SERVLET BASICS

- Introduction
- Servlet lifecycle methods
- Servlet lifecycle phases
- Servlet Advantages
- Tasks of Servlet Container
- Get vs. Post Request
- Structure of Web Applications
- Creating First Dynamic Web Project
- Exploring Dynamic Web Project
- Creating First Servlet
- Registering Servlet on web.xml file
- Browsing Servlet
- Creating Servlet with HTML Tags
    - Example-1
    - Example-2

- **Introduction**

  A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol. To implement this interface, you can write a generic servlet that extends javax.servlet.GenericServlet or an HTTP servlet that extends javax.servlet.http.HttpServlet.

  This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods and are called in the following sequence:

  - The servlet is constructed, then initialized with the init method.
  - Any calls from clients to the service method are handled.
  - 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.

- Servlet Lifecycle Methods
  - **init()**
  - **service()**
  - **destroy()**

init()

The init method is designed to be called only once. If an instance of the servlet does not exist, the web container:

- Loads the servlet class
- Creates an instance of the servlet class
- Initializes it by calling the init method

```
public void init() throws ServletException {
    // Initialization code like set up database etc....
}
```

service()

This method is only called after the servlet's init() method has completed successfully. The Container calls the **service()** method to handle requests coming from the client, interprets the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

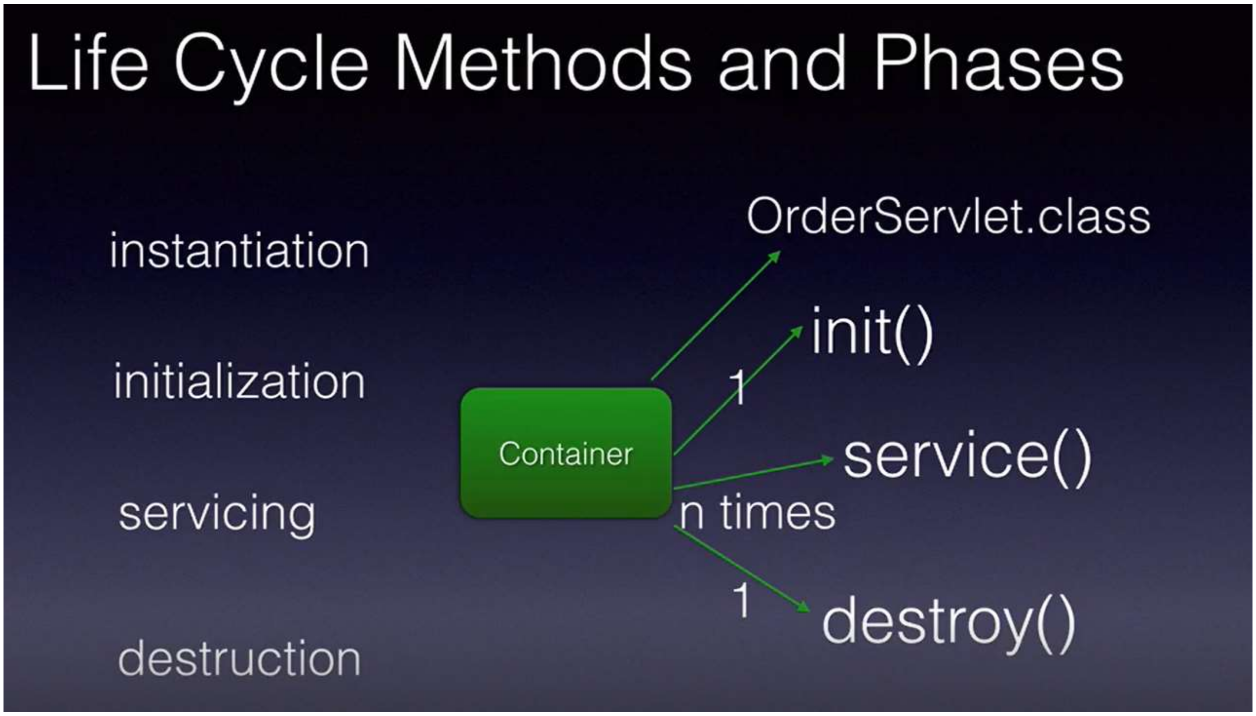
```
public void service(ServletRequest request, ServletResponse
response) throws ServletException, IOException {
    // ...
}
```

destroy()

Called by the Servlet Container to take the Servlet out of service. This method is only called once all threads within the servlet's service method have exited or after a timeout period has passed. After the container calls this method, it will not call the service method again on the Servlet.

```
        public void destroy() {
            //
        }
```

- **Servlet Lifecycle Phases**
    - Instantiation (declare)
    - Initialization (init)
    - Servicing (service)
    - Destruction (desctroy)



Life Cycle Methods and Phases

instantiation

initialization

servicing

destruction

- **Servlet Advantages**
    - Less response time because each request runs in a separate thread
    - Servlets are scalable
    - Servlets are robust and object-oriented
    - Servlets are platform-independent
    - Servlets are secure and offer portability

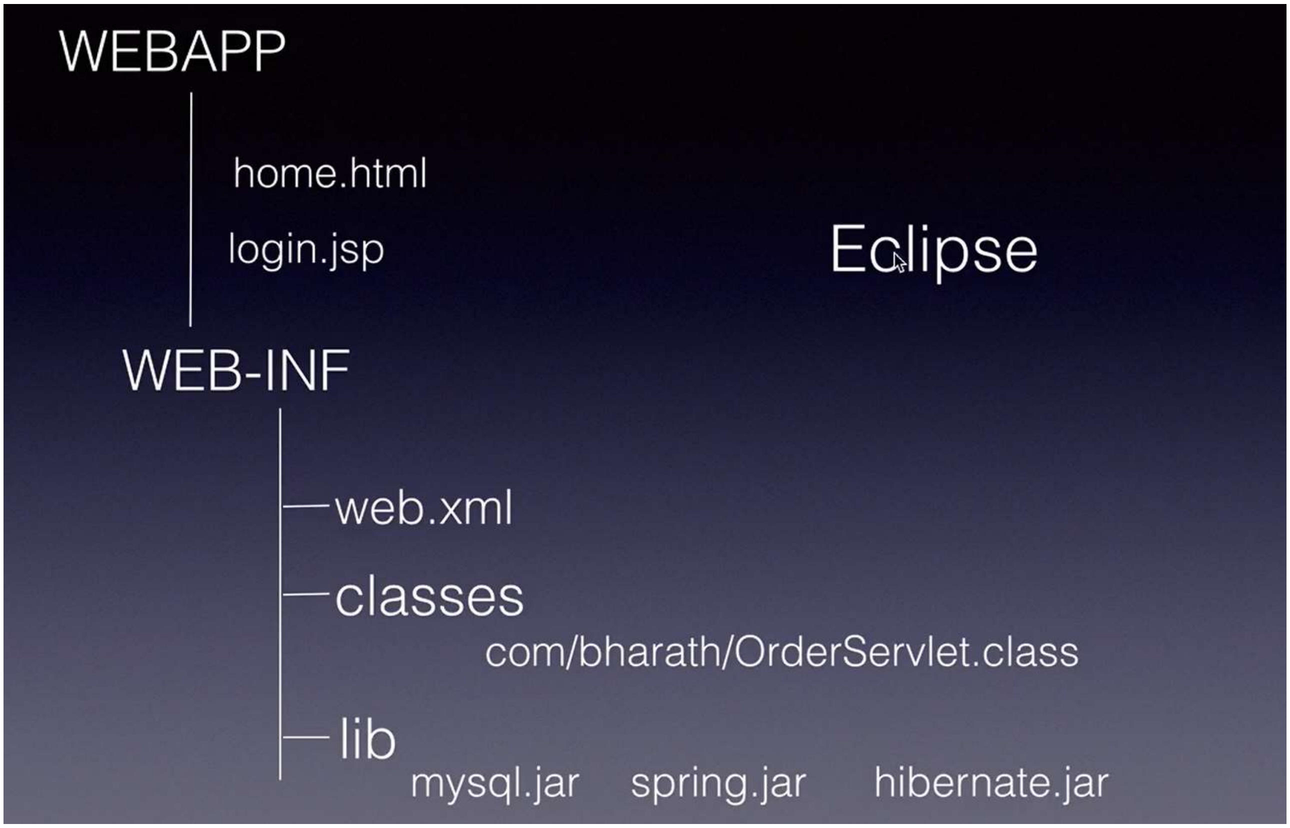
- **Tasks of Servlet Container**
    - Life Cycle Management
    - Multithreaded Support
    - Object Pooling
    - Security

- **Get vs. Post Request**

Feature	GET	POST

Sending of data	Client data is appended to URL and sent	Client data is sent separately
Storing in the Browser History	As data is appended, the client data is stored in the browser history	As data is sent separately, the client data is not stored in the browser history
Bookmark	The URL with client data can be bookmarked. Thereby, later without filling the HTML form, the same data can be sent to server	Not possible to bookmark
Encoding or enctype	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. For binary data, multipart enctype to be used
Limitation of data sent	Limited to 2048 characters (browser dependent)	Unlimited data
Hacking easiness	Easy to hack the data as the data is stored in the browser history	Difficult to hack as the data is sent separately in an HTML form
Type of data sent	Only ASCII data can be sent	Any type of data can be sent including the binary data
Data secrecy	Data is not secret as other people can see the data in the browser history	Data is secret as not stored in the browser history
When to be used	Prefer when data sent is not secret. Do not use for passwords etc.	Prefer for critical and sensitive data like passwords etc.
Cache	Can be caught	Cannot be caught
Default	If not mentioned, GET is assumed as default	Should be mentioned explicitly
Performance	Relatively faster as data is appended to URL	A separate message body is to be created

- **Structure of Web Applications**



WEBAPP

home.html

login.jsp

Eclipse

WEB-INF

web.xml

classes

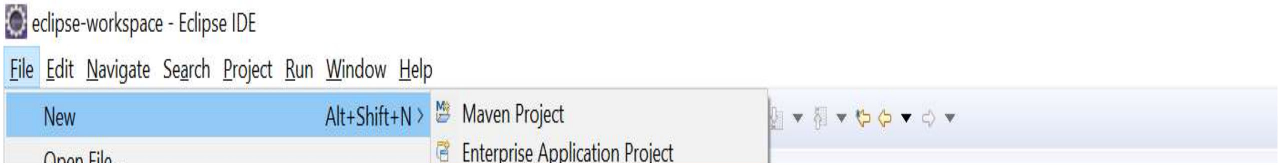
com/bharath/OrderServlet.class

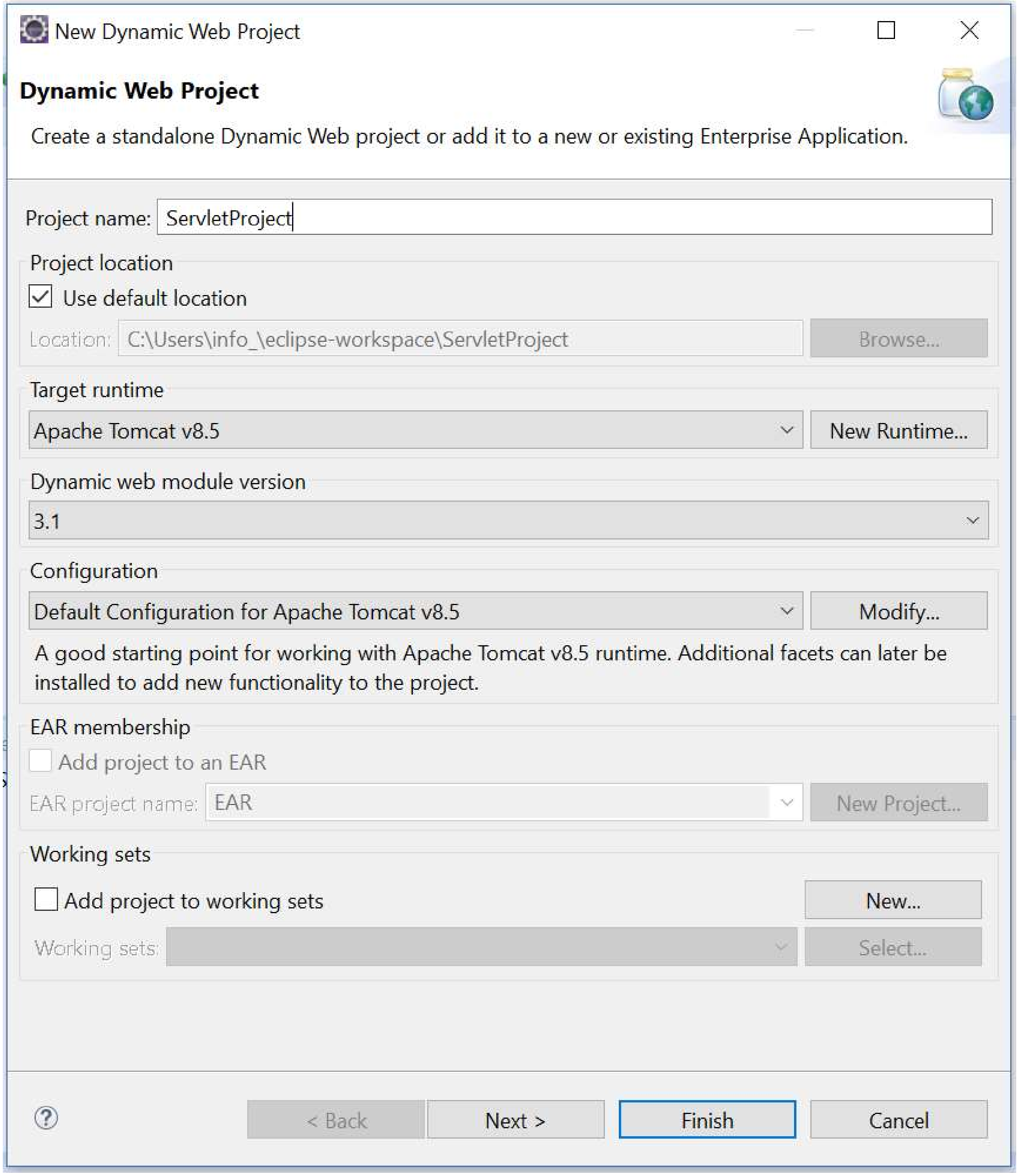
lib

mysql.jar spring.jar hibernate.jar

Creating First Dynamic Web Project
Steps:

- **File > New > Dynamic Web Project**


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N >
Maven Project
Enterprise Application Project
Open File



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: ServletProject

Project location

☑ Use default location

Location: C:\Users\info_\eclipse-workspace\ServletProject

Target runtime: Apache Tomcat v8.5

Dynamic web module version: 3.1

Configuration: Default Configuration for Apache Tomcat v8.5

A good starting point for working with Apache Tomcat v8.5 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

□ Add project to an EAR

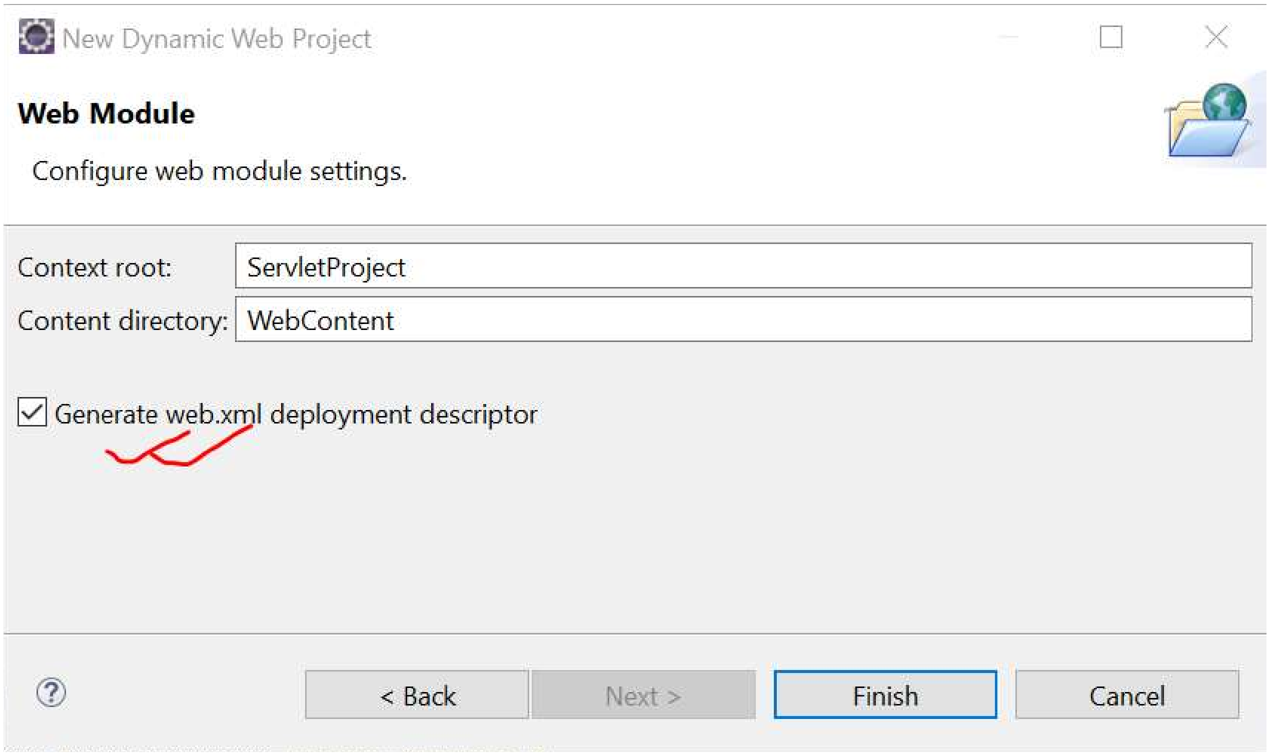
EAR project name: EAR

Working sets

□ Add project to working sets

Working sets:

- **Next > Next**



Web Module

Configure web module settings.

Context root: ServletProject

Content directory: WebContent

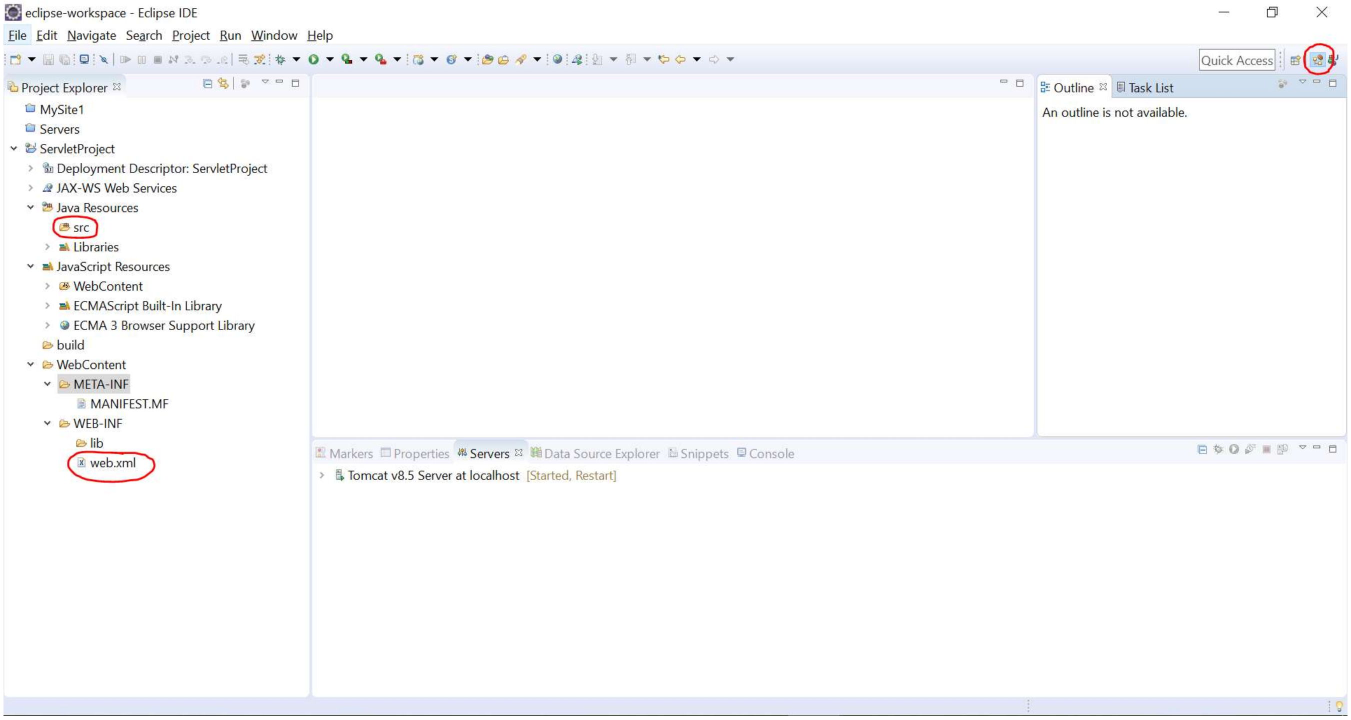
☑ Generate web.xml deployment descriptor

- Click on **Finish**

Note:

- Close Project
- Open Existing Project

Exploring Dynamic Web Project

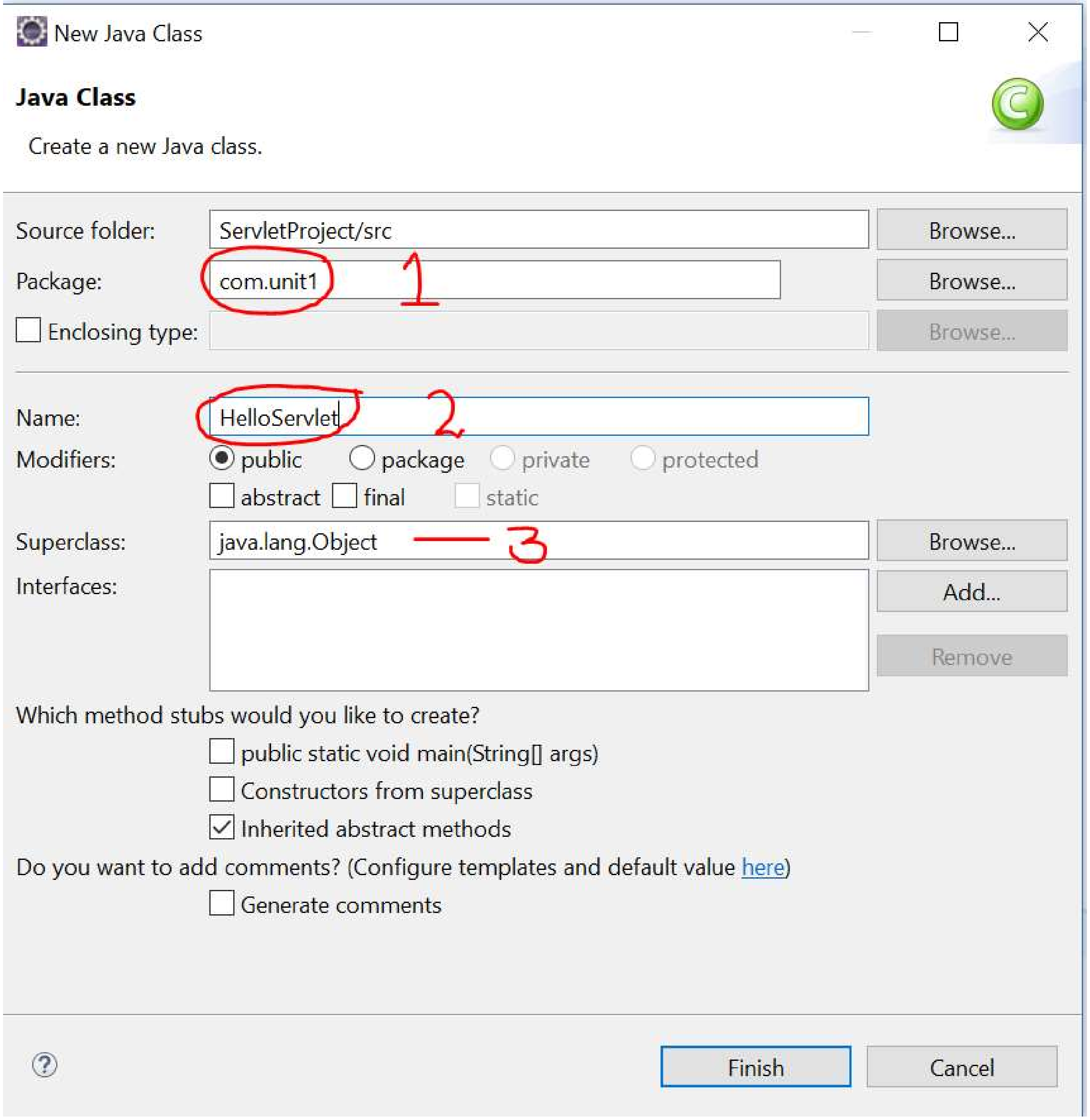


An outline is not available.

Tomcat v8.5 Server at localhost [Started, Restart]

Creating First Servlet

- Right Click on **Src** of **Java Resources** > **New** > **Class**



New Java Class

Java Class

Create a new Java class.

