HIGH TECH

ДЖЕФФ ФОРСЬЕ
ПОЛ БИССЕКС
УЭСЛИ ЧАН

Django

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ НА PYTHON

Символ®

Python Web Development with Django

Jeff Forcier,
Paul Bissex, Wesley Chun

Addison-Wesley

HIGH TECH

Django

Разработка веб-приложений на Python

Джефф Форсье,
Пол Биссекс, Уэсли Чан

Санкт-Петербург — Москва
2009

Серия «High tech»

Джефф Форсье, Пол Биссекс, Уэсли Чан

Django. Разработка веб-приложений на Python

Перевод А. Киселева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Редактор	*Ю. Бочина*
Корректор	*С. Минин*
Верстка	*Д. Орлова*

Форсье Дж., Биссекс П., Чан У.

Django. Разработка веб-приложений на Python. – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 456 с., ил.

ISBN: 978-5-93286-167-7

На основе простой и надежной платформы Django на Python можно создавать мощные веб-решения всего из нескольких строк кода. Авторы, опытные разработчики, описывают все приемы, инструменты и концепции, которые необходимо знать, чтобы оптимально использовать Django 1.0, включая все основные особенности новой версии.

Это полное руководство начинается с введения в Python, затем подробно обсуждаются основные компоненты Django (модели, представления и шаблоны) и порядок организации взаимодействия между ними. Описываются методы разработки конкретных приложений: блог, фотогалерея, система управления содержимым, инструмент публикации фрагментов кода с подсветкой синтаксиса. После этого рассматриваются более сложные темы: расширение системы шаблонов, синдицирование, настройка приложения администрирования и тестирование веб-приложений.

Авторы раскрывают разработчику секреты Django, давая подробные разъяснения и предоставляя большое количество примеров программного кода, сопровождая их построчным описанием и иллюстрациями.

ISBN: 978-5-93286-167-7
ISBN: 978-0-13-235613-8 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 02.06.2009. Формат 70x100 1/16 . Печать офсетная.
Объем 28,5 печ. л. Тираж 1000 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Брайану Левайну (Brian Levine), открывшему
мне Python – маленький шаг, имевший большое значение.
Моим родителям – за то, что позволили мне монополизировать
домашний компьютер, пока я рос.
И моей супруге, за ее любовь, поддержку и понимание.

Джефф Форсье

Моему покойному отцу Генри, научившему меня работать;
моей матери Гленде, научившей меня писать;
и моей жене Кетлин, озарившей мою жизнь.

Пол Биссекс

Моим замечательным детям Леанне Син-Йи и Дейлину Син-Жи,
которые вынудили меня выработать способность находиться
во всех местах одновременно, чтобы уследить за ними, и которые
чудесным образом способны перемещать меня назад во времени,
напоминая мне мое детство с его удивительными чудесами.

Уэсли Чан

Оглавление

Предисловие

Добро пожаловать в Django!

Поздравляем вас и добро пожаловать в Django! Мы рады, что вы присоединились к нашему путешествию. Вы откроете для себя мощную платформу разработки веб-приложений, которая позволит вам быстро выполнять свою работу, – от проектирования и разработки оригинального приложения до его обновления и расширения его возможностей без необходимости вносить существенные изменения в программный код.

Об этой книге

В магазинах уже можно найти несколько книг о Django, но наша книга отличается от них тем, что основное внимание в ней уделяется трем областям: изучению основ Django, различных примеров приложений и дополнительных свойств Django. Наша цель состоит в том, чтобы в этой книге настолько полно охватить предмет обсуждения, чтобы вы нашли ее полезной независимо от уровня своей подготовленности и получили полное представление о платформе и ее возможностях.

Путеводитель по главам

На рис. 1 вы увидите рекомендуемые стартовые точки начала чтения книги в зависимости от вашего уровня знания языка программирования Python и платформы Django. Безусловно, мы рекомендуем прочитать ее от корки до корки, и тем не менее данная диаграмма поможет вам, если вы ограничены во времени. В любой момент независимо от уровня подготовки вы можете перейти к изучению приложений, потому что чтение и изучение программного кода – один из лучших способов обучения. Ниже приводится краткое содержание по главам, чтобы помочь вам выяснить, что следует прочитать.

Часть I «Введение»

Часть I содержит базовые сведения, необходимые для тех, кто не знаком еще с Django и/или Python, хотя даже опытным читателям мы рекомендуем в чтении начать с главы 3 «Начало».

