- **Understanding the need of web application:-**

In standalone app and desktop app our logic and data available to single computer/layer.

Ex: - 1) java class contains main method (standalone app).

2) AWT (abstract window tool) frame app (Desktop app).

In two-tier applications, the data and logic of server app can be accessible to multiple client of the internet network. (LAN). To provide global visibility of data and logic we need to create a web site because website resides in internet environment. A Web application, which is placed in internet having domain name, is called website.

Website gives 24/7 accessibility of data and logic because it resides and execute in internet environment. To develop website in java we will used jsp(java server pages)and servlets. To develop website in .net we will used asp.net(active server pages) and asp.net mvc(model view control).We can develop website using php also(personal homepage for hypertext preprocessor).To develop Large scale web application we should used java. To develop medium scale web application we should used .net. To develop small-scale web application we should used php.

- **WEB Application:-**

It is collection of web resource program having capability to generate web pages. Internet looks like a spider web that why internet network is also known as web.

- **Types of web pages in web application:-**

There are two types of web pages they are

1. Static web page/passive web page:-

It shows the same content for multiple requests.

Ex:-contact us, about us, services, contant us and etc.

2. Dynamic web pages/active web pages:-

It is varied from request to request based on time.

Ex:-stock market share values, gmail inbox, live cricket score etc.

- **Types of web resource program:-**

There are two types of web resource program they are

1. Static web resource program:-

It is used to develop static web pages.

Ex:-html programs and java script program.

2. Dynamic web resource program:-

It is used to develop dynamic web pages.

Ex:-jsp programs and servelets programs.

- **Understanding the setup of web application and web resource: -**

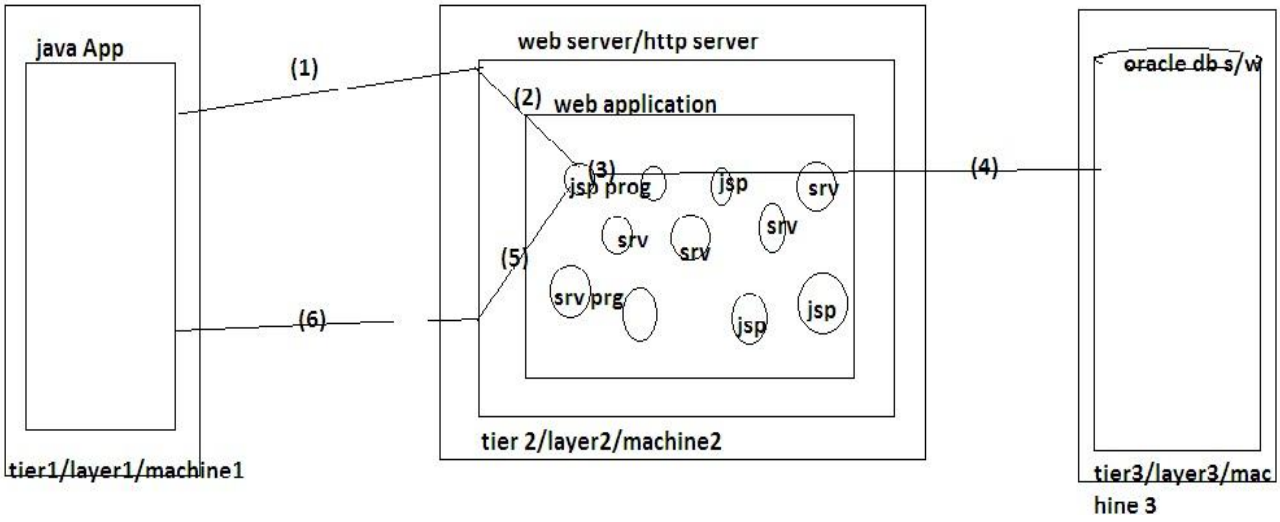


Figure 1

The java application will execute manually.web application and web resource program will execute automatically at the time they have requested.

Web server is piece of software, which is used to automate web resource and web application execution. So they is no chance of executing web resource program manually.

- **With respect to the diagram:-**

1. First java application will gives request to web resource program.
2. Web server traps that request and sends request to specific web resource program.
3. Web resource program execute the logic to process the request
4. Web resource program interact with DB s/w if necessary.
5. The output of web resource program gives to web server.
6. Web server sends output to browser window as dynamic web page.

The process of keeping web application in web server is called deployment and reverse is called undeployment. We can develop web app as 2-tier app without db s/w. we can develop web app as 3-tier app with db s/w. If we develop web app as 2-tier app then it will looks like thin client -fat server application.

Based on the execution of web resource program we can say they are two types of web resource programs.

1. Server side web resource program:-

The program, which executes at server side when requested is called server side web resource program.

Ex: jsp programs, servlets programs etc.

By default every dynamic web resource programs are server side web resource program.

2. client side web resource program:-

The program, which executes at browser window when it is requested is called client side web resource program.

Ex:-html programs, java script programs etc.

By default every static web resource programs are client side web resource program.

Note:-To know whether it is server side/client side web resource program we need to check where it will be executes and but not where it is resides.

- **Web server:-**

It is a piece of software, which is used to manage and execute web application. It is used to automate all the process of web application and web resource program execution.

Ex:-Tomcat, resin etc.

- **Web server Responsibilities:-**

It takes client request continuously. It takes request and sends that request to appropriate web resource program. It provides environment to execute all web resource program. It allows to execute client side web resource program in browser window. It provides environment to deploy and undeploy the application. Apply middle ware services (additional services) by applying to the application.

Ex: - security,jdbc connection pooling and etc.

It takes the result from web resource program and generate web page. It automates all the process of web application and web resource program execution. To execute java application we will use jre/jvm. To execute Applets we will used Applet viewer/Applet container. To execute servlet app we will used Servlet container.simillarly to execute jsp app we will used jsp container.

- **Container:-**

It is a software application/program, which is used to manage whole life cycle of web resource program. (From birth to death). Servlet Container manage whole life cycle of servlet program. Jsp container manage whole life cycle of jsp program.

Some part of industry considers servlet container and jsp containers are web containers. Every web server support servelt container and jsp container so we do not need to arrange it separately.

- **Architecture of java based web server:-**

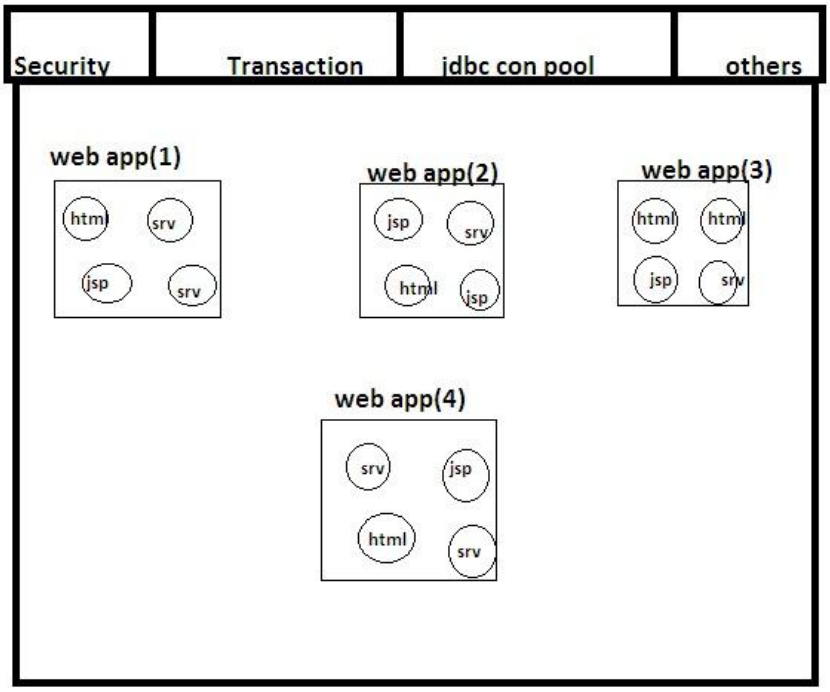


Figure 2

Middle ware service will added to deployed application only.When servlet program requested it goes to servlet container for execution. When jsp program requested it goes to jsp container for execution. To execute jsp program jsp container take the supports of servlet container and it will generate servlet equivalent jsp program. When html gets requested it goes to browser window for execution.

