Django - Web Framework

Sopan Shewale

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

Agenda - Day 1

- Introduction to Django
- Starting Project and Starting Application
- Templates
- Developing views
- URLConf
- Heroku Deployment
- Models
- Admin Interface

Agenda - Day 2

- Web Application - use Matplotlib, Numpy or Pandas to deal with data (simple one)
- Rest API framework
- git - Managing your code versions
- Heroku - Deploy your Application

Django Framework

Web Development Framework - based on Python Programming

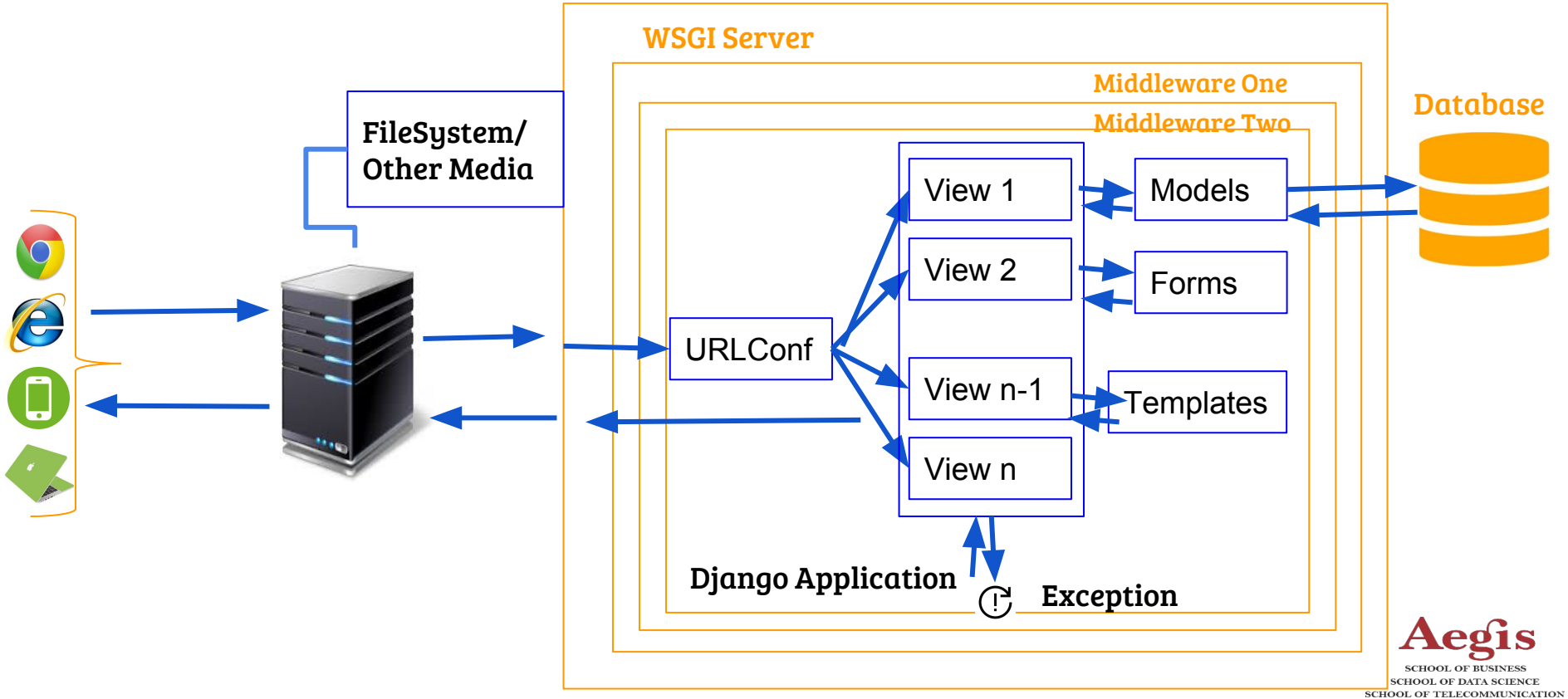
Why Django?

With Django, you can take Web applications from concept to launch in a matter of hours. Django takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

- Fast
- Fully Loaded
- Secure
- Scalable
- Versatile - used for multiple purposes

Easy - But not so easy, But Large Community
Online Documentation - Most updated

Request Processing in Django Application



Django - Project Details

-Project Name: “Lychee”

-Application Name: “temperature”

(Possible Future Apps - whether)

Setting up Project Infrastructure

```
$ pip install virtualenv
Collecting virtualenv
  Downloading https://files.pythonhosted.org/packages/ed/ea/e20b5cbebf45d3096e8138ab74eda139595d827677f
38e9dd543e6015bdf/virtualenv-15.2.0-py2.py3-none-any.whl (2.6MB)
    100% |████████████████████████████████| 2.6MB 4.2MB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-15.2.0
```

```
$ virtualenv lychee_env
Using base prefix '/anaconda'
New python executable in /Users/sopanshewale/lychee/lychee_env/bin/python
copying /anaconda/bin/python => /Users/sopanshewale/lychee/lychee_env/bin/python
copying /anaconda/bin/../lib/libpython3.6m.dylib => /Users/sopanshewale/lychee/lychee_env/lib/libpython
3.6m.dylib
Installing setuptools, pip, wheel...done.
```

```
$ source lychee_env/bin/activate
(lychee_env) $
(lychee_env) $
(lychee_env) $ python --version
Python 3.6.1 :: Continuum Analytics, Inc.
(lychee_env) $
```

Setting up Project Infrastructure (Cont …)

```
(lychee_env) $ pip install django
Collecting django
  Downloading https://files.pythonhosted.org/packages/23/91/2245462e57798e9251de87c88b2b8f996d10ddcb682
06a8a020561ef7bd3/Django-2.0.5-py3-none-any.whl (7.1MB)
    100% |████████████████████████████████| 7.1MB 686kB/s
Collecting pytz (from django)
  Downloading https://files.pythonhosted.org/packages/dc/83/15f7833b70d3e067ca91467ca245bae0f6fe56ddc74
51aa0dc5606b120f2/pytz-2018.4-py2.py3-none-any.whl (510kB)
    100% |████████████████████████████████| 512kB 4.3MB/s
Installing collected packages: pytz, django
Successfully installed django-2.0.5 pytz-2018.4
(lychee_env) $
(lychee_env) $ python
Python 3.6.1 |Continuum Analytics, Inc.| (default, May 11 2017, 13:04:09)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> print (django.VERSION)
(2, 0, 5, 'final', 0)
>>>
```

