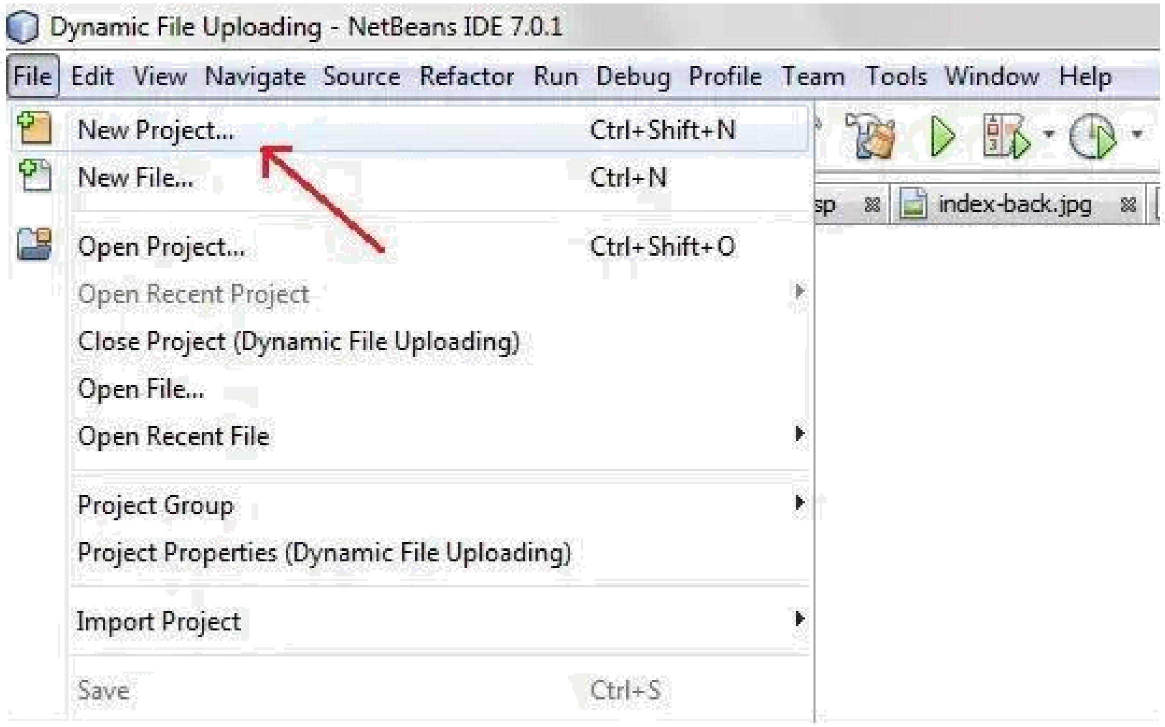
Following are the steps to create a servlet using Glassfish / Apache Tomcat Server:

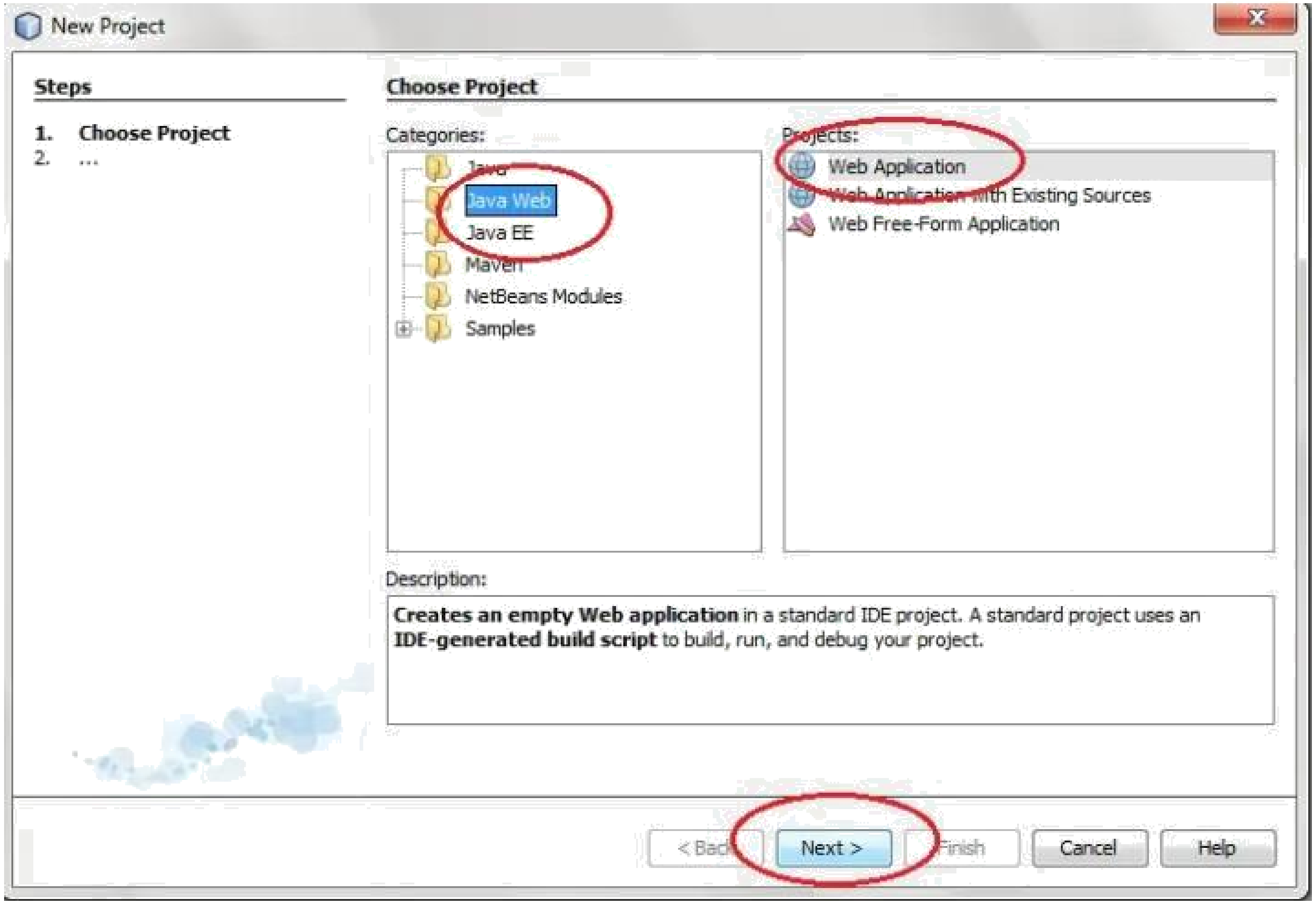
1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment
5. Start the server and deploy the project
6. Access the servlet

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

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

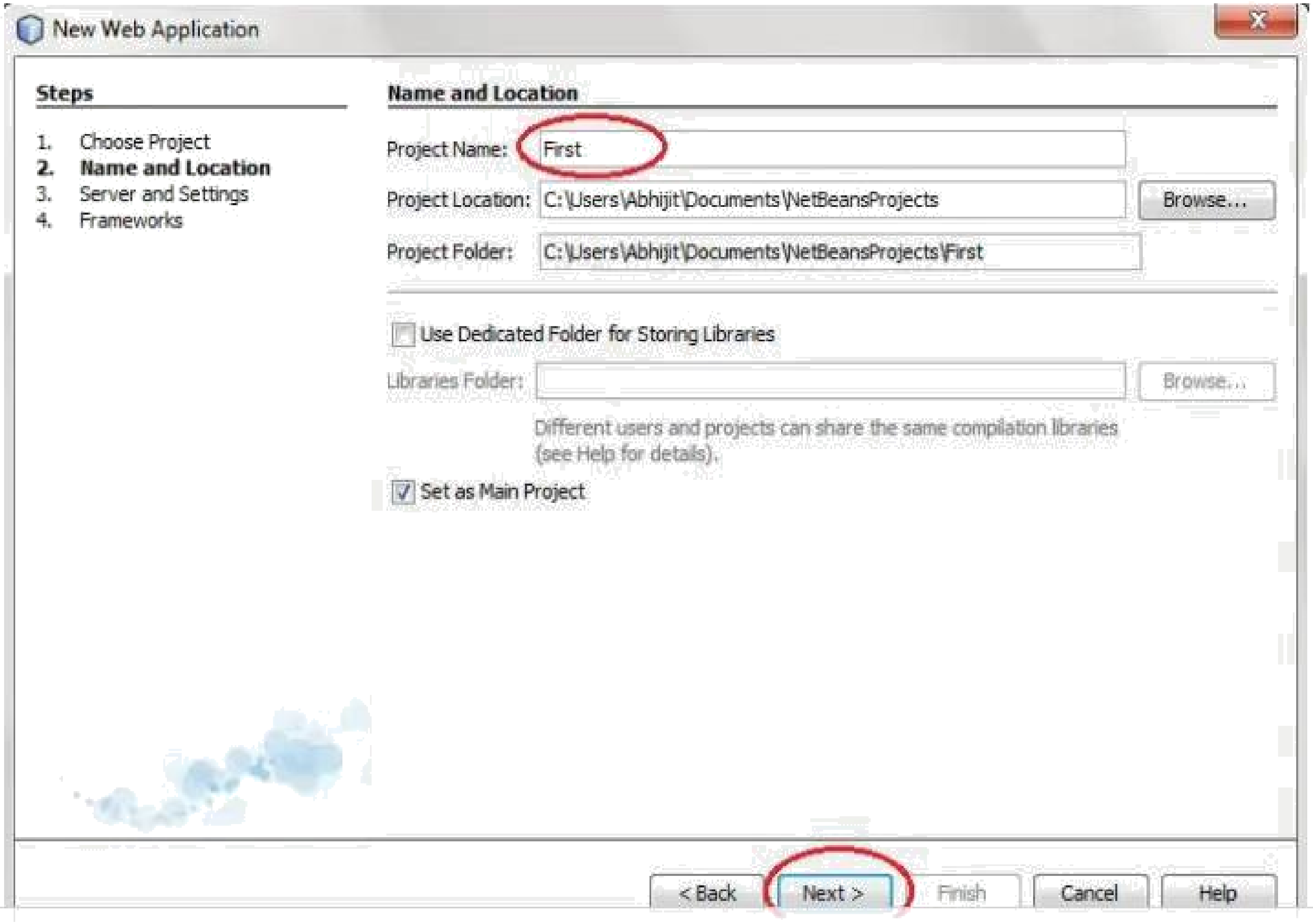

Dynamic File Uploading - NetBeans IDE 7.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
New Project... Ctrl+Shift+N
New File... Ctrl+N
Open Project... Ctrl+Shift+O
Open Recent Project
Close Project (Dynamic File Uploading)
Open File...
Open Recent File
Project Group
Project Properties (Dynamic File Uploading)
Import Project
Save Ctrl+S
index-back.jpg

