Fpt University

JSP Tag Libraries

Tag Libraries
JSTL
The “Classic” Custom Tag Event Model
Custom Tag

#JSTL #CustomTag #Dynamic

Lê Võ Minh Thư

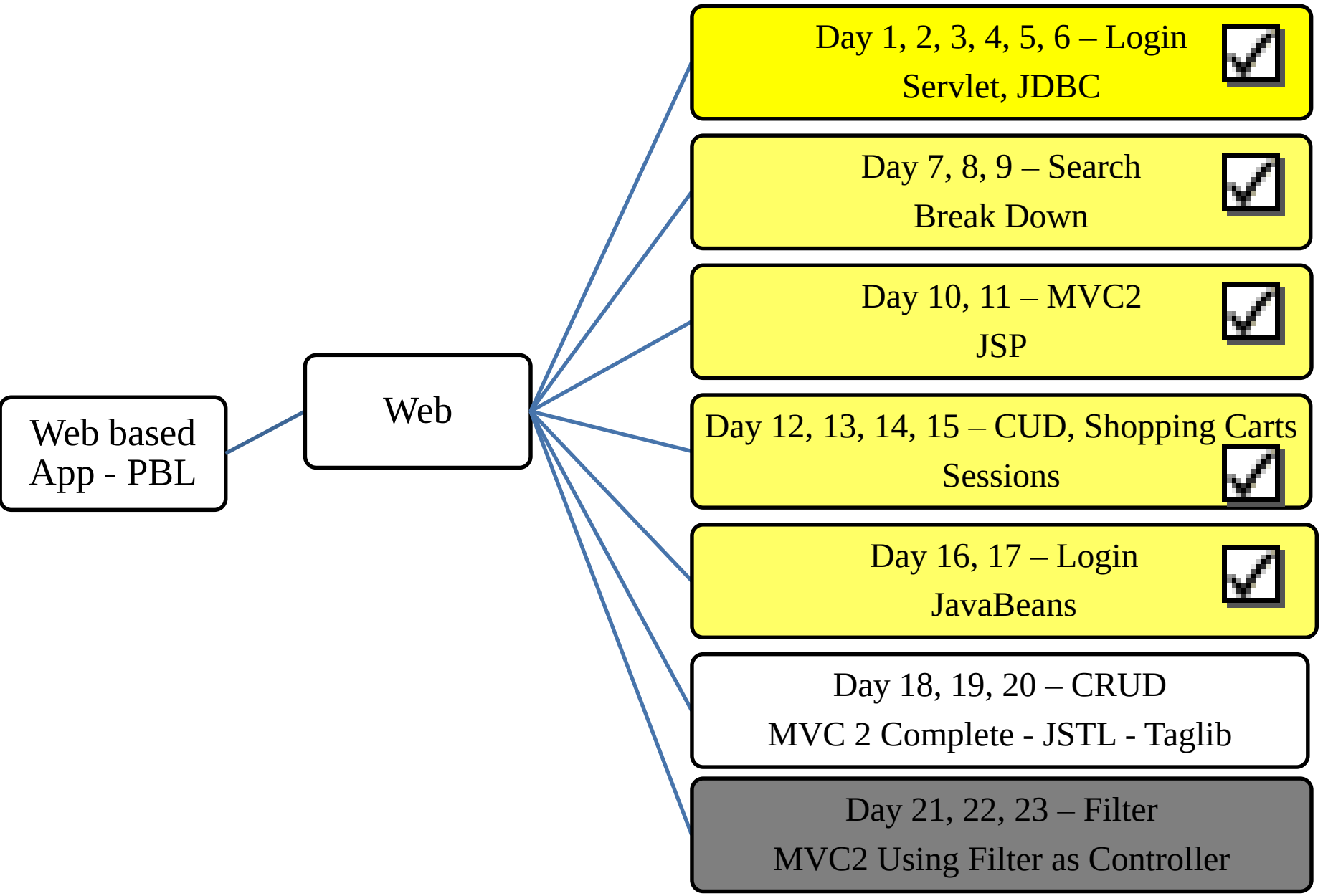
Review

- **JSP Standard Actions**
  - jsp:method attribute
- **Java Beans**
  - jsp:useBean, jsp:setPropety, jsp:getProperty
  - Scopes: page, request, session, application
- **Dispatching Mechanisms**
  - jsp:include, jsp:forward, jsp:param
- **Expression Language – EL**
  - ${EL Expression}
  - Operators, Implicit Objects, Scope Variables
  - Functions
  - Coersions

Objectives

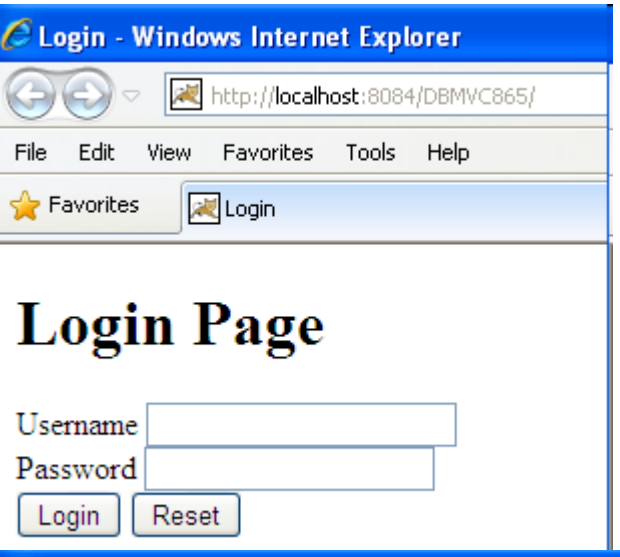
- **How to remove all java code in JSP (View)? Complete the MVC 2 Design Pattern with View**
  - **JSTL**
- **How to build the data grid tag library using in JSP?**
  - **Tag Libraries**
    - Model
    - Classical, Simple, and Handles
    - How to implement the custom Tag Lib and use it in JSP

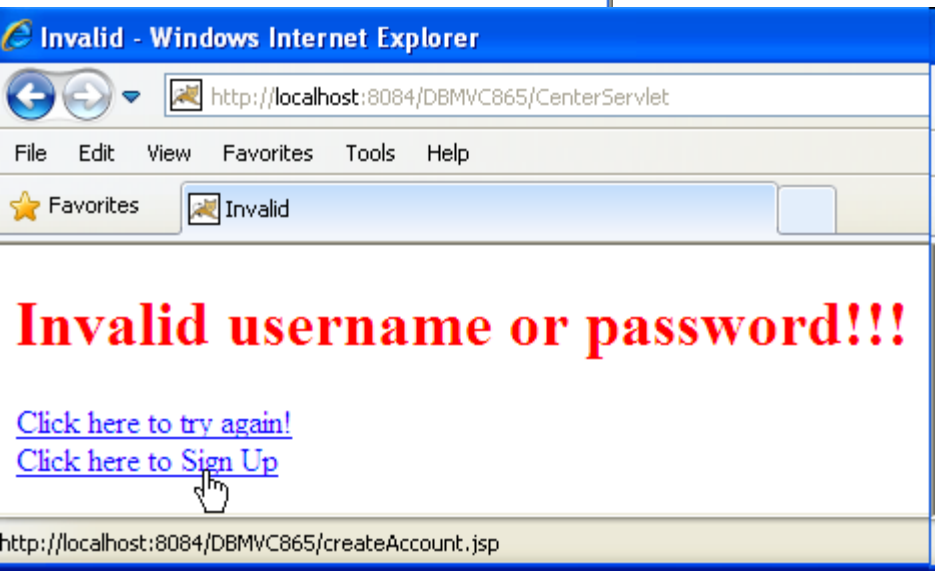
Objectives


Web based App - PBL
Web
Day 1, 2, 3, 4, 5, 6 – Login
Servlet, JDBC
Day 7, 8, 9 – Search
Break Down
Day 10, 11 – MVC2
JSP
Day 12, 13, 14, 15 – CUD, Shopping Carts
Sessions
Day 16, 17 – Login
JavaBeans
Day 18, 19, 20 – CRUD
MVC 2 Complete - JSTL - Taglib
Day 21, 22, 23 – Filter
MVC2 Using Filter as Controller

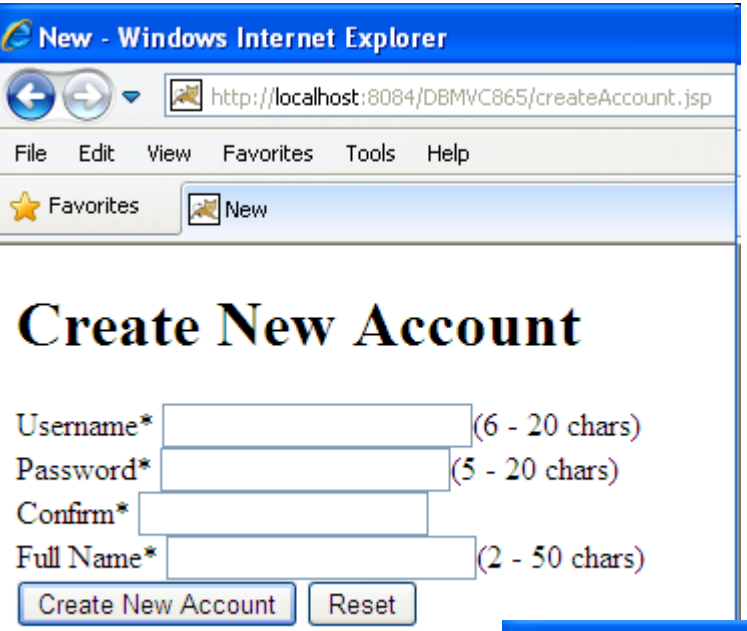
Complete MVC 2
Requirements

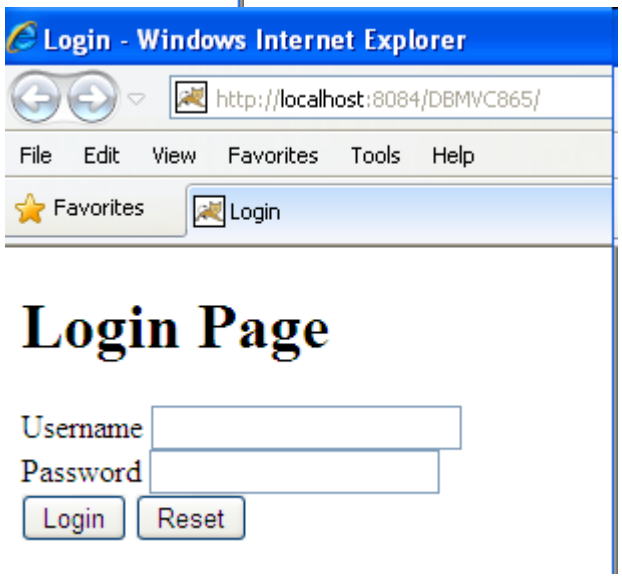
- Complete the MVC 2 with CRUD Web Application in previous topic

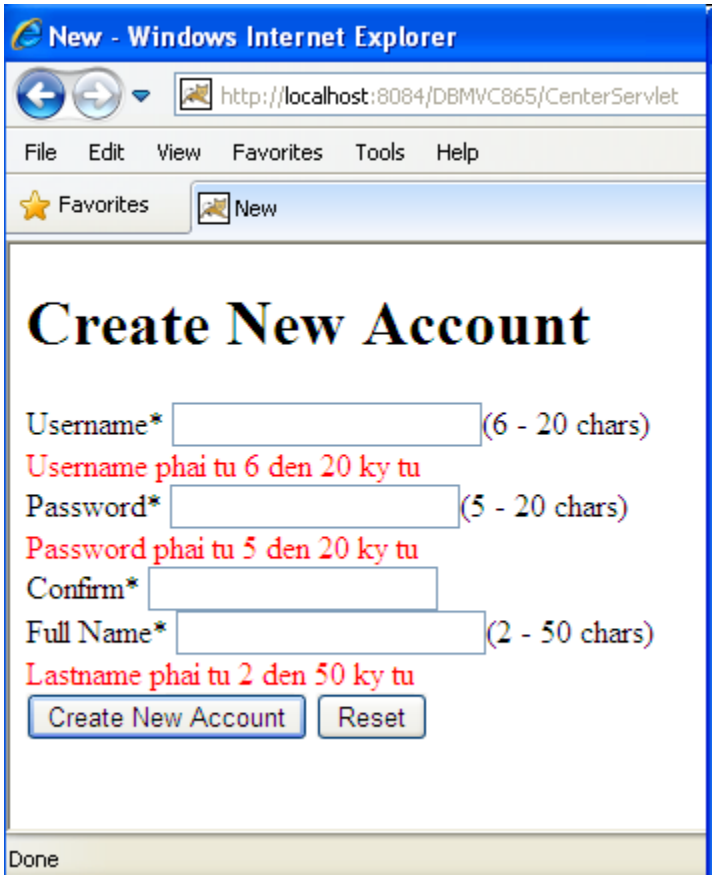




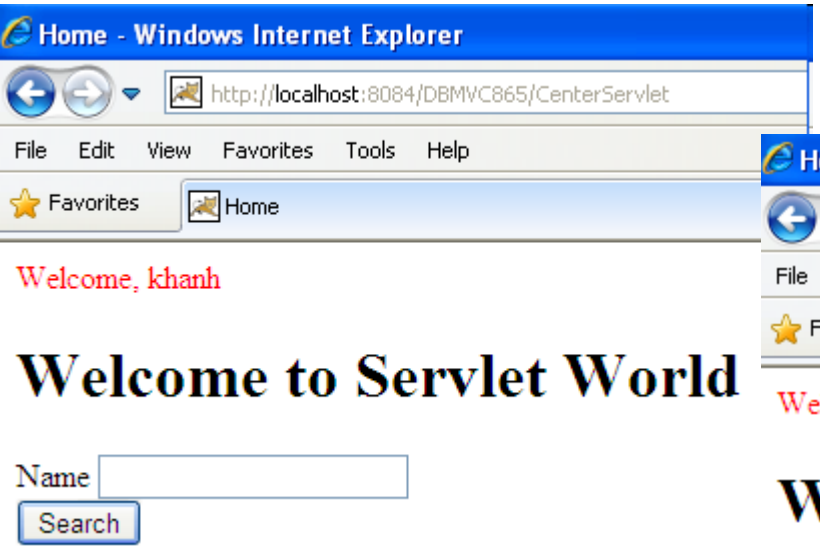
Complete MVC 2
Requirements



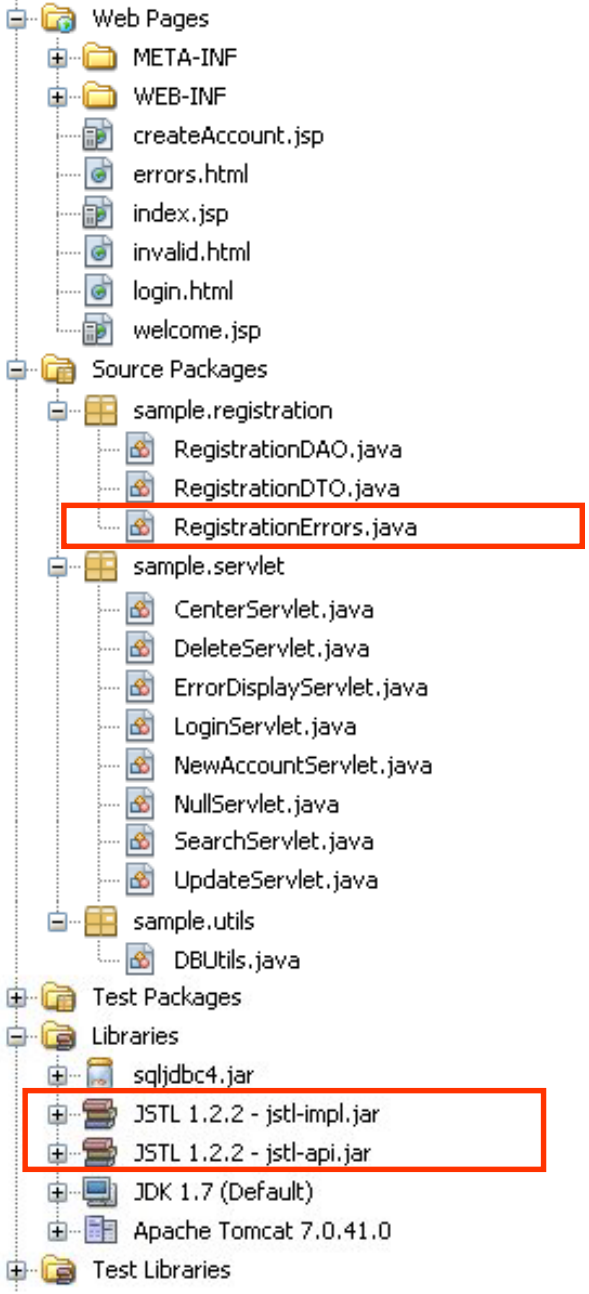




Complete MVC 2
Requirements



Complete MVC 2
Expectation

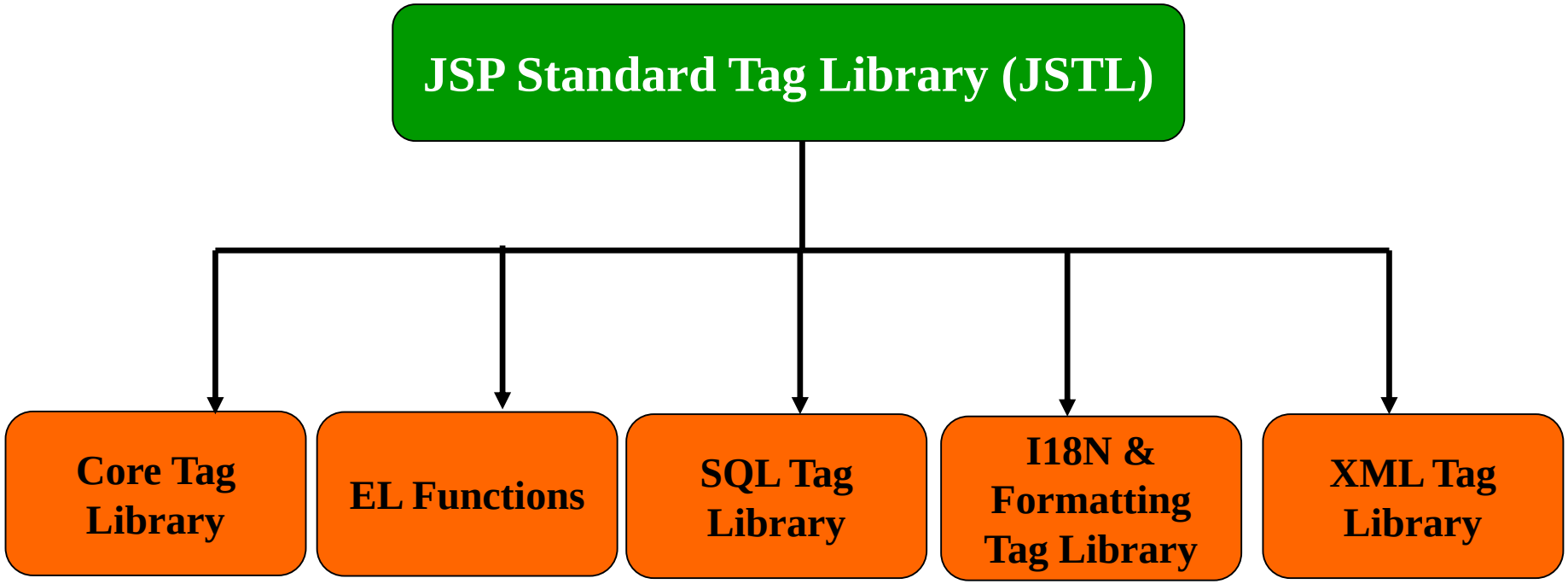


JSTL
JSP Standard Tag Library

- Is a new component being offered by the Sun for JSP programming.
- A library of **predefined tags** (by Sun) that provides **a set of reusable standard tags** that works in the similar manner everywhere
- Allows programming **using tags rather than scriptlet code** JSTL has reusable standard set of tags.
- Provides the user with a script-free environment
- Are **easier for non-programmers & inexperienced programmers**
- **Disadvantages**
  - It adds **processing overhead** to the server. The **tag** libraries are **compiled into** a **servlet**, which is **then executed by** the servlet **container**. The server is added with much more code by the JSTL.
  - JSTL is also **less powerful than the scriptlets** as embedded java codes can do everything in JSP pages

JSTL

Type of Tag Library



Fpt University

JSTL

Core Tag Library

- The library contains the tags for **looping, expression evaluation, handle flow controls, and basic input and output.**
- It can be **declared** by

<%@ taglib prefix="c" uri= "http://java.sun.com/jsp/jstl/core" %>

- **General Purpose Tags**
  - Are used to **set, remove and display variable values** that are created within a JSP page.
  - The core tag library contains tags for **getting, setting and displaying attribute values**.

JSTL

Core Tag Library – General Purposes

Tags	Descriptions
<c:set>	- **<c:set var="varName" value="value/exp"** **scope="page|request|session|application" />** - **Assigns or update a value to a variable in scope** - **Similar** to set attribute to some scope using **setAttribute**() method - **Ex**: <c:set var="simple" value="${2+1}" scope="session"/> - Similar to <% session.setAttribute("simple", new Integer(2+1)) %> - The var of set can be **accessed using** ${varName} on page, or **${scope.varName}** on page or other pages, or <%= scope.getAttribute("varName") %>
<c:remove>	- **<c:remove var="varName"** **scope="page|request|session|application" />** - **Remove** a scope variable. This is an **empty tag** - **Similar** to remove attribute from some scope using **removeAttribute**() method - **Ex** : <c:remove var="simple" scope="session"/> - Similar to <% session.removeAttribute("simple") %>

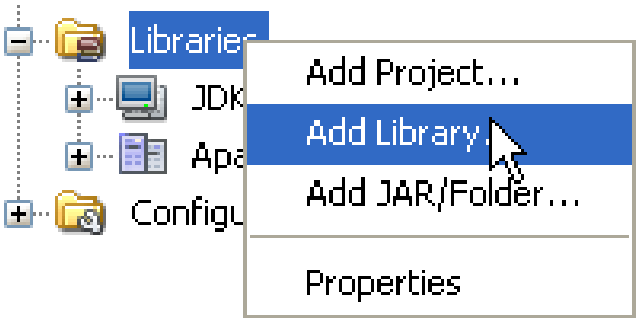
JSTL

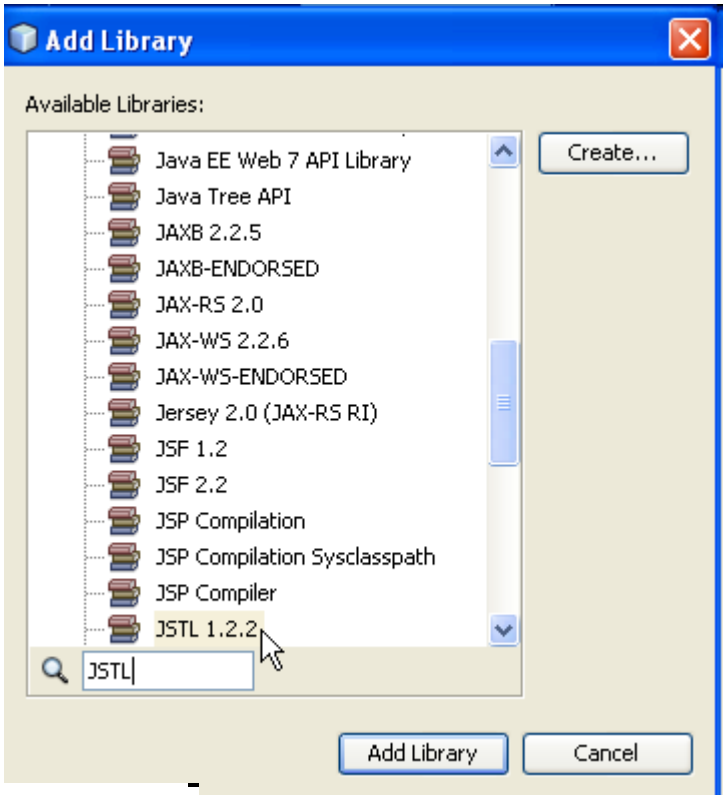
Core Tag Library – General Purposes

Tags	Descriptions
<c:out>	- **<c:out value="value|expression" escapeXml="true|false" default="defaultValue" />** - **Evaluate** an **expression** & **store** the **result** in **the current JspWrite object** - Default value if the resulting value is null - Similar to print out the data using out object in JSP Implicit Object - **Ex**: <c:out value="${simple}"/> similar to <%= pageContext.getAttribute(simple) %>
<c:catch>	- **<c:catch [var="varName"]>…</c:catch>** - **Provides** an **exception handling functionality**, such as **try-catch**, inside JSP pages **without using scriptlets**

JSTL
Requirement

- Required Library for JSTL
  - JSTL 1.2.2 (jstl-impl.jar, jstl-api.jar)



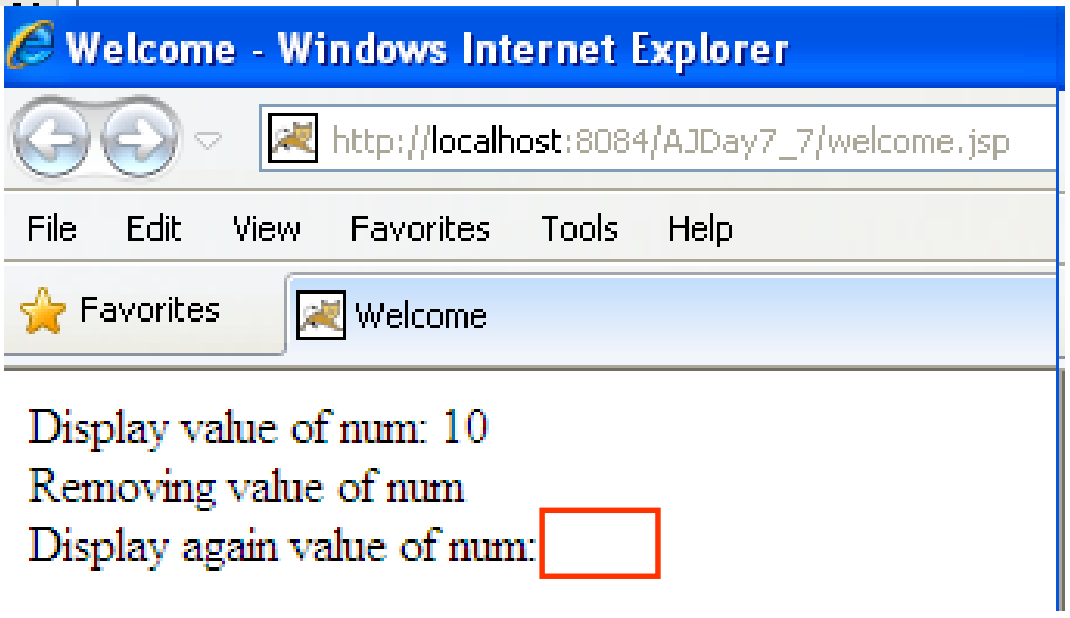


- Choose JSTL 1.2.2

JSTL

Core Tag Library – Example

```
10  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
11  <html>
12      <head>
13          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14          <title>Welcome</title>
15      </head>
16      <body>
17          <c:set var="num" value="10"/>
18          Display value of num: <c:out value="${num}"/><br/>
19          Removing value of num <c:remove var="num"/><br/>
20          Display again value of num: <c:out value="${num}"/><br/>
```



JSTL

Core Tag Library – Example

```
<c:set var="addInfo" value="INFO"/>
${addInfo} <br/>
<%= request.getAttribute("addInfo")%> <br/>
```

INFO
null

```
<c:set var="addInfo" value="INFO" scope="request"/>
${addInfo} <br/>
<%= request.getAttribute("addInfo")%> <br/>
```

INFO
INFO

```
<c:set var="addInfo" value="INFO" scope="session"/>
${addInfo} <br/>
<%= request.getAttribute("addInfo")%> <br/>
${requestScope.addInfo}<br/>
<%= session.getAttribute("addInfo") %><br/>
${sessionScope.addInfo}
```

INFO
null

INFO
INFO

JSTL

Core Tag Library – Decision Making

- **Support conditions** in a JSP page
- Are necessary as the contents or the output of the JSP page is often conditional based on the value of the dynamic application data

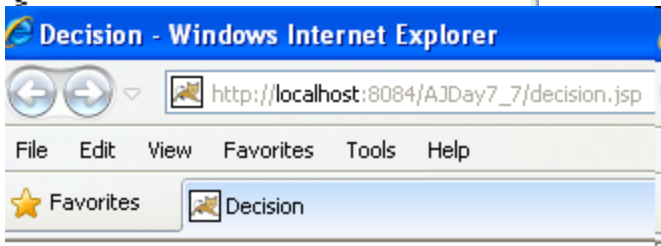
Tags	Descriptions
<c:if>	- **<c:if test =“testcondition” var=“varName” scope=“page | request | session | application”>…</c:if>** - Is used for **conditional execution** of the code - Is a container tag that allows the execution of the body if the test attribute evaluates to true
<c:choose>	- **<c:choose> <c:when test=”cond”>...</c:when> <c:otherwise>…</c:otherwise> </c:choose>** - Is similar to the **switch statement** in Java (multiple conditions) - **Performs conditional block execution (replace to c:if)** - Multiple <c:when> tags can be embedded in a <c:choose> tag - If none of the conditions evaluates to true, then the body of <c:otherwise> tag is processed.

