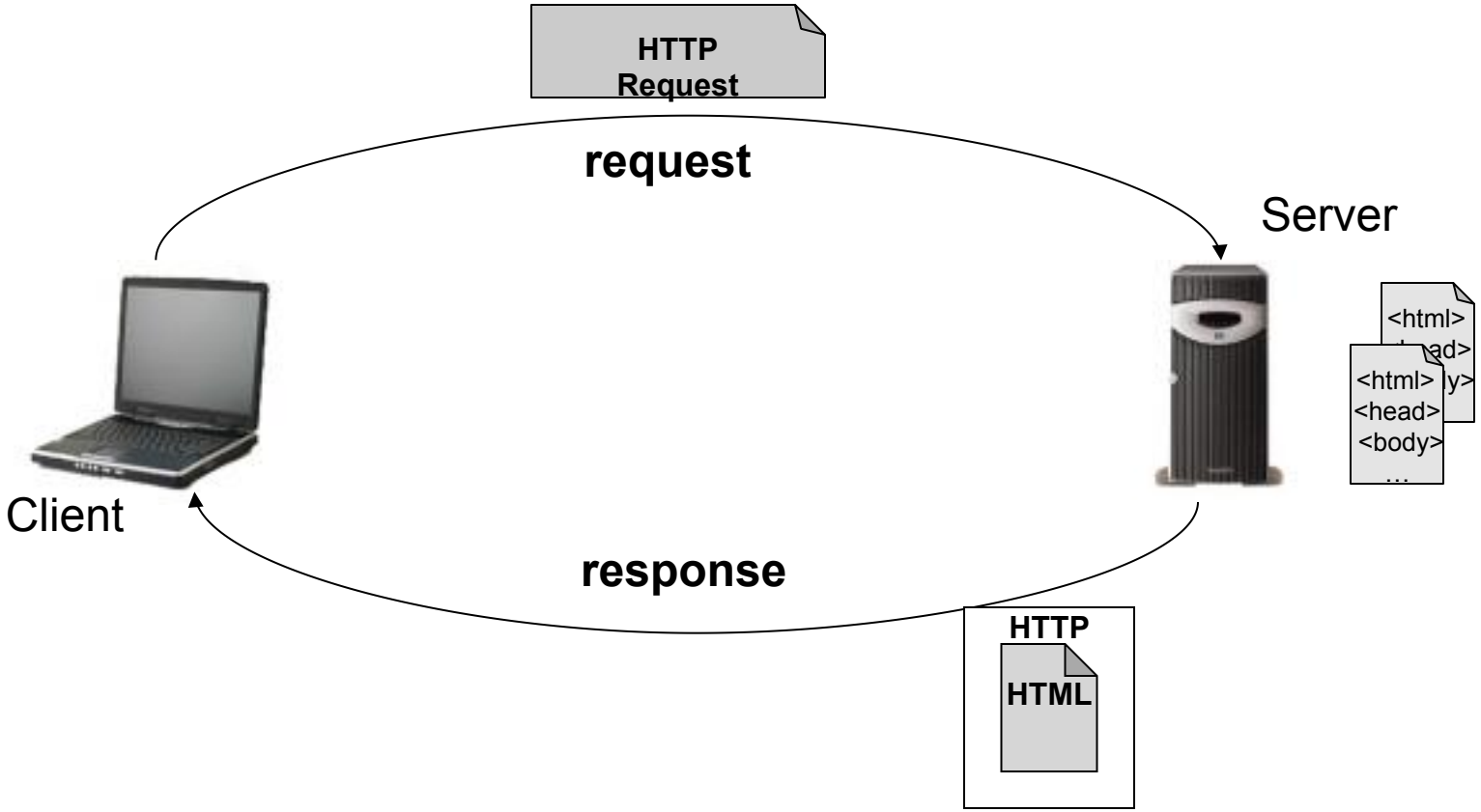
Servlets & JSPs

Agenda

- Introduction
- Servlet Architecture
- Servlet lifecycle
- Request and Response
- Being a Web Container
- Session management
- Overview of JSP
- JSP Elements
- Q & A

Introduction - request-response model

- **Request-response model.**



Introduction – what is a request and response

HTTP Request

Key elements of a "**request**" stream:

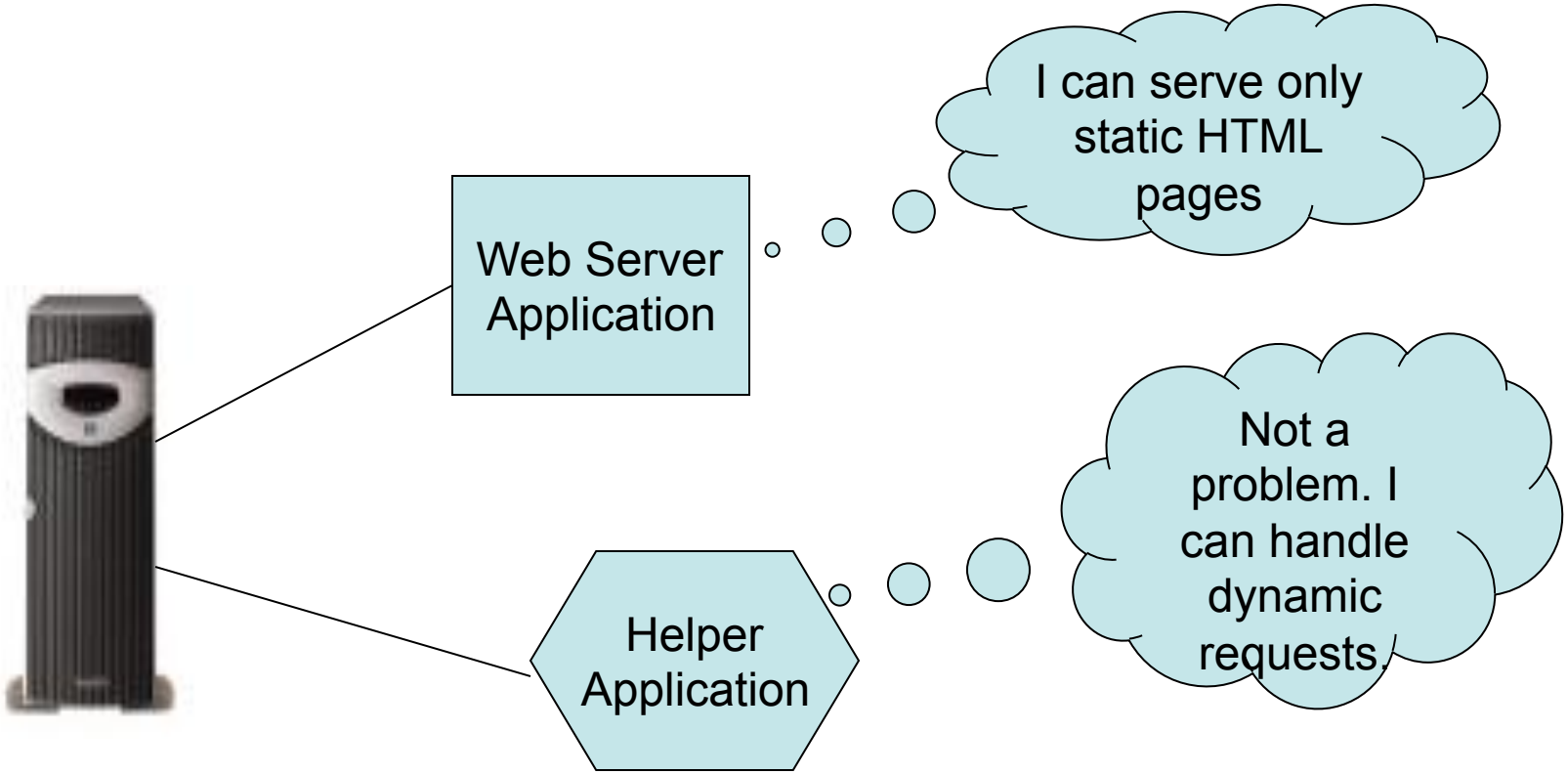
- HTTP method (action to be performed).
- The page to access (a URL).
- Form parameters.

HTTP Response

Key elements of a "**response**" stream:

- A status code (for whether the request was successful).
- Content-type (text, picture, html, etc…).
- The content (the actual content).

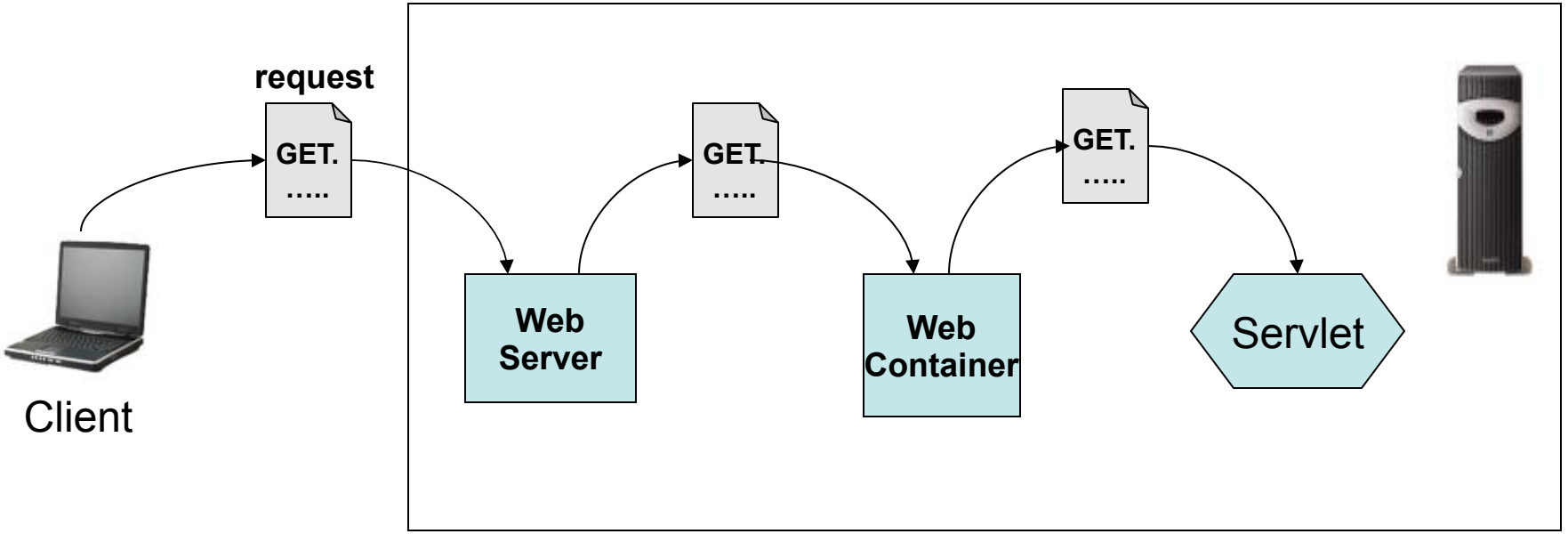
Where does Servlet come into the picture?



"The Helper Application is nothing but a **SERVLET**"

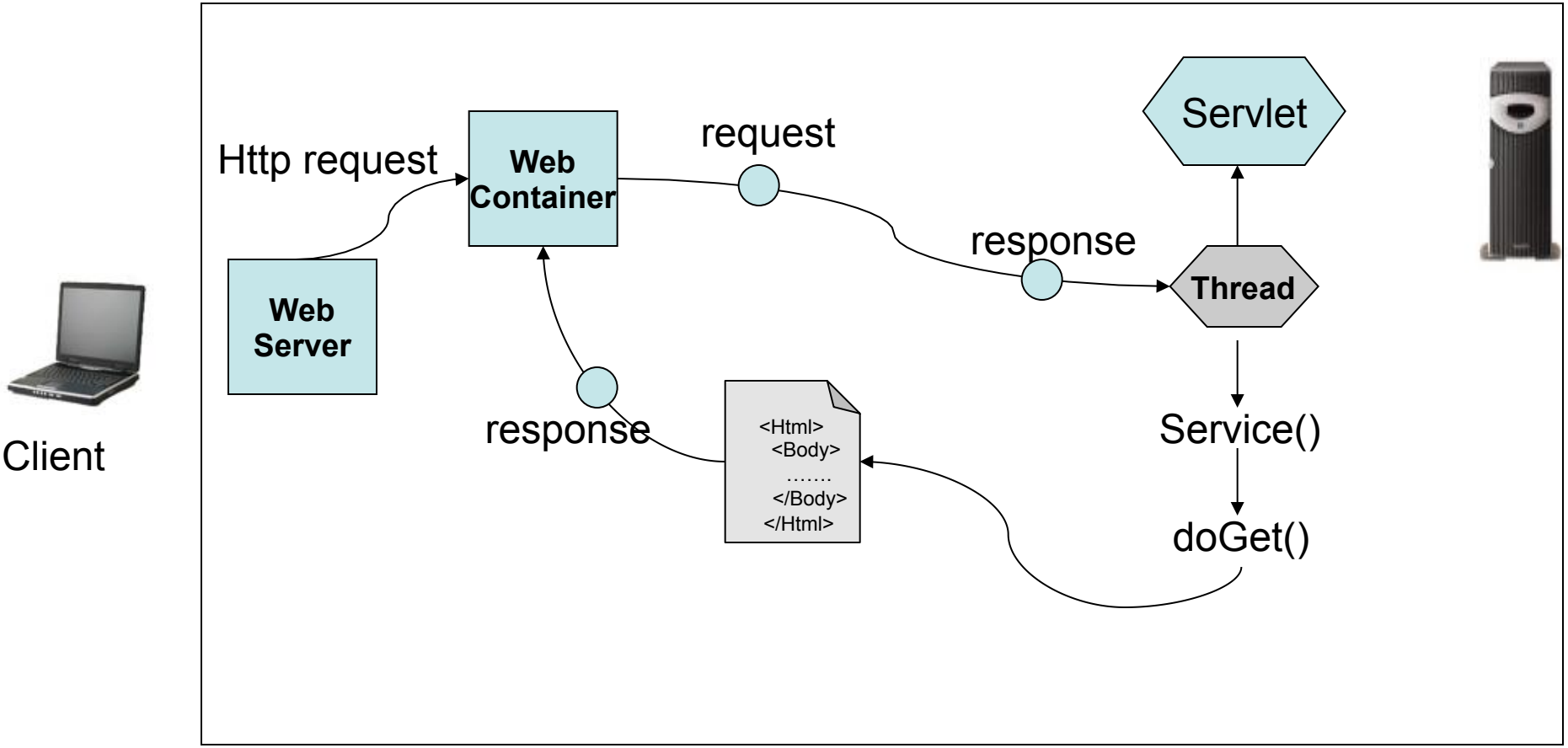
Servlet Architecture -Web Container

- **What is a Web Container?**



Servlet Architecture - Web Container

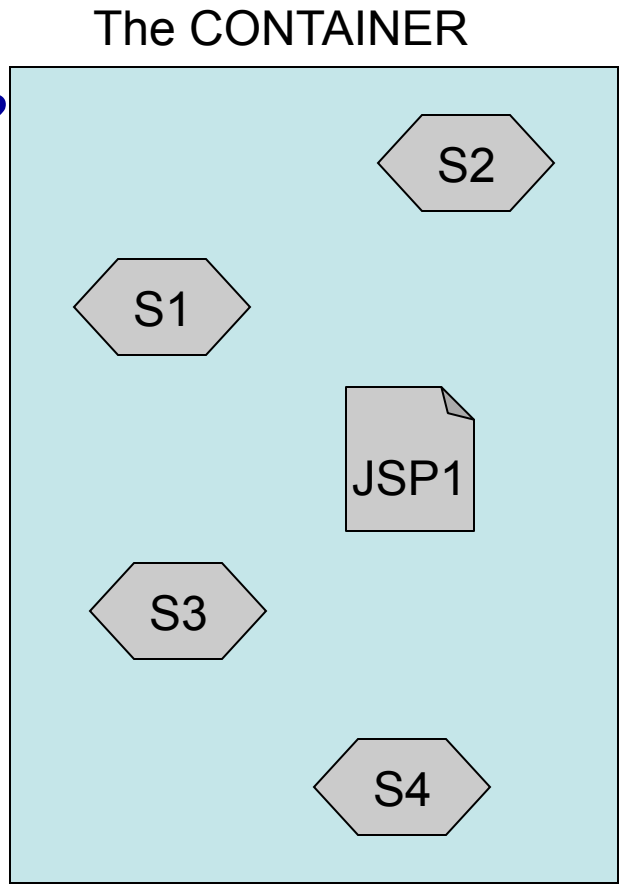
- **How does the Container handle a request?**



Servlet Architecture - Web Container

What is the role of Web Container ?

- Communication Support
- Lifecycle Management
- Multi-threading support
- Security
- JSP Support



The container can contain multiple Servlets & JSPs within it

Servlet Architecture - Deployment Descriptor

- **How does the Container know which Servlet the client has requested for?**

A Servlet can have 3 names

- Client known URL name
- Deployer known secret internal name
- Actual file name

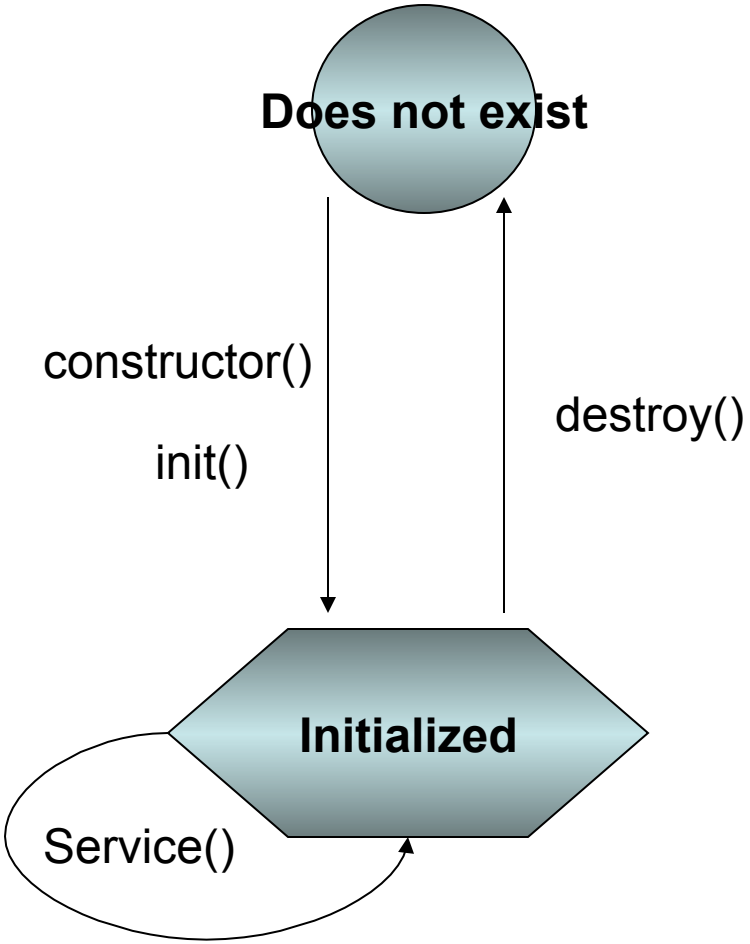
```
<web-app>
  ……
  <servlet>
    <servlet-name>LoginServ</servlet-name>
     <servlet-class>com.Login</servlet-class>
   </servlet>

  <servlet-mapping>
    <servlet-name>LoginServ</servlet-name>
    <url-pattern>/Logon</url-pattern>
  </servlet-mapping>
  ……..
  ……..
</web-app>
```

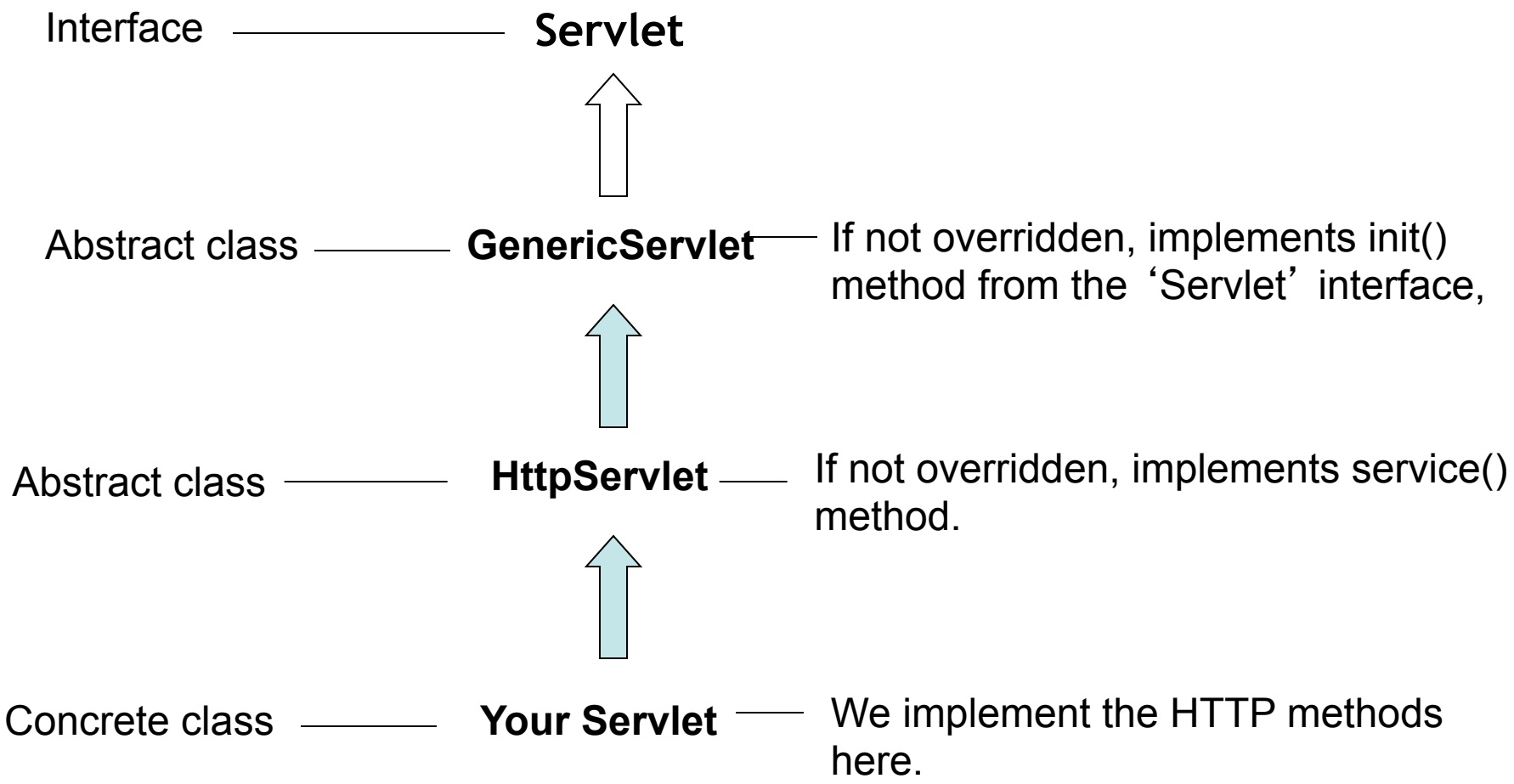
Web.xml

Servlet Lifecycle

- **The Servlet lifecycle is simple, there is only one main state - "Initialized".**



