Adv. Java means DURGA SIR..

ADV.JAVA

With

SCWCD / OCWCD

Question Bank

Chapter : 9,10. Building JSP Pages Using Tag Libraries and Custom Tag Library

DURGA M.Tech

(Sun certified & Realtime Expert)

Ex. IBM Employee

Trained Lakhs of Students
for last 14 years across INDIA

India's No.1 Software Training Institute

DURGASOFT

www.durgasoft.com Ph: 9246212143 ,8096969696

Plot No : 202, IInd Floor ,HUDA Maitrivanam,Ameerpet, Hyd-5000038.
www.durgasoft.com

9&10. Building JSP Pages Using Tag Libraries and Custom Tag Library

- For a custom tag library or a library of Tag Files, create the 'taglib' directive for a JSP page.
- Given a design goal, create the custom tag structure in a JSP page to support that goal.
- Given a design goal, use an appropriate JSP Standard Tag Library (JSTL v1.1) tag from the "core" tag library.
- Describe the semantics of the "Classic" custom tag event model when each event method (doStartTag, doAfterBody, and doEndTag) is executed, and explain what the return value for each event method means; and write a tag handler class.
- Using the PageContext API, write tag handler code to access the JSP implicit variables and access web application attributes.
- Given a scenario, write tag handler code to access the parent tag and an arbitrary tag ancestor.
- Describe the semantics of the "Simple" custom tag event model when the event method (doTag) is executed; write a tag handler class; and explain the constraints on the JSP content within the tag.
- Describe the semantics of the Tag File model; describe the web application structure for tag files; write a tag file; and explain the constraints on the JSP content in the body of the tag.

Q1. You have built a collection of custom tags for your web application. The TLD file is located in the file: /WEB-INF/myTags.xml. You refer to these tags in your JSPs using the symbolic name: myTags. Which deployment descriptor element must you use to make this link between the symbolic name and the TLD file name?

A. <taglib>
<name>myTags</name>
<location>/WEB-INF/myTags.xml</location>
</taglib>

B. <tags>
<name>myTags</name>
<location>/WEB-INF/myTags.xml</location>
</tags>

C. <tags>
<tags-uri>myTags</taglib-uri>
<tags-location>/WEB-INF/myTags.xml</tags-location>
</tags>

D. <taglib>
<taglib-uri>myTags</taglib-uri>
<taglib-location>/WEB-INF/myTags.xml</taglib-location>
</taglib>

Answer: D

Q2. Click the Exhibit button.
The attribute "name" has a value of "Foo,"
What is the result if this tag handler's tag is invoked?

```
 5. public class MyTagHandler extends
TagSupport {
 6.   public int doStartTag() throws
JspException {
 7.     try {
 8.       Writer out =
pageContext.getResponse().getWriter();
 9.       String name =
pageContext.findAttribute("name");
10.      out.print(name);
11.    } catch(Exception ex) { /* handle
exception */ }
12.    return SKIP_BODY;
13.  }
14
15.  public int doAfterBody() throws
JspException {
16.    try {
17.      Writer out =
pageContext.getResponse().getWriter();
18.      out.print("done");
19.    } catch(Exception ex) { /* handle
exception */ }
20.    return EVAL_PAGE;
21.  }
...
42. }
```

A. Foo
B. done
C. Foodone
D. An exception is thrown at runtime.
E. No output is produced from this code.
F. Compilation fails because of an error in this code.
Answer: A

Q3. You are building a web application that will be used throughout the European Union; therefore, it has significant internationalization requirements. You have been tasked to create a custom tag that generates a message using the java.text.MessageFormat class. The tag will take the resourceKey attribute and a variable number of argument attributes with the format, arg<N>. Here is an example use of this tag and its output:
<t:message resourceKey='diskFileMsg' arg0='MyDisk' arg1='1247' />
generates:The disk "MyDisk" contains 1247 file(s).
Which Simple tag class definition accomplishes this goal of handling a variable number of tag attributes?

A. public class MessageTag extends SimpleTagSupport
implements VariableAttributes {
private Map attributes = new HashMap();
public void setVariableAttribute(String uri,
String name, Object value) {
this.attributes.put(name, value);
}
// more tag handler methods
}
B. The Simple tag model does NOT support a variable number of attributes.
C. public class MessageTag extends SimpleTagSupport
implements DynamicAttributes {
private Map attributes = new HashMap();
public void putAttribute(String name, Object value) {
this.attributes.put(name, value);
}
// more tag handler methods
}
D. public class MessageTag extends SimpleTagSupport
implements VariableAttributes {
private Map attributes = new HashMap();
public void putAttribute(String name, Object value) {
this.attributes.put(name, value);
}
// more tag handler methods
}
E. public class MessageTag extends SimpleTagSupport
implements DynamicAttributes {
private Map attributes = new HashMap();
public void setDynamicAttribute(String uri, String name,
Object value) {
this.attributes.put(name, value);
}
// more tag handler methods
}

Q4. Given the JSP code:
<% request.setAttribute("foo", "bar"); %>
and the Classic tag handler code:
5. public int doStartTag() throws JspException {
6. // insert code here
7. // return int
8. }
Assume there are no other "foo" attributes in the web application.
Which invocation on the pageContext object, inserted at line 6, assigns "bar" to the variable x?
A. String x = (String) pageContext.getAttribute("foo");
B. String x = (String) pageContext.getRequestScope("foo");
C. It is NOT possible to access the pageContext object from within doStartTag.
D. String x = (String)
pageContext.getRequest().getAttribute("foo");
E. String x = (String) pageContext.getAttribute("foo",
PageContext.ANY_SCOPE);
Answer: D

Q5. Which two statements about tag files are true? (Choose two.)

