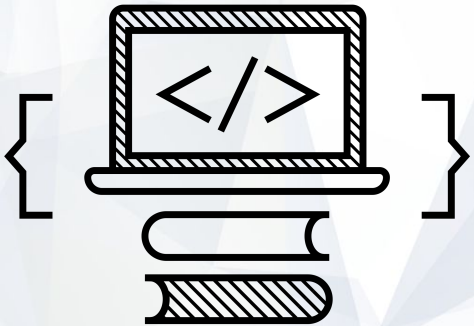
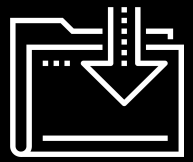


Model-View-Controller (MVC)

Coding Boot Camp

Module 14



What is the separation of concerns principle?

Separation of concerns is a design principle that each section of a program should address a separate concern.

Separation of Concerns

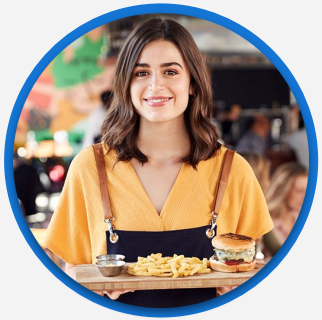
In a restaurant:

Chef

The chef's concern is to cook the food.

Server

The server's concern is to take orders and serve food.



Customer

A customer's concern is to order and eat.

We do not expect the customer to go back into the kitchen to cook food or the server to sit at a table and order, etc.

Layered Architecture in Software

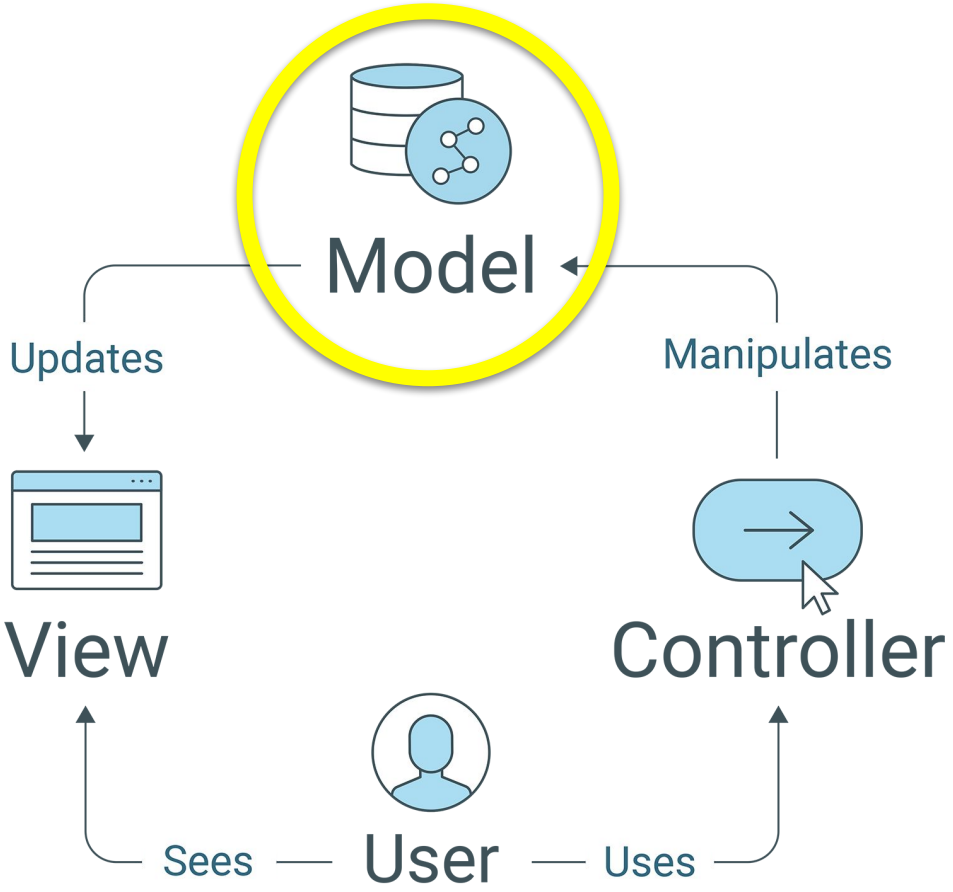
- Software design often uses a layered architecture.
- Each layer of the system handles its distinct concern.
- This structured approach aids in maintainability, scalability, and understandability.
- As we will see, MVC is a prominent example of such a layered architecture.

