Developing the First Servlet

Creating a Servlet Using the NetBeans IDE

The following tutorial consists of steps to create a new Java EE 7 project named FirstServlet. Three files are created for the project: index.html, AddServlet.java, and web.xml. index.html is automatically added to the project and the file will be modified. You may notice that NetBeans and the projects are installed on drive N: (a USB flash drive). However, this is only done for portability and instructor testing. It is strongly recommended that you install NetBeans and all other required software development tools on the local C: drive. The configurations are much easier to manage and troubleshoot when all components are installed on the C: drive. Otherwise, you may encounter configuration issues which are difficult to resolve. Also, due to the permissions managed by UAC (User Account Control) in Windows, I recommend that software development tools not be installed in the Program Files directories.

It is important to follow the steps very precisely to help reduce the chances for incorrect configurations and errors. Be patient and take your time. Also, repeating the tutorials in the course for practice will reinforce the concepts, enhance understanding, and improve skills.

When working through this or any other software development course it is especially important to get and remain organized. Devote time to planning an organization strategy for your project files and folders. Organization is an essential skill for web developers. Also, regularly backup your work. Modern hard drives are reliable but drive and system failures do occur.

Note: Some of the early tutorials in this course (INEW 2338) were constructed prior to the release of NetBeans 8.0 and are shown using NetBeans 7.4. All of the NetBeans 7.4 instructions in the course can be performed with NetBeans 8.0. When the tutorials begin using NetBeans 8.0, instructions will be provided to perform the minor project revisions for the upgrade from 7.4 to 8.0.
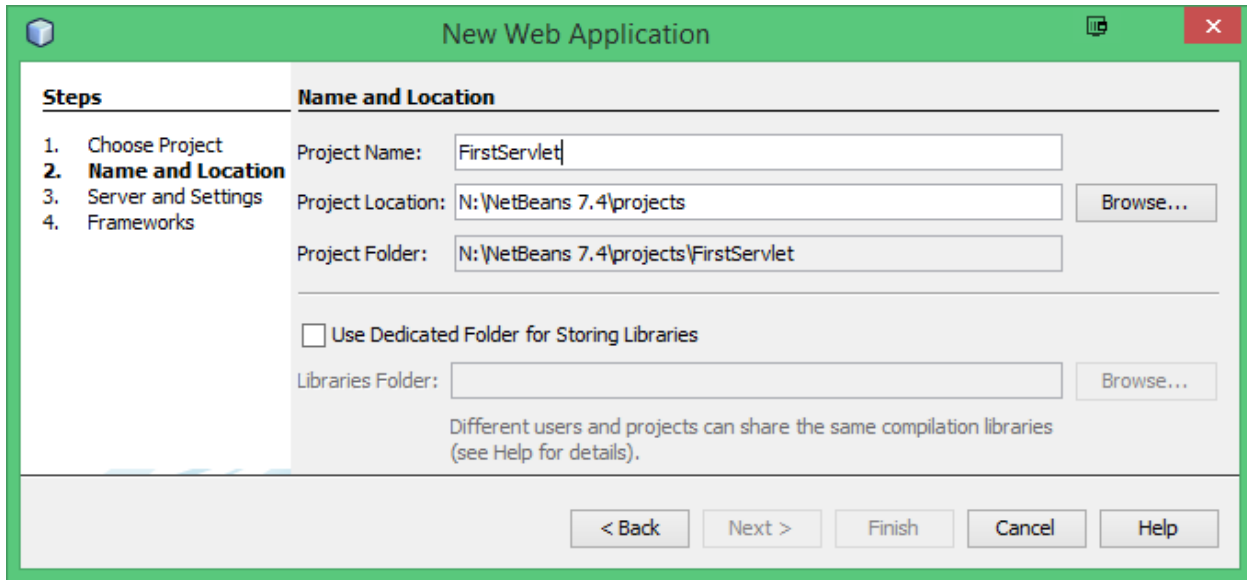
Steps:

Create a new project by clicking the New Project button, making the selections, and then click Next.
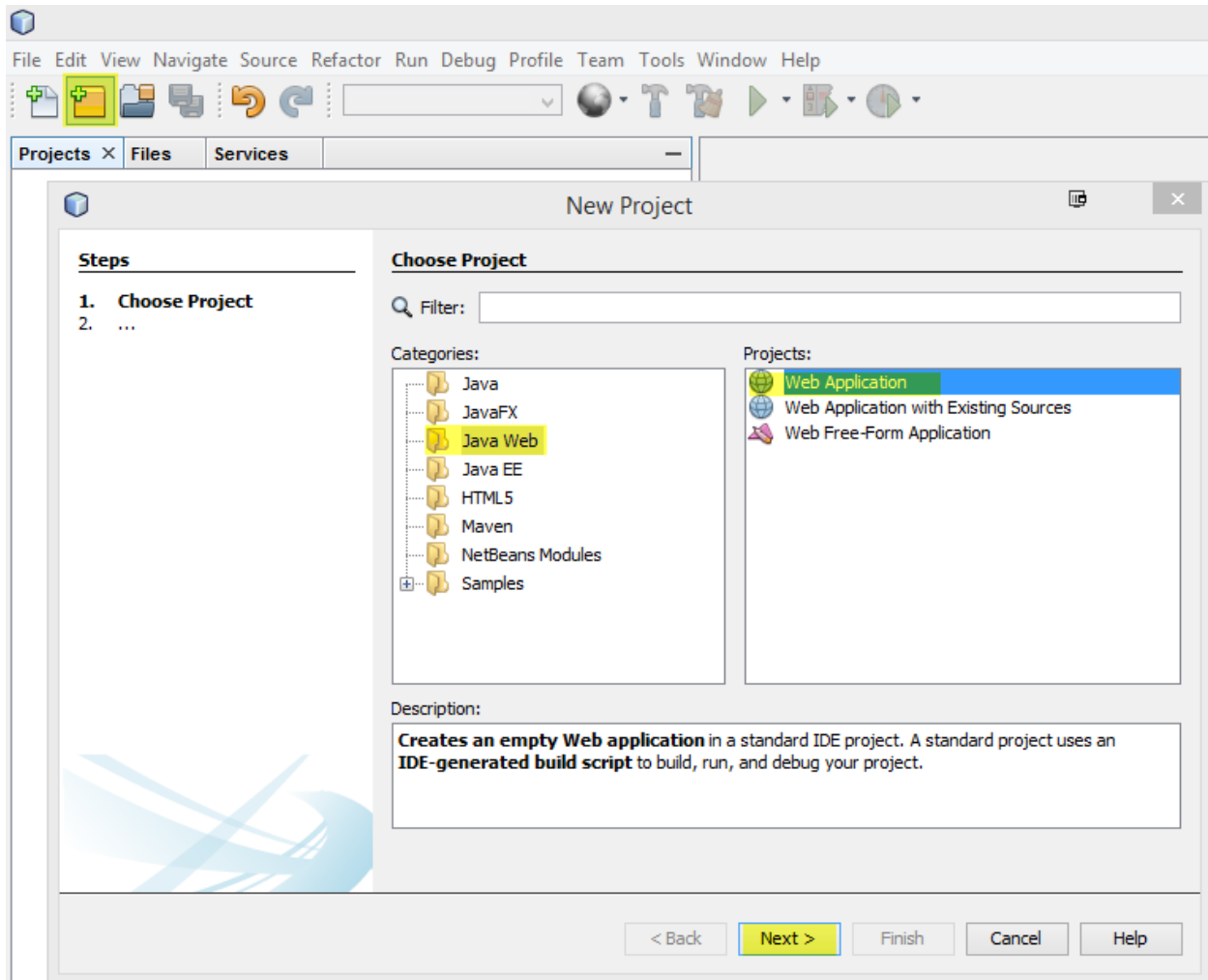
Make selections like those shown (your location/folder will likely be different) and then click Next.

Make selections like those shown and then click Next. Notice the GlassFish Server and Java EE 7 Web items are selected.

