Appendix A:
Final Mock Exam

Do NOT try to take this exam until you believe you're ready for the real thing. If you take it too soon, then when you come back to it again you'll already have some memory of the questions, and it could give you an artificially high score. We really do want you to pass the *first* time. (Unless there were some way to convince you that you need to buy a fresh copy of this book each time you retake the exam...)

To help defeat the "I remember this question" problem, we've made this exam just a little *harder* than the real exam, by *not* telling you how many answers are correct for each of our questions. Our questions and answers are virtually identical to the tone, style, difficulty, and topics of the real exam, but by not telling you how many answers to choose, you can't automatically eliminate any of the answers. It's cruel of us, really, and we wish we could tell you that it hurts us more than it hurts you to have to take the exam this way. (But be grateful—until a few years ago, Sun's real Java exams *were* written this way, where most questions ended with "Choose all that apply.")

Most exam candidates have said that our mock exams *are* a little more difficult than the real SCWCD, but that their scores on our exam and on the real one were very close. This mock exam is a perfect way to see if you're ready, but only if you:

1) Give yourself no more than two hours and 15 minutes to complete it, just like the real exam.

2) Don't look anywhere else in the book while you're taking the exam!

3) Don't take it over and over again. By the fourth time, you might be getting 98% and yet still not be able to pass the real exam, simply because you were memorizing our exact questions and answers.

4) Wait until *after* you finish the exam to consume large quantities of alcohol or other mind-altering substances...

Final Mock Exam

1 This statement describes the potential benefits of a design pattern:

These components pre-process or post-process incoming requests and outgoing responses in a web application, and can be cleanly plugged into the application depending on whether the application requires the component's specialized task.

Which design pattern is being described?

- ❑ A. Transfer Object
- ❑ B. Intercepting Filter
- ❑ C. Model-View-Controller
- ❑ D. Business Delegate
- ❑ E. Front Controller

2 Which are true about the **`jsp:useBean`** standard action? (Choose all that apply.)

- ❑ A. The **`name`** attribute is mandatory.
- ❑ B. The **`scope`** attribute defaults to the page scope.
- ❑ C. The **`type`** and **`class`** attributes must NOT be used together.
- ❑ D. The **`type`** attribute is ONLY used when the bean already exists in one of the four JSP scopes.
- ❑ E. The **`jsp:useBean`** standard action may be used to declare a scripting variable that may be used in scriptlets, such as **`<% myBean.method(); %>`**

3 Given this partial deployment descriptor:

```
12.    <context-param>
13.      <param-name>email</param-name>
14.      <param-value>foo@bar.com</param-value>
15.    </context-param>
16.    <servlet>
17.      <servlet-name>a</servlet-name>
18.      <servlet-class>com.bar.Foo</servlet-class>
19.      <init-param>
20.        <param-name>email</param-name>
21.        <param-value>baz@bar.com</param-value>
22.      </init-param>
23.    </servlet>
```

And, assuming **scfg** is a **ServletConfig** object and **sctx** is a **ServletContext** object, which statement is true?

- ❑ A. **sctx.getInitParameter("email")** will return **baz@bar.com**.
- ❑ B. **scfg.getInitParameter("email")** will return **foo@bar.com**.
- ❑ C. An error will occur because the **email** parameter is defined twice.
- ❑ D. **scfg.getServletContext().getInitParameter("email")** will return **baz@bar.com**.
- ❑ E. **scfg.getServletContext().getInitParameter("email")** will return **foo@bar.com**.
- ❑ F. An error will occur because servlet context initialization parameters should be defined using **init-param**, NOT **context-param**.

4 Given:

```
public class DoubleTag extends SimpleTagSupport {
  private String data;
  public void setData(String d) { data = d; }
  public void doTag() throws JspException, IOException {
    getJspContext().getOut().print(data + data);
  }
}
```

Which is an equivalent tag file?

- ❑ A. `${param.data}${param.data}`
- ❑ B. `<%@ attribute name="data" %>`
  `${data}${data}`
- ❑ C. `<%@ variable name-given="data" %>`
  `${data}${data}`
- ❑ D. `<%@ attribute name="data" %>`
  `<% pageContext.getOut().print(data + data); %>`
- ❑ E. `<%@ variable name-given="data" %>`
  `<% pageContext.getOut().print(data + data); %>`

5 Given a session object **sess** with an attribute named **myAttr**, and an **HttpSessionBindingEvent** object **bind** bound to **sess**.

Which will retrieve **myAttr**? (Choose all that apply.)

- ❑ A. `long myAttr = sess.getAttribute("myAttr");`
- ❑ B. `Object o = sess.getAttribute("myAttr");`
- ❑ C. `String s = sess.getAttribute("myAttr");`
- ❑ D. `Object o = bind.getAttribute("myAttr");`
- ❑ E. `Object o = bind.getSession().getAttribute("myAttr");`

6 Which about JSP Documents (XML-based documents) are true? (Choose all that apply.)

- ❑ A. A JSP document must have a **<jsp:root>** as its top element.
- ❑ B. The following would be valid as the first line of a JSP document: **<jsp:root xmlns:uri="http://java.sun.com/JSP/Page" version="2.0">**.
- ❑ C. In a JSP document, page directives are defined using **<jsp:directive.page>** instead of **<%@ %>**.
- ❑ D. The **<c:forEach>** tag CANNOT be used unless the **c** prefix has been introduced through a namespace attribute.
- ❑ E. Both the standard **<%! %>** syntax as well as **<jsp:declaration>** may be used to declare variables in a JSP document.

7 Which statements about EL implicit objects are true? (Choose all that apply.)

- ❑ A. **${param.name}** produces the value of the parameter **name**.
- ❑ B. **${init.foo}** produces the value of the context initialization parameter named **foo**.
- ❑ C. The implicit object **param** is a **Map** that maps parameter names to single String parameter values.
- ❑ D. **pageContext**, **param**, **header**, and **cookie** are all implicit objects available to EL expressions in JSP pages.
- ❑ E. **page**, **request**, **session**, and **application** are implicit objects that map attribute names to values in the corresponding scopes.

8 Given this JSP snippet:

```
10. <!--X-->
11. <%=A.b()%>
12. <!--<%=A.b()%>-->
13. <%--Y--%>
```

Assume that a call to **A.b()** is valid and returns the text **test**. Ignoring whitespace in the generated response, which represents the HTML this JSP would produce?

- ❑ A. **<!--X-->test<!--<%=A.b()%>-->**
- ❑ B. **<!--X-->test<!--test-->**
- ❑ C. **test**
- ❑ D. **<!--X-->test<!--<%=A.b()%>-->
<%--Y--%>**
- ❑ E. **test<%--Y--%>**
- ❑ F. The generated HTML will be blank.

9 Which are true about tag libraries in web applications? (Choose all that apply.)

- ❑ A. A TLD file must exist in the **WEB-INF/tlds/** directory.
- ❑ B. A TLD file may exist in any subdirectory of **WEB-INF**.
- ❑ C. A TLD file may exist in the **WEB-INF** directory in a JAR file.
- ❑ D. A TLD file may exist in the **META-INF** directory in a JAR file.
- ❑ E. A TLD file in a JAR file must be located at **META-INF/taglib.tld**.

10 You store the SQL source files for web-database work in a web application's top-level **sql** directory, but you do NOT want to make this directory accessible to HTTP requests.

How do you configure the web application to forbid requests to this directory? (Choose all that apply.)

- ❑ A. Protect the server with a firewall.
- ❑ B. Specify the directory with a **<security-role> element in** deployment descriptor.
- ❑ C. Move the directory beneath **WEB-INF**, the contents of which are NOT accessible to application users.
- ❑ D. Create a **<security-constraint>** element in the DD to prevent access to the **sql** directory.

11 Given:

```
11. <% java.util.Map map = new java.util.HashMap();
12.    map.put("a", "1");
13.    map.put("b", "2");
14.    map.put("c", "3");
15.    request.setAttribute("map", map);
16.    request.setAttribute("map-index", "b");
17. %>
18. <%-- insert code here --%>
```

Which , inserted at line 18, are valid and evaluate to **2**? (Choose all that apply.)

- ❑ A. `${map.b}`
- ❑ B. `${map[b]}`
- ❑ C. `${map.map-index}`
- ❑ D. `${map[map-index]}`
- ❑ E. `${map['map-index']}`
- ❑ F. `${map[requestScope['map-index']]}`

12 Given this tag handler class excerpt:

```
11. public class WorthlessTag extends TagSupport {
12.   private String x;
13.   public void setX(String x) { this.x = x; }
14.   public int doStartTag() throws JspException {
15.     try { pageContext.getOut().print(x); }
16.     catch (IOException e) { }
17.     if ("x".equals(x))
18.       return SKIP_BODY;
19.     else
20.       return EVAL_BODY_INCLUDE;
21.   }
22.   public int doEndTag() throws JspException {
23.     try { pageContext.getOut().print("E"); }
24.     catch (IOException e) { }
25.     if ("y".equals(x))
26.       return SKIP_PAGE;
27.     else
28.       return EVAL_PAGE;
29.   }
30. }
```

and given this TLD excerpt:

```
21. <tag>
22.     <name>worthless</name>
23.     <tag-class>com.mycom.WorthlessTag</tag-class>
24.     <body-content>empty</body-content>
25.     <attribute>
26.         <name>x</name>
27.         <required>true</required>
28.         <rtexprvalue>true</rtexprvalue>
29.     </attribute>
30. </tag>
```

(continued on next page.)

12, cont.

and given this complete JSP page:

```
1. <%@ taglib uri="somevaliduri" prefix="w" %>
2. <w:worthless x="x" />
3. <w:worthless x="<%=Boolean.TRUE.toString()%>" />
4. <w:worthless x="y" />
5. <w:worthless x="z" />
```

What output does the JSP generate?

- ❑ A. **xE**
- ❑ B. **x trueE yE**
- ❑ C. **xE trueE yE**
- ❑ D. **xE trueE yE zE**
- ❑ E. **x <%=Boolean.TRUE.toString()%>E yE**
- ❑ F. **xE <%=Boolean.TRUE.toString()%>E yE**
- ❑ G. **xE <%=Boolean.TRUE.toString()%>E yE zE**

13 A user submits a form using an HTML page containing :

```
<form action="/handler">
  <!-- form tags here -->
</form>
```

The URL pattern **/handler** is mapped to an HTTP servlet.

Which **HttpServlet** service method will the web container call in response to this form submit?

- ❑ A. **doHead**
- ❑ B. **doPost**
- ❑ C. **Get**
- ❑ D. **doGet**

14 Which statements concerning welcome files are true? (Choose all that apply.)

- ❑ A. They can be declared in the DD.
- ❑ B. They can be used to respond to 'partial requests'.
- ❑ C. If multiple welcome files are declared for a web app, their ordering in the DD is NOT meaningful.
- ❑ D. J2EE 1.4 compliant containers are required to match partial requests to URLs in the welcome file list using a specified algorithm.
- ❑ E. If a J2EE 1.4 compliant container receives a partial request for which it CANNOT find a match, it must return an HTTP 404 error code.

15 Once a session has been invalidated, which **HttpSession** methods can be called on that session without throwing an **IllegalStateException**? (Choose all that apply.)

- ❑ A. **invalidate**
- ❑ B. **getAttribute**
- ❑ C. **setAttribute**
- ❑ D. **getServletContext**
- ❑ E. **getAttributeNames**

16 Which statements about the **taglib** directive are true? (Choose all that apply.)

- ❑ A. A **taglib** directive always identifies a tag prefix that will distinguish usage of actions in the library.
- ❑ B. **<% taglib uri="http://www.mytags.com/mytags" prefix="mytags" %> is an example of a valid taglib** directive.
- ❑ C. Every **taglib** directive must specify a value for the **uri** attribute.
- ❑ D. Every **taglib** directive must specify a non-empty value for the **prefix** attribute.
- ❑ E. There are three attributes defined for the **taglib** directive: **uri**, **tagdir**, and **prefix**.

17 Which statements about making a servlet's **doGet()** method **synchronized** are true? (Choose all that apply.)

- ❑ A. It will make access to **ServletRequest** attributes thread-safe.
- ❑ B. It will NOT make access to **HttpSession** attributes thread-safe.
- ❑ C. It may have a negative performance impact because the servlet will only be able to process one request at a time.
- ❑ D. It is necessary if the method will be using **HttpSession.getAttribute() or HttpSession.setAttribute()**.
- ❑ E. It is necessary if the method will be using **ServletContext.getAttribute() or ServletContext.setAttribute()**.

18 Which are valid EL implicit variables? (Choose all that apply.)

- ❑ A. **out**
- ❑ B. **request**
- ❑ C. **response**
- ❑ D. **pageContext**
- ❑ E. **contextParam**

19 Given the following URL:

```
http://www.example.com/userConfig/searchByName.do
?first=Bruce&middle=W&last=Perry
```

Which servlet code fragment from a service method, for example **doPost()**, will retrieve the values of all of the query string parameters?

❑ A.
```
String value
  = request.getParameter("Bruce");
```

❑ B.
```
String value
  = getServletContext().getInitParameter("first");
```

❑ C.
```
String value
  = getServletConfig().getInitParameter("first")
```

❑ D.
```
java.util.Enumeration enum
  = request.getParameterNames();
while (enum.hasMoreElements()) {
  String name = (String) enum.nextElement();
  String value = request.getParameter(name);
}
```

❑ E.
```
java.util.Enumeration enum
  = request.getParameterNames();
while (enum.hasMoreElements()) {
  String value = (String) enum.nextElement();
}
```

20 Which are true about EL operators?
(Choose all that apply.)

❑ A. The logical operators treat a **null** value as **false**.

