Object Oriented Software Engineering (SE313)
Lab Session # 09

Objective Web Development using python framework (Django)

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it 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.

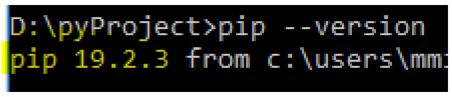
Installing an official release of Django with pip (Python Package Index)

Step 1: First, let's check whether you already have **pip** installed:

1. Type the following command into the command prompt and press Enter to see if pip is already installed:

```
pip --version
```

2. If pip is installed and working, you will see a version number like this:



```
D:\pyProject>pip --version
pip 19.2.3 from c:\users\mm
```

3. If you do, you're ready to use pip to install any Python module you like by typing the following into a command prompt

```
pip install name-of-module
```

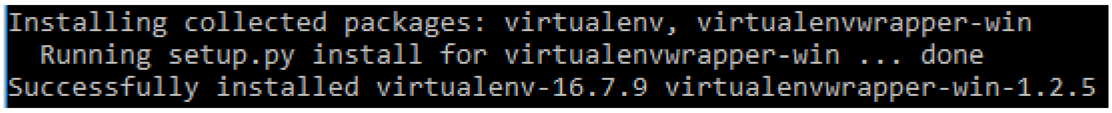
pip is the package installer for Python. Most distributions of Python come with pip preinstalled.

Step 2: Install **virtualenv** and **virtualenvwrapper**

1. While not mandatory, this is considered a best practice and will save you time in the future when you're ready to deploy your project. To do this, run

```
...\> py -m pip install virtualenvwrapper-win
```

It will start installation and later you will see the message below.



```
Installing collected packages: virtualenv, virtualenvwrapper-win
  Running setup.py install for virtualenvwrapper-win ... done
Successfully installed virtualenv-16.7.9 virtualenvwrapper-win-1.2.5
```

2. Then create a virtual environment for your project:

```
...\> mkvirtualenv myproject
```

3. The virtual environment will be activated automatically and you'll see "(myproject)" next to the command prompt to designate that. If you start a new command prompt, you'll need to activate the environment again using:

```
...\> workon myproject
```

virtualenv and **virtualenvwrapper** provide a dedicated environment for each Django project you create.

Step 3: Install Django

1. **Django** can be installed easily using pip within your virtual environment.
   In the command prompt, ensure your virtual environment is active, and
   execute the following command:
   ```
   (myproject) D:\pyProject>python -m pip install django
   ```
   This will download and install the latest Django release within your virtual environment.
   And you will see the following message (installation successfully)
   ```
   Installing collected packages: pytz, asgiref, sqlparse, django
   Successfully installed asgiref-3.2.3 django-3.0 pytz-2019.3 sqlparse-0.3.0
   ```

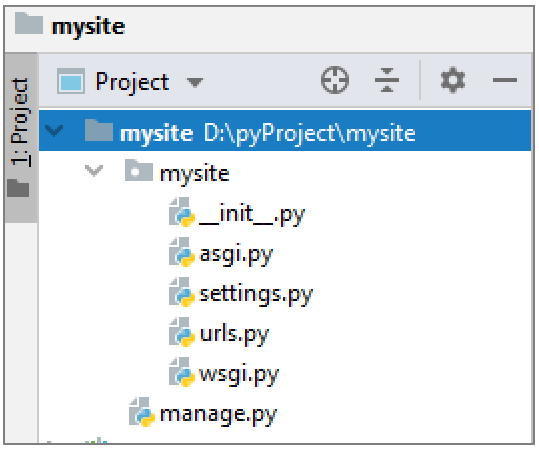
2. After the installation has completed, you can verify your Django installation by executing the following command in the command prompt.
   ```
   (myproject) D:\pyProject>django-admin --version
   3.0
   ```
   OR
   ```
   (myproject) D:\pyProject>python -m django --version
   3.0
   ```

Create your First Django Project:

