US05CBCA25

Object Oriented Programming – III

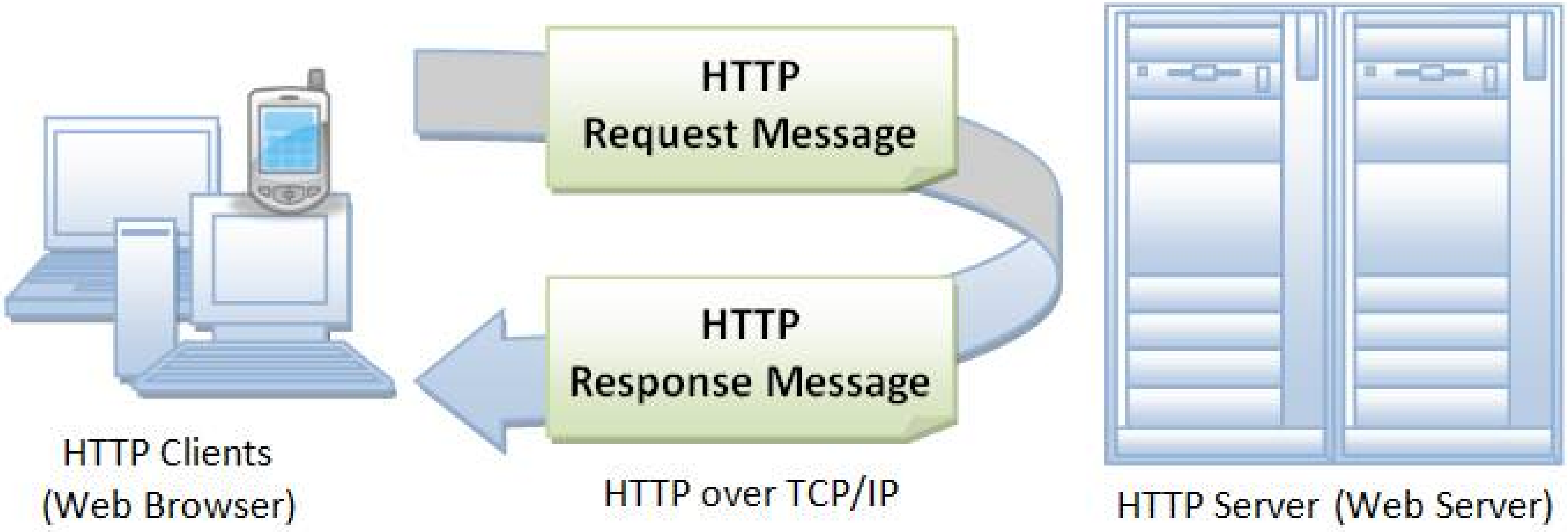
UNIT - II

2.1 Introduction to Servlet

“Servlet is java class
which
extends functionality of web server
by
dynamically generating web pages.”

- Client or web client: It helps to communicate with the server. We can call it browsers (IE, Mozilla, Chrome)
- Server or web server: It is one which holds the web site. It takes requests from client, process it and send response to it. (Apache, Tomcat, IIS)
- HTTP: Server and client both will communicate with each other by predefined rules called HTTP protocol.
- HTTP Request: clients send the request to server by HTTP request.
- HTTP Response: Server will send the response to the client request by HTTP Response.

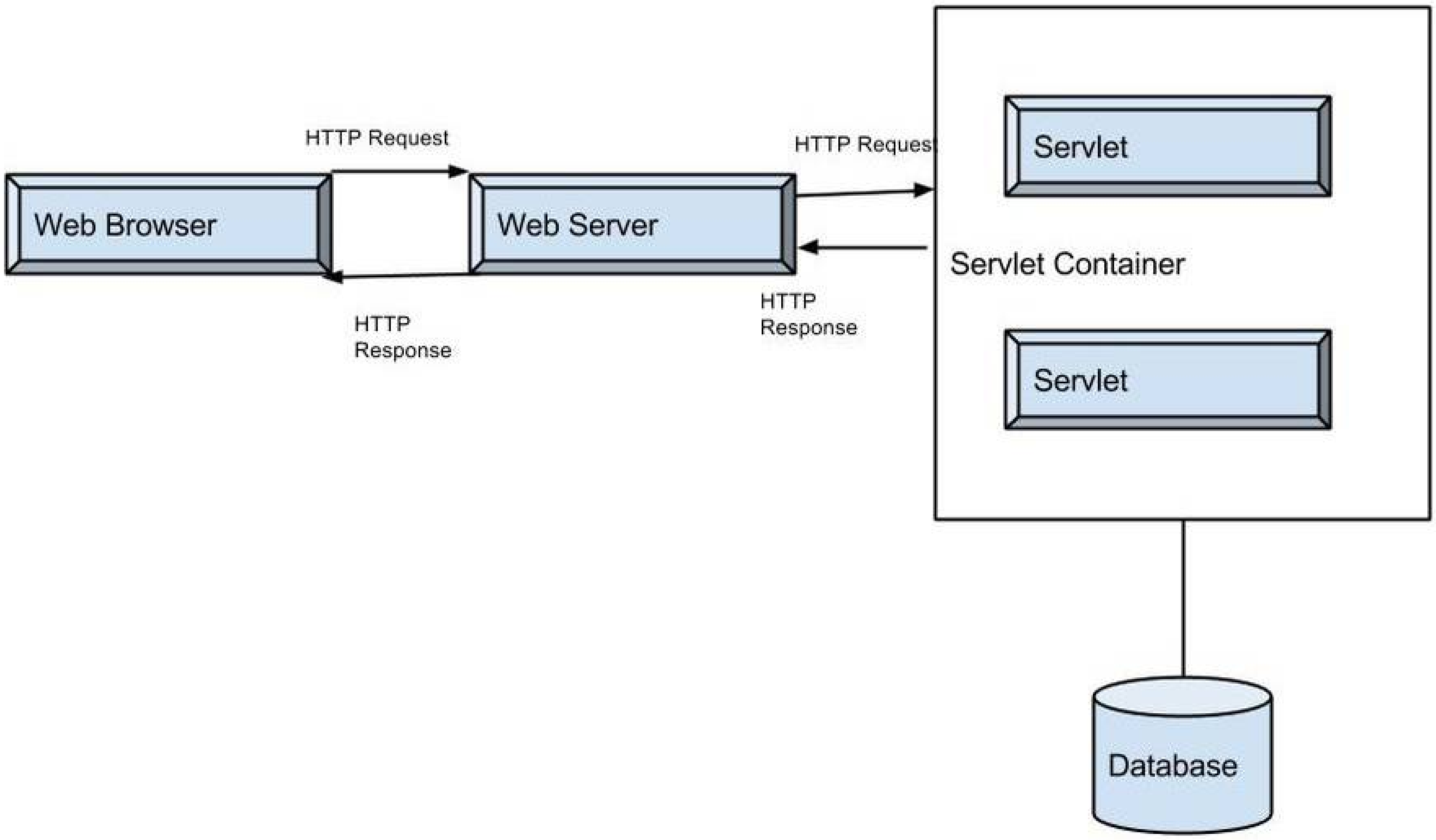
2.1 Introduction to Servlet



2.1 Introduction to Servlet

- Servlet technology is used to create dynamic web applications.
- Servlets provide a component-based, platform-independent method for building Web-based applications.
- Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.
- Servlets can be created using the javax.servlet and javax.servlet.http packages, which are a standard part of the Java's enterprise edition (J2EE)
- Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

2.1 Introduction to Servlet



2.1 Introduction to Servlet

- Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.
- A number of Web Servers that support servlets are available in the market. Some web servers are freely downloadable and Tomcat is one of them.
- Apache Tomcat is an open source software implementation of the Java Servlet and Java Server Pages technologies
- Servlets are the Java programs that runs on the Java-enabled web server or application server. They are used to handle the request obtained from the web server, process the request, produce the response, then send response back to the web server.

<u>Properties of Servlets :</u>

- Servlets work on the server-side.
- Servlets are capable of handling complex requests obtained from web server.

2.1 Advantage of Servlet

<u>Advantages of Servlet</u>

- Performance is significantly better.
- Servlets execute within the address space of a Web server. It is not necessary to create a separate process to handle each client request.
- Servlets are platform-independent because they are written in Java.
- Java security manager on the server enforces a set of restrictions to protect the resources on a server machine. So servlets are trusted.
- The full functionality of the Java class libraries is available to a servlet. It can communicate with applets, databases, or other software via the sockets and RMI mechanisms that you have seen already.
- Java servlets have been created and compiled just like any other Java class. After you install the servlet packages and add them to your computer's Classpath, you can compile servlets with the JDK's Java compiler or any other current compiler.

