JSTL Validations

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

<html>
<head>
<title>User Info Entry Form</title>
</head>

<body bgcolor="white">

<form action="validate_jstl.jsp" method="post">

<input type="hidden" name="submitted" value="true">
<table>

<c:if test="${param.submitted && empty param.userName}">
<tr><td></td>
<td colspan="2"><font color="red">
Please enter your Name
</font></td></tr>
</c:if>
<tr>
<td>Name:</td>
<td>
<input type="text" name="userName"
value="<c:out value="${param.userName}" />">
</td>
</tr>

<c:if test="${param.submitted && empty param.birthDate}">
<tr><td></td>
<td colspan="2"><font color="red">
Please enter your Birth Date
</font></td></tr>
</c:if>
<tr>
<td>Birth Date:</td>
<td>
<input type="text" name="birthDate"
value="<c:out value="${param.birthDate}" />">
</td>
<td>(Use format yyyy-mm-dd)</td>
```

```
</tr>
<c:if test="${param.submitted && empty param.emailAddr}">
<tr><td></td>
<td colspan="2"><font color="red">
Please enter your Email Address
</font></td></tr>
</c:if>
<tr>
<td>Email Address:</td>
<td>
<input type="text" name="emailAddr"
value="<c:out value="${param.emailAddr}" />">
</td>
<td>(Use format name@company.com)</td>
</tr>

<c:if test="${param.submitted && param.gender != 'm' && param.gender != 'f'}">
<tr><td></td>
<td colspan="2"><font color="red">
Please select a valid Gender
</font></td></tr>
</c:if>
<tr>
<td>Gender:</td>
<td>
<c:choose>
<c:when test="${param.gender == 'f'}">
<input type="radio" name="gender" value="m">
Male<br>
<input type="radio" name="gender" value="f" checked>
Female
</c:when>
<c:otherwise>
<input type="radio" name="gender" value="m" checked>
Male<br>
<input type="radio" name="gender" value="f">
Female
</c:otherwise>
</c:choose>
</td>
</tr>



<c:if test="${param.submitted && (param.luckyNumber < 1 || param.luckyNumber >
100)}">
<tr><td></td>
```

```
<td colspan="2"><font color="red">
Please enter a Lucky Number between 1 and 100
</font></td></tr>
</c:if>
<tr>
<td>Lucky number:</td>
<td>
<input type="text" name="luckyNumber"
value="<c:out value="${param.luckyNumber}" />">
</td>
<td>(A number between 1 and 100)</td>
</tr>
```

```
<c:forEach items="${paramValues.food}" var="current">
<c:choose>
<c:when test="${current == 'z'}">
<c:set var="pizzaSelected" value="true" />
</c:when>
<c:when test="${current == 'p'}">
<c:set var="pastaSelected" value="true" />
</c:when>
<c:when test="${current == 'c'}">
<c:set var="chineseSelected" value="true" />
</c:when>
<c:otherwise>
<c:set var="invalidSelection" value="true" />
</c:otherwise>
</c:choose>
</c:forEach>
<c:if test="${invalidSelection}">
<tr><td></td>
<td colspan="2"><font color="red">
Please select only valid Favorite Foods
</font></td></tr>
</c:if>
<tr>
<td>Favorite Foods:</td>
<td>
<input type="checkbox" name="food" value="z" ${pizzaSelected ? 'checked' : ''}>Pizza<br>
<input type="checkbox" name="food" value="p" ${pastaSelected ? 'checked' : ''}>Pasta<br>
<input type="checkbox" name="food" value="c" ${chineseSelected ? 'checked' : ''}>Chinese
</td>
</tr>
<tr>
<td colspan="3">
```

```
<input type="submit" value="Send Data">
</td>
</tr>
</table>
</form>
</body>
</html>
```

The first thing to notice in this example is the HTML field of type "hidden", named submitted with the value true. The browser doesn't display a hidden field, but its value is sent as a regular request parameter when the form is submitted. I use it in this example to avoid displaying error messages when the page is loaded for the first time, before the user has had a chance to enter any data. The submitted parameter isn't part of the first request for the page, but when the user submits the form, the submitted parameter is sent along with all parameters representing the other HTML fields. Hence, it can be used to test if the parameters should be validated or not.
The validation of the Name field illustrates how it works. The JSTL <c:if> action, is used with an EL expression that evaluates to true only if the submitted parameter has the value true and the userName parameter is empty. Since the submitted parameter isn't part of the initial request to load the page, it doesn't have the value true, causing the EL expression to evaluate to false. The <c:if> action's body is therefore ignored in this case.
After submitting the form, however, the submitted parameter has the value true, so if the username parameter contains an empty string (the user didn't enter a value in the Name field), the body is processed, adding the error message.

empty operator : It's included in the EL to avoid having to deal with the difference between a null value (the absence of a value) and the empty string value ("") because in a web application, you typically want to treat both cases the same way. In addition to empty strings and null, it also evaluates to true for an empty array, java.util.Collection, or java.util.Map. In other words, you can use it to test for empty collections of all types.

Another fairly unique feature in the EL is that you have a choice with regards to the symbols for the common operators. For instance, instead of using && as the logical AND operator, || for logical OR, and ! for logical NOT, you can use and, or, and not. The relational operators can be written as ==, !=, <, <=, >, and >=, or as eq, ne, lt, le, gt, and ge, respectively.

The Gender field isn't represented by a text field but by a radio button, so another approach is also needed for initializing it with the submitted value. To make a radio button be displayed as selected, the checked attribute must be added to the HTML element. The JSTL <c:choose> action helps us with this task. The <c:choose> action has no attributes; it just groups and controls any number of nested <c:when> actions and optionally one <c:otherwise> action. These are the only actions that are accepted as direct child elements of a <c:choose> element. A <c:choose> block is used to pick one of a set of related, mutually exclusive alternatives. The <c:choose> action makes sure that only the first <c:when> action with a test attribute value that evaluates to true is processed. If no <c:when> action meets its test condition, the <c:otherwise> body is processed instead.

The <c:choose> action contains one <c:when> action that tests if the gender parameter has the value f, and if so, adds both radio button fields with the one representing the f choice as selected. The <c:otherwise> action adds the radio button fields with the one representing m as selected.

The effect is that the m choice becomes the default, used if the submitted value is invalid. It may seem redundant to handle invalid values for a parameter representing a radio button, but it isn't. Even though using a group of radio buttons helps the regular user pick a valid value, you must guard against requests submitted through other means than the form. It's easy for someone to submit an HTTP request to your page with any value.