A. Classic tag handlers and tag files CANNOT reside in the same tag library.
B. A file named foo.tag, located in /WEB-INF/tags/bar, is recognized as a tag file by the container.
C. A file named foo.tag, bundled in a JAR file but NOT defined in a TLD, triggers a container translation error.
D. A file named foo.tag, located in a web application's root directory, is recognized as a tag file by the container.
E. If files foo1.tag and foo2.tag both reside in /WEB-INF/tags/bar, the container will consider them part of the same tag library.

Answer: B, E

Q6. The sl:shoppingList and sl:item tags output a shopping list to the response and are used as follows:
11. <sl:shoppingList>
12. <sl:item name="Bread" />
13. <sl:item name="Milk" />
14. <sl:item name="Eggs" />
15. </sl:shoppingList>

The tag handler for sl:shoppingList is ShoppingListTag and the tag handler for sl:item is ItemSimpleTag.
ShoppingListTag extends BodyTagSupport and ItemSimpleTag extends SimpleTagSupport.
Which is true?

A. ItemSimpleTag can find the enclosing instance of ShoppingListTag by calling getParent() and casting the
result to ShoppingListTag.
B. ShoppingListTag can find the child instances of ItemSimpleTag by calling super.getChildren() and casting
each to an ItemSimpleTag.
C. It is impossible for ItemSimpleTag and ShoppingListTag to find each other in a tag hierarchy because one is
a Simple tag and the other is a Classic tag.
D. ShoppingListTag can find the child instances of ItemSimpleTag by calling getChildren() on the PageContext
and casting each to an ItemSimpleTag.
E. ItemSimpleTag can find the enclosing instance of ShoppingListTag by calling findAncestorWithClass() on
the PageContext and casting the result to ShoppingListTag.

Answer: A

Q7. Assume that a news tag library contains the tags lookup and item:
lookup Retrieves the latest news headlines and executes the tag body once for each headline.
Exposes a NESTED page-scoped attribute called headline of type com.example.Headline containing details for that headline.
item Outputs the HTML for a single news headline. Accepts an attribute info of type com.example.Headline containing details for the headline to be rendered.Which snippet of JSP code returns the latest news headlines in an HTML table, one per row?
A. <table>
<tr>
<td>

```
<news:lookup />
<news:item info="${headline}" />
</td>
</tr>
</table>
```

B.
```
<news:lookup />
<table>
<tr>
<td><news:item info="${headline}" /></td>
</tr>
</table>
```

C.
```
<table>
<news:lookup>
<tr>
<td><news:item info="${headline}" /></td>
</tr>
</news:lookup>
</table>
```

D.
```
<table>
<tr>
<news:lookup>
<td><news:item info="${headline}" /></td>
</news:lookup>
</tr>
</table>
```

Answer: C

Q8. Which JSTL code snippet can be used to perform URL rewriting?

A. <a href='<c:url url="foo.jsp"/>' />
B. <a href='<c:link url="foo.jsp"/>' />
C. <a href='<c:url value="foo.jsp"/>' />
D. <a href='<c:link value="foo.jsp"/>' />

Answer: C

Q9. Assume the scoped attribute priority does NOT yet exist. Which two create and set a new request-scoped attribute priority to the value "medium"? (Choose two.)

A. ${priority = 'medium'}
B. ${requestScope['priority'] = 'medium'}
C. <c:set var="priority" value="medium" />
D. <c:set var="priority" scope="request">medium</c:set>
E. <c:set var="priority" value="medium" scope="request" />
F. <c:set property="priority" scope="request">medium</c:set>
G. <c:set property="priority" value="medium" scope="request" />

Answer: D, E

Q10. You are creating a JSP page to display a collection of data. This data can be displayed in several different ways so the architect on your project decided to create a generic servlet that generates a comma-delimited string so that various pages can render the data in different ways. This servlet takes on request parameter: objectID. Assume that this servlet is mapped to the URL pattern:/WEB-INF/data.

In the JSP you are creating, you need to split this string into its elements separated by commas and generate an HTML <ul> list from the data.

Which JSTL code snippet will accomplish this goal?

A. <c:import varReader='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>

<c:forTokens items'${dataString.split(",")}' var='item'>
${item}
</c:forTokens>

B. <c:import varReader='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>

```
<ul>
<c:forTokens items'${dataString}' delims=',' var='item'>
<li>${item}</li>
</c:forTokens>
</ul>
```

C.
```
<c:import var='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>
<ul>
<c:forTokens items'${dataString.split(",")}' var='item'>
<li>${item}</li>
</c:forTokens>
</ul>
```

D.
```
<c:import var='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>
<ul>
<c:forTokens items'${dataString}' delims=',' var='item'>
<li>${item}</li>
</c:forTokens>
</ul>
```

Answer: D

Q11. Which three are true about TLD files? (Choose three.)

A. The web container recognizes TLD files placed in any subdirectory of WEB-INF.
B. When deployed inside a JAR file, TLD files must be in the META-INF directory, or a subdirectory of it.
C. A tag handler's attribute must be included in the TLD file only if the attribute can accept request-time expressions.
D. The web container can generate an implicit TLD file for a tag library comprised of both simple tag handlers and tag files.
E. The web container can automatically extend the tag library map described in a web.xml file by including entries extracted from the web application's TLD files.

Answer: A, B, E

Q12. Your management has required that all JSPs be created to generate XHTML-compliant content and to facilitate that decision, you are required to create all JSPs using the JSP Document format. In the reviewOrder.jspx page, you need to use several core JSTL tags to process the collection of order items in the customer's shopping cart. Which JSP code snippets must you use in the reviewOrder.jspx page?

