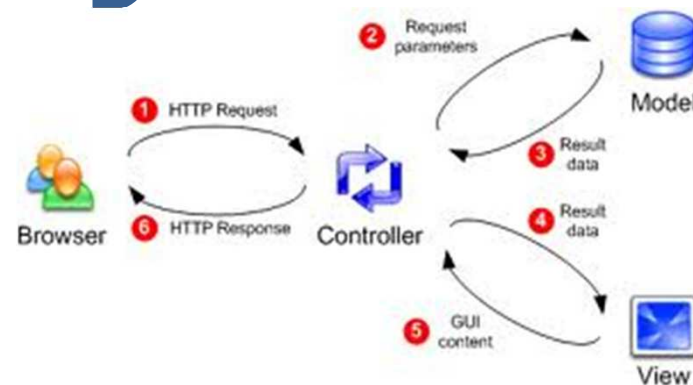
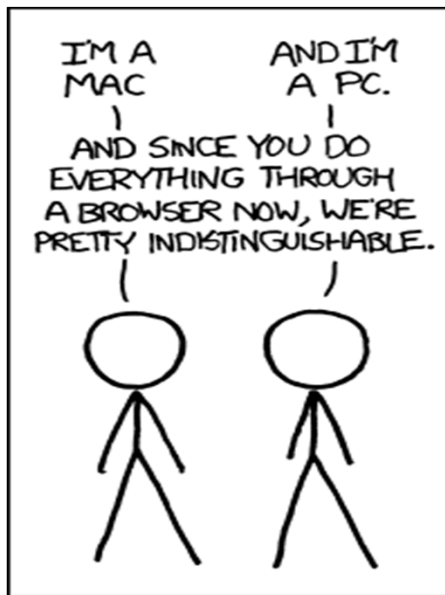




# MIS6070



# Web Based Information Systems



## Lesson 2



# Web based application Design

- ✿ Any complex system must begin with a cohesive fundamental design
  - ✿ *Anything you think that it is designed well/poorly in your world?*
- ✿ Multiple aspects of Web based application design
  - ✿ **Product Design:** how the overall application will function, including:
    - ✿ **Interaction (User-Interface) Design:** The operation, look, and feel of the user interface
    - ✿ **Software Design:** The required software components and how they will interact



# Web based application Design

- ✿ Earlier days coding was done in a flat way (**flat coding**) where **no standard was followed**.
- ✿ **Design related stuffs, business logic and database related queries all at the same place.**
- ✿ This approach is not good for bigger websites.



# Web based application Design

- ❖ Problems associated with flat coding.
  - ❖ No Modular approach, so not much scalability
  - ❖ Hard to isolate problematic area of application logic complex
  - ❖ Hard to maintain, if application logic is complex



# Web Application Design

- ❖ High level designs that govern the most basic notions of the structure and components of a system are usually referred to as **architectural styles**, e.g. **client server**.
- ❖ Low-level designs that govern **how individual components are build** are referred to as **design patterns**.



# Design Patterns

- ✿ A software **design pattern** is an abstraction of some commonly used software design
- ✿ This abstraction includes enough information to enable reuse of the pattern with new applications
- ✿ A pattern for architectural (high-level) design is also called an **architectural style**
- ✿ **Model-View-Controller (MVC) design pattern** has been a very successful basis for web applications.