Django - Project Architecture

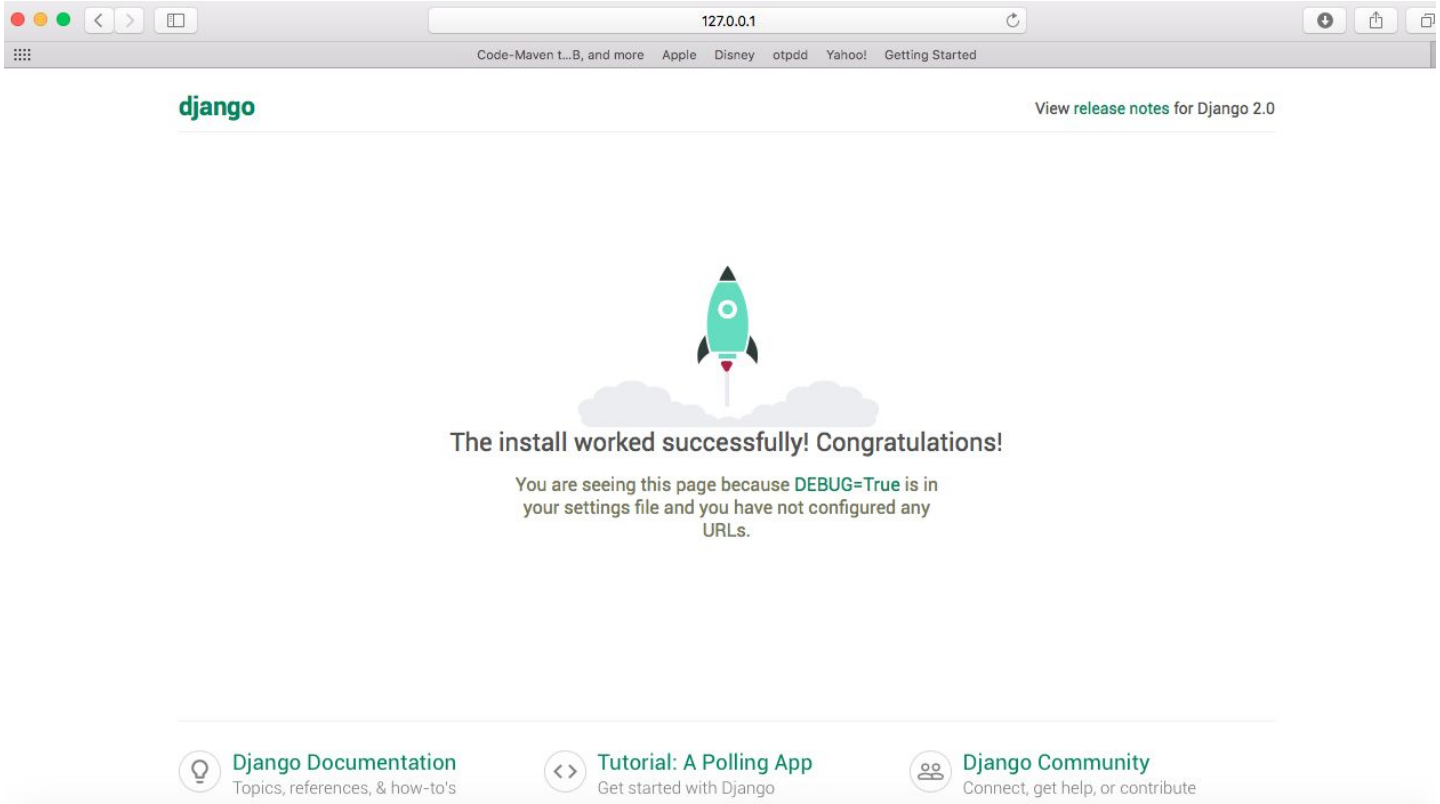
```
(lychee_env) $ django-admin startproject lychee
(lychee_env) $ tree lychee
lychee
├── lychee
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py

1 directory, 5 files
(lychee_env) $
```

- **__init__.py:** It says the “lychee” directory is Python package
- **settings.py :** Django Project settings (e.g. database configuration details)
- **urls.py:** Pattern details to URL’s
- **wsgi.py:** Important file for deployment (Web Server utilize this file)

Starting Project

```
(lychee_env) $ python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 14 unapplied migration(s). Your project may not work properly until you apply the migrations f
or app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

May 10, 2018 - 14:48:22
Django version 2.0.5, using settings 'lychee.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Starting project (Cont …)



Starting project (Cont ...)

```
(lychee_env) $ tree
.
├── db.sqlite3
├── lychee
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-36.pyc
│   │   ├── settings.cpython-36.pyc
│   │   ├── urls.cpython-36.pyc
│   │   └── wsgi.cpython-36.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py

2 directories, 10 files
(lychee_env) $
```

Starting project (Cont ...)

```
System check identified no issues (0 silenced).

You have 14 unapplied migration(s). Your project may not work properly until you apply the migrations f
or app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
```

```
(lychee_env) $ python manage.py makemigrations
No changes detected
(lychee_env) $ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
(lychee_env) $
```

Starting project (Cont ...)

```
sqlite> .tables
auth_group                  auth_user_user_permissions
auth_group_permissions      django_admin_log
auth_permission             django_content_type
auth_user                   django_migrations
auth_user_groups            django_session
sqlite>
```

manage.py makemigrations
Created many Databases

Starting project (Cont …)

You can also change the "hostname/IP" and "Port" Number to run the application

```
(lychee_env) $ python manage.py runserver 0:8181
Performing system checks...

System check identified no issues (0 silenced).
May 10, 2018 - 15:09:00
Django version 2.0.5, using settings 'lychee.settings'
Starting development server at http://0:8181/
Quit the server with CONTROL-C.
```

You are on 0 (i.e - any IP/name of your machine) and Port - 8181

Starting project (Cont …)

A Django project is collection of

- *Configurations*
- *applications*

that together make up a given web application or website

Each Application - typically handles one task/feature of the Web Application. The applications can be used (or integrated) in other Django Projects with minimum efforts.

Let us get started with Application

```
(lychee_env) $ python manage.py startapp temperature
```

```
(lychee_env) $ tree
.
├── db.sqlite3
├── lychee
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-36.pyc
│   │   ├── settings.cpython-36.pyc
│   │   ├── urls.cpython-36.pyc
│   │   └── wsgi.cpython-36.pyc
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── manage.py
└── temperature
    ├── __init__.py
    ├── admin.py
    ├── apps.py
    ├── migrations
    │   └── __init__.py
    ├── models.py
    ├── tests.py
    └── views.py

4 directories, 17 files
(lychee_env) $
```

Django Application

- **__init__.py:** Indicates Python to treat directory as package
- **models.py:** A place to store applications mode (defines entities relations, types etc)
- **tests.py:** Application test cases
- **Admin.py:**

“views.py” and “models.py” - most important and frequently used files to develop application

Django Application (Cont …)

lychee/settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'temperature',
]
```

Django Application (cont …)

temperature/views.py

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse

def index(request):
    return HttpResponse("Hello, world. You are developing Temerature Application")
```

Django Application (Cont …)

temperature/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

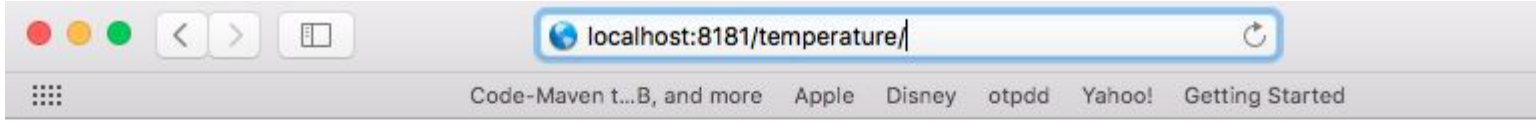
Django Application (Cont …)

lychee/urls.py

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('temperature/', include('temperature.urls')),
]
```

Django Application (Cont …)

```
(lychee_env) $ python manage.py runserver 0:8181
Performing system checks...

System check identified no issues (0 silenced).
May 10, 2018 - 18:48:15
Django version 2.0.5, using settings 'lychee.settings'
Starting development server at http://0:8181/
Quit the server with CONTROL-C.
```



Hello, world. You are developing Temerature Application

```
[10/May/2018 18:48:31] "GET /temperature HTTP/1.1" 301 0
[10/May/2018 18:48:31] "GET /temperature/ HTTP/1.1" 200 55
```

Starting Application - Summary of Actions

1. $python manage.py startapp <appname> - Helps create new application
2. Tell your Django project about the new application - Edit INSTALLED_APPS tuple in your project's settings.py file.
3. Add urls.py file in your application directory
4. Modify your project "urls.py" - add mapping to your new application (via include)
5. In your application's view.py, create the required views ensuring that they return a HttpResponse object

Task - 1

Create “http://localhost:8181/temperature/about” view

-Add **“about”** method in temperature/views.py module

-Update “temperature/urls.py” to support “temperature/about” mapping

-Do you need that restart in your development server?

settings

<project>/settings.py has all the configuration details about your project.

