TestpassportQ&A

Bessere Qualität , bessere Dienstleistungen

http://www.testpassport.de

Wir bieten Ihnen einen kostenlosen einjährigen Upgrade Service an

Exam : **310-083**

Title : Sun Certified Web Component Developer for J2EE 5

Version : Demo

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.

Answer: 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

Answer: BDE

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>

Answer: 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

Answer: D

5. 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' %>

Answer: A

6. 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

7. 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
}
```

Answer: E

8. 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

9. 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: BE

10. 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

11. Servlet A receives a request that it forwards to servlet B within another web application in the same web container. Servlet A needs to share data with servlet B and that data must not be visible to other servlets in A's web application. In which object can the data that A shares with B be stored?

A. HttpSession

B. ServletConfig

C. ServletContext

D. HttpServletRequest

E. HttpServletResponse

Answer: D

12. Your web site has many user-customizable features, for example font and color preferences on web pages. Your IT department has already built a subsystem for user preferences using the Java SE platform's lang.util.prefs package APIs, and you have been ordered to reuse this subsystem in your web application. You need to create an event listener that constructs the preferences factory and stores it in the application scope for later use. Furthermore, this factory requires that the URL to a database must

be declared in the deployment descriptor like this:

```
42. <context-param>
43.     <param-name>prefsDbURL</param-name>
44.     <param-value>
45.         jdbc:pointbase:server://dbhost:4747/prefsDB
46.     </param-value>
47. </context-param>
```

Which partial listener class will accomplish this goal?

A.
```
public class PrefsFactoryInitializer implements ContextListener {
public void contextInitialized(ServletContextEvent e) {
ServletContext ctx = e.getContext();
String prefsURL = ctx.getParameter("prefsDbURL");
PreferencesFactory myFactory = makeFactory(prefsURL);
ctx.putAttribute("myPrefsFactory", myFactory);
}
// more code here
}
```

B.
```
public class PrefsFactoryInitializer implements ServletContextListener {
public void contextCreated(ServletContext ctx) {
String prefsURL = ctx.getInitParameter("prefsDbURL");
PreferencesFactory myFactory = makeFactory(prefsURL);
ctx.setAttribute("myPrefsFactory", myFactory);
}
// more code here
}
```

C.
```
public class PrefsFactoryInitializer implements ServletContextListener {
public void contextInitialized(ServletContextEvent e) {
ServletContext ctx = e.getServletContext();
String prefsURL = ctx.getInitParameter("prefsDbURL");
PreferencesFactory myFactory = makeFactory(prefsURL);
ctx.setAttribute("myPrefsFactory", myFactory);
}
// more code here
}
```

D. public class PrefsFactoryInitializer implements ContextListener {
public void contextCreated(ServletContext ctx) {
String prefsURL = ctx.getParameter("prefsDbURL");
PreferencesFactory myFactory = makeFactory(prefsURL);
ctx.putAttribute("myPrefsFactory", myFactory);
}
// more code here
}

Answer: C

13. A developer wants a web application to be notified when the application is about to be shut down. Which two actions are necessary to accomplish this goal? (Choose two.)

A. include a listener directive in a JSP page

B. configure a listener in the TLD file using the <listener> element

C. include a <servlet-destroy> element in the web application deployment descriptor

D. configure a listener in the application deployment descriptor, using the <listener> element

E. include a class implementing ServletContextListener as part of the web application deployment

F. include a class implementing ContextDestroyedListener as part of the web application deployment

G. include a class implementing HttpSessionAttributeListener as part of the web application deployment

Answer: DE

14. You want to create a filter for your web application and your filter will implement javax.servlet.Filter. Which two statements are true? (Choose two.)

A. Your filter class must implement an init method and a destroy method.

B. Your filter class must also implement javax.servlet.FilterChain.

C. When your filter chains to the next filter, it should pass the same arguments it received in its doFilter method.

D. The method that your filter invokes on the object it received that implements javax.servlet.FilterChain can invoke either another filter or a servlet.

E. Your filter class must implement a doFilter method that takes, among other things, an HTTPServletRequest object and an HTTPServletResponse object.

Answer: AD

15. Which three are true about the HttpServletRequestWrapper class? (Choose three.)

A. The HttpServletRequestWrapper is an example of the Decorator pattern.

B. The HttpServletRequestWrapper can be used to extend the functionality of a servlet request.

C. A subclass of HttpServletRequestWrapper CANNOT modify the behavior of the getReader method.

D. An HttpServletRequestWrapper may be used only by a class implementing the javax.servlet.Filter interface.

E. An HttpServletRequestWrapper CANNOT be used on the request passed to the RequestDispatcher.include method.

F. An HttpServletRequestWrapper may modify the header of a request within an object implementing the javax.servlet.Filter interface.

Answer: ABF

16. Click the Exhibit button.

The resource requested by the RequestDispatcher is available and implemented by the DestinationServlet.

What is the result?