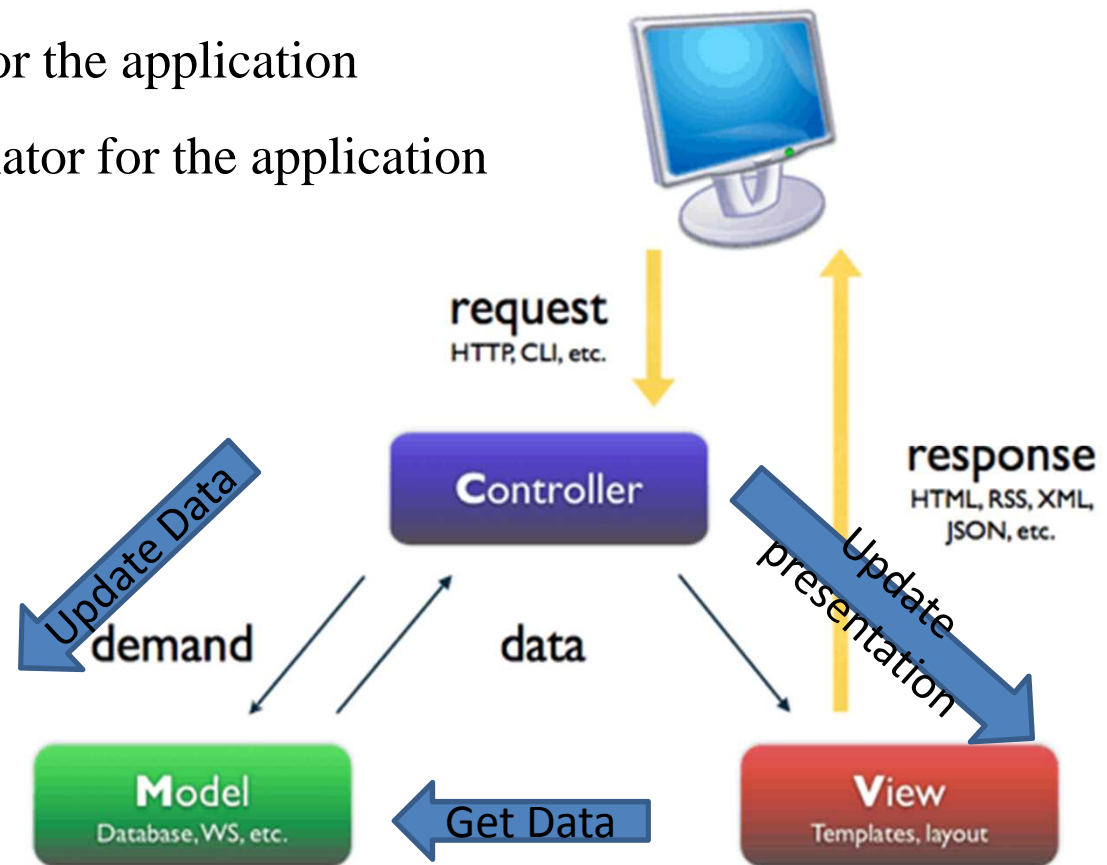
# Web Application Design

The MVC Pattern organises application functionality into three components

**Model:** Responsible for maintaining the state of the application

**View:** Present a user interface for the application

**Controller:** The central coordinator for the application



# Evolution of Web Application Design Architecture



✿ First phase of the evolution:

✿ **Just static HTML pages** were used to display static information.

✿ Then in the subsequent evolution phase:

✿ Dynamic contents generation technologies such as CGI initially

✿ *Common Gateway Interface (CGI)* is a standard method used to generate dynamic content on web pages and web applications

✿ Then servlet and JSP are introduced to generate and display dynamic contents along with static contents.

✿ A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across the HyperText Transfer Protocol.

✿ Java Server Page (JSP) is a technology for controlling the content or appearance of Web pages through the use of servlets.





# Page-centric Architecture

- ❁ Composed of a series of interrelated JSP pages
  - ❁ JSP pages handle all aspects of the application - presentation, control, and business process
- ❁ Business process logic and control decisions are hard coded inside JSP pages
  - ❁ In the form of JavaBeans, scriptlets, expression
- ❁ Next page selection is determined by
  - ❁ A user clicking on a hyper link, e.g. `<A HREF="find.jsp">`
  - ❁ Through the action of submitting a form, e.g. `<FORM ACTION="search.jsp">`



## Page-centric: Simple Application

- ✿ One page might display a menu of options, another might provide a form for selecting items from the catalog, and another would be to complete shopping process
- ✿ This doesn't mean we lose separation of presentation and content



# Servlet-centric Architecture

- ✿ In servlet-centric architecture, *JSP pages are used only for presentation and control* and *business process are handled by a servlet or a set of servlets*.
- ✿ So here, this servlet serves as a gatekeeper providing common services such as authentication, logging, translation, and transformation for all incoming requests.
- ✿ How Do I Decide when to?
- ✿ Use page-centric
  - ✿ If the application is simple enough that links from page to page.
- ✿ Use servlet-centric
  - ✿ Each link or button click requires a great deal of processing and decision-making about what should be displayed next.