JSTL
Core Tag Library – Example

decision.jsp

Source | History

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <!DOCTYPE html>
9   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
10  <html>
11      <head>
12          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13          <title>Decision</title>
14      </head>
15      <body>
16          <h1>Decision Demo</h1>
17          <c:set var="num1" value="20" scope="page" />
18          <c:set var="num2" value="60" scope="page" />
19          <c:if test="${num1 < num2}">
20              Num1 is less than num 2
21          </c:if>
22      </body>
23  </html>
```



JSTL

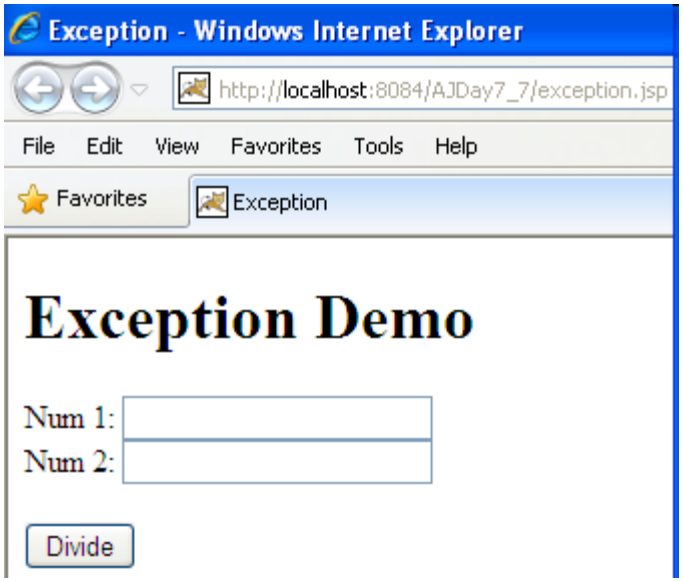
Core Tag Library – Example

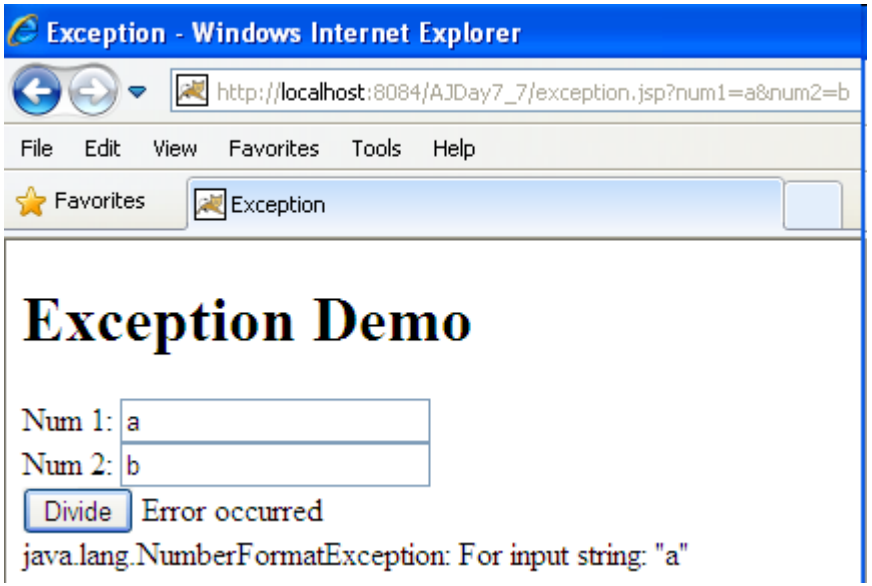
exception.jsp

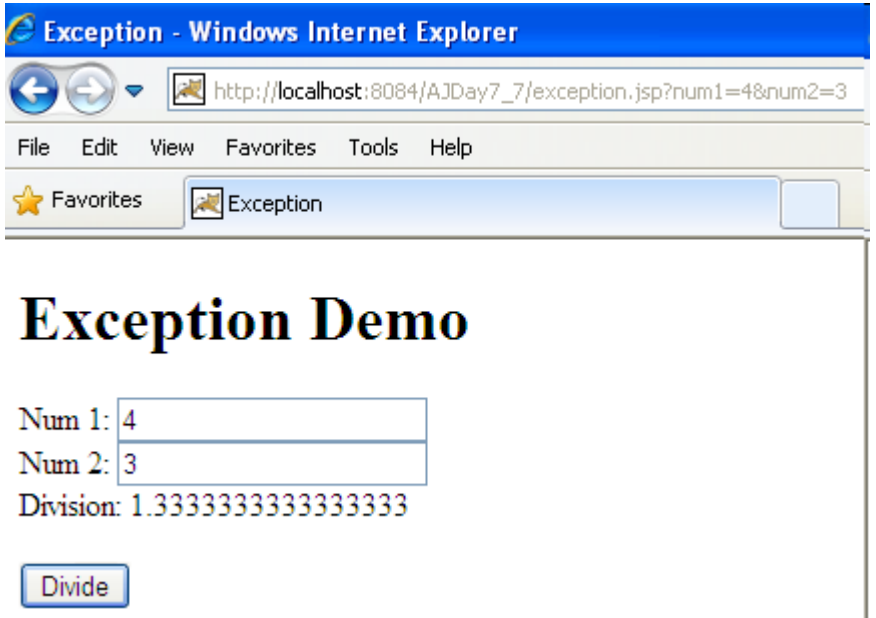
```
6   <%@page contentType="text/html" pageEncoding="UTF-8"%>
7   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
8       "http://www.w3.org/TR/html4/loose.dtd">
9   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
10  <html>
11      <head>
12          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13          <title>Exception</title>
14      </head>
15      <body>
16          <h1>Exception Demo</h1>
            <form>
18              Num 1: <input type="text" name="num1" value="${param.num1}" /><br/>
19              Num 2: <input type="text" name="num2" value="${param.num2}" /><br/>
20              <c:catch var="ee">
21                  <c:if test="${not empty param.num1 and not empty param.num2}">
22                      <c:set var="division" value="${param.num1 / param.num2}" />
23                      Division: <c:out value="${division}" /><br/>
24                  </c:if><br/>
25              </c:catch>
26
27              <input type="submit" value="Divide" />
28              <c:if test="${not empty ee}">
29                  Error occurred<br/>
30                  <c:out value="${ee}" /><br/>
31              </c:if>
32          </form>
33      </body>
34  </html>
```

JSTL

Core Tag Library – Example







JSTL

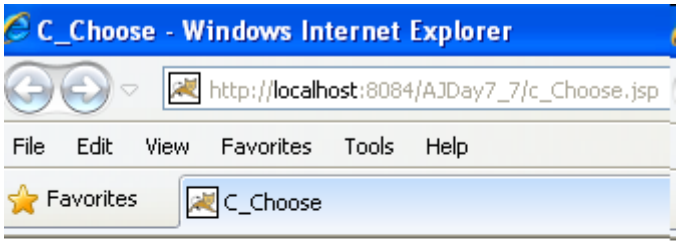
Core Tag Library – Example

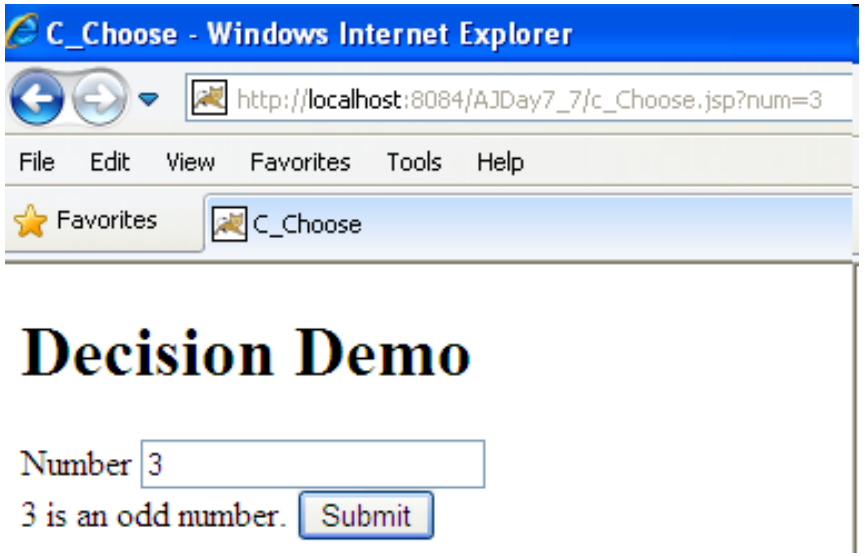
c_Choose.jsp

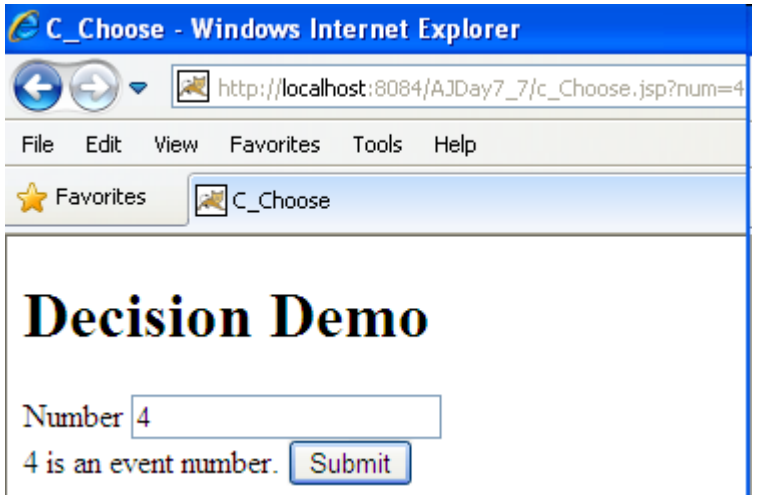
```
8   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9       "http://www.w3.org/TR/html4/loose.dtd">
10  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
11  <html>
12      <head>
13          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14          <title>C_Choose</title>
15      </head>
16      <body>
17          <h1>Decision Demo</h1>
            <form>
19              Number <input type="text" name="num" value="${param.num}" /><br/>
20              <c:choose >
21                  <c:when test="${empty param.num}">
22                  </c:when>
23                  <c:when test="${param.num%2 != 0}">
24                      ${param.num} is an odd number.
25                  </c:when>
26                  <c:when test="${param.num%2 == 0}">
27                      ${param.num} is an event number.
28                  </c:when>
29                  <c:otherwise>
30                  </c:otherwise>
31              </c:choose>
32              <input type="submit" value="Submit" />
33          </form>
34      </body>
35  </html>
```

JSTL

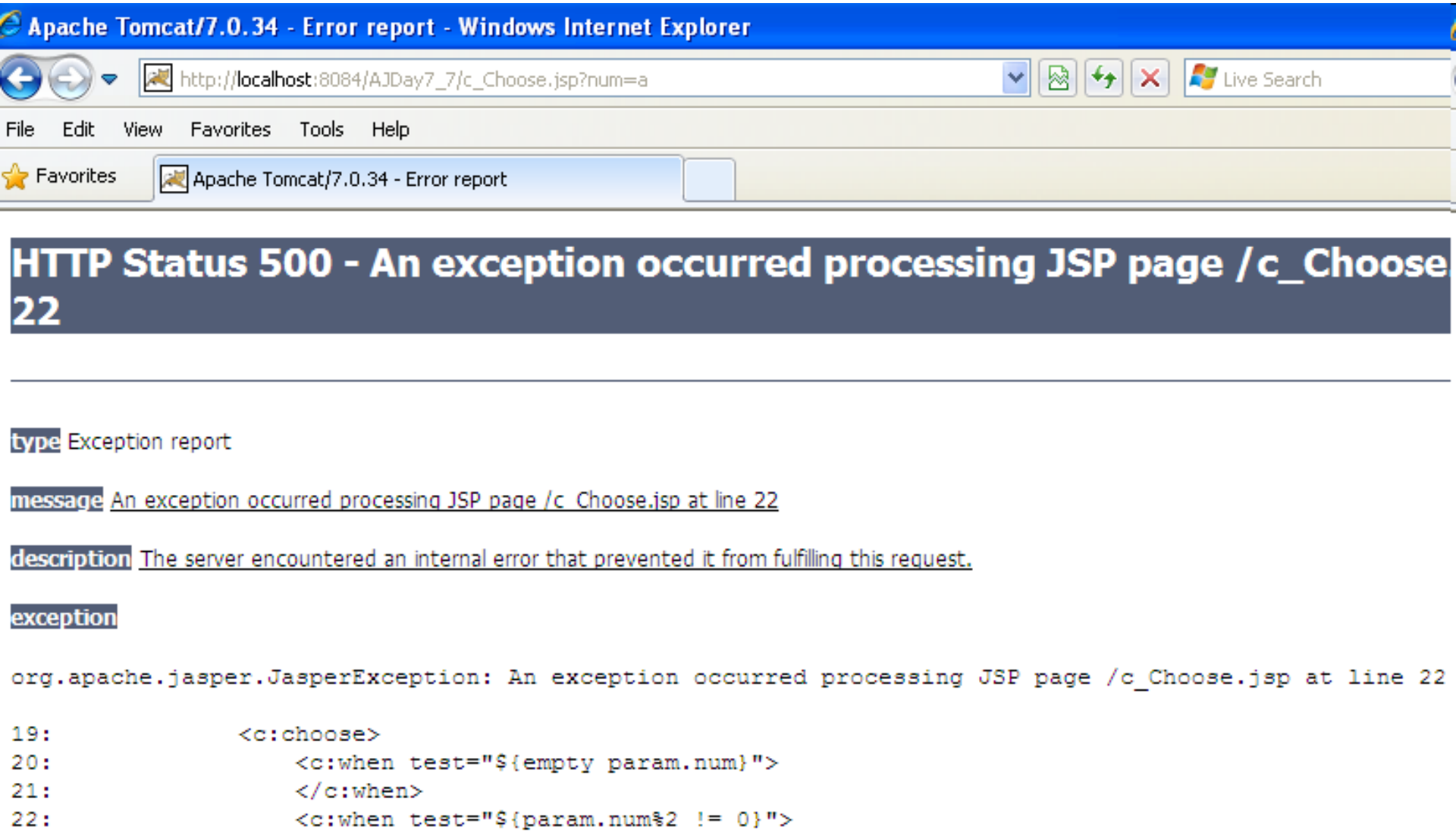
Core Tag Library – Example





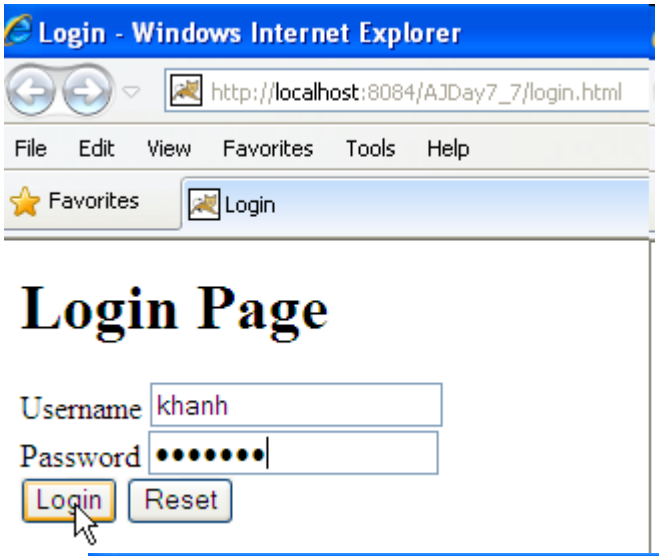


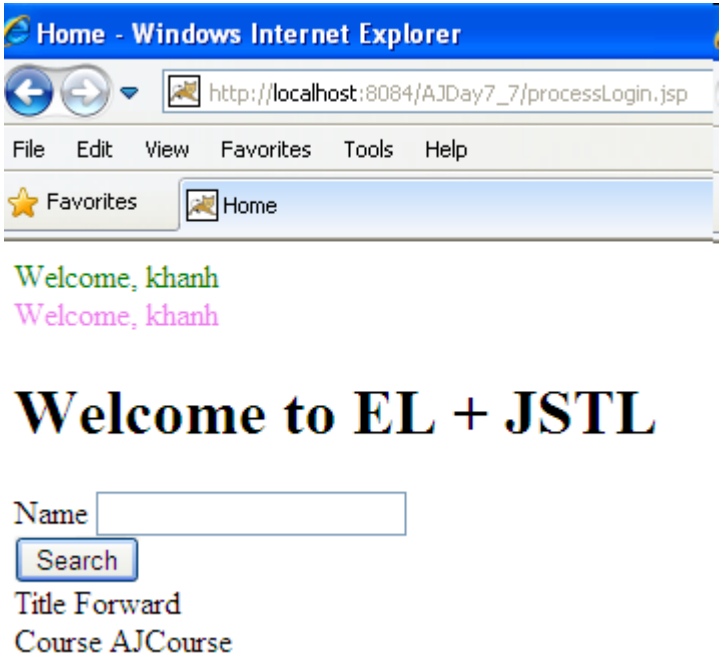
JSTL
Core Tag Library – Example

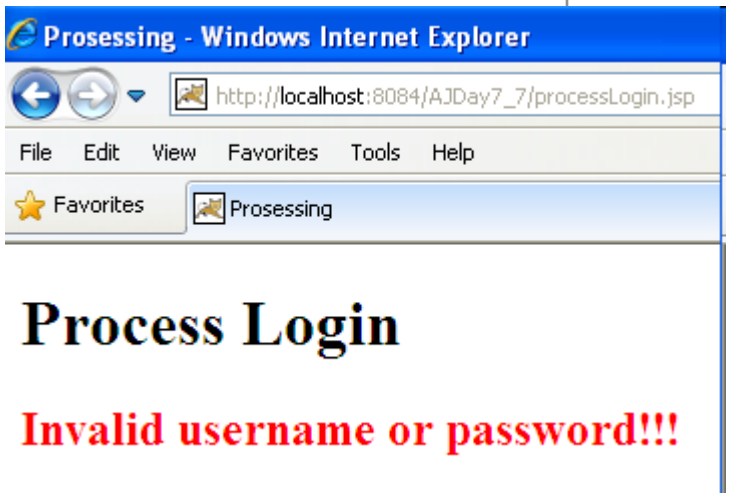


JSTL
Example

- Improve the 6th topic using EL combining with JSTL







JSTL
Example

login.html

Source | History

```
-->
<!DOCTYPE html>
<html>
    <head>
        <title>Login</title>
    </head>
    <body>
        <h1>Login Page</h1>
        <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>
    </body>
</html>
```

JSTL
Example

processLogin.jsp

Source | History

```
4       Author      : Trong Khanh
5   --%>
6   <%@page contentType="text/html" pageEncoding="UTF-8"%>
7   <!DOCTYPE html>
8   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
9   <html>
10      <head>
11          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12          <title>Prosessing</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="*" />
18          <c:if test="${loginAtt.checkLogin()}">
19              <jsp:forward page="home.jsp">
20                  <jsp:param name="title" value="Forward"/>
21                  <jsp:param name="course" value="AJCourse"/>
22              </jsp:forward>
23          </c:if>
            <h2><font color="red">Invalid username or password!!!</font> </h2>
25      </body>
26  </html>
```

JSTL

Core Tag Library – Iterations

- Is required for performing **looping function**
- The object can be **retrieved from a collection** in the JavaBeans component and assigned to a scripting variable by using iteration tags

Tags	Descriptions
<c:forEach>	- **<c:forEach var="varName" items="collection" begin="begin" end="end" step="step" varStatus="internalLoopobject">…</c:forEach>** - Is used **repeat** the body content over a **collection of objects** - Will continue for a number of times specified by the user in the code - **varStatus's methods**: **count** (int), **current** (obj), **index** (int)
<c:forTokens>	- **<c:forTokens items="stringofToken" delims="delimiters" var="varName" >...</c:forTokens>** - Is used to **iterate** over a **collection of tokens separated by user-specified delimiters** - It is a container tag

JSTL

Core Tag Library – Example

iterator.jsp

Source | History

```
4       Author      : Trong Khanh
5   --%>
6   <%@page contentType="text/html" pageEncoding="UTF-8"%>
7   <!DOCTYPE html>
8   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
9   <html>
10      <head>
11          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"
12          <title>Iterator</title>
13      </head>
14      <body>
15          <h1>Iterator Demo</h1>
16          <c:set var="language" value="Core Java:Servlet;JSP:EJB" scope="page"/>
17          <%! String[] names = {"Sun Microsystems", "Microsoft", "Oracle", "IBM"};%>
18          <b>Company</b><br/>
19          <c:forEach var="company" items="<%= names%>" >
20              <c:out value="${company}" /><br/>
21          </c:forEach> <hr/>
22          <b>Language</b><br/>
23          <c:forTokens items="${language}" delims=":;" var="lang">
24              <c:out value="${lang}"/><br/>
25          </c:forTokens> <hr/>
26          <b>Counter</b>
27          <c:forEach begin="1" end="10" step="1" var="counter">
28              <c:out value="${11 - counter}"/>
29          </c:forEach>
30      </body>
31  </html>
```

Iterator Demo

Company
Sun Microsystems
Microsoft
Oracle
IBM

Language
Core Java
Servlet
JSP
EJB

Counter 10 9 8 7 6 5 4 3 2 1

JSTL

Core Tag Library – URL-Related Actions

- Are used to import resources from a **given URL, re-encode URLs, redirect to URLs, as well as pass additional request parameters where necessary**

Tags	Descriptions
<c:import>	- **<c:import url="url of resource import" var="varName" scope="scopeName">…</c:import>** - **Imports** the **content** of a **URL-based resource**. - Action main include nested <**c:param**> **tags** to **specify the query string** (unless the varReader attribute is specified). - **Ex**: <c:import var="xml" url="WEB-INF/abc.xml"/>
<c:url>	- <c:url var="varName" value="path" scope="scopeName" >...</c:url> - **Builds** a **URL** with the proper rewriting rules applied (**only relative URLs are rewritten**). - Action may include nested <c:param> tags to specify the query string - **Ex**: <c:url var="addr" value="ProcessServlet"/> click here

Fpt University

JSTL

Core Tag Library – URL-Related Actions

Tags	Descriptions
<c:redirect>	- **<c:redirect url="url path">…</c:redirect>** - Sends the **client** a **response** to **redirect** to the specified URL. This action willabort processing of the current page. - Action may include **nested <c:param> tags** to specify the query string. - **Ex**: <c:redirect url="abc.jsp"/>
<c:param>	- **<c:param name="nameOfQuery" value="ValueOfParameter">...</c:param>** - **Adds request parameters** to a URL. - This action can only be **nested within <c:import>, <c:url>, or <c:redirect>**

JSTL

Core Tag Library – URL-Related Actions

urlJSTL.jsp

Source | History

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <!DOCTYPE html>
9   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
10  <html>
11      <head>
12          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13          <title>URL</title>
14      </head>
15      <body>
16          <h1>JSTL URL Relating Demo</h1>
17          Demo url in href<br/>
18          <c:url var="nextLink" value="nextPage.jsp">
19              <c:param name="user" value="khanh"/>
20          </c:url>
21          <a href="${nextLink}">Click here to go to next page</a>
22
23      </body>
24  </html>
```

Same as “nextPage.jsp?user=khanh

JSTL

Core Tag Library – URL-Related Actions

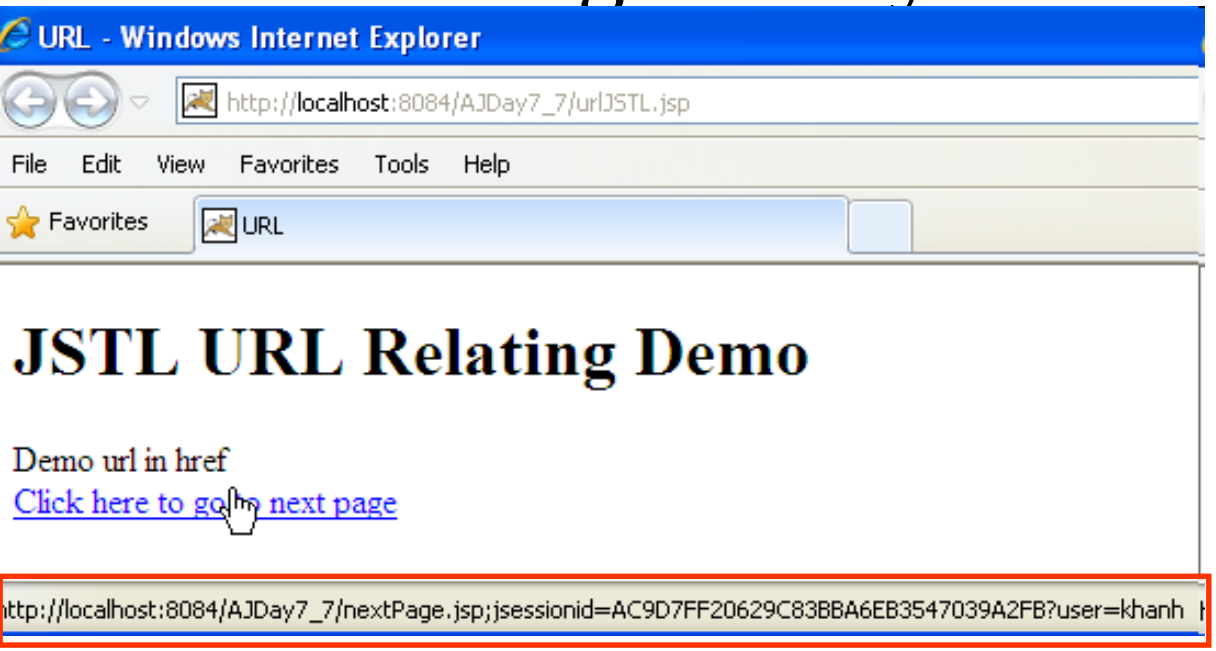
nextPage.jsp

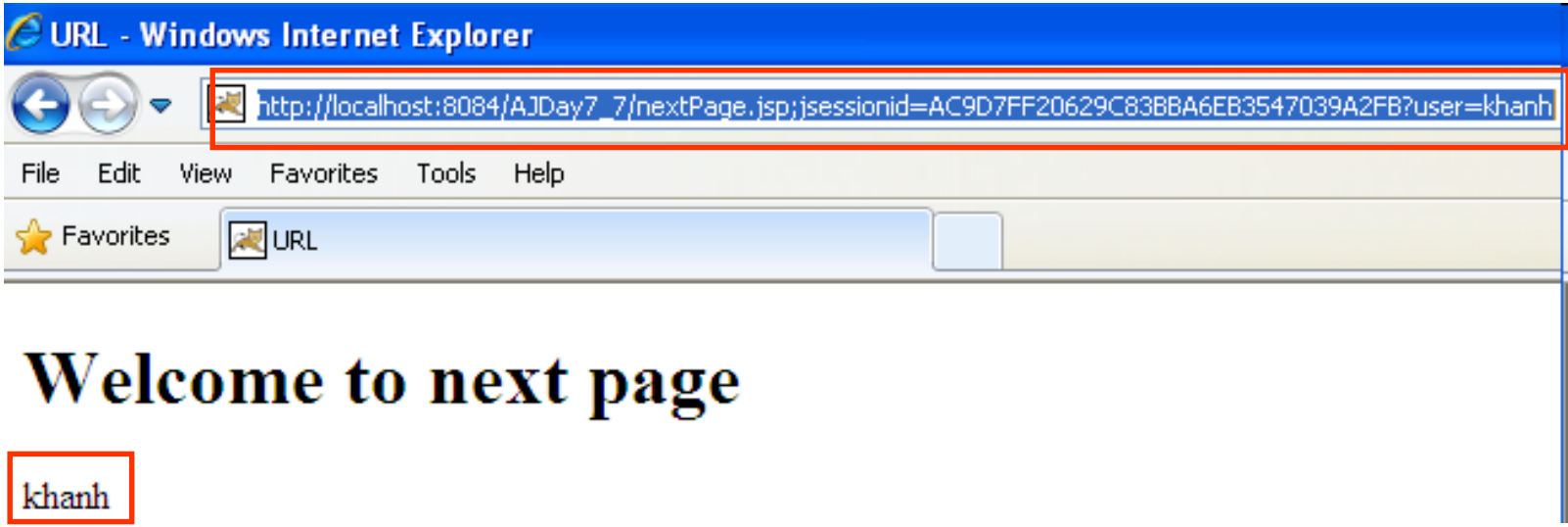
Source | History

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <!DOCTYPE html>
9   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
10  <html>
11      <head>
12          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13          <title>URL</title>
14      </head>
15      <body>
16          <h1>Welcome to next page</h1>
17          ${param.user}
18      </body>
19  </html>
```

JSTL

Core Tag Library – URL-Related Actions

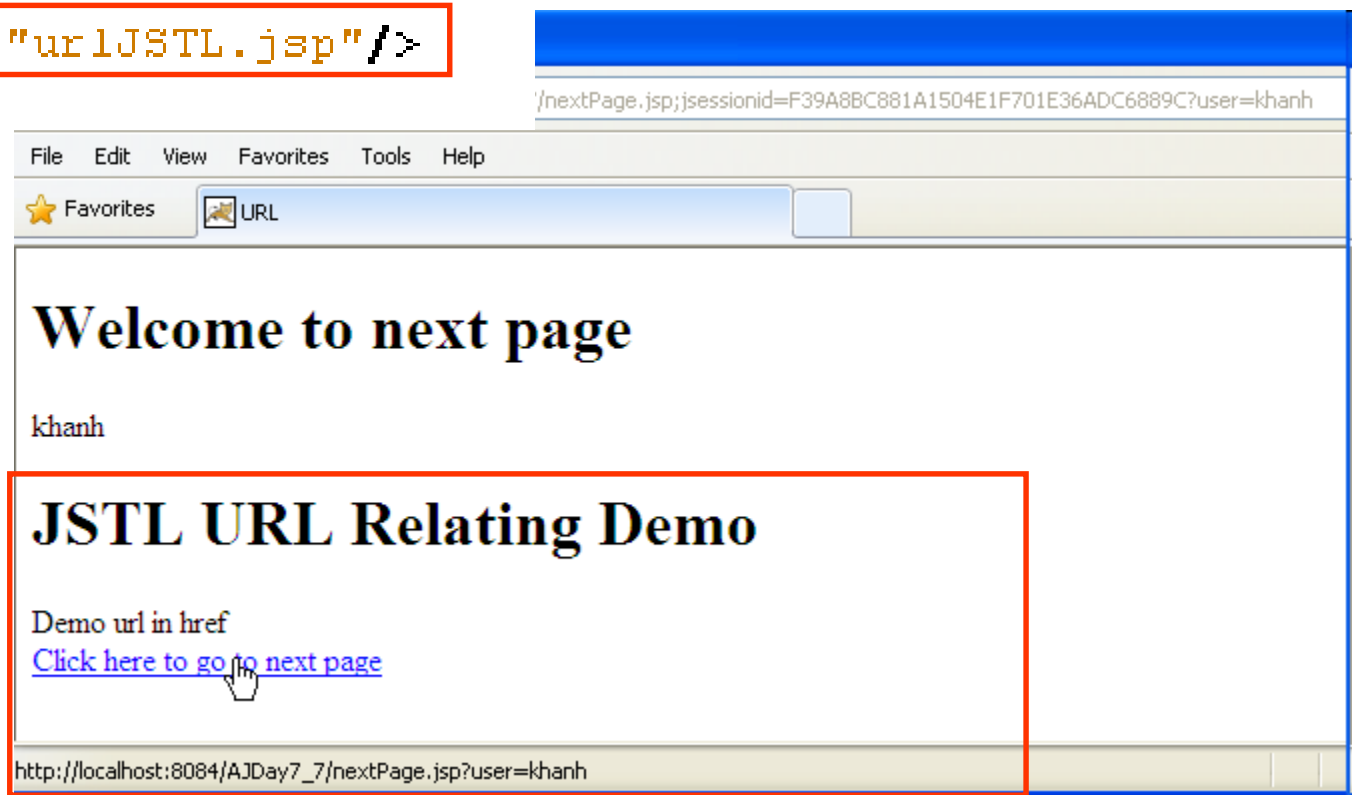




JSTL

Core Tag Library – URL-Related Actions

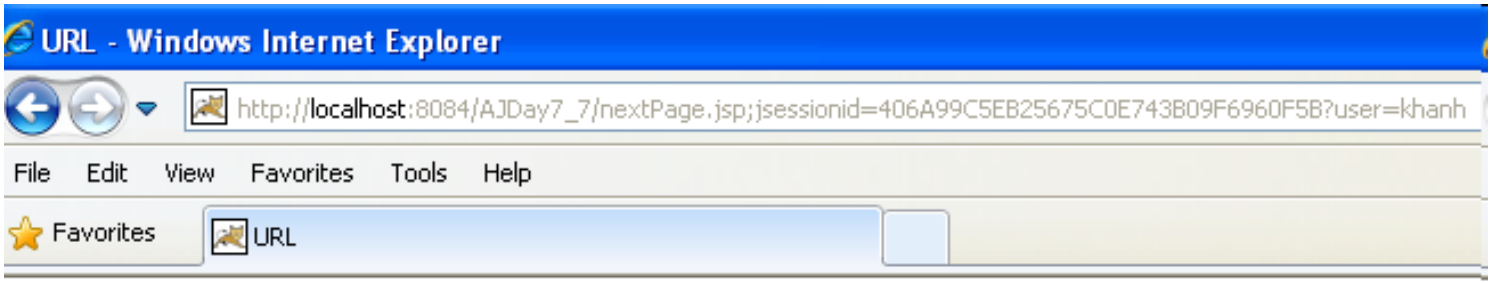
```
    <title>URL</title>
</head>
<body>
    <h1>Welcome to next page</h1>
    ${param.user}

