**Django Framework**

**What is a framework?**

A framework is essentially a structured foundation for the application development process. With the help of a framework, can avoid writing everything from scratch. Frameworks provide a set of tools and elements that help in the speedy development process. It acts like a template that can be used and even modified to meet the project requirements.

Why is Framework Used?

Writing the code from scratch is a tedious task full of possible risks and errors. You need to make the code clean, well-tested, bugs and errors free. It will be difficult for other developers to understand the code and work on it. So, it is better to work with the frameworks that meet your requirements. They make the development process easy with fewer errors and saves time. It is a general template that can be used and modified as per the requirement. It will be easy for others to understand your code as they are also familiar with frameworks.

* Frameworks provide many advantages such as:
* Easy to test your code and debug it.
* Clean code is much easy to understand and work with.
* Reduces redundancy of code in the project.
* Reduces the time and cost of the project with the enhanced application.
* Features and functionalities provided by the framework can be modified and extended.

**What is Django, what are the advantages of using Django?**

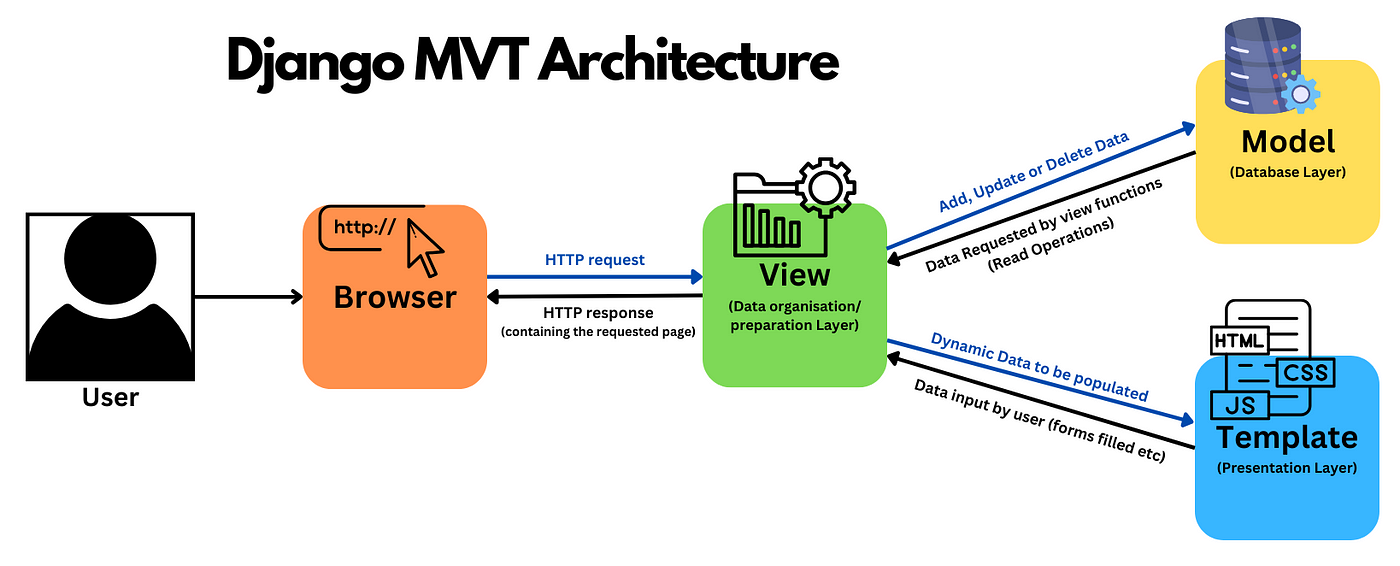
Django is a free and open-source, Python-based web framework that runs on a web server. It follows the  Model-View-Template (MVT) architectural pattern and emphasizes reusability of components, also referred to as DRY (Don't Repeat Yourself). Django comes with ready-to-use features like login system, database connection and CRUD operations (Create Read Update Delete).

