1 How would servlet code from a service method (e.g., **doPost()**) retrieve the value of the "User-Agent" header from the request? (Choose all that apply.) (API)

- ❑ A. `String userAgent = request.getParameter("User-Agent");`
- ☑ B. `String userAgent = request.getHeader("User-Agent");`
- ❑ C. `String userAgent = request.getRequestHeader("Mozilla");`
- ❑ D. `String userAgent = getServletContext().getInitParameter("User-Agent");`

–Option B shows the correct method call passing in the header name as a String parameter.

2 Which HTTP methods are used to show the client what the server is receiving? (Choose all that apply.) (HF 4, HTTP methods)

- ❑ A. GET
- ❑ B. PUT
- ☑ C. TRACE
- ❑ D. RETURN
- ❑ E. OPTIONS

–This method is typically used for troubleshooting, not for production.

3 Which method of **HttpServletResponse** is used to redirect an HTTP request to another URL? (API)

- ❑ A. **sendURL()**
- ❑ B. **redirectURL()**
- ❑ C. **redirectHttp()**
- ☑ D. **sendRedirect()**
- ❑ E. **getRequestDispatcher()**

– Option D is correct, and of the methods listed, it's the only one that exists in HttpServletResponse

4 Which HTTP methods are NOT considered idempotent? (Choose all that apply.)

(HF 4, idempotent requests)

- ❑ A. GET
- ☑ B. POST
- ❑ C. HEAD
- ❑ D. PUT

–By design, POST is meant to convey requests to update the state of the server. In general the same update should not be applied multiple times.

5 Given **req** is a **HttpServletRequest**, which gets a binary input stream? (Choose all that apply.)

(API)

- ❑ A. `BinaryInputStream s = req.getInputStream();`
- ☑ B. `ServletInputStream s = req.getInputStream();`
- ❑ C. `BinaryInputStream s = req.getBinaryStream();`
- ❑ D. `ServletInputStream s = req.getBinaryStream();`

–Option B specifies the correct method and the correct return type.

6 How would you set a header named "CONTENT-LENGTH" in the **HttpServletResponse** object? (Choose all that apply.)

(API)

- ❑ A. `response.setHeader(CONTENT-LENGTH,"numBytes");`
- ☑ B. `response.setHeader("CONTENT-LENGTH","numBytes");`
- ❑ C. `response.setStatus(1024);`
- ❑ D. `response.setHeader("CONTENT-LENGTH",1024);`

–Option B shows the correct way to set an HTTP header with two String parameters, one representing the header name and the other the value.

7 Choose the servlet code fragment that gets a binary stream for writing an image or other binary type to the **HttpServletResponse**.

(API)

- ❑ A. `java.io.PrintWriter out = response.getWriter();`
- ☑ B. `ServletOutputStream out = response.getOutputStream();`
- ❑ C.
```
java.io.PrintWriter out =
        new PrintWriter(response.getWriter());
```
- ❑ D. `ServletOutputStream out = response.getBinaryStream();`

–Option A is incorrect because it uses a character-oriented PrintWriter

8 Which methods are used by a servlet to handle form data from a client? (Choose all that apply.) (API)

- ❑ A. **`HttpServlet.doHead()`**
- ☑ B. **`HttpServlet.doPost()`**
- ❑ C. **`HttpServlet.doForm()`**
- ❑ D. **`ServletRequest.doGet()`**
- ❑ E. **`ServletRequest.doPost()`**
- ❑ F. **`ServletRequest.doForm()`**

–Options C–F are wrong because these methods don't exist.

9 Which of the following methods are declared in **`HttpServletRequest`** as opposed to in **`ServletRequest`**? (Choose all that apply.) (API)

- ☑ A. **`getMethod()`**
- ☑ B. **`getHeader()`**
- ☑ C. **`getCookies()`**
- ❑ D. **`getInputStream()`**
- ❑ E. **`getParameterNames()`**

–Options A, B, and C all relate to components of an HTTP request.

