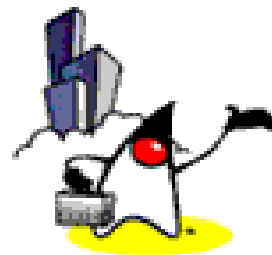




# MVC Pattern (MVC Framework)

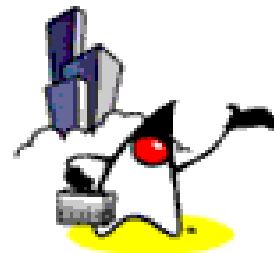


# Agenda

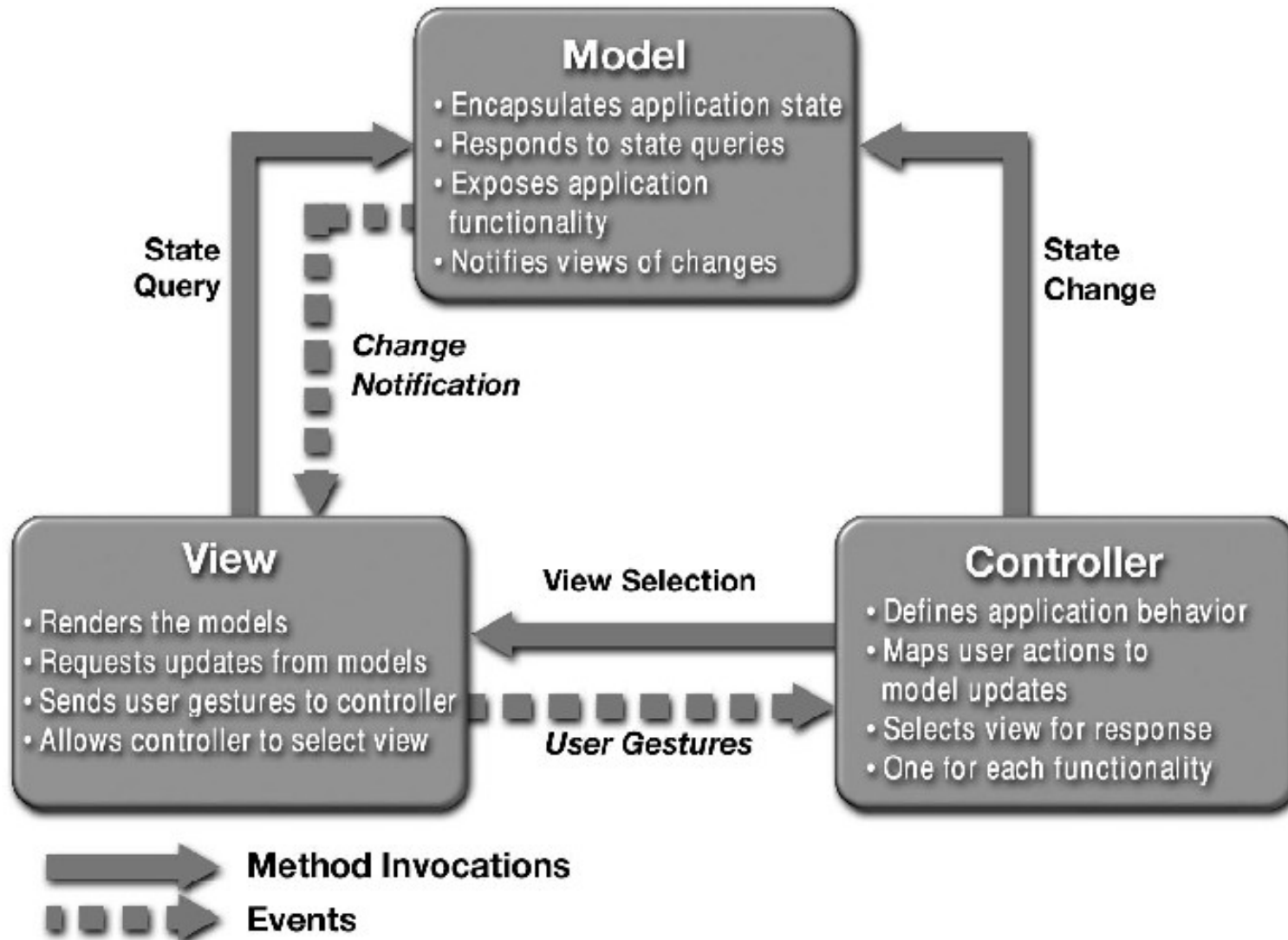
- Introduction of MVC pattern
- Evolution of Web Application design architecture
  - Model 1
  - Model 2
  - Application frameworks