Step 1 Creating Django project

1. From the command line, **cd** into a directory where you'd like to store your code, then run the following command:
   ```
   (myproject) D:\pyProject>django-admin startproject mysite
   ```
   This will create a **mysite** directory in your current directory

2. Let's look at what **startproject** created:



```
mysite
Project
mysite D:\pyProject\mysite
    mysite
        __init__.py
        asgi.py
        settings.py
        urls.py
        wsgi.py
    manage.py
```

These files are:

I. The outer **mysite/** root directory is a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.

II. **manage.py**: A command-line utility that lets you interact with this Django project in various ways. You can read all the details about **manage.py** in django-admin and manage.py.

III. The inner **mysite/** directory is the actual Python package for your project. Its name is the Python package name you'll need to use to import anything inside it (e.g. **mysite.urls**).

IV. **mysite/__init__.py**: An empty file that tells Python that this directory should be considered a Python package. If you're a Python beginner, read more about packages in the official Python docs.

V. **mysite/settings.py**: Settings/configuration for this Django project. Django settings will tell you all about how settings work.

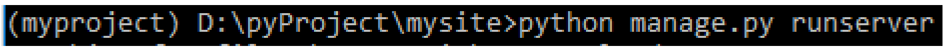
VI. **mysite/urls.py**: The URL declarations for this Django project; a “table of contents” of your Django-powered site. You can read more about URLs in URL dispatcher.

VII. **mysite/asgi.py**: An entry-point for ASGI-compatible web servers to serve your project. See How to deploy with ASGIfor more details.

VIII. **mysite/wsgi.py**: An entry-point for WSGI-compatible web servers to serve your project. See How to deploy with WSGIfor more details.

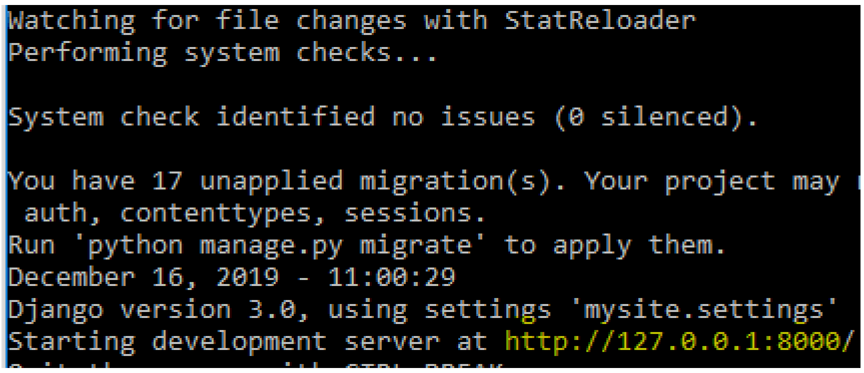
Step 2 Executing Django project

1. Let’s verify your Django project works. Change into the outer **mysite** directory, if you haven’t already, and run the following commands:



```
(myproject) D:\pyProject\mysite>python manage.py runserver
```

