MVC
Architecture Pattern

By: Edwin Choi

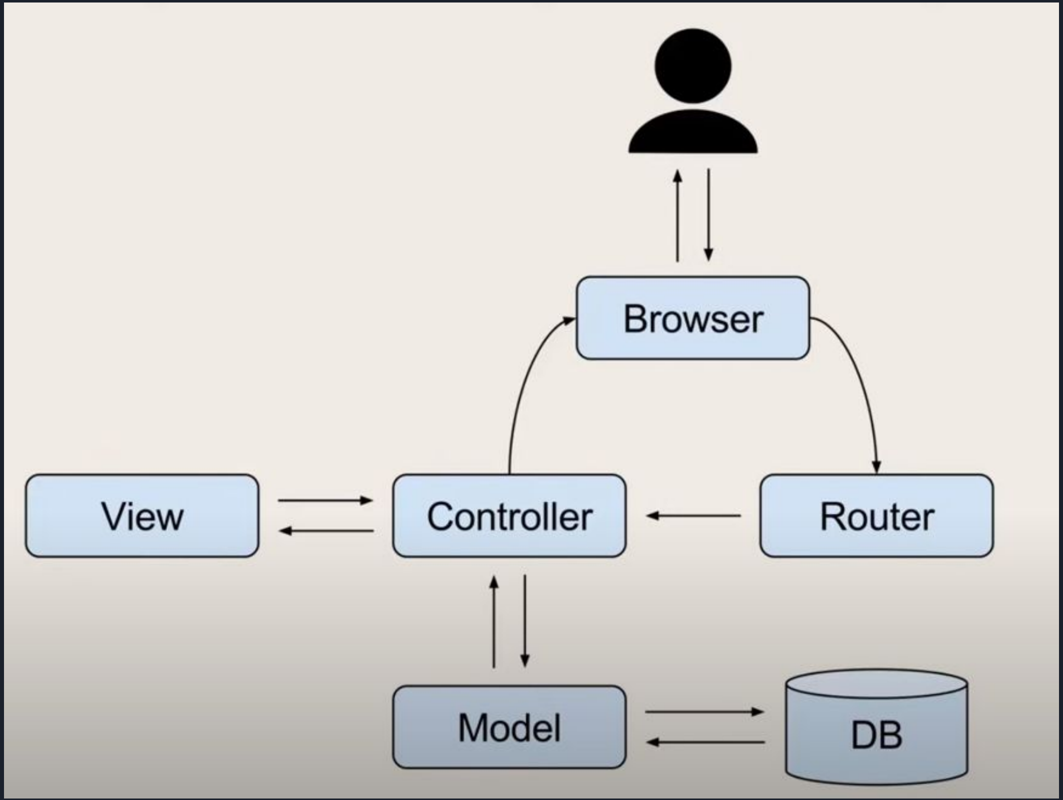
What is MVC?

