Design and Maintenance of Java Server Pages

James Baldo Jr.

SWE 432
Design and Implementation of Software for the Web

JSP Maintenance Problems

- Presentation and content are not always well separated
  - Java mixed with the 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

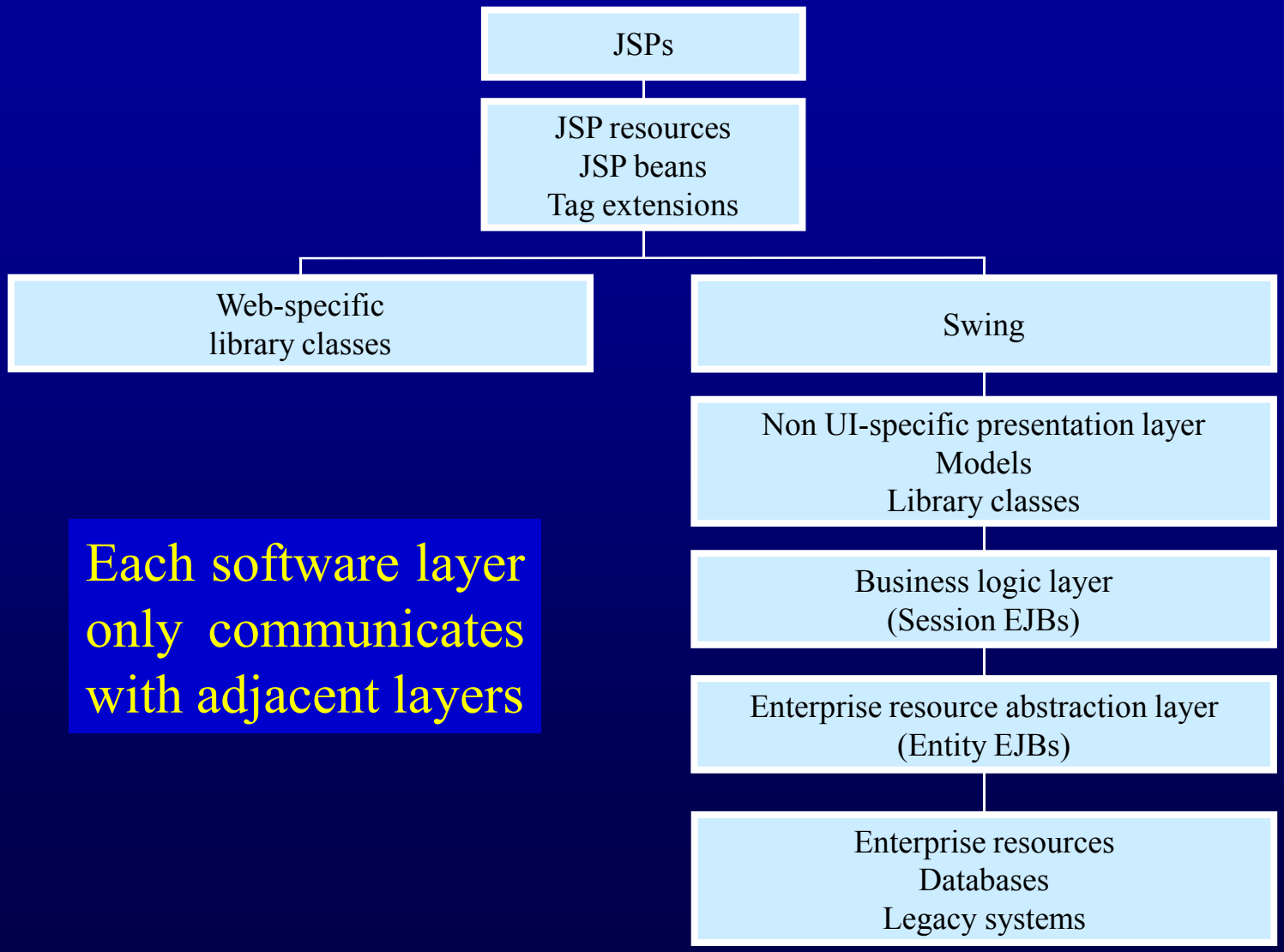
The system design must support these goals

