CÔNG NGHỆ JAVA

JSP - Standard Tag Library (JSTL)

Nguyễn Hữu Thể

JavaServer Pages Standard Tag Library (JSTL)

- Thư viện thẻ chuẩn, cung cấp các thẻ để kiểm soát trang, lặp và các lệnh điều khiển, các thẻ quốc tế hóa, và các thẻ SQL.
- JSTL là một phần của Java EE API.
- Để sử dụng JSTL => cần phải tải về các thư viện JSTL, đặt các thư viện này vào thư mục WEB-INF/lib của project.
- Các nhóm thư viện thẻ JSTL:
  1. Core Tags: Nhóm thẻ cơ bản
  2. Formatting tags: Nhóm thẻ định dạng
  3. SQL tags: Nhóm thẻ SQL
  4. XML tags: Nhóm thẻ XML
  5. JSTL Functions: Nhóm hàm JSTL

Các thẻ cơ bản (Core Tags)

- Các thẻ cơ bản cung cấp hỗ trợ cho bộ lặp (iteration), các điều kiện logic, bắt ngoại lệ, url, chuyển tiếp (forward) hoặc chuyển hướng (redirect),...

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

Các thẻ định dạng (Formatting and Localization Tags)

- Những thẻ này cung cấp định dạng cho các con số, ngày tháng

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

Các thẻ SQL (SQL Tags)

- Các thẻ JSTL SQL cung cấp các hỗ trợ cho việc tương tác với cơ sở dữ liệu quan hệ như Oracle, MySql...

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

Các thẻ XML (XML Tags)

- Thẻ XML được sử dụng để làm việc với các tài liệu XML như phân tích cú pháp XML, chuyển đổi dữ liệu XML và XPath đánh giá biểu thức.

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

Các thẻ hàm JSTL (JSTL Functions Tags)

- Thẻ JSTL cung cấp một số chức năng mà chúng ta có thể sử dụng để thực hiện các toán tử dùng chung, xử lý chuỗi

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

Download thư viện JSTL

Thư viện JSTL (Cung cấp bởi Apache)	Thư viện JSTL (Cung cấp bởi Glassfish)
taglibs-standard-spec-*.jar	javax.servlet.jsp.jslt-api-*.jar
taglibs-standard-impl-*.jar	javax.servlet.jsp.jstl-*.jar

Các thẻ JSTL cơ bản (JSTL Core Tags)

Thẻ	Miêu tả
<c:out >	Giống <%= ... >, nhưng cho các Expression
<c:set >	Thiết lập kết quả của một ước lượng Expression trong một 'scope'
<c:remove >	Gỡ bỏ một biến mục tiêu (từ một biến scope cụ thể, nếu đã xác định)
<c:catch>	Bắt bất kỳ Throwable mà xuất hiện trong thân của nó và trưng bày nó một cách tùy ý
<c:if>	Thẻ điều kiện đơn giản, mà ước lượng phần thân của nó nếu điều kiện đã cho là true
<c:choose>	Thẻ điều kiện đơn giản mà thiết lập một context cho các hoạt động điều kiện loại trừ, được đánh dấu bởi <when> và <otherwise>
<c:when>	Thẻ phụ của <choose> mà include phần thân của nó nếu điều kiện được ước lượng là true

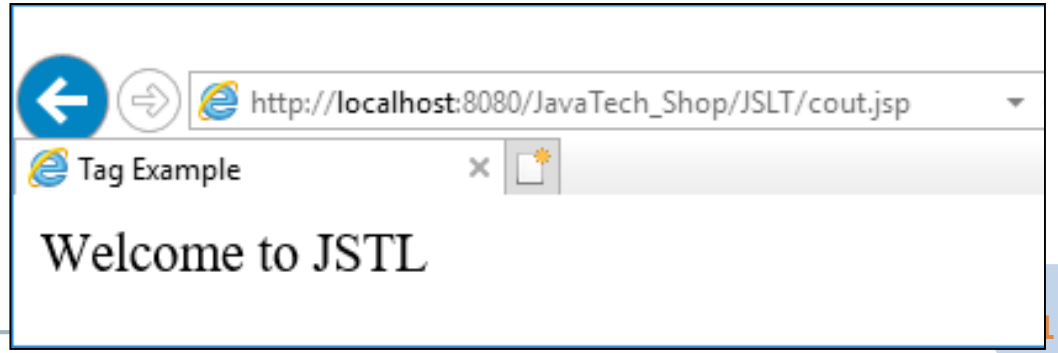
Các thẻ JSTL cơ bản (JSTL Core Tags)

Thẻ	Miêu tả
<c:otherwise >	Thẻ phụ của <choose> mà theo sau thẻ <when> và chỉ chạy nếu tất cả điều kiện trước được ước lượng là 'false'
<c:import>	Thu nhận một URL tuyệt đối hoặc quan hệ và trưng bày nội dung của nó tới hoặc trang đó, một String trong 'var', hoặc một Reader trong 'varReader'
<c:forEach >	Thẻ lặp cơ bản, chấp nhận nhiều kiểu tập hợp khác nhau và hỗ trợ subsetting (chia tập con) và tính năng khác
<c:forTokens>	Lặp qua các token, được phân biệt bởi các dấu phân tách (delimiter) đã cung cấp
<c:param>	Thêm một parameter tới một URL của thẻ đang chứa 'import'
<c:redirect >	Redirect tới một URL mới
<c:url>	Tạo một URL với các tham số truy vấn tùy ý