Servlet Lifecycle - Hierarchy

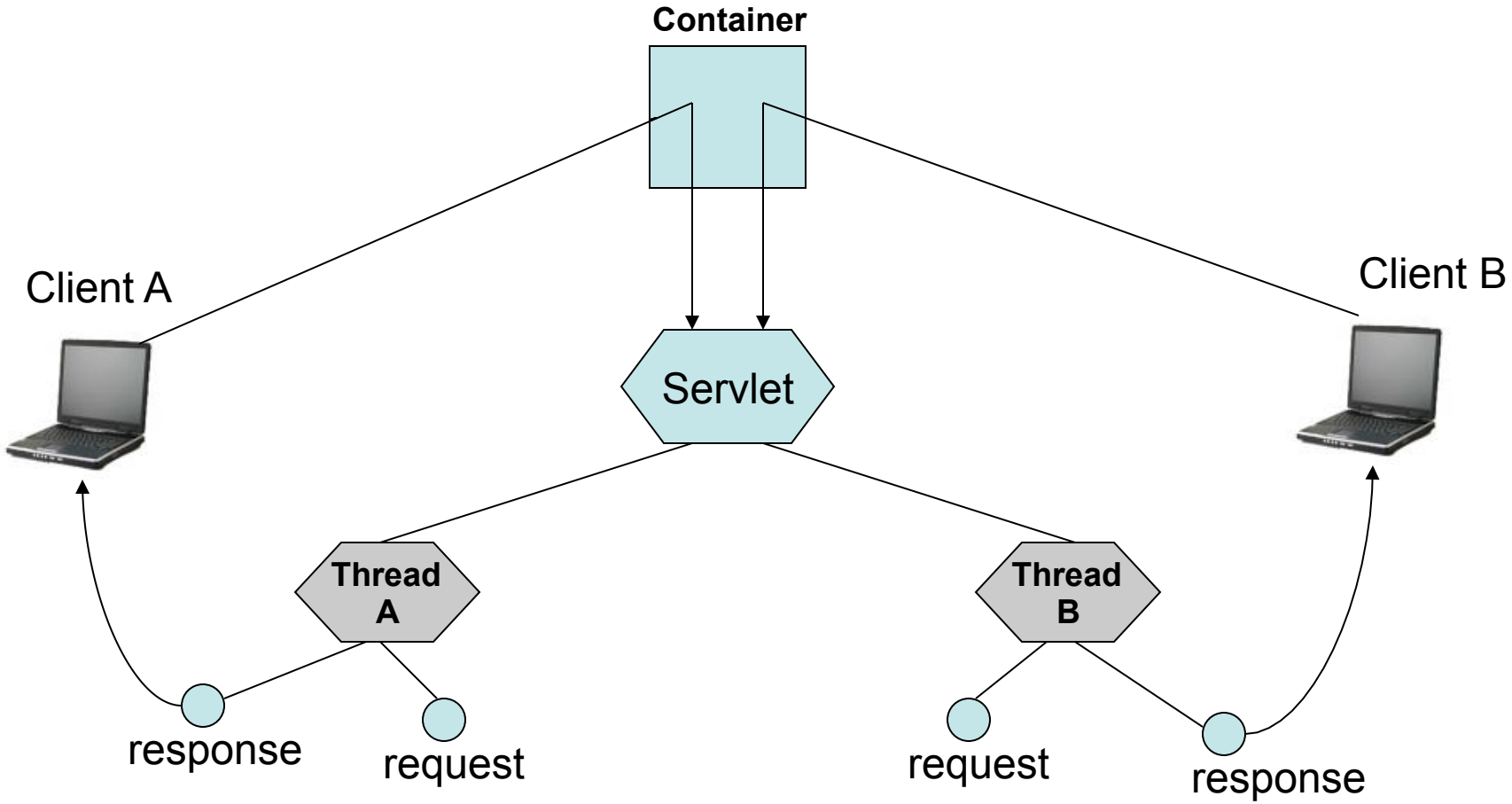


Servlet Lifecycle - 3 big moments

	When is it called	What it's for	Do you override it
init()	The container calls the init() before the servlet can service any client requests.	To initialize your servlet before handling any client requests.	Possibly
service()	When a new request for that servlet comes in.	To determine which HTTP method should be called.	No. Very unlikely
doGet() or doPost()	The service() method invokes it based on the HTTP method from the request.	To handle the business logic.	Always

Servlet Lifecycle - Thread handling

- **The Container runs multiple threads to process multiple requests to a single servlet.**

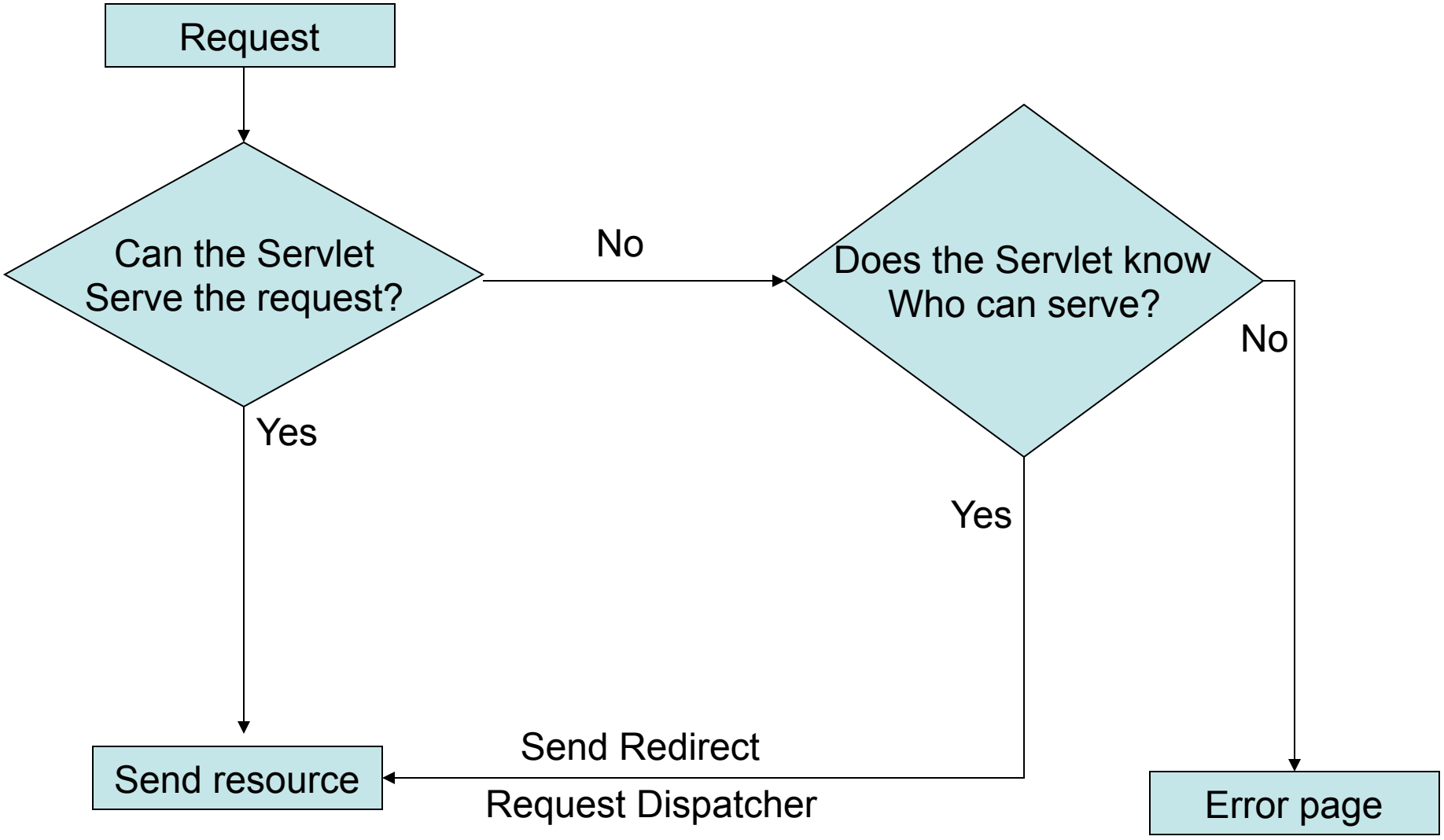


Request and Response – GET v/s POST

- **The HTTP request method determines whether doGet() or doPost() runs.**

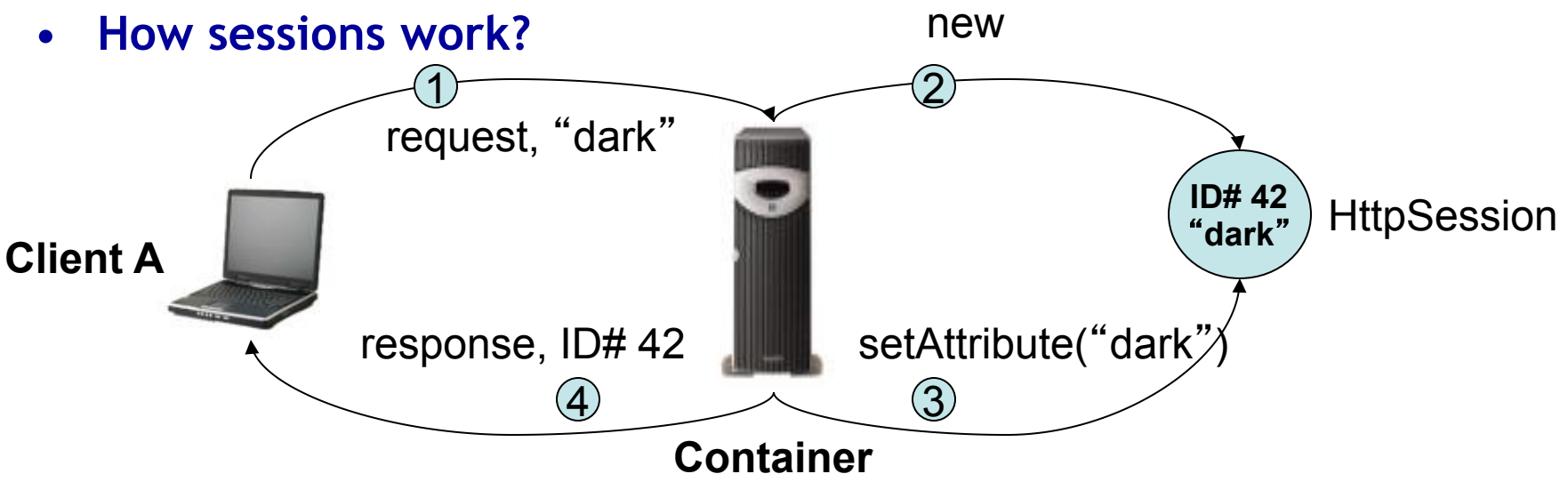
	GET (doGet())	POST (doPost())
HTTP Request	The request contains only the request line and HTTP header.	Along with request line and header it also contains HTTP body.
Parameter passing	The form elements are passed to the server by appending at the end of the URL.	The form elements are passed in the body of the HTTP request.
Size	The parameter data is limited (the limit depends on the container)	Can send huge amount of data to the server.
Idempotency	GET is Idempotent	POST is not idempotent
Usage	Generally used to fetch some information from the host.	Generally used to process the sent data.

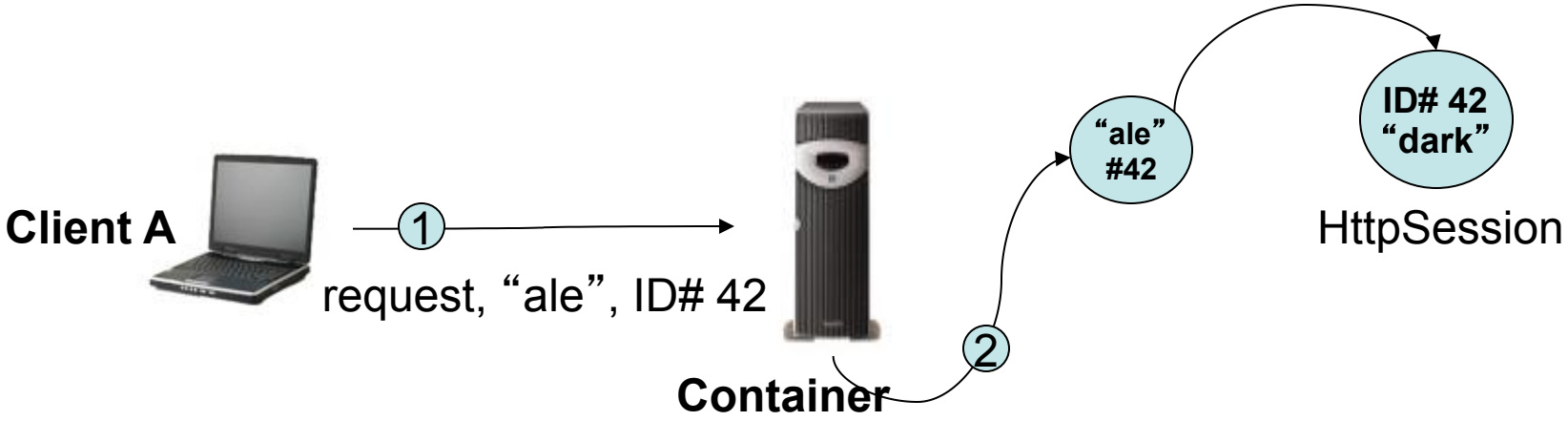
Request and Response - The response



Session Management – Session Tracking

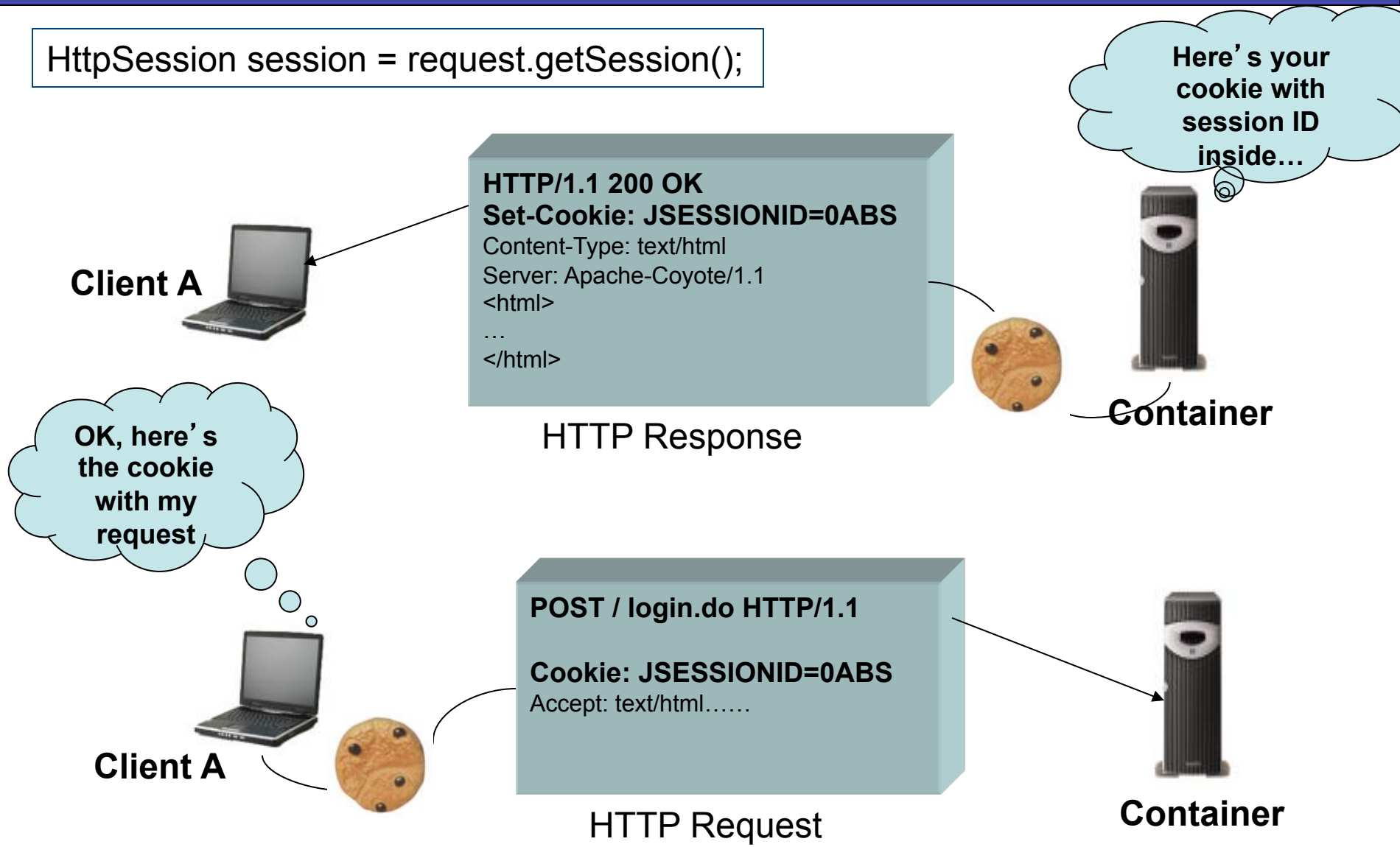
- **How sessions work?**





Session Tracking – Cookies

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



Session Tracking – URL Rewriting

URL + ;jsessionid=1234567

Client A

HTTP/1.1 200 OK
Content-Type: text/html
Server: Apache-Coyote/1.1
<html>
<body>
< a href =“ http:// www.sharmanj.com/Metavante;jsessionid=0AAB”>
**click me **
</html>

HTTP Response

Container

GET /Metavante;jsessionid=0AAB

HTTP / 1.1
Host: www.sharmanj.com
Accept: text/html

HTTP Request

Client A

Container

JSP Overview - Servlets v/s JSPs

Servlets : HTML within Java business logic

```
public void doGet(request, response)
{
PrintWriter out = response.getWriter();
String name =
      request.getParameter(name);
out.println(“<html><body>”);
out.println("Hello” + name);
out.println(“</body></html>”);
}
```

JSPs : Java within HTML Presentation logic

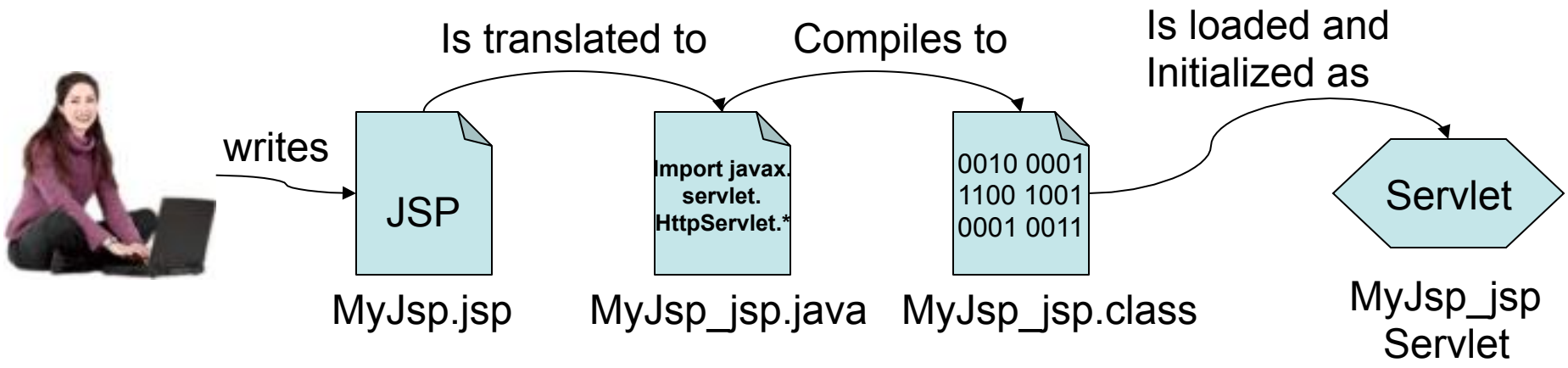
```
<html>
<body>
<% String name =
   request.getParameter(name); %>

Hello <%= name %>

</body>
</html>
```

JSP Overview - What is a JSP

- **In the end, a JSP is just a Servlet.**



JSP Elements

- **Scriptlets**
  - **<% int I = 10; %>**
  - **<% Dog d = new Dog(); %>**
- **Expressions**
  - **<%= i %>**
  - **<%= d.getName() %>**

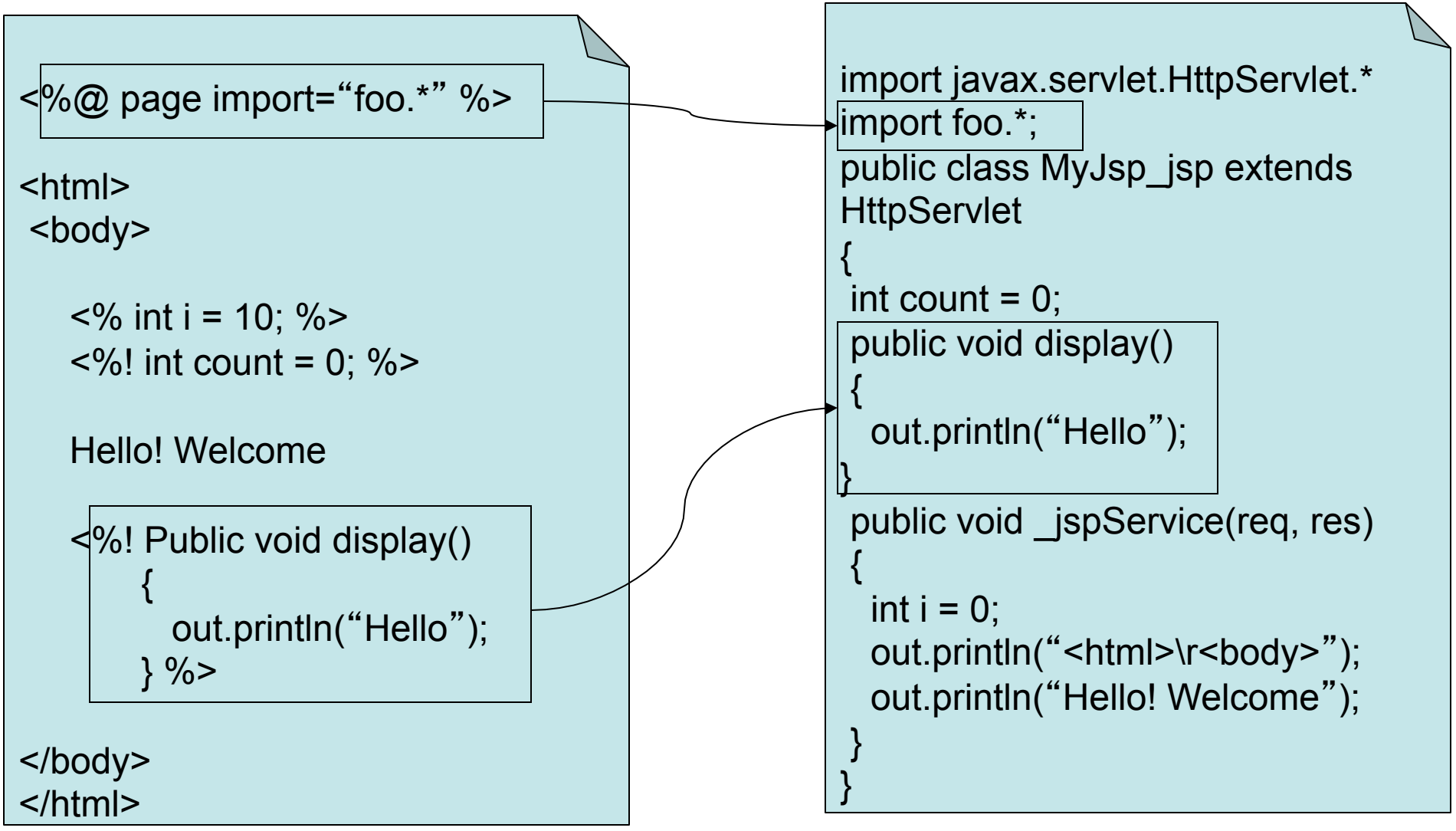
= out.println(i);

= out.println(d.getName());

- **Declarations**
  - **<%! int i=10; %>**
  - **<%! void display() { System.out.println("Hello"); } %>**
- **Directives**
  - **Pages - <%@ page import="foo.*, bar.*" %>**
  - **include - <%@ include file="/foo/myJsp.jsp" %>**
  - **taglib - <%@ taglib uri="Tags" prefix="cool" %>**

JSP Elements – JSP to Servlet

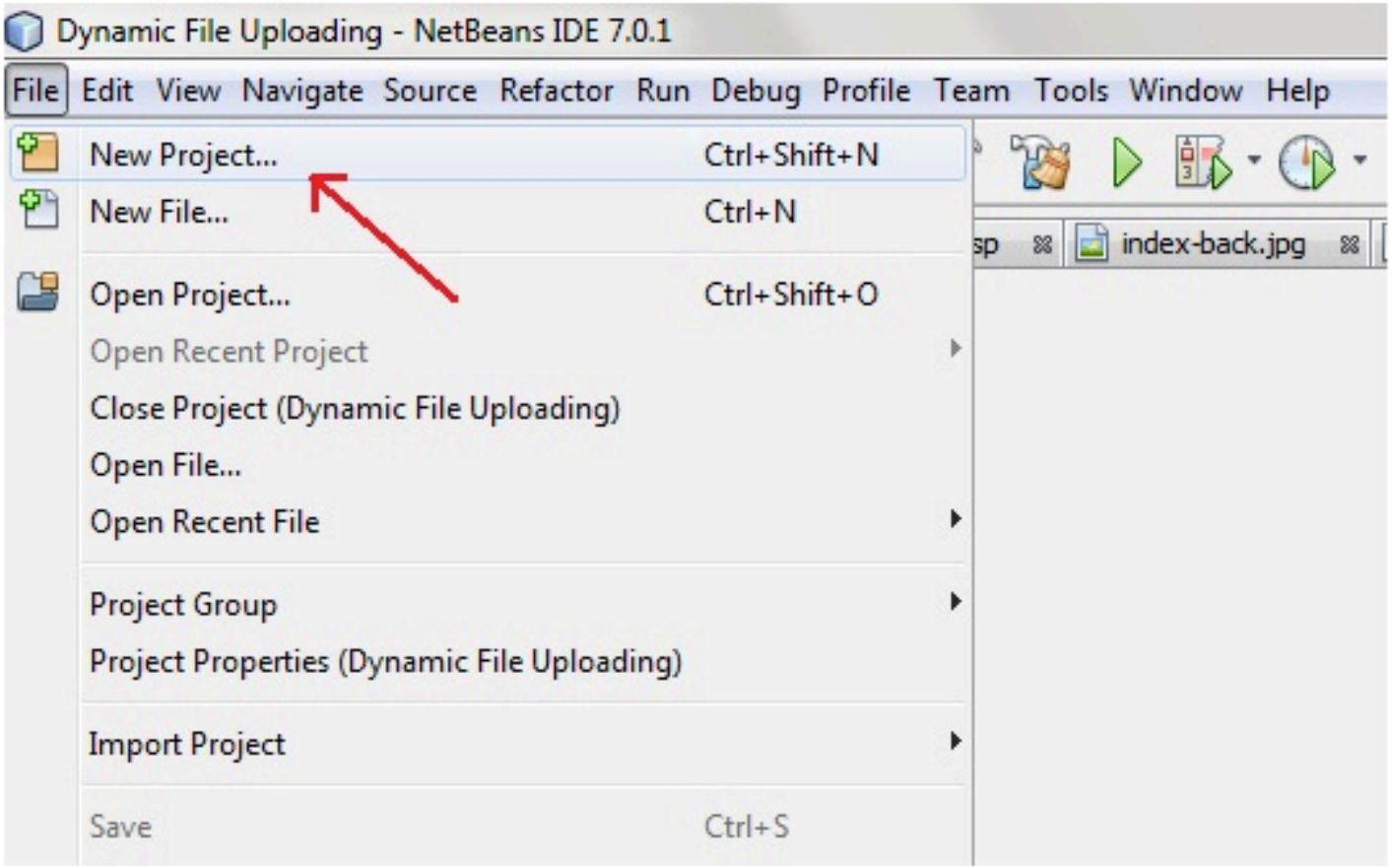
- **Where does the JSP code land in the Servlet?**