Examples of Websites Using Django: Instagram, Disqus, Spotify, YouTube, Bitbucket, Dropbox, etc.

How does Django Work?

Django follows the MVT design pattern (Model View Template).

* **Model** - The data you want to present, usually data from a database. Django uses Object-Relational Mapping (ORM) to interact with the database, allowing developers to use Python code to handle database operations instead of writing raw SQL queries.
* **View** - The View is responsible for presenting data to the user and handling user input. In Django, views are Python functions or classes that take a web request and return a web response. They determine what data to display and which template to use for rendering. Views interact with the Model to retrieve or update data.
* **Template** - The Template is the presentation layer responsible for generating HTML to be sent to the client's browser. Django templates are used to define the structure and layout of the output. Templates can include dynamic content through the use of variables, loops, and conditional statements. They are processed by the Django template engine before being sent to the user.

****

1. User makes a request: When a user accesses a Django-powered website, their browser sends a request to the Django web server.
2. URL routing: Django uses a URL dispatcher to match the requested URL to a specific view. The URL dispatcher is configured in the urls.py file of the Django project.
3. View processing: The matched view function or class is executed. Views interact with the Model to retrieve or update data as needed. They also determine which template to use for rendering.
4. Template rendering: The selected template is processed by the Django template engine. Variables, logic, and template tags are replaced with actual data, creating an HTML page.
5. HTML response: The generated HTML, along with any static files (like CSS and JavaScript), is sent as a response to the user's browser.
6. User sees the result: The user's browser renders the HTML received from the server, displaying the final web page.
7. Database interaction: If the view involves database operations, Django handles the interaction with the database using the defined Models and the ORM.

**Advantages of Django Web Framework.**

1.   Fast and Rapid Development:

Django was intended to help developers make applications as fast as could be expected under the circumstances with less coding. It also provides a nice ready-to-use user interface for administrative activities which helps a lot in quick Web-Application development.

2.   Ready-Made Components:

Django provides many additional items you'll use to deal with normal Web development tasks. Django deals with client validation, content organization, site maps, RSS channels, and many other tasks which becomes too easy with the use of this framework.

3.   Security:

Django pays attention to security and assists developers with keeping away from numerous normal security problems, like SQL injections, cross-site scripting, cross-site request forgery, and clickjacking. Its client verification framework gives safe gratitude to manage client records and passwords.

4.   Scalable:

It can scale up to satisfy the heaviest traffic demands. Some of our busiest sites use the Django Web Framework just because of its scalable nature.

5.   Versatile:

Organizations, associations, and governments have utilized Django to make a wide range of things from Content Administration to Interpersonal Organizations.

6.   No-Repetition:

Make any component exactly one time only, after that we can use inheritance to use that pre-made component fast and efficiently.

7.   Neat and Clean Code:

Django carefully keeps all codes neat and clean with the help of its project structure and file management and makes it simple to follow best web-advancement rehearses.

8.   Object Relational Mapping (ORM) Support

Django gives an extension between the information model and the database motor and supports a huge arrangement of database frameworks including MySQL, Oracle, Postgres, and so forth. Django also supports for NoSQL database and there are some main NoSQL databases that are MongoDB and google firebase.

9.   Multi-Languages Support

With the help of Internationalization framework, Django supports multilingual websites.

**What is MVC and MVT pattern? What is the difference between them.**  
**MVC (Model-View-Controller) Pattern:**

The MVC pattern is a software architectural design pattern commonly used in web development. It separates an application into three interconnected components:

1. Model:
   * Represents the application's data and business logic. It manages the data, logic, and rules of the application. The Model receives input from the Controller, processes it, and sends the updated data to the View.
2. View:
   * Displays the data to the user and handles the presentation logic. The View receives data from the Model and presents it to the user. It also sends user input back to the Controller for processing.
3. Controller:
   * Manages user input and updates the Model accordingly. The Controller receives input from the user through the View, processes it, and updates the Model. It acts as an intermediary between the Model and the View.