Source folder: ServletProject/src

Package: com.unit1

□ Enclosing type:

Name: HelloServlet

Modifiers: ● public ○ package ○ private ○ protected

□ abstract □ final □ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

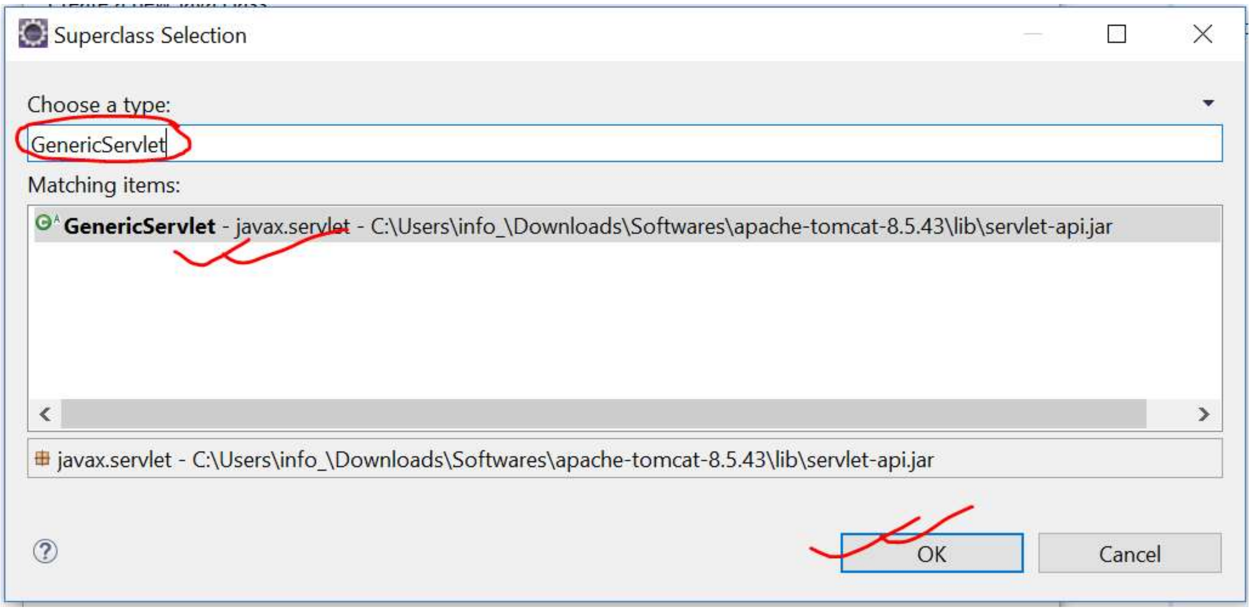
□ public static void main(String[] args)

□ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

□ Generate comments



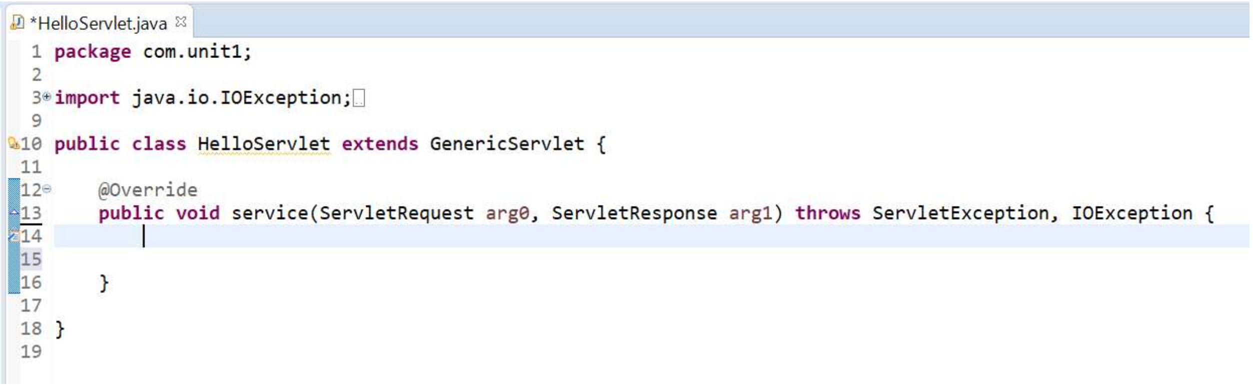
Choose a type:

GenericServlet

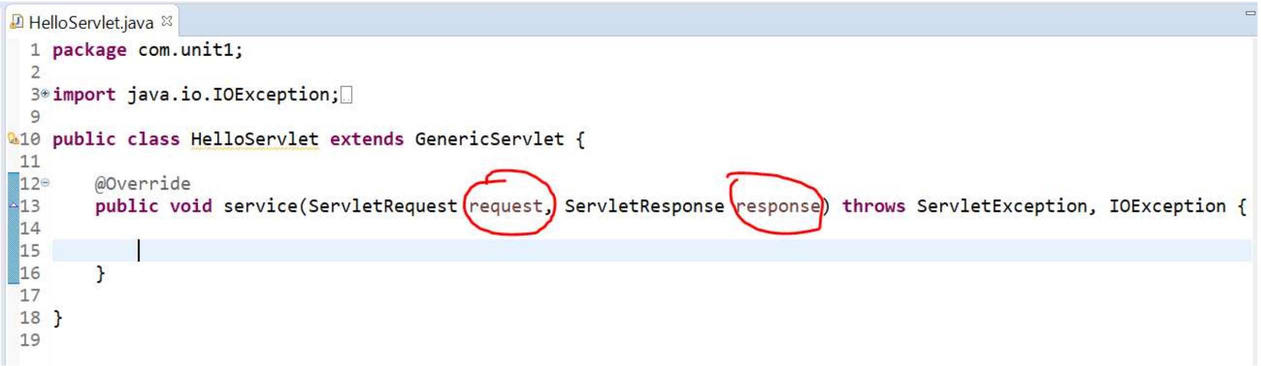
Matching items:

GenericServlet - javax.servlet - C:\Users\info_\Downloads\Softwares\apache-tomcat-8.5.43\lib\servlet-api.jar

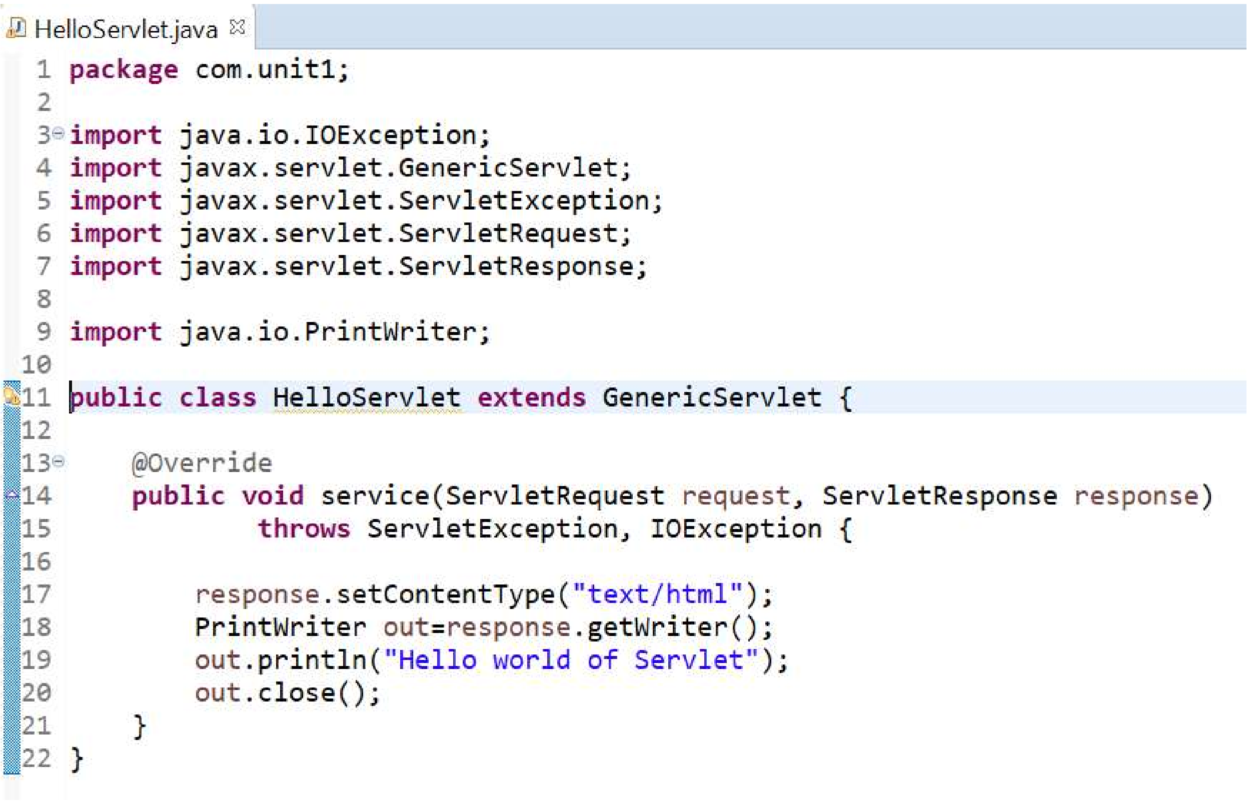
javax.servlet - C:\Users\info_\Downloads\Softwares\apache-tomcat-8.5.43\lib\servlet-api.jar



```
package com.unit1;

import java.io.IOException;

public class HelloServlet extends GenericServlet {

    @Override
    public void service(ServletRequest arg0, ServletResponse arg1) throws ServletException, IOException {
        

    }

}

```

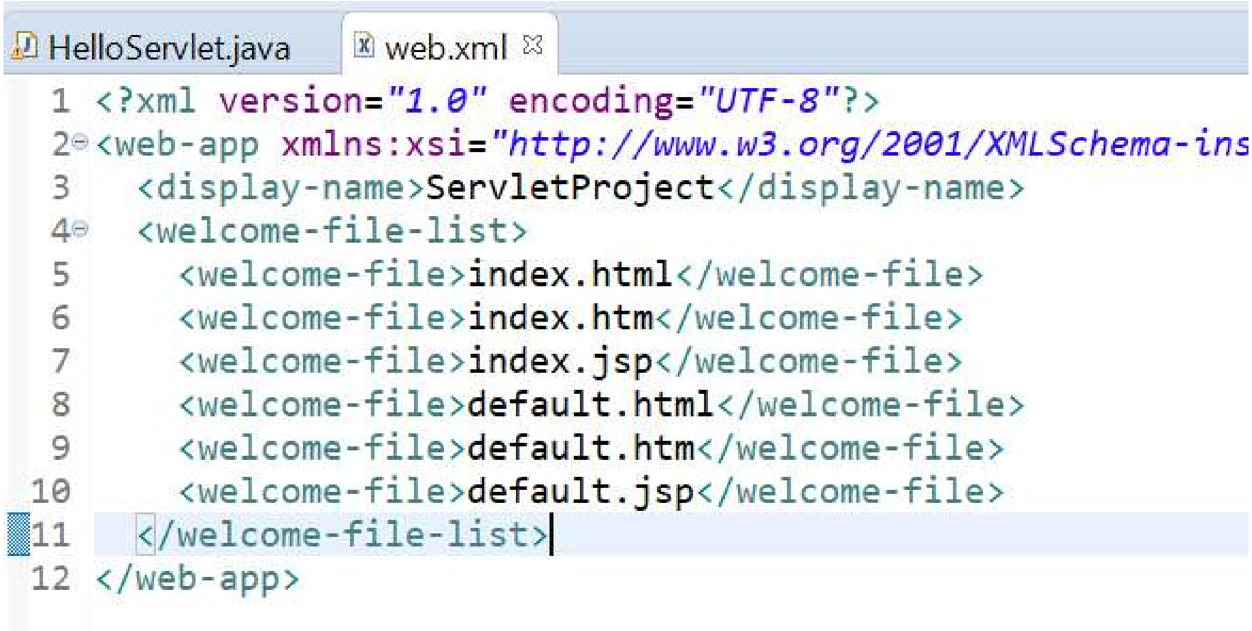


```
package com.unit1;

import java.io.IOException;

public class HelloServlet extends GenericServlet {

    @Override
    public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException {

        
    }

}

```

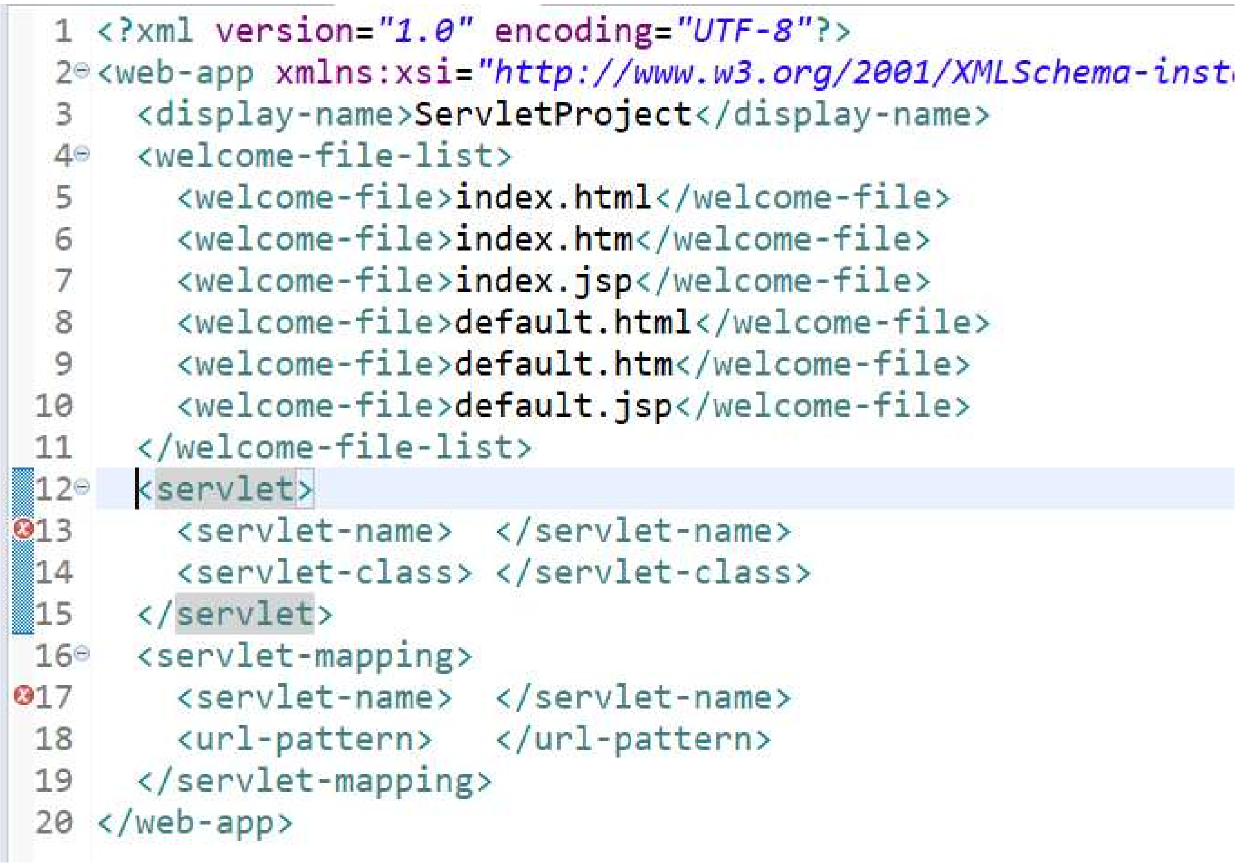


```java
package com.unit1;

import java.io.IOException;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

import java.io.PrintWriter;

public class HelloServlet extends GenericServlet {

    @Override
    public void service(ServletRequest request, ServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("Hello world of Servlet");
        out.close();
    }
}
```

- Registering Servlet on **web.xml** file



```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-ins
  <display-name>ServletProject</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

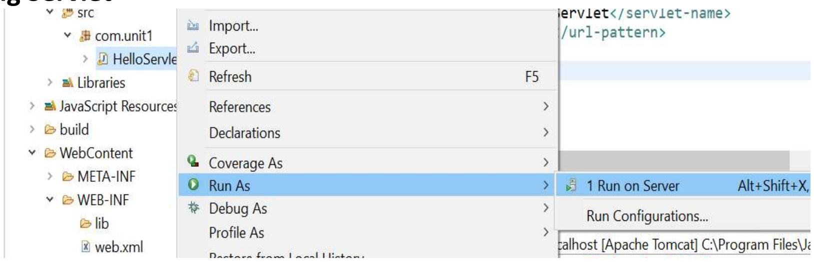


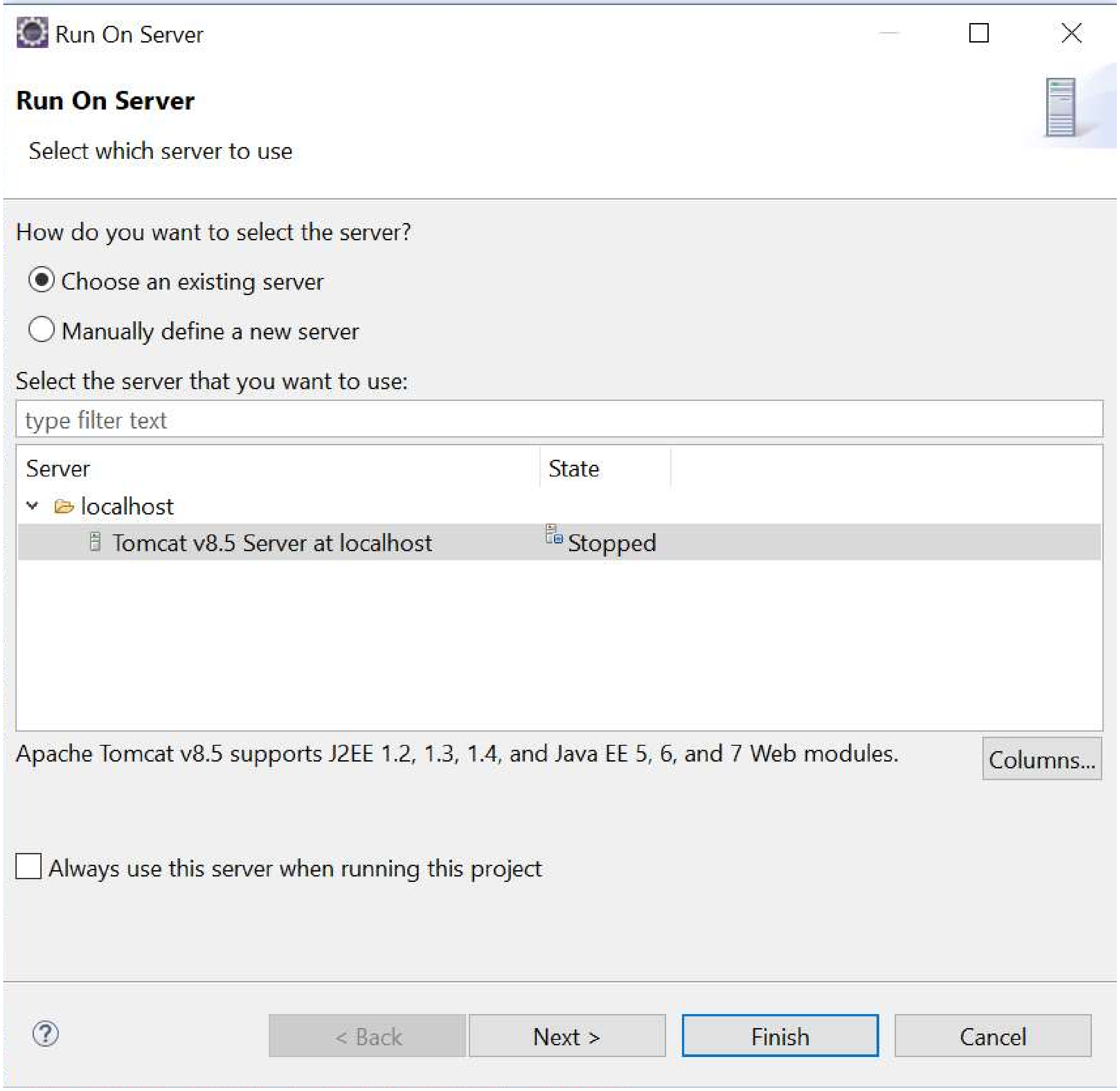
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
  <display-name>ServletProject</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>  </servlet-name>
    <servlet-class> </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>  </servlet-name>
    <url-pattern>   </url-pattern>
  </servlet-mapping>
</web-app>
```

Note:

```
<servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>com.unit1.HelloServlet</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

Browsing Servlet





Run On Server

Run On Server

Select which server to use

How do you want to select the server?

● Choose an existing server

○ Manually define a new server

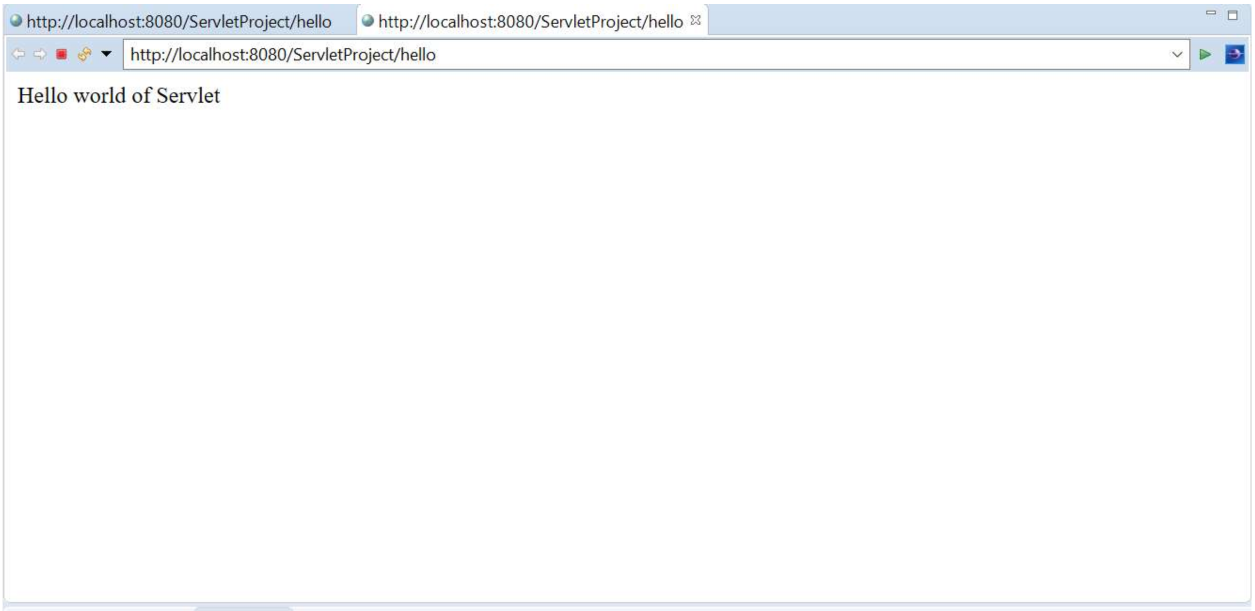
Select the server that you want to use:

type filter text

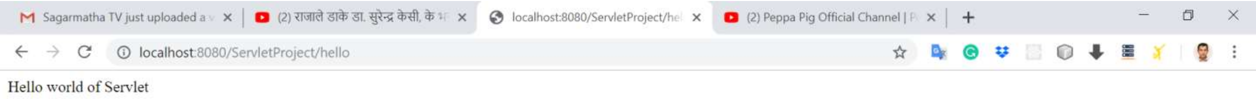
Server	State
localhost	
Tomcat v8.5 Server at localhost	Stopped

Apache Tomcat v8.5 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, and 7 Web modules.

□ Always use this server when running this project

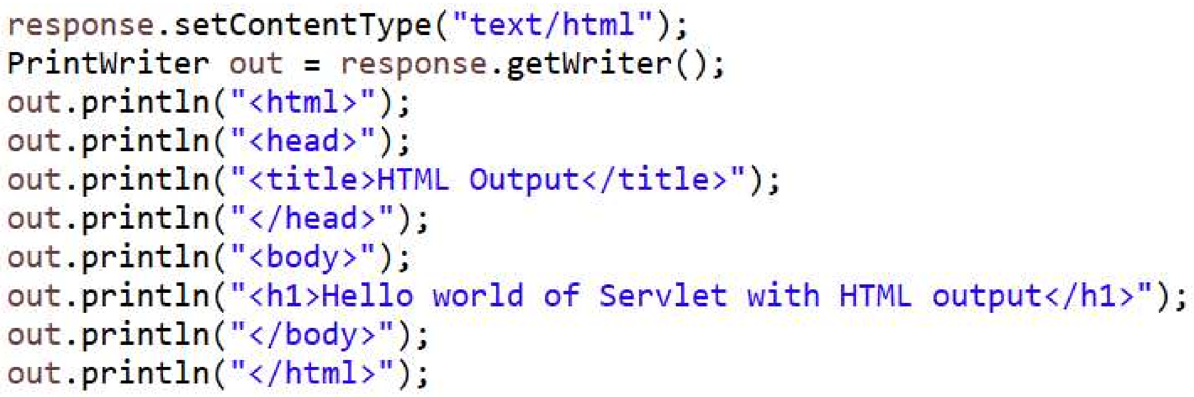

http://localhost:8080/ServletProject/hello
http://localhost:8080/ServletProject/hello
http://localhost:8080/ServletProject/hello

```
Hello world of Servlet
```

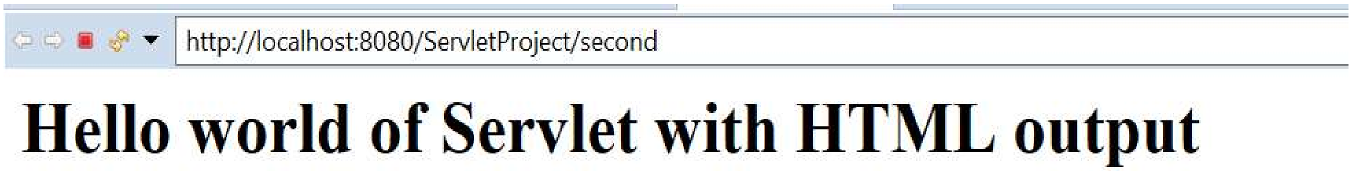

Sagarmatha TV just uploaded a v
(2) राजाले डाके डा. सुरेन्द्र केसी, के भ
localhost:8080/ServletProject/hel
(2) Peppa Pig Official Channel | P
localhost:8080/ServletProject/hello

```
Hello world of Servlet
```

Creating Servlet with HTML Tags

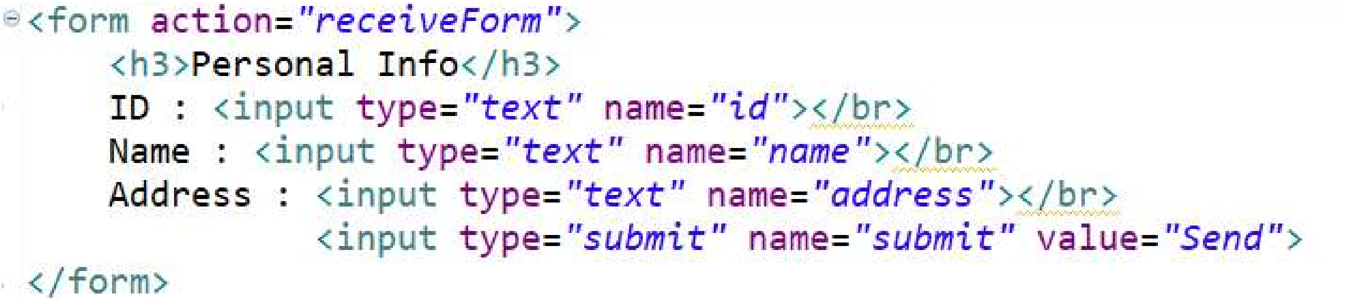


```java
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
out.println("<title>HTML Output</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Hello world of Servlet with HTML output</h1>");
out.println("</body>");
out.println("</html>");
```



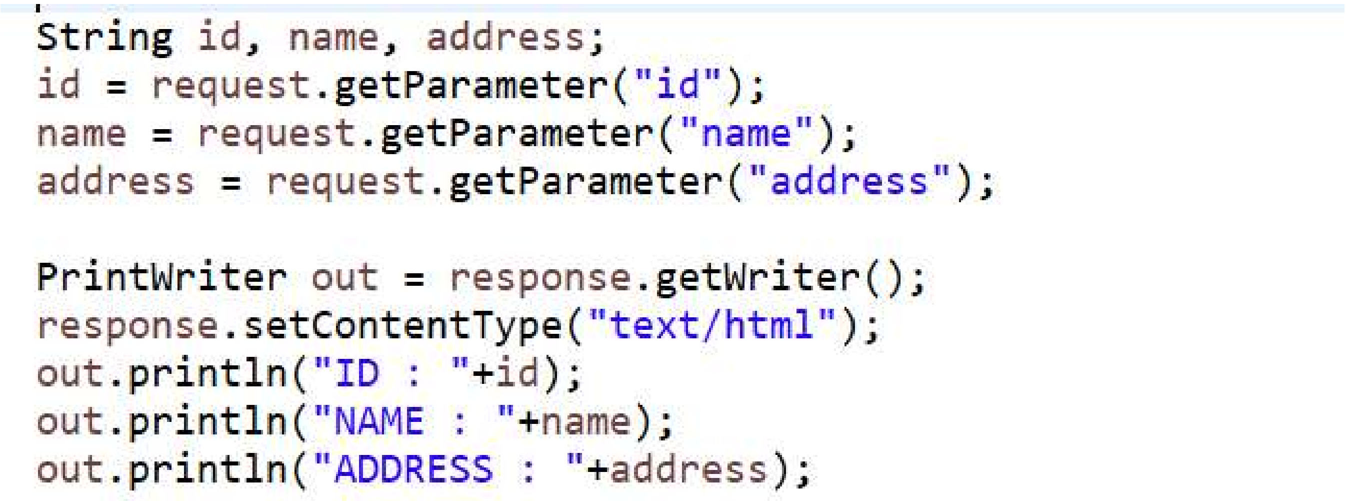
Hello world of Servlet with HTML output

Sending data from web form
EXAMPLE-1
EntryForm.html



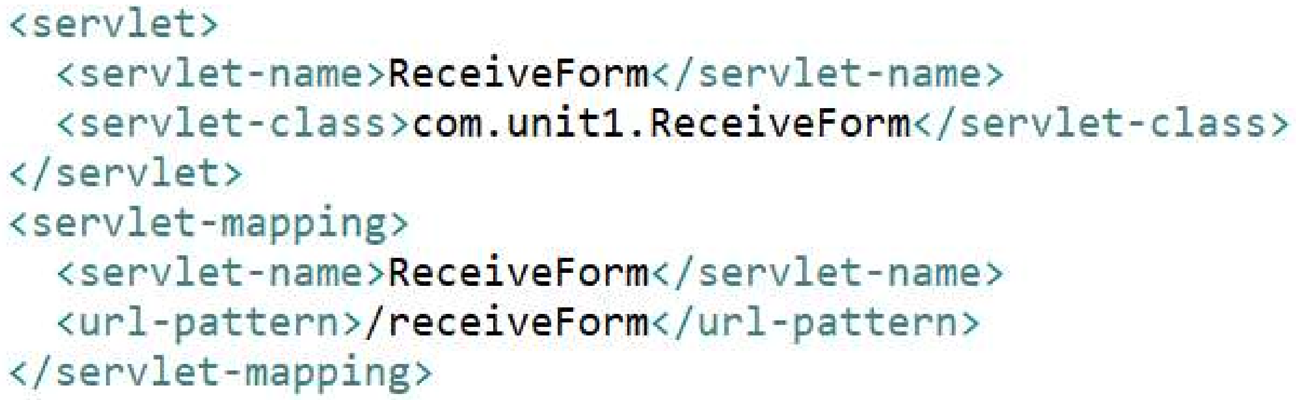
```html
<form action="receiveForm">
    <h3>Personal Info</h3>
    ID : <input type="text" name="id"></br>
    Name : <input type="text" name="name"></br>
    Address : <input type="text" name="address"></br>
              <input type="submit" name="submit" value="Send">
</form>
```

ReceiveForm.java

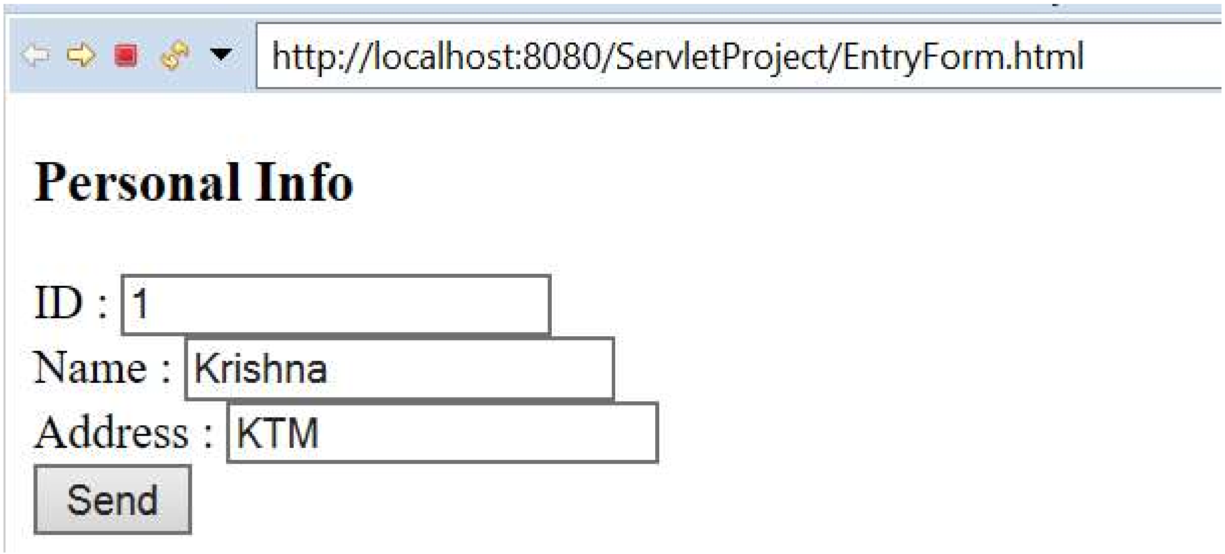


