

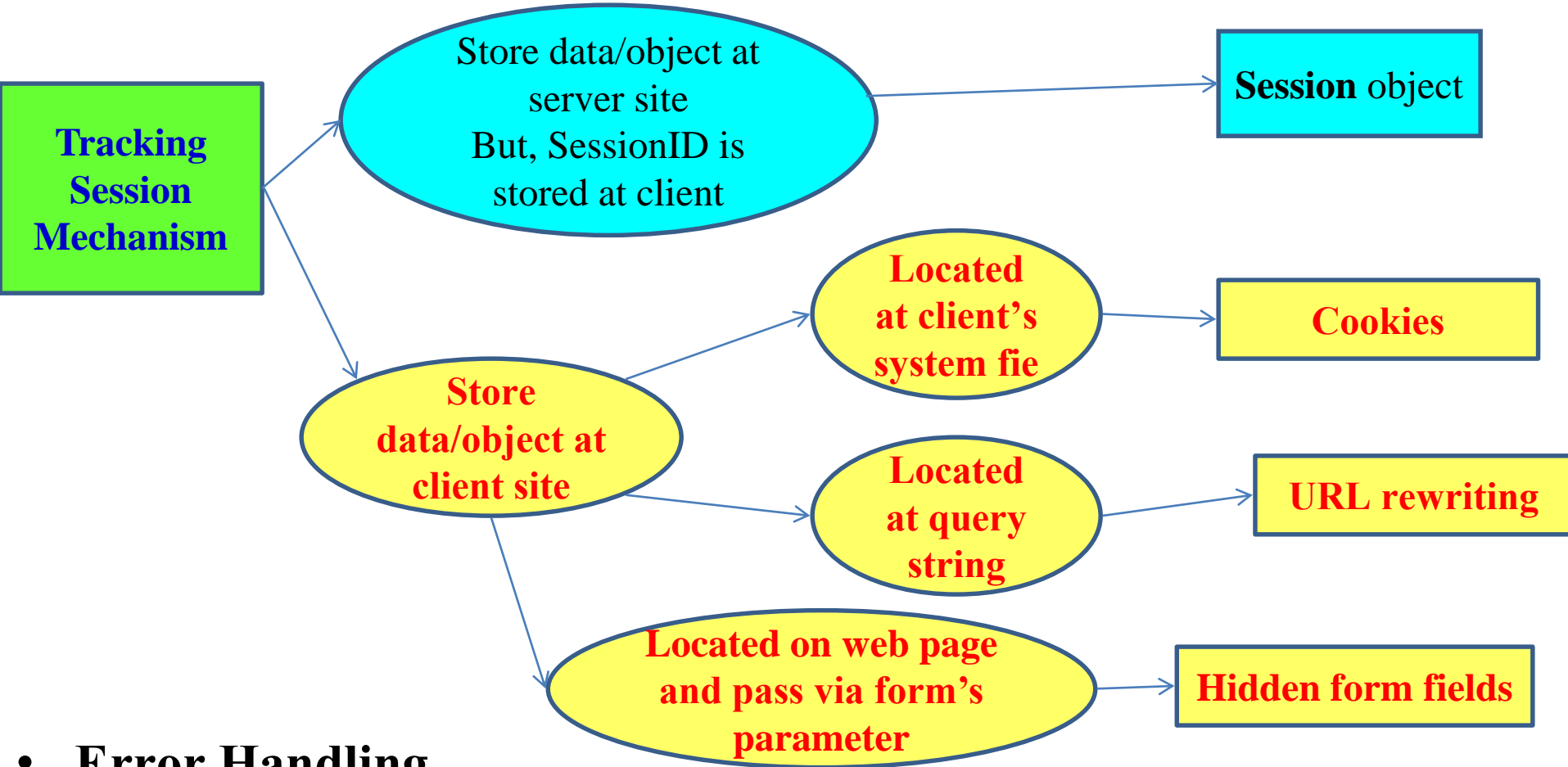
JavaBeans

JSP Standard Actions
Dispatching Mechanisms
Expression Language
JSPs in XML

Review

- **Session Tracking Mechanism**

- Client must be stored the value that transfer to server in each its request



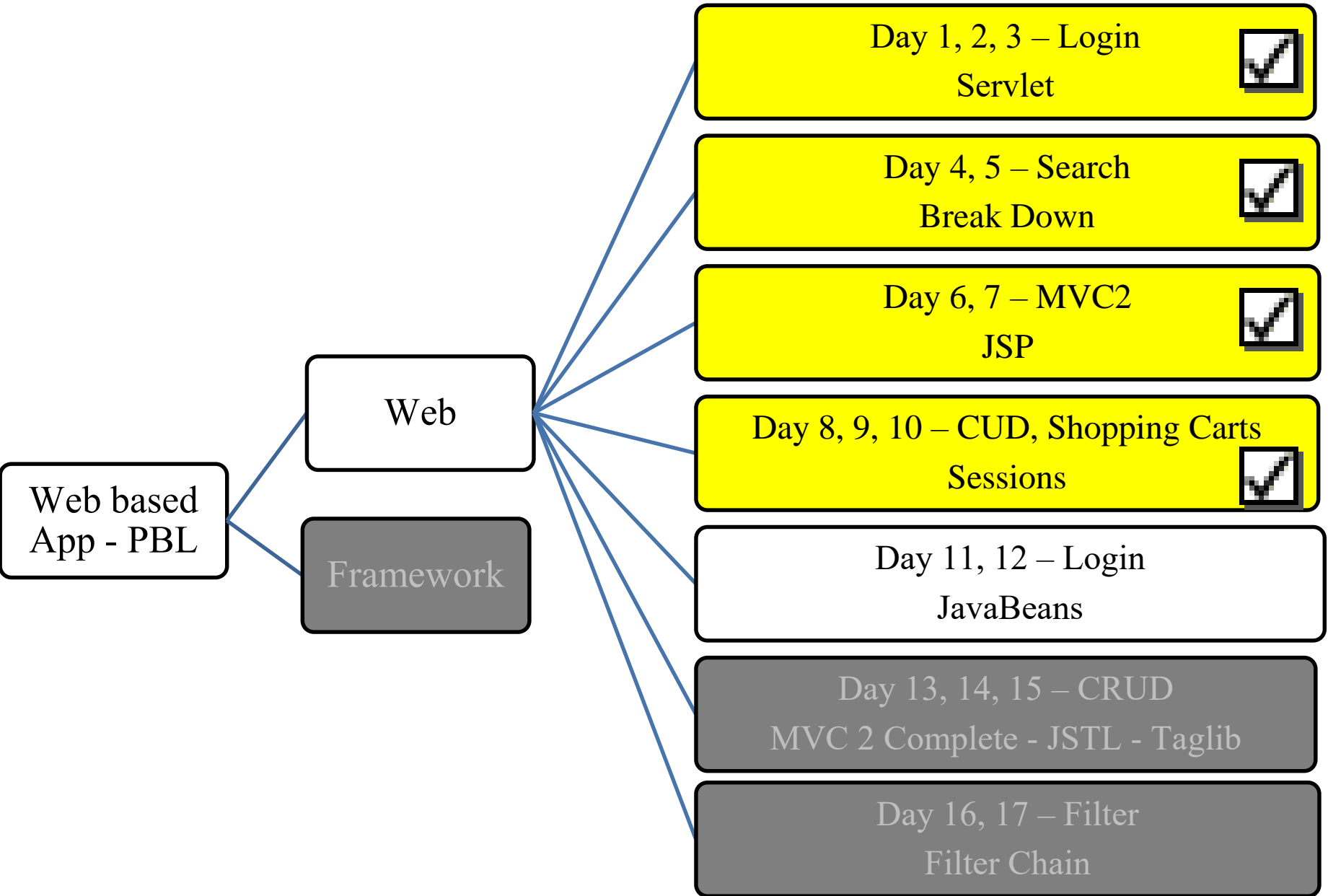
- **Error Handling**

- Reporting Error: create the friendly UI to user when the system's errors occur.
- Logging Error: store the errors (users or/and app) to the file to improve the application and get users' behaviors

Objectives

- **How to build the Web Application using MVC1?**
 - Standard Actions
 - Dispatching Mechanisms
- **How to remove the scripting element (Java Code) in the jsp (view)?**
 - Expression Language

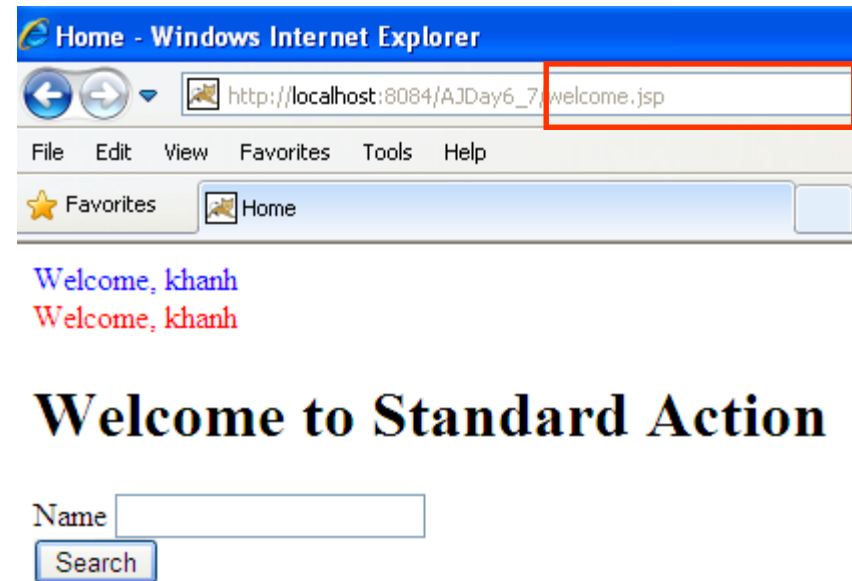
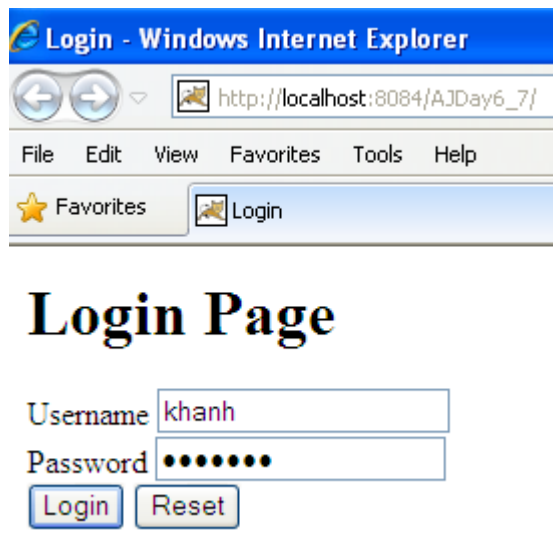
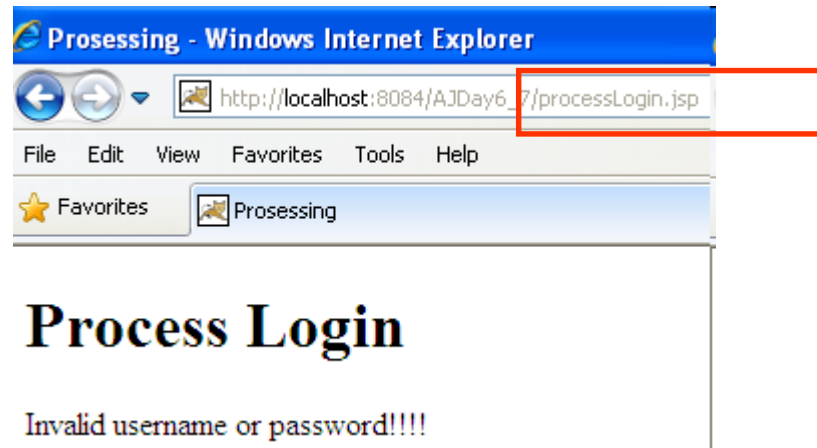
Objectives



MVC 1

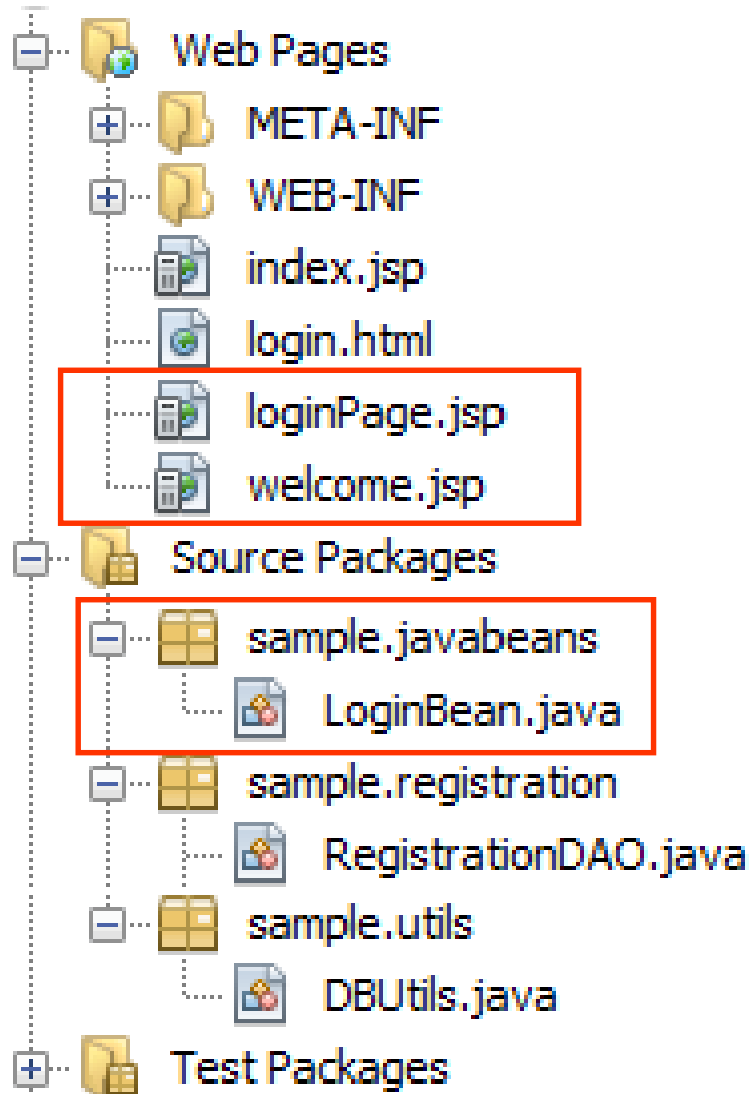
Requirements

- Building the web application with authentication functions
 - The GUI is shown as



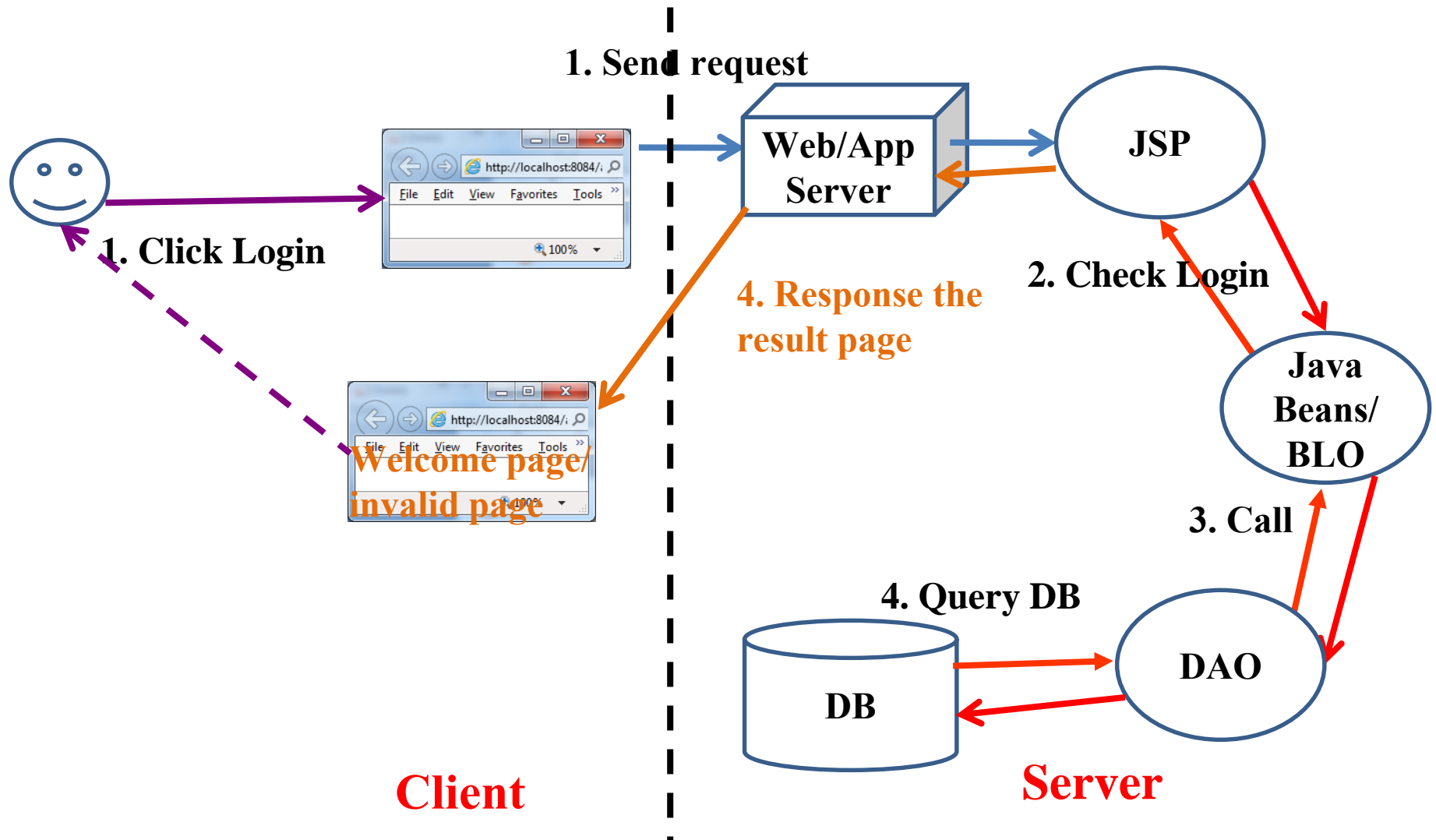
MVC 1

Expectation



MVC1

Interactive Server Model



JSP Standard Actions

Java Beans

- JavaBeans technology is the **component architecture** for the Java 2 Platform, Standard Edition (J2SE).
- JavaBeans technology is based on the **JavaBeans specification**.
- Components (JavaBeans) are **reusable** software programs that you can develop and **assemble easily** to create sophisticated applications.
 - Are reusable components which define the interactivity of Java objects
 - Are reusable software components that work **independently** on a workstation and with a set of other distributed components.
- **Encapsulate state and behavior** of software component
- The **different** point between Java Bean and Java class
 - Java Bean **implemented from Serializable**
 - The process of **create a copy** of an object **suitable** for **passing to another object/ package classes into streams** of bytes that **are transmitted through networks** or saved to the **disk**

