А. ГОЛОВАТЫЙ,
Д. КАПЛАН-МОСС

второе издание

Django

ПОДРОБНОЕ РУКОВОДСТВО

СИМВОЛ®

The Definitive Guide to Django

Second Edition

Adrian Holovaty,
Jacob Kaplan-Moss

Apress®

HIGH TECH

Django

Подробное руководство

Второе издание

Адриан Головатый,
Джейкоб Каплан-Мосс

Символ®

Санкт-Петербург — Москва
2010

Серия «High tech»

Адриан Головатый, Джейкоб Каплан-Мосс

Django. Подробное руководство, 2-е издание

Перевод А. Киселева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Выпускающий редактор	*П. Щеголев*
Научный редактор	*А. Киселев*
Редактор	*Ю. Бочина*
Корректор	*Е. Кирюхина*
Верстка	*К. Чубаров*

Головатый А., Каплан-Мосс Дж.

Django. Подробное руководство, 2-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 560 с., ил.

ISBN 978-5-93286-187-5

Эта книга посвящена Django 1.1 – последней версии фреймворка для разработки веб-приложений, который позволяет создавать и поддерживать сложные и высококачественные веб-ресурсы с минимальными усилиями. Django – это тот инструмент, который превращает работу в увлекательный творческий процесс, сводя рутину к минимуму. Данный фреймворк предоставляет общеупотребительные шаблоны веб-разработки высокого уровня абстракции, инструменты для быстрого выполнения часто встречающихся задач программирования и четкие соглашения о способах решения проблем.

Авторы подробно рассматривают компоненты Django и методы работы с ним, обсуждают вопросы эффективного применения инструментов в различных проектах. Эта книга отлично подходит для изучения разработки интернет-ресурсов на Django – от основ до таких специальных тем, как генерация PDF и RSS, безопасность, кэширование и интернационализация. Издание ориентировано на тех, кто уже имеет навыки программирования на языке Python и знаком с основными принципами веб-разработки.

ISBN 978-5-93286-187-5
ISBN 978-1-4302-1936-1 (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.

Подписано в печать 29.04.2010. Формат 70×100 $^{1}/_{16}$. Печать офсетная.
Объем 35 печ. л. Тираж 1500 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

Посвящается сообществу Django

Оглавление

Предисловие

Приветствуем читателей второго издания «Django. Подробное руководство», которое неформально называют «Django Book»! В этой книге мы постараемся научить вас, как эффективно разрабатывать сайты с помощью фреймворка Django.

Когда мы с Джейкобом Каплан-Моссом писали первое издание этой книги, еще не была выпущена версия Django 1.0. После выхода версии 1.0, которая не обладала полной обратной совместимостью, первое издание, естественно, устарело, и читатели стали требовать обновления. Рад сообщить, что это издание охватывает версию Django 1.1, так что какое-то время вам послужит.

Выражаю признательность многочисленным участникам дискуссий, присылавшим свои замечания, исправления и пожелания на сайт *http://djangobook.com/*, служащий дополнением к этой книге. Именно там я выкладывал черновые варианты глав по мере их написания. Ребята, вы молодцы!

Адриан Головатый, один из создателей
и Великодушных Пожизненных Диктаторов Django

Об авторах

Адриан Головатый (Adrian Holovaty) – один из создателей и Великодушных Пожизненных Диктаторов Django. Он руководит недавно созданной веб-компанией EveryBlock. Живет в Чикаго, в свободное время пытается играть на гитаре в стиле Джанго Рейнхардта.

Джейкоб Каплан-Мосс (Jacob Kaplan-Moss) – ведущий разработчик и второй Великодушный Пожизненный Диктатор Django. Джейкоб – совладелец консалтинговой компании Revolution Systems, помогающей клиентам извлекать максимум пользы из программного обеспечения с открытым исходным кодом. Ранее Джейкоб работал в газете *Lawrence Journal-World*, выходящей в городе Лоуренс, штат Канзас, где и был разработан фреймворк Django. Там Джейкоб был ведущим разработчиком коммерческой платформы публикации в веб под названием Ellington, предназначенной для медийных компаний.

О рецензенте оригинального издания

Шон Легассик (Sean Legassick) уже более 15 лет занимается разработкой программного обеспечения. Его работа по проектированию архитектуры фреймворка Chisimba с открытым исходным кодом стала существенным вкладом в культуру разработки ПО в Южной Африке и других развивающихся странах. Он один из основателей компании MobGeo, занимающейся созданием инновационных решений по мобильному маркетингу с привязкой к местонахождению пользователя. Помимо разработки программ пишет статьи о политике и культуре.

Благодарности

Спасибо всем, кто принимал участие в написании первых заметок к этой книге в Сети, и сотрудникам издательства Apress за великолепно выполненное редактирование.

Введение

Когда-то давно веб-разработчики писали все страницы вручную. Для обновления сайта необходимо было редактировать HTML-код; изменение дизайна сайта влекло за собой переработку каждой страницы по отдельности.

Шло время, сайты становились все более объемными и навороченными, и очень скоро стало очевидно, что такая работа слишком утомительна, отнимает много времени и вообще никуда не годится. Группа энтузиастов в NCSA (Национальный центр по использованию суперкомпьютеров, в котором был создан первый графический броузер Mosaic) решила эту проблему, позволив веб-серверу запускать внешние программы, которые умели динамически генерировать HTML. Протокол взаимодействия с такими программами, названный ими Common Gateway Interface, или CGI (интерфейс общего шлюза), навсегда изменил Всемирную паутину.

Сейчас даже трудно вообразить, каким откровением стал тогда CGI: он позволял рассматривать HTML-страницы не как простые файлы на диске, а как ресурсы, генерируемые динамически по запросу. Изобретение CGI ознаменовало рождение первого поколения динамических веб-сайтов.

Однако у CGI были и недостатки. CGI-сценарии были вынуждены содержать много повторяющегося от сценария к сценарию кода, имели сложности с повторным использованием, а писать и даже читать такие сценарии бывало поначалу довольно трудно.

Многие из этих проблем были решены с появлением технологии PHP, которая штурмом захватила весь мир. Сейчас это самое популярное средство создания динамических сайтов. Его принципы были позаимствованы десятками других языков и сред разработки (ASP, JSP и т. п.). Главным новшеством PHP стала простота использования: PHP-код вкрапляется прямо в HTML. Кривая обучения для любого знакомого с HTML человека на удивление полога.

Но и технология PHP не лишена недостатков. Будучи чрезвычайно простой в применении, она провоцирует создание небрежного, плохо продуманного кода с большим количеством повторяющихся фрагментов. Хуже того, PHP почти не защищает от написания уязвимого кода, поэтому многие программисты на PHP начинают интересоваться вопросами безопасности только тогда, когда уже слишком поздно.

