

## Servlets:-

## Servlets introduction:-

- servlets are used to write the programming at server side.
  - Servlet is a object executed at server side.
  - servlets are used to design request response programming model.
  - A Jakarta servlet (formerly java servlet)
5. present version of servlets is: servlets 4.0

Different servers:-

vendors

- Tomcat
- WebLogic
- Websphere
- Jboss
- Glassfish

Apache software foundation  
Oracle  
IBM  
redHat  
sun micro systems (oracle)

There are two types of servers:-

- a. web servers:- supports only web applications.  
web server contains web container.

```
web container = servlet container + jsp engine
                servlet engine      jasper
                catalina
```

- 1.servlet container will execute the servlets.
- 2.JSP engine will execute JSP.

Eg:-tomcat 4.0, jetty, IIS, ..

- b. Application server:-supports both web applications and enterprise applications. Application server contains both web container and EJB container.

Eg:-tomcat, glassfish, websphere, weblogic.

### Servlets predefined packages:-

javax.servlet----- this package will handle the normal request & response  
 javax.servlet.http----- this package will handle the http request & response  
 javax.servlet.annotation-- this package will handle the annotations.

## Download & install the server:- tomcat

download Link:

<https://tomcat.apache.org/download-90.cgi>

zip (pgp, sha512) tar.gz (pgp, sha512)

32-bit Windows zip (pgp, sha512)

64-bit Windows zip (pgp, sha512)

32-bit/64-bit Windows Service Installer (pgp, sha512) [download this file]

2 Install the software:

a. select the type of install: full

b. port numbers

(by default it is taking normal)

http port: by default 8080: change the port number to any four digit number: 9999

c. give the username & password

user name: tomcat

password: tomcat

d. run apache tomcat : uncheck the check box (by default checked mode)

Once the installation completed the folder Location is,

C:\Program Files\Apache Software Foundation\Tomcat 9.0

Tomcat contains total 7-folders,

1. bin: it contains binary executables to start & stop the server.  
start the server using tomcat service.

Once the server started by using client send the request using below url:-  
<http://localhost:9999/>

the server will give tomcat home page.

manager app: this option shows the list of applications

To check the predefined examples: click on servlet examples

<http://localhost:9999/examples/>

2. config: it configurations

server.xml: this file contains port numbers information, to change the port number use this file tomcat-user.xml: this file contains username & password  
context.xml

3. temp :

to change the username & password use this file

: to configure the connection pool information use this file.

it will store the temp files

4. Log it will store the log file when the server starts time when server stops time

error information: time

5. web apps: web application

This folder contains all web application deployed in server.

Subscribe Now

6. work:

every jsp will convert into servlet format,

the converted servlets are stored in work folder.

7. Lib: this folder contains jar files

Servlets :

the servlet predefined classes & interfaces are present in jar file servlet-api.jar.

Servlet api is given by sun micro System

API contains more interfaces & less classes

The servlet implementations are given by server vendors

tomcat

---> servlet-api.jar

glassfish ---> jee.jar

We are developing the servlet project: Tomcat server: servlet-api.jar

Later we decided to change the server, in this case no need to change the project code just change the server implementations jar file.

CGI request processing vs. Servlets request processing

CGI vs Servlets:-

For every request one process is created. For every request one thread is created.

Technology(j2ee)

Servlets JSP

framework

structs

JSF

83. Web application contains multiple resources Like,

java files

xml files

html files

jsp files

JS files

.jar files.....etc

It is not recommended to place all the files in single folder. so web application follows standard directory structure.

