Model-View-Control Pattern for User Interactive Systems

In various forms and guises
Contents

- Key Model-View-Control (MVC) concepts
- Web MVC
- UI Controller in MVC ≠ GRASP Application Controller
  - Two layers of “control”
- Other ways people think about MVC
“MVC” Key Separation of Concerns

- Evolution: Identifying Separate Concerns
  1. UI separate from application data and logic (model)
  2. Separate UI into user display components (views) and user input handling components (UI controls)
  3. For complex systems, provide an application controller to invoke and coordinate application functions
MVC Components

UI View
- Renders display, information
- create and update
- notify of changes

UI Control
- Actor Events
- delegate to, modify

Application Controller
- coordinate and invoke behavior, access data

Application Model
Other Interpretations of MVC

Consistent Design Decisions for a Given Context

- Wikipedia: “The MVC pattern has subsequently evolved, giving rise to variants such as hierarchical model–view–controller (HMVC), model–view–adapter (MVA), model–view–presenter (MVP), model–view–viewmodel (MVVM), and others that adapted MVC to different contexts.”

- MVC 5 (Microsoft .NET) [https://docs.microsoft.com/en-us/aspnet/mvc/mvc5](https://docs.microsoft.com/en-us/aspnet/mvc/mvc5)


- Model-View-ViewModel (Microsoft Presentation Framework)
  - Also known as Model-View-Binder

- Model-View-Presenter
  - [https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93presenter)
An Example Web MVC-like Architecture
Divide the UI Across Web Tiers

Browser-Side HTML5, JavaScript, CSS, Frameworks, JSON, etc.

Core application classes
- Plain-Old Java Objects (POJOs), etc.
- Get out of the UI framework and into the application as quickly as possible
Two Layers of Control

- **UI controller** (necessary): Route HTTP requests from a browser to the appropriate model and application controller objects
- **Application controller** (optional but common – use to manage complexity): Coordinate (“orchestrate”) complex interactions of objects and the services they provide – Business process workflow controller
  - Think about the sequence of tasks in an Amazon search and purchase
    - Task sequencing, service invocation, exception handling, Undo/Redo
  - Often implemented as a facade and/or mediator pattern
  - Think of as a GRASP Controller (mostly)
Remember the architecture for the 261 term project

- **Client UI**
  - User
  - HTML, CSS & JavaScript
  - Any Browser
  - Any OS/HW

- **Server UI**
  - Network Connection
  - Spark & FreeMarker
  - Java Web server (Jetty)
  - Any OS and HW

- **Application**
  - This is where you put the HTTP request handlers and view generation templates.
  - Application Controllers and Orchestration
  - Application behavior, data

- **Model**
MVC Lessons Learned

- MVC is powerful
  - (Separation of concerns is powerful)
- There are multiple design choices around MVC, allowing multiple design alternatives to the MVC pattern
- The Web throws complexity, and choice, into MVC design
- Understand the UI/Application separation of concerns offered by your chosen framework
  - Stick with the decisions of others, or you will suffer