```
// From file SourceServlet.java
11. public class SourceServlet extends
HttpServlet {
12.   public void service(HttpServletRequest
request,
13.                       HttpServletResponse
response)
14.         throws ServletException,
IOException {
15.     ServletContext
cxt=getServletConfig().getServletContext();
16.     RequestDispatcher rd =
17.               cxt.getRequestDispatcher("/dest
n");
18.     response.getWriter().println("hello
from source");
19.     response.flushBuffer();
20.     rd.forward(request, response);
21.   }
22. }

// From file DestinationServlet.java
11. public class DestinationServlet extends
HttpServlet {
12.   public void service(HttpServletRequest
request,
13.                       HttpServletResponse
response)
14.         throws ServletException,
IOException {
15.     response.getWriter().println("hello
from dest");
17.     response.flushBuffer();
18.   }
19. }
```

A. An exception is thrown at runtime by SourceServlet.

B. An exception is thrown at runtime by DestinationServlet.

C. Only "hello from dest" appears in the response output stream.

D. Both "hello from source" and "hello from dest" appear in the response output stream.

Answer: A

17. A developer wants to make a name attribute available to all servlets associated with a particular user, across multiple requests from that user, from the same browser instance.

Which two provide this capability from within a tag handler? (Choose two.)

A. pageContext.setAttribute("name", theValue);

B. pageContext.setAttribute("name", getSession());

C. pageContext.getRequest().setAttribute("name", theValue);

D. pageContext.getSession().setAttribute("name", theValue);

E. pageContext.setAttribute("name", theValue,

PageContext.PAGE_SCOPE);

F. pageContext.setAttribute("name", theValue,

PageContext.SESSION_SCOPE);

Answer: DF

18. Given the definition of MyServlet:

```
11. public class MyServlet extends HttpServlet {
12.     public void service(HttpServletRequest request,
13.                                 HttpServletResponse response)
14.                 throws ServletException, IOException {
15.         HttpSession session = request.getSession();
16          session.setAttribute("myAttribute","myAttributeValue");
17.         session.invalidate();
18.         response.getWriter().println("value=" +
19.                                 session.getAttribute("myAttribute"));
20.     }
21. }
```

What is the result when a request is sent to MyServlet?

A. An IllegalStateException is thrown at runtime.

B. An InvalidSessionException is thrown at runtime.

C. The string "value=null" appears in the response stream.

D. The string "value=myAttributeValue" appears in the response stream.

Answer: A

19. You need to store a Java long primitive attribute, called customerOID, into the session scope. Which two code snippets allow you to insert this value into the session? (Choose two.)

A. long customerOID = 47L;

session.setAttribute("customerOID", new Long(customerOID));

B. long customerOID = 47L;

session.setLongAttribute("customerOID", new Long(customerOID));

C. long customerOID = 47L;

session.setAttribute("customerOID", customerOID);

D. long customerOID = 47L;

session.setNumericAttribute("customerOID", new Long(customerOID));

E. long customerOID = 47L;

session.setLongAttribute("customerOID", customerOID);

F. long customerOID = 47L;

session.setNumericAttribute("customerOID", customerOID);

Answer: AC

20. A developer for the company web site has been told that users may turn off cookie support in their browsers. What must the developer do to ensure that these customers can still use the web application?

A. The developer must ensure that every URL is properly encoded using the appropriate URL rewriting APIs.

B. The developer must provide an alternate mechanism for managing sessions and abandon the HttpSession mechanism entirely.

C. The developer can ignore this issue. Web containers are required to support automatic URL rewriting when cookies are not supported.

D. The developer must add the string id=<sessionid> to the end of every URL to ensure that the conversation with the browser can continue.

Answer: A

21. Your web application requires the adding and deleting of many session attributes during a complex use case. A bug report has come in that indicates that an important session attribute is being deleted too soon and a NullPointerException is being thrown several interactions after the fact. You have decided to create a session event listener that will log when attributes are being deleted so you can track down when the attribute is erroneously being deleted.

Which listener class will accomplish this debugging goal?

A. Create an HttpSessionAttributeListener class and implement the attributeDeleted method and log the attribute name using the getName method on the event object.

B. Create an HttpSessionAttributeListener class and implement the attributeRemoved method and log the attribute name using the getName method on the event object.

C. Create an SessionAttributeListener class and implement the attributeRemoved method and log the attribute name using the getAttributeName method on the event object.

D. Create an SessionAttributeListener class and implement the attributeDeleted method and log the attribute name using the getAttributeName method on the event object.

Answer: B

22. As a convenience feature, your web pages include an Ajax request every five minutes to a special servlet that monitors the age of the user's session. The client-side JavaScript that handles the Ajax callback displays a message on the screen as the session ages. The Ajax call does NOT pass any cookies, but it passes the session ID in a request parameter called sessionID. In addition, assume that your webapp keeps a hashmap of session objects by the ID. Here is a partial implementation of this servlet:

```
10. public class SessionAgeServlet extends HttpServlet {
11.     public void service(HttpServletRequest request, HttpServletResponse) throws IOException {
12.         String sessionID = request.getParameter("sessionID");
13.         HttpSession session = getSession(sessionID);
14.         long age = // your code here
15.         response.getWriter().print(age);
16.     } ...    // more code here
 47. }
```

Which code snippet on line 14, will determine the age of the session?

A. session.getMaxInactiveInterval();

B. session.getLastAccessed().getTime() - session.getCreationTime().getTime();

C. session.getLastAccessedTime().getTime() - session.getCreationTime().getTime();

D. session.getLastAccessed() - session.getCreationTime();

E. session.getMaxInactiveInterval() - session.getCreationTime();

F. session.getLastAccessedTime() - session.getCreationTime();

Answer: F

23. Which statement is true about web container session management?

A. Access to session-scoped attributes is guaranteed to be thread-safe by the web container.

B. To activate URL rewriting, the developer must use the HttpServletResponse.setURLRewriting method.

C. If the web application uses HTTPS, then the web container may use the data on the HTTPS request stream to identify the client.

D. The JSESSIONID cookie is stored permanently on the client so that a user may return to the web application and the web container will rejoin that session.

Answer: C

24. One of the use cases in your web application uses many session-scoped attributes. At the end of the use case, you want to clear out this set of attributes from the session object. Assume that this static variable holds this set of attribute names:

```
201. private static final Set<String> USE_CASE_ATTRS;
202. static {
203.     USE_CASE_ATTRS.add("customerOID");
204.     USE_CASE_ATTRS.add("custMgrBean");
205.     USE_CASE_ATTRS.add("orderOID");
206.     USE_CASE_ATTRS.add("orderMgrBean");
207. }
```

Which code snippet deletes these attributes from the session object?

A. session.removeAll(USE_CASE_ATTRS);

B. for (String attr : USE_CASE_ATTRS) {
session.remove(attr);
}

C. for (String attr : USE_CASE_ATTRS) {
session.removeAttribute(attr);
}

D. for (String attr : USE_CASE_ATTRS) {
session.deleteAttribute(attr);
}

E. session.deleteAllAttributes(USE_CASE_ATTRS);

Answer: C

25. 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

26. 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

27. 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: DE

28. 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

29. 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: ABE

30. 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