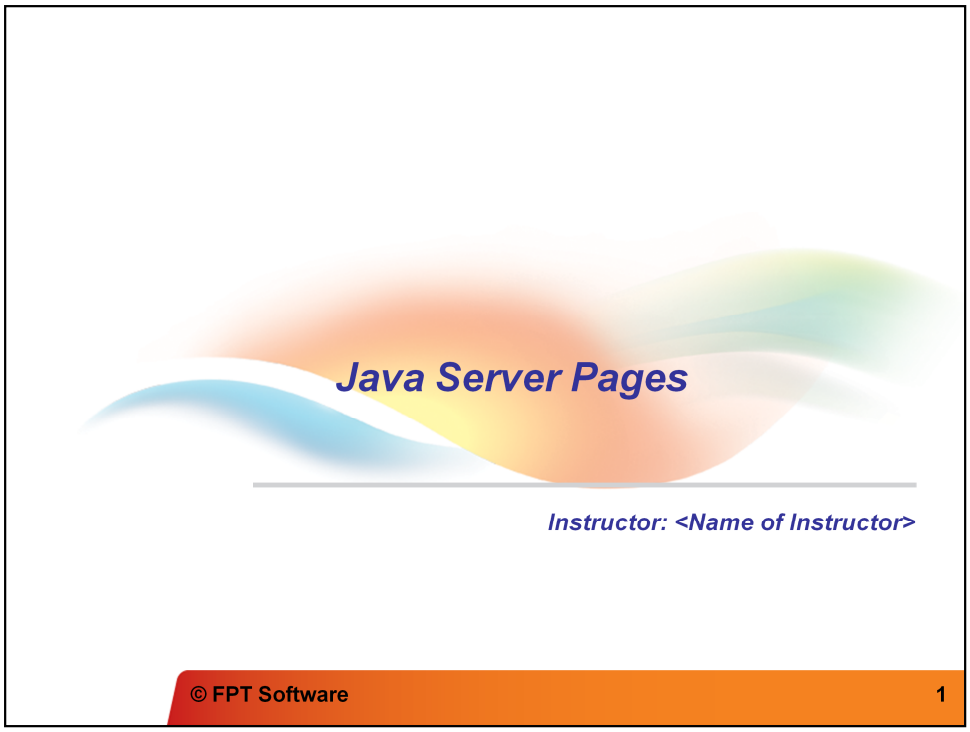


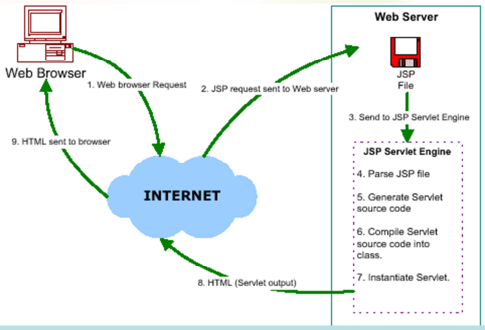
Latest updated by: HanhTT1

Agenda

- Introduction
- A First JSP Example
- Implicit Objects
- Scripting
- Standard Actions
- Directives
- Database Connectivity
- Session Tracking
- Exception Handling

JSP Introduction

- As an extension of Servlet technology, JSPs are essential an HTML page with special JSP tags embedded. These JSP tags can contain Java code. The JSP file extension is .jsp rather than .html or .html
- Steps required for a JSP request:



1. The user goes to web side made using JSP. The web browser makes the request via the Internet.
2. The JSP request gets sent to the Web server.
3. The Web server recognizes that the file required is special (.jsp), therefore passes the JSP file to the JSP servlet engine
4. If the JSP file has been called the first time, the JSP file is parsed, otherwise goto step 7.
5. The next step is to generate a special Servlet from the JSP file. All the HTML required is converted to println statements.
6. The Servlet source code is compiled into a class.
7. The servlet is instantiated, calling the init and service methods.
8. HTML from the Servlet output is sent via the Internet.
9. HTML results are displayed on the user's web browser.

JSP Introduction (cont.)