A. <html xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0">
<jsp:directive.taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" />
<!-- page content -->
</html>
B. <html xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0"
xmlns:c="http://java.sun.com/jsp/jstl/core">
<!-- page content -->
</html>
C. <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0">
<jsp:directive.taglib prefix="c"
uri="http://java.sun.com/jsp/jstl/core" />
<!-- page content -->
</jsp:root>
D. <jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0"
xmlns:c="http://java.sun.com/jsp/jstl/core">
<!-- page content -->
</jsp:root>
Answer: D

Q13. Which two JSTL URL-related tags perform URL rewriting? (Choose two.)

A. url
B. link
C. param
D. import
E. redirect

Answer: A, E

Q14. A custom JSP tag must be able to support an arbitrary number of attributes whose names are unknown when the tag class is designed. Which two are true? (Choose two.)

A. The <body-content> element in the echo tag TLD must have the value JSP.
B. The echo tag handler must define the setAttribute(String key, String value) method.
C. The <dynamic-attributes>true</dynamic-attributes> element must appear in the echo tag TLD.
D. The class implementing the echo tag handler must implement the javax.servlet.jsp.tagext.IterationTag interface.
E. The class implementing the echo tag handler must implement the javax.servlet.jsp.tagext.DynamicAttributes interface.

Answer: C, E

Q15. After a merger with another small business, your company has inherited a legacy WAR file but the original source files were lost. After reading the documentation of that web application, you discover that the WAR file contains a useful tag library that you want to reuse in your own webapp packaged as a WAR file.
What do you need to do to reuse this tag library?

A. Simply rename the legacy WAR file as a JAR file and place it in your webapp's library directory.
B. Unpack the legacy WAR file, move the TLD file to the META-INF directory, repackage the whole thing as a JAR file, and place that JAR file in your webapp's library directory.

C. Unpack the legacy WAR file, move the TLD file to the META-INF directory, move the class files to the top-level directory, repackage the whole thing as a JAR file, and place that JAR file in your webapp's library directory.
D. Unpack the legacy WAR file, move the TLD file to the META-INF directory, move the class files to the top-level directory, repackage the WAR, and place that WAR file in your webapp's WEB-INF directory.

Answer: C

Q:16. You want to create a valid directory structure for your Java EE web application, and your application uses tag files and a JAR file. Which three must be located directly in your WEB-INF directory (NOT in a subdirectory of WEB-INF)? (Choose three.)

A. The JAR file
B. A directory called lib
C. A directory called tags
D. A directory called TLDs
E. A directory called classes
F. A directory called META-INF

Answer: B, C, E

Q17. Assume the custom tag my:errorProne always throws a java.lang.Runtime Exception with the message "File not found."
An error page has been configured for this JSP page.
Which option prevents the exception thrown by my:errorProne from invoking the error page mechanism, and outputs the message "File not found" in the response?

A. <c:try catch="ex">
<my:errorProne />
</c:try>
${ex.message}
B. <c:catch var="ex">
<my:errorProne />
</c:catch>
${ex.message}
C. <c:try>
<my:errorProne />
</c:try>
<c:catch var="ex" />
${ex.message}
D. <c:try>
<my:errorProne />
<c:catch var="ex" />
${ex.message}

</c:try>
E. <my:errorProne>
<c:catch var="ex">
${ex.message}
</c:catch>
</my:errorProne>
Answer: B

Q18. A JSP page contains a taglib directive whose uri attribute has the value dbtags. Which XML element within the web application deployment descriptor defines the associated TLD?

A. <tld>
<uri>dbtags</uri>
<location>/WEB-INF/tlds/dbtags.tld</location>
</tld>
B. <taglib>
<uri>dbtags</uri>
<location>/WEB-INF/tlds/dbtags.tld</location>
</taglib>
C. <tld>
<tld-uri>dbtags</tld-uri>
<tld-location>/WEB-INF/tlds/dbtags.tld</tld-location>
</tld>
D. <taglib>
<taglib-uri>dbtags</taglib-uri>
<taglib-location>
/WEB-INF/tlds/dbtags.tld
</taglib-location>
</taglib>

Answer: D

Q19. Click the Exhibit button.

Assuming the tag library in the exhibit is imported with the prefix stock, which custom tag invocation outputs the contents of the variable exposed by the quote tag?

```
 1. <?xml version="1.0" encoding="UTF-8" ?>
 2.
 3. <taglib
xmlns="http://java.sun.com/xml/ns/j2ee"
 4.    xmlns:xsi="http://www.w3.org/2001/XMLSchem
a-instance"
 5.    xsi:schemaLocation="http://java.sun.com/xm
l/ns/j2ee web-jsptaglibrary_2_0.xsd"
 6.    version="2.0">
 7.    <tlib-version>1.0</tlib-version>
 8.    <short-name>stock</short-name>
 9.    <uri>http://example.com/tld/stock</uri>
10.    <tag>
11.      <name>quote</name>
12.
<tag-class>com.example.QuoteTag</tag-class>
13.      <body-content>empty</body-content>
14.      <variable>
15.        <name-from-attribute>var</name-from-att
ute>
16.        <scope>AT_BEGIN</scope>
17.      </variable>
18.      <attribute>
19.        <name>symbol</name>
20.        <required>true</required>
21.        <rtexprvalue>true</rtexprvalue>
22.      </attribute>
23.      <attribute>
24.        <name>var</name>
25.        <required>true</required>
26.        <rtexprvalue>false</rtexprvalue>
27.      </attribute>
28.    </tag>
29. </taglib>
```

A. <stock:quote symbol="SUNW" />
${var}
B. ${var}
<stock:quote symbol="SUNW" />
C. <stock:quote symbol="SUNW">
${var}
</stock:quote>
D. <stock:quote symbol="SUNW" var="quote" />
${quote}

E. <stock:quote symbol="SUNW" var="quote">
<%= quote %>
</stock:quote>

Answer: D

Q20. You are creating a JSP page to display a collection of data. This data can be displayed in several different ways so the architect on your project decided to create a generic servlet that generates a comma-delimited string so that various pages can render the data in different ways. This servlet takes on request parameter: objectID. Assume that this servlet is mapped to the URL pattern: /WEB-INF/data.