-Database configuration

-Static File Location Details

-Template Details

-May be, you would like to configure a few Constants/Variables used everywhere in your project (e.g. DEPLOYMENT_TYPE=development)

settings (cont...)

```
root@ba896a66dd36:/app# more restbook/settings/development.py
from .base import *  # noqa: F403
DEBUG = True
SECRET_KEY = '_1)+c%uy*ham17*vjcf0ubl2h0!bwbed((83&q0#f9i-po!!@8'
ALLOWED_HOSTS = ['*',]
APPLICATION_ENV='dev'

DATABASES = {
    'default': {
        'ENGINE': 'django.
        'NAME':'postgres',
        'USER': 'postgres'
        'PASSWORD': 'postg
        'HOST': 'postgres'
        'PORT': '',
    }
}

SMTP = {
        'smtp_server': 'smtp.gmail.com',
        'smtp_port' : 587,
        'mail_user' :'sopan.shewale@fyra.biz',
        'mail_password': 'xxxxx',
        'from_mail' : 'sopan.shewale@fyra.biz',
}
```

```
# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

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

Static Files

Let us configure static file storage directory

lychee/settings.py

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]
```

Added lychee_logo.png in “static/img/” directory

```
(lychee_env) $ tree static/
static/
└── img
    └── lychee_logo.png

1 directory, 1 file
```

static/img/lychee_logo.png
“mapped to
http://localhost/static/img/lychee_logo.png

static files (cont ...)

Static files (Cont …)

- Store images
- CSS files and Libraries, Frameworks (e.g. Bootstrap)
- Javascript Files and Libraries (e.g. jQuery)
- Any static content required for your Web Application

Templates

- It's programming language itself
- Django supports powerful templating Engine
  - Helps Rendering/presenting content on browser/any client

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ["templates",],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Templates (Cont …)

BACKEND - template engine class implementing Django's template backend API.

Possible default Options:

- **django.template.backends.django.DjangoTemplates**
- **django.template.backends.jinja2.Jinja2**

Most engines load templates from files

- **DIRS** defines a list of directories where the engine should look for template files
- **APP_DIRS** tells whether the engine should look for templates inside installed applications

Template (Cont …)

Let us add template to display our first page

template/temperature/index.html

```
<!DOCTYPE html>
<html>
<head>
   <title>Temperature Application </title>
</head>
<body>

<h1>"Hello, world. You are developing Temerature Application"</h1>

</body>
</html>
```

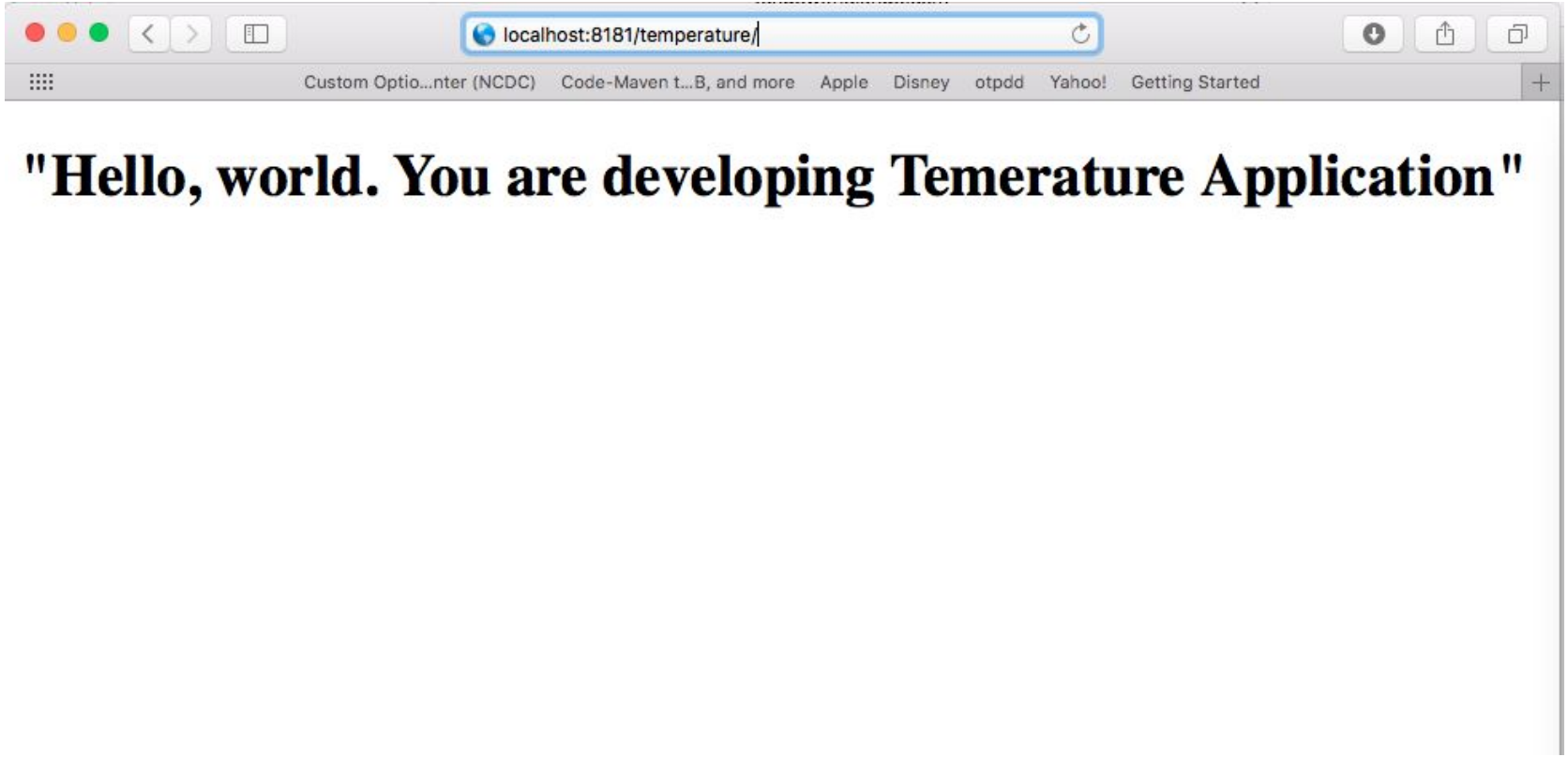
Template (Cont …)

temperature/views.py

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse

def index(request):
    #return HttpResponse("Hello, world. You are developing Temerature Application")
    return render(request, 'index.html')

def about(request):
    return HttpResponse("The Web Application is owned by Data Science Folks at Aegis")
```

"Index.html"
Mapped to
template/temperature/index.html

Template (Cont ...)



Template (Cont …)

template/temperature/sample_temp.html

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

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Template (Cont ...)

```
<ul>
{% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
{% endfor %}
</ul>


{% if athlete_list %}
    Number of athletes: {{ athlete_list|length }}
{% elif athlete_in_locker_room_list %}
    Athletes should be out of the locker room soon!
{% else %}
    No athletes.
{% endif %}
```

Template (Cont …)

template/temperature/index.html

```
<!DOCTYPE html>

{% load staticfiles %}

<html>
<head>
   <title>Temperature Application </title>
</head>
<body>


<img src="{% static "img/lychee_logo.png" %}" alt="Lychee Logo" height="60" width="60" />

<hr>
<h1>"Hello, world. You are developing Temerature Application"</h1>

</body>
</html>
```

Template (Cont …)

Task - 2

“About Page” - Add template for “/temperature/about” URL. Also add logo to the page.

one_lychee.tar.gz

two_lychee.tar.gz

Views - do more!

Let us display records via “views” method

temperature/Temperature.py

