JavaServer Pages Technology

SL-315

Preface

About This Course

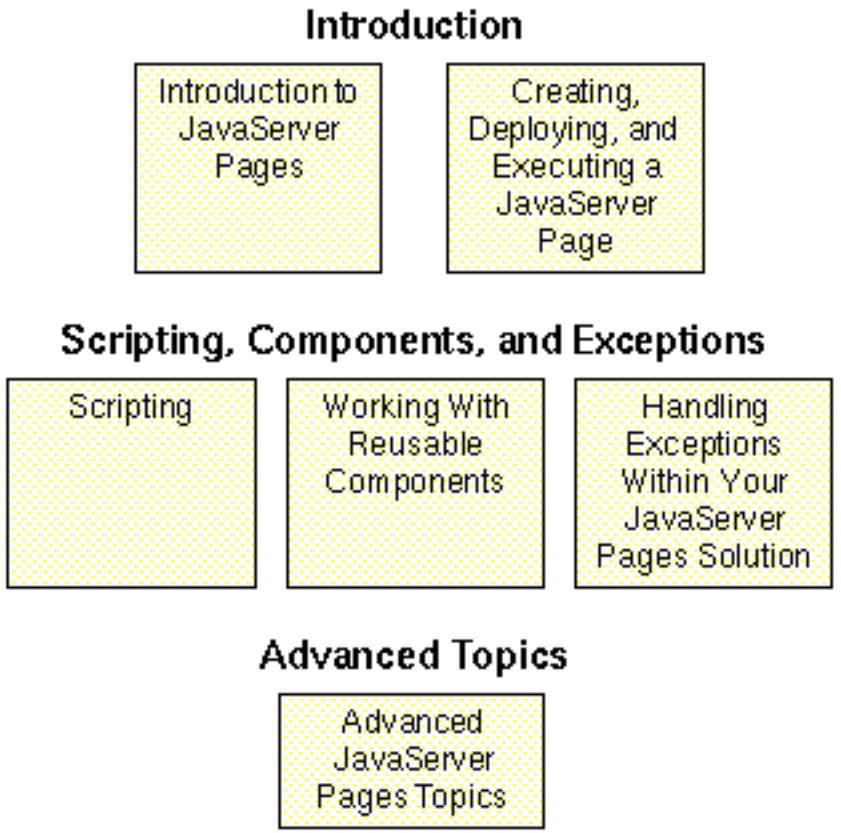
Course Goal

- The *JavaServer Pages™ Technology* course provides students with the knowledge to create Web-based solutions using JSP.

Course Overview

- Overview JSP: compare CGI, servlets, and JSP
- Create a first JavaServer page
- Develop Java™ scriplets
- Learn about component-based JSP solutions
- Handle exceptions with JSP
- Learn about the JSP engine, two- and multi-tier architectures, and custom JSP tag sets
- Create and deploy several Web-based solutions

Course Map



Module-by-Module Overview

- Module 1 – "Introduction to JavaServer Pages"
- Module 2 – "Creating, Deploying, and Executing a JavaServer Page"
- Module 3 – "Scripting"
- Module 4 – "Working With Reusable Components"
- Module 5 – "Handling Exceptions Within Your JavaServer Pages Solution"
- Module 6 – "Advanced JavaServer Pages Topics"

Course Objectives

- Compare and contrast JSP with CGI and servlet technologies
- Develop a basic JavaServer page
- Deploy JavaServer Pages
- List JSP directives
- Integrate JSP with JavaBeans™ components
- Handle JSP exceptions

Course Objectives

- Compare two-tier and multi-tier Web application architectures
- Explain advanced JSP features such as custom tag sets and the `javax.servlet.jsp` package

Skills Gained by Module

Meaning of:

- Black and grey boxes

Skills Gained	Module 1	2	3	4	5	6
Apply JavaServer Pages and servlets appropriately in your projects	▒					▒
Develop JavaServer Pages		■	■	■	■	
Use directives within your JavaServer Pages		■	▒	▒	▒	
Deploy and test JavaServer Pages		■	▒	▒	▒	
Use scripting elements within your JavaServer Pages			■	▒	▒	
Use actions to access component functionality from your JavaServer Pages				■	▒	
Create an exception tracking JavaServer page					■	
Identify advanced JavaServer Pages functions						■

Guidelines for Module Pacing

Module	**Day 1**
"About This Course"	A.M.
"Introduction to JavaServer Pages"	A.M.
"Creating, Deploying, and Executing a JavaServer Page"	A.M.
"Scripting"	A.M. / P.M.
"Working With Reusable Components"	P.M.
"Handling Exceptions Within Your JavaServer Pages Solution"	P.M.
"Advanced JavaServer Pages Topics"	P.M.

Topics Not Covered

- Object-oriented concepts – Covered in SL-210: *Migrating to OO Programming With Java Technology*.
- Object-oriented design and analysis – Covered in OO-226: *Object-Oriented Application Analysis and Design for Java Technology (UML).*
- Java programming language constructs – Covered in SL-110: *Java Programming for Non-Programmers* and SL-275: *Java Programming*.
- System administration concepts – Covered in SA-135: *Solaris 2.X Administration Essentials* and SA-285: *Solaris 2.X System Administration*.

How Prepared Are You?

- Can you develop CGI scripts or servlet extensions to a Web server?
- Can you describe the concept of a servlet?
- Can you create Web pages using Hyper Text Markup Language (HTML) or a similar markup language?
- Can you load and use a Web browser?
- Can you describe the concept of, and use, a Web server?

Introductions

- Name
- Company affiliation
- Title, function, and job responsibility
- Web development experience
- Reasons for enrolling in this course
- Expectations for this course

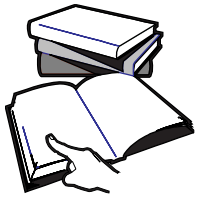
How to Use Course Materials

- Course map
- Objectives
- Relevance
- Overhead image
- Lecture
- Exercise
- Check Your Progress
- Think Beyond

How to Use the Icons

- Demonstration

- Reference



- Discussion

- Exercise

Typographical Conventions and Symbols

- `Courier` is used for the names of commands, files, and directories, as well as on-screen computer output.
- **`Courier bold`** is used for characters and numbers that you type.
- *`Courier italic`* is used for variables and command-line placeholders that are replaced with a real name or value.
- *Palatino italics* is used for book titles, new words or terms, or words that are emphasized.

Java Programming language examples use the following additional conventions:

- `Courier` is used for the class names, methods, and keywords.
- Methods are not followed by parentheses unless a formal or actual parameter list is shown.
- Line breaks occur where there are separations, conjunctions, or white space in the code.
- If a command is different on the Solaris and Microsoft Windows platforms, both commands are shown.