JSP Standard Actions

Java Beans

- In order to function as a JavaBean class, an object class **must obey certain conventions** about **method naming, construction, and behavior**.
 - These conventions make it possible to have tools that can **use, reuse, replace, and connect** JavaBeans.
- A Bean a **simple Java Class** that follows certain **coding conventions**
 - Bean class should **always use a package name**
 - Bean class **must have a public no-argument constructor**
 - The **properties of bean** (persistence) is **not declared “public”**. They are **accessed through getter and setter methods**.
 - Public **getter** method is used to **retrieve the properties of a bean class**
 - Public **setter** method is used to **set the properties of a bean class**
- **Notes:**
 - The first character of each property should **name in lower case** then the **accessor** methods are used along with property name **with the first character of each word in upper case** (Ex: length – getLength và setLength)
 - The dataType of properties is **boolean** then the getter method is **isXxx** instead of **getXxx**

JSP Standard Actions

Java Beans

- A **JSP** page **accesses Java Beans using tag action and gets the result of processing without how to JavaBeans implementation and process**
- Java Bean tags are combined with JSP elements
- Java Bean tags are **translated into single java Servlet classes on the server**
- JSP technology directly supports using JavaBeans components with JSP language elements

JSP Standard Actions

Standard Actions

- An alternative to inserting java code into designed presentation page
- Are performed when a browser requests for a JSP page
- Are used for
 - Forwarding requests and performing includes in page
 - Embedding the appropriate HTML on pages
 - Interacting between pages and JavaBeans
 - Providing additional functionality to tag libraries
- Syntax: **<prefix:actionName att="...">...</prefix:actionName>**
- Some properties
 - It use **"jsp"** prefix
 - The attributes are **case sensitive**
 - Value in the attributes must be **enclosed in double quotes**
 - Standard actions can be either **an empty or a container tag**
- A JSP provides **03 tags** that use **interact** with JavaBean
 - **jsp:useBean, jsp:setProperty, jsp:getProperty**

JSP Standard Actions

The `<jsp:useBean>` tag

- Is used to **locate or instantiate** a JavaBeans component
- **First tries to locate** an instance of the Bean **otherwise it instantiates** the Bean from a class
- To locate or instantiate the Bean, the `<jsp:useBean>` follows the following steps:
 - **Attempts to locate** a Bean (*means attribute containing object*) within the scope
 - **Defines an object reference variable with the name**
 - Stores a reference to it in the variable, if it retrieves the Bean
 - **Instantiates** it from the specified class, **if it cannot retrieve the Bean**
- **Syntax**
`<jsp:useBean id="<identifier>" class="<class name>" [scope = "scope name"]/>`
 - **id**: is used to name the **attribute name**, to **refer** to the Bean **instance in specify scope**
 - **class**: a **class name implementing** a Bean processing.
 - **scope**: the **lifespan** of a bean (sharable). Default value is **page**

JSP Standard Actions

The `<jsp:useBean>` tag

- Use scriptlet element to declare Java Bean:

`<%`

```
className id = (className) scope.getAttribute("identifier");
if (id == null) {
    id = new className();
    scope.setAttribute ("identifier", id);
}
```

`%>`

- **Ex**

- `<jsp:useBean id="book1" class="store.book"/>`

- **similar to**

`<%`

```
store.book book1 = (store.book) pageContext.getAttribute("book1");
if (book1 == null) {
    book1 = new store.book();
    pageContext.setAttribute("book1", book1);
}
```

`%>`

JSP Standard Actions

The `<jsp:useBean>` tag

- **Casting**

```
<jsp:useBean id="<identifier>" class="<class name>" type = "<dataType>"  
[scope = "scope type"]/>
```

- **type:** Java – DataType

- **JSP Scriptlet element**

```
<%
```

```
    dataType id = (dataType) scope.getAttribute("identifier");  
    if (id == null) {  
        id = (dataType) (new className());  
        scope.setAttribute ("identifier", id);  
    }
```

```
%>
```

- **Ex**

- `<jsp:useBean id="book1" class="store.book" type="library.magazine" />`

- **Similar to**

```
<% library.magazine book1 = (library.magazine) pageContext.getAttribute("book1");  
    if (book1 == null) {  
        book1 = (library.magazine) (new store.book());  
        pageContext.setAttribute("book1", book1);  
    } %>
```

JSP Standard Actions

The `<jsp:useBean>` tag

- **Other syntax**

`<jsp:useBean ...> statement </jsp:useBean>`

- **Ex**

```
<jsp:useBean id="count" class="ABean.AccessBean" scope="application">  
  <jsp:setProperty name="count" property="firstPage" value="ATest.jsp" />  
</jsp:useBean>
```

- **Notes:**

- If using some specified symbol in command, the symbol “\” should put such as ‘(\’); “(\”), \ (\\), % (\%), ...
 - The JSP scriptlets use **id as a variable**.

JSP Standard Actions

The `<jsp:getProperty>` tag

- **Retrieves a bean property value** using the **getter methods** and **displays the output** in a JSP page
- **Before** using `<jsp:getProperty>` you **must create or locate a bean** with `<jsp:useBean>`
- **Drawback**
 - Fails to retrieve the values of an indexed property
 - Fails to directly access enterprise beans components
- **Syntax**
`<jsp:getProperty name="<identifier>" property="<attr name>" />`
 - **name**: the identifier declared in `jsp:useBean`
 - **property**: the property name is implemented in Java Bean
- **Use scriptlet command**
`<%= <identifier>.getXxx() %>`
- **Ex**
`<jsp:getProperty name="book1" property="title"/>`
Similar to `<%= book1.getTitle() %>`

JSP Standard Actions

The `<jsp:setProperty>` tag

- Sets the value of the properties in a bean, using the bean's setter methods
- Before using `<jsp:setProperty>` you must create or locate a bean with `<jsp:useBean>`
- Syntax

`<jsp:setProperty name="<identifiers>" property="<attr name>" value="<const/ expression>" />`

- Use scriptlet command:

```
<%  
    id.setXxx(<value>);  
    scope.setAttribute("identifier", id);  
%>
```

- **Ex:** `<jsp:setProperty name="book1" property="title" value="JSP Book" />`
Similar to `<% book1.setTitle("JSP Book");
 pageContext.setAttribute("book1", book1);
 %>`

JSP Standard Actions

The `<jsp:setProperty>` tag

- **Assign a expression to action setProperty**
 - `String sMsg = request.getParameter("sms");`
 - `<jsp:setProperty name="msg" property="message" value="<%= sMsg %>" />`
- **Use param properties in setProperty:** receive an inputted value from a request (from other JSP page, application, or URL)
 - `<jsp:setProperty name="msg" property="message" param="message" />`
- **To set values to whole bean properties,** the symbol “*” is assigned to setProperty

`<jsp:setProperty name="msg" property="*" />`

- **Notes:**
 - The Bean Action is **executed only** if all of **properties** is **assigned** values **because** the engine **does not assign a null value with lacked properties**.
 - In some web server, the **exceptions** are **thrown** if a **property** value is **assigned** to **double**
 - The **converted mechanism** does **not** ensure a **valid value** for **property** (instead of manual casting)
 - The **parameters name** should **similar** to **properties name** implemented in **Bean**

JSP Standard Actions

Steps in design Web Application
following MVC patterns

- **Step 1**
 - create **View**
- **Step 2**
 - **Depends on View, determine some persistence controls** that are implemented as the Java Bean (Model) 's properties
 - **Implement** the Java Bean with **accessor methods** and **some utilitie methods** that are used to query the properties of Java Bean
 - The methods are designed following the OO standard
- **Step 3**
 - **Create servlet (Controller)/JSP page combining View & Model**

MVC1

Requirements

Login - Windows Internet Explorer

http://localhost:8084/AJDay6_7/

File Edit View Favorites Tools Help

★ Favorites Login

Login Page

Username

Password

Processing - Windows Internet Explorer

http://localhost:8084/AJDay6_7/processLogin.jsp

File Edit View Favorites Tools Help

★ Favorites Processing

Process Login

Invalid username or password!!!!

Home - Windows Internet Explorer

http://localhost:8084/AJDay6_7/welcome.jsp

File Edit View Favorites Tools Help

★ Favorites Home

Welcome, khanh
Welcome, khanh

Welcome to Standard Action

Name

JSP Standard Actions

Scope of JavaBeans

- **page**
 - Accessible for **current page**. The life span till the current page displayed
 - The information is **stored in pageContext** (get values through the `getAttribute` method) – **default scope**
- **application**
 - Current and any successive request from the same Web Application. Global to all JSP and servlet.
 - The information is **stored in ServletContext**.
 - The life span is Application
- **session**
 - The information is **stored in HttpSession combined current request** (get values through the `getValues` method of Session interface (Implicit Object)).
 - The life span till the session destroyed, time out, or Web browser closed.
- **request**
 - Accessible for **current request** (get values through the `getAttribute` methods)
 - The information is **stored in ServletRequest**
 - The life Span till the request processed and the response is sent back to the Web browser

JSP Standard Actions

Scope of JavaBeans

- **Notes:**
 - Using JavaBeans with Session or Application scope, a JSP page **must declare a tag action `jsp:useBean` with a same id and class name.**
 - JSP, JSP/ Servlet **engine defines scope** in executing Web application, **if the instance bean existed**, the **engine does not instantatiate new one** (the tag `jsp:useBean` is **omited**) to execute Bean's methods. **Otherwise, the engine create a new instance.**

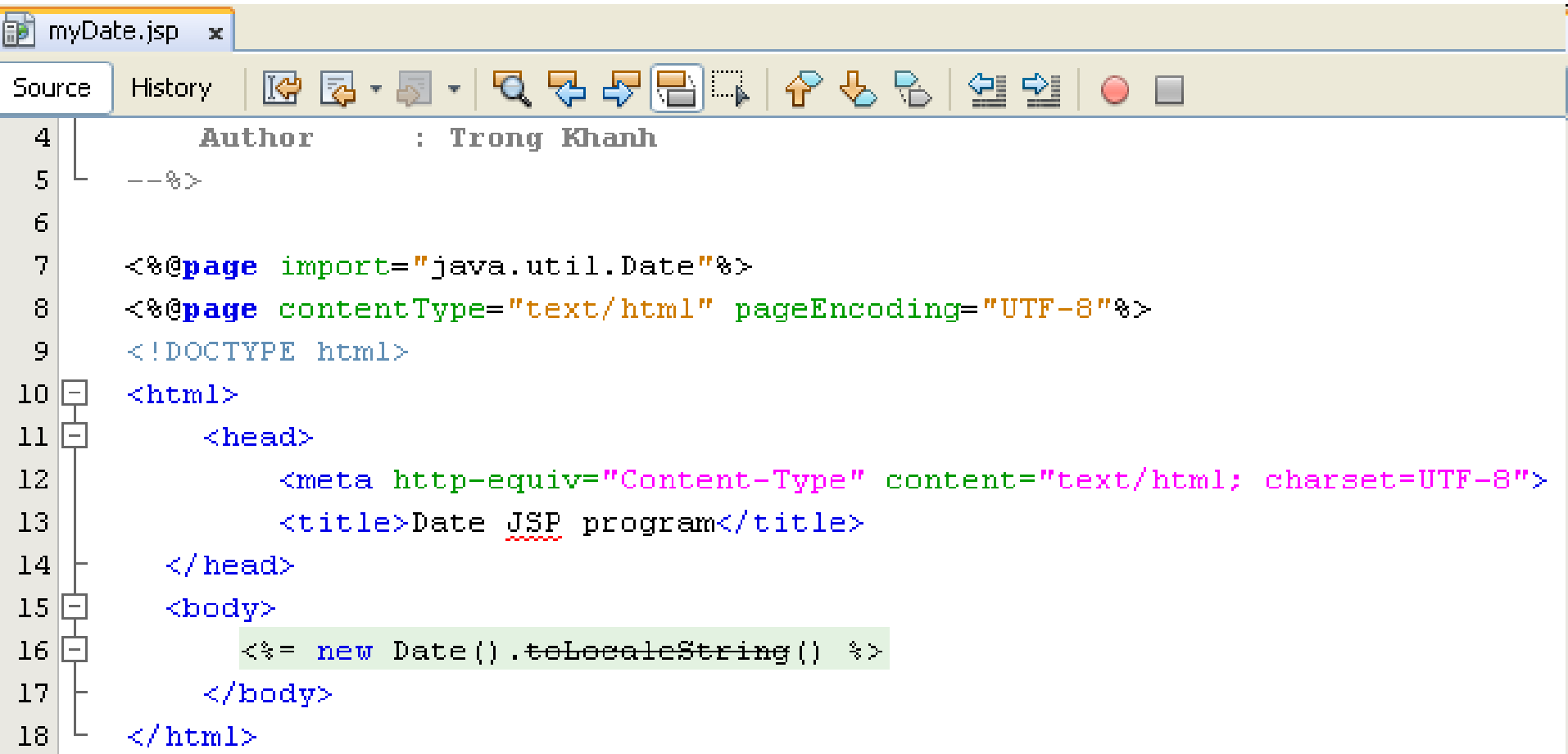
Dispatching Mechanisms

The `<jsp:include>` tag

- Can be used to **include** the **response** from **another file** within the **JSP** page **output**.
- To **incorporate** the content or **insert** a file from another page into current page
- **Syntax**
`<jsp:include page=“...” flush=“true” />`
- The file whose response should be included **has to reside somewhere** in your web application **but doesn't have to be present until the page is actually requested**

Dispatching Mechanisms

The <jsp:include> tag – Example

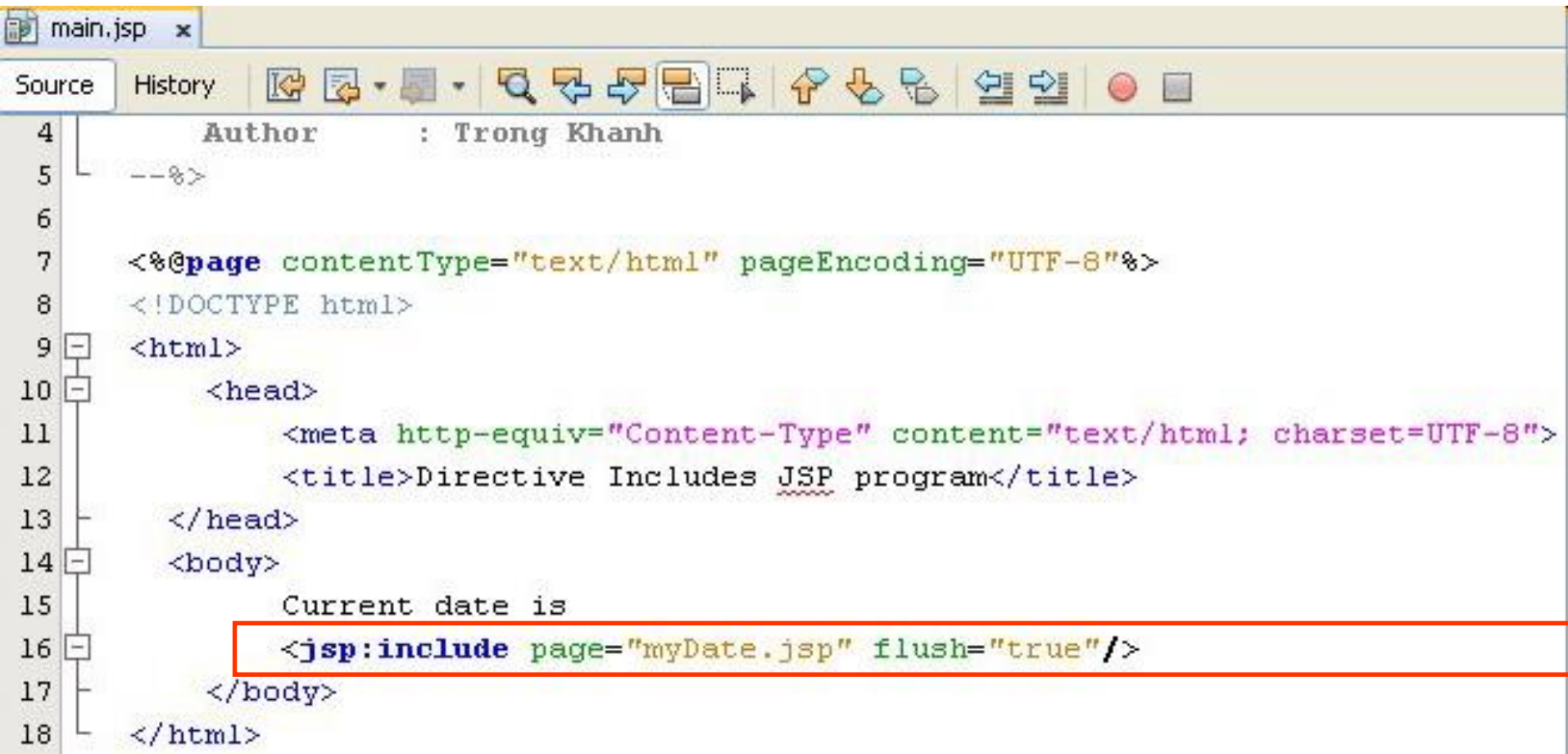


```

4      Author      : Trong Khanh
5      --%>
6
7      <%@page import="java.util.Date"%>
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>
9      <!DOCTYPE html>
10     <html>
11     <head>
12         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13         <title>Date JSP program</title>
14     </head>
15     <body>
16         <%= new Date().toLocaleString() %>
17     </body>
18 </html>
  