10 How should servlet developers handle the **`HttpServlet`**'s **`service()`** method when extending **`HttpServlet`**? (Choose all that apply.) (API)

- ❑ A. They should override the **`service()`** method in most cases.
- ❑ B. They should call the **`service()`** method from **`doGet()`** or **`doPost()`**
- ❑ C. They should call the **`service()`** method from the **`init()`** method.
- ☑ D. They should override at least one **`doXXX()`** method (such as **`doPost()`**).

–Option D is correct, developers typically focus on the doGet(), and doPost() methods

Chapter 5 Answers

1 When using a **RequestDispatcher**, the use of which methods can often lead to an **IllegalStateException**? (Choose all that apply.)

(Servlet v2.4 pg. 167)

- ❑ A. **read**
- ☑ B. **flush**
- ❑ C. **write**
- ❑ D. **getOutputStream**
- ❑ E. **getResourceAsStream**

–An IllegalStateException is caused when a response has already been 'committed' to the client (the flush method does that), and then you attempt a forward.

2 Which statements about **ServletContext** initialization parameters are true? (Choose all that apply.)

(Servlet v2.4 pg. 31)

- ☑ A. They should be used for data that changes rarely.
- ❑ B. They should be used for data that changes frequently.
- ❑ C. They can be accessed using **ServletContext.getParameter()**.
- ☑ D. They can be accessed using **ServletContext.getInitParameter()**.
- ❑ E. They should be used for data that is specific to a particular servlet.
- ☑ F. They should be used for data that is applicable to an entire web application.

–Option B is incorrect because ServletContext init parameters are only read at Container start-up time.

–Option C is incorrect because this method does not exist.

–Option E is incorrect because there is only one ServletContext object per web application.

3 Which types define the methods **getAttribute()** and **setAttribute()**? (Choose all that apply.)

(Servlet v2.4 pgs. 32, 36, 59)

- [x] A. **HttpSession**
- [x] B. **ServletRequest**
- [ ] C. **ServletResponse**
- [x] D. **ServletContext**
- [ ] E. **ServletConfig**
- [ ] F. **SessionConfig**

4 If a servlet is invoked using the **forward** or **include** method of **RequestDispatcher**, which methods of the servlet's request object can access the request attributes set by the container? (Choose all that apply.)

(Servlet v2.4 65-66)

- [ ] A. **getCookies()**
- [x] B. **getAttribute()**
- [ ] C. **getRequestPath()**
- [ ] D. **getRequestAttribute()**
- [ ] E. **getRequestDispatcher()**

-Option B is the correct method. With it you can access the container populated javax.servlet.forward.Xxx and javax.servlet.include.Xxxx attributes.

-Options C and D refer to methods that don't exist.

5 Which calls provide information about initialization parameters that are applicable to an entire web application? (Choose all that apply.)

(Servlet v2.4 pg. 32)

- [ ] A. **ServletConfig.getInitParameters()**
- [ ] B. **ServletContext.getInitParameters()**
- [ ] C. **ServletConfig.getInitParameterNames()**
- [x] D. **ServletContext.getInitParameterNames()**
- [ ] E. **ServletConfig.getInitParameter(String)**
- [x] F. **ServletContext.getInitParameter(String)**

-Options A and B are incorrect because these methods do not exist.

-Options C and E are incorrect because they provide access to servlet-specific initialization parameters.

6 Which statements about listeners defined in the **`javax.servlet`** package are true? (Choose all that apply.)

(Servlet v2.4 pg. 80)

- ❑ A. A **`ServletResponseListener`** can be used to perform an action when a servlet response has been sent.
- ☑ B. An **`HttpSessionListener`** can be used to perform an action when an **`HttpSession`** has timed out.
- ☑ C. A **`ServletContextListener`** can be used to perform an action when the servlet context is about to be shut down.
- ☑ D. A **`ServletRequestAttributeListener`** can be used to perform an action when an attribute has been removed from a **`ServletRequest`**.
- ❑ E. A **`ServletContextAttributeListener`** can be used to perform an action when the servlet context has just been created and is available to service its first request.

