**Practical 1**

**APPLICATION DEVELOPMENT FRAMEWORKS**

2CSDE86

**Mistry Unnat**

20BCE515



Department of Computer Science and Engineering

Institute of Technology

Nirma University

Ahmedabad

**Aim :**

* **Introduction to MVC and MVT architecture.**
* **Comparative Study on MVC and MVT.**
* **Explore the Django application structure.**

**Also added comparison of various web development frameworks.**

**Introduction to MVC and MVT architecture**

**1. Model View Controller (MVC) :**

It is a software design pattern that is used to implement user interfaces and gives emphasis on separating data representation from the components which interact and process the data.

It has 3 components and each component has a specific purpose:

* This **Model** is the central component of this architecture and manages the data, logic as well as other constraints of the application.
* The **View** deals with how the data will be displayed to the user and provides various data representation components.
* The **Controller** manipulates the Model and renders the view by acting as a bridge between both of them.

**2. Model View Template (MVT) :**

This is yet another design pattern similar to MVC. It is also used for implementing web interfaces and applications but in contrast to MVC, the controller part is taken care for us by the framework itself.

It has 3 components and each component has a specific purpose:

* This **Model** similar to MVC acts as an interface for your data and is basically the logical structure behind the entire web application which is represented by a database such as MySql, PostgreSQL.
* The **View** executes the business logic and interacts with the Model and renders the template. It accepts HTTP request and then return HTTP responses.
* The **Template** is the component which makes MVT different from MVC. Templates act as the presentation layer and are basically the HTML code that renders the data. The content in these files can be either static on dynamic.

**Comparative Study on MVC and MVT**

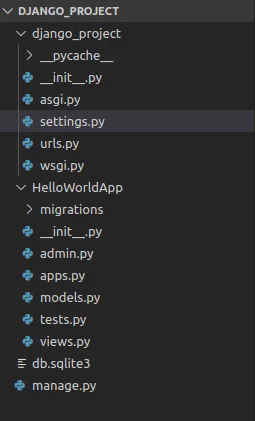
|  |  |
| --- | --- |
| **Model View Controller (MVC)** | **Model View Template (MVT)** |
| MVC has controller that drives both Model and View. | MVT has views for receiving HTTP request and returning HTTP  response. |
| Highly Coupled | Loosely coupled |
| Changes are difficult | Changes are easy |
| It doesn’t involve mapping of  URLs. | URL pattern mapping takes place. |
| Flow is clearly defined thus easy  to understand | Flow is sometimes harder to  understand as compared to MVC. |
| We have to write all control  specific code. | Controller part is managed by  framework itself. |
| **Ex:** ASP.NET, Spring MVC, etc. | Django uses MVT pattern. |

**Comparison of various web development frameworks**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Parameter** | **Django** | **React** | **Laravel** | **Flutter** | **Spring** |
| **Performance** | Moderate | Fast | High | High | Moderate |
| **Language** | Python | Javascript | PHP | Dart | Java |
| **Security** | High | High | Medium | High | High |
| **Framework** | Front end | Front  end | Front  end | Front  end | Backend |
| **Release**  **Date** | 2005 | 2013 | 2011 | 2017 | 2003 |
| **Open**  **Source** | Yes | Yes | Yes | Yes | Yes |
| **Architecture** | MVT | MVC | MVC | MVC | MVC |
| **PROS** | * Secure * Flexible * Scalable | * Extensive tool set | * Fast building cycle * Large community on Github | * Needs less code * Lest reload * response | * Flexible nature |
| **CONS** | Monolithic  Not the fastest one | No clear documentation | Execution problem | Contains limited  library | Complex to learn |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Ruby on**  **Rails** | **Vue.js** | **Angular** | **Express JS** |
| **Performance** | Slow | Fast | Moderate | Slow |
| **Language** | Ruby | Javascript | Javascript | Java  script |
| **Security** | High | High | High | Medium |
| **Framework** | Back  End | Frontend | Frontend | Backend |
| **Release**  **Date** | 2004 | 2014 | 2010 | 2010 |
| **Open**  **Source** | Yes | Yes | Yes | Yes |
| **Architecture** | MVC | CBA | MVC | MVC |
| **PROS** | * Preferred for Proto typing. * Rapid app develop ment | * Easy to discover errors * Clear documentation | * Quick development approach | * Flexible * Simple |
| **CONS** |  | It is way to flexible | Difficult to learn | Obstruct ive error message |

**Django App Structure**

* **Django App Structure :**

### ****\_init\_.py****

This file has the same functionality just as in the \_init\_.py file in the Django project structure. It remains empty and is present just to indicate that the specific app directory is a package.

### ****admin.py****

* As the name suggests, this file is used for registering the models into the Django administration.
* The models that are present have a superuser/admin who can control the information that is being stored.
* **This admin interface is pre-built and we don’t need to create it.**

### ****apps.py****

This file deals with theapplication configuration of the apps. The default configuration is sufficient enough in most of the cases and**hence we won’t be doing anything here in the beginning.**

### ****models.py****

* This file contains the models of our web applications (usually as classes).
* Models are basically the blueprints of the database we are using and hence contain the information regarding attributes and the fields etc of the database.

### ****views.py****

This file is a crucial one, it contains all the Views(usually as classes). Views.py can be considered as a file **that interacts with the client**. Views are a user interface for what we see when we render a Django Web application.

### urls.py

Just like the project **urls.py** file, this file handles all the URLs of our web application. This file is just to link the Views in the app with the host web URL. The settings **urls.py** has the endpoints corresponding to the Views.

### ****tests.py****

This file contains the code that contains different test cases for the application. It is used to test the working of the application.

**\*\* END \*\***