MULTIPLE CHOICES QUESTIONS – PRJ301

BY PHONGNTSE150974

|  |  |
| --- | --- |
| QN=1 | What is a Servlet? |
| a. | A Servlet is a Java program. |
| b. | A Servlet is a Java program which receives and responds to requests from Web clients. |
| c. | A Servlet is a small Java program that runs within a Web server. Servlets receive to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol. |
| d. | A Servlet is a Java program that runs within the Web server which receives and responds to requests from Web clients, usually across HTTP. |
| e. | A Servlet is a server program that can compile Java code into byte code. |
| f. | A Servlet is the Controller of the Web application that can process request from the Web Clients. |
| ANSWER: | D |
| MARK: | 0.2 |
| UNIT: | Chap 3 |
| LO: |  |
| MIX CHOICES: | The Java EE API specification defines a Servlet as follows:  A Servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.  <http://docs.oracle.com/javaee/7/api/javax/servlet/Servlet.html> |

|  |  |
| --- | --- |
| QN=2 | Which statement below is correct? |
| a. | We can define a method within Scriptlets <% %>. |
| b. | All the variables and methos that are defined in Declarations <%! %> is ***private*** by default. |
| c. | We can’t write multi-line code in Scriptlets <% %>. |
| d. | The variables and methods defined in Declarations <%! %> are inserted outside the Servlet class when JSP page is compiled. |
| e. | The expressions within Expressions <%= %> are compiled and put into Response Object. |
| f. | Code within 3 types of Scripting Elements are inserted into the *\_jspservice()* method when they are compiled. |
| ANSWER: | E |
| MARK: | 0.2 |
| UNIT: | Chap 4 |
| LO: |  |
| MIX CHOICES: | (A) We can’t define a method within Scriptlets <% %> because the code within the Scriptlets will be put into the *\_jspservice()* method where we can’t define any method.  (B) All the variables and methos that are defined in Declarations <%! %> is ***default*** by default.  (C) We can write multi-line code in Scriptlets <% %> based on Java rules.  (D) The variables and methods defined in Declarations <%! %> are inserted inside the Servlet class but outside the *\_jspservice()* method when JSP page is compiled.  (E) The expressions within Expressions <%= %> are compiled and put into Response Object (Correct).  (F) The code within Declarations <%! %> is put inside the Servlet class but outside the *\_jspservice()* method. |

|  |  |
| --- | --- |
| QN=3 | Which of the following is a correct JSP syntax to print the value of *x*? |
| a. | <% int x = 1 %>  <%= x %> |
| b. | <% int x = 1; %>  <%= x %> |
| c. | <% x = 1 %>  <%= x %> |
| d. | <% int x = 1; %>  <%= x; %> |
| e. | <% int x = 1;  x;  %> |
| f. | <%! x; %>  <%= x %> |
| ANSWER: | B |
| MARK: | 0.2 |
| UNIT: | Chap 3 |
| LO: |  |
| MIX CHOICES: | - All the code within the Scriptlets must follow Java rules, which is, in this situation, that every code line is ended with a **;** .  - The code within the Expressions has to be a Statement which means that it can’t be ended with a **;** .  - To print value of a variable into the Client browser we use the Expressions or the out.print() method within the Scriptlets. |

|  |  |
| --- | --- |
| QN=4 | Why do all the Java classes within the Web server need to implement the *java.io.Serializable* interface? |
| a. | Because the Oracle Document tell us to do that. |
| b. | Because we want our Java classes to communicate with the client. |
| c. | Because the Java classes within the Web server need to communicate with each other through the network environment and the *java.io.Serializable* helps them do that. |
| d. | Because the Java classes need to be converted into byte code so that the Container can understand and process them. |
| e. | Because the Java classes need to communicate with Tomcat and the *java.io.Serializable* helps them do that. |
| f. | Because the Servlet implements the *java.io.Serializable* so that other Java classes have to follow. |
| ANSWER: | C |
| MARK: | 0.2 |
| UNIT: | Chap 3 |
| LO: |  |
| MIX CHOICES: | Because we are working within a Web application so that we need to communicate through network. The *java.io.Serializable* interface helps us convert Java Objects into byte stream that we can send over the network. |