    <c:import url="urlJSTL.jsp"/>
</body>
```



JSTL

Core Tag Library – URL-Related Actions

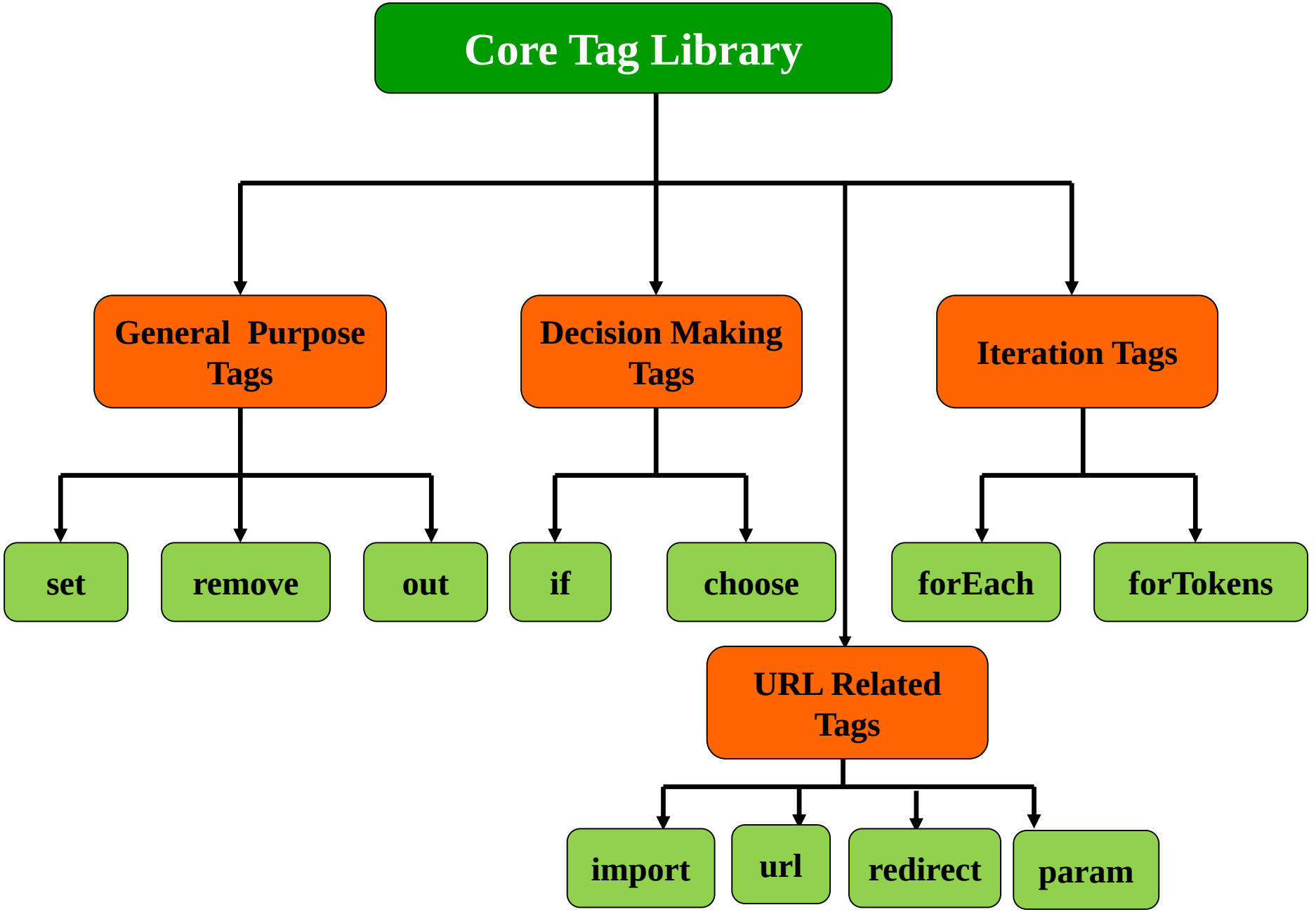
```
    <title>URL</title>
</head>
<body>
    <h1>Welcome to next page</h1>
    ${param.user}

    <c:import var="s" url="urlJSTL.jsp"/>
</body>
```



Welcome to next page

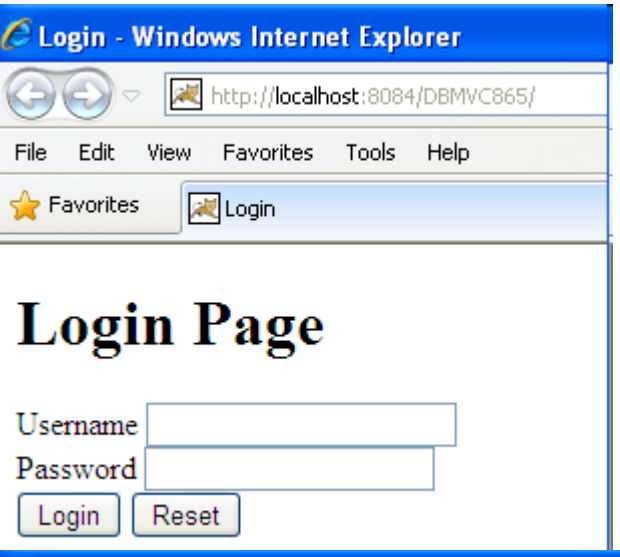
khanh

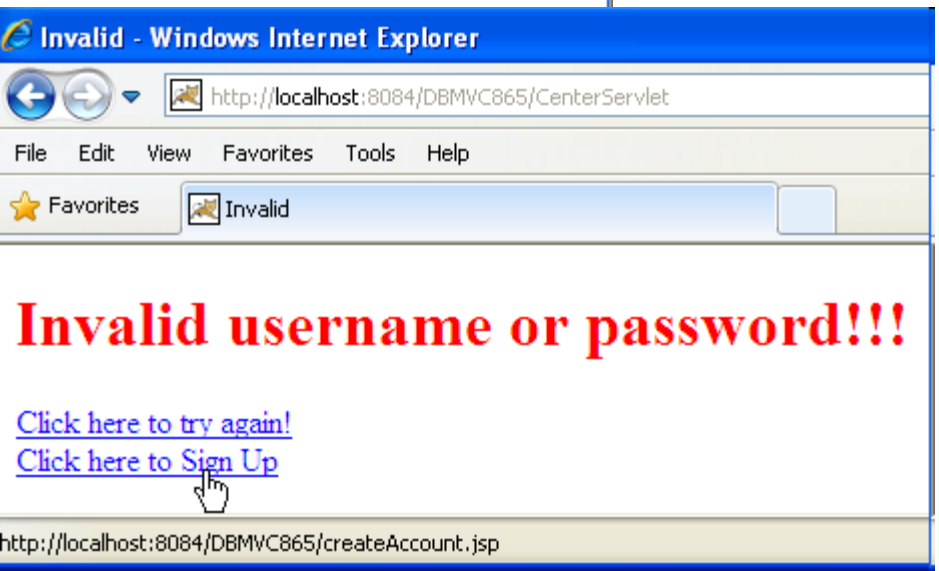
JSTL Core Tag Library – Summary



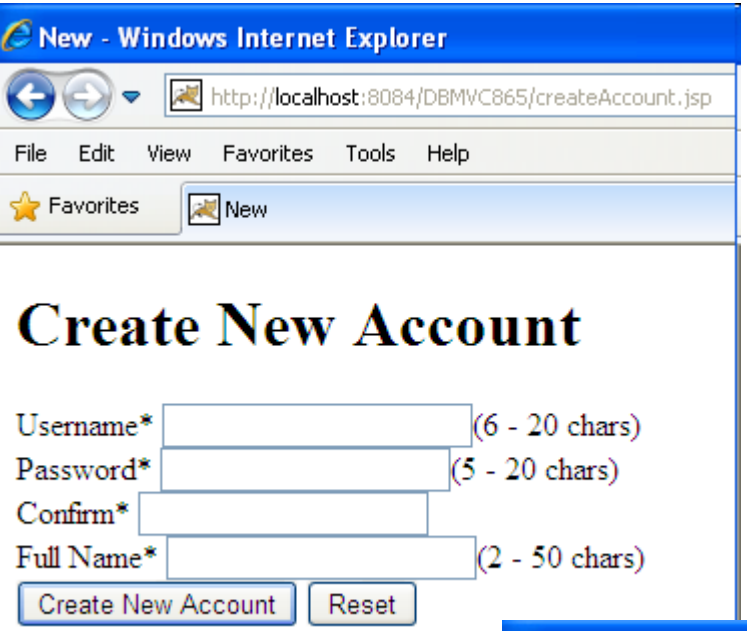
Complete MVC 2
Requirements

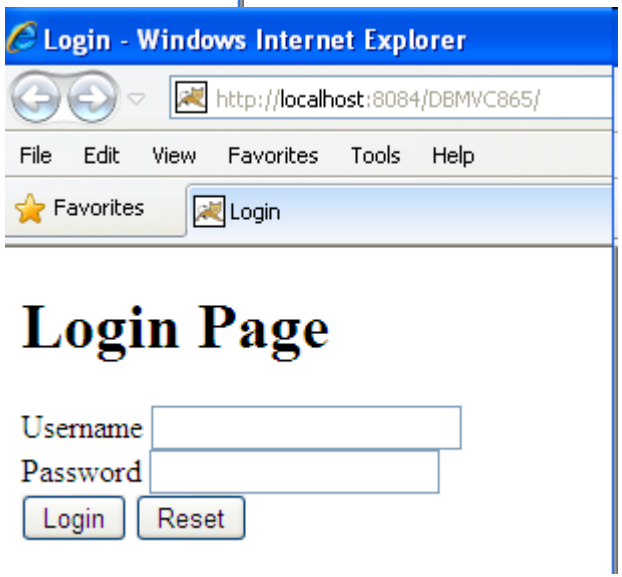
- Complete the MVC 2 with CRUD Web Application in previous topic

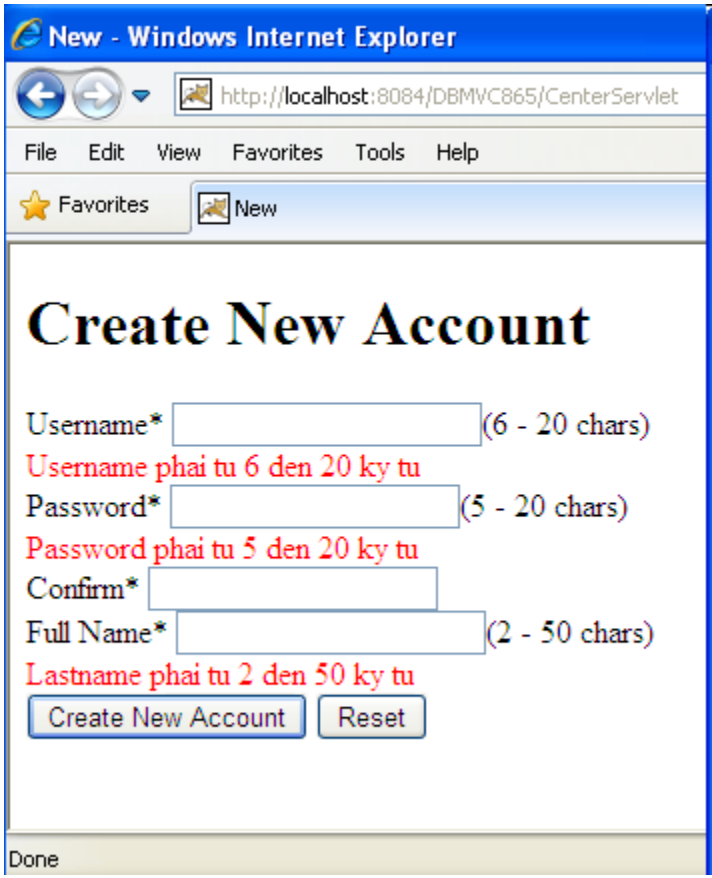




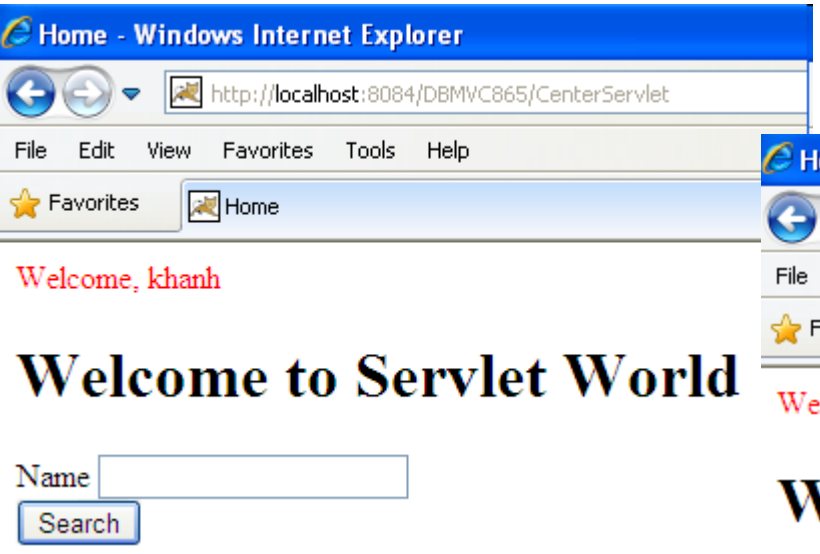
Complete MVC 2
Requirements







Complete MVC 2
Requirements



Fpt University

JSTL
Functions Tag Libraries

- The **library contains tags**, which **provides** the **utility functions** in process on jsp page combining jstl
- All functions **treat null Strings as empty Strings**
- It can be **declared** by

<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

- Some **functions**
  - fn:length
  - fn:contains
  - fn:containsIgnoreCase
  - fn:indexOf
  - fn:split
  - fn:trim
  - fn:substring
  - …

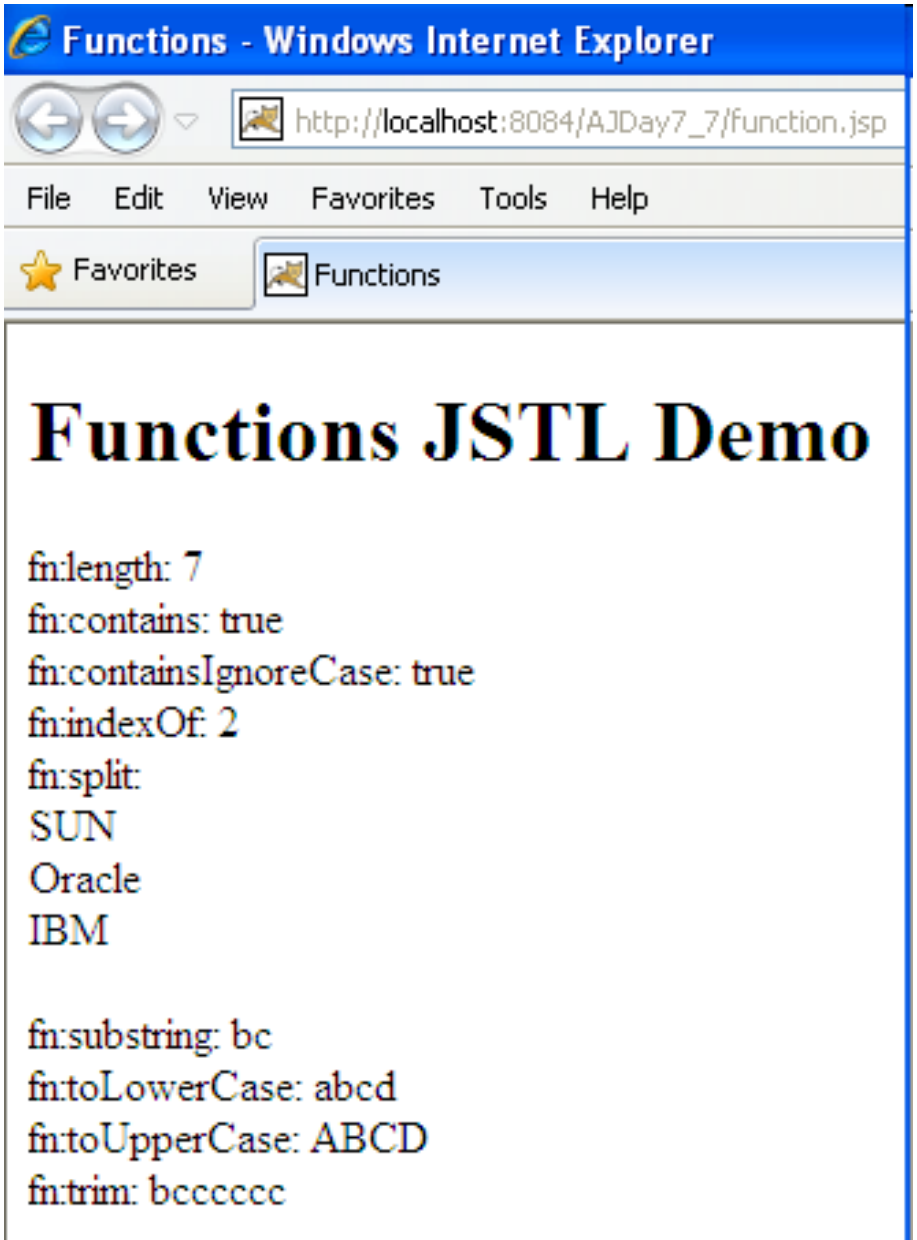
JSTL
Functions Tag Libraries

function.jsp

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
9      "http://www.w3.org/TR/html4/loose.dtd">
10  <%@taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
11  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
12  <html>
13      <head>
14          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15          <title>Functions</title>
16      </head>
17      <body>
18          <h1>Functions JSTL Demo</h1>
19          fn:length: ${fn:length("This is")}<br/>
20          fn:contains: ${fn:contains("abc", "a")}<br/>
21          fn:containsIgnoreCase: ${fn:containsIgnoreCase("ABC", "a")}<br/>
22          fn:indexOf: ${fn:indexOf("abcd", "c")}<br/>
23          fn:split: <br/>
24          <c:forEach var="item" items="${fn:split('SUN;Oracle;IBM', ';')}">
25              ${item}<br/>
26          </c:forEach><br/>
27          fn:substring: ${fn:substring("abcd", 1, 3)}<br/>
28          fn:toLowerCase: ${fn:toLowerCase("ABCD")}<br/>
29          fn:toUpperCase: ${fn:toUpperCase("abcd")}<br/>
30          fn:trim: ${fn:trim("      bccccc     ")}<br/>
31      </body>
32  </html>
```

Fpt University

JSTL
Functions Tag Libraries



JSTL
SQL Tag Library

- The library contains tags, which are used to **access SQL databases**, are used to perform the database related tasks, such as **selection, insertion, deletion and modification without using java codes**
- It provides an interface for executing SQL queries on database through JSP.
- **It can be declared by**

<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>

- **The "setDataSource" Tag**
  - Is used to **set a data source** for the **database**.
  - Is an empty tag and allows the user to set data source information for the database
  - **Syntax**:

<sql:setDataSource **dataSource="datasource"** | url="jdbcurl" driver="jdbcclassdriver" user="username" password="password" **var="varName" scope="page | request | session | application"** />

  - If the DataSource attribute is used, then url attribute cannot be used.

Fpt University

JSTL
SQL Tag Library

- **The "query" Tag**
  - Searches the database and returns a result set containing rows of data. (**Passing database queries function**)
  - The tag can either be an empty tag or a container tag.
  - The SELECT statement is used to select data from the table.
  - **Syntax**

```
<sql:query var="varName" dataSource="datasource" scope="page | request | session | application">
        SQL Statement
        <sql:param value="value" />
</sql:query>
```

- **The "update" Tag**
  - Executes the **INSERT, UPDATE AND DELETE** statements.
  - Zero is returned if no rows are affected by statements
  - **Syntax**

```
<sql:update var="varName" dataSource="datasource" scope="page | request | session | application">
    SQL Statement
    <sql:param  value="value" />
</sql:update>
```

JSTL
SQL Tag Library

- **The var attribute of query Tag**
  - Is **javax.servlet.jsp.jstl.sql.Result** interface
  - **Methods**
    - **columnNames**: Returns the names of the columns in the result
    - **rowCount**: Returns the number of rows in the cached ResultSet
    - **rows**: Returns the result of the query as an array of SortedMap objects
    - **rowsByIndex**: Returns the result of the query as an array of arrays
    - **limitedByMaxRows**: Returns true if the query was limited by a maximum row setting

JSTL
SQL Tag Library – Example

Fpt University

Login - Windows Internet Explorer

http://localhost:8084/AJDay7_7/login.html

File Edit View Favorites Tools Help

Favorites Login

Login Page

Username khanh

Password •••••••

Login Reset

Welcome, khanh

Welcome, khanh

Welcome to EL + JSTL

Name a

Search

Title

Course

No.	Username	Password	Lastname	Roles
1	Danh Vong	12345	Danh Dai Ca	false
2	kieukhanh	123445	Khanh Kieu Trong	true

Create new account

Username

Password

Lastname

Create New Account

Home - Windows Internet Explorer

http://localhost:8084/AJDay7_7/processLogin.jsp

File Edit View Favorites Tools Help

Favorites Home

Welcome, khanh

Welcome, khanh

Welcome to EL + JSTL

Name

Search

Title Forward

Course AJCourse

No.	Username	Password	Lastname	Roles
1	1008	333	345	
2	aptech	aptech1	0000	false
3	Danh Vong	12345	Danh Dai Ca	false
4	HieuLegend	legend1345	Hieu kk	false
5	khanh	kieu123	kieu	true
6	kieukhanh	123445	Khanh Kieu Trong	true
7	Phong	123456	Phong cui	false

Create new account

Username

Password

Lastname

Create New Account

Fpt University

JSTL
SQL Tag Library

- **The "transaction" Tag**
  - Is used to **established a transaction** context for <**sql:query**> and <**sql:update**> tags.
  - The connection object is obtained from <sql:transaction> as this tag is responsible for managing access to the database.
  - **Syntax**

```
<sql:transaction dataSource="datasource" isolation="isolationLevel">
    <sql:update> or <sql:query> Statements
</sql:transaction>
```

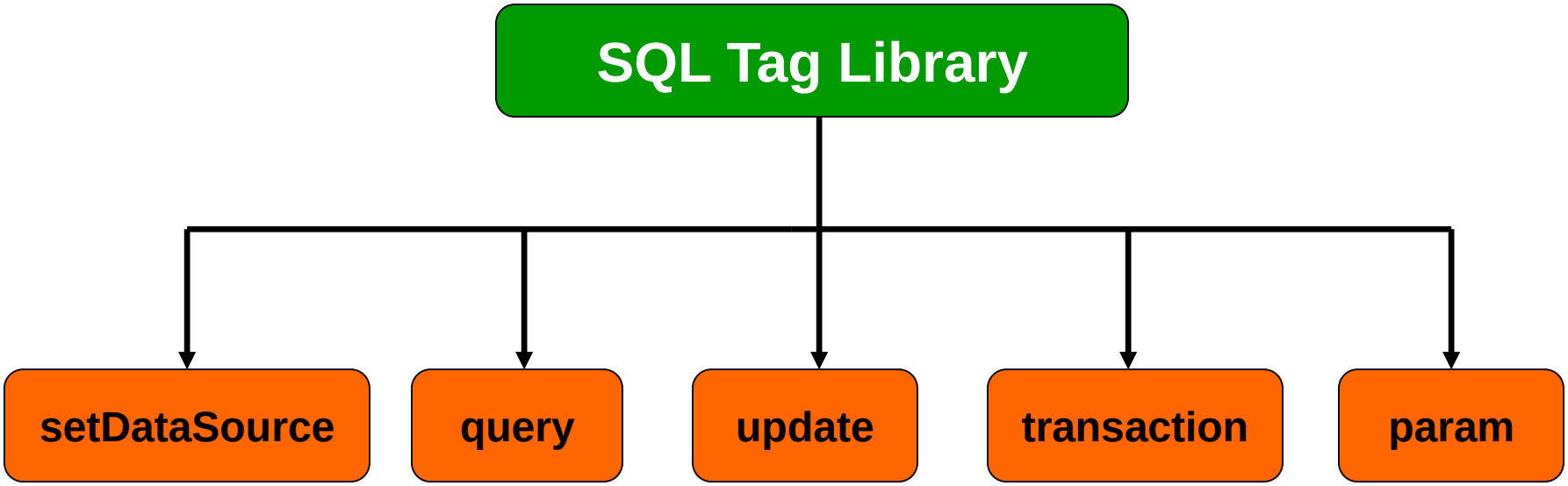
- **The "param" Tag**
  - Is used to **set values for parameters markers ("?")** in SQL statements.
  - It acts as a sub tag for <sql:query> and <sql:update>
  - **Syntax**

<sql:param value="value"/>

Fpt University

JSTL

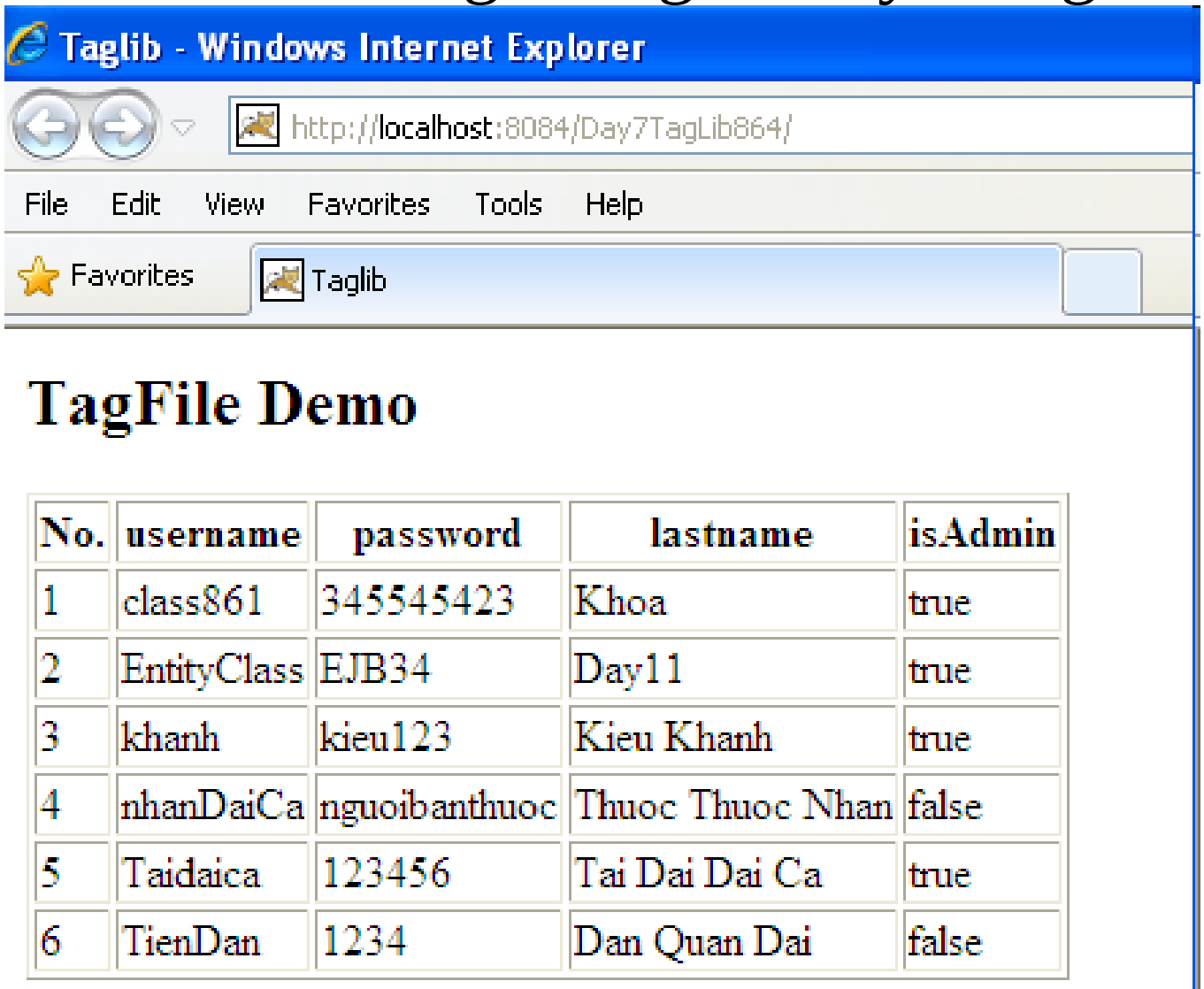
SQL Tag Library



Build Library Tag Lib Library

Requirements

- Build the data grid tag library using in JSP as



No.	username	password	lastname	isAdmin
1	class861	345545423	Khoa	true
2	EntityClass	EJB34	Day11	true
3	khanh	kieu123	Kieu Khanh	true
4	nhanDaiCa	nguoibanthuoc	Thuoc Thuoc Nhan	false
5	Taidaica	123456	Tai Dai Dai Ca	true
6	TienDan	1234	Dan Quan Dai	false

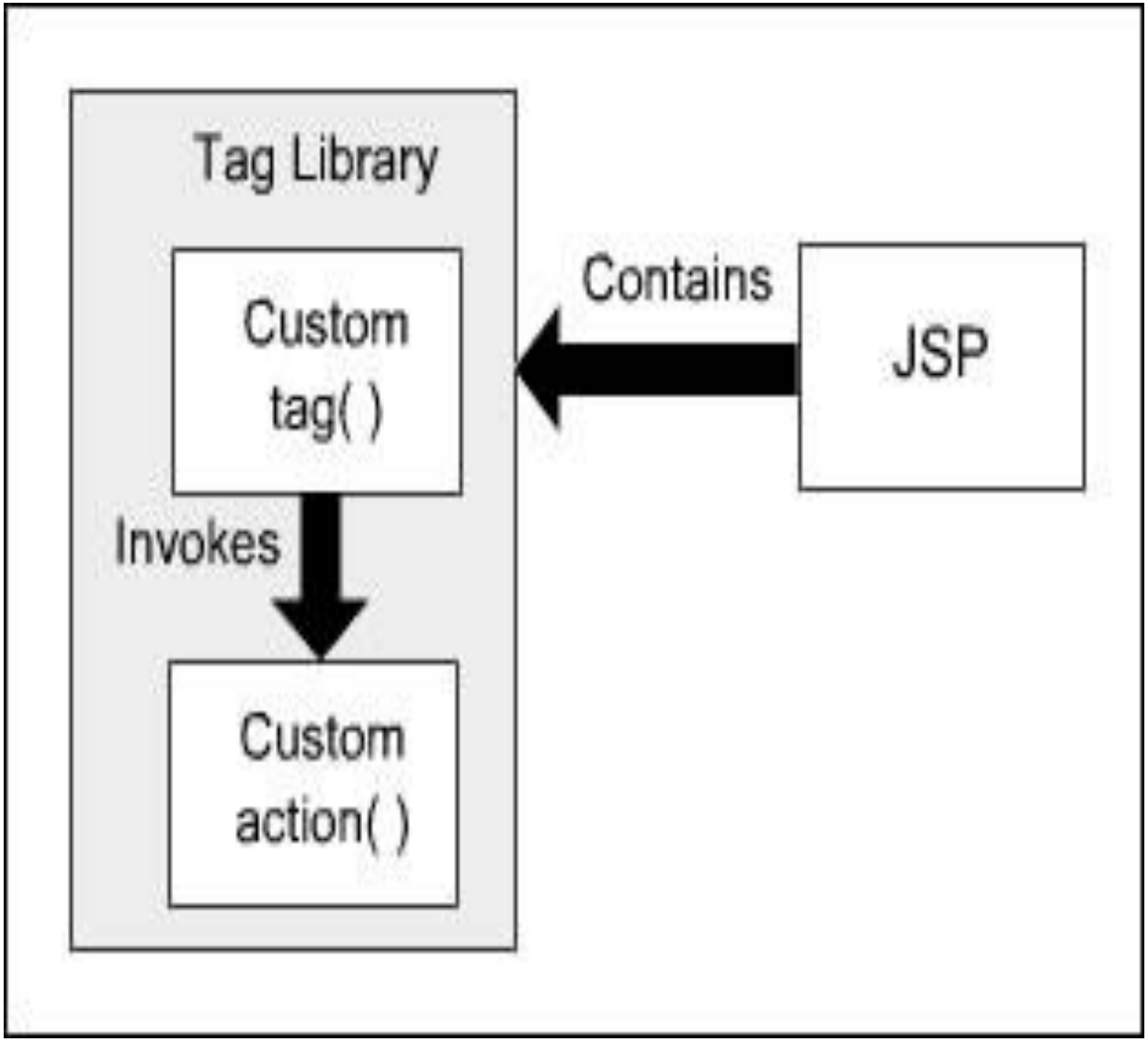
Tag Libraries

Tag Extension Mechanism

- Enables **creation** of custom tags
  - Java code can be **avoided** using custom tags
  - **Separates the work profiles** of Web designers and developers (representation and business logic)
- Custom tags **can be reused**
- **Written using XML** syntax
- **Encapsulation**
- Components of custom tag
  - **Tag Library Descriptor (TLD):** an XML document contains information about the contents of a tag library.
  - **Custom Action**: a Java class implements the contents in the TLD
  - **Web deployment descriptor**: locate the custom tag library to use
- A Tag is implemented through extending some interfaces of the **javax.servlet.jsp.tagext** package