❑ B. The arithmetic operators treat a **null** value as **Double.NaN** (not a number).

❑ C. Divide by zero, **${x div 0}**, throws a runtime exception.

❑ D. Strings in EL expressions are automatically converted into the appropriate numeric or boolean values.

❑ E. A **NullPointerException** is thrown when a **null** is encountered in an arithmetic EL expression.

21 Given the partial TLD:

```
11.   <tag>
12.     <name>getTitle</name>
13.     <tag-class>com.example.taglib.GetTitleTagHandler</tag-class>
14.     <body-content>empty</body-content>
15.     <attribute>
16.       <name>story</name>
17.       <required>false</required>
18.     </attribute>
19.   </tag>
20.   <tag>
21.     <name>printMessage</name>
22.     <tag-class>com.example.taglib.PrintMessageTagHandler</tag-class>
23.     <body-content>JSP</body-content>
24.     <attribute>
25.       <name>section</name>
26.       <required>true</required>
27.     </attribute>
28.   </tag>
```

Which are valid invocations of these tags within a JSP? (Choose all that apply)

- ❑ A.
```
<my:getTitle>
  <my:printMessage />
</my:getTitle>
```
- ❑ B.
```
<my:printMessage section="47">
  <my:getTitle />
</my:printMessage>
```
- ❑ C.
```
<my:getTitle story="">
  <my:printMessage section="47" />
</my:getTitle>
```
- ❑ D.
```
<my:printMessage section="47">
  <my:getTitle story="Shakespear_RJ"></my:getTitle>
</my:printMessage>
```

22 Which JSP code would you use to include static content in a JSP? (Choose all that apply.)

- ❑ A. `<%@ include file="/segments/footer.html" %>`
- ❑ B. `<jsp:forward page="/segments/footer.html" />`
- ❑ C. `<jsp:include page="/segments/footer.html" />`
- ❑ D.
```
RequestDispatcher dispatcher
  = request.getRequestDispatcher("/segments/footer.html");
dispatcher.include(request,response);
```

23 Which statements about the deployment descriptor (DD) are true? (Choose all that apply.)

- ❑ A. The DD must contain at least one **<context-param>** element.
- ❑ B. The DD must be a well-formed XML file.
- ❑ C. The DD can be a text-based properties file or an XML file.
- ❑ D. You can leave out the XML form of the DD and provide a DD as a Java object.
- ❑ E. The **<web-app>** element must be the parent element of all of the other DD elements.

24 Which steps occur before **jspInit()** is called? (Choose all that apply.)

- ❑ A. A class instantiation occurs.
- ❑ B. A Java source file is compiled.
- ❑ C. The **_jspService()** method is called.
- ❑ D. The JSP page is translated to source.
- ❑ E. The **jspCreate()** method is calleD.
- ❑ F. The container supplies a **ServletConfig** reference.

25 Given a Simple tag handler class:

```
11. public class MyTagHandler
            extends SimpleTagSupport {
12.   public void doTag() throws JspException {
13.     try {
14.       // insert code 1
15.       // insert code 2
16.       // insert code 3
17.       JspWriter out = tagContext.getOut();
18.       out.print(requestURI);
19.     } catch (IOException ioe) {
20.       throw new JspException(ioe);
21.     }
22.   }
23. }
```

Which, inserted at lines 14, 15, and 16, will print the request-URI to the response stream?

❑ A.
```
14. JspContext tagContext = pageContext;
15. ServletRequest request
        = (ServletRequest) tagContext.getRequest();
16. String requestURI = request.getRequestURI();
```

❑ B.
```
14. PageContext tagContext = (PageContext) jspContext;
15. ServletRequest request
        = (ServletRequest) tagContext.getRequest();
16. String requestURI = request.getRequestURI();
```

❑ C.
```
14. JspContext tagContext = getJspContext();
15. HttpServletRequest request
        = (HttpServletRequest) tagContext.getRequest();
16. String requestURI = request.getRequestURI();
```

❑ D.
```
14. PageContext tagContext = (PageContext) getJspContext();
15. HttpServletRequest request
        = (HttpServletRequest) tagContext.getRequest();
16. String requestURI = request.getRequestURI();
```

26 Given the following scriptlet code:

```
11. <% String cityParam = request.getParameter("city");
12.     if ( cityParam != null ) { %>
13.       City: <input type='text' name='city' value='<%= cityParam %>' />
14. <% } else { %>
15.       City: <input type='text' name='city' value='Paris' />
16. <% } %>
```

Which JSTL code snippet produces the same result?

❑ A.
```
<c:if test='${not empty param.city}'>
  City: <input type='text' name='city' value='${param.city}' />
<c:else/>
  City: <input type='text' name='city' value='Paris' />
</c:if>
```

❑ B.
```
<c:if test='${not empty param.city'>
 <c:then>
    City: <input type='text' name='city' value='${param.city}' />
  </c:then>
  <c:else>
    City: <input type='text' name='city' value='Paris' />
  </c:else>
</c:if>
```

❑ C.
```
<c:choose test='${not empty param.city}'>
  City: <input type='text' name='city' value='${param.city}' />
<c:otherwise/>
  City: <input type='text' name='city' value='Paris' />
</c:choose>
```

❑ D.
```
<c:choose>
  <c:when test='${not empty param.city}'>
    City: <input type='text' name='city' value='${param.city}' />
  </c:when>
  <c:otherwise>
    City: <input type='text' name='city' value='Paris' />
  </c:otherwise>
</c:choose>
```

27 How would you redirect an HTTP request to another URL? (Choose all that apply)

- ❑ A. `response.sendRedirect("/anotherUrl");`
- ❑ B. `response.encodeRedirectURL("/anotherUrl");`
- ❑ C.
```
response.sendRedirect(
  response.encodeRedirectURL("/anotherUrl"));
```
- ❑ D.
```
RequestDispatcher dispatcher
  = request.getRequestDispatcher("/anotherUrl");
dispatcher.forward(request,response);
```
- ❑ E.
```
RequestDispatcher dispatcher
  = request.getRequestDispatcher("/anotherUrl");
dispatcher.redirect(request,response);
```

28 Given:

```
<%@ page isELIgnored="true" %>
```

Which statements are true? (Choose all that apply.)

- ❑ A. This is an example of a directive.
- ❑ B. This is NOT an example of a directivE.
- ❑ C. It will cause `${a.b}` to be ignored by the container.
- ❑ D. It will NOT cause `${a.b}` to be ignored by the container.
- ❑ E. It will cause the EL expression in `<c:out value="${a.b}"/>` to be ignored by the container.
- ❑ F. It will NOT cause the EL expression in `<c:out value="${a.b}"/>` to be ignored by the container.

29 Given a deployment descriptor with three valid **<security-constraint>** elements, all constraining web resource A. And, given that two of the **<security-constraint>** elements respective **<auth-constraint>** sub--elements are:

```
<auth-constraint>Bob</auth-constraint>
```

and

```
<auth-constraint>Alice</auth-constraint>
```

And that the third **<security-constraint>** element has no **<auth-constraint>** sub-element.

Who can access resource A?

- ❑ A. no one
- ❑ B. anyone
- ❑ C. only Bob
- ❑ D. only Alice
- ❑ E. only Bob and Alice
- ❑ F. anyone but Bob or Alice

30 Given:

```
51. <function>
52.    <name>myfun</name>
53.    <function-class>com.example.MyFunctions</function-class>
54.    <function-signature>
55.       java.util.List getList(java.lang.String name)
56.    </function-signature>
57. </function>
```

Which is true about an invocation of this EL function mapping?

Assume that **pre** is correctly declared by a **taglib** directive.

- ❑ A. EL functions are NOT allowed to return collections.
- ❑ B. **${pre:getList("visitors")[0]}** is a valid invocation.
- ❑ C. **${pre:myfun("visitors")[0]}** is a valid invocation.
- ❑ D. The function signature is invalid because you do NOT need to specify the package information **java.lang** on the method parameter.

31 In an HTML page with a rich, graphical layout, how would you write the JSP standard action code to import a JSP segment that generates a menu that is parameterized by the user's role?

- A.
```
<jsp:include page="user-menu.jsp">
  <jsp:param name="userRole"
             value="${user.role}" />
</jsp:include>
```
- B.
```
<jsp:import page="user-menu.jsp">
  <jsp:param name="userRole"
             value="${user.role}" />
</jsp:import>
```
- C.
```
<jsp:include page="user-menu.jsp">
  <jsp:parameter name="userRole"
                 value="${user.role}" />
</jsp:include>
```
- D.
```
<jsp:import page="user-menu.jsp">
  <jsp:parameter name="userRole"
                 value="${user.role}" />
</jsp:import>
```
- E. This CANNOT be done using a JSP standard action.

32 Given that **resp** is an **HttpServletResponse**, and no custom headers exist in this response before this snippet executes:

```
30.  resp.addHeader("myHeader", "foo");
31.  resp.addHeader("myHeader", "bar");
32.  resp.setHeader("myHeader", "baz");
33.  String [] s = resp.getHeaders("myHeader");
```

What is the value of **s[0]**?

- A. **foo**
- B. **bar**
- C. **baz**
- D. Compilation fails
- E. An exception is thrown at runtime

33 Given a servlet that stores a customer bean in the session scope with the following code snippet:

```
11.   public void doPost(HttpServletRequest req,
12.                       HttpServletResponse resp) {
13.     HttpSession session = req.getSession();
14.     com.example.Customer cust
15.       = new com.example.Customer();
16.     cust.setName(req.getParameter("full_name"));
17.     session.setAttribute("customer", cust);
18.     RequestDispatcher page
19.       = req.getRequestDispatcher("page.jsp");
20.     page.forward(req, resp);
21.   }
```

Which of these complete JSPs will print the customer's name? (Choose all that apply..)

❑ A.
```
1. <%= customer.getName() %>
```

❑ B.
```
1. <jsp:useBean id="customer"
2.              type="com.example.Customer"
3.              scope="session" />
4. <%= customer.getName() %>
```

❑ C.
```
1. <jsp:useBean id="customer"
2.              type="com.example.Customer"
3.              scope="session">
4.   <%= customer.getName() %>
5. </jsp:useBean>
```

❑ D.
```
1. <jsp:useBean id="customer"
2.              type="com.example.Customer"
3.              scope="session" />
4. <jsp:getProperty name="customer"
5.                  property="name" />
```

❑ E.
```
1. <jsp:useBean id="customer"
2.              type="com.example.Customer"
3.              scope="session">
4.   <jsp:getProperty name="customer"
5.                   property="name" />
6. </jsp:useBean>
```

34 Which are valid elements in a DD? (Choose all that apply.)

- ❑ A.
```
<filter>
  ...
  <dispatcher>ERROR</dispatcher>
</filter>
```
- ❑ B.
```
<filter>
  ...
  <dispatcher>CHAIN</dispatcher>
</filter>
```
- ❑ C.
```
<filter>
  ...
  <dispatcher>FORWARD</dispatcher>
</filter>
```
- ❑ D.
```
<filter-mapping>
  ...
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
```
- ❑ E.
```
<filter-mapping>
  ...
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```
- ❑ F.
```
<filter-mapping>
  ...
  <dispatcher>RESPONSE</dispatcher>
</filter-mapping>
```

35 Given that **req** is an **HttpServletRequest**, which returns the names of all the parameters in the request? (Choose all that apply.)

- ❑ A. `Map names = req.getParameterNames();`
- ❑ B. `String [] names = req.getParameters();`
- ❑ C. `Enumeration names = req.getParameters();`
- ❑ D. `String [] names = req.getParameterNames();`
- ❑ E. `Enumeration names = req.getParameterNames();`

36 Which of the following are legal **<error-page>** elements? (Choose all that apply.)

- ❑ A.
```
<error-page>
  <exception-type>java.lang.Throwable</exception-type>
  <location>/error/generic-error.jsp</location>
</error-page>
```
- ❑ B.
```
<error-page>
  <error-code>404</error-code>
  <location>/error/file-not-found.jsp</location>
</error-page>
```
- ❑ C.
```
<error-page>
  <error-code>404</error-code>
  <error-code>403</error-code>
  <location>/error/generic-error.jsp</location>
</error-page>
```
- ❑ D.
```
<error-page>
  <error-code>404</error-code>
  <location>/error/file-not-found.jsp</location>
  <location>/error/generic-error.jsp</location>
</error-page>
```
- ❑ E.
```
<error-page>
  <error-code>404</error-code>
  <exception-type>java.lang.Throwable</exception-type>
  <location>/error/generic-error.jsp</location>
</error-page>
```

37 Given that there exists a **HashMap** attribute called **preferences** in the session-scope.

Which JSTL code structure will put an entry, **color**, into the map with the value of the color request parameter?

- ❑ A.
```
<c:set target="${sessionScope.preferences}"
 property="color" value="${param.color}" />
```
- ❑ B.
```
<c:put map="${sessionScope.preferences}"
 property="color" value="${param.color}" />
```
- ❑ C.
```
<c:set scope="session" var="preferences"
 property="color" value="${param.color}" />
```
- ❑ D.
```
<c:put scope="session" map="preferences"
 property="color" value="${param.color}" />
```

Note: This is a simulated 'Drag and Drop' question, something like what you'll see on the exam:

38 Given the Implicit Objects listed on the left, and actual Java types listed on the right, match the Implicit Objects to their correct Java type:

`out`	`Object`
`application`	`JspWriter`
`config`	`PageAttributes`
`page`	`Writer`
	`JspContext`
	`JspConfig`
	`System`
	`ServletConfig`
	`ServletContext`

39 Which Simple tag handler will iterate over the body content five times?

❑ A.
```
public class MySimpleTag extends SimpleTagSupport {
  public void doTag() throws IOException, JspException {
    for ( int i=0; i<5; i++ ) {
      getJspBody().invoke(null);
    }
  }
)
```

❑ B.
```
public class MySimpleTag extends SimpleTagSupport {
  int count=0;
  public int doTag() throws IOException, JspException {
    getJspBody().invoke(null);
    count++;
    return ( (count<5) ? EVAL_BODY_AGAIN : SKIP_BODY );
  }
}
```

❑ C.
```
public class MySimpleTag extends SimpleTagSupport {
  int count=0;
  public int doStartTag() {
    return EVAL_BODY_INCLUDE;
  }
  public int doEndTag() {
    count++;
    return ( (count<5) ? EVAL_BODY_AGAIN : SKIP_BODY );
  }
}
```

❑ D.
```
public class MySimpleTag extends SimpleTagSupport {
  int count=0;
  public int doStartTag() {
    return EVAL_BODY_INCLUDE;
  }
  public int doAfterBody() {
    count++;
    return ( (count<5) ? EVAL_BODY_AGAIN : SKIP_BODY );
  }
}
```

40 Which of the following statements about the servlet lifecycle are true? (Choose all that apply.)

- ❑ A. The web container calls the **`init()`** and **`service()`** methods in response to each servlet request.
- ❑ B. The web application developer uses an object that implements the **`java.servlet.Filter`** interface to instantiate one or more servlets.
- ❑ C. The web container calls the servlet's **`destroy()`** method when the servlet is destroyed or unloaded.
- ❑ D. The web container loads and instantiates a servlet class, then initializes the servlet by calling its **`init()`** method exactly once, passing into **`init()`** an object that implements the **`javax.servlet.ServletConfig`** interface that the container creates for the servlet.

41 Which about web attributes are true? (Choose all that apply.)

- ❑ A. No attributes are longer lived than session attributes.
- ❑ B. In all scopes, attributes can be retrieved using a **`getAttribute()`** method.
- ❑ C. Context attributes survive a session time-out.
- ❑ D. Only session and context attributes can be retrieved in an enumeration.
- ❑ E. Data stored in both request and session objects is thread safe.

42 Given a JSP page:

```
11. <my:tag1>
12.   <my:tag2>
13.     <%-- JSP content --%>
14.   </my:tag2>
15. </my:tag1>
```

The tag handler for **my:tag1** is **Tag1Handler** and extends TagSupport. The tag handler for **my:tag2** is **Tag2Handler** and extends SimpleTagSupport.

The tag handler for **my:tag1** must have access to the tag handler for **my:tag2**. What must you do to make this work?

- ❑ A. The instance of **Tag1Handler** must use the **getChildren** method in order to retrieve the collection of child tag instances. The instance of **Tag1Handler** will only be able to access the registered tags during the **doAfterBody** and **doEndTag** methods.
- ❑ B. The instance of **Tag2Handler** must use the **getParent** method in order to register itself with the instance of **Tag1Handler**. The instance of **Tag1Handler** will only be able to access the registered tags during the **doAfterBody** and **doEndTag** methods.
- ❑ C. The instance of **Tag1Handler** must use the **getChildren** method in order to retrieve the collection of child tag instances. The instance of **Tag1Handler** will be able to access the registered tags in any of the tag event methods, but NOT in the attribute setter methods.
- ❑ D. The instance of **Tag2Handler** must use the **getParent** method in order to register itself with the instance of **Tag1Handler**. The instance of **Tag1Handler** will be able to access the registered tags in any of the tag event methods, but NOT in the attribute setter methods.

43 Given that a deployment descriptor has only one security role, defined as:

```
21. <security-role>
22.    <role-name>Member</role-name>
23. </security-role>
```

Which are valid **<auth-constraint>** elements that will allow users to access resources constrained by the security role declared? (Choose all that apply.)

- ❑ A. **<auth-constraint/>**
- ❑ B. **<auth-constriant>*</auth-constraint>**
- ❑ C. **<auth-constraint>Member</auth-constraint>**
- ❑ D. **<auth-constraint>MEMBER</auth-constraint>**
- ❑ E. **<auth-constraint>"Member"</auth-constraint>**

44 Given this list of features:

- might create stale data
- can increase the complexity of code having to deal with concurrency issues

Which design pattern is being described?

- ❑ A. Transfer Object
- ❑ B. Service Locator
- ❑ C. Front Controller
- ❑ D. Business Delegate
- ❑ E. Intercepting Filter
- ❑ F. Model-View-Controller

45 Where are servlet classes located inside a Web Application Archive (WAR) file?

- ❑ A. Only in **/WEB-INF/classes**.
- ❑ B. Only in a JAR file in **/WEB-INF/lib**.
- ❑ C. Either in a JAR file in **/WEB-INF/lib** or under **/WEB-INF/classes**.
- ❑ D. At the top level of the directory tree inside the WAR so that the web container can easily find them upon deployment.

46 Which code snippets properly map the **com.example.web.BeerSelect** servlet to the **/SelectBeer.do** URL? (Choose all that apply.)

- ❑ A.
```
<servlet-map>
  <servlet-class>com.example.web.BeerSelect</servlet-class>
  <url-pattern>/SelectBeer.do</url-pattern>
<servlet-map>
```
- ❑ B.
```
<servlet>
  <servlet-mapping>
    <servlet-class>com.example.web.BeerSelect</servlet-class>
    <url-pattern>/SelectBeer.do</url-pattern>
  </servlet-mapping>
</servlet>
```
- ❑ C.
```
<servlet-mapping>
  <servlet-name>com.example.web.BeerSelect</servlet-name>
  <url-pattern>/SelectBeer.do</url-pattern>
</servlet-mapping>
```
- ❑ D.
```
<servlet>
  <servlet-name>BeerServlet</servlet-name>
  <servlet-class>com.example.web.BeerSelect</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>BeerServlet</servlet-name>
  <url-pattern>/SelectBeer.do</url-pattern>
</servlet-mapping>
```

47 Which statements about **HttpSession** objects are true? (Choose all that apply.)

- ❑ A. A session may become invalid due to inactivity.
- ❑ B. A new session is created each time a user makes a request.
- ❑ C. A session may become invalid as a result of a specific call by a servlet.
- ❑ D. Multiple requests made from the same browser may have access to the same session object.
- ❑ E. A user who accesses the same web application from two browser windows is guaranteed to have two distinct session objects.

48 Given a partial deployment descriptor:

```
11.   <servlet>
12.     <servlet-name>ServletIWantToListenTo</servlet-name>
13.     <servlet-class>com.example.MyServlet</servlet-class>
14.   </servlet>
15.   <listener>
16.     <listener-class>com.example.ListenerA</listener-class>
17.   </listener>
18.   <listener>
19.     <listener-class>com.example.ListenerB</listener-class>
20.     <listener-type>Session</listener-type>
21.   </listener>
22.   <listener>
23.     <listener-class>com.example.ListenerC</listener-class>
24.     <description>A session listener.</description>
25.   </listener>
26.   <listener>
27.     <listener-class>com.example.ReqListener</listener-class>
28.     <servlet-name>ServletIWantToListenTo</servlet-name>
29.   </listener>
```

Which are valid **listener** elements (identify each listener by the line number it starts on)? (Choose one.)

- ❑ A. Only 15.
- ❑ B. Only 18.
- ❑ C. Only 26.
- ❑ D. Both 15 and 22.
- ❑ E. Both 18 and 26.
- ❑ F. 15, 22 and 26.
- ❑ G. All four are valid.

49 Which statements concerning **/META-INF** are true? (Choose all that apply.)

- ❑ A. This directory is optional when creating a WAR file.
- ❑ B. The contents of this directory can be served directly to clients only if HTTPS is activated.
- ❑ C. Servlets can access the contents of the **/META-INF** directory via methods in the **ServletContext** class.
- ❑ D. Servlets can access the contents of the **/META-INF** directory via methods in the **ServletConfig** class.

50 Which security mechanisms can be applied to specific resources by specifying URL patterns in the deployment descriptor? (Choose all that apply.)

- ❑ A. authorization
- ❑ B. data integrity
- ❑ C. authentication
- ❑ D. confidentiality
- ❑ E. form-based login

51 Your company is in the process of integrating several different back office applications and creating a single web UI to present the entire back office suite to your clients. The design of the front end will be finished long before the design of the back end. Although the details of the back end design are still very rough, you have enough information to create some temporary back end 'stubs' to use to test the UI.

Which design pattern can be used to minimize the overhead of modifying the UI once the back end is complete?

- ❑ A. Transfer Object
- ❑ B. Front Controller
- ❑ C. Business Delegate
- ❑ D. Intercepting Filter
- ❑ E. Model-View-Controller

52 Given:

`fc` is of type **`FilterChain`** and
`req` and **`resp`** are request and response objects.

Which line of code in a class implementing **`Filter`** will invoke the target servlet if there is only one filter in the chain?

- ❑ A. `fc.chain(req, resp);`
- ❑ B. `fc.doFilter(req, resp);`
- ❑ C. `fc.doForward(req, resp);`
- ❑ D. `req.chain(req, resp, fc);`
- ❑ E. `req.doFilter(req, resp, fc);`
- ❑ F. `req.doForward(req, resp, fc);`

53 What type of listener could be used to log the user name of a user at the time that she logs into a system? (Choose all that apply.)

- ❑ A. **HttpSessionListener**
- ❑ B. **ServletContextListener**
- ❑ C. **HttpSessionAttributeListener**
- ❑ D. **ServletContextAttributeListener**

54 Given a tag library descriptor located at **/mywebapp/WEB-INF/tlds/mytags.tld**, which would be the correct **taglib** directive? Assume **mywebapp** is the web application root and that there are no **<taglib>** tags in the deployment descriptor.

- ❑ A.
```
<%@ taglib uri="/mytags.tld"
    prefix="my" %>
```
- ❑ B.
```
<%@ taglib uri="/tlds/mytags.tld"
    prefix="my" %>
```
- ❑ C.
```
<%@ taglib uri="/WEB-INF/tlds/mytags.tld"
    prefix="my" %>
```
- ❑ D.
```
<%@ taglib uri="/mywebapp/WEB-INF/tlds/mytags.tld"
    prefix="my" %>
```
- ❑ E. **/mywebapp/WEB-INF/tlds/mytags.tld** is NOT a valid location for a tag library descriptor, so none of these will work.

55 Given:

```
11. public class ServletX extends HttpServlet {
12.   public void doGet(HttpServletRequest req,
13.                     HttpServletResponse res)
14.           throws IOException, ServletException {
15.     req.getSession().setAttribute("key", new X());
16.     req.getSession().setAttribute("key", new X());
17.     req.getSession().setAttribute("key", "x");
18.     req.getSession().removeAttribute("key");
19.   }
20. }
```

and given a listener class:

```
11. public class X implements HttpSessionBindingListener {
12.   public void valueBound(HttpSessionBindingEvent event) {
13.     System.out.print("B");
14.   }
15.   public void valueUnbound(HttpSessionBindingEvent event) {
16.     System.out.print("UB");
17.   }
18. }
```

Which logging output would be generated by an invocation of the **doGet** method?

- ❑ A. **UBUBUB**
- ❑ B. **BBUBUB**
- ❑ C. **BBUBUBB**
- ❑ D. **BUBBUBB**
- ❑ E. **BBUBUBBUB**
- ❑ F. **BBUBBUBBUB**

56 Given:

```
10. public void doGet(HttpServletRequest req,
11.                   HttpServletResponse res)
12.         throws IOException, ServletException {
13.     RequestDispatcher rd1
14.       = getServletContext().getRequestDispatcher("/xyz");
15.     RequestDispatcher rd2
16.       = req.getRequestDispatcher("/xyz");
17.     RequestDispatcher rd3
18.       = getServletContext().getRequestDispatcher("xyz");
19.     RequestDispatcher rd4
20.       = req.getRequestDispatcher("xyz");
21. }
```

Which statements are true? (Choose all that apply.)

- ❑ A. **rd3** will never map to a servlet since the given path does NOT begin with **/**.
- ❑ B. **rd4** will never map to a servlet since the given path does NOT begin with **/**.
- ❑ C. **rd2.forward(req, res)** and **rd4.forward(req, res)** may forward to the same resource.
- ❑ D. **rd1.forward(req, res)** and **rd2.forward(req, res)** would always forward to the same resource.
- ❑ E. **rd3.forward(req, res)** and **rd4.forward(req, res)** would always forward to the same resource.

57 Which JSTL tag performs URL rewriting?

- ❑ A. **link**
- ❑ B. **aHref**
- ❑ C. **import**
- ❑ D. **url**

58 Given:

```
11. public void doGet(HttpServletRequest req,
12.                       HttpServletResponse res)
13.         throws IOException, ServletException {
14.     String url = res.encodeRedirectURL("/redirectme");
15.     boolean test = "/redirectme".equals(url);
16.     res.sendRedirect(url);
17. }
```

Which statements are true? (Choose all that apply.)

- ❑ A. After line 15, **test** will always be **true**.
- ❑ B. After line 15, **test** will always be **false**.
- ❑ C. Line 14 demonstrates a session management mechanism called URL rewriting.
- ❑ D. After line 15, **test** could be either **true** ot **false**.
- ❑ E. The **encodeURL** method should have been used instead of the **encodeRedirectURL** method in line 14.
- ❑ F. The **encodeRedirectURL** method shown in line 14 should only be used when clients have disabled cookies.

59 What happens when a container migrates a session from one VM to another?

- ❑ A. **sessionWillPassivate()** will be called on any objects that implement the **HttpSessionActivationListener** and are currently bound to the session.
- ❑ B. **sessionWillPassivate()** will be called on any objects that implement the **HttpSessionPassivationListener** and are currently bound to the session.
- ❑ C. **sessionWillPassivate()** will be called on any objects that implement the **HttpSessionListener** interface.
- ❑ D. **sessionWillPassivate()** will be called on any objects that implement the **HttpSessionBindingListener** and are currently bound to the session.

60 Given an existing class:

```
 1. package com.example;
 2. public class MyFunctions {
 3.    private int x;
 4.    public MyFunctions()
 5.       { x = 0; }
 6.    public int modX(int y)
 7.       { return (y mod x); }
 8.    public static int funct(int x, int y)
 9.       { return (2*x + y - 5); }
10. }
```

Which are true about EL functions? (Choose all that apply.)

- ❑ A. The **MyFunctions** class may NOT be used by EL because it has NOT been declared **final**.
- ❑ B. The **funct** method may be used by EL because it has been declared static.
- ❑ C. The **funct** method may NOT be use by EL because the calling arguments and return value must be object references.
- ❑ D. The **modX** method may be used by EL because it is an instance method.
- ❑ E. The **MyFunctions** class may be used by EL even though it has a public constructor.

61 Which statements about ignoring EL in your JSPs are true? (Choose all that apply.)

- ❑ A. You can instruct the container to ignore EL in your JSPs by using the **<el-ignored>** tag in the DD.
- ❑ B. You can instruct the container to ignore EL in your JSPs by using the **<is-el-ignored>** tag in the DD.
- ❑ C. You can instruct the container to ignore EL in a JSP by using the **elIgnored** attribute of the **page** directive.
- ❑ D. When using the DD to instruct the container to ignore EL, you can specify which JSPs to constrain.
- ❑ E. You CANNOT constrain both scripting and EL in the same JSP.

62 You have purchased a purchase order web application that uses programmatic authorization that uses security roles that are not used in your organization.

Which deployment descriptor element must you use to make this webapp work in your organization?

- ❑ A. `<login-config>`
- ❑ B. `<security-role>`
- ❑ C. `<security-role-ref>`
- ❑ D. `<security-constraint>`

63 Given:

```
1. <%@ taglib uri="http://www.mycompany.com/mytags"
               prefix="mytags" %>
2. <mytags:foo bar="abc" />
3. <mytags:forEach><mytags:doSomething /></mytags:forEach>
4. <jsp:setProperty name="x" property="a" value="b" />
5. <c:out value="hello" />
```

Assuming this is a complete JSP, which is true?

(For options E and F, ignore the fact that an error in one line might keep a subsequent line from being reached)

- ❑ A. Only line 2 will definitely generate an error.
- ❑ B. Only line 3 will definitely generate an error.
- ❑ C. Only line 4 will definitely generate an error.
- ❑ D. Only line 5 will definitely generate an error.
- ❑ E. Lines 4 and 5 will both definitely generate errors.
- ❑ F. Lines 2, 3, 4 and 5 will all definitely generate errors.
- ❑ G. The entire JSP could execute without generating any errors.

64 Which authentication mechanism employs a base64 encoding scheme to protect user passwords?

- ❑ A. HTTP Basic Authentication
- ❑ B. Form Based Authentication
- ❑ C. HTTP Digest Authentication
- ❑ D. HTTPS Client Authentication

65 Which concepts are common to all four authentication mechanisms supported by J2EE 1.4 compliant web containers? (Choose all that apply.)

- ❑ A. passwords
- ❑ B. realm names
- ❑ C. generic error pages
- ❑ D. secured web resources
- ❑ E. automatic SSL support
- ❑ F. target server authentication

66 How are cookies used to support session management?

- ❑ A. A cookie is created for each attribute stored in the session.
- ❑ B. A single cookie is created to hold an ID that uniquely identifies a session.
- ❑ C. A single cookie is created to hold the serialized **`HttpSession`** object.
- ❑ D. The session ID is encoded as a path parameter in the URL string called **`jsessionid`**.
- ❑ E. Cookies CANNOT be used to support session management because it is possible for a user to disable cookies in their browser.

67 You are developing a web application for an organization that needs to display the results of database searches to several different types of clients, including browsers, PDAs, and kiosks. The application will have to examine the request to determine which type of client has initiated it, and then dispatch the request to the proper component.

Which J2EE design pattern is designed for this type of application?

- ❑ A. Transfer Object
- ❑ B. Service Locator
- ❑ C. Model-View-Controller
- ❑ D. Business Delegate
- ❑ E. Intercepting Filter

68 Which is true about the differences between the Classic and Simple tag models?

- ❑ A. A nested Classic tag is allowed to access its parent tag, but this is NOT supported in the Simple tag model.
- ❑ B. In the Classic model, you may gain access to the evaluated body content using the **BodyTag** interface. In the Simple model, you can invoke the body, but you CANNOT gain access to the content generated in the invocation.
- ❑ C. The Tag interface has two event methods (**doStartTag** and **doEndTag**), but the **SimpleTag** interface only has one event method (**doTag**).
- ❑ D. Both tag models support iteration. In the **SimpleTag.doTag** method, you can invoke the body within a Java-based iteration. In the Classic model, iteration may be supported by returning the **EVAL_BODY_AGAIN** constant from the **Tag.doEndTag** method.

69 Given this class:

```
1. package biz.mybiz;
2. public class BeanX {
3.     private String a,b,c;
4.     public BeanX() {a="A";b="B";c="C";}
5.     public void setA(String a) { this.a = a; }
6.     public void setB(String b) { this.b = b; }
7.     public void setC(String c) { this.c = c; }
8.     public String getAll() { return a+b+c; }
9. }
```

and the JSP:

```
1. <jsp:useBean id="x" class="biz.mybiz.BeanX" />
2. <jsp:setProperty name="x" property="*" />
3. <jsp:getProperty name="x" property="all" />
4. <jsp:setProperty name="x" property="a" param="b" />
5. <jsp:setProperty name="x" property="b" param="c" />
6. <jsp:setProperty name="x" property="c" param="a" />
7. <jsp:getProperty name="x" property="all" />
```

What will be generated by the JSP when invoked with the query string **a=X&b=Y&c=Z**?

- ❑ A. **ABC YZX**
- ❑ B. **XYZ XYZ**
- ❑ C. **ABC ABC**
- ❑ D. **YXZ YZX**
- ❑ E. **XYZ ZXY**
- ❑ F. **XYZ YZX**
- ❑ G. **nullnullnull YZX**

Final Exam Answers

1 This statement describes the potential benefits of a design pattern:

These components pre-process or post-process incoming requests and outgoing responses in a web application, and can be cleanly plugged into the application depending on whether the application requires the component's specialized task.

Which design pattern is being described?

(Core J2EE Patterns 2nd ed. pg. 145)

- ☐ A. Transfer Object
- ☑ B. Intercepting Filter
- ☐ C. Model-View-Controller
- ☐ D. Business Delegate
- ☐ E. Front Controller

–One of the most powerful features of Intercepting Filter is that filters can be chained together in different sequences and added and subtracted from an application declaratively.

2 Which are true about the **`jsp:useBean`** standard action? (Choose all that apply.)

(JSP v2.0 section 5.1)

- ☐ A. The **`name`** attribute is mandatory.

–Option A is invalid because there is no name attribute in this standard action (pg 1–104).

- ☑ B. The **`scope`** attribute defaults to the page scope.
- ☐ C. The **`type`** and **`class`** attributes must NOT be used together.

–Option C is invalid because type and class may be used together (pg 1–103).

- ☐ D. The **`type`** attribute is ONLY used when the bean already exists in one of the four JSP scopes.

–Option D is invalid because the type attribute may be used in conjunction with the class attribute when creating a new object.

- ☑ E. The **`jsp:useBean`** standard action may be used to declare a scripting variable that may be used in scriptlets, such as **`<% myBean.method(); %>`**

–Option E is correct, because the bean name specified in the id attribute creates a local variable in the JSP's _jspService method.

3 Given this partial deployment descriptor:

(Servlet v2.4 pg. 135)

```
12.   <context-param>
13.     <param-name>email</param-name>
14.     <param-value>foo@bar.com</param-value>
15.   </context-param>
16.   <servlet>
17.     <servlet-name>a</servlet-name>
18.     <servlet-class>com.bar.Foo</servlet-class>
19.     <init-param>
20.       <param-name>email</param-name>
21.       <param-value>baz@bar.com</param-value>
22.     </init-param>
23.   </servlet>
```

And, assuming **scfg** is a **ServletConfig** object and **sctx** is a **ServletContext** object, which statement is true?

- ☐ A. **sctx.getInitParameter("email")** will return **baz@bar.com**.

–Option A is invalid because this call would return the servlet context initialization parameter (foo@bar.com).

- ☐ B. **scfg.getInitParameter("email")** will return **foo@bar.com**.

–Option B is invalid because this call would return the servlet-specific value baz@bar.com.

- ☐ C. An error will occur because the **email** parameter is defined twice.

–Option C is invalid because there are no naming restrictions between servlet context parameter names and servlet parameter names.

- ☐ D. **scfg.getServletContext().getInitParameter("email")** will return **baz@bar.com**.

–Option D is invalid because this call would return the servlet context initialization parameter (foo@bar.com).

- ☑ E. **scfg.getServletContext().getInitParameter("email")** will return **foo@bar.com**.

- ☐ F. An error will occur because servlet context initialization parameters should be defined using **init-param**, NOT **context-param**.

–Option F is invalid because servlet context initialization parameters are defined using <context-param>.

4 Given: (JSP v2.0 pg. 1-182)

```
public class DoubleTag extends SimpleTagSupport {
  private String data;
  public void setData(String d) { data = d; }
  public void doTag() throws JspException, IOException {
    getJspContext().getOut().print(data + data);
  }
}
```

Which is an equivalent tag file?

- ☐ A. `${param.data}${param.data}`

  – Option A is invalid because it would print the parameter named data, not the tag attribute data.

- ☑ B.
```
<%@ attribute name="data" %>
${data}${data}
```

- ☐ C.
```
<%@ variable name-given="data" %>
${data}${data}
```

  – Option C is invalid because the attribute directive should be used, not the variable directive.

- ☐ D.
```
<%@ attribute name="data" %>
<% pageContext.getOut().print(data + data); %>
```

  – Option D is invalid because the JSP variable pageContext is not available here. However, replacing pageContext with getJspContext() would work here instead.

- ☐ E.
```
<%@ variable name-given="data" %>
<% pageContext.getOut().print(data + data); %>
```

  – Option E is invalid for the reasons listed for Options C and D.

5 (API)

Given a session object **sess** with an attribute named **myAttr**, and an **HttpSessionBindingEvent** object **bind** bound to **sess**.

Which will retrieve **myAttr**? (Choose all that apply.)

- ☐ A. `long myAttr = sess.getAttribute("myAttr");`
- ☑ B. `Object o = sess.getAttribute("myAttr");`
- ☐ C. `String s = sess.getAttribute("myAttr");`
- ☐ D. `Object o = bind.getAttribute("myAttr");`
- ☑ E. `Object o = bind.getSession().getAttribute("myAttr");`

–Options A and C are invalid because getAttribute returns an Object.

–Option D is invalid because the event class has no getAttribute method.

6 Which about JSP Documents (XML-based documents) are true? (Choose all that apply.)

(JSP v2.0 section 6)

- ☐ A. A JSP document must have a **<jsp:root>** as its top element.

–Option A is invalid because the <jsp:root> elements is optional in JSP v2.0 (section 6.2.2).

- ☐ B. The following would be valid as the first line of a JSP document: **<jsp:root xmlns:uri="http://java.sun.com/JSP/Page" version="2.0">**.

–Option B is invalid because it does not introduce the prefix jsp used in the <jsp:root> element (it introduces the prefix uri).

- ☑ C. In a JSP document, page directives are defined using **<jsp:directive.page>** instead of **<%@ %>**.

- ☑ D. The **<c:forEach>** tag CANNOT be used unless the **c** prefix has been introduced through a namespace attribute.

- ☐ E. Both the standard **<%! %>** syntax as well as **<jsp:declaration>** may be used to declare variables in a JSP document.

–Option E is invalid because only the <jsp:declaration> syntax is valid in a JSP document.

7 Which statements about EL implicit objects are true? (Choose all that apply.)

(JSP v2.0 section 2.2.3)

- ☑ A. **${param.name}** produces the value of the parameter **name**.

- ☐ B. **${init.foo}** produces the value of the context initialization parameter named **foo**.

–Option B is invalid because the implicit object used for initialization parameters is initParam not init.

- ☑ C. The implicit object **param** is a **Map** that maps parameter names to single String parameter values.

- ☑ D. **pageContext**, **param**, **header**, and **cookie** are all implicit objects available to EL expressions in JSP pages.

- ☐ E. **page**, **request**, **session**, and **application** are implicit objects that map attribute names to values in the corresponding scopes.

–Option E is invalid because each of the implicit object names need the word Scope appended to the (e.g. sessionScope).

8 Given this JSP snippet:

(JSP v2.0 section 1.5)

