Web Development

Using Django


django

- Introduction
- Installation
- Setting up Environment
- Starting Django Project
- Using Models
- Writing Views
- Templates
- References
- Other Web Tools

Introduction

“Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.”

Introduction (con't)

"Django is a high-level Python Web framework…"

Introduction (con't)

"…that encourages rapid development …"

Introduction (con't)

"… and clean, pragmatic design."

Introduction (con't)

- What is Django?
  - A high-level Python Web framework
  - Uses a Model-View-Template (MVT) pattern
  - Not programming language
  - Not webserver
- Why Django?
  - Easy to learn

Introduction (con't)

More information about Django philosophy…

- - https://docs.djangoproject.com/en/dev/misc/design-philosophies/

Use DRY (Don't Repeat Yourself) principles

- http://c2.com/cgi/wiki?DontRepeatYourself

Installation

- See..
  - https://docs.djangoproject.com/en/1.6/intro/install/
- Prerequistics
  - Virtualenv, text editor, Ipython Notebook(?)
- Use Virtual environment
  - Sandboxed python environmen
  - Packages you install are not viewable by system
  - Multiple virtualenvs isloated separate projects

Setting Up Your Environment

- Create a Clean Workspace

```
$ mkdir tutorial
$ virtualenv ./tutorial/
New python executable in ./tutorial/bin/python
Installing setuptools, pip...done.Setuptools (easy_install)
$ source ./tutorial/bin/activate
(tutorial)$
```

- Install Django and Ipython in vitrualenv

```
(tutorial)$ pip install django
Downloading/unpacking django
  Downloading Django-1.6.5-py2.py3-none-any.whl (6.7MB): 6.7MB downloaded
Installing collected packages: django
Successfully installed django
Cleaning up...
(tutorial)$ pip install ipython
…
```

Starting Django Project

- Creating a Project

```
(tutorial)$ django-admin.py startproject addressbook

<!-- Check your directory list using your own command line -->
addressbook/
     manage.py
     addressbook/
          __init__.py
          settings.py
          urls.py
          wsgi.py
```

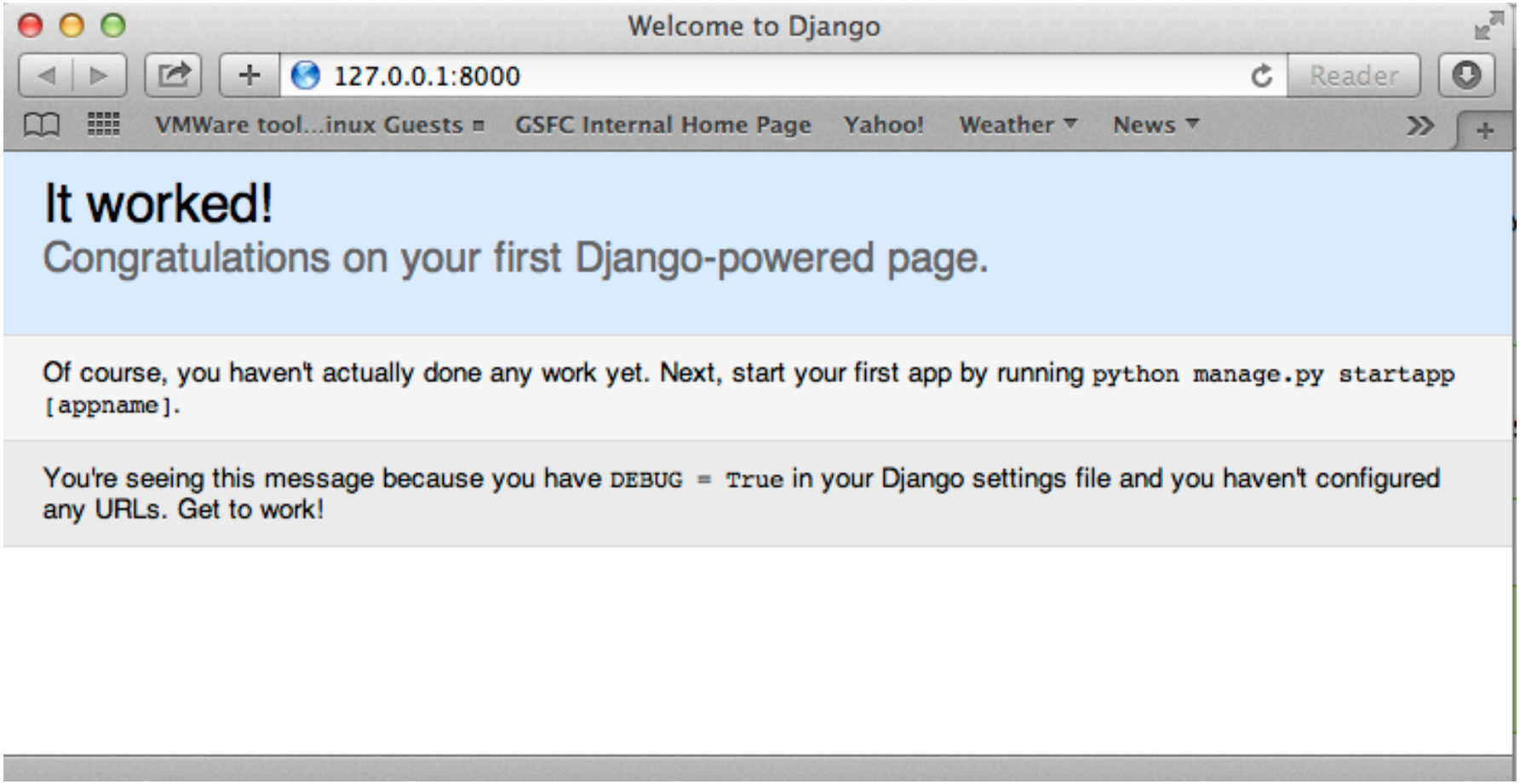
- Project Layout
  - manage.py — A command-line utility for interacting your project
  - __init__.py — An empty file that can be considered a Python package
  - settings.py — Settings/configuration for this project
  - urls.py: — The URL declaration for this Django project
  - wsgi.py — An entry-point for WSGI-compatible web servers

