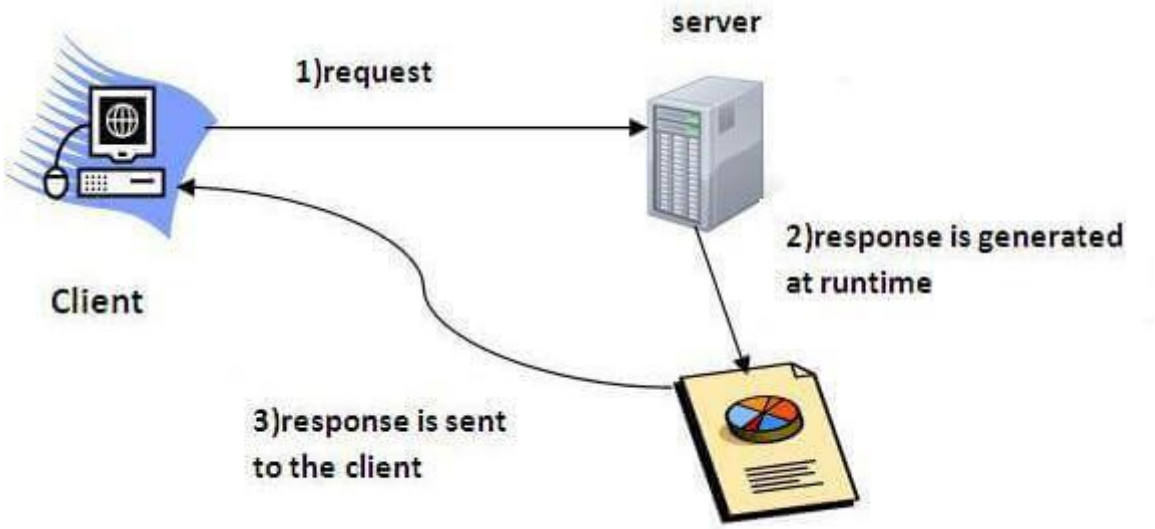
chapter 6

Servlet(14M)

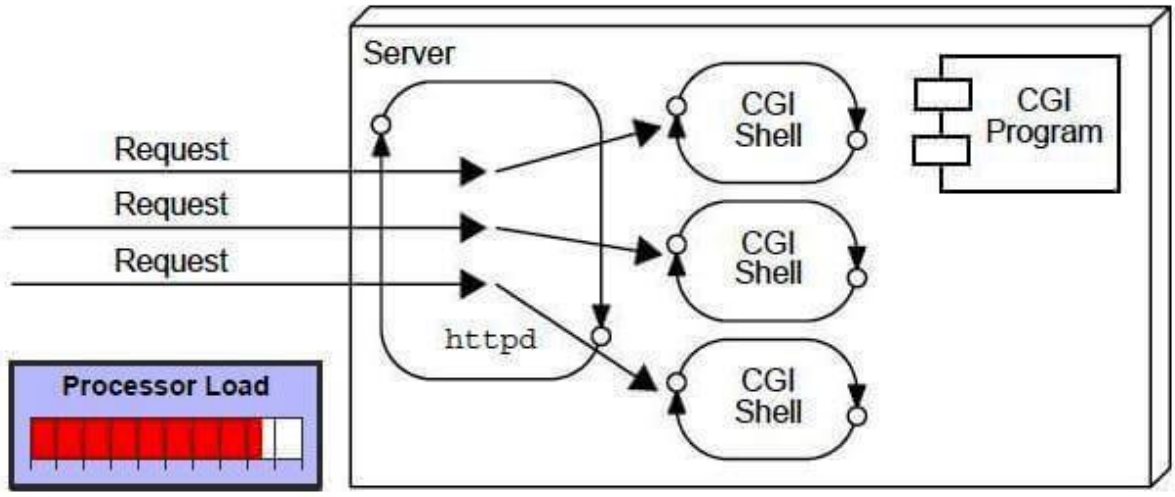
- **Servlets are small programs that execute on the server side** of a Web connection.
- **Applets dynamically extend the functionality of a Web browser**, **servlets dynamically extend the functionality of a Web server**
- Servlet **technology is used to create a web application** (**resides at server side** and generates a dynamic web page).
- Servlet is a web component **that is deployed on the server** to **create a dynamic web page**.



- **Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language.**
- However, **there were many disadvantages** to this technology. We have discussed these disadvantages below.

CGI (Common Gateway Interface)

CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. **For each request, it starts a new process.**

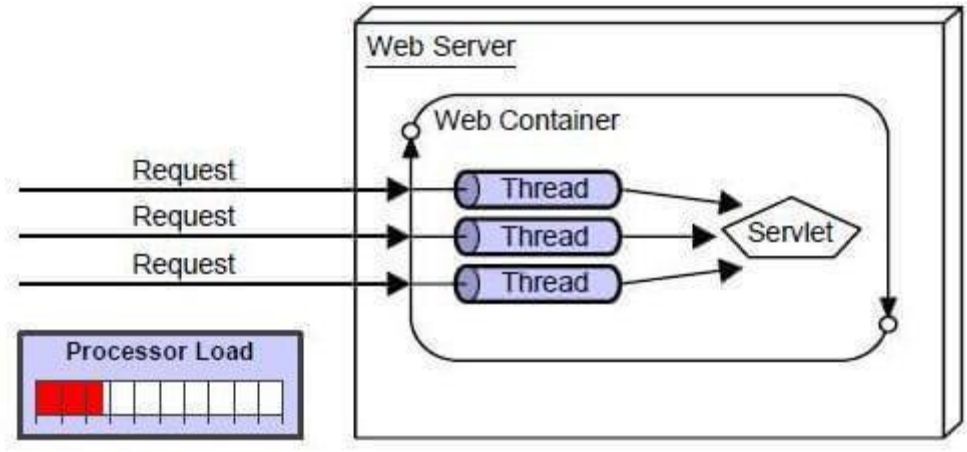


Disadvantages of CGI

There are many problems in CGI technology:

1. If the number of clients increases, **it takes more time** for sending the response.
2. **For each request, it starts a process**, and the web server is limited to start processes.
3. It **uses platform dependent language e.g. C, C++, perl**.

Advantages of Servlet



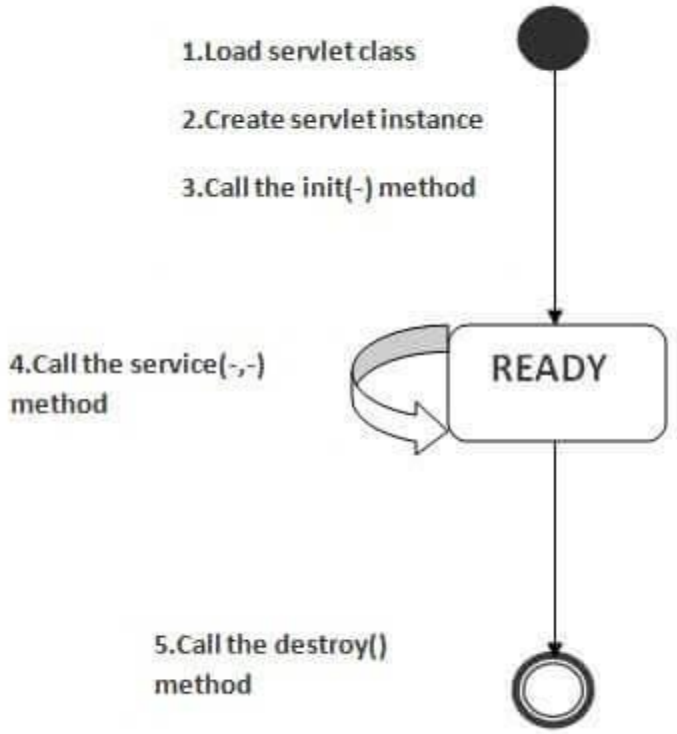
There are many **advantages of Servlet over CGI**. The web container **creates threads** for handling the multiple requests to the Servlet. Threads have many benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The advantages of Servlet are as follows:

1. **Better performance:** because **it creates a thread** for each request, not process.
2. **Portability:** because it **uses Java** language.
3. **Robust: JVM manages Servlets**, so we don't need to worry about the memory leak, garbage collection, etc.
4. **Secure:** because **it uses java** language.

Life Cycle of a Servlet (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.



The Servlet API

Two packages contain the classes and interfaces that are required to build servlets. These are javax.servlet and javax.servlet.http

Interfaces in javax.servlet package

Interface	Description
Servlet	Declares life cycle methods for a servlet.
ServletConfig	Allows servlets to get initialization parameters
ServletContext	Enables servlets to log events and access information about their environment
ServletRequest	Used to read data from a client request

ServletResponse	Used to write data to a client response.
SingleThreadModel	Indicates that the servlet is thread safe

Classes in javax.servlet package

Class	Description
GenericServlet	Implements the Servlet and ServletConfig interfaces.
ServletInputStream	Provides an input stream for reading requests from a client.
ServletOutputStream	Provides an output stream for writing responses to a client.
ServletException	Indicates a servlet error occurred.
UnavailableException	Indicates a servlet is unavailable.

The Servlet Interface

All servlets must implement the Servlet interface. It declares the init( ), service( ), and destroy( ) methods that are called by the server during the life cycle of a servlet.

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet.

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

The ServletConfig Interface

An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.

If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.

Methods of ServletConfig interface

Method	**Description**
ServletContext getServletContext( )	Returns the context for this servlet
String getInitParameter(String param)	Returns the value of the initialization parameter named param.
Enumeration getInitParameterNames( )	Returns an enumeration of all initialization parameter names.
String getServletName( )	Returns the name of the invoking servlet.

The ServletContext Interface

The ServletContext interface is implemented by the server. It enables servlets to obtain information about their environment.

Various Methods Defined by ServletContext Interface

Method	**Description**
Object getAttribute(String attr)	Returns the value of the server attribute named attr.
String getMimeType(String file)	Returns the MIME type of file.
String getRealPath(String vpath)	Returns the real path that corresponds to the virtual path vpath.
String getServerInfo( )	Returns information about the server.
void log(String s)	Writes s to the servlet log.
void log(String s, Throwable e)	Write s and the stack trace for e to the servlet log.
void setAttribute(String attr, Object val)	Sets the attribute specified by attr to the value passed in val.

The ServletRequest Interface

The ServletRequest interface is implemented by the server. It enables a servlet to obtain information about a client request.

Various Methods Defined by ServletRequest Interface

Method	**Description**
Object getAttribute(String attr)	Returns the value of the attribute named attr
String getCharacterEncoding( )	Returns the character encoding of the request.
int getContentLength( )	Returns the size of the request. The value –1 is returned if the size is unavailable
String getContentType( )	Returns the type of the request. A null value is returned if the type cannot be determined.
ServletInputStream getInputStream( ) throws IOException	Returns a ServletInputStream that can be used to read binary data from the request. An IllegalStateException is thrown if getReader( ) has already been invoked for this request.
String getParameter(String pname)	Returns the value of the parameter named pname.
Enumeration getParameterNames( )	Returns an enumeration of the parameter names for this request.
String[ ] getParameterValues(String name)	Returns an array containing values associated with the parameter specified by name.
String getProtocol( )	Returns a description of the protocol.
BufferedReader getReader( ) throws IOException	Returns a buffered reader that can be used to read text from the request. An IllegalStateException is thrown if getInputStream( ) has already been invoked for this request.
String getRemoteAddr( )	Returns the string equivalent of the client IP address.
String getRemoteHost( )	Returns the string equivalent of the client host name.
String getScheme( )	Returns the transmission scheme of the URL used for the request (for example, “http”, “ftp”).
String getServerName( )	Returns the name of the server.
int getServerPort( )	Returns the port number.