Module 1

Introduction to JavaServer Pages

Overview

- Objectives
- Relevance

History of Web Application Development

- Dynamic content:
  - Began with CGI scripts
  - Improved with Java servlets
- JavaServer Pages (JSP) – filling the gaps:
  - JSP technology addresses the shortcomings of CGI-BIN and Java servlets
  - Based on extensive industry cooperation

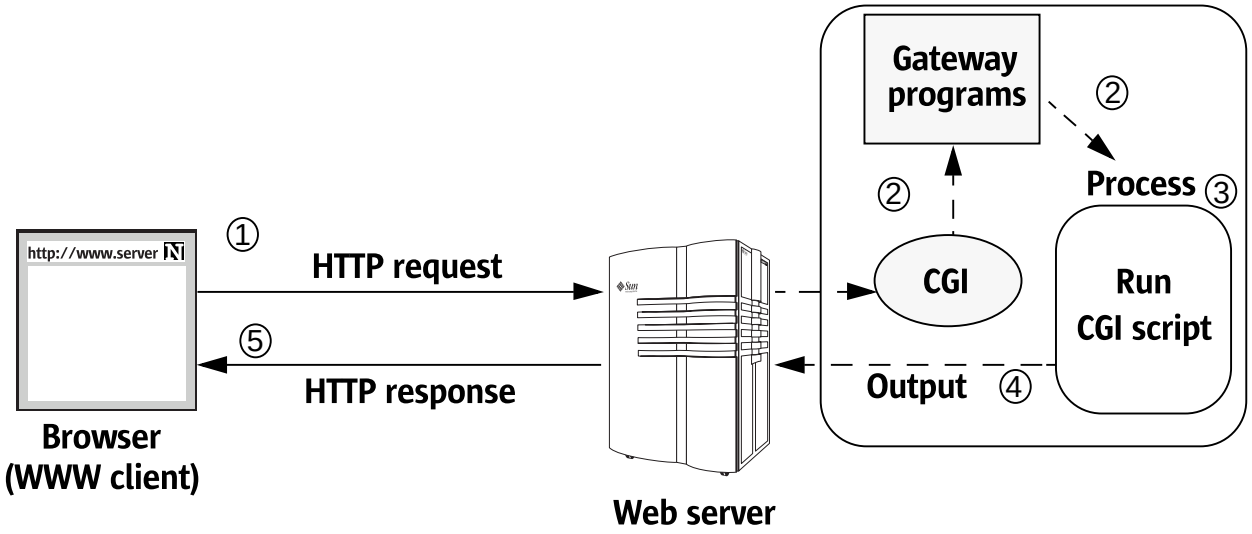
An Overview of Web Application Development

- There are three primary Web server technologies (called extensions):
  - Common Gateway Interface (CGI) scripts
  - Java servlets
  - JavaServer Pages (JSP)

A Review of Common Gateway Interface (CGI)

- CGI scripts execute programs on the server
- CGI scripts can be written using C, C++, Visual Basic, and Perl

A Review of CGI Request and Response



An Example `HelloWorld` CGI Script

```
1 #!/bin/perl
2
3 # Print out a content-type for HTTP/1.0 compatibility
4 print "Content-type: text/html\n\n";
5
6 print "<HTML><HEAD><TITLE>Hello World</TITLE>
</HEAD><BODY><h1>Hello World</h1></BODY></HTML>";
```

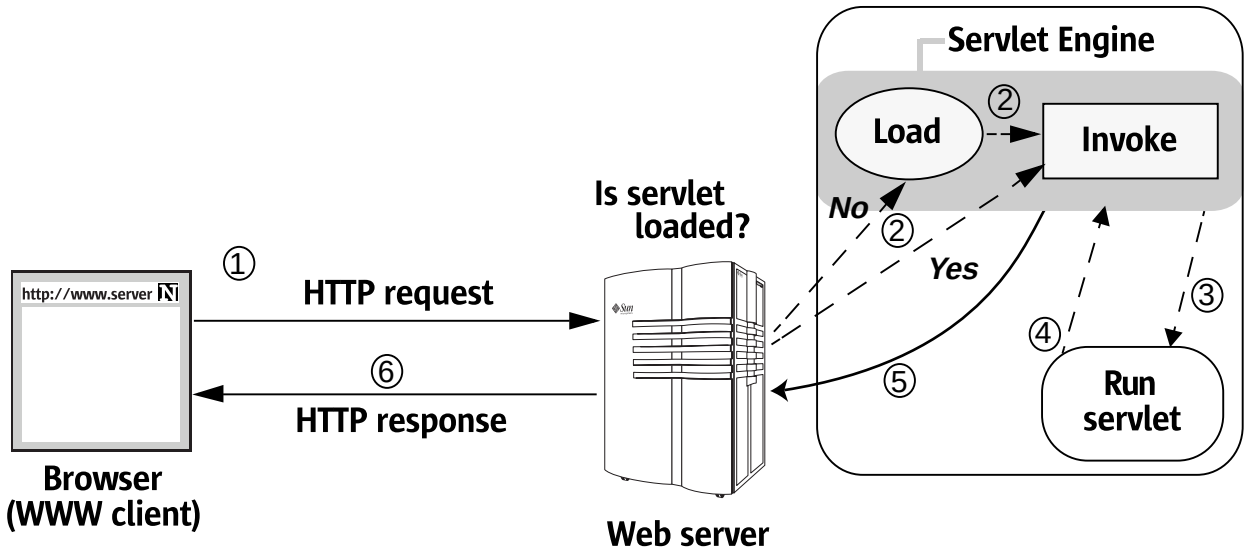
The Benefits of CGI Scripts

- Scripts can be written with any programming or scripting language supported by a Web server
- Scripts extend the functionality of the Web server
- Clients can execute scripts on the server

A Review of Java Servlets

- Java technology is the technology of choice for extending and enhancing Web servers.
- Java servlets are similar to applets except they run on the server side.