Эти и другие подобные недостатки стали побудительным мотивом для создания целого ряда популярных фреймворков третьего поколения, в частности Django и Ruby on Rails, доказывающих осознание возросшей в последнее время значимости Интернета.

Лавинообразный рост количества веб-приложений породил еще одно требование: веб-разработчики должны успевать все больше и больше за тот же отрезок времени.

Django был задуман как ответ на эти требования. Этот фреймворк позволяет создавать насыщенные, динамичные, привлекательные сайты в кратчайшие сроки. Django спроектирован так, чтобы разработчик мог сосредоточиться на решении увлекательных, содержательных задач, а не отвлекаться на повторяющуюся рутину. Для достижения этой цели он предоставляет общеупотребительные шаблоны веб-разработки высокого уровня абстракции, инструменты для быстрого выполнения часто встречающихся задач программирования и четкие соглашения о способах решения проблем. В то же время Django старается не мешать программисту, позволяя при необходимости выходить за рамки фреймворка.

Мы написали эту книгу, потому что твердо верим, что Django улучшает процесс веб-разработки. Книга построена так, чтобы вы могли как можно скорее приступить к созданию собственных проектов. А прочитав ее до конца, вы узнаете все, что необходимо для успешного проектирования, реализации и развертывания сайта, которым можно было бы гордиться.

Нас очень интересует ваше мнение. Электронная версия этой книги, размещенная на сайте *http://djangobook.com/*, позволяет оставлять замечания о любой части книги и обсуждать ее с другими читателями. По мере сил мы стараемся прочитывать все замечания и отвечать на них. Если вы предпочитаете общаться по электронной почте, пишите нам на адрес *feedback@djangobook.com*. Так или иначе мы с нетерпением ждем ваших отзывов!

Мы рады, что вы заинтересовались этой книгой, и надеемся, что разработка с помощью Django станет для вас увлекательным, приятным и полезным занятием.

I

Начальные сведения

1

Введение в Django

Эта книга посвящена Django – фреймворку веб-разработки, который позволяет экономить ваше время и превращает разработку веб-приложений в удовольствие. Используя Django, вы сможете с минимальными усилиями создавать и поддерживать высококачественные веб-приложения.

При наличии хороших инструментов веб-разработка – это увлекательный творческий процесс, а если таких инструментов нет, то она может оказаться скучной чередой повторяющихся действий. Django дает возможность сосредоточиться на приятных моментах работы – ключевой части веб-приложения, сводя рутину к минимуму. Для достижения этой цели он предоставляет общеупотребительные шаблоны веб-разработки высокого уровня абстракции, инструменты для быстрого выполнения часто встречающихся задач программирования и четкие соглашения о способах решения проблем. В то же время Django старается не мешать вам и при необходимости позволяет выходить за рамки фреймворка.

Задача этой книги – сделать вас экспертом по Django. Мы подойдем к ее решению с двух сторон. Во-первых, подробно объясним, как именно работает Django и как с его помощью строятся веб-приложения. Во-вторых, там, где это уместно, мы будем обсуждать общие концепции, отвечая на вопрос: «Как можно эффективно применять эти инструменты в своих проектах?». По мере чтения книги вы приобретете навыки, необходимые для быстрой разработки сложных веб-сайтов, код которых понятен и прост в сопровождении.

Что такое веб-фреймворк?

Django – один из наиболее заметных представителей *веб-фреймворков* нового поколения. Но что на самом деле означает этот термин?

Для ответа на этот вопрос имеет смысл рассмотреть структуру веб-приложения, написанного на языке Python *без* применения фреймворка. Подобный подход мы будем использовать на протяжении всей книги: демонстрируя, каких трудов стоит решение без вспомогательных средств, мы надеемся, что вы оцените их полезность. (Кстати, знать, как можно добиться результата без использования вспомогательных средств, полезно еще и потому, что эти функции не всегда доступны. Но главное – понимание того, *почему* нечто работает так, а не иначе, повышает вашу квалификацию как веб-разработчика.)

Один из самых простых и незамысловатых способов создать веб-приложение на Python с нуля – это воспользоваться стандартом Common Gateway Interface (CGI), который приобрел популярность примерно в 1998 году. Сделать это можно, в общих чертах, следующим образом: создайте сценарий на языке Python, который будет возвращать HTML, сохраните его на веб-сервере с расширением `.cgi`[1] и зайдите на эту страницу с помощью броузера. Вот и все.

Ниже приведен пример CGI-сценария на языке Python, который выводит названия десяти свежеизданных книг из базы данных. Не вдаваясь в детали синтаксиса сценария, попробуйте понять, как он работает[2]:

```
#!/usr/bin/env python

import MySQLdb

print "Content-Type: text/html\n"
print "<html><head><title>Книги</title></head>"
print "<body>"
print "<h1>Книги</h1>"
print "<ul>"

connection = MySQLdb.connect(user='me', passwd='letmein', db='my_db')
cursor = connection.cursor()
cursor.execute("SELECT name FROM books ORDER BY pub_date DESC LIMIT 10")

for row in cursor.fetchall():
    print "<li>%s</li>" % row[0]

print "</ul>"
```

[1] Необязательно давать файлу расширение `.cgi`, но совершенно необходимо поместить его в каталог `cgi-bin` и сделать выполняемым с помощью команды `chmod +x <имя_файла>`. В любом другом каталоге веб-сервер будет интерпретировать сценарий как простой текстовый файл и просто выведет его содержимое в окне броузера, а если файл сценария не сделать выполняемым, при обращении к нему веб-сервер вернет сообщение об ошибке. – *Прим. науч. ред.*

[2] Если в сценарии используются кириллические символы, как в данном примере, и при этом в региональных настройках системы выбрана кодировка символов UTF-8, в начало сценария (ниже первой строки) следует добавить строку: «`# -*- coding: utf-8 -*-`». – *Прим. науч. ред.*

```
    print "</body></html>"

    connection.close()
```

Сначала, чтобы удовлетворить требования CGI, сценарий выводит строку «Content-Type», а за ней – пустую строку. Далее выводится вводная часть HTML-документа, устанавливается соединение с базой данных и выполняется запрос, который выбирает из базы данных названия десяти книг, изданных последними. Перебирая в цикле данные о книгах, сценарий генерирует HTML-список из их названий. В заключение выводится оставшаяся часть HTML-документа, после чего закрывается соединение с базой данных.

Если нужно написать всего одну изолированную страницу, то описанный лобовой подход не так уж плох. Этот код легко понять – даже новичок сумеет прочесть приведенный код и разобраться в том, что он делает, с начала и до конца. Больше ничего изучать не надо, читать еще какой-то код нет нужды. Да и развертывать его просто: достаточно поместить код в файл с расширением `.cgi`, скопировать его на веб-сервер и зайти на страницу с помощью броузера.

Однако этому подходу присущ целый ряд проблем и неудобств. Задайте себе следующие вопросы.