Tomcat web application directory structure:

LoginApplication

(project name)

|---> .html

|---> .js

|---> .jsp

|---> images

---> WEB-INF (all are capital Letters)

|---> .xml

|---> Lib(folder)

|---> \*.jar

---> classes(folder)

--> \*.class

BEclipse IDE web applicaiton directory structure:

LoginApplication

|---> webcontent

|---->.html

|---> .js

|---> .jsp

|---> images

|---> WEB-INF

--> .xml

--> Lib

|--->\*.jars

---> java resources/

---->\*.class

There are three ways to create a servlet,

a. implements Servlet

b. extends GenericServlet c. extends HttpServlet

```
package javax.servlet;
```

```
import java.io.IOException;
```

```
public interface Servlet
```

```
{
```

```
void init(final ServletConfig pe) throws ServletException;
```

```
ServletConfig getServletConfig();
```

```
void service(final ServletRequest req, final ServletResponse res) throws
ServletException, IOException;
String getServletInfo();
void destroy();
```

```
class MyServlet implements Servlet
{
    override 5-methods
}
```

There are three life cycle methods, (these methods are automatically called by server)

- 1.init()
- 2.service()
- 3.destroy()

## Servlet Life cycle methods:-

1. The servlet is constructed, then initialized with the init method.
2. Any calls from clients to the service method are handled.
3. The servlet is taken out of service, then destroyed with the destroy method, then garbage collected and finalized.

In addition to the life-cycle methods, this interface provides the `getServletConfig` method, which the servlet can use to get any startup information, and the `getServletInfo` method, which allows the servlet to return basic information about itself, such as author, version, and copyright.

`init()` servlet initialization

:

The servlet initialization done only once when we send the first request.

1-request--->init() service()  
                  2ns     2ns     =4ns

2-request --->     service()=2ns  
                          2ns

3-request --->     service()=2ns  
                          2ns

When we do the servlet initialisation during servlet startup, then when we send first request directly service method will be executed.

To do the servlet initialisation during servlet start-up use `<load-on-startup>` in `web.xml`.

In `<load-on-startup>` it will take positive number it is a priority.

If we are use negative or zero number without error it wont prove any data.

`service()`: request processing

Any calls from clients to the service method are handled.

This method contains the business Logics

destroy():The servlet is taken out of service, then destroyed with the destroy method, then garbage collected and finalized.

We will write logics of servlets using service().

Note:

corejava contain main method called by JVM.

servlets does not contains main method, servlets contains life cycle methods called by server.

Void setContenttype(java.lang.String type);

Sets the content type of the response being sent to the client.

Default response type is:text/html.

response.setContenttype("text/html");

to add response into response object using PrintWriter object.

Writer.println("This is my first Application");

To print the response.

## Second Approach to Create Servlet:-

abstract class GenericServlet implements Servlet

{

4-methods default implementations completed 1-abstract method service()

}

Approach-2

class MyServlet extends GenericServlet

{

Override only 1-method service()

}

## HttpServlet:-

Using Http mechanism it is possible to send different types of requests.

We have total 7 request types:-

Put-doPut()  
Post-doPost()  
Get-doGet()  
Delete-doDelete()  
Trace-doTrace()  
Option-doOption()  
Head-doHead()

The default request type is get() request.

### Eg:-Servlet example

```
import javax.servlet.ServletException;
import javax.servlet.ServletResponse;
public class FirstServlet implements Servlet {
    public FirstServlet() {
    }
    public void init(ServletConfig config) throws ServletException {
        // TODO Auto-generated method stub
        System.out.println("init method() started");
    }
    public void destroy() {
        // TODO Auto-generated method stub
        System.out.println("destroy method() executed");
    }
    public ServletConfig getServletConfig() {
        // TODO Auto-generated method stub
        return null;
    }
    public String getServletInfo() {
        // TODO Auto-generated method stub
        return null;
    }
    public void service(ServletRequest request, ServletResponse response) throws ServletException {
        // TODO Auto-generated method stub
        System.out.println("service method() started");
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();
        writer.println("<h1>This is first servlet application</h1>");
    }
}
```

