Q.20 Which directive initializes an AngularJS application ?

a ng-init
b ng-app
c ngSrc
d ng-model

Answers Keys for Multiple Choice Questions :

Q.1	b	Q.2	c	Q.3	b	Q.4	c	Q.5	c
Q.6	d	Q.7	a	Q.8	d	Q.9	c	Q.10	a
Q.11	b	Q.12	d	Q.13	d	Q.14	b	Q.15	c
Q.16	a	Q.17	c	Q.18	c	Q.19	d	Q.20	b

UNIT- III

3 Java Servlets and XML

Syllabus

Servlet : Servlet architecture overview, A "Hello World" servlet, Servlets generating dynamic content, Servlet life cycle, parameter data, sessions, cookies, URL rewriting, other Servlet capabilities, data storage, Servlets concurrency, databases (MySQL) and Java Servlets.
XML : XML documents and vocabularies, XML declaration, XML Namespaces, DOM based XML processing, transforming XML documents, DTD : Schema, elements, attributes.
AJAX : Introduction, Working of AJAX.

Contents

3.1 Introduction to Server Side Programming
3.2 Servlet Architecture Overview
3.3 A "Hello World" Servlet ..... **May-19,** ..... Marks 5
3.4 Servlets Generating Dynamic Content
3.5 Servlet Life Cycle ..... **March-20,** ..... Marks 5
3.6 Servlet API
3.7 Parameter Data
3.8 Sessions
3.9 Cookies ..... **Dec.-19,** ..... Marks 5
3.10 URL Rewriting
3.11 Other Servlet Capabilities
3.12 Data Storage
3.13 Servlets and Concurrency
3.14 Databases (MySQL) and Java Servlets
3.15 XML Documents and Vocabularies ..... **Dec.-18, 19, May-19,** ..... Marks 5
3.16 XML Declaration

3.17 Rules
3.18 XML Namespaces
3.19 Document Type Definition (DTD) ..... May-18, ... Marks 5
3.20 Schema ..... May-18, ... Marks 5
3.21 DOM based XML Processing
3.22 Transforming XML Documents ..... May-19, March-20, ... Marks 5
3.23 Introduction
3.24 Working of AJAX
3.25 XMLHttpRequest Object
3.26 Coding AJAX Script ..... Dec.-18, 19, May-18,19 ..... Marks 8
3.27 Multiple Choice Questions

Part I : Servlet

3.1 Introduction to Server Side Programming

- Server side technology deals with the data submitted by the client.
- Server side program basically **handles the request** made by the client. This program then **prepares the response** for the client.
- The server side programming also act as a link between backend database and front end client side programming.
- Various server side technologies include – servlet, JSP, ASP.NET, PHP and so on.

Difference between client side and server side scripting

Sr. No.	Server side scripting	Client side scripting
1.	The server side scripting is used to create the web pages that provide some services.	The client side scripting is used to create the web pages as a request or response to server. These pages are displayed to the user on web browser.
2.	These scripts generally run on web servers.	These scripts generally run on web browser.
3.	A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser.	The processing of these scripts takes place on the end users computer. The source code is transferred from the web server to the users computer over the internet and run directly in the browser.
4.	Uses : Processing of user request, accessing to databases.	Uses : Making interactive web pages, for interacting with temporary storages such as cookies or local storage, sending request to server and getting the response and displaying that response in web browser.
5.	Example : PHP, ASP.NET, nearly all the programming languages including C++, Java and C#	Example : HTML, CSS, JavaScript(primarily)

3.2 Servlet Architecture Overview

- Servlets are basically the Java programs that run on server. These are the programs that are requested by the XHTML documents and are displayed in the browser window as a response to the request.

- The servlet class is instantiated when web server begins the execution.
- The execution of servlet is managed by **servlet container,** such as Tomcat.
- The servlet container is used in java for dynamically generate the web pages on the server side. Therefore the servlet container is the part of a web server that interacts with the servlet for handling the dynamic web pages from the client.
- Servlets are most commonly used with HTTP (i.e. Hyper Text Transfer Protocol) hence sometimes servlets are also called as "HTTP Servlet".
- The main purpose of servlets is to add up the functionality to a web server.

Working of How Servlet Works ?

- Before learning the actual servlet programming it is very important to understand how servlet works.

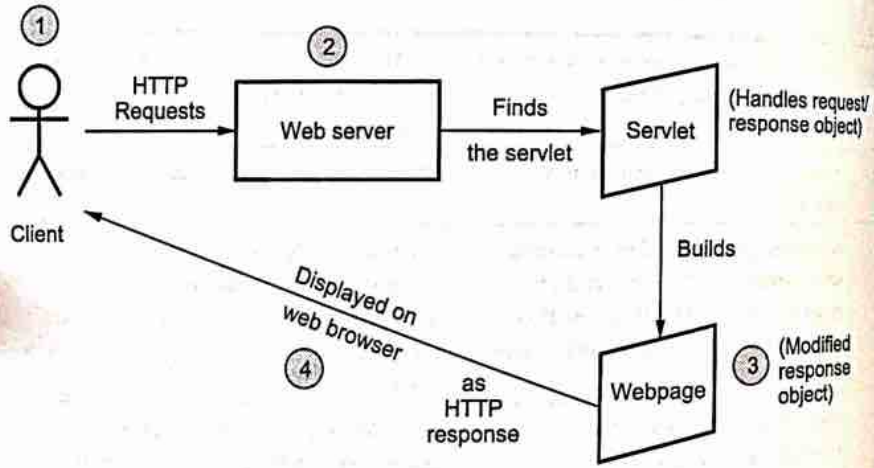


Fig. 3.2.1 How servlet works ?

1. When a client make a request for some servlet, he/she actually uses the Web **browser** in which request is written as a URL.
2. The web browser then sends this request to **Web server.** The web server first finds the requested servlet.
3. The obtained servlet gathers the relevant information in order to satisfy the client's request and builds a **web page** accordingly.
4. This web page is then **displayed** to the client. Thus the request made by the client gets satisfied by the servlets.

3.2.1 Need and Advantages

- Servlets can process and store the data submitted by an HTML form.
- Servlets are useful for providing the dynamic contents. For example retrieving and updating the databases.
- Servlets can be used in the cookies. Cookies are small programs which can make use of the information submitted on currently accessed web pages.
- Similarly servlets are used in session tracking. The session tracking are the useful programs that keeps track of all previously accessed the web pages.

Advantages

- The servlets are very **efficient** in their performance and get executed in the address space of the belonging web server.
- The servlets are **platform independent** and can be executed on different web servers.
- The servlets working is based on **Request-Response.** Any HTML form can take the user input and can forward this input to the servlet. The servlets are then responsible to communicate with the back-end database and manipulate the required business logic. These servlets embedded on the web servers using **Servlets API.**
- Servlets provide a way to generate the **dynamic document.** For instance : A servlet can display the information of current user logged in, his logging time, his last access, total number of access he made so far and so on.
- **Multiple users** can keep a co-ordination for some application among themselves using servlets.
- Using servlets **multiple requests** can be synchronized and then can be concurrently handled.

Disadvantages

- Servlet is a mixture of java skills and web related HTML skills. Hence it is difficult to write servlets.
- It slows down the application execution.
- The java runtime environment is required on the server, if you want to execute servlet application.

3.3 A "Hello World" Servlet

SPPU : May-19, Marks 5

- When we write a servlet program, it is necessary to - i) Either implement **Servlet** interface or ii) Extend a class that implements **Servlet** interface.

- While implementing **Servlet** interface we must include **javax.servlet** package. Hence the first line in out servlet program must be,

```
import javax.servlet.*;
```

- **GenericServlet** class is a predefined implementation of **Servlet** interface. Hence we can extend GenericServlet class in our servlet program. Similarly, the **HttpServlet** class is a child class of **GenericServlet** class, hence we can extend this class as well while writing the servlet program.
- Hence following are the two ways by which we can write servlet program

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Test extends GenericServlet
{
  //body of servlet
}
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Test extends HttpServlet
{
  //body of servlet
}
```

- The servlet gets the request from the client for some service. The servlet then processes the request and sends the response back to the client. In order to handle these issues **HttpServletRequest** and **HttpServletResponse** are used in servlet program.
- These requests are handled with the help of some methods that are popularly known as **methods of HttpServlet.** These methods are as follows,

Method	Purpose
doGet	This method handles the HTTP GET request
doPost	This method handles the HTTP POST request
doPut	This method handles the HTTP Put request.
doDelete	This method handles the DELETE request.

The doGet and doPost methods

- The **doGet** method requests the data from the source.
- The **doPost** method submits the processed data to the source.
- The protocol of **doGet** method is as follows,

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException,IOException
```

- The **ServletException** and **IOException** are thrown to handle the Servlet problems gracefully.

- The **HttpServletRequest** request : Contain the client request made by client.
- The **HttpServletResponse** response : Contains the response made by servlet back to the client.
- The protocol of doPost method is same as doGet method. It is as follows -

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException,IOException
```

- The GET request is more efficient than the POST request.
- The GET request is less secure than the POST request.

How to Write Servlet Program ?

Open notepad and write the first servlet code to display greeting messages. It is as follows,

FirstServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FirstServlet extends HttpServlet
{
public void doGet(HttpServletRequest request,HttpServletResponse response)
throws IOException,ServletException
{
 response.setContentType("text/html");
 PrintWriter out=response.getWriter();
 out.println("<html>");
 out.println("<head>");
 out.println("<title>My First Servlet</title>");
 out.println("<body>");
 out.println("<h1>Hello How are U?</h1>");
 out.println("<h2>I am enjoying this Servlet Application</h2>");
 out.println("<h3>See You later!</h3>");
 out.println("</body>");
 out.println("</html>");
}
}
```

Program Explanation :

- In the above program, we have imported following files,

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

- Out of these files **java.io** package is useful for taking care of I/O operations.
- The **javax.servlet** and **javax.servlet.http** are important packages containing the classes and **interfaces** that are required for the operation of servlets. The most commonly used interface from **javax.servlet** package is **Servlet**. Similarly most commonly used class in this package is **GenericServlet.** The **ServletRequest** and **ServletResponse** are another two commonly used interfaces defined in **javax.servlet** package.
- In the **javax.servlet.http** package **HttpServletRequest** and **HttpServletResponse** are two commonly used interfaces. The **HttpServletRequest** enables the servlet to read data from the HTTP request and **HttpServletResponse** enables the servlet to write data to HTTP response. The **cookie** and **HttpServlet** are two commonly used classes that are defined in this package.
- We have given class name **FirstServlet** which should be derived from the class **HttpServlet.** (Sometimes we can derive our class from **GenericServlet**).
- Then we have defined **doGet method** to which the HTTP request and response are passed as parameters. The commonly used basic exceptions for the servlets are **IOException** and **ServletException.**
- The MIME type is specified as using the **setContentType()** method. This method sets the content type for the HTTP response to type. In this method "text/html" is specified as the MIME type. This means that the browser should interpret the contents as the HTML source code.
- Then an output stream is created using **PrintWriter().** The **getWriter()** method is used for obtaining the output stream. Anything written to this stream is sent to the client as a response. Hence using the object of output stream 'out', we can write the HTML source code in **println** method as HTTP response.

How to execute Servlet program ?

Step 1 : Compile the above program using the javac command at command prompt.

```
D:\test>javac FirstServlet.java
```

The class file for this program gets generated.

Step 2 : Before running any servlet program, it is necessary to have

1. JDK installed
2. Tomcat installed.
3. Class path for above two packages must be set.

For Tomcat installation, I prefer to install the package XAMPP. The XAMP contains a directory for tomcat. The XAMPP package contains Apache Web server, MySQL, PHP and Perl support. It can work on both Windows and Linux operating System.

Step 3 : Copy the class file generated in Step 1 to the path

C:\xampp\tomcat\webapps\examples\WEB-INF\classes

Step 4 : Go to the directory

C:\xampp\tomcat\webapps\examples\WEB-INF

Open **web.xml** file and edit it as follows

```
<servlet>
  <servlet-name>FirstServlet</servlet-name>
  <servlet-class> FirstServlet </servlet-class>
</servlet>
...
...
...
<servlet-mapping>
   <servlet-name> FirstServlet </servlet-name>
   <url-pattern>/servlet/ FirstServlet </url-pattern>
  </servlet-mapping>
```

The **web.xml** file is popularly known as **deployment descriptor.** This is basically the configuration file that describes how to map the URL of web application to servlet.

```
<servlet>
                                   Internal alias Name
   <servlet-name>FirstServlet</servlet-name>
                                   Actual Name of server
   <servlet-class> FirstServlet </servlet-class>
  </servlet>
...
...
...
                                   Internal alias Name
<servlet-mapping>
   <servlet-name> FirstServlet </servlet-name>
                                   External alias Name
   <url-pattern>/servlet/ FirstServlet </url-pattern>
  </servlet-mapping>
```

The Servlet comes with two alias names, internal and external. The internal name is used by the Tomcat and the external name is given (to be written in <FORM> tag of HTML file) to the client to invoke the Servlet on the server. That is, there exists alias to

alias. All this is for security. Observe, the names are given in two different XML tags, in the web.xml file, to make it difficult for hacking.

To invoke the **FirstServlet** Servlet, the client calls the server with the name **servlet/FirstServlet.**

When **servlet/FirstServlet** call reaches the server, the Tomcat server opens the **web.xml** file to check the deployment particulars. Searches such a <servlet-mapping> tag that matches **servlet/FirstServlet. servlet/FirstServlet** is exchanged with **FirstServlet.**

Then, searches such a <servlet> tag that matches **FirstServlet** and exchanges with **FirstServlet class.** Now the server, loads **FirstServlet** Servlet, executes and sends the output of execution as response to client.

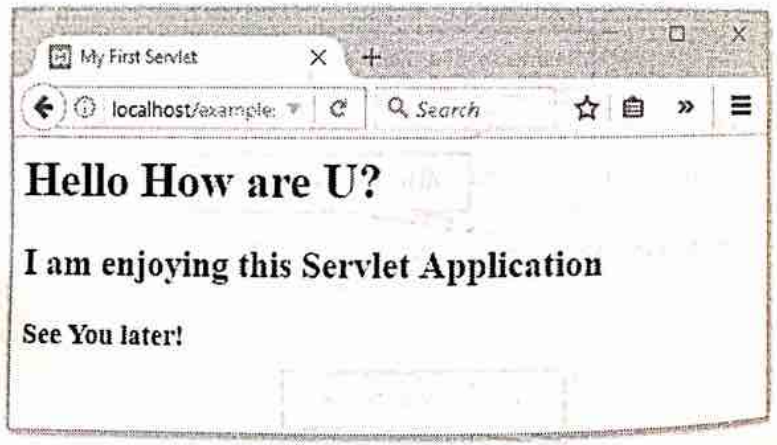
Step 5 : Start tomcat and xampp

Step 6 : Open web browser and type the command

http ://localhost/examples/servlet/FirstServlet

The output will be

Output



Review Question

1. *Write a Java servlet program which will display "welcome to servlet" message.* **SPPU : May-19, Marks 5**

3.4 Servlets Generating Dynamic Content

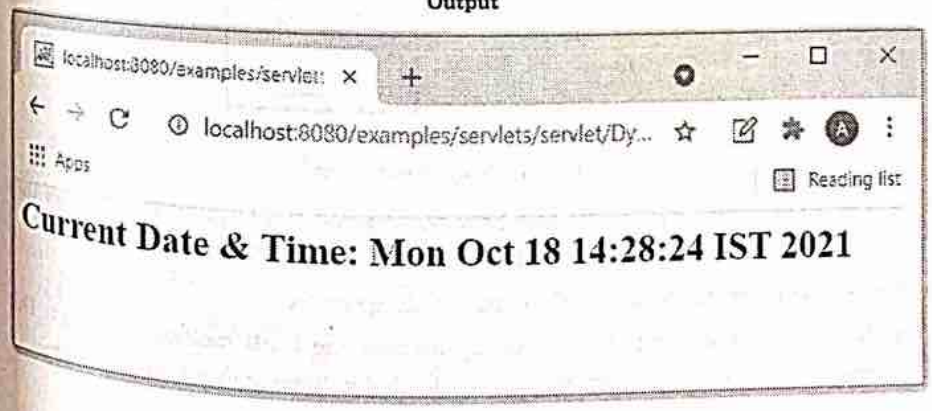
- In modern web technology, the web site must be able to generate dynamic contents as per the requirements. Servlets can generate static, dynamic or interactive web pages.

- Following is a simple servlet program that displays the current date and time. Note that the time is continuously changing and on refreshing the same page, we can note down the changes.

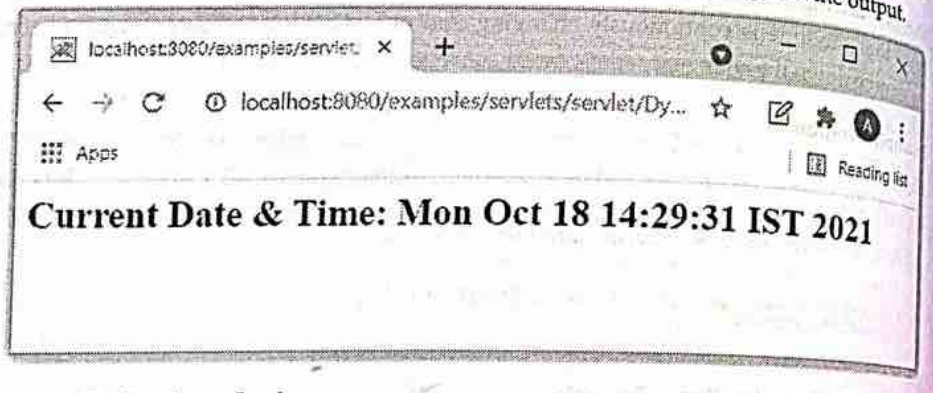
Example Program

```
import java.io.*;
import javax.servlet.*;
public class DynamicServletDemo extends GenericServlet
{
  //implement service()
  public void service(ServletRequest req, ServletResponse res)
  throws IOException, ServletException
  {
    //set response content type
    res.setContentType("text/html");
    //get stream obj
    PrintWriter out = res.getWriter();
    java.util.Date date = new java.util.Date();
    out.println("<h2>"+"Current Date & Time: " +date.toString()+"</h2>");
    //close stream object
    out.close();
  }
}
```

Output



Press F5 and refresh the current page, you will get the some changes in the output.



Program Explanation : In above program,

1. We have used Date() method of java.util class that is intended to display current date and time.
2. The date is displayed using the **date.toString()** method.

3.5 Servlet Life Cycle

SPPU : March-20, Marks 5

In the life cycle of servlet there are three important methods. These methods are,

1. **Init**
2. **Service**
3. **Destroy**

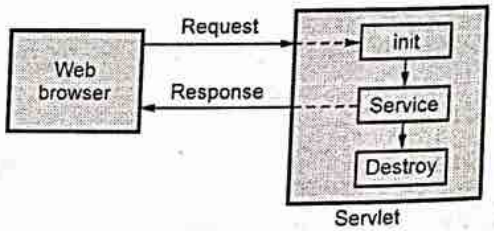


Fig. 3.5.1 Life cycle of servlet

- The client enters the URL in the web browser and makes a request. The browser then generates the HTTP request and sends it to the Web server. (Refer Fig. 3.5.1)
- Web server maps this request to the corresponding servlet.

1. **Init ( ) Method :** The server basically invokes the **init()** method of servlet. This method is called only when the servlet is loaded in the memory for the first time.

Using this method initialization parameters can also be passed to the servlet in order to configure itself.

2. **service ( ) Method :** Server can invoke the service for particular HTTP request using **service()** method. The servlets can then read the data provided by the HTTP request with the help of **service()** method.
3. **destroy ( ) Method :** Finally server unloads the servlet from the memory using the **destroy()** method.

Example Program

Following is a simple servlet in which these three methods viz. init(), service() and destroy() are used -

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class LifeCycle extends GenericServlet
{
  public void init(ServletConfig config) throws ServletException
  {
    System.out.println("init");
  }

  public void service(ServletRequest request, ServletResponse response)
  throws ServletException, IOException
  {
    System.out.println("from service");
    PrintWriter out = response.getWriter();
    out.println("Twinkle Twinkle Little Stars.");
    out.print("How I wonder what you are.");
  }
  public void destroy()
  {
    System.out.println("destroy");
  }
}
```

Program Explanation : In the above program,

1. We have first of all used an **init()** method to which object of the **ServletConfig** interface is passed. This interface allows the servlet to get the initialization parameters.
2. Then we have used the **service()** method to which the ServletRequest and

ServletResponse parameters are passed for making the HTTP request and getting the HTTP response from the sevlet.

3. The output stream is created using **getWriter()** and along this output stream some messages are written. These messages will then be displayed on the web browser when the servlet gets executed.
4. Finally the **destroy()** function is written in order to unload the servlet from the memory.

Review Question

1. *Explain the lifecycle of servlet. Write a simple servlet to print "Hello, Good Day".* **SPPU : March-20, Marks 5**

3.6 Servlet API

There are two packages used to implement the **Servlet**. These packages are,

1) javax.servlet

2) javax.servlet.http

The classes and interfaces defined in these packages help in implementing the servlet. Let us discuss them in detail -

3.6.1 The javax.servlet Package

- This package is used to implement the servlet. Out of many other interfaces and classes defined in this package the most important interface is **Servlet** . All the servlet must implement this interface. The interfaces and classes that are defined in this package are enlisted below -

3.6.1.1 Interfaces

Interface	Description
Servlet	This interface defines all the life cycle methods.
ServletConfig	This interface obtains the initialization parameters.
ServletContext	Using this interface the events can be logged.
ServletRequest	This interface is useful in reading the data from the client request.
ServletResponse	This interface is useful in writing the data to the client response.

1. The Servlet Interface

- This interface defines all the life cycle methods such as **init(),destroy()** and **service()**. All the servlets must implement this interface. Some of the methods provided by this interface are -

Method	Description
void init(ServletConfig s)	This method is called for initialising the servlet. The initialisation parameter is obtained from the ServletConfig interface.
void destroy()	This method is called when the servlet has to be unloaded.
ServletConfig getServletConfig()	This method is used to obtain the initialisation parameters.
void Service(ServletRequest req, ServletResponse res)	This method is used to implement the service that a servlet should provide. The clients request is processed and a response is given.
String getServletInfo()	This method is used to obtain description of the servlet.

2. The ServletConfig Interface

- This interface is used to obtain the initialisation parameters. Various methods that are used by this interface are -

Method	Description
String getServletName()	This method is used to obtain the name of the servlet.
String getInitParameter(String *p*)	This method returns the value of the parameter *p*
Enumeration getinitParamernames()	This method returns the names of initialisation parameters.
ServletContext getServletContext()	This method returns the context for the servlet.

3. The ServletContext Interface

- This is the interface that enables the servlet to log events and access information about their environment. Various methods of this interface are -

Method	Description
Object getAttribute(String *attribute_name*)	The value of the attribute *attribute_name* in the current session is returned.

void setAttribute(String *attribute_name*, object *value*)	The *attribute_name* is passed to the object *value*.
String getServerInfo()	This method returns the information about the server.
String log(String *str*)	Writes the *str* to the servlet log.
String getMimeType(String *file*)	It returns the MIME type of the file.

4. The ServletRequest Interface

- The **ServletRequest** interface helps in gathering the information about the client. Various methods in ServletRequest are -

Method	Description
Object getAttribute(String *attribute_name*)	The value of the attribute *attribute_name* in the current session is returned.
int getContentlength()	It returns the content size of the request.
String getContentType()	It returns the type of the request.
getInputStream()	This method is used to read the binary data from the request.
getProtocol()	Returns the name of the protocol used.
getReader()	This method is useful for reading the text from the request.
getServerName()	It returns the name of the server on which the servlet is running.
int getServerPort()	It returns the port number of the server.

5. The ServletResponse Interface

- This interface helps the servlet to formulate the response for the client's request. Various methods of this interface are -

Method	Description
String getCharacterEncoding()	This method returns the character encoding.
ServletOutputStream getOutputStream()	This method returns outputstream which is used to write the data for responding.

PrintWriter getWriter()	This method is used to write the character data to the response.
void setContentLength(int *size*)	This method sets the length of the content equal to the *size*.
void setContentType(String *Type*)	This methods sets the type of the content

3.6.1.2 Classes

- The **javax.servlet** package contains following classes.

Class	Description
GenericServlet	This class implements the **Servlet** and **ServletConfig** interfaces.
ServletInputStream	This class provides the input stream for reading the client's request.
ServletOutputStream	This class provides the output stream for writing the client's response.
ServletException	This class is used to raise the exception when an error occurs.

1. The GenericServlet class

- This class is useful for implementing the life cycle methods. It implements the **servlet** and **ServletConfig** interfaces. It is called "Generic" because it does not assume that the protocol it will process will be HTTP. If we want to write our own protocol, then we must extend this class.

2. The ServletInputStream class

- The **ServletInputStream** class extends the **InputStream**. This class provides an input stream which is used by a servlet for reading the data from a client request. For reading the bytes from a client the method used is **readLine**.

3. The ServletOutputStream class

- The **ServletOutputStream** class extends the **OutputStream**. This class provides an output stream which is used by a servlet for writing the data for a client response. It also defines **print** and **println** method.

4. The ServletException class

- The javax.servlet defines two servlets ServletException and UnavailableException

3.6.2 The javax.servlet.http Package

- The javax.servlet.http package contains a number of interfaces and classes and their functionality makes it easy to build servlets. These servlets then work with HTTP requests and responses.

3.6.2.1 Interface

- The following table describes the interfaces that are provided in this package

Interface	Description
HttpSession	The session data can be read or written using this interface.
HttpServletRequest	The servlet can read the information from the HTTP request using this interface.
HttpServletResponse	The servlet can write the data to HTTP response using this interface.
HttpSessionBindingListener	This interface tells the object about its binding with the particular session.

1. The HttpSession

- At every HTTP session, the state information is gathered. The servlet can read or write this information using **HTTPSession** interface. This interface is implemented by the server.
- Various methods used by this interface are as given below -

Method	Description
String getId()	This method returns the session ID.
Object getAttribute(String *attribute_name*)	The value of the attribute *attribute_name* in the current session is returned.
Enumeration getAttributeNames()	Returns the attribute names.

2. The HttpServletRequest

- The **HTTPServletRequest** is used to obtain the information from client's HTTP request.

Method	Description
String getMethod()	It returns the HTTP method for the client request.
String getPathInfo()	Returns the path information about the servlet path.
HttpSession getSession()	Returns the current session.

String getHeader(String *fields*)	It returns the value of header fields.
Cookie [ ] getCookies()	It returns the information in the cookies in the request made.
String getAuth Type()	It returns the type of Authentication.

3. The HttpServletResponse

- This **HtttpServletResponse** interface is used to formulate an HTTP response to the client. Various methods are -

Method	Description
void addCookie(Cookie *cookie*)	This method is used to add cookie in the response.
String encodeURL(String url)	This method is used to encode the specified URL.
Boolean containsHeader(String *f*)	This method returns **true** if the response header contains the field *f*.
void sendError(int *code*)	This method sends error code to the client.

4. The HttpSessionBindingListener

- When particular listener gets bound or unbound from a value within **HttpSession** object then this class is required. It extends the **HttpSessionEvent**. Various methods that can be defined by this interface are -

Method	Description
Object getValue()	This function returns the value of bounded or unbounded attribute.
String getName()	This function returns the name being bound or unbound.
HttpSession getSession()	This function returns the session to which the listener can be bound or unbound.

3.6.2.2 Classes

- The following table describes the classes that are provided in this package

Class	Description
Cookie	This class is used to write the cookies.
HttpServlet	It is used when developing servlets that receive and process HTTP requests.
HttpSessionEvent	This class is used to handle the session events.
HttpSessionBindingEvent	When a listener is bound to a value.

1) The Cookie class

- A cookie is a small piece of information that is stored in the client's machine. Sometimes cookies are useful to keep track of the user using the client machine. Various methods used by this class are -

Class	Description
String getValue()	This function returns a value of the cookie.
void setValue(String s)	This function sets the value to the cookie.
String getName()	This function returns the name.

2) The HttpServlet class

- The **HttpServlet** class extends **GenericServlet**. It is used when developing servlets that receive and process HTTP requests. Various methods of this class are -

Class	Description
Void doGet(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP get request.
Void doPost(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP Post request.
Void doPut(HttpServletRequest req, HttpServletResponse res)	This method performs HTTP Put request.
Void service(HttpServletRequest req, HttpServletResponse res)	This method is invoked for processing HTTP request and response.

3) The HttpSessionEvent

- This method encapsulates the session event.

4) HttpSessionBindingEvent

- The **HttpSessionBindingEvent** class extends **HttpSessionEvent**. It is generated when a listener is bound to a value. The **getSession()** method obtains the session to which the listener is being bound or unbound.

3.7 Parameter Data

- Many times we need to pass some information from web browser to Web server. In such case, the HTML document containing the FORM is written which sends the data to the servlet present on the web server.

- To make the form works with Java servlet, we need to specify the following attributes for the <form> tag :
  i) **method="method name":** To send the form data as an HTTP POST request to the server.
  ii) **action="URL/address of the servlet":** Specifies relative URL of the servlet which is responsible for handling data posted from this form.
- The browser uses two methods to pass this information to web server. These methods are GET Method and POST Method.
  i) **GET method :** The GET method sends user information along with ? symbol called query string. For instance

```
http://localhost/hello?user = aaaa&age = 20
```

In servlet, this information is processed using **doGet()** method.

ii) **POST method :** This is the most reliable method of sending user information to the server from HTML form. Servlet handles this request using **doPost** method.

Difference between GET and POST

GET	POST
Using GET request limited amount of information can be sent.	Using POST large amount of information can be sent.
GET request is not secured as information is visible in URL	This is a secured request.
This request is can be bookmarked.	This request can not be bookmarked.
This request is more efficient.	This request is less efficient.

How does servlet read form data ?

Servlet makes use of following three methods to read the data entered by the user on the HTML form

1. **getParameter()** – You call request.getParameter() method to get the value of a form parameter.
2. **getParameterValues()** – Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
3. **getParameterNames()** – Call this method if you want a complete list of all parameters in the current request.

Programming Examples

Example 3.7.1 *Write a HTML that shows the following list :*

C,C++,JAVA,C#

Define a form that contains a select statement and submit button. If the user selects the java and press the submit the web page displays "The selected language is Java". Write a servlet program using HttpServlet and doGet method.

Solution :

- We will write two source files first one is the HTML file named **test.html** in which the list of all the desired languages is displayed along with the submit button. User has to select the language of his choice and then press the submit button.
- This data made by this request will be received by the servlet named **my_choiceservlet.java** which reads the selected parameter and displays the message about the selection made. The source files are as follows -

test.html

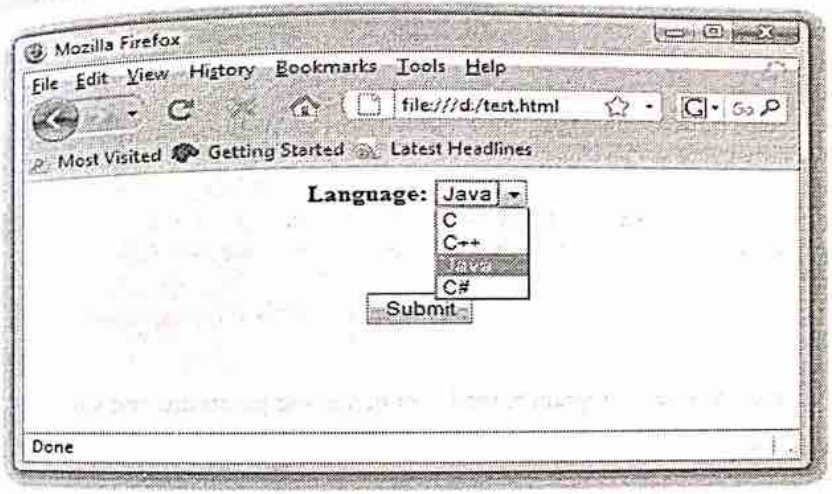
```
<html>
  <body>
  <center>
  <form name="form1" method=GET
  action="http://localhost:4040/servlets-examples/servlet/my_choiceservlet">
    <b>Language:</b>
    <select name="Language" size="1">
      <option value="C">C</option>
      <option value="C++">C++</option>
      <option value="Java">Java</option>
      <option value="C#">C#</option>
    </select>
    <br><br>
    <input type="submit" value="Submit">
  </form>
  </body>
</html>
```

my_choiceservlet.java

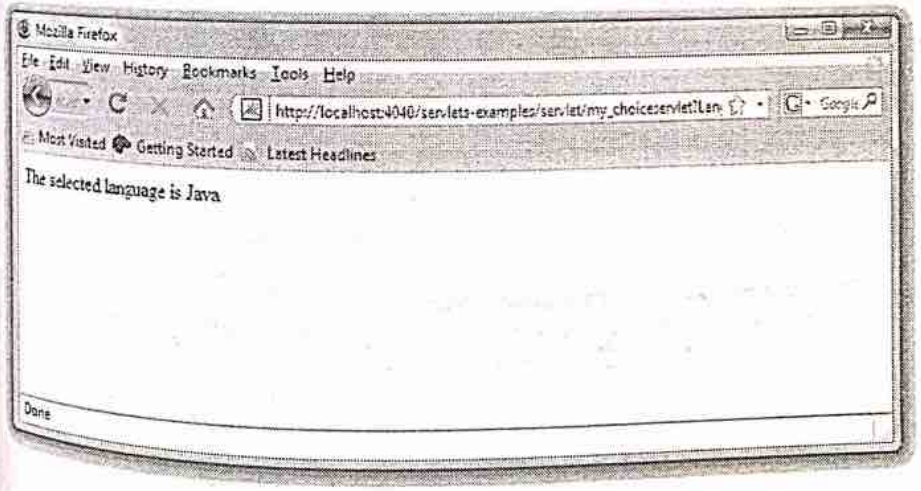
```
import java.io.*;
importjavax.servlet.*;
importjavax.servlet.http.*;

public class my_choiceservlet extends HttpServlet
{
  public void doGet(HttpServletRequestreq,HttpServletResponse res)
```



```
throwsServletException,IOException
{
  String lang=req.getParameter("Language");
  res.setContentType("text/html");
  PrintWriter out=res.getWriter();
  out.println("The selected language is "+lang);
  out.close();
}
}
```



On clicking the submit button we will get following output.



Example 3.7.2 *Write HTML form to read user name and password. This data is sent to the servlet. If the correct user name and password is given then welcome him/her by his/her name otherwise display the message for invalid user.*

Solution :

Step 1 : Create HTML form for accepting user name and password

Input.html

```
<html>
<head>
</head>
<body>
<form action="http://localhost/examples/servlets/servlet/Welcome" method ="get">
User Name:<input type="text" name="uname"/>
<br/>
Password:<input type="password" name="pwd"/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Step 2 : Create the servlet program to read user name and password and validate it.

Welcome.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Welcome extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
    throws ServletException,IOException
    {
        PrintWriter out=res.getWriter();
        res.setContentType("text/html");

        String username=req.getParameter("uname");
        String password=req.getParameter("pwd");
        if ((username==="Ankita")&&(password==="1234"))
            out.print("Welcome "+username);
        else
            out.println("Invalid username");
    }
}
```


Example 3.7.3 *Write a servlet which accept two numbers using POST methods and display the maximum of them.*

Solution :

Step 1 : The HTML document for inputting two numbers is as follows -

NumbersInput.html

```
<html>
<head>
<body>
<div align="center">
  <br> <br>
  <form action="http ://localhost/examples/servlets/servlet/MaxNumber" method="post">
    Enter First Number :
    <input type="text" value="" name="Number1" size='5'>
    <br/><br/> Enter Second Number :
    <input type="text" value="" name="Number2" size='5'>
    <br/> <br/> <br/>
    <input  type="submit" value="Submit">
  </form>
</div>
</body>
</html>
```

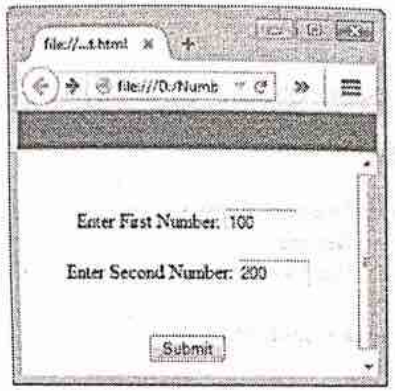
Step 2 : The servlet code that handles the post method and finds the maximum of the two input numbers is as follows -

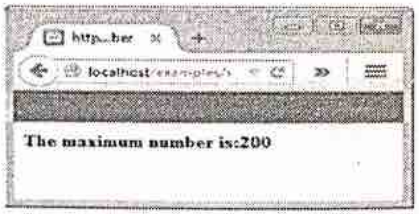
MaxNumber.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MaxNumber extends HttpServlet
  {
  protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out=res.getWriter();

    // get request parameters for userID and password
    int a = Integer.parseInt(req.getParameter("Number1"));
    int b = Integer.parseInt(req.getParameter("Number2"));
```

```
    if (a>b)
        out.println("<h4>The maximum number is:"+a+"</h4>");
    else
        out.println("<h4>The maximum number is :"+b+"</h4>");

  }
}
```

Step 3 : The output is as follows -





Example 3.7.4 *Write the code for converting currencies to US Dollar.*

Solution :

Step 1 : We will write the simple HTML document, which will create the interface. The user can select the type of currencies and will enter the value.

```
<html>
 <body>
  <formname="form1"method="post"
action="http://localhost/examples/servlets/servlet/DollarDemo">
 Select the type of Currency:
  <Select name="currency">
   <option value="Euro">Euro</option>
   <option value="Pound">Pound</option>
   <option value="Rupees">Rupees</option>
```



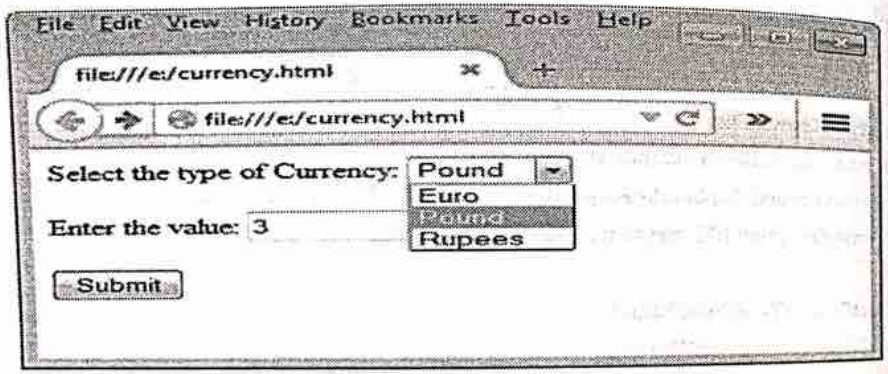
```
 </select>
 <br/>  <br/>
 Enter the value:
 <input type="text" name="currencyVal" value=""/>
 <br/>  <br/>
 <input type="submit" value="Submit"/>
 </form>
</body>
</html>
```

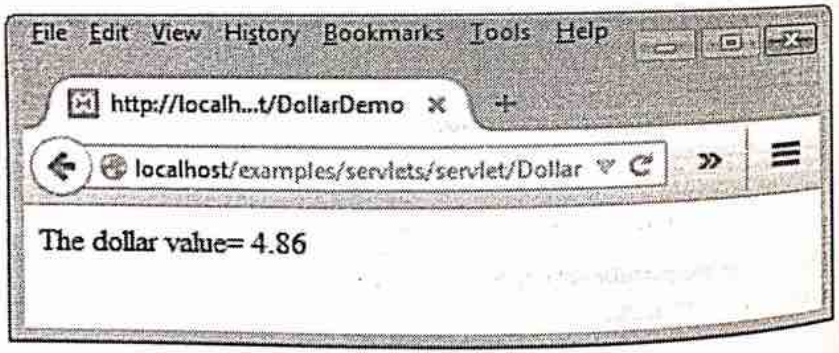
Step 2 : The java servlet for converting the currency value to dollar is as given below -

```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DollarDemo extends HttpServlet {
public void doPost(HttpServletRequest req,HttpServletResponse res)throws
   ServletException,IOException,NullPointerException
{
 res.setContentType("text/html");
 PrintWriter out=res.getWriter();
 String c=req.getParameter("currency");
 double RupeeValue,EuroValue,PoundValue;
 double DollarValue=0;
 String value=req.getParameter("currencyVal");

 if(c.equals("Rupees"))
 {
  RupeeValue=Double.parseDouble(value);
  DollarValue=RupeeValue/61;
 }
 if(c.equals("Euro"))
 {
  EuroValue=Double.parseDouble(value);
  DollarValue=EuroValue/0.77;
 }
 if(c.equals("Pound"))
 {
```

```
PoundValue=Double.parseDouble(value);
DollarValue=PoundValue*1.62;
}
out.println("<p> The dollar value= "+DollarValue+"</p>");
out.close();
}
}
```

Output





3.8 Sessions

- When you open some application, use it for some time and then close it. This entire scenario is named as **session.**
- **Session state** is a server-based state mechanism that lets web applications store and retrieve objects of any type for each **unique user session.** That is, each browser session has its own session state stored as a serialized file on the server, which is deserialized and loaded into memory as needed for each request. Refer Fig. 3.8.1.

- Because server storage is a finite resource, objects loaded into memory are released when the request completes, making room for other requests and their session objects. This means there can be more active sessions on disk than in memory at any one time.

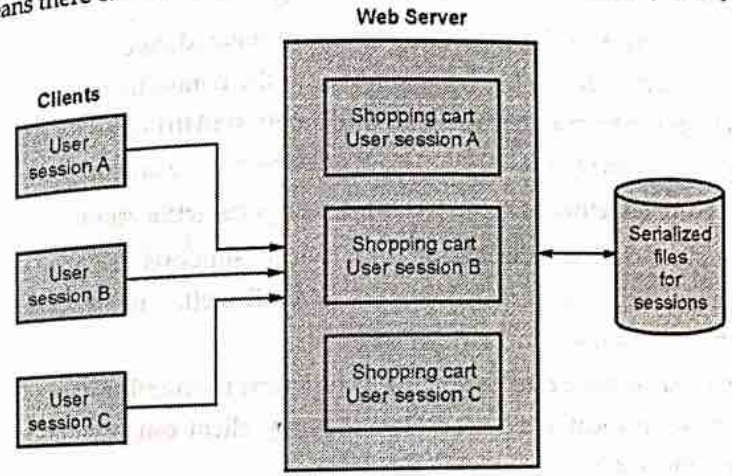


Fig. 3.8.1 Session state

- Sometimes the information about the session is required by the server. This information can be collected during the session. This process is called **session tracking.**
- There are three commonly used techniques used for session tracking and those are,
  - Session ID
  - Cookies
  - URL Rewriting.

3.8.1 Session Management using Session ID

- HTTP is a **stateless protocol** in which each request is independent of the previous request. And HTTP is a protocol using which user can interact with the server via web browser and it cannot remember previously held communications but sometimes there is serious need to keep track of previous communication sessions. This can be achieved by **session tracking.**
- The **session tracking technique** is a mechanism by which we can keep track of previous sessions between server and the browser.
- For sending all state information to and fro between browser and server, usually an ID is used. This ID is basically a **session-ID.**

- Thus session-ID is passed between the browser and server while processing the information. This method of keeping track of all the information between server and browser using session-ID is called **session tracking.**
- In servlets, for creating the sessions **getSession()** method can be used. This method returns the object which stores the bindings with the names that use this object. And these bindings can be managed using **getAttribute(), setAttribute(), removeAttribute()** methods. Actually in session tracking two things are playing an important role -
  1. One is **HttpServletRequest** interface which supports **getSession()** method.
  2. The another class is **HttpSession** class which supports the binding managing functions such as **getAttribute(), setAttribute(), removeAttribute()** and **getAttributeNames().**

Let us first discuss the servlet program which returns the number of previous sessions established between client and server. Note that any client can communicate with the server using some web browser only!

Example 3.8.1 *Write a servlet program to keep track of number of times user is visiting the page. Display the count appropriately.*

Solution : Servlet Program(SessionServletDemo)

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SessionServletDemo extends HttpServlet
{
 public void doGet(HttpServletRequest req,HttpServletResponse res)
 throws ServletException,IOException
 {
  res.setContentType("text/html");
  HttpSession session=req.getSession();
  String heading;
  Integer cnt=(Integer)session.getAttribute("cnt");
  if(cnt==null)
  {
   cnt=new Integer(0);
   heading="Welcome You are accessing the page for the First Time";
  }
  else
  {
   heading="Welcome once again!";
```



```
   cnt=new Integer(cnt.intValue()+1);
  }
  session.setAttribute("cnt",cnt);
  PrintWriter out=res.getWriter();
  out.println("<html>");
  out.println("<head>");
  out.println("</head>");
  out.println("<body>");
  out.println("<center>");
  out.println("<h1>"+heading);
  out.println("<h2> The number of previous access= "+cnt);
  out.println("</center>");
  out.println("</body>");
  out.println("</html>");
 }
}
```

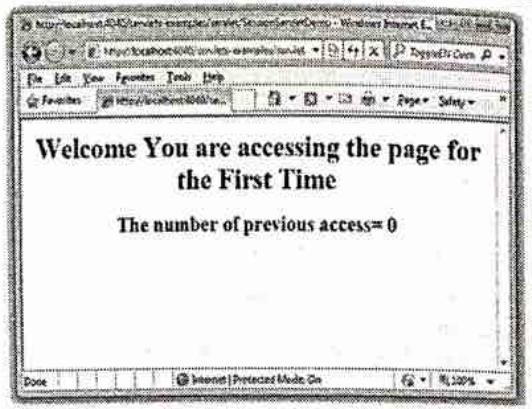
Output

To get an output we will use following steps -

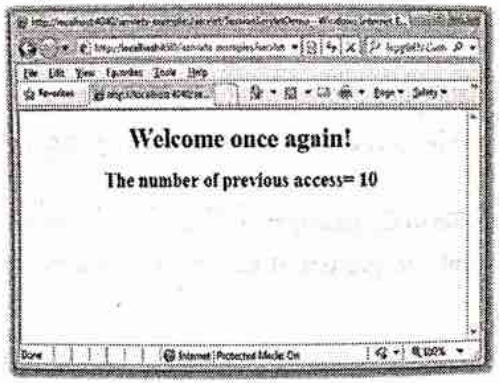
1. Compile the Servlet (i.e. a java program) using **javac.** This will generate a class file.
2. Copy this class file to C:*your tomcat directory**examples-folder**WEB-INF**classes*
3. Edit the **web.xml** file present at C:*your tomcat directory**examples-folder**WEB-INF* like this

```
...
<servlet>
   <servlet-name>SessionServletDemo</servlet-name>
   <servlet-class>SessionServletDemo</servlet-class>
 </servlet>
...
<servlet-mapping>
        <servlet-name>SessionServletDemo</servlet-name>
         <url-pattern>/servlet/SessionServletDemo</url-pattern>
</servlet-mapping>
```

4. Start tomcat server.
5. Open the web browser and give the URL for **SessionServletDemo.** Following output can be seen.



If you simply refresh your browser either by pressing F5 or clicking the refresh button then you will get following kind of output.



Program Explanation

1. Our class *SessionServletDemo* is inherited from **HttpServlet** class. And inside the **doGet()** method we can create an object *session* of **HttpSession** class. This method returns the current session for the corresponding request.
2. The **getAttribute(string** *attribute)* returns the value associated with the *attribute*. We can pass this *attribute* as a parameter to this method. If there is no attribute present then this method returns null. The data type of getAttribute() method is **void.**
3. The setAttribute(**string** *attribute,* **object** *value)* is a method which assigns the value passed as the object value to the attribute name. The data type of this method is **void.**

SPPU : Dec.-19, Marks 5

3.9 Cookies

Definition : Cookies are some little information that can be left on your computer by the other computer when we access an internet.

- Generally this information is left on your computer by some advertising agencies on the internet. Using the information stored in the cookies these advertising agencies can keep track of your internet usage, liking of users.
- For the applications like on-line purchase systems once you enter your personal information such as your name or your e-mail ID then it can be remembered by these systems with the help of cookies.
- Sometimes cookies are very much dangerous because by using information from your local disk some malicious data may be passed to you. So it is upto you how to maintain your own privacy and security.

3.9.1 Servlet Support for Cookies

- The **cookie** class is used to create cookies in servlet.
- **Syntax :** The Syntax for constructor for cookies are,
  - Cookie()
  - Cookie(String name, String value)
- Various methods used in Cookie are described in following table

Sr. No.	Method	Purpose
1.	public string getName()	It returns the name of the cookie.
2.	public String getValue()	It returns the value of the cookie.
3.	public string setName()	It sets or changes the name of the cookie.
4.	public String setValue()	It sets or changes the value of the cookie.
5.	public void addCookie(Cookie c)	The cookie is added in the response object of HttpServletResponse interface.
6.	public Cookie[] getCookies()	All the cookies can be returned using this method with the help of HttpServletRequest interface.

- In Servlet following steps are used to support for cookies

Step 1 : Creation of Cookies :

The cookie can be created in Servlet and added to response object using **addCookie** method

```
Cookie cookie=new Cookie("user","XYZ");//creating cookie object
response.addCookie(cookie);//adding cookie in the response
```

Step 2 : Reading Cookies :

The value from the cookie can be obtained using the **getName()** and **getValue()** methods.

```
Cookie[ ] my_cookies=req.getCookies();
int n=my_cookies.length;
for(int i=0;i<n;i++)
{
String name=my_cookies[i].getName();
String value=my_cookies[i].getValue();
}
```

3.9.2 Examples

Below is simple HTML form in which a servlet is invoked. This servlet creates a cookie by the name **My_Cookie** and stores the value entered by you in the textbox of HTML form. You can further get the information stored in the cookie by another servlet program **getCookieServlet**. Hence we will write three programs -

1. Our normal HTML script in which some value is entered in the textbox.
2. The servlet program named mycookieservlet which will set a cookies and take the value entered by you in the HTML form.
3. The another servlet program named *getCookieServlet* which helps us to view the cookie.

HTML Program

```
<html>
<head>
<title>Demo of Cookie</title>
</head>
<body>
<form name="form1" method="post"
action="http ://localhost:8080/examples/servlet/mycookieservlet">
<h3> Enter the value for my Cookie: </h3>
<input type=text name="txt_data" size =30 value=" ">
<input type=submit value="Submit">
</form>
</body>
</html>
```


Servlet Program [mycookieservlet.java]

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class mycookieservlet extends HttpServlet
{
public void doPost(HttpServletRequest req,HttpServletResponse res)
throws ServletException, IOException
{
String txt_data = req.getParameter("txt_data");
// Create cookie.
Cookie cookie = new Cookie("My_Cookie", txt_data);
// Adding cookie to HTTP response.
res.addCookie(cookie);
// Write friendly output to browser.
res.setContentType("text/html");
PrintWriter out = res.getWriter();
out.println("<h2>MyCookie has been set to : ");
out.println(txt_data);
out.println("<br><br><br>");
out.println("This page shows that the cookie has been added");
out.close();
}
}
```

We have first created an object of Cookie class using which the cookie can be added using **addCookie()** method.

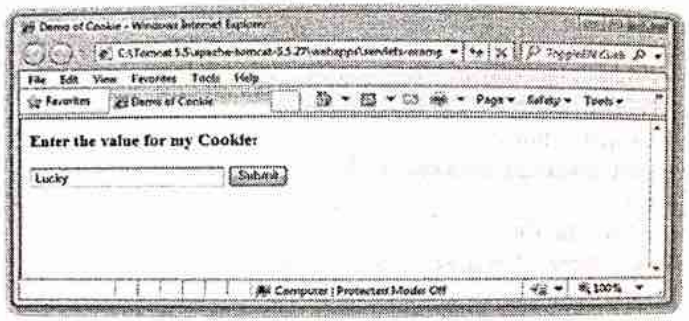
Servlet Program[getcookieservlet.java]

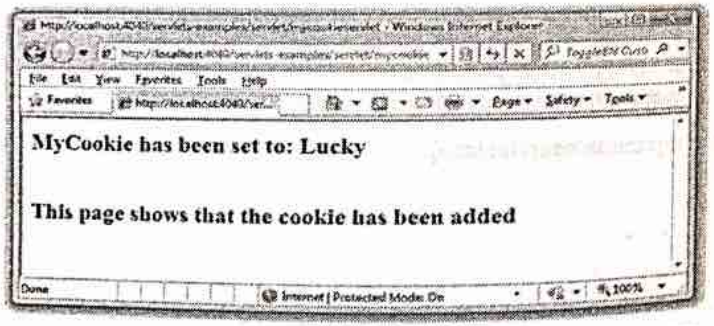
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class getCookieServlet extends HttpServlet
{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
Cookie[ ] my_cookies=req.getCookies();
res.setContentType("text/html");
PrintWriter out=res.getWriter();
out.println("<b>");
int n=my_cookies.length;
for(int i=0;i<n;i++)
{
```

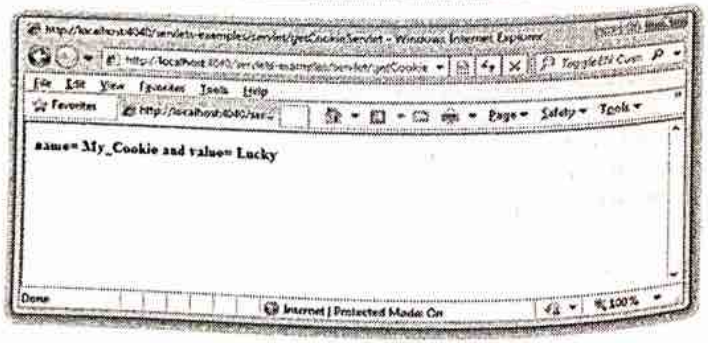
```
String name=my_cookies[i].getName();
String value=my_cookies[i].getValue();
out.println("name= "+name);
out.println("and value= "+value);
}
out.close();
}
}
```

In the above program using **getName()** and **getValue()** functions we can get the name of the cookie as well as the value of the cookie respectively. The output of all the above given three programs is as given below -

Output







Example 3.9.1 *Write a program that allows the user to select a favorite programming language and post the choice to the servlet. The response is a web page in which user can click a link to view list of book recommendations. The cookies previously stored on the client are read by the servlet and form a web page containing the book recommendation. Use servlet, cookies and HTML.*

Solution :

Step 1 : Create an HTML page which allows the user to select the favourite programming language. The user can submit his choice by clicking the submit button.

InputForm.html

```
<html>
<body>
<form name="form1" method="post"
action="http://localhost/examples/servlets/servlet/MyProgram">
<h3> Enter the favourite Programming Language: </h3>
<select name="Lang_ch">
   <option value="C++">C++</option>
   <option value="Java">Java</option>
   <option value="PHP">PHP</option>
</select>
<input type=submit  value="Submit">
</form>
</body>
</html>
```

Step 2 : Now create a web servlet program which reads the choice of programming language and set this choice as cookie. The web page containing a hyperlink is created in this servlet program. The code for this Servlet program is as follows -

MyProgram.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyProgram extends HttpServlet
{
   public void doPost(HttpServletRequest req,HttpServletResponse res)
   throws ServletException, IOException
   {
      String choice = req.getParameter("Lang_ch");
      // Create cookie.
      Cookie cookie = new Cookie("My_Cookie", choice);
```

```
        // Adding cookie to HTTP response.
        res.addCookie(cookie);
        // Write friendly output to browser with web link
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<a href='ReadChoice'><p>Click Here to read about your
        favorite programming language </p></a>");
        out.println("<br><br><br>");
        out.println("This page shows that the cookie has been added");
        out.close();
    }
}
```

Step 3 : On clicking the hyperlink present in the above servlet, we can read the value of cookie(which is actually the name of the favorite programming language). The following program displays the recommended book for the programming language you have chosen. The code is as follows -

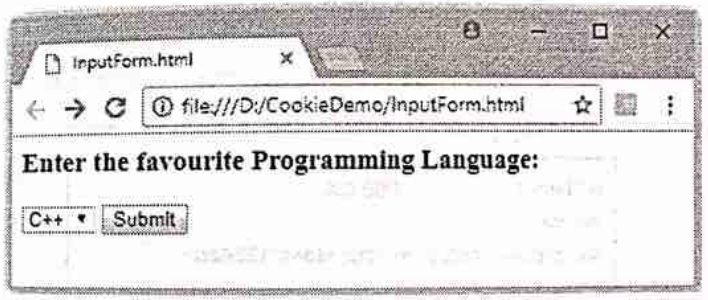
ReadChoice.java

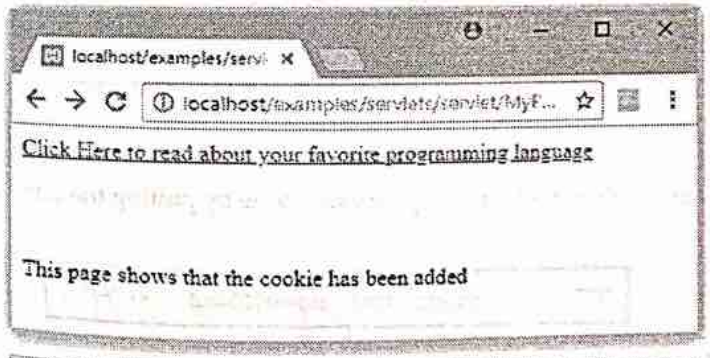
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ReadChoice extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
     throws ServletException,IOException
    {
        Cookie[] my_cookies=req.getCookies();
        res.setContentType("text/html");
        PrintWriter out=res.getWriter();
        out.println("<b>");
        int n=my_cookies.length;
        for(int i=0;i<n;i++)
        {
          String name=my_cookies[i].getName();
          String value=my_cookies[i].getValue();
          if(value.equals("PHP"))
          {
            out.println("<b>Programming Language: </b>"+value);
            out.println("<br/><b>Book Name: </b>"+"PHP and MySQL Web
            Development by Luke Welling and Laura Thompson");
          }
          else if(value.equals("Java"))
          {
            out.println("<b>Programming Language: </b>"+value);
```

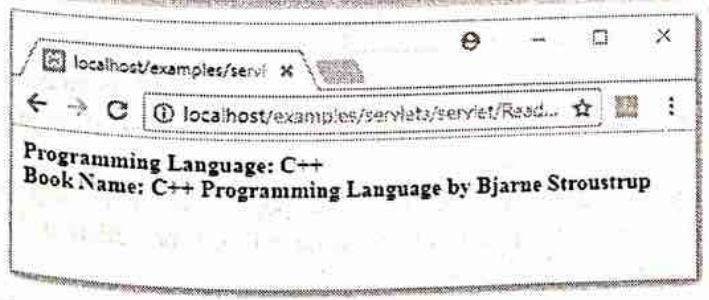


```
            out.println("<br/><b>Book Name: </b>"+"The Complete
            Reference Java");
          }
          else
          {
            out.println("<b>Programming Language: </b>"+value);
            out.println("<br/><b>Book Name: </b>"+"C++
            Programming Language by Bjarne Stroustrup");
          }
        }
        out.close();
    }
}
```

Output







Review Question

1. *Explain session and cookies in servlets ? Write any one program to demonstrate session or cookies* **SPPU : Dec.-19, Marks 5**

3.10 URL Rewriting

This is more precise technique in which information is embedded into URL. For instance.

```
GET    / introduction.html    HTTP P.1

HOST   www.mywebpage.com
```

Browser —— **Requests** ——→ **Server**

The server then repsonds by putting **sid=xf1234ad** in response body.

```
HTTP/1.1            200 OK
<html>
<a  href = Introduction.html; sid=xf1234ad>
    Click here to go to Next page </a>
    : : :
</html>
```

Browser ←—— **Response** —— **Server**

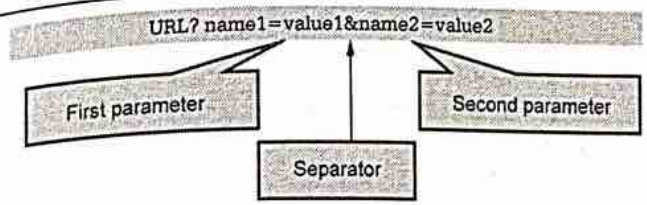
The browser will demand for the required web page by putting the same URL string in the request.

```
GET    /introduction.html  sid=xf1234ad    HTTP/1.1
```

Browser —— **Request** ——→ **Server**

When browser makes requests for another page same URL is embedded. There by server will come to know that the request is related to previous page and the work in same session will get co-related.

We can send the name-value pair from one page to another using URL rewriting method. In this method the token of name-value pair can be added at the end of URL. The URL writing occurs in the following format



The parameter sent by the URL is obtained at the server side by **getParameter** method. The syntax of this method is

String getParameter(String)

Example 3.10.1 *Write a servlet that Welcomes the user by his/her name only if the valid password is entered by the user. Make use of URL rewriting method.*

Solution :

Step 1 : Create a registration form. I have named it as registration.html. The code for it is as follows :

registration.html

```
<!doctype html>
<html>
<head>
<title> Demo For URL Rewriting </title>
<body>
<form method ="post" action="http://localhost/examples/servlets/servlet/firstpg">
User:<input type ="text" name="user" ><br/>
Password:<input type ="password" name="pwd"> <br/><br/>
<input type="submit" value="Submit">
</form>
</body>
```

Step 2 : Create a servlet program that reads the user name and password values entered by the user. This servlet also compares the entered password with the valid password. If the user enters valid password, then the request is redirected to another servlet page. Otherwise *wrong password* message will be displayed. The first servlet is,

firstservlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class firstpg extends HttpServlet {
 protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
```

```
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String uname = request.getParameter("user");
    String pass = request.getParameter("pwd");
    if(pass.equals("password"))
    {
      response.sendRedirect("secondpg?user_name="+uname+"");
    }
    else
      out.println("<i> You have entered wrong password!!!</i>");
  }
}
```

Step 3 : If user enters the valid password, then he/she is redirected to main page. This page welcomes him/her by his/her name. The code for this servlet is as follows -

secondpg.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class secondpg extends HttpServlet {
 protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String user = request.getParameter("user_name");
    out.println("<h3>Welcome "+user+" to main page!!!</h3>");
  }
}
```

Step 4 : Compile the java files created in step 2 and step 3. Copy the class files to classes folder of Tomcat's WEB-INF folder. Modify the web.xml file as follows -

```
<?xml version="1.0" encoding="ISO-8859-1"?>
...
  <servlet>
    <servlet-name>firstpg</servlet-name>
    <servlet-class>firstpg</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>secondpg</servlet-name>
    <servlet-class>secondpg</servlet-class>
  </servlet>
```

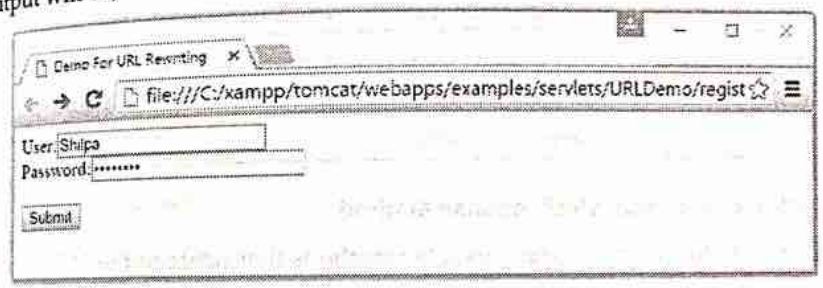


```
  <servlet-mapping>
    <servlet-name>firstpg</servlet-name>
    <url-pattern>/servlets/servlet/firstpg</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>secondpg</servlet-name>
    <url-pattern>/servlets/servlet/secondpg</url-pattern>
  </servlet-mapping>
</web-app>
```

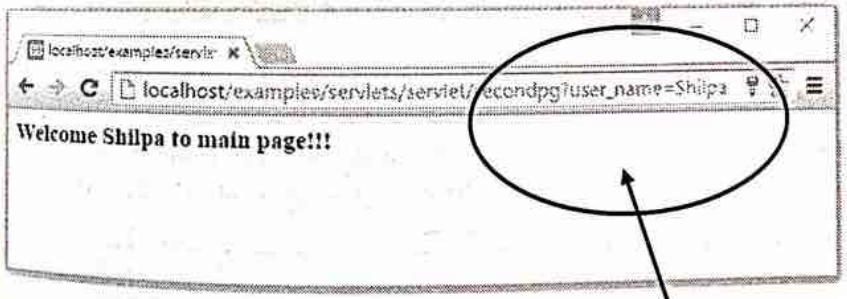
Start the Tomcat web server.

Open the suitable web browser and invoke the HTML document as given in Step 1.

The output will be,



On clicking submit button we will get the following page on valid password



URL Rewriting with name as "user_name and value as "Shilpa"

3.11 Other Servlet Capabilities

In this section we will discuss various methods of **HttpServletRequest** and **HttpServletResponse** class.

3.11.1 Additional HttpServletRequest Method

HttpServletRequest interface adds the methods that relates to the **HTTP** protocol. Following table lists various useful methods of **HttpServletRequest**

Method	Description
String getRemoteAddr()	This method returns IP address of the client who is making the request.
String getRemoteHost()	This method returns the name of the client who is making the request.
String getProtocol()	This method returns the name of the protocol, client is using.
String getHeader(String field)	When the field name is given the value for the header is returned.
StringBuffer getRequestURL()	This method returns the String containing the URL which is used to access the servlet. This string does not contain the query string appended to URL.

3.11.2 Additional HttpServletResponse Method

- **HttpServletResponse** interface adds the methods that relate to the **HTTP** response. It extends the ServletResponse interface.
- Following table lists various useful methods of **HttpServletResponse**

Methods	Description
void addCookie(Cookie cookie)	Adds the specified cookie to the response.
void sendRedirect(String location)	Sends a temporary redirect response to the client using the specified redirect location URL and clears the buffer.
int getStatus()	Gets the current status code of this response.
String getHeader(String name)	Gets the value of the response header with the given name.
void setHeader(String name, String value)	Sets a response header with the given name and value.
void setStatus(int sc)	Sets the status code for this response.
void sendError(int sc, String msg)	Sends an error response to the client using the specified status and clears the buffer.

Example Program

Step 1 :

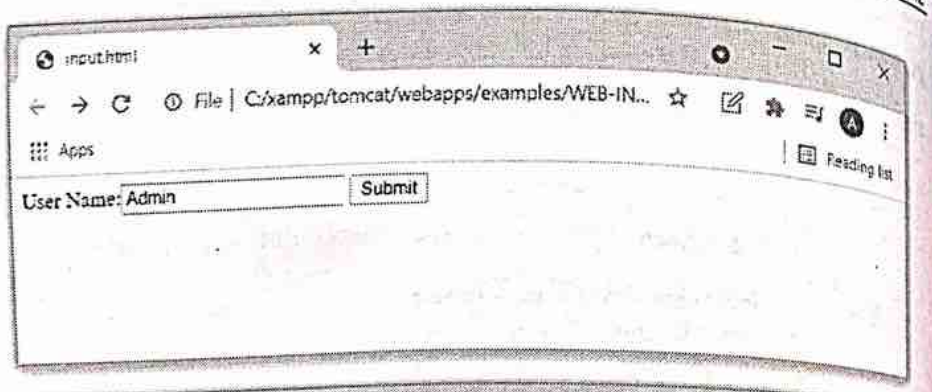
```
<html>
<head>
</head>
<body>
   <form action="http://localhost:8080/examples/servlets/servlet/ClientAddressServlet"
method ="get">
      User Name:<input type="text" name="uname"/>
      <input type="submit" value="Submit"/>
   </form>
</body>
</html>
```

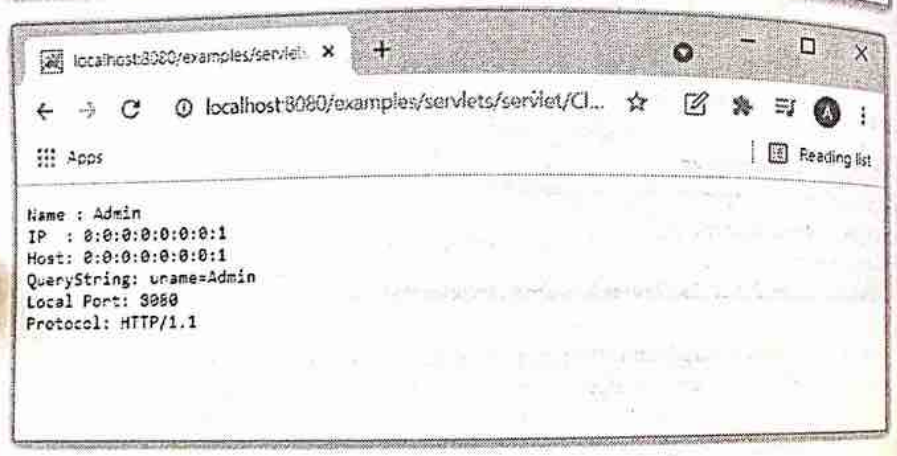
Step 2 :

ClientAddressServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class ClientAddressServlet extends HttpServlet
{
   public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
   {
      // Get client's IP address
      String name = request.getParameter("uname");
      String clientIP = request.getRemoteAddr();
      // Get client's host name
      String clintHost = request.getRemoteHost();
      String queryString = request.getQueryString();
      String protocol = request.getProtocol();
      int localPort = request.getLocalPort();
      response.setContentType("text/plain");
      PrintWriter out = response.getWriter();

      out.println("Name : " + name);
      out.println("IP  : " + clientIP);
      out.println("Host: " + clintHost);
      out.println("QueryString: "+queryString);
      out.println("Local Port: "+localPort);
      out.println("Protocol: "+protocol);
   }
}
```

Output





Program Explanation : In above program,

1. In step 1 we have created an HTML form on which there is one text box and one submit button. The name entered in the textbox is identified by the parameter **uname.**
2. In step 2, we have two objects - request and response. These are basically the objects of HttpRequest and HttpResponse classes.
3. Using **request.getParameter** method we invoke the **uname** parameter. By this, we can access the user name entered by the user in step 1.
4. In the above program, we have used **getRemoteAddr(), getRemoteAddr(), getProtocol()** methods and display the corresponding information on the web browser.

3.12 Data Storage

- Typical web application contains the HTML documents that often interact with some form of data storage. This data storage can be a file system or a database management system.
- The efficient use of file system or database management system is required by every web application.
- There are two commonly used techniques for data storage - **Java Object Serialization** and **Java Database Connectivity(JDBC)**
- **Serialization in Java** is a mechanism of writing the state of an object into a byte-stream.
- JDBC is an API used to assist the Java programs to interact with the database management systems(popularly handled using MySQL)
- Following is an example program that shows **serialization in Java.**

Serialization is the process of writing the state of the object to the byte stream. This technique is useful when we want to store the current state of the object to the File. For serialising the object we need to implement **serializable** interface. This interface does not define any member.

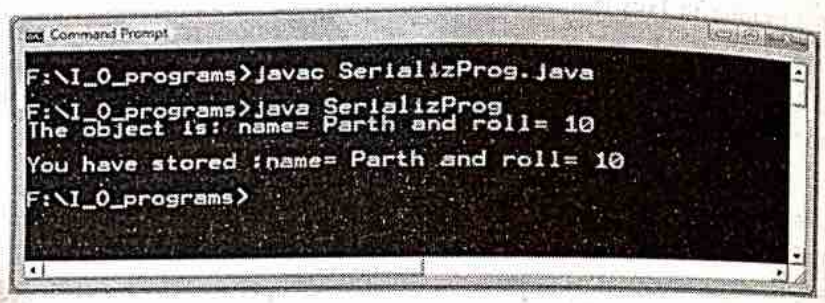
Java Program[SerializProg.java]

```
import java.io.*;
class SerializProg
{
public static void main(String[] args)throws ClassNotFoundException,IOException
{
    Myclass myobject=new Myclass("Parth",10);//creating the object
    System.out.println("The object is: "+myobject);//displaying the contents of the object
    ObjectOutputStream o=new ObjectOutputStream(new FileOutputStream("object.txt"));
    o.writeObject(myobject);//writing the object to a file
    o.flush();
    o.close();
    ObjectInputStream i=new ObjectInputStream(new FileInputStream("object.txt"));
    Myclass newobject=(Myclass)(i.readObject());//reading the object from the file
    i.close();
    System.out.println("\nYou have stored :"+newobject);//displaying the object's value
}
static class Myclass implements Serializable
{
String name;
int roll;
```

Implements the serializable interface

```
public Myclass(String n,int r)
{
 this.name=n;
 this.roll=r;
}
public String toString()
{
 return "name= "+name+" and roll= "+roll;
}
}
}
```

Output



Program Explanation

In above program,

1. We have created an object of class named **Myclass**. The members of this class are **name** and **roll number.**
2. From the main function assigns the values to these members. This object is written to the file named **object.txt** using the **writeObject** method.
3. Later on the value of this object can be read using the **readObject** method.
4. For the creation of the object we have to define a class who implements the **Serializable** interface.
5. It is necessary to throw the exceptions **ClassNotFoundException** and **IOException**

3.13 Servlets and Concurrency

- Concurrency allows multiple users to access the system. It also allows system programs to execute concurrently. Also, the web applications implement the concurrency. For instance, when web browser is loading some image, at the same time it also responds to mouse clicks. This type of concurrency is achieved at the client end in client's browser.

- On the server side the situation is different. There can be hundreds or even thousands of users simultaneously accessing a web site.
- Concurrent web servers handle multiple users as follows - It begins with one servlet, executes some instruction, then suspends it for a while and start executing the other servlet. Eventually completes the second servlet and returns back to first servlet to execute the remaining instructions. Refer Fig. 3.13.1.

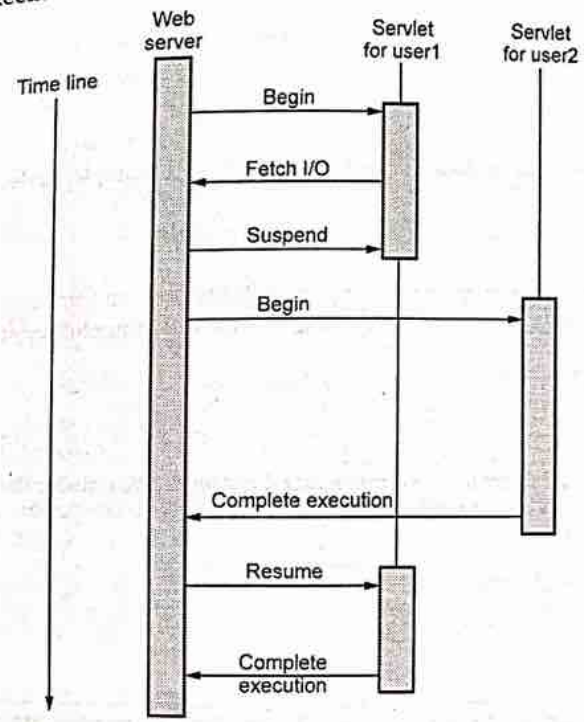


Fig. 3.13.1 Concurrency achieved by web server

Concept of Thread :

- Thread is a light weight process. It is used for concurrent execution of the tasks.
- The Java Virtual Machine maintains a data structure containing the information about each thread. The server creates a thread for each HTTP request. The data structure maintains the information about the state of the thread along with the references to the **HttpServletRequest** and **HttpServletResponse** objects created by the server.
- The Java Virtual Machine also maintains the information whether the particular thread is currently running or not.

- When one thread is suspended then the state of other executing thread is loaded into Java Virtual Machine.
- Following example shows the concurrent execution of the servlet with the help of threads

Step 1 :

```
<HTML>
 <BODY>
  <TABLE>
   <TR>
    <TD>
     <a href ="http://localhost:8080/examples/servlets/servlet/SimpleServlet">servlet1</a>
    </TD>
   </TR>
   <TR>
    <TD>
     <a href ="http://localhost:8080/examples/servlets/servlet/SimpleServlet">servlet2</a>
    </TD>
   </TR>
   <TR>
    <TD>
     <a href ="http://localhost:8080/examples/servlets/servlet/SimpleServlet">servlet3</a>
    </TD>
   </TR>
  </TABLE>
 </BODY>
</HTML>
```

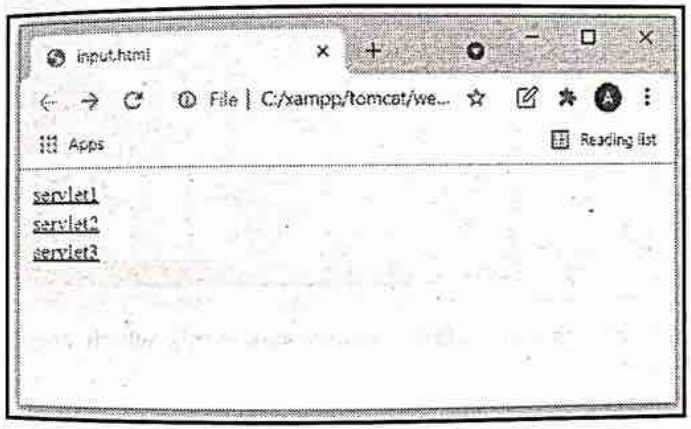
Step 2 :

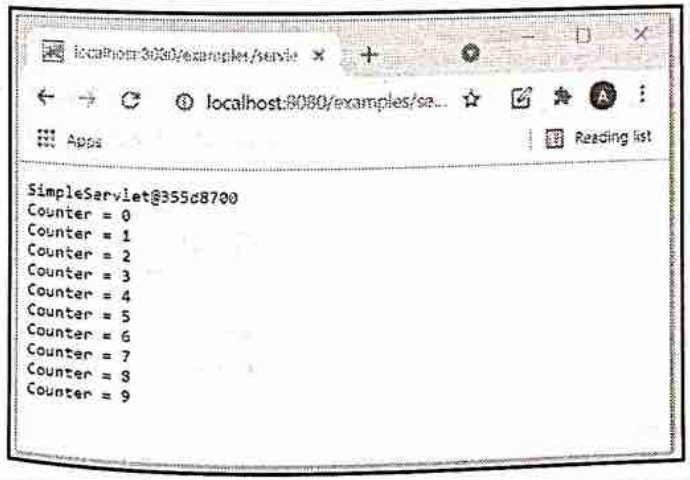
```
import java.io.IOException;
import javax.servlet.*;
import javax.servlet.http.*;
import java.math.*;
public class SimpleServlet extends HttpServlet
{

  public void doGet(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException
  {
    doPost(req, resp);
  }
  public void doPost(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException
```



```
  {
    int counter = 0;
    resp.getWriter().println(this);
    for (int i = 0; i < 10; i++)
    {
      resp.getWriter().println("Counter = " + counter);
      try
      {
        Thread.currentThread().sleep((long) Math.random() * 1000);
        counter++;
      }
      catch (InterruptedException exc) { }
    }
  }
}
```

Output





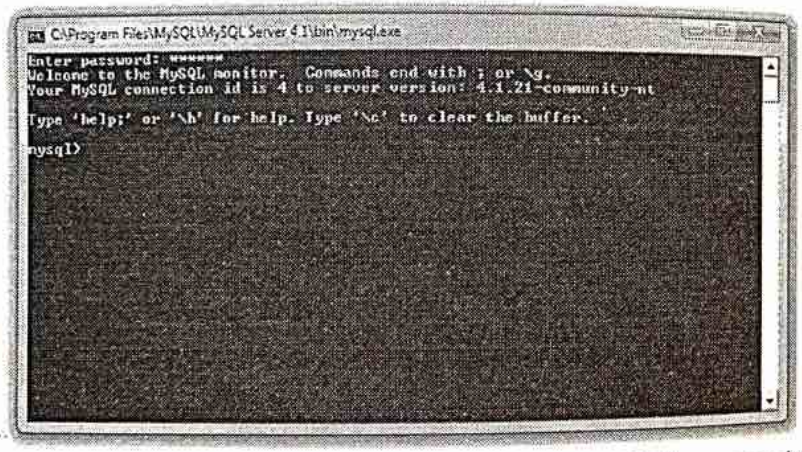
3.14 Databases (MySQL) and Java Servlets

JDBC is a Java API for database access. A servlet can connect to MYSQL and sends the SQL commands to the database and get the required data. Hence we will first get introduced with structured query language SQL.

3.14.1 Structured Query Language using MySQL

The MYSQL is a open source database product which can be downloaded from the web site.

After getting installed on the machine the command prompt window for MYSQL can appear. The screenshot is as follows -



Now let us go through some MYSQL query statements which are required while handling the database.

1. Creating database

```
mysql> CREATE DATABASE mydb;
Query OK, 1 row affected (0.15 sec)
```

2. Displaying all the databases

```
mysql> SHOW DATABASES;
+------------+
| Database   |
+------------+
| mydb       |
| mysql      |
| students   |
| test       |
+------------+
4 rows in set (0.06 sec)
```


3. Selecting particular database

```
mysql> USE MYDB;
Database changed
```

4. Creating table

We must create a table inside a database hence it is a common practice to use create table command after USE database command. While creating a table we must specify the table fields.

```
mysql> CREATE TABLE my_table(id INT(4),name VARCHAR(20));
Query OK, 0 rows affected (0.28 sec)
```

Use of Primary Key : The primary key contains the unique value. The primary key column can not contain NUL value. Each table can have only one primary key. Following is a SQL statement used to create PRIMARY KEY.

```
CREATE TABLE student_table(
roll_no INT(4) NOT NULL AUTO_INCREMENT,
name VARCHAR(50) NOT NULL,
address VARACHAR(50) NOT NULL,
PRIMARY KEY(roll_no)
);
```

In above example the roll_no acts as a primary key for the student_table.

5. Displaying a table

After creating the table using SHOW command we can see all the existing tables in the current database.

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_mydb  |
+-----------------+
| my_table        |
+-----------------+
1 row in set (0.00 sec)
```

6. Displaying the table fields

For knowing the various fields of the table we may use following command

```
mysql> DESCRIBE my_table;
+-------+-------------+-------+-----+---------+--------+
| Field | Type        | Null  | Key | Default | Extra  |
+-------+-------------+-------+-----+---------+--------+
| id    | int(4)      | YES   |     | NULL    |        |
| name  | varchar(20) | YES   |     | NULL    |        |
+-------+-------------+-------+-----+---------+--------+
2 rows in set (0.07 sec)
```

7. Inserting values into the table

We can insert only one complete record at a time. It is as shown below -

```
mysql> INSERT INTO my_table
   -> VALUES(1,'SHILPA');
Query OK, 1 row affected (0.05 sec)
```

8. Displaying the contents of the table

```
mysql> SELECT * FROM my_table;
+------+-----------+
| id   | name      |
+------+-----------+
|  1   | SHILPA    |
+------+-----------+
1 row in set (0.06 sec)
```

We can also write SELECT statement for selecting particular row by specifying some condition such as -

```
mysql> SELECT * FROM my_table where id=1;
or
mysql> SELECT * FROM my_table where name='SHILPA';
```

Thus we can insert the rows into the table by repeatedly giving the INSERT command.

If we want to get the records in sorted manner then we use ORDER BY clause

```
mysql> SELECT * FROM my_table;
+------+-------------+
| id   | name        |
+------+-------------+
|  1   | SHILPA      |
|  2   | SUPRIYA     |
|  3   | YOGESH      |
|  4   | MONIKA      |
+------+-------------+
4 rows in set (0.00 sec)
mysql> SELECT * FROM my_table ORDER BY name;
+------+-----------+
| id   | name      |
+------+-----------+
|  4   | MONIKA    |
|  1   | SHILPA    |
|  2   | SUPRIYA   |
|  3   | YOGESH    |
+------+-----------+
4 rows in set (0.00 sec)
```


9. Updating the record

For updating the record in the database following command can be used -

```
mysql> UPDATE my_table
   -> SET name='PRIYANKA'
   -> WHERE id=4;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
mysql> SELECT * FROM my_table;
+----+-------------+
| id | name        |
+----+-------------+
|  1 | SHILPA      |
|  2 | SUPRIYA     |
|  3 | YOGESH      |
|  4 | PRIYANKA    |
+----+-------------+
4 rows in set (0.00 sec)
```

10. Deleting record

For deleting particular record from a database

```
mysql> DELETE FROM my_table
   -> WHERE id=3;
Query OK, 1 row affected (0.04 sec)
```

Then use SELECT statement for displaying the contents of the table we use following command

```
mysql> SELECT * FROM my_table;
+----+-------------+
| id | name        |
+----+-------------+
|  1 | SHILPA      |
|  2 | SUPRIYA     |
|  4 | PRIYANKA    |
+----+-------------+
3 rows in set (0.00 sec)
```

11. For deleting the table

The table can be deleted using the command

mysql>drop table my_table;

12. Inner join

This query used to display the result from both the tables when at least one column from both the tables is matching -

For example

Consider table **Customer**

Cust_Id	Name	City
1	Rahul	Bombay
2	Priyanka	Pune
3	Supriya	Banglore

Consider table **Order**

Order_Id	Cust_Id	Order_Number
10	3	1234
20	1	5678
30	1	8900

Then we can make use of inner join query as follows -

```
SELECT Customer.Name, Customer.city, Order.Order_Number
FROM Customer
INNER JOIN Order
ON Customer.Cust_Id=Order.Order_Id
```

Name	City	Order_Number
Rahul	Bombay	5678
Rahul	Bombay	8900

13. Order by clause

If we want to get the records in sorted manner then we use ORDER BY clause

```
mysql> SELECT * FROM my_table;
```

id	name
1	SHILPA
2	SUPRIYA
3	YOGESH
4	MONIKA

```
mysql> SELECT * FROM my_table ORDER BY name;
```

id	name
4	MONIKA
1	SHILPA
2	SUPRIYA
3	YOGESH

3.14.2 JDBC Perspectives

- JDBC stands for Java DataBase Connectivity.
- JDBC is nothing but an API (i.e. Application Programming Interface).
- It consists of various classes, interfaces, exceptions using which Java application can send SQL statements to a database. The SQL is a Structured Query Language used for accessing the database.
- JDBC is specially used for having connectivity with the RDBMS packages (such as Oracle or MYSQL) using corresponding JDBC driver.

The **JDBC API** is a set of classes, interfaces and exceptions used for establishing connection with data source. This JDBC API is defined in the **java.sql** and **javax.sql** packages. We use following core JDBC classes and interfaces that belong to **java.sql** package.

- **DriverManager** - When Java application needs connection to the database it invokes the **DriverManager** class. This class then loads JDBC **drivers** in the memory. The DriverManager also attempts to open a connection with the desired database. For JSP-MYSQL connectivity we can get DriverManager class as -

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

1) **Connection** - This is an interface which represents connectivity with the data source. The **connection** is used for creating the **Statement** instance.

```
Connection conn=
DriverManager.getConnection("jdbc:mysql://localhost:3306/databaseName",
"userName","password");
```

2) **Statement** - This interface is used for representing the SQL statements. Some examples of SQL statements are,
   - SELECT * FROM students_table;
   - UPDATE students_table set name='Neel' where roll_no='1';
   - DELETE from students_table WHERE roll_no='10';

The code for statement is as follows -

```
Statement stmt=conn.createStatement();
stmt.executeQuery("select * from mytab");
```

There are two specialised Statement types: **PreparedStatement** and **CallableStatement.** The **PreparedStatement** represents the precompiled SQL statements.

For instance :

```
SELECT * from students_table WHERE name=?
```

The placeholder represented by ? is used in this type of statement. There are special type setter methods that assign the values to the placeholders before the SQL statement is executed.

CallableStatement is represents the stored procedures. These are similar to **PreparedStatement.** In this type of Statements we can assign the methods for the types of output arguments.

3) **ResultSet** - This interface is used to represent the database result set. After using SELECT SQL statement, the information obtained from the database can be displayed using **ResultSet.**

4) **SQLException** - For handling SQL exceptions this interface is used.

3.14.2.1 How JDBC Works ?

Following is a way by which JDBC works -

1) First of all Java application establishes connection with the data source.
2) Then Java application invokes classes and interfaces from JDBC driver for sending queries to the data source.
3) The JDBC driver connects to corresponding database and retrieves the result.
4) These results are based on SQL statements which are then returned to Java application.
5) Java application then uses the retrieved information for further processing.

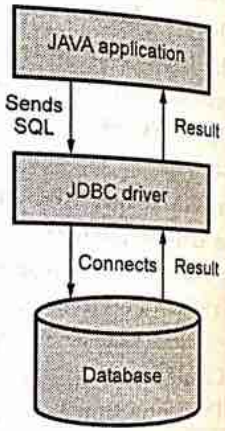


Fig. 3.14.1 Role of JDBC

3.14.2.2 Difference between JDBC and ODBC

The JDBC is similar to the ODBC i.e. Open Database Connectivity. But the ODBC is used for managing any kind of database and JDBC is specially created to support the Java programs.

3.14.2.3 Uses of JDBC

1. JDBC helps the client to store and retrieve the data to the databases.
2. JDBC allows the client to update the databases.

3.14.3 Steps for Connectivity between Servlet and Database

Following steps are used to connect JDBC to MYSQL

Step 1 : Import java.sql.* package in the JDBC program

Following line can be included in your JDBC program at the beginning.

```
import java.sql.*
```

Step 2 : Load JDBC Driver.

The JDBC driver for MYSQL can be loaded using following statement

```
Class.forName("com.mysql.jdbc.Driver");
```

Step 3 : Get connection using the Driver Manager

```
conn =
riverManager.getConnection("jdbc:mysql://localhost:3306/my_database","root","password");
```

Step 4 : Create Statement

```
stmt = conn.createStatement();
```

Step 5 : Execute Query

```
String sql = "SELECT Roll,StudName FROM my_table";
ResultSet rs = stmt.executeQuery(sql);
```

Step 6 : Display the result

Example 3.14.1 *Explain the JDBC URL with appropriate examples.*

Solution : JDBC URL with examples : The databases are referenced by Java runtime engine using JDBC URL. The syntax of these URLs depends on the database driver you are using.

The list of common drivers along with JDBC URL is,

1. ODBC Driver

Syntax :

```
jdbc:odbc:<data source name>
```

Example :

```
jdbc:odbc:MyDataSource
```

2. MYSQL

Syntax :

```
jdbc:mysql://[host][:port]/[database]
```

Example :

```
jdbc:mysql://localhost:3306/Mydatabase
```

3. ORACLE

Syntax :

```
jdbc:oracle:<drivertype>:<user>/<password>@<database>
```

Example :

```
jdbc:oracle:thin:myuser/mypassword@localhost:mydatabse
```

3.14.4 Implementation

Prerequisite : We need following things for connecting to database using JDBC

1. MySQl is already installed and required database is created.
2. JDK is installed.
3. A database connector mysql-connector-java-X.X.XX-bin must be downloaded and present at the path

```
C:\your_tomcat_directory\common\lib
```

4. Tomcat is running.

For connecting java application with the mysql database, following steps are needed to follow -

1. **Driver class :** The driver class for the mysql database is **com.mysql.jdbc.Driver**. Here we can instantiate the object for JDBC driver by using following statement.