```


Dispatching Mechanisms

The <jsp:include> tag – Example

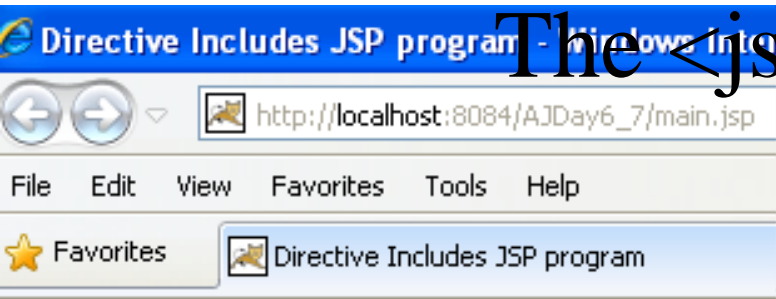


```

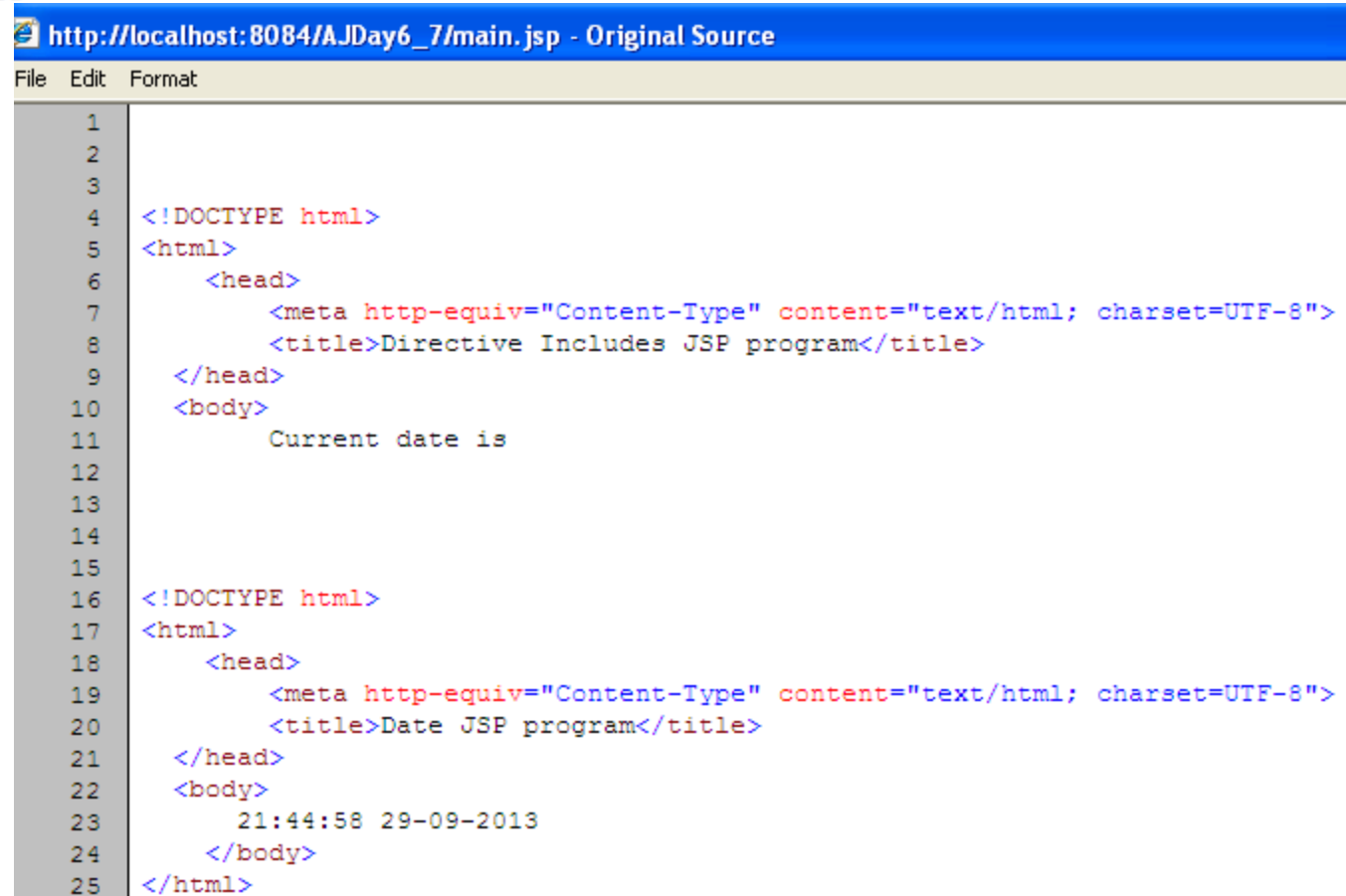
4      Author      : Trong Khanh
5      --%>
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE html>
9      <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Directive Includes JSP program</title>
13     </head>
14     <body>
15         Current date is
16         <jsp:include page="myDate.jsp" flush="true"/>
17     </body>
18 </html>
  
```

Dispatching Mechanisms

The `<jsp:include>` tag – Example



Current date is 21:44:58 29-09-2013



Dispatching Mechanisms

The <jsp:include> tag – Example

C:\Documents and Settings\Trong Khanh\Application Data\NetBeans\7.2.1\apache-tomcat-7.0.27.0_base\work\Catalina\localhost\AJDay6_7\org\apache\jsp

main_jsp CLASS File 5 KB

main_jsp Java Source File 4 KB

myDate_jsp CLASS File 5 KB

myDate_jsp Java Source File 4 KB

main_jsp.java x

Source History

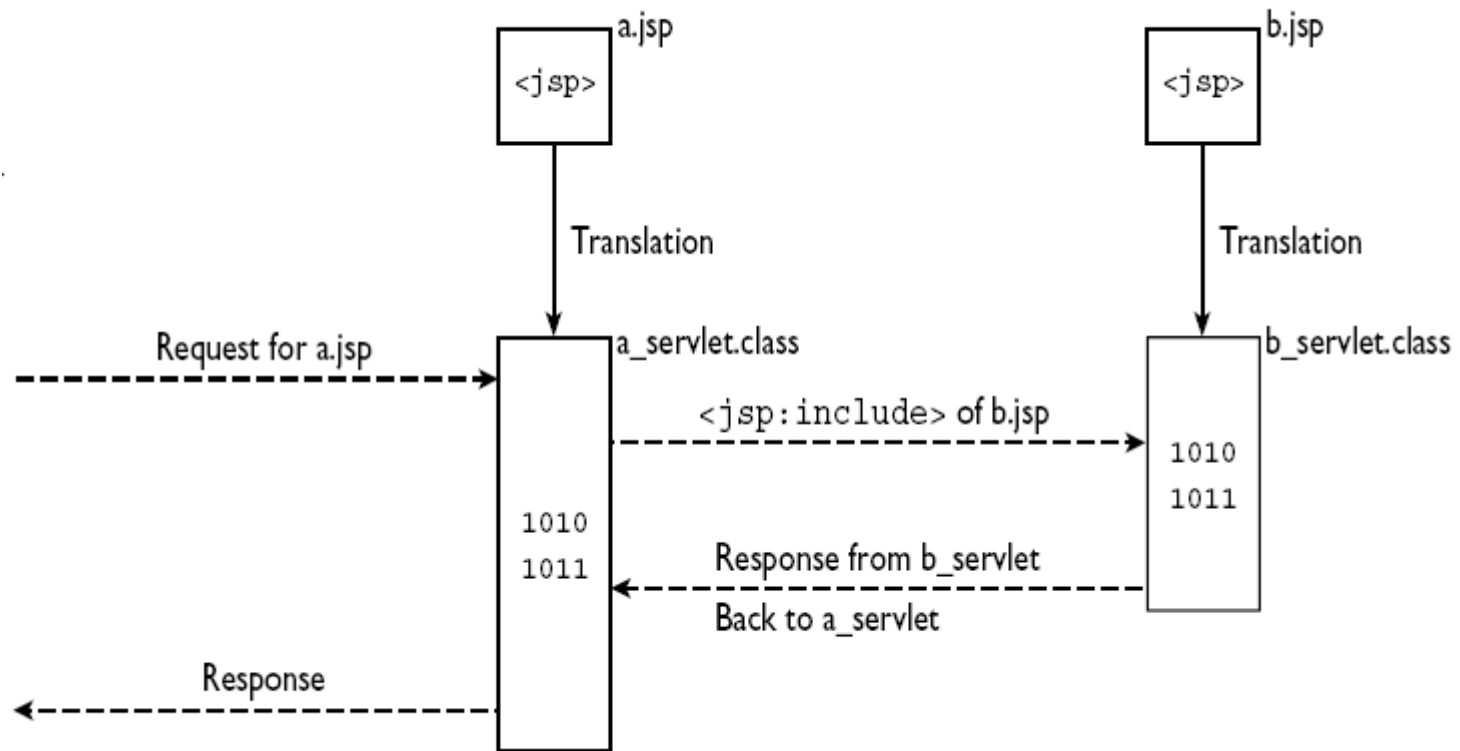
```

64 out.write("\n");
65 out.write("<!DOCTYPE html>\n");
66 out.write("<html>\n");
67 out.write("    <head>\n");
68 out.write("        <meta http-equiv=\"Content-Type\" o
69 out.write("        <title>Directive Includes JSP progr
70 out.write("    </head>\n");
71 out.write("    <body>\n");
72 out.write("        \tCurrent date is\n");
73 out.write("    ");
74 org.apache.jasper.runtime.JspRuntimeLibrary.include(
75     request, response, "myDate.jsp", out, true);
76 out.write("\n");
77 out.write("    </body>\n");
78 out.write("</html>\n");

```

Dispatching Mechanisms

`<jsp:include>` vs. `<%@ include ...%>`

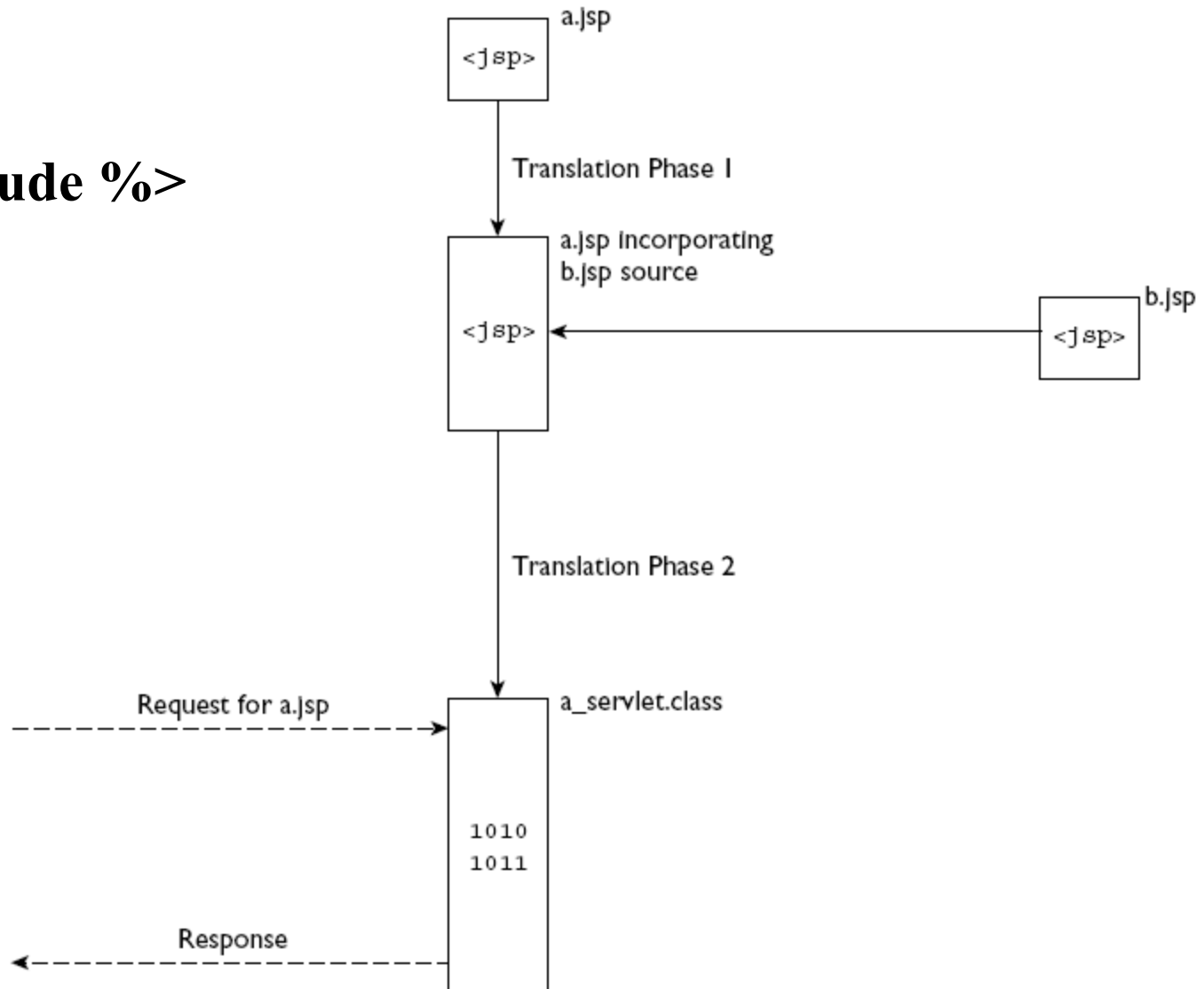


`<jsp:include>`

Dispatching Mechanisms

`<jsp:include>` vs. `<%@ include ... %>`

`<%@ include %>`



Dispatching Mechanisms

The `<jsp:forward>` tag

- **Forwards processing to another resource** within the web application
- To permanently **transfer control** from a **JSP page** to **another location** on the local server
- Is used to **redirect** the **request object** containing the client **request** from one **JSP file** to another file. The **target** file can be an **HTML** file, another **JSP** file or a **servlet**
- The code after the `<jsp:forward>` element is not processed
- **Syntax**

`<jsp:forward page="..." />` or

`<jsp:forward page="..." >`

`<jsp:param name="..." value="..." />`

`</jsp:forward>`

Dispatching Mechanisms

The <jsp:param> tag

- Pass **one or more name and value pairs** as parameters to an **included or forwarded** resource like a JSP page, servlet or other resource that can process the parameter (such as jsp:forward and jsp:include)
- **Syntax**
<jsp:param name="..." value="{parValue|<%=exp%>}" />

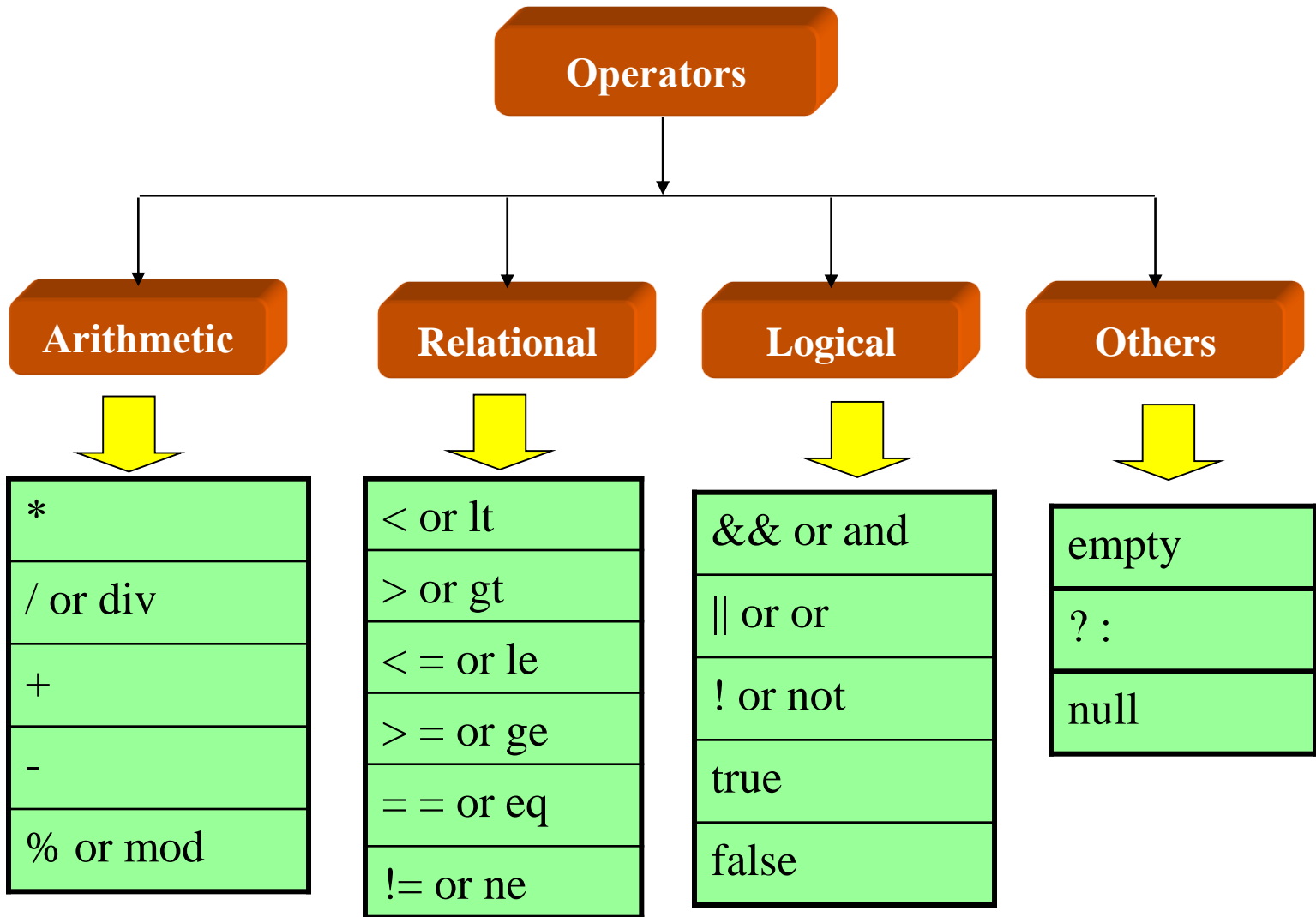
Expression Languages

EL Language Basics

- **New feature** of JSP 2.0
- Allows JSP developers to **access and manipulating java objects** via a **compact, easy-to-use shorthand style** (similar to JavaScript)
- It can **handle both expressions and literals**
- Can be used to display the **generated dynamic content** in a table on the web page and can also be used in HTML tags
- **Developed by two groups**
 - JSP Standard Tag Library expert group
 - JSP 2.0 expert group
- **Syntax: `${EL Expression}`**
- **JSP EL expressions are used in**
 - **Static** text
 - **Standard** and **Custom** tags
- **Advantages**
 - **Clear** syntax
 - **Simple & robust**
 - **Easy access** application **data stored** (in **JavaBeans**)