The next dialog box presents an opportunity to add frameworks to the application. No need to add a framework to this project. Use of the JSF (JavaServer Faces) framework will be covered in later tutorials. Click Finish.
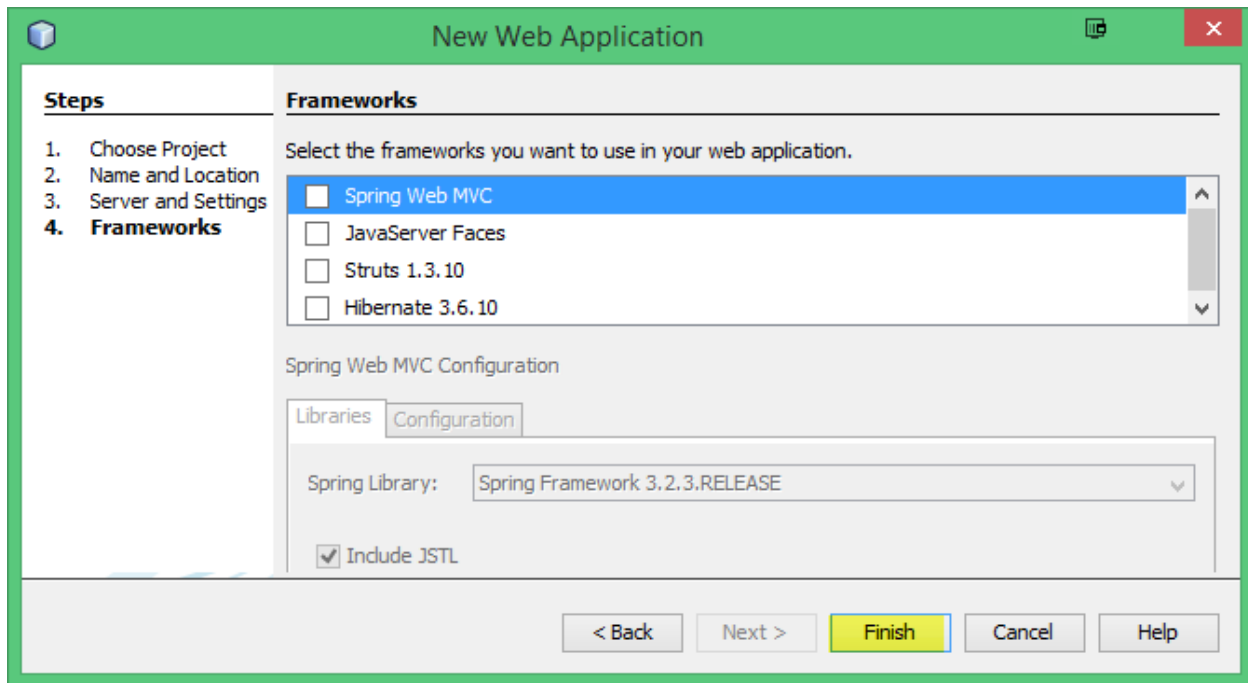
The project is created with the configuration shown in the Projects group (highlighted).
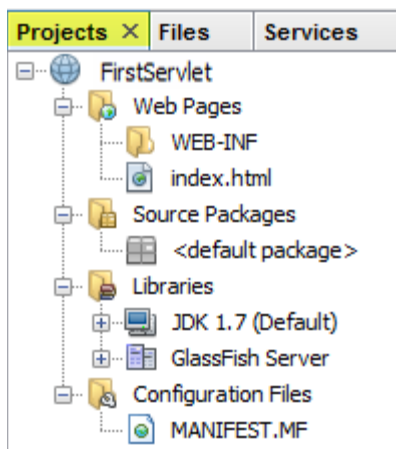
Review the file structure NetBeans automatically created for the project by selecting the Files group (highlighted).
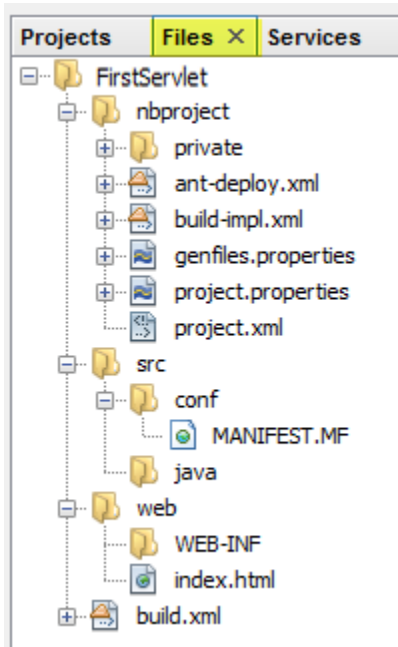
The index.html file is shown below with its default (starting) content.
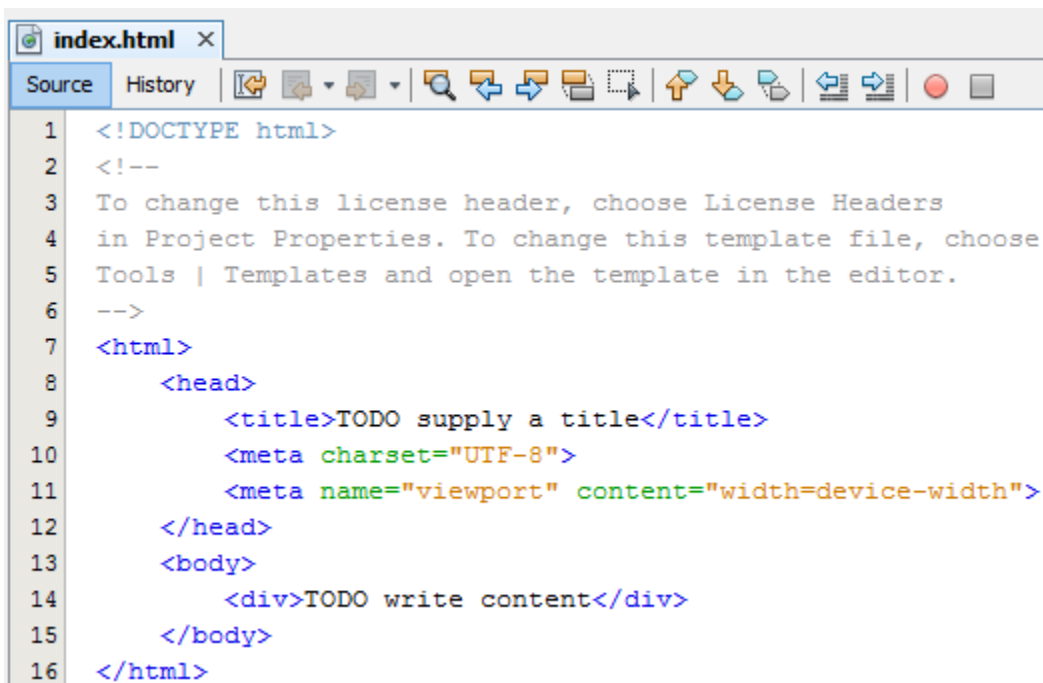

```
1   <!DOCTYPE html>
2   <!--
3   To change this license header, choose License Headers
4   in Project Properties. To change this template file, choose
5   Tools | Templates and open the template in the editor.
6   -->
7   <html>
8       <head>
9           <title>TODO supply a title</title>
10          <meta charset="UTF-8">
11          <meta name="viewport" content="width=device-width">
12      </head>
13      <body>
14          <div>TODO write content</div>
15      </body>
16  </html>
```