- Servlets
  - Used when small amount of content is fixed-template data
    - Most content generated dynamically
  - Some servlets do not produce content: invoke other servlets and JSPs
- JSPs
  - Look like standard HTML or XHTML
    - Normally include HTML or XHTML markup
      - Known as fixed-template data
  - Used when content is mostly fixed-template data
    - Small amounts of content generated dynamically
  - Use special tags (JSP tags) to include existing Java components OR codes: declaration, expression, directive, scriptlet, action, custom tag library

JSP Introduction (cont.)

- Declaration tag (<%! %>)
  - Allow the developer to declare variables or methods
  - For Example

```
<%!
  private int counter = 0;
  private String getAccount(int accountNo);
%>
```

- Expression tag (<%= %>)
  - Allow the developer to embed any Java expression and is short for out.println()
  - For Example

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

JSP Introduction (cont.)

- Directive (<%@ directive... %>
  - Give special information about the page to the JSP container
  - Enable programmers to specify:
    - Page settings (page directive)
    - Content to include from other resources (Include directive)
    - Tag libraries to be used in the page (Tag library)
- Action (<%action... %>
  - Predefined JSP tags that encapsulate functionality
  - Often performed based on information from client request
  - Can be used to create Java objects for use in scriptlets

JSP Introduction (cont.)

- Scriptlet (<% … %>) - also called “Scripting Elements”
  - Enable programmers to insert Java code in JSPs
  - Performs request processing
    - Interacts with page elements and other components to implement dynamic pages
    - Can access any variable or bean declared.
  - For example, to print a variable:

```
<%
    String username = “alliant”;
    out.println(“username”);
%>
```

- Custom Tag Library
  - JSP’s tag extension mechanism
  - Enables programmers to define new tags
    - Tags encapsulate complex functionality
  - Tags can manipulate JSP content

A First JSP Example

- Simple JSP example
  - Demonstrates
    - Fixed-template data (XHTML markup)
    - Creating a Java object (java.util.Date)
    - Automatic conversion of JSP expression to a String
  - First invocation of clock.jsp
    - Notice the delay while:
      - JSP container translates the JSP into a servlet
      - JSP container compiles the servlet
      - JSP container executes the servlet
    - Subsequent invocations should not experience the same delay

Using a JSP expression to insert the date and time in a Web page (Part 1).

Line 10

Line 30

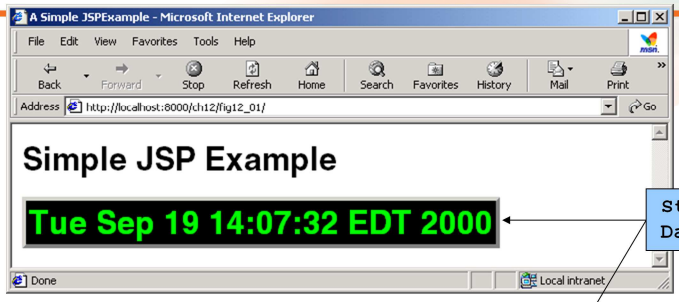
```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8
9     <head>
10       <meta http-equiv = "refresh" content = "60" />
11
12       <title>A Simple JSP Example</title>
13
14       <style type = "text/css">
15          .big { font-family: helvetica, arial, sans-serif;
16                 font-weight: bold;
17                 font-size: 2em; }
18       </style>
19    </head>
20
21    <body>
22       <p class = "big">Simple JSP Example</p>
23
24       <table style = "border: 6px outset;">
25          <tr>
26             <td style = "background-color: black;">
27                <p class = "big" style = "color: cyan;">
28
29                <!-- JSP expression to insert date/time -->
30                <%= new java.util.Date() %>
31
32                </p>
33             </td>
34          </tr>
35       </table>
36    </body>
37
38 </html>
```

`meta` element refreshes the Web page every **`60`** seconds

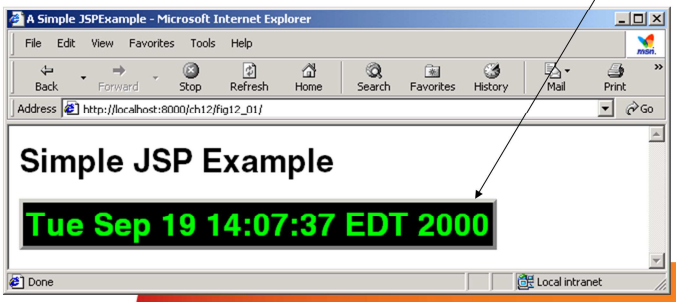
Creates **`Date`** object that is converted to a **`String`** implicitly and displayed in paragraph (**`p`**) element

Using a JSP expression to insert the date and time in a Web page (Part 2).