<c:out>

- The <c:out> tag is similar to JSP expression tag, but it can only be used with expression. It will display the result of an expression, similar to the way < %=...% > work.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
    <c:out value="${'Welcome to JSTL'}"/>
</body>
</html>
```



<c:set>

- It is used to set the result of an expression evaluated in a 'scope'.

Scope: có 4 phạm vi:

1. page
2. application
3. request
4. session

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<body>
    <c:set var="Income" scope="session" value="${4000*4}" />
    <c:out value="${Income}" />
</body>
</html>
```

Output:

16000

<c:set>

Attribute	Description	Required	Default
value	Information to save	No	body
target	Name of the variable whose property should be modified	No	None
property	Property to modify	No	None
var	Name of the variable to store information	No	None
scope	Scope of variable to store information	No	Page

<c:import>

- The <c:import> is similar to jsp 'include', with an additional feature of including the content of any resource either within server or outside the server.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
    <c:import var="data" url="http://www.javatpoint.com"/>
    <c:out value="${data}"/>
</body>
</html>
```

<c:if>

- The < c:if > tag is used for testing the condition and it display the body content, if the expression evaluated is true.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<body>
    <c:set var="income" scope="session" value="${4000*4}" />
    <c:if test="${income > 8000}">
        <p>My income is: <c:out value="${income}" /><p>
    </c:if>
</body>
</html>
```

- Output:
  - My income is: 16000

<c:if>

Attribute	Description	Required	Default
test	Condition to evaluate	Yes	None
var	Name of the variable to store the condition's result	No	None
scope	Scope of the variable to store the condition's result	No	page

<c:remove>

- It is used for removing the specified variable from a particular scope.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
    <c:set var="income" scope="session" value="${4000*4}"/>
    <p>Before Remove Value is: <c:out value="${income}"/></p>
    <c:remove var="income"/>
    <p>After Remove Value is: <c:out value="${income}"/></p>
</body>
</html>
```

Output:

Before Remove Value is: 16000

After Remove Value is:

<c:catch>

- It is used for Catches any Throwable exceptions that occurs in the body and optionally exposes it. In general it is used for error handling and to deal more easily with the problem occur in program.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
      <c:catch var ="catchtheException">
        <% int x = 2/0;%>
      </c:catch>

      <c:if test = "${catchtheException != null}">
        <p>The type of exception is : ${catchtheException} <br />
        There is an exception: ${catchtheException.message}</p>
      </c:if>
</body>
</html>
```

Output:

The type of exception is : java.lang.ArithmaticException: / by zero
There is an exception: / by zero

<c:choose>, <c:when>, <c:otherwise>

- The < c:choose > tag is a conditional tag that establish a context for mutually exclusive conditional operations. It works like a Java **switch** statement in which we choose between a numbers of alternatives.
- The <c:when > is subtag of <choose > that will include its body if the condition evaluated be 'true'.
- The < c:otherwise > is also subtag of < choose > it follows <when> tags and runs only if all the prior condition evaluated is 'false'.
- The c:when and c:otherwise works like if-else statement. But it must be placed inside c:choose tag.

<c:choose>, <c:when>, <c:otherwise>

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
    <c:set var="income" scope="session" value="${4000*4}"/>
    <p>Your income is : <c:out value="${income}"/></p>
    <c:choose>
      <c:when test="${income <= 1000}">
        Income is not good.
      </c:when>
      <c:when test="${income > 10000}">
         Income is very good.
      </c:when>
      <c:otherwise>
        Income is undetermined...
      </c:otherwise>
    </c:choose>
</body>
</html>
```

Output:

Your income is : 16000
Income is very good.

<c:choose>, <c:when>, <c:otherwise>

Even/Odd Example using c:when and c:otherwise

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
    <c:set value="10" var="num"></c:set>
    <c:choose>
        <c:when test="${num%2==0}">
            <c:out value="${num} is even number"></c:out>
        </c:when>
        <c:otherwise>
            <c:out value="${num} is odd number"></c:out>
        </c:otherwise>
    </c:choose>
 </body>
</html>
```

Output:
10 is even number

<c:forEach>

- The <c:for each > is an iteration tag used for repeating the nested body content for fixed number of times or over the collection.

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

<html>
<body>
    <c:forEach var="i" begin="1" end="3">
      Item <c:out value="${i}"/><p>
    </c:forEach>
</body>
</html>
```

Output:

Item 1
Item 2
Item 3

<c:forEach>

Attribute	Description	Required	Default
items	Information to loop over	No	None
begin	Element to start with (0 = first item, 1 = second item, ...)	No	0
end	Element to end with (0 = first item, 1 = second item, ...)	No	Last element
step	Process every step items	No	1
var	Name of the variable to expose the current item	No	None
varStatus	Name of the variable to expose the loop status	No	None

< c:forTokens >

- The < c:forTokens > tag iterates over tokens which is separated by the supplied delimeters. It is used for break a string into tokens and iterate through each of the tokens to generate output.

Attribute	Description	Required	Default
delims	Characters to use as delimiters	Yes	None

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %
<html>
<body>
    <c:forTokens items="Rahul-Nakul-Rajesh" delims="-"
    var="name">
      <c:out value="${name}"/><p>
    </c:forTokens>
</body>
</html>
```