# Web Application Design

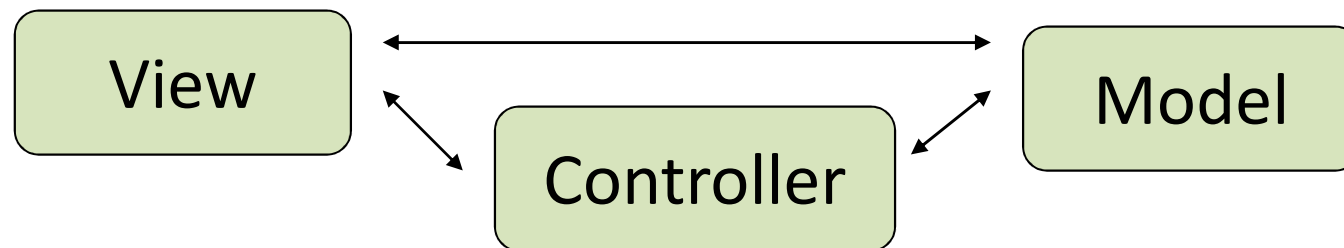


- ✿ In recent years, MVC has become a popular strategy for building websites.
- ✿ The **Model View Controller (MVC)** design pattern is a way of separating the user-interface from the substance of the application.
- ✿ There are now web-MVC frameworks available for many programming languages, for instance **Struts for Java**, **Maypole for Perl** and **Rails for Ruby**.
- ✿ MVC is an object-oriented design pattern
- ✿ The idea of the MVC pattern is to divide an application into 3 parts: the Model, View and Controller



# Model-View-Controller

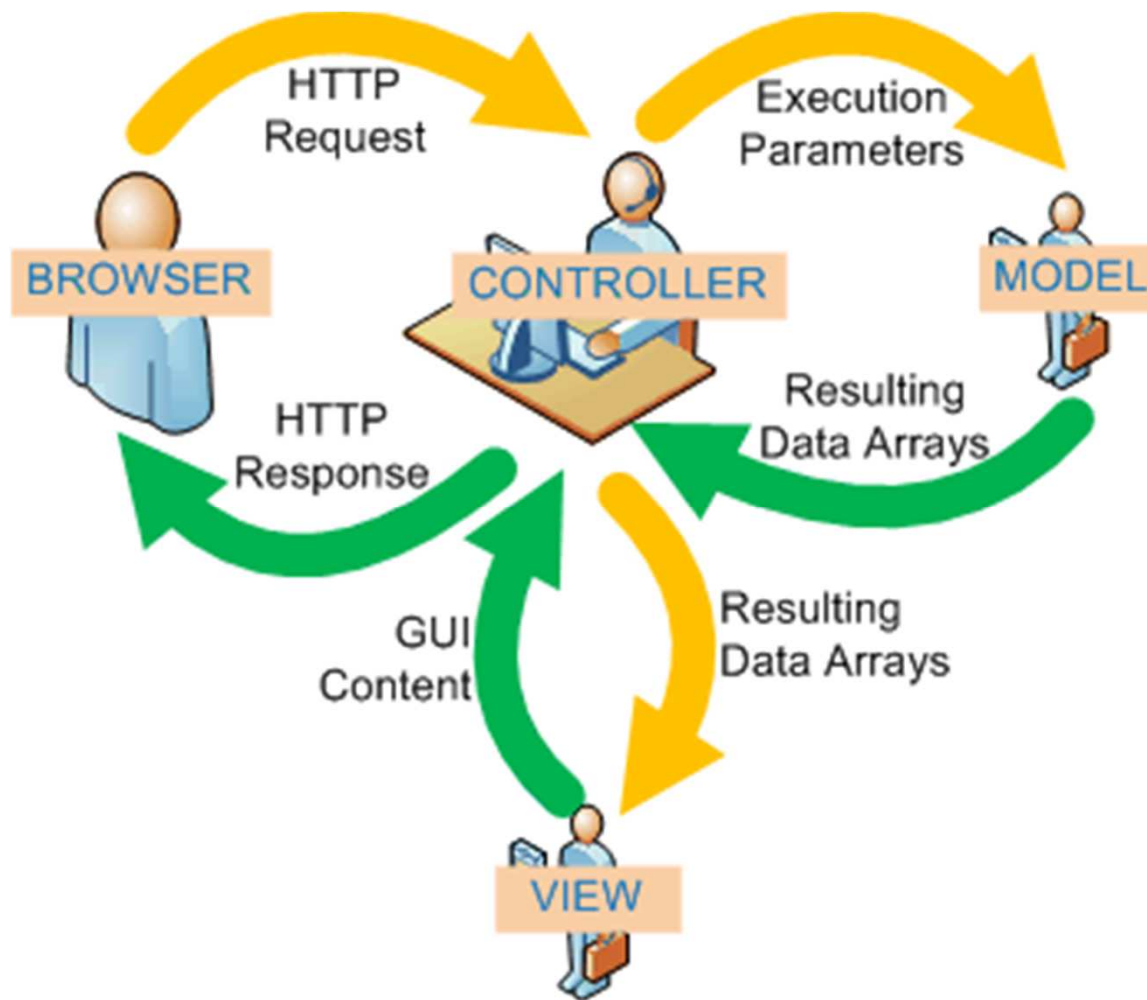
- Many Web based applications are based on the well-known **model-view-controller** design pattern