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

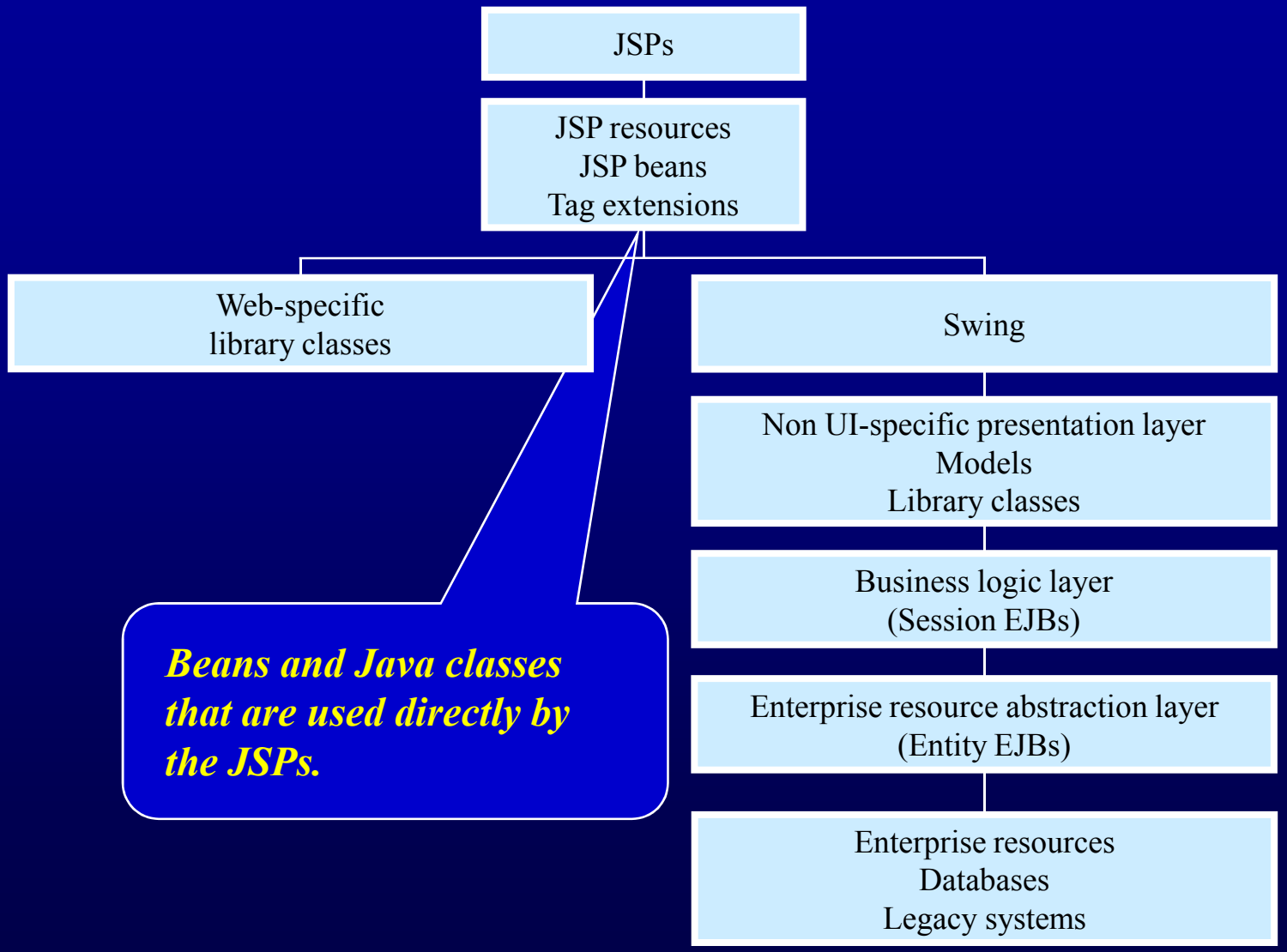
- J2EE assumes that data:
  - values change very frequently (during execution)
  - structure changes very infrequently
  - presentation changes occasionally