In the JSP you are creating, you need to split this string into its elements separated by commas and generate an HTML <ul> list from the data.
Which JSTL code snippet will accomplish this goal?

A. <c:import varReader='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>

<c:forTokens items'${dataString.split(",")}' var='item'>
${item}
</c:forTokens>

B. <c:import varReader='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>

<c:forTokens items'${dataString}' delims=',' var='item'>
${item}
</c:forTokens>

C. <c:import var='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>

<c:forTokens items'${dataString.split(",")}' var='item'>
${item}
</c:forTokens>

D. <c:import var='dataString' url='/WEB-INF/data'>
<c:param name='objectID' value='${currentOID}' />
</c:import>

<c:forTokens items'${dataString}' delims=',' var='item'>
${item}
</c:forTokens>

Answer: D

Q21. A web application contains a tag file called beta.tag in /WEB-INF/tags/alpha. A JSP page called sort.jsp exists in the web application and contains only this JSP code:
1. <%@ taglib prefix="x"
2. tagdir="/WEB-INF/tags/alpha" %>
3. <x:beta />
The sort.jsp page is requested.
Which two are true? (Choose two.)

A. Tag files can only be accessed using a tagdir attribute.
B. The sort.jsp page translates successfully and invokes the tag defined by beta.tag.
C. The sort.jsp page produces a translation error because a taglib directive must always have a uri attribute.
D. Tag files can only be placed in /WEB-INF/tags, and NOT in any subdirectories of /WEB-INF/tags.
E. The tagdir attribute in line 2 can be replaced by a uri attribute if a TLD referring to beta.tag is created and added to the web application.
F. The sort.jsp page produces a translation error because the tagdir attribute on lines 1-2 specifies a directory other than /WEB-INF/tags, which is illegal.

Answer: B, E

Q22. Given a JSP page:

11. <n:recurse>
12. <n:recurse>
13. <n:recurse>
14. <n:recurse />
15. </n:recurse>

16. </n:recurse>
17. </n:recurse>

The tag handler for n:recurse extends SimpleTagSupport.
Assuming an n:recurse tag can either contain an empty body or another n:recurse tag, which strategy allows the tag handler for n:recurse to output the nesting depth of the deepest n:recurse tag?

A. It is impossible to determine the deepest nesting depth because it is impossible for tag handlers that extend SimpleTagSupport to communicate with their parent and child tags.
B. Create a private non-static attribute in the tag handler class called count of type int initialized to 0. Increment count in the doTag method. If the tag has a body, invoke the fragment for that body. Otherwise, output the value of count.
C. Start a counter at 1. Call getChildTags(). If it returns null, output the value of the counter. Otherwise, increment counter and continue from where getChildTags() is called. Skip processing of the body.
D. If the tag has a body, invoke the fragment for that body.Otherwise, start a counter at 1. Call getParent(). If it returns null, output the value of the counter Otherwise, increment the counter and continue from where getParent() is called.

Answer: D

Q23. Click the Exhibit button.
The h:highlight tag renders its body, highlighting an arbitrary number of words, each of which is passed as an attribute (word1, word2, ...). For example, a JSP page can invoke the h:highlight tag as follows:
11. <h:highlight color="yellow" word1="high" word2="low">
12. high medium low
13. </h:highlight>
Given that HighlightTag extends SimpleTagSupport, which three steps are necessary to implement the tag handler for the highlight tag? (Choose three).

```
 1. <?xml version="1.0" encoding="UTF-8" ?>
 2.
 3. <taglib
xmlns="http://java.sun.com/xml/ns/j2ee"
 4.    xmlns:xsi="http://www.w3.org/2001/XMLSchem
a-instance"
 5.    xsi:schemaLocation="http://java.sun.com/xm
l/ns/j2ee web-jsptaglibrary_2_0.xsd"
 6.    version="2.0">
 7.    <tlib-version>1.0</tlib-version>
 8.    <short-name>h</short-name>
 9.
<uri>http://example.com/tld/highlight</uri>
10.    <tag>
11.      <name>highlight</name>
12.      <tag-class>com.example.HighlightTag</tag-
lass>
13.
<body-content>scriptless</body-content>
14.      <attribute>
15.        <name>color</name>
16.        <required>true</required>
17.      </attribute>
18.
<dynamic-attributes>true</dynamic-attributes>
19.    </tag>
20. </taglib>
```

A. add a doTag method
B. add a doStartTag method
C. add a getter and setter for the color attribute
D. create and implement a TagExtraInfo class
E. implement the DynamicAttributes interface
F. add a getter and setter for the word1 and word2 attributes
Answer: A, C, E

Q24. Given:

5. public class MyTagHandler extends TagSupport {
6. public int doStartTag() throws JspException {
7. try {
8. // insert code here
9. } catch(Exception ex) { /* handle exception */ }
10. return super.doStartTag();
11. }
...
42. }

Which code snippet, inserted at line 8, causes the value foo to be output?

A. JspWriter w = pageContext.getOut();
w.print("foo");

B. JspWriter w = pageContext.getWriter();
w.print("foo");
C. JspWriter w = new JspWriter(pageContext.getWriter());
w.print("foo");
D. JspWriter w = new JspWriter(pageContext.getResponse());
w.print("foo");

Answer: A

Q25. Given:

6. <myTag:foo bar='42'>
7. <%="processing" %>
8. </myTag:foo>
and a custom tag handler for foo which extends TagSupport.