The ServletResponse Interface

The ServletResponse interface is implemented by the server. It enables a servlet to formulate a response for a client.

Various Methods Defined by ServletResponse Interface

Method	Description
String getCharacterEncoding( )	Returns the character encoding for the response
ServletOutputStream getOutputStream( ) throws IOException	Returns a ServletOutputStream that can be used to write binary data to the response. An IllegalStateException is thrown if getWriter( ) has already been invoked for this request
PrintWriter getWriter( ) throws IOException	Returns a PrintWriter that can be used to write character data to the response. An IllegalStateException is thrown if getOutputStream( ) has already been invoked for this request.
void setContentLength(int size)	Sets the content length for the response to size
void setContentType(String type)	Sets the content type for the response to type.

The GenericServlet Class

The GenericServlet class provides implementations of the basic life cycle methods for a servlet and is typically subclassed by servlet developers. GenericServlet implements the Servlet and ServletConfig interfaces.

The javax.servlet.http Package

The javax.servlet.http package contains a number of interfaces and classes that are commonly used by servlet developers. You will see that its functionality makes it easy to build servlets that work with HTTP requests and responses. The following table summarizes the core interfaces that are provided in this package:

Interface	Description
HttpServletRequest	Enables servlets to read data from an HTTP request.
HttpServletResponse	Enables servlets to write data to an HTTP response.
HttpSession	Allows session data to be read and written
HttpSessionBindingListener	Informs an object that it is bound to or unbound from a session.

The following table summarizes the core classes that are provided in this package. The most important of these is HttpServlet. Servlet developers typically extend this class in order to process HTTP requests.

Class	**Description**
Cookie	Allows state information to be stored on a client machine
HttpServlet	Provides methods to handle HTTP requests and responses.
HttpSessionEvent	Encapsulates a session-changed event.
HttpSessionBindingEvent	Indicates when a listener is bound to or unbound from a session value, or that a session attribute changed.

The HttpServletRequest Interface

The HttpServletRequest interface is implemented by the server. It enables a servlet to obtain information about a client request.

Method	**Description**
String getAuthType( )	Returns authentication scheme.
Cookie[ ] getCookies( )	Returns an array of the cookies in this request.
long getDateHeader(String field)	Returns the value of the date header field named field.
String getHeader(String field)	Returns the value of the header field named field.
Enumeration getHeaderNames( )	Returns an enumeration of the header names.
int getIntHeader(String field)	Returns the int equivalent of the header field named field.
String getMethod( )	Returns the HTTP method for this request.
String getPathInfo( )	Returns any path information that is located after the servlet path and before a query string of the URL
String getPathTranslated( )	Returns any path information that is located after the servlet path and before a query string of the URL after translating it to a real path.
String getQueryString( )	Returns any query string in the URL
String getRemoteUser( )	Returns the name of the user who issued this request.
String getRequestedSessionId( )	Returns the ID of the session.

String getRequestURI( )	Returns the URI
StringBuffer getRequestURL( )	Returns the URL.
String getServletPath( )	Returns that part of the URL that identifies the servlet.
HttpSession getSession( )	Returns the session for this request. If a session does not exist, one is created and then returned.
HttpSession getSession(boolean new)	If new is true and no session exists, creates and returns a session for this request. Otherwise, returns the existing session for this request.
boolean isRequestedSessionIdFromCookie( )	Returns true if a cookie contains the session ID. Otherwise, returns false.
boolean isRequestedSessionIdFromURL( )	Returns true if the URL contains the session ID. Otherwise, returns false.
boolean isRequestedSessionIdValid( )	Returns true if the requested session ID is valid in the current session context.

The HttpServletResponse Interface

The HttpServletResponse interface is implemented by the server. It enables a servlet to formulate an HTTP response to a client.

Several methods of this interface are summarized in Table.

Method	**Description**
void addCookie(Cookie cookie)	Adds cookie to the HTTP response.
boolean containsHeader(String field)	Returns true if the HTTP response header contains a field named field.
String encodeURL(String url)	Determines if the session ID must be encoded in the URL identified as url. If so, returns the modified version of url. Otherwise, returns url. All URLs generated by a servlet should be processed by this method.
String encodeRedirectURL(String url)	Determines if the session ID must be encoded in the URL identified as url. If so, returns the modified version of url. Otherwise, returns url. All URLs passed to sendRedirect( ) should be processed by this method.
void sendError(int c) throws IOException	Sends the error code c to the client.

void sendError(int c, String s) throws IOException	Sends the error code c and message s to the client.
void sendRedirect(String url) throws IOException	Redirects the client to url.
void setDateHeader(String field, long msec)	Adds field to the header with date value equal to msec (milliseconds since midnight, January 1, 1970, GMT).
void setHeader(String field, String value)	Adds field to the header with value equal to value.
void setIntHeader(String field, int value)	Adds field to the header with value equal to value.
void setStatus(int code)	Sets the status code for this response to code.

The HttpSession Interface

The HttpSession interface is implemented by the server. It enables a servlet to read and write the state information that is associated with an HTTP session.

Several of its methods are summarized in Table.

Method	**Description**
Object getAttribute(String attr)	Returns the value associated with the name passed in attr. Returns null if attr is not found.
Enumeration getAttributeNames( )	Returns an enumeration of the attribute names associated with the session
long getCreationTime( )	Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when this session was created.
String getId( )	Returns the session ID.
long getLastAccessedTime( )	Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the client last made a request for this session
void invalidate( )	Invalidates this session and removes it from the context.
boolean isNew( )	Returns true if the server created the session and it has not yet been accessed by the client.
void removeAttribute(String attr0029	Removes the attribute specified by attr from the session.
void setAttribute(String attr, Object val)	Associates the value passed in val with the attribute name passed in attr.

The Cookie Class

The Cookie class encapsulates a cookie. A cookie is stored on a client and contains state information. Cookies are valuable for tracking user activities. For example, assume that a user visits an online store. A cookie can save the user's name, address, and other information. The user does not need to enter this data each time he or she visits the store.

A servlet can write a cookie to a user's machine via the addCookie( ) method of the HttpServletResponse interface. The data for that cookie is then included in the header of the HTTP response that is sent to the browser.

The names and values of cookies are stored on the user's machine. Some of the information that is saved for each cookie includes the following:

- The name of the cookie
- The value of the cookie
- The expiration date of the cookie
- The domain and path of the cookie

There is one constructor for Cookie. It has the signature shown here:

Cookie(String name, String value)

The methods of the Cookie class are summarized in Table

Method	**Description**
Object clone( )	Returns a copy of this object.
String getComment( )	Returns the comment.
String getDomain( )	Returns the domain.
int getMaxAge( )	Returns the age (in seconds).
String getName( )	Returns the name.
String getPath( )	Returns the path.
boolean getSecure( )	Returns true if the cookie must be sent using only a secure protocol. Otherwise, returns false.
String getValue( )	Returns the value.
int getVersion( )	Returns the cookie protocol version. (Will be 0 or 1.)
void setComment(String c)	Sets the comment to c.
void setDomain(String d)	Sets the domain to d

void setMaxAge(int secs)	Sets the maximum age of the cookie to secs. This is the number of seconds after which the cookie is deleted. Passing –1 causes the cookie to be removed when the browser is terminated.
void setPath(String p)	Sets the path to p.
void setSecure(boolean secure)	Sets the security flag to secure, which means that cookies will be sent only when a secure protocol is being used.
void setValue(String v)	Sets the value to v.
void setVersion(int v)	Sets the cookie protocol version to v, which will be 0 or 1.

The HttpServlet Class

The HttpServlet class extends GenericServlet. It is commonly used when developing servlets that receive and process HTTP requests. The methods of the HttpServlet class are summarized in Table.

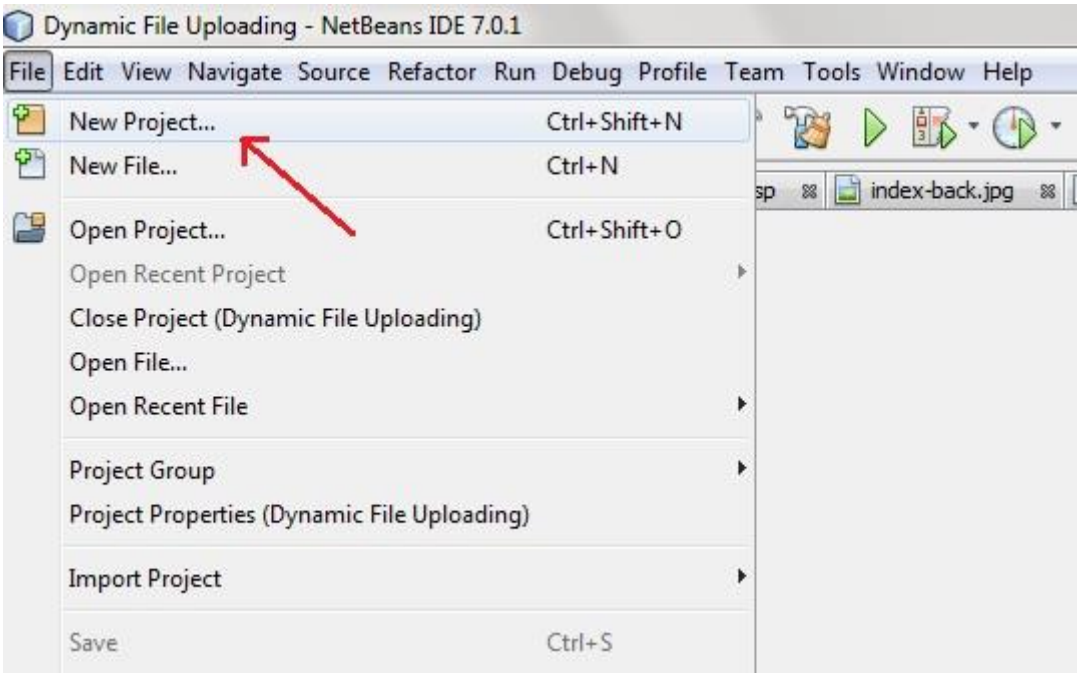
Method	**Description**
void doDelete(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException	Performs an HTTP DELETE.
void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException	Performs an HTTP GET.
void doHead(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException	Performs an HTTP HEAD
void doOptions(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException	Performs an HTTP OPTIONS
void doPost(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException	Performs an HTTP POST.
void doPut(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException	Performs an HTTP PUT.
void doTrace(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException	Performs an HTTP TRACE.

long getLastModified(HttpServletRequest req)	Returns the time (in milliseconds since midnight, January 1, 1970, GMT) when the requested resource was last modified.
void service(HttpServletRequest req, HttpServletResponse res) throws IOException, ServletException	Called by the server when an HTTP request arrives for this servlet. The arguments provide access to the HTTP request and response, respectively.