-Option A is incorrect because these is no ServletResponseListener interface.

-Option E is incorrect because a ServletContextListener would be used for this purpose.

7 Which is most logically stored as an attribute in session scope?

(Servlet v2.4 pg. 58)

- ❑ A. A copy of a query parameter entered by a user.
- ❑ B. The result of a database query to be returned immediately to a user.
- ❑ C. A database connection object used by all web components of the system.
- ☑ D. An object representing a user who has just logged into the system.
- ❑ E. A copy of an initialization parameter retrieved from a **`ServletContext`** object.

-Option A is incorrect because a query parameter is more typically used immediately to perform an operation.

-Option B is incorrect because such data is typically either immediately returned or stored in request scope.

-Option C is incorrect because (since it is not specific to a particular session) it should be stored in context scope.

-Option E is incorrect because servlet context parameters should stay with the ServletContext object.

8 Given this code from an otherwise valid **HttpServlet** that has also been registered as a **ServletRequestAttributeListener**:

(Servlet v2.4 pg. 199-200)

```
10. public void doGet(HttpServletRequest req,
                      HttpServletResponse res)
11.         throws IOException, ServletException {
12.    req.setAttribute("a", "b");
13.    req.setAttribute("a", "c");
14.    req.removeAttribute("a");
15. }
16. public void attributeAdded(ServletRequestAttributeEvent ev) {
17.    System.out.print(" A:" + ev.getName() + "->" + ev.getValue());
18. }
19. public void attributeRemoved(ServletRequestAttributeEvent ev) {
20.    System.out.print(" M:" + ev.getName() + "->" + ev.getValue());
21. }
22. public void attributeReplaced(ServletRequestAttributeEvent ev) {
23.    System.out.print(" P:" + ev.getName() + "->" + ev.getValue());
24. }
```

What logging output is generated?

- ❑ A. **A:a->b P:a->b**
- ❑ B. **A:a->b M:a->c**
- ☑ C. **A:a->b P:a->b M:a->c**
- ❑ D. **A:a->b P:a->b P:a->null**
- ❑ E. **A:a->b M:a->b A:a->c M:a->c**
- ❑ F. **A:a->b M:a->b A:a->c P:a->null**

-Tricky! The getValue method returns the OLD value of the attribute if the attribute was replaced.

9 When declaring a listener in the DD, which sub-elements of the **<listener>** element are required? (Choose all that apply.)

(Servlet v2.4 section 10.4, & 13.4.9)

- ❑ A. **<description>**
- ❑ B. **<listener-name>**
- ❑ C. **<listener-type>**
- ☑ D. **<listener-class>**
- ❑ E. **<servlet-mapping>**

-The <listener-class> sub-element is the ONLY required sub-element of the <listener> element.

10 (API)

Which types of objects can store attributes? (Choose all that apply.)

- ❑ A. **ServletConfig**
- ❑ B. **ServletResponse**
- ❑ C. **RequestDispatcher**
- ☑ D. **HttpServletRequest**
- ❑ E. **HttpSessionContext**

–Options A, B, and C are invalid because these types do not store attributes.

–Option E is invalid because there is no such type.

Note: The other two types related to servlets, that can store attributes are HttpSession and ServletContext.

11 (Servlet v2.4 pgs. 81–84)

Which are true? (Choose all that apply.)

- ❑ A. When a web application is preparing to shutdown, the order of listener notification is not guaranteed.
- ❑ B. When listener-friendly events occur, listener invocation order is not predictable.
- ☑ C. The container registers listeners based on declarations in the deployment descriptor.
- ❑ D. Only the container can invalidate a session.

–Options A and B are incorrect because the container uses the DD to determine the notification order of registered listeners.