|  |  |
| --- | --- |
| QN=5 | Which of the following is the correct sequence for a JDBC execution? |
| a. | Load Driver -> Open Connection -> Create Statement and Assign Parameters (If any) -> Execute Query |
| b. | Open Connection -> Load Driver -> Create Statement or Assign Parameters (If any) -> Execute Query |
| c. | Load Driver -> Create Statement and Assign Parameters (If any) -> Open Connection -> Execute Query |
| d. | Create Statement and Assign Parameters (If any) -> Open Connection -> Execute Query -> Load Driver |
| e. | Open Connection -> Create Statement and Assign Parameters (If any) -> Load Driver -> Execute Query |
| f. | Open Connection -> Create Statement and Assign Parameters (If any) -> Execute Query -> Load Driver |
| ANSWER: | A |
| MARK: | 0.2 |
| UNIT: | Oracle Docs for JDBC |
| LO: |  |
| MIX CHOICES: | When we want to use JDBC API within our Web Application, we have to follow the ordered steps as follow:   1. Load Driver. 2. Open Connection. 3. Create Statement and Assign Parameters (If any). 4. Execute Query (or Update). |

|  |  |
| --- | --- |
| QN=6 | How many HTTP Methods are there? |
| a. | 2 |
| b. | 3 |
| c. | 4 |
| d. | 5 |
| e. | 6 |
| f. | 7 |
| ANSWER: | F |
| MARK: | 0.2 |
| UNIT: | Chap 3 |
| LO: |  |
| MIX CHOICES: | There are **7** HTTP Methods:   1. GET. 2. POST. 3. HEAD. 4. OPTIONS. 5. PUTS. 6. DELETE. 7. TRACE. |

|  |  |
| --- | --- |
| QN=7 | When will we use the RequestDispatcher in a Servlet? |
| a. | We always use RequestDispatcher in our Servlets. |
| b. | When we want to create a new Request. |
| c. | When we want to forward the current Request Object and Response Object to another resource for processing but don’t want to create a new Request. |
| d. | We can’t use RequestDispatcher in a Servlet. |
| e. | When we want to redirect the client to another resource. |
| f. | When we want to create a new Request to process the Client information. |
| ANSWER: | C |
| MARK: | 0.2 |
| UNIT: | Chap 3 |
| LO: |  |
| MIX CHOICES: | We use RequestDispatcher when we want to forward the request to another resource for processing but still remain the Request Object’s existed time for a long time by preventing Response Message being returned. |

|  |  |
| --- | --- |
| QN=8 | What is the output of a JSP page that contains the following code snippet on the second hits?  ***<% int x = 0; %>***  ***<% x++; %>***  ***<%= x %>*** |
| a. | 0 |
| b. | 1 |
| c. | 2 |
| d. | 3 |
| e. | 4 |
| f. | 500 HTTP Internal Error |
| ANSWER: | B |
| MARK: | 0.2 |
| UNIT: | Chap 4 |
| LO: |  |
| MIX CHOICES: | The *x* variable is declared within the Scriptlets so that it is a local variable within the *\_jspservice()* method, which means that it will be initialized every time the JSP page is requested and destroyed every time the JSP page responses to the Client. So that, on any hit, the out of the JSP page with the above code snippet will be **2**. |