Starting Django Project (con't)

- Running the development server

```
(tutorial)$ cd addressbook
(tutorial)$ python manage.py runserver
Validating models...

0 errors found
May 22, 2014 - 15:57:35
Django version 1.6.5, using settings 'addressbook.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Starting Django Project (cont'd)



Using Models

- Configuring the Database
  - Edit `addressbook/settings.py` for DATABASE definition

```
# Database
# https://docs.djangoproject.com/en/1.6/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

  - If you don't want to use SQLite3, you can set other database.
    - MySQL, PostgreSQL, Oracle
    - See more information at the Django Web site
- Setting Time zone by editing that file

```
TIME_ZONE = 'America/New_York'
```

Using Models (cont'd)

- Run syncdb

```
(tutorial)$ python manage.py syncdb
Creating tables ...
Creating table django_admin_log
Creating table auth_permission
Creating table auth_group_permissions
Creating table auth_group
Creating table auth_user_groups
Creating table auth_user_user_permissions
Creating table auth_user
Creating table django_content_type
Creating table django_session

You just installed Django's auth system, which means you don't have any superusers
defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use '<username>'):
Email address: <your_email>
Password:
Password (again):
Superuser created successfully.
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
```

Using Models (cont'd)

- After syncdb using SQLite3,
  - The file 'db.sqlite3' is created in the `addressbook` top-level directory

```
./addressbook
./contacts
db.sqlite3 – (new SQLite3 database file)
manage.py
```

Using Models (cont'd)

- Difference between projects and apps
  - An app is a web applications
  - A project is a collection of configuration

- Creating a Contact "app"

```
(tutorial)$ python manage.py startapp contacts

<!— list of created directory and files -->
manage.py
./addressbook    (Already showed it)
./contacts       (Created new <app> directory)
     __init__.py
     admin.py
     models.py
     tests.py
     views.py
```

Using Models (cont'd)

- Edit `contacts/models.py`

```
from django.db import models


class Contact(models.Model):

    first_name = models.CharField(max_length=25)
    last_name = models.Charfield(max_length=50)
    email = models.EmailField()

    def __str__(self):
        return ' '.join([self.first_name, self.last_name])
```

Using Models (cont'd)

- Run syncdb

```
(tutorial)$ python manage.py syncdb
Creating tables ...
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
(tutorial)$
```

- 'contact' table?