What is MVC?

The Model-View-Controller (MVC) framework is an architectural pattern that adheres to the separation of concerns principle.

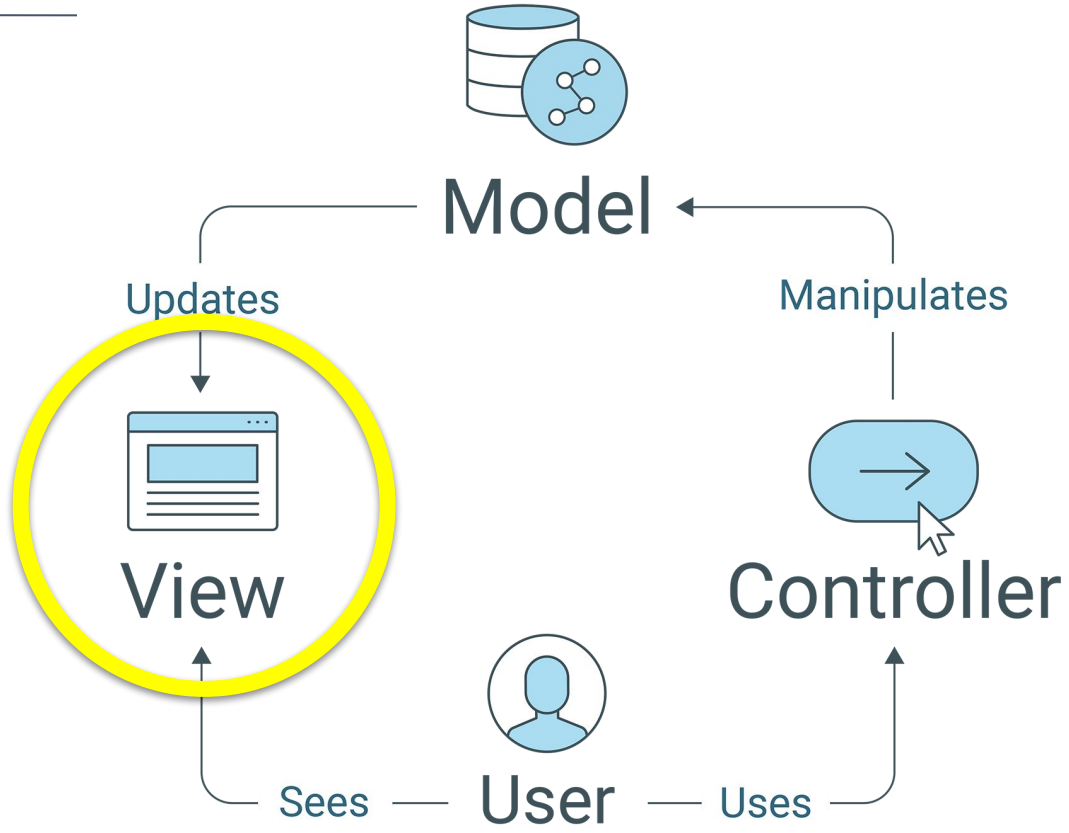
MVC Framework

The **Model** stores data and data-related logic.



