KTest

Exam : SUN 310-083

Title : SUN 310-083 Exam

Update : Demo

1.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='${currentOID}' />
</c:import>
<ul>
  <c:forTokens items'${dataString.split(",")}' var='item'>
    <li>${item}</li>
  </c:forTokens>
</ul>
```

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

2.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

3.Which two are characteristics of the Service Locator pattern? (Choose two.)

A. It encapsulates component lookup procedures.

B. It increases source code duplication and decreases reuse.

C. It improves client performance by caching context and factory objects.

D. It degrades network performance due to increased access to distributed lookup services.

Answer:AC

4.Which three are valid URL mappings to a servlet in a web deployment descriptor? (Choose three.)

A. */*

B. *.do

C. MyServlet

D. /MyServlet

E. /MyServlet/*

F. MyServlet/*.jsp

Answer:BDE

5.Which two are true regarding a web application class loader? (Choose two.)

A. A web application may override the web container's implementation classes.

B. A web application running in a J2EE product may override classes in the javax.* namespace.

C. A web application class loader may NOT override any classes in the java.* and javax.* namespaces.

D. Resources in the WAR class directory or in any of the JAR files within the library directory may be accessed using the J2SE semantics of getResource.

E. Resources in the WAR class directory or in any of the JAR files within the library directory CANNOT be accessed using the J2SE semantics of getResource.

Answer: CD

6.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

7.Given an HttpServletRequest request and an HttpServletResponse response:

```
41: HttpSession session = null;
42: // insert code here
43: if(session == null) {
44:     // do something if session does not exist
45: } else {
46:     // do something if session exists
47: }
```

To implement the design intent, which statement must be inserted at line 42?

A. session = response.getSession();

B. session = request.getSession();

C. session = request.getSession(true);

D. session = request.getSession(false);

E. session = request.getSession("jsessionid");

Answer:D

8. Given:

```
 3: class MyServlet extends HttpServlet {
 4:     public void doPut(HttpServletRequest req,
                          HttpServletResponse resp)
                   throws ServletException, IOException {
 5:         // servlet code here
...
26:     }
27: }
```

If the DD contains a single security constraint associated with MyServlet and its only <http-method> tags and <auth-constraint> tags are:

<http-method>GET</http-method>

<http-method>PUT</http-method>

<auth-constraint>Admin</auth-constraint>

Which four requests would be allowed by the container? (Choose four.)

A. A user whose role is Admin can perform a PUT.

B. A user whose role is Admin can perform a GET.

C. A user whose role is Admin can perform a POST.

D. A user whose role is Member can perform a PUT.

E. A user whose role is Member can perform a POST.

F. A user whose role is Member can perform a GET.

Answer: ABCE

9.For debugging

purposes,

---

you need to record how many times a given JSP is invoked before the user's session has been created. The JSP's destroy method stores

---

this information to a database.

Which JSP code snippet keeps

---

track of this count for the lifetime of the JSP page?

A. <%! int count = 0; %>
<% if (request.getSession(false) == null) count++; %>

B. <%@ int count = 0; %>
<% if (request.getSession(false) == null) count++; %>

C. <% int count = 0;
if (request.getSession(false) == null) count++; %>

D. <%@ int count = 0;
if (request.getSession(false) == null) count++; %>

E. <%! int count = 0;
if (request.getSession(false) == null) count++; %>

Answer:A

10.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

11.1: package com:example;

2: import java:util:*;

3: public class Appliance {

4: private Map props;

5: public Appliance() {

6: this:props = new HashMap getProperties() {

10: return this:props;

11: }

12: private void initialize() {

13: // code to load appliance properties

14: }

15: }

Click the Exhibit button .

The Appliance class is a Singleton that loads a set of properties into a Map from an external data source.

Assume:

? An instance of the Appliance class exists in the application-scoped attribute, appl

? The appliance object includes

---

the name property that

---

maps to the value Cobia

? The request-scoped attribute, prop, has the value name.

Which two EL code snippets will display the string Cobia? (Choose two.)

A. ${appl.properties.name}

B. ${appl.properties.prop}

C. ${appl.properties[prop]}

D. ${appl.properties[name]}

E. ${appl.getProperties().get(prop)}

F. ${appl.getProperties().get('name')}

Answer:AC

12.1: package com:example;

2:

3: public class Product {

4: private String name;

5: private double price;

6:

7: public Product() {

8: this("Default", 0:0);

9: }

10:


```
11: public Product( String name, double price ) {
12: this:name = name;
13: this:price = price;
14: }
15:
16: public String getName() {
17: return name;
18: }
19:
20: public void setName(String name) {
21: this:name = name;
22: }
23:
24: public double getPrice() {
25: return price;
26: }
27:
28: public void setPrice(double price) {
29: this:price = price;
30: }
31: }
```

Click the Exhibit button.

Given:

```
10: <form action='create_product.jsp'>
11:     Product Name: <input type='text' name='prodName'/><br/>
12:     Product Price: <input type='text' name='prodPrice'/><br/>
13: </form>
```

For a given product instance, which three jsp:setProperty attributes must be used to initialize its properties from the HTML form? (Choose three.)

A. id

B. name

C. type

D. param

E. property

F. reqParam

G. attribute

Answer:BDE

13.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

14.You are creating a web form with this HTML:

```
11: <form action="sendOrder.jsp">
12:     <input type="text" name="creditCard">
13:     <input type="text" name="expirationDate">
14:     <input type="submit">
15: </form>
```

Which HTTP method is used when sending this request from the browser?

A. GET

B. PUT

C. POST

D. SEND

E. FORM

Answer: A

15.Your web application requires the ability to load and remove web files dynamically to the web container's file system.

Which two HTTP methods are used to perform these actions? (Choose two.)

A. PUT

B. POST

C. SEND

D. DELETE

E. REMOVE

F. DESTROY

Answer:AD

16.A web browser need NOT always perform a complete request for a particular page that it suspects might NOT have changed. The HTTP specification provides a mechanism for the browser to retrieve only

_____

a partial response from the web server; this response includes

_____

information,

_____

such as the Last-Modified date but NOT the body of the page.

Which HTTP method will

_____

the browser use to retrieve such a partial response?

A. GET

B. ASK

C. SEND

D. HEAD

E. TRACE

F. OPTIONS

Answer: D

17.Your web page includes a Java SE v1.5 applet with the following declaration:

11: <object classid='clsid:CAFEEFAC-0015-0000-0000-ABCDEFFEDCBA'

12: width='200' height='200'>

13: <param name='code' value='Applet.class' />

14: </object>

Which HTTP method is used to retrieve the applet code?

A. GET

B. PUT

C. POST

D. RETRIEVE

Answer:A

18.Which retrieves all cookies sent in a given HttpServletRequest request?

A. request.getCookies()

B. request.getAttributes()

C. request.getSession().getCookies()

D. request.getSession().getAttributes()

Answer:A

19. Given a header in an HTTP request:

X-Retries: 4

Which two retrieve the value of the header from a given ————————————HttpServletRequest request? (Choose two.)

A. request.getHeader("X-Retries")

B. request.getIntHeader("X-Retries")

C. request.getRequestHeader("X-Retries")

D. request.getHeaders("X-Retries").get(0)

E. request.getRequestHeaders("X-Retries").get(0)

Answer:AB

20.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='${currentOID}' />
</c:import>
<ul>
  <c:forTokens items'${dataString.split(",")}' var='item'>
    <li>${item}</li>
  </c:forTokens>
</ul>
```

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

21.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:BE

22.Given:

```
10: public void service(ServletRequest request,
11:                     ServletResponse response) {
12:    ServletInputStream sis =
13:       // insert code here
14: }
```


Which retrieves the binary input stream on line 13?

A. request.getWriter();

B. request.getReader();

C. request.getInputStream();

D. request.getResourceAsStream();

E. request.getResourceAsStream(ServletRequest.REQUEST);

Answer:C

23.You need to retrieve the username cookie from an HTTP request. If this cookie does NOT exist, then the c variable will be null.