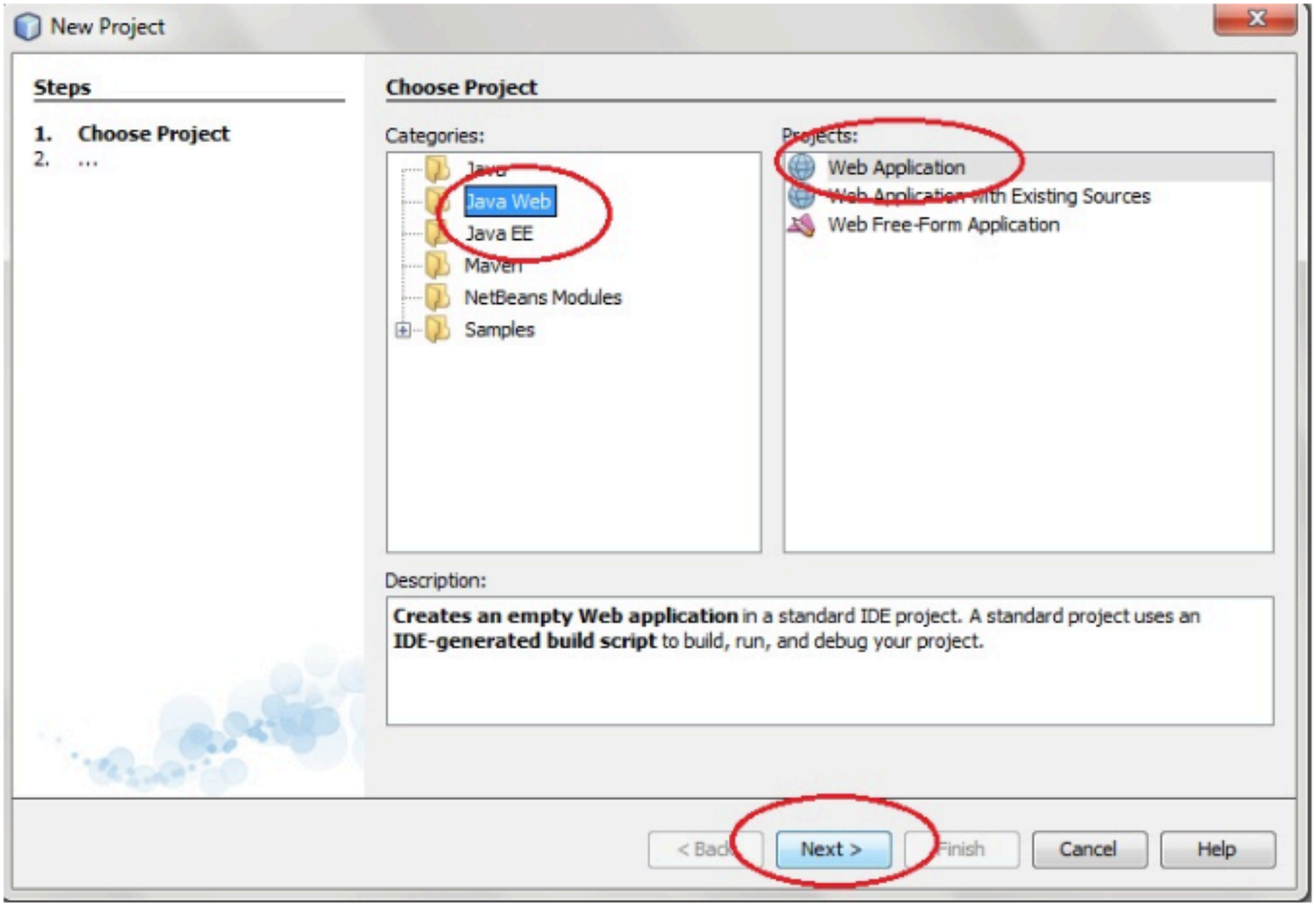
Steps to create sevlet application in Netbeans IDE

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

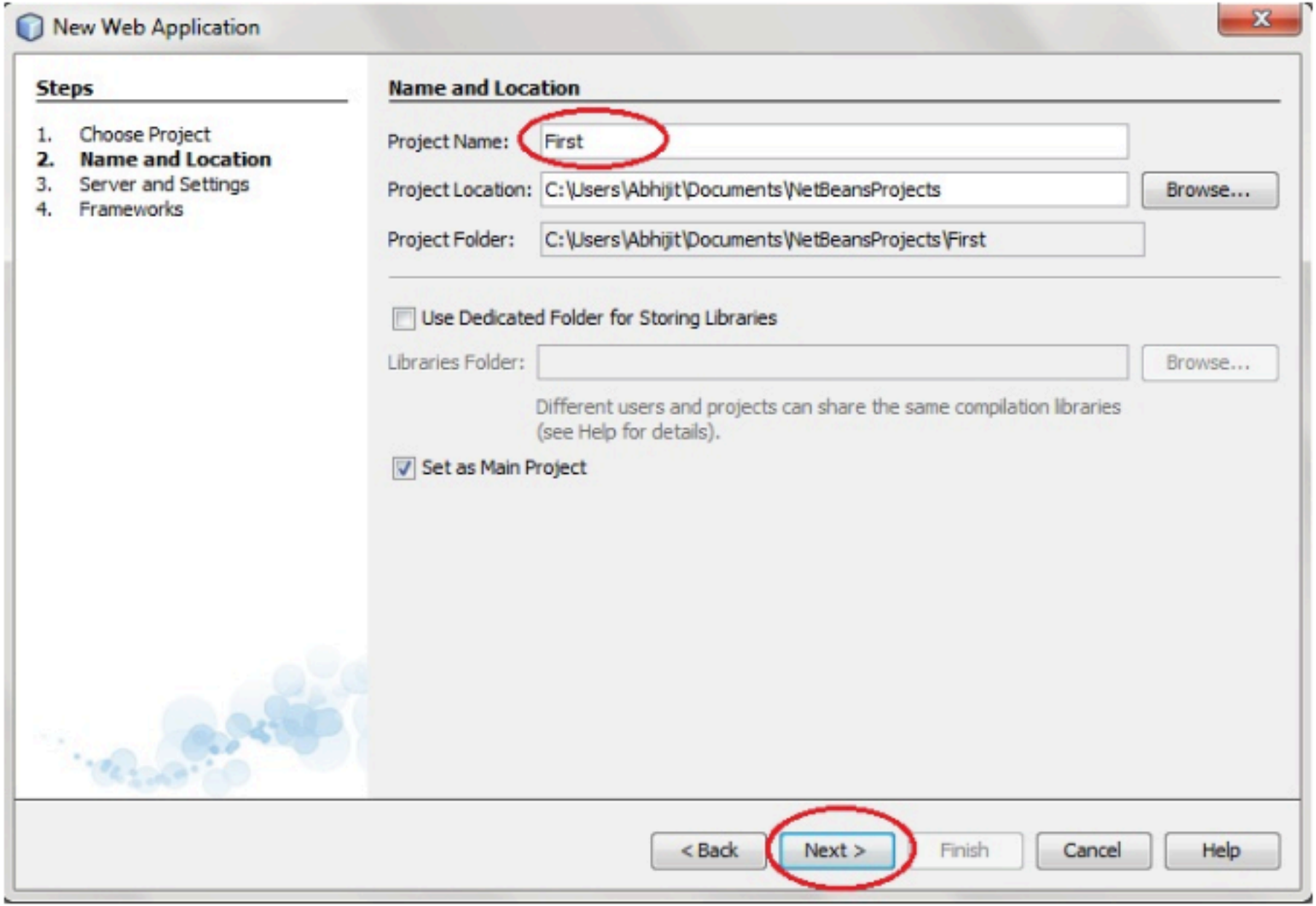
1. Open Netbeans, Select file -> New Project



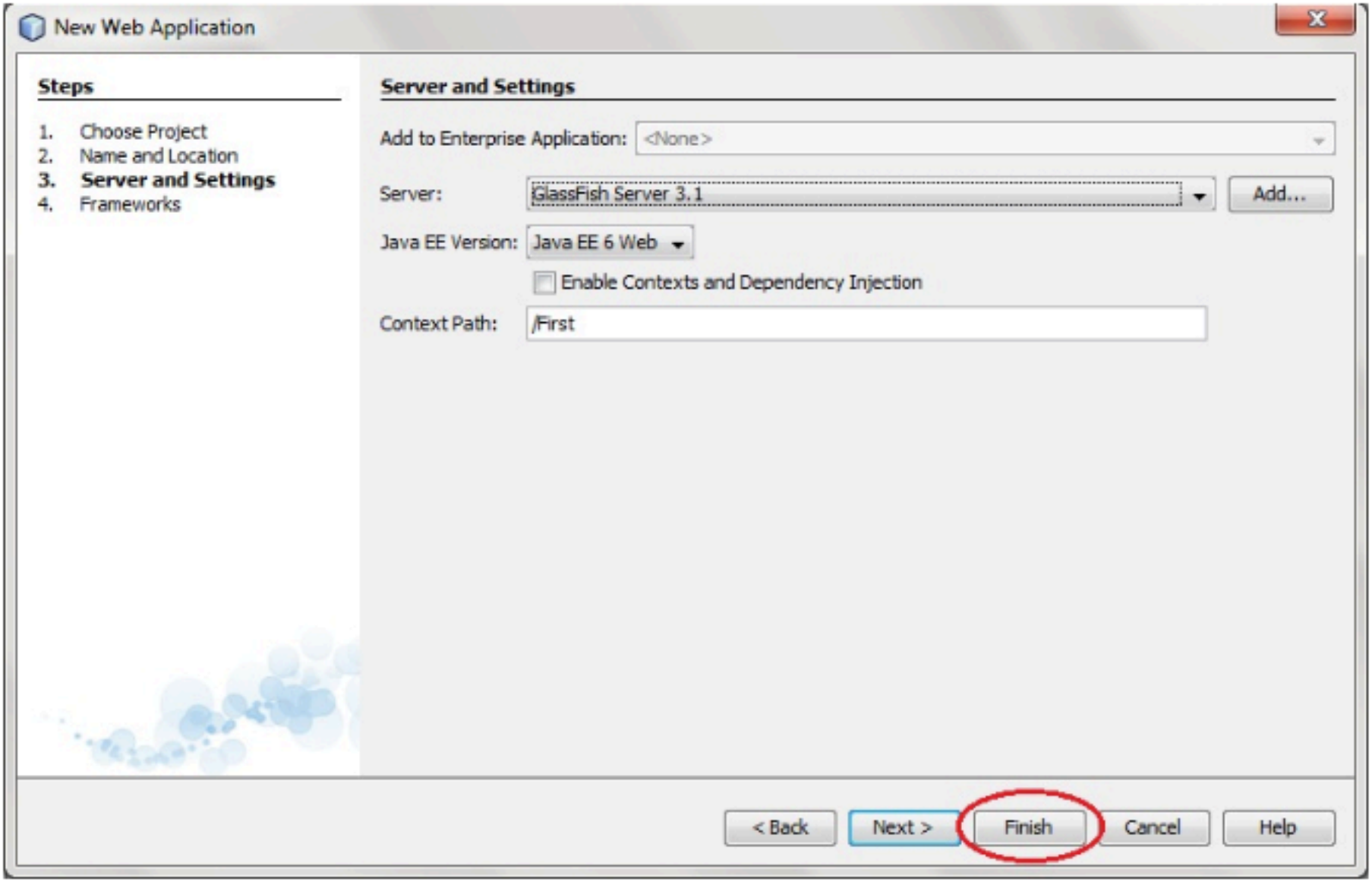
2. Select Java Web -> Web Application, then click on next

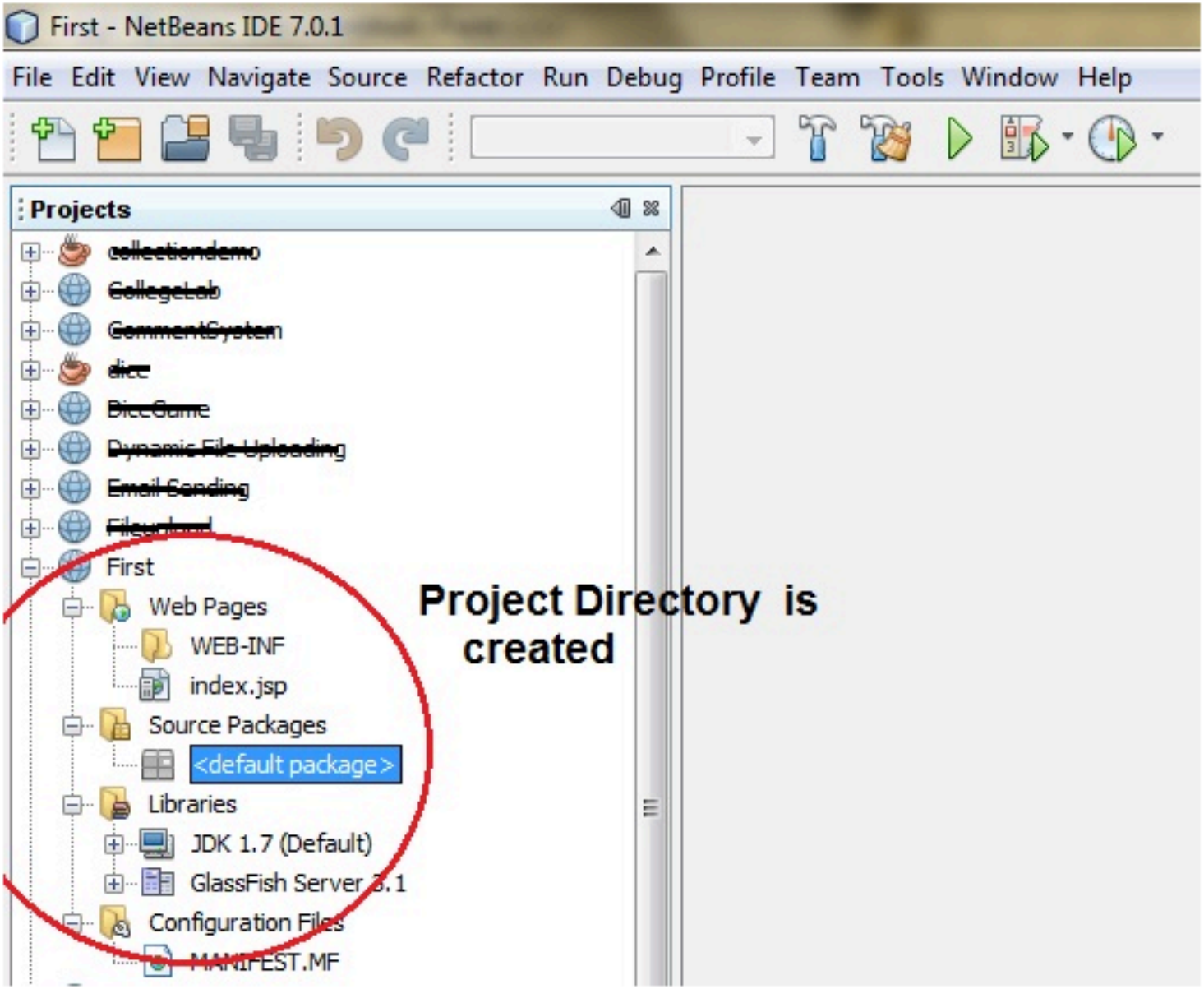


3. Give a name to your project and click next

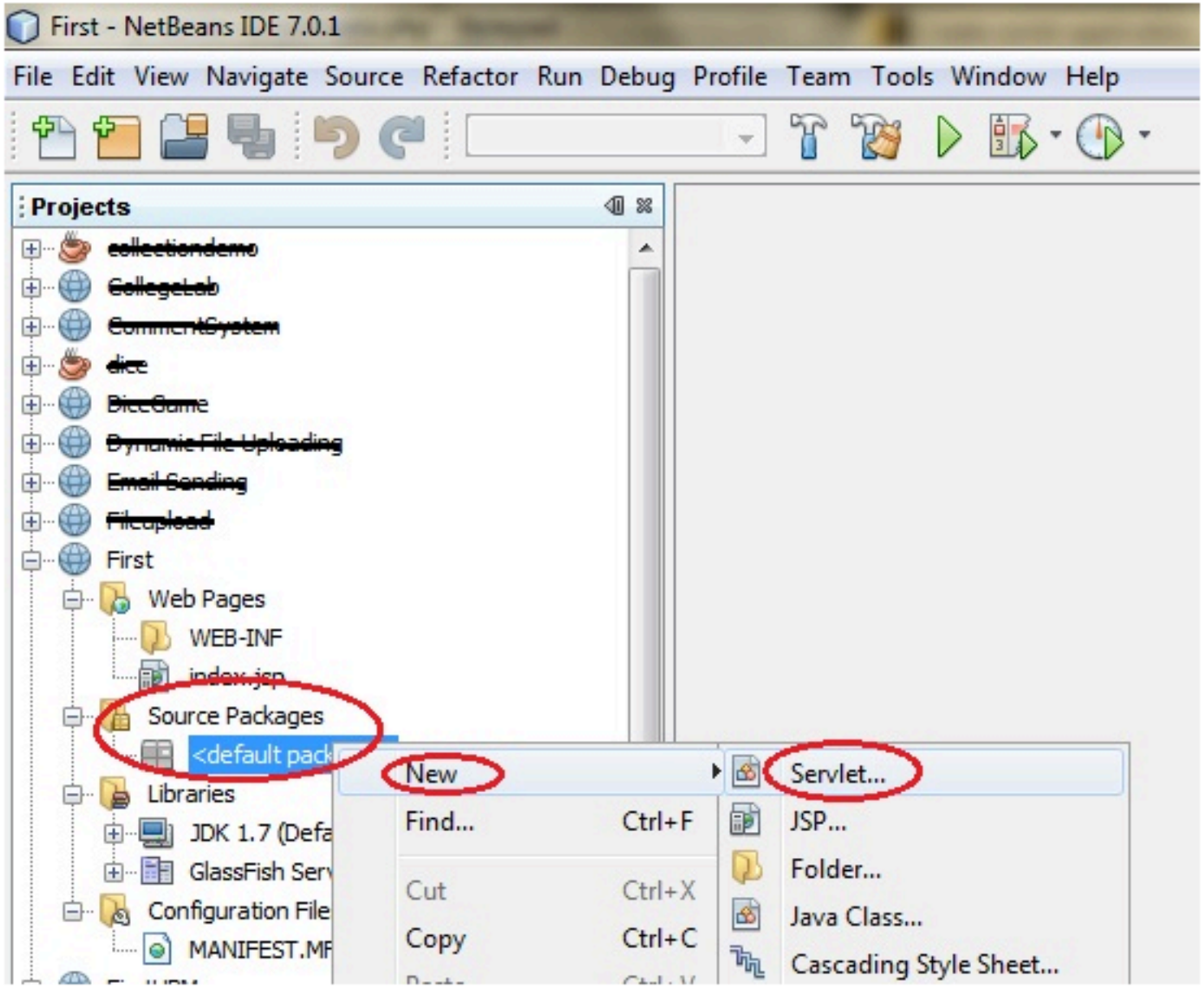


4. Click finish

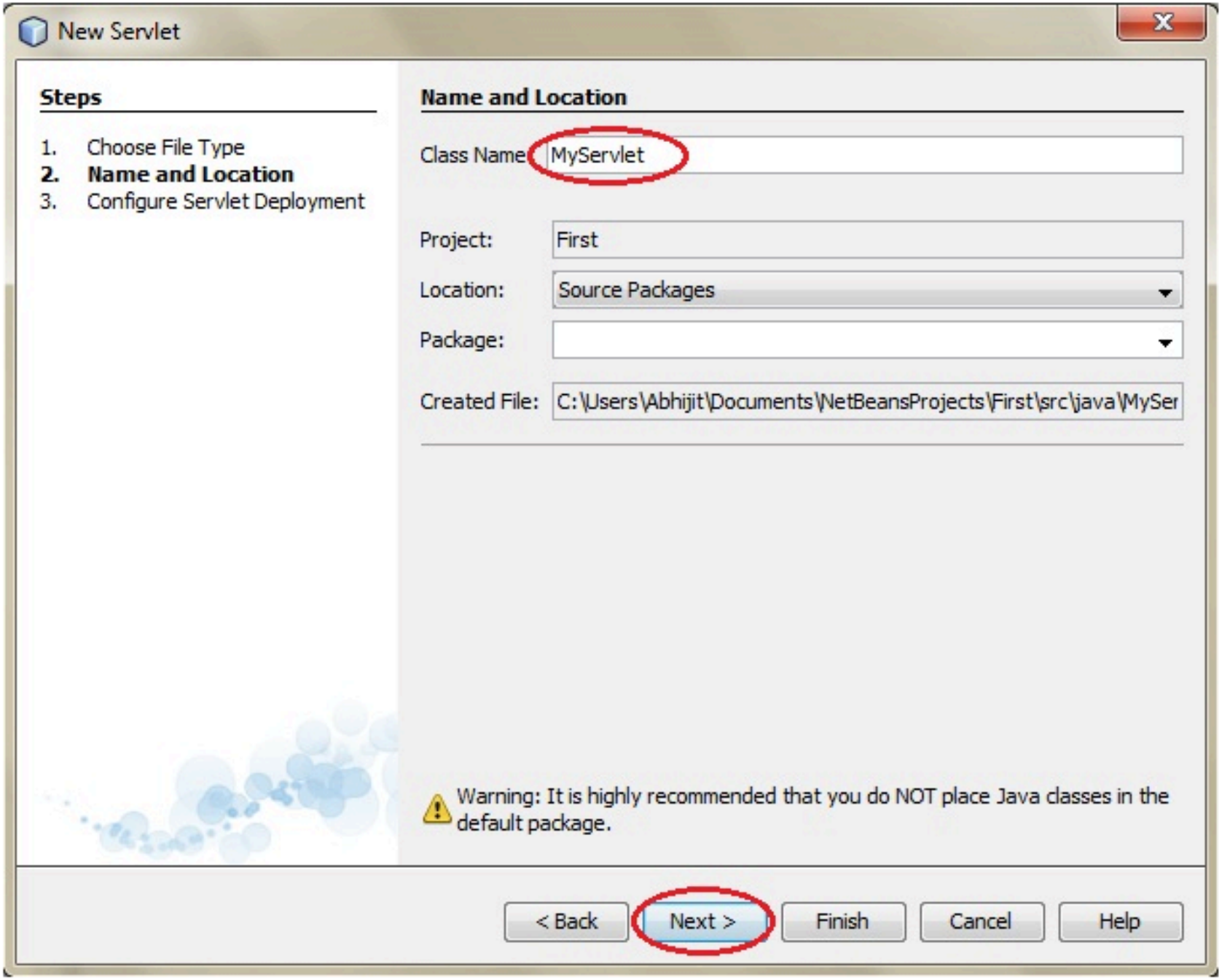


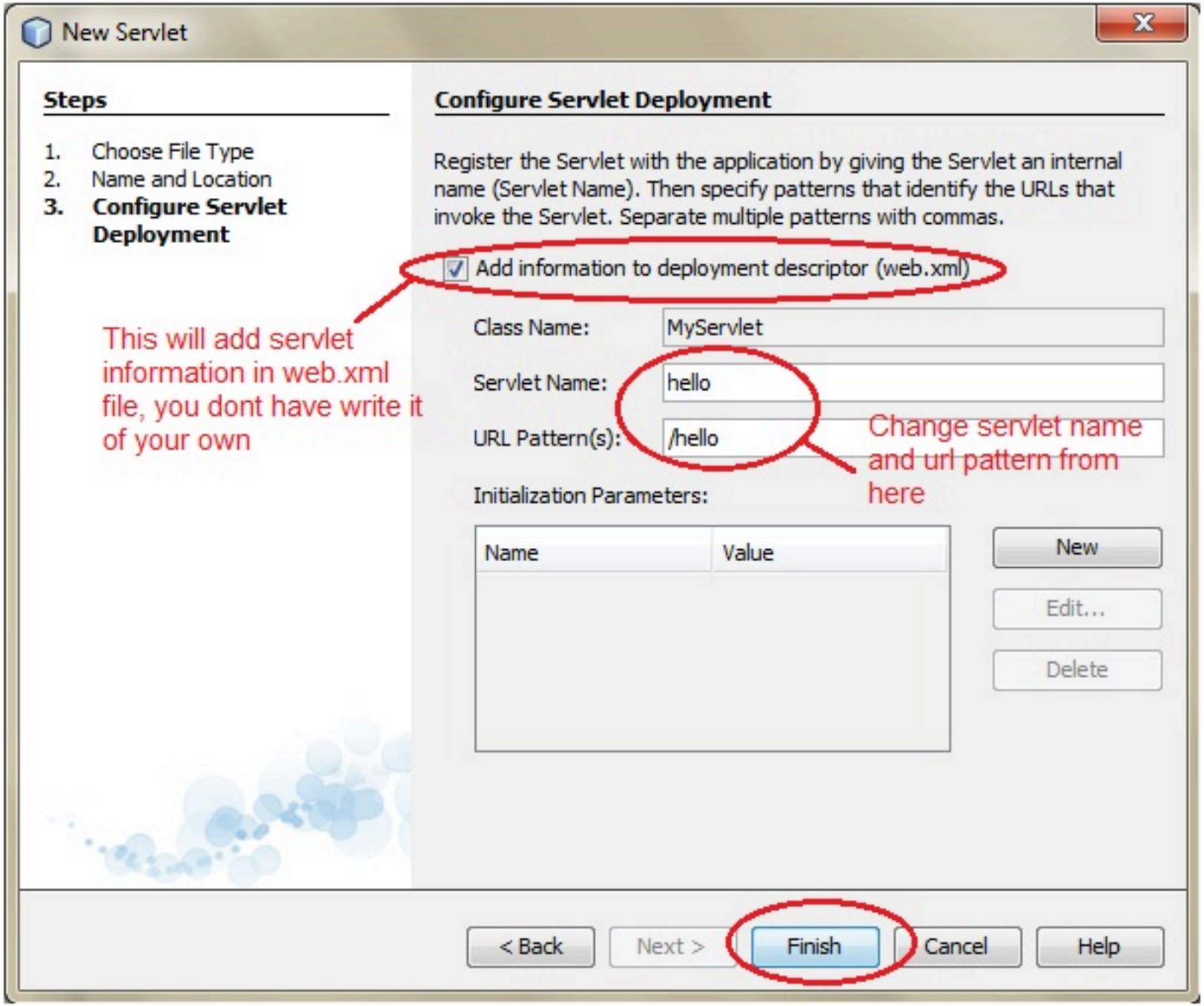

First - NetBeans IDE 7.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
collectiondemo
CollegeLab
CommentSystem
dice
DiceGame
Dynamic File Uploading
Email Sending
FileUpload
First
Web Pages
WEB-INF
index.jsp
Source Packages
<default package>
Libraries
JDK 1.7 (Default)
GlassFish Server 3.1
Configuration Files
MANIFEST.MF
Project Directory is created