- Как быть, если к базе данных нужно подключаться из разных мест в приложении? Очевидно, было бы крайне нежелательно дублировать код, выполняющий соединение с базой данных, в каждом CGI-сценарии. Правильнее было бы вынести его в общую функцию.
- Так ли уж надо разработчику помнить о выводе строки «Content-Type» и о необходимости закрывать соединение с базой данных? Необходимость писать подобный повторяющийся во многих местах код только снижает продуктивность программиста и создает лишнюю возможность сделать ошибку. Такого рода инициализацию и очистку лучше поручить каким-нибудь общим компонентам системы.
- Что если код будет эксплуатироваться в разных условиях, например будут меняться база данных и пароль доступа к ней? Здесь не обойтись без средства, позволяющего задавать свои настройки конфигурации для каждого случая. Что если веб-дизайнер, не умеющий программировать на языке Python, захочет изменить дизайн страницы? Один неверно набранный символ – и все приложение перестает работать. В идеале логика работы страницы – выборка названий книг из базы данных – должна быть отделена от ее HTML-представления, чтобы дизайнер мог редактировать последнюю, не затрагивая первую.

Именно такие задачи и призван решать веб-фреймворк. Он предоставляет готовый набор взаимосвязанных компонентов для программирования приложений, помогая сосредоточиться на написании ясного, удобного для сопровождения кода и не изобретать каждый раз велосипед. Именно этим, если коротко, Django и полезен.

Шаблон проектирования MVC

А теперь копнем поглубже и рассмотрим пример, демонстрирующий разницу между описанным ранее подходом и решением на основе веб-фреймворка. Ниже показано, как можно было бы написать тот же код, если воспользоваться Django. Прежде всего, отметим, что мы разбили его на три Python-файла (`models.py`, `views.py`, `urls.py`) и один HTML-шаблон (`latest_books.html`):

```
# models.py (таблицы базы данных)

from django.db import models

class Book(models.Model):
    name = models.CharField(max_length=50)
    pub_date = models.DateField()


# views.py (бизнес-логика)

from django.shortcuts import render_to_response
from models import Book

def latest_books(request):
    book_list = Book.objects.order_by('-pub_date')[:10]
    return render_to_response('latest_books.html', {'book_list': book_list})


# urls.py (конфигурация URL)

from django.conf.urls.defaults import *
import views

urlpatterns = patterns('',
    (r'^latest/$', views.latest_books),
)
# latest_books.html (шаблон)

<html><head><title>Книги</title></head>
<body>
<h1>Книги</h1>
<ul>
{% for book in book_list %}
<li>{{ book.name }}</li>
{% endfor %}
</ul>
</body></html>
```

Не будем копаться в деталях; главное – уловить общий смысл. Самое важное здесь – *разграничение обязанностей*.

- В файле `models.py` содержится описание таблицы базы данных, представленной классом Python. Этот класс называется *моделью*. Используя его для создания, выборки, изменения и удаления записей

таблицы, мы должны написать лишь небольшое количество простого кода на языке Python и никаких однообразных повторяющихся SQL-конструкций.

- В файле `views.py` находится бизнес-логика страницы. Функция `latest_books()` называется *представлением*.
- В файле `urls.py` описывается, какое представление следует вызывать для URL, заданного в виде шаблона. В данном случае URL, заканчивающийся на `/latest/`, будет обрабатываться функцией `latest_books()`. Другими словами, если ваш сайт находится в домене example.com, то любое посещение URL `http://example.com/latest/` будет обработано функцией `latest_books()`.
- Файл `latest_books.html` – это HTML-шаблон, описывающий дизайн страницы. В нем используется специальный язык шаблонов, включающий основные логические конструкции, например `{% for book in book_list %}`.

Описанные выше файлы в совокупности представляют собой разновидность[1] шаблона проектирования Модель-Представление-Контроллер (Model-View-Controller – MVC). Говоря простыми словами, MVC – это такой способ разработки программного обеспечения, при котором код определения и доступа к данным (модель) отделен от логики взаимодействия с приложением (контроллер), которая, в свою очередь, отделена от пользовательского интерфейса (представление). (Более подробно мы будем рассматривать MVC в главе 5.)

Главное достоинство такого подхода состоит в том, что компоненты *слабо связаны*. У каждого компонента веб-приложения, созданного на базе Django, имеется единственное назначение, поэтому его можно изменять независимо от остальных компонентов. Например, разработчик может изменить URL некоторой части приложения, и это никак не скажется на ее реализации. Дизайнер может изменить HTML страницы, не трогая генерирующий ее код на языке Python. Администратор базы данных может переименовать таблицу базы данных и описать это изменение в одном-единственном месте, а не заниматься контекстным поиском и заменой в десятках файлов.

В этой книге каждой составной части шаблона MVC посвящена отдельная глава. В главе 3 рассматриваются представления, в главе 4 – шаблоны, а в главе 5 – модели.

---

[1] Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представлением (*View*), а презентационная логика Представления реализуется в Django на уровне Шаблона (*Template*). Из-за этого архитектуру уровней Django часто называют «Модель-Шаблон-Вид» (MTV). – *Прим. науч. ред.*

История развития Django

Прежде чем продолжить рассмотрение кода, посвятим несколько минут знакомству с историей Django. Выше мы сказали, что будем показывать, как можно решить задачу, *не* прибегая к вспомогательным средствам, чтобы вы лучше поняли механизм работы последних. При этом полезно понимать, *для чего* был создан фреймворк Django, поскольку именно в историческом контексте становится ясно, почему Django работает так, а не иначе.

Если у вас есть достаточно продолжительный опыт создания веб-приложений, то вы наверняка знакомы с проблемами, присущими рассмотренному выше примеру CGI-сценария. Классический веб-разработчик проходит такой путь:

1. Пишет веб-приложение с нуля.
2. Пишет еще одно веб-приложение с нуля.
3. Осознает, что первое веб-приложение имеет много общего со вторым.
4. Перерабатывает код так, чтобы некоторые вещи из первого приложения можно было использовать повторно во втором.
5. Повторяет шаги 2–4 несколько раз.
6. Понимает, что он придумал фреймворк.

Именно так и был создан Django!

Django развивался естественным образом по ходу разработки настоящих коммерческих приложений, написанных группой веб-разработчиков из Лоуренса, штат Канзас, США. Он появился на свет осенью 2003 года, когда два программиста, работавших в газете *Lawrence Journal-World*, Адриан Головатый (Adrian Holovaty) и Саймон Уиллисон (Simon Willison), начали использовать для создания приложений язык Python.

