Лекция 1 Веб-фреймворка Django

Цель лекции: ознакомление с новыми функциональными возможностями и архитектурой Django 4.

Новые функциональные возможности Django 4

Django 4 вводит набор новых функциональных возможностей, включающий несколько обратно несовместимых изменений, в то же время объявляя о скорой замене других функциональностей и устраняя старые. Поскольку релиз Django 4 основан на времени, в нем нет кардинальных изменений, приложения Django 3 легко мигрируются в версию 4.1. В отличие от релиза Django 3, куда впервые была включена поддержка интерфейса шлюза асинхронного сервера **ASGI**1, в Django 4.0 добавлено несколько характерных особенностей, таких как функциональные ограничения по уникальности для моделей Django, встроенная поддержка кеширования данных с использованием сервера хранилища Redis, новая стандартная реализация часового пояса с применением стандартного пакета Python zoneinfo, новый механизм хеширования паролей scrypt, шаблонно-ориентированная прорисовка форм, а также другие новые второстепенные функциональности. В Django 4.0 отменяется поддержка Python 3.6 и 3.7. В нем также прекращается поддержка PostgreSQL 9.6, Oracle 12.2 и Oracle 18c. В Django 4.1 вводятся асинхронные обработчики представлений на основе классов, асинхронный интерфейс ORM-преобразователя, новая валидация модельных ограничений и новые шаблоны прорисовки форм. В версии 4.1 прекращается поддержка PostgreSQL 10 и MariaDB 10.2.

Общий обзор веб-фреймворка Django

Django – это веб-фреймворк, состоящий из набора компонентов, которые решают распространенные задачи веб-разработки. Компоненты Django слабо сцеплены между собой, и поэтому ими можно управлять независимо, что помогает разделять обязанности разных слоев веб-фреймворка; слой базы данных ничего не знает о том, как данные отображаются на странице, система шаблонов ничего не знает о веб-запросах и т. д.

Django обеспечивает максимальную возможность реиспользования исходного кода, следуя принципу DRY (от англ. don’t repeat yourself -не повторяйся). Django также способствует быстрой разработке и позволяет использовать меньше исходного кода, применяя динамические возможности языка Python, такие как интроспекция.

Главные компоненты веб-фреймворка

Django подчиняется шаблону архитектурного дизайна **MTV** (**Model-Template-View**). Он немного похож на хорошо известный шаблон архитектурного дизайна **MVC** (**Model-View-Controller**)3, где Template

(Шаблон) действует как View (Представление), а сам веб-фреймворк действует как Controller (Контроллер).
Обязанности в шаблоне архитектурного дизайна MTV Django распределены следующим образом:

- **модель** – определяет логическую структуру данных и является обработчиком данных между базой данных и их представлением;
- **шаблон** – это слой представления. В Django используется система текстовых шаблонов, в которой хранится все, что браузер прорисовывает на страницах;
- **представление** – взаимодействует с базой данных через модель и передает данные в шаблон для их прорисовки и просмотра.

Сам веб-фреймворк выступает в качестве контроллера. Он отправляет запрос в надлежащее представление в соответствии с конфигурацией URL-адреса. При разработке любого проекта Django вы всегда будете работать с моделями, представлениями, шаблонами и URL-адресами. В этой главе вы узнаете, как они сочетаются друг с другом.

Архитектура Django

На рис. 1.1 показано, как Django обрабатывает запросы и как различные главные компоненты Django – модели, шаблоны, URL-адреса и представления– управляют циклом запроса/ответа.

