Model View Controller Design Pattern (MVC)
Compiled by: Rajashekar Tuppada
Document Reference Code: i3Q-SDTC-MVCA-0118

1. PURPOSE

The Purpose of this document is to provide the Overview of Model View Controller Design Pattern (MVC)

2. TARGET AUDIENCE

This document is targeted for the Developers and Designers involved in Java implementation for various applications

3. INTRODUCTION

Websites have shifted from simple HTML pages with a bit of CSS to incredibly complex applications with thousands of developers working on them at the same time.

To work with these complex web applications developers use different design patterns to lay out their projects, to make the code less complex and easier to work with.

The most popular of these patterns is MVC also known as **Model View Controller.**

4. WHAT IS MVC

Model-View-Controller (MVC) is an architectural design pattern that divides an application into three interconnected parts to separate business logic from the user interface design

Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other.

It was traditionally used for desktop **graphical user interfaces (GUIs)**. Nowadays, MVC is one of the most frequently used industry-standard web development frameworks to create scalable and extensible projects. It is also used for designing mobile apps.

5. FEATURES OF MVC

- It provides a clear separation of **business logic, UI logic, and input logic**
- It offers full control over your HTML and URLs which makes it easy to design web application architecture
- It is a powerful URL-mapping component using which we can build applications that have comprehensible and searchable URLs
- It supports **Test Driven Development (TDD)**

6. BENEFITS OF MVC

6.1. Organizes large-size web applications

As there is segregation of the code among the three levels, it becomes extremely easy to divide and organize web application logic into large-scale applications (which are required to be managed by large teams of developers). The major advantage of using such code practices is that it helps to find specific portions of code quickly and allows the addition of new functionality with ease

6.2. Organizes large-size web applications Supports Asynchronous Method Invocation (AMI)

Since the MVC architecture works well with JavaScript and its frameworks, it is no surprise that it also supports the use of Asynchronous Method Invocation (AMI), allowing developers to build faster loading web applications

It means that MVC applications can be made to work even with PDF files, site-specific browsers, and also for desktop widgets

6.3. Easily Modifiable

Using the MVC methodology allows easy modification of the entire application. Adding/updating the new type of views is simplified in the MVC pattern (as a single section is independent of the other sections). So, any changes in a certain section

© i3Qube Software 2023 Confidential

of the application will never affect the entire architecture. This, in turn, will help to increase the flexibility and scalability of the application

6.4. Faster Development Process

As there is segregation of the code among the three levels, developing web applications using the MVC model allows one developer to work on a particular section (say, the view) while another can work on any other section (say, the controller) simultaneously. This allows for easy implementation of business logic as well as helps to accelerate the development process fourfold. It has been observed that when compared to other development models, the MVC model ends up showing higher development speeds (up to three times)

6.5. Easy Planning and Maintenance

The MVC paradigm is helpful during the initial planning phase of the application because it gives the developer an outline of how to arrange their ideas into actual code. It is also a great tool to help limit code duplication, and allow easy maintenance of the application

6.6. Returns Data Without Formatting

By returning un-formatted data, the MVC framework empowers you to create your own view engine. For example, any type of data can be formatted using HTML, but with the MVC framework, you can also format the data using the Macromedia Flash or Dream viewer. It is helpful for the developers because the same components can be re-used with any interface

6.7. Supports TTD (Test-Driven Development)

A major advantage of the MVC pattern is that it simplifies the testing process by a great deal. It makes it easier to debug large-scale applications as multiple levels are structurally defined and properly written in the application. Thus making it trouble-free to develop an application with unit tests

6.8. Multiple Views

In the MVC architecture, developing different view components for your model component is easily achievable. It empowers you to develop different view components, thus limiting code duplication as it separates data and business logic

6.9. SEO-Friendly Plaform

The MVC platform hugely supports the development of SEO-friendly web applications. To generate more visits from a particular application, MVC provides an easy way out to develop SEO-friendly RESTful URLs

Thus, the MVC design pattern is a great approach for developing web applications. Today, there are several enterprises opting for the development of such applications based on the MVC architecture

7. COMPONENTS OF MVC

The MVC framework includes the following 3 components

- Controller
- Model
- View

7.1. Controller

The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary. The controller doesn't have to worry about handling data logic, it just tells the model what to do. It processes all the business logic and incoming requests, manipulates data using the **Model** component, and interact with the **View** to render the final output