Группа World Online, отвечавшая за разработку и сопровождение нескольких местных новостных сайтов, трудилась в условиях, диктуемых жесткими сроками, характерными для журналистики. Журналисты (и руководство) требовали, чтобы новые функции и целые приложения реализовывались на всех сайтах, включая LJWorld.com, Lawrence.com и KUsports.com, в очень сжатые сроки, зачастую в течение нескольких дней или даже часов с момента постановки задачи. Необходимо было что-то предпринять. Саймон и Адриан вышли из положения, создав фреймворк для веб-разработки, который помогал экономить драгоценное время, – только так можно было писать поддающиеся сопровождению приложения в столь сжатые сроки.

Летом 2005 года, доведя фреймворк до состояния, когда на нем было реализовано большинство сайтов World Online, группа, в которую к тому времени вошел еще и Джейкоб Каплан-Мосс (Jacob Kaplan-Moss), решила выпустить его в виде ПО с открытым исходным кодом. Фреймворк

был выпущен в июле 2005 года и назван Django в честь джазового гитариста Джанго Рейнхардта (Django Reinhardt).

Сегодня, по прошествии нескольких лет, Django превратился в популярный проект с открытым исходным кодом, имеющий десятки тысяч пользователей и разработчиков по всему миру. Два первоначальных разработчика World Online («Великодушные Пожизненные Диктаторы» Адриан и Джейкоб) по-прежнему определяют общее направление развития фреймворка, но в целом он в гораздо большей степени является плодом коллективных усилий.

Мы рассказали эту историю, чтобы помочь вам понять две важные вещи. Первая – это основное назначение Django. Так как Django родился в мире новостей, он включает ряд функций (например, административный интерфейс, рассматриваемый в главе 6), специально предназначенных для сайтов с богатым информационным наполнением, таких как Amazon.com, Craigslist и The Washington Post, где публикуется динамично меняющаяся информация, извлекаемая из базы данных. Но не отворачивайтесь сразу – хотя Django особенно хорош для разработки таких сайтов, ничто не мешает использовать его в качестве эффективного инструмента создания любых динамичных сайтов. (Есть разница между *особенно эффективен* для чего-то и *неэффективен* для всего остального.)

Второе, на что стоит обратить внимание, – как происхождение Django сформировало культуру его открытого сообщества. Поскольку Django – плод практического программирования, а не академических исследований, и не коммерческий продукт, то он «заточен» под решение тех задач веб-разработки, с которыми сталкивались – и продолжают сталкиваться – его авторы. Поэтому сам Django активно совершенствуется чуть ли не ежедневно. Программисты, сопровождающие фреймворк, кровно заинтересованы в том, чтобы Django экономил время разработчиков, помогал создавать приложения, которые было бы легко сопровождать, и показывал хорошую производительность в условиях высокой нагрузки. Даже не будь других мотивов, достаточно и эгоистичного желания сэкономить собственное время и получать удовольствие от работы. (Проще говоря, это их хлеб.)

Как читать эту книгу

Мы стремились к тому, чтобы эту книгу можно было как читать подряд, так и использовать в качестве справочника, но предпочтение отдавали первой цели. Как уже отмечалось, наша задача – сделать из вас эксперта по Django, и мы полагаем, что наилучший путь для этого – связное повествование и множество примеров, а не исчерпывающий, но сухой перечень всех функций Django. (Как говорится, нельзя выучить язык, просто освоив алфавит.)

Поэтому мы рекомендуем читать главы с 1 по 12 по порядку. В них говорится об основах работы с Django; прочитав эти главы, вы сможете создавать и развертывать сайты, построенные на основе этого фреймворка. Главы 1–7 составляют «базовый курс», в главах 8–11 описываются более развитые средства Django, а глава 12 посвящена развертыванию. В оставшихся главах, с 13 по 20, рассматриваются конкретные особенности Django – их можно читать в любом порядке.

Приложения содержат справочный материал. Скорее всего, вы время от времени будете обращаться к ним и к документации на сайте *http://www.djangoproject.com/*, чтобы вспомнить синтаксис или найти краткое описание работы отдельных частей Django.

Необходимые знания по программированию

Читатель должен быть знаком с основами процедурного и объектно-ориентированного программирования: управляющими конструкциями (например, `if`, `while`, `for`), структурами данных (списками, хешами/словарями), понятиями переменной, класса и объекта.

Опыт веб-разработки, естественно, был бы весьма кстати, но для понимания материала необязателен. На протяжении всей книги мы стараемся знакомить читателей, не имеющих опыта, с рекомендуемыми приемами веб-разработки.

Необходимые знания о языке Python

В своей основе Django – это просто набор библиотек, написанных на языке программирования Python. При разработке сайта с использованием Django вы будете писать код на языке Python, обращающийся к этим библиотекам. Таким образом, изучение Django сводится к изучению двух вопросов: как программировать на Python и как устроены библиотеки Django.

Если у вас уже есть опыт программирования на Python, то больших проблем с погружением в Django не возникнет. По большому счету в коде Django не так уж много «магии» (программных трюков, реализацию которых трудно понять и объяснить). Для вас изучение Django будет означать изучение соглашений и API, принятых в этом фреймворке.

А если вы не знакомы с Python, то вас ждет увлекательный опыт. Изучить этот язык легко, а программировать на нем – сплошное удовольствие! Хотя мы не включили в книгу полный учебник по Python, но по мере необходимости рассказываем о его особенностях и возможностях, особенно когда код интуитивно не очевиден. Тем не менее мы рекомендуем прочитать официальное руководство по Python, имеющееся на сайте *http://docs.python.org/tut/*. Мы также рекомендуем бесплатную книгу

Марка Пилгрима (Mark Pilgrim) «Dive Into Python» (Apress, 2004)[1], которая выложена на сайте *http://www.diveintopython.org/* и опубликована издательством Apress.

Какая версия Django необходима

В этой книге рассматривается версия Django 1.1.

Разработчики Django гарантируют обратную совместимость в пределах «основного номера версии». Это означает, что приложение, написанное для Django 1.1, будет работать также с версиями 1.2, 1.3, 1.9 и вообще с любой версией, номер которой начинается с «1.». Но при переходе на версию 2.0 приложение, возможно, придется переписать; впрочем, до версии 2.0 еще далеко. Просто для справки скажем, что для выпуска версии 1.0 потребовалось более трех лет. (Эта политика совместимости действует и для самого языка Python: код, написанный для версии Python 2.0, будет работать с Python 2.6, но необязательно с Python 3.0.) Поскольку эта книга ориентирована на Django 1.1, она еще некоторое время вам послужит.

Где получить помощь

Одно из величайших достоинств Django – его доброжелательное и всегда готовое прийти на помощь сообщество. Если возникло затруднение в любом аспекте Django – будь то установка, проектирование приложения, проектирование базы данных или развертывание, – не стесняйтесь задавать вопросы в Сети.