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



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

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



Project Name: First

Project Location: C:\Users\Abhijit\Documents\NetBeansProjects

Project Folder: C:\Users\Abhijit\Documents\NetBeansProjects\First

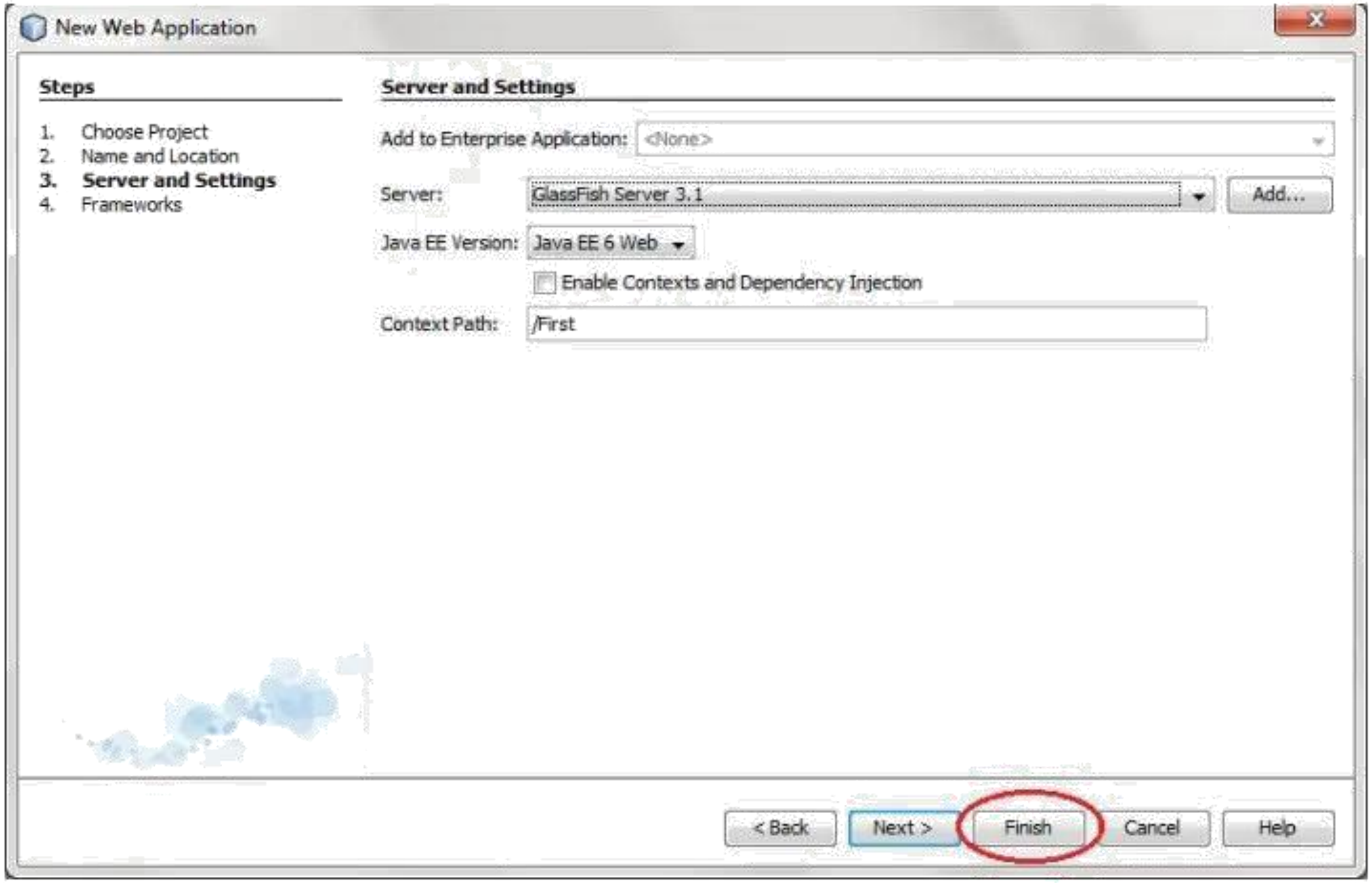
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☑ Set as Main Project

4. and then, Click **Finish**



New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

Add to Enterprise Application: <None>

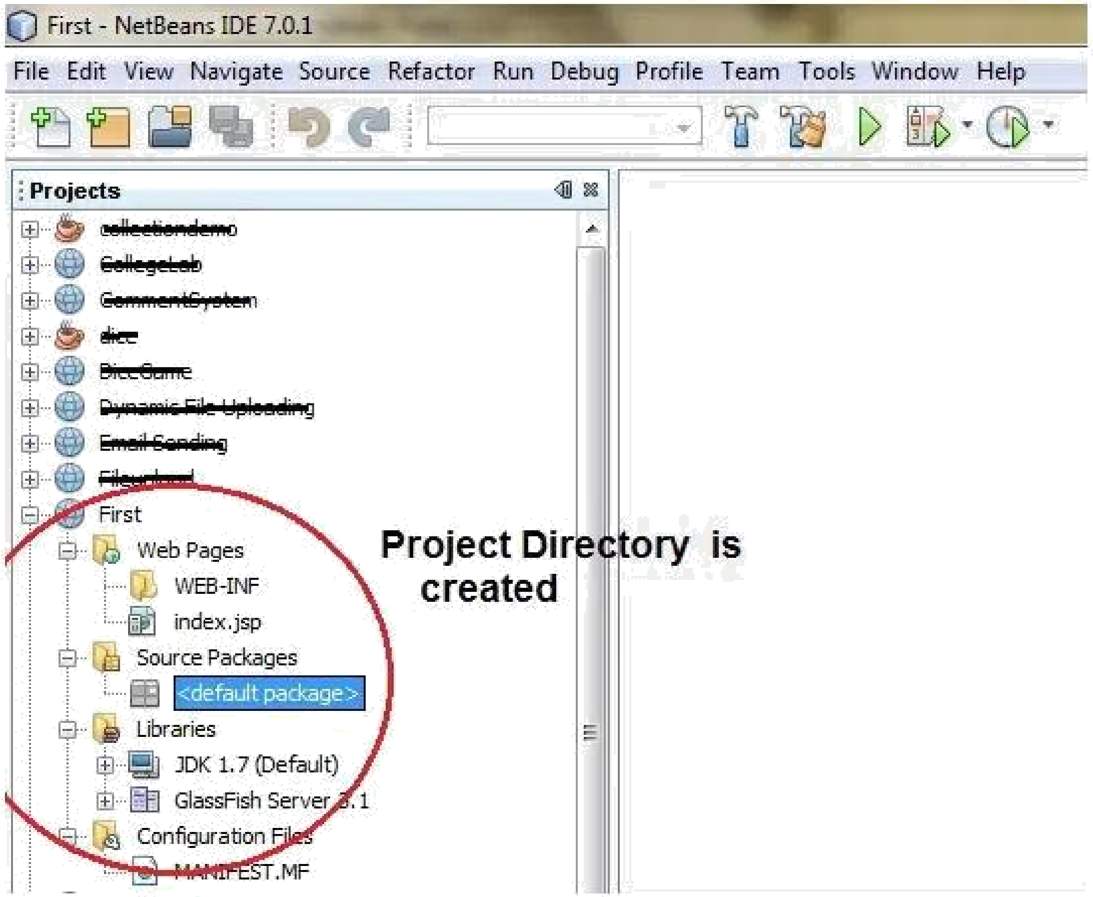
Server: GlassFish Server 3.1

Java EE Version: Java EE 6 Web

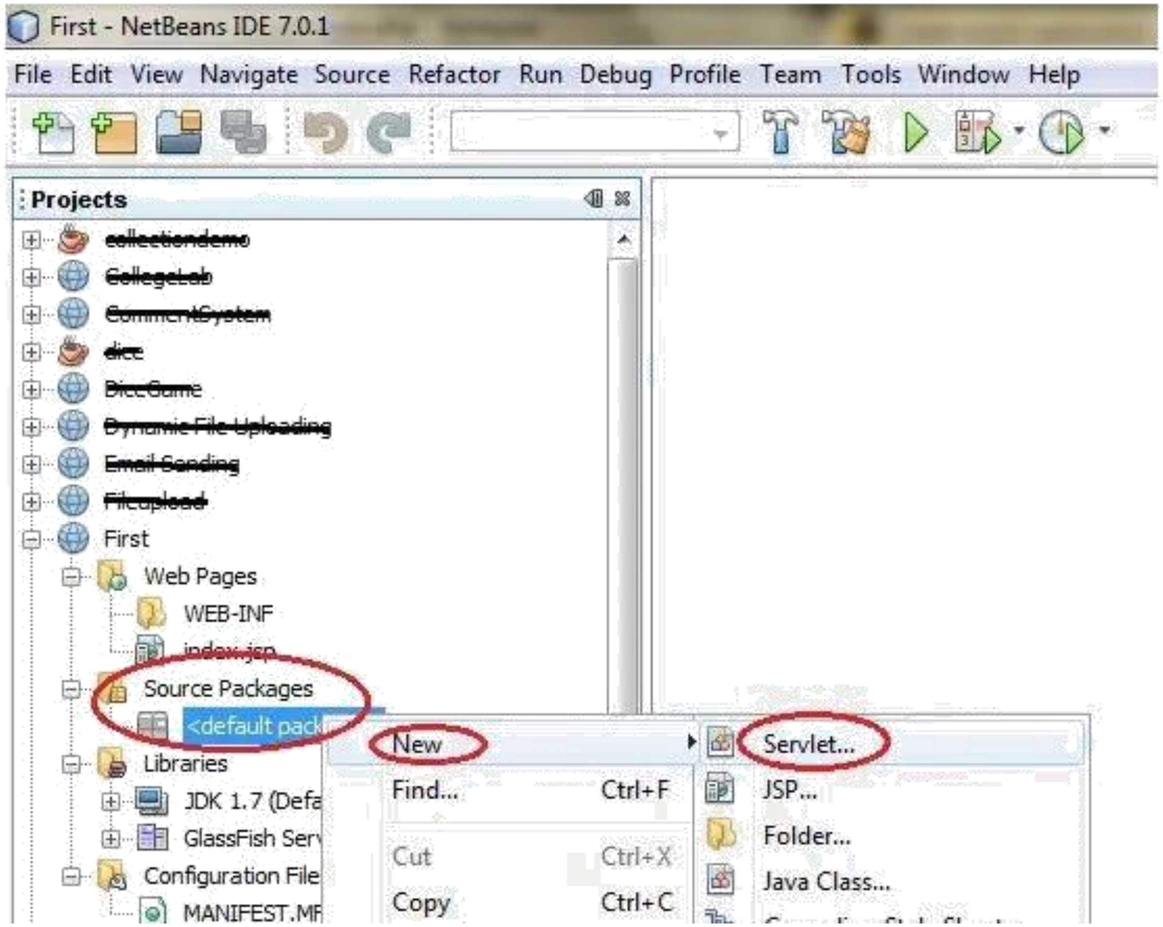
☐ Enable Contexts and Dependency Injection