–Option D is incorrect because a servlet can invalidate a session using the HttpSession.invalidate() method.

12 (Servlet v2.4 pg. 65)

Which statements about **RequestDispatcher** are true (where applicable, assume the **RequestDispatcher** was not obtained via a call to **getNamedDispatcher()**)? (Choose all that apply.)

- ☑ A. A **RequestDispatcher** can be used to forward a request to another servlet.
- ❑ B. The only method in the **RequestDispatcher** interface is **forward()**.
- ❑ C. Parameters specified in the query string used to create a **RequestDispatcher** are not forwarded by the **forward()** method.
- ☑ D. The servlet to which a request is forwarded may access the original query string by calling **getQueryString()** on the **ServletRequest**.
- ☑ E. The servlet to which a request is forwarded may access the original query string by calling **getAttribute("javax.servlet.forward.query_string")** on the **ServletRequest**.

–Option B is incorrect because the interface also contains an include method.

–Option C is incorrect because such parameters are forwarded in this case.

13 Which statements accurately describe how many instances of a servlet the servlet container instantiates for each web application? (Choose all that apply.)

(Servlet spec p 24)

- [x] A. If the servlet implements **`javax.servlet.SingleThreadModel`**, the container may create one instance for each request.
- [ ] B. If the servlet does not implement **`SingleThreadModel`**, the container may create multiple instances of the servlet in the same JVM.
- [ ] C. The **`<load-on-startup>`** **`web.xml`** element can determine how many instances are created.
- [x] D. If the servlet does not implement **`SingleThreadModel`**, the container will create no more than one instance per JVM.

-Option C is incorrect because the <load-on-startup> deployment-descriptor element determines the order of instantiation, not the number of instances.

14 What is the recommended way to deal with servlets and thread safety?

(Servlet spec p 27)

- [ ] A. Write the servlet code to extend **`ThreadSafeServlet`**.
- [ ] B. Have the servlet implement **`SingleThreadModel`**.
- [ ] C. Log all servlet method calls.
- [x] D. Use local variables exclusively, and if you have to use instance variables, synchronize access to them.

-Option A and B are incorrect because ThreadSafeServlet does not exist in the Servlet API and the .SingleThreadModel is deprecated in version 2.4 and not recommended..

1 Given:

(Servlet Spec p. 59)

```
10. public class MyServlet extends HttpServlet {
11.   public void doGet(HttpServletRequest req,
                        HttpServletResponse res)
12.          throws IOException, ServletException {
13.   // request.getSession().setAttribute("key", "value");
14.   // request.getHttpSession().setAttribute("key", "value");
15.   // ((HttpSession)request.getSession()).setAttribute("key", "value");
16.   // ((HttpSession)request.getHttpSession()).setAttribute("key", "value");
17.   }
18. }
```

Which line(s) could be uncommented without causing a compile or runtime error? (Choose all that apply.)

- ❑ A. Line 13 only.
- ❑ B. Line 14 only.
- ❑ C. Line 15 only.
- ❑ D. Line 16 only.
- ☑ E. Line 13 or line 15.
- ❑ F. Line 14 or line 16.

–Option E is correct because both lines 13 and 15 make the correct method call. The cast to HttpSession is NOT necessary, but it does reflect the correct type, so it is valid.

2 If a client will NOT accept a cookie, which session management mechanism could the web container employ? (Choose one.)

(Servlet v2.4 pg. 57)

- ❑ A. Cookies, but NOT URL rewriting.
- ☑ B. URL rewriting, but NOT cookies.
- ❑ C. Either cookies or URL rewriting can be used.
- ❑ D. Neither cookies nor URL rewriting can be used.
- ❑ E. Cookies and URL rewriting must be used together.

–Option B is correct because cookies CANNOT be used, but URL rewriting does NOT depend on cookies being enabled.