Expression Languages

EL Operators



Expression Languages

EL Operators – Example

Source History

math.jsp x

Author : Trong Khanh

```

4
5 --%>
6
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"%>
12 <title>Math</title>
13 </head>
14 <body>
15 <h1>EL Expression</h1>
16 <table border="1">
17 <tr>
18 <td>Concept</td>
19 <td>Expression</td>
20 <td>Result</td>
21 </tr>
22 <tr>
23 <td>Literal</td>
24 <td>${"10"}</td>
25 <td>10</td>
26 </tr>
27 <tr>
28 <td>Addition</td>
29 <td>${"10" + 10}</td>
30 <td>20</td>
31 </tr>

```

```

32 <tr>
33 <td>Subtraction</td>
34 <td>${"10" - 10}</td>
35 <td>0</td>
36 </tr>
37 <tr>
38 <td>Multiplication</td>
39 <td>${"10" * 10}</td>
40 <td>100</td>
41 </tr>
42 <tr>
43 <td>Division</td>
44 <td>${"10" / 3}</td>
45 <td>3.33</td>
46 </tr>
47 <tr>
48 <td>Modulus</td>
49 <td>${"10" % 3}</td>
50 <td>1</td>
51 </tr>

```

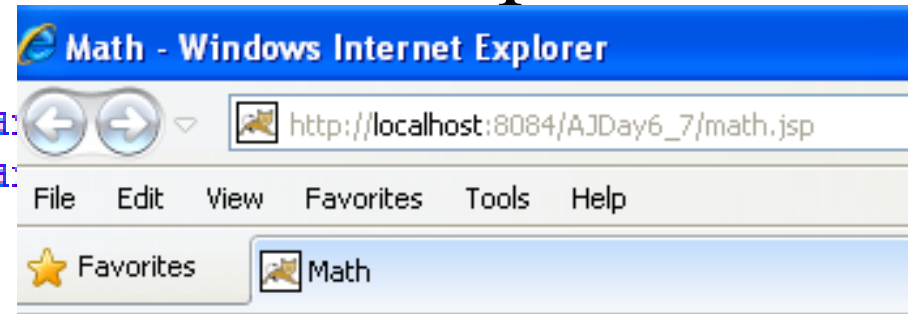
Expression Languages

EL Operators – Example

```

52 <tr>
53   <td>Divide by Zero</td>
54   <td>${"${"}10 / 0}</td>
55   <td>${10 / 0}</td>
56 </tr>
57 <tr>
58   <td>Exponential</td>
59   <td>${"${"}2E2}</td>
60   <td>${2E2}</td>
61 </tr>
62 <tr>
63   <td>Unary Minus</td>
64   <td>${"${"}-10}</td>
65   <td>${-10}</td>
66 </tr>
67 </table>
68 </body>
69 </html>
70

```



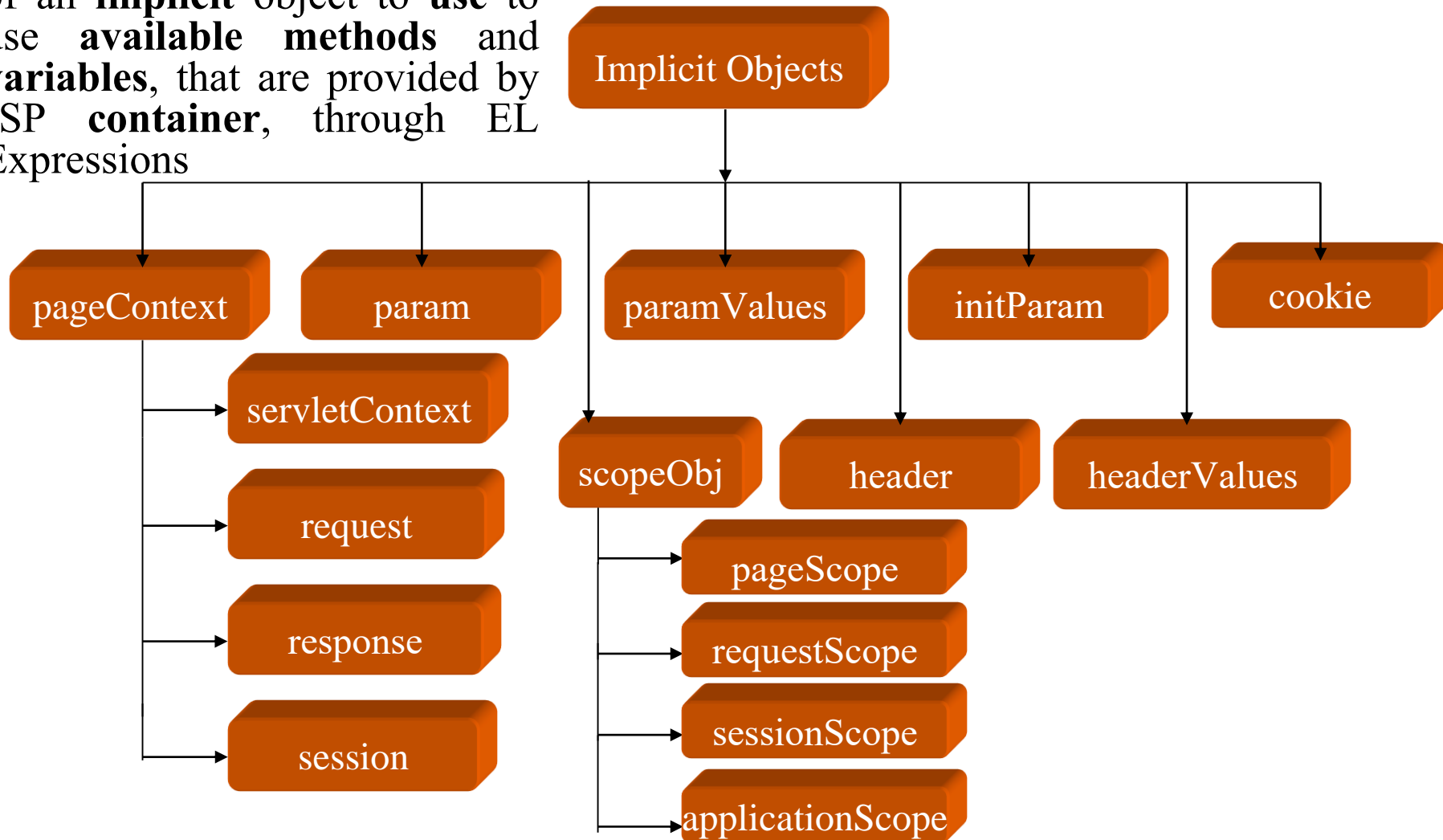
EL Expression

Concept	Expression	Result
Literal	<code>\${10}</code>	10
Addition	<code>\${10 + 10}</code>	20
Subtraction	<code>\${10 - 10}</code>	0
Multiplication	<code>\${10 * 10}</code>	100
Division	<code>\${10 / 3}</code>	3.3333333333333335
Modulus	<code>\${10 % 3}</code>	1
Divide by Zero	<code>\${10 / 0}</code>	Infinity
Exponential	<code>\${2E2}</code>	200.0
Unary Minus	<code>\${-10}</code>	-10

Expression Languages

EL Implicit Objects

- Are a **standard set of classes**.
- The user **creates** an **instance** of an **implicit** object to **use** to use **available methods** and **variables**, that are provided by **JSP container**, through **EL Expressions**



EL Implicit Objects

Objects	Descriptions
pageContext	<ul style="list-style-type: none"> - Can be used without creating an instance of the object - Provides access to page attributes - Can be used to access different page attributes
servletContext	- Specifies the JSP page, servlet and Web components contained in the same application. Communication with servlet container
session	- Represents the session created for the client sending a request
request	- Represents the request accepted by the JSP page from client
response	- Represents the response sent to the client by the JSP page. The response contains the data passed between a client and servlet
param	- Returns a value that maps a request parameter name to a single string value
paramValues	- Returns an array of values , which is mapped to the request parameters from client
header	- Returns a request header name & maps the value to single string value
headerValues	- Returns an array of values that is mapped to the request header
cookie	- Returns the cookie name mapped to a single cookie object
initParam	- Returns a context initialization parameter name , which is mapped to a single value

Expression Languages

EL Implicit Objects

- **getParameter**
 - **`${param.param_Name}`**
 - Similar to: `request.getParameter("param_Name")`
- **Get Properties in Java Beans**
 - **`${bean_id.property_name}` or `${bean_id["property_name"]}`**
 - Similar to: `<%= bean_id.getProperty_name %>`
 - Or, `<jsp:getProperty name="bean_id" property="prop_name"/>`

Expression Languages

Scoped Variables

- **Variables**
 - Are used to **store and access values** in JSP program
 - **Refers as a attributes** that are stored in standard scope such as **page, request, session and application**
 - **Ex:** `<% xxxContext.setAttribute(“info”, “att”) %>`
`${info}` or `${xxxScope.info}`
 - **Dot operator “.” or square brackets []** can be used to access value of variable
 - The **“.” and []** is used to **display a property** of java bean (**getter method**)
 - They are **not limited one level**
 - They support **accessing the arrays, lists, and map element** (e.g. `a[i]`, `list.get(i)`, `map.getValue(key)`)
 - **Ex**
 - `${pageScope.color}`
 - `${pageScope[“color”]}`
- The term **scoped variable** means that the variable confined to the **mentioned context only**.
- The EL **enhances supporting the retrieval of the stored objects as scoped variables**

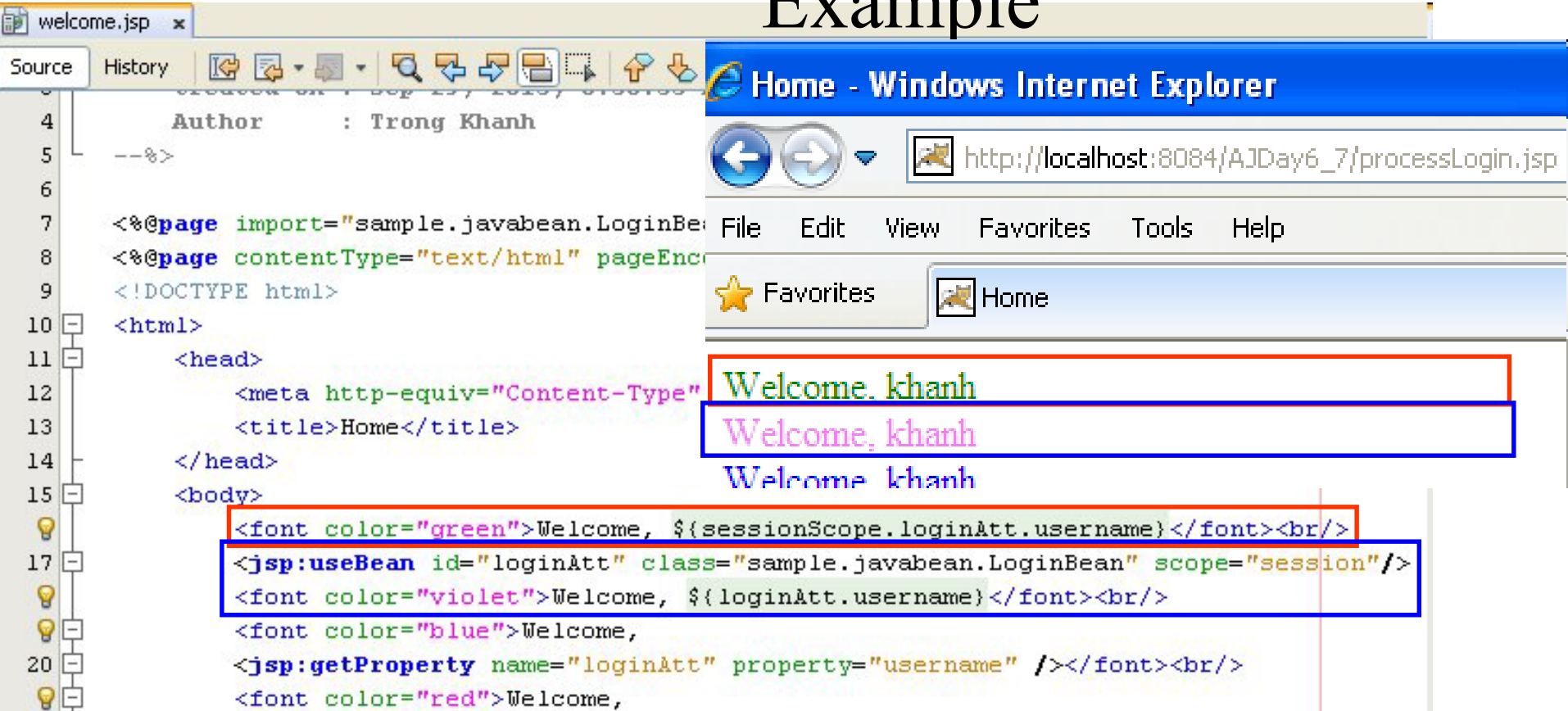
Expression Languages

Scoped Variables

Scopes	Descriptions
pageScope	<ul style="list-style-type: none">- Returns page-scoped variable names, which are mapped to their values- Is accessible from the JSP page that creates the object
requestScope	<ul style="list-style-type: none">- Provides access to the attributes of request object- Returns requests coped variable names, which are mapped to their values- Is accessible from web components handling a request that belongs to the session
sessionScope	<ul style="list-style-type: none">- Returns session-scoped variable names, which are mapped to their values- Is accessible from Web components handling a request that belong to the session
applicationScope	<ul style="list-style-type: none">- Returns application-scoped variable and maps the variable name to their values

Expression Languages

Example



The image shows a side-by-side comparison of JSP code and its rendered output in a browser.