# Introduction to MVC Pattern



# MVC Pattern



# Three Logical Layers in a Web Application: Model

- Model (Business process layer)
  - Models the **data and behavior** behind the business process
  - Responsible for actually doing
    - Performing DB queries
    - Calculating the business process
    - Processing orders
  - Encapsulate of data and behavior which are **independent of presentation**

# Three Logical Layers in a Web Application: View

- View (Presentation layer)
  - **Display** information according to client types
  - Display result of business logic (Model)
  - Not concerned with how the information was obtained, or from where (since that is the responsibility of Model)

# Three Logical Layers in a Web Application: Controller

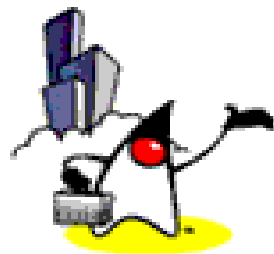
- Controller (Control layer)
  - Serves as the logical connection between the user's interaction and the business services on the back
  - Responsible for making decisions among multiple presentations
    - e.g. User's language, locale or access level dictates a different presentation.
  - A request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

# Web Applications

- It is often advantageous to treat each layer as an independent portion of your application
- Do not confuse logical separation of responsibilities with actual separation of components
- Some or of the layers can be combined into single components to reduce application complexity



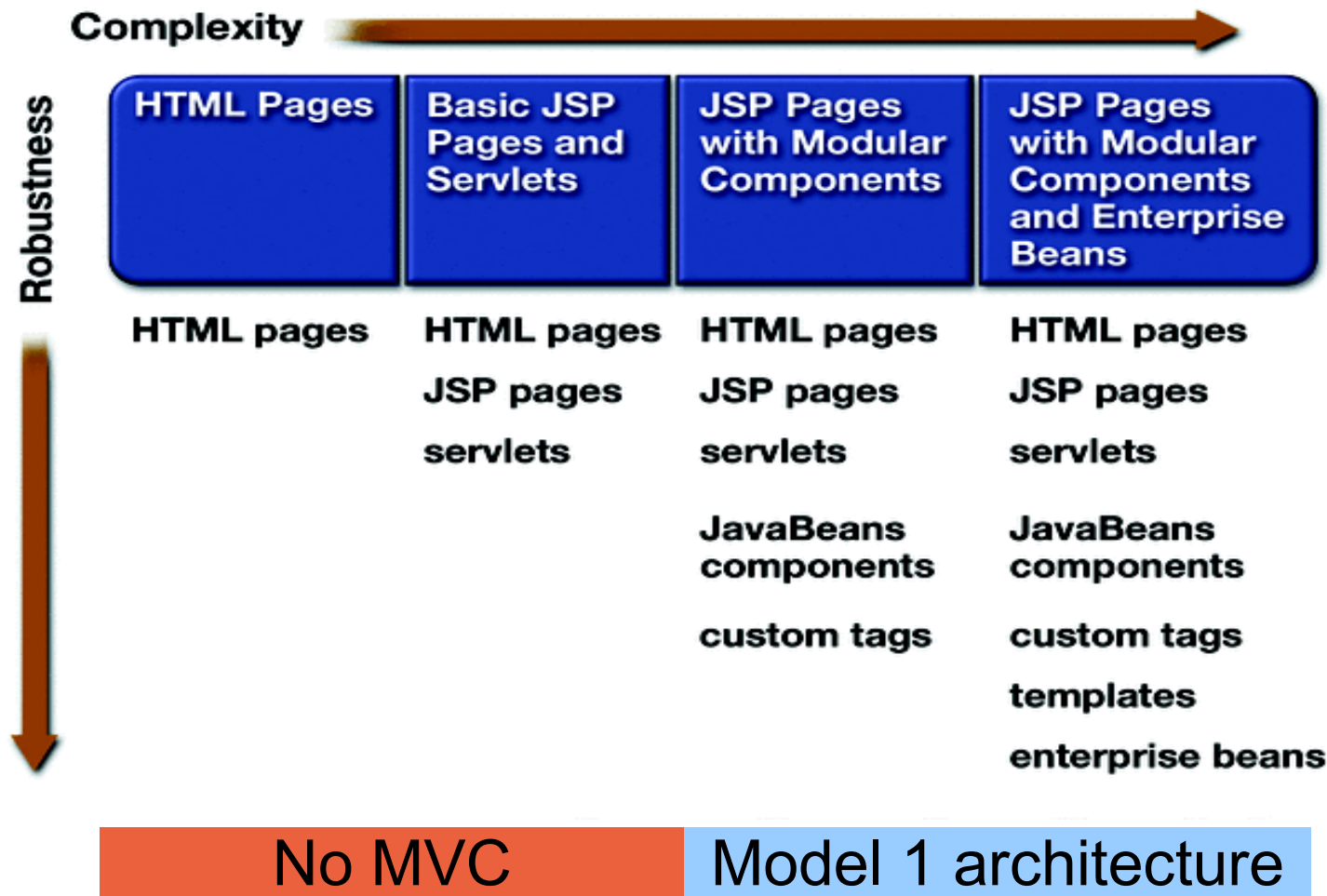
# Evolution of Web Application Design Architecture

