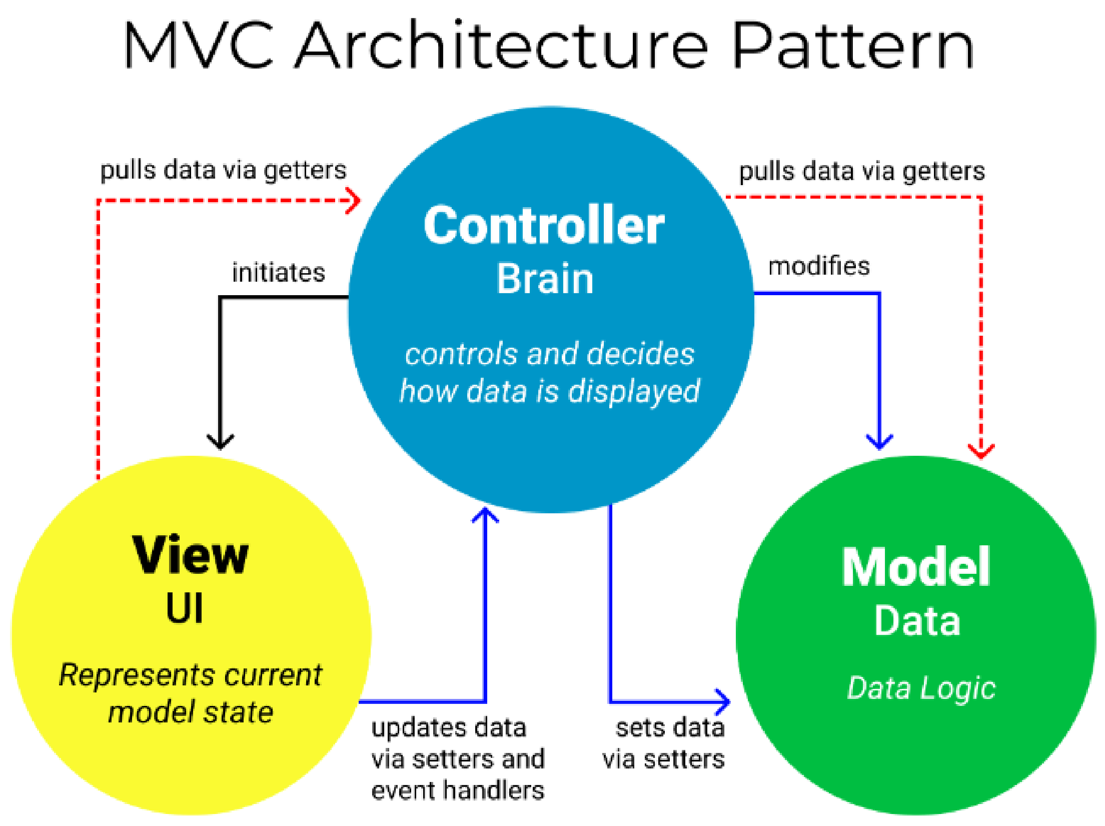
Architectual Pattern MVC

Author: Trisha Sarkar

MODEL VIEW CONTROLLER ARCHITECTURAL DESIGN PATTERN

The Model-View-Controller (MVC) is a design pattern used in software engineering to separate the concerns of an application into three interconnected components: the Model, the View, and the Controller. This separation allows for the modularization of code, making it easier to manage, test, and scale. MVC is widely used in web development frameworks like ASP.NET, Django, Ruby on Rails, and others. The concept of MVCs was first introduced by Trygve Reenskaug, who proposed it as a way to develop desktop application GUIs.

VISUAL REPRESENTATION



KEY COMPONENTS OF MVC

Model:

- The backend that contains all the data logic.
- It directly manages the data, logic, and rules of the application.
- The Model receives inputs from the Controller and updates itself and the View accordingly.

View:

- The frontend or graphical user interface (GUI).
- It displays the data to the user and sends user commands to the Controller.
- The View only represents the Model's output to the user.

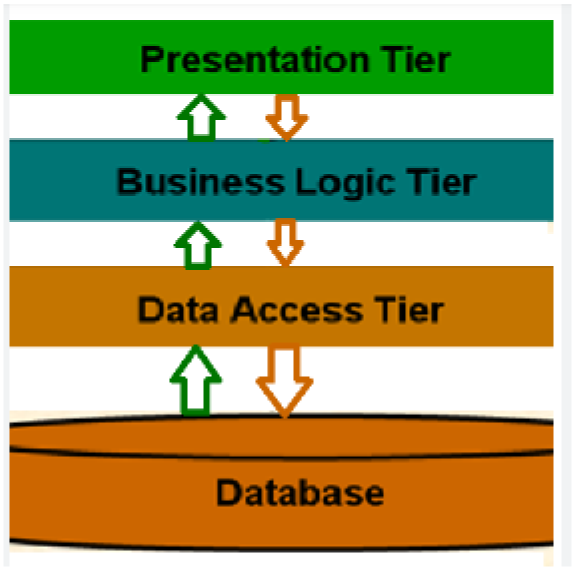
Controller:

- The brains of the application that controls how data is displayed.
- The Controller acts as an intermediary between the Model and the View.

