Server-Side Web Framework

Avinash Maskey

Overview

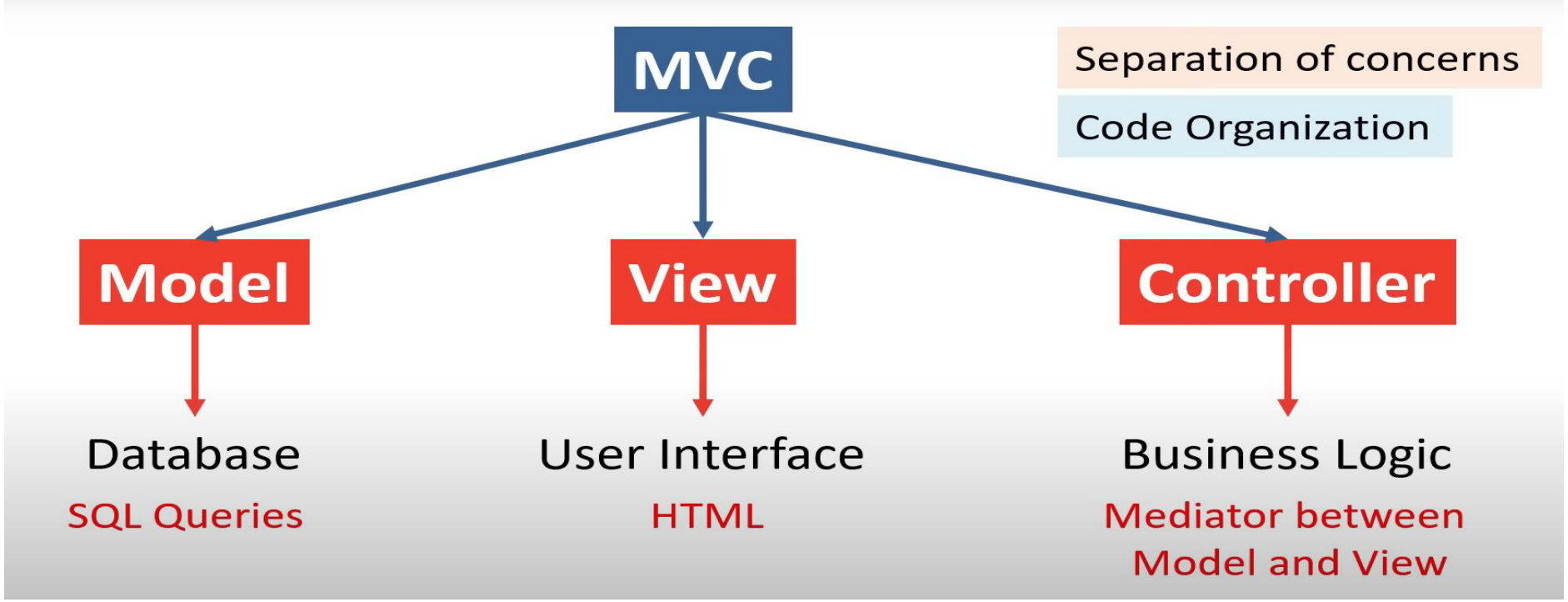
- A **server-side web framework** is a software framework designed to support the development of dynamic websites, web applications, and web services.
- A **framework** in programming is a tool that provides ready-made components or solutions that are customized to speed up development.
- It provides a standard way to build and deploy web applications on the internet. Server-side frameworks handle the HTTP requests from clients, process them on the server, and then send the response back to the client.
- This typically involves database interaction, backend logic, data processing, and rendering the appropriate HTML, CSS, and JavaScript for the client.

Key Components and Features of Server Side Framework

- **Routing:** Maps incoming requests to the appropriate controller and action/method.
- **Controllers:** Handle incoming requests, process data, and return responses.
- **Models:** Represent the application's data structure, usually mirroring database tables.
- **Views:** Templates for the HTML output that the application sends to browsers.
- **Middleware:** Filters that execute before or after controllers to modify requests or responses.
- **Database Abstraction:** Provides a way to interact with databases using high-level APIs instead of raw SQL.
- **Authentication and Authorization:** Manage user identification and permission levels.
- **Caching:** Temporarily stores content to reduce load times and database access.
- **Session Management:** Tracks user sessions across requests.
- **Error Handling:** Manages errors gracefully, providing useful feedback to users and developers.

What is MVC Framework?

- The **Model-View-Controller (MVC) framework** is a software architectural pattern that separates an application into three main logical components: **Model**, **View**, and **Controller**.
- Each of these components is built to handle specific aspects of the application's development.
- The MVC pattern is widely used in web application development with numerous benefits, including modularity, simplicity in managing large applications, and ease of maintenance.