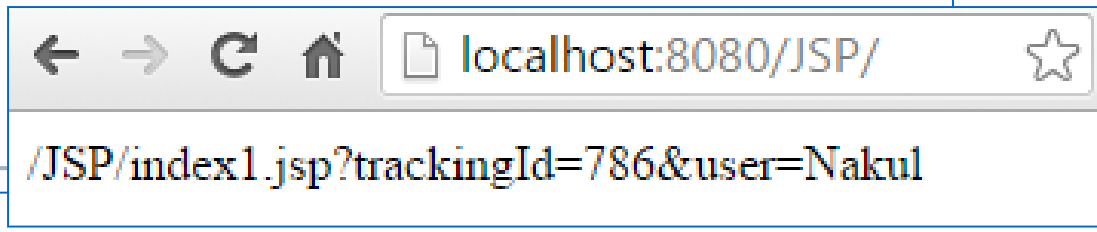
Output:

Rahul
Nakul
Rajesh

< c:param >

- The < c:param > tag add the parameter in a containing 'import' tag's URL. It allow the proper URL request parameter to be specified within URL and it automatically perform any necessary URL encoding.
- Inside < c:param > tag, the value attribute indicates the parameter value and name attribute indicates the parameter name.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
    <c:url value="/index1.jsp" var="completeURL"/>
        <c:param name="trackingId" value="786"/>
        <c:param name="user" value="Nakul"/>
    </c:url>
    ${completeURL}
</body>
</html>
```



< c:redirect >

- The < c:redirect > tag redirects the browser to a new URL. It supports the context-relative URLs, and the < c:param > tag.
- It is used for redirecting the browser to an alternate URL by using automatic URL rewriting.

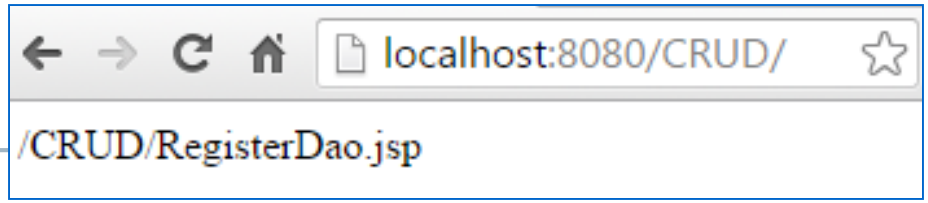
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
  <c:set var="url" value="0" scope="request"/>
  <c:if test="${url<1}">
     <c:redirect url="http://javatpoint.com"/>
  </c:if>
  <c:if test="${url>1}">
     <c:redirect url="http://facebook.com"/>
  </c:if>
</body>
</html>
```

Output:
http://javatpoint.com

< c:url >

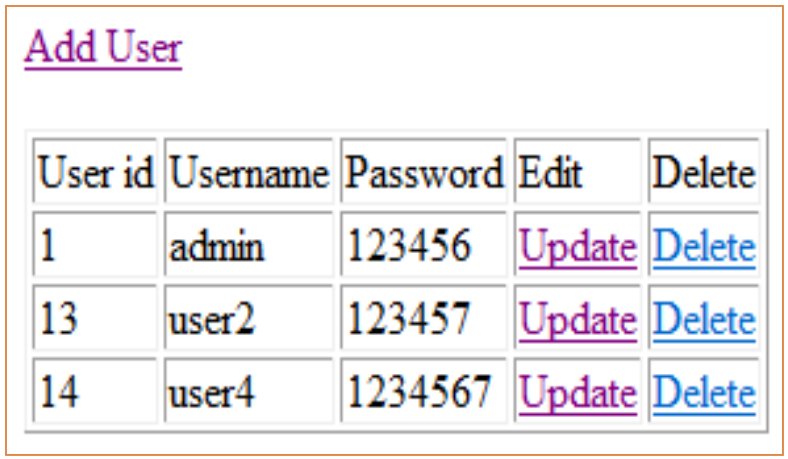
- The < c:url > tag creates a URL with optional query parameter. It is used for url encoding or url formatting. This tag automatically performs the URL rewriting operation.
- The JSTL url tag is used as an alternative method of writing call to the response.encodeURL() method. The advantage of url tag is proper URL encoding and including the parameters specified by children. **param** tag.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<body>
    <c:url value="/RegisterDao.jsp"/>
</body>
</html>
```



```
<body>
<p><a href="UserController?action=insert">Add User</a></p>
<table border=1>
<tr>
<td>User id</td><td>Username</td><td>Password</td><td>Edit</td><td>Delete</td>
</tr>
<%
UserDao ud = new UserDao();
ResultSet rs = ud.selectUser();
while(rs.next()){
%>
      <tr>
      <td><%= rs.getInt(1) %></td>
      <td><%= rs.getString(2) %></td>
      <td><%= rs.getString(3) %></td>
      <td><a href="UserController?action=edit&userid=<%=rs.getInt(1)%>">Edit</a></td>
      <td><a
      href="UserController?action=delete&userid=<%=rs.getInt(1)%>">Delete</a></td>
      </tr>
<%
}
%>
</table>
</body>
</html>
```

Code JSP