|  |  |
| --- | --- |
| QN=9 | Which of the following is the most correct? |
| a. | We have to use RequestDispatcher forward() method for URL Rewriting. |
| b. | The Hidden Form Field Mechanism actually is a technique to remind the Container about client information by sending extra parameters to the server without the awareness of the client. |
| c. | The URL Rewriting Mechanism is a dynamic way to keep track of user’s information at the server. |
| d. | Cookies are actually files at server. |
| e. | The Session config such as session timeout can be found in the context.xml file. |
| f. | In order to remove an Attribute in Session scope, we use the method *invalidate().* |
| ANSWER: | B |
| MARK: | 0.2 |
| UNIT: | Chap 5 |
| LO: |  |
| MIX CHOICES: | (A) We must use response.sendRedirect() for URL Rewriting in order not to duplicate parameters.  (B) Hidden Form Field mechanism uses a *hidden* text box or form control with different value (alternative field) in order to send extra parameters to the Server to remind Container about client’s information. (correct)  (C) The URL Rewriting Mechanism is a static way to keep track of user’s information at the server because it is mostly hard-coded by the developers.  (D) Cookies are files at client.  (E) The Session config such as session timeout is configured in the web.xml file or Servlet but not context.xml file.  (F) To remove an Attribute in Session scope, we just use the *removeAttribute()* method. The *invalidate()* will destroy the entire Session scope. |

|  |  |
| --- | --- |
| QN=10 | What will be the output of the following code snippet within a JSP page?  ***\${“5” + “3”} = ${“5” + “3”}*** |
| a. | 53 = 53 |
| b. | 53 = 8 |
| c. | “5” + “3” = “5” + “3” |
| d. | 0 |
| e. | “5” + “3” = 8 |
| f. | ${“5” + “3”} = 8 |
| ANSWER: | F |
| MARK: | 0.2 |
| UNIT: | Chap 6 |
| LO: |  |
| MIX CHOICES: | The ***\*** is a escape character so that the string behind it, which is ***${“5” + “3”}*** ***=*** will be printed into Response Object without any doubt. The Expression ***${“5” + “3”}*** is Expression Language, “5” and “3” will be converted to 5 and 3 numerically and the operator **+** will be executed. |

|  |  |
| --- | --- |
| QN=11 | Which of the following is not an implicit object to use in Expression Language? |
| a. | param |
| b. | sessionScope |
| c. | cookies |
| d. | requestScope |
| e. | paramValues |
| f. | All is implicit objects in Expression Language |
| ANSWER: | C |
| MARK: | 0.2 |
| UNIT: | Chap 6 |
| LO: |  |
| MIX CHOICES: | The implicit objects to use in Expression Language are:   * pageContext. * param. * paramValues. * initParam. * **cookie**. * header. * headerValues. * pageScope. * requestScope. * sessionScope. * applicationScope. |

|  |  |
| --- | --- |
| QN=12 | Which of the following is similar to the below code snippet in a JSP page?  ***<c:set var="subject" value="PRJ"/>*** |
| a. | <% page.setAttribute(“subject”, “PRJ”); %> |
| b. | <% pageContext.setAttribute(“PRJ”, “subject”); %> |
| c. | <% page.setAttribute(“PRJ”, “subject”); %> |
| d. | <% session.setAttribute(“subject”, “PRJ”); %> |
| e. | <% pageContext.setAttribute(“subject”, “PRJ”); %> |
| f. | <% sessionScope.setAttribute(“subject”, “PRJ”); %> |
| ANSWER: | E |
| MARK: | 0.2 |
| UNIT: | Chap 7 |
| LO: |  |
| MIX CHOICES: | - The <c:set /> JSTL is similar to *setAttribute()* method of some scopes.  - The default scope for <c:set /> is page scope.  - The page scope can be accessed through the implicit object pageContext.  - The **var** attribute in the <c:set /> is the identifier and the name of the attribute within scope. The **value** attribute in the <c:set /> obviously the value of the attribute in the scope.  So that, the correct syntax is:  ***<% pageContext.setAttribute(“subject”, “PRJ”); %>*** |