3 Which statements about **HttpSession** objects are true? (Choose all that apply.) (Servlet v2.4 p. 59)

- [x] A. A session whose timeout period has been set to **-1** will never expire.
- [ ] B. A session will become invalid as soon as the user closes all browser windows.
- [x] C. A session will become invalid after a timeout period defined by the servlet container.
- [ ] D. A session may be explicitly invalidated by calling **HttpSession.invalidateSession()**.

-Option B is incorrect because there is no explicit termination signal in the HTTP protocol.

-Option D is incorrect because the method that should be used is called invalidate().

4 Which of the following are NOT listener event types in the J2EE 1.4 API? (Choose all that apply.) (API)

- [ ] A. **HttpSessionEvent**
- [ ] B. **ServletRequestEvent**
- [ ] C. **HttpSessionBindingEvent**
- [x] D. **HttpSessionAttributeEvent**
- [ ] E. **ServletContextAttributeEvent**

-HttpSessionBindingEvents are used for both HttpSessionBindingListeners AND HttpSessionAttributeListeners.

5 Which statements about session tracking are true? (Choose all that apply.) (Servlet v2.4 p. 57)

- [x] A. URL rewriting may be used by a server as the basis for session tracking.
- [x] B. SSL has a built-in mechanism that a servlet container could use to obtain data used to define a session.
- [ ] C. When using cookies for session tracking, there is no restriction on the name of the session tracking cookie.
- [x] D. When using cookies for session tracking, the name of the session tracking cookie must be **JSESSIONID**.
- [ ] E. If a user has cookies disabled in their browser, the container may choose to use a **javax.servlet.http.CookielessHttpSession** object to track the user's session.

-Option C is incorrect because the specification dictates that the session tracking cookie must be JSESSIONID.

-Option E is incorrect because there is no such class.

6

Given: (Servlet v2.4 p. 276)

```
1. import javax.servlet.http.*;
2. public class MySessionListener
          implements HttpSessionListener {
3.      public void sessionCreated() {
4.          System.out.println("Session Created");
5.      }
6.      public void sessionDestroyed() {
7.          System.out.println("Session Destroyed");
8.      }
9. }
```

What is wrong with this class? (Choose all that apply.)

- [x] A. The method signature on line 3 is NOT correct.
- [x] B. The method signature on line 6 is NOT correct.
- [ ] C. The import statement will NOT import the **HttpSessionListener** interface.
- [ ] D. **sessionCreated** and **sessionDestroyed** are NOT the only methods defined by the **HttpSessionListener** interface.

– Options A and B are correct because these methods should have an HttpSessionEvent parameter.

– Option C is incorrect because the listener is defined in the imported package.

– Option D is incorrect because these are the only two methods in this interface.

7

Which statements about session attributes are true? (Choose all that apply.) (Servlet v2.4 p. 59)

- [x] A. The return type of **HttpSession.getAttribute(String)** is **Object**.
- [ ] B. The return type of **HttpSession.getAttribute(String)** is **String**.
- [x] C. Attributes bound into a session are available to any other servlet that belongs to the same **ServletContext**.
- [ ] D. Calling **setAttribute("keyA", "valueB")** on an **HttpSession** which already holds a value for the key **keyA** will cause an exception to be thrown.
- [x] E. Calling **setAttribute("keyA", "valueB")** on an **HttpSession** which already holds a value for the key **keyA** will cause the previous value for this attribute to be replaced with the String **valueB**.

– Option B is incorrect because the return type is Object.

– Option D is incorrect because this call will simply replace the existing value.

8 Which interfaces define a **getSession()** method? (Choose all that apply.) (Servlet v2.4 pg. 243)

- ❑ A. **ServletRequest**
- ❑ B. **ServletResponse**
- ☑ C. **HttpServletRequest**
- ❑ D. **HttpServletResponse**

---

9 Given a session object **s**, and the code: (Servlet v2.4 pg. 80)

```
s.setAttribute("key", value);
```

Which listeners could be notified? (Choose one.)

