код работает в рамках интерактивной цифровой книги. В реальном проекте надо снять комментарий со строки `# user = request.user` и удалить или закомментировать строку `user = User.objects.get(id=2)`.

Итак, к этому моменту создан набор модулей, которые обеспечивают вывод страницы магазина пользователя с выборкой и фильтрацией продуктов по их категориям в древовидном мегаменю. Теперь можно проверить работу этих модулей.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 6** и перейти по ссылке **Мой магазин**.

6.4.6. Проверка работы модулей магазина пользователя с фильтрацией данных по дереву категорий продуктов в виде мегаменю

При первоначальной загрузке шаблона user_shop_master.html (магазин пользователя) в окне браузера он будет выглядеть так, как показано на рис. 6.14.



Рис. 6.14. Шаблон user_shop_master.html (магазин пользователя) при первоначальной загрузке

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

Как здесь можно видеть, левая боковая панель пока пустая (ее, собственно, еще вовсе нет), а в центральной панели показаны некоторые элементы персонального магазина пользователя:

- кнопка **Выбор типа каталога продуктов**;
- имя владельца персонального магазина;
- количество доступных товаров в БД;
- ссылка на корзину покупателя;
- ссылка на активный заказ покупателя (не оплаченный);
- ссылка на покупки клиента (оплаченные заказы);

- баннер платежной системы, с которой работает магазин пользователя.

Ссылки на корзину, заказ и покупки мы пока не рассматриваем (еще не описаны соответствующие программные модули), а если нажать на кнопку **Выбор типа каталога продуктов**, появятся следующие опции выпадающего меню (рис. 6.15).

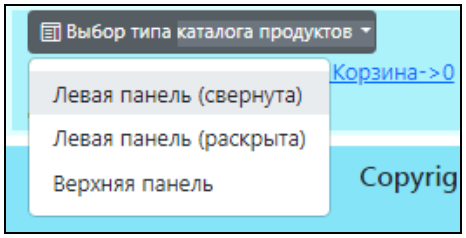


Рис. 6.15. Опции выбора типа каталога продуктов

Если выбрать здесь опцию **Верхняя панель** (показать каталог категорий продуктов в виде древовидного мегаменю), откроется шаблон user_catalog_up.html (рис. 6.16).

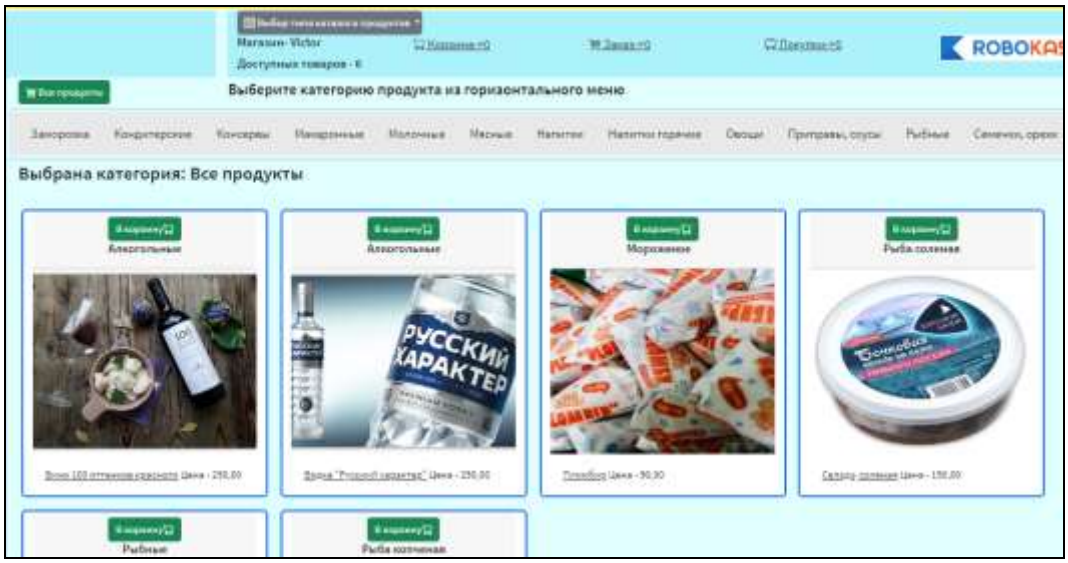


Рис. 6.16. Шаблон user_catalog_up.html при первоначальной загрузке

Как здесь можно видеть, левая боковая панель пустая, а в центральной панели отображаются категории продуктов в виде мегаменю, ниже которого находятся карточки с информацией о продуктах. При первоначальной загрузке этого шаблона на него будут выведены карточки всех продуктов из БД (с учетом постраничной пагинации). Если же в каталоге категорий продуктов выбрать одну из категорий — например, **Рыбные**, то содержимое центральной панели изменится, и в ней будут показаны продукты выбранной категории, включая дочерние (рис. 6.17).

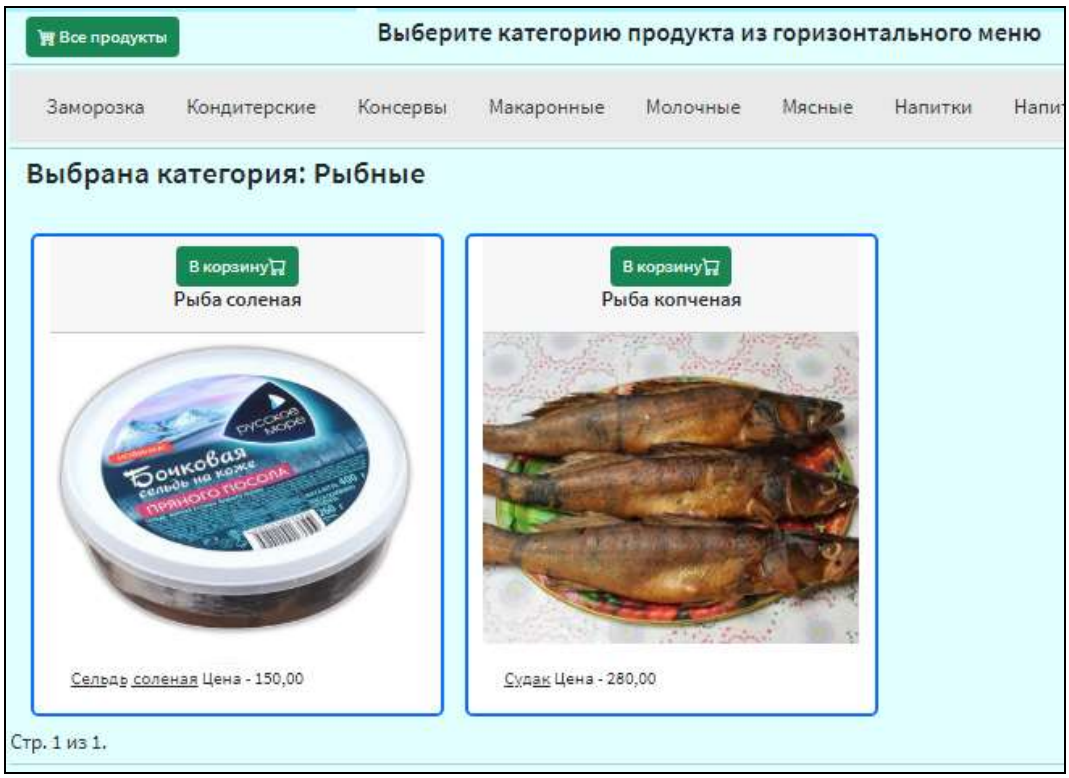


Рис. 6.17. Шаблон user_catalog_up.html при выборе категории продуктов **Рыбные**

Если теперь в дочерних категориях мегаменю выбрать конечную ветку, — например, **Рыба соленая**, то в центральной панели останутся карточки с продуктами только этой категории (рис. 6.18).

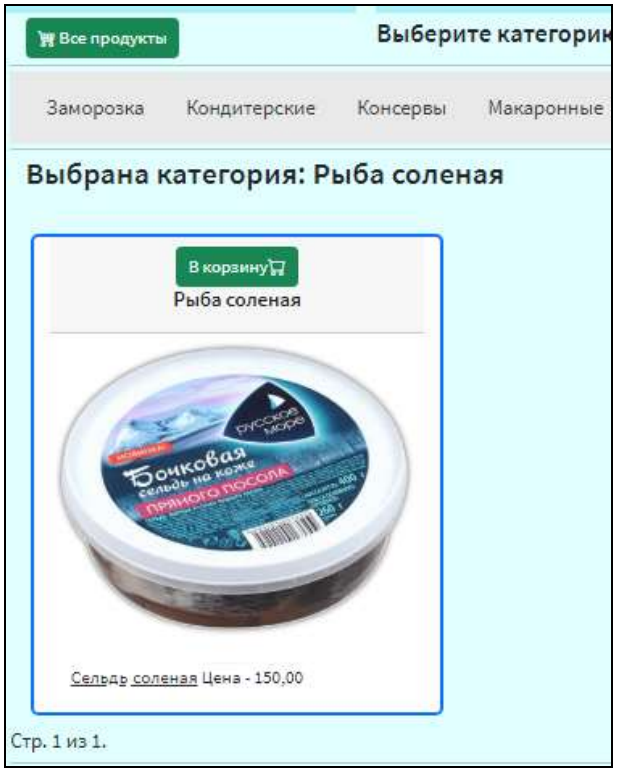


Рис. 6.18. Шаблон user_catalog_up.html при выборе категории продуктов **Рыба соленая**

Обратите внимание, что наименование продукта представляет собой ссылку, при переходе по которой откроется страница с полноразмерной фотографией и полной информацией о данном продукте (рис. 6.19).



Рис. 6.19. Страница с подробными сведениями о продукте

На рисунках, приведенных в предыдущих разделах, в карточках продуктов имеется кнопка **В корзину** — она предназначена для того, чтобы пользователь положил этот товар в корзину (создал предварительный заказ). Настало время создать программный код, который обеспечит формирование корзины (предварительного заказа).

6.5. Модули формирования корзины (предварительного заказа)

При формировании корзины покупатель должен иметь следующие возможности:

- добавить товар (продукт) в корзину;
- скорректировать информацию о товарах (продуктах), находящихся в корзине;
- удалить товар (продукт) из корзины.

Создадим программные модули для выполнения всех этих действий.

6.5.1. Добавление продуктов (товаров) в корзину

В шаблоне с карточками продуктов user_product.html (см. листинг 6.12), имеется кнопка `В корзину` (фактически их там две, одна — традиционная — закомментирована). При нажатии на эту кнопку выбранный продукт будет добавлен в корзину. С традиционной (закомментированной) кнопкой связана следующая ссылка:

```
href="{% url 'master:cart' id=mb.id %}"
```

С кнопкой, для которой предусмотрен вращающийся спин, связана следующая ссылка:

```
action="{% url 'master:cart' id=mb.id %}"
```

По этим ссылкам вызывается URL-маршрут `cart` (листинг 6.25), в который передается идентификатор выбранного продукта (`id=mb.id`).

Листинг 6.25. URL-маршрут cart

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту активируется представление `cart` (листинг 6.26).

Листинг 6.26. Представление cart

Примечание.

В демо-версии книги удалены листинги программ.

Это представление выполняет несколько функций:

- получить данные о продукте (объект `product`);
- получить данные о пользователе (объект `user`);
- добавить или открыть заказ (на этом этапе — корзину);
- сформировать номер заказа (если его еще нет);
- проверить, есть ли выбранный товар в корзине (если товар уже добавлен в корзину, то выдать соответствующее предупреждение);
- выполнить расчет суммы заказа (стоимость продуктов в корзине);
- вернуться к списку товаров выбранной категории.

Представление `cart` получает идентификатор продукта, который нужно добавить в корзину (`id`), загружает в объект `product` сведения об этом продукте, после чего ему нужно получить данные о зарегистрированном пользователе системы (объект `user`).

Примечание

В цифровой интерактивной книге в качестве примера из БД выбирается тестовый пользователь с `id=2`, в реальном приложении нужно использовать инструкцию `user = request.user`.

Затем добавляется новый заказ (корзина) или открывается существующий. Если заказ создан, и у него нет номера, то формируется номер заказа. В нашем приложении номер заказа составной и уникальный — он складывается из id заказа в БД, id пользователя и текущей даты.

Далее проверяется условие — есть ли выбранный продукт в заказе (в корзине). Если его в заказе (в корзине) нет, то он добавляется в таблицу (в модель) `Product_OrderItem`. Если же заданный продукт уже добавлен в корзину, то эта процедура пропускается, и формируется текст сообщения `Этот товар уже добавлен в корзину`.

На следующем этапе делается пересчет суммы заказа (стоимость всех продуктов корзины) и она обновляется в таблице (в модели) `Product_Order`.

В завершение из переменной сессии определяется, с каким типом каталога категорий продуктов работал пользователь, и делается возврат к соответствующему шаблону. При этом отрабатывает функция контекстного процессора `cart_kol`, в которой определяется количество продуктов, добавленных в корзину пользователя, и это значение автоматически обновляется в поле `{{cart_kol}}` шаблона user_shop_master.html (рис. 6.20).

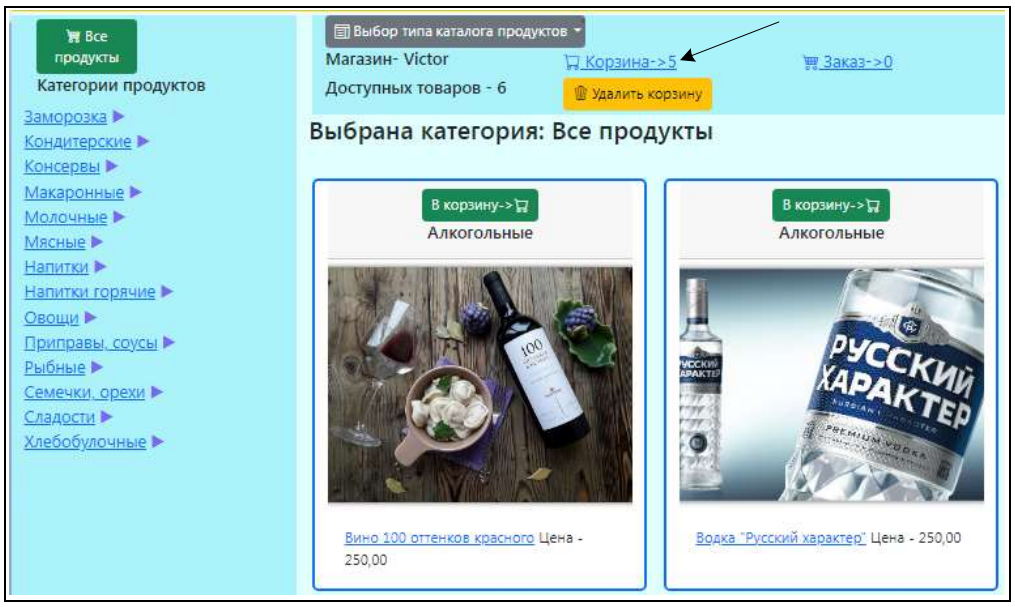


Рис. 6.20. Ссылка на корзину с указанием количества добавленных в нее продуктов

6.5.2. Вывод содержимого корзины (предварительного заказа)

Когда корзина сформирована, мы должны предоставить пользователю возможность посмотреть ее содержание и выполнить некоторые действия: изменить количество товара в корзине, удалить товар из корзины или на основе корзины сформировать окончательный заказ. Создадим набор программных модулей, которые реализуют все эти действия.

На странице магазина пользователя (в шаблоне user_shop_master.html) имеется ссылка `Корзина<количество товаров>`, в которой показано количество товаров в корзине:

```
<a href="{% url 'master:cart_edit' %}"><i class="bi bi-cart"></i> Корзина->{{cart_kol}}</a>
```

По этой ссылке вызывается URL-маршрут `cart_edit` (листинг 6.27).

Листинг 6.27. URL-маршрут cart_edit

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `cart_edit` (листинг 6.28) для просмотра и корректировки товаров в корзине

Листинг 6.28. Представление cart_edit

Примечание.

В демо-версии книги удалены листинги программ.

Здесь объект user_id получает идентификатор текущего пользователя, затем из БД загружается набор параметров корзины, которые в итоге передаются в словарь context, а этот словарь передается в шаблон cart_edit.html.

В представлении `cart_edit` нужно обратить внимание на одну особенность: в словарь `context` передается объект `many` (валюта), однако в теле представления объект `many` не получает значения. Разберемся, что это за объект, и откуда он взялся. Этот объект содержит значок валюты — специальный знак рубля, доллара, евро и т. п. Для того чтобы этот объект был доступен в любом модуле проекта, его нужно определить в файле settings.py (листинг 6.29).

Листинг 6.29. Определение объекта many в файле settings.py

Примечание.
В демо-версии книги удалены листинги программ.

Как можно видеть, здесь активной валютой является российский рубль. Но чтобы объект `many` стал доступным для всех представлений приложения, его нужно импортировать из файла settings.py (листинг 6.30).

Листинг 6.30. Импорт объекта MANY из файла settings.py

Примечание.
В демо-версии книги удалены листинги программ.

В завершение представление `cart_edit` загружает шаблон cart_edit.html (листинг 6.31).

Листинг 6.31. Шаблон cart_edit.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне есть несколько ссылок:

- на названия продуктов;
- на количество товаров в корзине;
- на опции `Удалить`;
- на создание заказа.

С названием продукта связана следующая ссылка:

```
<a href="{% url 'master:product_detail' pk=obj.product.id
%}">{{obj.product.name}}</a>
```

По этой ссылке будет показан страница с детальной информацией о продукте (программный код обработки этой ссылки уже реализован и показан в предыдущих разделах).

Создадим теперь программный код модулей для корректировки количества товара в корзине.

6.5.3. Корректировка количества товара в корзине

Для корректировки количества товара в корзине необходимо создать соответствующую форму — `Product_kol_ModelForm` (листинг 6.32).

Листинг 6.32. Форма Product_kol_ModelForm

Примечание.
В демо-версии книги удалены листинги программ.

В этой форме для корректировки доступно всего одно поле — `quantity` (количество товара в корзине).

При активации в шаблоне cart_edit.html ссылка на количестве товаров в корзине имеет следующий вид:

```
<a href="{% url 'master:kol_product_update' id=obj.id %}">{{obj.quantity}}</a>
```

По этой ссылке активируется URL-маршрут `kol_product_update` (листинг 6.33), в который передается идентификатор текущего продукта (`id=obj.id`).

Листинг 6.33. URL-маршрут kol_product_update

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `kol_product_update` (листинг 6.34), в которое передается идентификатор текущего продукта (`int:id`).

Листинг 6.34. Представление kol_product_update

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор текущего продукта (`id`), на основе которого в объект `my_product` загружаются сведения о текущем продукте, а в объект `name_product` — наименование продукта. Затем объекту `form` передается форма `Product_kol_ModelForm`, в которую загружаются сведения о текущем продукте. Полученные параметры загружаются в контекстный словарь `content`, который передается шаблону kol_product_form.html (листинг 6.35).

Листинг 6.35. Шаблон kol_product_form.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне для редактирования доступно всего одно поле — количество продукта в корзине. После нажатия на кнопку `Сохранить` будет в виде POST-запроса выполнен возврат к представлению `kol_product_update`, где произойдет обновление количества товара в корзине и общей суммы заказа.

Для пересчета и обновления суммы заказа служит функция `sum_zak(request)` (листинг 6.36).

Листинг 6.36. Функция sum_zak

Примечание.

В демо-версии книги удалены листинги программ.

Не нужно забывать, что в файле views.py код функции `sum_zak` должен быть определен до того, как написан код представления `kol_product_update`.

Итак, посмотрим, как работают модули корректировки количества продукта в корзине.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 6** и перейти по ссылке **Мой магазин**.

При вызове шаблона user_shop_master.html (магазин пользователя) в окне браузера он будет выглядеть так, как показано на рис 6.21.

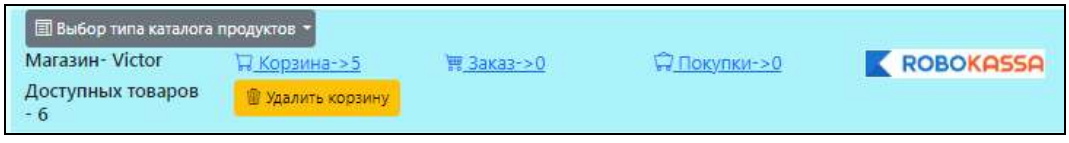


Рис. 6.21. Ссылка на корзину с пятью продуктами в шаблоне user_shop_master.html

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном

сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

Если здесь щелкнуть левой кнопкой мыши на ссылке `Корзина->5`, будет показан шаблон cart_edit.html (рис. 6.22).

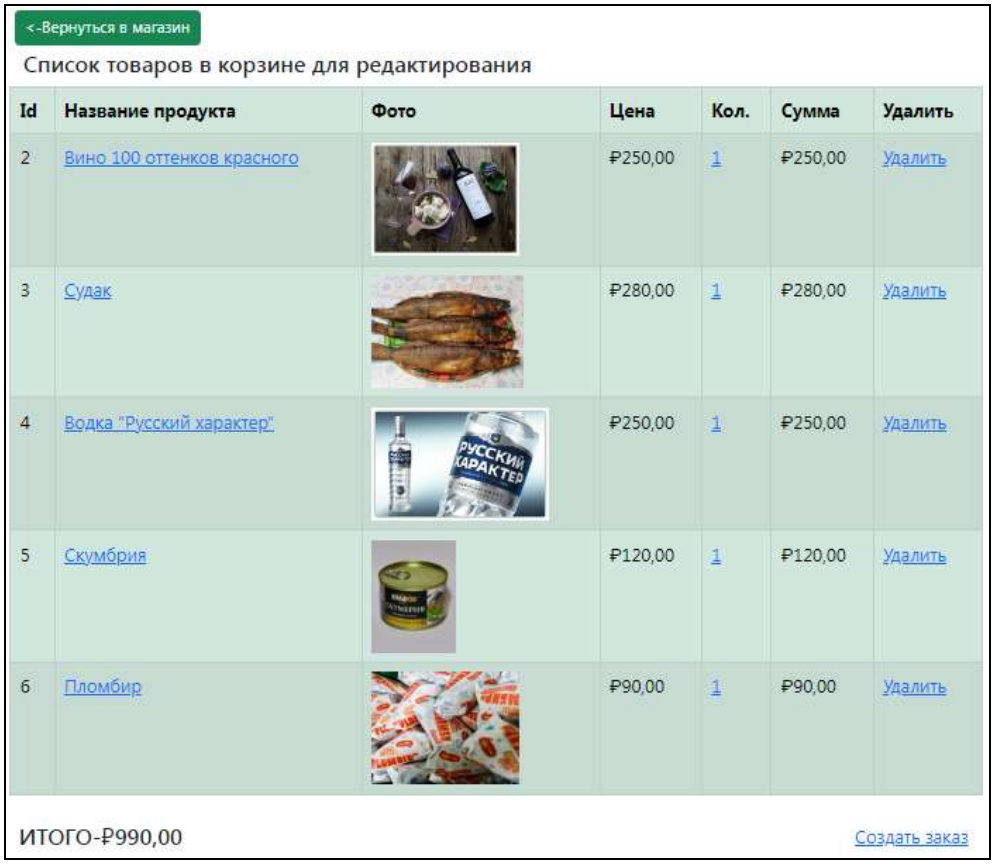


Рис. 6.22. Шаблон cart_edit.html со списком продуктов, добавленных в корзину

Как здесь можно видеть, в шаблоне в виде таблицы выведен список товаров, находящихся в корзине. Под ним приведена сумма предварительного заказа и ссылка **Создать заказ**. В каждой строке таблицы наименование продукта и количество продукта в корзине представлены в виде ссылок.

Щелчок левой кнопкой мыши на наименовании продукта откроет страницу с подробной информацией о нем (рис. 6.23), а на количестве продукта в корзине — шаблон kol_product_form.html для корректировки этого количества (рис. 6.24).



Рис. 6.23. Страница с подробными сведениями о продукте

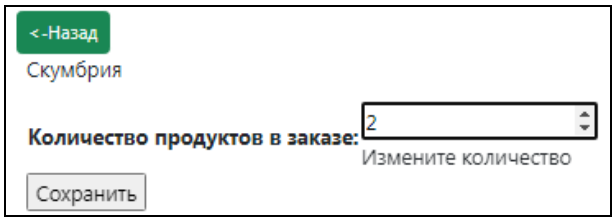


Рис. 6.24. Страница для корректировки количества продукта в корзине (в предварительном заказе)

По нажатию на кнопку **Сохранить** будет выполнен возврат к шаблону cart_edit.html, где будет отображена скорректированная информация (рис. 6.25). Как здесь можно видеть, в корзине теперь находятся две банки продукта «Скумбрия», изменились сумма за этот товар и общая стоимость предварительного заказа.

<-Вернуться в магазин

Список товаров в корзине для редактирования

Id	Название продукта	Фото	Цена	Кол.	Сумма	Удалить
2	Вино 100 оттенков красного		₽250,00	1	₽250,00	Удалить
3	Судак		₽280,00	1	₽280,00	Удалить
4	Водка "Русский характер"		₽250,00	1	₽250,00	Удалить
5	Скумбрия		₽120,00	2	₽240,00	Удалить
6	Пломбир		₽90,00	1	₽90,00	Удалить

ИТОГО-₽1110,00 Создать заказ

Рис. 6.25. Шаблон cart_edit.html с отображением скорректированной информацией

6.5.4. Удаление товара из корзины

В шаблоне cart_edit.html с опцией `Удалить` связана следующая ссылка:

```
<a href="{% url 'master:cart_product_del' id=obj.id %}">Удалить</a>
```

По этой ссылке активируется URL-маршрут `cart_product_del` (листинг 6.37), в который передается идентификатор текущего продукта (`id=obj.id`).

Листинг 6.37. URL-маршрут cart_product_del

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `cart_product_del` (листинг 6.38), в которое передается идентификатор текущего продукта (`int:id`).

Листинг 6.38. Представление cart_product_del

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении сначала запоминаются сведения об удаляемом товаре, которые затем передаются в универсальный шаблон del_yesno.html для подтверждения или отмены удаления. Нажатие в шаблоне del_yesno.html кнопки **Удалить** ведет к удалению из корзины выбранного продукта, а нажатие кнопки **Отмена** — к отказу от выполнения процедуры удаления. После удаления продукта из корзины вызывается функция sum_zak(request), в которой выполняется пересчет суммы предварительного заказа.

Итак, посмотрим, как работают модули удаления продукта из корзины.

По щелчку на ссылке **Удалить** в строке шаблона cart_edit.html с продуктом «Пломбир», будет вызван универсальный шаблон del_yesno.html (рис. 6.26).

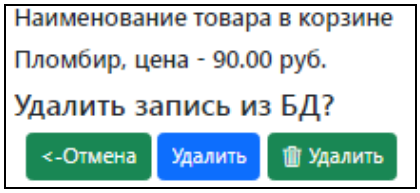


Рис. 6.26. Шаблон del_yesno.html с данными об удаляемом продукте

По щелчку здесь на кнопке **Удалить** будет выполнен возврат к шаблону cart_edit.html, в котором строки с этим продуктом уже не будет, а сумма предварительного заказа пересчитана (рис. 6.27).

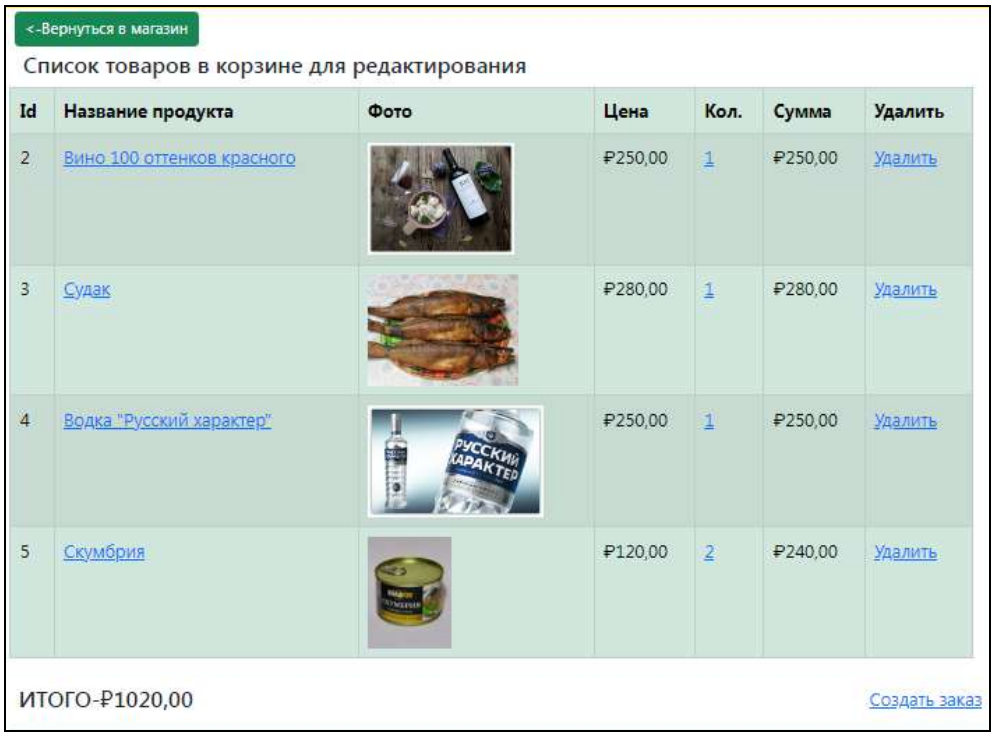


Рис. 6.27. Шаблон cart_edit.html после удаления продукта **Пломбир**

6.5.5. Формирование заказа для оплаты на основе набора продктов, помещенных в корзину

Когда пользователь завершит формирование корзины, ее нужно трансформировать в заказ, который можно оплатить. Для этого достаточно в модели `Product_Order` изменить значение поля `status` с `cart` на `for_payment`.

В шаблоне cart_edit.html ссылка на создание заказа имеет следующий вид:

```
<a href=«{% url 'master:make_order' %}»>Создать заказ</a>
```

По этой ссылке активируется URL-маршрут `make_order` (листинг 6.39).

Листинг 6.39. URL-маршрут make_order

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `make_order` (листинг 6.40).

Листинг 6.40. Представление make_order

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении проверяется условие: есть ли у пользователя активные заказы. Если активных заказов нет, то формируется заказ (`order.status = 'for_payment'`) и делается возврат к одной из страниц со списком товаров магазина. Если у пользователя есть активный заказ, то на странице cart_edit.html выдается соответствующее предупреждение.

На странице магазина пользователя user_shop_master.html имеется ссылка на активный заказ: `Заказ-><сумма заказа>`, в которой показана сумма активного заказа. При переходе по этой ссылке пользователь может посмотреть содержание активного заказа. С этой ссылкой связан следующий программный код:

```
<a href=«{% url 'master:my_order' %}»><i class=«bi bi-cart4»></i> Заказ-
>{{sum_cart}}</a>
```

По этой ссылке вызывается URL-маршрут `my_order` (листинг 6.41).

Листинг 6.41. URL-маршрут my_order

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `my_order` (листинг 6.42).

Листинг 6.42. Представление my_order

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении сначала подключаются модули платежной платформы Robokassa, после чего создается объект `user` (зарегистрированный пользователь магазина) и проверяется условие, есть ли у него активный заказ. Если активный заказ есть, то в набор объектов из БД загружаются параметры заказа и выполняется проверка, есть ли заказанные продукты в наличии (пока пользователь формировал корзину, часть товаров могли приобрести другие покупатели, и их на складе магазина теперь нет).

Если хотя бы одного элемента заказа в наличии нет, то для пользователя формируется сообщение: `Вы опоздали с оплатой, часть товара уже продана, создайте новый заказ или ждите поступления товара`. Если же весь набор продуктов заказа на складе магазина присутствует, то формируются текущая дата и время и ссылка на сервисы платежной платформы Robokassa.

После этого список продуктов заказа вместе со стоимостями элементов заказа упаковывается в объект `mylist`.

В завершение все сформированные объекты упаковываются в словарь `context`, который передается в шаблон my_order.html (листинг 6.43).

Листинг 6.43. Шаблон my_order.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне имеются две таблицы: со сведениями о заказе (номер заказа и способ оплаты) и с продуктами, которые включены в заказ. После второй таблицы выводится итоговая сумма к оплате. Далее следуют две ссылки: `Оплатить заказ (успешно)` и `Оплатить заказ (ошибка)` — они служат для отладки работы модулей, которые будут обрабатывать ответ от платежной платформы. При переходе по этим ссылкам реального обращения к пла-

тежной платформе не произойдет, а будет выполнена соответственно: имитация ответа от платежной платформы об успешной оплате заказа банковской картой или имитация ответа о том, что оплата заказа банковской картой не прошла.

> ***Пояснение***
> Оплата заказа банковской картой на платежной платформе Robokassa может производиться в двух режимах: тестовой оплаты и реальной оплаты. Подробные сведения об этих режимах и параметрах платежной системы Robokassa приводятся в *разд. 6.5.8–6.5.11*.

Реальное обращение к платежной платформе Robokassa будет происходить при нажатии на кнопку `Оплатить заказ`. С этой кнопкой связана следующая ссылка:

```
<a href="{{link}}" class="btn btn-success my-1">
    <i class="bi bi-credit-card-2-front"></i> Оплатить заказ</a>
```

То есть в ней указана переменная шаблона `{{link}}`, значение которой присваивается в представлении `my_order`.

6.5.6. Удаление корзины или активного заказа

Сведения о продуктах в корзине и в предварительном заказе хранятся в одной модели (таблице) — `Products_order`. Корзину от предварительного заказа и от оплаченного заказа (покупки) можно отличить по содержанию поля `status`. У корзины это поле имеет значение `cart`, у предварительного заказа — `for_payment` (к оплате), у оплаченного заказа — `paid` (оплачен). Перечень товаров, которые находятся в корзине, в предварительном заказе и в оплаченном заказе содержится в модели (таблице) `Products_orderItem`.

Таблица `Products_order` связана с пользователем системы и зависима от него по схеме каскадного удаления — то есть при удалении пользователя автоматически будут удалены и все его заказы. При этом сама таблица `Products_order` является главной сущностью для таблицы с элементами заказов `Products_orderItem`, которая, будучи дочерней по отношению к таблице `Products_order`, также связана с ней по схеме каскадного удаления — то есть при удалении заказа автоматически будут удалены и все элементы заказа. Кроме того, элементы заказа связаны с родительской таблицей `Products` (продукты) по схеме связи `PROTECT`, определяющей запрещение удаления продукта в том случае, если он присутствует в одном из заказов.

Таким образом, для удаления корзины, предварительного заказа или оплаченного заказа достаточно удалить запись из таблицы `Products_order` — при этом в связанной таблице `Products_orderItem` все связанные записи должны удалиться автоматически.

Кнопки для удаления активного заказа и корзины находятся на странице магазина пользователя в шаблоне user_shop_master.html:

```
<a href="{% url 'master:order_del' id=1 id_zak=id_cart_m %}"
   class="btn btn-warning btn-sm my-1">
    <i class="bi bi-trash3"></i> Удалить корзину</a>


<a href="{% url 'master:order_del' id=2 id_zak=id_zak_m %}"
   class="btn btn-warning btn-sm my-1">
    <i class="bi bi-trash3"></i> Удалить заказ</a>
```

Здесь активируется URL-маршрут `order_del`, но в него передаются разные значения аргументов:

- `Удалить корзину` — идентификатор нажатой кнопки `id=1` и идентификатор удаляемой записи `id_zak=id_cart_m` (идентификатор корзины);
- `Удалить заказ` — идентификатор нажатой кнопки `id=2` и идентификатор удаляемой записи `id_zak=id_zak_m` (идентификатор активного заказа).

То есть в обоих случаях срабатывает URL-маршрут `order_del` (листинг 6.44).

Листинг 6.44. URL-маршрут order_del

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

Этот маршрут принимает два параметра: `int:id` — идентификатор нажатой кнопки и `int:id_zak` — идентификатор либо корзины, либо предварительного заказа.

По этому маршруту вызывается представление `order_del` (листинг 6.45).

Листинг 6.45. Представление order_del

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

Это представление получает два параметра: `id` — идентификатор нажатой кнопки и `id_zak` — идентификатор либо корзины, либо предварительного заказа (если эти два параметра пусты, то производится возврат на страницу магазина пользователя). Затем создается объект `text1`, в который — в зависимости от параметра `id` — заносятся разные текстовые значения.

Объект `text1` через словарь `content` передается в универсальный шаблон-форму del_yesno.html, где пользователю будет предложено удалить либо активную корзину, либо активный предварительный заказ. Из этого шаблона в виде POST-запроса будет получен идентификатор нажатой кнопки и идентификатор удаляемой записи. Затем, в зависимости от нажатой кнопки, будет удалена запись с содержанием корзины или с содержанием предварительного заказа.

> ***Пояснение***
> В этом проекте корзина и предварительный заказ находятся в одной записи, и ее можно было бы удалить в одной инструкции: `order.delete()` — без анализа нажатой кнопки. Однако с учетом того, что в ваших проектах для хранения корзины и предварительного заказа могут быть задействованы разные модели, инструкции `order.delete()` разделены.

6.5.7. Проверка работы модулей для перевода корзины в предварительный заказ

Итак, набор модулей для перевода корзины в предварительный заказ готов, и можно проверить его работу.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 6** и перейти по ссылке **Мой магазин**.

На странице магазина пользователя присутствует ссылка на корзину, в которой указано количество продуктов, в нее помещенных (рис. 6.28).

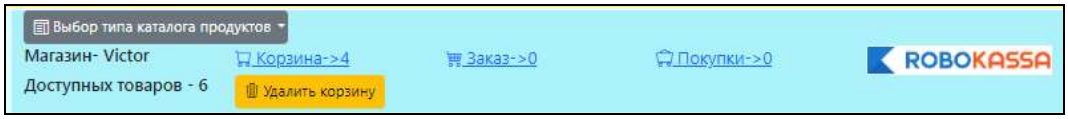


Рис. 6.28. Магазин пользователя со ссылкой на активную корзину

> ***Примечание***
> В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

Как здесь можно видеть, в корзину положено 4 продукта, а под ссылкой на корзину появилась кнопка **Удалить корзину**. Если в этом шаблоне щелкнуть левой кнопкой мыши на ссылке **Корзина**, откроется шаблон cart_edit.html со списком продуктов, которые лежат в корзине (рис. 6.29).

<-Вернуться в магазин

Список товаров в корзине для редактирования

Id	Название продукта	Фото	Цена	Кол.	Сумма	Удалить
2	Вино 100 оттенков красного		₽250,00	1	₽250,00	Удалить
3	Судак		₽280,00	1	₽280,00	Удалить
4	Водка "Русский характер"		₽250,00	1	₽250,00	Удалить
5	Скумбрия		₽120,00	2	₽240,00	Удалить

ИТОГО-₽1020,00 Создать заказ

Рис. 6.29. Шаблон cart_edit.html со списком продуктов, которые лежат в корзине

Если в этом шаблоне щелкнуть левой кнопкой мыши на ссылке **Создать заказ**, состояние корзины изменится (в ней останется 0 продуктов), а также изменится состояние ссылки на предварительные заказы покупателя (рис. 6.30).

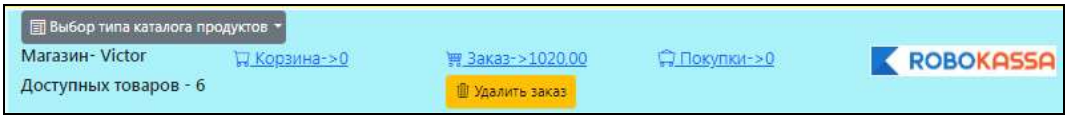


Рис. 6.30. Результаты работы модулей перевода корзины в статус предварительного заказа

Как здесь можно видеть, корзина обнулилась, а на ссылке с предварительным заказом появилась сумма заказа: **Заказ->1020.00**. При этом кнопка **Удалить корзину** исчезла, но появилась кнопка **Удалить заказ**.

Если в этом шаблоне щелккнуть левой кнопкой мыши на ссылке **Заказ->1020.00**, откроется шаблон my_order.html со списком продуктов, которые лежат в предварительном заказе (рис. 6.31).

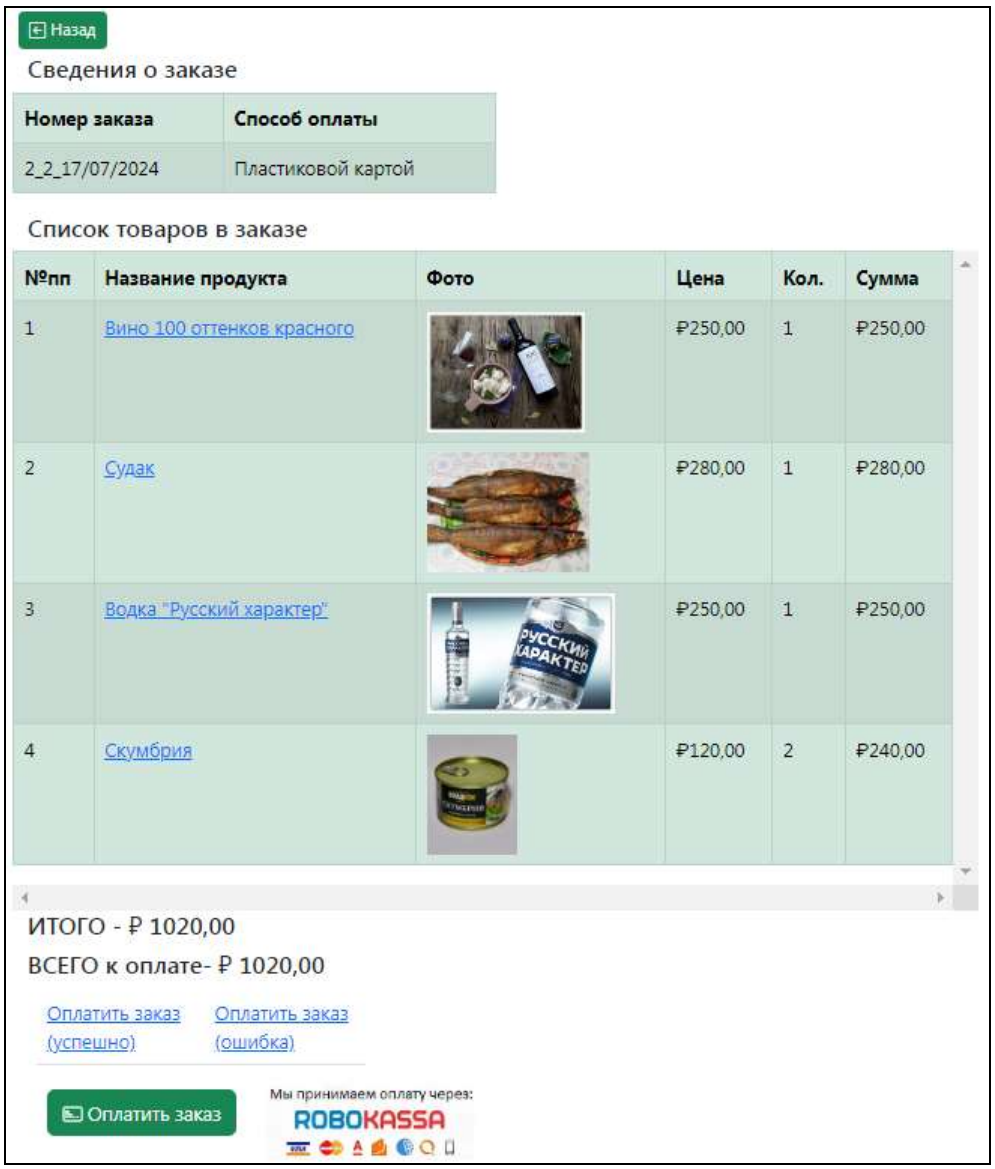


Рис. 6.31. Список продуктов, которые лежат в предварительном заказе

Как здесь можно видеть, в предварительном заказе лежат те же продукты, что были помещены в корзину, название продукта представляет собой ссылку — переход по этой ссылке откроет страницу с подробной информацией о продукте (рис (6.32), имеются также две ссылки: **Оплатить заказ успешно** и **Оплатить заказ (ошибка) —** и кнопка **Оплатить заказ**.



Рис. 6.32. Сведения о продукте, полученные из предварительного заказа

Функциональность ссылок **Оплатить заказ (успешно)**, **Оплатить заказ (ошибка)** и кнопки **Оплатить заказ** мы рассмотрим немного позднее, после создания модулей приема оплаты по банковским картам, а пока проверим, как работает кнопка **Удалить корзину**. Для этого пополним корзину, положив в нее какой-нибудь продукт (рис. 6.33).

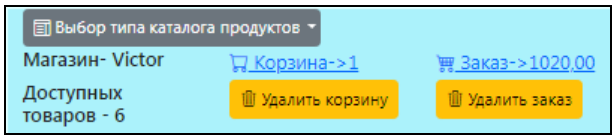


Рис. 6.33. Активная корзина с одним продуктом

Если здесь нажать на кнопку **Удалить корзину**, то откроется шаблон с вопросом на подтверждение или отмену удаления корзины (рис. 6.34).

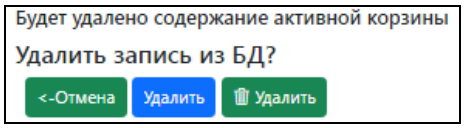


Рис. 6.34. Подтверждение/отмена удаления корзины

По нажатию на кнопку **Удалить** записи БД с корзиной и ее содержимым будут удалены, а на странице магазина пользователя показана корзина с нулевым содержанием (рис. 6.35).

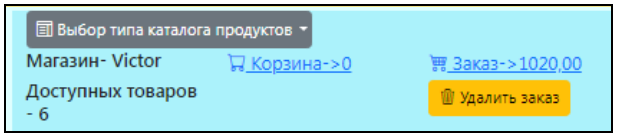


Рис. 6.35. Состояние корзины после удаления данных

Если теперь перейти по ссылке с обнуленной корзиной, пользователь получит сообщение о том, что корзина пуста (рис. 6.36).

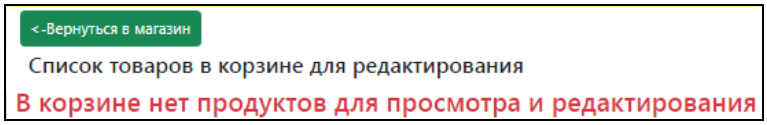


Рис. 6.36. Сообщение об отсутствии в корзине каких-либо продуктов

А теперь проверим, как работает кнопка **Удалить заказ** — для чего положим в корзину какой-нибудь продукт и переведем ее в состояние предварительного заказа (рис. 6.37).

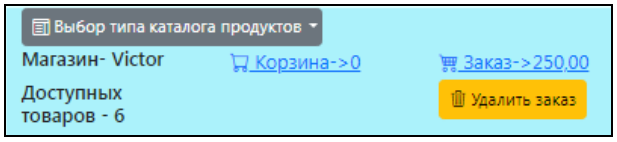


Рис. 6.37. Магазин пользователя со сформированным предварительным заказом

Если здесь нажать на кнопку **Удалить заказ**, то откроется шаблон с вопросом на подтверждение или отмену удаления заказа (рис. 6.38).

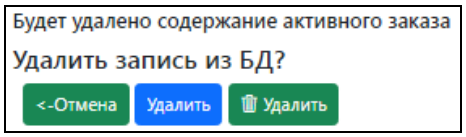


Рис. 6.38. Подтверждение/отмена удаления заказа

По нажатию на кнопку **Удалить** записи БД с предварительным заказом и ее содержимым будут удалены, а на странице магазина пользователя показан предварительный заказ с нулевым содержанием (рис. 6.39).

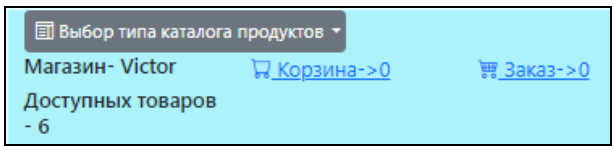


Рис. 6.39. Состояние предварительного заказа после удаления данных

Если теперь перейти по ссылке с обнуленным предварительным заказом, пользователь получит сообщение о том, что у него активных заказов нет (рис. 6.40).

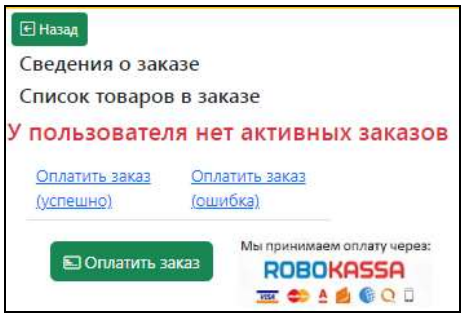


Рис. 6.40. Сообщение об отсутствии активного заказа

6.5.8. Интеграция интернет-магазина с платежной платформой Robokassa

Чтобы реализовать оплату заказов банковскими картами нужно загрузить в приложение параметры платежной системы, а для работы с системой пройти регистрацию на платежной платформе, получить параметры интернет-магазина, пароли для обмена данными между приложением и платежной платформой, а также набор соответствующих модулей или скриптов. Сведения о том, как зарегистрировать интернет-магазин на платежной платформе Robokassa и получить набор необходимых атрибутов приводятся в *приложении 2*. А здесь мы подробно рассмотрим, как эти параметры использовать в своем приложении.

Параметры доступа к платежной платформе Robokassa прописываются в файле settings.py вашего проекта (листинг 6.46).

Листинг 6.46. Параметры платежной платформы Robokassa в файле settings.py

Примечание.

В демо-версии книги удалены листинги программ.

В этом файле нужно указать следующие регистрационные параметры:

- идентификатор магазина: ROBOKASSA_LOGIN;
- реальные пароли: ROBOKASSA_PASSWORD1 и ROBOKASSA_PASSWORD2;
- тестовые пароли: ROBOKASSA_PASSWORD1_TEST и ROBOKASSA_PASSWORD2_TEST;
- тестовая оплата: IS_TEST = 1;
- реальная оплата: IS_TEST = 0.

При отладке программных модулей надо активировать режим тестовой оплаты (IS_TEST = 1) — в этом режиме деньги с банковских карт сниматься не будут. Когда работа всех моделей будет отлажена, нужно закомментировать или удалить строку IS_TEST = 1 и снять комментарий со строки IS_TEST = 0. После этого при оплате заказов деньги с реальных карт покупателей будут поступать на счет магазина продавца.

Чтобы представления имели доступ к параметрам магазина, нужно выполнить соответствующую процедуру импорта в файле view.py вашего приложения (листинг 6.47).

Листинг 6.47. Загрузка параметров магазина Robokassa в файле views.py

Примечание.

В демо-версии книги удалены листинги программ.

Эти параметры можно использовать в представлении my_order:

```
# сформировать ссылку на сервис Robokassa
if is_test == 1:
    pass_1 = pass1_test
else:
    pass_1 = pass1
link = rk.generate_payment_link(id_shop, pass_1, sum_to_pay, order.id, or-
der.num_order, is_test)
```

Здесь делается проверка того, в каком режиме выполняется оплата заказа: если в тестовом режиме, то в объект pass_1 загружаются тестовые пароли платежной платформы, если в реальном — в объект pass_1 загружаются рабочие пароли. После этого формируется объект link для соединения с платежной платформой, для чего используется функция generate_payment_link модуля rk (rk.generate_payment_link). Модуль rk был подключен в первой строке представления my_order (см. листинг 6.42):

```
# Подключение модулей платежной системы robokassa
    import moduli.robokassa as rk
```

Программный модуль robokassa (мы дали ему новое имя as rk) можно получить на официальном сайте платежной системы Robokassa и адаптировать под свой проект. В нашем проекте мы создали каталог moduli и добавили в него файл robokassa.py (листинг 6.48).

Листинг 6.48. Программный код модуля robokassa.py

Примечание.

В демо-версии книги удалены листинги программ.

Этот файл вы можете подключать к тем своим проектам, которые используют платежную платформу Robokassa. Здесь в функции __main__ можно задействовать параметры вашего магазина для проверки корректности подключения его к платежной платформе (ссылка на платежную платформу будет сохранена в объекте p).

На этом мы закончили знакомство с процедурой интеграции нашего интернет-магазина с платежной платформой Robokassa. Теперь можно вернуться к представлению my_order и шаблону my_order.html, чтобы познакомится с модулями ответа платежной платформы об успешной (или не успешной) оплате заказа банковской картой.

6.5.9. Обработка сбоя оплаты заказа пластиковой картой

В случае сбоя оплаты заказа банковской картой пользователь должен получить соответствующее сообщение. Разработаем набор модулей, которые будут обрабатывать такую ошибку. В шаблоне my_order.html со ссылкой `Оплатить заказ (ошибка)` связан следующий программный код:

```
<a href="{% url 'master:pay_order_bad' %}">Оплатить заказ (ошибка)</a>
```

По этой ссылке будет активирован URL-маршрут `pay_order_bad` (листинг 6.49).

Листинг 6.49. URL-маршрут pay_order_bad

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `pay_order_bad` (листинг 6.50).

Листинг 6.50. Представление pay_order_bad

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении формируются две текстовые переменные: `txt1` и `txt2`, которые через словарь `content` передаются в шаблон-форму pay_order_bad.html (листинг 6.51). После возврата из формы по POST-запросу делается возврат к маршруту `my_order` (переход к заказу).

Листинг 6.51. Шаблон-форма pay_order_bad.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне-форме выводится текстовое сообщение, а по нажатию кнопки `Вернуться к заказу` будет выполнен возврат к странице с текущим заказом.

6.5.10. Обработка успешной оплаты заказа банковской картой

В случае успешной оплаты заказа банковской картой пользователь должен получить соответствующее сообщение, а заказ — сменить статус с предварительного, на оплаченный. Разработаем набор модулей, которые будут обрабатывать успешную оплату заказа. В шаблоне my_order.html со ссылкой `Оплатить заказ (успешно)` связан следующий программный код:

```
<a href="{% url 'master:pay_order' %}">Оплатить заказ (успешно)</a>
```

По этой ссылке будет активирован URL-маршрут `pay_order` (листинг 6.52).

Листинг 6.52. URL-маршрут pay_order

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `pay_order` (листинг 6.53).

Листинг 6.53. Представление pay_order

Примечание.

В демо-версии книги удалены листинги программ.

Это представление имеет префикс `@transaction.atomic`, а это значит, что оно обертывается в транзакцию (вся цепочка операций по оплате заказа должна быть обязательно реализована без ошибок). Для этого необходимо произвести импорт пакета для выполнения транзакций:

```
from django.db import transaction
```

Кроме того, для этого представления необходимо также выполнить импорт и следующего пакета:

```
from django.http import HttpResponseRedirect
```

В представлении `pay_order` объект `user` получает параметры пользователя магазина. Затем делается проверка — есть ли активный заказ (в нашем приложении он всегда один). Если активного заказа нет, то просто вызывается шаблон с пустыми заказами пользователя. Если активный заказ есть, то он переводится в категорию оплаченных заказов (`order.status = 'paid'`), то есть совершенных покупок. После этого активируется . URL-маршрут `pay_order_ok` (листинг 6.54).

Листинг 6.54. URL-маршрут pay_order_ok

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `pay_order_ok` (листинг 6.55).

Листинг 6.55. Представление pay_order_ok

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении формируются две текстовые переменные: `txt1` и `txt2`, которые через словарь `content` передаются в шаблон-форму pay_order_ok.html (листинг 6.56). После возврата из формы по POST-запросу производится возврат к маршруту `user_shop_master` (переход к магазину пользователя).

Листинг 6.56. Шаблон-форма pay_order_ok.html

Примечание.

В демо-версии книги удалены листинги программ.

6.5.11. Обработка реальной оплаты заказа банковской картой на платежной платформе Robokassa

В шаблоне my_order.html для выполнения реальной оплаты заказа через платежную платформу Robokassa имеется кнопка `Оплатить заказ`. С этой кнопкой связана следующая ссылка:

```
<a href=«{{link}}» class=«btn btn-success my-1»>
    <i class=«bi bi-credit-card-2-front»></i> Оплатить заказ</a>
```

В этой ссылке присутствует переменная шаблона `{{link}}`, значение которой присваивается в представлении `my_order`:

```
link = rk.generate_payment_link(id_shop, pass_1, sum_to_pay, order.id, or-
der.num_order, is_test)
```

В этой инструкции на платежную платформу Robokassa через аргументы функции `rk.generate_payment_link` передаются следующие параметры:

- `id_shop` — идентификатор магазина на платежной платформе Robokassa;
- `pass_1` — пароль для входа в магазин на платежной платформе Robokassa;
- `sum_to_pay` — сумма к оплате (будет снята с банковской карты покупателя);
- `order.id` — идентификатор заказа в БД сайта магазина продавца;
- `order.num_order` — номер заказа на сайте магазина продавца;
- `is_test` — код типа оплаты (тестовая — `1` или реальная — `0`).

В итоге, по нажатию на кнопку **Оплатить заказ** покупатель будет перенаправлен на страницу с формой для ввода параметров банковской карты сайта платежной платформы Robokassa. После ввода параметров банковской карты и подтверждения оплаты на платежной платформе деньги будут сняты с банковской карты покупателя и переведены на счет магазина продавца. После этого пользователь будет возвращен на сайт магазина продавца в одно из представлений:

- в представление `pay_order` — в случае успешного завершения оплаты;
- в представление `pay_order_bad` — в случае ошибки при выполнении оплаты.

6.6. Проверка работы модулей оплаты предварительного заказа

Теперь посмотрим, как работают модули оплаты заказа.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 6** и перейти по ссылке **Мой магазин**.

Вернемся к странице магазина пользователя (рис. 6.41).