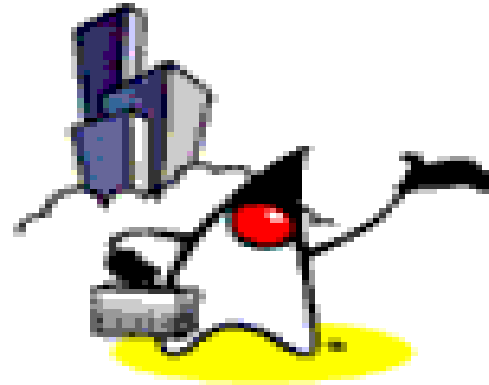


# Evolution of MVC Architecture

- 1.No MVC
- 2.MVC Model 1 (Page-centric)
- 3.MVC Model 2 (Servlet-centric)
- 4.Web application frameworks
  - Struts
- 5.Standard-based Web application framework
  - JavaServer Faces (JSR-127)

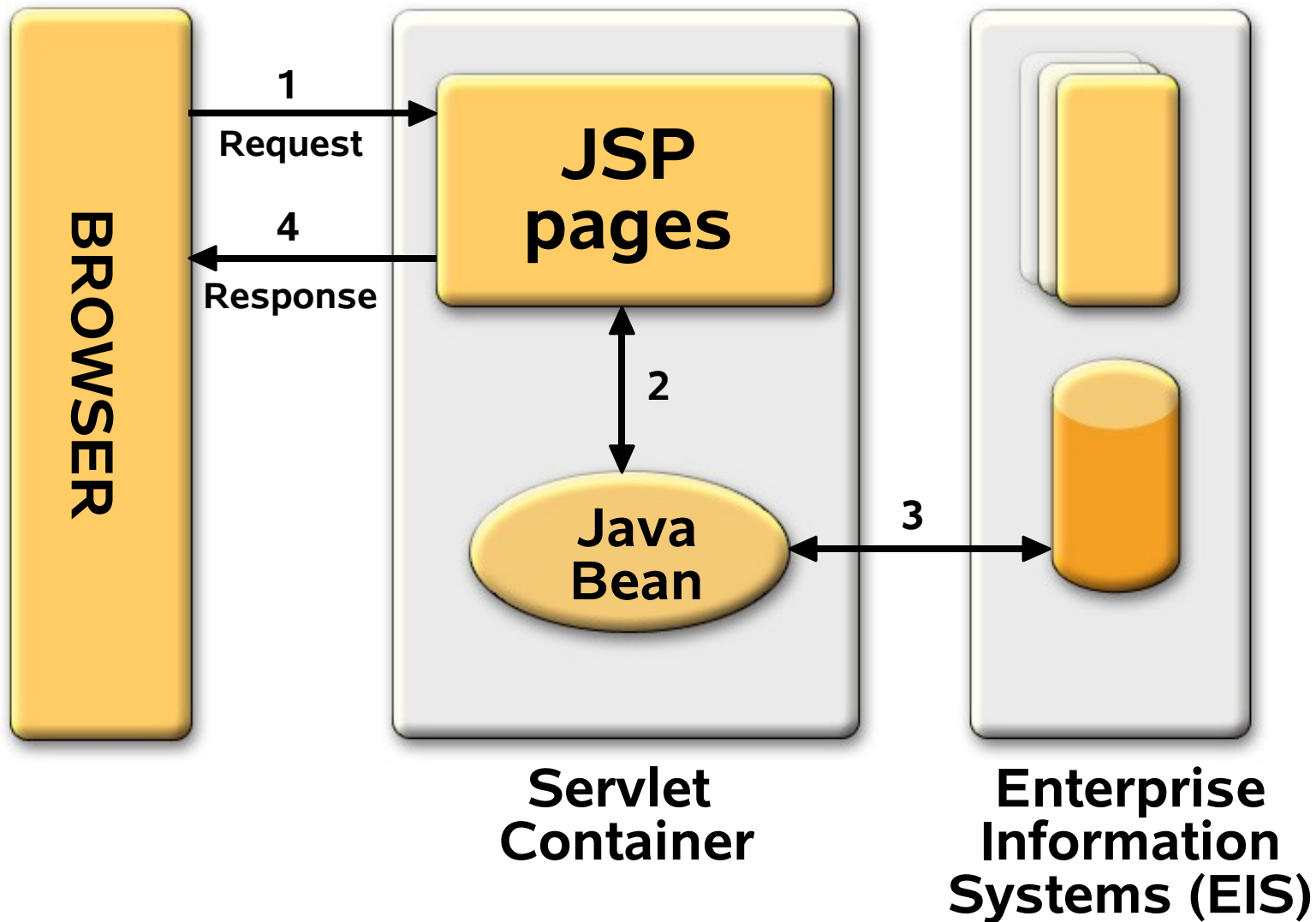
# Evolution of Web Application Design until Model 1 Architecture