```
String id, name, address;
id = request.getParameter("id");
name = request.getParameter("name");
address = request.getParameter("address");

PrintWriter out = response.getWriter();
response.setContentType("text/html");
out.println("ID : "+id);
out.println("NAME : "+name);
out.println("ADDRESS : "+address);
```

web.xml



```
<servlet>
  <servlet-name>ReceiveForm</servlet-name>
  <servlet-class>com.unit1.ReceiveForm</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>ReceiveForm</servlet-name>
  <url-pattern>/receiveForm</url-pattern>
</servlet-mapping>
```



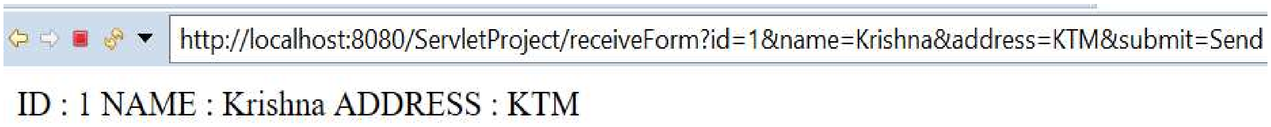
http://localhost:8080/ServletProject/EntryForm.html

Personal Info

ID : 1

Name : Krishna

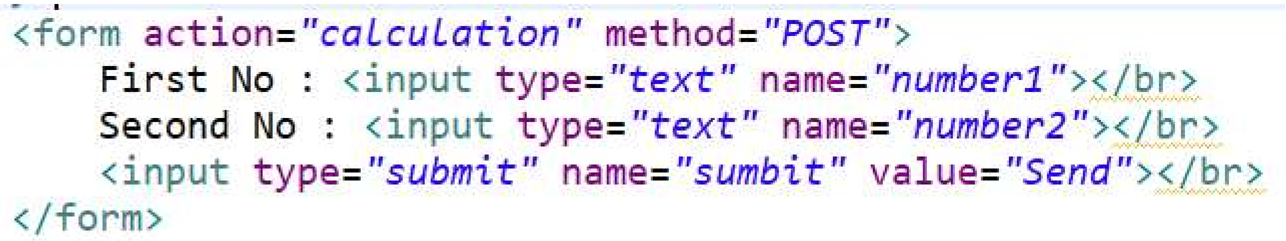
Address : KTM



```
http://localhost:8080/ServletProject/receiveForm?id=1&name=Krishna&address=KTM&submit=Send
```

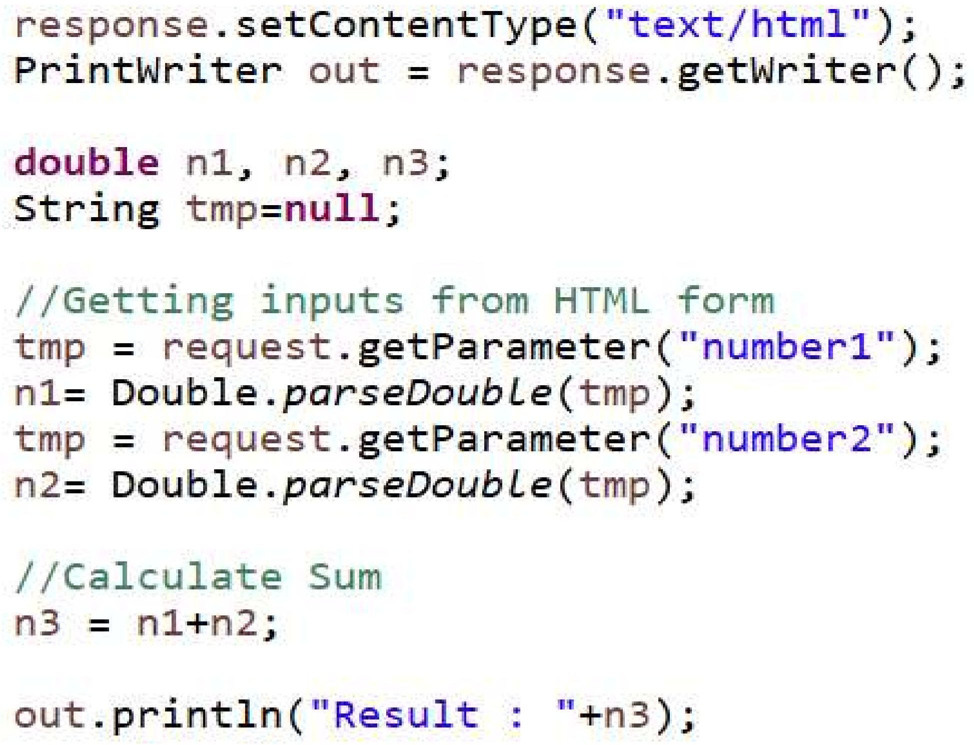
ID : 1 NAME : Krishna ADDRESS : KTM

EXAMPLE-2
Web From

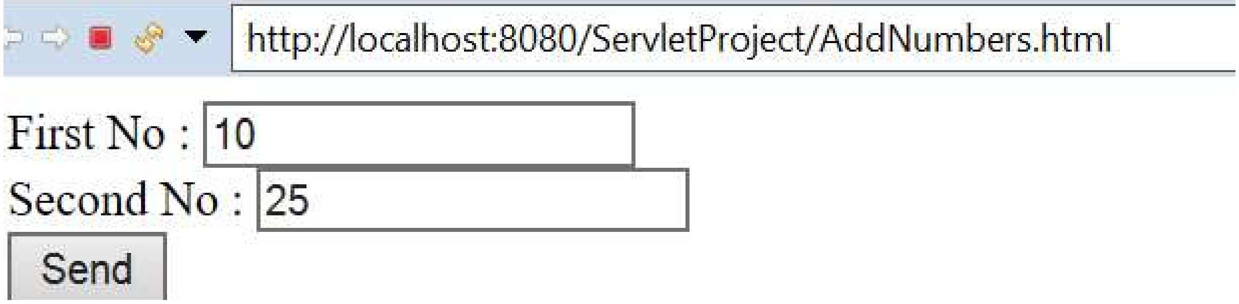


```html
<form action="calculation" method="POST">
    First No : <input type="text" name="number1"></br>
    Second No : <input type="text" name="number2"></br>
    <input type="submit" name="sumbit" value="Send"></br>
</form>
```

Servlet



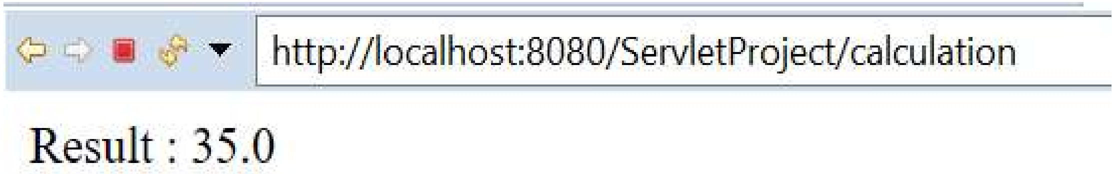
```java
response.setContentType("text/html");
PrintWriter out = response.getWriter();

double n1, n2, n3;
String tmp=null;

//Getting inputs from HTML form
tmp = request.getParameter("number1");
n1= Double.parseDouble(tmp);
tmp = request.getParameter("number2");
n2= Double.parseDouble(tmp);

//Calculate Sum
n3 = n1+n2;

out.println("Result : "+n3);
```



http://localhost:8080/ServletProject/AddNumbers.html

First No : 10

Second No : 25



http://localhost:8080/ServletProject/calculation

Result : 35.0

SERVLET IN DETAILS

- GenericServlet Class
- ServletRequest Interface
- ServletResponse Interface
- HttpServlet Class
- HttpServletRequest Interface
- HttpServletResponse Interface
- Response Content Type
- Passing Form Parameters
- RequestDispatcher
- Send Redirect
- Servlet Config/Context
- Http Session
- Http Cookies
- URL Rewriting
- Servlet Filter
- Hit Counter
- Upload File
- Download File
- Send Plaintext Email
- Send Email with Attachment
- Database Connectivity
- Servlet Annotations

GenericServlet Class

- javax.servlet.GenericServlet

Constructors

- GenericServlet()

Methods

- **void destroy()**
  Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.

- String getInitParameter(String name)
  Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.

- Enumeration<String> getInitParameterNames()
  Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.

- ServletConfig getServletConfig()
  Returns this servlet's ServletConfig object.

- ServletContext getServletContext()
  Returns a reference to the ServletContext in which this servlet is running.

- String getServletInfo()
  Returns information about the servlet, such as author, version, and copyright.

- String getServletName()
  Returns the name of this servlet instance.

- **void init()**
  A convenience method which can be overridden so that there's no need to call super.init(config).

- void init(ServletConfig config)
  Called by the servlet container to indicate to a servlet that the servlet is being placed into service.

- void log(String msg)
  Writes the specified message to a servlet log file, prepended by the servlet's name.

- void log(String message, Throwable t)
  Writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file, prepended by the servlet's name.

- **abstract void service(ServletRequest req, ServletResponse res)**
  Called by the servlet container to allow the servlet to respond to a request.

ServletRequest Interface

- AsyncContext getAsyncContext()
  Gets the AsyncContext that was created or reinitialized by the most recent invocation of startAsync() or startAsync(ServletRequest,ServletResponse) on this request.

- Object getAttribute(java.lang.String name)
  Returns the value of the named attribute as an Object, or null if no attribute of the given name exists.

- Enumeration getAttributeNames()
  Returns an Enumeration containing the names of the attributes available to this request.

- String getCharacterEncoding()
  Returns the name of the character encoding used in the body of this request.

- int getContentLength()
  Returns the length, in bytes, of the request body and made available by the input stream, or -1 if the length is not known.

- String getContentType()
  Returns the MIME type of the body of the request, or null if the type is not known.

- DispatcherType getDispatcherType()
  Gets the dispatcher type of this request.

- ServletInputStream getInputStream()
  Retrieves the body of the request as binary data using a ServletInputStream.

- String getLocalAddr()
  Returns the Internet Protocol (IP) address of the interface on which the request was received.

- Locale getLocale()
  Returns the preferred Locale that the client will accept content in, based on the Accept-Language header.

- Enumeration getLocales()
  Returns an Enumeration of Locale objects indicating, in decreasing order starting with the preferred locale, the locales that are acceptable to the client based on the Accept-Language header.

- String getLocalName()
  Returns the host name of the Internet Protocol (IP) interface on which the request was received.

- int getLocalPort()
  Returns the Internet Protocol (IP) port number of the interface on which the request was received.

- String getParameter(java.lang.String name)
  Returns the value of a request parameter as a String, or null if the parameter does not exist.

- Map getParameterMap()

Returns a java.util.Map of the parameters of this request.

- Enumeration getParameterNames()
  Returns an Enumeration of String objects containing the names of the parameters contained in this request.

- String[] getParameterValues(String name)
  Returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist.

- String getProtocol()
  Returns the name and version of the protocol the request uses in the form protocol/majorVersion.minorVersion, for example, HTTP/1.1.

- BufferedReader getReader()
  Retrieves the body of the request as character data using a BufferedReader.

- String getRealPath(String path)
  Deprecated. As of Version 2.1 of the Java Servlet API, use ServletContext#getRealPath instead.

- String getRemoteAddr()
  Returns the Internet Protocol (IP) address of the client or last proxy that sent the request.

- String getRemoteHost()
  Returns the fully qualified name of the client or the last proxy that sent the request.

- int getRemotePort()
  Returns the Internet Protocol (IP) source port of the client or last proxy that sent the request.

- RequestDispatcher getRequestDispatcher(String path)
  Returns a RequestDispatcher object that acts as a wrapper for the resource located at the given path.

- String getScheme()
  Returns the name of the scheme used to make this request, for example, http, https, or ftp.

- String getServerName()
  Returns the host name of the server to which the request was sent.

- int getServerPort()
  Returns the port number to which the request was sent.

- ServletContext getServletContext()
  Gets the servlet context to which this ServletRequest was last dispatched.

- boolean isAsyncStarted()
  Checks if this request has been put into asynchronous mode.

- boolean isAsyncSupported()
  Checks if this request supports asynchronous operation.

- boolean isSecure()
  Returns a boolean indicating whether this request was made using a secure channel, such as HTTPS.

- void removeAttribute(String name)
  Removes an attribute from this request.

- void setAttribute(String name, Object o)
  Stores an attribute in this request.

- void setCharacterEncoding(String env)
  Overrides the name of the character encoding used in the body of this request.

- AsyncContext startAsync()
  Puts this request into asynchronous mode, and initializes its AsyncContext with the original (unwrapped) ServletRequest and ServletResponse objects.

- AsyncContext startAsync(ServletRequest servletRequest, ServletResponse servletResponse)
  Puts this request into asynchronous mode, and initializes its AsyncContext with the given request and response objects.

ServletResponse Interface

- void flushBuffer()
  Forces any content in the buffer to be written to the client.

- int getBufferSize()
  Returns the actual buffer size used for the response.

- String getCharacterEncoding()
  Returns the name of the character encoding (MIME charset) used for the body sent in this response.

- String getContentType()
  Returns the content type used for the MIME body sent in this response.

- Locale getLocale()
  Returns the locale specified for this response using the setLocale(java.util.Locale) method.

- ServletOutputStream getOutputStream()
  Returns a ServletOutputStream suitable for writing binary data in the response.

- PrintWriter getWriter()
  Returns a PrintWriter object that can send character text to the client.

- boolean isCommitted()
  Returns a boolean indicating if the response has been committed.

- void reset()
  Clears any data that exists in the buffer as well as the status code and headers.

- void resetBuffer()
  Clears the content of the underlying buffer in the response without clearing headers or status code.

- void setBufferSize(int size)
  Sets the preferred buffer size for the body of the response.

- void setCharacterEncoding(java.lang.String charset)
  Sets the character encoding (MIME charset) of the response being sent to the client, for example, to UTF-8.

- void setContentLength(int len)
  Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.

- void setContentType(java.lang.String type)
  Sets the content type of the response being sent to the client, if the response has not been committed yet.

- void setLocale(java.util.Locale loc)
  Sets the locale of the response, if the response has not been committed yet.

HttpServlet Class

- javax.servlet.http.HttpServlet

  ```
  public abstract class HttpServlet extends GenericServlet {

  }
  ```

Constructors

- HttpServlet()

Methods

- protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
  Called by the server (via the service method) to allow a servlet to handle a DELETE request.

- **protected void doGet(HttpServletRequest req, HttpServletResponse resp)**
  Called by the server (via the service method) to allow a servlet to handle a GET request.

- protected void doHead(HttpServletRequest req, HttpServletResponse resp)
  Receives an HTTP HEAD request from the protected service method and handles the request.

- protected void doOptions(HttpServletRequest req, HttpServletResponse resp)
  Called by the server (via the service method) to allow a servlet to handle a OPTIONS request.

- **protected void doPost(HttpServletRequest req, HttpServletResponse resp)**
  Called by the server (via the service method) to allow a servlet to handle a POST request.

- protected void doPut(HttpServletRequest req, HttpServletResponse resp)
  Called by the server (via the service method) to allow a servlet to handle a PUT request.

- protected void doTrace(HttpServletRequest req, HttpServletResponse resp)
  Called by the server (via the service method) to allow a servlet to handle a TRACE request.

- protected long getLastModified(HttpServletRequest req)
  Returns the time the HttpServletRequest object was last modified, in milliseconds since midnight January 1, 1970 GMT.

- protected void service(HttpServletRequest req, HttpServletResponse resp)
  Receives standard HTTP requests from the public service method and dispatches them to the doXXX methods defined in this class.

- void service(ServletRequest req, ServletResponse res)
  Dispatches client requests to the protected service method.

HttpServletRequest Interface

- javax.servlet.http.HttpServletRequest

- public interface **HttpServletRequest** extends **ServletRequest** {

  }

Methods

- boolean authenticate(HttpServletResponse response)
  Use the container login mechanism configured for the ServletContext to authenticate the user making this request.

- String getAuthType()
  Returns the name of the authentication scheme used to protect the servlet.

- String getContextPath()
  Returns the portion of the request URI that indicates the context of the request.

- Cookie[] getCookies()
  Returns an array containing all of the Cookie objects the client sent with this request.

- long getDateHeader(String name)
  Returns the value of the specified request header as a long value that represents a Date object.

- String getHeader(String name)
  Returns the value of the specified request header as a String.

- Enumeration getHeaderNames()
  Returns an enumeration of all the header names this request contains.

- Enumeration getHeaders(String name)
  Returns all the values of the specified request header as an Enumeration of String objects.

- int getIntHeader(String name)
  Returns the value of the specified request header as an int.

- String getMethod()
  Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.

- Part getPart(String name)
  Gets the Part with the given name.

- Collection getParts()
  Gets all the Part components of this request, provided that it is of type multipart/form-data.

- String getPathInfo()
  Returns any extra path information associated with the URL the client sent when it made this request.

- String getPathTranslated()
  Returns any extra path information after the servlet name but before the query string, and translates it to a real path.

- String getQueryString()
  Returns the query string that is contained in the request URL after the path.

- String getRemoteUser()
  Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated.

- String getRequestedSessionId()
  Returns the session ID specified by the client.

- String getRequestURI()
  Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.

- StringBuffer getRequestURL()
  Reconstructs the URL the client used to make the request.

- String getServletPath()
  Returns the part of this request's URL that calls the servlet.

- HttpSession getSession()
  Returns the current session associated with this request, or if the request does not have a session, creates one.

- 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.

- Principal getUserPrincipal()
  Returns a java.security.Principal object containing the name of the current authenticated user.

- boolean isRequestedSessionIdFromCookie()
  Checks whether the requested session ID came in as a cookie.

- boolean isRequestedSessionIdFromURL()
  Checks whether the requested session ID came in as part of the request URL.

- boolean isRequestedSessionIdValid()
  Checks whether the requested session ID is still valid.

- boolean isUserInRole(java.lang.String role)
  Returns a boolean indicating whether the authenticated user is included in the specified logical "role".

- void login(String username, String password)
  Validate the provided username and password in the password validation realm used by the web container login mechanism configured for the ServletContext.

- void logout()
  Establish null as the value returned when getUserPrincipal, getRemoteUser, and getAuthType is called on the request.

ServletResponse Interface

- javax.servlet.http.HttpServletResponse

- public interface HttpServletResponse extends ServletResponse {

  }

Method Details

- void flushBuffer()
  Forces any content in the buffer to be written to the client.

- Int getBufferSize()
  Returns the actual buffer size used for the response.

- String getCharacterEncoding()
  Returns the name of the character encoding (MIME charset) used for the body sent in this response.

- String getContentType()
  Returns the content type used for the MIME body sent in this response.

- Locale getLocale()
  Returns the locale specified for this response using the setLocale(java.util.Locale) method.

- ServletOutputStream getOutputStream()
  Returns a ServletOutputStream suitable for writing binary data in the response.

- **PrintWriter getWriter()**
  Returns a PrintWriter object that can send character text to the client.

- boolean isCommitted()
  Returns a boolean indicating if the response has been committed.

- void reset()
  Clears any data that exists in the buffer as well as the status code and headers.

- void resetBuffer()
  Clears the content of the underlying buffer in the response without clearing headers or status code.

- void setBufferSize(int size)
  Sets the preferred buffer size for the body of the response.

- void setCharacterEncoding(String charset)
  Sets the character encoding (MIME charset) of the response being sent to the client, for example, to UTF-8.

- void setContentLength(int len)
  Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.

- void setContentType(String type)
  Sets the content type of the response being sent to the client, if the response has not been committed yet.

- void setLocale(Locale loc)
  Sets the locale of the response, if the response has not been committed yet.

Response Content Type
Content Type is also known as MIME Type. MIME stand for Multipurpose internet Mail Extension. It is a HTTP header that provides the description about what are you sending to the browser (like send image, text, video etc.).

File	MIME Type	Extension
xml	text/xml	.xml

HTML	text/html	.html
Plaintext File	text/plain	.txt
PDF	application/pdf	.pdf
gif Image	image/gif	.gif
JPEG Image	image/jpeg	.jpeg
PNG Image	image/x-png	.png
MP3 Music File	audio/mpeg	.mp3
MS Word Document	application/msword	.doc
Excel work sheet	application/vnd.ms-sheet	.xls
Power Point Document	application/vnd.ms-powerpoint	.ppt

MIME type have two parts, They are:

- Base name : It is the generic name of file.
- Extension name : It is extension name for specific file type.

EXAMPLE-1

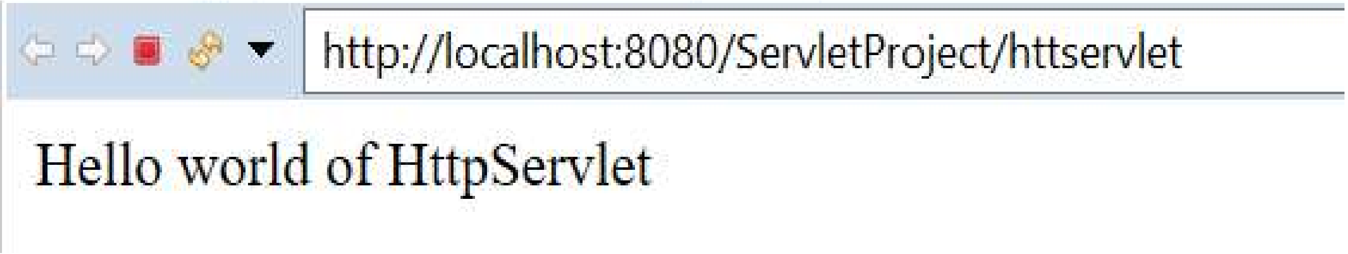
Servlet2_1.java

```
package com.unit2;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Servlet2_1 extends HttpServlet {
        @Override
        protected void  doGet(HttpServletRequest request, HttpServletResponse response) throws IOException
{
                response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                out.println("Hello world of HttpServlet");
        }
}
```

web.xml

```
<servlet>
        <servlet-name>ServletHttpServlet</servlet-name>
        <servlet-class>com.unit2.Servlet2_1</servlet-class>
</servlet>
<servlet-mapping>
        <servlet-name>ServletHttpServlet</servlet-name>
        <url-pattern>/httservlet</url-pattern>
</servlet-mapping>
```

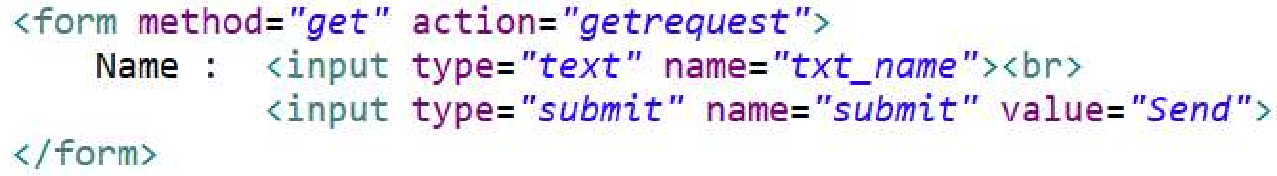
Output