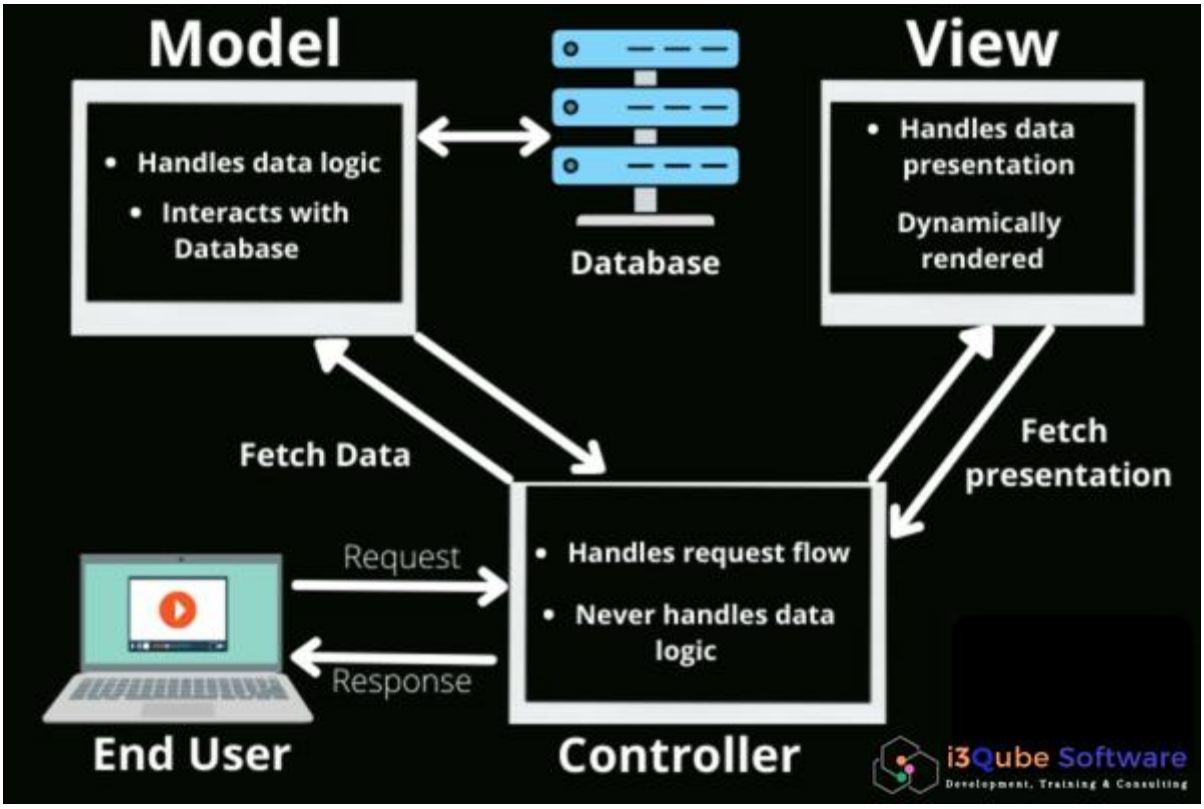
7.1.1. Responsibilities

- Receiving user input and interpreting it
- Updating the Model based on user actions

© i3Qube Software 2023 Confidential

- Selecting and displaying the appropriate View

Example
In a bookstore application, the Controller would handle actions such as searching for a book, adding a book to the cart, or checking out



MVC Architecture Design

7.2. View

The **View** component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data which is collected by the model component but these data aren't taken directly but through the controller. It only interacts with the controller

7.2.1. Responsibilities

- Rendering data to the user in a specific format
- Displaying the user interface elements
- Updating the display when the Model changes

Example:
In a bookstore application, the View would display the list of books, book details, and provide input fields for searching or filtering books

7.3. Model

The **Model** component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. It can add or retrieve data from the database. It responds to the controller's request because the controller can't interact with the database by itself. The model interacts with the database and gives the required data back to the controller