Program Output



`String` representation of `Date` object appears here.



12

Implicit Objects

- Provide access to many servlet capabilities within a JSP.
- Extend classes or implement interfaces
- Such objects can invoke public aspects of classes/interfaces
- All the implicit objects are divided by four variable scopes:
  - Application:
    - Objects owned by the container application
    - Any servlet or JSP can manipulate these objects
  - Page:
    - Objects that exist only in page in which they are defined
    - Each page has its own instance of these objects
  - Request:
    - Objects exist for duration of client request
    - Objects go out of scope when response sent to client
  - Session:
    - Objects exist for duration of client's browsing session
    - Objects go out of scope when client terminates session or when session timeout occurs

Implicit Objects (cont.)

Implicit Object	Description
Application Scope	
`application`	This `javax.servlet.ServletContext` object represents the container in which the JSP executes.
Page Scope	
`config`	This `javax.servlet.ServletConfig` object represents the JSP configuration options. As with servlets, configuration options can be specified in a Web application descriptor.
`exception`	This `java.lang.Throwable` object represents the exception that is passed to the JSP error page. This object is available only in a JSP error page.
`out`	This `javax.servlet.jsp.JspWriter` object writes text as part of the response to a request. This object is used implicitly with JSP expressions and actions that insert string content in a response.
`page`	This `java.lang.Object` object represents the `this` reference for the current JSP instance.
`pageContext`	This `javax.servlet.jsp.PageContext` object hides the implementation details of the underlying servlet and JSP container and provides JSP programmers with access to the implicit objects discussed in this table.

JSP implicit objects (part 1 of 2).

Implicit Objects (cont.)

Implicit Object	Description
`response`	This object represents the response to the client. The object normally is an instance of a class that implements **`HttpServletResponse`** (package **`javax.servlet.http`**). If a protocol other than HTTP is used, this object is an instance of a class that implements **`javax.servlet.ServletResponse`**.
Request Scope	
`request`	This object represents the client request. The object normally is an instance of a class that implements **`HttpServletRequest`** (package **`javax.servlet.http`**). If a protocol other than HTTP is used, this object is an instance of a subclass of **`javax.servlet.ServletRequest`**.
Session Scope	
`session`	This **`javax.servlet.http.HttpSession`** object represents the client session information if such a session has been created. This object is available only in pages that participate in a session.

JSP implicit objects (part 2 of 2).

Scripting

- Scripting
  - How JSP programmers can insert Java code and logic
  - Currently, JSP support scripting only with Java
- JSP scripting components
  - Scriptlets (delimited by <% and %>)
  - Comments (delimited by <%-- and --%>)
  - Expressions (delimited by <%= and %>)
  - Declarations (delimited by <%! and %>)
  - Escape sequences

Literal	Escape sequence	Description
<%	<\%	The character sequence <% normally indicates the beginning of a scriptlet. The <\% escape sequence places the literal characters <% in the response to the client.
%>	%\>	The character sequence %> normally indicates the end of a scriptlet. The %\> escape sequence places the literal characters %> in the response to the client.
' " \	\' \" \\	As with string literals in a Java program, the escape sequences for characters ', " and \ allow these characters to appear in attribute values. Remember that the literal text in a JSP becomes string literals in the servlet that represents the translated JSP.