- ❑ A. Only **HttpSessionListener**
- ❑ B. Only **HttpSessionBindingListener**
- ❑ C. Only **HttpSessionAttributeListener**
- ❑ D. **HttpSessionListener** and **HttpSessionBindingListener**
- ❑ E. **HttpSessionListener** and **HttpSessionAttributeListener**
- ☑ F. **HttpSessionBindingListener** and **HttpSessionAttributeListener**
- ❑ G. All three

–Option F is correct because an HttpSessionAttributeListener is notified any time an attribute is added and the value object will also be notified if it implements an HttpSessionBindingListener.

---

10 Given that **req** is an **HttpServletRequest**, which snippets create a session if one doesn't exist? (Choose all that apply.) (API)

- ☑ A. **req.getSession();**
- ☑ B. **req.getSession(true);**
- ❑ C. **req.getSession(false);**
- ❑ D. **req.createSession();**
- ❑ E. **req.getNewSession();**
- ❑ F. **req.createSession(true);**
- ❑ G. **req.createSession(false);**

–Options A and B will each create a new session if one doesn't exist. getSession(false) returns a null if the session doesn't exist.

11 Given a session object **s** with two attributes named **myAttr1** and **myAttr2**, which will remove both attributes from this session? (Choose all that apply.) (API)

- ❑ A. `s.removeAllValues();`
- ☑ B. `s.removeAttribute("myAttr1");`
  `s.removeAttribute("myAttr2");`
- ❑ C. `s.removeAllAttributes();`
- ❑ D. `s.getAttribute("myAttr1", UNBIND);`
  `s.getAttribute("myAttr2", UNBIND);`
- ❑ E. `s.getAttributeNames(UNBIND);`

–Option B is correct, removeAttribute() is the only way to remove attributes from a session object, and it removes only one attribute at a time.

12 Which statements about **HttpSession** objects in distributed environments are true? (Choose all that apply.) (Servlet v2.4 pg. 60)

- ❑ A. When a session is moved from one JVM to another, any attributes stored in the session will be lost.
- ❑ B. When a session is moved from one JVM to another, appropriately registered **HttpSessionBindingListener** objects will be notified.
- ☑ C. When a session is moved from one JVM to another, appropriately registered **HttpSessionActivationListener** objects will be notified.
- ☑ D. When a session is moved from one JVM to another, attribute values that implement **java.io.Serializable** will be transferred to the new JVM.

–Option A is incorrect because serializable attributes will be transferred.

–Option B is incorrect since attributes remain bound to the session.

13 Which statements about session timeouts are true? (Choose all that apply.) (API)

- ❑ A. Session timeout declarations made in the DD can specify time in seconds.
- ☑ B. Session timeout declarations made in the DD can specify time in minutes.
- ☑ C. Session timeout declarations made programmatically can specify time only in seconds.
- ❑ D. Session timeout declarations made programmatically can specify time only in minutes.
- ❑ E. Session timeout declarations made programmatically can specify time in either minutes or seconds.

–In the DD, using the <session-timeout> element, only minutes can be specified, using HttpSession's setMaxInactiveInterval() only seconds can be specified.

14 Choose the servlet code fragment that would retrieve from the request the value of a cookie named "ORA_UID"? (Choose all that apply.) (API)

❑ A. `String value = request.getCookie("ORA_UID");`

– Option A refers to a method that doesn't exist.

❑ B. `String value = request.getHeader("ORA_UID");`

☑ C.
```
javax.servlet.http.Cookie[] cookies =
  request.getCookies();
String cName = null;
String value = null;
if (cookies != null){
  for (int i = 0; i < cookies.length; i++){
    cName = cookies[i].getName();
    if (cName != null &&
        cName.equalsIgnoreCase("ORA_UID")){
      value = cookies[i].getValue();
    }
  }
}
```

– Option C gets a Cookie array using request.getCookies(), then checks for a Cookie of a specified name.