```
<%@ page language="java" co
<%@ taglib uri="http://java
<%@ taglib uri="http://java
<html>
<body>
<p><a href="UserController?action=insert">Add User</a></p>
<table border="1">
        <tr>
                <td>User id</td>
                <td>Username</td>
                <td>Password</td>
                <td>Edit</td>
                <td>Delete</td>
        </tr>
<c:forEach items="${users}" var="user">
        <tr>
                <td><c:out value="${user.userid}" /></td>
                <td><c:out value="${user.username}" /></td>
                <td><c:out value="${user.password}" /></td>
                <td><a
        href="UserController?action=edit&userid=<c:out
value="${user.userid}"/>">Update</a></td>
                <td><a href="UserController?action=delete&userid=<c:out
```

Code JSTL

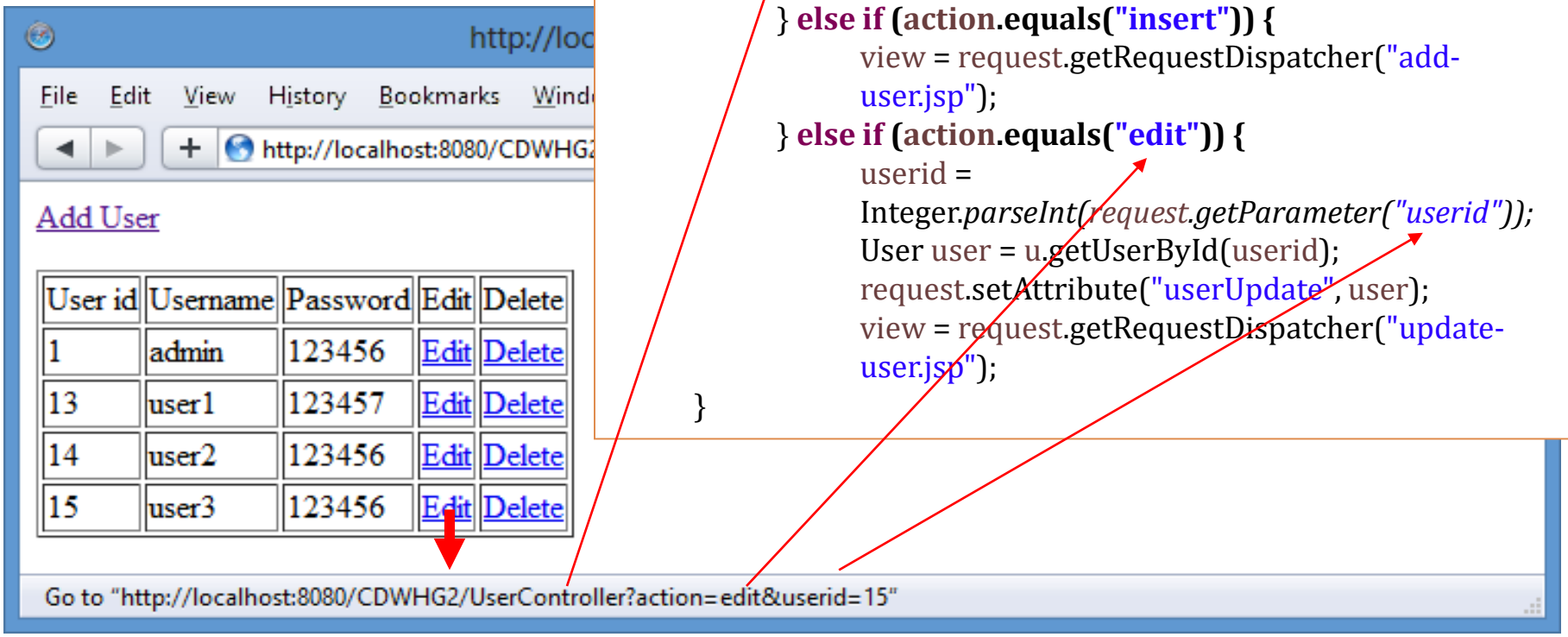
```
public class UserController extends HttpServlet {
protected void doGet(...){
        ...
        if(action.equals("all")){
                request.setAttribute("users", dao.getAllUsers());
                view = request.getRequestDispatcher("manage-user.jsp");
        }
```

Add User

User id	Username	Password	Edit	Delete
1	admin	123456	Update	Delete
13	user2	123457	Update	Delete
14	user4	1234567	Update	Delete

Sửa 1 dòng dữ l

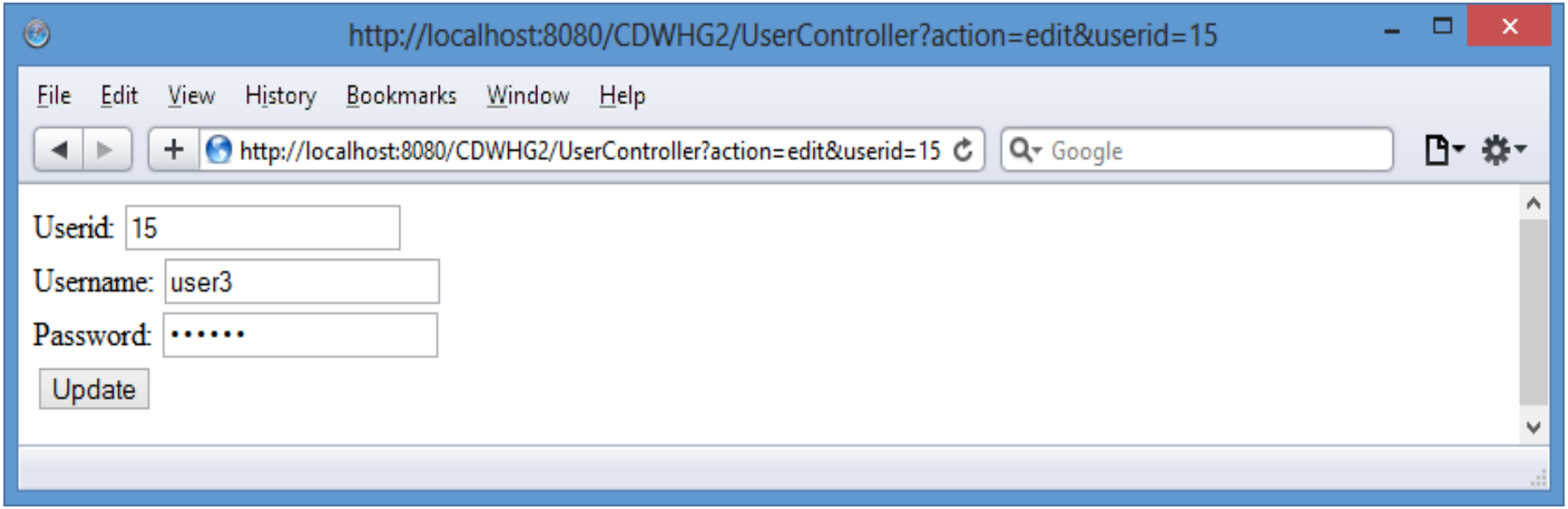
Sửa 1 dòng dữ liệu, rê ch