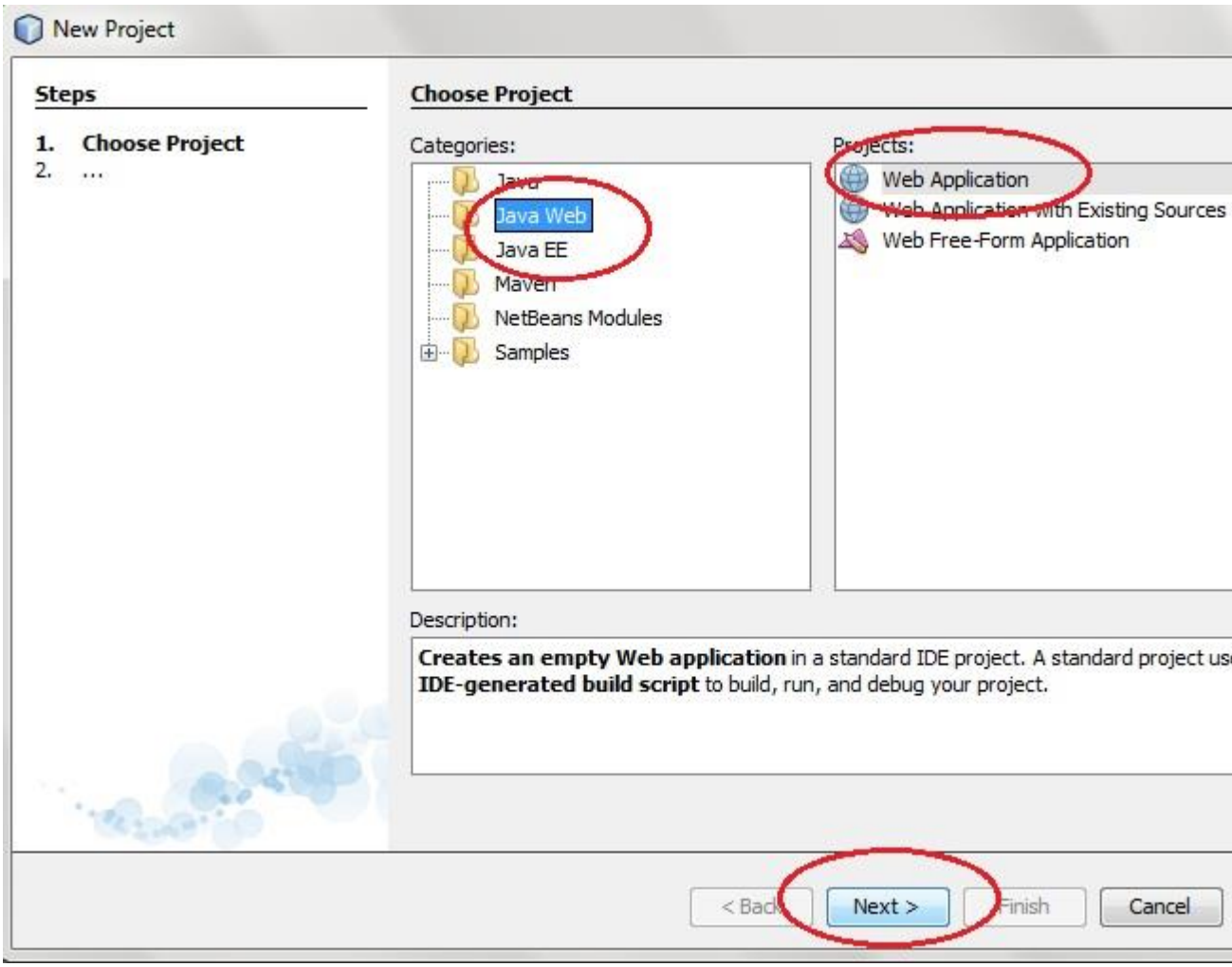
Steps to Create Servlet Application in Netbeans IDE

To create a servlet application in Netbeans IDE, you will need to follow the following (simple) steps :

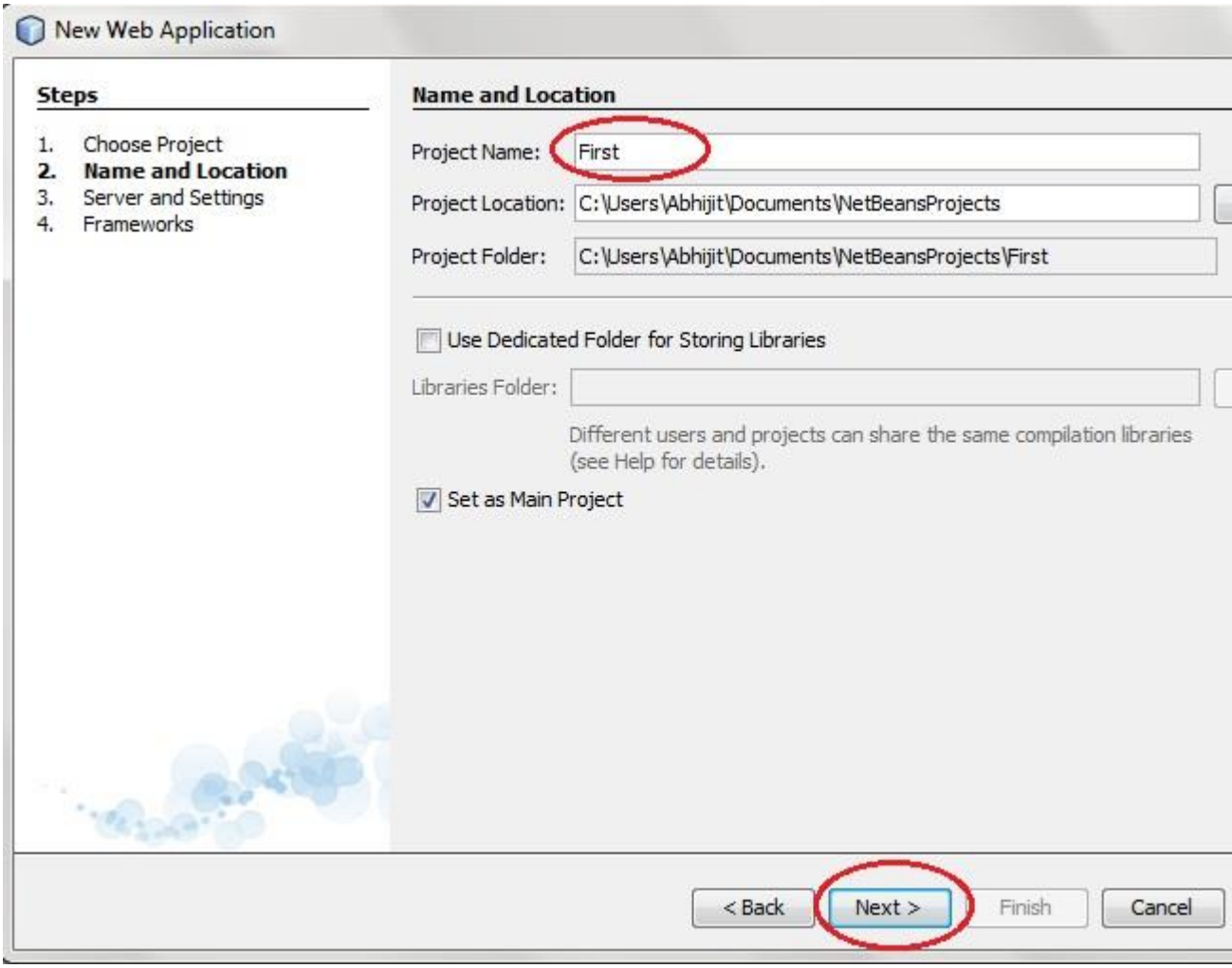
1. Open Netbeans IDE, Select **File** -> **New Project**



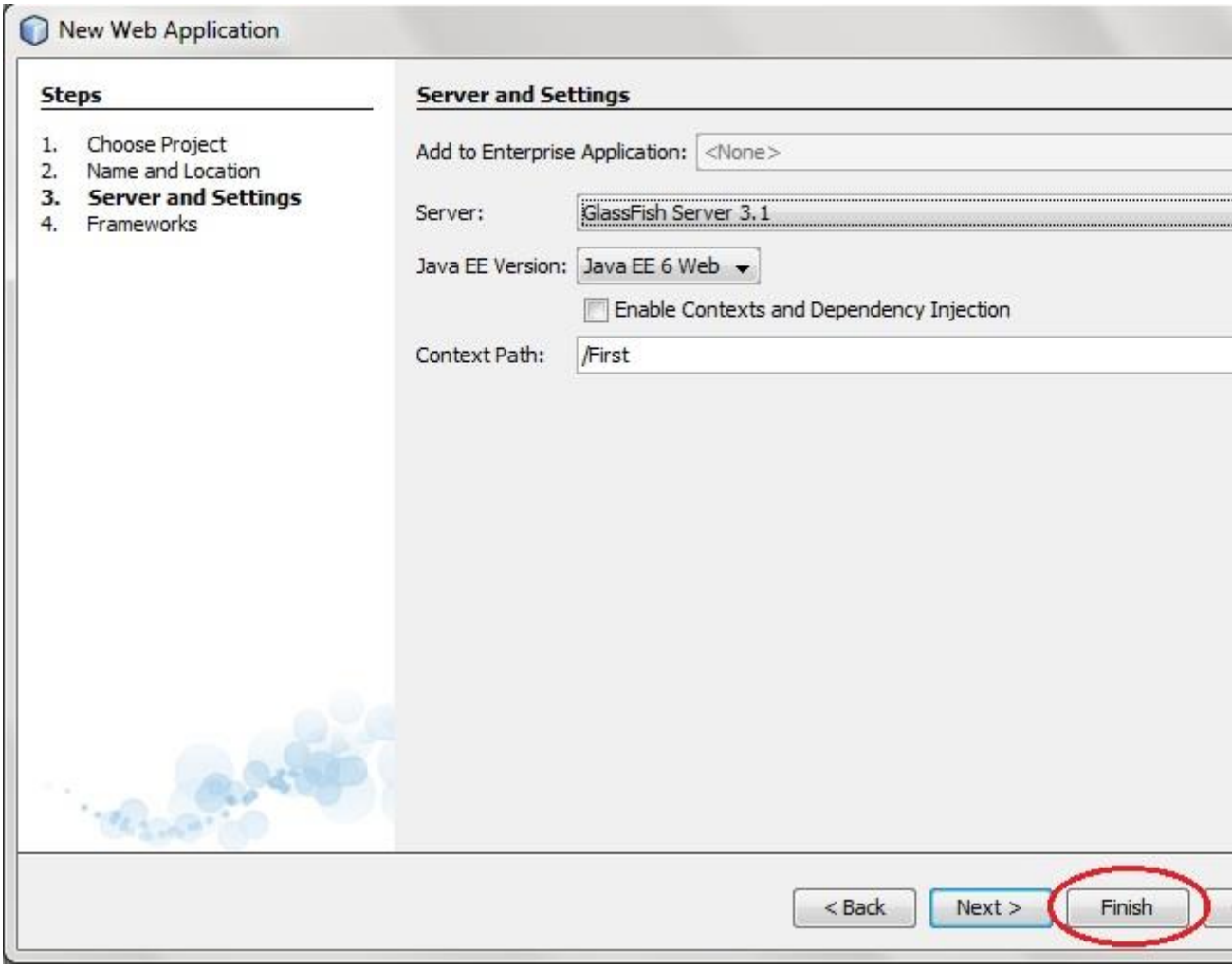
2. Select **Java Web** -> **Web Application**, then click on Next,