```
from datetime import datetime
from django.utils import timezone

class Temperature(object):
     def __init__(self, temperature =0, max_temp = 0, min_temp = 0, record_day = None):
         self.temperature = temperature
         self.max_temp = max_temp
         self.min_temp = min_temp
         if record_day is None:
            self.record_day = timezone.now
         else:
            self.record_day = record_day
```

Views (cont …)

Look at “records” method from temperature.views

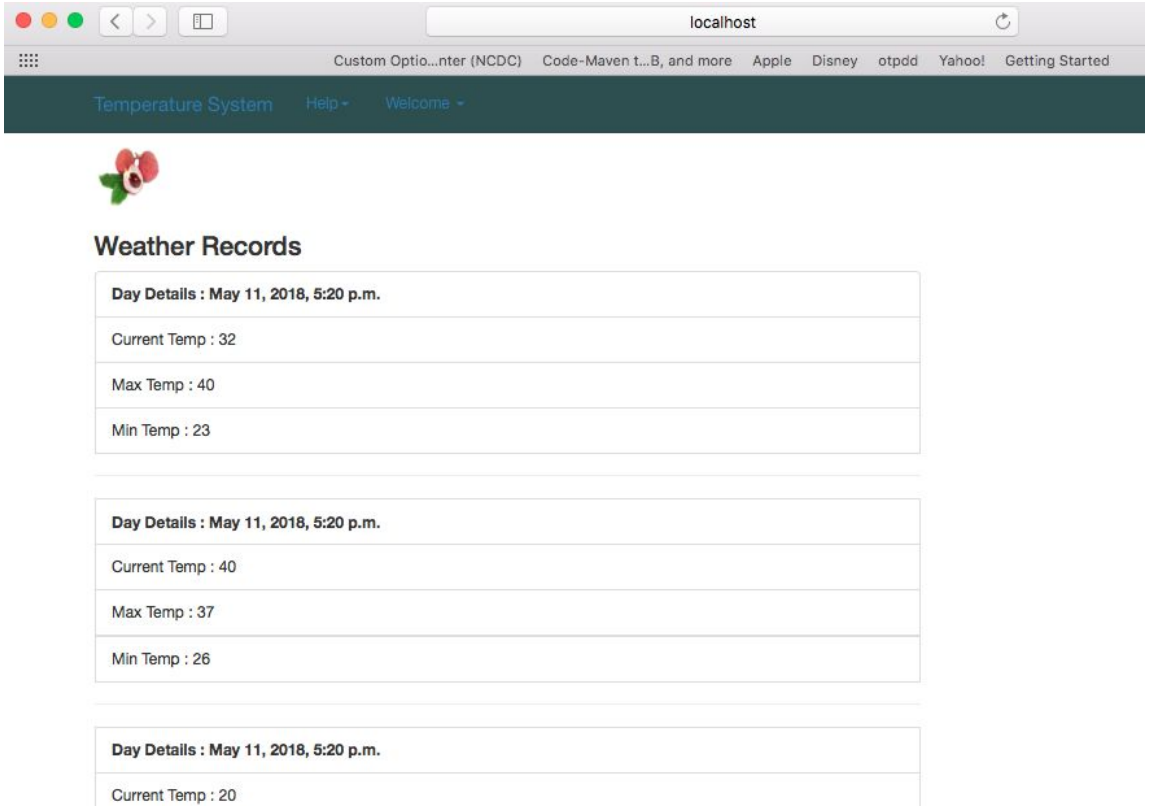
```
def records(request):
    weather_1 = Temperature(
                            temperature=random.randint(20,40),
                            max_temp=random.randint(30,40),
                            min_temp=random.randint(20,30)
                           )
    weather_2 = Temperature(
                            temperature=random.randint(20,40),
                            max_temp=random.randint(30,40),
                            min_temp=random.randint(20,30)
                           )
    weather_3 = Temperature(
                            temperature=random.randint(20,40),
                            max_temp=random.randint(30,40),
                            min_temp=random.randint(20,30)
                           )
    weather_4 = Temperature(
                            temperature=random.randint(20,40),
                            max_temp=random.randint(30,40),
                            min_temp=random.randint(20,30)
                           )
    weather_list = [weather_1, weather_2, weather_3, weather_4]
    context = {
        'temp':weather_list
    }

    return render(request, 'weather.html', context)
```

Views (cont …)

temperature/templates/weather.html

```
{% extends 'base.html' %}
<!DOCTYPE html>

{% load staticfiles %}

{% block content %}

<img src="{% static "img/lychee_logo.png" %}" alt="Lychee Logo" height="60" width="60" />

<h3>Weather Records</h3>

<ul class="list-group">
{% for record in temp %}
   <li class="list-group-item"><b>Day Details :  {{ record.record_day }} </b></li>
   <li class="list-group-item">Current Temp :  {{ record.temperature }} </li>
   <li class="list-group-item">Max Temp :  {{ record.max_temp }} </li>
   <li class="list-group-item">Min Temp :  {{ record.min_temp }} </li>
   <hr>
{% endfor %}
</ul>

{% endblock %}
```

Views (Cont …)


localhost
Custom Optio...nter (NCDC)
Code-Maven t...B, and more
Apple
Disney
otpdd
Yahoo!
Getting Started
Temperature System
Help
Welcome
Weather Records
Day Details : May 11, 2018, 5:20 p.m.
Current Temp : 32
Max Temp : 40
Min Temp : 23
Day Details : May 11, 2018, 5:20 p.m.
Current Temp : 40
Max Temp : 37
Min Temp : 26
Day Details : May 11, 2018, 5:20 p.m.
Current Temp : 20

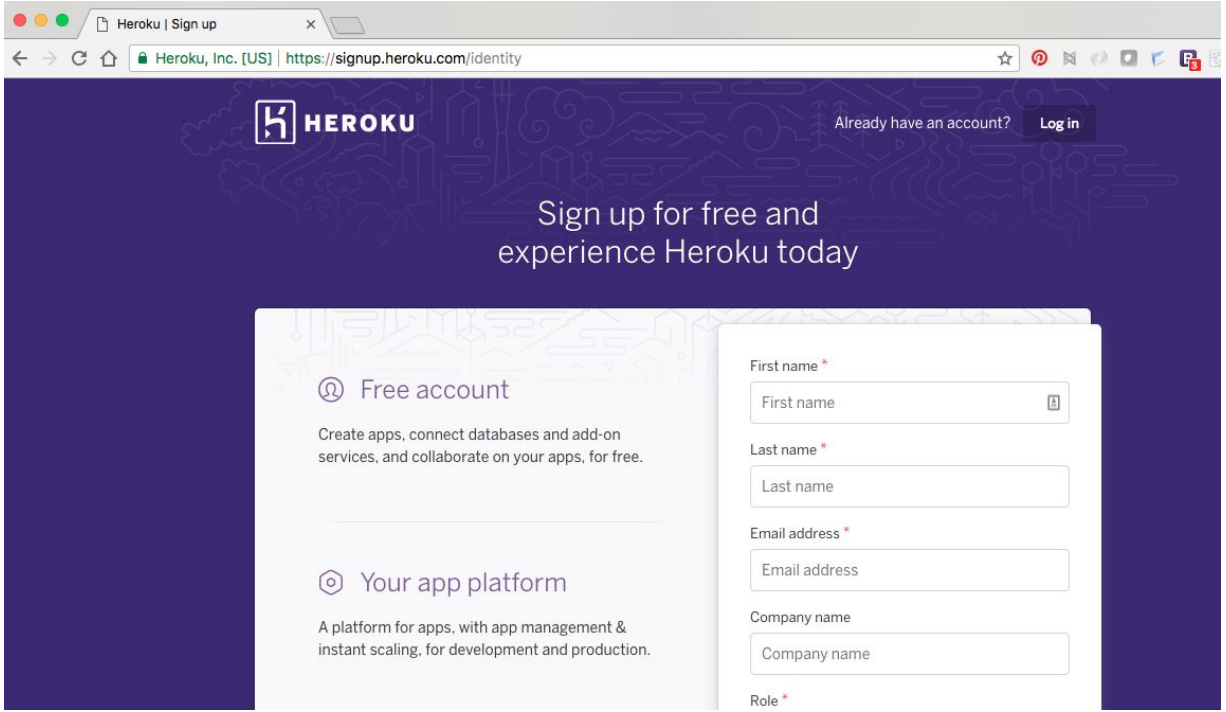
Task-3

Using “Temperature Class” - print weeks temperature records.

You need to create week days list using today’s date.

Heroku Deployment

https://signup.heroku.com/identity



Heroku Deployment

Install Command Line utility for Heroku

```
(lychee_env) hari-sadu:lychee sopanshewale$ heroku login
heroku: Enter your login credentials
Email [sopan.shewale@gmail.com]:
Password: **********
Logged in as sopan.shewale@gmail.com
```