- Список рассылки Django – это место, в котором тысячи пользователей задают вопросы и отвечают на них. Бесплатно подписаться можно на странице *http://www.djangoproject.com/r/django-users*.
- IRC-канал Django – это место, где пользователи встречаются и помогают друг другу в ходе непосредственного общения. Присоединяйтесь, зарегистрировавшись в канале `#django` IRC-сети Freenode.

Что дальше?

В следующей главе мы приступим к изучению Django, начав с установки и начальной настройки.

---

[1] Имеется неполный перевод этой книги на русский язык: *http://ru.diveintopython.org/toc.html. – Прим. науч. ред.*

2

Приступая к работе

Современные среды веб-разработки состоят из множества взаимосвязанных компонентов, поэтому и процесс установки Django обычно имеет несколько шагов. В этой главе мы покажем, как установить сам фреймворк и некоторые дополнительные компоненты окружения, которые нам понадобятся.

Поскольку Django написан исключительно на языке Python, то он работает везде, где могут выполняться программы на этом языке, в том числе и на некоторых мобильных телефонах! В этой главе мы рассмотрим лишь типичные сценарии установки Django и будем считать, что вы устанавливаете фреймворк на настольный ПК/ноутбук или сервер.

Позже (в главе 12) мы покажем, как можно развернуть Django для промышленного использования.

Установка Python

Так как Django целиком написан на языке Python, то первым шагом в процессе установки фреймворка будет установка среды выполнения языка Python.

Версии Python

Ядро фреймворка Django работает с версиями языка Python от 2.3 до 2.6 включительно, а для поддержки необязательной геоинформационной системы (ГИС) требуется версия в диапазоне от 2.4 до 2.6.

Если вы не уверены в том, какую версию Python вам лучше установить, и свободны в своем выборе, то лучше выберите последнюю версию серии 2.x, то есть 2.6. Хотя Django одинаково хорошо работает с любой версией от 2.3 до 2.6, более поздние версии Python обладают лучшей производительностью и имеют дополнительные возможности, которые могут пригодиться в ваших приложениях. Кроме того, не исключено,

что вы захотите воспользоваться некоторыми дополнительными модулями Django, для которых требуется версия Python выше, чем 2.3, так что, устанавливая самую свежую версию, вы предоставляете себе максимальную свободу.

> **Django и Python 3.0**
>
> Во время работы над этой книгой вышла версия Python 3.0, но Django ее пока не поддерживает. Количество несовместимых изменений языка, реализованных в Python 3.0, весьма велико, поэтому мы думаем, что для модернизации большинства крупных библиотек и фреймворков, написанных на Python, потребуется несколько лет.
>
> Если вы только приступаете к работе с Python и размышляете, начать ли изучение с версии 2.*x* или 3.*x*, наш совет – держитесь Python 2.*x*.

Установка

Если вы работаете с операционной системой Linux или Mac OS X, то, скорее всего, Python уже установлен. Введите в командной строке (в случае OS X – в программе Приложения/Служебные/Терминал) слово `python`. Если в ответ будет напечатан показанный ниже текст (или подобный ему), значит, Python установлен:

```
Python 2.4.1 (#2, Mar 31 2005, 00:05:10)
[GCC 3.3 20030304 (Apple Computer, Inc. build 1666)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

В противном случае придется скачать и установить Python. Это совсем несложно и не займет много времени; подробные инструкции есть на сайте *http://www.python.org/download/*.

Установка Django

В любой момент времени вам доступны две особенные версии Django: последний официальный выпуск и самая свежая версия основной линии разработки (*trunk*). Какую версию выбрать, зависит от ваших задач. Вам нужна стабильная, протестированная версия Django или та, в которую включены самые последние функции, быть может, для того, чтобы предложить собственные наработки для Django? Но учтите, за актуальность придется расплачиваться стабильностью работы.

Мы рекомендуем пользоваться официальным выпуском, однако знать о существовании версии основной линии разработки необходимо, пото-

му что она часто упоминается в документации и в разговорах между членами сообщества.

Установка официального выпуска

Номера версий официальных выпусков имеют вид 1.0.3 или 1.1, причем самая последняя версия всегда доступна на странице *http://www.djangoproject.com/download/*.

Если вы работаете с дистрибутивом Linux, в который включен пакет Django, то имеет смысл пользоваться версией из дистрибутива. Тогда вы будете получать все обновления безопасности.

Если же доступа к готовому пакету у вас нет, то можно скачать и установить платформу вручную. Для этого сначала загрузите архив, который называется примерно так: `Django-1.0.2-final.tar.gz`. (Не имеет значения, в каком локальном каталоге будет сохранен загруженный файл; программа установки автоматически скопирует файлы Django в нужное место.) Затем распакуйте архив и запустите сценарий `setup.py` точно так же, как любой другой Python-сценарий.

Вот как выглядит процесс установки в UNIX-системах:

1. `tar xzvf Django-1.0.2-final.tar.gz`
2. `cd Django-*`
3. `sudo python setup.py install`

Для распаковки `tar.gz`-архивов в Windows мы рекомендуем пользоваться программой 7-Zip (*http://www.djangoproject.com/r/7zip/*). Распаковав архив, запустите командную оболочку с привилегиями администратора и выполните следующую команду, находясь в каталоге, имя которого начинается с `Django-`:

```
python setup.py install
```

Для тех, кому интересно, сообщим, что файлы Django помещаются в подкаталог `site-packages` каталога установки Python – именно там Python ищет сторонние библиотеки. Обычно это каталог */usr/lib/python2.4/site-packages*.

Установка версии основной линии разработки

Самую свежую версию Django, которая называется версией основной линии разработки, можно получить из репозитория Subversion проекта Django. Эту версию следует устанавливать, если вам нравится находиться на острие событий или вы хотите предложить для Django собственный код.

Subversion – бесплатная система управления версиями с открытым исходным кодом. Команда Django применяет ее для управления всем кодом Django. Чтобы извлечь самую свежую версию исходного кода из репозитория, вам понадобится программа-клиент для Subversion. Созданную

при этом локальную копию кода Django в любой момент можно обновить, получив из репозитория последние изменения и улучшения, внесенные разработчиками Django.

Работая с версией основной линии разработки, имейте в виду, что никто не дает гарантий отсутствия ошибок. Однако честно предупредив вас о возможных последствиях, добавим, что некоторые члены команды Django все же применяют такие версии для промышленного использования, так что они сами заинтересованы в ее стабильности.

Чтобы получить последнюю версию основной линии разработки Django, выполните следующие действия:

1. Убедитесь в том, что установлен клиент Subversion. Скачать бесплатную программу можно с сайта *http://subversion.tigris.org/*, а прекрасно написанная документация имеется на сайте *http://svnbook.red-bean.com/*.

   Тем, кто работает на платформе Mac с операционной системой версии OS X 10.5 или выше, повезло больше – система Subversion там уже установлена. Убедиться в этом можно, введя команду `svn --version` в терминале.

2. Извлеките версию основной линии разработки, выполнив команду `svn co http://code.djangoproject.com/svn/django/trunk djtrunk`.

3. Найдите в каталоге установки Python подкаталог `site-packages`; обычно он находится в `/usr/lib/python2.4/site-packages`. Если не получается, введите такую команду:

   ```
   python -c 'import sys, pprint; pprint.pprint(sys.path)'
   ```

   В полученных результатах будет указано, в частности, местоположение каталога `site-packages`.

   В каталоге `site-packages` создайте файл `django.pth` и укажите в нем полный путь к своему каталогу `djtrunk`. Например, файл может содержать такую строку:

   ```
   /home/me/code/djtrunk
   ```

4. Включите каталог `djtrunk/django/bin` в переменную окружения `PATH`. В этом каталоге находятся утилиты управления, такие как `django-admin.py`.

Совет

Если вы раньше не встречались с pth-файлами, то можете прочитать о них на странице *http://www.djangoproject.com/r/python/site-module/*.

Если вы уже загрузили файлы из репозитория Subversion и выполнили описанные выше действия, то запускать команду `python setup.py` не нужно – все уже сделано!

Поскольку версия основной линии разработки Django часто изменяется в результате исправления ошибок и добавления новых возможностей, вам нужно будет время от времени обновлять ее. Чтобы обновить код, достаточно выполнить команду `svn update`, находясь в каталоге `djtrunk`. При этом клиент Subversion соединится с сервером *http://code.djangoproject.com*, проверит, появились ли какие-нибудь изменения, и включит в локальную копию все изменения, внесенные с момента последнего обновления. Все происходит автоматически и очень быстро.

Наконец, имея дело с версией основной линии разработки, вы должны уметь определять номер версии, с которой вы работаете в данный момент. Номер версии понадобится, если вы захотите обратиться к сообществу за помощью или предложить свои улучшения. В этом случае нужно будет сообщить номер используемой версии (он называется также *номером ревизии* или *набором изменений*). Чтобы узнать номер ревизии, введите команду `svn info`, находясь в каталоге `djtrunk`, и запишите число, стоящее после слова Revision. Этот номер увеличивается на единицу при каждом изменении Django, будь то исправление ошибки, добавление новой функции, обновление документации или еще что-то. В сообществе Django считается особым шиком сказать: «Я пользуюсь Django начиная с [какой-то низкий номер ревизии]».

Проверка установки Django

По завершении установки потратьте немного времени, чтобы проверить, нормально ли работает только что установленная система. Находясь в оболочке, перейдите в какой-нибудь каталог, не содержащий подкаталог `django`, и запустите интерактивный интерпретатор Python, введя команду `python`. Если установка прошла успешно, то вы сможете импортировать модуль `django`:

```
>>> import django
>>> django.VERSION
(1, 1, 0, 'final', 1)
```

Примеры работы с интерактивным интерпретатором

Интерактивный интерпретатор Python – это программа, позволяющая интерактивно исполнять команды на языке Python. Чтобы запустить ее, введите в оболочке команду `python`.

В этой книге часто будут встречаться сеансы работы с интерактивным интерпретатором Python. Распознать такие примеры можно по трем символам `>>>`, обозначающим приглашение интерпретатора. Если вы будете копировать примеры из книги, то эти символы следует опускать.

Инструкции, занимающие несколько строк в интерактивном интерпретаторе, начинаются с троеточия (...), например:

```
>>> print """Это
... строка, продолжающаяся
... на трех строчках."""
Это
строка, продолжающаяся
на трех строчках.
>>> def my_function(value):
...     print value
>>> my_function('Привет')
Привет
```

Троеточия в начале дополнительных строчек вставляет сам интерпретатор Python, они не являются частью выводимой информации. Мы включаем их, чтобы сохранить фактический вид сеанса работы с интерпретатором. Когда будете копировать примеры, эти точки следует опускать.

Настройка базы данных

Уже сейчас вы могли бы приступить к написанию веб-приложения на Django, поскольку единственное необходимое условие – наличие Python. Однако, скорее всего, вы все-таки будете разрабатывать сайт с базой данных, а потому придется настроить сервер базы данных.

Если вы хотите только поэкспериментировать с Django, то можете сразу перейти к разделу «Создание проекта», но имейте в виду, что во всех примерах этой книги предполагается наличие настроенной и действующей базы данных.

Фреймворк Django поддерживает четыре СУБД:

- PostgreSQL (*http://www.postgresql.org/*)
- SQLite 3 (*http://www.sqlite.org/*)
- MySQL (*http://www.mysql.com/*)
- Oracle (*http://www.oracle.com/*)

По большей части все они одинаково хорошо работают с ядром фреймворка Django. (Исключение составляет дополнительный модуль ГИС, который лучше всего использовать с PostgreSQL.) Если вы не связаны необходимостью поддерживать какую-то существовавшую ранее систему и можете выбирать СУБД по своему усмотрению, то мы рекомендуем PostgreSQL, которая обеспечивает великолепное сочетание стоимости, функциональности, быстродействия и стабильности.

Процедура настройки базы данных состоит из двух шагов.

1. Во-первых, необходимо установить и настроить сервер базы данных. Описание этого шага выходит за рамки настоящей книги, но на вебсайте любой из четырех СУБД имеется подробная документация. (Если ваш провайдер предоставляет виртуальный хостинг, то, скорее всего, сервер СУБД уже имеется.)
2. Во-вторых, необходимо установить библиотеку Python для выбранной СУБД. Это сторонний программный код, обеспечивающий интерфейс между Python и базой данных. В последующих разделах мы расскажем о том, что требуется для каждой СУБД.

Если вы просто знакомитесь с Django и не хотите устанавливать полноценную СУБД, то обратите внимание на продукт SQLite. Его уникальная особенность состоит в том, что при использовании версии Python 2.5 или выше предшествующие шаги вообще не нужны. Эта СУБД просто читает и записывает данные в единственный файл, а ее поддержка уже встроена в версию Python 2.5 и выше.

На платформе Windows найти двоичный дистрибутив драйвера СУБД может оказаться затруднительно. Если вам не терпится поскорее начать, то мы рекомендуем Python 2.5 со встроенной поддержкой SQLite.

Использование Django в сочетании с PostgreSQL

Для работы с PostgreSQL потребуется установить один из пакетов `psycopg` или `psycopg2` с сайта *http://www.djangoproject.com/r/python-pgsql/*. Мы рекомендуем `psycopg2`, так как он более новый, активно разрабатывается и проще в установке. В любом случае запомните, на какой версии остановились – 1 или 2, так как позже эта информация понадобится.

Если вы хотите работать с PostgreSQL на платформе Windows, то на странице *http://www.djangoproject.com/r/python-pgsql/windows/* вы сможете найти скомпилированные файлы `psycopg`.

При работе в Linux проверьте, есть ли в вашем дистрибутиве пакет `python-psycopg2`, `psycopg2-python`, `python-postgresql` или что-то подобное.

Использование Django в сочетании с SQLite 3

Если вы работаете с версией Python 2.5 или выше, считайте, что вам повезло; никакой специальной установки СУБД не потребуется, так как в Python уже встроена поддержка SQLite. Так что можете переходить к следующему разделу.

При работе с версией Python 2.4 или ниже потребуется скачать версию SQLite 3 – а *не* 2 – со страницы *http://www.djangoproject.com/r/sqlite/* и пакет `pysqlite` со страницы *http://www.djangoproject.com/r/python-sqlite/*. Версия `pysqlite` должна быть не ниже 2.0.3.

В Windows можно пропустить первый шаг (установку отдельного двоичного дистрибутива SQLite), поскольку код статически скомпонован с файлами пакета `pysqlite`.

В Linux проверьте, есть ли в вашем дистрибутиве пакет `python-sqlite3`, `sqlite-python`, `pysqlite` или нечто подобное.

Использование Django в сочетании с MySQL

Для Django требуется MySQL версии 4.0 или выше. В версиях 3.*x* не поддерживаются вложенные подзапросы и некоторые другие стандартные средства SQL.

Кроме того, потребуется установить пакет `MySQLdb` со страницы *http://www.djangoproject.com/r/python-mysql/*.

В Linux проверьте, есть ли в вашем дистрибутиве пакет `python-mysql`, `python-mysqldb`, `pysqlite`, `mysql-python` или нечто подобное.

Использование Django в сочетании с Oracle

Django работает с Oracle Database Server версии 9*i* и выше.

Для работы с Oracle потребуется установить библиотеку `cx_Oracle` со страницы *http://cx-oracle.sourceforge.net/*. Берите версию 4.3.1 или выше, но только не версию 5.0, поскольку в ней имеется ошибка.

Использование Django без СУБД

Как уже отмечалось, для работы Django база данных необязательна. Ничто не мешает использовать этот фреймворк для создания динамических страниц без обращения к базе.

Однако имейте в виду, что для некоторых дополнительных инструментов, поставляемых в комплекте с Django, база данных необходима, поэтому, отказываясь от нее, вы лишаете себя доступа к некоторым функциям. (О том, что это за функции, мы будем говорить ниже.)

Создание проекта

Установив Python, Django и (необязательно) СУБД и библиотеку для нее, можно приступать к разработке приложения Django. И первым шагом будет создание *проекта*.

Проект представляет собой набор параметров настройки отдельного экземпляра Django, в том числе параметров базы данных, параметров самого фреймворка и параметров приложения.

При первом использовании Django придется выполнить кое-какую начальную настройку. Создайте новый каталог, назвав его, скажем, `/home/username/djcode/`.

Где должен находиться этот каталог?

Если вы имеете опыт работы с PHP, то, наверное, привыкли помещать код непосредственно в корневой каталог документов веб-сервера (к примеру, `/var/www`). Но в Django так не делают. Помещать код на Python в корневой каталог веб-сервера опасно, потому что любой человек сможет просмотреть исходный текст вашего приложения через Интернет. Хорошего в этом мало.

Размещайте свой код в каталоге, находящемся за пределами корневого каталога документов.

Перейдите в созданный каталог и выполните команду `django-admin.py startproject mysite`. Она создаст каталог `mysite` в текущем каталоге.

Примечание

Если вы устанавливали Django с помощью прилагаемой к нему утилиты `setup.py`, то сценарий `django-admin.py` уже включен в системный список путей. При работе с версией основной линии разработки этот сценарий вы найдете в каталоге `djtrunk/django/bin`. Поскольку сценарий `django-admin.py` используется очень часто, имеет смысл добавить этот каталог в список путей. В UNIX для этого можно создать символическую ссылку на сценарий из каталога `/usr/local/bin` с помощью команды `sudo ln -s /path/to/django/bin/django-admin.py /usr/local/bin/django-admin.py`. В Windows нужно будет изменить переменную окружения `PATH`. Если фреймворк Django был установлен из пакета, входящего в состав дистрибутива Linux, то сценарий `django-admin.py` может называться `django-admin`.

Если при выполнении команды `django-admin.py startproject` вы увидите сообщение «permission denied», то нужно будет изменить разрешения на доступ к файлу. Для этого перейдите в каталог, где находится сценарий `django-admin.py` (например, `cd /usr/local/bin`), и выполните команду `chmod +x django-admin.py`.

Команда `startproject` создает каталог с четырьмя файлами:

```
mysite/
__init__.py
  manage.py
  settings.py
  urls.py
```

Опишем их назначение:

- `__init__.py`: этот файл необходим для того, чтобы Python рассматривал каталог `mysite` как пакет (группу Python-модулей). Этот файл пуст, и добавлять в него, как правило, ничего не требуется.

- `manage.py`: эта командная утилита позволяет различными способами взаимодействовать с проектом Django. Чтобы понять, что она умеет делать, введите команду `python manage.py`. Изменять этот файл не следует, он создан в каталоге проекта исключительно для удобства.
- `settings.py`: параметры настройки данного проекта Django. Загляните в файл, чтобы понять, какие вообще имеются параметры и каковы их значения по умолчанию.
- `urls.py`: URL-адреса для данного проекта Django. Это «оглавление» вашего сайта. Пока что оно пусто.

Несмотря на небольшой размер, эти файлы уже составляют работоспособное приложение Django.

Запуск сервера разработки

Чтобы лучше понять, что было сделано во время установки, давайте запустим сервер разработки Django и посмотрим, как работает заготовка приложения.

Сервер разработки Django – это встроенный упрощенный веб-сервер, которым можно пользоваться в ходе разработки сайта. Он включен в состав Django для того, чтобы можно было быстро создать сайт, не отвлекаясь на настройку полноценного сервера (например, Apache) до тех пор, пока разработка не будет завершена. Сервер разработки наблюдает за вашим кодом и автоматически перезагружает его, так что вам не нужно ничего перезапускать самостоятельно после внесения изменения в код.

Чтобы запустить сервер, перейдите в каталог проекта (`cd mysite`), если вы еще не находитесь в нем, и выполните команду:

```
python manage.py runserver
```

Она выведет примерно такой текст:

```
Validating models...
0 errors found.

Django version 1.0, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Теперь сервер запущен локально, прослушивает порт 8000 и принимает запросы на соединение только от вашего компьютера. Укажите URL *http://127.0.0.1:8000/* в адресной строке броузера. Вы увидите страницу «Welcome to Django» в приятных синих пастельных тонах. Работает!

Перед тем как продолжить, стоит сделать еще одно замечание о сервере разработки. Хотя на этапе разработки он очень удобен, не поддавайтесь искушению использовать его в среде, хотя бы отдаленно напоминающей производственную. Сервер разработки способен надежно обрабатывать

лишь один запрос в каждый момент времени и не проходил никакой аудиторской проверки на безопасность. Когда придет время запустить сайт в работу, обратитесь к главе 12, где рассказано о развертывании Django.