R-click the project name "FirstServlet" in the Project group and select Run to run the project. Glassfish is automatically started if it is not running and the index.html page is displayed in a browser as shown.
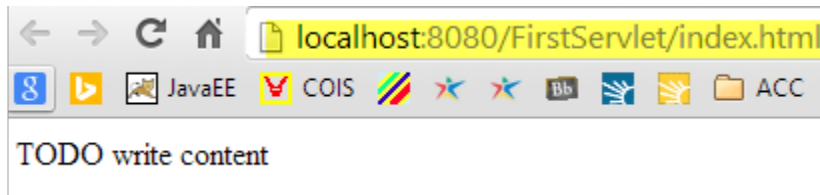
Now we are ready to write a servlet and access it from the index.html page which will be modified in a few steps. In the Project group, R-click Source Packages | New | Java Class…
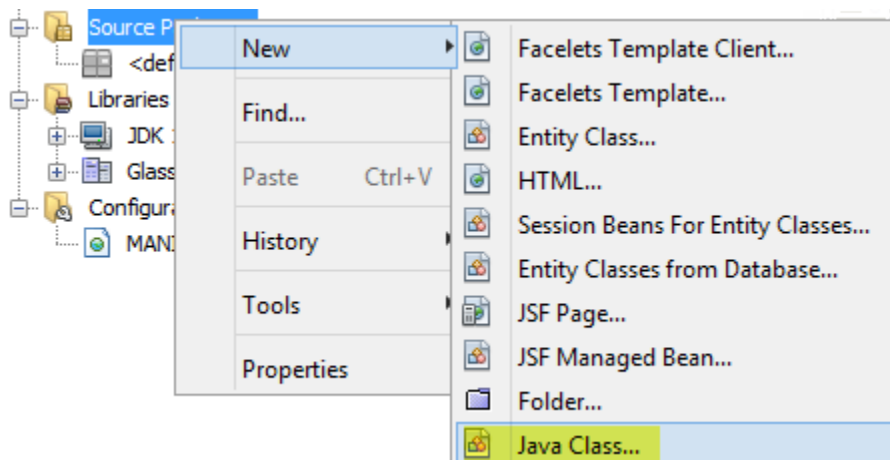
Make the following settings/selections. Any name can be chosen for the Package. It is common practice to name packages using the reverse order of your domain to help reduce the possibility of namespace collisions. Select Finish.
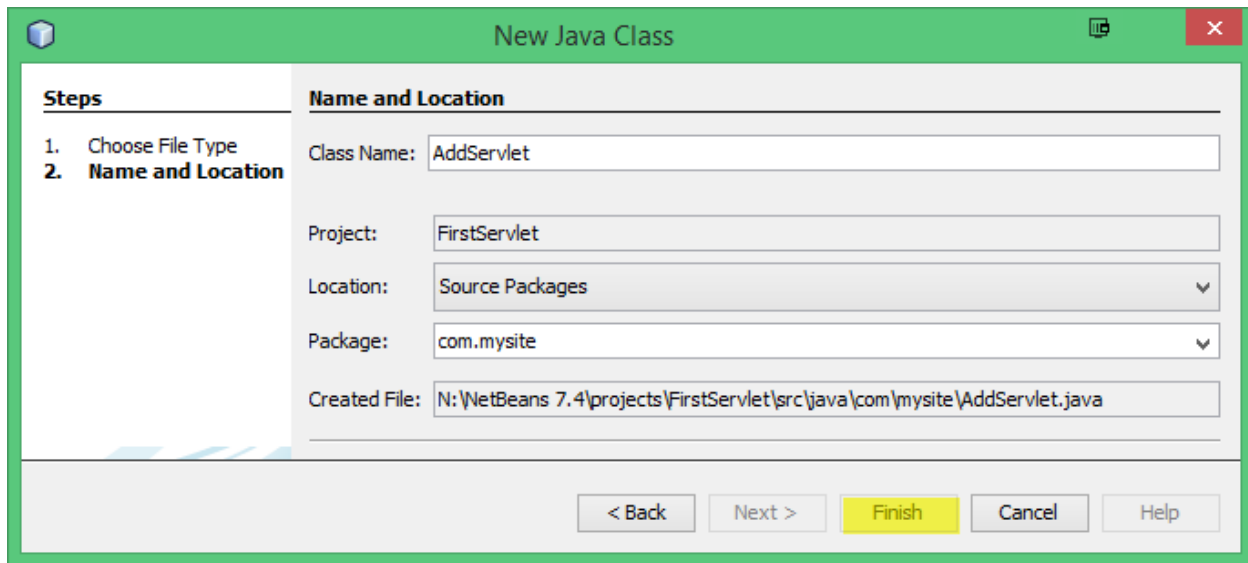
The file and package are created as shown.
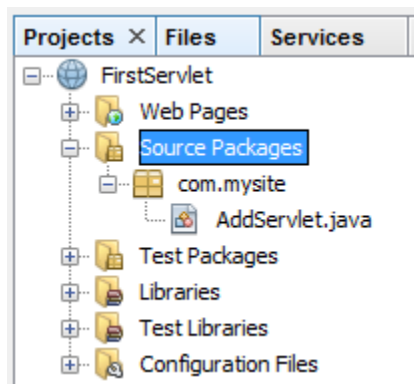
Replace the starting content of AddServlet.java with that shown. Java EE web servlets must extend HttpServlet which occurs on line 9. HTTP servlets have been routinely used to process POST and GET requests submitted by a form. In upcoming tutorials, we will use the more powerful technique of binding client HTML controls directly to server