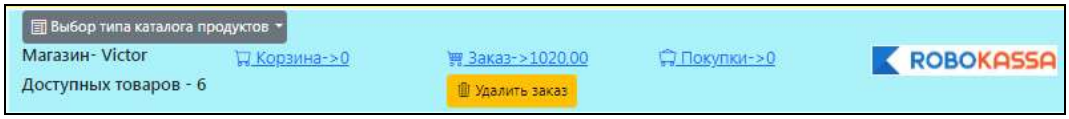


Рис. 6.41. Магазин пользователя со сформированным предварительным заказом

Если пользователь сформировал из корзины предварительный заказ, то в ссылке **Заказ** будет показана сумма предварительного заказа (в нашем случае: **Заказ->1020.00**). По щелчку левой кнопкой мыши на этой ссылке откроется шаблон my_order.html с информацией об элементах заказа и кнопками оплаты (рис. 6.42).

> ***Примечание***
> В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

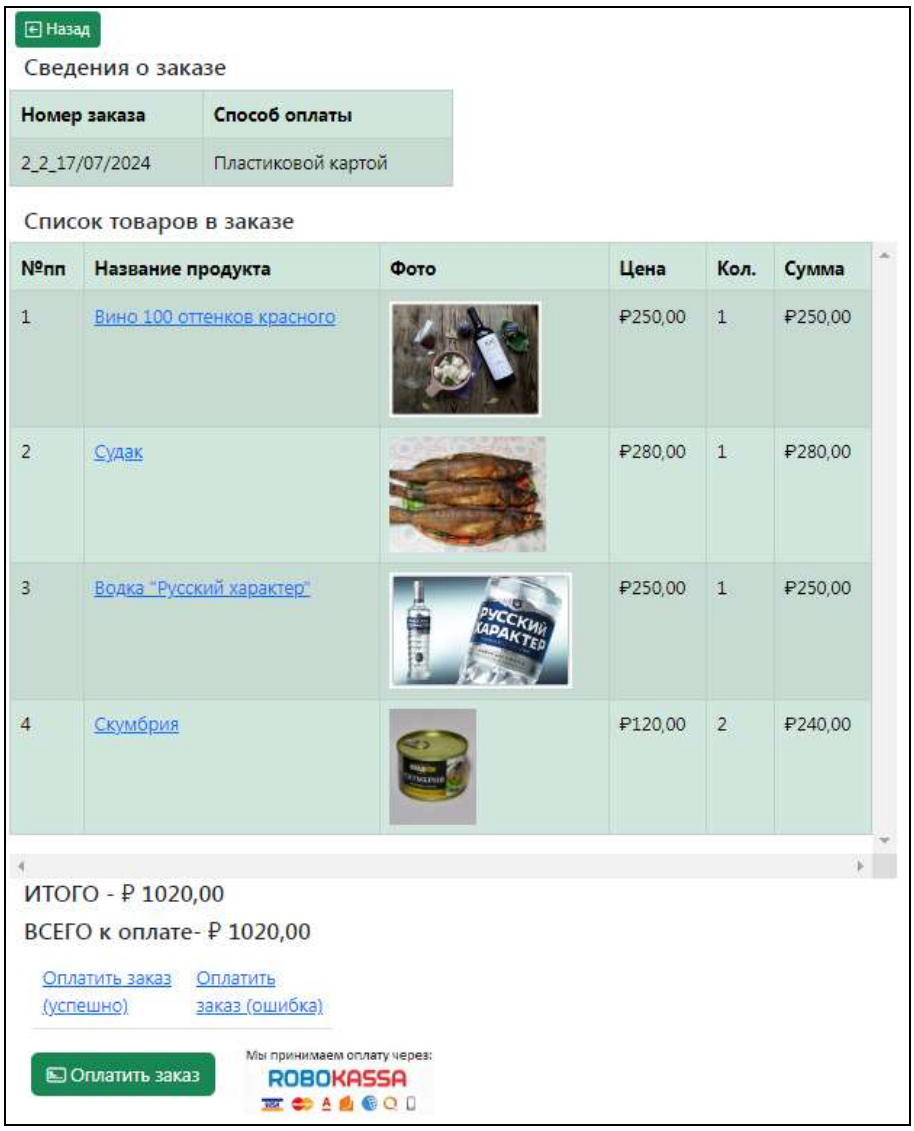


Рис. 6.42. Шаблон my_order.html с информацией об элементах заказа и кнопками оплаты

Здесь можно увидеть, что в предварительном заказе лежат продукты, которые пользователь ранее поместил в корзину, название продукта представляет собой ссылку на него, ниже таблицы с заказом расположены две ссылки: **Оплатить заказ (успешно)** и **Оплатить заказ (ошибка)**, а также кнопка **Оплатить заказ**.

Как уже ранее отмечалось, ссылки **Оплатить заказ (успешно)** и **Оплатить заказ (ошибка)** служат для проверки работы модулей, в которых выполняется обработка ответа платежной системы о том, что оплата прошла или не прошла соответственно. Эти ссылки нужно оставлять доступными только на этапе разработки и отладки работы программных модулей, а при размещении сайта на публичном сервере их следует удалить или закомментировать.

Так, при переходе по ссылке **Оплатить заказ (ошибка)** пользователь получит следующее сообщение (рис. 6.43).

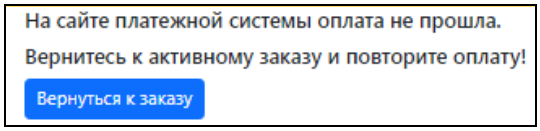


Рис. 6.43. Сообщение об ошибке при оплате заказа банковской картой

То есть, если на реальной платежной платформе произойдет сбой перевода денег с банковской карты покупателя на счет продавца, то покупатель будет направлен на страницу с этим сообщением. По нажатию на кнопку **Вернуться к заказу** покупатель вернется на страницу с текущим заказом.

Точно так же при переходе по ссылке **Оплатить заказ (успешно)** пользователь получит следующее сообщение (рис. 6.44).

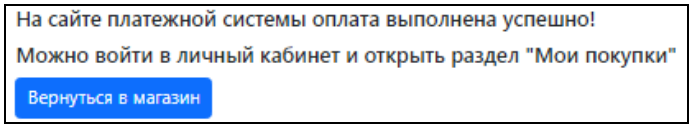


Рис. 6.44. Сообщение об успешной оплате заказа пластиковой картой

То есть, если на реальной платежной платформе произойдет корректный перевод денег с банковской карты покупателя на счет продавца, покупатель будет направлен на страницу с этим сообщением. По нажатию на кнопку **Вернуться в магазин**, покупатель вернется на страницу своего магазина, где сумма предварительного заказа обнулится, и заказ будет переведен в статус оплаченного (рис. 6.45).

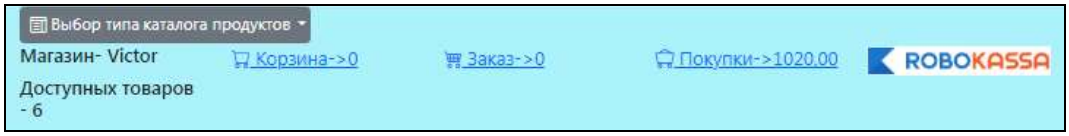


Рис. 6.45. Магазин покупателя после завершения оплаты предварительного заказа

Как здесь можно видеть, сумма предварительного заказа обнулилась, а сумма покупок появилась в ссылке **Покупки->1020.00**. Эта сумма автоматически подсчитывается в функции `sum_but_order` контекстного процессора и выводится в шаблоне user_shop_master.html.

По нажатию в шаблоне my_order.html (см. рис. 6.42) на кнопку **Оплатить заказ** пользователь будет направлен на сайт платежной платформы, и ему будет показана форма для ввода параметров банковской карты (рис. 6.46).

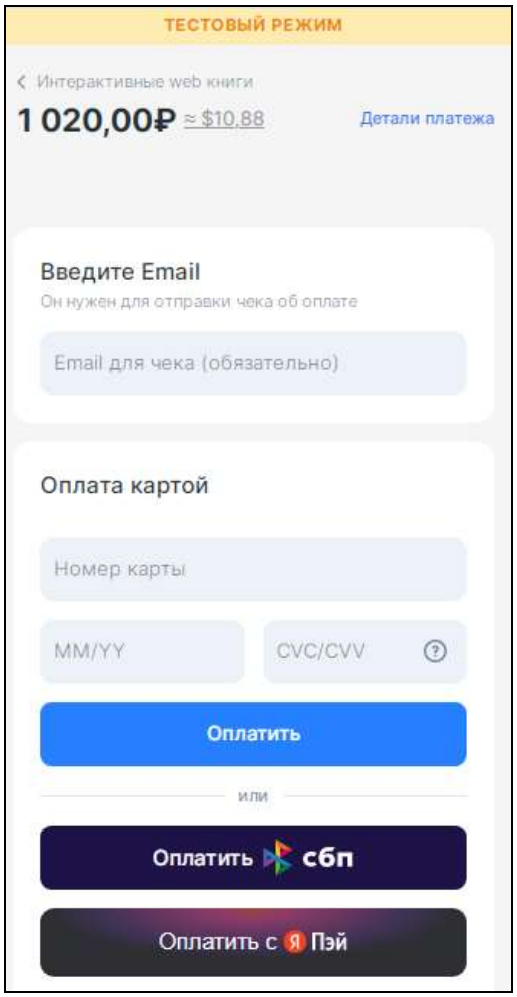


Рис. 6.46. Форма для ввода параметров банковской карты на платежной платформе Robokassa (тестовая оплата)

В этой форме покупатель должен заполнить необходимые поля (ввести адрес своей электронной почты и параметры банковской карты). По нажатию на кнопку **Оплатить** деньги с карты покупателя будут списаны и зачислены на счет магазина продавца, а покупатель будет возвращен на сайт магазина. Обратите внимание, что в показанном случае используется тестовый режим — то есть идет проверка и отладка работы программных модулей, а реального списания денег со счета покупателя не происходит.

При размещении сайта на публичном сервере вам понадобится сменить режим оплаты заказа с тестового на режим реальной оплаты. Для это надо в файле settings.py изменить признак:

```
IS_TEST = 0  # Реальная оплата
```

После этого форма для ввода параметров банковской карты на сайте Robokassa приобретет следующий вид (рис. 6.47).

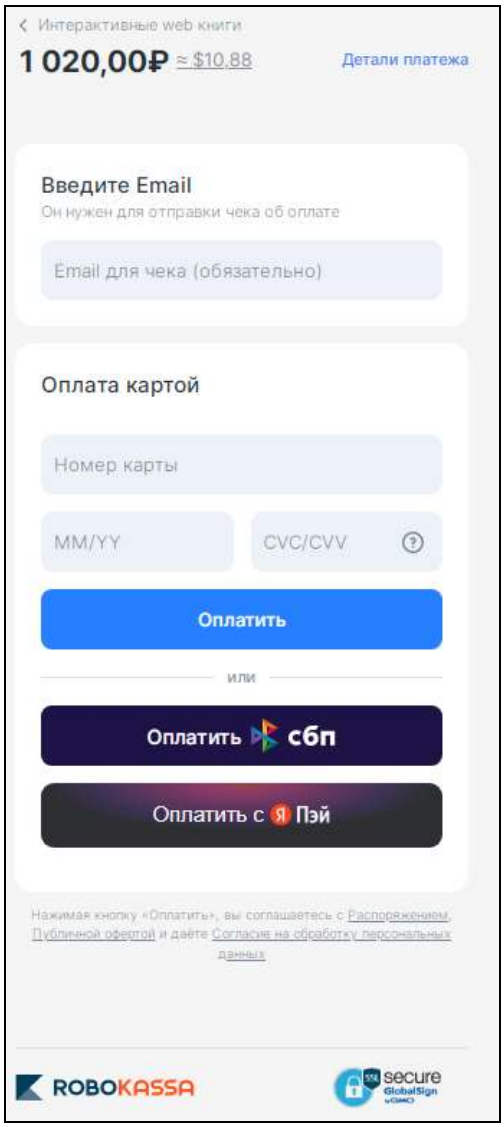


Рис. 6.47. Форма для ввода параметров банковской карты на платежной платформе Robokassa (реальная оплата)

При переходе на странице магазина (см. рис. 6.45) по ссылке **Покупки** пользователь получит информацию обо всех оплаченных заказах. Если оплаченных заказов нет, то он должен получить следующую страницу (рис. 6.48).

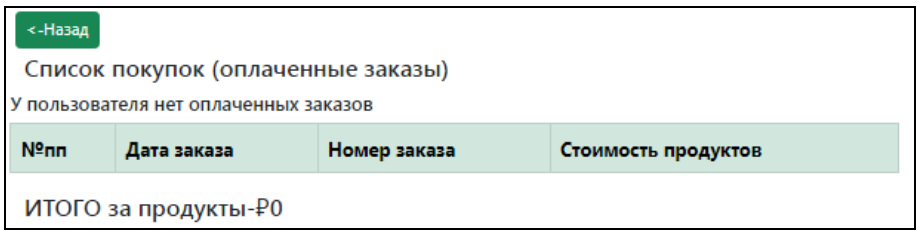


Рис. 6.48. Страница для вывода пустого списка оплаченных заказов

Впрочем, вы в своем проекте при переходе по этой ссылке можете получить ошибку, поскольку мы еще не создали набор модулей для показа оплаченных заказов (покупок), — этому вопросу посвящен следующий раздел.

6.7. Модули для показа оплаченных заказов (покупок)

В шаблоне магазина пользователя для показа оплаченных заказов покупателя используется следующая ссылка:

```
<a href="{% url 'master:my_buy' %}">
    <i class="bi bi-minecart-loaded"></i> Покупки->{{sum_my_buy}}</a>
```

По этой ссылке активируется URL-маршрут `my_buy` (листинг 6.57).

Листинг 6.57. URL-маршрут my_buy

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `my_buy`, обеспечивающее просмотр всех оплаченных заказов заказа (листинг 6.58).

Листинг 6.58. Представление my_buy

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении объект `user` получает текущего пользователя, после чего в объект `orders` из БД загружаются все его оплаченные заказы. Затем формируется словарь `context`, принимающий список оплаченных заказов, и этот словарь передается в шаблон my_buy.html (листинг 6.59).

Листинг 6.59. Шаблон my_buy.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне выводится таблица со списком оплаченных заказов, где номер заказа представлен в виде ссылки:

```
<a href="{% url 'master:buy_order_list' id=obj.id %}">{{obj.num_order}}</a>
```

По этой ссылке активируется URL-маршрут `buy_order_list` (листинг 6.60), в который передается идентификатор оплаченного заказа (`id=obj.id`).

Листинг 6.60. URL-маршрут buy_order_list

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `buy_order_list` (листинг 6.61), в которое передается идентификатор оплаченного заказа (`int:id`).

Листинг 6.61. Представление buy_order_list

Примечание.

В демо-версии книги удалены листинги программ.

Это представление получает идентификатор оплаченного заказа (`id`), и реквизиты текущего заказа загружаются в объект `order`. Далее выполняется расчет суммы оплаты по элементам этого заказа (`s`), и в объект `products` загружается список приобретенных продуктов. Сведения о продуктах в оплаченном заказе упаковываются в объект `mylist`. Все полученные параметры загружаются в контекстный словарь `context`, который передается в шаблон buy_order_list.html (листинг 6.62).

Листинг 6.62. Шаблон buy_order_list.html

Примечание.

В демо-версии книги удалены листинги программ.

Этот шаблон представляет собой таблицу, в которой показан список продуктов, находящихся в оплаченном заказе. Наименование товара представлено в нем в виде ссылки, при переходе по которой будет выведена полная информация о приобретенном продукте.

6.8. Проверка работы модулей для показа оплаченных заказов (покупок)

Теперь посмотрим, как работают модули показа оплаченных заказов.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 6** и перейти по ссылке **Мой магазин**.

Вернемся к странице магазина пользователя (рис. 6.49).

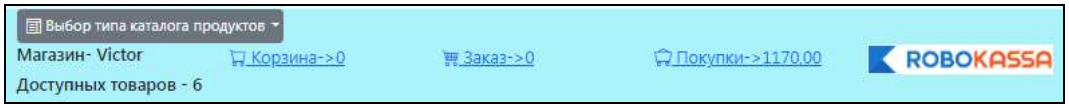


Рис. 6.49. Страница магазина покупателя при наличии оплаченных заказов

Если пользователь имеет оплаченные заказы (то есть он совершал реальные покупки), в ссылке **Покупки** будет показана сумма по всем оплаченным заказам (в нашем случае: **Покупки->1170.00**). Эта сумма автоматически подсчитывается в функции `sum_but_order` контекстного процессора и выводится в шаблоне user_shop_master.html. По щелчку левой кнопкой мыши на этой ссылке откроется шаблон my_buy.html со списком оплаченных заказов (рис. 6.50).



Рис. 6.50. Страница магазина покупателя со списком оплаченных заказов

Примечание

В интерактивной цифровой книге таблица будет заполнена тестовыми данными. При этом, чтобы читатели интерактивной цифровой книги не испортили тестовые данные, на публичном сервере в модулях корректировки данных строки программ, обновляющие записи в БД, закомментированы.

Как здесь можно видеть, в нашем случае покупатель имеет два оплаченных заказа, и по щелчку левой кнопкой мыши на номере заказа откроется шаблон by_order_list.html со списком продуктов, находящихся в заказе (рис. 6.51).

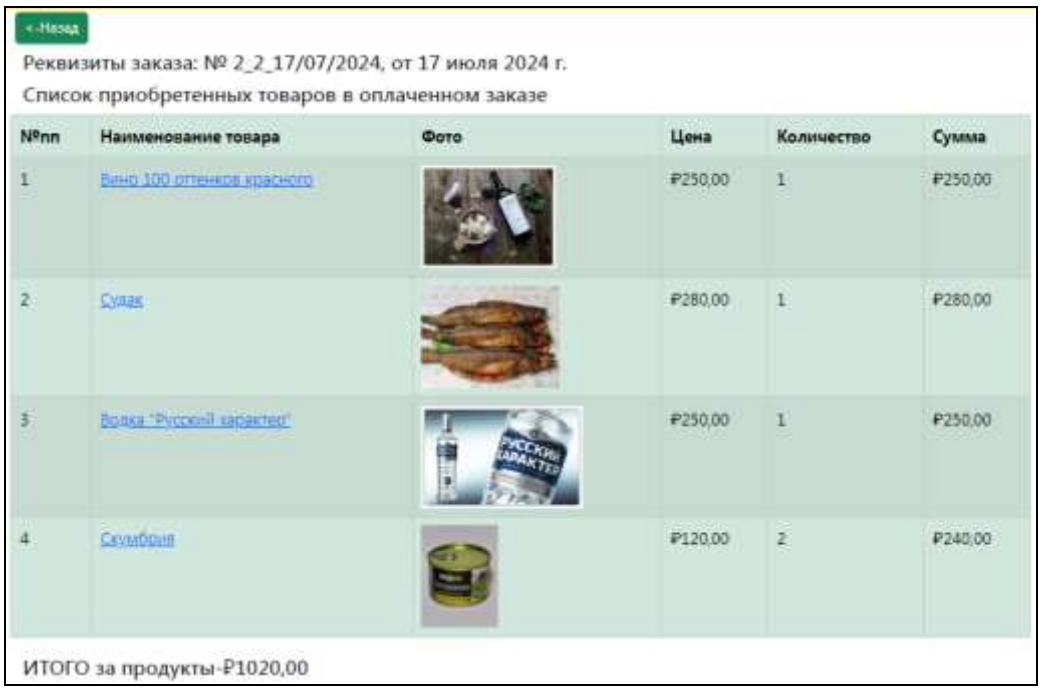


Рис. 6.51. Страница магазина покупателя со списком оплаченных заказов

Наименование товара здесь представлено в виде ссылки, и если щелкнуть на ней левой кнопкой мыши, откроется шаблон product_detail.html, демонстрирующий подробные сведения о продукте (рис. 6.52).



Рис. 6.52. Шаблон product_detail.html с подробными сведениями о продукте

6.9. Краткие итоги

В этой главе представлен набор связанных программных модулей, из которых, как из кирпичиков, можно собирать полноценные сайты электронной торговли. Вы можете скачать коды представленных здесь примеров и использовать их в своих проектах (новостные сайты, блоги, интернет-магазины и т. п.).

Теперь можно перейти к следующей главе, в которой приведены некоторые полезные советы и приемы автора и сообщества программистов, практикующих создание приложений на Django.

Глава 7. Полезные советы

В этой главе приведены некоторые полезные советы автора и сообщества программистов, работающих с Django. Достаточно часто в процессе работы над проектом разработчик становится в тупик из-за того, что по всем канонам работающий модуль выдает ошибку. Как правило, это некая мелочь типа: не там стоит точка или запятая, отсутствует или, наоборот, присутствует лишний знак /, вместо знака / должно стоять двоеточие и т. п. В море документации найти ответ на встретившуюся проблему порой бывает затруднительно и приходится искать ответ в Интернете у тех, кто с подобной проблемой сталкивался. В любом случае — это потеря времени. Поэтому здесь приводятся варианты решения некоторых таких проблем с демонстрацией их в виде работающих программных модулей.

Из материалов этой главы вы узнаете:

- как создать документ формата PDF, чтобы на сайте у него присутствовало оглавление;
- как на HTML-странице показать:
  - документ формата PDF из представления;
  - документ формата PDF из шаблона Django;
  - документ PDF в формате флеш-анимации;
- как переводить страницы сайта на другие языки;
- как отправлять с сайта почтовые сообщения;
- как использовать Java-скрипты в шаблонах Django;
- как использовать фреймворк PyScript в шаблонах Django;
- как использовать пакет Brython для выполнения на HTML-страницах программ на Python;
- как правильно откатить миграцию, не нарушив целостность БД.

7.1. Показываем на сайте документ PDF

Весьма часто на станицах сайта требуется показать документ в формате PDF: цифровую электронную книгу, инструкцию к приложению, методическое пособие и т. п. Традиционно пользователи готовят исходный документ в текстовом редакторе Word с сохранением его в формат PDF. При этом очень важно задать при сохранении документа нужные параметры — особенно, в том случае, когда он достаточно большой и имеет оглавление.

7.1.1. Создание документа в формате PDF

Если в редакторе Word сохранить документ в формате PDF с параметрами по умолчанию, то при выводе такого документа на сайте у него не будут отображаться опции оглавления. Поэтому, чтобы они всегда отображались, в редакторе Word перед сохранением документа в формате PDF нужно выполнить следующие действия:

7. В меню **Файл** выбрать опцию **Сохранить как** (рис. 7.1).

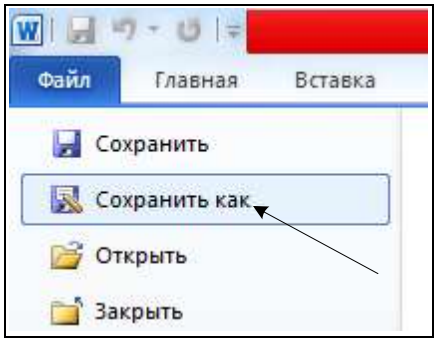


Рис. 7.1. Опция **Сохранить как** в меню **Файл** редактора Word

8. В открывшемся окне **Сохранение документа** (рис. 7.2) выбрать в разделе **Тип файла** опцию **PDF (*.pdf)**.

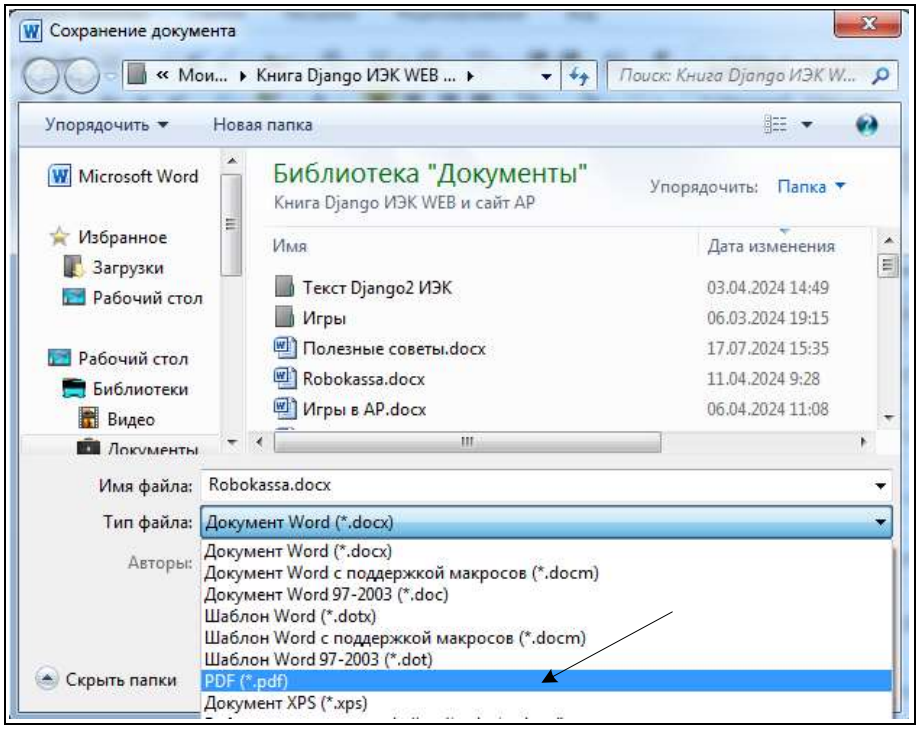


Рис. 7.2. Опция **PDF (*.pdf)** в разделе **Тип файла** окна **Сохранение документа** редактора Word

9. В открывшейся в нижней части окна **Сохранение документа** области (рис. 7.3) нажать на кнопку **Параметры**.

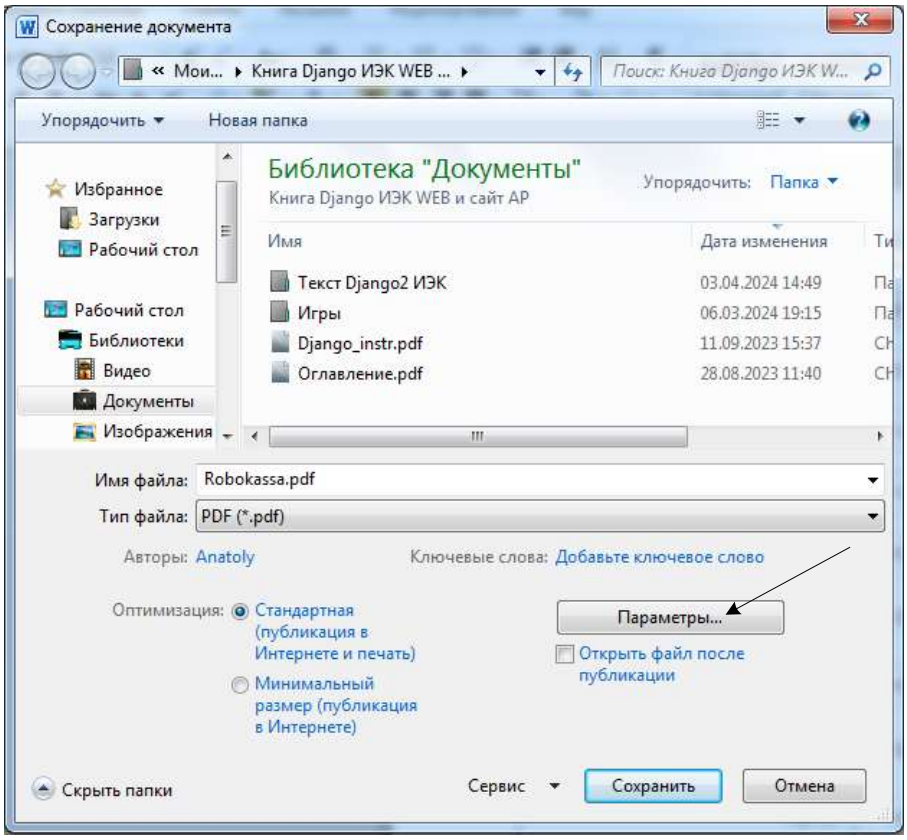


Рис. 7.3. Кнопка **Параметры** в окне сохранения документа редактора Word

10. В открывшейся панели **Параметры** (рис. 7.4) установить флажок **Создать закладки, используя:**, включить опцию **заголовки** и нажать на кнопку **OK**.

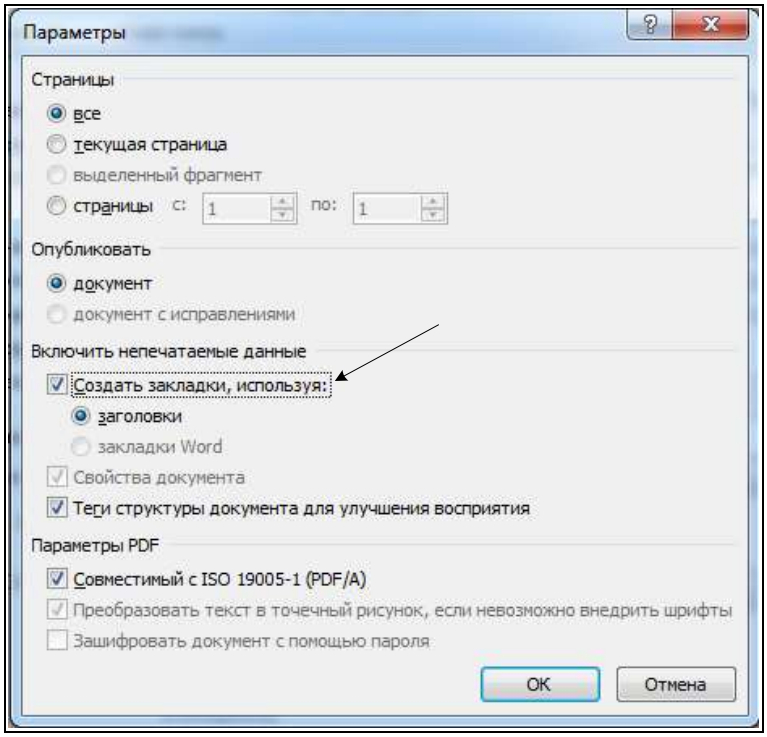


Рис. 7.4. Окно для задания дополнительных параметров сохранения документа редактора Word

11. Вернувшись в окно **Сохранение документа** (см. рис. 7.3), выбрать одну из опций оптимизации:
    - Стандартная (публикация в Интернете и печать);
    - Минимальный размер (публикация в Интернете)

    и нажать кнопку **Сохранить**.

В результате будет создан файл документа в формате PDF, предусматривающий показ его оглавления.

7.1.2. Вывод документа формата PDF из представления

Предположим, что PDF-файл с именем Python_for_kid_demo.pdf загружен на сайт в каталог static/pdf/, и нам необходимо вывести содержимое этого документа, указав его параметры в представлении.

Прежде всего, в главном меню или на одной из страниц сайта следует указать ссылку на маршрут, через который будет вызвано это представление:

```
<a href="{% url 'master:pdf_view'%}" target ="_blank">Показать PDF из представления</a>
```

По этой ссылке будет активирован URL-маршрут `pdf_view` (листинг 7.1).

Листинг 7.1. URL-маршрут pdf_view

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `pdf_view` (листинг 7.2).

Листинг 7.2. Представление pdf_view

Примечание.

В демо-версии книги удалены листинги программ.

Для работы этого представления выполнен импорт необходимых библиотек, а в самом представлении создан объект `filepath`, в котором указан полный путь к PDF-файлу. Отображение PDF-файла на новой вкладке браузера осуществляется с использованием функции Django `FileResponse()`. Обратите внимание, что шаблон для отображения PDF-файла в этом случае не создается.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Показать PDF из представления**.

Если запустить на выполнение созданные в этом разделе программные модули, то в окне браузера страница с содержимым PDF-файла будет выглядеть так, как показано на рис. 7.5.

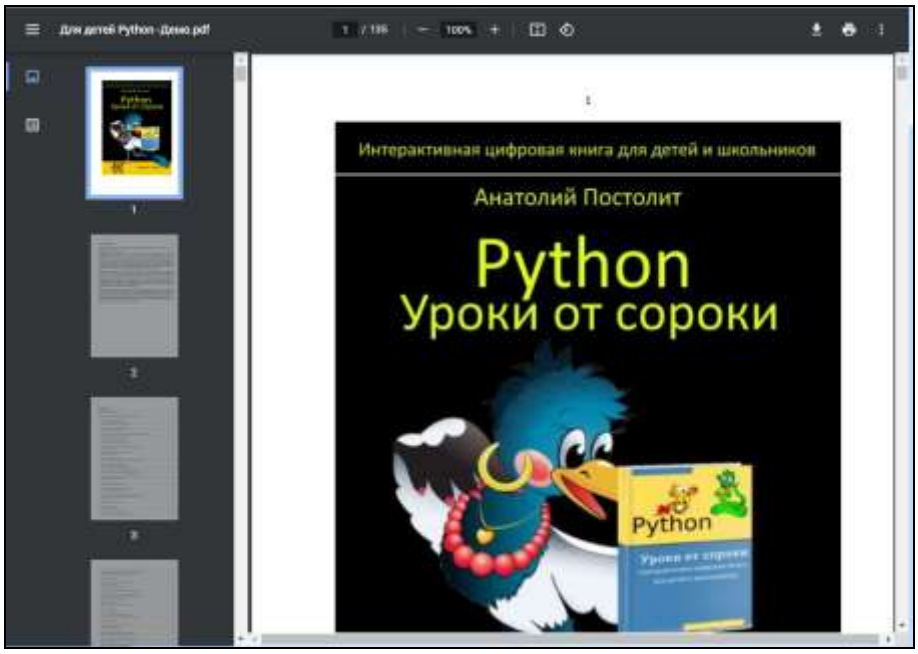


Рис. 7.5. Вкладка браузера с содержимым PDF-файла, выведенного из представления

Как здесь можно видеть, страница с содержимым PDF-файла имеет титульную строку и две панели: левую и центральную. В левом углу титульной строки расположена кнопка — она позволяет скрыть/показать левую боковую панель, на которой находятся миниатюры страниц PDF-документа и две кнопки: верхняя кнопка — предоставляет возможность отобразить миниатюры страниц, нижняя кнопка — показывает оглавление документа. С помощью левой боковой панели — либо из миниатюр страниц, либо из оглавления — можно быстро выбрать нужную страницу документа. Так, при выборе из оглавления какого-либо раздела документа, в центральной панели сразу открывается первая страница выбранного раздела (рис. 7.6).

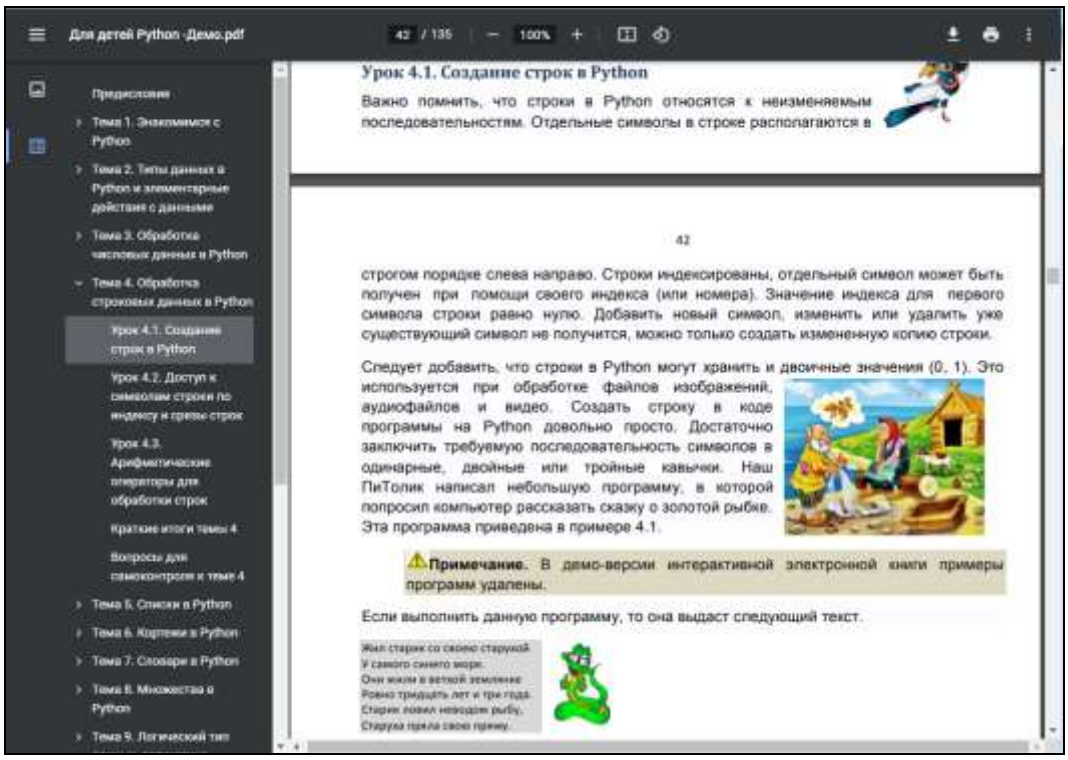


Рис. 7.6. Быстрое пролистывание содержимого документа с использованием оглавления

В центре титульной строки имеются две кнопки: кнопка ➕ — позволяет увеличить масштаб отображаемой страницы, а кнопка ➖ — его соответственно уменьшить. Правее находятся еще две кнопки: первая ⬜ — позволяет раскрыть страницу на всю ширину вкладки и свернуть ее до минимального размера, вторая 🔄 — поворачивает страницу на угол 90 градусов.

В правом углу титульной строки находятся еще три кнопки: ⬇ — скачать файл с документом, 🖨 — вывести текст документа на печать, ⋮ — выполнить еще ряд действий, предусмотренных этим меню.

Левая и центральная панель оснащены полосами прокрутки, которые позволяют быстро прокручивать страницы текста.

7.1.3. Вывод документа формата PDF из шаблона Django

Предположим, что PDF-файл с именем Python_for_kid_demo.pdf загружен на сайт в каталог static/pdf/, и нам необходимо вывести содержимое этого документа, указав его параметры в шаблоне.

Прежде всего, в главном меню или на одной из страниц сайта следует указать ссылку на маршрут, через который в конечном итоге будет вызван этот документ:

```
<a href="{% url 'master:pdf_template'%}" target ="_blank">Показать PDF из шаблона</a>
```

По этой ссылке будет активирован URL-маршрут `pdf_template` (листинг 7.3).

Листинг 7.3. URL-маршрут pdf_template

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `pdf_template` (листинг 7.4).

Листинг 7.4. Представление pdf_template

Примечание.

В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон pdf_template.html (листинг 7.5).

Листинг 7.5. Шаблон pdf_template.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне создан тег `object`, в атрибуте `data` которого указаны следующие параметры:

- URL-адрес, по которому на сайте находится документ в виде PDF-файла: `{% static "pdf/Python_for_kid_demo.pdf" %}`;
- тип выводимого документа: `type="application/pdf"`;
- ширина и высота окна в пикселах, в котором будет показан документ: `width="100%" height="1000"`).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Показать PDF из шаблона**.

Если запустить на выполнение созданные в этом разделе программные модули, то в окне браузера страница с содержимым PDF-файла будет выглядеть так, как показано на рис. 7.7.

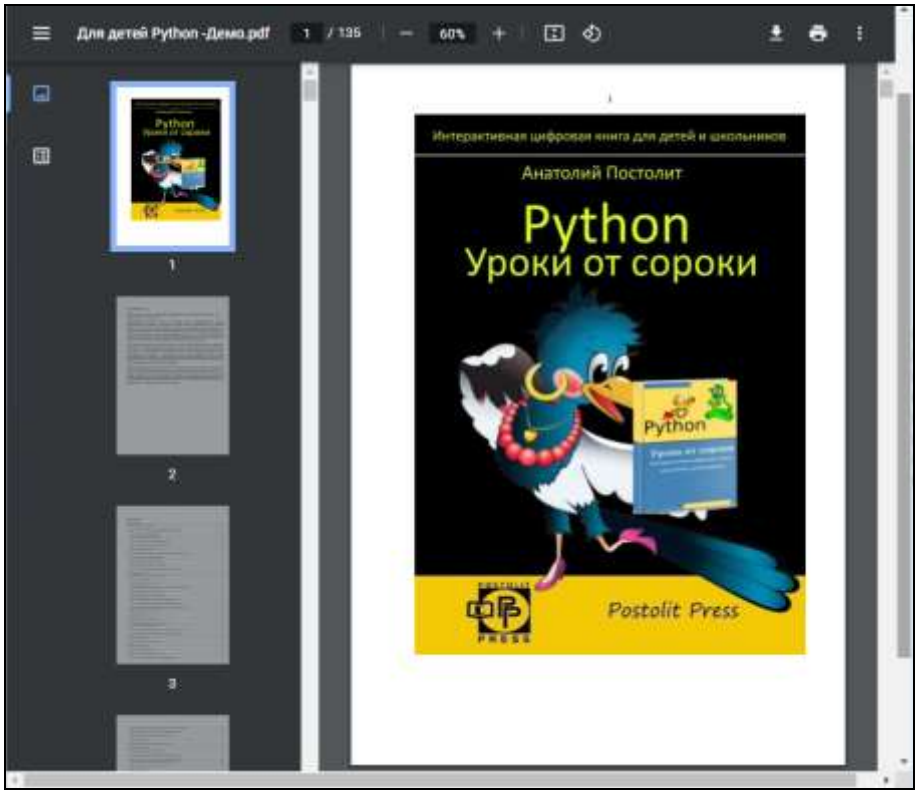


Рис. 7.7. Вкладка браузера с содержимым PDF-файла, выведенного из шаблона

В тех случаях, когда файл с PDF-документом загружен на сайт через административную панель или через интерфейс, и его URL-адрес хранится в БД в одной из моделей, то вывести его на вкладку браузера можно с использованием в шаблоне следующего кода (в этом примере URL-адрес файла с PDF-документом хранится в поле с именем `file_pdf` модели `book`):

```
<object
    data='{{book.file_pdf.url}}'
```

```
        type="application/pdf" width="100%" height="1000">
</object>
```

7.1.4. Вывод PDF-документа в формате флеш-анимации

Любой документ, который хранится в формате PDF-файла, можно конвертировать в файл с флеш-анимацией (в документ формата SWG или формата Canvas HTML5). Такой документ создается с помощью специального инструментария Flip PDF Professional и имеет более привлекательный вид и расширенную функциональность. Внешне документ с флеш-анимацией выглядит как книга или журнал с цветными иллюстрациями — его можно просматривать, перелистывая страницы или быстро перемещаясь по оглавлению. Кроме того, на любую его страницу можно встроить аудио- или видеофайл. Документ в формате флеш-анимации имеет серьезную защиту от незаконного копирования, поскольку его содержимое невозможно скопировать в буфер обмена. Конвертацию PDF-документа в формат флеш-анимации мы здесь рассматривать не станем (для этого надо обратиться к документации по приложению Flip PDF Professional), а сосредоточимся на том, как такой документ можно отобразить в шаблоне Django.

В качестве примера мы возьмем интерактивную цифровую книгу «Детям о птицах». Эта книга подготовлена в текстовом редакторе Word, сохранена в формате PDF, а затем с использованием приложения Flip PDF Professional, конвертирована в формат Canvas HTML5 (рис. 7.8).

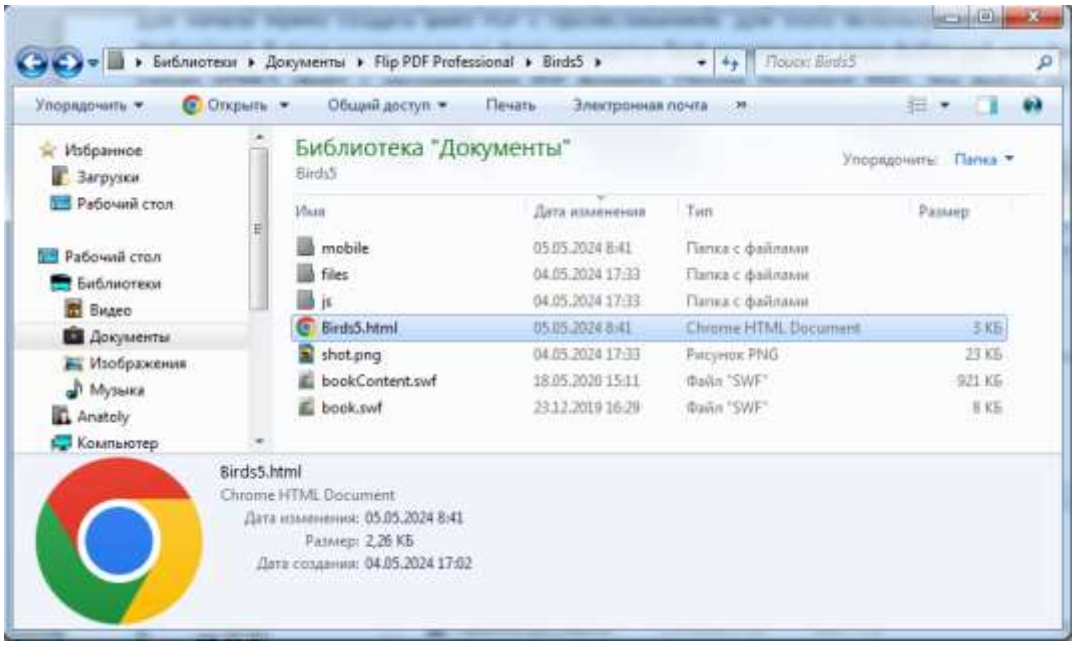


Рис. 7.8. Каталог **Birds5** с цифровой интерактивной книгой «Детям о птицах»

Стартовым для запуска анимированной книги в веб-браузере является находящийся в каталоге Birds5 файл Birds5.html — именно его нужно указывать в URL-адресе, когда книгу необходимо открыть на сайте. Этот файл сопровождается другими файлами с изображениями и Java-скриптами. Все их нужно перенести на сайт — то есть, в нашем случае, на сайт нужно скопировать весь каталог Birds5 — с использованием FTP-терминала выложить этот каталог в каталог static на сервере (рис. 7.9).

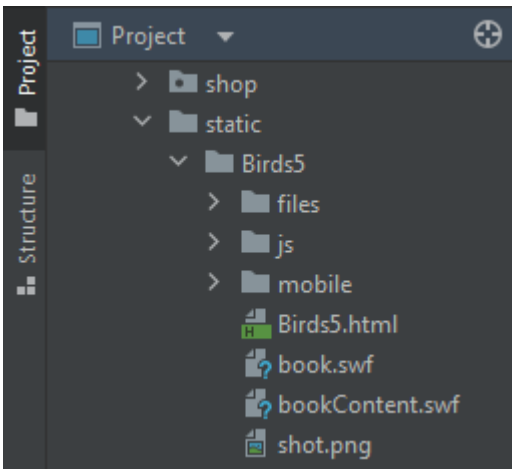


Рис. 7.9. Каталог **Birds5** с цифровой интерактивной книгой «Детям о птицах» помещен в каталог **static** веб-приложения

Чтобы открыть документ формата Canvas HTML5, надо в главное меню или в любой другй шаблон добавить его URL-адрес: static/Birds5/Birds5.html.

В нашей интерактивной цифровой книге для вызова этого документа создадим ссылку:

```
<a href="{% url 'master:pdf_animate'%}" target ="_blank">Показать PDF с анимацией</a>
```

По этой ссылке будет вызван URL-маршрут `pdf_animate` (листинг 7.6).

Листинг 7.6. URL-маршрут pdf_animate

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту будет вызвано представление `pdf_animate` (листинг 7.7).

Листинг 7.7. Представление pdf_animate

Примечание.

В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон pdf_animate.html (листинг 7.8).

Листинг 7.8. Шаблон pdf_animate

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне создан тег `object`, в атрибуте `data` которого указаны следующие параметры:

- URL-адрес, по которому на сайте находится документ в виде анимационного PDF-файла: `{% static "Birds5/Birds5.html" %}`;
- тип выводимого документа: `type="application/pdf"`;
- ширина и высота окна в пикселах, в котором будет показан документ: `width="100%" height="1000"`.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Показать PDF с анимацией**.

Если запустить на выполнение созданные в этом разделе программные модули, то в окне браузера страница с содержимым анимационного файла формата Canvas HTML5 будет выглядеть так, как показано на рис. 7.10, — документ здесь представлен в виде книги, которую можно перелистывать (рис. 7.11).



Рис. 7.10. Документ Birds5.html в шаблоне pdf_animate.html



Рис. 7.11. Документ Birds5.html с раскрытыми страницами

На второй странице книги имеется оглавление — кроме него, в нее, как показано на рис. 7.11, встроен также фрейм с видеофайлом. Этот видеофайл допускается просмотреть, не покидая страницу, а оглавление представляет собой набор ссылок, щелкая на которых, можно сразу переходить в соответствующие разделы книги. Когда открывается первая страница раздела с тем или иным видом птицы, автоматически проигрывается аудиофайл с ее голосом.

На нижней панели имеется ряд кнопок: перелистнуть страницу, перейти на первую (последнюю) страницу, показать оглавление в дополнительной вкладке (рис. 7.12) — с его помощью также можно переходить от раздела к разделу.



Рис. 7.12. Переход на страницу **Ворона** по оглавлению на левой вкладке

В момент перелистывания страницы выполняется анимация перелистывания — тем самым имитируется листание страниц бумажной книги (рис. 7.13).



Рис. 7.13. Анимация перелистывания страниц книги

В нижней строке с функциональными кнопками имеется кнопка, которая формирует и показывает QR-код (рис. 7.14). По этому коду можно загрузить документ на смартфон и просматривать его на мобильном устройстве.



Рис. 7.14. QR-код для загрузки цифрового анимационного документа на мобильное устройство

7.2. Переводим страницы сайта на другие языки

Самый простой способ обеспечить быстрый перевод страниц вашего сайта на другие языки — использовать виджет Google Переводчик. Этот виджет позволяет переводить страницы сайта на более чем 100 языков мира, а для его подключения к сайту достаточно написать всего несколько строк программного кода.

Лучше всего загрузить виджет для выбора языка перевода на базовую страницу сайта (base.html) — в этом случае переводчик будет доступен на всех других страницах, поскольку, как правило, они являются дочерними по отношению к базовой странице.

Прежде всего, к базовой странице сайта надо подключить скрипты, которые обеспечат загрузку переводчика Google:

```
<!-- Скрипты переводчика начало -->
<script type="text/javascript">
     function googleTranslateElementInit() {
      new google.translate.TranslateElement({pageLanguage: 'ru'},
'google_translate_element');
      }
</script>
<script type="text/javascript"

src="//translate.google.com/translate_a/element.js?cb=googleTranslateElementInit">
</script>
<!-- Скрипты переводчика конец -->
```

Затем поместить сам виджет (выпадающий список для выбора языка перевода) в один из элементов страницы — в главное меню, в титульную строку или в «подвал» главной страницы:

```
<div id="google_translate_element" class="text-center"></div>
```

В нашей интерактивной цифровой книге для демонстрации работы переводчика Google создана следующая ссылка:

```
<a href="{% url 'master:g_translate'%}" target ="_blank">Перевод страниц
сайта</a>
```

По этой ссылке активируется URL-маршрут `g_translate` (листинг 7.9).

Листинг 7.9. URL-маршрут g_translate

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `g_translate` (листинг 7.10).

Листинг 7.10. Представление g_translate

Примечание.

В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон g_translate.html (листинг 7.11).

Листинг 7.11. Шаблон g_translate.html

Примечание.

В демо-версии книги удалены листинги программ.

В этот демонстрационный шаблон подгружены необходимые скрипты, вставлен виджет для выбора языка перевода и добавлен простой текст, который должен быть переведен.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Перевод страниц сайта**.

Если запустить на выполнение созданные в этом разделе программные модули, то в окне браузера страница с переводчиком Goggle будет выглядеть так, как показано на рис. 7.15.



Рис. 7.15. Демонстрационная страница с переводчиком Google (по умолчанию установлен русский язык)

Здесь по умолчанию установлен русский язык, но пользователь может из выпадающего списквв выбрать и другой язык — например, английский (рис. 7.16). После этого текстовое содержание всех страниц сайта будет переведено на английский язык (рис. 7.17).

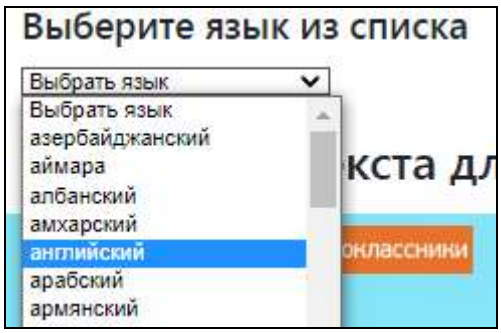


Рис. 7.16. Выбор из списка английского языка для перевода страниц сайта



Рис. 7.17. Демонстрационная страница переведена переводчиком Google на английский язык

Выбрав из выпадающего списка какой-нибудь экзотический язык — например, арабский, мы получим страницу, показанную на рис. 7.18, а выбор из списка китайского языка, на котором разговаривают порядка полутора миллиарда человек, приведет нас на страницу, показанную на рис. 7.19.



Рис. 7.18. Демонстрационная страница переведена переводчиком Google на арабский язык

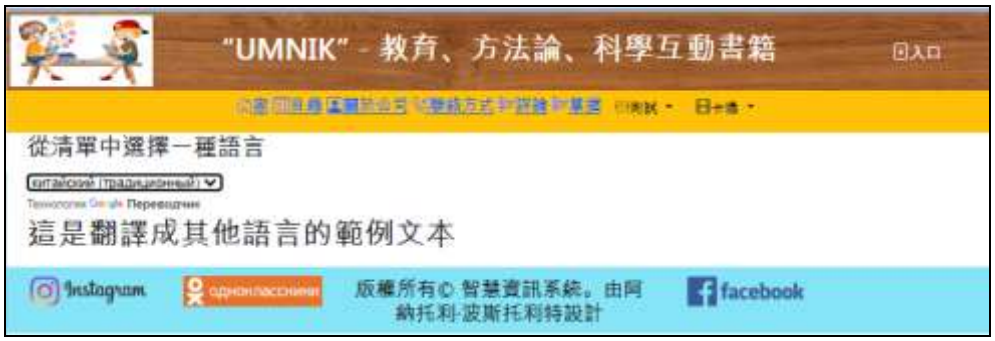


Рис. 7.19. Демонстрационная страница переведена переводчиком Google на китайский язык

Обратите внимание, что на тот или иной язык переводится не только демонстрационная страница, но и все другие страницы сайта.

7.3. Отправляем с сайта почтовые сообщения

При создании сайтов достаточно часто требуется отправлять пользователям почтовые сообщения — например, подтвердить оплату заказа банковской картой, сообщить о выезде курьера для доставки товара, информировать о новых услугах и т. п. Такие сообщения должны формироваться и отправляться пользователям сайта автоматически в фоновом режиме. Для реализации указанных функций нужно подключить к сайту и настроить почтовый сервис. Выбор почтового сервиса, который будет использоваться для отправки почты, предоставляется программисту. Это могут быть сервисы **mail.ru**, **yandex.ru**, **gmail.com** и пр. Кроме того, вы можете задействовать почтовую службу провайдера, на серверной площадке которого размещен ваш сайт. Поскольку сайт с нашей интерактивной цифровой книгой размещен на серверной площадке российского провайдера Timeweb, в этом разделе мы рассмотрим настройку отправки почты с помощью его почтового сервиса. Но для начала разберемся с протоколом почтовых сообщений.

> ***Протокол SMTP***
> SMTP (Simple Mail Transfer Protocol) — это протокол, который используется для отправки электронной почты через Интернет. Если вы хотите отправлять электронные письма со своего веб-сайта или из приложения, вам необходимо правильно настроить SMTP-сервер. Timeweb является одним из популярных хостинг-провайдеров, и в этом разделе мы рассмотрим, как настроить SMTP-протокол Timeweb.

Прежде всего, для работы по SMTP-протоколу Timeweb необходимо создать на своем домене почтовый ящик. Это можно сделать через панель управления Timeweb. Создайте новый ящик, выбрав адрес электронной почты, пароль и прочие настройки по своему усмотрению. Убедитесь, что вы запомнили данные для доступа к этому почтовому ящику, так как они понадобятся вам позже.

Создав почтовый ящик, перейдите к настройке SMTP-протокола Timeweb на своем веб-сайте или в приложении. Вам понадобятся следующие данные: адрес SMTP-сервера (обычно **smtp.timeweb.ru**), порт (обычно 25 или 587), адрес вашей электронной почты и пароль доступа к нему. В некоторых случаях может понадобиться указать SSL- или TLS-соединение для безопасной отправки почты.

В каждом веб-сайте или приложении настройка SMTP-протокола может выполняться поразному, поэтому следуйте инструкциям, предоставленным разработчиками программного обеспечения. Обычно вам достаточно найти раздел настроек электронной почты и ввести все данные, полученные на предыдущем шаге.

Завершив настройку проверьте работу SMTP-протокола Timeweb, отправив тестовое письмо, например, на адрес своей электронной почты. Убедитесь, что письмо успешно отправлено и доставлено адресату. Если возникнут проблемы, проверьте правильность введенных данных и свяжитесь со службой поддержки Timeweb для получения дополнительной помощи.

7.3.1. Регистрация в почтовом сервисе Timeweb

Чтобы начать работу с почтовым сервисом Timeweb, необходимо в нем зарегистрироваться. Это можно сделать, выполнив несколько простых шагов:

1. Откройте официальный сайт Timeweb[1] и перейдите в раздел **Почта** личного кабинета на сайте (рис. 7.20).
2. Выберите на открывшейся странице (рис. 7.21) тот домен, для которого требуется настроить почту (в нашем случае **rumnik.ru**), и перейдите по ссылке **Добавить**.
3. В открывшемся окне для создания нового почтового ящика (рис. 7.22) задайте адрес электронной почты отправителя (вашего сайта), пароль для отправки почтовых сообщений и нажмите на кнопку **Создать ящик** — вам будет показано окно сформированного почтового ящика (рис. 7.23).

[1] См. **https://hosting.timeweb.ru/**.