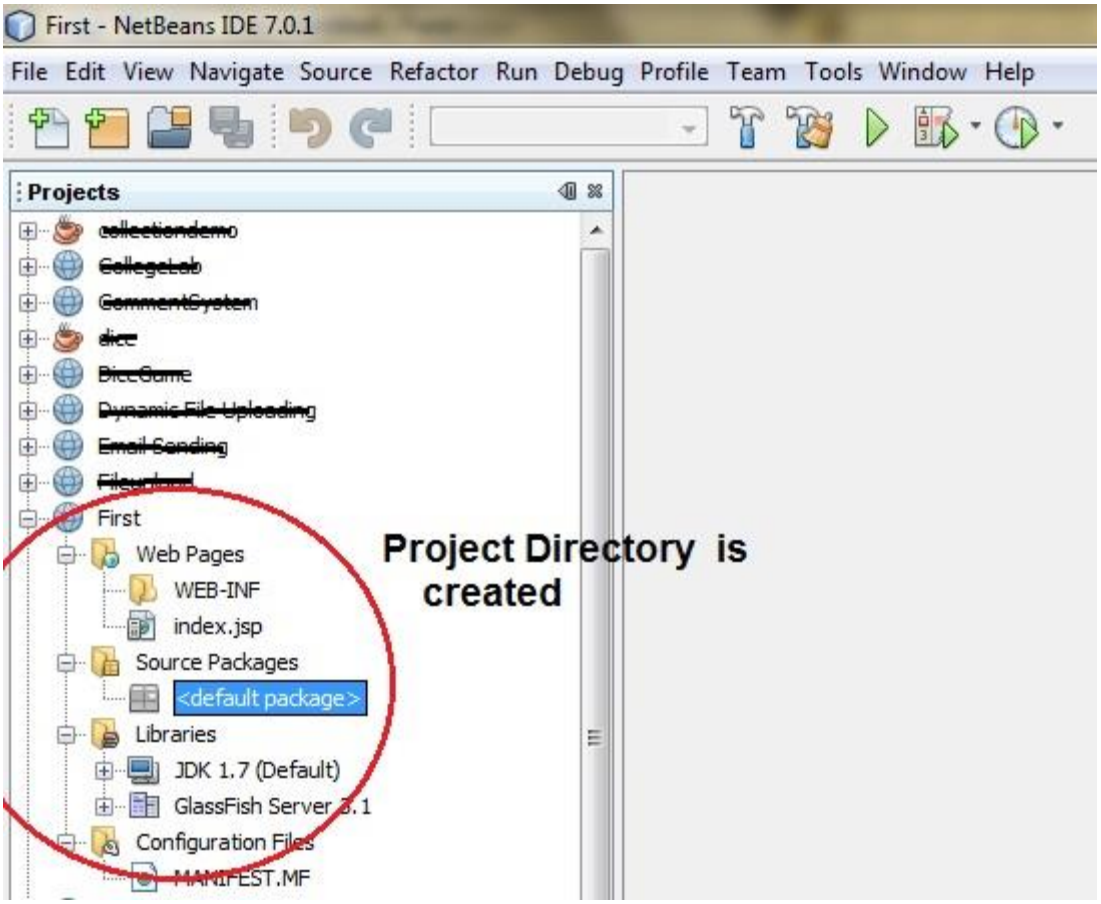
3. Give a name to your project and click on Next,



