Web Programming in Python with Django

IAP 2013

Student Information Processing Board

Luke O’Malley ‘14

Overview

Class Material
mit.edu/omalley1/django

Install Django

SIPB: Django Web Programming, IAP 2013

```
<your terminal>$ pip install django
```

or

```
<your terminal>$ easy_install django
```

or

```
<your terminal>$ apt-get install python-django
```

Websites and Their Pieces

Front-end:

- Broadly, it is what the user interacts with
- Where data is entered and displayed
- Often sends information to backend for processing and storage

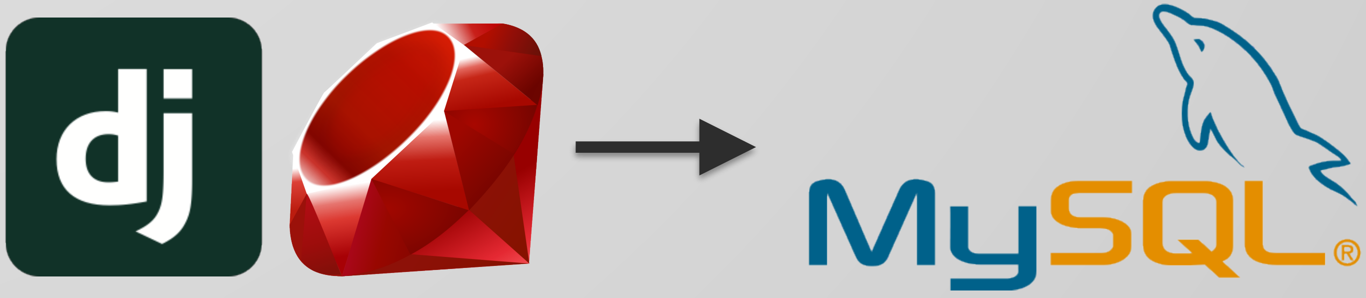
jQuery
write less, do more.

HTML

CSS

Back-end:

- Broadly, it receives data from front-end and processes and stores it
- Responsible for “serving” content
- Often composed of a database and management layer



frameworks database

The “10,000 foot view” of Django

What is Django?

What is Django?

- Bridges the gap between what the user sees and the database
- Databases are hard, but Django makes it easy to work with them
- Popular databases include:

Why Django?

Why Django?

- Python, easy to read and understand
- Don’t Repeat Yourself (DRY) Philosophy!
- The community has done a lot of the thinking for you, including security

How Does Django Work?

Model-Template-View (MTV)

- Model
  - Anything dealing with data and its representation (i.e. a user or even data validation)
- Template
  - How data is displayed, what it looks like, this is the presentation layer
- View
  - What data is presented to the template, the control logic, bridging models and template

Django gives you:

- Object-relational mapper
  - Define Python classes
  - Rich way of interacting with database
- Automatic admin interface
  - Don't waste your time creating an admin page, Django does this for you
- Elegant URL design
  - Regex matching
- Templating system
  - Fill in web pages on the fly!

Running a Django Server Locally

In your terminal, from the website directory, type:

```
<your terminal>$ python manage.py runserver

Validating models...

0 errors found
Django version 1.4, using settings 'rsvp.settings'
Development server is running at http://
127.0.0.1:8000/
Quit the server with CONTROL-C
```

Open your browser and visit:

```
localhost:8000
```

Starting a Project and Adding an App

```
<your terminal>$ django-admin.py startproject <name>
```

This creates the following project structure:

```
<name>/
    manage.py
    <name>/
        __init__.py
        settings.py
        urls.py
        wsgi.pyls
```

Our Django project:

```
event/ <- developer added app
    __init__.py
    admin.py <- developer added
    models.py
    test.py
    views.py
manage.py
rsvp/
    __init__.py
    settings.py
    static/ <- developer added
    templates/ <- developer added
    urls.py
    wsgi.pyls
```

Add an app:

```
<your terminal>$ python manage.py startapp <name>
```

Register it in ‘rsvp/settings.py’:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    # Uncomment the next line to enable the admin:
    'django.contrib.admin',
    # Uncomment the next line to enable admin 
documentation:
    # 'django.contrib.admindocs',
    '<name>'
)
```

From URL to Page Render

What happens when a user enters this in the browser?

```
www.your-site-domain.com
```

What happens when a user enters this in the browser?