```
public class UserController extends HttpServlet {
    protected void doGet(...) {
        int userid = 0;
        RequestDispatcher view = null;
        UserDao u = new UserDao();
        if (action.equals("all")) {
            request.setAttribute("users", dao.getAllUsers());
            view = request.getRequestDispatcher("listUser.jsp");
        } else if (action.equals("insert")) {
            view = request.getRequestDispatcher("add-
            user.jsp");
        } else if (action.equals("edit")) {
            userid =
            Integer.parseInt(request.getParameter("userid"));
            User user = u.getUserById(userid);
            request.setAttribute("userUpdate", user);
            view = request.getRequestDispatcher("update-
            user.jsp");
    }
```

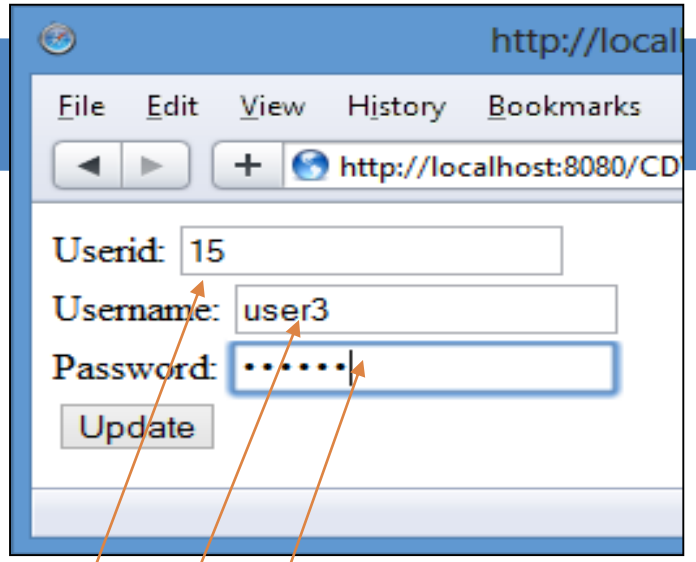
Sửa 1 dòng dữ liệu

Chuyển dữ liệu của dòng người dùng muốn sửa, trình bày dữ liệu ra giao diện cập nhật



Để có được giao diện này, chúng ta phải tạo 1 file **update-user.jsp** để trình bày dữ liệu cần cập nhật

update-user.jsp

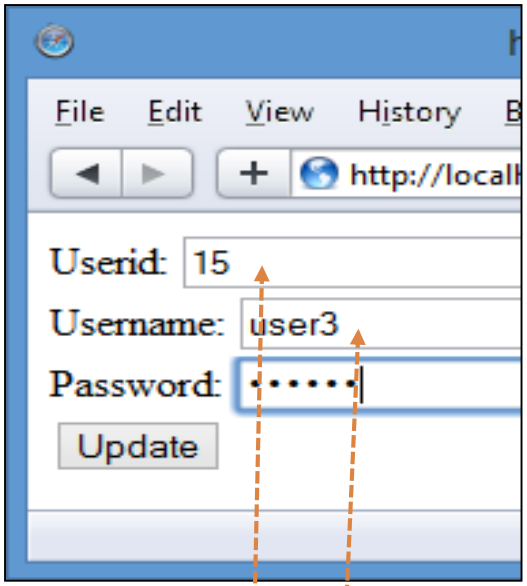