2.2 Types of Servlet

- Two Types of servlets 1. Generic Servlets and 2. HTTP Servlets

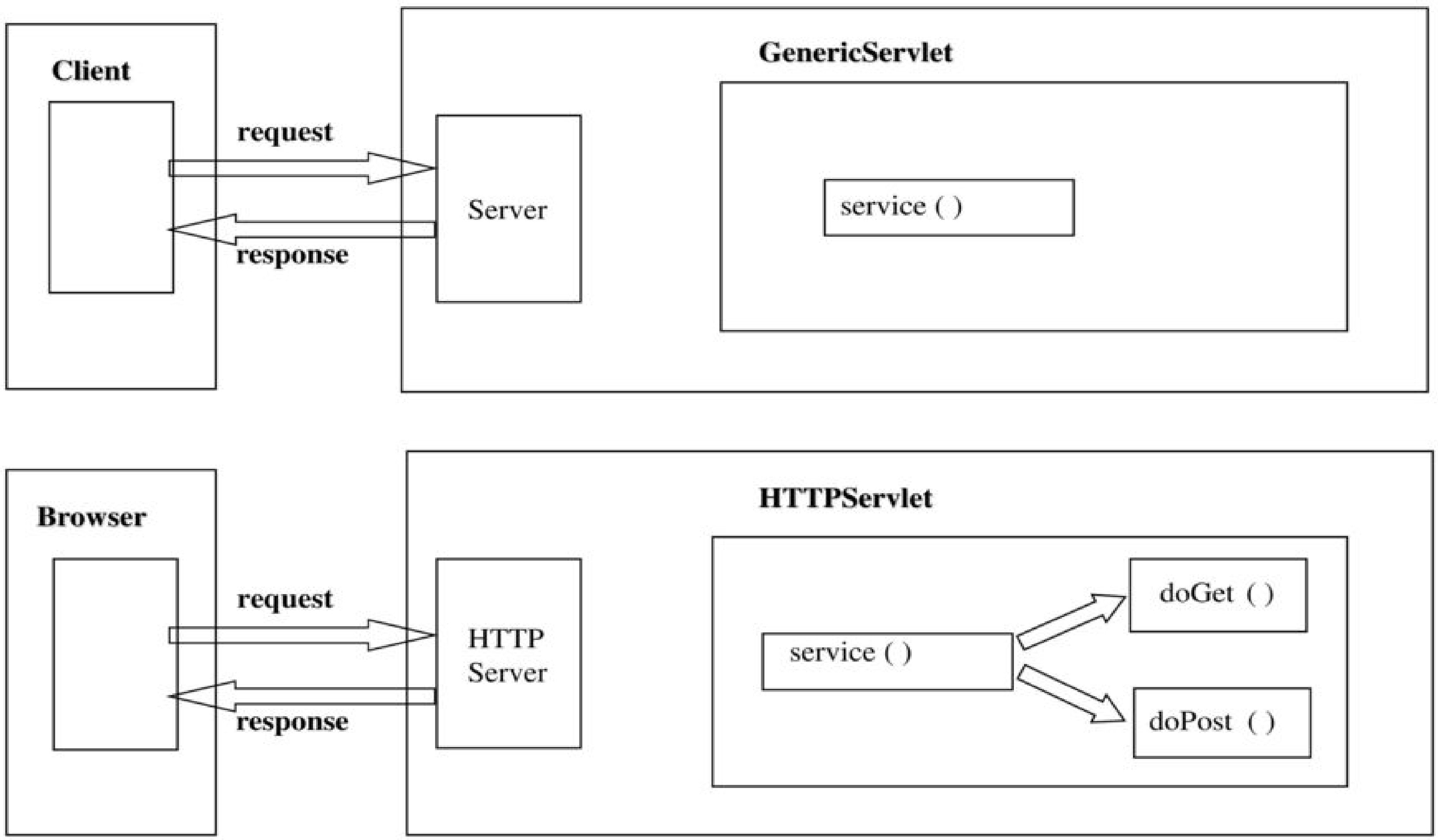
- ***<u>Generic servlets :</u>***
- It is sub classes of javax.servlet.GenericServlet . The service() method needs to be implemented to handle requests.
- Protocol independent. It can handle all types of protocols like http, ftp, smtp.
- Class is direct subclass of servlet interface.
- It is an abstract class which implements servlet, servlet config interfaces.
- Contains service method.
- Implements servlet config and way to accept initialization param passed to servlet from xml. vice.
- Extend javax.servlet.GenericServlet.

2.2 Types of Servlet

- ***<u>HTTP servlets:</u>***
- It is sub classes of javax.servlet.HttpServlet. It has built-in HTTP protocol support. The doGet and doPost methods are used to handle client requests (GET or POST requests).
- Protocol dependent supports only http.
- Direct subclass of generic servlet.
- It is an abstract class which extends generic servlet and implements java.io.serializable
- uses doGet, doPost and doDelete methods
- Public void service and protected void service.
- Extend javax.servlet.HttpServlet.

2.2 Types of Servlet

Generic Servlet vs. HTTP Servlet

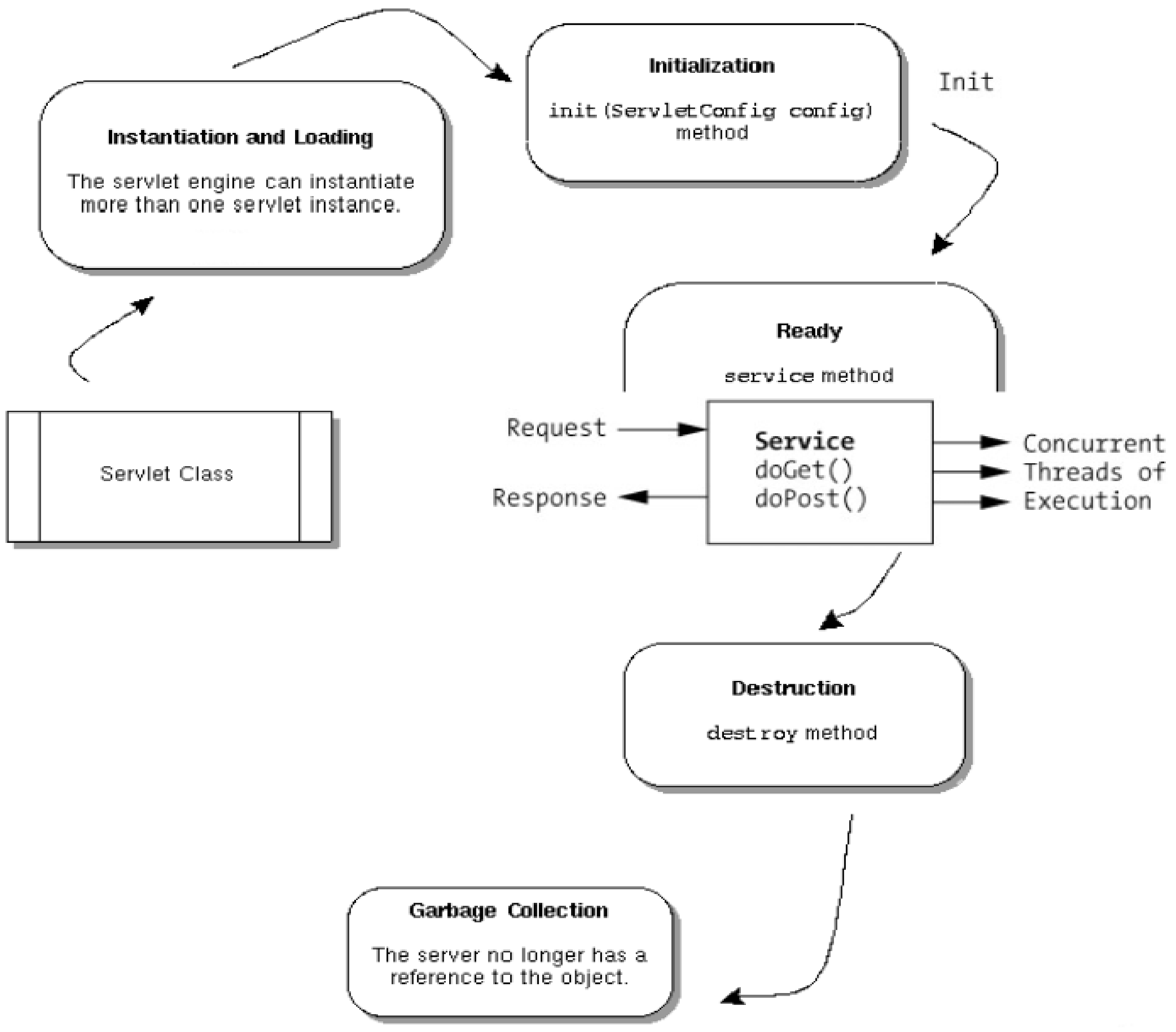


2.3 Servlet Life Cycle

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.

2.3 Servlet Life Cycle


Instantiation and Loading
The servlet engine can instantiate more than one servlet instance.
Initialization
init (ServletConfig config) method
Init
Ready
service method
Request
Response
Service
doGet()
doPost()
Concurrent Threads of Execution
Servlet Class
Destruction
destroy method
Garbage Collection
The server no longer has a reference to the object.

2.3 Servlet Life Cycle

1) Servlet class is loaded

The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.

2) Servlet instance is created

The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

3) init method is invoked

The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

```
public void init(ServletConfig config) throws ServletException
```

2.3 Servlet Life Cycle

4) service method is invoked
The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:
public void service(ServletRequest request, ServletResponse response) **throws** ServletException, IOException

5) destroy method is invoked
The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:
public void destroy()

2.3 Servlet Life Cycle

doGet() vs doPost()

doGet()	doPost()
In this method, parameters are appended to the URL and sent along with header information	In doPost(), parameters are sent in separate line in the body
Maximum size of data that can be sent using doGet() is 240 bytes	There is no maximum size for data
Parameters are not encrypted	Parameters are encrypted here
Application: Used when small amount of insensitive data like a query has to be sent as a request. *It is default method.*	***Application***: Used when comparatively large amount of sensitive data has to be sent. E.g. submitting sign_in or login form.
doGet() is faster comparatively	doPost() is slower compared to doGet() since doPost() does not write the content length

2.4 Servlet Execution

Before Start the servlet, you should have knowledge about the following technology.

- HTML
- XML
- Core JAVA
- SQL

2.4 Servlet Configuration

Lets see how to execute servlet, we have to make our system as web server(because servlet can executes on web server only).

We have to download the “apache tomcat” web server and configure into our system.

Step 1: Download the “apache tomcat” from following link.

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

Step 2: Unzip the folder and copy it into your java directory(suggested).

Step 3: Rename the folder as “tomcat9” (remember that the webapp and other folders can be directly access from the tomcat9 folder)

2.4 Servlet Configuration

Now Lets see that how to configure servlet with netbeans

Step1: Click on “window” menu and go to “services”.

Now right panel displays services. Click on “servers” defualt glassfish server is installed but we want to use tomcat.

Step2: Right click on “servers” and click on “Add server”

Step3: Select “Apache tomcat” from list.

Step4: Click on browse for set “server location” and set the path of “tomcat9”.