In the AddServlet.java example, POST requests are processed by the doPost method which accepts requests and response objects as parameters. The MIME type of the response is specified on line 15 which will be text/html. On lines 16 and 17 the values of num1 and num2 are extracted from the request object. On line 18, the try with resources syntax is used which was introduced with Java EE 7. HTML output is produced from lines 19 through 35.

```java
1   /* AddServlet.java */
2   package com.mysite;
3
4   import java.io.IOException;
5   import java.io.PrintWriter;
6   import javax.servlet.*;
7   import javax.servlet.http.*;
8
9   public class AddServlet extends HttpServlet {
10
11      @Override
        public void doPost(HttpServletRequest request,
13              HttpServletResponse response)
14              throws IOException, ServletException {
15          response.setContentType("text/html");
16          String num1 = request.getParameter("num1");
17          String num2 = request.getParameter("num2");
18          try (PrintWriter out = response.getWriter()) {
19              out.println("<html>");
20              out.println("<head>");
21              out.println("<title>First Servlet</title>");
22              out.println("<style>body { font-size: 18px; }</style>");
23              out.println("</head>");
24              out.println("<body>");
25              try {
26                  int sum = Integer.valueOf(num1) + Integer.valueOf(num2);
27                  out.println("<p>Sum: " + num1 + " + " + num2 + " = " + sum + "</p>");
28
29              } catch (java.lang.NumberFormatException ex) {
30                  out.println("<p>Please enter numbers only.</p>");
31              }
32              out.println("<br/><br/>");
33              out.println("<a href='index.html'>Back to index.html</a>");
34              out.println("</body>");
35              out.println("</html>");
36          }
37      }
38  }
```

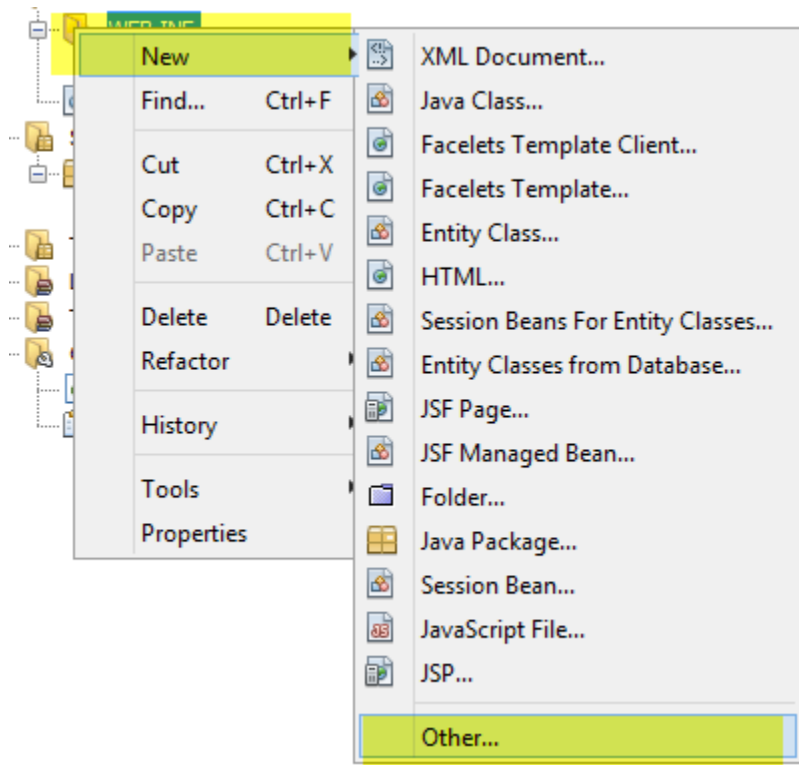Replace the starting content of index.html with that shown.

```
1   <!DOCTYPE html>
2   <!-- index.html -->
3   <html>
4   <head>
5   <title>First Servlet</title>
6   </head>
7   <body>
8        <h1>First Servlet Example</h1>
9        <form method="POST" action="AddServlet">
10           <label for="num1">Enter First Number: </label>
11           <input type="text" id="num1" name="num1"/><br><br>
12           <label for="num2">Enter Second Number: </label>
13           <input type="text" id="num2" name="num2"/><br/><br/>
14           <input type="submit" value="Send to Servlet"/>
15           <input type="reset" value="Reset Form"/>
16       </form>
17  </body>
18  </html>
```

