



SUN 310-083

Exam Name: Sun Certified Web Component Developer for J2EE 5

Q & A : 276 Q&As

Pdf Demo

Quality and Value for the 310-083 Exam

[Exam4Dumps Practice test](#) for SUN SCWCD 310-083 are written to the highest standards of technical accuracy, using only certified subject matter experts and published authors for development. our products of the latest 310-083 exam dumps,310-083 questions and answers is the real 310-083 practice test.

[SCWCD Certification](#) 310-083 Q&A are created by senior IT lecturers in exam4dumps certification Q&A network and SCWCD product experts combination PROMETRIC or VUE true-to-date environmental examination of the original title.

The Questions & Answers cover the latest real **310-083 practice test** and with all the correct answer.we promise the 310-083 Q&A for SCWCD 310-083 (Sun Certified Web Component Developer for J2EE 5) examination of original title complete coverage.310-083 Questions & Answers help you pass the exam. Otherwise,we will give you a full refund.

exam4dumps professional provide SCWCD 310-083 the newest Q&A, completely covers 310-083 test original topic. With our complete SCWCD braindumps resources, you will minimize your SCWCD cost and be ready to pass your 310-083 tests on Your First Try, 100% Money Back Guarantee included!

100% Guarantee to Pass Your 310-083 Exam

If you do not pass the SCWCD 310-083 exam on your first attempt using our Exam4Dumps **310-083 testing engine and pdf study guide**, we will give you a FULL REFUND of your purchasing fee.

Downloadable, Interactive 310-083 Testing engines and PDF Version

Our Braindumps Preparation Material provides you everything you will need to take a [SCWCD certification](#) examination. Details are researched and produced by [SUN Certification](#) Experts who are constantly using industry experience to produce precise, and logical.

Free 310-083 Exams:

This is demo only, this pdf do not include the questions and answers pictures

1. 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

2. }

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

3. <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

4. 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

5. 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

6. 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

```

7. 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

8. 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

9. Click the Exhibit button.

The attribute "name" has a value of "Foo,"

What is the result if this tag handler's tag is invoked?

- 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

10. 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

11. 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

- 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

13. `</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

14. 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

15. 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

16. `<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

17. }

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

18. `<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

19. 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

20. Click the Exhibit button.

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

What is the result?

- 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

21. 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

More [310-083 dumps](#) Information

Related 310-083 dumps

310-083	310-081	310-084	310-082
---------	---------	---------	---------

Other SUN dumps

310-014	310-083	310-302	212-302	310-066
310-560	310-880	311-203	212-812	310-084
310-056	310-878	310-016	310-043	310-813
310-231	310-811	310-015	310-330	310-400