Práctica 6
Macedo Borbolla Eduardo Sai
JSP Custom Tag Libs
Web Application Development

Resumen

Reporte del desarrollo de CustomTagLib
19-1

Teoría: Las Taglibs nos permiten el desarrollo de etiquetas con las cuales solo llamaremos el código de funcionalidad Java mediante una estructura ya definida en un TLD y el espacio de nombres XML para así respetar el uso del patrón MVC en donde en la vista no debe reflejarse ninguna otra capa.

Desarrollo:

Primero desarrollamos la funcionalidad en un archivo .java.

Wad.java.

Primera parte

```
package mx.ipn.escom.wad.tarea6.taglibs;
import java.io.IOException;
public class Wad extends SimpleTagSupport{

    private String value;
    @Override
    public void doTag() throws JspException, IOException{
        super.doTag();
        Properties properties= new Properties();
        JspWriter out=  getJspContext().getOut();//Obtener contexto de la aplicacion
        if(this.value != null && !this.value.equals("")){
          StringBuilder sb= new StringBuilder();//Construir cadena con el valor de la taglib.
          sb.append("value=\"" +this.value + "\" ");
          String partes[]=value.split("\\."); //Dividir la cadena para obtener el archivo
          String clase=partes[0];    //Obtener el nombre de la clase a cargar
          final String PROPERTIES_FILE_NAME = clase+".properties";//Obtener el nombre del archivo a cargar.
```

Primero obtenemos el contexto de la aplicacción para obtener la ubicación del archivo y el valor de la cadena de donde obtenderemos el archivo de propiedades.

Para el archivo de propiedades instanciamos uno de la clase Properties.

Verificamos que la cadena dada no sea nula ni vacía de ser así la añadimos a un StringBuilder y dividimos con el método .split para buscar crear una constante con el archivo de tipo “clase.properties”

```
        try { //Método para obtener las propiedades del archivo
              input = Thread.currentThread().getContextClassLoader().getResourceAsStream(PROPERTIES_FILE_NAME);
              properties.load(input);
              out.print(properties.getProperty(value));
          } catch (IOException ex) {

          } finally {
              if (input != null) {
                  try {
                      input.close();

                  } catch (IOException e) {
                      //
                  }
              }
          }
    }
}
```

Creamos un flujo de datos de tipo InputStream y mediante un hilo cargamos la clase. Finalmente imprimimos en el JSP nuestro atributo y cerramos el flujo de datos.

Después (o antes) de crear la funcionalidad se debe definir el formato de la etiqueta con su TLD para ser llamado tanto al JSP como por el espacio de nombres XML.

Wad.tld

```
<?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>
    <jspversion>2.2</jspversion>
    <short-name>wad</short-name>
    <uri>/WEB-INF/tags</uri>
    <tag>
        <name>text</name>
        <tag-class>mx.ipn.escom.wad.tarea6.taglibs.Wad</tag-class>
        <body-content>empty</body-content>
        <attribute>
            <name>value</name>
            <required>true</required>
            <rtexprvalue>true</rtexprvalue>
        </attribute>
    </tag>
</taglib>
```

La etiqueta se llama text, contiene un campo body vació el cual no se evalua, solo tiene un atributo llamado “value” el cual es requerido y regresa una expresión que puede ser generada en tiempo de ejecución.

Dentro del JSP

Primero definimos el espacio de nombres para llamar al JCTL

```
<?xml version="1.0" encoding="UTF-8" ?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
    xmlns:wad="/WEB-INF/tags"  version="2.0">
    <jsp:directive.page language="java"
        contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" />
    <jsp:text>
        <![CDATA[ <?xml version="1.0" encoding="UTF-8" ?> ]]>
    </jsp:text>
    <jsp:text>
        <![CDATA[ <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> ]]>
    </jsp:text>
    <html xmlns="http://www.w3.org/1999/xhtml">
```

Finalmente llamamos a nuestro JSP el archivo.

```
<body>
    <h1>Ingresar</h1>
    <form action="login/LoginCtrl" method="post">
        Usuario: <input type="text" name="username" /><br /> Contraseña: <input
            type="password" name="password" /><br /> <input type="submit" value="Ingresar"/>
    <h1><wad:text value="registrarPersona.titulo"/></h1>
    <h1><wad:text value="registrarPersona.nombre" /></h1>
    </form>
</bodv>
```

Prueba:

Conclusiones:

Si bien es bueno saber usar las taglibs que alguien ya desarrollo, puede que sea necesario, cómodo, seguro o de alguna manera mejor en el desarrollo de alguna aplicación que el desarrollador decida usar TagLibs a la medida, las cuales pueden ser eficientes o no.