Using Models (cont'd)

- Edit `addressbook/settings.py` for INSTALLED_APPS

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'contacts',
)
```

- Run syncdb again

```
(tutorial)$ python manage.py syncdb
Creating tables ...
Creating table contacts_contact
Installing custom SQL ...
Installing indexes ...
Installed 0 object(s) from 0 fixture(s)
(tutorial)$
```

Using Models (cont'd)

- Interacting the Model

```
(tutorial)$ python manage.py shell
…
In [1]: from contacts.models import Contact
In [2]: Contact.objects.all()
Out[2]: []
In [3]: Contact.objects.create(first_name='John', last_name='Doe')
Out[3]: <Contact: John Doe>
In [4]: Contact.objects.all()
Out[4]: [<Contact: John Doe>]
In [5]: john = Contact.objects.get(first_name='John')
In [6]: john
Out[6]: <Contact: John Doe>
In [7]: print john
John Doe
In [8]: john.id
Out[8]: 1
```

Using Models (cont'd)

- Writing the Test
  - Edit `contacts/tests.py`

```
from django.test import TestCase
from contacts.models import Contact


class ContactTests(TestCase):
    """Contact model tests."""

    def test_str(self):
        contact = Contact(first_name='John', last_name='Smith')
        self.assertEquals(str(contact), 'John Smith')
```

Using Models (cont'd)

- Running the Test

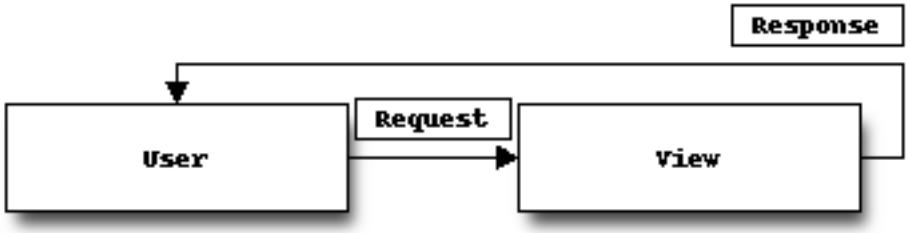
```
(tutorial)$ python manage.py test contacts
Creating test database for alias 'default'...
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
Destroying test database for alias 'default'...
(tutorial)$
```

Using Models (cont'd)

- Review
  - Models define the fields in a table, and can contain business logic.
  - The syncdb manage command creates the tables in your database from models
  - The model Manager allows you to operate on the collection of instances: querying, creating, etc.
  - Write unit tests for methods you add to the model
  - The test manage command runs the unit tests

Writing Views

- View Basics
  - Take HTTP Request
  - Return HTTP Response
  - Edit `contacts/views.py`




```
from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello world.  You are at the contact index")
```

Writing Views (cont'd)

- Defining URLs
  - Use `urlpattern` in the file 'urls.py'
  - Edit `contacts/urls.py` and `addressbook/urls.py`
  - Run the development server using `http://127.0.0.1/contacts`

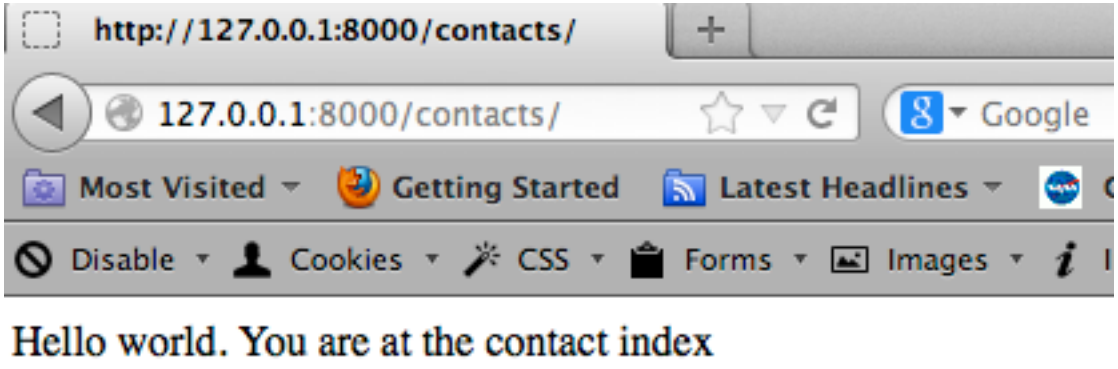
Writing Views (cont'd)

```
# contacts/urls.py

from django.conf.urls import patterns, url
from contacts import views


urlpatterns = patterns('',
    url(r'^$', views.index, name='index')
)
```

```
# addressbook/urls.py

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

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    url(r'^contacts/', include('contacts.urls')),
    url(r'^admin/', include(admin.site.urls)),
)
```

Writing Views (cont'd)

```
(tutorial)$ python manage.py runserver
Validating models...