```
<%@ page import="org.dhcl.model.*" %>
<%@ page import="java.sql.*" %>
<html>
<body>
<%
//gọi  biến userUpdate đã lưu khi xử lý ở Controller
User userUpdate = (User)request.getAttribute("userUpdate");
%>
<form action="UserController" method="post">
        Userid: <input type="text" name="userid"
                readonly value="<%=userUpdate.getUserid() %>"><br>
        Username: <input type="text" name="username"
                value="<%=userUpdate.getUsername() %>"><br>
        Password: <input type="password" name="password"
                value="<%=userUpdate.getPassword() %>"><br>
        <input type="submit" value="Update">
</form>
</body></html>
```

```jsp
<%
//gọi  biến userUpdate đã lưu khi xử lý ở Controller
User userUpdate = (User)request.getAttribute("userUpdate");
%>

<form action="UserController" method="post">
        Userid: <input type="text" name="userid"
                readonly value="<%=userUpdate.getUserid() %>"><br>
        Username: <input type="text" name="username"
                value="<%=userUpdate.getUsername() %>"><br>
        Password: <input type="password" name="password"
                value="<%=userUpdate.getPassword() %>"><br>
        <input type="submit" value="Update">
</form>
```

Code JSP



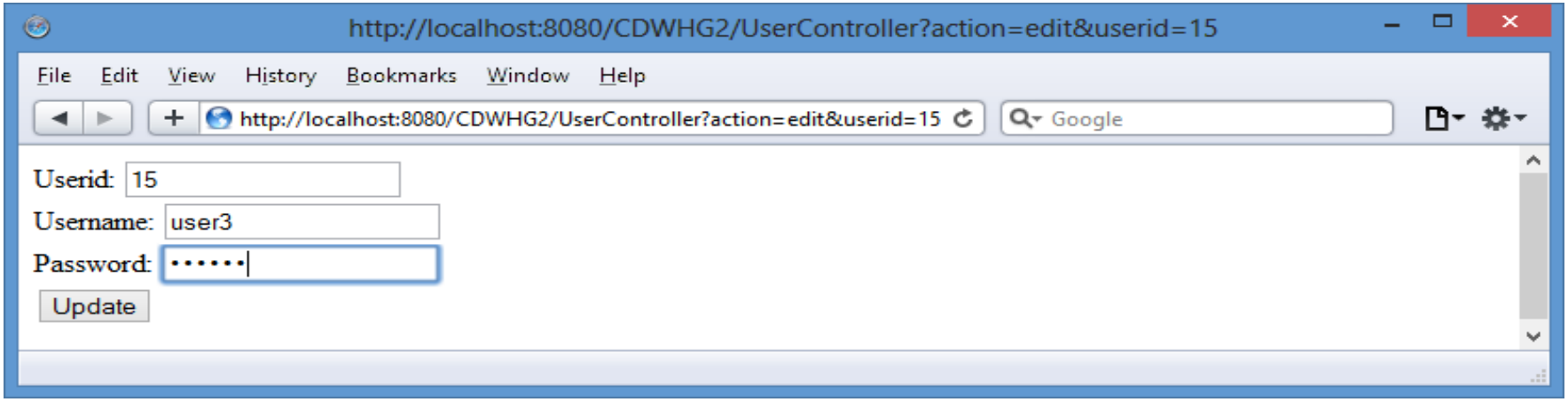

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"%>
<%@ page import="org.dhcl.model.*" %>
<%@ page import="java.sql.*" %>

<form action="UserController" method="post">
        Userid: <input type="text" name="userid" readonly value="${userUpdate.userid}"><br>
        Username: <input type="text" name="username" value="${userUpdate.username}"><br>
        Password: <input type="password" name="password"
value="${userUpdate.password}"><br>
        <input type="submit" value="Update">
</form>
```