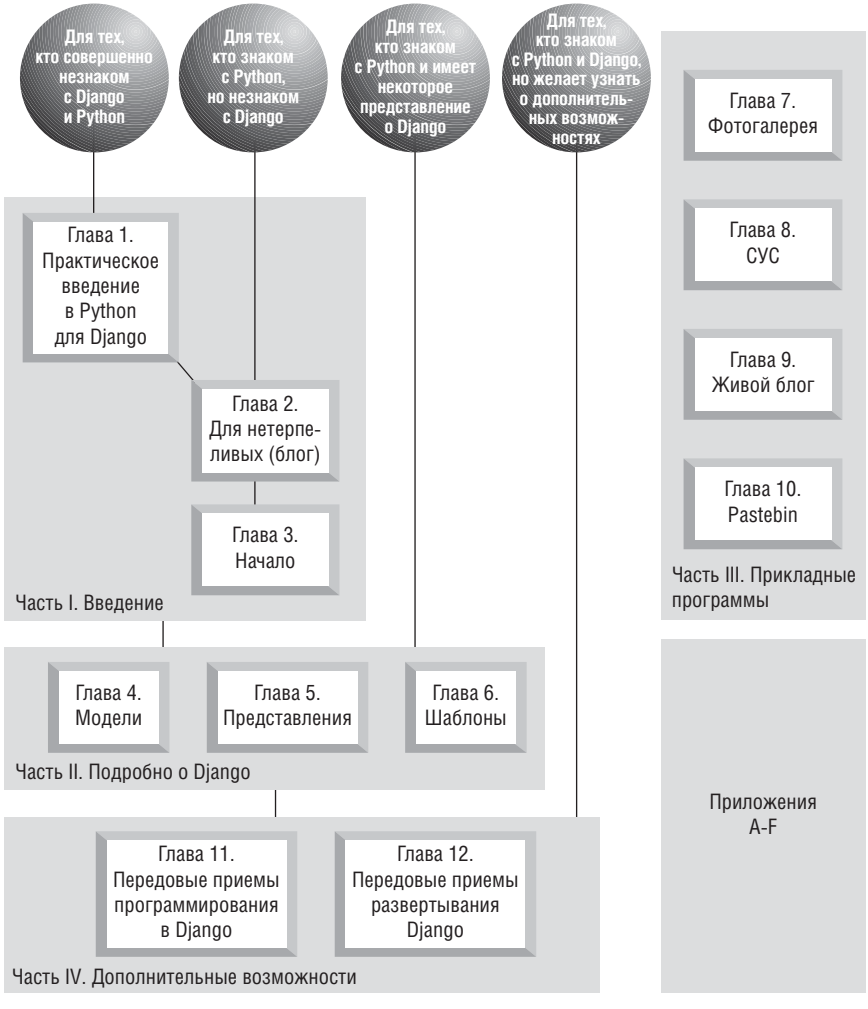

Рис. 1. *Рекомендуемые отправные точки для чтения в зависимости от уровня подготовки*

Глава 1 «Практическое введение в Python для Django»

Эта вводная глава предназначена для тех, кто не знаком с языком программирования Python. В этой главе демонстрируются не только синтаксические особенности, но также углубленный взгляд на модель распределения памяти в Python и типы данных, особенно на те, что наиболее часто используются в Django.

Глава 2 «Django для нетерпеливых: создание блога»

Эта глава предназначена для тех, кто, перепрыгнув через введение в язык программирования Python, стремится сразу же погрузиться в изучение приложения, которое можно создать на платформе Django за 15–20 минут. Эта глава содержит отличный обзор возможностей платформы Django.

Глава 3 «Начало»

Эта глава предназначена для более методичных читателей и служит введением в основы разработки веб-приложений (в ней содержатся полезные сведения как для начинающих, так и для опытных программистов). После введения всех формальностей здесь описывается место и роль каждой концепции в мире Django, а также философия платформы и ее отличия от других платформ.

Часть II «Подробно о Django»

Вторая часть описывает базовые компоненты платформы, закладывая фундамент для примеров приложений в третьей части «Приложения Django в примерах»

Глава 4 «Определение и использование моделей»

В главе рассказывается, как определять и использовать модели данных, включая основную объектно-реляционную проекцию (Object-Relational Mapping, ORM) Django, от простых полей до сложных схем отношений.

Глава 5 «Адреса URL, механизмы HTTP и представления»

В этой главе подробно описывается, как платформа Django работает с адресами URL и остальными особенностями протокола HTTP, включая промежуточную обработку, а также как использовать универсальные представления Django и как создавать собственные представления или модернизировать существующие.

Глава 6 «Шаблоны и обработка форм»

Глава охватывает последнюю значительную часть платформы. Здесь исследуется язык шаблонов, используемый в Django, и механизмы обработки форм, а также рассказывается, как отобразить данные перед пользователями и как получать данные от них.

Часть III «Приложения Django в примерах»

В третьей части будут созданы четыре различных приложения, чтобы обозначить различные аспекты или компоненты, используемые в процессе разработки приложений на платформе Django. В каждом из примеров будут вводиться новые идеи и расширяться концепции, представленные в частях I и II.

Глава 7 «Фотогалерея»

В главе демонстрируется применение правила «не повторяй самого себя» к структуре адресов URL, а также создание поля с эскизами изображений для простого приложения фотогалереи.

Глава 8 «Система управления содержимым»

В главе рассказывается о двух взаимосвязанных подходах к созданию СУС и подобных им систем, а также охватывает вопросы использования некоторых приложений, «пожертвованных» проекту Django.

Глава 9 «Живой блог»

В главе демонстрируется процесс создания «живого блога» – сайта, написанного с использованием передовых приемов программирования на JavaScript, который может служить примером применения технологии AJAX в проектах на платформе Django и показывает, насколько просто можно использовать любые инструменты поддержки AJAX.

Глава 10 «Pastebin»

В главе демонстрируется широта возможностей универсальных представлений Django на примере веб-приложения pastebin, при создании которого практически не потребовалась наша собственная реализация логики.

Часть IV «Дополнительные возможности и особенности Django»

Часть IV представляет собой коллекцию более сложных тем, начиная от настройки приложения администратора Django и заканчивая созданием сценариев командной строки, способных взаимодействовать с приложениями на платформе Django.

Глава 11 «Передовые приемы программирования в Django»

Глава затрагивает темы, связанные с процессом разработки программного кода приложений, такие как генерирование лент RSS, расширение языка шаблонов и улучшение приложения администрирования Django.

Глава 12 «Передовые приемы развертывания Django»

В главе рассматривается множество хитростей, связанных с развертыванием приложений на платформе Django, и работа с приложением из-за пределов Django, например из сценариев командной строки, заданий планировщика cron, а также при тестировании и импортировании данных.

Часть V «Приложения»

Эта часть восполняет недостающую информацию и рассматривает остальные темы, которые не были выделены в отдельные главы. Здесь изучаются основы командной строки UNIX, проблемы установки

и стратегии развертывания Django, инструменты разработки и многое другое.

Приложение A «Основы командной строки»

В приложении рассматриваются основы командной строки UNIX для тех, кто раньше не сталкивался с ней. Поверьте – это очень полезно!

Приложение B «Установка и запуск Django»

В приложении демонстрируется процесс установки компонентов, необходимых для запуска Django, включая особенности установки при наличии различных баз данных и веб-серверов, а также даются некоторые советы по развертыванию.

Приложение C «Инструменты разработки для платформы Django»

В приложении перечисляются некоторые инструменты разработки, с которыми вы можете уже быть знакомы, включая управление версиями исходных текстов, текстовые редакторы и многое другое.

