65083

(3 Hours)[MODEL ANSWERS] **[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) Discuss any five CSS text properties. **5**

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

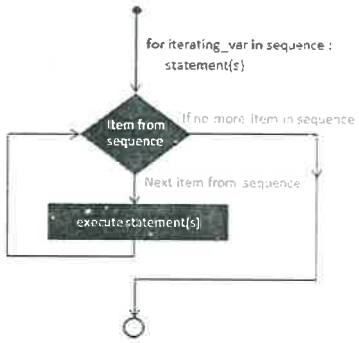
(b) Explain the *for* loop used in PHP. **5**

Ans: Loops in PHP are used to execute the same block of code a specified number of times.

- **for** – loops through a block of code a specified number of times.

The for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.



Syntax

```
for (initialization; condition; increment){
   code to be executed;
}
```

The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

Example

The following example makes five iterations and changes the assigned value of two variables on each pass of the loop –

Live Demo

```
<html>
   <body>

      <?php
         $a = 0;
         $b = 0;

         for( $i = 0; $i<5; $i++ ) {
            $a += 10;
            $b += 5;
         }

         echo ("At the end of the loop a = $a and b = $b" );
      ?>

   </body>
</html>
```

This will produce the following result –

```
At the end of the loop a = 50 and b = 25
```

(c) Explain the functions of a web server. **5**

Ans:

1) The primary function of a web server is to deliver web pages on requests from clients using HTTP.
2) It can be referred as hardware (computer) or the software (computer application).
3) Hosts websites/web application to serve www.
4) Found embedded in devices like printers, routers, web-cams to serve local network.
5) Able to map URL to a local file system resource (static requests) as well as internal or external program name (dynamic requests).

(d) 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.

2. (a)Write a program that shows a message as Good Morning, Good Afternoon or Good Night according to the current time by using the *if* statement in JavaScript. **10**

Ans)

Write a program in JavaScript to demonstrate the use of the if statement in JavaScript.

A program to demonstrate the use of the if statement in JavaScript is as follows:

```
<SCRIPT>
var x="";
var time=new Date().getHours();
if (time<12)
  {
  x="Good Morning";
  }
else if(time<17)
{
x="Good Afternoon"
}
else
{
x="Good Night"
}
alert(x);
</SCRIPT>
```

The preceding program shows a message as Good Morning, Good Afternoon or Good Night according to the current time.

(b) Write HTML code to draw the following table: **10**

Time Table					
Hours	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**
	Science	Maths	Science	Maths	Arts
	Social	History	English	Social	Sports
	Lunch				

	Science	Maths	Science	Maths	Project
	Social	History	English	Social	

Ans:

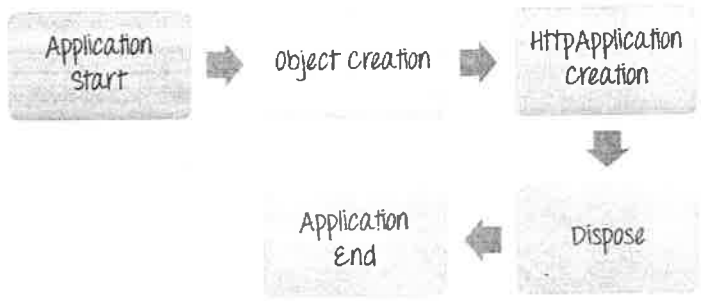
<HTML>

```
<BODY>
<Font Face = "Verdana">
<TABLE style="border-collapse: collapse;" Border = "1" Cellpadding = "5" Cellspacing = "5">
<TR>
<TH Colspan = "6" Align = "center">Time Table</TH>
</TR>
<TR>
<TH Rowspan = "6">Hours</TH>
<TH>Mon</TH>
<TH>Tue</TH>
<TH>Wed</TH>
<TH>Thu</TH>
<TH>Fri</TH>
</TR>
<TR>
<TD>Science</TD
><TD>Maths</TD>
<TD>Science</TD>
<TD>Maths</TD>
<TD>Arts</TD>
</TR>
<TR>
<TD>Social</TD>
<TD>History</TD>
<TD>English</TD>
<TD>Social</TD>
<TD>Sports</TD>
</TR>
<TR>
<TH Colspan = "5" Align = "center">Lunch</TH>
</TR>
<TR>
<TD>Science</TD>
<TD>Maths</TD>
<TD>Science</TD>
<TD>Maths</TD>
<TD Rowspan = "2">Project</TD>
</TR>
<TR>
<TD>Social</TD>
<TD>History</TD>
<TD>English</TD>
<TD>Social</TD>
</TR>
</TABLE>
```

```
</Font>
</BODY>
</HTML>
```

