Python Web Framework

Balakumar Parameshwaran

24-10-2013

http://www.balakumarp.com

Agenda

- Introduction
- Features
- Installation
- Django Architecture
- Project structure
- Settings
- Project / Site creation
- URL Dispatcher
- Who uses it?
- Questions

Introduction

Django is a free and open source web application framework, written in Python, which follows the Model–View–Controller architectural pattern.

It is maintained by the Django Software Foundation (DSF), an independent organization.

- Encourages rapid development and clean, pragmatic design.
- Named after famous Guitarist Django Reinhardt
- Developed by Adrian Holovaty & Jacob Kaplan-moss
- Created in 2003, open sourced in 2005
- 1.0 Version released in Sep 3 2008, now 1.5.4

Features

- Object Relational Mapper - ORM
- MVC (MVT) Architecture
- Focuses on automating as much as possible and adhering to the DRY principle
- Template System
- Out of the box customizable Admin Interface, makes CRUD easy
- Built-in light weight Web Server
- Elegant URL design
- Custom Middleware
- Authentication / Authorization
- Internationalization support
- Cache framework, with multiple cache mechanisms
- Fast Development
- Free, and Great Documentation

Installation

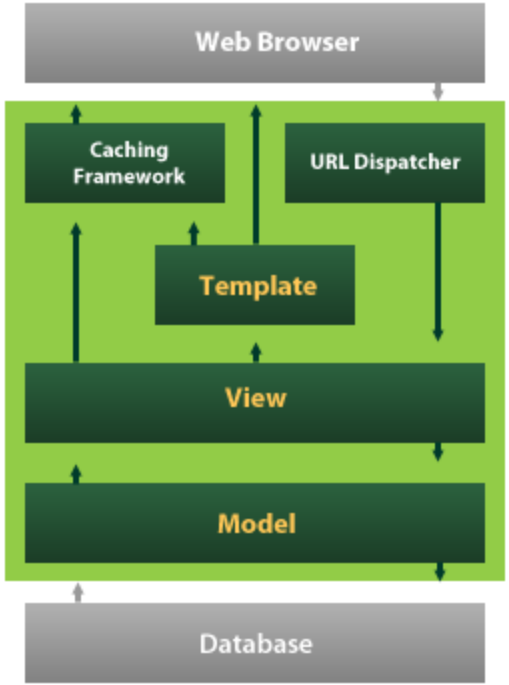
Prerequisites

- ✓ Python
- ✓ PIP for installing Python packages (http://www.pip-installer.org/en/latest/installing.html)

- ➢ pip install Django==1.5.4
  - OR https://www.djangoproject.com/download/ - python setup.py install
- ➢ pip install mysql-python
  - MySQL on windows https://pypi.python.org/pypi/MySQL-python/1.2.4
- ➢ Add Python and Django to env path
  - PYTHONPATH D:\Python27
  - Path D:\Python27; D:\Python27\Lib\site-packages; D:\Python27\Lib\site-packages\django\bin;
- ➢ Testing installation
  - shell> import django; django.VERSION;

Django Architecture

Models	Describes your data
Views	Controls what users sees
Templates	How user sees it
Controller	URL dispatcher



1. The URL dispatcher (urls.py) maps the requested URL to a view function and calls it. If caching is enabled, the view function can check to see if a cached version of the page exists and bypass all further steps, returning the cached version, instead. Note that this page-level caching is only one available caching option in Django. You can cache more granularly, as well.
2. The view function (usually in views.py) performs the requested action, which typically involves reading or writing to the database. It may include other tasks, as well.
3. The model (usually in models.py) defines the data in Python and interacts with it. Although typically contained in a relational database (MySQL, PostgreSQL, SQLite, etc.), other data storage mechanisims are possible as well (XML, text files, LDAP, etc.).
4. After performing any requested tasks, the view returns an HTTP response object (usually after passing the data through a template) to the web browser. Optionally, the view can save a version of the HTTP response object in the caching system for a specified length of time.
5. Templates typically return HTML pages. The Django template language offers HTML authors a simple-to-learn syntax while providing all the power needed for presentation logic.