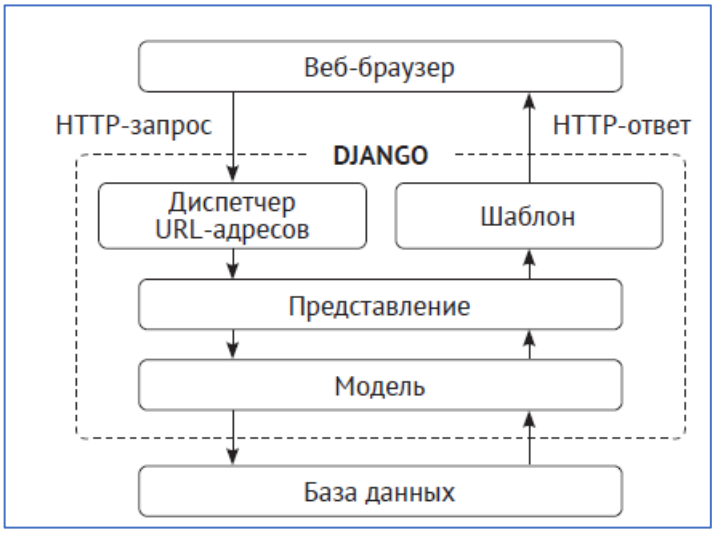

Рис. 1.1. Архитектура Django

Вот как Django оперирует HTTP-запросами и генерирует ответы:
1. Веб-браузер запрашивает страницу по ее URL-адресу, и веб-сервер передает HTTP-запрос веб-фреймворку Django.
2. Django просматривает свои сконфигурированные шаблоны URL-адресов и останавливается на первом, который совпадает с запрошенным URL-адресом.
3. Django исполняет представление, соответствующее совпавшему шабло-

ну URL-адреса.
4. Представление потенциально использует модели данных, чтобы извлекать информацию из базы данных.
5. Модели данных обеспечивают определение данных и их поведение. Они используются для запроса к базе данных.
6. Представление прорисовывает (согласно документации Django под рендерингом понимается взятие промежуточного представления шаблона вместе с контекстом и его превращение в итоговый поток байтов, который может быть передан клиенту (например, браузеру). На техническом языке это интерполяция (т. е. заполнение) шаблона контекстными данными и возврат результирующего строкового литерала, или трансляция данных в другой формат. В переводе при изложении тем, связанных с архитектурой MVP и шаблонами, используется термин «прорисовка» шаблона, а при обсуждении темы API – термин «трансляция») шаблон (обычно с использованием HTML), чтобы отображать данные на странице, и возвращает их вместе с HTTP-ответом).
В составе Django также имеются перехватчики (син. зацепки, хуки), вызываемые внутри процесса запроса/ответа, которые называются промежуточными программными компонентами (англ. middleware). Промежуточные программные компоненты были намеренно исключены из этой диаграммы ради простоты.

Проекты и приложения

На протяжении всей этой книги вы снова и снова будете сталкиваться с терминами «**проект**» и «**приложение**». В Django проектом считается установленный веб-фреймворк Django с несколькими настроечными параметрами.
Приложение – это группа моделей, шаблонов, URL-адресов и представлений. Приложения взаимодействуют с веб-фреймворком с целью обеспечения определенных функциональностей, и их можно реиспользовать в разных проектах. Проект можно трактовать как свой собственный веб-сайт, содержащий несколько приложений, таких как блог, вики или форум, который другие проекты Django тоже могут использовать. На рис. 1.3 показана структура проекта Django.

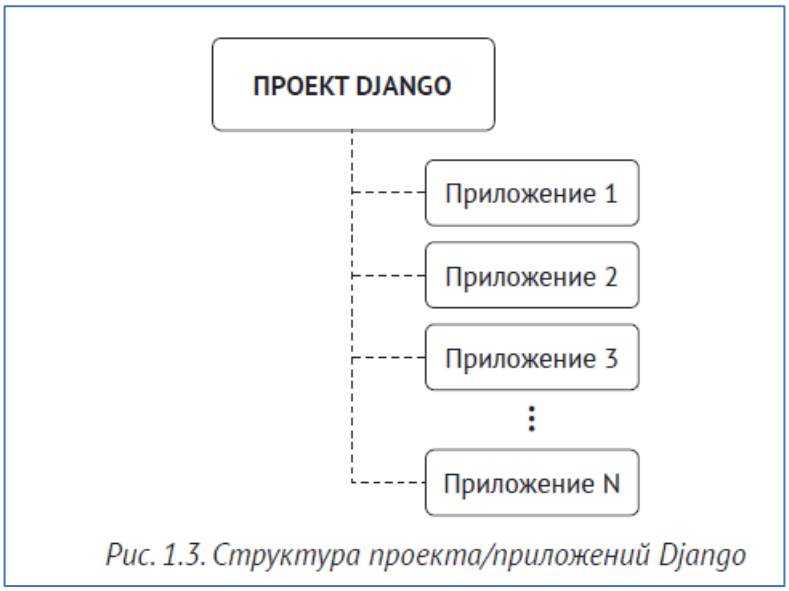

Рис. 1.3. Структура проекта/приложений Django