```
Hello world of HttpServlet
```

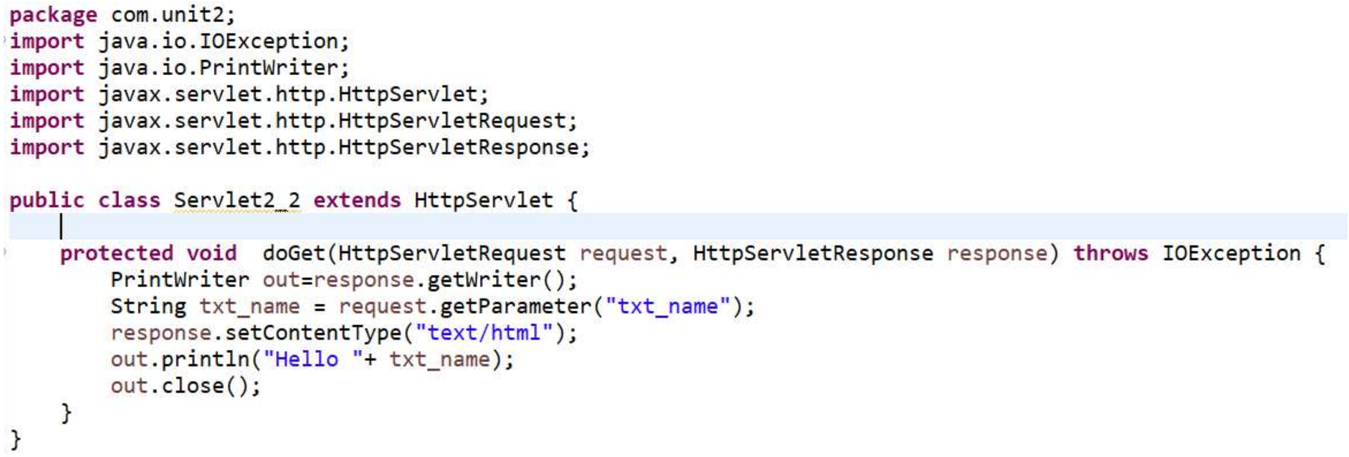
EXAMPLE-2 [WEB FORM]

Form1.html



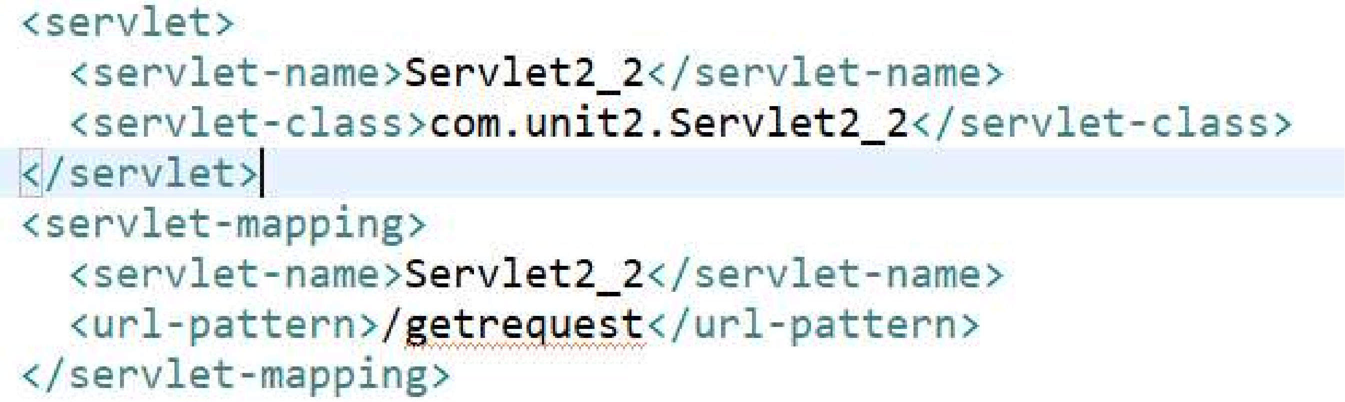
```html
<form method="get" action="getrequest">
    Name :  <input type="text" name="txt_name"><br>
            <input type="submit" name="submit" value="Send">
</form>
```

Servlet2_2.java



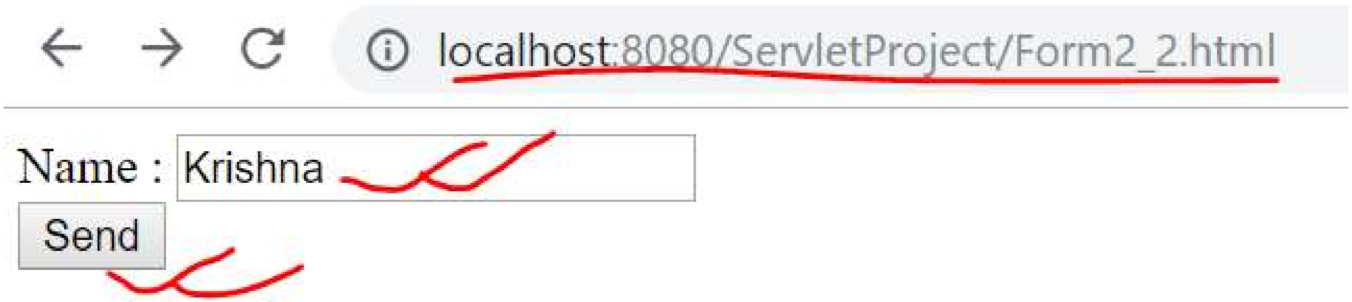
```java
package com.unit2;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Servlet2_2 extends HttpServlet {

    protected void  doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {
        PrintWriter out=response.getWriter();
        String txt_name = request.getParameter("txt_name");
        response.setContentType("text/html");
        out.println("Hello "+ txt_name);
        out.close();
    }
}
```

web.xml



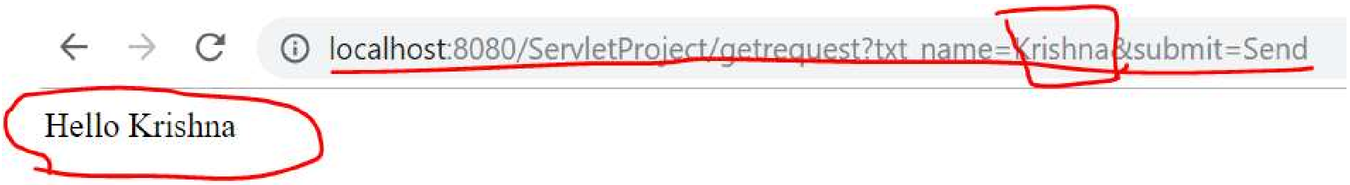
```xml
<servlet>
  <servlet-name>Servlet2_2</servlet-name>
  <servlet-class>com.unit2.Servlet2_2</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>Servlet2_2</servlet-name>
  <url-pattern>/getrequest</url-pattern>
</servlet-mapping>
```

Output:



localhost:8080/ServletProject/Form2_2.html

Name : Krishna



localhost:8080/ServletProject/getrequest?txt_name=Krishna&submit=Send

Hello Krishna

Tasks

- 1. Create a web form which send personal details (id, name, addrerss, email and mobile) and display.
- 2. Create a web form which send two numbers and display sum.
- 3. Create a web form which two numbers, and command to calculation (add, sub, prd, div, pow) and display result after process.
- 4. Create a web form which browse and select image file and upload to web server.
- 5. Create a web form which collect all personal information (data and file) and send, upload to server.

RequestDispatcher

Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server. The servlet container creates the RequestDispatcher object, which is used as a wrapper around a server resource located at a particular path or given by a particular name.

RequestDispatcher Interface

- javax.servlet.RequestDispatcher
- public interface RequestDispatcher{

  }

Fields

- ERROR_EXCEPTION
- ERROR_EXCEPTION_TYPE
- ERROR_MESSAGE
- ERROR_REQUEST_URI
- ERROR_SERVLET_NAME
- ERROR_STATUS_CODE
- FORWARD_CONTEXT_PATH
- FORWARD_PATH_INFO
- FORWARD_QUERY_STRING
- FORWARD_REQUEST_URI

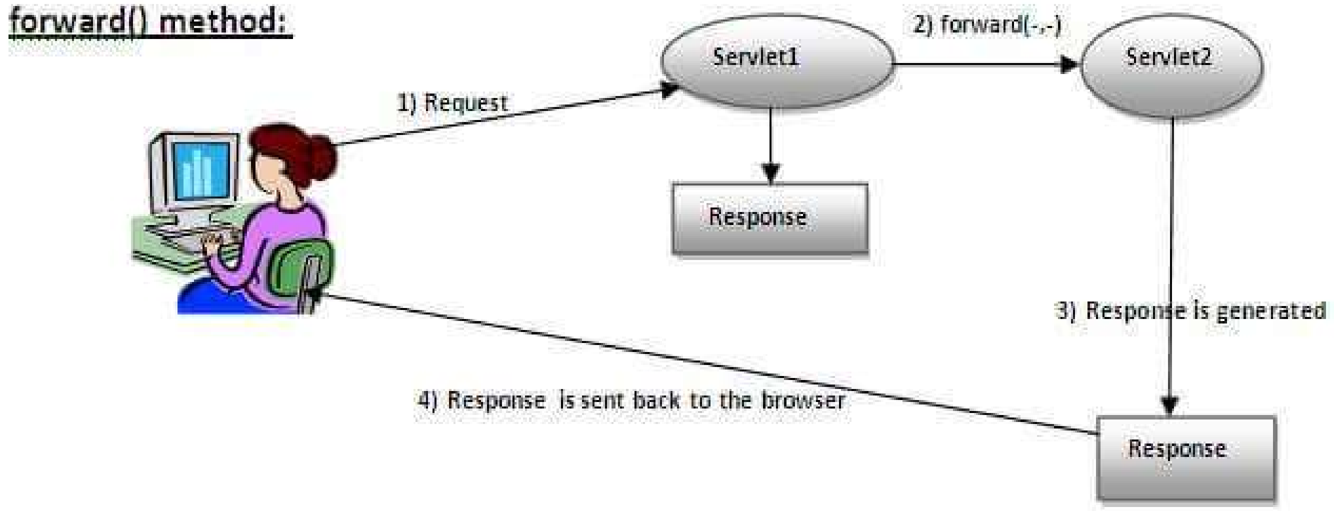
- FORWARD_SERVLET_PATH
- INCLUDE_CONTEXT_PATH
- INCLUDE_PATH_INFO
- INCLUDE_QUERY_STRING
- INCLUDE_REQUEST_URI
- INCLUDE_SERVLET_PATH

Methods

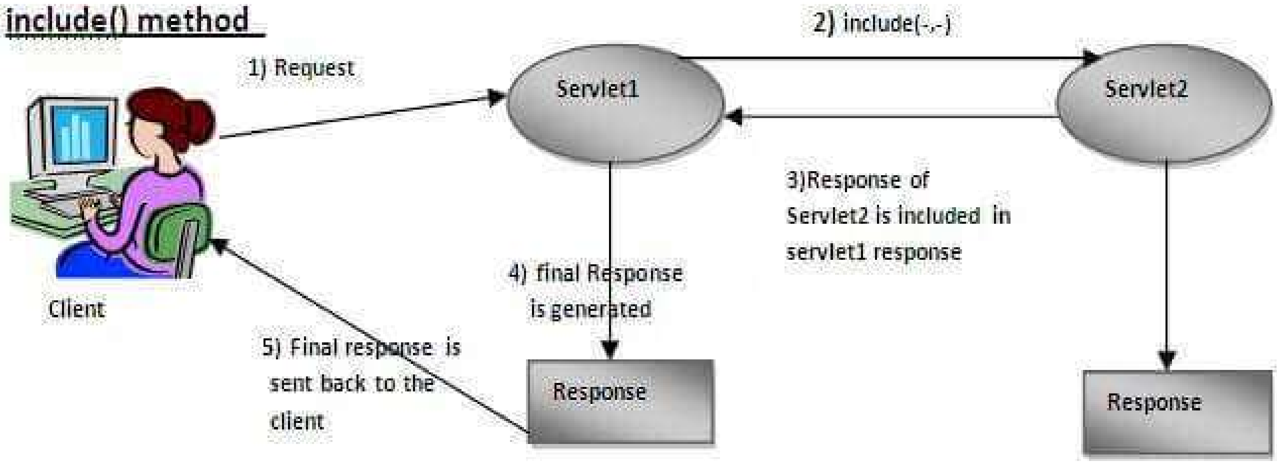
- **void forward(ServletRequest request, ServletResponse response)**
  Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

- **void include(ServletRequest request, ServletResponse response)**
  Includes the content of a resource (servlet, JSP page, HTML file) in the response.

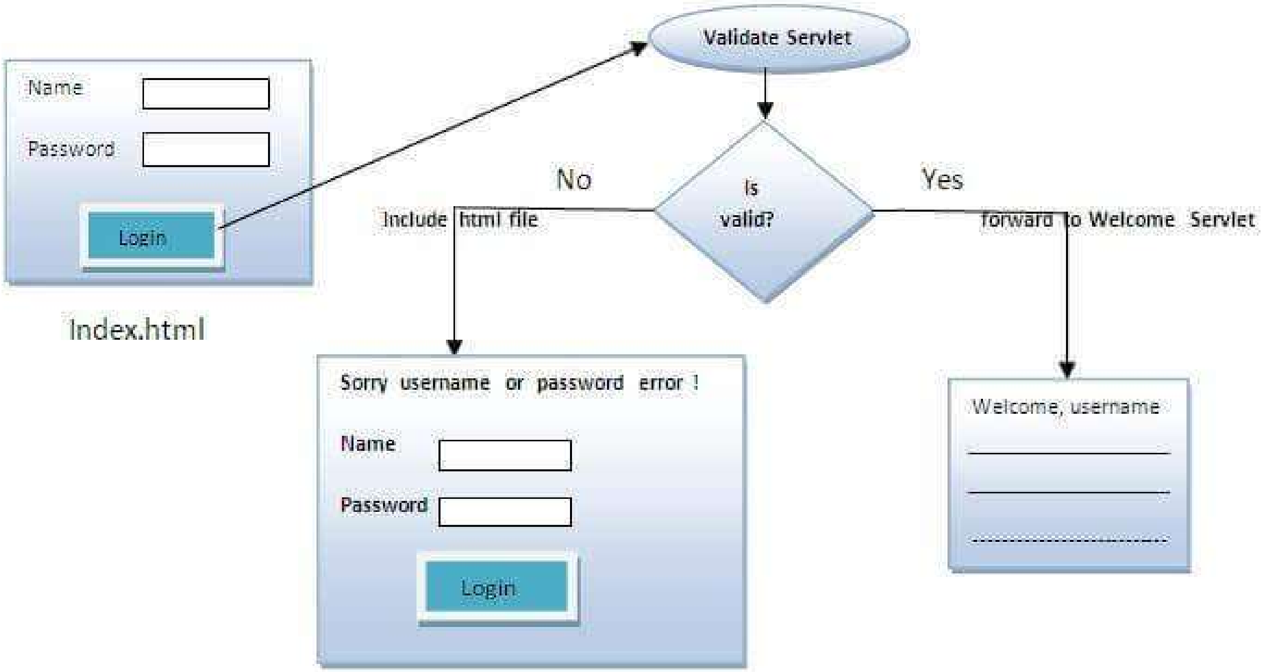
forward() method



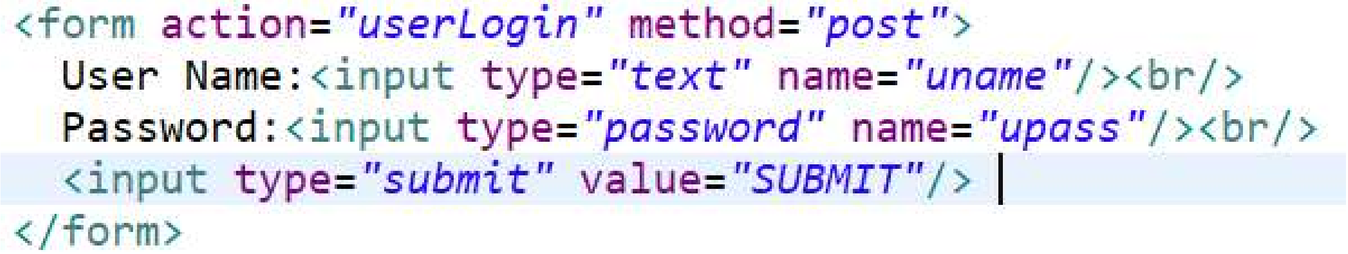
Include() method



Include() and forward()

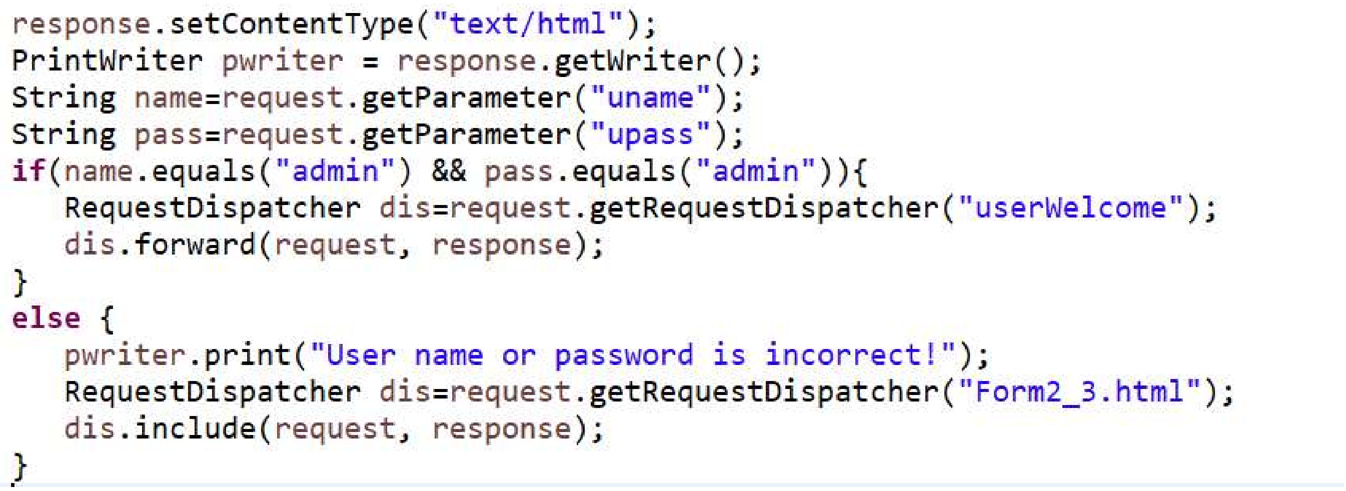


EXAMPLE-3
LoginForm



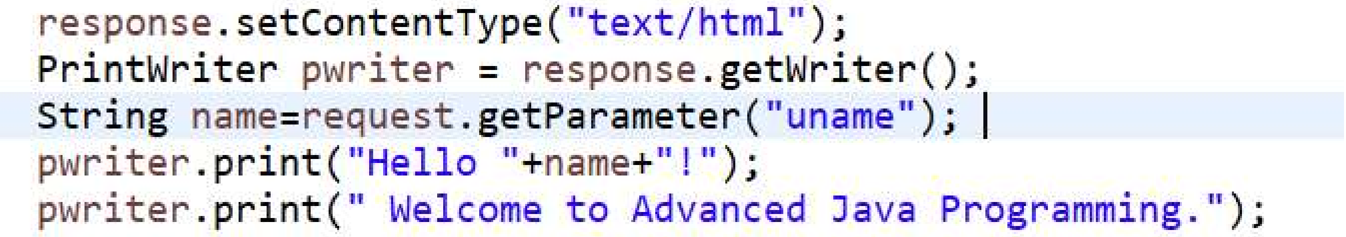
```
<form action="userLogin" method="post">
  User Name:<input type="text" name="uname"/><br/>
  Password:<input type="password" name="upass"/><br/>
  <input type="submit" value="SUBMIT"/>
</form>
```

Validator Sevlet



```
response.setContentType("text/html");
PrintWriter pwriter = response.getWriter();
String name=request.getParameter("uname");
String pass=request.getParameter("upass");
if(name.equals("admin") && pass.equals("admin")){
   RequestDispatcher dis=request.getRequestDispatcher("userWelcome");
   dis.forward(request, response);
}
else {
   pwriter.print("User name or password is incorrect!");
   RequestDispatcher dis=request.getRequestDispatcher("Form2_3.html");
   dis.include(request, response);
}
```

Welcome Servlet



```
response.setContentType("text/html");
PrintWriter pwriter = response.getWriter();
String name=request.getParameter("uname");
pwriter.print("Hello "+name+"!");
pwriter.print(" Welcome to Advanced Java Programming.");
```

Send Redirect

The sendRedirect() method of HttpServletResponse interface can be used to redirect response to another resource, it may be servlet, jsp or html file.

- It accepts relative as well as absolute URL.
- It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.

Method Signature

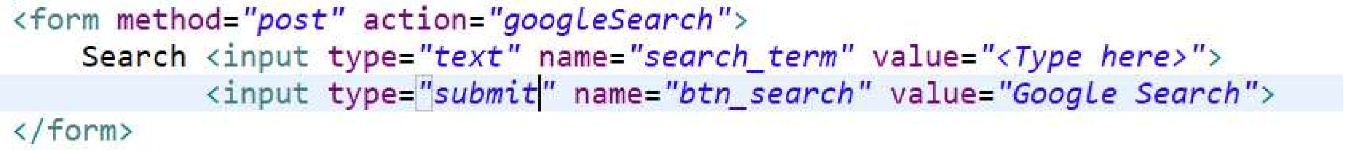
```
public void sendRedirect(String URL)throws IOException;
```

Example

response.sendRedirect(“http://google.com.np”);

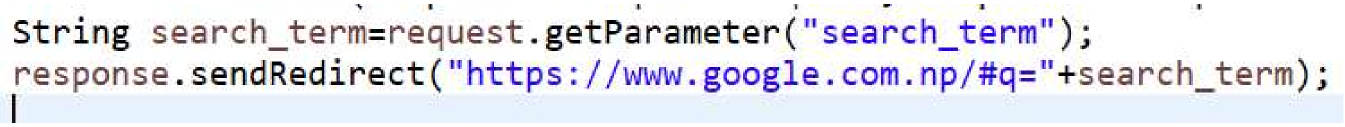
EXAMPLE-4

Search Form



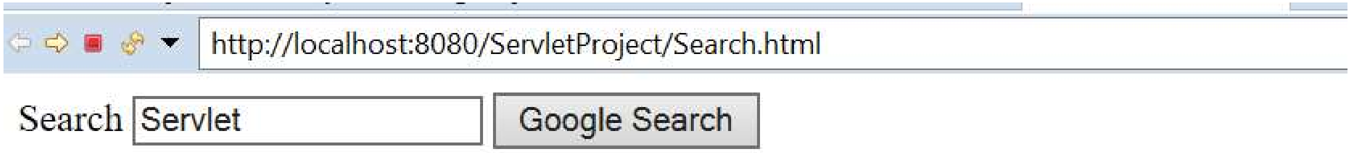
```
<form method="post" action="googleSearch">
    Search <input type="text" name="search_term" value="<Type here>">
           <input type="submit" name="btn_search" value="Google Search">
</form>
```

Servlet



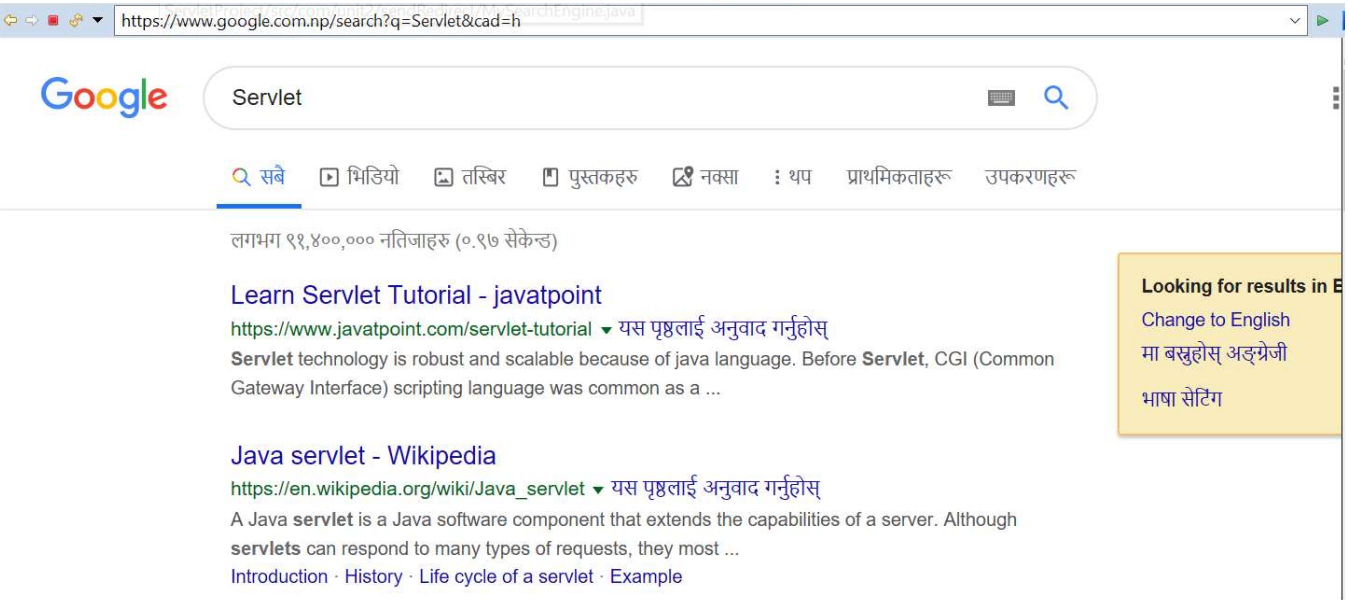
```
String search_term=request.getParameter("search_term");
response.sendRedirect("https://www.google.com.np/#q="+search_term);
```

Output



http://localhost:8080/ServletProject/Search.html

Search Servlet



https://www.google.com.np/search?q=Servlet&cad=h

लगभग ९१,४००,००० नतिजाहरु (०.९७ सेकेन्ड)

Learn Servlet Tutorial - javatpoint
https://www.javatpoint.com/servlet-tutorial ▾ यस पृष्ठलाई अनुवाद गर्नुहोस्
Servlet technology is robust and scalable because of java language. Before **Servlet**, CGI (Common Gateway Interface) scripting language was common as a ...

Java servlet - Wikipedia
https://en.wikipedia.org/wiki/Java_servlet ▾ यस पृष्ठलाई अनुवाद गर्नुहोस्
A Java **servlet** is a Java software component that extends the capabilities of a server. Although **servlets** can respond to many types of requests, they most ...
Introduction · History · Life cycle of a servlet · Example

Looking for results in E
Change to English
मा बस्नुहोस् अङ्ग्रेजी
भाषा सेटिंग

Servlet Config/Context

ServletContext

ServletContext is used to read the configration information defined in context param in web container file(web.xml), This context information can be read by all the servlets of current web application.

ServletConfig

A servlet configuration object used by a servlet container to pass information to a servlet during initialization using init param. Only the specific servlet can access it.

Exploring ServletConfig Interface

- **javax.servlet.ServletConfig**
- **public interface ServletConfig { }**

Methods

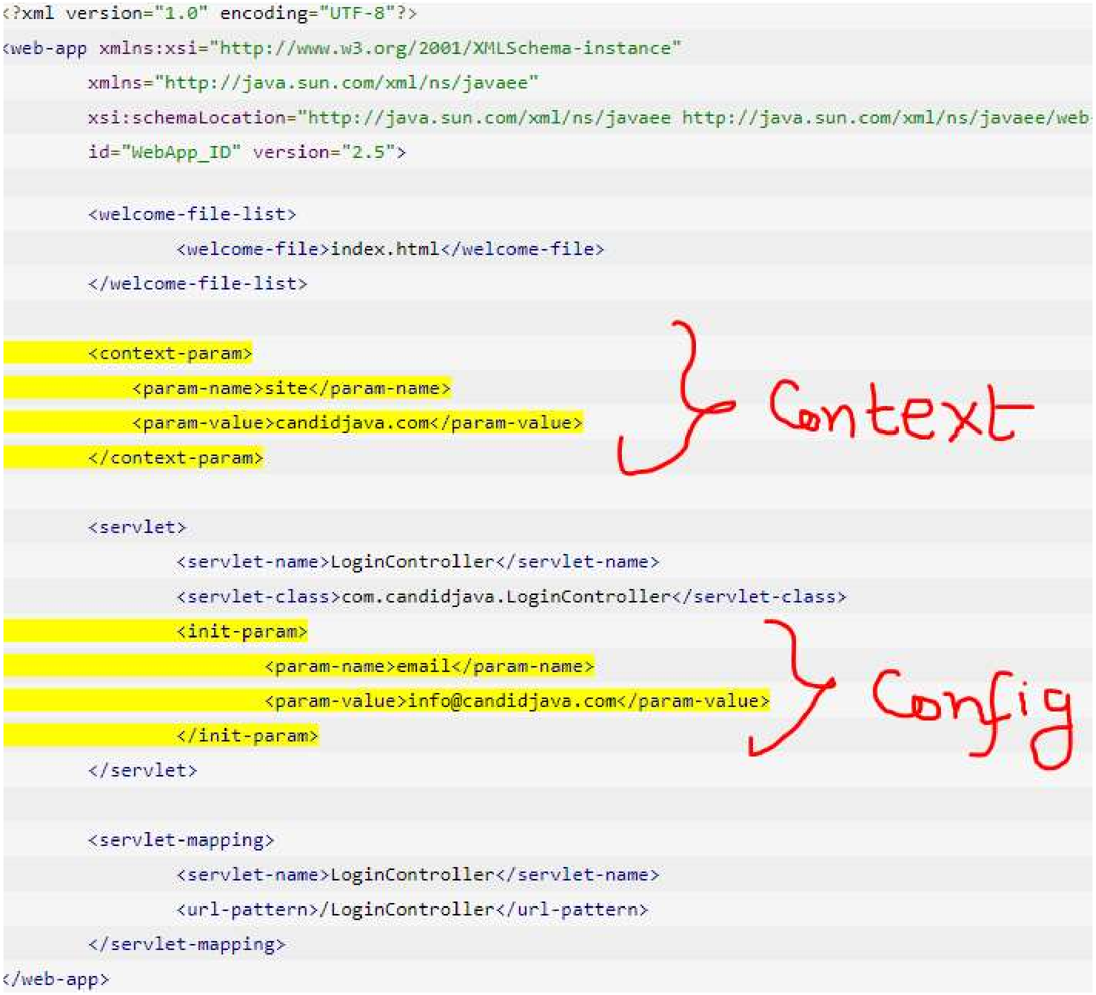
- **String getInitParameter(String name)**
  Gets the value of the initialization parameter with the given name.

- **Enumeration<String> getInitParameterNames()**
  Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.

- **ServletContext getServletContext()**
  Returns a reference to the ServletContext in which the caller is executing.

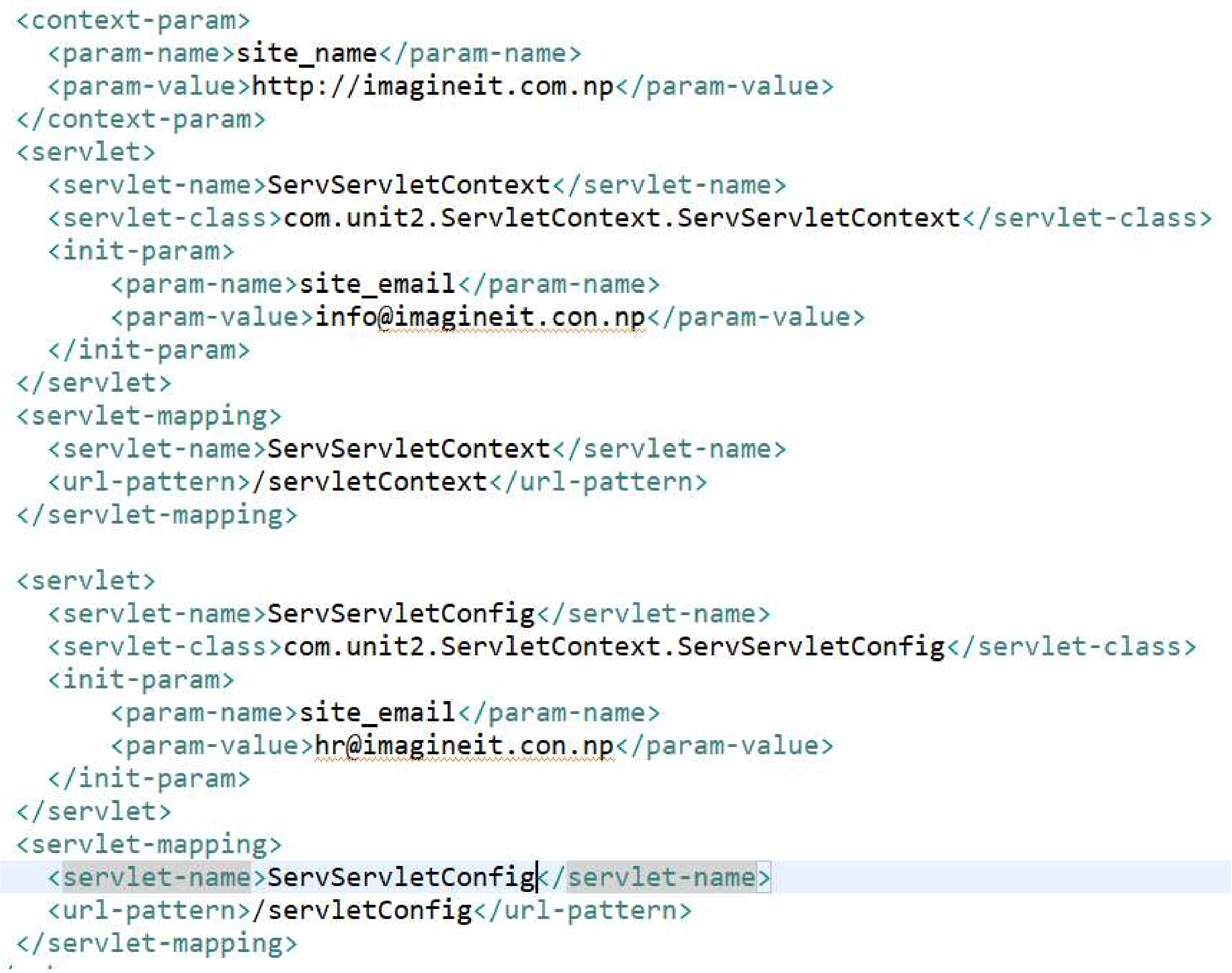
- **String getServletName()**
  Returns the name of this servlet instance.

CASE-1



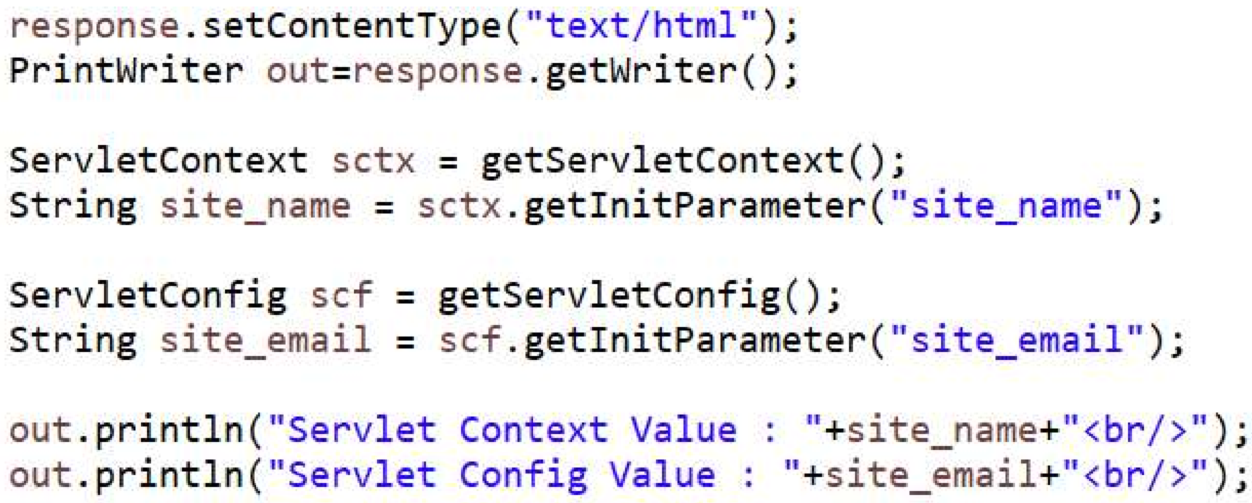
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
        id="WebApp_ID" version="2.5">

        <welcome-file-list>
                <welcome-file>index.html</welcome-file>
        </welcome-file-list>

        <context-param>
            <param-name>site</param-name>
            <param-value>candidjava.com</param-value>
        </context-param>

        <servlet>
                <servlet-name>LoginController</servlet-name>
                <servlet-class>com.candidjava.LoginController</servlet-class>
                <init-param>
                        <param-name>email</param-name>
                        <param-value>info@candidjava.com</param-value>
                </init-param>
        </servlet>

        <servlet-mapping>
                <servlet-name>LoginController</servlet-name>
                <url-pattern>/LoginController</url-pattern>
        </servlet-mapping>
</web-app>
```