❑ D.
```
javax.servlet.http.Cookie[] cookies =
    request.getCookies();
if (cookies.length > 0){
  String value = cookies[0].getValue();
}
```

– Option D only looks at the first Cookie in the array.

1 Given this DD element: (JSP v2.0 pg 1-87)

```
47. <jsp-property-group>
48.    <url-pattern>*.jsp</url-pattern>
49.    <el-ignored>true</el-ignored>
50. </jsp-property-group>
```

What does the element accomplish? (Choose all that apply.)

- ❑ A. All files with the specified extension mapping should be treated by the JSP container as well-formed XML files.
- ❑ B. All files with the specified extension mapping should have any Expression Language code evaluated by the JSP container.
- ☑ C. By default, all files with the specified extension mapping should NOT have any Expression Language code evaluated by the JSP container.
- ❑ D. Nothing, this tag is NOT understood by the container.
- ❑ E. Although this tag is legal, it is redundant, because the container behaves this way by default.

–Option C turns off the evaluating of EL expressions by a JSP 2.0 container and by default the container does evaluate EL.

---

2 Which directives specify an HTTP response that will be of type "image/svg"? (Choose all that apply.) (JSP v2.0 section 1.10.1)

- ❑ A. `<%@ page type="image/svg" %>`
- ❑ B. `<%@ page mimeType="image/svg" %>`
- ❑ C. `<%@ page language="image/svg" %>`
- ☑ D. `<%@ page contentType="image/svg" %>`
- ❑ E. `<%@ page pageEncoding="image/svg" %>`

–Option D is the correct syntax for this directive.

3 Given this JSP: (JSP v2.0 section 1)

```
1.  <%@ page import="java.util.*" %>
2.  <html><body> The people who like
3.  <%= request.getParameter("hobby") %>
4.  are: <br>
5.  <% ArrayList al = (ArrayList) request.getAttribute("names"); %>
6.  <% Iterator it = al.iterator();
7.     while (it.hasNext()) { %>
8.       <%= it.next() %>
9.  <br>
10. <% } %>
11. </body></html>
```

Which types of code are used in this JSP? (Choose all that apply.)

- ☐ A. EL
- ☑ B. directive
- ☑ C. expression
- ☑ D. template text
- ☑ E. scriptlet

–There's no EL in this JSP. There's a directive on line 1, expressions on lines 3 and 8, template text all over (like line 2), and of course scripting elements.

4 Which statements about **jspInit()** are true? (Choose all that apply.) (JSP v2.0 section 11.2.1)

- ☑ A. It has access to a **ServletConfig**.
- ☑ B. It has access to a **ServletContext**.
- ☑ C. It is only called once.
- ☑ D. It can be overridden.

5 Which types of objects are available to the **jspInit()** method? (Choose all that apply.)

(JSP v2.0 section 11.2.1)

- [x] A. **ServletConfig**
- [x] B. **ServletContext**
- [ ] C. **JspServletConfig**
- [ ] D. **JspServletContext**
- [ ] E. **HttpServletRequest**
- [ ] F. **HttpServletResponse**

–JSPs turn into plain old servlets, so they have access to the plain old ServletConfig and ServletContext objects... and it's just a little early in the lifecycle to be talking about requests and responses.

6 Given:

(JSP v2.0 pg 1-49)

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

What is the effect? (Choose all that apply.)

- [ ] A. Nothing, this **page** directive is NOT defined.
- [ ] B. The directive turns off the evaluation of Expression Language code by the JSP container in all of the web application's JSPs.
- [ ] C. The JSP containing this directive should be treated by the JSP container as a well-formed XML file.
- [ ] D. The JSP containing this directive should NOT have any Expression Language code evaluated by the JSP container.
- [x] E. This page directive will only turn off EL evaluation if the DD declares a **<el-ignored>true</el-ignored>** element with a URL pattern that includes this JSP.

–Option B is incorrect because the directive only affects the enclosing JSP.