Context Path: /First

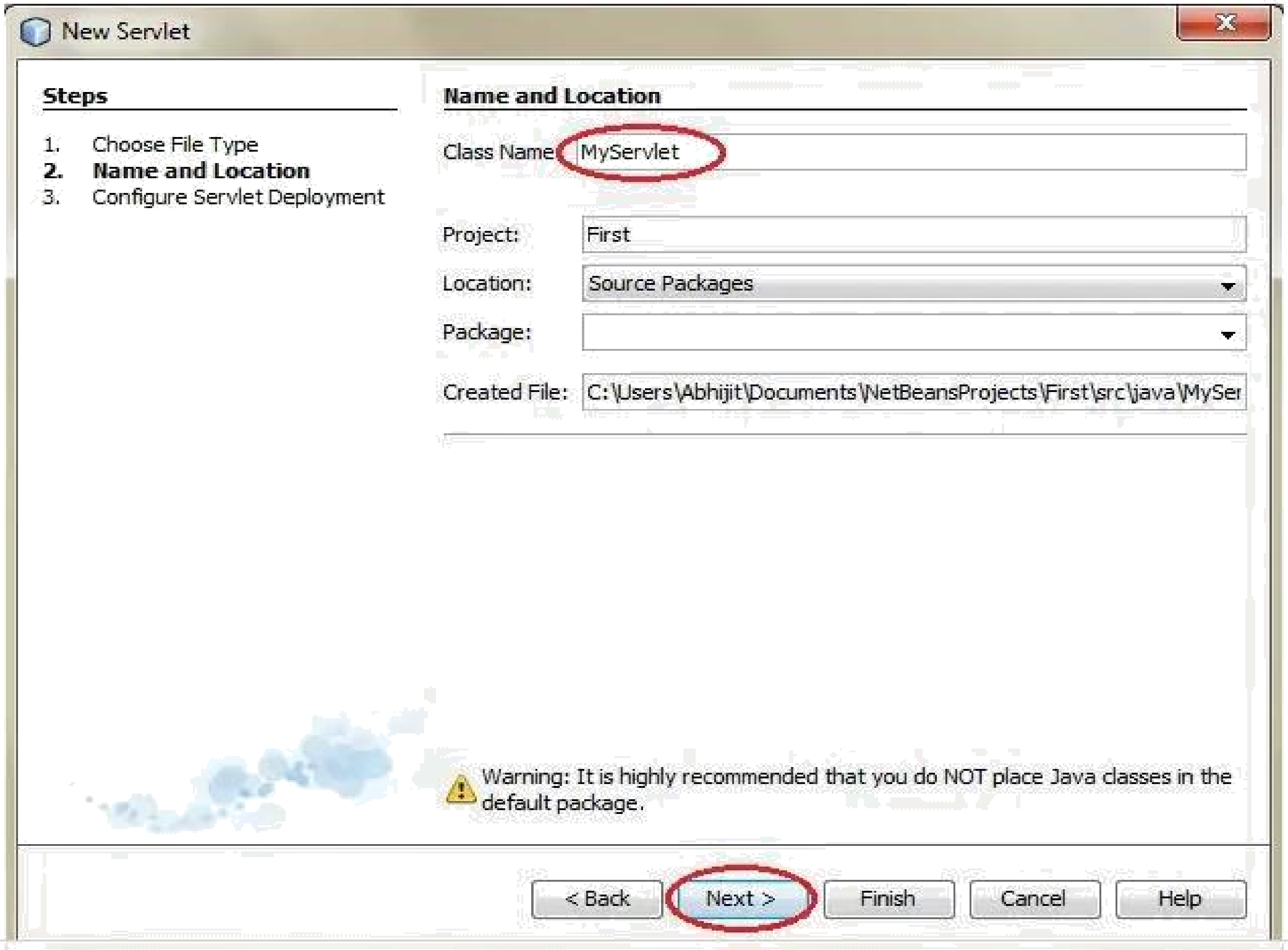
5. The complete directory structure required for the Servlet Application will be created automatically by the IDE.



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



7. Give a Name to your Servlet class file,



Class Name: MyServlet

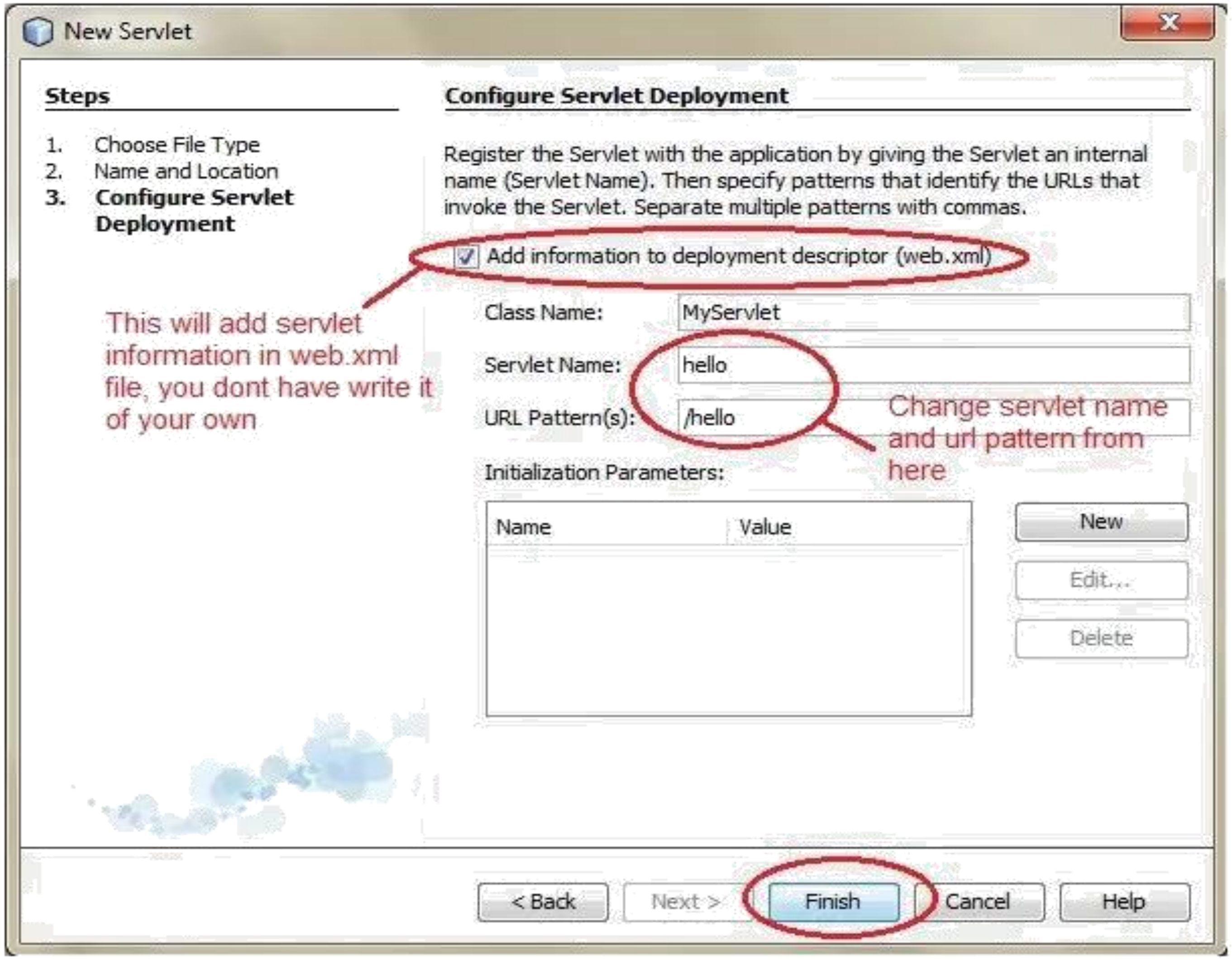
Project: First

Location: Source Packages

Package:

Created File: C:\Users\Abhijit\Documents\NetBeansProjects\First\src\java\MySer

Warning: It is highly recommended that you do NOT place Java classes in the default package.



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

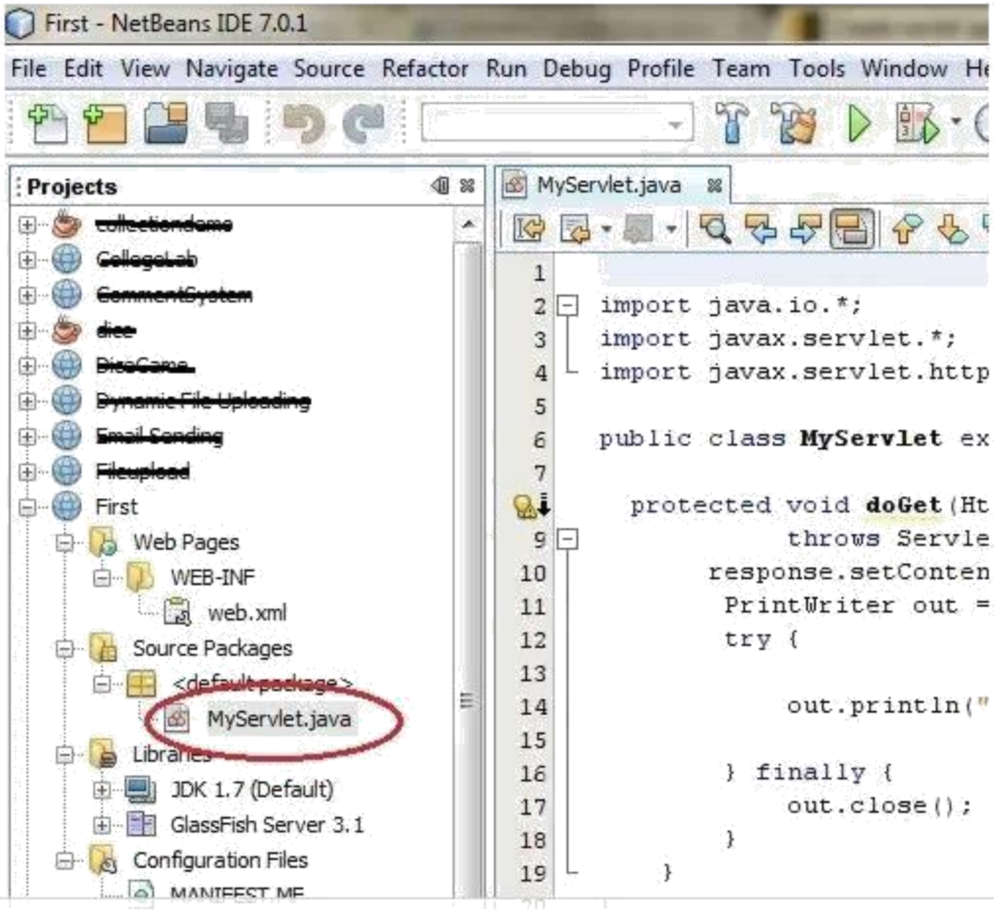
Class Name:	MyServlet
Servlet Name:	hello
URL Pattern(s):	/hello