EXAMPLE-5 [ServletContext/ServletConfig]
web.xml



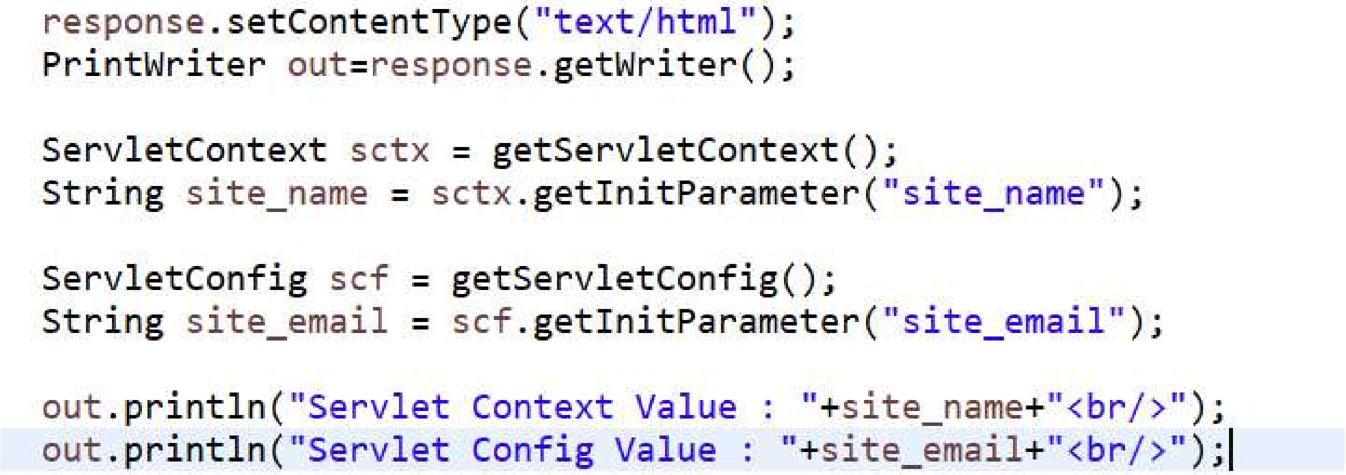
```xml
<context-param>
  <param-name>site_name</param-name>
  <param-value>http://imagineit.com.np</param-value>
</context-param>
<servlet>
  <servlet-name>ServServletContext</servlet-name>
  <servlet-class>com.unit2.ServletContext.ServServletContext</servlet-class>
  <init-param>
      <param-name>site_email</param-name>
      <param-value>info@imagineit.con.np</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>ServServletContext</servlet-name>
  <url-pattern>/servletContext</url-pattern>
</servlet-mapping>

<servlet>
  <servlet-name>ServServletConfig</servlet-name>
  <servlet-class>com.unit2.ServletContext.ServServletConfig</servlet-class>
  <init-param>
      <param-name>site_email</param-name>
      <param-value>hr@imagineit.con.np</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>ServServletConfig</servlet-name>
  <url-pattern>/servletConfig</url-pattern>
</servlet-mapping>
```

ServServletContext



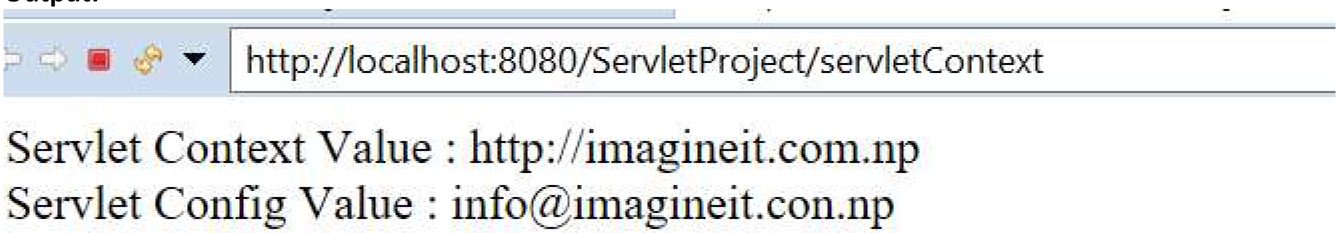
```java
response.setContentType("text/html");
PrintWriter out=response.getWriter();

ServletContext sctx = getServletContext();
String site_name = sctx.getInitParameter("site_name");

ServletConfig scf = getServletConfig();
String site_email = scf.getInitParameter("site_email");

out.println("Servlet Context Value : "+site_name+"<br/>");
out.println("Servlet Config Value : "+site_email+"<br/>");
```

ServServletConfig



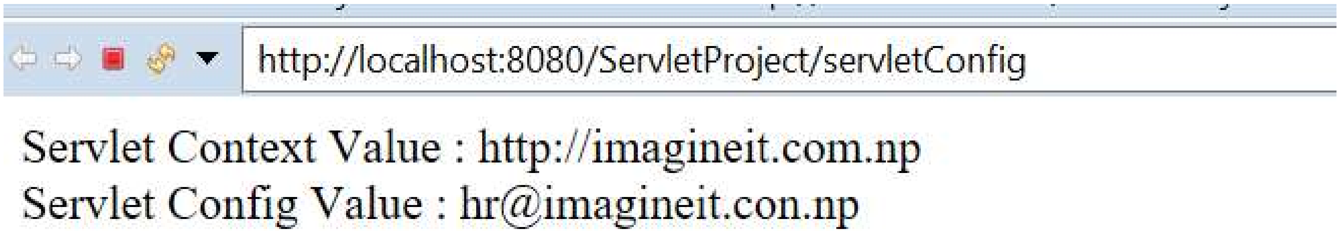
```
response.setContentType("text/html");
PrintWriter out=response.getWriter();

ServletContext sctx = getServletContext();
String site_name = sctx.getInitParameter("site_name");

ServletConfig scf = getServletConfig();
String site_email = scf.getInitParameter("site_email");

out.println("Servlet Context Value : "+site_name+"<br/>");
out.println("Servlet Config Value : "+site_email+"<br/>");
```

Output:



http://localhost:8080/ServletProject/servletContext

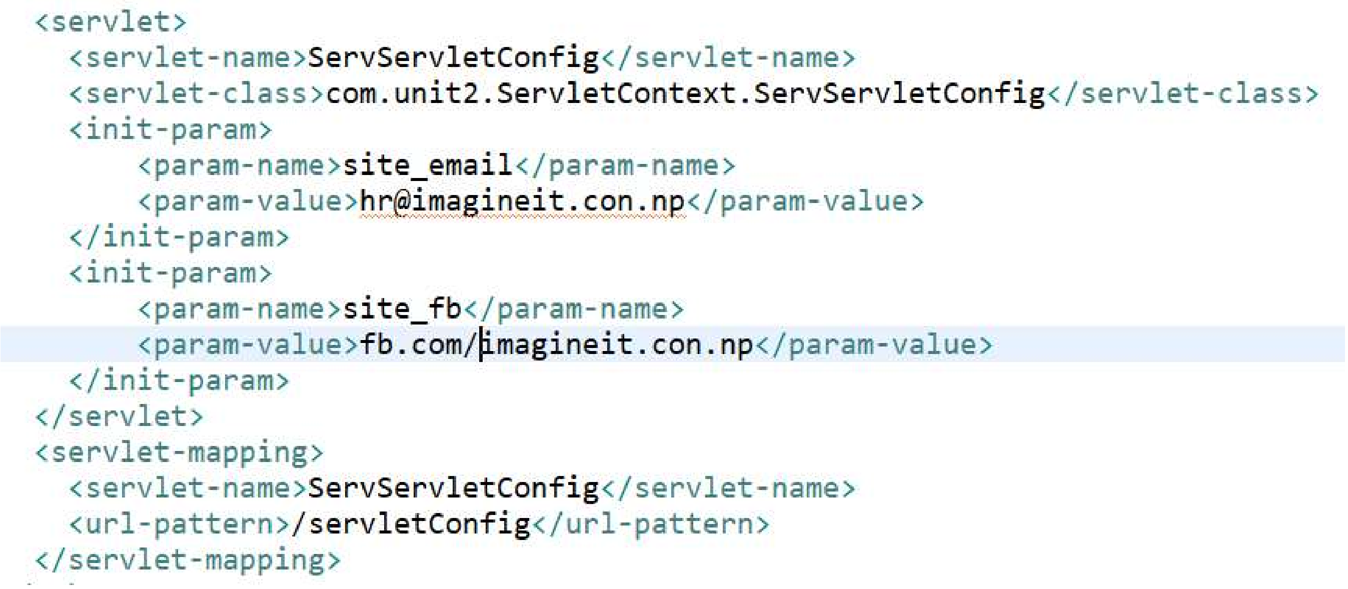
```
Servlet Context Value : http://imagineit.com.np
Servlet Config Value : info@imagineit.con.np
```



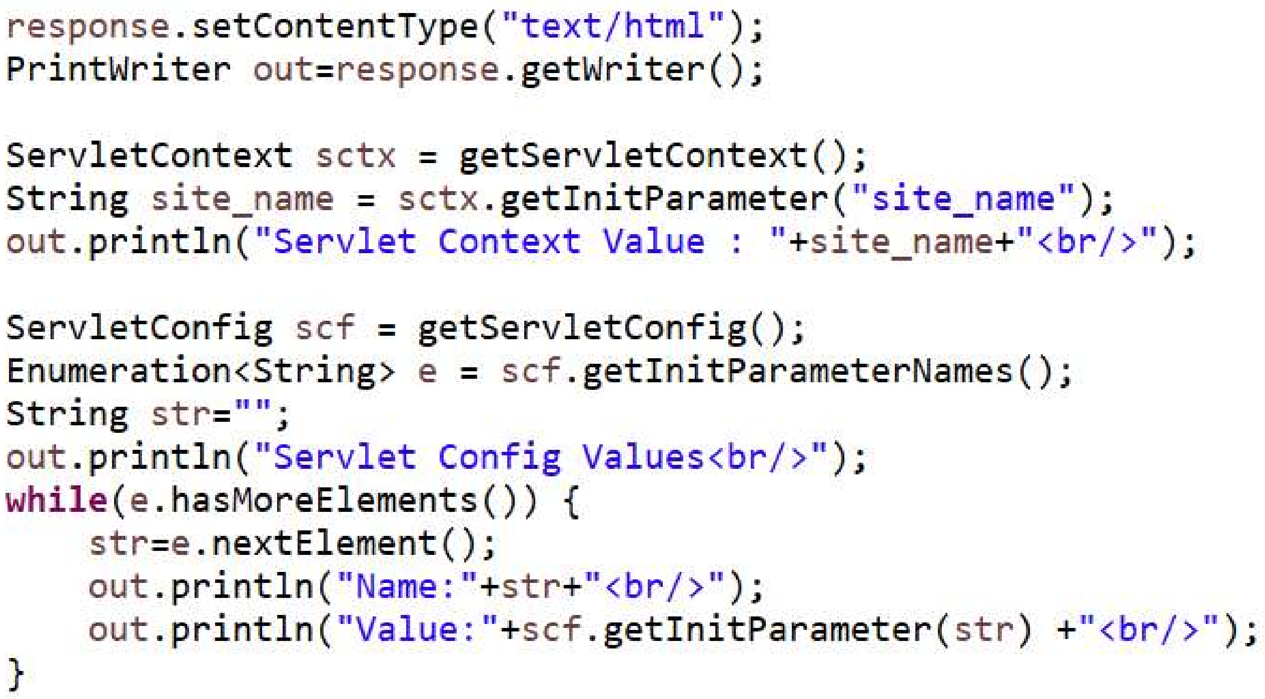
http://localhost:8080/ServletProject/servletConfig

```
Servlet Context Value : http://imagineit.com.np
Servlet Config Value : hr@imagineit.con.np
```

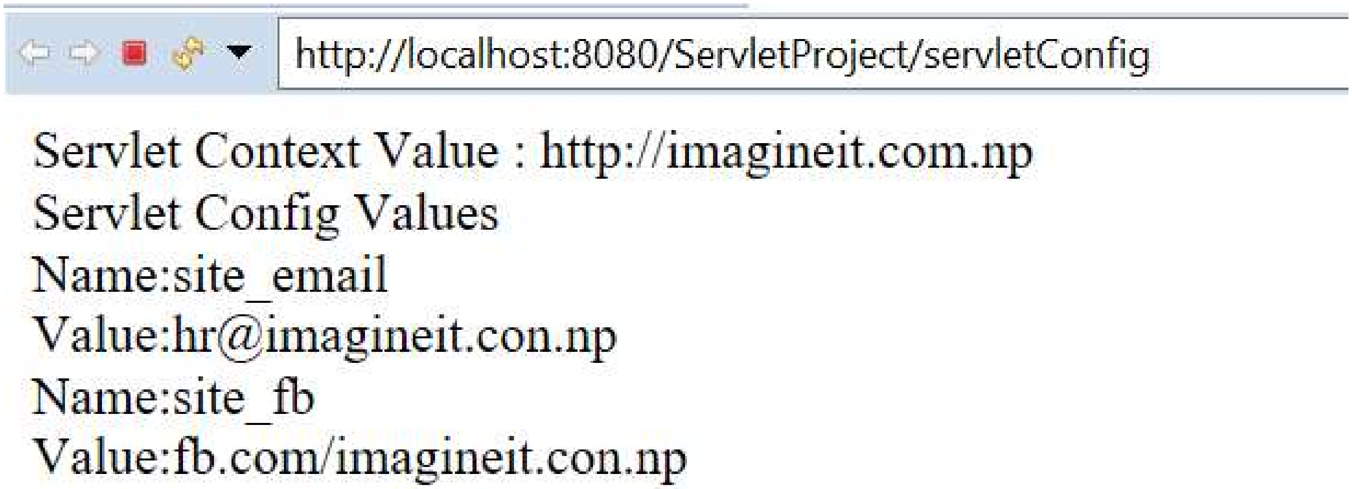
ServletConfig-2



```
<servlet>
  <servlet-name>ServServletConfig</servlet-name>
  <servlet-class>com.unit2.ServletContext.ServServletConfig</servlet-class>
  <init-param>
      <param-name>site_email</param-name>
      <param-value>hr@imagineit.con.np</param-value>
  </init-param>
  <init-param>
      <param-name>site_fb</param-name>
      <param-value>fb.com/imagineit.con.np</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>ServServletConfig</servlet-name>
  <url-pattern>/servletConfig</url-pattern>
</servlet-mapping>
```



```
response.setContentType("text/html");
PrintWriter out=response.getWriter();

ServletContext sctx = getServletContext();
String site_name = sctx.getInitParameter("site_name");
out.println("Servlet Context Value : "+site_name+"<br/>");

ServletConfig scf = getServletConfig();
Enumeration<String> e = scf.getInitParameterNames();
String str="";
out.println("Servlet Config Values<br/>");
while(e.hasMoreElements()) {
    str=e.nextElement();
    out.println("Name:"+str+"<br/>");
    out.println("Value:"+scf.getInitParameter(str) +"<br/>");
}
```



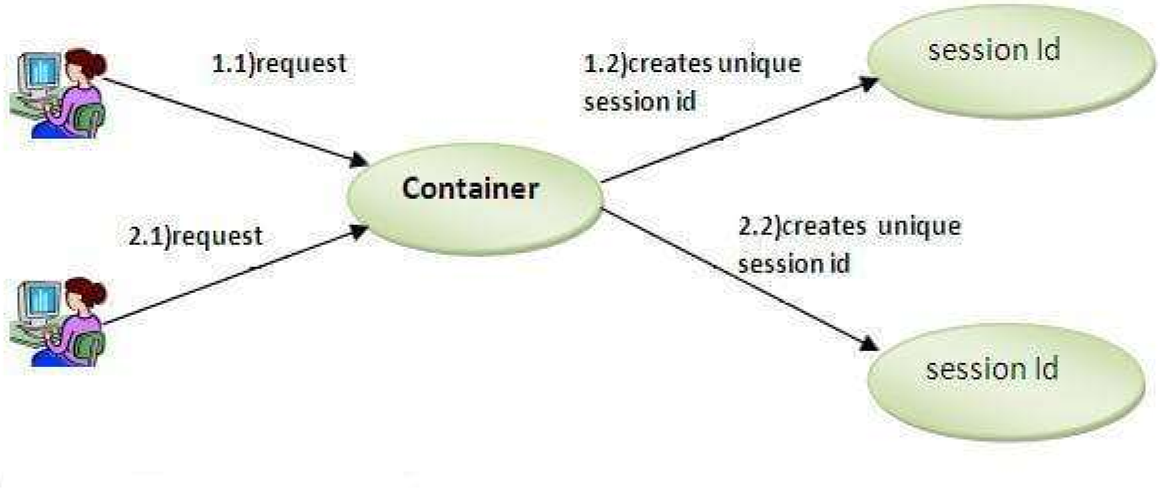
http://localhost:8080/ServletProject/servletConfig

```
Servlet Context Value : http://imagineit.com.np
Servlet Config Values
Name:site_email
Value:hr@imagineit.con.np
Name:site_fb
Value:fb.com/imagineit.con.np
```

HttpSession

A session contains information specific to a particular user across the whole application. When a user enters into a website (or an online application) for the first time HttpSession is obtained via request.getSession(), the user is given a unique ID to identify his session. This unique ID can be stored into a cookie or in a request parameter.

The HttpSession stays alive until it has not been used for more than the timeout value specified in tag in deployment descriptor file(web.xml). The default timeout value is 30 minutes, this is used if you don't specify the value in tag. This means that when the user doesn't visit web application time specified, the session is destroyed by servlet container.



HttpSession Interface

- javax.servlet.http.HttpSession
- public interface HttpSession { }

Methods

- **Object getAttribute(String name)**
  Returns the object bound with the specified name in this session, or null if no object is bound under the name.

- **Enumeration<String> getAttributeNames()**
  Returns an Enumeration of String objects containing the names of all the objects bound to this session.

- **long getCreationTime()**
  Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.

- **String getId()**
  Returns a string containing the unique identifier assigned to this session.

- **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, and marked by the time the container received the request.

- **int getMaxInactiveInterval()**
  Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.

- **ServletContext getServletContext()**
  Returns the ServletContext to which this session belongs.

- **void invalidate()**
  Invalidates this session then unbinds any objects bound to it.

- **boolean isNew()**
  Returns true if the client does not yet know about the session or if the client chooses not to join the session.

- **void removeAttribute(String name)**
  Removes the object bound with the specified name from this session.

- **void setAttribute(String name, Object value)**
  Binds an object to this session, using the name specified.

- **void setMaxInactiveInterval(int interval)**
  Specifies the time, in seconds, between client requests before the servlet container will invalidate this session.

Creating Session

```
HttpSession session = req.getSession();
```

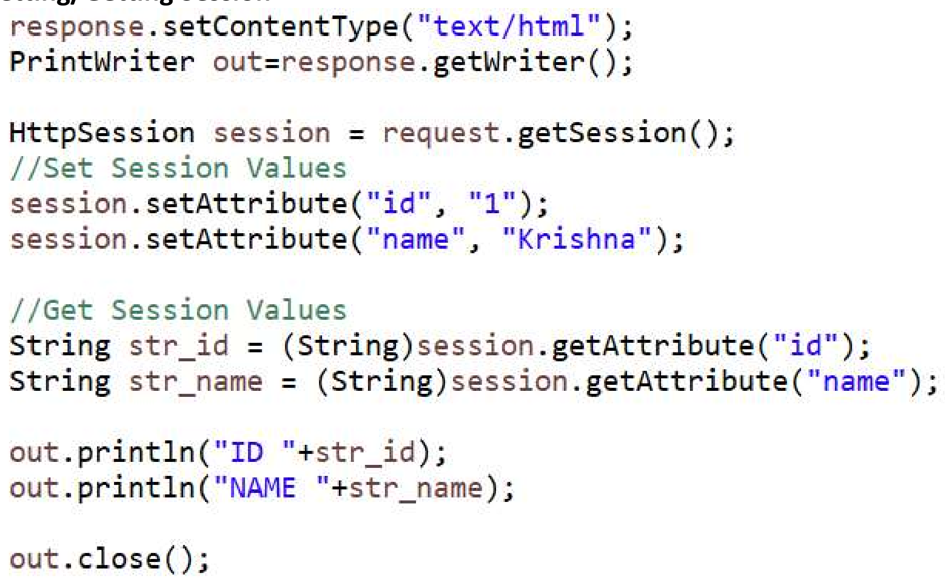
Setting Values on Session

```
session.setAttribute("userName", "admin");
session.setAttribute("userEmail", "info@gmail.com");
session.setAttribute("userMobile", "9851123456");
```

Getting Values from Session

```
String userName = (String) session.getAttribute("userName");
String userEmailId = (String) session.getAttribute("userEmail");
String userAge = (String) session.getAttribute("userMobile");
```

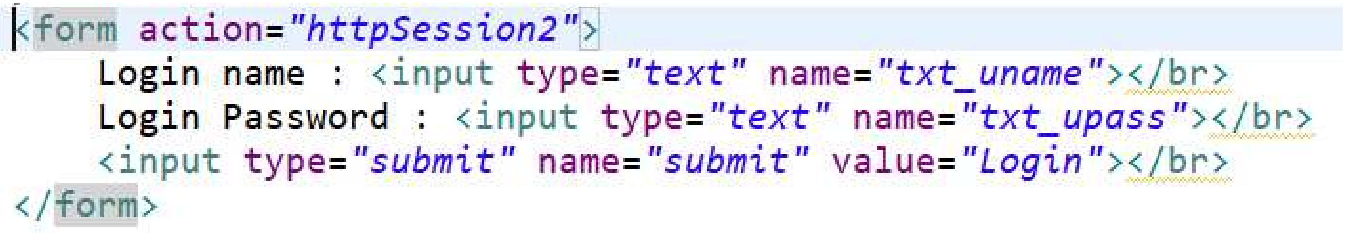
Setting/Getting Session



```
response.setContentType("text/html");
PrintWriter out=response.getWriter();

HttpSession session = request.getSession();
//Set Session Values
session.setAttribute("id", "1");
session.setAttribute("name", "Krishna");

//Get Session Values
String str_id = (String)session.getAttribute("id");
String str_name = (String)session.getAttribute("name");

out.println("ID "+str_id);
out.println("NAME "+str_name);

out.close();
```