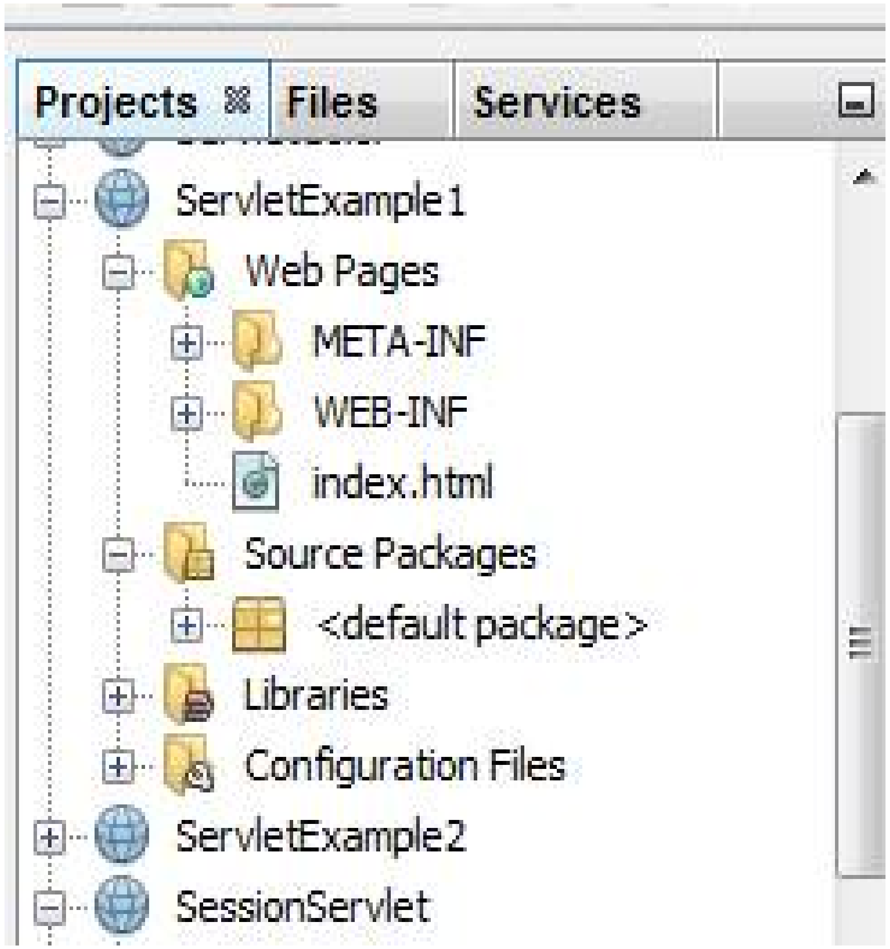
Step5: Set the username and password and click on finish.

2.4 Servlet Execution

Creating servlet based web application

1. Go to File menu -> Select “New Project”
2. Select “Java Web” From categories and “web applications” from project.
3. Select Your path and give the appropriate project name.
4. Select the “Tomcat” as server. If it doesn’t exist you have to add it as I mentioned in previous slide.
5. Click on next and finish the project creation.

2.4 Servlet Execution

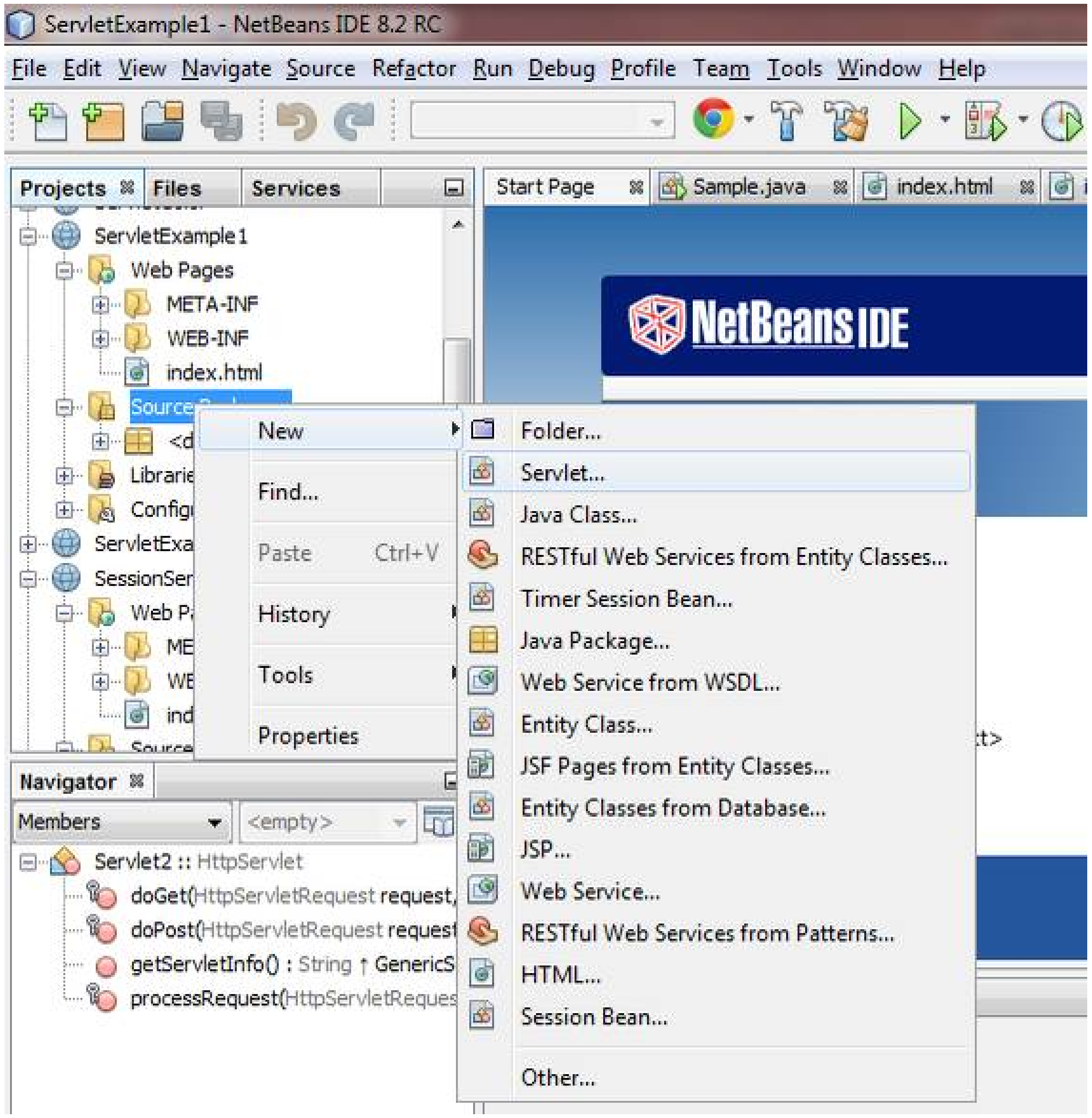


You can see the directory structure that created by netbeans.

ServletExample1 is th project name which includes:

- “Web pages” – all html or jsp file stored here.
  - “WEB-INF” – It includes the web.xml file.(it is automatically create in netbeans)
- “Source Packages” – all java/servlet file stored here.

2.4 Servlet Execution


ServletExample1 - NetBeans IDE 8.2 RC
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
Start Page Sample.java index.html
ServletExample1
Web Pages
META-INF
WEB-INF
index.html
Source Packages
Libraries
Configuration Files
New
Find...
Paste Ctrl+V
History
Tools
Properties
Folder...
Servlet...
Java Class...
RESTful Web Services from Entity Classes...
Timer Session Bean...
Java Package...
Web Service from WSDL...
Entity Class...
JSF Pages from Entity Classes...
Entity Classes from Database...
JSP...
Web Service...
RESTful Web Services from Patterns...
HTML...
Session Bean...
Other...
Navigator
Members
<empty>
Servlet2 :: HttpServlet
doGet(HttpServletRequest request,
doPost(HttpServletRequest request
getServletInfo() : String ↑ GenericS
processRequest(HttpServletReques
NetBeans IDE

For servlet creation

Right click on "Source Packages" -> new -> servlet.

Give the proper servlet name and do next.

Tick on "Add information to deployment descriptor (web.xml). And finish.