Which two are true about the tag handler referenced by foo? (Choose two.)

A. The doStartTag method is called once.
B. The doAfterBody method is NOT called.
C. The EVAL_PAGE constant is a valid return value for the doEndTag method.
D. The SKIP_PAGE constant is a valid return value for the doStartTag method.
E. The EVAL_BODY_BUFFERED constant is a valid return value for the doStartTag method.

Answer: A, C

Q26. Which three are valid values for the body-content attribute of a tag directive in a tag file? (Choose three.)
A. EL
B. JSP
C. empty
D. dynamic
E. scriptless
F. tagdependent

Answer: C, E, F

Q27. You have a new IT manager that has mandated that all JSPs must be refactored to include no scritplet code. The IT manager has asked you to enforce this. Which deployment descriptor element will satisfy this constraint?

A. <jsp-property-group>
<url-pattern>*.jsp</url-pattern>

<permit-scripting>false</permit-scripting>
</jsp-property-group>
B. <jsp-config>
<url-pattern>*.jsp</url-pattern>
<permit-scripting>false</permit-scripting>
</jsp-config>
C. <jsp-config>
<url-pattern>*.jsp</url-pattern>
<scripting-invalid>true</scripting-invalid>
</jsp-config>
D. <jsp-property-group>
<url-pattern>*.jsp</url-pattern>
<scripting-invalid>true</scripting-invalid>
</jsp-property-group>

Answer: D

Q28. Assume the tag handler for a st:simple tag extends SimpleTagSupport. In what way can scriptlet code be used in the body of st:simple?

A. set the body content type to JSP in the TLD
B. Scriptlet code is NOT legal in the body of st:simple.
C. add scripting-enabled="true" to the start tag for the st:simple element
D. add a pass-through Classic tag with a body content type of JSP to the body of st:simple, and place the
scriptlet code in the body of that tag
Answer: B

Q29. Which statement is true if the doStartTag method returns EVAL_BODY_BUFFERED ?

A. The tag handler must implement BodyTag.

B. The doAfterBody method is NOT called.
C. The setBodyContent method is called once.
D. It is never legal to return EVAL_BODY_BUFFERED from doStartTag.

Answer: C

Q30. You are creating a library of custom tags that mimic the HTML form tags. When the user submits a form that fails validation, the JSP form is forwarded back to the user. The <t:textField> tag must support the ability to re-populate the form field with the request parameters from the user's last request. For example, if the user entered "Samantha" in the text field called firstName, then the form is re-populated like this:
<input type='text' name='firstName' value='Samantha' />

Which tag handler method will accomplish this goal?

A. public int doStartTag() throws JspException {
JspContext ctx = getJspContext();
String value = ctx.getParameter(this.name);
if (value == null) value = "";
JspWriter out = pageContext.getOut();
try {
out.write(String.format(INPUT, this.name, value));
} (Exception e) { throw new JspException(e); }
return SKIP_BODY;
}
private static String INPUT
= "<input type='text' name='%s' value='%s' />";
B. public void doTag() throws JspException {
JspContext ctx = getJspContext();
String value = ctx.getParameter(this.name);
if (value == null) value = "";
JspWriter out = pageContext.getOut();
try {
out.write(String.format(INPUT, this.name, value));
} (Exception e) { throw new JspException(e); }
}
private static String INPUT
= "<input type='text' name='%s' value='%s' />";
C. public int doStartTag() throws JspException {
ServletRequet request = pageContext.getRequest();
String value = request.getParameter(this.name);
if (value == null) value = "";
JspWriter out = pageContext.getOut();
try {

```
out.write(String.format(INPUT, this.name, value));
} (Exception e) { throw new JspException(e); }
return SKIP_BODY;
}
private static String INPUT
= "<input type='text' name='%s' value='%s' />";
D. public void doTag() throws JspException {
ServletRequet request = pageContext.getRequest();
String value = request.getParameter(this.name);
if ( value == null ) value = "";
JspWriter out = pageContext.getOut();
try {
out.write(String.format(INPUT, this.name, value));
} (Exception e) { throw new JspException(e); }
}
private static String INPUT
= "<input type='text' name='%s' value='%s' />";
```

Answer: C

Q31. Which two directives are applicable only to tag files? (Choose two.)

A. tag
B. page
C. taglib
D. include
E. variable
Answer: A, E

Q32. The tl:taskList and tl:task tags output a set of tasks to the response and are used as follows:

11. <tl:taskList>

12. <tl:task name="Mow the lawn" />
13. <tl:task name="Feed the dog" />
14. <tl:task name="Do the laundry" />
15. </tl:taskList>
The tl:task tag supplies information about a single task while the tl:taskList tag does the final output. The tag handler for tl:taskList is TaskListTag. The tag handler for tl:task is TaskTag. Both tag handlers
extend BodyTagSupport.
Which allows the tl:taskList tag to get the task names from its nested tl:task children?

A. It is impossible for a tag handler that extends BodyTagSupport to communicate with its parent and child
tags.
B. In the TaskListTag.doStartTag method, call super.getChildTags() and iterate through the results. Cast each
result to a TaskTag and call getName().
C. In the TaskListTag.doStartTag method, call getChildTags() on the PageContext and iterate through the
results. Cast each result to a TaskTag and call getName().
D. Create an addTaskName method in TaskListTag. Have the TaskListTag.doStartTag method, return
BodyTag.EVAL_BODY_BUFFERED. In the TaskTag.doStartTag method, call super.getParent(), cast it to a
TaskListTag, and call addTaskName().
E. Create an addTaskName method in TaskListTag. Have the TaskListTag.doStartTag method, return
BodyTag.EVAL_BODY_BUFFERED. In the TaskTag.doStartTag method, call findAncestorWithClass() on the
PageContext, passing TaskListTag as the class to find. Cast the result to TaskListTag and call addTaskName().