0 errors found
May 23, 2014 - 12:33:31
Django version 1.6.5, using settings 'addressbook.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```



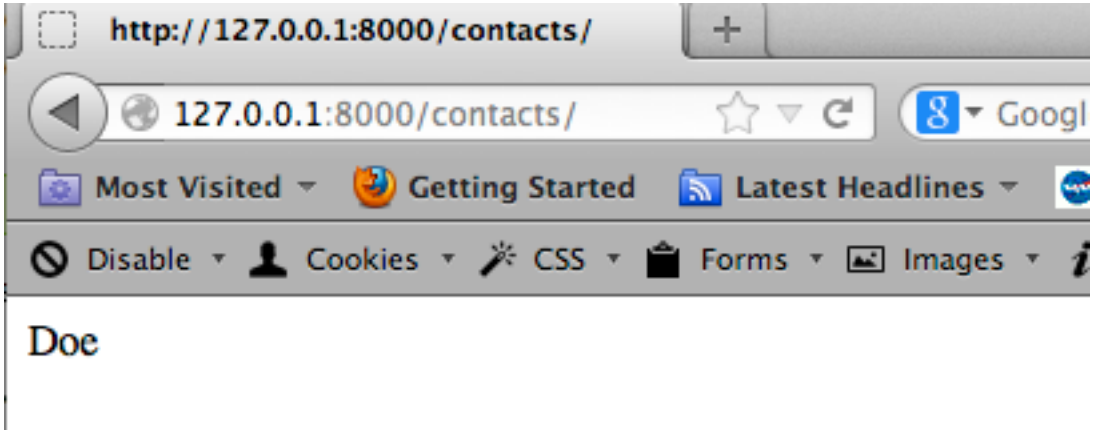
Writing Views (cont'd)

- Writing more view…
  - Edit `contacts/views.py`
  - Re-run the development server to verify

```
from django.http import HttpResponse

from contacts.models import Contact

def index(request):
    contact_list = Contact.objects.all()
    output = ', '.join([c.last_name for c in contact_list])
    return HttpResponse(output)
```

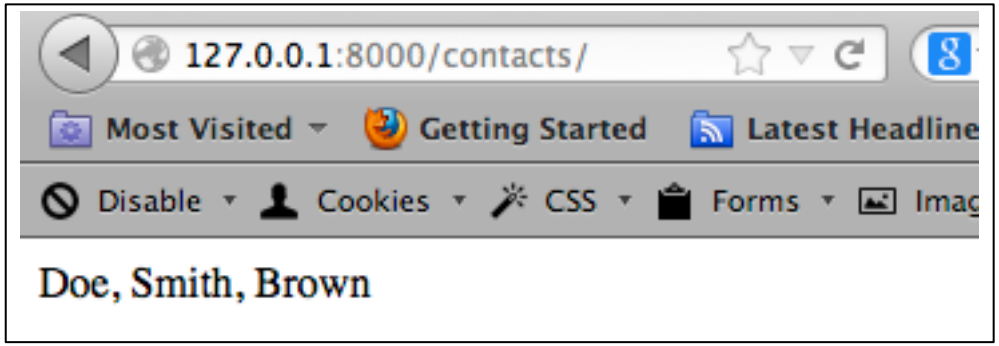
Writing Views (cont'd)

```
(tutorial)$ python manage.py runserver
Validating models...

0 errors found
May 23, 2014 - 12:33:31
Django version 1.6.5, using settings 'addressbook.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```



Writing Views (cont'd)

```
(tutorial)$ python manage.py shell

In [1]: from contacts.models import Contact

In [2]: Contact.objects.create(first_name='Jane', last_name='Smith')
Out[2]: <Contact: Jane Smith>

In [3]: Contact.objects.all()
Out[3]: [<Contact: John Doe>, <Contact: Jane Smith>]

In [4]: Contact.objects.create(first_name='Bob', last_name='Brown')
Out[4]: <Contact: Bob Brown>

In [5]: Contact.objects.all()
Out[5]: [<Contact: John Doe>, <Contact: Jane Smith>, <Contact: Bob Brown>]
```



Templates

- Customizing project's templates
  - Edit `addressbook/settings.py` to add this

