Django Basics

Part II

Greg (mastergreg) Liras, John (nemo) Giannelos

FOSS NTUA

April 6, 2012

Outline

What is Django?

- Django is a web-framework written in Python

What is Django?

- Django is a web-framework written in Python
- It encourages rapid development and clean pragmatic design

What is Django?

- Django is a web-framework written in Python
- It encourages rapid development and clean pragmatic design
- Follows the MVC (Model-View-Controller) architecture

What is Django?

- Django is a web-framework written in Python
- It encourages rapid development and clean pragmatic design
- Follows the MVC (Model-View-Controller) architecture
- Focuses on automating common actions (DRY principle)

What is Django?

- Django is a web-framework written in Python
- It encourages rapid development and clean pragmatic design
- Follows the MVC (Model-View-Controller) architecture
- Focuses on automating common actions (DRY principle)
- Allows the use of pluggable applications

But first of all...

What is a web-framework?

What is a web-framework?

As quoted from wikipedia:

> *A web application framework is a software framework that is designed to support the development of dynamic websites, web applications and web services. The framework aims to alleviate the overhead associated with common activities performed in Web development.*

- At first, WWW was only a bunch of hardcoded HTML pages.

- At first, WWW was only a bunch of hardcoded HTML pages.
- Need for more and more dynamic content and user interaction.

- At first, WWW was only a bunch of hardcoded HTML pages.
- Need for more and more dynamic content and user interaction.
- New languages designed specifically for web use.

- At first, WWW was only a bunch of hardcoded HTML pages.
- Need for more and more dynamic content and user interaction.
- New languages designed specifically for web use.
- Common tasks-practices showed up in web development.

- At first, WWW was only a bunch of hardcoded HTML pages.
- Need for more and more dynamic content and user interaction.
- New languages designed specifically for web use.
- Common tasks-practices showed up in web development.

As a result, complete software-stacks appeared to support the web development process. Think of it as a collection of libraries that do common tasks in web development and are designed to work together.

The MVC architecture

- MVC stands for Model-View-Controller

The MVC architecture

- MVC stands for Model-View-Controller
- Seperates the modeling of the domain, the presentation and the logic/actions of the project, permitting independent development of each.

The MVC architecture

- MVC stands for Model-View-Controller
- Seperates the modeling of the domain, the presentation and the logic/actions of the project, permitting independent development of each.
  - Model: Database and data logic/behaviour

The MVC architecture

- MVC stands for Model-View-Controller
- Seperates the modeling of the domain, the presentation and the logic/actions of the project, permitting independent development of each.
  - Model: Database and data logic/behaviour
  - View: (What it says), manages what user views

The MVC architecture

- MVC stands for Model-View-Controller
- Seperates the modeling of the domain, the presentation and the logic/actions of the project, permitting independent development of each.
  - Model: Database and data logic/behaviour
  - View: (What it says), manages what user views
  - Controller: Interpretation of user's actions

The MVC architecture

- MVC stands for Model-View-Controller
- Seperates the modeling of the domain, the presentation and the logic/actions of the project, permitting independent development of each.
  - Model: Database and data logic/behaviour
  - View: (What it says), manages what user views
  - Controller: Interpretation of user's actions
- Django implements MVC slightly different

Django's MVC

As mentioned before, in the Django framework things are a little bit different.

Welcome to "MTV" (model-template-view).

Django's MVC

- Model: Pretty much the same as MVC. Describes our database using ORM (object relation mapper).

Django's MVC

- Model: Pretty much the same as MVC. Describes our database using ORM (object relation mapper).
- Template: Seperates our website content from its presentation "style". To achieve this we use a templating language embedded in our HTML.

Django's MVC

- Model: Pretty much the same as MVC. Describes our database using ORM (object relation mapper).
- Template: Seperates our website content from its presentation "style". To achieve this we use a templating language embedded in our HTML.
- View: Describes which data should be presented (decided using appropriate functions) and sends them to our template. It also takes care of the url handling.

Django's MVC

- Model: Pretty much the same as MVC. Describes our database using ORM (object relation mapper).
- Template: Seperates our website content from its presentation "style". To achieve this we use a templating language embedded in our HTML.
- View: Describes which data should be presented (decided using appropriate functions) and sends them to our template. It also takes care of the url handling.
  - According to Django's FAQ page:

Django's MVC

- Model: Pretty much the same as MVC. Describes our database using ORM (object relation mapper).
- Template: Seperates our website content from its presentation "style". To achieve this we use a templating language embedded in our HTML.
- View: Describes which data should be presented (decided using appropriate functions) and sends them to our template. It also takes care of the url handling.
  - According to Django's FAQ page:

    ...a view is the Python callback function for a particular URL, because that callback function describes which data is presented.

Django admin-site

Django has a built-in website for various (common) administrative jobs. It allows you:

- To manage site users
- To change form display and handling
- To make various validations (add,remove,publish *e.g* *"articles"*)

Django snippets

Object Relational Mapper example:

```
from django.db import models

class Publisher(models.Model):
    name = models.CharField(maxlength=30)
    address = models.CharField(maxlength=50)
    city = models.CharField(maxlength=60)
    state_province = models.CharField(maxlength=30)
    country = models.CharField(maxlength=50)
    website = models.URLField()
```

Django snippets

URL dispatcher example:

```
from django.conf.urls import patterns, url, include

urlpatterns = patterns
    ('',
    (r'^articles/2003/$', 'news.views.special_case_2003'),
    (r'^articles/(\d{4})/$', 'news.views.year_archive'),
    (r'^articles/(\d{4})/(\d{2})/$', 'news.views.month_archive'),
    (r'^articles/(\d{4})/(\d{2})/(\d+)/$', 'news.views.article_detail'),)
```

Django snippets

Simple views example:

```
from django.http import HttpResponse
import datetime

def current_datetime(request):
    now = datetime.datetime.now()
    html = "<html><body>It is now %s.</body></html>" % now
    return HttpResponse(html)

def hours_ahead(request, x):
    x = int(offset)
    dt = datetime.datetime.now() + datetime.timedelta(hours=x)
    html ="<html><body>In %s hour(s), it will be %s.</body></html>" % (x, dt)
    return HttpResponse(html)
```

Django snippets

HTML templates example:

```
<html>
<head><title>Ordering notice</title></head>
  <body>
    <p>Dear {{ person_name }},</p>
    <p>Here are the items you have ordered:</p>
    <ul>
      {% for item in item_list %}
      <li>{{ item }}</li>
      {% endfor %}
    </ul>
    {% if ordered_warranty %}
    <p>Your warranty information will be included in the packaging.</p>
    {% endif %}
  </body>
</html>
```

References

- Django Documentation: `https://docs.djangoproject.com/`
- The Django Book: `http://www.djangobook.com/`
- Django community (IRC,mailing lists): `https://www.djangoproject.com/community/`
- Python Package Index: `http://pypi.python.org/`