4. and then, Click **Finish**

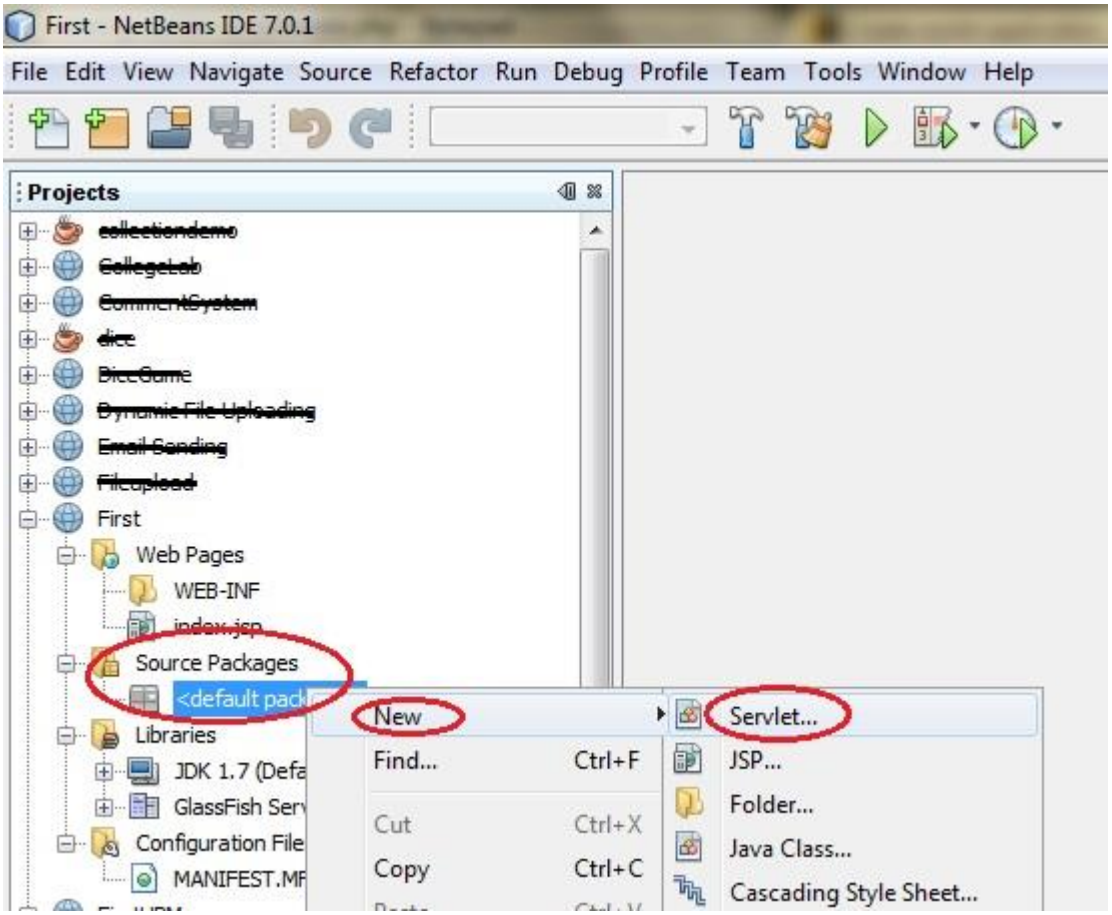


5. The complete directory structure required for the Servlet Application will be created automatically by the IDE.

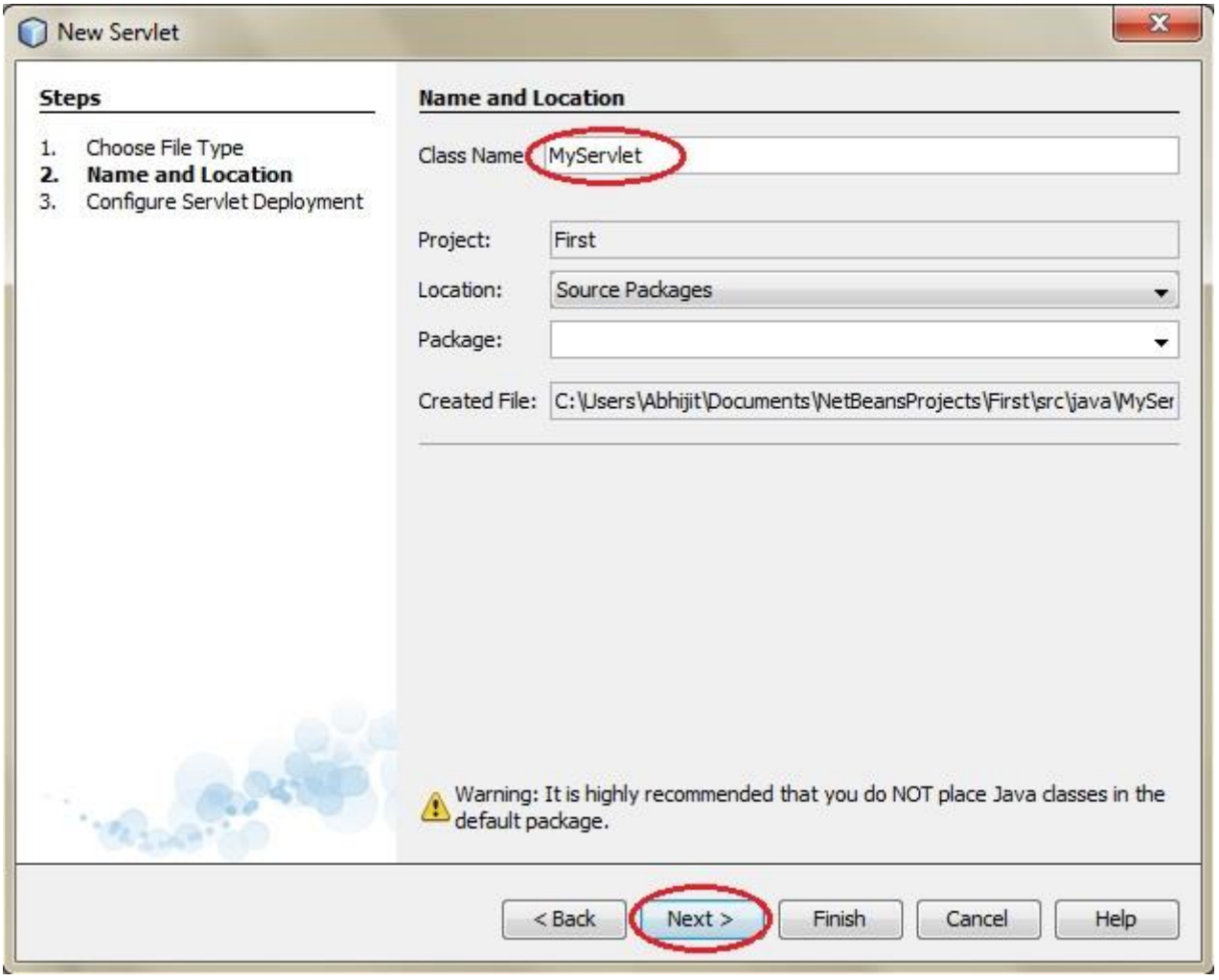


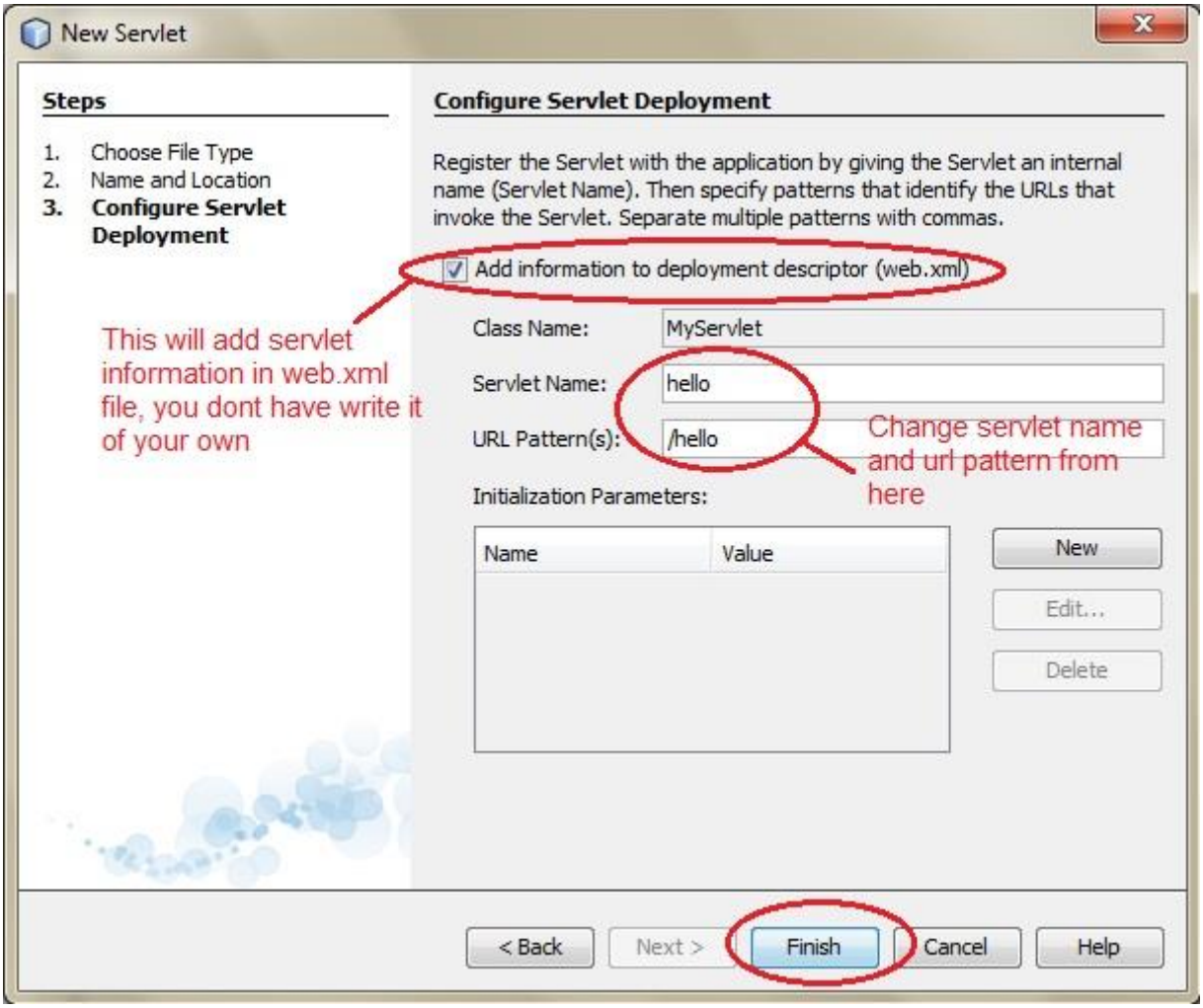
---

6. To create a Servlet, open **Source Package**, right click on **default packages -> New -> Servlet**.



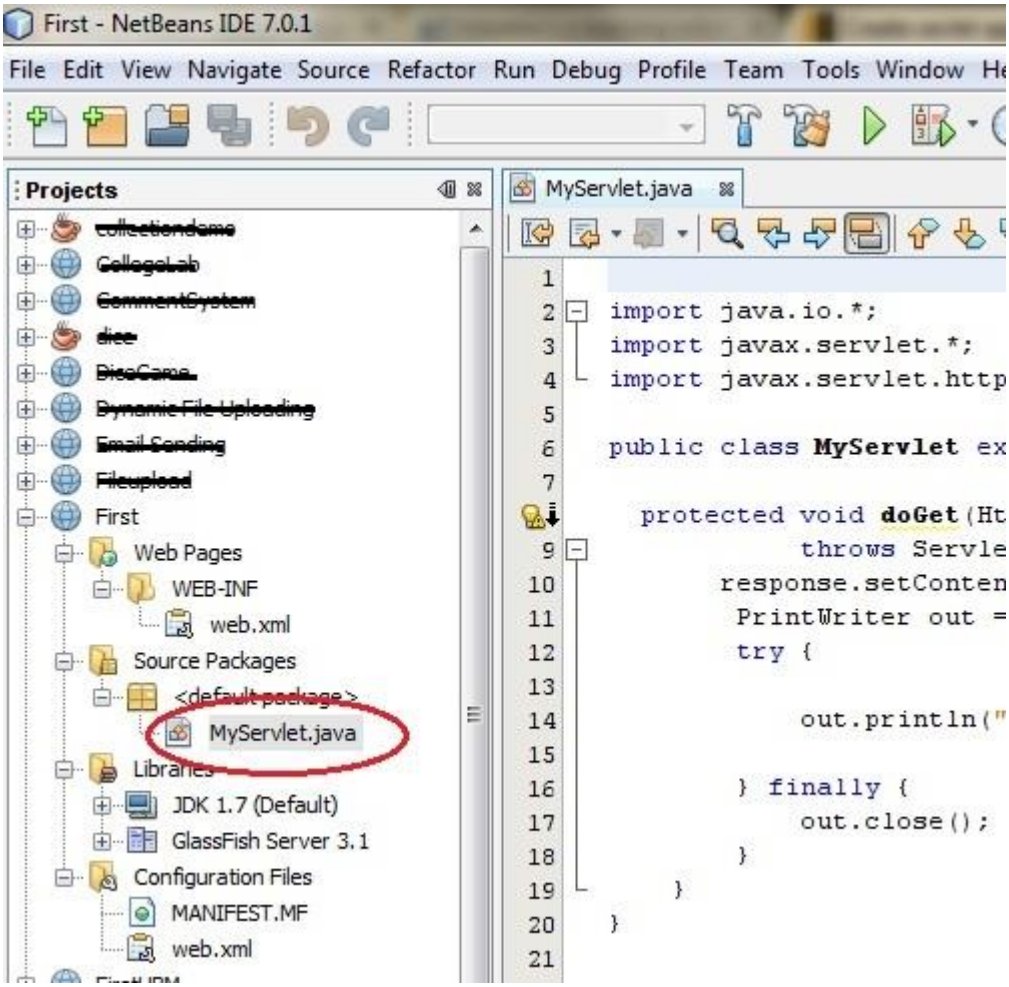
7. Give a Name to your Servlet class file,


New Servlet
Steps
1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment
Name and Location
Class Name: MyServlet
Project: First
Location: Source Packages
Package:
Created File: C:\Users\Abhijit\Documents\NetBeansProjects\First\src\java\MySer
Warning: It is highly recommended that you do NOT place Java classes in the default package.
< Back
Next >
Finish
Cancel
Help