Left Panel (Source Code):

```

4      Author      : Trong Khanh
5      --%>
6
7      <%@page import="sample.javabean.LoginBean"%>
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>
9      <!DOCTYPE html>
10     <html>
11     <head>
12         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"%>
13         <title>Home</title>
14     </head>
15     <body>
16         <font color="green">Welcome, ${sessionScope.loginAtt.username}</font><br/>
17         <jsp:useBean id="loginAtt" class="sample.javabean.LoginBean" scope="session"/>
18         <font color="violet">Welcome, ${loginAtt.username}</font><br/>
19         <font color="blue">Welcome,
20         <jsp:getProperty name="loginAtt" property="username" /></font><br/>
21         <font color="red">Welcome,
  
```

Right Panel (Browser Preview):

Home - Windows Internet Explorer

Address bar: http://localhost:8084/AJDay6_7/processLogin.jsp

Menu: File Edit View Favorites Tools Help

Favorites: Home

Rendered Output:

- Line 16: Welcome, khanh
- Line 18: Welcome, khanh
- Line 19: Welcome, khanh

Expression Languages

Example

```

home.jsp x
Source History
4      Author      : Trong Khanh
5      --%>
6
7      <%@page import="sample.javabeen.LoginBean"%>
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>
9      <!DOCTYPE html>
10     <html>
11     <head>
12         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13         <title>Home</title>
14     </head>
15     <body>
16         <font color="green">Welcome, ${sessionScope.loginAtt.username}</font><br/>
17         <font color="violet">Welcome, ${loginAtt.username}</font><br/>
18         <h1>Welcome to EL + JSTL</h1>
19         <form action="welcome.jsp">
20             Name <input type="text" name="txtName" value="" /><br/>
21             <input type="submit" value="Search" />
22         </form>
23         Title ${param.title}<br/>
24         Course ${param.course}<br/>
25     </body>
26 </html>
  
```

Expression Languages

Example

EL - Windows Internet Explorer

http://localhost:8084/AJDay6_7/elDemo.jsp

File Edit View Favorites Tools Help

★ Favorites EL

Calculating with EL

Num 1

Num 2

Total: 0

Avg: 0.0

EL - Windows Internet Explorer

http://localhost:8084/AJDay6_7/elDemo.jsp

File Edit View Favorites Tools Help

★ Favorites EL

Calculating with EL

Num 1

Num 2

Total: 11

Avg: 5.5

```
<form method="POST">
  Num 1 <input type="text" name="num1" value="${param.num1}" /><br/>
  Num 2 <input type="text" name="num2" value="${param.num2}" /><br/>
  Total: ${param["num1"] + param.num2}<br/>
  Avg: ${ (param.num1 + param["num2"]) / 2}<br/>
  <input type="submit" value="Calculate" />
</form>
```

Summary

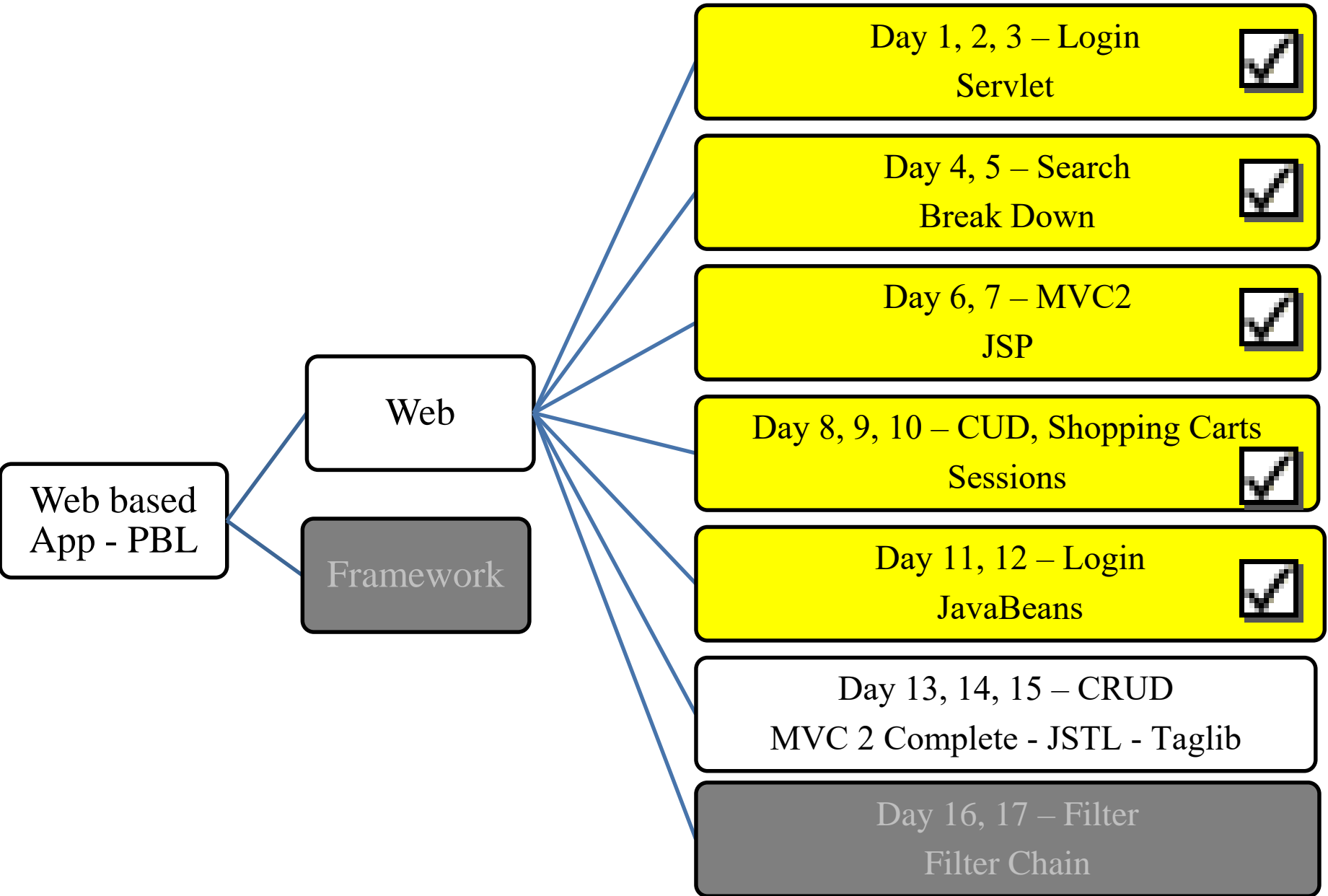
- **How to build the Web Application using MVC1?**
How to remove the java code in the jsp (view)?
 - Standard Actions
 - Dispatching Mechanisms
 - Expression Language

Q&A

Next Lecture

- **How to remove completed Scripting Element (Java Code) in JSP (View)? Complete the View of MVC 2 Design Pattern**
 - JSTL
- **How to build the data grid tag library using in JSP?**
 - **Tag Libraries**
 - Model
 - Classical, Simple, and Handles
 - How to implement the custom Tag Lib and use it in JSP

Next Lecture



Appendix - MVC1

Login page

```
login.html x
Source History
5 <!DOCTYPE html>
6 <html>
7   <head>
8     <title>Login</title>
9   </head>
10  <body>
11    <h1>Login Page</h1>
12    <form action="processLogin.jsp" method="POST">
13      Username <input type="text" name="txtUsername" value="" /><br/>
14      Password <input type="password" name="txtPassword" value="" /><br/>
15      <input type="submit" value="Login" />
16      <input type="reset" value="Reset" />
17    </form>
18  </body>
19 </html>
```


MVC1

Login Beans

```

LoginBean.java
Source History
14  * @author Trong Khanh
15  */
16  public class LoginBean implements Serializable {
17      private String username;
18      private String password;
19
20      public LoginBean() {...2 lines }
21
22
23      /**...3 lines */
24
26      public String getUsername() {...3 lines }
27
28
29
30      /**...3 lines */
31
33      public void setUsername(String username) {...3 lines }
34
35
36
37      /**...3 lines */
38
40      public String getPassword() {...3 lines }
41
42
43
44      /**...3 lines */
45
47      public void setPassword(String password) {...3 lines }
48
49
50
51      public boolean checkLogin () {
52          RegistrationDAO dao = new RegistrationDAO();
53          boolean result = dao.checkLogin(username, password);
54
55          return result;
56      }
57  }

```


MVC1

Process Login JSP

```

processLogin.jsp x
Source History
4      Author      : Trong Khanh
5      --%>
6      <%@page contentType="text/html" pageEncoding="UTF-8"%>
7      <!DOCTYPE html>
8      <html>
9      <head>
10         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
11         <title>Processing</title>
12     </head>
13     <body>
14         <h1>Process Login</h1>
15         <jsp:useBean id="loginAtt" class="sample.javabeans.LoginBean" scope="session"/>
16         <jsp:setProperty name="loginAtt" property="username"
17             value='<%= request.getParameter("txtUsername") %>' />
18         <jsp:setProperty name="loginAtt" property="password"
19             value='<%= request.getParameter("txtPassword") %>' />
20         <%
21             if (loginAtt.checkLogin()) {
22                 response.sendRedirect("welcome.jsp");
23             } else {
24                 out.print("Invalid username or password!!!!");
25             }
26         %>
27     </body>
28 </html>

```

MVC1

Process Login JSP

```

processLogin.jsp x
Source History
4      Author      : Trong Khanh
5      --%>
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE html>
9      <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12         <title>Processing</title>
13     </head>
14     <body>
15         <h1>Process Login</h1>
16         <jsp:useBean id="loginAtt" class="sample.javabean.LoginBean" scope="session"/>
17         <jsp:setProperty name="loginAtt" property="username" param="txtUsername"/>
18         <jsp:setProperty name="loginAtt" property="password" param="txtPassword"/>
19
20         <%
21             if (loginAtt.checkLogin()) {
22                 response.sendRedirect("welcome.jsp");
23             } else {
24                 out.print("Invalid username or password!!!!");
25             }
26         %>
27     </body>
28 </html>
  
```

MVC1

Welcome page

```

welcome.jsp x
Source History
4      Author      : Trong Khanh
5      --%>
6
7      <%@page import="sample.javabeen.LoginBean"%>
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>
9      <!DOCTYPE html>
10     <html>
11     <head>
12         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13         <title>Home</title>
14     </head>
15     <body>
16         <jsp:useBean id="loginAtt" class="sample.javabeen.LoginBean" scope="session"/>
17         <font color="blue">Welcome,
18         <jsp:getProperty name="loginAtt" property="username" /></font><br/>
19         <font color="red">Welcome,
20         <%= ((LoginBean) session.getAttribute("loginAtt")).getUsername() %></font><br/>
21         <h1>Welcome to Standard Action</h1>
22         <form action="welcome.jsp">
23             Name <input type="text" name="txtName" value="" /><br/>
24             <input type="submit" value="Search" />
25         </form>
26     </body>
27 </html>

```

MVC1

Optimizing

```
<form action="processLogin.jsp" method="POST">
    Username <input type="text" name="username" value="" /><br/>
    Password <input type="password" name="password" value="" /><br/>
    <input type="submit" value="Login" />
    <input type="reset" value="Reset" />
</form>
```

```
<jsp:useBean id="loginAtt" class="sample.javabean.LoginBean" scope="session"/>
<jsp:setProperty name="loginAtt" property="username" />
<jsp:setProperty name="loginAtt" property="password" />
```

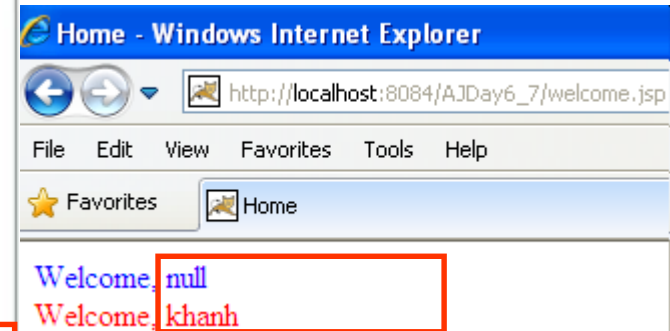
```
<jsp:useBean id="loginAtt" class="sample.javabean.LoginBean" scope="session"/>
<jsp:setProperty name="loginAtt" property="*" />
```

Appendix – JSP Standard Actions

Scope of JavaBeans – Example

```
processLogin.jsp
Source History
4      Author      : Trong Khanh
5      --%>
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE html>
9      <html>
10     <.../>
14     <body>
15         <h1>Process Login</h1>
16         <jsp:useBean id="loginAtt" class="sample.javabeen.LoginBean" scope="session"/>
17         <jsp:setProperty name="loginAtt" property="*" />
```

```
welcome.jsp
Source History
4      Author      : Trong Khanh
5      --%>
6
7      <%@page import="sample.javabeen.LoginBean"%>
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>
9      <!DOCTYPE html>
10     <html>
11     <.../>
15     <body>
16         <jsp:useBean id="loginAtt" class="sample.javabeen.LoginBean"/>
```



JSP Standard Actions

Scope of JavaBeans – Example

```

welcome.jsp
Source History
4      Author      : Trong Khanh
5      -->
6
7      <%@page import="sample.javabeen.LoginBean"%>
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>
9      <!DOCTYPE html>
10     <html>
11     <.../>
12
13     <body>
14
15     <font color="blue">Welcome,
16
17     <jsp:getProperty name="loginAtt" property="username" /></font><br/>
18

```

Apache Tomcat/7.0.34 - Error report - Windows Internet Explorer

http://localhost:8084/AJDay6_7/welcome.jsp

File Edit View Favorites Tools Help

★ Favorites Apache Tomcat/7.0.34 - Error report

HTTP Status 500 - file:/welcome.jsp(18,8) jsp:getProperty for bean with name 'loginAtt'. Name was not previously introduced as per JSP.5.3

type Exception report

message file:/welcome.jsp(18,8) jsp:getProperty for bean with name 'loginAtt'. Name was not previously introduced as per JSP.5.3

description The server encountered an internal error that prevented it from fulfilling this request.

exception

org.apache.jasper.JasperException: file:/welcome.jsp(18,8) jsp:getProperty for bean with name 'loginAtt'.

JSP Standard Actions

Scope of JavaBeans, accessing on Servlet via HttpSession– Example

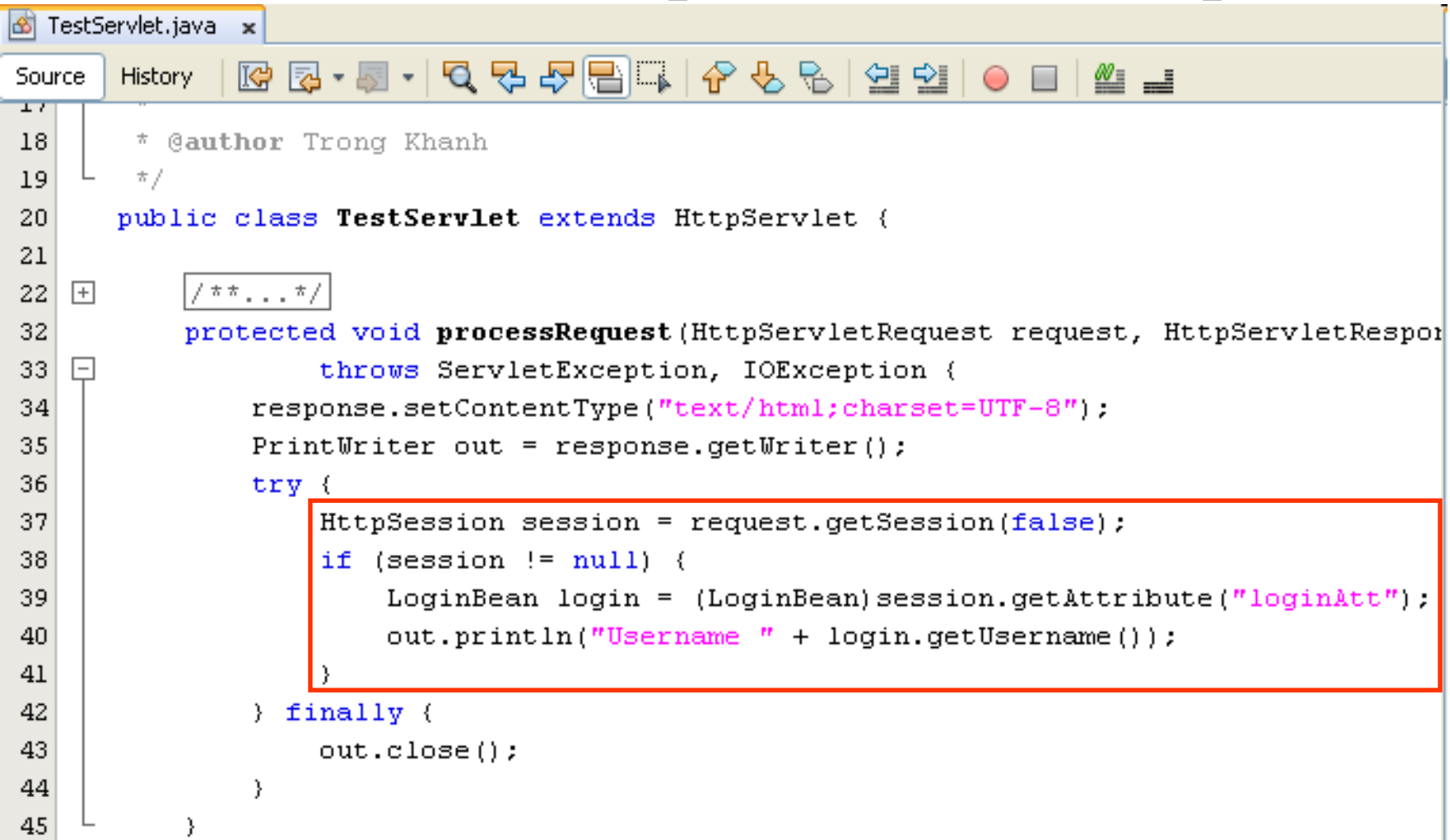
```

welcome.jsp x
Source History
4      Author      : Trong Khanh
5      --%>
6
7      <%@page import="sample.javabeen.LoginBean"%>
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>
9      <!DOCTYPE html>
10     <html>
11         <.../>
12     </html>
13
14     <body>
15
16         <jsp:useBean id="loginAtt" class="sample.javabeen.LoginBean" scope="session"/>
17         <font color="blue">Welcome,
18         <jsp:getProperty name="loginAtt" property="username" /></font><br/>
19         <font color="red">Welcome,
20         <%= ((LoginBean) session.getAttribute("loginAtt")).getUsername() %></font><br/>
21         <h1>Welcome to Standard Action</h1>
22         <form action="welcome.jsp">
23             Name <input type="text" name="txtName" value="" /><br/>
24             <input type="submit" value="Search" />
25         </form>
26         <a href="TestServlet">Go to Test Servlet </a>
27     </body>
28 </html>