A Review of Java Servlet Request and Response



An Example HelloWorld Servlet

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4
5  public class HelloWorld extends HttpServlet{
6
7     public void doGet (HttpServletRequest req,
8        HttpServletResponse res) {
9
10       res.setContentType("text/html");
11
12       try{
13          PrintWriter out = res.getWriter();
14          out.println("<HTML>");
15          out.println("<HEAD><TITLE>Hello World
              </TITLE></HEAD>");
16          out.println("<BODY>");
17          out.println("<h1>Hello World</h1>");
18          out.println("</BODY></HTML>");
19          out.close();
20       } catch(IOException ioe) {
21            getServletContext().log (ioe,"Error in HelloWorld");
22       }
23    }
24 }
```

The Benefits of Java Servlets

- Component-based, platform- and server-independent
- No CGI limitations
- Abundant third-party tool and Web server support
- Access to entire family of Java APIs
- Performance and scalability
- Reliability

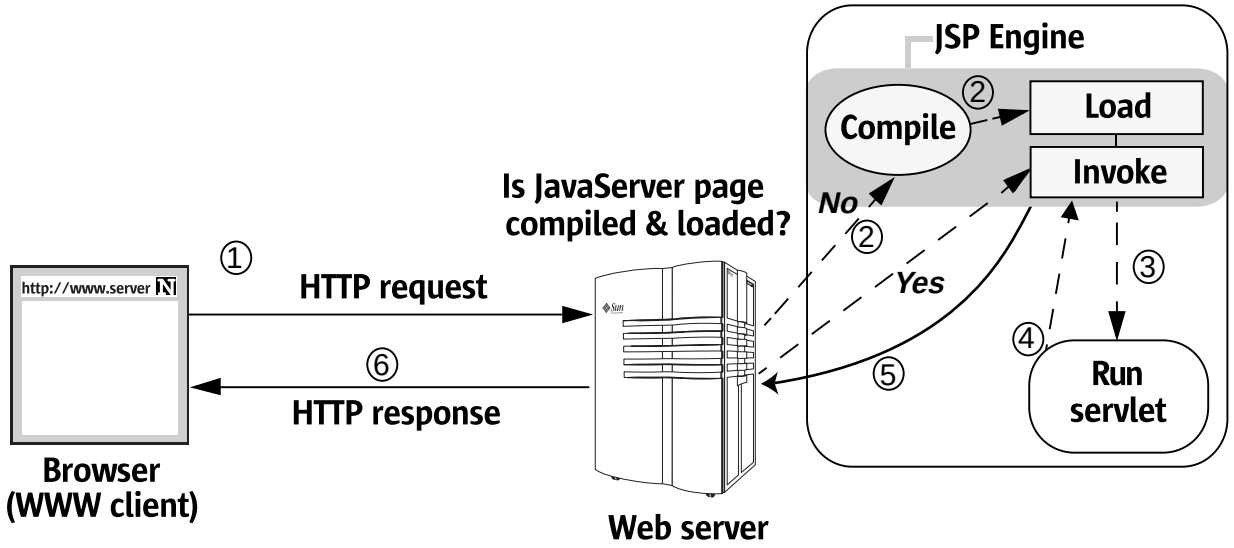
A Review of the Shortcomings of Servlets and CGI

- Solutions prevent software reuse by combining HTML and code
- Solutions require Web designer to have expertise in both Web content and code development
- CGI-based Web applications are difficult to maintain, non-scalable, non-manageable, and platform- and application-specific

JavaServer Pages

- Are text-based documents capable of returning dynamic content to a client browser
- Can contain a mix of HTML code, programming code, and JSP tags
- Allow access to components

JavaServer Pages Request and Response



The Benefits of JavaServer Pages

- Support a component model and software reuse through the use of components
- Recompile automatically when changes are made to the source file
- Simplify page development with JSP and custom tags
- Ability to seperate the Web content from the code
- Are platform-independent

The Benefits of JavaServer Pages

- Performance and scalability
- Reliability
- Integrate into enterprise as part of J2EE

JavaServer Pages Versus Servlets

- Recommended Uses of Servlets:
  - Extend the functionality of a Web server
  - Generate objects that do not contain HTML
  - Initialize a Web application
- Recommended Uses of JavaServer Pages:
  - Access application logic separated from Web content and embedded in components
  - Present dynamic portions of content, which is tailored to a specific user.

Check Your Progress

- Compare and contrast three methods for creating dynamic HTML
- Discuss the separation of business logic and content within JavaServer Pages
- Compare the primary uses for servlets with the primary uses for JavaServer Pages

Think Beyond

What are some other reasons for using JavaServer Pages instead of CGI scripts or Java servlets?

Module 2

Creating, Deploying, and Executing a JavaServer Page

Overview

- Objectives
- Relevance

JavaServer Pages Development Preparation

- To develop, deploy, and test JavaServer Pages, you need:
  - A Web browser
  - A Web server supporting JSP and servlets
  - A text-based editor

Your First JavaServer Page

Hello, World!

Your First JavaServer Page

```
1  <%@ page info="a hello world example" %>
2
3  <html>
4  <head><title>Hello, World</title></head>
5  <body bgcolor="#ffffff" background="background.gif">
6
7  <%@ include file="dukebanner.html" %>
8
9  <center>
10 <h1>Hello, World!</h1>
11 </center>
12
13 </body>
14 </html>
```

Your First JavaServer Page

- JSP element syntax
  - Start and end tags <% %>
- Directives
  - The page directive
  - <%@ page info="a hello world example" %>
  - The include directive
  - <%@ include file="dukebanner.html" %>
- Saving JavaServer Pages

Deploying a JavaServer Page

1. Create a directory to hold the JavaServer Pages.
2. Copy your JSP files to the newly-created directory.

Executing and Testing a JavaServer Page

1. Load your Web server.
2. Load your Web browser.
3. Access your JavaServer page.

Debugging and Development Tips

- Develop your JavaServer Pages incrementally
- If page does not compile:
  - Examine errors or exceptions displayed by the Web server
  - Use "trial-and-error" method to debug the page
- Create an exception page

Exercise: Create, Deploy, and Test a Basic JavaServer Page

- Objectives
- Tasks
- Discussion
- Solutions

Check Your Progress

- Prepare for JavaServer page development
- Write a JavaServer page
- Discuss the `page` directive
- Explain the `include` directive
- Deploy your first JavaServer page
- Execute and test a JavaServer page

Think Beyond

Aside from headers and footers, what are some other areas on a Web site applicable for HTML reuse?

Module 3

Scripting

Overview

- Objectives
- Relevance

Scripting Elements

- Three classes of scripting elements to insert Java code into your JavaServer Pages:
  - Declarations <%! %>
  - Scriptlets <% %>
  - Expressions <%= %>
  - Directives <%@ %>

Declarations

- Used to identify variables, methods, and other scripting language constructs
- Syntax:

  `<%! declaration %>`

- Examples:
  - Variable declaration

    ```
    <%! int i = 0; %>
    ```

  - Method declaration

    ```
    <%! public String f(int i) { if (i<3) return("...");
    ...} %>
    ```

Declarations

- Declaration rules:
  - Variables declared result in member variables in the compiled servlet. These will be shared by all users who make simultaneous requests on the same JSP.
  - You must declare a variable or method in a JavaServer page before you use it in the page.
  - The scope of the declaration is usually a JSP file. However, if the JSP file includes other files within the `include` directive, the scope expands to cover the included files as well.
  - Declarations must end with a semi-colon. You can also use semi-colons to separate two or more declarations.

Scriptlets

- Programming code fragments that perform tasks beyond the capabilities of markup languages
- Syntax:

  `<% scriptlet %>`

- Example – Determining the time of day by accessing functionality within a calendar object:

```
<% if (Calendar.getInstance().get(Calendar.AM_PM) ==
Calendar.AM) {%>
Good Morning
<% } else { %>
Good Afternoon
<% } %>
```

Scriptlets

- Scriptlet rules:
  - The scripting language you use determines the rules for the scriptlet.
  - A scriptlet statement must end in a semi-colon if required by the scripting language.
  - You can use any of the objects or classes imported into the JavaServer page using the `page` directive, declared in a `declaration`, or identified within a `useBean` tag.

Expressions

- Scripting language items evaulated during the generation of a response
- Syntax:

  `<%= expression %>`

- Example – Inserting the current date into a JavaServer page:

```
<%= (new java.util.Date()).toLocaleString() %>
```

Expressions

- Expression rules:
  - The scripting language you use determines the rules for expressions.
  - Expressions are evaluated in a left-to-right order.
    <%= count++ %> <%= newValue=count %>
  - Semi-colons are not allowed for expressions.

A Simple Visitor Count Example

- SimpleVisitor.jsp

```
1   <html>
2   <head><title>My Page</title></head>
3   <body>
4   <%! int count = 0; %>
5   <H1>Welcome to my page.</H1>
6   <% count++; %>
7   You are my <%= count %> visitor.
8   </BODY>
9   </HTML>
10
```

- Resulting servlet