Приложение D «Поиск, оценка и использование приложений на платформе Django»

Хороший разработчик пишет свой программный код, но отличный разработчик использует повторно уже написанный кем-то ранее программный код! В приложении D мы поделимся некоторыми советами о том, где и как искать приложения Django для повторного использования.

Приложение E «Django и Google App Engine»

В приложении E рассматривается, как в приложениях на платформе Django можно использовать преимущества нового механизма Google App Engine и демонстрируется, как запускать приложения на платформе Django под управлением платформы App Engine.

Приложение F «Участие в проекте Django»

В приложении вы узнаете, как принять участие в развитии проекта Django и как стать членом сообщества.

Типографские соглашения

На протяжении всей книги **жирным шрифтом** будут выделяться новые или важные термины, *курсивный шрифт* будет использоваться, чтобы обратить ваше внимание и для обозначения адресов URL, а `моноширинный шрифт` – для выделения программного кода на языке Python, например имен переменных и функций. Многострочные блоки программного кода или примеры команд будут оформлены моноширинным шрифтом в виде блоков:

```
>>> print "This is Python!"
This is Python!
```

В процессе создания этой книги и примеров приложений мы использовали три основные платформы – Mac OS X, Linux и Windows. Кроме того, мы использовали все основные броузеры (хотя не все они были представлены на снимках экрана), а именно: Firefox, Safari, Opera и Internet Explorer.

Ресурсы для книги

Обратиться к коллективу авторов можно по электронной почте *authors@withdjango.com*. На нашем веб-сайте, *http://withdjango.com*, содержится значительный объем вспомогательных материалов, ссылки на которые часто встречаются в книге.

Благодарности

Мое имя первым стоит в списке авторов, но эта книга не увидела бы свет без усилий других авторов. Пол и Уэсли – джентльмены и специалисты высочайшего класса, и совместная работа с ними стала для меня удивительным опытом.

Джентльменами и специалистами можно назвать всех разработчиков, составляющих ядро команды Django. Первоначальные авторы Django – Адриан Холовати (Adrian Holovaty), Якоб Каплан-Мосс (Jacob Kaplan-Moss), Саймон Уиллисон (Simon Willison) и Уилсон Майнер (Wilson Miner) – заложили (и продолжают развивать) удивительный фундамент, который был дополнен благодаря усилиям Малкольма Трединника (Malcolm Tredinnick), Джорджа Бауэра (Georg Bauer), Люка Планта (Luke Plant), Рассела Кейт-Маги (Russell Keith-Magee) и Роберта Уитамса (Robert Wittams). Каждый из этих парней поражает меня, а поразить меня совсем непросто.

Мне также хотелось бы поблагодарить двух сотрудников «Djangonauts» и ветеранов IRC – Кевина Менарда (Kevin Menard) и Джеймса Беннета (James Bennett), а также членов группы NYCDjango, где собрались удивительные и талантливые члены сообщества Django.

Наконец, огромное спасибо сотрудникам издательства Pearson, включая наших редакторов и технических рецензентов (Уэсли упомянет этих людей чуть ниже!), и особое спасибо техническим редакторам, чье внимание к деталям трудно переоценить.

Джефф Форсье (Jeff Forcier)
Нью-Йорк (штат Нью-Йорк)
Август 2008

Спасибо сообществам людей, сплотившихся вокруг Django, Python и других платформ, предназначенных для разработки веб-приложений и распространяемых с открытыми исходными текстами. Благодаря усилиям тысяч разработчиков и пользователей создаются мощные пакеты свободно распространяемого программного обеспечения.

Мои соавторы стали настоящей находкой и источником основных знаний и умений, а также проявили удивительную преданность делу. Несмотря на тот факт, что мы живем в разных концах континента, мне посчастливилось повстречать Джеффа и Уэса.

Хочу выразить благодарность группе разработчиков из Western Massachusetts Developers Group за интересные и жаркие дискуссии и огромный энтузиазм, проявленный к проекту книги.

Спасибо Джорджу Дж. Роса III (George J. Rosa III), президенту института фотографии Hallmark Institute of Photography, за то, что поддерживал меня и доверил мне выбор инструментов для работы, включая, конечно, и Django.

Летом 2008 года, после серьезной автомобильной аварии, я получил удивительный всплеск внимания и поддержки от моей семьи, друзей и сообщества. Для меня огромное значение имело все – и добрые пожелания, и оплата счетов, и деньги, и еда. В этих словах вы сами узнаете себя, и я еще раз благодарю вас.

И моей замечательной супруге Кетлин – спасибо тебе за внимание и понимание, поддержку и любовь.

Пол Биссекс (Paul Bissex)

Нортхемптон (штат Массачусетс)

Сентябрь 2008

Работа над второй моей книгой стала бесценным опытом. Я хотел бы поприветствовать двух моих соавторов, работа с которыми доставила мне огромное удовольствие. Они способны любого, имеющего некоторый опыт работы с языком программирования Python, привести в мир Django. Я счастлив тем, что сумел внести свой вклад в эту замечательную книгу о Django, и надеюсь, что в будущем мне доведется еще работать с ними. Было чрезвычайно приятно работать над этой книгой, как если бы это был открытый проект, используя те же инструменты, которые применяют разработчики каждый день для разработки программного обеспечения, изменяющего мир.

Я благодарен Дебре Уильямс Коли (Debra Williams Cauley) за помощь в управлении всем процессом с самого начала работы над этим проектом. Нас преследовали многочисленные изменения в составе сотрудников, но она позволила нам быть сосредоточенными только на рукописи. Хотя не было никаких гарантий, что будет создана книга о Django, которая будет пользоваться спросом, но она верила в наше стремление написать «правильную» книгу, которая будет востребована всем сообществом. Спасибо всем нашим техническим рецензентам – Майклу Торстону (Michael Thurston) (редактор-консультант по аудитории), Джо Блейлоку (Joe Blaylock) и Антонио Кангиано (Antonio Cangiano), а также всем, кто присылал свои отзывы Рафу Катсу (Rough Cuts), что позволило улучшить эту книгу по сравнению с первоначальным вариантом. Я также хотел бы выразить благодарность Мэтту Брауну (Matt Brown), лидеру проекта «Django Helper for Google App Engine», за его помощь в рецензировании приложения E, а также Эрику Уолстаду (Eric Walstad) и Эрику Эвенсону (Eric Evenson) за их заключительное рецензирование всей книги и комментарии.