Which code snippet must be used to retrieve this cookie object?

A. 10: Cookie c = request.getCookie("username");

B. 10: Cookie c = null;
11: for (Iterator i = request.getCookies();
12: i.hasNext();) {
13: Cookie o = (Cookie) i.next();
14: if (o.getName().equals("username")) {
15: c = o;
16: break;
17: }
18: }

C. 10: Cookie c = null;
11: for (Enumeration e = request.getCookies();
12: e.hasMoreElements();) {
13: Cookie o = (Cookie) e.nextElement();


```
14:     if ( o.getName().equals("username") ) {
15:         c = o;
16:         break;
17:     }
18: }
```

D.

```
10: Cookie c = null;
11: Cookie[] cookies = request.getCookies();
12: for ( int i = 0; i < cookies.length; i++ ) {
13:     if ( cookies[i].getName().equals("username") ) {
14:         c = cookies[i];
15:         break;
16:     }
17: }
```

Answer:D

24.You need to create a servlet filter that stores all request headers to a database for all requests to the web

application'

---

s home page "/index.jsp".

Which HttpServletRequest method allows you to retrieve all of the request headers?

A. String[] getHeaderNames()

B. String[] getRequestHeaders()

C. java.util.Iterator getHeaderNames()

D. java.util.Iterator getRequestHeaders()

E. java.util.Enumeration getHeaderNames()

F. java.util.Enumeration getRequestHeaders()

Answer:E

25.For an HttpServletResponse response, which two create a custom header? (Choose two.)

A. response.setHeader("X-MyHeader", "34");

B. response.addHeader("X-MyHeader", "34");

C. response.setHeader(new HttpHeader("X-MyHeader", "34"));

D. response.addHeader(new HttpHeader("X-MyHeader", "34"));

E. response.addHeader(new ServletHeader("X-MyHeader", "34"));

F. response.setHeader(new ServletHeader("X-MyHeader", "34"));

Answer:AB

26.For a given ServletResponse response, which two retrieve an object for writing text data? (Choose two.)

A. response.getWriter()

B. response.getOutputStream()

C. response.getOutputWriter()

D. response.getWriter().getOutputStream()

E. response.getWriter(Writer.OUTPUT_TEXT)

Answer:AB

27.Given an HttpServletRequest request and HttpServletResponse response, which sets a cookie "username" with the value "joe" in a servlet?

A. request.addCookie("username", "joe")

B. request.setCookie("username", "joe")

C. response.addCookie("username", "joe")

D. request.addHeader(new Cookie("username", "joe"))

E. request.addCookie(new Cookie("username", "joe"))

F. response.addCookie(new Cookie("username", "joe"))

G. response.addHeader(new Cookie("username", "joe"))

Answer:F

28.Your company has a corporate policy that prohibits storing a customer's credit card number in any corporate database. However, users have complained that they do NOT

want to re-enter their credit card number for each transaction. Your management has decided to use client-side cookies to record the user's credit card number for 120 days. Furthermore, they also want to protect this information during transit from the web browser to the web container; so the cookie must only be transmitted over HTTPS.

Which code snippet creates

the "creditCard" cookie and adds

it to the out

going response to be stored on the user's web browser?

A. 10: Cookie c = new Cookie("creditCard", usersCard);
   11: c.setSecure(true);
   12: c.setAge(10368000);
   13: response.addCookie(c);

B. 10: Cookie c = new Cookie("creditCard", usersCard);
   11: c.setHttps(true);
   12: c.setMaxAge(10368000);
   13: response.setCookie(c);

C. 10: Cookie c = new Cookie("creditCard", usersCard);
   11: c.setSecure(true);
   12: c.setMaxAge(10368000);
   13: response.addCookie(c);

D. 10: Cookie c = new Cookie("creditCard", usersCard);
11. c.setHttps(true);
12: c.setAge(10368000);
13: response.addCookie(c);

E. 10: Cookie c = new Cookie("creditCard", usersCard);
11: c.setSecure(true);
12: c.setAge(10368000);
13: response.setCookie(c);

Answer:C

29.You are creating a servlet that generates stock market graphs. You want to provide the web browser with

_____

precise information about the amount of data being sent in the response stream.