### Eg:-GenericServlet example

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class SecondServlet extends GenericServlet {
    private static final long serialVersionUID = 1L;
    public SecondServlet() {
        super();
        // TODO Auto-generated constructor stub
    }
    public void service(ServletRequest request, ServletResponse response) throws ServletException {
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();
        writer.println("This is second Servlet Application");
    }
}
```

## Eg:-HttpServlet Example:-doGEt() example

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ThirdServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public ThirdServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
        // TODO Auto-generated method stub
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();
        writer.println("This is third application");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
        // TODO Auto-generated method stub
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();
        writer.println("This is third application post approach");
    }
}
```

## HttpServlet example using submit button:-

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ThirdServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public ThirdServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
        // TODO Auto-generated method stub
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();
        writer.println("This is third application");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
        // TODO Auto-generated method stub
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();
        writer.println("This is third application post approach");
    }
}
```

Form.html:-



from the client we can send the request to server in three ways:

1. By typing the url in browser
2. By clicking the submit button in html pages
3. By clicking the hyper links

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <form method="post" action="ThirdServlet">
9 <input type="submit" value="clickme">
10 </form>
11
12 </body>
13 </html>
```

from the client we can send the request to server in three ways:

1. By typing the url in browser
2. By clicking the submit button in html pages
3. By clicking the hyper links

Q. How the server is calling doGet()?

Ans:

class MyServlet extends HttpServlet

```
{
}
```

HttpServlet contains to service() method,

- a. public service()
  - b. protected service()
1. public service():

public void service (final ServletRequest req, final ServletResponse res) throws ServletException, IOException {

HttpServletRequest request; HttpServletResponse response;

```
try {
}
```

request = (HttpServletRequest)req; response = (HttpServletResponse)res;

```
catch (ClassCastException e) {
```

throw new ServletException (HttpServlet.IStrings.getString("http.non\_http"));

this.service(request, response);

```
}
}
```

- a. This method called by server.
- b. This method will convert normal request & response into Http types.
- c. This method will call protected service by passing http types of arguments.

2. protected service()

protected void service(final HttpServletRequest req, final HttpServletResponse resp) throws ServletException, IOException {

final String method = req.getMethod();

```

if (method.equals("GET")) {
}
this.doGet(req, resp);
else if (method.equals("HEAD")) { this.doHead (req, resp);
}
else if (method.equals("POST")) { this.doPost(req, resp);
}
else if (method.equals("PUT")) { this.doPut (req, resp);
}

}
else if (method.equals("DELETE")) {
}
this.doDelete(req, resp);
else if (method.equals("OPTIONS")) {l this.doOptions (req, resp);
}
else if (method.equals("TRACE")) {
}
this.doTrace(req, resp);
else {
}
String errMsg =
HttpServlet.IStrings.getString("http.method_not_implemented");

```

- a. protected service() called by public service()
  - b. This method is identifying the request type then calling the request method.
- Client sends request  
server will call public service()  
public service() will call protected service()  
protected service will call doGet()

## Request url mappings:-

### 1.Exact match method:-

The servlet contains only one url, so the client can access the servlet using that url.

```

<servlet-mapping>
<servlet-name>FirstServlet</servlet-name>
<url-pattern>/FirstServlet</url-pattern>
</servlet-mapping>

```

<http://localhost:9999/FirstApplication/FirstServlet--valid>

http://localhost:9999/FirstApplication/First--invalid

http://localhost:9999/First Application/FirstServlet/abc--invalid

## 2. directory match method:-

I want to execute single servlet for multiple urls, the urls are starting with specific data

```
<servlet-mapping>
<servlet-name>FirstServlet</servlet-name>
<url-pattern>/abc/*</url-pattern>
</servlet-mapping>
```

http://localhost:9999/FirstApplication/abc/xyz-valid  
http://localhost:9999/FirstApplication/abc-valid  
http://localhost:9999/FirstApplication/abc/abc-valid  
http://localhost:9999/FirstApplication/xyz/abc-invalid

## 3.extension match method:-

This servlet will be executed when we send urls having any extensions(.do,.am,.ch,...)