```
1   public class JSPGeneratedServlet extends HTTPServlet {
2     int count = 0;    // declaration creates member
3
4     public void service(. . .) {
5       // Code to create HTML here
6
7       count++;      // scriplet added to service
8
9       // More code to create next HTML
10
11      out.println(count);// addition from expression to output
12
13      // End of HTML
14    }
15  }
```

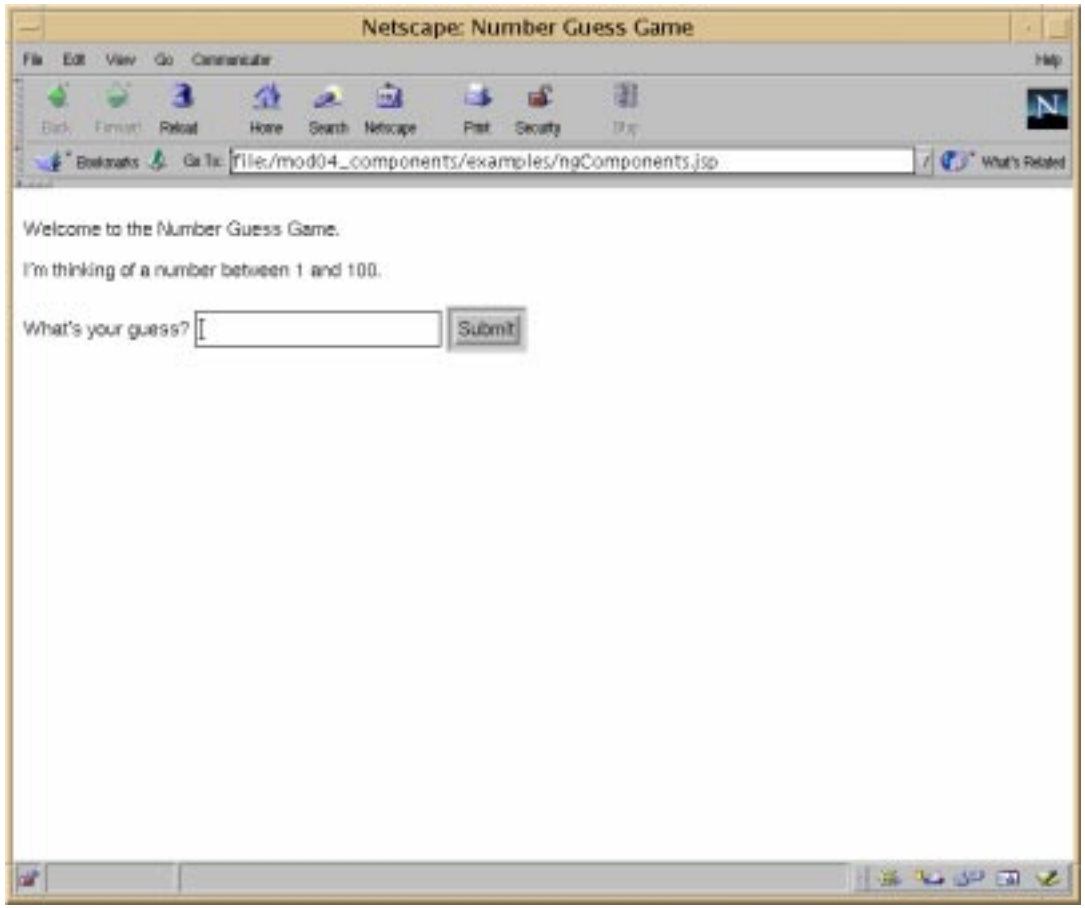
The Number Guess Game Example

```
1  <!--
2    Number Guess Game
3    Based on the JavaServer Page Written by Jason Hunter
<jasonh@kasoftware.com>, CTO, K&A Software
4    Copyright 1999, K&A Software, distributed by Sun with permission
5  -->
6
7  <html>
8  <head><title>Number Guess</title></head>
9  <body>
10 <%@ page import = "java.util.*" %>
11
12 <%! int answer = 0; %>
13 <%! int numGuesses=0; %>
14
15 <% String guess = request.getParameter("guess");
16
17    if(guess == null) { %>
18
19       Welcome to the Number Guess game.<p>
20       <% answer = Math.abs(new Random().nextInt() % 100) + 1;
21       numGuesses = 0;
22
23    } else {
24
25       int value = Integer.parseInt(guess);
26
27       if(value == answer) { %>
28
29          Congratulations! You got it.<br>
30          And after just <%= numGuesses %> tries.<p>
31
32           Care to <a href="ngScriplet.jsp">try again</a>?<p>
33
34       <% } else { %>
35
```

The Number Guess Game Example

```
36          Good guess, but nope. Try
37
38          <% numGuesses++;
39
40          if(value < answer) { %>
41
42              <b>higher</b>.<p>
43
44           <% } else if(value > answer) { %>
45
46              <b>lower</b>.<p>
47
48           <% }
49
50         }
51
52     } %>
53     I'm thinking of a number between 1 and 100.<p>
54
55     <form method=get>
56        What's your guess? <input type=text name=guess>
57        <input type=submit value="Submit">
58     </form>
59
60 </body>
61 </html>
```

The Number Guess Game Example



The Number Guess Game Example

- Reviewing form processing
  - Using HTTP `GET` and `POST` methods
- Declaring variables
- Implicit object references
  - The `request` implicit object reference

```
<% if (request.getParameter("guess") != null) { %>
     <%@ include file="response.jsp" %>
<% } %>
```

The Number Guess Game Example

- Implicit object references
  - The `request` implicit object reference

Expression/Scriptlet	**Use**
`String getParameter(name)`	Returns the value of a parameter if you provide the name.
`Enumeration getParameterNames()`	Returns an enumeration of strings containing the names of the parameters that the request currently contains.
`String[] getParameterValues(name)`	Returns an array of strings containing values of the parameters that the request currently contains.

The Number Guess Game Example

- Coding scriptlets
- Intermingling scriptlet elements with tags

```
<% } else if (condition) {
    out.println("print a message");
} else { %>
  print a different message
  <% } else if (another_condition) { %>
    print a message
  <% } else if (yet_another_condition) { %>
    print an another message
  <% } %>
<% } %>
```

Comments

- There are two types of comments in JSP:
  - Comments that document what the JavaServer page is doing. The following is the syntax for these comments:

```
<%-- this is a comment ... --%>
```

or

```
<% /** this is a comment ... **/ %>
```

  - Comments that are sent as a response to users. The following is the syntax for these comments:

```
<!-- comments ... -->
```

or

```
<!-- comments <%=expression %> more comments ... -->
```

JavaServer Pages Processing

- A JSP source file is processed in two stages:
  - JSP Page Translation – The page is compiled into a Java class. All HTML tags and all JSP tags are processed (to create a servlet), however, the scriplets and expressions are not executed.
  - Request Processing – This happens when the URL requested by the client browser is directed by the Web server to a JavaServer page.
    A request object is created, parsed, and submitted to the compiled JavaServer page servlet.
    When the servlet processes the request it executes the previously processed scriplets and expressions.

Disadvantages and Guidelines for Using Scripting Code

- Disadvantages:
  - Overuse of scripting code can make JavaServer Pages confusing and difficult to maintain.
  - Scripting code defeats two main JSP advantages: software reuse and separation of programming from content.
- Guidelines:
  - Use scripting code only when component functionality is unavailable or when a JavaServer page requires limited scripting.

Exercise: Incorporating Scripting Into a JavaServer Page

- Objectives
- Tasks
- Discussion
- Solutions

Check Your Progress

- List the three categories of JSP scripting elements
- Make declarations within your JavaServer Pages
- Create scriptlets within your JavaServer Pages
- Use expressions within your JavaServer Pages
- Identify the phase in which each category of scripting element is evaluated

Check Your Progress

- List advantages and disadvantages of scripting within a JavaServer page

Think Beyond

How do you foresee using scripting in your JavaServer Pages?

Module 4

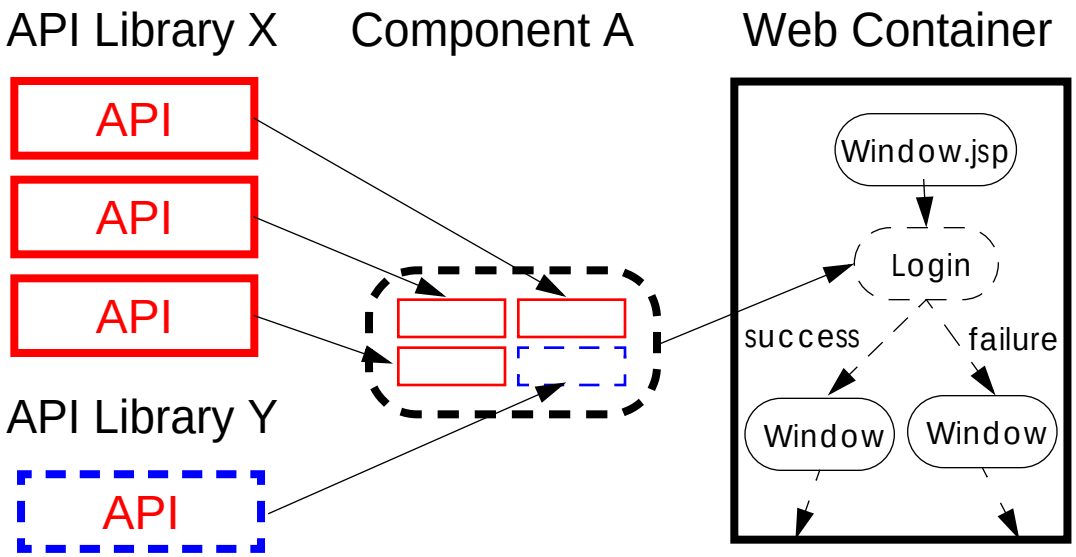
Working With Reusable Components

Overview

- Objectives
- Relevance

What Are Software Components?

- Collections of useful, low-level APIs grouped into reusable programs that perform high-level tasks



What Is JavaBeans?

- JavaBeans is a portable, platform-independent component model written in Java for creating reusable components
- Beans can be combined to create robust, cross-platform applets and applications

What Is Enterprise JavaBeans?

- A server-side component architecture for rapid and simplified development of distributed, secure, and portable enterprise applications, such as:
  - Transaction processing
  - Object-to-relational mapping
  - Business logic encapsulation
- Types of enterprise Beans:
  - Session Beans
  - Entity Beans

JavaBeans Versus Enterprise Beans

JavaBeans	**Enterprise Beans**
Visual and non-visual. Can be deployed on client and server.	Non-visual. Deployed only on a server.
Deployed as any Java applet or application class.	Deployed in a *container* that manages the propagation of transactions, security, concurrency, and state (persistence).
Properties and behaviors usually introspected by a builder tool.	Properties and context discovered by container using standardized *deployment descriptor* file accompanying each Bean.
Events driven.	Although events are normally not used, EJBs can use events using Java Message Service (JMS).

Components and JavaServer Pages

- JavaServer Pages can access Beans and Enterprise JavaBeans as needed.
- Actions
  - JavaServer Pages use `action` tags to use, modify, and create server-side objects (such as Beans).

The Revised Number Guess Game Example

- Uses JavaBeans instead of declarations and scriptlets for random number generation

The Revised Number Guess Game Example

```
1  <!--
2    Number Guess Game
3    Based on the Number Guess Game Written by Jason Hunter
4     <jasonh@kasoftware.com>, CTO, K&A Software
5    Copyright 1999, K&A Software, distributed by Sun with permission
6  -->
7
8  <%@ page import = "numguess.NumberGuessBean" %>
9
10 <jsp:useBean id="numguess" class="numguess.NumberGuessBean"
scope="session"/>
11 <jsp:setProperty name="numguess" property="*"/>
12
13 <html>
14 <head><title>Number Guess</title></head>
15 <body bgcolor="white">
16 <font size=4>
17
18 I'm thinking of a number between 1 and 100.<p>
19
20 <form method=get>
21 What's your guess? <input type=text name=guess>
22 <input type=submit value="Submit">
23 </form>
24
25 <% if (numguess.getSuccess()) { %>
26
27   Congratulations!  You got it.
28   And after just <jsp:getProperty name="numguess"
       property="NumGuesses"/> tries.<p>
29
30   <% numguess.reset(); %>
31
32   Care to <a href="numguess.jsp">try again</a>?
33
34 <% } else if (numguess.getNumGuesses()!=0) { %>
```

The Revised Number Guess Game Example

```
35
36   Good guess, but nope.  Try <b><jsp:getProperty
name="numguess"
       property="Hint"/></b>.
37
38   You have made <jsp:getProperty name="numguess"
       property="NumGuesses"/> guess(es).<p>
39
40 <% } %>
41
42 </font>
43 </body>
44 </html>
```

The Revised Number Guess Game Example

- Sending data from the form to a component
- The `jsp:useBean` action
  - Attributes
  - With a body

```
<jsp:actionName id="name"
scope="page|request|session|application|  typeSpec >
     body
</jsp:useBean>
```

The JSP:useBean Action

Attribute	Meaning
id	The name used to identify the object instance in the specified scope's namespace, and also the scripting variable name declared and initialized with that object's reference.
scope	The scope within which the reference is available. Valid values for scope are page, request, session and application. Page: The bean will exist in one page per request.This is the default. Request: The bean will exist in all pages in this request. Session: The bean will exist in all pages of this session. Application: The bean will exist shared in the web container.
class	The fully qualified name of the class that defines the implementation of the object. If the class and the `beanName` attributes are not specified, the object must be present in the given scope.
beanName	The name of the Bean.
type	Defines the type of the object referenced by the `id` attribute. If unspecified, the value is the same as the value of the `class` attribute.

The Revised Number Guess Game Example

- The `jsp:setProperty` action
  - Attributes
- Getting data from a component
  - The `jsp:getProperty` action
  - Attributes

The NumberGuessBean Class