3. (a) Explain ASP.NET application lifecycle. **10**

Ans: When an ASP.Net application is launched, there are series of steps which are carried out. These series of steps make up the lifecycle of the application.



ASP.Net Lifecycle

1) Application Start - The life cycle of an ASP.NET application starts when a request is made by a user. This request is to the Web server for the ASP.Net Application. This happens when the first user normally goes to the home page for the application for the first time. During this time, there is a method called Application_start which is executed by the web server. Usually, in this method, all global variables are set to their default values.

2) Object creation - The next stage is the creation of the HttpContext, HttpRequest & HttpResponse by the web server. The HttpContext is just the container for the HttpRequest and HttpResponse objects. The HttpRequest object contains information about the current request, including cookies and browser information. The HttpResponse object contains the response that is sent to the client.

3) HttpApplication creation - This object is created by the web server. It is this object that is used to process each subsequent request sent to the application. For example, let's assume we have 2 web applications. One is a shopping cart application, and the other is a news website. For each application, we would have 2 HttpApplication objects created. Any further requests to each website would be processed by each HttpApplication respectively.

4) Dispose - This event is called before the application instance is destroyed. During this time, one can use this method to manually release any unmanaged resources.

5) Application End - This is the final part of the application. In this part, the application is finally unloaded from memory.

(b)Describe *string manipulation* and *date and time* built-in functions in PHP **10**

Ans:

The PHP Date() Function

The PHP `date()` function convert a timestamp to a more readable date and time.

The computer stores dates and times in a format called UNIX Timestamp, which measures time as a number of seconds since the beginning of the Unix epoch (midnight Greenwich Mean Time on January 1, 1970 i.e. January 1, 1970 00:00:00 GMT).

Since this is an impractical format for humans to read, PHP converts a timestamp to a format that is readable to humans and dates from your notation into a timestamp the computer understands. The syntax of the PHP `date()` function can be given with:

```
date(format, timestamp)
```

The *format* parameter in the `date()` function is required which specifies the format of returned date and time. However the *timestamp* is an optional parameter. If not included then current date and time will be used. The following statement displays today's date:

Example

Run this code »

```
<?php
$today = date("d/m/Y");
echo $today;
?>
```

The PHP time() Function

The `time()` function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1 1970 00:00:00 GMT).

Example

Run this code »

```
<?php
$t=time();
echo($t . "<br>");
?>
```

The above example produce the following output:

```
1394003959
```

We can convert this timestamp to a human readable date through passing it to the previously introduce `date()` function.

Example

Run this code »

```
<?php
$timestamp = 1394800238;
echo(date("F d, Y h:i:s", $timestamp));
?>
```

The above example produce the following output:

```
March 05, 2014 07:10:18
```

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

4. (a) How a database can be connected using ADO.Net? Explain with a suitable example. **10**

Ans)

ADO.NET is the new database technology of the .NET (Dot Net) platform, and it builds on **Microsoft ActiveX® Data Objects** (ADO).

ADO is a language-neutral object model that is the keystone of Microsoft's Universal Data Access strategy.

ADO.NET is an integral part of the .NET Compact Framework, providing access to relational data, XML documents, and application data. **ADO.NET** supports a variety of development needs. You can create database-client applications and middle-tier business objects used by applications, tools, languages or Internet browsers.

ADO.NET defines `DataSet` and `DataTable` objects which are optimized for moving disconnected sets of data across intranets and Internets, including through firewalls. It also includes the traditional `Connection` and `Command` objects, as well as an object called a `DataReader` that resembles a forward-only, read-only ADO `recordset`. If you create a new application, your application requires some form of data access most of the time.