Answer: D

Q33. You are developing several tag libraries that will be sold for development of third-party web applications. You are about to publish the first three libraries as JAR files:container-tags.jar, advanced-html-form-tags.jar, and basic-html-form-tags.jar. Which two techniques are appropriate for packaging the TLD files for these tag libraries? (Choose two.)

A. The TLD must be located within the WEB-INF directory of the JAR file.
B. The TLD must be located within the META-INF directory of the JAR file.
C. The TLD must be located within the META-INF/tld/ directory of the JAR file.
D. The TLD must be located within a subdirectory of WEB-INF directory of the JAR file.
E. The TLD must be located within a subdirectory of META-INF directory of the JAR file.
F. The TLD must be located within a subdirectory of META-INF/tld/ directory of the JAR file.
Answer: B,

Q34. A custom tag is defined to take three attributes. Which two correctly invoke the tag within a JSP page? (Choose two.)

A. <prefix:myTag a="foo" b="bar" c="baz" />
B. <prefix:myTag attributes={"foo","bar","baz"} />
C. <prefix:myTag jsp:attribute a="foo" b="bar" c="baz" />
D. <prefix:myTag>
<jsp:attribute a:foo b:bar c:baz />
</prefix:myTag>
E. <prefix:myTag>
<jsp:attribute ${"foo", "bar", "baz"} />
</prefix:myTag>
F. <prefix:myTag>
<jsp:attribute a="foo" b="bar" c="baz"/>
</prefix:myTag>
G. <prefix:myTag>
<jsp:attribute name="a">foo</jsp:attribute>
<jsp:attribute name="b">bar</jsp:attribute>
<jsp:attribute name="c">baz</jsp:attribute>
</prefix:myTag>

Answer: A, G

Q35. You have been contracted to create a web site for a free dating service. One feature is the ability for one client to send a message to another client, which is displayed in the latter client's private page. Your contract explicitly states that security is a high priority. Therefore, you need to prevent cross-site hacking in

which one user inserts JavaScript code that is then rendered and invoked when another user views that content. Which two JSTL code snippets will prevent cross-site hacking in the scenario above? (Choose two.)

A. <c:out>${message}</c:out>
B. <c:out value='${message}' />
C. <c:out value='${message}' escapeXml='true' />
D. <c:out eliminateXml='true'>${message}</c:out>
E. <c:out value='${message}' eliminateXml='true' />

Answer: B, C

Q36. Click the Exhibit button.
Assuming the tag library in the exhibit is imported with the prefix forum, which custom tag invocation produces a translation error in a JSP page?

```
 1. <?xml version="1.0" encoding="UTF-8" ?>
 2.
 3. <taglib
xmlns="http://java.sun.com/xml/ns/j2ee"
 4.    xmlns:xsi="http://www.w3.org/2001/XMLSchem
a-instance"
 5.    xsi:schemaLocation="http://java.sun.com/xm
l/ns/j2ee web-jsptaglibrary_2_0.xsd"
 6.    version="2.0">
 7.    <tlib-version>1.0</tlib-version>
 8.    <short-name>forum</short-name>
 9.    <uri>http://example.com/tld/forum</uri>
10.    <tag>
11.      <name>message</name>
12.
<tag-class>com.example.MessageTag</tag-class>
13.
<body-content>scriptless</body-content>
14.      <attribute>
15.        <name>from</name>
16.        <rtexprvalue>true</rtexprvalue>
17.      </attribute>
18.      <attribute>
19.        <name>subject</name>
20.        <required>false</required>
21.        <rtexprvalue>true</rtexprvalue>
22.      </attribute>
23.    </tag>
24. </taglib>
```

A. <forum:message from="My Name" subject="My Subject" />
B. <forum:message subject="My Subject">
My message body.
</forum:message>
C. <forum:message from="My Name" subject="${param.subject}">
${param.body}
</forum:message>
D. <forum:message from="My Name" subject="My Subject">
<%= request.getParameter("body") %>
</forum:message>

E. <forum:message from="My Name"
subject="<%= request.getParameter("subject") %>">
My message body.
</forum:message>
Answer: D

Q37. Which JSTL code snippet can be used to import content from another web resource?

A. <c:import url="foo.jsp"/>
B. <c:import page="foo.jsp"/>
C. <c:include url="foo.jsp"/>
D. <c:include page="foo.jsp"/>
E. Importing cannot be done in JSTL. A standard action must be used instead.

Answer: A

Q38. Click the Exhibit button.
Assume the tag library in the exhibit is placed in a web application in the path /WEB-INF/tld/example.tld.
1.
2. <ex:hello />
Which JSP code, inserted at line 1, completes the JSP code to invoke the hello tag?

A. <%@ taglib prefix="ex" uri="/WEB-INF/tld" %>
B. <%@ taglib uri="/WEB-INF/tld/example.tld" %>
C. <%@ taglib prefix="ex"
uri="http://localhost:8080/tld/example.tld" %>
D. <%@ taglib prefix="ex"
uri="http://example.com/tld/example" %>

Answer: D

Q39. Which JSTL code snippet produces the output "big number" when X is greater than 42, but outputs "small number" in all other cases?