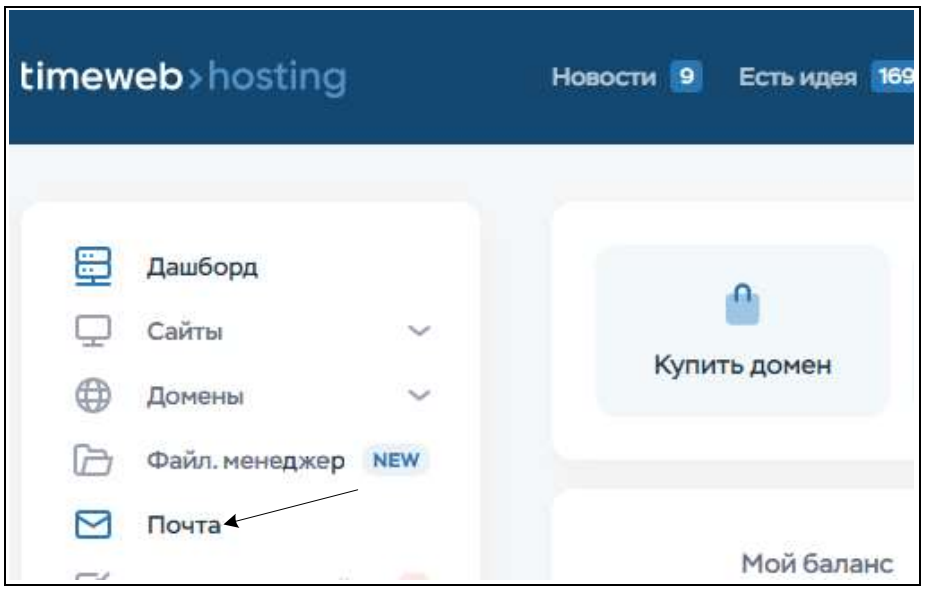


Рис. 7.20. Раздел **Почта** личного кабинета на сайте провайдера Timeweb



Рис. 7.21. Добавление почтового ящика к домену

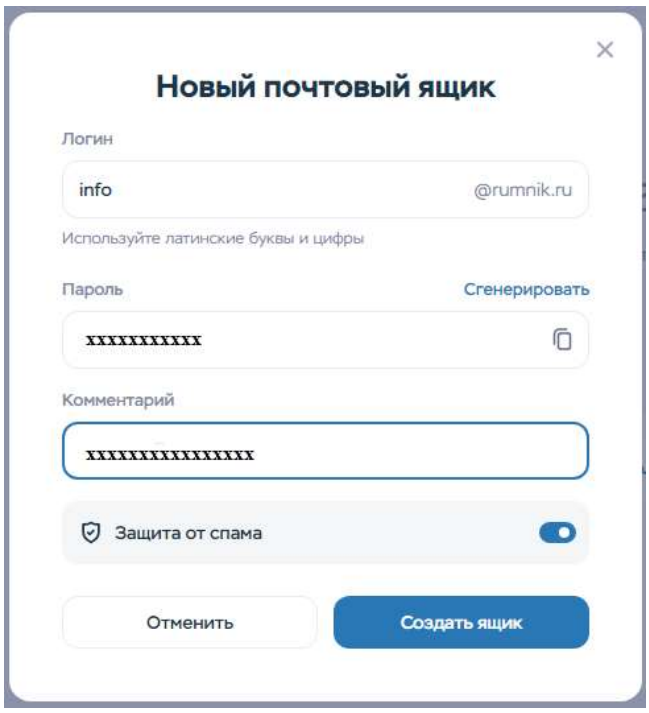


Рис. 7.22. Окно создания нового почтового ящика

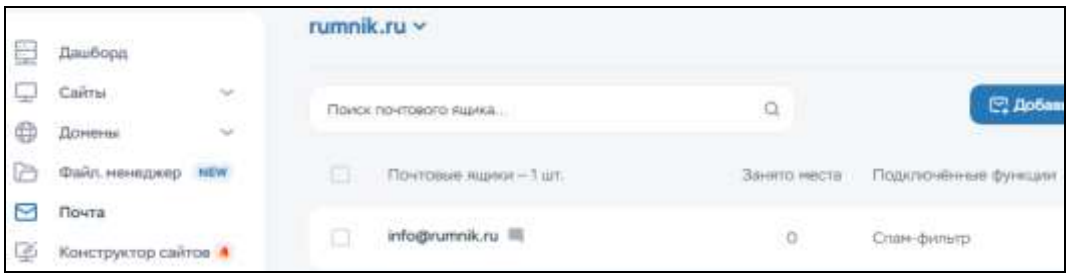


Рис. 7.23. Созданный почтовый ящик

Теперь откройте файл settings.py вашего проекта и настройте почтовый клиент, добавив в конец этого файла следующий код (листинг 7.12).

Листинг 7.12. Параметры почтового клиента в файле settings.py

Примечание.
В демо-версии книги удалены листинги программ.

На этом создание почтового клиента для сайта **rumnik.ru** с адресом электронной почты **info@rumnik.ru** завершено. Теперь мы можем приступить к созданию набора модулей для отправки почтовых сообщений со страниц сайта.

7.3.2. Программные модули для отправки почтовых сообщений

В нашей интерактивной цифровой книге для демонстрации работы модулей отправки электронных писем создана следующая ссылка:

```
<a href="{% url 'master:user_mail'%}" target ="_blank">Отправка почты</a>
```

По этой ссылке будет вызван URL-маршрут `user_mail` (листинг 7.13).

Листинг 7.13. URL-маршрут user_mail

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `user_mail` (листинг 7.14).

Листинг 7.14. Представление user_mail

Примечание.
В демо-версии книги удалены листинги программ.

Обратите внимание, что для работы этого представления нужно выполнить импорт дополнительных пакетов Django. Кроме того, для нашего примера в файле forms.py проекта необходимо создать форму `MailForm` для отображения и корректировки параметров почтового сообщения (листинг 7.15).

Листинг 7.15. Форма MailForm

Примечание.
В демо-версии книги удалены листинги программ.

В этой форме для ввода и корректировки данных созданы три поля:

- `user_mail` — e-mail получателя;

- `teme_mail` — тема письма;
- `body_mail` — содержание письма.

В представлении `user_mail` сначала формируются параметры электронного письма по умолчанию:

- `tem` — тема сообщения;
- `text_ru` — текст почтового сообщения;
- `user_mail` — электронный адрес получателя.

Здесь приведены данные простого примера для проверки работы почтового сервиса. В реальном приложении тема сообщения и текст письма могут формироваться динамически с выборкой информации из БД (например, номера заказа, стоимости продуктов в заказе, даты оплаты заказа и т. п.). Электронный адрес получателя также может быть считан в БД из модели с профилем зарегистрированного пользователя. То есть формирование всех параметров письма и его отправка клиенту может быть выполнена автоматически в фоновом режиме. В нашем же примере после формирования параметров письма по умолчанию создается форма (объект `form`), в которую передаются параметры письма для их корректировки. Затем эта форма с параметрами письма по умолчанию загружается в словарь `context`, который передается в шаблон user_mail.html (листинг 7.16) для корректировки.

Листинг 7.16. Шаблон user_mail.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне выводится набор полей для корректировки данных и имеются две кнопки: `Отмена` — для отмены отправки почтового сообщения и `Отправить письмо` — для отправки почтового сообщения.

В форме этого шаблона вы можете написать свой адрес электронной почты, скорректировать текст темы и содержания письма. После нажатия на кнопку `Отправить письмо` пользователь будет возвращен в преставление `user_mail` в режиме POST-запроса, при этом объект `form` получит из формы `MailForm()` скорректированные пользователем данные.

После проверки корректности формы из нее считывается скорректированная информация, и формируются следующие объекты:

- `user_email = form.cleaned_data['user_mail']` — электронный адрес получателя;
- `tem = form.cleaned_data['teme_mail']` — тема письма;
- `text_ru = form.cleaned_data['body_mail']` — содержание письма.

Затем вызывается функция Django `send_mail()`, в которую передается набор параметров почтового сообщения:

```
send_mail(tem, text_ru, settings.EMAIL_HOST_USER, [user_email])
```

В этой функции присутствуют следующие параметры:

- `tem` — тема почтового сообщения;
- `text_ru` – содержание почтового сообщения на русском языке (к нему можно добавить копию этого сообщения на других языках);
- `settings.EMAIL_HOST_USER` — электронный адрес отправителя (в нашем случае это адрес на серверной площадке Timeweb: `info@rumnik.ru`);
- `user_email` — электронный адрес получателя (по умолчанию `mupmup@mail.ru` или тот адрес, который пользователь ввел в форме `MailForm`).

Именно эта функция активирует отправку почтового сообщения. После отправки почтового сообщения пользователь будет направлен на страницу `book_master`.

7.3.3. Проверка работы модулей отправки почтовых сообщений

Итак, теперь можно проверить, как работают модули отправки почтовых сообщений.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Отправка почты** (рис. 7.24).

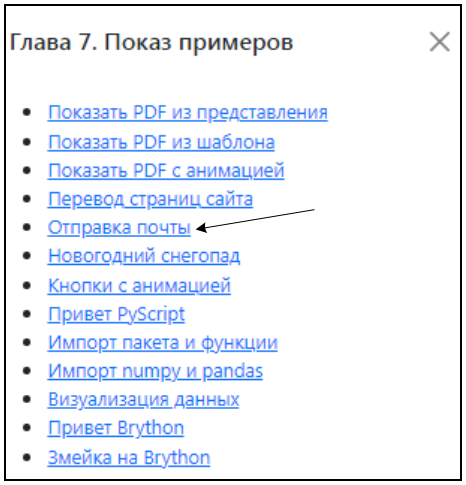


Рис. 7.24. Вкладка **Показ примеров**: перейдите по ссылке **Отправка почты**

Переход по ссылке **Отправка почты** откроет форму `MailForm()` (рис. 7.25).

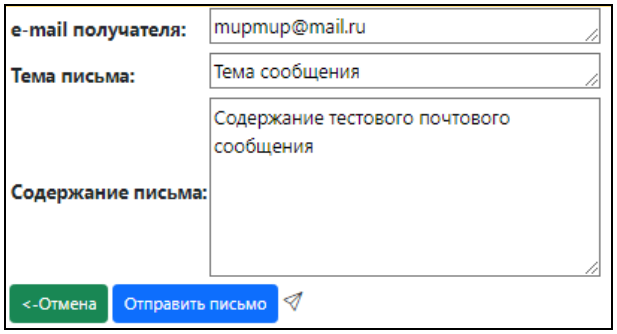


Рис. 7.25. Форма `MailForm()` для корректировки параметров почтового сообщения

Измените в этой форме почтовый адрес получателя (здесь вы можете указать свой личный адрес электронной почты), скорректируйте тему и содержание письма и нажмите на кнопку **Отправить письмо** — в результате вы будете возвращены на страницу `book_master` с вкладками интерактивной цифровой книги (рис. 7.26).

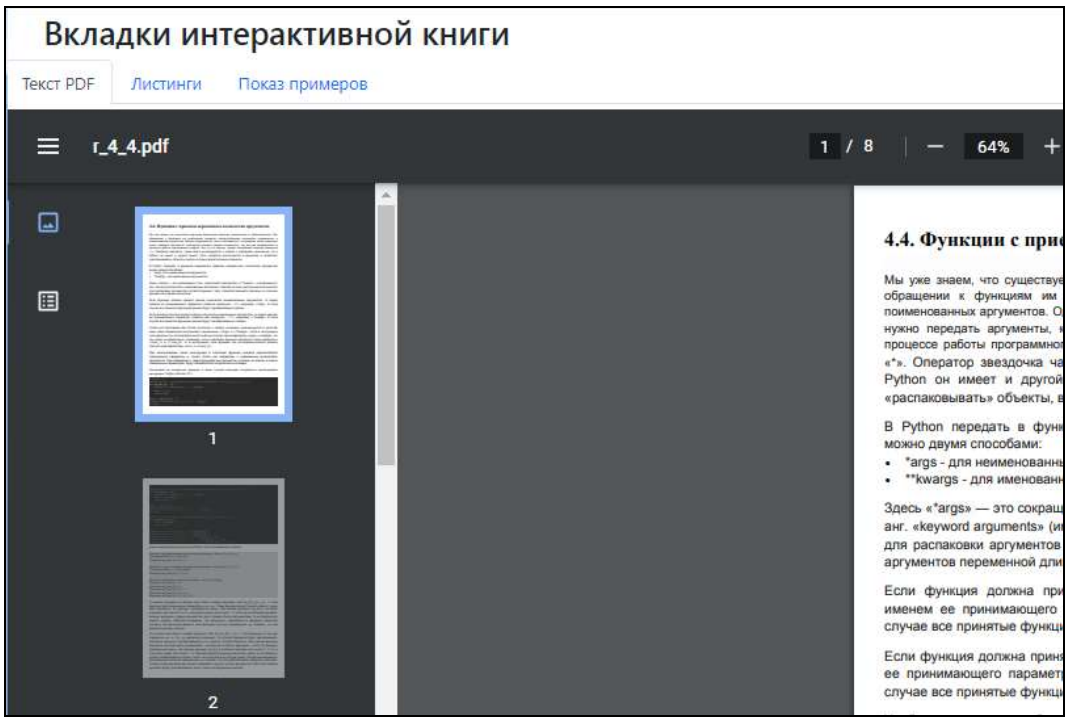


Рис. 7.26. Страница `book_master` после отправки почтового сообщения

Откройте теперь свою личную почту — если все настройки были выполнены корректно, то на нее придет электронное письмо от почтовой службы сайта **rumnik.ru** (рис. 7.27).

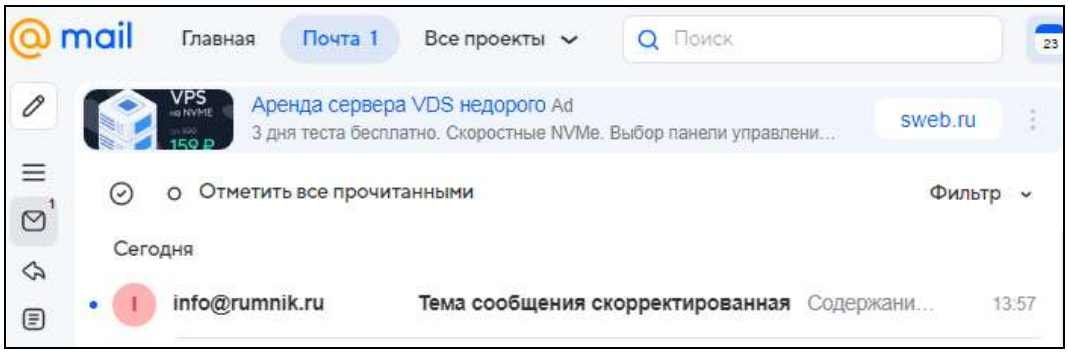


Рис. 7.27. Письмо, полученное от почтовой службы сайта **rumnik.ru**

Вы можете открыть это письмо и прочитать его содержание (рис. 7.28). Как здесь можно видеть, все настройки были выполнены корректно, и почтовая служба сайта доставила получателю электронное письмо с актуальным содержанием.

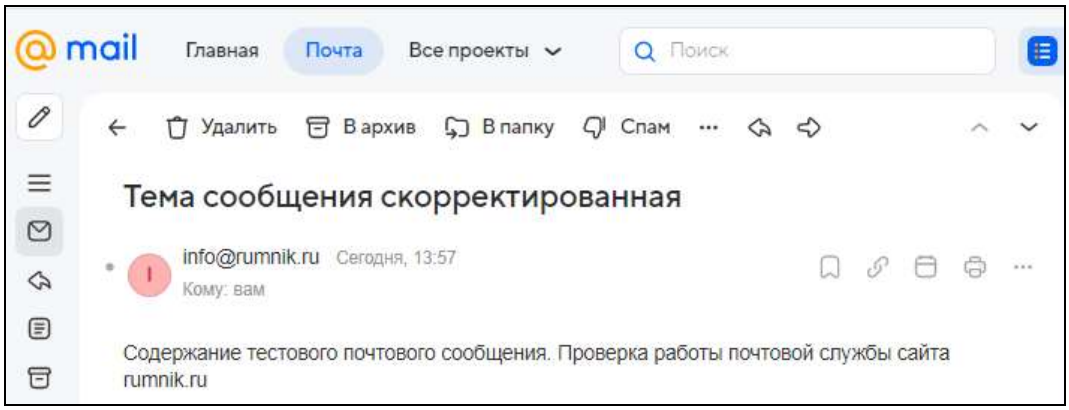


Рис. 7.28. Содержание полученного сообщения от почтовой службы сайта **rumnik.ru**

7.4. Используем скрипты в шаблонах Django

Для встраивания на страницу сайта исполняемого кода используется элемент HTML — тег `<script>`. Обычно такие теги служат для встраивания кода JavaScript. Однако этот элемент также можно задействовать для встраивания кода и на других языках — например, на Python.

Чтобы включить код JavaScript в шаблон Django, можно применить самостоятельный тег `<script>` (в этом случае скрипт будет работать только в рамках текущего шаблона):

```
    <!—Включение кода скрипта -->
    <script>
        // Код функции на JavaScript
        function myFunction() {
            // Программный код тела функции
        }
    </script>
```

а можно использовать тег `<script>` вместе с тегом Django `{% block %}` (обернуть код скрипта в тег Django):

```
{% block javascript %}
    <!—Включение кода скрипта -->
    <script>
        // Код функции на JavaScript
        function myFunction() {
            // Программный код тела функции
        }
    </script>
{% endblock javascript %}
```

В этом случае в дочерних шаблонах можно будет переопределить код текущего скрипта.

Все мы знаем, что в Django на языке Python выполняется создание модулей, работающих на стороне сервера, однако Python также можно использовать вместо JavaScript и для создания скриптов, работающих на стороне клиента. Для этого достаточно подключить к своему проекту пакет Brython.

Brython расшифровывается как «браузерный Python» — реализация Python3, запущенная в браузере. Считается, что основная цель Brython — это возможность заменить JavaScript в качестве языка сценариев для браузера. Большинство современных браузеров, работающих на компьютерах и смартфонах, поддерживают Brython.

Brython, в свою очередь, поддерживает большую часть синтаксиса Python3, включая генераторы, классы, импорт и пр., а также многие модули дистрибутива CPython. Кроме того, он включает библиотеки для взаимодействия с существующими библиотеками JavaScript — такими как jQuery, 3D, Highcharts, Raphael и др., поддерживает последние спецификации HTML5/CSS3 и может использовать CSS-фреймворки — такие как Bootstrap3, LESS, SASS и т. п.

В 2022 году на конференции PyCon 2022 компания Anaconda анонсировала фреймворк PyScript, который позволяет задействовать Python в интернет-браузерах на стандартных HTML-страницах. В последующих разделах мы на небольших примерах покажем, как можно применять Java-скрипты, Python-скрипты и фреймворк PyScript в шаблонах Django.

7.4.1. Скрипты для придания страницам сайта новогоднего стиля

Многие сайты в преддверии праздников меняют оформление своих страниц. Например, перед Новым годом добавляют снежинки и сосульки, перед 23 февраля — элементы воинской атрибутики, перед 8 марта — букеты цветов. Для этого в шаблоны Django встраиваются Java-скрипты, реализующие указанные эффекты. При этом можно определить автоматическую активацию таких скриптов при наступлении заданных дат и отключение их, когда праздничные дни завершаются.

В этом разделе мы рассмотрим, как можно в шаблон Django встроить Java-скрипт, создающий иллюзию падающих снежинок. Его также можно активировать перед Новым годом, а после новогодних праздников — закомментировать. Самое главное здесь то, что для создания всех этих эффектов нет необходимости разрабатывать программный код на языке Java — достаточно подключить готовый скрипт.

Для запуска страницы с новогодним снегопадом в главном меню или на одной из страниц вашего сайта нужно создать ссылку:

```
<a href="{% url 'master:book_snow'%}" target ="_blank">Новогодний снегопад</a>
```

По этой ссылке будет вызван URL-маршрут `book_snow` (листинг 7.17).

Листинг 7.17. URL-маршрут book_snow

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `book_snow` (листинг 7.18).

Листинг 7.18. Представление book_snow

Примечание.

В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон book_snow.html (листинг 7.19).

Листинг 7.19. Шаблон book_snow.html

Примечание.

В демо-версии книги удалены листинги программ.

Этот шаблон является наследником базового шаблона — base_snow.html (листинг 7.20). Именно в этом базовом шаблоне подключены Java-скрипты, обеспечивающие анимацию новогоднего снегопада.

Листинг 7.20. Шаблон base_snow.html

Примечание.

В демо-версии книги удалены листинги программ.

У этого шаблона достаточно большой и сложный код, но нам сейчас важны всего несколько его строчек — тех, где происходит подключение и активация скриптов:

```
<!-- Подключение скриптов Новогоднего снегопада-->
<link href="https://cdn.jsdelivr.net/gh/Alaev-Co/snowflakes/dist/snow.min.css"
rel="stylesheet">
<!-- Запуск скриптов для анимации снегопада-->
<script src="https://cdn.jsdelivr.net/gh/Alaev-
Co/snowflakes/dist/Snow.min.js"></script>
<script> new Snow (); </script>
<!-- Конец снегопада-->
```

Здесь с использованием тега `<link … >` происходит подключение к удаленным скриптам в сети Интернет, а в двух следующих тегах `<script>…</script>` обеспечивается запуск кода скриптов на выполнение.

Итак, теперь можно проверить, как работают модули придания странице сайта новогоднего стиля.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Новогодний снегопад** (рис. 7.29).

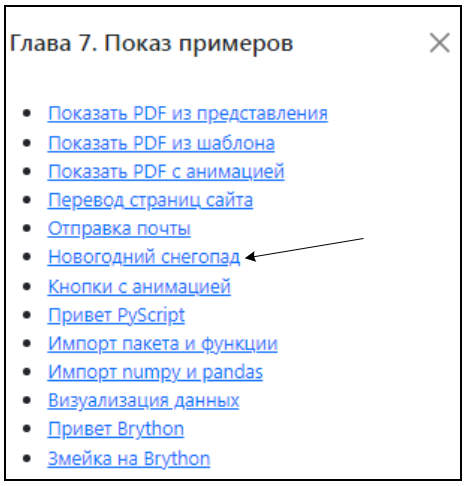


Рис. 7.29. Вкладка **Показ примеров**: вызов примера страницы с новогодним стилем

Переход по ссылке **Новогодний снегопад** вызовет открытие страницы с шаблоном book_snow.html (рис. 7.30).



Рис. 7.30. Страницы в новогоднем стиле с падающими снежинками и растущими сугробами

Как здесь можно видеть, сверху появились падающие снежинки, а в нижней части страницы растут снежные сугробы. Когда сугробы достигают определенной высоты, выезжает трактор и отвалом убирает выросшие сугробы (рис. 7.31).



Рис. 7.31. Трактор, убирающий снежные сугробы

Всего несколько строк программного кода для подключения Java-скриптов, добавленные в этом примере, придают страницам книги живость за счет симпатичных анимационных эффектов. Это обеспечивает привлекательность сайта и создает у пользователей праздничное настроение.

7.4.2. Скрипты для анимации кнопок

Страница сайта становится более «живой» и в том случае, когда кнопки откликаются на действия пользователя, — например, при нажатии на них меняют цвет, размер или надпись. Так, при большом количестве записей в БД добавление, обновление или удаление данных может занимать некоторое время. В этом случае при нажатии, например, на кнопку с надписью **Удалить** на ней может появиться вращающий спин, и он должен присутствовать на кнопке до тех пор, пока процедура удаления записи из БД не будет завершена. Или — другой вариант: при нажатии на кнопку с надписью **Удалить** она должна стать недоступной, а надпись на ней смениться на **ЖДИТЕ…**. Для анимации кнопок также используются Java-скрипты.

Напишем пример программного кода, в котором создадим три кнопки с анимацией: первая кнопка при нажатии будет уменьшаться в размерах, на второй надпись в процессе выполнения некоторого действия будет заменена на вращающийся спин, третья кнопка после нажатия станет недоступной с заменой надписи.

Для демонстрации страницы с анимированными кнопками в главном меню или на одной из страниц вашего сайта нужно создать ссылку:

```
<a href="{% url 'master:a_button'%}" target ="_blank">Кнопки с анимацией</a>
```

По этой ссылке будет вызван URL-маршрут `a_button` (листинг 7.21).

Листинг 7.21. URL-маршрут a_button

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `a_button` (листинг 7.22).

Листинг 7.22. Представление a_button

Примечание.
В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон a_button.html (листинг 7.23).

Листинг 7.23. Шаблон a_button.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне созданы три кнопки:

- `Кнопка с анимацией` — уменьшается при нажатии;

- `Кнопка со спином` — при первом нажатии появляется спин, при втором нажатии он исчезает;
- `УДАЛИТЬ ЗАПИСЬ ИЗ БД` — при нажатии кнопка становится недоступной и на ней меняется надпись.

Для кнопки с анимацией добавлены стили и скрипт, для двух других кнопок добавлены скрипты. Первая и третья кнопки для связи со скриптами имеют идентификаторы (`id`).

Теперь можно проверить, как работают модули анимации кнопок.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Кнопки с анимацией** (рис. 7.32).

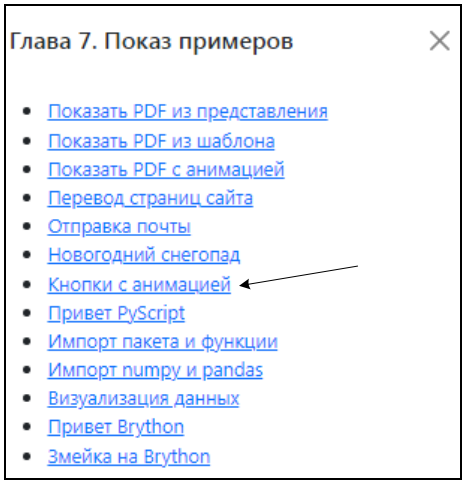


Рис. 7.32. Вкладка **Показ примеров**: вызов примера страницы с анимированными кнопками

При переходе по ссылке **Кнопки с анимацией** будет показан шаблон a_button.html (рис. 7.33).

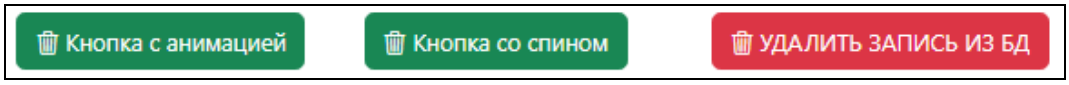


Рис. 7.33. Внешний вид страницы с анимированными кнопками

Если здесь нажать на кнопку с надписью **Кнопка с анимацией**, она уменьшится в размерах, а когда вы ее отпустите, она вернется к своим прежним размерам.

Если нажать на кнопку с надписью **Кнопка со спином**, надпись на кнопке будет заменена на вращающийся спин (рис. 7.34), а при повторном нажатии на нее она примет прежний вид.



Рис. 7.34. Внешний вид кнопки с вращающимся спином

В реальном проекте спин будет вращаться на кнопке до тех пор, пока процесс, с которым она связана, не завершится (например, это может быть удаление записи из БД). А по завершении процесса произойдет переход к другой странице сайта, и кнопка с вращающимся спином просто исчезнет с экрана.

При нажатии на кнопку **УДАЛИТЬ ЗАПИСЬ ИЗ БД** она заблокируется и станет недоступной, при этом на кнопке появится новая надпись: **ЖДИТЕ…** (рис. 7.35).

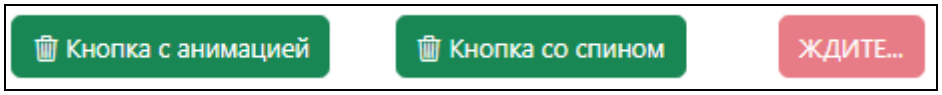


Рис. 7.35. Внешний вид кнопки **УДАЛИТЬ ЗАПИСЬ ИЗ БД** после блокировки и смены надписи

7.5. Используем фреймворк PyScript в шаблонах Django

PyScript — это интерфейсная среда Python, которая позволяет пользователям создавать программы на Python с использованием HTML-интерфейса в браузере. Она была разработана с использованием возможностей современных веб-технологий.

За последние пару десятилетий Python и продвинутые языки пользовательского интерфейса — такие как современные HTML, CSS и JavaScript — существовали в отрыве друг от друга. У Python не было простого механизма для создания привлекательных пользовательских интерфейсов в веб-приложениях, в то время как HTML, CSS и JavaScript активно развивались в этом направлении. Однако среда PyScript не предназначена для того чтобы заменить JavaScript в браузере — скорее, она призвана предоставить разработчикам Python, особенно специалистам по обработке данных, больше гибкости и удобств работы при разработке веб-приложений.

PyScript предоставляет язык программирования с единообразными соглашениями о стиле, большей выразительностью и простотой изучения и реализует следующие возможности:

- работает непосредственно в браузере, обеспечивая поддержку кода на Python без дополнительных настроек;
- осуществляет двунаправленное взаимодействие между объектами и пространствами имен Python и JavaScript;
- поддерживает популярные пакеты Python, такие как pandas, NumPy и многие другие;
- позволяет легко создавать расширяемые компоненты непосредственно на Python;
- позволяет разработчикам выполнять импорт файлов и пакетов, которые следует добавить в код своей страницы;
- позволяет создавать проекты с использованием доступных компонентов пользовательского интерфейса, таких как кнопки, контейнеры и др.

Чтобы использовать PyScript на HTML-странице, нужно обеспечить доступ к интерфейсу PyScript с использованием следующего кода:

```
<link rel="stylesheet" href="https://pyscript.net/alpha/pyscript.css" />
<script defer src="https://pyscript.net/alpha/pyscript.js"></script
```

7.5.1. PyScript: отображение в шаблоне Django содержимого символьной переменной и списка

Напишем простой код на PyScript, который отобразит в шаблоне Django содержимое символьной переменной и списка. Для демонстрации страницы с кодом на PyScript в главном меню или на одной из страниц вашего сайта нужно создать ссылку:

```
<a href="{% url 'master:hello_ps'%}" target ="_blank">Привет PyScript</a>
```

По этой ссылке будет вызван URL-маршрут `hello_ps` (листинг 7.24).

Листинг 7.24. Маршрут URL с именем hello_ps

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `hello_ps` (листинг 7.25).

Листинг 7.25. Представление hello_ps

Примечание.
В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон hello_ps.html (листинг 7.26).

Листинг 7.26. Шаблон hello_ps.html

Примечание.

В демо-версии книги удалены листинги программ.

В самом начале этого шаблона выполняется подключение модулей фреймворка PyScript. Затем создается тег `<py-script>…</py-script>`, в теле которого содержится самый обычный код на Python: печатается текстовая строка и создается список `list`, который выводится на печать с использование цикла `for`.

Теперь можно проверить, как работают скрипты фреймворка PyScript.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Привет PyScript** (рис. 7.36).

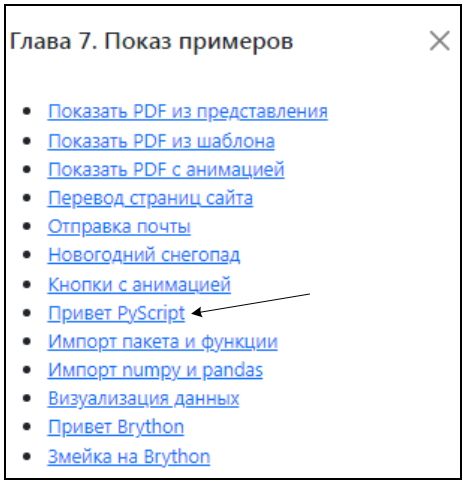


Рис. 7.36. Вкладка **Показ примеров**: вызов примера страницы с показом содержимого символьной переменной и списка

При переходе по ссылке **Привет PyScript** будет показан шаблон hello_ps.html (рис. 7.37). Как здесь можно видеть, скрипт вывел на HTML-страницу содержимое текстовой переменной и список фруктов.

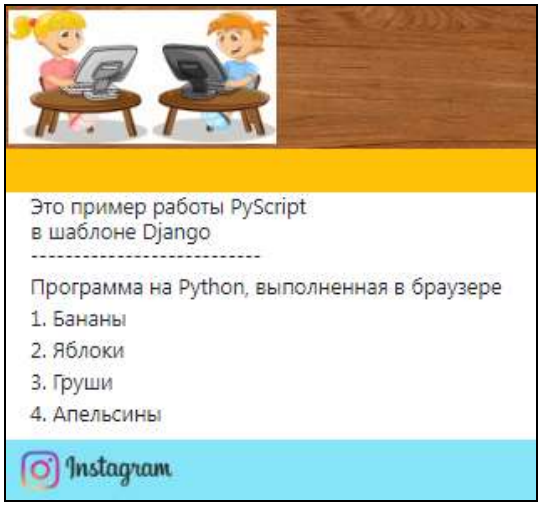


Рис. 7.37. Внешний вид страницы с показом содержимого символьной переменной и списка

7.5.2. PyScript: импорт дополнительных библиотек, создание функции и выполнение обращений к ним

В скриптах PyScript можно выполнять импорт дополнительных библиотек, создавать функции и выполнять обращение к ним.

Для демонстрации возможностей импорта пакетов и создания функций на PyScript в главном меню или на одной из страниц вашего сайта нужно создать ссылку:

```
<a href="{% url 'master:import_p'%}" target ="_blank">Импорт пакета и
функции</a>
```

По этой ссылке будет вызван URL-маршрут `import_p` (листинг 7.27).

Листинг 7.27. URL-маршрут import_p

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `import_p` (листинг 7.28).

Листинг 7.28. Представление import_p

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон import_p.html (листинг 7.29).

Листинг 7.29. Шаблон import_p.html

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

В самом начале этого шаблона выполняется подключение модулей фреймворка PyScript. Затем создается тег `<py-script>…</py-script>`, в теле которого содержится самый обычный код на Python. В этом коде выполняется импорт пакета для генерации целых случайных чисел: `from random import randint`. Определены также две функции: для сложения двух чисел: `def add_two_numbers(x, y)` и для генерации случайного числа в интервале от 0 до 10: `def generate_random_number()`.

Теперь можно проверить, как работают скрипты PyScript.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Импорт пакета и функции** (рис. 7.38).

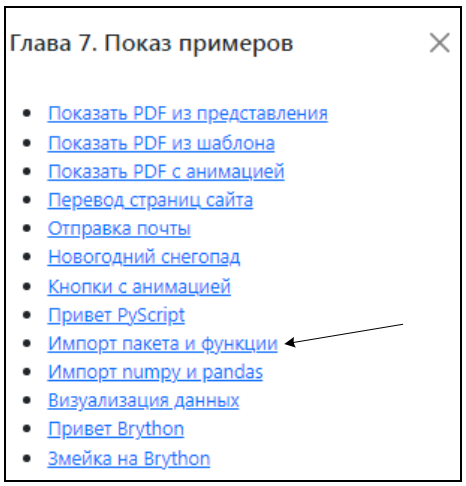


Рис. 7.38. Вкладка **Показ примеров**: вызов примера страницы для импорта дополнительных библиотек, создания функции и выполнения обращений к ним

При переходе по ссылке **Импорт пакета и функции** будет показан шаблон import_p.html (рис. 7.39). Как здесь можно видеть, скрипт вывел на HTML-страницу результаты работы функций. Поскольку функции работают со случайными числами, то при каждом обновлении страницы будет получен другой результат.

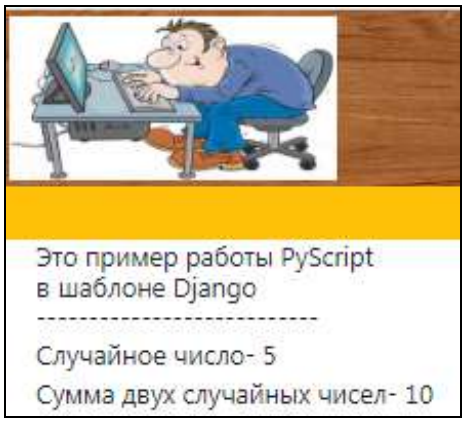


Рис. 7.39. Внешний вид страницы с результатами работы функций

7.5.3. PyScript: работа с пакетами NumPy и pandas

В скриптах PyScript можно задействовать такие библиотеки для обработки данных, как NumPy и pandas.

Для демонстрации возможностей работы с пакетами NumPy и pandas в главном меню или на одной из страниц вашего сайта нужно создать ссылку:

```
<a href="{% url 'master:import_p2'%}" target ="_blank">Импорт numpy и pandas</a>
```

По этой ссылке будет вызван URL-маршрут `import_p2` (листинг 7.30).

Листинг 7.30. URL-маршрут import_p2

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `import_p2` (листинг 7.31).

Листинг 7.31. Представление import_p2

Примечание.

В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон import_p2.html (листинг 7.32).

Листинг 7.32. Шаблон import_p2.html

Примечание.

В демо-версии книги удалены листинги программ.

В самом начале этого шаблона выполняется подключение модулей фреймворка PyScript. Затем создается тег `<py-script>…</py-script>`, в теле которого содержится самый обычный код на Python. В этом коде выполняется импорт пакетов numpy и pandas для обработки статистики и пределен объект `x`, в который с использованием библиотеки pandas загружен одномерный массив. Затем с помощью цикла `for` этот массив выводится на печать.

Теперь можно проверить, как работают скрипты фреймворка PyScript.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Импорт numpy и pandas** (рис. 7.40).

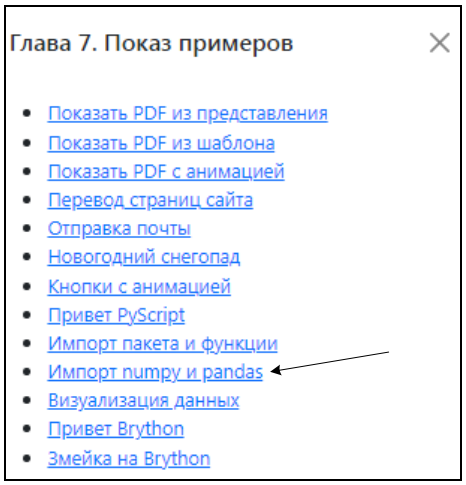


Рис. 7.40. Вкладка **Показ примеров**: вызов примера страницы для работы с пакетами NumPy и pandas

При переходе по ссылке **Импорт numpy и pandas** будет показан шаблон import_p2.html (рис. 7.41).

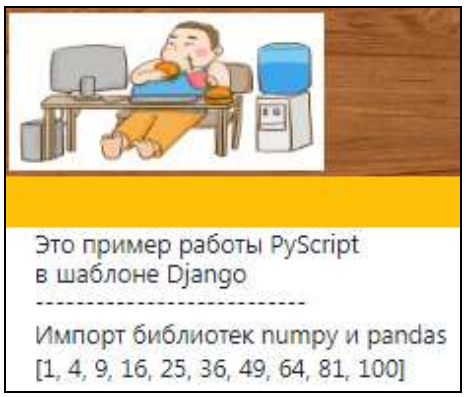


Рис. 7.41. Внешний вид страницы с результатами работы с пакетами NumPy и pandas

7.5.4. PyScript: работа с внешними библиотеками для обработки и визуализации данных

В скриптах PyScript можно задействовать внешние библиотеки для обработки и визуализации данных.

Для демонстрации возможностей работы библиотеки для обработки и визуализации данных в главном меню или на одной из страниц вашего сайта нужно создать ссылку:

```
<a href="{% url 'master:import_p3'%}" target ="_blank">Визуализация данных</a>
```

По этой ссылке будет вызван URL-маршрут `import_p3` (листинг 7.33).

Листинг 7.33. URL-маршрут import_p3

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `import_p3` (листинг 7.34).

Листинг 7.34. Представление import_p3

Примечание.
В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон import_p3.html (листинг 7.35).

Листинг 7.35. Шаблон import_p3.html

Примечание.
В демо-версии книги удалены листинги программ.

В самом начале этого шаблона выполняется подключение модулей фреймворка PyScript и ряда других java-скриптов. Затем создается тег `<py-script>…</py-script>`, в теле которого содержится самый обычный код на Python. В этом коде выполняется импорт пакетов для обработки и визуализации данных и создаются два списка со статистическими данными. Затем с использованием библиотек для визуализации данных выводится гистограмма продаж овощей за последний месяц.

Теперь можно проверить, как работают эти скрипты фреймворка PyScript.

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Визуализация данных** (рис. 7.42).

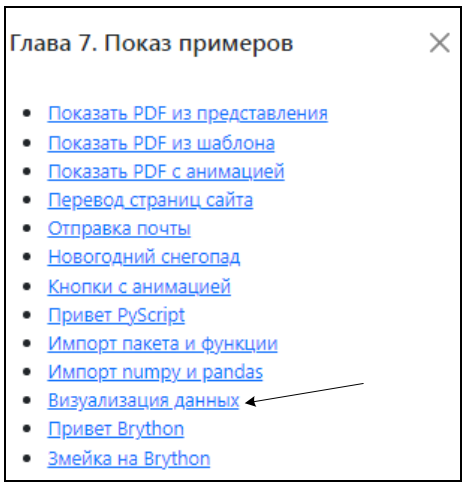


Рис. 7.42. Вкладка **Показ примеров**: вызов примера страницы для работы с внешними библиотеками, обеспечивающими обработку и визуализацию данных

При переходе по ссылке **Визуализация данных** будет показан шаблон import_p3.html (рис. 7.43).

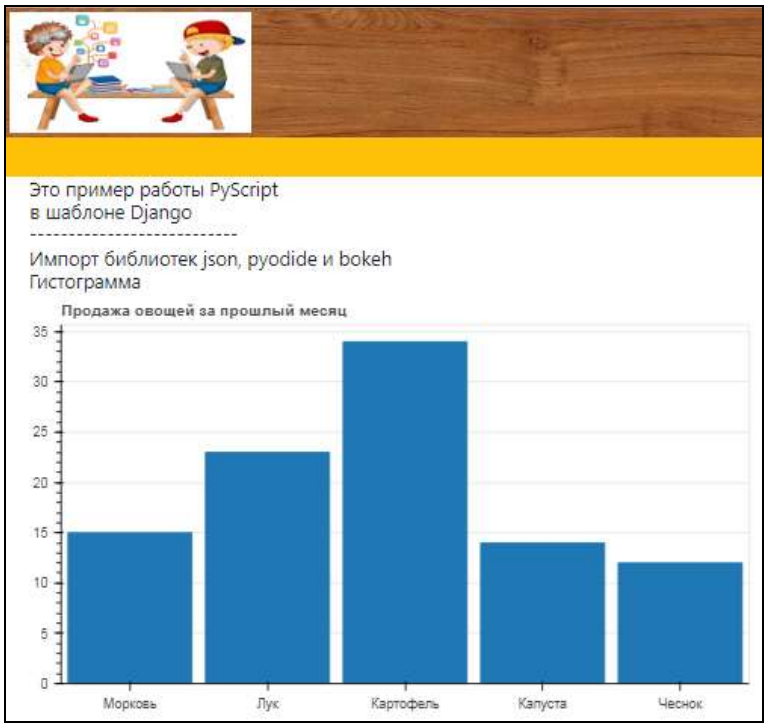


Рис. 7.43. Внешний вид страницы с результатами работы с внешних библиотек, обеспечивающих обработку и визуализацию данных

* * *

Из материалов этого раздела вы узнали, что такое PyScript, и как использовать его для запуска кода Python в веб-браузерах. PyScript находится пока на ранних стадиях развития (альфа-тестирование) и сталкивается с такой проблемой, как длительное время загрузки модулей. Но он будет развиваться и совершенствоваться, и уже сейчас вы можете использовать его в своих проектах.

Как показано в рассмотренных в этом разделе примерах, фреймворк PyScript может быть прекрасным дополнением к фреймворку Django. Оба фреймворка используют язык программирования Python, а значит, вы можете в проектах задействовать знакомый и привычный набор библиотек Python для обработки и визуализации данных.

7.6. Используем пакет Brython для создания программ на Python для HTML-страниц

Хотя JavaScript является общепринятым языком интерфейсной веб-разработки, у вас может быть несколько причин искать альтернативные возможности:

- вам не нравится писать код на JavaScript, и вы его плохо знаете;
- у вас нет времени на изучение JavaScript.
- у вас большой опыт и практические навыки в использовании Python;

Какова бы ни была причина, многие разработчики предпочли бы найти в Python альтернативу JavaScript для создания программ, работающих не только на сервере, но и в браузере клиента. Однако одним из негативных эффектов использования Python в браузере является потеря производительности по сравнению с тем же кодом на JavaScript, и пакет Brython позволяет преодолеть этот эффект, компилируя код на Python в код на JavaScript.

Чтобы использовать Brython на HTML-странице, нужно обеспечить доступ к модулям этого пакета:

```
<script type = "text/javascript"
        src = "https://cdn.jsdelivr.net/npm/brython@3.8.9/brython.min.js">
</script>


<body onload = "brython()">
    <script type = "text/python">
         <код на Python>
    </script>
</body>
```

Здесь сначала загружаются Java-скрипты, обеспечивающие работу программных модулей Brython в среде браузера (`brython.min.js`). После этого вызывается функция `brython()`, которая читает код Python в текущей области видимости и компилирует его в JavaScript. Затем тег `<script type = "text/python">…</script>` устанавливает тип сценария `text/python` — это указывает Brython, что внутри тега находится код Python, который необходимо скомпилировать и выполнить.

7.6.1. Brython: вывод на HTML-страницу обычной символьной строки

Напишем для примера простой код на Python, который отобразит на HTML-странице обычную символьную строку. Для демонстрации этой страницы в главном меню или на одной из страниц вашего сайта нужно создать ссылку:

```
<a href="{% url 'master:hello_br'%}" target ="_blank">Привет Brython</a>
```

По этой ссылке будет вызван URL-маршрут `hello_br` (листинг 7.36).

Листинг 7.36. URL-маршрут hello_br

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `hello_br` (листинг 7.37).

Листинг 7.37. Представление hello_br

Примечание.
В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон hello_br.html (листинг 7.38).

Листинг 7.38. Шаблон hello_br.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне сначала загружаются Java-скрипты, а затем вызывается функция `brython()`, которая читает код Python, в котором выполняется импорт пакета Brython, и в окне браузера выводится строка `Привет Brython!`.

Теперь можно проверить, как работает код Python в среде Brython.

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Привет Brython** (рис. 7.44).

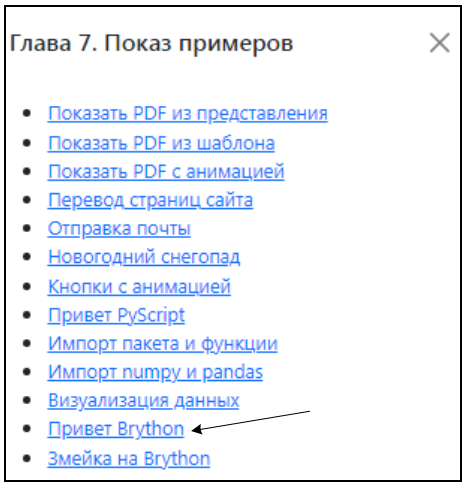


Рис. 7.44. Вкладка **Показ примеров**: вызов примера страницы для вывода с помощью пакета Brython простой символьной строки

При переходе по ссылке `Привет Brython` будет показан шаблон hello_br.html (рис. 7.45). Как здесь можно видеть, скрипт вывел на HTML-страницу содержимое текстовой переменной.

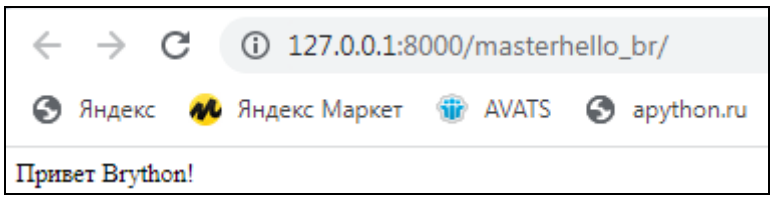


Рис. 7.45. Внешний вид страницы, демонстрирующей вывод с помощью пакета Brython простой символьной строки

7.6.2. Brython: игра «Змейка» (создание окон, функций и обращений к ним)

Рассмотрим пример с более сложным кодом, где создаются окна, функции и обращения к ним (например, игру «Змейка»). Для вызова такого кода создадим следующую ссылку:

```
<li><a href="{% url 'master:hello_br1'%}" target ="_blank">Змейка на
Brython</a></li>
```

По этой ссылке будет вызван URL-маршрут `hello_br1` (листинг 7.39).

Листинг 7.39. URL-маршрут hello_br1

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `hello_br1` (листинг 7.40).

Листинг 7.40. Представление hello_br1

Примечание.

В демо-версии книги удалены листинги программ.

Из этого представления вызывается шаблон hello_br1.html (листинг 7.41).

Листинг 7.41. Шаблон hello_br1.html

Примечание.

В демо-версии книги удалены листинги программ.

В этом шаблоне сначала загружаются Java-скрипты, задаются стили элементам интерфейса, а затем вызывается функция `brython()`,читающая и исполняющая код на Python, в котором создано несколько функций для обработки нажатия клавиш со стрелками.

Теперь можно проверить, как работает код Python в среде Brython.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 7** и перейти по ссылке **Змейка на Brython** (рис. 7.46).

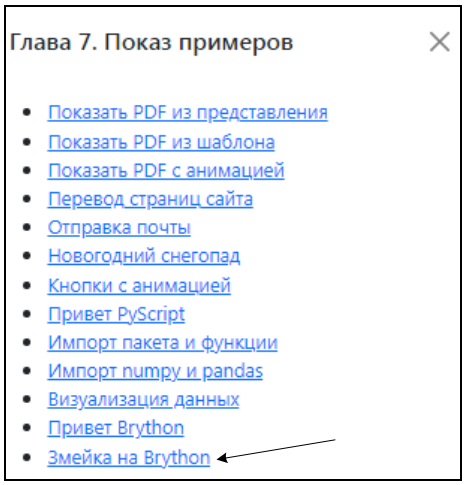


Рис. 7.46. Вкладка **Показ примеров**: вызов примера страницы со скриптами игры «Змейка» на Python

При переходе по ссылке **Змейка на Brython** будет показан шаблон hello_br1.html (рис. 7.47).

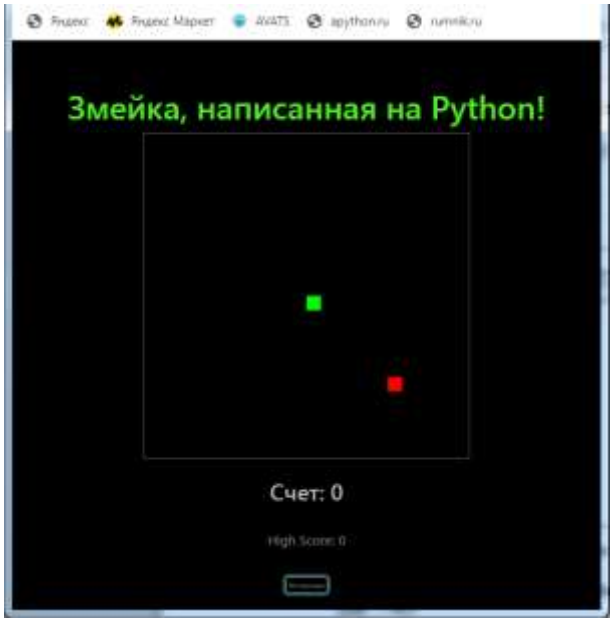


Рис. 7.47. Внешний вид страницы с игрой «Змейка»

Как здесь можно видеть, в окне браузера имеется поле, в котором передвигается змейка (зеленый квадрат). Нажатием на стрелки на клавиатуре компьютера запускается перемещение змейки (рис. 7.48). Стрелки («вверх», «вниз», «вправо» и «влево») меняют направления перемещения змейки, а клавиша «пробел» — делает паузу в игре. В игре надо менять направления движения змейки таким образом, чтобы она съела мышку (красный квадрат).

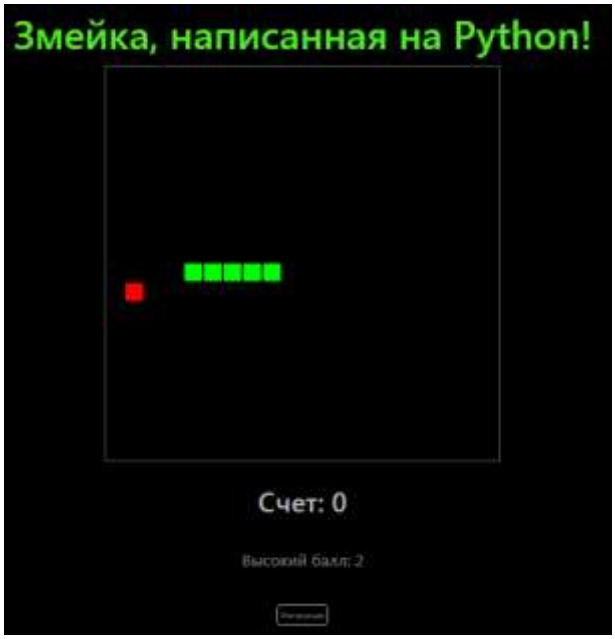


Рис. 7.48. Момент игры «Змейка» в окне браузера

7.7. Откатываем миграцию

В процессе разработки и отладки веб-приложения часто возникают ситуации, когда нужно откатить миграцию, не нарушив при этом целостность БД. Для отката миграции необходимо в терминале PyCharm выполнить определенные команды. Например, чтобы откатить последнюю миграцию: `0011_migration_to_revert` до предпоследней: `0010_previous_migration` — следует выполнить команду:

```
python manage.py migrate my_app 0010_previous_migration
```

где `my_app` — имя вашего приложения.

Впрочем, даже не обязательно использовать полное имя миграции — достаточно указать лишь ее номер:

```
python manage.py migrate my_app 0010
```

Затем вы можете в приложении удалить файл миграции `0011_migration_to_revert`.

Для отката всех миграций нужно выполнить команду:

```
python manage.py migrate <your_app_name> zero
```

после чего удалить все файлы миграций.

7.8. Краткие итоги

В этой главе приведены советы по решени. только небольшого круга задач. При возникновении любой проблемы или вопроса, на просторах Интернета вы найдете ответ от сообщества пользователей Django, но для этого понадобится время и терпение. А чтобы уменьшить непроизводительные затраты времени, документируйте и описывайте разрабатываемые вами программные модули. Через какое то время именно в них вы будете быстрее находить ответы на возникшие вопросы.

Глава 8. Визуализация данных в Django

В этой главе описываются способы вывода в шаблоны Django данных в виде статичных и интерактивных 2D-графиков, а также в виде динамичных трехмерных 3D-изображений линий и поверхностей. Для этих целей мы задействуем библиотеки Plotly, NumPy, pandas, Skimage и ряд других. Построение и вывод графиков с использованием этих библиотек на «чистом» Python не вызывает трудностей. Однако вывод с их помощью данных на HTML-страницы может вызвать затруднения даже у опытных программистов. В документации на упомянутые библиотеки содержатся только фрагменты элементарных примеров, а на просторах Интернета присутствует достаточно много не полных и не всегда работающих примеров использования этих библиотек. Кроме того, создание «живых» интерактивных графиков требует взаимодействия шаблонов Django с Java-скриптами, при этом любая элементарная ошибка в скрипте приводит к «обвалу» целой цепочки программных модулей.

Здесь же приводятся проверенные и реально работающие программные модули, которые в том или ином виде встроены в шаблоны Django на сайтах в сети Интернет, а также делается разбор функциональной сущности наиболее значимых фрагментов их программного кода.

Из материалов этой главы вы узнаете:

- как реализовать визуализацию данных в виде статичных 2D-графиков;
- как «оживить» 2D-графики и сделать их динамичными (интерактивными);
- как реализовать визуализацию данных в виде объемных 3D-графиков;
- как реализовать анимацию 3D-диаграмм;
- как вывести наборы данных на географические карты;
- как реализовать анимацию объектов, выведенных на географические карты.

8.1. Визуализация данных с библиотекой plotly.py: двумерная графика (2D)

Библиотека Plotly Python — это интерактивная библиотека с открытым исходным кодом для построения графиков, которая поддерживает более 40 уникальных типов диаграмм, охватывающих широкий спектр статистических, финансовых, географических и научных вариантов использования. Созданная на основе библиотеки Plotly JavaScript (plotly.js), Plotly Python позволяет пользователям Python создавать красивые интерактивные веб-визуализации, которые можно сохранять в автономных файлах HTML или задействовать как часть веб-приложений, созданных на «чистом» Python, а также с использованием фреймворков Dash или Django. Библиотеку Plotly Python иногда называют «plotly.py» — чтобы отличать ее от библиотеки Plotly JavaScript (plotly.js).

> ***Совет***
> Для использования plotly.py не требуется подключения к Интернету, создания учетной записи или оплаты, и до версии 4 эта библиотека могла работать как в онлайн-, так и в офлайн-режиме. Однако, начиная с версии 4, все онлайн-функции были из пакета plotly удалены, но стали доступны в другом самостоятельном пакете: Chart-studio. Исходя из этого, нужно очень осторожно относиться к примерам программного кода, циркулирующим в Интернете, — многие примеры просто не работают или работают не корректно, так как в них нет ссылок на версию библиотеки plotly.py, а используемая там версия может не соответствовать той, которую используете вы.