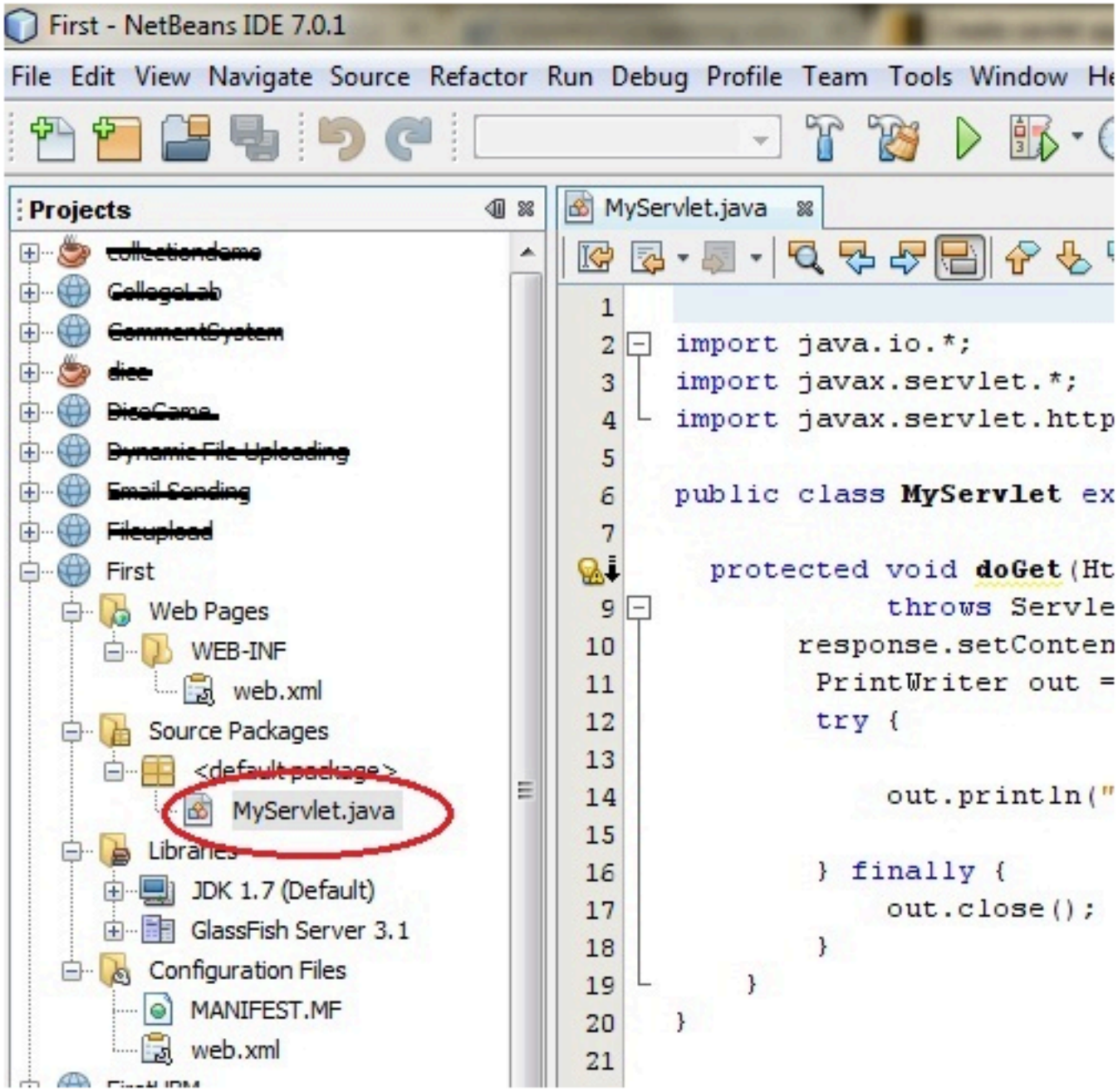
5. To create a servlet, open **Source Package**, right click on **default packages -> New -> Servlet**



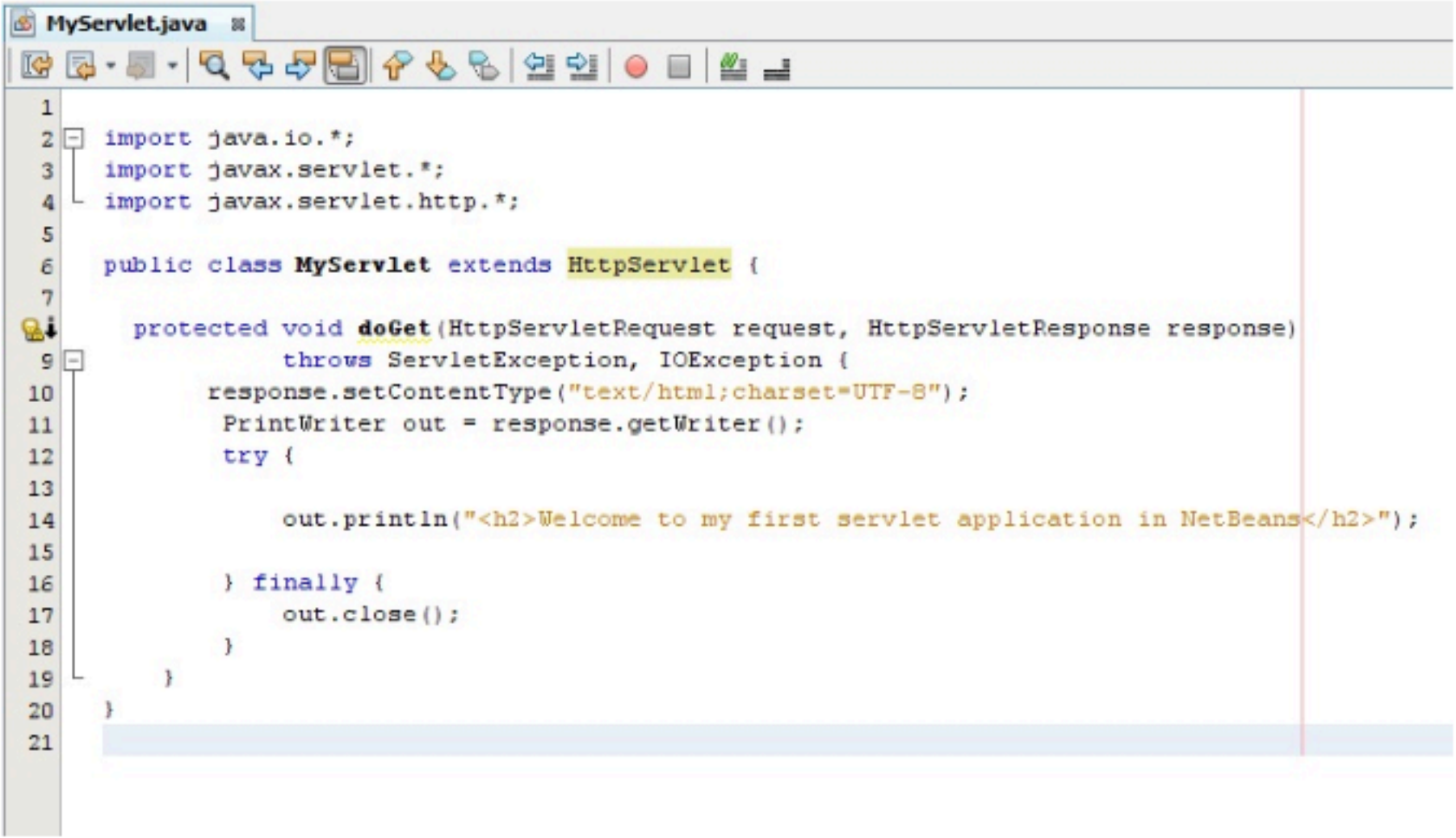
6. Give a name to your servlet class file



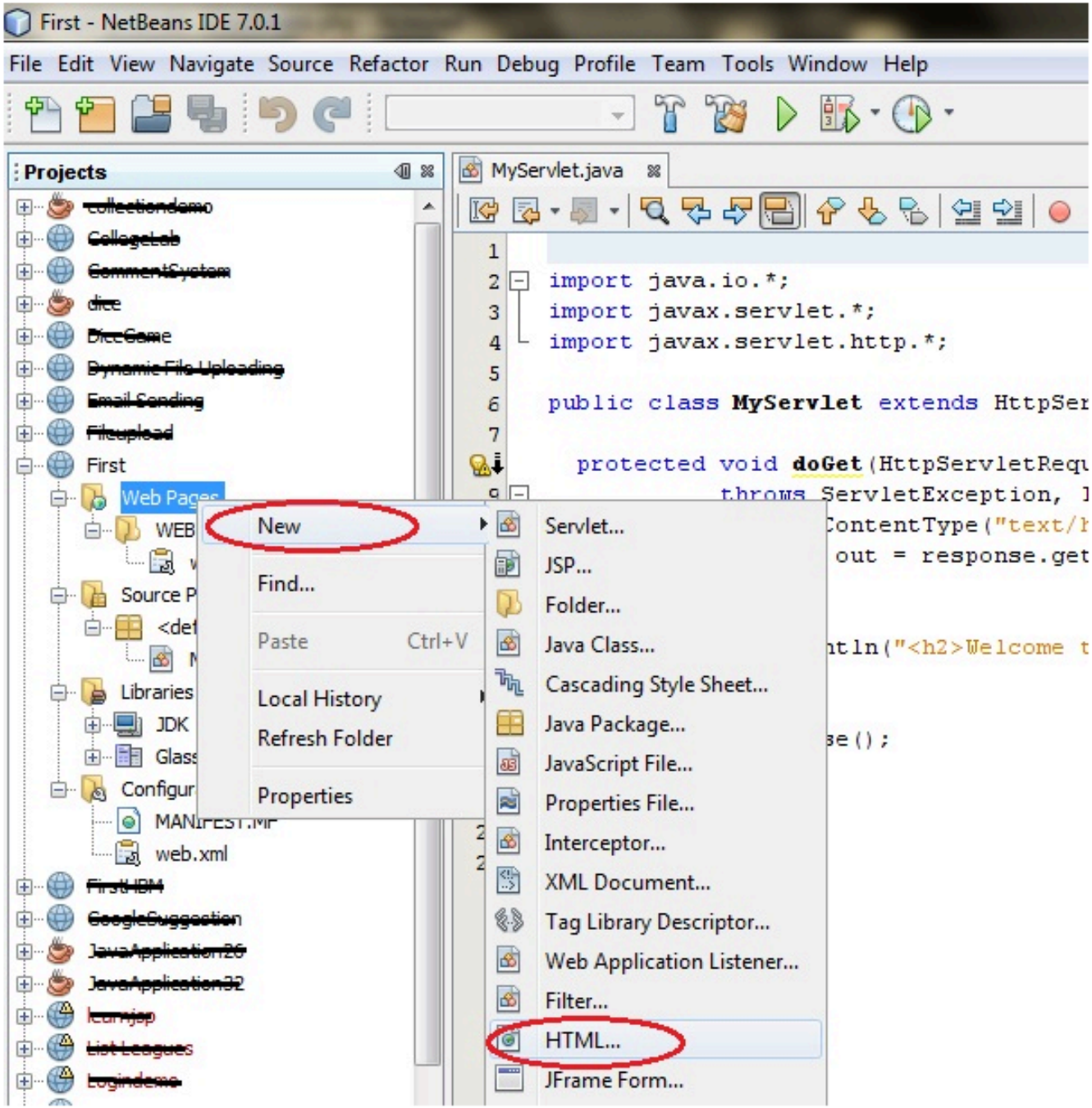

New Servlet
Steps
1. Choose File Type
2. Name and Location
3. Configure Servlet Deployment
Configure Servlet Deployment
Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.
Add information to deployment descriptor (web.xml)
This will add servlet information in web.xml file, you dont have write it of your own
Class Name: MyServlet
Servlet Name: hello
URL Pattern(s): /hello
Change servlet name and url pattern from here
Initialization Parameters:
Name
Value
New
Edit...
Delete
< Back
Next >
Finish
Cancel
Help

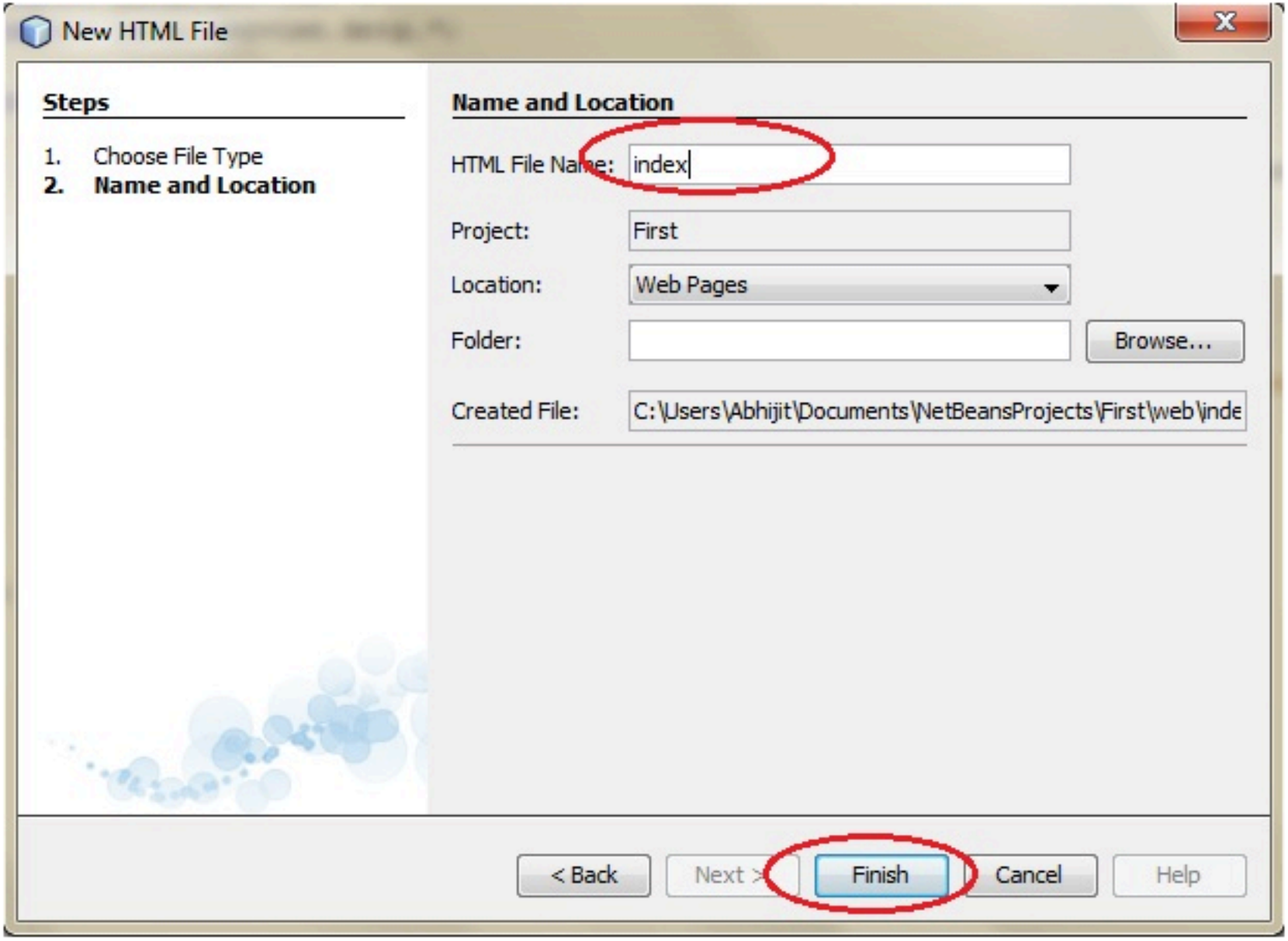

First - NetBeans IDE 7.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window He
Projects
collectiondemo
CollegeLab
CommentSystem
dice
DiceGame
Dynamic File Uploading
Email Sending
Fileupload
First
Web Pages
WEB-INF
web.xml
Source Packages
<default package>
MyServlet.java
Libraries
JDK 1.7 (Default)
GlassFish Server 3.1
Configuration Files
MANIFEST.MF
web.xml
MyServlet.java
1
2 import java.io.*;
3 import javax.servlet.*;
4 import javax.servlet.http
5
6 public class MyServlet ex
7
protected void doGet(Ht
9 throws Servle
10 response.setConten
11 PrintWriter out =
12 try {
13
14 out.println("
15
16 } finally {
17 out.close();
18 }
19 }
20 }
21

7. Write some code inside your servlet class

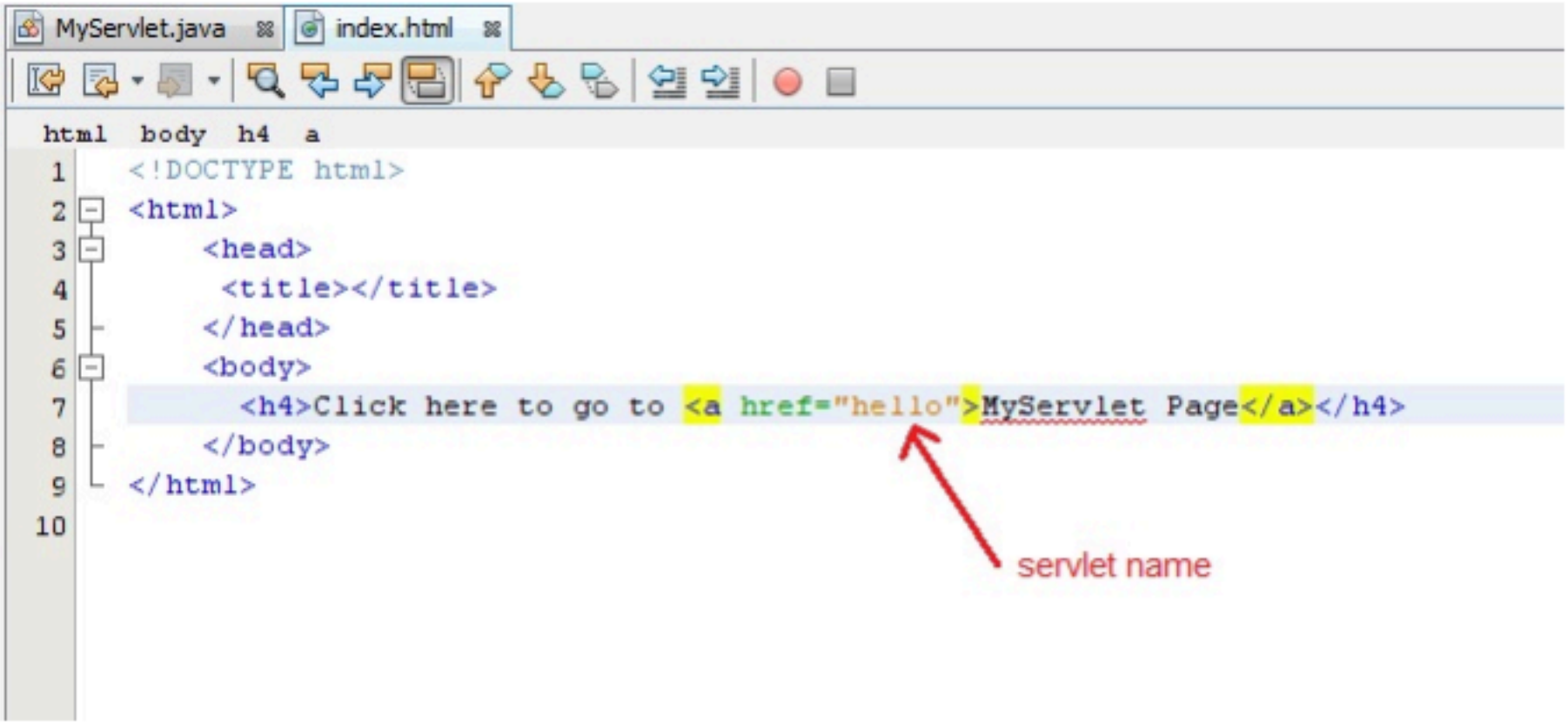


8. Create a html file, right click on **Web Pages -> New -> HTML**

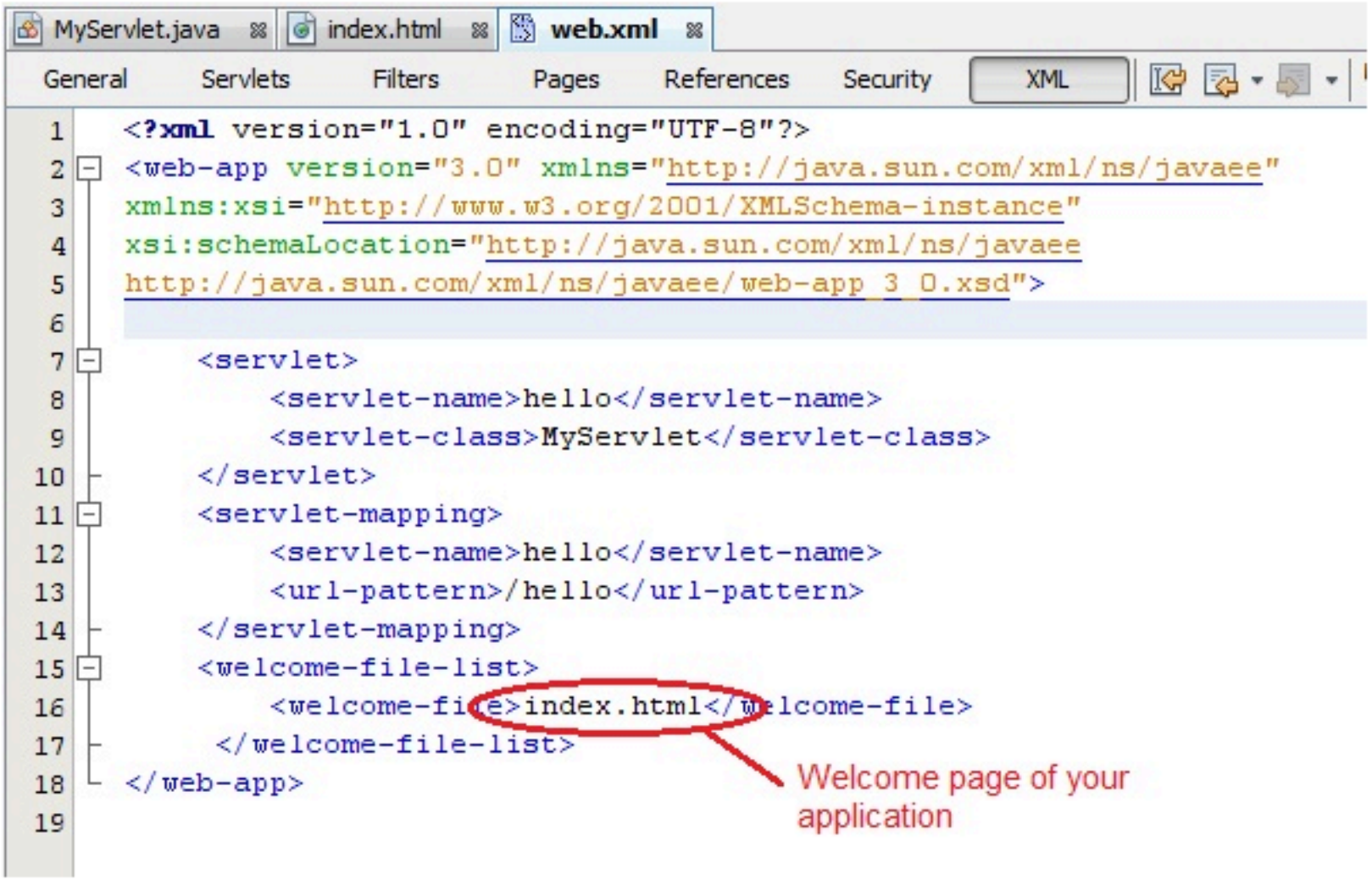



New HTML File
Steps
1. Choose File Type
2. Name and Location
Name and Location
HTML File Name: index
Project: First
Location: Web Pages
Folder:
Browse...
Created File: C:\Users\Abhijit\Documents\NetBeansProjects\First\web\inde
< Back
Next >
Finish
Cancel
Help