Change servlet name and url pattern from here

Initialization Parameters:

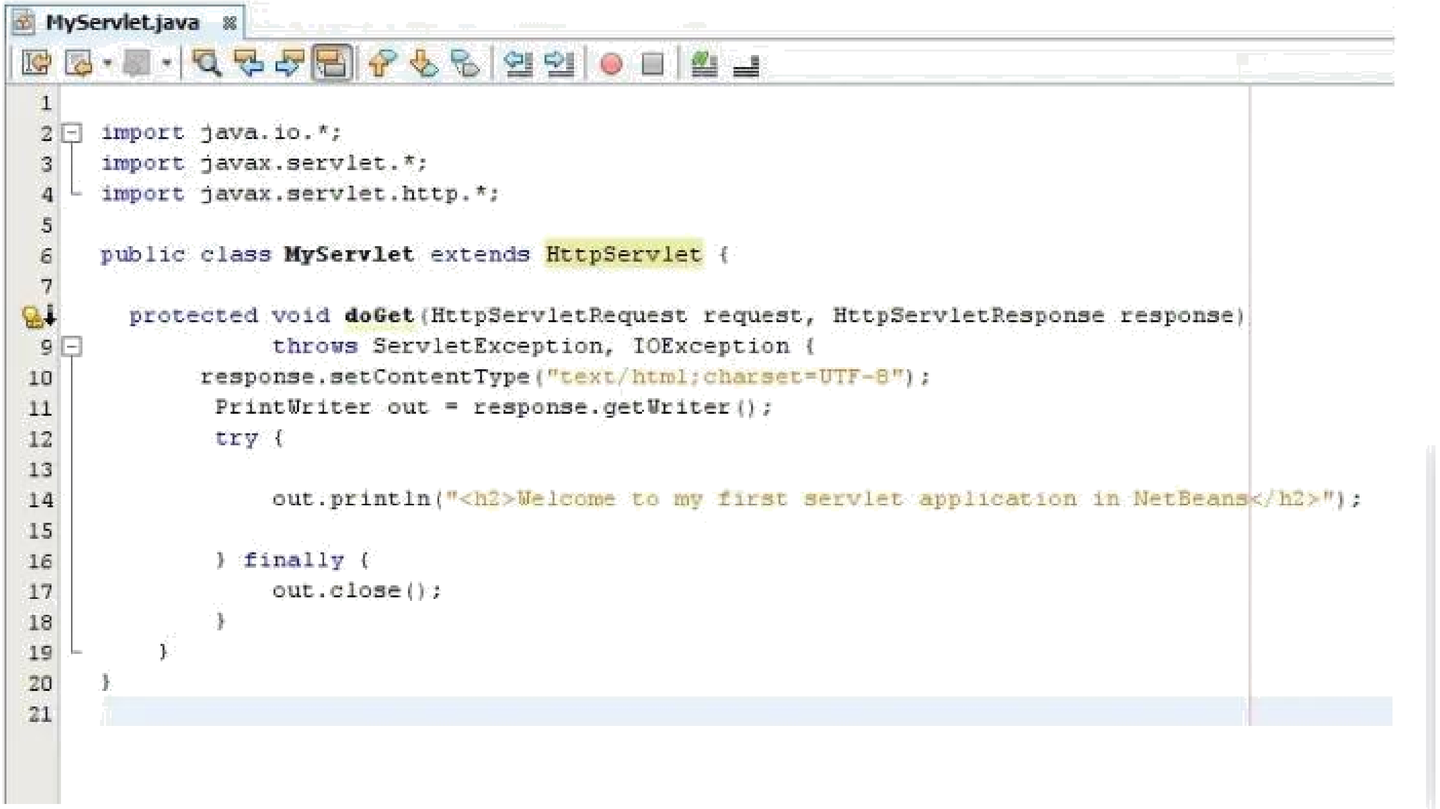
Name	Value

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



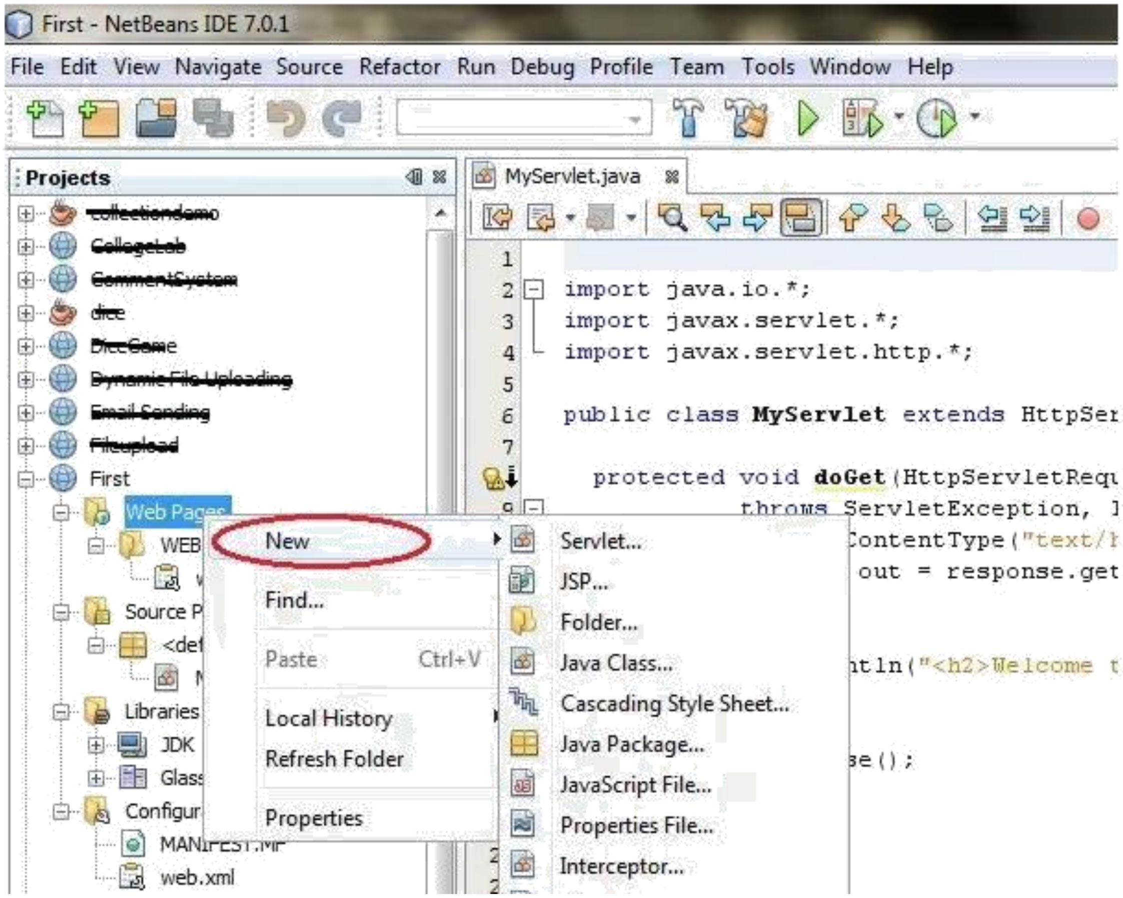
```

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

public class MyServlet ex

  protected void doGet(Ht
            throws Servle
        response.setConten
         PrintWriter out =
         try {

             out.println("

         } finally {
             out.close();
         }
    }
```

9. Write some code inside your Servlet class.



```

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

public class MyServlet extends HttpServlet {

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

            out.println("<h2>Welcome to my first servlet application in NetBeans</h2>");

         } finally {
            out.close();
         }
     }
}

```

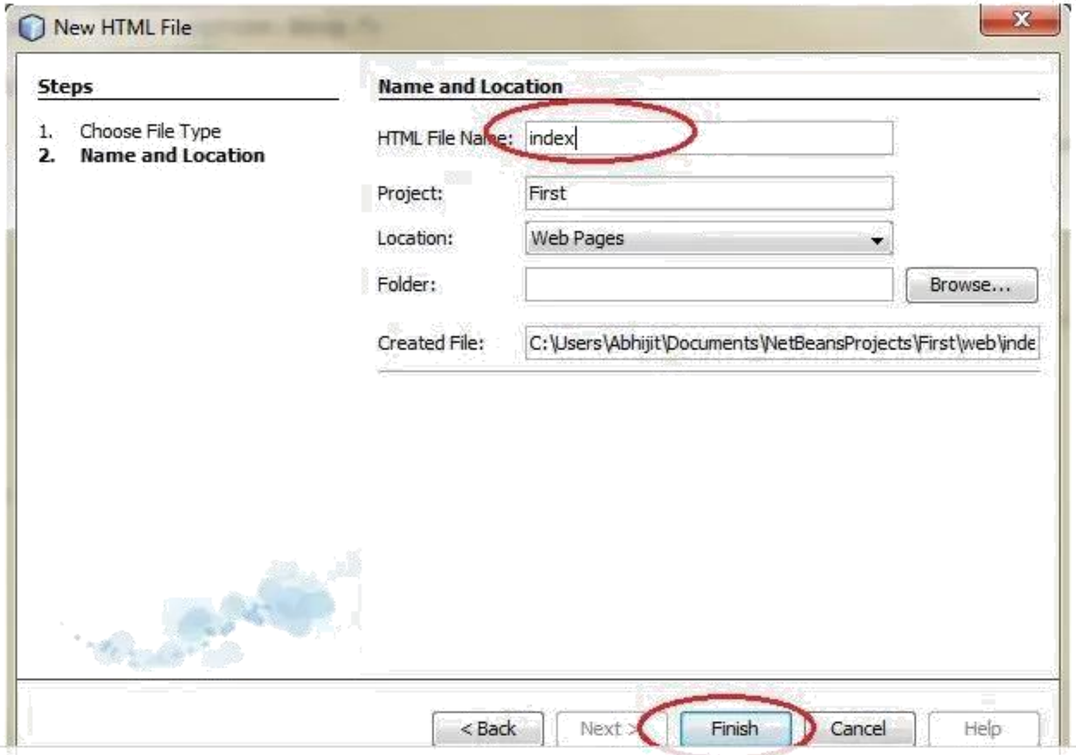
10. Create an HTML file, right click on **Web Pages** -> **New** -> **HTML**