# Model 1 (Page-Centric Architecture)

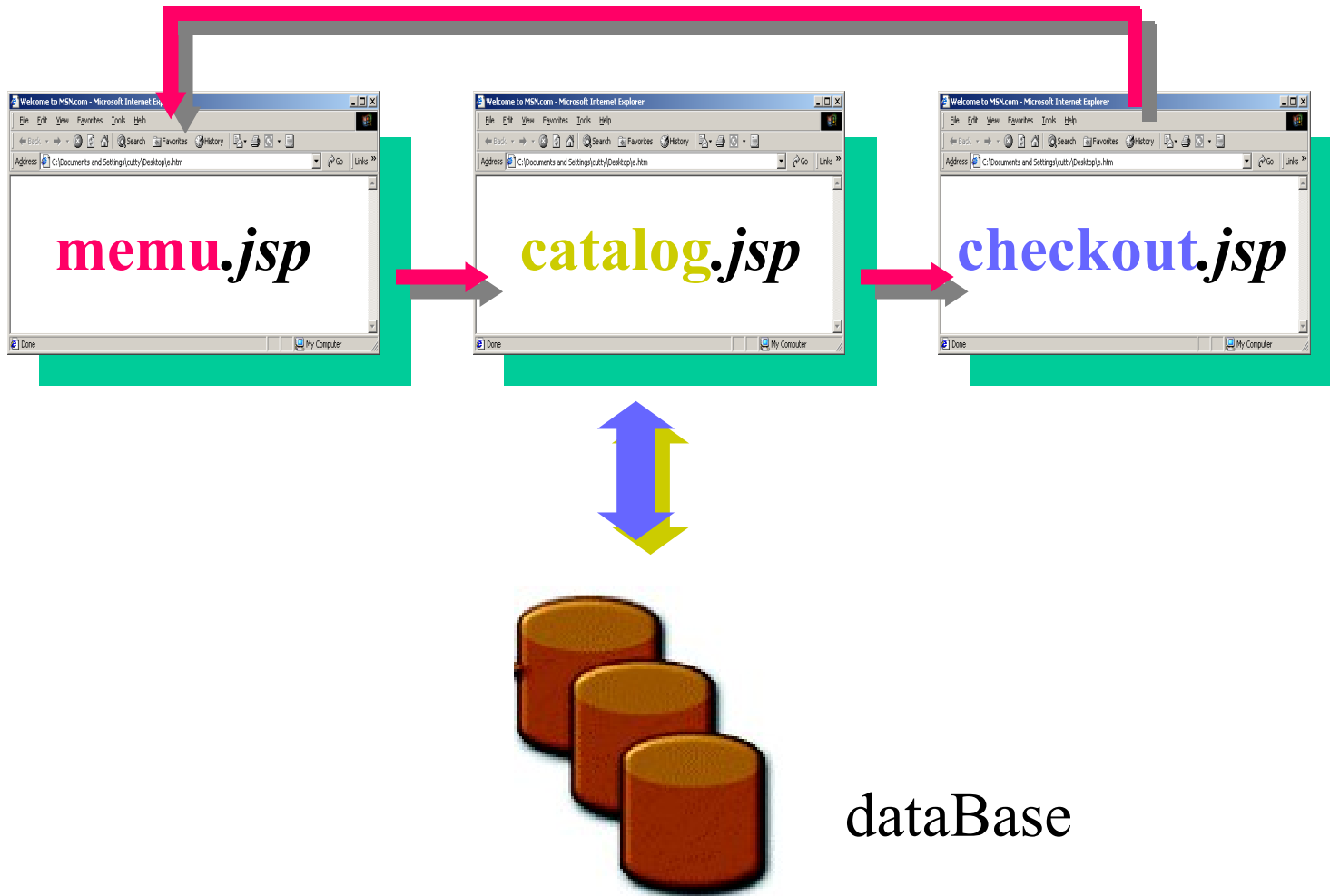
# Model 1 Architecture (Page-centric)