2. You will see the result as follow

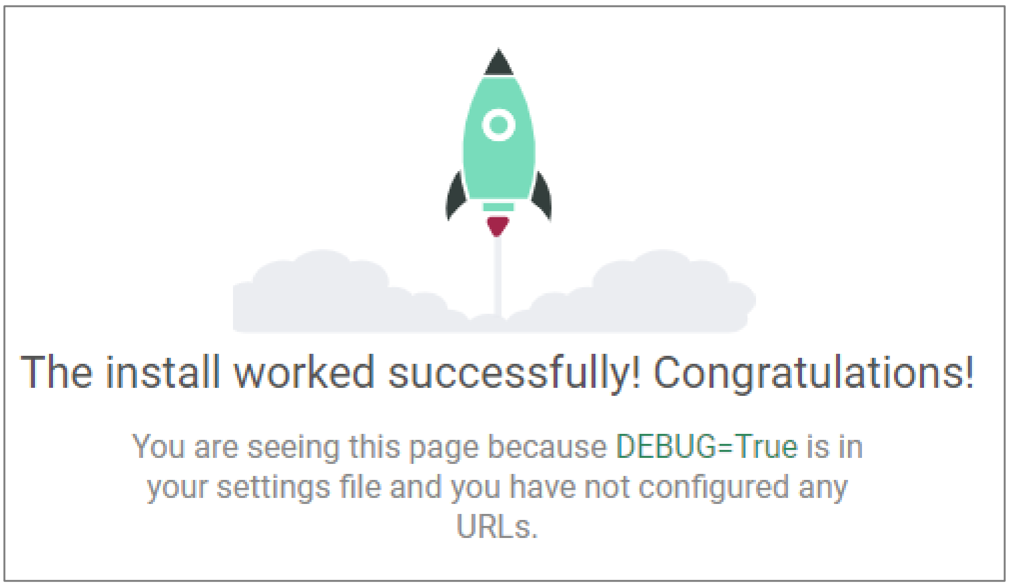


```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may
 auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 16, 2019 - 11:00:29
Django version 3.0, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
```

3. Copy the server address (as indicate in IP http://127.0.0.1:8000) and past it into any browser and get the following output



The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Creating the Polls app

Step 1 Setting up an app

Each application you write in Django consists of a Python package that follows a certain convention. Django comes with a utility that automatically generates the basic directory structure of an app, so you can focus on writing code rather than creating directories.

Projects vs. apps
What's the difference between a project and an app?
An app is a Web application that does something – e.g., a Weblog system, a database of public records or a small poll app.

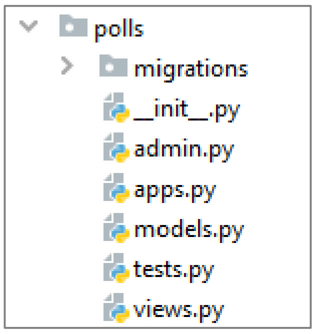
A project is a collection of configuration and apps for a particular website. A project can contain multiple apps. An app can be in multiple projects.

Your apps can live anywhere on your Python path. In this tutorial, we'll create our poll app right next to your **manage.py** file so that it can be imported as its own top-level module, rather than a submodule of **mysite**.

To create your app, make sure you're in the same directory as manage.py and type this command:

```
(myproject) D:\pyProject\mysite>python manage.py startapp polls
```

That'll create a directory polls, which is laid out like this:



```
polls
  migrations
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  views.py
```

Step 2 Write your first view

Let's write the first view. Open the file **polls/views.py** and put the following Python code in it:

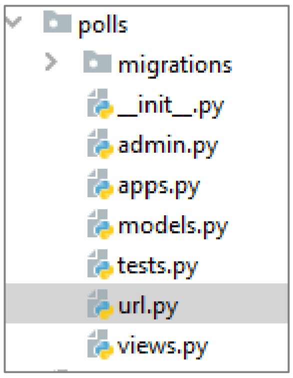
```
polls/views.py

from django.http import HttpResponse


def index(request):
    return HttpResponse("Hello, world. You're at the polls index.")
```

To call the view, we need to map it to a URL - and for this we need a URLconf.

To create a URLconf in the polls directory, create a file called **urls.py**. Your app directory should now look like:



```
polls
  migrations
  __init__.py
  admin.py
  apps.py
  models.py
  tests.py
  url.py
  views.py
```

In the **polls/urls.py** file include the following code:

```
polls/urls.py

from django.urls import path

from . import views

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

The next step is to point the root URLconf at the **polls.urls** module. In **mysite/urls.py**, add an import for **django.urls.include** and insert an **include()** in the **urlpatterns** list, so you have:

```
mysite/urls.py

from django.contrib import admin
from django.urls import include, path

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

Go to http://localhost:8000/polls/ in your browser, and you should see the text "*Hello, world. You're at the polls index.*", which you defined in the index view.

Student Task

Create your own Django Project (with your name) and an app as well under it, which we have learn in this lab