JSPs in a Multi-Tier Architecture

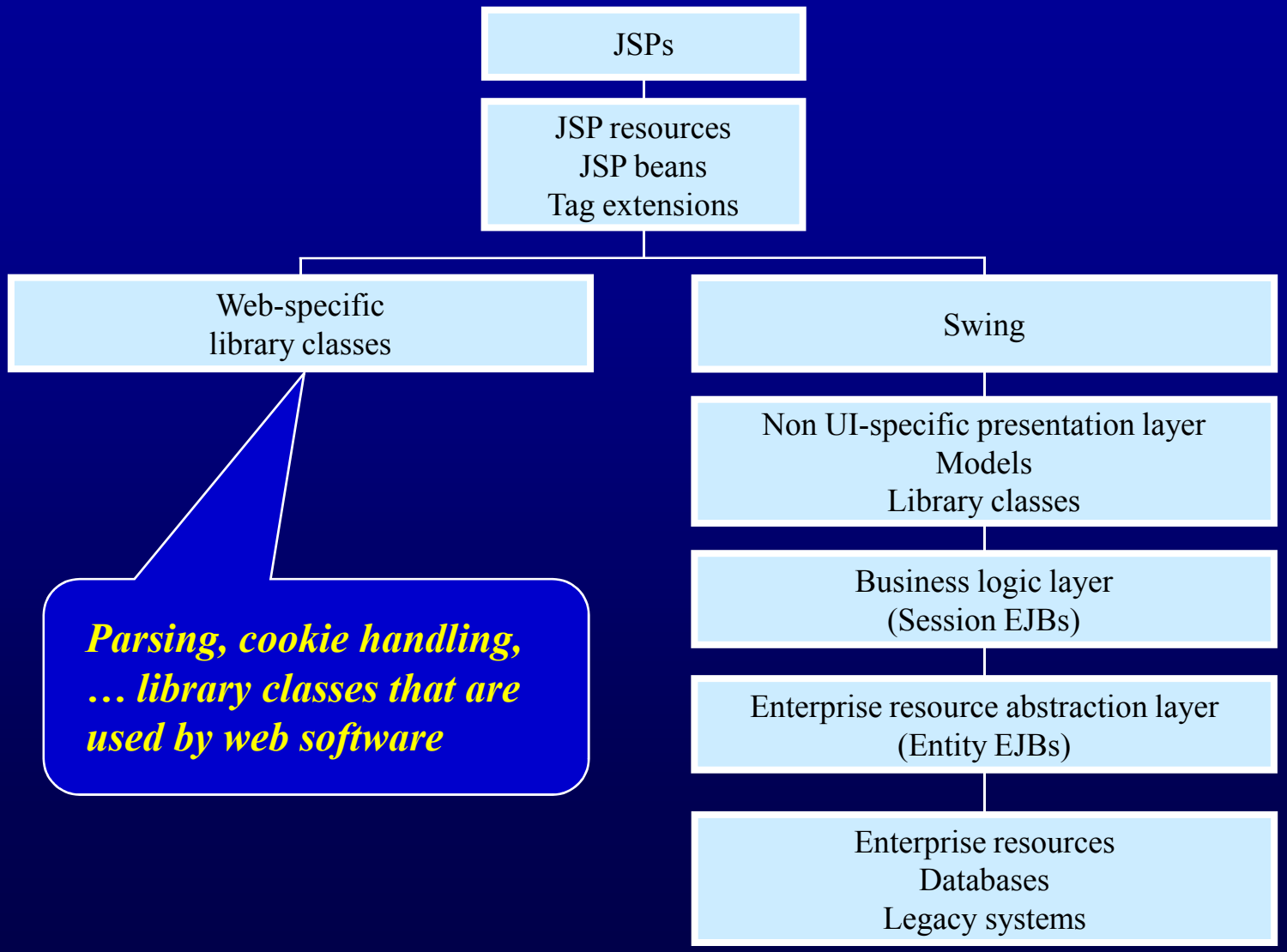


Each software layer only communicates with adjacent layers

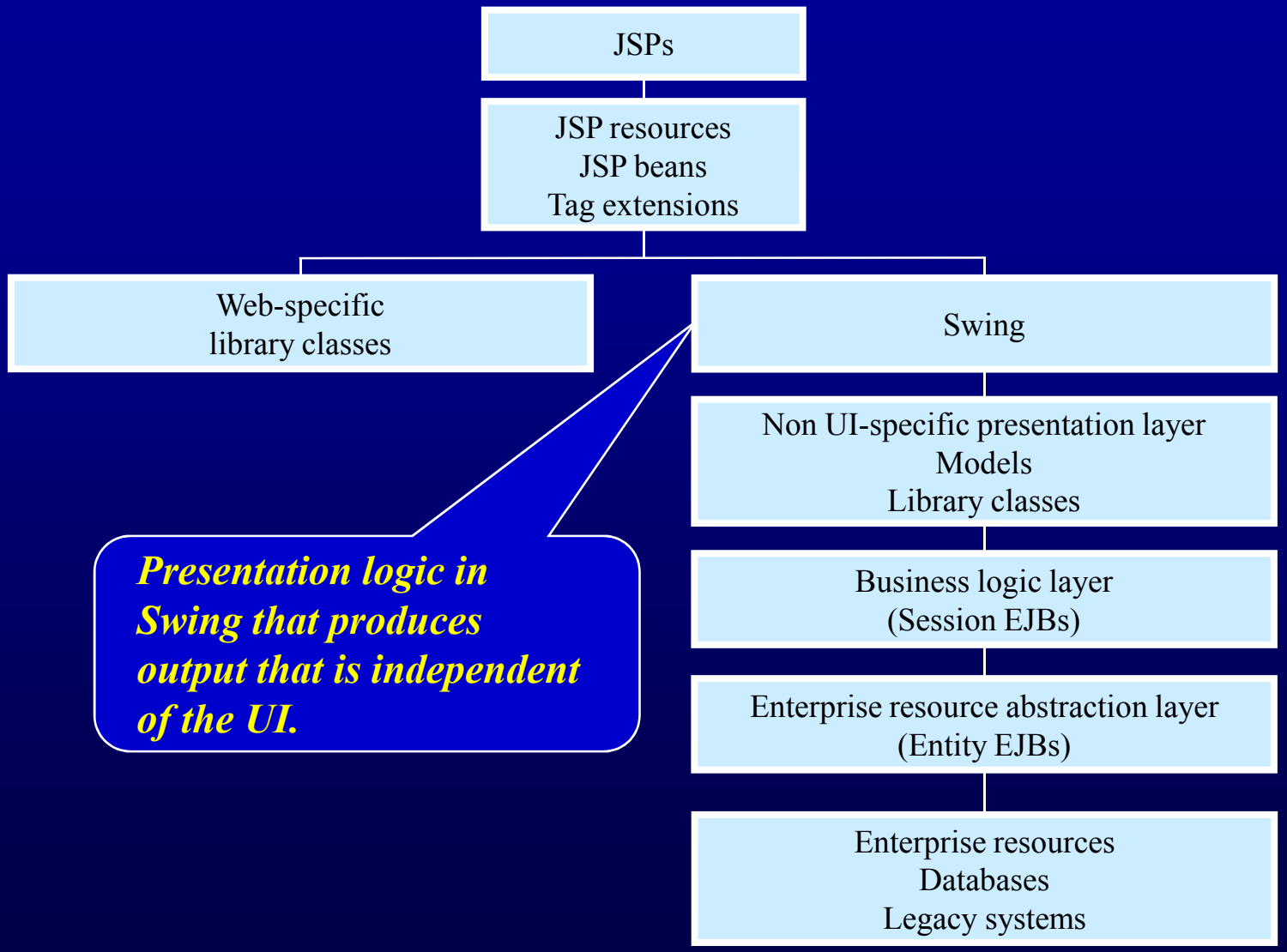
JSPs in a Multi-Tier Architecture (2)