Scripting – Practice 1

- Demonstrate basic scripting capabilities that Responding to get requests

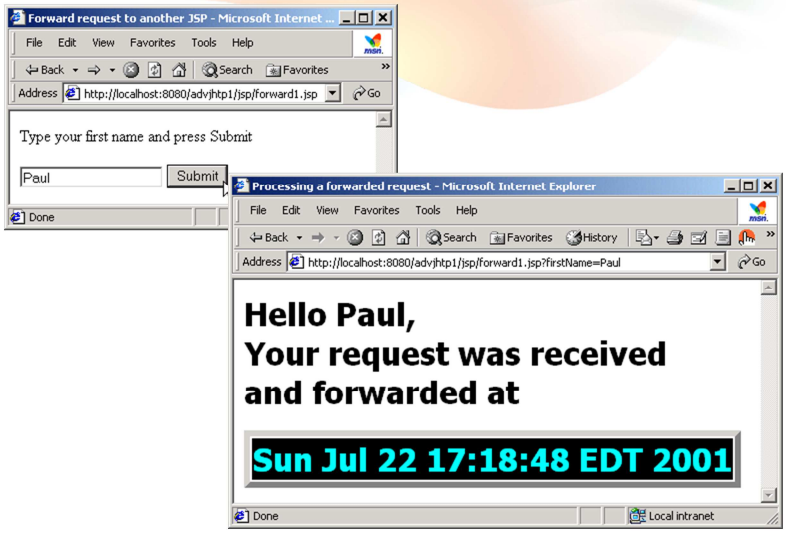
Standard Actions 1/2

- JSP standard actions
  - Provide access to common tasks performed in a JSP
    - Including content from other resources
    - Forwarding requests to other resources
    - Interacting with JavaBeans
  - JSP containers process actions at request time
  - Delimited by <jsp:action> and </jsp:action>

Standard Actions 2/2

Action	Description
<jsp:include>	Dynamically includes another resource in a JSP. As the JSP executes, the referenced resource is included and processed.
<jsp:forward>	Forwards request processing to another JSP, servlet or static page. This action terminates the current JSP's execution.
<jsp:plugin>	Allows a plug-in component to be added to a page in the form of a browser-specific **object** or **embed** HTML element. In the case of a Java applet, this action enables the downloading and installation of the *Java Plug-in*, if it is not already installed on the client computer.
<jsp:param>	Used with the **include**, **forward** and **plugin** actions to specify additional name/value pairs of information for use by these actions.
JavaBean Manipulation	
<jsp:useBean>	Specifies that the JSP uses a JavaBean instance. This action specifies the scope of the bean and assigns it an ID that scripting components can use to manipulate the bean.
<jsp:setProperty>	Sets a property in the specified JavaBean instance. A special feature of this action is automatic matching of request parameters to bean properties of the same name.
<jsp:getProperty>	Gets a property in the specified JavaBean instance and converts the result to a string for output in the response.

Standard Actions – Practice 2



JSP Directives

- Messages to JSP container
- Enable programmer to:
  - Specify page settings
  - Include content from other resources
  - Specify custom-tag libraries
- Delimited by <%@ and %>

Directive	Description
page	Defines page settings for the JSP container to process.
include	Causes the JSP container to perform a translation-time insertion of another resource's content. As the JSP is translated into a servlet and compiled, the referenced file replaces the **include** directive and is translated as if it were originally part of the JSP.
taglib	Allows programmers to include their own new tags in the form of *tag libraries*. These libraries can be used to encapsulate functionality and simplify the coding of a JSP.

Database Connectivity

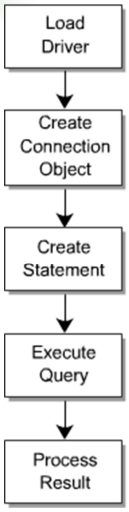
- Review
  - Provides a programming interface that is used to request a connection between the application and database
  - JDBC API executes SQL statements (in the Java code to retrieve data) and sends the results through a single API
- Five steps to work with database
  - Load driver: Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  - Create Connection object

    Connection **con**= DriverManager.getConnection(

    jdbc:odbc:**Datasource**, "**userid**", "**pwd**");
  - Create Statement: Statement stat = con.createStatement() ;
  - Execute Query

    String query = "Select * from table_name";

    ResultSet resultset = stat.executeQuery(query);
  - Process Results: `next()` method of the ResultSet object is used to process the results from the database