```
10. <!--X-->
11. <%=A.b()%>
12. <!--<%=A.b()%>-->
13. <%--Y--%>
```

❑ Assume that a call to **A.b()** is valid and returns the text **test**. Ignoring whitespace in the generated response, which represents the HTML this JSP would produce?

A. **<!--X-->test<!--<%=A.b()%>-->**

–Options A and D are invalid because the expression A.b() is evaluated, despite being part of a comment.

☑ B. **<!--X-->test<!--test-->**

❑ C. **test**

–Option C is invalid because simple HTML comments are not removed.

❑ D. **<!--X-->test<!--<%=A.b()%>--> <%--Y--%>**

–Option D is also invalid because the JSP comments are removed from the page.

❑ E. **test<%--Y--%>**

–Option E is invalid for the reasons described for Options C and D.

❑ F. The generated HTML will be blank.

–Option F is invalid for many of the reasons described above.

9 Which are true about tag libraries in web applications? (Choose all that apply.)

(JSP v2.0 section 7.3.1)

❑ A. A TLD file must exist in the **WEB-INF/tlds/** directory.

–Option A is invalid because the statement is too strong; TLDs can exist in other WEB-INF subdirectories.

☑ B. A TLD file may exist in any subdirectory of **WEB-INF**.

❑ C. A TLD file may exist in the **WEB-INF** directory in a JAR file.

–Option C is invalid because a JAR file need not have a WEB-INF directory.

☑ D. A TLD file may exist in the **META-INF** directory in a JAR file.

❑ E. A TLD file in a JAR file must be located at **META-INF/taglib.tld**.

–Option E is invalid because the statement is too strong; more than one TLD file can exist in a single JAR with different file names.

10 You store the SQL source files for web-database work in a web application's top-level **sql** directory, but you do NOT want to make this directory accessible to HTTP requests. (Servlet v2.4 pg. 70)

How do you configure the web application to forbid requests to this directory? (Choose all that apply.)

- ☐ A. Protect the server with a firewall.
- ☐ B. Specify the directory with a **<security-role> element in** deployment descriptor.
- ☑ C. Move the directory beneath **WEB-INF**, the contents of which are NOT accessible to application users.
- ☑ D. Create a **<security-constraint>** element in the DD to prevent access to the **sql** directory.

–Option A is invalid because the firewall would not affect access to the /sql/* components in the webapp file structure.

–Option D could be used to restrict directory access using the <security-constraint> element's child element <web-resource-collection>, for instance.

11 Given: (JSP v2.0 section 2.3.4, 2.3.5, and 2.2.3)

```
11. <% java.util.Map map = new java.util.HashMap();
12.    map.put("a", "1");
13.    map.put("b", "2");
14.    map.put("c", "3");
15.    request.setAttribute("map", map);
16.    request.setAttribute("map-index", "b");
17. %>
18. <%-- insert code here --%>
```

Which , inserted at line 18, are valid and evaluate to **2**? (Choose all that apply.)

- ☑ A. **${map.b}**
- ☐ B. **${map[b]}**
- ☐ C. **${map.map-index}**
- ☐ D. **${map[map-index]}**
- ☐ E. **${map['map-index']}**
- ☑ F. **${map[requestScope['map-index']]}**

–Option A is correct because map.b is equivalent to map['b'].

–Option B is invalid because 'b' is not a defined attribute.

–Option C is invalid because this expression is really ${map.map – index} --a subtraction-- which is invalid no matter how you look at it.

–Option D is invalid because this expression is really ${map[map – index]} which is invalid for the same reasons as in Option C.

–Option E is invalid because there is no map key called 'map-index'.

12 Given this tag handler class excerpt:

(JSP v2.0 sections 7.3 and 13)

```
11. public class WorthlessTag extends TagSupport {
12.   private String x;
13.   public void setX(String x) { this.x = x; }
14.   public int doStartTag() throws JspException {
15.     try { pageContext.getOut().print(x); }
16.     catch (IOException e) { }
17.     if ("x".equals(x))
18.       return SKIP_BODY;
19.     else
20.       return EVAL_BODY_INCLUDE;
21.   }
22.   public int doEndTag() throws JspException {
23.     try { pageContext.getOut().print("E"); }
24.     catch (IOException e) { }
25.     if ("y".equals(x))
26.       return SKIP_PAGE;
27.     else
28.       return EVAL_PAGE;
29.   }
30. }
```

and given this TLD excerpt:

```
21. <tag>
22.     <name>worthless</name>
23.     <tag-class>com.mycom.WorthlessTag</tag-class>
24.     <body-content>empty</body-content>
25.     <attribute>
26.         <name>x</name>
27.         <required>true</required>
28.         <rtexprvalue>true</rtexprvalue>
29.     </attribute>
30. </tag>
```

(continued on next page.)

12, cont. and given this complete JSP page:

```
1. <%@ taglib uri="somevaliduri" prefix="w" %>
2. <w:worthless x="x" />
3. <w:worthless x="<%=Boolean.TRUE.toString()%>" />
4. <w:worthless x="y" />
5. <w:worthless x="z" />
```

What output does the JSP generate?

- ❑ A. **xE**

  – Option A is invalid because the SKIP_BODY return value does not keep the rest of the page from being evaluated.

- ❑ B. **x trueE yE**

  – Option B is invalid because the SKIP_BODY return value does not keep doEndTag() from being called.

- ☑ C. **xE trueE yE**

- ❑ D. **xE trueE yE zE**

  – Option D is invalid because the value SKIP_PAGE is returned from the third use of the tag, so the remainder of the page is ignored.

- ❑ E. **x <%=Boolean.TRUE.toString()%>E yE**

- ❑ F. **xE <%=Boolean.TRUE.toString()%>E yE**

  – Options E, F and G are invalid because the expression is properly evaluated before being passed to the setX() method.

- ❑ G. **xE <%=Boolean.TRUE.toString()%>E yE zE**

13 A user submits a form using an HTML page containing :

(Servlet v2.4 pg. 23)

```
<form action="/handler">
  <!-- form tags here -->
</form>
```

The URL pattern **/handler** is mapped to an HTTP servlet.

Which **HttpServlet** service method will the web container call in response to this form submit?

- ☐ A. **doHead**
- ☐ B. **doPost**
- ☐ C. **Get**
- ☑ D. **doGet**

–The default HTTP method for the form tag, if not specified by the tag's method attibute, is GET. The correct answer is option D, the HttpServlet's doGet() method.

---

14 Which statements concerning welcome files are true? (Choose all that apply.)

(Servlet v2.4 section 9.10)

- ☑ A. They can be declared in the DD.
- ☑ B. They can be used to respond to 'partial requests'.
- ☐ C. If multiple welcome files are declared for a web app, their ordering in the DD is NOT meaningful.
- ☑ D. J2EE 1.4 compliant containers are required to match partial requests to URLs in the welcome file list using a specified algorithm.
- ☐ E. If a J2EE 1.4 compliant container receives a partial request for which it CANNOT find a match, it must return an HTTP 404 error code.

–Option C is invalid because the list of welcome files is searched by the container in the order declared in the DD.

–Option E is invalid because while a given container might return a 404, it is not required to do so.

---

15 Once a session has been invalidated, which **HttpSession** methods can be called on that session without throwing an **IllegalStateException**? (Choose all that apply.)

(API)

- ☐ A. **invalidate**
- ☐ B. **getAttribute**
- ☐ C. **setAttribute**
- ☑ D. **getServletContext**
- ☐ E. **getAttributeNames**

–Since the ServletContext survives, getServletContext can be called successfully on an invalid session.

16 Which statements about the **taglib** directive are true? (Choose all that apply.)

(JSP v2.0 section 1.10.2)

- [x] A. A **taglib** directive always identifies a tag prefix that will distinguish usage of actions in the library.
- [ ] B. **<% taglib uri="http://www.mytags.com/mytags" prefix="mytags" %> is an example of a valid taglib** directive.
- [ ] C. Every **taglib** directive must specify a value for the **uri** attribute.
- [x] D. Every **taglib** directive must specify a non-empty value for the **prefix** attribute.
- [x] E. There are three attributes defined for the **taglib** directive: **uri**, **tagdir**, and **prefix**.

–Option B is invalid because directives must begin with <%@.

–Option C is invalid because the uri attribute is not required as long as the tagdir attribute is included instead.

17 Which statements about making a servlet's **doGet()** method **synchronized** are true? (Choose all that apply.)

(Servlet v2.4 section 2.3.3.1)

- [ ] A. It will make access to **ServletRequest** attributes thread-safe.
- [x] B. It will NOT make access to **HttpSession** attributes thread-safe.
- [x] C. It may have a negative performance impact because the servlet will only be able to process one request at a time.
- [ ] D. It is necessary if the method will be using **HttpSession.getAttribute() or HttpSession.setAttribute()**.
- [ ] E. It is necessary if the method will be using **ServletContext.getAttribute() or ServletContext.setAttribute()**.

–Option A is invalid because it doesn't make request attributes any more thread-safe than they already are.

–Options D and E are invalid because it does nothing to help make these attribute scopes more thread-safe since other servlets could access them concurrently despite the synchronization.

18 Which are valid EL implicit variables? (Choose all that apply.)

(JSP v2.0 section 2.2.3)

- [ ] A. **out**
- [ ] B. **request**
- [ ] C. **response**
- [x] D. **pageContext**
- [ ] E. **contextParam**

–Options A, B, and C are JSP implicit variables, but not in EL.

–Option E is a tricky one because context params are available in EL, but the variable is called initParam.

19 Given the following URL: (API)

```
http://www.example.com/userConfig/searchByName.do
?first=Bruce&middle=W&last=Perry
```

Which servlet code fragment from a service method, for example **`doPost()`**, will retrieve the values of all of the query string parameters?

❑ A.
```
String value
  = request.getParameter("Bruce");
```

❑ B.
```
String value
  = getServletContext().getInitParameter("first");
```

❑ C.
```
String value
  = getServletConfig().getInitParameter("first")
```

☑ D.
```
java.util.Enumeration enum
  = request.getParameterNames();
while (enum.hasMoreElements()) {
  String name = (String) enum.nextElement();
  String value = request.getParameter(name);
}
```

–Option D stores all parameter names in a java.util.Enumeration, then gets each value by calling request.getParameter().

❑ E.
```
java.util.Enumeration enum
  = request.getParameterNames();
while (enum.hasMoreElements()) {
  String value = (String) enum.nextElement();
}
```

NOTE: You can also use the getParameterMap() method on the request to access all querystring values.

---

20 Which are true about EL operators? (JSP v2.0 section 2.8)
(Choose all that apply.)

☑ A. The logical operators treat a **`null`** value as **`false`**. –Option A is correct (2.8.5).

❑ B. The arithmetic operators treat a **`null`** value as **`Double.NaN`** (not a number).

–Option B is invalid because a null is coerced to a zero (2.8.3).

❑ C. Divide by zero, **`${x div 0}`**, throws a runtime exception.

–Option C is invalid because division by zero returns infinity.

☑ D. Strings in EL expressions are automatically converted into the appropriate numeric or boolean values.

❑ E. A **`NullPointerException`** is thrown when a **`null`** is encountered in an arithmetic EL expression.

–Option E is invalid because null is coerced to a zero (2.8.3).

21 Given the partial TLD:

(JSP v2.0 pgs 3-21 and 3-30)

```
11.   <tag>
12.     <name>getTitle</name>
13.     <tag-class>com.example.taglib.GetTitleTagHandler</tag-class>
14.     <body-content>empty</body-content>
15.     <attribute>
16.       <name>story</name>
17.       <required>false</required>
18.     </attribute>
19.   </tag>
20.   <tag>
21.     <name>printMessage</name>
22.     <tag-class>com.example.taglib.PrintMessageTagHandler</tag-class>
23.     <body-content>JSP</body-content>
24.     <attribute>
25.       <name>section</name>
26.       <required>true</required>
27.     </attribute>
28.   </tag>
```

Which are valid invocations of these tags within a JSP? (Choose all that apply)

- ☐ A.
```
<my:getTitle>
  <my:printMessage />
</my:getTitle>
```
- ☑ B.
```
<my:printMessage section="47">
  <my:getTitle />
</my:printMessage>
```
- ☐ C.
```
<my:getTitle story="">
  <my:printMessage section="47" />
</my:getTitle>
```
- ☑ D.
```
<my:printMessage section="47">
  <my:getTitle story="Shakespear_RJ"></my:getTitle>
</my:printMessage>
```

-The getTitle tag must have an empty body, which knocks out options A and C.

The printMessage tag is required to use the section attribute, which also knocks out option A.

That leaves options B and D as valid uses of these tags.

---

22 Which JSP code would you use to include static content in a JSP? (Choose all that apply.)

(JSP v2.0 sections 1.10.3 and 5.4)

- [x] A. `<%@ include file="/segments/footer.html" %>`
- [ ] B. `<jsp:forward page="/segments/footer.html" />`
- [x] C. `<jsp:include page="/segments/footer.html" />`
- [ ] D.
```
RequestDispatcher dispatcher
  = request.getRequestDispatcher("/segments/footer.
html");
```

–Option A is correct because it uses an include directive, which includes the bytes of the referenced resource prior to the JSP's translation into a servlet.

---

23 Which statements about the deployment descriptor (DD) are true? (Choose all that apply.)

(Servlet v2.4 section 13)

- [ ] A. The DD must contain at least one **<context-param>** element.
- [x] B. The DD must be a well-formed XML file.
- [ ] C. The DD can be a text-based properties file or an XML file.
- [ ] D. You can leave out the XML form of the DD and provide a DD as a Java object.
- [x] E. The **<web-app>** element must be the parent element of all of the other DD elements.

–The deployment descriptor has to be well-formed XML and <web-app> is the parent element.

---

24 Which steps occur before **jspInit()** is called? (Choose all that apply.)

(JSP v2.0 section 1.1)

- [x] A. A class instantiation occurs.
- [x] B. A Java source file is compiled.
- [ ] C. The **_jspService()** method is called.
- [x] D. The JSP page is translated to source.
- [ ] E. The **jspCreate()** method is calleD.
- [x] F. The container supplies a **ServletConfig** reference.

–There is no jspCreate() method, and the _jspService() method is called after jspInit is called.

25 Given a Simple tag handler class:

(JSP v2.0 SimpleTagSupport API pg 2-86, JSP v2.0 PageContext API pg 2-30, and Servlet v2.4 HttpServletRequest API pg 242)

