Model View Controller (MVC)

Bigger than a Pattern: It’s an Architecture

Model View Controller (MVC)

- The intent of MVC is to keep neatly separate objects into one of tree categories
  - Model
    - The data, the business logic, rules, strategies, and so on
  - View
    - Displays the model and usually has components that allows user to edit change the model
  - Controller
    - Allows data to flow between the view and the model
    - The controller mediates between the view and model

Model

- The Model's responsibilities
  - Provide access to the state of the system
  - Provide access to the system's functionality
  - Can notify the view(s) that its state has changed

View

- The view's responsibilities
  - Display the state of the model to the user
- At some point, the model (a.k.a. the observable) must registers the views (a.k.a. observers) so the model can notify the observers that its state has changed

Controller

- The controller's responsibilities
  - Accept user input
    - Button clicks, key presses, mouse movements, slider bar changes
  - Send messages to the model, which may in turn notify it observers
  - Send appropriate messages to the view
- In Java, listeners are controllers

MVC Misunderstood

- MVC is understood by different people in different ways
- It is often misunderstood, but most software developers will say it is important; powerful
- Lets start it right, a little history, first Smalltalk

Smalltalk-80™

- In the MVC paradigm, the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of objects, each specialized for its task.
  – The **view** manages the graphical and/or textual output to the portion of the bitmapped display that is allocated to its application.
  – The **controller** interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate.
  – The **model** manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).

Smalltalk-80™ *continued*

- The formal separation of these three tasks is an important notion that is particularly suited to MVC Smalltalk-80 where the basic behavior can be embodied in abstract objects: *View*, *Controller, Model,* and *Object*
- MVC was discovered by Trygve Reenskaug in 1979

Sun says

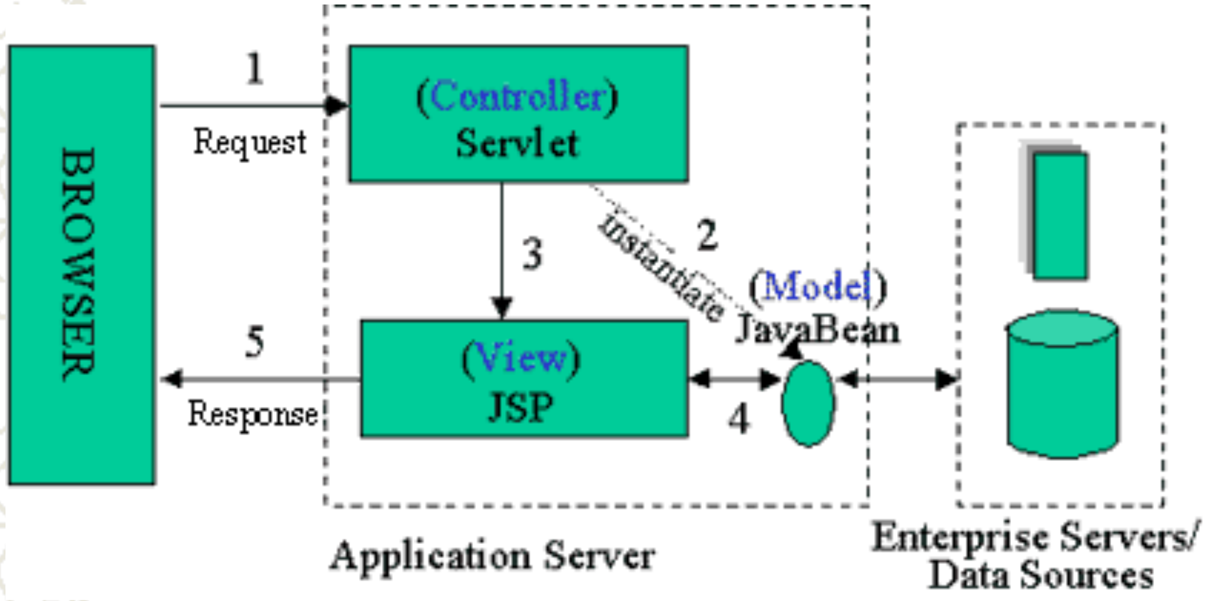
- Model-View-Controller ("MVC") is the recommended architectural design pattern for interactive applications
- MVC organizes an interactive application into three separate modules:
  - one for the application model with its data representation and business logic,
  - the second for views that provide data presentation and user input, and
  - the third for a controller to dispatch requests and control flow.

Sun continued

- Most Web application frameworks use some variation of the MVC design pattern
- The MVC (architectual) design pattern provides a host of design benefits

Java Server Pages

- Model: Enterprise Beans with data in the DBMS
  - JavaBean: a class that encapsulates objects and can be displayed graphically
- Controller: Servlets create beans, decide which JSP to return
  - Servlet object do the bulk of the processing
- View: The JSPs generated in the presentation layer (the browser)



OO-tips writes

- The MVC paradigm is a way of breaking an application, or even just a piece of an application's interface, into three parts: the model, the view, and the controller.
- MVC was originally developed to map the traditional input, processing, output roles into GUIs:
  - Console Input → Processing → Output
  - GUI Input → Controller → Model → View

Wikipedia writes

- **Model-View-Controller (MVC)** is a software architecture that separates an application's data model, user interface, and control logic into three distinct components so that modifications to one component can be made with minimal impact to the others.
- MVC is often thought of as a software design pattern. However, MVC encompasses more of the architecture of an application than is typical for a design pattern. Hence the term architectural pattern may be useful (Buschmann, et al 1996), or perhaps an aggregate design pattern.

