Design and Maintenance of Web Applications in J2EE

Jeff Offutt

http://www.cs.gmu.edu/~offutt/

SWE 432

Design and Implementation of Software for the Web

Design and Maintenance Issues

Topic Outline

1. JSP Formatting Issues
2. General Design Issues
3. Tiered Architectures
4. Page Centric and Dispatcher Designs
5. Model View Controller (MVC)

JSP Maintenance Problems

- Presentation and content are not always well separated
  - Java mixed with HTML can be very hard to understand
- Most developers are not yet good at establishing levels of abstraction in JSP pages
- Books, articles, and web resources focus on JSP syntax, not style and design

First Rule of Formatting JSP

- JSP is somewhat messy (like JavaScripts)
  - Hard to read
  - Hard to debug
  - Hard to get right
  - Hard to maintain
- Strategy:

  Keep a minimum of Java in the JSP, do most of the programming with separate Java:
  - Servlets
  - Beans

  This allows separation of concerns – good OO design

JSP: Readable HTML

- Make JSP look like HTML with Java calls,
  not Java with some HTML
- Move all of the business logic out of the JSP
- Java that generates HTML is hard to maintain:
  - Humans have trouble viewing HTML as "normal text"
  - The quotes ("\"") are very hard to read
- Let HTML developers write HTML, and Java developers write Java

The system design must support these goals

Design and Maintenance Issues

Topic Outline

1. JSP Formatting Issues
2. General Design Issues
3. Tiered Architectures
4. Page Centric and Dispatcher Designs
5. Model View Controller (MVC)

Design Goals

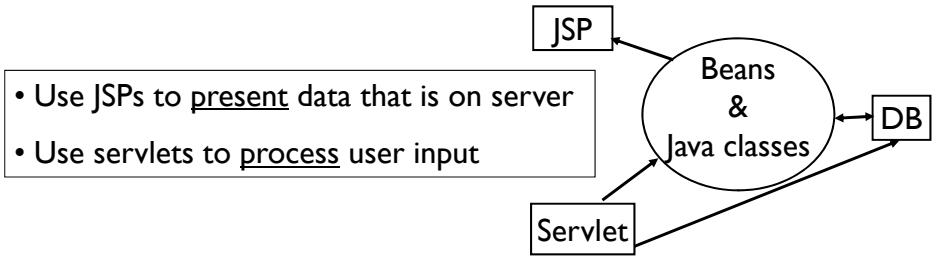
- A major design goal of the web application design is separation of concerns :
  - Presentation
  - Logic
  - Data
- Also to support our seven criteria from week one :
  1. Reliability
  2. Usability
  3. Security
  4. Availability
  5. Scalability
  6. Maintainability
  7. Performance & Time to market

Separation of Concerns

- Presentation → HTML & JSP
- Data content → Servlets & beans
- Data representation → Data structures, Beans & Java classes
- Data storage → Data base & files, Oracle, SQL

Separation of Concerns (2)

- *doGet*() and *doPost*() should be simple and short
  - Shift processing to other methods and classes
- Put complicated logic in non-servlet classes
- Put almost no logic in JSPs
  - JSPs should present data they get from other classes



Design Specification

- Software Requirements Baseline
- Information Architecture Specification
  - Site map, Web Page Flows, Compositions, Labeling, Data Element Mappings
- Web Application Design
  - High-Level Software Design
  - Software Architecture and System Architecture Diagram
  - Class Diagrams
  - Sequence Diagrams
  - Class Specifications