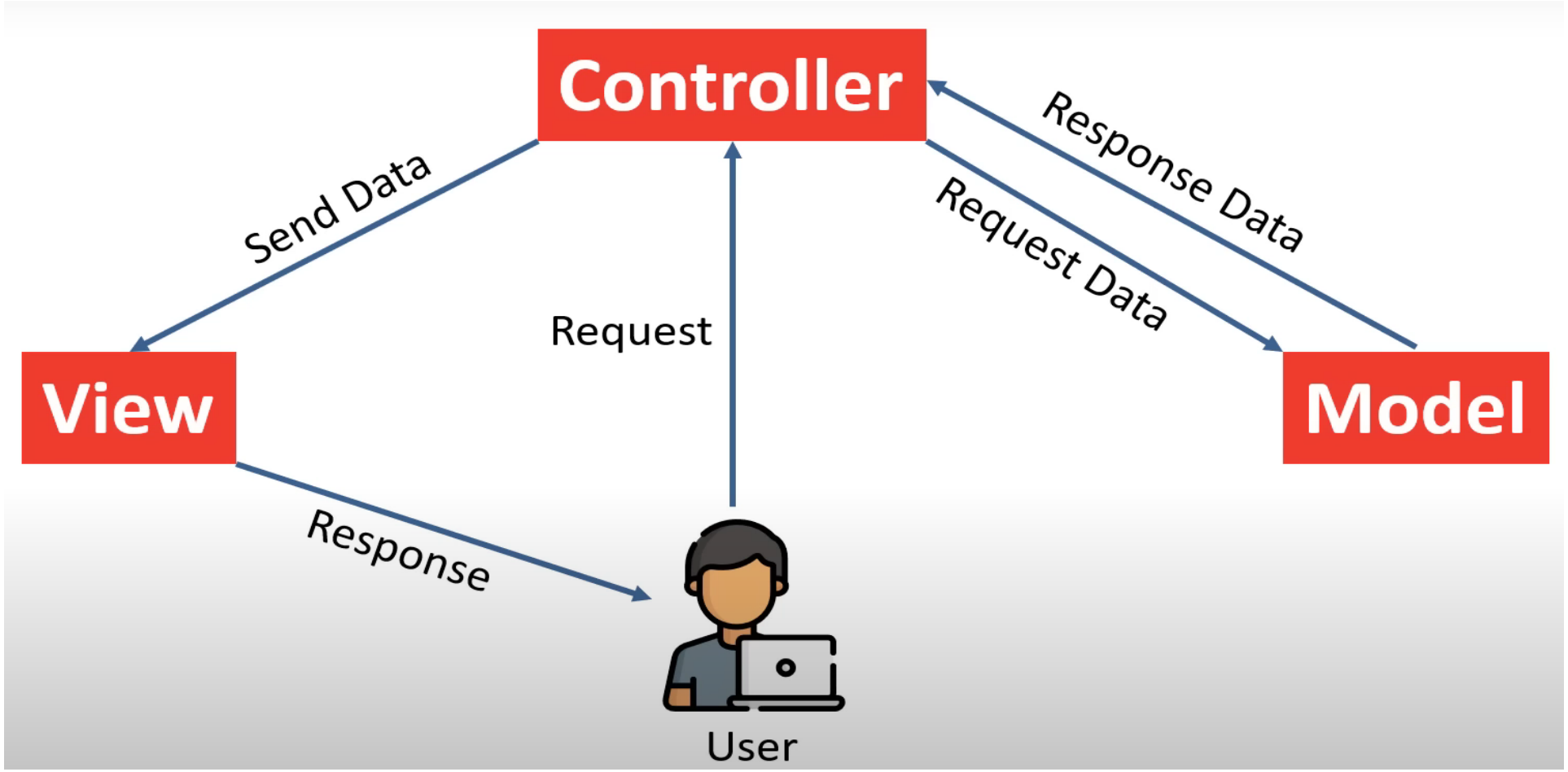
Components of MVC (1)

- **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 to the data.
  - For example, if you're building a book library system, the Model might represent a book with properties like title, author, and ISBN, and it might contain methods to fetch, save, or update the book data in the database.
- **View:**
  - The View component is used for all the UI logic of the application. For views, you create all the UI components that the users interact with. This includes everything from buttons and input fields to entire layouts and frames.
  - Using the book library example, the View could be the pages that show a list of books, a page for adding a new book, or a page for editing an existing book's details.