First - NetBeans IDE 7.0.1

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

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

public class MyServlet extends HttpSer

  protected void doGet(HttpServletRequ
                throws ServletException, I
```

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



HTML File Name: index

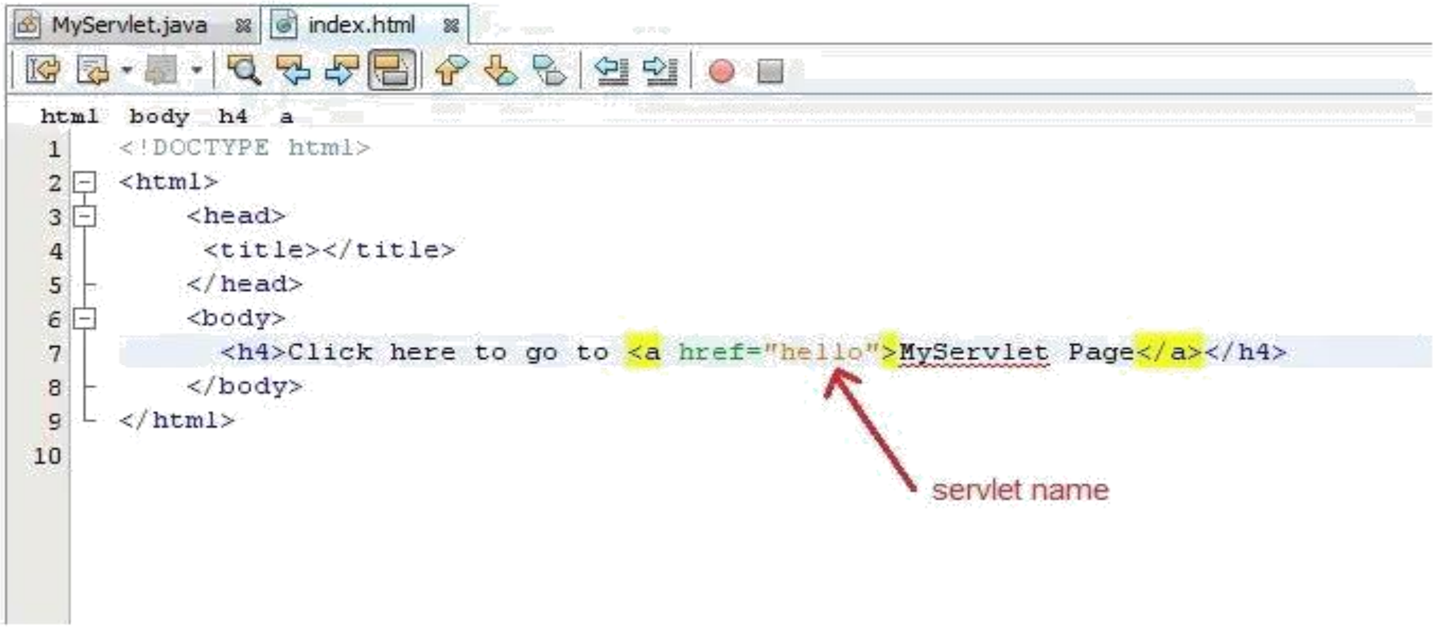
Project: First

Location: Web Pages

Folder:

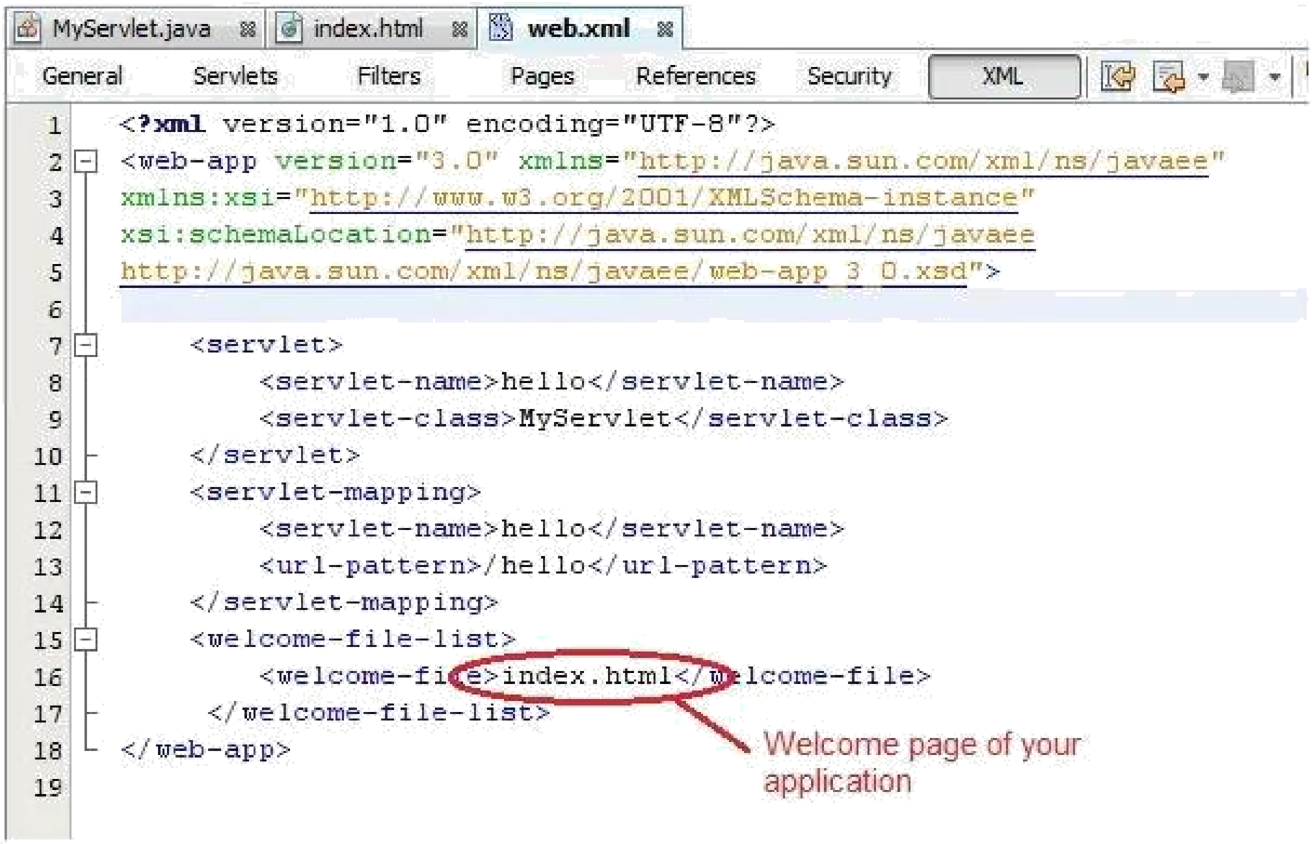
Created File: C:\Users\Abhijit\Documents\NetBeansProjects\First\web\inde

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



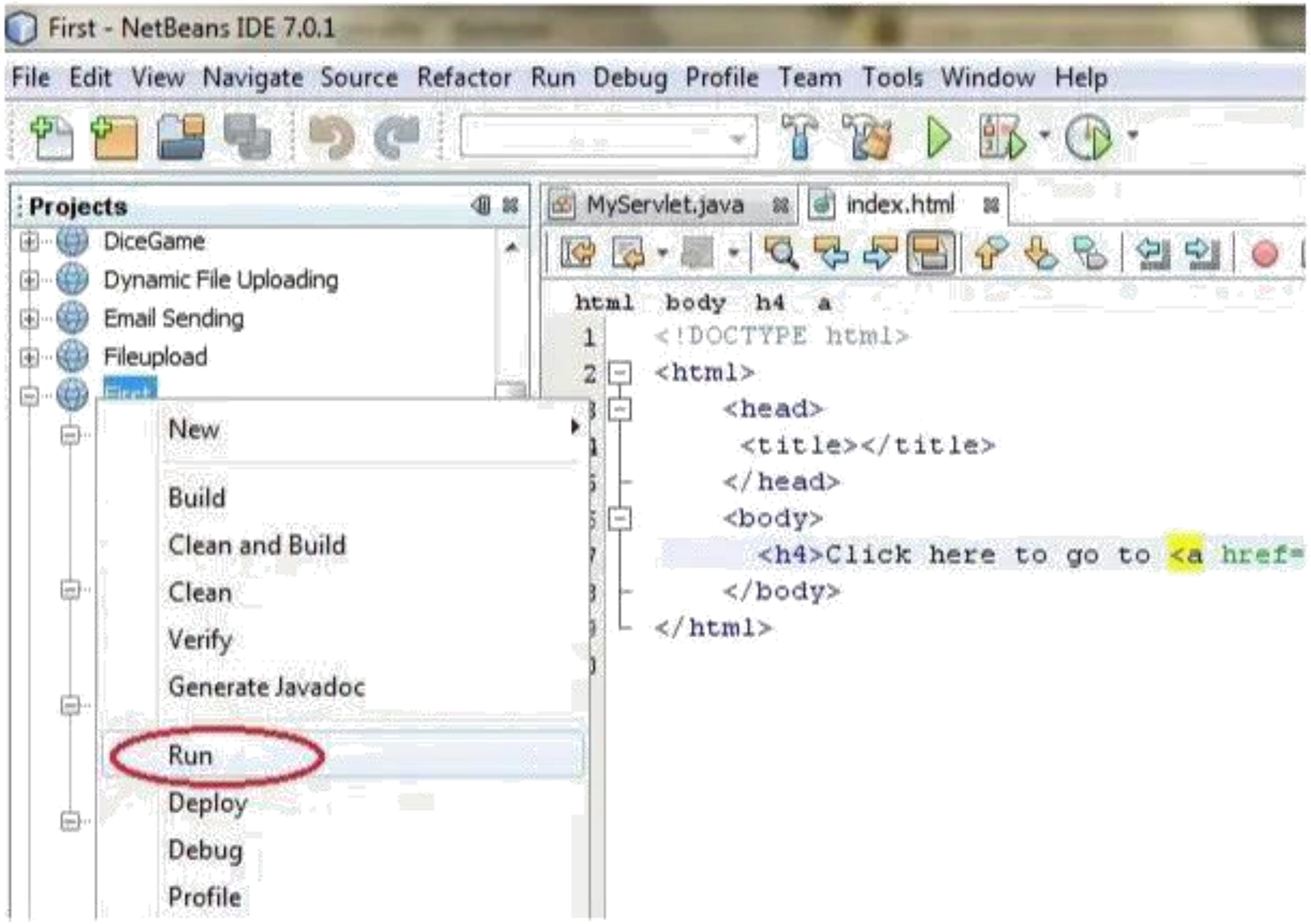
```
<!DOCTYPE html>
<html>
    <head>
     <title></title>
    </head>
    <body>
      <h4>Click here to go to <a href="hello">MyServlet Page</a></h4>
    </body>
</html>
```

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



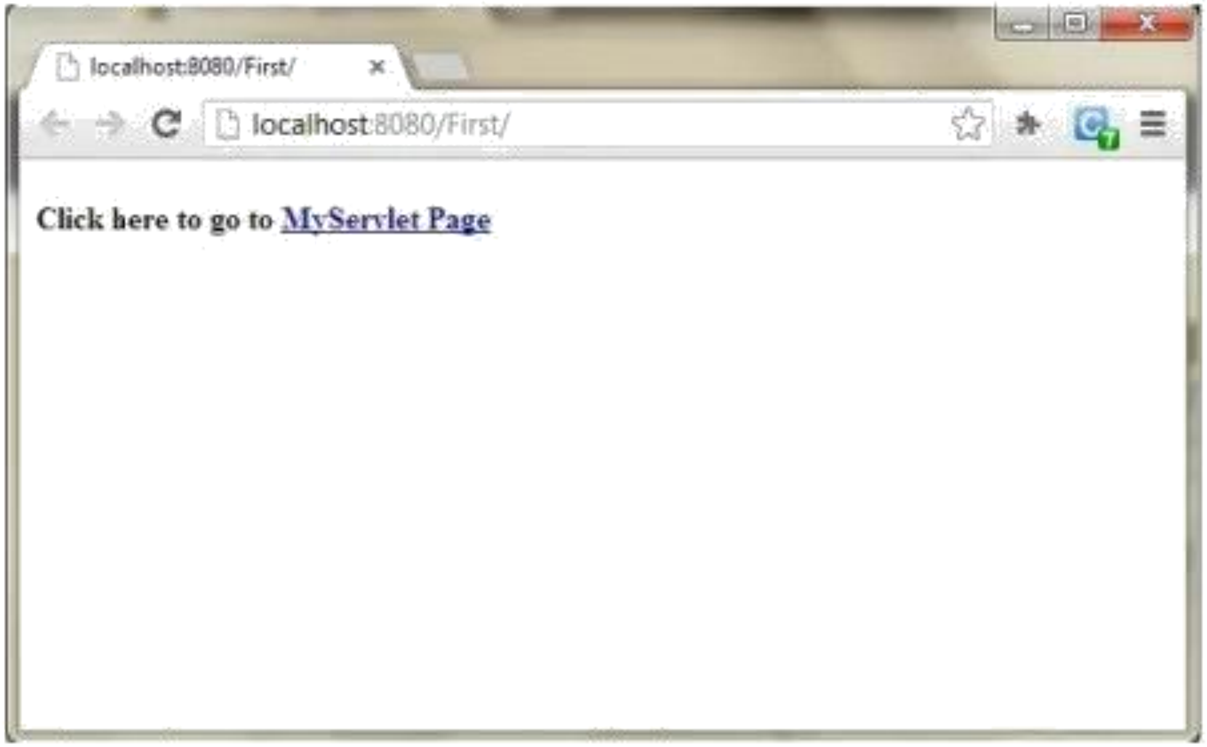
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">

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

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