Tag Libraries

Tag Extension Mechanism



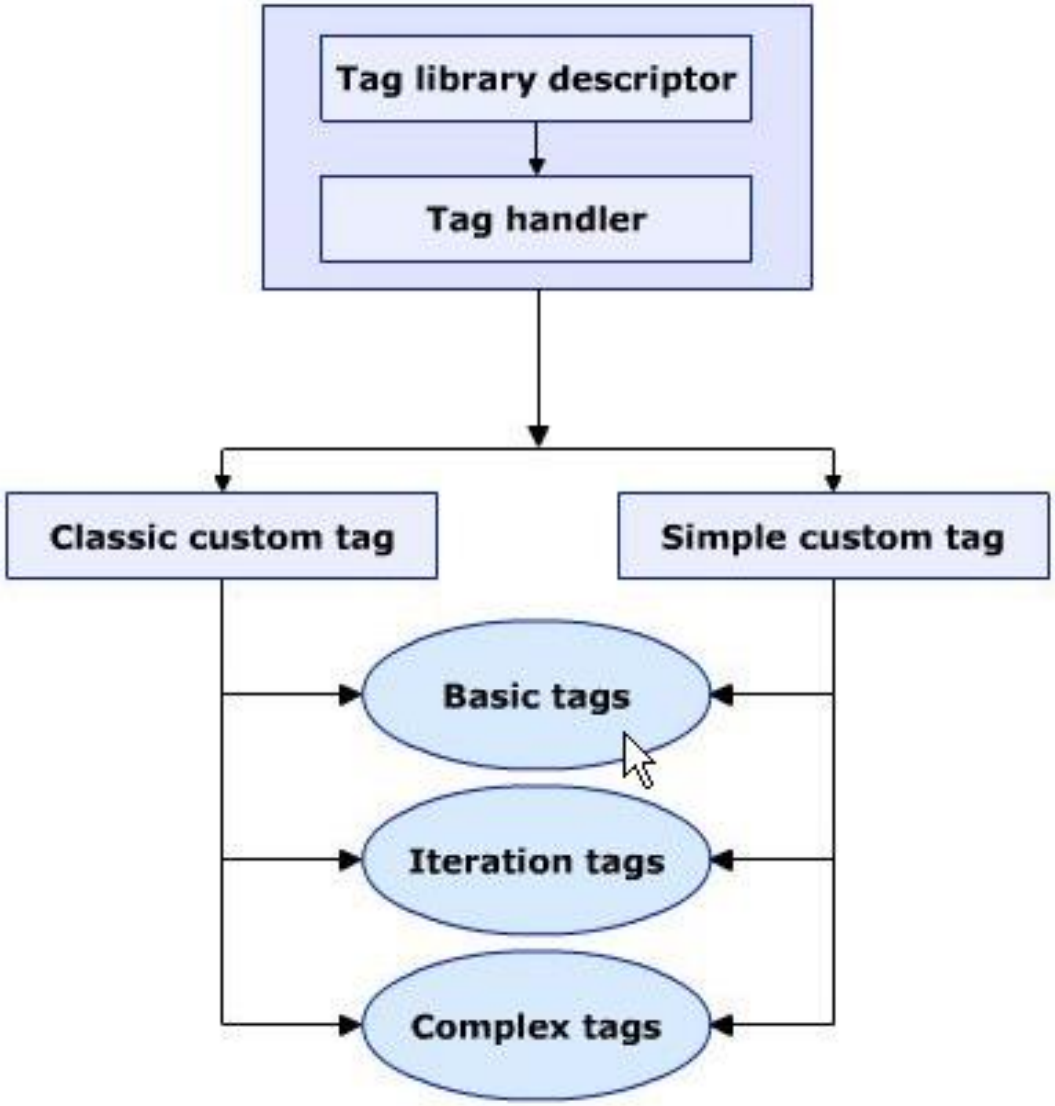
Tag Libraries

Terminologies

- **Classic** custom tag is defined by the classic tag handler class.
- **Simple** custom tag is defined by the simple tag handler classes.
- **Basic** tags
  - Generally do not contain any body. (Even if it contains body, the body is not processed by the tag.)
  - The tag handler class of the basic tag implements the Tag interface and extends the TagSupport class.
- **Iteration** tags
  - Are valuated time and again.
  - Are generally empty tags.
  - The tag handler class for an iteration tags needs implementation of IterationTag interface that extends the Tag interface.
- **Complex** tags
  - Support evaluation of the body content of the tag including other services provided in the basic tag or iteration tag.
  - Implement BodyTag interface and extend the BodyTagSupport class.

Tag Libraries

Terminologies



Custom Tags Terminology

Tag Libraries

Custom Tags

- Allow Java programmer to **embed Java codes in JSP documents**
- Helps the developer **reducing the overhead of writing** the same business logic again for a particular action to be repeated in programs
- Supporting **the front end,** developer **need not bother about the complexity of the logic of a tag** because the tag is created by a Java developer and the front end developer only imports it to the JSP page.
- Provide a mechanism to **reuse and encapsulate complex recurring code or tasks** in JSP
- Provide **simplicity and reusability** of Java code
- Custom tag body can include **static text, HTML, and JSP elements like scriptlets**, between the start and the end tag

Tag Libraries

The Custom Tag Development Process

- There are 4 essential steps to writing a custom tag
  - **Create a Tag Library Descriptor (.tld) – TLD file**
    - Is an XML document that is used to validate the tags.
    - Contains the information on each tag available in the library
    - Provides the JSP engine with meta-information about the custom tag and about its implementation
    - Contains the list and description of all custom tags in the library
    - Usually locates at the tlds folder inside WEB-INF directory
  - **Create a Tag Handler class** in proper package with tld file.
    - Is a Java class that defines a tag described in the TLD file.
    - What a tag will implement depends on what methods can be called and what is required to be implemented.
    - These are of two types, classic and simple.
  - **Provide details of where to find the TLD file in web.xml (option – required if the tld files don't put at the WEB-INF/tlds directory)**
  - **Creating a JSP page** that will access the custom tags: using a **taglib directive** in the page before any custom tag is used.

Tag Libraries

The TLD file

```
<?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>
 <short-name>shortName-prefix</short-name>
 <uri>/WEB-INF/tlds/CustomTag</uri>
<tag>
  <name>tagName-method</name>
  <tag-class>TagHandlerName</tag-class>
  <body-content>JSP|EMPTY|scriptless|tagdependent</body-content>
  <attribute>
   <name>attributeName</name>
   <required>false/true</required>
   <rtexprvalue>false/true</rtexprvalue>
  </attribute>
 </tag>
/<taglib>
```

Tag Libraries

Mapping TLD file in web.xml

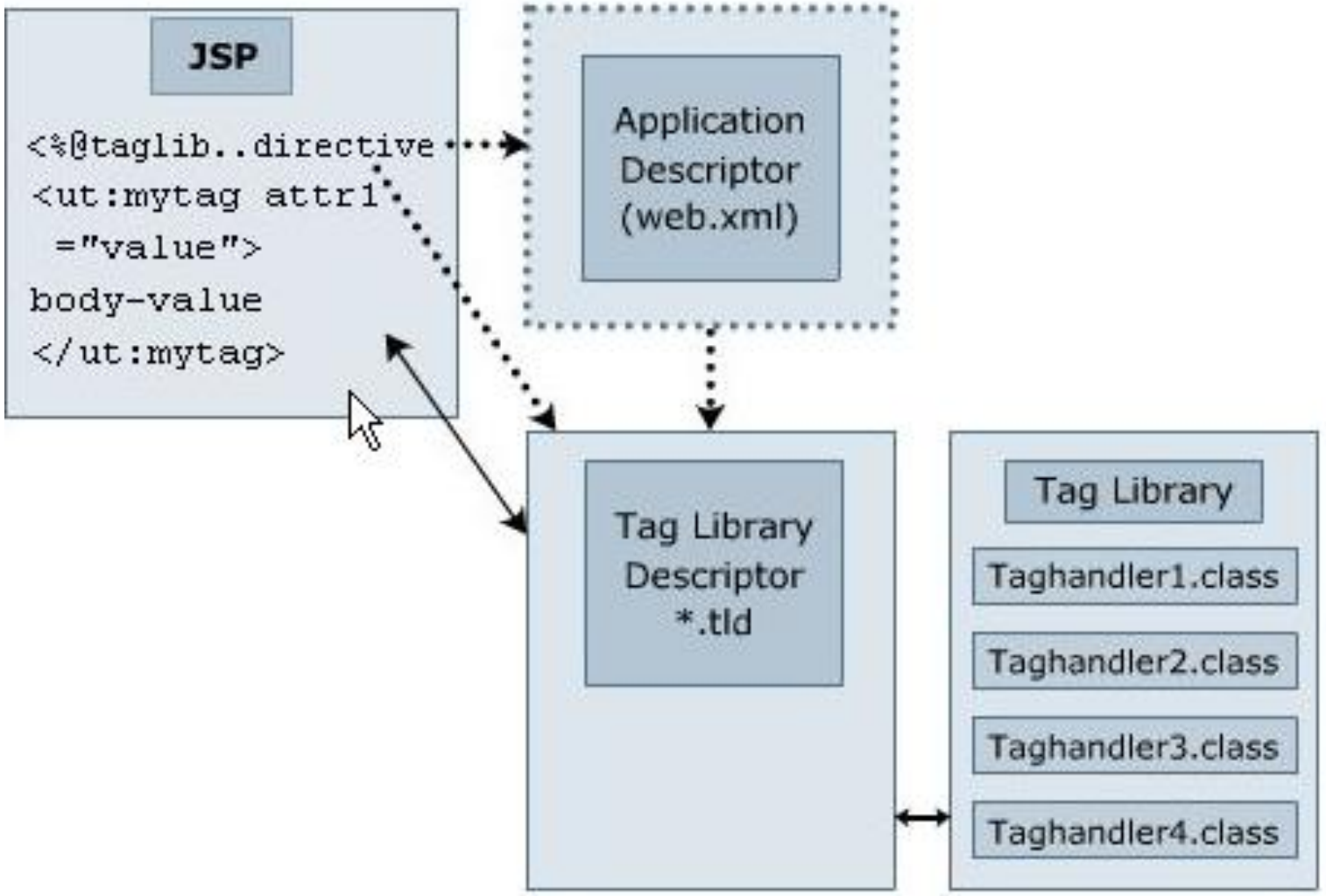
```
<web-app>
    ...
    <jsp-config>
      <taglib>
        <taglib-uri>tld file name</taglib-uri>
        <taglib-location>tld location path</taglib-location>
      </taglib>
    </jsp-config>
    ...
</web-app>
```

- **Ex:**

```
<taglib>
      <taglib-uri>CustTags-taglib.tld</taglib-uri>
      <taglib-location>/WEB-INF/classes/CustTags-taglib.tld</taglib-location>
  </taglib>
```

Tag Libraries

Hunting the Tag



Working of Custom Tag Libraries

Tag Libraries

The Tag Libraries

- Is a collection of custom tags
- Allows the developer to **write reusable code fragments**, which assign functionality to tags
- The reference to each tag is stored in the tag library
- When a tag is used in a JSP page, **the taglib directive must be imported as**

  <%@ taglib uri="/WEB-INF/tlds/TagHandler" prefix="u" %>
- **Locating** TLD File: **after iterating** with the **<@%taglib…> directive**, **traces** the **location** of the **tag library descriptor** file (through **URI**) and **subsequently** to the **location** of the **tag handle class**
- **Associating URIs with TLD** File Locations: The uri attribute is mapped to the .tld file **by two methods**
  - **Implicit**
    - By referring to the tag library descriptor from the expanded WEB-INF folder.
    - By referring to a JAR file containing a tag library.
  - **Explicit:** By defining reference to the tag library descriptor in the deployment descriptor, web.xml of the Web application

Tag Libraries

The Tag Libraries

- **Tag Library Prefix**
  - **Distinguish** these **tags from one another**, they are **named uniquely** by **using prefix attributes in** the **taglib directive**
  - Is used as a **reference by the container** to invoke its respective TLD file and tag handler class
  - The **value** of **prefix and** the **sub element <short-name> inside** the **<taglib> within** the TLD file **should match**
- **Notes**:
  - The tld files can be **mapped in implicit** (e.g. they should be **put at the WEB-INF/tlds)**
  - The **tld** files can be **packaged into a jar file**, and the tld files within the jar file must have a path that **begins /META-INF directory**
  - The **tld** files must **contain the optional <uri> element**, which must **match the uri of the taglib directive or the namespace value**

The "Classic" Custom Tag Event Model

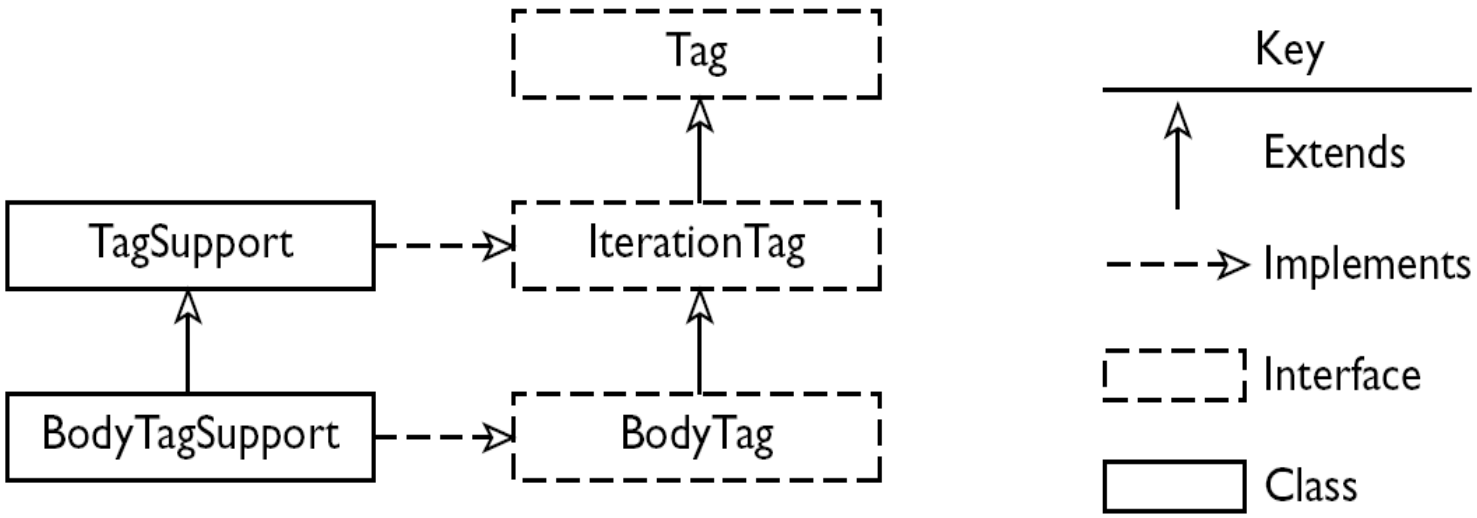
Basics

- Uses a tag handler class, which **implements Tag, IterationTag and BodyTag interfaces** or **extends the classes TagSupport or BodyTagSupport**

Interfaces/Classes	Descriptions
Tag	- Allows communication between a **tag handler and the servlet of a JSP page**. - Is particularly useful when a **classic custom tag is without any body** or even if it has body when it is not for manipulation
IterationTag	- Is used to by tag handlers that require executing the **tag, time and again without manipulating its content.** - **Extends** to the **Tag interface**
BodyTag	- The tag handler class of a **tag** which manipulates **its body** needs to implement - **Extends** to the **IterationTag** and in turn to the **Tag interface.**
TagSupport Class	- Acts as the base class for most tag handlers which support, **empty tag, tag with attributes and a tag with body iterations.** - Implements the Tag and IterationTag interfaces. Thus, it implements all the methods in these interfaces. - Contains methods such as, **doStartTag(),doEndTag() and doAfterBody().**
BodyTagSupport	- Can support tags that need to access and manipulate the **body content** of a tag. - Implements the BodyTag interface. - Extends the TagSupport class. - Contains the following methods: **setBodyContent() and doInitBody().**

The “Classic” Custom Tag Event Model

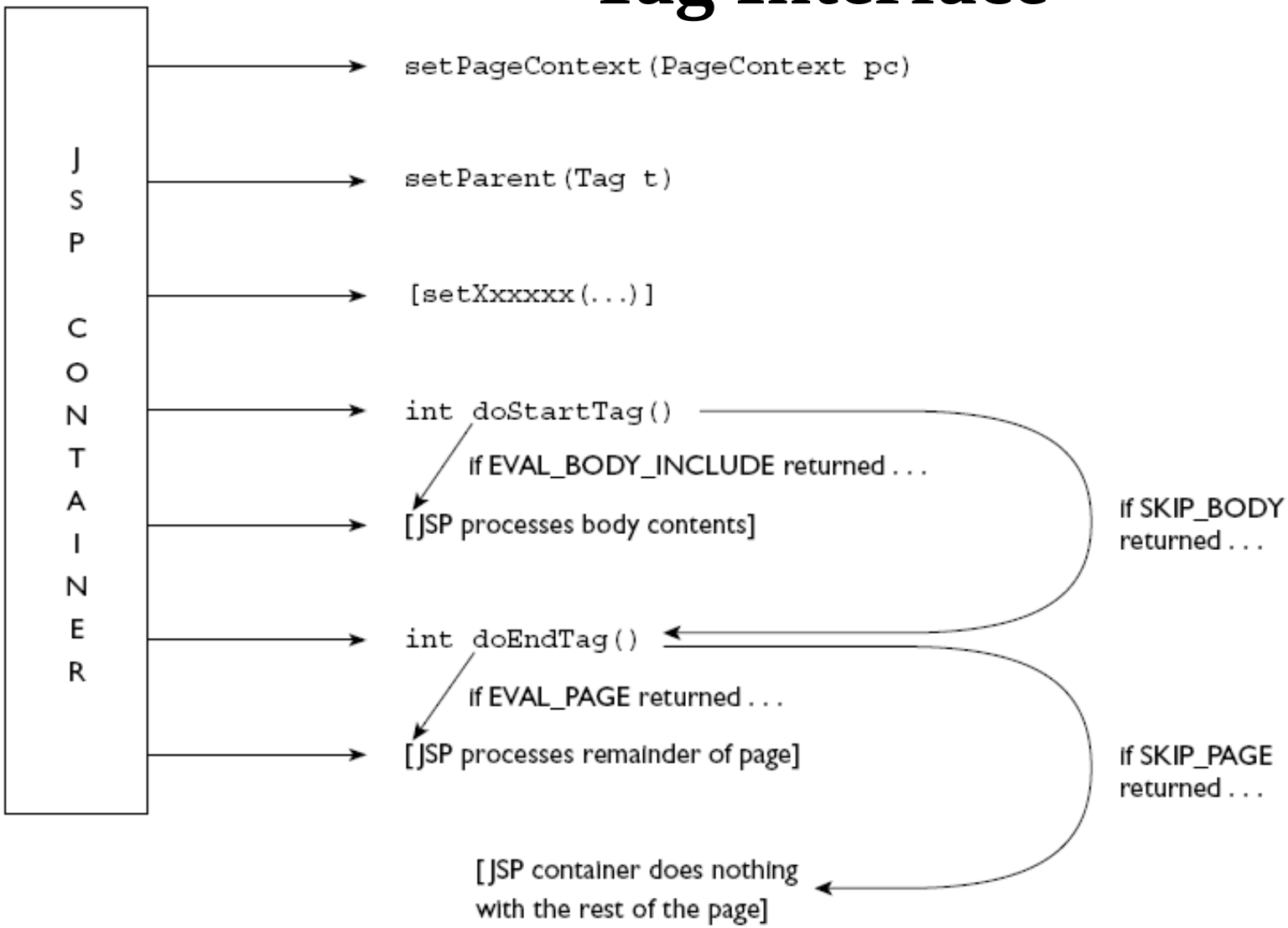
Basics



Tag Interfaces and Classes in javax.servlet.jsp.tagext

The "Classic" Custom Tag Event Model

Tag Interface



Tag Interface

Methods	Descriptions
doStartTag	- **public int doStartTag() throws JSPException** - Is **invoked** when a request is made to a tag or encountering the **starting tag** of the element - Returns the **SKIP_BODY** constant if there is **no body to evaluate** and returns the **EVAL_BODY_INCLUDE** constant if the **body after evaluation needs to be sent to** the **output stream.** - In case of the BodyTag interface, it can return **EVAL_BODY_BUFFERED** that the **evaluated body is to be stored in a buffer temporarily**.
doEndTag	- **public int doEndTag() throws JSPException** - Is invoked when the JSP page **encounters the end tag.** - Generally follows the execution of the **doStartTag**() if the tag is empty. - Returns the **SKIP_PAGE** constant if there is no need to evaluate the rest of the page and returns **EVAL_PAGE** constant if the rest of the page needs to be evaluated.
setParent	- **public void setParent(Tag t)** - Sets the current **nesting or parent tag** of a **nested tag.** - Is invoked prior to **doStartTag**()
getParent	- **public Tag getParent()** - Returns the current parent tag
setPageContext	- **public void setPageContext(PageContext pc)** - Sets the **current page context.** - Is called by the page implementation **prior to doStartTag().**
release	- **public void release()** - Is invoked so that the tag handler will release the resource it holds. The resources utilized by the tag instance are now free for further processing inside the remaining JSP page

Tag Implementation

Empty Custom Tags – Reference uri

- Affecting uri on tld file

custom.tld

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <taglib version="2.1" xmlns="http://java.sun.com/
3    <tlib-version>1.0</tlib-version>
4    <short-name>custom</short-name>
5    <uri>http://kieutrongkhanh.net/customTag</uri>
```

customTag.jsp

```
4        Author      : Trong Khanh
5    --%>
6
7    <%@page contentType="text/html" pageEncoding="UTF-8"%>
8    <!DOCTYPE html>
9    <%@taglib uri="http://kieutrongkhanh.net/customTag" prefix="ct"%>
```

customTag.jsp

