Q. P. Code: 4D129 SE Comp

(1)

(3 Hours) [Total Marks: 80]

N.B. : (1) Question **No. 1** is compulsory.

(2) Solve any **three** questions out of remaining questions.

(3) Assume suitable data if required.

1. **(a) List and explain common cross browser compatibility issues.** **5**

Ans: Margins/ borders inconsistencies, image rendering, image border, font rendering, fonts, font size, and expanding box.

(b).Differentiate between GET and POST **5**

Ans:

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

(c)Explain different stages of an ASP.NET web page. **5**

Ans: Page request, start, page initialization, load, postback event handling, rendering, and unload.

(d)How is type casting done in PHP? **5**

Ans: In PHP, data type does not require explicit definition in variable declaration and the data type conversion is done implicitly. However, there are two methods for explicitly converting the data type of a variable. One way to type cast is to put the variable that is to be cast after the name of the chosen type enclosed within parenthesis. For example, *variable = (target_type)variable*

2. **(a)Write HTML code to draw the following table:** **10**

Table I: Cricket Analysis

Country	Matches			Net RR
	Played	Won	Lose	
INDIA	30	28	2	+0.394
PAKISTAN	30	03	27	−1.09

AUSTRALIA	36	10	16	+0.12
SRILANKA	25	5	20	−0.80

Ans:*Hints:* Use tags like <table>, <tr>, <th>, <td>, <b>, <align=top>, <rowspan>

(b)What is JQUERY? Write a program to validate a form using JQUERY. 10

Ans:JQUERY is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML. It is free, open-source software using the permissive MIT License. Web analysis indicates that it is the most widely deployed JavaScript library by a large margin.

Program (with o/p) to validate a form using JQUERY

```
<form action="/registration" method="POST">
  <p>
    User name (4 characters minimum, only alphanumeric characters):
    <input data-validation="length alphanumeric" data-validation-length="min4">
  </p>
  <p>
    Year (yyyy-mm-dd):
    <input data-validation="date" data-validation-format="yyyy-mm-dd">
  </p>
  <p>
    Website:
    <input data-validation="url">
  </p>
  <p>
    <input type="submit">
  </p>
</form>
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery-form-validator/2.3.26/jquery.form-
validator.min.js"></script>
<script>
  $.validate({
    lang: 'es'
  });
</script>
```

User name (4 characters minimum, only alphanumeric characters):

Year (yyyy-mm-dd):

Website:

Submit

3. (a)Explain servlet lifecycle in detail. 10

Ans:

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.

03

The init() Method

The init method is called only once. It is called only when the servlet is created, and not called for any user requests afterwards. So, it is used for one-time initializations, just as with the init method of applets.

The servlet is normally created when a user first invokes a URL corresponding to the servlet, but you can also specify that the servlet be loaded when the server is first started.

When a user invokes a servlet, a single instance of each servlet gets created, with each user request resulting in a new thread that is handed off to doGet or doPost as appropriate. The init() method simply creates or loads some data that will be used throughout the life of the servlet.

The init method definition looks like this –

```
public void init() throws ServletException {
   // Initialization code...
}
```

The service() Method

The service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client(browsers) and to write the formatted response back to the client.

Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

Here is the signature of this method –

```
public void service(ServletRequest request, ServletResponse response)
   throws ServletException, IOException {
}
```

The service () method is called by the container and service method invokes doGet, doPost, doPut, doDelete, etc. methods as appropriate. So you have nothing to do with service() method but you override either doGet() or doPost() depending on what type of request you receive from the client.

The doGet() and doPost() are most frequently used methods with in each service request. Here is the signature of these two methods.