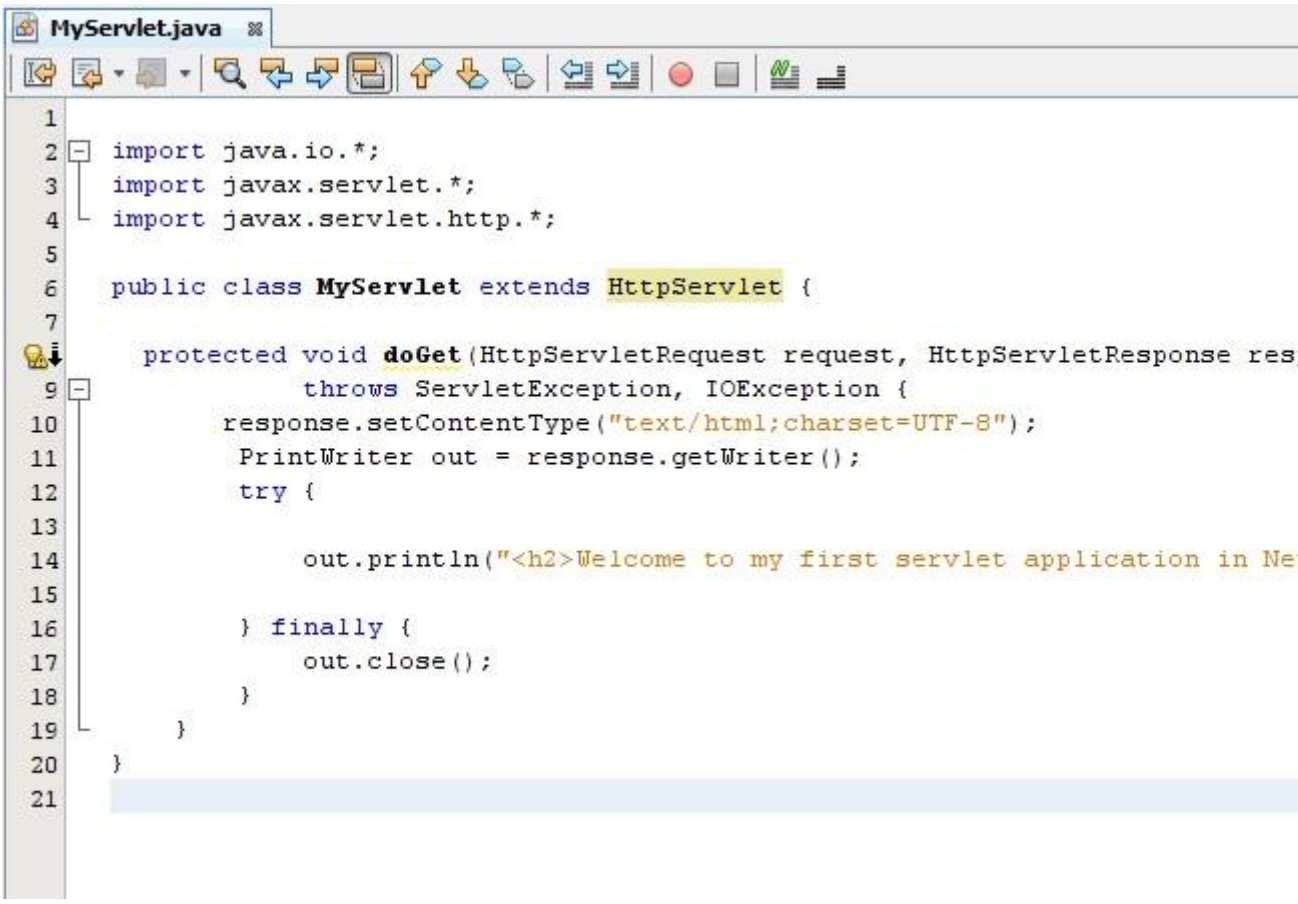
---

8. Now, your Servlet class is ready, and you just need to change the method definitions and you will good to go.



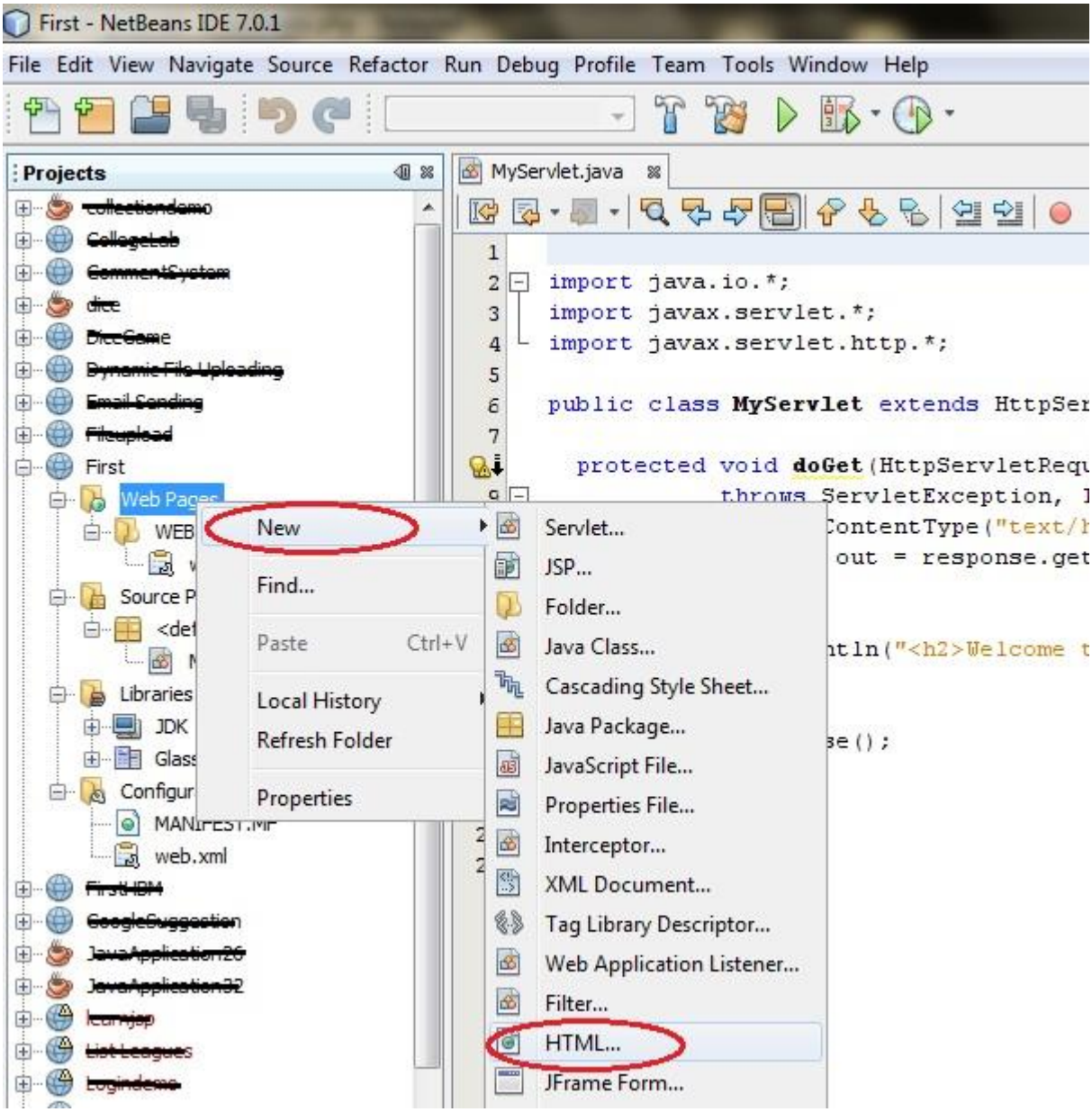
9. Write some code inside your Servlet class.

MyServlet.java

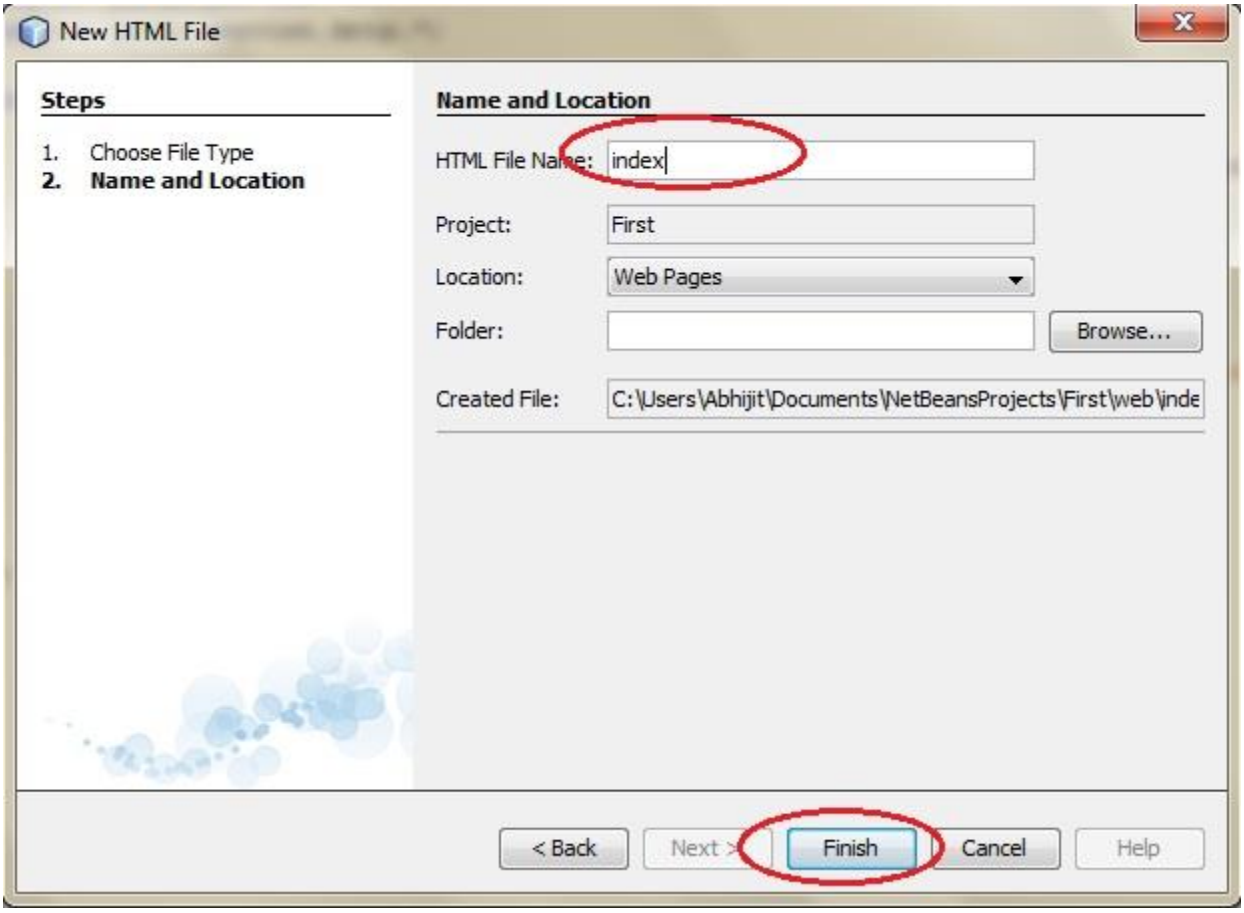


```
1
2  import java.io.*;
3  import javax.servlet.*;
4  import javax.servlet.http.*;
5
6  public class MyServlet extends HttpServlet {
7
8    protected void doGet(HttpServletRequest request, HttpServletResponse res
9              throws ServletException, IOException {
10          response.setContentType("text/html;charset=UTF-8");
11           PrintWriter out = response.getWriter();
12           try {
13
14               out.println("<h2>Welcome to my first servlet application in Ne
15
16           } finally {
17               out.close();
18           }
19       }
20 }
21
```

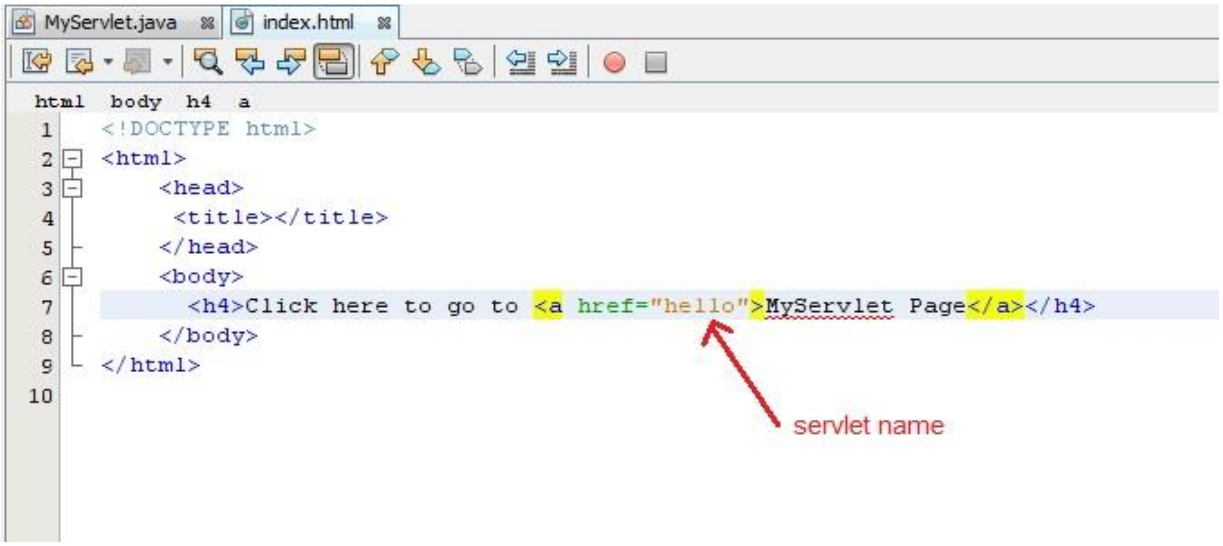
10.Create an HTML file, right click on **Web Pages -> New -> HTML**