```
4        Author      : Trong Khanh
5    --%>
6
7    <%@page contentType="text/html" pageEncoding="UTF-8"%>
8    <!DOCTYPE html>
9    <%@taglib uri="/WEB-INF/tlds/custom.tld" prefix="ct"%>
```

Test Tag - Microsoft Internet Explorer

http://localhost:8086/CustomTagMDL/index.jsp

Custom Tag Demo

Before Custom Tag is invoked!

Empty Tag Create Empty Tag using Tag

After Custom Tag finished!

The "Classic" Custom Tag Event Model

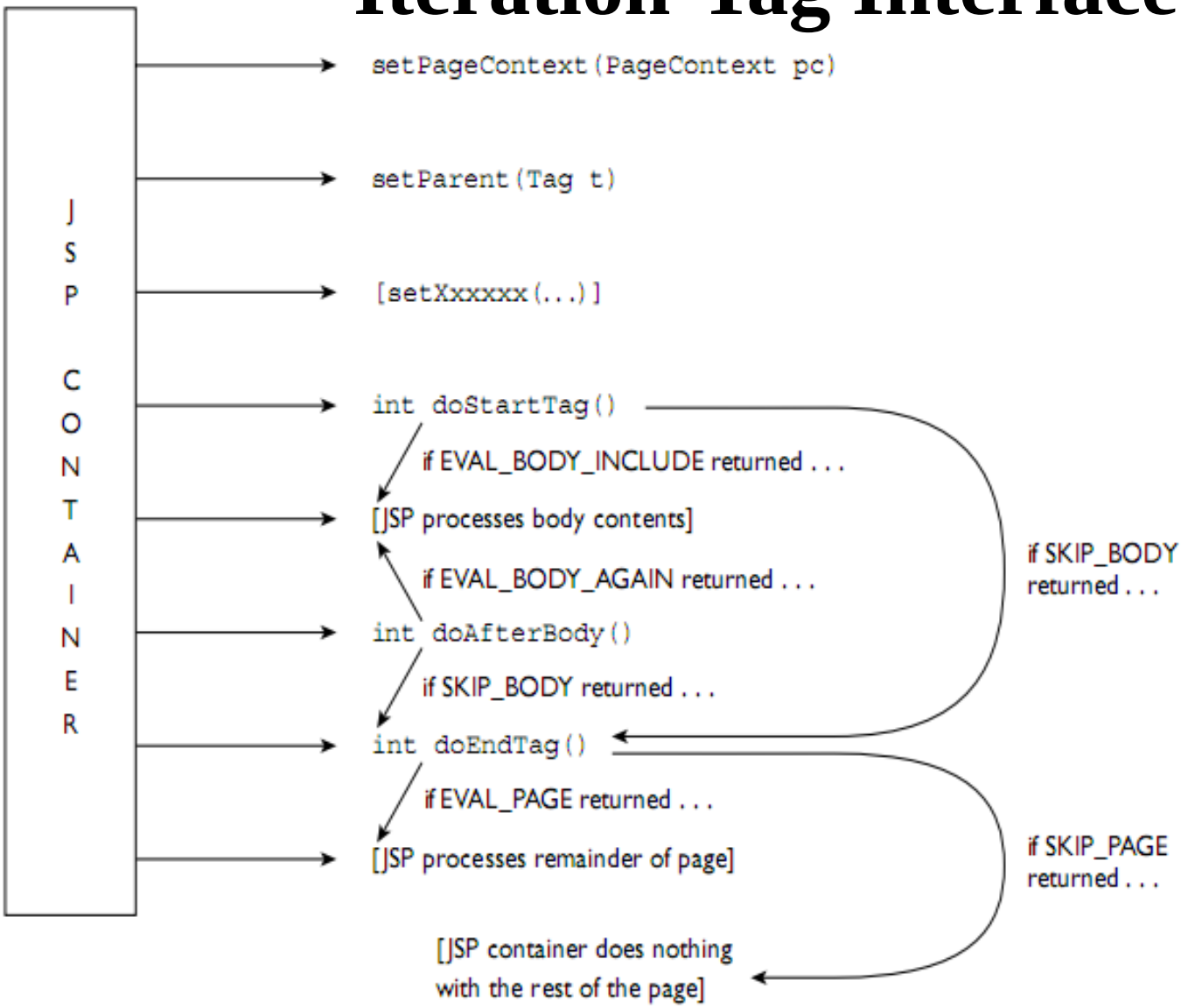
Iteration Tag Interface

- Is implemented to the tag handler class to create tags, which **repeatedly evaluates the body inside it**.
- Is an **extension of the Tag interface**

Methods	Descriptions
doAfterBody	- **public int doAfterBody() throws JSPException** - Allows the user to conditionally **re-evaluate the body** of the tag. - Is **invoked after evaluating the body of the tag**. - Returns the constant **SKIP_BODY** or **EVAL_BODY_AGAIN**. - If the **doStartTag()** method **returns EVAL_BODY_INCLUDE** then this method **returns EVAL_BODY_AGAIN**. - Then the **doStartTag**() is evaluated once again. But when the **doAfterBody**() returns **SKIP_BODY**, the **doEndTag**() method is invoked

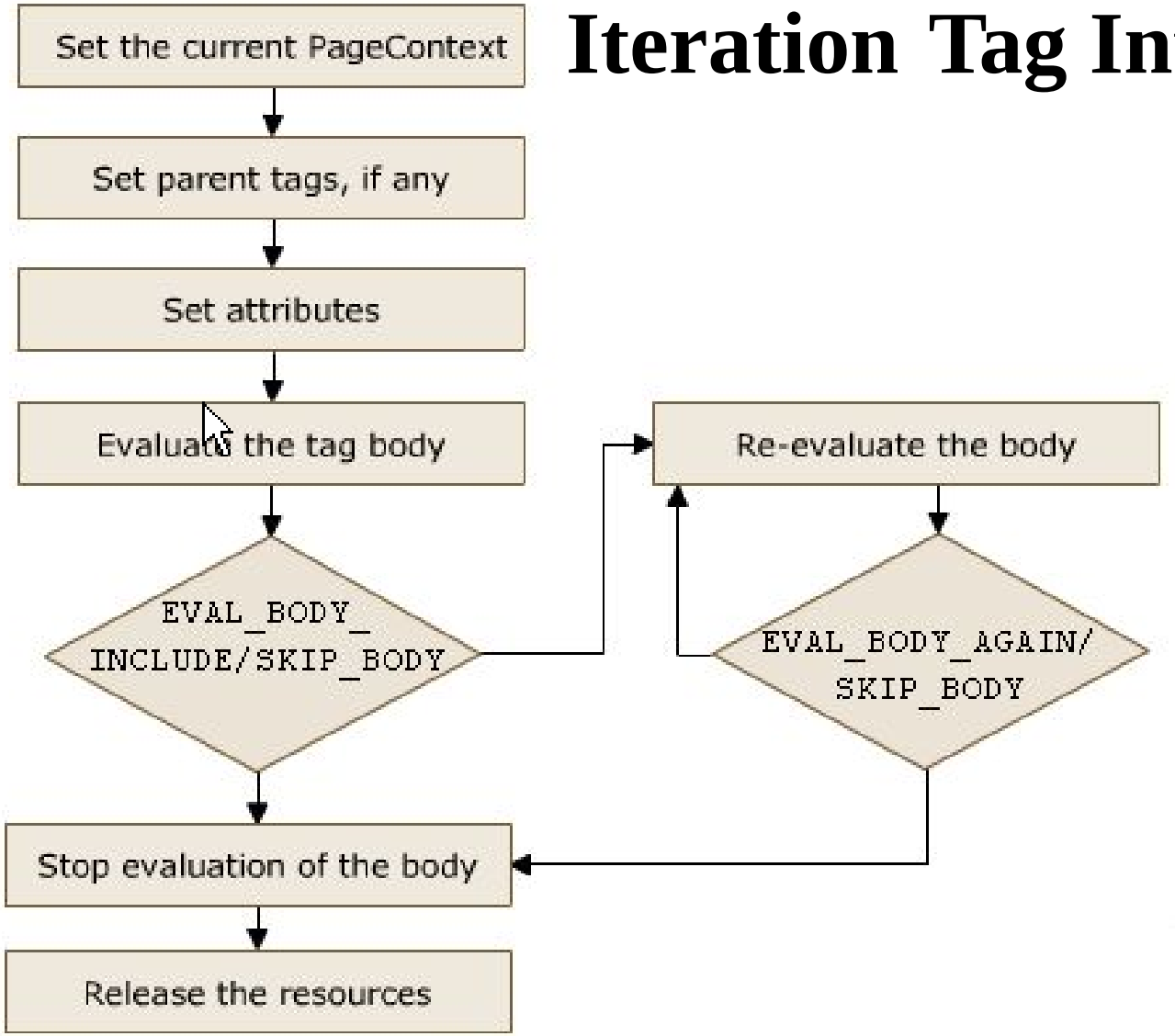
The "Classic" Custom Tag Event Model

Iteration Tag Interface



The "Classic" Custom Tag Event Model

Iteration Tag Interface



Implementing IterationTag Interface

The “Classic” Custom Tag Event Model

TagSupport Class

- Implements the **Tag or IterationTag** interface
- Makes it easy to create a handler class by **limiting the implementation of number of methods**
- Acts as the base class for all tag handlers for the tags with out body and iterative tags
- Suffers from a limitation of **not being able to manipulate the body content**

The "Classic" Custom Tag Event Model

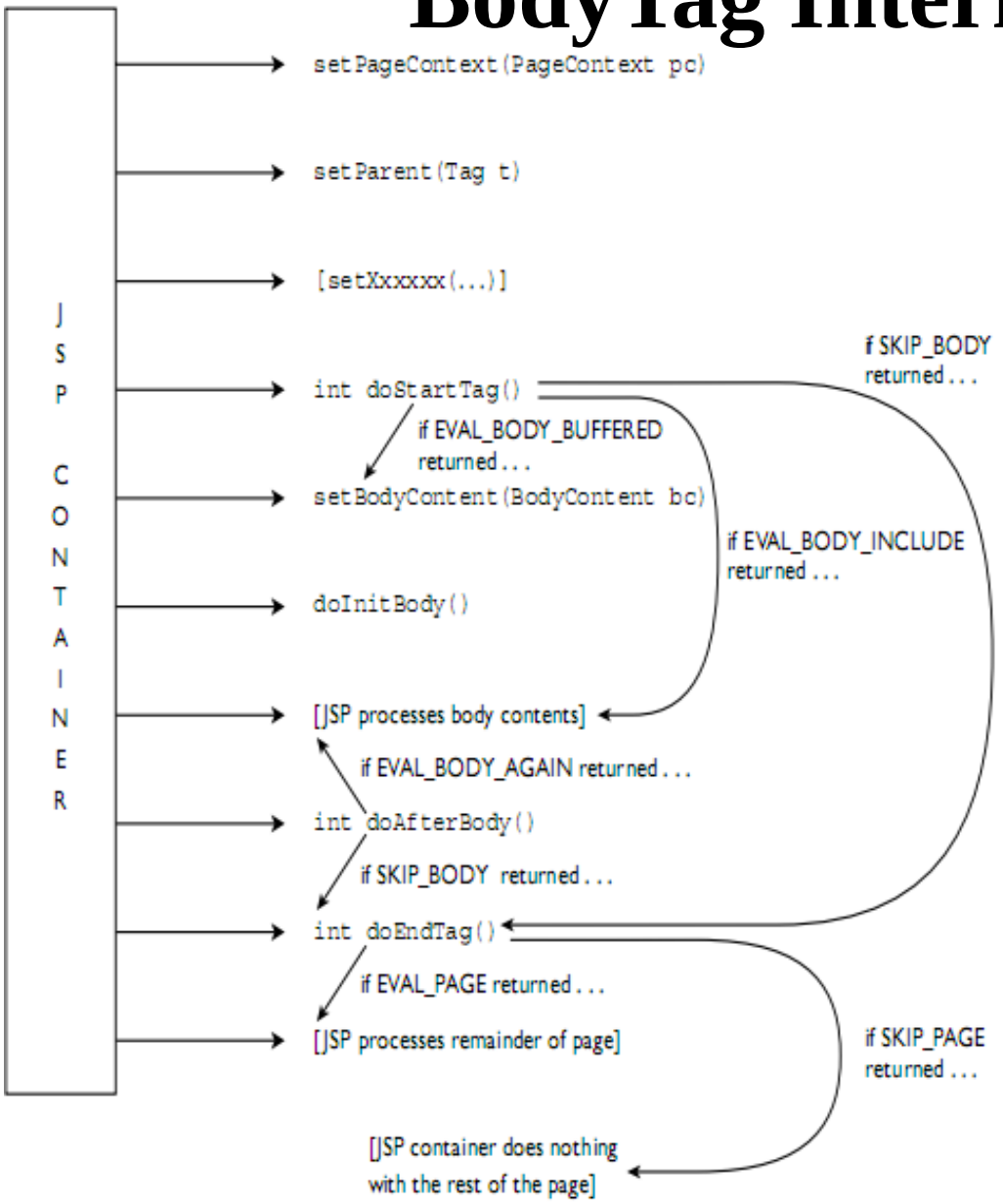
BodyTag Interface

- The tag handler class to **implement the BodyTag interface** that can include body inside it and manipulate the content of the body. This also inherits all the methods from the Tag interface

Methods	Descriptions
doInitBody	- **public void doInitBody()** - Is invoked **immediately after** the **setBodyContent() method returns** the **bodyContent** object and **before** the **first body evaluation.** - Can only be invoked if the tag contains any body
setBodyContent	- **public void setBodyContent(BodyContent bc)** - Sets the **bodyContent** object for the tag handler, the bodyContent object of the BodyContent class **encapsulates the body content of the custom tag for processing.** - Is automatically **invoked** by the **JSP page before** the **doInitBody**() method.

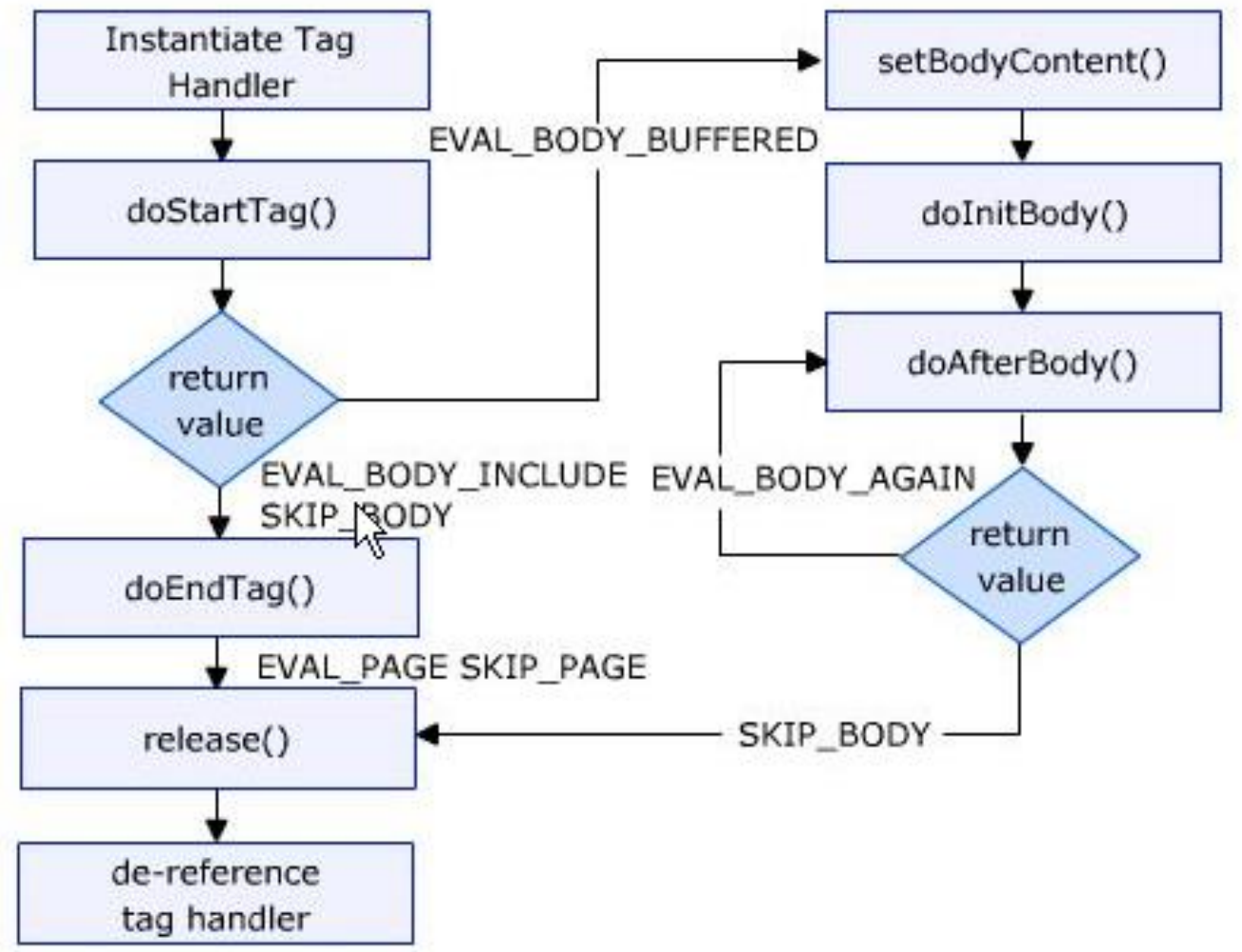
The "Classic" Custom Tag Event Model

BodyTag Interface



The “Classic” Custom Tag Event Model

BodyTag Interface



Implementing BodyTag Interface

The "Classic" Custom Tag Event Model

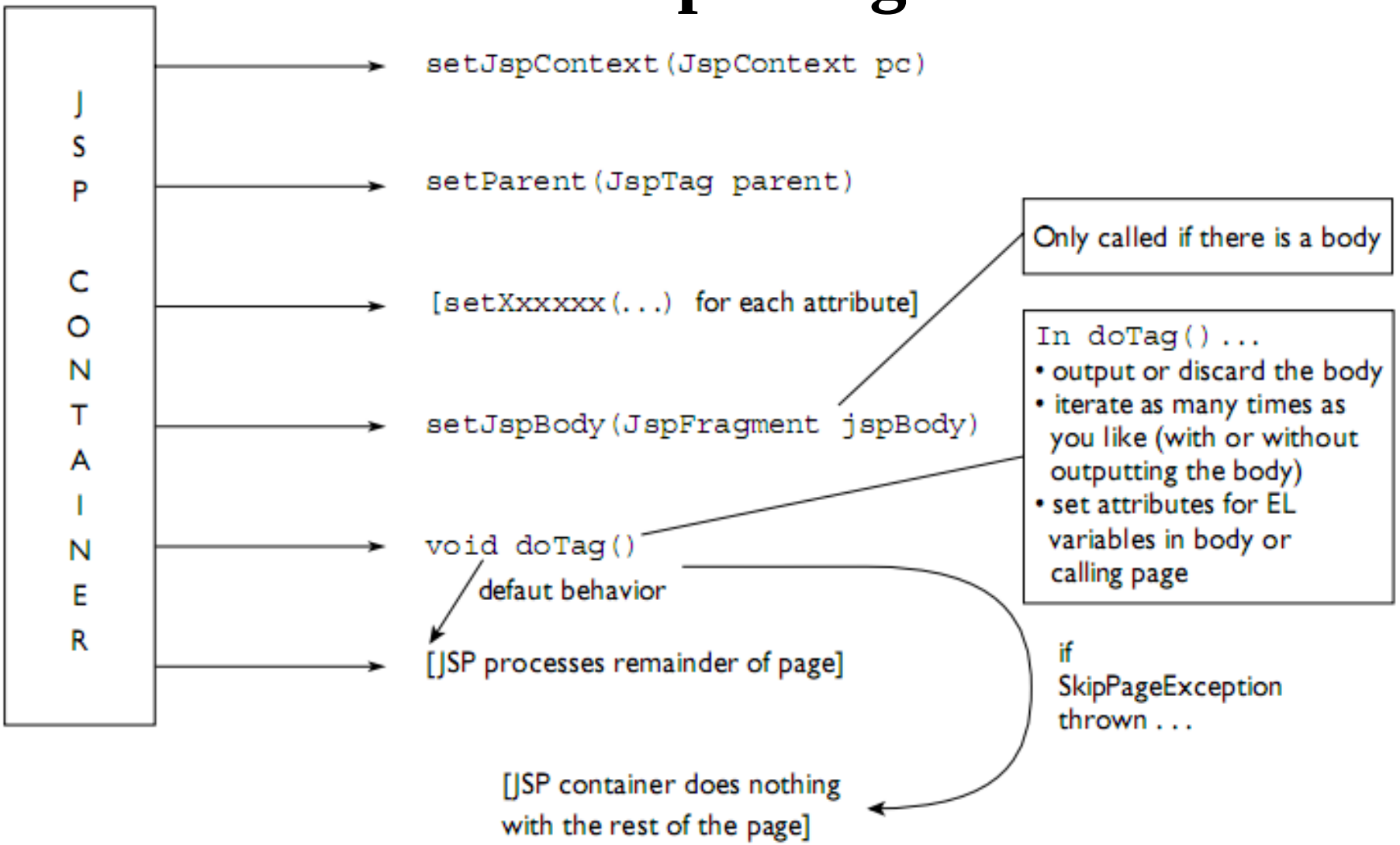
BodyTagSupport Class

- By default **implements the BodyTag interface** and also **extends to the TagSupport class**
- The handler class only **needs to override the methods of the BodyTagSupport class**. All the methods of the TagSupport class and Bodytag interface **are implemented by default** in the handler class. This makes the creation of tag handler class a lot easier.

Methods	Descriptions
getBodyContent	- **public BodyContent getBodyContent()** - **Retrieves** the **bodyContent** object of the class BodyContent. - Contains the body of the tag which is accessed by the container for manipulation.
setBodyContent	- **public void setBodyContent(BodyContent bc)** - The bodyContent object is set by the setBodyContent() method

The "Simple" Custom Tag Event Model

The Simple Tag Interface



The "Simple" Custom Tag Event Model

The Simple Tag Interface

- Provides **doTag() method**, which is **supported by** the Classic Tag Handlers.
- The servlet container **invokes the setJspContext(), setParent(), and methods setting attribute, setting JspFragment and then doTag() method**

Methods	Descriptions
doTag	- **public void doTag() throws JSPException, IOException** - Is used for **handling tag processing.** - Retains **body iteration after being processed.** - Is called whenever there is a requirement of tag invocation. All the manipulations and evaluations are carried out by this method.
setParent	- **public void setParent(JSPTag parent)** - Sets the parent of a specified tag
getParent	- **public JspTag getParent()** - Returns the parent if the specified tag
setJspContext	- **public void setJspContext(JspContext jc)** - Sets the context to the tag handler for invocation by the container.
setJspBody	- **public void setJspBody(JspFragment jspBody)** - Is provided by the body of the specified tag, which can be invoked any number of times by the tag handler

The "Simple" Custom Tag Event Model

The SimpleTagSupport Class

- Acts as a base class for simple tag handlers.
- Implements the SimpleTag interface. It **adds several useful** methods
- The basic implementation of SimpleTag interface is managed by javax.servlet.jsp.tagext.SimpleTagSupport class. The interface is implemented by extending the SimpleTagSupport class and overriding the doTag() method

Methods	Descriptions
getJspContext	- **protected JspContext getJspContext()** - Returns the **context passed into** the **container** by setJspContext() method
getJspBody	- **protected JspFragment getJspBody()** - **Returns** the **fragment encapsulating** the **body of** this **tag**, set by setJspContext().

Custom Tags

Tags and Implicit Variables

JSP Implicit Variable	PageContext Method Used to Obtain Equivalent Object
Request	`getRequest()`
Response	`getResponse()`
Out	`getOut()` (inherited from PageContext's parent class JspWriter)
Session	`getSession()`
Config	`getServletConfig()`
Application	`getServletContext()`
Page	`getPage()`
PageContext	(This *is* the PageContext object passed to your tag handler)
Exception	`getException()`

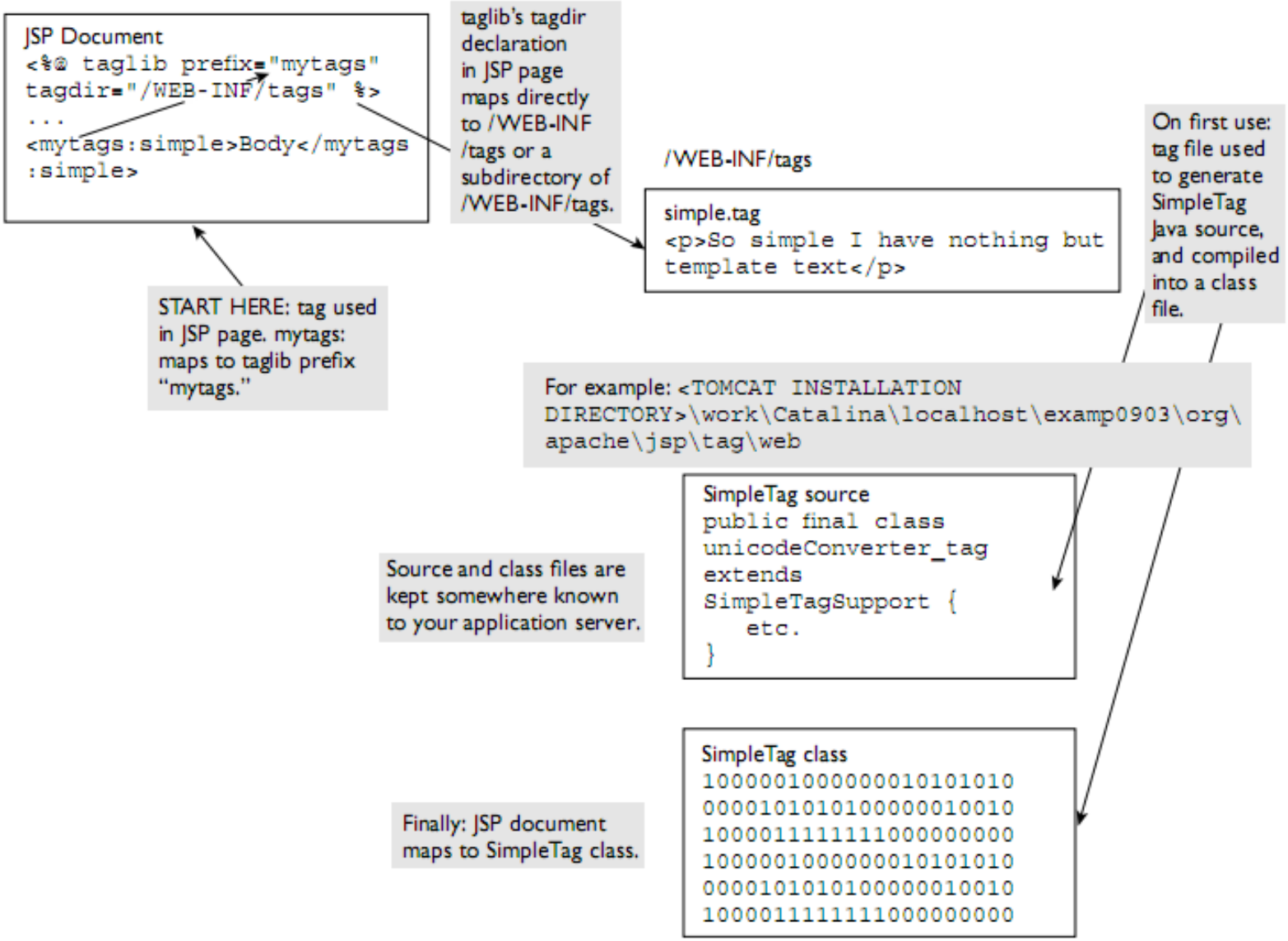
The Tag File Model

Overview

- Are simple tags whose source is **in JSP-syntax form**
- Provide an **advanced way to build standard template** and use it when required.
- Is a text file with **.tag extension stored** in the **WEB-INF/tags** directory of the Web application.
- **Acts** as a **tag handler file**. During the execution of a JSP page, while coming across a custom tag is initially taken to tag file for processing the tag definition.
- **Separate tag file are not required because** the **tag** files **contain** the total **implementation** of the **custom tag**
- Can be access through taglib directive as form

<%@ taglib **tagdir**=”/WEB-INF/tags” prefix=”prefixName”%>

The Tag File Model
Overview



Fpt University

The Tag File Model

Tag File vs. TLD

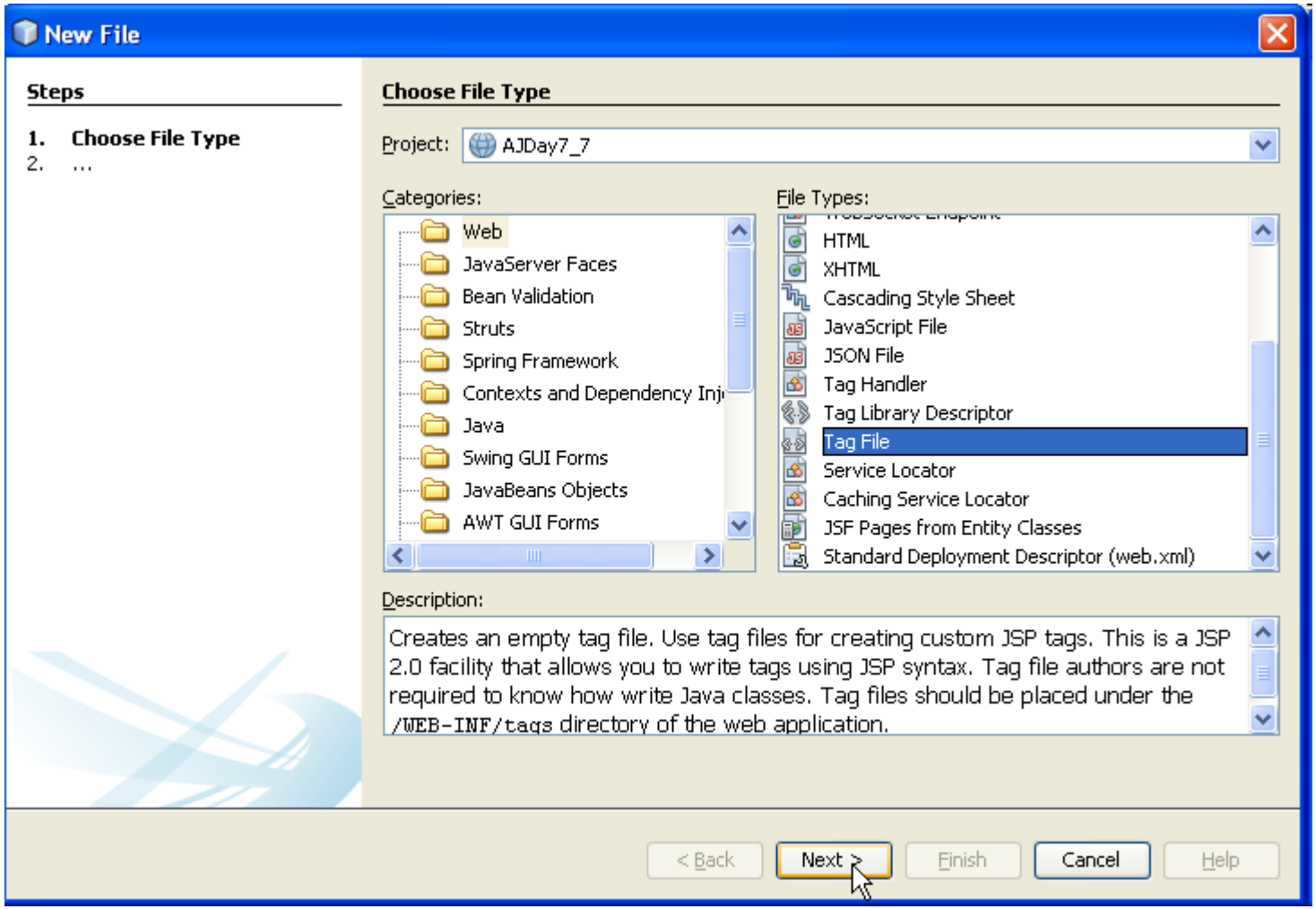
Tags	TLDs
Reusable components of JSP pages	- Local to a project
Can hide and eliminate scriptlets	- Cannot hide and eliminate scriptlets
Can be written easily	- High end programming required
Follow syntax closer to HTML	- Follow java programming syntax
High level as compared to TLD	- Low level components
Need to be defined earlier	- Generated automatically

Tag File Directive

- Is used for **instructing** the **container to compile or interpret** efficiently
- During deployment, the Web container **generates** its **own TLD files with** the **help** of these **directives**. The **tag receives inputs** through its **directives**. It can also be used to generate data by creating EL variables

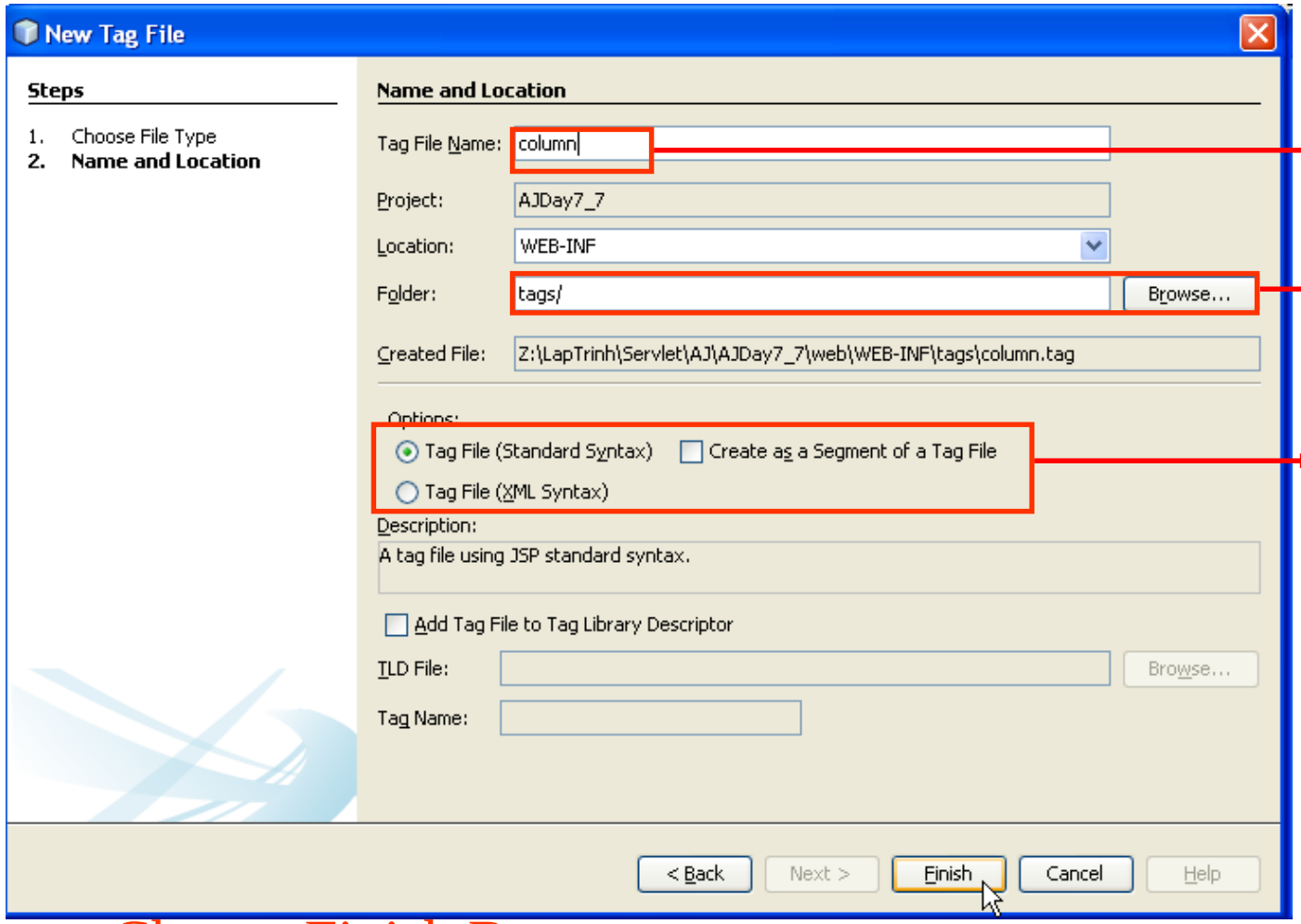
Directives	Descriptions
tag	- **<%@ tag attr1="…" …%>** - Is similar to a page directive but it is used in JSP tag files. - It is used to declare custom tag properties. - Ex: <%@ tag import="java.util.*"%>
include	- **<%@ tag include file="file name"%>** - Directive is used to include the contents of other files in the present file. - Can be accessed by more then one tag files if a common source is present. - Ex: <%@tag include file="process.jsp"%>
taglib	- **<%@ taglib uri="tagLibURI" prefix="tagPrefix"%>** - Custom actions that need to be related from a tag file can be used by the taglib directive - Ex: <%@taglib tagdir="WEB-INF/tags" prefix="demo"%>
attribute	- **<%@ attribute att1="value1" …%>** - Supports the use of **attributes in a tag file.** - Is similar to attribute element in a TLD. - Ex: <%@ attribute name="format" required="false"%>
variable	- **<%@ variable att1="varName" att2="value"…%>** - Is used to **define a variable that** can be **used by the calling JSP page**. - Ex: <%@variable name-given="price"%>

The Tag File Model
Implementation



- Click Next Button

The Tag File Model
Implementation



Fill your tagfile name

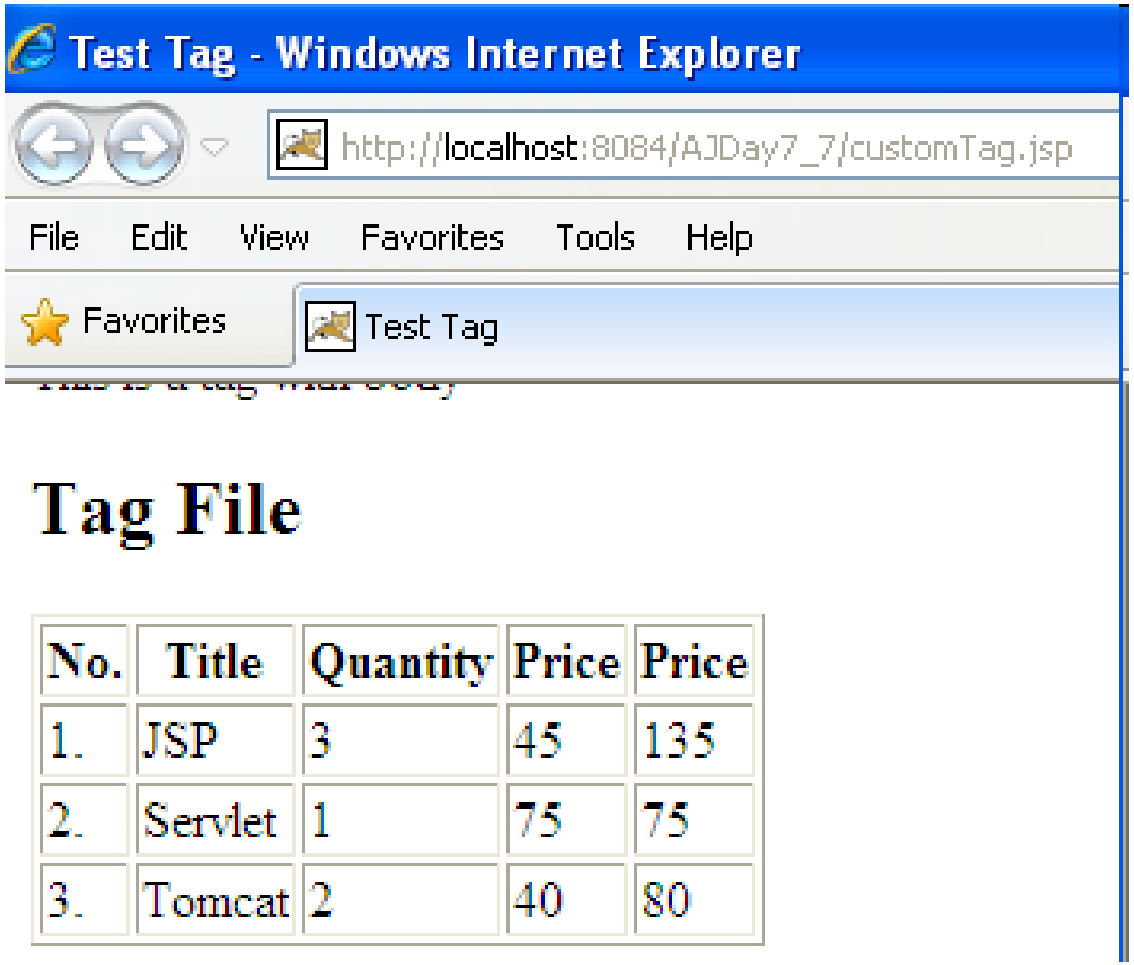
Modify or Browser the location where tags are stored. Should not be changed

Choose TagFile (Standard Syntax)

- Choose Finish Button
- The tags file is automatically created at WEB-INF/tags
- Coding tags body

The Tag File Model
Implementation

- Building the web application using tag file model has GUI presenting as following



The Tag File Model
Implementation

header.tag

Source | History

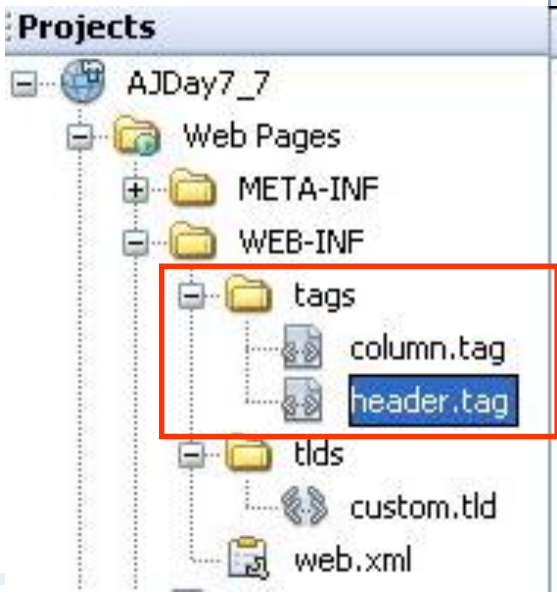
```
4       Author      : Trong Khanh
5   --%>
6
7   <%@tag description="put the tag description her
8
9   <%-- The list of normal or fragment attributes
10  <%@attribute name="header1" required="true"%>
11  <%@attribute name="header2" required="true"%>
12  <%@attribute name="header3" required="true"%>
13  <%@attribute name="header4" required="true"%>
14  <%@attribute name="header5" required="true"%>
15
16  <%-- any content can be specified here e.g.: --
17  <tr>
18      <th>${header1}</th>
19      <th>${header2}</th>
20      <th>${header3}</th>
21      <th>${header4}</th>
22      <th>${header5}</th>
23  </tr>
```

The Tag File Model
Implementation

column.tag

Source | History

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@tag description="put the tag description here" pageEncoding="UTF-8"%>
8
9   <%-- The list of normal or fragment attributes can be specified here: --%>
10  <%@attribute name="column1" required="true"%>
11  <%@attribute name="column2" required="true"%>
12  <%@attribute name="column3" required="true"%>
13  <%@attribute name="column4" required="true"%>
14
15  <%-- any content can be specified here e.g.: --%>
16  <tr>
17      <td>${column1}</td>
18      <td>${column2}</td>
19      <td>${column3}</td>
20      <td>${column4}</td>
21      <td>${column3 * column4}</td>
22  </tr>
```