```
Class.forName("com.mysql.jdbc.Driver");
```

2. **Connection URL :** The connection URL for the mysql is

```
databases jdbc:mysql://localhost:3306/my_database
```

Where

jdbc is the API,

mysql is the database,

localhost is the server name on which mysql is running, we may also use IP address,

3306 is the port number

my_database is the database name. (We may use any database, in such case, you need to replace the my_database with your database name.)

Username : The default username for the mysql database is **root**. If you haven't set any user name it could be blank.

Password : Password is given by the user at the time of installing the mysql database.

- The connection URL can be obtained using the method **getConnection**. This method is defined in class named **DriverManager**. Note that this **DriverManager** class is defined in **java.sql.*** package. We can obtain the instance for connection


```
conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/my_database","","");
```

Let us understand this with the help of examples.

Example 3.14.2 *Write a Servlet program to connect to a MySQL database. Create a student database named StudentDB.*

Solution :

1. Create a StudentDB in MySQL using following query

```
create database StudentDB;
```

2. The Servlet program can be written as follows

```
import java.io.*;
import java.sql.Connection;
import java.sql.DriverManager;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ServletDemo extends HttpServlet {
 public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  try {
      Class.forName("com.mysql.jdbc.Driver");
      Connection con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/StudentDB","","");
      Statement stmt = con.createStatement();
      out.print("You are successfully Connected to Database...");
  }
  catch (SQLException e) {
     throw new ServletException("Could not display records.", e);
  }
  catch (ClassNotFoundException e) {
      throw new ServletException("JDBC Driver not found.", e);
  }
  out.close();
 }
}
```

Example 3.14.3 *Create a students database in any relational database package. The table which you create in this database must have following fields - Roll_No, name, address and phone. The Roll_No must be a primary key. You have to create a servlet at the fornt end which will perform following operations - 1. Insertion of record 2. Deletion of the record using the primary key 3. Updation on any one field of the record. 4. Display of all the records from the database.*

Solution : We will perform following steps -

Step 1 : First of all we will create a database named **students** and then create a table named **students_table** in this database in MYSQL. This can be down using following SQL commands -

```
mysql>CREATE DATABASE students;
mysql>use students;
mysql>create table students_table(Roll_no INT(4) NOT NULL AUTO_INCREMENT,name
VARCHAR(50) NOT NULL,address VARCHAR(30) NOT NULL,phone VARCHAR(11) NOT
NULL,PRIMARY KEY(Roll_no));
```

Now we have simply created a table without any content within it. Our intention is to insert the records in this table using the front end - servlet program.

Step 2 : The first page of this application will be **Databasedemo.html.** The code for this html document will be as given below -

DatabaseDemo.html

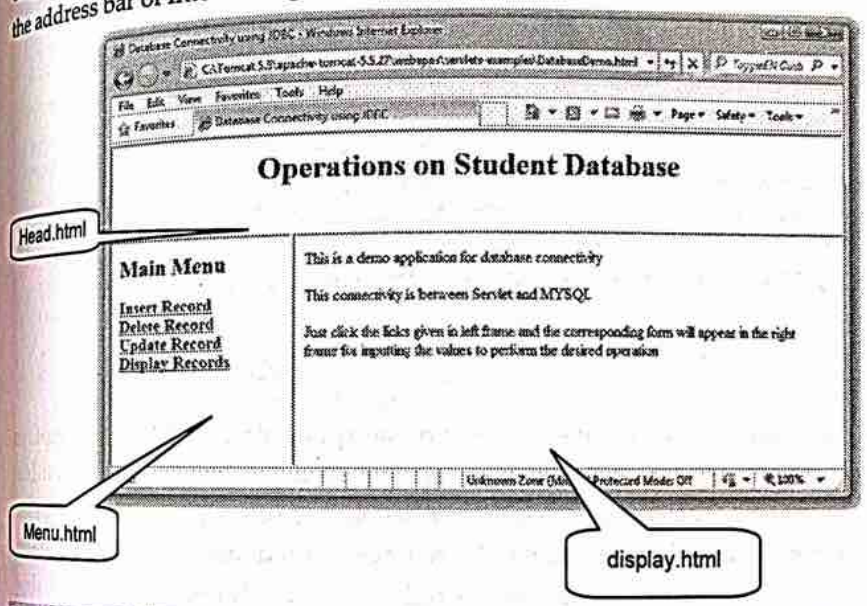
```
<html>
<head>
<title>Database Connectivity using JDBC</title>
</head>
<frameset rows="100,*">
 <framesrc="http://www.localhost.com:4040/servlets-examples/Head.html"
name="top_frame">
 <frameset cols="200,*">
  <framesrc="http://www.localhost.com:4040/servlets-examples/Menu.html"
name="left_frame">
  <framesrc="http://www.localhost.com:4040/servlets-examples/display.html"
name="right_frame">
 </frameset>
</frameset>
</html>
```

The **Databasedemo.html** will simply provide the initial interface for this application. This HTML document consists of three frames. The top frame will simply display the name of the application. The left frame consists of another html page in which the four

links for each operation are specified. Each link will be a separate servlet program communication with the database. The right frame will display the results of each operation.

The output will be as follows -

Step 1 : Open the DatabaseDemo.html web page by giving the path for this html file on the address bar of Internet explorer -



Head.html

```
<html>
 <head>
   <title>Database Connectivity</title>
 </head>
 <body>
   <center><h1>Operations on Student Database</h1></center>
 </body>
</html>
```

Menu.html

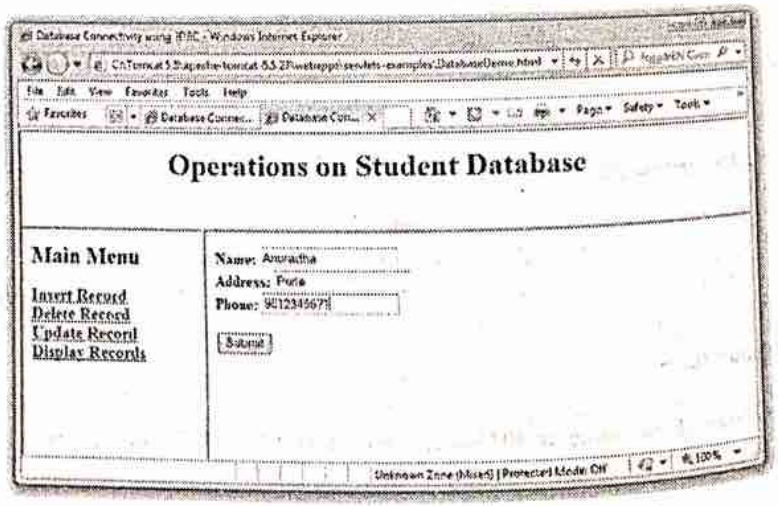
```
<html>
<body>
<h2> Main Menu</h2>
<h3>
<a href="http://www.localhost.com:4040/servlets-examples/servlet/Insert_Query"
target="right_frame">Insert Record</a>
<br/>
```

```
<a href="http://www.localhost.com:4040/servlets-examples/servlet/Delete_Query"
target="right_frame">Delete Record</a>
<br/>
<a href="http://www.localhost.com:4040/servlets-examples/servlet/Update_Query"
target="right_frame">Update Record</a>
<br/>
<a href="http://www.localhost.com:4040/servlets-examples/servlet/ServletDBDemo"
target="right_frame">Display Records</a>
</h3>
</body>
</html>
```

display.html

```
<html>
<body>
<p> This is a demo application for database connectivity</p>
<p>This connectivity is between Servlet and MYSQL</p>
<p>Just click the links given in left frame and the corresponding form will
appear in the right frame for inputting the values to perform the desired operation
</p>
</body>
</html>
```

Step 3 : The top frame simply displays the name of the application the left frame displays the various operations that can be performed on the database. There are four sample operations like insertion, deletion, updation and displaying of data from the database. Just click on corresponding links to get the desired operation done. For instance I have clicked on **Insert Record** hyperlink and in the right frame a form appears. Fill up this form and then click the Submit button.



A servlet named **Insert_Query.java** is for inserting a record in the database. It is as follows -

```
Insert_Query.java
import java.io.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Insert_Query extends HttpServlet
{
  public void service(HttpServletRequest request,HttpServletResponse response)
  throws IOException, ServletException
  {
   response.setContentType("text/html");
   PrintWriter out = response.getWriter();
   out.println("<html>");
   out.println("<head><title>Servlet Database Connectivity</title></head>");
   out.println("<body>");
   out.println("</body></html>");
   // connecting to database
   Connection con = null;
   Statement stmt = null;
   ResultSet rs = null;
   try
   {

    Class.forName("com.mysql.jdbc.Driver");
    con=DriverManager.getConnection("jdbc:mysql://localhost:3306/students","root", "system");
    stmt = con.createStatement();

    out.println("\n\n");
    String Studname=request.getParameter("Sname");
    String Studaddress=request.getParameter("Sadd");
    String Studphone=request.getParameter("Sph");
      if(Studname!=null&&Studaddress!=null&Studphone!=null)
      if(Studname.length()>0)
    stmt.executeUpdate("insert into students_table(name,address,phone)
    values("+Studname+"','"+Studaddress+"','"+Studphone+"')");
```

```
rs=stmt.executeQuery("select * from students_table");
out.println("<html>");
out.println("<body>");
out.println("<form action='Insert_Query' method='post'>");
out.println("<strong>Name:  </strong><input name='Sname' type='text'>");
out.println("<br/>");
out.println("<strong>Address: </strong><input name='Sadd' type='text'>");
out.println("<br/>");
out.println("<strong>Phone: </strong><input name='Sph' type='text'>");
out.println("<br/><br/>");
out.println("<input type='submit' value='Submit'>");
out.println("</form>");
   out.println("</body>");
out.println("</html>");

}catch (SQLException e)
{
throw new ServletException("Servlet Could not display records.", e);
}
catch (ClassNotFoundException e)
{
  throw new ServletException("JDBC Driver not found.", e);

}
 finally
 {
   try
   {
    if(rs != null)
    {
     rs.close();
     rs = null;
    }
    if(stmt != null)
    {
     stmt.close();
     stmt = null;
    }
    if(con != null)
    {
     con.close();
     con = null;
    }
   }
```



```
   catch (SQLException e) {}
   }//end of finally
   out.close();
  }//end of service function
 } //end of class
```

Now save this program at the path

```
C://your_tomcat_directory/webapps/servlets-examples/WEB-INF/classes
```

Then compile the above code using following command at this path -

```
javac Insert_Query.java
```

If the code is error free then a class file named **Insert_Query.class** gets generated at the above mentioned path.

Now edit the **web.xml** file present in the **../servlets-examples/WEB-INF** directory as follows [Note that at two fragments of code must be added in the web.xml file which are given below]

```
<servlet>
   <servlet-name>Insert_Query</servlet-name>
   <servlet-class>Insert_Query</servlet-class>
</servlet>
...
...
...
<servlet-mapping>
   <servlet-name>Insert_Query</servlet-name>
   <url-pattern>/servlet/Insert_Query</url-pattern>
</servlet-mapping>
```

Save and close this web.xml file.

The other operations can be performed using following servlet programs. What you need to do is create the following files in Notepad, execute them individually using **javac** command and then edit the **web.xml** file by specifying the name of corresponding servlet.

Delete_Query.java

```
import java.io.*;
import java.lang.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
blic class Delete_Query extends HttpServlet

public void service(HttpServletRequest request,HttpServletResponse response)
throws IOException, ServletException,NumberFormatException
{
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 out.println("<html>");
 out.println("<head><title>Servlet Database Connectivity</title></head>");
 out.println("<body>");
 out.println("</body></html>");
 // connecting to database
  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;
  try
   {
    Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection ("jdbc:mysql://localhost:3306/students","root",
"system");
     stmt = con.createStatement();
     out.println("\n\n");
      String Studroll=request.getParameter("SRoll");
      if(Studroll==null)//this parsing is done to
     Studroll="0";//avaiod NumberFormatException
     int n=Integer.parseInt(Studroll);//converting text to int
     stmt.executeUpdate("delete from students_table where Roll_no="+n);
      rs=stmt.executeQuery("select * from students_table");
      out.println("<html>");
      out.println("<body>");
      out.println("<form action='Delete_Query' method='post'>");
      out.println("<strong>Enter Roll Number for deletion : </strong>");
      out.print("<br/>");
     out.print("<input name='SRoll' type='text'>");
      out.println("<br/><br/>");
      out.println("<input type='submit' value='Submit'>");
      out.println("</form>");
     out.println("</body>");
      out.println("</html>");

     }catch (SQLException e)
     {
      throw new ServletException("Servlet Could not display records.", e);
```



```
 }
 catch (ClassNotFoundException e)
 {
  throw new ServletException("JDBC Driver not found.", e);
 }

 finally
 {
  try
  {
   if(rs != null)
   {
    rs.close();
    rs = null;
   }
   if(stmt != null)
   {
    stmt.close();
    stmt = null;
   }
   if(con != null)
   {
    con.close();
    con = null;
   }
  }
  catch (SQLException e) {}
 }//end of finally
 out.close();
}//end of service function
} //end of class
```

Update_Query.java

```
import java.io.*;
import java.lang.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
public class Update_Query extends HttpServlet
{
 public void service(HttpServletRequest request,HttpServletResponse response)
 throws IOException, ServletException,NumberFormatException
 {
 response.setContentType("text/html");
 PrintWriter out = response.getWriter();
 out.println("<html>");
 out.println("<head><title>Servlet Database Connectivity</title></head>");
 out.println("<body>");
 out.println("</body></html>");
 // connecting to database
 Connection con = null;
 Statement stmt = null;
 ResultSet rs = null;
 try
 {
  Class.forName("com.mysql.jdbc.Driver");
  con=DriverManager.getConnection("jdbc:mysql://localhost:3306/students","root", "system");
  stmt = con.createStatement();
  out.println("\n\n");
  String Studroll=request.getParameter("SRoll");
  String Studadd=request.getParameter("Sadd");
   if(Studroll==null)//this parsing is done to
   Studroll="0";//avoid NumberFormatException
  int n=Integer.parseInt(Studroll);//converting text to int
  stmt.executeUpdate("update students_table set address='"+Studadd+"' where Roll_no='"+n;
   rs=stmt.executeQuery("select * from students_table");
   out.println("<html>");
   out.println("<body>");
   out.println("<form action='Update_Query' method='post'>");
   out.println("<strong>Enter Roll Number for updation : </strong>");
   out.print("<br/>");
      out.print("<input name='SRoll' type='text'>");
   out.print("<br/>");
   out.println("<strong>Enter new address : </strong>");
   out.print("<br/>");
      out.print("<input name='Sadd' type='text'>");

   out.println("<br/><br/>");
   out.println("<input type='submit' value='Submit'>");
   out.println("</form>");
   out.println("</body>");
```



```
   out.println("</html>");

  }catch (SQLException e)
  {
  throw new ServletException("Servlet Could not display records.", e);
  }
  catch (ClassNotFoundException e)
  {
  throw new ServletException("JDBC Driver not found.", e);
  }

  finally
  {
   try
   {
    if(rs != null)
    {
     rs.close();
     rs = null;
    }
    if(stmt != null)
    {
     stmt.close();
     stmt = null;
    }
    if(con != null)
    {
     con.close();
     con = null;
    }
   }
   catch (SQLException e) {}
  }//end of finally
  out.close();
 }//end of service function
} //end of class
```

ServletDBDemo.java

```
import java.io.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ServletDBDemo extends HttpServlet {
 public void service(HttpServletRequest request,HttpServletResponse response)
  throws IOException, ServletException{
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  out.println("<html>");
  out.println("<head><title>Servlet Database Connectivity</title></head>");
  out.println("<body>");
  out.println("</body></html>");
  // connecting to database
  Connection con = null;
  Statement stmt = null;
  ResultSet rs = null;
  try {
   Class.forName("com.mysql.jdbc.Driver");
   con =DriverManager.getConnection ("jdbc:mysql://localhost:3306/students","root", "system");
   stmt = con.createStatement();
   out.println("Connected to Database...");
   out.println("\n\n");
   rs = stmt.executeQuery("SELECT * FROM students_table");
   // displaying records
   out.print("<center>");
   out.print("<h2>Students Database</h2>");
   out.print("<table border=3>");
   out.print("<th>Roll</th>");
   out.print("<th>Name</th>");
   out.print("<th>Address</th>");
   out.print("<th>Phone</th>");
   while(rs.next())
   {
   out.print("<tr>");
   out.print("<td>");
   out.print(rs.getObject(1).toString());
   out.print("</td>");
   out.print("<td>");
   out.print(rs.getObject(2).toString());
   out.print("</td>");
   out.print("<td>");
   out.print(rs.getObject(3).toString());
   out.print("</td>");
```



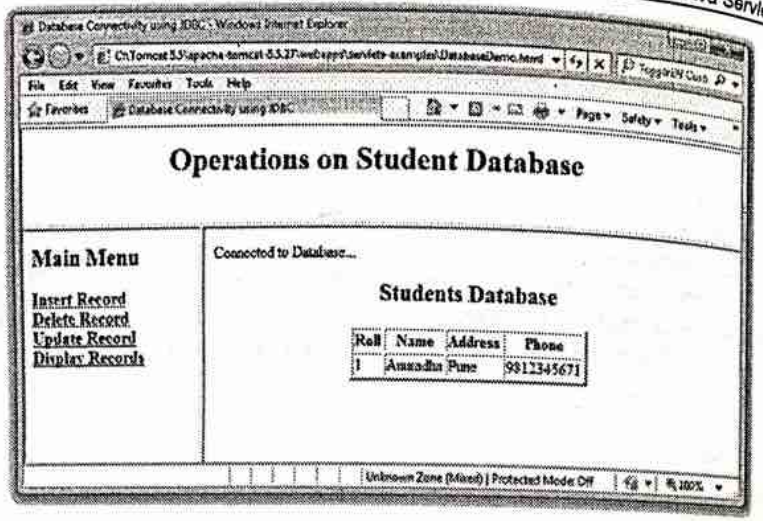
```
   out.print("<td>");
   out.print(rs.getObject(4).toString());
   out.print("</td>");
   out.print("</tr>");
   }
   out.print("</table>");
   out.print("</center>");
   }catch (SQLException e) {
   throw new ServletException("Servlet Could not display records.", e);
   } catch (ClassNotFoundException e) {
   throw new ServletException("JDBC Driver not found.", e);
   } finally {
   try {
    if(rs != null) {
     rs.close();
     rs = null;
    }
    if(stmt != null) {
     stmt.close();
     stmt = null;
    }
    if(con != null) {
     con.close();
     con = null;
    }
   } catch (SQLException e) {}
   }
   out.close();
  }
}
```

Key Point *All the above servlet code must be compiled in order to generate the class file. Note that all these class files must be present in the folder*

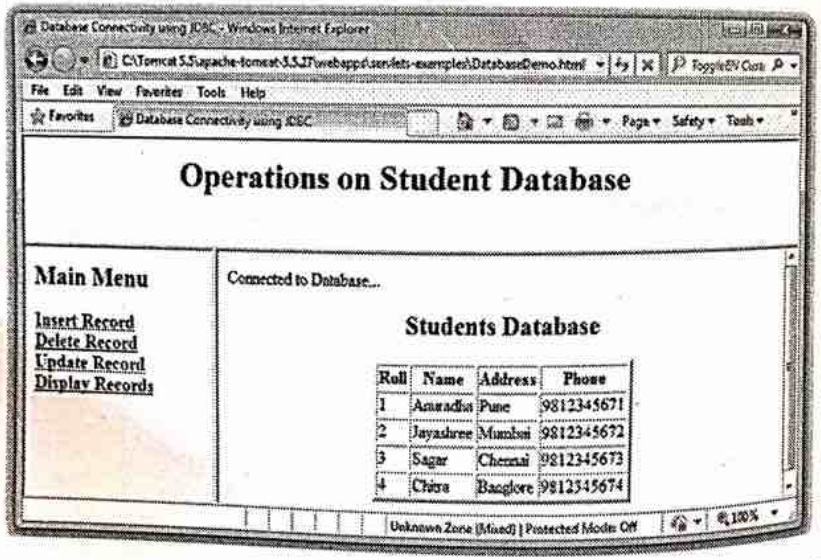
C:\your_tomcat_directory\webapps\servlets-examples\WEB-INF\classes

And the web.xml file must be edited by specifying the class file of each of these servlets.

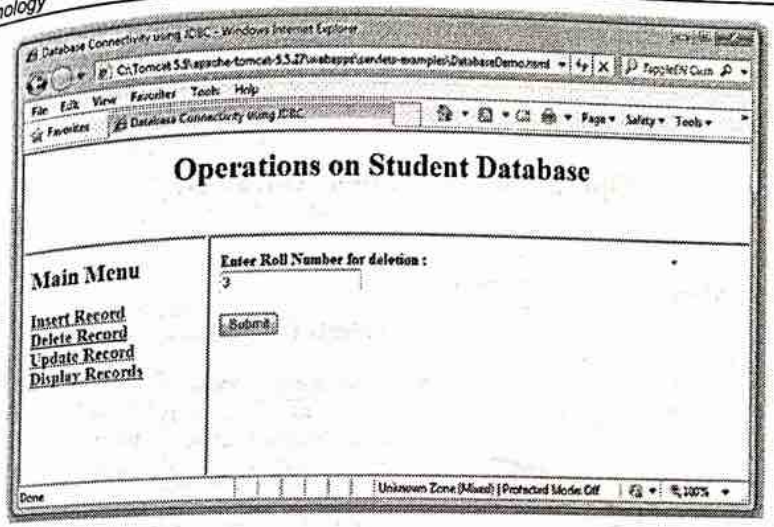
Step 3 : Click on the hyperlink **Display Records** present in the left frame. The contents from the database will be displayed in the right frame. The sample output is -



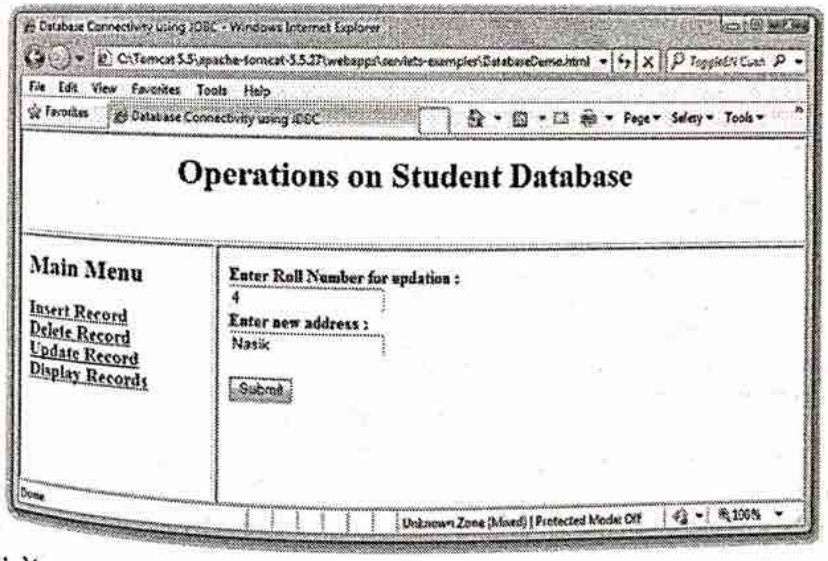
Step 4 : We can insert as many records as we want by clicking the **Insert Record** and filling up the form.



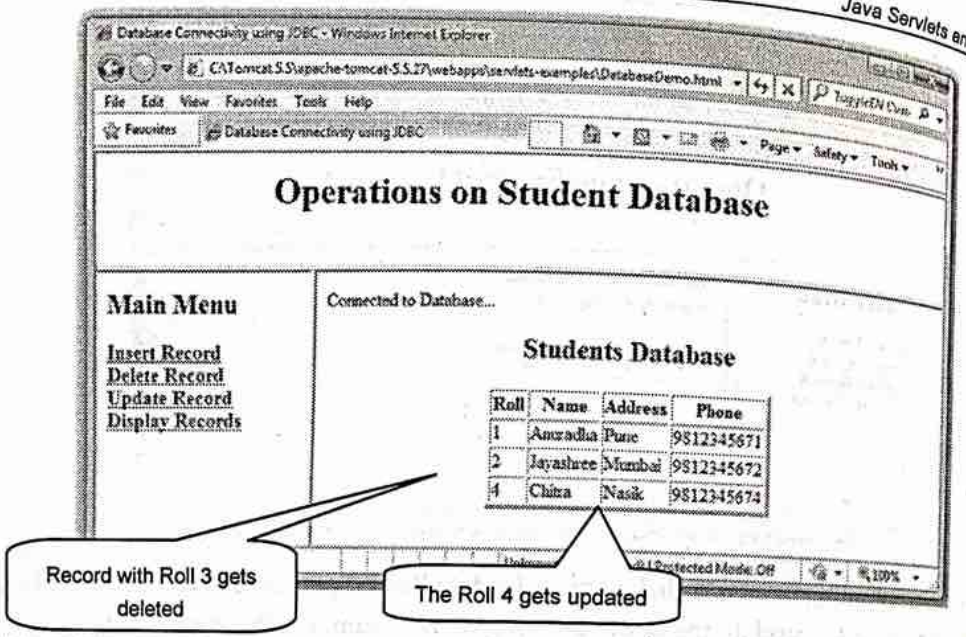
Step 5 : If we want to delete particular record then using the primary key **Roll_No** the corresponding record can be deleted. Just click on the **Delete Record** link present in the left frame and enter the roll number for the record to be deleted. And then click the Submit button. The sample output is follows -



Step 6 : For updating of record click the link **Update Record** present in the left frame. This application allows to update the address using the Roll number. The sample output will be as follows -



Step 7 : You can each time click on the **Display Records** link to see the effects of these operations. The sample output will be -



Example 3.14.4 *Consider a database table with the following schema.(PNR No,status). Write a servlet program to display the status, given the PNR number.*

Solution : Java Program[PNRDemo.Java]

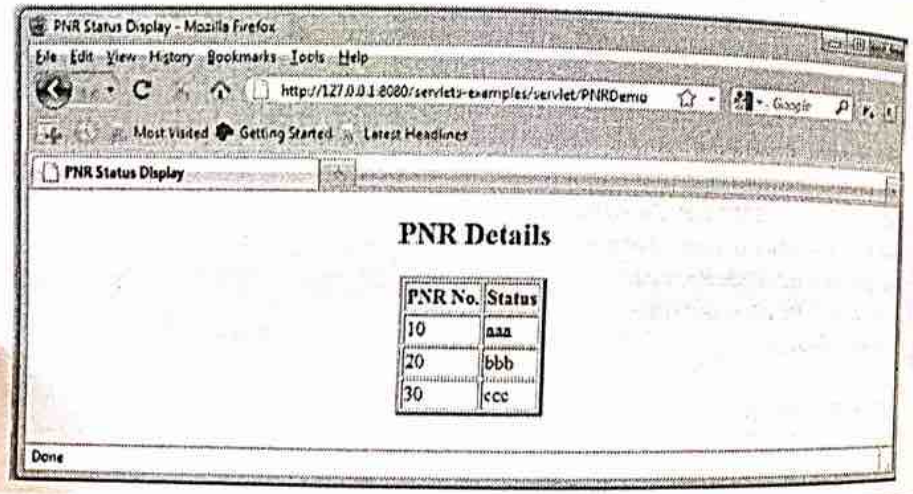
```
import java.io.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class PNRDemo extends HttpServlet
{
  public void service(HttpServletRequest request,HttpServletResponse response)
  throws IOException, ServletException
  {
   // connecting to database
   Connection con = null;
   Statement stmt = null;
   ResultSet rs = null;
   PrintWriter out = response.getWriter();
```



```
try
{
 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
 con =DriverManager.getConnection ("jdbc:odbc:PNRDemo","","");
}catch(ClassNotFoundException e){e.printStackTrace();}
catch(SQLException e){e.printStackTrace();}
catch(Exception e){e.printStackTrace();}
try
{
  stmt = con.createStatement();
 rs = stmt.executeQuery("SELECT * FROM PNR_table");
 // displaying records
 response.setContentType("text/html");
  out.println("<html>");
  out.println("<head><title>PNR Status Display</title></head>");
  out.println("<body>");
 out.print("<center>");
 out.print("<h2>PNR Details </h2>");
 out.print("<table border=3>");
 out.print("<th>PNR No.</th>");
 out.print("<th>Status</th>");
 while(rs.next())
 {
 out.print("<tr>");
 out.print("<td>");
 out.print(rs.getObject(1).toString());
 out.print("</td>");
 out.print("<td>");
       out.print(rs.getObject(2).toString());
 out.print("</td>");
 out.print("</tr>");
 }
 out.print("</table>");
 out.print("</center>");
 out.println("</body></html>");
 }catch (SQLException e)  { }
finally {
 try {
  if(rs != null) {
   rs.close();
   rs = null;
  }
  if(stmt != null) {
   stmt.close();
   stmt = null;
```

```
      }
      if(con != null) {
        con.close();
        con = null;
      }
    } catch (SQLException e) {}
  }
  out.close();
 }//end of service function
} //end of class
```

Output



Example 3.14.5 *Assume that a database has a table Employee with two columns EmployeeID and Name. Assume that the administrator user id and password to access the database table are, scott and tiger. Write a JDBC program that can query and print all the entries in the table Employee. Make the database connection using a type2 driver database.driver and connection string jdbc:db:oci*

Solution : The JDBC code for displaying the employees in Employee table is as follows :

DisplayEmp.Java

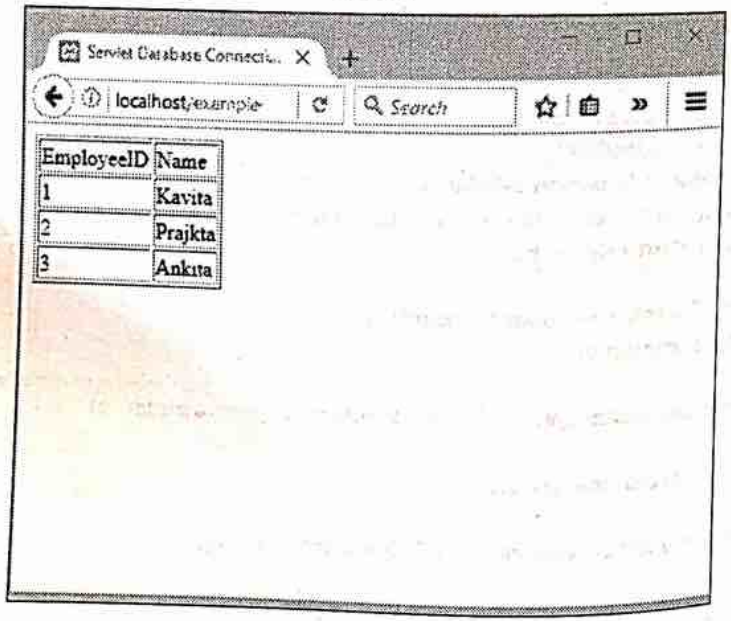
```
import java.io.*;
import java.util.*;
import javax.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;

public class DisplayEmp extends HttpServlet
```



```
{
 public void service(HttpServletRequest request,HttpServletResponse response)
    throws IOException, ServletException

 {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  out.println("<html>");
  out.println("<head><title>Servlet Database Connectivity</title></head>");
  out.println("<body>");
  // connecting to database
  Connection con = null;
  Statement stmt = null;
  ResultSet rs=null;
  try
  {

   Class.forName("oracle.jdbc.OracleDriver");
   String dbURL = "jdbc:oracle:oci:@my_database";
     Properties properties = new Properties();
   properties.put("user", "scott");
   properties.put("password", "tiger");
   properties.put("defaultRowPrefetch", "20");
   con =DriverManager.getConnection (dbURL,properties);
   stmt = con.createStatement();
   rs=stmt.executeQuery("select * from Employee");
   out.print("<table border=1>");
   out.print("<tr><td>EmployeeID</td><td>Name</td></tr>");
   while(rs.next())
   {
    //Retrieve by column name
    int id  = rs.getInt("ID");
    String name = rs.getString("Name");
    out.print("<tr><td>"+id+"</td><td>"+name);
    out.print("</td></tr>");
   }
   out.println("</table></body></html>");
  }catch (SQLException e)
  {

   throw new ServletException("Servlet Could not display records.", e);
  }
  catch (ClassNotFoundException e)
  {

   throw new ServletException("JDBC Driver not found.", e);
  }
```

```
    finally
    {
      try
      {
        if(rs != null)
        {
          rs.close();
          rs = null;
        }
        if(stmt != null)
        {
          stmt.close();
          stmt = null;
        }
        if(con != null)
        {
          con.close();
          con = null;
        }
      }
      catch (SQLException e) {}
    }//end of finally
    out.close();
  }//end of service function
} //end of class
```

Output



Part II : XML

3.15 XML Documents and Vocabularies

SPPU : Dec.-18, 19, May-19, Marks 5

- XML stands for eXtensible Markup Language.
- This scripting language is similar to HTML. That means, this scripting language contains various tags. But these tags are not predefined tags, in-fact user can define his own tags.
- Thus HTML is designed for **representation of data** on the web page whereas the XML is designed for **transport or to store data.**

Uses of XML

1. XML is used to display the meta contents i.e. XML describes the content of the document.
2. XML is useful in exchanging data between the applications.
3. The data can be extracted from database and can be used in more than one application. Different applications can perform different tasks on this data.

Advantages of XML

1. XML document is human readable and we can edit any XML document in simple text editors.
2. The XML document is language neutral. That means a Java program can generate an XML document and this document can be parsed by Perl.
3. Every XML document has a tree structure. Hence complex data can be arranged systematically and can be understood in simple manner.
4. XML files are independent of an operating system.

Goals of XML

Following are the goals of XML -

1. User must be able to define and use his own tag. This allows us to restrict the use of the set of tags defined by proprietary vendors.
2. Allow user to build his own tag library based on his web requirement.
3. Allow user to define the formatting rules for the user defined tags.
4. XML must support for storage or transport of data.

Features of XML

Following are some features which are most suitable for creating web related applications.

- XML is EXtesible Markup Language intended for **transport** or storage of the data.
- The most important feature of XML is that user is able to **define and use his own tag.**
- XML contains only data and does not contain any formatting information. Hence document developers **can decide how to display the data.**
- XML permits the document writer to create the tags for any type of information. Due to this virtually any kind of information can be such as mathematical formulae, chemical structures, or some other kind of data can be described using XML.
- Searching sorting, rendering or manipulating the XML document is possible using Extended Stylesheet Language (XSL).
- The XML document can be validated using some external tools.
- Some commonly used web browsers like Internet Explorer and Firefox Mozilla provide support to the tags in XML. Hence XML is not at all vendor specific or browser specific.

Difference between XML and HTML

Sr. No.	HTML	XML
1.	HTML stands for Hypertext Markup Language.	XML stands for eXtensible Markup Language.
2.	HTML is designed to display data with the focus on look and feel of data.	XML is used to transport and store data, with focus on what data is.
3.	HTML is case insensitive.	XML is case sensitive.
4.	HTML has predefined tags.	XML has custom tags can be defined and the tags are invented by the author of the XML document.
5.	As HTML is for displaying the data it is static.	As XML is for carrying the data it is dynamic.
6.	It can not preserve white space.	It can preserve the white space.

Strengths and Limitations

Strengths :

1. XML document is human readable and we can edit any XML document in simple text editors.
2. The XML document is language neutral. That means a Java program can generate an XML document and this document can be parsed by Perl.

3. Every XML document has a tree structure. Hence complex data can be arranged systematically and can be understood in simple manner.
4. XML files are independent of an operating system.

Limitations :

1. XML syntax is verbose and redundant as compared to other text based data transmission formats.
2. Redundancy in syntax of XML causes higher storage and transportation cost when there is large volume of data.
3. XML does not support array.
4. XML document is less readable.
5. Due to verbose nature of XML syntax the XML file size is usually very large.

Review Questions

1. *What are the strengths of XML technology ? Also list the limitations of using XML.* **SPPU : Dec.-18, Marks 5**
2. *What are the strengths of XML technology ? Explain the need for XML.* **SPPU : May-19, Marks 5**
3. *Describe the following web technology : XML.* **SPPU : Dec.-19, Marks 2**

3.16 XML Declaration

Elements and Attributes

- Various building blocks of XML are -

1. Elements

- The basic entity is **element.** The elements are used for defining the tags. The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.

2. Attribute

- The attributes are generally used to specify the values of the element. These are specified within the double quotes.

For example -

```
<flag type="True">
```

- The type attribute of the element flag is having the value True.

3. CDATA

CDATA stands for Character Data. This character data will be parsed by the parser.

4. PCDATA

It stands for Parsed Character Data (i.e. text).

3.16.1 Declaring Elements

- The element type data can be declared using the **ELEMENT**.
- The syntax for defining the ELEMENT is,

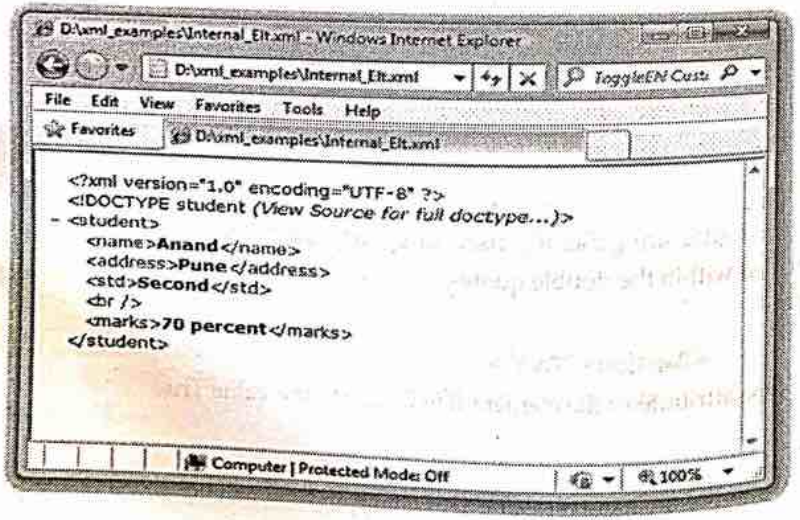
```
<!ELEMENTelt_name (elt1,elt2,..)>
```

XML Document [Internal_Elt.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name,address,std,br,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENTstd (#PCDATA)>
<!ELEMENTbr EMPTY>
<!ELEMENT marks (#PCDATA)>
]>
<student>
    <name>Anand</name>
    <address>Pune</address>
    <std>Second</std>
    <br />
    <marks>70 percent</marks>
</student>
```

Here we are defining the DTD internally

Output



- The ELEMENT must be defined in the beginning of the XML file. It must be enclosed within *<!DOCTYPEroot_element ....>*
- There are various ways by which the actual element can be **defined**. It is enlisted in the following table -

Category	Syntax	Description
Empty	<!ELEMENT elt_nameEMPTY>	Empty elements can be declared.
Character data	<!ELEMENT elt_name (#PCDATA)>	Character data can be declared.
Any content	<!ELEMENT elt_nameANY>	Any combination of data can be declared.
Sequence	<!ELEMENTelt_name(elt1,elt2,...)>	Sequence of elements that must appear in specific order.

- Sometimes we can define the ELEMENT along with the iterator characters. The iterator characters are the special character which are associated with some meaning.
- Some commonly used iterator characters are as given below -

Iterator character	Meaning	Example
?	The preceding choice appears zero or once.	<!ELEMENT(marks?)> The child element marks can appear for zero or one time.
*	The preceding choice appears for any number of time(zero or more times).	<!ELEMENT (person*)> The person can appear for any number of times.
+	The preceding choice appears for at least once (one or more times).	<!ELEMENT (person+)> The person can appear for at least once.

3.16.2 Declaring Attributes

- The attribute type declaration is done using the string ATTLIST.
- The attribute type specifies the type of data which can be used for specifying the attribute. The syntax for specifying the attribute declaration is as follows -

```
<!ATTLISTelement_nameattribute_nameattribute_typedefault_value>
```

- **For example**

```
<!ATTLIST address phone CDATA #REQUIRED>
```

Element_name → address; Attribute_type → phone; Attribute_name → CDATA; Default_value → #REQUIRED

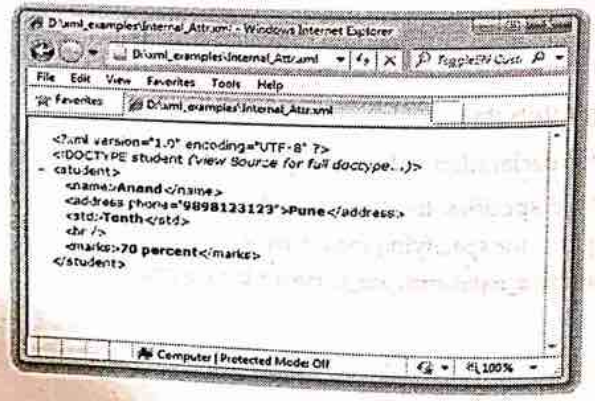
- The XML document containing attributes is as given below -

XML Document [Internal_Attr.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name,address,std,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENTstd (#PCDATA)>
<!ELEMENT marks (#PCDATA)>

<!ATTLIST address phone CDATA #REQUIRED>

]>
<student>
    <name>Anand</name>
    <address phone="9898123123">Pune</address>
    <std>Tenth</std>
    <br />
    <marks>70 percent</marks>
</student>
```

Using **ATTLIST** the attribute can be defined.

Output



- Following are some **attribute type** -

Attribute_Type	Syntax	Description
Character data	CDATA	Attribute value is character data.
Enumerated	(str1|str2|...)	Attribute value must be one from an enumerated list.
Identifier	ID	Attribute value is a unique id.
Identifier reference	IDREF	Attribute value is the id of another element.
Identifier reference list	IDREFS	Attribute value is a list of other ids.
Name token	NMTOKEN	Attribute value is a valid XML name.
Name token list	NMTOKENS	Attribute value is a list of valid XML names.

The default_values are -

Default_Value	Meaning
Value	Default value can be defined. For example - <!ATTLIST Index flag="0"> In XML document <Index flag="1" /> If the **flag** is not assigned with any value then by default it will be 0.
#REQUIRED	Required attribute forces the attribute to be present even if there is no default value.
#IMPLIED	The default value is not provided and attribute can be optionally used.
#FIXED value	The attribute value is fixed and cannot be changed.

3.16.3 Declaring Entities

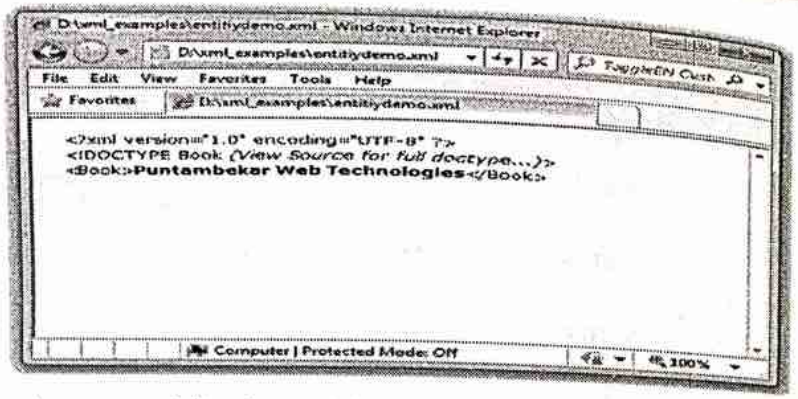
- The attribute type declaration is done using the string **ENTITY**.
- Inside the XML file at the beginning the entity can be defined using the <!DOCTYPE ...> and with the help of string ENTITY. Following XML document demonstrates it -

XML Document [EntityDemo.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Book[
<!ENTITY writer "Puntambekar">
<!ENTITYnbsp " ">
<!ENTITY title "Web Technologies">
```

```
]>
<Book>&writer; &title;</Book>
```

Output



- Note that we can define the predefined entities ** ** along with the other entity declaration.
- The entity can be used using following syntax.

```
&name_of_entity;
```

- In above script we have used two entities namely **writer** and **title**. For displaying the space within these two entities we have used an entity **nbsp** and for defining it we have used **#160**.

3.17 Rules

- **Rules that must be followed while writing XML -**

  Here are some rules that must be followed while writing the XML programs -

  1. XML is a **case sensitive**. For example -

```
<hobby>I like drawing</Hobby>or
<Hobby>I like drawing</hobby>
```

  is not allowed because the words **hobby** and **Hobby** are treated differently.

  2. In XML each **start tag** must have **matching end tag**. For example -

```
<staring> XML is funny to write
<simple> It is simple to implement </simple>
```

  That means a closing tag is a must.

  3. The elements in XML must be **properly nested**. For example-

```
<one><two>Hello how are you?</one></two>   is wrong but
<one><two>Hello how are you?</two></one>   is correct
```

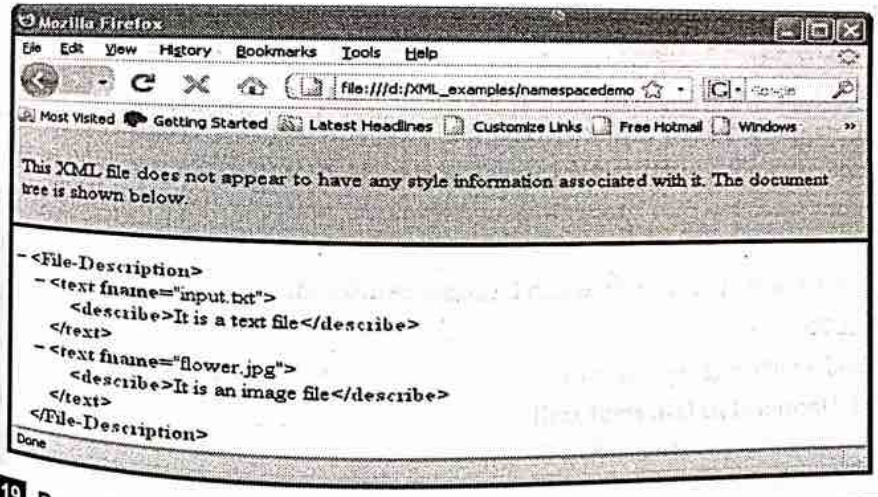

3.18 XML Namespaces

- Sometimes we need to create two different elements by the same name. The xml document allows us to create **different elements** which are having the **common name**. This technique is known as **namespace**.
- In some web documents it becomes necessary to have the same name for two different elements. Here different elements mean the elements which are intended for different purposes. In such a case support for namespace technique is very much helpful.
- For example : Consider the following xml document -

namespacedemo.xml

```
<File-Description>
  <text fname="input.txt">
      <describe>It is a text file</describe>
  </text>
  <text fname="flower.jpg">
      <describe>It is an image file</describe>
  </text>
</File-Description>
```

The above document does not produce any error although the element **text** is used for two different attribute values. The output will be,



3.19 Document Type Definition (DTD)

SPPU : May-18, Marks 5

- The document type definition is used to define the basic building block of any xml document.
- Using DTD we can specify the various elements types, attributes and their relationship with one another.

- Basically DTD is used to specify the set of rules for structuring data in any XML file.
- **For example :** If we want to put some information about some students in XML file then, generally we use tag **student** followed by his/her **name, address, standard and marks.** That means we are actually specifying the manner by which the information should be arranged in the XML file. And for this purpose the document type definition is used.
- Various building blocks of XML are -

1. Elements

The basic entity is **element.** The elements are used for defining the tags. The elements typically consist of opening and closing tag. Mostly only one element is used to define a single tag.

2. Attribute

The attributes are generally used to specify the values of the element. These are specified within the double quotes.

For example -

```
<flag type="True">
```

The type attribute of the element flag is having the value *True.*

3. CDATA

CDATA stands for character data. This character data will be parsed by the parser.

4. PCDATA

It stands for Parsed Character **Data** (i.e. text). Any parsable character data should not contain the markup characters. The markup characters are < or > or &. If we want to use less than, greater than or ampersand characters then make use of <, > or &

- There are two ways by which DTD can be defined.

Internal DTD

Consider following xml document -

XML Document [DTDDemo1.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE student [
<!ELEMENT student (name,address,std,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT std (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
]>
<student>
```

Here we are defining the DTD internally


```
<name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
  <marks>70 percent</marks>
</student>
```

Output

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
- <student>
    <name>Anand</name>
    <address>Pune</address>
    <std>Second</std>
    <marks>70 percent</marks>
  </student>
```

External DTD

In this type, an external DTD file is created and its name must be specified in the corresponding XML file. Following XML document illustrates the use of external DTD.

Step 1 : Creation of **DTD file [*student.dtd*]**

Open some suitable text editor or a notepad. Type following code into it -

```
<!ELEMENT student (name,address,std,marks)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
<!ELEMENT std (#PCDATA)>
<!ELEMENT marks (#PCDATA)>
```

Now save this file as ***student.dtd***

Step 2 : Creation of XML document -

XML Document [*DTDDemo.xml*]

Now create a XML document as follows -

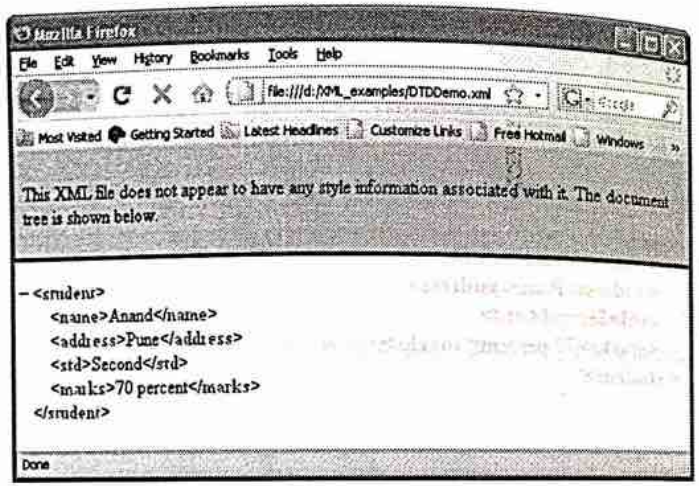
```
<?xml version="1.0"?>
<!DOCTYPE student SYSTEM "student.dtd">
<student>
  <name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
```

The external DTD file is created

```
    <marks>70 percent</marks>
</student>
```

Step 3 : Using some web browser open the XML document.

Output



Example 3.19.1 *Create a DTD for a catalog of four stroke motorbikes, where each motor bike has the following child elements -make, model, year, colo, engine, chasis number and accessories. The engine element has the child elements engine number, number of cylinders, type of fuel. The accessories element has the attributes like disc brake,auto-start and radio, each of which is required and has the possible values yes and no.*

Solution :

automobile.dtd

```
<!ELEMENT Catalog (MotorBike)*>
<!ELEMENT MotorBike (make, model, year,color, engine,chasis-num,accessories ) >
<!ELEMENT make (#PCDATA)>
<!ELEMENT model (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT color (#PCDATA)>
<!ELEMENT engine (engin-num,cylinders-num,fuel-type)>
<!ELEMENT chasis-num (#PCDATA)>
<!ELEMENT accessories (#PCDATA)>
<!ATTLIST accessories disc-brake (yes|no) #REQUIRED
                      auto-start (yes|no) #REQUIRED
                      radio (yes|no) #REQUIRED>
```


Example 3.19.2 *Consider a hospital management system. Write a DTD program to consolidate and show the bill to be paid by the inpatients (Assume your own data).*

Solution : Patients.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Patients [
<!ELEMENT Patients (Patient)*>
<!ELEMENT Patient (Name, Address,Test,Date,Doctor,Bill)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT Test (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Doctor (#PCDATA)>
<!ELEMENT Bill (#PCDATA)>
]>
<Patients>
  <Patient>
    <Name>Anuradha</Name>
    <Address>Pune</Address>
    <Test>ECG</Test>
    <Date>1-1-2010</Date>
    <Doctor>Mr.XYZ</Doctor>
    <Bill>Rs.500</Bill>
  </Patient>
  <Patient>
    <Name>Archana</Name>
    <Address>Mumbai</Address>
    <Test>Blood Test</Test>
    <Date>10-3-2010</Date>
    <Doctor>Mr.PQR</Doctor>
    <Bill>Rs.100</Bill>
  </Patient>
</Patients>
```

Merits of DTD

1. DTDs are used to define the structural components of XML document
2. These are relatively simple and compact.
3. DTDs can be defined inline and hence can be embedded directly in the XML document.

Demerits of DTD

1. The DTDs are very basic and hence cannot be much specific for complex documents.

2. The language that DTD uses is not an XML document. Hence various frameworks used by XML cannot be supported by the DTDs.
3. The DTD cannot define the type of data contained within the XML document. Hence using DTD we cannot specify whether the element is numeric, or string or of date type.
4. There are some XML processor which do not understand DTDs.
5. The DTDs are not aware of namespace concept.

Review Question

1. *Explain the differences between external and internal DTD.* **SPPU : May-18, Marks 5**

3.20 Schema

SPPU : May-18, Marks 5

- The XML schemas are used to represent the **structure of XML document.**
- The goal or purpose of XML schema is to define the building blocks of an XML document. These can be used as an alternative to XML DTD.
- The XML schema language is called as **XML Schema Definition (XSD) language.**
- XML schema defines elements, attributes, elements having child elements, order of child elements. It also defines fixed and default values of elements and attributes.
- XML schema also allows the developer to use **data types.**

How to write a simple schema ?

Step 1 : We will first write a simple xsd file in which the desired structure of the XML document is defined. I have named it as ***StudentSchema.xsd***

This file contains the complex type with four child simple type elements.

XML Schema [StudentSchema.xsd]

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
                <xs:element name="Student">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="std" type="xs:string"/>
        <xs:element name="marks" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```


Script Explanation :

1. The **xs** is qualifier used to identify the schema elements and types.
2. The document element of schema is **xs:schema.** The **xs:schema** is the root element. It takes the attribute **xmlns:xs** which has the value **http://www.w3.org/2001/XMLSchema.** This declaration indicates that document should follow the rules of XML schema. The XML schema rules are defined by the W3 recommendation in year 2001.
3. Then comes **xs:element** which is used to define the xml element. In above case the element **Student** is of complex type who have four child elements : *name, address, std* and *marks*. All these elements are of simple type **string.**

Step 2 : Now develop XML document in which the desired values to the XML elements can be given. I have named this file as **MySchema.xml**

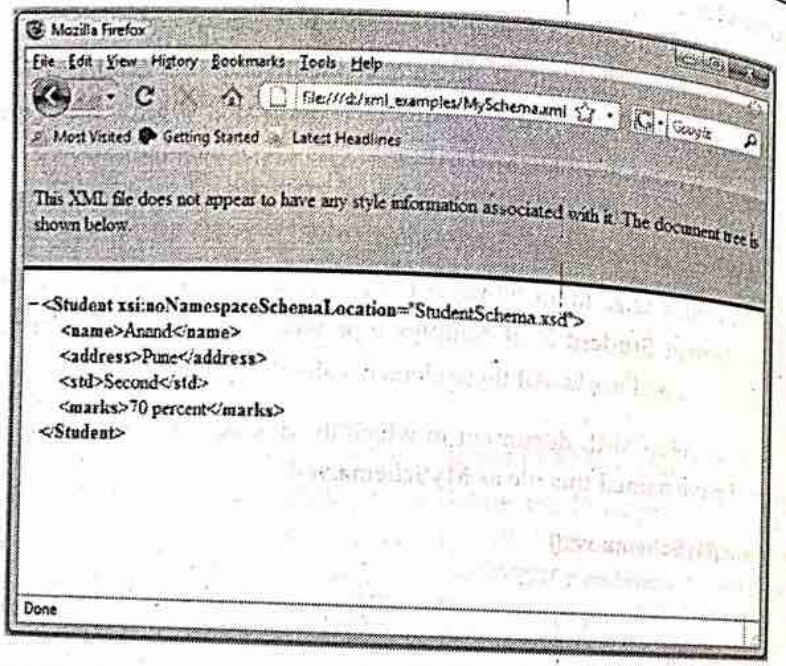
XML Document *[MySchema.xml]*

```
<?xml version="1.0" encoding="UTF-8"?>
<Student xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="StudentSchema.xsd">
<name>Anand</name>
<address>Pune</address>
<std>Second</std>
<marks>70 percent</marks>
</Student>
```

Program Explanation :

1. In above XML document, the first line is typical in which the version and encoding is specified.
2. There is **xmlns:xsi** attribute of document which indicates that XML document is an instance of XML schema. And it has come from the namespace "http://www.w3.org/2001/XMLSchema-instance."
3. To tie this XML document with the some Schema definition we use the attribute **xsi:noNamespace SchemaLocation.** The value that can be passed to this attribute is the name of xsd file "StudentSchema.xsd".
4. After this we can define the child elements.

Step 3 : See the output in the browser window as follows -



3.20.1 Data Types

Various data types that can be used to specify the data types of an element are :

- String
- Date
- Numeric
- Boolean.

Let us discuss each of these data types with the help of examples -

1. String data type

The string data type is used to define the **element** containing characters, lines, tabs or white spaces. Following schema definition illustrates this data type.

We can write a schema definition in which the data type of element Student_Name is declared as of string type.

XML Schema [stringType.xsd]

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >
        <xs:element name="Student_Name" type="xs:string" />
</xs:schema>
```


2. Date data type

For specifying the date we use date data type. The format of this date is yyyy-mm-dd where yyyy denotes the year, mm denotes the month and dd specifies the day. While specifying the date if any entry is single digit then it must be with leading zero. All the entries (i.e. yyyy,mm and dd) must be present.

Here is an illustration.

XML Schema [dateType.dtd]

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="Date_of_Birth" type="xs:date"/>
</xs:schema>
```

XML Document [dateTypeDemo.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<Date_of_Birth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dateType.xsd">
2009-09-01
</Date_of_Birth>
```

We can also specify the time by specifying the time as follows -

XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<Date_of_Birth xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="dateType.xsd">
2009-09-01-01:57
</Date_of_Birth>
```

3. Numeric

If we want to use numeric value for some element then we can use the data types as either **decimal** or **integer.**

XML Schema [decimal Type.xsd]

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="My_Marks" type="xs:decimal"/>
</xs:schema>
```

XML Document [decimal Type Demo.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<My_Marks xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="decimalDemo.xsd">
99.99
</My_Marks>
```

Using decimal data type we can specify the value that may contain some decimal point. We can have another data type called **integer** which is used to specify the numeric values that are without any decimal point.

XML Schema [integer Type.xsd]

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="My_Number" type="xs:integer"/>
</xs:schema>
```

4. Boolean

For specifying the true or false values we must use the boolean data type.

XML Schema [booleantype.xsd]

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:element name="Flag" type="xs:boolean"/>
</xs:schema>
```

3.20.2 Advantages and Disadvantages of Schema

Advantages of schema

1. The schemas are more specific.
2. The schema provide the support for data types.
3. The schema is aware of namespaces.
4. The XML schema is written in XML itself and has a large number of built in and derived types.
5. The XML schema is the W3C recommendation. Hence it is supported by various XML validator and XML processors.

Disadvantages of schema

1. The XML schema is complex to design and hard to learn.
2. The XML document cannot be if the corresponding schema file is absent.
3. Maintaining the schema for large and complex operations sometimes slows down the processing of XML document.

Advantages of schema over DTD

1. Both the schemas and DTDs are useful for defining structural components of XML. But the DTDs are basic and cannot be much specific for complex operations. On the other hand schemas are more specific.
2. The schemas provide support for defining the type of data. The DTDs do not have this ability. Hence content definition is possible using schema.
3. The schemas are namespace aware and DTDs are not.
4. The XML schema is written in XML itself and has a large number of built in and derived types.
5. The schema is the W3C recommendation. Hence it is supported by various XML validator and XML processors but there are some XML processors which do not support DTD.
6. Large number of web applications can be built using XML schema. On the other hand relatively simple and compact operations can be built using DTD.

Review Question

1. *What are XML schemas ? How are they better than DTDs ?* **SPPU : May-18, Marks 5**

3.21 DOM based XML Processing

What is DOM ?

Document Object Model (DOM) is a set of platform independent and language neutral Application Programming Interface (API) which describes how to access and manipulate the information stored in XML or in HTML documents.

Thus XML DOM is for

- Loading the XML document.
- Accessing the elements of XML document.
- Deleting the elements of XML document.
- Changing the elements of XML document.

1. Loading an XML file

We will write a simple script by which any desired XML file can be loaded. Note that these scripts are dependant upon the browser in which these are getting opened. We will write the following script which can be opened in any browser like internet explorer or Mozilla Firefox.

Step 1 : Open some simple text editor like Notepad and type the following script. I have named it as first.html

```
<html>
<!--Simple DOM example for loading some xml file -->
<body>
<script type="text/javascript">
try
 {
 xmlDocument=new ActiveXObject("Microsoft.XMLDOM");
 }
catch(e)
 {
 try
  {
  xmlDocument=document.implementation.createDocument("","",null);
  }
 catch(e) {alert(e.message)}
 }
try
 {
 xmlDocument.async=false;
 xmlDocument.load("student.xml");
 document.write("XML Document student.xml is loaded");
 }
catch(e) {alert(e.message)}
</script>
</body>
</html>
```

The **load** function is used to load the desired XML file. Note that when we specify some xml file for loading then it must be present at the current path. As an illustration the **student.xml** file is given as below -

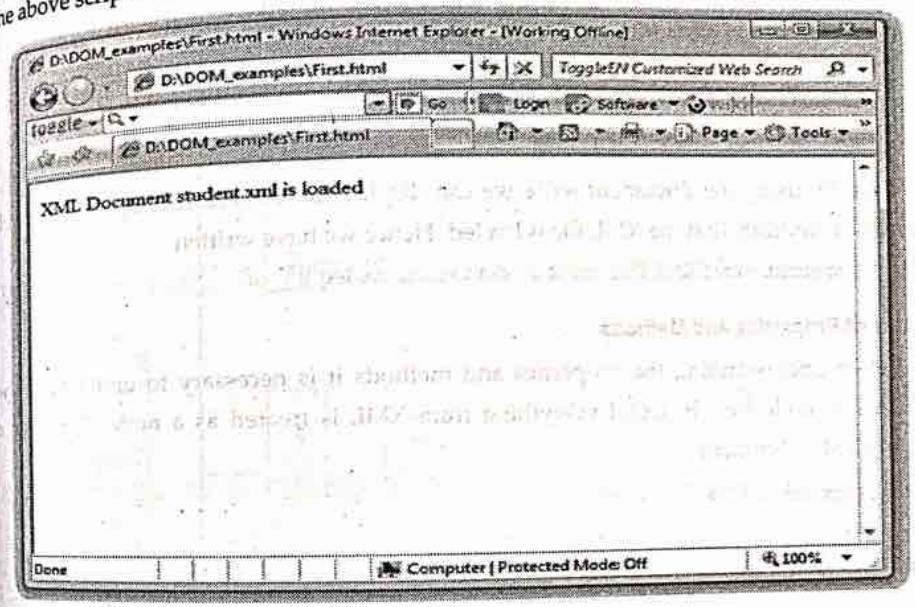
XML Document [student.xml]

```
<?xml version="1.0" ?>
<student>
   <Roll_No>10</Roll_No>
   <Personal_Info>
      <Name>Parth</Name>
      <Address>Pune</Address>
      <Phone>1234567890</Phone>
   </Personal_Info>
   <class>Second</class>
```



```
   <Subject>Mathematics</Subject>
   <Marks>100</Marks>
</student>
```

Step 2 : Then open some web browser like Firefox or Internet explorer and the output for the above script can be seen as follows -



Script Explanation

There is an XML parser which reads any XML file. All the modern web browsers like Internet explorer have built in XML parser. The parser converts the XML into javascript accessible object. Hence we have mentioned the script type as -

```
<script type="text/javascript">
```

The parser reads the XML file, stores it into a memory and converts it into the XML DOM object. These objects can be accessible by the **JavaScript**.

Internet Explorer 5 and above have the built in XML parser. Hence in above given script we have written -

```
xmlDocument=new ActiveXObject("Microsoft.XMLDOM");
```

This statement means create an empty XML document object. We have taken this object in the variable **xmlDocument**.

After getting the document loaded fully in the memory the parser will continue for execution of the script, hence we must set -

```
xmlDocument.async=false;
```

Then we should specify the name of the xml file which we want to get loaded.

```
xmlDocument.load("student.xml");
```

Then by using the **document.write** we can display the desired message on the web browser to indicate that the XML file is loaded. Hence we have written

```
document.write("XML Document student.xml is loaded");
```

2. Use of Properties and Methods

Before understanding the properties and methods it is necessary to understand the concept of node tree. In DOM everything from XML is treated as a **node**. Consider a simple XML document,

```
<?xml version="1.0" ?>
<student>
    <Roll_No>10</Roll_No>
    <Personal_Info>
      <Name>Parth</Name>
      <Address>Pune</Address>
      <Phone>1234567890</Phone>
  </Personal_Info>
  <class>Second</class>
  <Subject>Mathematics</Subject>
  <Marks>100</Marks>
</student>
```


We can draw a node tree for this document as follows -

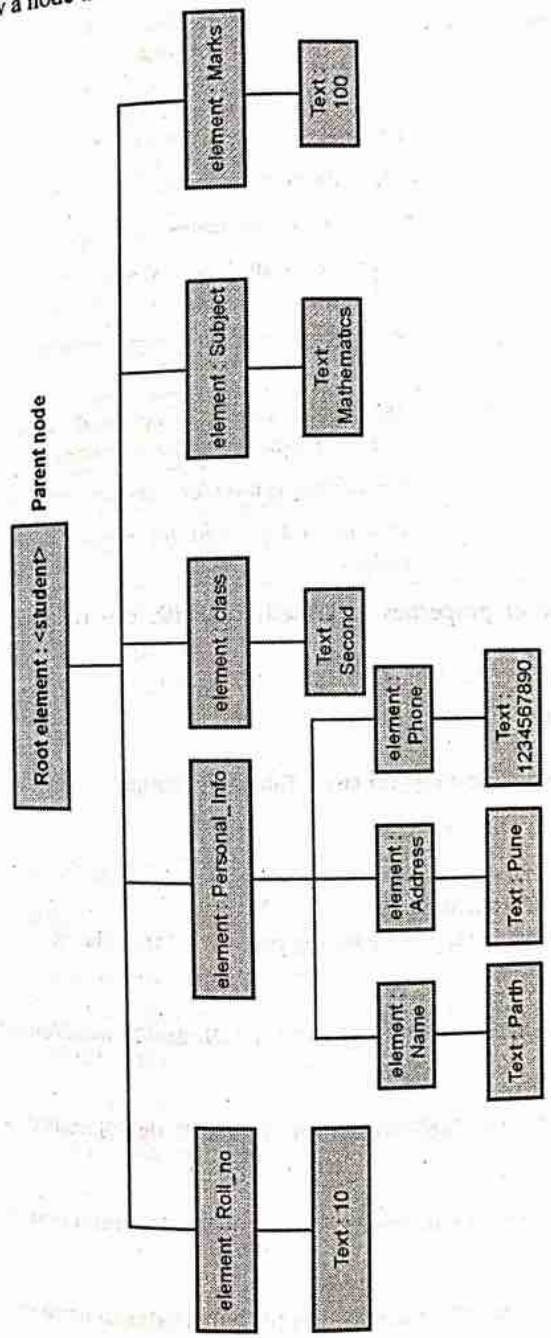


Fig. 3.21.1 Node-tree for student.XML

Various properties are,

Property	Meaning
nodeName	Finding the name of the node.
nodeValue	Obtaining the value of a node.
parentNode	Getting the parent element.
childNodes	Obtaining the child nodes.
attributes	For getting the attributes values.

Various methods are,

Method	Meaning
getElementsByTagName(name)	This method is used to access all the elements that are specified with the tag name.
appendChild(node)	This method is used for inserting a child node.
removeChild(node)	This method is used for removing the child nodes.

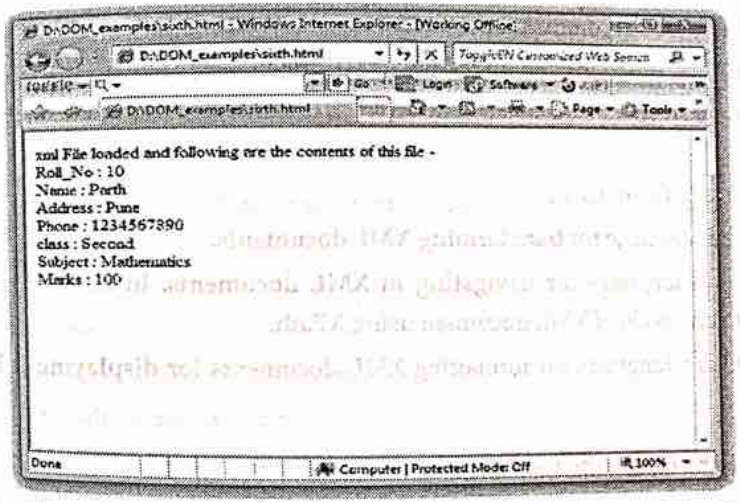
Let us understand the use of properties and methods with the help of following example -

```
<html>
<!-- loading and accessing the contents of xml file -->
<head>
      <script type="text/javascript" src="my_function_file.js"></script>
</head>
<body>
<script type="text/javascript">
   xmlDocument=My_Function("student.xml");
   document.write("xml File loaded and following are the contents of this file -");
      document.write("</br>");
   document.write("Roll_No :
   "+xmlDocument.getElementsByTagName("Roll_No")[0].childNodes[0].nodeValue);
   document.write("</br>");
   document.write("Name :
      "+xmlDocument.getElementsByTagName("Name")[0].childNodes[0].nodeValue);
   document.write("</br>");
      document.write("Address :
      "+xmlDocument.getElementsByTagName("Address")[0].childNodes[0].nodeValue);
   document.write("</br>");
      document.write("Phone :
      "+xmlDocument.getElementsByTagName("Phone")[0].childNodes[0].nodeValue);
   document.write("</br>");
```



```
      document.write("class :
      "+xmlDocument.getElementsByTagName("class")[0].childNodes[0].nodeValue);
   document.write("</br>");
      document.write("Subject :
      "+xmlDocument.getElementsByTagName("Subject")[0].childNodes[0].nodeValue);
   document.write("</br>");
      document.write("Marks :
      "+xmlDocument.getElementsByTagName("Marks")[0].childNodes[0].nodeValue);
   document.write("</br>");
      </script>
      </body>
      </html>
```

Output



Script Explanation

In above given script, we have specified an external file **my_function_file.js** in the head section. This is a file in which the function My_Function is defined. This function performs the task of loading an XML document **student.xml**

We will first obtain a document object using the statement,

```
xmlDocument=My_Function("student.xml");
```

Note that **xmlDocument** is a document object referring the document **student.xml**. Using this object we can use the properties and methods. The method **getElementsByTagName** is used and to this method the parameter will be a tag names such as "Roll_No", "Name" "class" and so on.

```
xmlDocument.getElementsByTagName("Roll_No")[0].childNodes[0].nodeValue);
```

- This method is used for accessing the element by specifying the tag name
- Tag name from **student.xml**
- The first child of the <Roll_No>
- The value of node Roll_No (i.e. text)

Hence as an output we can see the value of every element. That means the contents of **student.xml** file will be displayed.

3.22 Transforming XML Documents

SPPU : May-19, March-20, Marks 5

- The XSLT stands for **XSL** Transformations and XSL stands for eXtensible Stylesheet Language.
- An XSLT is a W3C recommendation.
- The XSLT is used for defining the XML document transformations and presentations. Hence XSL decides how an XML document should look on the web browser.
- XSL consists of three parts :
  1. XSLT is a language for **transforming XML documents.**
  2. XPath is a language for **navigating in XML documents.** In other words we can reach to any node of XML document using XPath.
  3. XSL-FO is a language for formatting XML documents for **displaying it in desired manner.**

3.22.1 Transforming XML into XSLT

- Using XSLT we can decide the way by which we want the element to get displayed.
- XSLT uses **XPath** to find information in an XML document. XPath is used to navigate through elements and attributes in XML documents. We can sort these elements, hide or display particular element and can apply some decision making logic on these elements.
- XSLT processors **take two input documents** one is **XML document** and another is **XSLT document.** The XSLT document is nothing but a program and XML document is nothing but the input data, thus this program works on the XML input data. Then some or whole part of the XML document is selected, modified and merged with XSLT program document in order to produce another document.
- This newly produced document is provided as input to the XSLT processor which in turn produce another document called the **XSL document.**

- The XSL document is used along with the application so that particular application can be displayed on the web browser in some desired manner.

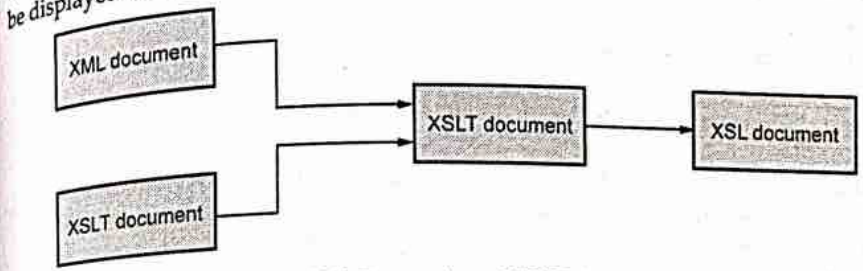


Fig. 3.22.1 Processing of XSLT document

- The XSLT document makes use of **templates** using which particular code can be described in XML document.
- XML document and then particular code in XSLT is executed when the match is found.
- XSLT processor processes the XML document sequentially by reading each line one by one.
- The XSLT model can be described as **template driven** or **data driven** model.

3.22.2 XSL Elements

1. The <xsl:template>

- The <xsl:template> is for building the templates.
- The **match** attribute is associated with the template element. The match attribute can also be used to define a template for the entire XML document. The **match="/"** defines the whole document.
- In the following example we store the XML elements in the tabular form using the <xsl:template> and <match>

Step 1 : Write a simple XML document. One sample example is as below –

Open Notepad and write the following code. Save it by the name **SimpleXml.xml**

XML Document[SimpleXml.xml]

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="SimpleXml.xsl"?>
<Student>
<Person-Details>
  <name>Anand</name>
  <address>Pune</address>
  <std>Second</std>
```

We have defined this file in step 2

```
    <marks>70 percent</marks>
</Person-Details>
    <Person-Details>
    <name>Anuradha</name>
    <address>Chennai</address>
    <std>Second</std>
    <marks>90 percent</marks>
    </Person-Details>
<Person-Details>
    <name>Archana</name>
    <address>Mumbai</address>
    <std>Forth</std>
    <marks>80 percent</marks>
</Person-Details>
<Person-Details>
    <name>Monika</name>
    <address>Delhi</address>
    <std>Tenth</std>
    <marks>77 percent</marks>
</Person-Details>
    </Student>
```

Step 2 : Now create a XSL stylesheet. Note that this document must have file name extension as **.xsl**. Open notepad and save following code in it.

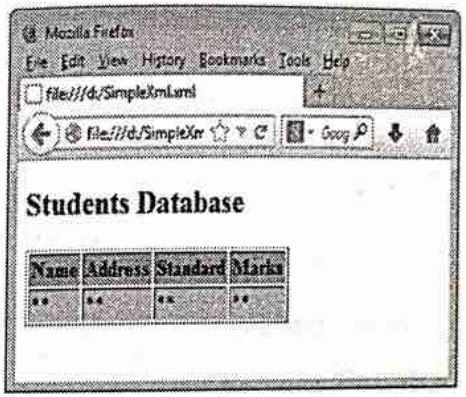
XSLT Document [SimpleXml.xsl]

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<trbgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>
<trbgcolor="pink">
<td>**</td>
<td>**</td>
<td>**</td>
```



```
<td>**</td>
</tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Step 3 : Open the web browser and the output will be as follows -



2. The <xsl:value-of>

The **value-of** is used to extract the value of xml element and add it to output stream of the XSL transformation. Following example illustrates this :

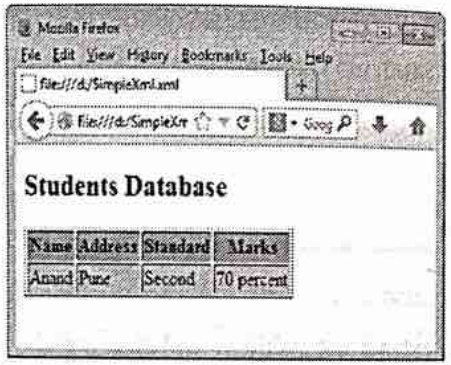
Step 1 : Keep the same xml**[SimpleXml.xml]** file which you have created in section 3.22.2 (1) Step 1.

Step 2 : Create the xsl file as follows -

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<trbgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>
```

```
<trbgcolor="pink">
<td><xsl:value-of select="Student/Person-Details/name"/></td>
<td><xsl:value-of select="Student/Person-Details/address"/></td>
<td><xsl:value-of select="Student/Person-Details/std"/></td>
<td><xsl:value-of select="Student/Person-Details/marks"/></td>
</tr>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Step 3 : Open the web browser and the output will be as follows -



3. The <xsl:for-each>

The **for-each** element allows to traverse through each element of the node

Step 1 : Keep the same xml**[SimpleXml.xml]** file which you have created in section 3.22.2(1) Step 1.

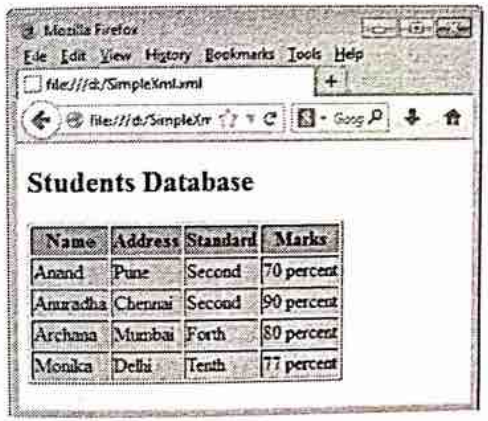
Step 2 : Now create .xsl document illustrating <xsl:for-each>. It is as follows -

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<trbgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
```



```
<th>Marks</th>
</tr>
<xsl:for-each select="Student/Person-Details" >
<trbgcolor="pink">
<td><xsl:value-of select="name"/></td>
<td><xsl:value-of select="address"/></td>
<td><xsl:value-of select="std"/></td>
<td><xsl:value-of select="marks"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Step 3 : Open the web browser and the output will be as follows -



4. The <xsl:sort>

We can also display the sorted data using **<xsl:sort>**.

Step 1 : Keep the same xml**[SimpleXml.xml]** file which you have created in section 3.22.2 (1) Step 1.

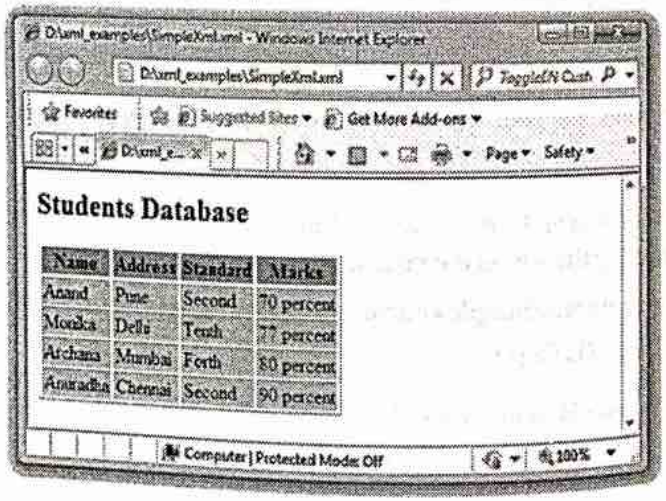
Step 2 : Create the .xsl file as follows -

SimpleXml.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
```

```
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<trbgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>
<xsl:for-each select="Student/Person-Details">
<xsl:sort select="marks"/>
<trbgcolor="pink">
<td><xsl:value-of select="name"/></td>
<td><xsl:value-of select="address"/></td>
<td><xsl:value-of select="std"/></td>
<td><xsl:value-of select="marks"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Output



5. The <xsl:if>

For selection specific record based on some condition the <xsl:if> is used.

Step 1 : Keep the same xml**[SimpleXml.xml]** file which you have created in section 3.22.2(1) Step 1.

Step 2 : Create the .xsl file as follows -

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<trbgcolor="gray">
<th>Name</th>
<th>Address</th>
<th>Standard</th>
<th>Marks</th>
</tr>
<xsl:for-each select="Student/Person-Details" >
<xsl:if test="marks>77">
<trbgcolor="pink">
<td><xsl:value-of select="name"/></td>
<td><xsl:value-of select="address"/></td>
<td><xsl:value-of select="std"/></td>
<td><xsl:value-of select="marks"/></td>
</tr>
</xsl:if>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

6. The <xsl:choose>

The <xsl:choose> element is used along with <xsl:when> and <xsl:otherwise> to express multiple conditional tests.

Step 1 : Keep the same xml**[SimpleXml.xml]** file which you have created in section 3.22.2 (1) Step 1.

Step 2 : Create the .xsl file as follows -

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
<h2>Students Database</h2>
<table border="1">
<trbgcolor="gray">
<th>Name</th>
<th>Marks</th>
</tr>
<xsl:for-each select="Student/Person-Details">
<tr>
<td><xsl:value-of select="name"/></td>
<xsl:choose>
<xsl:when test="marks &lt; 77">
<td bgcolor="pink">
<xsl:value-of select="marks"/></td>
</xsl:when>
<xsl:otherwise>
<td><xsl:value-of select="marks"/></td>
</xsl:otherwise>
</xsl:choose>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```


3.22.3 Difference between XML and XSLT

Sr.No.	XML	XSLT
1.	XML stands for extensible markup language.	XSLT stands for extensible stylesheet language transformations.
2.	XML is used to store data in structured format.	XSLT is used for transforming and formatting the XML file.
3.	XML does not perform transformations.	XSLT can transform one XML format into another XML format or into HTML or into plain text.
4.	XML contains the user defined tags.	XSLT contains special XSL tags and programming constructs.
5.	XML uses the user defined raw data for storage.	XSLT uses XPath for transformation and formatting of XML file.

Review Questions

1. *Discuss the XSLT technology with an example* **SPPU : May-19, Marks 5**
2. *Write at least any five differences between XML and XSLT.* **SPPU : March-20, Marks 5**

Part III : AJAX

3.23 Introduction

- AJAX is a Asynchronous **Java Script** and XML
- Here
  - **Asynchronous** means, the execution of script does not disturb the user's work.
  - **JavaScript** because, it makes use of Javascripting to do the actual work.
  - **XML** because along with JavaScript the XML is also supported to perform the given task in AJAX.
- It is not a new programming language but it is a kind of web document which adopts certain standards.
- AJAX allows the developer to exchange the data with the server and updates the part of web document without reloading the web page.

3.24 Working of AJAX

- When user makes a request, the browser creates a object for the **HttpRequest** and a request is made to the server over an internet.

- The server processes this request and sends the required data to the browser.
- At the browser side the returned data is processed using JavaScript and the web document gets updated accordingly by sending the appropriate response.

Following Fig. 3.24.1 illustrates this working.

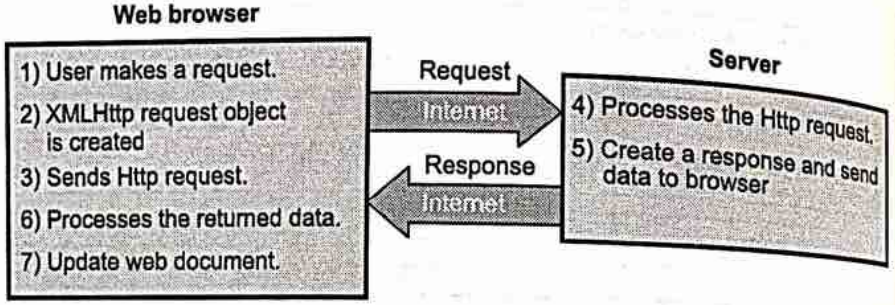


Fig. 3.24.1 Working of AJAX

Similarities and Differences

Sr.No.	Traditional Web Application Architecture	Ajax Based Web Application Architecture
1.	The web applications architectures are based on HTTP request response protocol.	
2.	At the client side the browser client has only one component and that is – user interface.	At the client side the browser client has user interface and AJAX Engine due to which the client gets quick response from the server.
3.	The architecture is based on client server communication.	
4.	It makes use of synchronous communication. The client has to wait to get the response from the server then only client can make the request to the server. It is start-stop-start-stop kind of communication.	By adding a new layer of AJAX Engine at the client side, it eliminates the start-stop-start-stop nature of communication and makes the communication asynchronous.
5.	With traditional web application architecture user cannot get rich user interface experience.	With AJAX architecture user gets rich user interface experience.
6.	It is less responsive.	It is more responsive.

7.	Building web application is simple.	The development time is more while designing the Ajax based web application.
8.	Limited use of JavaScript, CSS and XML technologies. In fact - Most user actions in the interface trigger an HTTP request back to a web server.	Extensive use of JavaScript, CSS and XML - i) standards-based presentation using XHTML and CSS; ii) Dynamic display and interaction using the Document Object Model(DOM); iii) Data interchange and manipulation using XML and XSLT; iv) Asynchronous data retrieval using XMLHttpRequest; v) JavaScript binding everything together

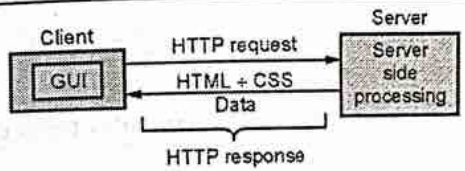


Fig. 3.24.2 Traditional web application architecture

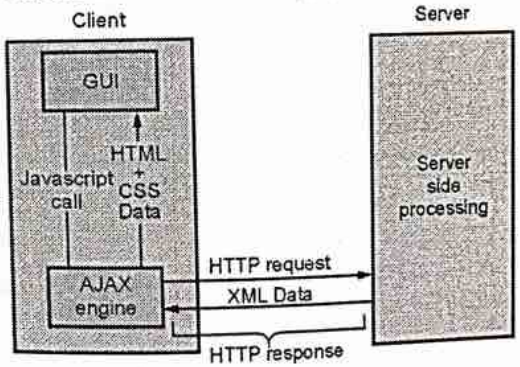


Fig. 3.24.3 AJAX web application architecture

Merits of AJAX

1. Response time is faster, hence increases performance and speed.
2. XMLHttpRequest object call as a asynchronous HTTP request to the Server for transferring data both side. It's used for making requests to the non-Ajax pages.
3. It is used in form validation.
4. It performs fetching of data from database and storing of data into database without reloading page.

Demerits of AJAX

1. It has browsing compatibility issues.
2. It is impossible to bookmark AJAX updated page contents.
3. Search engine would not crawl AJAX generated content. Hence, Search Engines like Google cannot index AJAX pages.

3.25 XMLHttpRequest Object

- XMLHttpRequest object is an important element in AJAX.
- XMLHttpRequest is an API which is used by JavaScript, VBScript and some other scripting languages.
- The methods of XMLHttpRequest are used in transferring data between a Web browser and a web server.
- It is because of XMLHttpRequest object, to update parts of web page without reloading the whole page.

Syntax

```
Variable = new XMLHttpRequest();
```

Various Methods of XMLHttpRequest object

Method	Purpose
new XMLHttpRequest()	Creates new XMLHttpRequest object.
Open(method, URL, async,user,pwd)	To make request this method is used. method: The request can be GET or POST URL: The location of file async: true(for asynchronous) and false(synchronous) user: user name(optional) pwd:password(optional)
send()	sends the request to the server.
send(string)	sends the request to the server.
setRequestHeader()	Adds the label- value pair to the header.

Various Properties of XMLHttpRequest object

Property	Description
readyState	Specifies the ready state of XMLHttpRequest 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
onreadystatechange	When readystate property changes the onreadystatechange event listener will be automatically invoked
responseText	Returns the response data as a string
status	Returns the status number. Following is the meaning of various numbers returned by the status property 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status text as "OK", or "Forbidden"

Example 3.25.1 *Create an XMLHttpRequest to retrieve data from an XML file and display the data in an HTML table. The data to be retrieved is a collection of stationary items stored in an XML file.*

Solution : Step 1 : Create an xml file storing the stationary items. The code is as follows -

```
/ELEMENTS>
<STATIONARY>
<ITEM>Notebook</ITEM>
<PRICE>40</PRICE>
</STATIONARY>
<STATIONARY>
<ITEM>Punching Machine</ITEM>
<PRICE>60</PRICE>
</STATIONARY>
<STATIONARY>
<ITEM>Pencil Box</ITEM>
<PRICE>40</PRICE>
</STATIONARY>
```

```
<STATIONARY>
<ITEM>Eraser</ITEM>
<PRICE>5</PRICE>
</STATIONARY>
</ELEMENTS>
```

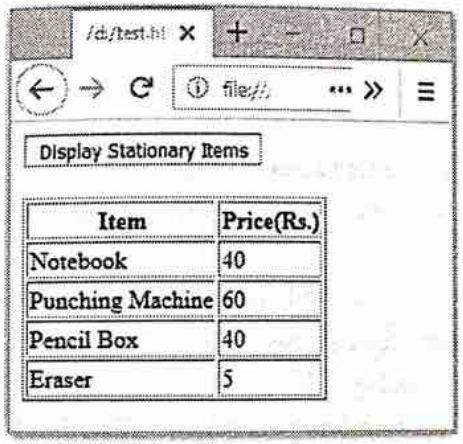
Step 2 : The XML file displaying the stationary items in a tabular form is as given below.

```
test.html
<!DOCTYPE html>
<html>
<body>
<button type="button" onclick="loadXMLDoc()">Display Stationary Items</button>
<br><br>
<table id="demo" border="1"></table>
<script>
function loadXMLDoc()
{
 if(window.XMLHttpRequest)
 {
   xmlhttp = new XMLHttpRequest();
 }
    else
{
   xmlhttp = new ActiveXObject("Microsoft.req");//to execute on Inter. Explorer
 }
 xmlhttp.onreadystatechange = function()
 {
    if (this.readyState == 4 && this.status == 200) {
       myFunction(this);
 }
 };
       xmlhttp.open("GET","stationary.xml",true);
   xmlhttp.send();
}
function myFunction(xml)
{
 var i;
    var xmlDoc = xml.responseXML;
    var table="<tr><th>Item</th><th>Price(Rs.)</th></tr>";
    var x = xmlDoc.getElementsByTagName("STATIONARY");
    for (i = 0; i <x.length; i++)
    {
   table += "<tr><td>" +
   x[i].getElementsByTagName("ITEM")[0].childNodes[0].nodeValue +
```



```
   "</td><td>" +
   x[i].getElementsByTagName("PRICE")[0].childNodes[0].nodeValue +
   "</td></tr>";
 }
 document.getElementById("demo").innerHTML = table;
}
</script>
</body>
</html>
```

Output



3.26 Coding AJAX Script

SPPU : Dec.-18, 19, May-18, 19, Marks 8

Let understand how AJAX works with the help of programming example

```
<html>
 <head>
  <script type="text/javascript">
   function MyFun()
   {
        if (window.XMLHttpRequest)
    {
      req=new XMLHttpRequest();
    }
    else
     {
       req=new ActiveXObject("Microsoft.req");
     }
    req.onreadystatechange=function()
     {
       if (req.readyState==4 && req.status==200)
       {
```

2

3

```
            document.getElementById("myID").innerHTML=req.responseText;
        }
    }
    req.open("GET","newdata.txt",true);        (4)
    req.send();  (5)
  }
</script>
</head>
<body>
<div id="myID">This text can be changed</div>

<button type="button" onclick="MyFun()">Change</button>        (1)
</body>
</html>
```

Script Explanation (Above numbered steps are explained here)

In above script, we have written some text which can be replaced by some another text on button click.

1. On button click a function **MyFun** is invoked. Thus client triggers the event.
2. **XMLHttpRequest** object is used to exchange data with a server. This object allows the user to change/update the parts of the web page without reloading it fully. The modern web browsers such as IE7+, FireFox,Chrome have built in **XMLHttpRequest** but old web browsers make use of **ActiveXObject**.
3. When a request to a server is sent, then **onreadystatechange** event is triggered.

The **readyState** property holds the status of the XMLHttpRequest. The **readystate=4** means request is finished and response is ready. The **status = 200** means "OK"

The DOM is modified and response is modified using the statement

```
document.getElementById("myID").innerHTML=req.responseText;
```

4. The request can be sent to the server by using two functions **open()** and **send()**.

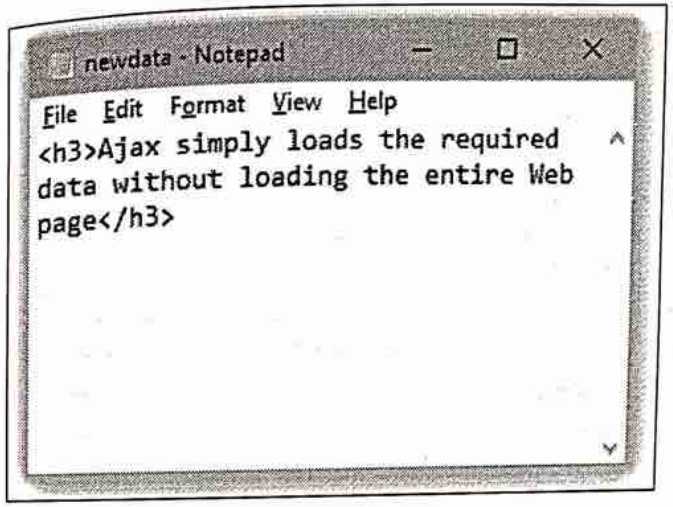
```
req.open("GET","newdata.txt",true);
```

- GET or POST method
- Location of file on server
- true: means asynchronous; false : means asynchronous

Asynchronous communication allows fast processing of the data.