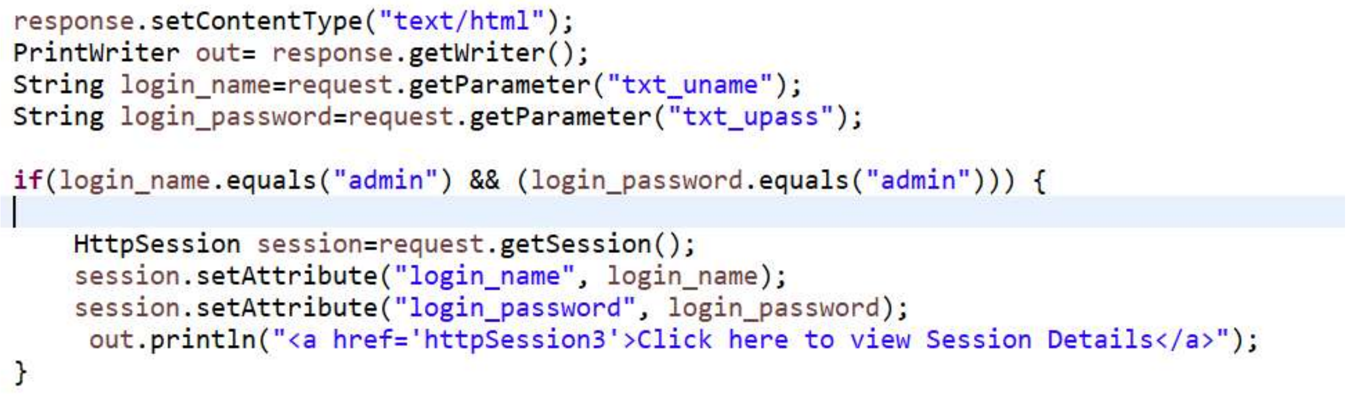
EXAMPLE-6

LoginForm



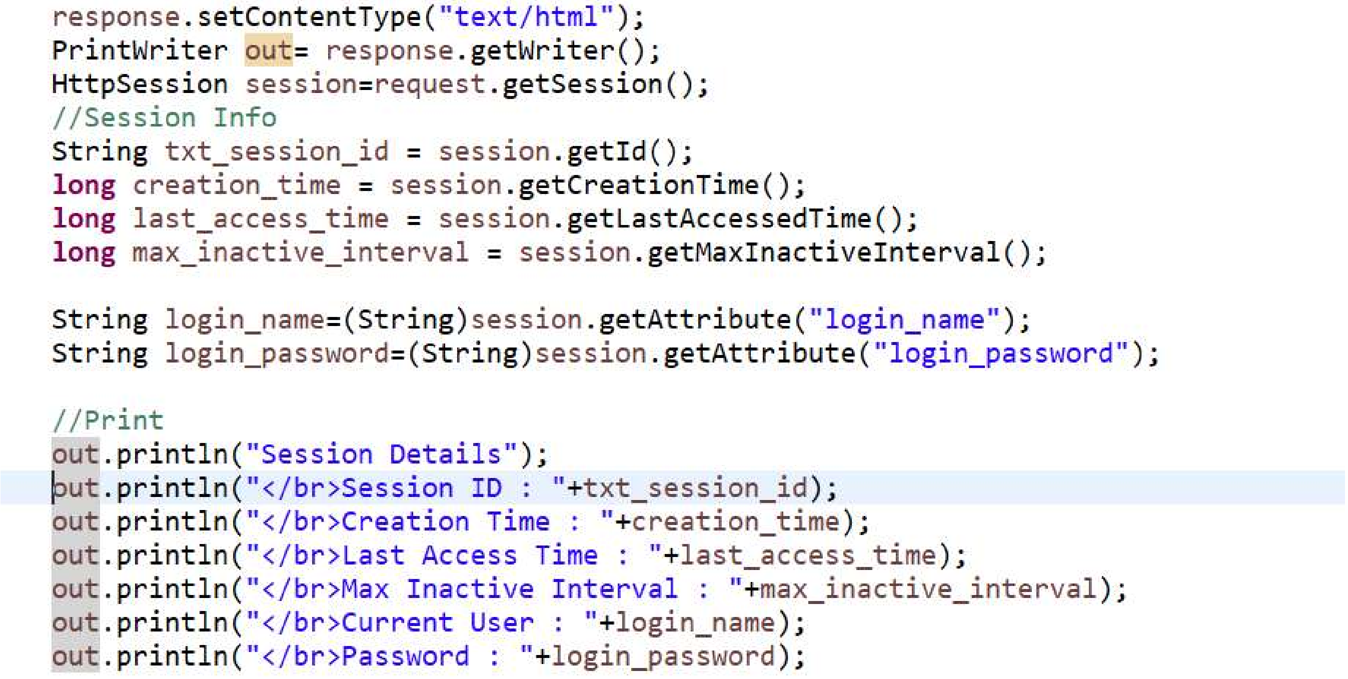
```
<form action="httpSession2">
    Login name : <input type="text" name="txt_uname"></br>
    Login Password : <input type="text" name="txt_upass"></br>
    <input type="submit" name="submit" value="Login"></br>
</form>
```

LoginServlet



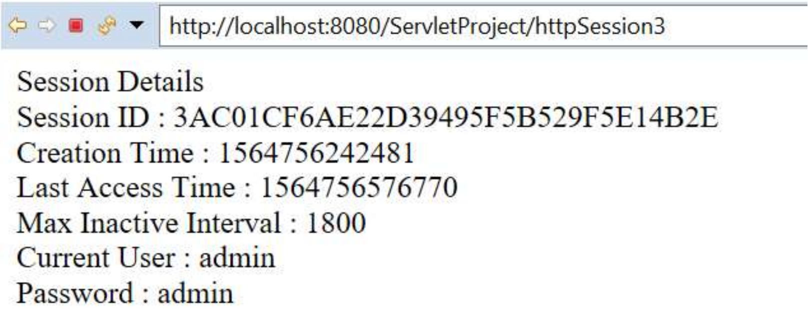
```
response.setContentType("text/html");
PrintWriter out= response.getWriter();
String login_name=request.getParameter("txt_uname");
String login_password=request.getParameter("txt_upass");

if(login_name.equals("admin") && (login_password.equals("admin"))) {

    HttpSession session=request.getSession();
    session.setAttribute("login_name", login_name);
    session.setAttribute("login_password", login_password);
     out.println("<a href='httpSession3'>Click here to view Session Details</a>");
}
```

DisplayDetails



```
response.setContentType("text/html");
PrintWriter out= response.getWriter();
HttpSession session=request.getSession();
//Session Info
String txt_session_id = session.getId();
long creation_time = session.getCreationTime();
long last_access_time = session.getLastAccessedTime();
long max_inactive_interval = session.getMaxInactiveInterval();

String login_name=(String)session.getAttribute("login_name");
String login_password=(String)session.getAttribute("login_password");

//Print
out.println("Session Details");
out.println("</br>Session ID : "+txt_session_id);
out.println("</br>Creation Time : "+creation_time);
out.println("</br>Last Access Time : "+last_access_time);
out.println("</br>Max Inactive Interval : "+max_inactive_interval);
out.println("</br>Current User : "+login_name);
out.println("</br>Password : "+login_password);
```

Output:

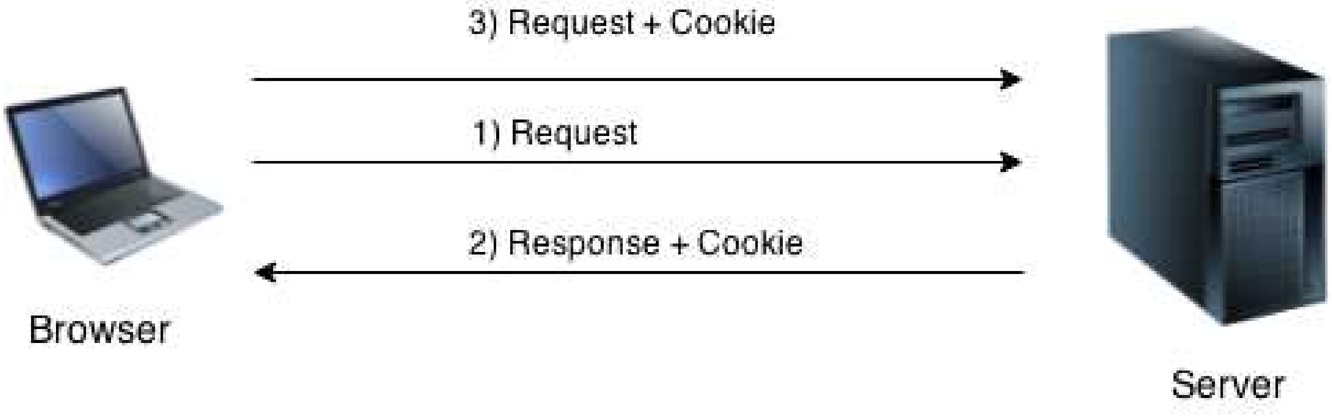


http://localhost:8080/ServletProject/httpSession3

```
Session Details
Session ID : 3AC01CF6AE22D39495F5B529F5E14B2E
Creation Time : 1564756242481
Last Access Time : 1564756576770
Max Inactive Interval : 1800
Current User : admin
Password : admin
```

HttpCookies

A cookie is a small piece of information as a text file stored on client's machine by a web application. As HTTP is a stateless protocol so there is no way to identify that it is a new user or previous user for every new request. In case of cookie a text file with small piece of information is added to the response of first request. They are stored on client's machine. Now when a new request comes cookie is by default added with the request. With this information we can identify that it is a new user or a previous user.



Types of cookies

1. Session cookies/Non-persistent cookies

These types of cookies are session dependent i.e. they are accessible as long as session is open and they are lost when session is closed by exiting from the web application.

2. Permanent cookies/Persistent cookies

These types of cookies are session independent i.e. they are not lost when session is closed by exiting from the web application. They are lost when they expire.

Advantages of cookies

- They are stored on client side so don't need any server resource.
- Easy technique for session management.

Disadvantages of cookies:

- Cookies can be disabled from the browser.
- Security risk is there because cookies exist as a text file so any one can open and read user's information.

Exploring Cookie Class

- javax.servlet.http.Cookie
- public class Cookie { }

Constructor

- Cookie(String name, String value)

Methods

- **Object clone()**
  Overrides the standard java.lang.Object.clone method to return a copy of this cookie.

- **void setComment(String purpose)**
  Specifies a comment that describes a cookie's purpose.

- **String getComment()**
  Returns the comment describing the purpose of this cookie, or null if the cookie has no comment.

- **void setDomain(String pattern)**
  Specifies the domain within which this cookie should be presented.

- **void setMaxAge(int expiry)**
  Sets the maximum age of the cookie in seconds.

- **String getDomain()**
  Returns the domain name set for this cookie.

- **int getMaxAge()**
  Returns the maximum age of the cookie, specified in seconds, By default, -1 indicating the cookie will persist until browser shutdown.

- **String getName()**
  Returns the name of the cookie.

- **void setPath(String uri)**
  Specifies a path for the cookie to which the client should return the cookie.

- **String getPath()**
  Returns the path on the server to which the browser returns this cookie.

- **void setSecure(boolean flag)**
  Indicates to the browser whether the cookie should only be sent using a secure protocol, such as HTTPS or SSL.

- **boolean getSecure()**
  Returns true if the browser is sending cookies only over a secure protocol, or false if the browser can send cookies using any protocol.

- **void setValue(java.lang.String newValue)**
  Assigns a new value to a cookie after the cookie is created.

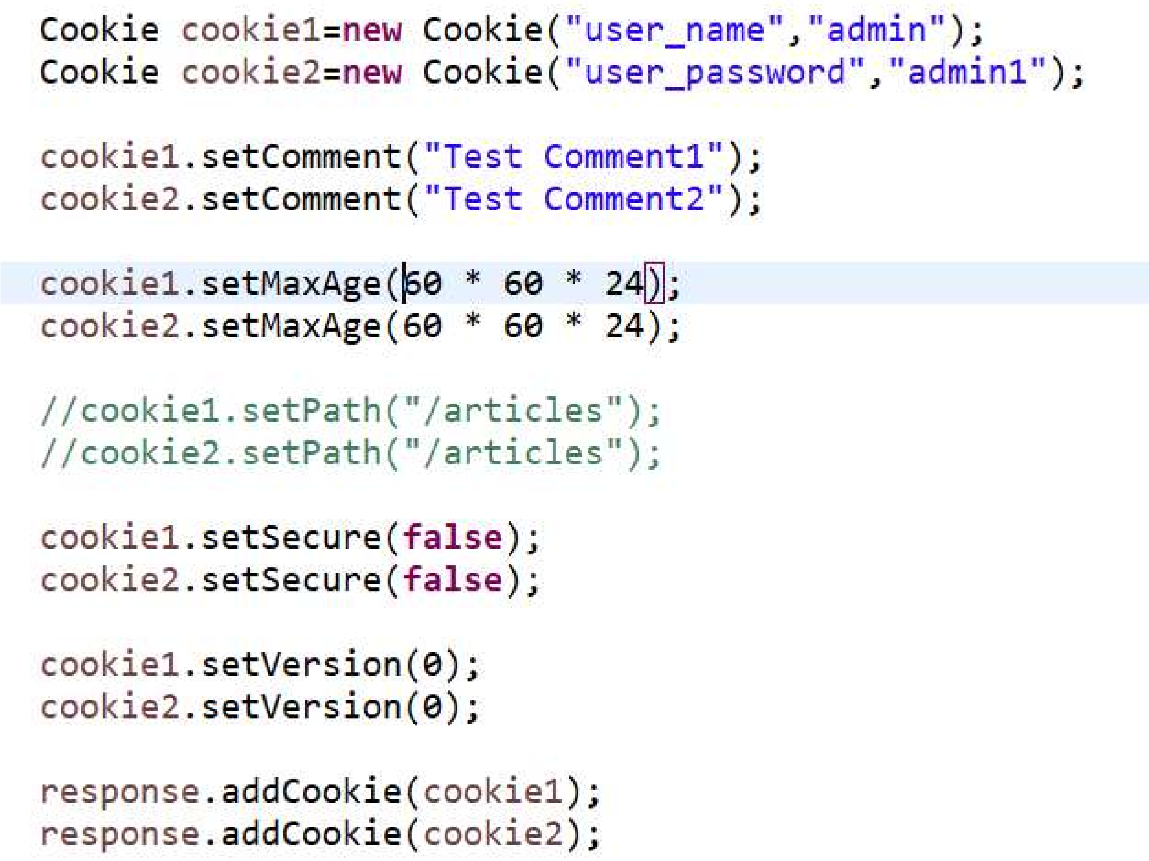
- **String getValue()**
  Returns the value of the cookie.

- **void setVersion(int v)**

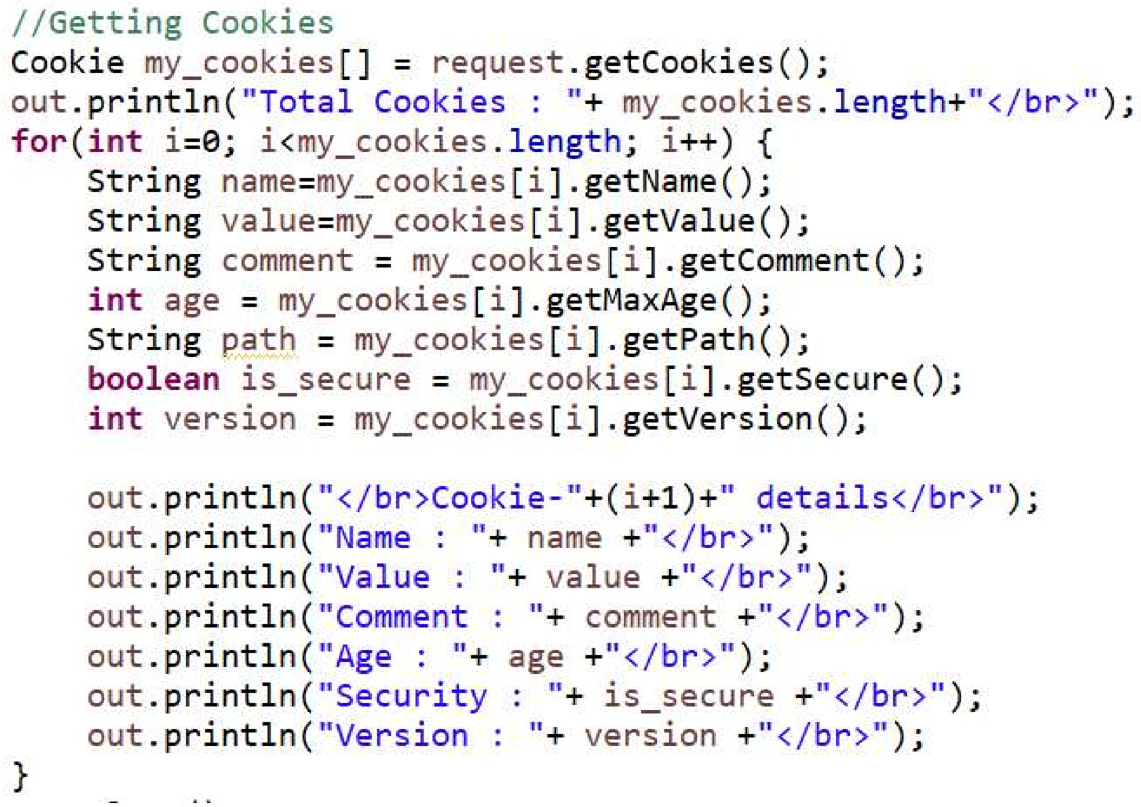
Sets the version of the cookie protocol this cookie complies with.

- **int getVersion()**
  Returns the version of the protocol this cookie complies with.

Setting/Getting Cookie



```
Cookie cookie1=new Cookie("user_name","admin");
Cookie cookie2=new Cookie("user_password","admin1");

cookie1.setComment("Test Comment1");
cookie2.setComment("Test Comment2");

cookie1.setMaxAge(60 * 60 * 24);
cookie2.setMaxAge(60 * 60 * 24);

//cookie1.setPath("/articles");
//cookie2.setPath("/articles");

cookie1.setSecure(false);
cookie2.setSecure(false);

cookie1.setVersion(0);
cookie2.setVersion(0);

response.addCookie(cookie1);
response.addCookie(cookie2);
```



```
//Getting Cookies
Cookie my_cookies[] = request.getCookies();
out.println("Total Cookies : "+ my_cookies.length+"</br>");
for(int i=0; i<my_cookies.length; i++) {
    String name=my_cookies[i].getName();
    String value=my_cookies[i].getValue();
    String comment = my_cookies[i].getComment();
    int age = my_cookies[i].getMaxAge();
    String path = my_cookies[i].getPath();
    boolean is_secure = my_cookies[i].getSecure();
    int version = my_cookies[i].getVersion();

    out.println("</br>Cookie-"+(i+1)+" details</br>");
    out.println("Name : "+ name +"</br>");
    out.println("Value : "+ value +"</br>");
    out.println("Comment : "+ comment +"</br>");
    out.println("Age : "+ age +"</br>");
    out.println("Security : "+ is_secure +"</br>");
    out.println("Version : "+ version +"</br>");
}
```

URL Rewriting

URL rewriting is a way of appending data at the end of URL. Data is appended in name value pair form. Multiple parameters can be appended in one URL with name value pairs.

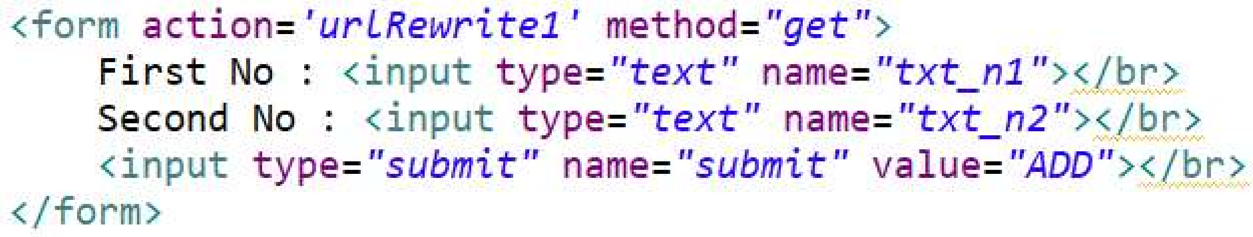
Advantages

- As data is appended in the URL it is easy to debug.
- It is browser independent.

Disadvantages

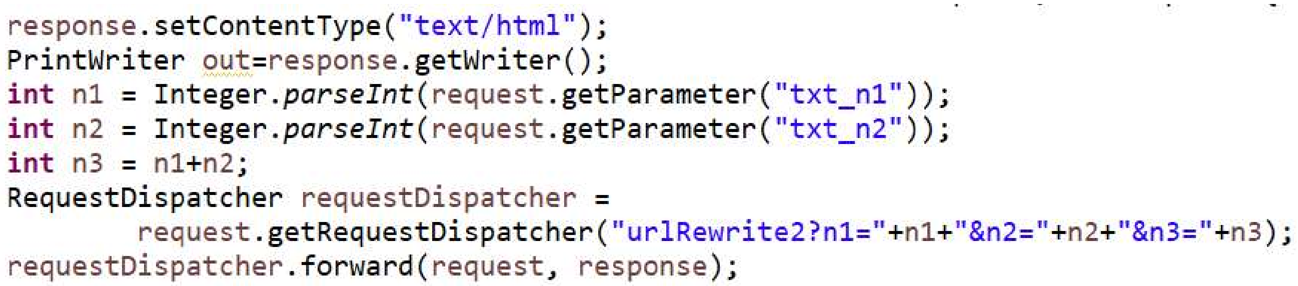
- Not secure because data is appended in the URL.
- Can't append large no. of parameters because URL length is limited.

Web Form



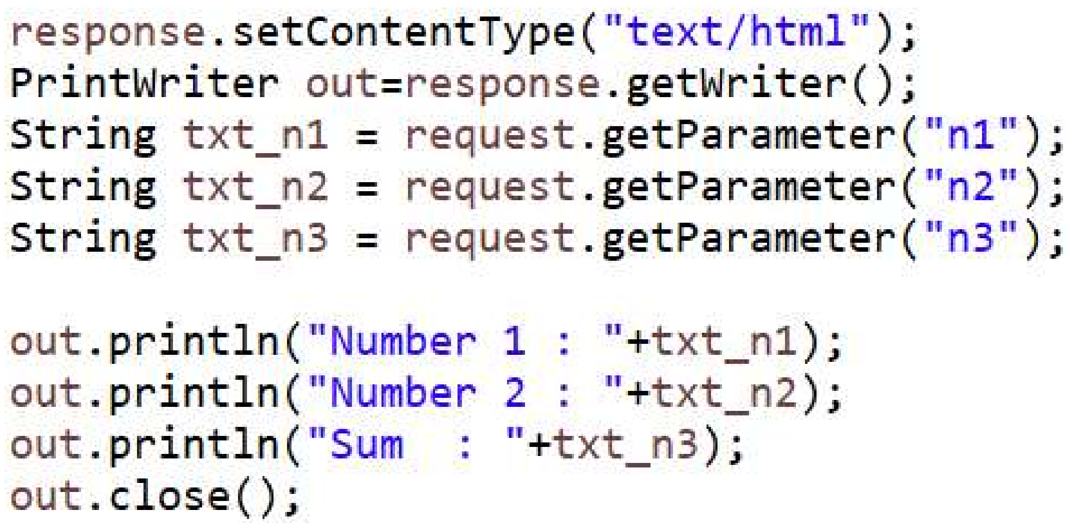
```
<form action='urlRewrite1' method="get">
    First No : <input type="text" name="txt_n1"></br>
    Second No : <input type="text" name="txt_n2"></br>
    <input type="submit" name="submit" value="ADD"></br>
</form>
```

Process Data



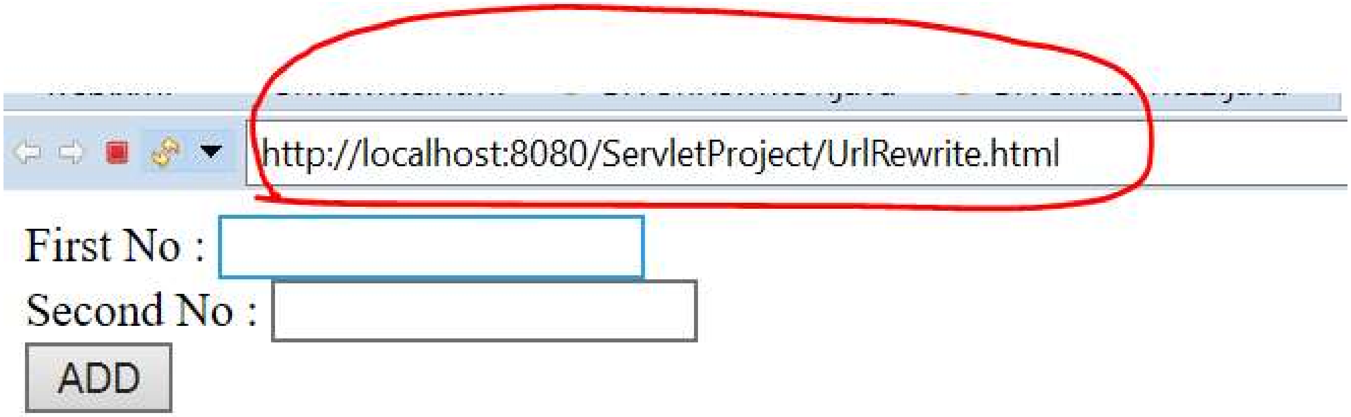
```
response.setContentType("text/html");
PrintWriter out=response.getWriter();
int n1 = Integer.parseInt(request.getParameter("txt_n1"));
int n2 = Integer.parseInt(request.getParameter("txt_n2"));
int n3 = n1+n2;
RequestDispatcher requestDispatcher =
        request.getRequestDispatcher("urlRewrite2?n1="+n1+"&n2="+n2+"&n3="+n3);
requestDispatcher.forward(request, response);
```

Display Result



```
response.setContentType("text/html");
PrintWriter out=response.getWriter();
String txt_n1 = request.getParameter("n1");
String txt_n2 = request.getParameter("n2");
String txt_n3 = request.getParameter("n3");

out.println("Number 1 : "+txt_n1);
out.println("Number 2 : "+txt_n2);
out.println("Sum  : "+txt_n3);
out.close();
```

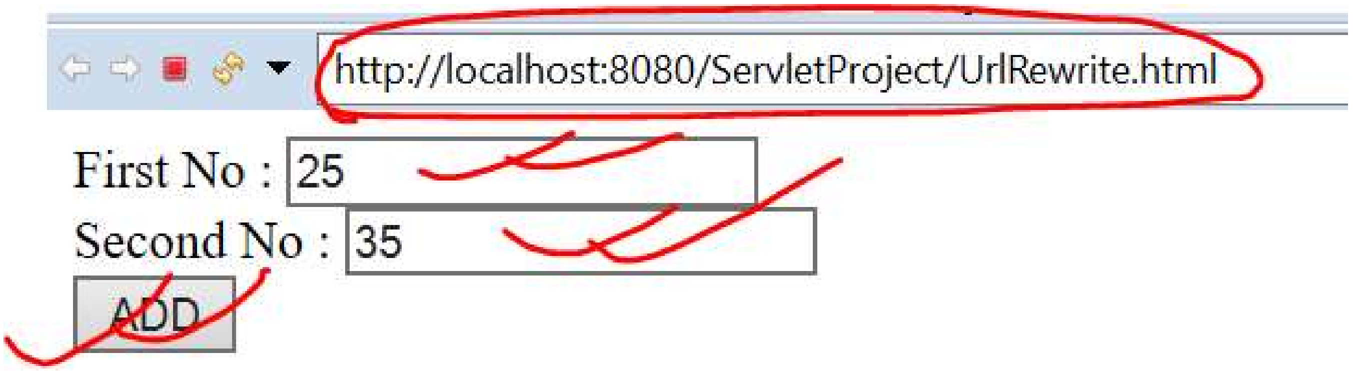
Output:



http://localhost:8080/ServletProject/UrlRewrite.html

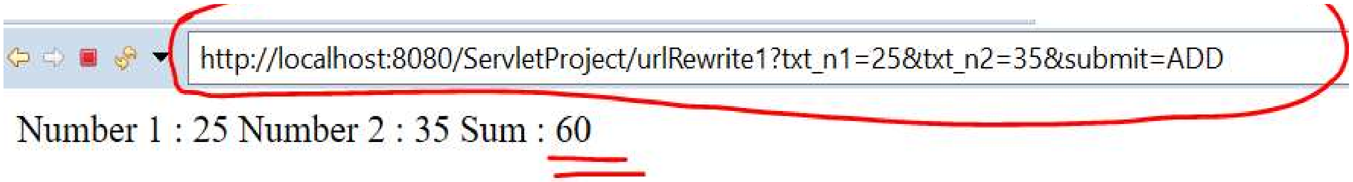
First No :

Second No :


http://localhost:8080/ServletProject/UrlRewrite.html

First No : 25

Second No : 35


http://localhost:8080/ServletProject/urlRewrite1?txt_n1=25&txt_n2=35&submit=ADD

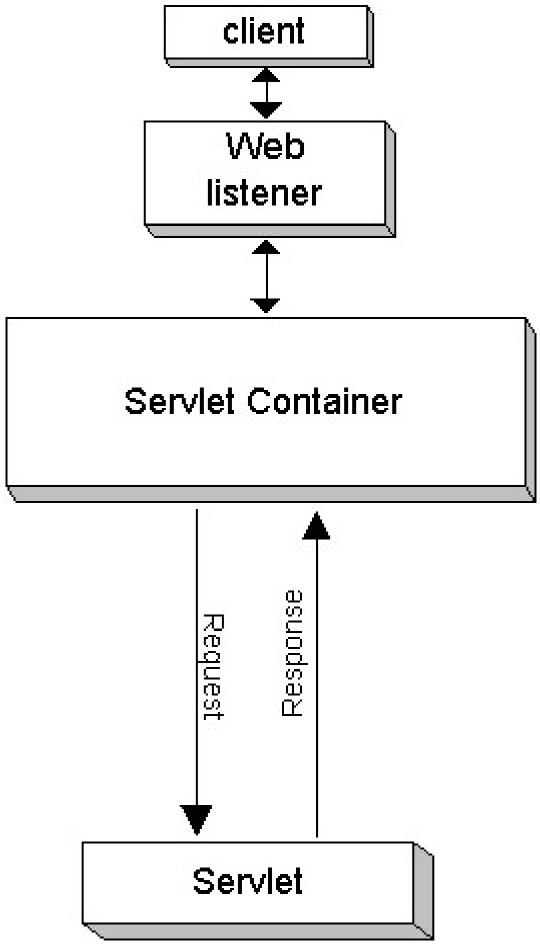
Number 1 : 25 Number 2 : 35 Sum : 60

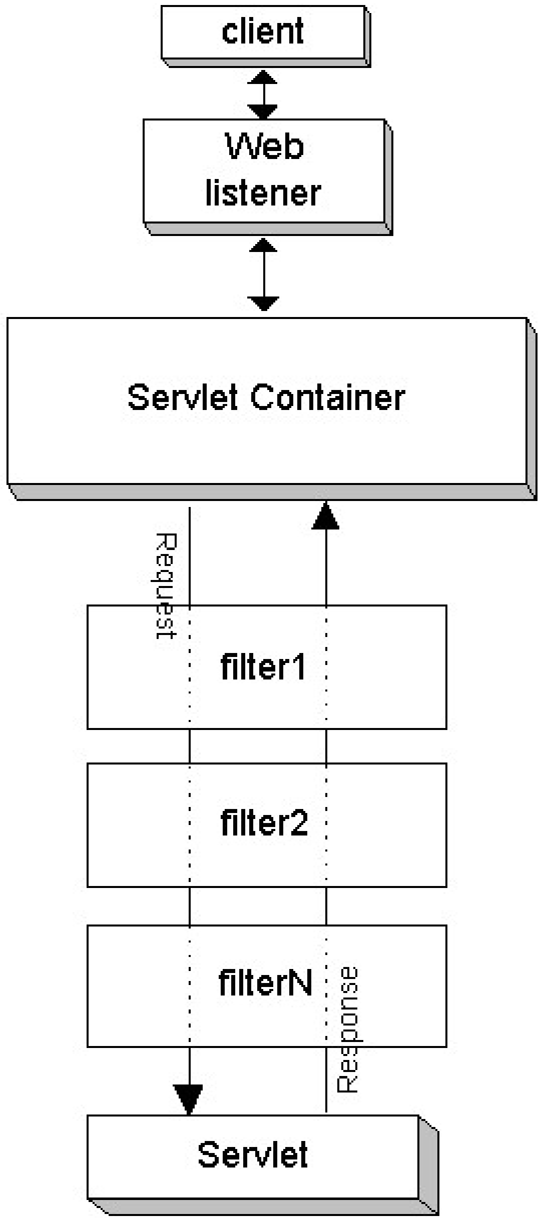
Servlet Filter

A Servlet Filter is an object that is invoked at the **pre-processing** and **post-processing** of a request. In other words, it is typically used to perform a particular piece of functionality either before or after the primary functionality that a web application is performed. Servlet Filter is mainly used to perform filtering tasks such as **Conversion**, **Logging**, **Compression**, **Request Encryption** and **Decryption**, **Input Validation** etc.