```
11. public class MyTagHandler
           extends SimpleTagSupport {
12.   public void doTag() throws JspException {
13.     try {
14.       // insert code 1
15.       // insert code 2
16.       // insert code 3
17.       JspWriter out = tagContext.getOut();
18.       out.print(requestURI);
19.     } catch (IOException ioe) {
20.       throw new JspException(ioe);
21.     }
22.   }
23. }
```

This item is testing two APIs. First, that a Simple tag handler must use the getJspContext (and cast it) to retrieve the PageContext object. Second, that the request object can only be retrieved from a PageContext and not a JspContext. Option D is the only valid combination of code to make the question of this item true.

Which, inserted at lines 14, 15, and 16, will print the request-URI to the response stream?

❑ A.
```
14. JspContext tagContext = pageContext;
15. ServletRequest request
        = (ServletRequest) tagContext.getRequest();
16. String requestURI = request.getRequestURI();
```

-Option A is invalid because there is no protected pageContext variable as there is for Classic tags.

❑ B.
```
14. PageContext tagContext = (PageContext) jspContext;
15. ServletRequest request
        = (ServletRequest) tagContext.getRequest();
16. String requestURI = request.getRequestURI();
```

-Option B is invalid because there is no jspContext protected variable.

❑ C.
```
14. JspContext tagContext = getJspContext();
15. HttpServletRequest request
        = (HttpServletRequest) tagContext.getRequest();
16. String requestURI = request.getRequestURI();
```

-Option C is invalid because you cannot access the request object from the JspContext API.

☑ D.
```
14. PageContext tagContext = (PageContext) getJspContext();
15. HttpServletRequest request
        = (HttpServletRequest) tagContext.getRequest();
16. String requestURI = request.getRequestURI();
```

26

Given the following scriptlet code:

(JSTL v1.1 section 5.1)

```
11. <% String cityParam = request.getParameter("city");
12.     if ( cityParam != null ) { %>
13.       City: <input type='text' name='city' value='<%= cityParam %>' />
14. <% } else { %>
15.       City: <input type='text' name='city' value='Paris' />
16. <% } %>
```

Which JSTL code snippet produces the same result?

❑ A.
```
<c:if test='${not empty param.city}'>
  City: <input type='text' name='city' value='${param.city}' />
<c:else/>
  City: <input type='text' name='city' value='Paris' />
</c:if>
```

❑ B.
```
<c:if test='${not empty param.city'>
 <c:then>
    City: <input type='text' name='city' value='${param.city}' />
  </c:then>
  <c:else>
    City: <input type='text' name='city' value='Paris' />
  </c:else>
</c:if>
```

❑ C.
```
<c:choose test='${not empty param.city}'>
  City: <input type='text' name='city' value='${param.city}' />
<c:otherwise/>
  City: <input type='text' name='city' value='Paris' />
</c:choose>
```

☑ D.
```
<c:choose>
  <c:when test='${not empty param.city}'>
    City: <input type='text' name='city' value='${param.city}' />
  </c:when>
  <c:otherwise>
    City: <input type='text' name='city' value='Paris' />
  </c:otherwise>
</c:choose>
```

–To mimic an if–then–else statement you need to use the choose/when/otherwise tags. Option D is the only valid usage pattern.

27 How would you redirect an HTTP request to another URL? (Choose all that apply)

(HttpServletResponse API)

- [x] A. `response.sendRedirect("/anotherUrl");`
- [ ] B. `response.encodeRedirectURL("/anotherUrl");`
- [x] C. `response.sendRedirect(response.encodeRedirectURL("/anotherUrl"));`
- [ ] D. `RequestDispatcher dispatcher = request.getRequestDispatcher("/anotherUrl"); dispatcher.forward(request,response);`
- [ ] E. `RequestDispatcher dispatcher = request.getRequestDispatcher("/anotherUrl"); dispatcher.redirect(request,response);`

–Option B is invalid because the encodeRedirectURL method only performs the URL rewriting, and not the actual redirection.

–Option D is invalid because a forward is server-side only, but a redirect must tell the browser to change URLs.

–Option E is invalid because there is no such method on a RequestDispatcher.

28 Given:

(JSP v2.0 pg. 1-49)

```
<%@ page isELIgnored="true" %>
```

Which statements are true? (Choose all that apply.)

- [x] A. This is an example of a directive.
- [ ] B. This is NOT an example of a directivE.
- [x] C. It will cause `${a.b}` to be ignored by the container.
- [ ] D. It will NOT cause `${a.b}` to be ignored by the container.
- [x] E. It will cause the EL expression in `<c:out value="${a.b}"/>` to be ignored by the container.
- [ ] F. It will NOT cause the EL expression in `<c:out value="${a.b}"/>` to be ignored by the container.

–Options D and F are invalid because the isELIgnored directive, when set to true, indicates that the container should ignore all EL expressions in this JSP.

29 Given a deployment descriptor with three valid **<security-constraint>** elements, all constraining web resource A. And, given that two of the **<security-constraint>** elements respective **<auth-constraint>** sub--elements are:

(Servlet v2.4 sec. 12.8.1)

<auth-constraint>Bob</auth-constraint>

and

<auth-constraint>Alice</auth-constraint>

And that the third **<security-constraint>** element has no **<auth-constraint>** sub-element.

Who can access resource A?

- ☐ A. no one
- ☑ B. anyone
- ☐ C. only Bob
- ☐ D. only Alice
- ☐ E. only Bob and Alice
- ☐ F. anyone but Bob or Alice

–Option B is correct. The existence of a <security–constraint> element with no <auth–constraint> element overrides all other <auth–constraint> elements that refer to that resource, granting access to everyone.

30 Given:

(JSP v2.0 section 2.6)

```
51. <function>
52.    <name>myfun</name>
53.    <function-class>com.example.MyFunctions</function-class>
54.    <function-signature>
55.       java.util.List getList(java.lang.String name)
56.    </function-signature>
57. </function>
```

Which is true about an invocation of this EL function mapping?

Assume that **pre** is correctly declared by a **taglib** directive.

–Option A is invalid because an EL function may return any object type.

- ☐ A. EL functions are NOT allowed to return collections.

–Option B is invalid because EL uses the <name> mapping, not the real method name.

- ☐ B. **${pre:getList("visitors")[0]}** is a valid invocation.

–Option C is correct because it uses the <name> mapping.

- ☑ C. **${pre:myfun("visitors")[0]}** is a valid invocation.

–Option D is invalid because you DO need the package information on all reference types, including classes in the java.lang package.

- ☐ D. The function signature is invalid because you do NOT need to specify the package information **java.lang** on the method parameter.

31 In an HTML page with a rich, graphical layout, how would you write the JSP standard action code to import a JSP segment that generates a menu that is parameterized by the user's role?

(JSP v2.0 section 5.4 and 5.6)

☑ A.
```
<jsp:include page="user-menu.jsp">
  <jsp:param name="userRole"
             value="${user.role}" />
</jsp:include>
```

☐ B.
```
<jsp:import page="user-menu.jsp">
  <jsp:param name="userRole"
             value="${user.role}" />
</jsp:import>
```
–Option B is invalid because there is no import standard action.

☐ C.
```
<jsp:include page="user-menu.jsp">
  <jsp:parameter name="userRole"
                 value="${user.role}" />
</jsp:include>
```
–Option C is invalid because there is no parameter standard action.

☐ D.
```
<jsp:import page="user-menu.jsp">
  <jsp:parameter name="userRole"
                 value="${user.role}" />
</jsp:import>
```
–Option D is invalid because there are no import and parameter standard actions.

☐ E. This CANNOT be done using a JSP standard action.

–Option E is invalid because this CAN be done using the include/param actions.

---

32 Given that **resp** is an **HttpServletResponse**, and no custom headers exist in this response before this snippet executes:

(API)

```
30.  resp.addHeader("myHeader", "foo");
31.  resp.addHeader("myHeader", "bar");
32.  resp.setHeader("myHeader", "baz");
33.  String [] s = resp.getHeaders("myHeader");
```

What is the value of **s[0]**?

☐ A. **foo**

☐ B. **bar**

☐ C. **baz**

☑ D. Compilation fails

☐ E. An exception is thrown at runtime

–Option D is correct. Compilation fails because there is no getHeaders() method in HttpServletResponse. Note that line 31 would add "bar" as an additional value, and line 32 would reset the value of "myHeader" to "baz".

---

33 Given a servlet that stores a customer bean in the session scope with the following code snippet:

(JSP v2.0 sections 5.1 and 5.3)

```
11.   public void doPost(HttpServletRequest req,
12.                       HttpServletResponse resp) {
13.     HttpSession session = req.getSession();
14.     com.example.Customer cust
15.       = new com.example.Customer();
16.     cust.setName(req.getParameter("full_name"));
17.     session.setAttribute("customer", cust);
18.     RequestDispatcher page
19.       = req.getRequestDispatcher("page.jsp");
20.     page.forward(req, resp);
21.   }
```

Which of these complete JSPs will print the customer's name? (Choose all that apply..)

☐ A.
```
1. <%= customer.getName() %>
```
–Option A is invalid because the customer variable has not yet been initialized.

☑ B.
```
1. <jsp:useBean id="customer"
2.              type="com.example.Customer"
3.              scope="session" />
4. <%= customer.getName() %>
```
–Option B is correct because the useBean tag initializes the customer variable.

☐ C.
```
1. <jsp:useBean id="customer"
2.              type="com.example.Customer"
3.              scope="session">
4.   <%= customer.getName() %>
5. </jsp:useBean>
```
–Option C is invalid because the body of the useBean tag will not be invoked because the bean already exists in the session scope.

☑ D.
```
1. <jsp:useBean id="customer"
2.              type="com.example.Customer"
3.              scope="session" />
4. <jsp:getProperty name="customer"
5.                  property="name" />
```
–Option D is correct because the getProperty tag prints the property.

☐ E.
```
1. <jsp:useBean id="customer"
2.              type="com.example.Customer"
3.              scope="session">
4.   <jsp:getProperty name="customer"
5.                    property="name" />
6. </jsp:useBean>
```
–Option E is invalid because the body of the useBean tag will not be invoked because the bean already exists in the session scope.

34 Which are valid elements in a DD? (Choose all that apply.)

(Servlet v2.4 section 6.2.5)

- ☐ A.
```
<filter>
  ...
  <dispatcher>ERROR</dispatcher>
</filter>
```
- ☐ B.
```
<filter>
  ...
  <dispatcher>CHAIN</dispatcher>
</filter>
```
- ☐ C.
```
<filter>
  ...
  <dispatcher>FORWARD</dispatcher>
</filter>
```
- ☑ D.
```
<filter-mapping>
  ...
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
```
- ☑ E.
```
<filter-mapping>
  ...
  <dispatcher>REQUEST</dispatcher>
</filter-mapping>
```
- ☐ F.
```
<filter-mapping>
  ...
  <dispatcher>RESPONSE</dispatcher>
</filter-mapping>
```

–Options A, B and C are invalid because the <dispatcher> element is a sub-element of <filter-mapping>, although ERROR is a valid value for the <dispatcher> element.

–Option F is invalid because the <dispatcher> is only applied to requests.

35 Given that **req** is an **HttpServletRequest**, which returns the names of all the parameters in the request? (Choose all that apply.)

(API)

- ☐ A. `Map names = req.getParameterNames();`
- ☐ B. `String [] names = req.getParameters();`
- ☐ C. `Enumeration names = req.getParameters();`
- ☐ D. `String [] names = req.getParameterNames();`
- ☑ E. `Enumeration names = req.getParameterNames();`

–Option E specifies the correct method name and the correct return type.

36 Which of the following are legal **<error-page>** elements? (Choose all that apply.)

(Servet v2.4 pgs. 142 & 306)

☑ A.
```
<error-page>
  <exception-type>java.lang.Throwable</exception-type>
  <location>/error/generic-error.jsp</location>
</error-page>
```

☑ B.
```
<error-page>
  <error-code>404</error-code>
  <location>/error/file-not-found.jsp</location>
</error-page>
```

☐ C.
```
<error-page>
  <error-code>404</error-code>
  <error-code>403</error-code>
  <location>/error/generic-error.jsp</location>
</error-page>
```

☐ D.
```
<error-page>
  <error-code>404</error-code>
  <location>/error/file-not-found.jsp</location>
  <location>/error/generic-error.jsp</location>
</error-page>
```

☐ E.
```
<error-page>
  <error-code>404</error-code>
  <exception-type>java.lang.Throwable</exception-type>
  <location>/error/generic-error.jsp</location>
</error-page>
```

–An <error-page> element can have either a single <error-code> element OR a single <exception-type> element, but not both. In addition, an <error-page> element must have a single <location> subelement.

37 Given that there exists a **HashMap** attribute called **preferences** in the session-scope.

(JSTL v1.1 section 4.3)

Which JSTL code structure will put an entry, **color**, into the map with the value of the color request parameter?

☑ A.
```
<c:set target="${sessionScope.preferences}"
       property="color" value="${param.color}" />
```

☐ B.
```
<c:put map="${sessionScope.preferences}"
property="color" value="${param.color}" />
```

☐ C.
```
<c:set scope="session" var="preferences"
 property="color" value="${param.color}" />
```

☐ D.
```
<c:put scope="session" map="preferences"
 property="color" value="${param.color}" />
```

–Option B is invalid because there is no put tag.

–Option C is invalid because the var and property attributes of the set tag are not a valid combination.

–Option D is invalid because there is no put tag and no map attribute.

Note: This is a simulated 'Drag and Drop' question, something like what you'll see on the exam:

(JSP v2.0 section 1.8.3)

38 Given the Implicit Objects listed on the left, and actual Java types listed on the right, match the Implicit Objects to their correct Java type:

out	Object
application	JspWriter
config	PageAttributes
page	Writer
	JspContext
	JspConfig
	System
	ServletConfig
	ServletContext

Answer:

```
out ----------> JspWriter
application --> ServletContext
config --------> ServletConfig
page ---------> Object
```

—These are the correct mappings from implicit object type to Java type.

39 Which Simple tag handler will iterate over the body content five times? (JSP v2.0 section 13.6)

☑ A.
```
public class MySimpleTag extends SimpleTagSupport {
  public void doTag() throws IOException, JspException {
    for ( int i=0; i<5; i++ ) {
      getJspBody().invoke(null);
    }
  }
)
```
– Option A is correct; iteration can be performed right in the doTag method.

☐ B.
```
public class MySimpleTag extends SimpleTagSupport {
  int count=0;
  public int doTag() throws IOException, JspException {
    getJspBody().invoke(null);
    count++;
    return ( (count<5) ? EVAL_BODY_AGAIN : SKIP_BODY );
  }
}
```
– Option B is invalid because the doTag method does not return an int flag as is done in the Classic tag model.

☐ C.
```
public class MySimpleTag extends SimpleTagSupport {
  int count=0;
  public int doStartTag() {
    return EVAL_BODY_INCLUDE;
  }
  public int doEndTag() {
    count++;
    return ( (count<5) ? EVAL_BODY_AGAIN : SKIP_BODY );
  }
}
```
– Options C and D are invalid because a Simple tag does not have the doStartTag, doAfterBody, or doEndTag methods, which are part of the Classic model.

☐ D.
```
public class MySimpleTag extends SimpleTagSupport {
  int count=0;
  public int doStartTag() {
    return EVAL_BODY_INCLUDE;
  }
  public int doAfterBody() {
    count++;
    return ( (count<5) ? EVAL_BODY_AGAIN : SKIP_BODY );
  }
}
```
BTW, Option D is a valid implementation of this tag behavior using the Classic model; except that MySimpleTag must extend TagSupport.

40 Which of the following statements about the servlet lifecycle are true? (Choose all that apply.)

(Servlet v2.4 section 2.3)

- ☐ A. The web container calls the **`init()`** and **`service()`** methods in response to each servlet request.

–Option A is invalid because the init() method is only called once after the servlet has been initialized.

- ☐ B. The web application developer uses an object that implements the **`java.servlet.Filter`** interface to instantiate one or more servlets.

–Option B is completely bogus because filters are not used this way.

- ☑ C. The web container calls the servlet's **`destroy()`** method when the servlet is destroyed or unloaded.

- ☑ D. The web container loads and instantiates a servlet class, then initializes the servlet by calling its **`init()`** method exactly once, passing into **`init()`** an object that implements the **`javax.servlet.ServletConfig`** interface that the container creates for the servlet.

–Option D shows the process by which a web container loads and initializes a servlet.

41 Which about web attributes are true? (Choose all that apply.)

(Servlet v2.4 general knowledge)

- ☐ A. No attributes are longer lived than session attributes.

–Option A is wrong because context attributes tend to be longer lived.

- ☑ B. In all scopes, attributes can be retrieved using a **`getAttribute()`** method.

- ☑ C. Context attributes survive a session time-out.

- ☐ D. Only session and context attributes can be retrieved in an enumeration.

–Option D is wrong because request attributes can also be retrieved in an enumeration.

- ☐ E. Data stored in both request and session objects is thread safe.

–Option E is wrong because, in the 'two–browser windows' scenario, session objects may not be thread safe.

42

Given a JSP page:

```
11. <my:tag1>
12.   <my:tag2>
13.     <%-- JSP content --%>
14.   </my:tag2>
15. </my:tag1>
```

(JSP v2.0 SimpleTagSupport API, JSP v2.0 TagSupport API, and JSP v2.0 Classic tag lifecycle pg 2–59)

The tag handler for **my:tag1** is **Tag1Handler** and extends TagSupport. The tag handler for **my:tag2** is **Tag2Handler** and extends SimpleTagSupport.

The tag handler for **my:tag1** must have access to the tag handler for **my:tag2**. What must you do to make this work?

- ❑ A. The instance of **Tag1Handler** must use the **getChildren** method in order to retrieve the collection of child tag instances. The instance of **Tag1Handler** will only be able to access the registered tags during the **doAfterBody** and **doEndTag** methods.

–TagSupport and SimpleTagSupport both have a getParent method, but there is no such method as getChildren. This fact eliminates Options A and C.

- ☑ B. The instance of **Tag2Handler** must use the **getParent** method in order to register itself with the instance of **Tag1Handler**. The instance of **Tag1Handler** will only be able to access the registered tags during the **doAfterBody** and **doEndTag** methods.

- ❑ C. The instance of **Tag1Handler** must use the **getChildren** method in order to retrieve the collection of child tag instances. The instance of **Tag1Handler** will be able to access the registered tags in any of the tag event methods, but NOT in the attribute setter methods.

- ❑ D. The instance of **Tag2Handler** must use the **getParent** method in order to register itself with the instance of **Tag1Handler**. The instance of **Tag1Handler** will be able to access the registered tags in any of the tag event methods, but NOT in the attribute setter methods.

–The child tags must be activated (via the attribute setters or tag event methods) in order to register with the parent. Therefore, the parent tag must have invoked the tag body at least once. Thus, only the doAfterBody and doEndTag methods will have access to the registered inner tags. This fact eliminates Option D.

43 Given that a deployment descriptor has only one security role, defined as: (Servlet v2.4 section 12.8)

```
21. <security-role>
22.    <role-name>Member</role-name>
23. </security-role>
```

Which are valid **<auth-constraint>** elements that will allow users to access resources constrained by the security role declared? (Choose all that apply.)

- ☐ A. **<auth-constraint/>**
- ☑ B. **<auth-constriant>*</auth-constraint>**
- ☑ C. **<auth-constraint>Member</auth-constraint>**
- ☐ D. **<auth-constraint>MEMBER</auth-constraint>**
- ☐ E. **<auth-constraint>"Member"</auth-constraint>**

–Options B and C are correct. Role names are case sensitive in the deployment descriptor, and an empty <auth-constraint> element signifies that no users can access the resource being requested.

44 Given this list of features: (Core J2EE Patterns 2nd ed., pg. 424)

- might create stale data
- can increase the complexity of code having to deal with concurrency issues

Which design pattern is being described?

- ☑ A. Transfer Object
- ☐ B. Service Locator
- ☐ C. Front Controller
- ☐ D. Business Delegate
- ☐ E. Intercepting Filter
- ☐ F. Model-View-Controller

–The creation of stale data is a common side effect whenever you decouple data from its remote source. Remote sources do not typically broadcast updates to 'subscribers' to their data.

45 Where are servlet classes located inside a Web Application Archive (WAR) file? (Servlet v2.4 section 9.5)

- ☐ A. Only in **/WEB-INF/classes**.
- ☐ B. Only in a JAR file in **/WEB-INF/lib**.
- ☑ C. Either in a JAR file in **/WEB-INF/lib** or under **/WEB-INF/classes**.
- ☐ D. At the top level of the directory tree inside the WAR so that the web container can easily find them upon deployment.

–Option C shows the correct options for storing servlet classes in a WAR.

46 Which code snippets properly map the **com.example.web.BeerSelect** servlet to the **/SelectBeer.do** URL? (Choose all that apply.)

(Servlet v2.4 section 13.5.1)

☐ A.
```
<servlet-map>
  <servlet-class>com.example.web.BeerSelect</servlet-class>
  <url-pattern>/SelectBeer.do</url-pattern>
<servlet-map>
```

☐ B.
```
<servlet>
  <servlet-mapping>
    <servlet-class>com.example.web.BeerSelect</servlet-class>
    <url-pattern>/SelectBeer.do</url-pattern>
  </servlet-mapping>
</servlet>
```

☐ C.
```
<servlet-mapping>
  <servlet-name>com.example.web.BeerSelect</servlet-name>
  <url-pattern>/SelectBeer.do</url-pattern>
</servlet-mapping>
```

☑ D.
```
<servlet>
  <servlet-name>BeerServlet</servlet-name>
  <servlet-class>com.example.web.BeerSelect</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>BeerServlet</servlet-name>
  <url-pattern>/SelectBeer.do</url-pattern>
</servlet-mapping>
```

–Option D is correct. The <servlet-name> element is used internally within the DD to tie the <servlet> and <servlet-mapping> elements to each other.

47 Which statements about **HttpSession** objects are true? (Choose all that apply.)

(Servlet v2.4 section 7)

☑ A. A session may become invalid due to inactivity.

☐ B. A new session is created each time a user makes a request.

–Option B is invalid because the purpose of a session is to span multiple requests.

☑ C. A session may become invalid as a result of a specific call by a servlet.

☑ D. Multiple requests made from the same browser may have access to the same session object.

☐ E. A user who accesses the same web application from two browser windows is guaranteed to have two distinct session objects.

–Option E is invalid because multiple browser windows will typically share a session.

48 Given a partial deployment descriptor:

(Servlet v2.4 pg. 139)

```
11.   <servlet>
12.     <servlet-name>ServletIWantToListenTo</servlet-name>
13.     <servlet-class>com.example.MyServlet</servlet-class>
14.   </servlet>
15.   <listener>
16.     <listener-class>com.example.ListenerA</listener-class>
17.   </listener>
18.   <listener>
19.     <listener-class>com.example.ListenerB</listener-class>
20.     <listener-type>Session</listener-type>
21.   </listener>
22.   <listener>
23.     <listener-class>com.example.ListenerC</listener-class>
24.     <description>A session listener.</description>
25.   </listener>
26.   <listener>
27.     <listener-class>com.example.ReqListener</listener-class>
28.     <servlet-name>ServletIWantToListenTo</servlet-name>
29.   </listener>
```

Which are valid **listener** elements (identify each listener by the line number it starts on)? (Choose one.)

- ❑ A. Only 15.
- ❑ B. Only 18.
- ❑ C. Only 26.
- ☑ D. Both 15 and 22.
- ❑ E. Both 18 and 26.
- ❑ F. 15, 22 and 26.
- ❑ G. All four are valid.

-Option A is invalid because servlets do not act as listeners.

-Options B, E and G are incorrect because there is no <listener-type> element.

-Options C, E, F and G are invalid because the <servlet-name> element is not applicable to the <listener> element.

49 Which statements concerning **/META-INF** are true? (Choose all that apply.)

(Servlet v2.4 section 9.6)

- ❑ A. This directory is optional when creating a WAR file.
- ❑ B. The contents of this directory can be served directly to clients only if HTTPS is activated.
- ☑ C. Servlets can access the contents of the **/META-INF** directory via methods in the **ServletContext** class.
- ❑ D. Servlets can access the contents of the **/META-INF** directory via methods in the **ServletConfig** class.

50 Which security mechanisms can be applied to specific resources by specifying URL patterns in the deployment descriptor? (Choose all that apply.)

(Servlet v2.4 section 12)

- [x] A. authorization
- [x] B. data integrity
- [ ] C. authentication
- [x] D. confidentiality
- [ ] E. form-based login

–Options A, B, and D are correct. The <security-constraint> element allows security to be mapped to specific URLs and within that element, the <user-data-constraint> element allows the deployer to declare a transport guarantee to provide data integrity and confidentiality.

51 Your company is in the process of integrating several different back office applications and creating a single web UI to present the entire back office suite to your clients. The design of the front end will be finished long before the design of the back end. Although the details of the back end design are still very rough, you have enough information to create some temporary back end 'stubs' to use to test the UI.

(Core J2EE Patterns 2nd ed., pg. 302)

Which design pattern can be used to minimize the overhead of modifying the UI once the back end is complete?

- [ ] A. Transfer Object
- [ ] B. Front Controller
- [x] C. Business Delegate
- [ ] D. Intercepting Filter
- [ ] E. Model-View-Controller

–The Business Delegate can be used when you want to minimize coupling between clients and business services. It is also appropriate when you need to hide implementation details or, in this case, partition them since they are temporary.

52 Given:

`fc` is of type **`FilterChain`** and
`req` and **`resp`** are request and response objects.

(Filter API)

Which line of code in a class implementing **`Filter`** will invoke the target servlet if there is only one filter in the chain?

- [ ] A. **`fc.chain(req, resp);`**
- [x] B. **`fc.doFilter(req, resp);`**
- [ ] C. **`fc.doForward(req, resp);`**
- [ ] D. **`req.chain(req, resp, fc);`**
- [ ] E. **`req.doFilter(req, resp, fc);`**
- [ ] F. **`req.doForward(req, resp, fc);`**

–Option B is the correct method call regardless of whether the target servlet is next in the chain.

53 What type of listener could be used to log the user name of a user at the time that she logs into a system? (Choose all that apply.)

(API)

- ☐ A. **`HttpSessionListener`**
- ☐ B. **`ServletContextListener`**
- ☑ C. **`HttpSessionAttributeListener`**
- ☐ D. **`ServletContextAttributeListener`**

–Option A is incorrect because the user name would not be known when the session is initially created. Since logging is desired at the time of the login, the listener's invalidation and timeout methods would not be helpful.

–Options B and D are incorrect because these listeners are used for servlet-context notifications.

54 Given a tag library descriptor located at **`/mywebapp/WEB-INF/tlds/mytags.tld`**, which would be the correct **`taglib`** directive? Assume **`mywebapp`** is the web application root and that there are no **`<taglib>`** tags in the deployment descriptor.

(JSP v2.0 section 7.3.6.3)

- ☐ A. `<%@ taglib uri="/mytags.tld" prefix="my" %>`
- ☐ B. `<%@ taglib uri="/tlds/mytags.tld" prefix="my" %>`
- ☑ C. `<%@ taglib uri="/WEB-INF/tlds/mytags.tld" prefix="my" %>`
- ☐ D. `<%@ taglib uri="/mywebapp/WEB-INF/tlds/mytags.tld" prefix="my" %>`
- ☐ E. **`/mywebapp/WEB-INF/tlds/mytags.tld`** is NOT a valid location for a tag library descriptor, so none of these will work.

–Option C is correct because, in the absence of a <taglib> element in the DD, the URI must be a full path relative to the application root.

55 Given:

(Servlet v2.4 sections 7.4 and 15.1.11)