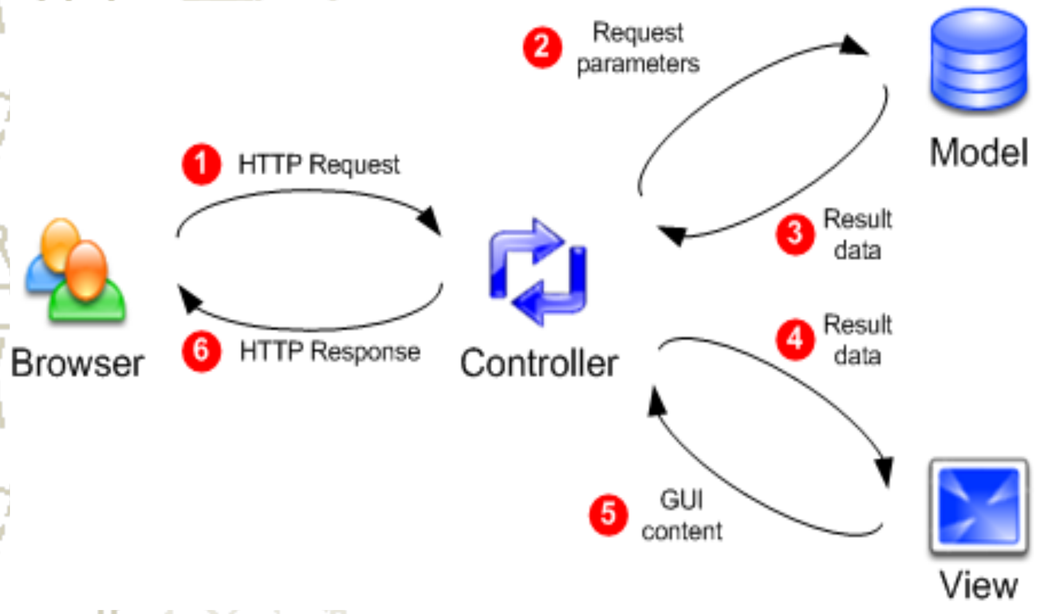
MVC Benefits

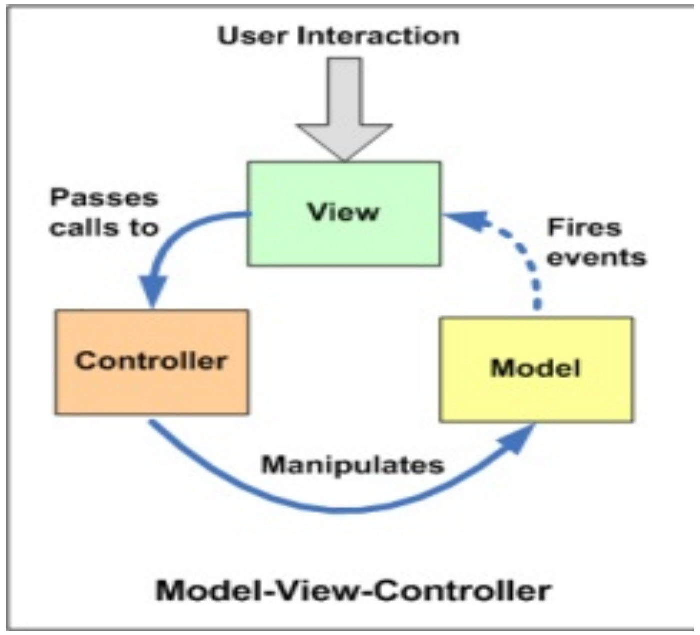
- Clarity of design
  - easier to implement and maintain
- Modularity
  - changes to one doesn't affect other modules
  - can develop in parallel once you have the interfaces
- Multiple views
  - spreadsheets, powerpoint, file browsers, games, Eclipse, UML reverse engineering, ….

Summary (MVC)

- The intent of MVC is to keep neatly separate objects into one of tree categories
  - Model
    - The data, the business logic, rules, strategies, and so on
  - View
    - Displays the model and often has components to allow users to change the state of the model
  - Controller
    - Allows data to flow between the view and the model
    - The controller mediates between the view and model

Two Views of MVC


1 HTTP Request
2 Request parameters
3 Result data
4 Result data
5 GUI content
6 HTTP Response
Browser
Controller
Model
View


User Interaction
View
Passes calls to
Fires events
Controller
Model
Manipulates
Model-View-Controller