7 Which statement concerning JSPs is true? (Choose one.)

(JSP v2.0 section 11)

- [ ] A. Only **jspInit()** can be overridden.
- [ ] B. Only **jspDestroy()** can be overridden.
- [ ] C. Only **_jspService()** can be overridden.
- [x] D. Both **jspInit()** and **jspDestroy()** can be overridden.
- [ ] E. **jspInit()**, **jspDestroy()**, and **_jspService()** can all be overridden.

–Remember the underscore is your clue that a method can't be overridden.

8 Which JSP lifecycle step is out of order? (JSP v2.0 section 11)

- ❑ A. Translate the JSP into a servlet.
- ❑ B. Compile servlet source code.
- ☑ C. Call **`_jspService()`** — The _jspService method can never be called before jspInit.
- ❑ D. Instantiate the servlet class.
- ❑ E. Call **`jspInit()`**
- ❑ F. Call **`jspDestroy()`**

---

9 Which are valid JSP implicit variables? (Choose all that apply.) (JSP v2.0 section 1.8.3)

- ❑ A. **`stream`**
- ❑ B. **`context`**
- ☑ C. **`exception`**
- ❑ D. **`listener`**
- ☑ E. **`application`**

Options A, B, and D don't exist as implicit objects created by the container for JSPs.

---

10 Given a request with two parameters: one named "first" represents a user's first name and another named "last" represents his last name. (JSP v2.0 pg 1-41)

Which JSP scriptlet code outputs these parameter values?

- ☑ A.
```
<% out.println(request.getParameter("first"));
   out.println(request.getParameter("last")); %>
```
Option A uses the "out" implicit object and its println() method.

- ❑ B.
```
<% out.println(application.getInitParameter("first"));
   out.println(application.getInitParameter("last")); %>
```

- ❑ C.
```
<% println(request.getParameter("first"));
   println(request.getParameter("last")); %>
```

- ❑ D.
```
<% println(application.getInitParameter("first"));
   println(application.getInitParameter("last")); %>
```

Options C and D are missing the "out" implicit object.

11 Given: (JSP v2.0 pg 1-10)

```
11. Hello ${user.name}!
12. Your number is <c:out value="${user.phone}"/>.
13. Your address is <jsp:getProperty name="user" property="addr" />
14. <% if (user.isValid()) {%>You are valid!<% } %>
```

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

- [x] A. Lines 11 and 12 (and no others) contain examples of EL elements.
- [x] B. Line 14 is an example of scriptlet code.
- [ ] C. None of the lines in this example contain template text.
- [ ] D. Lines 12 and 13 include examples of JSP standard actions.
- [ ] E. Line 11 demonstrates an invalid use of EL.
- [x] F. All four lines in this example would be valid in a JSP page.

-Option C is incorrect because all four lines include template text.

-Option D is incorrect because line 12 does not include a JSP standard action.

-Option E is incorrect because the EL in line 11 is valid.

---

12 Which JSP expression tag will print the context initialization parameter named "javax.sql.DataSource"? (JSP v2.0 pg 1-41)

- [ ] A. `<%= application.getAttribute("javax.sql.DataSource") %>`
- [x] B. `<%= application.getInitParameter("javax.sql.DataSource") %>`
- [ ] C. `<%= request.getParameter("javax.sql.DataSource") %>`
- [ ] D. `<%= contextParam.get("javax.sql.DataSource") %>`

-Option B shows the correct use of the application implicit object.

---

13 Which statements about disabling scripting elements are true? (Choose all that apply.) (JSP v2.0 section 3.3.3)

- [ ] A. You can't disable scripting via the DD.
- [ ] B. You can only disable scripting at the application level.
- [ ] C. You can disable scripting programmatically by using the **`isScriptingEnabled`** page directive attribute.
- [x] D. You can disable scripting via the DD by using the **`<scripting-invalid>`** element.

-You can only disable scripting elements through the DD. The <jsp-property-group> element allows you to disable scripting in selective JSPs by specifying URL patterns to be disabled.