Project Directory Structure

- demosite/ — Just a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.
  - manage.py — A command-line utility that lets you interact with this Django project in various ways. Type python manage.py help. You should never have to edit this file.
  - demosite/ — Actual Python package for your project. Use this name to import anything inside it (e.g. import demosite.settings)
    - __init__.py — A file required for Python to treat the demosite directory as a package.
    - settings.py — Settings/configuration for this Django project
    - urls.py — Root URL config, the URLs for this Django project, provides mapping to views
    - wsgi.py — An entry-point for WSGI-compatible webservers to serve your project
    - templates/ — HTML files , renders based on views. You can change to any dir, configurable in settings.py
    - static/ — CSS, JS, images.. etc, configurable in settings.py
    - demoapp/
      - __init__.py
      - urls.py
      - views.py — Responsible for processing a user's request and for returning the response
      - models.py — A model is the single, definitive source of information about your data. Generally, each model maps to a single database table.
      - admin.py — It reads metadata in your model to provide a powerful and production-ready interface
      - forms.py — To create and manipulate form data

Settings

- Project settings.py overrides from <python>/Lib/site-packgaes/django/conf/global_settings.py
- Set DJANGO_SETTINGS_MODULE for your Project, tells django which settings to be used. (demoproject.settings)
  - export/set DJANGO_SETTINGS_MODULE=demoproject.settings
  - For server mod_wsgi: os.environ['DJANGO_SETTINGS_MODULE'] = 'demoproject.settings'
- DEBUG True or False
- DATABASES ENGINE postgresql_psycopg2', 'mysql', 'sqlite3' or 'oracle'.. etc
- ROOT_URLCONF
- MEDIA_ROOT directory that will hold user-uploaded files
- MEDIA_URL To serve media files
- STATIC_ROOT To any server static files css, js.. and admin UI files (can add more dirs to STATICFILES_DIRS)
- STATIC_URL To serve static files
- TEMPLATE_DIRS Template directories

Using settings in Python code

```
from django.conf import settings
if settings.DEBUG:
    # Do something
```

Project / Site Creation

- Creating new Project
  - django-admin.py startproject demoproject

  A project is a collection of applications
- Creating new Application
  - python manage.py startapp demosite

  An application tries to provide a single, relatively self-contained set of related functions
- Using the built-in web server
  - python manage.py runserver
  - python manage.py runserver 80

  Runs by default at port 8000

  It checks for any error and validate the models. Throws errors/warnings for any misconfigurations and invalid entries.

URL Dispatcher / Patterns

- Root URL should be configured in settings.py
  - ROOT_URLCONF = 'app.urls'
- Syntax

```
patterns(prefix,
    (regular expression, Python callback function [, optional dictionary [, optional name]])
)
```

Example:

```
urlpatterns = patterns(' ',
     (r'^articles-year/$', 'mysite.news.views.articles_year'),
 )
```

Note:

- No need to add a leading slash (/articles-year)
- The 'r' in front of each regular expression string is optional but recommended. It tells Python that a string is "raw" -- that nothing in the string should be escaped.

In python, the '\' backslash character in control chars must be escaped for regular expression use. Basically we have to add one more slash i.e \\t, \\b. To work around backslash plague, you can raw string, by prefixing the string with the letter r.

- Can include other URLconf modules

```
urlpatterns = patterns(' ',
      url(r'^support/', include('demoproject.support.urls')),
)
```

- Using Prefix

```
urlpatterns = patterns(' ',
    (r'^articles/(\d{4})/$', 'mysite.news.views.articles_year'),
    (r'^articles/(\d{4})/(\d{2})/$', 'mysite.news.views.articles_month'),
)
```

Here mysite.news.views is common, so can be rewritten as follows

```
urlpatterns = patterns('mysite.news.views',
    (r'^articles/(\d{4})/$', 'articles_year'),
    (r'^articles/(\d{4})/(\d{2})/$', 'articles_month'),
)
```

- Passing extra arguments and Dictionary mapping

```
patterns(' ',
     (r'^articles/(?P<year>\d{4})/$', 'articles_year'), {'foo': 'bar'}),
)
```

We can get the values in views.py as year='2005', foo='bar'

Who uses Django?

- BitBucket
- DISQUS (serving 400 million people)
- Pinterest
- Instagram
- dPaste
- Mozilla (support.mozill, addons.mozilla)
- NASA
- PBS (Public Broadcasting Service)
- The Washington Post, NY Times, LA Times, The Guardian
- National Geographic, Discovery Channel