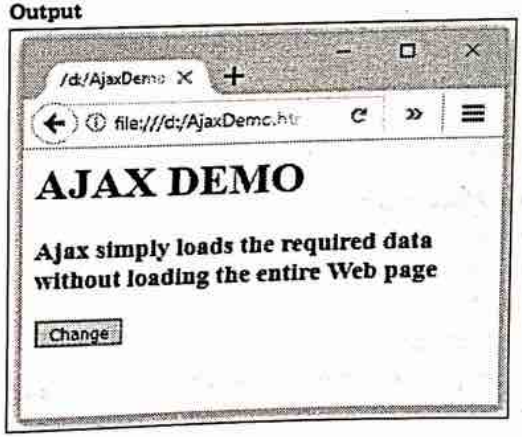
The **newdata.txt** file contains some updating text using which our web page can be updated.

newdata.txt



5. The **send()** method sends the request to the server.

Output



Example 3.26.1 *Write AJAX script to obtain the student information stored in XML document. The information should be displayed on clicking the button. It should be displayed in tabular form.*

Solution :

Step 1 : Create an XML file for storing the student information. The XML file is as follows

Student.xml

```
<Student>
   <student_data>
      <Name>AAA</Name>
      <Marks>45</Marks>
   </student_data>
   <student_data>
      <Name>BBB</Name>
      <Marks>55</Marks>
   </student_data>
   <student_data>
      <Name>CCC</Name>
      <Marks>67</Marks>
   </student_data>
   <student_data>
      <Name>DDD</Name>
      <Marks>84</Marks>
   </student_data>
</Student>
```

Step 2 : Create a AJAX script as follows -

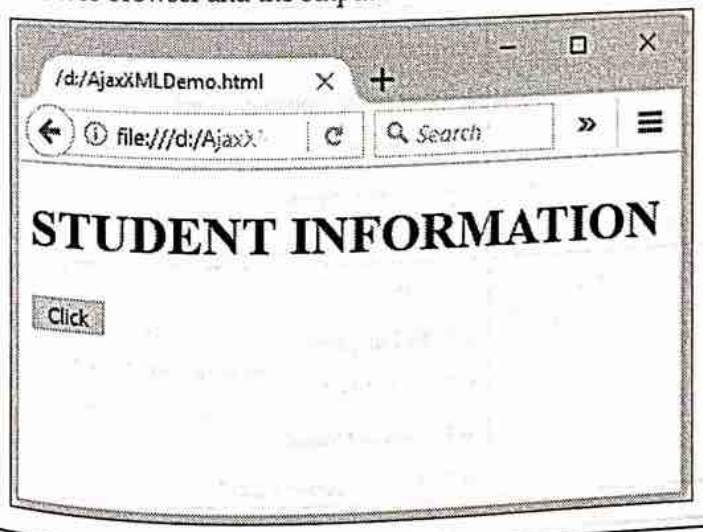
AjaxXMLDemo.html

```
<!DOCTYPE html>
<html>
<body>
<h1>STUDENT INFORMATION</h1>
<button type="button" onclick="MyFun()">Click</button>
<br><br>
<table border="1" id="demo"></table>
<script>
function MyFun()
{
   var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function()
   {
      if (this.readyState == 4 && this.status == 200)
      {
```

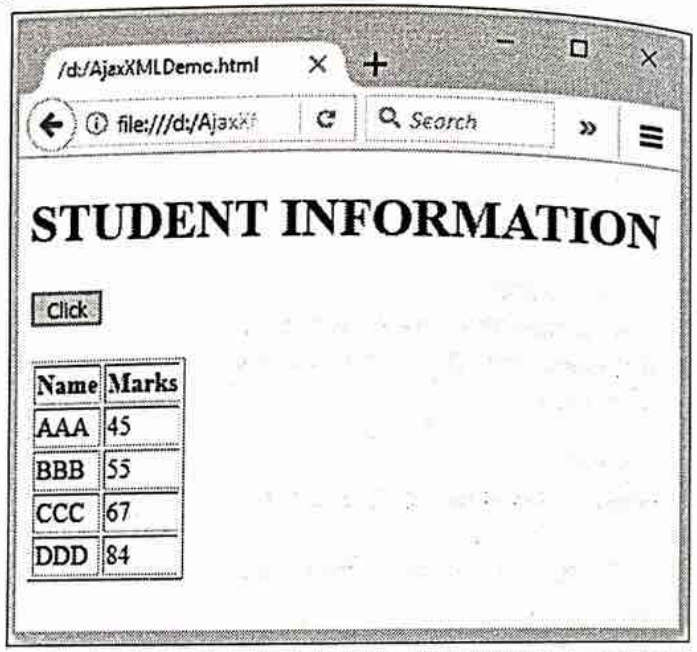


```
         Load_XML_File(this);
      }
   };
   xhttp.open("GET", "Student.xml", true);
   xhttp.send();
}
function Load_XML_File(xml)
{
   var i;
   var xmlDoc = xml.responseXML;
   var table="<tr><th>Name</th><th>Marks</th></tr>";
   var x = xmlDoc.getElementsByTagName("student_data");
   for (i = 0; i <x.length; i++)
   {
      table += "<tr><td>" +
      x[i].getElementsByTagName("Name")[0].childNodes[0].nodeValue +
      "</td><td>" +
      x[i].getElementsByTagName("Marks")[0].childNodes[0].nodeValue +
      "</td></tr>";
   }
   document.getElementById("demo").innerHTML = table;
}
</script>
</body>
</html>
```

Step 3 : Open the web browser and the output will be as follows -



On clicking the button we will get



Example 3.26.2 *List and explain different values of ready state and status property of the HTTPRequestObject.*

SPPU : Dec.-18, Marks 6

Solution :

readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 400:"Bad Request" 403: "Forbidden" 404: "Page not found" 500: "Internal Server Error"

Review Questions

1. *Draw and explain how AJAX works with the help of suitable example.* **SPPU : May-18, Marks 6**
2. *What is use of XMLHttpRequest object ? Explain its use with the help of simple JavaScript document.* **SPPU : May-18, Marks 6**
3. *Write different applications without AJAX and with AJAX.* **SPPU : May-18, Marks 4**
4. *Explain with simple example the XMLHttp request object. List and explain different values of readystate and status property of the HTTP request object.* **SPPU : May-19, Marks 5**
5. *What is the purpose of AJAX ? Explain with relevant example.* **SPPU : Dec.-19, Marks 8**

3.27 Multiple Choice Questions

Q.1 The Java ____ specification defines an application programming interface for communication between the web server and the application program.

a servlet b server
c program d none of these

Q.2 Servlet runs each request in _____.

a JVM b JRE
c servlet container d none of these

Q.3 What is bytecode ?

a Machine - specific code b Java code
c Machine - independent code d None of these

Q.4 What are the functions of Servlet container ?

a Lifecycle management b Communication support
c Multithreading support d All of the above

Q.5 Which of the following package contains servlet classes ?

a javax.servlet b javax.servlet.http
c both a and b d none of these

Q.6 Which method is used to specify before any lines that uses the PrintWriter ?

a setPageType() b setContentType()
c setResponseType() d setContextType()

Q.7 What is the lifecycle of a servlet ?

a Servlet class is loaded b Servlet instance is created
c init, Service, destroy method is invoked
d All of the above

Q.8 What type of servlets use these methods doGet(), doPost(),doHead, doDelete(), doTrace() ?

a Genereic Servlets
b HttpServlets
c All of the above
d None of the above

Q.9 Which of the following code is used to get cookies in servlet ?

a response.getCookies()
b request.getCookies()
c Cookies.getCookies()
d None of the above.

Q.10 If the client has disabled cookie in the browser which session management mechanism could the web container employ ?

a Session management using URL rewriting
b Session management using Cookies
c Either Cookies or URL rewriting
d Cookies and URL rewriting must be used together

Q.11 Servlet is initialized by calling the following method :

a service()
b destroy()
c init()
d none

Q.12 Common gateway interface is used to :

a [illegible]xecutable files from web content by web server

Q.14 Correct version of the declaration which defines the XML version _____ ?

a <?xml version ="1.0" />
b <?xml version="1.0"?>
c <xml version="1.0">
d none of these

Q.17 XML is _____ ?

a platform independent
b language independent
c both a and b
d none of these

Q.18 Which language is case sensitive ?

a HTML
b XML
c Both a and b
d None of these

Q.19 DTD stands for _____ ?

a Document Trial Data
b Document Trial Definition
c Data Type Definition
d Document Type Definition

Q.20 XML DOM is a standard for performing the following operation on XML elements _____ ?

a change
b add
c delete
d all the above

Q.21 DOM parser is used on _____ ?

a client side
b server side
c both a and b
d none

Q.22 Which are techniques for defining the structure of a specific type of XML documents ?

a Schema
b DTD
c Both a and b
d none of the above

Q.25 Parameter entities can appear in _____.

a xml file
b dtd file
c xsl file
d both 1 and 2

Q.26 Which characters are illegal in XML elements ?

a >
b &
c Both a and b
d None of these

Q.27 The syntax for parameter entity is _____.

a <! ENTITY % NAME DEFINITION>
b < ENTITY % NAME DEFINITION>
c <! ENTITY $ NAME DEFINITION>
d < ENTITY $ NAME DEFINITION>

Q.28 The XML DOM object is _____.

a entity
b entity reference
c comment reference
d comment data

Q.29 Which of the following is not a XML storage option ?

a Native storage as XML data type
b Mapping between XML and relational storage
c Small object storage
d None of the mentioned

Q.30 Which of the following part of the XML data stored in an XML column is very important for locking ?

a Granularity
b Degree of structure
c Hierarchy
d None

Answer Keys for Multiple Choice Questions :

Q.1	a	Q.2	c	Q.3	c	Q.4	d	Q.5	c
Q.6	b	Q.7	d	Q.8	b	Q.9	b	Q.10	a
Q.11	c	Q.12	a	Q.13	a	Q.14	c	Q.15	a
Q.16	b	Q.17	c	Q.18	b	Q.19	c	Q.20	d
Q.21	c	Q.22	c	Q.23	b	Q.24	a	Q.25	b
Q.26	c	Q.27	a	Q.28	b	Q.29	c	Q.30	a

□□□

UNIT - IV

4 JSP and Web Services

Syllabus

JSP : Introduction to Java Server Pages, JSP and Servlets, running JSP applications, Basic JSP, JavaBeans classes and JSP, Support for the Model-View-Controller paradigm, JSP related technologies. ***Web Services :*** *Web Service concepts, Writing a Java Web Service, Writing a Java web service client, Describing Web Services : WSDL, Communicating Object data: SOAP.* ***Struts :*** *Overview, architecture, configuration, actions, interceptors, result types, validations, localization, exception handling, annotations.*

Contents

4.1 Introduction to Java Server Pages
4.2 JSP and Servlets ..... April-18, ... Marks 5
4.3 Basics of JSP ..... April-18, May-18, 19, ... Dec.-18, 19, ... Marks 5
4.4 Implicit Objects ..... May-18, 19, Dec.-18, ... Marks 5
4.5 JavaBeans Classes and JSP ..... May-18, ... Marks 5
4.6 Support for the Model-View-Controller Paradigm ..... Dec.-18, ... Marks 6
4.7 Web Services Concepts ..... May-18,19, Dec.-18, ... Marks 8
4.8 Writing a Java Web Service
4.9 Describing Web Services : WSDL
4.10 Communicating Object Data : SOAP
4.11 Overview ..... Dec.-18, ... Marks 4
4.12 Architecture ..... May-18, 19 ..... Marks 6
4.13 Execution of First Struts Applications
4.14 Configuration ..... Dec.-18,19, ... Marks 6
4.15 Actions ..... Dec.-18, May-19, ... Marks 6

4.16 Interceptors .....May-18, ...Marks 6
4.17 Result Types
4.18 Validations
4.19 Localization
4.20 Exception Handling
4.21 Annotations
4.22 Multiple Choice Questions

Part I : JSP

4.1 Introduction to Java Server Pages

- JSP stands for Java Server Pages.
- It is an alternative way than the servlet to build the dynamic web pages. It is built on top of the servlet.

4.2 JSP and Servlets

SPPU : April-18, Marks 5

- Following are some advantages of Java Server pages -
  1. JSP is useful for **server side programming.**
  2. JSP can be used **along with servlets.** Hence **business logic** for any application can be developed using JSP.
  3. **Dynamic contents** can be handled using JSP because JSP allows scripting and element based programming.
  4. JSP allows creating and using our own **custom tag libraries.** Hence any application specific requirements can be satisfied using custom tag libraries. This helps the developer to develop any kind of application.
  5. JSP is a **specification and not a product.** Hence developers can develop **variety of applications** and add up to performance and quality of software products. Due to this many companies are ready to invest in JSP technology.
  6. JSP is an essential component of J2EE. Hence using JSP it is possible to develop simple as well as complex applications.

Difference between Servlet and JSP

Sr. No.	JSP	Servlet
1.	JSP is a scripting language that generates dynamic content.	Servlets are the Java programs that can be compiled to generate dynamic content.
2.	JSP runs slower than servlet.	Servlet runs faster than JSP.
3.	In Model View Controller JSP acts as view.	In Model View Controller servlet acts as controller.
4.	JSP can build custom tags.	There is no facility of creating custom tags.
5.	JSP can directly call Java beans.	Servlets have no facility of calling Java bean.
6.	It is easier to code in JSP.	The servlets are basically complex Java programs.

Review Question

1. *Write advantages of JSP over servlets. Also explain the life cycle of JSP.*

SPPU : April-18, Marks 5

4.3 Basics of JSP

SPPU : April-18, May-18, 19, Dec.-18, 19, Marks 5

4.3.1 JSP Document Structure

Every JSP document contains four elements -

1. **Template text :** The XML of HTML code can be directly used in JSP document. This is called the static or fixed part of the document. This markup is called template text. This part is not modified by JSP container.
2. **Action elements :** The action elements are used to create the documents dynamically. The action elements is written in the form
   i) Opening tag ii) Action body iii) Closing tag.

For example

<jsp:include>	This tag works as a subroutine. It includes the response from servlet or JSP when request is being processed.
<jsp:forward>	This tag helps in forwarding the current request to servlet or to JSP page.
<jsp:element>	This tag generated the XML elements dynamically.
<jsp:text>	This tag is used to handle template text. When JSP pages are written as XML documents then this tag is used.

3. **Directives :** JSP directives control the processing of entire JSP Page. It gives direction to the server regarding the processing of a page. The directive elements are enclosed within the <%@ .... %> delimiters. For example -

```
<%@page import="java.io.*" %>
<%@page language="java" %>
<%@page contentType="text/html" %>
```

4. **Scriptlets :** Scriptlets are nothing but java code enclosed within <% and %> tags. For example we can write following statement in JSP document

```
<p>
    <% out.print("JSP is eqaul to HTML and JAVA "); %>  //Scriptlet
</p>
```


Example of JSP document

```
<%@ page language="java" contentType="text/html" %>
<html>
    <head>
      <title>Demo for Template Text</title>
    </head>
    <body bgcolor="gray">
      <h1>Twinkle Twinkle little stars</h1>
      <h2>How I wonder what you are!!!</h2>
      <li>like a diamond in the sky</li>
      <p>
        <% out.print("JSP is eqaul to HTML and JAVA "); %>
      </p>
    </body>
</html>
```

Directive

Template text

Scriptlets

4.3.2 JSP Processing

- JSP pages can be processed using **JSP container** only. Following are the steps that need to be followed while processing the request for JSP page -

1. Client makes a request for required JSP page to the server. The server must have JSP container so that JSP request can be processed. For instance : Let the client makes a request for *xyz.jsp* page.
2. On receiving this request the JSP container searches and then reads the desired JSP page. Then this JSP page is straightaway converted to corresponding **servlet.** Basically any JSP page is a combination of template text and JSP element. Every **template text** is translated into corresponding **println** statement.

For instance :

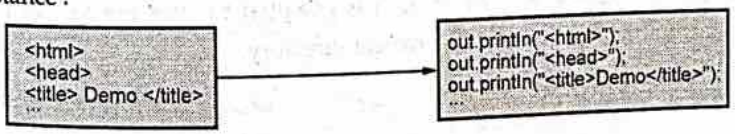


Fig. 4.3.1

Every **JSP element** is converted into corresponding **Java code.** This phase is called **translation phase.** The output of translation phase is a servlet. For instance : our *xyz.jsp* gets converted to *xyzServlet.java.*

3. This servlet is then compiled to generate the servlet class file. Using this class the response can be generated. This phase is called **request processing phase.**
4. The JSP container thus **executes** the servlet class file.
5. A requested page is then returned to the client as a **response.** Refer Fig. 4.3.2.

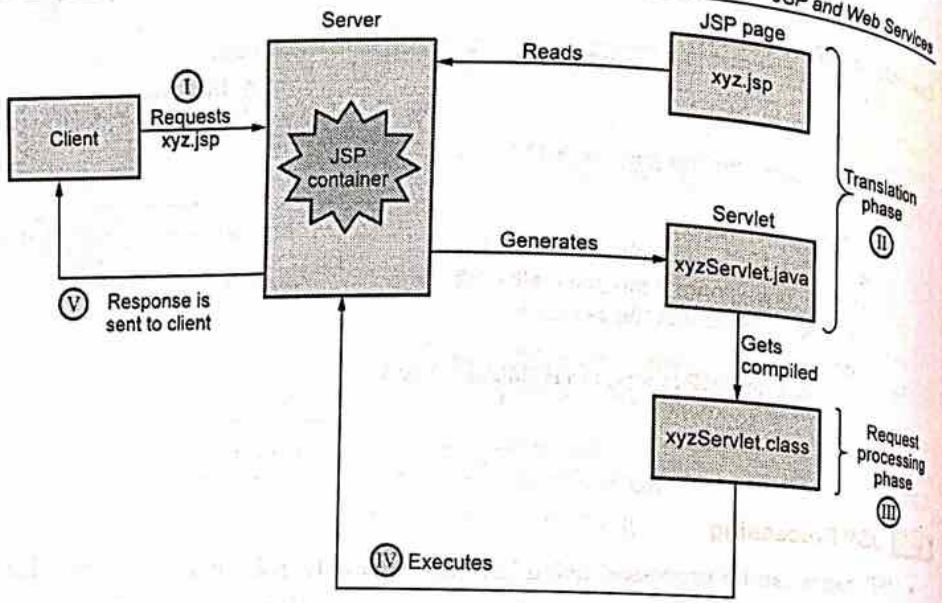


Fig. 4.3.2 JSP processing

4.3.3 Running JSP Applications

For execution of JSP pages following installations are needed.

1. **Java Development Kit :** Download and install the latest version of JDK.
2. **Tomcat WebServer :** The Tomcat is a web container provided by Apache Web server. Now a days XAMPP software package is installed for running JSP, Servlets and PHP programs.

Note that before running any JSP page it is essential to start the tomcat by double clicking the startup.bat file present in the tomcat directory.

Example 4.3.1 *Write a simple JSP page for displaying the message "This is my first JSP page!!!"*

SPPU : May-18, Marks 5

Solution : We follow these steps to get the JSP page displayed -

Step 1 : First of all open some text editor like Notepad and type the following code.

hello.jsp

```
<%@ page language="java" contentType="text/html" %>
<%@ page import ="java.util.*" %>
<html>
<!-- This is basic JSP page -->
<title>JSP Demo </title>
```

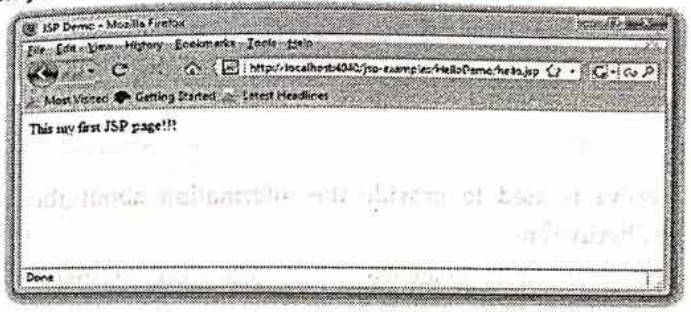


```
<body>
<%-- Displaying the message on the browser --%>
    <% out.println("This my first JSP page!!!"); %>
  </body>
</html>
```

Step 2 : Copy this file to webapps directory of your tomcat folder.

Step 3 : Start the Tomcat Web server.

Step 4 : Open some suitable web browser like Firefox or Internet Explorer. Type the path for the JSP document and the output will be as follows -



Script explanation

The first line is the **page directive** statement

```
<%@ page language="java" contentType="text/html" %>
```

using this line we are specifying that the language used in this script is Java and the contentType specifies the MIME type which is text/html.

Then on the next line we have imported the Java library package **java.util** using import statement -

```
<%@ page import ="java.util.*" %>
```

We can specify the JSP comments in two ways

```
<!—HTML style comment statement -->
<%-- JSP comment statement --%>
```

As JSP is commonly used along with HTML, it also supports the HTML comments.

For displaying the message on the web browser we have written the message using the Java statement

```
out.println("This my first JSP page!!!");
```

Normally any Java code must be written within <% and %>.

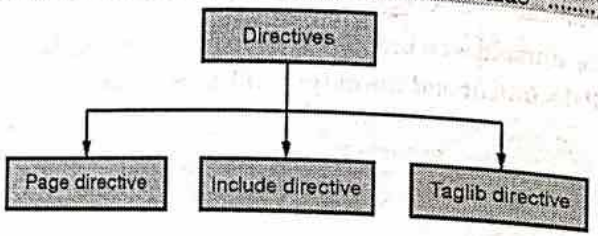
Then all the corresponding tags are closed.

4.3.4 Directives

JSP directives control the processing of entire JSP Page. It gives direction to the server regarding the processing of a page.

Syntax

```
<%@ directive name [attribute name="value" attribute name="value" .....]%>
```



1. Page directive :

The page directive is used to provide the information about the page. Various attributes of page directive are -

i) import	ii) language	iii) ContentType
iv) extends	v) session	vi) buffer
vii) autoFlush	viii) isThreadSafe	ix) info
x) errorPage	xi) IsErrorPage	

For example -

```
<%@page import="java.io.*" %>
<%@page language="java" %>
<%@page contentType="text/html" %>
```

2. Include directive :

Include directive is used to copy the content of one JSP page to another. It's like including the code of one file into another.

For example

```
<% @include file=calculate.jsp %>
```

3. Taglib directive :

This directive basically allows user to use custom tags in JSP. The custom tags are those which are created by user.

For example -

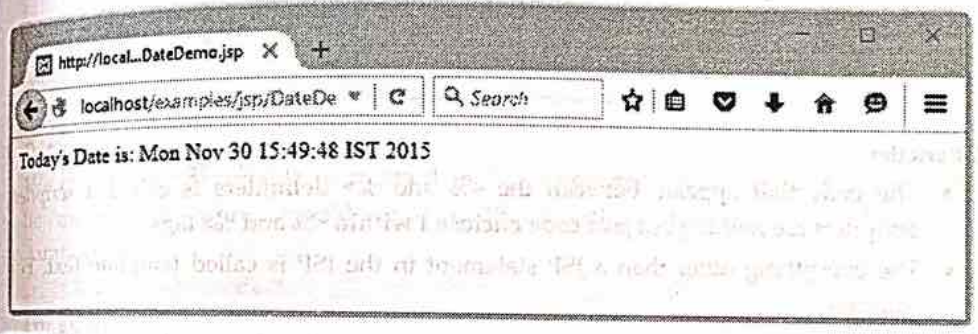
```
<%@ taglib uri="mytaglib" prefix="sampletag" %>
```


Example 4.3.2 *Write a JSP program to display current date and time.*

Solution : DateDemo.jsp

```
<%@ page language="java" contentType="text/html" %>
<%@ page import ="java.util.*" %>
<html>
<body>Today's Date is:
    <%= new Date().toString() %>
</body>
</html>
```

Output



4.3.5 Comments

We can specify the JSP comments in two ways

```
<!—HTML style comment statement -->
<%-- JSP comment statement --%>
```

As JSP is commonly used along with HTML, it also supports the HTML comments.

4.3.6 Scripting Elements

There three types of scripting elements in JSP.

1. Expression 2. Scriptlet 3.Declaration

1. Expression

The expression tag is used to represent the expression in JSP page. The syntax of writing expression is -

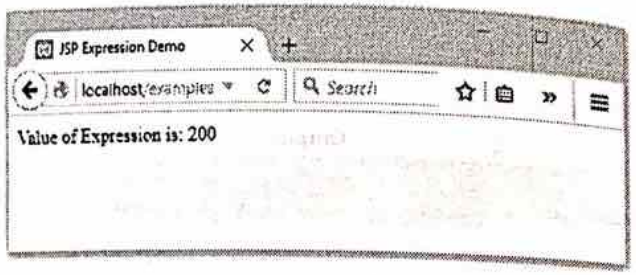
```
<%= Java Expression %>
```

For example -

```
<html>
  <head>
    <title>JSP Expression Demo</title>
  </head>
```

```
<body>
  Value of Expression is :
  <%= (10*20) %>
</body>
</html>
```

Output



2. Scriptlet

- The code that appears between the <% and %> delimiters is called *a scriptlet*. Scriptlets are nothing but java code **enclosed within** <% and %> tags.
- The everything other than a JSP statement in the JSP is called template text. For example.

Templatetext.jsp

```
<%@ page language="java" contentType="text/html" %>
<html>
    <head>
      <title>Demo for Tempplate Text</title>
    </head>
    <body bgcolor="gray">
      <h1>Twinkle Twinkle little stars</h1>
      <h2>How I wonder what you are!!!</h2>
      <li>like a diamond in the sky</li>
      <p>
        <% out.print("JSP is eqaul to HTML and JAVA "); %>
      </p>
  </body>
</html>
```

In the above code the boldfaced code is a template text. The output for the above code will be

Output

3. Declaration

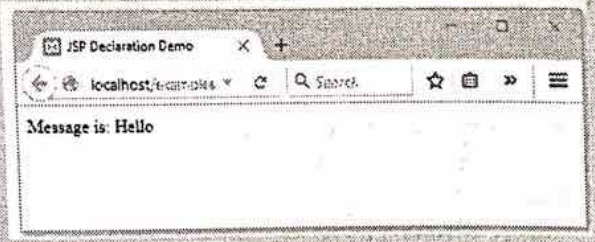
We can declare a variable or method in JSP inside declaration tag. We can declare static member, instance variable and methods inside declaration tag. Syntax of declaration tag -

```
<%! Declaration code %>
```

For example -

```
<html>
  <head>
    <title>JSP Declaration Demo</title>
  </head>
  <%!
    String msg = "Hello";
  %>
  <body>
    Message is:
    <% out.println(msg); %>
  </body>
</html>
```

Output



The above JSP code contains the declaration within <%! %> tag.

Example 4.3.3 *Develop a simple JSP application that accepts user's age and displays a message according to the following rules :*

i) Age<15 : Message : You are a kid!

ii) Age between 16 and 40 : You are young!

iii) Age above 40 : You are old!

Solution : For developing this application we will create an HTML form for inputting the age value and then invoke the JSP page on which the appropriate messages are given based on the age.

Step 1 :

InputAge.html

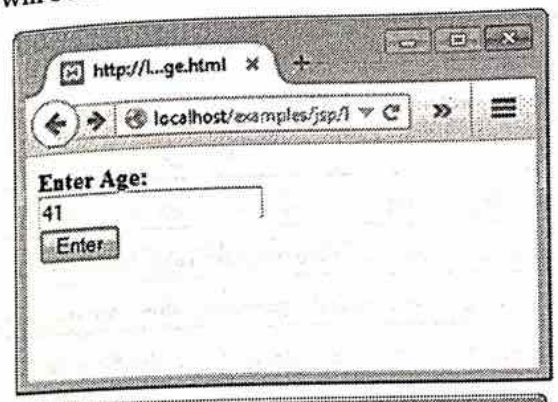
```
<html>
<head>
</head>
<body>
<form method="get" action="AgeMessage.jsp">
<strong>Enter Age:</strong>
<br/>
<input type="text" name="Age">
<br/>
<input type="submit" value="Enter">
</form>
</body>
</html>
```

Step 2 : The JSP code is as follows -

AgeMessage.jsp

```
<html>
<body>
<%@ page language="java" contentType="text/html" %>
<%@page import="java.io.*" %>
<%
    String msg="";
    String age;
    int age_val;
    age=request.getParameter("Age");
    age_val=Integer.parseInt(age);
    if(age_val<15)
        msg+="You are a kid!";
    if((age_val>16)&&(age_val<40))
        msg+="You are young!";
    if(age_val>40)
        msg+="You are old!";
    out.println("<h3>"+msg+"</h3>");
%>
</body>
</html>
```


Step 3 : The output will be as follows -



Example 4.3.4 *Write a JSP scriptlet for displaying even numbers between 1 to 50.*

SPPU : April-18, In Sem, Marks 5

Solution :

```
<html>
<body>
<%@ page language="java" contentType="text/html"%>
<%@ page import="java.io.*"%>
<%
  for(int i=1;i<=50;i++)
  {
    if(i%2==0)
      out.println("<p>"+i+"</p>");
  }
%>
</body>
</html>
```

4.3.7 Actions and Templates

- The standard actions are those actions that can be defined by the JSP specification itself.
- Following is a list of standard actions.

Action Element	Description
<jsp:useBean>	This tag is used to instantiate the object of java bean.
<jsp:setProperty>	This tag is used to set the property value for the java bean.
<jsp:getProperty>	This tag is used to get the property value from the java bean.
<jsp:include>	This tag works as a subroutine. It includes the response from servlet or JSP when request is being processed.
<jsp:param>	For adding the specific parameter to the request this tag is used. Normally this tag is used inside <jsp:include> or <jsp:forward>
<jsp:forward>	This tag helps in forwarding the current request to servlet or to JSP page.
<jsp:plugin>	This tag is used to generate HTML code and to embed the applet into it.
<jsp:attribute>	This tag sets the value of the action attribute
<jsp:element>	This tag generated the XML elements dynamically.
<jsp:text>	This tag is used to handle template text. When JSP pages are written as XML documents then this tag is used.
<jsp:body>	This tag is used to set the body element.

Let us now discuss the standard actions by creating appropriate JSP pages.

Include action

Using the **<jsp:include>** action we can include different files in current jsp pages.

For example -

Include_action.jsp

```
<html>
  <head>
    <title> Include action Demo </title>
  </head>
  <body>
    <h1>This page contains include action</h1>
    <jsp:include page="test_include.html" />
    <jsp:include page="test_jsp.jsp" />
  </body>
</html>
```


Now the static html page can be

test_include.html

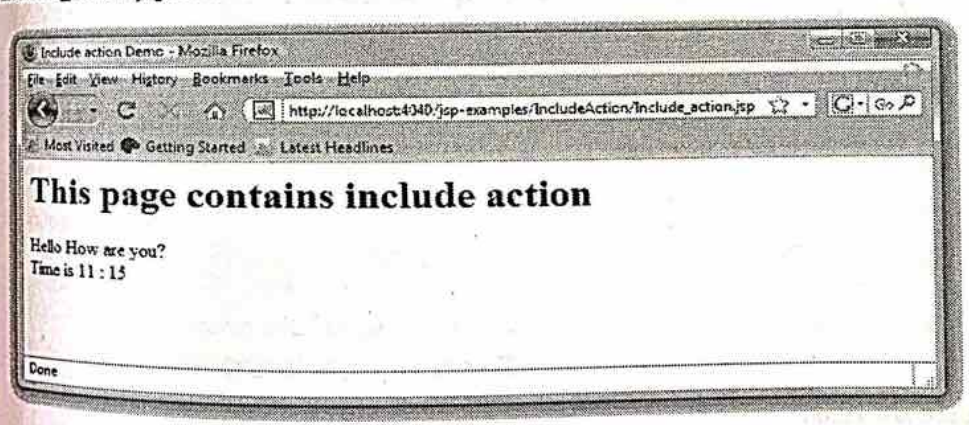
```
<html>
<body>
  Hello How are you?
</body>
</html>
```

Now the JSP page for displaying dynamic content is

test_jsp.jsp

```
<html>
  <body>
  <br>Time is <%= new java.util.Date().getHours()%>
  : <%= new java.util.Date().getMinutes()%>
  </body>
</html>
```

For getting the output, open the web browser and open our main JSP page **Include_action.jsp**, it is as follows-



Forward action

- Using forward action we can forward the current request to corresponding JSP page or servlet.
- Here is a simple demonstration in which we have created two web pages the first one is a main JSP page named as first.jsp in which we are using JSP action forward. And forwarding the current request to an HTML page which is named as Second.html.

first.jsp

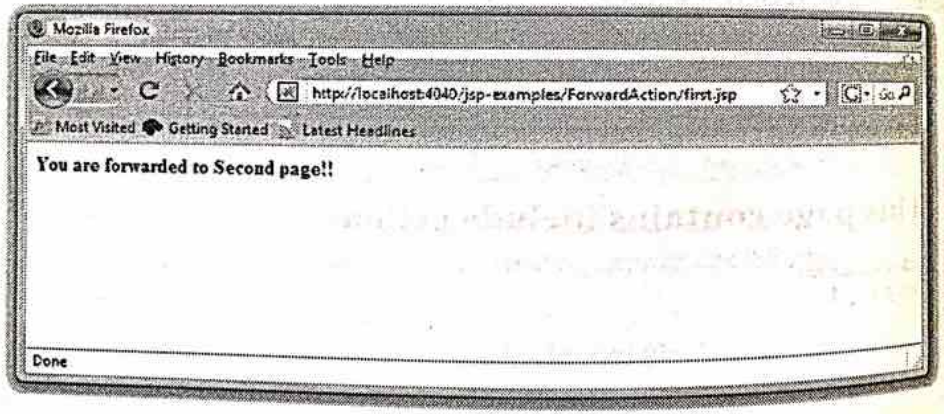
```
<html>
   <head>
      <title> Forward action Demo </title>
   </head>
   <body>
      <h1>This page contains forward action</h1>
      <jsp:forward page="Second.html" />
   </body>
</html>
```

Second.html

```
<html>
<body>
   <strong>You are forwarded to Second page!!</strong>
</body>
</html>
```

For getting the output open the web page and open the main JSP page that is first.jsp and you will get the following output -

Output



useBean action

- Java beans are reusable components. We can use simple Java bean in the JSP.
- Beans are used in the JSP pages as the instance of class. We must specify the scope of the bean in the JSP page. Here scope of the bean means the range time span of the bean for its existence in JSP.
- Bean is basically a Java class with get and set methods.
- Let us understand the useBean action with illustrative example.

Example 4.3.5 *Write a JSP document to display number of times user is visiting particular page.*

Solution :

Step 1 : Create a simple bean in which a variable count is maintained. This count can be incremented using the JSP page.

CounterDemo.java

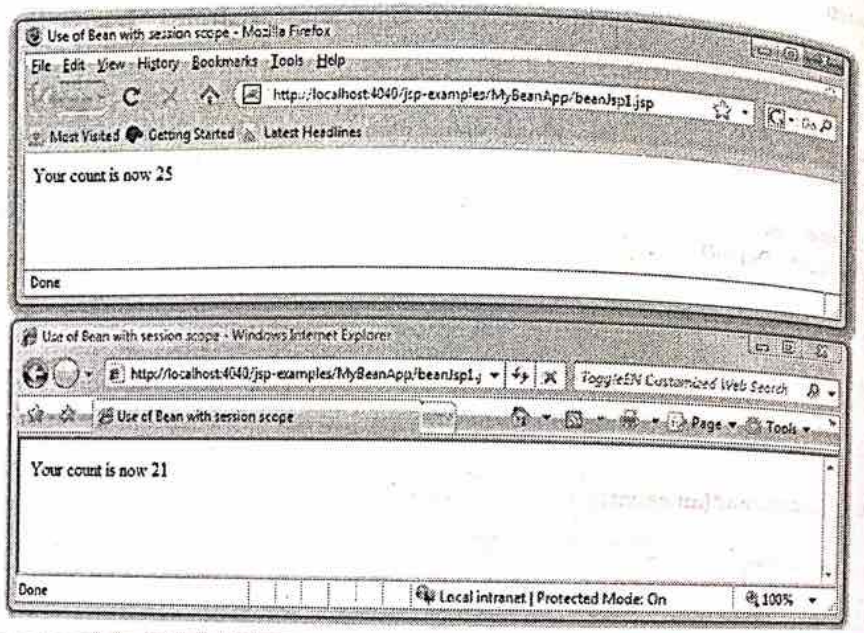
```
public class CounterDemo
{
public int count;
public CounterDemo()
{
count=0;
}
public int getCount()
{
return count;
}
public void setCount(int count)
{
this.count=count;
}
}
```

Step 2 : Now create a JSP page which make use of the bean created in step 1 in the session scope.

beanJsp1.jsp

```
<html>
<head>
<title> Use of Bean with session scope </title>
</head>
<body>
<jsp:useBean id="bean_id" class="CounterDemo" scope="session" />
<%
bean_id.setCount(bean_id.getCount()+1);
%>
Your count is now <%= bean_id.getCount() %>
</body>
</html>
```

Step 3 : Open two different web browsers Mozilla Firefox and Internet Explorer and open the same JSP page in both the browsers, repeatedly refresh the JSP page in both the browsers and observe it.



Example 4.3.6 *Write the differences between include action and include directive in JSP.*

SPPU : Dec.-18, May-19, End Sem, Marks 5

Solution : The difference is as follows -

1) Include directive includes the file at translation time - that is when the JSP gets converted into the equivalent servlet whereas the include action includes the file at runtime.
2) If the included file is changed but not the JSP which is including it then the changes will reflect only when we use include action tag. The changes will not reflect if you are using include directive as the JSP is not changed so it will not be translated for request processing and hence the changes will not reflect.
3) Syntax difference : Include directive : <%@ include file="file_name" %> whereas include action has like this <jsp:include page="file_name" />
4) When using include action tag we can also pass the parameters to the included page by using param action tag but in case of include directive it's not possible.

Review Questions

1. *What are the usage of JSP directives and JSP actions.* **SPPU : April-18, Marks 5**
2. *Write JSP program to demonstrate use of page directive, scriptlet and expression tag.* **SPPU : Dec.-19, Marks 5**

4.4 Implicit Objects

SPPU : May-18, 19, Dec.-18, Marks 5

- The implicit objects are predefined variables used to access request and application data. These objects are used by the scripting elements. Following is a list of implicit objects.

Variable name	Class/Interface name	Meaning	Sample methods
application	javax.servlet.ServletContext	This object provides the resources shared within a web application.	log() getServerinfo()
config	javax.servlet.ServletConfig	It helps in passing the information to servlet or JSP page during initialization.	getInitParameter() getServletName()
request	javax.servlet.http.HttpServletRequest	It provides the methods for accessing the information made by current request.	getContentLength() getLocalAddr() getServerName()
response	javax.servlet.http.HttpServletResponse	It provides the methods related to adding cookies, sessions or setting header.	addCookie() addHeader() flushBuffer() getContentType() setContentType()

out	javax.servlet.jsp.JspWriter	It provides the methods related to I/O	clear() newLine()
page	java.lang.Object	This variable is assigned to instance of JSP implementation class. It is rarely used variable.	
pageContext	javax.servlet.jsp.PageContext	It provides access to several JSP page attributes	getPage() getRequest() getResponse() getSession()

Review Questions

1. *Create a simple JSP pages which will demonstrates use of implicit session objects in JSP.* **SPPU : May-18, Dec.-18, Marks 5**
2. *List and elaborate any five JSP implicit objects.* **SPPU : May-19, Marks 5**

4.5 JavaBeans Classes and JSP

SPPU : May-18, Marks 5

- Java beans are reusable components. We can use simple Java bean in the JSP.
- Beans are used in the JSP pages as the instance of class. We must specify the scope of the bean in the JSP page. Here scope of the bean means the range time span of the bean for its existence in JSP.
- Bean is basically Java class with **getXX** and **setXX** methods.

Characteristics for JavaBean Class

1) Java Bean class contains a default **no argument constructor.** This constructor must have the access specifier as **public.**
2) It should be serializable and should **implement serializable interface.**
3) It can have any number of properties which can be read or written. For these properties we write **getter and setter methods.**

JavaBean Properties

- A JavaBean Property is a named attribute that can be accessed using the object.
- The JavaBean properties can be accessed through two methods namely

1) **getPropertyName() :** This method is called **accessor** method. It returns the property value. This method is written in some special way. For example - If the property is **rollNo** then the method name for **rollNo** is **getRollNo()**
2) **setPropertyName() :** This is called **mutator** method. This method is generally used to assign the value to the properties. This method is written in some special way. For example - If the property is **rollNo** then method name will be **setRollNo()**

The read-only attribute is for **getPropertyName()** method and write-only attribute is for **setPropertyName()** method.

Example : Bean program using JSP for Employee Information.

Step 1 : Create a Bean program in some package. I have created a folder(as a package) named **BeanProg**. The code for JavaBean class is written as follows -

Employee.java

```
package BeanProg;
public class Employee implements java.io.Serializable {
  private String empName=null;
  private int empNo=0;
  public Employee() { //no argument constructor
  }
  public String getEmpName() {
    return empName;
}

  public int getEmpNo() {
    return empNo;
  }
  public void setEmpName(String empName) {
    this.empName=empName;
  }
  public void setEmpNo(int empNo) {
    this.empNo=empNo;
  }
}
```

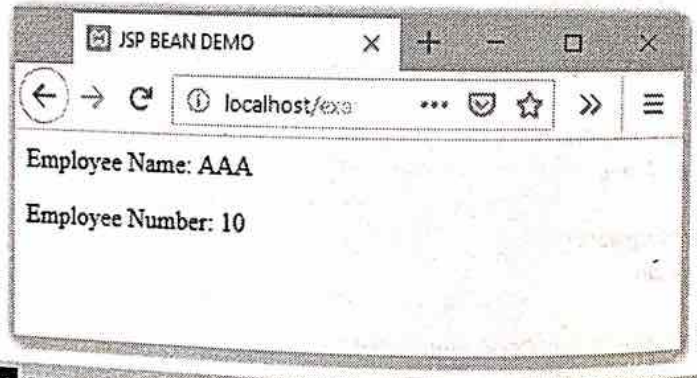
Step 2 : The JSP code can be written as follows -

EmpBeanDemo.jsp

```
<html>
  <head>
    <title> JSP BEAN DEMO </title>
  </head>
  <body>
    <jsp:useBean id="empID" class="BeanProg.Employee">
      <jsp:setProperty name = "empID" property = "empName" value = "AAA"/>
        <jsp:setProperty name = "empID" property = "empNo" value = '10'/>
    </jsp:useBean>
    <p>Employee Name:
    <jsp:getProperty name = "empID" property = "empName"/>
    </p>
    <p>Employee Number:
    <jsp:getProperty name = "empID" property = "empNo"/>
    </p>
  </body>
</html>
```

Step 3 : Start tomcat and open the JSP page in Web browser you will get the output as follows -

Output



Example 4.5.1 *Write a client server JSP program to find the simple interest and display the result in the client.*

Solution : This program is created using following steps.

Step 1 : Create an simple HTML form for inputting the values of P, N and R. The HTML document for this is as given below.

Input.html

```
<html>
<head><title>Input Form</title></head>
<body>
Enter following values:<br/>
<form method="post" action="interest.jsp">
Amount:<input type="text" name="amount" value="" size="10"/>
Period:<input type="text" name="period" value="" size="3"/>
Rate of Interest:<input type="text" name="rate" value="" size="3"/>
<input type="submit" value="Submit"/>
</form>
</body>
</html>
```

Step 2 : Create JSP page which acts as a client to receive the values for P,N and R. These values will be sent to the server Java program for calculating the simple interest. The calculated interest value is displayed by this same JSP page. The code for this is as follows -

Interest.jsp

```
<html>
<head>
  <title> Simple Interest Calculation Demo </title>
</head>
<body>
<jsp:useBean id="bean_id" class="interestcalcDemo.InterestDemo" scope="session" />
<% String s_p=request.getParameter("amount");
int p=Integer.parseInt(s_p);
   bean_id.setP(p);

String s_n=request.getParameter("period");
int n=Integer.parseInt(s_n);
bean_id.setN(n);

String s_r=request.getParameter("rate");
int r=Integer.parseInt(s_r);
   bean_id.setR(r);
%>
</form>
Amount = <%= p%><br/>
Period = <%= n%><br/>
Rate = <%= r%><br/>
<strong>Interest is now <%= bean_id.getI() %><br/></strong>
</body>
</html>
```

Calling server Java program for computing the simple interest.

Collecting values from **input.html** created in **Step 1** and sending them to sever program using method **set**

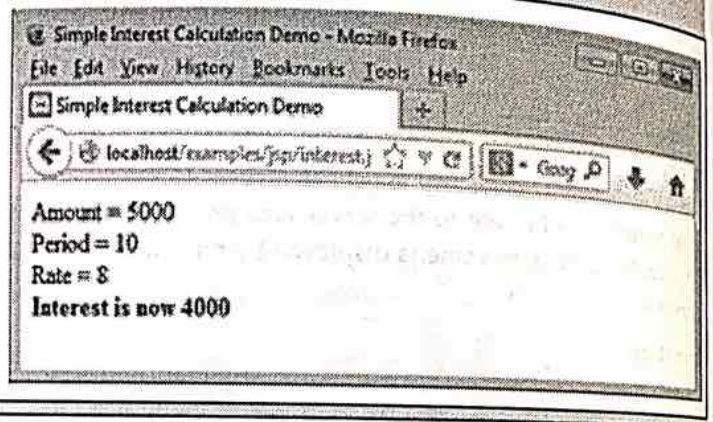
Obtaining interest value from server

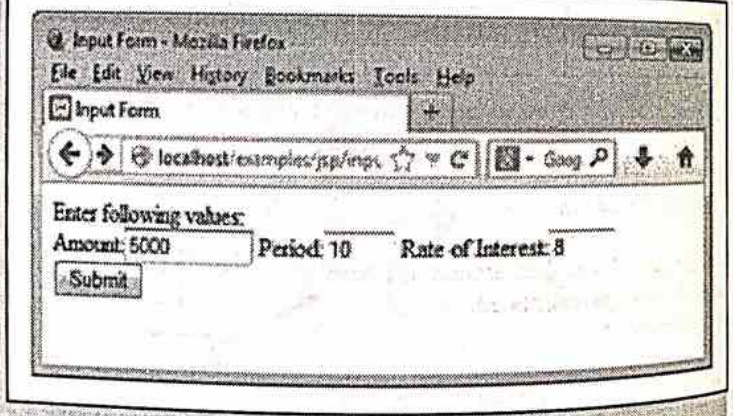
Step 3 : Following is a Java bean program which acts as a Server. It calculated the simple interest using the formula (p*n*r)/100. This value is then returned to the client JSP program created in Step 2. The code for server java program is as follows -

InterestDemo.java

```
package interestcalcDemo;
public class InterestDemo
{
public int p,n,r;
public InterestDemo()
{
    p=0;n=0;r=0;
}
public int getI()
{
    int i=(p*n*r)/100;
    return i;
}
public void setP(int p)
{
    this.p=p;
}
public int getP()
{
    return p;
}
public void setN(int n)
{
    this.n=n;
}
public int getN()
{
    return n;
}
public void setR(int r)
{
    this.r=r;
}
public int getR()
{
    return r;
}
}
```

Output





Review Question

1. *What is the JSP action tags ? Discuss in details jsp:useBean action tag.* **SPPU : May-18, Marks 5**

4.6 Support for the Model-View-Controller Paradigm

SPPU : Dec.-18, Marks 6

- The design model of JSP application is called MVC model. The MVC stands for Model-View-Controller. The basic idea in MVC design model is to separate out design logic into three parts - modelling, viewing and controlling.
- Any server application is classified in three parts such as business logic, presentation and request processing.
- The **business logic** means the coding logic applied for manipulation of application data. The **presentation** refers to the code written for look and feel of the web page. For example: background color, font style, font size, placing of controls such as text boxes, command buttons and so on. The request processing is nothing but a combination of business logic and presentation. The request processing is always done in order to generate the response.

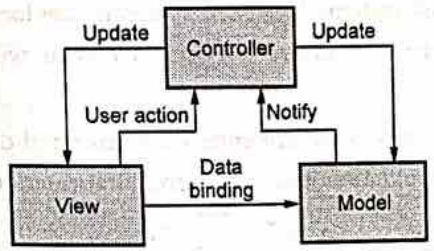


Fig. 4.6.1 Model View Controller (MVC)

- According to the MVC design model, the model corresponds to business logic, view corresponds to presentation and controller corresponds to request processing.

Advantages of using MVC design model

- The use of MVC architecture allows the developer to keep the separation between business logic, presentation and request processing.
- Due to this separation it becomes easy to make **changes** in presentation without disturbing the business logic. The changes in presentation are often required for accommodating the new presentation interfaces.

Review Question

1. *What is MVC ? Draw and explain MVC architecture for developing web applications.* **SPPU : Dec.-18, Marks 6**

Part II : Web Services

4.7 Web Services Concepts

SPPU : May-18,19, Dec.-18, Marks 8

Definition : The Web Services are the software systems that are displayed by the web browser using the web protocol. These software systems are used by the some software applications rather than by end-users directly.

- Web service is a software system designed which is **independent** of specific hardware or software on which it is running.
- Examples of a web service are -
  1. **Credit card validation system -** For using this service the client simply enters the ID and password for the credit card and web service reports the validity of it.
  2. **Whether forecast system -** The clients submits the location and the web service returns the information about current whether as well as it may report some predictions.
  3. **Currency converter -** The client enter the source and destination currency along with the amount and appropriate converted information will be displyed.

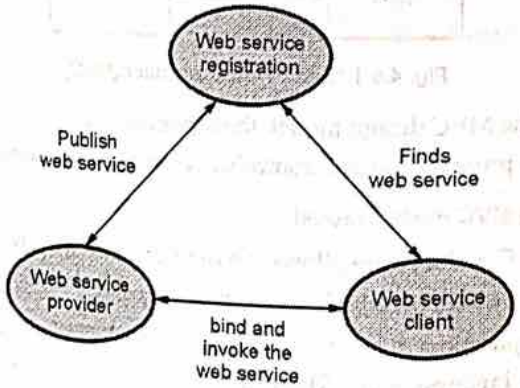


Fig. 4.7.1 Basic web service model

Following are the steps that a web service model follows -

Step 1 : In the first step, a service provider **publishes** a web service in the **web service registry.**

Step 2 : A web client who demands for some web service **searches** in the registry. After finding a match for the desired web service in the registry. After finding a match for the desired web service in the registry, the client chooses it.

Step 3 : The client **binds** to the corresponding web service provider and **invokes** for the service.

4.7.1 Components of Web Services

The role of various components such as XML, SOAP, WSDL, UDDI in web service building is as shown below -

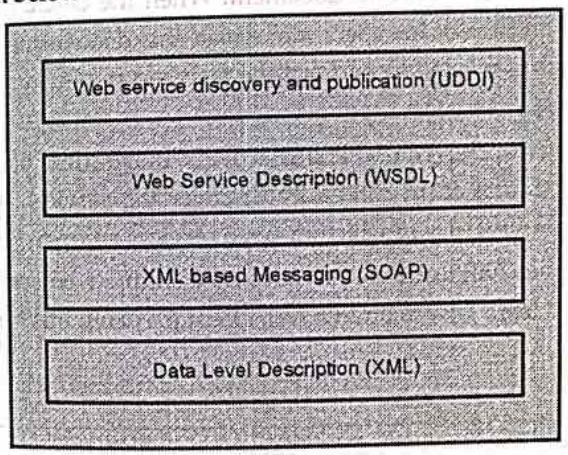


Fig. 4.7.2 Web service elements

- At the lowest level of web services there are simply XML files that are for data level description.
- SOAP - Simple Object Access Protocol is a simple XML based protocol which allows exchange of messaging over HTTP.
- Typically web services make use of this protocol for exchange of information .
  - The SOAP makes use of JAX-RPC component. These are JAVA API for XML that makes use of Remote Procedure Calls.
  - Basically SOAP defines message structure, encoding rules and conventions for exchange of information in RPC programming model.
  - The following table describes core JAX-RPC interfaces and classes.

Interface / Class	Description
Service	Main client interface. Used for both static and dynamic invocations.
Service Factory	This is a factory class which is used to create services
Stub	It is used by client for invoking the web services.
Call	For invoking the web services the call is used.
JAXRPCException	This exception is thrown when an error occurs while invoking web service.

- At the higher level very important element of web service lies i.e. WSDL. The Web Service Description Language is used for abstract description of web services. This description is provided in XML document. When the client wants to make use of web service, the users of the client make read this description file and understand what input data is needed to perform the web service. Such a special file is called WSDL i.e. Web Services Description Language.
- UDDI i.e. Universal Description and Discovery Integration, provides a directory of web services, so that client can easily discover the services of their choices. The UDDI maintains a registry in which the web services are registered.
- In this chapter we will learn how to develop a simple web service using JAX-RPC. Further we will learn how to build a **web client** for using this service.

Review Questions

1. *Identify and justify benefits of using web services ?* **SPPU : May-18, Marks 6**
2. *What are web services ? List and discuss components of web services.* **SPPU : Dec.-18, Marks 6**
3. *What are the web services ? List and explain layers in protocol stack of web service architecture.* **SPPU : May-19, Marks 8**

4.8 Writing a Java Web Service

In this section, we will create two Web Projects – one for Server and other for Client. The Server defines the web service and client invokes this web service

We can develop a web service using Apache Axis2 and Eclipse.

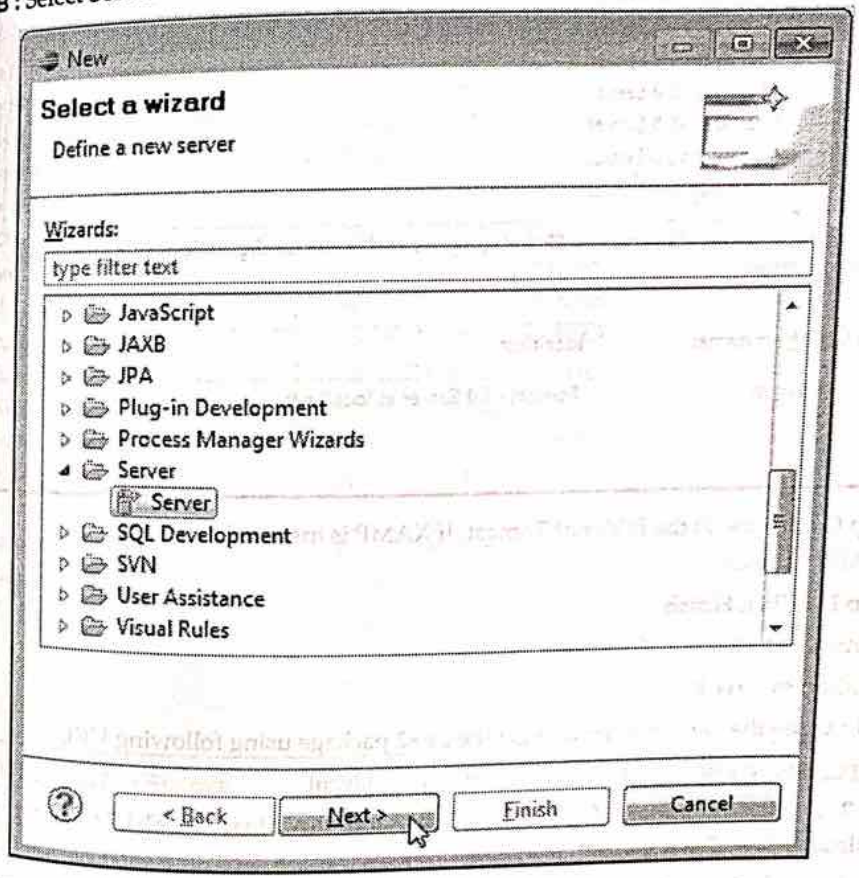
Prerequisite :

1) JDK must be installed.
2) Eclipse must be installed
3) Apache Server, Tomcat must be installed. Install the software package XAMPI for this purpose.

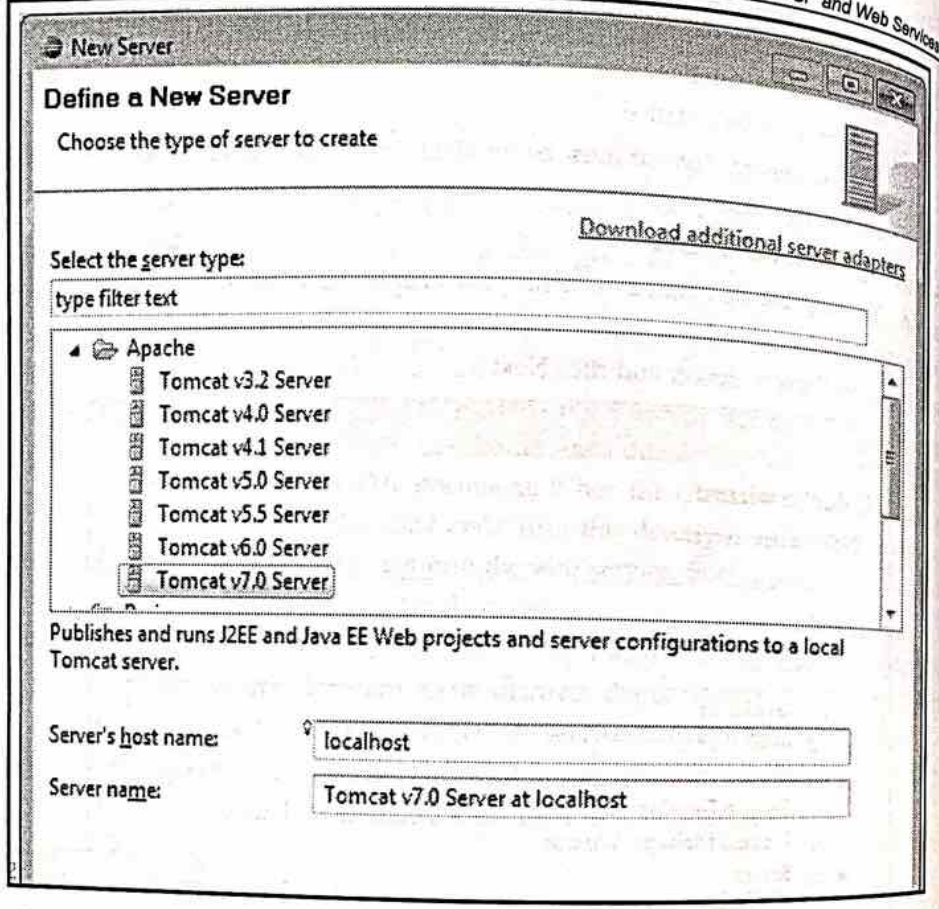
Configuring Tomcat to Eclipse

Step A : To integrate Tomcat Server with your Eclipse, Select **File->New->Other...** from Eclipse main menu.

Step B : Select Server->Server and click Next



Step (3) Select Tomcat Version XX and click **Next.**



Step C : Browse to the folder of Tomcat. If XAMP is installed then it is c:\XAMPP\tomcat

Step D : Click **Finish**

4) Install Axis2.

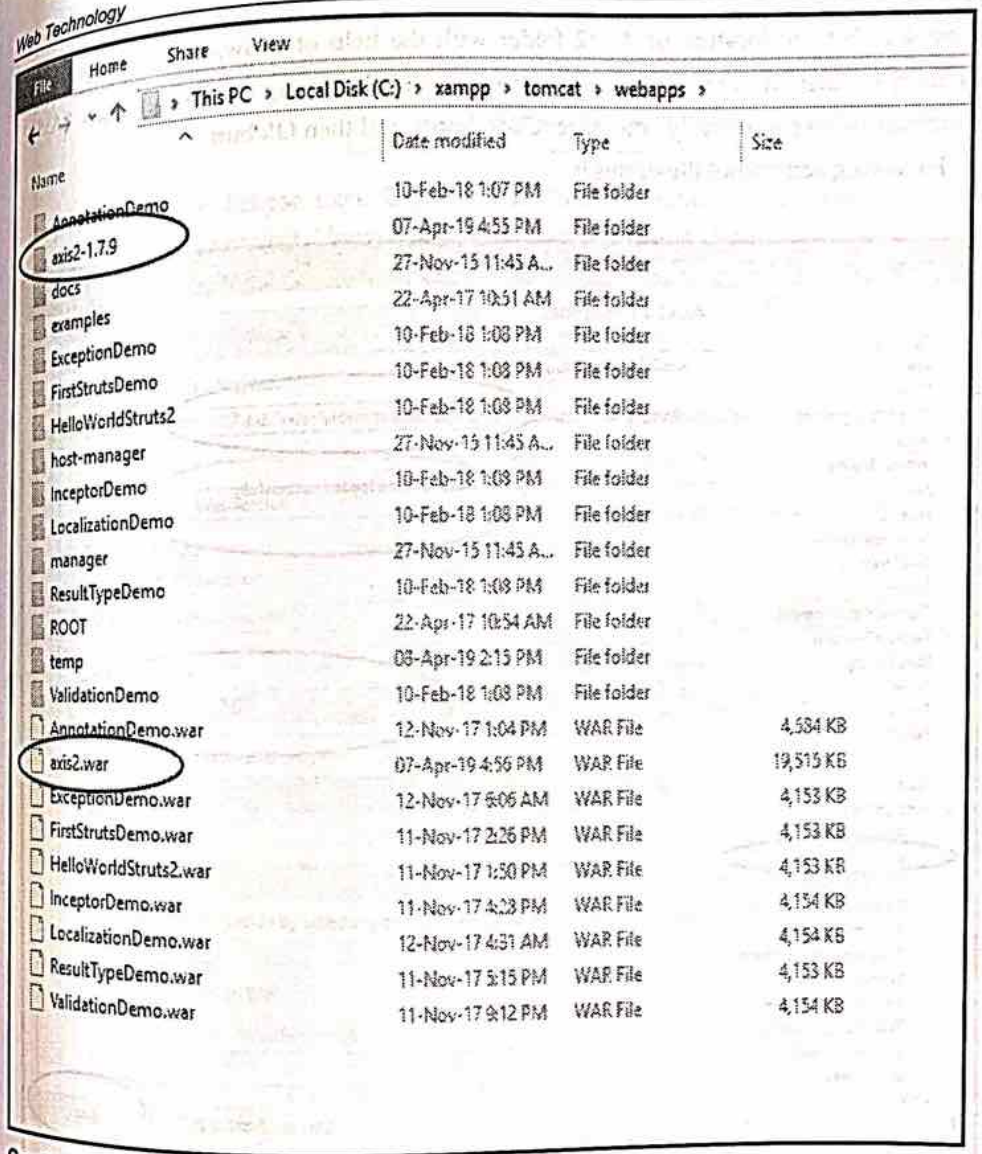
Installation of Axis 2 :

Step 1 : Open the web browser and get the axis2 package using following URL

http://axis.apache.org/axis2/java/core/download.html

Step 2 : Click on Binary Distribution Link and download axis2-1.7.9-bin.zip. Also download axis2-1.7.9-war.zip

Step 3 : Unzip both the folders and copy these folders at your c:\tompcat-directory\webapps folder. Following screenshot helps in understanding the location



Configuring Axis2 in Eclipse

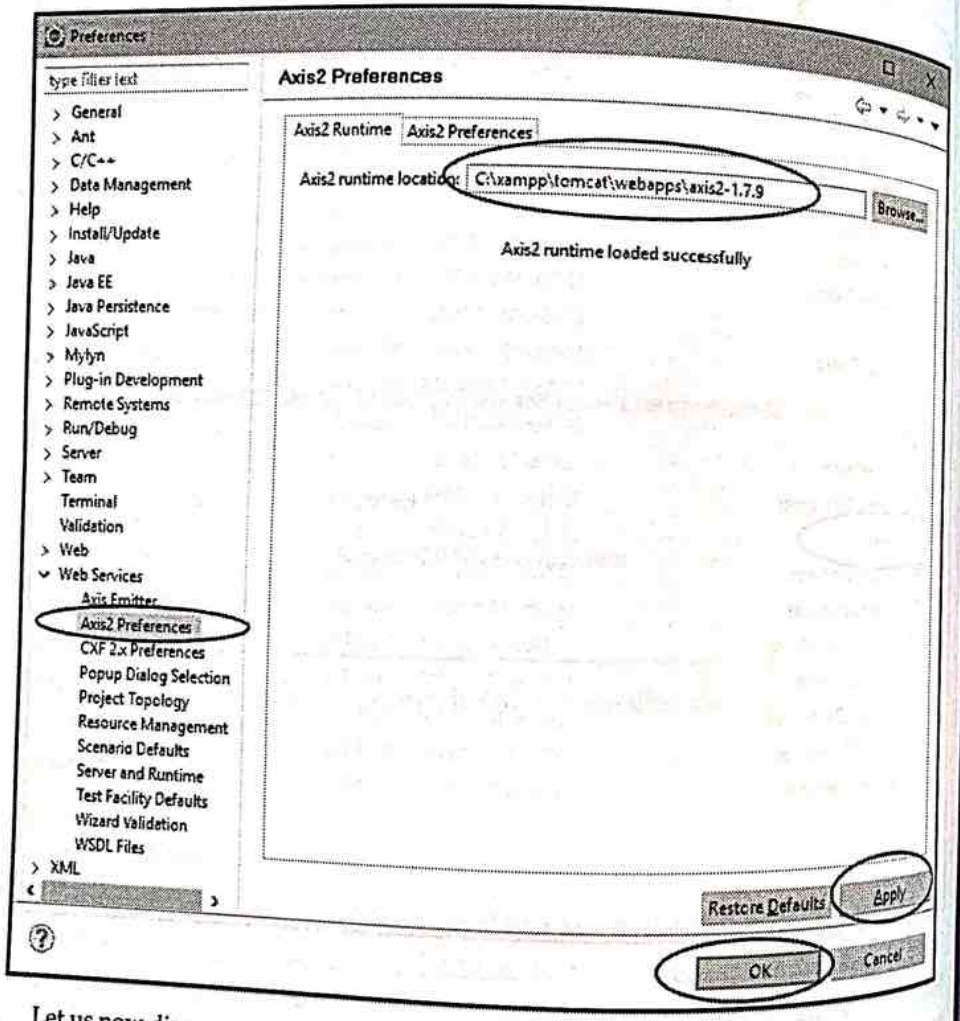
Step 1 : Open the Eclipse.

Step 2 : Click on **Window -> Preferences.**

Step 3 : Locate **Web Services. Click Axis2 Preferences.**

Step 4 : Set the location of Axis2 folder with the help of **Browse..** button. It is your Webapps folder in Tomcat directory. On successful deploying of Axis2, we get "Axis2 runtime loaded successfully" message. Click **Apply** and then **Ok** button.

Following screenshot illustrates it.



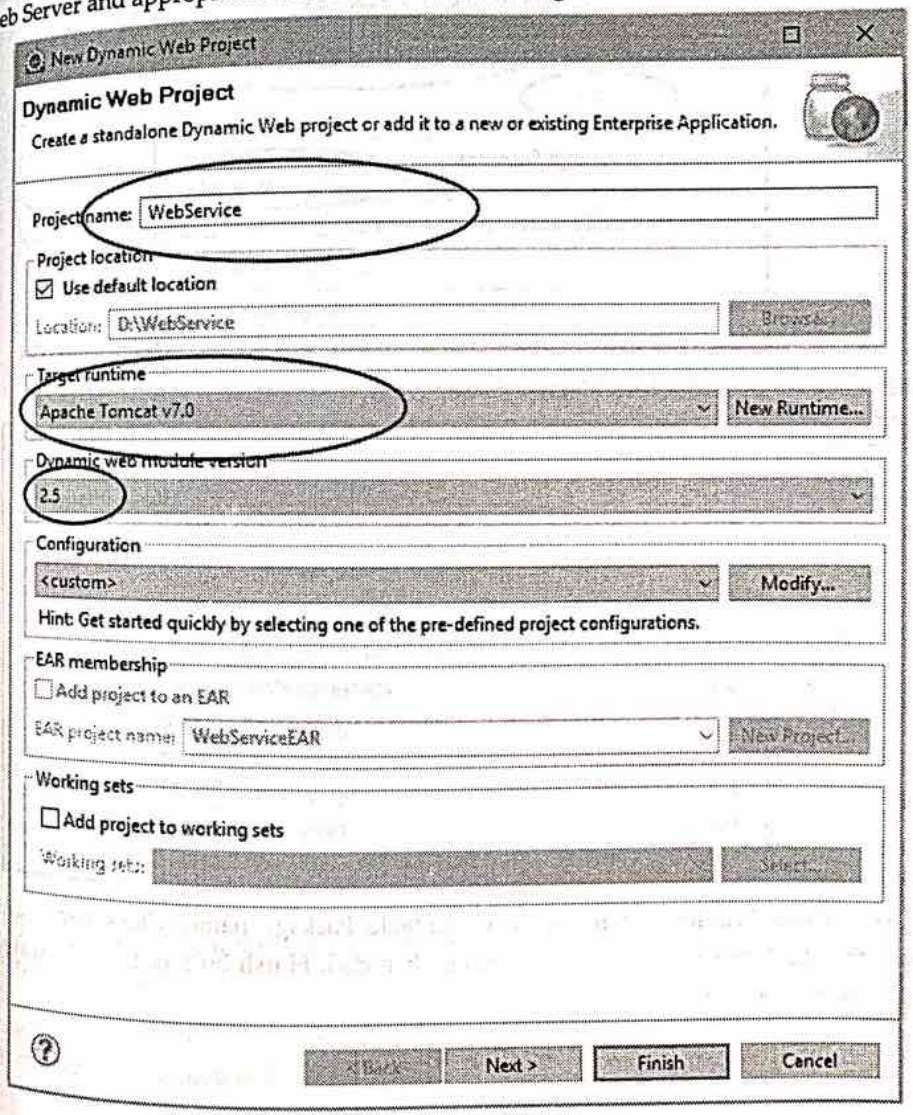
Let us now discuss step by step - how to create a simple Web Service.

Example 4.8.1 *Create a web service in java environment to return the sum of two integers with necessary deployment procedures.*

Solution :

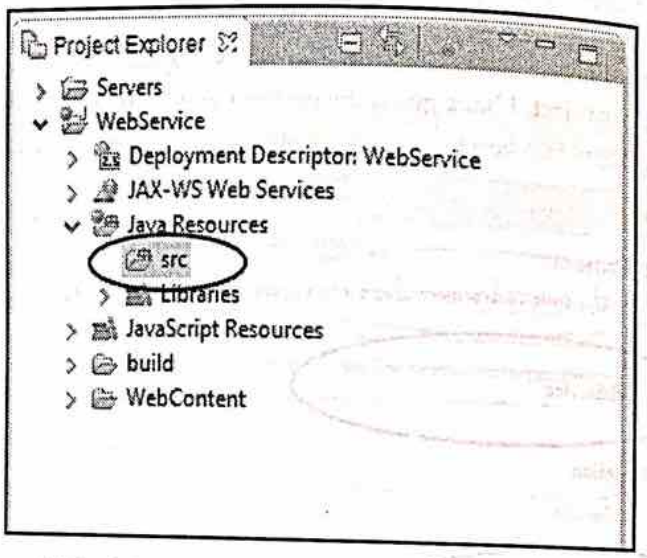
Stage I : Creating Server

Step 1 : Click on Eclipse icon. Click on File->New->Dynamic Web Project. Give the suitable name to the project. I have given the project name as **WebService**. Set the Tomcat Web Server and appropriate version number. Following screenshot illustrates this.

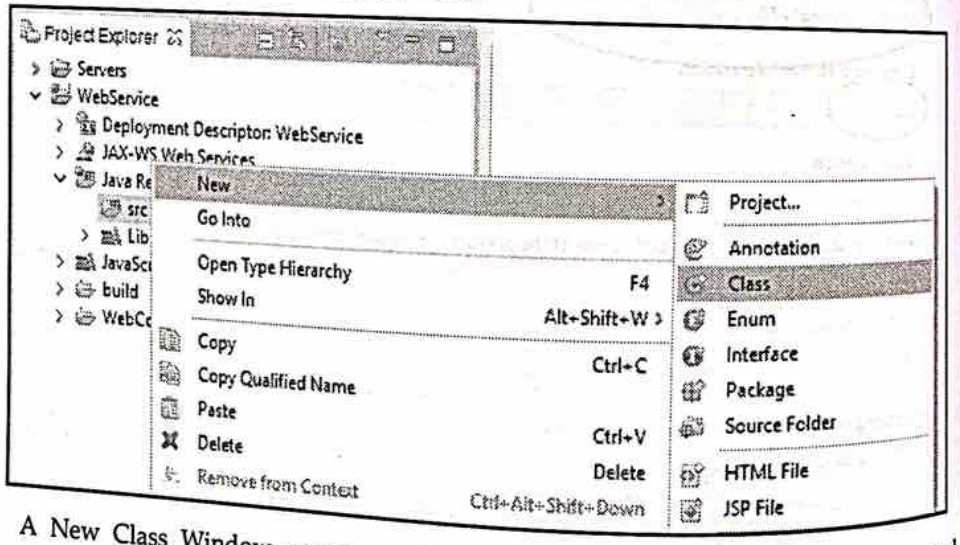


Click **Finish** button.

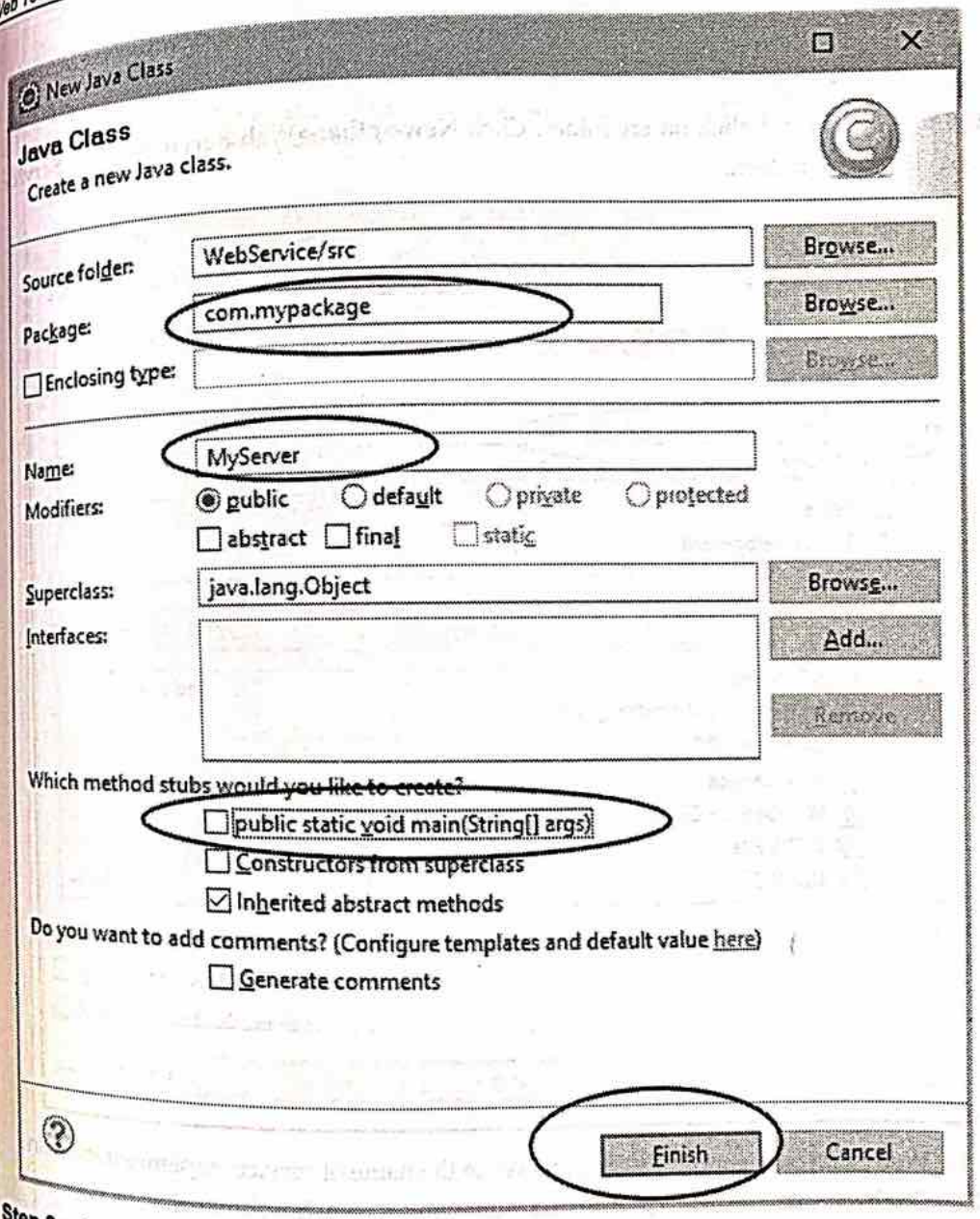
Step 2 : Locate src folder under the project in WebService project. The snapshot of Project Explorer is as shown below



Right click on src folder Select **New-> Class**



A New Class Window opens up. Give suitable Package name, Class name and uncheck the check box for void main function. Then click **Finish** button. It is illustrated by following screenshot.



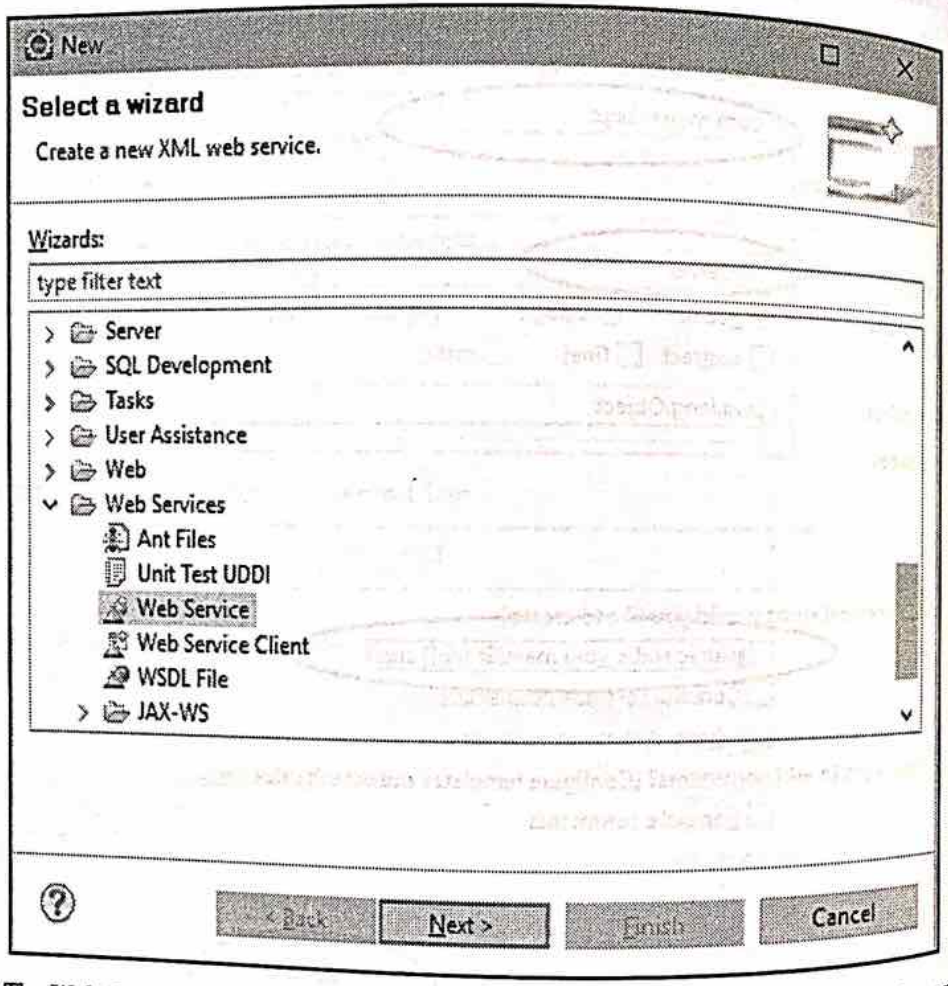
Step 3 : Now write the following code for **MyServer.java** .

```
package com.mypackage;

public class MyServer {
    public int add(int a,int b){
```

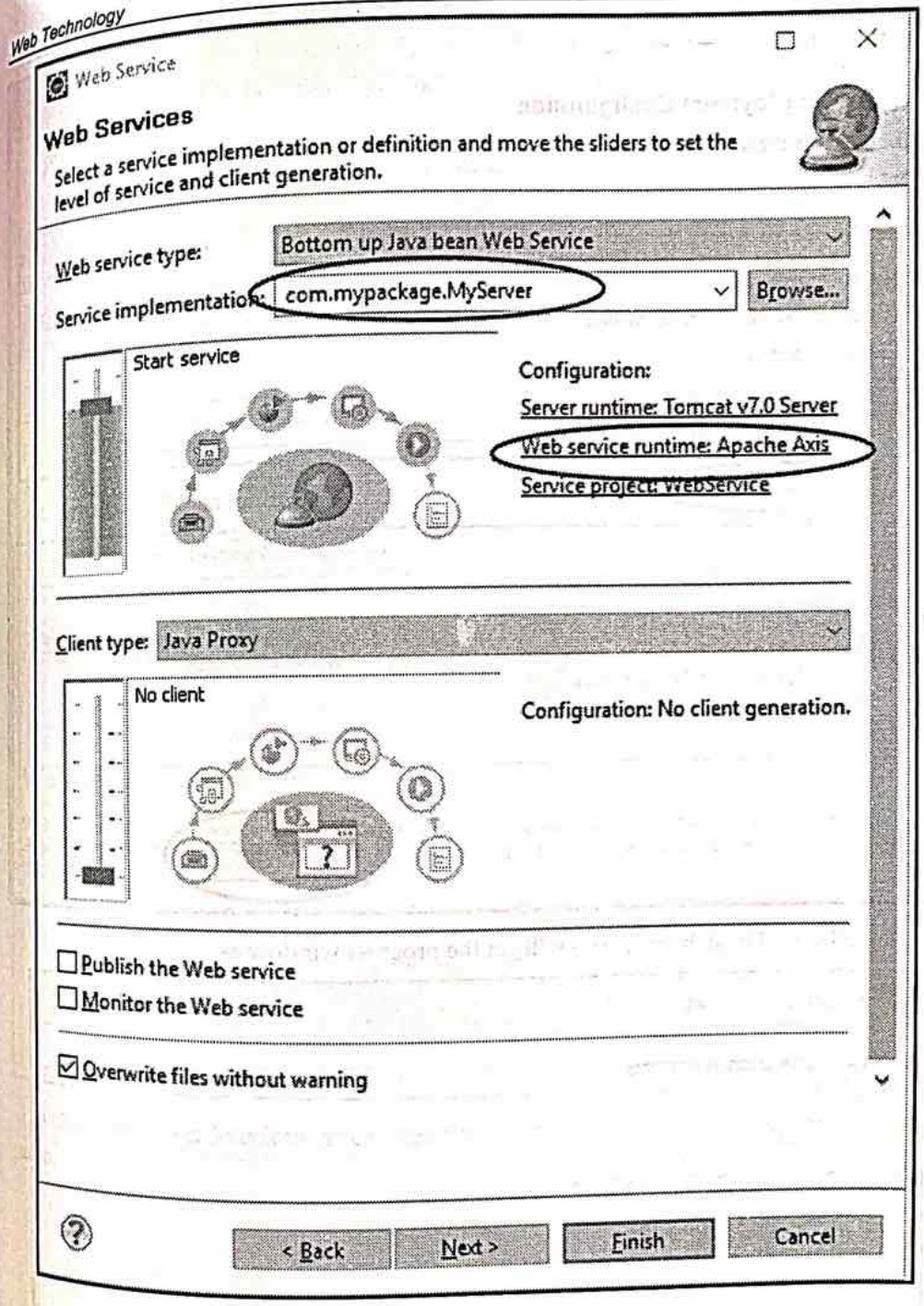
```
    return a+b;
  }
}
```

Save the above code.

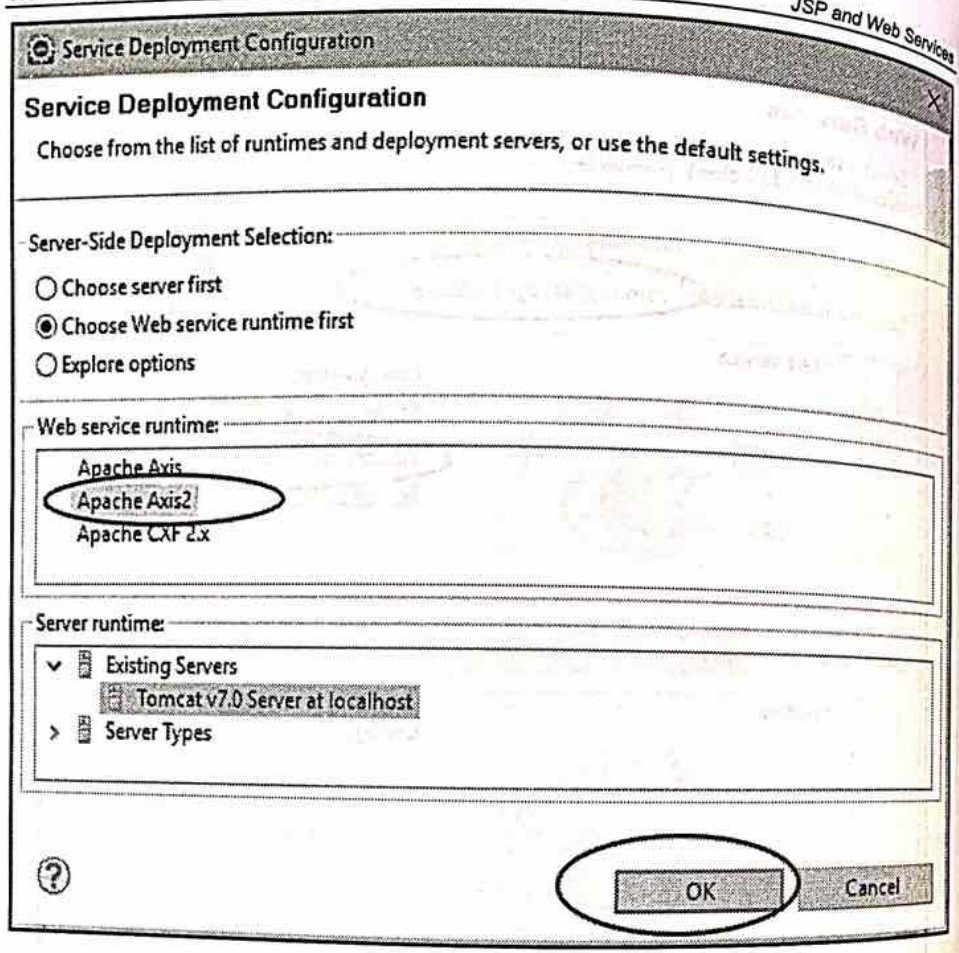
Step 4 : Now right click on **src** folder. Click **New->other->Web Services->Web Service.** Then click **Next** button.



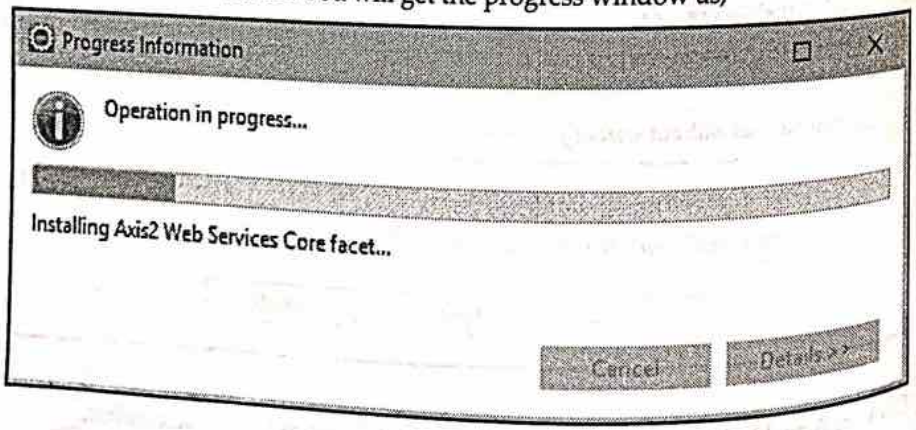
The **Web Services** window will appear. Write the name of service implementation and click on **Web service runtime:Apache Axis.** Following screenshot shows this -



Click on **Apache Axis2** and then click **Ok** button. Refer following screenshot.



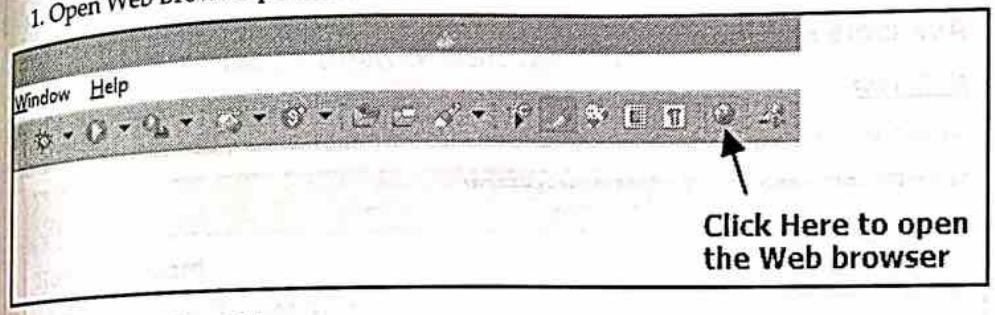
Then click on Finish button. You will get the progress window as,



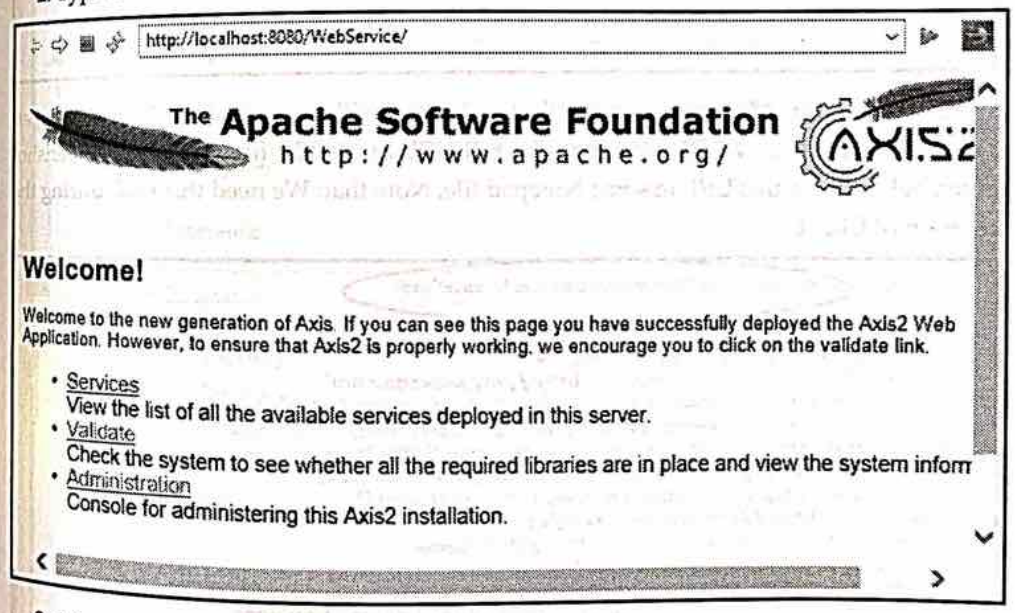
This will generate the WSDL file for your application.

Step 5 : To view the generated WSDL file we need to open up the Web Browser in Eclipse. For that

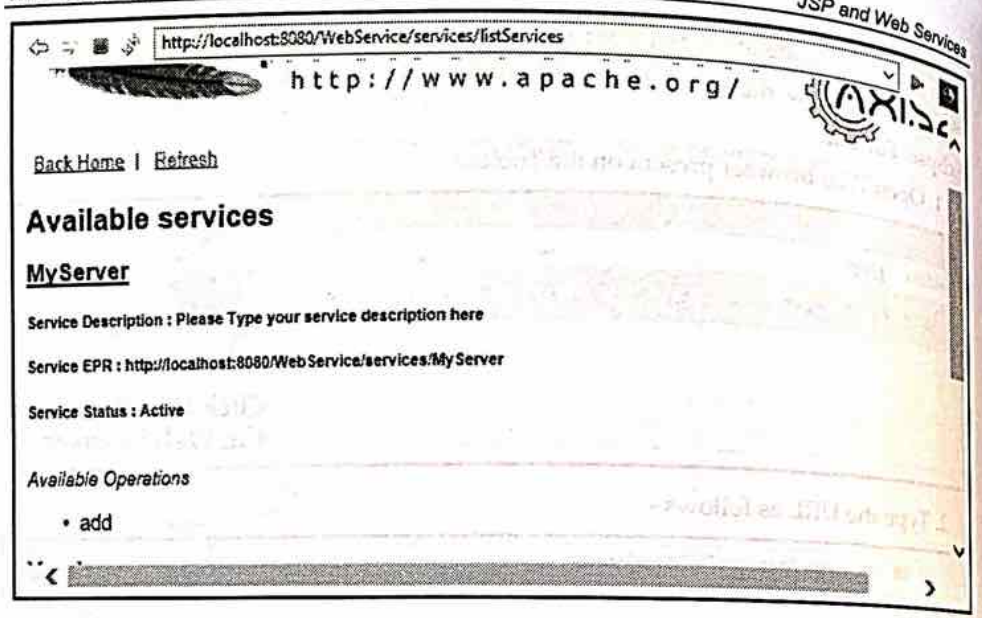
1. Open Web Browser present on the Toolbar



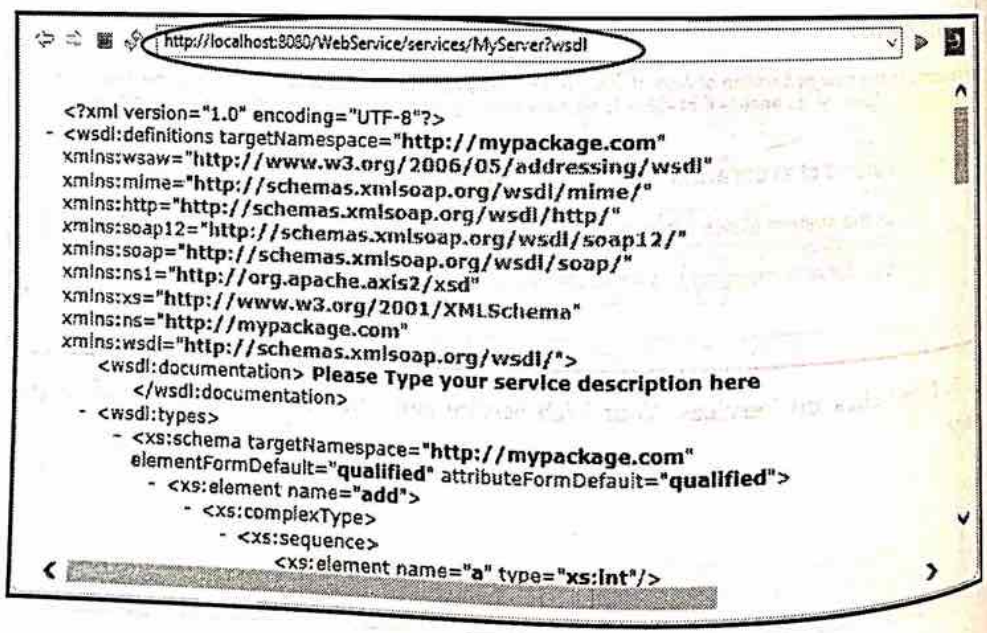
2. Type the URL as follows -



3. Now click on **Services**. Your Web Service name **MyServer** can be located on the page.



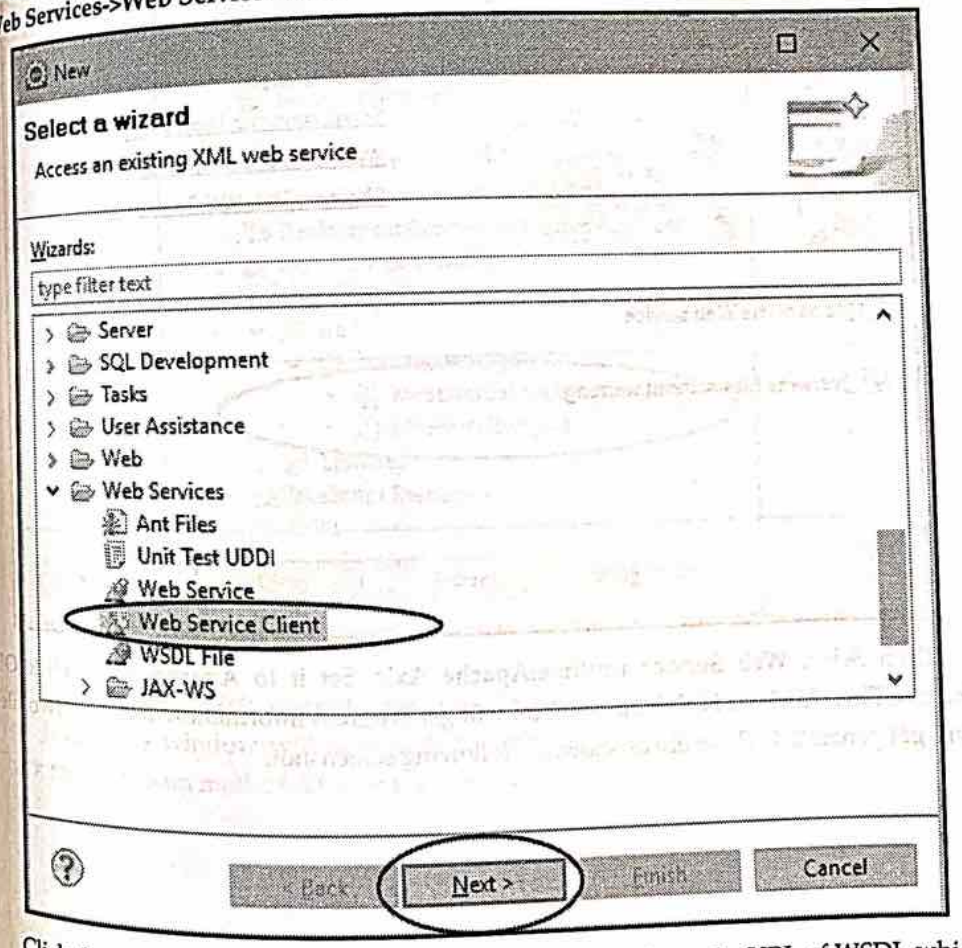
4. Just click on **MyServer** link another web page will be opened up in the browser. This is actually your WSDL file. Copy the full URL for WSDL file marked in screenshot given below. Save this URL in some Notepad file. Note that: We need this URL during the creation of Client.



