**JSP Custom Tags Quiz**

1. Rewrite the JSP page in the first demo (forEach) to use scripting instead of the JSTL forEach.

String c= request.getParameter("color");

BeerExpert be = new BeerExpert();

List result = be.getBrands(c);

Iterator it = result.iterator();

while(it.hasNext()) {

out.println("<br>try: " +it.next());

}

//JSTL

<%

String c= request.getParameter("color");

BeerExpert be = **new** BeerExpert();

List result = be.getBrands(c);

request.setAttribute("beers", result);

%>

Beer Selection Advice <br>

<c:forEach var=*"brand"* items=*"*${beers }*"*>

<br>try: ${ brand}

</c:forEach>

// another

<%

List styles = (List)request.getAttribute(“styles”);

Iterator it = styles.iterator();

while(it.hasNext()) {

out.print(“<br>try: “ + it.next());

}

%>

1. How do custom tags relate to JSTL?

key steps

◦ define a tag including attributes and body

◦ write a Tag Library Descriptor (TLD) that the container will read

◦ write a tag handler class that implements the tag functionality

◦ use the tag on a JSP page and link it to the tag descriptor

1. What is the role of the URI in the TLD and the taglib directive?

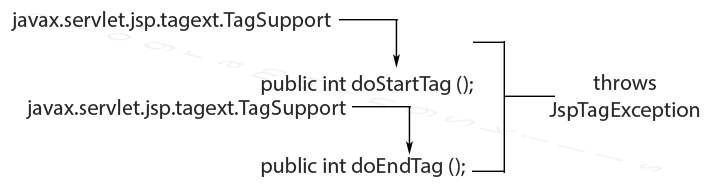
The uri attribute value resolves to a location the container understands and the prefix attribute informs a container what bits of markup are custom actions.

1. What is a tag handler class?

Extends the tag support framework class

– javax.servlet.jsp.tagext.TagSupport

A **tag handler class** is basically a JavaBeans **class**, it should contain set of set methods and set of get methods. All the attributes of custom **tag** must be used as properties or data members in **tag handler class** in the same order in which they appear



1. What is the role of attribute setters in a tag handler class?

Tag handler class needs to implement setter methods to accept an attribute value like a java bean. So we create the setter methods of the “from” and “to” attributes. To access the body of a tag in the tag handler class, we call the getJspBody() method and call the invoke() method on it with a writer as a parameter. In this way, the text in the body of the tag is written to the given writer parameter. As you notice, we call sw.toString() to get the body text.

package com.javacodegeeks.examples.jspcustomtagexample.tag;

import java.io.IOException;

import java.io.StringWriter;

import javax.servlet.jsp.JspException;

import javax.servlet.jsp.tagext.SimpleTagSupport;

public class SendMessageTag extends SimpleTagSupport {

private String from;

private String to;

public void setFrom( final String from ) {

this.from = from;

}

public void setTo( final String to ) {

this.to = to;

}

@Override

public void doTag() throws JspException, IOException {

final StringWriter sw = new StringWriter();

getJspBody().invoke( sw );

getJspContext().getOut().println( "There is a message from '" +

from + "' to '" + to + "'. Message content is '" +

sw.toString() + "'" );

}

}

The JSP engine invokes the **setter** methods for each **attribute** appropriately after the **tag handler** is initialized and before the doStartTag() method is called. Generally, you should implement the **setter** methods to store the **attribute** value in a member variable that is accessible to the other methods of the **tag handler**.

1. What is the role of the doTag() method in a tag handler class?

The **doTag() method** is invoked. The implementation of **doTag()** performs its **function**, potentially calling other **tag handlers** (if the **tag handler** is implemented as a **tag** file) and invoking fragments. The **doTag() method** returns, and the **tag handler** instance is discarded.

1. What does the operation getJspContext().getOut().write(“Hi Bob”) do when called in a doTag() method?

getJspContext().getOut().write("Hi Bob"); // write Hi Bob to output

It prints "Hi bob" to the screen.

1. What does the operation getJspBody().invoke(null) do when called in a doTag() method?

Following some website introduce about .invoke(null) will present the content of the body tag. But It made a result error on my computer.