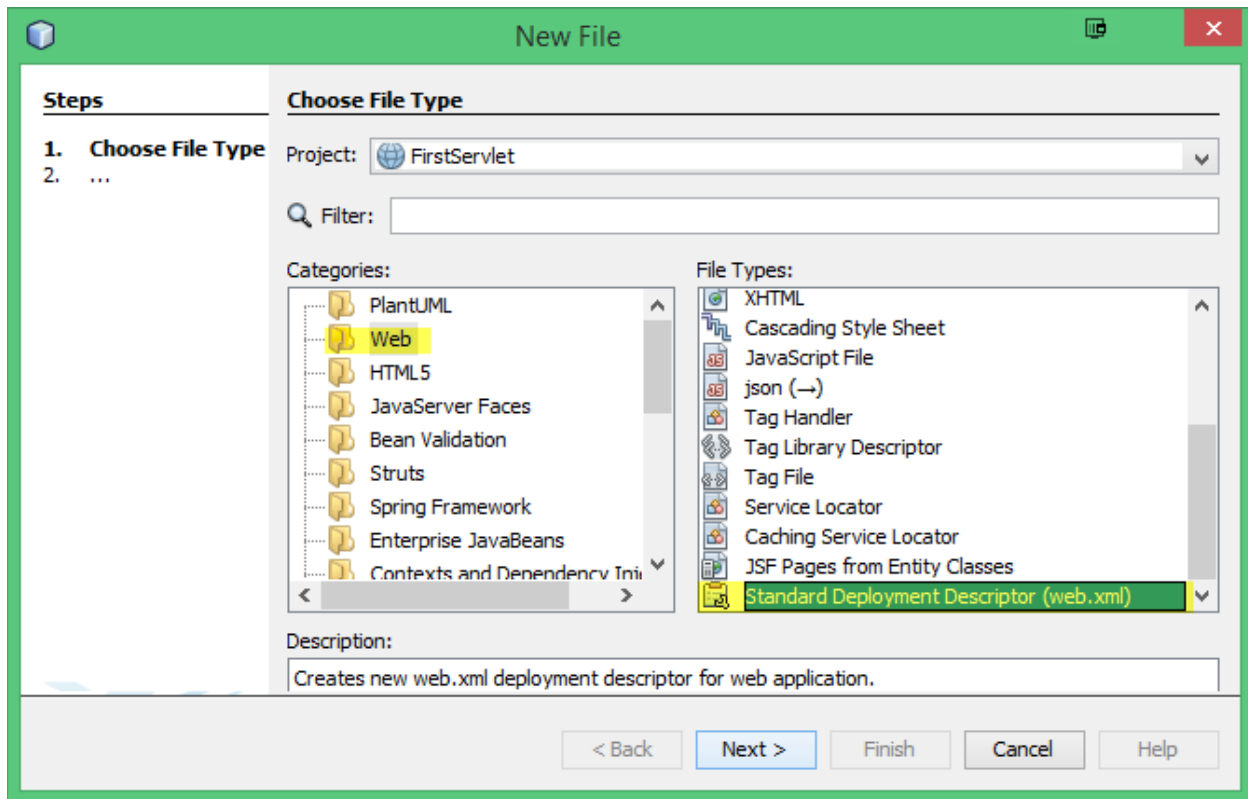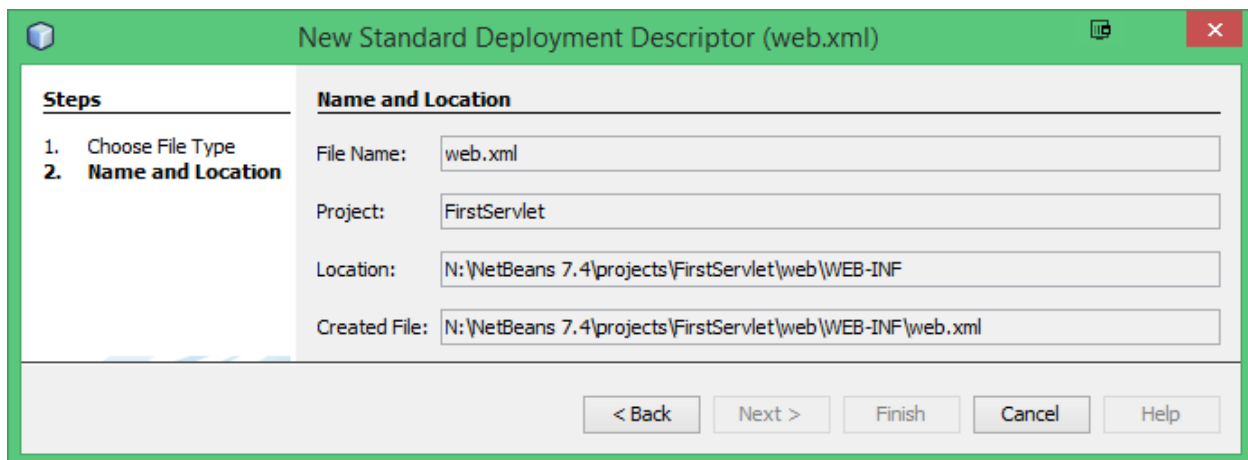R-click on the WEB-INF folder in the Projects group and select New | Other.