# Page-centric Architecture

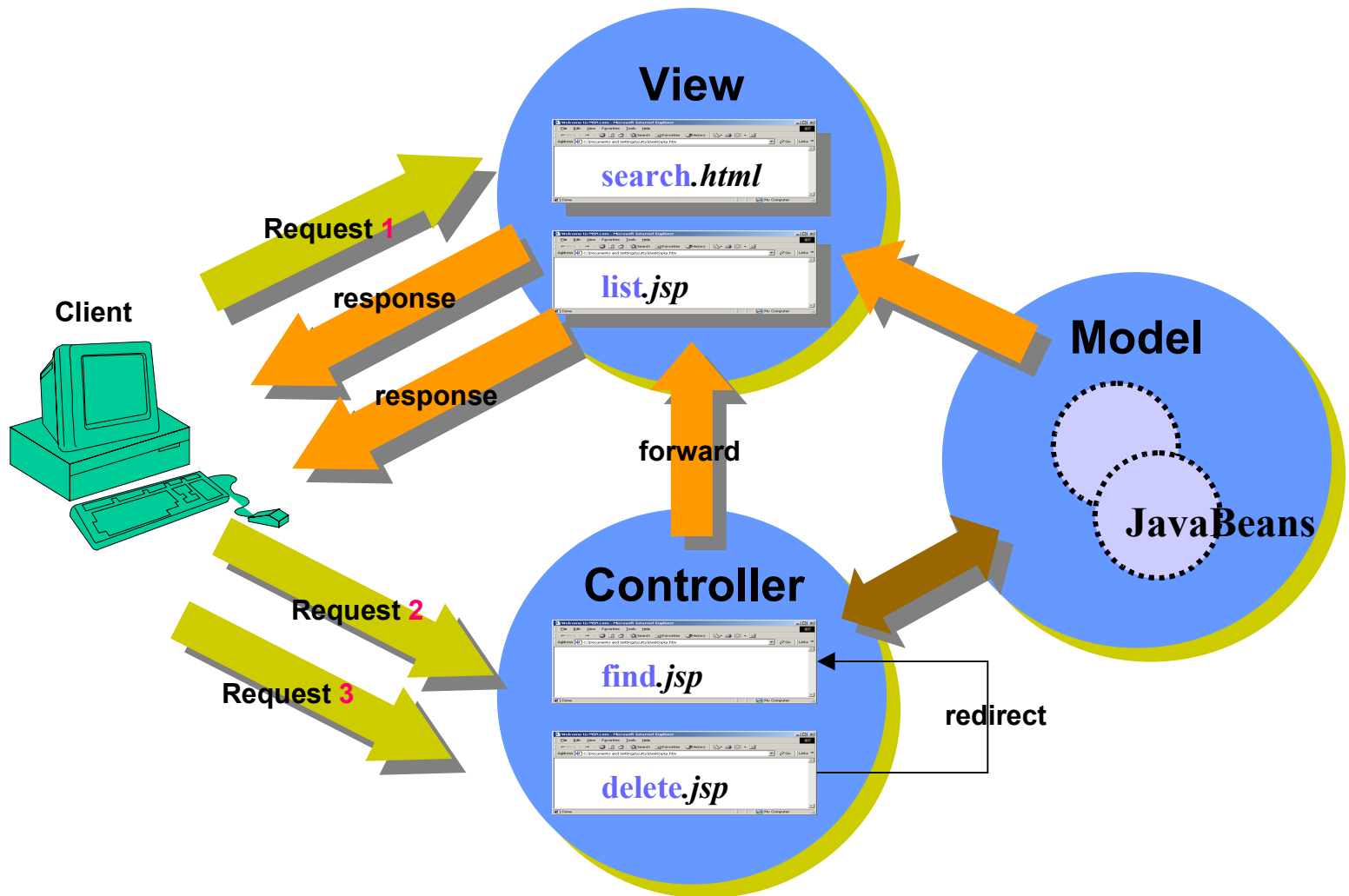
- Composed of a series of interrelated JSP pages
  - JSP pages handle all aspects of the application - presentation, control, and business process
- Business process logic and control decisions are hard coded **inside JSP pages**
  - in the form of JavaBeans, scriptlets, expression
- Next page selection 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">`