Наконец, без поддержки наших семей эта книга была бы просто невозможна.

Уэсли Чан (Wesley Chun)

Кремниевая долина (штат Калифорния)

Август 2008

Введение

Если вы веб-разработчик, то есть программист, занимающийся созданием веб-сайтов, платформа Django легко может изменить вашу жизнь, как она изменила нашу.

Любой, кто обладает даже незначительным опытом разработки динамических веб-сайтов, знает, насколько сложно изобретать одно и то же снова и снова. Необходимо создать схему базы данных. Необходимо реализовать запись и извлечение данных из базы. Необходимо предусмотреть анализ адресов URL. Необходимо фильтровать данные, вводимые человеком. Необходимо создать инструменты редактирования информационного наполнения. Необходимо постоянно помнить о безопасности и удобстве использования. И так далее.

Как появились веб-платформы

В некоторый момент вы начинаете понимать, насколько это расточительно – тратить время на повторную реализацию одних и тех же особенностей в каждом новом проекте, и решаете создать свои собственные библиотеки с чистого листа или, что более вероятно, извлечь эти библиотеки из последнего и наиболее удачного проекта. После того как вы приступаете к работе над новым проектом, первое, что вы делаете – устанавливаете библиотеки. Благодаря этому вы экономите свои силы и время.

Однако начинают проявляться некоторые недостатки. Клиенты выражают желание получить функциональные возможности, отсутствующие в вашей библиотеке, поэтому вы добавляете в нее необходимый программный код. У разных клиентов возникают разные желания, в результате вы получаете несколько различных версий своей библиотеки, установленные на разных серверах. Сопровождение таких веб-приложений превращается в кошмар.

Поэтому, закаленный опытом, вы берете свою основную библиотеку и все лучшие дополнения из ваших проектов и объединяете их. Для большинства проектов вам уже не приходится изменять программный код своей библиотеки – вместо этого вы просто изменяете конфигура-

ционный файл. Ваша библиотека стала больше и сложнее, но при этом она обладает большими возможностями.

Примите наши поздравления – вы написали веб-платформу.

И пока вы (ваш коллектив, компания или ваши клиенты) продолжаете использовать библиотеку, вы несете ответственность за сохранение ее работоспособности. Не приведет ли к нарушениям в работе переход на новую версию операционной системы, веб-сервера или языка программирования? Обладает ли она достаточной гибкостью, чтобы в будущем в нее можно было вносить изменения без особых сложностей? Поддерживает ли она такие сложные, но нужные особенности, как управление сеансами, локализация и средства доступа к базам данных? А как насчет проверочных тестов?

Более простой путь

Вы взяли эту книгу в руки, потому что хотите отыскать более простой путь. Вам требуется мощная, гибкая, тщательно протестированная и первоклассная платформа для разработки веб-приложений, *но вам не хочется заниматься ее поддержкой*.

Вам необходим настоящий язык программирования – мощный, ясный, зрелый, хорошо документированный. Вам необходимо, чтобы для используемого языка программирования имелась обширная стандартная библиотека и широчайший выбор разнообразных и высококачественных пакетов сторонних разработчиков, начиная от пакетов создания файлов в формате CSV или круговых диаграмм и завершая пакетами реализации научных расчетов или обработки файлов изображений.

Вам необходима платформа, поддерживаемая энергичным, готовым прийти на помощь сообществом пользователей и разработчиков. Платформа, которая функционирует как хорошо отлаженная машина, но чьи компоненты слабо связаны между собой, благодаря чему вы легко можете менять их по мере необходимости.

Проще говоря, вам необходимы язык Python и платформа Django. Мы написали эту книгу, чтобы помочь вам изучить и начать пользоваться Django при решении реальных задач настолько легко, быстро и эффективно, насколько это возможно.

Мы уже не в Канзасе

Первоначально платформа Django была написана Адрианом Холовати (Adrian Holovaty) и Саймоном Уиллисоном (Simon Willison), работавшим в World Online – семейной веб-компании, находившейся в городе Лоуренс, штат Канзас. Она появилась из-за необходимости быстро

разрабатывать приложения баз данных, наполняемых содержимым новостей.

После того как платформа Django доказала свою состоятельность, в июле 2005 года она была выпущена как проект с открытыми исходными текстами – это было время, когда, по иронии судьбы, было широко распространено мнение, что на языке Python реализовано *слишком мало* веб-платформ, – и быстро получила мощную поддержку. В настоящее время эта платформа является одним из лидеров не только среди платформ для разработки веб-приложений на языке Python, но и среди всех веб-платформ.

Конечно, платформа Django по-прежнему интенсивно используется компанией World Online, и некоторые из основных разработчиков платформы продолжают работать в этой компании и ежедневно пользуются платформой. Но, так как Django является программным продуктом, распространяемым с открытыми исходными текстами, большое число компаний и организаций по всему миру выбирают и используют ее в своих больших и маленьких проектах. В число этих компаний входят:

- The Washington Post
- The Lawrence Journal-World
- Google
- EveryBlock
- Newsvine
- Curse Gaming
- Tabblo
- Pownce

Безусловно, существуют тысячи других сайтов, созданных на основе Django, названия которых пока не так широко известны. Но по мере развития Django неизбежно будет увеличиваться число популярных сайтов, работающих на ее основе, и мы надеемся, что ваш сайт будет одним из них.

Разрабатывать веб-приложения лучше с использованием Python и Django

Разработка веб-приложений – это не самое простое дело. Вам приходится бороться с несовместимостью броузеров, со злонамеренными ботами, с ограничениями полосы пропускания и сервера и общей архитектурой, трудно поддающейся тестированию.