ADO.NET provides data access services in the Microsoft **.NET** platform.

You can use **ADO.NET** to access data by using the new .NET Framework data providers which are:

- Data Provider for SQL Server (`System.Data.SqlClient`).
- Data Provider for OLEDB (`System.Data.OleDb`).
- Data Provider for ODBC (`System.Data.Odbc`).
- Data Provider for Oracle (`System.Data.OracleClient`).

ADO.NET is a set of classes that expose data access services to the .NET developer. The **ADO.NET** classes are found in *System.Data.dll* and are integrated with the XML classes in *System.Xml.dll*.

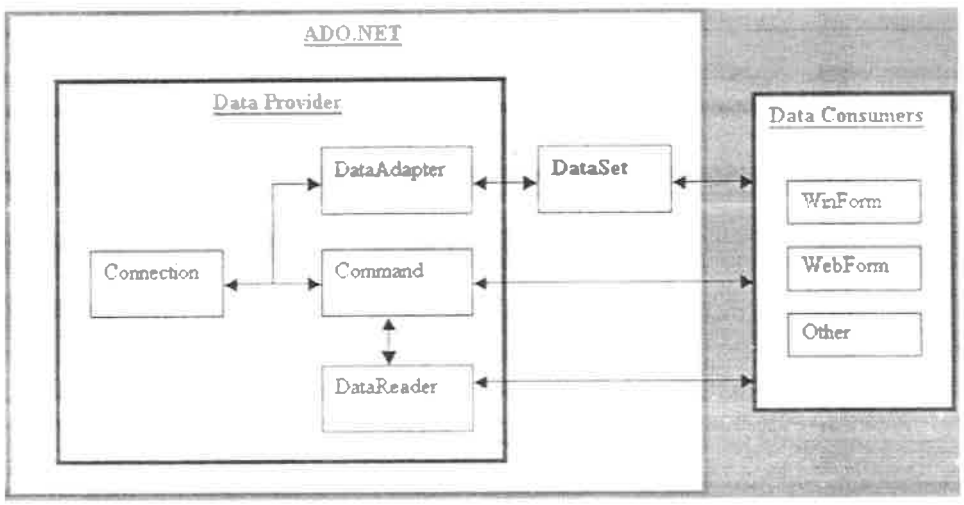
- There are two central components of **ADO.NET** classes: the `DataSet`, and the **.NET** Framework **Data Provider**.

Data Provider is a set of components including:

- the **Connection object** (`SqlConnection`, `OleDbConnection`, `OdbcConnection`, `OracleConnection`)
- the **Command object** (`SqlCommand`, `OleDbCommand`, `OdbcCommand`, `OracleCommand`)
- the **DataReader object** (`SqlDataReader`, `OleDbDataReader`, `OdbcDataReader`, `OracleDataReader`)
- and the **DataAdapter object** (`SqlDataAdapter`, `OleDbDataAdapter`, `OdbcDataAdapter`, `OracleDataAdapter`).

`DataSet` object represents a disconnected cache of data which is made up of `DataTables` and `DataRelations` that represent the result of the command.

The ADO.NET Object Model



Before working with a database, you have to add (here) the `OleDb` .NET Data Provider namespace, by placing the following at the start of your code module:

```
using System.Data.OleDb;
```

Similarly for the `SqlClient` .NET Data Provider namespace:

```
using System.Data.SqlClient;
```

The `using` statement should be positioned first in your code.

Now, we have to declare a connection string pointing to a MS Access database "*PersonDatabase.mdb*".

Hide Copy Code

```
public string
  conString=@"Provider=Microsoft.Jet.OLEDB.4.0;" +
            @" DataSource=..\\..\\PersonDatabase.mdb";
```

The database should be in the specified path, otherwise you should change the path accordingly.

The next step is to create an `OleDbConnection` object. We pass then the connection string to this `OleDbConnection` object. You can code now to create a new **ADO.NET** Connection object in order to connect to an OLE DB provider database.

Hide Copy Code

```
OleDbConnection con = new OleDbConnection(conString);
```

You can also explicitly reference declared objects if you don't mind typing a lot.