Select as shown to create the web.xml which is known as the deployment descriptor and then click Next.

Make the appropriate settings (should be prepopulated as the default settings) and click Finish.

Replace the starting contents of web.xml with that shown. The file contains application configuration settings. In particular, notice the servlet and servlet mapping tags that provide settings for the AddServlet.java file in a previous image.

```xml
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!--web.xml-->
3   <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5            xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
6               http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd">
7       <servlet>
8           <servlet-name>AddServlet</servlet-name>
9           <servlet-class>com.mysite.AddServlet</servlet-class>
10          <load-on-startup>1</load-on-startup>
11      </servlet>
12      <servlet-mapping>
13          <servlet-name>AddServlet</servlet-name>
14          <url-pattern>/AddServlet</url-pattern>
15      </servlet-mapping>
16      <session-config>
17          <session-timeout>
18              30
19          </session-timeout>
20      </session-config>
21      <welcome-file-list>
22          <welcome-file>index.html</welcome-file>
23      </welcome-file-list>
24  </web-app>
```

R-click the project name "FirstServlet" in the Project group and select Run to run the project. Glassfish is automatically started if it is not running and the index.html page is shown. Enter numbers to test the application.

First Servlet Example

Enter First Number: 9

Enter Second Number: 9

Send to Servlet   Reset Form

Select Send to Servlet on the index.html page to see the AddServlet output as shown.

Sum: 9 + 9 = 18

Back to index.html

Thus far, a starting Java EE 7 web application was created which included the files index.html, AddServlet.java, and web.xml. In the next few steps, we will create a copy of the FirstServlet project and name the copy FirstServletAnn. We will make adjustments to use a feature known as annotations which is syntactic metadata. The @Override annotation has already been used on line 11 of AddServlet.java which is used to cause a compilation error if the following method is not found in a parent class or implemented interface. The @Override on line 11 applies to the doPost method which overrides the doPost method of HttpServlet.java. This can be observed by hovering over or clicking on the green dot on line 12.

R-click on the FirstServlet project in the Project group and select Copy… Make the appropriate settings in the Copy Project dialog and select Copy.
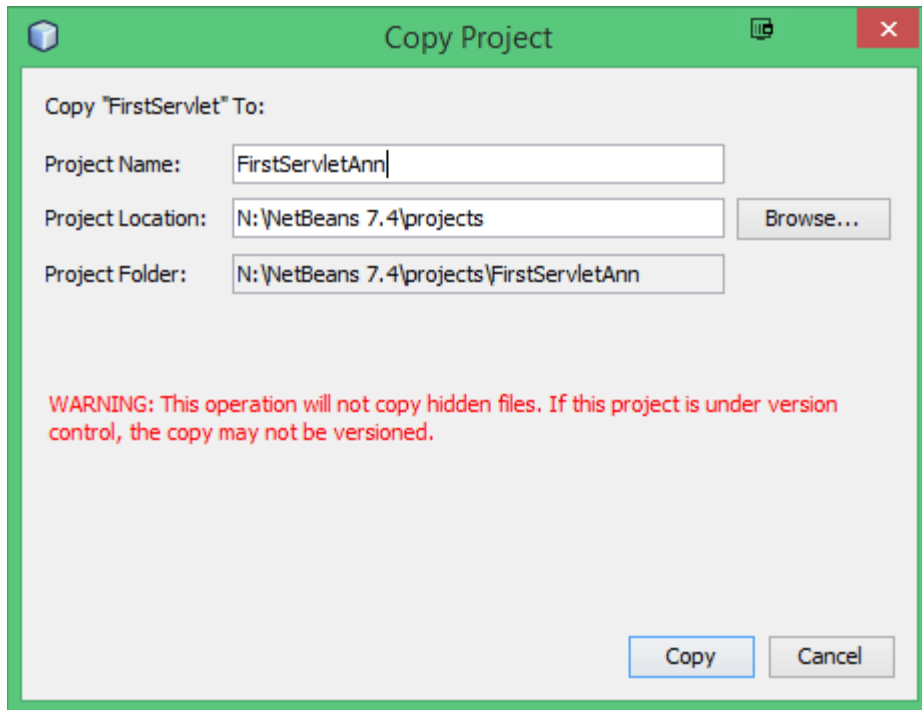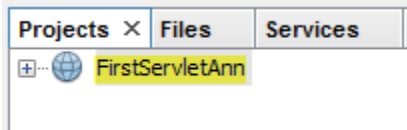

Notice that the FirstServletAnn appears in the Project group. R-click FirstServlet to close the previous version of the project. After you close a project, it can be reopened when/if desired.
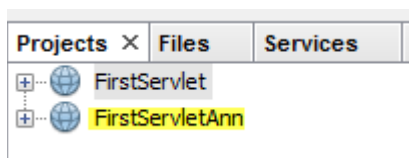
FirstServletAnn should now be the only project open in the Project group as shown.