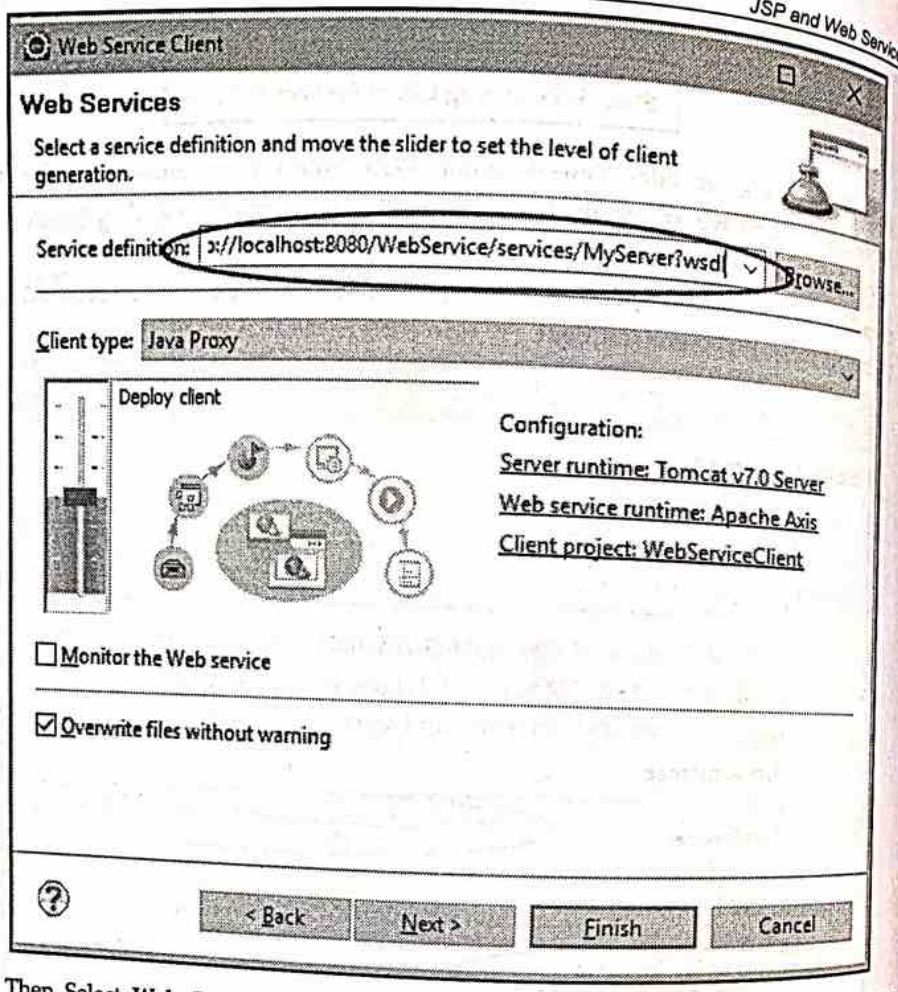
Stage II : Creating Client for Web Service

Step 6 : Now go to **File->New->Dynamic Web Project**. Give suitable name for your application. Here we are creating a client for the Web service created in above steps. Hence I have given the name **WebServiceClient**. Then click **Finish** button.

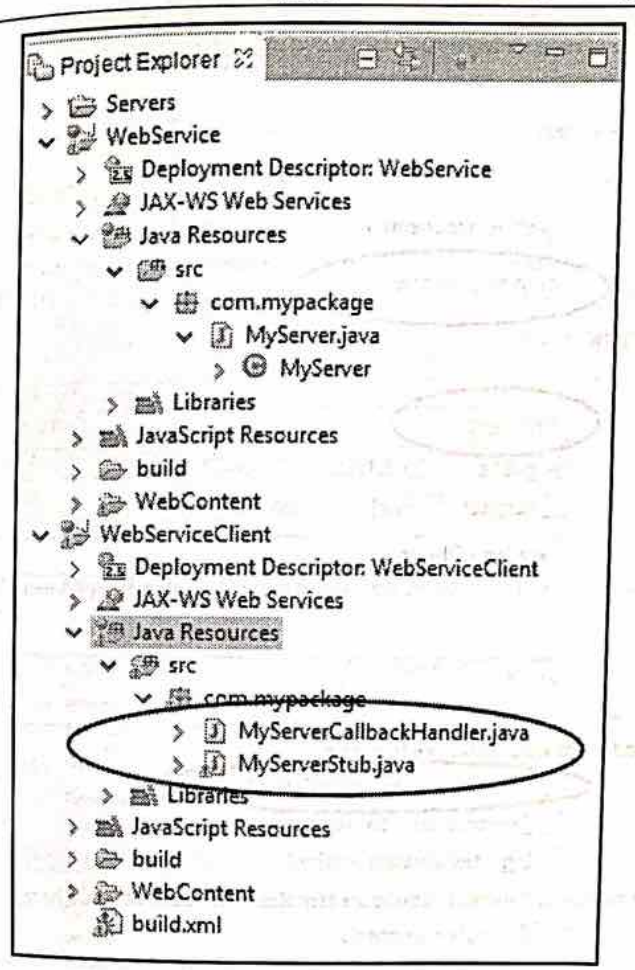
Step 7 : Now under **WebServiceClient**, right click on **src** folder. Click on **New->Other->Web Services->Web Service Client**



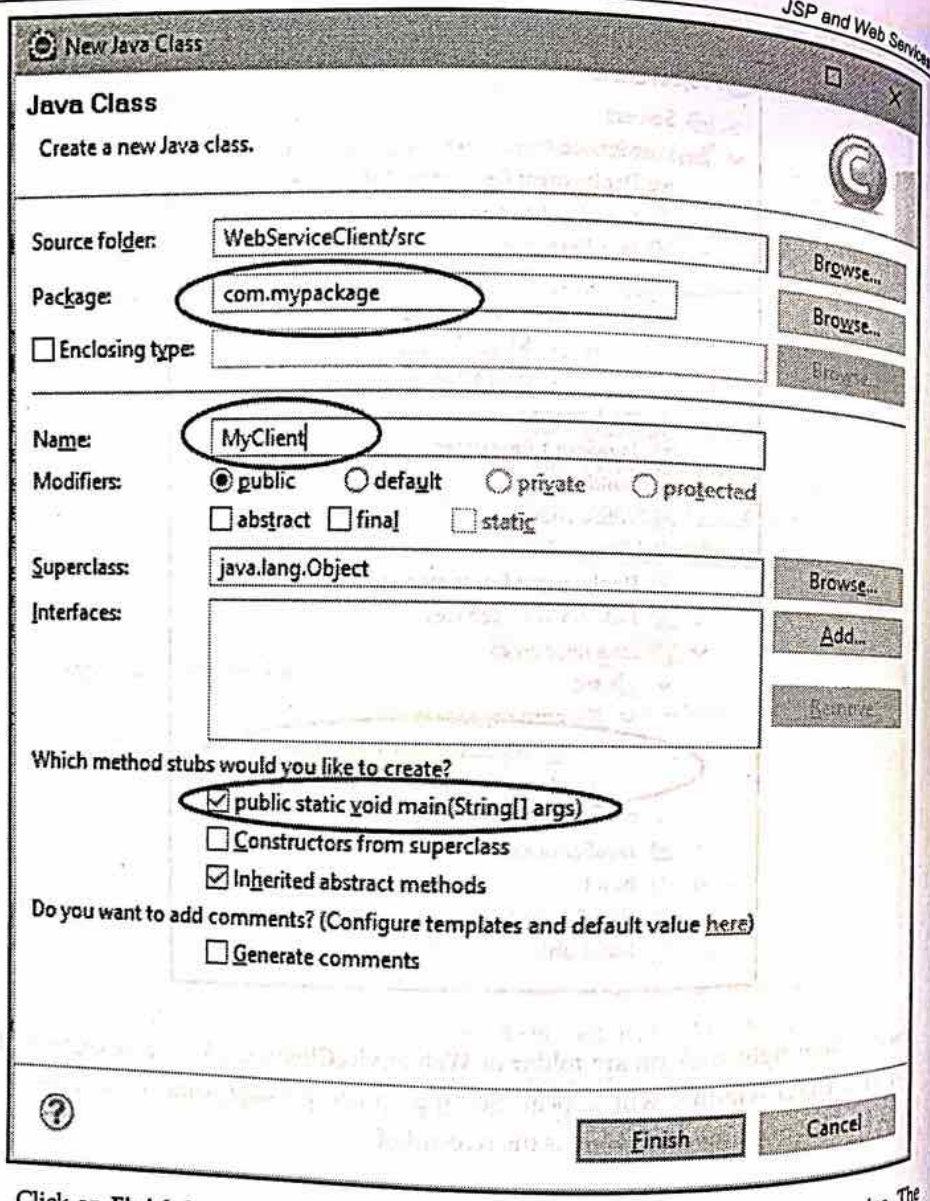
Click the **Next** button. Now on **Web Services** window. Copy the URL of WSDL which you have copied in Step 5(4).



Then Select **Web Service runtime:Apache Axis**. Set it to **Apache Axis2** click OK button. Then Click on **Finish** button. You will get Progress Information and then two files will get generated. These are as shown in following screenshot.



Step 8 : Now right click on **src** folder of **WebServiceClient**. Click on **New->Class**. The **New Java Class** window will appear. Set appropriate package name, class name and Check for void main method. Here is the screenshot.



Click on **Finish** button. You can now write the Java Client code for web service. The code is as follows,

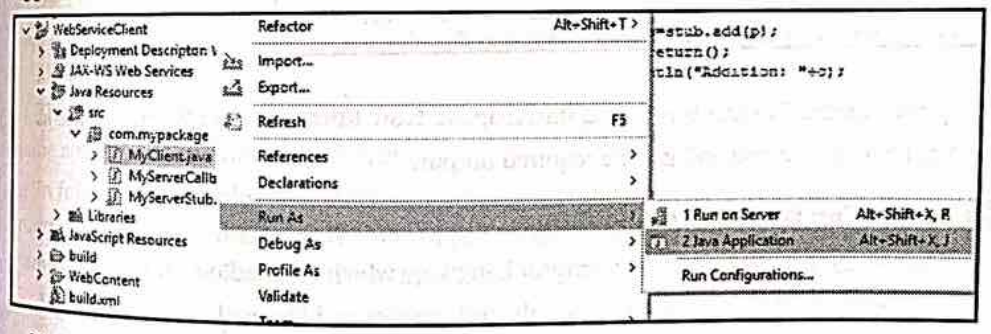
MyClient.java

```
package com.mypackage;
import com.mypackage.MyServerStub.Add;
import com.mypackage.MyServerStub.AddResponse;
```

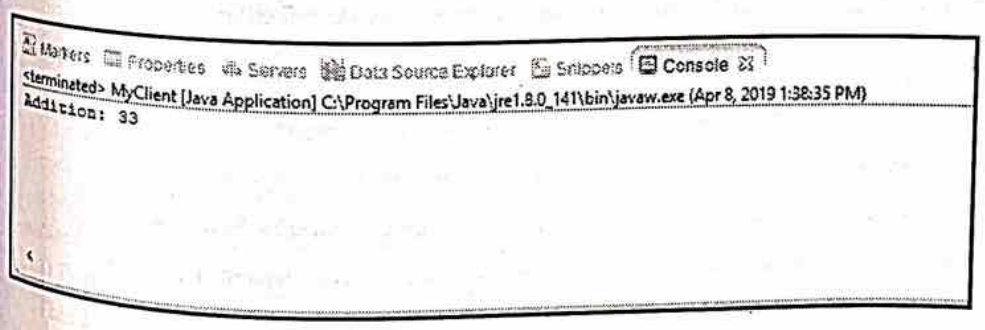


```
public class MyClient {
  /**
   * @param args
   */
  public static void main(String[] args) throws Exception{
    // TODO Auto-generated method stub
    MyServerStub stub=new MyServerStub();
    Add p=new Add();
    p.setA(11);
    p.setB(22);
    AddResponse res=stub.add(p);
    int c=res.get_return();
    System.out.println("Addition: "+c);
  }
}
```

Step 8 : Right click on **MyClient.java** file in Project Explorer. Click on **Run As -> Java Application**



You will get the output at the Console window as follows,



Troubleshooting :

We may get following error on running the above client

```
Exception in thread "main" org.apache.axis2.AxisFault:...
...
...
```

To overcome through this Exception, we need to make some changes in **services.xml** file. It can be located in the Web Services folder. To locate this **xml** file, follow the path as **Web Content-> WEB-INF-> Services->MyServer->META-INF. Locate services.xml**

Change following lines

```
<messageReceivermep=http://www.w3.org/2004/08/wsdl/in-only
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver" />
<messageReceivermep=http://www.w3.org/2004/08/wsdl/in-out
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
```

To

```
<messageReceivermep=http://www.w3.org/ns/wsdl/in-only
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"/>
<messageReceivermep=http://www.w3.org/ns/wsdl/in-out
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
```

Stop the Apache Tomcat Server and start it again. Now Run your **MyClient.java** file as Java Application and you will get the required output.

4.9 Describing Web Services : WSDL

- The WSDL is a Web Service Descriptor Language which is based on XML.
- The purpose of this file is to describe the web service and the method of accessing it.
- WSDL is a W3C recommendation.
- It contain various definitions. Following elements are described in WSDL file.

Elements	Description
types	It specifies the data types of the symbols used by the web service.
messages	It specifies the messages used by the web service.
portType	It specifies the name of the operation (function) defined by the web service
binding	It specifies the name of the protocol of the web service, typically it is SOAP.

1) Types

If we declare and use some variable in the web service then their data types can be specified.

For example -

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SimpleCalculator" targetNamespace="http://tempuri.org/wsdl" >
 <types>
  <schema targetNamespace="http://tempuri.org/types">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
  <complexType name="CalciClass">
   <sequence>
    <element name="operator1" type="double"/>
    <element name="operator2" type="double"/>
    <element name="operation" type="String"/>
   </sequence>
  </complexType>
 </schema>
</types>
```

2) Messages

The messages define the **part** of the element along with the data types. The message name specified the name of the function.

In following example the **HelloFun** is a name of the function which is declared in the interface named **MyInterface.** The return parameter is specified by **name="MyInterface_HelloFunResponse".**

For example -

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SimpleHelloMessage" targetNamespace="http://tempuri.org/wsdl" >
<types/>
 <message name="MyInterface_HelloFun">
  <part name="String_1" type="xsd:string"/>
 </message>
 <message name="MyInterface_HelloFunResponse">
  <part name="result" type="xsd:string"/>
 </message>
```

3) PortTypes

This is an important element of the WSDL file. It defined various **operations** that can be performed by the web service.

Various operation types are -

Elements	Description
one-way	This operation will send the message but it will not get any response.
request-response	This operation will send a message and will receive a response.
Solicit-response	This operation will send a message and will wait for a response.
Notification	This operation will send a message and will not wait for a response.

The messages in portTypes define the name of the operation along with the prefix tns where tns stands for TypeNamespace.

4) Binding

The binding defines the message format and details of the protocol.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions  name="SimpleHelloMessage"  targetNamespace="http://tempuri.org/wsdl" >
<types/>
...
...

<binding name="MyInterfaceBinding" type="tns:MyInterface">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
   <operation name="HelloFun">
...
...
...
```

There are three main elements -

binding

The binding defines **name** which defines the name of the binding and **type** attributes specifies the port which is typically the name of the interface file.

soap:binding

The soap:binding defines **transport** and **style** attributes. The style is remote procedure call which allows the distributed clients to access the web service.

operation

The operations defines the port operations. Each operation contains the **soap:operation, input** and **output** elements. The **soap:body** element provides the communication details such as whether or not the associated message is encoded and if it is encoded then the **encodingStyle** is specified.

5) Service

The service element specifies the name for overall web service.


```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SimpleHelloMessage" targetNamespace="http://tempuri.org/wsdl" >
<types/>
...
...
...
<service name="SimpleHelloMessage">
  <port name="MyInterfacePort" binding="tns:MyInterfaceBinding">
   <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
  </port>
</service>
</definitions>
```

4.10 Communicating Object Data : SOAP

- **Simple Object Access Protocol (SOAP)** is a protocol based on XML. It is used by the web services for exchange of information. It is a W3C recommendation.
- This exchange of information is done over HTTP. This is a platform and language independent protocol.
- The client-server communication is based on RPC. The HTTP does not designed to handle the distributed objects that are required by the RPC. Hence another application protocol is build over HTTP which popularly known as SOAP. SOAP allows to talk different applications that are running in two different operating systems.

4.10.1 Structure of SOAP

- The structure of the SOAP is defined by four building blocks.

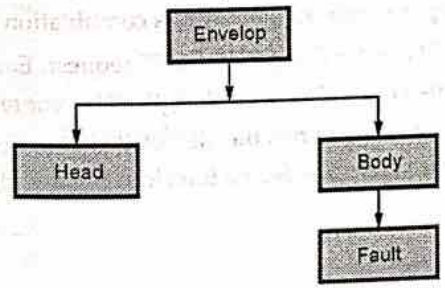


Fig. 4.10.1 SOAP building blocks

1) Envelop

The Envelop is the root of any SOAP document. It identifies that the XML document as a SOAP message. It defines two things - **Namespace** and **encodingStyle**. The Namespace value can be given as -

```
xmlns:soap="http://www.w3.org/2001/12/soap-envelope".
```

The encoding style is useful to define the data types used in the document. For example the value of encoding can be given as -

```
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding"
```

2) Header

The SOAP header field is optional. It contains the header information. The header contains three attributes such as **mustUnderstand, actor** and **encodingStyle**

The **mustUnderstand** attribute specifies 0 or 1. If the value is 1 then it means that the receiver should process the header.

Normally the SOAP message travels over a message path having the endpoints at the two ends. The SOAP **actor** is used to specify the URI for this endpoint.

The **encodingStyle** is useful to define the data types used in the document

The SOAP header defines how the recipient should process the SOAP message.

3) Body

The required SOAP Body element contains the actual SOAP message intended for the ultimate endpoint of the message.

4) Fault

This is optional element of the SOAP message. It is used to represent the error code. It consists of elements such as **faultcode** which is a code for fault. The **faultstring** message gives the details about the fault.

4.10.2 SOAP and HTTP

HTTP communicates over TCP/IP. HTTP is based on Request-Response mechanism. SOAP is a HTTP Request-Response protocol. SOAP is a combination of HTTP and XML.

The SOAP request can be HTTP GET or HTTP POST request. Each request consists of two HTTP headers : **content-type** and **content-length.** The content-type specifies the MIME type and the content-length specifies the size of the message in bytes. After this header the body part begins which defines SOAP Envelop, Header, Body and Fault.

Example

The SOAP request can be

```
POST /mycreation HTTP/1.0
Host: www.mywebsite.com
Content-Type: text/xml; charset=utf-8
Content-Length: 500
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
```



```
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<SOAP-ENV:Body xmlns:m="http://www.mywebsite.com/mycreation">
  <m:GetDetails>
    <m:Name>ABCD</m:Name>
  </m:GetDetails>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The SOAP response can be

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 500
<?xml version="1.0"?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <SOAP-ENV:Body xmlns:m="http://www.mywebsite.com/mycreation">
    <m:GetDetailsResponse>
      <m:ItemDetails>This is a my new creation</m:ItemDetails>
    </m:GetDetailResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4.10.3 SOAP Encoding

SOAP makes use of XML Schema to encode the programming constructs. Following are examples of SOAP encoding.

1. Encoding of Struct Data

Following is a code fragment which can be encoded as follows -

```
<xs:element name="Student">
<xs:complexType>
  <xs:sequence>
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="RollNo" type="xs:integer"/>
  <xs:element name="address" type="tns:Address"/>
  </xs:sequence>
  <xs:element name="Address">
  <xs:complexType>
    < xs:sequence>
     < xs:element name="street" type="xsd:string"/>
     < xs:element name="city" type="xsd:string"/>
     < xs:element name="country" type="xsd:string"/>
    </ xs:sequence>
  </xs:complexType>
</e:element>
```

The following is an example of how the preceding schema definition could be subsequently used in a SOAP encoding.

```
<e:Student>
<Name>AAA</Name>
<RollNo>11</RollNo>
<address>
    <street>XYZ Road</street>
    <city>PQR City</city>
    <country>India</country>
</address>
</e:Student>
```

2. Encoding of Arrays

An array in SOAP is of type **SOAP-ENC:Array**

```
<element name="myArray"
type="SOAP-ENC:Array"/>
```

The preceding schema can be represented in SOAP encoding as follows -

```
<myArray SOAP-ENC:arrayType="xsd:int[3]">
<number>1</number>
<number>2</number>
<number>3</number>
</myArray>
```

The above encoding shows that there is an array of three integers. All these members of array are called **number.**

4.10.4 RPC Representation

- The RPC (Remote Procedure Call) is a mechanism used when the client makes a call to a method that resides on another machine. The SOAP has a support for RPC representation.
- The **style** attribute of **soap:binding** element in the WSDL document makes use of string "rpc" to denote that the communication with the client is in RPC style.

```
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="rpc"/>
```

- The **soap:body** element indicated using an encoding specified by SOAP 1.1 to represent RPC.
- In SOAP specified representation of RPC calls, the call is modeled as **struct.**
- The specified structure used in SOAP representation of RPC call has a name for the remote procedure call and has one accessor for each parameter that is to be passed to the procedure. This struct us then encoded as XML markup. This markup is then made a child of SOAP body element.

For example - Following is a soap document sent by the client to display message web service.

```
<?xml version="1.0" encoding ="UTF-8"?>
<env:Envelop
xmlns:env="http://schemas.xmlsoap.org/soap/envelop/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:enc="http:// schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://tempuri.org/wsdl"
xmlns:ns1="http://tempuri.org/types"
env:encodingStyle="http:// schemas.xmlsoap.org/soap/encoding/">
<env:Body>
<ns0:HelloMsg>          ← Call to procedure which is RPC
  <msg_1 xsi: type=:xsd:string">Welcome</msg_1>
</ns0:HelloMsg>
</env:Body>
</env:Envelop>
```

Part III : Struts

4.11 Overview

SPPU : Dec.-18, Marks 4

- Struts is a framework software used for developing the Java Web Applications.
- The struts framework was initially created by Craig McClanahan and donated to Apache Software Foundation.
- It is an open source product.
- The latest commonly used release is **struts 2.**

Features

1. It is based on Model view Controller (MVC) architecture.
2. The action class is POJO (Plain Old Java Object) that means it is a simple Java class.
3. Struts2 provides the support to AJAX technology. That means only required field can be sent to the server and no need to send entire page. Due to this feature the execution is fast.
4. The struts2 framework can be integrated with other applications like Hibernate, Spring and so on.
5. Struts2 provides tag support such as UI tags, Data tags, control tags etc.
6. It provides support for views and templates.
7. Struts2 support multiple views for the same application.

Advantages of Struts

1. The complex web applications can be built using simple Model, View and Controller components.
2. The designs made using struts are simpler, consistent and easy to maintain.

Review Question

1. *Write various features of struts framework.*

SPPU : Dec.-18, Marks 4

4.12 Architecture

SPPU : May-18,19, Marks 6

- The struts framework is based on **Model, View** and **Controller** architecture (MVC).

I. Normally when client wants any web page he demands it using get or post request or by clicking Submit button, then a controller is invoked.

II. A Controller is nothing but a Java class in which **business logic** is written. The job of Controller is to take user input and pass it to the **Model**. The controller in turn invokes the **Model.**

III. Model stores the user's data may be in databases or so, and returns some result to the Controller.

IV. Using this result, the Controller figures out user requirements and accordingly forwards the result to the View page.

V. Finally required view can be obtained as a response (i.e. a display of required web page is obtained) on the web browser. This can be diagrammatically shown as below -

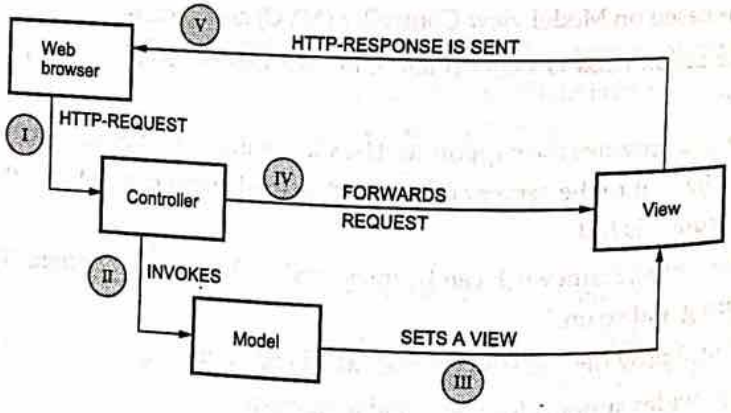


Fig. 4.12.1

In struts,

- **Model** components provide a model for "business logic" and data behind struts program.
- **View** components in struts are those components that present the information to the user or accept the user input. Typically these components are JSP or HTML pages. View components are responsible for providing the display of required information. Struts provide large number of JSP custom tags or **struts tags** to simplify the development of view components.
- **Controllers** are the Java classes which allow developers to remove error handling activities from JSP pages. These classes are typically referred as **action classes.**

Review Question

1. *Draw and explain neat diagram which depicts MVC to the struts architecture.*

SPPU : May-18,19, Marks 6

4.13 Execution of First Struts Applications

Environment : Before execution of any struts programs following software must be installed.

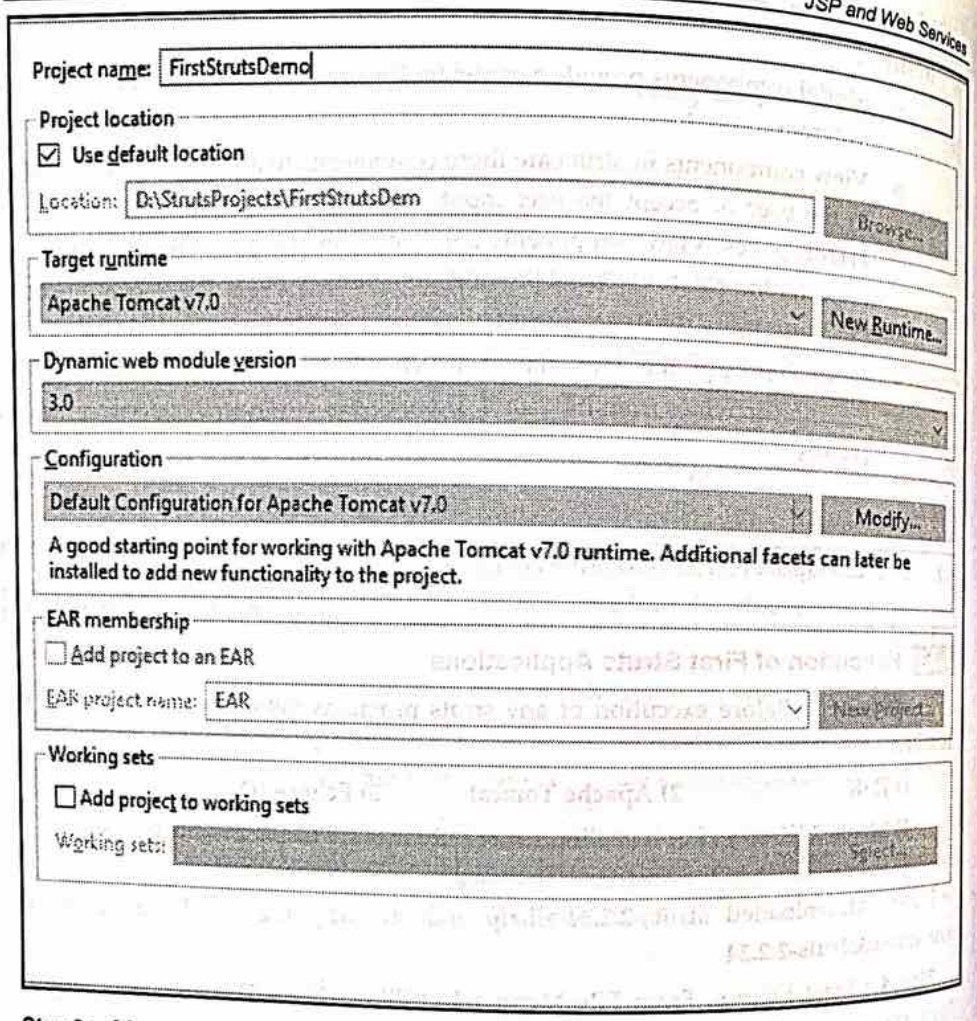
1) JDK **2) Apache Tomcat** **3) Eclipse IDE**

4) Struts 2 library : For installing struts 2 libraries get them downloaded from

https ://struts.apache.org/download.cgi

I have downloaded **struts-2.2.34-all.zip** and on unzipping it will give directory structure **c:\struts-2.2.34**

Step 1 : Start Eclipse. From **File Menu select File -> New->Dynamic Web Project.** Type the name of the project. In my case I have created the project by the name **FirstStrutsDemo.** The screen-shot will be Just click next button with default options and check the **Generate Web.xml deployment descriptor** option. This will create a dynamic web project in Eclipse.



Step 2 : Now copy the Jar files present in the **c:\struts-2.2.34\lib** folder to our project's WEB-INF\lib folder. Following files need to be copied

commons-fileupload-1.3.2

commons-io-2.2

commons-lang-2.4

commons-lang3-3.2

commons-logging-1.1.3

freemarker-2.3.22

javassist-3.11.0.GA

ognl-3.0.21

struts2-core-2.3.34

xwork-core-2.3.34

Step 3 : Now create an action class which is a Java class. It should be created under **Java-Resourcs->src** folder. Under **src** folder create a package with suitable name. I have created package named **myPkg** and Java source class as **MyBigHello.java**. The code for this is as follows -

MyBigHello.java

```
package myPkg;
public class MyBigHello {
    private String name;
    public String execute() throws Exception {
      return "success";
    }
    public String getName() {
      return name;
    }
    public void setName(String name) {
      this.name = name;
    }
}
```

The action class gets executed when user clicks the URL. Based on it, the JSP page will be displayed.

Step 4 : Now we will create a **view** page. I have named it as **MyHello.jsp.** in the **WebContent** folder in your eclipse project. To do this, right click on the **WebContent** folder in the project explorer and select New >JSP File.

MyHello.jsp

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<html>
  <head>
    <title>My First Struts Demo Application</title>
  </head>
  <body>
    <h3>Welcome my friend, <s:property value = "name"/></h3>
  </body>
</html>
```

In above code, the taglib directive tells us the Servlet container that this page is using Struts 2 tag and the tags will be preceded by s.

The **name** property define in **MyBigHello.java** class will be returned by the function **getName()** of **MyBigHello** class.

Step 5 : Now we will create the main page. In struts 2 the main page is denoted by **index.jsp** in the WebContent folder. This file serves as the initial action URL and calls the method defined in **MyBigHello** class, accordingly the display of **MyHello.jsp** page will be updated.

Index.jsp

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
  pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>My First Struts Demo Application</title>
  </head>
  <body>
    <h2>My Struts2 Application</h2>
    <form action = "hello">
      <label for = "name">Enter your name</label><br/>
      <input type = "text" name = "name"/>
      <input type = "submit" value = "Submit"/>
    </form>
  </body>
</html>
```

The **hello** action defined in the above view file will be mapped to the **MyBigHello** class and its execute method using **struts.xml** file. When a user clicks on the Submit button it will cause the Struts 2 framework to run the execute method defined in the **MyBigHello** class and based on the returned value of the method, an appropriate view will be selected and rendered as a response.

Step 6 : There is a file that ties **MyBigHello class(model)** with **MyHello(view)**. This file is **struts.xml**. This file must be present in the classes folder. Hence under WebContent/WEB-INF, create a folder named classes and add new file named struts.xml. The code for this file is

struts.xml

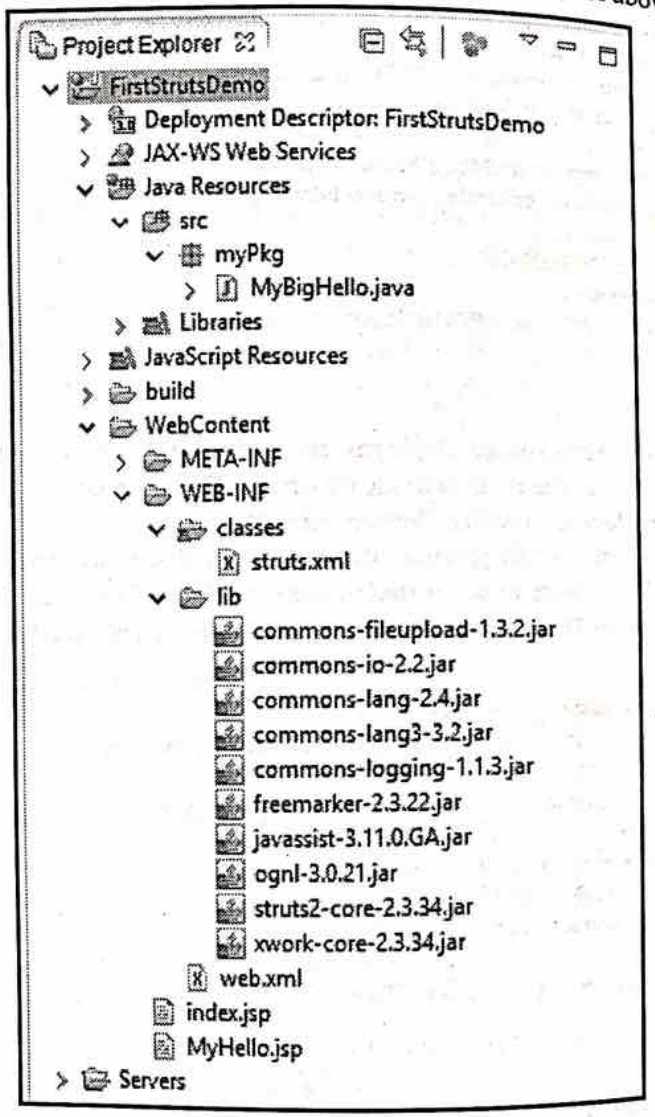
```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name = "struts.devMode" value = "true" />
  <package name="default" extends="struts-default">
    <action name = "hello"
      class = "myPkg.MyBigHello"
      method = "execute">
      <result name = "success">/MyHello.jsp</result>
    </action>
  </package>
</struts>
```

Step 7 : Now create web.xml i.e. deployment descriptor file. This file acts as an entry point for the request made by struts 2. The entry point will be **filer defined**. Hence, we will define an entry of org.apache.struts2.dispatcher.FilterDispatcher class in web.xml. The web.xml file needs to be created under the WEB-INF folder under WebContent. The **web.xml** generated by Eclipse can be modified as -

web.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xmlns = "http://java.sun.com/xml/ns/javaee"
  xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id = "WebApp_ID" version = "3.0">
  <display-name>Struts 2</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

Following screen-shot of Project-Explorer represents the location of above created files.

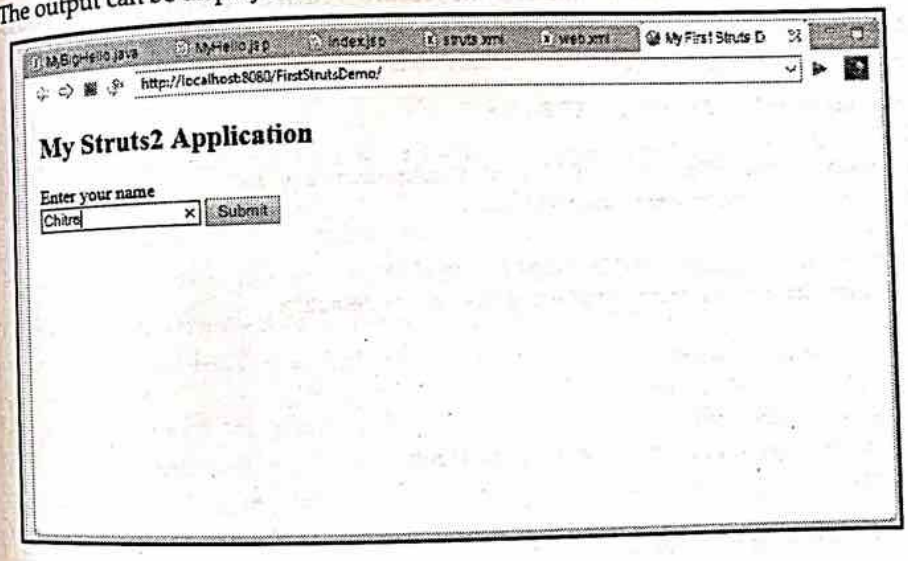


Step 8 : Execution of application :

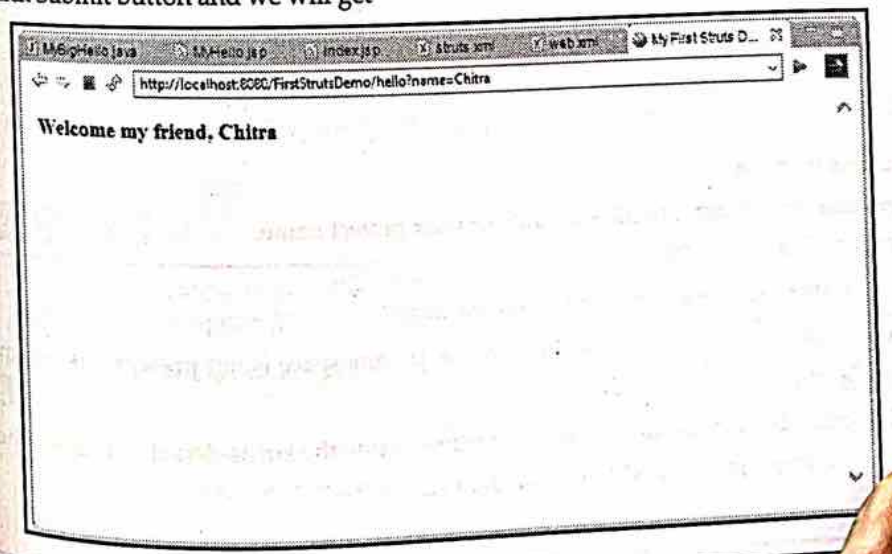
i) For executing the application right click on the project name and click **Export > WAR** File to create a War file. Then deploy this WAR in the Tomcat's webapps directory. For a **destination** click the **Browse** button and select the path c:/your_tomcat_directory/webapps folder by accepting the default name of the war file. Normally "your_project_name.war" file gets generated. Here we get **FirstStrutsDemo.war** in the c:/tomcat_directory/webapps.

ii) Now run the Tomcat Web server. By selecting the **Run configurations...** from the Menu bar of Eclipse. Select the Apache, under which select the Tomcat server which is same as the Tomcat server which you have already installed on your computer. I have selected Tomcat v7.0 as the version of my already installed Tomcat. Then as a destination select the path of your tomcat directory. After finishing this the Tomcat web server gets started.

The output can be displayed on the web browser as



Click Submit button and we will get



4.14 Configuration

SPPU : Dec.-18,19, Marks 6

There are two main configuration files used in struts and those are **struts.xml** and **web.xml**.

There is **struts.properties** file is used to override the default values of default.xml file provided by struts framework. So it is not mandatory.

struts.xml

The struts.xml file is used to initialize the resources such as inceptors, Actions classes. This file is present inside the **WEB-INF/classes** folder.

The structure of this file is

```
<?xml version = "1.0" Encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
 <constant name = "struts.devMode" value = "true" />
 <package name = "FirstStrutDemo" extends = "struts-default">

  <action name = "demo"
   class = "myPkg.TestAction"
   method = "execute">
   <result name = "success">/MyHello.jsp</result>
  </action>

  <-- more actions can be listed here -->

 </package>
 <-- more packages can be listed here -->

</struts>
```

1) Package Element

The package element specifies module or your project name.
It has following attributes -

i) **name :** The name must define the package

ii) **namespace :** This is optional attribute. If namespace is not present, / is assumed as the default namespace.

iii) **extends :** The package element mostly extends the struts-default package where interceptors and result types are defined.

2) Action Element

- This element represents action. Various attributes of this element are -

i) **name :** This is any name with which the action is associated.

ii) **class :** The name of the action class is specified. This name must be accompanied with the package name inside which the action class is defined.

iii) **method :** This is optional field. By default the **execute** method is associated with this attribute.

iv) **result name :** The result name can be success, error and so on. In struts, on result element the page to be displayed on success or error is specified.

web.xml

The web.xml is a configuration file which determines how elements of the HTTP request are processed by the servlet container. It is not strictly a Struts2 configuration file, but it is a file that needs to be configured for Struts2 to work.

The web.xml file needs to be created under the folder **WebContent/WEB-INF**.

This file contains the <welcome-file-list> tag using which the first file to be loaded on execution of struts project is specified. For example

```
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

The simple meaning of above code is that as soon as you execute your struts application, load the **index.jsp** file.

```
<filter-mapping>
  <filter-name>struts2</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

The above code indicates that <url-pattern> is /* means that all urls will be parsed by the struts filter.

Review Question

1. *What are the different configuration files are require to develop any struts application ? Explain each configuration file.*

SPPU : Dec.-18, 19, Marks 6

4.15 Actions

SPPU : Dec.-18, May-19, Marks 6

- Actions are important element of struts application. The actions are basically the Java programs that contain the processing logic and satisfy the requests made by the user.

- Each URL is mapped to these actions.
- Action performs two tasks -
  - Action plays an important role from request to view. This view is written in JSP.
  - Action directs the framework to display particular view as an response to the request.
- Action class is a POJO (Plain Old Java Object) class which has attributes and methods. There must be one method that should always be present in action class and that is - **execute()** method.
- By default **execute()** method is called when client request goes to action class.
- This class is bean class. That means it is a Java class containing **getXX** and **setXX** methods.
- To understand the **getXX** and **setXX** methods - consider that there is a Java class named Test inside which there are two attributes namely -name and roll. Then in this case **getName(), setName , getRoll()** and **setRoll()** methods will be created and defined. The sample code for such **Test** bean class would be

```
Public class Test
{
private String name;
private int roll;
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
public String getRoll() {
    return roll;
}
public void setRoll(int roll) {
    this.roll = roll;
}
```

Note that the attribute name must appear after get or set and the first letter of the method name must be capital

We can write simple **Java class** as action class or extend that class for **ActionSupport** class. This **ActionSupport** class implements an important interface named **Action**. The **Action** interface is as follows -

```
public interface Action {
    public static final String SUCCESS = "success";
public static final String NONE = "none";
public static final String ERROR = "error";
public static final String INPUT = "input";
public static final String LOGIN = "login";
```



```
public String execute() throws Exception;
}
```

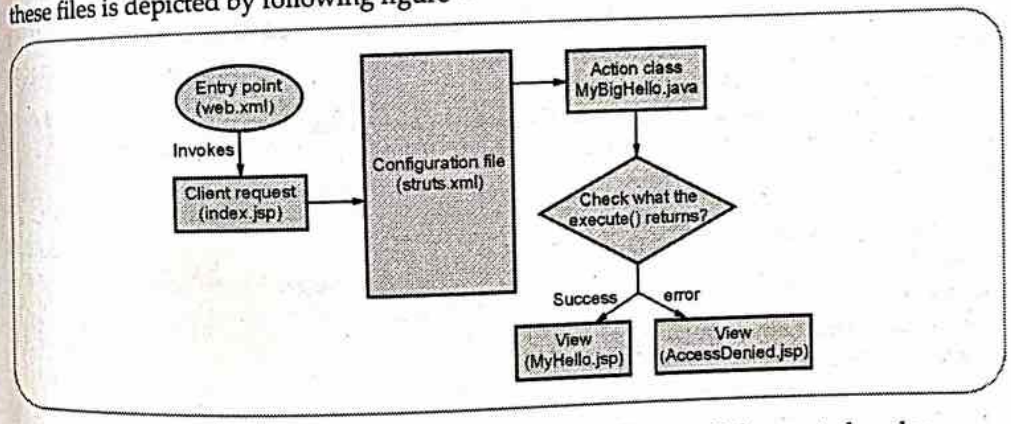
How to Implement Action class Using struts ?

Step 1 : Start Eclipse. From File Menu select **File -> New->Dynamic Web Project.** Type the name of the project.

Just click **next** button with default options and check the **Generate Web.xml deployment descriptor** option. This will create a dynamic web project in Eclipse

Step 2 : Now copy the required Jar files present in the **c:\struts-2.2.34\lib** folder to our project's WEB-INF\lib folder.

Before actually building the struts application just refer following figure. Because in this application, we are going to create multiple files. The relationship and purpose of these files is depicted by following figure -



Step 3 : Now create an action class which is a Java class. It should be created under **Java-Resourcs->src** folder. Under **src** folder create a package with suitable name. I have created package named **myPkg** and Java source class as **MyBigHello.java.** The code for this is as follows -

MyBigHello.java

```
package myPkg;
public class MyBigHello {
    private String name;
    public String execute() throws Exception {
        return "success";
    }
```

```
    public String getName() {
      return name;
    }
    public void setName(String name) {
      this.name = name;
    }
}
```

The action class gets executed when user clicks the URL. Based on it, the JSP page will be displayed.

Step 4 : Now we will create the main page. In struts 2 the main page is denoted by **index.jsp** in the **WebContent** folder. This file serves as the initial action URL and calls the method defined in **MyBigHello** class(We have defined this class in pervious step)

Index.jsp

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
  pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>My First Struts Demo Application</title>
  </head>
  <body>
    <h2>My Struts2 Application</h2>
    <form action = "hello">
      <label for = "name">Enter your name</label><br/>
      <input type = "text" name = "name"/>
      <input type = "submit" value = "Submit"/>
    </form>
  </body>
</html>
```

The **hello** action defined in the above view file will be mapped to the **MyBigHello** class and it executes method using **struts.xml** file. When a user clicks on the Submit button it will cause the Struts 2 framework to run the execute method defined in the **MyBigHello** class and based on the returned value of the method, an appropriate view will be selected and rendered as a response.

Step 5 : Create a file named **struts.xml**. This file must be present in the **classes** folder. Hence under WebContent/WEB-INF, create a folder named **classes** and add new file named **struts.xml**. The code for this file is

struts.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
 <constant name = "struts.devMode" value = "false" />
 <package name="default" extends="struts-default">
  <action name = "hello"
   class = "myPkg.MyBigHello"
   method = "execute">
    <result name = "success">/MyHello.jsp</result>
    <result name = "error">/AccessDenied.jsp</result>
  </action>
 </package>
</struts>
```

The **struts.xml** is a file that ties **MyBigHello class(model)** with **MyHello(view)**. This MyHello.jsp file is basically a view file. It is created in next step.

Step 6 : We will create a **view** page. I have named it as **MyHello.jsp.** in the **WebContent** folder in your eclipse project. To do this, right click on the **WebContent** folder in the project explorer and select New >JSP File.

MyHello.jsp

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<html>
 <head>
  <title>My First Struts Demo Application</title>
 </head>
 <body>
  <h3>Welcome my friend, <s:property value = "name"/></h3>
 </body>
</html>
```

The **name** property define in **MyBigHello.java** class will be returned by the function **getName()** of **MyBigHello** class.

Step 7 : If we do not enter the name as "Admin" then the error page will be displayed as

AccessDenied.jsp

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<html>
 <head>
```

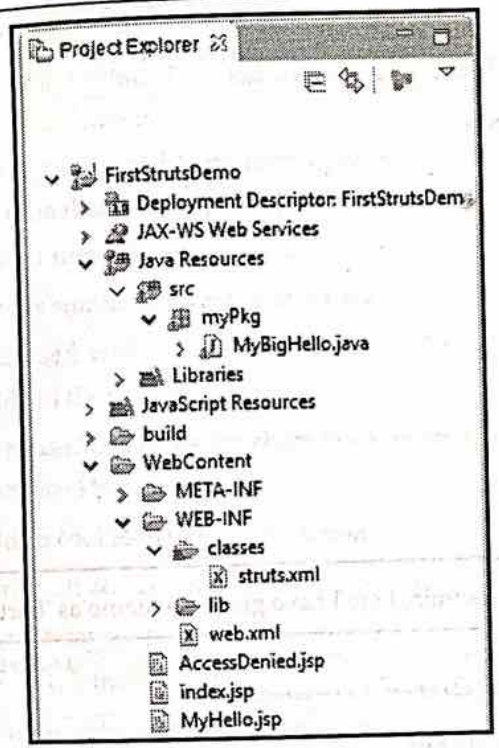
```
  <title>Access Denied</title>
 </head>
  <body>
  Error!!!You are not allowed to view this page
 </body>
</html>
```

Step 8 : Now create web.xml i.e. deployment descriptor file. This file acts as an entry point for the request made by struts 2. The entry point will be filer defined. Hence, we will define an entry of **org.apache.struts2.dispatcher.FilterDispatcher** class in **web.xml**. The **web.xml** file needs to be created under the WEB-INF folder under WebContent. The **web.xml** generated by Eclipse can be modified as -

web.xml

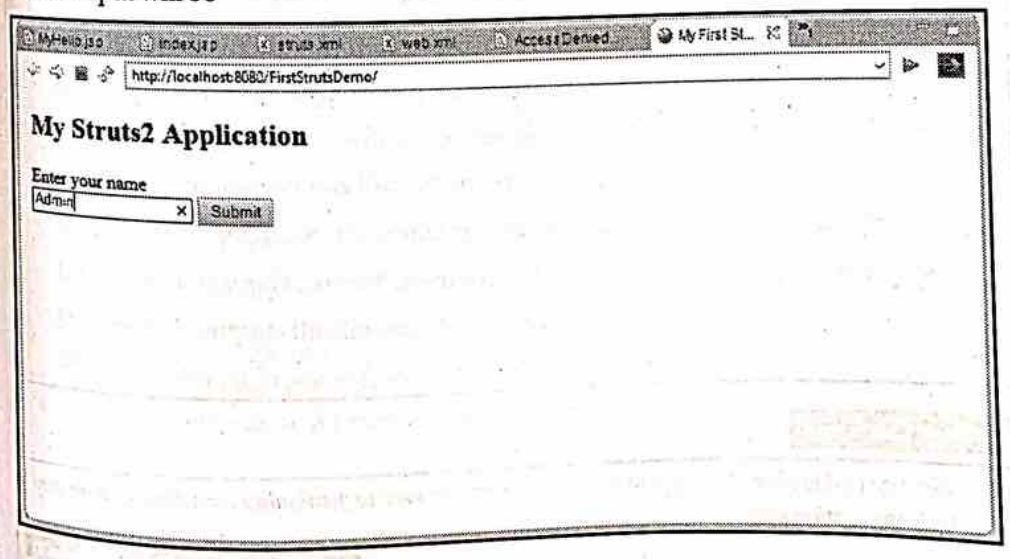
```
<?xml version = "1.0" encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns = "http://java.sun.com/xml/ns/javaee"
 xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id = "WebApp_ID" version = "3.0">
 <display-name>Struts 2</display-name>
 <welcome-file-list>
   <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>

 <filter>
   <filter-name>struts2</filter-name>
   <filter-class>
     org.apache.struts2.dispatcher.FilterDispatcher
   </filter-class>
 </filter>
 <filter-mapping>
   <filter-name>struts2</filter-name>
   <url-pattern>/*</url-pattern>
 </filter-mapping>
</web-app>
```

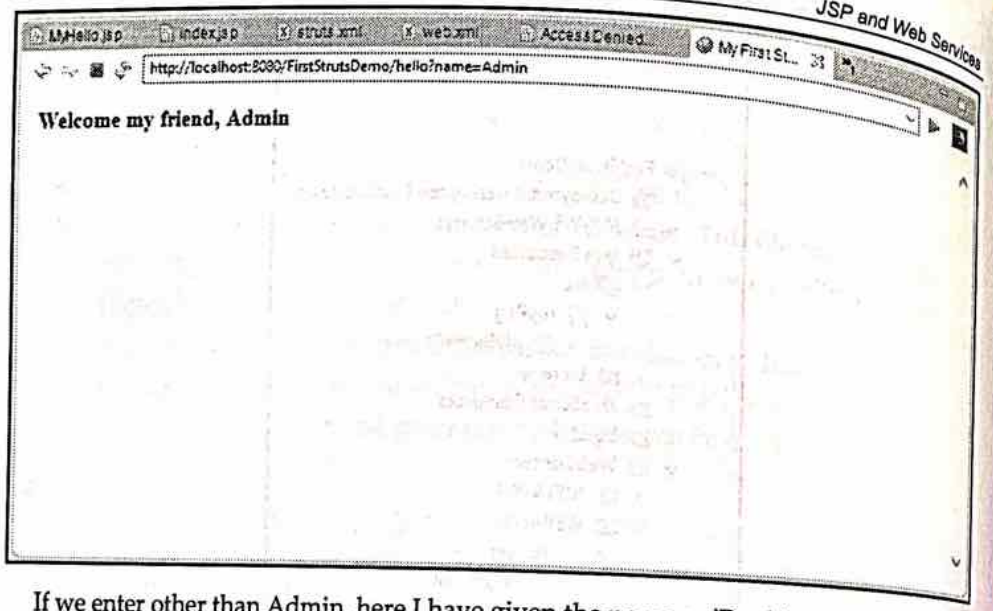
For understanding the location of above files I am giving the following screen-shot. Note that these files must be created in respective folders only -



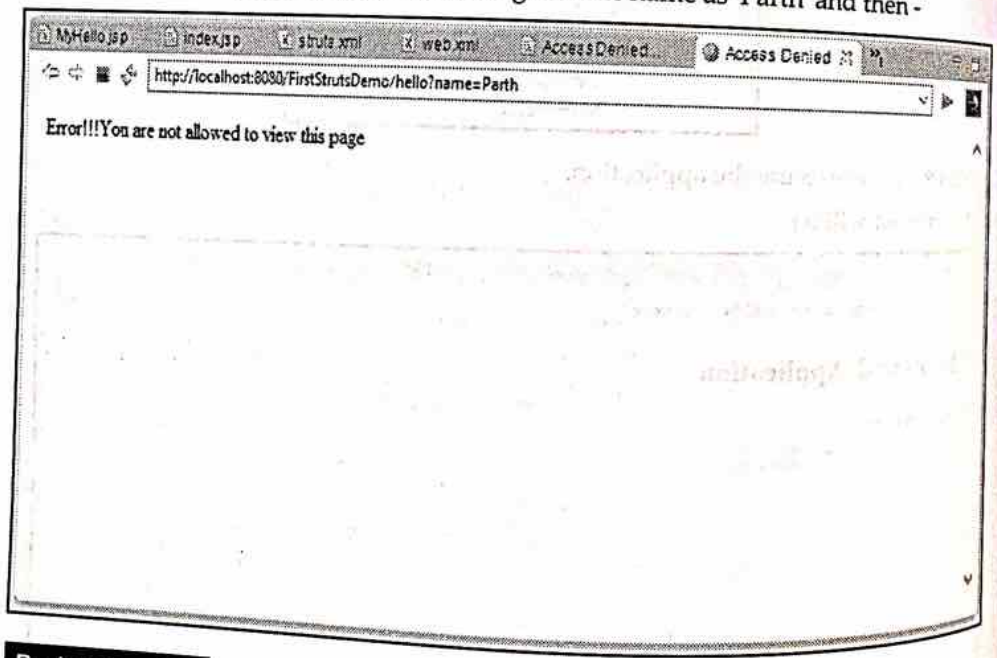
Step 9 : Now execute the application.

The output will be





If we enter other than Admin, here I have given the name as 'Parth' and then -



Review Question

1. *Identify and explain data tags from struts2 which are used to manipulate data displayed on any web application page.*

SPPU : Dec.-18, May-19, Marks 6

4.16 Interceptors

SPPU : May-18, Marks 6

- Interceptor is an object which intercepts an action dynamically. It executed before and after the action execution.
- In Struts 2, interceptor is used to perform operations such as validation, exception handling, internationalization, displaying intermediate result etc.
- Following is a list of Inceptors that are preconfigured and can be ready to use.

1) **alias :** It converts similar parameters that have different names between requests.
2) **chain :** If it is used with chain result type, it makes the properties of previous action available in the current action.
3) **checkbox :** It is used to handle the check boxes in the form. By this, we can detect the unchecked checkboxes.
4) **cookie :** It adds a cookie to the current action.
5) **conversionError :** It adds conversion errors to the action's field errors.
6) **createSession :** It creates and HttpSession object if it doesn't exists.
7) **clearSession :** It unbinds the HttpSession object.
8) **debugging :** It provides support of debugging.
9) **execAndWait :** It sends an intermediate waiting page for the result.
10) **exception :** It maps exception to a result.
11) **fileUpload :** It provides support to file upload in struts 2.
12) **i18n :** It provides support to internationalization and localization.
13) **store :** It stores and retrieves action messages, action errors or field errors for action that implements ValidationAware interface.
14) **params :** It populates the action properties with the request parameters.
15) **scope :** It is used to store the action state in the session or application scope.
16) **timer :** It outputs the time needed to execute an action.
17) **validation :** It provides support to input validation.

- We can create our own Inceptor object. This custom based inceptor can be plugged in struts.
- Framework creates an object of **ActionInvocation** that encapsulates the action and all the interceptors configured for that action.

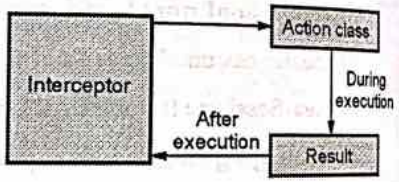
- Each interceptors are called before the action gets called.
- Once the action is called and result is generated, each interceptors are again called in reverse order to perform post processing work.
- Interceptors can alter the workflow of action. It may prevent the execution of action.

Advantages of Interceptors

1. If we want pre-processing services and post-processing services for an Action then these services can be provided by interceptor.
2. It provides pluggable functionalities for validation, exception handling and logging. We do not need to redeploy the application.
3. Interceptors can be coupled with request and response cycle.

How to Create Custom Intercept ?

- For creating the custom intercept we need interface Interceptor This interface contains a method **intercept()** which ultimately calls the **invoke()** method.



- The interceptor can be executed before calling the action, during execution of action and after execution of action.
- Let us create an interceptor in struts framework.

Step 1 : Start Eclipse. From File Menu select **File -> New->Dynamic Web Project.** Type the name of the project.

Just click **next** button with default options and check the **Generate Web.xml deployment descriptor** option. This will create a dynamic web project in Eclipse

Step 2 : Now copy the required Jar files present in the **c:\struts-2.2.34\lib** folder to our project's WEB-INF\lib folder

Step 3 : Inside the **Java Resources-> src** folder create a package(I have created **myPkg** package). Inside that package create a Java class for inceptor named **MyInterceptor** as follows -

MyInterceptor.Java

```
package myPkg;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.AbstractInterceptor;
public class MyInterceptor extends AbstractInterceptor{
  public String intercept(ActionInvocation invocation)throws Exception {
    System.out.println("Before Execution");
    String result = invocation.invoke();
    System.out.println("After Execution");
    return result;
  }
}
```

Remember these interceptors will shows the result in the console automatically

Step 4 : Now create the Java class as action class in which execute() method is defined. Again this class is created inside the user defined package of src folder. The code for this is as follows -

Test.Java

```
package myPkg;
import com.opensymphony.xwork2.ActionSupport;
public class Test extends ActionSupport{
  private String name;
  public String execute() throws Exception {
    System.out.println("Inside execute() method!!!");
    return "success";
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
}
```

Step 5 : Now we will create a **view** page. I have named it as **MyHello.jsp** in the **WebContent** folder in your eclipse project. To do this, right click on the **WebContent** folder in the project explorer and select **New >JSP File.**

MyHello.jsp

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>
<html>
  <head>
```

```
    <title>My Interceptor Demo Application</title>
  </head>
  <body>
    <h3>Welcome my friend, <s:property value = "name"/></h3>
  </body>
</html>
```

Step 6 : Now create the main page as **index.jsp** in the **WebContent** folder.

Index.jsp

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
  pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Struts Inceptor Demo Application</title>
  </head>
  <body>
    <h2>My Struts2 Application</h2>
    <form action = "hello">
      <label for = "name">Enter your name</label><br/>
      <input type = "text" name = "name"/>
      <input type = "submit" value = "Submit"/>
    </form>
  </body>
</html>
```

Step 7 : Create the configuration file struts.xml as follows-

struts.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
 <constant name = "struts.devMode" value = "true" />
 <package name="default" extends="struts-default">
 <interceptors>
    <interceptor name = "myinterceptor"
```



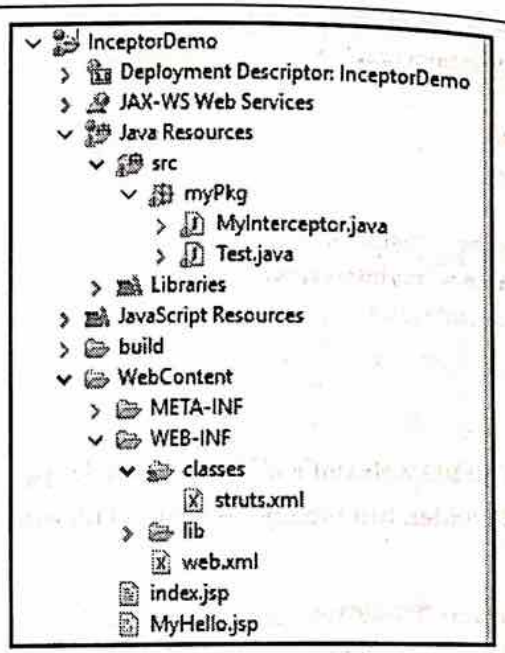
```
    class = "myPkg.MyInterceptor" />
  </interceptors>
  <action name = "hello"
  class = "myPkg.Test"
  method = "execute">
    <interceptor-ref name = "params"/>
    <interceptor-ref name = "myinterceptor" />
    <result name = "success">/MyHello.jsp</result>
  </action>
</package>
</struts>
```

Step 8 : The configuration file **web.xml** will invoke the **index.jsp** file. It is present inside WEB-INF folder. Just modify the web.xml file as follows -

web.xml

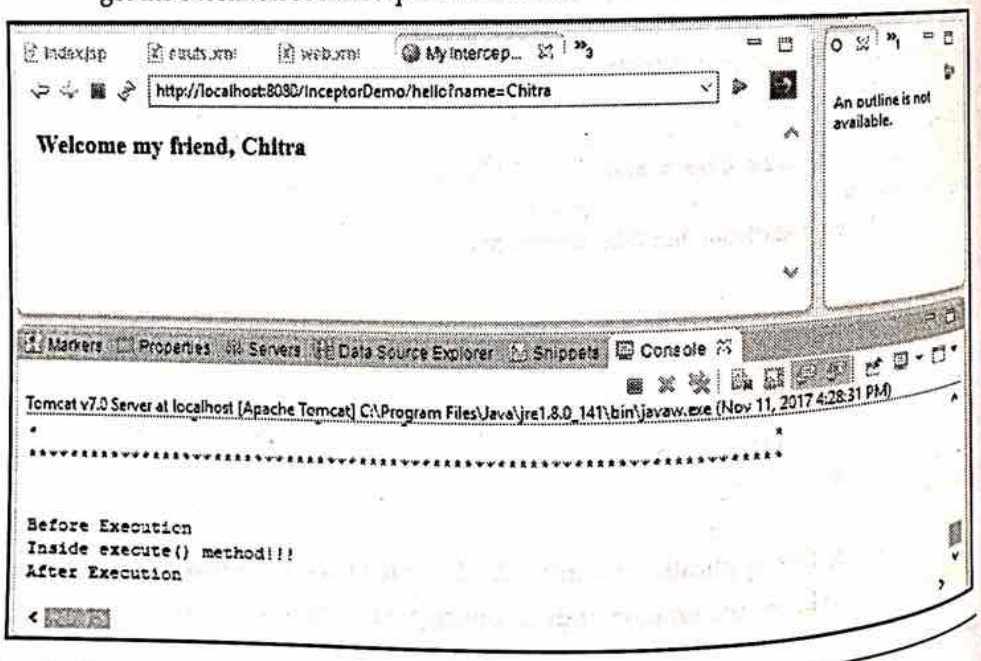
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>InceptorDemo</display-name>
 <welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>
 <filter>
   <filter-name>struts2</filter-name>
   <filter-class>
    org.apache.struts2.dispatcher.FilterDispatcher
   </filter-class>
 </filter>

 <filter-mapping>
   <filter-name>struts2</filter-name>
   <url-pattern>/*</url-pattern>
 </filter-mapping>
</web-app>
```

While creating this application in struts 2, please refer the following screen-shot of Project Explorer to know the location of the above created files.



Step 9 : Export the WAR file and start Tomcat server. Run the program on Tomcat server. The output will be obtained as follows -

Enter some name inside the text box and click the Submit button. You will get the execution of interceptor as follows :



Review Question

1. *How to use interceptors in Struts 2 ? List and describe important interceptors provided by Struts 2 framework.* **SPPU : May-18, Marks 6**

4.17 Result Types

- The result type is used to change the behaviour of view page to render the contents of action class.
- After executing the Action class, the target view page will be executed based on configuration in struts.xml.
- The Struts2 framework provides set of Result types like; chaining, dispatcher, redirect action, velocity and freemarket etc., that has to be configure in struts.xml for different kind of result.
- The <result> tag is used in **struts.xml** to render the view page. This result tag is provided with predefined result types such as dispatcher, redirect and so on. For example

```
<result name = "success" type="redirect">
```

- The **dispatcher** result type is the default type, and is used if no other result type is specified. It's used to forward to a servlet, JSP, HTML page, and so on, on the server. It can be written as

```
<result name="success" type="dispatcher">
<param name="location">/Success.jsp</param>
</result>
```

Let us understand how to use result type with demonstration.

How to use result type ?

In this application we will use result type for redirecting the view page to a new JSP page. Just follow the simple steps given below -

Step 1 : Create the files MyBigHello.java, index.jsp, MyHello.jsp and web.xml as it is discussed in section 4.16.

Step 2 : The struts.xml file is little bit changed. It contains the **<result ... type>** parameter. Following file tells us that on success redirect to a new page. This new page is **NewLocation.jsp**

struts.xml

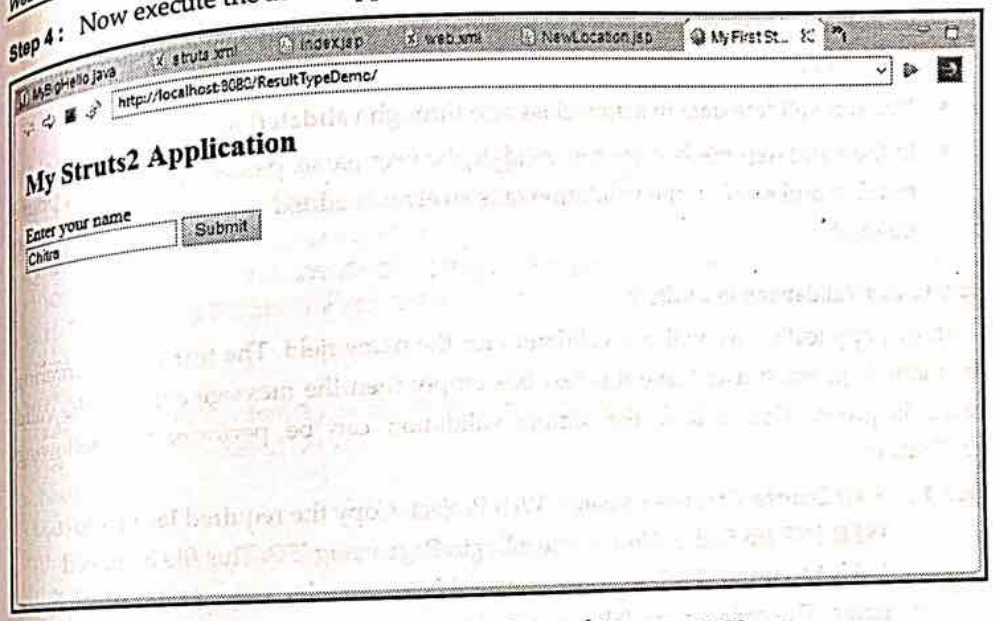
```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
  <constant name = "struts.devMode" value = "false" />

 <package name="default" extends="struts-default">
   <action name = "hello"
      class = "myPkg.MyBigHello"
      method = "execute">
      <result name = "success" type="redirect">
      <param name="location">
      /NewLocation.jsp
      </param>
      </result>
   </action>
      <action name = "index">
      <result >/index.jsp</result>
   </action>
  </package>
</struts>
```

Step 3 : The **NewLocation.jsp** file will be

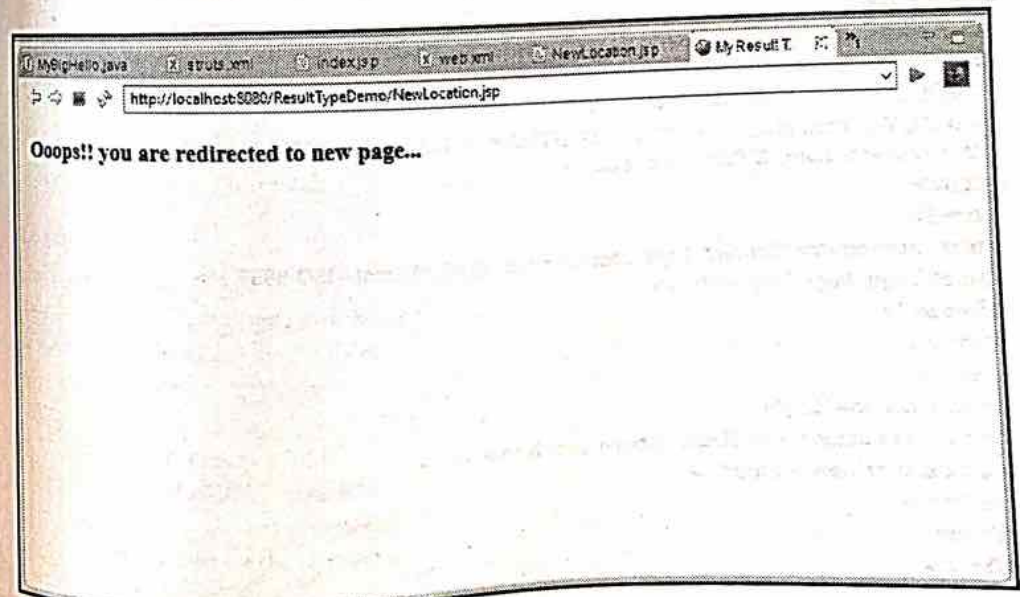
NewLocation.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>My Result Type Struts Demo Application</title>
  </head>
  <body>
    <h3>Ooops!! you are redirected to new page...</h3>
  </body>
</html>
```


Step 4 : Now execute the above application and the output can be obtained as follows -



Click the submit button, and you will be redirected to a new page



4.18 Validations

- In struts, validation framework help to perform validation before the action method is executed.
- You can validate data in action class also through **validate()** method.
- In the **validate()** method we can validate the user name, password,age,email, phone number and so on. If the validation fails an error is added using the **addFieldError()** method.

How to use Validations in Struts ?

In this application we will use validation for the name field. The text-box for entering the name is given, if user leave this text box empty then the message entering the valid name is given. This is how the simple validation can be performed in following application.

Step 1 : Start Eclipse. Create a Dynamic Web Project. Copy the required Jar files to **WEB-INF/lib** folder. Now create a Login Page using JSP. This file is created in **WebContent** . In this page one textfields is created to enter valid user name. The code for this JSP script is as follows-

LoginPg.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib uri="/struts-tags" prefix="s" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login Page Demo</title>
<s:head />
</head>
<body>
<s:form action="Login">
<s:textfield name="userName" label="User Name" />
<s:submit value="Submit" />
</s:form>
</body>
</html>
```


Step 2 : Create one more JSP document under **WebContent** folder named **Welcome.jsp** . This JSP file is for displaying welcome message for authentic user. The code is

Welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>login Page Demo</title>
</head>
<body>
<h1>Welcome, User <s:property value="userName"/> is authentic user!!!</h1>
</body>
</html>
```

Step 3 : Now we will write the Java File for validating the user name. Two validations are used -

i) the user name must not be blank and ii) the user name must be **root**. For validation purpose we will write one function named **validate()** in this Java program. This method is written in action class, hence this program will be created within package name **myPkg** under the **src** directory.

LoginProg.Java

```
package myPkg;
import com.opensymphony.xwork2.ActionSupport;
public class LoginProg extends ActionSupport {
    private String userName;

    public String execute() {
      return SUCCESS;
    }
    public String getUserName() {
      return userName;
    }

    public void setUserName(String userName) {
```

```
        this.userName = userName;
    }
    public void validate() {
        if (getUserName().length() == 0) {
            addFieldError("userName", "UserName.required");
        } else if (!getUserName().equals("root")) {
            addFieldError("userName", "Invalid User");
        }
    }
}
```

Step 4 : Now we will write the configuration files. The struts.xml file under the WEB-INF/classes folder will be -

struts.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name="struts.custom.i18n.resources" value="LoginProg" />
 <package name="default" extends="struts-default" namespace="/">
   <action name = "Login"
     class = "myPkg.LoginProg"
     method = "execute">
     <result name = "success">/Welcome.jsp</result>
     <result name = "input">/LoginPg.jsp</result>
   </action>
 </package>
</struts>
```

Step 5 : The web.xmlfile present inside the WEB-INF folder is modified as -

web.xml

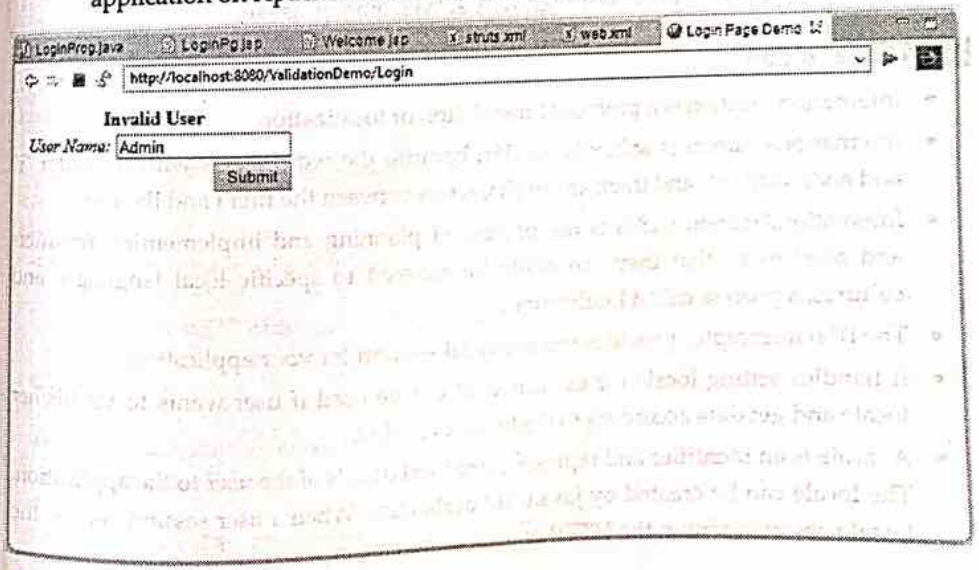
```
<?xml version = "1.0" encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns = "http://java.sun.com/xml/ns/javaee"
 xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id = "WebApp_ID" version = "3.0">
 <display-name>Struts 2</display-name>
 <welcome-file-list>
```

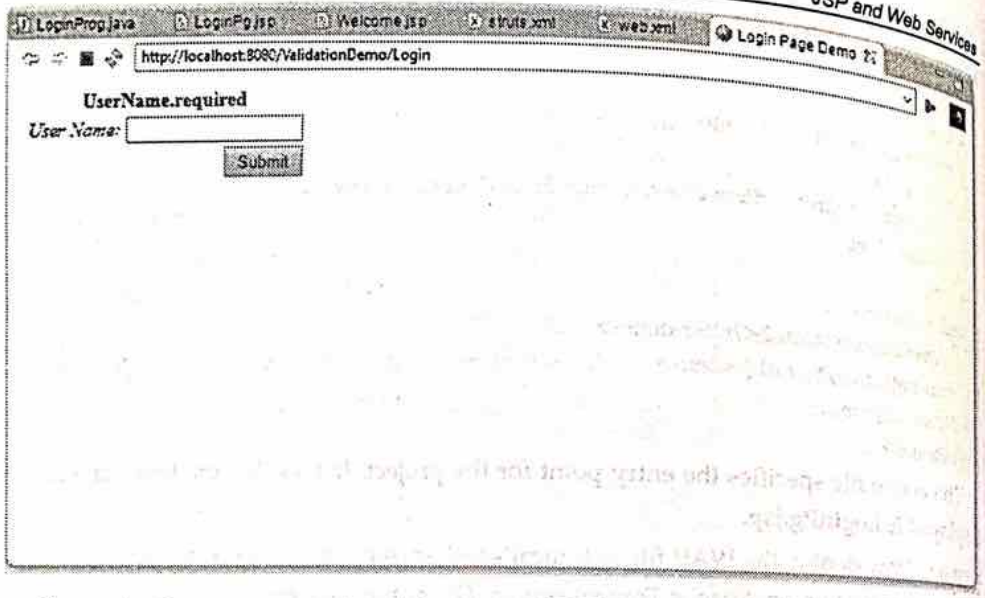


```
  <welcome-file>LoginPg.jsp</welcome-file>
 </welcome-file-list>
 <filter>
  <filter-name>struts2</filter-name>
  <filter-class>
   org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
  </filter-class>
 </filter>
 <filter-mapping>
  <filter-name>struts2</filter-name>
  <url-pattern>/*</url-pattern>
 </filter-mapping>
</web-app>
```

The above file specifies the entry point for the project. It says that the first page to be displayed is **LoginPg.jsp.**

Step 6 : Now deploy the WAR file to tomcat's webapps directory and run the above application on Apache Tomcat server. The output will be





If we enter the user name as 'root' then we get Welcome message.

4.19 Localization

- Internationalization is a process of translation or localization.
- Internationalization is abbreviated i18n because the word starts with the letter "i" and ends with "n", and there are 18 characters between the first i and the last n.
- Internationalization (i18n) is the process of planning and implementing products and services so that they can easily be adapted to specific local languages and cultures, a process called localization.
- The i18n interceptor provides multi-lingual support for your application.
- It handles setting locale for the action. It can be used if user wants to set his/her locale and get data according to the locale provided.
- A locale is an identifier and represents the local details of the user to the application. The locale can be created by **java.util.Locale** class. When a user session begins, the Locale object is sent in the HTTP request as parameter.

Concept of Resource Bundle

Struts2 makes use of resource bundle to provide multiple languages and locale options to the users. The resource bundle is a resource file in which key-value pairs are stored.

The name to these resource files is given in following naming format

Bundlename_language_country.properties

The **language_country** represents the country locale. For example English United States is represented as en_US, For Spanish it will be _es and so on. These properties file can be named as

global.properties : This is a default file for the English language.

global_fr.properites : This file specifies the French locale.

global_cn.properties : It specifies the code in Chinese.

global_hi.properties : It specifies the code in Hindi

How to create application to support Localization ?

Step 1 : Create a Dynamic project. Copy all the necessary .jar files to WEB-INF/lib folder.

Create initial JSP page inside the WebContent folder, name it as **index.jsp**. The code for it will be as follows -

Index.jsp

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
 pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <title>Student Form with Multilingual Support</title>
 </head>
 <body>
  <h1><s:text name = "global.heading"/></h1>

  <s:url id = "indexEN" namespace="/" action = "locale" >
   <s:param name = "request_locale" >en</s:param>
  </s:url>

  <s:url id = "indexDE" namespace="/" action = "locale" >
   <s:param name = "request_locale" >de</s:param>
  </s:url>
  <s:url id = "indexHI" namespace="/" action = "locale" >
   <s:param name = "request_locale" >hi</s:param>
  </s:url>

  <s:a href="%{indexEN}" >English</s:a>
  <s:a href="%{indexDE}" >German</s:a>
  <s:a href="%{indexHI}" >Hindi</s:a>
```

```
  <s:form action = "studentinfo" method = "post" namespace = "/">
    <s:textfield name = "name" key = "global.name" size = "20" />
    <s:submit name = "submit" key = "global.submit" />
  </s:form>
 </body>
</html>
```

Step 2 : Now create another JSP page to handle the successful submission of data. This page will display the message that you are authenticated user. The JSP file is created inside the **WebContent** folder.

The code is very simple and it is

Welcome.jsp

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
   pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
   <title>Success</title>
 </head>
 <body>
   <s:property value = "getText('global.success')" />
 </body>
</html>
```

In above code have used **getText** method to access the messages. To display i18n text, use a call to getText in the property tag.

Step 3 : Now create the Java Class files inside the src folder. First create the package with some suitable name under **Java Resources/src**. Here we will create two Java Class files - first Java class file will take care of Locale and display the **index.jsp** with different languages and second Java class file will handle the form submission in which execute, getXX and setXX methods are defined. The code for these Java files is as follows

StudLocale.java

```
package myPkg;
import com.opensymphony.xwork2.ActionSupport;
public class StudLocale extends ActionSupport {
    public String execute() {
      return SUCCESS;
    }
}
```


TestAction.java

```
package myPkg;
import com.opensymphony.xwork2.ActionSupport;
public class TestAction extends ActionSupport {
    private String name;
    public String execute() {
      return SUCCESS;
    }
    public String getName() {
      return name;
    }
    public void setName(String name) {
      this.name = name;
    }
}
```

Step 4 : Now we can create **global.properties** files. These files can be created for different languages so that the web page can be displayed in those corresponding languages. Here I will write the global file for displaying the message English only. This file is created inside WEB-INF folder inside which the **classes** folder is created. Save **global.properties** file inside this classes folder.

global.properties

```
global.name = Name
global.submit = Submit
global.heading = Select desired Locale
global.success = You are authenticated user!!!
```

In this file,we will display the name of the fields and their corresponding values. For instance for **global.success** the message "You are authenticated user!!!" message must be displayed. For **global.heading** the "Select desired Locale" must be displayed.

Step 5 : Now create the configuration files. First create the **struts.xml** file inside **WEB-INF\classes** folder. The code for this file would be as follows -

struts.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
 <constant name = "struts.devMode" value = "false" />
 <constant name = "struts.custom.i18n.resources" value = "global" />
 <package name = "localizationdemo" extends = "struts-default" namespace="/">
```

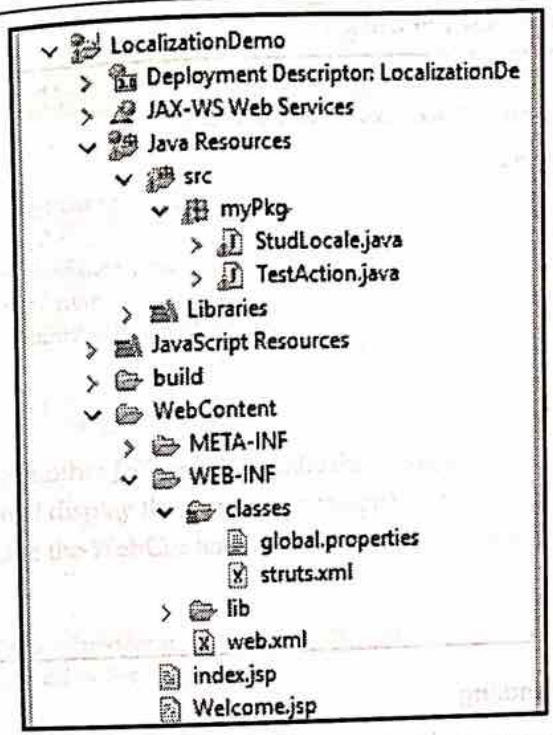
```
  <action name = "studentinfo"
    class = "myPkg.TestAction"
    method = "execute">
    <result name = "input">/index.jsp</result>
    <result name = "success">/Welcome.jsp</result>
  </action>
  <action name = "locale"
    class = "myPkg.StudLocale"
    method = "execute">
    <result name = "success">/index.jsp</result>
  </action>
 </package>
</struts>
```

Step 6 : Modify the web.xml file present in WEB-INF folder. The code is as follows -

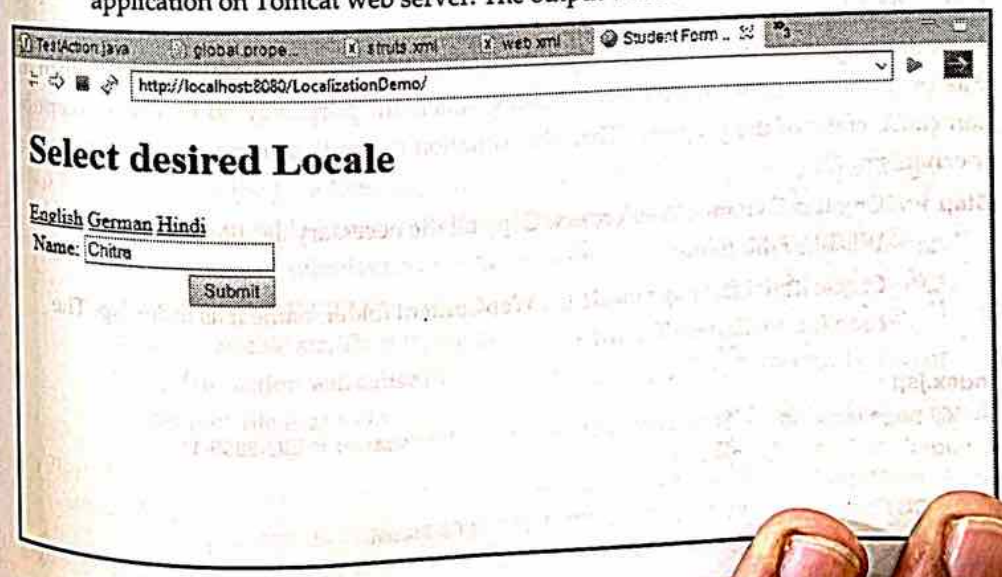
web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
 <display-name>LocalizationDemo</display-name>
 <welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
 </welcome-file-list>
  <filter>
   <filter-name>struts2</filter-name>
   <filter-class>
    org.apache.struts2.dispatcher.FilterDispatcher
   </filter-class>
  </filter>
  <filter-mapping>
   <filter-name>struts2</filter-name>
   <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

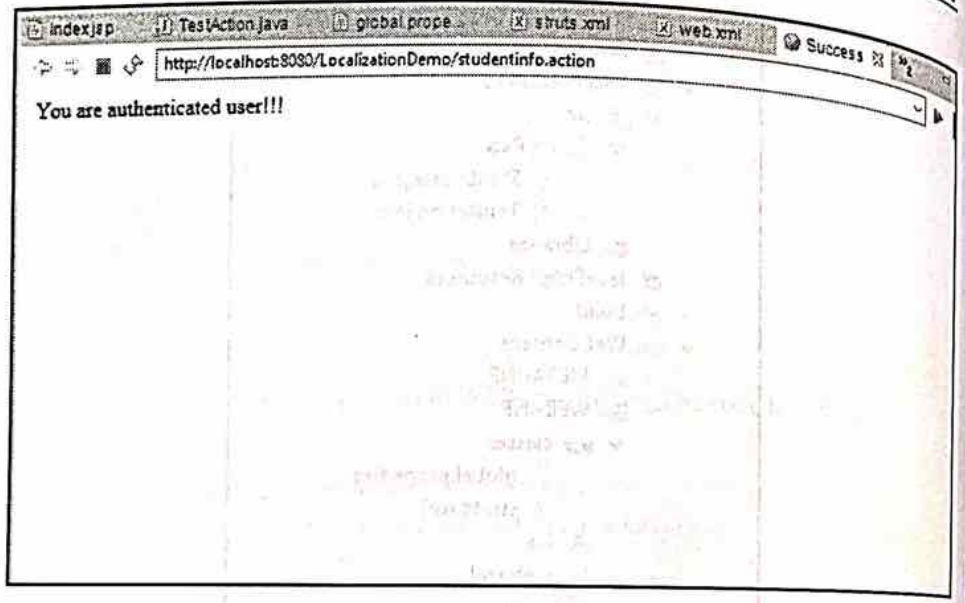
Here is the screen-shot for Project Explorer to understand the parent folders of above created files -



Step 7 : Then right click on the project name and click **Export-> WAR** file for exporting the war file in Tomcat directory. Then execute the above application on Tomcat web server. The output will be



Click Submit button and you will get



4.20 Exception Handling

When a program crashes due to some run time error, then this is an undesirable situation. This situation is called exception. Struts provide an efficient way to handle uncaught exception. When such situation occurs, struts redirect to error message display page elegantly.

Following demonstrative application illustrates the exception handling mechanism. In this program we have created error causing statement purposely so that this exception can cause crash of the program. Then this situation is handled using exception handling mechanism.

Step 1 : Create a Dynamic Web project. Copy all the necessary .jar files to WEB-INF/lib folder.

Create initial JSP page inside the **WebContent** folder, name it as index.jsp. The code for it will be as follows -

Index.jsp

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
 pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```



```
<html>
 <head>
  <title>My First Struts Demo Application</title>
 </head>
 <body>
  <h2>My Struts2 Application</h2>
  <form action = "hello">
   <label for = "name">Enter your name</label><br/>
   <input type = "text" name = "name"/>
   <input type = "submit" value = "Submit"/>
  </form>
 </body>
</html>
```

Step 2 : Now create another JSP page to handle the successful submission of data. This page will display the welcome message by user name. The JSP file is created inside the **WebContent** folder. The code is very simple and it is -

MyHello.jsp

```
<%@ page contentType = "text/html; charset = UTF-8" %>
<%@ taglib prefix = "s" uri = "/struts-tags" %>

<html>
 <head>
  <title>Exception Demo Application</title>
 </head>

 <body>
  <h3>Welcome my friend, <s:property value = "name"/></h3>
 </body>
</html>
```

Step 3 : Now create the Java Class files inside the **src** folder. First create the package with some suitable name under **Java Resources/src**. This Java class file will handle the form submission in which execute, getXX and setXX methods are defined. Note that purpose, inside the execute method I have created a null string in variable **str**. Then trying to find the Uppercase string of this null string. This action will cause the **NullPointerAssignment** problem. The code for this Java file is as follows -

MyBigHello.java

```
package myPkg;
import com.opensymphony.xwork2.ActionSupport;
public class MyBigHello extends ActionSupport{
```

```
    private String name;

    public String execute() {
        String str=null;
        str=str.toUpperCase();
        return SUCCESS;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

These statements are causing Null Pointer

Step 4 : Now create the configuration files. First create the **struts.xml** file inside **WEB-INF\classes** folder. The code for this file would be as follows -

struts.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name = "struts.devMode" value = "false" />
<package name="default" extends="struts-default">
   <action name = "hello"
     class = "myPkg.MyBigHello"
     method = "execute">
     <result name = "success">/MyHello.jsp</result>
   </action>
  </package>
</struts>
```

Step 5 : Another configuration file **web.xml** is modified which is present inside the WEB-INF folder.

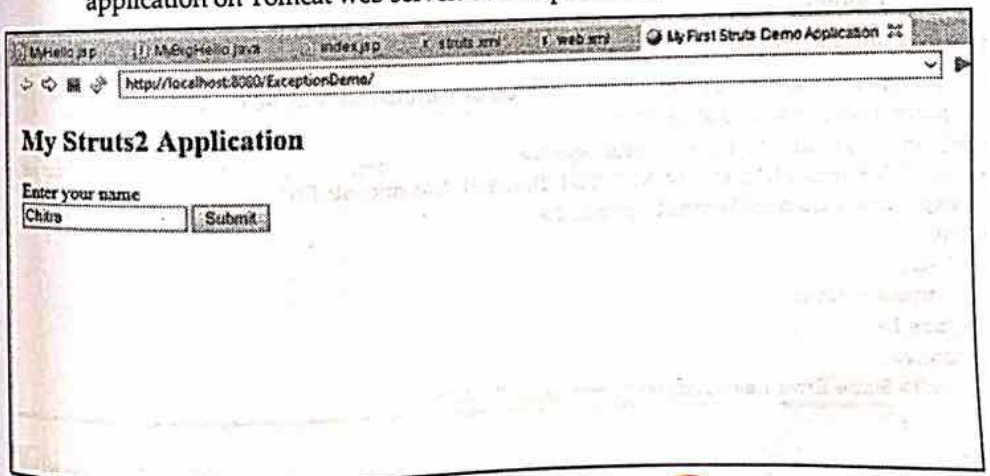
web.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns = "http://java.sun.com/xml/ns/javaee"
 xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id = "WebApp_ID" version = "3.0">
```

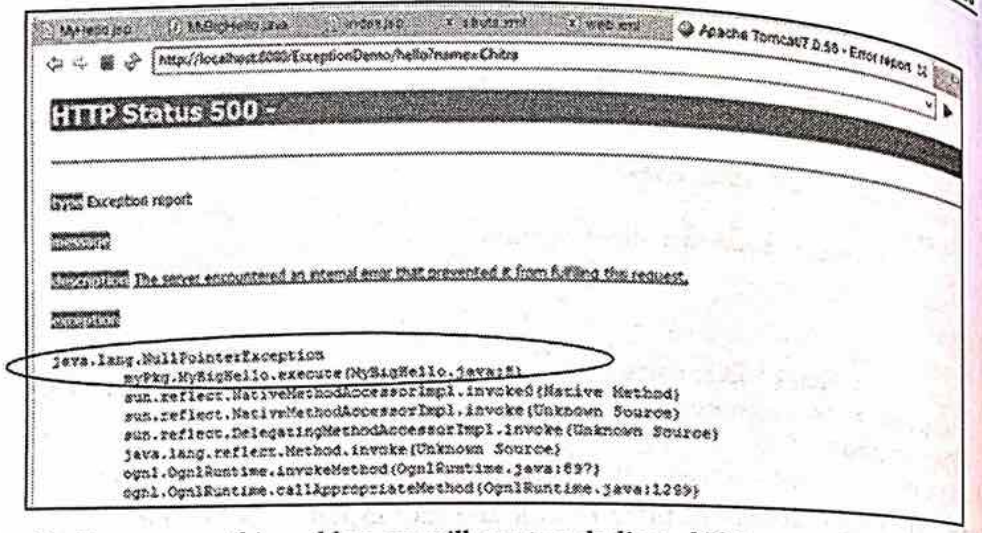


```
  <display-name>Struts 2</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
    <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.FilterDispatcher
    </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

Step 6 : Then right click on the project name and click Export-> WAR file for exporting the war file in Tomcat directory. Then execute the above application on Tomcat web server. The output will be



Click Submit button and following exception will be raised -



Step 7 : To overcome this problem we will create a dedicated JSP page to handle exception. I have created a JSP page named **ErrorHandle.jsp** page as follows -

ErrorHandle.jsp

```
<%@ page language = "java" contentType = "text/html; charset = ISO-8859-1"
   pageEncoding = "ISO-8859-1"%>
<%@ taglib prefix = "s" uri = "/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title></title>
  </head>
  <body>
    <h3>Some Error has occurred in your page!!!</h3>
  </body>
</html>
```

Step 8 : Now in order to display the above created JSP page on exception we need to modify our **struts.xml** file. The code will be

struts.xml

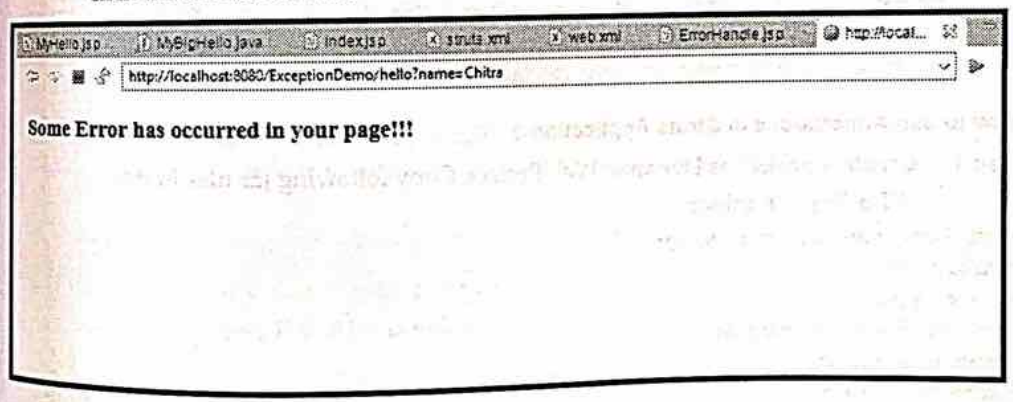
```
<?xml version = "1.0" encoding = "UTF-8"?>
<!DOCTYPE struts PUBLIC
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
 "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
```



```
  <constant name = "struts.devMode" value = "false" />
<package name="default" extends="struts-default">
  <action name = "hello"
   class = "myPkg.MyBigHello"
   method = "execute">
   <exception-mapping exception = "java.lang.NullPointerException"
   result = "error" />
   <result name = "success">/MyHello.jsp</result>
   <result name = "error">/ErrorHandle.jsp</result>
  </action>
 </package>
</struts>
```

In above code, in order to handle the exception the **<exception-mapping>** tag is used. This tag is assigned with the value by the Java exception name **NullPointerException.** Since exceptions in Java are defined in package **java.lang** we denote it as **"java.lang.NullPointerException"**

Step 9 : Now run the above code by exporting WAR file and then on Tomcat server, and following output can be obtained when we enter some user name and click the Submit button.



4.21 Annotations

- Java annotation is a tag used to represent the metadata. The metadata is a data about classes, methods or fields.
- Annotations in Java are used to provide some addition information. For instance - If user is entering his age in the text field, then that age must not be negative or it should not be blank - Such type of information can be provided to the user using annotations.

- A Java annotation in its shortest form looks like this :

```
@Entity
```

- The @ character signals to the compiler that this is an annotation. The name following the @ character is the name of the annotation.
- Struts provides two forms of configuration.
  - o The traditional way is to use the **struts.xml** file for all the configurations.
  - o The other way of configuring Struts is by using the Java 5 Annotations feature.
- Using annotation we do not need **struts.xml** for configuration. Thus we can achieve zero configuration using annotation.
- Commonly, we can use 3 annotations :
  1) **@Action** annotation is used to mark the action class.
  2) **@Results** annotation is used to define multiple results for one action.
  3) **@Result** annotation is used to display single result.
- To support the annotations we require following files in the Java program to be imported as -

```
import org.apache.struts2.convention.annotation.Action;
import org.apache.struts2.convention.annotation.Result;
import org.apache.struts2.convention.annotation.Results;
import com.opensymphony.xwork2.validator.annotations.*;
```

How to use Annotations in Struts Application ?

Step 1 : Create a project as Dynamic Web Project. Copy following Jar files in the WEB-INF\lib folder.