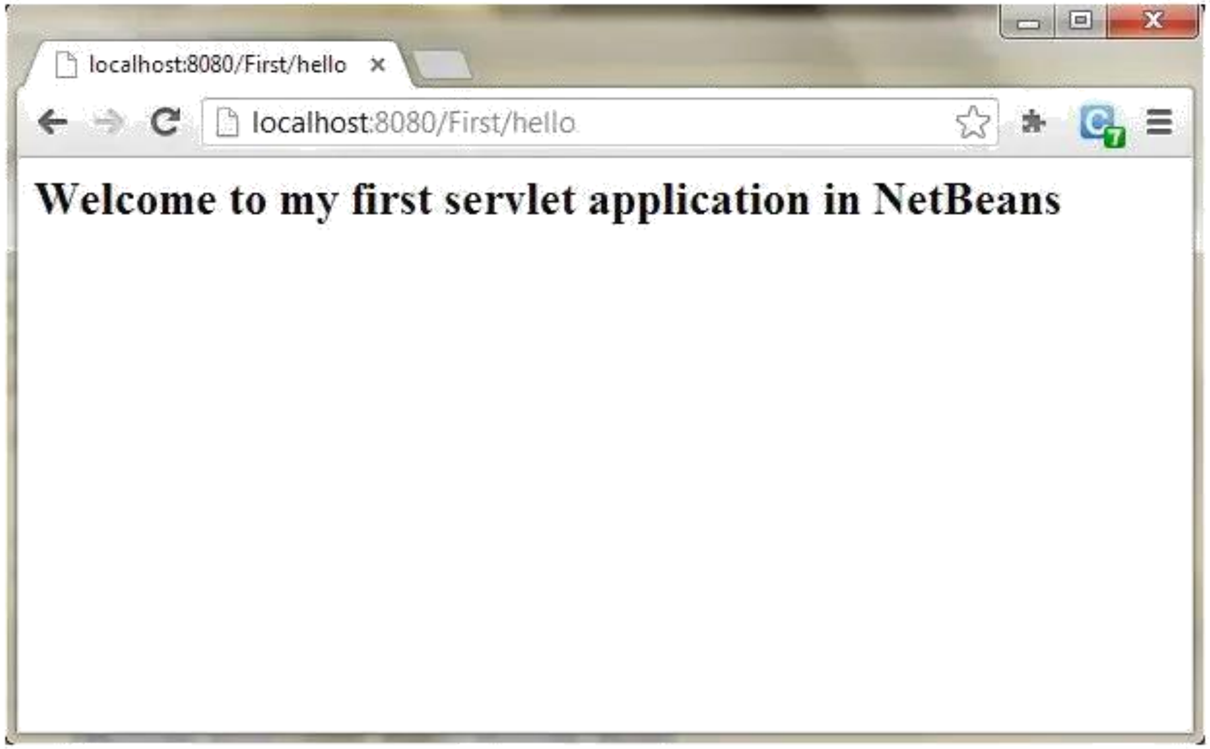
```
<!DOCTYPE html>
<html>
    <head>
     <title></title>
    </head>
    <body>
      <h4>Click here to go to <a href=
    </body>
</html>
```

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



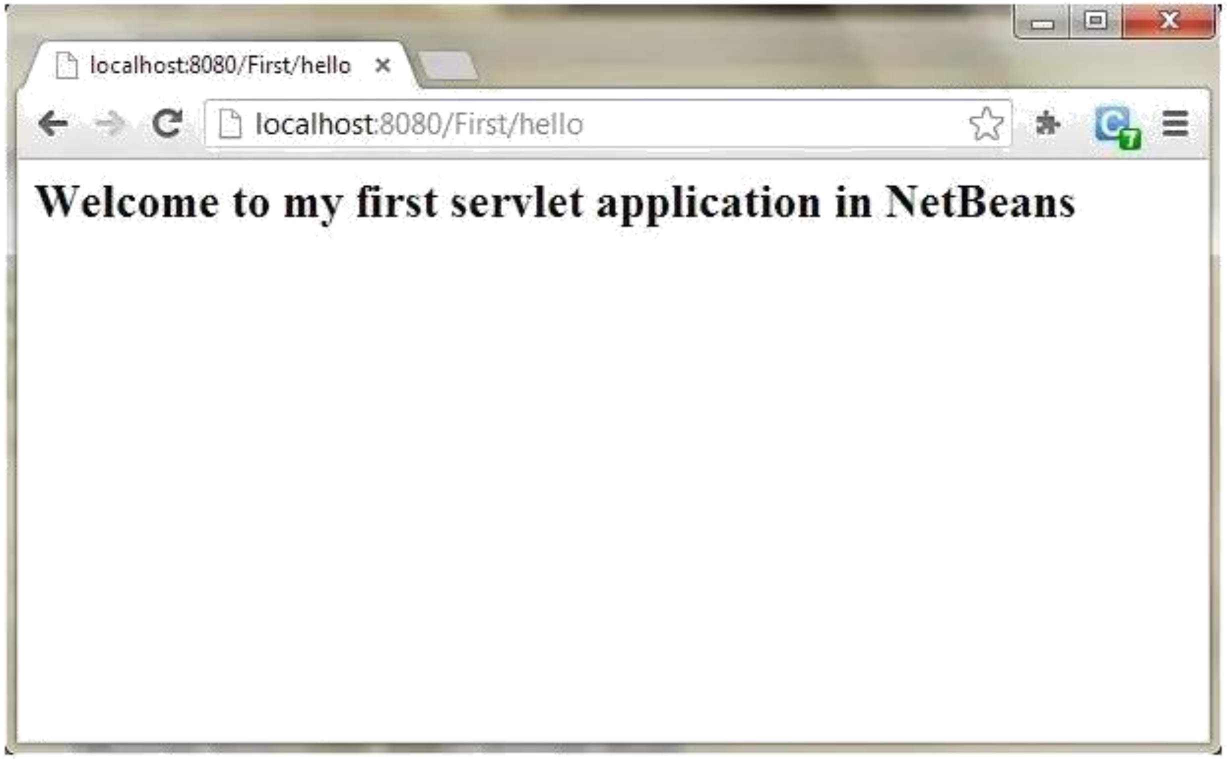
Click here to go to MyServlet Page

5. Hurray! Our First Servlet class is running.



Welcome to my first servlet application in NetBeans

16. Hurray! Our First Servlet class is running.



Welcome to my first servlet application in NetBeans

DemoServlet.java

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class DemoServlet extends HttpServlet{

      public void doGet(HttpServletRequest req,HttpServletResponse res)
            throws ServletException,IOException
      {
            res.setContentType("text/html");//setting the content type
            PrintWriter out=res.getWriter();//get the stream to write
            the data

            //writing html in the stream
            out.println("<html><body>");
            out.println("Welcome to servlet");
            out.println("</body></html>");

      }
}
```