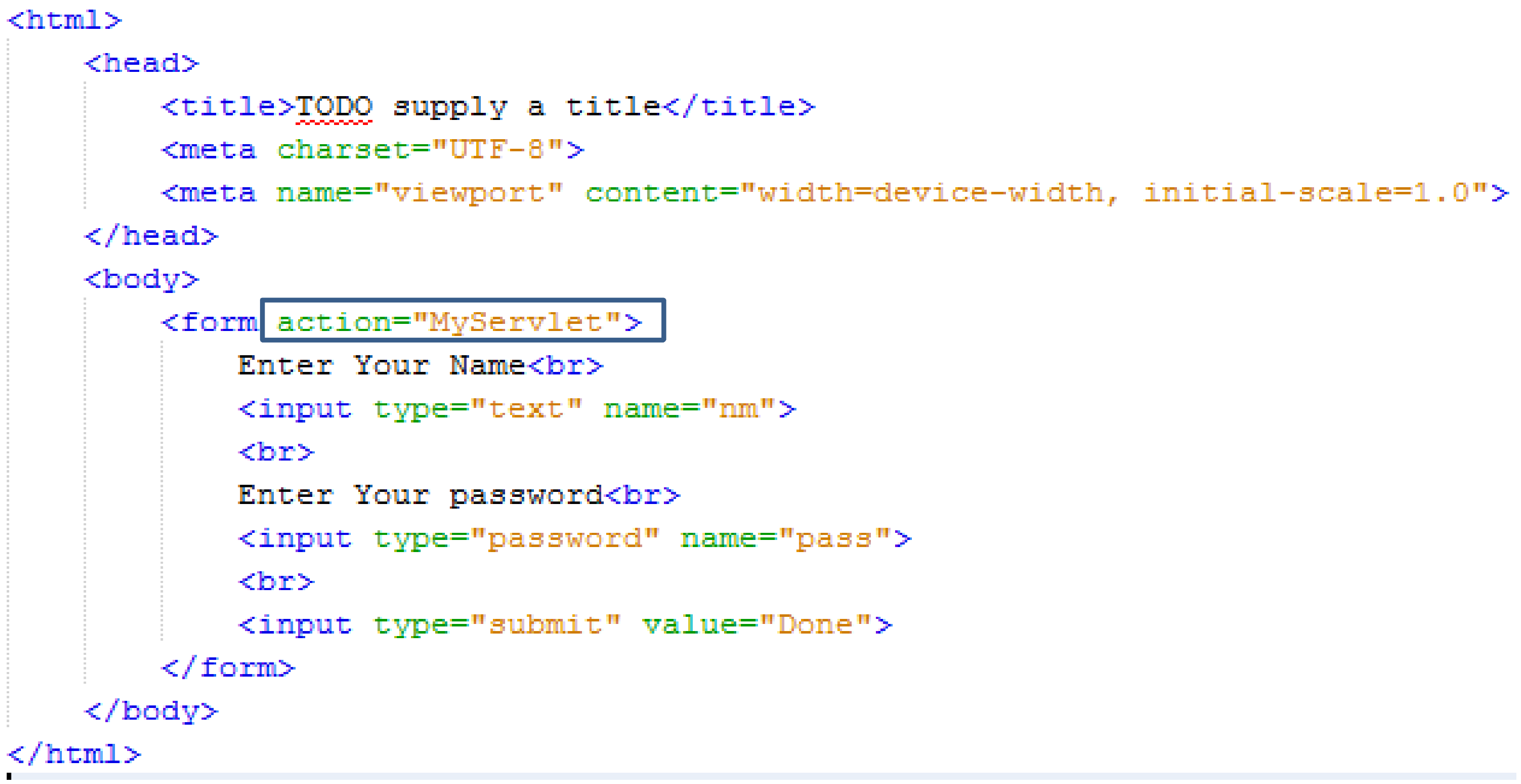
2.4 Servlet Execution Program

In general three files are required…

1. Html file
2. Servlet file
3. Web.xml file

2.4 Servlet Execution Program

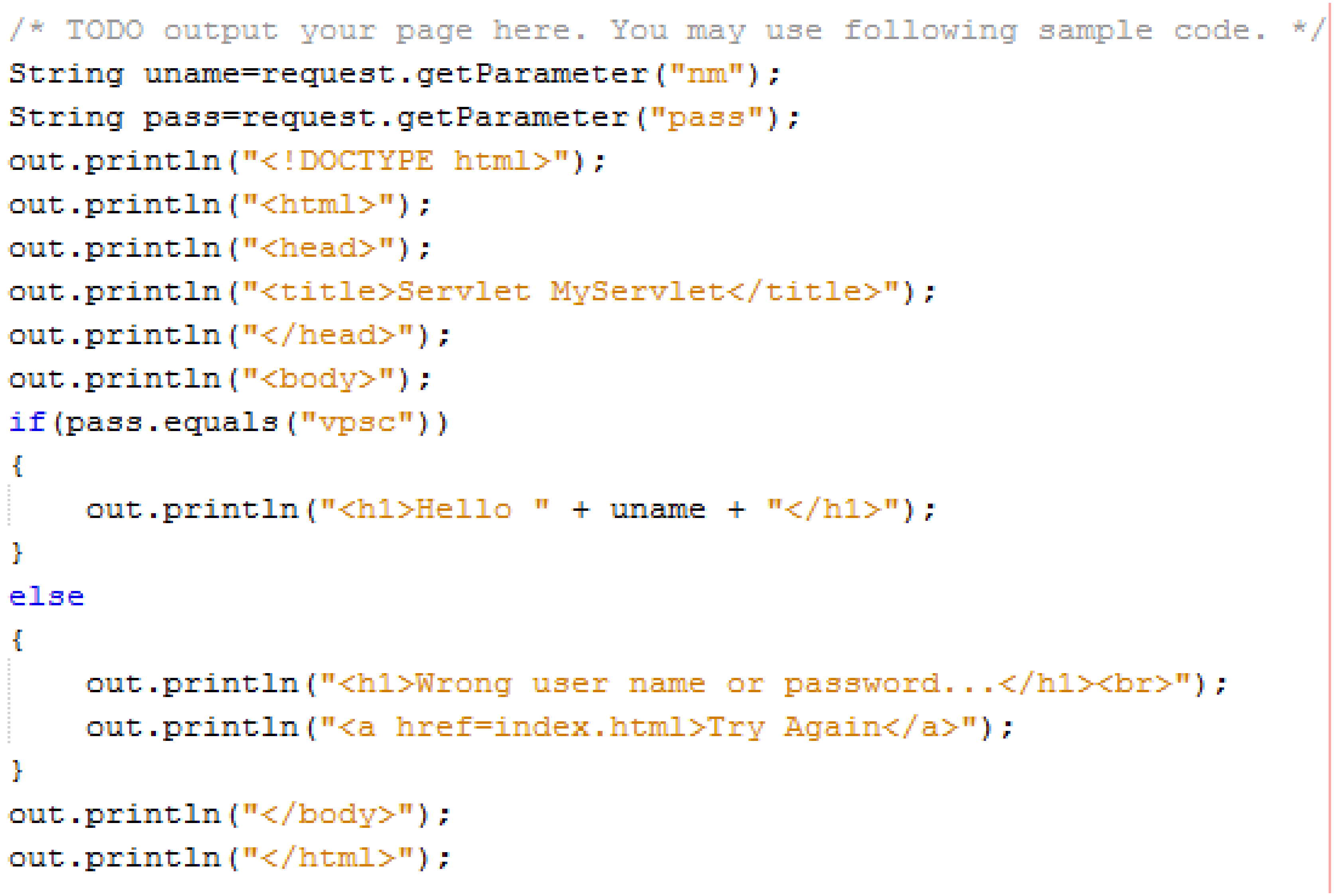
Create the index.html in “Web Pages” directory (netbeans automatically create it) and write the following code.



```
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="MyServlet">
            Enter Your Name<br>
            <input type="text" name="nm">
            <br>
            Enter Your password<br>
            <input type="password" name="pass">
            <br>
            <input type="submit" value="Done">
        </form>
    </body>
</html>
```

2.4 Servlet Execution Program

Create the MySerlvet.java in “Source Packages” directory and write the following code in “processRequest()” method.



```java
/* TODO output your page here. You may use following sample code. */
String uname=request.getParameter("nm");
String pass=request.getParameter("pass");
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet MyServlet</title>");
out.println("</head>");
out.println("<body>");
if(pass.equals("vpsc"))
{
    out.println("<h1>Hello " + uname + "</h1>");
}
else
{
    out.println("<h1>Wrong user name or password...</h1><br>");
    out.println("<a href=index.html>Try Again</a>");
}
out.println("</body>");
out.println("</html>");
```

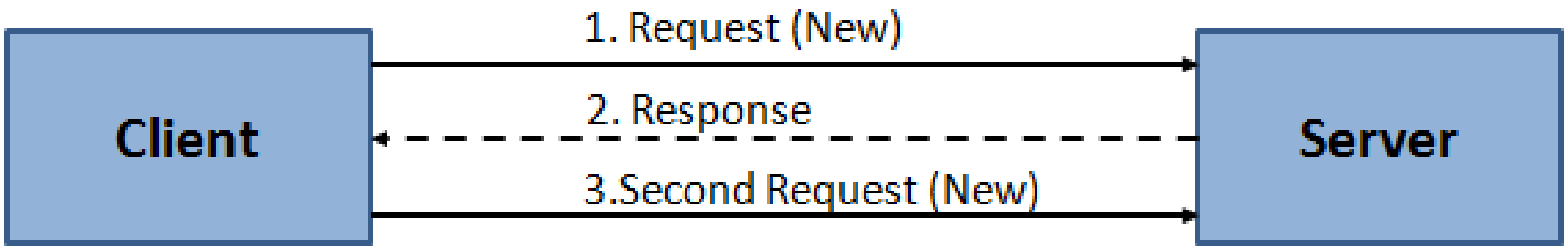
2.4 Servlet Execution Program

1. We created index.html file and call the servlet by clicking the submit button.
2. We created servlet which called by index.html and pass the textbox parameter. It will authenticate the value of text box and display message according to it.
3. Web. Xml created automatically by netbeans because we tick on “add information to deployment descriptor” when create the servlet.
4. Now right click on project and select the “Clean and Build”.
5. Right click on servlet and click on “Compile file”.
6. Right click on “index.html” and click on “run file”.

2.5 Session in Servlet

Why we require Session?

- HTTP is a "stateless" protocol which means each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server automatically does not keep any record of previous client request.
- Session is required to keep track of users and their information.



2.5 Session in Servlet

Example: Application of Session

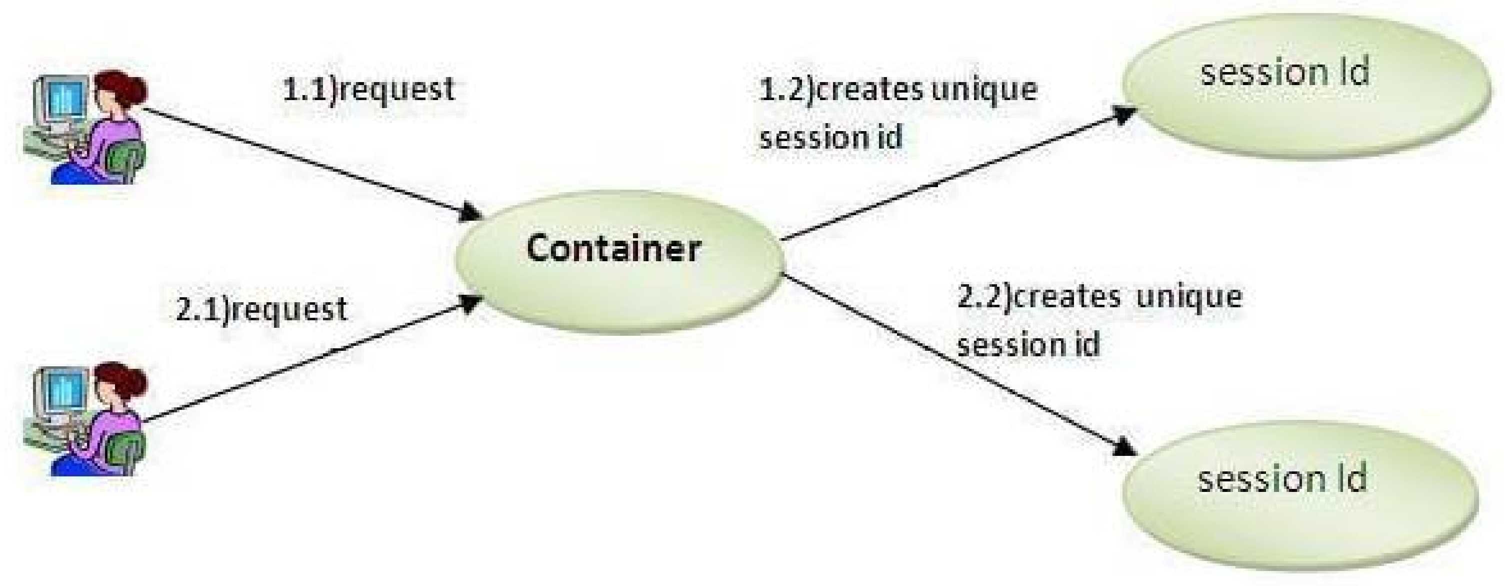
When a User logs into your website, no matter on which web page he visits after ***logging in***, his credentials will be with the server, until user ***logs out***.

So this is managed by creating a session.

2.5 Session in Servlet

In such case, container creates a session id for each user. The container uses this id to identify the particular user. An object of HttpSession can be used to perform two tasks:

1. bind objects
2. view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.