Hide Copy Code

```
System.Data.OleDb.OleDbConnection con =
  new System.Data.OleDb.OleDbConnection(conString);
```

Here is the code snippet for connection to a database:

```
//using declaration for OLE DB
using System.Data.OleDb;
//specify the ConnectionString property
public string conString=
  @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=..\\..\\PersonDatabase.mdb";
//Initializes a new instance of the OleDbConnection
OleDbConnection con = new OleDbConnection(conString);
// open the database connection with the property settings
// specified by the ConnectionString "conString"
con.Open();
```

In many earlier applications, the tendency was to open a connection when you start the application and not close the connection until the application terminates. It is an expensive and time-consuming operation to open and close a database connection. Most databases have a limit on the number of concurrent connections that they allow.

For example: each connection consumes a certain amount of resources on the database server and these resources are **not** infinite. Most modern OLE DB providers (including SQL Server provider) implement *connection pooling*. if you create database connections, they are held in a pool. When you want a connection for an application, the OLE DB provider extracts the next available connection from the pool. When your application closes the connection, it returns to the pool and makes itself available for the next application that wants a connection.

This means that opening and closing a database connection is no longer an expensive operation. If you close a connection, it does **not** mean you disconnect from the database. It just returns the connection to the pool. If you open a connection, it means it's simply a matter of obtaining an already open connection from the pool. It's recommended in many **ADO.NET** books **not** to keep the connections longer than you need to. Therefore, you should:

- Open a connection when you need it, and
- Close it as soon as you have finished with it.

For example: here is another way to get a connection to a database:

```
// setup the global SqlConnection object and constr in your class
  private SqlConnection con = null;
  private string constr ="Integrated Security=SSPI;" +
  "Initial Catalog=Northwind;" +
  "Data Source=SCNY\\MYSQLSERVER;";

  private void fnGetConnection()
  {
    try
    {
       // setup the database connection
       con = new SqlConnection(constr);
       con.Open();
    }catch (Exception ex) {
       MessageBox.Show("Error in connection : "+ex.Message);
    }finally {
       // dispose of open objects
       if (con != null)
         con.Close();
    } //finally
  }
```

For example: you want to open the connection, fill the `DataSet`, and close the connection. If the connection fails, you want to get the error message.

```
try
{
  con.Open();
  dadapter.Fill(dataset1);
  con.Close();
} catch (Exception ex) {
   MessageBox.Show("Error in retrieving data: " + ex.Message);
}
```

For example: if you want to save the data you changed, then you just open the connection, update the data, and close the connection and accept the changes. If it fails, display an error message, reject the changes, and close the connection.

Hide Copy Code

```
try
{
  DataSet changes = dataset.GetChanges();
  con.Open();
  datapter.Update(changes);
  con.Close();
  dataset1.AcceptChanges();
}catch (Exception ex) {
  MessageBox.Show("ErrorR: " + ex.Message);
  dataset1.RejectChanges();
  con.Close();
}
```

(b) Explain different types of XSL elements. **10**

Ans:

Contents

Standard XSL Elements	xsl:element	xsl:param
xsl:apply-imports	xsl:fallback	xsl:processing-instruction
xsl:apply-templates	xsl:for-each	xsl:preserve-space
xsl:attribute	xsl:if	xsl:script
xsl:attribute-set	xsl:include	xsl:sort
xsl:call-template	xsl:import	xsl:strip-space
xsl:choose	xsl:key	xsl:stylesheet
xsl:comment	xsl:message	xsl:template
xsl:copy	xsl:namespace-alias	xsl:text
xsl:copy-of	xsl:number	xsl:value-of
xsl:decimal-format	xsl:otherwise	xsl:variable
xsl:document	xsl:output	xsl:when
		xsl:with-param
		Literal Result Elements

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 set 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.