A. <c:if test='<%= (X > 42) %>'>
<c:then>big number</c:then>
<c:else>small number</c:else>
</c:if>
B. <c:if>
<c:then test='<%= (X > 42) %>'>big number</c:then>
<c:else>small number</c:else>

</c:if>
C. <c:choose test='<%= (X > 42) %>'>
<c:then>big number</c:when>
<c:else>small number</c:otherwise>
</c:choose>
D. <c:choose test='<%= (X > 42) %>'>
<c:when>big number</c:when>
<c:otherwise>small number</c:otherwise>
</c:choose>
E. <c:choose>
<c:when test='<%= (X > 42) %>'>big number</c:when>
<c:otherwise>small number</c:otherwise>
</c:choose>

Answer: E

Q40. After a merger with another small business, your company has inherited a legacy WAR file but the original source files were lost. After reading the documentation of that web application, you discover that the WAR file contains a useful tag library that you want to reuse in your own webapp packaged as a WAR file.

What do you need to do to reuse this tag library?

A. Simply rename the legacy WAR file as a JAR file and place it in your webapp's library directory.
B. Unpack the legacy WAR file, move the TLD file to the META-INF directory, repackage the whole thing as
a JAR file, and place that JAR file in your webapp's library directory.
C. Unpack the legacy WAR file, move the TLD file to the META-INF directory, move the class files to the

top-level directory, repackage the whole thing as a JAR file, and place that JAR file in your webapp's library
directory.
D. Unpack the legacy WAR file, move the TLD file to the META-INF directory, move the class files to the
top-level directory, repackage the WAR, and place that WAR file in your webapp's WEB-INF directory.

Answer: C

Q41. Given:

3. public class MyTagHandler extends TagSupport {
4. public int doStartTag() {
5. // insert code here
6. // return an int
7. }
8. // more code here
...

18. }
There is a single attribute foo in the session scope.
Which three code fragments, inserted independently at line 5, return the value of the attribute? (Choose three.)

A. Object o = pageContext.getAttribute("foo");
B. Object o = pageContext.findAttribute("foo");
C. Object o = pageContext.getAttribute("foo",
PageContext.SESSION_SCOPE);
D. HttpSession s = pageContext.getSession();
Object o = s.getAttribute("foo");
E. HttpServletRequest r = pageContext.getRequest();
Object o = r.getAttribute("foo");

Answer: B, C, D

Q42. You are creating a content management system (CMS) with a web

application front-end. The JSP that displays a given document in the CMS has the following general
structure:
1. <%-- tag declaration --%>
2. <t:document>
...
11. <t:paragraph>... <t:citation docID='xyz' /> ...</t:paragraph>
...
99. </t:document>
The citation tag must store information in the document tag for the document tag to generate a reference section at the end of the generated web page.
The document tag handler follows the Classic tag model and the citation tag handler follows the Simple
tag model. Furthermore, the citation tag could also be embedded in other custom tags that could have
either the Classic or Simple tag handler model.
Which tag handler method allows the citation tag to access the document tag?

A. public void doTag() {
JspTag docTag = findAncestorWithClass(this, DocumentTag.class);
((DocumentTag)docTag).addCitation(this.docID);
}
B. public void doStartTag() {
JspTag docTag = findAncestorWithClass(this, DocumentTag.class);
((DocumentTag)docTag).addCitation(this.docID);
}
C. public void doTag() {
Tag docTag = findAncestor(this, DocumentTag.class);
((DocumentTag)docTag).addCitation(this.docID);

}
D. public void doStartTag() {
Tag docTag = findAncestor(this, DocumentTag.class);
((DocumentTag)docTag).addCitation(this.docID);
}
Answer: A

Q43. Given:
6. <myTag:foo bar='42'>
7. <%="processing" %>
8. </myTag:foo>
and a custom tag handler for foo which extends TagSupport.

Which two are true about the tag handler referenced by foo? (Choose two.)

A. The doStartTag method is called once.
B. The doAfterBody method is NOT called.
C. The EVAL_PAGE constant is a valid return value for the doEndTag method.
D. The SKIP_PAGE constant is a valid return value for the doStartTag method.
E. The EVAL_BODY_BUFFERED constant is a valid return value for the doStartTag method.

Answer: A, C

Q44. Which two are true concerning the objects available to developers creating tag files? (Choose two.)

A. The session object must be declared explicitly.
B. The request and response objects are available implicitly.
C. The output stream is available through the implicit outStream object.
D. The servlet context is available through the implicit servletContext object.
E. The JspContext for the tag file is available through the implicit jspContext object.

Answer: B, E

Q45. You web application uses a lot of Java enumerated types in the domain model of the application. Built into each enum type is a method, getDisplay(), which

returns a localized, user-oriented string. There are many uses for presenting enums within the web application, so your manager has asked you to create a custom tag that iterates over the set of enum values and processes the body of the tag once for each value; setting the value into a page-scoped attribute called, enumValue.
Here is an example of how this tag is used:
10. <select name='season'>
11. <t:everyEnum type='com.example.Season'>
12. <option value='${enumValue}'>${enumValue.display}</option>
13. </t:everyEnum>
14. </select>
You have decided to use the Simple tag model to create this tag handler.