```

JSP Standard Actions

Scope of JavaBeans, accessing on Servlet via HttpSession– Example



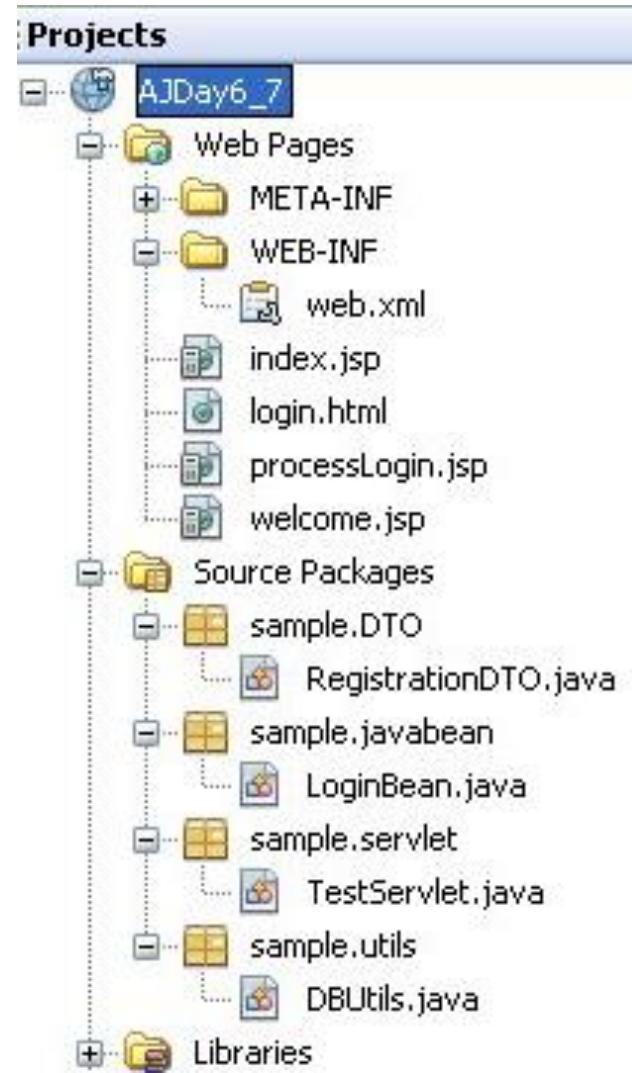
```

17
18  * @author Trong Khanh
19  */
20  public class TestServlet extends HttpServlet {
21
22      /**...*/
23
24      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
25          throws ServletException, IOException {
26          response.setContentType("text/html; charset=UTF-8");
27          PrintWriter out = response.getWriter();
28          try {
29              HttpSession session = request.getSession(false);
30              if (session != null) {
31                  LoginBean login = (LoginBean) session.getAttribute("loginAtt");
32                  out.println("Username " + login.getUsername());
33              }
34          } finally {
35              out.close();
36          }
37      }
38  }
39
40
41
42
43
44
45

```

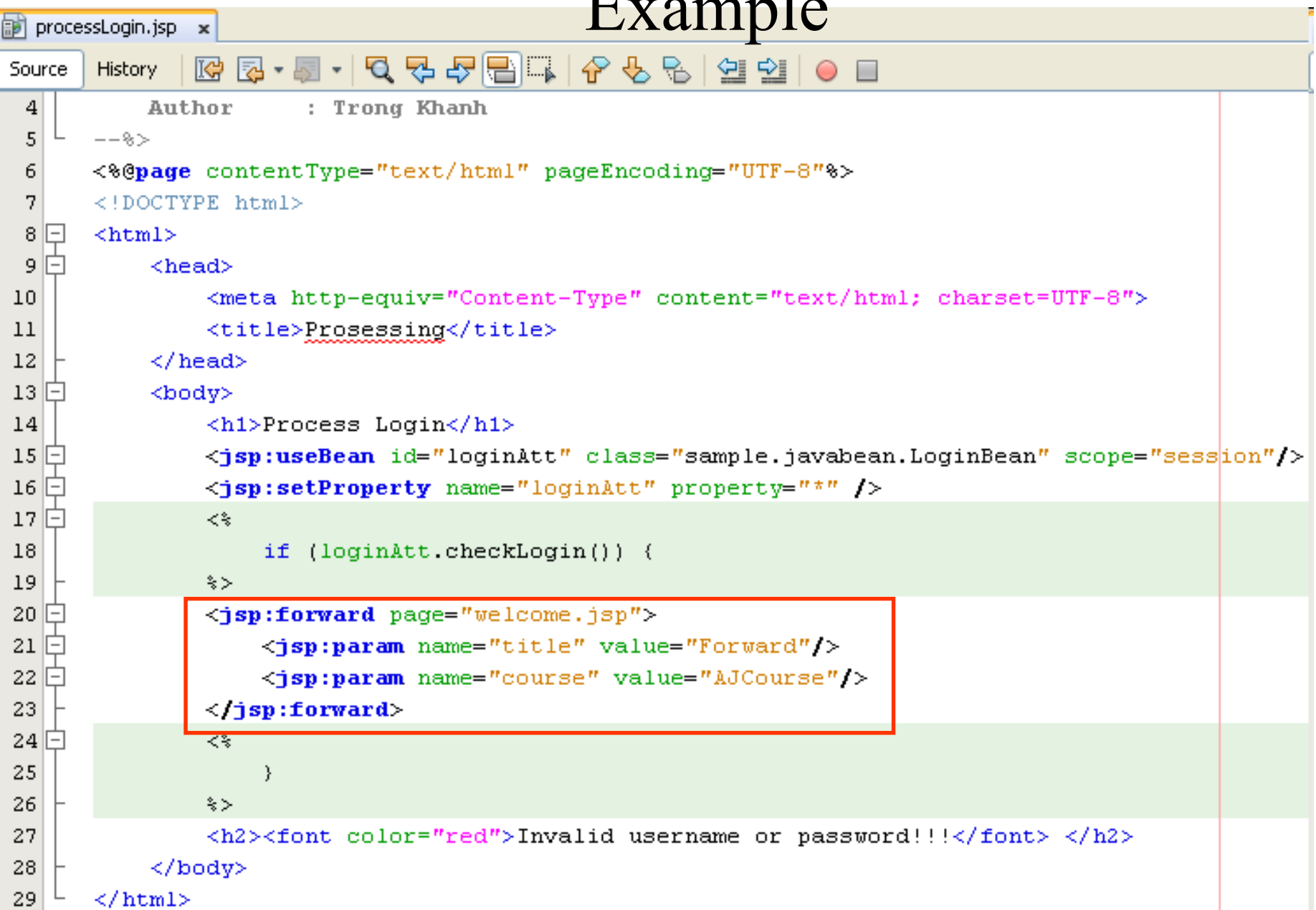

JSP Standard Actions

Scope of JavaBeans, accessing on Servlet via HttpSession– Example



Appendix – Dispatching Mechanisms

Example



```
processLogin.jsp x
Source History
4 Author : Trong Khanh
5 --%>
6 <%@page contentType="text/html" pageEncoding="UTF-8"%>
7 <!DOCTYPE html>
8 <html>
9 <head>
10 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
11 <title>Processing</title>
12 </head>
13 <body>
14 <h1>Process Login</h1>
15 <jsp:useBean id="loginAtt" class="sample.javabean.LoginBean" scope="session"/>
16 <jsp:setProperty name="loginAtt" property="*" />
17 <%
18     if (loginAtt.checkLogin()) {
19
20         <jsp:forward page="welcome.jsp">
21             <jsp:param name="title" value="Forward"/>
22             <jsp:param name="course" value="AJCourse"/>
23         </jsp:forward>
24     }
25 }
26 %>
27 <h2><font color="red">Invalid username or password!!!</font> </h2>
28 </body>
29 </html>
```

Dispatching Mechanisms

Example

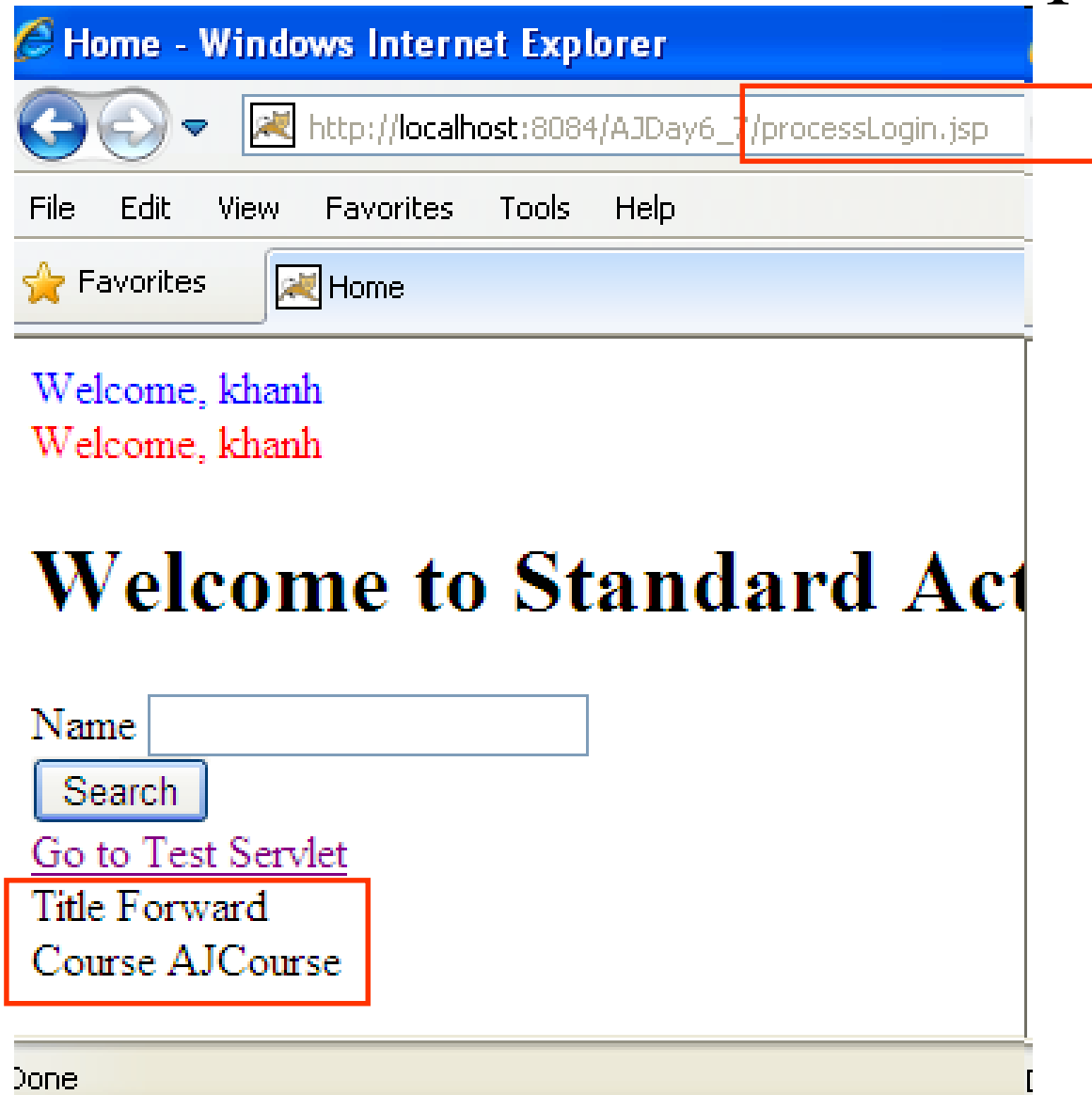
```

welcome.jsp x
Source History
4      Author      : Trong Khanh
5      --%>
6
7      <%@page import="sample.javabeen.LoginBean"%>
8      <%@page contentType="text/html" pageEncoding="UTF-8"%>
9      <!DOCTYPE html>
10     <html>
11     <head>
12         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13         <title>Home</title>
14     </head>
15     <body>
16         <jsp:useBean id="loginAtt" class="sample.javabeen.LoginBean" scope="session"/>
17         <font color="blue">Welcome,
18         <jsp:getProperty name="loginAtt" property="username" /></font><br/>
19         <font color="red">Welcome,
20         <%= ((LoginBean) session.getAttribute("loginAtt")).getUsername() %></font><br/>
21         <h1>Welcome to Standard Action</h1>
22         <form action="welcome.jsp">
23             Name <input type="text" name="txtName" value="" /><br/>
24             <input type="submit" value="Search" />
25         </form>
26         <a href="TestServlet">Go to Test Servlet </a><br/>
27         Title <%= request.getParameter("title") %><br/>
28         Course <%= request.getParameter("course") %><br/>
29     </body>
30 </html>

```

Dispatching Mechanisms

Example



Appendix

XML for JSPs

- **HTML is narrowly focused** on marking up take
- **XML can be defined by users** and can **used** for **making up text** (as in XHTML), but it has a pretty much infinite set of the other possible uses
- **The tag in jsp has**
 - **Opening tag**
 - **Closing tag**
 - **Data content between** the opening and closing tags (called as the body)
 - The **opening tag can have a prefix** and can **contain many attributes**
 - **Contain the nested tag**

Appendix

XML-Friendly Syntax

- **JSP documents**
 - Are JSP source files written entirely in XML syntax
 - Have a **.jspx** extension
 - Can be also be **identified by setting in web deployment descriptor**

```
<jsp-config>
```

```
  <jsp-property-group>
```

```
    <url-pattern>/jspx/*</url-pattern>
```

```
    <is-xml>true</is-xml>
```

```
  </jsp-property-group>
```

```
</jsp-config>
```

- Or, Define in a JSP document as form

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page">
```

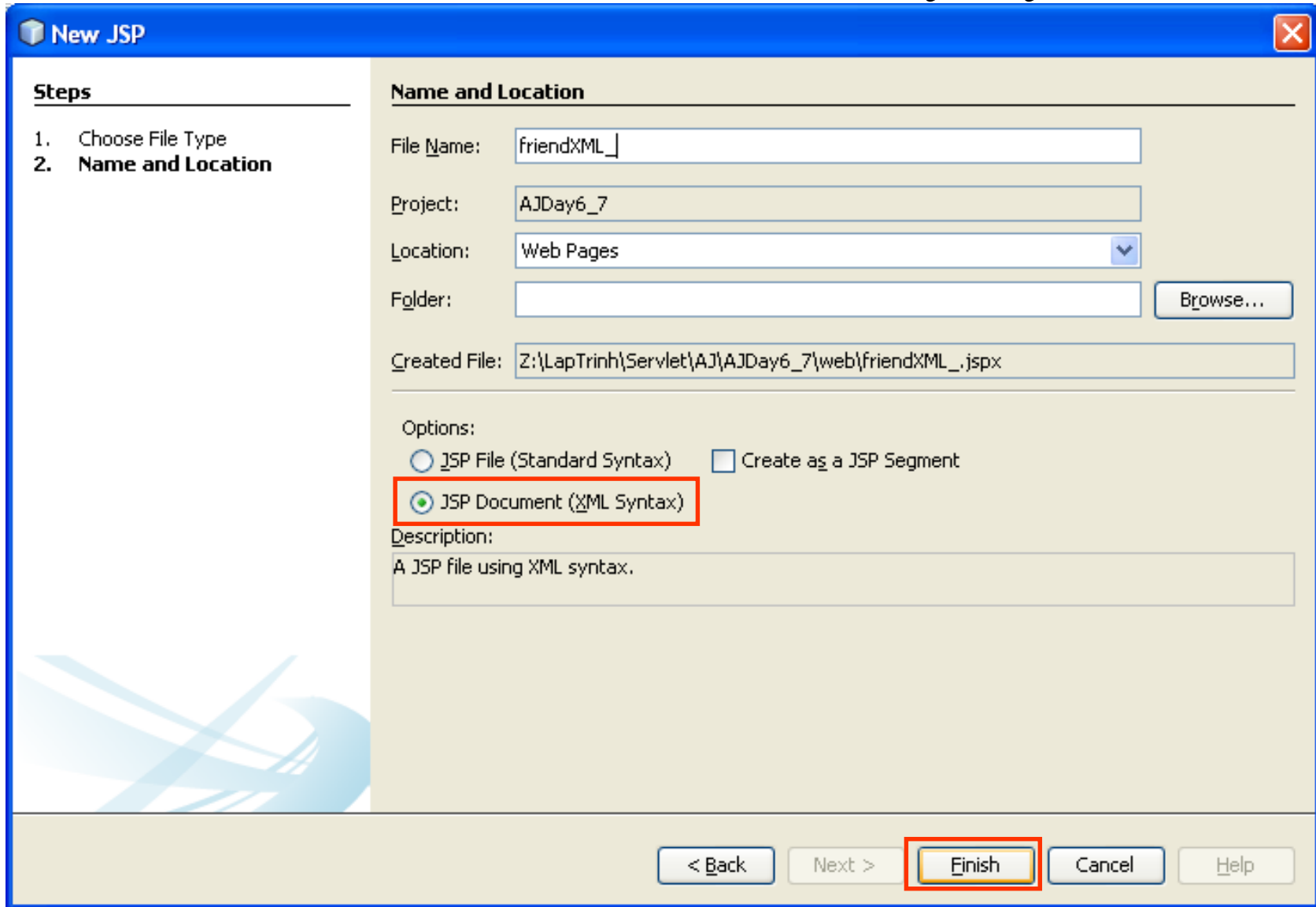
Appendix

XML-Friendly Syntax

- JSP document **syntax provides replacements for all <%-type scripting element syntax**
 - `<jsp:scriptlet>...</jsp:scriptlet>` replace `<%...%>`
 - `<jsp:expression>...</jsp:expression>` replace `<%= ... %>`
 - `<jsp:declaration>...</jsp:declaration>` replace `<%! ... %>`
 - `<jsp:directive.page .../>` replace `<%@page ...%>`
 - `<jsp:directive.include .../>` replace `<%@ include ...%>`
 - `<!-- ... -->` replace `<%-- ... --%>`
 - There are some things within the Java language itself that are anathema to XML validators (ex: “<” symbol looks like the beginning of an opening or closing tag, and an XML validator will assuredly treat it as such)
 - Using as an **entity** in **XML** that **begin** with an **ampersand (&)** and **end** on a **semicolon (;)**. Ex: `<`
 - **Using** `<![CDATA[symbol]]>` or `<![CDATA[command]]>`
 - Ex: `<![CDATA[for(int i=0; i<10; i++)]]>` or `i <![CDATA[<]]> 10`

Appendix

XML-Friendly Syntax



New JSP

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

File Name: friendXML_

Project: AJDay6_7

Location: Web Pages

Folder: Browse...

Created File: Z:\LapTrinh\Servlet\AJ\AJDay6_7\web\friendXML_.jspx

Options:

☐ JSP File (Standard Syntax) ☐ Create as a JSP Segment

☒ **JSP Document (XML Syntax)**

Description:

A JSP file using XML syntax.

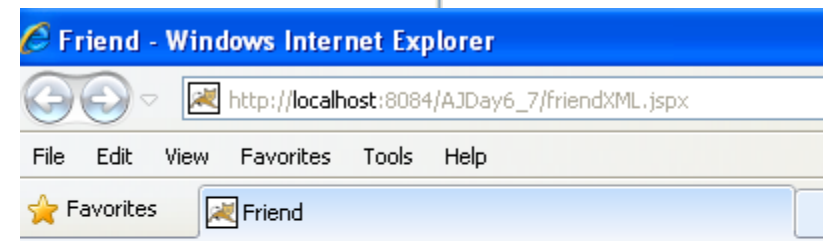
< Back Next > **Finish** Cancel Help

Appendix

XML-Friendly Syntax – Sample