5. (a) What is JQUERY? Illustrate the use of JQUERY for form validation. **10**

Ans:

- JQuery is a light, "write less, do more", Javascript library.
- The purpose of JQuery is to make it much easier to use JavaScript on your website. It simplifies HTML document traversing, event handling, animating and AJAX interactions for rapid web development.
- The jQuery library is a single JavaScript file, and you reference it with the HTML

```
<script> tag (notice that the <script> tag should be inside the <head> section):
<head>
<script src="jquery-1.11.3.min.js"></script>
</head>
```

jQuery Syntax

- The jQuery syntax is tailor made for selecting HTML elements and performing some action on the element(s).
- Basic syntax is: $(selector).action()

1. A $ sign to define/access jQuery
2. A (selector) to "query (or find)" HTML elements
3. A jQuery action() to be performed on the element(s)

Examples:

4. $(this).hide() - hides the current element.
5. $("p").hide() - hides all elements.
6. $(".test").hide() - hides all elements with class="test".
7. $("#test").hide() - hides the element with id="test".

jQuery Selectors

- jQuery selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
- All selectors in jQuery start with the dollar sign and parentheses: $().

The element Selector

- The jQuery element selector selects elements based on the element name. • You can select all elements on a page like this: S("p")
- Example: When a user clicks on a button, all elements will be hidden:
- Example

```
\$(document).ready(function(){
\$("button").click(function(){
$("p").hide();
            });
        });

**The #id Selector**

- The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
- An id should be unique within a page, so you should use the #id selector when you want to find a s
ingle, unique element.
- To find an element with a specific id, write a hash character, followed by the id of the HTML elem
ent:\$("#test")
- Example: When a user clicks on a button, the element with id="test" will be hidden
- Example

        \$(document).ready(function(){
        \$("button").click(function(){
        \$("#test").hide();
            });
        });

**The .class Selector**

- The jQuery class selector finds elements with a specific class.
- To find elements with a specific class, write a period character, followed by the name of the clas
s:$(".test")
```

- Example: When a user clicks on a button, the elements with class="test" will be hidden
- Example

```
\$(document).ready(function(){
\$("button").click(function(){
\$(".test").hide();
    });
});
```

JQuery for Form Validation:

Step 1 – Include the latest version of the jQuery Library.

//hosted by Microsoft Ajax CDN

```
<script src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.9/jquery.validate.min.js"></script>
```

Step 2 – Download the jQuery Validation Plugin.

Step 3 – Add the following JavaScript validation rules to your webpage (or include as separate js include).

The code below contains the input field validation rules for the form and also includes a direct submit handler (works for mutiple forms on same page).

```
(function($,W,D)
    {
    var JQUERY4U = {};

        JQUERY4U.UTIL =
        {
    setupFormValidation: function()
            {
                //form validation rules
                $("#register-form").validate({
rules: {
firstname: "required",
lastname: "required",
email: {
required: true,
email: true
                    },
password: {
required: true,
minlength: 5
                    },
agree: "required"
                },
messages: {
firstname: "Please enter your firstname",
lastname: "Please enter your lastname",
password: {
required: "Please provide a password",
minlength: "Your password must be at least 5 characters long"
                    },
email: "Please enter a valid email address",
agree: "Please accept our policy"
                },
submitHandler: function(form) {
form.submit();
                }
            });
        }
    }

    //when the dom has loaded setup form validation rules
    $(D).ready(function($) {
JQUERY4U.UTIL.setupFormValidation();
    });

})(jQuery, window, document);
```

Step 4 – Add the HTML for the form and some styles.

