Wexdi SoftSkills

Introduction to Django

What is Django?

- Django is a Python web framework that helps you build high-performance, elegant and reusable code.

Advantages and features.

- **Complete and Versatile**
  - Django follows the "Batteries included" philosophy, providing almost everything developers might need out of the box. This includes a robust user authentication system, a dynamic administrative interface, and tools for generating RSS and Atom feeds
  - It can be used to build almost any type of website, from content management systems and wikis to social networks and news sites

Advantages and features.

- **Secure**
  - Django helps developers avoid many common security mistakes by providing a framework that has been engineered to protect the website automatically. It includes protection against SQL injection, cross-site scripting, cross-site request forgery, and clickjacking

Advantages and features.

- **Scalable**
  - Django uses a component-based "shared-nothing" architecture, meaning each part of the architecture is independent of the others. This allows it to scale for increased traffic by adding hardware at any level, such as caching servers, database servers, or application servers

Advantages and features.

- **Maintainable**
  - Django promotes the creation of maintainable and reusable code by following the Don't Repeat Yourself (DRY) principle. It encourages the grouping of related functionality into reusable "applications" and uses the Model View Template (MVT) architecture

Advantages and features.

- **Portable**
  - Django is written in Python, which runs on many platforms, including various flavors of Linux, Windows, and macOS. This means you are not tied to any particular server platform

Setting up a Django project and environment..

Wexdi SoftSkills

- **Prepare Your Environment**

  - When you're ready to start your new Django web application, create a new folder and navigate into it. In this folder, you'll set up a new virtual environment using your command line:

    $ python3 -m venv env

  - This command sets up a new virtual environment named env in your current working directory. Once the process is complete, you also need to activate the virtual environment:

    $ source env/bin/activate

  - If the activation was successful, then you'll see the name of your virtual environment, (env), at the beginning of your command prompt. This means that your environment setup is complete.

  - You can learn more about how to work with virtual environments in Python, and how to perfect your Python development setup, but for your Django setup, you have all you need. You can continue with installing the django package.

Setting up a Django project and environment..

Wexdi SoftSkills

- **Install Django and Pin Your Dependencies**
  - Once you've created and activated your Python virtual environment, you can install Django into this dedicated development workspace:

(env) $ python -m pip install django

- This command fetches the django package from the Python Package Index (PyPI) using pip. After the installation has completed, you can pin your dependencies to make sure that you're keeping track of which Django version you installed:

(env) $ python -m pip freeze > requirements.txt

- This command writes the names and versions of all external Python packages that are currently in your virtual environment to a file called requirements.txt. This file will include the django package and all of its dependencies.

Setting up a Django project and environment..

Wexdi SoftSkills

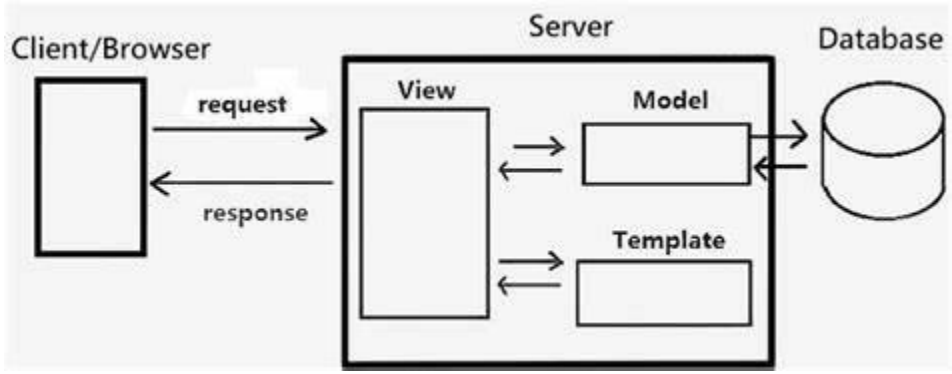
- **Install Django and Pin Your Dependencies**
  - **(env) $ python -m pip install django==2.2.11**
    - This command installs the version 2.2.11 of Django to your environment instead of fetching the most recent version. Replace the number after the double equals sign (==) with the specific Django version you need to install.
    - You should always include a record of the versions of all packages you used in your project code, such as in a requirements.txt file. The requirements.txt file allows you and other programmers to reproduce the exact conditions of your project build.
    - Suppose you're working on an existing project with its dependencies already pinned in a requirements.txt file. In that case, you can install the right Django version as well as all the other necessary packages in a single command:
  - **(env) $ python -m pip install -r requirements.txt**
    - The command reads all names and versions of the pinned packages from your requirements.txt file and installs the specified version of each package in your virtual environment.
    - Keeping a separate virtual environment for every project allows you to work with different versions of Django for different web application projects. Pinning the dependencies with pip freeze enables you to reproduce the environment that you need for the project to work as expected.

Understanding the MTV (Model-Template-View) architecture.

Wexdi SoftSkills

- Django is based on MVT (Model-View-Template) architecture. MVT is a software design pattern for developing a web application.

- MVT Structure has the following three parts –



- Model: The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySql, Postgres). To check more, visit – Django Models

- View: The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/Javascript and Jinja files. To check more, visit – Django Views.

- Template: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted. To check more, visit – Django Templates

Creating your first Django project.

Wexdi SoftSkills

- **Set Up a Django Project**
  - After you've successfully installed Django, you're ready to create the scaffolding for your new web application. The Django framework distinguishes between projects and apps:

  - A Django project is a high-level unit of organization that contains logic that governs your whole web application. Each project can contain multiple apps.
  - A Django app is a lower-level unit of your web application. You can have zero to many apps in a project, and you'll usually have at least one app. You'll learn more about apps in the next section.
  - With your virtual environment set up and activated and Django installed, you can now create a project:

  (env) $ django-admin startproject <project-name>
  - This tutorial uses setup as an example for the project name:

  (env) $ django-admin startproject setup
  - Running this command creates a default folder structure, which includes some Python files and your management app that has the same name as your project:

Django project structure and files explanation.

Wexdi SoftSkills

- **Set Up a Django Project**
- setup/
- │
- ├── setup/
- │ ├── __init__.py
- │ ├── asgi.py
- │ ├── settings.py
- │ ├── urls.py
- │ └── wsgi.py
- │
- └── manage.py

Django project structure and files explanation.

Wexdi SoftSkills

- **Set Up a Django Project**
  - In the code block above, you can see the folder structure that the startproject command created for you:

  - setup/ is your top-level project folder.
  - setup/setup/ is your lower-level folder that represents your management app.
  - manage.py is a Python file that serves as the command center of your project. It does the same as the django-admin command-line utility.
  - The nested setup/setup/ folder contains a couple more files that you'll edit when you work on your web application.

  - Note: If you want to avoid creating the additional top-level project folder, you can add a dot (.) at the end of the django-admin startproject command:

Running the development server.

(env) $ python manage.py runserver

This command is used to run the server