- **What is difference b/w web server and web container?**

Web server	Web container
It is responsible to take the request and gives response.	It is responsible to execute all web resource program
It is also responsible to manage and execute web application.	It is responsible to manage whole life cycle of web resource program

Note:-By default, every web server provides built-in containers.

Ex:-Servelt container and jsp container.

- **Tomcat:-**

Type	:	web server
Version	:	7.x
Default port no	:	8080
Creator	:	David dunconson
Company	:	from apache foundation
To download	:	www.apache.org.
For doc	:	www.doc.apache.org.
Source	:	open source
Container	:	ServletContainer and JSPContainer
ServeltConatiner	:	Catalina
JSPContainer	:	jasper

Tomcat is not a container it is a web server contains servlet container and jsp container. Before 6.x tomcat is called web server but from 6.x onwards it is called Application server. Tomcat installation will give following things.

1. JRE location (parallel to jdk).
2. Tomcat installation location.
3. Port no
4. Administrative username and password.

- **Servlets:-**

Servlet is a java based web resource program that enhances the functionality of web server/application server/http server. Servlet is a java based web resource program to generate dynamic web resource program having capability to generate dynamic web pages. Servlet is a "single instance multi thread" java based web resource program which is used to create web applications.

Ex:-If 100 requests are coming to servelt program then serveltcontainer will create only one object and create multiple threads representing the multiple requests.

- **Servlet api's:-**

  javax.servlet.*;

  javax.servlet.http.*;

- **Important resources of servlet:-**

```
   javax.servlet.Servlet
  |       |
  |       |
          | implements
          |
  javax.servlet.GenericServlet
            (abstract class)
          |
          |
          | extends
          |
  javax.servlet.http.HttpServlet
```

- **First java web application development having servlet program as web resource program:-**

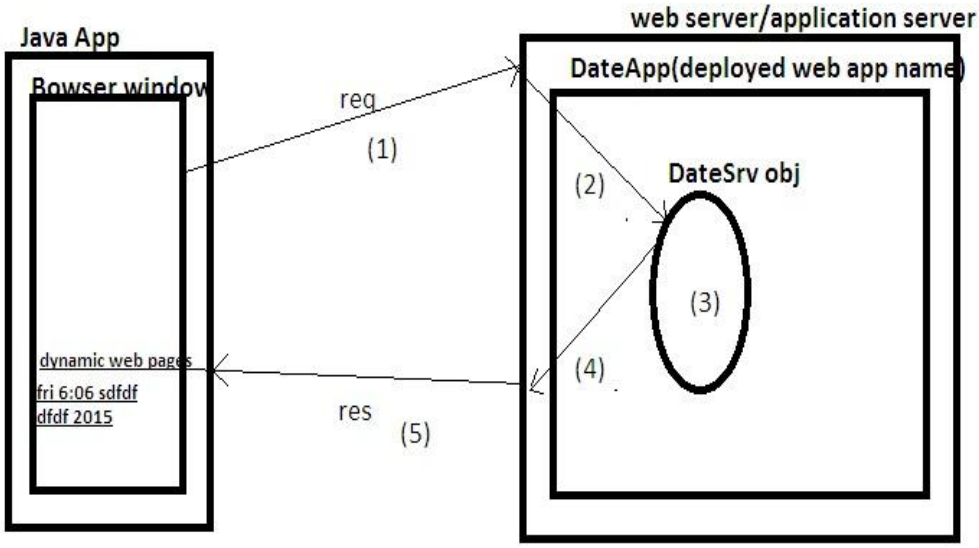
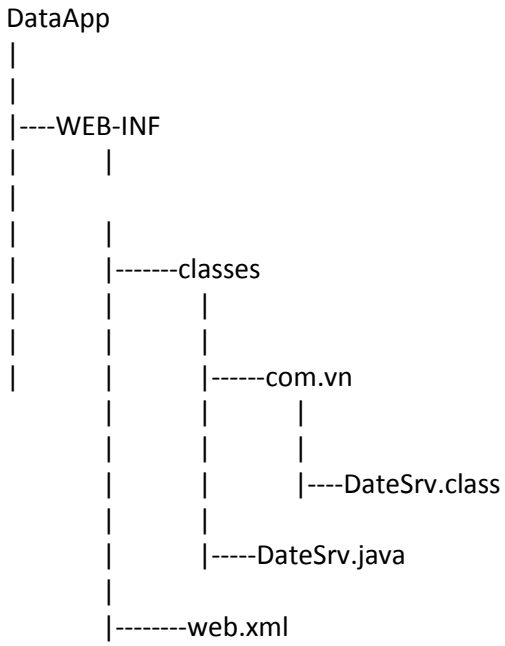


Figure 3

Step1:

Create deployment directory structure for web application.



Step2:

Create DateSrv.java program and placed in WEB-INF/classes folder.

```
Package com.vn;
import javax.servlet.*;
import java.io.*;
import java.util.*;
class DateSrv extends GenericSevlet
{
public void service(ServletRequest req,ServletResponse res)
throws ServeltException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("text/html");
```

```
Date d=new Date();
System.out.println("<b><center>"+d+"</b></center>");
pw.close();
}
}
```

Step3:

Add servlet-api.jar in CLASSPATH environmental variables.

D:\Tomcat 6.0\lib\servlet-api.jar

Step4:

Configure web.xml file

```
<web-app>
 <servlet>
 <servlet-name>abc</servlet-name>
 <servlet-class>com.vn.DateSrv</servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name>abc</servlet-name>
 <url-pattern>/test</url-pattern>
 </servlet-mapping>
 </web-app>
```

Step 5:

Compile DateSrv prog with javac -d . DateSrv.java.

Step6:

Copy DateApp web application and deployed in Tomcat/web apps folder.

Step7:

Start tomcat server from Tomcat/bin folder.

Step8:

Check the web application. (open browser window and type given url)

http://localhost:2525/DateApp/test

Understanding the url pattern:-

Every Servlet program will recognized by its url pattern. Client, other web resource program and serveltcontainer will recognize our servlet program by url pattern only. This url pattern is used to hide the servlet class name and technology which is used to develop web application from outsider.

According to servlet specification there are three types of url pattern.Every servlet container server is design to support these three types of url patterns.

1. Exact Match Url pattern
2. Directory Match Url pattern
3. Extension Match Url pattern.

1. Exact Match Url pattern:-

It should start with "/" symbol and it should not end with "*" symbol.

- In web.xml:

  <url-pattern>/test</url-pattern>

- Request url:

  http://localhost:2525/DateApp/test (valid)

  http://localhost:2525/DateApp/turl (invalid)

  http://localhost:2525/DateApp/demo (invalid)

2. **Directory Match url pattern**:-

It should start with "/" symbol and it should end with "*" symbol only.

- In web.xml:

  <url-pattern>/x/test/*</url-pattern>

- Request url

  http://localhost:2525/DateApp/x/test/demo (valid)

  http://localhost:2525/DateApp/x/demo/test (invalid)

  http://localhost:2525/DateApp/x/turl/demo (invalid)

3. **Extension Match Url pattern:-**

It should start with "*" symbol having extension.

- In web.xml:

  <url-pattern>*.do</url-pattern>

- Request url:

  http://localhost:2525/DateApp/first (invalid)

  http://localhost:2525/DateApp/demo/first (invalid)

  http://localhost:2525/DateApp/demofirst.do (valid)

- **Other content Type/MIME Type**:-

1. Text/html:

   It is used to display web page as html format.