Common Tasks of Filter

- Logging request parameters to log files.
- Authentication and autherization of request for resources.
- Formatting of request body or header before sending it to servlet.
- Compressing the response data sent to the client.
- Alter response by adding some cookies, header information etc.





Exploring Filter Interface

- **javax.servlet.Filter**

- **public interface Filter{ }**

Methods

- **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 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.

FilterChain Interface
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. Filters use the FilterChain to invoke the next filter in the chain, or if the calling filter is the last filter in the chain, to invoke the resource at the end of the chain.

Exploring FilterChain

- javax.servlet.FilterChain

- public interface FilterChain

Methods

- **void doFilter(ServletRequest request, ServletResponse response)**
  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.

FilterConfig Interface
A filter configuration object used by a servlet container to pass information to a filter during initialization.

Exploring FilterConfig

- **javax.servlet.FilterConfig**

- **public interface FilterConfig { }**

Methods

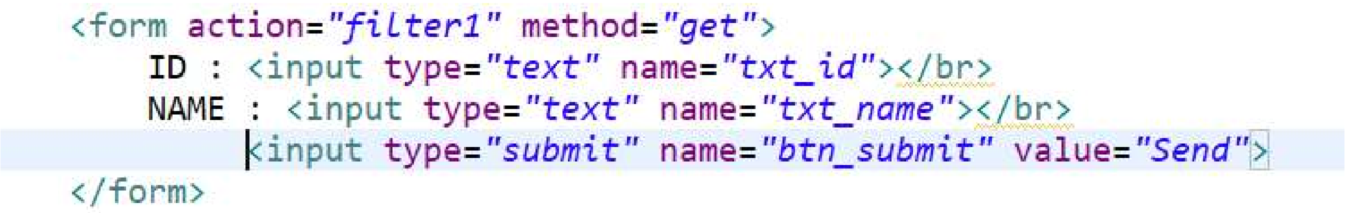
- **String getFilterName()**
  Returns the filter-name of this filter as defined in the deployment descriptor.

- **String getInitParameter(java.lang.String name)**
- Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.

- **Enumeration getInitParameterNames()**
  Returns the names of the filter's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the filter has no initialization parameters.

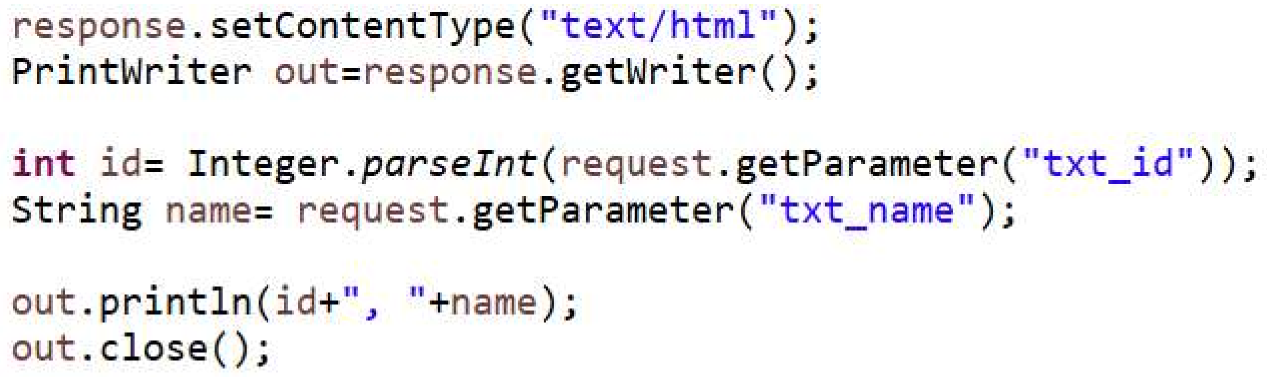
- **ServletContext getServletContext()**
  Returns a reference to the ServletContext in which the caller is executing.

Creating Filter
WebForm



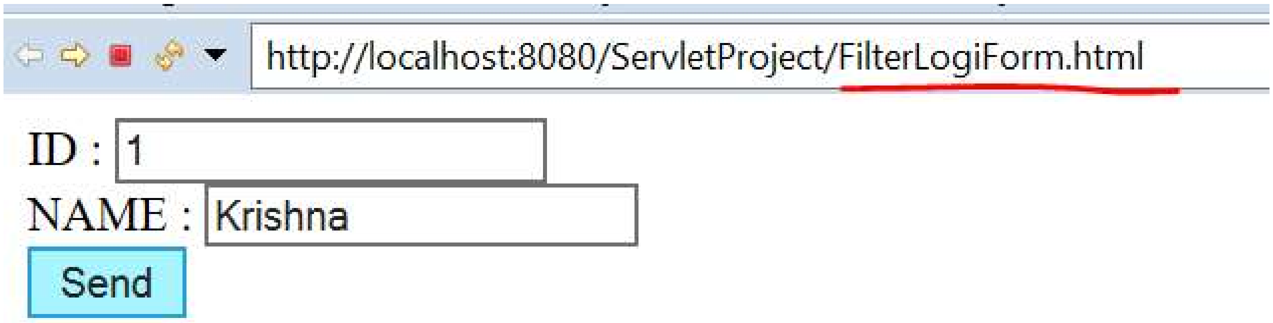
```
<form action="filter1" method="get">
    ID : <input type="text" name="txt_id"></br>
    NAME : <input type="text" name="txt_name"></br>
         <input type="submit" name="btn_submit" value="Send">
</form>
```

Servlet



```
response.setContentType("text/html");
PrintWriter out=response.getWriter();

int id= Integer.parseInt(request.getParameter("txt_id"));
String name= request.getParameter("txt_name");

out.println(id+", "+name);
out.close();
```

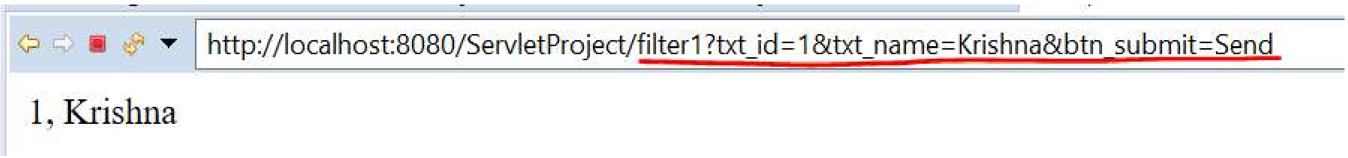
Task-1



http://localhost:8080/ServletProject/FilterLogiForm.html

ID : 1

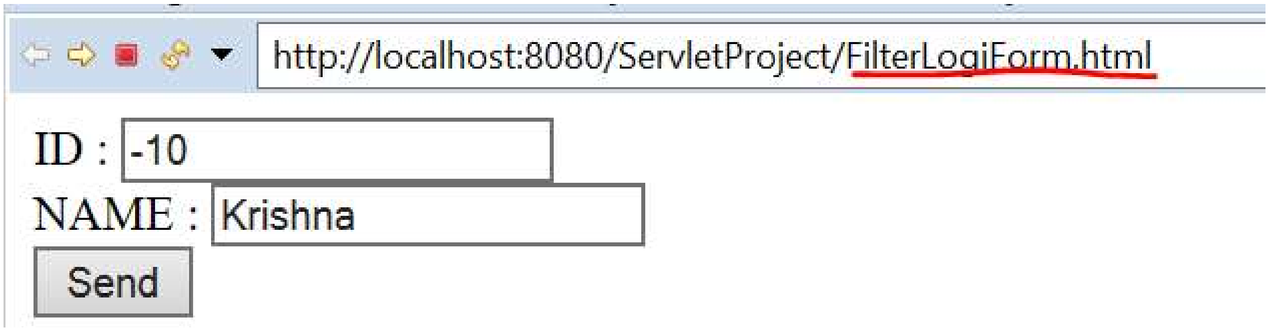
NAME : Krishna



http://localhost:8080/ServletProject/filter1?txt_id=1&txt_name=Krishna&btn_submit=Send

1, Krishna

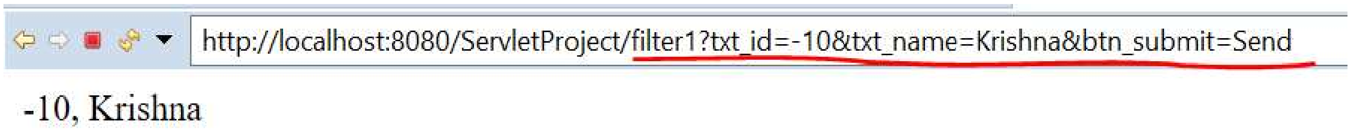
Task-2



http://localhost:8080/ServletProject/FilterLogiForm.html

ID : -10

NAME : Krishna

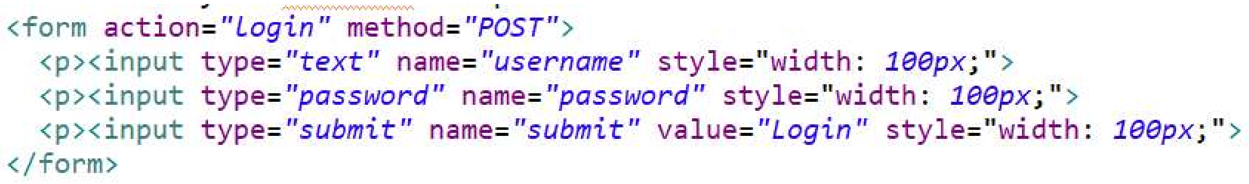


http://localhost:8080/ServletProject/filter1?txt_id=-10&txt_name=Krishna&btn_submit=Send

-10, Krishna

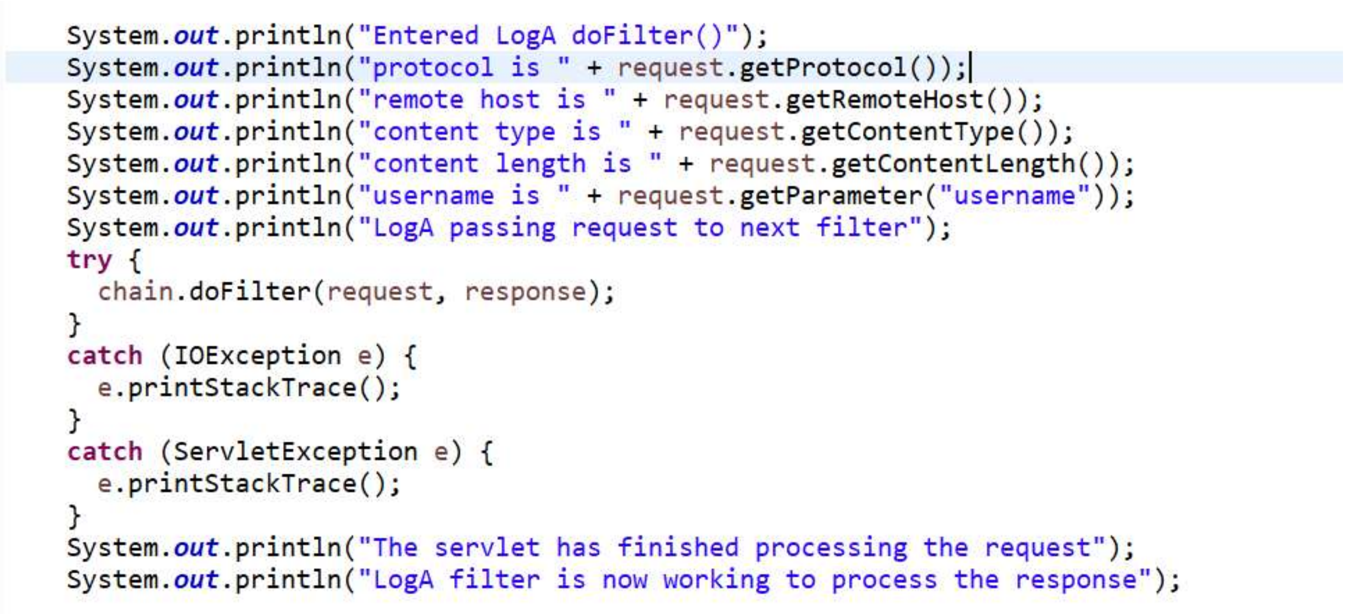
Creating Filter Chain

Login Form



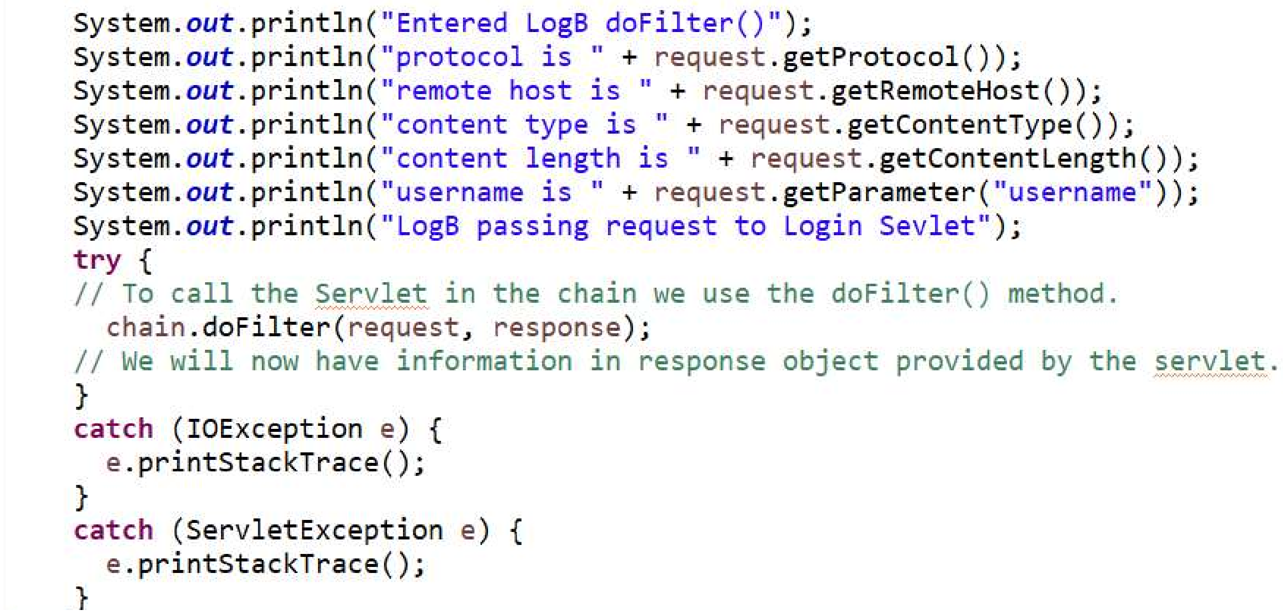
```
<form action="login" method="POST">
  <p><input type="text" name="username" style="width: 100px;">
  <p><input type="password" name="password" style="width: 100px;">
  <p><input type="submit" name="submit" value="Login" style="width: 100px;">
</form>
```

LogA [First Filter]



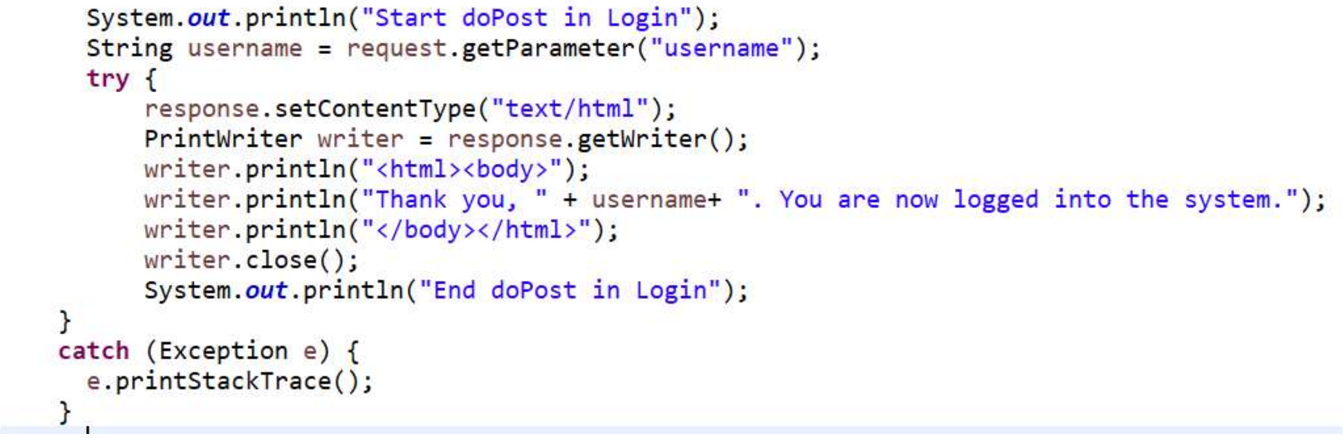
```
System.out.println("Entered LogA doFilter()");
System.out.println("protocol is " + request.getProtocol());
System.out.println("remote host is " + request.getRemoteHost());
System.out.println("content type is " + request.getContentType());
System.out.println("content length is " + request.getContentLength());
System.out.println("username is " + request.getParameter("username"));
System.out.println("LogA passing request to next filter");
try {
  chain.doFilter(request, response);
}
catch (IOException e) {
  e.printStackTrace();
}
catch (ServletException e) {
  e.printStackTrace();
}
System.out.println("The servlet has finished processing the request");
System.out.println("LogA filter is now working to process the response");
```

LogB [Second Filter]



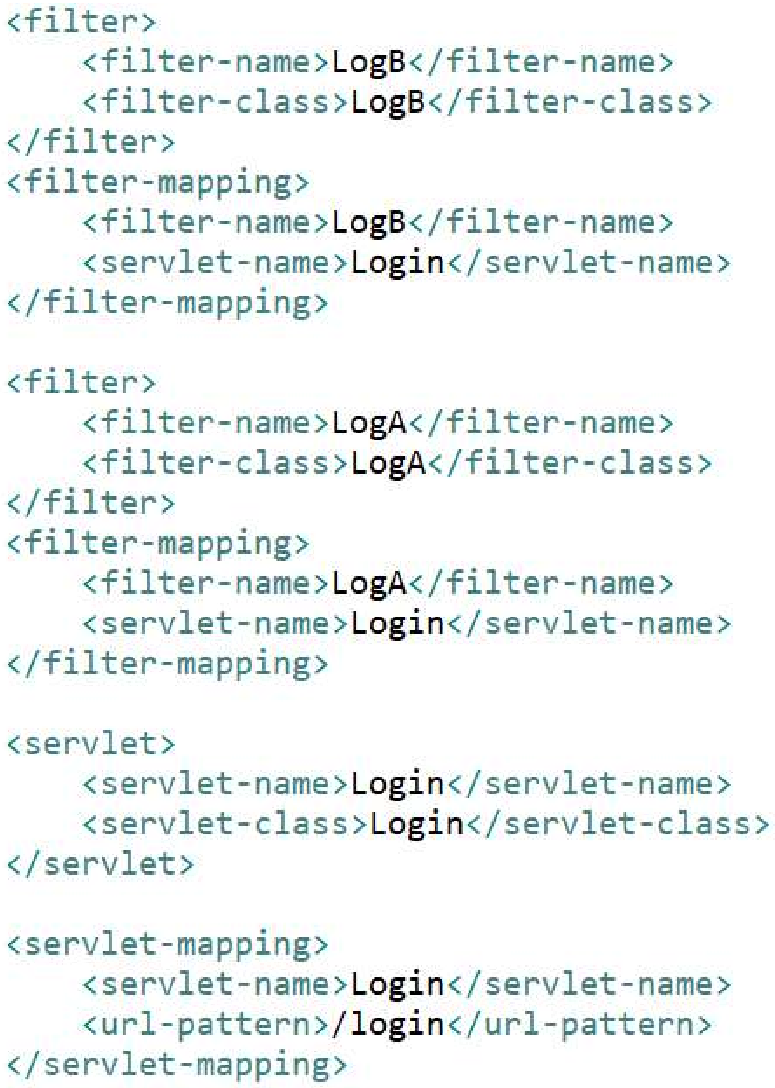
```
System.out.println("Entered LogB doFilter()");
System.out.println("protocol is " + request.getProtocol());
System.out.println("remote host is " + request.getRemoteHost());
System.out.println("content type is " + request.getContentType());
System.out.println("content length is " + request.getContentLength());
System.out.println("username is " + request.getParameter("username"));
System.out.println("LogB passing request to Login Sevlet");
try {
// To call the Servlet in the chain we use the doFilter() method.
  chain.doFilter(request, response);
// We will now have information in response object provided by the servlet.
}
catch (IOException e) {
  e.printStackTrace();
}
catch (ServletException e) {
  e.printStackTrace();
}
```

Login [Servlet]



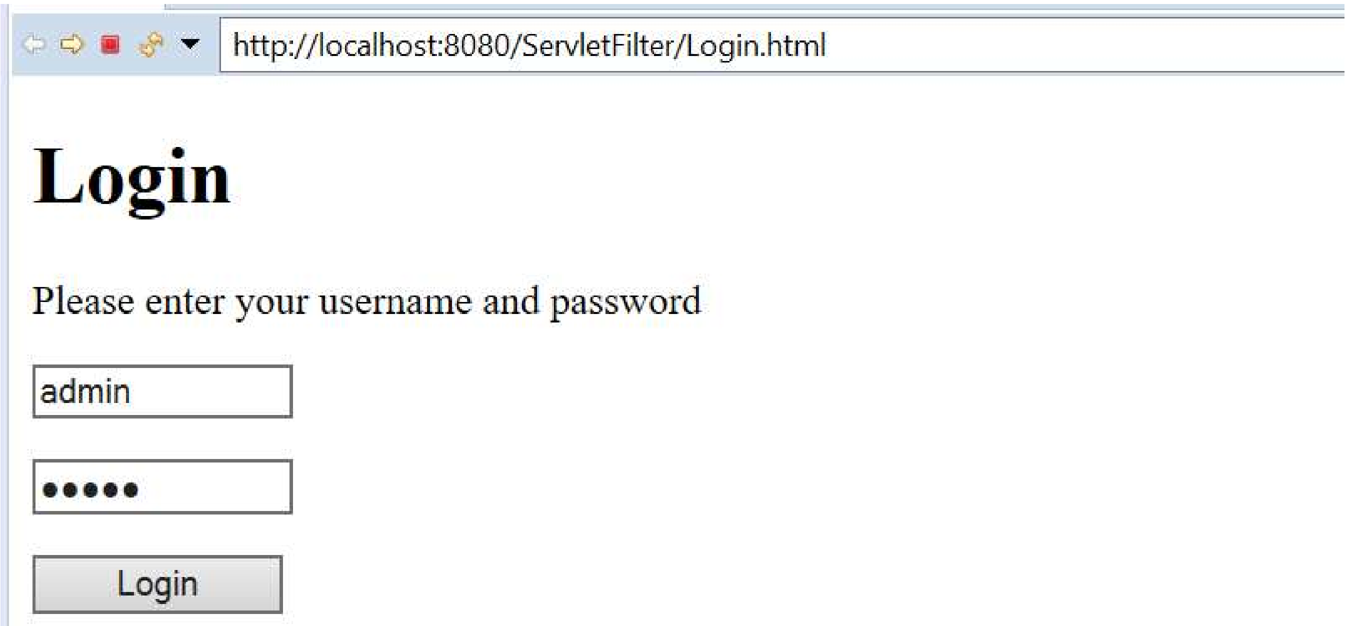
```
  System.out.println("Start doPost in Login");
  String username = request.getParameter("username");
  try {
      response.setContentType("text/html");
      PrintWriter writer = response.getWriter();
      writer.println("<html><body>");
      writer.println("Thank you, " + username+ ". You are now logged into the system.");
      writer.println("</body></html>");
      writer.close();
      System.out.println("End doPost in Login");
}
catch (Exception e) {
  e.printStackTrace();
}
```

web.xml



```
<filter>
    <filter-name>LogB</filter-name>
    <filter-class>LogB</filter-class>
</filter>
<filter-mapping>
    <filter-name>LogB</filter-name>
    <servlet-name>Login</servlet-name>
</filter-mapping>

<filter>
    <filter-name>LogA</filter-name>
    <filter-class>LogA</filter-class>
</filter>
<filter-mapping>
    <filter-name>LogA</filter-name>
    <servlet-name>Login</servlet-name>
</filter-mapping>

<servlet>
    <servlet-name>Login</servlet-name>
    <servlet-class>Login</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>Login</servlet-name>
    <url-pattern>/login</url-pattern>
</servlet-mapping>
```

Output:



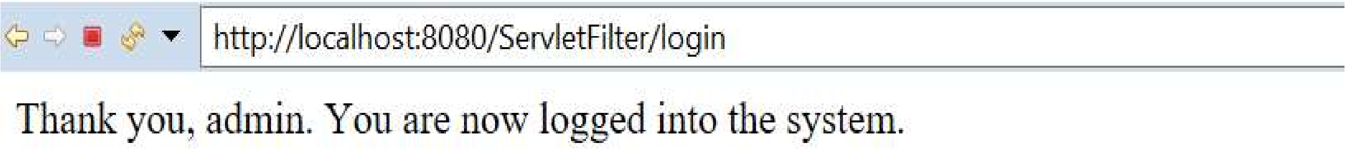
http://localhost:8080/ServletFilter/Login.html