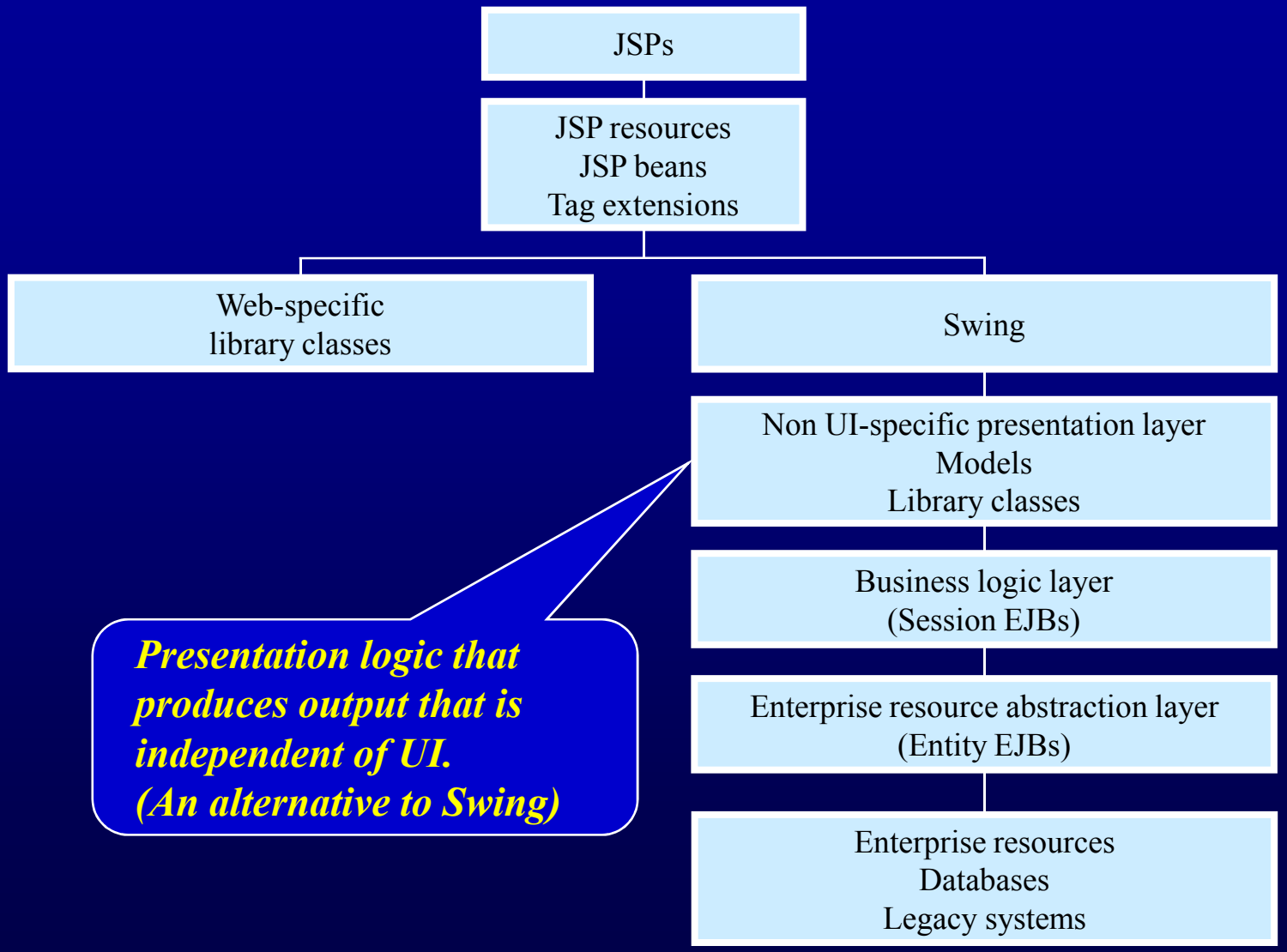
JSPs in a Multi-Tier Architecture (3)