|  |  |
| --- | --- |
| QN=13 | Which of the following is correct? |
| a. | A Filter must implement Filter interface. |
| b. | A Filter is declared in web.xml within a <servlet> and <servlet-mapping> tag. |
| c. | To apply a Filter to all requests to web application, we configure the url-pattern as **/** in web.xml. |
| d. | By default, the Filter is applied to all the forwards. |
| e. | All the code that we want to be processed in Filter is put in the *init()* method. |
| f. | All are incorrect |
| ANSWER: | A |
| MARK: | 0.2 |
| UNIT: | Chap 9 |
| LO: |  |
| MIX CHOICES: | (A) A Filter must implement Filter interface. (Correct)  (B) A Filter is declared in web.xml within a <filter> and <filter-mapping> tag.  (C) To apply a Filter to all requests to web application, we configure the url-pattern as **/\*** in web.xml.  (D) By default, the Filter is applied to all Request. The forwards are applied only when we configure with the tag <dispatcher>FORWARD</dispatcher> within the <filter-mapping> tag in web.xml.  (E) All the code that we want to be processed in Filter is put in the *doFilter()* method. |

|  |  |
| --- | --- |
| QN=14 | What would be the best directory in which to store a supporting JAR file for a web application? All directories below begin from the context root. |
| a. | \WEB-INF\lib |
| b. | \jars |
| c. | \WEB-INF |
| d. | \WEB-INF\classes |
| e. | \META-INF |
| f. | All are incorrect |
| ANSWER: | A |
| MARK: | 0.2 |
| UNIT: | Chap 1 |
| LO: |  |
| MIX CHOICES: | - All the supporting libraries that we use in our web application is stored in the \WEB-INF\lib directory.  - The \WEB-INF\classes stores our compile Java class code.  - The \META-INF stores the context.xml file. |

|  |  |
| --- | --- |
| QN=15 | Consider the following form:  ***<form action=”LoginServlet”>***  ***Username: <input type=”text” name=”username” /><br/>***  ***Password: <input type=”password” name=”password” /> <br/>***  ***<input type=”submit” name="btnAction” value=”Login” />***  ***</form>***  Which of the following is correct when the user click **Login** button without entering username and password into the form controls? |
| a. | Both parameters with the name *username* and *password* are not existed. |
| b. | There is a parameter at the server with the name of *btnAction* and its value is *Login*. |
| c. | The browser is sending 2 parameters to the server. |
| d. | Only the parameter with the name *btnAction* is sent to the server. |
| e. | The browser will not send request to server until client enter both username and password into the text boxes. |
| f. | All are incorrect. |
| ANSWER: | B |
| MARK: | 0.2 |
| UNIT: | Chap 3 |
| LO: |  |
| MIX CHOICES: | (A) With the input type of *text*, if the user didn’t enter anything, a request is created with the parameter that is not null, which means the it still exists at server, and its value is an empty string.  (B) The button is sent to server with a parameter whose name is button’s name attribute – *btnAction* and value is button’s value attribute – *Login*. (correct)  (C) The form includes 3 controls with 2 text boxes and 1 buttons so that there are 3 parameters sent to server.  (D) The form includes 3 controls with 2 text boxes and 1 buttons so that there are 3 parameters sent to server.  (E) With the input type of *text*, if the user didn’t enter anything, a request is created with the parameter that is not null, which means the it still exists at server, and its value is an empty string. |