The Tag File Model
Implementation

customTag.jsp

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/custom.tld" prefix="ct"%>
10  <%@taglib tagdir="/WEB-INF/tags/" prefix="tag"%>
```

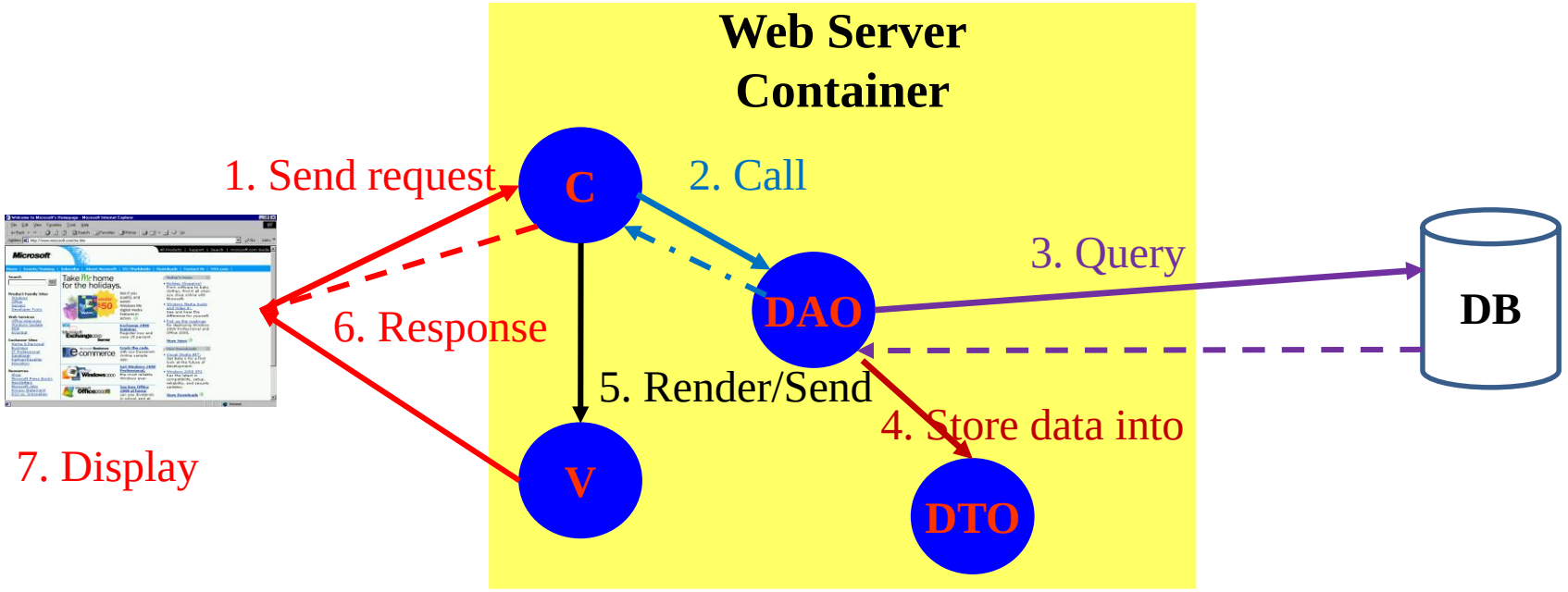
```
<h2>Tag File</h2>
<table border="1">
    <tag:header header1="No." header2="Title"
                header3="Quantity" header4="Price" header5="Price"/>
    <tag:column column1="1." column2="JSP" column3="3" column4="45"/>
    <tag:column column1="2." column2="Servlet" column3="1" column4="75"/>
    <tag:column column1="3." column2="Tomcat" column3="2" column4="40"/>
</table>
```

Summary

- **How to remove all java code in JSP (View)? Complete the MVC 2 Design Pattern with View**
  - **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

Q&A

Summary


Web Server
Container
1. Send request
C
2. Call
3. Query
DB
DAO
6. Response
5. Render/Send
4. Store data into
7. Display
V
DTO

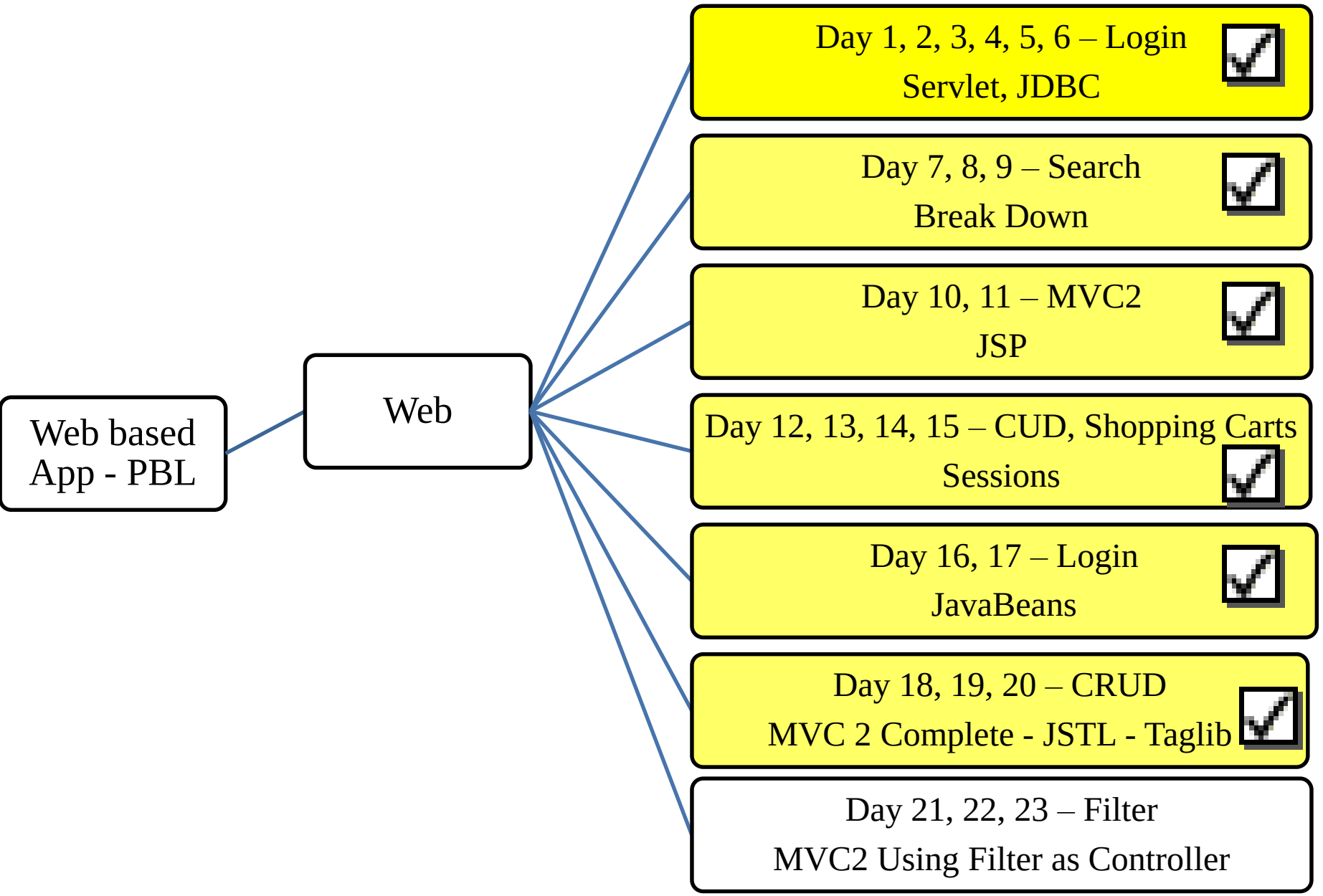
Q&A

Fpt University

Next Lecture

- **How to preprocess in web application?**
  - Filter
  - Filter Chain
  - Using Filter as Controller in MVC2 Design Pattern

Next Lecture



Appendix – Complete MVC 2

Welcome page

welcome.jsp

Source | History

```
4       Author      : Trong Khanh
5   --%>
6   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
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>Home</title>
13      </head>
14      <body>
15          <font color="red">Welcome, ${sessionScope.USERS}</font>
16          <h1>Welcome to Servlet World</h1>
17          <form action="CenterServlet">
18              Name <input type="text" name="txtName" value="" /><br/>
19              <input type="submit" value="Search" name="btAction" />
20          </form> <br/>
```

Complete MVC 2
Welcome page

```
22          <c:set var="name" value="${param.txtName}"/>
23          <c:if test="${not empty name}">
24              <c:set var="info" value="${requestScope.INFO}"/>
25              <c:if test="${not empty info}">
26                  <table border="1">
27                      <thead>
28                          <tr>
29                              <th>No.</th>
30                              <th>Username</th>
31                              <th>Password</th>
32                              <th>Lastname</th>
33                              <th>Roles</th>
34                              <th>Delete</th>
35                              <th>Update</th>
36                          </tr>
37                      </thead>
38                      <tbody>
39                          <c:forEach var="dto" items="${info}" varStatus="counter">
40                          <form action="CenterServlet">
41                              <tr>
42                                  <td>${counter.count}</td>
43                                  <td>
44                                      ${dto.username}
45                                      <input type="hidden" name="txtUser"
46                                             value="${dto.username}" />
47                                  </td>
48                                  <td>
49                                      <input type="text" name="txtPass"
50                                             value="${dto.password}" />
51                                  </td>
```

Complete MVC 2
Welcome page

```
52                                  <td>
53                                      ${dto.lastname}
54                                  </td>
55                                  <td>
56                                      <input type="checkbox" name="chkAdmin"
57                                             value="ADMIN"
58                                             <c:if test="${dto.roles}">
59                                                 checked="checked"
60                                             </c:if>
61                                             />
62                                  </td>
63                                  <c:url var="delLink" value="CenterServlet">
64                                      <c:param name="btAction" value="delete"/>
65                                      <c:param name="user" value="${dto.username}"/>
66                                      <c:param name="searchValue" value="${name}"/>
67                                  </c:url>
68                                  <td><a href="${delLink}">Delete</a> </td>
69                                  <td>
70                                      <input type="hidden" name="txtSearchValue"
71                                             value="${name}" />
72                                      <input type="submit" value="Update" name="btAction" />
73                                  </td>
74                              </tr>
75                          </form>
76                      </c:forEach>
77                  </tbody>
78              </table>
79
80          </c:if>
```

Complete MVC 2

Welcome page

```
81          <c:if test="${empty info}">
82              <h2>No record is matched!!!</h2>
83          </c:if>
84      </c:if>
```

Complete MVC 2
Register Errors Object

RegistrationErrors.java

Source | History

```
13      * @author Trong Khanh
14      */
15     public class RegistrationErrors implements Serializable {
16         private String usernameErrs;
17         private String passwordErrs;
18         private String confirmErrs;
19         private String lastnameErrs;
20         private String duplicateUsername;
21
22         public RegistrationErrors() {...2 lines }
24
25         public RegistrationErrors(String usernameErrs, String passwordErrs,
26                 String confirmErrs, String lastnameErrs, String duplicateUsername) {
27             this.usernameErrs = usernameErrs;
28             this.passwordErrs = passwordErrs;
29             this.confirmErrs = confirmErrs;
30             this.lastnameErrs = lastnameErrs;
31             this.duplicateUsername = duplicateUsername;
32         }
33
34         /**...3 lines */
37         public String getUsernameErrs() {...3 lines }
40
41         /**...3 lines */
44         public void setUsernameErrs(String usernameErrs) {...3 lines }
47
48         /**...3 lines */
51         public String getPasswordErrs() {...3 lines }
```

Complete MVC 2
Register Errors Object

```
54
55      /**...3 lines */
58      public void setPasswordErrs(String passwordErrs) {...3 lines }
61
62      /**...3 lines */
65      public String getConfirmErrs() {...3 lines }
68
69      /**...3 lines */
72      public void setConfirmErrs(String confirmErrs) {...3 lines }
75
76      /**...3 lines */
79      public String getLastnameErrs() {...3 lines }
82
83      /**...3 lines */
86      public void setLastnameErrs(String lastnameErrs) {...3 lines }
89
90      /**...3 lines */
93      public String getDuplicateUsername() {...3 lines }
96
97      /**...3 lines */
100     public void setDuplicateUsername(String duplicateUsername) {...3 lines }
103
```

Complete MVC 2
Register Page

createAccount.jsp

Source | History

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
9   <!DOCTYPE html>
10  <html>
11      <head>
12          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13          <title>New</title>
14      </head>
15      <body>
16          <h1>Create New Account</h1>
17          <form action="CenterServlet" method="POST">
18              <c:set var="err" value="${requestScope.ERRORS}"/>
19              Username* <input type="text" name="txtUsername" value="" />(6 - 20 chars)<br/>
20              <c:if test="${not empty err.usernameErrs}">
21                  <font color="red">${err.usernameErrs}</font><br/>
22              </c:if>
23              Password* <input type="password" name="txtPassword" value="" />(5 - 20 chars)<br/>
24              <c:if test="${not empty err.passwordErrs}">
25                  <font color="red">${err.passwordErrs}</font><br/>
26              </c:if>
```

Complete MVC 2
Register Page

```
27              Confirm* <input type="password" name="txtConfirm" value="" /><br/>
28              <c:if test="${not empty err.confirmErrs}">
29                  <font color="red">${err.confirmErrs}</font><br/>
30              </c:if>
31              Full Name* <input type="text" name="txtLastname" value="" />(2 - 50 chars)<br/>
32              <c:if test="${not empty err.lastnameErrs}">
33                  <font color="red">${err.lastnameErrs}</font><br/>
34              </c:if>
35              <input type="submit" value="Create New Account" name="btAction" />
36              <input type="reset" value="Reset" /><br/>
37              <c:if test="${not empty err.duplicateUsername}">
38                  <font color="red">${err.duplicateUsername}</font><br/>
39              </c:if>
40          </form>
41      </body>
42  </html>
```

Complete MVC 2
Register Servlet

NewAccountServlet.java

Source | History

```
21      * @author Trong Khanh
22      */
23     public class NewAccountServlet extends HttpServlet {
24         private final String errorDisplayServlet = "createAccount.jsp";
25         private final String loginPage = "login.html";
26         /** Processes requests for both HTTP <code>GET</code> and <code>POST</cc
35         protected void processRequest(HttpServletRequest request, HttpServletRes
36                 throws ServletException, IOException {
37             response.setContentType("text/html;charset=UTF-8");
38             PrintWriter out = response.getWriter();
39             try {
40                 String username = request.getParameter("txtUsername");
41                 String password = request.getParameter("txtPassword");
42                 String confirm = request.getParameter("txtConfirm");
43                 String lastname = request.getParameter("txtLastname");
44
45                 boolean errs = false;
46                 RegistrationErrors errObj = new RegistrationErrors();
47                 if (username.length() < 6 || username.length()>20) {
48                     errs = true;
49                     errObj.setUsernameErrs("Username phai tu 6 den 20 ky tu");
50                 }
```

Complete MVC 2
Register Servlet

```
52              if (password.length() < 5 || password.length()>20) {
53                  errs = true;
54                  errObj.setPasswordErrs("Password phai tu 5 den 20 ky tu");
55              } else if (!confirm.equals(password)) {
56                  errs = true;
57                  errObj.setConfirmErrs("Password khong giong Confirm");
58              }
59
60              if (lastname.length() < 2 || lastname.length()>50) {
61                  errs = true;
62                  errObj.setLastnameErrs("Lastname phai tu 2 den 50 ky tu");
63              }
64
65              request.setAttribute("ERRORS", errObj);
66              String url = errorDisplayServlet;
67              System.out.println("dddd");
68              if (!errs) {
69                  RegistrationDAO dao = new RegistrationDAO();
70                  boolean result = dao.createAccount(username, password, lastname, false);
71
72                  if (result) {
73                      url = loginPage;
74                  } else {
75                      errObj.setDuplicateUsername(username + " da ton tai");
76                      request.setAttribute("ERRORS", errObj);
77                  }
78              }
```

Complete MVC 2
Register Servlet

```
79
80              RequestDispatcher rd = request.getRequestDispatcher(url);
81              rd.forward(request, response);
82          } finally {
83              out.close();
84          }
85      }
86
87      HttpServlet methods. Click on the + sign on the left to edit the code.
125
126 }
127
```

Appendix – JSTL
SQL Tag Library – Example

home.jsp

Source | History

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

JSTL
SQL Tag Library – Example

```
28  <c:set var="username" value="${param.txtUsername}"/>
29  <c:set var="password" value="${param.txtPassword}"/>
30  <c:set var="lastname" value="${param.txtLastname}"/>
31  <c:if test="${not empty username and not empty password and not empty lastname }">
32      <c:catch var="ex">
33          <sql:setDataSource var="con"
34                             driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
35                             url="jdbc:sqlserver://localhost:1433;databaseName=Sinhvien;instanceName=SQL2005"
36                             user="sa" password="trongkhanh"/>
37          <c:if test="${not empty con}">
38              <sql:update dataSource="${con}">
39                  Insert into Registration(username, password, lastname, isAdmin) values (?, ?, ?, 0)
40                  <sql:param value="${username}"/>
41                  <sql:param value="${password}"/>
42                  <sql:param value="${lastname}"/>
43              </sql:update>
44          </c:if>
45      </c:catch>
46      <c:if test="${not empty ex}">
47          <h3>
                <font color="red">
49              Errors occur in Inserting:<br/>
50              ${ex}
51              </font>
52          </h3>
53      </c:if>
54  </c:if>
```

JSTL
SQL Tag Library – Example

```
56      <c:set var="name" value="${param.txtName}"/>
57      <c:catch var="ee">
58          <sql:setDataSource var="con"
59                             driver="com.microsoft.sqlserver.jdbc.SQLServerDriver"
60  url="jdbc:sqlserver://localhost:1433;databaseName=Sinhvien;instanceName=SQL2005"
61                             user="sa" password="trongkhanh"/>
62          <c:if test="${not empty con}">
63              <sql:query var="rs" dataSource="${con}">
64                  Select * From Registration Where lastname Like ?
65                  <sql:param value="%${name}%"/>
66              </sql:query>
67              <c:if test="${not empty rs}">
                    <table border="1">
69                      <thead>
70                          <tr>
71                              <th>No.</th>
72                              <th>Username</th>
73                              <th>Password</th>
74                              <th>Lastname</th>
75                              <th>Roles</th>
76                          </tr>
77                      </thead>
```

JSTL
SQL Tag Library – Example

```
78                      <tbody>
79                          <c:forEach var="record" items="${rs.rows}" varStatus="counter">
80                              <tr>
81                                  <td>${counter.count}</td>
82                                  <td>${record.username}</td>
83                                  <td>${record.password}</td>
84                                  <td>${record.lastname}</td>
85                                  <td>${record.isAdmin}</td>
86                              </tr>
87                          </c:forEach>
88                      </tbody>
89                  </table>
90
91              </c:if>
92          </c:if>
93      </c:catch>
94      <c:if test="${not empty ee}">
95          <h3>
              <font color="red">
97            Errors occur:<br/>
98            ${ee}
99            </font>
100         </h3>
101     </c:if>
```

JSTL
SQL Tag Library – Example

```
102         <h1>Create new account</h1>
103         <form action="home.jsp" method="post">
104             Username <input type="text" name="txtUsername" value="" /><br/>
105             Password <input type="password" name="txtPassword" value="" /><br/>
106             Lastname <input type="text" name="txtLastname" value="" /><br/>
107             <input type="submit" value="Create New Account" />
108         </form>
109     </body>
110 </html>
```

JSTL
SQL Tag Library – Example – Improvement

```
<table border="1">
    <thead>
        <tr>
            <th>No.</th>
            <c:forEach var="columnName" items="${rs.columnNames}">
                <th>${columnName}</th>
            </c:forEach>
        </tr>
    </thead>
    <tbody>
        <c:forEach var="row" items="${rs.rowsByIndex}" varStatus="counter">
            <tr>
                <td>${counter.count}</td>
                <c:forEach var="field" items="${row}">
                    <td>${field}</td>
                </c:forEach>
            </tr>
    </c:forEach>
</tbody>
```

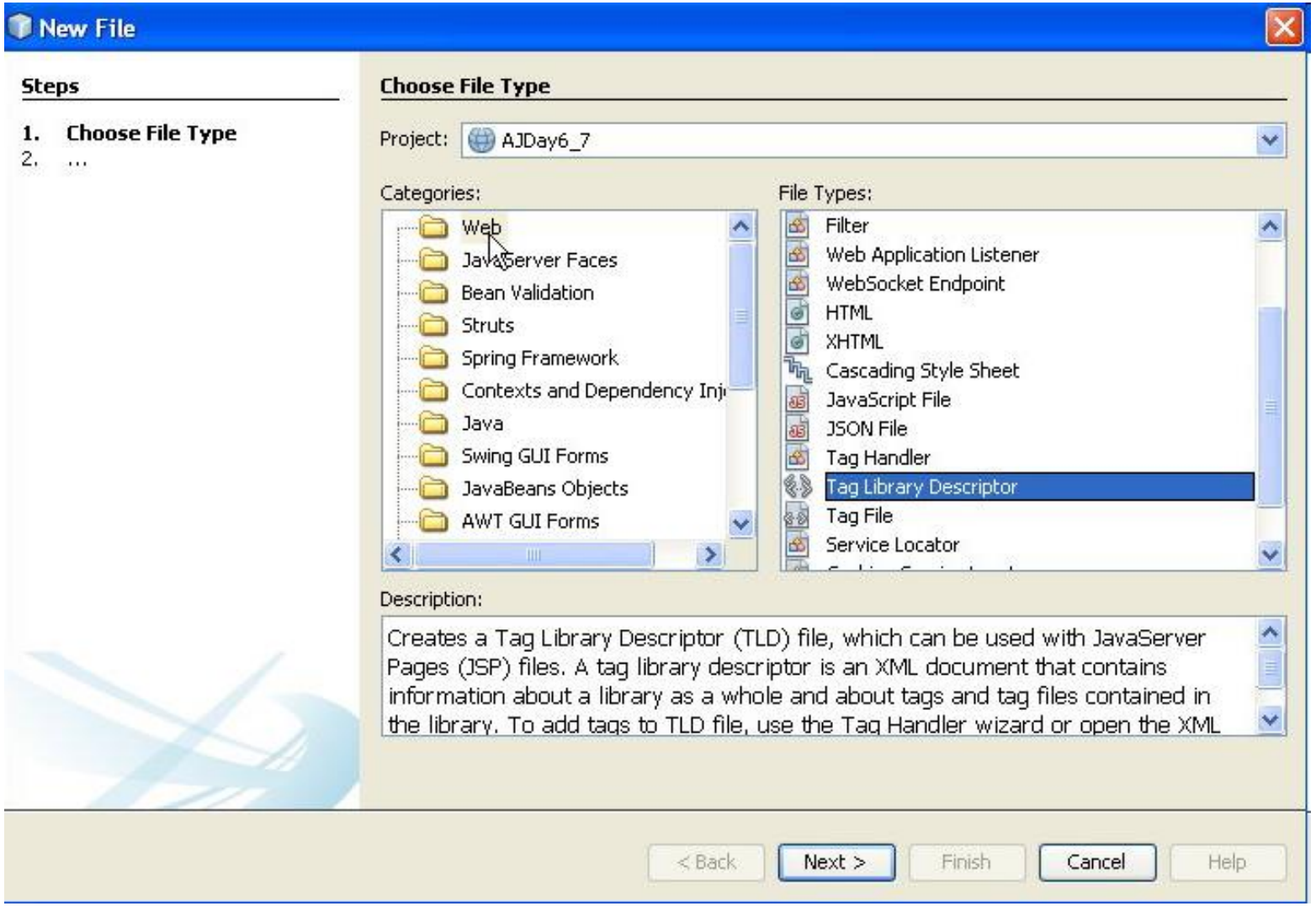
JSTL

SQL Tag Library – Example – Improvement

No.	username	password	lastname	isAdmin
1	1008	333	345	
2	aptech	aptech1	0000	false
3	Danh Vong	12345	Danh Dai Ca	false
4	HieuLegend	legend1345	Hieu kk	false
5	java7	8765	JDK 7	false
6	khanh	kieu123	kieu	true
7	kieukhanh	123445	Khanh Kieu Trong	true
8	Phong	123456	Phong cui	false

Appendix – Tag Implementation

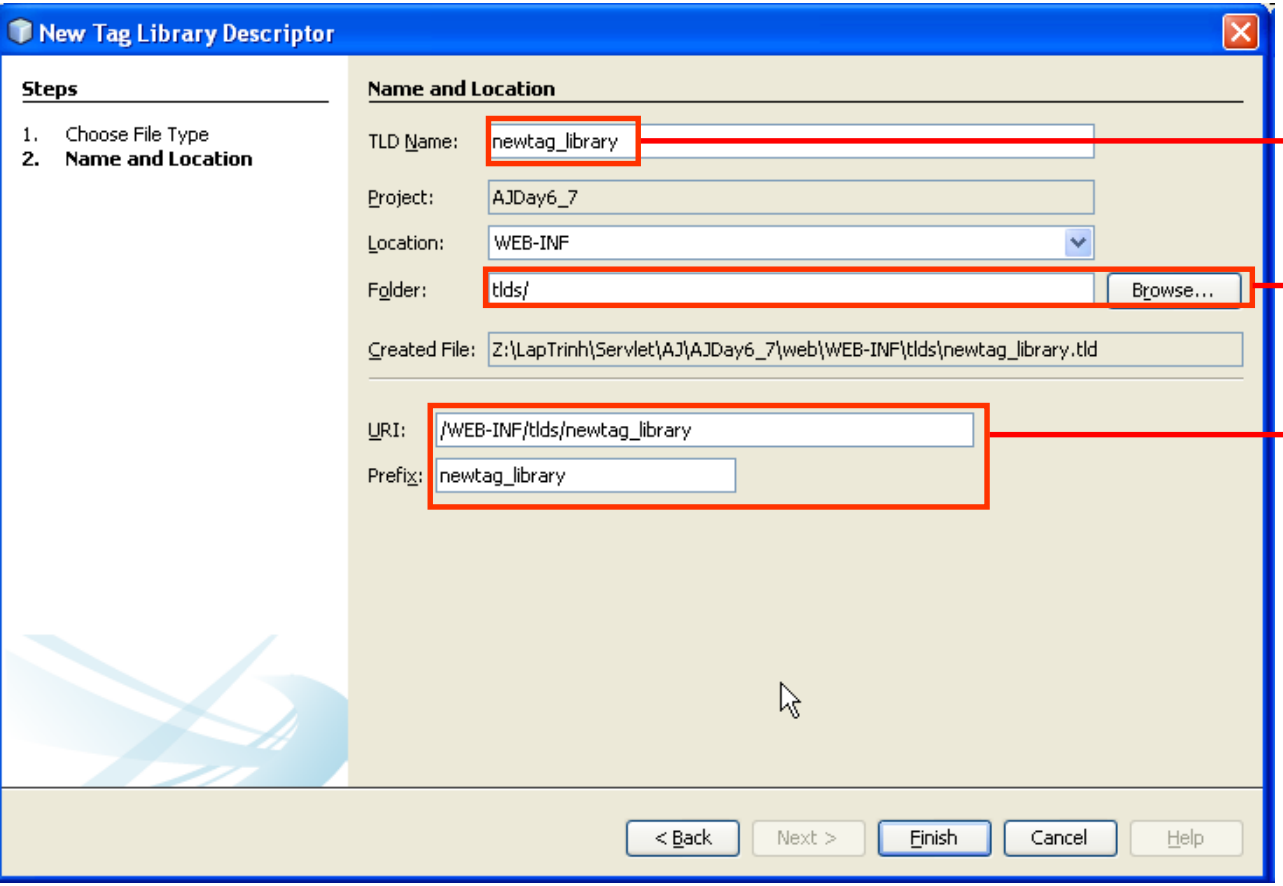
Empty Custom Tags – Create TLD



- Click Next Button

Tag Implementation

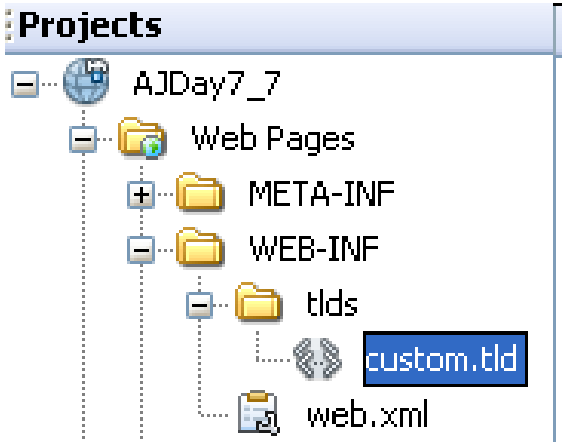
Empty Custom Tags – Create TLD



Fill your tld name

Modify or Browse the location that is used to store tag lib.