**MVT (Model-View-Template) Pattern :**

Django, a web framework for Python, follows a similar but slightly different pattern known as MVT, where:

1. Model:
   * Similar to the MVC pattern, the Model represents the application's data structure and business logic. It interacts with the database using Django's Object-Relational Mapping (ORM) system.
2. View:
   * In Django's MVT, the View corresponds to the presentation logic and business logic that decides what data to display and how to display it. Views are responsible for processing user requests, interacting with the Model, and rendering the appropriate Template.
3. Template:
   * Unlike the traditional MVC pattern, Django introduces the Template component, which handles the presentation and generation of HTML. Templates are responsible for defining the structure and layout of the output, incorporating dynamic content.

**Difference between MVC and MVT design patterns :**

|  |  |  |
| --- | --- | --- |
|  | **Model View Controller (MVC)** | **Model View Template (MVT)** |
| 1. | MVC has controller that drives both Model and View. | MVT has Views for receiving HTTP request and returning HTTP response. |
| 2. | View tells how the user data will be presented. | Templates are used in MVT for that purpose. |
| 3. | In MVC, we have to write all the control specific code. | Controller part is managed by the framework itself. |
| 4. | Highly coupled | Loosely coupled |
| 5. | Modifications are difficult | Modifications are easy |
| 6. | Suitable for development of large applications but not for small applications. | Suitable both small and large applications. |
| 7. | Flow is clearly defined thus easy to understand. | Flow is sometimes harder to understand as compared to MVC. |
| 8. | It doesn’t involve mapping of URLs. | URL pattern mapping takes place. |
| 9. | Examples are ASP.NET MVC, Spring MVC etc. | Django uses MVT pattern. |

**What is a virtual environment? Why is it recommended to use the virtual environment in projects?**

A Virtual Environment is a tool that allows you to create isolated Python environments. It is recommended to use a virtual environment when working with Django or Python because it helps to manage **dependencies** and **avoid conflicts** with system-level Python installations. By creating a virtual environment, you can have a separate Python installation and install packages specifically for your application, without affecting the system’s Python installation or other applications. This allows you to work on multiple projects with different dependencies on the same system without any conflicts.

Here are key reasons why using a virtual environment is recommended for Python projects:

1. Isolation of Dependencies:
   * Different projects may require different versions of Python packages or libraries. By using a virtual environment, you create an isolated space for each project, preventing conflicts between dependencies. This ensures that each project has its own set of libraries, regardless of what is installed globally on the system.
2. Version Compatibility:
   * Some projects may rely on specific versions of Python. With virtual environments, you can easily switch between Python versions for different projects, allowing each project to use the version that is compatible with its requirements.
3. Dependency Management:
   * Virtual environments make it easier to manage dependencies for a project. You can specify the exact versions of packages needed for your project, and these dependencies are stored within the virtual environment. This helps ensure consistency across different development and deployment environments.
4. Ease of Sharing and Reproducibility:
   * When you share your project with others or deploy it to a server, including the virtual environment makes it easier for others to set up the project with the exact same dependencies. This enhances reproducibility and reduces the likelihood of issues caused by differences in the Python environment.
5. Cleaner Project Structure:
   * Including a virtual environment in your project directory keeps the project structure clean and self-contained. It makes it clear which dependencies are required for that specific project and facilitates easier collaboration with others.

**Create virtual environment in Django**

We should first go the directory where we want to create the [virtual environment](https://www.geeksforgeeks.org/python-virtual-environment/) then we type the following command to create virtual environment in Django.

**python -m venv env\_name**

then we need to activate virtual environment in Django

**Activate the virtual environment:** Run the activation script located in the bin directory within the virtual environment folder

**env\_name\Scripts\activate**