```
struts2-convention-plugin-x.y.z.jar
asm-x.y.jar
antlr-x.y.z.jar
commons-fileupload-x.y.z.jar
commons-io-x.y.z.jar
commons-lang-x.y.jar
commons-logging-x.y.z.jar
commons-logging-api-x.y.jar
freemarker-x.y.z.jar
javassist-.xy.z.GA
ognl-x.y.z.jar
struts2-core-x.y.z.jar
xwork-core.x.y.z.jar
```


Step 2 : Create the same project as discussed in section 4.18 for validation. The **LoginPg.jsp, Welcome.jsp** will be as it is. The only change will be in the Java file. In the Java file we introduce annotations. The code will be as follows -

LoginProg.java

```
package myPkg;
import com.opensymphony.xwork2.ActionSupport;
import org.apache.struts2.convention.annotation.Action;
import org.apache.struts2.convention.annotation.Result;
import org.apache.struts2.convention.annotation.Results;
import com.opensymphony.xwork2.validator.annotations.*;
@Results({
    @Result(name = "success", location = "/Welcome.jsp"),
    @Result(name = "input", location = "/LoginPg.jsp")
  })
public class LoginProg extends ActionSupport {
    private String userName;
    @Action(value = "/Login")
    public String execute() {
      return SUCCESS;
    }
    @RequiredFieldValidator( message = "The user Name is required" )
    public String getUserName() {
      return userName;
    }
    public void setUserName(String userName) {
      this.userName = userName;
    }
}
```

Step 3 : Now we just do not require struts.xml file. The web.xml is the only configuration file required. It will be as follow -

web.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
 xmlns = "http://java.sun.com/xml/ns/javaee"
 xmlns:web = "http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 xsi:schemaLocation = "http://java.sun.com/xml/ns/javaee
 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 id = "WebApp_ID" version = "3.0">

 <display-name>Struts 2</display-name>

 <welcome-file-list>
  <welcome-file>LoginPg.jsp</welcome-file>
```

```
  </welcome-file-list>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
      org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
    </filter-class>
    <init-param>
      <param-name>struts.devMode</param-name>
      <param-value>true</param-value>
    </init-param>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

Step 4 : Now Export-War file and execute the above application. If the name field is kept empty and Submit button is clicked then "The user Name is required" message will be displayed.

4.22 Multiple Choice Questions

Q.1 JSP is transformed into _____.

a HTML
b PHP
c Byte Code
d Java Servlet

Q.2 Which programming language does JSP support ?

a JavaScript only
b VB Script only
c Java Only
d All of the above

Q.3 Which of the scripting of JSP not putting content into service method of the converted servlet ?

a Declarations
b Scriptlets
c Expressions
d None of these

Q.4 The difference between Servlets and JSP is the _____.

a translation
b compilation
c syntax
d both a and b

Q.5 Which is not a directive ?

a Include
b Page

Q.6 Which of the following are the valid scopes in JSP ?

a request, page, session, application
b request, page, session, global
c response, page, session, applicati
d request, page, context, application

Q.7 "request" is an instance of which one of the following classes :

a Request
b HttpRequest
c HttpServletRequest
d ServletRequest

Q.8 Which http method send by browser that asks the server to get data _____.

a get
b post
c option
d put

Q.9 Which attribute specifies a JSP page that should process any exceptions thrown caught in the current page ?

a ErrorPage attribute
b IsErrorPage attribute
c thisErrorPage attribute
d none of these

Q.10 Which technology do we use to mix business logic with presentation logic ?

a JSP
b servlet
c both a and b
d none of these

Q.11 Which tag is used to execute java source code in JSP ?

a declaration tag
b scriptlet tag
c expression tag
d both b and c

Q.12 In JSP page directive which attribute defines the MIME(Multipurpose Inter Extension) type of HTTP response ?

a import
b info
c extends
d ContentType

Q.13 Choose the correct alternative

a Servlets are built on JSP semantics and all servlets are compiled to JS for runtime usage.
b JSP and servlets are unrelated technologies.
c Servlets and JSP are compet plogies fo g web reque Servlets are being supersed which i n combi The two technologies are r nd all JS bil
d JSPs are built on servlet s runtime usage.

Q.14 What is the initial contact point for handling a web request in a Page-Centric architecture ?

a A JSP page
b A JavaBean.
c A servlet
d A session manager.

Q.15 What is the key difference between using a <jsp:forward> and HttpServletResponse.sendRedirect() ?

a Forward executes on the server while sendRedirect() executes on the client.
b Forward executes on the client while sendRedirect() executes on the server.
c The two methods perform identically.
d None of these

Q.16 In Struts2 Action classes are the components of _____.

a view
b controller
c model
d deployment

Q.17 In Struts2, What does validate() method of ActionForm returns ?

a ActionErrors
b ActionForward
c ActionMapping
d ActionError

Q.18 Which of the following methods is overridden by Action class ?

a run()
b destroy()
c execute()
d service()

Answers Keys for Multiple Choice Questions :

Q.1	d	Q.2	c	Q.3	c	Q.4	c	Q.5	c
Q.6	a	Q.7	c	Q.8	a	Q.9	a	Q.10	b
Q.11	b	Q.12	d	Q.13	d	Q.14	a	Q.15	a
Q.16	b	Q.17	b	Q.18	c				

□□□

UNIT- V

5 Server Side Scripting Languages

Syllabus

PHP: Introduction to PHP, uses of PHP, general syntactic characteristics, Primitives, operations and expressions, output, control statements, arrays, functions, pattern matching, form handling, files, cookies, session tracking, using MySQL with PHP, WAP and WML. ***Introduction to ASP.NET :*** *Overview of the .NET Framework, Overview of C#, Introduction to ASP.NET, ASP.NET Controls, Web Services. Overview of Node JS.*

Contents

5.1 Introduction to PHP
5.2 General Syntactic Characteristics
5.3 Primitives ..... May-18,19, ... Marks 4
5.4 Type Conversion
5.5 Operations and Expressions
5.6 Output
5.7 Statements Control
5.8 Arrays ..... May-18,19, Dec.-18, ... Marks 6
5.9 Functions
5.10 Built in Functions
5.11 Form Handling
5.12 Pattern Matching
5.13 Files ..... Dec.-18, ... Marks 6
5.14 Cookies
5.15 Session Tracking ..... Dec.-19, ... Marks 8
5.16 Using MySQL with PHP ..... May-18, Dec.-19, ... Marks 8

5.17 WAP
5.18 WML
5.19 Introduction to ASP.NET
5.20 Overview of the .NET Framework
5.21 Overview of C#
5.22 Introduction to ASP.NET
5.23 ASP.NET Web Form Model
5.24 ASP.NET Controls
5.25 Web Services
5.26 Overview of Node JS ..... May-18, ... Marks 4
5.27 Multiple Choice Questions

Part - I : PHP

5.1 Introduction to PHP

5.1.1 Origins and Uses of PHP

- PHP was developed in 1994 by Apache group.
- PHP stands for PHP : Hypertext Preprocessor.
- PHP is a server-side scripting language. It is mainly used for form handling and database access.
- It is free to download and use.

5.1.2 Overview

- PHP is a server side scripting language embedded in XHTML. It is an alternative to CGI, ASP, ASP.NET and JSP.
- The extensions to the PHP files are .php, .php3 or .phtml.
- The php processor works in **two modes**. If the PHP processor finds XHTML tags in the PHP script then the code is simply copied to the output file. But when the PHP processor finds the PHP code in the script then that code is simply interpreted and the output is copied to the output file.
- If you click for view source on the web browser you can never see the PHP script because the output of PHP script is send directly to the browser but you can see the XHTML tags.
- PHP makes use of dynamic typing that means there is no need to declare variables in PHP. The type of variable gets set only when it is assigned with some value.
- PHP has large number of library functions which makes it flexible to develop the code in PHP.

5.1.3 Installation of PHP

- There are various methods of getting PHP installed on your machine.
- The PHP requires the Apache Web server to execute its code. The Apache web server is open source software and can be easily downloaded from the Internet. The site for getting the Apache web server is **http://httpd.apache.org/download.cgi**.
- Then PHP can be installed on your machine from the web site **http://www.php.net**.
- Another approach which is the most efficient way to install Apache, PHP, MYSQL on your machine is to use the packages like XAMPP (The X stands for any OS) or

WAMPP (the W stands for Windows OS). These packages support the Apache, PHP, MYSQL, PERL.

- I have installed the XAMPP package. The root directory can be accessed using the http://localhost/\

5.2 General Syntactic Characteristics

- PHP code can be embedded in the XHTML document. The code must be enclosed within **<?php** and **>**
- If the PHP script is stored in some another file and if it needs to be referred then **include** construct is used. For instance :

```
Include("myfile.inc")
```

- The variable names in PHP begin with the $ sign.
- Following are some reserved keywords that are used in PHP.

and	default	false	if	or	this
break	do	For	include	require	true
case	else	foreach	list	return	var
class	elseif	function	new	static	virtual
continue	extends	gloabal	not	switch	while
					xor

- The **comments** in PHP can be #,//, /* ...*/
- The PHP statements are terminated by semicolon.

Open some suitable text editor like Notepad and type the following code. Save your code by the extension .php.

It is expected that the PHP code must be stored in **htdocs** folder of Apache.

As I have installed **xampp,** I have got the directory c:\xampp\htdocs. I have created a folder named php-examples inside the htdocs and stored all my PHP documents in that folder.

Hence when I want to get the output of the PHP code I always give the URL

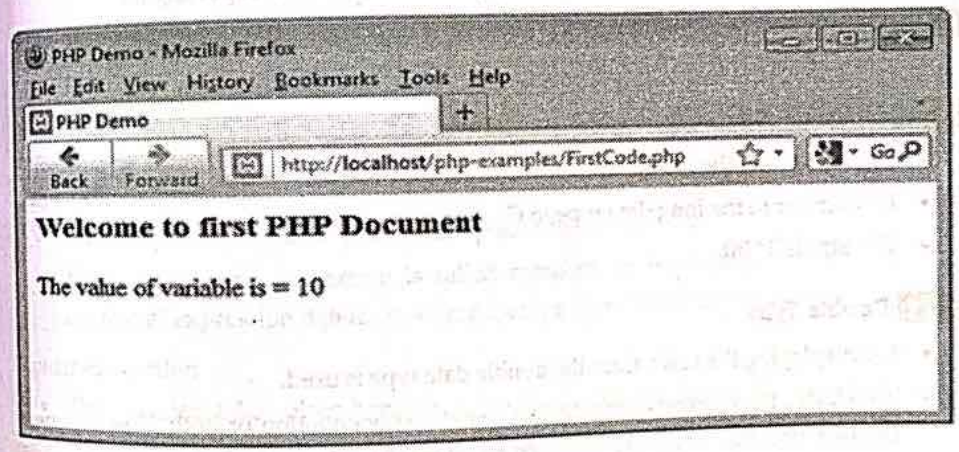
```
http://localhost/php-examples/programmName.php
The http://localhost refers to the path c:\xampp\htdocs
```


Following is the first example of PHP script

PHP Document[FirstCode.php]

```
<html>
<head>
 <title> PHP Demo </title>
</head>
<body>
 <?php
 $i=10;
 echo "<h3>Welcome to first PHP Document</h3>";
 echo "The value of variable is = $i";
 ?>
</body>
</html>
```

Output



5.2.1 Variables

- Variables are the entities that are used for storing the values.
- PHP is a dynamically typed language. That is PHP has no type declaration.
- The value can be assigned to the variable in following manner -

```
$variable_name=value
```

- If the value is not assigned to the variable then by default the value is NULL. The unsigned variables are called **unbound variable.**
- If the unbound variable is used in the expression then its NULL value is converted to the value 0.

- Following are some rules that must be followed while using the variables -
  1. The variable must start with letter or underscore _ , but it should not begin with a number.
  2. It consists of alphanumeric characters or underscore.
  3. There should not be space in the name of the variable.
  4. While assigning the values to the variable the variable must start with the $. For example

```
$marks=100;
```

- Using the function **IsSet** the value of the variable can be tested. That means if IsSet($marks) function returns **TRUE** then that means some value is assigned to the variable **marks.**
- If the unbound variable gets referenced then the error reporting can be done with the help of a function **error_reporting(7)**. The default error reporting level is 7.

5.3 Primitives

SPPU : May-18,19, Marks 4

5.3.1 Integer Type

- For displaying the integer value the Integer type is used.
- It is similar to the **long** data type in C.
- The size is 32 bit.

5.3.2 Double Type

- For displaying the real values the double data type is used.
- It includes the numbers with decimal point, exponentiation or both. The exponent can be represented by E or e followed by integer literal.
- It is not compulsory to have digits before and after the decimal point. For instance .123 or 123. is allowed in PHP.

5.3.3 String Type

- There is no character data type in PHP. If the character has to be represented then it is represented using the string type itself; but in this case the string is considered to be of length 1.
- The string literals can be defined using either single or double quotes.
- In single quotes the escape sequence or the values of the literals can not be recognized by PHP but in double quotes the escape sequences can be recognized.

For example -

```
'The total marks are= $marks'
```

will be typed as it is but

```
"The total marks are= $marks"
```

will display the value of $marks variable.

5.3.4 Boolean Type

- There are only two types of values that can be defined by the Boolean type and those are TRUE and FALSE.
- If Boolean values are used in context of integer type variable then TRUE will be interpreted as 1 and FALSE will be interpreted as 0.
- If Boolean values are used in context of double type then the FALSE will be interpreted as 0.0.

Review Question

1. *Classify data types of PHP and describe various data types in each type.* **SPPU : May-18,19, Marks 4**

5.4 Type Conversion

- PHP supports for both implicit and explicit type conversion.
- The implicit type conversion is called **coercion.** In implicit type conversion the context of expression determines the data type that is expected or required.

Implicit conversion

- The coercion takes place between integer and double types and between Boolean and other scalar type.
- The coercion is also between the numeric and string types. That means whenever a numeric value appears in string context then it is converted to string type, similarly when the string value appears in numeric context it is automatically converted to numeric type.
- When the double value is converted to integer the fractional part is eliminated and the value is not rounded.

Explicit conversion

- There are three ways by which the explicit conversion takes place.
  1. The syntax is

```
(datatype)$variable_name
```

For example -

```
(int)$marks
```

2. The syntax is -

```
Conversion_function($variable_name)
```

For example -

```
Intval($marks)
```

3. The syntax is -

```
settype($variable_name,"datatype")
```

For example -

```
settype($marks,"integer");
```

- The gettype function is useful to obtain the data type of the variable.

5.5 Operations and Expressions

5.5.1 Arithmetic Operators

- PHP supports the collection of arithmetic operators such as +, -, /, *, %, ++ and - with their usual meaning.
- While using the arithmetic operators if both the operands are integer then the result will be integer itself.
- If either of the two operands is double then the result will be double.
- PHP has large number of predefined functions. Some of these functions are enlisted in the following table -

Function	Purpose
floor	The largest integer less than or equal to the parameter is returned. For example floor(4.9) will return 4,
ceil	The smallest integer greater than or equal to the parameter is returned. For example ceil(4.9) will return 5,
round	Nearest integer is returned.
abs	Returns the absolute value of the parameter.
min	It returns the smaller element.
max	It returns the larger element.

5.5.2 Relational Operators

- There are eight relational operators used in PHP.
- These are <, >, <=, >=, !=, == has their usual meaning. These are six traditional operators.
- The operator === is used in PHP. It returns true if both operands that are using === have same type and have same value.
- The operator !== is opposite of ===.
- If one of the operand in the six operators is not same then the coercion will occur automatically.

5.5.3 Boolean Operators

- The Boolean operators are

Operator	Meaning
and &&&	The binary AND operation is performed.
or ||	The binary OR operation is performed.
xor	The XOR operation will be performed.

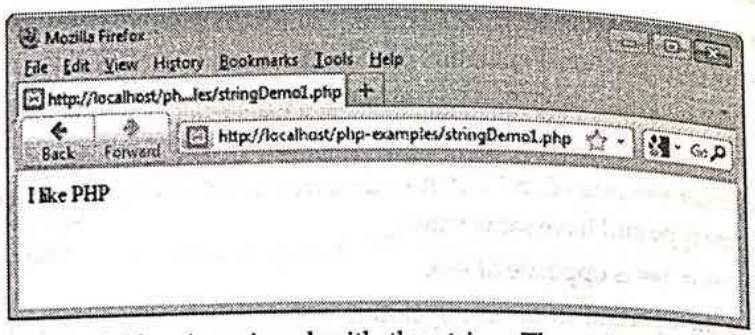
5.5.4 String Operators

- Concatenation is the only one operator used in string. It is denoted by dot.
- Strings are treated as the array of characters. The first position of the character is indexed as 0.
- The sample PHP script that stores the string in a variable is as given below -

PHPScript[stringDemo1.php]

```
<?php
$s="I like PHP";
echo $s;
?>
```

Output



Note that the variable **s** is assigned with the string. The string is given in a double quote. Then using the **echo** whatever string is stored in the variable **s** is displayed on the console.

5.6 Output

- The PHP is a server side scripting language and is used to submit the web page to the client browser. Hence it is a standard method of embedding the PHP code within the XHTML document. That means we can use the XHTML tags in PHP while displaying the output.
- The **print** function is used to create simple unformatted output. For example : The string can be displayed as follows

```
print "I am proud of my <b>country</b>"
```

The numeric value can also be displayed using the **print**. For example -

```
print(100);
```

It will display the output as 100.

- PHP also makes use of the **printf** function used in C. For example

```
printf("The student %d has %f marks",$roll_no,$marks);
```

- Following is a simple PHP document which makes use of the statements for displaying the output.

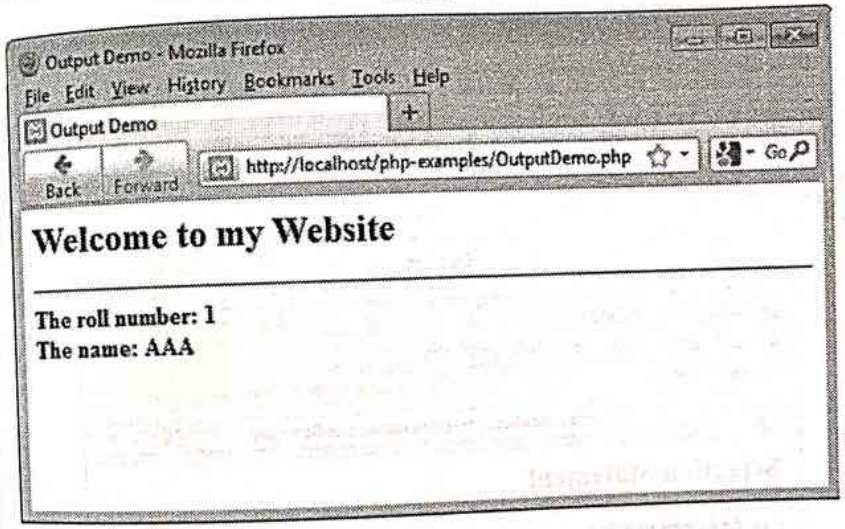
PHP Document[OutputDemo.php]

```
<html>
<head>
   <title> Output Demo</title>
</head>
<body>
 <?php
   print "<h2>Welcome to my Website </h2>";
```



```
   print "<hr/>";
   $roll_no=1;
   $name="AAA";
   printf("<b>The roll number: %d</b>",$roll_no);
print "<br/><b>";
   printf("The name: %s",$name);
   print "</b>";
?>
</body>
</html>
```

Output



5.7 Statements Control

- The control statements in PHP are similar to the control statements that are used in C.
- The control statements include if statements, while, do while and for loops.

5.7.1 The Selection Statement

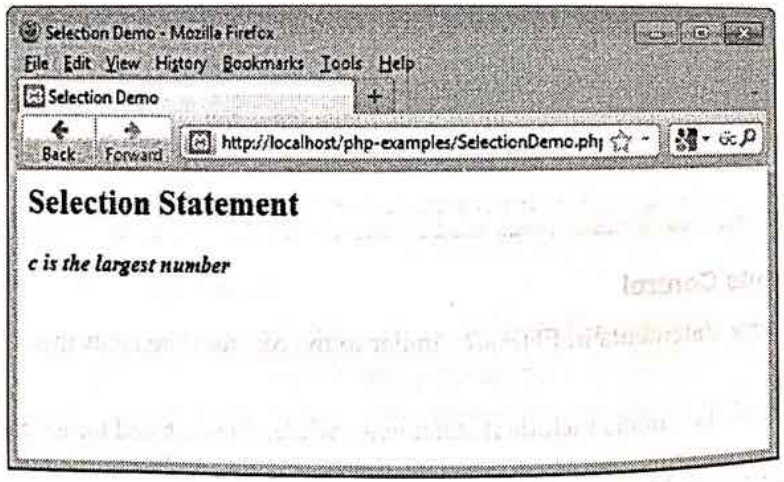
- The **if** statement, the **if ... else** statement or **if...elseif** statements are used in selection statements. Following is an example of it.

PHPDocument[selection.php]

```
<html>
<head>
   <title>Selection Demo</title>
</head>
```

```
<body>
<?php
   print "<h2>Selection Statement </h2>";
   $a=10;
   $b=20;
   $c=30;
   if($a>$b)
   if($a>$c)
   print "<b><I>a is the largest number </I></b>";
else
      print "<b><I> c is the largest number</I></b>";
   else
   if($b>$c)
      print "<b><I>b is the largest number</I></b>";
else
   print "<b><I>c is the largest number</I> </b>";
?>
</body>
</html>
```

Output



- Similar to **if** statement the **switch** statement can also be used for selection. Following is a simple PHP script for demonstrating switch statement -

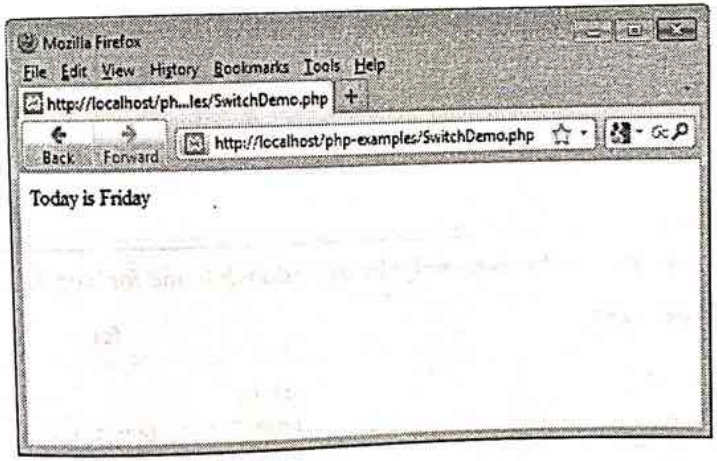
PHPDocument[SwitchDemo.php]

```
<?php
$today = getdate();
switch($today['weekday'])
{
```



```
   case "Monday":print "Today is Monday";
      break;
   case "Tuesday":print "Today is Tuesday ";
      break;
   case "Wednesday":print "Today is Wednesday ";
      break;
   case "Thursday":print "Today is Thursday";
      break;
   case "Friday":print "Today is Friday";
      break;
   case "Saturday":print "Today is Saturday";
      break;
   case "Sunday":print "Today is Sunday ";
      break;
default: print "Invalid input";
}
?>
```

Output



5.7.2 Looping Structure

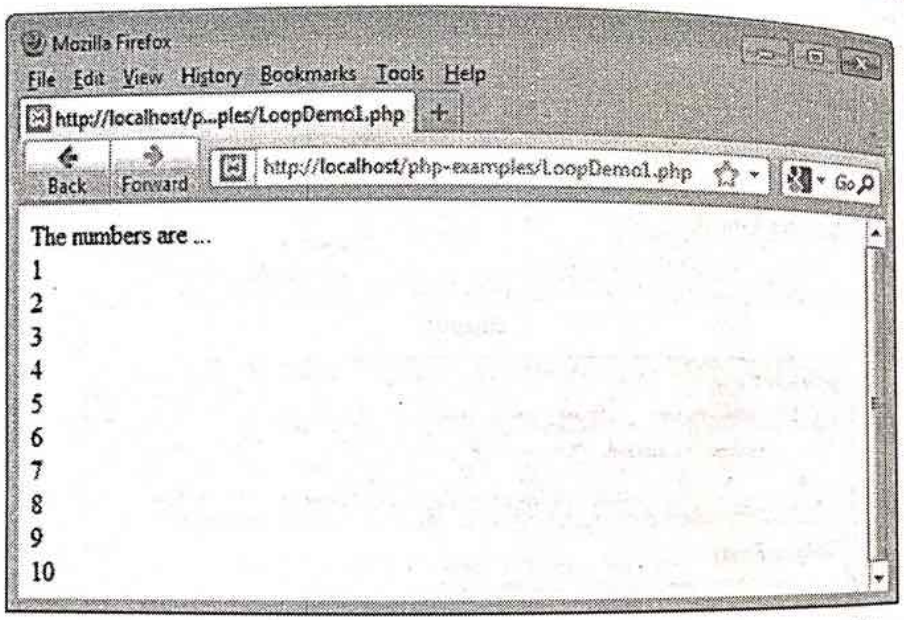
- The **while**, for and **do-while** statements of PHP are similar to JavaScript.
- Following is a simple PHP script which displays the first 10 number.

PHPDocument[LoopDemo1.php]

```
<?php
$i=1;
print "The numbers are ...";
print "<br/>";
```

```
while($i<=10)
{
    print $i;
    print "<br/>";
    $i++;
}
?>
```

Output



Similarly we can modify the above script by using do-while and for loop as follows -

do ...while

```
<?php
$i=1;
print "The numbers are ...";
print "<br/>";
do
{
    print $i;
    print "<br/>";
    $i++;
}while($i<=10);
?>
```

for

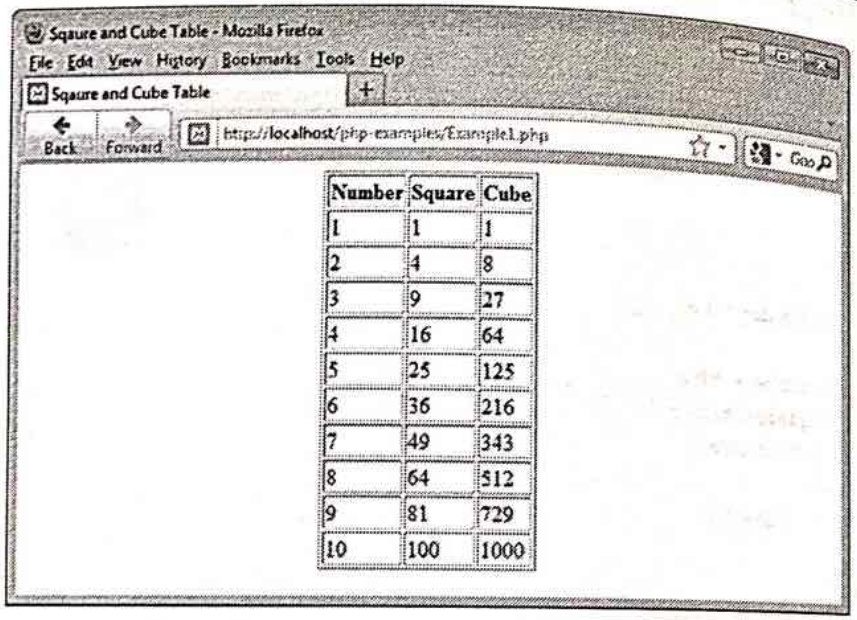
```
<?php
print "The numbers are ...";
print "<br/>";
for($i=1;$i<n)
{
    print $i;
    print "<br/>";
    $i++;
}
?>
```


Example 5.7.1 *Write a PHP script to display the squares and cubes of 1 to 10 numbers.*

Solution :

```
<html>
<head>
<title>Square and Cube Table </title>
</head>
<body>
<center>
<?php
print "<table border =1>";
print "<tr>";
print "<th>Number</th>";
print "<th>Square</th>";
print "<th>Cube</th>";
print "</tr>";
for($i=1;$i<=10;$i++)
{
print "<tr>";
print "<td>$i";
print "</td>";
print "<td>";
print $i*$i;
print "</td>";
print "<td>";
printpow($i,3);
print "</td>";
print "</tr>";
}
print "</table>";
?>
</center>
</body>
</html>
```

Output



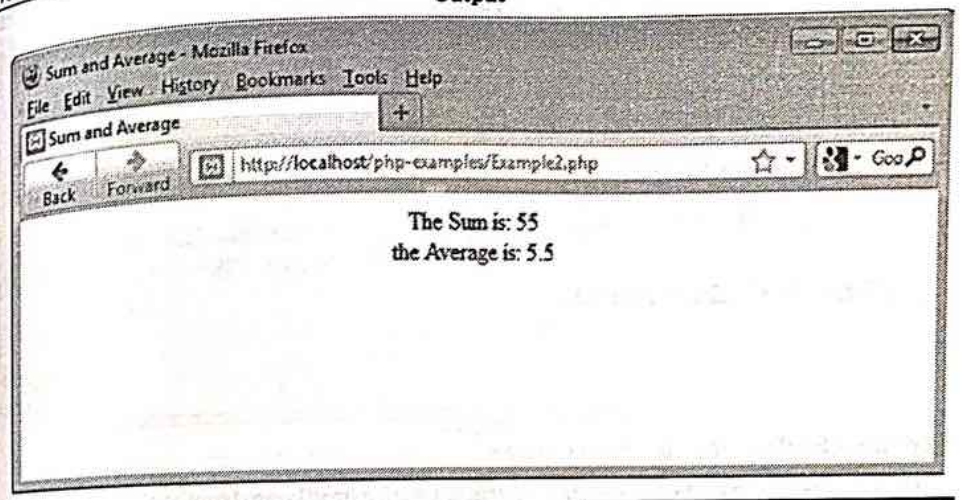
Number	Square	Cube
1	1	1
2	4	8
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

Example 5.7.2 *Write a PHP script to compute the sum and average of N numbers.*

Solution :

```
<html>
<head>
<title> Sum and Average </title>
</head>
<body>
<center>
<?php
$sum=0;
for($i=1;$i<=10;$i++)
{
    $sum+=$i;
}
$avg=$sum/10;
print "The Sum is:  $sum";
print "<br/>";
print "the Average is: $avg";
?>
</center>
</body>
</html>
```


Output



5.8 Arrays

SPPU : Dec.-18, May-18,19, Marks 6

- An array is a collection of similar type of elements, but in PHP you can have the elements of mixed type together in single array.
- In each PHP, each element has two parts key and value.
- The key represents the index at which the value of the element can be stored.
- The keys are positive integers that are in ascending order.

5.8.1 Creating and Manipulating Arrays

For creating an array in PHP, we use **array()** construct.

Syntax :

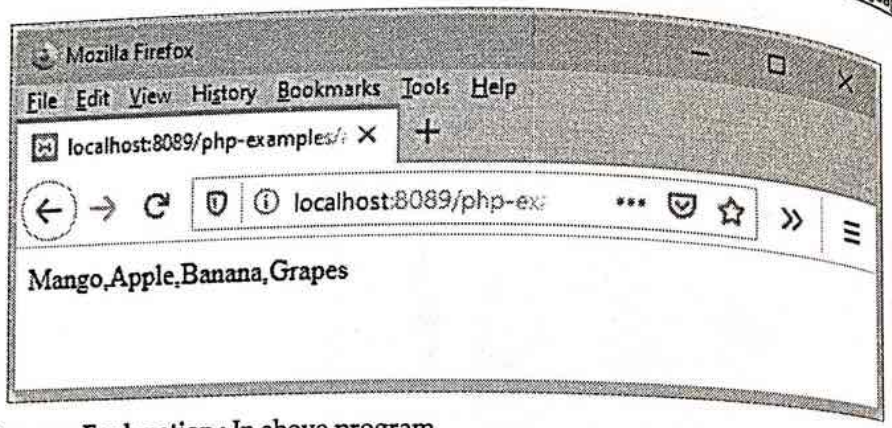
```
$array_name=array(value)
```

Example : Following PHP script shows how to create an array in PHP.

ArrayCreateDemo.php

```
<!DOCTYPE html>
<html>
<body>

<?php
$fruits = array("Mango", "Apple", "Banana","Grapes");
echo $fruits[0] . "," . $fruits[1] . "," . $fruits[2] . "," . $fruits[3];
?>
</body>
</html>
```

Output



Program Explanation : In above program,

1) We have created an array named **fruits** using **array()** construct
2) In the echo statement we are displaying each element of the array with the help of array index such as fruts[0],fruits[1],... and so on.

Adding and Deleting Elements

- In PHP, **arrays are dynamic.** That means they can grow in size or can shrink.
- We can add the element in the array using key/index that hasn't used. For example -

```
$name[5]="CCC";
```

- As there is no current value for index 5, the array will grow. Similarly we can skip the index value and add the element as follows -

```
$name[ ]="CCC";
```

- The advantage to this approach is that we don't have to worry about skipping an index key.
- We can skip some index and can insert the element in the array. For example -

```
$names=array("AAA","BBB","CCC","DDD");
$names[8]="HHH";
```

- If we iterate through the array elements then -

```
Array ([0]=>AAA [1]=>BBB [2]=>CCC [3]=>DDD [8] =>HHH)
```

- Thus there is now **gap** in our array. If we try to reference the array[4] then it will return NULL value which represents that there is no value present at that index.
- Using the **unset** function we can create gaps in the PHP array. For example -

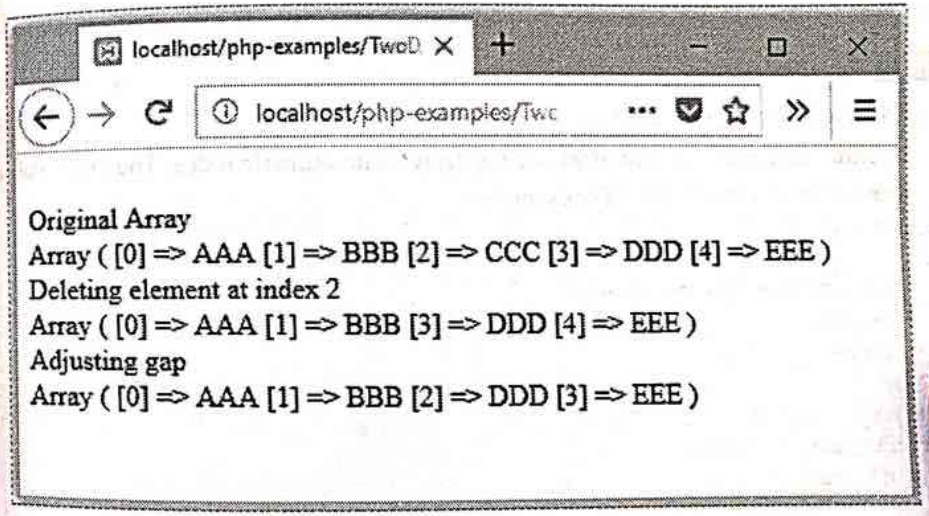
PHP Document

```
<!DOCTYPE html>
<html>
<body>
```



```
<?php
$Student = array("AAA","BBB","CCC","DDD","EEE");
echo "<br/>Original Array<br/>";
print_r($Student);
echo "<br/>Deleting element at index 2<br/>";
unset($Student[2]);
print_r($Student);
echo "<br/>Adjusting gap<br/>";
$Student=array_values($Student);
print_r($Student);
?>
</body>
</html>
```

Output



Script Explanation :

1. In above script we have used **unset** function to create a gap at index 2.
2. Later on we have used **array_values()** to adjust this gap by subsequent element.
3. Using the **print_r** function the array can be displayed on the browser window.

Checking if value exists

- The **isset ()** function is used to check whether **a variable is set or not.**
- If a variable is already unset with **unset()** function, it will no longer be set.
- The isset() function return false if testing variable contains a NULL value.

PHP Document

```
<!DOCTYPE html>
<html>
<body>
<?php
$Student =  array(1=>"AAA",2=>"BBB");
echo "<br/>Original Array<br/>";
print_r($Student);
if(isset($Student[0]))
    echo "<br/>There is no value set at index 0<br/>";
if(isset($Student[1]))
    echo "The value present at index 1 is: ".$Student[1];
?>
</body>
</html>
```

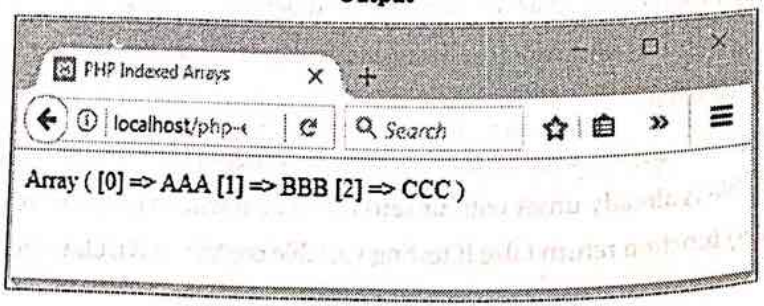
5.8.2 Types of Arrays

There are three types of arrays.

1. Indexed array : Indexed array are the arrays with numeric index. The array values can be stored from index 0. For example -

```
<html>
<head>
    <title>PHP Indexed Arrays</title>
    </head>
   <body>
<?php
$names = array("AAA", "BBB", "CCC");
// Printing array structure
print_r($names);
?>
</body>
</html>
```

Output



Here values get stored at corresponding index as follows -

```
$mylist[0]    =    10;
$mylist[1]    =    20;
$mylist[2]    =    30;
$mylist[3]    =    40;
$mylist[4]    =    50;
```

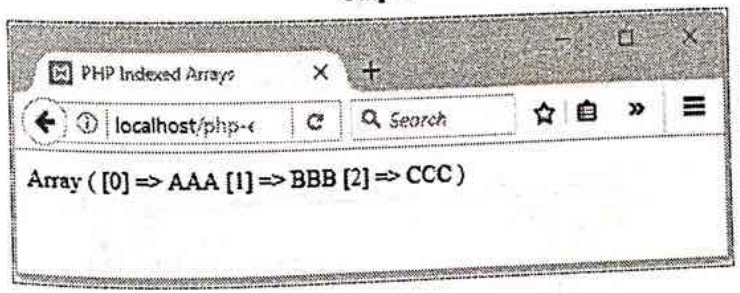
We can directly assign some value at specific index.

```
$mylist[5]    =    100;
```

2. Associated array : Associated arrays are the arrays with named keys. It is a kind of array with **name** and **value** pair. For example -

```
<html>
<head>
  <title>PHP Associative Array</title>
</head>
<body>
<?php
$city["AAA"] = "Pune";
$city["BBB"] = "Mumbai";
$city["CCC"] = "Chennai";

// Printing array structure
print_r($city);
?>
</body>
</html>
```

Output



3. Multidimensional Arrays

- PHP support for multidimensional arrays.
- We can store the elements in two dimensional array as

```
$Student = array
```

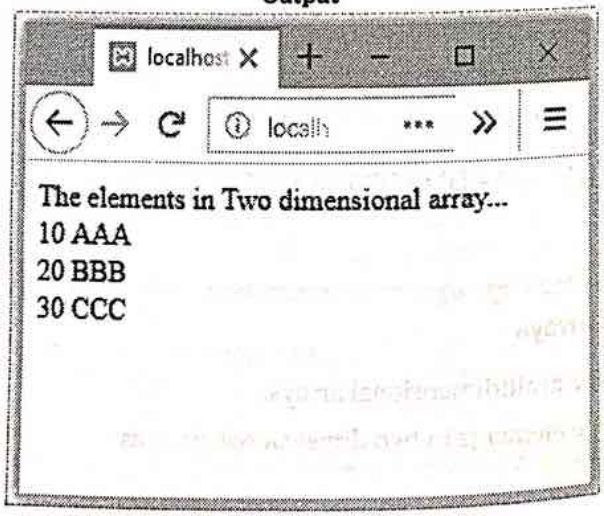
```
(
 array(10,"AAA"),
 array(20,"BBB"),
 array(30,"CCC"),
);
```

- The complete PHP program in which the multidimensional array is created and accessed is as follows -

PHP Document

```
<!DOCTYPE html>
<html>
<body>
<?php
$Student = array
(
 array(10,"AAA"),
 array(20,"BBB"),
 array(30,"CCC"),
);
echo "The elements in Two dimensional array...<br/>";
for ($row = 0; $row < 3; $row++) {
 for ($col = 0; $col < 2; $col++) {
  echo " ".$Student[$row][$col];
 }
 echo "<br/>";
}
?>
</body>
</html>
```

Output



5.8.3 Extracting Data from Arrays, Implode, Explode and Flip Functions

I) The extract function :

Using the extract() function the array keys becomes the variable name and the array values become the variable values.

Syntax :

```
int extract($input_array, $extract_rule, $prefix)
```

Where

$input_array : is the name of the array whose values need to be extracted. This is a required parameter.

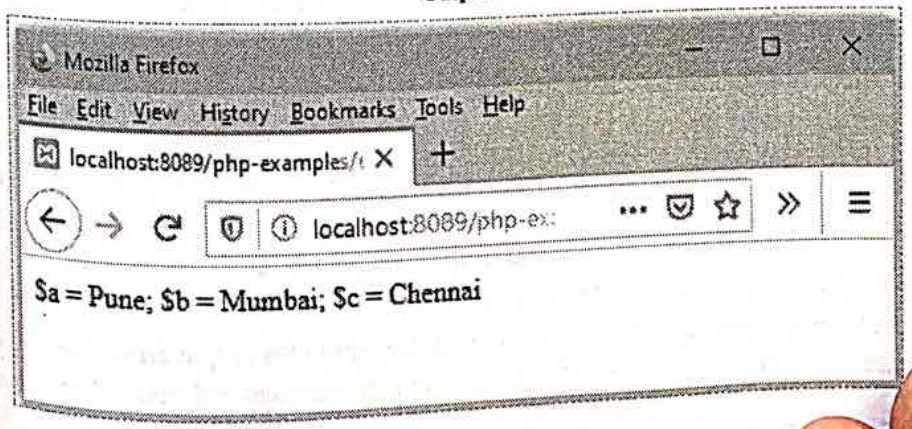
$extract_rule : This is an optional parameter. This parameter specifies how invalid variable names will be treated.

$prefix : The prefix is automatically separated from the array key by an underscore character. This is an optional parameter.

Example Program

```
<!DOCTYPE html>
<html>
<body>

<?php
$arr = array("a" => "Pune","b" => "Mumbai", "c" => "Chennai");
extract($arr);
echo "\$a = $a; \$b = $b; \$c = $c";
?>
</body>
</html>
```

Output



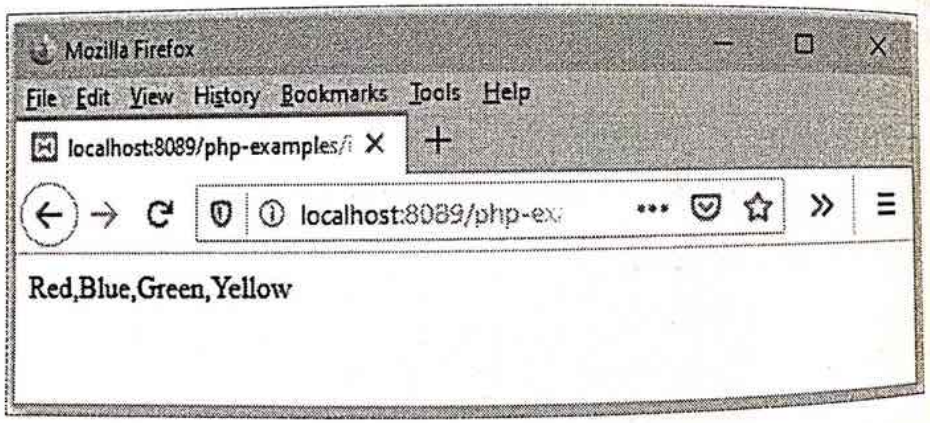
ii) The implode function :

The implode function converts array into the string. For example -

implodeDemo.php

```
<!DOCTYPE html>
<html>
<body>
<?php
$arr[0] = "Red";
$arr[1] = "Blue";
$arr[2] = "Green";
$arr[3] = "Yellow";
$text = implode(",",$arr);
echo $text;
?>
</body>
</html>
```

Output



iii) The explode function

The explode() function is to split a string.

Syntax :

```
explode(delimiter, string_name, limit)
```

Where

Delimiter : It sets the boundary string within the input string

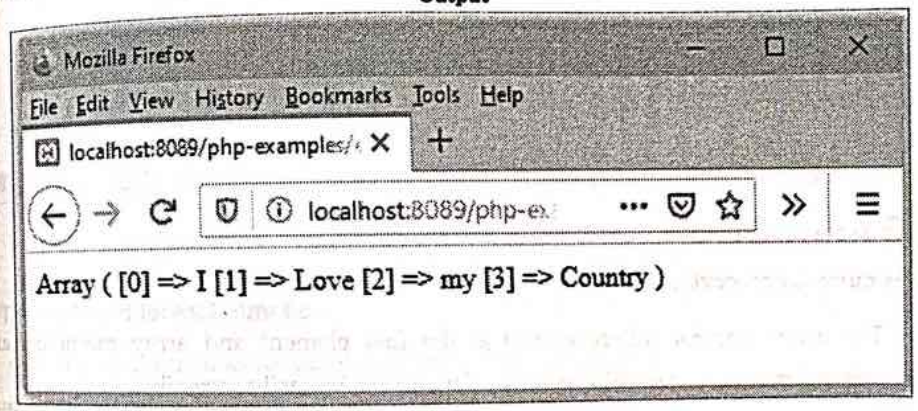
String_name : The name of the string to be split

Limit : It indicates the maximum number of elements in the output array if set to +ve value. If set to -ve value, all but the last element will be present in the output array.

Example Program

```
<!DOCTYPE html>
<html>
<body>

<?php
$str="I Love my Country";
$arr=explode(" ",$str);
print_r($arr);
?>
</body>
</html>
```

Output



iv) The flip function

The array_flip() function is used to exchange the keys with their associated values in array.

That means after applying the array_flip function we get keys from array become values and values from array become keys.

Syntax :

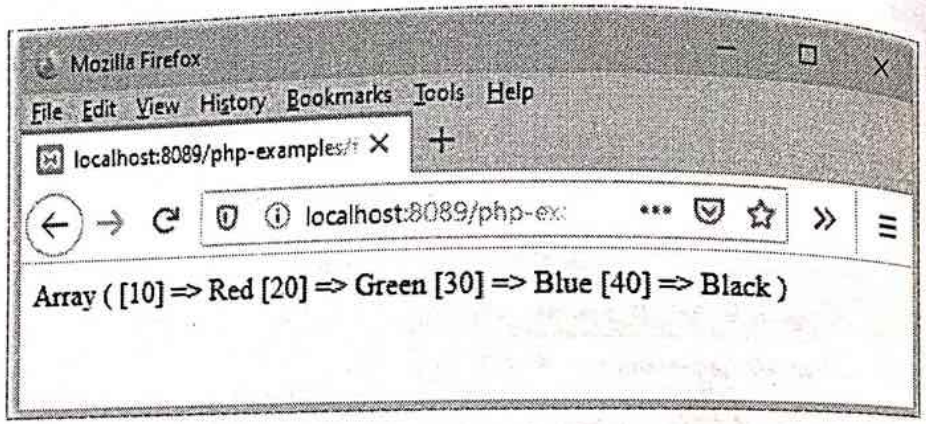
```
array_flip(name_of_array)
```

Example Program

```
<!DOCTYPE html>
<html>
<body>

<?php
$arr=array("Red"=>10,"Green"=>20,"Blue"=>30,"Black"=>40);
$result=array_flip($arr);
```

```
print_r($result);
?>

</body>
</html>
```

Output



5.8.4 Traversing Arrays

1. The current and next function

- The array element reference start at the first element and array maintains an internal pointer using which the next element can be easily accessible.
- This helps to access the array elements in sequential manner.
- The pointer **current** is used to point to the current element in the array. Using the **next** function the next subsequent element can be accessed. Following PHP code illustrates this idea -

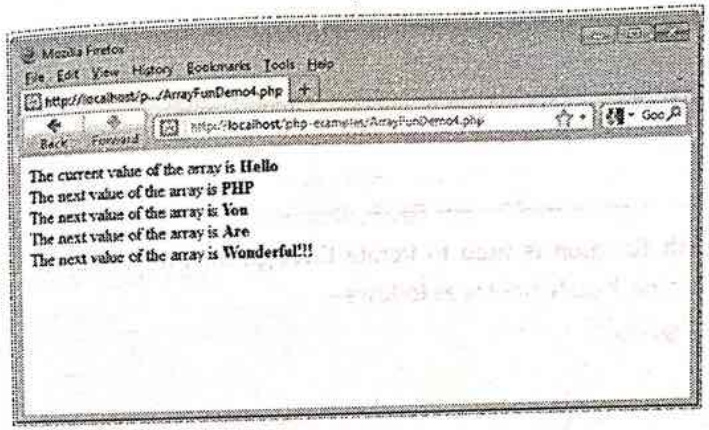
PHP Document[ArrayFunDemo4.php]

```
<?php
$mylist = array("Hello", "PHP","You","Are","Wonderful!!!");
$myval=current($mylist);
print("The current value of the array is <b>$myval</b>");
print "<br/>";
$myval=next($mylist);
print("The next value of the array is <b>$myval</b>");
print "<br/>";
$myval=next($mylist);
print("The next value of the array is <b>$myval</b>");
```



```
print "<br/>";
$myval=next($mylist);
print("The next value of the array is <b>$myval</b>");
print "<br/>";
$myval=next($mylist);
print("The next value of the array is <b>$myval</b>");
?>
```

Output



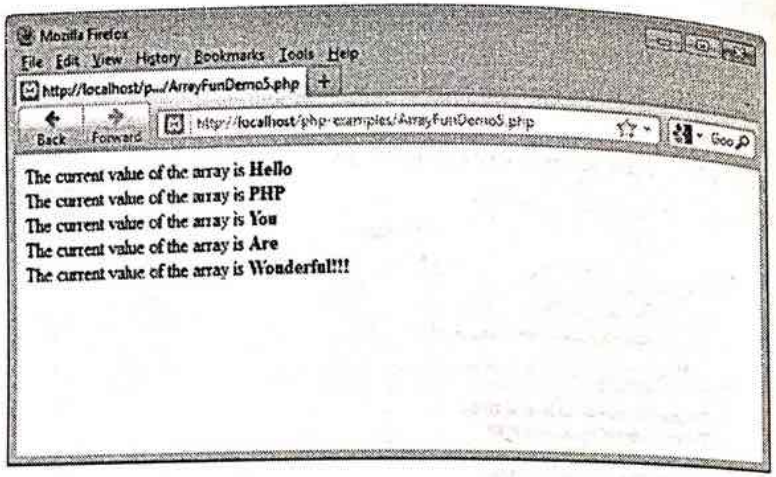
2. The each and foreach function

- Using **each** function we can iterate through the array elements.

PHP Document[ArrayFunDemo5.php]

```
<?php
$mylist = array("Hello", "PHP","You","Are","Wonderful!!!");
while($myval=each($mylist))
{
  $val=$myval["value"];
  print("The current value of the array is <b>$val</b>");
  print "<br/>";
}
?>
```

Output



- The **foreach** function is used to iterate through all the elements of the loop. The syntax of foreach statement is as follows -

```
foreach($array as $value)
{
  statements to be executed
}
```

- The above code can be modified and written as follows -

PHP Document[ArrayFunDemo6.php]

```
<?php
$mylist = array("Hello", "PHP","You","Are","Wonderful!!!");

foreach($mylist as $value)
{
   print("The current value of the array is <b>$value</b>");
   print "<br/>";
}
?>
```

The output will be the same as above.

Review Questions

1. *What is associative arrays in PHP ? Explain it with the help of simple PHP code.* **SPPU : May-18,19, Marks 6**
2. *What is multi-dimensional arrays in PHP ? Explain it with simple PHP code.* **SPPU : Dec.-18, Marks 6**

5.9 Functions

The functions in PHP are very much similar to the functions in C. Let us discuss it in detail.

5.9.1 General Characteristics of Functions

- The syntax of the function definition is as follows -

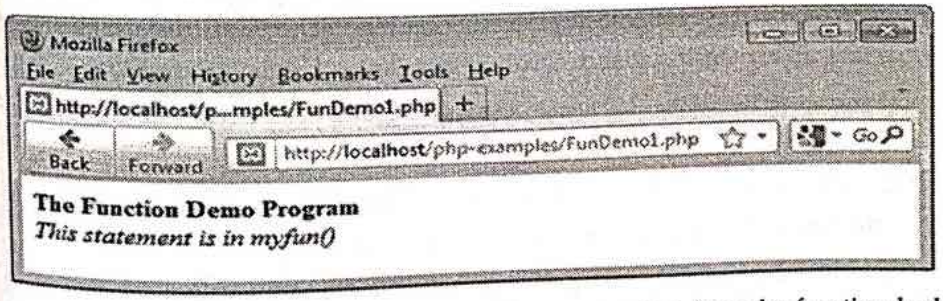
```
functionname_of_function(parameter list)
{
statements to be executed in function
...
...
...}
```

- The function gets executed only after the call to that function. The call to the function can be from anywhere in the PHP code. For example -

PHPDocument[FunDemo1.php]

```
<?php
functionmyfun()
{
   print "<i>This statement is in myfun()</i>";
}
print "<b>The Function Demo Program</b>";
print "<br/>";
myfun();
?>
```

Output



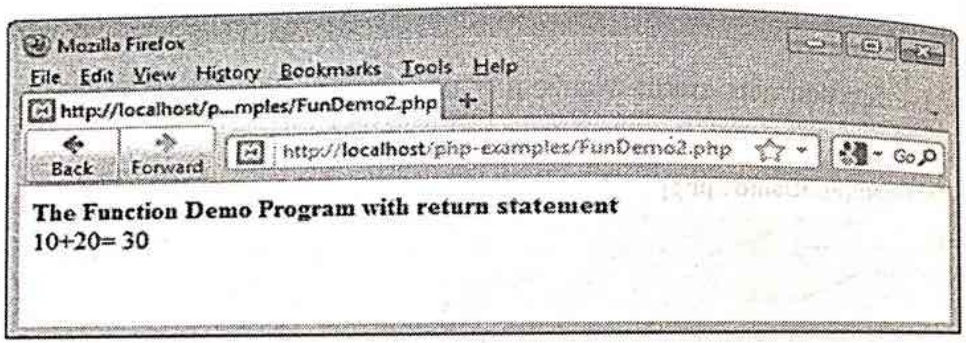
- The **return** statement is used for returning some value from the function body. Following PHP scripts shows this idea.

PHPDocument[FunDemo2.php]

```
<?php
function Addition()
```

```
{
    $a=10;
    $b=20;
    $c=$a+$b;
    return $c;
}
print "<b>The Function Demo Program with return statement</b>";
print "<br/>";
print "10+20= ".Addition();
?>
```

Output



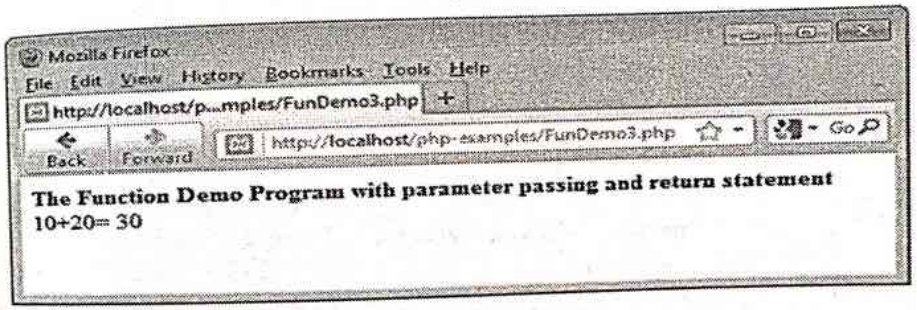
5.9.2 Parameters

- The parameter that we pass to the function during the call is called the **actual parameter.** These parameters are generally the expressions.
- The parameters that we pass to the function while defining it is called the **formal parameters.** These are generally the variables. It is not necessary that the number of actual parameters should match with the number of formal parameters.
- If there are few actual parameter and more formal parameters then the value of formal parameter will be some unbounded one.
- If there are many actual parameters and few formal parameters then the excess of actual parameters will be ignored.
- The default parameter passing technique in PHP is pass by value. The parameter passing by value means the values of actual parameters will be copied in the formal parameters. But the values of formal parameters will not be copied to the actual parameters.
- Following PHP script illustrates the functions with parameters.

PHP Document[FunDemo3.php]

```
<?php
function Addition($a,$b)
{
    $c=$a+$b;
    return $c;
}
print "<b>The Function Demo Program with parameter passing and return
statement</b>";
print "<br/>";
$x=10;
$y=20;
print "10+20= ".Addition($x,$y);
?>
```

Output



5.10 Built in Functions

PHP has number of predefined functions.

5.10.1 Functions based on Basic Operations

These function help in performing basic operations such floor, ceil, round and so on.

Function	Purpose
floor	The largest integer less than or equal to the parameter is returned. For example - floor(4.9) will return 4,
ceil	The smallest integer greater than or equal to the parameter is returned. For example - ceil(4.9) will return 5,
round	Nearest integer is returned.
abs	Returns the absolute value of the parameter.

min	It returns the smaller element.
max	It returns the larger element.

5.10.2 String Handling Functions

Various functions used for string handling are -

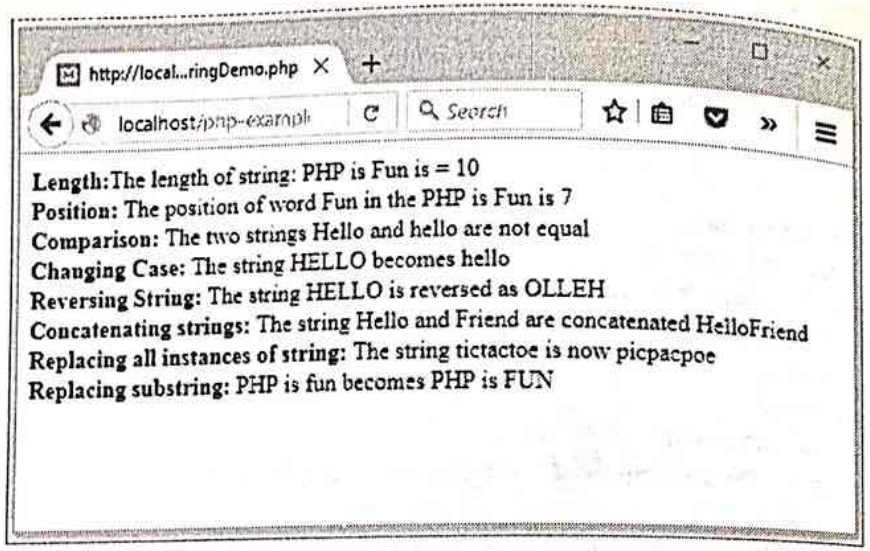
Function	Purpose	Sample PHP code	Output
strlen(string1)	It finds the total number of characters in the string	<?php $s="Friend"; echo $s <?php	6
strcmp (string1,string2)	It compares the two strings. It is case sensitive. If this function returns 0 then two strings are equal. If this function returns >0 then string1 is greater than string2. If this function returns <0 then string1 is less than string2.	<?php echo strcmp("PHP","PHP"); ?>	0
strtolower (string1)	This function converts the characters in string1 to lower case.	<?php echo strtolower("PHP."); ?>	php
strtoupper (string1)	This function converts the characters in string1 to upper case.	<?php echo strtolower("php"); ?>	PHP
trim(string1)	This function eliminates the white space from both the ends of the string.	<?php $str = " PHP "; echo "<h3>: $str</h3>"; echo "<h3>:".trim($str); echo "</h3>"; ?>	: PHP : PHP

Example 5.10.1 *Write a PHP program to do string manipulations.*

Solution : For this program we will apply various built in string manipulating functions to the string. The PHP code is as follows -

```
<?php
  $Str1="PHP is Fun";
  $length = strlen($Str1);
  echo "<b> Length:</b>The length of string: $Str1 is = $length";
  echo "<br/><b>Position: </b>The position of word Fun in the $Str1 is ".strpos($Str1,'Fun');
  $Str1="Hello";
  $Str2="hello";
  if(strcmp($Str1,$Str2))
     echo "<br/><b>Comparison: </b> The two strings $Str1 and $Str2 are not equal";
  else
     echo "<br/><b>Comparision: </b> The two strings $Str1 and $Str2 are equal";
  $Str1="HELLO";
  echo "<br/><b>Changing Case: </b> The string $Str1 becomes ".strtolower($Str1);
  echo "<br/><b>Reversing String: </b> The string $Str1 is reversed as ".strrev($Str1);
  $Str1="Hello";
  $Str2="Friend";
  echo "<br/><b>Concatenating strings: </b> The string $Str1 and $Str2 are concatenated
".$Str1.$Str2;
  echo "<br/><b>Replacing all instances of string: </b> The string tictactoe is now ";
  echo str_replace("t","p","tictactoe");
  $Str1="PHP is fun";
  $newstring=substr_replace($Str1,"FUN",7,9);
  echo "<br><b>Replacing substring: </b> $Str1 becomes $newstring";
?>
```

Output



5.10.3 Data Type Conversion Functions

Syntax :

```
Conversion_function($variable_name)
```

Example :

```
Intval($marks)
```

Syntax :

```
settype($variable_name,"datatype")
```

Example :

```
settype($marks,"integer");
```

- The **gettype** function is useful to obtain the data type of the variable.

5.10.4 Array Handling Functions

Function	Purpose
array()	Creates an array.
array_change_key_case(array,nameof case)	Changes the elements of array from lower to upper or from upper to lower case.
array_fill()	Fills the array with values.

array_keys()	Returns all the keys of the array.
array_merge()	Merges one or more arrays into one array.
array_search()	Searches array for given value and returns the keys.
array_value()	Returns the values of array.
count()	Returns the elements in array.
current()	Returns current element in an array.
each()	Returns current key and value pair from array.
end()	Returns last element value form array.
next()	Advances the internal pointer of the array to point to the next element.
sizeof()	Returns the total number of elements.
sort()	This function is for sorting the array.

5.11 Form Handling

- PHP is used for form handling. For that purpose the simple form can be designed in HTML and the values of the fields defined on the form can be transmitted to the PHP script using GET and POST methods.
- For forms that are submitted via "GET" method, we can obtain the form via the **$_GET** array variable.
- For forms that are submitted via "POST" method, we can obtain the form via the **$_POST** array variable.

Example 5.11.1 *Create HTML form with one textbox to get user's name. Also write PHP code to show length of entered name when, the HTML form is submitted.*

Solution : Step 1 : The HTML form can be created as follows.

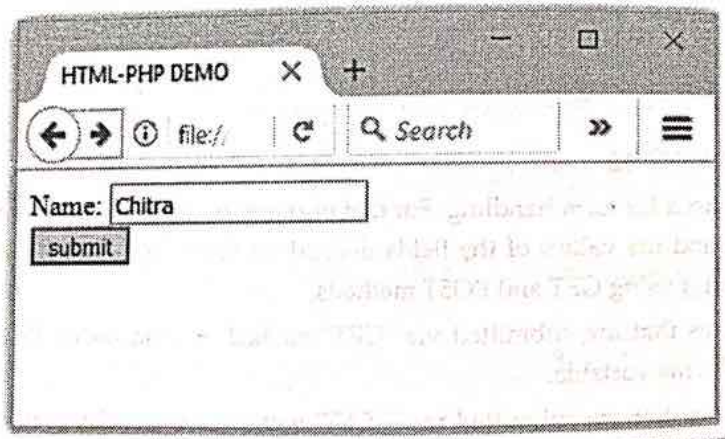
```
<!DOCTYPE html>
<html>
<head><title> HTML-PHP DEMO</title>
</head>
<body>
  <form method="post"
  action="http://localhost/getdata.php">
```

```
  Name: <input type="text" name="myname" size="20"/>
  <br/>
  <input type="submit" name="submit"
  value="submit"/>
 </form>
</body>
</html>
```

Step 2 : The PHP script to display the length of submitted name is as written below.

```
<?php
print "The name is: ";
print $_POST["myname"];
$len= strlen($_POST["myname"]);
print "<br/> The length of name is: ";
print $len;
?>
```

Output



Example 5.11.2 *Create HTML form to enter one number. Write PHP code to display the message about number is odd or even.*

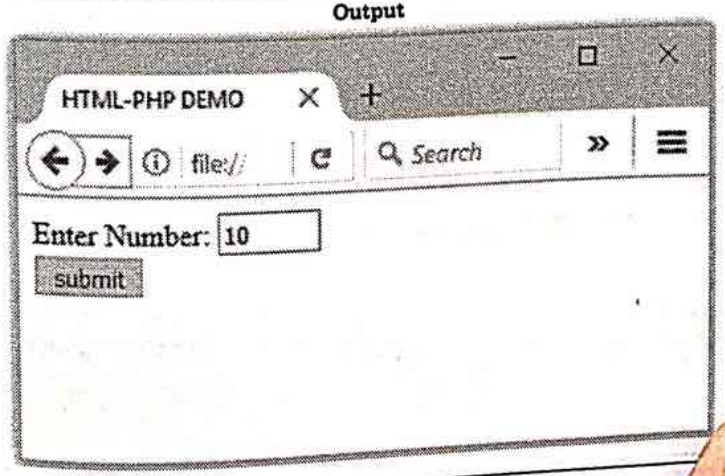
Solution : Step 1 : The HTML form for accepting number is created as below -

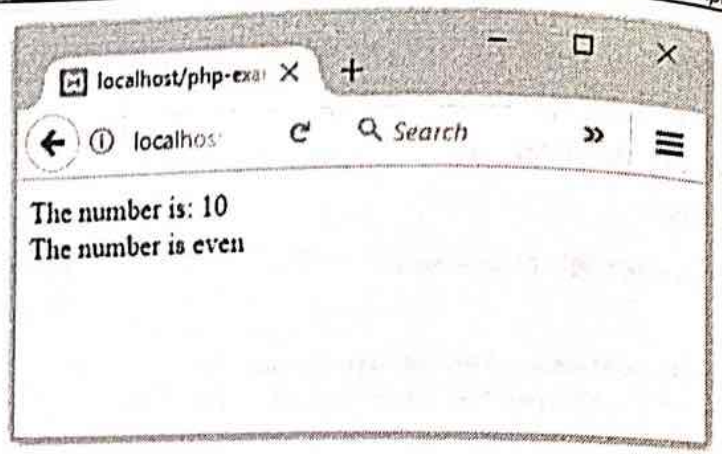
```
<!DOCTYPE html>
<html >
 <head><title> HTML-PHP DEMO</title>
</head>
<body>
  <form method="post" action="http://localhost/getdata.php">
   Enter Number: <input type="text" name="mynum"     size="5"/>
   <br/>
  <input type="submit" name="submit" value="submit"/>
  </form>

</body>
</html>
```

Step 2 : The PHP script deciding whether the number is even or odd is as given below -

```
<?php
print "The number is: ";
print $_POST["mynum"];
$a= $_POST["mynum"];
if($a%2==1)
 print "<br/> The number is odd ";
else
 print "<br/> The number is even ";
?>
```

Output





Example 5.11.3 *Create a form containing information sl.no, title of the book, publishers, quantity, price read the data from the form and display it using PHP script.*

Solution : Step 1 : Create an HTML page for inputting the data. Following is the code for HTML script .

HTML Document[Input.html]

```
<!doctype html>
<html>
<head>
<title> Book Order Form </title>
</head>
<body>
<h3> Enter the Book Data </h3>
<form method="post" action="http://localhost/php-examples/formdemo.php">
<table>
<tr>
<td>Sr.No.</td>
<td><input type="text" name="SLNo"></td>
</tr>
<tr>
<td>Book name</td>
<td><input type="text" name="BName"></td>
</tr>
<tr>
<td>Publisher</td>
<td><input type="text" name="PUBName"></td>
</tr>
<tr>
```



```
<td>Price</td>
<td><input type="text" name="Price">></td>
</tr>
<td>Quantity</td>
<td><input type="text" name="Qty">></td>
</tr>
<tr>
<td><input type="submit" value="Submit">></td>
<td><input type="submit" value="Clear">></td>
</tr>
</table>
</form>
</body>
</html>
```

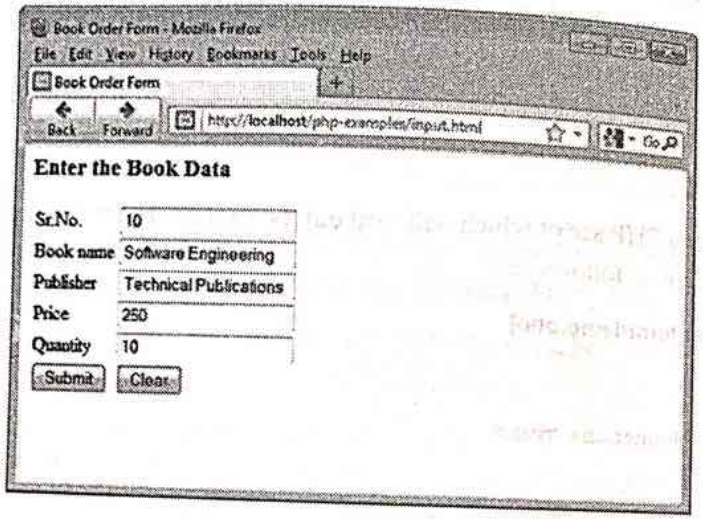
Step 2 : Create a PHP script which will read out the data entered by the user using HTML form. The code is as follows -

PHP Document[formdemo.php]

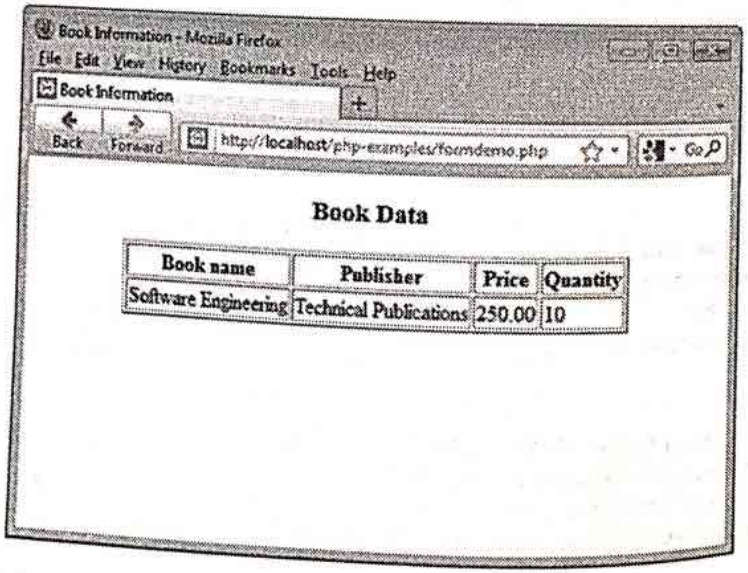
```
<html>
<head>
<title>Book Information</title>
</head>
<body>
<?php
$BName=$_POST["BName"];
$PUBName=$_POST["PUBName"];
$Price=$_POST["Price"];
$Qty=$_POST["Qty"];
?>
<center>
<h3> Book Data </h3>
<table border=1>
<tr>
<th>Book name</th>
<th>Publisher</th>
<th>Price</th>
<th>Quantity</th>
</tr>
<tr>
<td><?php print ("$BName"); ?></td>
<td><?php print ("$PUBName"); ?></td>
<td><?php printf("%3.2f",$Price); ?></td>
<td><?php printf("%d",$Qty); ?></td>
```

```
</tr>
</table>
</center>
</body>
</html>
```

Step 3 : Open some suitable web browser and enter the address for the HTML file which you have created in step 1.



Now click on the Submit button and the PHP file will be invoked. The output will then be as follows -

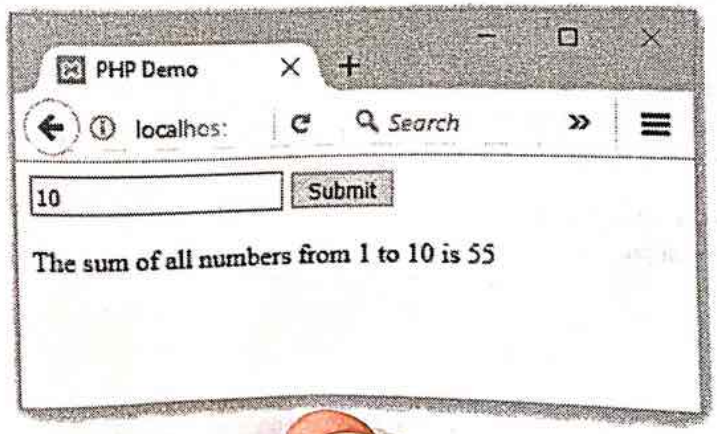


Example 5.11.4 *Write a PHP program to accept a positive integer 'N' through a HTML form and to display the sum of all the numbers from 1 to N.*

Solution :

```
<html>
  <head>
    <title> PHP Demo </title>
  </head>
  <body>
    <form method="post">
      <input type="text" name="num"/>
      <input type="submit" value="Submit"/>
    </form>
  </body>
</html>
<?php
  //getting values from HTML form
  $n = intval($_POST['num']);
  $sum=0;
  for($i=1;$i<=$n;$i++)
    $sum=$sum+$i;
  echo "The sum of all numbers from 1 to ".$n." is ".$sum;
?>
```

Output



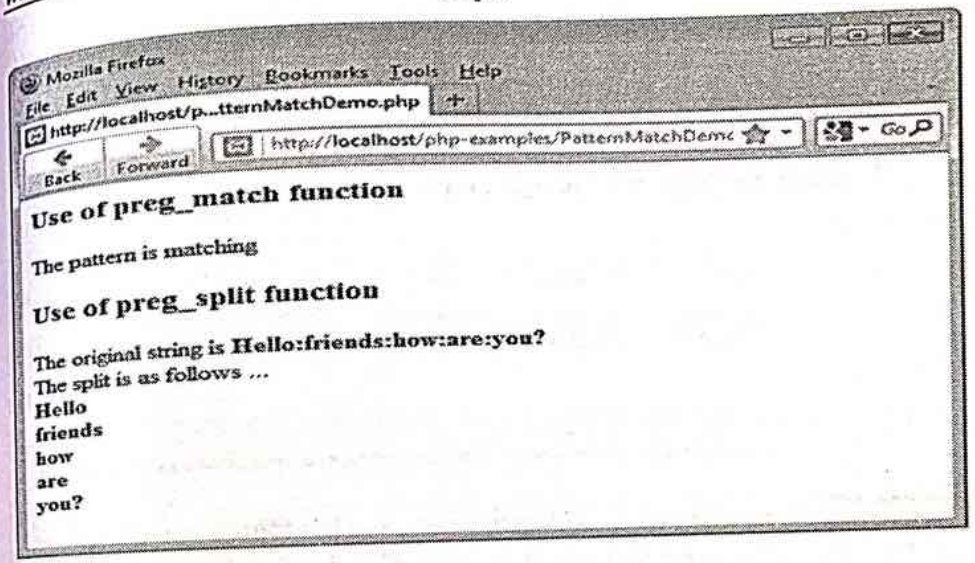
5.12 Pattern Matching

- The pattern matching technique which is used in PHP is of two types -The first one is based on **POSIX regular expression** and the another is based on **Perl regular expression.**
- There is a **preg_match** function which is used to check the matching of the pattern from the text. It takes two parameters the first parameter is the pattern to be matched and the second parameter is the text against which the pattern is to be matched.
- The **preg_split** is the function that splits the string into the chunks.
- Following is a PHP script in which both of these functions are discussed.

PHP Document[PatternMatchDemo.php]

```
<?php
print "<h3>Use of preg_match function</h3>";
if (preg_match("/gram/", "I like Programming in PHP!"))
    print "The pattern is matching <br />";
else
    print "The pattern does not match <br />";
print "<h3>Use of preg_split function</h3>";
$text ="Hello:friends:how:are:you?";
print "The original string is<b> $text</b>";
print "<br/>";
print "The split is as follows ...<br/>";
$chunks=preg_split(":/:",$text);
while($myval=each($chunks))
{
   $val=$myval["value"];
   print("<b>$val</b>");
   print "<br/>";
}
?>
```


Output



Example 5.12.1 *Explain the string comparison capability of PHP using regular expression with an example.*

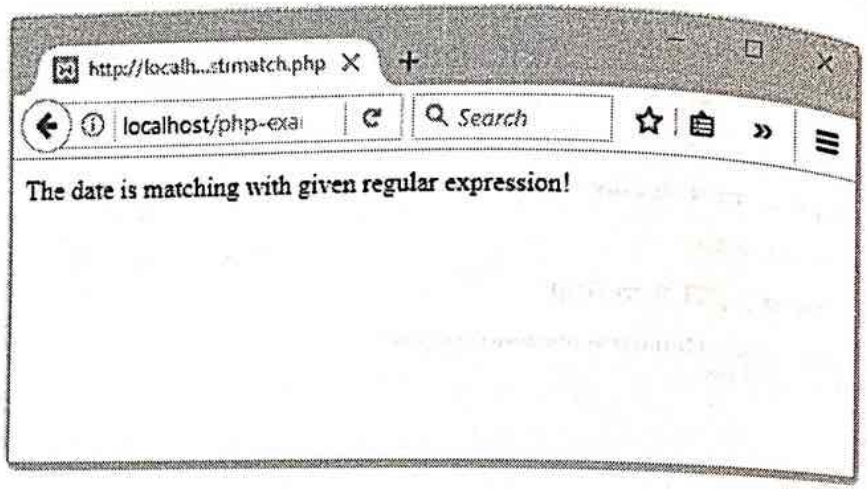
Solution : The preg_match function is used to compare two strings using regular expression. The first parameter to this function is a regular expression and second parameter is the string which is used for matching with the regular expression.

Following PHP program illustrates, the matching of date pattern using regular expression.

Example program in PHP

```
<?php
$regex = "/[a-zA-Z]+ \d+/";
if (preg_match($regex, "May 16"))
{
  // The expression "[a-zA-Z]+ \d+" matches the date string
  echo "The date is matching with given regular expression!";
}
else
{
  // If preg_match() returns false, then the regex does not
  // match the string
  echo "The regex pattern does not match.";
}
?>
```

Output



Example 5.12.2 *Write a PHP program that tests whether an email address is input correctly. Verify that the input begins with a series of characters followed by the @ character, another series of characters, a period '.' And a final series of characters. Test your program with both valid and invalid email address.*

Solution : Step 1 : The HTML form containing the text box for inputting the email address is created as follows -

```
<html>
<head>
    <title> Inserting Data into Database Table</title>
</head>
<body>
    <form method = "post" action = "http://localhost/php-examples/ValidateEmail.php ">
    <label>Email</label>
    <input type="text" name="email" />
    <br />
    <input type="submit" value="Submit">
    </form>
</body>
</html>
```

Step 2 : The PHP script that validates the given email address using regular expression is as follows -

ValidateEmail.php

```
<?php
$reg = '/^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$/';
```



```
$email=$_POST["email"];
if (preg_match($reg, $email))
{
    echo "Valid Email ID";
}
else
{
    echo "Invalid Email ID.";
}
?>
```

Example 5.12.3 *Design application to send an email using PHP.*

Solution :

```
<?php
    $to = "xyz@mysite.com";
    $subject = "Hi!";
    $body = "Hi,\n\nHow are you?";
    if (mail($to, $subject, $body))
    {
        echo("<p>Email successfully sent!</p>");
    }
    else
    {
        echo("<p>Failure in Email delivery!!!</p>");
    }
?>
```

5.13 Files

SPPU : Dec.-18, Marks 6

PHP is known as a server-side scripting language. Hence file handling functions such as create, read, write, append are some file related operations that are supported by PHP.

5.13.1 Opening and Closing Files

- The first step in file handling is opening of the file.
- It takes two parameters - The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened.

Modes	Description
r	Read only. Starts reading from the beginning of the file.
r+	Read/Write. Starts reading from the beginning of the file.

w	Write only. Opens and clears the contents of file; or creates a new file if it is not created
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it is not created.
a	Append. Opens and writes to the end of the file or creates a new file if it is not created.
a+	Read/Append. Preserves file content by writing to the end of the file.

For example :

```
$my_file = 'file.txt';
$file_handle = fopen($my_file, 'a') or die('Cannot open file: '.$my_file);
```

The **fopen** function returns TRUE if the required file is opened.

5.13.2 Reading from File

- The **fread** is the function which is used to read the file.
- It takes two parameters. The first parameter is the handle to the file and the second parameter is the number of bytes to be read. The **filesize** is the function which takes the filename as the parameter. For example -

```
$mystring = fread($file_handle, filesize("file.txt"));
```

- There is another function named **file_get_contents** using which the contents of the file can be obtained.
- The **fgets()** function is used to read a single line from the file. For example, following code displays the contents of the file line by line.