```
www.your-site-domain.com
```

```
event/ <- developer added app
    __init__.py
    admin.py <- developer added
    models.py
    test.py
    views.py
manage.py
rsvp/
    __init__.py
    settings.py
    static/ <- developer added
    templates/ <- developer added
    urls.py
    wsgi.pyls
```

‘rsvp/settings.py’ stores location of our URL config file

```
ROOT_URLCONF = 'rsvp.urls'
```

We then reference ‘rsvp/urls.py’

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

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

We then reference ‘events/views.py’

```
from django.shortcuts import render

def home(request):
    return render(request, 'index.html')
```

Django knows where to find template directory because of ‘rsvp/settings.py’

```
TEMPLATE_DIRS = (
    os.path.join(PROJECT_ROOT, 'templates/'),
)
```

Work with Data

Define a model for events:

```
class Event(models.Model):
    title = models.CharField(max_length=128)
    description = models.TextField()
    date = models.DateField()
    location = models.CharField(max_length=128)

    def __unicode__(self):
        return self.title
```

Manually manipulate data using Django shell:

```
<your terminal>$ python manage.py shell
    >>> from event.models import Event
    >>> Event.objects.all()
    []
    >>> e = Event(title="Birthday", date="2013-01-13
    07:00”, location=“4-231”, description=“Awesome”)
    >>> e.save()
```

Render a template with data:

```
# View to return all events
def all_events(request):
    events = Event.objects.all()
    return render(request, 'events.html',
                  {'events': events})
```

Return JSON:

```
import json
from django.http import HttpResponse

def get_events(request):
    events = [event.title for event in Event.objects.all()]
    return HttpResponse(json.dumps({'titles':events}),
                        content_type="application/json")
```

Templating

Create a template to display all events:

```
{% extends "base.html" %}

{% block content %}
...

  {% for event in events %}
      <tr>
       <td>{{ event.title }}</td>
       <td>{{ event.date }}</td>
        <td>{{ event.description }}</td>
       <td>{{ event.location }}</td>
      </tr>
  {% endfor %}


...
{% endblock%}
```

‘base.html’ is extended by ‘events.html’:

```
<div class="container">
     {% block content%}
     {% endblock %}
</div>
```

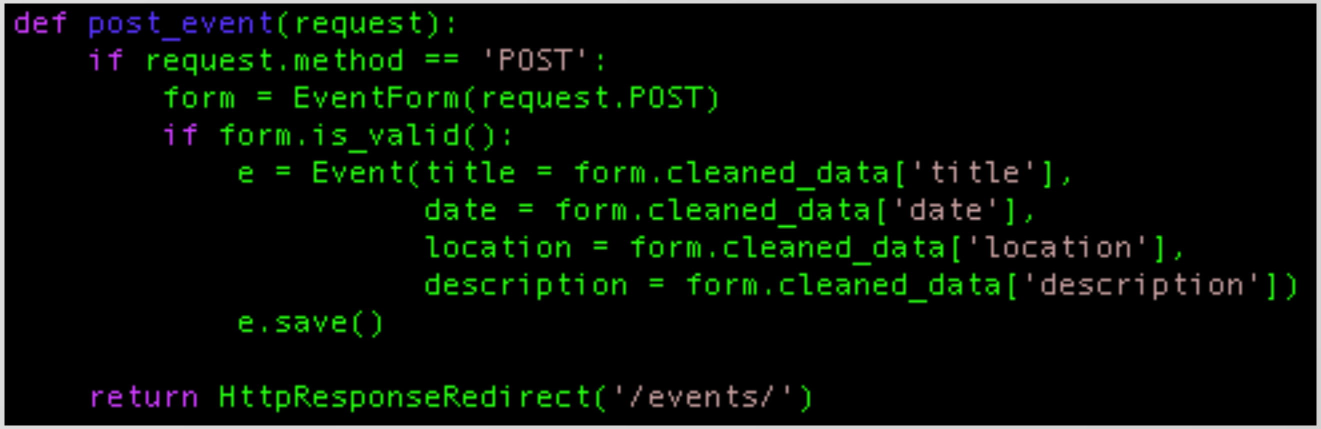
Form Generation

Create a form from a model:

```
from django.forms import ModelForm

class EventForm(ModelForm):
    class Meta:
    model = Event
```

Create a form from a model:



```
def post_event(request):
    if request.method == 'POST':
        form = EventForm(request.POST)
        if form.is_valid():
            e = Event(title = form.cleaned_data['title'],
                      date = form.cleaned_data['date'],
                      location = form.cleaned_data['location'],
                      description = form.cleaned_data['description'])
            e.save()

    return HttpResponseRedirect('/events/')
```

Adding Admin Capabilities

https://docs.djangoproject.com/en/dev/intro/tutorial02/

Web Programming in Python with Django

IAP 2013

Student Information Processing Board
Luke O'Malley '14