Database Connectivity – Sample

Importing Java.sql.* package

Loading the driver

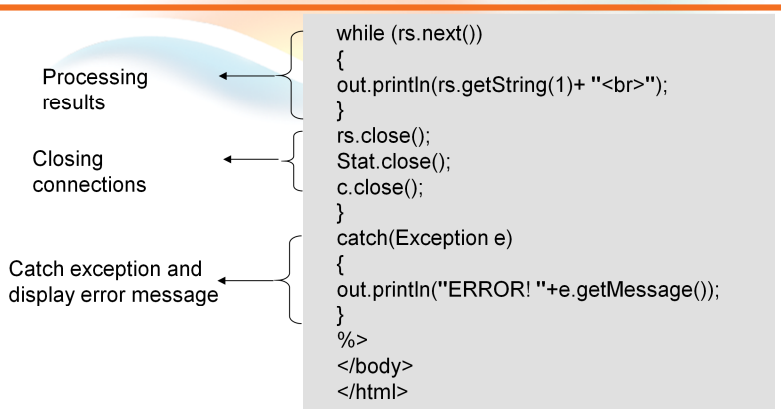
Creating Connection object

Creating Statement

Executing the query

```
<html>
<head>
<title>DB Test</title>
</head>
<body>
<%@ page language="java" import="java.sql.* " %>
<%
try
{
 out.println("loading driver...<br>");
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver ");
out.println("connecting...<br>");
Connection con=
DriverManager.getConnection(jdbc:odbc:Datasource, "userid ",
"pwd ")
out.println("querying database...<br>");
Statement stat = con.createStatement();
String query = "Select * from table_name ";
ResultSet rs = statement.executeQuery(query);
```

Database Connectivity – Sample



Session Tracking

- Maintains a session till the user is browsing the Web site
- Used in interactive Web applications to store the information of the user logged in to the Web site
- The information stored is used to identify the user sending a request to the Web server
- Session tracking helps to maintain the session information and keeps track of the multiple requests made by the client
- HTTP – stateless protocol
  - Does not support persistent information
- Track clients individually
  - Cookies
  - Session tracking
  - `hidden` type `input`
  - URL rewriting

Session Tracking - Cookies

- Cookies are text files stored on the user's computer containing the session Id of the user sent by the Web server
- The cookie is sent back to the Web server with every subsequent request made by the user in the same session
- The cookie includes a name, a single value and optional attributes
- Deleted automatically when they expire
- Servlet **`CookieServlet`**
  - Handles both `get` and `post` requests

Session Tracking – Cookies (cont.)

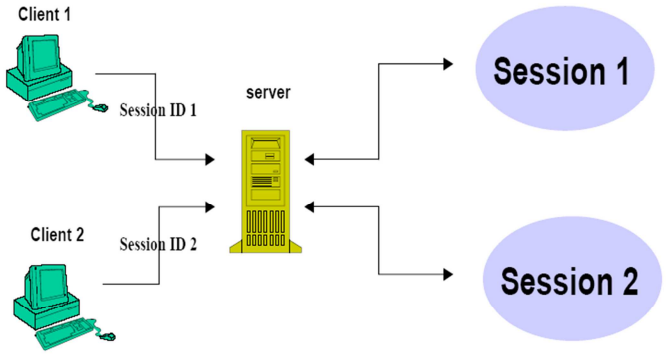
- Advantages
  - Remember user IDs and password.
  - To track visitors on a Web site for better service and new features.
  - Cookies enable efficient ad processing.
- Disadvantages
  - The size and number of cookies stored are limited.
  - Personal information is exposed to the other users.
  - Cookies fails to work if the security level is set too high in the Internet browser.

Session Tracking - How Do We Need HTTP State?

- Web applications need to track the users across a series of requests:
  - Online shopping (e.g. Order books)
  - Financial portfolio manager
  - Movie listings
- HTTP does not support directly
- Need a mechanism to maintain state about a series of requests from the same user (or originating from the same browser) over some period of time

Session Tracking with HttpSession

- The servlet API has a built-in support for session tracking
- Session objects live on the server
  - Each user has associated an HttpSession object—one user/session
  - HttpSession object operates like a hashtable



Session Tracking with HttpSession
Session Life Cycle

- The server assigns a unique ID to the session created for a particular user request.
- This session ID is passed to the client as a cookie or a hidden variable.
- The session is considered new until the client returns the session ID to the server through a cookie or as a part of the requested URL.
- A session exists on the server until it becomes invalid or the server is stopped.
- The HttpSession objects are used to store the session data in the current servlet context.

