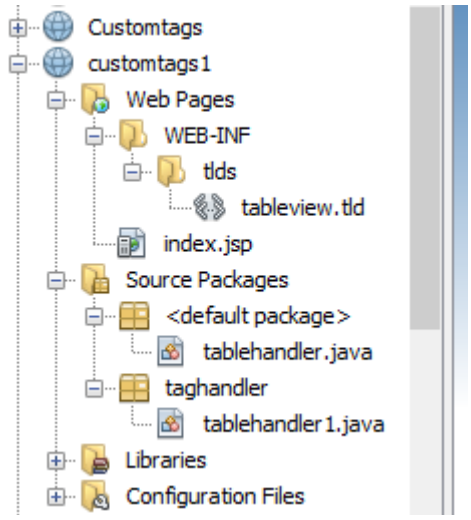
Cusotm tags in JSP



Step 1: create the Tag library descriptor as mentioned in tableview.tld
step 2: create the Tag handler as mentioned in tablehandler1.java
step 3: create the index.jsp

step 1

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.1" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-jsptaglibrary_2_1.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>t</short-name>
  <uri>/WEB-INF/tlds/tableview</uri>
  <!-- A validator verifies that the tags are used correctly at JSP
         translation time. Validator entries look like this:
      <validator>
          <validator-class>com.mycompany.TagLibValidator</validator-class>
          <init-param>
            <param-name>parameter</param-name>
            <param-value>value</param-value>
          </init-param>
      </validator>
    -->
  <!-- A tag library can register Servlet Context event listeners in
        case it needs to react to such events. Listener entries look
        like this:
     <listener>
        <listener-class>com.mycompany.TagLibListener</listener-class>
     </listener>
    -->
 <tag>
   <name>tablehandler</name>
```

```
    <tag-class>tablehandler</tag-class>
    <body-content>scriptless</body-content>
  </tag>
  <tag>
    <name>tablehandler1</name>
    <tag-class>taghandler.tablehandler1</tag-class>
    <body-content>JSP</body-content>
    <attribute>
      <name> rollno </name>
      <required> true</required>
    </attribute>
    <attribute>
      <name> value </name>
      <required> true</required>
    </attribute>
  </tag>
</taglib>
```

step2:

```
package taghandler;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.IOException;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.BodyContent;
import javax.servlet.jsp.tagext.BodyTagSupport;

/**
 *
 * @author admin
 */
public class tablehandler1 extends BodyTagSupport {

    /**
     * Creates new instance of tag handler
     */
    public tablehandler1() {
        super();
    }

    ////////////////////////////////////////////////////////////
    ///                                                      ///
    ///   User methods.                                      ///
```

```
///                                                         ///
///   Modify these methods to customize your tag handler.    ///
///                                                         ///
/////////////////////////////////////////////////////////////////
/**
 * Method called from doStartTag(). Fill in this method to perform other
 * operations from doStartTag().
 */
private void otherDoStartTagOperations() {
    // TODO: code that performs other operations in doStartTag
    //       should be placed here.
    //       It will be called after initializing variables,
    //       finding the parent, setting IDREFs, etc, and
    //       before calling theBodyShouldBeEvaluated().
    //
    //       For example, to print something out to the JSP, use the following:
    //
    //   try {
    //       JspWriter out = pageContext.getOut();
    //       out.println("something");
    //   } catch (IOException ex) {
    //       // do something
    //   }
}

/**
 * Method called from doEndTag() Fill in this method to perform other
 * operations from doEndTag().
 */
private void otherDoEndTagOperations() {
    // TODO: code that performs other operations in doEndTag
    //       should be placed here.
    //       It will be called after initializing variables,
    //       finding the parent, setting IDREFs, etc, and
    //       before calling shouldEvaluateRestOfPageAfterEndTag().
}

/**
 * Fill in this method to process the body content of the tag. You only need
 * to do this if the tag's BodyContent property is set to "JSP" or
 * "tagdependent." If the tag's bodyContent is set to "empty," then this
 * method will not be called.
 */
private String rollno;
/**
 * Fill in this method to process the body content of the tag. You only need
 * to do this if the tag's BodyContent property is set to "JSP" or
 * "tagdependent." If the tag's bodyContent is set to "empty," then this
 * method will not be called.
 */
```

```
    private String value;

    private void writeTagBodyContent(JspWriter out, BodyContent bodyContent) throws
IOException {
        // TODO: insert code to write html before writing the body content.
        // e.g.:
        //
        // out.println("<strong>" + attribute_1 + "</strong>");
        // out.println("   <blockquote>");

        // write the body content (after processing by the JSP engine) on the output Writer
        out.println("<table border="'+value+'">");
        out.println("<tr><tr>\n" +
"    <td>AAA</td>\n" +
"    <td>BBB</td>\n" +
"    <td>1000</td>\n" +
"  </tr>");
        bodyContent.writeOut(out);
        out.println("</table>");