Design and Maintenance Issues

Topic Outline

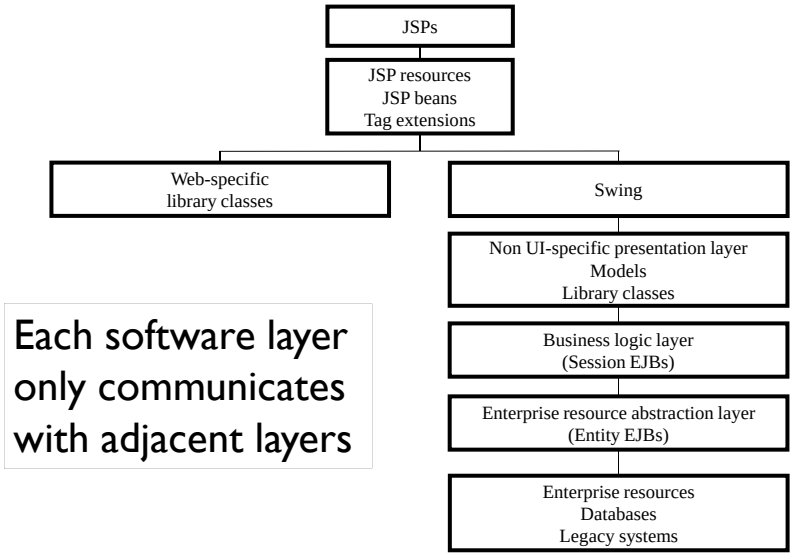
1. JSP Formatting Issues
2. General Design Issues
3. Tiered Architectures
4. Page Centric and Dispatcher Designs
5. Model View Controller (MVC)

J2EE Assumptions about Data

- Data values: The contents of memory
- Data structure: Types, organization and relationships of different data elements
- Data presentation: How the data is shown to humans

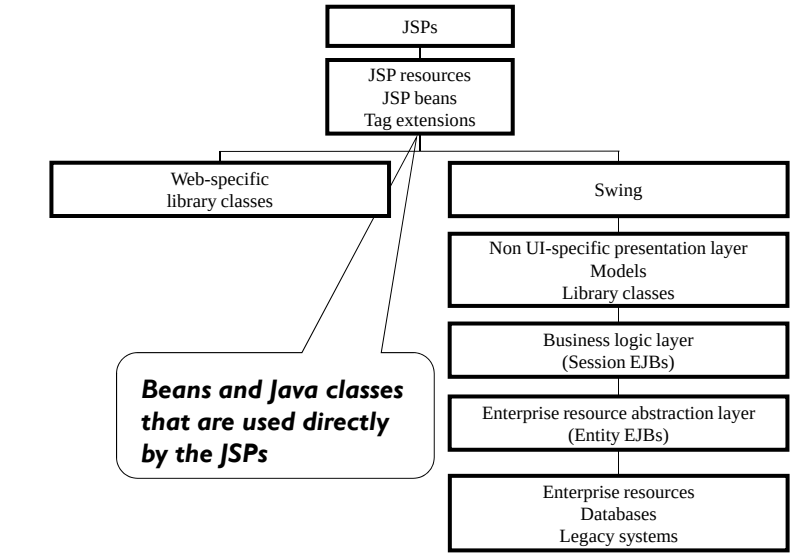
- J2EE assumes that data:
  - values change very frequently (during execution)
  - structure changes very infrequently
  - presentation changes occasionally

