Программирование на Python: Часть 11. Web-программирование: Django

Сергей Яковлев
Консультант
независимый специалист

07.09.2010

Завершает цикл о языке программирования Python вводная статья о популярном фреймворке Django. Разумеется, все возможности этого инструмента нельзя описать в рамках одного материала, но говорить о Python и не упомянуть о Django автор счел невозможным.

Введение

Одной из наиболее фундаментальных архитектур для приложений является так называемая архитектура модель-представление-контроллер (Model-View-Controller) или MVC, которая разделяет базовый функционал приложения на ряд отдельных компонентов. При этом достигается главная цель: одна модель на много приложений.

Джанго – это высокоуровневый питоновский Web-фреймворк, который реализован на основе архитектуры MVC. Джанго имеет прозрачный дизайн, дает возможность для оперативной разработки Web-приложений, позволяет разрабатывать динамические Web-сайты.

Отличительные особенности джанго:

- любой запрос обрабатывается программно и перенаправляется на свой адрес(url);
- разделение контента и представления с помощью шаблонов;
- абстрагирование от низкого уровня баз данных.

Джанго-приложение состоит из четырех основных компонентов.

1. Модель данных: данные являются сердцевиной любого современного Web-приложения. Модель – важнейшая часть приложения, которое постоянно обращается к данным при любом запросе из любой сессии. Любая модель является стандартным питоновским классом. Объектно-ориентированный маппер (ORM) обеспечивает таким классам доступ непосредственно к базам данных. Если бы не было ORM, программисту пришлось бы писать запросы непосредственно на SQL. Модель обеспечивает облегченный механизм доступа к слою данных, инкапсулирует бизнес-логику. Модель

не зависит от конкретного приложения. Данными можно манипулировать даже из командной строки, не используя при этом Web-сервер.
2. Представление (view): вьюхи в джанге выполняют разнообразные функции, в том числе контролируют запросы пользователя, выдают контекст в зависимости от его роли. View – это обычная функция, которая вызывается в ответ на запрос какого-то адреса (url) и возвращает контекст.
3. Шаблоны: они являются формой представления данных. Шаблоны имеют свой собственный простой метаязык и являются одним из основных средств вывода на экран.
4. URL: это всего лишь механизм внешнего доступа к представлениям (view). Встроенные в урлы регулярные выражения делают механизм достаточно гибким. При этом одно представление может быть сконфигурировано к нескольким урлам, предоставляя доступ различным приложениям. Здесь поддерживается философия закладок: урлы становятся как бы самодостаточными и начинают жить независимо от представления.

Для новичков, которые впервые сталкиваются с джанго и питоном, может показаться, что тут используется какой-то особый метаязык. Но это не так. Все, что можно сделать в питоне, можно сделать и в джанго: тут есть доступ ко всем стандартным, и не только, питоновским библиотекам, плюс встроенный функционал джанго. Проекты, реализованные на джанго:

http://projects.washingtonpost.com/congress/

http://www.ljworld.com/

http://www.lawrence.com/

Сегодня мы рассмотрим следующие темы.

1. Как инсталлировать джанго.
2. Первое приложение.
3. Подключение базы данных.
4. Первое представление.
5. Шаблоны.
6. Администрирование.

1. Как инсталлировать джанго

Прежде всего, у вас должен быть установлен питон не ниже версии 2.3. Установить джанго можно по-разному: можно взять его из локального репозитария, можно взять официальный релиз – в этом случае ищите его тут:

http://www.djangoproject.com/download/.

После разархивации в каталоге проекта запустите команду:

```
setup.py install
```

Далее нужно сделать проверку, запустить питон и выполнить команду:

```
> import django
```

Проверка версии джанго:

```
> django.VERSION
```

На данный момент официальная стабильная версия – 1.1.1.

2. Первое приложение

Для создания первого приложения зайдем в свой домашний каталог и запустим команду:

```
  django-admin.py startproject mysite
```

Будет создан подкаталог mysite, где и будет лежать приложение. Заходим в созданный каталог и видим следующую файловую структуру:

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

__init__.py – пустой файл, который подключает текущий каталог как стандартный питоновский пакет.

manage.py – утилита, управляющая сайтом.

settings.py – конфигурация сайта.

urls.py – таблица урлов или таблица для всего контента сайта.

Для того чтобы загрузить наше Web-приложение, запустим команду:

```
 python manage.py runserver
```

Вывод:

```
Validating models...
0 errors found.
 Django version 1.1, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Если теперь в браузере открыть адрес http://127.0.0.1:8000/, то убедимся в том, что приложение запущено: появится стандартное приглашение.

3. Подключение базы данных

Джанго поддерживает следующие серверы баз данных: PostgreSQL, SQLite, MySQL, Microsoft SQL Server, Oracle.

В нашем примере мы выбираем постгрес – не потому, что он лучше, это не имеет принципиального значения. Убедитесь в том, что постгрес уже установлен на вашей системе, в противном случае его нужно проинсталлировать. После этого в файле settings.py пропишем настройки для постгреса:

```
DATABASE_ENGINE = 'postgresql_psycopg2'
DATABASE_NAME   = 'mysite'
DATABASE_USER   = 'postgres'
TIME_ZONE = 'Europe/Moscow'
LANGUAGE_CODE = 'ru-ru'
```

Запускаем две команды, которые создают базу данных mysite – постгрес перед этим, естественно, должен быть уже запущен:

```
psql -d template1 -U postgres -c "DROP DATABASE mysite;"
psql -d template1 -U postgres -c "CREATE DATABASE mysite
 WITH OWNER postgres ENCODING='UNICODE';"
```

Запускаем команду, которая создает в этой базе около 10 системных таблиц:

```
  python manage.py syncdb
```

4. Первое представление

Теперь сгенерируем каркас для нашего Web-приложения:

```
  python manage.py startapp People
```

Внутри каталога mysite появится подкаталог People со следующей структурой:

```
 People/
    __init__.py
    models.py
    tests.py
    views.py
```

Добавим в settings.py последнюю строку, которая добавляет новый путь:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'mysite.People'
)
```

В settings.py в качестве корневого адреса должен быть:

```
 ROOT_URLCONF = 'mysite.urls'
```

Создадим модель Person – стандартный питоновский класс с двумя атрибутами – имя, и-мейл. Файл models.py:

```
from django.db import models
class Person(models.Model):
    name = models.CharField('name', max_length=200)
    email = models.EmailField('Email', blank=True)
    def __str__(self):
        return '%s' % (self.name)
```

После того как добавляется новая модель, ее нужно синхронизировать с базой данных. Для этого из командной строки выполняем команду:

```
  python manage.py syncdb
```

После этого в базе данных появится новая таблица. Нужно заметить, что если вы после этого будете вносить изменения в уже существующую модель и пытаться синхронизировать эти изменения с базой данных, то джанго никак не будет на это реагировать: в базу добавляются только новые объекты. Для внесения изменений уже придется пользоваться sql-скриптами на апдэйт конкретных таблиц.

Теперь напишем простейшее представление, которое выведет приглашение-файл views.py:

```
from django.shortcuts import HttpResponse
from mysite.People.models import Person
def index(request):
    html = "<H1>People !!!</H1><HR>"
    return HttpResponse(html)
```

Добавим урл – файл urls.py:

```
from django.conf.urls.defaults import *
urlpatterns = patterns('',
    (r'^People/$', 'mysite.People.views.index')
)
```

Вновь запускаем Web-приложение:

```
  python manage.py runserver
```

Открываем адрес в браузере и видим приглашение:

```
  http://127.0.0.1:8000/People/
```

5. Шаблоны

В корне каталога mysite создадим каталог templates. В файле settings.py добавим строку:

```
TEMPLATE_DIRS = (
     "/home/mysite/templates"
)
```

Добавленный путь должен быть абсолютным. В добавленный каталог положим файл person.htnl следующего содержания:

```
<H1>People !!!</H1><HR>
<table width=100%>
<tr><td colspan="3">
Personal Information
</td></tr>
<tr valign="top"><td width=30%>
    <li>Name: {{p.name}}</li>
</td>
<td width=30%>
    <li>Email: {{ p.email }}</li>
</td></tr>
</table>
```

В нашем единственном представлении index мы сделаем изменения: создадим объект Person и передадим его в вызываемый шаблон в качестве параметра. В шаблоне мы берем атрибуты этого параметра с помощью префикса двойных кавычек:

```
  {{p.name}}
  {{p.email}}
```

Файл views.py после подключения шаблона теперь будет выглядеть так:

```
from mysite.People.models import Person
from django.shortcuts import render_to_response, get_object_or_404

def index(request):
    person = Person()
    person.name = 'Alex'
    person.email = 'Alex@gmail.com'
    return render_to_response('person.html', {'p': person})
```

После перезагрузки Web-страницы в браузере мы увидим эти атрибуты.

6. Администрирование

Администраторская часть в джанго по умолчанию отключена. Для ее включения добавим последнюю строку в settings.py:

```
 INSTALLED_APPS = (
     'django.contrib.auth',
     'django.contrib.contenttypes',
     'django.contrib.sessions',
     'django.contrib.sites',
     'mysite.People',
     'django.contrib.admin'
 )
```

Для работы с админской частью нужно иметь рутовые права, поэтому нужно вначале создать суперпользователя:

```
  python manage.py createsuperuser
```

После этого запустим команду:

```
 python manage.py syncdb
```

Файл urls.py с добавленным админским функционалом будет иметь следующий вид:

```
from django.conf.urls.defaults import *
from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    (r'^admin/', include(admin.site.urls)),
    (r'^People/$', 'mysite.People.views.index')
)
```

Запускаем Web-приложение:

```
   python manage.py runserver
```

Набираем в браузере адрес: http://127.0.0.1:8000/admin/. И попадаем на страницу логина. После ввода логина и пароля суперпользователя открывается админская часть. В ней можно добавлять и редактировать объекты.

Заключение

Итак, сегодня мы узнали, что такое Джанго и почему стоит остановить на нем свой выбор. Это бесплатный фреймворк, который поддерживает модель MVC, имеет несколько программных интерфейсов к различным базам данных, интуитивно понятный интерфейс администрирования с возможностью расширения, поддержку многоязычности и т.д. Этот Web-фреймворк обладает рядом преимуществ: джанго хорошо документирован – сайт разработчиков http://www.djangoproject.com/ тому подтверждение. Легкость установки и настройки помогают значительно экономить время. Малое число внешних зависимостей удобно для пользования: все, что нам нужно – это питон. На этом наш цикл статей можно считать завершенным. Конечно, автор не ставил перед собой задачи написать подробный учебник или справочник, скорее здесь речь идет о кратком вводном курсе для разработчиков, желающих освоить новый для себя инструмент.

Код примеров проверялся на версии питона 2.6.

<< Предыдущая статья.

Об авторе

Сергей Яковлев

Яковлев Сергей — независимый разработчик с многолетним опытом прикладного и системного программирования; вносит вклад в развитие open-source на своем персональном сайте www.iakovlev.org. Консультант.