2. Text/xml:

   It is used to display web page as xml format.

3. Application/ms-word:

   It is used to display web page as ms-word format.

4. Application/vnd.ms-excel:

   It is used to display web page as ms-excel format.

- **MIME Type:-Multi -purpose internet mail extension:-**

A web application can contain multiple servlet programs. Each servelt program we need to configure in web.xml file with multiple <servlet> ,<servlet-mapping> tags.

- **Develop web application to display output in different format using MIME Types:-**

  - **Develop deployment directory structure**:-

```
MimeApp
|
|
|-----WEB-INF
        |
        |
        |----------classes
        |                 |
        |                 |----HtmlSrv.java
        |                 |----XmlSrv.java
        |                 |----MswSrv.java
        |---web.xml       |----MseSrv.java
                          |
                          |--com.vn
                                  |
                                  |
                                  |---HtmlSrv.class
                                  |---XmlSrv.class
                                  |---MswSrv.class
                                  |---MseSrv.class
```

  - **Program-1**

```
package com.vn;

import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;

class HtmlSrv extends HttpServlet

{

public void service(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException

{

PrintWriter pw=res.getWriter();
```

```
res.setContentType("text/html");
pw.println("<table border=1>");
pw.println("<tr><th>Players</th><th>Hobbie</th></tr>");
pw.println("<tr><td>Sachine</td><td>Batsman</td></tr>");
pw.println("<tr><td>Zaheer</td><td>bowler</td></tr>");
pw.println("</table>");
pw.close();
}
}
```

- **Program-2**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
class MseSrv extends HttpServlet
{
public void service(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("application/vnd.ms-excel");
pw.println("<table border=1>");
pw.println("<tr><th>Players</th><th>Hobbie</th></tr>");
pw.println("<tr><td>Sachine</td><td>Batsman</td></tr>");
pw.println("<tr><td>Zaheer</td><td>bowler</td></tr>");
pw.println("</table>");
pw.close();}}
```

- **Program-3**

```
package com.vn;

import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;

class MswSrv extends HttpServlet

{

public void service(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException

{

PrintWriter pw=res.getWriter();

res.setContentType("application/ms-word");

pw.println("<table border=1>");

pw.println("<tr><th>Players</th><th>Hobbie</th></tr>");

pw.println("<tr><td>Sachine</td><td>Batsman</td></tr>");

pw.println("<tr><td>Zaheer</td><td>bowler</td></tr>");

pw.println("</table>");

pw.close();

}

}
```

- **Program-4**

```
package com.vn;

import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;

class XmlSrv extends HttpServlet

{
```

```
public void service(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("text/xml");
pw.println("<table border=1>");
pw.println("<tr><th>Players</th><th>Hobbie</th></tr>");
pw.println("<tr><td>Sachine</td><td>Batsman</td></tr>");
pw.println("<tr><td>Zaheer</td><td>bowler</td></tr>");
pw.println("</table>");
pw.close();
}
}
```

- **configuring servelt programs in web.xml file:-**

```
<web-app>
<servlet>
<servlet-name>abc</servlet-name>
<servlet-class>com.vn.HtmlSrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>abc</servlet-name>
<url-pattern>/hurl</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>xyz</servlet-name>
<servlet-class>com.vn.XmlSrv</servlet-class>
</servlet>
<servlet-mapping>
```

```
<servlet-name>xyz</servlet-name>
<url-pattern>/xurl</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>pqr</servlet-name>
<servlet-class>com.vn.MswSrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>pqr</servlet-name>
<url-pattern>/mwurl</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>mno</servlet-name>
<servlet-class>com.vn.MseSrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>mno</servlet-name>
<url-pattern>/meurl</url-pattern>
</servlet-mapping>
</web-app>
```

- **Types of communications:-**

We can communicate with servlet in three ways

1. Browser to Servlet communication.
2. Html to servlet communication.
3. servlet to servlet communication.

In Browser to servlet communication we need to write request url in browser address bar to generate the request. Writing request url in browser address bar is not possible for end user.to overcome this limitation we need to go with html to servelt communication.

Intially,All application which we developed are called browser to servlet communication only. In html to servelt communication, we will create request url by using html programs as hyperlink and form page. The request, which is generated with hyperlink, does not carry data from end user.The request which is generated with formpage carry data from end user.

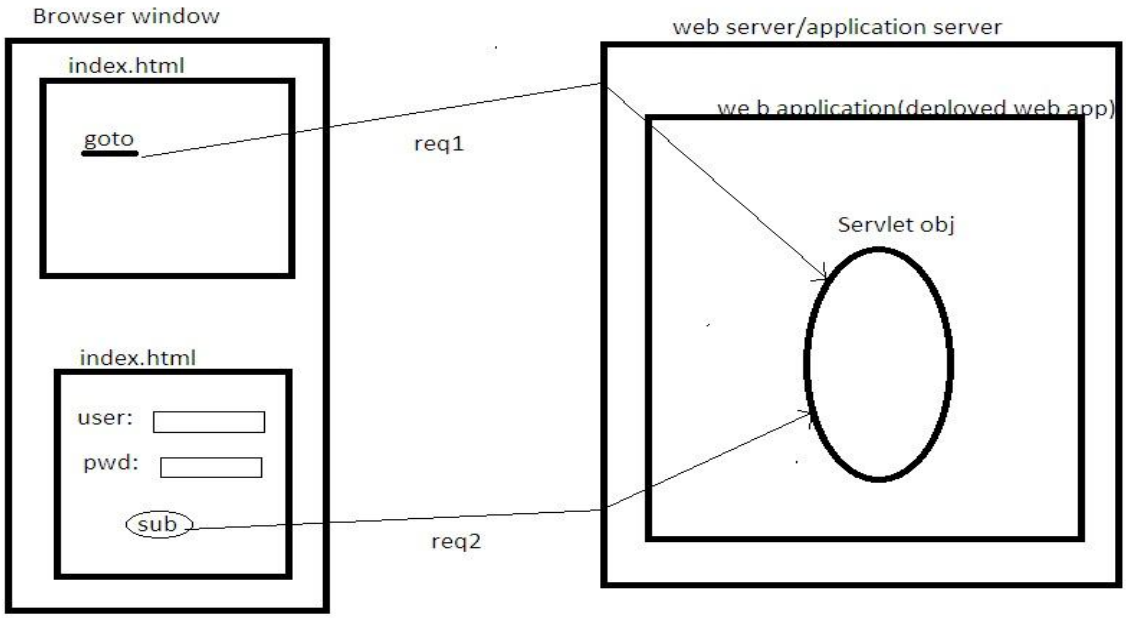


Figure 4

In hyperlink based html to servlet communication we need to take request url of servlet program as href url.

Ex :-goto

In form page based html to servlet communication we need to take request url as action url of form page.

Ex :-<form action="http://localhost:2525/DateApp/test">

-
-
-
-

</form>

- **Example application html to servlet communication:-**

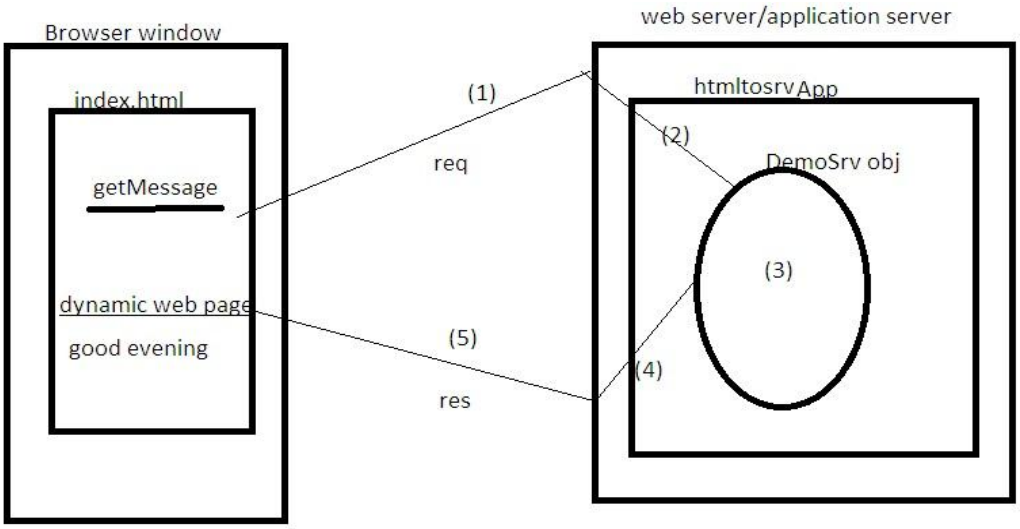


Figure 5

- **Deployment directory structure:-**