- The MVC pattern specifies the responsibilities of three major components, and the nature of their interactions



# Model-View-Controller

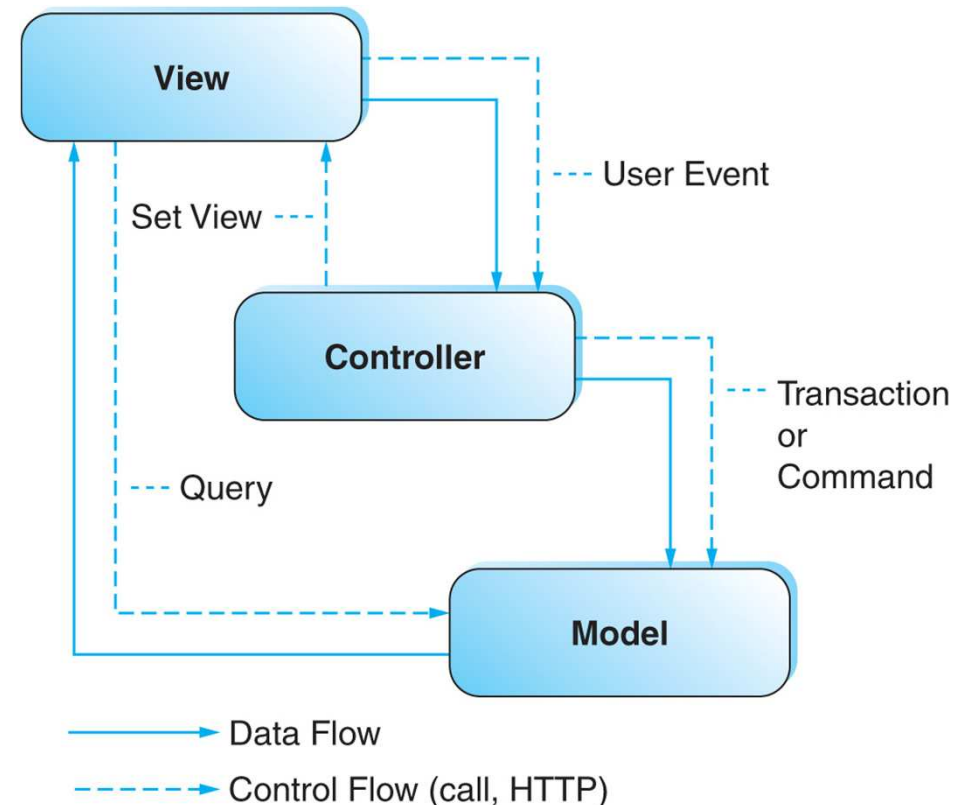


# MVC Component Responsibilities

*Shows the flow of data and control through the components.*

*A control flow is a pathway for issuing requests while data flow is a pathway for sending application data*


- ✿ **View:** Present the user interface
- ✿ **Model:** Maintain the application state
- ✿ **Controller:** Handle user actions