```
while(!feof($file_handle))
{
   echo fgets($file_handle). "<br />";
}
```

5.13.3 Writing to a File

- The **fwrite** is the function which is used to write the contents to the file.
- It takes two parameters - the first parameter is the handle to the file and the second parameter is the number of bytes to be written. For example -

```
$Written_string = fread($file_handle, $my_data);
```

Example 5.13.1 *Create a form containing information sl.no, title of the book, publishers, quantity, price read the data from the form and write onto the file using PHP script.*

Solution : Step 1 : We will create an input form for the user to enter the data. The HTML code is as follows. (It is the same HTML form which we have discussed in the last section)

HTML Document[Input.html]

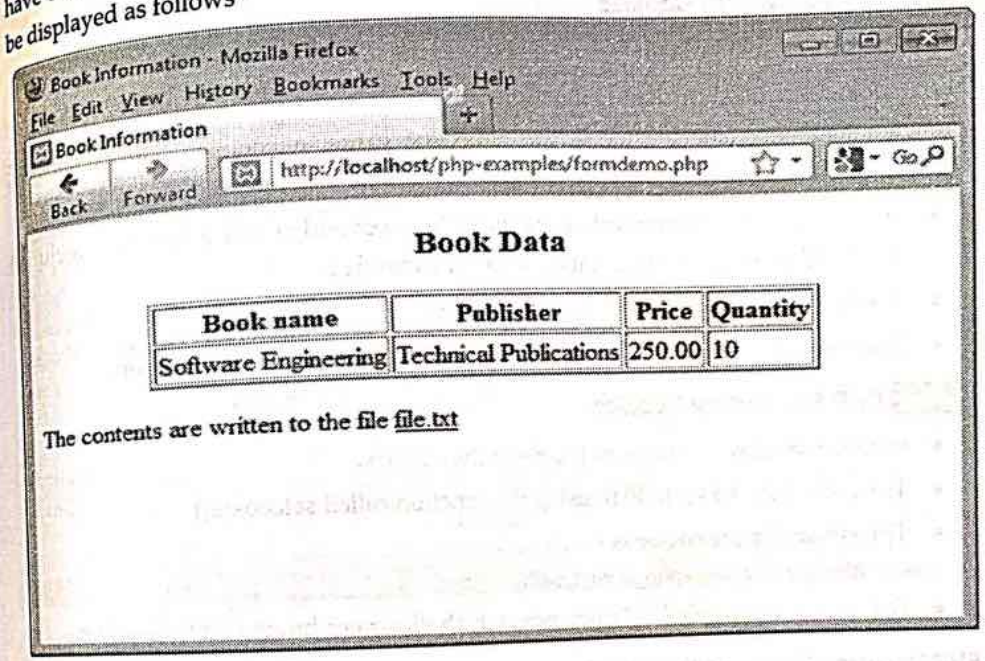
```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<title> Book Order Form </title>
</head>
<body>
<h3> Enter the Book Data </h3>
<form method="post" action="http://localhost/php-examples/formdemo.php">
<table>
<tr>
<td>Sr.No.</td>
<td><input type="text" name="SLNo"></td>
</tr>
<tr>
<td>Book name</td>
<td><input type="text" name="BName"></td>
</tr>
<tr>
<td>Publisher</td>
<td><input type="text" name="PUBName"></td>
</tr>
<tr>
<td>Price</td>
<td><input type="text" name="Price"></td>
</tr>
<td>Quantity</td>
<td><input type="text" name="Qty"></td>
</tr>
<tr>
<td><input type="submit" value="Submit"></td>
<td><input type="submit" value="Clear"></td>
</tr>
</table>
</form>
</body>
</html>
```

Step 2 : Following is a simple PHP script which displays the data and then writes it to the file.

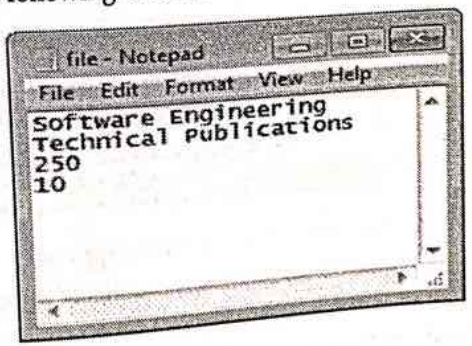
PHP Document

```
<html>
<head>
 <title>Book Information</title>
</head>
<body>
<?php
$BName=$_POST["BName"];
$PUBName=$_POST["PUBName"];
$Price=$_POST["Price"];
$Qty=$_POST["Qty"];
?>
<center>
<h3> Book Data </h3>
<table border=1>
<tr>
 <th>Book name</th>
 <th>Publisher</th>
 <th>Price</th>
 <th>Quantity</th>
</tr>
<tr>
 <td><?php print ("$BName"); ?></td>
 <td><?php print ("$PUBName"); ?></td>
 <td><?php printf("%3.2f",$Price); ?></td>
 <td><?php printf("%d",$Qty); ?></td>
</tr>
</table>
</center>
<?php
$my_file = 'file.txt';
$file_handle = fopen($my_file, 'a') or die('Cannot open file:  '.$my_file);
fwrite($file_handle, $BName);
fwrite($file_handle, "\r\n");
fwrite($file_handle, $PUBName);
fwrite($file_handle, "\r\n");
fwrite($file_handle, $Price);
fwrite($file_handle, "\r\n");
fwrite($file_handle, $Qty);
fclose($file_handle);
?>
<p>The contents are written to the file <u>file.txt<u></p>
</body>
</html>
```


Step 3 : Now open the web browser and type the address of the html file which you have created in step 1. On clicking the submit button on the this input form the data will be displayed as follows -



Step 4 : Open the current directory folder (The folder in which your html and PHP scripts are present). You will obtain a text file file.txt. Just Open the file and see the contents. I have got the following contents -



Review Question

1. *Explain server side include in PHP with sample code.* **SPPU : Dec.-18, Marks 6**

5.14 Cookies

In this section we will discuss how to create and read cookies.

5.14.1 Introduction to Cookies

- Cookie is a small file that server embeds in the user's machine.
- Cookies are used to identify the users.
- A cookie consists of **a name** and **a textual value**. A cookie is created by some software system on the server.
- In every HTTP communication between browser and server a header is included. The header stores the information about the message.
- The header part of http contains the cookies.
- There can be one or more cookies in browser and server communication.

5.14.2 PHP Support for Cookies

- PHP can be used to create and retrieve the cookies.
- The cookie can be set in PHP using the function called setcookie()
- The syntax for the cookie is -

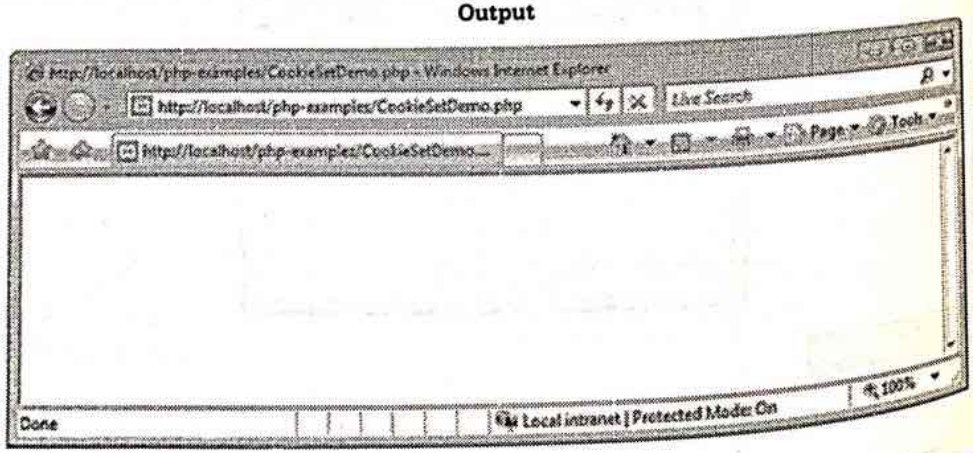
```
setcookie(name,value,expire_period,path,domain)
```

- Following is a simple PHP document which illustrates how to set the cookies -

PHP Document[CookieSetDemo.php]

```
<?php
$Cookie_period=time()+60*60*24*30;
setcookie("Myname", "Monika", $Cookie_period);
?>
```

Output



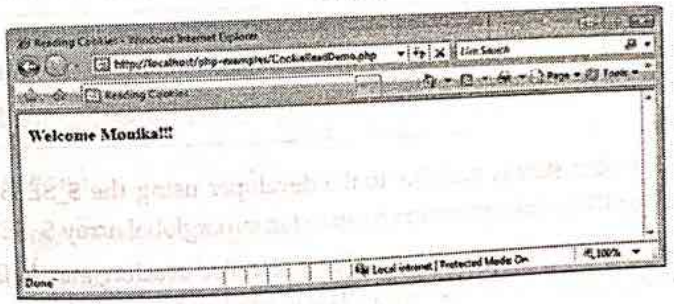
Note that you have got the blank screen it indicates that the cookie is set. In above PHP document we have set the PHP script for one month. Just observer the third parameter of the setcookie function.

Now you can retrieve the cookie and read the value to ensure whether or the cookie is set.

PHP Document[CookieReadDemo.php]

```
<html>
<head><title>Reading Cookies</title>
<body>
<?php
if (isset($_COOKIE["Myname"]))
 echo "<h3>Welcome " . $_COOKIE["Myname"]."!!!</h3>";
else
 echo "<h3>Welcome guest!</h3>";
?>
</body>
</html>
```

Output



Program Explanation

The isset function is used for checking whether or not the cookie is set. Then using the $_COOKIE the value of the cookie can be retrieved.

SPPU : Dec.-19, Marks 8

5.15 Session Tracking

- When you open some application, use it for some time and then close it. This entire scenario is named as **session.**
- **Session state** is a server-based state mechanism that lets web applications store and retrieve objects of any type for each **unique user session.** That is, each browser session has its own session state stored as a serialized file on the server, which is deserialized and loaded into memory as needed for each request. Refer Fig. 5.15.1.

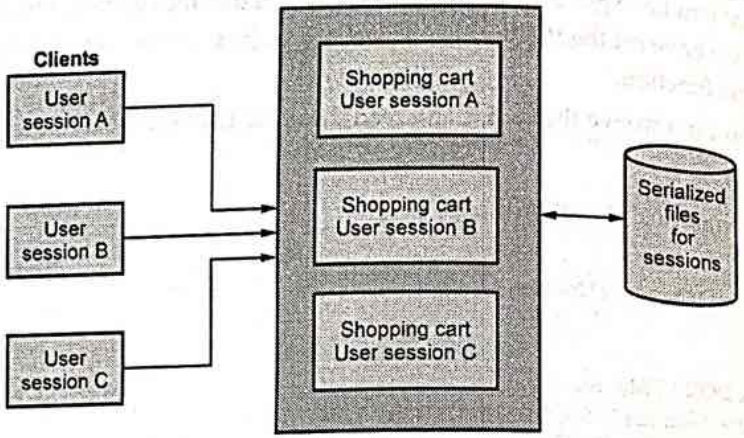


Fig. 5.15.1 Session state

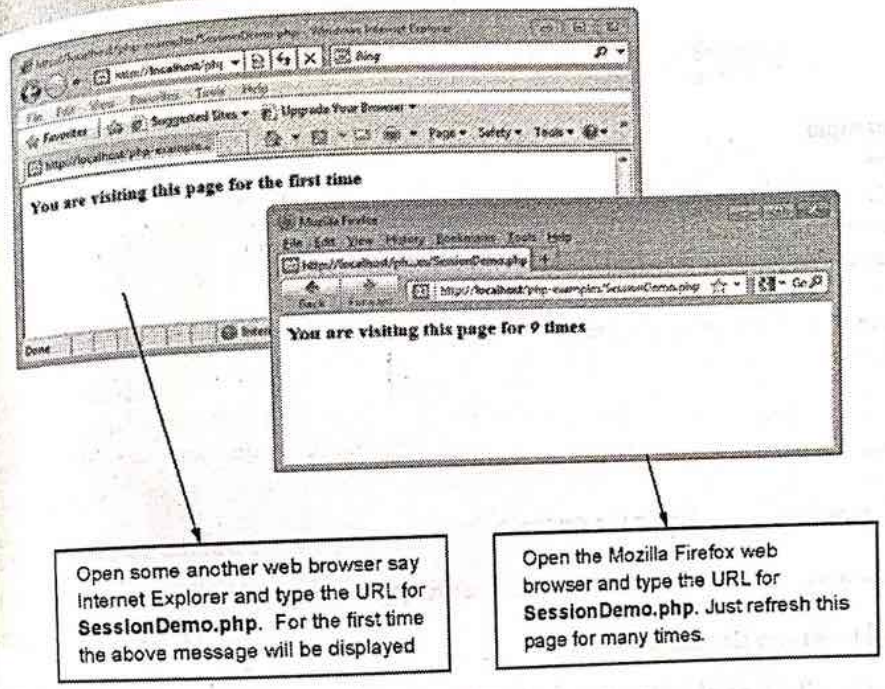
- Because server storage is a finite resource, objects loaded into memory are released when the request completes, making room for other requests and their session objects. This means there can be more active sessions on disk than in memory at any one time.
- Sometimes the information about the session is required by the server. This information can be collected during the session. This process is called **session tracking.**
- In PHP, session state is available to the developer using the $_SESSION variable. The unique ID for the sessions can be stored in superglobal array $_SESSION.
- PHP keeps track of session by using a function called **session_start().** Due to the call to **session_start()** function the session ID is created and recorded.
- Following is a simple PHP script in which the information about session is tracked.

PHP Document[SessionDemo.php]

```
<?php
session_start();
if(isset($_SESSION['pgvisit']))
{
   $_SESSION['pgvisit']=$_SESSION['pgvisit']+1;
echo "<h3>You are visiting this page for <i>". $_SESSION['pgvisit']."</i> times</h3>";
}
else
{
   $_SESSION['pgvisit']=1;
```



```
echo "<h3>You are visiting this page for the first time</h3>";
}
?>
```



Program Explanation :

- In above program, we have started the session by using **session_start()** function.
- The **isset()** function checks if the **pgvisit** variable has already been set. If **pgvisit** has been set, we can increment our counter. If **pgvisit** is not set, then we create a set it to 1.
- The value of **pgvisit** is displayed on the screen.

Review Question

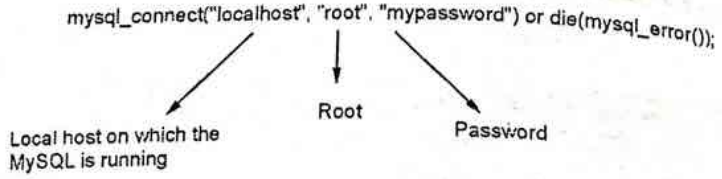
1. *Explain how session and cookies are used for session management in PHP.* **SPPU : Dec.-19, Marks 8**

5.16 Using MySQL with PHP

SPPU : May-18, Marks 6, Dec.-19, Marks 8

5.16.1 Connecting to Database

The PHP function **mysql_connect** connects to the MySQL server. There are three parameters that can be passed to this function. For example -



For example

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost:3306/mydb","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
else
{
print"connected";
}
mysql_close($conn); //closing the database
?>
```

The database can be closed using the function **mysql_close**

5.16.2 Creating Database

We can create a database using the function **mysql_query**. The **mysql_error()** function is used to obtain the error messages if any command gets failed.

For example -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost ","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
//Create a database
if(mysql_query("CREATE DATABASE mydb",$conn))
  {
  print "Database created";
  }
  else
  {
  print "Error creating database: " . mysql_error();
  }
```



```
mysql_close($conn); //closing the database
?>
```

5.16.3 Selecting Database

The database can be selected using the function **mysql_select_db().**

For example -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost:3306/mydb ","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
//Select a database
mysql_select_db("mydb", $conn);
mysql_close($conn); //closing the database
?>
```

5.16.4 Listing Database

There are various databases present in the MySQL which can be displayed using the function **mysql_list_db().**

For example -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost:3306/mydb ","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}

//Show databases
$db_lists=mysql_list_db($conn)
while($i=$mysql_fetch_object($db_list))
{
echo $i->Database. "\n"; //Present Databases will be displayed
}
mysql_close($conn); //closing the database
?>
```

5.16.5 Listing Table Names

The tables are present inside the databases. The function **mysql_list_tables** function is used to display the tables present in the database or we can use **mysql_query**

For example -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost ","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
$table_list=mysql_query("SHOW TABLES FROM mydb",$con)
if(!$table_list)
{
 print "Error!!! " . mysql_error();
}
while($i=$mysql_fetch_row($table_list))
{
echo $i[0]. "\n"; //Listing the tables
}
mysql_close($conn); //closing the database
?>
```

5.16.6 Creating a Table

Before creating the table a database must be created and within which the table can be created. Note that before creating a table the desired database must be selected.

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost ","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}

//Create a database
if(mysql_query("CREATE DATABASE mydb",$con))
   {
   print "Database created";
   }
   else
   {
```



```
   print "Error creating database: " . mysql_error();
   }
mysql_select_db("mydb",$conn); //before creating table select the database
$query="CREATE TABLE my_table
      (
         id INT(4),
         name VARCHAR(20)
      )";
mysql_query($query,$conn);//Execution of Query

mysql_close($conn); //closing the database
?>
```

5.16.7 Inserting Data

Using PHP we can insert the data in the database created in the MySQL.

For example -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
mysql_select_db("mydb",$conn); //select the database
$query=" INSERT INTO my_table (id,name)  VALUES(1,'SHILPA')";
mysql_query($query,$conn);//Execution of Query
$query=" INSERT INTO my_table (id,name)  VALUES(2,'MONIKA')";
mysql_query($query,$conn);//Execution of Query

mysql_close($conn); //closing the database
?>
```

Sometimes values that can be inserted in the table can be obtained from some another script and these values might be present in the variables. Insertion of such data can be done using $_POST variables. It is as shown below -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
mysql_select_db("mydb",$conn); //select the database
```

```
$query=" INSERT INTO my_table (id,name)
   VALUES('$_POST[MyID]','$_POST[MyName]')";
mysql_query($query,$conn);//Execution of Query

mysql_close($conn); //closing the database
?>
```

We can display these records using the SELECT query. For example -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost ","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
mysql_select_db("mydb",$conn); //select the database
//Execution of Query for displaying the data
$result=mysql_query("SELECT * FROM my_table");
while($row = mysql_fetch_array($result))
    {
        echo $row['id'] . " " . $row['name'];        // Each record will be displayed line by line

        echo "<br />";
    }
mysql_close($conn); //closing the database
?>
```

5.16.8 Altering Table

The ALTER table command is useful for various reasons such as to add some columns in the existing table, to delete some column from the table, for changing the size of some field or to change the name of some column. In the following example we are adding one column named **addr** to the existing table **my_table**

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
mysql_select_db("mydb",$conn); //select the database
$query=" ALTER TABLE my_table ADD COLUMN addrVARCHAR(30)";
mysql_query($query,$conn);//Execution of Query
mysql_close($conn); //closing the database
?>
```


5.16.9 Deleting Database

We can delete the values from the database using the DELETE query.

For example -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
mysql_select_db("mydb",$conn); //select the database
$query=" DELETE FROM my_table WHERE id=1";
mysql_query($query,$conn);//Execution of Query
mysql_close($conn); //closing the database
?>
```

Similarly we can delete a database using the query DROP.

For example -

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost","root","mypassword");
if(!$conn)
{
die('error in connection'.mysql_error());
}
mysql_select_db("mydb",$conn); //select the database
$query=" DROP DATABASE mydb";
mysql_query($query,$conn);//Execution of Query
mysql_close($conn); //closing the database
?>
```

Examples

Example 5.16.1 *Write a PHP script to create a new database table with 4 fields of your choice and perform following database operations.*

1. Insert 2. Delete 3. Update 4. Display

Solution : We will create a table in the database test. The name of the table ismytable. Then we will insert the record into the table using the INSERT command, update particular field of the record using the command UPDATE and delete the record using the command DELETE.

The PHP script is as follows -

PHPDocument[DBDemo.php]

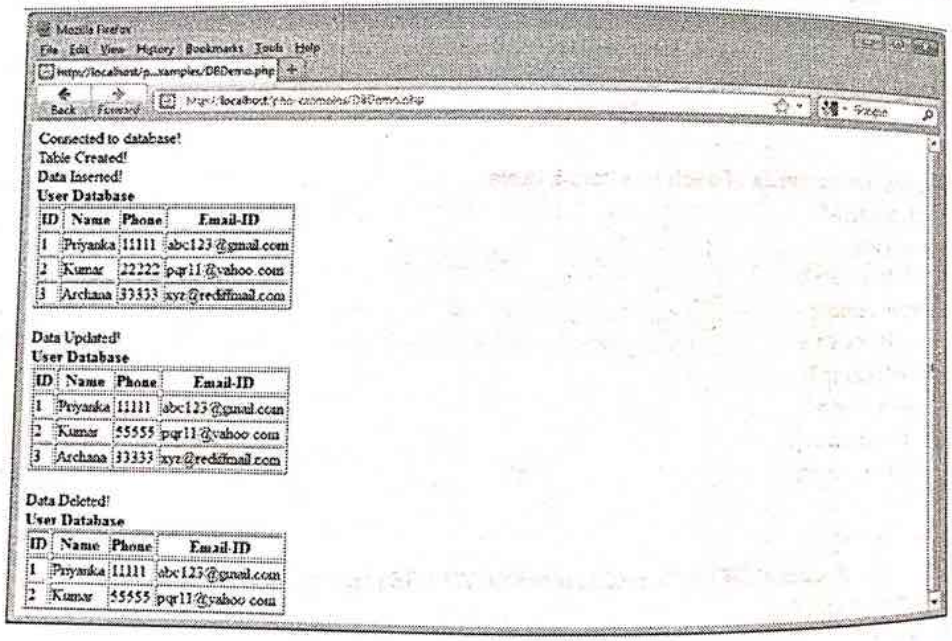
```
<?php
// Make a MySQL Connection
mysql_connect("localhost", "root", "mypassword") or die(mysql_error());
mysql_select_db("test") or die(mysql_error());
echo "Connected to database!";
mysql_query("CREATE TABLE mytable(id INT NOT NULL AUTO_INCREMENT, PRIMARY
KEY(id), name VARCHAR(30),
phoneINT,emailIdVARCHAR(30))")
or die(mysql_error());
print "<br/>";
echo "Table Created!";

// Insert a row of information into the table
mysql_query("INSERT INTO mytable
(name, phone,emailId) VALUES('Priyanka', '11111','abc123@gmail.com' ) ")
or die(mysql_error());

mysql_query("INSERT INTO mytable
(name, phone,emailId) VALUES('Kumar', '22222','pqr11@yahoo.com' ) ")
or die(mysql_error());

mysql_query("INSERT INTO mytable
(name, phone,emailId) VALUES('Archana', '33333','xyz@rediffmail.com' ) ")
or die(mysql_error());
print "<br/>";
echo "Data Inserted!";
$result =mysql_query("SELECT * FROM mytable")
or die(mysql_error());
print "<br/>";
print "<b>User Database</b>";
echo "<table border='1'>";
echo "<tr><th>ID</th><th>Name</th><th>Phone</th><th>Email-ID</th>
</tr>";
while($row = mysql_fetch_array( $result ))
  {
  // Print out the contents of each row into a table
  echo "<tr><td>";
  echo $row['id'];
  echo "</td><td>";
  echo $row['name'];
  echo "</td><td>";
```



```
  echo $row['phone'];
  echo "</td><td>";
  echo $row['emailId'];
  echo "</td></tr>";

}
echo "</table>";
$result = mysql_query("UPDATE mytable SET phone='55555' WHERE phone='22222'")
or die(mysql_error());

print "<br/>";
echo "Data Updated!";
$result =mysql_query("SELECT * FROM mytable")
or die(mysql_error());
print "<br/>";
print "<b>User Database</b>";
echo "<table border='1'>";
echo "<tr><th>ID</th><th>Name</th><th>Phone</th><th>Email-ID</th>
</tr>";
while($row = mysql_fetch_array( $result ))
{
// Print out the contents of each row into a table
echo "<tr><td>";
echo $row['id'];
echo "</td><td>";
echo $row['name'];
echo "</td><td>";
echo $row['phone'];
echo "</td><td>";
echo $row['emailId'];
echo "</td></tr>";
}
echo "</table>";
$result = mysql_query("DELETE FROM mytable WHERE phone='33333'")
or die(mysql_error());
print "<br/>";
echo "Data Deleted!";
$result =mysql_query("SELECT * FROM mytable")
or die(mysql_error());
print "<br/>";
print "<b>User Database</b>";
echo "<table border='1'>";
echo "<tr><th>ID</th><th>Name</th><th>Phone</th><th>Email-ID</th>
</tr>";
while($row = mysql_fetch_array( $result ))
```

```
    {
    // Print out the contents of each row into a table
    echo "<tr><td>";
    echo $row['id'];
    echo "</td><td>";
    echo $row['name'];
    echo "</td><td>";
    echo $row['phone'];
    echo "</td><td>";
    echo $row['emailId'];
    echo "</td></tr>";
}
echo "</table>";
?>
```

Output



Example 5.16.2 *Create a HTML form "result.html" with a text box and a submit button to accept registration number of the student. Write a "result.php" code to check the status of the result from the table to display whether the student has "PASS" or "FAIL" status. Assume that the MYSQL database "my_db" has the table "result_table" with two columns REG_NO and STATUS.*

Solution : Step 1 : Create a database named my_db. Create a table result_table for this database and insert the values in this table. The table is created as follows -

Sr. No.	REG_NO	STATUS
1	101	PASS
2	102	FAIL
3	103	PASS
4	104	FAIL
5	105	PASS

Table 5.16.1

Step 2 : Create an HTML form to accept the registration number; the HTML document is as follows -

```
result.html
<!DOCTYPE html>
<html>
   <head>
      <title> STUDENT RESULT </title>
   </head>
   <body>
      <form name="myform" method="post" action="http://localhost/php-
examples/result.php">
         <input type="text" name="reg_no"/>
         <input type="submit" value="Submit"/>
      </form>
   </body>
</html>
```

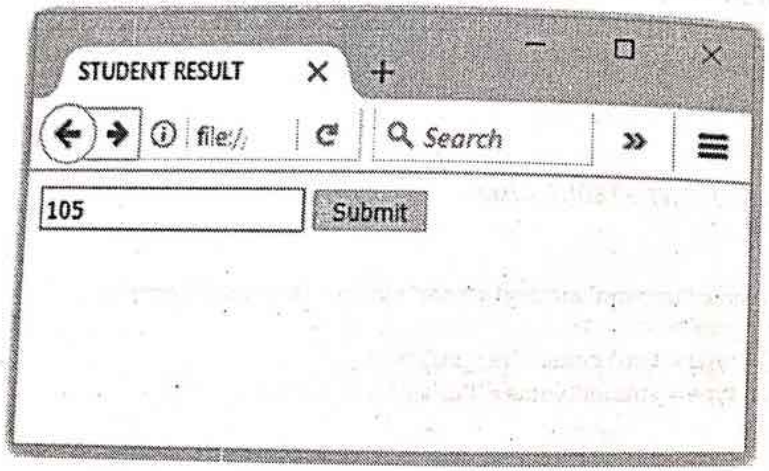
Step 3 : Create a PHP script to accept the registration number. This php script will connect to MYSQL database and the status (PASS or FAIL) of the corresponding registration number will be displayed.

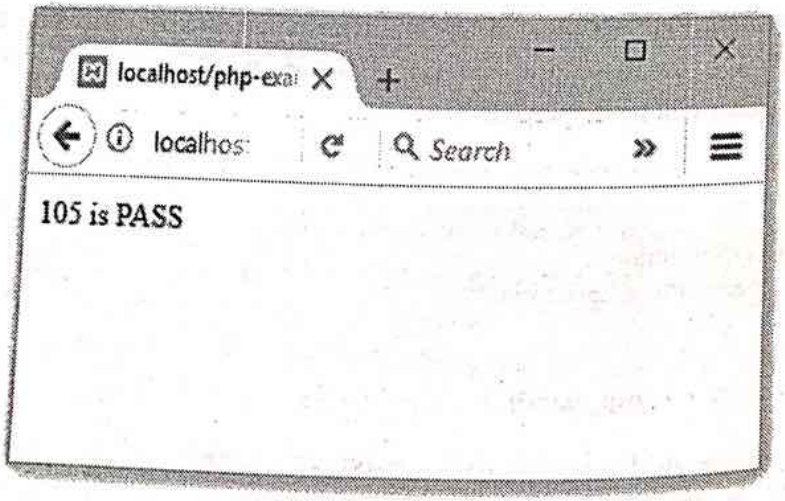
result.php

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost","root","");
if(!$conn)
{
die('error in connection'.mysql_error());
}
mysql_select_db("my_db",$conn); //select the database
//Execution of Query for displaying the data
```

```
$reg_no = intval($_POST['reg_no']);

$result=mysql_query("SELECT REG_NO,STATUS FROM result_table
where REG_NO=$reg_no ");
 while($row = mysql_fetch_array($result))
  {
   echo $row['REG_NO'] . " is " . $row['STATUS'];
   echo "<br />";
  }
mysql_close($conn); //closing the database
?>
```

Step 4 : Load the HTML form created in Step 2 and click the submit button by entering some registration number.





Example 5.16.3 *Write a PHP program to delete a record from result_table. (Refer example 5.16.2)*

Solution :

```
<?php
// Make a MySQL Connection
$conn=mysql_connect("localhost","root","");
if(!$conn)
{
die('error in connection'.mysql_error());
}
mysql_select_db("my_db",$conn); //select the database
//Execution of Query for displaying the data
$reg_no = intval($_POST['reg_no']);

$result=mysql_query("DELETE FROM result_table where REG_NO=$reg_no");
 while($row = mysql_fetch_array($result))
 {
  echo $row['REG_NO'] . " is " . $row['STATUS'];        //Each record will be displayed line by line

  echo "<br />";
 }

mysql_close($conn); //closing the database
?>
```

Example 5.16.4 *Write a user defined function 'CalculateInterest' using PHP to find the simple interest to be paid for a loan amount. Read the loan amount, the number of years and rate of interest from a database table called LOANDETAILS having three fields AMT, YEARS, and RATE, and calculate the interest using the user defined function.*

Solution : Step 1 : Create a database table named LOANDETAILS having three fields AMT, YEARS and RATE. Insert the values in it. The sample table will be as follows -

AMT	YEARS	RATE
10000	5	6.9

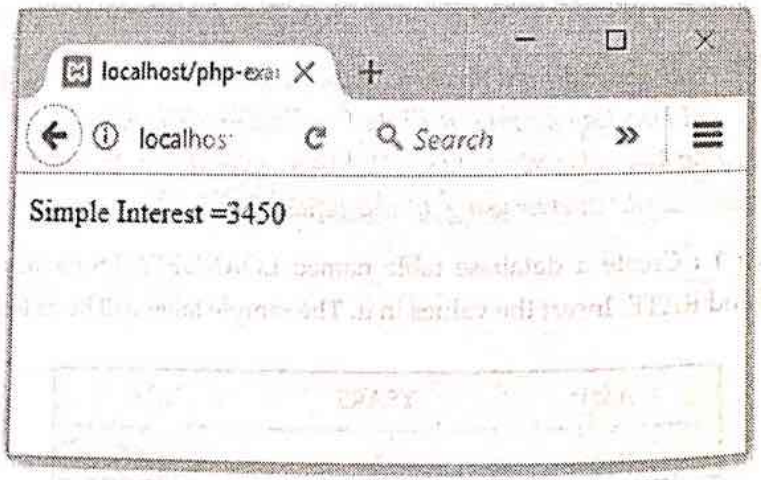
Table 5.16.2

Step 2 : The PHP code for calculating the simple interest the above values in a function will be as follows -

Interest.php

```
<?php
function CalculateInterest()
{
    // Make a MySQL Connection
    $conn=mysql_connect("localhost","root","");
    if(!$conn)
    {
        die('error in connection'.mysql_error());
    }
    mysql_select_db("my_db",$conn); //select the database
    //Execution of Query for displaying the data
    $result=mysql_query("SELECT * FROM LOANDETAILS");
    $row = mysql_fetch_array($result);
    $amount= $row['AMT'];
    $rate= $row['RATE'];
    $years= $row['YEARS'];
    $interest=($amount * $rate *$years)/100;
    mysql_close($conn); //closing the database
    return $interest;
}
print "Simple Interest =".CalculateInterest();
?>
```

Output



Review Question

1. *Identify and explain steps involved in connecting to MySQL with PHP.*
   SPPU : May-18, Marks 6, Dec.-19, Marks 8

5.17 WAP

- Wireless Application Protocol (WAP) is a set of standards designed to extend Internet services to mobile phones and Personal Digital Assistants (PDA).
- A WAP browser is a web browser for mobile devices such as mobile phones that uses the protocol.
- The mobile network gets connected to the Internet through **WAP Gateway**. The function of WAP gateway is to convert the **WAP request** to a **WEB request** or a WEB request to a WAP request.
- The mobile phones act as a client when gets connected to the internet. The demanded web pages can be displayed on the mobile phones if it is WAP enabled.
- The Web pages are normally written in HTML language but the web pages written in HTML gets loaded very slowly on the mobile phones. Hence the web pages for mobile phones are written in the language called **Wireless Markup Language (WML)**
- The WAP model follows the OSI model and is represented by Fig. 5.17.1.

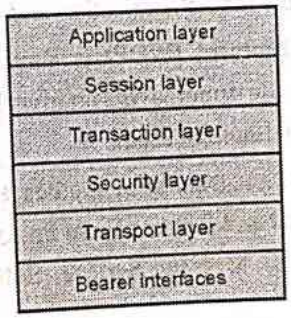


Fig. 5.17.1 WAP model

Application Layer

- The WAP's application layer is the Wireless Application Environment (WAE).
- The WAE supports WAP application development using the Wireless Markup Language (WML).
- The WAE also includes the Wireless Telephony Application Interface (WTA) which provides the programming interface to telephones. This allows WAP technology to support for sending messages, initiating calls and other networking capability.

Session Layer

- The session layer makes use of **Wireless Session Protocol** (WSP).
- This protocol is similar to HTTP. But HTTP is not efficient for displaying the web pages on mobile devices, WAP makes uses of WSP.
- The WSP conserves the bandwidth on the wireless links.
- The HTTP works with textual data whereas the WSP work with binary data.
- It supports for both **connection-oriented** and **connectionless** sessions.

Transaction, Security and Transport Layer

There are three different protocols used in these layers.

- **Wireless Transaction Protocol (WTP)** provides transaction level services. These services are used for both reliable and unreliable transports. If packets gets dropped then WTP performs the retransmission of packets. Similarly it avoids transmission of duplicate copies of packets.
- **Wireless Transaction Layer Security (WTLS)** is just similar to Secured Socket Layer (SSL). It performs the authentication and encryption functionality.
- **Wireless Datagram Protocol (WDP)** is similar to User Datagram Protocol (UDP). WDP offers to the upper layers an invisible interface independent of the underlying network technology used.
- At the lowest layer of the stack, there are **bearer** services. WAP supports dial-up networking using IP and Point to Point Protocol (PPP) as a bearer service. It also supports for Short Message Service (SMS) and General Packet Radio Service (GPRS).

5.18 WML

- The topmost layer in the WAP architecture is made up of WAE (Wireless Application Environment), which consists of WML and WML scripting language.
- WML stands for Wireless Markup Language.
- The role of WML is the same as that of HTML in web applications. WAP sites are written in WML, while web sites are written in HTML.
- WML is basically the web language for making sites on mobile phones.
- WML is an application of XML, which is defined in a document-type definition.

How to write WML ?

- The WML is a scripting language just similar to HTML. The web script can be written in Notepad and executed on WAP simulator.
- The extension to the WML file is .wml.
- I have used WinWap for executing the WML scripts. The WinWAP is a web browser for WAP made by Winwap Technologies available for Microsoft Windows. Get it downloaded from internet using the site www.winwap.com, if you wish to use WinWap browser for executing the WML scripts.

5.18.1 Cards and Decks

- In HTML, we can navigate from one page to another. But in case of WML you can not move from page to page. The WML contains the very basic unit called **card.**
- WML file contains multiple cards. The multiple cards in a WML file forms a **deck.** User can move from one card to another using the same deck.
- We can put text, images, links, form components and so on in the card.

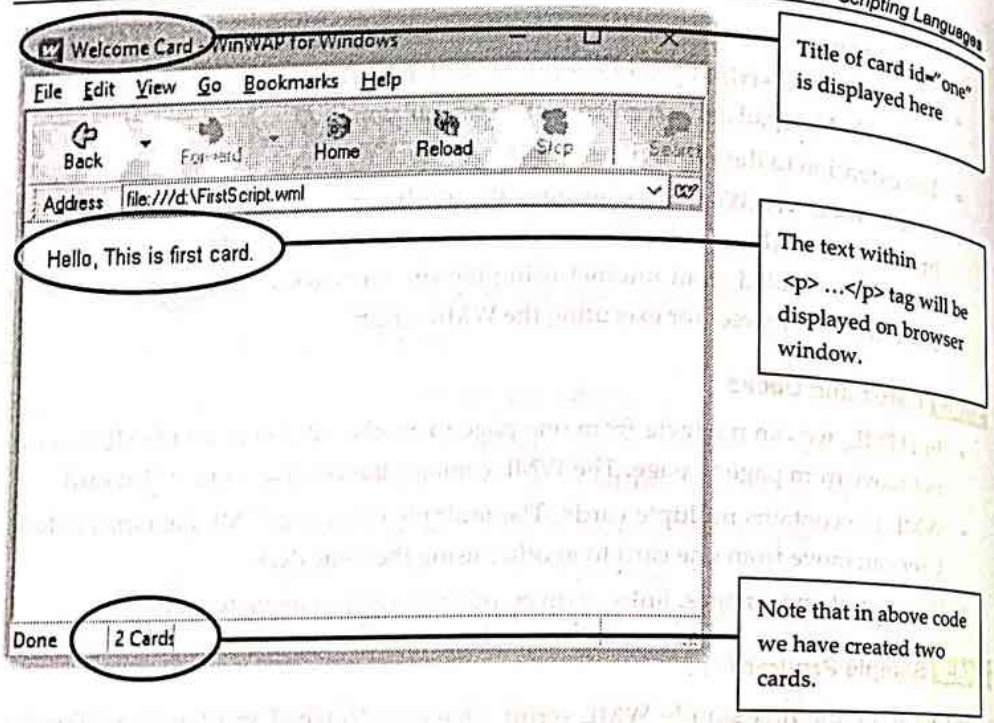
5.18.2 Sample Program

Let us write the first simple WML script. Open the Notepad and type the following code.

WML Document[FirstScript.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card id="one" title="Welcome Card">
<p>
Hello, This is first card.
</p>
</card>
<card id="two" title="GoodBye Card">
<p>
Good bye, this is second card.
</p>
</card>
</wml>
```

Now open the WinWap browser and execute the above code. It will be as follows



Script Explanation :

1) The first line of the script says that it is XML document.
2) The second line indicates that document type definition (DTD). These are the standard lines written in every WML script.
3) Then within <wml> ...</wml> tag the entire code is written.
4) The two cards are created with their own ids. The title tag is used to display the content on the title bard. Within the ... i.e. paragraph tag the text which is to be displayed on the browser window is displayed.

```
<card id="one" title="Welcome Card">

<p>
Hello, This is first card.
</p>
</card>
```

For displaying on title bar for first card

For displaying on browser window for first card


```
<card id="two" title="GoodBye Card">

<p>
Good bye, this is second card.
</p>
</card>
```

For displaying on title bar for second card

For displaying on browser window for second card

Instead of having pages, WAP sites have cards. These are what is displayed on the screen at one time, just like a page. More than one card can be inserted in each WML document. To declare a card, insert the following :

```
<card id="some id" title="some text">
```

Note that : Only one card will be displayed at a time on the web browser.

Closing Tag

Unlike HTML it is important to close every opened tag. Otherwise WML page will not get executed. The card and wml tags are closed as follows

```
</card>
</wml>
```

5.18.3 Comments

Comments are placed inside <!-- --> in WML. This is the same as HTML. For example, the following lines are comments. WAP browsers ignore all comments.

```
<!-- This is a comment in WML -->
<!-- This is a multi-line
  comment -->
```

5.18.4 Paragraph

The tag defines the paragraph of text. The paragraph in WML is always displayed on a new line. The attributes of paragraph are as follows -

Attribute	Value	Purpose
align	Left right center	The text is displayed using left, right or center alignment.
mode	Wrap nowrap	The paragraph lines can be wrapped or not wrapped.
Id	Some user defined ID	The unique ID of the element is specified.

Example

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card id="one" title="Welcome Card">
<p align="left">
Hello, This is first card.
</p>
</card>
<card id="two" title="description Card">
<p align="center">
And, this is second card.
</p>
</card>
<card id="three" title="GoodBye Card">
<p align="right">
Good bye, this is third card.
</p>
</card>
</wml>
```

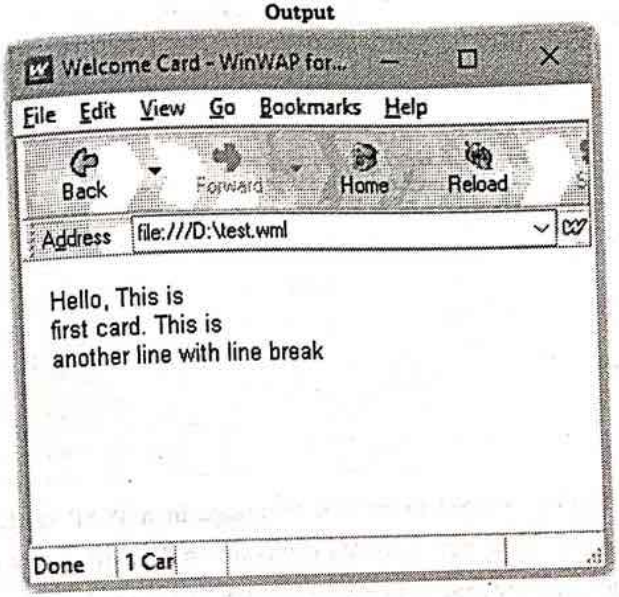
5.18.5 Line Break

The line break in WML is specified using <br/> tag. Note that it is important to close the tag.

Example

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card id="one" title="Welcome Card">
<p>
Hello, This is<br/> first card.
This is <br/> another line with line break
</p>
</card>
</wml>
```


Output



5.18.6 Table

The table tags are just similar to HTML tags. For example - using <table> ... </table> the table can be created. The <tr> </tr>tag is used for creating rows and <td> </td> is used for creating columns. The <table> element is placed within   tag.

Various attributes used along with the <table> tag are

Attribute	Value	Purpose
align	L C or R	The text is displayed using left (L), right (R) or center (C) alignment. If we specify LCR then that means, first column is left allgned, Second column is center aligned and third column is right aligned.
Id	Some user defined ID	The unique ID of the element is specified.

WML Document[TableDemo.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card title="Displaying Tables">
<p>
<table columns="3" align="LCR">
```

```
    <tr>
     <td>One</td>
     <td>Two</td>
     <td>Three</td>
    </tr>
    <tr>
     <td>Four</td>
     <td>Five</td>
     <td>Six</td>
    </tr>
</table>
</p>
</card>
</wml>
```

5.18.7 Image

- The <img> element is used to include an image in a WAP card. WAP-enabled wireless devices only supported the Wireless Bitmap (WBMP) image format.
- WBMP images can only contain two colors : black and white.
- The file extension of WBMP is ".wbmp"
- There are free tools available on internet to convert the jpg or GIF images to .wbmp format. For example Nokia Mobile Internet Toolkit, ImageMagic and so on. Hence to display image on the browser using wml, we must convert our image file to .wbmp format first.
- **Example**

WML Document[ImageDemo.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card title="Displaying Image">
<p>
Good Morning!!!
<img src="flower.wbmp" alt="flower image"/>
</p>
</card>
</wml>
```


Various commonly used attributes of <img> tag are -

Attribute	Value	Purpose
align	Top, middle or bottom	Image is aligned at the top, middle or bottom.
src	Image file path	
alt	Alternative text	The alternative text will be displayed if the image is not getting loaded on the page.
width	% or pixels	The number of pixels or % is specified to mention the width of the image.
height	% or pixels	The number of pixels or % is specified to mention the height of the image.
Id	Some user defined ID	The unique ID of the element is specified.

5.18.8 Hyperlinks

The hyperlink can be used in WML using the anchor tag.

In WML we can use <anchor>, <a> tag to create anchor link.

Various attributes of **<a> tag** are as given below.

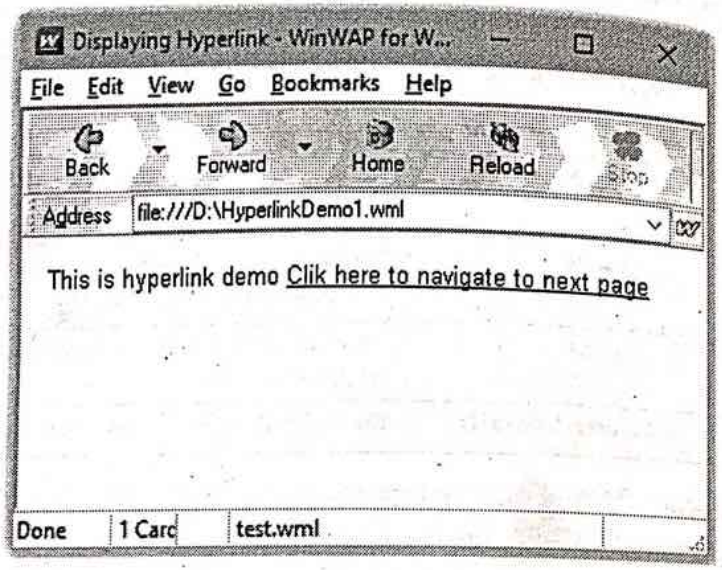
Attribute	Value	Purpose
href	URL or address of the file	The specified URL is linked with page and can be opened up on clicking the link.
title	String	The text for identifying the link is specified
Id	Some user defined ID	The unique ID of the element is specified.

Example

WML Document[HyperlinkDemo1.wm]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card title="Displaying Hyperlink">
 <p> This is hyperlink demo
   <a href="test.wml">Clik here to navigate to next page</a>
 </p>
</card>
</wml>
```

Output



- Similarly we can use **<anchor> tag** for using hyperlink in WML document. This tag is basically used along with <go/>, <refresh> or <prev/> tag.
- **Example**

WML Document[HyperlinkDemo2.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card title="Displaying Hyperlink">
<p> This is hyperlink demo
   <anchor>
      <go href="test.wml"/>
   </anchor>
</p>
<p>
<anchor>
<prev/>
</anchor>
</card>
</wml>
```

Using <prev/> tag we can go to Back page

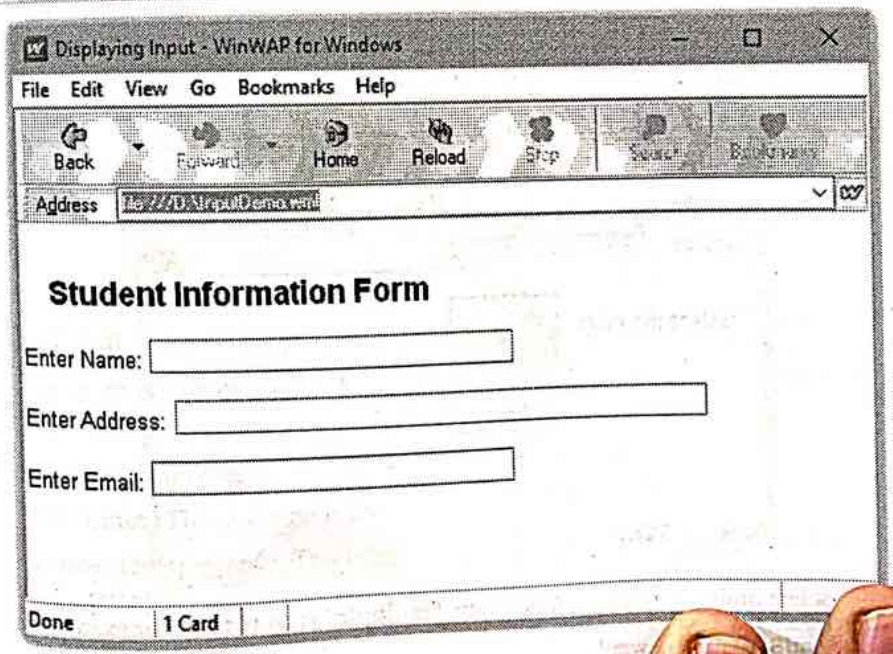
5.18.9 Inputs

The **<input/>** tag allows the user to enter alphanumeric characters such as name, age, address, phone number, email address and so on.

For example -

WML Document[InputDemo.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card title="Displaying Input">
<p>
   <h2>Student Information Form</h2>
   Enter Name: <input name="name" size="20"/><br/><br/>
   Enter Address: <input name="address" size="30"/><br/><br/>
   Enter Email: <input name="email" size="20"/><br/>
</p>
</card>
</wml>
```



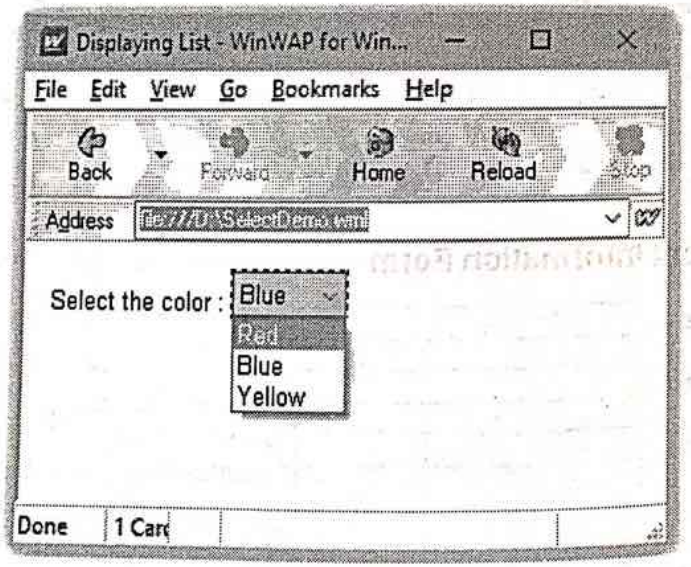
5.18.10 Select

The <select>...</select> WML elements are used to define a selection list and the <option>...</option> tags are used to define an item in a selection list.

WML Document[Select.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card title="Displaying List">
<p> Select the color :
 <select>
  <option value="red">Red</option>
  <option value="blue">Blue</option>
  <option value="yellow">Yellow</option>
 </select>
</p>
</card>
</wml>
```

Output



You can select multiple value if the attribute **"multiple"** is set to true. For example

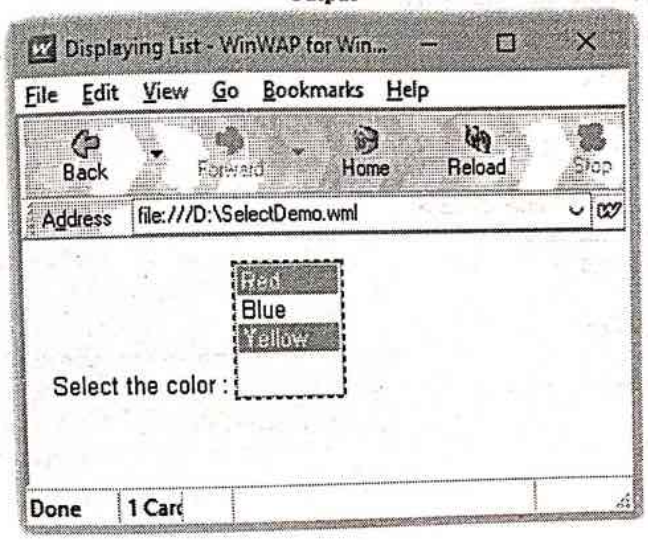
WML Document[SelectDemo.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
```



```
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card title="Displaying List">
<p> Select the color :
<select multiple="true">
<option value="red">Red</option>
<option value="blue">Blue</option>
<option value="yellow">Yellow</option>
</select>
</p>
</card>
</wml>
```

Output



5.18.11 Event

- The event is supported in WML using <onevent></onevent> tag. We can specify the action to be taken whenever an event occurs.
- There are four types of events that are supported in WML -

  (1) **ontimer :** This event occurs when a timer expires.

  (2) **onenterbackward :** This event occurs when a user goes back to a card through the WAP browser's URL history.

  (3) **onenterforward :** This event occurs when a user goes to a card in the forward direction. For example, if a user goes to a card by entering its URL directly in the WAP browser, the *onenterforward* event will be triggered.

(4) **onpick** : This event occurs when an item of a selection list is selected or deselected.

- The syntax of specifying the event tag is as follows -

```
<onevent type="event_type">
  Task to be performed when event occurs
</onevent>
```

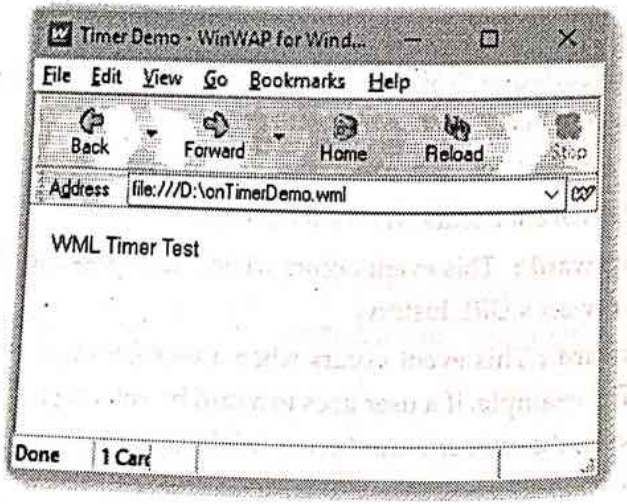
Example using ontimer Event :

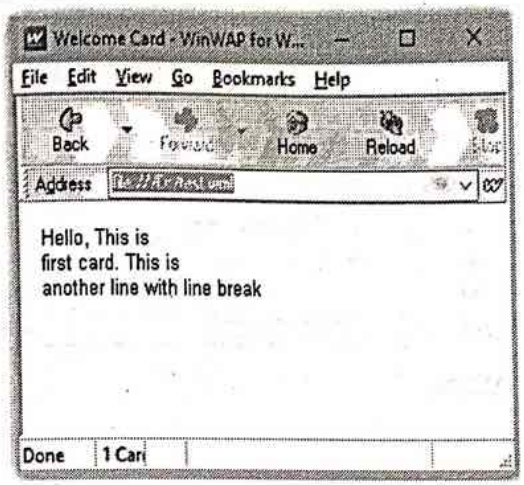
Using ontimer event, we can load the specific web page after some desired amount of time automatically. User need not have to click any hyper link in this case. Following script loads the **test.wml** web page after 3 second. The timer value is specified using **value** attribute and the time unit is 1/10 seconds.

WML Document[onTimerDemo.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
 <card id="card1" title="Timer Demo">
  <onevent type="ontimer">
   <go href="test.wml"/>
  </onevent>
  <timer value="30"/>
  <p>
   WML Timer Test
  </p>
 </card>
</wml>
```

Output





After 3 seconds

Example using onenterbackward Event :

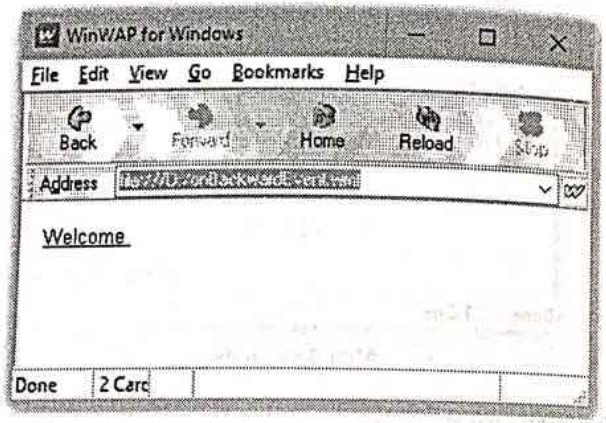
The onenterbackward event is triggered if a user goes back to a previous card through the WAP browser's URL history, and the WML code placed in the event handler will be executed.

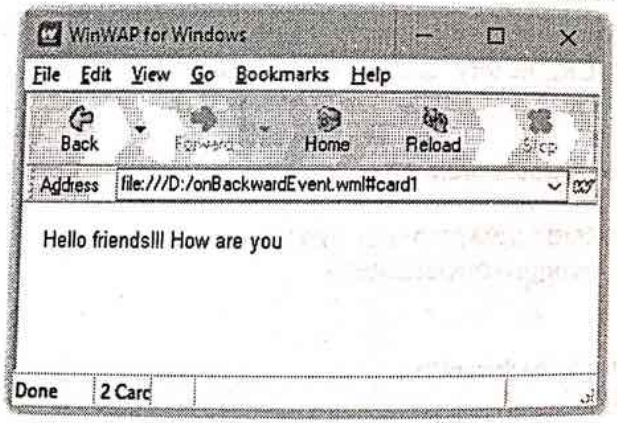
WML Document[onenterBackward.wml]

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<template>
 <onevent type="onenterbackward">
  <go href="#card1"/>
 </onevent>
</template>
<card id="card2">
<p>
<anchor>
<prev/>
Welcome
</anchor>
</p>
</card>
<card id="card1">
<p>
Hello friends!!! How are you
```

```
</p>
</card>
</wml>
```

Output





Example using onenterforward

The onenterforward event is triggered when a user goes to a card in the forward direction. For example, if you go to a card by entering the URL directly or by following an anchor link of which the action is <go>, the onenterforward event will be triggered and the WML code associated with the event will be executed.

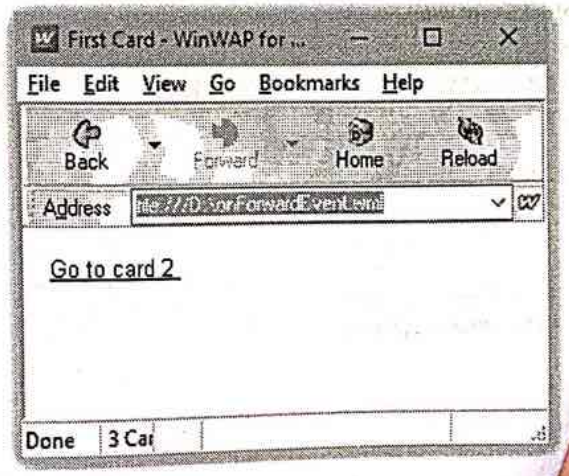
WML Document[onForwardEvent.wml]

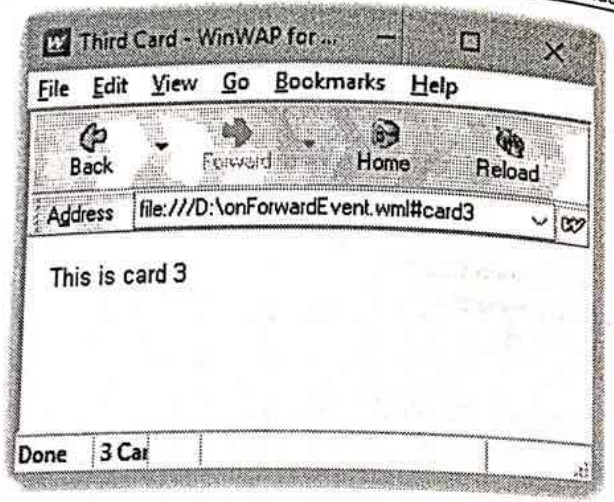
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
```



```
<card id="card1" title="First Card">
<p>
 <anchor>
  <go href="#card2"/>
  Go to card 2
 </anchor>
</p>
</card>
<card id="card2" title="Second Card">
<onevent type="onenterforward">
 <go href="#card3"/>
</onevent>
<p>
 This is card 2
</p>
</card>
<card id="card3" title="Third Card">
<p>
This is card 3
</p>
</card>
</wml>
```

Output





Note that on clicking the link Go to card2 the control will move directly to card3, because in card2 the **onenterforward** event will occur and control will be transferred directly to card3.

Example using onpick

When an item from the selection list is selected or deselected then this even occurs.

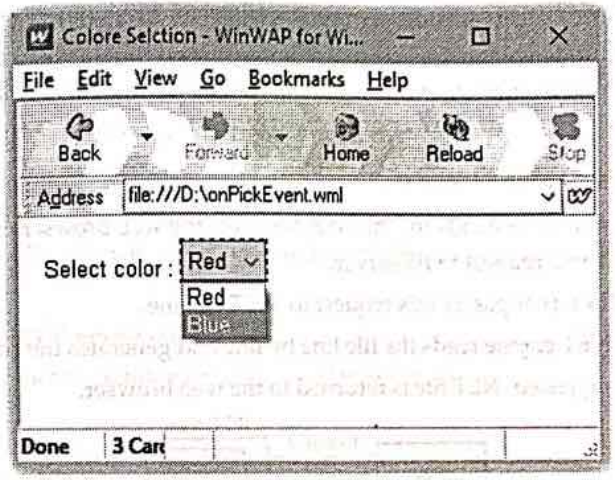
WML Document[onPickEvent.wml]

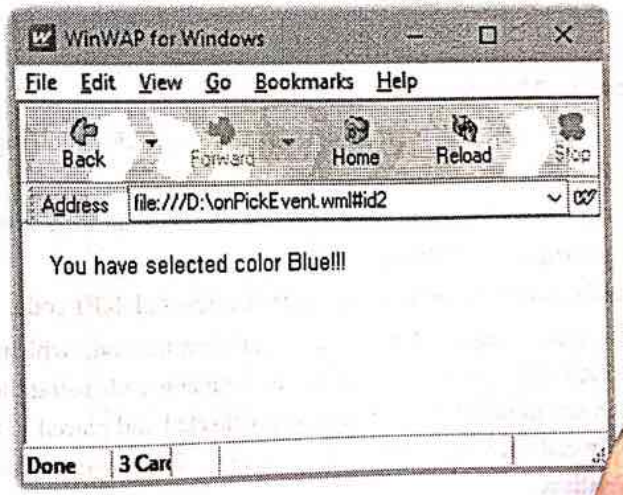
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card id="MainCard" title="Color Selection">
<p>
 Select color :
 <select>
  <option onpick="#id1">Red</option>
  <option onpick="#id2">Blue</option>
 </select>
</p>
</card>
<card id="id1">
<p>
You have selected color RED!!!
</p>
</card>
<card id="id2">
```



```
<p>
You have selected color Blue!!!
</p>
</card>
</wml>
```

Output





Part - II : ASP.NET

5.19 Introduction to ASP.NET

Active Server Pages (ASP) is a server side scripting language. It is a part of **Internet Information Server** (IIS). This is basically a product of Microsoft.

The ASP.NET is a server side scripting technology offered by Microsoft. It is a powerful scripting language for creating dynamic and interactive web pages. The ASP.NET is next generation ASP but it is entirely new technology. The ASP.NET runs under IIS server.

5.19.1 How does ASP.NET Works ?

Step 1 : When user demands for the ASP.NET file, the web browser of the client's PC sends this request to IIS server.

Step 2 : The IIS server passes this request to .NET Engine.

Step 3 : The .NET engine reads the file line by line and generates the script in a file.

Step 4 : The requested .NET file is returned to the web browser.

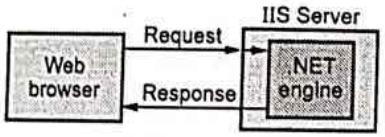


Fig. 5.19.1

5.19.2 Features of ASP.NET

- ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language.
- ASP.NET pages are called **web forms.** These web forms have the extensions **.aspx.**
- These files contain static XHTML markup.
- The static and dynamic content can be inserted in the ASP.NET code.
- The code for appearance can be separated out from the code which is responsible for responding the events. Microsoft treats the dynamic code using the **code behind the model.** In this model the dynamic code is collected and placed in a separate file. Such a file typically has the names like xyz.aspx.cs. This separates out the **contents** from **presentation.**
- ASP.NET supports various types of controls. It also allows the programmer to build the **custom controls.** These controls can be built in a .DLL file and can be loaded whenever required.

- ASP.NET supports some advanced techniques such as cookies, sessions handling, query strings.

5.20 Overview of the .NET Framework

- The .NET framework is a software development platform developed by Microsoft. It is designed for building and running windows applications.
- The first version of .NET framework was released in the year 2002. This version was known as .NET framework 1.0.
- The .NET framework is used to create both form based and web based applications.
- The .NET framework is also useful in building the web services.
- It supports various programming languages like C#, VB.NET, Visual Basic and many more. Developers can choose and select the language to develop the required application.

Architecture of .NET Framework

The .NET framework architecture consists of three major components -

1. CLR
2. Library
3. Language

Let us discuss them in detail.

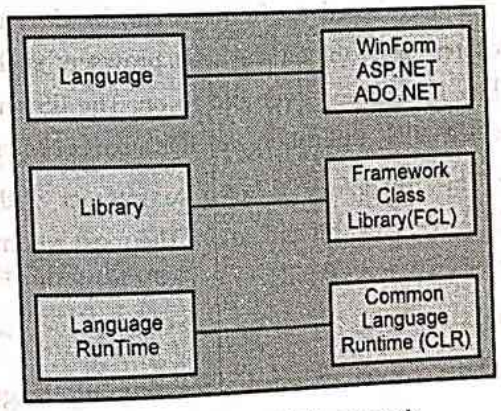


Fig. 5.20.1 The .NET framework

1. CLR

- The CLR stands for Common Language Runtime. It is a platform on which the .NET programs are executed. Any .NET programming language can be executed by this platform.

- Just in Time (JIT) is a part of CLR. It converts the source code into native intermediate code and then into machine instructions.
- The CLR is also responsible for performing some other tasks such as exception handling, memory management and garbage collection.
- The important features of CLR are interoperability, security and portability.

2. Framework Class Library (FCL)

- It is a standard library which is actually a collection of various classes, interfaces, and namespaces that can be used to build the application.
- It is the key component of .NET framework that provides the core functionalities of .NET.
- Most of the methods of classes are split either into System.* or Microsoft.* namespaces. A namespace is a logical separation of methods.

3. Language

- The .NET supports many languages. Developer can choose the desired language for development of required application.
- Following are the categories of various types of applications that can be developed using .NET
  - **WinForms :** This is useful for developing form based applications.
  - **ASP.NET :** The web based applications can be developed using ASP.NET. These applications will run on the web browsers such as Google Chrome, Mozilla FireFox, Internet Explorer and so on. The IIS server is responsible for executing the ASP.NET application which is a part of Microsoft component.
  - **ADO.NET :** This technology is used to interact with the databases like Microsoft SQL servers, Oracle and so on. The most commonly used languages for this category is C# and VB.NET.

5.21 Overview of C#

- The C# is a modern, **object oriented programming** language developed by Microsoft.
- C# was developed by Anders Hejlsberg during the development of .NET framework.

- C# is designed for **Common Language Infrastructure (CLI),** which consists of the executable code and runtime environment that allows use of various high-level languages to be used on different computer platforms and architectures.
- Various **features** of C# are
  1. Object Oriented
  2. Standard Library
  3. Automatic Garbage Collection
  4. Boolean Condition
  5. Properties and Events
  6. Simple Multithreading
  7. Integration with Windows.

Required Environment

For executing the C# programs we need Visual Studio 2019. Install the Microsoft Visual Studio 2019 edition.

5.21.1 Parts of C# Program

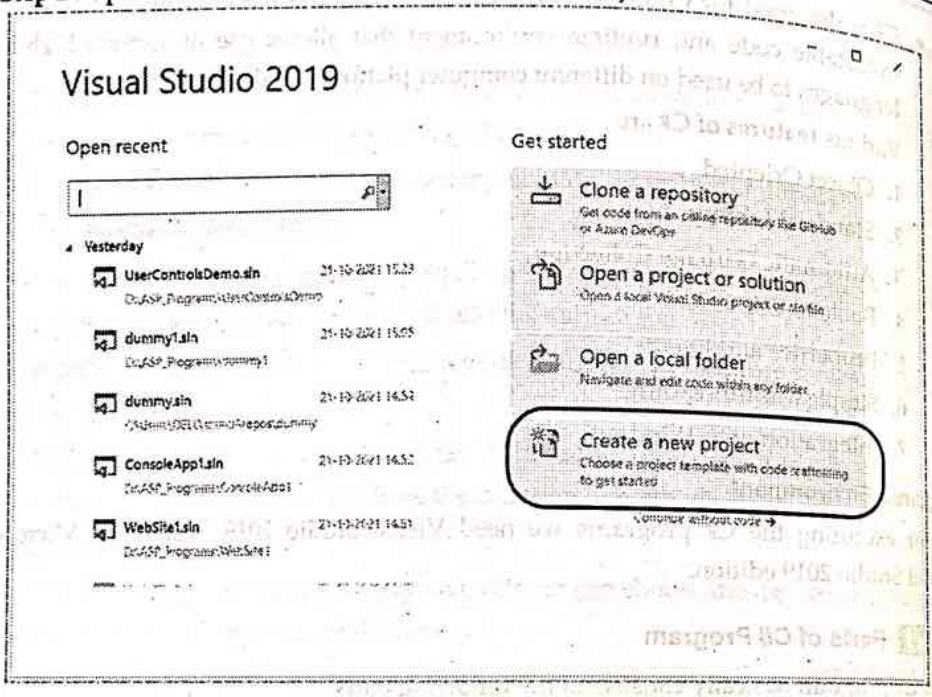
A C# program basically consists of the following parts :

- Namespace declaration
- A class
- Class methods
- Class attributes
- A Main method
- Statements and Expressions
- Comments

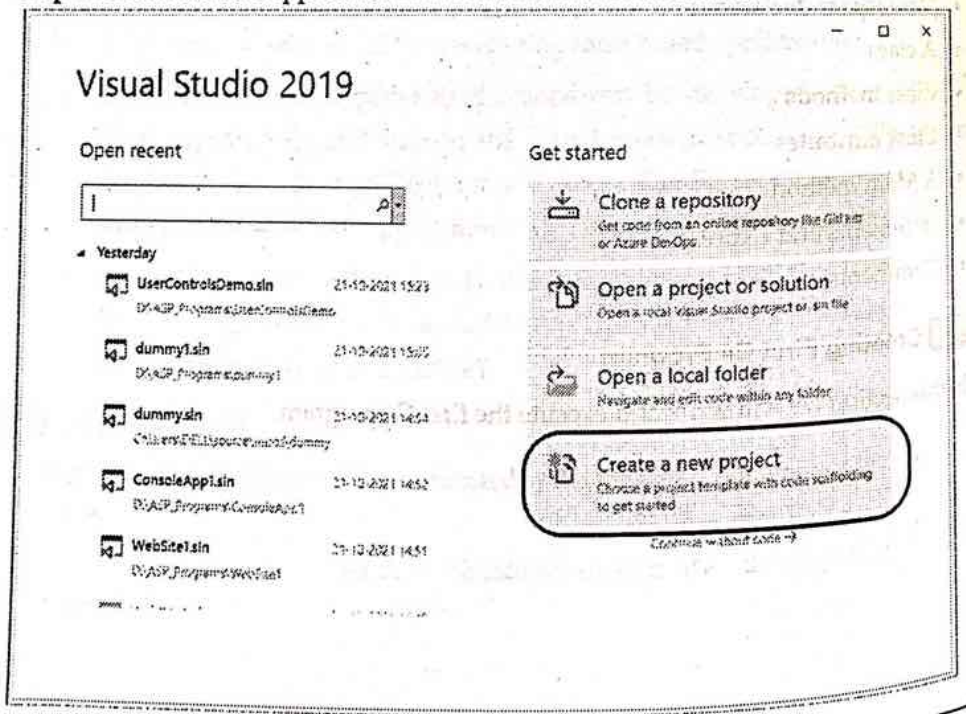
5.21.2 Creating First C# Program

In this section we will write and execute the first C# program.

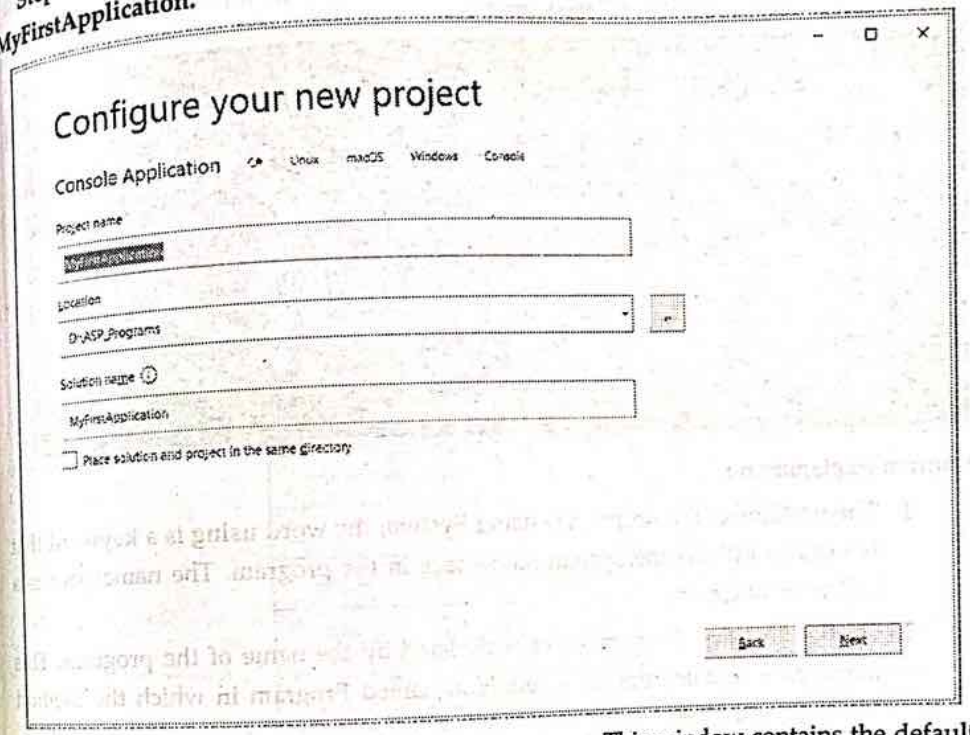
Step 1 : Open Visual Studio. Click on 'Create a new Project'



Step 2 : Select Console application and then click on Next button.



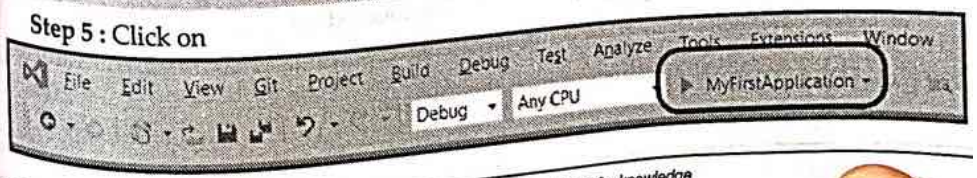
Step 3 : Give any suitable name to your project. I have given the name as **MyFirstApplication.**



Step 4 : The source code window will appear now. This window contains the default C# code. We can write our first C# code as follows -

```
using System;

namespace MyFirstApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Welcome User!!!!");
    }
  }
}
```

Step 5 : Click on



The console window will appear and display the output of the above program.

```
Select Microsoft Visual Studio Debug Console
Welcome User!!!!

D:\ASP_Programs\MyFirstApplication\MyFirstApplication\bin\Debug\netcorea
pp3.1\MyFirstApplication.exe (process 13776) exited with code 0.
To automatically close the console when debugging stops, enable Tools->O
ptions->Debugging->Automatically close the console when debugging stops.

Press any key to close this window . . .
```

Program Explanation :

1. The first line of the program is **using System;** the word **using** is a keyword that is used to include the **System** namespace in the program. The namespace is a collection of classes.
2. On the next line the namespace is declared by the name of the program. This namespace contains the class definition named **Program** in which the method Method() is defined.
3. This is the only method defined in the class. This is the **entry point** for all C# programs. The **Main** method states what the class will do when executed.
4. Inside the Main method we have written **Console.writeln** statement. The Console is the class and we are invoking the **writeln** method of this class. By this statement we can display the desired message on the output screen.

Comment Statement

The comments in C# are similar to C++ comment statements. The // and /* ... */ are used as comments in C#.

Points to Remember

1. C# is case sensitive
2. The execution of the program starts at Main method.
3. Every statement in C# must be terminated by semicolon.
4. The file name can be different from the class name.

5.21.3 Variables

Variables are used for storing values. The syntax for variables is as follows -

```
Data type variable = value
```

For example -

```
int age = 20;
```

5.21.4 Data Types

The data types are used for specifying the type of the variable.

Following table shows various data types used -

Data Types	Memory Size
char	1 byte
signed char	1 byte
unsigned char	1 byte
short	2 byte
signed short	2 byte
unsigned short	2 byte
int	4 byte
signed int	4 byte
unsigned int	4 byte
long	8 byte
signed long	8 byte
unsigned long	8 byte
float	4 byte
double	8 byte
decimal	16 byte

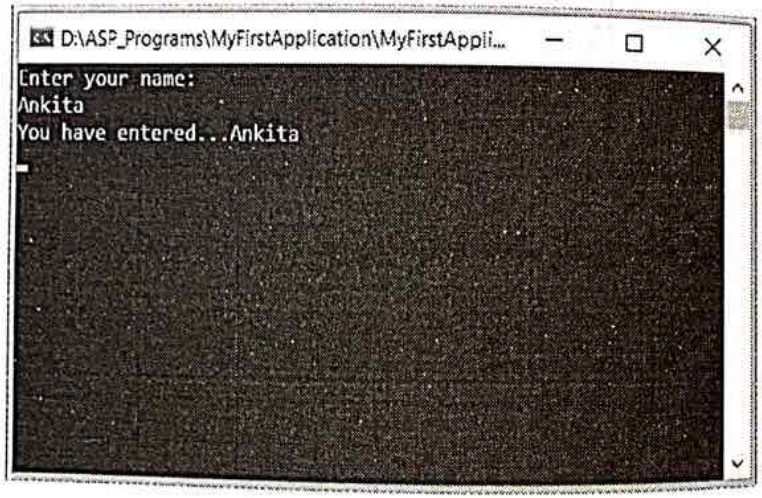
5.21.5 Input and Output

The user input is given by **Console.ReadLine()** and the output is displayed using **Console.WriteLine()** method. For example :

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;

namespace MyFirstApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Enter your name: ");
      String name = Console.ReadLine();
      Console.WriteLine("You have entered..."+name);
      Console.ReadKey();
    }
  }
}
```

Output



5.21.6 Operators

The operators are the symbols that are used to perform various operations.

Various operators that are used in Java are enlisted in following table -

Type	Operator	Meaning	Example
Arithmetic	+	Addition or unary plus	c=a+b
	-	Subtraction or unary minus	d=-a
	*	Multiplication	c=a*b

	/	Division	c=a/b
	%	Mod	c=a%b
Relational	<	Less than	a<4
	>	Greater than	b>10
	<=	Less than equal to	b<=10
	>=	Greater than equal to	a>=5
	==	Equal to	x==100
	!=	Not equal to	m!=8
Logical	&&	And operator	0&&1
	||	Or operator	0 ||1
Assignment	=	is assigned to	a=5
Increment	++	Increment by one	++i or i++
Decrement	--	Decrement by one	--k or k--

Conditional operator

The conditional operator is "?" The syntax of conditional operator is

```
Condition?expression1:expression2
```

where expression1 denotes the true condition and expression2 denotes false condition.

For example :

```
a>b?true:false
```

This means that if a is greater than b then the expression will return the value true otherwise it will return false.

Program for Demonstrating Arithmetic Operations

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyFirstApplication
{
  class Program
  {
    static void Main(string[] args)
    {
```

```
        Console.WriteLine("Arithmetic Operations");
        int a = 20;
        int b = 10;
        int c;
        Console.WriteLine("a = " + a);
        Console.WriteLine("b = " + b);
        c = a + b;
        Console.WriteLine("Addition of two numbers..."+c);
        c = a - b;
        Console.WriteLine("Subtraction of two numbers..." + c);
        c = a * b;
        Console.WriteLine("Multiplication of two numbers..." + c);
        c = a / b;
        Console.WriteLine("Division of two numbers..." + c);

        Console.ReadKey();
    }
  }
}
```

Output

```
D:\ASP_Programs\MyFirstApplication\MyFirst...
Arithmetic Operations
a = 20
b = 10
Addition of two numbers...30
Subtraction of two numbers...10
Multiplication of two numbers...200
Division of two numbers...2
```

Program Demonstrating Relational Operators

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```



```
namespace MyFirstApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("\n Performing Relational operations ");
      int a = 10, b = 20, c = 10;
      Console.WriteLine("a= " + a);
      Console.WriteLine("b= " + b);
      Console.WriteLine("c= " + c);
      Console.WriteLine("\n The a<b is " + (a < b));
      Console.WriteLine("\n The a>b is " + (a > b));
      Console.WriteLine("\n The a==c is " + (a == c));
      Console.WriteLine("\n The a<=b is " + (a <= b));
      Console.WriteLine("\n The a>=c is " + (a >= c));
      Console.WriteLine("\n The a!=b is " + (a != b));
      Console.ReadKey();
    }
  }
}
```

Output

```
D:\ASP_Programs\MyFirstApplication\MyFirst...
 Performing Relational operations
a= 10
b= 20
c= 10

 The a<b is True

 The a>b is False

 The a==c is True

 The a<=b is True

 The a>=c is True

 The a!=b is True
```

Program Demonstrating Logical Operators

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyFirstApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("\n Performing Logical operations ");
      Boolean oper1, oper2;
      oper1 = true;
      oper2 = false;
      Boolean ans1, ans2;
      ans1 = oper1 & oper2;
      ans2 = oper1 | oper2;
      Console.WriteLine("The oper1 is: " + oper1);
      Console.WriteLine("The oper2 is: " + oper2);
      Console.WriteLine("The oper1&oper2 is: " + ans1);
      Console.WriteLine("The oper1|oper2 is: " + ans2);
      Console.ReadKey();
    }
  }
}
```

Output

```
D:\ASP_Programs\MyFirstApplication\MyFirstA...

 Performing Logical operations
The oper1 is: True
The oper2 is: False
The oper1&oper2 is: False
The oper1|oper2 is: True
```


Program Demonstrating Conditional Operator

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyFirstApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      Console.WriteLine("\n Performing Conditional operations ");
      int a, b, c;
      a = 10;
      b = 20;
      Console.WriteLine("a = " + a);
      Console.WriteLine("b = " + b);
      c = a > b ? a : b;
      Console.WriteLine(c + " is greatest");
      Console.ReadKey();
    }
  }
}
```

Output

```
D:\ASP_Programs\MyFirstApplication\MyFirs...

 Performing Conditional operations
a = 10
b = 20
20 is greatest
```

5.21.7 Control Structure

Various control structures are -

1. if statement
2. while statement
3. do-while statement
4. switch case statement
5. for loop

Let us discuss these control structures with the help of simple examples.

1. If statement

- There are two types of if statements - simple if statement and compound if statement. The simple if statement is a kind of if statement which is followed by single statement. The compound if statement is a kind of if statement for which a group of statements are followed. These groups of statements are enclosed within the curly brackets.
- If statement can also be accompanied by the else part.
- Following table illustrates various forms of if statement.

Type of statement	Syntax	Example
simple if	if(condition) statement	if(a<b) Console.WriteLine("a is smaller than b");

compound if	if(condition)
	{
	statement 1;
	...
	...
	}
		if(a<b)
		{
		Console.WriteLine("a is smaller than b");
		Console.WriteLine("b is larger than a");
		}

if...else	if(condition)
	statement;
	else
	statement;
		if(a<b)
		Console.WriteLine("a is smaller than b");
		else
		Console.WriteLine("a is larger than b");

compound if...else	if(condition)
	{
	statement 1;
	...
	...
	}
	else
	{
	statement 1;
	...
	...
	}
		if(a<b)
		{
		Console.WriteLine("a is smaller than b");
		Console.WriteLine("b is larger than a");
		}
		else
		{
		Console.WriteLine("a is larger than b");
		Console.WriteLine("b is smaller than a");
		}

if...else if	if(condition)
	{
	statement 1;
	...
	...
	}
	else if(condition)
	{
	statement 1;
	...
	...
	}
	else
	{
	statement 1;
	...
	...
	}
		if(a<b)
		Console.WriteLine("a is smaller than b");
		else if(a<c)
		Console.WriteLine("a is smaller than b");
		else
		Console.WriteLine("a is larger than b and c");

2. while statement

The while statement is executing repeatedly until the condition is false. The while statement can be simple while or compound while. Following table illustrates the forms of while statements -

Type of statement	Syntax	Example
simple while	while(condition) statement	while(a<10) Console.WriteLine("a is smaller than 10");

compound while	while(condition)
	{
	statement 1;
	...
	...
	}
		while(a<10)
		{
		Console.WriteLine("a is less than b");
		a++;
		}

3. do..while

- The do...while statement is used for repeated execution. The difference between do while and while statement is that, in while statement the condition is checked

before executing any statement whereas in do while statements the statement is executed first and then the condition is tested. There is at least one execution of statements in case of do...while. Following table shows the use of do while statement.

Type of statement	Syntax	Example

do...while	do
	{
	statement 1;
	...
	...
	}while(condition);
		do
		{
		Console.WriteLine("a is less than b");
		a++;
		} while(a<10);

Note that the while condition is terminated by a semicolon.

4. Switch statement

From multiple cases if only one case is to be executed at a time then switch case statement is executed. Following table shows the use of switch case statements.

Type of statement	Syntax	Example

switch ...case	switch(condition)
	{
	case caseno:statements
	break;
	....
	default: statements
	}
		Console.WriteLine("\n Enter choice");
		cin>>choice;
		switch(choice)
		{
		case 1: Console.WriteLine("You have selected 1");
		break;
		case 2: Console.WriteLine("You have selected 2");
		break;
		case 3: Console.WriteLine("You have selected 3");
		break;
		default: Console.WriteLine("good bye");
		}

5. for loop

The for loop is a not statement it is a loop, using which the repeated execution of statements occurs. Following table illustrates the use of for loop -

Loop	Syntax	Example
Simple for loop	for(initialization; termination; step count) statement	for(i=0;i<10;i++) c[i]=a[i]+b[i];

Compound for loop	for(initialization; termination; step count)
	{
	statement 1;
	statement 2;
	...
	}
		for(i=0;i<10;i++)
		{
		for(j=0;j<10;j++)
		c[i][j]=a[i][j]+b[i][j];
		}

5.21.8 The Break and Continue Statements

The break and continue statements are used to alter the normal control flow of the program.

Break statement : The break statement is used to terminate the loops such as for, while, do...while and switch.