```
htmltosrvApp
|
|
|----WEB-INF
        |
        |
        |---------classes
        |                  |
        |                  |
        |------web.xml  |-----DemoSrv.java
                          |-----com.vn
                                  |
                                  |---DemoSrv.class
                                  |
```

- **web.xml:-**

```
<web-app>
<servlet>
<servlet-name>mno</servlet-name>
<servlet-class>com.vn.DemoSrv</servlet-class>
</servlet>
<servlet-mapping>
```

<servlet-name>mno</servlet-name>

<url-pattern>/hurl</url-pattern>

</servlet-mapping>

- **DemoSrv.java:-**

```
package com.vn;

import javax.servlet.*;

import javax.servlet.http.*;

class DemoSrv extends HttpServlet

{

publi void service(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException

{

PrintWriter pw=res.getWriter();

res.setContentType("text/html");

//to get time in 24 hours

Calendar c=Calendar.getInstance();//24 hour time

int h=c.get.(Calendar.HOUR_OF_DAY);

if(h<12)

pw.println("<b><i><center>Good Morning</center></b></i>");

else if(h<16)

pw.println("<b><i><center>Good Afternoon</center></b></i>");

else if(h<20)

pw.println("<b><i><center>Good Evening</center></b></i>");

else

pw.println("<b><i><center>Good night</center></b></i>");

pw.println

("<a href='http://localhost:2525/htmltosrvApp/hurl'>home</a>");
```

```
pw.close();
}
}
```

- **Index.html:-**

```
<b><center>
<a href="http://localhost:2525/htmltosrvApp/hurl">GetMsg</a>
<b></center>
```

- **request url:-**

http://localhost:2525/htmltosrvApp/index.html

- **Html form page to servlet communication:-**

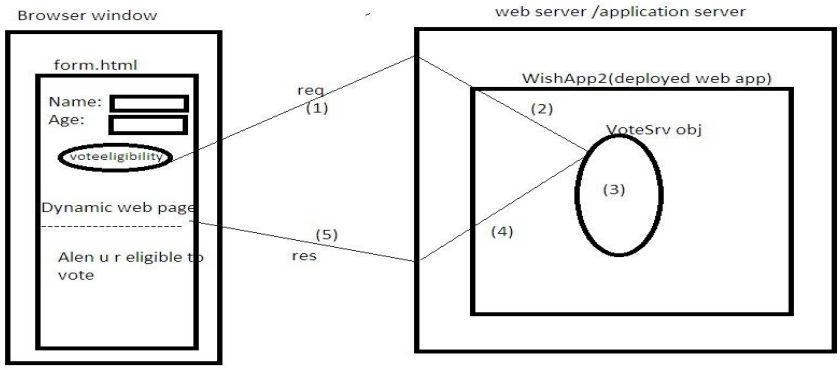


Figure 6

- **Deployment directory structure:-**

```
WishApp2
    |
    |
    |----WEB-INF
    |          |
    |form.html|
              |----classes
              |          |
              |          |
              |-web.xml|---VoteSrv.java
                        |
                        |--com.vn
                              |
                              |---VoteSrv.class
```

We can send the request to servlet program in two methodologies

1. "Get" methodology/method:

   It will carry limited amount of data along with the request.

2. "Post" methodology/method:

   It will carry unlimited amount of data long with the request.

While developing servlet program based HttpServlet class it is not recommanded to used service (-,-) method.it is recommanded to work with doXxx(-,-,) methods. Service (-,-) method is not design based on protocol http but doXxx(-,-) method is design based on protocol http. We will used doGet(-,-) method to process get methodology and doPost(-,-) method to process post methodology.

- **Types of doXxx(-,-) methods:-**

  There are 7 types of doXxx(-,-) are there

  1. doGet(-,-)
  2. doPost(-,-)
  3. doPut(-,-)
  4. doDelete(-,-)
  5. doOption(-,-)
  6. doTrace(-,-)
  7. doHead(-,-)

- **Protocol of doXxx(-,-):-**

  Protected void doGet(HttpServetRequest req,HttpServletResponse res) throws ServletException,IOException

- **form.html:-**

  <form action="http://localhost:2525/WishApp2/wurl" method="GET">

  Name: <input type="text" name="tname">

  Age: <input type="text" name="tage">

```
<input type="submit" value="vote">
</form>
```

- **web.xml:-**

```
<web-app>
<servlet>
<servlet-name>abc</servlet-name>
<servlet-class>com.vn.VoteSrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>abc</servlet-name>
<url-pattern>/wurl</url-pattern>
</servlet-mapping>
</web-app>
```

- **VoteSrv.java:-**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
class VoteSrv extends HttpServlet
{
protected void doGet(HttpServletRequest req,HttpServletResponse res)
                        throws ServletException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("text/html");
//reading input values from request
```

```
String name=req.getParameter("tname");
String cage=req.getParameter("tage");
int age=Integer.parseInt(cage);
if(age>=18)
pw.println("<font color='green'>"+name+"U r aligible to vote</font>");
else
pw.println("<font color='red'>"+name+"U r not aligible to vote</font>");
pw.close();}}
```

- **request url:-**

  http://localhost:2525/WishApp2/form.html

- **Difference between GET and POST:-**

GET	POST
It is a default methodology	It is not a default methodology
It is design to send thedata to server while sending request.	It is design to post the data to server.
It carries limited amount of data upto 256kb	It carries unlimited amount of data
It sends the request fastly.	It sends the request Bit slow
It is not suitable for encryption and file uploading	It is suitable for encryption and file uploading.
we will used doGet(-,-) to process GET methodology	We will used doPost(-,-) to process POST methodology.

- **Applets to Servlet Communication:-**

Applets are compiled java class which is used to send over the network as webpages.Applets are java based web pages. If we want to develop form page with security then we should used applets otherwise we can used html. Bydefault every Applets are untrusted Applets it means once if we download any Applets in browser window it does not perform any operation so there is a security. If we want to develop form page with security use Applets. If we want to develop form page with performance use Html.