# Model-View-Controller



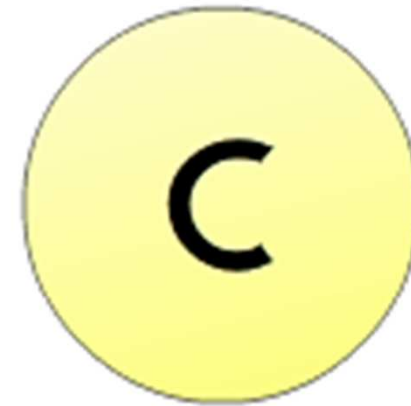
 model

database related,  
not necessary  
database. Data  
can be XML or  
even text files



 view

your website  
design/ HTML  
files. NO images,  
css, etc.here.  
only html layout



 controller

application logic,  
process the user  
request and get  
appropriate data,  
then output a design  
from View





# Model View Controller (MVC)

## ❁ Model

- ❁ The Model handles the state of the application.
  - ❁ **The state is what your application is *about*.**
- ❁ The Model does not know anything about HTML, or web servers or anything like that.
  - ❁ **It just supplies ways to query the state, and ways to change that state.**
- ❁ It entails both storing the data that constitute the application state such as the database and handling the transactions that affect the state known as application logic or business logic.



# Model

## ❄ **Model** (Business process layer)

- ❄ Models the data and behaviour behind the business process
- ❄ Responsible for actually doing
  - ❄ Performing DB queries (perform database access)
  - ❄ Calculating the business process
  - ❄ Processing orders
- ❄ Encapsulate of data and behaviour which are independent of presentation



# Model Role

- ❖ The Model maintains the application state, which includes:
  - ❖ persistent information stored in databases
  - ❖ current information related to active sessions
- ❖ Responsibilities include:
  - ❖ applying rules / transactions to modify state
  - ❖ providing information to the View as required
  - ❖ taking instructions from the Controller



# View

- ✿ The View represents presentation layer.
- ✿ It presents a user interface for the application
- ✿ The view handles displaying information according to client type.
- ✿ The view displays result of business logic processing, that is Model.
- ✿ Since View is independent of Model, the view is not concerned with how and where the information in Model was obtained since that is the responsibility of Model.



# View Role

- ❖ The View presents a user interface, including information and transactional controls
- ❖ Responsibilities include:
  - ❖ Present required information to user
  - ❖ Request data from Model as needed (to create displays for the user)



# Controller

- ❖ Serves as the logical connection between the user's interaction and the business services on the back
  - ❖ Responsible for making decisions among multiple presentations
  - ❖ e.g. User's language, locale or access level dictates a different presentation.
- ❖ A request enters the application through the control layer, it will decide how the request should be handled and what information should be returned