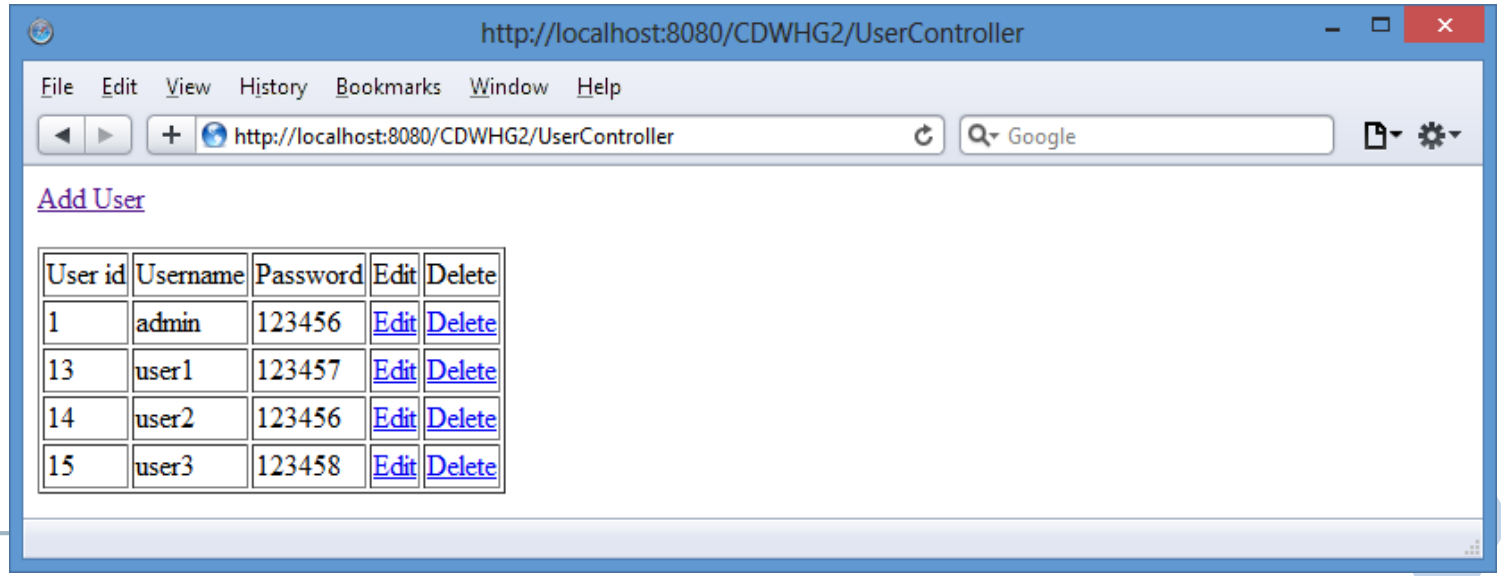
Code JSTL

update-user.jsp

Giả sử chúng ta sửa password lại thành 123458, và nhấn nút Update



Kết quả xử lý như sau:



JSTL Functions

Hàm	Miêu tả
fn:contains()	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho
fn:containsIgnoreCase()	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho trong trường hợp không phân biệt kiểu chữ
fn:endsWith()	Kiểm tra nếu một chuỗi input kết thúc với suffix đã cho
fn:escapeXml()	Các ký tự thoát mà có thể được phiên dịch như XML markup
fn:indexOf()	Trả về index bên trong một chuỗi về sự xuất hiện đầu tiên của chuỗi phụ
fn:join()	Kết hợp tất cả phần tử trong một mảng thành một chuỗi
fn:length()	Trả về số item trong một tập hợp, hoặc số ký tự trong một chuỗi
fn:replace()	Trả về một chuỗi là kết quả của việc thay thế một chuỗi input với một chuỗi đã cho

JSTL Functions

Hàm	Miêu tả
fn:split()	Chia một chuỗi thành một mảng các chuỗi phụ
fn:startsWith()	Kiểm tra nếu một chuỗi input bắt đầu với prefix đã cho
fn:substring()	Trả về một tập con của một chuỗi
fn:substringAfter()	Trả về một tập con của một chuỗi ở sau một chuỗi phụ đã cho
fn:substringBefore()	Trả về một tập con của một chuỗi ở trước một chuỗi phụ đã cho
fn:toLowerCase()	Biến đổi tất cả ký tự của một chuỗi thành chữ thường
fn:toUpperCase()	Biến đổi tất cả ký tự của một chuỗi thành chữ hoa
fn:trim()	Gỡ bỏ các khoảng trống trắng từ hai đầu của một chuỗi

fn:contains()

- Is used for testing if the string containing the specified substring. If the specified substring is found in the string, it returns true otherwise false.

```
boolean contains(java.lang.String, java.lang.String)
```

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<body>
<c:set var="theString" value="Welcome to JSTL"/>
<c:if test="${fn:contains(theString, 'abc')}">
   <p>Found abc string<p>
</c:if>

<c:if test="${fn:contains(theString, 'ABC')}">
   <p>Found ABC string<p>
</c:if>
</body>
</html>
```

Output:
Found abc string

fn:containsIgnoreCase()

- determines whether an input string contains a specified substring. While doing search it ignores the case.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<html>
<body>
<c:set var="theString" value="I am a test String" />

<c:if test="${fn:containsIgnoreCase(theString, 'test')}">
    <p>Found test string</p>
</c:if>

<c:if test="${fn:containsIgnoreCase(theString, 'TEST')}">
    <p>Found TEST string</p>
</c:if>
</body>
</html>
```

Found test string
Found TEST string

fn:endsWith()

- Determines if an input string ends with a specified suffix.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<html>
<body>
<c:set var="theString" value="I am a test String 123" />

<c:if test="${fn:endsWith(theString, '123')}">
    <p>String ends with 123</p>
</c:if>

<c:if test="${fn:endsWith(theString, 'TEST')}">
    <p>String ends with TEST</p>
</c:if>
</body>
</html>
```

String ends with 123

fn:escapeXml()

- escapes characters that can be interpreted as XML markup.

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<%@ taglib uri = "http://java.sun.com/jsp/jstl/functions" prefix = "fn" %>
<html>
  <body>
     <c:set var = "string1" value = "This is first String."/>
     <c:set var = "string2" value = "This <abc>is second String.</abc>"/>

     <p>With escapeXml() Function:</p>
     <p>string (1) : ${fn:escapeXml(string1)}</p>
     <p>string (2) : ${fn:escapeXml(string2)}</p>

     <p>Without escapeXml() Function:</p>
     <p>string (1) : ${string1}</p>
     <p>string (2) : ${string2}</p>
  </body>
</html>
```

With escapeXml() Function:
string (1) : This is first String.
string (2) : This <abc>is second String.</abc>
Without escapeXml() Function –
string (1) : This is first String.
string (2) : This is second String.

fn:indexOf()

- returns the index within a string of a specified substring.