```
(lychee_env) hari-sadu:lychee sopanshewale$ heroku create lychee-temp
Creating ⬢ lychee-temp... done
https://lychee-temp.herokuapp.com/ | https://git.heroku.com/lychee-temp.git
(lychee_env) hari-sadu:lychee sopanshewale$
```

-create requirement.txt

-create Procfile

-Create Pipfile

- git init
- git add .
- git commit -am "my lychee app"
- git remote add heroku https://git.heroku.com/lychee-temp.git
- git push heroku master

Assignment

Create Account at Heroku

Deploy application at Heroku

Enable Static Files Delivery

Models and Databases

- Almost all hassle is taken care by Django's *object relational mapping (ORM)*
- Django encapsulates databases tables through models
- model is a Python object that describes your data model/table
- That helps you manipulate the Python object than playing with SQL Queries

lychee/settings.py

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

Models (Cont ...)

temperature/models.py

```
from django.db import models
from django.utils.translation import ugettext_lazy as _
from django.utils import timezone
from datetime import datetime

class Temperature(models.Model):
       temperature = models.PositiveIntegerField(
                     verbose_name=_('Current Temperatuer in Cel'),
                     default = 0,
       )
       min_temp = models.PositiveIntegerField(
                     verbose_name=_('Current Temperatuer in Cel'),
                     default = 0,
       )
       max_temp = models.PositiveIntegerField(
                     verbose_name=_('Current Temperatuer in Cel'),
                     default = 0,
       )
       entry_time = models.DateTimeField(
                    default= timezone.now,
                    verbose_name=_('Entry Time'),
                    blank=True, null=True,
      )
```

```
(lychee_env) hari-sadu:lychee sopanshewale$ python manage.py makemigrations temperature
Migrations for 'temperature':
  temperature/migrations/0001_initial.py
    - Create model Temperature
```

```
(lychee_env) hari-sadu:lychee sopanshewale$ more temperature/migrations/0001_initial.py
# Generated by Django 2.0.5 on 2018-05-12 07:14

from django.db import migrations, models
import django.utils.timezone


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Temperature',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('temperature', models.PositiveIntegerField(default=0, verbose_name='Current Temperatuer in Cel')),
                ('min_temp', models.PositiveIntegerField(default=0, verbose_name='Current Temperatuer in Cel')),
                ('max_temp', models.PositiveIntegerField(default=0, verbose_name='Current Temperatuer in Cel')),
                ('entry_time', models.DateTimeField(blank=True, default=django.utils.timezone.now, null=True, verbose_name='Entry Time')),
            ],
        ),
    ]
```

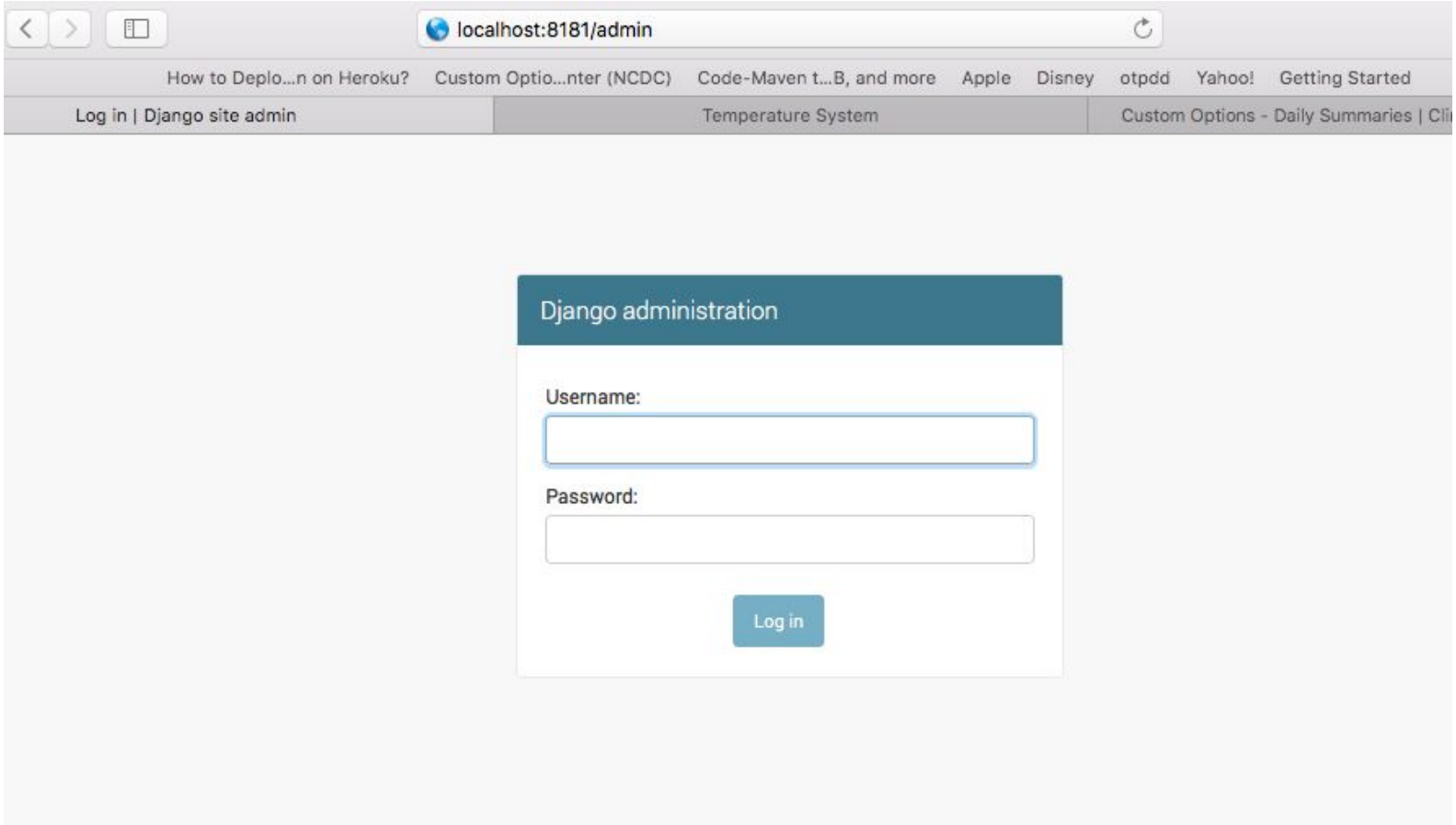
```
(lychee_env) hari-sadu:lychee sopanshewale$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, temperature
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
  Applying temperature.0001_initial... OK
```

```
sqlite> .tables
auth_group                  django_admin_log
auth_group_permissions      django_content_type
auth_permission             django_migrations
auth_user                   django_session
auth_user_groups            temperature_temperature
auth_user_user_permissions
sqlite>
```

Admin Interface

Create Admin User

```
(lychee_env) $ python manage.py createsuperuser
Username (leave blank to use 'sopanshewale'): info@aegis.com
Email address: info@aegis.com
Password:
Password (again):
Superuser created successfully.
(lychee_env) $
```


localhost:8181/admin
How to Deplo...n on Heroku?
Custom Optio...nter (NCDC)
Code-Maven t...B, and more
Apple
Disney
otpdd
Yahoo!
Getting Started
Log in | Django site admin
Temperature System
Custom Options - Daily Summaries | Cli
Django administration
Username:
Password:
Log in

localhost

How to Deplo...n on Heroku? Custom Optio...nter (NCDC) Code-Maven t...B, and more Apple Disney otpdd Yahoo! Getting Started

Authentication and Authorization administration | Django site a... | Temperature System | Custom Options - Daily Summaries | Climate Data Online (...

Django administration

WELCOME, **INFO@AEGIS.COM**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization

Authentication and Authorization administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

Aegis
SCHOOL OF BUSINESS
SCHOOL OF DATA SCIENCE
SCHOOL OF TELECOMMUNICATION

```
from django.contrib import admin

from temperature.models import Temperature
admin.site.register(Temperature)
```

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

TEMPERATURE		
Temperatures	+ Add	Change

Recent actions

My actions

None available

thx

Agenda - Day 2

- URL parameters
- Models
- Forms
- Executing commands - from OS
- Web Application - use Matplotlib, Numpy or Pandas to deal with data (simple one)
- Rest API framework
- Heroku - Deploy your Application

URL - More Information

Capturing parameters

temperature/urls.py