|  |  |
| --- | --- |
| QN=16 | Which of the given jsp statement is equivalent to ***<%= userBean.getAge() %>*** ?  Assume that ***userBean.getAge()*** returns an integer. |
| a. | <jsp:getProperty name=“userBean” property= “Age” /> |
| b. | <jsp:getProperty name=“userBean” method= “getAge” /> |
| c. | <jsp:getProperty name=“userBean” property= “age” /> |
| d. | <jsp:getProperty name=“userBean” property= “getAge” /> |
| e. | <jsp:setProperty name=“userBean” property= “Age” /> |
| f. | <jsp:setProperty name=“userBean” property= “age” /> |
| ANSWER: | C |
| MARK: | 0.2 |
| UNIT: | Chap 4 |
| LO: |  |
| MIX CHOICES: | - The code ***<%= userBean.getAge() %>*** is similar to the <jsp:getProperty /> action.  - <jsp:getProperty /> syntax:  ***<jsp:getProperty name=“<identifier>” property=“<attrName>” />***  So that, the correct action is: ***<jsp:getProperty name=“userBean” property= “age” />*** |

|  |  |
| --- | --- |
| QN=17 | Identify the parent element of <session-timeout> element in the web.xml file. |
| a. | <session-configuration> |
| b. | <session config> |
| c. | <session-config> |
| d. | <webapp> |
| e. | <web-app> |
| f. | <session configuration> |
| ANSWER: | C |
| MARK: | 0.2 |
| UNIT: | Chap 5 |
| LO: |  |
| MIX CHOICES: | To configure the session timeout in the web.xml file, we have this code snippet:  ***<web-app>***  ***…***  ***<session-config>***  ***…***  ***<session-timeout>***  ***60***  ***</session-timeout>***  ***</session-config>***  ***</web-app>*** |

|  |  |
| --- | --- |
| QN=18 | A JavaBeans component has the following field:  ***private boolean enabled;***  Which pairs of method declarations follow the JavaBeans standard for accessing this field? |
| a. | public boolean setEnabled (boolean enabled)  public boolean getEnabled () |
| b. | public void setEnabled (boolean enabled)  public boolean getEnabled () |
| c. | public Boolean setEnabled (boolean enabled)  public boolean isEnabled() |
| d. | public void setEnabled ()  public void isEnabled () |
| e. | public void setEnabled (boolean enabled)  public boolean isEnabled () |
| f. | public boolean setEnabled (boolean enabled)  public boolean isEnabled () |
| ANSWER: | E |
| MARK: | 0.2 |
| UNIT: | Chap 4 |
| LO: |  |
| MIX CHOICES: | - The accessor methods are used along with property name with the first character of each word in upper case => **getEnabled** and **setEnabled**  - The dataType of properties **enabled** is *boolean* then the getter method is **isEnabled**instead of **getEnabled**.  - So the correct pairs of methods are:  ***public void setEnabled (boolean enabled)***  ***public boolean isEnabled ()*** |

|  |  |
| --- | --- |
| QN=19 | Consider that we want to read the cookies that had been saved. Which of the following is suitable? |
| a. | response.getCookies() |
| b. | response.getCookie() |
| c. | request.getCookies() |
| d. | request.getCookie() |
| e. | response.getValue() |
| f. | request.getValue() |
| ANSWER: | C |
| MARK: | 0.2 |
| UNIT: | Chap 5 |
| LO: |  |
| MIX CHOICES: | - We use the **request** for any actions in terms of reading Cookies. And the methods to read all cookies is: ***request.getCookies()*** which returns an array of instance of **Cookie**.  - We use the **response** when we want to do any action in terms of writing Cookies. |

|  |  |
| --- | --- |
| QN=20 | What is the output of the following code snippet of a JSP page?  ***${empty “”} <br/>***  ***${empty “sometext”}*** |
| a. | true  false |
| b. | false  true |
| c. | false |
| d. | true |
| e. | true  true |
| f. | false  false |
| ANSWER: | A |
| MARK: | 0.2 |
| UNIT: | Chap 4 |
| LO: |  |
| MIX CHOICES: | - The *empty* operator in EL is used to test if the string is empty, the object is null or the object is not null but its length is 0.  - The **“”** is an empty string so the first line returns **true**.  - The **“sometext”** is not null and its length is 8 which is greater than 0, so the second line returns **false**.  - So that the output is:  ***true***  ***false*** |