web.xml file

```
<web-app>

<servlet>
<servlet-name> DemoServlet </servlet-name>
<servlet-class>DemoServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name> DemoServlet </servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

</web-app>
```

<web-app> represents the whole application.
<servlet> is sub element of **<web-app>** and represents the servlet.
<servlet-name> is sub element of <servlet> represents the name of the servlet.
<servlet-class> is sub element of <servlet> represents the class of the servlet.
<servlet-mapping> is sub element of <web-app>. It is used to map the servlet.
<url-pattern> is sub element of <servlet-mapping>. This pattern is used at client side to invoke the servlet.

Example of Servlet Interface

```
import java.io.*;
import javax.servlet.*;
public class DemoServlet implements Servlet{ ServletConfig
   config=null;
   public void init(ServletConfig config){
      this.config=config;
      System.out.println("Initialization complete");
   }

   public void service(ServletRequest req,ServletResponse res) throws
   IOException,ServletException{
       res.setContentType("text/html");
       PrintWriter pwriter=res.getWriter();
       pwriter.print("<html>");
       pwriter.print("<body>");
       pwriter.print("<h1>Servlet Example Program</h1>");
       pwriter.print("</body>");
       pwriter.print("</html>");
   }
   public void destroy(){
       System.out.println("servlet life cycle finished");
   }
   public ServletConfig getServletConfig(){ return config;

   }
   public String getServletInfo(){
       return "Servlet Test";
   }
}
```

Example of Generic Servlet Class:

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

public class ExampleGeneric extends GenericServlet{

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

              res.setContentType("text/html");
              PrintWriter pwriter=res.getWriter();
              pwriter.print("<html>");
              pwriter.print("<body>");
              pwriter.print("<h2>Generic Servlet Example</h2>");
              pwriter.print("</body>");
              pwriter.print("</html>");
       }
 }
```

Since Generic Servlet is a class, so we don’t need to write all the methods of this class like in Servlet Interface.

Reading Servlet Parameters (Handling form data)

Servlets handles form data parsing automatically using the following methods depending on the situation −

- **getParameter()** − You call request.getParameter() method to get the value of a form parameter.
- **getParameterValues()** − Call this method if the parameter appears more than once and returns multiple values, for example checkbox.
- **getParameterNames()** − Call this method if you want a complete list of all parameters in the current request.

Reading data with GET Request

Example is shown below:

test.html

```
<html>
<body>
   <form method="GET" action="FirstServlet">
           Name: <input type="text" name="name"/>
           <br/>
           Address: <input type="text" name="address"/>
           <br/>
           <input type="submit" value="Submit"/>
   </form>
</body>
</html>
```

FirstServlet.java

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

public class FirstServlet extends HttpServlet {

      //doGet handles get request
public void doGet(HttpServletRequest req, HttpServletResponse
            res) throws ServletException,IOException{
            //reading form data
            String name=req.getParameter("name");
            String address=req.getParameter("address");

            //setting content type
            res.setContentType("text/html");
            PrintWriter out=res.getWriter();
            //displaying data in html
            out.println("<html>");
            out.println("<body><p> Name: "+name+"<br>");
            out.println("Address: "+address+"</p></body></html>");

      }
}
```

Reading data with POST Request

Servlet handles this type of requests using **doPost()** method.

test.html

```
<html>
<body>
   <form method="POST" action="FirstServlet">
           Name: <input type="text" name="name"/>
           <br/>
           Address: <input type="text" name="address"/>
           <br/>
           <input type="submit" value="Submit"/>
   </form>
</body>
</html>
```

FirstServlet.java

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

public class FirstServlet extends HttpServlet
        { //doPost handles post request
public void doPost(HttpServletRequest req, HttpServletResponse res)
                throws ServletException,IOException{
                //reading form data
                String name=req.getParameter("name");
                String address=req.getParameter("address");

                //setting content type
                res.setContentType("text/html");
                PrintWriter out=res.getWriter();
                //displaying data in html
                out.println("<html>");
                out.println("<body><p> Name: "+name+"<br>");
                out.println("Address: "+address+"</p></body></html>");
        }
}
```

Reading All Form Parameters

getParameterNames() method of HttpServletRequest is used to read all the available form parameters.
using ***hasMoreElements()*** method to determine when to stop and using ***nextElement()*** method to get each parameter name.

Example is shown below:

test.html

```
<html>
<body>
   <form method="POST" action="FirstServlet">
           Courses:
           <input type="checkbox" name="spring" value="Spring"
                 checked/> Spring
```

```
            <input type="checkbox" name="django" value="Django"/>
                  Django
            <input type="checkbox" name="laravel" value="Laravel"/>
                  Laravel
            <input type="checkbox" name="dotnet" value="Dot Net"/>
                  Dot Net
            <br>
            <input type="submit" value="Submit"/>
    </form>
</body>
</html>
```

FirstServlet.java

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

public class FirstServlet extends HttpServlet
     { //doPost handles get request
    public void doPost(HttpServletRequest req, HttpServletResponse
            res) throws ServletException,IOException{

           res.setContentType("text/html");
           PrintWriter out=res.getWriter();
           out.println("<html><body>");
           out.println("You have Selected: <br>");

           //getting all parameters at once
           Enumeration paramNames=req.getParameterNames();

           //looping to get all parameters
           while(paramNames.hasMoreElements()) {
                 //getting single parameter
                 String pname=(String) paramNames.nextElement();

           /*
                we can put all values in single array and access
                later String values[]=req.getParameterValues(pname);
           */

                 //getting parameter value individually
                 String value=req.getParameter(pname);
                 //displaying value
                 out.println(value+"<br>");
           }
           out.println("</body></html>");
     }
}
```

HTTP Header Response Example

You already have seen setContentType() method working in previous examples and following example would also use same method, additionally we would use **setIntHeader()** method to set **Refresh** header.

```
import javax.servlet.*;
```

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

@WebServlet("/HeaderEx")
public class HeaderEx extends HttpServlet {

protected void doGet(HttpServletRequest req, HttpServletResponse
                 res) throws ServletException, IOException {

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

           //refreshing page in every 5 seconds
           res.setIntHeader("Refresh", 5);

           //getting current date and time
           Date date=new Date();

           out.println("Page Refreshed at: "+date);
      }
}
```

Request Dispatcher

The RequestDispatcher interface defines an object that receives the request from client and dispatches it to the resource(such as servlet, JSP, HTML file).

Example of Request Dispatcher:

index.html

```html
<form action="Login" method="post">
      Name:<input type="text" name="userName"/><br/>
      Password:<input type="password" name="userPass"/><br/>
      <input type="submit" value="login"/>
</form>
```

Login.java

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Login extends HttpServlet {
   public void doPost(HttpServletRequest request, HttpServletResponse r
   esponse) throws ServletException, IOException {
       response.setContentType("text/html");
       PrintWriter out = response.getWriter();
```

```
        String n=request.getParameter("userName");
        String p=request.getParameter("userPass");
        if(p.equals("ram")){
                RequestDispatcher rd=request.
                        getRequestDispatcher("WelcomeServlet");
            rd.forward(request, response);
        }
        else{
            out.print("Sorry UserName or Password
            Error!"); RequestDispatcher rd=request.
                  getRequestDispatcher("/index.html");

            rd.include(request, response); }

        }
    }
```

WelcomeServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class WelcomeServlet extends HttpServlet {
   public void doPost(HttpServletRequest request, HttpServletResponse
           response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String n=request.getParameter("userName");
    out.print("Welcome "+n); }

}
```

Example of Cookie in Servlet:

Here we're create **two servlets**, one for setting a cookie and another for accessing and displaying cookie values.

index.html

```
<html>
<body>
     <a href="first">Set Cookie</a>
     <a href="second">Display Cookie</a>
</body>
</html>
```

FirstServlet.java

```
import javax.servlet.*;
```

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

@WebServlet("/first")
public class FirstServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException,IOException{
           //setting a cookie values
           Cookie c1=new Cookie("username","ram");
           Cookie c2=new Cookie("password","admin");

           //adding a cookie in response
           res.addCookie(c1);
           res.addCookie(c2);

           res.setContentType("text/html");
           PrintWriter out=res.getWriter();
           out.println("Cookie Added Successfully!");
      }
}
```

SecondServlet.java

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

@WebServlet("/second")
public class SecondServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException,IOException{

           //accessing cookie values
           Cookie c[]=req.getCookies();

           //displaying cookies
           res.setContentType("text/html");
           PrintWriter out=res.getWriter();

           out.println("Username: "+c[0].getValue()+"<br>");
           out.println("Password: "+c[1].getValue());

      }
}
```

Session Management in Servlet

The HttpSession object is used for session management. 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. The subsequent request will not be served from this session anymore, the servlet container will create a new session.

This is how you create a HttpSession object.

```
protected void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
        HttpSession session = req.getSession();
}
```

You can store the user information into the session object by using setAttribute() method and later when needed this information can be fetched from the session.

```
session.setAttribute("username", "ram");
session.setAttribute("password", "admin");
```

To get the value from session we use the **getAttribute() method of HttpSession interface**. Here we are fetching the attribute values using attribute names.

```
String userName = (String) session.getAttribute("username");
```

Methods of HttpSession

- **public void setAttribute(String name, Object value)**: Binds the object with a name and stores the name/value pair as an attribute of the HttpSession object. If an attribute already exists, then this method replaces the existing attributes.
- **public Object getAttribute(String name)**: Returns the String object specified in the parameter, from the session object. If no object is found for the specified attribute, then the getAttribute() method returns null.
- **public Enumeration getAttributeNames()**: Returns an Enumeration that contains the name of all the objects that are bound as attributes to the session object.
- **public void removeAttribute(String name)**: Removes the given attribute from session.
- **setMaxInactiveInterval(int interval)**: Sets the session inactivity time in seconds. This is the time in seconds that specifies how long a sessions remains active since last request received from client.

Example of HttpSession

index.html

```
<html>
<body>
     <a href="first">Set Session</a>
     <a href="second">Access Session</a>
</body>
</html>
```

FirstServlet.java

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

```java
@WebServlet("/first")
public class FirstServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException,IOException{

           //setting a session
           HttpSession session=req.getSession();
           session.setAttribute("userid", "10115");
           session.setAttribute("username", "Ram");

           res.setContentType("text/html");
           PrintWriter out=res.getWriter();
           out.println("Session Set Successfully!");
     }
}
```

SecondServlet.java

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

@WebServlet("/second")
public class SecondServlet extends HttpServlet {

public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException,IOException{

           //accessing session
        HttpSession session=req.getSession(false);
        String userId=(String) session.getAttribute("userid");
        String username=(String) session.getAttribute("username");



           //displaying session values
           res.setContentType("text/html");
           PrintWriter out=res.getWriter();

           out.println("User Id: "+userId+"<br>");
           out.println("Username: "+username);

     }
}
```

CRUD operation using Servlet

- As we know that for database operation, we need JDBC driver JAR file. Since, we create web application using servlet, **we need to put JAR file under webapp/WEB-INF/lib directory.**

DbConnection.java

```
import java.sql.*;
public class DbConnection {
	public static Connection getConn() throws Exception{
	    Class.forName("org.sqlite.JDBC");
	    String dbUrl="jdbc:sqlite:mydb";
	    Connection conn=DriverManager.getConnection(dbUrl);

	    Statement st=conn.createStatement();
	    String sql="CREATE TABLE If not exists employees(eid INT, name
	              VARCHAR(30), address VARCHAR(30))";
	    st.execute(sql);
	    return conn;
	}
}
```

Index.html

```
<html>
<head>
<title>Employee Crud</title>
</head>
<body>
	<form method="POST" action="myservlet">
	Employee Id: <input type="text" name="eid"/>
	<br>
	Name: <input type="text" name="name"/>
	<br>
	Address: <input type="text" name="address"/>
	<br> <br>
	<input type="submit" value="Insert" name="insert"/>
	<input type="submit" value="Update" name="update"/>
	<input type="submit" value="Delete" name="delete"/>
	</form>
	<br><br>
	<a href="view">View Employee Records</a>
</body>
</html>
```

MyServlet.java

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

@WebServlet("/myservlet")
public class MyServlet extends HttpServlet{

public void doPost(HttpServletRequest req, HttpServletResponse res) throws
                ServletException,IOException{

        //getting request values
        int eid=Integer.parseInt(req.getParameter("eid")); String
        name=req.getParameter("name");
        String address=req.getParameter("address");

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

          try {
              Connection conn=DbConnection.getConn();