The doGet() Method

A GET request results from a normal request for a URL or from an HTML form that has no METHOD specified and it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // Servlet code
}
```

The doPost() Method

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    // Servlet code
}
```

The destroy() Method

The destroy() method is called only once at the end of the life cycle of a servlet. This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

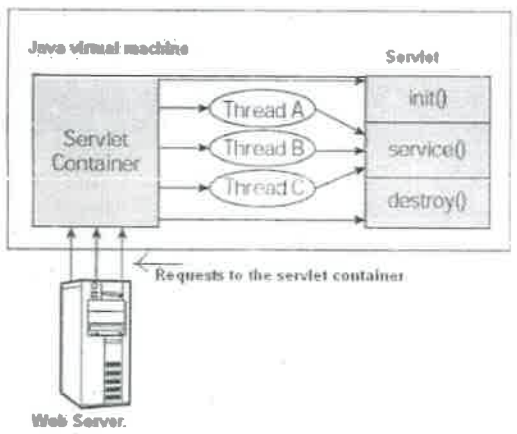
After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this –

```
public void destroy() {
  // Finalization code...
}
```

Architecture Diagram

The following figure depicts a typical servlet life-cycle scenario.

- First the HTTP requests coming to the server are delegated to the servlet container.
- The servlet container loads the servlet before invoking the service() method.
- Then the servlet container handles multiple requests by spawning multiple threads, each thread executing the service() method of a single instance of the servlet.



(b)Give details aboutJDBC connectivitythrough an example. **10**

Example JDBC program

1. open connection to DB
2. execute SQL statement
3. process result
4. close connection to DB

```
1  import java.sql.*;
2  class SelectProducts
3  {
4   public static void main(java.lang.String[ ] args)
5   {
6    try
7     {
8         Class.forName( "COM.ibm.db2.jdbc.app.DB2Driver" );                                          1
9         Connection con = DriverManager.getConnection( "jdbc:db2:TEST", "db2admin", "db2admin" );
10        Statement statement = con.createStatement( );                                               2
11        ResultSet rs = statement.executeQuery("SELECT NAME, PRICE FROM PRODUCT" );
12        while ( rs.next( ) )
13         {
14           String name = rs.getString( "NAME" );                                                    3
15           float price = rs.getFloat( "PRICE" );
16           System.out.println("Name: "+name+", price: "+price);
17         }
18        statement.close( );                                                                         4
19        con.close( );
20    }
21   catch( Exception e ) { e.printStackTrace( ); }
22  }
23 }
```

4. **(a) Explain any 5string manipulation functions in PHP with examples. 10**

Ans: String manipulation functions: bin2hex(), chr(), chunk_split(), convert_cyr_str_string(), count_chars(), echo(), fprint(), etc.

String-manipulation functions

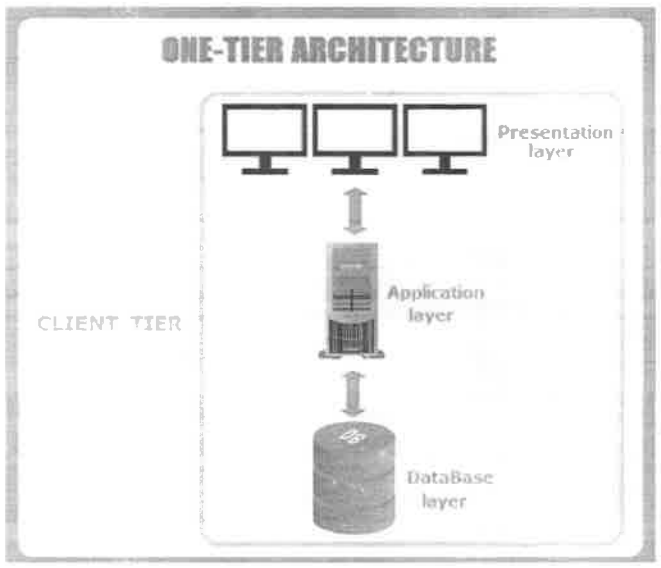
- PHP provides huge range of string-manipulation functions:
  - addcslashes -- Quote string with slashes in a C style
  - addslashes – Quote string with slashes
  - count_chars – Return information about characters used in a string
  - echo – Output one or more strings.
  - explode -- Split a string by string
  - implode – Join array elements with a string
  - join – Join array elements with a string
  - ltrim -- Strip whitespace from the beginning of a string
  - md5 – Calculate the md5 hash of a string
  - strpos – Find position of first occurrence of a string

