Exam Code: 310-083

Exam Name: Sun Certified Web Component Developer for J2EE 5

Vendor: Sun

Version: DEMO

Part: A

1: To take advantage of the capabilities of modern browsers that use web standards, such as XHTML and CSS, your web application is being converted from simple JSP pages to JSP Document format. However, one of your JSPs, /scripts/screenFunctions.jsp, generates a JavaScript file. This file is included in several web forms to create screen-specific validation functions and are included in these pages with the following statement:

10. <head>

11.   <script src='/scripts/screenFunctions.jsp'

12.   language='javascript'

13.   type='application/javascript'> </script>

14. </head>

15. <!-- body of the web form -->

Which JSP code snippet declares that this JSP Document is a JavaScript file?

A.<%@ page contentType='application/javascript' %>

B.<jsp:page contentType='application/javascript' />

C.<jsp:document contentType='application/javascript' />

D.<jsp:directive.page contentType='application/javascript' />

E.No declaration is needed because the web form XHTML page already declares the MIME type of the /scripts/screenFunctions.jsp file in the <script> tag.

Correct Answers: D

2: Given the JSP code:

10. <html>

11. <body>

12. <jsp:useBean id='customer' class='com.example.Customer' />

13. Hello, ${customer.title} ${customer.lastName}, welcome

14. to Squeaky Beans, Inc.

15. </body>

16. </html>

Which three types of JSP code are used? (Choose three.)

A.Java code

B.template text

C.scripting code

D.standard action

E.expression language

Correct Answers: B D E

3: 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>

Correct Answers: D

4: Which implicit object is used in a JSP page to retrieve values associated with <context-param>
entries in the deployment descriptor?

A.config

B.request

C.session

D.application

Correct Answers: D

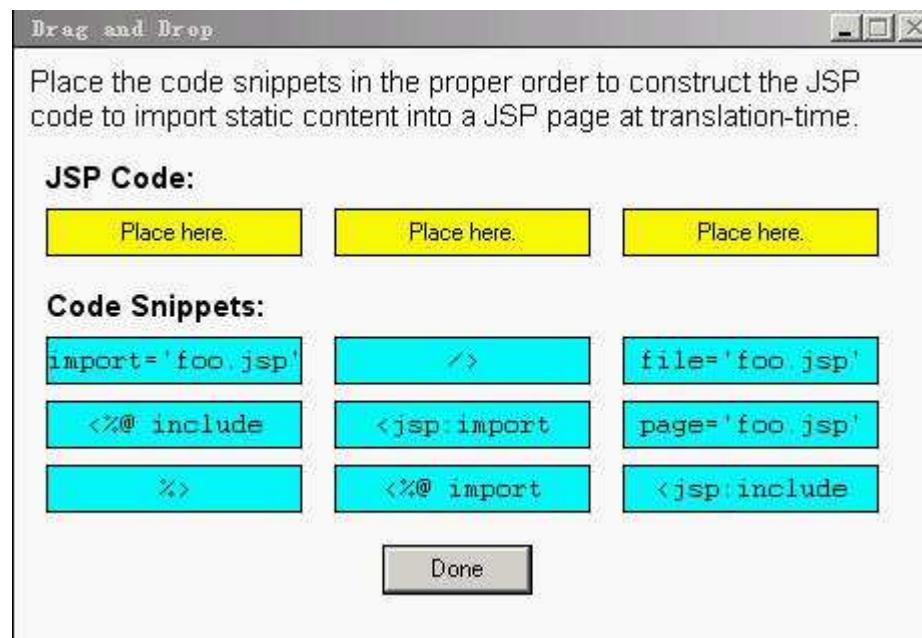5: Click the Task button.

Place the events in the order they occur.

Correct Answers:

Drag and Drop

Place the events in the order they occur.