Конечно, мы полагаем, что наша книга является отличным введением в основы Django, но при этом мы стараемся не оставлять без внимания

упомянутые сложности – 20 процентов работы может потребовать 80 процентов времени. Мы работали со многими разработчиками и помогли многим из них в решении проблем, связанных с применением платформы Django, и мы не забыли их вопросы и держали их в уме при работе над этой книгой.

Если бы мы не считали Django и Python отличными продуктами, мы не стали бы брать на себя труд писать целую книгу о них. И когда мы будем подходить к каким-либо ограничениям или подвохам, о которых вам следует знать, мы сообщим об этом. Наша цель состоит в том, чтобы помочь вам довести работу до результата.

I

Введение

1. Практическое введение в Python для Django
2. Django для нетерпеливых: создание блога
3. Начало

1

Практическое введение в Python для Django

Добро пожаловать в Django, а также, кроме того, в Python! Прежде чем перейти к Django, мы дадим краткий обзор языка, который является основой приложений, разрабатываемых на платформе Django. Знакомство с другими языками программирования высокого уровня (C/C++, Java, Perl, Ruby и т. д.) упростит усвоение материала этой главы.

Однако, если вы никогда раньше не занимались программированием, сам язык Python прекрасно подходит на роль первого языка. В конце главы приводится список книг, которые можно использовать для обучения программированию на языке Python. Тем, кто плохо знаком с программированием, мы рекомендуем обратиться сначала к этим книгам, а затем вернуться сюда – это поможет вам извлечь больше пользы из следующих разделов.

В этой главе будет представлен язык программирования Python, причем особое внимание будет уделено основам языка и приемам программирования, которые имеют отношение к разработке приложений на платформе Django. Для эффективной работы с Django вам нужно знать не только основы языка Python, но также внутреннее устройство и принцип его действия, поэтому, когда будут рассматриваться некоторые особенности или требования платформы Django, мы не оставим без внимания то, что происходит за кулисами. Те, кто плохо знаком с языком Python или с программированием вообще, извлекут больше пользы, если получат общие сведения о Python из других источников, перед тем как приступать к чтению этой главы, или будут обращаться к ним в процессе ее чтения – какой стиль изучения подходит лучше, решать вам.

Практические приемы владения Python и Django

Django представляет собой высокоуровневую платформу, которая позволяет создавать веб-приложения, написав всего несколько строк программного кода. Эта платформа отличается простотой и гибкостью, позволяя без труда создавать собственные решения. Платформа Django написана на языке Python, объектно-ориентированном языке программирования, который соединяет в себе мощь таких языков системного программирования, как C/C++ и Java, с непринужденностью и скоростью разработки языков сценариев, таких как Ruby и Visual Basic. Это дает пользователям возможность создавать приложения, способные решать самые разнообразные задачи.

В этой главе будет показаны некоторые практические приемы программирования на языке Python, которыми должен владеть любой разработчик, использующий платформу Django. Вместо того чтобы пытаться воссоздать универсальный учебник по языку Python, мы сосредоточимся на тех концепциях языка, которые «должны быть» в арсенале программиста, использующего платформу Django. Фактически в этой главе будут приводиться фрагменты программного кода из самой платформы Django.

Python 2.x и Python 3.x

Во время нашей работы над этой книгой начался переход с версии Python 2.x на новое поколение версий Python, начиная с версии 3.0. Семейство 3.x не гарантирует обратную совместимость с предыдущими версиями языка, поэтому вполне возможно, что программный код, написанный для версии 2.x не будет работать с интерпретатором версии 3.x. Однако команда разработчиков Python стремится сделать переход на использование новой версии как можно более безболезненным; они предоставят надежные инструменты переноса сценариев с версии 2.x на версию 3.x, да и сам переход будет выполняться достаточно продолжительное время, поэтому никто не останется брошенным.

Команда разработчиков Python не планирует выполнить переход на версию 3.0 прямо сейчас – как и в большинстве основополагающих проектов, такой переход может оказаться разрушительным и должен выполняться с большой осторожностью, поэтому об этом переходе мы будем говорить лишь вскользь. Наиболее вероятно, что сама платформа Django будет переведена на новую версию, только когда основная масса пользователей (и вы в том числе!) будет готова к этому.

Введение: интерактивный интерпретатор языка Python

Интерактивный интерпретатор – это один из самых мощных инструментов, используемых в процессе разработки программ на языке Python, обеспечивая возможность тестирования фрагментов, состоящих всего из нескольких строк программного кода, без необходимости создавать, редактировать, сохранять и запускать файлы с исходными текстами. Более того, интерактивная оболочка интерпретатора Python проверяет корректность вводимого программного кода и позволяет опробовать различные вещи на новом программном коде – например, проверять структуры данных или изменять ключевые значения, прежде чем добавить его в файл с исходными текстами.

Во время чтения этой главы мы рекомендуем запустить интерактивную оболочку интерпретатора Python, чтобы тут же опробовать фрагменты программного кода – большинство интегрированных сред разработки на языке Python легко запускаются из командной строки или из меню приложений операционной системы. Используя интерактивную оболочку, вы получите непосредственный контакт с интерпретатором и быстрее овладеете Python и Django. Опытные программисты, использующие Python, такие как авторы этой книги, по-прежнему продолжают использовать интерактивную оболочку Python каждый день, даже обладая десятилетним опытом!

На протяжении всей книги вам будут встречаться фрагменты программного кода, начинающиеся со строки приглашения к вводу: `>>>`. Эти примеры можно опробовать непосредственно в интерактивной оболочке. Выглядят они примерно так, как показано ниже:

```
>>> print 'Hello World!'
Hello World!
>>> 'Hello World!'
'Hello World!'
```

Инструкция `print` – ваш лучший друг. Она не только может использоваться в приложениях для вывода информации, но и представляет собой бесценное средство отладки. Хотя часто бывает возможно вывести значение переменной без явного использования инструкции `print`, как это только что было продемонстрировано, но вы должны знать, что при этом нередко выводимые результаты отличаются от тех, что выводит инструкция `print`. В данном примере, используя инструкцию `print`, мы требуем от интерпретатора вывести *содержимое* строки, которое, конечно, не включает кавычки. Данное конкретное отличие касается только строк, для чисел таких различий не наблюдается.

```
>>> 10
10
>>> print 10
10
```

Однако для составных объектов, к которым мы подойдем ниже, различия могут оказаться более существенными – это обусловлено тем, что язык Python дает нам в руки полный контроль над тем, как должен вести себя объект при выводе с использованием и без использования инструкции `print`.