        // Or else get the body content as a string and process it, e.g.:
        //     String bodyStr = bodyContent.getString();
        //     String result = yourProcessingMethod(bodyStr);
        //     out.println(result);
        // TODO: insert code to write html after writing the body content.
        // e.g.:
        //
        // out.println("   </blockquote>");
        // clear the body content for the next time through.
        bodyContent.clearBody();
    }

    ////////////////////////////////////////////////////////////////
    ///                                                          ///
    ///   Tag Handler interface methods.                         ///
    ///                                                          ///
    ///   Do not modify these methods; instead, modify the       ///
    ///   methods that they call.                                ///
    ///                                                          ///
    ////////////////////////////////////////////////////////////////
    /**
     * This method is called when the JSP engine encounters the start tag, after
     * the attributes are processed. Scripting variables (if any) have their
     * values set here.
     *
     * @return EVAL_BODY_BUFFERED if the JSP engine should evaluate the tag
     * body, otherwise return SKIP_BODY. This method is automatically generated.
     * Do not modify this method. Instead, modify the methods that this method
     * calls.
     */
```

```
@Override
public int doStartTag() throws JspException {
   otherDoStartTagOperations();

   if (theBodyShouldBeEvaluated()) {
      return EVAL_BODY_BUFFERED;
   } else {
      return SKIP_BODY;
   }
}

/**
 * This method is called after the JSP engine finished processing the tag.
 *
 * @return EVAL_PAGE if the JSP engine should continue evaluating the JSP
 * page, otherwise return SKIP_PAGE. This method is automatically generated.
 * Do not modify this method. Instead, modify the methods that this method
 * calls.
 */
@Override
public int doEndTag() throws JspException {
   otherDoEndTagOperations();

   if (shouldEvaluateRestOfPageAfterEndTag()) {
      return EVAL_PAGE;
   } else {
      return SKIP_PAGE;
   }
}

/**
 * This method is called after the JSP engine processes the body content of
 * the tag.
 *
 * @return EVAL_BODY_AGAIN if the JSP engine should evaluate the tag body
 * again, otherwise return SKIP_BODY. This method is automatically
 * generated. Do not modify this method. Instead, modify the methods that
 * this method calls.
 */
@Override
public int doAfterBody() throws JspException {
   try {
      // This code is generated for tags whose bodyContent is "JSP"
      BodyContent bodyCont = getBodyContent();
      JspWriter out = bodyCont.getEnclosingWriter();

      writeTagBodyContent(out, bodyCont);
   } catch (Exception ex) {
      handleBodyContentException(ex);
   }
```

```
    if (theBodyShouldBeEvaluatedAgain()) {
        return EVAL_BODY_AGAIN;
    } else {
        return SKIP_BODY;
    }
}

/**
 * Handles exception from processing the body content.
 */
private void handleBodyContentException(Exception ex) throws JspException {
    // Since the doAfterBody method is guarded, place exception handing code here.
    throw new JspException("Error in tablehandler1 tag", ex);
}

/**
 * Fill in this method to determine if the rest of the JSP page should be
 * generated after this tag is finished. Called from doEndTag().
 */
private boolean shouldEvaluateRestOfPageAfterEndTag() {
    // TODO: code that determines whether the rest of the page
    //       should be evaluated after the tag is processed
    //       should be placed here.
    //       Called from the doEndTag() method.
    //
    return true;
}

/**
 * Fill in this method to determine if the tag body should be evaluated
 * again after evaluating the body. Use this method to create an iterating
 * tag. Called from doAfterBody().
 */
private boolean theBodyShouldBeEvaluatedAgain() {
    // TODO: code that determines whether the tag body should be
    //       evaluated again after processing the tag
    //       should be placed here.
    //       You can use this method to create iterating tags.
    //       Called from the doAfterBody() method.
    //
    return false;
}

private boolean theBodyShouldBeEvaluated() {
    // TODO: code that determines whether the body should be
    //       evaluated should be placed here.
    //       Called from the doStartTag() method.
    return true;
}
```

```
    /**
     * @return the rollno
     */
    public String getRollno() {
        return rollno;
    }

    /**
     * @param rollno the rollno to set
     */
    public void setRollno(String rollno) {
        this.rollno = rollno;
    }

    /**
     * @return the value
     */
    public String getValue() {
        return value;
    }

    /**
     * @param value the value to set
     */
    public void setValue(String value) {
        this.value = value;
    }

}
```

step3:

```
<%--
    Document   : index
    Created on : 5 Sep, 2019, 11:42:25 PM
    Author     : admin
--%>
<%@taglib uri="/WEB-INF/tlds/tableview" prefix="t" %>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <t:tablehandler1 rollno="1001"  value="3">First cell</t:tablehandler1>
```

```
    </body>
</html>
```

output