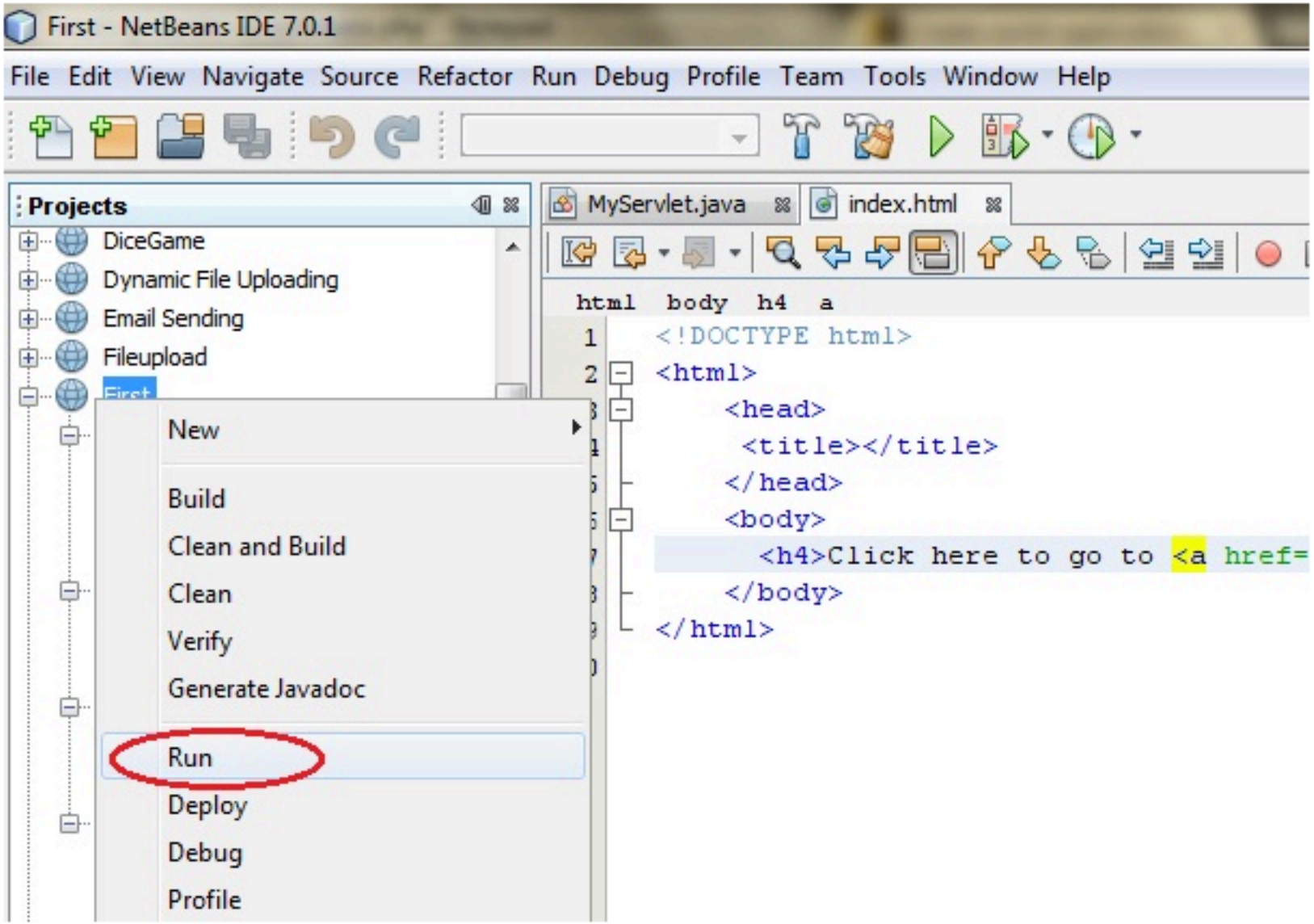
9. Write some code inside your html file

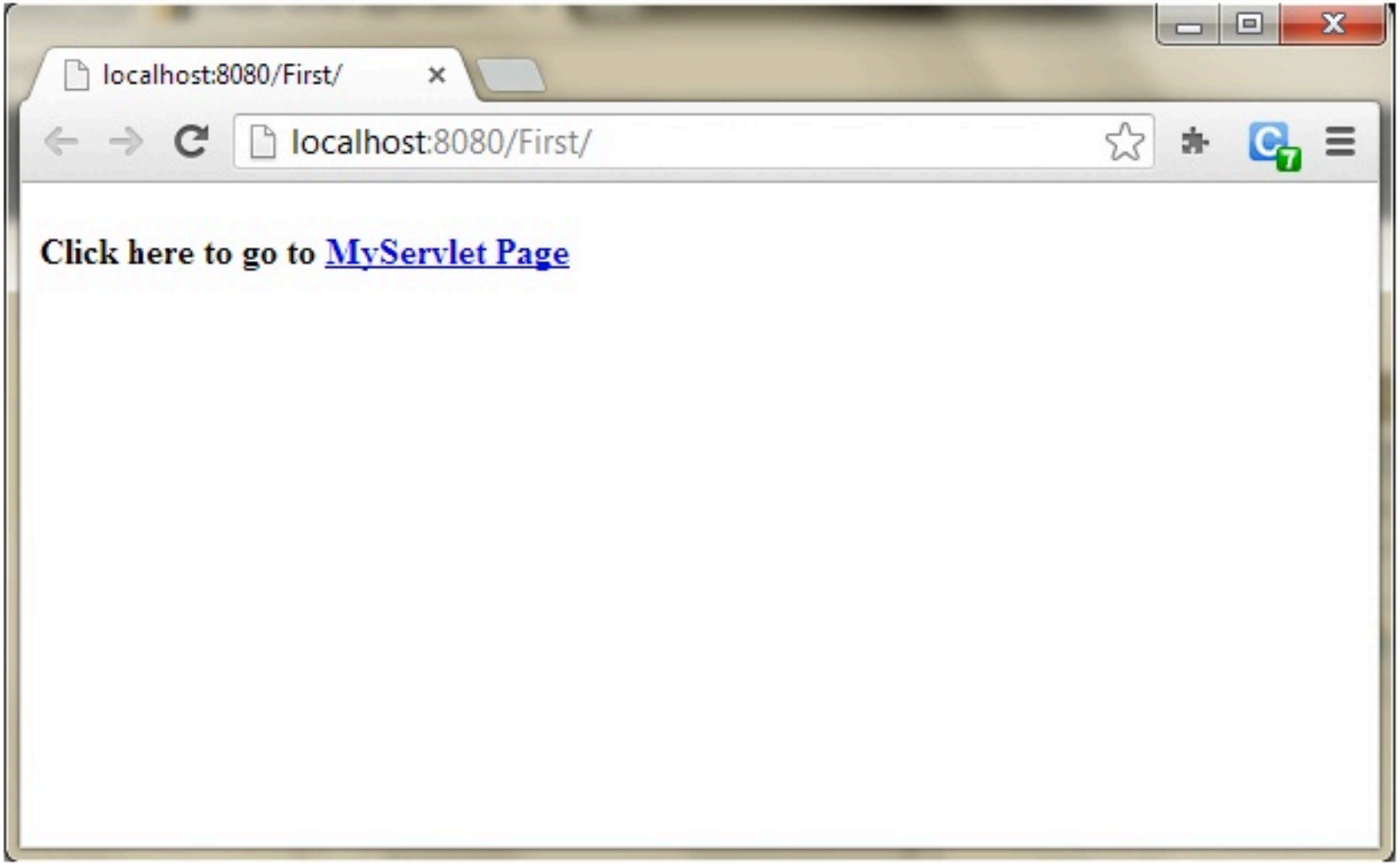


10. Edit **web.xml** file



11. Run your application, right click on your project and select **run**



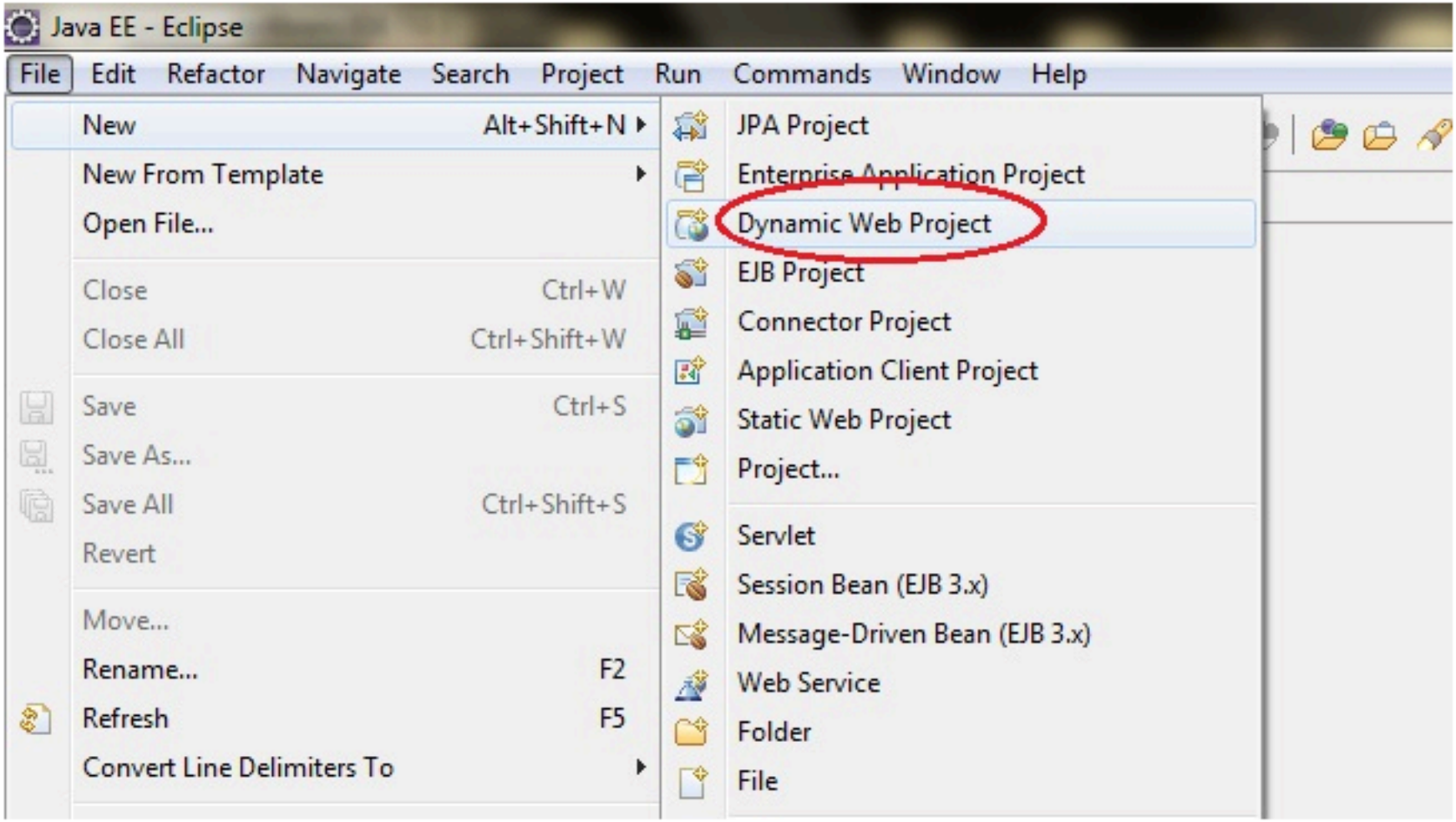

localhost:8080/First/
localhost:8080/First/
Click here to go to MyServlet Page


localhost:8080/First/hello
localhost:8080/First/hello
Welcome to my first servlet application in NetBeans

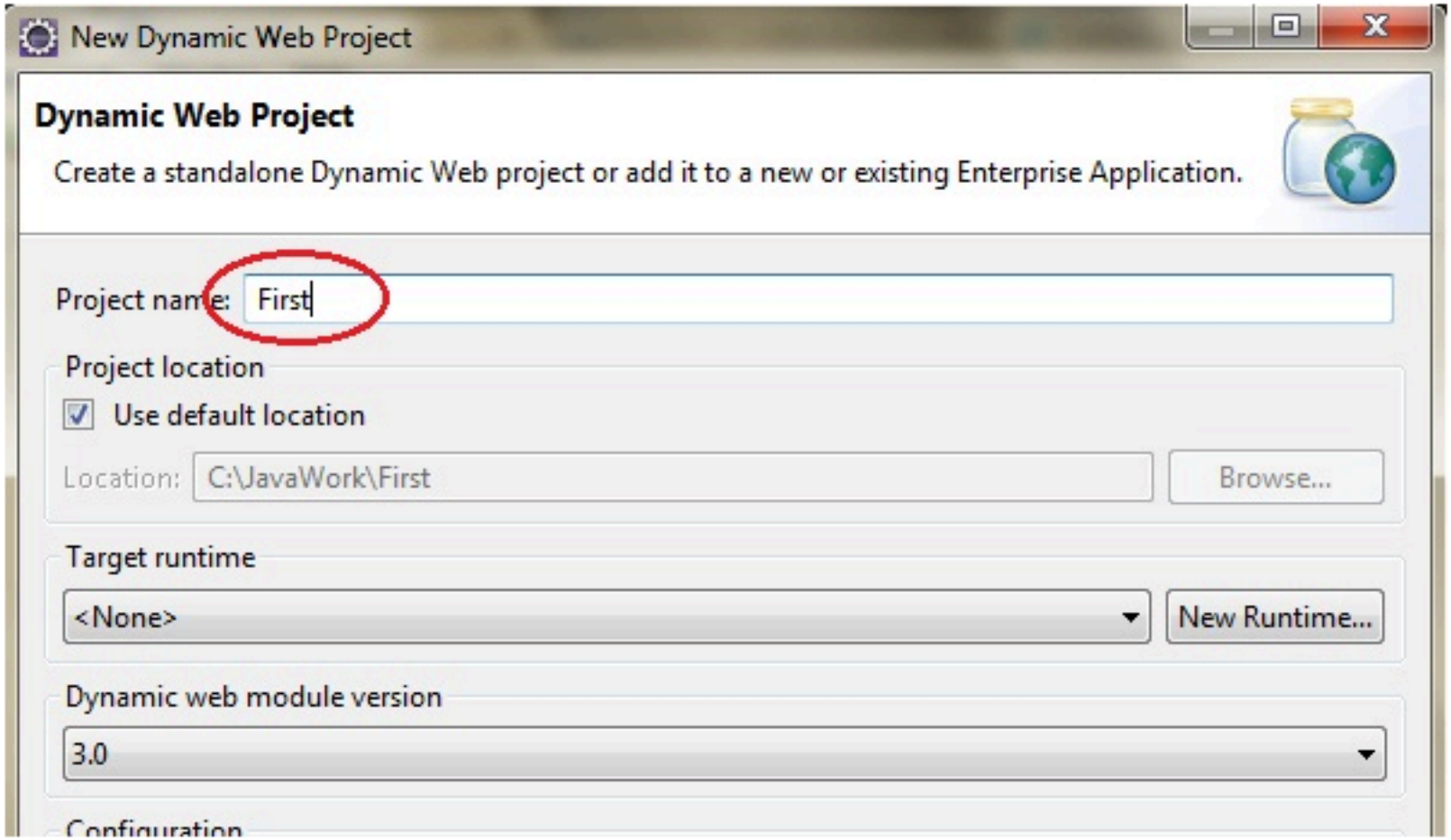
Steps to create servlet using Eclipse IDE

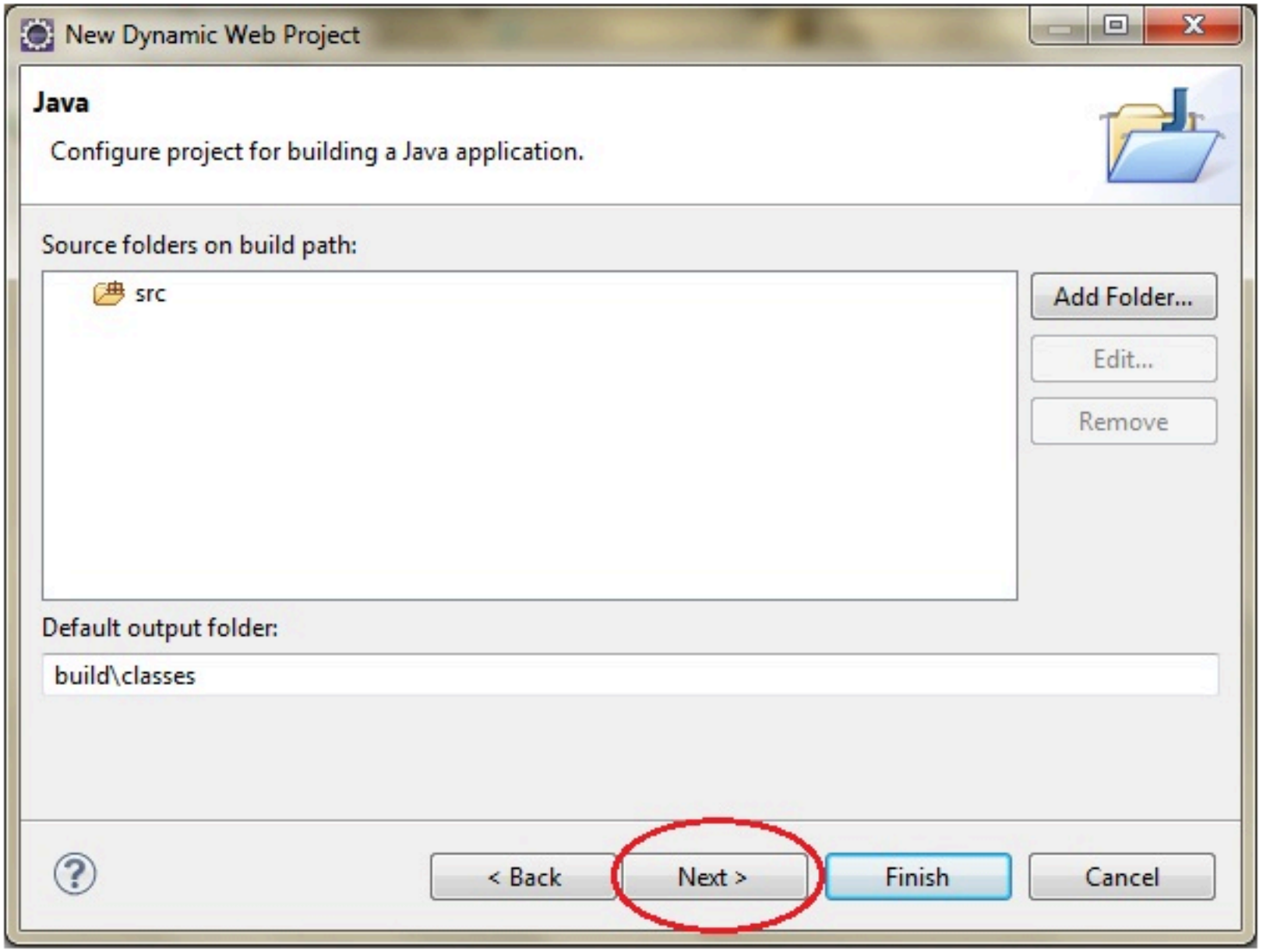
To create a servlet application in Eclipse IDE you will need to follow the following steps.

1. Goto **file -> New -> Dynamic Web Project**

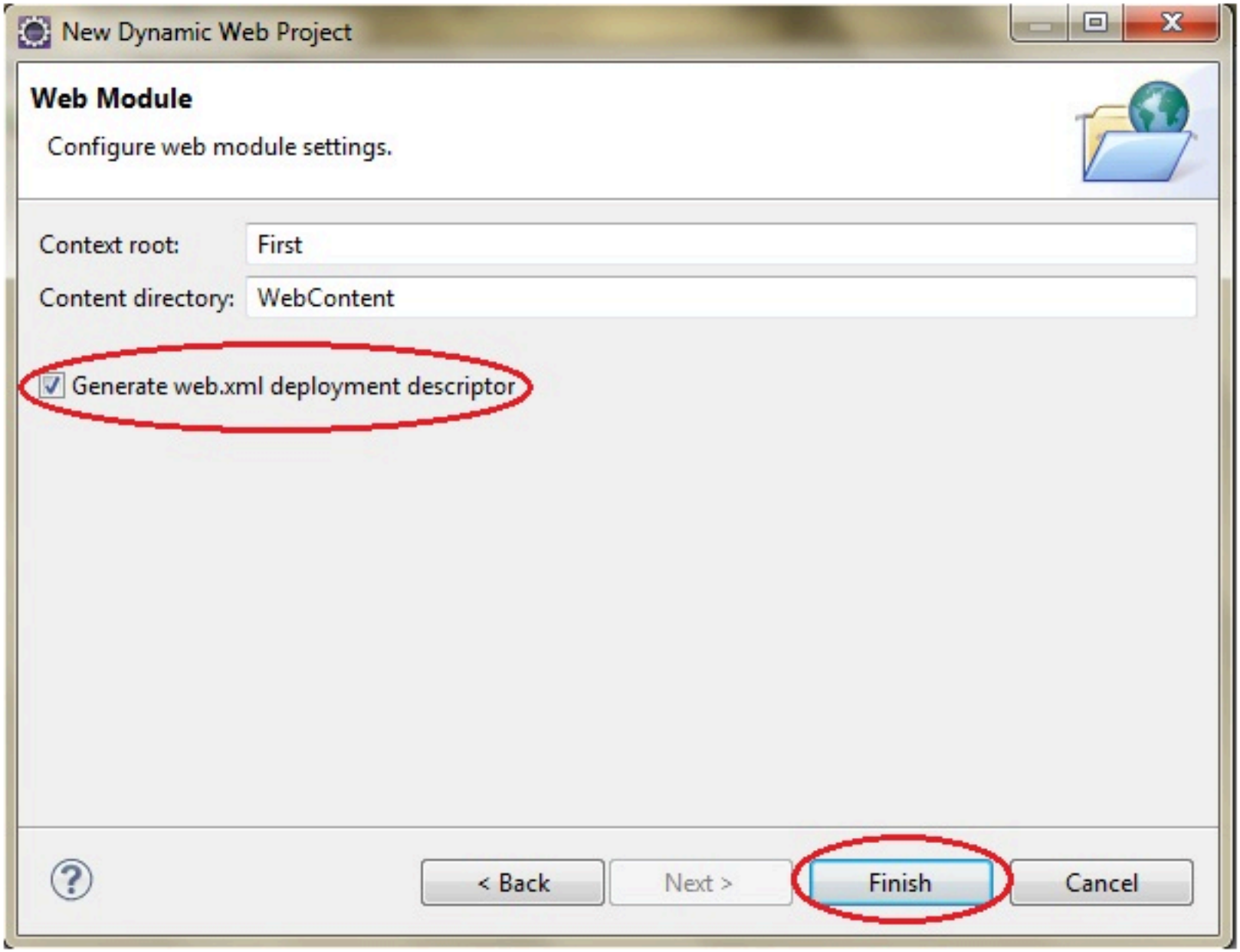


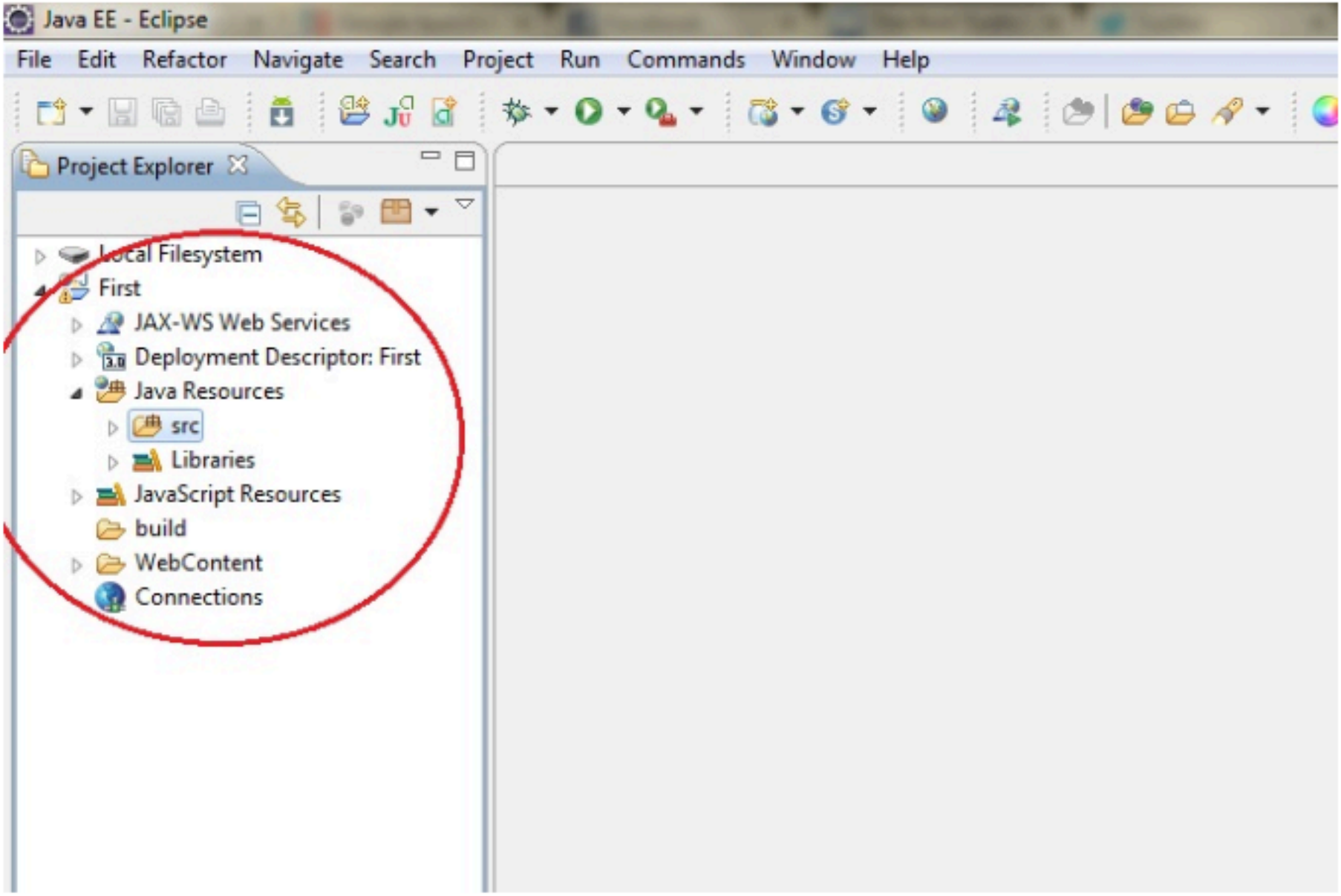
2. Give a name to your project and click **next**