Multi-Tier Architectures

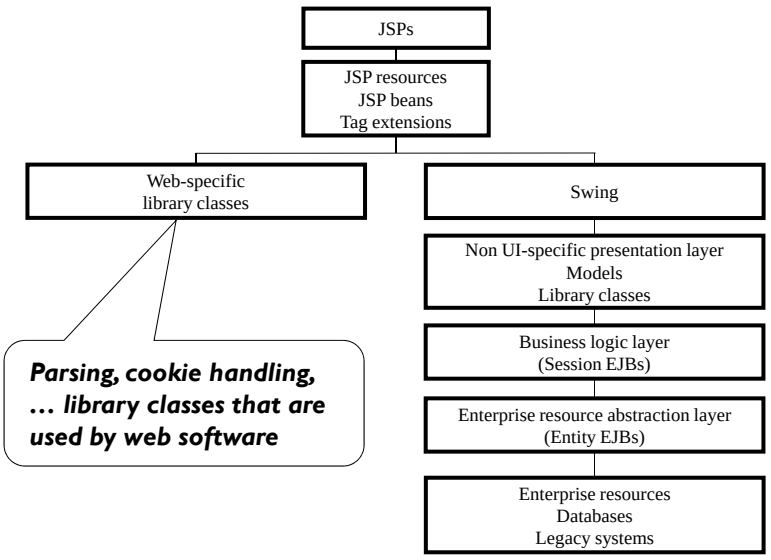


Each software layer only communicates with adjacent layers

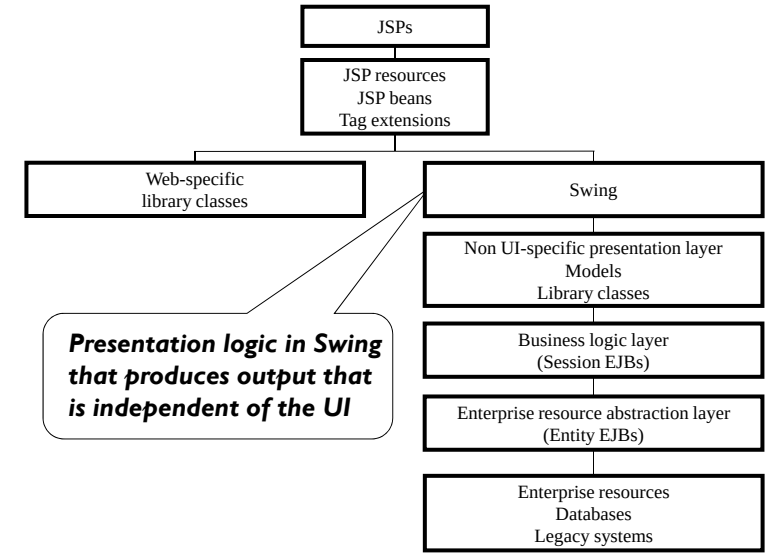
Multi-Tier Architectures (2)



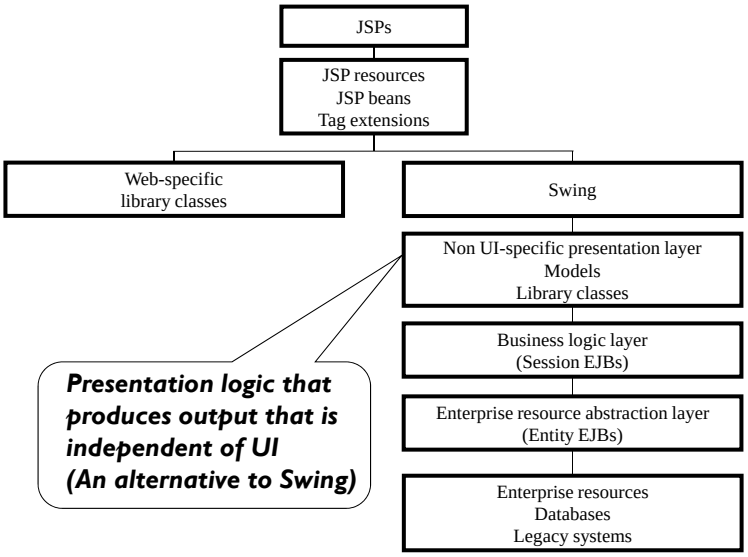
Multi-Tier Architectures (3)



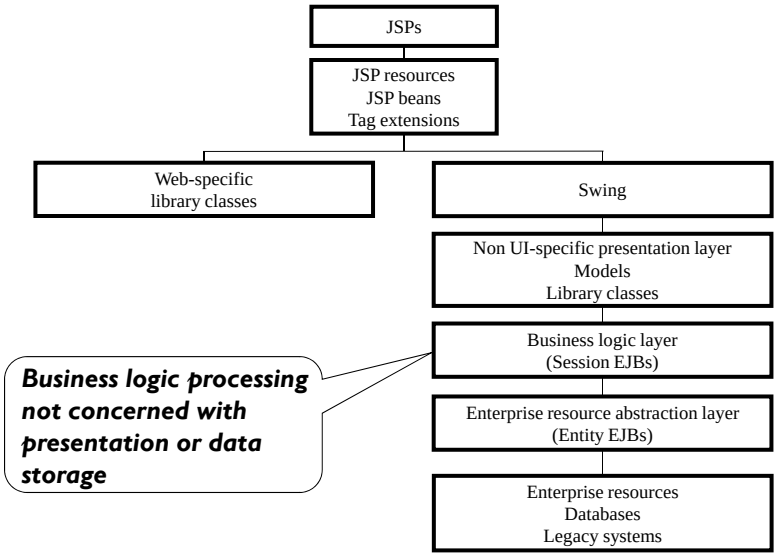
Multi-Tier Architectures (4)