- **Example Application:-**

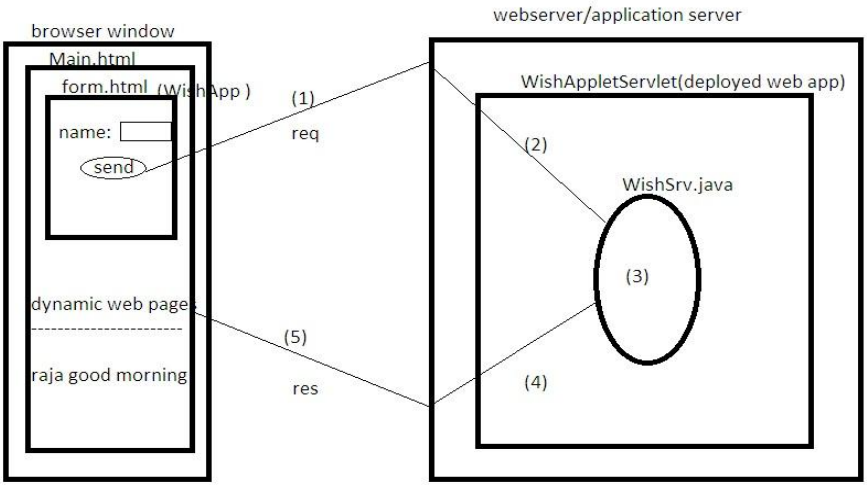


Figure 7

- **Deployment Directory Structure:-**

```
WishAppletServlet
|
|
|----------WEB-INF
|                |----------classes
|                |                |
|                |                |---WishSrv.java
|--Main.html     |                |---com.vn
|--form.html     |---web.xml                |
|--WishApp.java                                    |--WishSrv.class
|--WishApp.class
```

- **Main.html:-**

```
<frameset row="30%,*">

<frame src="form.html" name="f1"/>

<frame name="f2"/>

</frameset>
```

- **form.html:-**

```
<applet code="WishApp.class" width="400" height="400"/>
```

- **WishApp.java:-**

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
class WishApp extends Applet implements ActionListener
{
Label l1;
TextField tf1;
Button b1;
public WishApp()
{
setSize(300,400);
setBackground(Color.RED);
//add components
l1=new Label("Name");
add(l1);
tf1=new TextField(20);
add(tf1);
b1=new Button("send");
add(b1);
//perform action listener
b1.addActionListener(this);
setLayout(new FlowLayout());
}
        public void actionPerformed(ActionEvent ae)
        {  //try
```

```
{
String qstr="?prname="+tf1.getText().replace(' ','+');
URL myurl=new URL("http://localhost:2525/WishAppletServlet/wurl"+qstr);
//get Appletcontext obj
AppletContext apc=getAppletContext();
apc.showDocument(myurl,"f2");
}//try
catch(Exception e)
{
e.printStackTrace();
}
}
}//class
```

- **web.xml:-**

```
<web-app>
<servlet>
<servlet-name>abc</servlet-name>
<servlet-class>com.vn.WishSrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>abc</servlet-name>
<url-pattern>/wurl</url-pattern>
</servlet-mapping>
<welcome-file-list>
<welcome-file>page.html</welcome-file>
</welcome-file-list>
 </web-app>
```

- **WishSrv.java:-**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
class WishSrv extends HttpServlet
{
protected void doGet(HttpServletRequest req,HttpServletResponse res)
                        throws ServletException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("text/html");
//reading input values from request
String name=req.getParameter("prname");
Calendar c=Calendar.getInstance();
int h=c.get(Calendar.HOUR_OF_DAY);
if(h<=12)
pw.println("<h1><b>Good morning</h1></b>");
else if(h<=16)
pw.println("<h1><b>Good Afternoon</h1></b>");
else if(h<=20)
pw.println("<h1><b>Good Evening</h1></b>");
else
pw.println("<h1><b>Good Night</h1></b>");
pw.close();
}}
```

- **Form validation:-**

The processes of checking format and pattern of form data is called form validation and logic is called form validation logic.

- **What is difference b/w form validation logic and bussiness logic:-**

1. **Form validation logic:-**

Checking the format and pattern of form data is called form validation logic

Ex:-To check given numbers are numeric or not is called form validation logic.

2. **Business logic:-**

Checking given input values is inserted or not is called business logic.

Ex:- adding two input values to get the result is called business logic.

There are 4 approaches are there to perform form validation logic they are

- Approach1:

Do form validation at server side which increases network round trips between client and server if form is rejected for multiple times.

- Approach2:

Do form validations at client side, which will reduce network round trips. If client disable javascript code/execution client side form validation will not be done.

- Approach3:

Do form validation at client and server side but by default form validation will be done at server side only,It will issue performance of a system.

- Approach 4:

Do form validation at client and server side but perform server side form validation logic if client side form validation logic has not done.

- **Write a jdbc application to perform form validation logic:-**
  - **Deployment directory structure**

```
ValidationApp
|
|
|------------------WEB-INF
|                          |
|---main.html              |----------classes
|---validation.js          |                  |
|                          |--web.xml         |--VoterSrv.java
                                              |
                                              |--com.vn
                                                      |
                                                      |--VoteSrv.class
```

  - **Main.html:-**

```
<html>
<head>
<script Language="JavaScript" src="Validate.js">
</head>
<form action="/vturl" onsubmit="return validate(this)">
Name:<input type="text" name="name"/>
Age:<input type="text" name="page"/>
<!--  hidden box-->
<input type="hidden" value="no" name="vflag"/>
<input type="submit" value="voteeligibility"/>
</form>
```

  - **Validation.js:-**

```
function validate(frm)
{
frm.vflag.value="yes";
//read form data
var un=frm.pname.value;
```

```
var age=frm.page.value;
//perform client side form validation
if(un=="")
{
alert("user name is mandatory");
frm.pname.focus();
return false;
}
if(age==null)
{
alert("Person age is mandatory");
frm.page.focus();
return false;
}
else
{
if(isNaN(age))
{
alert("age must be numeric value");
frm.page.focus();
frm.page.value="";
return false;
}
}//else
return true;
}
//function
```

- **web.xml:-**

```
<web-app>
<servlet>
<servlet-name>abc</servlet-name>
<servlet-class>com.vn.VoterSrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>abc</servlet-name>
<url-pattern>/vturl</url-pattern>
</servlet-mapping>
</web-app>
```

- **VoterSrv.java:-**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
 public class VoterSrv extends HttpServlet
{
public void service(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("text/html");
//read form data
String name=req.getParameter("pname");
```

```
String tage=req.getParameter("page");
int age=0;
String vstatus=req.getParameter("vflag");
if(vstatus.equals("no"))
{
//server form validation
if(name.equals("")||name==null||name.length()==0)
{
pw.println("<font color='red'>Person name is mandatory</font>");
return;
}
if(tage.equals("")||tage==null||tage.length()==0)
{
pw.println("<font color='red'>Person age is mandatory</font>");
return;
}
else
{
try
{
age=Integer.parseInt(tage);
}
catch(NumberFormatException e)
{
pw.println("<font color='red'>Age should be numeric</font>");
return;
}
```

```
}//else

}//end of if

if(vstatus.equals("yes"))

{

age=Integer.parseInt(tage);

}

if(age<=18)

pw.println("<b><i><font color='green'>"+name+" u r not eligible to
vote</font></b></i>");

else

pw.println("<b><i><font color='green'>"+name+" u r eligible to vote</font></b></i>");

pw.close();