2.5 Session in Servlet

- Package: javax.servlet.http.HttpSession

  Interface

- The servlet container uses this interface to create a session between an HTTP client and an HTTP server.
- In this technique create a session object at server side for each client.
- Session is available until the session time out, until the client log out.
- The default session time is 30 minutes and can configure explicit session time in web.xml file.

2.5 Session in Servlet

How to get the HttpSession object ?

- The HttpServletRequest interface provides two methods to get the object of HttpSession:
- public HttpSession getSession(): Returns the current session associated with this request, or if the request does not have a session, creates one.
- public HttpSession getSession(boolean create): Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

2.5 Session in Servlet

Commonly used methods of HttpSession interface

- public String getId(): Returns a string containing the unique identifier value.
- public long getCreationTime(): Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
- public long getLastAccessedTime(): Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
- public void invalidate(): Invalidates this session then unbinds any objects bound to it.

2.5 Session in Servlet

How to create the session?

```
HttpSession hs=request.getSession();
hs.setAttribute("s_id", "diet054");
```

How to retrieve a session?

```
HttpSession hs=request.getSession(false);
String n=(String)hs.getAttribute("s_id");
```

How to invalidate a session?

```
hs.invalidate();
```

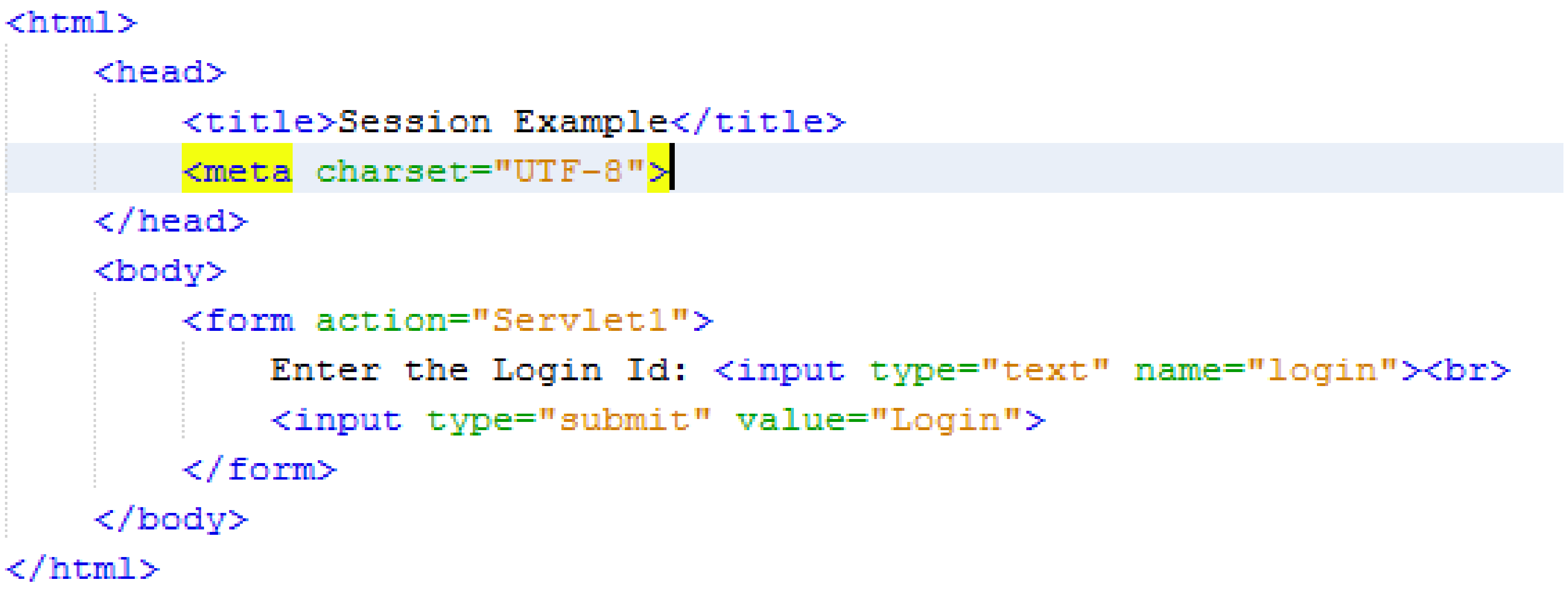
2.5 Session Example

Create following files.

1. index.html (for user inputs)
2. Servlet1.java (for authenticate user and create the session)
3. Servlet2.java (for retrieve the session and destroyed the session when click on log out button)
4. Web.xml(automatically created)

2.5 Session Example

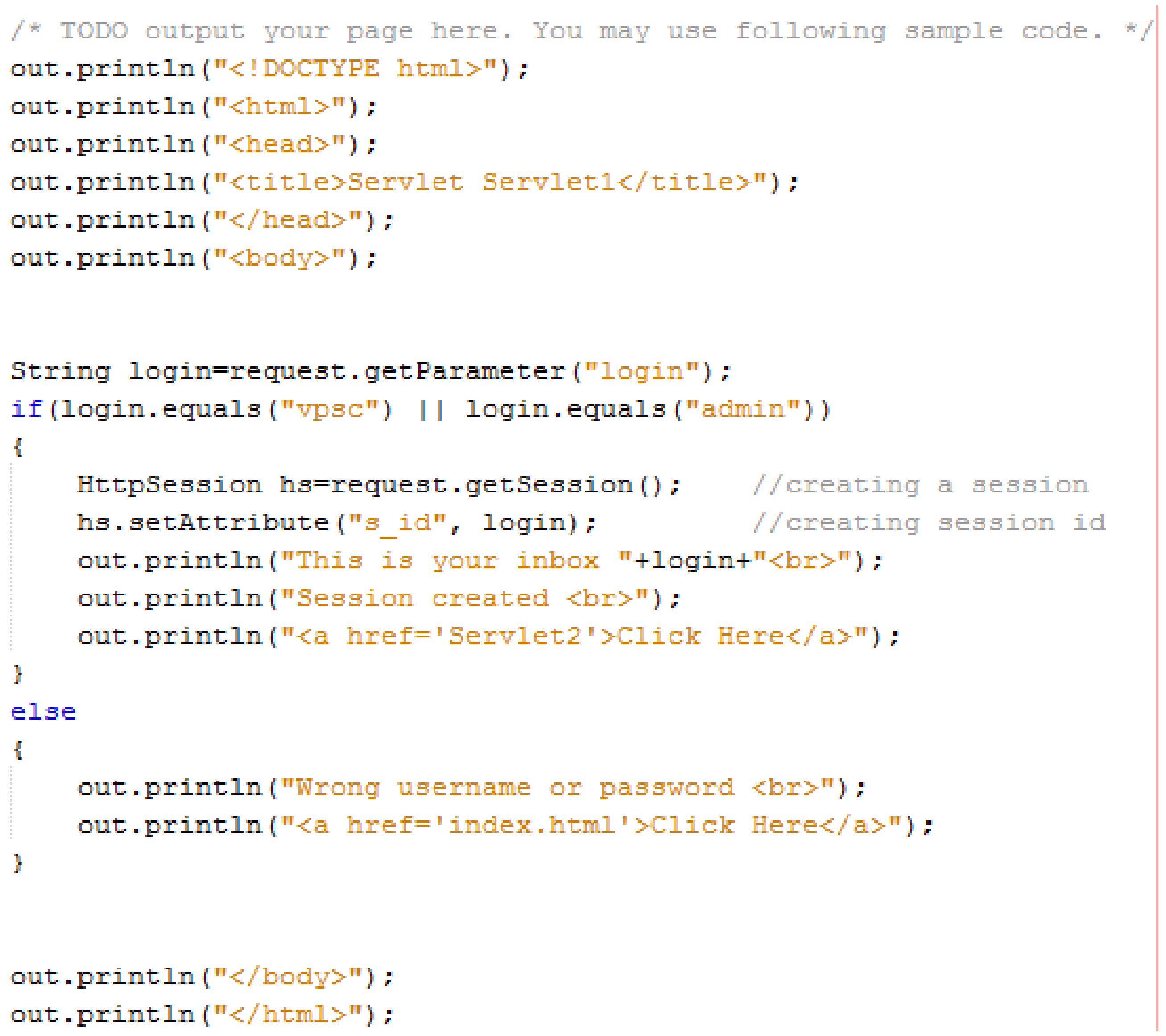
1. index. html



```
<html>
    <head>
        <title>Session Example</title>
        <meta charset="UTF-8">
    </head>
    <body>
        <form action="Servlet1">
            Enter the Login Id: <input type="text" name="login"><br>
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```

2.5 Session Example

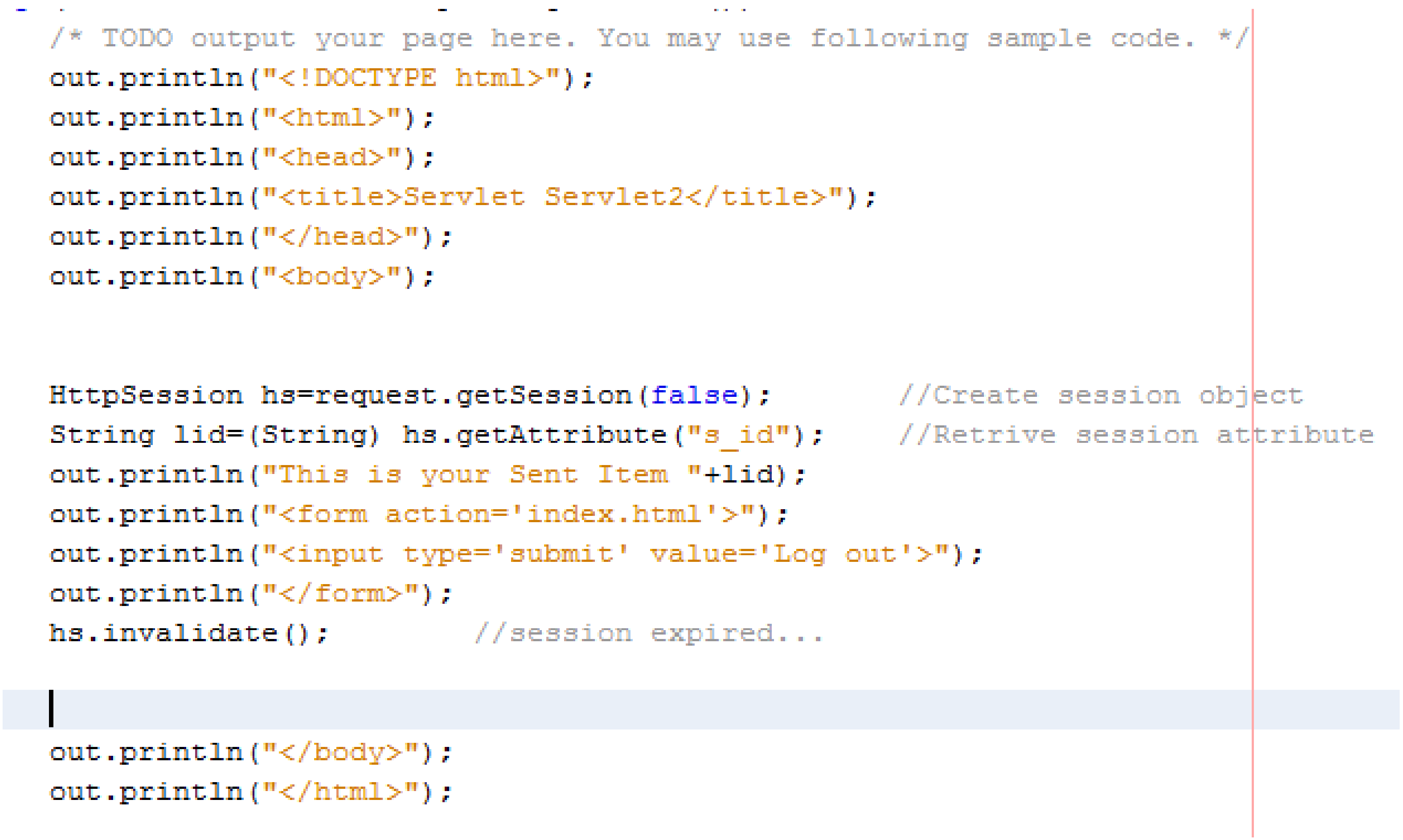
2. Servlet1.java