- It listens to the user input from the View, processes it (including invoking changes in the Model), and updates the View.

LAYERED STRUCTURE OF MVC

The MVC pattern follows a layered structure, with each layer having distinct responsibilities:
Presentation Layer (View): Manages the visual representation and user interaction.
Business Logic Layer (Controller): Processes user inputs, applies business rules, and manages the flow of data.
Data Layer (Model): Manages data, including database interactions and state.



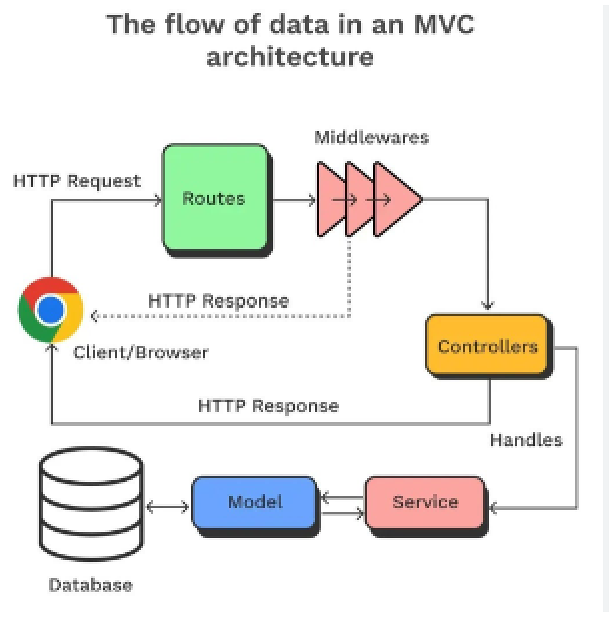
WHEN TO USE MVC?

MVC is suitable when:

- You need a separation of concerns for better code maintainability.
- You are building complex applications where multiple developers work on different parts simultaneously.
- The application requires multiple views for the same data.

HOW MVC PATTERN WORKS?

1. User Interaction: The user interacts with the View.
2. Controller Processing: The Controller receives the input from the View and processes it.
3. Model Update: The Controller updates the Model based on the input.
4. View Update: The Model notifies the View of any changes.
5. Display Update: The View updates the user interface.



BENEFITS OF MVC

- Separation of Concerns: MVC decouples data access, business logic, and UI, making it easier to manage and test.
- Scalability: Facilitates scaling by allowing modifications to individual components without affecting others.
- Reusability: Components can be reused across different applications.
- Testability: Each component can be tested independently

MVC vs MVT vs ECB vs CQRS

Aspect	MVC (Model-View-Controller)	MVT (Model-View-Template)	ECB (Entity-Control-Boundary)	CQRS (Command Query Responsibility Segregation)
Advantages	- Clear separation of concerns	- Simplicity and readability due to template usage	- Modular and highly reusable components	- Clearly separates read and write operations
	- Easy to maintain and scale	- Built-in support in frameworks like Django	- Effective in real-time and embedded systems	- Improves system scalability and performance
	- High reusability	- Easier to implement for web applications	- Offers good separation between UI, logic, and data	- Enables parallel development and separate scalability paths

	- Supports multiple views for the same model	- Promotes rapid development	- Facilitates real-time data processing	- Better aligns with complex business requirements
	- Good testability		- Supports complex UI requirements	- Enhances security by isolating different aspects of the application
Disadvantages	- Can become complex as application grows	- Less flexible than MVC	- Higher complexity in implementation	- Increased complexity due to separate models
	- Requires knowledge of front-end and back-end integration	- Logic is often embedded in templates, reducing testability	- Steep learning curve	- More complicated codebase with potential duplication
	- Tight coupling between Controller and View	- Not suitable for very large or complex systems	- Less common, limited community support	- Higher maintenance cost
Used When	- Building web and mobile applications	- Developing web applications with	- Implementing real-time, embedded systems, or complex UIs	- Creating complex systems with high

	with complex UI	frameworks like Django		scalability requirements
	- Requires a structured, testable architecture	- Need quick and efficient development	- Real-time or highly interactive system design	- Need to optimize performance and security for read/write operations
Why MVC is Better	- Provides a clean separation between logic and presentation	- Suitable for various platforms	- Better for complex UIs in real-time systems, but not as flexible	- Superior for maintaining code quality in complex projects
	- Allows independent development and testing of components	- Limited scalability and flexibility in MVT	- Limited use cases compared to MVC	- Not always necessary for simpler applications
	- Strong support and community			

CONCLUSION

The most attractive concept of the MVC pattern is separation of concerns.
Modern web applications are very complex, and making a change can sometimes be a big headache.
Managing the frontend and backend in smaller, separate components allows for the application to be scalable, maintainable, and easy to expand.

REFERENCES

FreeCodeCamp: Model-View-Controller