```
1  // Number Guess Game
2  // Based on the JavaBean Written by Jason Hunter
3  // <jasonh@kasoftware.com>, CTO, K&A Software
4  // Copyright 1999, K&A Software, distributed by Sun with permission
5
6  package numguess;
7
8  import java.util.*;
9
10 public class NumberGuessBean {
11
12    int answer;
13    boolean success;
14    String hint;
15    int numGuesses;
16
17    public NumberGuessBean() {
18     reset();
19    }
20
21    public void setGuess(String guess) {
22     numGuesses++;
23
24     int g;
25
26     g = Integer.parseInt(guess);
27
28     if (g == answer) {
29       success = true;
30     }
```

The NumberGuessBean Class

```
31    else if (g < answer) {
32       hint = "higher";
33     }
34     else if (g > answer) {
35       hint = "lower";
36     }
37    }
38
39    public boolean getSuccess() {
40     return success;
41    }
42
43    public String getHint() {
44     return hint;
45    }
46
47    public int getNumGuesses() {
48     return numGuesses;
49    }
50
51    public void reset() {
52     answer = Math.abs(new Random().nextInt() % 100) + 1;
53     success = false;
54     numGuesses = 0;
55    }
56 }
```

The `NumberGuessBean` Class

- Contains one set method:
  - `setGuess`
- Contains three get methods:
  - `getSuccess`
  - `getHint`
  - `getNumGuesses`

Exercise: Migrate the Payment Calculator to a Component Solution

- Objectives
- Tasks
- Discussion
- Solutions

Check Your Progress

- Define component
- Identify two Java-based component architectures
- Describe actions and attributes
- Identify the purpose of the `jsp:useBean` and `jsp:setProperties` actions
- Create a component-based JavaServer Pages solution

Think Beyond

What types of Beans do you foresee using in your JSP solutions?

Module 5

Handling Exceptions Within Your JavaServer Pages Solution

Overview

- Objectives
- Relevance

Run-Time Exceptions

- Run-time exceptions are recoverable errors that occur when a program is running.
- Exception information is available using an implicit `exception` object reference.
- You can create or generate a JavaServer page that utilizes the `exception` reference and displays exception information for users.

Creating an Exception Tracking Solution

1. Determine the exceptions thrown.
2. In each of your JavaServer Pages, include the name of the exception page you are going to create.
3. Develop an exception page.
4. In the exception page, use the `exception` reference to display exception information.
5. (Optional) – Integrate a tracking mechanism to determine what the user was doing when the exception occurred.

The Revised Number Guess Game Example With Exception Handling

```
1  <!--
2    Number Guess Game
3    Based on the Number Guess Game Written by Jason Hunter
4     <jasonh@kasoftware.com>, CTO, K&A Software
5    Copyright 1999, K&A Software, distributed by Sun with permission
6  -->
7
8  <%@ page import = "numguess.NumberGuessBean"
       errorPage="error.jsp" %>
9
10 <jsp:useBean id="numguess" class="numguess.NumberGuessBean"
       scope="session"/>
11 <jsp:setProperty name="numguess" property="*"/>
12
13 <html>
14 <head><title>Number Guess</title></head>
15 <body bgcolor="white">
16 <font size=4>
17
18 I'm thinking of a number between 1 and 100.<p>
19
20 <form method=get>
21 What's your guess? <input type=text name=guess>
22 <input type=submit value="Submit">
23 </form>
24
25 <% if (numguess.getSuccess()) { %>
26
27   Congratulations!  You got it.
28   And after just <jsp:getProperty name="numguess"
        property="NumGuesses"/> tries.<p>
29
30   <% numguess.reset(); %>
31
```

The Revised Number Guess Game Example With Exception Handling

```
32   Care to <a href="ngComponents.jsp">try again</a>?
33
34 <% } else if (numguess.getNumGuesses()!=0) { %>
35
36   Good guess, but nope.  Try <b><jsp:getProperty name="numguess"
       property="Hint"/></b>.
37
38   You have made <jsp:getProperty name="numguess"
       property="NumGuesses"/> guess(es).<p>
39
40 <% } %>
41
42 </font>
43 </body>
44 </html>
```

- Calling an exception page from another page

The Revised Number Guess Game Example With Exception Handling

```
1  <%@ page isErrorPage="true" import="num.NumberGuessBean" %>
2
3  <html>
4  <head><title>Number Guess</title></head>
5
6  <table border="0" cellspacing="0" cellpadding="5">
7
8  <tr>
9  <td width="150" align="right">   </td>
10 <td align="right" valign="bottom"> <h1> Number Guess </h1> </td>
11 </tr>
12
13 <tr>
14 <td width="150" align="right">   </td>
15 <td align="right"> <b>Oops! an exception occurred.</b> </td>
16 </tr>
17
18 <tr>
19 <td width="150" align="right">   </td>
20 <td align="center">The name of the exception is <%=
     exception.toString() %>.
21 </td>
22 </tr>
23
24 <tr>
25 <td width="150" align="right">   </td>
26 <td align="right">   </td>
27 </tr>
28
29 </table>
30
31 </body>
32 </html>
```

The Revised Number Guess Game Example With Exception Handling

- Writing an exception page
- The `exception` implicit object reference

Expression/Scriptlet	Use
`<%= exception.toString() %>`	Prints the name of the exception.
`<% exception.printStackTrace(); %>`	Prints a list of all errors in the current error stream (stack trace).
`<%= exception.getMessage() %>`	Prints a detailed message for the error.

Writing a Simple Tracking Mechanism

- Bean variable and method declarations:
  - Modify your Beans to contain an `action` property that holds the name of the most recent action the Bean performed.
  - Create `getAction` and `setAction` methods.

```
private String action;

public void setAction( String pageAction ) {
     action=pageAction;
}

public String getAction() {
     return action;
}
```

Writing a Simple Tracking Mechanism

- JavaServer Pages changes:
  - Set the `action` property to a value that represents the current action. For example, if the user is trying to guess a number, set the action property to "guess":

```
<% numguess.setAction (“guess”); %>
```

Writing a Simple Tracking Mechanism

- Exception pages changes:
  - Check the value of the Bean's `action` property when your exception page is invoked, and print additional exception information associated with each value. For example:

```
<% if (numguess.getAction() == "guess" ) { %>
     You must enter a valid number between 1 and 100.
<% } else if (numguess.getAction()="action2") { %>
     ... text message here ...
<% } %>
```

Exercise: Add Exception Handling to a JavaServer Page Solution

- Objectives
- Tasks
- Discussion
- Solutions

Check Your Progress

- Discuss run-time exceptions
- Create an exception page for displaying general exception information
- Designate an exception page within a JavaServer page
- Explain the use of the exception implicit object reference
- Provide detailed information about an exception through a simple exception tracking mechanism

Think Beyond

What are some ways to recover from exceptions, such as when a user does not submit data in a required form field, or a calculation within a component fails?

Module 6

Advanced JavaServer Pages Topics

Overview

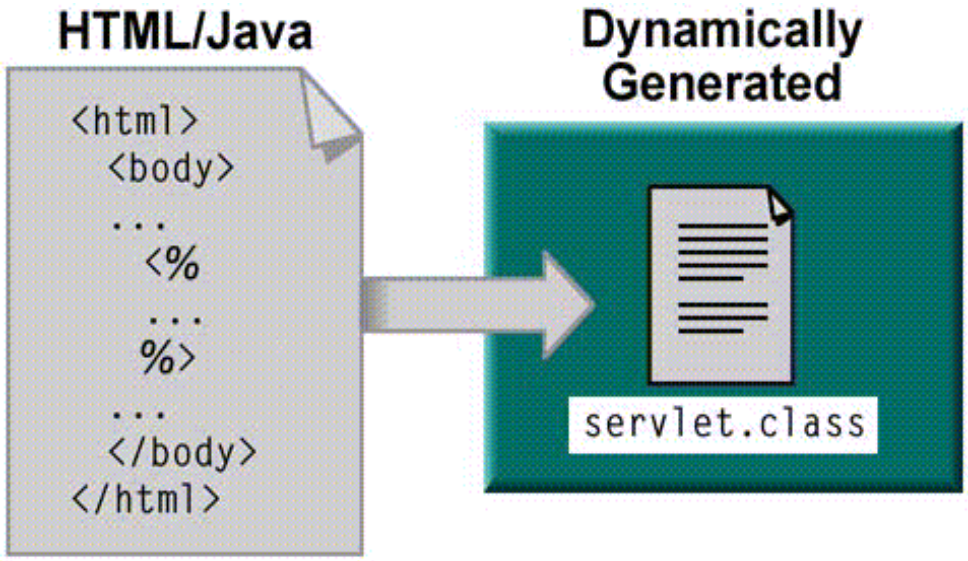
- Objectives
- Relevance

The JavaServer Pages Engine

- Executes JavaServer Pages on a Web server
- Delivers client requests to the appropriate compiled page and returns a response from the page back to the client
- Compiles JavaServer Pages into servlet classes that represent your JavaServer Pages on the Web server

The JavaServer Pages Engine

- Compiling a JavaServer page



The JavaServer Pages Engine

- Compiling a JavaServer page
  - A very simple JavaServer page

```
1 <html>
2 <body>
3 <%@ page info="Example JSP pre-compiled" %>
4 <p>
5 Hello World
6 </p>
7 </body>
8 </html>
```

The JavaServer Pages Engine

- Compiling a JavaServer page
  - The servlet resulting from compiling the JavaServer page

```
1  import javax.servlet.*;
2  import javax.servlet.http.*;
3  import javax.servlet.jsp*;
4
5  class _jsp_HelloWorld_XXX_Impl extends
PlatformDependent_Jsp_Super_Impl {
6     public void _jspInit() {
7        // ...
8     }
9
10    public void jspDestroy() {
11       // ...
12    }
13
14    public void _jspService(HttpServletRequest request,
HttpServletResponse response) throws IOException, ServletException {
15       Object page = this
16       HttpSession session = request.getSession();
17       ServletConfig config = getServletConfig();
18       ServletContext application = config.getServletContext();
19
20       JspFactory _factory = JspFactory.getDefaultFactory();
21       PageContext pageContext = _factory.getPageContext(this,
request, response, (String)NULL, true, JspWriter.DEFAULT_BUFFER,
true);
22       JspWriter out = pageContext.getOut();
23       // page context creates initial JspWriter “out”
24
25       try {
```

The JavaServer Pages Engine

- Compiling a JavaServer page
  - The servlet resulting from compiling the JavaServer page