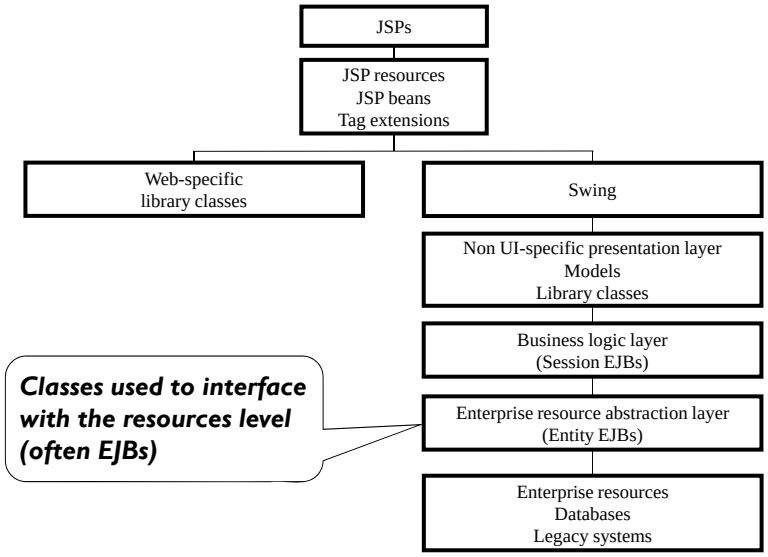
Multi-Tier Architectures (5)



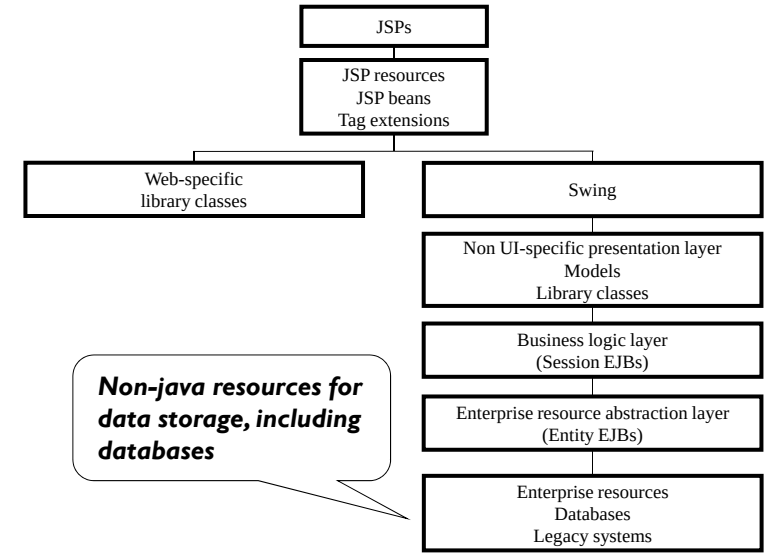
Multi-Tier Architectures (6)



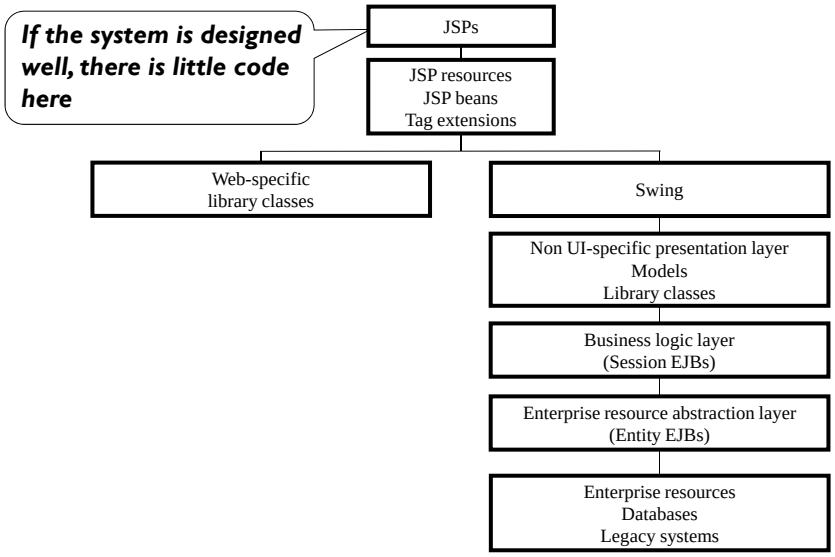
Multi-Tier Architectures (7)