}

}
```

- **Request url:-**

  http://localhost:2525/ValidationApp/main.html

➢ **Write a servlet program to show various components of form page:-**

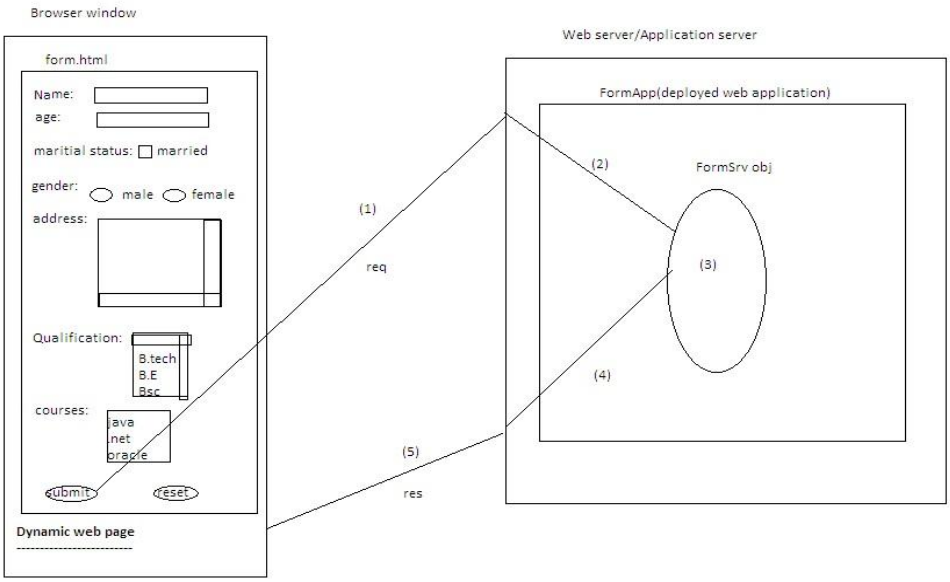
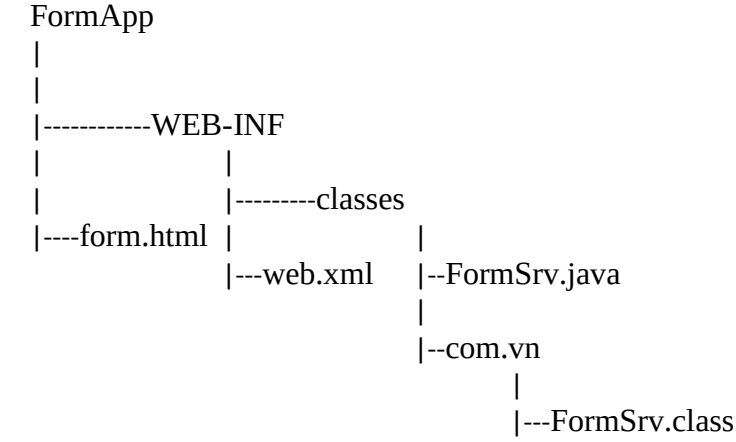


Figure 8

- **Deployment directory structure:-**



- **Form.html:-**

```
<form action="formurl" method="GET">
<table border="1">
<tr>
<td>Name:</td>
<td><input type="text" name="tname"></td>
</tr>
<tr>
<td>Age:</td>
<td><input type="text" name="tage"></td>
</tr>
<tr>
<td>Address:</td>
<td><textarea name="taddrs" cols="20" rows="10">Enter Add</textarea></td>
</tr>
<tr>
<td>Maritial Status:</td>
<td><input type="checkbox" name="ms" value="married">Married</td>
</tr>
```

```
<tr>
<td>Gender:</td>
<td><input type="radio" name="gen" value="m"/>Male
<input type="radio" name="gen" value="f"/>Female
</td>
</tr>
<tr>
<td>Qualification:</td>
<td><select name="qlfy">
<option value="Eng">Engineer</option>
<option value="b.E">B.E</option>
<option value="B.tech">B.tech</option>
</select>
</td>
</tr>
<tr>
<td>Courses:</td>
<td><select name="crs" multiple="yes">
<option value="Eng">Engineer</option>
<option value="b.E">B.E</option>
<option value="B.tech">B.tech</option>
</select>
</td>
</tr>
<tr>
<td><input type="submit" value="submit"></td>
<td><input type="reset" value="reset"></td>
```

```
</tr>
</table>
</form>
```

- **Web.xml:-**

```
<web-app>
<servlet>
<servlet-name>abc</servlet-name>
<servlet-class>com.vn.FormSrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>abc</servlet-name>
<url-pattern>/formurl</url-pattern>
</servlet-mapping>
</web-app>
```

- **FormSrv.java:-**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
 public class FormSrv extends HttpServlet
{
protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("text/html");
```

```
String name=req.getParameter("tname");
int age=Integer.parseInt(req.getParameter("tage"));
String addrs=req.getParameter("taddrs");
String ms=req.getParameter("ms");
String gender=req.getParameter("gen");
String qlfy=req.getParameter("qlfy");
String[] crs=req.getParameterValues("crs");
if(gender.equalsIgnoreCase("M"))
{
if(age<=5)
pw.println(name+"<b><i><Center>you r a baby</b></i>");
else if(age<=16)
pw.println(name+"<b><i><Center>you r a teen boy </b></i>");
else if(age<=20)
pw.println(name+"<b><i><Center>you r a young boy</b></i>");
else
pw.println(name+"<b><i><Center>you r very old man</b></i>");
}//if
if(gender.equalsIgnoreCase("F"))
{
if(age<=5)
pw.println(name+"<b><i><Center>you r a baby</b></i>");
else if(age<=16)
pw.println(name+"<b><i><Center>you r a teen Girl </b></i>");
else if(age<=20)
pw.println(name+"<b><i><Center>you r a young Girl</b></i>");
else
```

```
pw.println(name+"<b><i><Center>you r very old women</b></i>");
}//if
pw.println("<br>Name:"+name);
pw.println("<br>Age:"+age);
pw.println("<br>gender:"+gender);
pw.println("<br>Maritial status:"+ms);
pw.println("<br>Address:"+addrs);
pw.println("<br>Qualification:"+qlfy);
if(crs!=null)
{
for(int i=0;i<crs.length;i++)
pw.println(crs[i]+"...");
}
pw.close();
}}
```

- **File Uploading:-**

The process of capturing/select file from client machine file system and storing to server machine file system is called uploading and reverse is called file downloading.

While Dealing with Matremony app,Job portals app and profile management app we need to upload and download the file.We can place form component in form page by using select File:<input type="file" name="f1">

There is no specific API to perform file uploading in Servlet API, so we will used third party API called javazoom API. Javazoom API comes in ZIP file .once if we extract this zip file it will give 3 jar files.

1. uploadBean.java
2. struts.jar
3. cos.jar

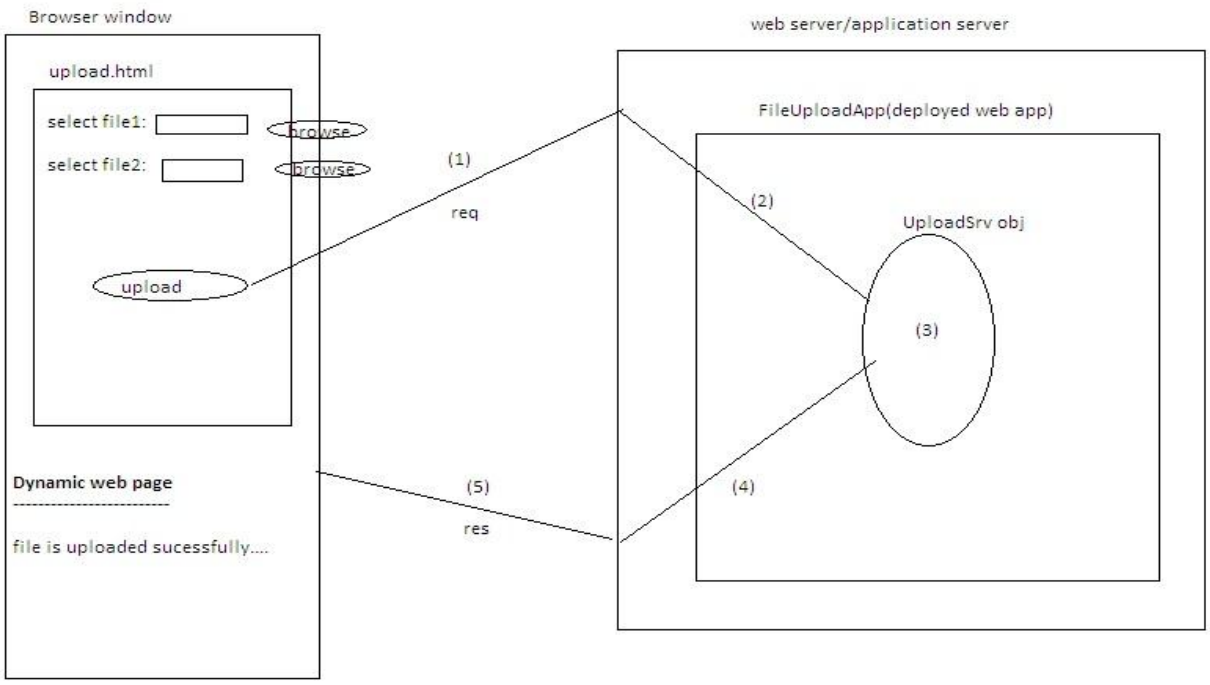


Figure 9

- **Deployment directory structure:-**

```
FileUploadApp
    |
    |------------------WEB-INF
    |                        |
    |----upload.html         |-------------classes
                             |                  |
                             |---web.xml        |---UploadSrv.java
                             |                  |
                             |---lib            |---com.vn
                               |                        |
                               |-uploadBean.jar         |--UploadSrv.class
                               |-struts.jar
                               |-cos.jar
```