```
/* TODO output your page here. You may use following sample code. */
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet Servlet1</title>");
out.println("</head>");
out.println("<body>");


String login=request.getParameter("login");
if(login.equals("vpsc") || login.equals("admin"))
{
    HttpSession hs=request.getSession();    //creating a session
    hs.setAttribute("s_id", login);         //creating session id
    out.println("This is your inbox "+login+"<br>");
    out.println("Session created <br>");
    out.println("<a href='Servlet2'>Click Here</a>");
}
else
{
    out.println("Wrong username or password <br>");
    out.println("<a href='index.html'>Click Here</a>");
}


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

2.5 Session Example

3. Servlet2.java



```
/* TODO output your page here. You may use following sample code. */
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet Servlet2</title>");
out.println("</head>");
out.println("<body>");


HttpSession hs=request.getSession(false);       //Create session object
String lid=(String) hs.getAttribute("s_id");    //Retrive session attribute
out.println("This is your Sent Item "+lid);
out.println("<form action='index.html'>");
out.println("<input type='submit' value='Log out'>");
out.println("</form>");
hs.invalidate();         //session expired...


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

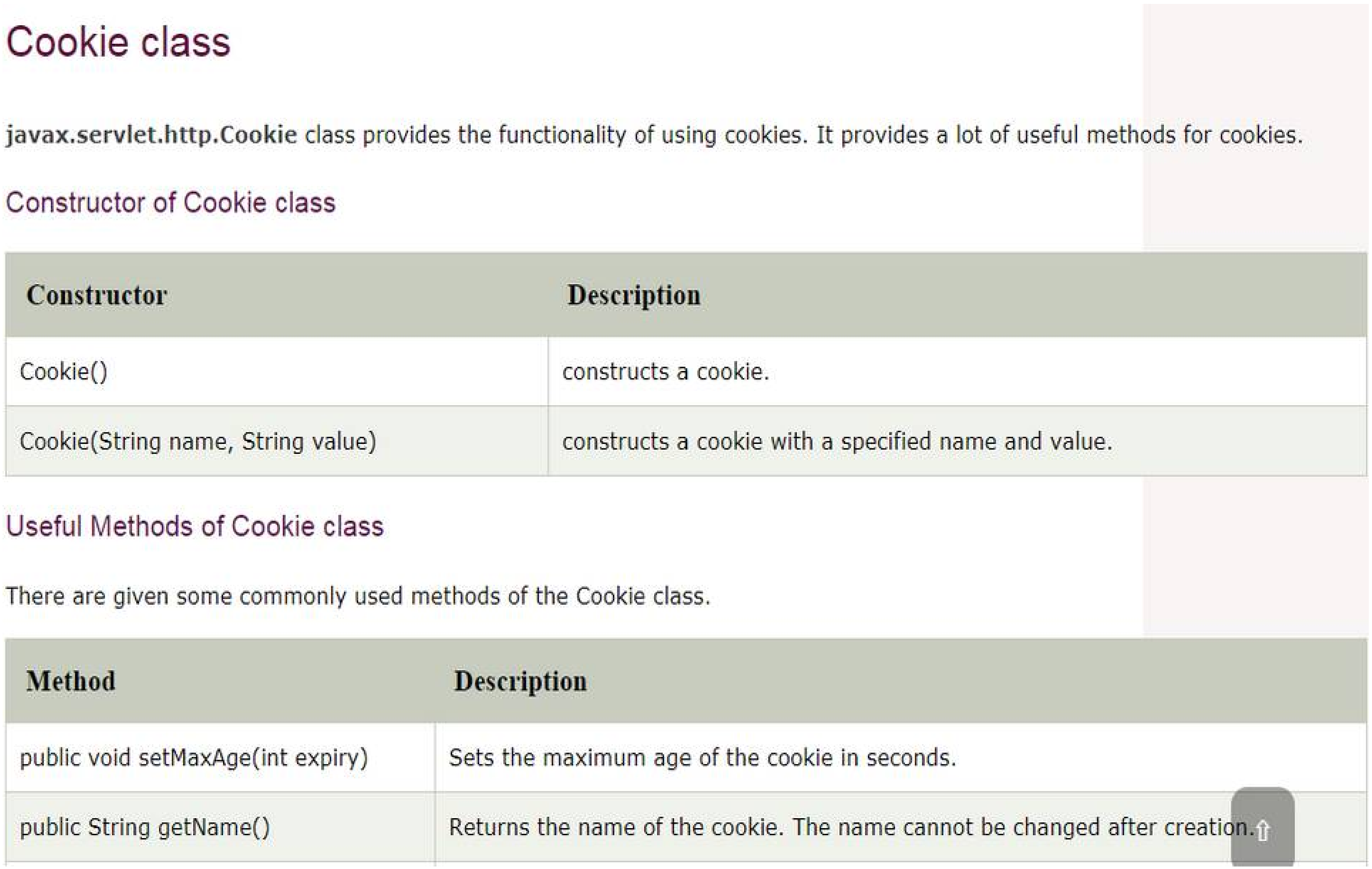
2.5 Servlet Cookies

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
- **Types of Cookie**
  There are 2 types of cookies in servlets.
  - Non-persistent cookie
  - Persistent cookie