Login

Please enter your username and password

admin

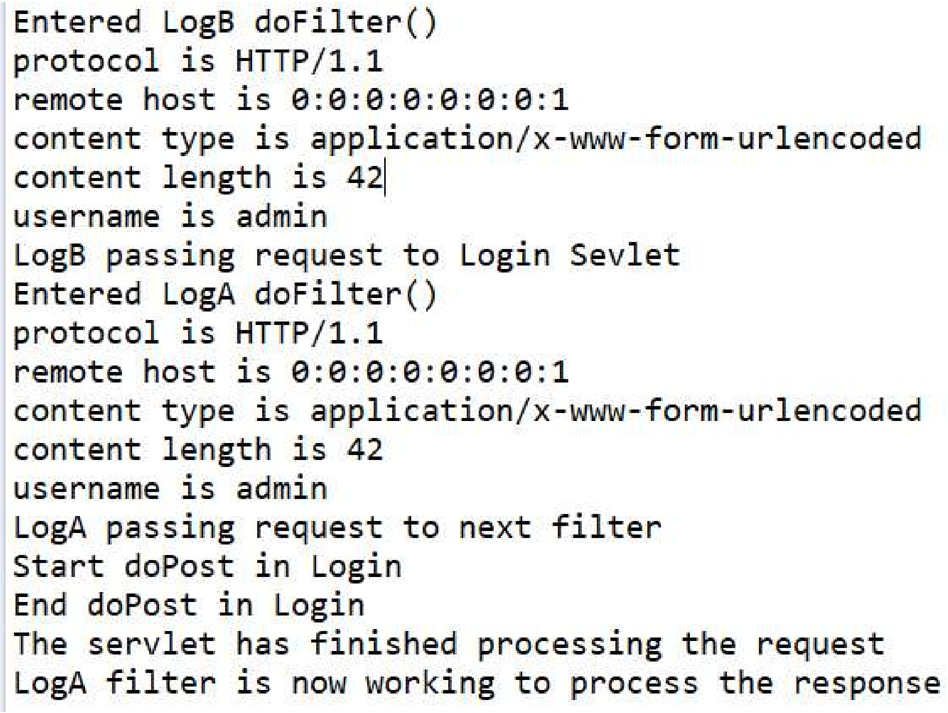
●●●●●



http://localhost:8080/ServletFilter/login

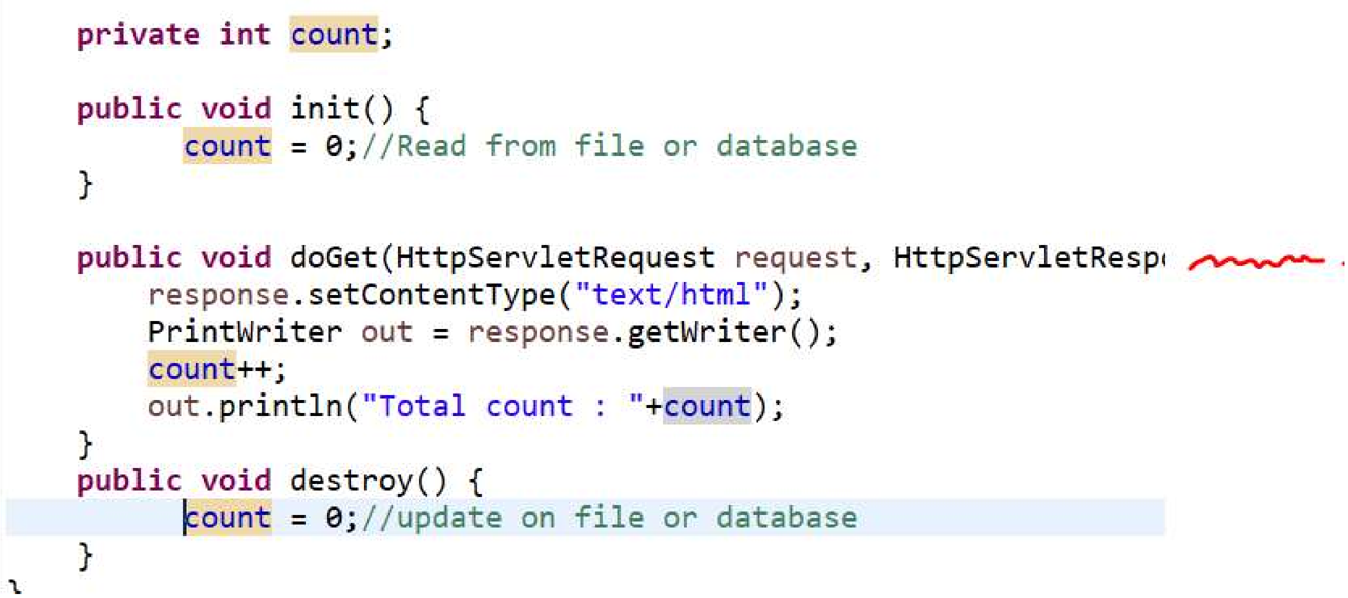
Thank you, admin. You are now logged into the system.

Log Details



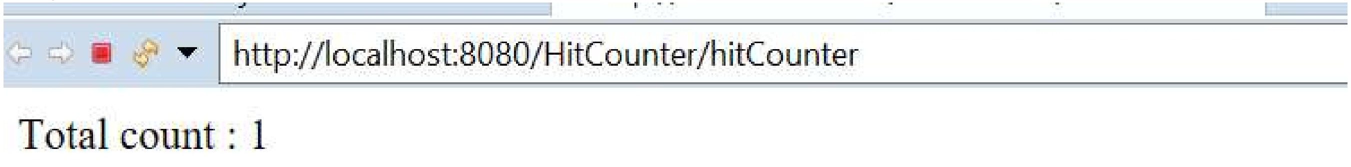
```
Entered LogB doFilter()
protocol is HTTP/1.1
remote host is 0:0:0:0:0:0:0:1
content type is application/x-www-form-urlencoded
content length is 42
username is admin
LogB passing request to Login Sevlet
Entered LogA doFilter()
protocol is HTTP/1.1
remote host is 0:0:0:0:0:0:0:1
content type is application/x-www-form-urlencoded
content length is 42
username is admin
LogA passing request to next filter
Start doPost in Login
End doPost in Login
The servlet has finished processing the request
LogA filter is now working to process the response
```

Creating Hit Counter



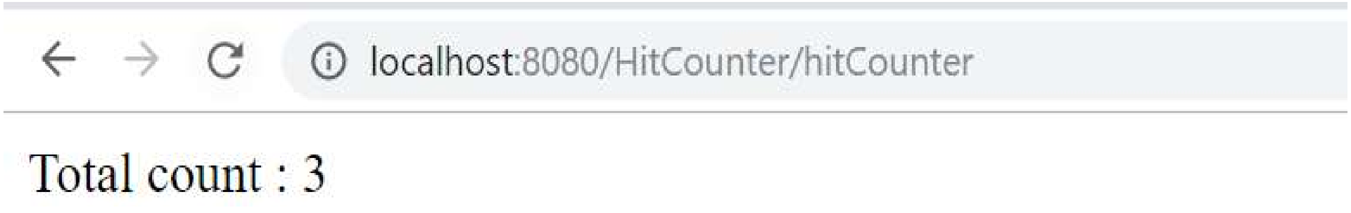
```
    private int count;

    public void init() {
          count = 0;//Read from file or database
    }

    public void doGet(HttpServletRequest request, HttpServletResp
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        count++;
        out.println("Total count : "+count);
    }
    public void destroy() {
          count = 0;//update on file or database
    }
}
```

Output:



http://localhost:8080/HitCounter/hitCounter

```
Total count : 1
```



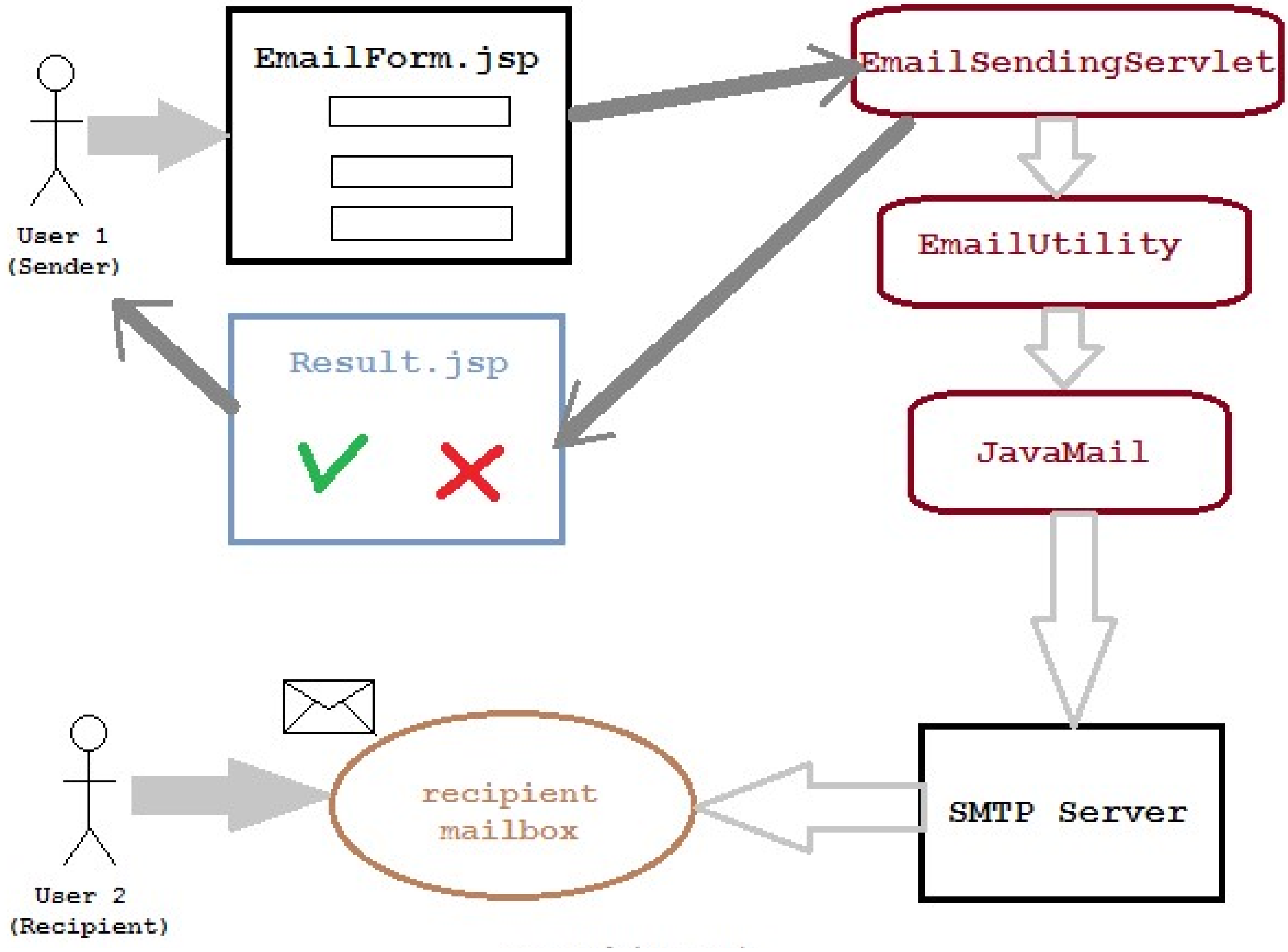
localhost:8080/HitCounter/hitCounter

```
Total count : 3
```

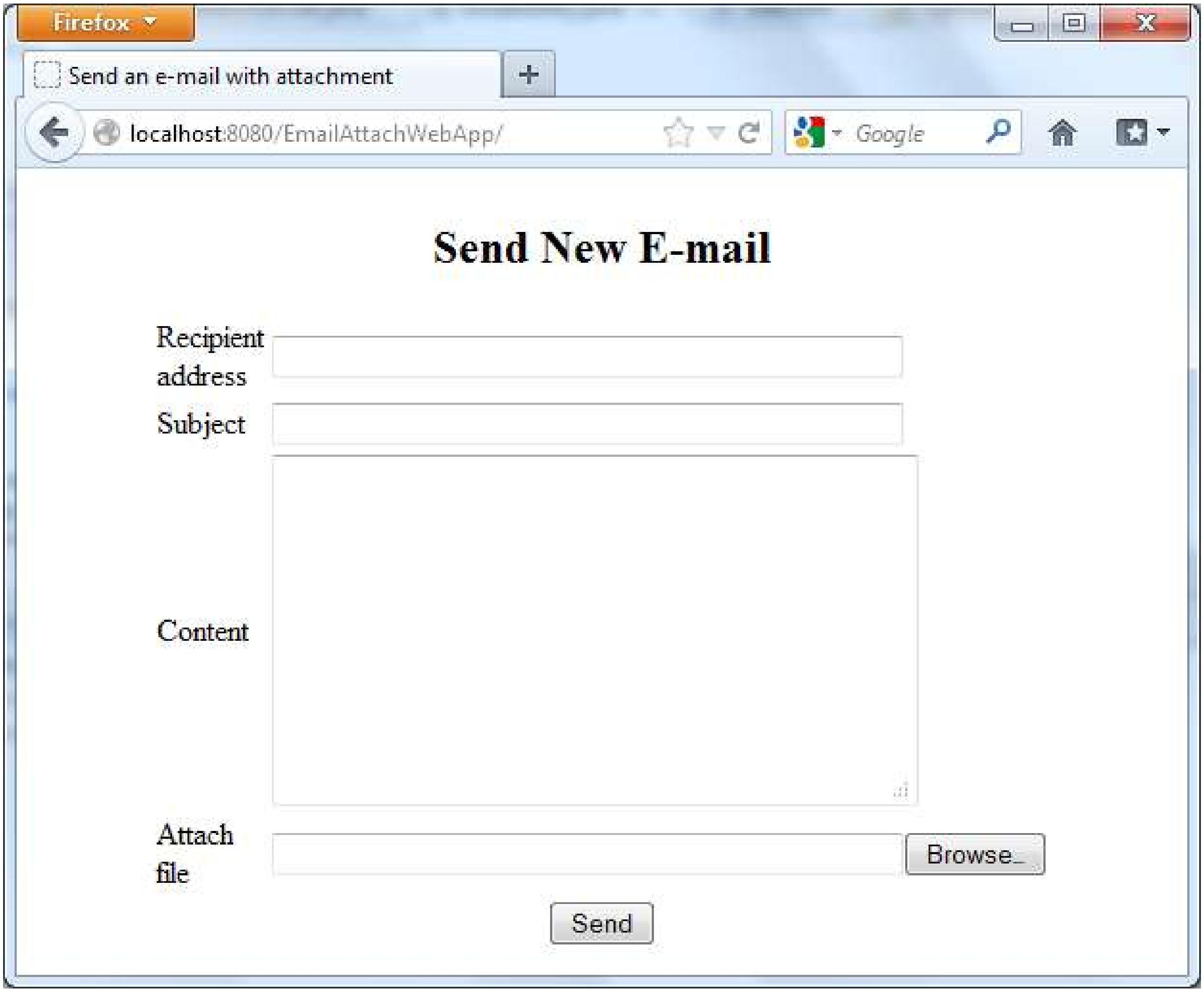
Send Email

- **EmailSend**
- **EmailAttachmentSend**

Sending Email



Sending Email with Attachment



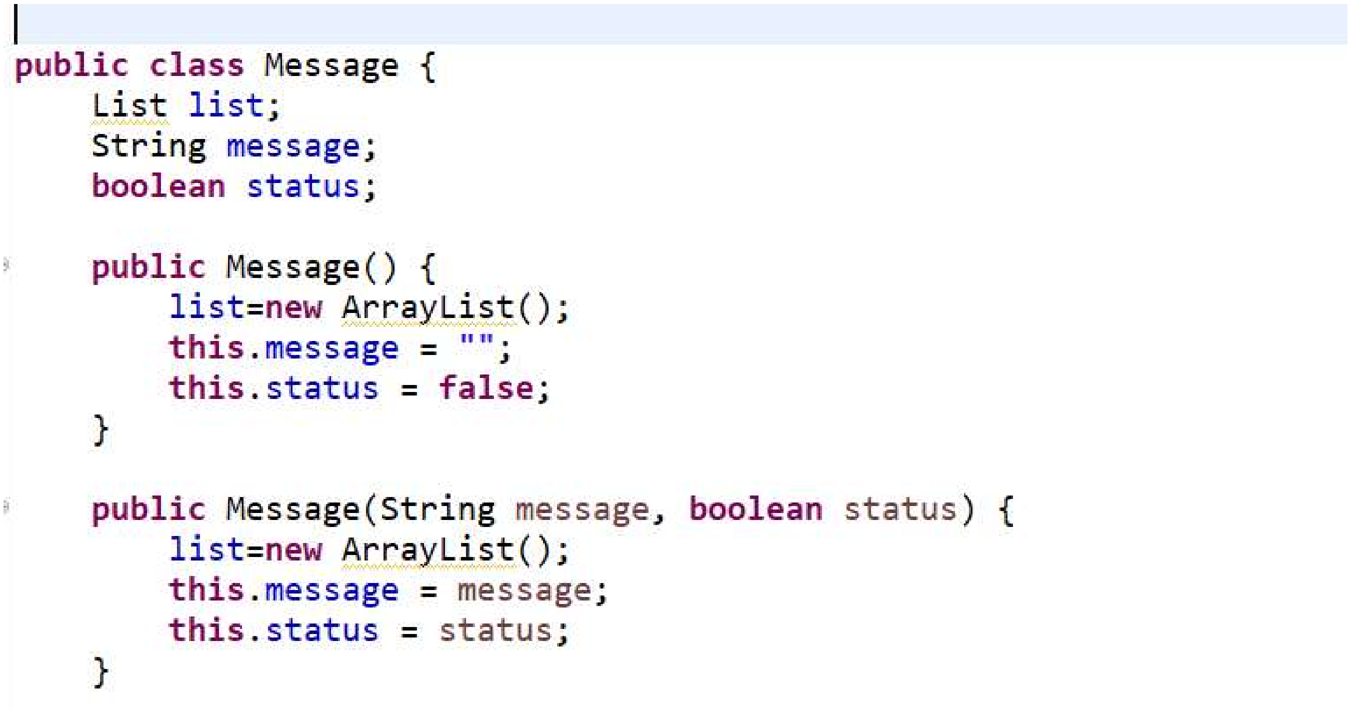
Upload/Download File

- **Upload File [FileUpload.zip]**
- **Upload File [FileUpload2.zip]**
- **Download File [DownloadFile.zip]**

Database Connectivity

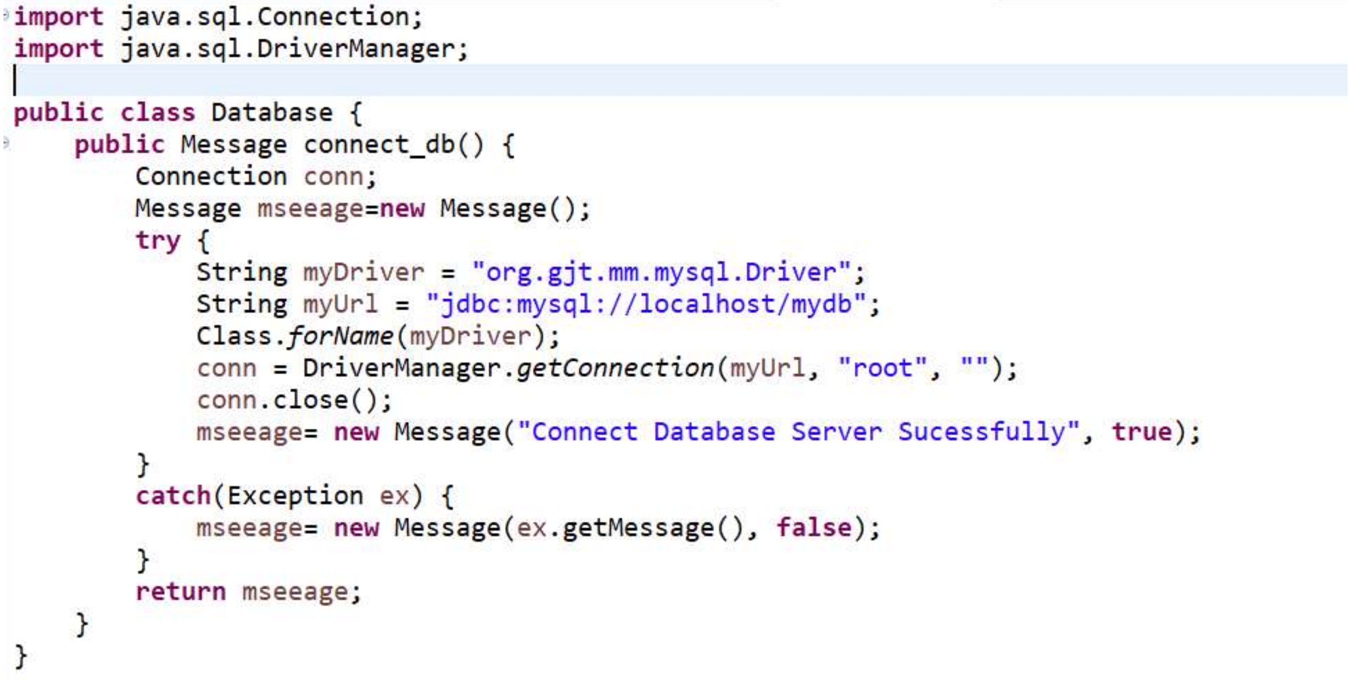
- **JDBC [JDBC-1.zip]**
- **JDBC [JDBC-1.zip]**

JDBC (Java Database Connectivity)-1
Message Class



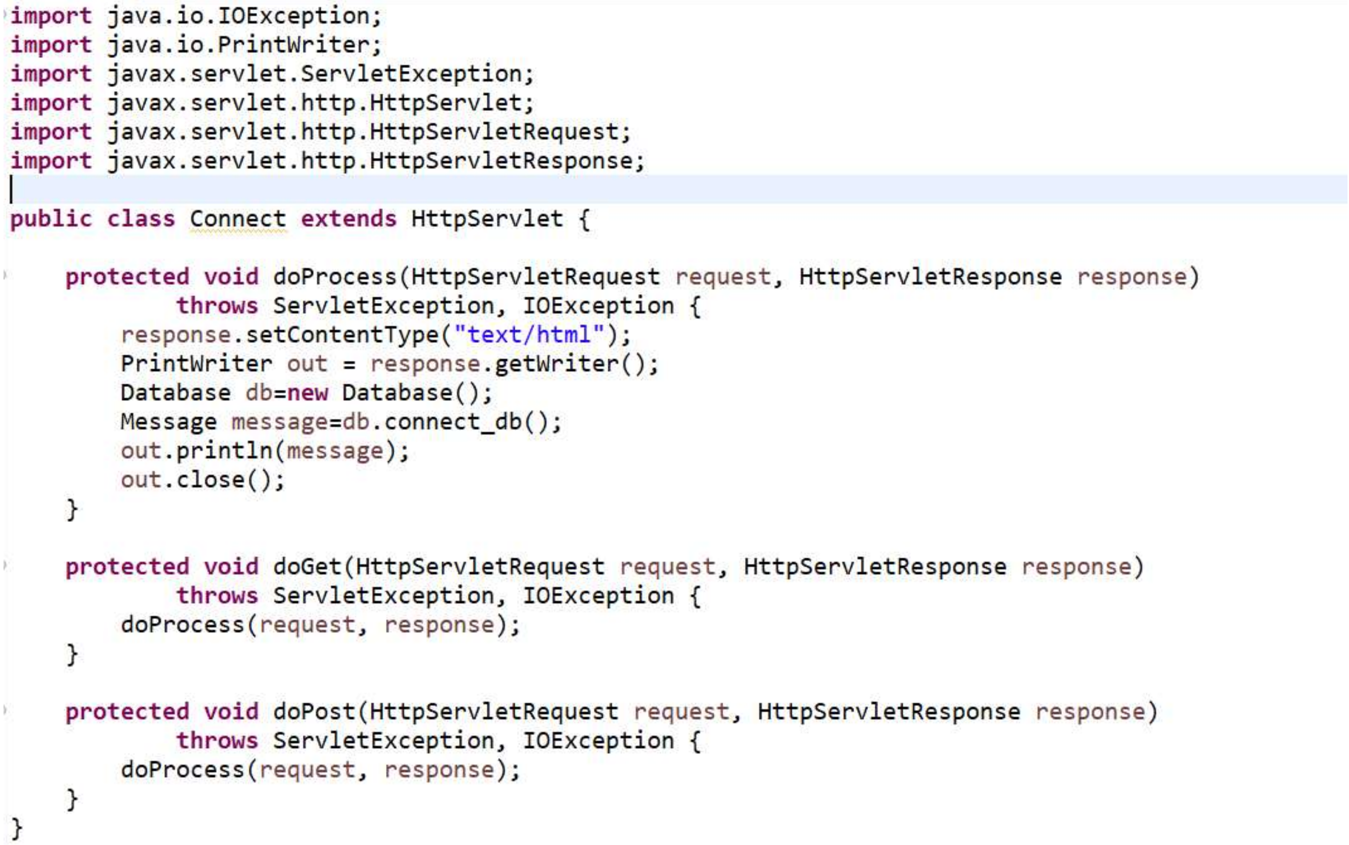
```java
public class Message {
    List list;
    String message;
    boolean status;

    public Message() {
        list=new ArrayList();
        this.message = "";
        this.status = false;
    }

    public Message(String message, boolean status) {
        list=new ArrayList();
        this.message = message;
        this.status = status;
    }
```

Database Class



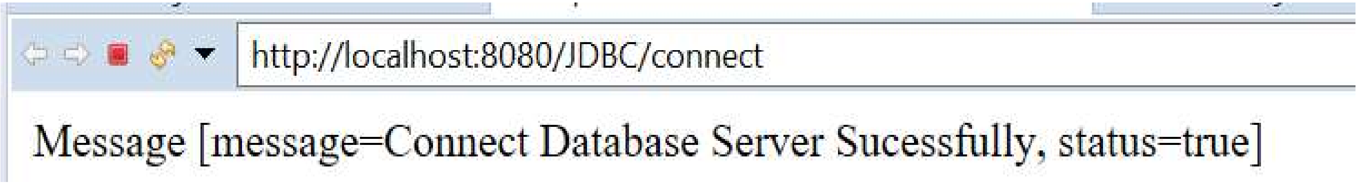
```java
import java.sql.Connection;
import java.sql.DriverManager;

public class Database {
    public Message connect_db() {
        Connection conn;
        Message mseeage=new Message();
        try {
            String myDriver = "org.gjt.mm.mysql.Driver";
            String myUrl = "jdbc:mysql://localhost/mydb";
            Class.forName(myDriver);
            conn = DriverManager.getConnection(myUrl, "root", "");
            conn.close();
            mseeage= new Message("Connect Database Server Sucessfully", true);
        }
        catch(Exception ex) {
            mseeage= new Message(ex.getMessage(), false);
        }
        return mseeage;
    }
}
```

Connect Servlet



```java
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 Connect extends HttpServlet {

    protected void doProcess(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        Database db=new Database();
        Message message=db.connect_db();
        out.println(message);
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doProcess(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        doProcess(request, response);
    }
}
```

Output



http://localhost:8080/JDBC/connect

```
Message [message=Connect Database Server Sucessfully, status=true]
```

Servlet Annotations

Servlet API 3.0 has introduced a new package called **javax.servlet.annotation**. It provides annotation types which can be used for annotating a servlet class. If you use annotation, then the deployment descriptor (web.xml) is not required. But you should use tomcat7 or any later version of tomcat.

SN	ANNOTATION	DESCRIPTION
1	**@WebServlet**	To declare a servlet.
2	**@WebInitParam**	To specify an initialization parameter.
3	**@WebFilter**	To declare a servlet filter.
4	**@WebListener**	To declare a WebListener
5	**@HandlesTypes**	To declare the class types that a ServletContainerInitializer can handle.
6	**@HttpConstraint**	This annotation is used within the ServletSecurity annotation to represent the security constraints to be applied to all HTTP protocol methods for which a corresponding HttpMethodConstraint element does NOT occur within the ServletSecurity annotation.
7	**@HttpMethodConstraint**	This annotation is used within the ServletSecurity annotation to represent security constraints on specific HTTP protocol messages.
8	**@MultipartConfig**	Annotation that may be specified on a Servlet class, indicating that instances of the Servlet expect requests that conform to the multipart/form-data MIME type.
9	**@ServletSecurity**	This annotation is used on a Servlet implementation class to specify security constraints to be enforced by a Servlet container on HTTP protocol messages.

@WebServlet

Examples

```
@WebServlet("/hello")
@WebServlet(name = "simpleServlet", urlPatterns = { "/hello" }, loadOnStartup = 1)
@WebServlet("/hello")
```

@WebInitParam

Example

```
@WebServlet(value = "/Simple", initParams = {
  @WebInitParam(name = "foo", value = "Hello "),
  @WebInitParam(name = "bar", value = " World!")
})
```

References

- https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html
- https://www.codejava.net/
- https://www.tutorialandexample.com/
- https://o7planning.org/en/10167/java-jdbc-tutorial
- https://www.codejava.net/coding/jsp-servlet-jdbc-mysql-create-read-update-delete-crud-example