Django Project Example Template - Installation Instructions

Nicholas Reitano ([reitanon@oregonstate.edu](mailto:reitanon@oregonstate.edu))

Local Website: http://127.0.0.1:8000/

Installation Instructions for Windows or Linux-based Operating Systems

1. Open a terminal (either Windows’ Command Prompt or a Linux-based shell) and navigate to the newly created directory with the extracted files.
2. If you have Python version 3.6 or higher already installed on your machine, skip this step.
   1. Otherwise, download and install python from: <https://www.python.org/downloads/>
      1. Install Python with all of the default recommended settings
   2. Once installed, be sure you set up the PATH variable for Python so you can enter commands in the terminal.
      1. For detailed instructions for this step, please view: <https://www.educative.io/edpresso/how-to-add-python-to-path-variable-in-windows>
3. Enter “virtualenv” in the terminal to check if it’s installed. If it is not recognized, run:
   1. pip install virtualenv
4. Enter the following command in the terminal to create a virtual environment:
   1. python -m venv venv
5. Enter the following command in the terminal to activate the virtual environment:
   1. Windows Operating System:
      1. .\venv\Scripts\activate
      2. If you receive an error that running scripts is disabled, enter the following before running the activation command (above) again: Set-ExecutionPolicy Unrestricted -Scope Process
   2. Linux-based Operating System (specifically a BASH shell):
      1. source venv/bin/activate
6. Enter the following command in the terminal to install all dependencies:
   1. pip install -r requirements.txt
7. Enter the following command in the terminal to launch the web site on your local machine:
   1. python manage.py runserver
8. In a browser, navigate to the following URL:
   1. <http://127.0.0.1:8000/>
9. Enter the following command to deactivate the virtual environment (Windows):
   1. deactivate

**Notes About Django in General:**

* Django uses a Model-View-Template structure.
  + Think of…
    - Model = Database
    - View = backend python code
    - Template = front end HTML/CSS (Jinja2 template engine)
* Django Applications:
  + Django applications are simply the folders that splits different core features from one another.
    - For instance, in the example, there is a Django App for “users” which handles the users account, profile, etc.. and there is also a Django App for “organizations” which houses the main features of the website: i.e. searching for charitable organizations, storing the user’s favorites, etc…
    - Splitting distinct features of the website into multiple Django Applications makes each application more flexible at the cost of developing the site a bit more complex.
      * For smaller projects, (like this), it may be easier to have all of our core features inside of one or two Django applications and then have another Django Application for the “users” like this example website demonstrates.
  + \_\_init.py\_\_ files:
    - Django needs these files to be present, but we don’t need to modify them.
  + Admin.py files:
    - These files configure the Django app to show up in the admin page
    - With the website running, if you navigate to <http://127.0.0.1:8000/admin>, you can log in and see the admin page where you can modify any of the database data.
    - I added both of you as superusers so you can log in as admins with the following credentials:
      * Username: Kayla
      * Password: testing321
      * Username: Richard
      * Password: testing321
  + Apps.py files:
    - Auto generated (I think…).
    - We don’t need to do anything with these files
  + Models.py files:
    - These are the database files that correspond to the given Django app.
      * i.e. the Organizations Application has a separate models.py file from the Users Application which has it’s own model.py file.
        + This makes it so the Django apps are extensible.
    - These files tell what the structure of the database should look like.
    - In the example, in the “organizations” application, there is a models.py file.
    - Text

      Description automatically generated
      * This file contains a single class, User\_Favorites and the name, User\_Favorites is the name of the database table.
      * The ukey, ein, and user are the names of table columns where the data are all textFields.
      * [Django](https://docs.djangoproject.com/en/3.2/topics/db/models/) has good documentation on different table columns and what not.
  + Tests.py files we can ignore and leave empty
  + Urls.py
    - This is the file that tells the project where the templates are located and what URL they should correspond to.
    - So if you create a new template, you need to update the urls.py file with the correct path.
  + Views.py
    - This is the file that acts as the backend
    - This is where you handle forms, API calls, and rendering pages and passing data into those templates.
* Django Project Structure:
  + Django\_project folder:
    - The only two files we care about are the settings.py and urls.py.
  + Media folder:
    - Used to store static images
  + The “organizations” folder, or the organizations Django app:
    - This folder is the core feature of the website and therefore houses the base template, the homepage, and other related pages such as the search feature.
    - This folder houses the base template which stores the navbar, footer, etc…
* The “Procfile” is used for Heroku hosting and this needs to be included in the main directory.
  + Nothing needs to be updated to it.
* The manage.py file is auto-generated and does not need to be updated.
* The requirements.txt file is a way to keep track of all of the python packages installed and this is used by Heroku to build the web app.
  + If a new package is installed, make sure to update this requirements.txt file by entering the following in the command line:
    - pip freeze > requirements.txt

And that’s it! I’ll update this document if I think of something else to add.