Which tag handler method will accomplish this goal?
A. public void doTag() throw JspException {
try {
for (Enum value : getEnumValues()) {
pageContext.setAttribute("enumValue", value);
getJspBody().invoke(getOut());
}
} (Exception e) { throw new JspException(e); }
}
B. public void doTag() throw JspException {
try {
for (Enum value : getEnumValues()) {
getJspContext().setAttribute("enumValue", value);
getJspBody().invoke(null);
}
} (Exception e) { throw new JspException(e); }
}
C. public void doTag() throw JspException {
try {
for (Enum value : getEnumValues()) {
getJspContext().setAttribute("enumValue", value);
getJspBody().invoke(getJspContext().getWriter());
}
} (Exception e) { throw new JspException(e); }
}
D. public void doTag() throw JspException {
try {
for (Enum value : getEnumValues()) {
pageContext.setAttribute("enumValue", value);
getJspBody().invoke(getJspContext().getWriter());
}
} (Exception e) { throw new JspException(e); }

}
Answer: B

Q46. Given in a single JSP page:
<%@ taglib prefix='java' uri='myTags' %>
<%@ taglib prefix='JAVA' uri='moreTags' %>
Which two are true? (Choose two.)
A. The prefix 'java' is reserved.
B. The URI 'myTags' must be properly mapped to a TLD file by the web container.
C. A translation error occurs because the prefix is considered identical by the web container.
D. For the tag usage <java:tag1/>, the tag1 must be unique in the union of tag names in 'myTags' and 'moreTags'.
Answer: A, B
Q47. In a JSP-centric shopping cart application, you need to move a client's home address of the Customer object into the shipping address of the Order object. The address data is stored in a value object class called Address with properties for: street address, city, province, country, and postal code. Which two JSP code snippets can be used to accomplish this goal? (Choose two.)
A. <c:set var='order' property='shipAddress'
value='${client.homeAddress}' />
B. <c:set target='${order}' property='shipAddress'
value='${client.homeAddress}' />
C. <jsp:setProperty name='${order}' property='shipAddress'
value='${client.homeAddress}' />
D. <c:set var='order' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:store>
E. <c:set target='${order}' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:set>
F. <c:setProperty name='${order}' property='shipAddress'>
<jsp:getProperty name='client' property='homeAddress' />
</c:setProperty>
Answer: B, E

Q48. Given that a scoped attribute cart exists only in a user's session, which two, taken independently, ensure the scoped attribute cart no longer exists? (Choose two.)

A. ${cart = null}
B. <c:remove var="cart" />
C. <c:remove var="${cart}" />
D. <c:remove var="cart" scope="session" />

E. <c:remove scope="session">cart</c:remove>
F. <c:remove var="${cart}" scope="session" />
G. <c:remove scope="session">${cart}</c:remove>
Answer: B, D

Q49. In which three directories, relative to a web application's root, may a tag library descriptor file reside when deployed directly into a web application? (Choose three.)
A. /WEB-INF
B. /META-INF
C. /WEB-INF/tlds
D. /META-INF/tlds
E. /WEB-INF/resources
F. /META-INF/resources

Answer: A, C, E
Q50. You have been contracted to create a web site for a free dating service. One feature is the ability for one client to send a message to another client, which is displayed in the latter client's private page. Your contract explicitly states that security is a high priority. Therefore, you need to prevent cross-site hacking in which one user inserts JavaScript code that is then rendered and invoked when another user views that content. Which two JSTL code snippets will prevent cross-site hacking in the scenario above? (Choose two.)

A. <c:out>${message}</c:out>
B. <c:out value='${message}' />
C. <c:out value='${message}' escapeXml='true' />
D. <c:out eliminateXml='true'>${message}</c:out>
E. <c:out value='${message}' eliminateXml='true' />

Answer: B, C

Q51. A custom tag is defined to take three attributes. Which two correctly invoke the tag within a JSP page? (Choose two.)
A. <prefix:myTag a="foo" b="bar" c="baz" />
B. <prefix:myTag attributes={"foo","bar","baz"} />
C. <prefix:myTag jsp:attribute a="foo" b="bar" c="baz" />
D. <prefix:myTag>
<jsp:attribute a:foo b:bar c:baz />
</prefix:myTag>
E. <prefix:myTag>
<jsp:attribute ${"foo", "bar", "baz"} />
</prefix:myTag>
F. <prefix:myTag>
<jsp:attribute a="foo" b="bar" c="baz"/>
</prefix:myTag>
G. <prefix:myTag>
<jsp:attribute name="a">foo</jsp:attribute>
<jsp:attribute name="b">bar</jsp:attribute>
<jsp:attribute name="c">baz</jsp:attribute>
</prefix:myTag>
Answer: A, G

Q52. Which two are true about the JSTL core iteration custom tags? (Choose two.)
A. It may iterate over arrays, collections, maps, and strings.
B. The body of the tag may contain EL code, but not scripting code.
C. When looping over collections, a loop status object may be used in the tag body.
D. It may iterate over a map, but only the key of the mapping may be used in the tag body.
E. When looping over integers (for example begin='1' end='10'), a loop status object may not be used in the tag body.
Answer: A, C
Q53. Which two are valid and equivalent? (Choose two.)

A. <%! int i; %>
B. <%= int i; %>
C. <jsp:expr>int i;</jsp:expr>
D. <jsp:scriptlet>int i;</jsp:scriptlet>
E. <jsp:declaration>int i;</jsp:declaration>

Answer: A, E

Plot No : 202, IInd Floor ,HUDA Maitrivanam,Ameerpet, Hyd-5000038.
www.durgasoft.com


LEARN FROM EXPERTS ...
COMPLETE JAVA
CORE JAVA, ADV. JAVA, ORACLE, STRUTS, HIBERNATE, SPRING, WEB SERVICES,...
COMPLETE .NET
C#.NET, ASP.NET, SQL SERVER, MVC 5 & WCF
TESTING TOOLS
MANUAL + SELENIUM
ORACLE
D2K
MSBI
SHARE POINT
HADOOP
ANDROID
C, C++, DS, UNIX
CRT & APTITUDE TRAINING
AN ISO 9001:2008 CERTIFIED
DURGA
Software Solutions®
202, 2nd Floor, HUDA Maitrivanam,
Ameerpet, Hyd. Ph: 040-64512786,
9246212143, 8096969696
www.durgasoft.com