Multi-Tier Architectures (8)



Multi-Tier Architectures (9)



Multi-Tier Architectures (10)

- This model allows very clean separation of the software that handles the data values, structure, presentation, and storage
- In small applications, some levels can be skipped
- Indeed, the need for this separation is hard to see with small applications – maintenance is only hard when systems get big

Design and Maintenance Issues

Topic Outline

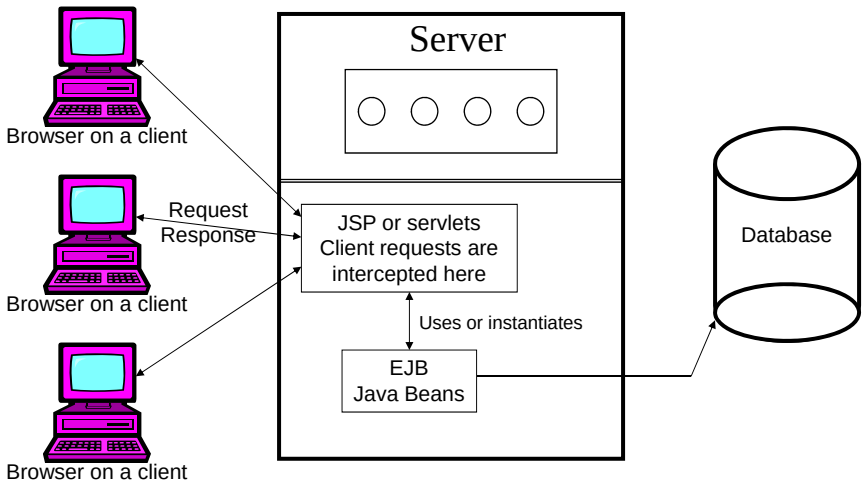
1. JSP Formatting Issues
2. General Design Issues
3. Tiered Architectures
4. Page Centric and Dispatcher Designs
5. Model View Controller (MVC)

Design Styles

1. Page-centric (client-server) : Requests are made to JSP pages, and the JSP pages respond to clients
2. Dispatcher (N-tier) : Requests are sent to JSPs or servlets that then forward the requests to another JSP or servlet

The goal is to separate logic from presentation and to separate as many concerns in the logic as possible

1) Page-centric Design



1. Page-centric Design (2)

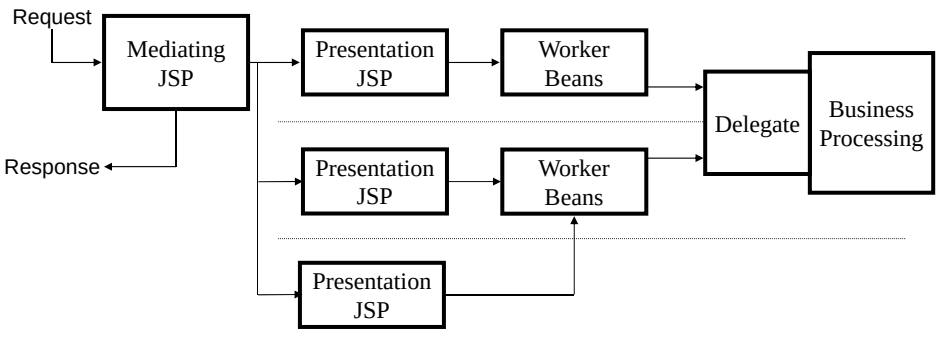
- This is a simple design to implement
- The JSP author can generate pages easily
- Two variants:
  - Page-View
  - Page-View with a Bean
- Does not scale up very well to large web sites
- Often results in a lot of Java code in the JSP
  - JSP authors must be Java programmers
  - Design is hard to see
  - Hard to maintain

2. Dispatcher Design

- A "dispatcher" accepts requests and routes them to the correct place
- In a dispatcher design, a front-end JSP (or servlet) looks at some portion of the request, and then chooses the correct place to forward it
- This is more sophisticated than the page-centric:
  - More flexible and scalable
  - More overhead that is wasteful with small applications
- Three versions
  - Mediator-View
  - Mediator-Composite View
  - Service to Workers