Подробнее о переменных и циклах будет говориться ниже и, тем не менее, чтобы получить некоторое представление о языке, взгляните на следующий, немного более сложный, фрагмент программного кода, в котором присутствует цикл `for`:

```
>>> for word in ['capitalize', 'these', 'words']:
...     print word.upper()
...
CAPITALIZE
THESE
WORDS
>>> for i in range(0, 5):
...     print i
...
0
1
2
3
4
```

Использование интерактивной оболочки при работе с Django

Очень удобно использовать интерактивную оболочку интерпретатора для экспериментов с программным кодом приложений, созданных на платформе Django, или с фрагментами реализации самой платформы. Но, если просто запустить интерпретатор и попытаться импортировать модули Django, будет получено сообщение об отсутствии переменной окружения `DJANGO_SETTINGS_MODULE`. Платформа Django предоставляет команду `manage.py shell`, которая выполняет все необходимые настройки окружения и позволяет избежать этой проблемы.

Команда `manage.py shell` по умолчанию использует оболочку iPython, если она установлена. Если же при установленной оболочке iPython вы все-таки хотите использовать стандартную интерактивную оболочку интерпретатора Python, используйте команду `manage.py shell plain`. В наших примерах мы будем использовать интерпретатор по умолчанию, но вам настоятельно рекомендуем пользоваться оболочкой iPython.

Важной особенностью языка Python является отказ от использования фигурных скобок ({}) для выделения блоков программного кода. Вместо скобок используются отступы: внутри данного фрагмента имеются различные уровни отступов. Обычно величина отступа составляет четыре пробела (хотя вы можете использовать любое число пробелов или символов табуляции). Если у вас есть опыт работы с другими языками программирования, может потребоваться некоторое время, чтобы привыкнуть к этой особенности, однако спустя короткое время вы поймете, что она не так плоха, как кажется на первый взгляд.

Последнее замечание об интерпретаторе: как только вы научитесь пользоваться интерактивной оболочкой, вы должны познакомиться с похожим инструментом, который называется iPython. Для тех, кто уже хорошо знаком с интерактивной оболочкой Python, можем сказать, что iPython – это еще более мощный инструмент! Он предоставляет множество таких возможностей, как доступ к командной оболочке операционной системы, нумерация строк, автоматическое оформление отступов, история команд и многое другое. Подробнее об iPython можно узнать на сайте *http://ipython.scipy.org*. Этот инструмент не распространяется в составе Python, но его свободно можно получить в Интернете.

Основы Python

В этом разделе мы познакомимся с некоторыми аспектами языка Python. Мы поговорим о комментариях, переменных, операторах и базовых типах данных. В следующих нескольких разделах мы еще ближе познакомимся с основными типами данных. Большая часть программного кода на языке Python (в том числе и программный код Django) находится в текстовых файлах с расширением `.py` – это стандартный способ сообщить системе, что это файл с программным кодом Python. Можно также встретить файлы с родственными расширениями, такими как `.pyc` или `.pyo` – они не будут вызывать проблем в системе и вы будете встречать их, но мы пока не будем отвлекаться на них.

Комментарии

Комментарии в языке Python начинаются с символа решетки (#). Если этот символ находится в начале строки с программным кодом, то вся строка является комментарием. Символ # может также появляться в середине строки, и тогда комментарием является часть от символа решетки и до конца строки. Например:

```
# эта строка целиком является комментарием
foo = 1         # короткий комментарий: переменной 'foo' присваивается число 1
print 'Python and %s are number %d' % ('Django', foo)
```

Комментарии используются не только для добавления пояснений к близлежащему программному коду, но и могут предотвращать вы-

полнение строк с программным кодом. Отличным примером такого способ использования комментариев могут служить файлы с настройками, такие как `settings.py` – параметры настройки, которые не нужны или имеют значения, отличные от значений по умолчанию, комментируются, благодаря чему их проще будет активировать вновь или сделать выбор конфигурационных параметров более очевидным.

Переменные и присваивание значений

Чтобы в языке Python использовать переменные, не требуется «объявлять» их тип, как в некоторых других языках программирования. Python – это язык программирования с «динамической типизацией». Переменные можно представить себе как имена, ссылающиеся на безымянные объекты, которые хранят фактические значения, то есть для любой переменной можно в любой момент изменить значение, как показано ниже:

```
>>> foo = 'bar'
>>> foo
'bar'
>>> foo = 1
>>> foo
1
```

В этом примере в переменную `foo` сначала записывается ссылка на строковый объект `'bar'`, а затем – на целочисленный объект `1`. Примечательно, что строка, на которую ссылается переменная `foo`, исчезнет, если нет какой-то другой переменной, ссылающейся на нее (что вполне возможно!).

Так как имеется возможность присваивать именам другие значения, как в данном случае, никогда нельзя быть абсолютно уверенным в типе данных объекта, на который ссылается переменная в каждый конкретный момент времени, если не попытаться запросить эту информацию у интерпретатора. Однако, пока данная переменная ведет себя, как некоторый тип данных (например, если она обладает всеми методами строковых объектов), она может рассматриваться как экземпляр данного типа, даже если она имеет дополнительные атрибуты. Это называется грубым определением типов, или «утиной типизацией» – если это ходит как утка и крякает как утка, значит – это утка.

Операторы

Операторы являются довольно универсальной особенностью, и в языке Python присутствуют практически те же самые операторы, что и во многих других языках программирования. Сюда входят арифметические операторы, такие как `+`, `-`, `*` и т. д., а также соответствующие им *комбинированные операторы присваивания*, `+=`, `-=`, `*=` и т. д. Благодаря этому выражение `x = x + 1` можно записать, как `x += 1`. В языке

Python отсутствуют операторы инкремента и декремента (++ и --), которые вы могли использовать в других языках.