Для начала работы с библиотекой Plotly необходимо ее установить, поскольку она не входит в стандартный пакет Python. Для этого рекомендуется воспользоваться командой `pip`:

```
pip install plotly
```

Для работы с графикой (загрузки и обработки сырых данных) нам также понадобятся библиотеки pandas, NumPy, Skimage и ряд других. Все эти библиотеки необходимо импортировать в модуль приложения view.py. Далее приведен перечень инструкций, которые использовались в книге для импорта необходимых пакетов:

```
# Импорт пакетов для построения графиков
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

```
import numpy as np
from decimal import Decimal
import pandas as pd
import math
```

> ***Совет***
>
> В целях исключения дублирования одних и тех же фрагментов программного кода в листингах программных модулей этой книги отсутствуют инструкции по импорту указанных библиотек и пакетов. Учтите это при копировании листингов программ для использования в своих проектах, и не забывайте о включении в них строк для импорта необходимых библиотек.

Кроме того, следует определить переменные с путями к базовым каталогам проекта:

```
# Для загрузки правильных путей в windows и linux
from pathlib import Path

# Каталоги проекта
base_dir = settings.BASE_DIR
media_dir = settings.MEDIA_ROOT
```

Объекты `base_dir` и `media_dir` и пакет `Path` используются в модулях, где необходимо чтение текстовых файлов или файлов с изображениями из каталогов приложения static и media. Дело в том, что в разных операционных системах для указания путей к файлам применяются разные типы слешей (косой черты): в Windows — «\», а в Linux — «/»). Из-за этого программисты, которые разработали веб-приложение на Windows, могут потерять и не загрузить нужные файлы после развертывания своих приложений на публичных серверах, которые, как правило, работают под Linux. Чтобы избежать таких ошибок, надо для указания путей к файлам задействовать пакет `Path`, который автоматически определяет тип операционной системы и корректно формирует пути к нужному файлу. Кроме того, в представлениях (views) следует избегать вызовов файлов из каталога static, так как этот каталог на локальном сервере и на публичном сервере доступен по разным путям. Для размещения подгружаемых файлов лучше использовать каталог media, так как пути к этому каталогу на локальном и публичном серверах идентичны.

В plotly.py для формирования и вывода фигур (графиков, диаграмм, карт, рисунков и т. п.) используется библиотека plotly.js языка JavaScript, но JavaScript вам знать и писать на нем код вовсе не придется. Фигуры представляются данными в формате JSON (JavaScript Object Notation), который по структуре мало чем отличается от словаря (`dict`) в Python. Доступ к элементам графика возможен через вызов класса `plotly.graph_objects.Figure`. Чтобы посмотреть структуру созданного вами графика в формате JSON, следует применять инструкцию `print(fig)`, а инструкция `fig.show()` выведет саму фигуру в формате HIML в окно браузера.

> ***Пояснение***
>
> Инструкция `fig.show()` применяется в тех случаях, когда вы работаете с приложением на чистом Python (например, в настольном приложении). В Django используется иной подход для вывода графиков: сначала объект `fig` преобразуется в другой объект формата HTML, который затем передается в шаблон Django для отображения в окне браузера.

Поскольку фигура (график) представляется в формате JSON, то доступ к атрибутам осуществляется за счет полного прописывания пути к нужному параметру. Например, запись `layout.width` обращается к ширине (`width`) макета (`layout`), в котором размещен график. Если же значением какого-либо атрибута является список (`list`), а не словарь, то используются квадратные скобки, — например: `layout.annotations[].text`.

Для изменения значений атрибутов можно либо просто передать им новое значение, либо задействовать методы обновления. Вот так в Python выглядят оба способа для изменения шрифта:

```
fig.layout.title.font.family="Open Sans"
fig.update_layout(title_font_family="Open Sans")
```

В Plotly есть три корневых атрибута: `data` (данные), `layout` (макет) и `frames` (кадр, рамка).

- Корневой атрибут `data` (данные).

  Первый корневой атрибут — `data` — представляет собой список словарей. Эти словари называются *трейсами* (traces — следы, трассы) и обладают следующими характеристиками:

- каждый атрибут `trace` имеет один из 40 типов (`type`) фигур: `scatter` — точки и линии, `bar` — столбчатая диаграмма, `pie` – круговая диаграмма и т. п.;
- каждый трейс отображается в собственном графическом окне (`sublot`);
- каждый трейс может иметь одну легенду, за исключением круговой диаграммы (`pie`) и воронкообразного графика (`funnel chart`);
- некоторые типы фигур поддерживают спектр цветов, которые ассоциируются с цветной полосой (цвета таких фигур могут изменяться напрямую, либо через атрибут `coloraxis` в `layout`).

❑ Корневой атрибут `layout` (макет).

Второй корневой атрибут — `layout` — представляет собой словарь. Этот атрибут обеспечивает настройку параметров, не связанных с данными фигуры. К таким параметрам относятся:

- размеры и координаты фигуры;
- макетные параметры фигуры: шаблоны, шрифты, цвета, метки при наведении;
- заголовок и легенда;
- графические окна для расположения в них самих графиков (`traces`).

Для каждого типа `traces` нужно задавать свои атрибуты:

- `xaxis`, `yaxis`, `xaxis2`, `yaxis2` и т. д. — для диаграмм в декартовой системе координат;
- `scene`, `scene2`, `scene3` и т. д. 00 для 3D-графиков;
- `ternary`, `ternary2`, `ternary3`, `polar`, `polar2`, `polar3`, `geo`, `geo2`, `geo3`, `mapbox`, `mapbox2`, `mabox3` и т. д. — для тернарных, полярных и географических окон соответственно.

Можно указать не относящиеся к данным метки, которые могут быть расположены в «макетных» координатах или в двумерной декартовой системе координат (о тех и других см. далее):

- аннотации;
- формы (линии, прямоугольники и т. д.);
- рисунки.

Можно также указать элементы управления, которые могут быть расположены в макетных координатах и способные запускать функции plotly.js при взаимодействии с пользователем:

- меню обновлений (`updatemenus`): кнопки, выпадающие списки, переключатели и т. д.;
- слайдеры (`sliders`).

❑ Корневой атрибут `frames` (рамка, кадр).

Третьим корневым атрибутом является `frames` — значения для фреймов представлены списком словарей. Эти словари соответствуют кадрам в анимированном графике. Каждый кадр содержит свой собственный атрибут данных, а также другие параметры. Анимация обычно запускается и управляется с помощью элементов управления, определенных в параметрах `layout.sliders` и (или) `layout.updatemenus`.

Позиционирование макетных координат

Различные компоненты Plotly поддерживают атрибуты `x` и `y`, которые определяют положение компонента. К таким атрибутам могут относиться, например, `layout.xaxis.domain`, `layout.legend.x` или `layout.annotation[].x`.

Позиционирование элементов макета в макетных координатах не соответствует экранному позиционированию в пикселах. Макетные координаты — это доли, которые приходятся на компоненты. Например, если два графика занимают половину пространства, тогда каждому приходится по $^1/_2$ доли. Макетные координаты определяются двумя характеристиками:

- начальными координатами: (0,0) в `layout.margin.l`, `layout.margin.b`;
- шириной и высотой: (1,1) в `layout.width-layout.margin.r`, `layout.height-layout.margin.t`.

Здесь символы `l`, `r`, `t` и `b` обозначают сокращения от английских слов: left (левый), right (правый), top (верхний) и bottom (нижний). Значения макетных координат могут быть меньше 0 и больше 1 — тогда они относятся к областям в пределах границ фигуры.

Декартовая система координат

В Plotly наиболее часто используются графические окна типа `subplots` — двумерные графики в декартовой системе координат. Фигуры таких окон поддерживают атрибуты `xaxis` и `yaxis`, значения которых должны относиться к соответствующим частям фигуры. Например, `xaxis=x` и `yaxis=y` (значения по умолчанию) отображает график (`trace`) в координатной сетке, которая определена в `layout.xaxis` и `layout.yaxis`. А вот график с параметрами `xaxis=x2` и `yaxis=y2` соответствует координатной сетке со значениями координат `layout.xaxis2` и `layout.yaxis2`. Атрибуты `layout.xaxis` и `layout.xaxis2` могут быть не прописаны явно — это значит, что они имеют значения по умолчанию.

Оси координат *X* и *Y* имеют свой тип (`type`), который может быть представлен:

- непрерывными значениями (`type=linear`, `type=log`);
- временными значениями (`type=date`), что соответствует датам;
- категориальными значениями (`type=categor`, `type=multicategory`).

В следующих разделах будут показаны примеры программного кода, изучая которые, вы сможете более детально познакомиться с параметрами фигур (графиков).

Знакомство с библиотекой plotly.py мы начнем с рассмотрения возможностей формирования статичной двумерной графики (2D). Для создания и отображения статичных графиков на HTML-страницах нужно последовательно выполнить следующие действия:

4. Получить массив данных по осям (`x`, `y`) для построения графика (данные могут быть загружены из Интернета, сформированы при обращении к специализированным функциям пользователя или заданы в виде числовых констант.
5. Сформировать фигуру (координатную сетку): `fig`.
6. Добавить к координатной сетке полученные данные: `fig.add_trace`.
7. Добавить к координатной сетке описание элементов графика: `fig.update_layout` (название графика, подписи осей, метки линий, размер и т. п.).
8. Трансформировать график в формат HTML: `fig.to_html`.
9. Загрузить график формата HTML в объект `context`.
10. Передать объект `context` в шаблон Django.

Далее мы рассмотрим несколько примеров визуализации данных формата 2D в шаблонах Django.

8.1.1. Создание статичных графиков (одна линия)

В качестве примера сформируем простейший 2D-график линии, построенной по точкам, для чего нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, активирующую нужный маршрут для формирования графика (листинг 8.1).

Листинг 8.1. Ссылка активации маршрута для построения графика линии

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `stat_graf_2D` (листинг 8.2).

Листинг 8.2. URL-маршрут stat_graf_2D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `stat_graf_2D` (листинг 8.3).

Листинг 8.3. Представление (view) stat_graf_2D

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- два списка с данными для графика: `x_data` — данные для оси *x* и `y_data` — данные для оси *y*;
- объект `data_fig` и в него переданы списки с данными, которые будут располагаться по осям графика *x* и *y*;
- элементы с описанием параметров графика (`layout`) — подписи для осей *x* и *y*, заголовок графика, ширина и высота графика в пикселах;
- объект `fig` (фигура, или собственно график) и в него переданы отображаемые данные (`data_fig`) и параметры элементов графика (`layout`).

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (листинг 8.4).

Листинг 8.4. Шаблон stat_graf_2d.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно простой шаблон — он содержит тег Django `{{ plot_div }}`, в котором и будет отображен график, сформированный в представлении. По сути, stat_graf_2d.html представляет собой универсальный шаблон, в котором можно отображать графики различной конфигурации.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **1. Статичный график 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.1.

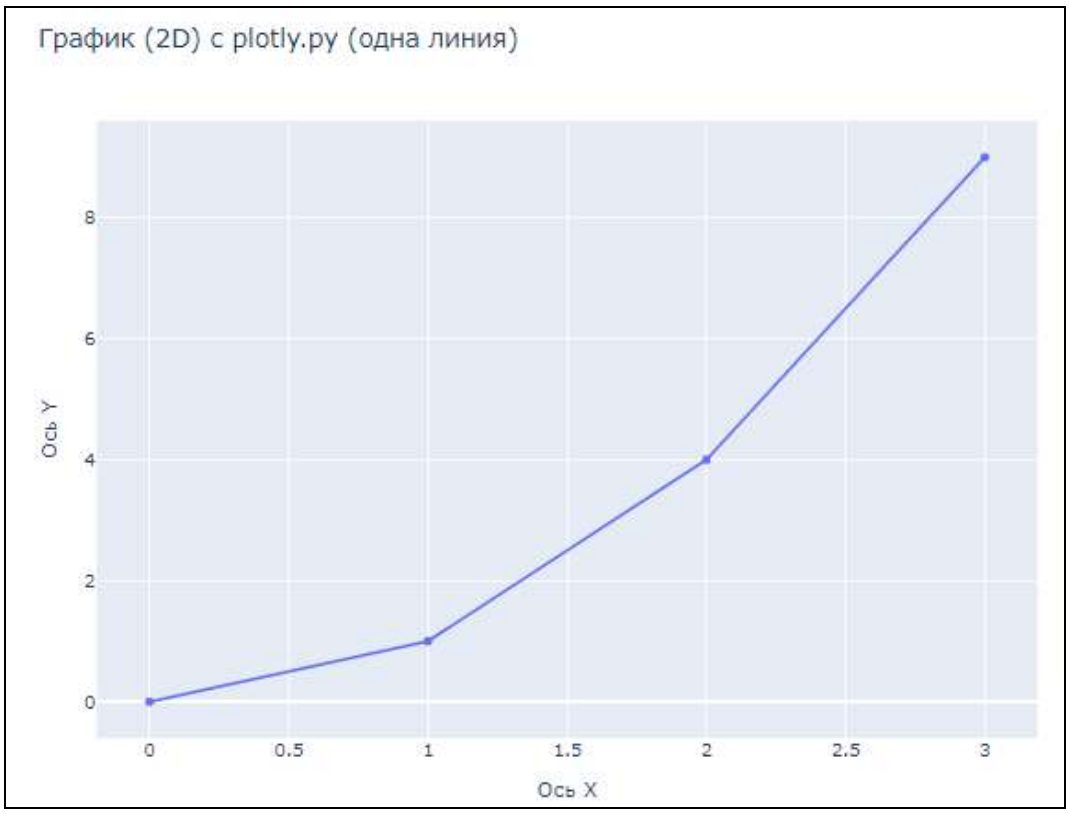


Рис. 8.1. Визуализация графика линии, построенной по точкам

Как здесь можно видеть, на графике отображена одна линия, график имеет подписанные оси *x* и *y*, сверху присутствует название графика.

Поскольку на графике имеется всего одна линия, то легенда, касающаяся этой линии, не приводится (даже если указать ее имя в параметрах графика), и чтобы показать легенду для одной линии графика, надо для параметра `layout` задать свойство `showlegend=True`.

8.1.2. Создание статичных графиков (несколько линий)

Если требуется на одной сетке показать несколько линий, то для каждой линии нужно сформировать свой набор данных, и для каждой группы данных создать свой набор объектов `trace`.

В качестве примера сформируем простейший 2D-график, на котором будут показаны две линии, построенные по точкам.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, активирующую нужный маршрут для формирования графика (листинг 8.5).

Листинг 8.5. Ссылка активации маршрута для построения графика двух линий

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `stat_graf_2D2` (листинг 8.6).

Листинг 8.6. URL-маршрут stat_graf_2D2

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `stat_graf_2D2` (листинг 8.7).

Листинг 8.7. Представление (view) stat_graf_2D2

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- две группы списков с данными для графика 1 (`x1`, `y1`) и для графика 2 (`x2`, `y2`;
- пустая сетка для будущего графика (`fig`).

На следующем шаге с использованием методов `fig.add_trace` в сетку графика загружены две группы списков сформированных данных: `x1`, `y1` и `x2`, `y2`. Для каждой линий создана легенда (`name`). Затем с помощью метода `fig.update_layout` к сетке добавлены параметры графика (подписи для осей, заголовок графика, ширина и высота графика в пикселах). Далее с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **2. Две линии на графике 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.2.

О цветных иллюстрациях глав 8 и 9

Поскольку цветные иллюстрации в черно-белом издании утрачивают свою информативность, все иллюстрации *глав 8* и *9*, содержащие важную для их понимания цветную составляющую, вынесены в PDF-файл, скачать который можно с сервера издательства «БХВ» по ссылке **https://zip.bhv.ru/XXXXXXXXXXXXX.zip**, а также со страницы книги на сайте **https://bhv.ru**.

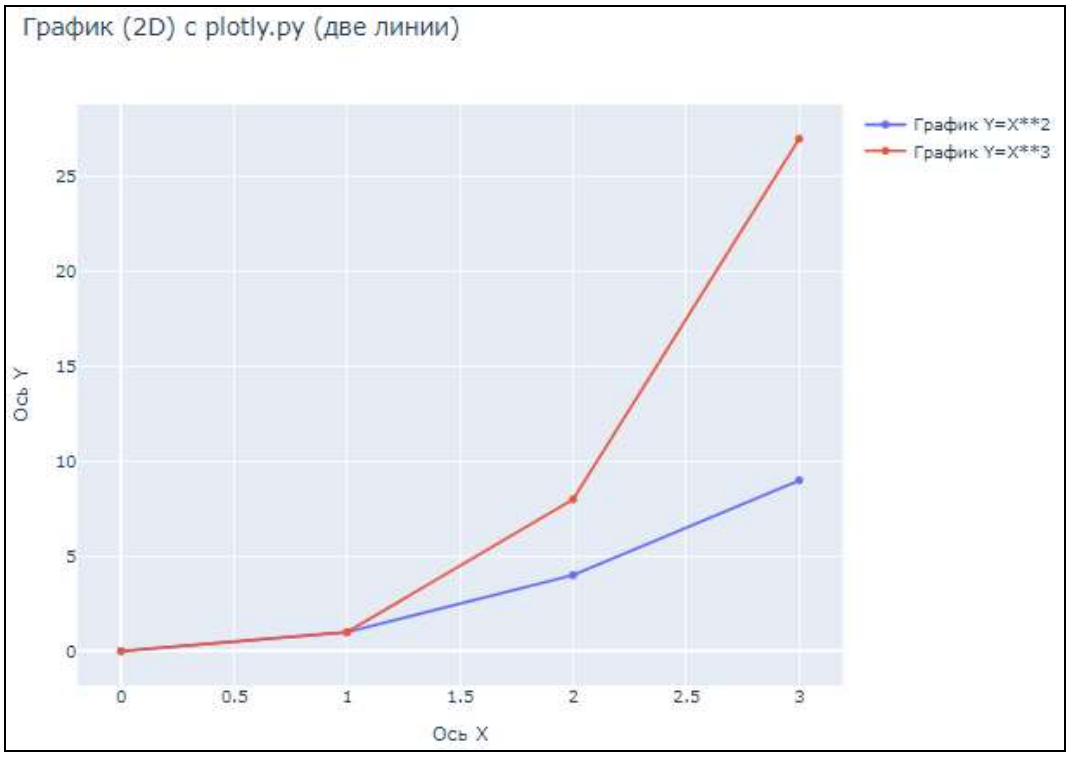


Рис. 8.2. Визуализация графика двух линий, построенных по точкам

Как здесь можно видеть, на графике в одной координатной сетке отображены две линии, график имеет подписанные оси *x* и *y*, сверху присутствует название графика.

Поскольку на графике имеются две линии, то рядом с сеткой показана легенда, касающаяся каждой линии. При наличии нескольких линий легенда, характеризующая сущность каждой линии, отображается всегда, даже если ее не задать в параметрах графика. Если пользователь не задал имена легендам, то по умолчанию им будут присвоены имена: `trace1`, `trace2`, `trace3` и т. д.

8.1.3. Отображение двух графиков в одном окне макета

Если требуется в одном окне создать две самостоятельные сетки, и в каждой сетке показать разные графики, то надо использовать функцию `make_subplots` библиотеки plotly.py и сформировать для каждой сетки свой набор данных.

В качестве примера сформируем два графика: на первом — отобразим линию, на втором — столбчатую гистограмму.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, активирующую нужный маршрут для формирования окна с двумя графиками (листинг 8.8).

Листинг 8.8. Ссылка активации маршрута для вывода двух графиков в одном окне

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `stat_2graf` (листинг 8.9).

Листинг 8.9. URL-маршрут stat_2graf

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `stat_2graf` (листинг 8.10).

Листинг 8.10. Представление (view) stat_2graf

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- две группы списков с данными для графика линии (`x1`, `y1`) и данными для столбчатой гистограммы (`x2`, `y2`);
- окно для будущих графиков (`fig`) с использованием функции `make_subplots`.

  В этой функции, по сути, сформирована таблица, состоящая и одной строки (`rows=1`) и двух колонок (`cols=2`). Для каждой колонки задано название графика: `subplot_titles=("График линии", "Гистограмма")`.

На следующем шаге с помощью методов `fig.add_trace` в сетки графиков загружены две группы списков сформированных данных (`x1`, `y1` и `x2`, `y2`):

- в первом методе `fig.add_trace` указано, что первый график будет представлять собой линию, построенную по точкам, задана для линии легенда и определено, в какой строке и в каком столбце мнимой таблицы будет расположен этот график:

  ```
  fig.add_trace(go.Scatter(x=x1, y=y1, name='График Y=X**2'), row=1, col=1)
  ```

- во втором методе `fig.add_trace` указано, что второй график будет представлять собой гистограмму, задана легенда и определено, в какой строке и в каком столбце мнимой таблицы будет расположен этот график:

  ```
  fig.add_trace(go.Bar(x=x2, y=y2, name='Гистограмма'), row=1, col=2)
  ```

После этого с использованием метода `fig.update_layout` к окну двух графиков добавлены параметры окна (заголовок для окна, ширина и высота окна в пикселах).

На следующем шаге выполняется формирование меток осей для двух графиков:

```
fig['layout']['xaxis']['title'] = 'Ось x-1'
fig['layout']['xaxis2']['title'] = 'Ось x-2'
fig['layout']['yaxis']['title'] = 'Ось y-1'
fig['layout']['yaxis2']['title'] = 'Ось y-2'
```

После этого с помощью метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **3. Два графика в одном окне**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.3.

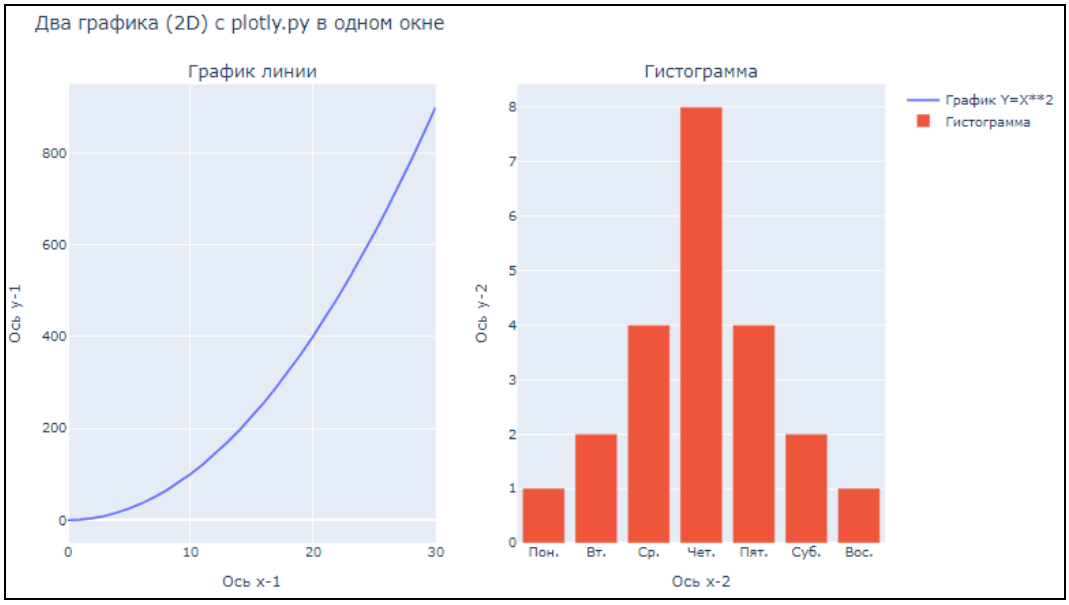


Рис. 8.3. Визуализация двух графиков в одном окне

Как здесь можно видеть, в одном окне показаны две координатные сетки и два разных графика. Каждый график имеет подписанные оси *x* и *y*, сверху присутствуют названия графиков. Вверху окна представлена надпись, относящаяся к обоим графикам.

Поскольку в окне содержатся два графика, то в правом углу окна приведена легенда, касающаяся каждого графика.

8.1.4. Отображение четырех графиков в одном окне макета

Если требуется в одном окне создать произвольную группу самостоятельных сеток и в каждой сетке показать разные графики, то для этого также надо использовать функцию `make_subplots` библиотеки plotly.py и сформировать для каждой сетки свой набор данных и свой набор параметров.

В качестве примера отобразим четыре графика в одном окне.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, активирующую нужный маршрут для формирования окна с группой из четырех графиков (листинг 8.11).

Листинг 8.11. Ссылка активации маршрута для вывода четырех графиков в одном окне

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `stat_4graf` (листинг 8.12).

Листинг 8.12. URL-маршрут stat_4graf

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `stat_4graf` (листинг 8.13).

Листинг 8.13. Представление (view) stat_4graf

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении на первом шаге с использованием функции `make_subplots` создано окно (`fig`) для отображения группы графиков:

```
fig = make_subplots(
        rows=2, cols=2,
        subplot_titles=("График 1", "График 2", "График 3", "График 4"))
```

Здесь указано, что в окне будет находиться мнимая таблица, состоящая из двух строк и двух колонок (всего четыре ячейки). В каждой ячейке должен располагаться свой график. Здесь же заданы наименования для каждого графика.

Далее с использованием метода `fig.add_trace` в окно (`fig`) добавлены четыре графика, и для каждого графика указаны следующие параметры:

- списки с данными для осей *x* и *y*;
- легенда (физический смысл отображаемых данных);
- номер ячейки мнимой таблицы (строки — `row` и столбца — `col`), в которой будет отображен график.

На следующем шаге с использованием методов `fig.update_xaxes` и `fig.update_yaxes` задаются названия осей *x* и *y* для всех четырех графиков, при этом указывается в какую ячейку мнимой таблицы нужно поместить каждое название:

```
# Обновить параметры для осей x (xaxis)
    fig.update_xaxes(title_text="Ось x-1", row=1, col=1)
    fig.update_xaxes(title_text="Ось x-2", range=[10, 50], row=1, col=2)
    fig.update_xaxes(title_text="Ось x-3", showgrid=False, row=2, col=1)
    fig.update_xaxes(title_text="Ось x-3", type="log", row=2, col=2)

    # Обновить параметры для осей y (yaxis)
    fig.update_yaxes(title_text="Ось y-1", row=1, col=1)
    fig.update_yaxes(title_text="Ось y-2", range=[40, 80], row=1, col=2)
    fig.update_yaxes(title_text="Ось y-3", showgrid=False, row=2, col=1)
    fig.update_yaxes(title_text="Ось y-4", row=2, col=2)
```

После этого с использованием метода `fig.update_layout` к окну четырех графиков добавлены параметры окна — заголовок для окна и высота окна в пикселах.

Затем с использованием метода `fig.to_html` выполнено преобразование графиков в формат HTML, и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **4. Четыре графика в одном окне**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.4.

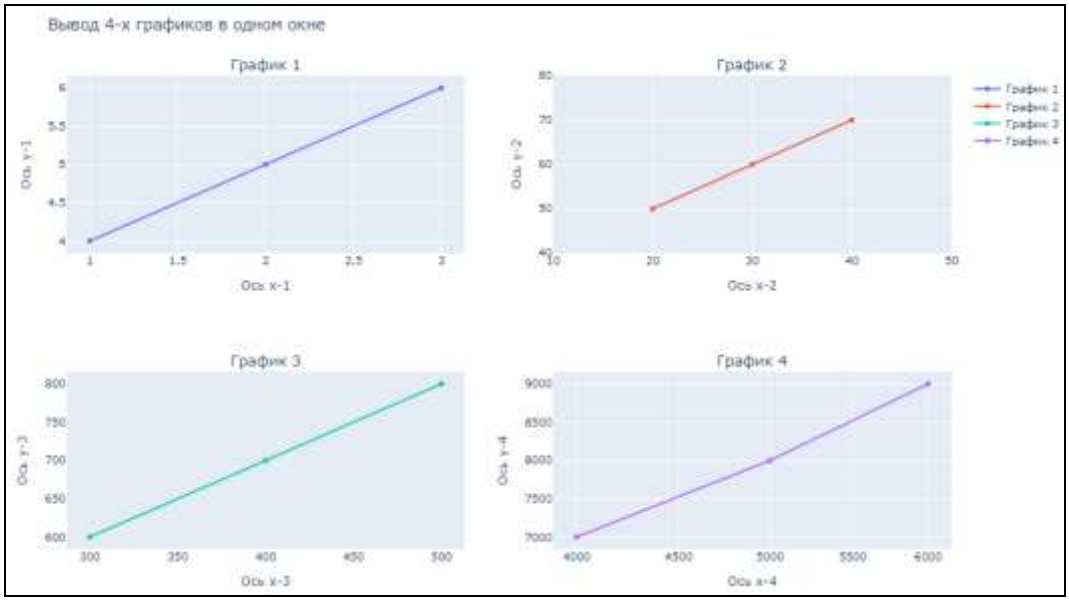


Рис. 8.4. Визуализация четырех графиков в одном окне

Как здесь можно видеть, в одном окне показаны четыре координатные сетки и четыре разных графика. Каждый график имеет подписанные оси *x* и *y*, сверху присутствуют названия графиков. Вверху окна представлена надпись, относящаяся ко всем графикам.

Поскольку в окне содержатся четыре графика, то в правом углу окна приведена легенда, касающаяся каждого графика.

Таким образом, с использованием функции `make_subplots` можно в едином окне создать мнимую таблицу с произвольным количеством ячеек, в которых отобразить совершенно разные графики.

8.1.5. Создание 2D-диаграммы рассеяния

Если требуется на одной сетке показать рассеяние множества точек, то для каждой группы точек нужно сформировать свой набор данных и свой набор объектов `trace`.

В качестве примера сформируем 2D-диаграмму рассеяния, на которой будут показаны две группы точек.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, активирующую нужный маршрут для формирования диаграммы (листинг 8.14).

Листинг 8.14. Ссылка активации маршрута для построения диаграммы рассеяния

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут stat_5graf (листинг 8.15).

Листинг 8.15. URL-маршрут stat_5graf

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `stat_5graf` (листинг 8.16).

Листинг 8.16. Представление (view) stat_5graf

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении с использованием функции `np.random.randint` библиотеки NumPy созданы две группы списков с данными о точках рассеяния (`x1`, `y1`) и (`x2`, `y2`). Затем создана пустая сетка для будущей диаграммы (`fig`). На следующем шаге с помощью метода `fig.add_trace` в сетку диаграммы загружены две группы сформированных данных (`x1`, `y1`) и (`x2`, `y2`) и для каждой группы точек создана легенда (`name`). Далее с использованием метода `fig.update_layout` к сетке добавлены параметры диаграммы (подписи для осей *x* и *y*, заголовок, ширина и высота в пикселах). После этого с помощью метода `fig.to_html` выполнено преобразование диаграммы в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **5. Диаграмма рассеяния 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.5.

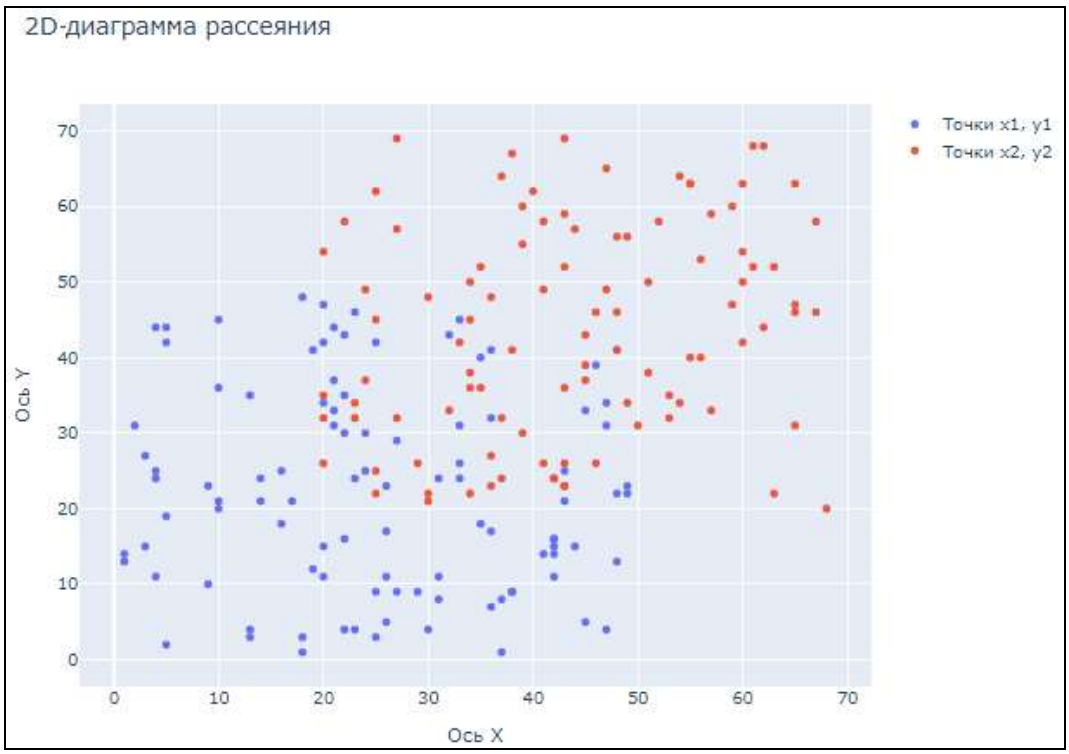


Рис. 8.5. Визуализация 2D-диаграммы рассеяния

Как здесь можно видеть, на диаграмме в одной координатной сетке отображены две группы точек, сетка имеет подписанные оси *x* и *y*, сверху присутствует название диаграммы.

Поскольку на диаграмме имеются две группы точек, то рядом с сеткой показана легенда, касающаяся каждой группы точек. При наличии нескольких групп точек легенда, характеризующая сущность каждой группы, отображается всегда, даже если ее не задать в параметрах диаграммы. Если пользователь не задал имена легендам, то по умолчанию легендам будут присвоены имена: `trace1`, `trace2`, `trace3` и т. д.

8.1.6. Создание круговой диаграммы

Круговая диаграмма
Круговая диаграмма — это окружность, разделенная на сектора, символизирующие долю или процент содержания неких элементов в общем объекте. Каждый сектор несет наименование и числовое значение. Круговые диаграммы имеют бо̲льшую наглядность, если каждый сектор выделен своим цветом.

В этом примере мы построим простейшую круговую диаграмму, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для построения диаграммы (листинг 8.17).

Листинг 8.17. Ссылка активации маршрута для построения диаграммы

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `krug_2D` (листинг 8.18).

Листинг 8.18. URL-маршрут krug_2D

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `krug_2D` (листинг 8.19).

Листинг 8.19. Представление (view) krug_2D

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении с использованием библиотеки pandas создается набор данных (`dataframe`), состоящий из пяти элементов (объект `fruit`). В объекте `fruit` содержится информация об объемах продаж фруктов в супермаркете за определенный период. В наборе данных `dataframe` массив `name` представляет собой наименования фруктов, а массив `number` — количество проданных фруктов.

На следующем шаге делается расчет суммарного объема продаж (объект `s`) и определяется доля каждого вида фруктов в этом объеме (`fruit['number']`). Эта доля округляется до двух знаков после запятой.

Затем с помощью метода `go.Figure` создается фигура `fig` (диаграмма) и выполняется заполнение диаграммы данными (`fig.add_trace`). Для заполнения данными круговой диаграммы используется метод `go.Pie`, в который передаются следующие параметры:

- `values=fruit['number']` — числовые значение для каждого сектора;
- `labels=fruit['name']` — наименования каждого сектора.

Далее создаются элементы с описанием параметров макета (`layout`): заголовок (`title`), ширина макета (`width`) и высота макета (`height`). По умолчанию легенда диаграммы будет расположена с правой стороны от макета.

Совет
Если нужно расположить легенду под макетом, то надо дополнительно задать еще один параметр: `legend_orientation="h"` — см. закомментированные строки (между тройными кавычками).

После этого с использованием метода `fig.to_html` выполнено преобразование диаграммы в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **6. Круговая диаграмма 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.6.

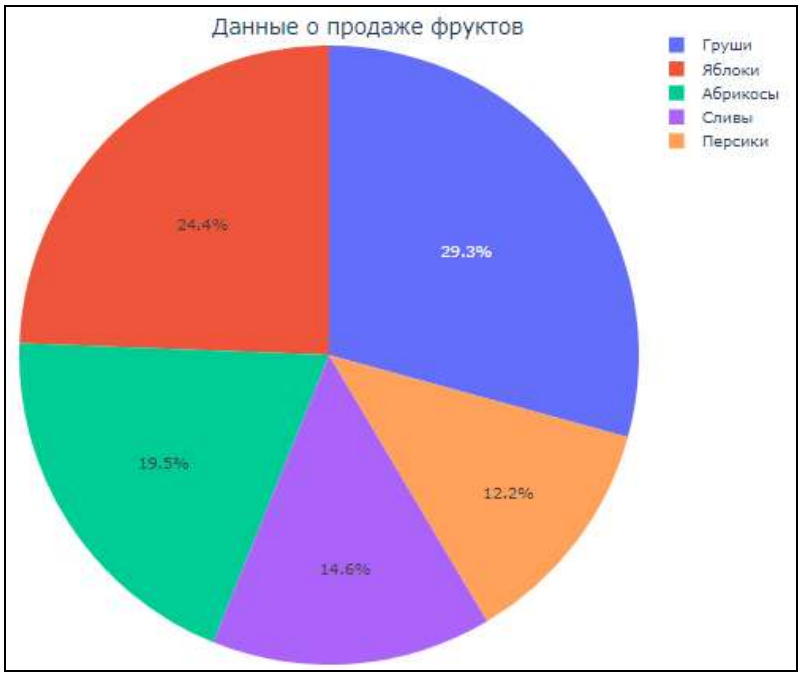


Рис. 8.6. Визуализация круговой диаграммы

Как здесь можно видеть, на графике отображены данные в виде круговой диаграммы с секторами разных цветов. Если навести курсор мыши на любой из секторов, то рядом с ним появится дополнительный поясняющий текст. Такие подсказки окажутся очень полезными в тех случаях, когда диаграмма состоит из большого количества секторов.

8.1.7. Создание столбчатой гистограммы

Если требуется на одной сетке показать столбчатую гистограмму, то в качестве параметра `x` необходимо задать подписи для столбцов, а в качестве параметра `y` — значение для каждого столбца. Если параметр `x` не задавать, то по умолчанию столбцы будут пронумерованы: 1, 2, 3, … и т. д.

В качестве примера сформируем простейший 2D-график, на котором будут показаны 7 столбцов столбчатой гистограммы.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика с гистограммой (листинг 8.20).

Листинг 8.20. Ссылка активации маршрута для построения гистограммы

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `stat_gist` (листинг 8.21).

Листинг 8.21. URL-маршрут stat_gist

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `stat_gist` (листинг 8.22).

Листинг 8.22. Представление (view) stat_gist

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- список с подписями для столбцов гистограммы (`x`) и значения для каждого столбца (`y`);
- пустая сетка для будущего графика (`fig`).

На следующем шаге с использованием метода `fig.add_trace` в сетку графика загружены исходные данные (`x`, `y`). Затем с помощью метода `fig.update_layout` к сетке добавлены параметры графика (подписи для осей, заголовок графика, ширина и высота графика в пикселах). После этого с использованием метода `fig.to_html` выполнено преобразование гистограммы в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **7. Столбчатая гистограмма 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.7.

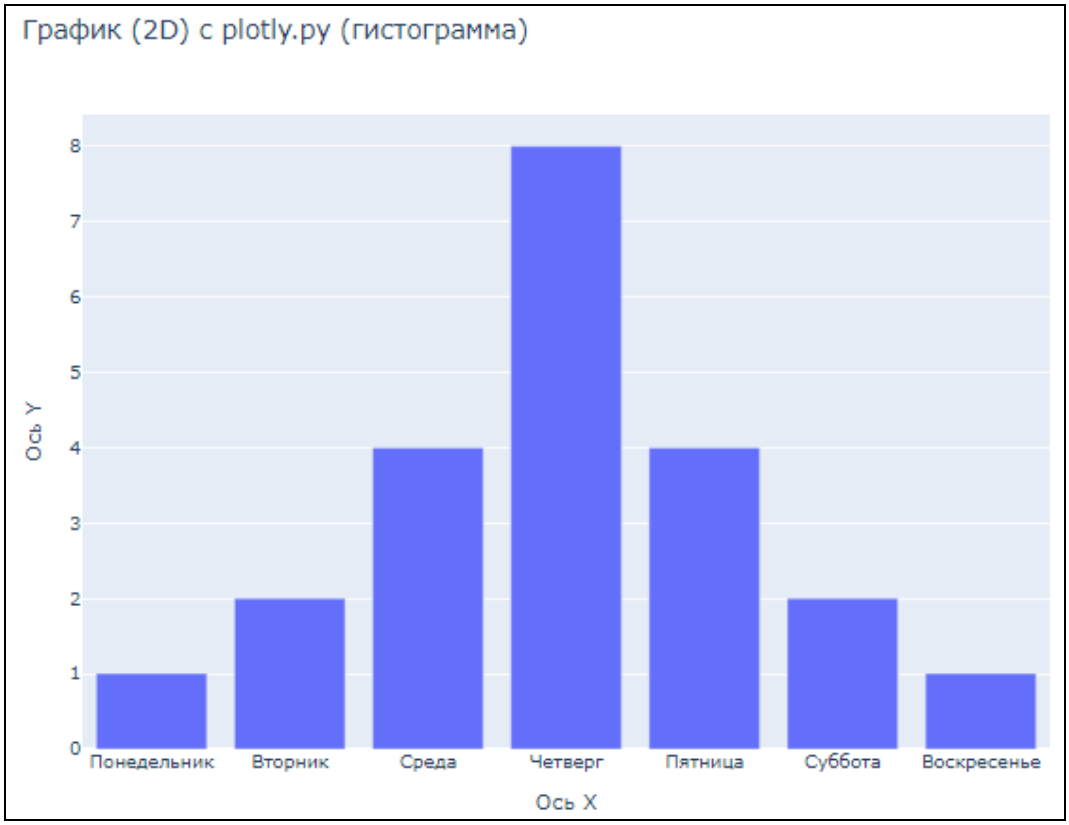


Рис. 8.7. Визуализация столбчатой гистограммы

Как здесь можно видеть, на графике в координатной сетке отображена гистограмма, которая имеет подписанные оси *x* и *y*, маркеры с наименованиями столбцов, сверху присутствует название графика.

8.1.8. Создание столбчатой гистограммы и сглаживающей кривой

Гистограмма

Гистограмма — это визуальное представление распределения набора количественных данных. Для построения гистограммы первым шагом является разделение всего диапазона значений на ряд интервалов с последующим подсчетом количества значений, попадающих в каждый интервал. Гистограммы дают грубое представление о плотности базового распределения данных, однако данные гистограмм можно аппроксимировать плавными кривыми.

В этом примере мы сгенерируем набор экспериментальных данных и на их основе построим гистограмму и аппроксимирующую ее кривую нормального распределения. Для генерации данных можно использовать библиотеку SciPy — в частности, ее модуль `stats`. Необходимые для расчета ее модули подключаются с использованием следующей инструкции:

```
from scipy.stats import norm
```

Для построения гистограммы нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для построения гистограммы (листинг 8.23).

Листинг 8.23. Ссылка активации маршрута для построения гистограммы

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `g_norm_2D` (листинг 8.24).

Листинг 8.24. URL-маршрут g_norm_2D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `g_norm_2D` (листинг 8.25).

Листинг 8.25. Представление (view) g_norm_2D

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении с использованием библиотеки NumPy формируются набор случайных данных нормального распределения (объект `r`) и набор данных для построения теоретической кривой нормального распределения (объекты `x_norm` и `y1`).

Внимание!

В листинге 8.25 ссылка на библиотеку SciPy, которую также можно использовать в качестве генератора чисел нормального распределения, закомментирована, поскольку обращаться с ней следует с осторожностью. Библиотека SciPy прекрасно работает на локальном сервере разработчика, но может отказаться работать после развертывания приложения на публичном сервере Apache. Автор столкнулся с этой проблемой, как и многие другие разработчики. Наличие строки `from scipy.stats import norm` просто «обваливает» все приложение без диагностических сообщений, и даже администраторы серверной площадки порой не могут найти причину такого сбоя. Похожие проблемы могут возникнуть при использовании библиотеки Skimage.

Далее с помощью метода `go.Figure` создается объект `fig` (фигура, диаграмма), который с использованием метода `fig.add_trace` заполняется случайными (экспериментальными) и теоретическими данными.

На следующем шаге создаются элементы с описанием параметров макета (`layout`):

- `title` — заголовок гисторраммы;
- `legend` — легенда;
- `width` — ширина;
- `height` — высота;
- `margin` — отступы от края.

После этого с использованием метода `fig.to_html` выполняется преобразование гистограммы в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **8. Гистограмма нормального распределения 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.8.

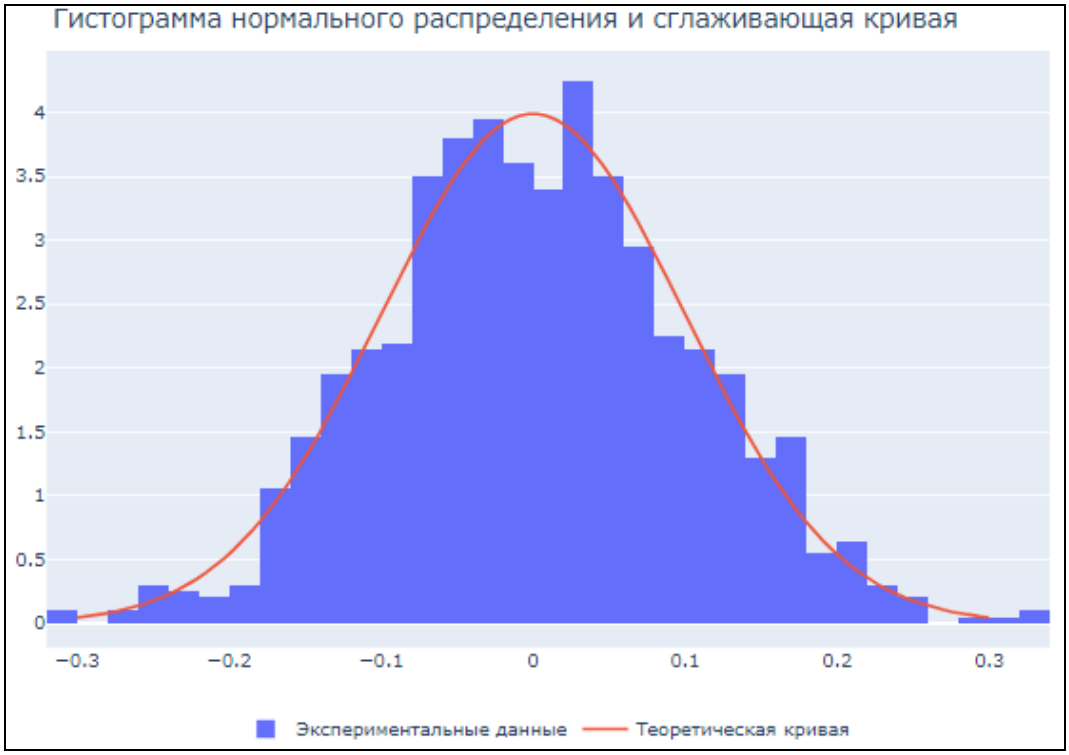


Рис. 8.8. Визуализация гистограммы нормального распределения и сглаживающей кривой

Как здесь можно видеть, на графике отображены данные в виде гистограммы со сглаживающей кривой. Если навести курсор мыши на любой из элементов гистограммы или на кривую, то рядом с ними появится дополнительный поясняющий текст.

8.1.9. Создание графиков с тепловой картой

> ***Тепловые карты***
> Тепловые карты — это различные дополнения к визуализации числовых данных. Как правило, тепловая карта представляет собой цветную шкалу со спектром цветов, где каждый оттенок цвета соответствует определенному числовому значению. Тепловая карта может быть показана в виде полупрозрачного слоя на некой поверхности (например, на географической карте), где

различной глубиной палитры цветов выделяется числовая информация (например, плотность населения)

Основная цель тепловых карт — повысить наглядность визуализации общего количества мест или событий, включенных в набор данных, а также направить внимание пользователей на наиболее важные аспекты отображаемых данных. В последние годы в области аналитики данных наблюдается рост использования тепловых карт — во многом это происходит, благодаря гибкости и эффективности, с которыми они выявляют повторяющиеся тенденции.

В качестве примера сформируем простейший 2D-график синусоиды, построенной по точкам различных цветовых оттенков, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.26).

Листинг 8.26. Ссылка активации маршрута для построения графика линии

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `inter_graf6_2D` (листинг 8.27).

Листинг 8.27. URL-маршрут inter_graf6_2D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `inter_graf6_2D` (листинг 8.28).

Листинг 8.28. Представление (view) inter_graf6_2D

Примечание.

В демо-версии книги удалены листинги программ.

Это представление для генерирования данных синусоиды и формирования палитры цветов тепловой карты задействует дополнительную функцию `sin(x)`.

Сначала с использованием библиотеки NumPy в нем генерируется массив данных `x` для построения синусоиды:

```
x = np.arange(0, 9.5, 0.1)
```

Затем создается объект `fig` (пустой макет), к которому добавлены данные (`trace`). При формировании этих данных и нашла применение дополнительная функция `sin()` — на ее основе получены данные для оси *y* (`y=sin(x)`) и сформирована палитра цветов для тепловой карты (`color=sin(x)`). Все параметры, связанные с тепловой картой, загружены в свойство `marker`, которое имеет тип «словарь».

На следующих шагах созданы элементы с описанием параметров графика (`layout`): заголовок (`title`), подписи для осей *x* и *y*, указано положение легенды и обязательность ее вывода (`showlegend=True`), а данные графика добавлены к основной фигуре (`fig.update_traces`).

После этого с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **9. График с тепловой картой 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.9.

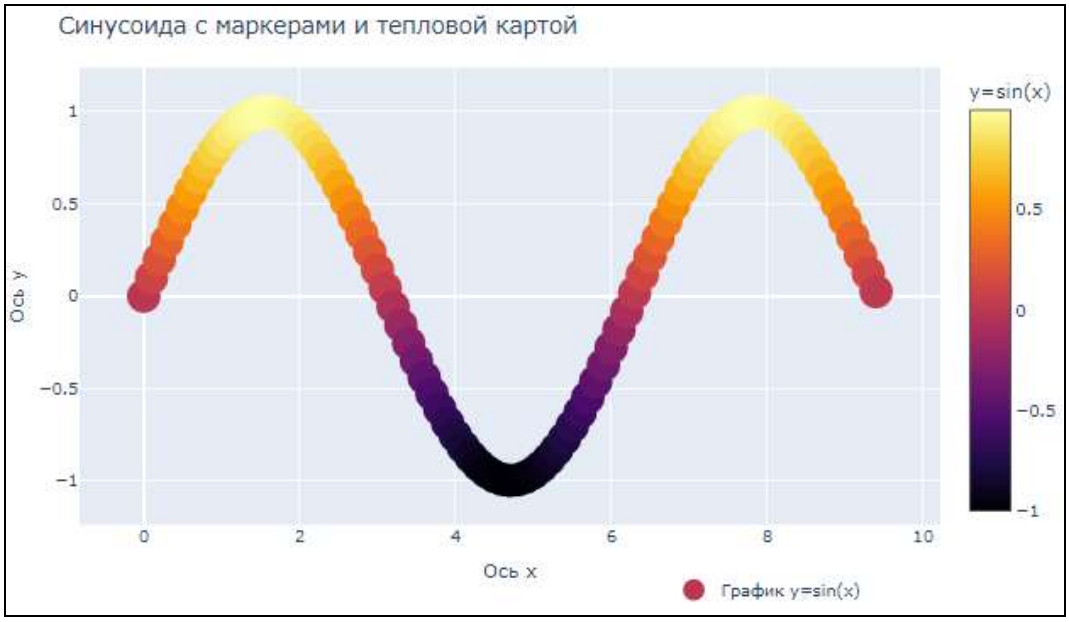


Рис. 8.9. Визуализация синусоиды в виде маркеров с тепловой картой

Как здесь можно видеть, на графике отображена цветная синусоида с маркировкой значений в виде различных цветов. Максимальные значения имеют более светлые и яркие оттенки, минимальные значения — более темные тона. Рядом с графиком показана шкала тепловой карты с палитрой цветов, относящихся к различным значениям синусоиды.

8.1.10. Создание пузырьковой диаграммы

> ***Пузырьковые диаграммы***
> Пузырьковые диаграммы — это такой тип 2D-диаграмм, в которых отображаются данные в трех измерениях. Каждый элемент связанных данных представляется в виде маркера (или диска), выражающего два значения через положение маркера на осях *x* и *y* , а третье — через размер маркера. Пузырьковые диаграммы могут облегчить понимание социальных, экономических, медицинских и других научных взаимосвязей. В целом, они являются разновидностью диаграмм рассеяния, в которых точки данных заменяются маркерами разных размеров или разного цвета (их еще называют *пузырьками*).

В качестве примера сформируем простейшую пузырьковую диаграмму с одноцветными маркерами разных размеров, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования диаграммы (листинг 8.29).

Листинг 8.29. Ссылка активации маршрута для формирования диаграммы

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `bubble_1` (листинг 8.30).

Листинг 8.30. URL-маршрут bubble_1

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `bubble_1` (листинг 8.31).

Листинг 8.31. Представление (view) bubble_1

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении с использованием метода `go.Figure` создается фигура `fig` (график). Этому методу мы передаем следующие параметры:

- `data` — данные для осей *x* и *y*;
- `mode='markers'` — режим отображения данных в виде маркеров (пузырьков);
- `marker` — словарь, в котором определены размеры маркеров (`size`) в виде списка.

На следующем шаге создаются элементы с описанием параметров диаграммы (`layout`): заголовок (`title`), подписи для осей *x* и *y*.

После этого с использованием метода `fig.to_html` выполняется преобразование диаграммы в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **10. Пузырьковая диаграмма 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.10.

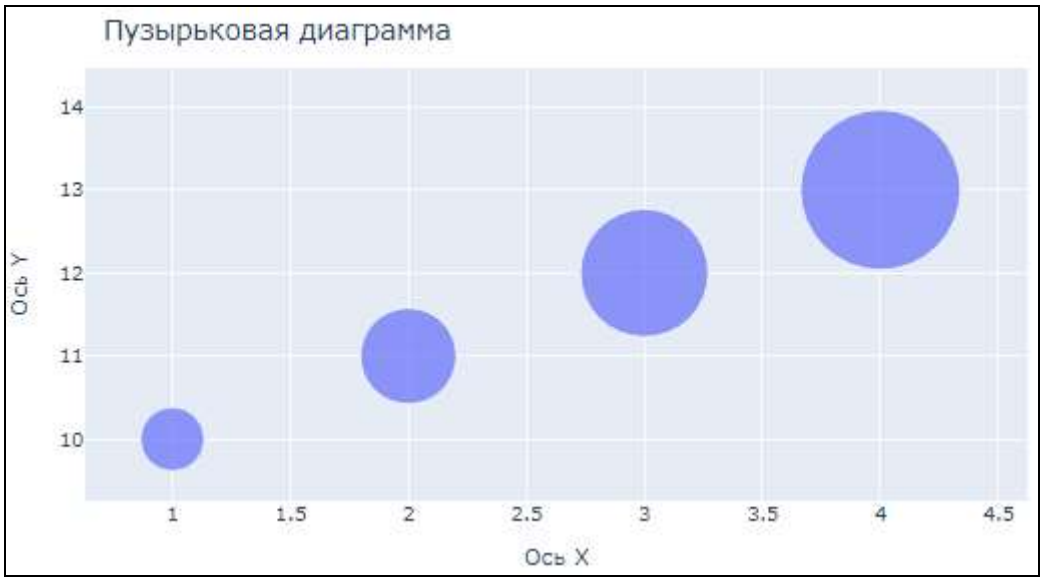


Рис. 8.10. Визуализация простой пузырьковой диаграммы

Как здесь можно видеть, на диаграмме отображены данные в виде маркеров различных размеров. Размер маркера определяет значение данных для третьего измерения.

8.1.11. Пузырьковая диаграмма с цветными маркерами

В пузырьковых диаграммах для выделения значений тех или иных данных можно использовать не только размер маркеров, но и их цвет.

В качестве примера сформируем простейшую пузырьковую диаграмму с разноцветными маркерами разных размеров, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования диаграммы (листинг 8.32).

Листинг 8.32. Ссылка активации маршрута для формирования диаграммы

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `bubble_2` (листинг 8.33).

Листинг 8.33. URL-маршрут bubble_2

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `bubble_2` (листинг 8.34).

Листинг 8.34. Представление (view) bubble_2

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении с использованием метода `go.Figure` создается фигура `fig` (график). Этому методу мы передаем следующие параметры:

- `data` — данные для осей *x* и *y*;
- `mode='markers'` — режим отображения данных в виде маркеров (пузырьков);
- `marker` — словарь, в котором определены размеры маркеров (`size`) в виде списка.

В этом примере для маркеров (пузырьков) заданы следующие параметры:

- `color` — список цветов;
- `opacity` — список степени прозрачности;
- `size` — список размеров.

На следующем шаге создаются элементы с описанием параметров диаграммы (`layout`): заголовок (`title`), подписи для осей *x* и *y*.

После этого с использованием метода `fig.to_html` выполняется преобразование диаграммы в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **11. Пузырьковая диаграмма с цветными маркерами 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.11.

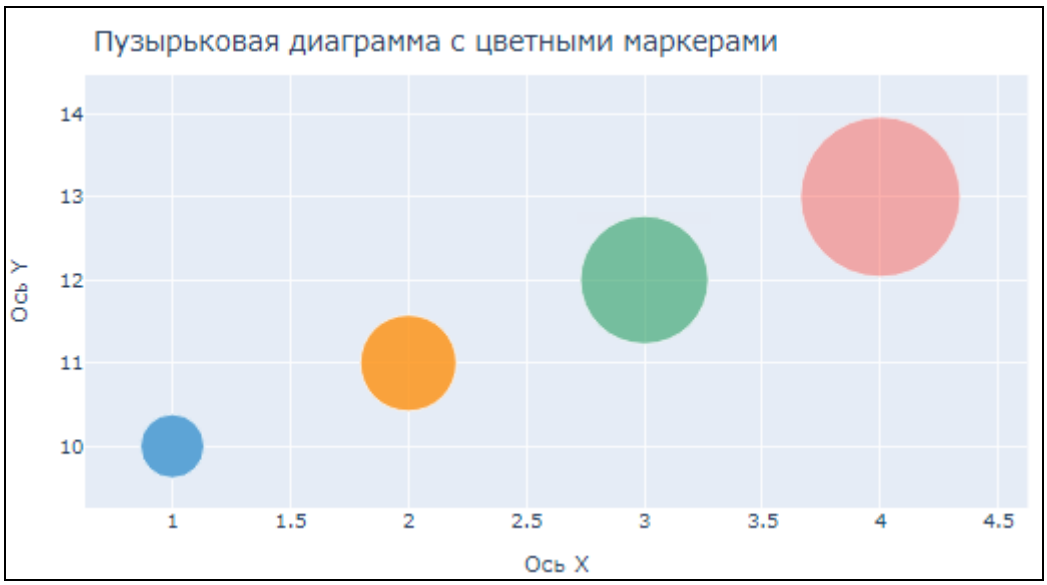


Рис. 8.11. Визуализация пузырьковой диаграммы с разноцветными маркерами разных размеров

Как здесь можно видеть, на диаграмме отображены данные в виде маркеров различных размеров и разных цветов. Размер и цвет маркера определяют значение данных для третьего измерения.

8.1.12. Пузырьковая диаграмма с масштабируемыми маркерами

В пузырьковых диаграммах данные, определяющие размер маркера, могут иметь достаточно большую величину (тысячи, миллионы и т. п.). Если для маркеров указывать такие абсолютные размеры, то они просто не поместятся в макете графика, а вернее, заполнят всё его пространство. Поэтому для данных с большими значениями маркеры нужно масштабировать — то есть уменьшать из размерность на заданную величину.

В качестве примера сформируем простейшую пузырьковую диаграмму с масштабируемыми размерами маркеров, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования диаграммы (листинг 8.35).

Листинг 8.35. Ссылка активации маршрута для формирования диаграммы

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `bubble_3` (листинг 8.36).

Листинг 8.36. URL-маршрут bubble_3

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `bubble_3` (листинг 8.37).

Листинг 8.37. Представление (view) bubble_3

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создан список с размерами маркеров (`size`), имеющими величину в несколько тысяч. Затем с использованием метода `go.Figure` создается фигура `fig` (график). Этому методу мы передаем следующие параметры:

- `x` — координаты точек для оси *x*;
- `y` — координаты точек для оси *y*;
- `mode='markers'` — режим отображения данных в виде маркеров (пузырьков);
- `marker` — словарь, в котором определены параметры маркеров.

Здесь для маркеров (пузырьков) заданы следующие параметры:

- `size` — список абсолютных размеров маркеров;
- `sizemode='area'` — режим отображения маркеров;
- `sizeref` — формула, по которой масштабируются абсолютные значения маркеров;
- `sizemin` — минимальный абсолютный размер маркера.

В этом примере абсолютные значения маркеров масштабируются по следующей формуле:

```
sizeref=2. * max(size) / (40. ** 2),
```

> ***Примечание***
> Обратите внимание, что установка `'sizeref'` на значение больше 1 уменьшает размеры отображаемых маркеров, а установка `'sizeref'` на значение меньше 1 увеличивает размеры отображаемых маркеров.

На следующем шаге создаются элементы с описанием параметров диаграммы (`layout`): заголовок (`title`), подписи для осей *x* и *y*.

После этого с использованием метода `fig.to_html` выполняется преобразование диаграммы в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **12. Пузырьковая диаграмма с масштабируемыми маркерами 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.12.

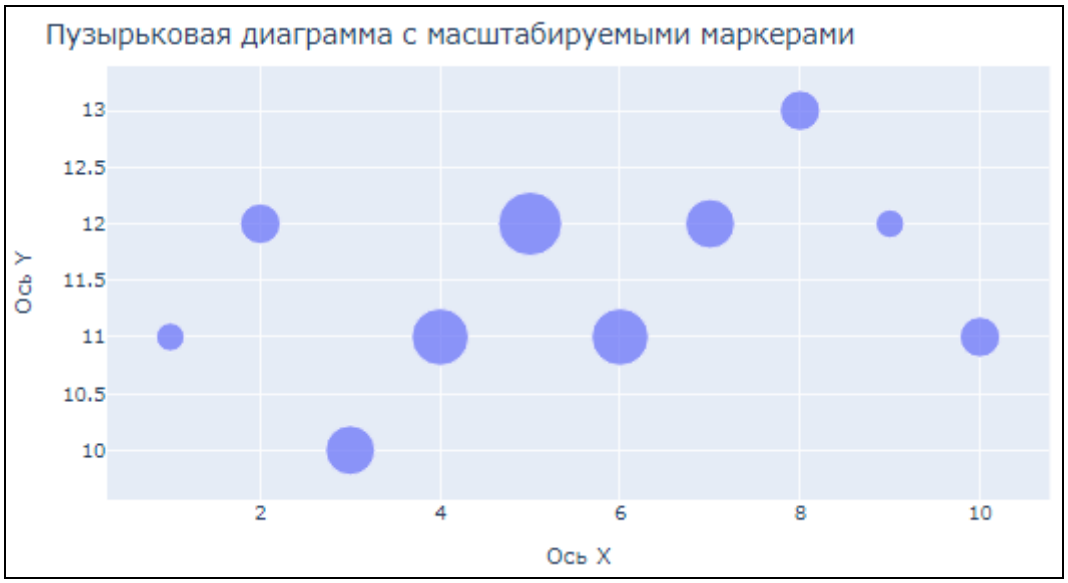


Рис. 8.12. Визуализация пузырьковой диаграммы с масштабируемыми маркерами

Как здесь можно видеть, на диаграмме отображены данные в виде маркеров различных размеров. Несмотря на то, что исходные размеры маркеров были в пределах от 2000 до 10000, они успешно масштабировались и поместились в пределах макета диаграммы.

8.1.13. Пузырьковая диаграмма с поясняющим текстом на маркерах

В пузырьковых диаграммах в рамках макета могут быть разбросаны десятки маркеров различных размеров и цветов. Для того чтобы конечный пользователь мог определить, какое значение имеет тот или иной маркер, на них нужно поместить поясняющий текст.

В качестве примера сформируем простейшую пузырьковую диаграмму с текстовыми метками на маркерах, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования диаграммы (листинг 8.38).

Листинг 8.38. Ссылка активации маршрута для формирования диаграммы

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `bubble_4` (листинг 8.39).

Листинг 8.39. URL-маршрут bubble_4

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `bubble_4` (листинг 8.40).

Листинг 8.40. Представление (view) bubble_4

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении с использованием метода `go.Figure` создается фигура `fig` (график). Этому методу мы передаем следующие параметры:

- `x` — координаты точек для оси *x*;
- `y` — координаты точек для оси *y*;
- `text` — список текстовых меток для каждого маркера;
- `mode='markers'` — режим отображения данных в виде маркеров (пузырьков);
- `marker` — словарь, в котором определены параметры маркеров.

Здесь для маркеров (пузырьков) заданы следующие параметры:

- `color` — список цветов маркеров;
- `size` — список размеров маркеров.

На следующем шаге создаются элементы с описанием параметров диаграммы (`layout`): заголовок (`title`), подписи для осей *x* и *y*.

После этого с использованием метода `fig.to_html` выполняется преобразование диаграммы в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **13. Пузырьковая диаграмма с текстовыми метками 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.13.



Рис. 8.13. Визуализация пузырьковой диаграммы с текстовыми метками на маркерах

Как здесь можно видеть, на диаграмме отображены данные в виде маркеров различных размеров и цветов. Если навести курсор на цветной маркер, то рядом с ним появится поясняющий текст со следующей информацией:

- координаты *x* и *y* размещения метки в макете диаграммы;
- текст с расшифровкой данных, относящихся к этой метке.

Такие подсказки окажутся очень полезными в тех случаях, когда на график вынесено большое количество меток.

8.1.14. Пузырьковая диаграмма с тепловой картой

Если в пузырьковой диаграмме много типов данных или количество данных меняется динамично, то вручную задать цвета маркерам окажется проблематично. В этом случае можно воспользоваться тепловой картой, благодаря которой каждому типу данных будет присвоен свой цветовой оттенок.

В качестве примера сформируем простейшую пузырьковую диаграмму с тепловой картой, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования диаграммы (листинг 8.41).

Листинг 8.41. Ссылка активации маршрута для формирования диаграммы

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `bubble_5` (листинг 8.42).

Листинг 8.42. URL-маршрут bubble_5

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `bubble_5` (листинг 8.43).

Листинг 8.43. Представление (view) bubble_5

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении с использованием метода `go.Figure` создается фигура `fig` (график). Этому методу мы передаем следующие параметры:

- `x` — координаты точек для оси *x*;
- `y` — координаты точек для оси *y*;
- `text` — список текстовых меток для каждого маркера;
- `mode='markers'` — режим отображения данных в виде маркеров (пузырьков);
- `marker` — словарь, в котором определены параметры маркеров.

Здесь для маркеров (пузырьков) заданы следующие параметры:

- `color` — список цветов маркеров;
- `size` — список размеров маркеров;
- `showscale=True` — показать тепловую карту.

На следующем шаге создаются элементы с описанием параметров диаграммы (`layout`): заголовок (`title`), подписи для осей *x* и *y*.

После этого с использованием метода `fig.to_html` выполняется преобразование диаграммы в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **14. Пузырьковая диаграмма с тепловой картой 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.14.

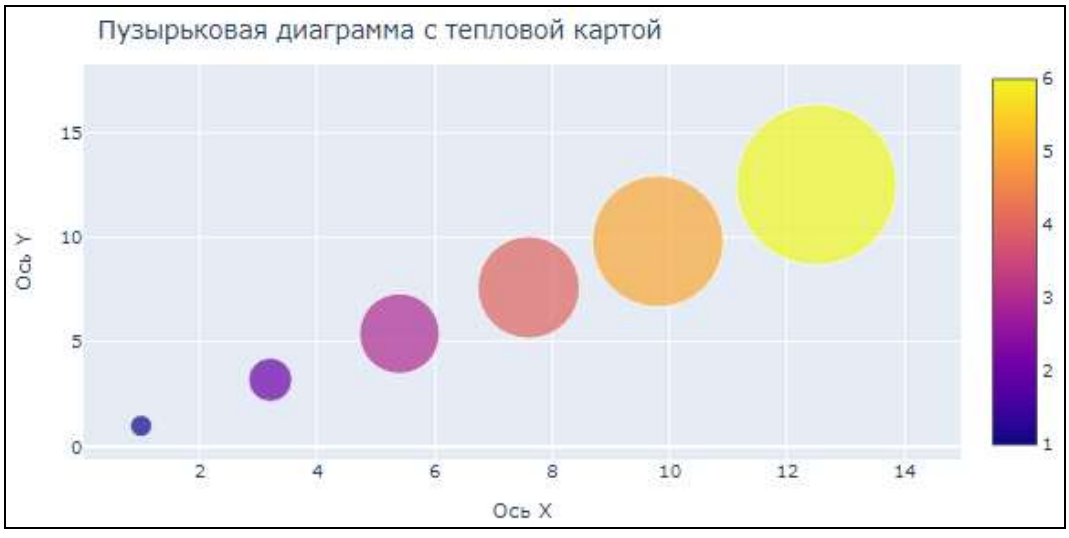


Рис. 8.14. Визуализация пузырьковой диаграммы с тепловой картой

Как здесь можно видеть, на диаграмме отображены данные в виде маркеров различных размеров и цветов. Маркеры находятся на разных уровнях, при этом цвет каждого маркера соответствует цвету на шкале тепловой карты.

8.1.15. Отображение статичных элементов на географических картах

Библиотека Plotly поддерживает множество разных видов диаграмм для визуализации данных, и показать здесь их все практически невозможно, однако нельзя не рассмотреть еще одну замечательную возможность этой библиотеки — вывод данных на географические карты. Для этого в библиотеке необходимо задействовать метод `go.Scattermapbox`.

В этом примере мы воспользуемся картой OSM (OpenStreetMap, открытая карта улиц) — это некоммерческий веб-картографический проект, в рамках которого создана карта мира, открытая для свободного и бесплатного использования. Она коренным образом отличается от коммерческих карт (например, от карты Google Maps), подключиться к которым можно только с разрешения правообладателя.

При работе с географическими картами важными атрибутами для отображения различных объектов являются широта и долгота их местоположения. Мы задействуем здесь немного модифицированный файл формата CSV с населенными пунктами РФ и их координатами. Оригинальный файл вы можете найти в Интернете по следующим ссылкам:

- **github.com/hflabs/city/blob/master/city.csv**;
- **https://raw.githubusercontent.com/hflabs/city/master/city.csv**.

Главными атрибутами, определяющими положение города на карте в этом файле, являются:

- `lat` (широта);
- `lon` (долгота).

Для отображения городов на карте нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для построения карты (листинг 8.44).

Листинг 8.44. Ссылка активации маршрута для построения карты

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `map_1` (листинг 8.45).

Листинг 8.45. URL-маршрут map_1

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `map_1` (листинг 8.46).

Листинг 8.46. Представление (view) map_1

Примечание.
В демо-версии книги удалены листинги программ.

Обратите внимание, что для работы этого модуля нужно обязательно подключить библиотеку pandas (она необходима для загрузки файла формата CSV из Интернета по приведенной в представлении ссылке):

```
import pandas as pd
```

В этом представлении данные из Интернета с использованием библиотеки pandas загружаются в объект `cities` (города), после чего с помощью метода `loc` выполняется фильтрация загруженных данных.

Пояснение

Функция `loc` — это мощный инструмент в библиотеке pandas для выбора данных из `dataframe` на основе набора меток. Обычно она используется для выбора подмножеств данных на основе меток строк и столбцов. Функция `loc` принимает два аргумента: метку строки и метку столбца — и возвращает подмножество `dataframe`, которое соответствует указанным меткам.

Обратите внимание, что загрузка данных в объект `cities` (города) из Интернета в рассматриваемом модуле закомментирована, и эти данные подгружаются из локального файла. Дело в том, что для корректной работы модуля в оригинальный файл, который выложен в Интернет, внесены некоторые изменения, и по указанному локальному пути загружается уже измененный файл. Кроме того, для корректного задания путей к локальному файлу в операционных системах Windows и Linux используется пакет `Path`, поэтому строка кода, в которой эта опция не задействована, также закомментирована.

Далее с использованием метода `go.Figure` создается объект `fig` (в нашем случае — карта), который заполняется данными из объекта `cities`:

- `lat=cities['geo_lat']` — широта расположения города;
- `lon=cities['geo_lon']` — долгота расположения города;
- `text=cities['city']` — название города.

На следующем шаге созданы элементы с описанием параметров макета (layout):

- `mapbox_style="open-street-map"` — подключена карта OpenStreetMap;
- `mapbox=dict(center=map_center, zoom=2)` — указан центр карты и масштаб увеличения;
- `title='Города России'` — задан заголовок макета;
- `width=1500, height=1000` — заданы ширина и высота макета.

После этого с использованием метода `fig.to_html` выполняется преобразование диаграммы в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **15. Географическая карта с городами России 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.15.

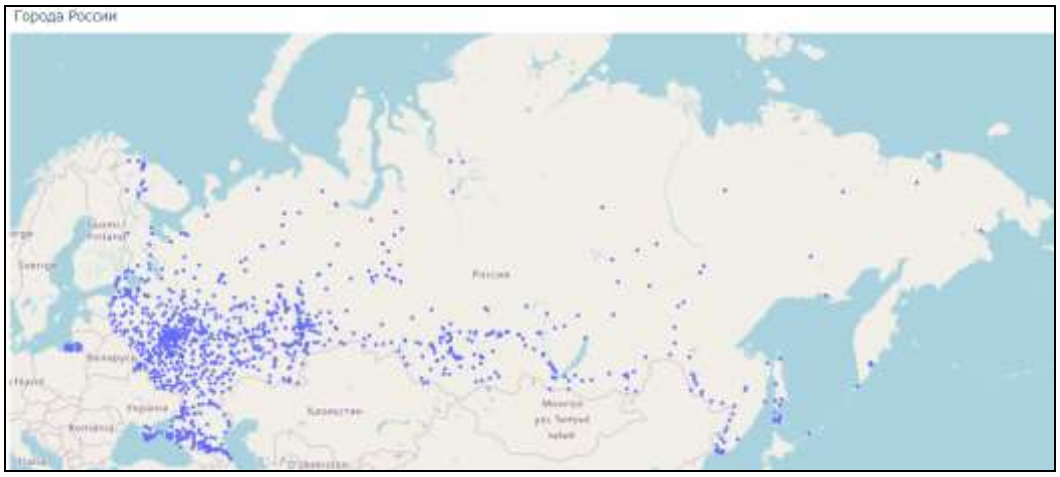


Рис. 8.15. Визуализация географической карты с городами России

Как здесь можно видеть, на карте точками показано расположение городов России в соответствии с их координатами. Если навести курсор мыши на любую из точек, то рядом с ней появятся координаты города и его название. Карту с помощью мыши можно перемещать и

масштабировать, увеличивая ее до показа региона, плана города, района города, улиц и номеров домов (рис 8.16).



Рис. 8.16. Визуализация плана города с названиями улиц и номерами домов

Используя библиотеку Plotly, на карту можно наносить различные объекты, указывая их координаты.

8.2. Визуализация данных с библиотекой plotly.py: трехмерная графика (3D)

Продолжим знакомство с библиотекой plotly.py и рассмотрим возможность формирования трехмерной графики (3D). Графики формата 3D имеют встроенную динамичность — то есть с помощью мыши можно масштабировать и поворачивать изображение вокруг его осей и отображать на нем координаты его элементов.

Для создания и отображения 3D-графиков на HTML-страницах надо последовательно выполнить следующие действия:

11. Получить массив данных по осям (x, y, z) для построения графика (данные могут быть загружены из Интернета, сформированы при обращении к специализированным функциям пользователя или заданы в виде числовых констант.
12. Сформировать пустую координатную сетку: `fig`.
13. Добавить к координатной сетке полученные данные: `fig.add_trace`.
14. Добавить к координатной сетке описание элементов графика: `fig.update_layout` (название графика, подписи осей, метки линий, размер и т. п.).
15. Трансформировать график в формат HTML: `fig.to_html`.
16. Загрузить график формата HTML в объект `context`.
17. Передать объект `context` в шаблон Django.

Далее мы рассмотрим несколько примеров визуализации данных формата 3D в шаблонах Django.

8.2.1. Создание 3D-графика линии по точкам

В качестве примера сформируем простейший 3D-график линии, построенной по точкам, для чего нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.47).

Листинг 8.47. Ссылка активации маршрута для построения 3D-графика линии

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `lin_3D_1` (листинг 8.48).

Листинг 8.48. URL-маршрут lin_3D_1

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `lin_3D_1` (листинг 8.49).

Листинг 8.49. Представление (view) lin_3D_1

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- три списка с данными для графика: `data_x1` – данные для оси *x*, `data_y1` — данные для оси *y*, `data_z1` — данные для оси *z*;
- объект `data_fig` и в него переданы списки с данными, которые будут располагаться по осям графика *x*, *y* и *z*;
- элементы с описанием параметров графика (`layout`): заголовок графика, ширина и высота графика в пикселах, отступы от края окна (слева — `l`, справа — `r`, снизу — `b`, сверху — `t`), подписи для осей *x*, *y* и *z*. При этом для осей *x* и *y* изменен порядок возрастания данных — от меньшего значения к большему (по умолчанию данные отображаются от большего значения к меньшему);
- объект `fig` (фигура, или собственно график) и в него переданы отображаемые данные (`data_fig`) и параметры элементов графика (`layout`).

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **16. Линия с параметрами по умолчанию 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.17.

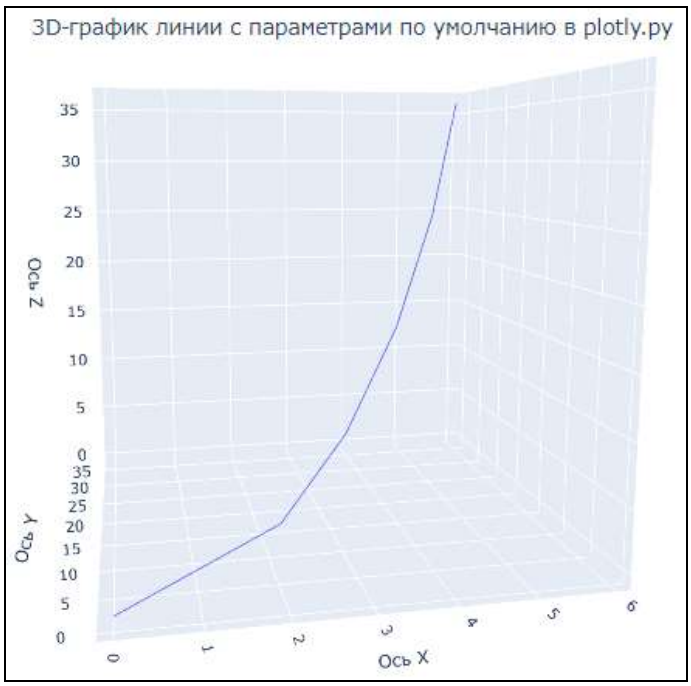


Рис. 8.17. Визуализация 3D-графика линии с параметрами по умолчанию

Как здесь можно видеть, на 3D-графике отображена одна линия, график имеет подписанные оси *x*, *y* и *z*, сверху присутствует название графика.

Поскольку на графике имеется всего одна линия, то легенда, касающаяся этой линии, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек.

8.2.2. Создание 3D-графика с указанием параметров линии

В качестве примера сформируем простейший 3D-график линии, построенной по точкам, изменив при этом цвет и толщину линии, для чего нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.50).

Листинг 8.50. Ссылка активации маршрута для построения 3D-графика линии

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `lin_3D_2` (листинг 8.51).

Листинг 8.51. URL-маршрут lin_3D_2

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `lin_3D_2` (листинг 8.52).

Листинг 8.52. Представление (view) lin_3D_2

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- три списка с данными для графика: `data_x1` – данные для оси *x*, `data_y1` — данные для оси *y*, `data_z1` — данные для оси *z*;
- объект `data_fig` и в него переданы списки с данными, которые будут располагаться по осям графика *x*, *y* и *z*;
- элементы с описанием параметров графика (`layout`): заголовок графика, ширина и высота графика в пикселах, отступы от края окна (слева — `l`, справа — `r`, снизу — `b`, сверху — `t`), подписи для осей *x*, *y* и *z*. При этом для осей *x* и *y* изменен порядок возрастания данных — от меньшего значения к большему (по умолчанию данные отображаются от большего значения к меньшему);
- объект `fig` (фигура, или собственно график) и в него переданы отображаемые данные (`data_fig`) и параметры элементов графика (`layout`).

Далее с использованием метода `fig.update_traces` заданы параметры линии: красный цвет (`color='red'`) и ширина 10 пикселов (`width=10`).

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **17. Линия с указанием ее параметров 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.18.

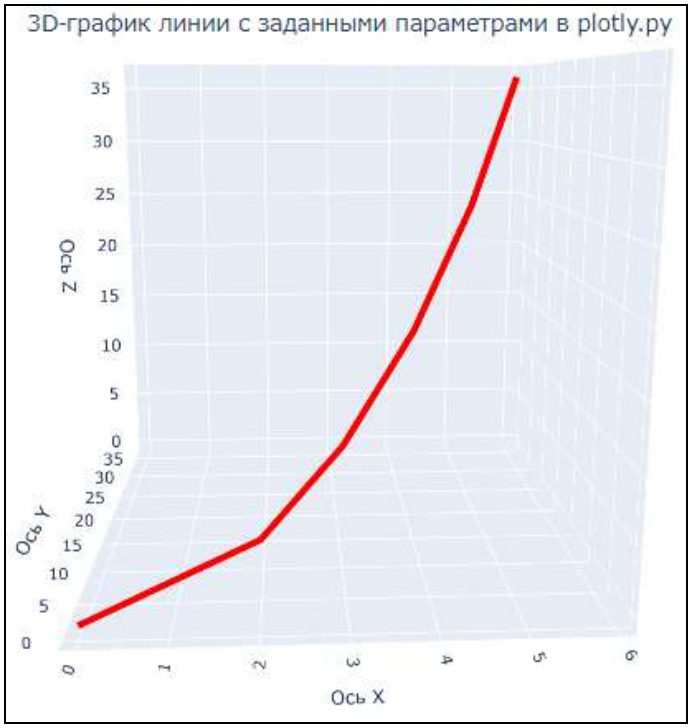


Рис. 8.18. Визуализация 3D-графика линии с указанием ее параметров

Как здесь можно видеть, на 3D-графике отображена одна широкая линия красного цвета, график имеет подписанные оси *x*, *y* и *z*, сверху присутствует название графика.

Поскольку на графике имеется всего одна линия, то легенда, касающаяся этой линии, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек.

8.2.3. Создание 3D-графика плавной витой линии

В качестве примера сформируем простейший 3D-график плавной витой линии, построенной с использованием функций, для чего нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.53).

Листинг 8.53. Ссылка активации маршрута для построения 3D-графика линии

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `lin_3D_3` (листинг 8.54).

Листинг 8.54. URL-маршрут lin_3D_3

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `lin_3D_3` (листинг 8.55).

Листинг 8.55. Представление (view) lin_3D_3

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- с использованием функций библиотеки NumPy три массива данных для графика: `x` — данные для оси *x*, `y` — данные для оси *y*, `z` — данные для оси *z*;
- объект `data_fig` и в него переданы переданы массивы с данными, которые будут располагаться по осям графика *x*, *y* и *z*;
- элементы с описанием параметров графика (`layout`): заголовок графика, ширина и высота графика в пикселах, отступы от края окна (слева — `l`, справа — `r`, снизу — `b`, сверху — `t`), подписи для осей *x*, *y* и *z*. При этом для осей *x* и *y* изменен порядок возрастания данных — от меньшего значения к большему (по умолчанию данные отображаются от большего значения к меньшему);
- объект `fig` (фигура, или собственно график) и в него переданы отображаемые данные (`data_fig`) и параметры элементов графика (`layout`).

Далее с использованием метода `fig.update_traces` заданы параметры линии: красный цвет (`color='red'`) и ширина 10 пикселов (`width=10`).

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **18. Плавная витая линия 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.19.

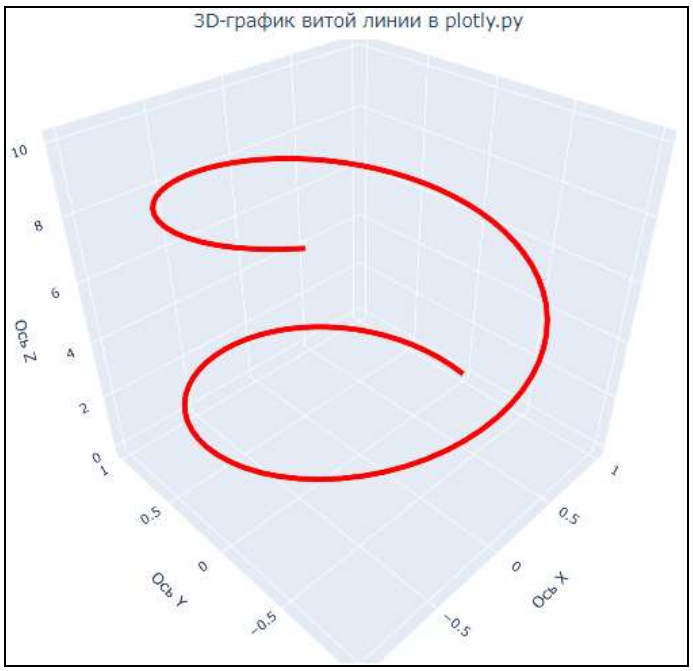


Рис. 8.19. Визуализация 3D-графика плавной витой линии

Как здесь можно видеть, на 3D-графике отображена плавная витая линия красного цвета, график имеет подписанные оси *x*, *y* и *z*, сверху присутствует название графика.

Поскольку на графике имеется всего одна линия, то легенда, касающаяся этой линии, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек.

8.2.4. Создание 3D-графика линии с ключевыми точками

В качестве примера сформируем простейший 3D-график линии с указанием ключевых точек. Ключевые точки позволят с помощью указателя мыши отобразить координаты этих точек на трехмерной координатной сетке. Для чего нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.56).

Листинг 8.56. Ссылка активации маршрута для построения 3D-графика линии

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `lin_3D_4` (листинг 8.57).

Листинг 8.57. URL-маршрут lin_3D_4

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `lin_3D_4` (листинг 8.58).

Листинг 8.58. Представление (view) lin_3D_4

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- три списка с данными для графика: `data_x1` — данные для оси *x*, `data_y1` — данные для оси *y*, `data_z1` — данные для оси *z*;
- объект `data_fig` и в него переданы списки с данными, которые будут располагаться по осям графика *x*, *y* и *z*;
- элементы с описанием параметров графика (`layout`): заголовок графика, ширина и высота графика в пикселах, отступы от края окна (слева — `l`, справа — `r`, снизу — `b`, сверху — `t`), подписи для осей *x*, *y* и *z*. При этом для осей *x* и *y* изменен порядок возрастания данных — от меньшего значения к большему (по умолчанию данные отображаются от большего значения к меньшему);
- объект `fig` (фигура, или собственно график) и в него переданы отображаемые данные (`data_fig`) и параметры элементов графика (`layout`).

Далее с использованием метода `fig.update_traces` заданы параметры ключевых точек (`marker`), и самой линии (`line`). Маркеры точек будут иметь синий цвет (`color='blue'`) и размер 10 пикселов (`size=10`), а линия — красный цвет (`color='red'`) и ширину 10 пикселов (`width=10`).

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **19. Линия с ключевыми точками 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.20.

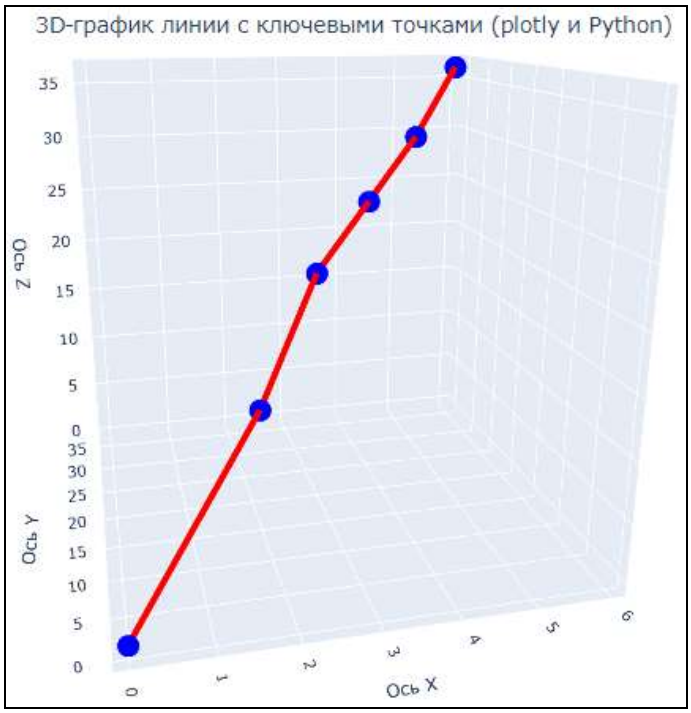


Рис. 8.20. Визуализация 3D-графика линии с ключевыми точками

Как здесь можно видеть, на 3D-графике отображена линия красного цвета с выделенными ключевыми токами синего цвета, график имеет подписанные оси *x*, *y* и *z*, сверху присутствует название графика.

Поскольку на графике имеется всего одна линия, то легенда, касающаяся этой линии, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек. При наведении указателя мыши на ключевую точку, на координатной сетке будут показаны координаты этой точки (рис. 8.21).

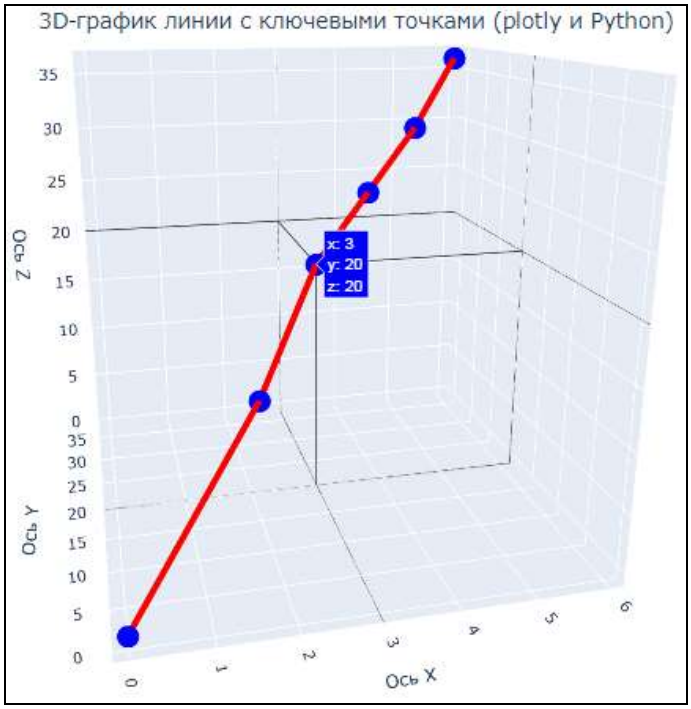


Рис. 8.21. Визуализация 3D-графика линии с отображением координат ключевой точки

8.2.5. Создание 3D-графика плоской поверхности

В качестве примера сформируем 3D-график поверхности, построенной по координатам точек, для чего нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.59).

Листинг 8.59. Ссылка активации маршрута для построения 3D-графика плоской поверхности

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `pl_3D_1` (листинг 8.60).

Листинг 8.60. URL-маршрут pl_3D_1

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `pl_3D_ 1` (листинг 8.61).

Листинг 8.61. Представление (view) pl_3D_1

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении созданы:

- три списка с данными для графика: `x` — данные для оси *x*, `y` — данные для оси *y*, `z` — данные для оси *z*;
- объект `fig` (пустая сетка для графика);
- объект `scatter` и в него загружены списки с данными, которые будут располагаться по осям графика *x*, *y* и *z*.

Далее с использованием метода `fig.update_layout` созданы элементы с описанием параметров графика (`layout`): заголовок графика, ширина и высота графика в пикселах, отступы от края окна (слева — `l`, справа — `r`, снизу — `b`, сверху — `t`), подписи для осей *x*, *y* и *z*. При этом для осей *x* и *y* изменен порядок возрастания данных — от меньшего значения к большему (по умолчанию данные отображаются от большего значения к меньшему).

Затем с помощью метода `fig.add_trace` в координатную сетку (`fig`) загружены данные из объекта `scatter`.

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **20. Плоская поверхность 3D**».

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.22.

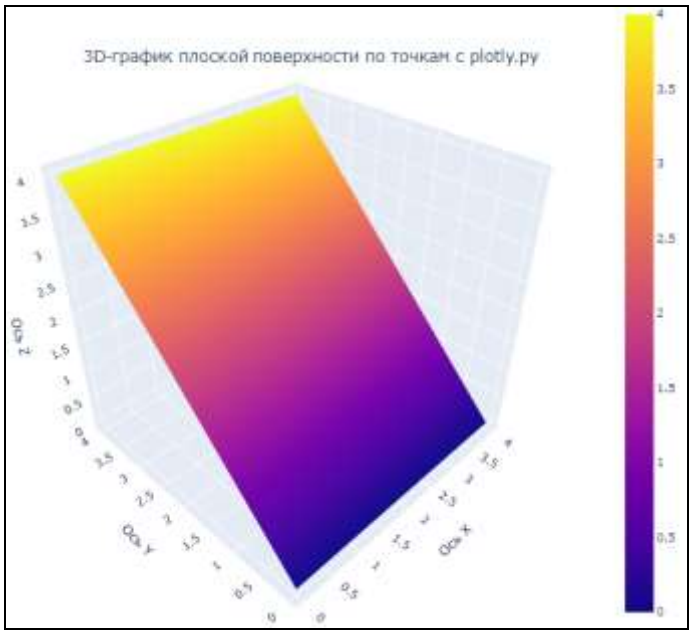


Рис. 8.22. Визуализация 3D-графика плоской поверхности

Как здесь можно видеть, на 3D-графике отображена плоская поверхность и шкала, на которой указана палитра цветов повышения уровня цвета поверхности по оси *z* относительно нулевой точки. Синий цвет показывает нижний уровень, красный и желтые цвета отображают более высокие уровни.

Поскольку на графике имеется всего одна плоскость, то легенда, касающаяся этой плоскости, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек плоскости на координатной сетке.

8.2.6. Создание 3D-графика изогнутой поверхности с использованием функций

В качестве примера сформируем 3D-график изогнутой поверхности. Множество точек, необходимых для создания нужной поверхности, мы получим с помощью функций.

Для формирования такого графика нам понадобится реализовать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.62).

Листинг 8.62. Ссылка активации маршрута для построения 3D-графика изогнутой поверхности

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `pov_3D` (листинг 8.63).

Листинг 8.63. URL-маршрут pov_3D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `pov_3D`. Поскольку данные для построения нашего графика будут получены из функции пользователя, приведем сначала программный код этой функции (листинг 8.64).

Листинг 8.64. Функция формирования поверхности полезности Кобба-Дугласа (data_kd)

Примечание.

В демо-версии книги удалены листинги программ.

Функция Кобба-Дугласа используется в экономике для формирования поверхности полезности при выборе двух товаров. Мы не станем вдаваться здесь в детали экономического анализа, а приведем лишь формулу, по которой вычисляются параметры поверхности полезности:

```
z = x ** a * y ** b
```

Исходя из этой формулы, в приведенном в листинге 8.64 программном коде функция `data_kd` принимает следующие параметры: `X_max` — максимальное значение данных по оси *x*, `Y_max` — максимальное значение данных по оси *y*, `a` — показатель степени при *x*, `b` — показатель степени при *y*.

В теле функции в цикле формируются два массива данных — для осей *x* и *y*. Затем с использованием функции `np.meshgrid` библиотеки NumPy создается координатная сетка для плоскости *x*–*y* и делается расчет координаты *z* для этой сетки по приведенной ранее формуле.

Код представления `pov_3D`, которое и создаст 3D-график, приведен в листинге 8.65.

Листинг 8.65. Представление (view) pov_3D

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении заданы и переданы в функцию `data_kd` исходные данные для построения графика: `X_max`, `Y_max`, `a`, `b`. Полученные из функции значения переданы в три

списка для формирования осей графика: `x` — данные для оси *x*, `y` — данные для оси *y*, `z` — данные для оси *z*, и полученные списки переданы в объект `data_fig` (это, по сути, и есть данные для загрузки в координатную сетку).

На следующем шаге с использованием метода `go.layout` создан объект `layout`, в котором заданы параметры графика: заголовок графика, ширина и высота графика в пикселах, отступы от края окна (слева — `l`, справа — `r`, снизу — `b`, сверху — `t`), подписи для осей *x*, *y* и *z*. При этом для осей *x* и *y* изменен порядок возрастания данных — от меньшего значения к большему (по умолчанию данные отображаются от большего значения к меньшему).

Затем с помощью метода `go.fig` создан объект `fig`, и в него передан массив данных для формирования графика (`data=data_fig`) и параметры координатной сетки (`layout=layout`).

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **21. Изогнутая поверхность 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.23.

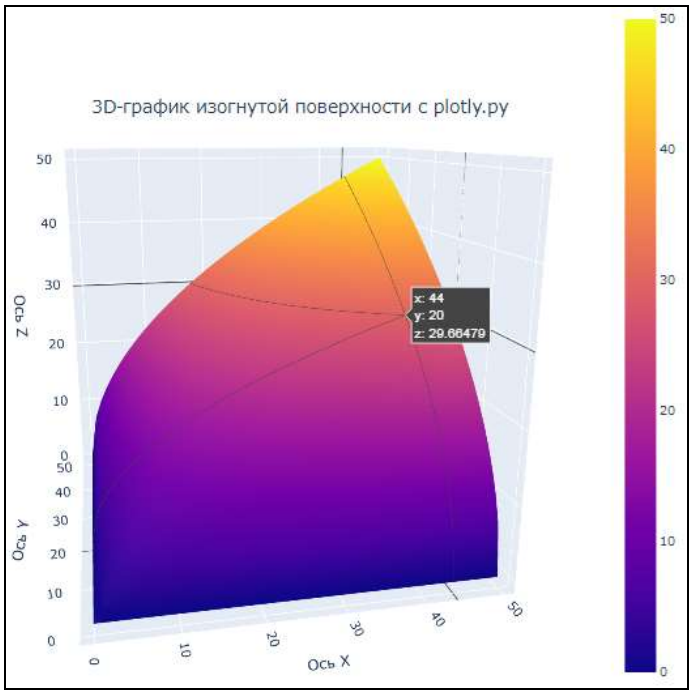


Рис. 8.23. Визуализация 3D-графика изогнутой поверхности

Как здесь можно видеть, на 3D-графике отображена плавная изогнутая поверхность и шкала, на которой указана палитра цветов повышения уровня цвета поверхности по оси *z* относительно нулевой точки. Синий цвет показывает нижний уровень, красный и желтые цвета отображают более высокие уровни.

Поскольку на графике имеется всего одна поверхность, то легенда, касающаяся этой поверхности, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек плоскости на координатной сетке.

8.2.7. Создание 3D-графика изогнутой поверхности с контурами по вертикальной оси

В качестве примера сформируем 3D-график изогнутой поверхности, на который, кроме самой поверхности, нанесены контурные линии, показывающие форму поверхности на различных уровнях. Такие контуры, например, используются на плоских географических картах для показа высоты местности над уровнем моря. По сути, эти линии представляют собой проекцию пересечения изогнутой поверхности горизонтальной плоскостью. Множество точек, необходимых для создания нужной поверхности, мы получим с помощью функций.

Для формирования такого графика нам понадобится реализовать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.66).

Листинг 8.66. Ссылка активации маршрута для построения 3D-графика изогнутой поверхности с контурными линиями по оси z

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `pov_contour_3D` (листинг 8.67).

Листинг 8.67. URL-маршрут pov_contour_3D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `pov_contour_3D` (листинг 8.68).

Листинг 8.68. Представление (view) pov_contour_3D

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении заданы и переданы в функцию `data_kd` (см. листинг 8.64) исходные данные для построения графика: `X_max`, `Y_max`, `a`, `b`. Полученные из функции значения переданы в три списка для формирования осей графика: `x` — данные для оси *x*, `y` — данные для оси *y*, `z` — данные для оси *z*.

Далее создан объект `contur`, в котором заданы параметры контурных линий для оси *z*, и этому объекту передан вложенный словарь, содержащий следующие элементы:

- `"z"` — ключ внешнего словаря (вертикальная ось трехмерного графика);
- `"show": True` — ключ вложенного словаря (предписывает отображать контурные линии);
- `"start": 1` — ключ вложенного словаря (начальное положение контурных линий по оси z);
- `"end": 50` — ключ вложенного словаря (конечное положение контурных линий по оси *z*);
- `"size": 5` — ключ вложенного словаря (размер шага между контурными линиями по оси z).

Затем объект `contur` и полученные из функции `data_kd` списки переданы в объект `data_fig` (это, по сути, и есть данные для загрузки в координатную сетку).

На следующем шаге с использованием метода `go.layout` создан объект `layout`, в котором заданы параметры графика: заголовок графика, ширина и высота графика в пикселах, отступы от края окна (слева — `l`, справа — `r`, снизу — `b`, сверху — `t`), подписи для осей *x*, *y* и

z. При этом для осей *x* и *y* изменен порядок возрастания данных — от меньшего значения к большему (по умолчанию данные отображаются от большего значения к меньшему).

Далее с помощью метода `go.fig` создан объект `fig`, и в него передан массив данных для формирования графика (`data=data_fig`) и параметры координатной сетки (`layout=layout`).

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **22. Поверхность с контурами по оси z 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.24.

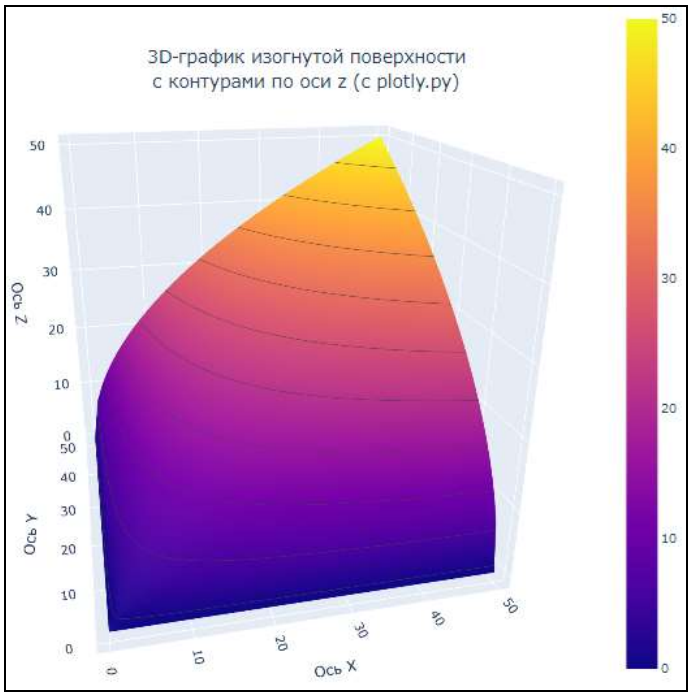


Рис. 8.24. Визуализация 3D-графика изогнутой поверхности с контурными линиями по оси z

Как здесь можно видеть, на 3D-графике отображена плавная изогнутая поверхность и шкала, на которой указана палитра цветов повышения уровня цвета поверхности по оси *z* относительно нулевой точки. Синий цвет показывает нижний уровень, красный и желтые цвета отображают более высокие уровни. Кроме того, на поверхность с заданным шагом нанесены контурные линии, показывающие форму сечения изогнутой поверхности горизонтальной плоскостью.

Поскольку на графике имеется всего одна поверхность, то легенда, касающаяся этой поверхности, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек плоскости на координатной сетке.

8.2.8. Создание 3D-графика поверхности с контурами по всем осям

В качестве примера сформируем 3D-график поверхности, на который, кроме самой поверхности, нанесены контурные линии по трем осям: *x*, *y* и *z*. Эти линии показывают форму поверхности не только на различных уровнях, но и в вертикальных разрезах. Такие контуры, например, используются для более выразительного показа решетки, образующей поверх-

ность. Множество точек, необходимых для создания нужной поверхности, мы получим из списков, заданных с помощью числовых констант.

Для формирования такого графика нам понадобится реализовать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.69).

Листинг 8.69. Ссылка активации маршрута для построения 3D-графика поверхности с контурными линиями по осям x, y и z

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `pov_contour2_3D` (листинг 8.70).

Листинг 8.70. URL-маршрут pov_contour2_3D

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `pov_contour2_3D` (листинг 8.71).

Листинг 8.71. Представление (view) pov_contour2_3D

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении создан объект `contur`, в котором заданы параметры контурных линий для трех осей: *x*, *y* и *z*, и этому объекту передан вложенный словарь, содержащий следующие ключи для каждой оси:

- `"show"` — ключ вложенного словаря (предписывает отображать контурные линии);
- `"start"` — ключ вложенного словаря (начальное положение контурных линий оси);
- `"end"` — ключ вложенного словаря (конечное положение контурных линий оси);
- `"size"` — ключ вложенного словаря (размер шага между контурными линиями оси).

Далее с использованием метода `go.Figure` создан объект `fig`, и в него переданы параметры контурных линий (`contours=contur`) и списки данных для формирования осей графика *x*, *y* и *z*.

На следующем шаге с помощью метода `fig.update_layout` заданы параметры графика: заголовок графика и его ширина и высота в пикселах.

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **23. Поверхность с контурами по осям x, y, z 3D**».

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.25.

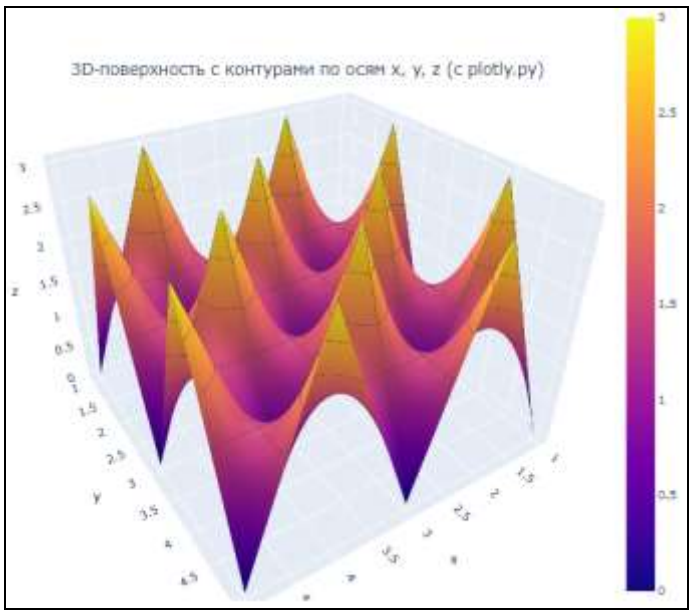


Рис. 8.25. Визуализация 3D-графика поверхности с контурными линиями по осям x, y, z

Как здесь можно видеть, на 3D-графике отображена изогнутая поверхность и шкала, на которой указана палитра цветов повышения уровня цвета поверхности по осям *x*, *y* и *z* относительно нулевой точки. Синий цвет показывает нижний уровень, красный и желтые цвета отображают более высокие уровни. Кроме того, на поверхность с заданным шагом нанесены контурные линии, показывающие форму сечения изогнутой поверхности горизонтальной плоскостью и вертикальными плоскостями.

Поскольку на графике имеется всего одна поверхность, то легенда, касающаяся этой поверхности, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек плоскости на координатной сетке.

8.2.9. Создание 3D-графика поверхности с указанием параметров сцены и положения камеры

Когда 3D-график выводится на HTML-страницу, то размер контейнера (пропорции 3D-куба), в котором расположен график, и положение камеры определяются параметрами по умолчанию, которые не всегда соответствуют требованиям пользователей.

Самостоятельно задать соотношение сторон куба, в котором помещен 3D-график, можно с помощью параметра `aspectratio`, который представляет собой словарь, имеющий три ключа: *x*, *y* и *z*. По умолчанию значения этих ключей равны единице (равносторонний куб). Если требуется, чтобы куб имел разные размеры граней, то эти параметры можно менять, например, следующим образом:

- если нужно, чтобы куб имел в два раза больший размер по оси *x*, то параметры следует задать так: `x=2, y=1, z=1`;
- если нужно, чтобы куб имел в два раза больший размер по оси *y*, то параметры следует задать так: `x=1, y=2, z=1`;
- если нужно, чтобы куб имел в два раза больший размер по оси *z*, то параметры то параметры следует задать так: `x=1, y=1, z=2`.

Самостоятельно задать положение камеры относительно центра сцены можно с помощьью параметра `camera_eye`, который представляет собой словарь, также имеющий три ключа: *x*, *y* и *z*. По умолчанию значения этих ключей равны 1,25 (число может быть положительным или отрицательным). Меняя эти параметры, камеру можно переместить в любую точку сцены (то есть куба): установить ее правее, левее, сделать вид сверху или снизу.

В качестве примера сформируем 3D-график поверхности и зададим ему параметры `camera_eye` и `aspectratio`. Множество точек, необходимых для создания этой поверхности, мы получим из списков, заданных с помощью числовых констант.

Для формирования такого графика нам понадобится реализовать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.72).

Листинг 8.72. Ссылка активации маршрута для построения 3D-графика поверхности с указанием положения камеры и размера сцены (куба)

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `pov_contour3_3D` (листинг 8.73).

Листинг 8.73. URL-маршрут pov_contour3_3D

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `pov_contour3_3D` (листинг 8.74).

Листинг 8.74. Представление (view) pov_contour3_3D

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении создан объект `contur`, в котором заданы параметры контурных линий для трех осей: *x*, *y* и *z*, и этому объекту передан вложенный словарь, содержащий следующие ключи для каждой оси:

- `"show"` — ключ вложенного словаря (предписывает отображать контурные линии);
- `"start"` — ключ вложенного словаря (начальное положение контурных линий оси);
- `"end"` — ключ вложенного словаря (конечное положение контурных линий оси);
- `"size"` — ключ вложенного словаря (размер шага между контурными линиями оси).

Далее с использованием метода `go.Figure` создан объект `fig`, и в него переданы параметры контурных линий (`contours=contur`) и списки данных для формирования осей графика *x*, *y* и *z*.

На следующем шаге с помощью метода `fig.update_layout` заданы параметры графика: заголовок графика и его ширина и высота в пикселах.

Сцена (`scene`) представляет собой вложенный словарь. В его внутренних словарях заданы следующие параметры:

- `"xaxis": {"nticks": 10}` — количество меток на оси *x*;
- `"yaxis": {"nticks": 10}` — количество меток на оси *y*;
- `"zaxis": {"nticks": 10}` — количество меток на оси *z*;
- `'camera_eye': {"x": -2, "y": -1, "z": -1}` — положение камеры (вид снизу);
- `"aspectratio": {"x": 1.5, "y": 1, "z": 1}` — соотношение сторон куба.

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **24. Поверхность с положением камеры 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.26.

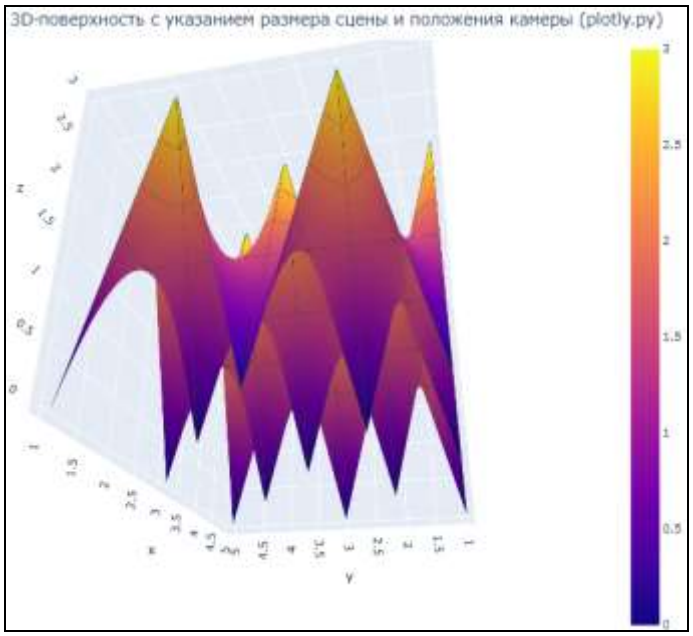


Рис. 8.26. Визуализация 3D-графика поверхности с положением камеры «вид снизу»

Как здесь можно видеть, на 3D-графике отображена изогнутая поверхность и шкала, на которой указана палитра цветов повышения уровня цвета поверхности по осям *x*, *y* и *z* относительно нулевой точки. Синий цвет показывает нижний уровень, красный и желтые цвета отображают более высокие уровни. Кроме того, на поверхность с заданным шагом нанесены контурные линии, показывающие форму сечения изогнутой поверхности горизонтальной плоскостью и вертикальными плоскостями. При заданных параметрах камеры при первоначальной загрузке графика пользователь видит график поверхности снизу.

Поскольку на графике имеется всего одна поверхность, то легенда, касающаяся этой поверхности, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек плоскости на координатной сетке. При этом, если мышью повернуть график и сместить положение камеры на вид сверху, то станет видно, что сцена (куб) вытянута в направлении оси *x* (рис 8.27), — по этой оси она имеет размер в полтора раза больше, чем по оси *y*, так как в параметрах сцены были указаны следующие значения свойства `aspectratio`: `{"x": 1.5, "y": 1, "z": 1}`.

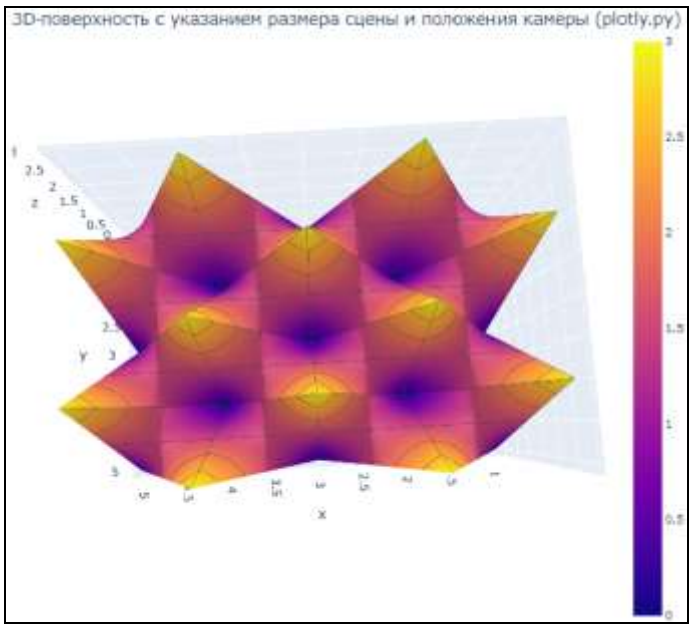


Рис. 8.27. Визуализация 3D-графика: изменены параметры сцены (соотношение сторон куба)

8.2.10. Создание 3D-графика поверхности с контурной сеткой

В качестве примера сформируем 3D-график поверхности, на который, кроме самой поверхности, нанесена контурная сетка по трем осям: *x*, *y* и *z*. Такая контурная сетка, состоящая из плавных кривых линий, позволяет более детально показать форму поверхности.

В этом примере множество точек, необходимых для создания такой поверхности, мы получим из списков, заданных с помощью числовых констант.

Для формирования такого графика нам понадобится реализовать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.75).

Листинг 8.75. Ссылка активации маршрута для построения 3D-графика поверхности с контурной сеткой по осям x, y и z

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `pov_contour4_3D` (листинг 8.76).

Листинг 8.76. URL-маршрут pov_contour4_3D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `pov_contour4_3D` (листинг 8.77).

Листинг 8.77. Представление (view) pov_contour4_3D

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении создан объект `contur`, в котором заданы параметры контурных линий для трех осей: *x*, *y* и *z*, и этому объекту передан вложенный словарь, содержащий следующие ключи для каждой оси:

- `"show"` — ключ вложенного словаря (предписывает отображать контурные линии);
- `"start"` — ключ вложенного словаря (начальное положение контурных линий оси);
- `"end"` — ключ вложенного словаря (конечное положение контурных линий оси);
- `"size"` — ключ вложенного словаря (размер шага между контурными линиями оси).

Расстояние между линиями сетки зависит от значения параметра `size` — чем меньше это значение, тем больше линий будет в сетке. В нашем примере параметр `size` имеет значение 0,05 для всех осей.

Далее с использованием метода `go.Figure` создан объект `fig`, и в него переданы параметры контурных линий (`contours=contur`) и списки данных для формирования осей графика *x*, *y* и *z*.

На следующем шаге с помощью метода `fig.update_layout` заданы параметры графика: заголовок графика и его ширина и высота в пикселах.

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **25. Поверхность с контурной сеткой 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.28.

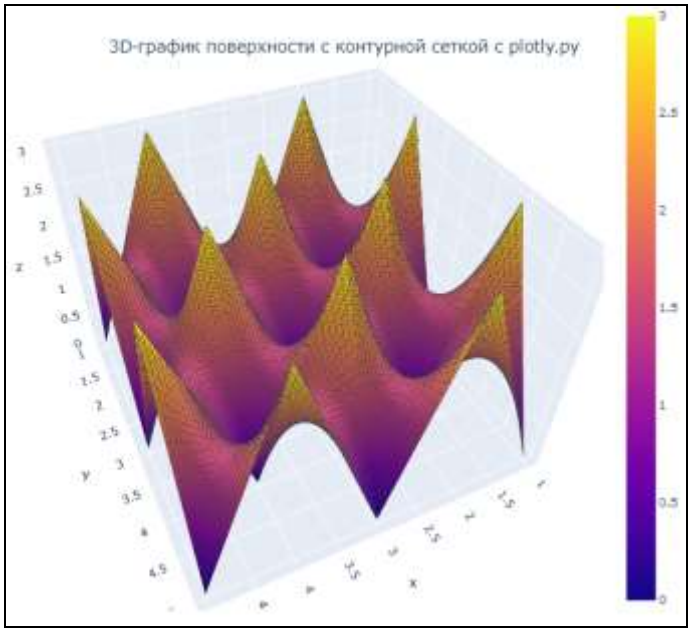


Рис. 8.28. Визуализация 3D-графика поверхности с контурной сеткой по осям x, y, z

Как здесь можно видеть, на 3D-графике отображена изогнутая поверхность и шкала, на которой указана палитра цветов повышения уровня цвета поверхности по осям *x*, *y* и *z* относительно нулевой точки. Синий цвет показывает нижний уровень, красный и желтые цвета отображают более высокие уровни. Кроме того, на поверхность с заданным шагом нанесена сетка из контурных линий, которая подчеркивает форму поверхности.

Поскольку на графике имеется всего одна поверхность, то легенда, касающаяся этой поверхности, не отображается (даже если указать ее имя в параметрах графика).

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек поверхности на координатной сетке..

8.2.11. Создание 3D-диаграммы рассеяния точек

На практике достаточно часто возникает потребность отобразить в пространстве рассеяние множества точек, принадлежащих разным выборкам.

В качестве примера сформируем 3D-диаграмму для двух наборов точек, для чего нам понадобится реализовать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для построения диаграммы (листинг 8.78).

Листинг 8.78. Ссылка активации маршрута для построения 3D-диаграммы рассеяния точек

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `pov_contour5_3D` (листинг 8.79).

Листинг 8.79. Маршрут URL-маршрут pov_contour5_3D

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `pov_contour5_3D` (листинг 8.80).

Листинг 8.80. Представление (view) pov_contour5_3D

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении с использованием функций библиотеки NumPy созданы два набора точек для трехмерного пространства: `x1`, `y1`, `z1` и `x2`, `y2`, `z2`. Точки закручены в две спирали в ограниченном пространстве.

Затем на основе сформированных наборов точек для диаграммы созданы два набора данных: `data_fig1` и `data_fig2`. В каждый набор переданы координаты точек `x1`, `y1`, `z1` и `x2`, `y2`, `z2` и указано, что отображать нужно не линии, а точки (`mode='markers'`). Для каждого набора точек задана легенда: `name='График 1'` и `name='График 2'`.

На следующем шаге с использованием метода `go.layout` создан объект `layout`, в котором заданы следующие параметры диаграммы: заголовок, ширина и высота диаграммы в пикселах, параметры сцены (подписи для осей).

Далее с помощью метода `go.fig` создан объект `fig`, и в него переданы данные для формирования осей диаграммы (`data_fig1`, `data_fig2`) и параметры сцены (`layout=layout`).

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **26. Диаграмма рассеяния 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.29.

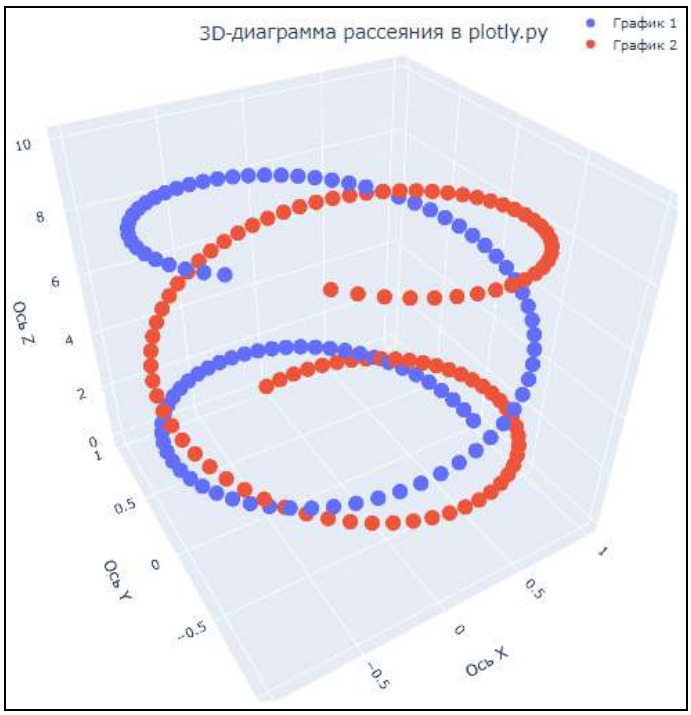


Рис. 8.29. Визуализация графика 3D - диаграмма рассеяния точек

Как здесь можно видеть, на 3D-диаграмме показаны положения точек из двух разных наборов данных в трехмерном пространстве. Каждый набор точек имеет свой цвет (красные и синие).

Поскольку на графике имеются два набора точек, то легенда, касающаяся каждого набора, отображается рядом с диаграммой.

С помощью мыши график можно вращать вокруг любой оси, масштабировать и отображать координаты точек на координатной сетке.

8.3. Визуализация интерактивных графиков с библиотекой plotly.py

Продолжим знакомство с библиотекой plotly.py и рассмотрим возможность формирования интерактивных графиков. Интерактивные графики отличаются от статичных тем, что имеют слайдер, с помощью которого пользователи могут анимировать («оживлять») элементы графика.

Для создания и отображения интерактивных графиков на HTML-страницах надо последовательно выполнить следующие действия:

18. Получить массив данных по осям (x, y) для построения статичного элемента графика (данные могут быть загружены из Интернета, сформированы при обращении к специализированным функциям пользователя или заданы в виде числовых констант).
19. Сформировать пустую координатную сетку (`fig`) и загрузить в нее данные статичного элемента графика.
20. Создать набор кадров интерактивной части графика и в каждый кадр загрузить данные динамичного элемента графика;
21. Создать список шагов и для каждого шага определить набор параметров, отображаемых в каждом кадре динамичного графика.
22. Создать слайдер (в виде словаря) и загрузить в него список созданных шагов (кадров), указав какой кадр нужно показать при начальном отображении графика.
23. Добавить слайдер к макету графика.
24. Трансформировать график в формат HTML: `fig.to_html`.
25. Загрузить график формата HTML в объект `context`.
26. Передать объект `context` в шаблон Django.

Далее мы рассмотрим несколько примеров визуализации интерактивных графиков в шаблонах Django.

8.3.1. Создание интерактивного 2D-графика со слайдером

В качестве примера сформируем простейший 2D-график линий, построенных по координатам точек, для чего нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.81).

Листинг 8.81. Ссылка активации маршрута для построения интерактивного 2D-графика линий

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `inter_graf_2D` (листинг 8.82).

Листинг 8.82. URL-маршрут inter_graf_2D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `inter_graf_2D` (листинг 8.83).

Листинг 8.83. Представление (view) inter_graf_2D

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении последовательно:

- созданы два списка с данными для статичного элемента графика: `x` — данные для оси *x*, `y` — данные для оси *y*;
- создан объект `fig` (фигура, или собственно график) и в него переданы отображаемые данные (`x` и `y`), а также некоторые параметры элементов графика: `mode='markers'` (показать график в виде точек) и `name='График статичный'` (легенда для этого графика);
- в цикле с помощью метода `fig.add_trace` создан набор трасс (набор данных для динамичного элемента графика).

  Фактически, это набор кадров статичных графиков при определенных значениях `x` и `y` (в нашем примере будет создано 40 таких кадров). В теле цикла, кроме формирования данных (`x` и `y`), создана легенда для динамичной линии графика (`name`) и задано значение видимости (`visible`) только одной трассы из сорока созданных кадров;
- создан список шагов слайдера (`steps`) — то есть список параметров каждого кадра, отображаемого при каждом шаге изменения значения слайдера. В этом цикле заданы следующие параметры:
  - скрыть (не показывать) все кадры: `visible": [False] * len(fig.data);`
  - отображать наименование (`title`) *i*-го кадра графика;
  - всегда сохранять видимость линии статичного графика: `(step["args"][0]["visible"][0] = True);`
  - отображать *i*-й кадр с линией динамичного графика: `(step["args"][0]["visible"][i] = True);`

- создан объект `sliders` (слайдер) в виде словаря, в котором заданы значения следующим ключам:
  - `active=20` — показать активным 20-й кадр при загрузке графика;
  - `currentvalue={"prefix": "Текущее значение: "}` — текущий кадр слайдера;
  - `steps=steps` — шаги слайдера (кадры);
- с помощью метода `fig.update_layout` к макету графика добавлены слайдер и следующие параметры :
  - `xaxis=dict(title="Ось X")` — подпись к оси *x*;
  - `yaxis=dict(title="Ось Y")` — подпись к оси *y*;
  - `width=800` — ширина сетки (макета) графика;
  - `height=500` — высота сетки (макета) графика;
  - `sliders=sliders` — слайдер;
  - `autosize=False` — запретить изменение размера слайдера;
  - параметры шрифтов графика.

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон stat_graf_2d.html (листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **27. Интерактивный график (линия) 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.30.

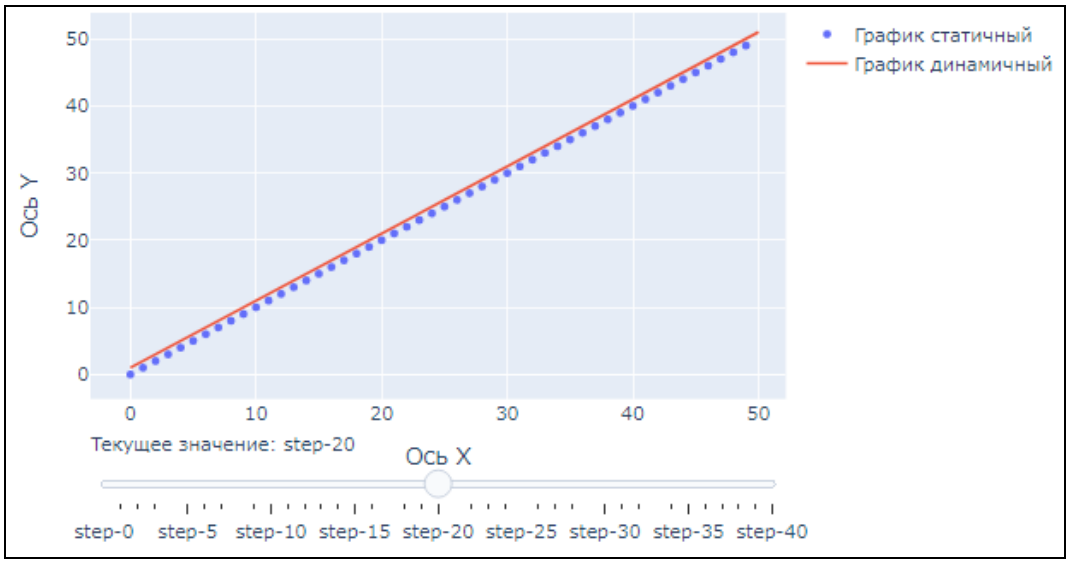


Рис. 8.30. Визуализация интерактивного 2D-графика двух линий (кадр 20)

Как здесь можно видеть, на 2D-графике отображены две линии: график линии, нарисованной точками, является статичным и не меняется при перемещении слайдера с помощью мыши, а линия, нарисованная красным цветом, меняет свое положение. Так, если бегунок слайдера переместить на 35-й кадр, то красная линия относительно координатной сетки плавно переместится вверх, и изменится содержание шкалы по оси *y*, а положение синей линии относительно координатной сетки не изменится (рис. 8.31). При этом чем больше кадров будет создано для слайдера, тем более плавно будет меняться содержимое графика.

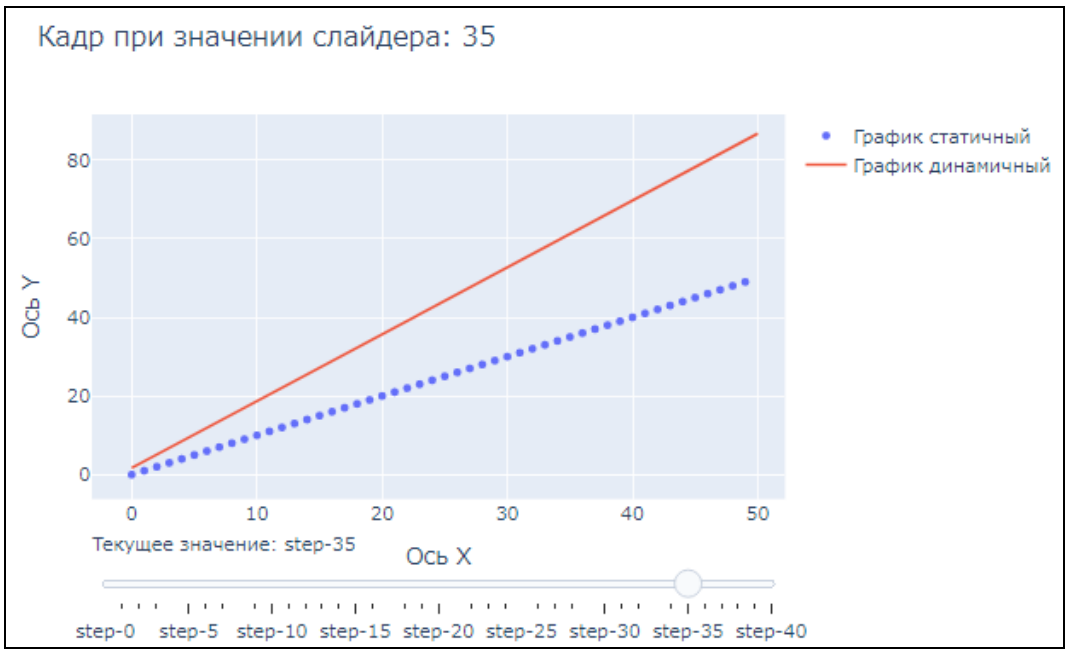


Рис. 8.31. Визуализация интерактивного 2D-графика двух линий (кадр 35)

8.3.2. Комбинирование интерактивного графика с анимированными изображениями

На практике достаточно часто возникает потребность в рамках одного шаблона показать интерактивную (управляемую) диаграмму вместе с анимированными или статичными изображениями.

В качестве примера создадим страницу, содержащую три фрагмента:

- статичное изображение формата JPG;
- анимированный рисунок формата GIF;
- интерактивную диаграмму, управляемую слайдером.

На статичном изображении покажем простейшую схему генератора переменного тока, на выходе которого формируется переменный ток с напряжением, меняющимся по синусоиде. На анимированном рисунке покажем схему работу трансформатора, который понижает (или повышает) напряжение переменного тока. На интерактивной (управляемой) диаграмме покажем, как меняется форма синусоиды в зависимости от частоты вращения ротора генератора.

Для реализации этого примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графиков (листинг 8.84).

Листинг 8.84. Ссылка активации маршрута для формирования группы графиков

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `inter_graf1_2D` (листинг 8.85).

Листинг 8.85. URL-маршрут inter_graf1_2D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `inter_graf1_2D` (листинг 8.86).

Листинг 8.86. Представление (view) inter_graf1_2D

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении:

- создан массив `x` (данные для оси *x*);
- в инструкции `x = np.linspace(0, 10, 101)` для оси *x* созданы 100 значений в интервале 0–10 с шагом 0.1 (0.0, 0.1, 0.2, ..., 10.0);
- сформирован пустой список (`data`) — объект для хранения данных интерактивного графика;
- в цикле `for` выполняется формирование элементов `trase` интерактивного графика (данные для осей *x* и *y*). В нашем случае создаются 50 шагов изменения положения слайдера (50 кадров) — то есть 50 наборов `trase`;
- в следующих инструкциях обеспечивается видимость 10-го кадра в начальном состоянии (при первом открытии шаблона Django):