2-A. Mediator-View Design

- The Mediating JSP sends requests to a JSP
- The JSP sets and gets beans and creates a page



Design and Maintenance Issues

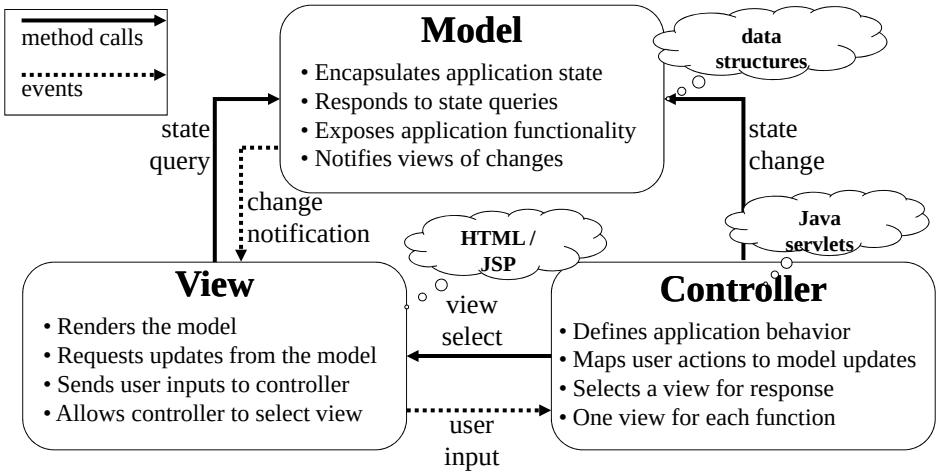
Topic Outline

1. JSP Formatting Issues
2. General Design Issues
3. Tiered Architectures
4. Page Centric and Dispatcher Designs
5. Model View Controller (MVC)

Model-View-Controller (MVC)

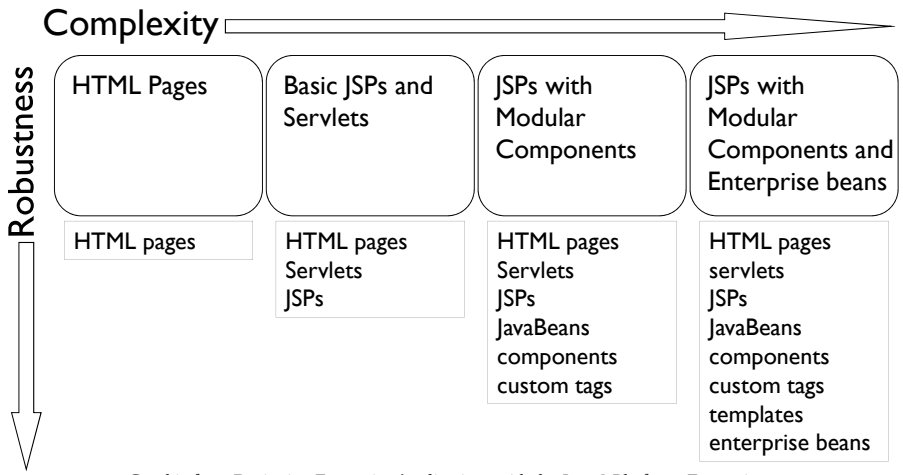
- The MVC architecture is an abstraction frequently used in web application design
- Provides a way to divide the responsibilities of objects
- Decreases coupling between objects and layers (supports easier maintenance)
- Helps divide the work – supports development expertise areas

Model-View-Controller (MVC)



* Graphic from Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition, Nicholas Kassem et al., October 2000

Web Application Design Complexity



Graphic from Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition, Nicholas Kassem et al., October 2000

J2EE Design Summary

- Common mistakes
  - No design and no comments
  - Not enough collaboration—entire team needs to understand the design
  - Using only parts of a design framework
- Requirements and design must be created first and be available to all team members
  - Every team member should be able to explain the design
- Must use meaningful names for packages, classes, methods, and variables
  - As teammates if they understand the names
- Follow correct OO principles (as in SWE 332)