```
11. public class ServletX extends HttpServlet {
12.   public void doGet(HttpServletRequest req,
13.                     HttpServletResponse res)
14.           throws IOException, ServletException {
15.     req.getSession().setAttribute("key", new X());
16.     req.getSession().setAttribute("key", new X());
17.     req.getSession().setAttribute("key", "x");
18.     req.getSession().removeAttribute("key");
19.   }
20. }
```

and given a listener class:

```
11. public class X implements HttpSessionBindingListener {
12.   public void valueBound(HttpSessionBindingEvent event) {
13.     System.out.print("B");
14.   }
15.   public void valueUnbound(HttpSessionBindingEvent event) {
16.     System.out.print("UB");
17.   }
18. }
```

Which logging output would be generated by an invocation of the **doGet** method?

- ❑ A. **UBUBUB** — Option A is incorrect because it implies that the valueBound method is never called.
- ☑ B. **BBUBUB**
- ❑ C. **BBUBUBB** — Options C and D are incorrect because they imply an extra call to the valueBound method when a String value is added to the session.
- ❑ D. **BUBBUBB**
- ❑ E. **BBUBUBBUB** — Options E and F are incorrect because they include calls to the valueBound and the valueUnbound methods when a String value is added to the session.
- ❑ F. **BBUBBUBBUB**

56 Given:

(Servlet v2.4 section 8)

```
10. public void doGet(HttpServletRequest req,
11.                   HttpServletResponse res)
12.         throws IOException, ServletException {
13.     RequestDispatcher rd1
14.       = getServletContext().getRequestDispatcher("/xyz");
15.     RequestDispatcher rd2
16.       = req.getRequestDispatcher("/xyz");
17.     RequestDispatcher rd3
18.       = getServletContext().getRequestDispatcher("xyz");
19.     RequestDispatcher rd4
20.       = req.getRequestDispatcher("xyz");
21. }
```

Which statements are true? (Choose all that apply.)

☑ A. **rd3** will never map to a servlet since the given path does NOT begin with **/**.

–Option A is correct because the path in this call must begin with "/".

☐ B. **rd4** will never map to a servlet since the given path does NOT begin with **/**.

–Option B is incorrect because a relative path is valid here.

☑ C. **rd2.forward(req, res)** and **rd4.forward(req, res)** may forward to the same resource.

–Option C is correct because these calls would refer to the same resource if the original request was for a resource at the top level (e.g. "/foo").

☑ D. **rd1.forward(req, res)** and **rd2.forward(req, res)** would always forward to the same resource.

–Option D is correct because both calls are using an absolute path relative to the servlet context root.

☐ E. **rd3.forward(req, res)** and **rd4.forward(req, res)** would always forward to the same resource.

–Option E is incorrect because rd3 is null.

57 Which JSTL tag performs URL rewriting?

(JSTL v1.1 section 7.6)

☐ A. **link**

☐ B. **aHref**

–Options A and B are not JSTL tags.

☐ C. **import**

☑ D. **url**

–Option C is invalid because the import tag does not perform URL rewriting.

58

(Servlet v2.4 pg. 258)

Given:

```
11. public void doGet(HttpServletRequest req,
12.                   HttpServletResponse res)
13.         throws IOException, ServletException {
14.     String url = res.encodeRedirectURL("/redirectme");
15.     boolean test = "/redirectme".equals(url);
16.     res.sendRedirect(url);
17. }
```

Which statements are true? (Choose all that apply.)

- ❑ A. After line 15, **test** will always be **truE**.
- ❑ B. After line 15, **test** will always be **false**.
- ☑ C. Line 14 demonstrates a session management mechanism called URL rewriting.
- ☑ D. After line 15, **test** could be either **true** ot **false**.
- ❑ E. The **encodeURL** method should have been used instead of the **encodeRedirectURL** method in line 14.
- ❑ F. The **encodeRedirectURL** method shown in line 14 should only be used when clients have disabled cookies.

–Options A and B are incorrect because the URL will be modified by line 14 only if necessary.

–Option E is incorrect because this is the correct method to be used with URLs that are to be passed to the sendRedirect method.

–Option F is incorrect because the encodeRedirectURL method should be used for all URLs sent through the sendRedirect method in order to support session management with browsers that do not support cookies.

59

(Servlet v2.4 pg. 80)

What happens when a container migrates a session from one VM to another?

- ☑ A. **sessionWillPassivate()** will be called on any objects that implement the **HttpSessionActivationListener** and are currently bound to the session.
- ❑ B. **sessionWillPassivate()** will be called on any objects that implement the **HttpSessionPassivationListener** and are currently bound to the session.
- ❑ C. **sessionWillPassivate()** will be called on any objects that implement the **HttpSessionListener** interface.
- ❑ D. **sessionWillPassivate()** will be called on any objects that implement the **HttpSessionBindingListener** and are currently bound to the session.

–Option A is the only one that refers to a valid interface/method combination.

60 Given an existing class:

(JSP v2.0 section 2.6)

```
 1. package com.example;
 2. public class MyFunctions {
 3.    private int x;
 4.    public MyFunctions()
 5.       { x = 0; }
 6.    public int modX(int y)
 7.       { return (y mod x); }
 8.    public static int funct(int x, int y)
 9.       { return (2*x + y - 5); }
10. }
```

❑ Which are true about EL functions? (Choose all that apply.)

❑ A. The **MyFunctions** class may NOT be used by EL because it has NOT been declared **final**.

–Option A is invalid because a class of functions need not be final.

☑ B. The **funct** method may be used by EL because it has been declared static.

–Option B is correct because only static methods may be used by EL.

❑ C. The **funct** method may NOT be use by EL because the calling arguments and return value must be object references.

–Option C is invalid because EL handles primitive values as well as objects.

❑ D. The **modX** method may be used by EL because it is an instance method.

–Option D is invalid because only static methods may be used by EL.

☑ E. The **MyFunctions** class may be used by EL even though it has a public constructor.

–Option E is correct because EL ignores all instance methods. EL doesn't care that it's dealing with a concrete class.

61 Which statements about ignoring EL in your JSPs are true? (Choose all that apply.)

(JSP v2.0 section 3.3.3)

☑ A. You can instruct the container to ignore EL in your JSPs by using the **<el-ignored>** tag in the DD.

❑ B. You can instruct the container to ignore EL in your JSPs by using the **<is-el-ignored>** tag in the DD.

❑ C. You can instruct the container to ignore EL in a JSP by using the **elIgnored** attribute of the **page** directive.

–Option C is invalid because the correct page directive attribute is isELIgnored.

☑ D. When using the DD to instruct the container to ignore EL, you can specify which JSPs to constrain.

–Option E is invalid because it's OK to constrain a given JSP from using both scripting and EL.

❑ E. You CANNOT constrain both scripting and EL in the same JSP.

62 You have purchased a purchase order web application that uses programmatic authorization that uses security roles that are not used in your organization.

(Servlet v2.4 section 12.3)

Which deployment descriptor element must you use to make this webapp work in your organization?

- ❑ A. **<login-config>**
- ❑ B. **<security-role>**
- ☑ C. **<security-role-ref>**
- ❑ D. **<security-constraint>**

–Option C is correct. The <security-role-ref> element is used to map roles hardcoded in a servlet to roles declared in the deployment descriptor. The other elements are used for declarative security.

63 Given:

(JSP v2.0 section 7)

```
1. <%@ taglib uri="http://www.mycompany.com/mytags"
               prefix="mytags" %>
2. <mytags:foo bar="abc" />
3. <mytags:forEach><mytags:doSomething /></mytags:forEach>
4. <jsp:setProperty name="x" property="a" value="b" />
5. <c:out value="hello" />
```

Assuming this is a complete JSP, which is true?

(For options E and F, ignore the fact that an error in one line might keep a subsequent line from being reached)

- ❑ A. Only line 2 will definitely generate an error.
- ❑ B. Only line 3 will definitely generate an error.
- ❑ C. Only line 4 will definitely generate an error.
- ☑ D. Only line 5 will definitely generate an error.
- ❑ E. Lines 4 and 5 will both definitely generate errors.
- ❑ F. Lines 2, 3, 4 and 5 will all definitely generate errors.
- ❑ G. The entire JSP could execute without generating any errors.

–Options A and B are incorrect because, assuming the tags used are appropriately defined in the mytags tag library, there is nothing wrong with them.

–Option C is incorrect because this is a valid JSP standard action. The jsp prefix does not need to be referenced in a taglib directive.

–Option D is correct because there is no taglib directive shown for the prefix c.

64 Which authentication mechanism employs a base64 encoding scheme to protect user passwords?

(Servlet v2.4 section 12.5.1)

- ☑ A. HTTP Basic Authentication
- ❑ B. Form Based Authentication
- ❑ C. HTTP Digest Authentication
- ❑ D. HTTPS Client Authentication

–Option A is correct. BTW, the base64 encoding scheme is considered to be a very weak protection scheme.

65 Which concepts are common to all four authentication mechanisms supported by J2EE 1.4 compliant web containers? (Choose all that apply.)

(Servlet v2.4 section 12.5)

- [x] A. passwords
- [ ] B. realm names
- [ ] C. generic error pages
- [x] D. secured web resources
- [ ] E. automatic SSL support
- [ ] F. target server authentication

–Options A and D are correct. All authentication schemes rely on passwords, and in J2EE 1.4, authentication is initiated only when a secured web resource is requested.

66 How are cookies used to support session management?

(Servlet v2.4 section 7.1.1)

- [ ] A. A cookie is created for each attribute stored in the session.
- [x] B. A single cookie is created to hold an ID that uniquely identifies a session.
- [ ] C. A single cookie is created to hold the serialized **`HttpSession`** object.
- [ ] D. The session ID is encoded as a path parameter in the URL string called **`jsessionid`**.
- [ ] E. Cookies CANNOT be used to support session management because it is possible for a user to disable cookies in their browser.

–Option A is incorrect because session data is not stored in cookies, just a session ID.

–Option C is incorrect because the session itself is not stored in the cookie, just a session ID.

–Option D is incorrect because it describes URL rewriting, not cookies.

–Option E is incorrect because cookies are the most commonly used session tracking mechanism (despite the possibility described here).

67 You are developing a web application for an organization that needs to display the results of database searches to several different types of clients, including browsers, PDAs, and kiosks. The application will have to examine the request to determine which type of client has initiated it, and then dispatch the request to the proper component.

(HFS – chap 14)

Which J2EE design pattern is designed for this type of application?

- [ ] A. Transfer Object
- [ ] B. Service Locator
- [x] C. Model-View-Controller
- [ ] D. Business Delegate
- [ ] E. Intercepting Filter

–One clue that MVC might be a good choice is when your application has to represent the same business data in several different views.

68 Which is true about the differences between the Classic and Simple tag models?

(JSP v2.0 section 13)

- ❑ A. A nested Classic tag is allowed to access its parent tag, but this is NOT supported in the Simple tag model.

–Option A is invalid because a Simple tag is allowed to access its parent.

- ❑ B. In the Classic model, you may gain access to the evaluated body content using the **`BodyTag`** interface. In the Simple model, you can invoke the body, but you CANNOT gain access to the content generated in the invocation.

–Option B is invalid because a Simple tag can pass in a Writer object to the JspFragment.invoke method that captures the output.

- ☑ C. The Tag interface has two event methods (**`doStartTag`** and **`doEndTag`**), but the **`SimpleTag`** interface only has one event method (**`doTag`**).

- ❑ D. Both tag models support iteration. In the **`SimpleTag.doTag`** method, you can invoke the body within a Java-based iteration. In the Classic model, iteration may be supported by returning the **`EVAL_BODY_AGAIN`** constant from the **`Tag.doEndTag`** method.

–Option D is invalid because it is not the doEndTag method that is used to perform iteration (it is the doAfterBody method).

69 Given this class:

(JSP v2.0 section 5.2)

```
1. package biz.mybiz;
2. public class BeanX {
3.     private String a,b,c;
4.     public BeanX() {a="A";b="B";c="C";}
5.     public void setA(String a) { this.a = a; }
6.     public void setB(String b) { this.b = b; }
7.     public void setC(String c) { this.c = c; }
8.     public String getAll() { return a+b+c; }
9. }
```

and the JSP:

```
1. <jsp:useBean id="x" class="biz.mybiz.BeanX" />
2. <jsp:setProperty name="x" property="*" />
3. <jsp:getProperty name="x" property="all" />
4. <jsp:setProperty name="x" property="a" param="b" />
5. <jsp:setProperty name="x" property="b" param="c" />
6. <jsp:setProperty name="x" property="c" param="a" />
7. <jsp:getProperty name="x" property="all" />
```

What will be generated by the JSP when invoked with the query string **a=X&b=Y&c=Z**?

- ❑ A. **ABC YZX**
- ❑ B. **XYZ XYZ**
- ❑ C. **ABC ABC**
- ❑ D. **YXZ YZX**
- ❑ E. **XYZ ZXY**
- ☑ F. **XYZ YZX**
- ❑ G. **nullnullnull YZX**

–Option F is correct because the first jsp:setProperty call sets all three properties based on the parameters in the query string and then the subsequent jsp:setProperty calls change their values.