Session Tracking with HttpSession
Using Session Object

- Session object can be used to store and read data.
- The session object acts almost like a bulletin board from where the objects can be written or read

Session Tracking with HttpSession
Using Session Object (cont.)

- The request() method requests for the session object.

```
…
…
// Obtain a session object
HttpSession session = request.getSession(true);
//Add an item to the session
Integer sessionData = new Integer (100);
Session.putValue("IntValue", sessionData);
…
…
```

Obtains a session object

Adds item to the session object

Session Tracking with HttpSession
Using Session Object (cont.)

- The session value can be read and cast to the appropriate object type.

```
…
// Obtain a session object
HttpSession session =
request.getSession(true);
// Read the session data and cast it to
the appropriate object type
Integer sessionInt = (Integer)
session.getValue("session");
int count = sessionInt.intValue();
…
…
```

Obtains a session object

Reads the session value

Casts the session value to appropriate datatype

Session Tracking with HttpSession
Using Session Object (cont.)

- The session can be invalidated using the `invalidate()` method of the HttpSession object.

```
<%
   String sessionval=(String)session.getAttribute("userid"));
   if(sessionval == null)
   {
      session.setAttribute("userid",sessionval);
      out.println(session.getAttribute("userid"));
   }
   else
  {
      out.println("User Session already created");
   }
%>
<b>click this link to
<a href="<%=session.removeAttribute("userid")%>">remove
session attribute</a></b><br/>
<b>click this link to <a href="<%=session.invalidate()%>">
invalidate the session</a></b><br/>
```

Accepts **userid**

If **sessionval** is null, the value of sessionval is set to **userid**.

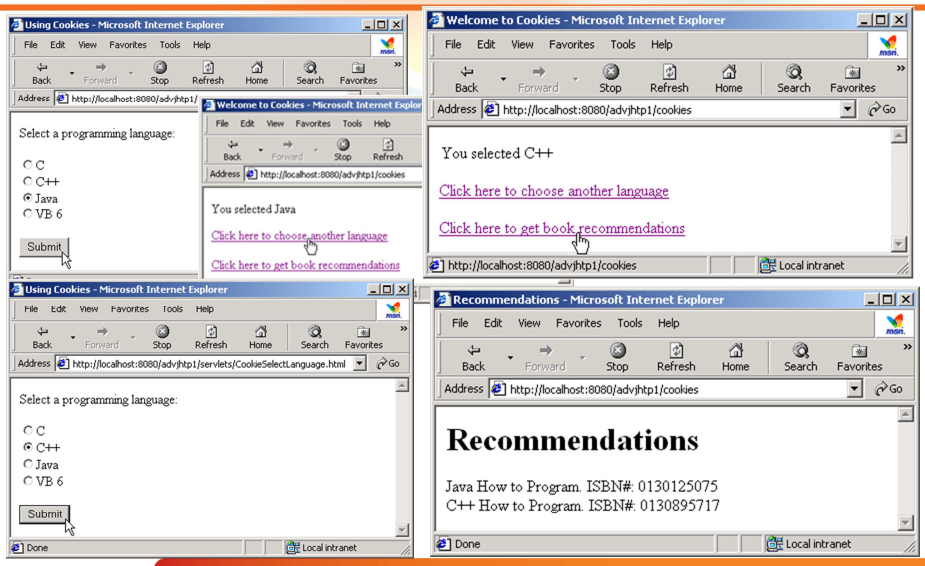
Removes the session

Invalidates the session

Session Tracking with HttpSession
Using Session Object (cont.)

- To get a user's existing or new session object:
  - HttpSession session = request.getSession(true);
  - "true" means the server should create a new session object if necessary
- To store or retrieve an object in the session:
  - Stores values: setAttribute("cartItem", cart);
  - Retrieves values: getAttribute("cartItem");