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<%@ taglib uri = "http://java.sun.com/jsp/jstl/functions" prefix = "fn" %>
<html>
<body>
    <c:set var = "string1" value = "This is first String."/>
    <c:set var = "string2" value = "This <abc>is second String.</abc>"/>
     <p>Index (1) : ${fn:indexOf(string1, "first")}</p>
     <p>Index (2) : ${fn:indexOf(string2, "second")}</p>
  </body>
</html>
```

Index (1) : 8
Index (2) : 13

fn:join()

- concatenates all the elements of an array into a string with a specified separator.

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

<html>
  <body>
     <c:set var = "string1" value = "This is first String."/>
     <c:set var = "string2" value = "${fn:split(string1, ' ')}" />
     <c:set var = "string3" value = "${fn:join(string2, '-')}" />
     <p>Final String : ${string3}</p>
  </body>
</html>
```

Final String : This-is-first-String.

fn:length()

- returns the string length or the number of items in a collection.

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

<html>
  <body>
     <c:set var = "string1" value = "This is first String."/>
     <c:set var = "string2" value = "This is second String." />
     <p>Length of String (1) : ${fn:length(string1)}</p>
     <p>Length of String (2) : ${fn:length(string2)}</p>
  </body>
</html>
```

Length of String (1) : 21
Length of String (2) : 22

fn:replace()

- replaces all occurrences of a string with another string.

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

<html>
  <body>
    <c:set var = "string1" value = "This is first String."/>
    <c:set var = "string2" value = "${fn:replace(string1, 'first', 'second')}" />
    <p>Final String : ${string2}</p>
  </body>
</html>
```

Final String : This is second String.

fn:split()

- splits a string into an array of substrings based on a delimiter string.

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<%@ taglib uri = "http://java.sun.com/jsp/jstl/functions" prefix = "fn" %>
<html>
  <body>
    <c:set var = "string1" value = "This is first String."/>
    <c:set var = "string2" value = "${fn:split(string1, ' ')}" />
    <c:set var = "string3" value = "${fn:join(string2, '-')}" />

    <p>String (3) : ${string3}</p>

    <c:set var = "string4" value = "${fn:split(string3, '-')}" />
    <c:set var = "string5" value = "${fn:join(string4, ' ')}" />

    <p>String (5) : ${string5}</p>
  </body>
</html>
```

String (3) : This-is-first-String.
String (5) : This is first String.

fn:startsWith()

- determines if an input string starts with a specified substring.

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<%@ taglib uri = "http://java.sun.com/jsp/jstl/functions" prefix = "fn" %>
<html>
  <body>
    <c:set var = "string" value = "Second: This is first String."/>

    <c:if test = "${fn:startsWith(string, 'First')}">
      <p>String starts with First</p>
    </c:if>

    <br />
    <c:if test = "${fn:startsWith(string, 'Second')}">
      <p>String starts with Second</p>
    </c:if>
   </body>
</html>
```

String starts with Second

fn:substring()

- returns a subset of a string specified by start and end indices.

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

<html>
  <body>
    <c:set var = "string1" value = "This is first String."/>
    <c:set var = "string2" value = "${fn:substring(string1, 5, 15)}" />

    <p>Final sub string : ${string2}</p>
  </body>
</html>
```

Final sub string : is first S

fn:substringAfter()

- returns the part of a string after a specified substring.

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

<html>
  <body>
     <c:set var = "string1" value = "This is first String."/>
     <c:set var = "string2" value = "${fn:substringAfter(string1, 'is')}" />

     <p>Final sub string : ${string2}</p>
  </body>
</html>
```

Final sub string : is first String.

fn:substringBefore()

- returns the part of a string before a specified substring.

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

<html>
  <body>
     <c:set var = "string1" value = "This is first String."/>
     <c:set var = "string2" value = "${fn:substringBefore(string1, 'first')}" />
     <p>Final sub string : ${string2}</p>
  </body>
</html>
```

```
Final sub string : This is
```

fn:toLowerCase()

- converts all the characters of a string to lowercase.

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

<html>
  <body>
    <c:set var = "string1" value = "This is first String."/>
    <c:set var = "string2" value = "${fn:toLowerCase(string1)}" />

    <p>Final string : ${string2}</p>
  </body>
</html>
```

fn:toUpperCase()

- converts all the characters of a string to uppercase.

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

<html>
  <body>
     <c:set var = "string1" value = "This is first String."/>
     <c:set var = "string2" value = "${fn:toUpperCase(string1)}" />

     <p>Final string : ${string2}</p>
  </body>
</html>
```

Final string : THIS IS FIRST STRING.

fn:trim()

- removes white space from both ends of a string.

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

<html>
  <body>
     <c:set var = "string1" value = "This is first String        "/>
     <p>String (1) Length : ${fn:length(string1)}</p>

     <c:set var = "string2" value = "${fn:trim(string1)}" />
     <p>String (2) Length : ${fn:length(string2)}</p>
     <p>Final string : ${string2}</p>
  </body>
</html>
```

String (1) Length : 29
String (2) Length : 20
Final string : This is first String