```
26          out.println(“<p>”);
27        out.println(“Hello World”);
28          out.println(“</p>”);
29        } catch (Exception e) {
30          pageContext.handlePageException(e);
31        } finally {
32          _factory.releasePageContext(pageContext);
33        }
34     }
35 }
```

The JavaServer Pages Packages

- JSP engines contain two packages that let you access internal JSP mechanisms within your JavaServer Pages:
  - `javax.servlet.jsp`
  - `javax.servlet.jsp.tagext`

The `javax.servlet.jsp` Package

- `JspPage` interface
- `HttpJspPage` interface
  - Redefining the `jspInit` method within a JavaServer page
  - Redefining the `jspDestroy` method within a JavaServer page

The `javax.servlet.jsp` Package

- Abstract classes within `javax.servlet.jsp` package
  - `JspEngineInfo` class
  - `JspFactory` class

Custom Tag Libraries

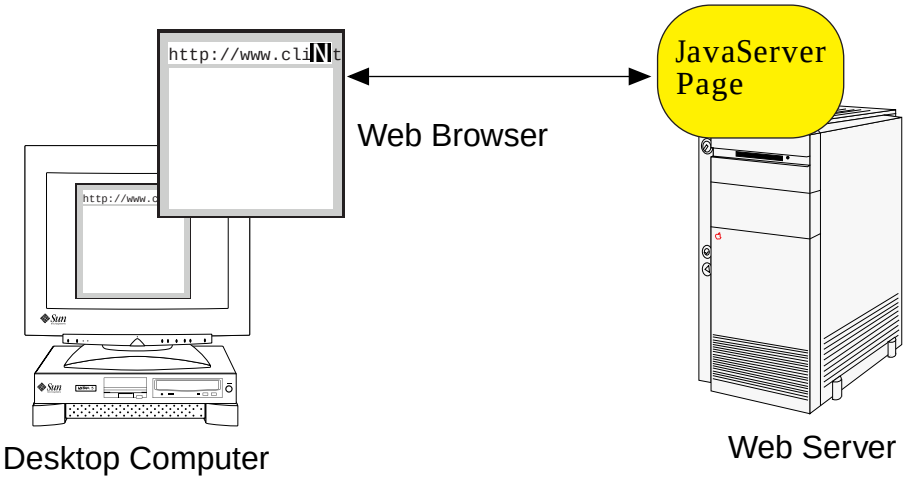
- Benefits
- Development overview
  - Create tag handlers for each action
  - Create a Tag Library Descriptor (TLD)

Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- An enterprise or Web application can have two or more tiers (hardware or software components)
- Two-tier architectures
- Multi-tier architectures

Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Two-tier architectures
  - Example 1: Web Browser and JSP

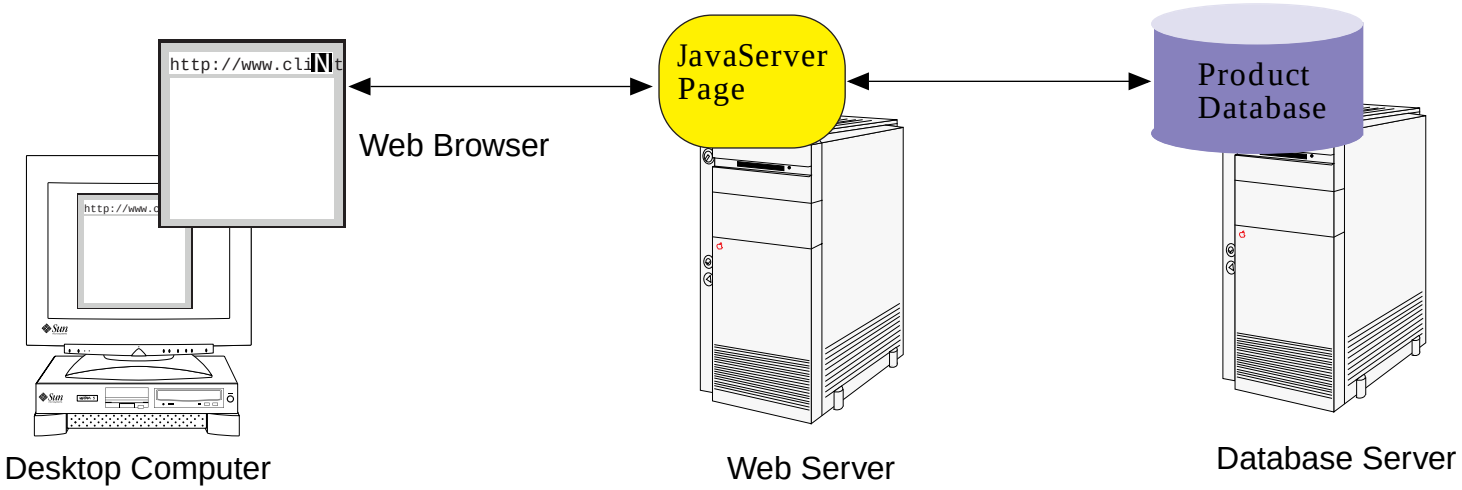


Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Two-tier architectures
  - Advantages
  - Disadvantages

Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Multi-tier architectures
  - Example 2: Web Browser, JSP, and JDBC

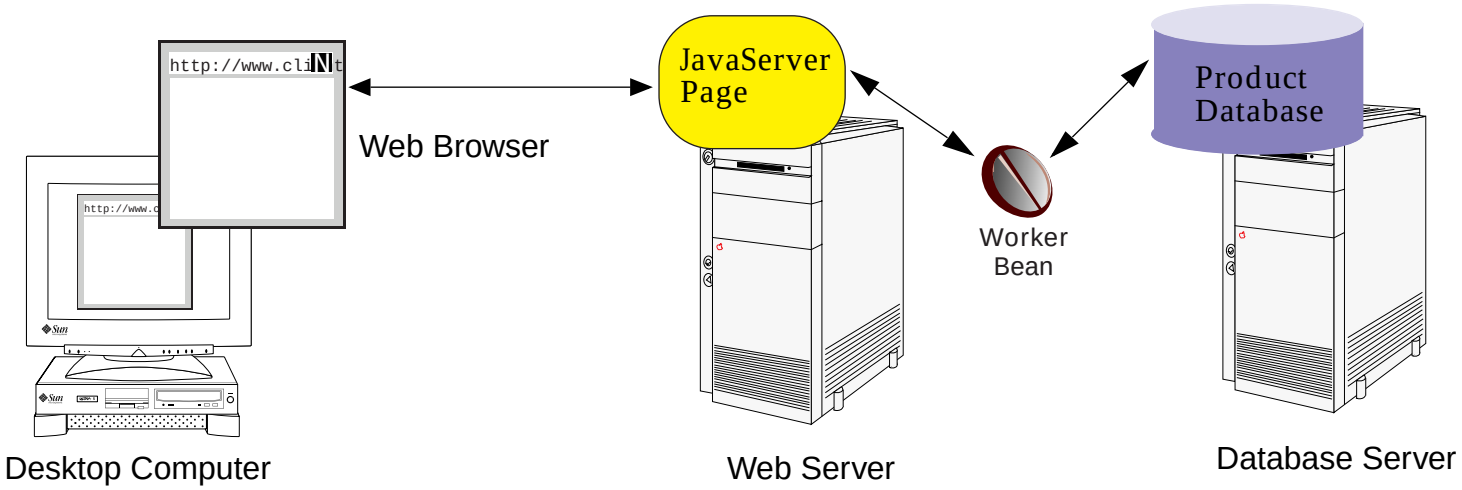


Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Multi-tier architectures
  - Advantages
  - Disadvantages

Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Multi-tier architectures
  - Example 3: Web Browser, JSP, Worker Beans, and JDBC

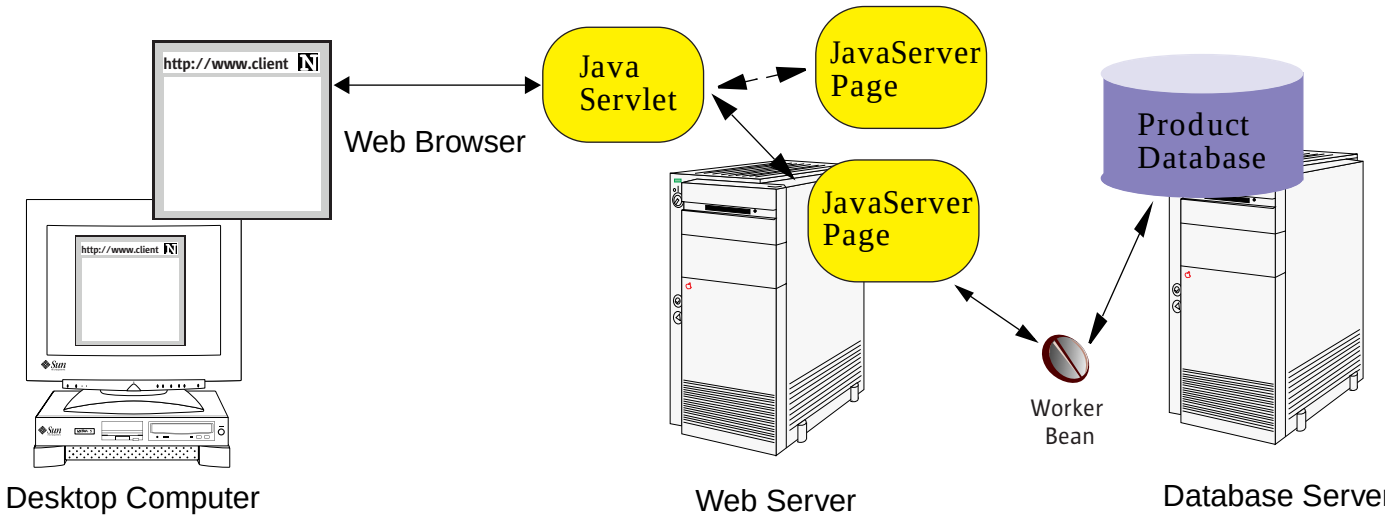


Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Multi-tier architectures
  - Advantages
  - Disadvantages

Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Multi-tier architectures
  - Example 4: Web Browser, Servlets, JSP, Worker Beans, and JDBC



Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Multi-tier architectures
  - Advantages
  - Disadvantages

Combining JavaServer Pages, Servlets, Worker Beans, and Enterprise JavaBeans Components

- Multi-tier architectures
  - Example 5: Web Browser, Servlets, JSP, Worker Beans, Enterprise JavaBeans and JDBC
  - Sun BluePrints Design Guidelines for the J2EE

Check Your Progress

- Discuss the primary tasks of the JSP engine
- Identify two interfaces within the `javax.servlet.jsp` package
- Explain the concept of a custom tag library
- Compare two-tier and multi-tier Web application architectures

Think Beyond

How do you foresee using JavaServer Pages in your Web solution architecture? Do any of the architectures in this module apply to your solution?

Course Contents