The input fields "name" attribute is important as it maps directly to the validation rules.

```
<!-- HTML form for validation demo -->
<form action="" method="post" id="register-form" novalidate="novalidate">

<h2>User Registration</h2>

<div id="form-content">
<fieldset>

<div class="fieldgroup">
<label for="firstname">First Name</label>
<input type="text" name="firstname"/>
</div>

<div class="fieldgroup">
<label for="lastname">Last Name</label>
<input type="text" name="lastname"/>
</div>

<div class="fieldgroup">
<label for="email">Email</label>
<input type="text" name="email"/>
</div>

<div class="fieldgroup">
<label for="password">Password</label>
<input type="password" name="password"/>
</div>

<div class="fieldgroup">
<p class="right">By clicking register you agree to our <a target="_blank" href="/policy">policy</a>.</p>
<input type="submit" value="Register" class="submit"/>
</div>

</fieldset>
</div>

<div class="fieldgroup">
<p>Already registered? <a href="/login">Sign in</a>.</p>
</div>
</form>
/* example styles for validation form demo */
#register-form {
background: url("form-fieldset.gif") repeat-x scroll left bottom #FBFBFB;
border: 1px solid #DFDCDC;
border-radius: 15px 15px 15px 15px;
display: inline-block;
margin-bottom: 30px;
margin-left: 20px;
margin-top: 10px;
```

```
padding: 15px 50px 10px;
width: 350px;
}

#register-form .fieldgroup {
background: url("form-divider.gif") repeat-x scroll left top transparent;
display: inline-block;
padding: 8px 10px;
width: 340px;
}

#register-form .fieldgroup label {
float: left;
padding: 15px 0 0;
text-align: right;
width: 110px;
}

#register-form .fieldgroup input, #register-form .fieldgroup textarea, #register-form .fieldgroup select {
float: right;
margin: 10px 0;
height: 25px;
}

#register-form .button {
padding: 10px;
width: 120px;
height: 40px !important;
}
#register-form .fieldgroup label.error {
color: #F83A84;
display: inline-block;
margin: 4px 0 5px 125px;
padding: 0;
text-align: left;
width: 220px;
}
```

(b) 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.

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.