7.3.1. Responsibilities

- Managing data: CRUD (Create, Read, Update, Delete) operations

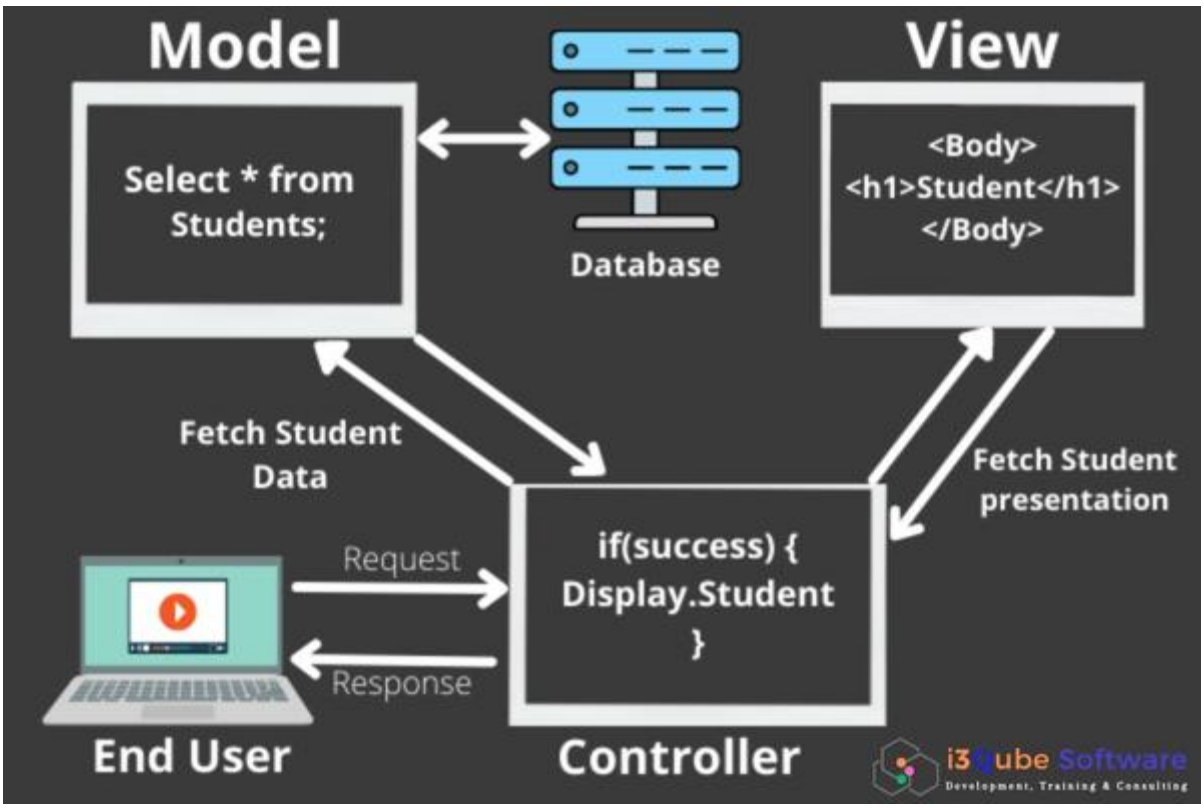
© i3Qube Software 2023 Confidential

- Enforcing business rules
- Notifying the View and Controller of state changes

Example
In a bookstore application, the Model would handle data related to books, such as the book title, author, price, and stock level

7.4. MVC Example

Let's imagine an end-user sends a request to a server to get a list of students studying in a class. The server would then send that request to that particular controller that handles students. That controller would then request the model that handles students to return a list of all students studying in a class



MVC Architecture Design Example

The model would query the database for the list of all students and then return that list back to the controller. If the response back from the model was successful, then the controller would ask the view associated with students to return a presentation of the list of students. This view would take the list of students from the controller and render the list into HTML that can be used by the browser

The controller would then take that presentation and returns it back to the user. Thus ending the request. If earlier the model returned an error, the controller would handle that error by asking the view that handles errors to render a presentation for that particular error. That error presentation would then be returned to the user instead of the student list presentation

As we can see from the above example, the model handles all of the data. The view handles all of the presentations and the controller just tells the model and view of what to do. This is the basic architecture and working of the MVC framework

The MVC architectural pattern allows us to adhere to the following design principles

1. Divide and conquer

   The three components can be somewhat independently designed

© i3Qube Software 2023 Confidential

2. Increase cohesion

   The components have stronger layer cohesion than if the view and controller were together in a single UI layer

3. Reduce coupling

   The communication channels between the three components are minimal and easy to find

4. Increase reuse

   The view and controller normally make extensive use of reusable components for various kinds of UI controls. The UI, however, will become application-specific, therefore it will not be easily reusable

5. Design for flexibility

   It is usually quite easy to change the UI by changing the view, the controller, or both

8. ADVANTAGES OF MVC

- Codes are easy to maintain and they can be extended easily
- The MVC **model** component can be tested separately
- The components of MVC can be developed simultaneously
- It reduces complexity by dividing an application into three units. **Model, view, and controller**
- It supports **Test Driven Development (TDD)**
- It works well for Web apps that are supported by large teams of web designers and developers
- This architecture helps to test components independently as all classes and objects are independent of each other
- **Search Engine Optimization (SEO)** Friendly

9. DISADVANTAGES OF MVC

- It is difficult to read, change, test, and reuse this model
- It is not suitable for building small applications
- The inefficiency of data access in view
- The framework navigation can be complex as it introduces new layers of abstraction which requires users to adapt to the decomposition criteria of MVC
- Increased complexity and Inefficiency of data

10. POPULAR MVC FRAMEWORKS

Some of the most popular and extensively used MVC frameworks are listed below

- Ruby on Rails
- Django
- CherryPy
- Spring MVC
- Catalyst
- Rails
- Zend Framework
- Fuel PHP
- Laravel
- Symphony

© i3Qube Software 2023 Confidential