Изменение адреса или номера порта сервера разработки

По умолчанию команда `runserver` запускает сервер разработки на порту 8000 и принимает запросы на соединения только с локального компьютера. Чтобы изменить номер порта, укажите его в командной строке:

```
python manage.py runserver 8080
```

Задав также IP-адрес, вы разрешите серверу принимать запросы на соединение с другого компьютера. Это удобно, когда необходимо использовать сервер разработки совместно с другими членами команды. IP-адрес `0.0.0.0` разрешает серверу прослушивать все сетевые интерфейсы:

```
python manage.py runserver 0.0.0.0:8000
```

Теперь пользователь на любом компьютере в локальной сети сможет увидеть ваш Django-сайт, введя в адресной строке своего броузера ваш IP-адрес (например, *http://192.168.1.103:8000/*).

Чтобы узнать адрес своего компьютера в локальной сети, нужно вывести параметры настройки сети. В UNIX для этого достаточно выполнить команду `ifconfig`, в Windows – `ipconfig`.

Что дальше?

Теперь, когда все установлено и сервер разработки запущен, можно переходить к изучению заложенных в Django принципов обслуживания веб-страниц.

3

Представления и конфигурирование URL

В предыдущей главе мы рассказали о том, как создать проект Django и запустить сервер разработки. В этой главе мы начнем знакомиться с основами создания динамических веб-страниц в Django.

Первая страница, созданная в Django: Hello World

Для начала создадим веб-страницу, которая выводит пресловутое сообщение «Hello world».

Чтобы опубликовать такую страницу без помощи веб-фреймворка, достаточно просто ввести строку «Hello world» в текстовый файл, назвать его `hello.html` и сохранить в каком-нибудь каталоге на веб-сервере. Отметим, что при этом определяются два ключевых свойства страницы: ее содержимое (строка "`Hello world`") и URL (*http://www.example.com/hello.html* или, быть может, *http://www.example.com/files/hello.html*, если вы решили поместить файл в подкаталог).

При работе с Django вы определяете те же свойства, но по-другому. Содержимое страницы порождает *функция представления*, а URL задается в *настройках URL*. Сначала напишем функцию представления.

Ваше первое представление

В каталоге `mysite`, который был создан командой `django-admin.py` в предыдущей главе, создайте пустой файл с именем `views.py`. Этот Python-модуль будет содержать все представления, рассматриваемые в данной главе. Отметим, что в имени `views.py` нет ничего особенного – скоро мы увидим, что Django все равно, как называется этот файл, однако принято называть его именно `views.py`, чтобы другие разработчики, читающие ваш код, сразу понимали, что в нем находится.

Представление «Hello world» очень простое. Ниже приведен код функции вместе с командами импорта, который нужно поместить в файл views.py:

```
from django.http import HttpResponse

def hello(request):
    return HttpResponse("Hello world")
```

Рассмотрим его построчно.

- Сначала импортируется класс HttpResponse, который находится в модуле django.http. Импортировать его необходимо, потому что он используется в коде функции ниже.
- Далее определяется функция представления hello.
- Любая функция представления принимает по меньшей мере один параметр, который принято называть request. Это объект, содержащий информацию о текущем веб-запросе, в ответ на который была вызвана функция; он является экземпляром класса django.http.HttpRequest. В данном примере мы не используем параметр request, тем не менее он должен быть первым параметром представления.
- Отметим, что имя функции представления не имеет значения, фреймворк Django не предъявляет каких-либо специфических требований к именам. Мы назвали ее hello просто потому, что это имя ясно показывает назначение представления, но могли бы назвать hello_wonderful_beautiful_world или еще как-то. В следующем разделе будет показано, каким образом Django находит эту функцию.
- Сама функция состоит всего из одной строки: она просто возвращает объект HttpResponse, инициализированный строкой "Hello world".

Главный урок состоит в том, что представление – обычная функция на языке Python, которая принимает экземпляр класса HttpRequest в качестве первого параметра и возвращает экземпляр класса HttpResponse. Чтобы функция на Python могла считаться функцией представления, она должна обладать этими двумя свойствами. (Существуют исключения, но о них мы поговорим позже.)

Ваша первая конфигурация URL

Если сейчас снова выполнить команду python manage.py runserver, то появится сообщение «Welcome to Django» без каких бы то ни было следов представления «Hello world». Объясняется это тем, что проект mysite еще ничего не знает о представлении hello; необходимо явно сообщить Django, что при обращении к некоторому URL должно активироваться это представление. (Если продолжить аналогию с публикацией статических HTML-страниц, то сейчас мы только создали файл, но еще не загрузили его в каталог на сервере.) Чтобы связать функцию представления с URL, в Django используется механизм конфигурации URL.

Можно сказать, что *конфигурация URL* – это оглавление веб-сайта, созданного с помощью Django. По сути дела, речь идет об установлении соответствия между URL и функцией представления, которая должна вызываться при обращении к этому URL. Мы говорим Django: «Для этого адреса URL следует вызвать эту функцию, а для этого – эту». Например, «При обращении к URL `/foo/` следует вызвать функцию представления `foo_view()`, которая находится в Python-модуле `views.py`».

Во время выполнения команды `django-admin.py startproject` в предыдущей главе сценарий автоматически создал конфигурацию URL: файл `urls.py`. По умолчанию она выглядит следующим образом:

```
from django.conf.urls.defaults import *

# Раскомментировать следующие две строки для активации
# административного интерфейса:
# from django.contrib import admin
# admin.autodiscover()

urlpatterns = patterns('',
    # Пример:
    # (r'^mysite/', include('mysite.foo.urls')),

    # Раскомментировать строку admin/doc ниже и добавить
    # 'django.contrib.admindocs' в INSTALLED_APPS для активации
    # документации по административному интерфейсу:
    # (r'^admin/doc/', include('django.contrib.admindocs.urls')),

    # Раскомментировать следующую строку для активации
    # административного интерфейса:
    # (r'^admin/', include(admin.site.urls)),
)
```

В этой конфигурации URL по умолчанию некоторые часто используемые функции Django закомментированы, для их активации достаточно раскомментировать соответствующие строки. Если не обращать внимания на закомментированный код, то конфигурация URL сведется к следующему коду:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
)
```

Рассмотрим этот код построчно:

- В первой строке импортируются все объекты из модуля `django.conf.urls.defaults` поддержки механизма конфигурации URL. В частности, импортируется функция `patterns`.
- Во второй строке производится вызов функции `patterns`, а возвращенный ею результат сохраняется в переменной `urlpatterns`. Функции `patterns` передается всего один аргумент – пустая строка. (С ее