Web Application Architectures

Module 5: Middleware
Lecture 5: The Model-View-Controller Design Pattern

THE UNIVERSITY of NEW MEXICO

Web Applications Architectures

Early web application architectures did not have much organization on the server side:

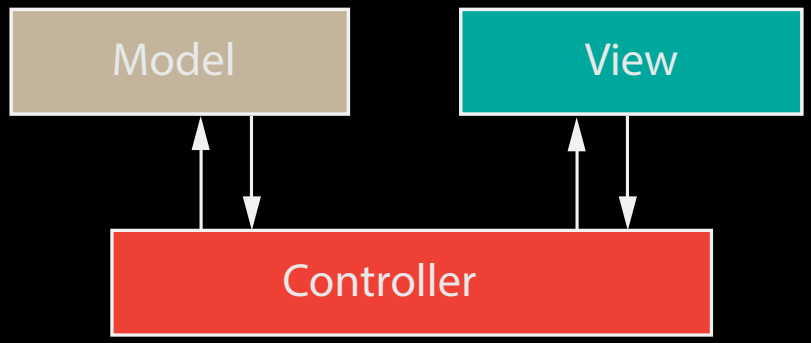
- Primarily fetching static web pages.
- No separation of data from its presentation.
- As applications became richer, server-side scripts became more complicated, and the applications became very difficult to maintain.

The Model-View-Controller (MVC) architectural design pattern, implemented over the middleware, is used in many web application frameworks and was introduced as a means of dealing with this complexity:

- Decouples data (model) and presentation (view).
- A controller handles requests, and coordinates between the model and the view.
- More robust applications, easier to maintain.

The MVC Design Pattern

The MVC design pattern is an architecture-level design pattern that is actually a collection of design patterns:



- Model – The domain-specific representation of the data over which the application operates, with domain logic that adds "meaning" to raw data. A database is often used to store the data.
- View – Renders the model in a view suitable for interaction, typically via a user interface. Multiple views can be created for a single model, each serving different purposes.
- Controller – Mediates between the model and the view.

The MVC Design Pattern

Although there are different varieties of MVC, the control flow is generally:

1. The user interacts with the user interface in some way (for example, by pressing a button).
2. The controller handles the input event from the user interface, often via a registered handler or callback, and converts the event into an appropriate user action, understandable for the model.
3. The controller notifies the model of the user action, possibly resulting in a change in the model's state. E.g., the controller may update the user's account information.
4. A view queries the model in order to generate an appropriate user interface (e.g., the user's account information). The view gets its own data from the model. In some implementations, the controller may issue an instruction to the view to render itself. In others, the view is automatically notified by the model of changes in state (Observer design pattern) that require a screen update.
5. The user interface waits for further user interactions, which restarts the control flow cycle.