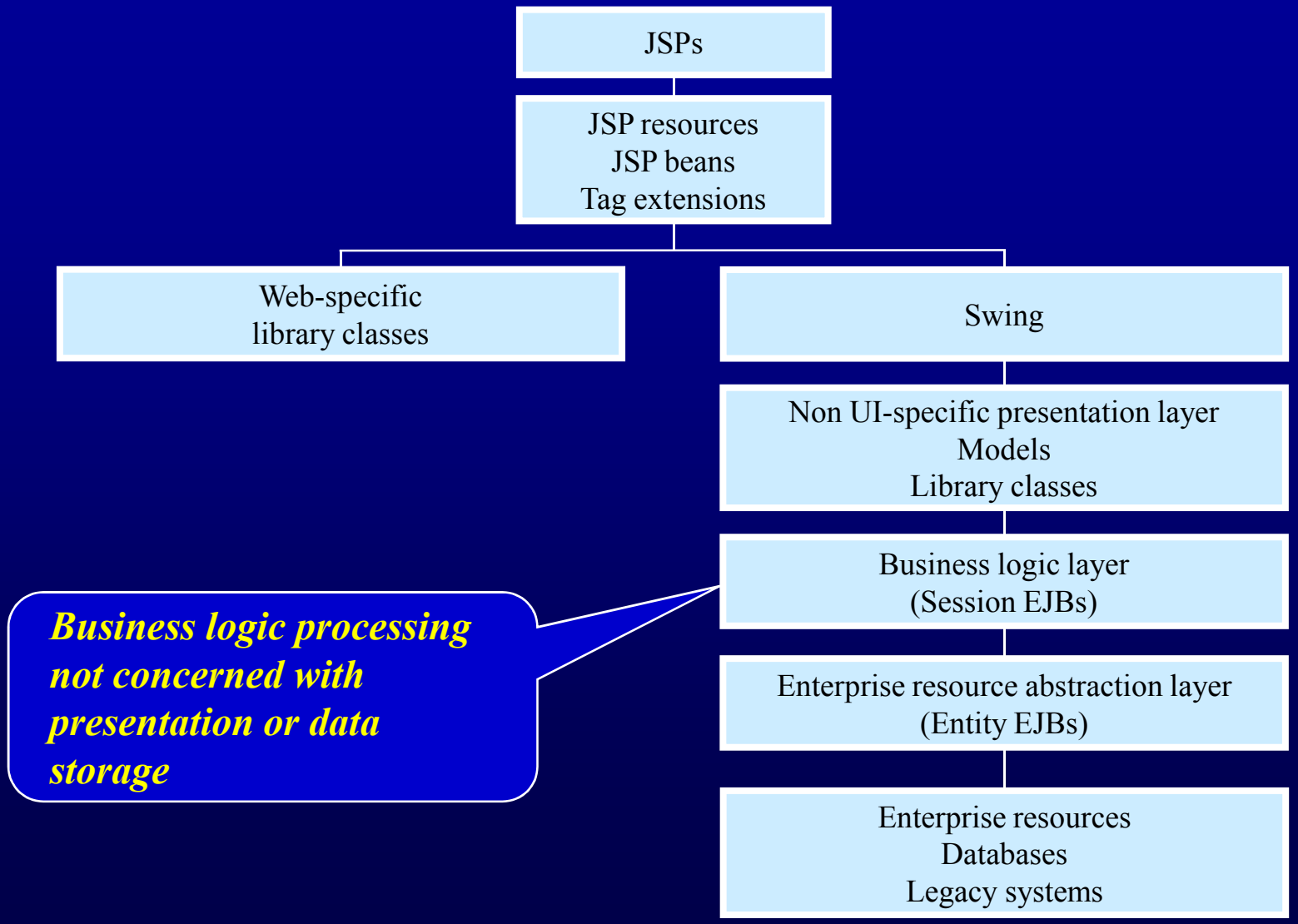
JSPs in a Multi-Tier Architecture (4)