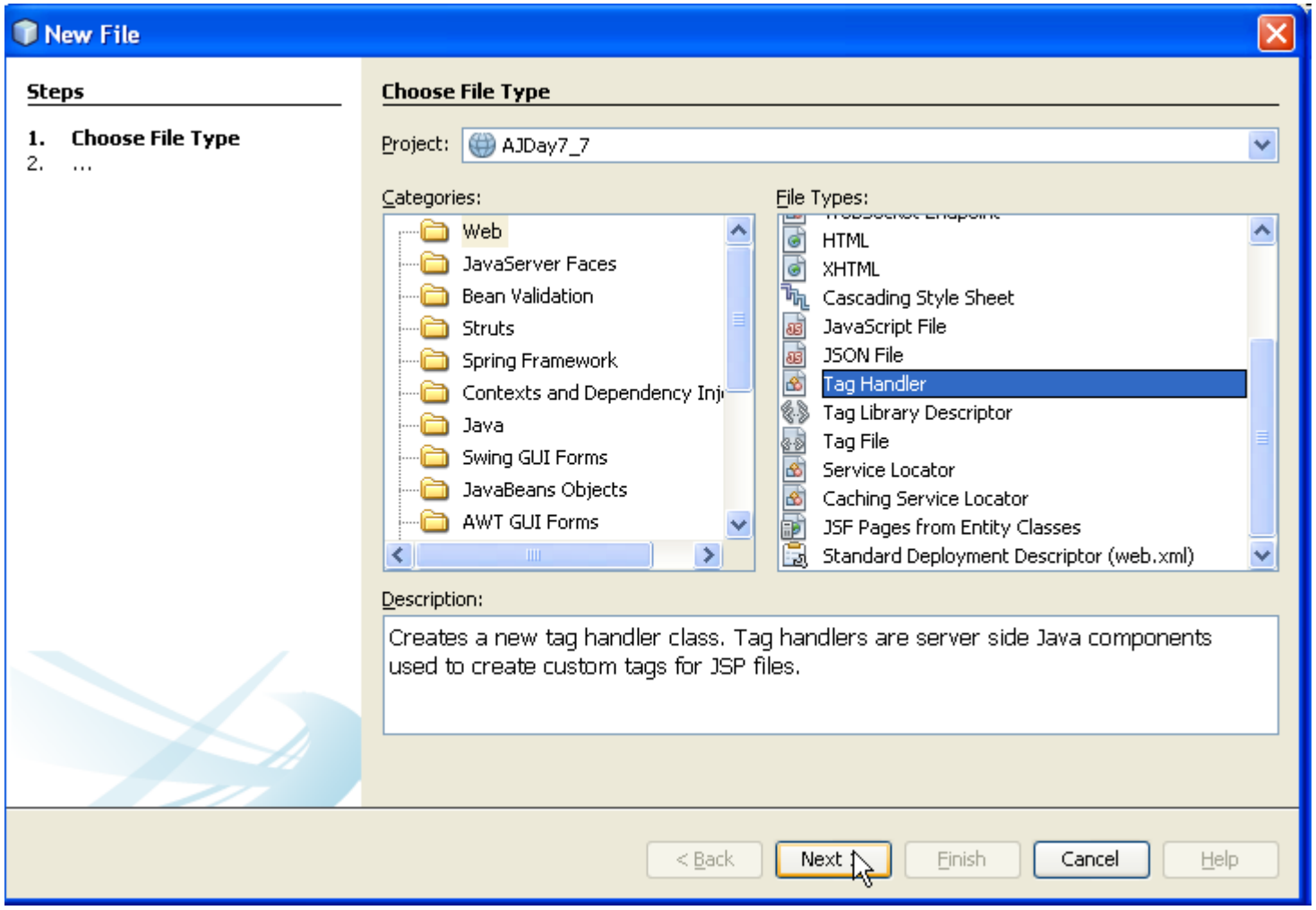
Modify the URI or prefix (shortname)



- Click **Finish** Button
- **Then**, the **tld** file is **automatically created** in **tlds directory**
- Do not modify anything in this tld file because the netbeans is automatically the tld information when the next step is executed

Tag Implementation

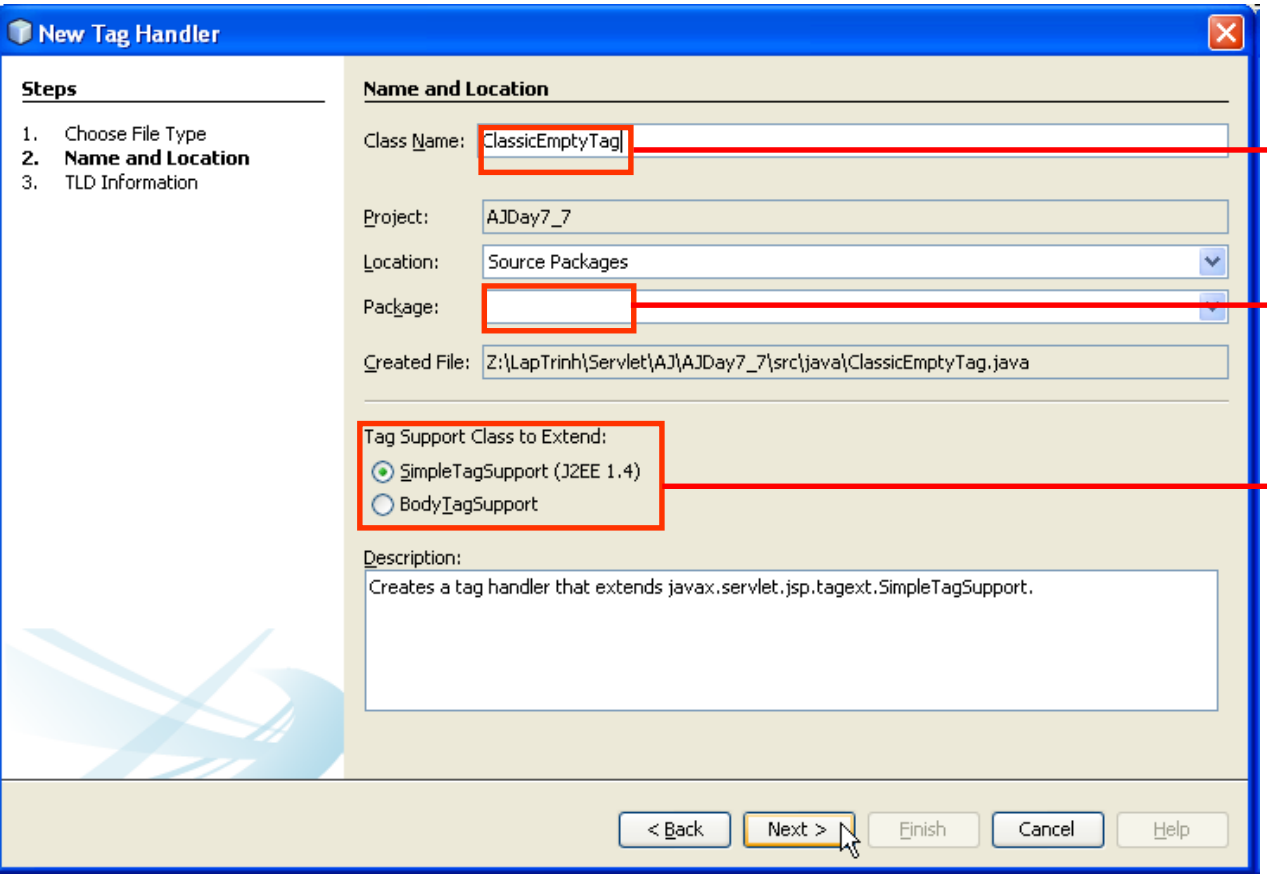
Empty Custom Tags – Create Tag Handler



- Click Next Button

Tag Implementation

Empty Custom Tags – Create Tag Handler



Fill your tld name

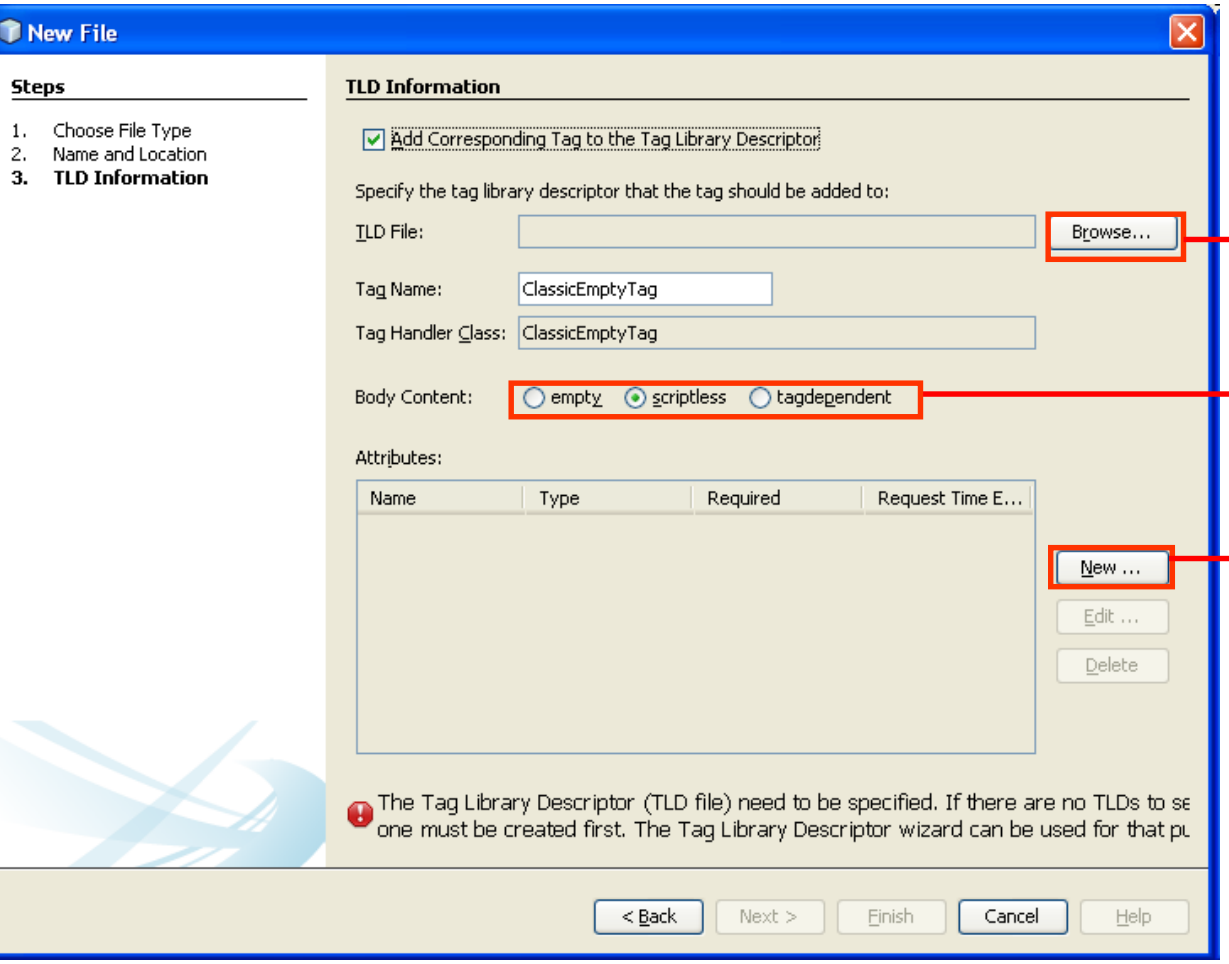
Choose or type the package

Choose Simple Tag Support

- Click Next Button

Tag Implementation

Empty Custom Tags – Create Tag Handler



Browse to choose the tld file that will be updated information

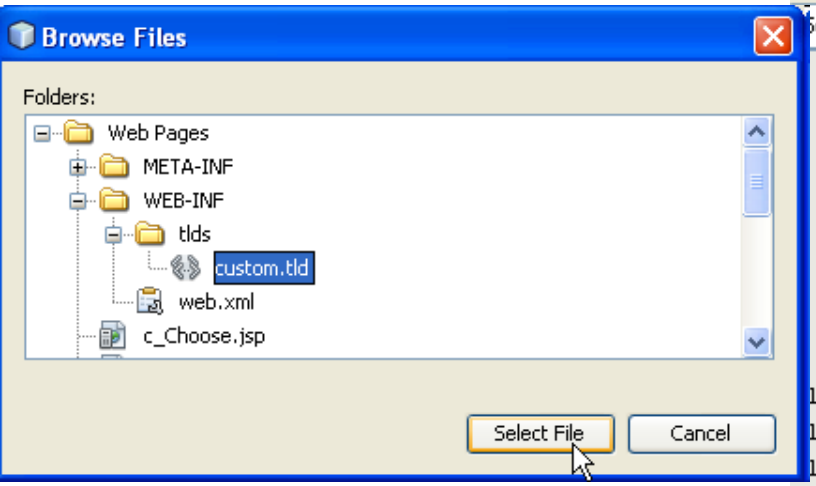
Choose the Body Tag, choose empty

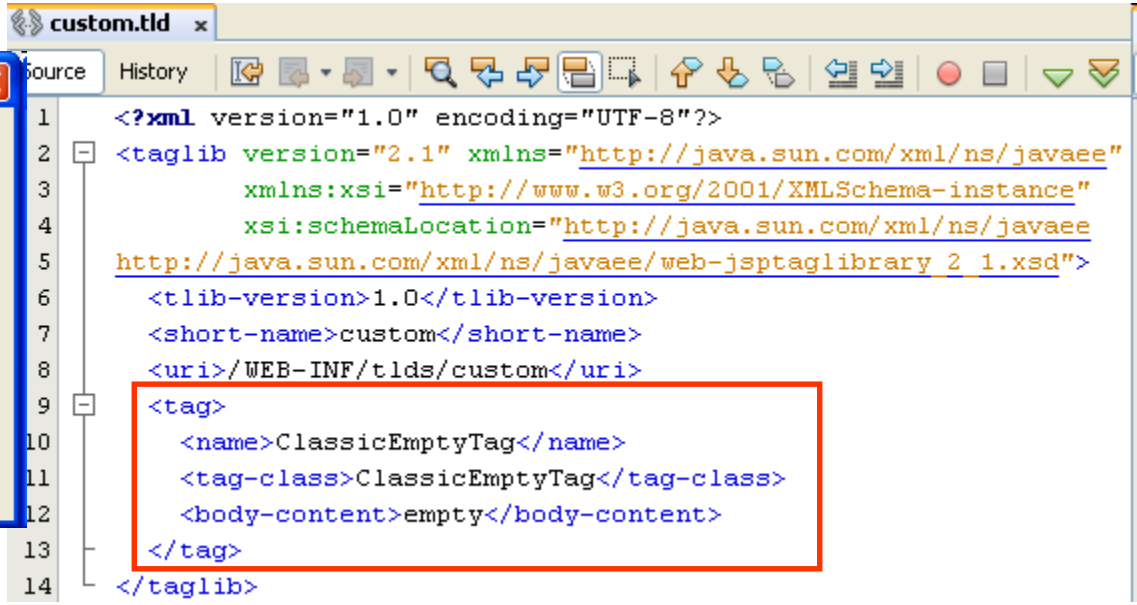
Add attribute to custom tag

- Click Browse Button to add the information to tld

Tag Implementation

Empty Custom Tags – Create Tag Handler





- Choose the tld file
- Then, Click Select File Button, then choose the empty option
- Click Finish Button on New Files Diaglog
- The TagHandler Class is created and the custom tag information is automatically updated in the selected tld file

Tag Implementation

Empty Custom Tags – Create Tag Handler

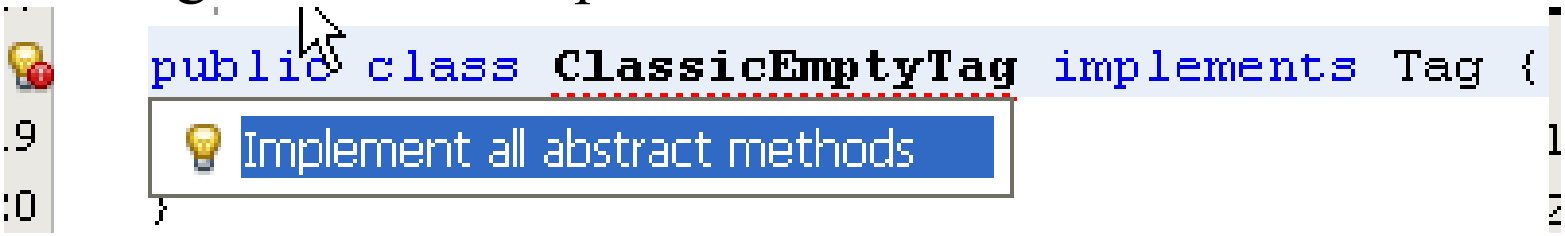
- To create emptyTag, clear the body contents in class

```
public class ClassicEmptyTag extends SimpleTagSupport {

}
```

- Change "extends SimpleTagSupport" to "implements Tag"

```
public class ClassicEmptyTag implements Tag {

}
```

- Click "Light Bulb" to implement all abstract methods



- Coding the doStartTag, doEndTag method
- Using the tag on the Jsp with taglib directive

Tag Implementation

Empty Custom Tags – Create Tag Handler

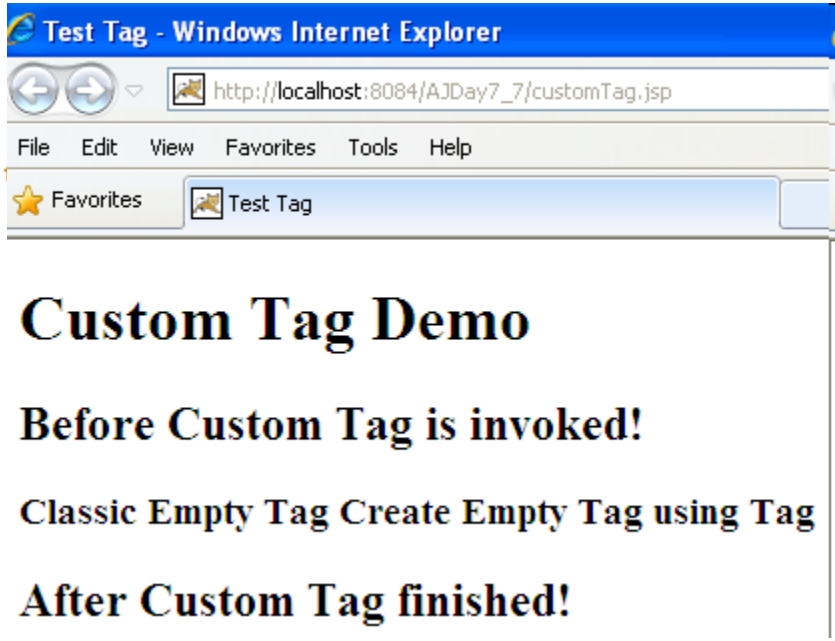
```
20  public class ClassicEmptyTag implements Tag {
21      private PageContext pageContext;
22      private Tag parent;
23      public void setPageContext(PageContext pc) {
24          this.pageContext = pc;
25      }
26      public void setParent(Tag t) {
27          this.parent = t;
28      }
29      public Tag getParent() {
30          return this.parent;
31      }
32      public int doStartTag() throws JspException {
33          try {
34              pageContext.getOut().println("Create Empty Tag using Tag");
35          }catch(IOException e){
36              e.printStackTrace();
37          }
38          return SKIP_BODY;
39      }
40      public int doEndTag() throws JspException {
41          return EVAL_PAGE;
42      }
43      public void release() {
44          System.out.println("Garbage Collection is invoked");
45      }
46  }
```

Tag Implementation

Empty Custom Tags – Reference

- Compile the TagHandler class
- Then create the JSP page with following contents

```
<%@taglib uri="/WEB-INF/tlds/custom" prefix="ct" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="tex
        <title>Test Tag</title>
    </head>
    <body>
        <h1>Custom Tag Demo</h1>
        <h2>Before Custom Tag is invoked!</h2>
        <h3>Empty Tag <ct:ClassicEmptyTag/></h3>
        <h2>After Custom Tag finished!</h2>
    </body>
</html>
```



- Change the EVAL_PAGE to SKIP_PAGE in doEndTag()

Appendix

Iteration Tag Interface – Example

```
public class ClassicInterationTag implements IterationTag {
    private String count;
    private PageContext pageContext;
    private Tag parent;
    private int n;
    private BodyContent bodyContent;
    public void setCount(String count) {
        this.count = count;
        try {
            n = Integer.parseInt(this.count);
        } catch(NumberFormatException e){
            n=10;
        }
    }
    public int doAfterBody() throws JspException {
        if(n>1){
            n--;
            return EVAL_BODY_AGAIN;//forces invoking doStartTag again
        } else {
            return SKIP_BODY;
        }
    }
    public void setPageContext(PageContext pc) {...}
    public void setParent(Tag t) {...}
    public Tag getParent() {...}
```

Appendix

Iteration Tag Interface – Example

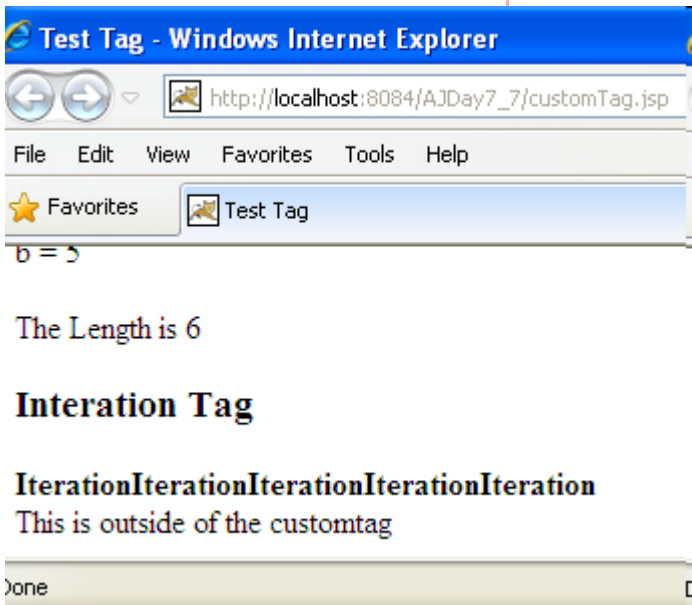
```
    public int doStartTag() throws JspException {
        if(n>0){
            return EVAL_BODY_INCLUDE;
        } else {
            return SKIP_BODY;
        }
    }
    public int doEndTag() throws JspException {
        try {
            if(bodyContent!=null){
                bodyContent.writeOut(bodyContent.getEnclosingWriter());
            }
        } catch(IOException e){
            throw new JspException(e.getMessage());
        }
        return EVAL_PAGE;
    }
    public void release() {...}

    public void setBodyContent(BodyContent bodyContent) {
        this.bodyContent = bodyContent;
    }
}
```

Appendix

Iteration Tag Interface – Example

```
<tag>
  <name>ClassicInterationTag</name>
  <tag-class>sample.custtag.ClassicInterationTag</tag-class>
  <body-content>scriptless</body-content>
  <attribute>
    <name>count</name>
    <required>true</required>
    <type>java.lang.String</type>
  </attribute>
</tag>
```

```
<h3>Interation Tag</h3>
<ct:ClassicInterationTag count="5"><b>Iteration</b></ct:ClassicInterationTag>
<br/>This is outside of the customtag <br/>
```



Appendix

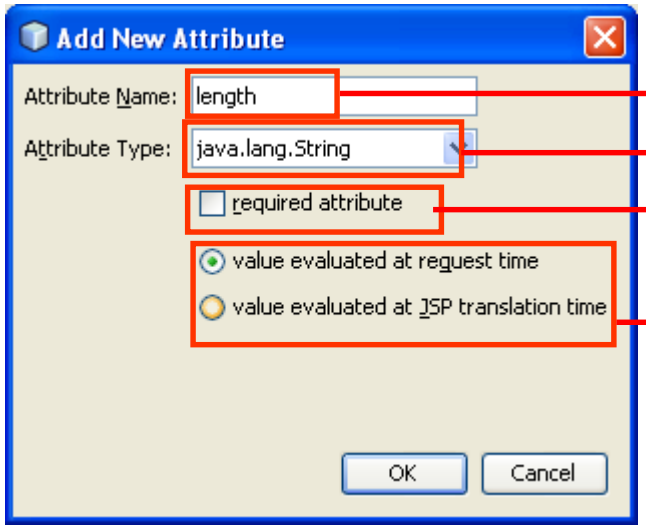
Custom Tags with Attributes – TagSupport

- **<prefix:name attr=”value”></prefix:name >**
- Tags **with or without** body **has attributes**
- Are called as **parameterized tags**.
- Attributes are passed to the tags **as arguments** are passed to method.
- Is done to customize the behavior of a custom tag.

Appendix

Custom Tags with Attributes – TagSupport

- After browsing to tld finish, choose the New button to Add Attribute to custom Tag



Fill the attribute name

Choose the datatype for attribute

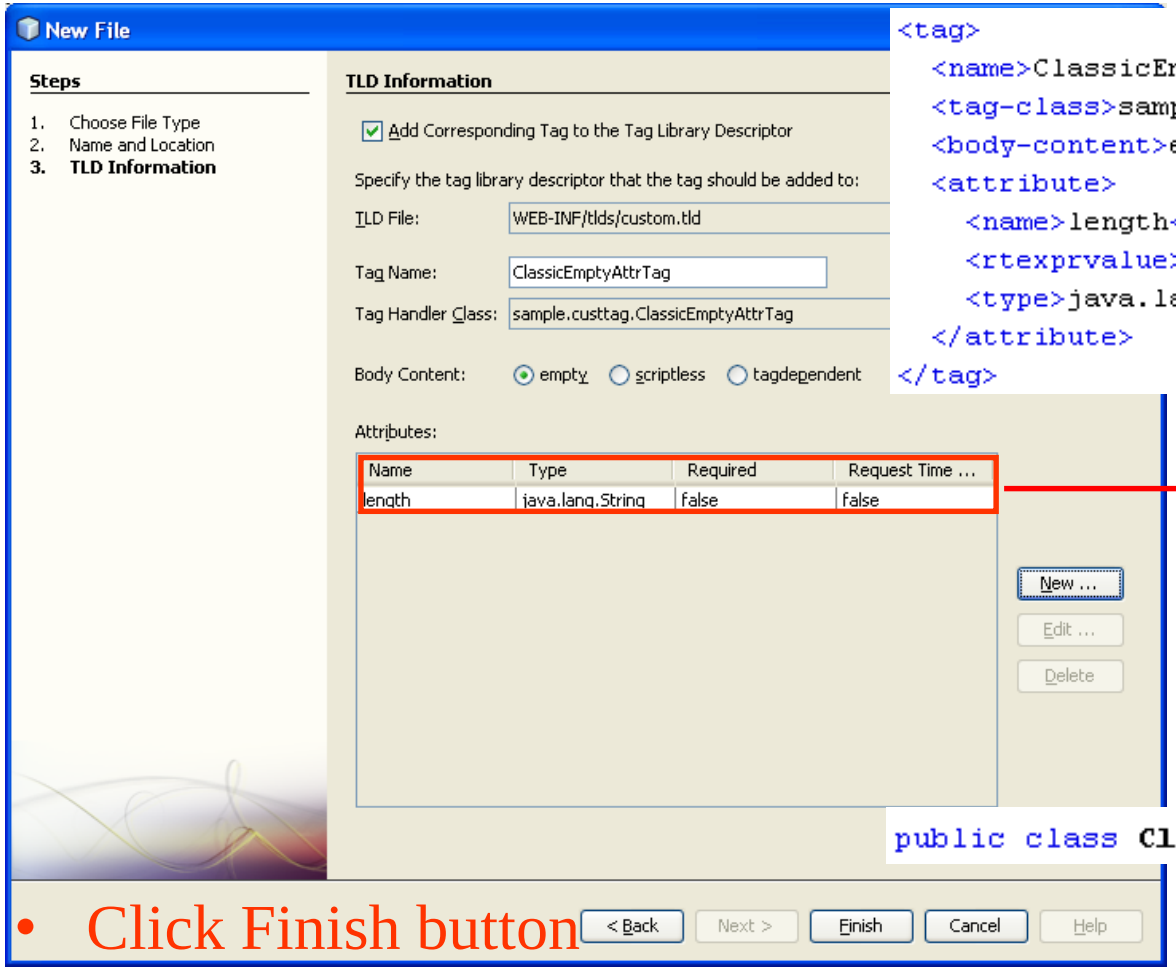
Checking if the attribute is required

Choose type for evaluated the attribute

- Click OK button

Appendix

Custom Tags with Attributes – TagSupport




```
<tag>
  <name>ClassicEmptyAttrTag</name>
  <tag-class>sample.custtag.ClassicEmptyAttrTag</tag-class>
  <body-content>empty</body-content>
  <attribute>
    <name>length</name>
    <rtexprvalue>true</rtexprvalue>
    <type>java.lang.String</type>
  </attribute>
</tag>
```

Attribute to add after the above step

```
public class ClassicEmptyAttrTag extends TagSupport {
```

- Click Finish button
- The attribute & tag information is automatically updated to the tld file
- Change the SimpleTagSupport to TagSupport in the TagHandler

Appendix

Custom Tags with Attributes – TagSupport

ClassicEmptyAttrTag.java

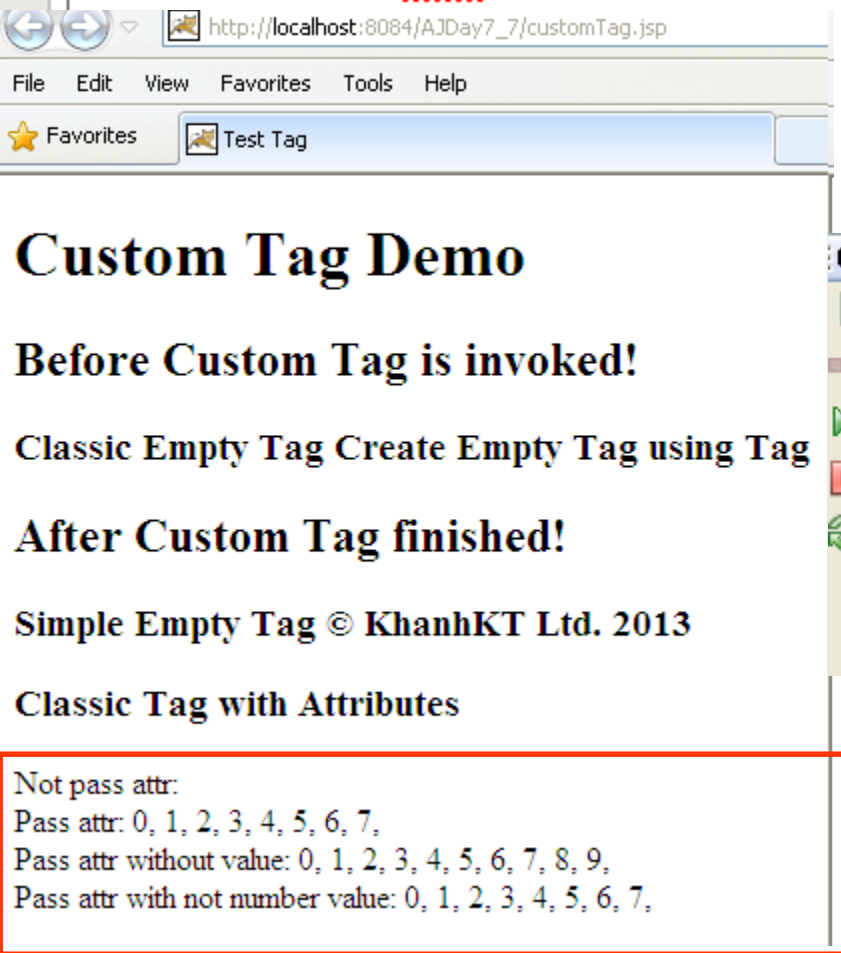
Source | History

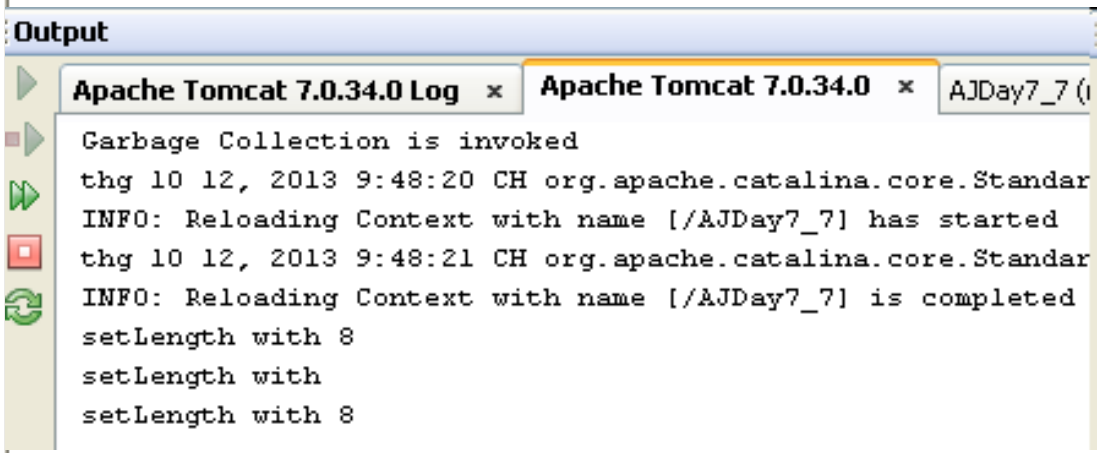
```
14      * @author Trong Khanh
15      */
16     public class ClassicEmptyAttrTag extends TagSupport {
17         private String length;
18         private int n;
19         public void setLength(String length) {
20             this.length = length;
21             System.out.println("setLength with " + length);
22             try {
23                 n = Integer.parseInt(this.length);
24             }catch(NumberFormatException e){
25                 n = 10;
26             }
27         }
28         public int doStartTag() throws JspException {
29             JspWriter out = pageContext.getOut();
30             try {
31                 for(int i=0; i<n; i++){
32                     out.print(i + ", ");
33                 }
34             } catch(IOException e){
35                 e.printStackTrace();
36             }
37             return SKIP_BODY;
38         }
39     }
```