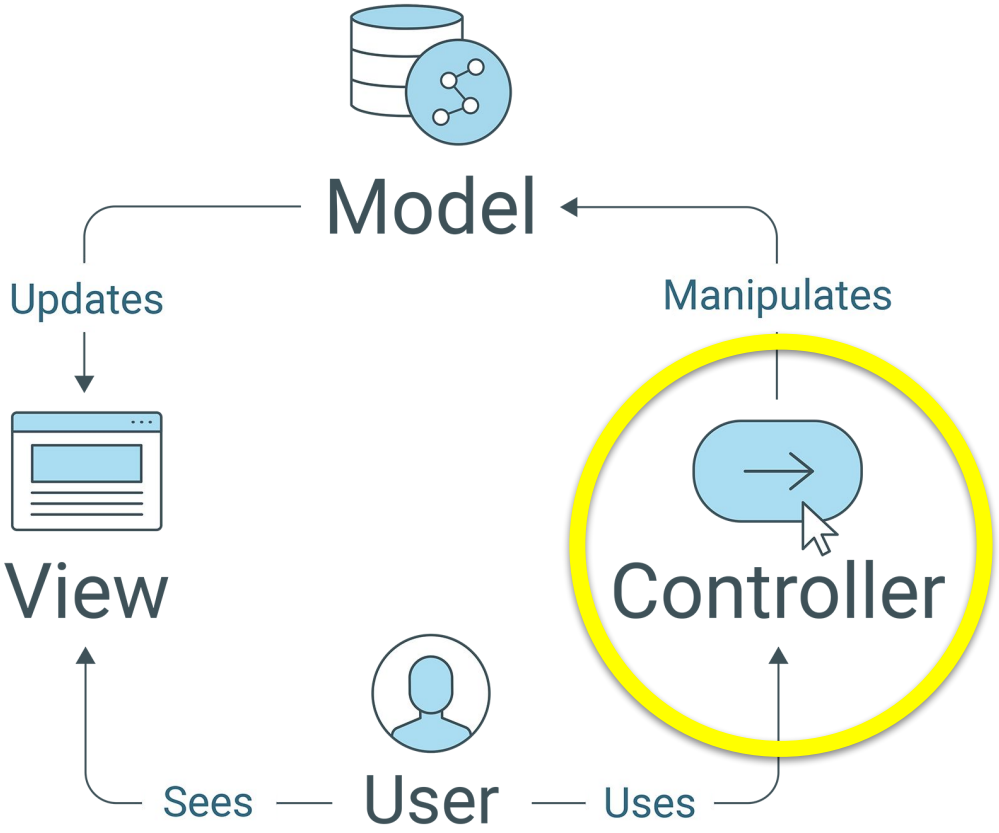
MVC Framework

The **View** is in charge of UI/UX concerns, or what a user will see and interact with.



MVC Framework

The **Controller** is the interface between Models and Views. It processes requests from the View, uses the Model to manipulate data, and sends data to the View to render.



What is a template engine?

A **template engine** is software that allows us to combine data with a static template to generate dynamic HTML.

Template Engines

Most template engines offer the following features:

Placeholders for data that we wish to include

Functions

Conditional rendering and looping

Text replacement

What are the advantages of using template engines?

Template Engines

Template engines provide the following benefits:

They help us follow the separation of concerns principle and MVC by providing an easy, clean way to separate HTML and JavaScript.

They offer tools that reduce repetition in code.

Templates are easy to create, use, and maintain.

They improve SEO and make fewer client-to-server requests.

What is Handlebars.js?

Handlebars.js is an extension of the templating language Mustache.

{{Handlebars.js}}

01

It is a logicless templating language that separates code from the View.

02

It compiles templates into a single resource and then returns the HTML after replacing variables with data.

03

It is a pure rendering engine—meaning that it has no built-in support for event handling, accessing back-end services, or making incremental DOM updates.

Why are we learning Handlebars.js?

Why Handlebars.js?

Handlebars.js provides the following benefits:

It gives us a great introduction to template engines because it is easy to use but offers a ton of functionality. It isn't the only option out there, but it is a great place to start!

It prepares us to encounter other languages that have this sort of templating built into them.

It helps us follow separation of concerns and the MVC framework.

It is a step towards learning to use heavier frameworks like React.js to build single-page applications.

How can we learn to implement MVC and Handlebars.js?

Handlebars.js and other template engines are designed to make implementing the MVC pattern easier.

How to learn MVC and Handlebars.js

You can try the following strategies to learn MVC and Handlebars.js:

Read the official documentation and practice with the provided examples.

Reverse-engineer finished code to see how something was accomplished.

Build something from scratch.

Debug a broken app.

And most importantly, ask questions!

Instructor Demonstration

Mini-Project