```
TEMPLATE_DIRS = [os.path.join(BASE_DIR, 'templates')]
```

  - Create new templates directories and file (see next slide)
  - Update contacts/views.py for using the templates (see next two slides)
  - Run the development server again to verify

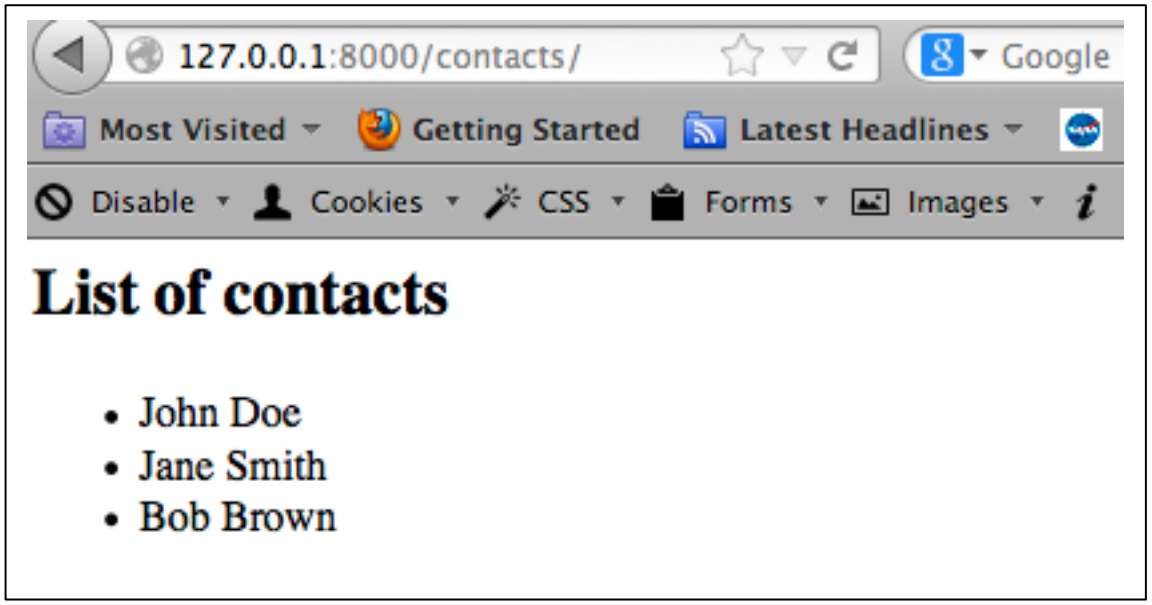
Templates (cont'd)

```
(tutorial)$ cd contacts/
(tutorial)$ mkdir templates
(tutorial)$ mkdir templates/contacts
(tutorial)$ cd templates/contacts/
(tutorial)$ <#   Edit the new file 'index.html'  #>
```

```
{% if contact_list %}
   <h2>List of contacts</h2>
   <ul>
   {% for person in contact_list %}
      <li>{{ person }}</li>
   {% endfor %}
   </ul>
{% else %}
   <p>No contacts are available.</p>
{% endif %}
```

Templates (cont'd)

```
from django.shortcuts import render
from django.http import HttpResponse

from contacts.models import Contact

def index(request):
    contact_list = Contact.objects.all()
    context = {'contact_list': contact_list}
    return render(request, 'contacts/index.html', context)
```



That’s it for Django

- We are done for the simple Django application.
- You can learn more from the Django Web Site!

References

- http://www.djangoprojects.com/
- http://www.effectivedjango.com/

Other Web development tools?

- **Python's own CGI**
- **Tornado**
  - **http://www.tornadoweb.org/en/stable/**
- **Google App Engine SDK for Python**
  - https://developers.google.com/appengine/docs/python/gettingstartedpython27/introduction
- **See more?**
  - https://wiki.python.org/moin/WebFrameworks

Backup slides….

Using IPython Notebook with Django

- IPython has a relatively new featured called the "Notebook," which improves on the traditional terminal shell in many ways.
- Notebook launches a web-based shell to an IPython session that has some very, very handy features, like the ability to save, edit and delete "notebooks" of code that are each comprised of organized cells of Python, text or Markdown. You can move the cells around, developing code interactively with documentation and notes to yourself, displaying anything that a browser could render: images, HTML, etc.
- But! You can't run a Django shell using notebook.

With Django Extensions

- The latest version of the Django Extensions app on Github has support for using the shell_plus command with Notebook. If you're up to date, you should be able to use the following command to run a Django shell with Notebook:
- $ ./manage.py shell_plus --notebook