Appendix

Custom Tags with Attributes – TagSupport

```
21          <h3>Classic Tag with Attributes</h3>
22          Not pass attr: <ct:ClassicEmptyAttrTag/><br/>
23          Pass attr: <ct:ClassicEmptyAttrTag length="8"/> <br/>
24          Pass attr without value: <ct:ClassicEmptyAttrTag length=""/> <br/>
25          Pass attr with not number value: <ct:ClassicEmptyAttrTag length="8"/> <br/>
```





Appendix

Custom Tags with Body – BodyTag

```
public class ClassicBodyTag extends BodyTagSupport {

    /**...*/
    public int doAfterBody() throws JspException {
        try {
            // This code is generated for tags whose bodyContent is "JSP"
            BodyContent bodyCont = getBodyContent();
            JspWriter out = bodyCont.getEnclosingWriter();
            String bodyString = bodyCont.getString();
            out.println(bodyString.toUpperCase());
            bodyCont.clear();
            writeTagBodyContent(out, bodyCont);
        } catch (Exception ex) {
            handleBodyContentException(ex);
        }
        if (theBodyShouldBeEvaluatedAgain()) {
            return EVAL_BODY_AGAIN;
        } else {
            return SKIP_BODY;
        }
    }
```

Appendix

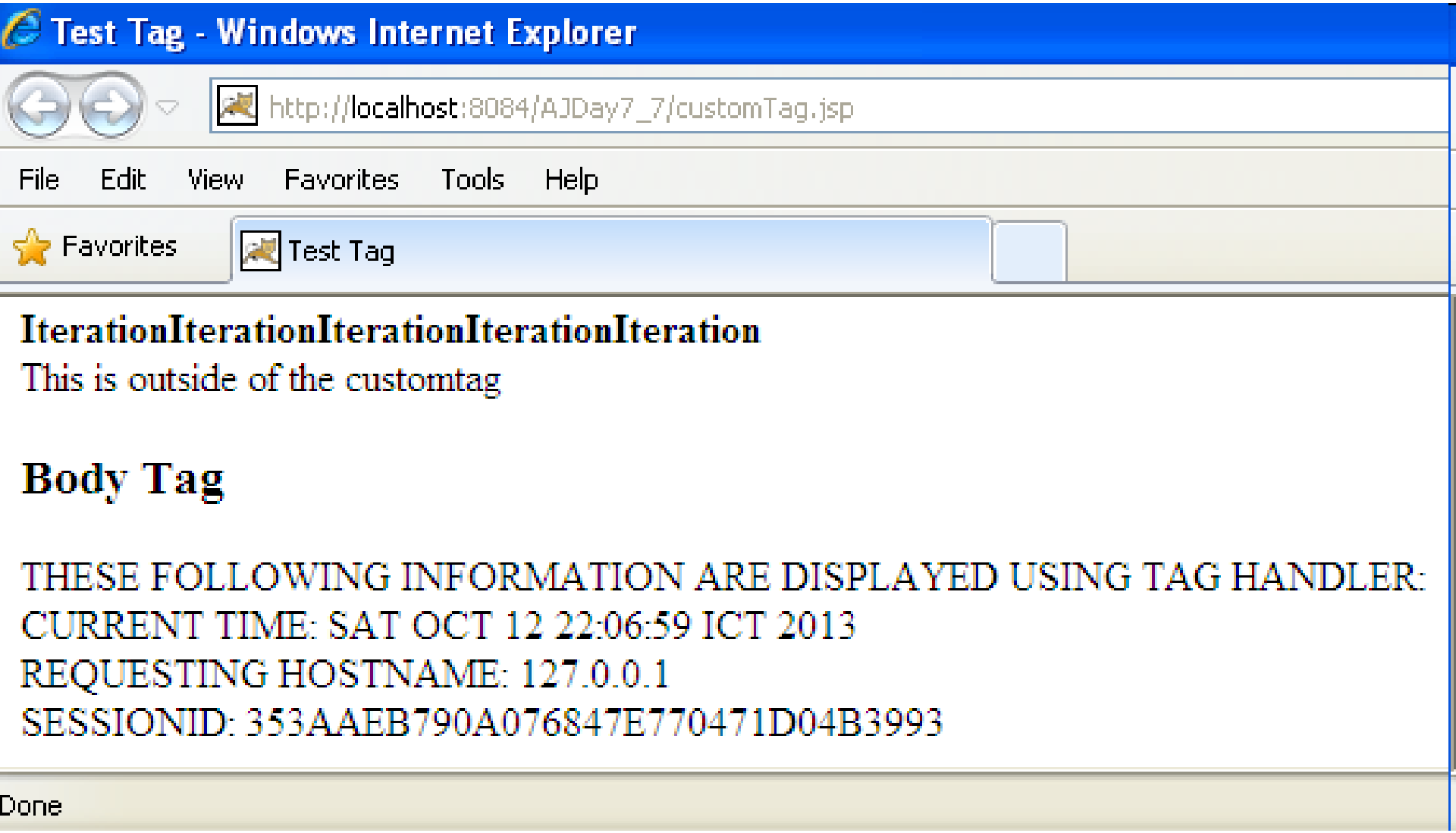
Custom Tags with Body – BodyTag

```
<tag>
  <name>ClassicBodyTag</name>
  <tag-class>sample.custtag.ClassicBodyTag</tag-class>
  <body-content>JSP</body-content>
</tag>
```

```
<h3>Body Tag</h3>
<ct:ClassicBodyTag>
    These following information are displayed using tag handler:
    <li>Current Time: <%= new java.util.Date()%></li>
    <li>Requesting hostname: <%= request.getRemoteHost()%></li>
    <li>SessionID: <%= session.getId()%></li>
</ct:ClassicBodyTag>
```

Appendix

Custom Tags with Body – BodyTag



Appendix

Empty Custom Tags – using Simple

SimpleEmptyTag.java

Source | History

```
14      * @author Trong Khanh
15      */
16     public class SimpleEmptyTag extends SimpleTagSupport {
17
18         /**...*/
23         @Override
24         public void doTag() throws JspException {
25             JspWriter out = getJspContext().getOut();
26             try {
27                 out.println("<b>&copy; KhanhKT Ltd. 2013</b>");
28
29                 JspFragment f = getJspBody();
30                 if (f != null) f.invoke(out);
31
32             } catch (java.io.IOException ex) {
33                 throw new JspException("Error in SimpleEmptyTag tag", ex);
34             }
35         }
36     }
```

```
<tag>
  <name>ClassicEmptyTag</name>
  <tag-class>sample.custtag.ClassicEmptyTag</tag-class>
  <body-content>empty</body-content>
</tag>
<tag>
  <name>SimpleEmptyTag</name>
  <tag-class>sample.custtag.SimpleEmptyTag</tag-class>
  <body-content>empty</body-content>
</tag>
```

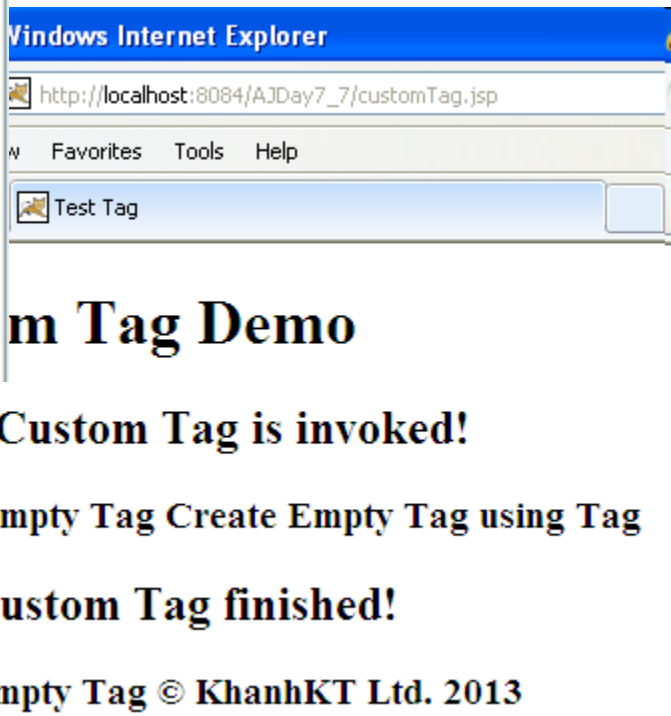
Appendix

Empty Custom Tags – using Simple

customTag.jsp

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/custom.tld" prefix="ct"%>
10  <html>
11      <head>
12          <meta http-equiv="Content-Type" content="text/htr
13          <title>Test Tag</title>
14      </head>
15      <body>
16          <h1>Custom Tag Demo</h1>
17          <h2>Before Custom Tag is invoked!</h2>
18          <h3>Classic Empty Tag <ct:ClassicEmptyTag/></h3>
19          <h2>After Custom Tag finished!</h2>
20          <h3>Simple Empty Tag <ct:SimpleEmptyTag/></h3>
```



Appendix

Custom Tags with Attributes – using Simple

- **Adding dynamic attributes** to SimpleTag
  - Implementing **javax.servlet.jsp.tagext.DynamicAttributes** interface
  - Define the **setDynamicAttribute**() method that accepts dynamic attributes is **passed an attribute** not present in the tag library

public void setDynamicAttribute(String uri, String localName, Object value) throws JspException

  - Addition the <**dynamic-attributes**>**element** to true in tld

Appendix

Custom Tags with Attributes – using Simple

Source | History

```
16      * @author Trong Khanh
17      */
18     public class SimpleDynaAttrTag extends SimpleTagSupport
19                                    implements DynamicAttributes{
20         private String length;
21         private ArrayList key = new ArrayList();
22         private ArrayList value = new ArrayList();
23         @Override
24         public void doTag() throws JspException {
25             JspWriter out = getJspContext().getOut();
26             try {
27                 for(int i=0; i<key.size(); i++){
28                     String strKey = key.get(i).toString();
29                     String strValue = value.get(i).toString();
30                     out.println("<li>" + strKey + " = " + strValue + "</li>");
31                 }
32                 out.print("<br/>The Length is " + length);
33                 JspFragment f = getJspBody();
34                 if (f != null) f.invoke(out);
35             } catch (java.io.IOException ex) {
36                 throw new JspException("Error in SimpleDynaAttrTag tag", ex);
37             }
38         }
39         public void setLength(String length) {...}
43         public void setDynamicAttribute(String uri, String localName, Object value)
44                 throws JspException {
45             this.key.add(localName);
46             this.value.add(value);
47         }
```

Appendix

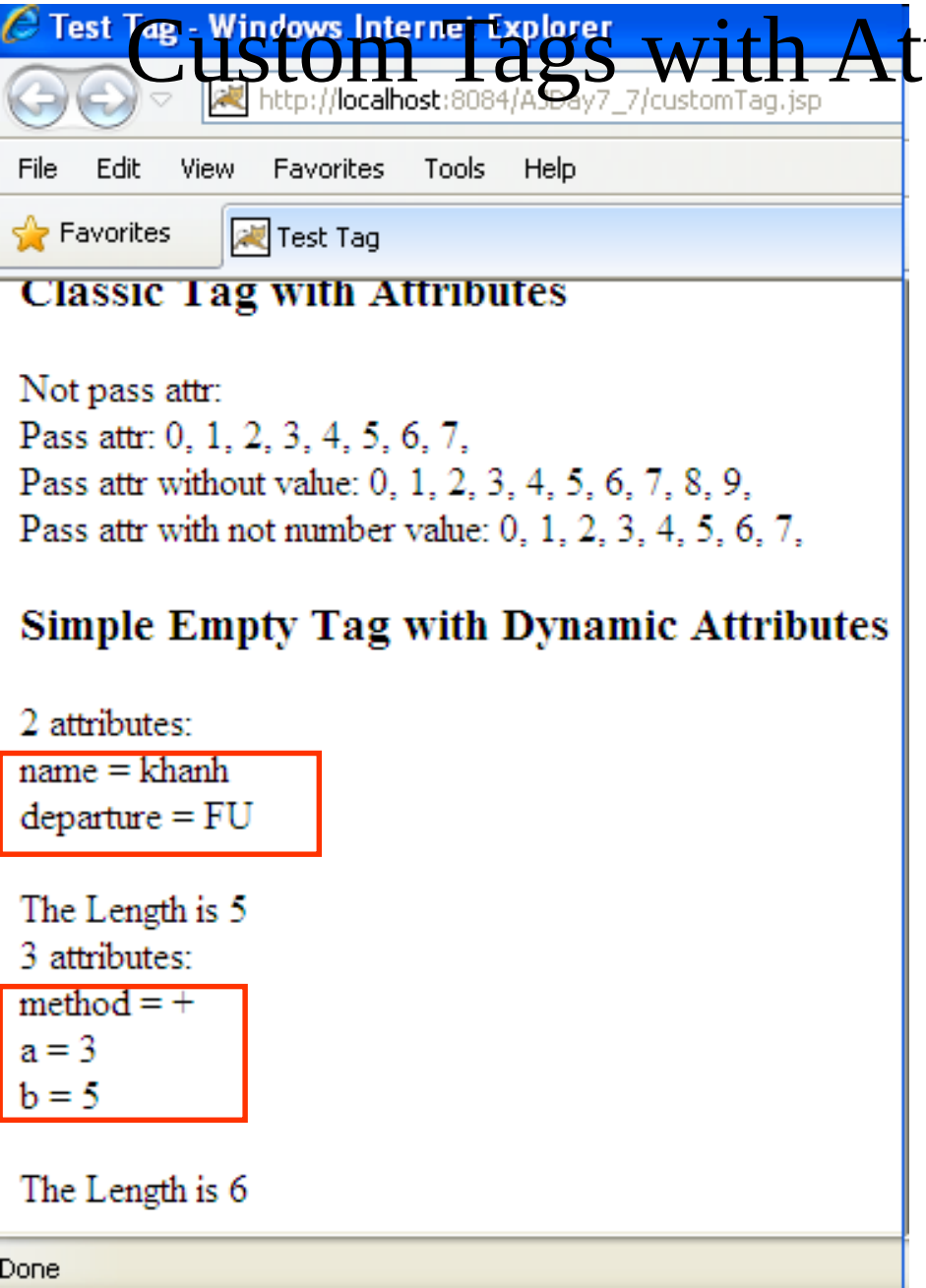
Custom Tags with Attributes – using Simple

```
44    <tag>
45      <name>SimpleDynaAttrTag</name>
46      <tag-class>sample.taglib.SimpleDynaAttrTag</tag-class>
47      <body-content>empty</body-content>
48      <dynamic-attributes>true</dynamic-attributes>
49      <attribute>
50        <name>length</name>
51        <required>true</required>
52        <rtexprvalue>true</rtexprvalue>
53        <type>java.lang.String</type>
54      </attribute>
55    </tag>
```

```
26      <h3>Simple Empty Tag with Dynamic Attributes</h3>
27      2 attributes: <ct:SimpleDynaAttrTag name="khanh" departure="FU" length="5"/><br/>
28      3 attributes: <ct:SimpleDynaAttrTag method="+" a="3" b="5" length="6"/>
```

Appendix

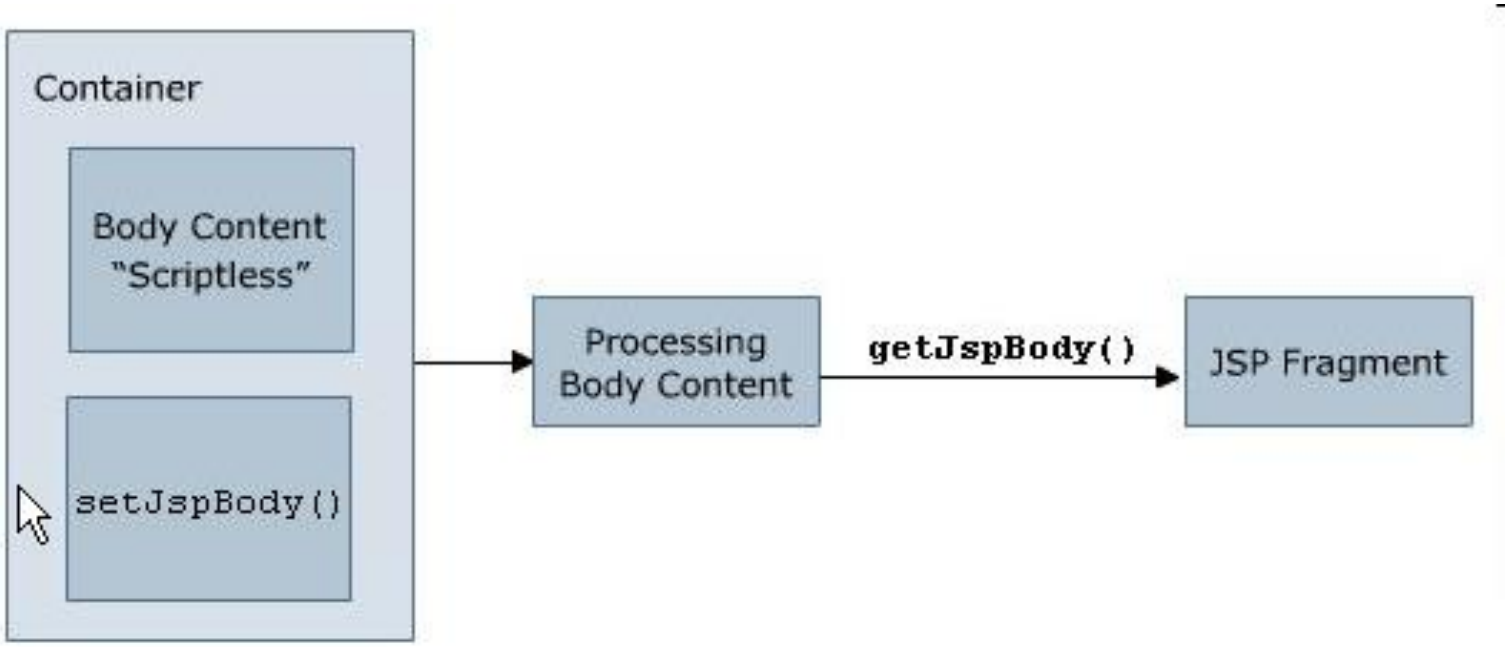
Custom Tags with Attributes – using Simple



Appendix

Custom Tags with Body – SimpleTag

- The **setJspBody**() method is called by the container. By the use of **getJspBody**() method, **fragment** can be **accessed** and is **executed** using the **invoke**() **method**.
- **Finally**, with the help of **out.println**(), the processed body content is **sent** to the **output stream** as a **response**.



Appendix

Custom Tags with Body – SimpleTag

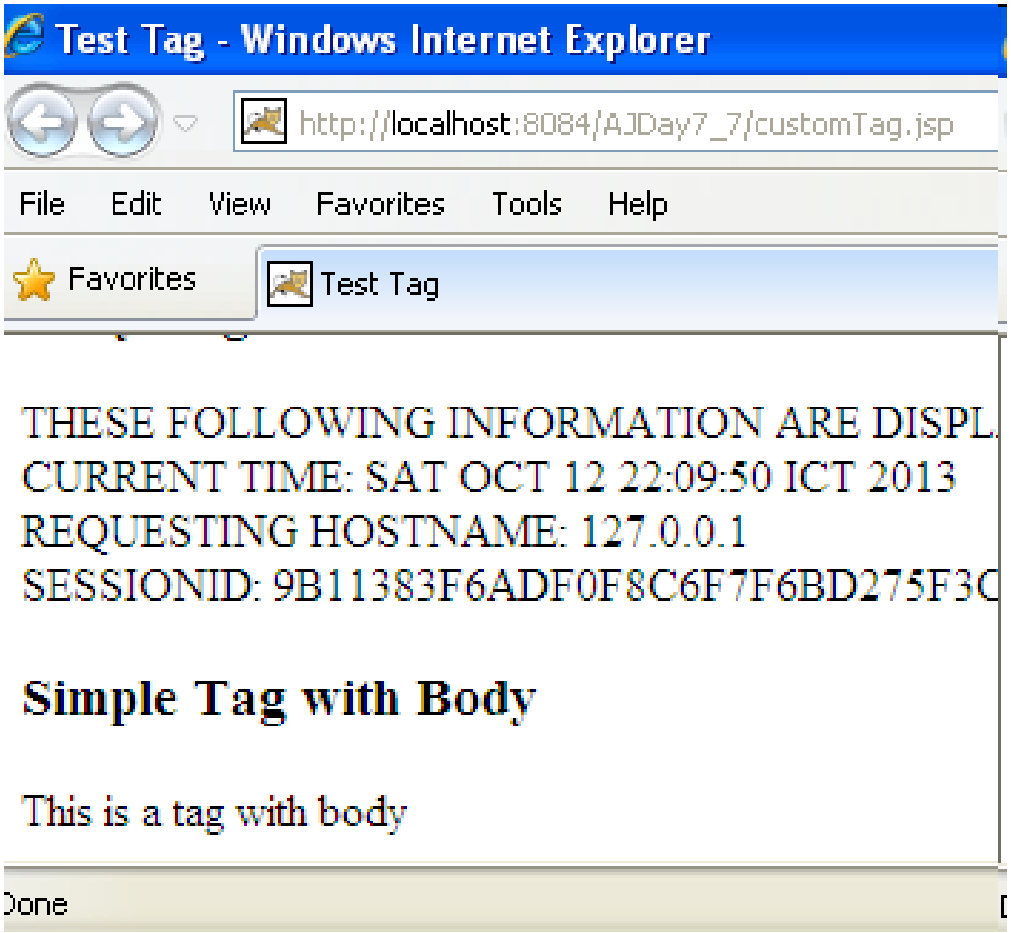
```
<tag>
   <name>SimpleBodyTag</name>
   <tag-class>sample.custtag.SimpleBodyTag</tag-class>
   <body-content>scriptless</body-content>
</tag>
```

```
public class SimpleBodyTag extends SimpleTagSupport {
    /**...*/
    public void doTag() throws JspException {
        JspWriter out = getJspContext().getOut();
        try {
            JspFragment f = getJspBody();
            if (f != null) f.invoke(out);
        } catch (java.io.IOException ex) {
            throw new JspException("Error in SimpleBodyTag tag", ex);
        }
    }

}
```

Appendix

Custom Tags with Body – SimpleTag

```
<h3>Simple Tag with Body</h3>
<ct:SimpleBodyTag>This is a tag with body</ct:SimpleBodyTag>
```



Appendix - The Tag File Model

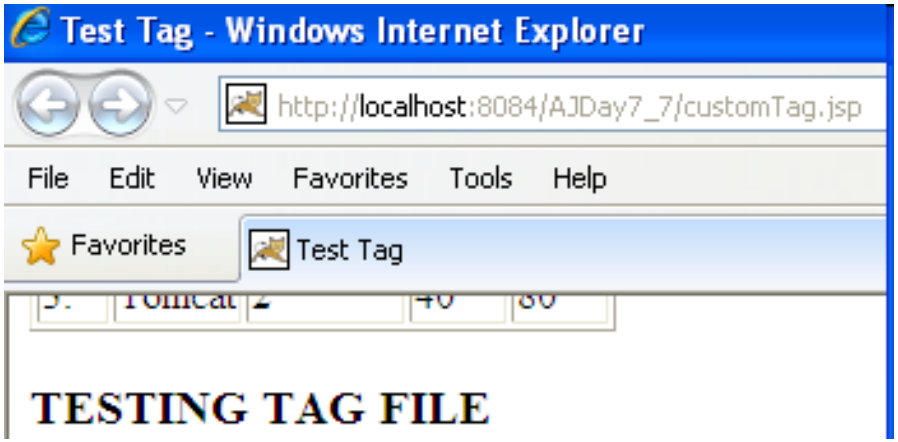
Implementation

upperTag.tag

Source | History

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@tag description="put the tag description here" pageEncoding="UTF-
8
9   <%-- The list of normal or fragment attributes can be specified here
10  <%@tag body-content="scriptless" %>
11  <jsp:doBody var="theBody" scope="session"/>
12  <% String bodyContent = (String)session.getAttribute("theBody"); %>
13  <%= bodyContent.toUpperCase() %>
```

```
<h3><tag:upperTag>Testing Tag file</tag:upperTag></h3>
```



Appendix - The Tag File Model

Implementation – Dynamic Attributes

dynaAttrTagfile.tag

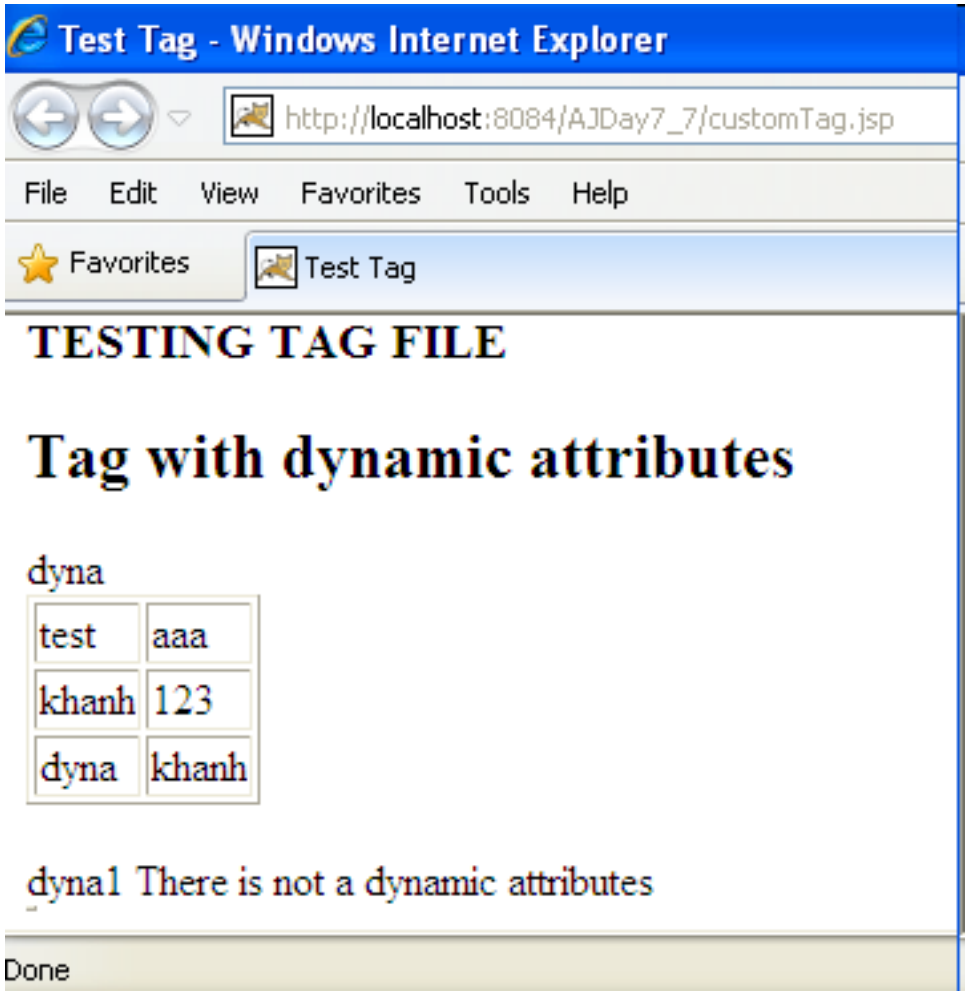
Source | History

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@tag description="put the tag description here" pageEncoding="UTF-8"%>
8   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
9   <%-- The list of normal or fragment attributes can be specified here: --%>
10  <%@attribute name="header1" required="true"%>
11  <%@tag dynamic-attributes="dynaAttrs" %>
12
13  ${header1}
14  <c:if test="${empty dynaAttrs}">There is not a dynamic attributes</c:if>
15  <table border="1">
16      <c:forEach var="attrs" items="${dynaAttrs}">
17          <tr>
18              <td>${attrs.key}</td>
19              <td>${attrs.value}</td>
20          </tr>
21      </c:forEach>
22  </table>
```

Appendix - The Tag File Model

Implementation – Dynamic Attributes

```
<br/>Tag file with dynamic attributes:<br/>
<tagFile:dynaAttrTagfile header1="dyna" khanh="123" dyna="khanh" test="aaa"/>
<tagFile:dynaAttrTagfile header1="dyna1"/>
```



Appendix – Build Library

Tag File

myDataGrid.tag

Source | History

```
4       Author      : Trong Khanh
5   --%>
6
7   <%@tag description="put the tag description here" pageEncoding="UTF-8"%>
8   <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
9   <%@taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
10
11  <%-- The list of normal or fragment attributes can be specified here: --%>
12  <%@attribute name="dataSource" required="true" rtexprvalue="true"%>
13  <%@attribute name="sql" required="true" %>
14  <%@tag dynamic-attributes="params" %>
15
16  <%-- any content can be specified here e.g.: --%>
17  <c:catch var="ex">
18      <sql:setDataSource var="con" dataSource="${dataSource}"/>
19      <c:if test="${not empty con}">
20          <sql:query var="rs" dataSource="${con}">
21              ${sql}
22              <c:forEach var="par" items="${params}">
23                  <sql:param value="${par.value}"/>
24              </c:forEach>
25          </sql:query>
```

Build Library Tag Lib Library

Tag File

```
26          <c:if test="${not empty rs}">
27              <table border="1">
28                  <thead>
29                      <tr>
30                          <th>No.</th>
31                              <c:forEach var="column" items="${rs.columnNames}">
32                              <th>${column}</th>
33                              </c:forEach>
34                      </tr>
35                  </thead>
36                  <tbody>
37                      <c:forEach var="row" items="${rs.rowsByIndex}" varStatus="counter">
38                          <tr>
39                              <td>${counter.count}</td>
40                              <c:forEach var="field" items="${row}">
41                                  <td>${field}</td>
42                              </c:forEach>
43                          </tr>
44                      </c:forEach>
45
46                  </tbody>
47              </table>
48
49          </c:if>
50      </c:if>
51  </c:catch>
52  <c:if test="${not empty ex}">
53      <font color="red">
54      Errors occur: ${ex}
55      </font>
56  </c:if>
```

Build Library Tag Lib Library

Apply

```
30          <h2>TagFile Demo</h2>
31          <tagFile:myDataGrid dataSource="myDS"
32                              sql="Select * From Registration Where lastname Like ?"
33                              name="%a%"/><br/>
34
35          <tagFile:myDataGrid dataSource="myDS"
36                              sql="Select * From Registration Where username = ? And password = ?"
37                              par1="khanh" par2="kieu123"/><br/>
38      </body>
```

Fpt University

Tag Hierarchies

- **Simple within Simple**
  - Using JSPTag with **getParent**() method
  - Using **instanceOf** to compare
- **Classic within Classic**
  - Using Tag with **getParent**() method
  - Using **instanceOf** to compare
- **Simple within Classic**
  - Using JSPTag with **getParent**() method
  - Using **instanceOf** to compare
- **Classic within Simple**
  - **getAdaptee**() return the wrapper-up JSPTag
  - **getParent**() method return the adaptee's parent