- **Non-persistent cookie**
  It is **valid for single session** only. It is removed each time when user closes the browser.
- **Persistent cookie**
  It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

2.4 Servlet Cookies

- **Advantage of Cookies**
  - Simplest technique of maintaining the state.
  - Cookies are maintained at client side.
- **Disadvantage of Cookies**
  - It will not work if cookie is disabled from the browser.
  - Only textual information can be set in Cookie object.

2.4 Servlet Cookies



Cookie class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of Cookie class

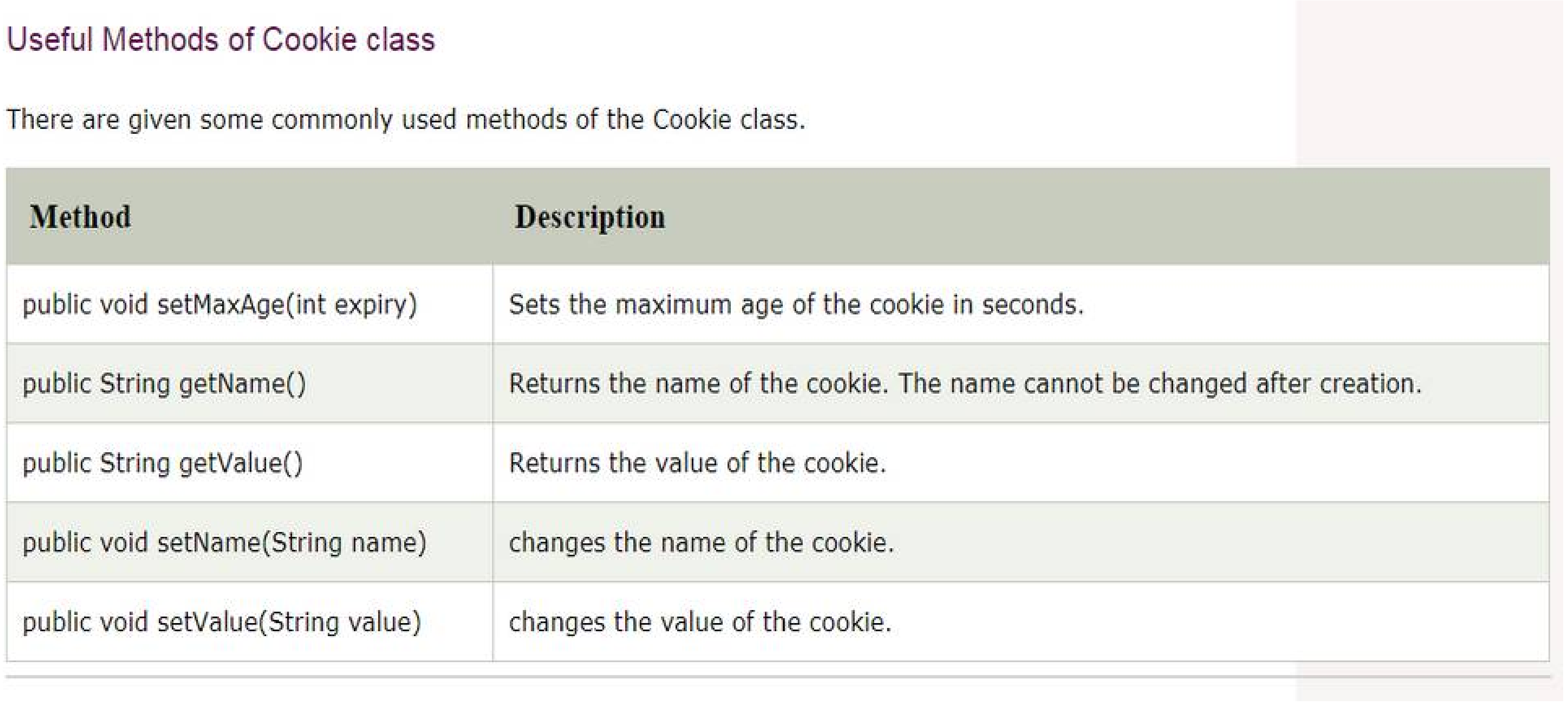
Constructor	Description
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.

2.4 Servlet Cookies

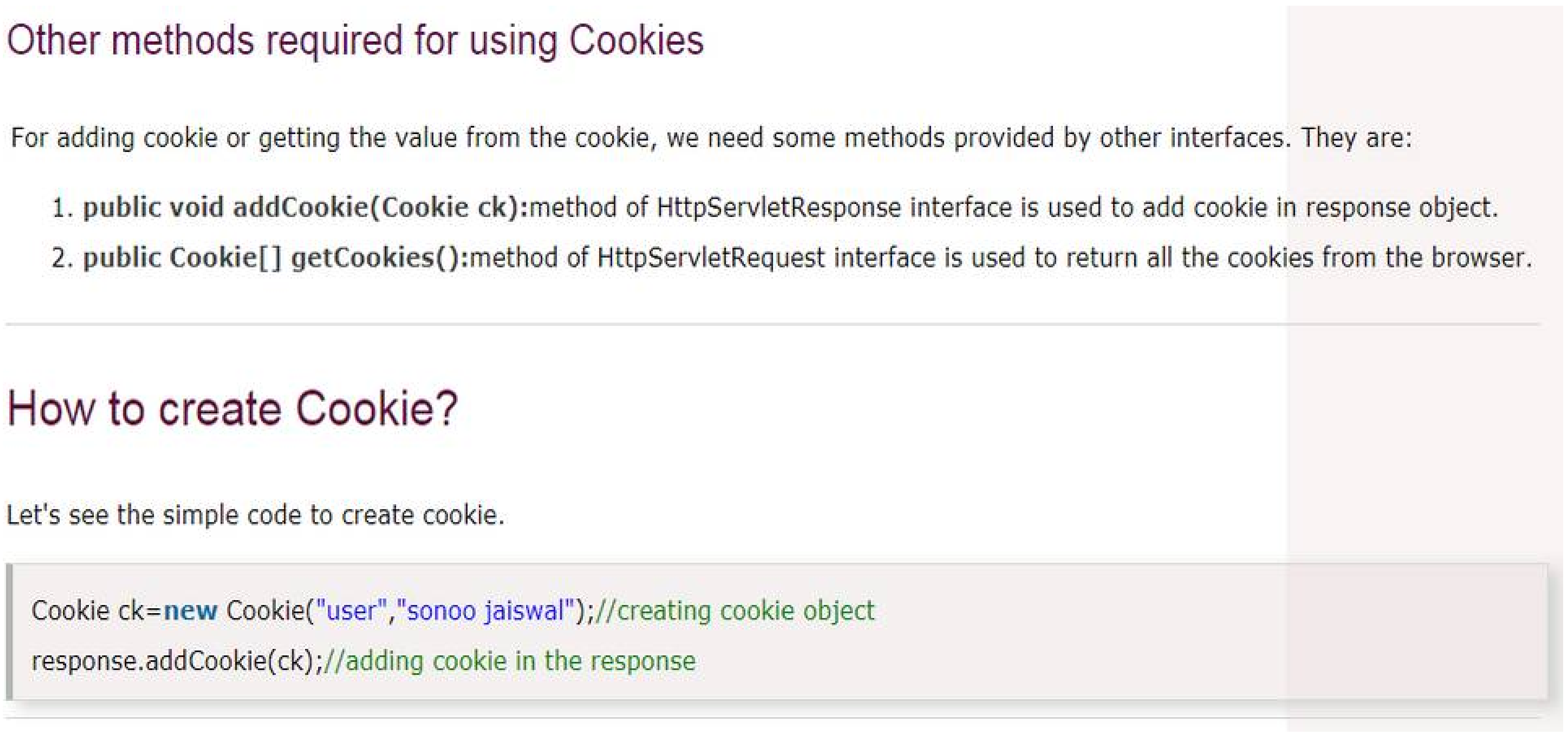


Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

2.4 Servlet Cookies



Other methods required for using Cookies

For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:

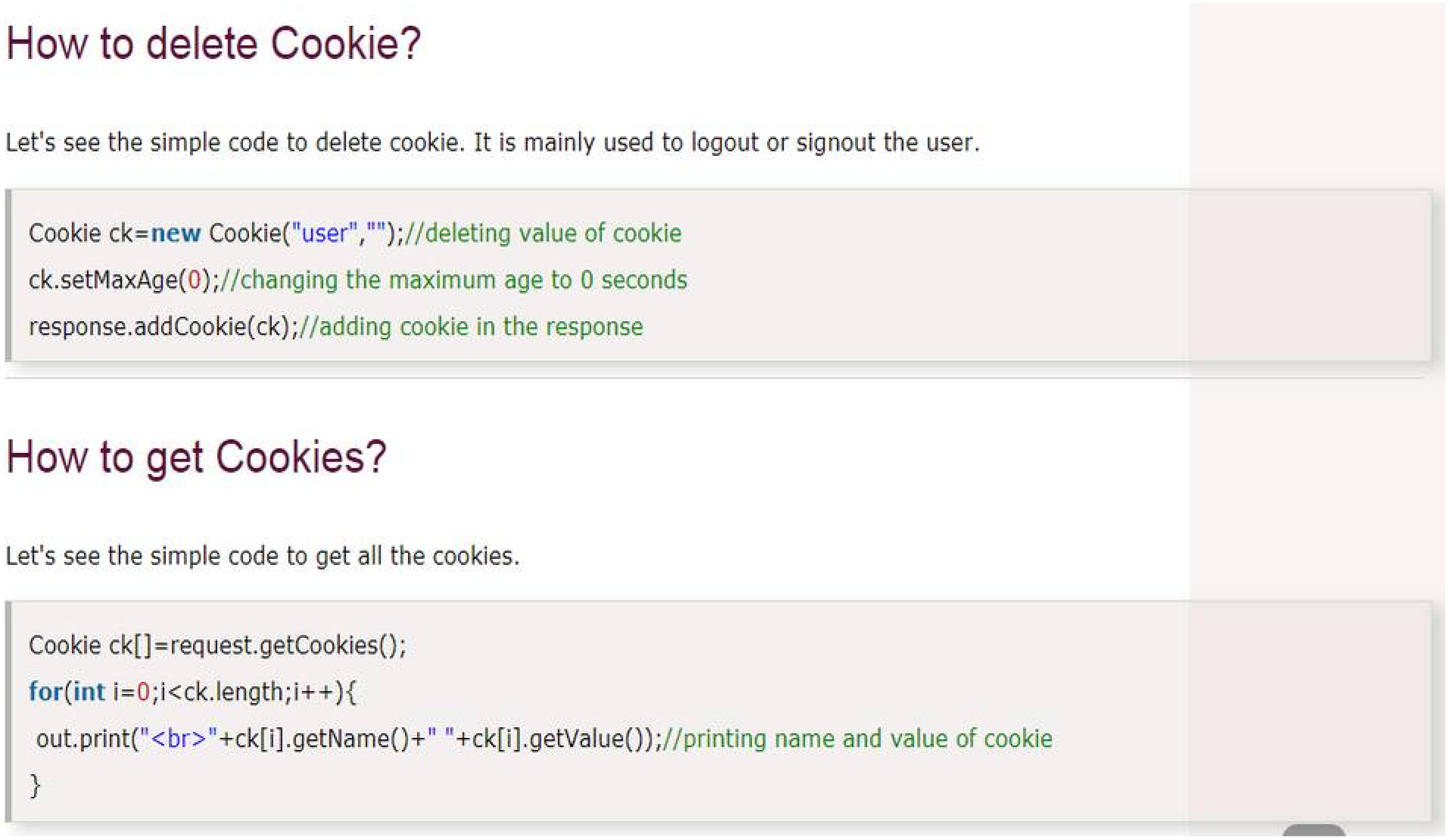
1. **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object.
2. **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser.

How to create Cookie?

Let's see the simple code to create cookie.

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

2.4 Servlet Cookies



How to delete Cookie?

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

```
Cookie ck=new Cookie("user","");//deleting value of cookie
ck.setMaxAge(0);//changing the maximum age to 0 seconds
response.addCookie(ck);//adding cookie in the response
```

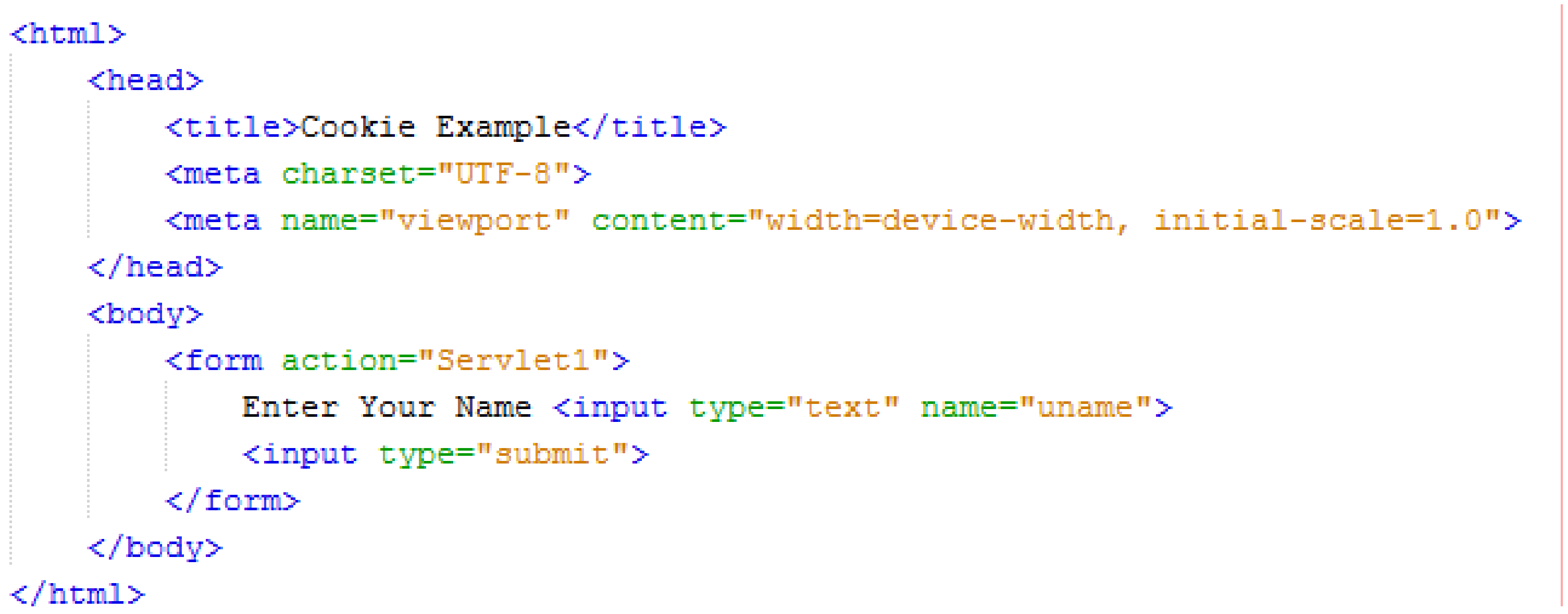
How to get Cookies?

Let's see the simple code to get all the cookies.