Имеются также стандартные операторы сравнения, такие как <, >=, ==, != и т. д., которые можно объединять с помощью логических операторов and и or. Имеется также логический оператор not, инвертирующий логический результат сравнивания. Ниже показано, как можно объединять операторы сравнения с помощью логического оператора and:

```
show_output = True
if show_output and foo == 1:
    print 'Python and %s are number %d' % ('Django', foo)
```

Вы уже знаете, что для выделения блоков программного кода в языке Python используются отступы, а не фигурные скобки. Ранее говорилось, что благодаря отступам очень легко определить, какому блоку принадлежит та или иная строка программного кода. Но, кроме того, при такой организации синтаксиса проблема «повисшего else» становится *невозможной* просто потому, что принадлежность любого предложения else определенному оператору if становится достаточно очевидной.

Следует также отметить, что в Python вообще не используются некоторые символы. Не только фигурные скобки не нужны, но и символ точки с запятой (;), завершающий строку программного кода, и символ доллара ($), и круглые скобки (()), окружающие условные выражения (как это видно в предыдущем примере). Иногда вам может встретиться символ @, обозначающий декораторы, и многочисленные варианты использования символа подчеркивания (_). Создатель языка Python полагает, что чем меньше символов, загромождающих программный код, тем проще его читать.

Стандартные типы данных в языке Python

Теперь мы познакомим вас со стандартными типами данных, которые вам придется использовать при работе с платформой Django. Сюда входят скаляры и литералы (такие, как числа и строки), а также «контейнеры» и структуры данных, используемые для группировки нескольких объектов. Прежде чем перейти к основным типам данных, следует заметить, что все объекты в языке обладают одной особенностью – все они обладают некоторым логическим значением.

Логические значения объектов

Как и в большинстве других языков программирования, в Python существует всего два логических значения: True и False. Значение любого типа в языке Python может быть представлено в виде логического значения независимо от фактического значения. Например, любое значение числового типа, равное нулю, рассматривается как значение False в логическом контексте, а все ненулевые значения – как значе-

ние True. Точно так же пустые контейнеры интерпретируются как значение False, а непустые – как значение True.

Для определения логического значения любого объекта можно использовать функцию bool(). Кроме того, значения True и False сами по себе являются обычными значениями, которые можно явно присваивать переменным.

```
>>> download_complete = False
>>> bool(download_complete)
False
>>> bool(-1.23)
True
>>> bool(0.0)
False
>>> bool("")
False
>>> bool([None, 0])
True
```

Предыдущие примеры и результаты вызова функции bool() должны быть понятны. Но последний пример таит в себе одну хитрость: несмотря на то, что оба элемента списка в логическом контексте имеют значение False, тем не менее, непустой список рассматривается как значение True. Возможность получения значения «истинности» объектов используется, когда эти объекты играют роль *условного выражения* в таких инструкциях, как if или while, где ход выполнения программы зависит от логического значения этих объектов.

Обратите также внимание на значение None в последнем примере. Это специальное значение, эквивалентное значению NULL или void в других языках. Значение None в логическом контексте всегда интерпретируется как False.

Логические значения являются литералами, так же как и числа, которые рассматриваются в следующем разделе.

Числа

В языке Python имеется два элементарных числовых типа: int (целые числа) и float (числа с плавающей точкой). В соответствии с мантрой «Простое лучше сложного», в языке Python имеется всего один целочисленный тип данных, int, в отличие от многих других языков программирования, имеющих несколько целочисленных типов.[1] В допол-

[1] Первоначально в языке Python имелся еще один целочисленный тип, который назывался long, но в настоящее время его функциональность была объединена в тип int. Однако в устаревшем программном коде и документации до сих пор можно увидеть завершающий символ «L», использовавшийся для представления длинных целых чисел, например: 1L, -42L, 99999999999999999L и т. д.

нение к обычной, десятичной форме записи, целые числа могут записываться в шестнадцатеричной (по основанию 16) и восьмеричной (по основанию 8) системах счисления. Тип `float` представляет вещественные числа с плавающей точкой двойной точности и должен быть знаком вам по другим языкам программирования. Ниже приводится несколько примеров целых чисел и чисел с плавающей точкой, а также некоторые операторы, применяемые к ним:

```
>>> 1.25 + 2.5
3.75
>>> -9 - 4
-13
>>> 1.1
1.1000000000000001
```

Ой! А что это получилось в последнем примере? Тип данных `float` может использоваться для представления огромного диапазона чисел, однако он не обладает высокой точностью, в терминах представления рациональных чисел с дробной частью в периоде. По этой причине был создан еще один тип данных, с именем `Decimal`, – для представления чисел с плавающей точкой; он не является встроенным типом данных и доступен в виде модуля `decimal`. Этот тип данных способен представлять более ограниченный диапазон данных, но с более высокой точностью. В языке Python имеется также встроенный числовой тип данных `complex`, который может использоваться в научных расчетах.

В табл. 1.1 перечисляются числовые типы данных, а также приводится несколько примеров.

Таблица 1.1. Встроенные числовые типы данных

Тип	Описание	Примеры
`int`	Целые числа со знаком (неограниченного размера)	`-1, 0, 0xE8C6, 0377, 42`
`float`	Числа с плавающей точкой двойной точности	`1.25, 4.3e+2, -5., -9.3e, 0.375`
`complex`	Комплексные числа (вещественная часть + мнимая)	`2+2j, .3-j, -10.3e+5-60j`

Числовые операторы

Числовые типы данных поддерживают основные арифметические операции, с которыми вы должны быть знакомы по другим языкам программирования: сложение (`+`), вычитание (`-`), умножение (`*`), деление (`/` и `//`), деление по модулю (`%`) и возведение в степень (`**`).