- **Upload.html:-**

```
<form action="test1" method="post" enctype="multipart/form-data">
Select file1:<input type="file" name="f1"><br>
Select file2:<input type="file" name="f2"><br>
<input type="submit" value="Upload"/>
</form>
```

- **Web.xml:-**

```
<web-app>
<servlet>
<servlet-name>abc</servlet-name>
<servlet-class>com.vn.UploadSrv</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>abc</servlet-name>
<url-pattern>/test1</url-pattern>
</servlet-mapping>
</web-app>
```

- **UploadSrv.java:-**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import javazoom.upload.*;
public class UploadSrv extends HttpServlet
{
public void service
(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException
{
PrintWriter pw=res.getWriter();
try{
UploadBean upb=new UploadBean();
upb.setFolderstore("C:/store");
```

```
upb.setOverwrite(false);
//create parse for parsing form data
MultipartFormDataRequest nreq=new MultipartFormDataRequest(req);
upb.store(nreq);//completes file uploading
//Display the names of uploaded file are
pw.println("<b>The name of uploaded file are </b>");
Hashtable ht=nreq.getFiles();
Enumeration e=ht.elements();
while(e.hasMoreElements())
{
UploadFile f1=(UploadFile)e.nextElement();
pw.println(f1.getFileName()+"<br>");
}
}
//try
catch(Exception e)
{
pw.println(e);
}
//catch
pw.println("<b>your files are safely uploaded</b>");
pw.close();
}
}
```

- **Request url:-**

  http://localhost:2525/FileUploadApp/upload.html

- **Write a demonstration on stateless behaviors:-**

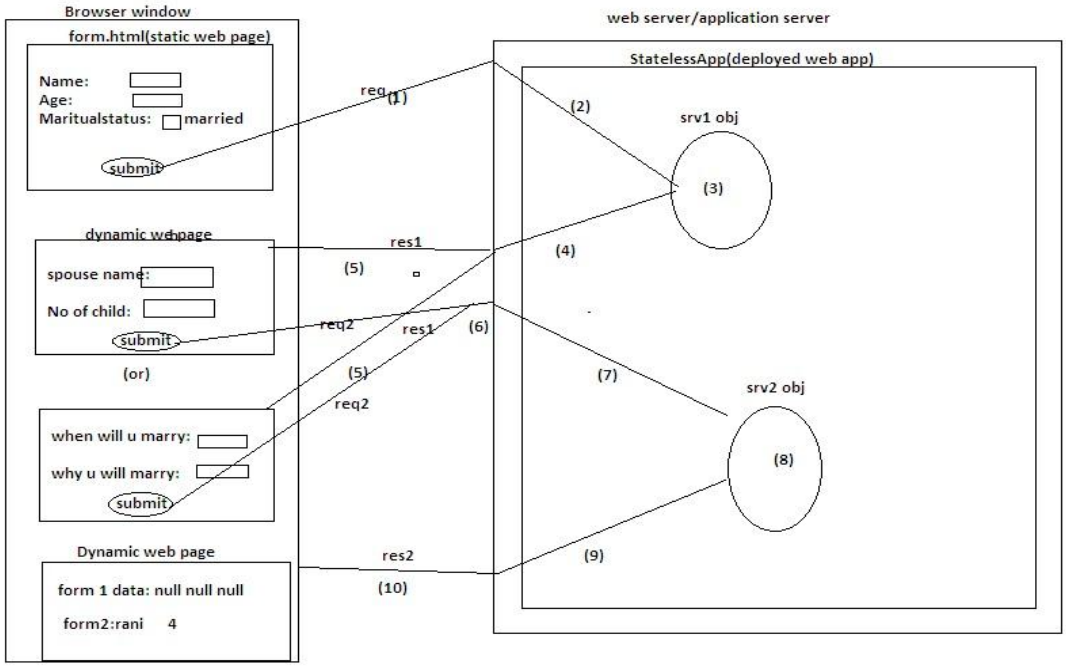


Figure 10

In the above diagram shows stateless behavious of the application along with dynamic form page generation that means form 2 content is generated based on form 1 data.but while processing request 2 we can't access form1 data this is nothing but stateless behaviour of an application.

- **Deployment directory structure:-**

```
StatelessApp
|
|--------------WEb-INF
|                  |
|---form.html      |-----------classes
                   |                 |
                   |---web.xml       |--Srv1.java
                                     |--Srv2.java
                                     |
                                     |--com.vn
                                            |
                                            |--Srv1.class
                                            |--Srv2.class
```

- **Form.html:-**

```
<form action="s1url">
<b>Name:<input type="text" name="pname"><br>
<b>Age:<input type="text" name="page"><br>
<b>Maritial status:</b><input type="checkbox" value="married" name="ms"><br>
<input type="submit" value="submit">
</form>
```

- **Web.xml:-**

```
<web-app>
<servlet>
<servlet-name>abc</servlet-name>
<servlet-class>com.vn.Srv1</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>abc</servlet-name>
<url-pattern>/s1url</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>xyz</servlet-name>
<servlet-class>com.vn.Srv2</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>xyz</servlet-name>
<url-pattern>/s2url</url-pattern>
</servlet-mapping>
</web-app>
```

- **Srv1.java:-**

```
package com.vn;

import javax.servlet.*;

import javax.servlet.http.*;

import java.io.*;

import java.util.*;

class Srv1 extends HttpServelt

{

public void service(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException

{

String pname=req.getParameter("pname");

int page=Integer.parseInt(req.getParameter("page"));

String mstatus=req.getParameter("ms");

PrintWriter pw=res.getWriter();

res.setContentType("text/html");

if(mstatus==null)

mstatus="married";

if(mstatus.equals("married"))

{

pw.println("<form action='s2url'>");

pw.println("<b>Spouse name:<input type='text' name='f1'><br>");

pw.println("<b>No of child:<input type='text' name='f2'><br>");

pw.println("<input type='submit' value='submit'>");

pw.println("</form>");

}

else
```

```
{
pw.println("<form action='s2url'>");
pw.println("<b>when will u marry:<input type='text' name='f1'><br>");
pw.println("<b>why will u marry:<input type='text' name='f2'><br>");
pw.println("<input type='submit' value='submit'>");
pw.println("</form>");
}
pw.close();
}}
```

- **Srv2.java:-**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
class Srv2 extends HttpServelt
{
public void service(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("text/html");
pw.println("<b><i>Reading form1 data </i></b>");
pw.println("<b><i>Name "+req.getParameter("pname")+"</i></b>");
pw.println("<b><i>Age "+req.getParameter("page")+"</i></b>");
pw.println("<b><i>Maritial status"+req.getParameter("ms")+"</i></b>");
pw.println("<b><i>Reading form2 data </i></b>");
```

```
pw.println
("<b><i>spouse/when will u marry "+req.getParameter("f1")+"</i></b>");
pw.println("<b><i>no child/why will u marry "+req.getParameter("f2")+"</i></b>");
pw.close();
}
}
```

- **Request url:-**

  http://localhost:2525/StatelessApp/form.html

➢ **Session:-**

The Process of continues and related operations performed on web application from client with the support of multiple request and response is called session.