```
Cookie ck[]=request.getCookies();
for(int i=0;i<ck.length;i++){
 out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
}
```

2.4 Servlet Cookies Example

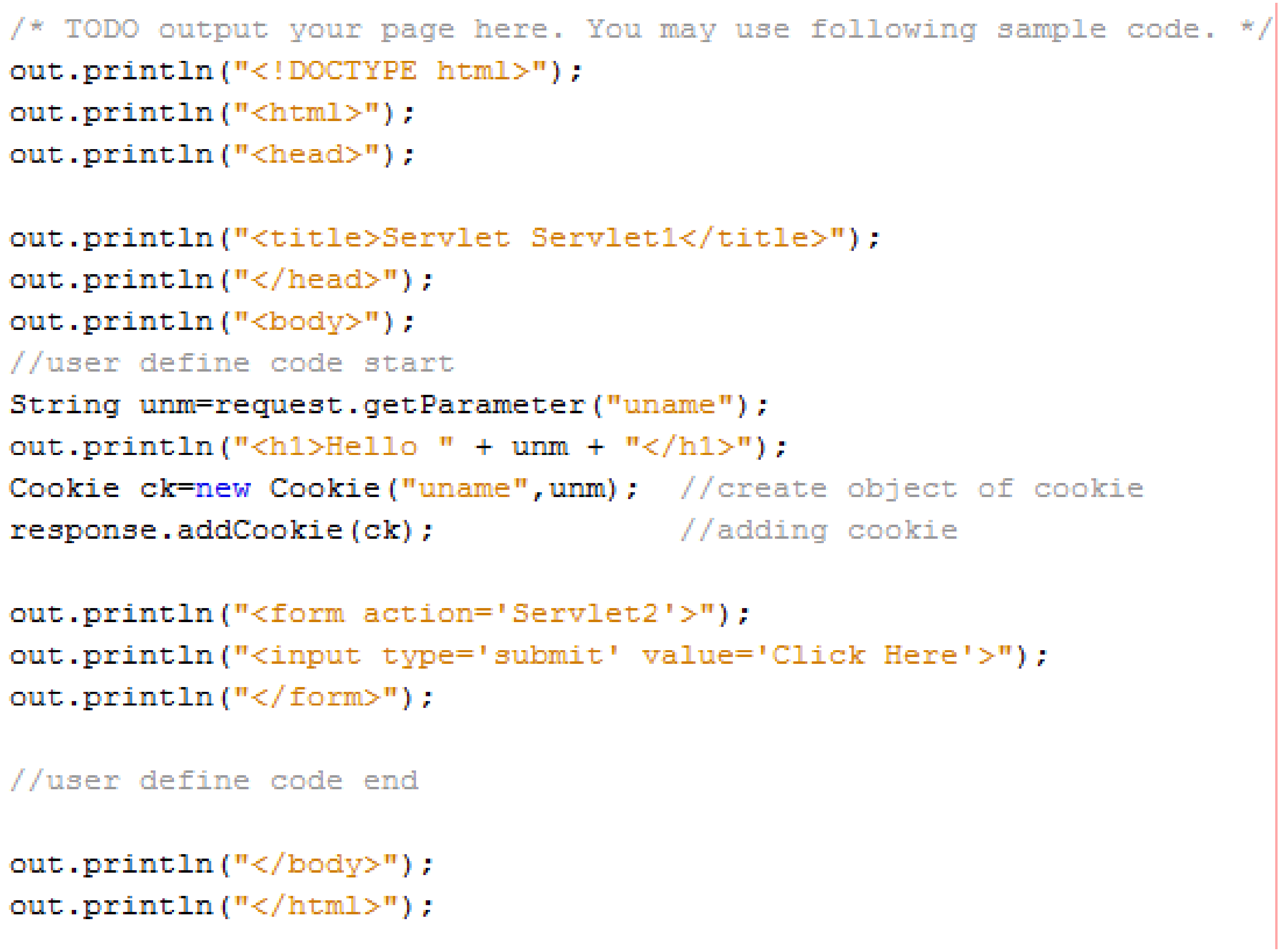
1. index. html



```html
<html>
    <head>
        <title>Cookie Example</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <form action="Servlet1">
            Enter Your Name <input type="text" name="uname">
            <input type="submit">
        </form>
    </body>
</html>
```

2.4 Servlet Cookies Example

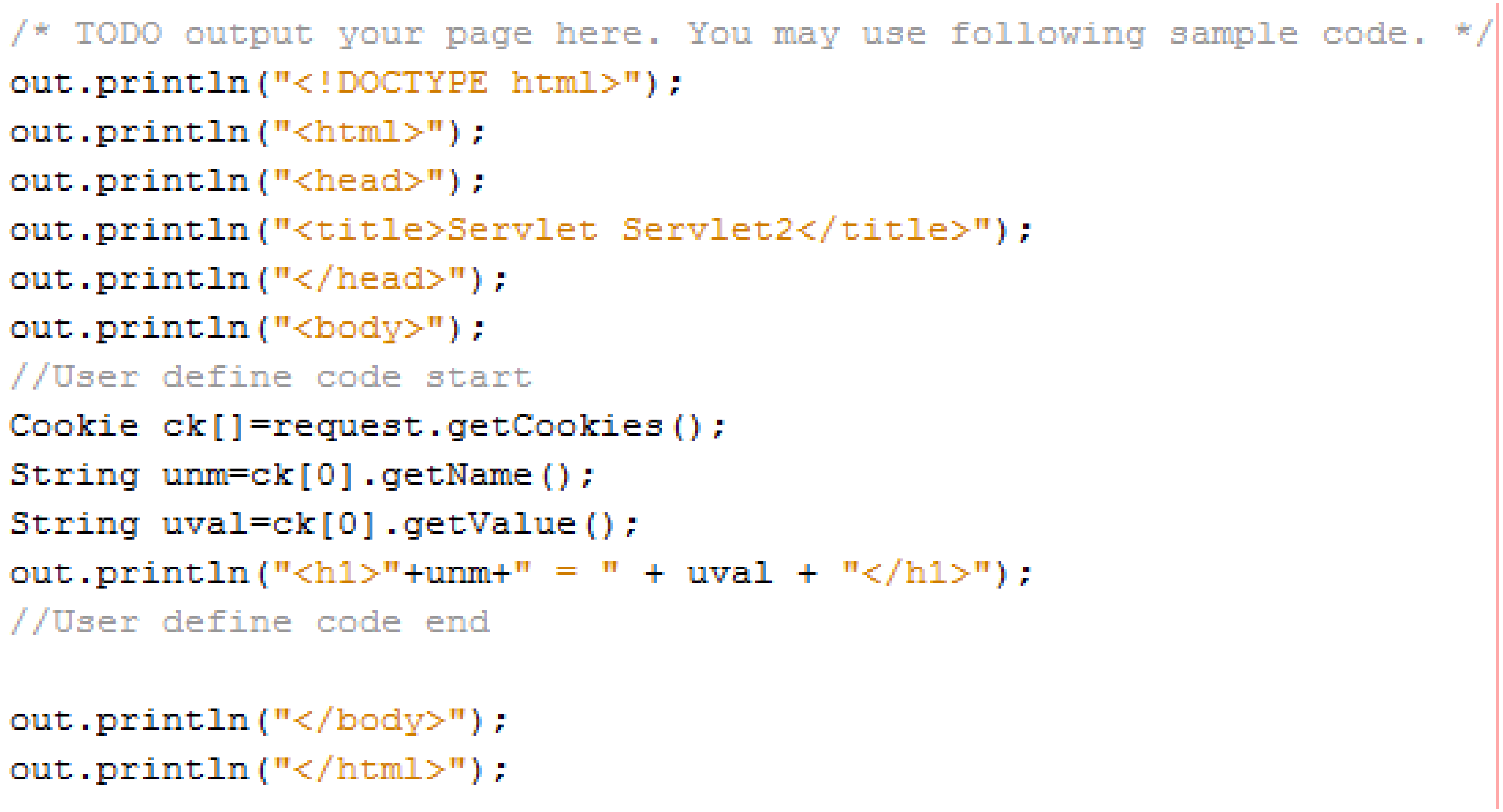
2. Servlet1.java



```
/* TODO output your page here. You may use following sample code. */
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");

out.println("<title>Servlet Servlet1</title>");
out.println("</head>");
out.println("<body>");
//user define code start
String unm=request.getParameter("uname");
out.println("<h1>Hello " + unm + "</h1>");
Cookie ck=new Cookie("uname",unm);  //create object of cookie
response.addCookie(ck);             //adding cookie

out.println("<form action='Servlet2'>");
out.println("<input type='submit' value='Click Here'>");
out.println("</form>");

//user define code end

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

2.4 Servlet Cookies Example

3. Servlet2.java



```
/* TODO output your page here. You may use following sample code. */
out.println("<!DOCTYPE html>");
out.println("<html>");
out.println("<head>");
out.println("<title>Servlet Servlet2</title>");
out.println("</head>");
out.println("<body>");
//User define code start
Cookie ck[]=request.getCookies();
String unm=ck[0].getName();
String uval=ck[0].getValue();
out.println("<h1>"+unm+" = " + uval + "</h1>");
//User define code end

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