(b)**Discuss various web system architectures.** **10**

Ans:

One Tier Architecture:

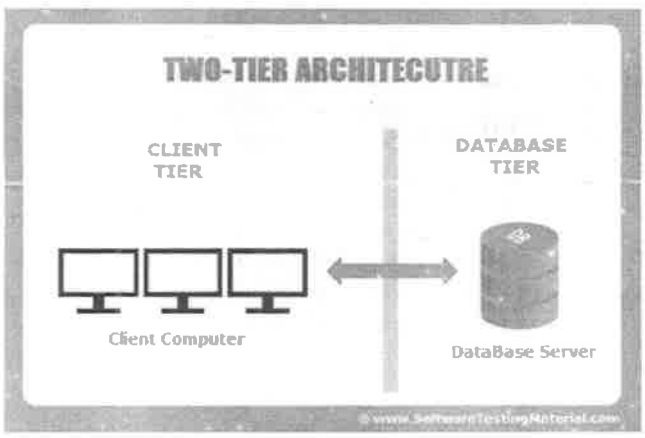
One Tier application AKA Standalone application



One tier architecture has all the layers such as Presentation, Business, Data Access layers in a single software package. Applications which handles all the three tiers such as MP3 player, MS Office are come under one tier application. The data is stored in the local system or a shared drive.

Two-Tier Architecture:

Two Tier application AKA Client-Server application



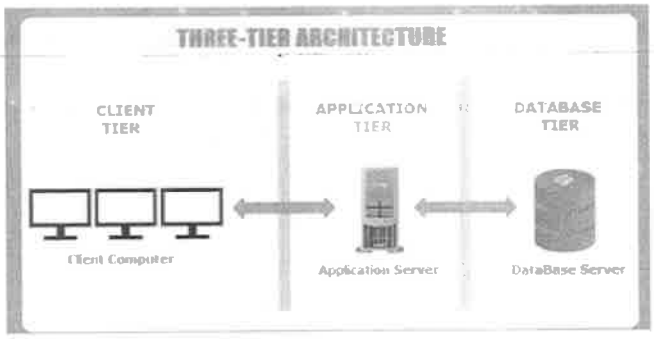
The Two-tier architecture is divided into two parts:

1. Client Application (Client Tier)
2. Database (Data Tier)

Client system handles both Presentation and Application layers and Server system handles Database layer. It is also known as client server application. The communication takes place between the Client and the Server. Client system sends the request to the Server system and the Server system processes the request and sends back the data to the Client System

Three-Tier Architecture:

Three Tier application AKA Web Based application



The Three-tier architecture is divided into three parts:

1. Presentation layer (Client Tier)
2. Application layer (Business Tier)
2. Database layer (Data Tier)

Client system handles Presentation layer, Application server handles Application layer and Server system handles Database layer.

Note: Another layer is N-Tier application. N-Tier application AKA Distributed application. It is similar to three tier architecture but number of application servers are increased and represented in individual tiers in order to distributed the business logic so that the logic will be distributed.

5. **(a) Explain cookies, its attributes and uses in detail.** **10**

Ans: A cookie is a small file containing information, which a server embeds on a user's computer. Each time the same computer requests for a web page from a server, the server refers to the created cookie for displaying the requested web page. The size of a cookie is dependent on a browser.

Uses: It is used to store the username and password information on a computer so that you need not enter this information each time you visit a website.

Attributes: name attribute, expires attribute, domain attribute, path attribute, secure attribute.

(b)Write an ASP.NET program to insert a new record in Student database. **10**

6

Default.aspx

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-trans
itional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:GridView ID="GridView1" runat="server"
OnRowCommand="gridview1_RowCommand">
</asp:GridView>

 <br />

<asp:Button ID="Button2" runat="server" Text="Insert" />

  
<br />

Roll_No :                 

<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<br />
<br />
        Student_Name  :    
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
<br />
<br />
        Phone_Number :    
<asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
<br />

<br />

Address :

<asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
<br />
<br />

        Comparing value :  
<asp:TextBox ID="TextBox5" runat="server"></asp:TextBox>
<br />
<br />
</div>
</form>
</body>
</html>
```

10

Default.aspx.vb