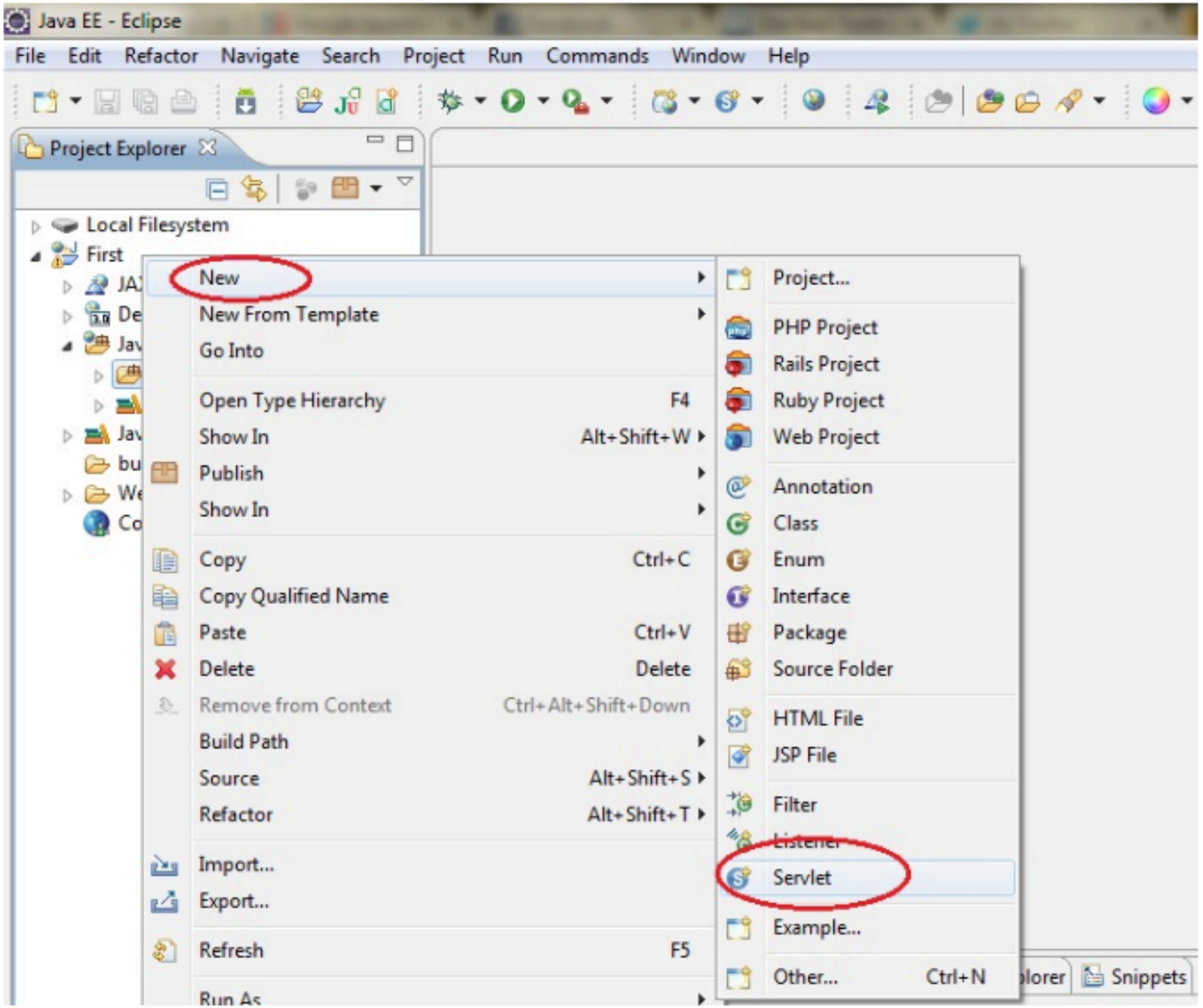

New Dynamic Web Project
Java
Configure project for building a Java application.
Source folders on build path:
src
Add Folder...
Edit...
Remove
Default output folder:
build\classes
< Back
Next >
Finish
Cancel

3. Check **Generate web.xml Deployment Descriptor** and click **finish**

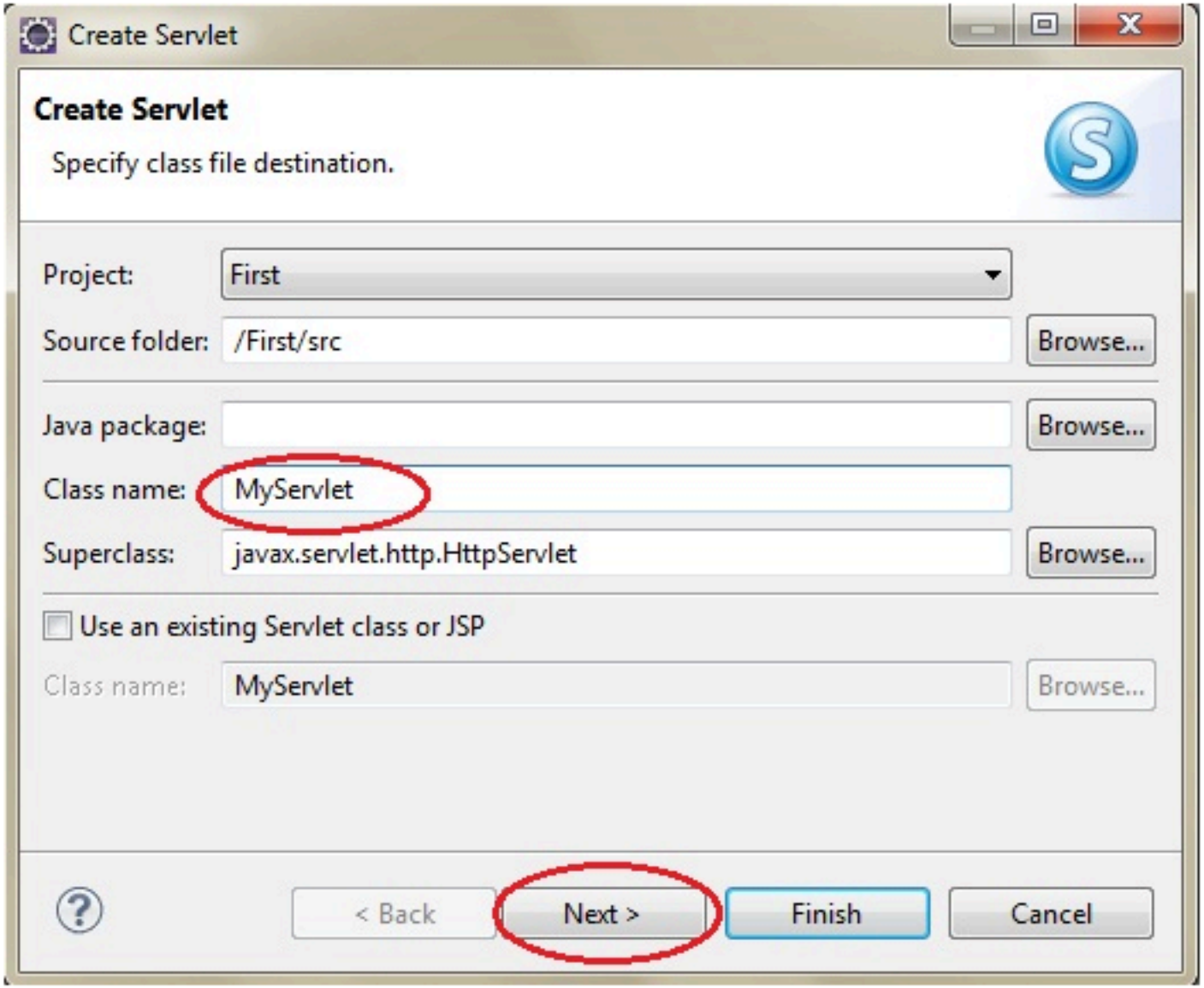


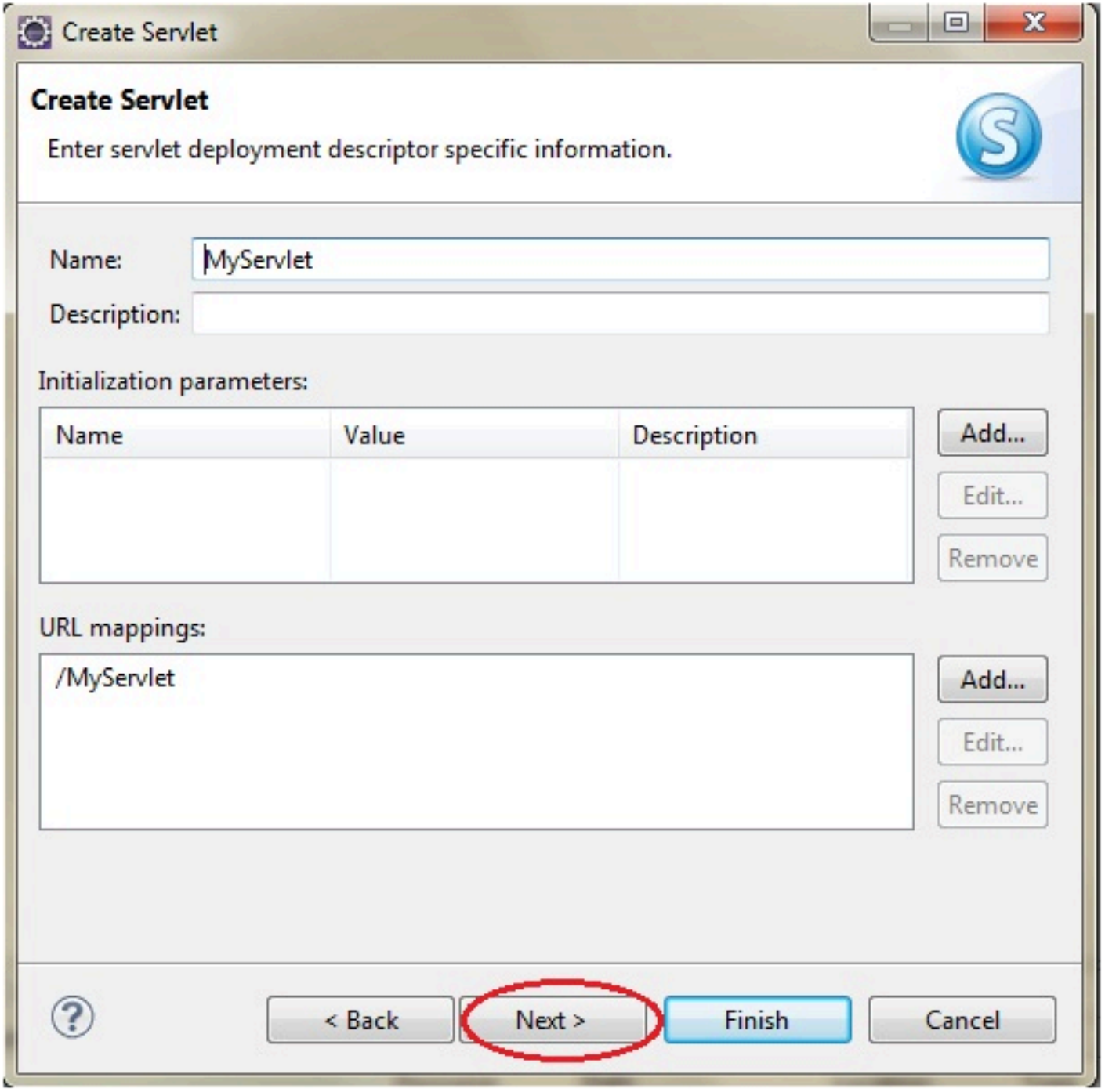

Java EE - Eclipse
File Edit Refactor Navigate Search Project Run Commands Window Help
Project Explorer
Local Filesystem
First
JAX-WS Web Services
Deployment Descriptor: First
Java Resources
src
Libraries
JavaScript Resources
build
WebContent
Connections

4. Click on **First** project, go to **Java Resources -> src**. Right click on **src** select -> **New -> Servlet**

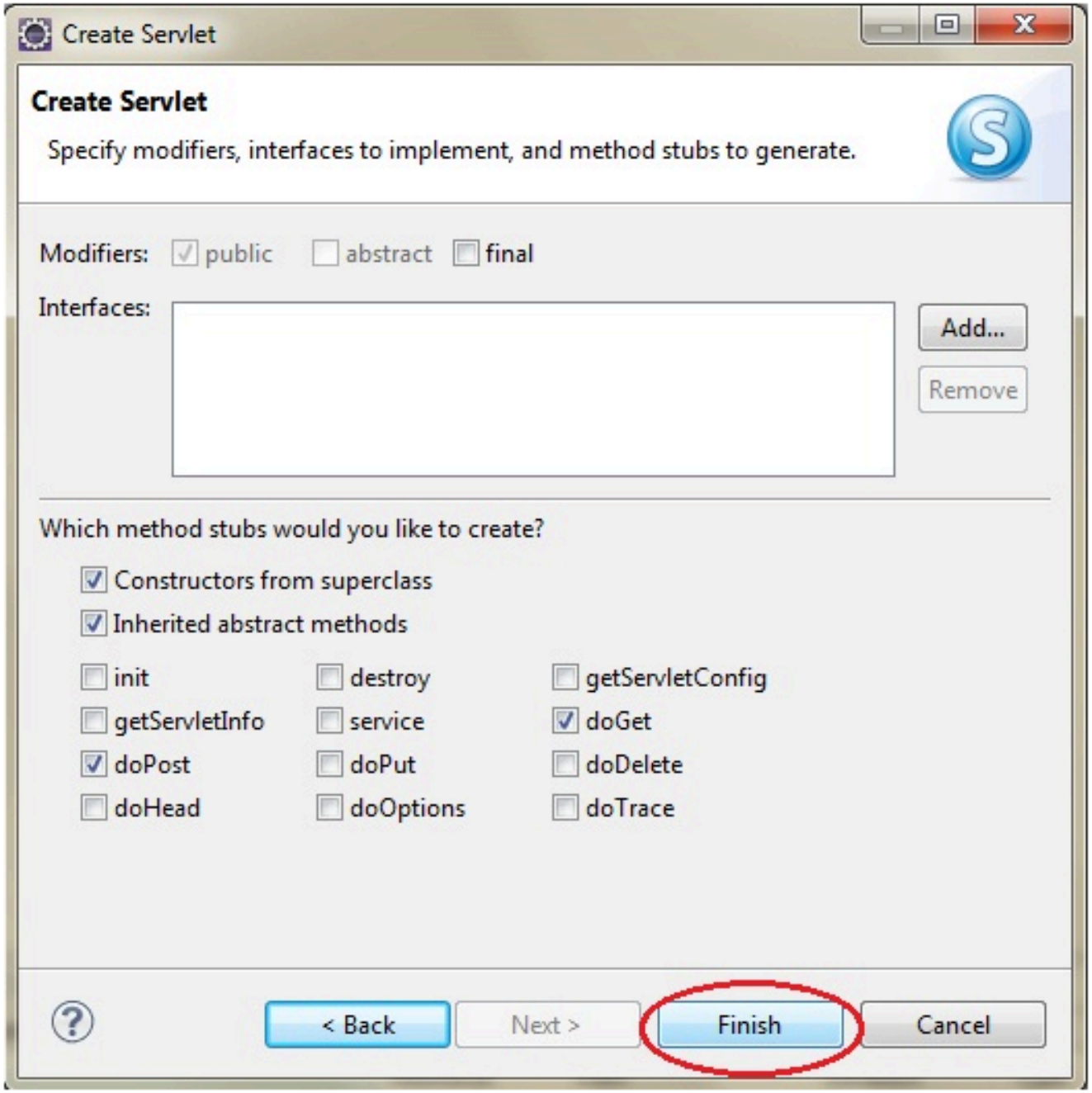


5. Give servlet class name and click **next**

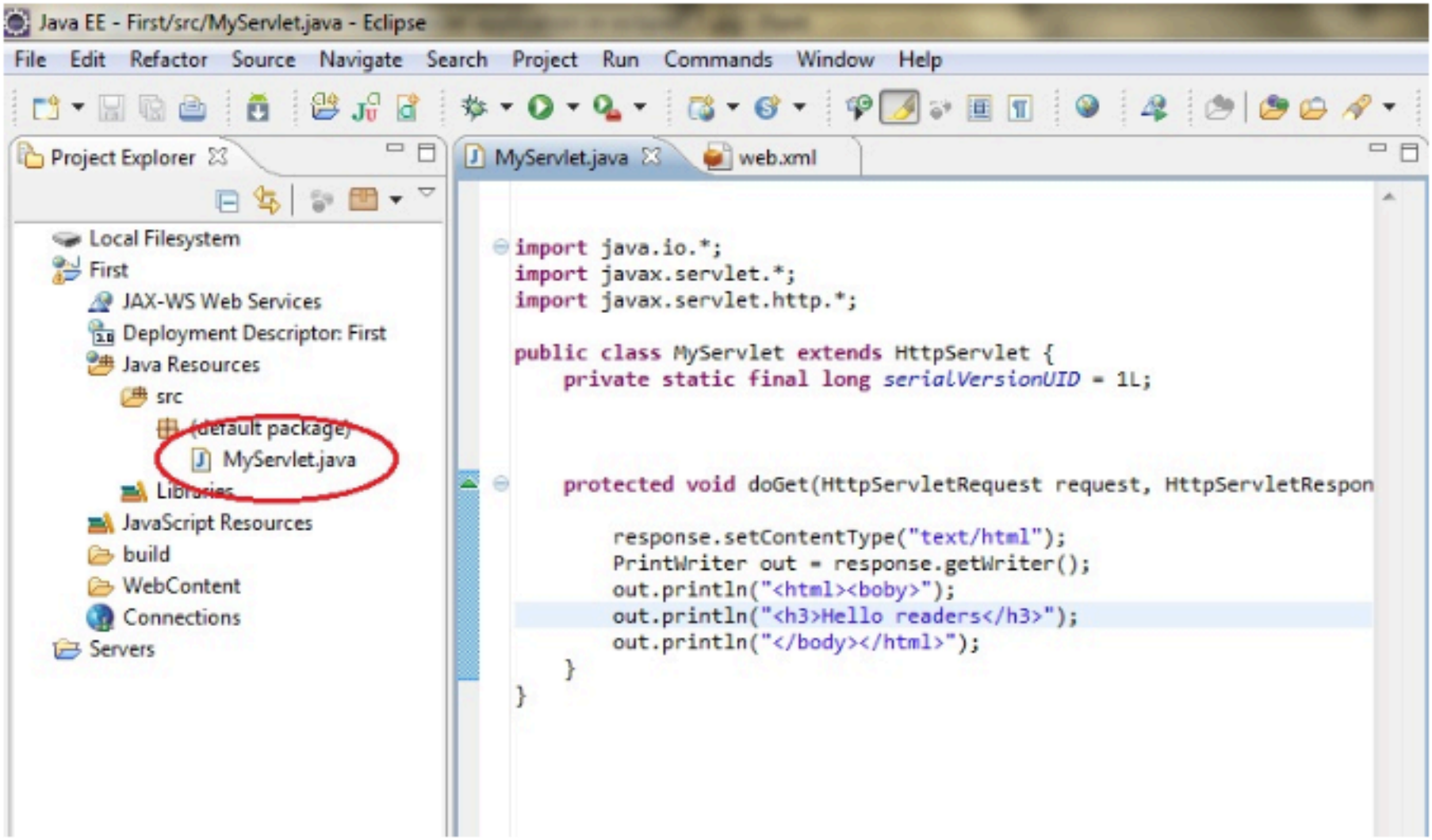



Create Servlet
Create Servlet
Enter servlet deployment descriptor specific information.
Name:
MyServlet
Description:
Initialization parameters:
Name
Value
Description
Add...
Edit...
Remove
URL mappings:
/MyServlet
Add...
Edit...
Remove
< Back
Next >
Finish
Cancel

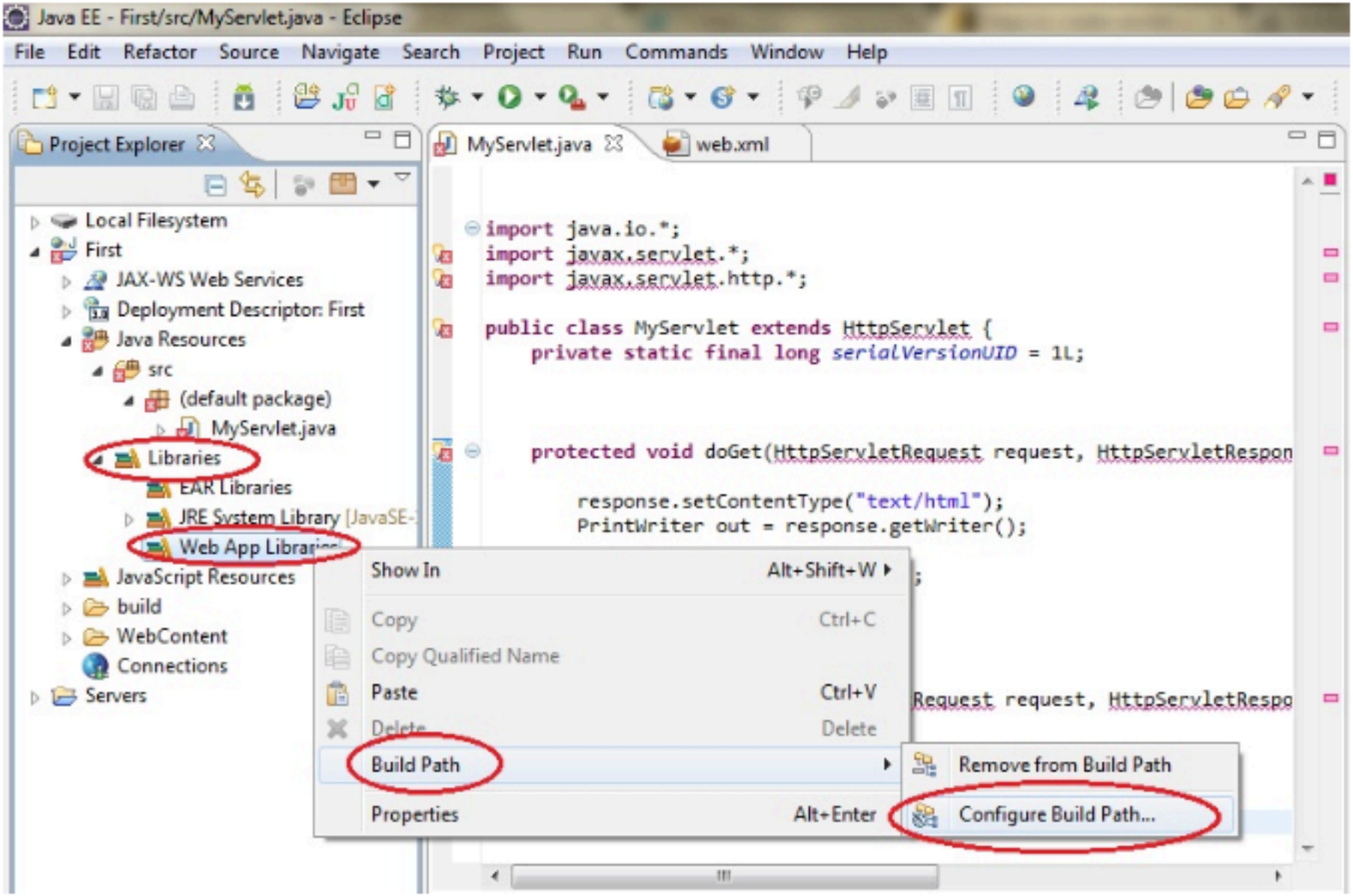
6. Leave everything else to default and click **finish**