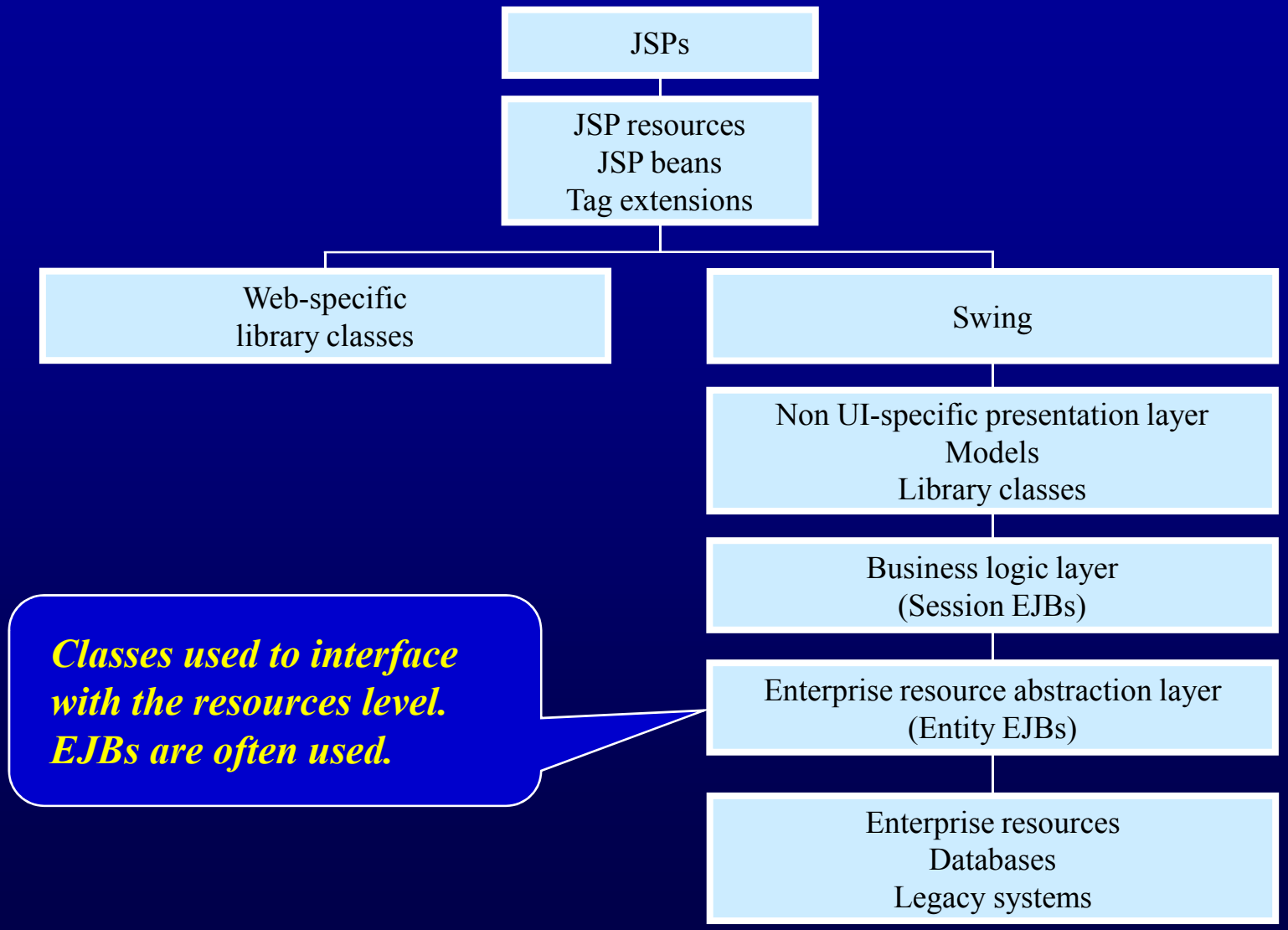
JSPs in a Multi-Tier Architecture (5)