Оператор деления `/` представляет операцию «классического деления» в том смысле, что он усекает дробную часть результата, когда в операции участвуют два целых числа, и операцию «истинного деления»,

когда в операции участвуют числа с плавающей точкой. В языке Python имеется также явный оператор «деления с усечением дробной части», который всегда возвращает целое число независимо от типов операндов:

```
>>> 1 / 2        # деление с усечением дробной части (операнды типа int)
0
>>> 1.0 / 2.0    # true division (операнды типа float)
0.5
>>> 1 // 2       # деление с усечением дробной части (оператор //)
0
>>> 1.0 // 2.0   # деление с усечением дробной части (оператор //)
0.0
```

Наконец, к целым числам могут применяться битовые операторы И (`&`), ИЛИ (`|`), ИСКЛЮЧАЮЩЕЕ-ИЛИ (`^`) и инверсия (`~`), а также операторы сдвига влево и вправо (`<<` и `>>`) и соответствующие им комбинированные операторы присваивания, такие как `&=`, `<<=` и т. д.

Встроенные и фабричные функции для работы с числами

Для каждого числового типа имеется *фабричная* функция, которая позволяет выполнять преобразование из одного числового типа в другой. Некоторые читатели могут сказать: не «преобразование», а «приведение», но мы не используем этот термин в языке Python, поскольку здесь не выполняется *изменение* типа существующего объекта. Функция возвращает новый объект, созданный на основе первоначального (откуда и взялось название «фабричная»). Вызвав функцию `int(12.34)`, легко можно создать целочисленный объект со значением `12` (с ожидаемым усечением дробной части), вызов `float(12)` вернет значение `12.0`. Наконец, у нас имеются типы `complex` и `bool`.

В языке Python имеется также несколько *встроенных* функций, применяемых к числам, такие как `round`, выполняющая округление чисел с плавающей точкой до указанного числа знаков после запятой, или `abs`, возвращающая абсолютное значение числа. Ниже приводятся примеры использования этих и других встроенных функций, предназначенных для работы с числами:

```
>>> int('123')
123
>>> int(45.67)
45
>>> round(1.15, 1)
1.2
>>> float(10)
10.0
>>> divmod(15, 6)
(2, 3)
```

```
>>> ord('a')
97
>>> chr(65)
'A'
```

За дополнительной информацией по этим и другим числовым функциям обращайтесь к главе «Numbers» в книге «Core Python Programming» (Prentice Hall, 2006) или к другой справочной литературе, можно также обратиться к документации по языку Python в Интернете. Теперь рассмотрим строки и другие основные контейнерные типы данных в языке Python.

Последовательности и итерируемые объекты

Во многих языках программирования имеются такие структуры данных, как массивы, которые обычно имеют фиксированный размер и содержат группу объектов одного типа, доступных по индексу. Последовательности в языке Python играют ту же роль, но могут содержать объекты разных типов, а также увеличиваться или уменьшаться в размерах. В этом разделе мы рассмотрим два наиболее популярных типа данных в языке Python: **списки (**`[1, 2, 3]`**)** и **строки (**`'python'`**)**. Они являются частью обширного множества структур данных, называемых **последовательностями**.

Последовательности являются одним из представителей **итерируемых объектов** – структур данных, по элементам которых можно выполнять «обход» или «итерации». Главная особенность итерируемых объектов состоит в том, что вы имеете возможность обращаться к ним, запрашивая доступ к следующему элементу с помощью метода `next`, который продолжает читать внутреннюю коллекцию своих объектов, пока не исчерпает ее. Последовательности в языке Python поддерживают не только последовательный способ доступа (хотя в 99 процентах случаев вы будете использовать циклы `for` вместо метода `next`), но и произвольный, что дает возможность получать доступ к определенному объекту в последовательности. Например, выражение `my_list[2]` вернет третий элемент списка (индексирование элементов последовательностей начинается с 0).

Третий тип последовательностей называется **кортежем**. Кортежи легко можно описать фразой: «списки с ограниченными возможностями, доступные только для чтения», вследствие чего они используются совершенно для других целей. Они вряд ли станут основной структурой данных в ваших приложениях, но мы должны рассказать, что они из себя представляют и для чего используются. Поскольку вы уже наверняка знаете, что такое строки, мы сначала рассмотрим списки, а кортежи обсудим в последнюю очередь. В табл. 1.2 перечислены все обсуждаемые типы последовательностей и даются некоторые примеры.

Таблица 1.2. Примеры использования последовательностей

Тип	Примеры
str	'django', '\n', "", "%s is number %d" % ('Python', 1), """hey there"""
list	[123, 'foo', 3.14159], [], [x.upper() for x in words]
tuple	(456, 2.71828), (), ('need a comma even with just 1 item',)

Извлечение срезов последовательностей

Чуть выше упоминалось, что существует возможность прямого обращения к элементам последовательностей по их индексам. Ниже приводятся несколько примеров такого способа обращения к строкам. В отличие от многих других языков, строки в языке Python могут рассматриваться и как самостоятельные объекты, и как списки отдельных символов.

```
>>> s = 'Python'
>>> s[0]
'P'
>>> s[4]
'o'
>>> s[-1]
'n'
```

В языке Python допускается также использовать отрицательные индексы. Наверняка вам приходилось использовать выражение `data[len(data)-1]` или `data[data.length-1]`, чтобы получить доступ к последнему элементу какого-нибудь массива. Как показывает последний пример в предыдущем фрагменте, для этого достаточно использовать индекс `-1`.

Точно так же с помощью индексов можно обратиться сразу к нескольким элементам последовательности – в языке Python эта операция называется **извлечением среза**. В операции извлечения среза участвует пара индексов, разделенных двоеточием (`:`), – скажем, *i* и *j*. Когда производится попытка извлечь срез последовательности, интерпретатор возвращает подмножество элементов, начиная с первого индекса *i* и заканчивая вторым индексом *j*, но не включая элемент с этим индексом в срез.

```
>>> s = 'Python'
>>> s[1:4]
'yth'
>>> s[2:4]
'th'
>>> s[:4]
'Pyth'
>>> s[3:]
'hon'
>>> s[3:-1]
'ho'
>>> s[:]
```