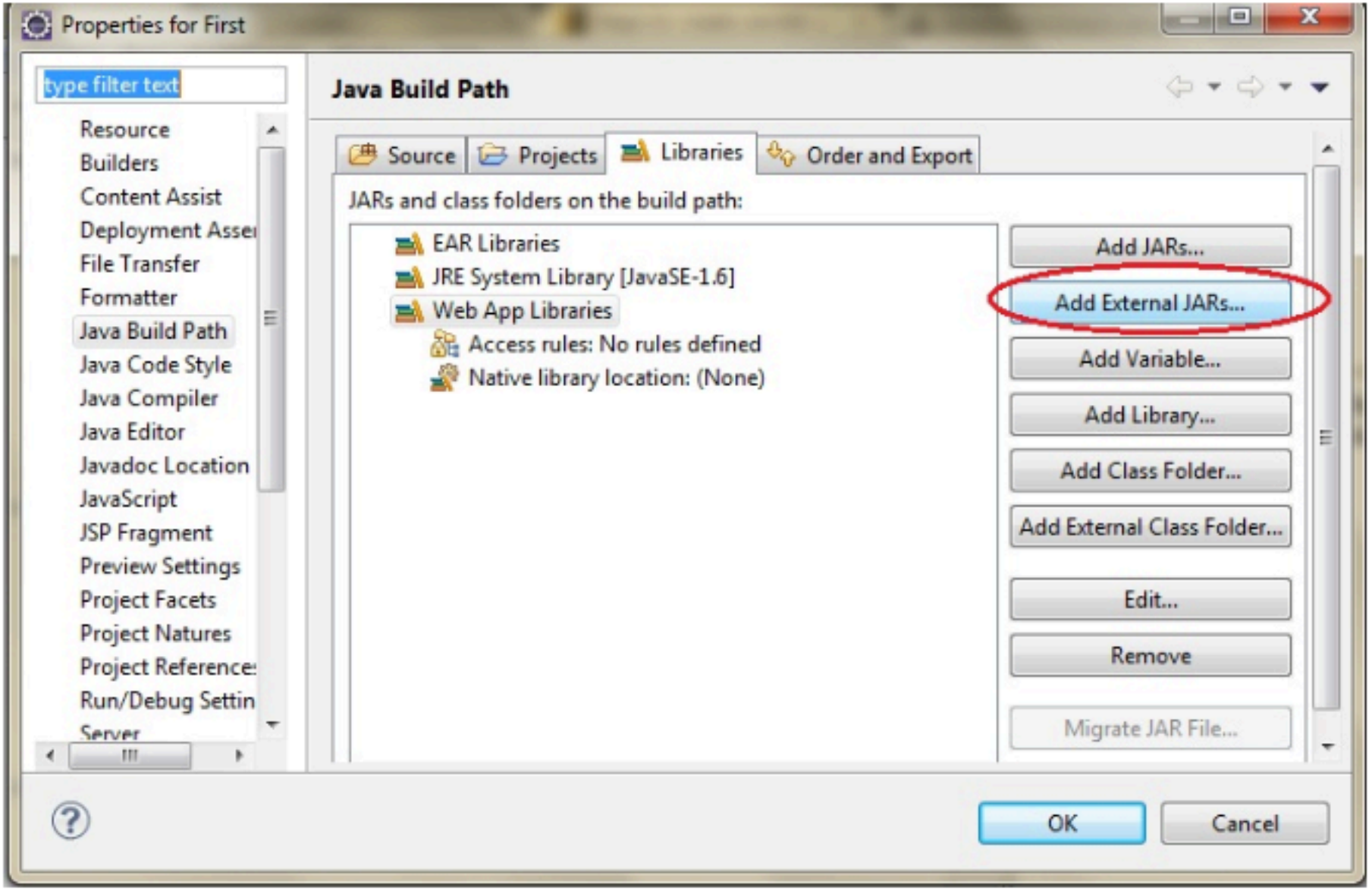
7. Now your servlet is created, write some code inside it.

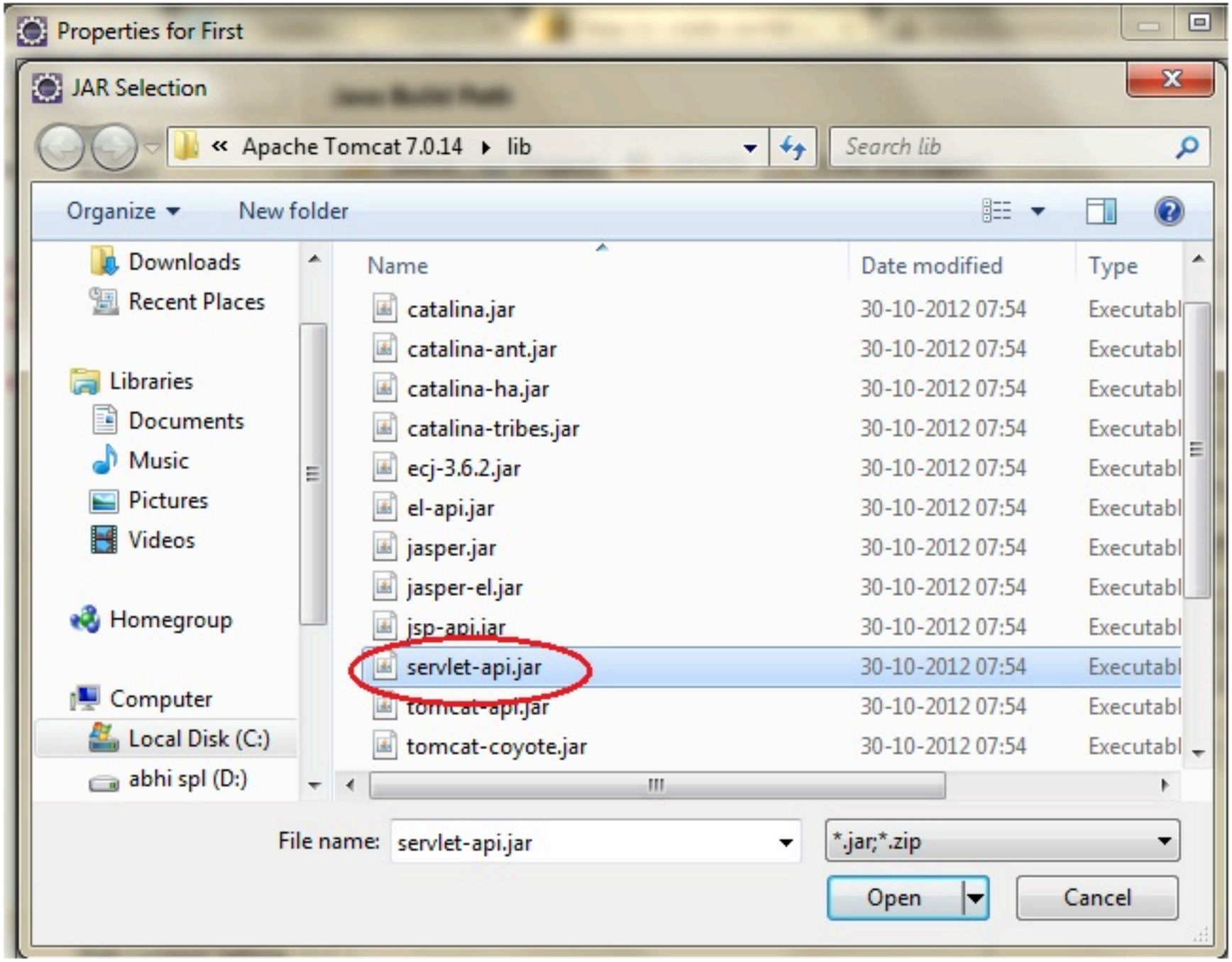


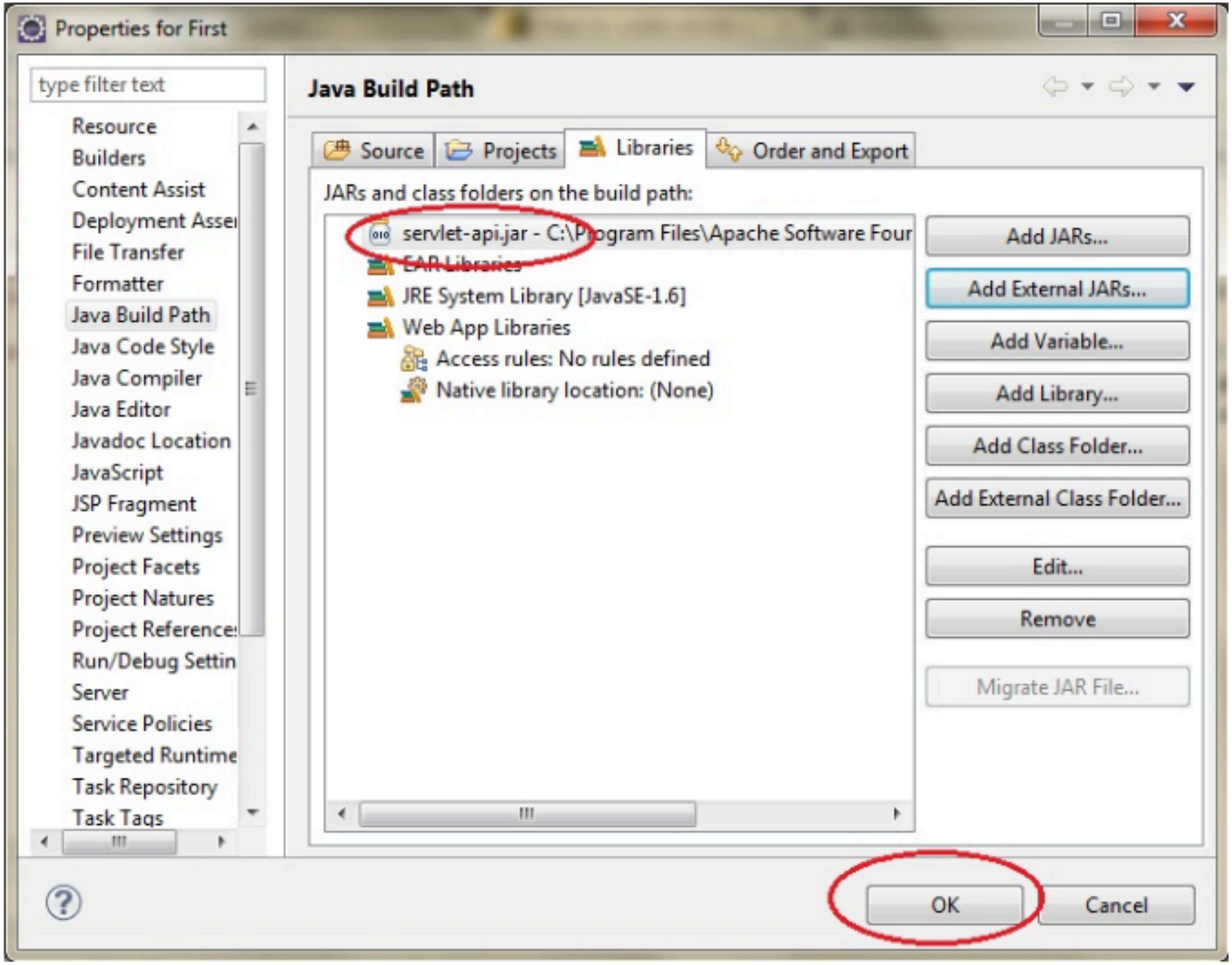
8. Add **servlet-api.jar** file to your project. Click in **Libraries**, right click on **Web App Libraries** select **Build Path -> Configure Build Path**



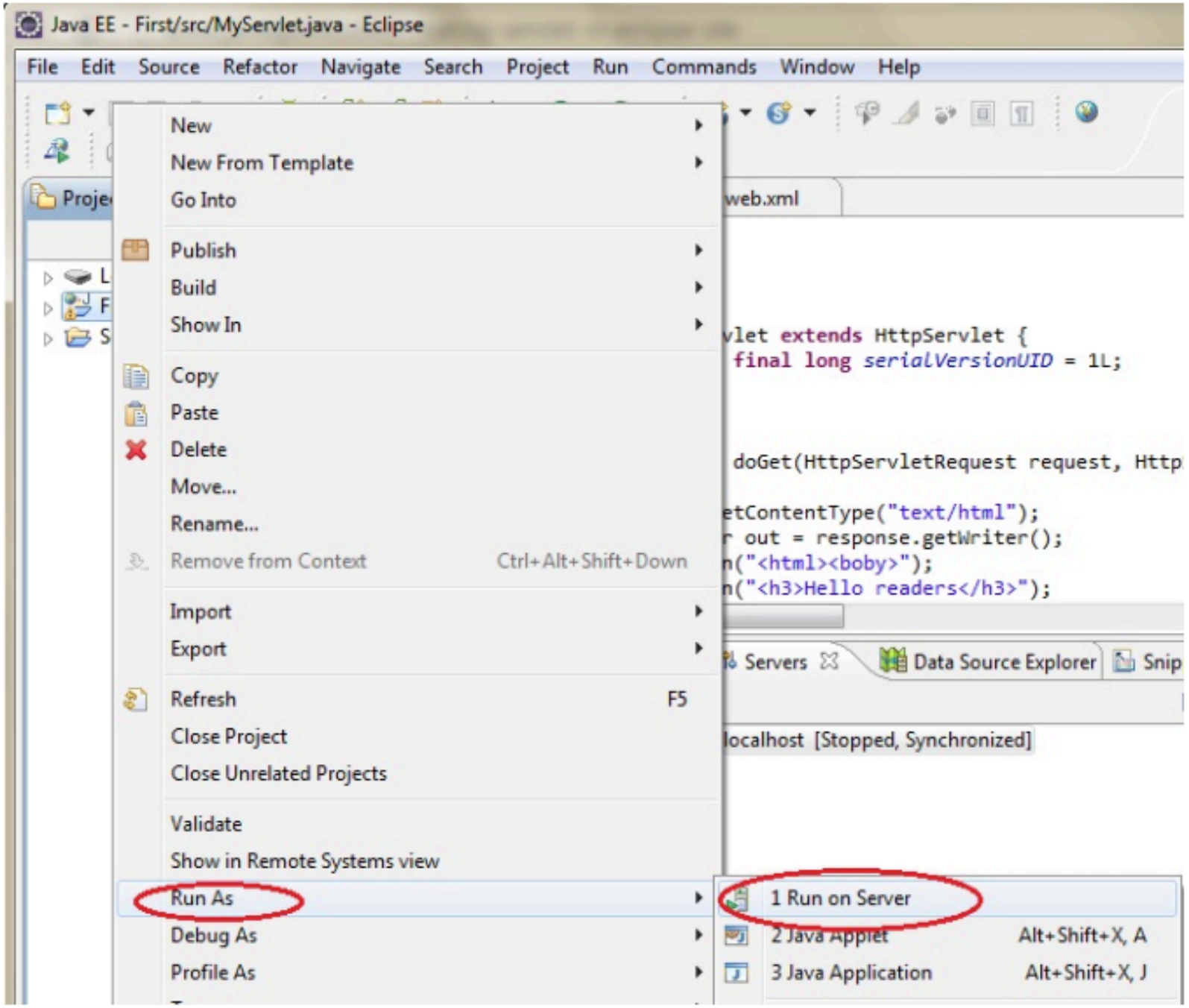
9. Click on **Add External JARs**



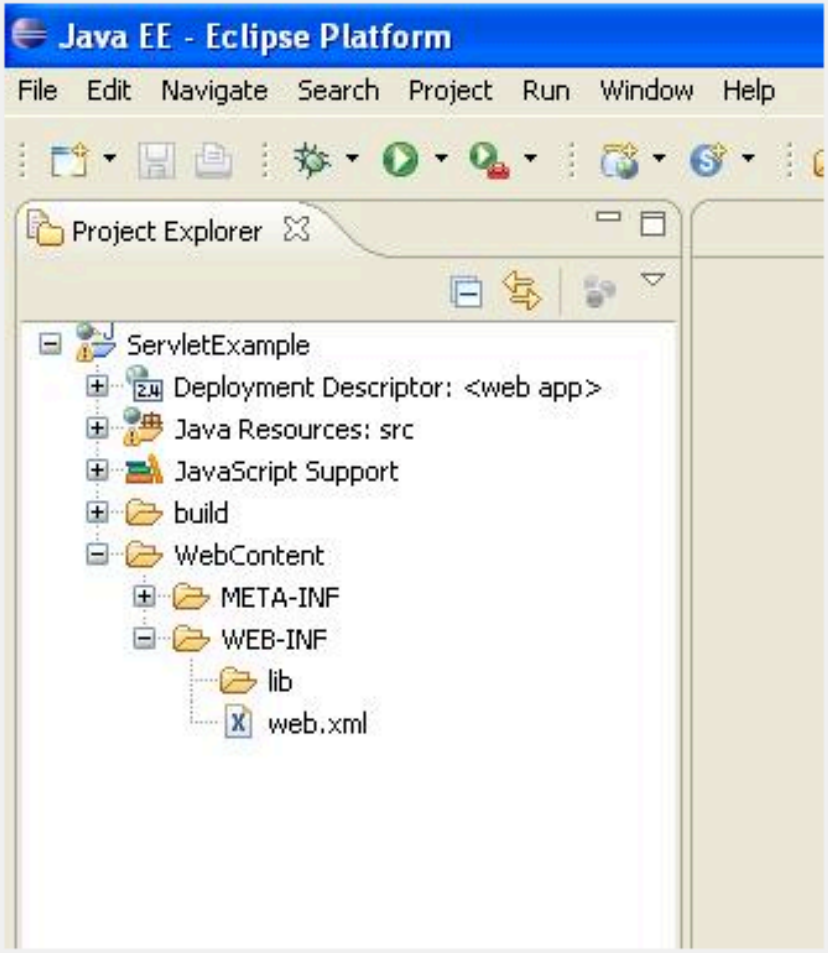

Properties for First
JAR Selection
« Apache Tomcat 7.0.14 ▸ lib
Search lib
Organize
New folder
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:)
abhi spl (D:)
Name
Date modified
Type
catalina.jar 30-10-2012 07:54 Executabl
catalina-ant.jar 30-10-2012 07:54 Executabl
catalina-ha.jar 30-10-2012 07:54 Executabl
catalina-tribes.jar 30-10-2012 07:54 Executabl
ecj-3.6.2.jar 30-10-2012 07:54 Executabl
el-api.jar 30-10-2012 07:54 Executabl
jasper.jar 30-10-2012 07:54 Executabl
jasper-el.jar 30-10-2012 07:54 Executabl
jsp-api.jar 30-10-2012 07:54 Executabl
servlet-api.jar 30-10-2012 07:54 Executabl
tomcat-api.jar 30-10-2012 07:54 Executabl
tomcat-coyote.jar 30-10-2012 07:54 Executabl
File name: servlet-api.jar
.jar;.zip
Open
Cancel


Properties for First
type filter text
Resource
Builders
Content Assist
Deployment Asse
File Transfer
Formatter
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
JavaScript
JSP Fragment
Preview Settings
Project Facets
Project Natures
Project Reference
Run/Debug Settin
Server
Service Policies
Targeted Runtime
Task Repository
Task Tags
Java Build Path
Source
Projects
Libraries
Order and Export
JARs and class folders on the build path:
servlet-api.jar - C:\Program Files\Apache Software Four
EAR Libraries
JRE System Library [JavaSE-1.6]
Web App Libraries
Access rules: No rules defined
Native library location: (None)
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
OK
Cancel

11. Start the server and run the application



1. **Create a new project**
    - In the menu bar, File / new / Dynamic web project.
    - Name your project - to be consistent with the example, name it "ServletExample". This By choosing "Dynamic web project", eclipse creates the default folders hierarchy. The folders hierarchy is shown in the "Project Explorer" view at the left side of the eclipse window.If it is not shown, you can show it from the menu bar, Window / Show View / Project Explorer. It should look like that:



JSP, CSS files, imagesâ€¦ etc will be placed in the "WebContent" folder. Java files (Servlets) will be placed in the "Java Resources: src" folder.

2. **Create the JSP file**
    - In the "Project Explorer" view, R-click "WebContent" / New / JSP.
    - Name your JSP - to be consistent with the example, name it "Home.jsp"
    - place the JSP the following code:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
        <title> My first JSP   </title>
    </head>
    <body>
        <form action="HelloServlet">
            Please enter a color <br>
            <input type="text" name="color"size="20px">
            <input type="submit" value="submit">
        </form>
    </body>
</html>
```

3. **Create the Servlet**
    - In the "Project Explorer" view, R-click "Java Resources: src" / New / Class
    - Name your class - to be consistent with the example, name it "HelloWorld"
    - place the following code in the class:

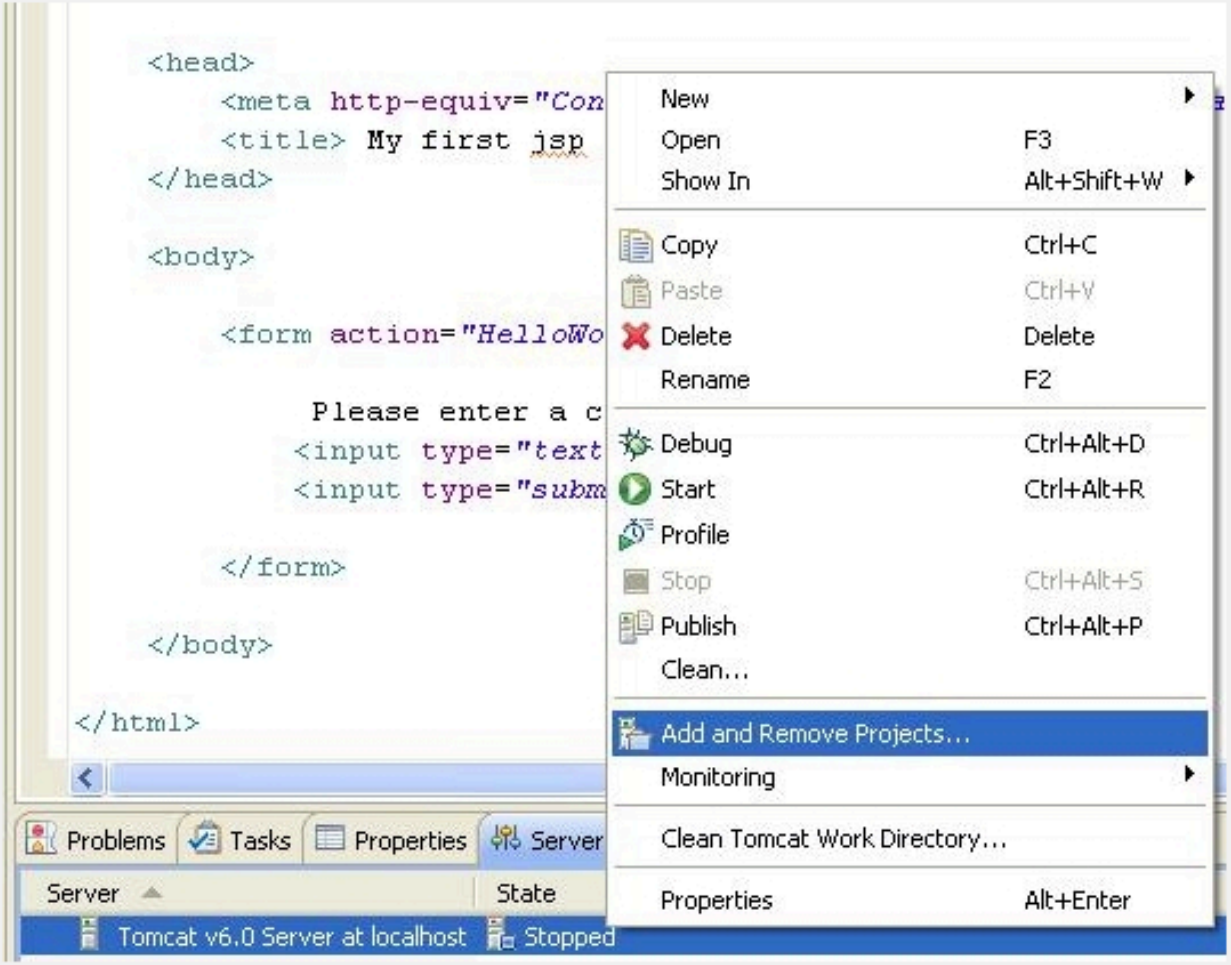
```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.PrintWriter;

public class HelloWorld extends HttpServlet {
 protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException
 {
  // reading the user input
  String color= request.getParameter("color");
  PrintWriter out = response.getWriter();
  out.println (
   "<!DOCTYPE html PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"
      \"http://www.w3.org/TR/html4/loose.dtd\">\n" +
   "<html> \n" +
    "<head> \n" +
     "<meta http-equiv=\"Content-Type\" content=\"text/html;
      charset=ISO-8859-1\"> \n" +
     "<title> My first jsp  </title> \n" +
    "</head> \n" +
    "<body> \n" +
     "<font size=\"12px\" color=\"" + color + "\">" +
      "Hello World" +
     "</font> \n" +
    "</body> \n" +
   "</html>"
  );
```

4. **Define your servlet in "web.xml"**
    - Open web.xml from the "Project Explorer" view, WebContent / Web-INF / R-click web.xml / Open With / Text Editor
    - Replace its content by the following code:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
 <servlet>
  <servlet-name>Hello</servlet-name>
  <servlet-class>HelloWorld</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Hello</servlet-name>
  <url-pattern>/HelloServlet</url-pattern>
 </servlet-mapping>
</web-app>
```