Syntax :

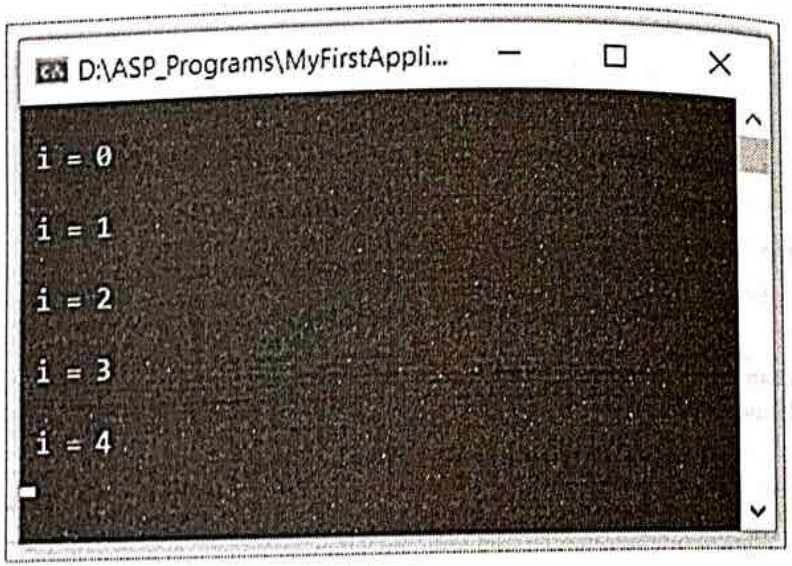
```
break;
```

Example Program :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyFirstApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      int i, count = 5;
      for (i = 0; i < 10; i++)
      {
        if (i == count)
        {
          break;
        }
        Console.WriteLine("\n i = "+i);
      }
```

```
        Console.ReadKey();
    }
  }
}
```

Output



Continue statement : Sometime we need to skip certain test condition within a loop. At that time continue statement is used.

Syntax :

```
continue;
```

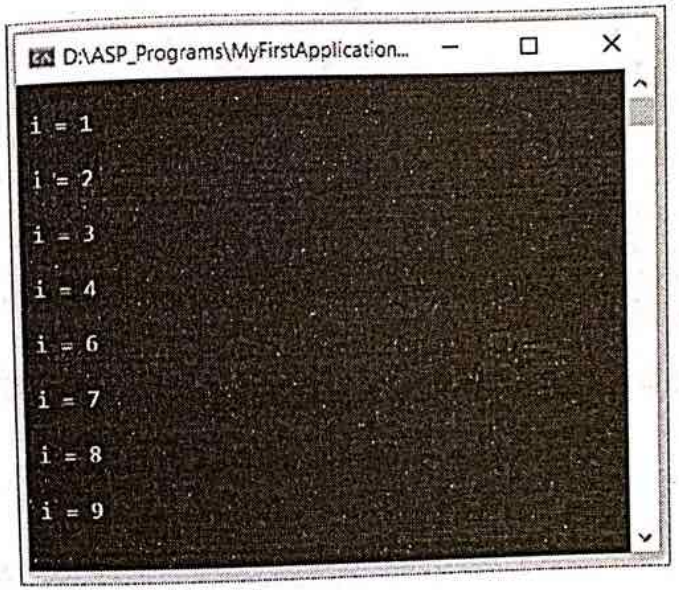
Example Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyFirstApplication
{
  class Program
  {
    static void Main(string[] args)
    {
      int i, count = 5;
      for (i = 0; i < 10; i++)
```



```
      {
        if (i == count)
        {
          continue;
        }
        Console.WriteLine("\n i = " + i);
      }
      Console.ReadKey();
    }
  }
}
```

Output



5.21.9 Class and Object

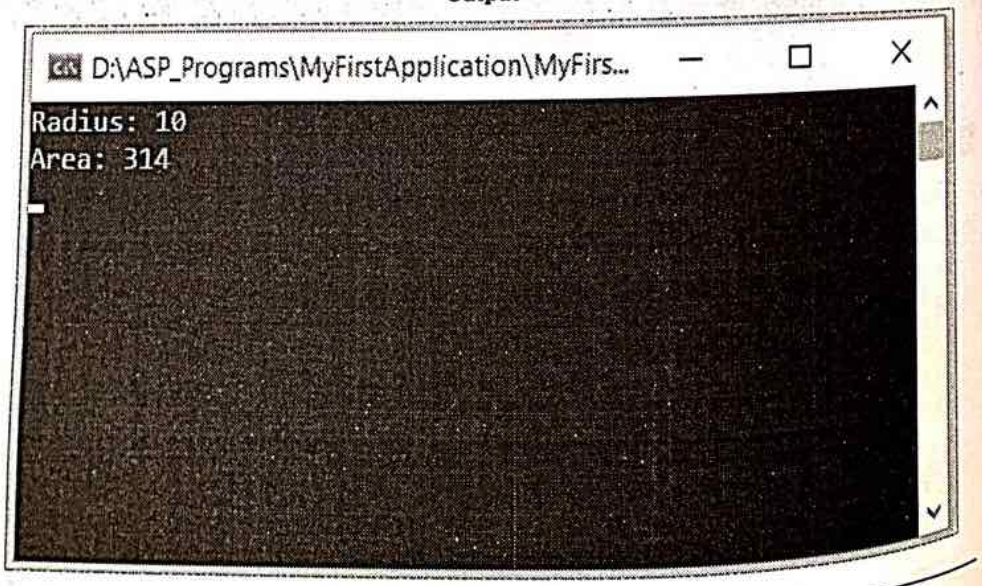
The programming style of C# is just similar to C++. Just similar to C++, you can write the class in C# and can access the methods of that class using the object of the class. Following example illustrates this concept.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication2
{
  class Circle
```

```
  {
    double radius;
    double pi=3.14;
    double a;
    public void getdata()
    {
      radius = 10;
    }
    public void area()
    {
      a = pi * radius * radius;
      Console.WriteLine("Radius: {0}", radius);
      Console.WriteLine("Area: {0}", a);
    }
  }
  class Program
  {
    static void Main(string[] args)
    {
      Circle obj = new Circle();
      obj.getdata();
      obj.area();
      Console.ReadKey();
    }
  }
}
```

Note that while displaying the variables they are displayed using the index{0},{1} and so on.
These two statements can be combined into one statement as follows -
Console.WriteLine("Radius: {0},Area: {1}", radius, a);

Output



5.22 Introduction to ASP.NET

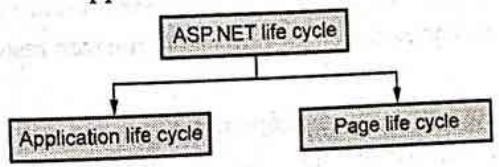
- ASP.NET is a platform for developing web applications. It works on **HTTP protocol** and makes use of HTTP commands.
- The ASP.NET application programs can be written in one of the following languages -
  1. C#
  2. Visual Basic .NET
  3. Jscript
  4. J#
- ASP.NET consists of large number of controls such as textboxes, buttons, labels and checkboxes and so on for developing the web pages. Basically the HTML tags are used to manipulate the code.

5.22.1 ASP.NET Component Model

- The ASP.NET component model provides various building blocks of the ASP.NET pages. It describes the -
  1. Server side **pages** that uses the HTML tags
  2. Server **controls** that are used for building complex web applications.
- In ASP.NET programming the **.aspx page** uses the HTML tags. During execution, the .aspx page is transformed into an **instance of a class** which inherits from the base class Page of the .Net framework.
- In this technology **each ASP.Net page is an object** and all its components i.e., the server-side controls are also objects.

5.22.2 Life Cycle of ASP.NET Application

The life cycle of ASP.NET application is divided into two categories -



Application life cycle

There are mainly two phases in the application life cycle -

1. Request phase

a) User makes a request for accessing particular resource.

b) Browser sends this request to the web server.

c) An object of **ApplicationManager** class is created.

d) An object of the **HostingEnvironment** class is created to provide information regarding the resources.

e) Required items in the application are compiled.

2. Response Phase

a) Response objects are created.

b) There are some application objects of the class **HttpContext, HttpRequest** and **HttpResponse.** These objects are initialized.

c) The request is processed by the **HttpApplication** class. Hence an instance of the **HttpApplication** object is created and assigned to the request.

d) Different events are raised by this **HttpApplication** class for processing the request.

3. Page life cycle

Various stages of page life cycle are -

1. **Page request :** The page request occurs before the page life cycle begins. When the page is requested by a user, ASP.NET determines whether the page needs to be parsed and compiled or whether a cached version of the page can be sent in response without running the page.
2. **Start :** In stage the page properties such as **Request** and **Response** are set. At this stage, the page determines whether the request is postback or it is a new request. It then sets the **IsPostBack** property. At this stage the **UICulture** property is also set.
3. **Initialization :** In this stage controls on the page are available. Each control's **UniqueID** property is set. A master page and themes are set. For a new request postback data is loaded and the control properties are restored to the view-state values.
4. **Load :** At this stage, the control properties are loaded with information obtained from view state and control state.
5. **Postback event handling :** If the request is a postback, then control event handlers are called. Then to validate the controls the **Validate** method is called.
6. **Rendering :** At this stage, the page calls the **Render** method for each control. The output of rendering is written to the **OutputStream** class of the Page's Response property.

7. **Unloading :** The Unload event is raised after the page is fully rendered. At this point, page properties such as Response and Request are unloaded and cleanup is performed.

Page life cycle events

Within each stage of the life cycle of a page, the page raises events. The following table lists the page life-cycle events that are used most commonly.

Page event	Purpose
PreInit	This event is raised after start stage is complete and before initialization stage. It Checks the **IsPostBack** property to determine whether this is the first time the page is being processed. It sets the Master page and theme property dynamically.
Init	This event is raised after all controls have been initialized.
InitComplete	It is raised at the end of the page's initialization stage
PreLoad	It is raised after the page loads view state.
Load	The **Page** object calls the **OnLoad** method. This events is raised for the page loading and then recursively for all the child controls.
LoadComplete	It is raised at the end of the event-handling stage. This event can be handled by overloading the **OnLoadComplete** method
PreRender	When all the controls for the page are created then this event is raised. The **Page** object raises the **PreRender** event on the **Page** object, and then recursively does the same for each child control.
PreRenderComplete	After completion of preRendering stage this event is raised.
Render	This is not an event; instead, at this stage of processing, the **Page** object calls this method on each control that writes out the control's markup to send to the browser
Unload	Raised for each control and then for the page. During the unload stage, the page and its controls have been rendered so further changes to response stream can not be done.
Control Events	These events are used to handle specific control events. For instance Button control has click event.

5.23 ASP.NET Web Form Model

- The ASP.NET web forms make use of events-driven techniques for the interaction of the web application. The browser actually submits the web form to the web server and then server returns the HTML page in response.
- The client side activities are forwarded to the server and the server sends the response to its client. The HTTP protocol plays an important role in this communication.
- The HTTP is a stateless protocol. So ASP.Net helps in storing the information regarding the state of the application. It consists of :
  1. Page state
  2. Session state

The page state is the state of the **client**. That means at this state, the client has the web application page having various input fields on the web form. On the other hand the **session state** is the state in which the information is collected from various pages the user visited or worked with. ASP.Net session state and server side infrastructure keeps track of the information collected globally over a session.

5.24 ASP.NET Controls

Various controls that can be used by the web forms are -

1. Button controls
2. Text boxes and labels
3. Check boxes and Radio buttons
4. List controls
5. List items

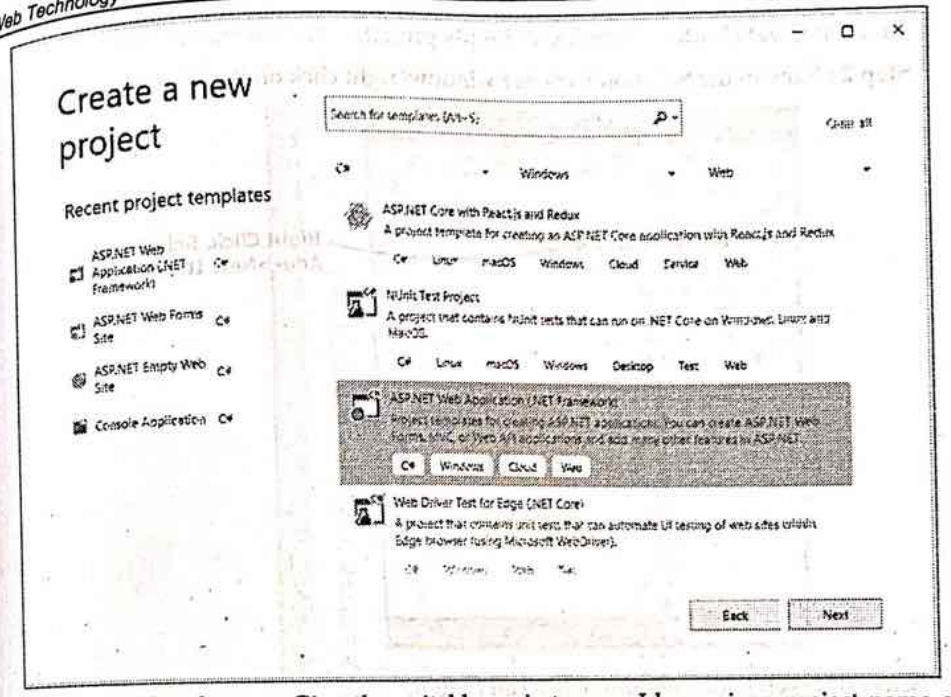
   and so on.

We will discuss some most commonly used controls for designing the web forms.

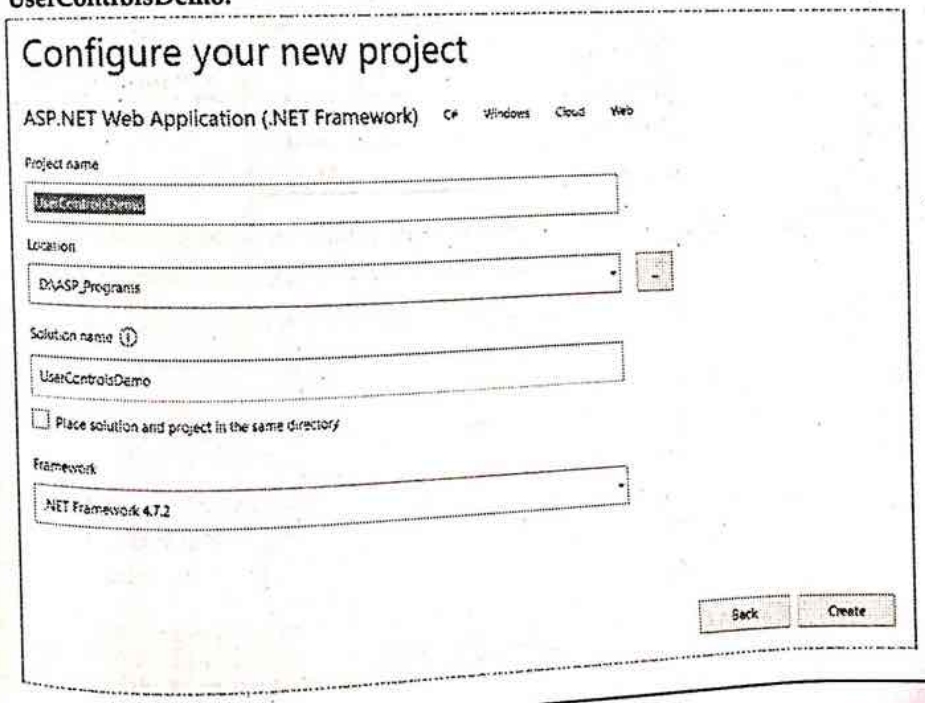
5.24.1 How to Create Web Form Application ?

Following are the steps to be taken to create and build the web forms

Step 1 : Start an instance of Visual Studio 2019 and create a new Web Application Project by clicking **Create a New Project.** Locate **ASP.NET Web Application (.NET Framework)**

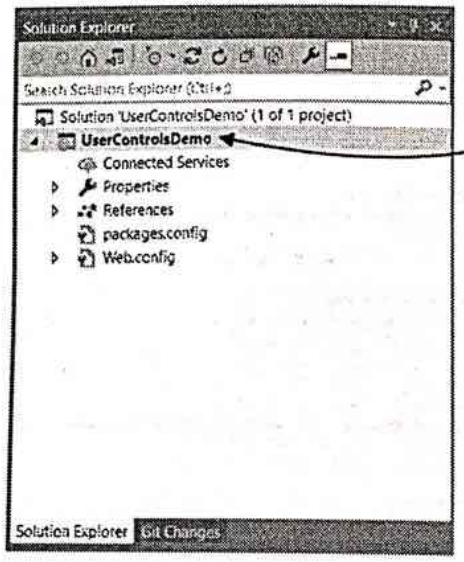


Click on Next button. Give the suitable project name. I have given project name as **UserControlsDemo.**

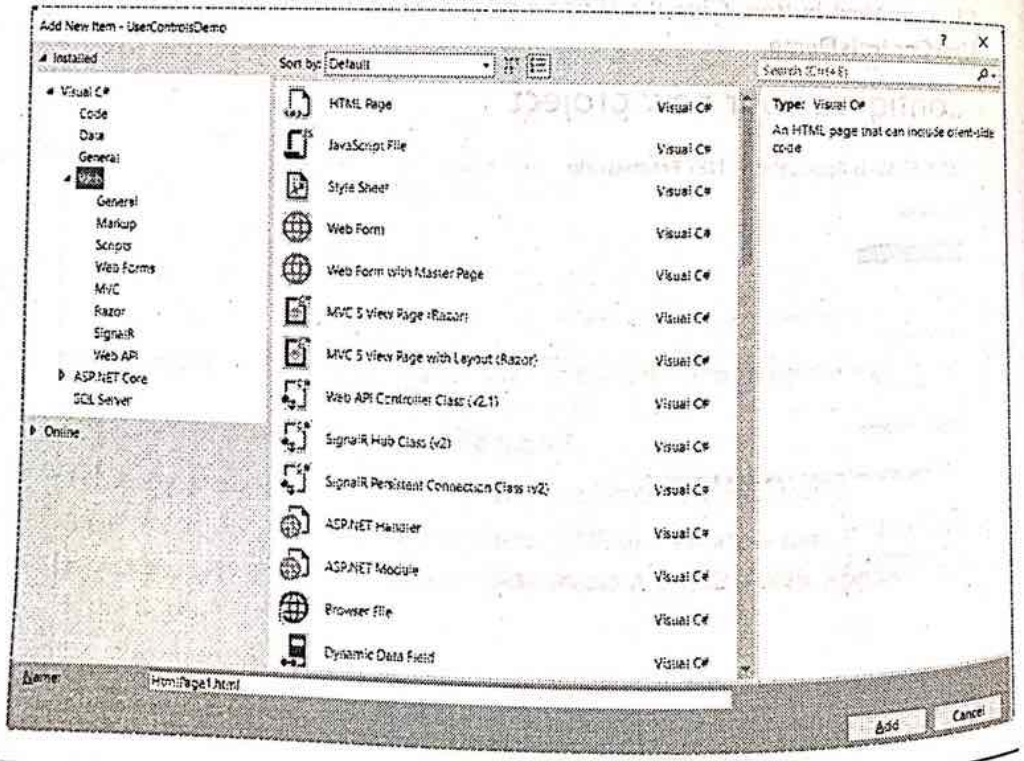


Click on **Create** button. Then Select **Empty** project.

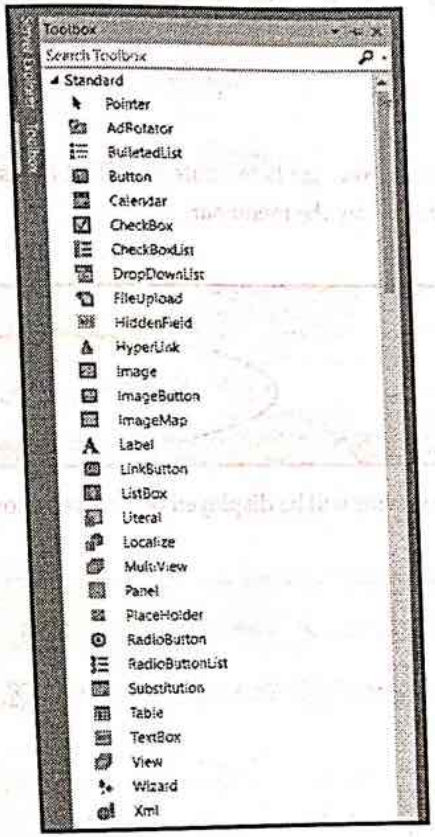
Step 2 : Now in the Solution Explorer window, right click on the project name



Select the **Web Form,** and then click **Add** button.



Step 3 : The default code for **WebForm1.aspx** will be displayed in the **Source** code window. We can add the user controls using **Toolbox.**

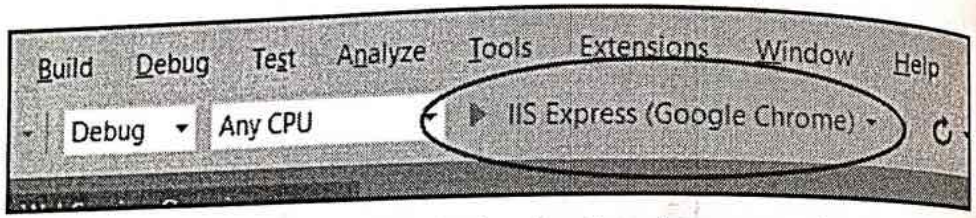


If we double click on some control, then the code for the user control will appear in the source code window. For instance if we want to insert a text box then,

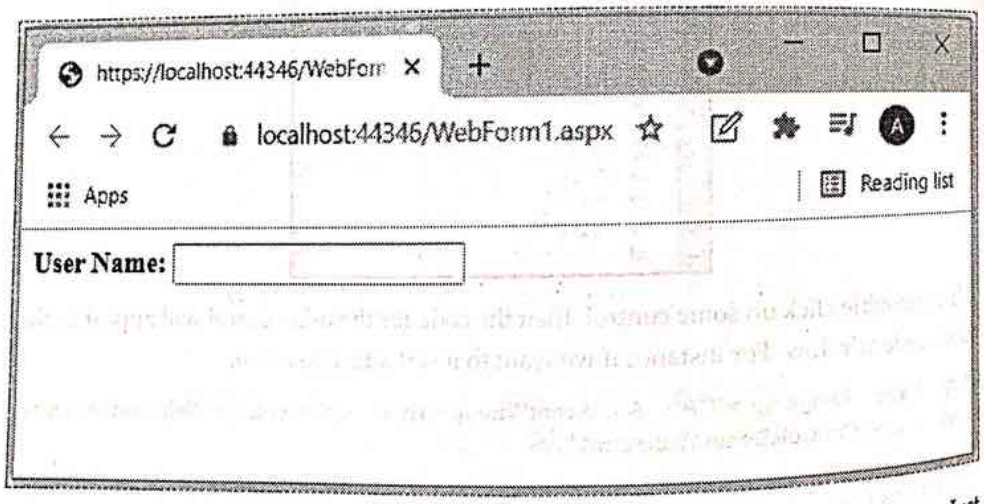
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="UserControlsDemo.WebForm1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
```

```
    <div>
      <b> User Name:</b>
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
</div>
  </form>
</body>
</html>
```

Step 4 : The above web from can be executed on **IIS Express** server. Click on the IIS Express option present below the menu bar.



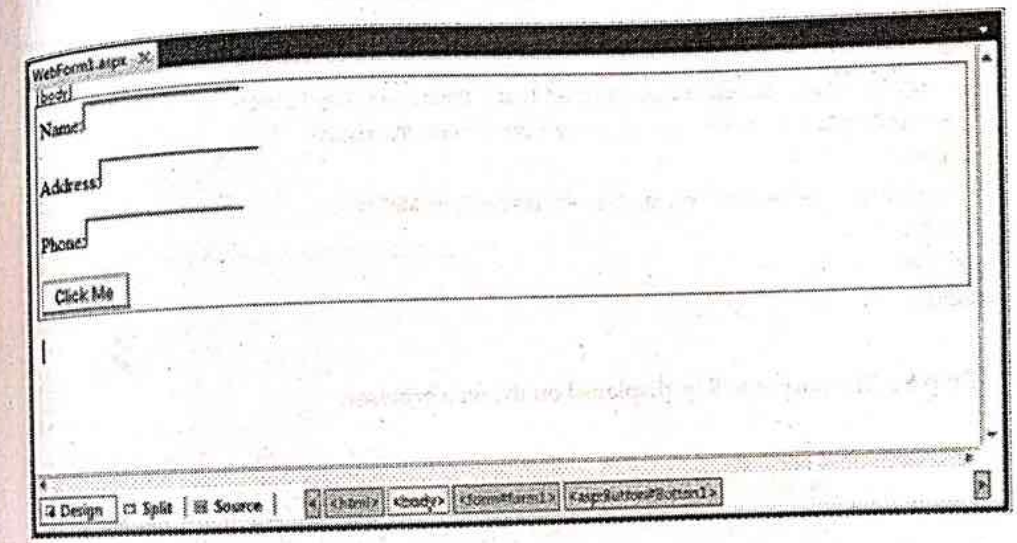
The output of the web service will be displayed on the web browser as follows -



Step 5 : The default code for WebForm1.aspx is displayed in the Code window. Just below that the **Design, Split** and **Source** options are located. Click on **Design.** The form window will get opened up. Now we can select the controls from the ToolBox under the **Standard** option. Click and drag the desired controls on the form. The most commonly used controls are Button, TextBoxes, Labels, Checkboxes, ListBoxes and so on.

Step 6 : Set the properties of the controls using the **Properties Window.** If you can not locate this window just press F4. You can set the properties like ID, Text, ForeColor, Font and so on.

Step 7 : A sample form is designed as follows by simply dragging the controls.

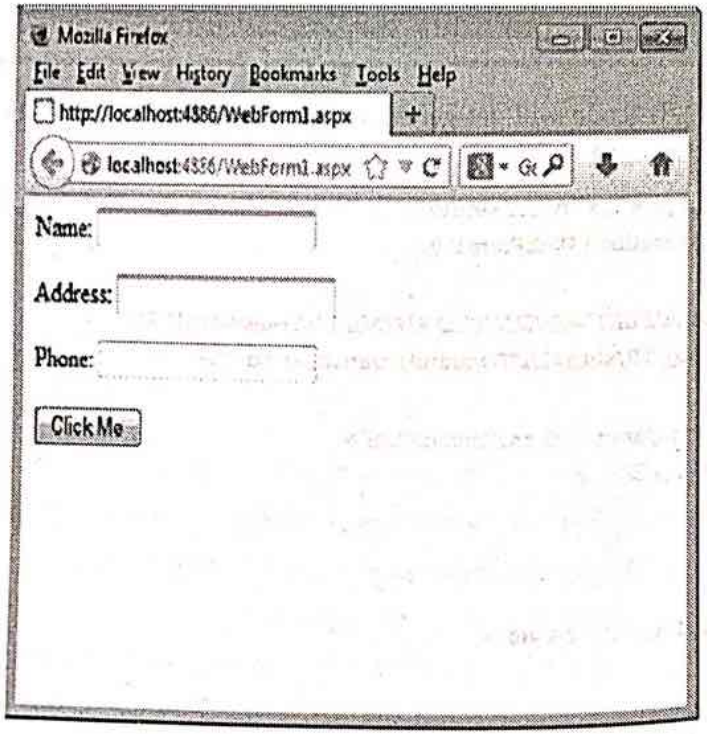


Automatically the code will get generated. This code can be viewed by clicking the **Source.**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication3.WebForm1" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
  <div>

    <asp:Label ID="Label1" runat="server" Text="Name:"></asp:Label>
```

```
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

  </div>
  <p>
    <asp:Label ID="Label2" runat="server" Text="Address:"></asp:Label>
    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
  </p>
  <p>
    <asp:Label ID="Label3" runat="server" Text="Phone:"></asp:Label>
    <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
  </p>
  <asp:Button ID="Button1" runat="server" Text="Click Me" />
  </form>
</body>
</html>
```

Step 8 : The output will be displayed on the web browser.



Examples on Web Form

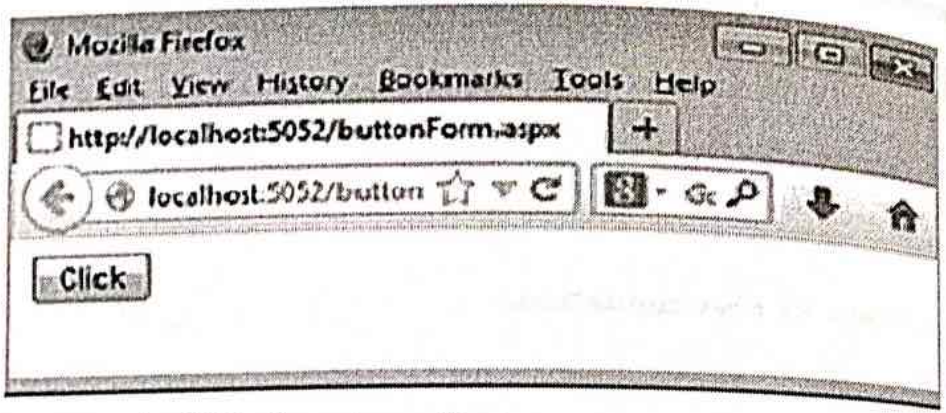
Example 5.24.1 *Write ASP.NET code to demonstrate the button control.*

Solution :

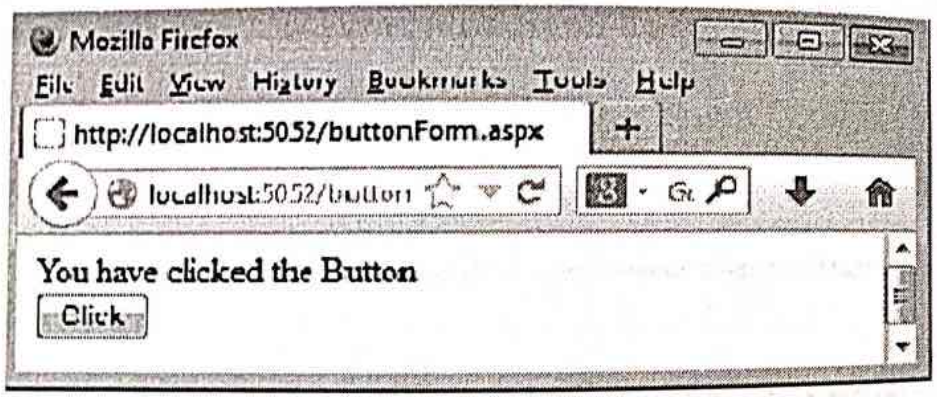
```
<%@ Page Language="C#" %>
<script runat="server">
void Button1_Click(object sender, EventArgs e)
{
  Label1.Text = "You have clicked the Button";
}
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:Label ID="Label1" runat="server"></asp:Label>
  </div>
  <div>
    <asp:Button ID="Button1" runat="server" Type="submit" onclick="Button1_Click"
Text="Click" />
  </div>
  </form>
</body>
</html>
```

In above code, on the Button click the function **Button_Click** is called. In this function we are just assigning the Button clicked message as a Label control's text.

The output will be



just click on the Click button and you will get



Example 5.24.2 *Write ASP.NET code to demonstrate the TextBox control.*

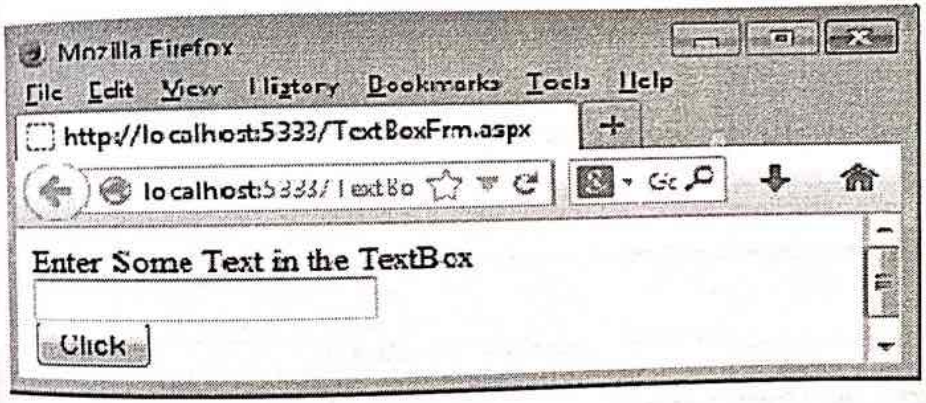
Solution :

```
<%@ Page Language="C#" %>
<script runat="server">
void Button1_Click(object sender, EventArgs e)
{
  Label1.Text ="Welcome "+ TextBox1.Text+ "!!";
}
</script>


<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
```

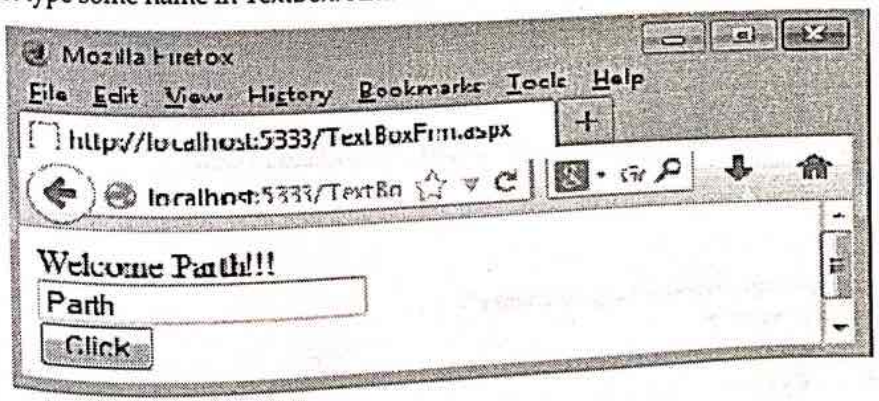


```
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:Label ID="Label1" runat="server" Text="Enter Some Text in the TextBox
">
</asp:Label>
  </div>
  <div>
  <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
  </div>
  <div>
  <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Click" />
  </div>
  </form>
</body>
</html>
```

Output



Just type some name in TextBox. And then click the button.



Example 5.24.3 *Write ASP.NET code to demonstrate the CheckBox and ListBox control.*

Solution :

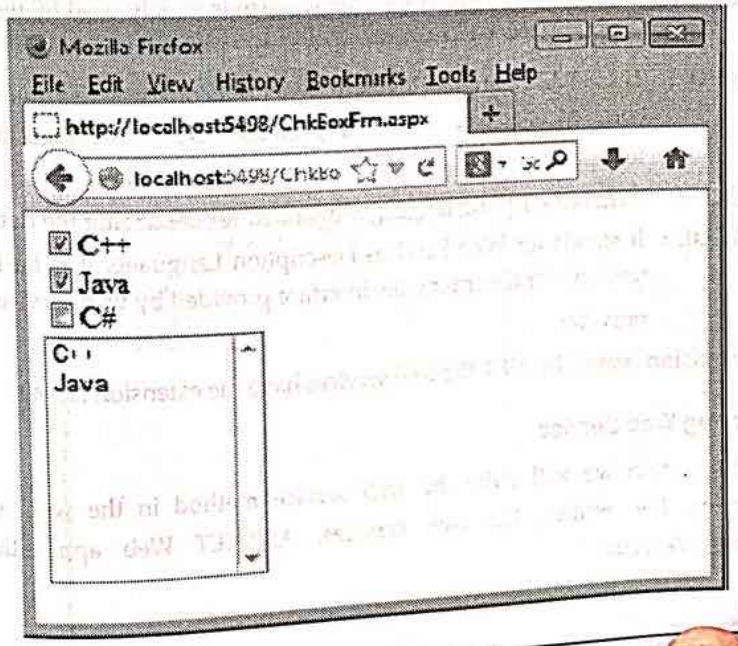
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="ChkBoxFrm.aspx.cs"
Inherits="CheckBoxDemo.ChkBoxFrm" %>

<script runat="server">
void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
  if (CheckBox1 .Checked==true)
  {
    ListBox1.Items.Add(CheckBox1.Text);
  }
  else
  {
    ListBox1.Items.Remove(ListBox1.Items.FindByText(CheckBox1.Text));
  }
}
void CheckBox2_CheckedChanged(object sender, EventArgs e)
{
  if (CheckBox2.Checked == true)
  {
    ListBox1.Items.Add(CheckBox2.Text);
  }
  else
  {
    ListBox1.Items.Remove(ListBox1.Items.FindByText(CheckBox2.Text));
  }
}
void CheckBox3_CheckedChanged(object sender, EventArgs e)
{
  if (CheckBox3.Checked == true)
  {
    ListBox1.Items.Add(CheckBox3.Text);
  }
  else
  {
    ListBox1.Items.Remove(ListBox1.Items.FindByText(CheckBox3.Text));
  }
}
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
```



```
<body>
  <form id="form1" runat="server">
  <div>
    <asp:CheckBox ID="CheckBox1" runat="server" AutoPostBack="True"
     oncheckedchanged="CheckBox1_CheckedChanged" Text="C++" />


  </div>
  <div>
    <asp:CheckBox ID="CheckBox2" runat="server" AutoPostBack="True"
     oncheckedchanged="CheckBox2_CheckedChanged" Text="Java" />
  </div>
  <div>
    <asp:CheckBox ID="CheckBox3" runat="server" AutoPostBack="True"
     oncheckedchanged="CheckBox3_CheckedChanged" Text="C#" />
  </div>
  <div>
  <asp:ListBox ID="ListBox1" runat="server"  Height="140px" Width="120px">
  </asp:ListBox>
  </div>
  </form>
</body>
</html>
```

Output

Just check or uncheck the checkboxes to experience the effect of this control.



5.25 Web Services

- Web service is a software program using which the information can be exchanged. That means using web service one application can invoke the method another application. These applications can be on the same computer or different computers.
- The web services make use of the standard protocols such as HTTP, XML and SOAP. As these are open and standard protocols the applications can exchange the information on any platform with the help of web services. For instance - Web service built using .NET application can be consumed using JSON or Java application.

Features

1) Web service is a language independent.
2) It is platform independent.
3) It is based on XML.
4) It is discoverable. That means, applications and developers can search for and locate the desired web services through registries.
5) It is based on a stateless service architecture.

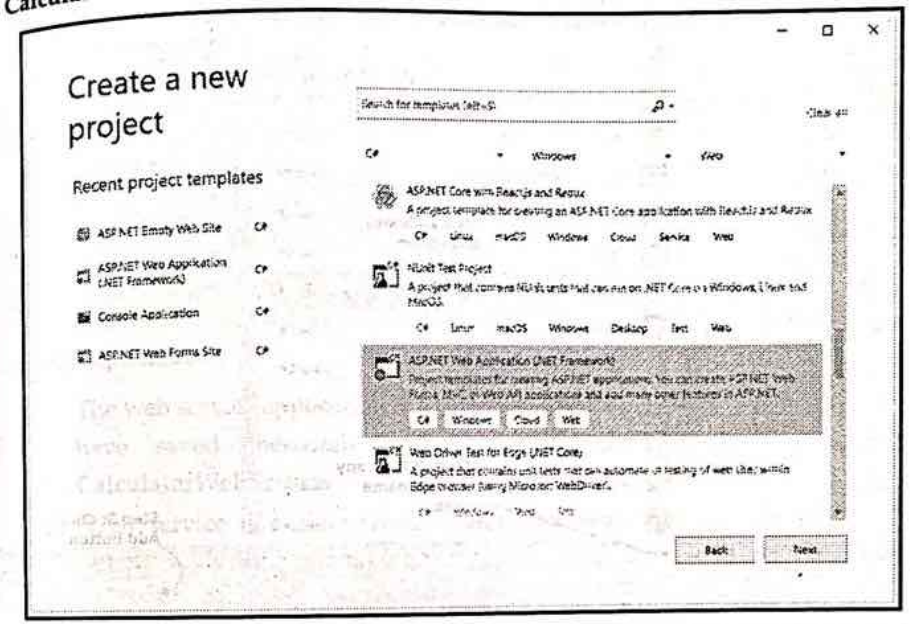
Basic terminologies used in web services

1) **HTTP :** The HTTP is a standard protocol which is widely used by the web services to send and receive messages.
2) **SOAP :** The SOAP (Simple Object Access Protocol) provides the mechanism between web services and applications. SOAP messages are in XML format.
3) **XML :** XML is a scripting languages intended for describing the data.
4) **WSDL :** It stands for Web Services Description Language. It is basically XML notation for describing the interface provided by web service provider.
5) **Extension .asmx :** In .NET the web services have the extension ASMX.

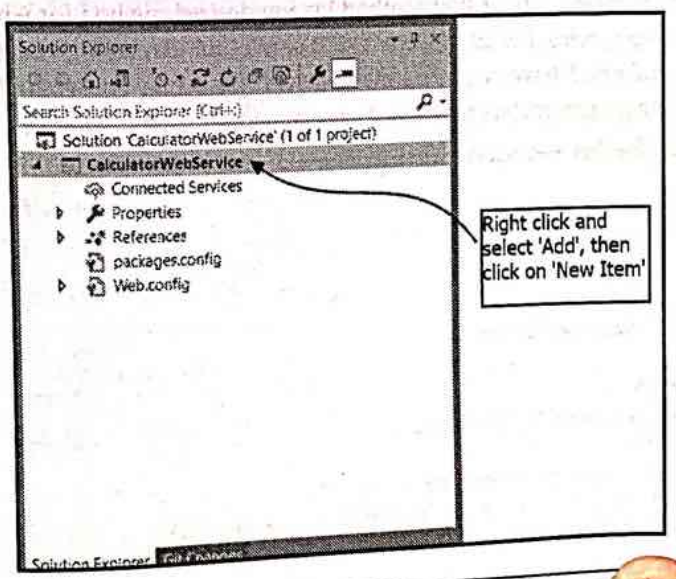
5.25.1 Writing Web Service

- In this section we will write the **web service method** in the web application program. For writing the web services, ASP.NET Web application (.NET Framework) is used.

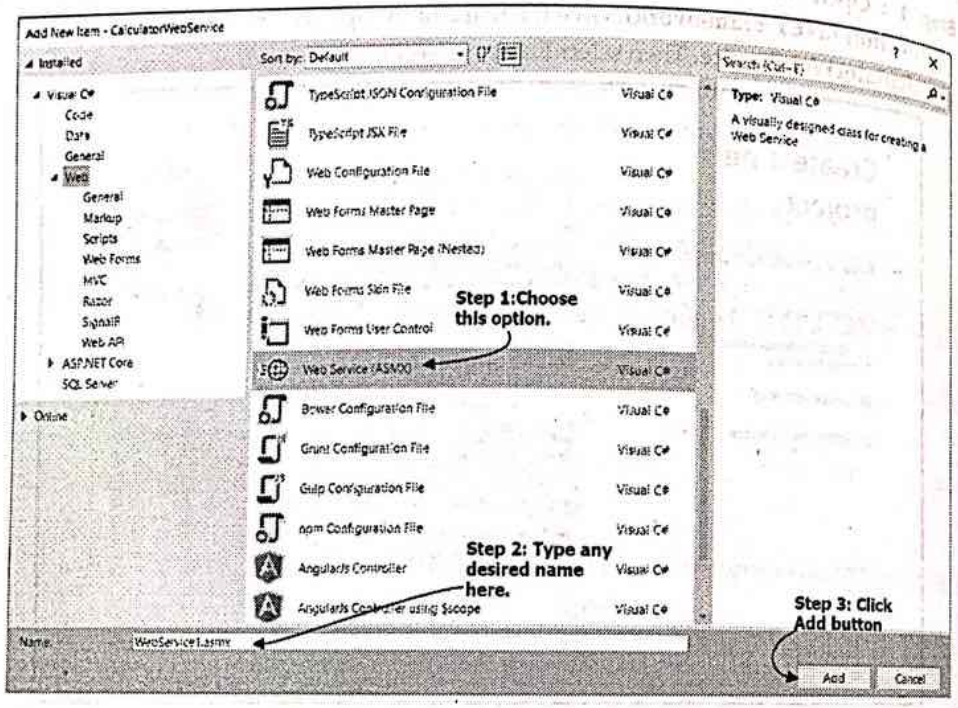
Step 1 : Open Visual Studio. Click on **Create a New Project.** Select **ASP.NET Web Application (.NET Framework).** Give the name of the application. I have given the name as **CalculatorWebService.** Then select **Empty** project template.



Step 2 : In the Solution Explorer window locate the name of the project. Right click on project name.



Locate Web service option. Following screenshot illustrates it.



Step 3 : The source code window appears. Initially the default source code will be present in the source code window. Inside the webservice class, there exists a default **[WebMethod]**. Replace default **WebMethod** by any desired method for which you want to develop a web service. I want to write a web service for displaying maximum number out of two numbers. I have written a method with the name **ShowMaxValue**. To this method two integer parameters are passed as arguments.

- The code for this web service is as follows -

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
namespace CalculatorWebService
{
  /// <summary>
  /// Summary description for WebService1
  /// </summary>
  [WebService(Namespace = "http://tempuri.org/")]
  [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
  [System.ComponentModel.ToolboxItem(false)]
```



```
  // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
  // [System.Web.Script.Services.ScriptService]
  public class WebService1 : System.Web.Services.WebService
  {
    [WebMethod]
    public int ShowMaxValue(int a,int b)
    {
      if (a > b)
        return a;
      else
        return b;
    }
  }
}
```

Program Explanation :

The web service application is written in some namespace. In above program we have saved the web service by the name of the namespace as **CalculatorWebService.**

1) Web service is a class which is declared with **[WebService]** attribute. It is inherited from **System.Web.Services.WebService** base class. The attribute [WebService] tells that this class contains the code for web services.

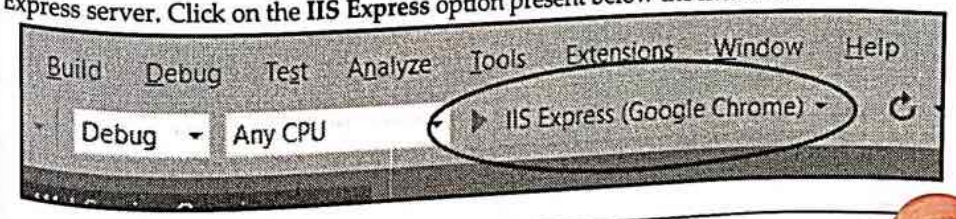
WebService namespace is used to uniquely identify our web service among all other web services present on the internet. It is generally written as follows -

```
[WebService(Namespace = "http://pragimtech.com/webservices")]
```

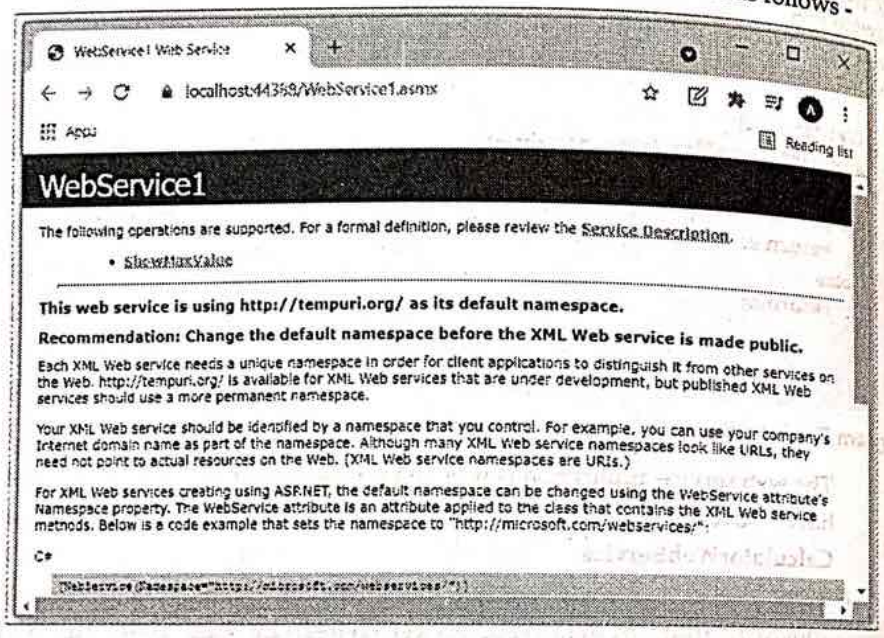
In above program, we have created a web service using the method **ShowMaxValue.** Before declaring this method we use the attribute **[WebMethod].** If you want a method name to be displayed on the web browser as a part of the web service, then it must be declared as **public** and it must be decorated with [WebMethod] attribute.

We have passed two arguments to **ShowMaxValue** method. This method finds the largest among the two numbers and return this largest number.

Step 4 : Here the web service is created. The above web service can be executed on IIS Express server. Click on the **IIS Express** option present below the menu bar.

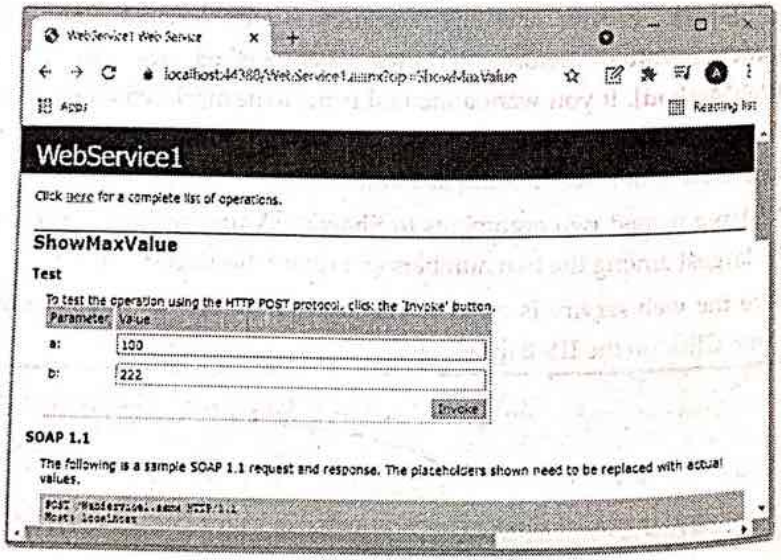


The output of the web service will be displayed on the web browser as follows -

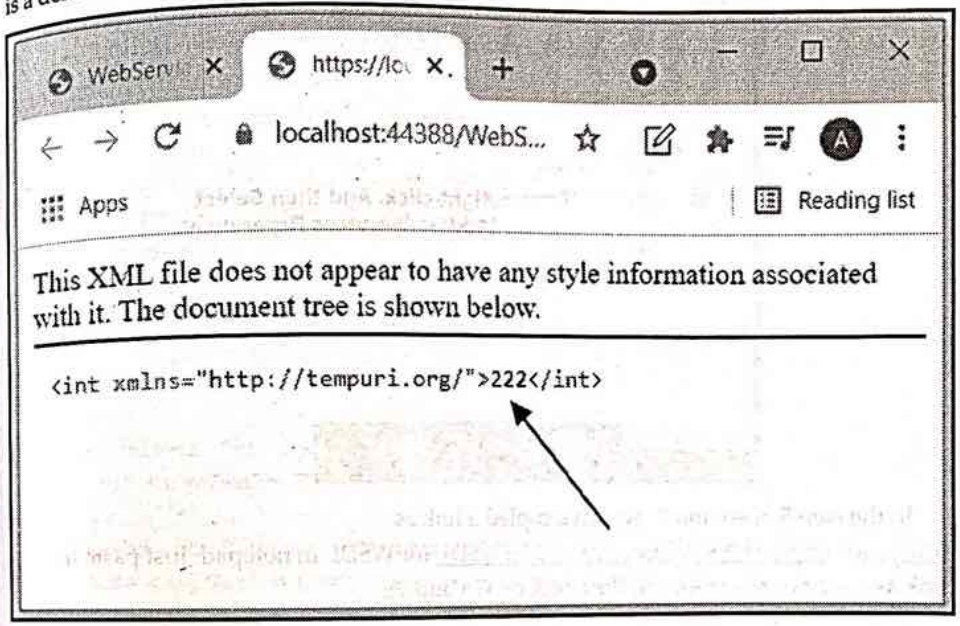


Step 5 : From above window, just click on **Service Description** link. In the address bar we get the URL as **https://localhost:44388/WebService1.asmx?WSDL**. Just copy this link and save it to notepad.

If we click on **ShowMaxValue** we get following screen.



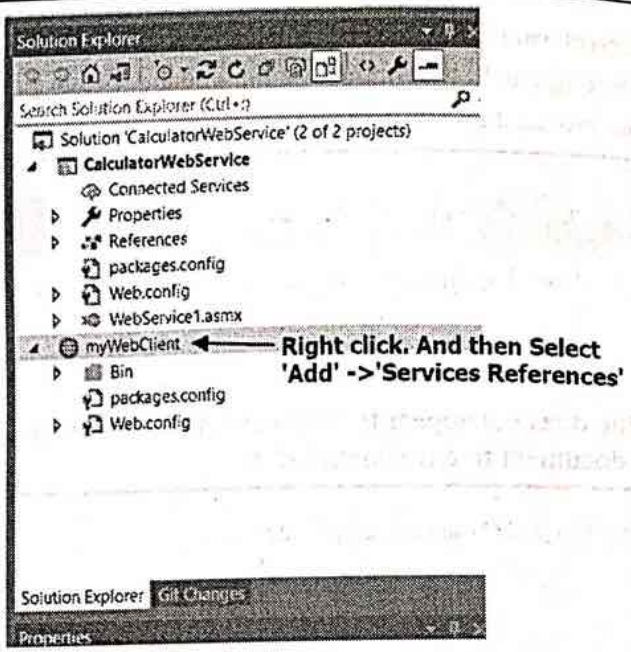
The two different values can be entered in the text boxes. On hitting the **Invoke** button, we get the result in the XML document. It is as follows. Note that the above layout is a default layout provided by .NET framework.



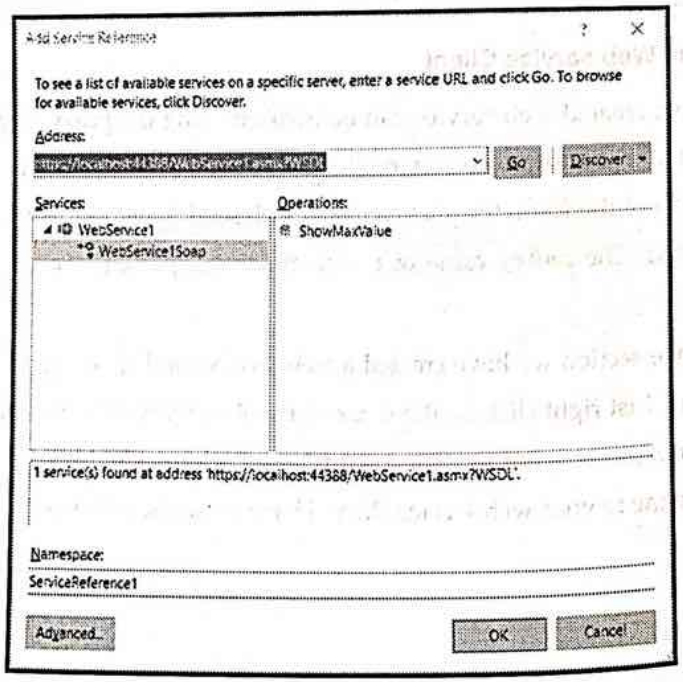
5.25.2 Writing Web service Client

- The above created web service can be invoked using the proxy web client. In this section we will create web service client by creating a HTML form. On the button click event of the form, the above created web service method **ShowMaxValue** will be invoked. The return value of this method is displayed as a result on the web browser.

Step 1 : In above section we have created a web service for displaying the maximum out of two numbers. Just right click on the name of the above web project which is displayed in the solution explorer window. Select **Add New Project->ASP.NET Empty Web Site.** Give suitable name to your web service client. I have given the name **myWebClient.**



In the step 5 of section **, we have copied a link as https://localhost:44388/WebService1.asmx?WSDL for WSDL in notepad. Just paste this link as a service reference and then click on **Go** button.



Click **OK** button. Thus the web service reference by the namespace **ServiceReference1** is created for the web service client.

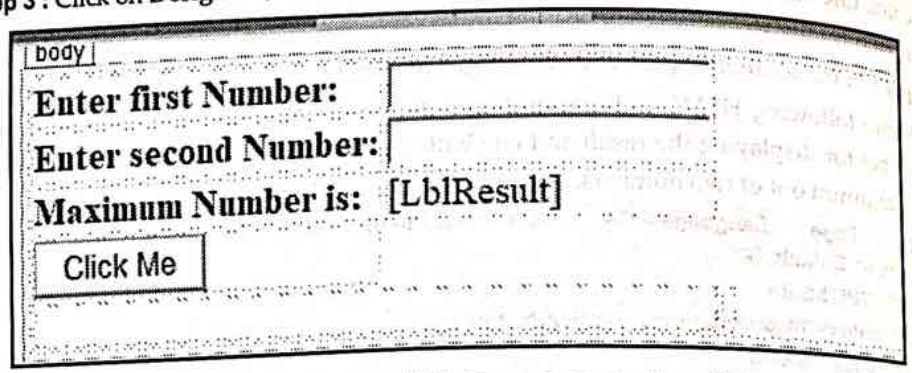
Step 2 : In the solution explorer window, right click on **myWebClient** and add a form.

Write following HTML code for creating two text boxes for inputting two numbers, one label for displaying the result and one button to invoke the web service for showing the maximum out of two numbers.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <table>
      <tr><td><b>Enter first Number:</b></td>
        <td> <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox></td>
      </tr>
      <tr><td><b>Enter second Number:</b></td>
        <td> <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox></td>
      </tr>
      <tr><td><b>Maximum Number is:</b></td>
        <td><asp:Label ID="LblResult" runat="server" Text=""></asp:Label> </td>
      </tr>
      <tr><td>
        <asp:Button ID="btnResult" runat="server" Text="Click Me" OnClick="btn_Click"/>
      </tr>
      <tr><td colspan="2">
        <td> </td>
      </tr>

    </table>
  </form>
</body>
</html>
```

Script Explanation : In above given HTML document we have created two text boxes for inputting two numbers. One label control for displaying the maximum of the two numbers. On button click, the web service method **ShowMaxValue** is called.

Step 3 : Click on **Design** tab, the form design will be as follows –



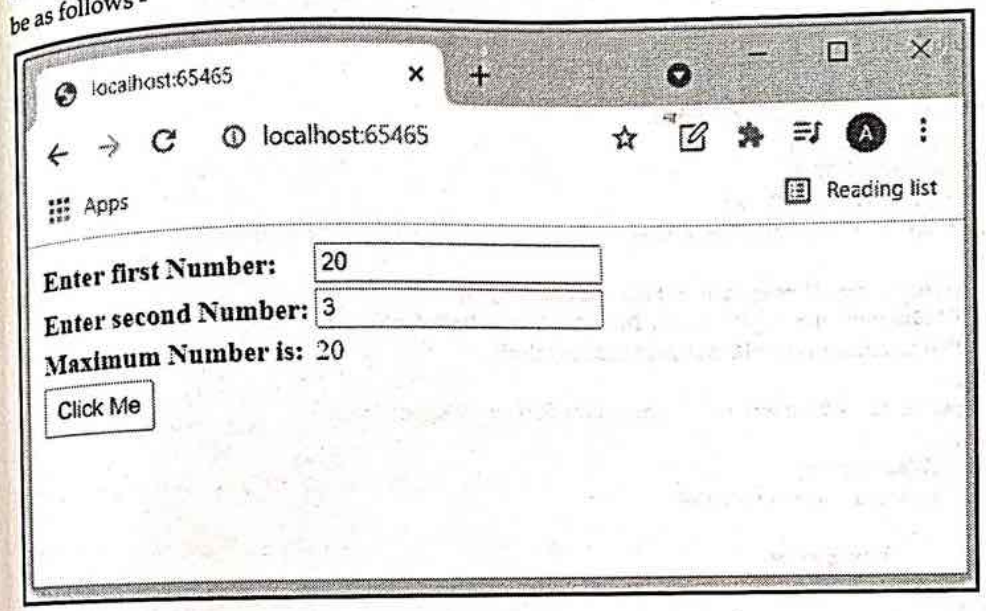
- Double click on 'Click Me' button. Write the code for button click event as follows –

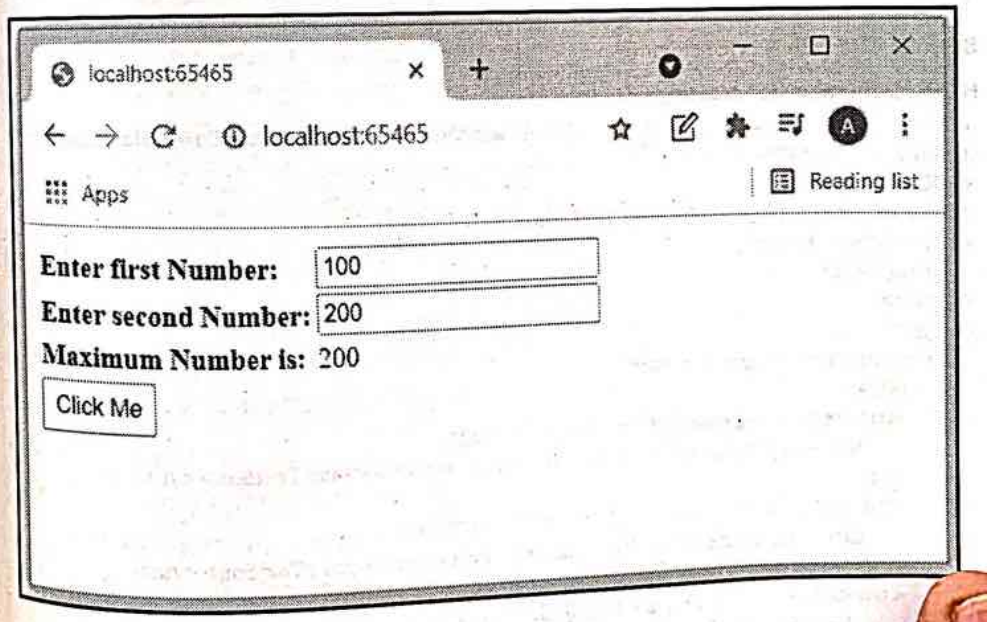
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
public partial class _Default : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {
  }
  protected void btn_Click(object sender, EventArgs e)
  {
      ServiceReference1.WebService1SoapClient client =
      new ServiceReference1.WebService1SoapClient();
      int result = client.ShowMaxValue(
Convert.ToInt32(TextBox1.Text), Convert.ToInt32(TextBox2.Text));
      LblResult.Text = result.ToString();
  }
}
```

Web Service method named **(ShowMaxValue)** written in previous section is invoked.

The **Convert.ToInt32** method is used to convert the textbox string to numeric value.

Step 4 : Right click on web client project and click on 'view in Browser'. The output will be as follows -





Example 5.25.1 *Write a web service and web client for implementing addition of two numbers using .NET.*

Solution : Step 1 : Web service application

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
namespace CalculatorWebService
{
  [WebService(Namespace = "http://tempuri.org/")]
  [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
  [System.ComponentModel.ToolboxItem(false)]

  public class WebService1 : System.Web.Services.WebService
  {
    [WebMethod]
    public int Add(int a,int b)
    {
        return a+b;
    }
  }
}
```

Step 2 : The web service client.

HTML Document for creating GUI

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
Inherits="_Default" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <table>
      <tr><td><b>Enter first Number:</b></td>
        <td> <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox></td>
      </tr>
      <tr><td><b>Enter second Number:</b></td>
        <td> <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox></td>
      </tr>
      <tr><td><b>Addition is:</b></td>
        <td><asp:Label ID="LblResult" runat="server" Text=""></asp:Label> </td>
```



```
      </tr>
      <tr><td>
        <asp:Button ID="btnResult" runat="server" Text="Click Me" OnClick="btn_Click"/>
      </tr>

    </table>
  </form>
</body>
</html>
```

C# Program for handling button click event

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
  protected void btn_Click(object sender, EventArgs e)
  {
    ServiceReference1.WebService1SoapClient client =
    new ServiceReference1.WebService1SoapClient();
    int result = client.Add(
    Convert.ToInt32(TextBox1.Text), Convert.ToInt32(TextBox2.Text));
      LblResult.Text = result.ToString();
  }
}
```

Part - III : Node.js

SPPU : May-18, Marks 4

5.26 Overview of Node JS

- NodeJS is an open source technology for server.
- Using Node.js we can run JavaScript on server.
- It runs on various platform such as Windows, Linux, Unix, and MacOS.

Uses of Node.js

- It can perform various tasks such as -
  1. It can create, open, read, delete, write and close files on the server.
  2. It can collect form data.
  3. It can also add, delete, modify data in databases.
  4. It generate dynamic web pages.

How to write and execute Node.js document ?

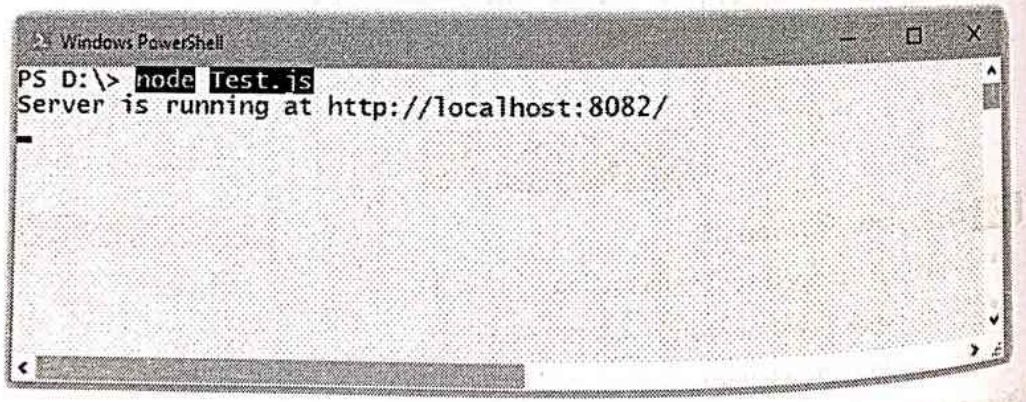
- For executing the Node.js scripts we need to install it. We can get it downloaded from https://nodejs.org/en.
- The installation can be done on Linux or Windows OS. It has very simple installation procedure.
- After successful installation we can now execute the Node.js document.
- The Node.js file has extension .js.

Step 1 : Open Notepad and type the following code. Save the file with some filename having extension .js

Test.js

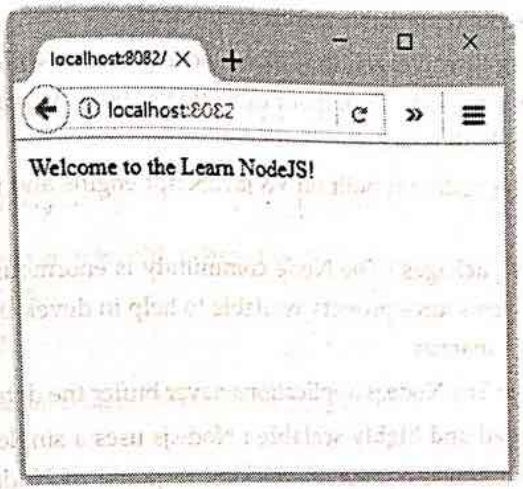
```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end("Welcome to the Learn NodeJS!");
}).listen(8082);
console.log('Server is running at http://localhost:8082/');
```

Step 2 : Open command prompt window and type the command for executing above script. The screen-shot for this execution is as follows -



Now open the web browser such as Mozilla Firefox or Chrome or Internet Explorer and in the address bar type the URL http://localhost:8082/

The output will be as follows –



Components of Node.js Application

- Module is similar to JavaScript libraries. It is basically the set of functions that need to be included in the application. For including the module we need a function **require()**. For example to include the HTTP module in the application we write -

```
var http = require('http');
```

- Node.js has a built-in module called HTTP, which allows Node.js to transfer data over the Hyper Text Transfer Protocol.
- The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client. Use the **createServer()** method to create an HTTP server. For example -

```
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end("Welcome to the Learn NodeJS!");
}).listen(8082);
```

- If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type. For example

```
res.writeHead(200, {'Content-Type': 'text/html'});
```

The status code, 200 means that all is OK, the second argument is an object containing the response headers.

- The above code can be executed and the web page can be displayed on the Web browser on particular port using **.listen(port number)**

5.26.1 Features of Node JS

- Following are some remarkable features of node.js –
  1) **Non blocking thread execution :** Non blocking means while we are waiting for a response for something during our execution of tasks, we can continue executing the next tasks in the stack.
  2) **Efficient :** The node.js is built on V8 JavaScript engine and it is very fast in code execution.
  3) **Open source packages :** The Node community is enormous and the number of permissive open source projects available to help in developing your application in much faster manner.
  4) **No buffering :** The Node.js applications never buffer the data.
  5) **Single threaded and highly scalable :** Node.js uses a single thread model with event looping. Similarly the non-blocking response of Node.js makes it highly scalable to sever large number of requests.

Review Question

1. *Write short note on NodeJS.* **SPPU : May-18, Marks 4**

5.27 Multiple Choice Questions

Q.1 In PHP, a variable starts with the sign _____.

a $ b &
c # d @

Q.2 PHP supports the following data types :

a String b Integer
c Array d All the above

Q.3 In PHP, to represent exponentiation, the following operator is used :

a * b **
c ^ d $

Q.4 A cookie is created by using the following function :

a getcookie b setcookie
c getcookie() d setcookie()

Q.5 The following mode opens the file for write only :

a w b r
c w+ d a+

Q.6 How do you get information from a form that is submitted using the "get" method ?

a $_GET[]; b Request.Form;
c Request.QueryString; d Request.GetQuery;

Q.7 Which superglobal variable holds information about header, paths and script, locatio

a $_GLOBALS b $-SERVER
c $-SESSION d $_GET

Q.8 The filesize() function returns the file size in _____.

a bits b bytes
c kilobytes d gigabyte

Q.9 Which function sets the file filename's last-modified and last-accessed times ?

a sets() b Set()
c touch() d touched()

Q.10 The file extension to ASP.NET web form is _____.

a .net b .aspx
c .docx d .html

Q.11 ASP.NET was developed by _____.

a Google b IBM
c Oracle d Microsoft

Q.12 The JIT is a part of _____ in .NET framework.

a Framework Class Library (FCL)
b Languages
c Common Language Runtime (CLR)
d All of the above

Q.13 In ASP.NET application, the DLL files are stored in _____ folder.

a bin b App_data
c App_code d App_LocalResources

14 Which of the following is the first event of ASP.NET page when the user requests a web page ?

a Init
b PreInit
c Render
d Control Events

15 Which of the following is the page life cycle event of ASP.NET page when the user makes use of button control to invoke click events a web page ?

a Init
b PreInit
c Render
d Control Events

6 Web services can be discovered using _____.

a UDDII
b UDDI
c UDDDI
d UDII

7 Which of the following is a directory service where enterprises register and search for web services ?

a WSDL
b UDDI
c Supporting protocols of SOAP
d None of the above

Node.js is written in _____.

a C
b C++
c JavaScript
d All of the above

The extension used to store the Node.js script is _____.

a .html
b .java
c .js
d .node

r Keys for Multiple Choice Questions :

Q.1	a	Q.2	d	Q.3	b	Q.4	b	Q.5	a
Q.6	a	Q.7	a	Q.8	b	Q.9	c	Q.10	b
Q.11	d	Q.12	c	Q.13	a	Q.14	b	Q.15	d
Q.16	b	Q.17	b	Q.18	d	Q.19	c		

UNIT - VI

6 Ruby and Rails

Syllabus

Introduction to Ruby : *Origins & uses of Ruby, scalar types and their operations, simple input and output, control statements, fundamentals of arrays, hashes, methods, classes, code blocks and iterators, pattern matching.* ***Introduction to Rails :*** *Overview of Rails, Document Requests, Processing Forms, Rails Applications and Databases, Layouts, Rails with Ajax. Introduction to EJB.*

Contents

6.1 Origins & Uses of Ruby
6.2 Scalar Types and their Operations
6.3 Simple Input and Output
6.4 Control Statements
6.5 Fundamentals of Arrays
6.6 Hashes
6.7 Methods
6.8 Classes
6.9 Code Blocks and Iterators
6.10 Pattern Matching
6.11 Overview of Rails
6.12 Document Requests
6.13 Processing Forms
6.14 Rails Applications and Databases
6.15 Layouts
6.16 Rails with Ajax ..... May-18, 19, Dec.-19, ... Marks 8
6.17 Introduction to EJB
6.18 Multiple Choice Questions

Part I : Introduction to Ruby

6.1 Origins & Uses of Ruby

- Ruby is a scripting language designed by Yukihiro Matsumoto also known as Matz.
- Ruby executes on variety of platforms such as Windows, Linux, Mac and so on.
- Ruby is a pure object oriented programming language. Ruby is an open source and it is freely available on the web.
- Ruby is similar to smalltalk, python, Perl.
- Ruby has the pattern matching feature which is similar to the Perl.
- Ruby can be easily connected to the databases like MYSQL, DB2 and Oracle.

Environment

- For executing the Ruby programs we need to install **Rubystack.**
- The Ruby Stack can be downloaded and installed from

  http://bitnami.com/stack/ruby.
- It can be downloaded on Windows/ Linux or MacOS platform. It consists of Ruby, Rails, Apache, MySQL, SQLite, Git, PHP and phpMyAdmin in one pack. Installing Ruby Stack is quick and easy. It is a **self - contained** system.
- After installing the Ruby Stack we can start the Apache Server and MYSQL Server using the **Managers.**

6.2 Scalar Types and their Operations

- There are three types of data types used in Ruby and those are Scalar, Arrays and hashes.
- The scalar types are further classified as **numeric** and **character strings.** Refer Fig. 6.2.1.

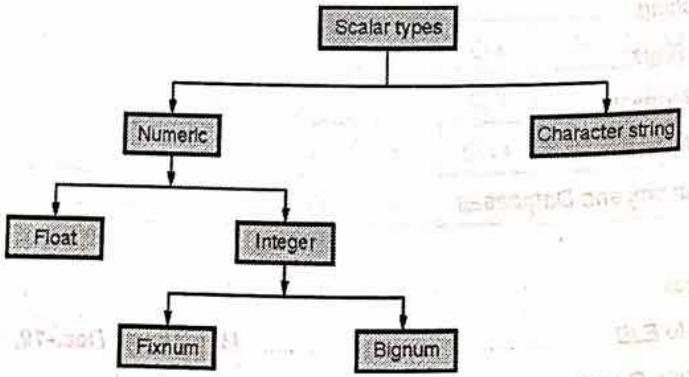


Fig. 6.2.1 Scalar types

6.2.1 Numeric and String Literals

- All the numeric data types in Ruby are derived from the **Numeric** class.
- The immediate children of Numeric class are **Float** and **Integer.**
- The **Integer** class has two child classes - **Fixnum** and **Bignum.**
- The integer objects which lies within the size of 32 bit is called the **Fixnum** object and the object which is greater in size of 32 bit is called the **Bignum** object.
- If a **Fixnum** object whose size grows beyond the 32 bit then it is implicitly typecast to **Bignum** object.
- Similarly in expression evaluation the size of particular Bignum object becomes less than 32 bit then it is converted to the **Fixnum** object.
- The integer literal may contain the underscore character to read out the large integer conveniently. For example 231456981 can be read using the underscore as 231_456_981.
- The numeric literal that contains the floating point is called the **Float** object. It denotes the double precision floating type value.
- The decimal point must be preceded and followed by at least one digit. For example .22 will be illegal literal in Ruby.
- The **string** object consists of the string literals that are nothing but the collection of characters.
- There are two types of strings - single quoted and double quoted string.
- If we want to print some character literally like apostrophe s then the \ is used.

Following is a simple script written in Ruby to demonstrate the use of string literal

Step 1 : Open the Notepad or some suitable text editor and type the following code -

Ruby Document [StringDemo1.rb]

```
puts 'Cheers!'
puts 'It\'s a beautiful day'
puts "An apple a day \n keeps the doctor away"
```

Step 2 : Open the command editor for ruby and you can see the following output -

6.2.2 Variable and Assignment Statements

- The variable used for storing the data values.
- The variables in Ruby are **case sensitive.**
- It may contain the letters, underscore and digits. Conventionally the variable must be declared in the lower case.

Ruby Script [VariableDemo.rb]

```
pi =3.14
r=15
puts "The value of pi variable is #{pi}"
puts "The radius of the circle is #{r}";
puts "The area of the circle is #{pi*r*r}"
```

Output

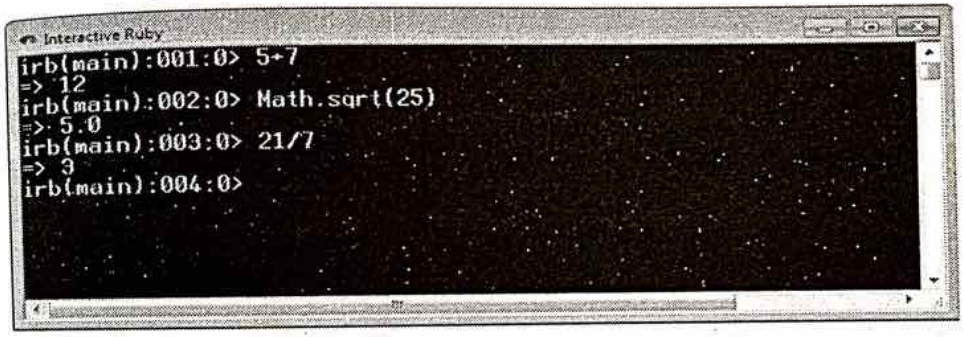
```
The value of pi variable is 3.14
The radius of the circle is 15
The area of the circle is 706.5
```

- There is no need to declare the variable in Ruby.
- In Ruby every data value is an object. Hence objects need references.
- If a scalar variable is not assigned with a value then it is assigned with the value nil.
- The constants in Ruby are declared with the first letter capital. A constant is created when a variable is assigned with some value it becomes a constant variable.
- There are special variables in variables which are called as **implicit variables.** The scalar variables that are implicit variables begin with dollar sign while other type of implicit variables can begin either by underscore _, carrot ^ or back slash \
- The variables in Ruby are **typeless.** All that is assigned in an assignment statement is address of an object.

6.2.3 Numeric Operators

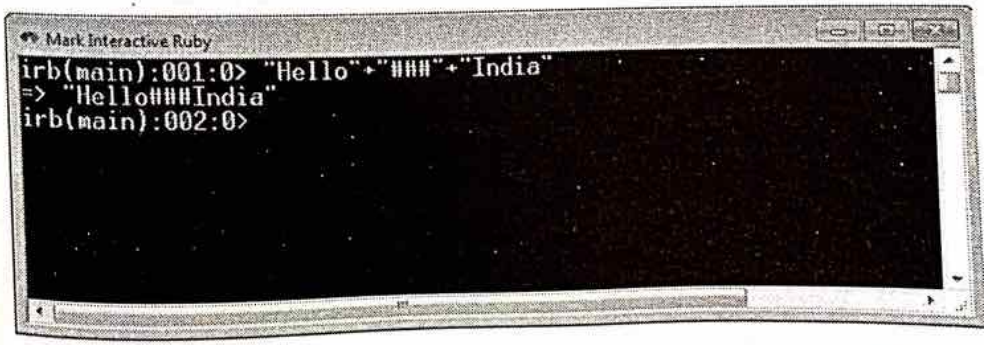
- The arithmetic operators such as + addition, - subtraction, * multiplication / division and mod operators.
- The **precedence rules** specify which operator is to be evaluated first when to operators of different level of precedence appear in the expression.
- The **associativity rule** specify which operator is to be evaluated first when two operators of same level of precedence appear in the expression.
- The Ruby does not include increment ++ or decrement -- operator.

- There is a **math** module supported by Ruby. This module provides the methods for basic trigonometric and mathematical functions. For instance the functions such as cos, sin, tan, sqrt are supported by this module.
- Along with the software package of Ruby an interactive interpreter is provided. This interpreter is called **Interactive Ruby.** The acronym for this interpreter is **irb.** The user can type the expression and get the answer of this expression immediately.
- For that purpose, click on Start->All Programs->Ruby 1.9.2->Interactive Ruby. Following is the illustration -



6.2.4 String Methods

- Ruby supports various string methods.
- String method supports concatenation operator. The operator is denoted by +. The operator + is used for concatenation.



- The operator << is used to append the string. For example :

```
Interactive Ruby
irb(main):001:0> str="Hello_"
=> "Hello_"
irb(main):002:0> str<<"friends"
=> "Hello_friends"
irb(main):003:0>
```

- Every variable in Ruby refers to some memory location. If one variable is assigned to another variable then that variable will refer to the same memory location. For example :

```
Interactive Ruby
irb(main):001:0> str1="MyRuby"
=> "MyRuby"
irb(main):002:0> str2=str1
=> "MyRuby"
irb(main):003:0> str2
=> "MyRuby"
irb(main):004:0> str1="Program"
=> "Program"
irb(main):005:0> str1
=> "Program"
irb(main):006:0> str2
=> "MyRuby"
irb(main):007:0>
```

For example : In above screen-shot you can see that the variable str1 contains the string "MyRuby". Another variable str2 is assigned with the value of str1. Hence on typing the string str2 we get the same string i.e."MyRuby". Now the string str1 is assigned with another string "Program". Hence we get str1="Program" but still the value of str2="MyRuby".

- Instead of directly assigning some string to str1, we can use the **replace** method for assigning some new string.

```
For example: Instead of
str1="Program"
we can write
str1.replace("Program");
```


- Following are some methods that are used for various string operations -

Method/Operator	Action	Example
Capitalize	This method converts the first letter to uppercase	"india".capitalize => "India" But "11a".capitalize => "11a"
chop	This method removes the last character.	"Lucky".chop => "Luck"
chomp	This method the newline from the right end, if there is one.	"Hello\n Friends\n".chomp => "Hello\n Friends"
equal ==	The method equal is used to compare two objects. The == equal to operator treats the string literally.	"Nisha".equal?("Nisha") => false because these are treated as two different objects "Nisha"=="Nisha" => true because the string is compared literally
upcase	This method converts all the lower case letters to the object case.	"nisha".upcase => "NISHA"
downcase	This method converts the upper case letters of the string to lower case.	" NISHA ".upcase => "nisha"
strip	This method removes the space on both ends.	" India ".strip => "India"
lstrip	This method removes the space from the left end	" India ".lstrip => "India "
rstrip	This method removes the space from the right end	" India ".strip => " India"
swapcase	This method converts the upper case to lower case and lower case to upper case.	"NiSha".swapcase => "nIsHA"
reverse	It reverses the string.	"Ruby".reverse => "ybuR"
*	The repetition operator is used to specify the repetition. The repetition is specified by asterisk(*)	"Hurray!!! "*3 =>"Hurray!!! Hurray!!! Hurray!!! "
<=>	This is called the spaceship operator. If two operands are equal then it returns the 0; If first operand is greater than the second operand then it returns 1; If the second operand is greater than the first one then it returns -1	"aaa"<=>"bbb" => -1 "bbb"<=>"aaa" => 1 "aaa"<=>"aaa" => 0

6.3 Simple Input and Output

- The input and output operations are the most important operations for any programming language.
- Typically the input is given by the keyboard and the output is displayed on the screen.

6.3.1 Screen Output

- The **puts** method is used to display the output on the screen.
- The operand for the **puts** method is **string**.
- The value is the variable can be displayed using the **puts** method but in that case #{...} is used. For example :

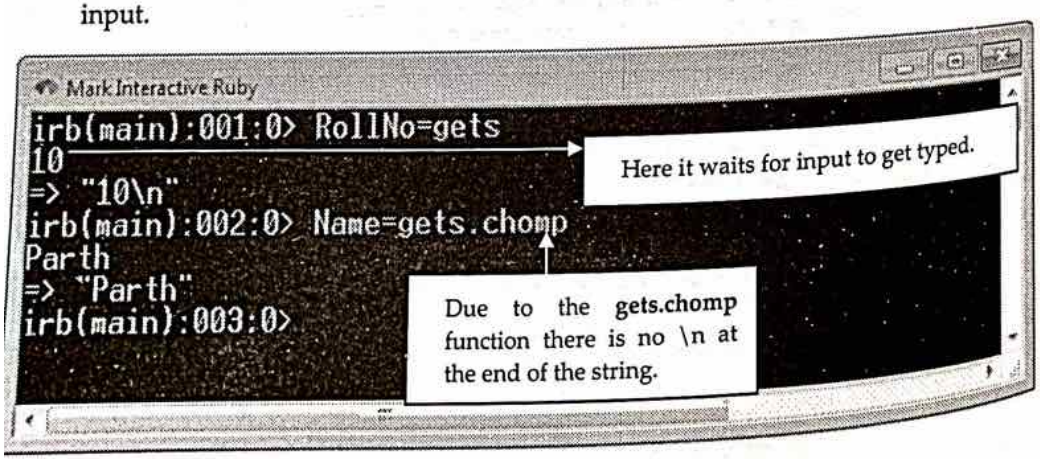
```
RollNo=10
Puts "My Roll number is #{RollNo}"
```

- The new line character can be appended at the end of the string so that next subsequent contents can be displayed on the screen. But it writes the word **nill** at the end. For example :

```
puts "My favorite langauge is #{str}"
My favorite langauge is Ruby
=> nil
```

6.3.2 Keyboard Input

- In Ruby the keyboard input is rarely needed but we can give the input to the program using keyboard.
- For the keyboard input the method **gets** is used. This method reads the line of input.



- The **gets.to_i** or the **gets.to_f** method is useful for obtaining the integer or the float value. Following are the simple steps that can be followed for executing the ruby program :

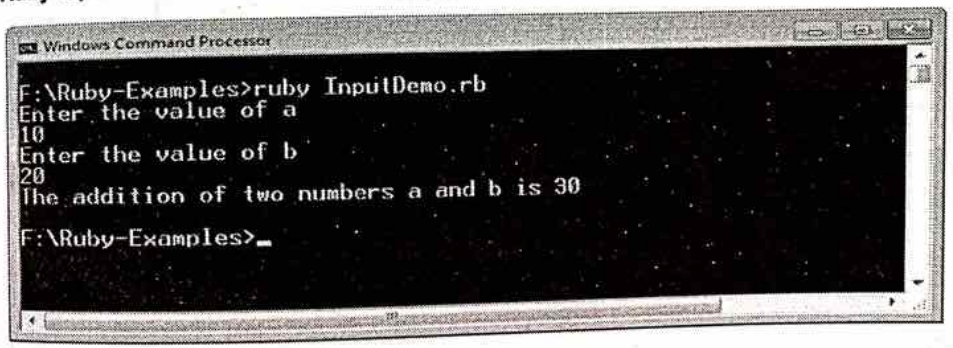
Step 1 : Open the suitable text-editor such as Notepad or Wordpad. Type the following code and save this file using the extension **.rb.**

InputDemo.rb

```
puts "Enter the value of a"
a=gets.to_i
puts "Enter the value of b"
b=gets.to_i
c=a+b
puts "The addition of two numbers a and b is #{c}"
```

Step 2 : Now type the command at the command prompt for executing the ruby program as

Ruby InputDemo.rb



6.4 Control Statements

- In Ruby the control statements are used for controlling the flow of control.
- It consists of control expressions, selection statements and looping statements.

6.4.1 Control Expressions

- The control expressions are Boolean type expressions.
- The relational expression has two operands and one operator.
- There are two sets of Boolean operators. The operators **&&(AND)**, **||(OR)**, and **!(NOT)** are of highest precedence while the operators **and**, **or** and **not** are of lower precedence.

- The operators are enlisted in the following table -

Operator	Meaning
==	Equal to operator. If two operands are equal then it returns true.
!=	Not equal to. If the two operands are not equal then it returns true.
<	The less than
>	The greater than
<=	Less than equal to
>=	Greater than equal to
<=>	Compares the two operands
eql?	It returns true if the receiver object and the parameter both have the same type and values.
Equal?	Compares the two objects.

- The associativity of the operators is as shown in the following table. The highest precedence operators are listed first and then the lowest precedence operators.

Operator	Description	Associativity
**	Exponentiation	Right
! ~ unary + unary -	Not, complement, unary plus and minus	Right
* / %	Multiply, divide and modulo	Left
+ -	Plus and minus	Left
&	Bitwise `and'	Left
<= < > >=	Comparison operators	Not associative
<=> == === != =~ !~	Equality and pattern match operators (!= and !~ may not be defined as methods)	Not associative
&&	Logical `and'	Left
||	Logical `or'	Left

=, %=, /=, -=, +=, |=, &=, >>=, <<=, *=, &&=, ||=, **=	Assignment	Right
Not	Logical negation	Right
or and	Logical composition	Right

6.4.2 Selection and Looping Statements

Explain how multiple selection constructs are implemented in Ruby.

- The control statements are for controlling the flow of control of the program.
- The **if statement** is of following form -

```
Syntax:
If expression
  Statement
end
```

→

```
Example:
If count<10
  put's "Hello"
end
```

- The **if else statement** is -

```
Syntax:
If expression
        Statement
elsif expression
        Statement
else
        Statement
end
```

→

```
Example:
If age<5
        puts "Kid"
elsif age<21
        puts "Youth"
else
        puts "Adult"
end
```

- Ruby allows two kinds of **multiple selection constructs**. Both named **case**. The first form of **case** statement is as follows -

```
Syntax:
case
when value then
        statement sequence
when value then
        statement sequence
...
else Expression
end
```

→

```
Example:
case
when 66 then
        puts "Distinction"
when 60 then
        puts "First Class"
when 50 then
        puts "Pass Class"
else puts "Fail"
end
```

- Another form of case statement

```
case
when Boolean expression then
statement sequence
when Boolean expression then
statement sequence
...
else Expression
end
```

- The while and for statements are similar to C language. These are as follows -

Syntax:	Example:
while *control expression* *Loop body* end	while *count<10* *puts "The count =#{count}"* *count=count+1* end

- The for statement is as follows -

Syntax:	Example:
for variable [, variable ...] In expression [do] Loop body end	for i in 0..10 puts "Value of i is #{i}" end

- The until statement is also used for looping. The body of until statement executes when the condition is false.

Syntax

```
until conditional [do]
 loop body statements
end
```

Ruby Script[UntilDemo.rb]

```
$i = 0;
$n = 3;
until $i > $n do
  puts("Inside loop with i = #$i");
  $i +=1;
end
puts("Outside the loop with i=#$i")
```

Output

```
Inside loop with i = 0
Inside loop with i = 1
Inside loop with i = 2
Inside loop with i = 3
Outside the loop with i=4
```


The break statement is used to terminate the current execution. For example :

Ruby Script[BreakDemo.rb]

```
for count in 0..7
 if count > 2*2 then
   break
 end
 puts "count= #{count}"
end
```

Output

```
count= 0
count= 1
count= 2
count= 3
count= 4
```

Example 6.4.1 *Write a Ruby program that inputs the list of numbers from keyboard and finds the second smallest number in the list, along with its position in the list.*

Solution :

Ruby Program[SecondDemo.rb]

```
i=0
arr=Array.new
puts "Enter the elements for the list. Press Cntrl+z for to terminate the list"
while(val=gets)
 arr[i]=val
 i+=1
end
first = arr[0]
 second = arr[0]
 i=0
 for value in arr
   if value < first
     second = first
     secondkey=i
     first = value
   elsif value < second
      second = value
      secondkey=i
   end
   i+=1
 end
 puts "The smallest element is #{second}and the position is #{secondkey}"
```

Output

```
Command Prompt
F:\Ruby-Examples>ruby SecondDemo.rb
Enter the elements for the list. Press Cntrl+z for to terminate the list
30
20
10
40
50
^Z
The smallest element is 20
and the position is 2

F:\Ruby-Examples>
```

6.5 Fundamentals of Arrays

- In Ruby the arrays are more flexible because of two reasons -
  - The arrays in Ruby are **dynamic** that means the array can grow and shrink.
  - The arrays in Ruby can store the elements of different data types.
- The arrays can be created in two different ways.
- The first way is to use the keyword **new**. For example :
  - Marks =Array.new
  - Marks=Array.new(10) ← the array of 10 locations gets created.
- The second way is to assign the list of literals to the variables. For example :
  - Student=[10,20,30,"Archana",[ ]]
- The subscript of array elements is integer and every array starts at the subscript 0.
- The **length** method is used to retrieve the length of the array.
- The **at** method returns the element present at the pointed subscript.

Ruby Script [ArrayDemo.rb]

```
A=[10,20,30,40,50]
puts "The array is : #{A}"
num=A.at(3)
puts "The element at 3rd index is :#{num}"
len=A.length
puts "The length of the array is :#{len}"
```

Output

```
The array is : [10, 20, 30, 40, 50]
The element at 3rd index is :40
The length of the array is :5
```


6.5.1 The for-in Statement

- The for-in statement is used to visit the elements of an array. Following is a simple example that illustrates the use of for-in statement -

Ruby Script [ForInDemo.rb]

```
A=[10,20,30,40,50]
for i in [0,1,2,3,4]
puts "For index: #{i}, value=#{A[i]}"
end
```

Output

```
For index: 0, value=10
For index: 1, value=20
For index: 2, value=30
For index: 3, value=40
For index: 4, value=50
```

6.5.2 Built in Methods for Arrays and Lists

- In this section, we use various built in methods for supporting array operations. Open the Interactive Ruby interpreter and invoke various methods.
- The **shift** method removes the first element of an array. For example :

```
irb(main):001:0> A=[10,20,30,40,50]
=> [10, 20, 30, 40, 50]
irb(main):002:0> num=A.shift
=> 10
irb(main):003:0> A
=> [20, 30, 40, 50]
irb(main):004:0>
```

- The **push** method is used to push the elements from the end in the array. For example :

```
irb(main):001:0> A=[10,20,30]
=> [10, 20, 30]
irb(main):002:0> A.push(40,50)
=> [10, 20, 30, 40, 50]
irb(main):003:0>
```

- The two arrays can be combined together using the **+ operator**. For example :

```
irb(main):001:0> A=[10,20,30]
=> [10, 20, 30]
irb(main):002:0> B=[1.1,1.2,1.3]
=> [1.1, 1.2, 1.3]
irb(main):003:0> C=A+B
=> [10, 20, 30, 1.1, 1.2, 1.3]
irb(main):004:0>
```

- The **reverse** method is used to reverse the contents of the array. For example :

```
irb(main):001:0> A=[10,20,30,40,50]
=> [10, 20, 30, 40, 50]
irb(main):002:0> A.reverse
=> [50, 40, 30, 20, 10]
irb(main):003:0> A
=> [10, 20, 30, 40, 50]
irb(main):004:0>
```

- The **sort** method is used to sort the contents of the arrays. For example

```
irb(main):001:0> A=[30,10,20,50,40]
=> [30, 10, 20, 50, 40]
irb(main):002:0> A.sort
=> [10, 20, 30, 40, 50]
irb(main):003:0>
```

- The **include?** is used to check whether the array contains particular element or not.

```
irb(main):001:0> A=[10,20,30,40]
=> [10, 20, 30, 40]
irb(main):002:0> A.include?(20)
=> true
irb(main):003:0> A.include?(99)
=> false
irb(main):004:0>
```

6.5.3 Examples

Following is a simple Ruby program that sorts the array and then searches the desired element from the array.

Ruby Program

```
i=0
A=Array.new
puts "Enter the elements in the array"
while(i<5)
  val=gets
  A[i]=val
  i+=1
end
A.sort!
puts "The sorted array is "
for val in A
  puts val
end
puts "Serching the elements from an array"
puts "Enter the number to be searched"
```



```
key=gets
if A.include?(key)
  puts "The element #{key} is present in the list"
else
  puts "The element #{key} is not present in the list"
end
```

Output(Run1)

```
Enter the elements in the array
30
10
50
40
20
The sorted array is
10
20
30
40
50
Serching the elements from an array
Enter the number to be searched
40
The element 40
 is present in the list
```

(Run1)

```
Enter the elements in the array
30
20
10
50
40
The sorted array is
10
20
30
40
50
Serching the elements from an array

Enter the number to be searched
88
The element 88
 is not present in the list
```

6.6 Hashes

- A hash is a collection of key-value pairs like this : "Roll" =>10
- Hashes are also called as **associative arrays.**
- The hashes are similar to arrays but the only difference between the two is that,

Hashes	Arrays
The hash uses the string values as a subscript to address the elements. These string values are basically the keys.	The arrays use the integer values as a subscript to address the element.
The elements in hash are not ordered. The arrangement of elements in hash resembles the **set.**	The elements in array are ordered by subscript. The arrangement of elements in array resembles the **list.**

- The hash can be created using either the **new** method or by assigning the key-value pair. For example :

Ruby Program [HashDemo1.rb]

```
A = Hash.new( "one" )
puts "\tFirst Method of Hash Creation"
puts "#{A[0]}"
puts "#{A[3]}"
puts "#{A[5]}"
puts "\tSecond Method of Hash Creation"
B=Hash["AAA"=>100,"BBB"=>200,"CCC"=>300]
puts "#{B['AAA']}"
puts "#{B['BBB']}"
puts "#{B['CCC']}"
puts "\t Displaying the keys of the array"
keys=B.keys
puts "#{keys}"
```

Output

```
    First Method of Hash Creation
one
one
one
    Second Method of Hash Creation
100
200
300
    Displaying the keys of the array
["AAA", "BBB", "CCC"]
```


6.7 Methods

- Methods in Ruby are similar to the functions.
- When a block of statements that get repeated many times in the program then the method is used.
- The method is defined outside the class.
- In this section we will discuss how to write the methods in Ruby and call them.

6.7.1 Fundamentals

- The method definition includes **method header** and sequence of statements.
- The **method header** consists of the reserved word **def.**
- **The method name must begin with lower case.**
- The syntax of method definition is as given below -

```
def method_name ( argument list )
  method body
end
 or
def method_name
  method body
end
```

- If the method has no parameter then the parenthesis can be omitted.

Ruby Program[MethodDemo.php]

```
def mymethod
  puts " This is the method demo"
end
mymethod
```

Output

```
This is the method demo
```

Program Explanation :

- In above, program, we have defined the method named **mymethod.** The method is called only by its name. The keyword **def** is used to define the method.
- Every method in Ruby returns something. The values can be returned using the keyword **return** If there is no return statement then by default the last sentence of the method is returned. For example :

```
def mymethod1
  i=10
  j=20
end
puts mymethod1
```

The above code will display the output as 20.

- We can make use of the **return** statement to return the values.
- In Ruby we can return more than one values.

For example :

```
def mymethod2
  i=10
  j=20
  return i,j
end
puts mymethod2
```

Output

```
10
20
```

6.7.2 Local Variables

- Local variables are the variables that are created inside the method.
- The scope of the method during the definition of the method.
- The **advantage** of local variables in the method is that if the name of the local variable conflicts with the name of the global variable then the conflict gets resolved automatically by using the local variable.
- The name of the local variable must begin with either the lower case letter or by underscore.
- The lifetime of the local variable is the period of time over which it exists.

6.7.3 Parameters

- There are two types of parameters - **Actual parameter** and the **formal parameter.** For example :

Ruby Program[testmethod.rb]

```
def testmethod(arg1,arg2)
  puts "The first parameter is #{arg1}"
  puts "The second parameter is #{arg2}"
end
puts testmethod 10,20
```

Output

```
The first parameter is 10
The second parameter is 20
```


- Variable number of parameters can be handled by the method using the asterisk *
  For example :

Ruby Program [testmethod1.rb]

```
def testmethod1(*arg)
  puts "The total number of parameters are: #{arg.length}"
  for i in 0...arg.length
    puts "The parameters are #{arg[i]}"
  end
end
testmethod1 10,20,30
testmethod1 11.1,12.2,13.3,14.4,15.5
```

Output

```
The total number of parameters are: 3
The parameters are 10
The parameters are 20
The parameters are 30
The total number of parameters are: 5
The parameters are 11.1
The parameters are 12.2
The parameters are 13.3
The parameters are 14.4
The parameters are 15.5
```

- The call to the method must have same number of actual parameters as the number of formal parameters.

6.8 Classes

- Ruby is purely an object oriented programming language. Hence **class** is an important factor of this programming language.
- The objects are created from the classes. In other words we can say object is an instance of a class.

6.8.1 Basics of Classes

- Every class contains two things the variables and the functions. The class in the Ruby can be defined as follows -

```
class className
...
end
```

- The class name must begin with the capital letter. The keyword **class** can be used to define the class. The class must be terminated by the keyword **end.**

- In Ruby there are two types of variables in class -
  - **Instance Variables :** The instance variables are defined in the method. The instance variables change from object to object. These variables are available across the methods. The syntax of these variables is ,

```
@variable_name
```

  - **Class Variables :** The class variables belongs to the class. They are used to define the characteristics of the class. These are available across the objects. The syntax of these variables is,

```
@@variable_name
```

- **Objects** are the instances of the class and these can be created using the method **new.**

  For example :

```
Obj1= Myclass.new
```

- There is a method named **initialize** is a constructor of the class which initializes the values of the variables. Using the **def** and the **end** the method initialize can be defined.
- The name of the method must be starting with the lower case letter. It must be ended by the keyword **end.**

Example : Create a class for obtaining the roll number,name and marks of the student. Display the records.

```
class Student
  @@TotalStudents=0
  def initialize(roll, name, marks)
    @stud_roll=roll
    @stud_name=name
    @stud_marks=marks
  end
  def display()
    puts "Roll Number: #@stud_roll"
    puts "name: #@stud_name"
    puts "Marks: #@stud_marks"
  end
  def total()
    @@TotalStudents += 1
    puts "\tTotal number of students: #@@TotalStudents"
  end
end
S1=Student.new("1", "Shilpa", 97.11)
```

Class Variables

Instance Variables


```
S2=Student.new("2", "Rashmi", 89.21)
# Call Methods
S1.display()
S1.total()
puts "----------------------------------------"
S2.display()
S2.total()
```

Output

```
F:\Ruby-Examples>Ruby classDemo.rb
Roll Number: 1
name: Shilpa
Marks: 97.11
Total number of students: 1
----------------------------------------
Roll Number: 2
name: Rashmi
Marks: 89.21
Total number of students: 2
```

6.8.2 Access Control

- There are three access control for the methods - **Public, private and protected.**
- The default access control for the method is **public.**
- The **public** access method means that the method can be called by any code.
- The **protected** access means only objects of defining class and its subclasses may call the method.
- The **private** access means private methods of another object can not be accessed by one object.
- For example

Ruby Program [Test.rb]

```
class Test
def method1
puts "This is the first method"
end
private
def method2
puts "This is second method"
end
protected
def method3
```

```
    puts "This is third method"
    end
    end
    obj=Test.new()
    obj.method1
```

Output

```
This is the first method
```

Note that if you call the obj.method2 or obj.method3 then we will get the error.

6.8.3 Inheritance

- Inheritance is the most important feature of any object oriented programming language.
- By inheritance the two classes are created one **base class** and another **derived class.**
- Using the object of derived class one can access the methods of itself and the base class both.
- The syntax for declaring the derived class is,

```
class Derived_class_name < Base_class_name
```

For example :

Ruby Program [InheritanceDemo.rb]

```
class A
 def method1
  puts "I am in method1 of class A."
 end
 def method2
  puts "I am in method2 of class A."
 end
end

class B < A
 def method3
  puts "I am in method3 of class B."
 end
end
obj = B.new
obj.method1
obj.method2
obj.method3
```

Class B is derived from class A

Output

```
I am in method1 of class A.
I am in method2 of class A.
I am in method3 of class B.
```


Note that in above code, using the object of derived class we can access the methods of base class (i.e. method1 and method2) and the derived class (method3) as well.

6.9 Code Blocks and Iterators

- Block consists of collection of code.
- The block is always enclosed within { and }.
- Blocks can be used with methods.
- Using the iterator **each** we can display each element of the array and apply it to block.
- For example :

Ruby Program [BlockDemo.rb]

```
A=[10,20,30,40]
A.each{|num| puts "The number is #{num}"}
```

Output

```
The number is 10
The number is 20
The number is 30
The number is 40
```

- Using the iterator **step** the specific values can be displayed. For Example

Ruby Program [BlockDemo1.rb]

```
10.step(20,2){|num| puts "The number is #{num}"}
```

Starting value → 10; ending value → 20; Step size → 2

Output

```
The number is 10
The number is 12
The number is 14
The number is 16
The number is 18
The number is 20
```

- Using the **yield** method the block can be invoked.