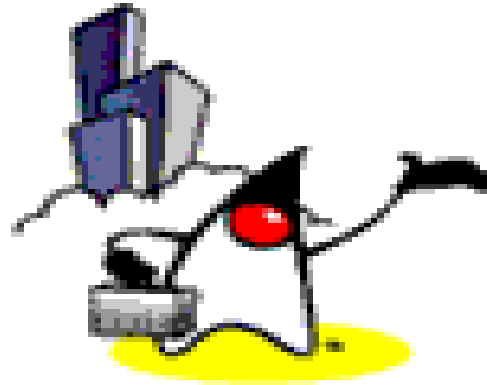
# Page-centric Architecture



**page-centric catalog application**

# Page-centric Scenario

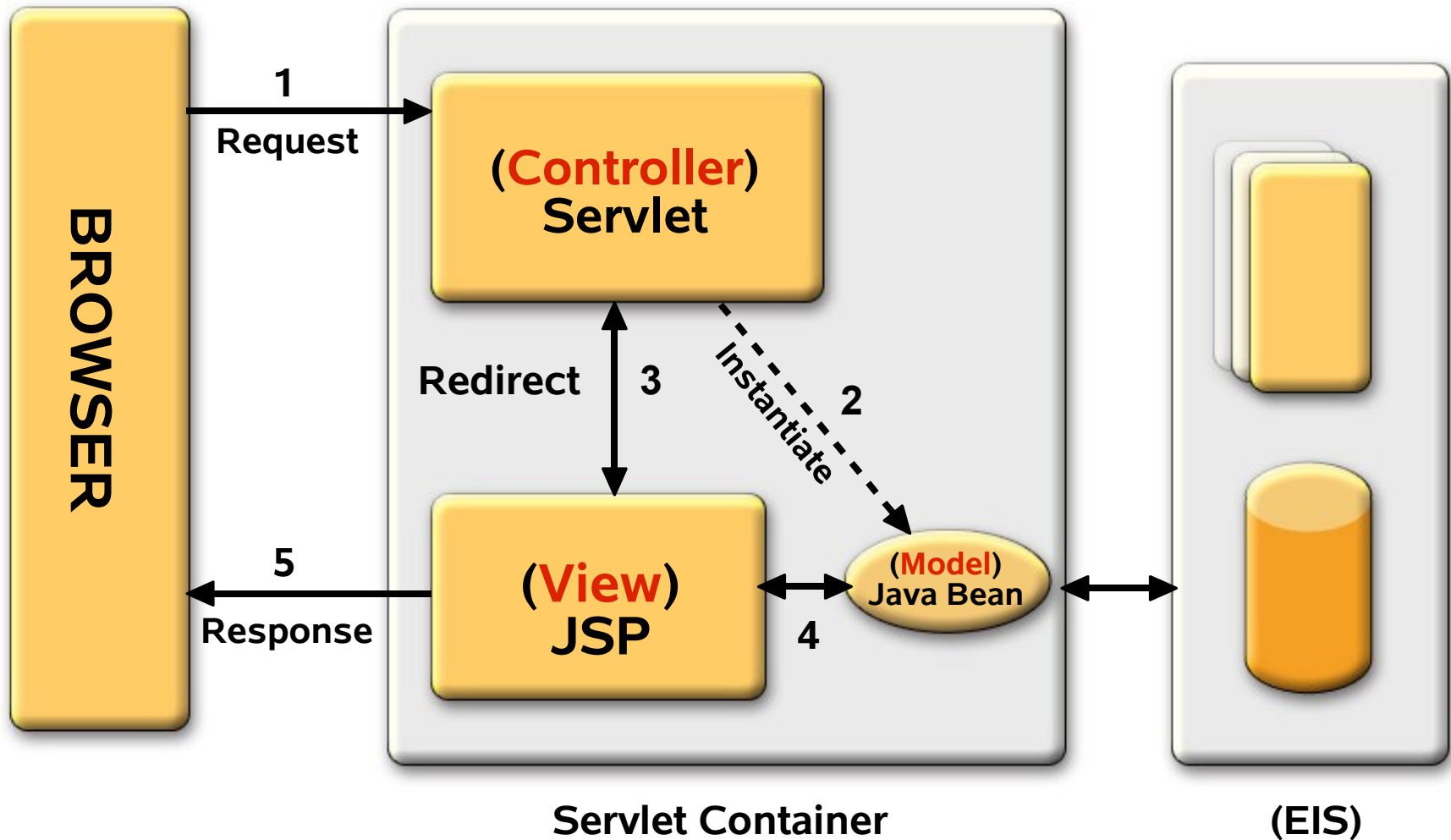