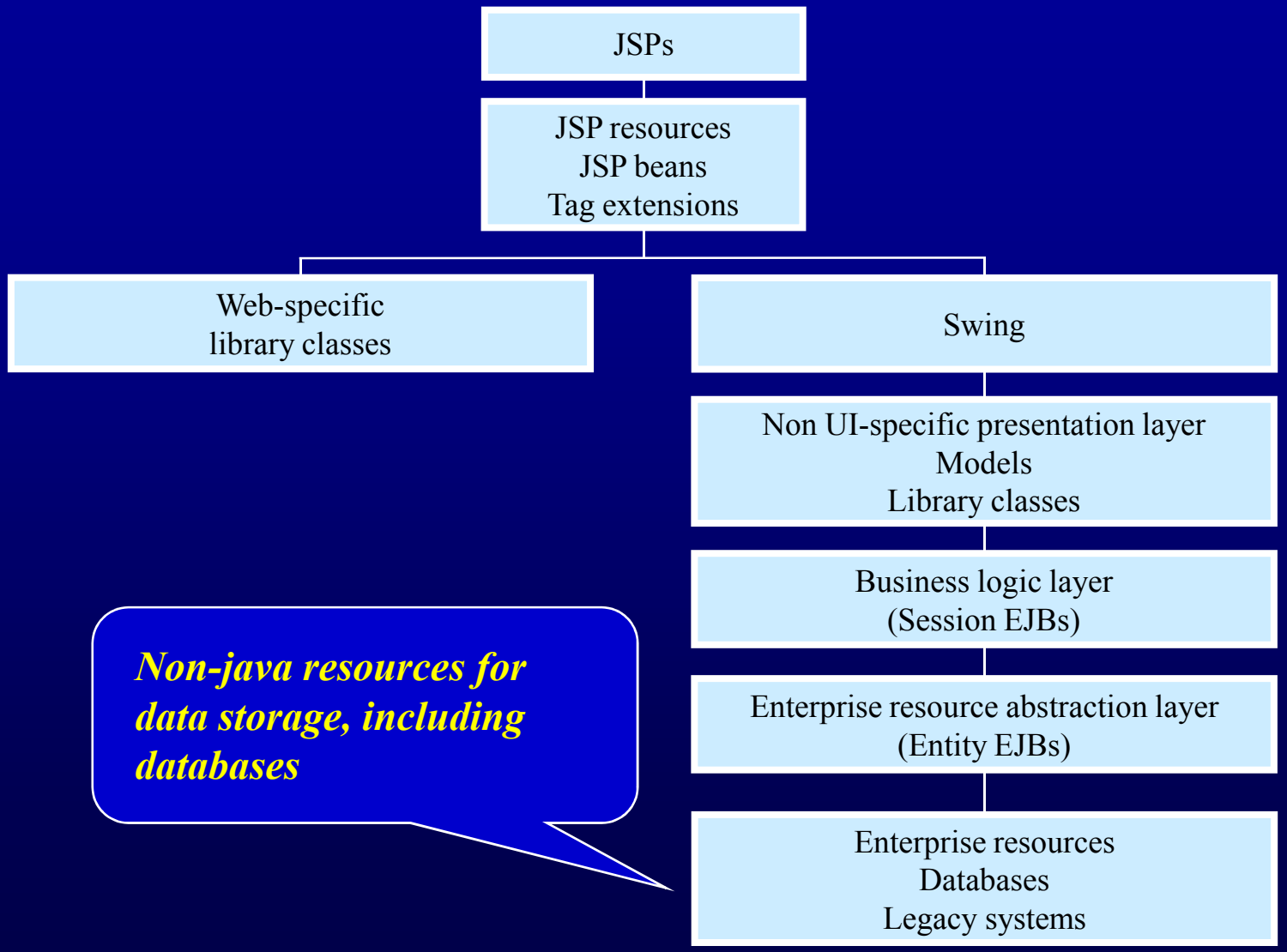
JSPs in a Multi-Tier Architecture (6)



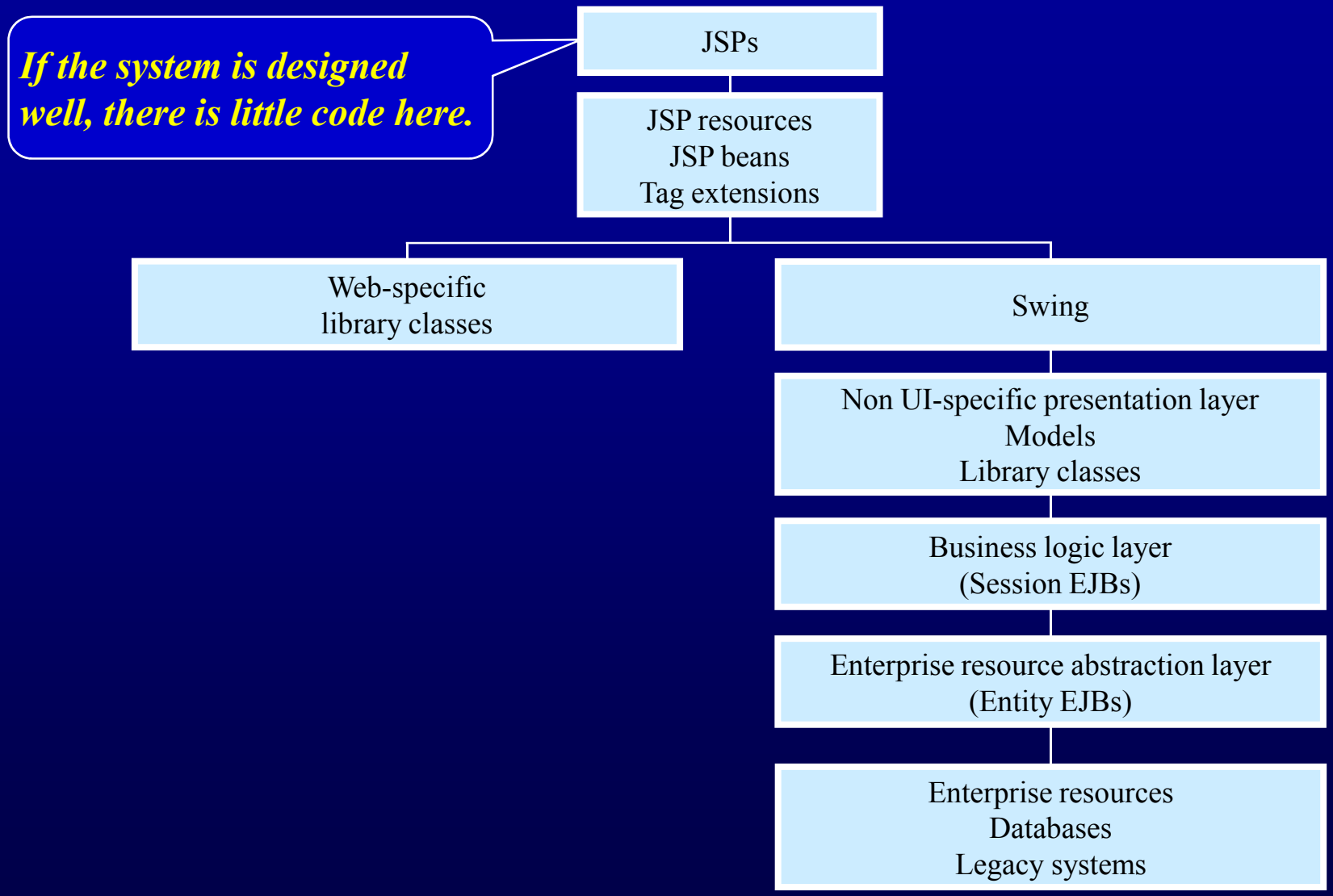
JSPs in a Multi-Tier Architecture (7)