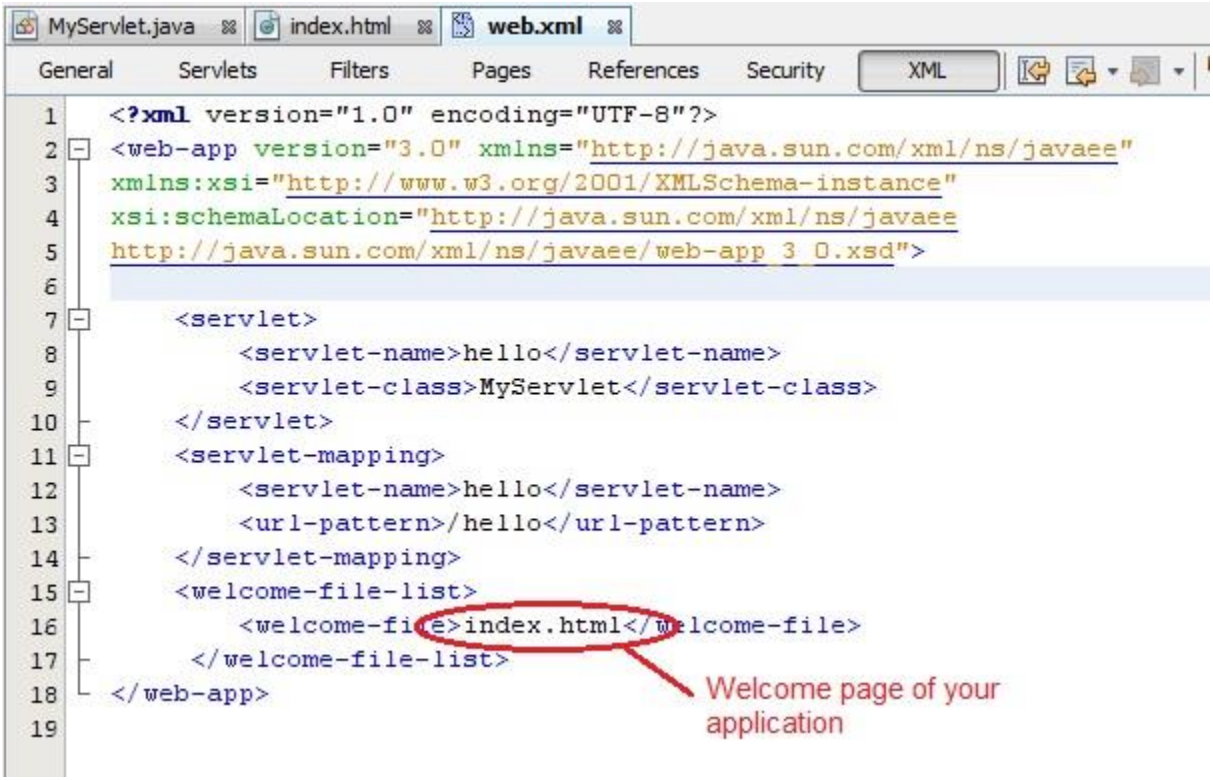
11. Give it a name. We recommend you to name it index, because browser will always pick up the index.html file automatically from a directory. Index file is read as the first page of the web application.



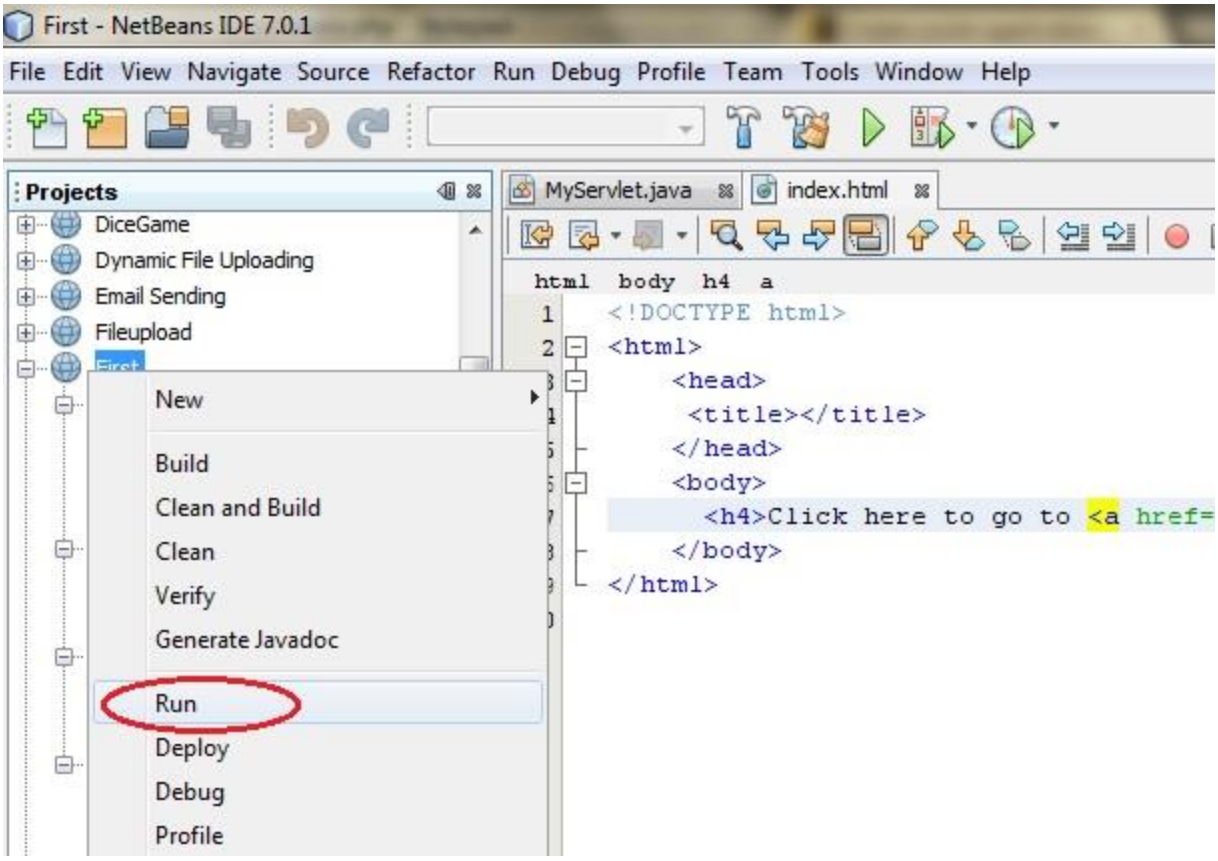
12. Write some code inside your HTML file. We have created a hyperlink to our Servlet in our HTML file.



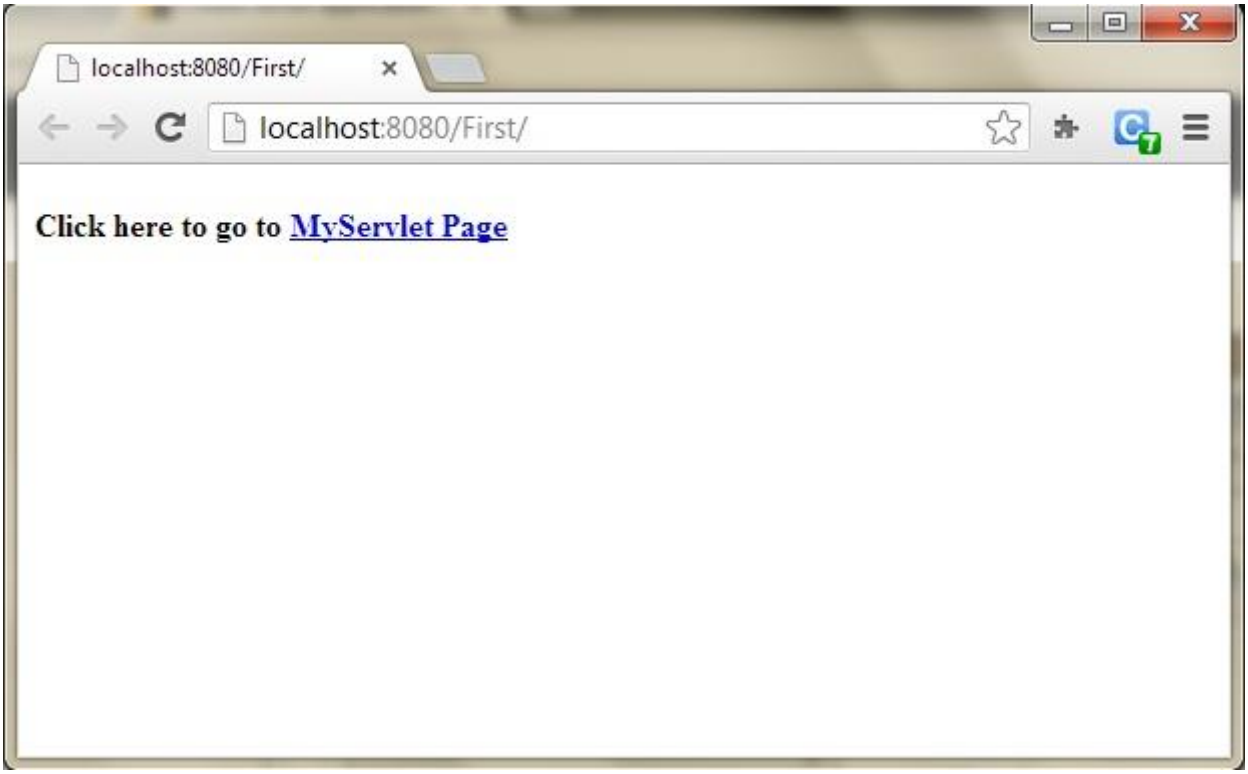
13. Edit **web.xml** file. In the web.xml file you can see, we have specified the **url-pattern** and the **servlet-name**, this means when hello url is accessed our Servlet file will be executed.



14. Run your application, right click on your Project and select **Run**



15.Click on the link created, to open your Servlet.



16.Hurray! Our First Servlet class is running.

The servlet can be created by three ways:

1. By **implementing Servlet interface**,
2. By **inheriting GenericServlet class**, (or)
3. By **inheriting HttpServlet class**

The GenericServlet Class

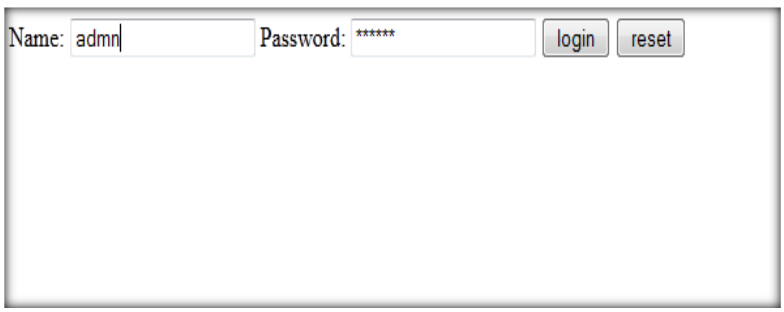
The GenericServlet class provides implementations of the basic life cycle methods for a servlet and is typically subclassed by servlet developers. **GenericServlet** implements the **Servlet** and **ServletConfig** interfaces.