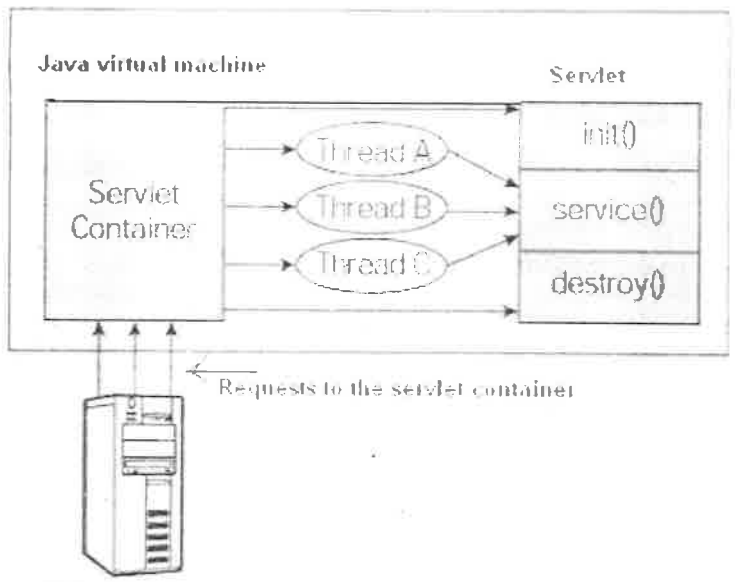
After the destroy() method is called, the servlet object is marked for garbage collection. The destroy method definition looks like this –

```
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.



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

(i) Three-tier architecture of web application

Ans:

- In 3 tier architecture, there are 3 components: Client PC, An Application server and A Database Server.
- The work of server is distributed among application server and database server.
- Application server has the required communication functions.
- The data required by the business logic exists in database server.
- The required data is returned to public servers and then to client PC.

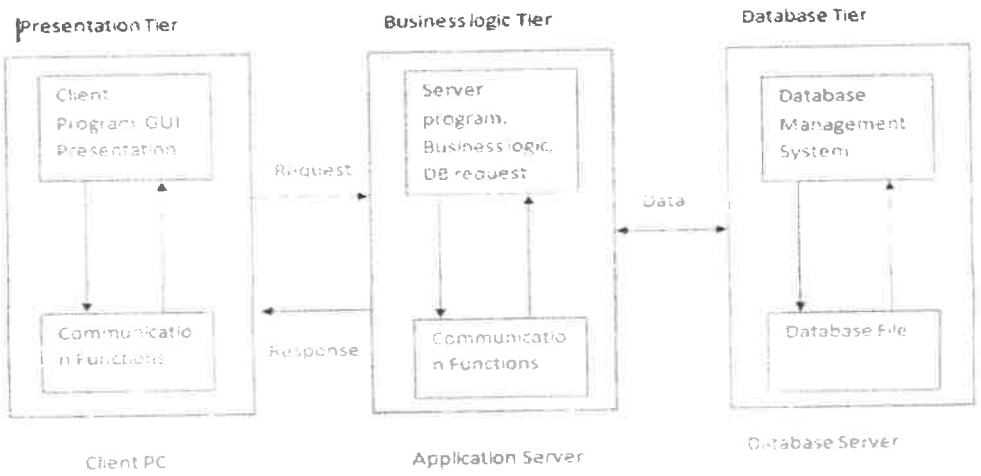


Figure 1.1: 3 tier architecture

Advantages:

- Improved Data Integrity
- High Degree of Flexibility in deployment platform and configurations
- Improved security
- High Performance and persistent objects
- Architecture is scalable, adding users and resources in future would be easy
- Maintenance and modifications can be done effectively
- Code and data reusability can be achieved

Disadvantages:

- 3 tier architecture is complex compared to 1 tier and 2 tier
- Cost of network maintenance and deployment is greater than 1 tier and 2 tier

(ii) Website design issues

Ans)

- **Simplicity**: Considering from a layman user, web site should be build to make it easier for user to use it. Web site might have information, pictures, effects, etc. This makes the website design enormous and it should be avoided
- **Identity**: Every web site should have its identity. Websites are categorized on various factors, considering their objective, category of users, etc.
- **Consistency**: The contents of web application should be consistent. There are various properties of a content of web site like text formatting, font style, colour scheme, Navigation mechanism, graphics design etc. It should be identical in all the pages.
- **Robustness**: The failure or missing of any functionality in a web site disappoints a user. User expects robust contents and functions of web application.
- **Navigability**: The navigation helps to move from one page to another. It should not be complex to make it difficult for lay man user. A developer should deploy predictive navigation functionality.
- **Visual Appeal**: A Web site should be able to attract more number of users. This can be achieved by implementing dynamic and attractive graphics in a web site. The factors include Look and feel, interface layout, colour co-ordination, balance of text, graphics and other various media.
- **Compatibility**: A web site should be build in a such a way that it should be implemented in various environment and configurations such as different browsers.

(iii) PHP and MySQL database connectivity

Ans:

PHP Connect to MySQL

PHP 5 and later can work with a MySQL database using:

- **MySQLi extension** (the "i" stands for improved)
- **PDO (PHP Data Objects)**

Earlier versions of PHP used the MySQL extension. However, this extension was deprecated in 2012.

Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".

Both MySQLi and PDO have their advantages:

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.

So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

Both are object-oriented, but MySQLi also offers a procedural API.

Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

MySQL Examples in Both MySQLi and PDO Syntax

In this, and in the following chapters we demonstrate three ways of working with PHP and MySQL:

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

MySQLi Installation

For Linux and Windows: The MySQLi extension is automatically installed in most cases, when php5 mysql package is installed.

For installation details, go to: http://php.net/manual/en/mysqli.installation.php

PDO Installation

For installation details, go to: http://php.net/manual/en/pdo.installation.php

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

PHP is an amazing and popular language!

Note on the object-oriented example above: $connect_error was broken until PHP 5.2.9 and 5.3.0. If you need to ensure compatibility with PHP versions prior to 5.2.9 and 5.3.0, use the following code instead:

```
// Check connection
if (mysqli_connect_error()) {
   die("Database connection failed: " . mysqli_connect_error());
}
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection Failed: " . mysqli_connect_error());
}
echo "Connected successfully";
```

Example (PDO)

```
<?php
$servername = "localhost";

$username = "username";
$password = "password";

try {
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username,
$password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Note: In the PDO example above we have also **specified a database (myDB).** PDO require a valid database to connect to. If no database is specified, an exception is thrown.

Tip: A great benefit of PDO is that it has an exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

Close the Connection

The connection will be closed automatically when the script ends. To close the connection before, use the following:

Example (MySQLi Object-Oriented)

```
$conn->close();
```

Example (MySQLi Procedural)

```
mysqli_close($conn);
```

Example (PDO)

```
$conn = null;
```

(iv) 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.

(v) 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.