```
Imports System.Data
Imports System.Data.SqlClient
Partial Class _Default
    Inherits System.Web.UI.Page
    Dim mycom As SqlCommand
    Dim mycon As SqlConnection
    Dim ds As DataSet
    Dim adp As SqlDataAdapter
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Button1.Click
mycon = New SqlConnection("server=webserver1;user id =sa;password=sa;database=VJW")
mycon.Open()
mycom = New SqlCommand("select * from t1", mycon)
ds = New DataSet
adp = New SqlDataAdapter(mycom)
adp.Fill(ds, "t1")
        GridView1.DataSource = ds
GridView1.DataBind()
mycon.Close()
    End Sub
    Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button2.Click
mycon = New SqlConnection("server=webserver1;user id =sa;password=sa;database=VJW")
mycon.Open()
mycom = New SqlCommand("insert into t1 (Roll_No,Student_Name,Phone_No,Address) values ('" & TextBox1.Text &
 "','" & TextBox2.Text & "','" & TextBox3.Text & "','" & TextBox4.Text & "')", mycon)
mycom.ExecuteNonQuery()
mycon.Close()
    End Sub

    Protected Sub GridView1_RowCommand(ByVal sender As Object, ByVal e As System.Web.UI.WebControls.GridView
CommandEventArgs) Handles GridView1.RowCommand
        If e.CommandName = "Select" Then
            Dim index As Integer = Convert.ToInt32(e.CommandArgument)
            Dim r As GridViewRow = GridView1.Rows(index)
Session.Add("Roll_No", r.Cells(1).Text)
Session.Add("Student_Name", r.Cells(2).Text)
Session.Add("Phone_No", r.Cells(3).Text)
Session.Add("Address", r.Cells(4).Text)
        End If
        TextBox1.Text = Session("Roll_No")
        TextBox2.Text = Session("Student_Name")
        TextBox3.Text = Session("Phone_No")
        TextBox4.Text = Session("Address")
    End Sub
End Class
```

6. Write short notes on**(any four)**: **20**

(i)Session tracking

Session Tracking

An application can be configured to use either cookies or query strings to track sessions. To configure an application *not* to use cookies to track sessions, you need to modify the `SessionState` section of the `Web.Config` file. The session collection contains many methods and attributes. The five main attributes used to configure session state management in ASP.NET are listed here:

- `Mode`: Specifies the persistence mode used to store session state. There are four modes to choose from: `Off`, `Inproc`, `StateServer`, and `SQLServer`.
- `Timeout`: Specifies the number of minutes of idle time before the session shuts down.
- `ConnectionString`: Required only if `Mode` is set to `StateServer`. `ConnectionString` specifies the port as well as the name or address of the server where session state is stored.
- `SQLConnectionString`: Required when `Mode` is set to `SQLServer`. It specifies the connection string needed to connect to a database server.
- `Cookieless`: A Boolean value that indicates whether the application should use cookies or munged URLs to track sessions.

(ii)Commonly used ADO.NET objects

Ans: Connection object, DataAdapter object, Command object, DataSet object, and DataTable object.

(iii)Use of RSS web feeds

Ans: RSS is one of the web feed formats which keeps you updated of the changes occurring in selected website. A web feed provides regularly updated content of a web page. It is a document (mostly XML-based) comprising content along with web links. Web feeds are designed in such a way that is machine readable (computer) instead of human readable. RSS also contains XML document that frequently scans the website's content for any update and then displays it to the user through feed. The update this is sent contains a headline and small amount of text. The text may be a summary or link to the whole text.

(iv)CSS text properties

Ans: text-indent, text-shadow, text-wrap, word-break, word-spacing, and word-wrap.

12

(v)Different types of XSL elements

Contents

xsl:apply-imports

The **xsl:apply-imports** element is used in conjunction with imported stylesheets. There are no attributes. The element may contain zero of more xsl:with-param elements (as permitted in XSLT 1.1)

At run-time, there must be a *current template*. A current template is established when a template is activated as a result of a call on xsl:apply-templates. Calling xsl:call-template does not change the current template. Calling xsl:for-each does not (as the XSLT standard says it should) cause the current template to become null.

The effect is to search for a template that matches the current node and that is defined in a stylesheet that was imported (directly or indirectly, possibly via xsl:include) from the stylesheet containing the current template, and whose mode matches the current mode. If there is such a template, it is activated using the current node. If not, the call on xsl:apply-imports has no effect.

It is not possible to supply parameters to a template invoked using xsl:apply-imports.

xsl:apply-templates

The xsl:apply-templates element causes navigation from the current element, usually but not necessarily to process its children. Each selected node is processed using the best-match **xsl:template** defined for that node.

The **xsl:apply-templates** element takes an optional attribute, **mode**, which identifies the processing mode. If this attribute is present, only templates with a matching **mode** parameter will be considered when searching for the rule to apply to the selected elements.

It also takes an optional attribute, **select**.

If the **select** attribute is omitted, apply-templates causes all the immediate children of the current node to be processed; that is, child elements and character content, in the order in which it appears. Character content must be processed by a template whose match pattern will be something like "*|text()". Child elements similarly are processed using the appropriate template, selected according to the rules given below under xsl:template.

If the **select** attribute is *included*, it must be a *node set expression* which identifies the nodes to be processed. All nodes selected by the expression are processed.

xsl:attribute

The **xsl:attribute** element is used to add an attribute value to an **xsl:element** element or general formatting element, or to an element created using **xsl:copy**. The attribute must be output immediately after the element, with no intervening character data. The name of the attribute is indicated by the **name** attribute and the value by the content of the xsl:attribute element.

The attribute name is interpreted as an *attribute value template*, so it may contain string expressions within curly braces. The full syntax of string expressions is given in XPath Expression Syntax.

For example, the following code creates a <FONT> element with several attributes:

```
<xsl:element name="FONT">
    <xsl:attribute name="SIZE">+2</xsl:attribute>
    <xsl:attribute name="FACE">Courier New</xsl:attribute>
Some output text
</xsl:element>
```

There are two main uses for the **xsl:attribute** element:

- It is the only way to set attributes on an element generated dynamically using xsl:element
- It allows attributes of a literal result element to be calculated using xsl:value-of.

xsl:attribute-set

The **xsl:attribute-set** element is used to declare a named collection of attributes, which will often be used together to define an output style. It is declared at the top level (subordinate to xsl:stylesheet).

An attribute-set contains a collection of xsl:attribute elements.

The attributes in an attribute-set can be used in several ways:

- They can be added to a literal result element by specifying xsl:use-attribute-sets in the list of attributes for the element. The value is a **space-separated** list of attribute-set names. Attributes specified explicitly on the literal result element, or added using xsl:attribute, override any that are specified in the attribute-set definition.
- They can be added to an element created using xsl:element, by specifying use-attribute-sets in the list of attributes for the xsl:element element. The value is a space-separated list of attribute-set names. Attributes specified explicitly on the literal result element, or added using xsl:attribute, override any that are specified in the attribute-set definition.
- One attribute set can be based on another by specifying use-attribute-sets in the list of attributes for the xsl:attribute-set element. Again, attributes defined explicitly in the attribute set override any that are included implicitly from another attribute set.

Attribute sets named in the xsl:use-attribute-sets or use-attribute-sets attribute are applied in the order given; if the same attribute is generated more than once, the later value always takes precedence.

xsl:call-template

The xsl:call-**template** element is used to invoke a named template.

The **name** attribute is mandatory and must match the name defined on an xsl:template element.

Saxon supports an additional attribute **saxon:allow-avt**. If this is present and is set to the value "yes", then the **name** attribute may be written as an attribute value template, allowing the called template to be decided at run-time. The string result of evaluating the attribute value template must be a valid QName that identifies a named template somewhere in the stylesheet.

Parameters to the called template may be defined using xsl:with-param elements nested within the xsl:call-template element.

The context of the called template (for example the current node and current node list) is the same as that for the calling template; however the variables defined in the calling template are not accessible in the called template.