```
path('trecords2003/', views.special_case_2003),
re_path(r'^trecords/(?P<year>[0-9]{4})/$', views.year_archive),
re_path(r'^trecords/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/$', views.month_archive),
re_path(r'^trecords/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/(?P<slug>[\w-]+)/$', views
```

temperature/views.py

```
def year_archive(request, year=None):
    print (year)
    print ("I can use this variable anywhere")
    return render(request, 'index.html')
```

Task - 4

Develop methods from **temperature/views.py** - demonstrate skill to capture URL parts.

Models

temperature/models.py

```
from django.db import models
from django.utils.translation import ugettext_lazy as _
from django.utils import timezone
from datetime import datetime

class Temperature(models.Model):
        temperature = models.PositiveIntegerField(
                      verbose_name=_('Current Temperatuer in Cel'),
                      default = 0,
        )
        min_temp = models.PositiveIntegerField(
                      verbose_name=_('Current Temperatuer in Cel'),
                      default = 0,
        )
        max_temp = models.PositiveIntegerField(
                      verbose_name=_('Current Temperatuer in Cel'),
                      default = 0,
        )
        entry_time = models.DateTimeField(
                     default= timezone.now,
                     verbose_name=_('Entry Time'),
                     blank=True, null=True,
       )
```

models

```
(lychee_env) $ python manage.py makemigrations
Migrations for 'temperature':
  temperature/migrations/0001_initial.py
    - Create model Temperature
```

```
(lychee_env) $ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, temperature
Running migrations:
  Applying temperature.0001_initial... OK
```

```
(lychee_env) $ python manage.py dbshell
SQLite version 3.13.0 2016-05-18 10:57:30
Enter ".help" for usage hints.
sqlite> .schema
```

models

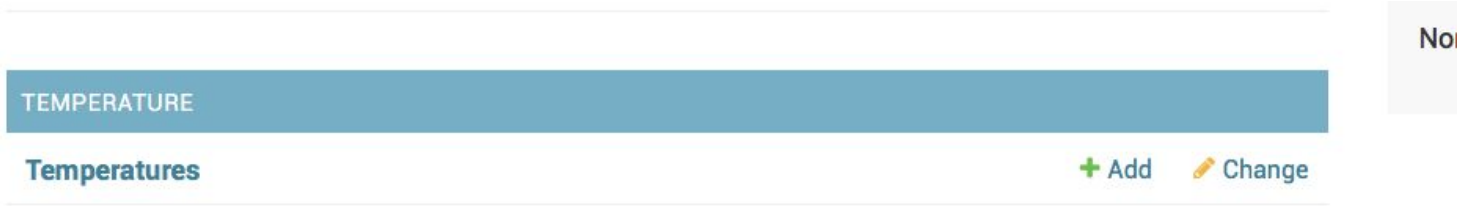
```
sqlite> .tables
auth_group                  django_admin_log
auth_group_permissions      django_content_type
auth_permission             django_migrations
auth_user                   django_session
auth_user_groups            temperature_temperature
auth_user_user_permissions
sqlite>
```

```
sqlite> select * from auth_user;
1|pbkdf2_sha256$100000$7dnVbg9FY1rN$UquM26nwGQAvBq9Fqurh6MF7GiCb19NzS/H4JJu9Ld8=|2018-05-12 13:54:44
.374643|1|info@aegis.com||info@aegis.com|1|1|2018-05-12 13:51:42.466946|
sqlite>
```

Admin - Accessing Model

temperature/admin.py

```
from django.contrib import admin

from temperature.models import Temperature
admin.site.register(Temperature)
```



Admin - Accessing Model (Cont ...)

Home › Temperature › Temperatures › Add temperature

Add temperature

Current Temperatuer in Cel: 0

Min Temperatuer in Cel: 0

Max Temperatuer in Cel: 0

Entry Time:

Date: 2018-05-13 Today |

Time: 02:20:49 Now |

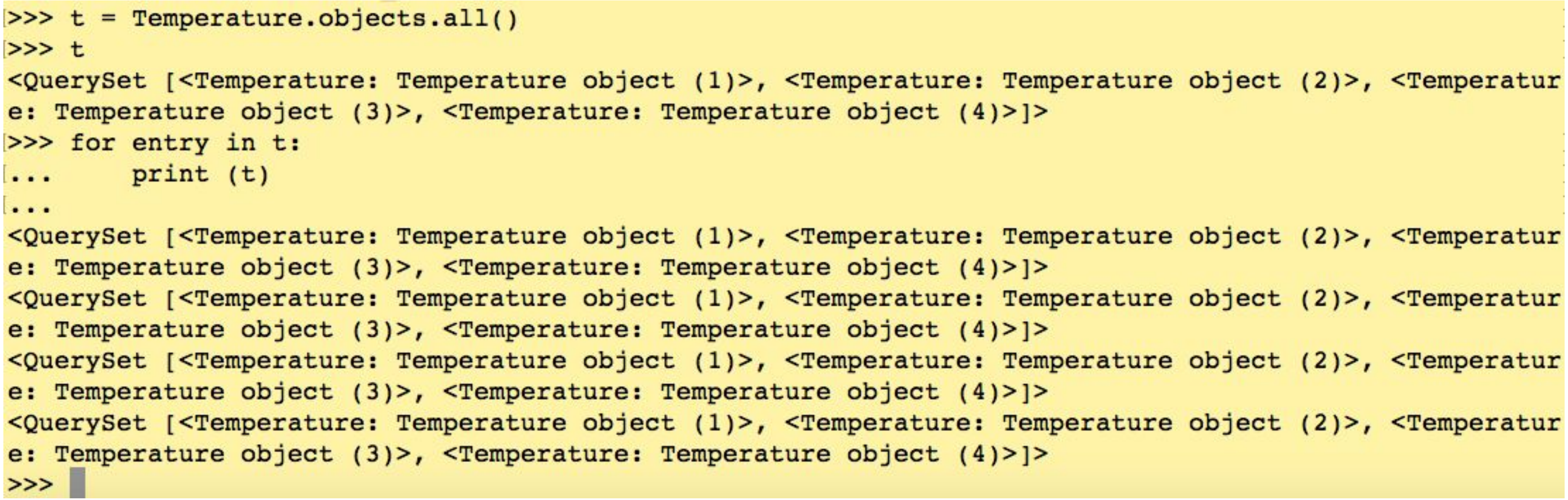
Note: You are 5.5 hours ahead of server time.

Django Shell

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

```
>>> from temperature.models import Temperature
>>> t = Temperature.
Temperature.DoesNotExist(                Temperature.max_temp
Temperature.MultipleObjectsReturned(     Temperature.min_temp
```

Django Shell (Cont ...)



```
>>> t = Temperature.objects.all()
>>> t
<QuerySet [<Temperature: Temperature object (1)>, <Temperature: Temperature object (2)>, <Temperatur
e: Temperature object (3)>, <Temperature: Temperature object (4)>]>
>>> for entry in t:
...     print (t)
...
<QuerySet [<Temperature: Temperature object (1)>, <Temperature: Temperature object (2)>, <Temperatur
e: Temperature object (3)>, <Temperature: Temperature object (4)>]>
<QuerySet [<Temperature: Temperature object (1)>, <Temperature: Temperature object (2)>, <Temperatur
e: Temperature object (3)>, <Temperature: Temperature object (4)>]>
<QuerySet [<Temperature: Temperature object (1)>, <Temperature: Temperature object (2)>, <Temperatur
e: Temperature object (3)>, <Temperature: Temperature object (4)>]>
<QuerySet [<Temperature: Temperature object (1)>, <Temperature: Temperature object (2)>, <Temperatur
e: Temperature object (3)>, <Temperature: Temperature object (4)>]>
>>> 
```

Django Shell (Cont ..)

temperature/models.py

```
    def __str__(self):
        return 'Today Temperature: ' + str(self.temperature)
```

```
>>> from temperature.models import Temperature
>>> t = Temperature.objects.all()
>>> t
<QuerySet [<Temperature: Today Temperature: 30>, <Temperature: Today Temperature: 29>, <Temperature:
 Today Temperature: 29>, <Temperature: Today Temperature: 27>]>
>>> for entry in t:
...     print (entry)
...
Today Temperature: 30
Today Temperature: 29
Today Temperature: 29
Today Temperature: 27
>>>
```

Django Shell

```
>>> new_t = Temperature(min_temp=10, max_temp=20, temperature=10)
>>> new_t.save()
```

```
>>> t = Temperature.objects.all()
>>> for entry in t:
...     print (entry)
...
Today Temperature: 30
Today Temperature: 29
Today Temperature: 29
Today Temperature: 27
Today Temperature: 10
>>>
```

Django Forms

temperature/forms.py

```
from django import forms
from temperature.models import Temperature


class TemperatureForm (forms.ModelForm):
    class Meta:
        model = Temperature
        fields = ('temperature', 'min_temp', 'max_temp')
```

Django Forms (Cont ...)