Reading Servlet Parameters

Program no.1=passing parameters using getParameter()

Index.html

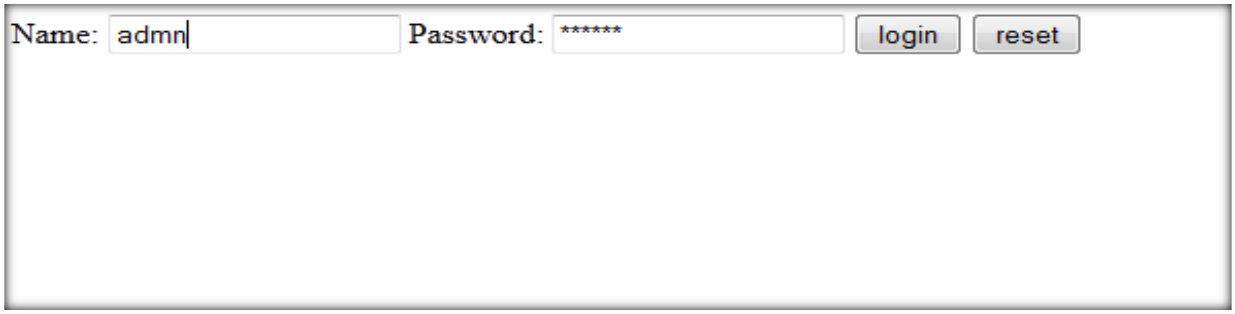
```
<html>
    <head>
        <title>JSP Page</title>
    </head>
   <body><form name=form1 method=get action="NewServlet">
Name: <input type=text name=user1 value="">
Password: <input type=text name=password1 value="">
<input type=submit value="login">
<input type=reset value="reset">
</form>
</body>
</html>
```

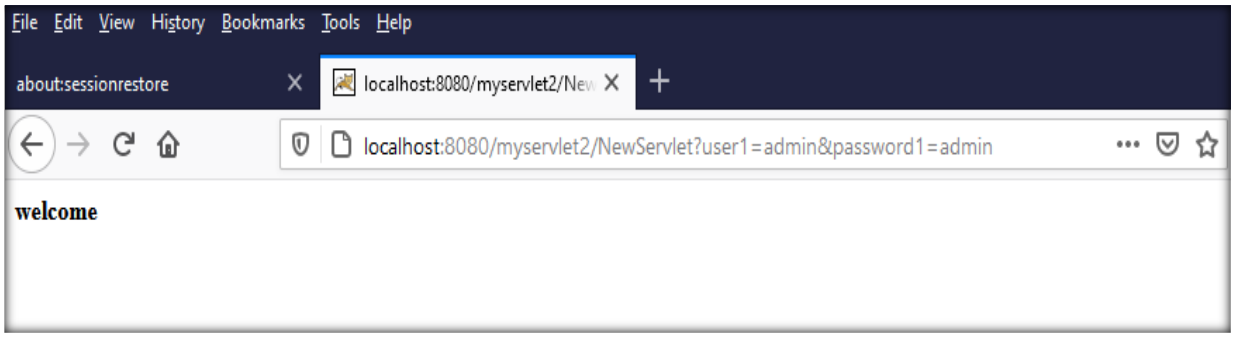


NewServlet.java

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


public class NewServlet extends GenericServlet
 {
public void service(ServletRequest request ,ServletResponse response) throws ServletException,IOException
{
response.setContentType("text/html");
PrintWriter out=response.getWriter();
String name=(String) request.getParameter("user1");
String password=(String) request.getParameter("password1");
if (name.equals("admin") && password.equals("admin"))
{
    out.println("<b>welcome</b>");
}
else
{
out.println("Invalid Name"+"<br>");
out.println(" Invalid Passsword"+"<br>");
}
}
}
```





doGet and doPost method

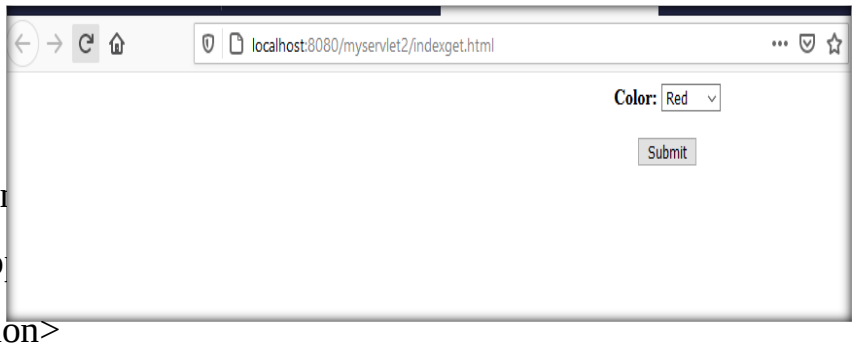
DoGet	DoPost
In doGet Method the parameters are appended to the URL and **sent along with header information**	In doPost, parameters are **sent in separate line** in the body
Maximum size of data that can be sent using doget is 240 bytes	There is no maximum size for data
Parameters are **not encrypted**	Parameters are **encrypted**
DoGet method generally is used to query or to get some information from the server	Dopost is generally used to update or post some information to the server
DoGet is faster if we set the response content length since the same connection is used. Thus increasing the performance	DoPost is slower compared to doGet since doPost does not write the content length
DoGet should be idempotent. i.e. doget should be able to be repeated safely many times	This method does not need to be idempotent. Operations requested through POST can have side effects for which the user can be held accountable, for example, updating stored data or buying items online.

DoGet should be safe without any side effects for which user is held responsible	This method does not need to be either safe

doGet Method

Indexget.html

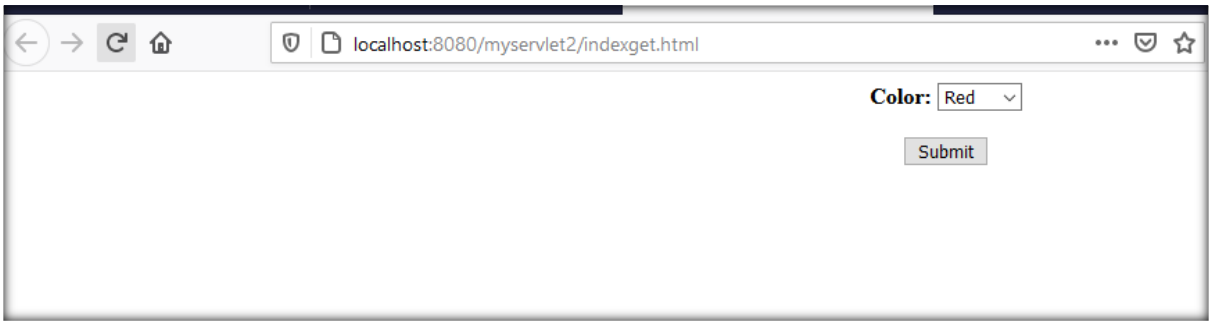
```
<html>
<body>
<center>
<form name="Form1" method=”get”
action="ColorGetServlet">
<B>Color:</B>
<select name="color" size="1">
<option value="Red">Red</optio
<option value="Green">Green</o
<option value="Blue">Blue</option>
</select>
<br><br>
<input type=submit value="Submit">
</form>
</body>
</html>
```



ColorGetServlet.java

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

```
public class ColorGetServlet extends HttpServlet
{
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException
{
String color = request.getParameter("color");
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>The selected color is: ");
pw.println(color);
pw.close();
}
}
```



doPost Method

Indexpost.html

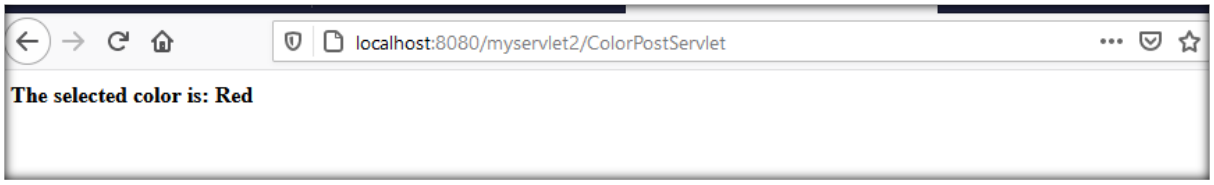
```
<html>
```

```
<body>
<center>
<form name="Form1" method=”post”
action="ColorPostServlet">
<B>Color:</B>
<select name="color" size="1">
<option value="Red">Red</option>
<option value="Green">Green</option>
<option value="Blue">Blue</option>
</select>
<br><br>
<input type=submit value="Submit">
</form>
</body>
</html>
```

ColorPostServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ColorPostServlet extends HttpServlet
 {
public void doPost(HttpServletRequest request,
HttpServletResponse response)throws ServletException, IOException
 {
String color = request.getParameter("color");
response.setContentType("text/html");
```

```
PrintWriter pw = response.getWriter();
pw.println("<B>The selected color is: ");
pw.println(color);
pw.close();
}
}
```



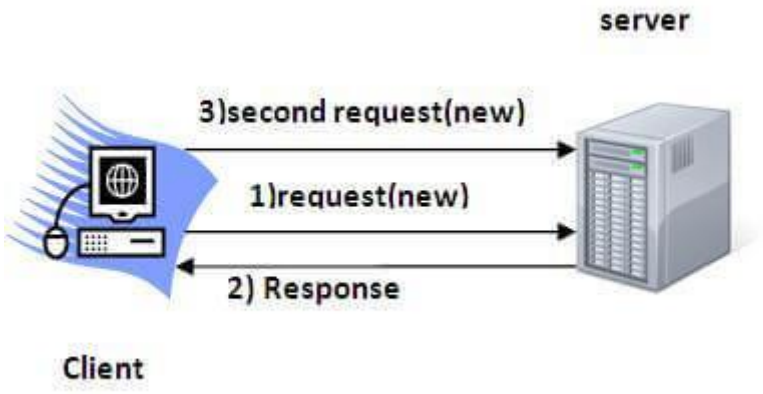
Session Tracking in Servlets

Session simply means a particular interval of time.

Session Tracking is a way to maintain state (data) of an user. It is also known as **session management** in servlet.

Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.

HTTP is stateless that means each request is considered as the new request. It is shown in the figure given below:




```
import java.io.*;

import java.util.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class DateServlet extends HttpServlet

{

public void doGet(HttpServletRequest request,

HttpServletResponse response)

throws ServletException, IOException

 {

// Get the HttpSession object.
```

```
HttpSession hs = request.getSession(true);
// Get writer.
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.print("<B>");
// Display date/time of last access.
Date date = (Date)hs.getAttribute("date");
if(date != null) {
pw.print("Last access: " + date + "<br>");
}
// Display current date/time.
date = new Date();
hs.setAttribute("date", date);
pw.println("Current date: " + date);
}
}
```

Cookies in Servlet

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.

The Cookie Class

The **Cookie** class encapsulates a cookie. A cookie is stored on a client and contains state information. Cookies are valuable for tracking user activities. For example, assume that a user visits an online store. A cookie can save the user's name, address, and other information. The user does not need to enter this data each time he or she visits the store.

A servlet can write a cookie to a user's machine via the **addCookie( )** method of the **HttpServletResponse** interface

---

The names and values of cookies are stored on the user's machine. Some of the information that is saved for each cookie includes the following:

■ The name of the cookie
■ The value of the cookie
■ The expiration date of the cookie
■ The domain and path of the cookie

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.

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 va

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.

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?

```
Cookie ck=new Cookie("user","admin");
//creating cookie object

response.addCookie(ck);//adding cookie in the response
```

program-

indexcookie.html

```
<html>
<body>
<center>
<form name="Form1"
method="post"
action="AddCookieServlet">
<B>Enter a value for MyCookie:</B>
<input type=textbox name="data" size=25 value="">
<input type=submit value="Submit">
</form>
</body>
</html>
```

AddCookieServlet.java

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

```
public class AddCookieServlet extends HttpServlet
 {
public void doPost(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
// Get parameter from HTTP request.
String data = request.getParameter("data");
// Create cookie.
Cookie cookie = new Cookie("MyCookie", data);
// Add cookie to HTTP response.
response.addCookie(cookie);
// Write output to browser.
response.setContentType("text/html");
PrintWriter pw = response.getWriter();
pw.println("<B>MyCookie has been set to");
pw.println(data);
pw.close();
}
}
```