# Model 2 (Servlet-Centric Architecture)

# Model 2 Architecture (Servlet-centric)

MVC Design Pattern



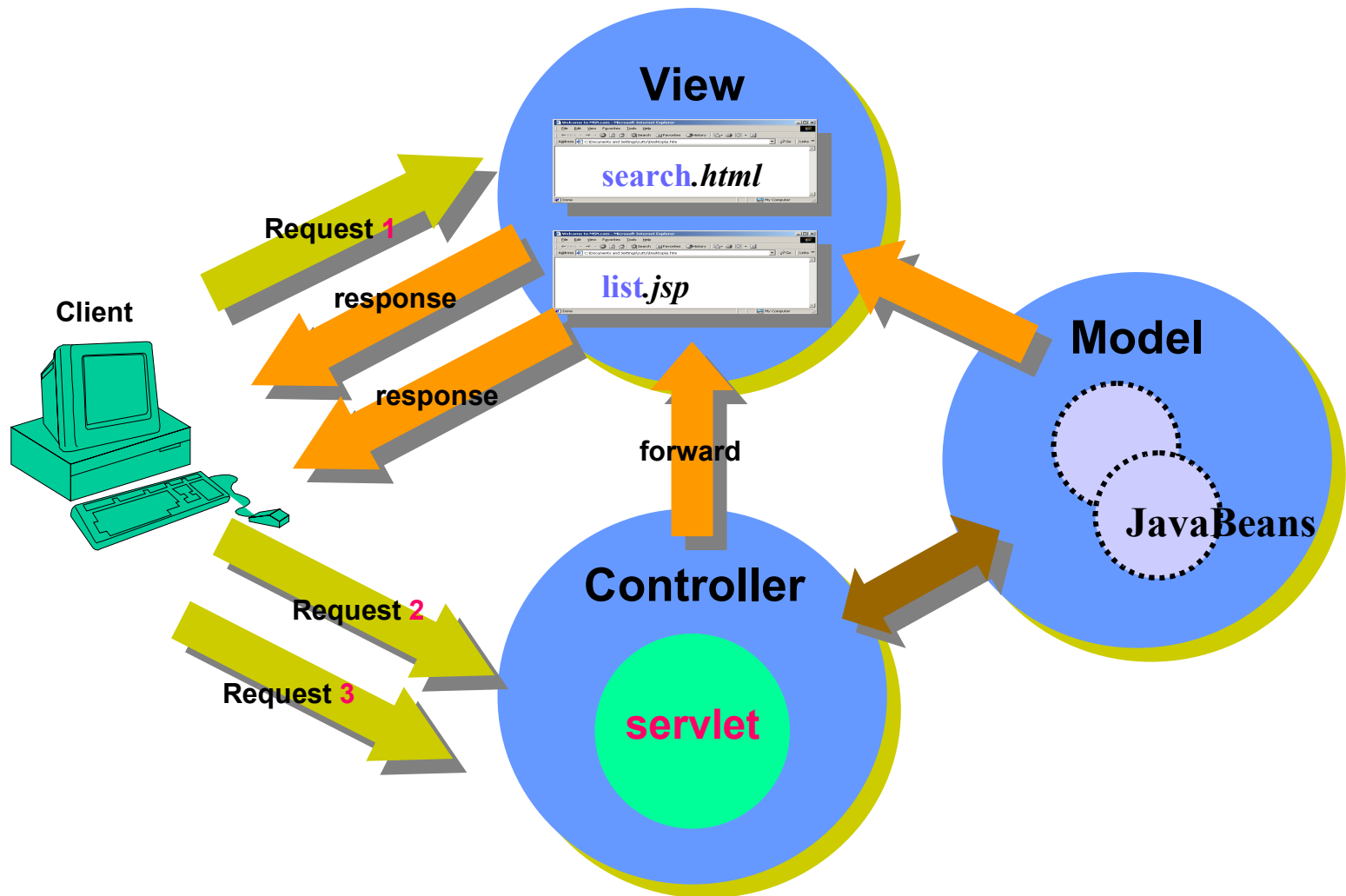
# Why Model 2 Architecture?