Order of Steps	Events
JSP page is translated	jspInit is called
JSP page is compiled	JSP page implementation class is loaded
JSP page implementation class is loaded	JSP page is compiled
JSP page implementation is instantiated	jspDestroy is called
jspInit is called	JSP page implementation is instantiated
_jspService is called	JSP page is translated
jspDestroy is called	_jspService is called

Done
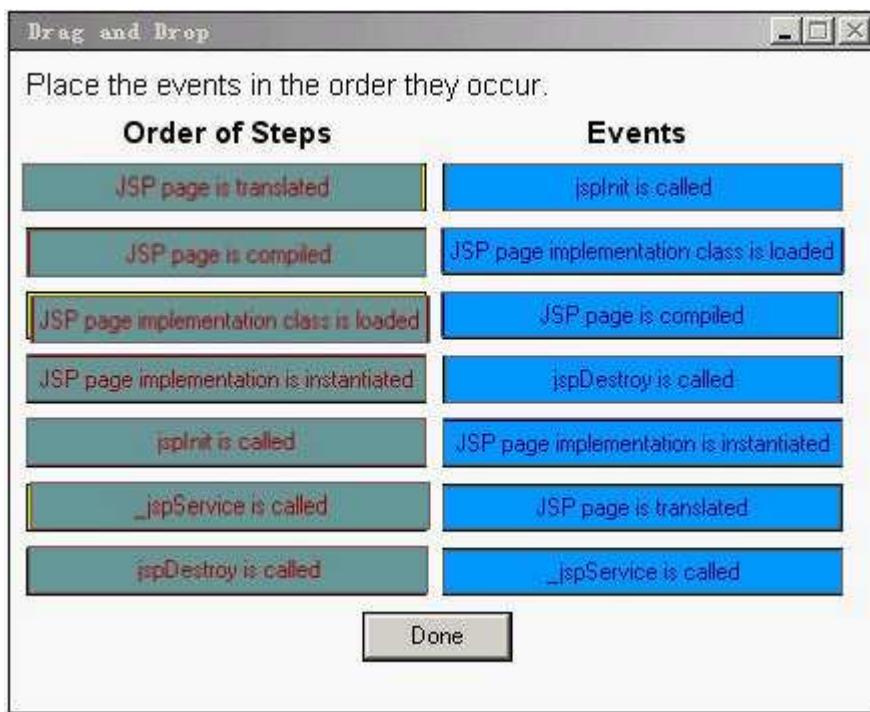
6: Click the Task button.

Place the code snippets in the proper order to construct the JSP code to import static content into a JSP page at translation-time.

Drag and Drop

Place the code snippets in the proper order to construct the JSP code to import static content into a JSP page at translation-time.

JSP Code:

Place here.	Place here.	Place here.

Code Snippets:

import='foo.jsp'	/>	file='foo.jsp'
<%@ include	<jsp:import	page='foo.jsp'
%>	<%@ import	<jsp:include

Done

Correct Answers:

Drag and Drop

Place the code snippets in the proper order to construct the JSP code to import static content into a JSP page at translation-time.

JSP Code:

| <%@ include | file='foo.jsp' | /> |

Code Snippets:

import='foo.jsp'	/>	file='foo.jsp'
<%@ include	<jsp:import	page='foo.jsp'
%>	<%@ import	<jsp:include

Done

7: You have created a JSP that includes instance variables and a great deal of scriptlet code. Unfortunately, after extensive load testing, you have discovered several race conditions in your JSP scriptlet code. To fix these problems would require significant recoding, but you are already behind schedule. Which JSP code snippet can you use to resolve these concurrency problems?

A.<%@ page isThreadSafe='false' %>

B.<%@ implements SingleThreadModel %>

C.<%! implements SingleThreadModel %>

D.<%@ page useSingleThreadModel='true' %>

E.<%@ page implements='SingleThreadModel' %>

Correct Answers: A

8: 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.

Correct Answers: A

9: 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

}

Correct Answers: E

10: 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);
Correct Answers: D