```

5      Author      : Trong Khanh
6      -->
7      <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
8          <jsp:directive.page contentType="text/html" pageEncoding="UTF-8"/>
9          <jsp:element name="text">
10              <title>Friend</title>
11              <jsp:body>
12                  <h1>XML Friendly Syntax Demo</h1>
13                  <jsp:declaration>
14                      public int add(int n) {
15                          return n + 5;
16                      }
17                  </jsp:declaration>
18                  <jsp:scriptlet>int a = add(5);</jsp:scriptlet>
19                  The Result is: <jsp:expression>a</jsp:expression>
20              </jsp:body>
21          </jsp:element>
22      </jsp:root>
  
```



XML Friendly Syntax Demo

The Result is: 10

Appendix

XML-Friendly Syntax – Sample



```

5  Author      : Trong Khanh
6  -->
7  <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
8    <jsp:directive.page contentType="text/html" pageEncoding="UTF-8"/>
9    <jsp:element name="text">
10      <title>Friend</title>
11      <jsp:body>
12        <h1>XML Friendly Syntax Demo</h1>
13        <jsp:declaration>
14          public int add(int n) {
15            return n + 5;
16          }
17        </jsp:declaration>
18        <jsp:scriptlet>int a = add(5);</jsp:scriptlet>
19        The Result is: <jsp:expression>a</jsp:expression>
20        <br/>
21        <jsp:scriptlet>
22          for (int i = 0; i < 5; i++) {
23            out.print("");
24          }
25        </jsp:scriptlet>
26      </jsp:body>
27    </jsp:element>
28  </jsp:root>
  
```

<jsp:scriptlet>
 for (int i = 0; <![CDATA[i < 5]]>; i++) {
 out.print("");
 }
 </jsp:scriptlet>

<jsp:scriptlet>
 <![CDATA[for (int i = 0; i < 5; i++)]]> {
 out.print("");
 }
 </jsp:scriptlet>

Appendix

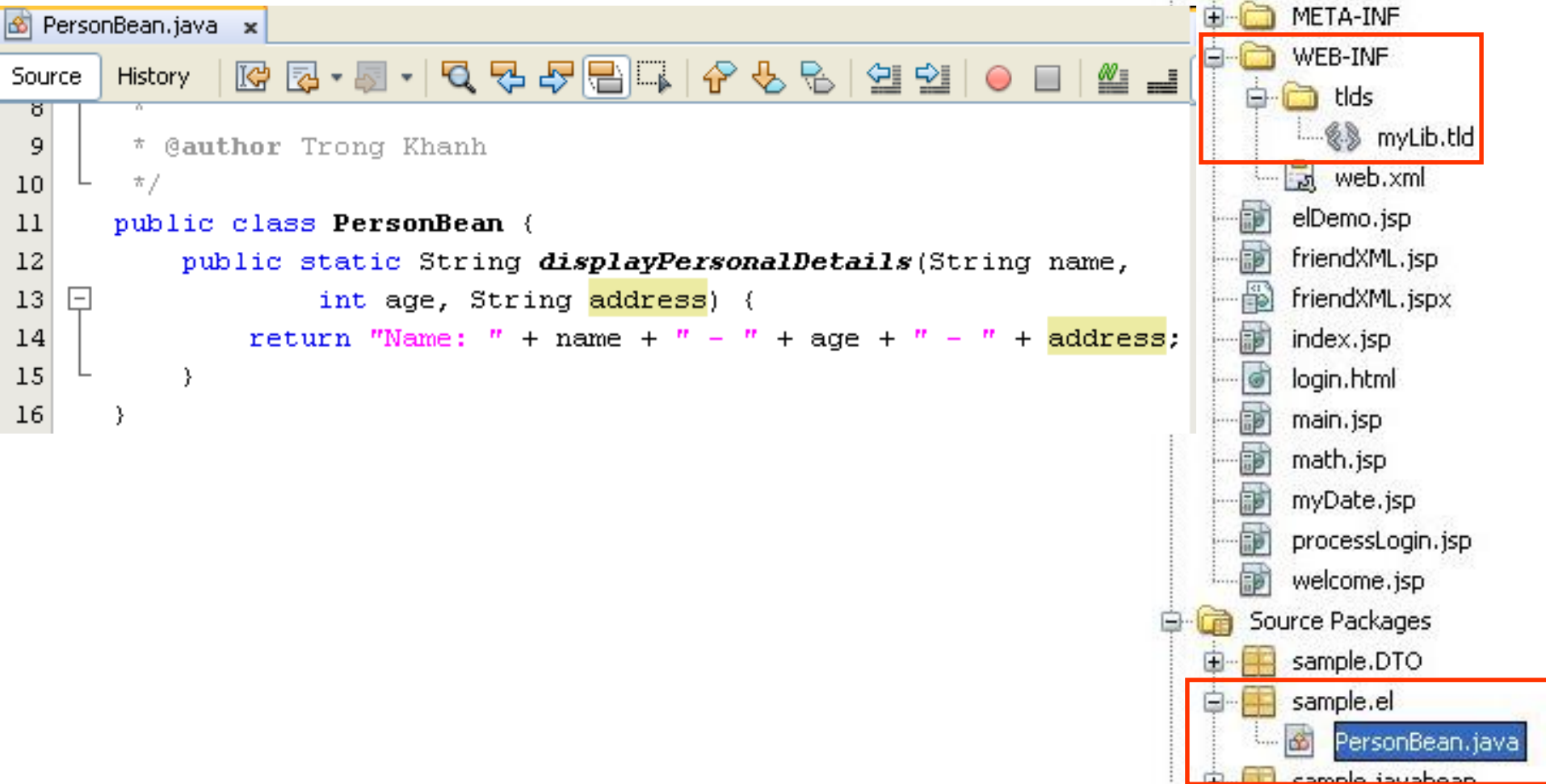
Functions using EL

- Supports **using** of the **Java function within** the JSP page is an **easy** as **using a tag**
- The **following steps** to set up EL to Java functions
 - **Step 1:** Creating “static” method
 - **Step 2:** Creating Tag Library Descriptor
 - **Step 3:** Modifying Deployment Descriptor and locating the TLD file in web deployment descriptor (if necessary)
 - **Step 4:** Access EL functions within JSP

Appendix

Creating “static” method

- The static java methods can be called within the EL expression
- Ex:



The screenshot displays an IDE with two main components: a source code editor on the left and a project explorer on the right.

Source Code Editor: The file `PersonBean.java` is open. It contains the following code:

```

8  "
9  * @author Trong Khanh
10 * /
11 public class PersonBean {
12     public static String displayPersonalDetails(String name,
13         int age, String address) {
14         return "Name: " + name + " - " + age + " - " + address;
15     }
16 }
    
```

Project Explorer: The project structure is shown on the right. The `Projects pane lists the following items:`

- `AJDay6_7` (Project)
 - `Web Pages` (Folder)
 - `META-INF` (Folder)
 - `WEB-INF` (Folder)
 - `tlds` (Folder)
 - `myLib.tld` (File) - **Highlighted with a red box**
 - `web.xml` (File)
 - `elDemo.jsp` (File)
 - `friendXML.jsp` (File)
 - `friendXML.jspx` (File)
 - `index.jsp` (File)
 - `login.html` (File)
 - `main.jsp` (File)
 - `math.jsp` (File)
 - `myDate.jsp` (File)
 - `processLogin.jsp` (File)
 - `welcome.jsp` (File)
 - `Source Packages` (Folder)
 - `sample.DTO` (Package)
 - `sample.el` (Package)
 - `PersonBean.java` (File) - **Highlighted with a red box**

Appendix

Creating Tag Library Descriptor

- **After defining** the functions, the **function name should be mapped** with EL **using a Tag Library Descriptor (TLD) file**
- A defined in a class with EL. **TLD file uses XML syntax to map the name of functions**
- Save this TLD file in the **/WEB-INF/tlds folder**, where tlds is a user-created folder

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-  
    jsptaglibrary_2_0.xsd">
```

```
<tlib-version>1.0</tlib-version>
```

```
<function>
```

```
  <description>information description</description>
```

```
  <name>functionName</name>
```

```
  <function-class>Java class</function-class>
```

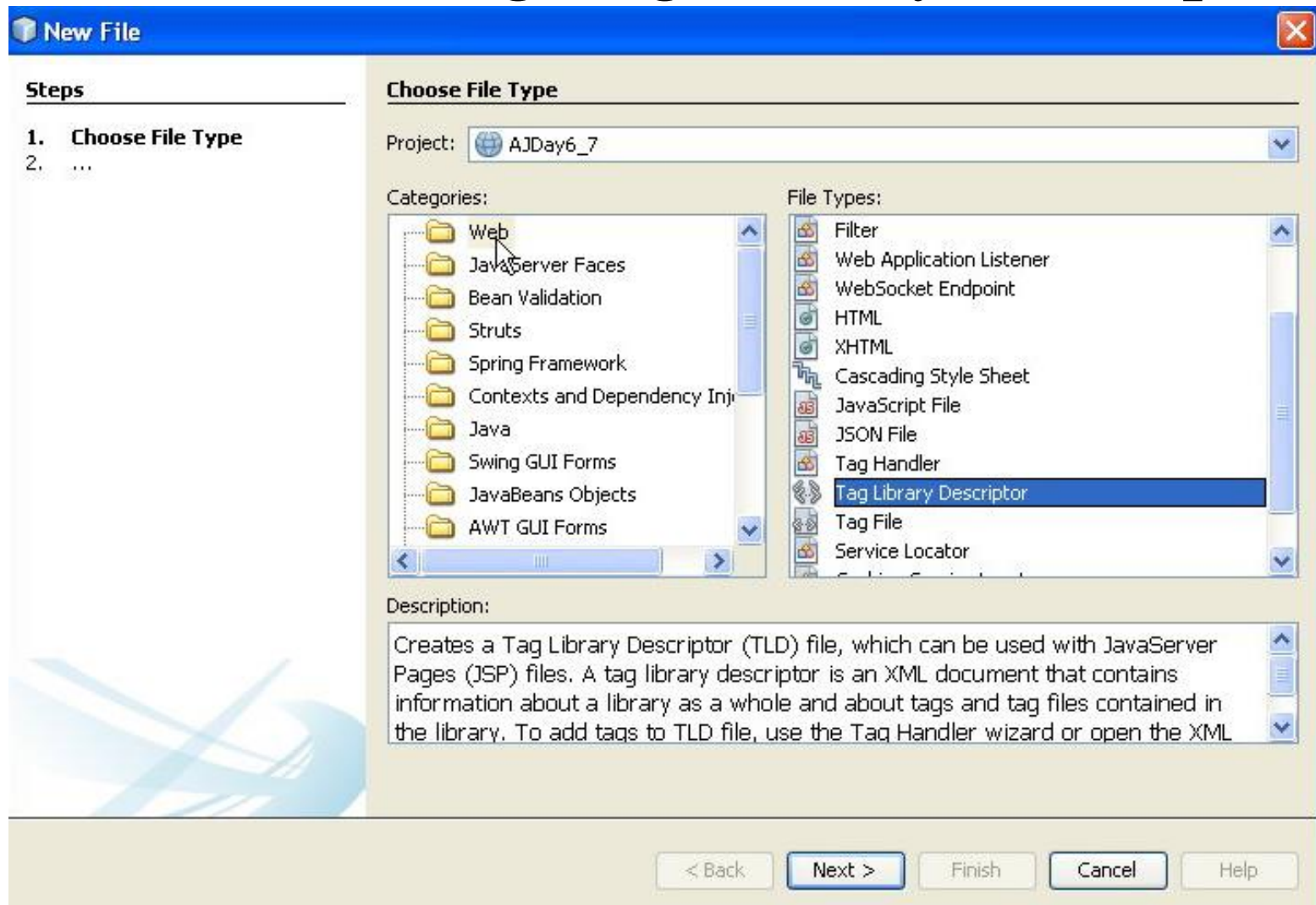
```
  <function-signature>declared method with parameters</function-signature>
```

```
</function>
```

```
</taglib>
```

Appendix

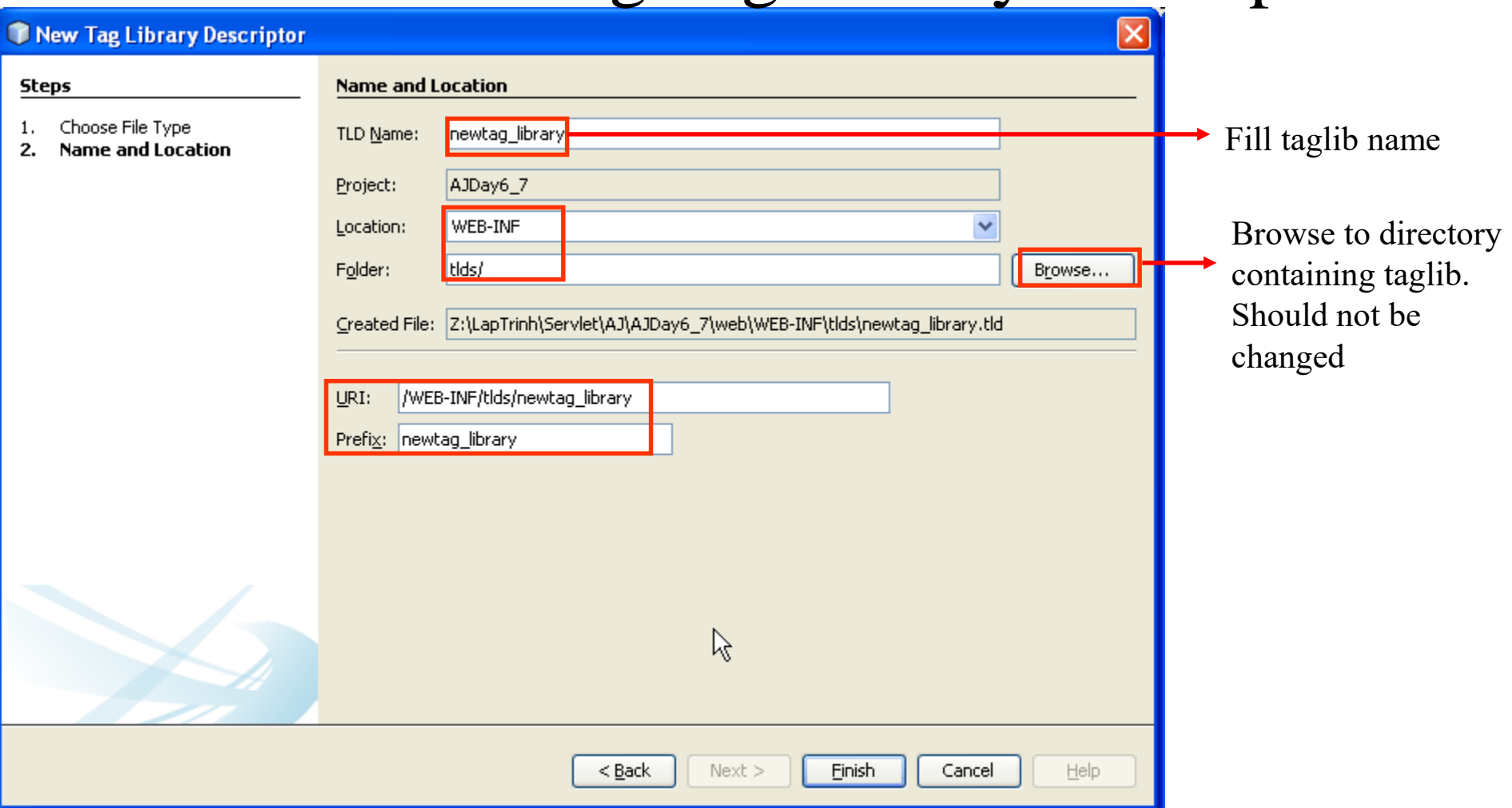
Creating Tag Library Descriptor



- Click Next button

Appendix

Creating Tag Library Descriptor



The screenshot shows the 'New Tag Library Descriptor' dialog box. On the left, a 'Steps' pane lists '1. Choose File Type' and '2. Name and Location'. The main area is titled 'Name and Location' and contains several input fields. Red boxes highlight the 'TLD Name' field (containing 'newtag_library'), the 'Location' dropdown (set to 'WEB-INF'), the 'Folder' field (containing 'tlds/'), the 'Browse...' button, the 'URI' field (containing '/WEB-INF/tlds/newtag_library'), and the 'Prefix' field (containing 'newtag_library'). Red arrows point from text annotations to these fields. The 'Created File' field shows the path 'Z:\LapTrinh\Servlet\AJ\AJDay6_7\web\WEB-INF\tlds\newtag_library.tld'. At the bottom are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

TLD Name: newtag_library → Fill taglib name

Project: AJDay6_7

Location: WEB-INF

Folder: tlds/ Browse... → Browse to directory containing taglib. Should not be changed

Created File: Z:\LapTrinh\Servlet\AJ\AJDay6_7\web\WEB-INF\tlds\newtag_library.tld