R-click AddServlet.java and select Refactor | Rename… Change the name to AddServletAnn and select Refactor. Refactoring helps to ensure the name change is made internally to the class and not just to the file name. To observe where refactoring will occur select Preview. After preview, select Do Refactoring at the bottom of the Preview window. Or, the preview can be skipped and select Refactor in the dialog below.
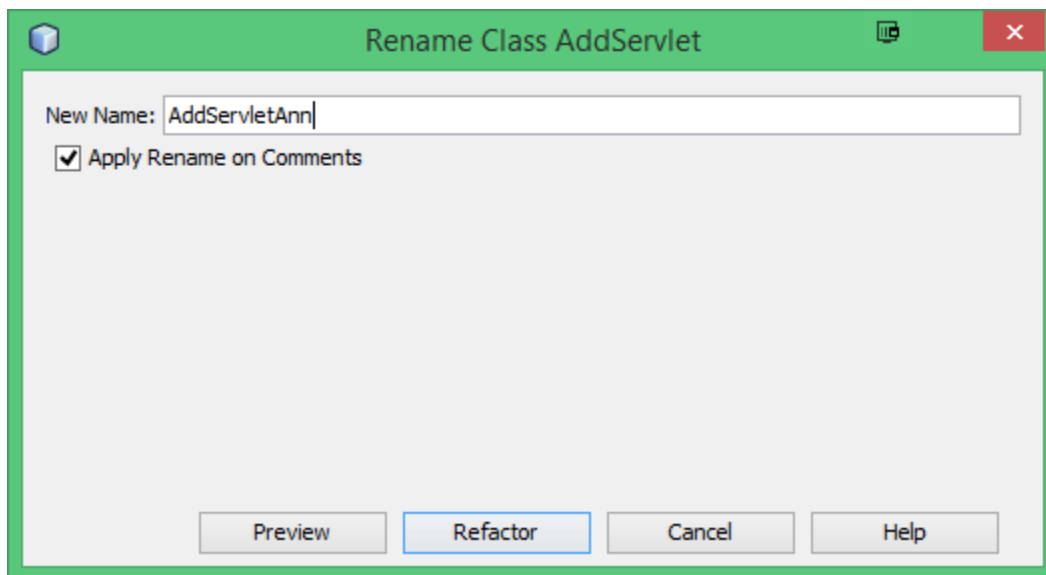
Update AddServeltAnn.java with the modifications shown. All changes occur prior to line 12. Be very attentive. Naturally, one letter discrepancies can prevent an entire web site from functioning.

```java
1   /* AddServletAnn.java */
2   package com.mysite;
3
4   import java.io.IOException;
5   import java.io.PrintWriter;
6   import javax.servlet.*;
7   import javax.servlet.annotation.WebServlet;
8   import javax.servlet.http.*;
9
10  @WebServlet(name = "AddServletAnn", urlPatterns = {"/AddServletAnn"})
11  public class AddServletAnn extends HttpServlet {
12
13      @Override
        public void doPost(HttpServletRequest request,
15              HttpServletResponse response)
16              throws IOException, ServletException {
17          response.setContentType("text/html");
18          String num1 = request.getParameter("num1");
19          String num2 = request.getParameter("num2");
20          try (PrintWriter out = response.getWriter()) {
21              out.println("<html>");
22              out.println("<head>");
23              out.println("<title>First Servlet</title>");
24              out.println("<style>body { font-size: 18px; }</style>");
25              out.println("</head>");
26              out.println("<body>");
27              try {
28                  int sum = Integer.valueOf(num1) + Integer.valueOf(num2);
29                  out.println("<p>Sum: " + num1 + " + " + num2 + " = " + sum + "</p>");
30
31              } catch (java.lang.NumberFormatException ex) {
32                  out.println("<p>Please enter numbers only.</p>");
33              }
34              out.println("<br/><br/>");
35              out.println("<a href='index.html'>Back to index.html</a>");
36              out.println("</body>");
37              out.println("</html>");
38          }
39      }
40  }
```

The @WebServlet annotation on line 10 provides the naming and mapping that was previously supplied by web.xml in the FirstServlet version of the project. Therefore, web.xml is no longer required and should be deleted. After saving the changes to AddServletAnn.java and deleting web.xml, try running the project by R-clicking FirstServletAnn and selecting Run. You should receive a 404 error like that shown. Why does the error appear? Devote a little time/energy to thinking about the cause of the error before moving on.

HTTP Status 404 - Not Found

type Status report

messageNot Found

descriptionThe requested resource is not available.

GlassFish Server Open Source Edition 4.0

To resolve the error, open index.html and modify as shown. There is only one change required. Save the file and go back to index.html in the browser, refresh (F5), and try the page again. It should work with the correction applied.

```
1  <!DOCTYPE html>
2  <!-- index.html -->
3  <html>
4  <head>
5  <title>First Servlet</title>
6  </head>
7  <body>
8      <h1>First Servlet Example</h1>
9      <form method="POST" action="AddServletAnn">
10         <label for="num1">Enter First Number: </label>
11         <input type="text" id="num1" name="num1"/><br><br>
12         <label for="num2">Enter Second Number: </label>
13         <input type="text" id="num2" name="num2"/><br/><br/>
14         <input type="submit" value="Send to Servlet"/>
15         <input type="reset" value="Reset Form"/>
16     </form>
17 </body>
18 </html>
```