Which two HttpServletResponse methods will

_____

you use to provide this information? (Choose two.)

A. response.setLength(numberOfBytes);

B. response.setContentLength(numberOfBytes);

C. response.setHeader("Length", numberOfBytes);

D. response.setIntHeader("Length", numberOfBytes);

E. response.setHeader("Content-Length", numberOfBytes);

F. response.setIntHeader("Content-Length", numberOfBytes);

Answer:BF

30.Which two prevent a servlet from handling requests? (Choose two.)

A. The servlet's init method returns a non-zero status.

B. The servlet's init method throws a ServletException.

C. The servlet's init method sets the ServletResponse's content length to 0.

D. The servlet's init method sets the ServletResponse's content type to null.

E. The servlet's init method does NOT return within a time period defined by the servlet container.

Answer: BE

31.Given an HttpSession session, a ServletRequest request, and a ServletContext context, which retrieves a URL to /WEB-INF/myconfig.xml within a web application?

A. session.getResource("/WEB-INF/myconfig.xml")

B. request.getResource("/WEB-INF/myconfig.xml")

C. context.getResource("/WEB-INF/myconfig.xml")

D. getClass().getResource("/WEB-INF/myconfig.xml")

Answer:C

32.1: package com.example;

2:

3: import javax.servlet.http.*;

4:

5: public class MyWebDAV extends HttpServlet {

6: private String resourceDirectory;

7:

8: public MyWebDAV(String resDir) {

9: this.resourceDirectory = resDir;

10: }

11: public void doPut(HttpServletRequest req,

12: HttpServletResponse resp) {

13: // store file to resourceDirectory (code not shown)

20: }

21: public void doDelete(HttpServletRequest req,

22: HttpServletResponse resp) {

23: // remove file from resourceDirectory (code not shown)

30: }

31: }

Click the Exhibit button.

As a maintenance

_____

feature, you have created this servlet to allow you to upload and remove files on your web server. Unfortunately, while testing this servlet,

_____

you try to upload a file using an HTTP request and

_____

on this servlet,

_____

the web container returns a 404 status.

What is wrong with this servlet?

A. HTTP does NOT support file upload operations.

B. The servlet constructor must NOT have any parameters.

C. The servlet needs a service method to dispatch the requests to the helper methods.

D. The doPut and doDelete methods d

_____

o NOT map to the proper HTTP methods.

Answer: B

33.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

34.5: 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: }

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

35. 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: AC

36.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:BCD

37.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

38.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

39.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

40.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.

T

---

he 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

41.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

42.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

43.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

44.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

45.1:

2:

3:

7: 1.0

8: h

9: http://example.com/tld/highlight

10:

11: highlight

12: com.example.HighlightTag

13: scriptless

14:

15: color

16: true

17:

18: true

19:

20:

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).

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: ACE

46.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

47. 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

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

A. tag

B. page

C. taglib

D. include

E. variable

Answer: AE

49.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: BE

50.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:CEF

KillTest.com was founded in 2006. The safer,easier way to help you pass any IT Certification exams . We provide high quality IT Certification exams practice questions and answers(Q&A). Especially Adobe, Apple, Citrix, Comptia, EMC, HP, HuaWei, LPI, Nortel, Oracle, SUN, Vmware and so on. And help you pass any IT Certification exams at the first try.

You can reach us at any of the email addresses listed below.

English Customer:	Chinese Customer:
Sales : sales@Killtest.com	sales@Killtest.net
Support: support@Killtest.com	support@Killtest.com

English Version	http://www.KillTest.com
Chinese (Traditional)	http://www.KillTest.net
Chinese (Simplified)	http://www.KillTest.cn