- A block is always invoked from a function. But the name of the function and the name of the block must be the same in this case.
- For example :

Ruby Program [BlockDemo1.rb]

```
def myfun
  puts "This statement is in method"
  yield
  puts "Returning back to the method"
end
myfun {puts "This statement is in block"}
```

Block is called

Block is defined

Output

```
This statement is in method
This statement is in block
Returning back to the method
```

6.10 Pattern Matching

6.10.1 Basics of Pattern Matching

- For pattern matching the operator =~ is used.
- The pattern can be placed within /
- For example :
- Open the Interactive Ruby interpreter and type the following commands and observe the results.

```
irb(main):001:0> T="Success"
=> "Success"
irb(main):002:0> T=~/es/
=> 4
irb(main):003:0>
```

- The **split** method is used to separate out the words from the given string. For example following commands on **Interactive Ruby interpreter** will show how to separate out the words from the string using split command.


```
irb(main):005:0> s="hello friends how are you"
=> "hello friends how are you"
irb(main):006:0> w=s.split(/[ ,.]\s*/)
=> ["hello", "friends", "how", "are", "you"]
irb(main):007:0>
```

6.10.2 Remembering Matches

- Ruby allows **to save** the **matched part** of the **pattern** in the variable. Later on we can use this variable to get the desired part of the pattern.
- For example - Open the Interactive Ruby interpreter and type the following command and observe the results -

```
irb(main):001:0> pattern="Anuradha-19999"
=> "Anuradha-19999"
irb(main):002:0> pattern=~ /(\w+)-(\d+)/
=> 0
irb(main):003:0> puts "First string is #{$1}"
First string is Anuradha
=> nil
irb(main):004:0> puts "Second string is #{$2}"
Second string is 19999
=> nil
irb(main):005:0>
```

- In above example we have first stored the pattern in a variable say **pattern**. Then this pattern is matched with the regular expression. The \w stands for matching the words. The \d stands for matching the digits. Thus we get the groups of substrings from the given pattern. Then using $1,$2,... we can retrieve particular part of the regular expression. The $1 stands for the value of first matched substring, $2 stands for the value of second matched string.

6.10.3 Substitutions

- There are two methods **sub** and **gsub** which are for searching the desired pattern and replacing them.
- The **sub** method searches the first part of the pattern and replace it.

- For example :

```
irb(main):001:0> mystr="PHP is a fun.Enjoy PHP"
=> "PHP is a fun.Enjoy PHP"
irb(main):002:0> mystr.sub(/PHP/,"Ruby")
=> "Ruby is fun.Enjoy PHP"
irb(main):003:0>
```

Note that the first occurrence of PHP is replaced by the word Ruby and the second occurrence remains

- The **gsub** method is similar to **sub** but it replaces all the occurrences of matched substring. For example :

```
irb(main):001:0> mystr="PHP is a fun.Enjoy PHP"
=> "PHP is a fun.Enjoy PHP"
irb(main):002:0> mystr.gsub(/PHP/,"Ruby")
=> "Ruby is a fun.Enjoy Ruby"
irb(main):003:0>
```

Note that the all the occurrences of PHP are replaced by the word Ruby and the second occurrence remains unchanged.

- Using **sub** or **gsub** have the mutator versions and those are **sub!** and **gsub!**. These versions replace the original string also. For example :

```
irb(main):001:0> str="PHP is a fun. I like PHP"
=> "PHP is a fun. I like PHP"
irb(main):002:0> str.sub!(/PHP/,"Ruby")
=> "Ruby is a fun. I like PHP"
irb(main):003:0> str
=> "Ruby is a fun. I like PHP"
irb(main):004:0> str1="Java is OOP.I program in Java"
=> "Java is OOP.I program in Java"
irb(main):005:0> str1.gsub!(/Java/,"Ruby");
irb(main):006:0> str1
=>"Ruby is OOP.I program in Ruby"
```

Part II : Introduction to Rails

6.11 Overview of Rails

- Rails is a development framework for web - based applications. As Rails are closely associated with Ruby it is often referred as **Ruby on Rails** or simply RoR.
- Rails was developed by **David Heinemeier Hansson** in early 2000 and was released in July 2004.

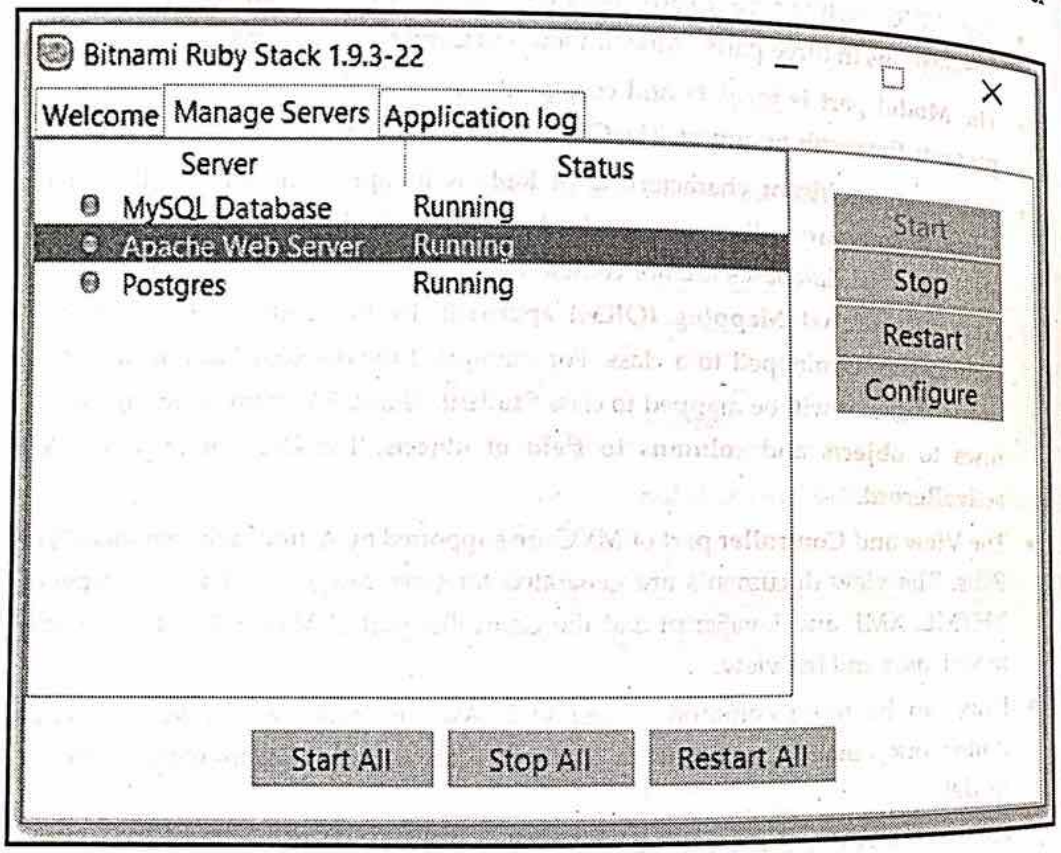
- The web application can be developed ten times faster with Rails than with a typical Java framework.
- Rails is based on **MVC architecture** - i.e. Model View Controller.
- The MVC architecture clearly separates the logical and physical parts of the applications in three parts - Model, View and Controller.
- The **Model** part is for data and constraints on data. The **View** part prepares and presents the result or output. The **Controller** controls the application.
- The most significant characteristic of Rails is its approach to connecting object oriented software with relational database. But both the object oriented software and relational databases are not compatible to each other. Hence Rails make use of **Object-Relational Mapping (ORM) approach**. By this each relational database table is simply mapped to a class. For example if the database has a table named **Student** then it will be mapped to class **Student**. Thus ORM maps **tables to classes, rows to objects** and **columns to field of objects**. The ORM of rails is called **activeRecord**.
- The **View** and **Controller** part of MVC are supported by **ActionPack** components of Rails. The view documents are generated for three categories of web documents XHTML, XML and JavaScript and the Controller part of MVC interacts with data model, user and the view.
- Rails can be more commonly used with Ajax. It makes use of the JavaScript framework named prototype to support Ajax and interaction with JavaScript model.
- Rails uses MVC architecture, consists of designing and building three parts of an MVC system.

6.12 Document Requests

- Rails is a software development **framework**. The software development framework consists of library of components that provide commonly needed services to an application.
- The easiest method of downloading the and installing Rails is to make use of a complete software system such as **InstantRails** or **Ruby Stack**.
- The Ruby Stack can be downloaded and installed from http://bitnami.com/stack/ruby. It can be downloaded on Windows/ Linux or MacOS platform. It consists of Ruby, Rails, Apache, MySQL, SQLite, Git, PHP and

phpMyAdmin in one pack. Installing Ruby Stack is quick and easy. It is a **self-contained** system.

- After installing the Ruby Stack we can start the Apache Server and MYSQL Server using the **Managers**. The screenshot is as follows.



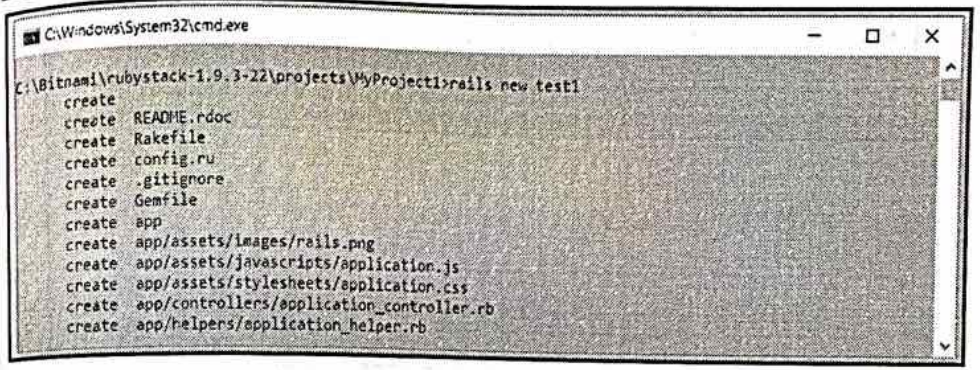
6.12.1 Static Documents - Creation of First Application in Rails

In this section, we will see the step by step method of building the first application in Rails. The **Ruby Stack** is installed on my machine at *C:\BitNami\rubystack-1.9.3-22*.

Step 1 : Go to the **Projects** directory within the Ruby Stack's directory. Create a folder of any name for storing various applications. I have created a folder named **MyProject1**. This is the base directory in which the developer can add new Rails applications. Now we will create another subdirectory say **test1** by issuing the following command

```
> rails new test1
```

Following screenshot illustrates it.



- Rails respond to this command by generating number of files and directories. Out of these directories the most important directory from the application point of view is **app**. It has four subdirectories namely **models, views, controllers and helpers**. The **models, views and controllers** are those directories that will support MVC of Rails.
- For the sake of understanding just look at the directory structure as given below -

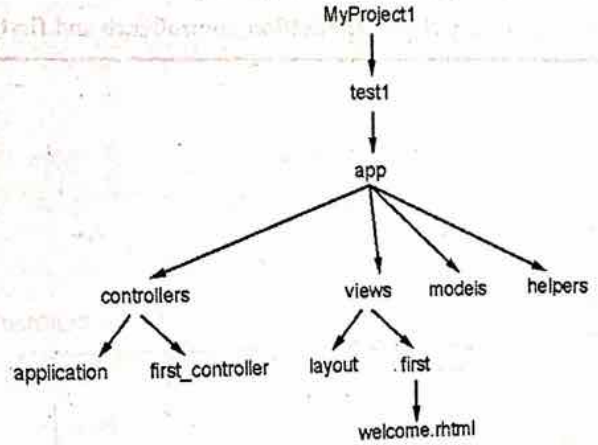


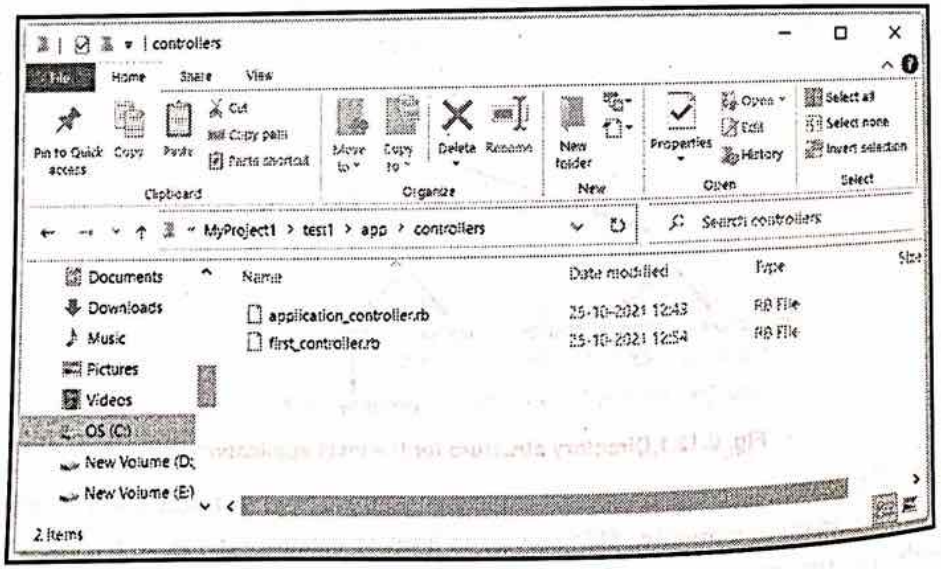
Fig. 6.12.1 Directory structure for the test1 application

Step 2 : This directory contains several important Ruby scripts that perform the services. One of the service is **generate**. This service is used to create or generate the controller methods. The two parameters are passed to this service - first is **controller** and second is the **name chosen for the controller**. The **controller** indicates that we want the **controller** class to be built. The command will be issued from the **test1** directory. The command will be as follows -

```
> rails generate controller first welcome
```

```
C:\Windows\System32\cmd.exe
operable program or batch file.

C:\Bitnami\rubystack-1.9.3-22\projects\MyProject1\test1>rails generate controller first welcome
      create  app/controllers/first_controller.rb
       route  get "first/welcome"
      invoke  erb
      create    app/views/first
      create    app/views/first/welcome.html.erb
      invoke  test_unit
      create    test/functional/first_controller_test.rb
      invoke  helper
      create    app/helpers/first_helper.rb
      invoke    test_unit
      create      test/unit/helpers/first_helper_test.rb
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/first.js.coffee
      invoke    scss
      create      app/assets/stylesheets/first.css.scss

C:\Bitnami\rubystack-1.9.3-22\projects\MyProject1\test1>
```

- Due to the above command the **FirstController** class will be generated in **first_controller.rb** at location **app/controllers/**
- There are now two Ruby classes **application_controller.rb** and **first_controller.rb**.



The **first_controller.rb** can be opened up in Notepad/Wordpad . It contains -

```
class FirstController < ApplicationController
end
```

Here **FirstController** is an empty class and is inherited from **ApplicationController** class.

These Controllers are useful for producing the HTTP responses. To respond the request there must be some **actionmethod**. This actionmethod simply indicate the document as a response. The **first_controller.rb** will be as given below in appearance -

```
class FirstController < ApplicationController
 def welcome
 end
end
```

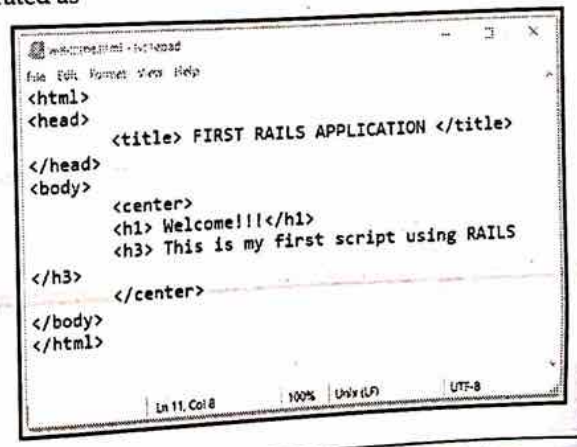
Step 3 : In above step the welcome is the action method name. It is the response to the URL http://localhost:3000/first/welcome.

This is actually the view document. The view document is called as **template**. These templates will be placed in **app/views** directory. The template will be written as an XHTML file. The extension to this file will be **.rhtml (or it can be html)**. The Ruby code can be inserted in this file. Hence the extension is derived from the word Ruby HTML. For our first application we can write the template as follows[Note that there is no Ruby code embedded in the following rhtml file, it simply makes use of HTML tags].

welcome.rhtml

```
<html>
<head>
<title> FIRST RAILS PROGRAM</title>
</head>
<body>
<center>
<h1>Welcome!!!</h1>
<h3>This is my First Program using Rails </h3>
</center>
</body>
</html>
```

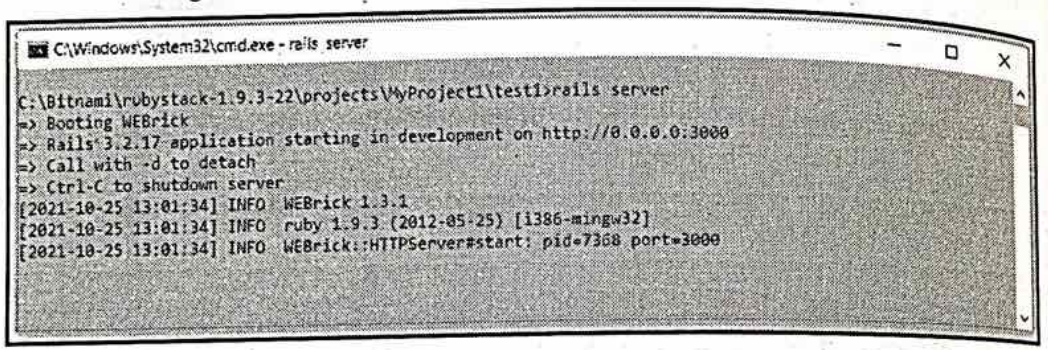
It can be illustrated as –



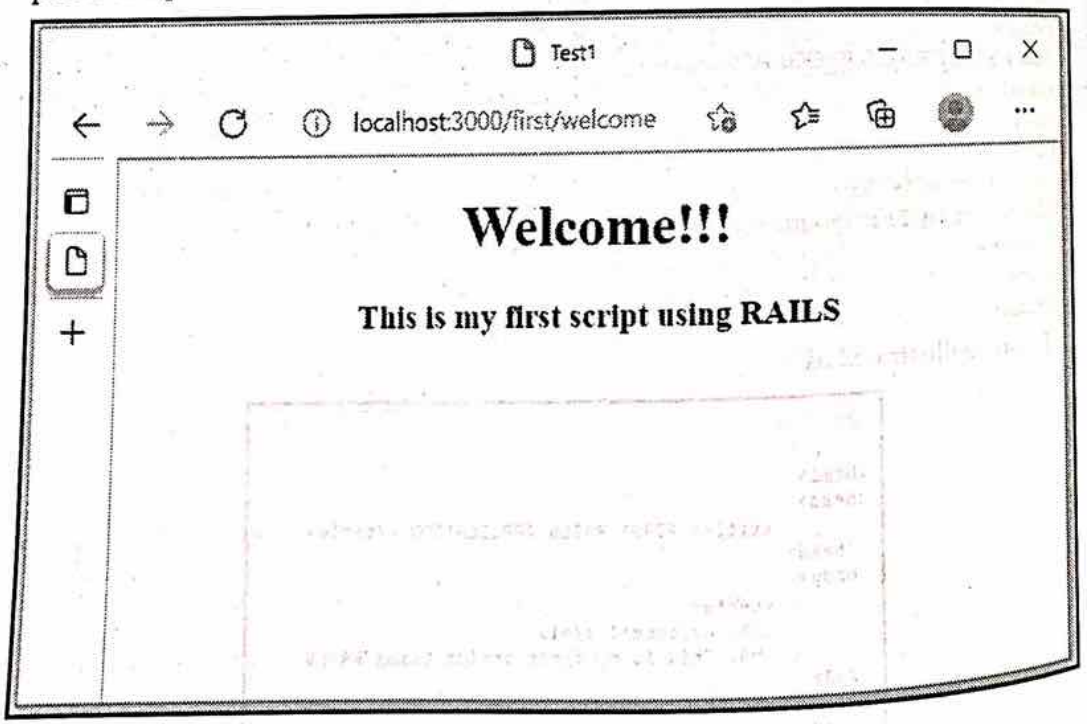
Step 4 : Now we will start the server for running the above created static document application by issuing the following command

```
> rails server
```

The following screenshot illustrates this command.



Now open suitable web browser and type the URL http://localhost:3000/first/welcome on the address bar. [Note that the URL specified during the installation is **localhost** and port is 3000]



6.12.2 Dynamic Documents

- Dynamic document is a kind of web document in which the contents are changing more often. For example if your web page is displaying the current date and time or stack market position or Cricket match score then that document is said to be a dynamic document.
- In this section we will build an application in which the dynamic web document is created for displaying the current date and time. The implementation of such web document is possible by embedding some Ruby code within it.
- The **Ruby code** can be **embedded** within the template file by using <% and %> markers.
- If the Ruby code produces some result then the result can be inserted within the template document by <%= and %> marker.
- Ruby's **Time.now** is used to display the current time. In earlier section we have seen how to display the static web document. We will follow steps 1 and 2 from the above section. The template document created in step 3 is slightly changed. It is as follows -

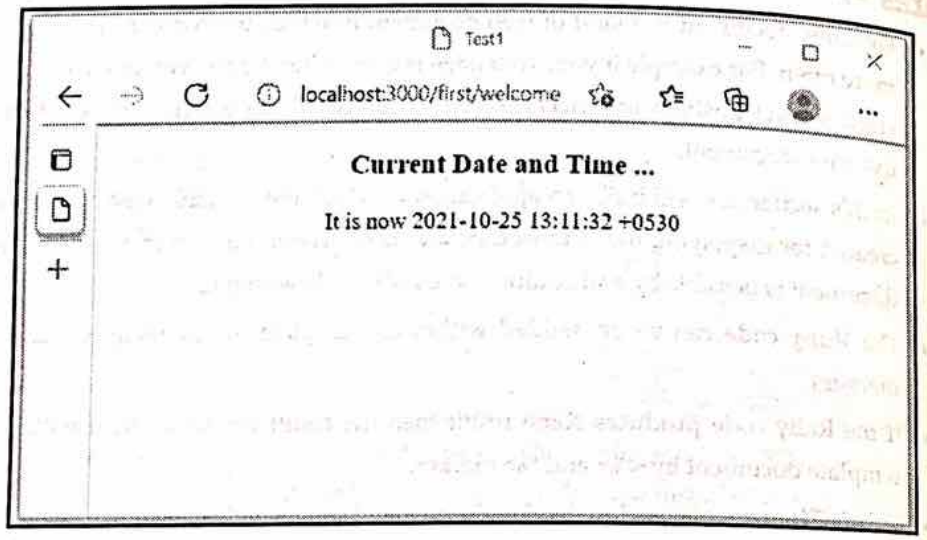
welcome.html

```
<html>
<head>
<title> FIRST RAILS PROGRAM</title>
</head>
<body>
<center>
<h3>Current Date and Time ... </h3>
<p> It is now <%= t = Time.now %> </p>
</center>
</body>
</html>
```

- Now start the server by going into the test1 directory by the command -

```
>Rails server
```

Open the suitable web browser and get the output as follows -



6.13 Processing Forms

Form is an useful tool used in building the GUI using which the user can interact with the system.

We will follow the following steps in processing form using Rails.

Step 1 : We have already created a folder named **MyProject1** in the **Projects** directory of **Ruby Stack**. Now go to the MyProject1 directory and create a **sampleForm** subdirectory using following command by using the command prompt -

```
C:\Windows\System32\cmd.exe
C:\Bitnami\rubystack-1.9.3-22\projects\MyProject1>rails new sampleForm
      create
      create  README.rdoc
      create  Rakefile
      create  config.ru
      create  .gitignore
      create  Gemfile
      create  app
      create  app/assets/images/rails.png
      create  app/assets/javascripts/application.js
      create  app/assets/stylesheets/application.css
      create  app/controllers/application_controller.rb
      create  app/helpers/application_helper.rb
      create  app/mailers
      create  app/models
      create  app/views/layouts/application.html.erb
      create  app/mailers/.gitkeep
      create  app/models/.gitkeep
```

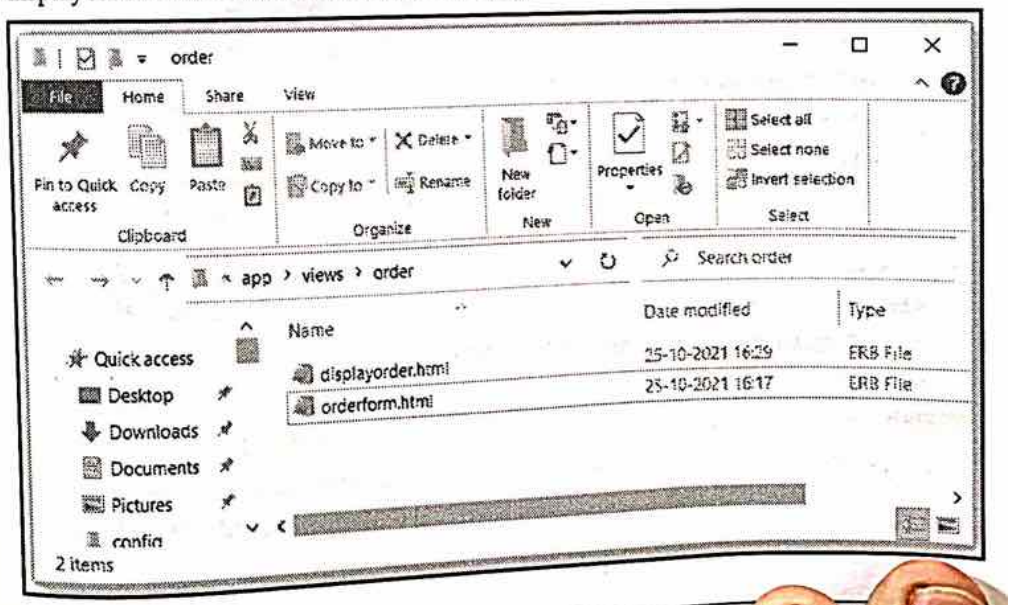

Now the **sampleForm** will have all those necessary directories such as **app, config, script** and so on generated within it.

Step 2 : Now go to the **sampleForm** directory and issue the following command on the command prompt for generating controller.

>rails generate controller order orderform displayorder

```
C:\Windows\System32\cmd.exe
C:\Bitnami\rubystack-1.9.3-22\projects\MyProject1>cd sampleForm

C:\Bitnami\rubystack-1.9.3-22\projects\MyProject1\sampleform>rails generate controller order orderform displayorder
      create  app/controllers/order_controller.rb
       route  get "order/displayorder"
       route  get "order/orderform"
      invoke  erb
      create    app/views/order
      create    app/views/order/orderform.html.erb
      create    app/views/order/displayorder.html.erb
      invoke  test_unit
      create    test/functional/order_controller_test.rb
      invoke  helper
      create    app/helpers/order_helper.rb
      invoke    test_unit
      create      test/unit/helpers/order_helper_test.rb
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/order.js.coffee
      invoke    scss
      create      app/assets/stylesheets/order.css.scss

C:\Bitnami\rubystack-1.9.3-22\projects\MyProject1\sampleform>
```

This will generate **order_controller.rb** file. Also create orderform.html and displayorder.html files in the **order** sub-directory.



Step 3 : The code for orderform.html file is as follows -

orderform.html

```
<html>
  <head>
    <title> Submitted Order</title>
  </head>
  <body>
  <h3> User Name: <%= @UserName %> </h3>
  <table border="1">
    <tr>
    <td><b>Product Name</b></td>
  <td><b>Price(Rs.)</b></td>
  <td><b>Quantity</b></td>
  <td><b>Cost</b></td>
    </tr>
    <tr>
      <td>Apples</td>
      <td>10 </td>
      <td><%= @qty1 %></td>
      <td><%= @apple_cost %></td>
    </tr>
    <tr>
      <td>Oranges</td>
      <td>7.50 </td>
      <td><%= @qty2 %></td>
      <td><%= @orange_cost %></td>
    </tr>
  </table>
  <br/>
  <h2> Total Cost = <%= @total_cost %> </h2>
  </body>
</html>
```


Step 4 : Edit the controller file **order_controller.rb** as follows -

order_controller.rb

```
class OrderController < ApplicationController
 def orderform
 end
 #displayorder method - it fetches the data from form and compute
 #total cost

 def displayorder
 @UserName =params[:UserName]
 @qty1 = params[:qty1].to_i
 @qty2 = params[:qty2].to_i

 #Computing the cost of each item
 @apple_cost = 10 * @qty1
 @orange_cost = 7.50 * @qty2

 #computing the total cost of purchased items
 @total_cost = @apple_cost+@orange_cost

 #Converting the values into strings with two digits
 @total_cost = sprintf("%4.2f",@total_cost)
 @apple_cost = sprintf("%4.2f",@apple_cost)
 @orange_cost = sprintf("%4.2f",@orange_cost)
 end
end
```

Invokes orderform.html

Invokes displayorder.html

Reading the input from the orderform.html and preparing the computed cost variables for displaying the result.

- Using the **params** symbol the parameters on the input rhtml file can be invoked. The **.to_i** method is used to convert the string to integers.
- The **sprintf** function of C can be used to format the numbers in two decimal places. This is done for displaying the floating point data in formatted manner.
- Save this file and close it. By the above code we have actually written the response methods namely **orderform** and **displayorder**. The **orderform.html** and **displayorder.html** files will be invoked as a response to these action methods. Out of these the **orderform.html** is already created. Now we will created **displayorder.html** web document.

Step 5 : Open **app/views/order** and create a file named **displayorder.html.** It is as follows -

displayorder.html

```
<html>
    <head>
      <title> Submitted Order</title>
    </head>
    <body>
    <h3> User Name: <%= @UserName %> </h3>
    <table border="1">
      <tr>
       <td><b>Product Name</b></td>
   <td><b>Price(Rs.)</b></td>
   <td><b>Quantity</b></td>
   <td><b>Cost</b></td>
      </tr>
      <tr>
        <td>Apples</td>
        <td>10 </td>
        <td><%= @qty1 %></td>
        <td><%= @apple_cost %></td>
      </tr>
      <tr>
        <td>Oranges</td>
        <td>7.50 </td>
        <td><%= @qty2 %></td>
        <td><%= @orange_cost %></td>
      </tr>
    </table>
    <br/>
    <h2> Total Cost = <%= @total_cost %> </h2>
    </body>
</html>
```

Step 6 : Now open command prompt and start the server using the command.

```
>rails server
```

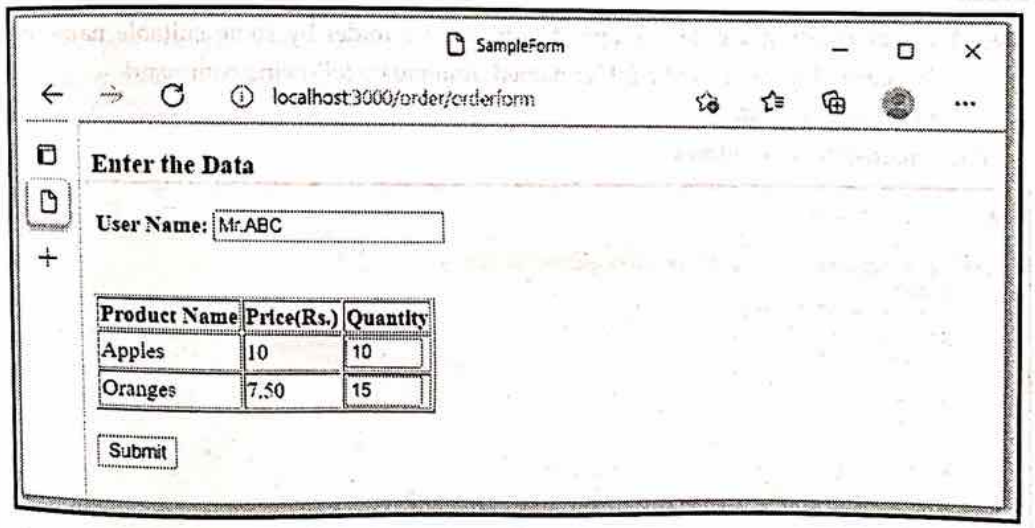


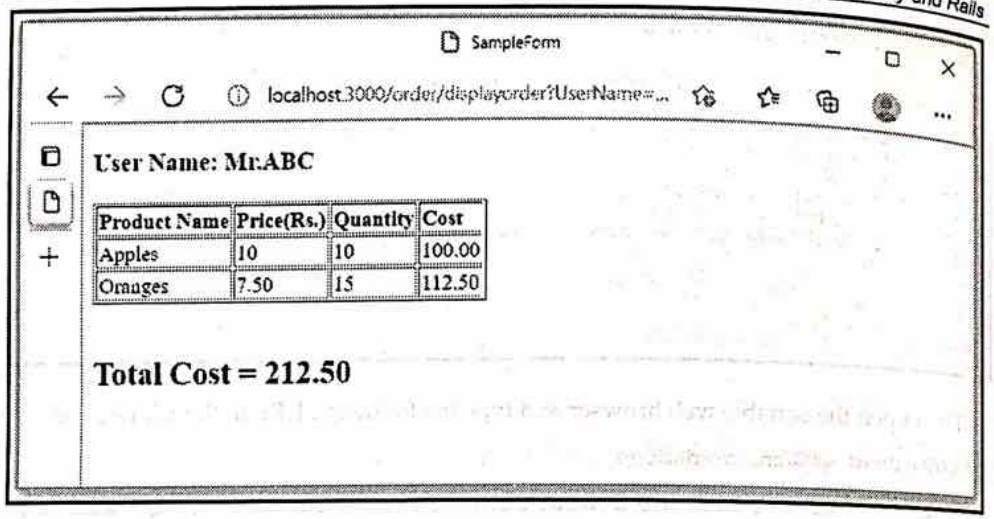
```
C:\Windows\System32\cmd.exe - rails server

C:\Bitnami\rubystack-1.9.3-22\projects\MyProject1\sampleForm>rails server
=> Booting WEBrick
=> Rails 3.2.17 application starting in development on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server
[2021-10-25 13:52:08] INFO  WEBrick 1.3.1
[2021-10-25 13:52:08] INFO  ruby 1.9.3 (2012-05-25) [i386-mingw32]
[2021-10-25 13:52:08] INFO  WEBrick::HTTPServer#start: pid=1520 port=3000
```

Then open the suitable web browser and type the following URL in the address bar.

```
http://localhost:3000/order/orderform
```

Enter the data and click the **Submit** button on that form. You will get following output.





6.14 Rails Applications and Databases

Step 1 : For creating a database application create a folder by some suitable name in Ruby's directory. I have created a folder named **dbdemo** by following command

```
> rails new dbdemo
```

The screenshot is as follows -

```
C:\Bitnami\rubystack-1.9.3-22\projects>rails new dbdemo
      create
      create  README.rdoc
      create  Rakefile
      create  config.ru
      create  .gitignore
      create  Gemfile
      create  app
      create  app/assets/images/rails.png
      create  app/assets/javascripts/application.js
      create  app/assets/stylesheets/application.css
      create  app/controllers/application_controller.rb
      create  app/helpers/application_helper.rb
      create  app/mailers
      create  app/models
      create  app/views/layouts/application.html.erb
      create  app/mailers/.gitkeep
```

By this command a directory **dbdemo** will be created with several useful subdirectories such as **app, config, doc, db, script** and so on.

Step 2 : Now generate the **controller** by the following command in **dbdemo** directory

```
> rails generate controller mybook
```



```
C:\Bitnami\rubystack-1.9.3-22\projects\dbdemo>rails generate controller mybook
      create  app/controllers/mybook_controller.rb
      invoke  erb
      create    app/views/mybook
      invoke  test_unit
      create    test/functional/mybook_controller_test.rb
      invoke  helper
      create    app/helpers/mybook_helper.rb
      invoke    test_unit
      create      test/unit/helpers/mybook_helper_test.rb
      invoke  assets
      invoke    coffee
      create      app/assets/javascripts/mybook.js.coffee
      invoke    scss
      create      app/assets/stylesheets/mybook.css.scss

C:\Bitnami\rubystack-1.9.3-22\projects\dbdemo>_
```

So a controller named **mybookController** will be created within the fil **mybook_controller.rb** of **controller** directory.

Step 3 : Now we will create a database in the MySQL by following command

```
>mysql -u root -p;
```

It will prompt for password if any. Then the **mysql** prompt will appear on the prompt window.

```
C:\Bitnami\rubystack-1.9.3-22\mysql\bin>mysql -u root -p
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.5.34 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

Step 4 : Now create a database named **bookinfo_development** using CREATE DATABASE command. Following screenshot illustrates it.

```
C:\Windows\System32\cmd.exe - mysql -u root -p
mysql> CREATE DATABASE bookinfo_development;
Query OK, 1 row affected (0.01 sec)

mysql>
```

Then create tables **books** and **subject**. Following screenshot illustrates the creation commands.

```
C:\Windows\System32\cmd.exe - mysql -u root -p
mysql> CREATE TABLE books (
    -> accNo INT(5) NOT NULL AUTO_INCREMENT,
    -> title CHAR(20),
    -> author CHAR(20),
    -> edition INT(3),
    -> publication CHAR(15),
    -> title_id INT(4) NOT NULL,
    -> PRIMARY KEY(accNo)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql>
```

```
C:\Windows\System32\cmd.exe - mysql -u root -p
mysql> CREATE TABLE subject (
    -> id INT(4) NOT NULL AUTO_INCREMENT,
    -> subject_name CHAR(20),
    -> PRIMARY KEY(id)
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
```


Step 5 : Now we will generate the model files that are required for this application. Use following command for this.

```
C:\Windows\System32\cmd.exe
C:\Bitnami\rubystack-1.9.3-22\projects\dbdemo>rails generate model book
      invoke  active_record
      create    db/migrate/20211026072342_create_books.rb
      create    app/models/book.rb
      invoke    test_unit
      create      test/unit/book_test.rb
      create      test/fixtures/books.yml

C:\Bitnami\rubystack-1.9.3-22\projects\dbdemo>rails generate model subject
      invoke  active_record
      create    db/migrate/20211026072358_create_subjects.rb
      create    app/models/subject.rb
      invoke    test_unit
      create      test/unit/subject_test.rb
      create      test/fixtures/subjects.yml

C:\Bitnami\rubystack-1.9.3-22\projects\dbdemo>
```

These command will generate the files namely book.rb and subject.rb in app/model directory.

Open these files using the text editor and edit them as follows -

book.rb

```
class Book < ActiveRecord::Base
 belongs_to :subject
end
```

subject.rb

```
class Subject < ActiveRecord::Base
 has_many :books
end
```

The **has_many** and **belongs_to** are the relationships used in MySql for specifying the classes associated with the database tables. When these relationships are provided then Rails generate the capabilities to manipulate the given relationship.

Step 6 : Now insert the data into tables **books** and **subject**. Following commands illustrates the insertion of data.

```
C:\Windows\System32\cmd.exe - mysql -u root -p
mysql> INSERT INTO books VALUES
    -> (10,"Quantum Physics","AAA",2,"XXX",5),
    -> (20,"Organic Chemistry","BBB",1,"ZZZ",3),
    -> (30,"Mental Maths","CCC",1,"YYY",2);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
C:\Windows\System32\cmd.exe - mysql -u root -p
mysql> INSERT INTO subject VALUES
    -> (5,"Physics"),
    -> (2,"Chemistry"),
    -> (3,"Mathematics");
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Step 7 : Now open the controller file which is present in the controller folder. Edit it as follows -

mybook_controller.rb

```
class MybookController < ApplicationController
    def welcome
    end
    def show
    @titlename = params[:title]
    @bookarr = book.find(:all, :conditions => ["title =?",@titlename])
    @subarr = subject.find(:all)
end
```

Step 8 : The **welcome.rhtml file** can be stored in app / views directory. It will be as follows -

```
<html>
<head><title> RAILS-MYSQL DEMO </title></head>
<body>
<form method="post" action="show">
 <input type="text" name="title" size="20"/>
 <input button="submit" name="Display" />
</form>
</body>
</html>
```

The **show.rhtml** file can be stored in app / views directory. It will be as follows -

```
<html>
   <head>
     <title> Book Display</title>
   </head>
   <body>
   <table border="1">
     <tr>
      <td><b>AccessNo</b></td>
  <td><b>Title</b></td>
  <td><b>Author</b></td>
  <td><b>Edition</b></td>
```



```
  <td><b>Publication</b></td>
  <td><b>Subject</b></td>

    </tr>
      <% @bookarr.each do |b|
      @accno=b.accNo
      @title=b.tilte
      @author=b.author
      @edition=b.edition
      @pub=b.publication
      @subject_row=@subarr[b.title_id]
      @subjectname=@subject_row. subject_name %>
     <tr>
      <td><b><%= accno %></b></td>
      <td><b><%= title %></b></td>
      <td><b><%= author %></b></td>
      <td><b><%= edition %></b></td>
      <td><b><%= pub %></b></td>
      <td><b><%= subjectname %></b></td>
     </tr>
     <% end %>
   </table>
   </body>
<html>
```

6.15 Layouts

- The **app/View** directory of each application has two directories, one is for the controller and another is named as **layouts.**
- A layout defines the surroundings of an HTML page. In layout directory you can place a file for common look and feel of your final output. For that purpose layout template is created.
- Rails must be informed about the name of the layout document by placing the **layout** command in the **ApplicationController** class of the application.
- In the **controller** directory of **app** there is a file named **application_controller.rb** in this file the following code can be placed -

```
class ApplicationController < ActionController::Base
 layout "mainlayout"
end
```

- Now if we add **mainlayout.rhtml** document to layout directory then the basic layout of the application can be defined.

6.16 Rails with Ajax

- AJAX stands for Asynchronous JavaScript and XML. It allows client side changes without reloading the page.
- When you type **http://localhost:3000** into your browser's address bar and hit 'Go', the browser (your 'client') makes a request to the server. It parses the response, then fetches all associated assets, like JavaScript files, stylesheets and images.
- It then assembles the page. If you click a link, it does the same processs : Fetch the page, fetch the assets, put it all together, show you the results. This is called the 'request response cycle'.
- JavaScript can also make requests to the server, and parse the response. It also has the ability to **update information** on the page. Combining these two powers, a JavaScript writer can make a web page that can **update just parts of itself**, without needing to get the full page data from the server. This is a powerful technique is called as Ajax.
- For example -

```
fetch("/index")
 .then((data) => data.text())
 .then((html) => {
  const results = document.querySelector("#compute");
  results.insertAdjacentHTML("before_exit", data);
 });
```

- This code fetches data from "/index" and then appends the result to the element with an id of **compute.**
- Rails provides quite a bit of built-in support for building web pages with this technique. Rails uses a technique called "Unobtrusive JavaScript" to handle attaching JavaScript to the DOM. This is a technique by which we can mix JavaScript code into HTML. With this, we can easily add behavior to any link by adding data attribute.

Part III : EJB

6.17 Introduction to EJB

SPPU : May-18, 19, Dec.-18, Marks 8

- J2EE application container contains the components that can be used by the clients for executing the business logic. These components are known as Enterprise Java Beans (EJB).
- EJB mainly contains the business logic and business data.

- Using the technology like EJB the server side, distributed Java components can be built.
- EJBs provide a core set of system services to Java component. **EJB components** always lie in some container which is called as **EJB container.**
- EJB component is an EJB class. It is a Java class written by EJB developer and this class implements business logic.
- EJB component is similar to traditional Java beans but there are some difference between the two. These differences are as given below -

Java Beans	EJB
The java bean components can be visible or invisible to the user.	The EJB components are always invisible to the user.
The Java beans normally run locally.	EJB are distributed components and always exist on the servers.
The Java beans can be used as Activex Control.	EJBs can't be used as Activex Control.
The Java beans can be described with properties, BeanInfo classes and Customizers.	EJBs can be described using DeploymentDescriptor.

6.17.1 Types of EJB

- Following are the types of EJB.

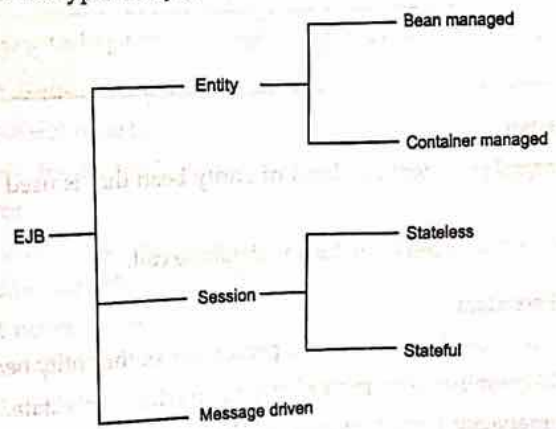


Fig. 6.17.1 Types of EJB

Entity Bean

- Entity beans are used for modelling the business concepts. Therefore entity beans represent the data. For example entity bean can represent the bank account or a customer. The **nouns** that are used in the problem description refer to the entity beans. Basically the entity beans represent the **real world objects.**
- Such beans then act as a **persistent record** in the database.

Session Bean

- Session beans are used for managing processes or tasks. For example *Compute net profit* or *transaction* or *browsing for some information on the web browser*. Hence session beans are used for managing activities. Basically session beans represent the **activities**. Hence session beans are **transient**, because activities generally exist for a short period. Session bean does not represent anything in the database but it can access the database. Session beans are client specific.

What are the differences between entity bean and session bean ?

Sr. No.	Entity bean	Session bean
1.	These are **persistent.**	These are **transient.**
2.	It refers to the **records** in the database.	These refers to the **activities** for the database.
3.	These can **survive** on server crash. (After server gets online these can be reconstructed from the underlying data).	Session objects generally **cannot survive** on server crashes.
4.	Entity beans are **client-independent** entities.	Session beans are **client-specific** entities.

Bean managed Persistent

- The bean managed persistent is a kind of entity bean that is used for saving bean's state.
- The container does not need to make any database call.

Container managed Persistent

- This kind of bean is used to manage the persistence of the entity bean.
- That means EJB container is responsible for saving the bean's state.
- For container managed persistent bean implementation is independent of the data source.

Stateless

- Stateless bean is designed for simple and lightweight components.
- This brings the scalability in server performance. As no state is maintained in this type of bean it is called stateless bean.
- The stateless beans are not tied to any specific client. Hence any available instance of stateless session bean can be used to service a client.

Stateful

- The stateful bean is used to store the internal states of the EJB.
- There is only one stateful session bean per EJB client.
- These types of beans can be saved across client sessions. And we can restore them back.

Message driven Bean

- Message driven is similar to the session bean but it gets activated only when asynchronous message arrives.
- A message-driven bean's instances retain no data or state for a specific client.
- A single message-driven bean can process messages from multiple clients.
- When a message arrives then the EJB container calls the message driven bean's onMessage method to process the method.

6.17.2 Benefits of EJB

1. Business to Business (B2B) e-commerce applications make use of EJB.
2. EJB can be used in web-centric as well as web-oriented applications.
3. Use of EJB is more suitable when there is large number of clients involved.
4. The Enterprise Application Integration (EAI) can use EJB for processing and mapping different applications. There, encapsulation of business logic with the data is possible due to EJB.
5. They can be customized at deployment time by editing the deployment descriptor.
6. System-level services, such as transaction management and security, are described separately from the enterprise bean.
7. A client never accesses an enterprise bean directly; the container environment mediates access for the client. This provides component-location transparency.
8. The EJB is designed to be portable across EJB servers provided by different vendors.
9. They can be included in an assembled application without requiring source code changes or recompilation of them.

6.17.3 Architecture

Components of EJB

- The EJB container provides following services to EJB component -
  - Security
  - Transactions
  - Messaging
  - Persistence
  - Distributions
  - Connectivity
- The idea behind developing EJB is to provide a framework for the components that can be plugged into a server. This ultimately helps to increase the functionality of the server.
- EJB is a managed component that can be created, controlled and managed by its container J2EE.
- There can be multiple EJBs in the container J2EE. The container J2EE provides the resources (such as memory and database connections) required by the EJB.
- Thus J2EE maintains a pool of EJB instances. These instances can be assigned to the clients.
- Client makes use of these EJB components and when it is no longer required by the client, these instances can be returned to the pool so that other clients can used these instances. The client who is using the EJB instance does not need to know about its container.

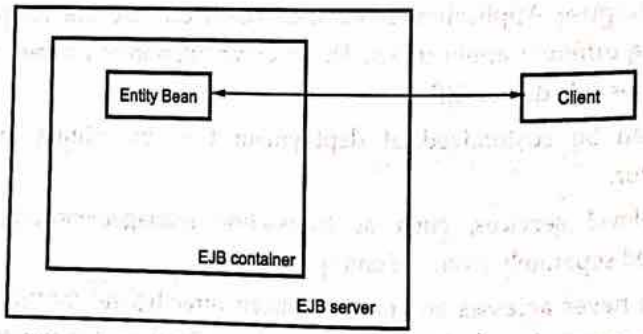


Fig. 6.17.2

- EJB has a technique that keeps business logic separate from the database. In other words, the business logic can deal with business data without caring about how data is stored in database.

- There are two main purposes of the container firstly control the EJB it contains and secondly provide the services to the EJB that are currently present in the container.
- The EJBs are typically one part of a three tier architecture as shown below.

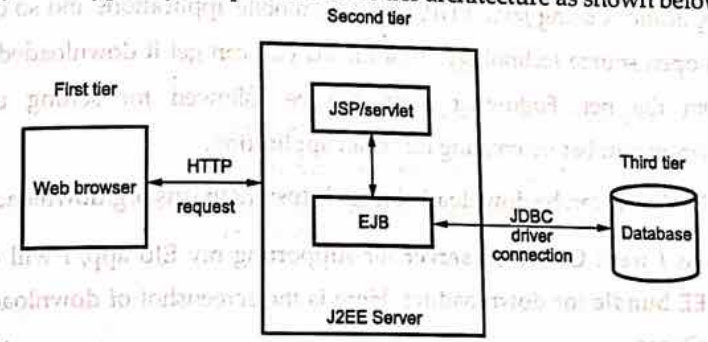


Fig. 6.17.3 3 Tier architecture

- **The following are the flows of the EJB architecture.**
  1. The client is working on a web browser.
  2. There is a database server that hosts a database, like MySQL / Oracle.
  3. The J2EE server machine is running on an application server.
  4. The client interface is provided with JSP / Servlet. The enterprise beans reside in the business tier providing to the client tier.
  5. The Application Server manages the relationships between the client and database machines.

6.17.4 EJB Technology

- Every EJB has three classes :
  1. A home interface which acts as a factory of remote objects.
  2. A remote object which is used for client interaction.
  3. A bean object which contains all the business logic.

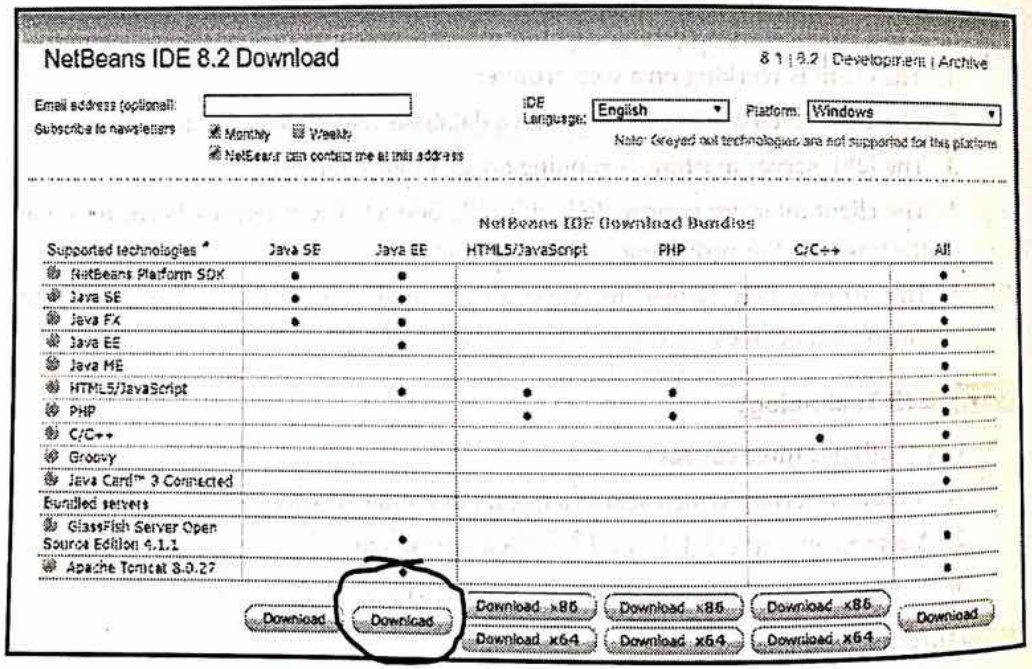
Setting Up the Environment

- In this section we will discuss how to develop an EJB stateless session bean.
- For that purpose we need to set up the environment for execution of EJB stateless bean.
- We need following software to be installed in our machine -
  - JDK
  - NetBeans or Eclipse
  - GlassFish

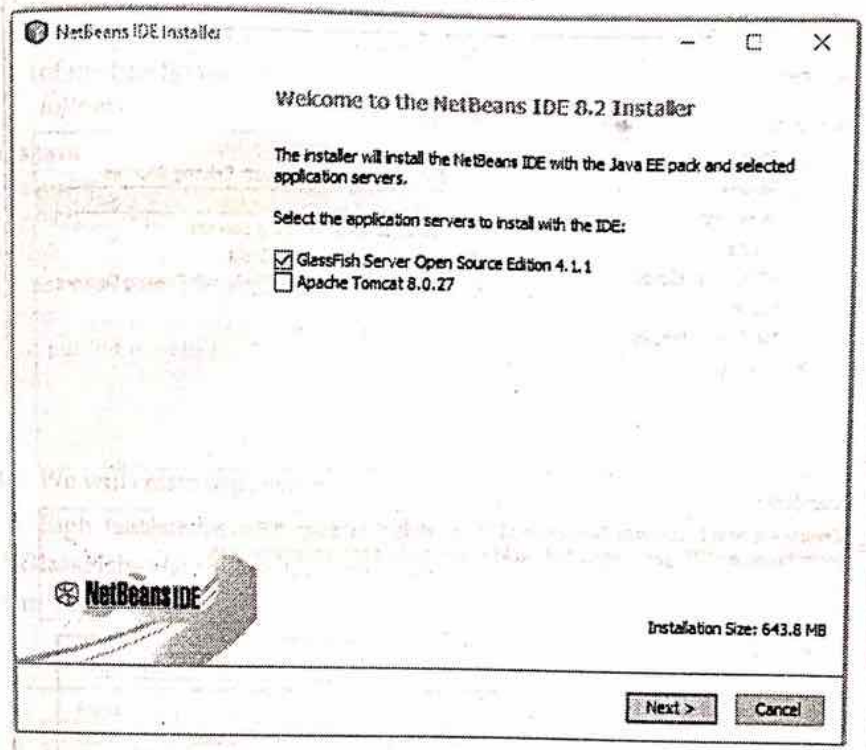
- I have used NetBeans. The NetBeans is basically an Integrated Development Environment (IDE). It is useful for developing various web applications using Java, PHP, C++, EJB, mobile applications and so on. It is an open source technology. That means you can get it downloaded freely from the net. Following steps can be followed for setting up the environment before creating our bean application.

Step 1 : The NetBeans can be downloaded from https://netbeans.org/downloads/

Now as I want **GlassFish server** for supporting my EJB app, I will choose **Java EE bundle** for downloading. Here is the screenshot of download page of NetBeans -



Step 2 : The exe file will start downloading on your machine. After getting the exe file, click on it and start running the installer.



As shown in above screen-shot, Just check for GlassFish Server option. The GlassFish is an Open Source server project started by Sun-Microsystems. It supports the implementation of Enterprise Java Beans, JSF, RMI, JSP and Servlets.

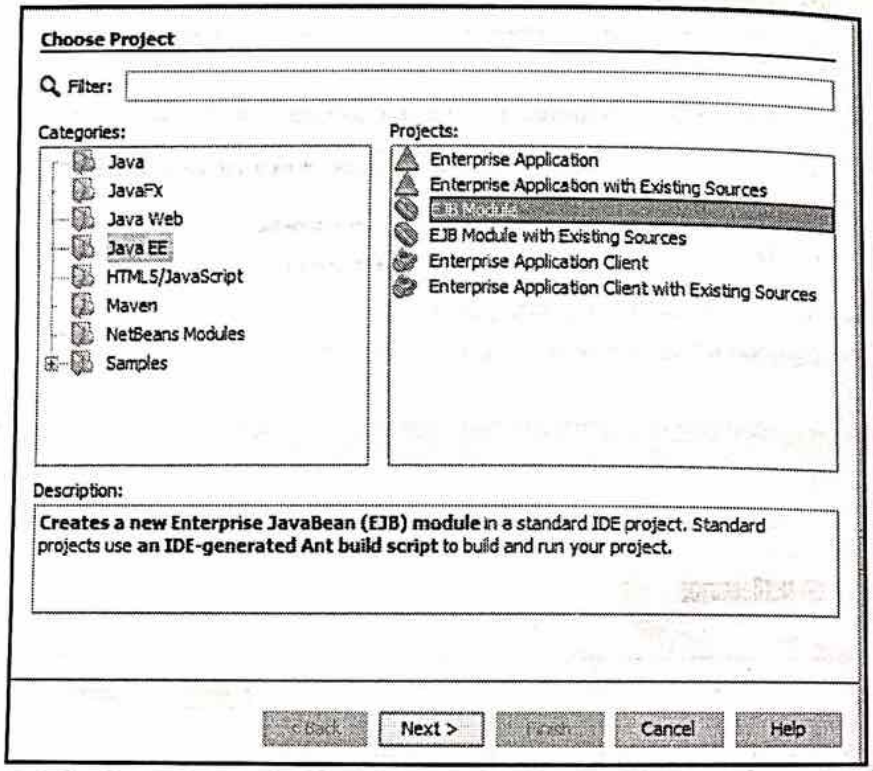
Step 3 : Just go on installing the software by clicking Next button. Just uncheck for Updates if you do not want any updates for NetBeans. Finally uncheck for "Contribute to the NetBeans project by providing ... " and click Finish button. Here completes the installation of your NetBeans software.

Creating a Simple EJB Stateless session Bean

We will follow the following steps for creation of this stateless bean -

Open NetBeans IDE

Step 1 : Select **File->New Project->Java EE-> EJB Module** and Click **Next Button.**



- Give the Suitable Name to the project. Here I am giving "MyAddApp". Click Next Button.
- Select the server as GlassFish.
- Select the J2EE version what ever is installed on your machine and then click Finish button.

Step 2 : Create a new Java Package called myPkg (I have given this name to my package. You can give any suitable name) under the Source Package Folder.

Then create an interface called "MyInterface" under myPkg package and add following code.

MyInterface.Java

```
packagemyPkg;
import javax.ejb.Remote;
@Remote
public interface MyInterface {
```



```
public int add(int a,int b);
}
```

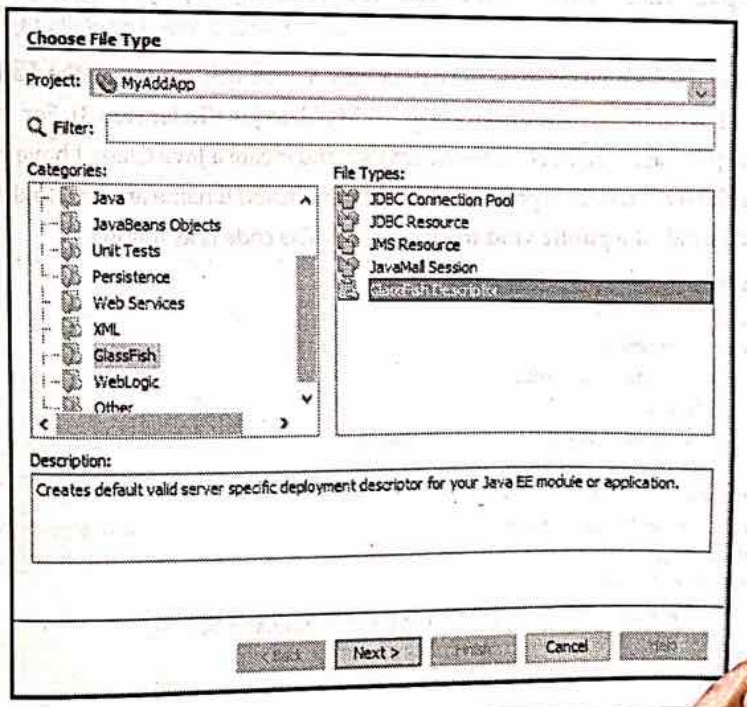
Step 3 : Right click on myPkg, and add the Java class for implementation of above interface. I have named this class as **MyClass.** This class contains the definition of the function **add** which we have declared in above interface. The code is as follows -

MyClass.Java

```
package myPkg;
import javax.ejb.Stateless;
@Stateless
public class MyClass implements MyInterface{
@Override
public int add(int a,int b) {
return a+b;
}
}
```

Step 4 : We will create deployment descriptor file i.e the GlassFish Descriptor file.

Righ Click on the Project "MyAddApp" and then choose **New -->Other-->GlassFish-->GlassFish Descriptor** and then click the "Next" button and click the "Finish " Button.



The **glassfish-ejb-jar.xml** file will be created under the Configuration Files Folder.

Set the JNDI Name for the **MyClass** Bean by editing the file or by clicking the TAB "EJB" and giving jndi name as "aaa". The edited xml file is as shown below -

Glassfish-ejb-jar.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-ejb-jar PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1
EJB 3.1//EN" "http://glassfish.org/dtds/glassfish-ejb-jar_3_1-1.dtd">
<glassfish-ejb-jar>
 <enterprise-beans>
  <ejb>
   <ejb-name>MyClass</ejb-name>
   <jndi-name>aaa</jndi-name>
  </ejb>
 </enterprise-beans>
</glassfish-ejb-jar>
```

Step 5 : Right click on Project node(MyAddApp) and select **Clean and Build.**

After clean and Build again right click and click on project node and select **deploy**

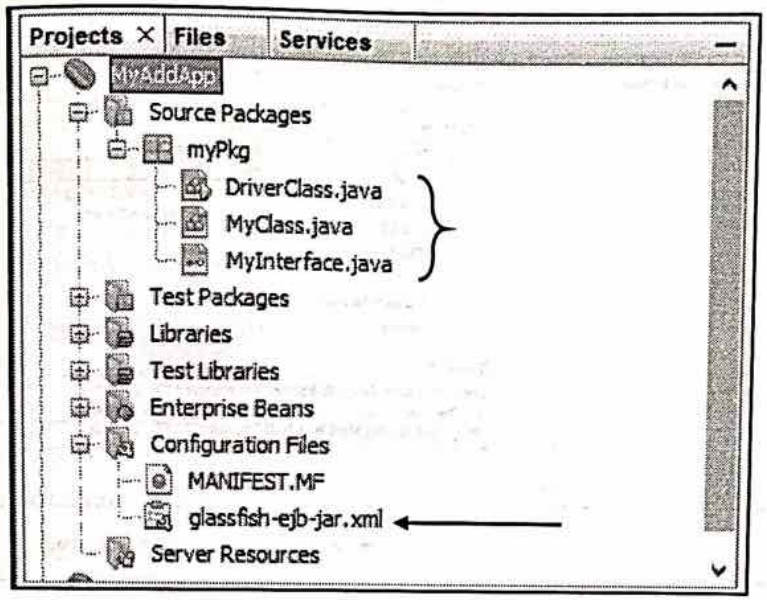
For **both** these actions **"BUILD SUCCESSFUL"** message will be displayed in the console output. Thus the JNDI name "aaa" is successfully deployed with our EJB bean **MyClass.**

Step 6 : Now it's the time to create the driver class which will call the method **add** which we have defined in our bean program **MyClass.java**(Refer step 3). For that purpose, again right click on the **myPkg** and create a Java Class. I have named it as **DriverClass** . This program will contain the JNDI name and call to the method **add** inside the **public void main** function. The code is as follows -

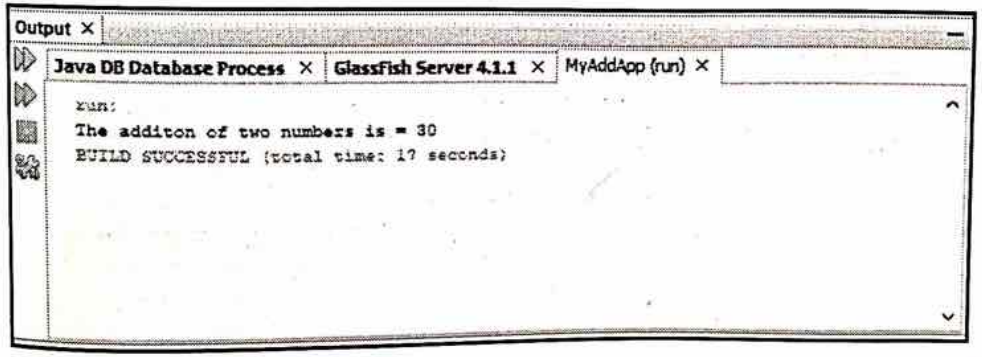
DriverClass.java

```
package myPkg;
import java.util.Properties;
import javax.naming.InitialContext;
public class DriverClass {
  public static void main(String arg[])throws Exception
{
Properties p=new Properties();
InitialContext c=new InitialContext();
Object obj=c.lookup("aaa");
MyInterface inter=(MyInterface)obj;
System.out.println("The additon of two numbers is = "+inter.add(10,20));
}
}
```


To let you know, the locations of these files I am giving the screen-shot of Project explorer window for **MyAddApp** project.



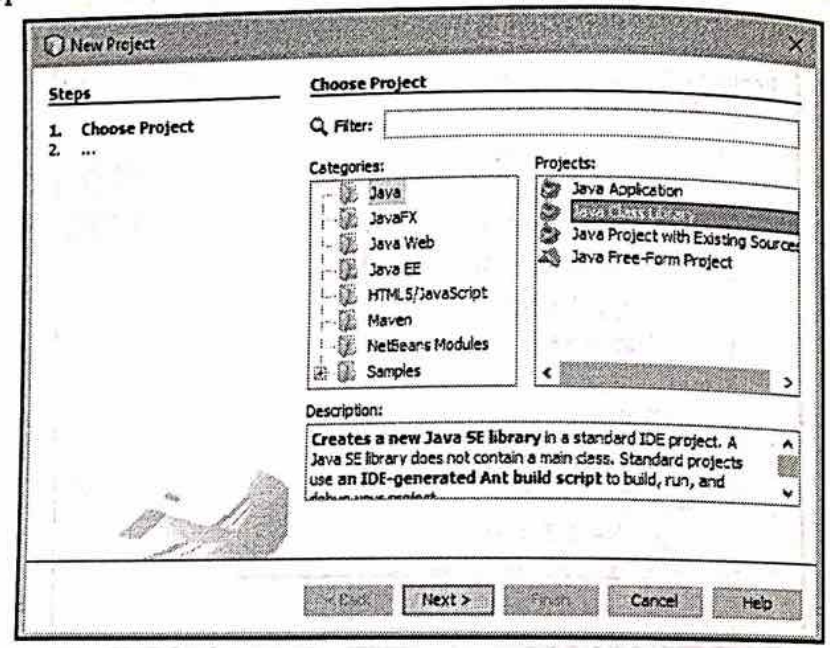
Step 7 : Right click on **DriverClass.java** click on **Run File** option and you will get the output on console window. Here is the output screenshot.



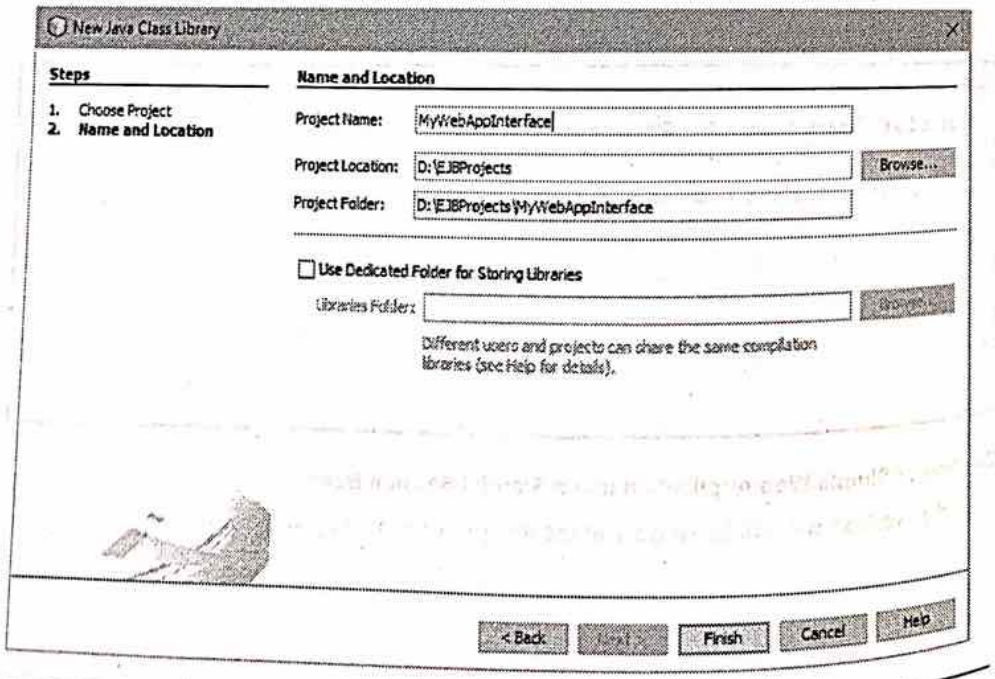
Creation of Simple Web Application using Stateful Session Bean

In this section we will learn how to create application using stateful session bean.

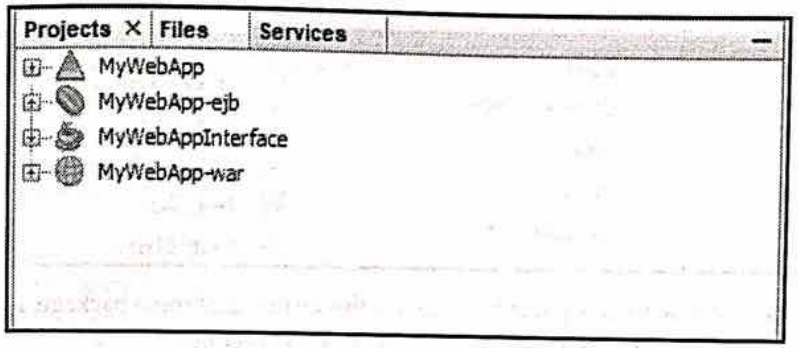
Step 1 : Click **File-> New Project -> Java-> Java Class Library.**



Click Next button. Add name to the Project. I have given the name as **MyWebAppInterface** and click **Finish** button.



Step 2 : Now click **File->New Project->Select ->Java EE->Enterprise Application.** Then click Next button. Give suitable name to your project. I have given the name **MyWebApp**. Then click Next button. Then just click **Finish** Button. Your Project explorer Window will look like this -

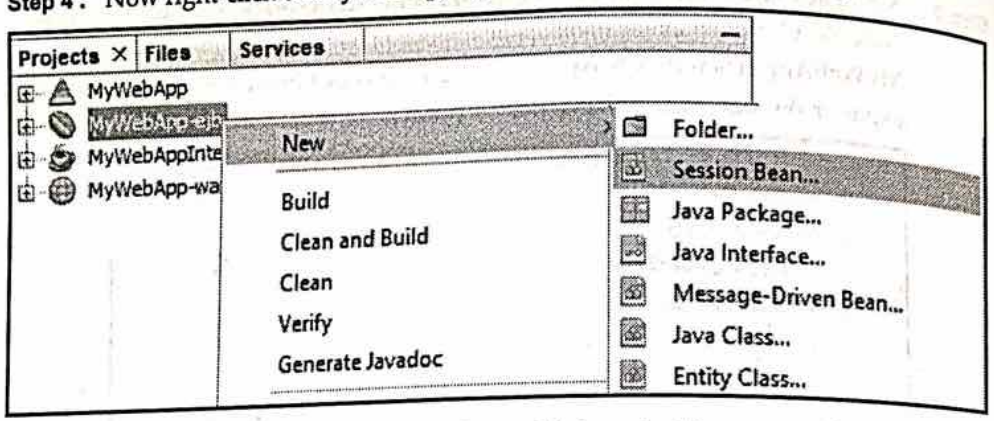


Step 3 : Now click on **MyWebApp**-War node, then click on **Web-Pages** folder, locate the **index.html** file. Double click this file and edit it as follows -

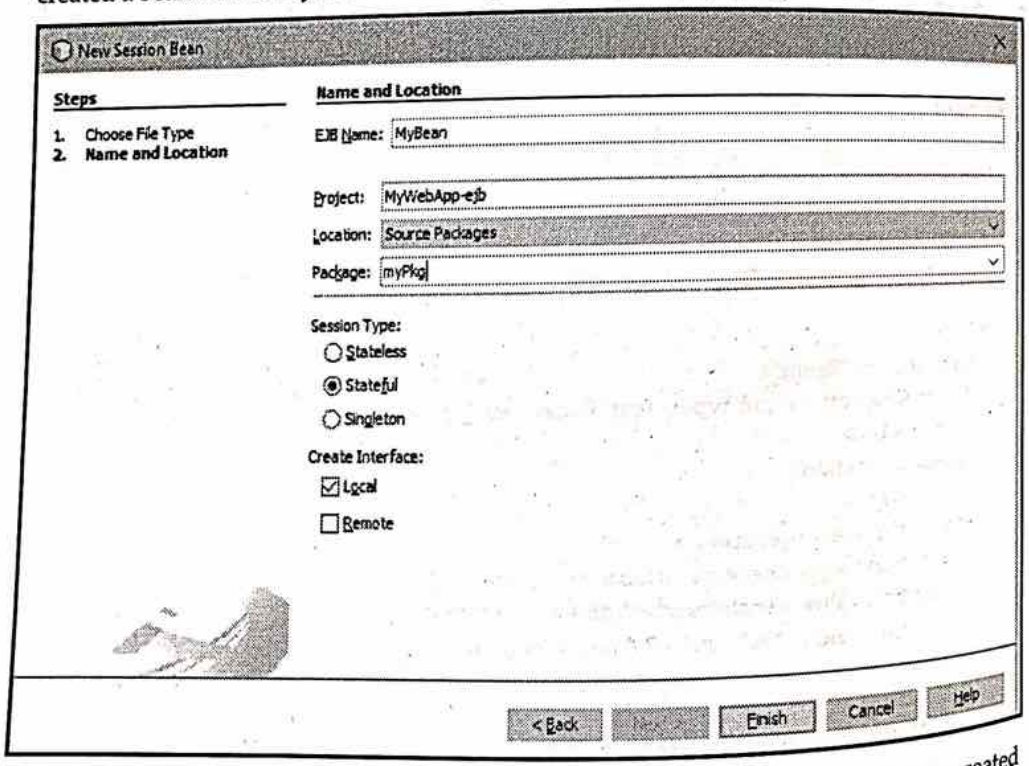
index.html

```
<html>
  <head>
    <title>Web Application For Bank Transactions</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
  </head>
  <body>
    <form action="Main">
      Enter Amount: <input type="text" name="amt_txt">
      <br/><br/>
      Select Operation:
      <br/><br/>
      <select name="operation">
        <option value="deposit">Deposit</option>
        <option value="withdraw">Withdraw</option>
        <option value="balance">Balance</option>
      </select>
      <br/><br/>
      <input type="Submit" value="Submit">
    </form>
  </body>
</html>
```

Step 4 : Now right-click on **MyWebApp-ejb** node, click New->Session bean



Now give suitable name to your bean. Create this bean inside some package. I have created a bean named **MyBean** inside the package named **mypkg.**



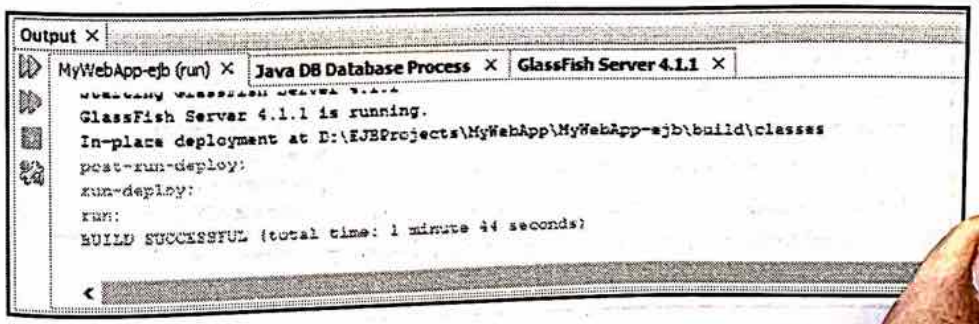
Since this is a **stateful EJB application,** Select the option **Stateful**. Here I have created a Local interface. For that purpose, **check** for **Local** interface option. Finally click **Finish** button. Edit the bean file as follows.

While inserting the code we can right click inside the class **MyBean** and add business methods as -

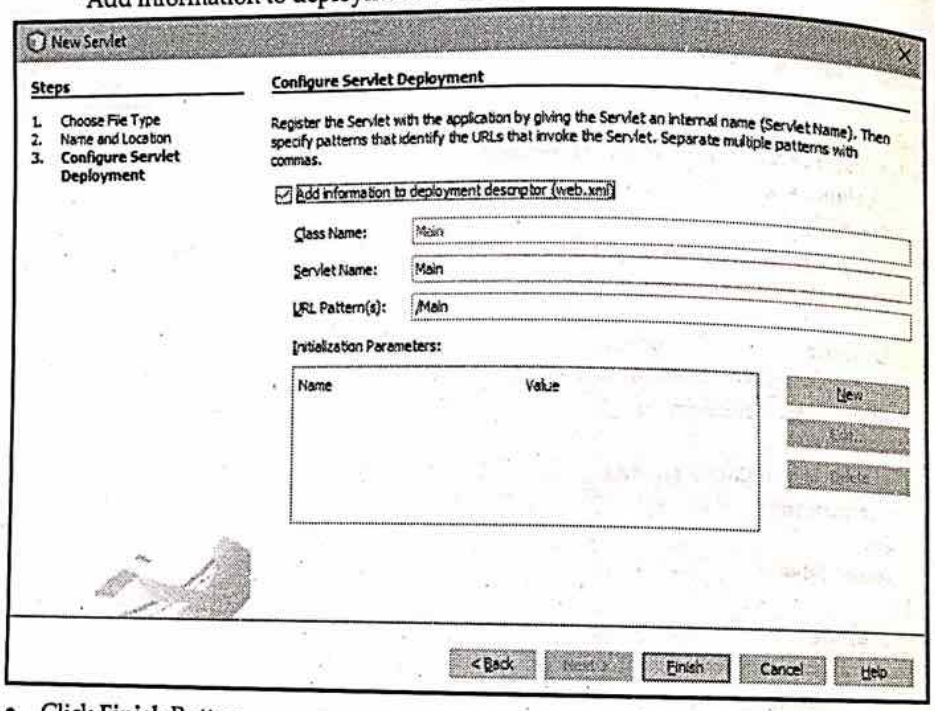
MyBean.Java

```
package myPkg;
import javax.ejb.Stateful;
@Stateful
public class MyBean implements MyBeanLocal {
  int balance=0;
  @Override
  public void deposit(int amount) {
    balance=balance+amount;
  }
  @Override
  public boolean withdraw(int amount) {
    if(amount<=balance)
    {
      balance=balance-amount;
      return true;
    }
    return false;
  }
  @Override
  public int getBalance() {
    return balance;
  }
}
```

Step 5 : Right click **MyWebApp-ejb** node and click **Run.** The GlassFish Server will start running and you will get **BUILD SUCCESSFUL** message in console window at the bottom of your NetBean IDE.



Step 6 : Now , inside the MyWebApp-War node there is a **Source Packages** , right click it and create Servlet. I have named this as **Main**. Click the next button and Check "Add information to deployment..." as shown -



- Click **Finish** Button.

- The servlet code window will appear with default code. Locate the method void ProcessRequest(...) inside this default code. Just above this method, right click and select Insert Code and select Call Enterprise Bean and then select MyWebApp-ejb node and select MyBean inside it. Click OK button. Due to this some built in code for Local Interface reference will appear inside your servlet code. This code will be for the lookup method.
- Inside this code you have to locate the instance for this Local lookup. This instance is generally located at the beginning of servlet definition as follows.

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import myPkg.MyBeanLocal;

/**
 *
 * @author Puntambekar
 */
public class Main extends HttpServlet {

    MyBeanLocal myBean = lookupMyBeanLocal();
```

Now using the instance **myBean** we can invoke the methods defined in our Java Bean. Note that this Java bean we have created in step 4.

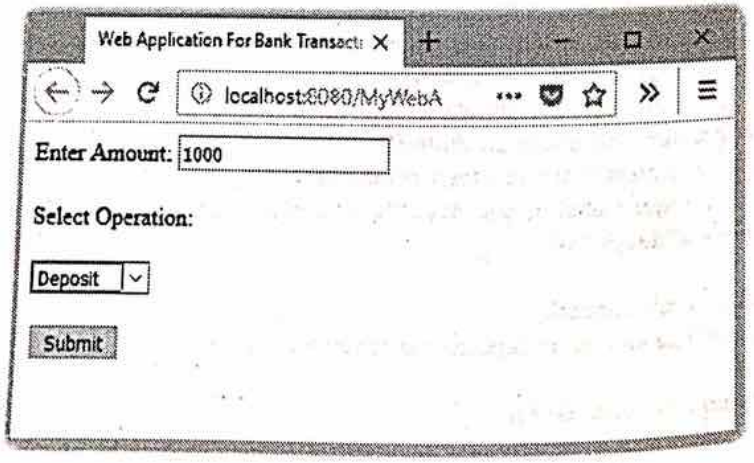
The edited code for servlet is as follows -

Main.Java

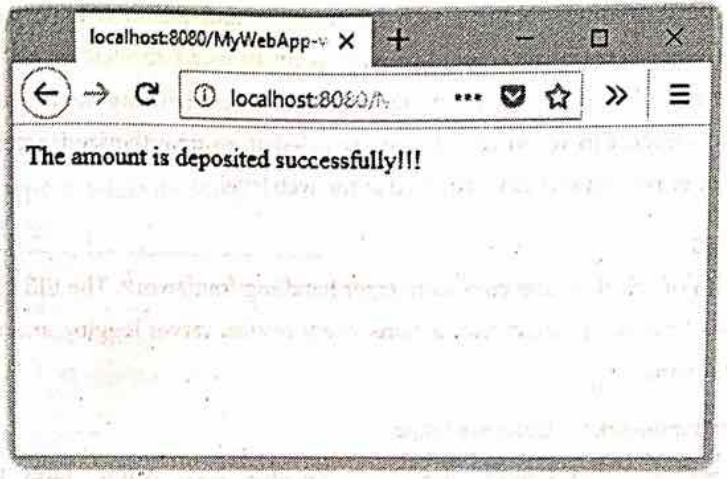
```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import myPkg.MyBeanLocal;
public class Main extends HttpServlet {
    MyBeanLocal myBean = lookupMyBeanLocal();
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      try (PrintWriter out = response.getWriter()) {
       String choice=request.getParameter("operation");
       int amount=Integer.parseInt(request.getParameter("amt_txt"));
       if(choice.equals("deposit"))
       {
         myBean.deposit(amount);
         out.println("The amount is deposited successfully!!!");
       }
         if(choice.equals("withdraw"))
       {
```

```
        if(myBean.withdraw(amount))
        {
          out.println("The amount is wihdrawn successfully!!!");
        }
        Else
          out.println("Sorry, Balance insufficient!!!");
      }
      if(choice.equals("balance"))
      {
        out.println("The total balance is = "+myBean.getBalance());
      }
    }
  }
...
....
  private MyBeanLocal lookupMyBeanLocal() {
    try {
      Context c = new InitialContext();
      return (MyBeanLocal) c.lookup("java:global/MyWebApp/MyWebApp-
ejb/MyBean!myPkg.MyBeanLocal");
    } catch (NamingException ne) {
      Logger.getLogger(getClass().getName()).log(Level.SEVERE, "exception caught", ne);
      throw new RuntimeException(ne);
    }
  }

}
```

Step 7 : Now right click the **MyWebApp-ejb** and deploy it. Again click on the node **MyWebApp** and Run. The browser window will execute the application as follows -



Select the operations as perform the transactions.



6.17.5 EJB Container and its Services

- The EJB container provides following services to EJB component -

Persistence

- The EJB container provides the persistence service. That means, EJB can save its state in web container. The persistence services include simple connection pooling, reading some data from databases to generate dynamic pages or update some simple information. Writing data access code is error prone and time consuming. Hence the enterprise beans are created by application development tools in such a way that the data access logic will be encapsulated within the components. Due to this service the real-world project can be benefited in terms of time, budget and personal constraints.

Declarative Transactions

- Transaction management is complex if multiple data access components or multiple data sources are involved. The EJB provides the transaction management service in such a way that the complex transaction can be managed without any coding.
- The Java API - **Java Transaction API** (JTA) provides standard interface between transaction manager, application server and component. There is another Java API **Java Transaction Service** (JTS) which is directly used by the application programmer.

Declarative Security

- It is most essential to have security for data and business logic functionality. Usually application developers provide security using servlets or JSP pages. But this is complex and error prone task. The EJB uses the simplified security model due to which EJB resources can be protected from unauthorized access. The EJB developer rely on security provided at the web level.

Error Handling

- The EJB offers clear and consistent error handling framework. The EJB specification defines how errors affect transactions, client results, server logging and recovery of components.

Component Framework for Business Logic

- The software that requires a complex business logic require large number of enterprise resources. EJB are server side components that can be used simultaneously by different clients. Using the EJB's common framework for the business logic the building of complex application is possible.

Scalability

- Using EJB specification rules it is possible for the application server to manage large number of simultaneous clients. EJB specification is useful to satisfy the multiple demands for business logic components and data access components.

Portability

- EJB's are written to an industry standard API. Hence they can be run on any new J2EE compliant application server. There is no need to modify the EJB components for running them on application servers.

Manageability

- EJB container provides the facility to manage access, add or remove users from appropriate role. Using EJB it is possible to manage secure access to every user interface that provides the functionality of accessing the users.

6.17.6 EJB Interfaces

- In EJB, the client can access the session bean using the business interface. Depending upon the location of client, there are two types of interfaces used -

1) **Remote Interface :** The remote interface is used by client that may be running on different machine than the bean. Using Remote Method Interface (RMI) the access to the bean can be made.
2) **Local Interface :** The local interface can be used by the client which is running on the same machine where the bean is residing.

Various aspects while deciding between local and remote interface are represented in the following table -

Aspect	Use of Remote Interface	Use of Local Interface
Type of client	If the clients are application clients, web components or EJBs executing on different machine.	If the clients are either web components or EJBs executing on the same machine.
Protocol used	If the RMI used for communication.	If the local function calls are used for communication.
Component distribution	If the component in the application are distributed, that means present on different machines.	If the components are located on the same machine where bean is present.
Loose or tight coupling	If the client and bean are coupled loosely.	If there is a requirement of tight coupling between EJB components.
Performance	Slow performance due to RMI calls.	Better performance due to local calls.
Location transparency	If the client need not know the location of EJB and access it transparently.	If the client wants to know that the current machine is the location of EJB.
Communication	If multiple objects are passed by value so that client and bean can operate on different copies of the object.	If objects are passed by reference and the client or bean can change the value of the same object.
Serialization	If the serialized objects are passed.	If the objects are not serialized.

6.17.7 EJB Clients

- EJB is the server-side and platform-independent Java Application Programming Interface (API) for Java Platform, Enterprise Edition (Java EE). EJB is used to simplify the development of large distributed applications. EJB client is a code which will use the EJB functionality. Clients cannot create objects of EJB class component. Clients can access any EJB only through the EJB container. Some of the EJB clients are as follows -

1) Java Applications
2) Other EJBs on same or another application server.
3) Java Applets
4) A web service can access EJB
5) Servlets, JSP- some server side technologies.

Review Questions

1. *Decide and explain various aspects while deciding between local and remote interface can be considered.* **SPPU : May-18, Marks 8**
2. *What is enterprise Java Bean ? Draw and explain main components of EJB architecture.* **SPPU : May-18, Marks 8**
3. *Draw and explain the role of EJB container in enterprise applications.* **SPPU : May-18, Marks 6**
4. *Draw and explain scenario of client accessing remote EJB. List some of the EJB clients.* **SPPU : Dec.-18, May-19, Marks 8**
5. *Draw and explain each tier of three tiers architecture using EJB.* **SPPU : May-19, Marks 8**

6.18 Multiple Choice Questions

Q.1 Ruby is a ___.
- a scripting language
- b server side programming language
- c object oriented programming language
- d procedural programming language

Q.2 ___ method in ruby removes the last character.
- a Pop
- b Remove
- c Del
- d Chop

Q.3 ___ method in ruby is used to insert the element from the end.
- a Insert
- b Push
- c Put
- d None of these

Q.4 What is the extension used for saving the ruby file ?
- a .ruby
- b .rb
- c .ry
- d None of these

Q.5 Which of the following is valid conditional statement in ruby ?
- a elseif
- b elsif
- c else if
- d elseiff

Q.6 What will any variable evaluate to if it is of Boolean data type ?
- a True
- b NULL
- c False
- d Either True or False

Q.7 Which services are provided to EJB components by the EJB container ?
- a Transaction support
- b Persistence support
- c Naming support
- d All mentioned above

Q.8 Session beans don't have _____.
- a ejbCreate() method
- b ejbStore() method
- c ejbRemove() method
- d none of these

Q.9 An entity bean's local interface MUST extend the _____ interface.
- a javax.ejb.EJBLocalObject
- b javax.ejb.EJBObject
- c javax.ejb.RemoteObject
- d none of the above

Q.10 What executes EJB components ?
- a A web server
- b An application server
- c An EJB container
- d A database server

Q.11 What represents a persistent global data from the database ?
- a Entity bean
- b Session bean
- c Both A & B
- d None of the above

Answer Keys for Multiple Choice Questions :

Q.1	c	Q.2	d	Q.3	b	Q.4	b
Q.5	b	Q.6	d	Q.7	d	Q.8	a
Q.9	a	Q.10	c	Q.11	a		

SOLVED MODEL QUESTION PAPER (End Sem)

Web Technology

T.E. (Computer Engineering) Semester - VI (As Per 2019 Pattern)

Time : 2 $\frac{1}{2}$ Hours] **[Total Marks : 70**

N. B. :

i) Attempt Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.

ii) Neat diagrams must be drawn wherever necessary.

ii) Figures to the right side indicate full marks.

iv) Assume suitable data, if necessary.

Q.1 a) *What is the difference between server side scripting language and client side scripting language.* **(Refer section 3.1)** **[5]**

b) *Write a Java servlet program which will display "welcome servlet" message.* **(Refer section 3.3)** **[8]**

c) *Write HTML form to read user name and password. This data is sent to the servlet. If the correct user name and password is given then welcome him/her by his/her name otherwise display the message for invalid user.* **(Refer example 3.7.2)** **[5]**

OR

Q.2 a) *Enlist the steps for connectivity between servlet and database.* **(Refer section 3.14.3)** **[5]**

b) *What are the strengths of XML technology ? Also list the limitations of XML.* **(Refer section 3.15)** **[5]**

c) *Write AJAX script to obtain the student information stored in XML document. The information should be displayed on clicking the button. It should be displayed in tabular form.* **(Refer example 3.26.1)** **[8]**

Q.3 a) *Write advantages of JSP over servlets. Also explain the life cycle of JSP.* **(Refer section 4.2)** **[5]**

b) *Explain the scripting elements in JSP.* **(Refer section 4.3.6)** **[5]**

c) *What is MVC ? Draw and explain MVC architecture for developing web applications.* **(Refer section 4.6)** **[7]**

OR

Q.4 a) *Write a short note on SOAP.* **(Refer section 4.10)** **[5]**

b) *Draw and explain neat diagram which depicts MVC to struts architecture.* **(Refer section 4.12)** **[5]**

c) *Write a client server JSP program to find the simple interest and display the result in the client.* **(Refer example 4.5.1)** **[7]**

Q.5 a) *Classify data types of PHP and describe various data types.* **(Refer section 5.3)** **[4]**

b) *Explain how session and cookies are used for session management in PHP.* **(Refer sections 5.14 and 5.15)** **[8]**

c) *What is associative arrays in PHP ? Explain it with the help of simple PHP code.* **(Refer section 5.8)** **[6]**

OR

Q.6 a) *Identify and explain steps involved in connecting to MySQL with PHP.* **(Refer section 5.16)** **[8]**

b) *How does ASP.NET works ? Also enlist the features of ASP.NET.* **(Refer section 5.19)** **[5]**

c) *Explain the life cycle of ASP.NET application.* **(Refer section 5.22.1)** **[5]**

Q.7 a) *Explain the scalar types and their operations in ruby.* **(Refer section 6.2)** **[6]**

b) *Write a program in rails for displaying the current date and time.* **(Refer section 6.12.2)** **[6]**

c) *Write a ruby program to add two numbers.* **(Refer section 6.3.2)** **[5]**

OR

Q.8 a) *Write a ruby program that sorts the array and searches the desired element from array.* **(Refer section 6.5.3)** **[8]**

b) *Explain how to write a student class in ruby program.* **(Refer section 6.8.1)** **[9]**

□□□