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

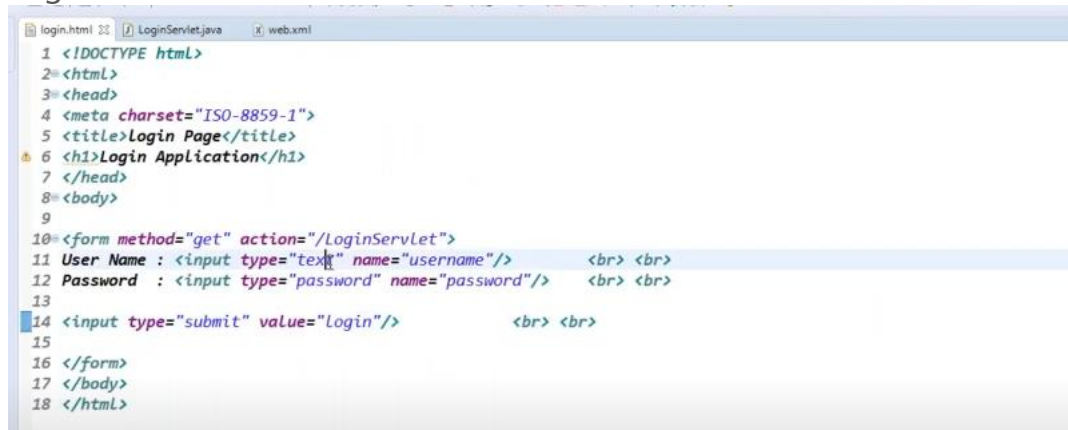
http://localhost:9999/FirstApplication/first.do --valid  
http://localhost:9999/FirstApplication/f123.do --valid  
<http://localhost:9999/FirstApplication/abc/xyz.do--valid>

## Login Application Processing:-

LoginServlet:-

```
7
8 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Serv
9 // General setting
10 response.setContentType("text/html");
11 PrintWriter writer = response.getWriter();
12
13 //Get the requested data
14 String username = request.getParameter("username");
15 String password = request.getParameter("password");
16
17 // request processing logics
18 if(username.equalsIgnoreCase("ratan") && password.equals("anu"))
19 { writer.println("Login Success...."+username);
20 }
21 else
22 { writer.println("Login Fail...try with valid data....");
23 }
24 }
```

## Login.html:-



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="ISO-8859-1">
5 <title>Login Page</title>
6 <h1>Login Application</h1>
7 </head>
8 <body>
9
10 <form method="get" action="/LoginServlet">
11 User Name : <input type="text" name="username"/> <br> <br>
12 Password : <input type="password" name="password"/> <br> <br>
13
14 <input type="submit" value="Login"/> <br> <br>
15
16 </form>
17 </body>
18 </html>
```

Ex:

```
//login.html <!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"> <title>login Page</title>
<h1>Login Application</h1> </head> <body>
<form method="get" action="./LoginServlet">
User Name: <input type="text" name="username"/><br> <br>

Password: <input type="password" name="password"/><br> <br>

<input type="submit" value="login"/><br> <br>

</form>
</body> </html>
<br> <br>
<br> <br>
```

## Filter

=====

Pre processing or request

client-F1---F2--- Servlet

post processing of response

Note: The request goes from filters The response back from filter

Filter Creation:

class MyFilter implements Filter

{

init()

doFilter(): here we will write the filter logics

destroy()

```
}
```

A filter is an object that performs filtering tasks on either, the request to a resource (a servlet or static content), or on the response from a resource, or both.

Filters perform filtering in the `doFilter` method.

Filters are configured in the deployment descriptor of a web application.

```
<filter>
```

```
<filter-mapping>
```

Examples that have been identified for this design are:

Authentication Filters

Logging and Auditing Filters

Image conversion Filters

Data compression Filters

Encryption Filters

Tokenizing Filters

Filters that trigger resource access events

XSL/T filters

Mime-type chain Filter

```
void destroy()
```

Called by the web container to indicate to a filter that it is being taken out of service.

`void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)` The `doFilter` method of the Filter is called by the container each time a request/response pair is passed a resource at the end of the chain.

through the chain due to a client request for a resource at the end of the chain.