Ex :->From login to logout in gmail is called session.

>Start java class and end of java class is called session.

As Protocol Http is stateless which makes web application as stateless it means no we resource program can access precious request data while processing current request during a session. To perform session tracking we need to deal with Http protocol and servlets of javax.servlet.http.Session.*;

➢ **Session Tracking and Session management:-**

To make web application as statefull web application even though protocol Http is stateless then we need to work with Session Tracking Session tracking can be done in the following ways.

1. Using hiddenbox field
2. http cookies
3. httpSession with cookies
4. HttpSession with URL rewriting

In stateless web application, no web resource program can access previous data while processing the current request during a session. In Statefull web application web resource program can access previous data while processing the current request during a session.

1. Using hiddenbox field:-

It hiddenbox values is a invsible text fields of form page. hidden box values goes to server as a request parameter values.

<input type="hidden" value="h1" name="hello"/>

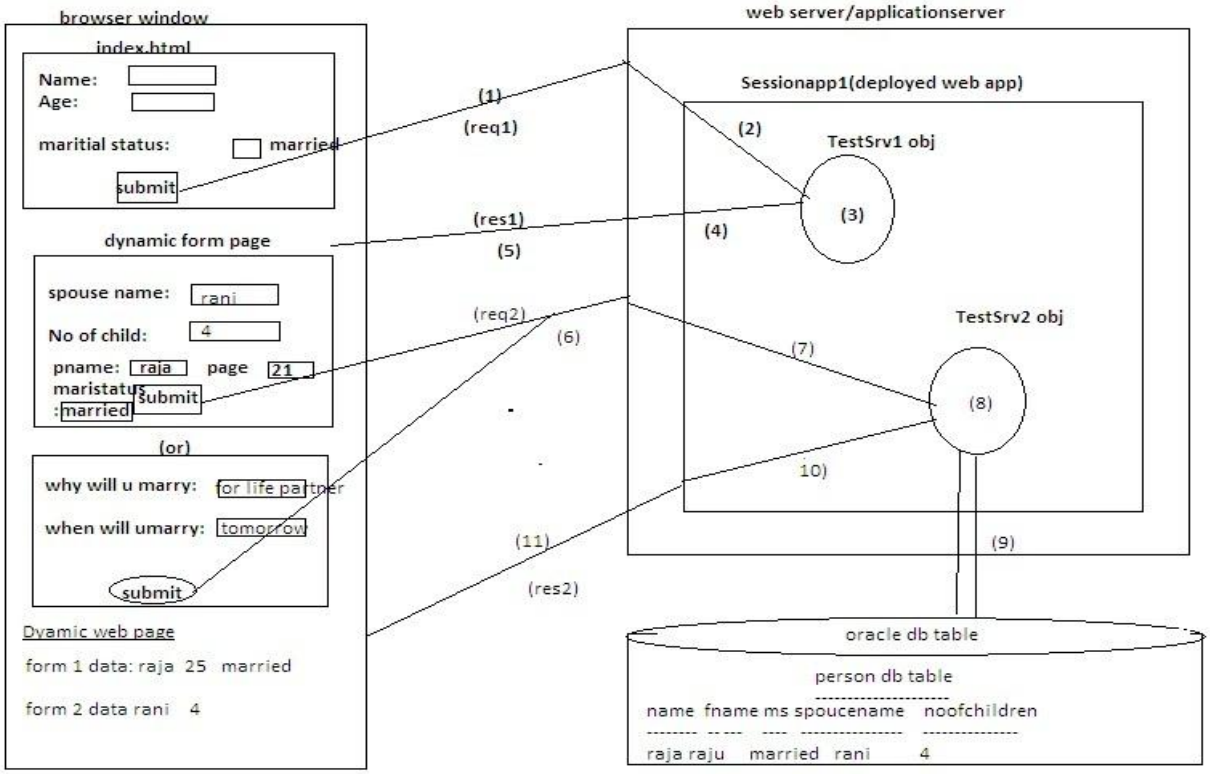


Figure 11

- **Deployment directory structure:-**

```
Sessionapp1
|
|--------------WEB-INF
|                  |
|---index.html     |-----------classes
                   |                  |
                   |---web.xml        |--TestSrv1.java
                                      |--TestSrv2.java
                                      |
                                      |--com.vn
                                               |
                                               |-TestSrv1.class
                                               |-TestSrv2.class
```

- **Index.html:-**

  <form action="t1url" method="GET">

  Name:<input type="text" name="tname">

  Father's Name:<input type="text" name="fname">

  Maritial Status:<input type="radio" name="ms" value="married">Married   : :

  <input type="radio" name="ms" value="single">Single

  <input type="submit" value="submt">

  </form>

- **Web.xml**

  <web-app>

  <servlet>

  <servlet-name>abc</servlet-name>

  <servlet-class>com.vn.TestSrv1</servlet-class>

  </servlet>

  <servlet-mapping>

  <servlet-name>abc</servlet-name>

  <url-pattern>/t1url</url-pattern>

```
</servlet-mapping>
<servlet>
<servlet-name>xyz</servlet-name>
<servlet-class>com.vn.TestSrv2</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>xyz</servlet-name>
<url-pattern>/t2url</url-pattern>
</servlet-mapping>
</web-app>
```

- **TestSrv1.java:-**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
 public class TestSrv1 extends HttpServlet
{
public void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
{
PrintWriter pw=res.getWriter();
res.setContentType("text/html");
String pname=req.getParameter("tname");
String pfname=req.getParameter("fname");
String status=req.getParameter("ms");
if(status.equals("single"))
{
```

```
pw.println("<form action='t2url' method='GET'>");
pw.println("When do yoy marry<input type='text' name='f2t1'><br>");
pw.println("Why do yoy marry<input type='text' name='f2t2'><br>");
pw.println("<input type='hidden' name='hname' value="+pname+">");
pw.println("<input type='hidden' name='hfname' value="+pfname+">");
pw.println("<input type='hidden' name='hms' value="+status+">");
pw.println("<input type='submit' value='submit'>");
pw.println("</form>");
}
else
{
pw.println("<form action='t2url' method='GET'>");
pw.println("Spouse name<input type='text' name='f2t1'><br>");
pw.println("no of children<input type='text' name='f2t2'><br>");
pw.println("<input type='hidden' name='hname' value="+pname+">");
pw.println("<input type='hidden' name='hfname' value="+pfname+">");
pw.println("<input type='hidden' name='hms' value="+status+">");
pw.println("<input type='submit' value='submit'>");
pw.println("</form>");
}
pw.close();
}//service
}//class
```

- **TestSrv2.java:-**

```
package com.vn;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
import java.io.*;

import java.util.*;

import java.sql.*;

 public class TestSrv2 extends HttpServlet

{

public void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException

{

PrintWriter pw=res.getWriter();

res.setContentType("text/html");

String name=req.getParameter("pname");

String fname=req.getParameter("pfname");

String ms=req.getParameter("status");

String f2val1=req.getParameter("f2t1");

String f2val2=req.getParameter("f2t2");

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","sys
tem");

PreparedStatement ps=con.prepareStatement

("insert into person values(?,?,?,?,?)");

ps.setString(1,name);

ps.setString(2,fname);

ps.setString(3,ms);

ps.setString(4,f2val1);

ps.setString(5,f2val2);
```

```
ps.executeUpdate();
pw.println("<h1>Record is Inserted...</h1>");
}
catch(Exception e)
{
e.printStackTrace();
}
pw.println("<br>form 1 data "+name+" "+fname+" "+ms);
pw.println("<br>form 2 data "+f2val1+" "+f2val2);
pw.close();
}//service
}//class
```

- **Request url:-**

  http://localhost:2525/Sessionapp1/index.html