Session Tracking - Practice 3

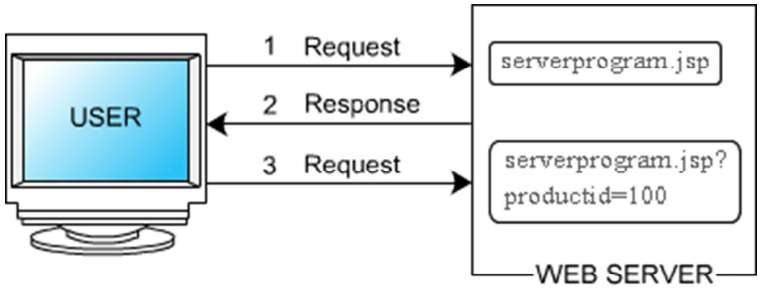


Session Tracking - URL Rewriting

- JSP hides the details of a cookie-based session tracking and supports the URL rewriting mechanism
- URL Rewriting works with Web browsers that do not support cookies or the cookies that are disabled on a Web browser
- Each URL that references the Web browser is returned to the user and contains additional information

Session Tracking - URL Rewriting (cont.)

The session ID is encoded in the URLs that are created by the JSP pages



Session Tracking - URL Rewriting (cont.)

```
<b>Search results for books</b>
<form method="post" action="serverprogram.jsp">
<input type="checkbox" name="productID" value="100">CD
MP3 Converter Kit For Your CAR<br>
<input type="checkbox" name="productID"
value="101">Front Loading Car MP3/CD Player With Anti
Shock Memory and FM<br>
<input type="checkbox" name="productID"
value="102">CAR/Home DVD/VCD/MP3 Playerwith anti
shock for Indian Roads<br>
<input type="submit" name="Submit" value="Add to
Cart"><br>
</form>
```

URL of server side program

Provides check box for different products

Submits the user input to URL

Session Tracking - URL Rewriting (cont.)

```
<b>Search results for books</b>
<form method="post"
action="serverprogram.jsp?productID=102">
<input type="checkbox" name="productID"
value="150">DVD Player with built in Amplifier
<br>
<input type="checkbox" name="productID"
value="160">Ultra Slim DVD Player Multi
Region 5.1 Digital<br>
<input type="submit" name="Submit" value =
"Add to Cart">
<br>
</form>
```

URL for server side program after the user selects a product and goes to another page

Provides check box for different products

Submits input to the URL

Session Tracking - URL Rewriting (cont.)

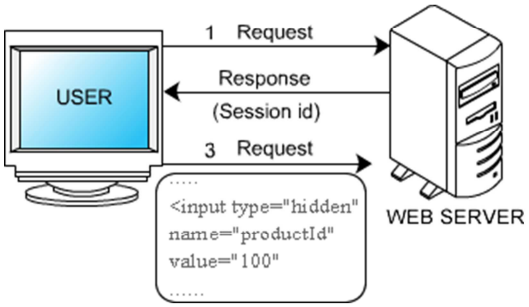
- Disadvantages
  - Server side processing is tedious.
  - Every URL that is returned to the user should have additional information appended to it.
  - If the user leaves the session and opens the Web page using a link or bookmark then the session information is lost .

Session Tracking – Hidden Form Fields

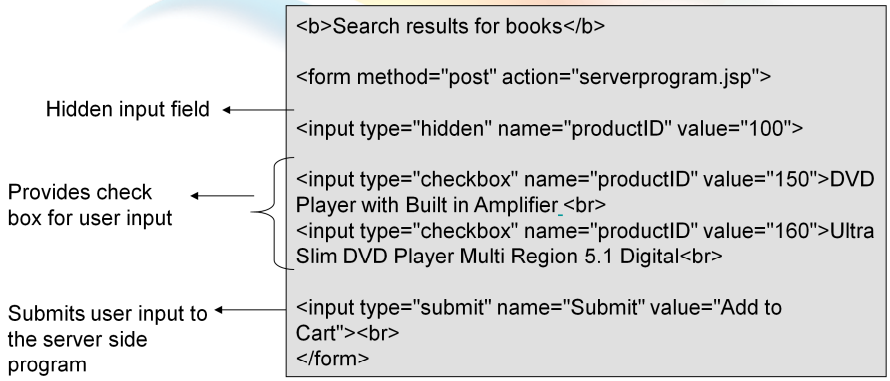
- Information from the Web browser is returned to the Web server in the form of HTTP parameters
- Utilizes the hidden fields in an HTML page
- Hidden fields in the form are used to send the information to the Web browser
- Stores information about a session
- Helps to carry the information from one HTML page to another

Session Tracking – Hidden Form Fields (cont.)