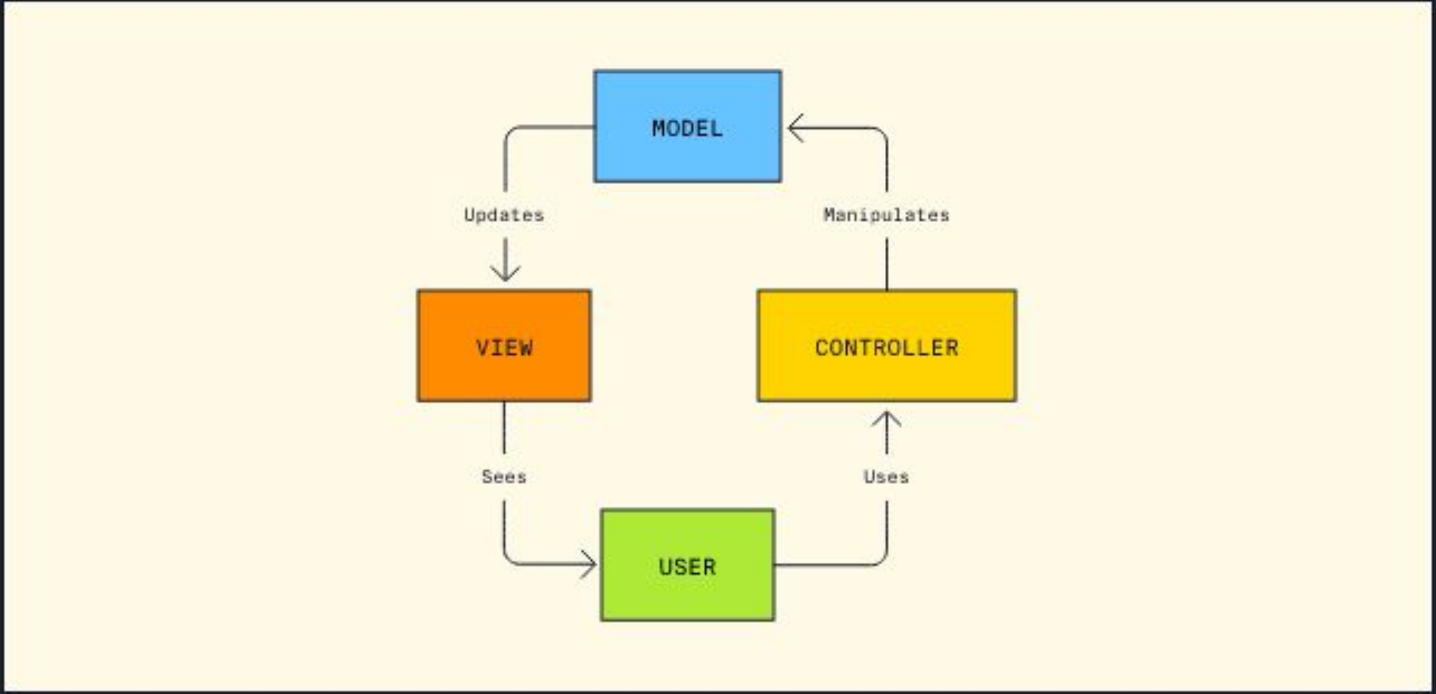
MVC, which stands for **Model-View-Controller**, is a coding architecture that converts sophisticated app development into a more simplified and manageable process.

Model: The back-end aspect responsible for containing all of the data logic

View: The front-end aspect or GUI (graphical user interface)

Controller: The think-tank of the app that controls how the data is displayed


Browser
View
Controller
Router
Model
DB


MODEL
Updates
Manipulates
VIEW
CONTROLLER
Sees
Uses
USER

The Benefits of MVC

Separation of Concerns - Allows you to divide the front-end and back-end code into individual components. This allows for code segments to be more easily managed, while ensuring that changes to one segment do not cause disruptions within others.

In essence, SoC separates software apps into distinct sections so that each section addresses a separate concern.

Simultaneous Workflow - It allows several developers to simultaneously work on any given app, without fear of overriding each other’s code.

Components of MVC
The Model (data)

The purpose of the model is to manage the data. It receives user input from the controller.

Regardless of where said originates from, whether it be a database, an API, or a JSON object, the model’s responsibility is to manage it.

The organization of the model is labeled as “schema”.

Components of MVC
Views (UI)

The purpose of the view is to decide what the user will see on their screen, and how.

In essence, it is responsible for rendering the presentation of the model in a particular format.

All of the functions that interact directly with the user can be considered core components of the view.

Components of MVC
Controller (Brain)

The purpose of the controller is to pull, modify, and provide data to the user.

The controller is essentially the middle-man between the view and the model.

Via getter and setter functions, the controller grabs data from the model and initializes the views.

If there are any updates from the views, the controller modifies the data with a setter function.

Examples of MVC
To-Do App

The **Model** in a To-Do app could define what a “task” is and how a “list” composes a collection of said tasks.

The **View** would define what the To-Do’s and lists appear like on a visual basis. The tasks might have bolded font and be a specific color.

The **Controller** would define how a user adds a new tasks, or marks another as being completed. It would also connect the View’s “Add Task” button to the Model, so that when the button is clicked, the Model adds a new task.

Examples of MVC
Thanksgiving Dinner

Model: Fridge - contains the raw materials needed to make the dinner.

View: Table, Silverware - the tools that allow your guests to eat (interact) with the dinner.

Controller: Recipe - dictates which materials in the fridge you’ll take out, how you’ll put them together, and how long they need to be cooked.

Summary

MVC architecture is excellent for organizing your code into nearly organized sections, based on their core functions. The end result is that your app becomes easier to read and make adjustments to, without the risk of small changes endangering large bundles of code.

MVC also lays the foundation for a developer to start translating his/her ideas into code, and also makes coming back to said code easier, since one can more easily identify which folder and its respective code does what.

By developing a system that allows you and other developers to more easily understand the code you write is essential to collaborating with other developers, and MVC is an important foundation of that skill.