Python Django + Data Analytics

By: Curt Lebensorger
Curtis Lebensorger

- Data Science Graduate Student
- Graduate Assistant at Lewis University
- E-mail: curtismlebensorger@lewisu.edu
Overview

- Create a virtual environment
- Setup a Django project
- Load information into a MySQL database
- Extract information from another website’s API
- Display results using Google’s visualization web services
Prerequisites

- LAMP installed (Linux Apache MySQL PHP)
  - Linux - Operating system
  - Apache - Freely available web server
  - MySQL - Database for storing data
  - PHP - Server-side scripting language for web development
- Anaconda installed
  - Integrated Development Environment (IDE for Python)
Django

User Interaction

View

Controller

Fires events on

Manipulates

Model

Passes calls to
Virtual environments

- Encapsulating our project and its applications
- Control of packages being installed
Create a Virtual Environment

- Open a terminal
  - `conda create --name webdevelopment django`
    - conda specifies an Anaconda (IDE) command
    - We want Anaconda to create a virtualenv --name webdevelopment
    - Inside our virtualenv named webdevelopment, install the django package

- Activate Virtualenv
  - `source activate webdevelopment`
Install Packages

- Install a MySQL connector
  - `sudo apt-get install python-dev libmysqlclient-dev`
  - `sudo apt-get install python3-dev`
  - `pip install mysqlclient`
Remove Virtualenv Packages

- Remove the pre-built django database
  - conda remove --name webdevelopment sqlite
- We will be using a MySQL database
Create Django Project

- Open a terminal
- switch directories into our virtualenv
  - `cd anaconda3/envs/webdevelopment`
- Create project
  - `Django-admin startproject project`
Django Project Contents

- manage.py
  - A command-line utility that lets you interact with this Django project in various ways.

- settings.py
  - Settings/configurations for this Django project
Create Database

- Open a web webserver
- Type localhost/phpmyadmin
- Enter your MySQL user name and password
- Click on the SQL tab
- Enter the following command - `CREATE DATABASE tutorial;`
- Hit Ok
Go into the project’s setup.py module
Configure the mysql database

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'tutorial',
        'USER': 'curt',
        'PASSWORD': mysql.mysql_password,
    }

```
Configure MySQL to Django Project

- python manage.py migrate
Run Project

- python manage.py runserver
Create an Application

- An app is a Web application that does something.
- A project is a collection of configuration and apps for a particular Web Site.
- A project can contain multiple apps.
- An app can be in multiple projects.
- Run the following command in the terminal:
  ```
  python manage.py startapp mysite
  ```
Folders Created With Application

- models.py - Create database tables
- views.py - Views to display for users
Create Database Table

- open mysite/models.py

```python
class Iris(models.Model):
    sepal_length = models.FloatField()
    sepal_width = models.FloatField()
    petal_length = models.FloatField()
    petal_width = models.FloatField()
    iris_class = models.CharField(max_length=200)
```
Install Application (mysite) To Project

- Go into project/settings.py

INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'mysite',
)
Django Table to MySQL Table

- `python manage.py makemigrations`
  - Made changes to our models, and we like to store them as migrations
- `python manage.py sqlmigrate polls 0001`
  - Displays how the Django table will be stored as a MySQL table
- `python manage.py check`
  - Checks to see if there are any errors in your code
- `python manage.py migrate`
  - Creates the MySQL table along with the other pre-built tables created by Django.