```
-void
```

```
init(FilterConfig filterConfig)
```

Called by the web container to indicate to a filter that it is being placed into service.

ex:

```
class MyFilter1 implements Filter
```

```
{ }
```

```
class MyFilter2 implements Filter
```

```
{ }
```

```
class MyServlet extends RegServlet
```

```
{
```

```
}
```

`void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)` throws `IOException`,

`ServletException`

`request` :stores the client requested data

`response` :filter response

`Chain`: forward the request to next filter

A FilterChain is an object provided by the servlet container to the developer giving a view into the invocation chain of a filtered request for a resource.

This FilterChain contains only one method that is doFilter()

void doFilter(ServletRequest request, ServletResponse response) throws IOException, ServletException

Causes the next filter in the chain to be invoked, or if the calling filter is the last filter in the chain, causes the resource at the end of the chain to be invoked.

Parameters:

request the request to pass along the chain.

response the response to pass along the chain.

Note:

Filter interface contains doFilter() method where we will write the filter logics.

FilterChain interface contains doFilter() method used to for

Filter interface contains doFilter() method taking three args( request response chain)

FilterChain interface contains doFilter() method taking two args(request response)

Filters example:-

Myfillers1.java:-

```
package com.tcs;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

/**
 * Servlet Filter implementation class myfilter1
 */
public class myfilter1 implements Filter {

    /**
     * Default constructor.
     */
    public myfilter1() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @see Filter#destroy()
     */
}
```

```

    public void destroy() {
        // TODO Auto-generated method stub
    }

    /**
     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
     */
    public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();
        int age=Integer.parseInt(request.getParameter("uage"));

        if(age>20) {
            chain.doFilter(request, response);
        }
        else {
            writer.println("you are not eligible for marriage and
need to have above 20");

            request.getRequestDispatcher("form.html").include(request, response);
        }
    }
    /**
     * @see Filter#init(FilterConfig)
     */
    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }
}

```

Myfillers2.java:-

```

package com.tcs;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

/**
 * Servlet Filter implementation class myfilter2
 */
public class myfilter2 implements Filter {

    /**
     * Default constructor.
     */
    public myfilter2() {
        // TODO Auto-generated constructor stub
    }

    /**
     * @see Filter#destroy()

```

```

    */
    public void destroy() {
        // TODO Auto-generated method stub
    }

    /**
     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
     */
    public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();

        String uaddr=request.getParameter("uaddr");
        if(uaddr.equalsIgnoreCase("Hyderabad")) {
            chain.doFilter(request, response);

        }
        else {
            writer.println("this application only associated to hyd
person");

            request.getRequestDispatcher("form.html").include(request, response);
        }
    }
    /**
     * @see Filter#init(FilterConfig)
     */
    public void init(FilterConfig fConfig) throws ServletException {
        // TODO Auto-generated method stub
    }
}

```

### Form.html:-

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form method="post" action="RegServlet">
    user name:<input type="text" name="uname"/><br>
    user age:<input type="text" name="uage"/><br>
    user address:<input type="text" name="uaddr"/><br>
    <input type="submit" value="registration">

</form>

</body>
</html>

```

### RegServlet.java:-

```

package com.tcs;

import java.io.IOException;
import java.io.PrintWriter;

```



```

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class RegServlet
 */
public class RegServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public RegServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request,
    HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter writer=response.getWriter();
        String uname=request.getParameter("uname");
        String uaddr=request.getParameter("uaddr");
        String uage=request.getParameter("uage");

        writer.println("your reg details are");
        writer.println(uname);
        writer.println(uage);
        writer.println(uaddr);
        writer.println("guru is good boy always helps friends");
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
    HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }

}

```

## Session Tracking machanism:-

- step 1: College details
- step 2: personal details
- step 3: cast details

step 4: Qualification information

At last it will display all the details then check the all details click on submit....