- When the user visits the next page, the server side program reads all the parameters that a user passes in the previous form



Session Tracking – Hidden Form Fields (cont.)



Session Tracking – Hidden Form Fields (cont.)

- The advantages of hidden form fields are:
  - Simplest way to implement session tracking
  - Displays nothing on the HTML page but can be used to hold any kind of data
  - Helps to maintain a connection between two pages
- The disadvantage of hidden form fields is that this method of session tracking displays sensitive information to the user.
- The information includes the data passed around to maintain a session.

Exception Handling

- Exceptions are errors that can occur in a JSP page
- The JSP page traps and handles request time errors
- Unhandled exceptions are forwarded to the error page
- Syntax

```
<%@ page errorPage="errorpage.jsp" %>
```

- Set the `isErrorPage` attribute of `page` directive to true, to make a JSP page an error handler
- Syntax

```
<%@ page isErrorPage="true" %>
```

Exception Handling - Cont…

- **Translation time** - Occurs when the JSP source file is converted to servlets class file. The JSP engine handles translation time errors.
- **Request time** - Occurs during the processing of the request. Request time errors are the runtime errors that throw exceptions.

```
<html>
<body>
<%@ page isErrorPage="true" %>
Detected Error: <br>
<%= exception.getMessage() %>
</body>
</html>
```

Makes JSP page an error handler

Returns error message

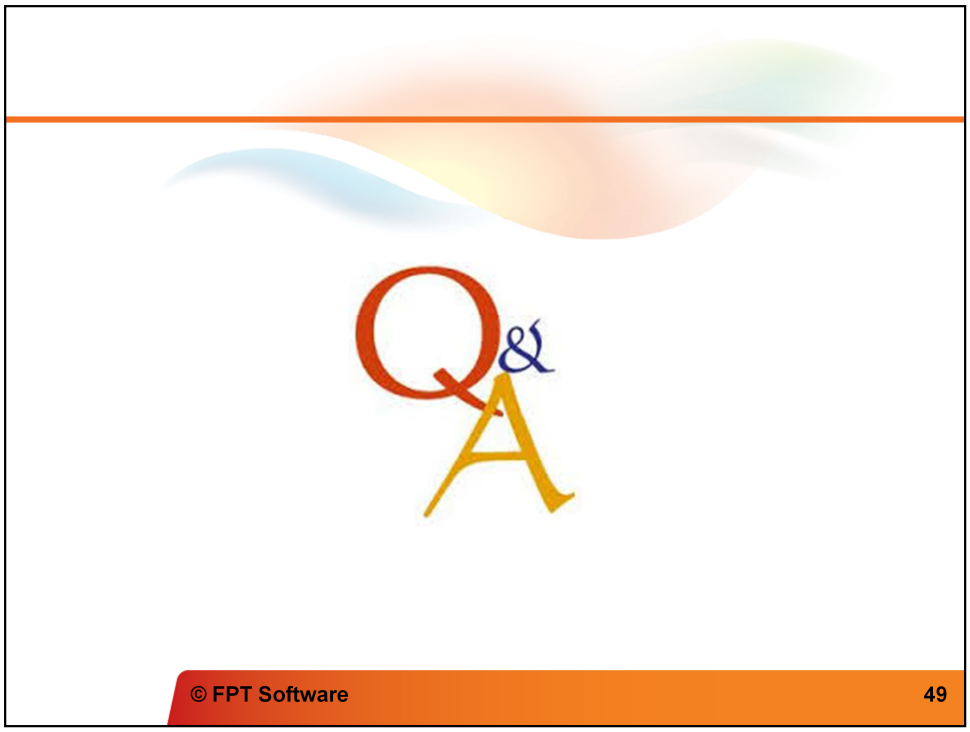
Code Snippet to create an error page

Exception Handling - Cont…

```
<%@ page errorPage="errorpage.jsp" %>
<%
if (request.getParameter("param ").equals("value "))
{
// code
}
//The test above will throw a NullPointerException if param is
// not part of the query string. A better implementation is:
if ("value".equals(request.getParameter("param ")))
{
 // code
}
%>
```

Forwards the unhandled exception to errorpage.jsp

Code Snippet to transfer control to the error page


Q&A
© FPT Software
49