Components of MVC (2)

- **Controller**
  - The Controller acts as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output.
  - For instance, when a user wants to edit a book in the library, they interact with the View, which then uses the Controller to retrieve the book's details from the Model, and after the edit, it updates the book information using the Model again.

MVC Framework Workflow



Example of MVC in Action

- Let's illustrate this with a simplified example of an MVC application for managing a library:

  Scenario: A user wants to view a list of all books in the library.

  - **User Action:** The user navigates to the library's webpage to see a list of books.
  - **Controller:** The web server routes this request to the appropriate controller for handling book lists. The controller processes the request, asking the Model for information.
  - **Model:** The Model queries the database for a list of all books available in the library. This list is then returned to the Controller.
  - **Controller:** With the list of books from the Model, the Controller selects a View that is designed for displaying a list of books.
  - **View:** The Controller passes the list of books to the View. The View renders the HTML UI, which includes the list of books. This rendered View is sent back to the user's web browser, effectively showing the user the list of books in the library.

Benefits of MVC

- **Separation of Concerns:** By dividing the application into models, views, and controllers, MVC makes it easier to manage code (or organize code) because each part can be developed independently.
- **Support for Parallel Development:** Different developers can work on the Model, View, and Controller simultaneously, which speeds up the development process.
- **Ease of Modification:** Because of the separation, changing the UI does not affect the data handling, and vice versa. For example, redesigning the webpage (View) won't require altering the data models.
- **Improved Support for Test-Driven Development (TDD):** Each component can be tested independently, which makes unit testing and debugging easier.

Other Architectural Frameworks Used In Server-side Development (1)

- **Model-View-ViewModel (MVVM)**
  - Model-View-ViewModel (MVVM) is a structural design pattern that is somewhat similar to MVC but is more commonly used in applications with rich user interfaces, including web applications developed with frameworks like AngularJS or Knockout.js. MVVM facilitates a clear separation between the presentation logic and the business logic of an application.
    - ❖Model: Represents the data and business logic, similar to MVC.
    - ❖View: The UI layer that displays the data (the user interface).
    - ❖ViewModel: Acts as an intermediary between the Model and the View. It handles the view logic, converting the data from the Model into a format that the View can display.
  - The key component here is the ViewModel, which is responsible for exposing methods, commands, and other properties that help maintain the state of the View, manipulate the Model as results of actions on the View, and trigger events in the UI.

Other Architectural Frameworks Used In Server-side Development (2)

- **Model-View-Presenter (MVP)**
  - Model-View-Presenter (MVP) is a derivative of the MVC framework that introduces a Presenter between the Model and View to take on the burden of handling all UI logic. MVP is particularly popular in the development of Android applications.
    - ❖Model: Contains the business logic and data. It notifies the Presenter of any data changes.
    - ❖View: Displays the data (the UI) and notifies the Presenter of any user actions.
    - ❖Presenter: Acts as a middleman that retrieves data from the Model and formats it for display in the View. Unlike MVC's Controller, the Presenter also decides what happens when a user interacts with the View.
  - In MVP, the View more actively involves the Presenter in decision-making about UI changes, making the Presenter somewhat more heavyweight than the Controller in MVC.

Other Architectural Frameworks Used In Server-side Development (3)

- **Model-View-Adapter (MVA)**
  - Model-View-Adapter (MVA) is a variation that introduces an Adapter between the View and the Model to allow for more flexibility in how data is presented to the user. The Adapter transforms data from the Model to a form that is more suitable for the View.
    - ❖Model: The data layer.
    - ❖View: The presentation layer (UI).
    - ❖Adapter: Transforms data from the Model for the View. It also handles user actions and updates the Model accordingly.
  - MVA is less common but can be useful in scenarios where the data model and the presentation layer need to be kept at a significant distance from one another, such as in applications requiring the transformation of data into different formats for presentation.

Other Architectural Frameworks Used In Server-side Development (3)

- **Three-Tier Architecture**
  - The Three-Tier Architecture is a broader architectural pattern that divides applications into three logical and physical computing tiers: presentation, application, and data. This is more of an architectural style than a strict framework and can encompass MVC, MVP, or other patterns in its layers.
    - ❖Presentation Tier: The user interface and user interaction management (could employ MVC, MVP, or MVVM).
    - ❖Application Tier (Business Logic Layer): Handles all application operations, business logic, and client communications.
    - ❖Data Tier: Manages data persistence and database operations.
  - This architecture is widely used in enterprise applications for its scalability, security, and ability to distribute across multiple platforms.

Popular MVC Frameworks

- Laravel
- Symfony
- CodeIgniter
- Yii
- CakePHP
- Zend Framework

THANK YOU!