```
{{ temp_form }}
```

Current Temperatuer in Cel: 0 Min Temperatuer in Cel:

0 Max Temperatuer in Cel: 0

Django Forms (Cont …)

temperature/templates/about.html

```
<form class="form-horizontal" method="post">{% csrf_token %}
    <fieldset>
        <legend>{{ title }}</legend>
        {% for field in temp_form %}
            {% if field.errors %}
                <div class="control-group error">
                    <label class="control-label">{{ field.label }}</label>
                    <div class="controls">{{ field }}
                        <span class="help-inline">
                            {% for error in  field.errors %}{{ error }}{% endfor %}
                        </span>
                    </div>
                </div>
            {% else %}
                <div class="control-group">
                    <label class="control-label">{{ field.label }}</label>
                    <div class="controls">{{ field }}
                        {% if field.help_text %}
                            <p class="help-inline"><small>{{ field.help_text }}</small></p
>
                        {% endif %}
                    </div>
                </div>
            {% endif %}
        {% endfor %}
    </fieldset>
    <div class="form-actions">
        <button type="submit" class="btn btn-primary" >Submit</button>
    </div>
</form>
```

Django Forms (Cont ...)

```
  {% for error in  field.errors %}{{ error }}{% endfor %}
```

```
 <p class="help-inline"><small>{{ field.help_text }}</small></p
```

```
 <label class="control-label">{{ field.label }}</label>
```

Task 5

Add “/temperature/add” - present form to capture temperature

“/temperature/add/record” - present details of particular record

(yes, that’s database primary key)

File Upload

temperature/forms.py

```
class UploadFileForm(forms.Form):
    title = forms.CharField(max_length=50)
    file = forms.FileField()
```

A view handling this form will receive the file data in **`request.FILES`**, which is a dictionary containing a key for each **`FileField`** (or **`ImageField`**, or other **`FileField`** subclass) in the form. So the data from the above form would be accessible as **`request.FILES['file']`**.

Note that **`request.FILES`** will only contain data if the request method was **`POST`** and the **`<form>`** that posted the request has the attribute **`enctype="multipart/form-data"`**. Otherwise, **`request.FILES`** will be empty.

File Upload (Cont …)

lychee/settings.py

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'static/media')
```

File Upload (Cont …)

temperature/templates/simple_upload.html

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

{% load static %}

{% block content %}
  <form method="post" enctype="multipart/form-data">
    {% csrf_token %}
    <input type="file" name="myfile">
    <button type="submit">Upload</button>
  </form>

  {% if uploaded_file_url %}
    <p>File uploaded at: <a href="{{ uploaded_file_url }}">{{ uploaded_file_url }}</a></p>
  {% endif %}

  <p><a href="{% url 'about' %}">Return to home</a></p>
{% endblock %}
```

File Upload(Cont ...)

temperature/views.py

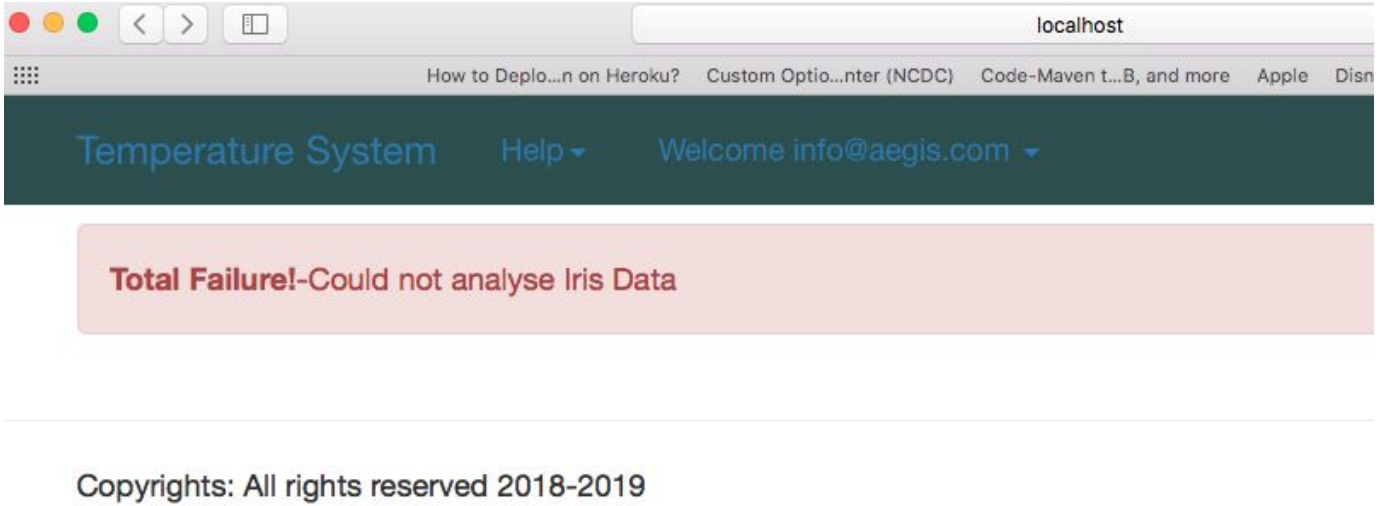
```
from temperature.forms import UploadFileForm
from django.conf import settings
```

```
def simple_upload(request):
    if request.method == 'POST' and request.FILES['myfile']:
        myfile = request.FILES['myfile']
        path= getattr(settings, 'MEDIA_ROOT')
        fs = FileSystemStorage(path)
        filename = fs.save(myfile.name, myfile)
        uploaded_file_url = fs.url(filename)
        return render(request, 'simple_upload.html', {
            'uploaded_file_url': uploaded_file_url
        })
    return render(request, 'simple_upload.html')
```

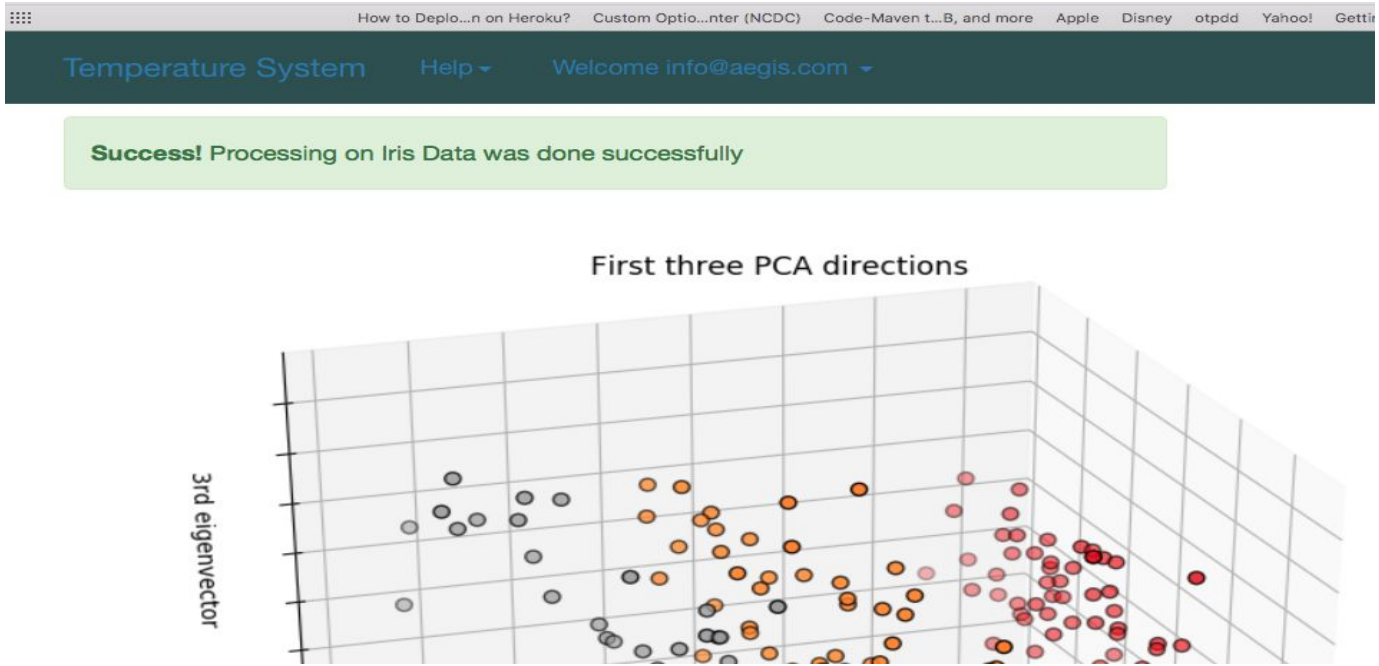
Iris Image