JSPs in a Multi-Tier Architecture (8)



JSPs in a Multi-Tier Architecture (9)



JSPs in a Multi-Tier Architecture (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

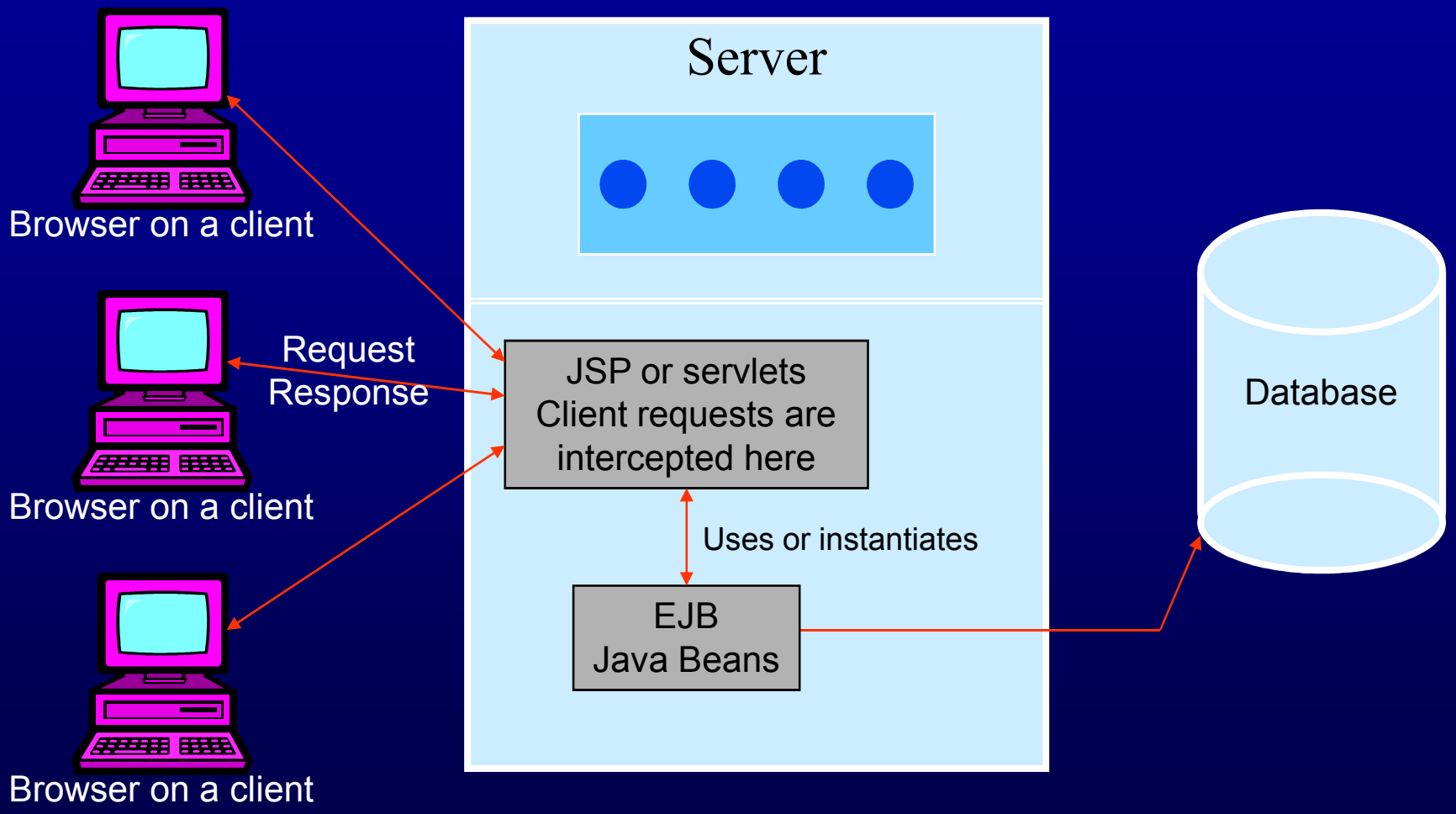
Design Styles

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

In both cases, 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