URI: /WEB-INF/tlds/newtag_library

Prefix: newtag_library

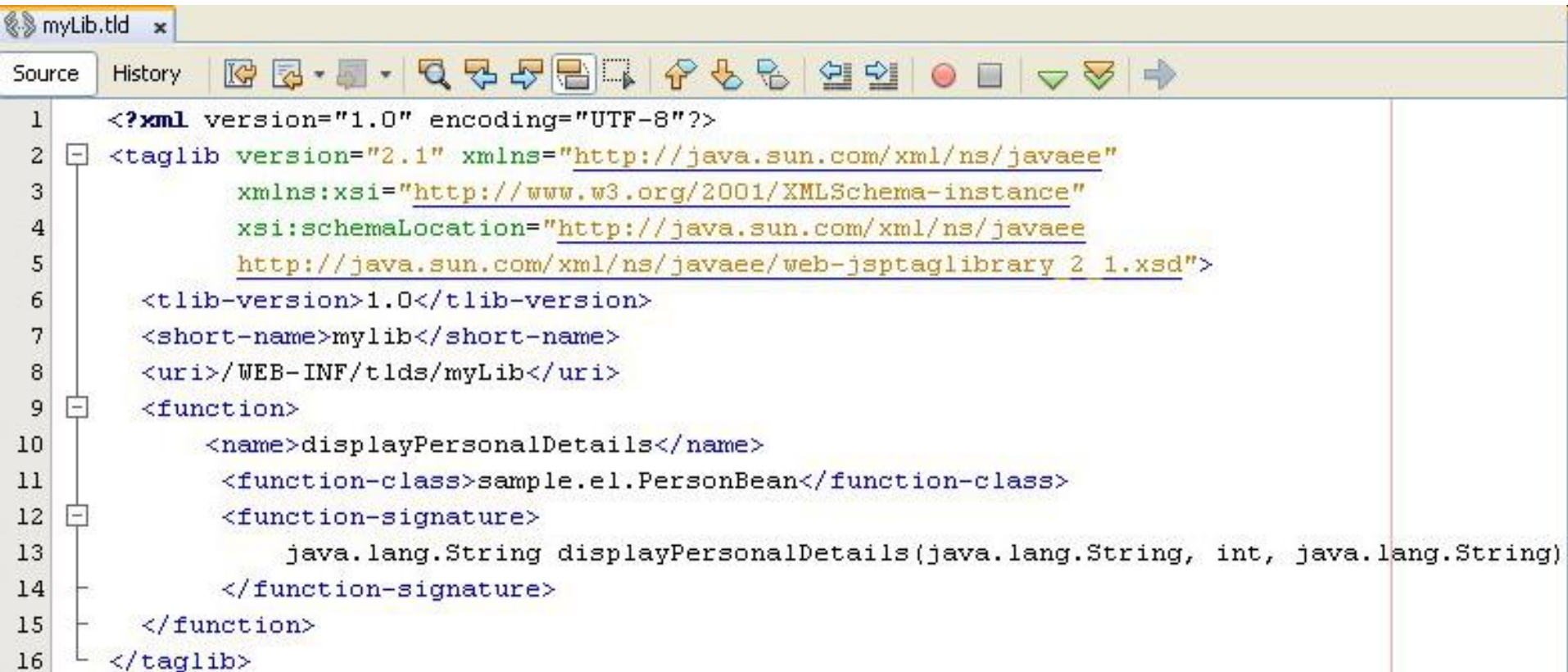
< Back Next > Finish Cancel Help

- Click **Finish** button
- The taglib file with extension is **tld** located at the **WEB-INF/tlds** directory

Appendix

Creating Tag Library Descriptor

- Modified the .tld file



```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5          http://java.sun.com/xml/ns/javaee/web-jsptaglibrary 2 1.xsd">
6      <tlib-version>1.0</tlib-version>
7      <short-name>mylib</short-name>
8      <uri>/WEB-INF/tlds/myLib</uri>
9      <function>
10         <name>displayPersonalDetails</name>
11         <function-class>sample.el.PersonBean</function-class>
12         <function-signature>
13             java.lang.String displayPersonalDetails(java.lang.String, int, java.lang.String)
14         </function-signature>
15     </function>
16 </taglib>
  
```


Appendix

Modifying the Deployment Descriptor

- The default mode for JSP pages delivered with JSP version 2.0 technology is to evaluate EL expressions
- JSP EL expression can be **enabled or disabled** with **02 ways**

- Using isELIgnored attribute in the JSP page directive

`<%@ page isELIgnored= “true | false” %>`

- Modifying in web.xml

`<jsp-config>`

`<jsp-property-group>`

`<url-pattern>*.jsp</url-pattern>`

`<el-ignored>false</el-ignored>`

`</jsp-property-group>`

`</jsp-config>`

Appendix

Accessing EL functions within JSP

- To **access** the **function** created in a TLD file using a JSP file, developer need **to import the TLD file** using the **taglib directive**. In the directive statement, developer **need** to **mention** a prefix for the tags and location of the TLD file

<%@taglib prefix="prefix" uri="path" %>

- **After importing** the TLD file, **developer** can **access the function** **using** an EL expression

\${prefix:funcName(args)}

Appendix

Example

```

Source History
4      Author      : Trong Khanh
5      --%>
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE html>
9      <%@taglib uri="/WEB-INF/tlds/myLib.tld" prefix="my"%>
10     <html>
11     <head>
12         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13         <title>ElFunc</title>
14     </head>
15     <body>
16         <h1>EL Function Demo</h1>
17         ${my:displayPersonalDetails("KhanhKT", 33, "QTSC")}<br/>
18         ${my:displayPersonalDetails("KTKhanh", 43, "HCMC")}<br/>
19         ${my:displayPersonalDetails("KhanhKK", 23, "HCMC")}
20     </body>
21 </html>

```

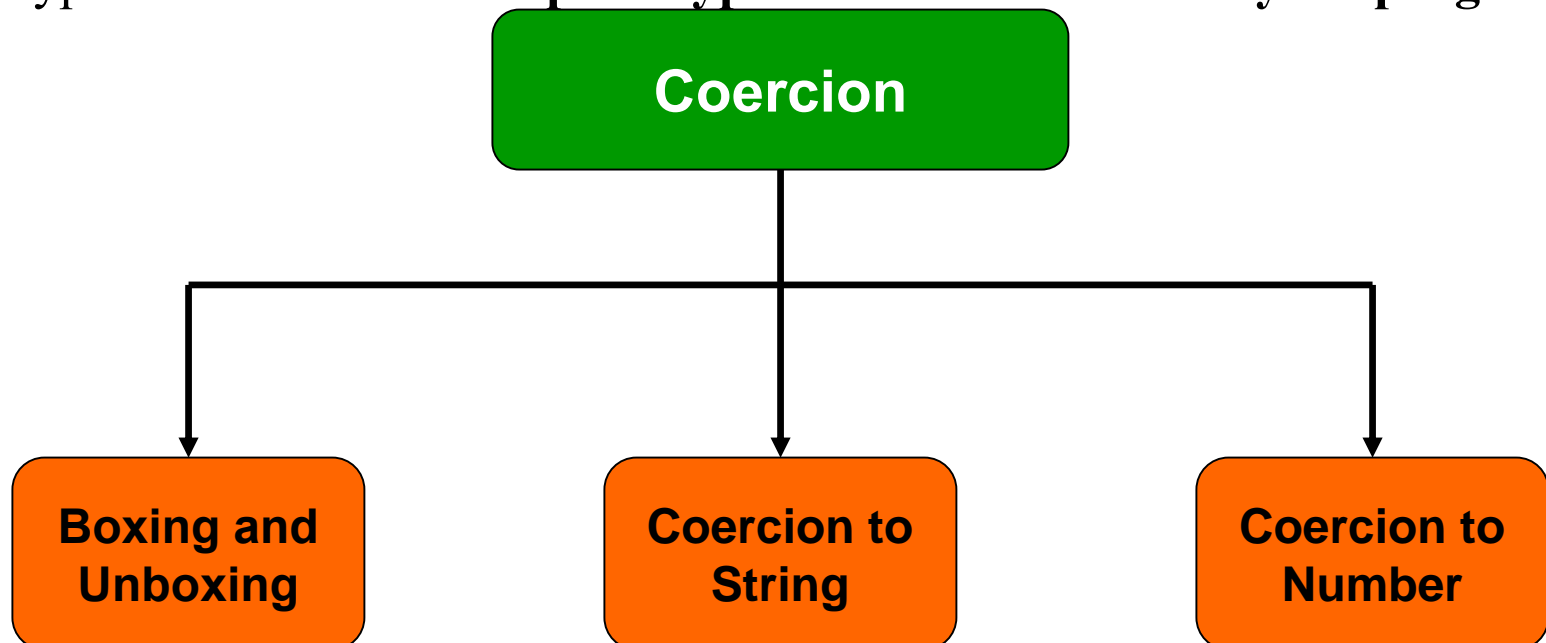
EL Function Demo

Name: KhanhKT - 33 - QTSC
 Name: KTKhanh - 43 - HCMC
 Name: KhanhKK - 23 - HCMC

Appendix

Coersion

- Means that the **parameters are converted to the appropriate objects or primitives automatically**
- EL defines appropriate **conversion with default values**
 - For example, a string parameter from a request will be coerced to the appropriate object or primitive.
- There is a **difference between type coercion and type conversion.**
 - **Coercion** is **implicit type conversion** and its usually **performed automatically by the compiler**
 - **Type conversion** is an **explicit type conversion** inserted by the **programmer**.



Appendix

Coersion

- **Boxing an Unboxing**

- **Boxing converts values of primitive type to corresponding values of reference type.**
 - If *i* is a **boolean** value, then boxing conversion **converts** *i* into a reference *r* or class and type **Boolean**, such that $r.value() = i$.
 - If *i* is a **byte** value, then boxing conversion **converts** *i* into a reference *r* of class and type **Byte**, such that $r.value() = i$
- **Unboxing converts values of reference type to corresponding values of primitive type**
 - If *r* is a **Boolean** reference, then unboxing **conversion converts** *r* into *v* of type **boolean**, such that $r.value() = v$.
 - If *r* is a **Byte** reference, then unboxing **conversion converts** *r* into a value *v* of type **byte**, such that $r.value() = v$.

- **Coercion to String**

- **A is String**, return **A**
- **A is null**, return **“”**
- **A.toString() throws exception**, return **error**. Otherwise return **A.toString()**

Appendix

Coersion

- **Coercion to Number**

- The rule to coerce a value to number type are If A is **null or “”**, return **0**
- A is **character** and is **converted to short**, developer apply following rules:
 - If A is **Boolean**, return **error**
 - If A is **number** type, return A
- A is **number**, coerce occurs quietly to type N using the following algorithm:
 - If N is **BigInteger**
 - If A is **BigDecimal**, return **A.toBigInteger()**
 - **Otherwise**, return **BigInteger.valueOf(A.longValue())**
 - If N is **BigDecimal**
 - If A is a **BigInteger**, return **new BigDecimal(A)**
 - **Otherwise**, return **new BigDecimal (A.doubleValue())**
 - If N is **Byte**, return **new Byte (A.byteValue())**
 - If N is **Short**, return **new Short (A.shortValue())**
 - If N is **Integer**, return **new Integer(A.intValue())**
 - If N is **Long**, return **new Long(A.longValue())**
 - If N is **Float**, return **new Float(A.floatValue())**
 - If N is **Double**, return **new Double(A.doubleValue())**
 - **Otherwise** return **error**

Appendix

Coersion – Example

`\${"2" + "2"} = ${"2" + "2"}`

`${"2" + "2"} = 4`

`\${"2" + "2"} = ${"2" + "2"}
`
`\${"2" + 2} = ${"2" + 2}`

`${"2" + "2"} = $ {"2" + "2"}`

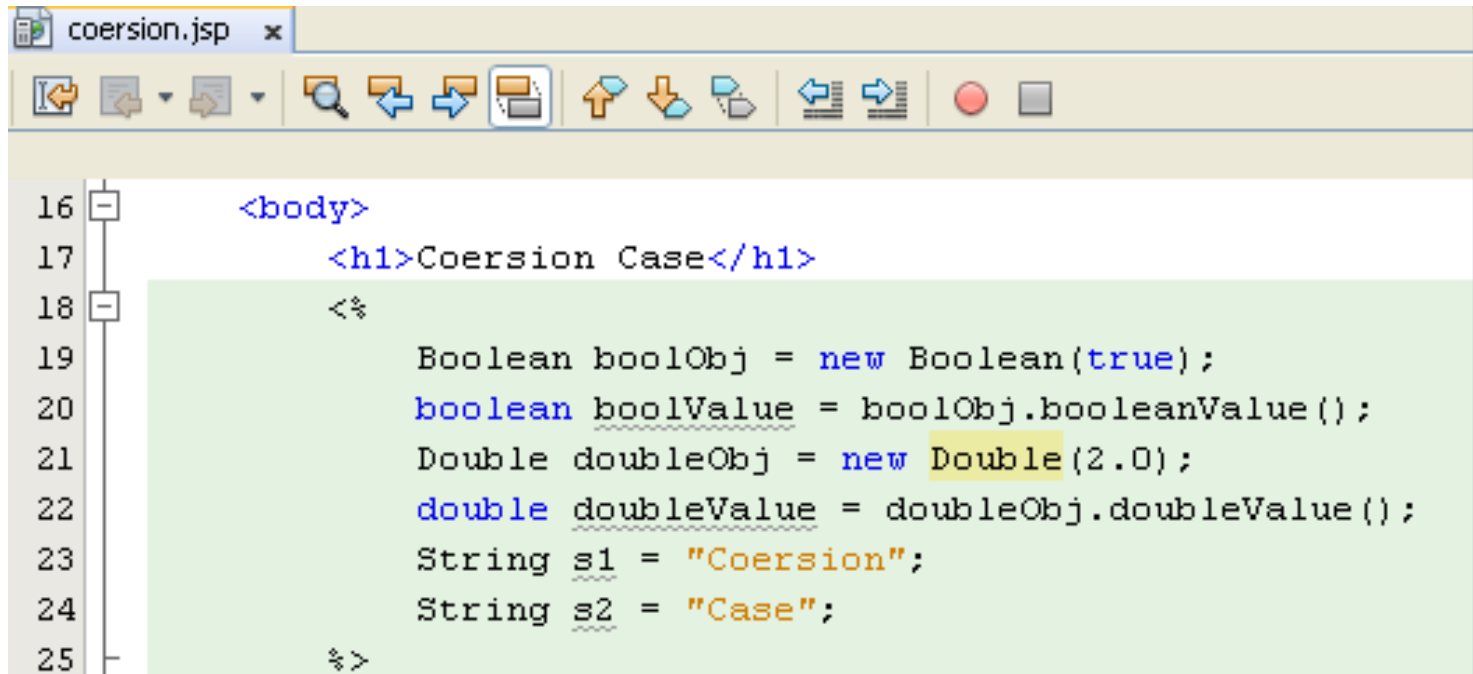
`${"2" + 2} = 4`

Appendix

Coersion

- Notes:

- **Variables declared in JSP page** (in scriptlet or declaration) **cannot be accessed** in JSP page or all of them can be presented with `${variable_name}`
- Therefore, if these **variables are forced in expression with EL**, the **value of these variables are converted to 0 (number) or false (boolean)**
- The EL operators' arithmetic or logical **is only applied to arithmetic or logic**. If the **value is not same type**, the **exception is thrown**
- The EL can **access the attribute in particular scope** (page, request, session, application)
- The **coersion** is **applied on the used operator in expression**



```

16 <body>
17   <h1>Coersion Case</h1>
18   <%
19       Boolean boolObj = new Boolean(true);
20       boolean boolValue = boolObj.booleanValue();
21       Double doubleObj = new Double(2.0);
22       double doubleValue = doubleObj.doubleValue();
23       String s1 = "Coersion";
24       String s2 = "Case";
25   %>

```


Appendix

Coersion – Example

```

26 Variable String concat: ${s1 + s2}<br/>
27 <% try { %>
28     Constant String concat: ${"abc" + "def"}<br/>
29 <% } catch (Exception e) {
30     out.print(e + "<br/>");
31 }
32 %>
33 ${doubleValue + doubleValue}<br/>
34 ${doubleObj + doubleObj}<br/>
35 ${boolObj && true}<br/>
36 ${doubleValue}<br/>
37 ${boolObj && doubleValue}<br/>
38 </body>
39 </html>

```

★ Favorites Coersion

Coersion Case

Variable String concat: 0

Constant String concat: java.lang.NumberFormatException: For input string: "abc"

0

0

false

false

Appendix

Coersion – Example

```

Source History
4      Author      : Trong Khanh
5      -->
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE html>
9      <html>
10     <head>
11         <meta http-equiv="Content-Type" content="text/html; charset=
12         <title>Coersion</title>
13     </head>
14     <body>
15         <h1>Coersion Demo</h1>
16         <%
17             pageContext.setAttribute("BOOLOBJ", new Boolean(true));
18             pageContext.setAttribute("STRING1", "khanhkt");
19             pageContext.setAttribute("STRING2", " AJ");
20             pageContext.setAttribute("DOUBLEOBJ", new Double(2.0));
21             pageContext.setAttribute("INTEGER", new Integer(2));
22         %>
23         Presenting the value using EL<br/>
24         ${BOOLOBJ}<br/>
25         ${STRING1}<br/>
26         ${DOUBLEOBJ}<br/>
27         ${INTEGER}<br/>

```

Appendix

Coersion – Example

```

28      Combination with operators in EL Expression<br/>
29      ${BOOLOBJ && BOOLOBJ}<br/>
30      ${BOOLOBJ && false}<br/>
31      ${DOUBLEOBJ + DOUBLEOBJ}<br/>
32      ${DOUBLEOBJ + 5.0}<br/>
33      ${DOUBLEOBJ + 5}<br/>
34      ${INTEGER + INTEGER}<br/>
35      ${INTEGER + DOUBLEOBJ}<br/>
36      ${INTEGER + 5}<br/>
37      ${INTEGER + 5.0}<br/>
38      <% try {%>
39          ${BOOLOBJ + DOUBLEOBJ}<br/>
40      <% } catch (Exception e) {
41          out.print(e + "<br/>");
42      }
43      %>
44      <% try {%>
45          ${STRING1 + STRING2}<br/>
46      <% } catch (Exception e) {
47          out.print(e + "<br/>");
48      }
49      %>
50      </body>
51      </html>

```

Coersion – Example