```
def iris(request):
   from subprocess import call
   status = call(["/anaconda/bin/python", "/Users/sopanshewale/lychee/lychee/scripts/plot_iris_datas
et.py"])
   if status == 0:
      return render(request, 'iris_success.html')
   else:
      return render(request, 'iris_fail.html')
```

Iris (Cont …)


localhost
How to Deplo...n on Heroku?
Custom Optio...nter (NCDC)
Code-Maven t...B, and more
Apple
Disn
Temperature System
Help
Welcome info@aegis.com
Total Failure!-Could not analyse Iris Data
Copyrights: All rights reserved 2018-2019

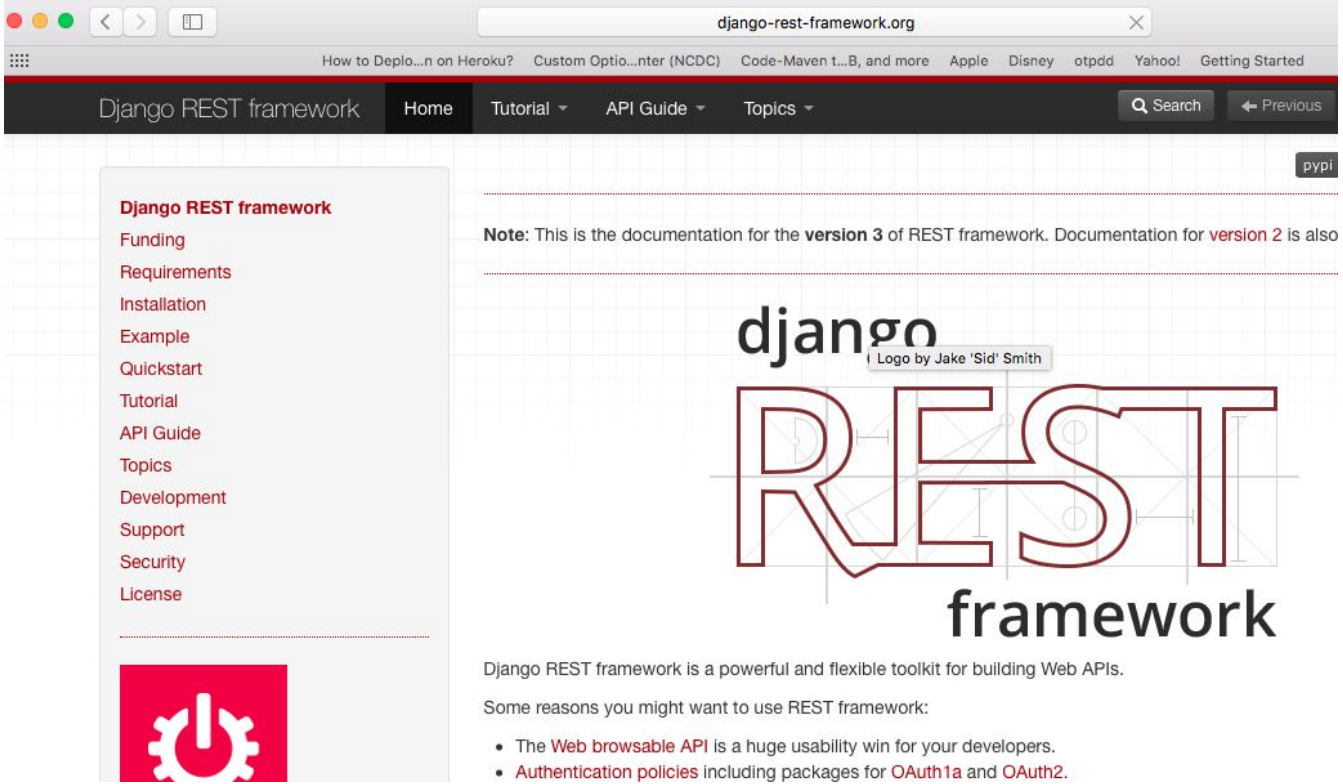
Iris (Cont…)


How to Deplo…n on Heroku?
Custom Optio…nter (NCDC)
Code-Maven t…B, and more
Apple
Disney
otpdd
Yahoo!
Temperature System
Help
Welcome info@aegis.com
Success! Processing on Iris Data was done successfully
First three PCA directions
3rd eigenvector

Task - 5
Google Chart

Use Google Stock Market data to chart image - use matplotlib

REST API


django-rest-framework.org
How to Deplo...n on Heroku?
Custom Optio...nter (NCDC)
Code-Maven t...B, and more
Apple
Disney
otpdd
Yahoo!
Getting Started
Django REST framework
Home
Tutorial
API Guide
Topics
Search
Previous
pypi
Django REST framework
Funding
Requirements
Installation
Example
Quickstart
Tutorial
API Guide
Topics
Development
Support
Security
License
Note: This is the documentation for the version 3 of REST framework. Documentation for version 2 is also
Logo by Jake 'Sid' Smith
REST
framework
Django REST framework is a powerful and flexible toolkit for building Web APIs.
Some reasons you might want to use REST framework:
The Web browsable API is a huge usability win for your developers.
Authentication policies including packages for OAuth1a and OAuth2.

Rest API(Cont ...)

api/urls.py

```
from django.conf.urls import url, include
from rest_framework import routers
from api.views import UserViewSet, TemperatureViewSet

router = routers.DefaultRouter()
router.register(r'users', UserViewSet)
router.register(r'temperatures', TemperatureViewSet)


urlpatterns = [
    url(r'^api/', include(router.urls)),
    url(r'^api-auth/', include('rest_framework.urls', namespace='rest_framework'))
]
```

REST API (Cont …)

api/views.py

```
from django.shortcuts import render
from api.serializers import UserSerializer, TemperatureSerializer
from django.contrib.auth.models import User
from rest_framework import routers, serializers, viewsets
from temperature.models import Temperature

class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer

class TemperatureViewSet(viewsets.ModelViewSet):
    queryset = Temperature.objects.all()
    serializer_class = TemperatureSerializer
```

REST API (Cont…)

api/serializers.py

```
from rest_framework import serializers
from temperature.models import Temperature
from django.contrib.auth.models import User
from django.contrib.auth.models import Group

class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ('url', 'username', 'email', 'is_staff')

class TemperatureSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Temperature
        fields = ('temperature', 'max_temp', 'min_temp',)
```

Project

For any Stock Symbol (scripe) - get data about stock and display graph on web page

The stock symbol should be taken as input from user - via browser (use forms)

If there are errors to fetch stock prices, throw an appropriate error.

thx