To maintain the client previous request details at the time of processing later request use session tracking mechanism.

There are three types of Session Tracking mechanism,

- a. HttpSession Tracking mechanism.
- b. Cookies session tracking mechanism.
- c. url rewriting session tracking mechanism.

HttpSession Tracking mechanism:

Here we will maintain the client previous request details at the time of processing later request with the help of HttpSession object.

Note:Session object is specific to the user.

1 form1.html

2 form2.html

form1.html:-

user name : xxx

```
user age: xxx next          FirstServlet    get the requested data
                                next          Create the HttpSession object.
                                place the data in session object.
                                render the second form to client.
```

form2.html:-

user qualification :

```
user designation :          SecondServlet  get the requested data
                                Locate the object : already object
                                is present place the data in session
                                object
                                place the data in session object.
```

render the Third form to client

form3.html:-

user email:

user mobile :

display                      DisplayServlet                      2-details----- request

4-details----- get it  
from session object

```
request.getSession();
```

// here the argument is by default true

It will create the new session object

```
request.getSession(false);
```

If the object is available it will locate the object. If the object is not available it will throw exception.

Note :

first request use              request.getSession();

second & third request

Note:

Last form data stored in request object.

ALL previous forms data will be stored in Session object.

Advantages:

Here we will maintain the previous requested data at server side using Session object so data is secure.

Limitation:

If number of request are increased, the session objects are increased at server side so it will effect on performance.

```
form1.html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<body bgcolor="pink">
<center>
<form method="post" action="FirstServlet">
    Name : <input type="text" name="uname"/><br><br>
    Age : <input type="text" name="uage"/><br><br>
    <input type="submit" value="Next"/>
</form>
</center>
</body>
</html>
```



```
form2.html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<body bgcolor="pink">
<center>
<form method="post" action="SecondServlet">
    Qualification : <input type="text" name="uqual"/><br><br>
    Designation : <input type="text" name="udesig"/><br><br>
    <input type="submit" value="Next"/>
</form>
</center>
</body>
</html>
```



```
form3.html
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<body bgcolor="pink">
<center>
<form method="post" action="DisplayServlet">
    Email : <input type="text" name="email"/><br><br>
    Mobile : <input type="text" name="mobile"/><br><br>
    <input type="submit" value="Display"/>
</form>
</center>
</body>
</html>
```



FiratServlet.java:-

```
import javax.servlet.http.HttpSession;

public class FirstServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public FirstServlet() {
    }

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

        String uname = request.getParameter("uname");
        String uage = request.getParameter("uage");

        HttpSession session = request.getSession();

        session.setAttribute("uname", uname);
        session.setAttribute("uage", uage);

        request.getRequestDispatcher("form2.html").forward(request, response);
    }
}
```

### SecondServlet.java:-

```
public class SecondServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public SecondServlet() {
        super();
        // TODO Auto-generated constructor stub
    }

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

        String uqual = request.getParameter("uqual");
        String udesig = request.getParameter("udesig");

        HttpSession session = request.getSession();

        session.setAttribute("uqual", uqual);
        session.setAttribute("udesig", udesig);

        request.getRequestDispatcher("form3.html").forward(request, response);
    }
}
```



### DisplayServlet.java:-

```
HttpSession session = request.getSession(false);
PrintWriter out = response.getWriter();
response.setContentType("text/html");

out.println("<html>");
out.println("<body bgcolor='blue'>");
out.println("<center><h1>");
out.println("<br><br>");

out.println("User Name....."+session.getAttribute("uname")+"<br><br>");
out.println("User Age....."+session.getAttribute("uage")+"<br><br>");
out.println("Qualification....."+session.getAttribute("uqual")+"<br><br>");
out.println("Designation....."+session.getAttribute("udesig")+"<br><br>");

out.println("Email....."+request.getParameter("email")+"<br><br>");
out.println("Mobile....."+request.getParameter("mobile")+"<br><br>");

out.println("</h1></center>");
out.println("</body>");
out.println("</html>");
```