```
v_step = 10
data[v_step].visible = True
```

- создается пустой список шагов слайдера (`steps[]`) и организован цикл `for`, в котором на каждом шаге слайдера делается видимым только один кадр с индексом `s`, а все остальные кадры скрываются;
- создается объект `sliders` (слайды, кадры), в который загружаются параметры всех шагов слайдера (параметры кадров каждого шага слайдера);
- задаются параметры макета интерактивного графика (`layout`):
  - `title` — заголовок интерактивного графика;
  - `xaxis` — подпись к оси *x*;
  - `yaxis` — подпись к оси *y*;
  - `font` — параметры шрифта для надписей макета;
  - `hovermode` — способ отображать значения чисел при наведении курсора мыши на график, может иметь следующие значения: `"closest"`, `False`, `"x unified"`, `"y unified"`;
  - `hoverlabel` — размера шрифта для данных графика;
  - `sliders` — добавить слайдер к макету;
  - `showlegen` — отображать легенду.

После этого с использованием метода `fig.to_html` выполнено преобразование графика в формат HTML и результаты преобразования сохранены в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передан в шаблон generat.html (листинг 8.87).

Листинг 8.87. Шаблон generat.html

Примечание.
В демо-версии книги удалены листинги программ.

В этом шаблоне созданы три адаптивных блока со следующим содержимым:

- изображение формата GPG;
- изображение формата GIF;
- интерактивный график со слайдером.

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **28. Интерактивный график с изображениями 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.32.

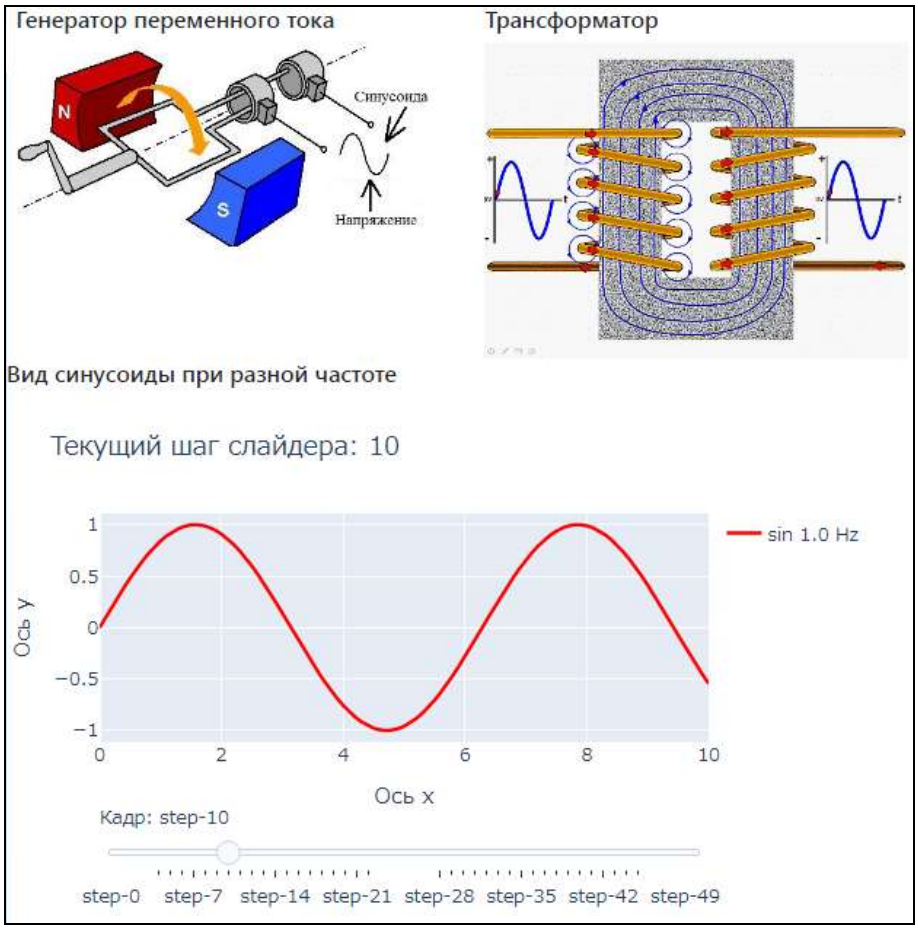


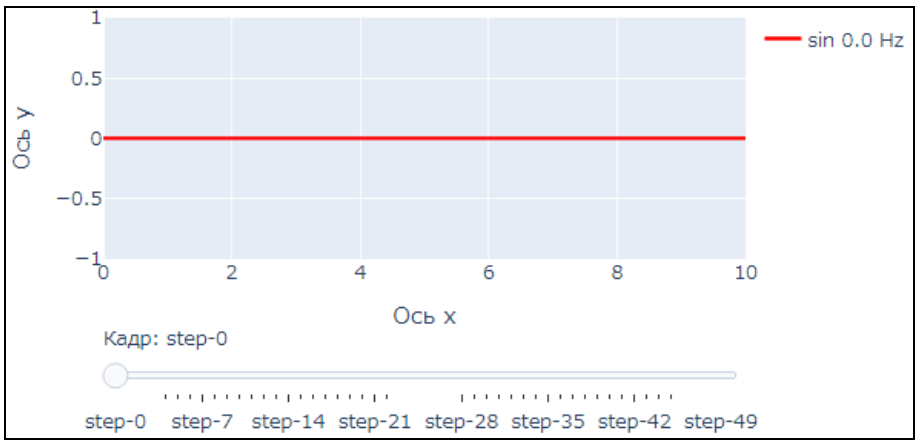
Рис. 8.32. Визуализация изображений и интерактивного графика

Как здесь можно видеть, на HTML-странице отображены три объекта:

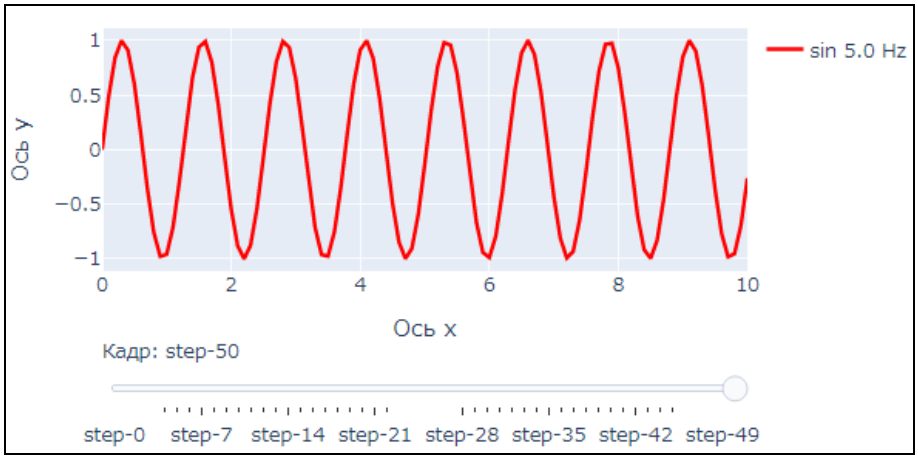
- изображение формата GPG;
- изображение формата GIF;
- интерактивный график (синусоида), управляемый с помощью слайдера.

Анимированное gif-изображение в динамике демонстрирует работу трансформатора: меняется направление движения тока и магнитных линий, а также характер напряжения.

На интерактивном графике синусоида меняет свою форму при изменении положения бегунка слайдера (изменении частоты). Так, если бегунок слайдера переместить на нулевой кадр (частота 0 Гц, ротор генератора не вращается, и выходное напряжение равно нулю), то синусоида превращается в линию (рис. 8.33, *а*). А если бегунок слайдера переместить на 50-й кадр (частота 5 Гц), то прямая линия принимает форму синусоиды с изменяющимся периодом (рис. 8.33, *б*). При этом чем больше кадров будет создано для слайдера, тем более плавно будет меняться содержимое интерактивного графика.



а



б

Рис. 8.33. Изменение формы синусоиды при перемещении слайдера

8.3.3. Вывод двух интерактивных графиков в одну координатную сетку

При визуализации данных весьма часто требуется на одной координатной сетке показать сразу несколько графиков и управлять их анимацией. Попробуем с помощью интерактивных графиков смоделировать характер изменения тока и напряжения в сети переменного тока с конденсатором.

Конденсатор

Конденсатор состоит из двух пластин, разделенных диэлектриком. При возрастании напряжения по цепи, содержащей конденсатор, начинают бежать электроны — то есть появляется ток. Электроны могут добежать только до пластин — дальше некуда. На пластинах «место» ограничено, поэтому, чем больше электронов там уже есть, тем медленнее прибегают новые, — то есть ток постепенно спадает. Из-за того, что напряжение у нас синусоидальное, ток спадает до нуля ровно в тот момент, когда напряжение достигает максимума. При снижении напряжения электронам на пластинах удерживаться сложнее — они начинают бежать обратно, то есть ток меняет направление. Быстрее всего они бегут тогда, когда напряжение равно нулю (ток становится максимальным). Далее все происходит в обратной последовательности, и этот процесс многократно повторяется. Изменения тока и напряжения носят синусоидальный характер (рис. 8.34).

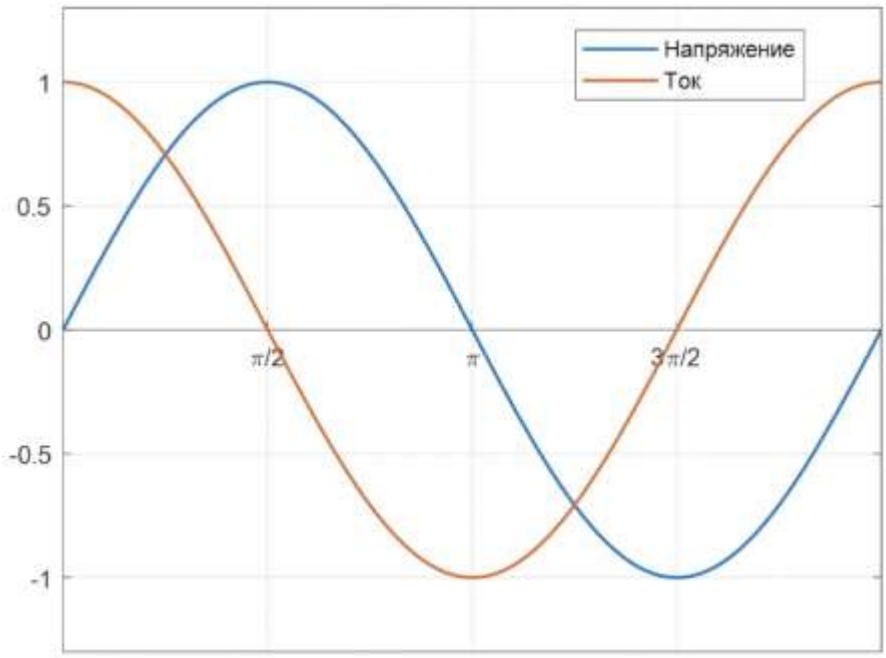


Рис. 8.34. Характер изменения тока и напряжения на конденсаторной емкости

В качестве примера создадим страницу, на которой будет показана динамика изменения тока и напряжения на конденсаторной емкости при изменении частоты переменного тока.

Для реализации этого примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графиков (листинг 8.88).

Листинг 8.88. Ссылка активации маршрута для формирования группы графиков

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `inter_graf2_2D` (листинг 8.89).

Листинг 8.89. URL-маршрут inter_graf2_2D

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `inter_graf2_2D` (листинг 8.90).

Листинг 8.90. Представление (view) inter_graf2_2D

Примечание.
В демо-версии книги удалены листинги программ.

Поскольку и ток, и напряжение изменяются по синусоиде, имея сдвиг по фазе на 90 градусов, для отображения динамики изменения напряжения можно использовать функцию sin(x), а для отображения динамики изменения тока — функцию cos(x).

Исходя из этого, в нашем представлении:

- создан массив `x` (данные для оси *x*);

- в инструкции `x = np.linspace(0, 10, 101)` для оси *x* созданы 100 значений в интервале 0–10 с шагом 0.1 (0.0, 0.1, 0.2, ..., 10.0);
- сформированы два пустых списка: `sin_data` — объект для хранения данных с динамикой изменения напряжения и `cos_data` — объект для хранения данных с динамикой изменения тока;
- в цикле `for` выполняется формирование элементов `trase` интерактивных графиков `sin_trace` и `cos_trace` (данные для осей *x*, *y* и *x*, *y2*). В нашем случае создаются 50 шагов изменения положения слайдера (50 кадров) — то есть 50 наборов `trase`;
- в следующих инструкциях обеспечивается видимость 10-го кадра в начальном состоянии (при первом открытии шаблона Django):

```
v_step = 10
sin_data[v_step].visible = True
cos_data[v_step].visible = True
```

- выполняется объединение данных для двух графиков в общий список:

```
data = sin_data + cos_data
```

- создается пустой список шагов слайдера (`steps[]`) и организован цикл `for`, в котором на каждом шаге слайдера делается видимым только один кадр с индексом `s`, а все остальные кадры скрываются;
- создается объект `sliders` (слайды, кадры), в который загружаются параметры всех шагов слайдера (параметры кадров каждого шага слайдера);
- задаются параметры макета интерактивного графика (`layout`):
  - `title` — заголовок интерактивного графика;
  - `xaxis` — подпись к оси *x*;
  - `yaxis` — подпись к оси *y*;
  - `font` — параметры шрифта для надписей макета;
  - `hovermode` — способ отображать значения чисел при наведении курсора мыши на график, может иметь следующие значения: `"closest"`, `False`, `"x unified"`, `"y unified"`;
  - `hoverlabel` — размера шрифта для данных графика;
  - `sliders` — добавить слайдер к макету;
  - `showlegen` — отображать легенду.

После этого с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **29. Два интерактивных графика (sin, cos) 2D**».

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.35.



Рис. 8.35. Визуализация двух графиков в одной координатной сетке

Как здесь можно видеть, на HTML-странице отображены два графика:

- синусоида красного цвета — характеризует колебания напряжения;
- синусоида синего цвета — характеризует колебания тока.

Чем больше кадров будет создано для слайдера, тем более плавно будет меняться содержимое интерактивного графика.

8.3.4. Вывод статичных и интерактивных графиков в одну координатную сетку

При визуализации данных иногда требуется на одной координатной сетке показать и статичные, и интерактивные графики.

В качестве примера создадим страницу, на которой будут показаны один статичный график и два интерактивных.

Для реализации этого примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графиков (листинг 8.91).

Листинг 8.91. Ссылка активации маршрута для формирования группы графиков

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `inter_graf3_2D` (листинг 8.92).

Листинг 8.92. URL-маршрут inter_graf3_2D

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `inter_graf3_2D` (листинг 8.93).

Листинг 8.93. Представление (view) inter_graf3_2D

Примечание.
В демо-версии книги удалены листинги программ.

В этом представлении:

- создан массив `x` (данные для оси *x*);
- в инструкции `x = np.linspace(0, 10, 101)` для оси *x* созданы 100 значений в интервале 0–10 с шагом 0.1 (0.0, 0.1, 0.2, ..., 10.0);
- создан пустой список для хранения данных (`data`) и сформированы данные для статичного графика (объект `trace` — нулевой элемент для списка `data`);
- объект `trase` добавлен к списку `data` инструкцией `data.append(trace)` — то есть объект `trace` стал нулевым элементом списка `data`. Обратите внимание, что все данные статичного графика будут видны всегда, поскольку параметр, обеспечивающий видимость, имеет значение `visible=True`;
- в цикле `for` выполняется формирование элементов `trase` интерактивных графиков `sin_trace` и `cos_trace` (данные для осей *x*, *y* и *x*, *y2*). В нашем случае создаются 50 шагов изменения положения слайдера (50 кадров) — то есть 50 наборов `trase`. Обратите внимание, что все данные интерактивных графиков будут скрыты, поскольку параметр, обеспечивающий их видимость, имеет значение `visible=False`;
- в следующих инструкциях обеспечивается видимость 10-го кадра интерактивных графиков `sin_data` и `cos_data` в начальном состоянии (при первом открытии шаблона Django):

```
v_step = 10
sin_data[v_step].visible = True
cos_data[v_step].visible = True
```

- к объекту `data` добавляются еще два набора данных (после этого список `data` будет содержать три элемента с индексами 0, 1 и 2):

```
data += sin_data + cos_data
```

- создается пустой список шагов слайдера (`steps[]`) и организовывается цикл `for`, в котором на каждом шаге слайдера все кадры интерактивных графиков скрываются и делается видимым только один кадр с индексом `s`;
- создается объект `sliders` (слайды, кадры), в который загружаются параметры всех шагов слайдера (параметры кадров каждого шага слайдера);
- задаются параметры макета интерактивного графика (`layout`):
  - `title` — заголовок интерактивного графика;
  - `xaxis` — подпись к оси *x*;
  - `yaxis` — подпись к оси *y*;
  - `font` — параметры шрифта для надписей макета;
  - `hovermode` — способ отображать значения чисел при наведении курсора мыши на график, может иметь следующие значения: `"closest"`, `False`, `"x unified"`, `"y unified"`;
  - `hoverlabel` — размера шрифта для данных графика;
  - `sliders` — добавить слайдер к макету;
  - `showlegen` — отображать легенду.

После этого создается объект `fig` (фигура или график) и с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML. Результаты этого преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **30. Сочетание статичных и интерактивных графиков 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.36.



Рис. 8.36. Визуализация трех графиков в одной координатной сетке

Как здесь можно видеть, на HTML-странице отображены три графика:

- интерактивная синусоида красного цвета с амплитудой 1;
- интерактивная синусоида синего цвета с амплитудой 1;
- статичная синусоида зеленого цвета с амплитудой 2.

При изменении положения бегунка слайдера интерактивные синусоиды будут изменяться, а статичная синусоида останется неизменной.

Чем больше кадров будет создано для слайдера, тем более плавно будет меняться содержимое интерактивных графиков.

8.3.5. Анимация графиков с исчезновением элементов

При визуализации интерактивных данных иногда требуется на одной координатной сетке скрыть/вернуть один из визуальных элементов.

В качестве примера создадим страницу, на которой будет показан один статичный график и два интерактивных. При этом по достижении бегунком слайдера заданной позиции один из интерактивных элементов исчезнет.

Для реализации этого примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графиков (листинг 8.94).

Листинг 8.94. Ссылка активации маршрута для формирования группы графиков

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `inter_graf4_2D` (листинг 8.95).

Листинг 8.95. URL-маршрут inter_graf4_2D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `inter_graf4_2D` (листинг 8.96).

Листинг 8.96. Представление (view) inter_graf4_2D

Примечание.

В демо-версии книги удалены листинги программ.

В этом представлении:

- создан массив `x` (данные для оси *x*);
- в инструкции `x = np.linspace(0, 10, 101)` для оси *x* созданы 100 значений в интервале 0–10 с шагом 0.1 (0.0, 0.1, 0.2, ..., 10.0);
- создан пустой список для хранения данных всего макета (`data`) и два пустых списка для хранения данных интерактивных графиков (`sin_data` и `cos_data`):

```
data = []
sin_data, cos_data = [], []
```

- сформированы данные для статичного графика (объект `trace` — нулевой элемент для списка `data`);
- объект `trase` добавлен к списку `data` инструкцией `data.append(trace)` — то есть объект `trace` стал нулевым элементом списка `data`. Обратите внимание, что все данные статичного графика будут видны всегда, поскольку параметр, обеспечивающий видимость, имеет значение `visible=True`;
- в цикле `for` выполняется формирование элементов `trase` интерактивных графиков `sin_trace` и `cos_trace` (данные для осей *x*, *y* и *x*, *y2*). В нашем случае создаются 50 шагов изменения положения слайдера (50 кадров) — то есть 50 наборов `trase`. Обратите внимание, что все данные интерактивных графиков будут скрыты, поскольку параметр, обеспечивающий их видимость, имеет значение `visible=False`. Кроме того, с использованием разветвления `if` данные объекта `cos_trase` формируются только для первых тридцати кадров, а начиная с 31-го кадра, они будут пустыми;
- в следующих инструкциях обеспечивается видимость 10-го кадра интерактивных графиков `sin_data` и `cos_data` в начальном состоянии (при первом открытии шаблона Django):

```
v_step = 10
sin_data[v_step].visible = True
cos_data[v_step].visible = True
```

- к объекту `data` добавляются еще два набора данных (после этого список `data` будет содержать три элемента с индексами 0, 1 и 2):

```
data += sin_data + cos_data
```

- создается пустой список шагов слайдера (`steps[]`) и организовывается цикл `for`, в котором на каждом шаге слайдера все кадры интерактивных графиков скрываются и делается видимым только один кадр с индексом `s`;
- создается объект `sliders` (слайды, кадры), в который загружаются параметры всех шагов слайдера (параметры кадров каждого шага слайдера);
- задаются параметры макета интерактивного графика (`layout`):
  - `title` — заголовок интерактивного графика;
  - `xaxis` — подпись к оси *x*;
  - `yaxis` — подпись к оси *y*;
  - `font` — параметры шрифта для надписей макета;
  - `hovermode` — способ отображать значения чисел при наведении курсора мыши на график, может иметь следующие значения: `"closest"`, `False`, `"x unified"`, `"y unified"`;
  - `hoverlabel` — размера шрифта для данных графика;
  - `sliders` — добавить слайдер к макету;
  - `showlegen` — отображать легенду.

После этого создается объект `fig` (фигура или график) и с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML. Результаты этого преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **31. Анимация с исчезновением элемента 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.37.



а



б

Рис. 8.37. Визуализация трех графиков в одной координатной сетке с исчезновением интерактивного элемента

Как здесь можно видеть, на HTML-странице отображены три графика:

- интерактивная синусоида красного цвета с амплитудой 1;
- интерактивная синусоида синего цвета с амплитудой 1;
- статичная синусоида зеленого цвета с амплитудой 2.

При положении бегунка слайдера в интервале 0–29 интерактивные синусоиды будут изменяться, но на графике будут видны все три синусоиды (см. рис. 8.37, *а*). При положении

бегунка слайдера в интервале 30–50 одна синусоида исчезнет (см. рис. 8.37, *б*) и появится лишь после того, как бегунок вернется в положение 0–29.

Чем больше кадров будет создано для слайдера, тем более плавно будет меняться содержимое интерактивных графиков.

8.3.6. Анимация графиков с появлением элементов

При визуализации интерактивных данных иногда требуется на одной координатной сетке показать какой-либо из визуальных элементов при определенном положении слайдера.

В качестве примера создадим страницу, на которой будет показан статичный график с одним интерактивным элементом. При этом по достижении бегунком слайдера заданной позиции на графике станет появляться новый элемент.

Для реализации этого примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графиков (листинг 8.97).

Листинг 8.97. Ссылка активации маршрута для формирования группы графиков

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `inter_graf5_2D` (листинг 8.98).

Листинг 8.98. URL-маршрут inter_graf5_2D

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `inter_graf5_2D` (листинг 8.99).

Листинг 8.99. Представление (view) inter_graf5_2D

Примечание.
В демо-версии книги удалены листинги программ.

В этом примере в качестве статичного и подвижного элементов используются кривые Кобба-Дугласа — это элемент экономической теории, на которой мы здесь останавливаться не станем. Отметим лишь, что данные для построения этих кривых формируются в функции с именем `data_opt`.

В приведенном в листинге 8.99 представлении `inter_graf5_2D`:

- задаются данные, необходимые для формирования кривой Кобба-Дугласа (`Px = 200`, `Py = 200`, `M = 10000`), и вычисляется максимальное количество товаров (`X_max`, `Y_max`), которые можно приобрести при заданных стоимостях и бюджете;
- определяются параметры некоторых отображаемых элементов графика: координаты точки оптимума (`xt_opt`, `yt_opt`) и координаты линий проекции точки оптимума на оси *x* и *y* (`x_lin`, `y_lin`);
- создается массив `x` (данные для оси *x*);
- в инструкции `x = np.linspace(0, 10, 101)` для оси *x* создаются 100 значений в интервале 0–10 с шагом 0.1 (0.0, 0.1, 0.2, ..., 10.0);
- создаются пустой список для хранения данных всего макета (`data`) и пустые списки для хранения данных других элементов графика:
  - `bud_data` — статичная линия бюджетного ограничения;

- `brl_data` — статичная кривая безразличия;
- `dash_data` — пунктирные линии проекций точки оптимума на оси *x* и *y*;
- `opt_data` — координаты точки оптимума;
- `u_data` — интерактивная (подвижная) кривая безразличия;

❑ формируются данные для следующих элементов графика (объекты `traces`):

- `x_b`, `u_b`, `y_b` — данные для статичной касательной линии;
- `bud_trace` — данные для статичной линии бюджета (`visible=True`);
- `brl_trace` — данные для статичной кривой безразличия (`visible=True`);
- `popt_trace` — данные для статичной точки оптимума (`visible=True`);
- `linop_trace` — данные интерактивных пунктирных линий для координат точки оптимума (`visible=False`).

Обратите внимание, что элемента статичных графиков будут видны всегда, поскольку параметр, обеспечивающий видимость, имеет значение `visible=True`. А вот последний элемент будет появляться, поэтому изначально он скрыт (`visible=False`).

❑ в цикле `for` выполняется формирование элементов `trase` интерактивного графика `u_trace` (данные для осей *x* и *y*). В нашем случае создаются 50 шагов изменения положения слайдера (50 кадров) — то есть 50 наборов `trase`. Обратите внимание, что все данные интерактивных графиков будут скрыты, поскольку параметр, обеспечивающий их видимость, имеет значение `visible=False`;

❑ в следующих инструкциях обеспечивается видимость 5-го кадра интерактивного элемента графиков `u_data` в начальном состоянии (при первом открытии шаблона Django):

```
v_step = 5
u_data[v_step].visible = True
```

❑ к объекту `data` добавляется набора данных `u_data` (после этого список `data` будет содержать четыре элемента с индексами 0, 1, 2 и 3):

```
data += u_data
```

❑ создается пустой список шагов слайдера (`steps[]`) и организовывается цикл `for`, в котором на каждом шаге слайдера все кадры интерактивных графиков скрываются и делается видимым только один кадр с индексом `s`.

Нулевая трасса будет видна всегда (по умолчанию), поэтому для нее признак видимости можно не менять. А вот графики с трассами, имеющими индекс 1 и 2, делаем видимыми принудительно для всех кадров:

```
step["args"][0]["visible"][1] = True
step["args"][0]["visible"][2] = True
```

Трассу же с индексом 3 делаем видимой только тогда, когда будет показан 22-й кадр:

```
if s == 22:
    step["args"][0]["visible"][3] = True
```

Чего мы добились в инструкциях этого цикла:

- статичные элементы графика будут видны всегда;
- динамичная линия кривой безразличия станет менять свое положение в каждом кадре;
- прерывистые линии проекции точки оптимума на оси *x* и *y* будут появляться только в кадре с номером 22.

❑ создается объект `sliders` (слайды, кадры), в который загружаются параметры всех шагов слайдера (параметры кадров каждого шага слайдера);

❑ задаются параметры макета интерактивного графика (`layout`):

- `title` — заголовок интерактивного графика;
- `xaxis` — подпись к оси *x*;
- `yaxis` — подпись к оси *y*;
- `width` — ширина макета;

- `height` — высота макета;
- `sliders` — добавить слайдер к макету;

- создается объект `fig` (фигура или график), в который передаются данные (`data`), описание макета (`layout`) и размерности осей *x* и *y*;
- с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML, и результаты этого преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4)..

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **32. Анимация с появлением элемента 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.38.

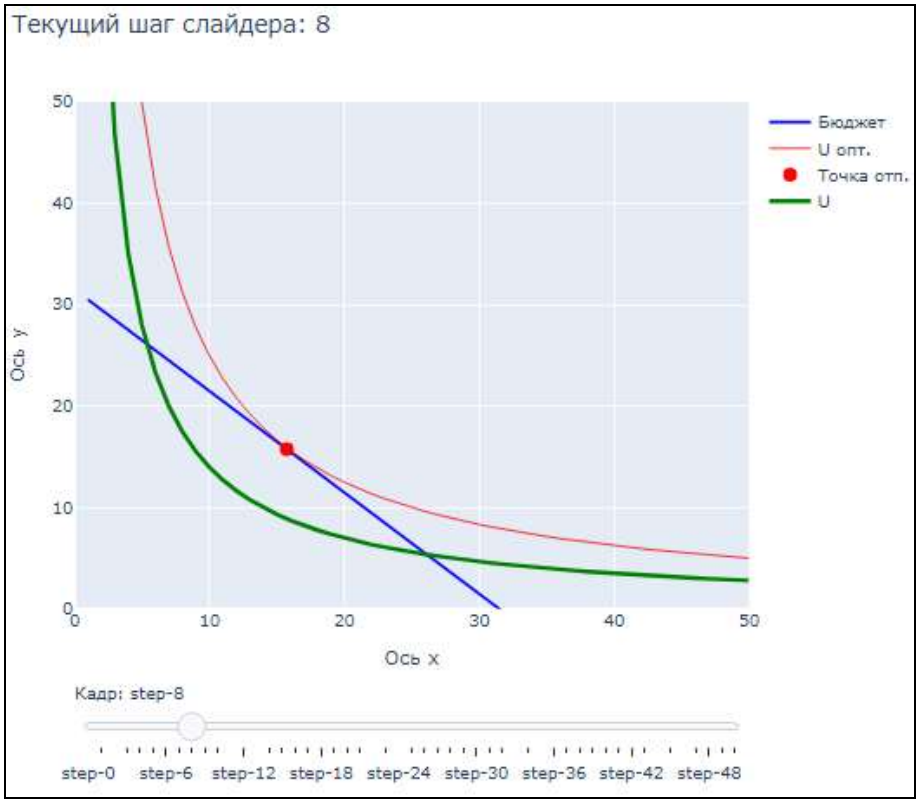


Рис. 8.38. Вид графиков при положении слайдера на шаге 8

Как здесь можно видеть, на HTML-странице отображены все статичные элементы графика и подвижная кривая безразличия.

При установке бегунка слайдера на шаг 22 (точка оптимума) на графике появится новый элемент — линии проекции точки оптимума на оси *x* и *y* (рис. 8.39).

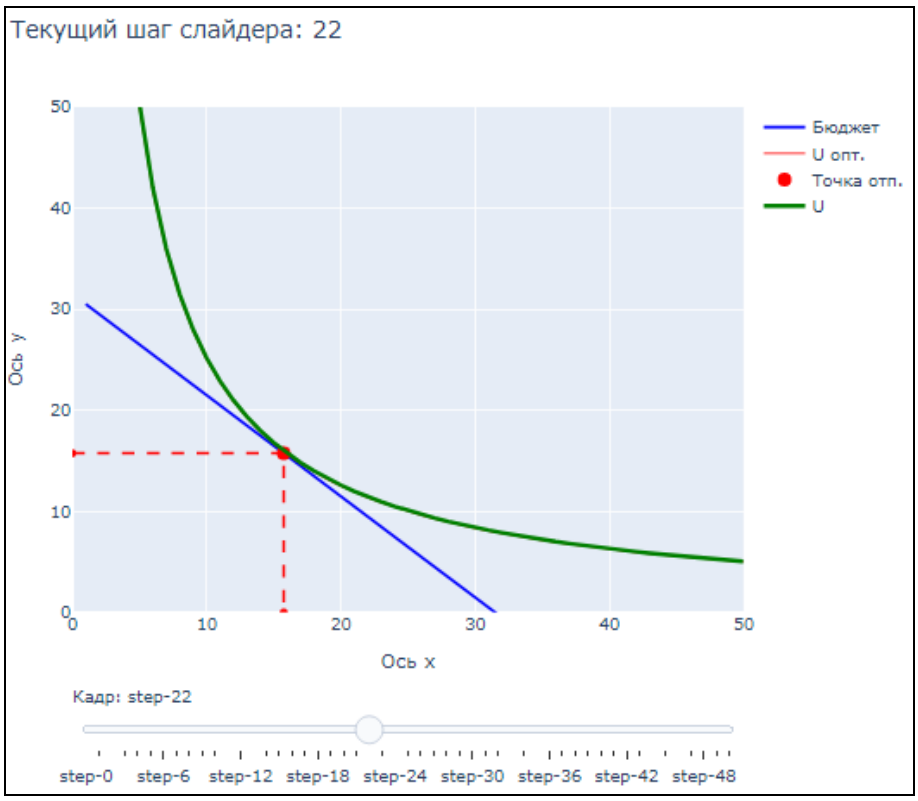


Рис. 8.38. Вид графиков при положении слайдера на шаге 22

Чем больше кадров будет создано для слайдера, тем более плавно будет меняться содержимое интерактивных графиков.

8.3.7. Анимация графиков с использованием фреймов и кнопок

Важными элементами для анимации данных являются фреймы (frames), значения которых представляют собой списки словарей. Эти словари соответствуют кадрам в анимированном графике: каждый кадр содержит свой собственный атрибут данных, а также другие параметры. Анимация обычно запускается и управляется с помощью элементов управления в виде кнопок или слайдера.

В качестве примера сформируем простейший 2D-график синусоиды, построенной по точкам, связанным с тепловой картой. Запуск и остановка анимации будет осуществляться с помощью кнопок.

Для реализации этого примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 8.100).

Листинг 8.100. Ссылка активации маршрута для построения графика линии

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `frames_2D` (листинг 8.101).

Листинг 8.101. URL-маршрут frames_2D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `frames_2D` (листинг 8.102).

Листинг 8.102. Представление (view) frames_2D

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

Это представление для генерирования данных синусоиды и формирования палитры цветов тепловой карты задействует дополнительную функцию `sin(x)`.

Сначала с использованием библиотеки NumPy в нем генерируется массив данных `x` для построения синусоиды:

```
x = np.arange(0, 9.5, 0.1)
```

Затем создается объект `fig` (пустой макет), к которому добавлены данные (`trace`). При формировании этих данных и нашла применение дополнительная функция `sin()` — на ее основе получены данные для оси *y* (`y=sin(x)`) и сформирована палитра цветов для тепловой карты (`color=sin(x)`). Все параметры, связанные с тепловой картой, загружены в свойство `marker`, которое имеет тип «словарь».

Далее создается пустой список фреймов (`frames = []`), который с использованием цикла `for` заполняется данными. В теле цикла формируются данные для осей графика *x* и *y*, а также цветовые данные для тепловой карты:

```
dict(color=sin(x[:i + 1]), size=25)
```

После чего с использованием метода `fig.frames` фреймы добавляются в макет графика:

```
fig.frames = frames
```

На следующих шагах созданы элементы с описанием параметров графика (`layout`): заголовок (`title`), подписи для осей *x* и *y*, указано положение легенды и обязательность ее вывода (`showlegend=True`). Здесь же к элементам графика добавляется важный параметр `updatemenus`, представляющий список словарей, в которых заданы параметры кнопок управления анимацией:

- словарь для задания положений кнопок в макете графика:
- словарь с параметрами кнопки запуска анимации;
- словарь с параметрами кнопки остановки анимации на паузу.

В завершение данные графика обновляются в основной фигуре (`fig.update_traces`).

После этого с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **33. Анимация диаграмм на базе Frames 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.40.

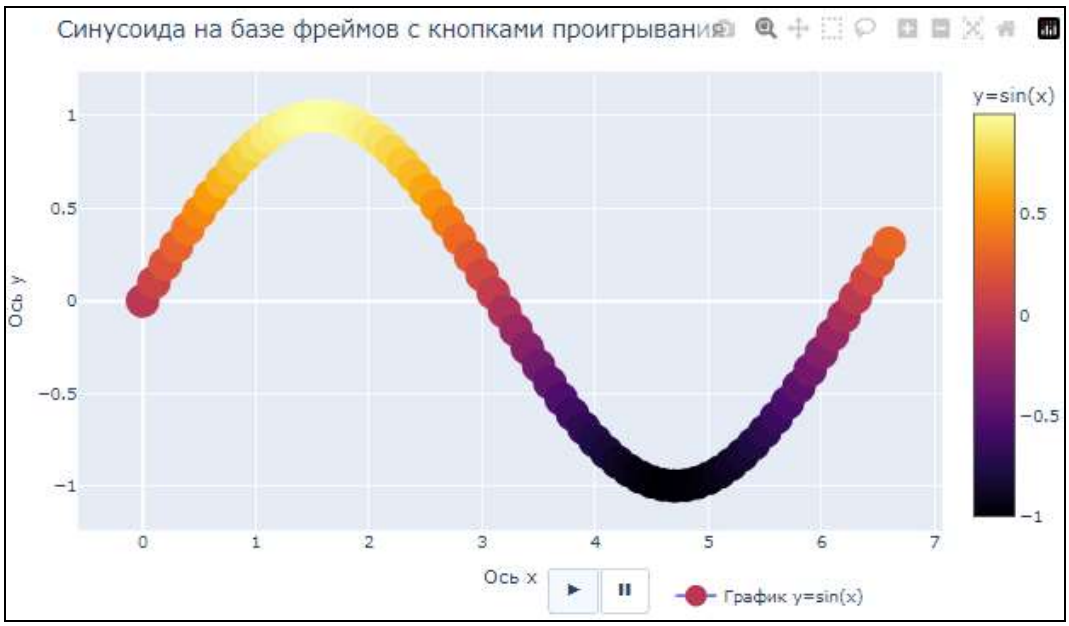


Рис. 8.40. Анимация синусоиды с использованием фреймов и кнопок управления

Как здесь можно видеть, на графике отображена цветная синусоида с маркировкой значений в виде различных цветов. В нижней части макета имеются две кнопки: ► — для запуска анимации и ▮▮— для остановки анимации на паузу. Рядом с графиком показана шкала тепловой карты с палитрой цветов, относящихся к различным значениям синусоиды.

8.3.8. Анимация графиков с использованием фреймов, кнопок и слайдера

В предыдущем разделе был показан пример, в котором анимация запускалась и управлялась с помощью кнопок. Здесь же будет рассмотрен пример, демонстрирующий, как можно управлять анимацией кнопками в сочетании со слайдером.

В качестве примера мы сформируем более сложный 2D-график, на котором выведем значения двух функций: sin(x) и cos(x) — кривые которых также построим по точкам, связанным с тепловой картой. Управление анимацией будет осуществляться как с помощью кнопок, так и с помощью слайдера.

Для реализации этого примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графиков (листинг 8.103).

Листинг 8.103. Ссылка активации маршрута для формирования графиков

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `frames2_2D` (листинг 8.104).

Листинг 8.104. URL-маршрут frames2_2D

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `frames2_2D` (листинг 8.105).

Листинг 8.105. Представление (view) frames2_2D

Примечание.

В демо-версии книги удалены листинги программ.

Сначала с использованием библиотеки NumPy в нем генерируется массив данных `x` для построения синусоиды и задается количество шагов слайдера:

```
x = np.arange(0, 9.5, 0.1)
num_steps = len(x)
```

Затем:

- создается объект `fig`, к которому добавлены данные для нулевого кадра с использованием функций `sin(x)` и `cos(x)`. На основе этих функций получены данные для оси *y*:

  ```
  y=[sin(x)[0]
  y=[cos(x)[0]
  ```

  а также сформирована палитра цветов для тепловой карты этих двух функций:

  ```
  color=[sin(x[0])]
  color=[cos(x[0])]
  ```

- все параметры, связанные с тепловой картой загружаются в свойство `marker`, которое имеет тип «словарь»;
- создается пустой список фреймов (`frames = []`), который с использованием цикла `for` заполняется данными. В теле цикла формируются данные для осей графика *x* и *y* двух функций:

  ```
  y=sin(x[:i + 1])
  y=cos(x[:i + 1])
  ```

  а также задаются цветовые данные для тепловой карты:

  ```
  dict(color=sin(x[:i + 1])
  dict(color=cos(x[:i + 1])
  ```

- с помощью цикла `for` формируется список параметров каждого шага слайдера (`steps`) и создается список самих сладов (`sliders`);
- создаются элементы с описанием параметров графика (`layout`): заголовок (`title`), подписи для осей *x* и *y*, указано положение легенды и обязательность ее вывода (`showlegend=True`). Здесь же к элементам графика добавляется важный параметр `updatemenus`, представляющий список словарей, в которых заданы параметры кнопок управления анимацией:
  - словарь для задания положений кнопок в макете графика:
  - словарь с параметрами кнопки запуска анимации;
  - словарь с параметрами кнопки остановки анимации на паузу.
- с использованием метода `fig.layout.sliders` все слайды добавляются в макет графика:

  ```
  fig.layout.sliders = sliders
  ```

- с помощью метода `fig.frames` фреймы добавляются в макет графика:

  ```
  fig.frames = frames
  ```

После этого с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга

В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **34. Анимация диаграмм на базе Frames и слайдера 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.41.

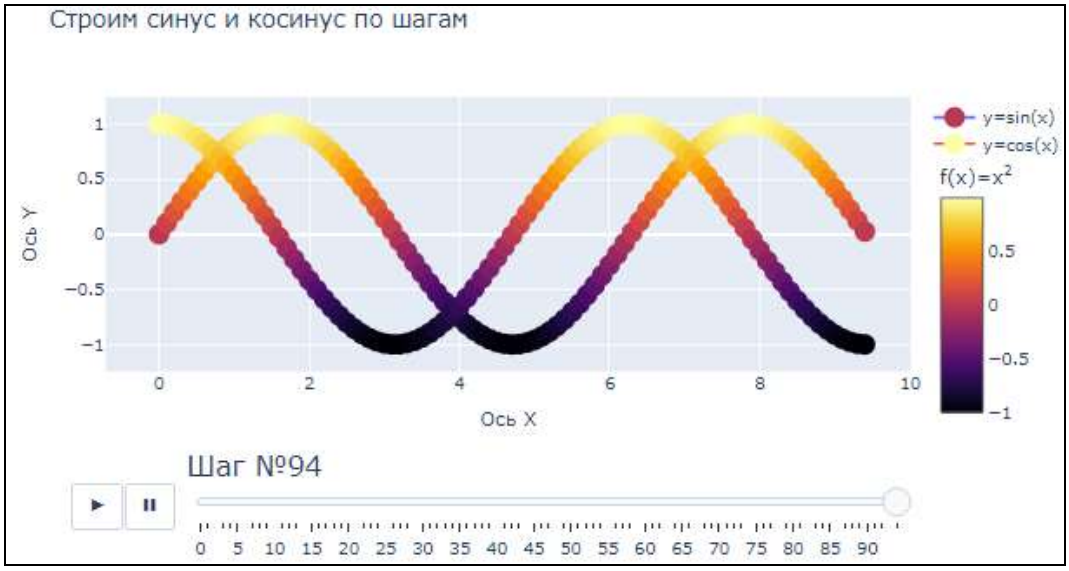


Рис. 8.41. Анимация двух графиков с использованием фреймов, кнопок управления и слайдера

Как здесь можно видеть, на графике отображены два цветных графика с маркировкой значений в виде маркеров различных цветов. В нижней части макета имеются две кнопки: ► — для запуска анимации и ▮▮— для остановки анимации на паузу, а также слайдер для управления фреймами. Рядом с графиками показана шкала тепловой карты с палитрой цветов, относящихся к различным значениям фрагментов графиков.

8.3.9. Анимация пузырьковой диаграммы

Пузырьковую диаграмму также можно анимировать с использованием кнопок и слайдера, показав на ней динамику изменения данных в зависимости от выбранного параметра.

В качестве примера сформируем пузырьковую диаграмму с анимацией на основе данных, загруженных из Интернета.

Для реализации этого примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования диаграммы (листинг 8.106).

Листинг 8.106. Ссылка активации маршрута для построения диаграммы

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `bubble_6` (листинг 8.107).

Листинг 8.107. URL-маршрут bubble_6

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `bubble_6` (листинг 8.108).

Листинг 8.108. Представление (view) bubble_6

Примечание.

В демо-версии книги удалены листинги программ.

Программный код этого модуля довольно сложный, но стоит потратить время и силы, чтобы разобраться с его фрагментами и получить возможность создавать эффектные динамичные анимированные диаграммы. Обратите внимание, что для работы модуля нужно обязательно подключить библиотеки pandas и Skimage:

```
import pandas as pd
from skimage import io
```

Здесь библиотека Skimage используется для загрузки удаленных данных из Интернета, а библиотека pandas — для выборки информации из наборов данных `dataset`. В этом примере задействован готовый набор данных, который приведен в документации на библиотеку plotly (связь между ожидаемой продолжительностью жизни и ВВП на душу населения в различные годы).

Внимание!

К использованию библиотеки Skimage следует подходить с осторожностью. Она прекрасно работает на локальном сервере разработчика, но может отказаться работать после развертывания приложения на публичном сервере Apache. Автор столкнулся с этой проблемой, как и многие другие разработчики. Наличие строки `from skimage import io` просто «обваливает» всё приложение без диагностических сообщений, и даже администраторы серверной площадке порой не могут найти причину такого сбоя.

Итак, в представлении `bubble_6`:

- с удаленного сервера скачан файл формата CSV с необходимыми данными, которые загружены в объект `dataset`:

  ```
  dataset = pd.read_csv(url)
  ```

- создан список из 11-ти элементов `years` (годы);
- создан пустой список континентов (`continents = []`), который с использованием цикла `for` заполнен данными из объекта `dataset`;
- создан словарь со следующими пустыми элементами диаграммы:
- `"data": []` — данные для диаграммы (список);
- `"layout": {}` — описание макета диаграммы (словарь);
- `"frames": []` — фреймы для анимации диаграммы (список);
- выполняется заполнение описания макета диаграммы с определением следующих параметров макета:
  - интервала значений и подписи для оси *x*;
  - подписи для оси *y*;
  - кнопки управления анимацией;
  - ряда других;
- формируется словарь с параметрами слайдера (`sliders_dict`) и задается начальное положение бегунка слайдера (`year = 1952`);
- с использованием цикла `for` из объекта `dataset` выбирается информация для диаграммы (`dataset_by_year`, `dataset_by_year_and_cont`), которая передается в словарь (`data_dict`), определяются данные для осей *x* и *y*, задаются параметры для маркеров диаграммы;
- с помощью двух вложенных циклов на основе списка годов (`years`) и списка континентов (`continents`) создается набор фреймов, которые сохраняются в словаре `fig_dict`:

  ```
  fig_dict["frames"].append(frame)
  ```

  Здесь же происходит заполнение словаря с шагами слайдера:

  ```
  sliders_dict["steps"].append(slider_step)
  ```

- после завершения работы этих двух циклов диаграмма дополняется словарем с параметрами слайдера:

```
fig_dict["layout"]["sliders"] = [sliders_dict]
```

- на основе собранного на предыдущих этапах списка словарей `fig_dict` создается объект `fig` (фигура, диаграмма), и к диаграмме добавляется заголовок:

```
fig = go.Figure(fig_dict)
fig.update_layout(title="Пузырьковая диаграмма с анимацией")
```

После этого с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **35. Анимация пузырьковой диаграммы 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.42.



Рис. 8.42. Пузырьковая диаграмма с анимацией

Как здесь можно видеть, на графике отображены данные в виде маркеров различных размеров и цветов. Маркеры находятся на разных уровнях, при этом цвет каждого маркера соответствует континенту, указанному в легенде к диаграмме. Под диаграммой расположены две кнопки: **Старт** и **Пауза**, предназначенные для запуска и приостановки анимации. Рядом с кнопками приведена шкала слайдера с бегунком, который позволяет анимировать диаграмму вручную путем перемещения бегунка с помощью указателя мыши.

При загрузке этой HTML-страницы анимация запускается автоматически — при этом на диаграмме меняется как положение цветных маркеров, так и их размеры. Если навести указатель мыши на маркер, появится всплывающая подсказка, в которой указано: название континента, название страны, ожидаемая продолжительность жизни и ВВП на душу населения (рис. 8.43).



Рис. 8.43. Пузырьковая диаграмма с дополнительной информацией на выделенном маркере

Анимацию можно запустить кнопкой **Старт** и приостановить копкой **Пауза**. Бегунок слайдера можно «зацепить» указателем мыши и перемещать по шкале с указанием годов. При этом, в зависимости от выбранного года, будет меняться как положение маркеров на диаграмме, так и их размер в соответствии с той информацией, которая содержится в CSV-файле с данными.

8.3.10. Анимация объемных 3D-срезов на примере снимков МРТ

Визуализация данных может быть чрезвычайно важна и полезна в области медицины. В этом разделе мы рассмотрим, как можно использовать визуализацию данных при анализе снимков МРТ.

> ***Магнитно-резонансная томография (МРТ)***
> МРТ — широко используемый метод медицинской визуализации, позволяющий получать подробные изображения внутренних структур человеческого тела. Визуализация данных МРТ может помочь врачам обнаружить следующее признаки болезней:
>
> - изменения анатомических структур внутренних органов;
> - изменение состава тканей;
> - потенциальные аномалии в тех или иных органах.

В этом примере снимки МРТ мы загрузим с удаленного сервера в Интернете и покажем, как можно создать послойную анимацию срезов МРТ.

Для реализации примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования визуализации (листинг 8.109).

Листинг 8.109. Ссылка активации маршрута для формирования визуализации

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `mrt_3D` (листинг 8.110).

Листинг 8.110. URL-маршрут mrt_3D

> ***Примечание.***
> В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `mrt_3D` (листинг 8.111).

Листинг 8.111. Представление (view) mrt_3D

Примечание.
В демо-версии книги удалены листинги программ.

Программный код этого модуля довольно сложный, но стоит потратить время и силы, чтобы разобраться с его фрагментами и получить возможность создавать эффектные динамичные анимированные трехмерные диаграммы. Обратите внимание, что для работы модуля можно подключить библиотеку Skimage:

```
from skimage import io
```

С использованием этой библиотеки можно было бы выполнять загрузку и чтение удаленных данных из Интернета, однако в рассматриваемом модуле эта возможность исключена. Дело в том, что библиотека Skimage прекрасно работает на локальном сервере разработчика, однако может отказаться работать на публичном сервере Apache. Исходя из этого, в нашем модуле для чтения файла формата TIF используется пакет tifffile.

Итак, в представлении `mrt_3D`:

- из каталога media сервера загружен файл attention-mri.tif (графический контейнер, хранящий растровые изображения);
- определена размерность массива данных и задано количество фреймов (`nb_frames=68`);
- с использованием метода `go.Figure` создан объект `fig`, в который загружено 68 фреймов (с индексами от 0 до 67);
- к объекту `fig` добавлен список трасс (`trace`);
- определена функция `def frame_args()`, которая необходима для задания параметров фреймам (слайдам). Эта функция используется в следующей за ней группе инструкций, создающих список слайдов, или шагов слайдера (`sliders`);
- формируется набор параметров 3D-диаграммы (`fig.update_layout`):
  - `title` — заголовок макета;
  - `width` — ширина макета;
  - `height` — высота макета;
  - `scene` — параметры сцены (трехмерного куба);
  - `updatemenus` — кнопки управления анимацией и их положение;
  - `sliders` — слайдер со слайдами.

После этого с использованием метода `fig.to_html` выполняется преобразование графика в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **36. Аанимация медицинских данных (MPT) 3D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.44.

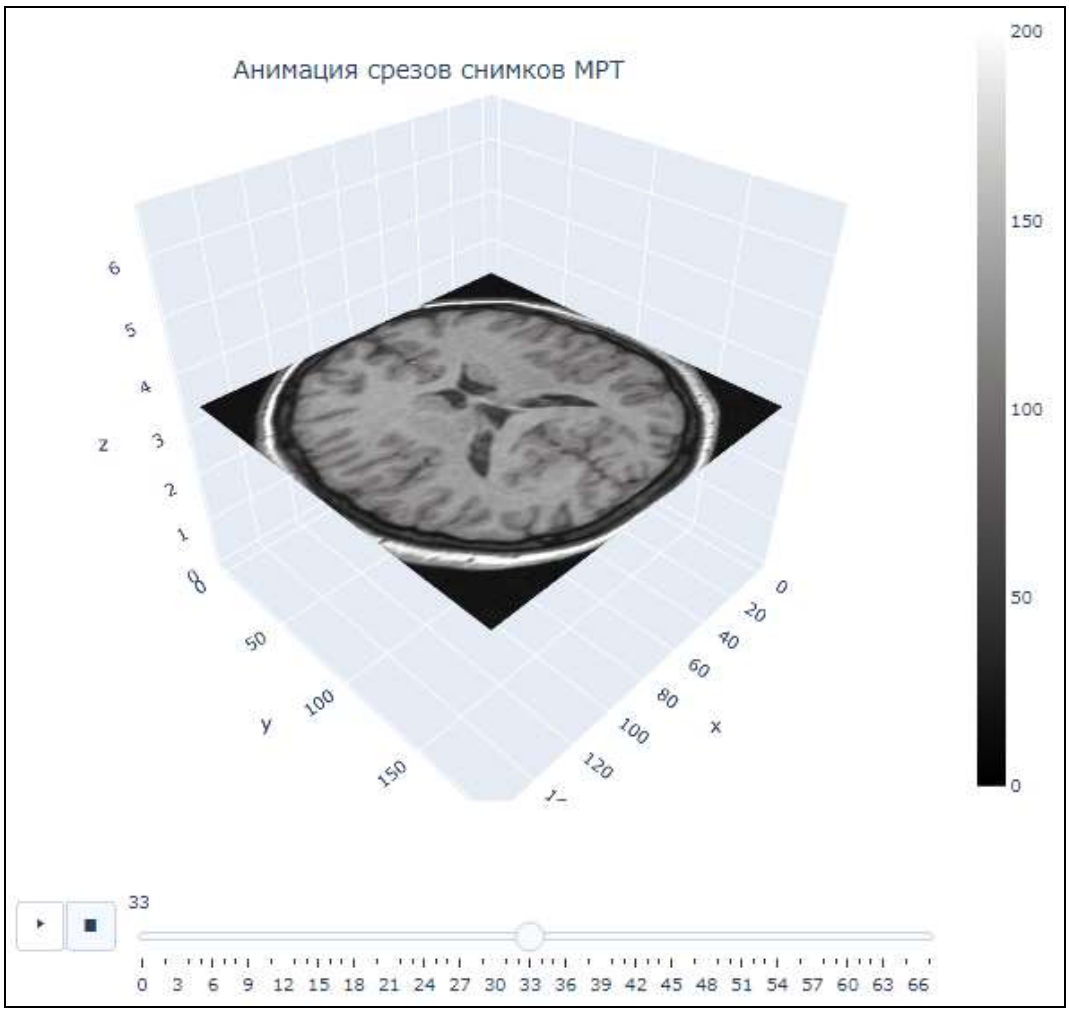


Рис. 8.44. Анимация медицинских данных (срезы снимков МРТ в формате 3D)

Как здесь можно видеть, на графике отображена трехмерная пространственная диаграмма со снимками срезов головного мозга. Пользователь, используя мышь, может поворачивать куб со снимками в любой плоскости и масштабировать изображение. Под диаграммой расположены две кнопки, предназначенные для запуска и приостановки анимации. Рядом с кнопками приведена шкала слайдера с бегунком, который позволяет анимировать диаграмму вручную путем перемещения бегунка с помощью указателя мыши, при этом будут отображаться снимки головного мозга в разных слоях.

8.3.11. Географические карты с анимацией данных пользовательских форм

В примере, приведенном в *разд. 8.1.15*, на карте отображались города — то есть стационарные объекты. Но на карте можно отображать и динамичные объекты — на основе данных, который вводит пользователь.

В этом примере мы будем строить маршруты полета воздушного транспорта между различными городами России и определять расстояния между городами, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования карты (листинг 8.112).

Листинг 8.112. Ссылка активации маршрута для формирования карты

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `map_2` (листинг 8.113).

Листинг 8.113. URL-маршрут map_2

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `map_2`, однако для его работы нам понадобится форма для ввода данных о трассе полета (листинг 8.114).

Листинг 8.114. Форма для ввода данных о трассе полета TrasForm

Примечание.
В демо-версии книги удалены листинги программ.

Будет нужна нам и функция `km`, обеспечивающая расчета расстояния между городами. Код этой функции и представления `map_2` приведен в листинге 8.115.

Листинг 8.115. Представление (view) map_2

Примечание.
В демо-версии книги удалены листинги программ.

Обратите внимание, что для работы этого модуля нужно обязательно подключить библиотеку pandas (она необходима для загрузки файла формата CSV из каталога media):

```
import pandas as pd
```

В представлении `map_2`:

- с использованием библиотеки pandas данные из локального каталога загружаются в объект `cities` (города);
- создаются два объекта: `start` и `finish`, в которые записываются названия городов начальной и конечной точек маршрута;
- с помощью метода `go.Figure` создается объект `fig` (в нашем случае карта), в который загружаются данные о городах из объекта `cities`:
  - `lat=cities['geo_lat']` — широта расположения города;
  - `lon=cities['geo_lon']` — долгота расположения города;
  - `text=cities['city']` — название города;
  - `name='Города'` — легенда;
- с использованием инструкции `if` в программе организуется разветвление: если условие в инструкции `if` не соблюдается (от пользователя поступил GET-запрос), то объект `fig` дополняется данными о маршруте полета (`fig.add_trace`). Здесь, по сути, вводятся координаты городов начала и конца маршрута:

```
lat=cities[cities['city'].isin([start, finish])]['geo_lat']
lon=cities[cities['city'].isin([start, finish])]['geo_lon']
```

- в объект `map_center` записываются координаты центра карты (координата карты, которая будет показана в центре макета), и с помощью функции `km` выполняется расчет расстояния между городами (по умолчанию будет рассчитано расстояние между городами Москва и Санкт-Петербург);
- создаются элементы с описанием параметров макета (`layout`):
  - `title` — заголовок макета;
  - `mapbox_style="open-street-map"` — подключение карты OpenStreetMap;

- `mapbox=dict(center=map_center, zoom=2)` — указание центра карты и масштаба увеличения;
- `width=1500`, `height=1000` — ширина и высота макета;
- `margin`— отступы от границ макета.

После этого с использованием метода `fig.to_html` выполняется преобразование карты в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4).

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **37. Анимация данных на географических картах 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.45.

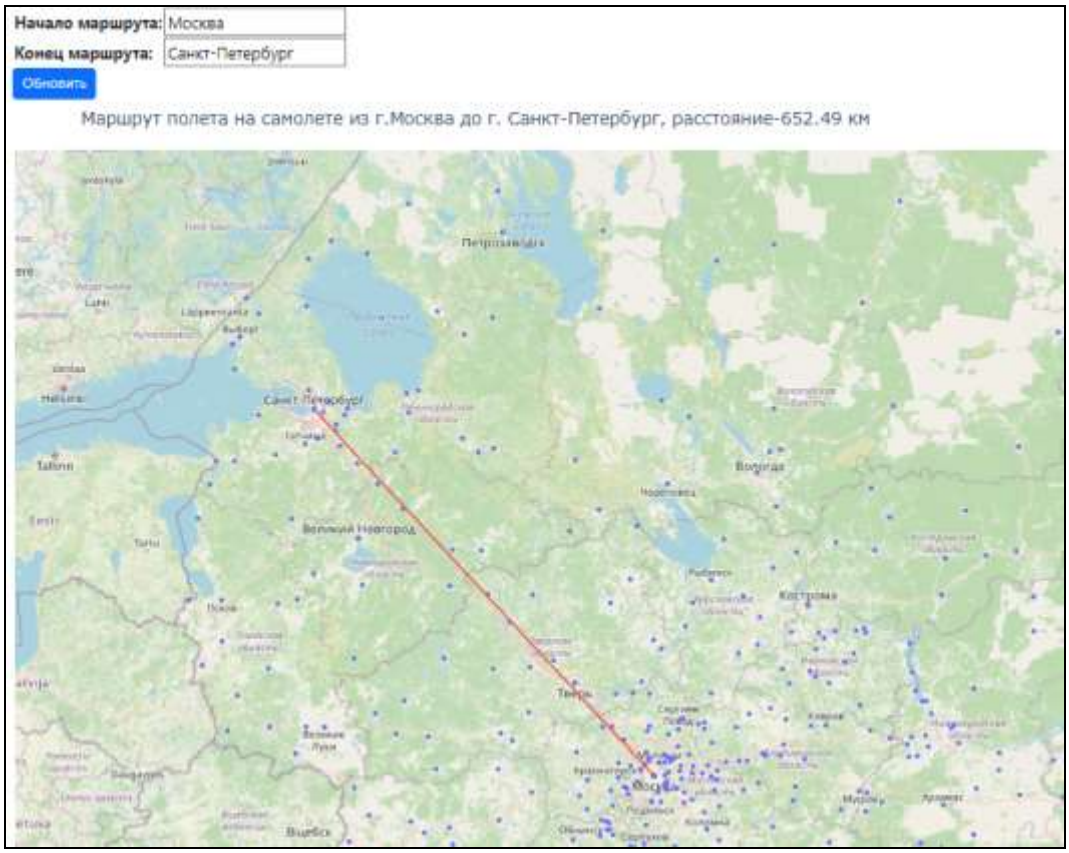


Рис. 8.45. Фрагмент карты с маршрутом полета между городами Москва и Санкт-Петербург

Как здесь можно видеть, на карте красным цветом показан маршрут полета в виде прямой линии, а в заголовке макета выведены названия городов и расстояние между ними.

> ***Примечание***
> Реальный коридор полета между городами Москва и Санкт-Петербург может отличаться от маршрута, приведенного на карте, но он будет близок к результату, показанному в этом учебном примере.

В верхней части HTML-страницы присутствует форма с названиями городов начала и конца маршрута. В эту форму пользователь может ввести названия других городов и нажать на кнопку **Обновить —** в результате произойдет обращение к представлению `map_2` уже в виде POST-запроса. При этом условие инструкции `if` будет выполнено:

```
if request.method == "POST":
```

и программа получит данные, введенные пользователем (названия других городов).

Это вызовет выполнение действий, аналогичных тем, которые были выполнены при поступлении GET-запроса: получение координат городов начала и конца маршрута, передача этих данных объекту `fig`, определение нового положения центра карты и т. д. Единственное отличие определяет фрагмент программного кода с обращением к функции расчета расстояния между городами— если пользователь сделет ошибку в наименовании города или укажет город, координат которого нет на карте, то расчет расстояния между городами выполнен не будет.

Предположим, что пользователь в качестве начальной и конечной точек маршрута указал города Москва и Казань. В этом случае будет получен следующий результат (рис. 8.46).

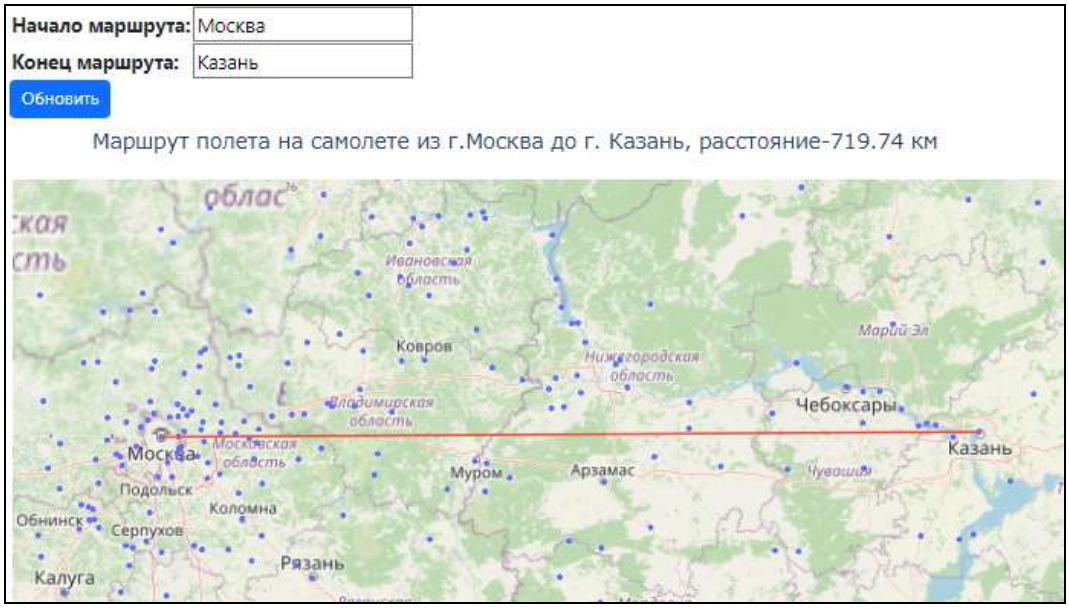


Рис. 8.46. Фрагмент карты с маршрутом полета между городами Москва и Казань

8.3.12. Географические карты с анимацией данных кнопками и слайдером

Географические карты могут содержать интерактивные элементы, которые можно «оживить» с использованием кнопок и слайдера.

В этом примере мы построим анимированный маршрут движения пассажирского поезда из Москвы в Санкт-Петербург, для чего нам потребуется создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования карты (листинг 8.116).

Листинг 8.116. Ссылка активации маршрута для построения карты

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `train` (листинг 8.117).

Листинг 8.117. URL-маршрут train

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `train` (листинг 8.118).

Листинг 8.118. Представление (view) map_2

Примечание.
В демо-версии книги удалены листинги программ.

Обратите внимание, что для работы этого модуля нужно обязательно подключить библиотеку pandas (она необходима для загрузки файла формата CSV из каталога media):

```
import pandas as pd
```

В представлении `train`:

- с использованием библиотеки pandas данные из локального каталога загружаются в два объекта: `cities` (города) и `train_russia` (маршрут поезда Москва – Санкт-Петербург);

  Пояснение
  Файл *train_russia.scv.txt* (описание маршрута поезда) содержит всего 4 колонки: `hour` (текущий час), `city` (город при движении по маршруту), `geo_lat` (широта города на карте), `geo_lon` (долгота города на карте).

- создается объект `data`, который из объекта `cities` заполняется данными с координатами городов России;
- организуется цикл `for`, в теле которого объект `data` дополняется данными о маршруте движения поезда (из объекта `train_russia`);
- с использованием метода `go.Figure` создается объект `fig` (в нашем случае карта), в который загружаются данные из объекта `data`;
- формируется набор фреймов (`frames`), или кадров, — для анимации движения поезда по маршруту с параметрами каждого шага слайдера (`steps`), и создается набор слайдов (`sliders`);
- для отображения центра карты (`map_center`) задаются координаты города Тверь;
- создаются элементы с описанием параметров макета (`layout`):
  - `title` — заголовок макета;
  - `legend_orientation="h"` — горизонтальная ориентация легенды;
  - `mapbox_style="open-street-map"`— подключение карты OpenStreetMap;
  - `mapbox=dict(center=map_center, zoom=5)` — указание центра карты и масштаба увеличения;
  - `width=1500`, `height=1000` — ширина и высота макета;
  - `margin` – отступы от границ макета;
  - `updatemenus` – параметры и расположение элементов анимации маршрута.

После этого с использованием метода `fig.to_html` выполняется преобразование карты в формат HTML и результаты преобразования сохраняются в объекте `plot_div`.

На заключительном этапе объект `plot_div` через словарь `context` передается в шаблон stat_graf_2d.html (см. листинг 8.4)..

Интерактивная цифровая книга
В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 8** и перейти по ссылке **38. Анимация объектов на географических картах 2D**.

Если запустить на выполнение код этого примера, то в окне браузера сформированная на его основе страница будет выглядеть так, как показано на рис. 8.47.

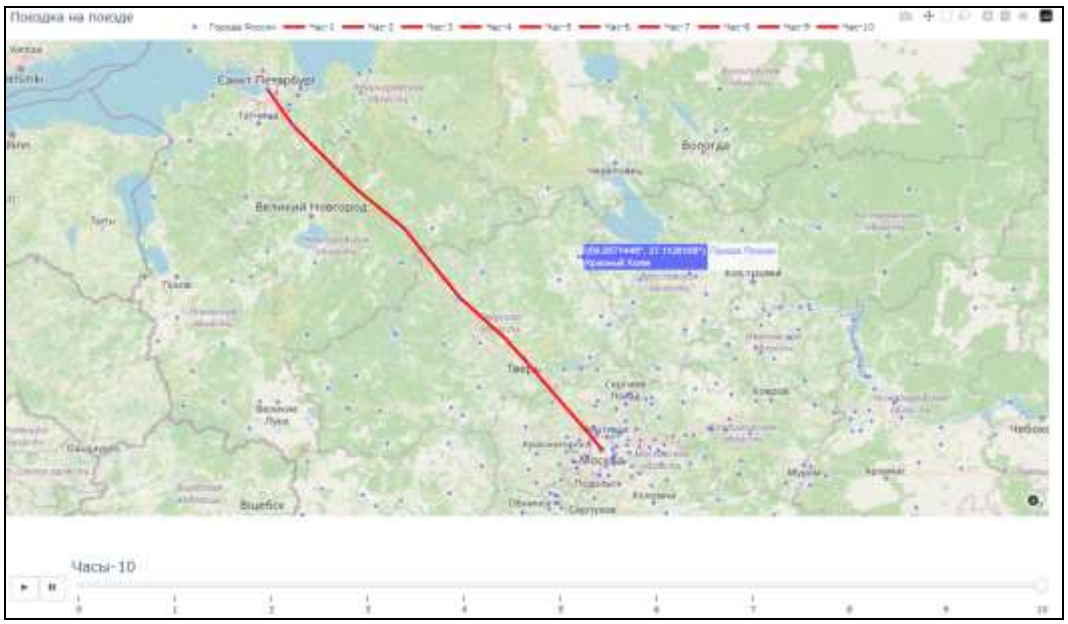


Рис. 8.47. Фрагмент карты с маршрутом движения пассажирского поезда Москва – Санкт-Петербург

Как здесь можно видеть, на карте красным цветом показан маршрут движения пассажирского поезда в виде ломаной линии. В нижней части карты имеются две кнопки: ► — для запуска и ■— для приостановки анимации, а также слайдер. Если пользователь с помощью мыши станет перемещать бегунок слайдера по шкале, то на карте будет отображаться положение поезда на маршруте после каждого часа движения.

> ***Пояснение***
> Реальное время и место положения поезда на трассе между городами Москва и Санкт-Петербург будет отличаться от анимации, приведенной в этом учебном примере, — ведь каждый поезд движется по своему расписанию. Однако в файл CSV можно внести реальное расписание движения поездов, и тогда анимация будет соответствовать реальным данным.

8.4. Краткие итоги

В этой главе приведены сведения о решении только небольшого круга задач, связанных с визуализацией данных на базе библиотеки plotly. Более детальную информацию о свойствах и методах элементов этой библиотеки можно найти в оригинальной документации, которая находится в Интернете в открытом доступе.

Кроме plotly, есть еще ряд библиотек, обеспечивающих графическое представление числовой информации: Matplotlib, Plotly Express, Dash и др. При возникновении любого вопроса, связанного с использованием этих библиотек, вы сможете на просторах Интернета найти нужный ответ от сообщества пользователей Django.

Глава 9. Визуализация данных в Django с анимацией на JavaScript

В этой главе описываются способы вывода в шаблоны Django данных в виде статичных и интерактивных 2D- и 3D-графиков с анимацией этих данных с помощью Java-скриптов, встроенных в шаблоны Django. Такая технология позволяет сократить объем программного кода в представлениях (view), однако при этом «утяжеляется» программный код шаблонов, поскольку создание «живых» интерактивных графиков требует встраивания скриптов на HTML-страницы и налаживания взаимодействия шаблонов Django с Java-скриптами. Этот подход оправдан для той группы программистов, которые имеют навыки и опыт работы с языком программирования Java.

Здесь приводятся проверенные и реально работающие программные модули, которые в том или ином виде встроены в шаблоны Django на сайтах в сети Интернет.

Из материалов этой главы вы узнаете:

- как реализовать визуализацию данных в виде статичных графиков в сочетании с анимированными 2D-графиками на основе JavaScript;
- как реализовать визуализацию данных в виде статичных графиков в сочетании с анимированными 3D-графиками на основе JavaScript;
- как реализовать обмен данными между элементами шаблонов Django и программами на JavaScript.

Создание приложений с использованием Java-скриптов требует особой аккуратности и внимательности, поскольку любая элементарная ошибка в скрипте приводит к «обвалу» целей цепочки программных модулей.

9.1. Включение JavaScript в шаблон Django

Чтобы включить JavaScript в шаблон Django, вы можете использовать либо тег Django для создания блока `{% block %}`, либо тег HTML-страницы `<script>`. При этом JavaScript может быть встроен как в базовый шаблон, так и в конкретную HTML-страницу: когда JavaScript встроен в базовый шаблон, то его функциональность может быть задействована в любом дочернем шаблоне, если же Java-скрипт встроен в конкретный шаблон с использованием тега `<script>`, то его функциональность будет использоваться только в этом шаблоне:

```
<script>
    // Ваш код на JavaScript
    function myFunction() {
        // Строки кода
    }
</script>
```

Кроме того, можно поместить тег со скриптом в тег Django для создания блока `{% block %}`, — если вы хотите разрешить дочерним шаблонам переопределять код JavaScript:

```
{% block javascript %}
    <script>
        // Ваш код на JavaScript
        function myFunction() {
            // Строки кода
        }
    </script>
{% endblock %}
```

В приведенных далее примерах Java-скрипты будут подключаться к конкретным шаблонам Django с использованием тега `<script>`.

Итак, рассмотрим несколько примеров анимации графиков с использованием скриптов java.

9.1.1. Анимация диаграмм формата 2D

В качестве примера сформируем два интерактивных графика с анимацией Java-скриптами.

В этом примере задействована используемая в экономических расчетах функция, определяющая количество взаимозаменяемых товаров *x* и *y*, которые может приобрести потребитель при заданном бюджетном ограничении. Мы не станем здесь вдаваться в детали экономической теории, отметим лишь только то, что эта функция характеризует возможности покупателя приобрести разное количество взаимозаменяемых товаров при заданных ценах и ограниченном бюджете.

Для реализации примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 9.1).

Листинг 9.1. Ссылка активации маршрута для построения графика линии

Примечание.
В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `prise_x` (листинг 9.2).

Листинг 9.2. URL-маршрут prise_x

Примечание.
В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `prise_x` (листинг 9.3).

Листинг 9.3. Представление (view) prise_x

Примечание.
В демо-версии книги удалены листинги программ.

В начале этого модуля формируются исходные данные, необходимые для расчетов, которые через словарь `context` передаются в шаблон price_x.html. Как здесь можно видеть, при использовании Java-скриптов программный код представления на Python получился достаточно простым и имеющим минимальное количество строк.

Однако при этом значительно «утяжеляется» программный код шаблона price_x.html (листинг 9.4), поскольку он должен содержать модули, написанные на JavaScript.

Листинг 9.4. Шаблон price_x.html

Примечание.
В демо-версии книги удалены листинги программ.

Это достаточно сложный шаблон с большим количеством строк программного кода на языке JavaScript, поэтому далее мы более детально рассмотрим его элементы.

Подключение библиотеки plotly.js

Для начала нужно подключить к шаблону библиотеку plotly.js. Подключение библиотеки выполняется по ссылке на CDN-сервер:

```
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
```

Тег *<style>*

Для анимации графиков в шаблоне используется HTML-слайдер. Поскольку по умолчанию слайдер имеет небольшой размер, и с ним не совсем удобно работать пользователям, то изменен его стиль. С теми параметрами, которые заданы в стиле, мы получаем более крупный, а главное — адаптивный слайд, который подстраивается под то или иное разрешение экрана.

Теги для визуализации объектов

Шаблон содержит следующие теги для визуализации объектов:

- `<div id="myDiv"></div>` — тег для отображения анимированного графика линий (идентификатор `id="myDiv"`);
- `<div id="myBar"></div>` — тег для отображения анимированной столбчатой диаграммы (идентификатор `id="myBar"`);
- `<div class="slidecontainer mx-2">` — тег для отображения слайдера (идентификатор `id="myRange"`).

Кроме того, в шаблоне имеются идентификаторы объектов для приема и показа динамичной информации из Java-скриптов:

- `id="demo"` — цена товара `x`;
- `id="pr_nx"` — количество товара `x`.

Идентификаторы всех этих элементов выполняют важную роль — они обеспечивают обмен информацией между шаблонами Django и Java-скриптами.

Тег для кода на языке Java

Тег `<script>` — самый большой тег в шаблоне. В этом теге содержится программный код для получения данных из шаблона, для возврата данных в шаблон, функции для генерации данных, модули для формирования графических объектов. Рассмотрим подробнее фрагменты этого кода.

На первом шаге выполняется объявление переменных, которые используются в скрипте:

- `var slider` — начальное значение слайдера;
- `var output` — начальное значение цены товара `x`;
- `var pr_nx` — начальное значение количества товара `x`

Затем скрипт получает из шаблона исходные параметры, необходимые для формирования данных для построения графиков:

- `let Px` — стоимость товара `x` (`Px`);
- `let Py` — стоимость товара `y` (`Py`);
- `let M` — бюджет покупателя (`M`).

А из скрипта в шаблон возвращается начальное значение слайдера и рассчитанное количество товара `x`:

```
//Вернуть начальное значение слайдера в шаблон
output.innerHTML = slider.value;
//Вернуть начальное значение pr_nx в шаблон
var n_x = Math.round(M / Px);
pr_nx.innerHTML = n_x;
```

Далее дважды делаются обращения к функции скрипта `graf_px`, в которых формируются данные для построения статичной (`x1`, `y1`) и динамичной (`x2`, `y2`) линий бюджетного ограничения, после чего формируются два объекта (`trace1` и `trace2`) — это данные для формирования статичной и динамичной линии бюджетного ограничения. Затем эти данные загружаются в один общий список (объект `data`).

На следующем шаге задаются параметры для макета диаграммы (`layout`) и формируется сама диаграмма с использованием следующего метода:

```
Plotly.newPlot('myDiv', data, layout)
```

Этому методу передаются:

- `myDiv` — идентификатор диаграммы в шаблоне Django;
- `data` — данные для построения диаграммы;
- `layout` — параметры макета диаграммы.

Эта диаграмма будет показана в следующем теге шаблона Django:

```
<div class="row">
    <!-- Интерактивный график, Line-->
    <div class="col-6 mx-2">
        <div id="myDiv"></div>
    </div>
</div>
```

После этого в аналогичной последовательности строится столбчатая диаграмма:

- подсчитывается количество товаров для столбцов диаграммы `x` и `y` (`n_x`, `n_y`);
- задается набор данных для столбчатой диаграммы (`data_bar`);
- задаются параметры макета диаграммы (`layout_bar`).

и формируется сама столбчатая диаграмма с использованием следующего метода:

```
Plotly.newPlot("myBar", data_bar, layout_bar);
```

Этому методу передаются:

- `myBar` —идентификатор диаграммы в шаблоне Django;
- `data_bar` — данные для построения диаграммы;
- `layout_bar` — параметры макета диаграммы.

Эта диаграмма будет показана в следующем теге шаблона Django:

```
<div class="row">
    <!-- Интерактивный график, Bar-->
    <div class="col-12 col-sm-6 col-md-6 col-lg-4 col-xl-4 col-xxl-2 mx-2">
        <div id="myBar"></div>
    </div>
</div>
```

Затем следует программный код главной функции, которая обеспечивает анимацию графиков:

```
// Функция изменения значения слайдера ############################
slider.oninput = function()
{
    output.innerHTML = this.value;
    sl = this.value;
        // Рисуем динамичный график бюджетных линий
    show_graf2(sl, Px, Py, M, layout);
        // Рисуем график bar
    show_bar(sl, Px, Py, M, layout_bar);
        // Меняем количество товара X
    var n_x = Math.round(M / sl);
    pr_nx.innerHTML = n_x;
}
```

Эта функция принимает текущее значение слайдера (`sl`) и передает его в другие функции, где выполняются перерасчет данных и перестроение диаграмм с учетом обновленных данных. Код этих функций мы подробно рассматривать не станем, так как он во многом аналогичен коду для построения первоначальных статичных диаграмм.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 9** и перейти по ссылке **Интерактивный график линии бюджетного ограничения**.

Если запустить на выполнение код этого примера, то мы получим два графика:

- интерактивных линий бюджетного ограничения;
- интерактивной столбчатой диаграммы.

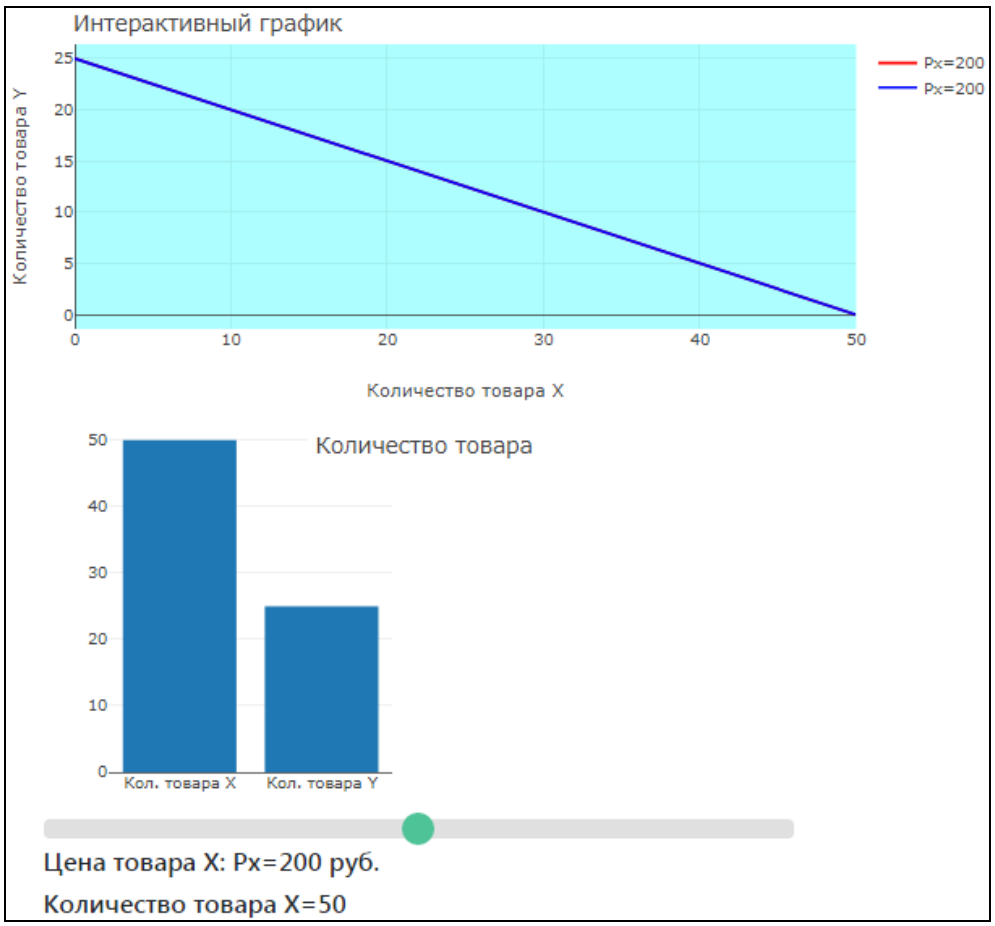


Рис. 9.1. Интерактивные графики в начальном состоянии (при Px = 200 руб.)

На рис. 9.1 приведены оба эти графика: интерактивных линий бюджетного ограничения (вверху) и столбчатой диаграммы (внизу) — в начальном состоянии (при цене товара Px = 200 руб.).

На верхнем графике присутствуют две линии бюджетного ограничения (статичная — синего цвета, и динамичная — красного цвета) — в начальном состоянии (при цене товара Px = 200 руб.) они совпадают.

На столбчатой диаграмме показано количество товаров X и Y, которые сможет купить потребитель при заданных ценах на товары.

В нижней части рисунка находится слайдер, с помощью которого можно изменять цену товара X, — если мышью менять положение бегунка слайдера, то цена на товар X станет изменяться, при этом модули, написанные на JavaScript, будут принимать новые данные из шаблона Django, выполнять необходимые расчеты, обновлять диаграммы и возвращать обновленные графики в шаблон Django. Так, если переместить бегунок слайдера крайнее правое положение, то цена на товар X достигнет значения Px = 400 руб., и графики примут вид, приведенный на рис. 9.2, — при изменении положения бегунка слайдера изменились как числовые параметры, так и сами графики.

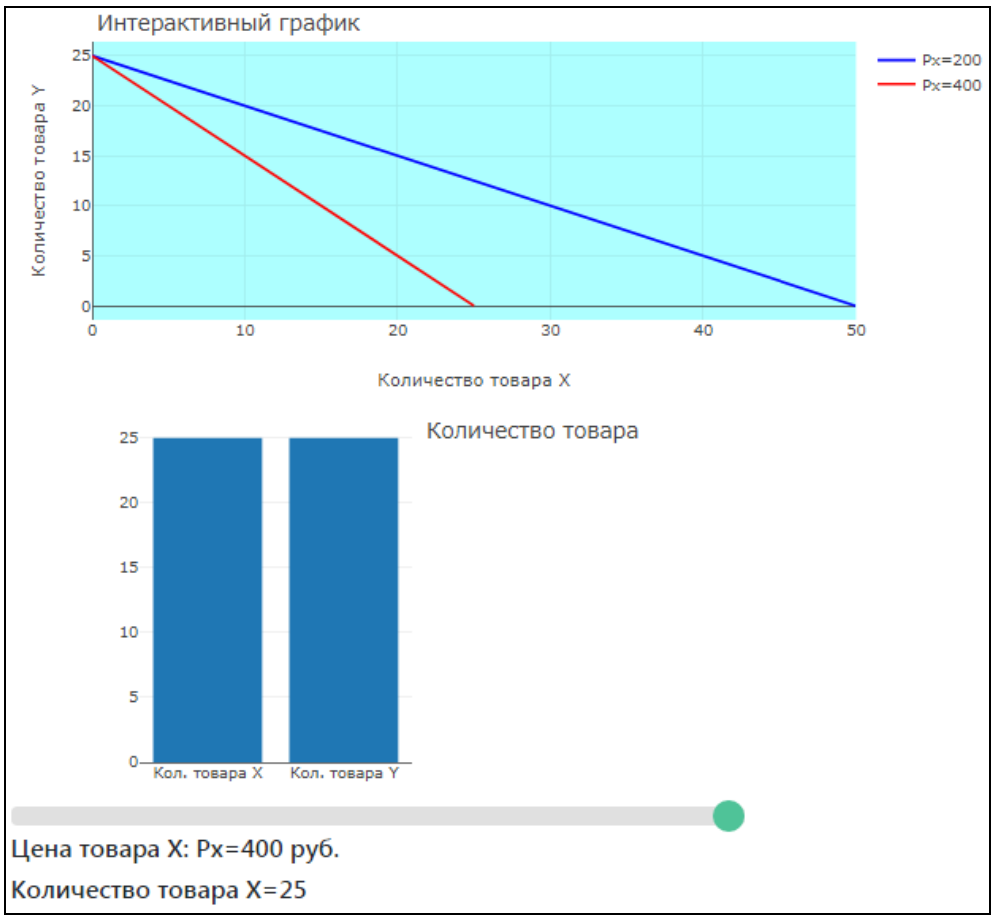


Рис. 9.2. Интерактивные графики при Px = 400 руб.

9.1.2. Анимация трехмерной диаграммы формата 3D

В качестве примера сформируем интерактивный 3D-график с анимацией Java-скриптами.

В этом примере задействована используемая в экономических расчетах функция построения кривой полезности Кобба-Дугласа при фиксированном бюджете. Мы не станем здесь вдаваться в детали экономической теории, отметим лишь только то, что эта функция характеризует форму кривой полезности при приобретении разного количества взаимозаменяемых товаров при заданных ценах и фиксированном бюджете.

В примере мы отобразим кривую полезности Кобба-Дугласа с точкой оптимума, расположенной в трехмерном пространстве, а потом будем менять значения показателей степени и следить за изменением формы этой кривой. Чтобы лучше представлять используемые алгоритмы, приведем формулу для расчета кривой полезности Кобба-Дугласа:

```
U(x,y) = X ** a * Y ** b
```

где:

- `U(x,y)` — параметр полезности потребителю от приобретения взаимозаменяемых товаров;
- `X` — количество товара *x*;
- `Y`— количество товара *y*;
- `a` и `b` — показатели степени при количестве товаров.

Для реализации примера нам понадобится создать несколько модулей, работающих в единой связке.

Прежде всего, надо в опции меню или на одной из страниц сайта создать ссылку, которая активирует нужный маршрут для формирования графика (листинг 9.5).

Листинг 9.5. Ссылка активации маршрута для построения 3D-диаграммы

Примечание.

В демо-версии книги удалены листинги программ.

По этой ссылке будет активирован URL-маршрут `f_kob_dug` (листинг 9.6).

Листинг 9.6. URL-маршрут f_kob_dug

Примечание.

В демо-версии книги удалены листинги программ.

По этому маршруту вызывается представление `f_kob_dug` (листинг 9.7).

Листинг 9.7. Представление (view) f_kob_dug

Примечание.

В демо-версии книги удалены листинги программ.

В начале этого модуля формируются исходные данные, необходимые для расчетов, которые через словарь `context` передаются в шаблон f_kob_dug.html. Как здесь можно видеть, при использовании Java-скриптов программный код представления на Python получился достаточно простым и имеющим минимальное количество строк.

Однако при этом значительно «утяжеляется» программный код шаблона f_kob_dug.html (листинг 9.8), поскольку он должен содержать модули, написанные на JavaScript.

Листинг 9.8. Шаблон f_kob_dug.html

Примечание.

В демо-версии книги удалены листинги программ.

Это достаточно сложный шаблон с большим количеством строк программного кода на языке JavaScript, поэтому далее мы более детально рассмотрим его элементы.

Подключение библиотеки plotly.js

Для начала нужно подключить к шаблону библиотеку plotly.js. Подключение библиотеки выполняется по ссылке на CDN-сервер:

```
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
```

Тег *<style>*

Для анимации графиков в шаблоне используется HTML-слайдер. Поскольку по умолчанию слайдер имеет небольшой размер, и с ним не совсем удобно работать пользователям, то изменен его стиль. С теми параметрами, которые заданы в стиле, мы получаем более крупный, а главное — адаптивный слайд, который подстраивается под то или иное разрешение экрана.

Теги для визуализации объектов

Шаблон содержит следующие теги для визуализации объектов:

- `<div id="my3D"></div>` -- тег для отображения анимированного 3D-графика (идентификатор `id="my3D"`);
- `<div class="slidecontainer mx-2">` -- тег для отображения слайдера (идентификатор `id="mySlider"`).

Кроме того, в шаблоне имеются идентификаторы объектов для приема и показа в шаблоне динамичной информации из Java-скриптов:

- `id="demo"` — текущее значение слайдера;

- `id="pr_nx"` — показатель степени при «x»;
- `id="pr_ny"` — показатель степени при «y»;
- `id="pr_x_max"` — максимальное количество товара `x`;
- `id="pr_y_max"` — максимальное количество товара `y`.

Идентификаторы всех этих элементов выполняют важную роль — они обеспечивают обмен информацией между шаблонами Django и Java-скриптами.

Тег для кода на языке Java

Тег `<script>` — самый большой тег в шаблоне. В этом теге содержится программный код для получения данных из шаблона, для возврата данных в шаблон, функции для генерации данных, модули для формирования графических объектов. Рассмотрим подробнее фрагменты этого кода.

На первом шаге выполняется объявление переменных, которые используются в скрипте:

- `var slider` — начальное значение слайдера;
- `var output` — возвращаемое в шаблон значение слайдера;
- `var pr_x_max` — возвращаемое количество товара `x`;
- `var pr_y_max` — возвращаемое количество товара `y`.

Затем скрипт получает из шаблона исходные параметры, необходимые для формирования данных для построения графиков:

- `let Px` — стоимость товара `x` (`Px`);
- `let Py` — стоимость товара `y` (`Py`);
- `let M` — бюджет покупателя (`M`).
- `let a` — показатель степени при `x` (`a`).

С использованием полученных данных делается расчет следующих значений:

- `let b = 1 - a` (показатель степени при `y` (`a`);
- `let X_max = a * M / Px` (максимальное количество товара `x`);
- `let Y_max = b * M / Py` (максимальное количество товара `y`).

После чего из скрипта в шаблон возвращается начально значение слайдера и ряд рассчитанных в скрипте параметров:

```
//Вернуть начальное значение слайдера в шаблон
output.innerHTML = slider.value / 100;
let sl = slider.value / 100;
let a1 = sl
let b1 = 1 - a1

//Вернуть начальное значение pr_nx, pr_ny в шаблон
var n_x = a1;  //Показатель степени a при x
var n_y = b1;  //Показатель степени b при y
pr_nx.innerHTML = n_x;
pr_ny.innerHTML = n_y;
pr_x_max.innerHTML = X_max;
pr_y_max.innerHTML = Y_max;
```

Далее делается обращение к функции Кобба-Дугласа `f_p_kd`, из которой возвращаются данные для построения трехмерной диаграммы. Эти данные записываются в переменные `x3`, `y3` и `z3`:

```
// Обращение к функции формирования значений x3, y3, z3
let x_y_z = f_p_kd(Px, Py, M, a1, b1);
var x3 = x_y_z[0];  // Массив значений x3
var y3 = x_y_z[1];  // Массив значений y3
var z3 = x_y_z[2];  // Массив значений z3
```

После этого формируются два объекта (`trace3` и `trace4`) — данные для формирования динамичной кривой полезности и точки оптимума. Затем эти данные загружаются в один общий список (объект `data`).

На следующем шаге задаются параметры для макета диаграммы (`layout1`) и формируется сама диаграмма с использованием следующего метода:

```
Plotly.newPlot('my3D', data, layout1)
```

Этому методу передаются:

- `my3D` — идентификатор диаграммы в шаблоне Django;
- `data` – данные для построения диаграммы;
- `layout1` — параметры макета диаграммы.

Эта диаграмма будет показана в следующем теге шаблона Django:

```
<div class="row">
    <!-- График, трехмерный-->
    <div class="col-5 mx-2">
        <div id="my3D"></div>
    </div>
</div>
```

Затем следует программный код главной функции, которая обеспечивает анимацию графиков: `slider.oninput = function()`. Эта функция принимает текущее значение слайдера (`sl`), делает перерасчет данных и выполняет обновление трехмерной диаграммы с учетом обновленных данных. Код этой функции мы подробно рассматривать не станем, так как он во многом аналогичен коду для построения первоначальной диаграммы.

> ***Интерактивная цифровая книга***
> В интерактивной цифровой книге (см. *приложение 1*) для запуска этого примера нужно на вкладке **Показ примеров** нажать на кнопку **Примеры главы 9** и перейти по ссылке **Кривая полезности Кобба-Дугласа при M-const**.

Если запустить на выполнение код этого примера, то мы получим 3D-диаграмму функции полезности Кобба-Дугласа с выделенной точкой оптимума. На рис. 9.3 приведена эта 3D-диаграмма в начальном состоянии (при показателях степени a = 0.5 и b = 0.5) — с помощью мыши диаграмму можно вращать в любой плоскости.

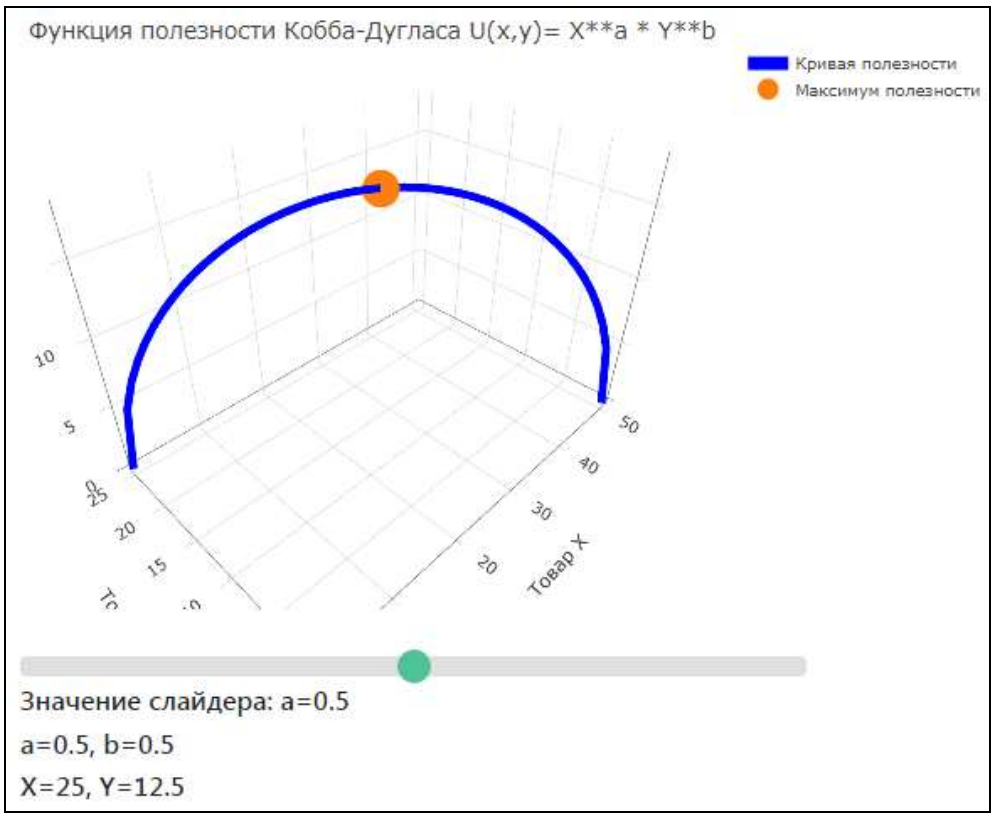


Рис. 9.3. Интерактивная 3D-диаграмма в начальном состоянии (при a = 0.5 и b = 0.5)

Если мышью менять положение бегунка слайдера, то модули, написанные на JavaScript, будут принимать новые данные из шаблона Django, выполнять необходимые расчеты, об-

новлять диаграмму и возвращать обновленный график в шаблон Django (рис. 9.4). Как здесь можно видеть, при изменении положения бегунка слайдера изменились как числовые параметры (значения показателей степени a и b и количества товаров X и Y), так и форма кривой полезности на графике и положение точки оптимума.

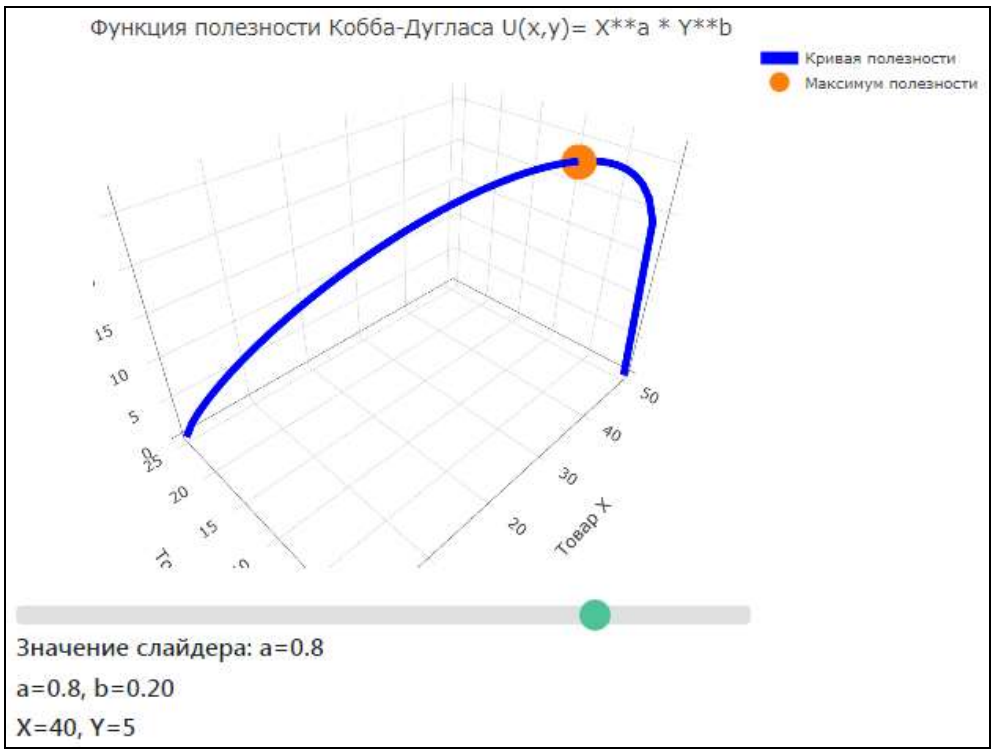


Рис. 9.4. Интерактивная 3D-диаграмма при значениях показателей степени a = 0.8 и b = 0.2

При перемещении указателя мыши вдоль кривой полезности координаты текущих точек будут отображаться на координатной оси. Если указатель мыши остановить на точке максимума красного цвета, то будет показано оптимальное сочетание количества товаров X и Y при заданном бюджетном ограничении (рис. 9.5).

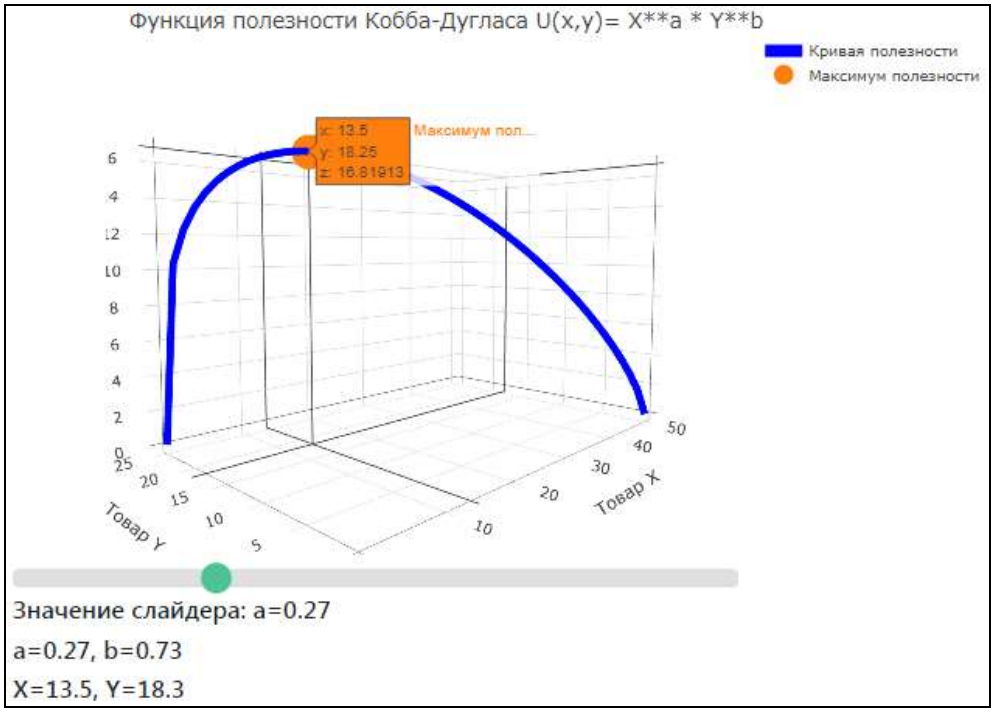


Рис. 9.5. Интерактивная 3D-диаграмма с указанием координат точки максимальной полезности

9.2. Краткие итоги

В этой главе приведены сведения о решении только небольшого круга задач, связанных с визуализацией данных на базе библиотеки plotly.js, и с Java-скриптами, встроенными в шаблоны Django. Более детальную информацию о свойствах и методах элементов этой библиотеки можно найти в исходной документации, которая находится в Интернете в открытом доступе.

При использовании Java-скриптов значительно упрощается программный код представлений (view), написанных на Python, однако при этом усложняется код шаблонов Django, так как в них встраивается код на языке JavaScript. Исходя из этого, библиотеку plotly.js рекомендуется использовать программистам, имеющим опыт и навыки написания программ на JavaScript. Тем же программистам, которые испытывают некоторые сложности при работе с Java-скриптами, будет удобнее и эффективнее использовать библиотеку plotly.py (см. *главу 8*).

Послесловие

В этой книге и в сопровождающей ее интерактивной цифровой платформе рассмотрены только некоторые приемы разработки веб-приложений, обеспечивающие создание статичных и динамичных элементов сайтов. Примеры и шаблоны HTML-страниц по возможности максимально упрощены. Основная цель книги — познакомить читателей с некоторыми тонкостями создания приложений на Django, принципами взаимодействия между шаблонами этого фреймворка с элементами фреймворка Bootstrap. Весьма ценно, что Django позволяет общаться к таблицам СУБД не напрямую через SQL-запросы, а через модели данных и через пользовательские формы. При этом использование Bootstrap значительно облегчает программирование пользовательского интерфейса.

Если после знакомства с приведенными материалами вы почувствовали, что Python и Django прекрасно взаимодействуют с Bootstrap, и что применение этого набора инструментов упрощает программистам разработку веб-приложений, то основная цель издания достигнута.

В книге охвачены далеко не все тонкости разработки веб-приложений. Достаточно актуальными являются вопросы удаленной загрузки и хранения изображений, а также создания географических веб-приложений. Для этого у Django есть прекрасное расширение — GeoDjango, а также библиотека Mapnik и геопространственное расширение PostGIS для СУБД PostgreSQL. Набор этих модулей позволяет максимально упростить создание географических веб-приложений и служб, основанных на местоположении. Кроме того, актуальными являются такие темы, как компьютерное зрение и системы искусственного интеллекта, где также используется Python и приложения на базе веб-технологий. Если это издание окажется востребованным, то следующим шагом автора, будет подготовка книги и интерактивной цифровой платформы по освещению этих технологий.

А в завершении остается пожелать удачного применения в своей практической деятельности тех навыков и знаний, которые вы приобрели при изучении материала этой книги.

Анатолий Постолит
д.т.н., профессор

Список источников и литературы

1. Верстайте адаптивный дизайн быстро и легко. **https://bootstrap-v5.ru/**.
2. Великолепный скрипт падающего снега на сайт на JS+CSS + сугробы!. **https://alaev.info/blog/post/9431?ysclid=lqmu8ngtxe230170405**.
3. Виртуальный хостинг на timeweb. **https://timeweb.com/ru/**.
4. Вкладки Bootstrap для сайта. **https://itchief.ru/bootstrap/tabs?ysclid=lp8ejxxfpo358100571**.
5. Главное руководство по редиректам Django. **https://www.codeflow.site/ru/article/django-redirects**.
6. Делаем сами: адаптивный сайт. **https://thecode.media/bootstrap/?ysclid=lq57ka6fg498611713**.
7. Древовидное представление со стрелкой. **https://www.schoolsw3.com/howto/tryhow_js_treeview.php?ysclid=ln78v19vib685266690**.
8. Как отображать файлы другого формата (.docx / .pdf / .txt) в шаблонах Django. **https://question-it.com/questions/2740675/kak-otobrazhat-fajly-drugogo-formata-docx-pdf-txt-v-shablonah-django**.
9. Как встроить видеоплеер с тегом HTML 5 Video. **https://dev-gang.ru/article/htmlvideo--kak-vstroit-videopleer-s-tegom-html--video-2q3ix69qmm/**.
10. Как сделать так, чтобы родительский элемент дерева в древовидном представлении был ссылкой. **https://digitrain.ru/questions/63379727/?ysclid=ln7a1jltzu686829280**.
11. Как сделать вкладки в HTML. **https://myrusakov.ru/js-tabs.html**.
12. Катастрофический дефицит. Цифровому прорыву предрекли острую нехватку IT-специалистов. **https://www.dp.ru/a/2020/01/24/Katastroficheskij_deficit**.
13. Компоненты. Десятки полезных компонентов встроены в Bootstrap. **https://bootstrap-ru.com/components.php**.
14. Легко настраивайте базу данных Windows SQLite и управляйте ею. **https://www.filehorse.com/download-sqlitestudio/**.
15. Модальное окно. **https://getbootstrap.ru/docs/5.1/components/modal/?ysclid=lo9047rwke363959463.**
16. Модальное окно Bootstrap для сайта. **https://itchief.ru/bootstrap/modal**.
17. Модели. Django ORM. **https://ru.hexlet.io/courses/python-django-basics/lessons/models/theory_unit**.
18. Навигация и вкладки. **https://bootstrap-5.ru/docs/5.0/components/navs-tabs/?ysclid=lp84mg3zwc988494364**.
19. Основы JavaScript. **https://html5book.ru/osnovy-javascript/**.
20. Отношения между моделями в Django. **http://sheregeda.github.io/blog/2015/01/30/models-in-django/**.
21. Постолит А. django_world_book на сайте **github.com**. **https://github.com/APostolit/django_world_book**.
22. Постолит А. Python, Django и Bootstrap для начинающих. Санкт-Петербург. «БХВ-Петербург», 2023. с. 623.
23. Прелоадер или загрузка другой страницы в ожидании ответа api (Django). **https://itfy.org/threads/preloader-ili-zagruzka-drugoj-stranicy-v-ozhidanii-otveta-api-django.1058/**.
24. Работа программистом Python: требования, вакансии и зарплаты. **https://pythonru.com/baza-znanij/python-vakansii**.
25. Руководство по веб-фреймворку Django. **https://metanit.com/python/django/**.
26. Сайт для тех, кто изучает веб-технологии и создает сайты. **https://html5book.ru/**.

27. Свойство CSS background. **https://zarabotat-na-sajte.ru/uroki-html/kak-sdelat-fon-dlya-sajta.html**.
28. Таблица цветов HTML. **https://zarabotat-na-sajte.ru/uroki-html/html-cveta-i-kodi-cvetov.html**.
29. Тег <style> в HTML. **https://zarabotat-na-sajte.ru/uroki-html/style-v-html.html**.
30. Тег bootstrap-table с использованием горизонтальной полосы прокрутки. **https://russianblogs.com/article/6934895780/**.

    Шаблоны Django. Наследование. **https://habr.com/ru/post/23132/**.
31. Django CRUD with Bootstrap Modal Form with HTMX. **https://adiramadhan17.medium.com/django-crud-with-bootstrap-modal-form-with-htmx-3cbe3830ecfa**.
32. Django: How to use modal as a form to update an object. **https://davy.ai/django-how-to-use-modal-as-a-form-to-update-an-object/?ysclid=lo8z2h487z731891779**.
33. Django реализует простое всплывающее предупреждение веб-страницы. **https://russianblogs.com/article/2578120604/**.
34. Django Bootstrap Modals 4. **https://django-bstrap-modals.readthedocs.io/en/latest/bsmodals.html**.
35. Django detailview in bootstrap modal. **https://davy.ai/django-detailview-in-bootstrap-modal/?ysclid=loczirch5254632135**.
36. Django работа с переменными в шаблоне. **https://indigotlt.ru/django-rabota-s-peremennymi-v-shablone/**.
37. Django База [2023]: Регистрация с подтверждением email адреса. **https://proghunter.ru/articles/django-base-2023-registration-with-email-address-verification-24?ysclid=lqmtsaaccr996036906**.
38. Django User Registration with Email Confirmation. **https://studygyaan.com/django/how-to-signup-user-and-send-confirmation-email-in-django?utm_content=cmp-true**.
39. How to add a spinner icon to button when it's in the Loading state?

    ttps://stackoverflow.com/questions/14793367/how-to-add-a-spinner-icon-to-button-when-its-in-the-loading-state.
40. How to Display Messages after Form Submit in Django and Bootstrap? **https://www.csestack.org/display-messages-form-submit-django/#google_vignette**.
41. How to use Django Bootstrap Modal Forms. **https://dihfahsih.blogspot.com/2019/08/how-to-use-django-bootstrap-modal-forms.html**.
42. How to implement modal popup django forms with bootstrap. **https://www.abidibo.net/blog/2014/05/26/how-implement-modal-popup-django-forms-bootstrap/**.
43. How to Handle Django Forms within Modal Dialogs. **https://dev.to/djangotricks/how-to-handle-django-forms-within-modal-dialogs-4k8b**.
44. HTML Цвета. https://open-source-**peace.github.io/w3schoolsrus/html/html_colors.html#gsc.tab=0**.
45. Modal Dialogs in Bootstrap. **https://htmx.org/examples/modal-bootstrap/**.
46. Modified Preorder Tree Traversal in Django**. https://stackabuse.com/modified-preorder-tree-traversal-in-django/**.
47. Responsive Vertical Navigation Tabs for Boostrap 5. **https://www.npmjs.com/package/bootstrap-5-vertical-tabs**.
48. Simple modal to open details of an item (Django + bootstrap). **https://medium.com/@antoinewg/simple-modal-to-open-details-of-an-item-django-bootstrap-ffeeb11f12c1**.
49. Tooltips. Use tooltips to indicate extra content for your users when hovering over an element. **https://themesberg.com/docs/django/volt-dashboard/components/tooltips/.**
50. WEB учебник. **https://www.schoolsw3.com/**.
51. What’s New in Bootstrap 5. **https://www.sitepoint.com/bootstrap-5-new-features-examples/**.

Приложение 1. Порядок работы с интерактивной цифровой платформой

Получать новые знания можно различными способами — путем изучения материалов книг, журналов, документов, а также на основе общения с преподавателями (на уроках, лекциях, конференциях, семинарах, тренингах). При этом сама изучаемая информация может храниться в различных видах: на бумажных или электронных носителях. В последние годы всё большую популярность приобретают цифровые носители информации — электронные книги. Такой способ хранения и предоставления учебной информации имеет следующие преимущества перед бумажными носителями:

- компактность и портативность;
- возможность быстрого контекстного поиска;
- возможность использования аудио- и видеоконтента;
- более низкая стоимость и более короткое время на издание и тиражирование;
- высокая скорость и удобство распространения через сеть Интернет и по электронной почте;
- экологичность (на издание электронных книг не расходуется древесина).

Особенно эффективно использовать цифровые книги при подготовке IT-специалистов. Изучение программирования по бумажным книгам — это уже устаревший и малоэффективный способ обучения. Во-первых, к моменту выхода книги в свет продукт, описываемый в ней, может настолько сильно измениться, что материалы книги окажутся уже не актуальны, — ведь IT-технологии меняются очень быстро. Во-вторых, для эффективного обучения необходима постоянная практика. Нужно не просто читать теорию и просматривать примеры программного кода — надо тут же самостоятельно писать программный код, запускать его на исполнение и исправлять ошибки — чтобы в цикле «тестирование-отладка» получить работающий модуль.

В книгах по программированию приводятся листинги программ, и их можно набрать на своем компьютере. Однако, во-первых, обучающийся тратит много времени на то, чтобы с клавиатуры заново набирать этот текст, — при этом возможны ошибки и опечатки, требующие много времени на их поиск. Во-вторых, при типографском издании книг в листингах программ часто игнорируются требуемые отступы и пробелы между фрагментами программного кода, что является нарушением их синтаксиса. К таким синтаксическим ошибкам особенно чувствителен язык программирования Python, и их достаточно трудно искать, особенно, новичкам. В-третьих, для работы этих программ часто требуется подключение дополнительных библиотек и сторонних модулей, которые приходится самостоятельно загружать из Интернета, причем они имеют много различных версий, не всегда «дружащих» между собой, а некоторые просто не совместимы в рамках одного программного модуля.

В таких ситуациях, если программа не работает, новичок просто теряется и не знает, где искать возникшие ошибки и как их исправить. В итоги читатели пишут гневные письма в издательство с претензиями на то, что примеры, приведенные в книге, не рабочие, а порой и требует возврата денег.

В этом приложении приводятся инструкции для читателей цифровых интерактивных книг (платформ), рассказано о том, как читать такие книги и просматривать их медиаконтент, а также как покупать платные цифровые интерактивные книги.

> ***Интерактивная цифровая платформа***
>
> Работать с интерактивной цифровой платформой можно, зайдя в нее по адресу **rumnil.ru**. Демо-версию платформы можно найти в опции **Каталог** ее главного меню. Полная версия платформы будет доступна в разделе **Мои книги** личного кабинета зарегистрированного пользователя после ее приобретения в магазине сайта.
>
> Более того, использование полной версии платформы избавляет вас от прибретения «бумажной» версии книги — если вам это покажется более удобным…

Просмотр карточек с загруженными книгами

Читатели, которые не проходили регистрацию на образовательной платформе «УМНИК», могут открывать и просматривать только бесплатные книги с открытым доступом.

Для доступа к цифровым книгам необходимо в меню главной страницы образовательной платформы перейти по ссылке **Каталог** (рис. П.1) — в результате откроется **Панель каталогов**, на которой можно выбрать тип каталога книг: **Статичный** или **Дерево** (рис. П.2).

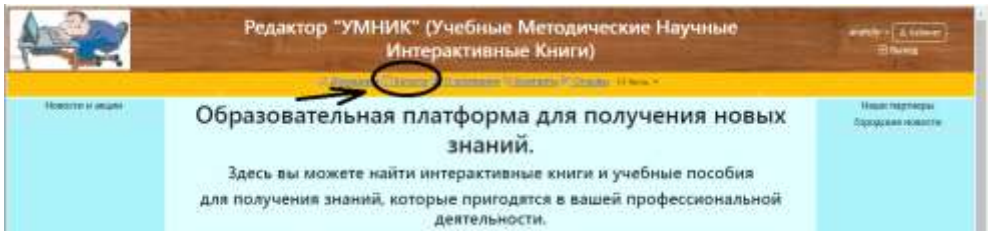


Рис. П.1. Опция **Каталог** в главном меню образовательной платформы «УМНИК»

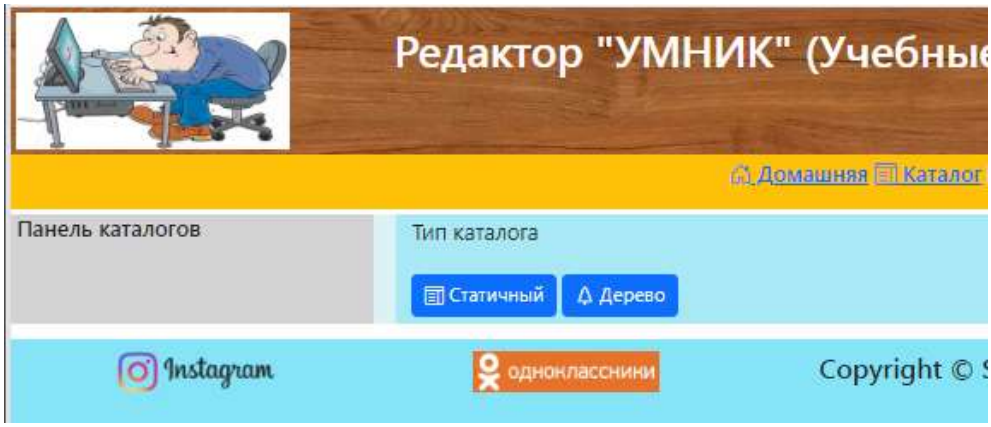


Рис. П.2. Варианты выбора типа каталога в Панели каталогов

По нажатию на кнопку **Статичный** откроется вкладка, содержащая на левой боковой панели полностью раскрытое (статичное) дерево категорий книг (рис. П.3).

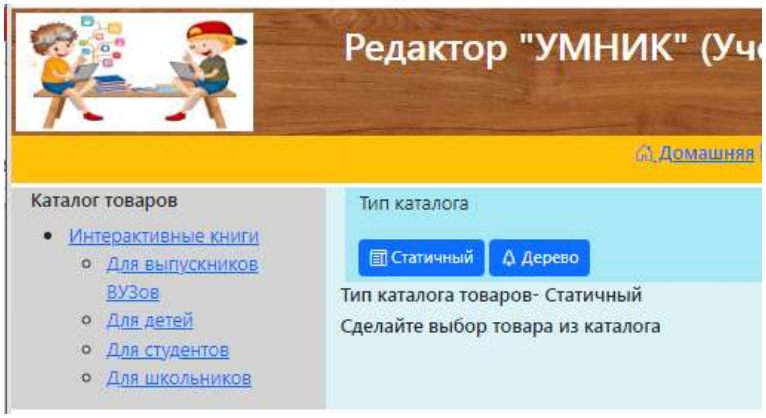


Рис. П.3. Статичный каталог категорий книг в виде полностью раскрытого дерева

По нажатию на кнопку **Дерево** откроется вкладка, содержащая на левой боковой панели динамичное раскрывающееся (но пока свернутое) дерево категорий книг (рис. П.4).

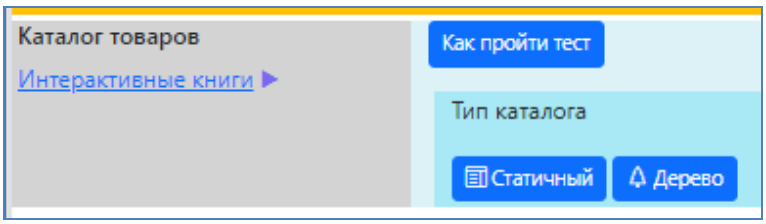


Рис. П.4. Динамичный каталог категорий книг в свернутом виде

При первой загрузке динамичного каталога отображается только корень дерева и ссылка **Интерактивные книги** для раскрытия его дочерних узлов (веток) следующего уровня. С помощью этой ссылки пользователь может раскрывать (сворачивать) дочерние узлы (рис. П.5).

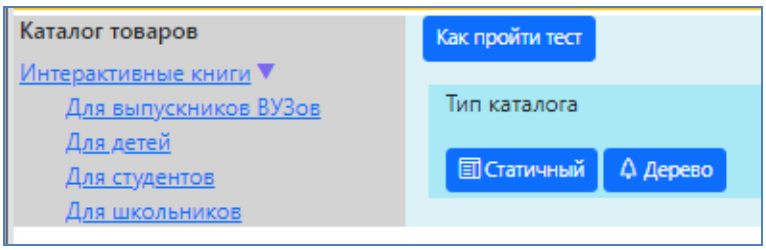


Рис. П.5. Динамичный каталог категорий книг в развернутом виде

В каталоге можно выбрать любую категорию книг, после чего в центральной панели откроются карточки книг, которые загружены в БД (рис. П.6).



Рис. П.6. Карточки интерактивных цифровых книг

Каждая карточка содержит следующую информацию о книге:

- категорию, к которой отнесена книга в каталоге;
- кнопку **Открыть книгу** (или **Открыть веб-книгу**) — для книги с нулевой стоимостью, которая доступна для всех пользователей;
- кнопку **Как купить книгу** (или **Открыть купленную книгу**) — для книг с заявленной стоимостью, доступных только зарегистрированным пользователям;

- изображение обложки книги;
- название книги в виде ссылки, переход по которой щелчком левой кнопки мыши откроет страницу с краткой аннотацией книги (рис. П.7).
- цену книги;
- количество просмотров книги пользователями Интернета.



Рис. П.7. Краткая аннотация интерактивной цифровой книги

Как уже отмечено ранее, в верхней части карточки книги присутствует кнопка **Открыть книгу** или **Открыть веб-книгу** — если книга находится в свободном доступе. По нажатию кнопки **Открыть книгу** будут открыты вкладки типовой цифровой интерактивной книги, загруженной через личный кабинет автора. По нажатию кнопки **Открыть веб-книгу** будут открыты вкладки уникальной цифровой интерактивной книги, которая создается в инструментарии Django как отдельное приложение и имеет специфический набор вкладок и библиотек для запуска содержащихся в ней примеров. Она может быть загружена на образовательную платформу только администратором сайта. Цифровая интерактивная книга «Мастер веб-разработки на Python и Django» относится этой категории, и для ее просмотра будет активна кнопка **Открыть веб-книгу**, но в бесплатном варианте станет доступна только демо-версия книги, из которой удалены листинги программ.

Если книга платная, и пользователь ее еще не купил, то кнопка будет иметь надпись **Как купить книгу**, и по ее нажатию откроется модальное окно, содержащее информацию о правилах приобретения платных книг (рис. П.8).

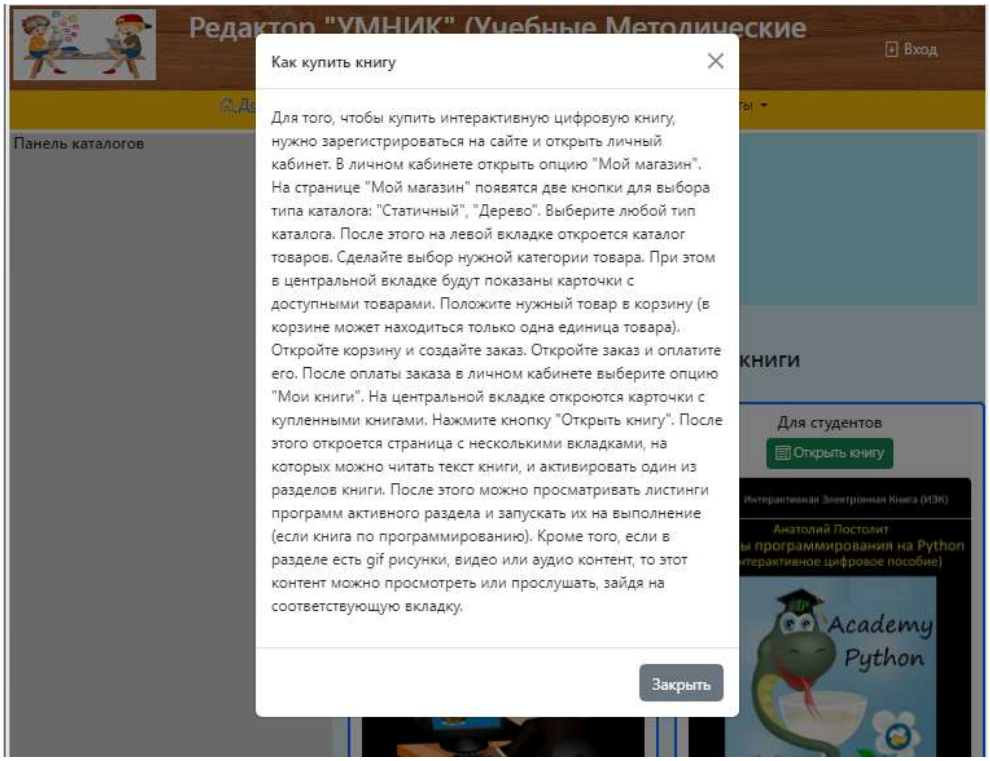


Рис. П.8. Инструкция с порядком приобретения платной интерактивной цифровой книги

Теперь познакомимся с содержанием вкладок интерактивной цифровой книги «Мастер веб-разработки на Python и Django».

Просмотр вкладок цифровой книги

Нажмите в карточке книги на кнопку **Открыть веб-книгу** — откроется страница с набором ее вкладок (рис. П.9).

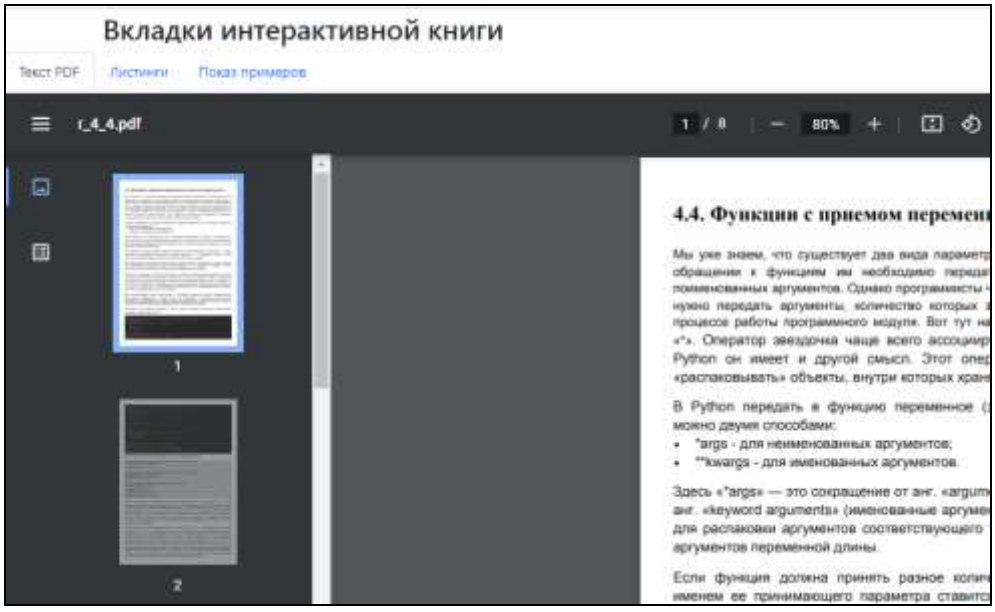


Рис. П.9. Вкладки интерактивной цифровой книги «Мастер веб-разработки на Python и Django»

Для этой книги будут показаны три вкладки:

- **Текст PDF** — текстовое содержание книги в виде PDF-файла;
- **Листинги** — листинги примеров программных модулей;
- **Показ примеров** — запуск примеров программных модулей книги на выполнение.

Вкладка *Текст PDF*

Вкладка **Текст PDF** (см. рис. П.9) имеет титульную строку и две панели: левую боковую и центральную. В левом углу титульной строки имеется кнопка в виде трех полосок, позволяющая скрыть/показать левую боковую панель, на которой находятся миниатюры страниц книги и две кнопки: верхняя кнопка дает возможность отобразить миниатюры страниц, а нижняя — показывает оглавление книги. Таким образом, левая боковая панель служит для быстрого выбора нужной страницы книги — либо из миниатюр страниц, либо из оглавления. Сами страницы с текстом книги отображаются в центральной панели.

В центре титульной строки имеются две кнопки: «+» и «–». Кнопка со знаком «+» позволяет увеличить масштаб отображаемой страницы, а кнопка со знаком «–» его соответственно уменьшить. Правее находятся еще две кнопки: первая — позволяет раскрыть страницу на всю ширину вкладки и свернуть ее до минимального размера, вторая — поворачивает страницу на угол 90 градусов.

В правом углу вкладки имеются еще три кнопки: скачать файл с книгой, вывести текст книги на печать, выполнить другие действия.

Левая и центральная панель имеют полосы прокрутки, которые позволяют быстро прокручивать страницы текста.

В левой боковой панели нижняя кнопка раскрывает оглавление книги. При выборе раздела оглавления в центральной странице стразу открывается первая страница выбранного раздела (рис. П.10).

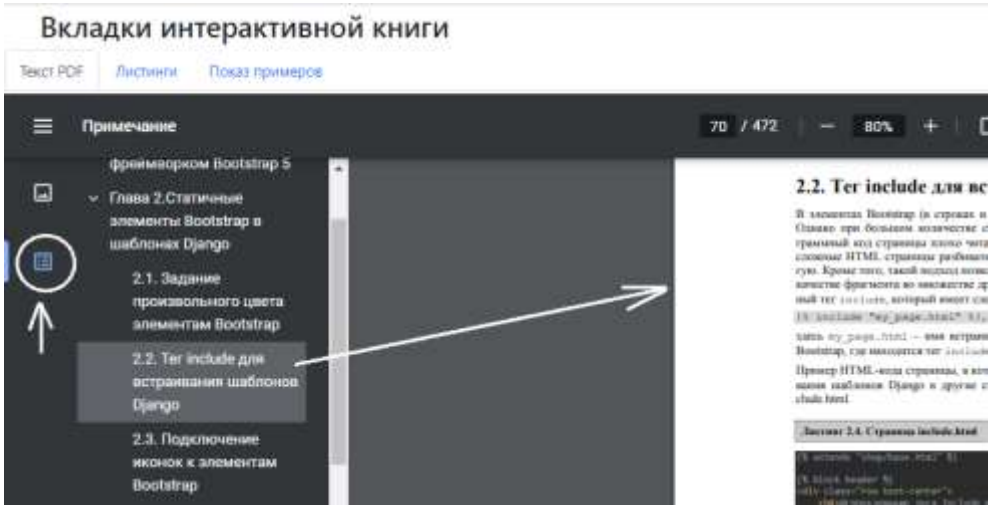


Рис. П.10. Навигация по страницам книги с использованием оглавления в левой боковой панели

Вкладка *Листинги*

При открытии вкладки **Листинги** активируется раздел текущей страницы с листингами программ в разрезе глав интерактивной цифровой книги. Каждая глава представлена в виде навигационной кнопки (рис. П.11).

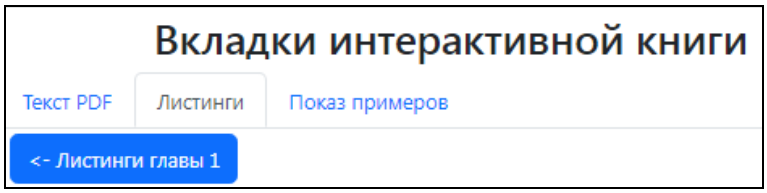


Рис. П.11. Навигационная кнопка для открытия листингов первой главы интерактивной цифровой книги

По нажатию на такую навигационную кнопку всплывает левая навигационная панель со списком листингов программ выбранной главы книги (рис. П.12).

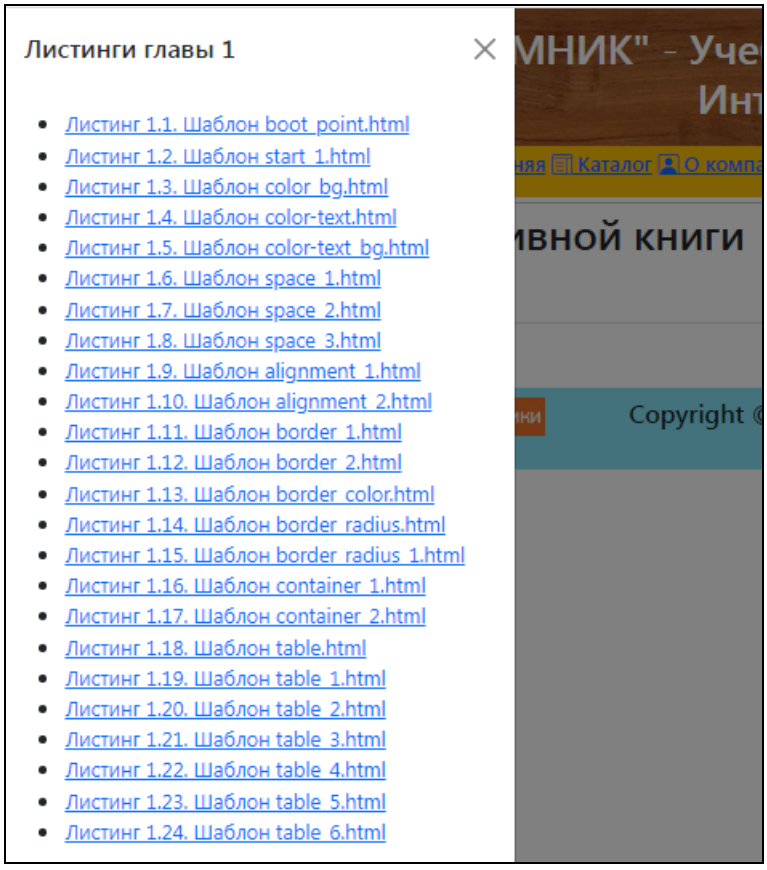


Рис. П.12. Выплывающая навигационная панель со списком листингов главы интерактивной цифровой книги

Наименование каждого листинга представлено в этой панели виде ссылки, при переходе по которой в новой вкладке интернет-браузера отображается листинг программы (рис. П.13).

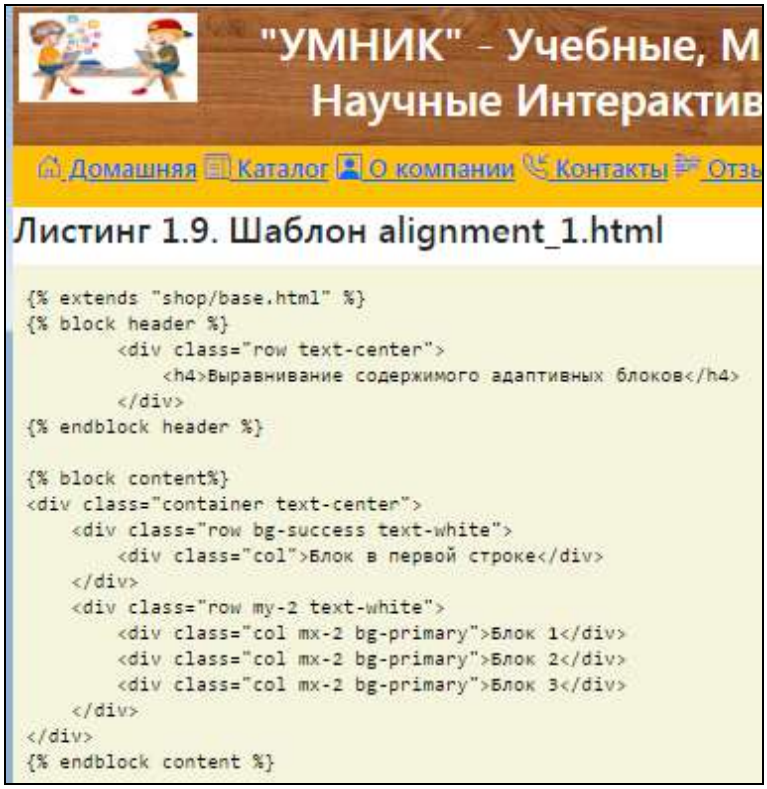


Рис. П.13. Вкладка интернет-браузера с листингом программы

Пользователь может выделить мышью весь код программы или его фрагмент, скопировать в буфер и сохранить на своем компьютере (рис. П.14).

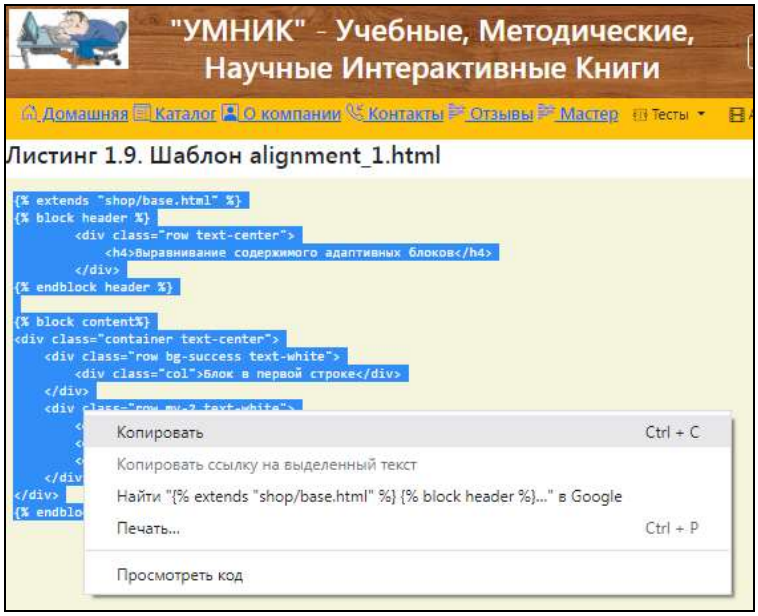


Рис. П.14. Копирование листинга программы

Как уже подчеркивалось ранее, программные коды любых примеров интерактивной цифровой книги пользователи могут использовать в своих проектах.

Вкладка *Показ примеров*

При открытии вкладки *Показ примеров* отображается раздел текущей страницы для активации выполнения примеров в разрезе глав интерактивной цифровой книги. Каждая глава представлена в виде навигационной кнопки (рис. П.15).

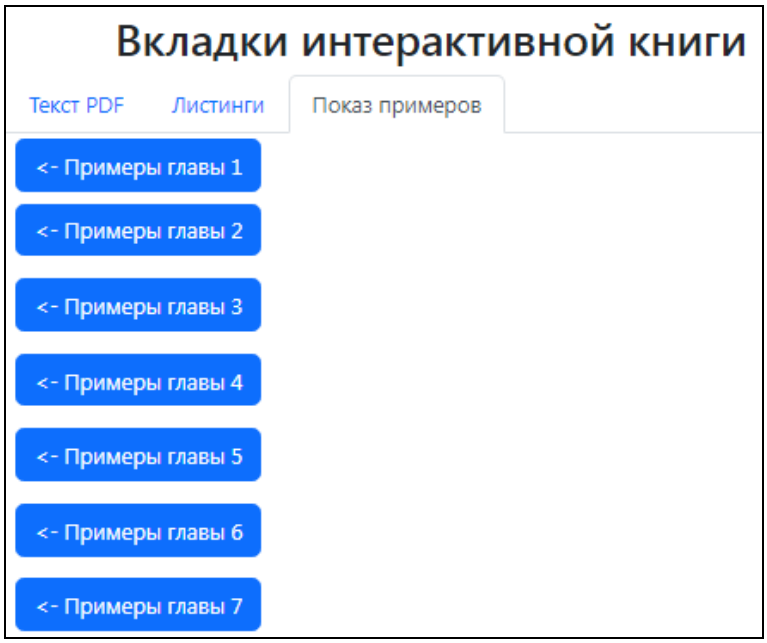


Рис. П.15. Навигационные кнопки для открытия примеров глав интерактивной цифровой книги для их выполнения

По нажатию на таукую навигационную кнопку всплывает левая навигационная панель со списком примеров выполняемых программ выбранной главы книги (рис. П.16).

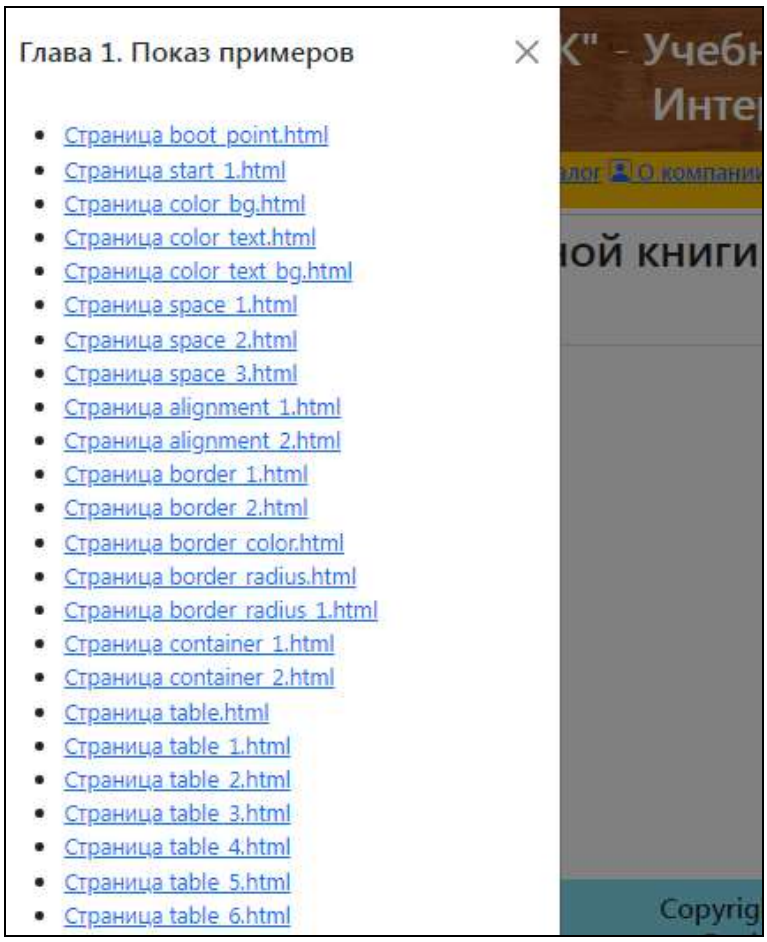


Рис. П.16. Список примеров выполняемых программ выбранной главы книги

Наименование каждого примера представлено в этой панели виде ссылки, при переходе по которой в новой вкладке интернет-браузера отображаются результаты работы программных модулей выбранного примера (рис. П.17).



Рис. П.17. Результаты работы программных модулей выбранного примера

Таким образом, читатель может не только познакомиться с теоретическими материалами книги, изучить программный код и скачать его на свой компьютер, но и выполнить любой пример любой главы книги, и посмотреть на результаты работы изучаемых программных модулей.

Приложение 2. Регистрация интернет-магазина на сайте платформы Robokassa

Для того чтобы на сайте интернет-магазина реализовать оплату по банковским картам через платежную платформу Robokassa, сначала нужно зарегистрироваться на сайте **https://robokassa.com**.

В процессе регистрации вам надо будет загрузить на платежную платформу ряд обязательных документов. После прохождения регистрации для вас будет создан личный кабинет, и только тогда вы сможете приступить к регистрации интернет-магазина на сайте Robokassa.

Когда клиенты вашего магазина будут совершать интернет-покупки, то деньги с их банковских карт станут поступать на счет вашего магазина, зарегистрированого на платежной платформе. И уже с этой платформы вы будете переводить полученные денежные суммы на свой личный счет или счет вашей фирмы. Естественно, для этого продавец (магазин) должен иметь статус ООО, ИП или самозанятого.

Итак, приступим к регистрации интернет-магазина. Для этого заходим в личный кабинет и выбираем опцию **Мои магазины** (рис. П.2.1).

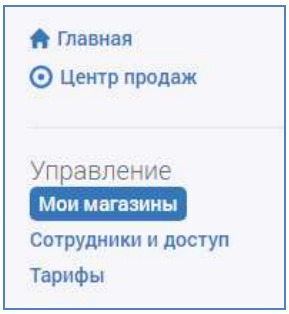


Рис. П.2.1. Опция **Мои магазины** в меню личного кабинета платежной платформы Robokassa

На открывшейся странице со списком магазинов (рис. П.2.2) нажимаем кнопку **Добавить новый** и в открывшейся форме (рис. П.2.3) формируем карточку магазина, заполняя следующие поля с его параметрами:

- наименование магазина;
- адрес для получения корреспонденции;
- интернет-адрес (URL) главной страницы сайта вашего магазина;
- счет, на который нужно будет выводить полученные деньги и способ вывода средств;
- ответственное лицо, которое отвечает за техническую поддержку сайта и его электронный адрес;
- наименование раздела, в котором можно будет найти ваш магазин;
- описание продаваемых товаров или выполняемых услуг.

Заполненную карточку магазина сохраняем нажатием на кнопку **Сохранить**.

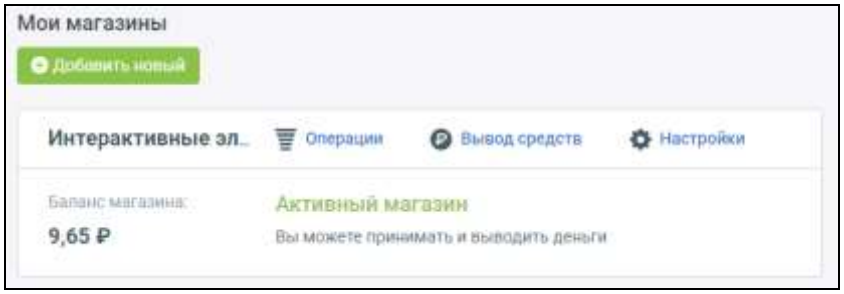


Рис. П.2.2. Страница со списком магазинов личного кабинета платежной платформы Robokassa

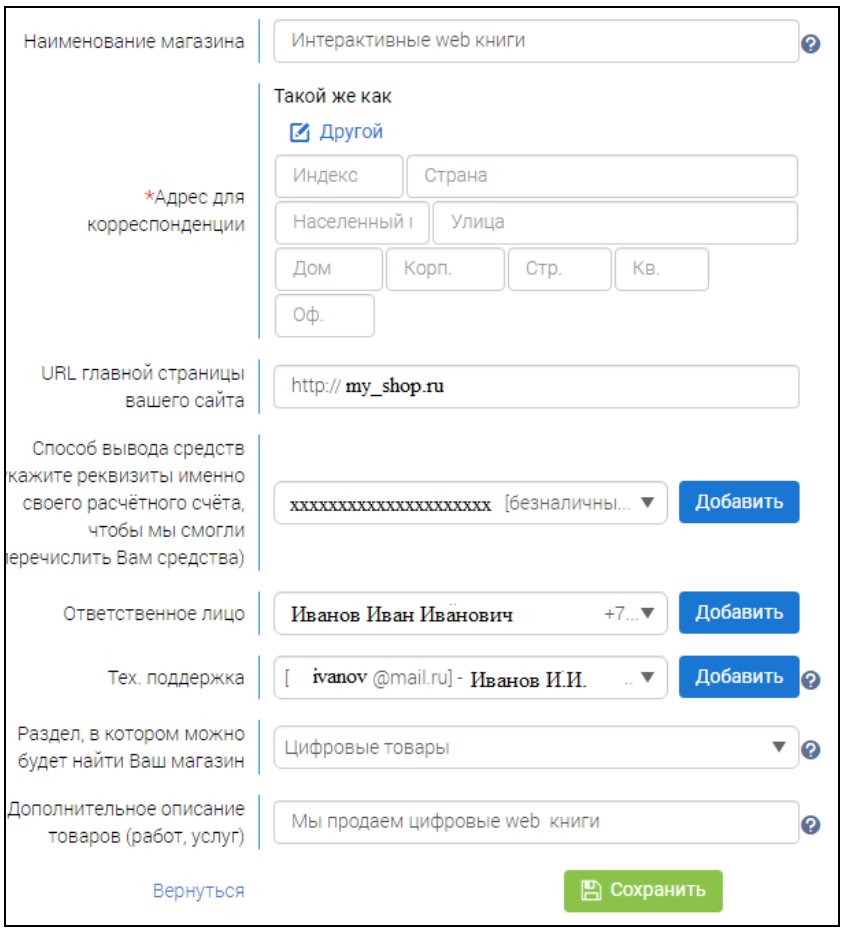


Рис. П.2.3. Форма для заполнения карточки интернет-магазина

После этого становится доступна страница с техническими настройками магазина. Здесь генерируем пароли и адреса перехода на страницы с успешной и не успешной оплатой (рис. П.2.4).

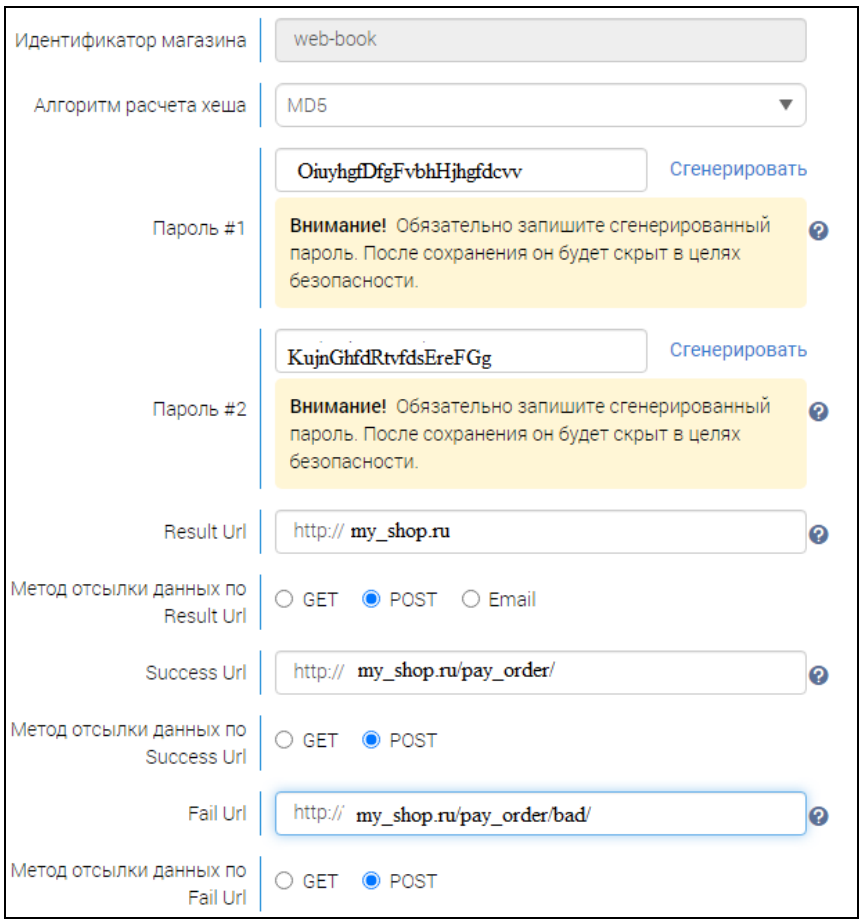


Рис. П.2.4. Страница с техническими настройками магазина

Сгенерированные пароли записываем на бумажный или иной носитель и сохраняем в защищенном месте (после сохранения страницы пароли перестанут отображаться):

```
Параметры магазина:
Идентификатор – web-book
Пароли реальной оплаты:
Пароль 1 – BcvfTYMJHfgderTREJhgD
Пароль 2 – LLkuiGFREBfGfdvCFDEd

Пароли тестовой оплаты:
Пароль 1 – YtghvdesDFderFgJuyGGVH
Пароль 2 – LyujhGFrDeSawQaHgTgf
```

> ***Пояснение.***
> Показанные здесь пароли — вымышленные. Для вас будут сгенерированы реальные пароли, и их нужно обязательно записать и сохранить, поскольку их понадобится указать в файле *settings.py* вашего проекта.

Далее вам будет предложено сгенерировать параметры тестовых платежей (рис. П.2.5) — сохраните на внешних носителях и эти пароли и нажмите на странице кнопку **Сохранить**.

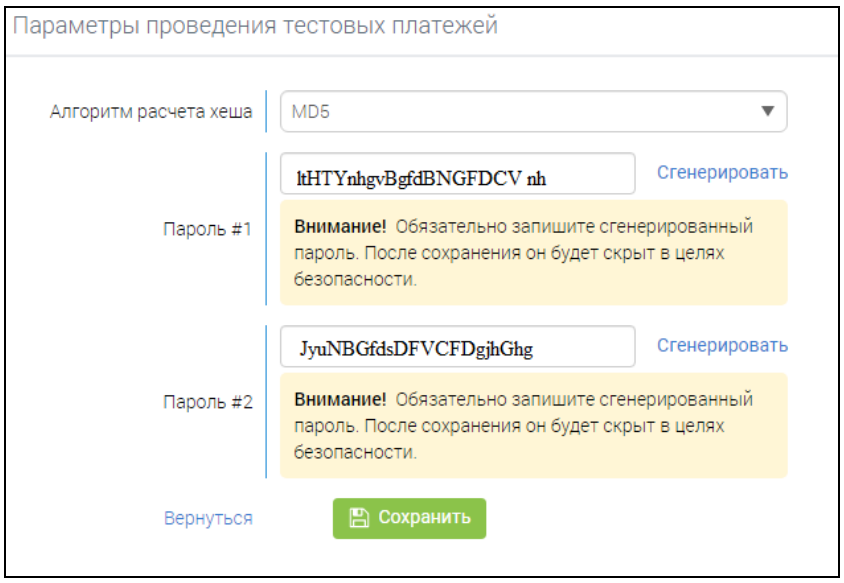


Рис. П.2.5. Страница с параметрами тестовых платежей

И вот теперь можно подавать заявку на активацию магазина — нажмите на странице, показанной на рис. П.2.6, кнопку **Запрос на активацию**.

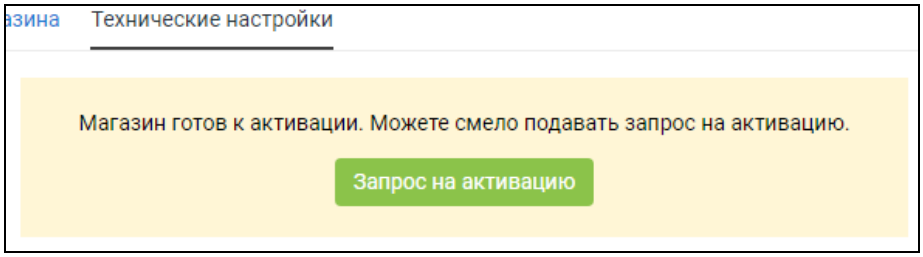


Рис. П.2.5. Вкладка с запросом на активацию магазина

Осталось только подождать ответа от менеджеров платежной платформы Robokassa. Они проверят поданные вами ранее (на этапе регистрации личного кабинета) документы, банковские и налоговые реквизиты вашей фирмы, ИП или самозанятого. Проверят они также сайт вашего интернет-магазина (он уже обязательно должен быть выставлен на публичный сервер) и наличие на его страницах обязательных реквизитов.

Если от менеджеров платежной системы поступят замечания, их следует устранить. Если замечаний не будет, то вам на электронную почту придет сообщение об успешной активации магазина.

Далее вы можете приступить к отладке работы модулей оплаты заказов банковскими картами на тестовых платежах, а при успешном завершении этого этапа — активировать выполнение реальных платежей.