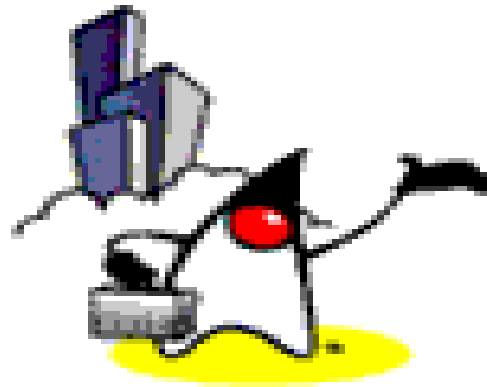
- What if you want to present different JSP pages depending on the data you receive?
  - JSP technology alone even with JavaBeans and custom tags (Model 1) cannot handle it well
- Solution
  - Use Servlet and JSP together (Model 2)
  - Servlet handles initial request, partially process the data, set up beans, then forward the results to one of a number of different JSP pages

# Servlet-centric Architecture

- JSP pages are used only for presentation
  - Control and application logic handled by a servlet (or set of servlets)
- Servlet serves as a **gatekeeper**
  - Provides common services, such as authentication, authorization, login, error handling, and etc
- Servlet serves as a **central controller**
  - Act as a state machine or an event dispatcher to decide upon the appropriate logic to handle the request
  - Performs redirecting

# Servlet-centric Scenario





# Web Application Frameworks

# Web Application Frameworks

- Based on MVC Model 2 architecture
- Web-tier applications share common set of functionality
  - Dispatching HTTP requests
  - Invoking model methods
  - Selecting and assembling views
- Provide classes and interfaces that can be used/extended by developers

# Why Web Application Framework?

- De-coupling of presentation tier and business logic into separate components
- Provides a central point of control
- Provides rich set of features
- Facilitates unit-testing and maintenance
- Availability of compatible tools
- Provides stability
- Enjoys community-supports
- Simplifies internationalization
- Simplifies input validation

# Why Web Application Framework?

- Frameworks have evolved with Java Server technology
- JSP/Servlets are still hard to use
- Frameworks define re-usable components to make this job easier.
- A good framework defines how components work to create a usable application.

# Web Application Frameworks

- Apache Struts
- JavaServer Faces (JSR-127)
  - A server side user interface component framework for Java™ technology-based web applications
- Echo
- Tapestry



# Passion!