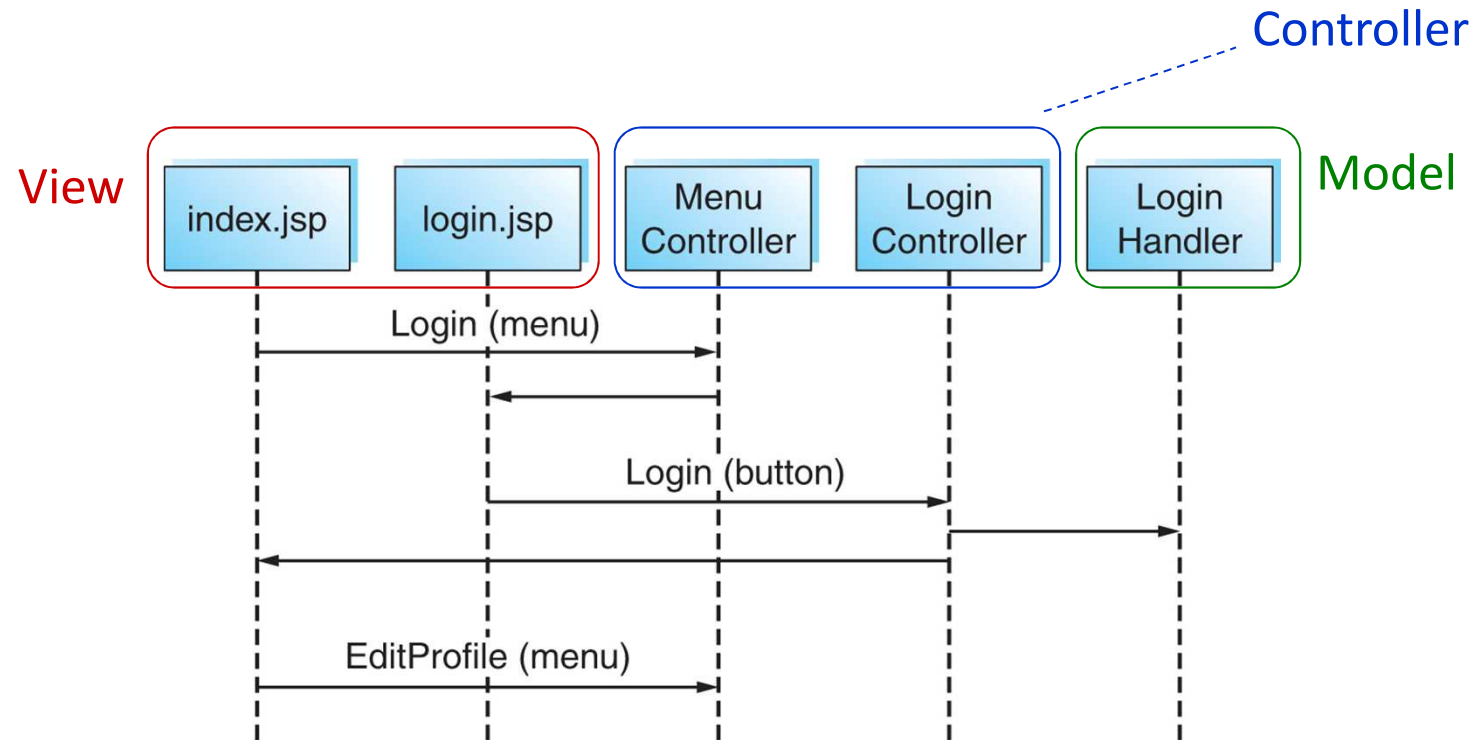


# Controller Role

- ❖ The Controller handles user actions
- ❖ Responsibilities include:
  - ❖ Handle user actions
  - ❖ Validate user requests (correctness, completeness)
  - ❖ Invoke Model components to handle requested transactions
  - ❖ Set the appropriate next View perspective