              //handling button clicks
              if(req.getParameter("insert")!=null) {
                      //insert button clicked //inserting
                      data
                      String sql="INSERT INTO employees
                              (eid,name,address) VALUES (?,?,?)";
                      PreparedStatement pst=conn.prepareStatement(sql);
                      pst.setInt(1,eid);
                      pst.setString(2, name);
                      pst.setString(3, address);
                      pst.executeUpdate();
                      //displaying message in javascript alert
              out.println("<script>alert('Inserted Successfully!');"
                          +"window.location.href='view'</script>");
              }

              else if(req.getParameter("update")!=null) {
                      //update button Clicked
                      String sql="UPDATE employees SET name=?,
                          address=? WHERE eid=?";
```

```
                    PreparedStatement pst=conn.prepareStatement(sql);
                    pst.setString(1, name);
                    pst.setString(2, address);
                    pst.setInt(3,eid);
                    pst.executeUpdate();
                    //displaying message in javascript alert
                out.println("<script>alert('Updated Successfully!');"
                        +"window.location.href='view'</script>");
                }

               else if(req.getParameter("delete")!=null) {
                    //delete button clicked
                    String sql="DELETE FROM employees WHERE eid=?";
                    PreparedStatement pst=conn.prepareStatement(sql);
                    pst.setInt(1, eid);
                    pst.executeUpdate();
                       //displaying message in javascript alert
               out.println("<script>alert('Deleted Successfully!');"
                        +"window.location.href='view'</script>");
                }

             }catch(Exception ex) {
                System.out.println(ex.toString());
             }

        }
}
```

ViewServlet.java

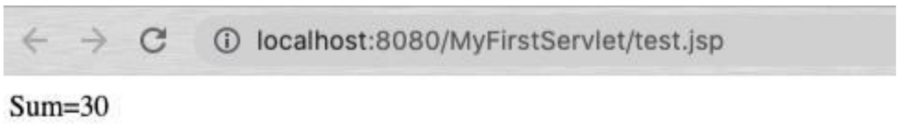
```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.*;
import java.sql.*;

@WebServlet("/view")
public class ViewServlet extends HttpServlet{

public void doGet(HttpServletRequest req, HttpServletResponse res) throws
             ServletException,IOException{

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

```
        //selecting data
            try {
                Connection conn=DbConnection.getConn();
                String sql="SELECT * FROM employees";
                PreparedStatement pst=conn.prepareStatement(sql);
                ResultSet rs=pst.executeQuery();

                out.println("<html><body>");
                out.println("<a href='index.html'>
                                Goto Index </a> <br><br>");
                //displaying data
                out.println("<table>");
                out.println("<tr>");
                out.println("<th> Eid </th>");
                out.println("<th> Name </th>");
                out.println("<th> Address </th>");
                out.println("</tr>");


                    while(rs.next()) {
                         out.println("<tr>");
                        out.println("<td>"+rs.getInt(1)+"</td>");
                        out.println("<td>"+rs.getString(2)+"</td>");
                        out.println("<td>"+rs.getString(3)+"</td>");
                        out.println("<td>");
                        out.println("</td>");
                        out.println("</tr>");
                }

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


            }catch(Exception ex) {
                    System.out.println(ex.toString());
            }
    }
}
```

Steps for creating and Running a JSP in Tomcat Server using Eclipse IDE:

1. Create a Dynamic Web Project.
2. Create a JSP page under webapp directory.
3. Write following JSP code:
```
<html>
<head>
<title>Insert title here</title>
</head>
<body>
    <%
    int a=10,b=20,sum;
    sum=a+b;
    out.print("Sum="+sum);
    %>
</body>
</html>
```
4. Run the project.

Output:


localhost:8080/MyFirstServlet/test.jsp

```
Sum=30
```

Handling form data using JSP

Form handling in JSP is similar to servlets. We can use request object for reading parameter values from HTML file or other JSP file. Methods supported by request object is similar to methods used in servlets.

index.html
```
<html>
<body>
     <form method="POST" action="add.jsp">
          First Number: <input type="text" name="first"/>
          <br>
          Second Number: <input type="text" name="second"/>
          <br><br>
          <input type="submit" value="Calculate"/>
     </form>
</body>
</html>
```

add.jsp
```
<html>
<body>
     <%!
        private int a,b,sum;
     %>

     <%
       a=Integer.parseInt(request.getParameter("first"));
       b=Integer.parseInt(request.getParameter("second"));
       sum=a+b;
```

```
        %>

    <h2 style="color:blue;">
       <%  out.println("Sum="+sum); %>
    </h2>
</body>
</html>
```

Working with database using JSP:

index.html

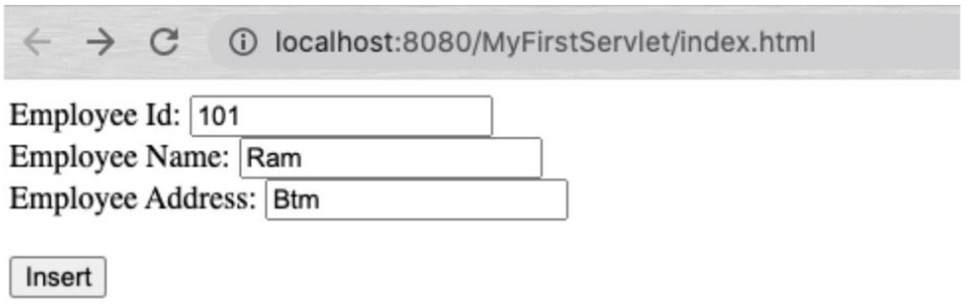
```
<html>
<body>
     <form method="POST" action="insert.jsp">
         Employee Id: <input type="text" name="eid"/>
         <br>
         Employee Name: <input type="text" name="name"/>
         <br>
         Employee Address: <input type="text" name="address"/>
         <br><br>
         <input type="submit" value="Insert"/>
     </form>
</body>
</html>
```

insert.jsp

```
<html>
<body>
    <%
       int eid=Integer.parseInt(request.getParameter("eid"));
       String name=request.getParameter("name");
       String address=request.getParameter("address");
    %>

    <%@ page import="java.sql.*" %>

    <%
       Class.forName("org.sqlite.JDBC");
       String dbUrl="jdbc:sqlite:mydb";
         Connection conn=DriverManager.getConnection(dbUrl);

         Statement st=conn.createStatement();
         String sql="CREATE TABLE If not exists employee(eid INT,
                 name VARCHAR(30), address VARCHAR(30))";
         st.execute(sql);
```

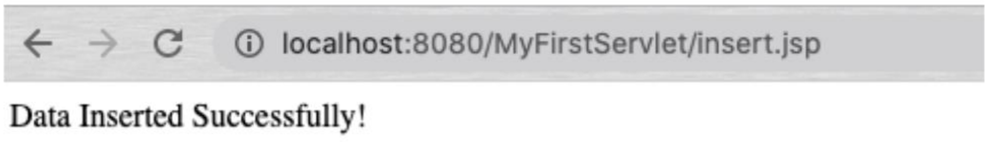
```
        //inserting data
        String sql1="insert into employee(eid,name,address) values
                           (?,?,?)";
        PreparedStatement pst=conn.prepareStatement(sql1);
        pst.setInt(1, eid);
        pst.setString(2,name);
        pst.setString(3,address);
        pst.executeUpdate();

        out.println("Data Inserted Successfully!");
    %>

</body>
</html>
```

Output:



localhost:8080/MyFirstServlet/index.html

Employee Id:	101
Employee Name:	Ram
Employee Address:	Btm



localhost:8080/MyFirstServlet/insert.jsp

Data Inserted Successfully!

Selecting and displaying data in Table:

view.jsp

```
<html>
<body>
   <%@ page import="java.sql.*" %>
       <%
          Class.forName("org.sqlite.JDBC");
          String dbUrl="jdbc:sqlite:mydb";
          Connection conn=DriverManager.getConnection(dbUrl);

           //selecting data
           String sql="select * from employee";
           PreparedStatement pst=conn.prepareStatement(sql);
           ResultSet rs=pst.executeQuery();
       %>
    <table>
       <tr>
            <th> Eid </th>
            <th> Name </th>
            <th> Address </th>
        </tr>
```

```
         <%
              while(rs.next()){
                out.print("<tr>");
                 out.print("<td>"+rs.getInt(1)+"</td>");
                 out.print("<td>"+rs.getString(2)+"</td>");
                 out.print("<td>"+rs.getString(3)+"</td>");
                 out.print("</tr>");
              }
         %>
     </table>
</body>
</html>
```

Session Management in JSP

Example is shown below:

index.html

```
<html>
<body>
    <form method="POST" action="setsession.jsp">
        Username:
        <input type="text" name="username"/>
        <br>
        <input type="submit" value="Submit"/>
    </form>
</body>
</html>
```

setsession.jsp

```
<html>
<body>
    <%
       String uname=request.getParameter("username");
       session.setAttribute("username", uname);
       out.println("Session set Successfully!");
    %>

    <a href="viewsession.jsp">Click to access session</a>
</body>
</html>
```

viewsession.jsp

```
<html>
<body>
     <%
        String uname=(String)session.getAttribute("username");
        out.println("Welcome: "+uname);
     %>
</body>
</html>
```

Exception Handling in JSP

Exception handling by the elements of page directive

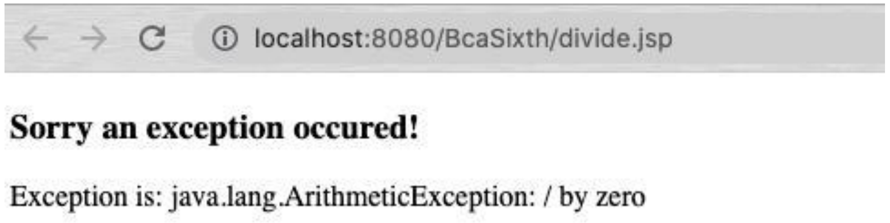
error.jsp

```
<html>
<body>
      <%@ page isErrorPage="true" %>

      <h3>Sorry an exception occured!</h3>

      Exception is: <%= exception %>

</body>
</html>
```

divide.jsp

```
<html>
<body>
    <%@page errorPage="error.jsp" %>
    <%
        int a=10,b=0,div;
        div=a/b;
        out.println("Result="+div);
    %>
</body>
</html>
```

Output:


localhost:8080/BcaSixth/divide.jsp

Sorry an exception occured!

Exception is: java.lang.ArithmeticException: / by zero

Exception handling by specifying the error-page element in web.xml file

web.xml

```
<error-page>
        <exception-type>java.lang.Exception</exception-type>
        <location>/error.jsp</location>
</error-page>
```

divide.jsp

Now, you don't need to specify the errorPage attribute of page directive in the jsp page.

```
                            <html>
                            <body>
                                <%
        int a=10,b=0,div;
        div=a/b;
        out.println("Result="+div);
    %>
</body>
</html>
```

error.jsp file is same as in the above example

Output is also same as in the above example