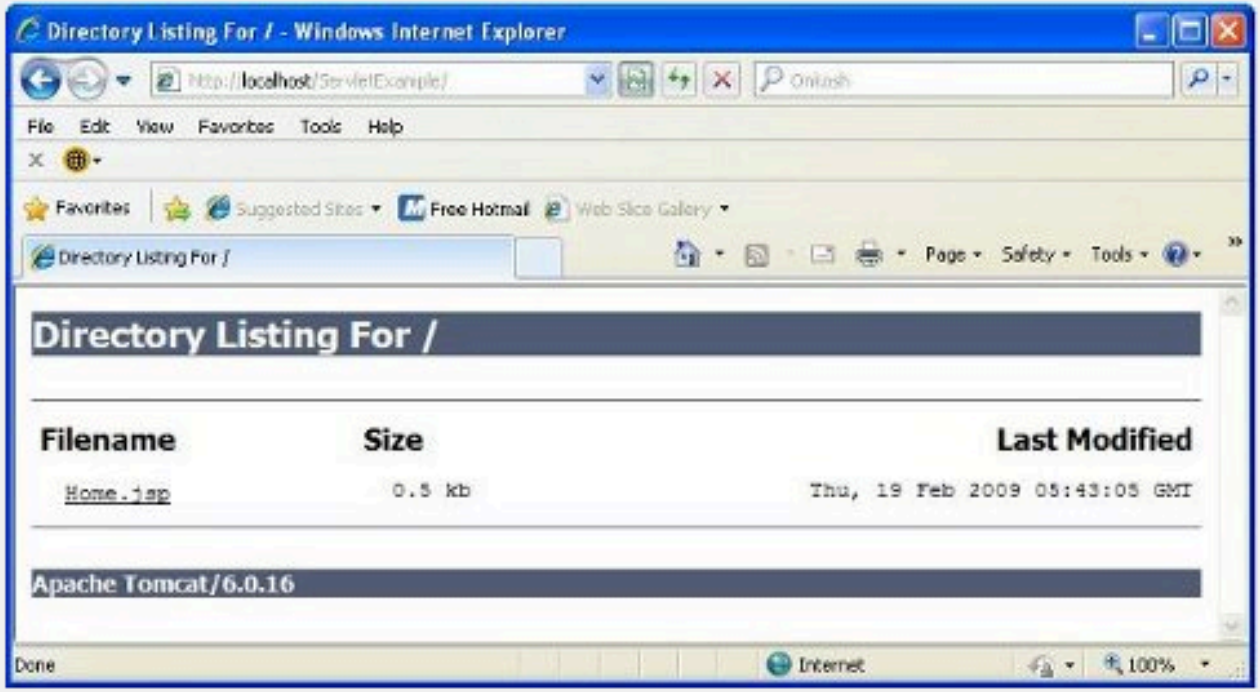
5. **Add your project to Tomcat**
    - R-click the Tomcat server record / Add and Remove Projects; add your project
    - Open the "Servers" view by clicking the "Servers" tab at the bottom of eclipse window. If the "Servers" tab is not shown, you can show it from the menu bar, Window / Show View/ Servers. Or Window / Show View / Otherâ€¦ / Server / Servers.
    - In the "Servers" view, R-click Tomcat-server record / Add and Remove Projects (If Tomcat-server record is not there, please check Steps 5-6 in the Configurations section by clicking next from here)
    - Add your project



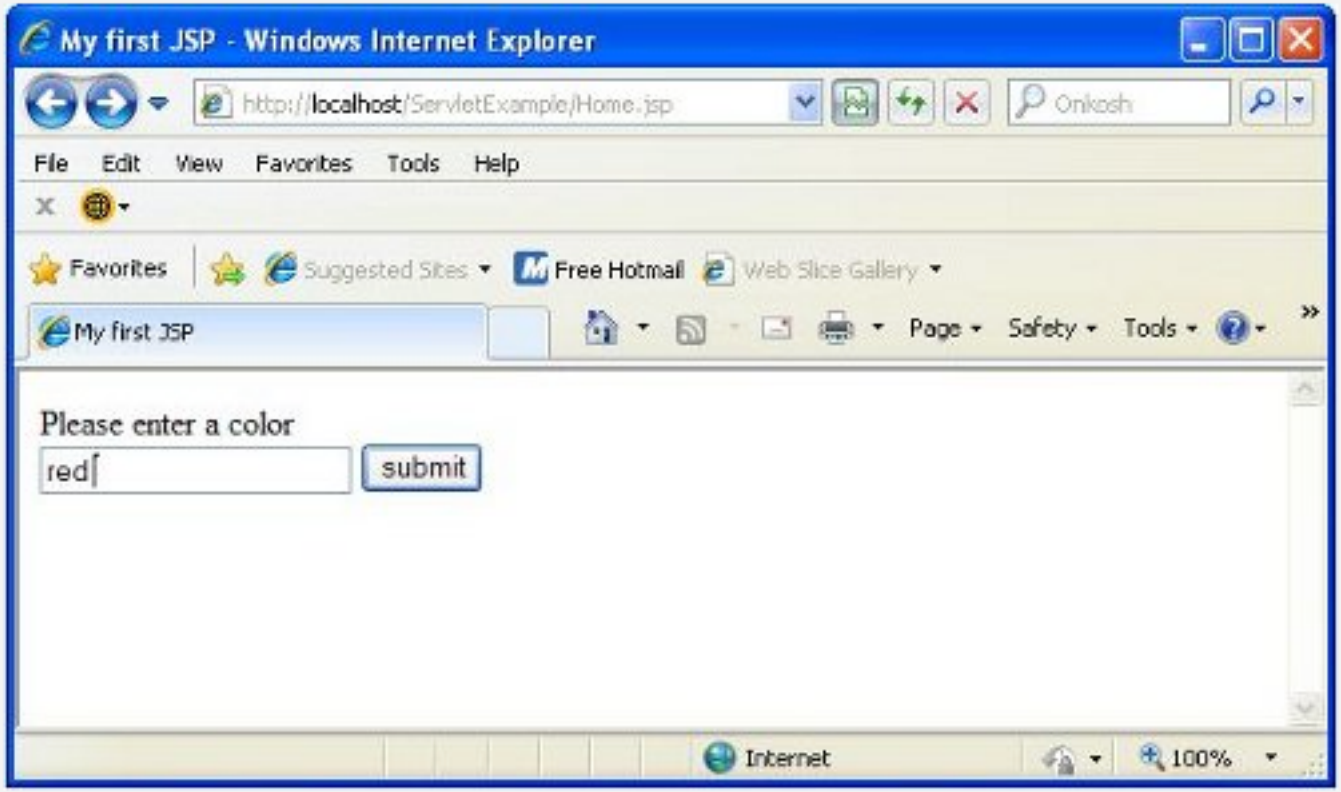
6. **Start "Tomcat":** In the "Servers" view, R-click Tomcat-server record / Start

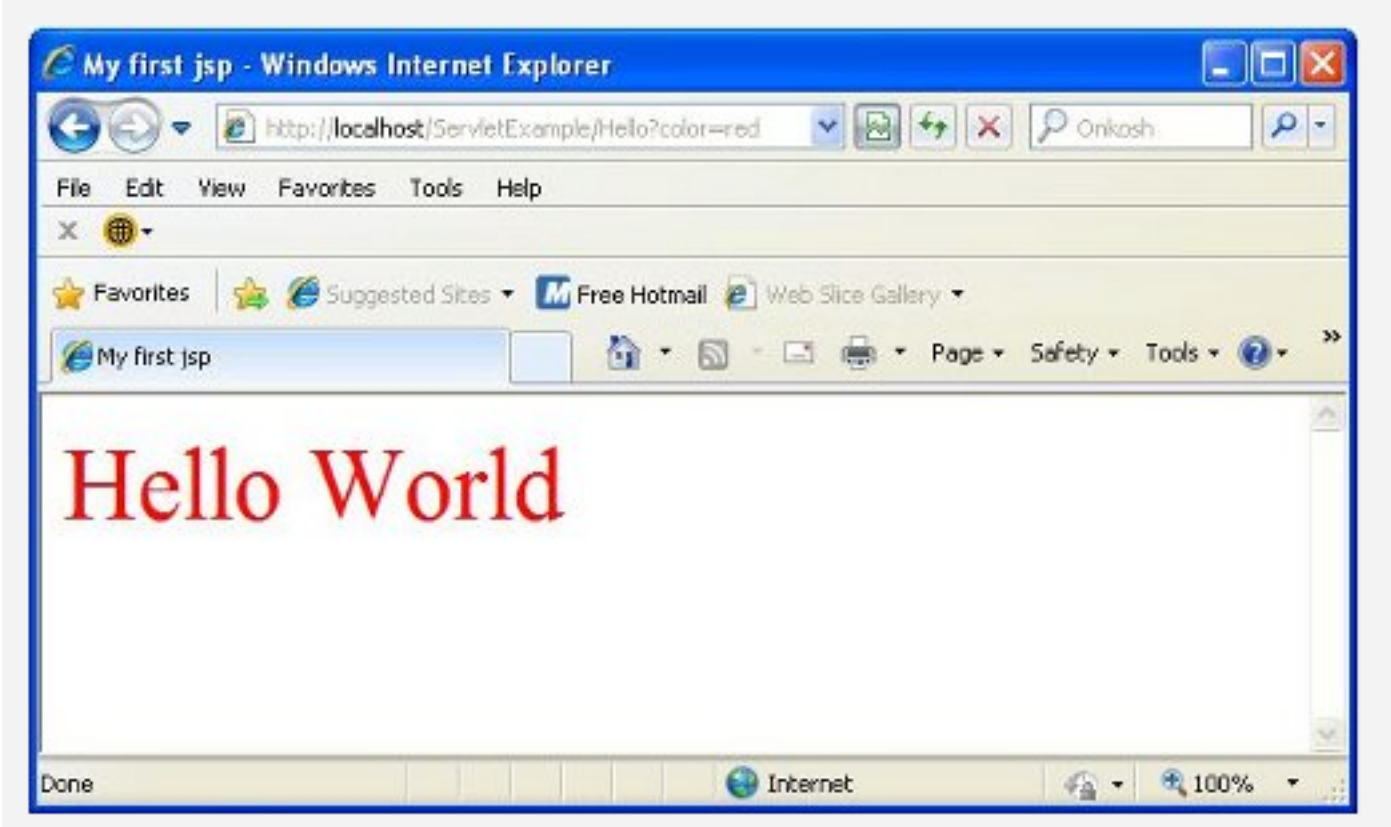
7. **Test your project:**
   - In your web browser (Internet Explorer or Firefox) :
     - If the port is set to 80: Type http://localhost/YourProjectName - In our example the URL is http://localhost/ServletExample
     - If the port is set to 8080: Type http://localhost:8080/YourProjectName - In our example the URL is http://localhost:8080/ServletExample

     Please note that Tomcat's default port is 8080, but the version refer to in the Downloads section (the pre-configured version), has its port set to 80. We will proceed in this tutorial assuming that the port is set to 80. (You can check your version's port from Tomcat-Installation-Directory / conf / server.xml)
   - Click your JSP file name shown in the directory listing - in our example it is Home.jsp



- This should be the result




My first jsp - Windows Internet Explorer
http://localhost/ServletExample/Hello?color=red
Onkosh
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Free Hotmail
Web Slice Gallery
My first jsp
Page
Safety
Tools
Hello World
Done
Internet
100%

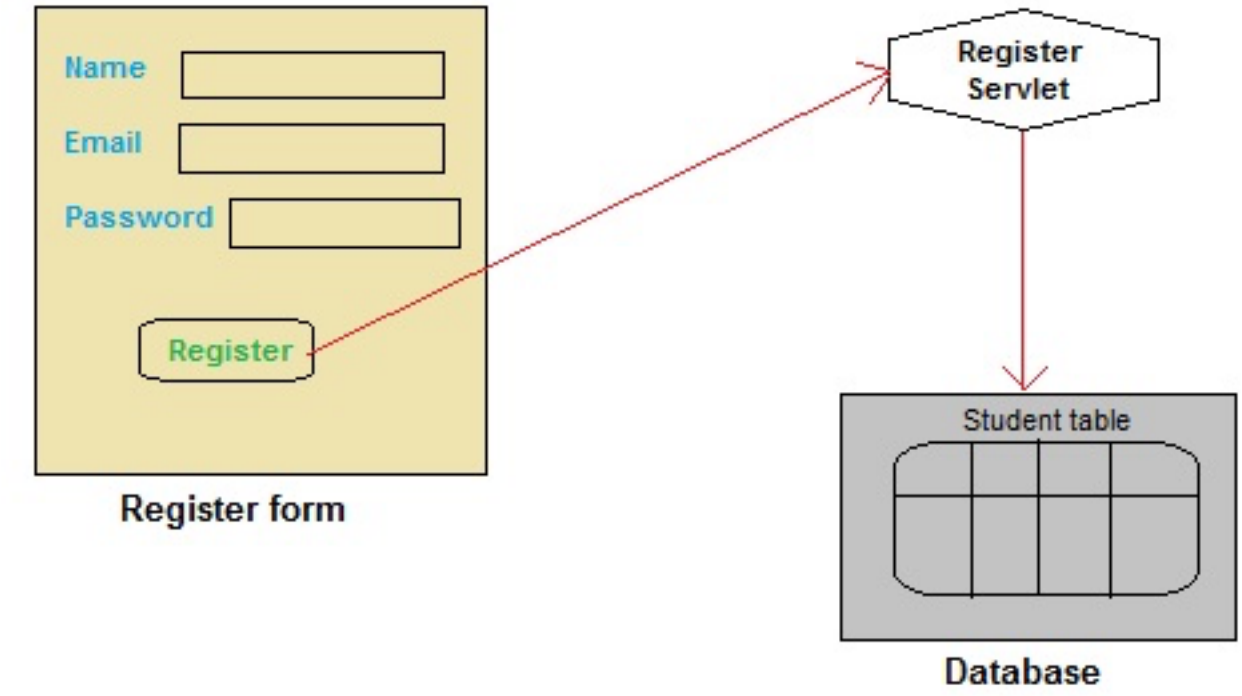
8. **Set your project's welcome file** (let the server open "Home.jsp" once you open the project)
    - Open "web.xml" (WebContent / Web-INF / R-click web.xml / Open With / Text Editor)
    - Set the "welcome-file-list" to be your home page - In our example it is Home.jsp. The code in "web.xml" should look like that:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
   http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
 <servlet>
  <servlet-name>Hello</servlet-name>
  <servlet-class>HelloWorld</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Hello</servlet-name>
  <url-pattern>/HelloServlet</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
  <welcome-file>Home.jsp</welcome-file>
 </welcome-file-list>
</web-app>
```

9. **Restart the server** and test your project again using http://localhost/yourProjectName - in our example, the URL will be http://localhost/ServletExample/.

   Note that the server does not recognize changes in "web.xml" except if you restarted, or started, the server.

Registration form in Servlet



Create a table into your database

```
create table Student
(
name varchar(60),
email varchar(60),
pass varchar(100)
)
```

index.html

```
<html>
    <head>
        <title>Register form</title>
    </head>
    <body>
        <form method="post" action="register">
        Name:<input type="text" name="name" /><br/>
        Email ID:<input type="text" name="email" /><br/>
        Password:<input type="text" name="pass" /><br/>
        <input type="submit" value="register" />
        </form>
    </body>
</html>
```

Register.java

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

public class Register extends HttpServlet {


    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String name = request.getParameter("name");
        String email = request.getParameter("email");
        String pass = request.getParameter("pass");
        try{
```

```
//loading driver
Class.forName("com.mysql.jdbc.Driver");

      //creating connection with the database
  Connection  con=DriverManager.getConnection
            ("jdbc:mysql://localhost:3306/test","username","password");

PreparedStatement ps=con.prepareStatement
          ("insert into Student values(?,?,?)");

ps.setString(1, name);
ps.setString(2, email);
ps.setString(3, pass);
int i=ps.executeUpdate();

if(i>0)
{
    out.println("You are sucessfully register");
}

}catch(Exception se)
{
    se.printStackTrace(); } } }
```

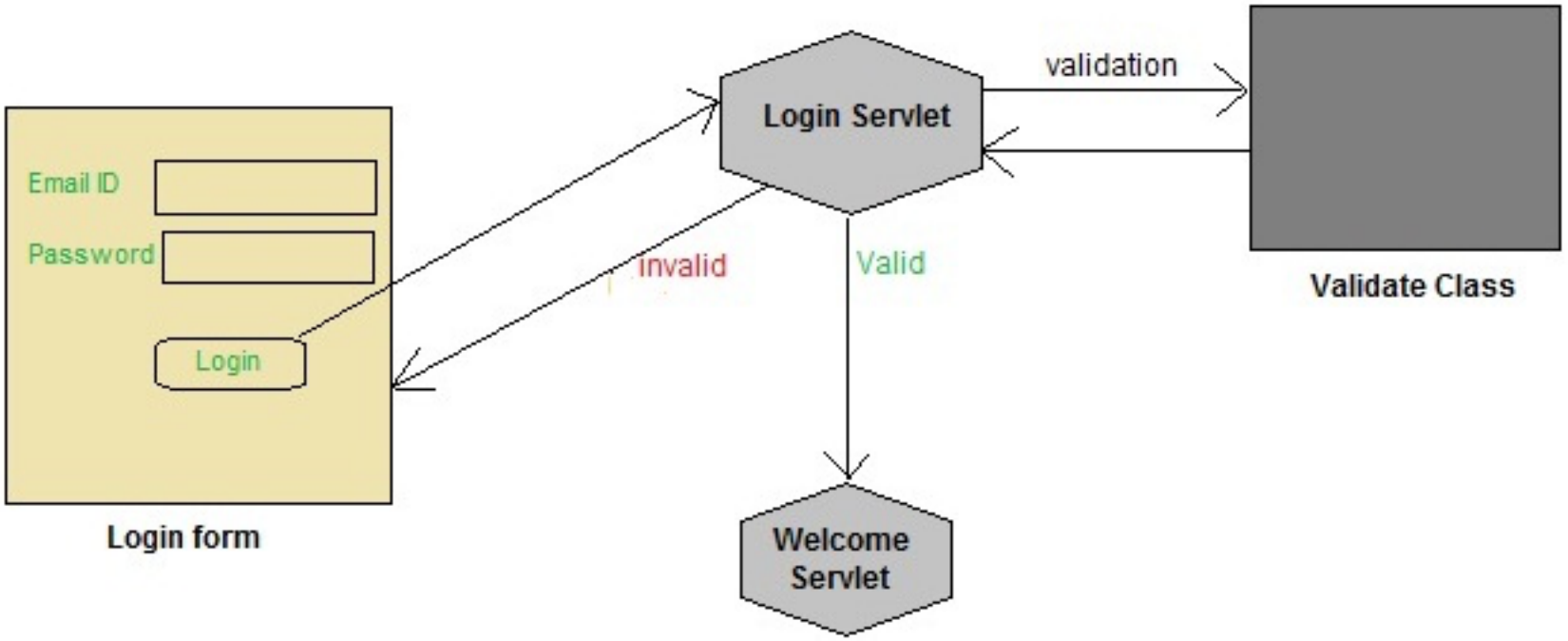
web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app  version="3.0"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
java.sun.com/xml/ns/javaee/web-app_3_0.xsd" >

    <servlet>
        <servlet-name>register</servlet-name>
        <servlet-class>Register</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>register</servlet-name>
        <url-pattern>/register</url-pattern>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```

Login System in Servlet



index.html

```
<html>
    <head>
        <title>login form</title>
    </head>
    <body>
        <form method="post" action="login">
        Email ID:<input type="text" name="email" /><br/>
        Password:<input type="text" name="pass" /><br/>
        <input type="submit" value="login" />
        </form>
    </body>
</html>
```

Login.java

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


public class Login extends HttpServlet {


    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String email = request.getParameter("email");
        String pass = request.getParameter("pass");
```

```
 if(Validate.checkUser(email, pass))
        {
            RequestDispatcher rs =
request.getRequestDispatcher("Welcome");
            rs.forward(request, response);
        }
        else
        {
           out.println("Username or Password incorrect");
           RequestDispatcher rs =
request.getRequestDispatcher("index.html");
           rs.include(request, response);
        }

    }
}
```

Validate.java

```
import java.sql.*;

public class Validate
 {

    public static boolean checkUser(String email,String pass)
    {
     boolean st =false;
     try{

            //loading driver
       Class.forName("com.mysql.jdbc.Driver");

            //creating connection with the database
       Connection con=DriverManager.getConnection
                  ("jdbc:mysql://localhost:3306/test","root","abhijit");
       PreparedStatement ps =con.prepareStatement
                        ("select * from register where email=? and pass=?");
```

```
 ps.setString(1, email);
        ps.setString(2, pass);
        ResultSet rs =ps.executeQuery();
        st = rs.next();

     }catch(Exception e)
     {
         e.printStackTrace();
     }
        return st;
  }
}
```

Welcome.java

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


public class Welcome extends HttpServlet {


    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("Welcome user");
      }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/
xml/ns/javaee/web-app_3_0.xsd" >
   <servlet>
      <servlet-name>login</servlet-name>
      <servlet-class>Login</servlet-class>
   </servlet>
   <servlet>
      <servlet-name>Welcome</servlet-name>
      <servlet-class>Welcome</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>login</servlet-name>
      <url-pattern>/login</url-pattern>
   </servlet-mapping>
   <servlet-mapping>
      <servlet-name>Welcome</servlet-name>
      <url-pattern>/Welcome</url-pattern>
   </servlet-mapping>

</web-app>
```

Email Sending using Servlet

index.html

```
<form action="mail" method="post">
 To:<input type="text" name="to"  /><br/>
 Subject:<input type="text" name="subject"  /><br/>
 Message:<input type="text" name="message"  /><br/>
 Your Email id:<input type="text" name="user" ><br/>
 Password<;input type="password" name="pass"   /><br/>
<input type="submit" value="send" />
 </form>
```

MailApp.java

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



public class MailApp extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String to = request.getParameter("to");
        String subject = request.getParameter("subject");
        String message =  request.getParameter("message");
        String user = request.getParameter("user");
        String pass = request.getParameter("pass");
        SendMail.send(to,subject, message, user, pass);
        out.println("Mail send successfully");
    }
}
```

SendMail.java

```
import java.io.*;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;


public class SendMail

  {

   public static void send(String to, String sub,
                    String msg, final String user,final String pass)
   {

    //create an instance of Properties Class
    Properties props = new Properties();

    /* Specifies the IP address of your default mail server
    for e.g if you are using gmail server as an email sever
    you will pass smtp.gmail.com as value of mail.smtp host. As shown here in the
   coding. Change accordingly, if your email id is not an gmail id*/

    props.put("mail.smtp.host", "smtp.gmail.com");
```

```
    props.put("mail.smtp.port", "587");            //this is optional
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.starttls.enable", "true");



    /*Pass Properties object(props) and Authenticator object
          for authentication to Session instance */

   Session session = Session.getInstance(props,
                     new javax.mail.Authenticator() {
  protected PasswordAuthentication getPasswordAuthentication() {
   return new PasswordAuthentication(user,pass);
   }
});
```

```
try
 {

 /* Create an instance of MimeMessage,
 it accept MIME types and headers */

 MimeMessage message = new MimeMessage(session);
 message.setFrom(new InternetAddress(user));
 message.addRecipient(Message.RecipientType.TO,new
InternetAddress(to));
 message.setSubject(sub);
 message.setText(msg);

 /* Transport class is used to deliver the message to the recipients */
 Transport.send(message);


 }catch(Exception e)
 {
     e.printStackTrace();
 }
    }

}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://
java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

    <servlet>
        <servlet-name>mail</servlet-name>
        <servlet-class>MailApp</servlet-class>
    </servlet>

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

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>
</web-app>
```