# MVC in Action

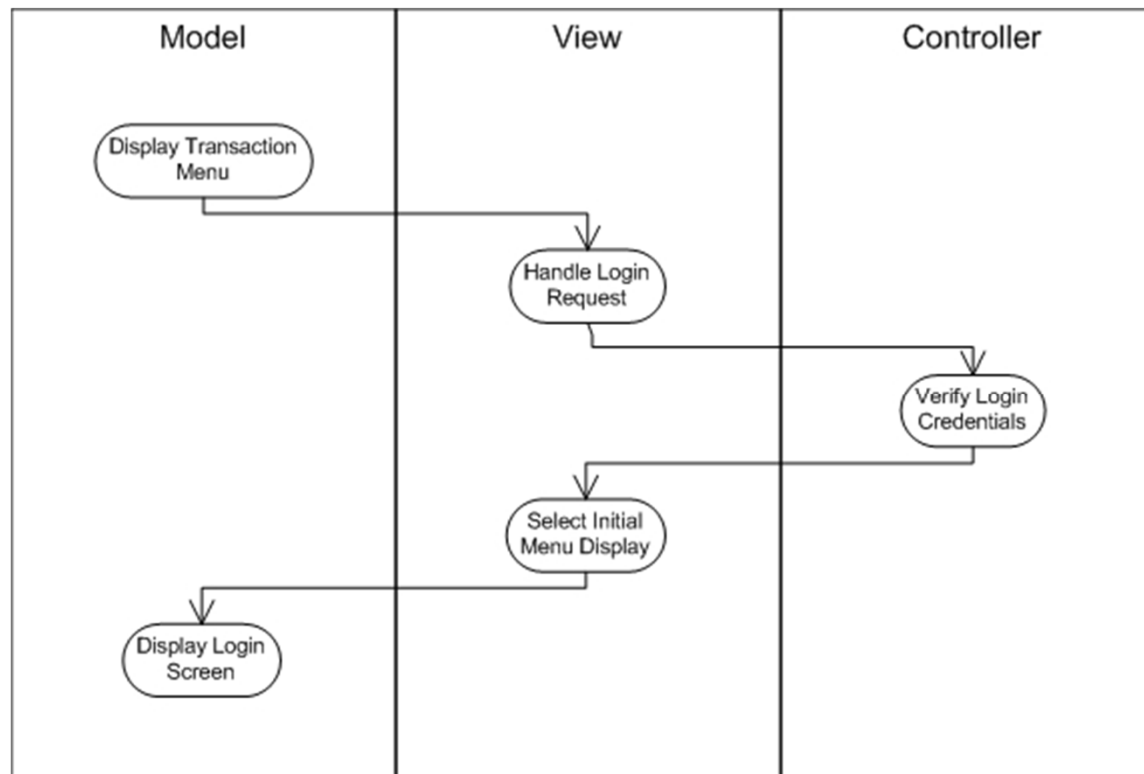


1. User requests "login" option; Controller sets "login.jsp" as the next view.
2. User enters credentials; Controller invokes Login Handler to handle transaction etc.





# MVC in Action





# Model View Controller (MVC)

- ✿ Relationships between each pair of components in a Model-View-Controller design
- ✿ The View can request data from the Model for presentation to the user.
- ✿ The Controller handles user actions by invoking the appropriate Model component to handle the requested action and by setting the subsequent View component for presentation to the user.
- ✿ The Model handles all transactions, including application of business rules and database operations. The Model can notify the View of changes in state.



# Flow of control and data

## ☀ View-Controller

- ☀ View responds to user interactions by notifying the controller of an event and passing the relevant data.
- ☀ Interactions can be triggered by a form submission, clicking on a URL or through AJAX.

## ☀ Controller-Model

- ☀ The controller issues transactions based on the requested application function to the model.
- ☀ In an e-commerce site this may include logging in, adding an item to a shopping cart or entering payment information



# Flow of control and data

## ☀ View-Model

- ☀ In a **data pull** relationship: the view requests data from the model, such that if the controller instructs the view to show a product page, the view will then request product information from the model
- ☀ In **data push** relationship: the model presents updates to the view as they occur, e.g. if it is a weather report, the model might push out weather updates every few minutes to which the view would respond by changing its user display accordingly.



# Consequences

## ❖ **Re-use of Model components.**

- ❖ The separation of model and view allows multiple views to use the same enterprise model.
- ❖ Consequently, an enterprise application's model components are easier to implement, test, and maintain, since all access to the model goes through these components.



# Consequences

## ❖ **Easier support for new types of clients.**

❖ To support a new type of client, you simply write a view and some controller logic and wire them into the existing enterprise application.

## ❖ **Increased design complexity.**

❖ This pattern introduces some extra classes due to the separation of model, view, and controller.



# Separation of Concerns

- ❖ MVC pattern was originally developed as part of the design of an object-oriented programming environment named Smalltalk.
- ❖ MVC is an application of the software engineering principle known as Separation of concerns.
- ❖ Separation of concerns implies that it is best to segregate different types of functionality within an application as much as possible



# Separation of Concerns

- ✿ There are several advantages to this principle as embodied in the MVC pattern for web applications
  - ✿ **Simplicity:** Each component can be designed and developed separately, reducing the overall complexity of the problem
  - ✿ **Independence:** Components can be interchanged, replaced, and replicated as needed. The HTML interface to an application can be replaced without affecting other components while additional views can be developed for other devices such as mobile phones





# Separation of Concerns

- ✿ **Scalability:** Components can be expanded with additional capabilities;
  - ✿ additional views and models can be added to an existing application.
- ✿ **Specialization:** Developers can specialize on one component, and can fine-tune their skills in that area

# MVC pattern applied to a Java web application

- ✿ View Component: Includes HTML, JSP and other documents as well as images, scripts and other supporting components
- ✿ Controller component include servlets that responds to HTTP requests. These include validating incoming requests and reformatting data for transaction processing.
- ✿ Model component includes the Java beans and helper classes that allow the application to manipulate and maintain its state. It also include database, DBMS and database driver.