

Creating Java EE Applications and Servlets with IntelliJ IDEA

In this tutorial you will:

1. Create IntelliJ IDEA project for Java EE application
2. Create Servlet
3. Deploy the application to JBoss server
4. Experience advanced IntelliJ IDEA coding assistance and code generation features

Prerequisites

To develop Java EE applications with IntelliJ IDEA, download the following software:

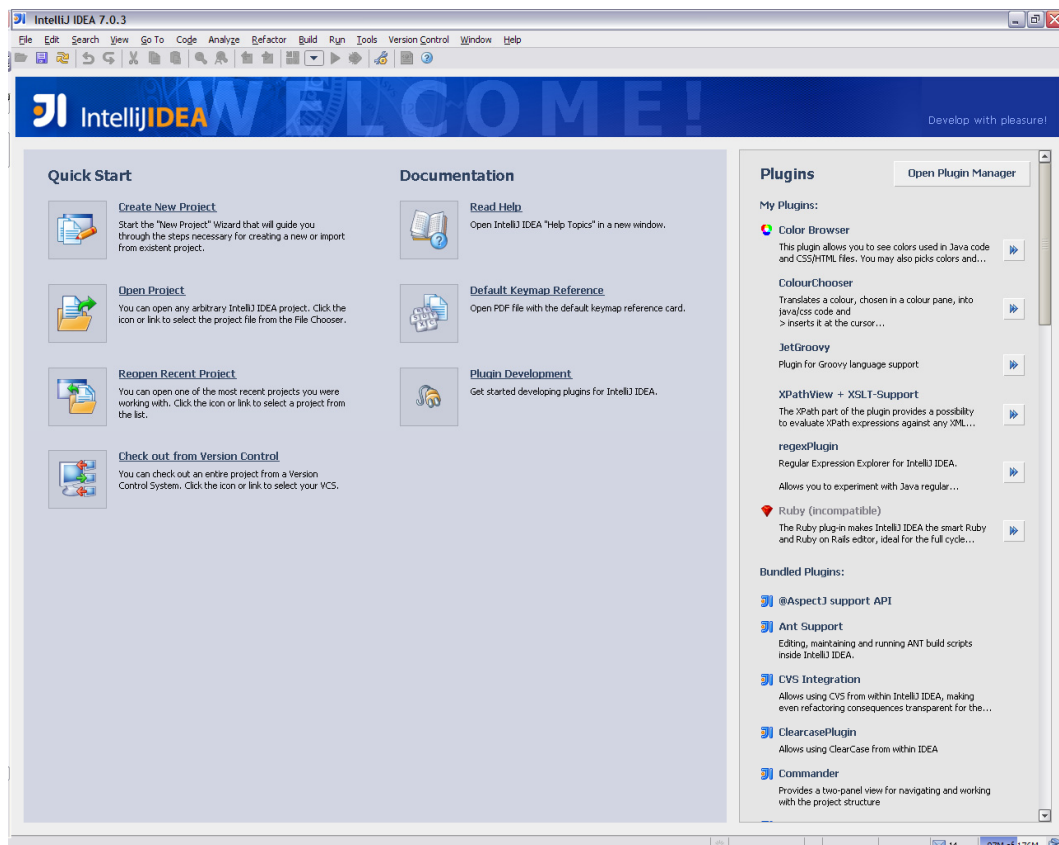
1. IntelliJ IDEA 7.0 or later release build. You can get it from <http://www.jetbrains.com/idea>
2. A compatible application server, for example, JBoss, available at <http://labs.jboss.com/projects/download/>

You may also want to check <http://www.jetbrains.com/idea/training/demos.html> and <http://www.jetbrains.com/idea/documentation/documentation.html> to get better insight into IntelliJ IDEA and its features.

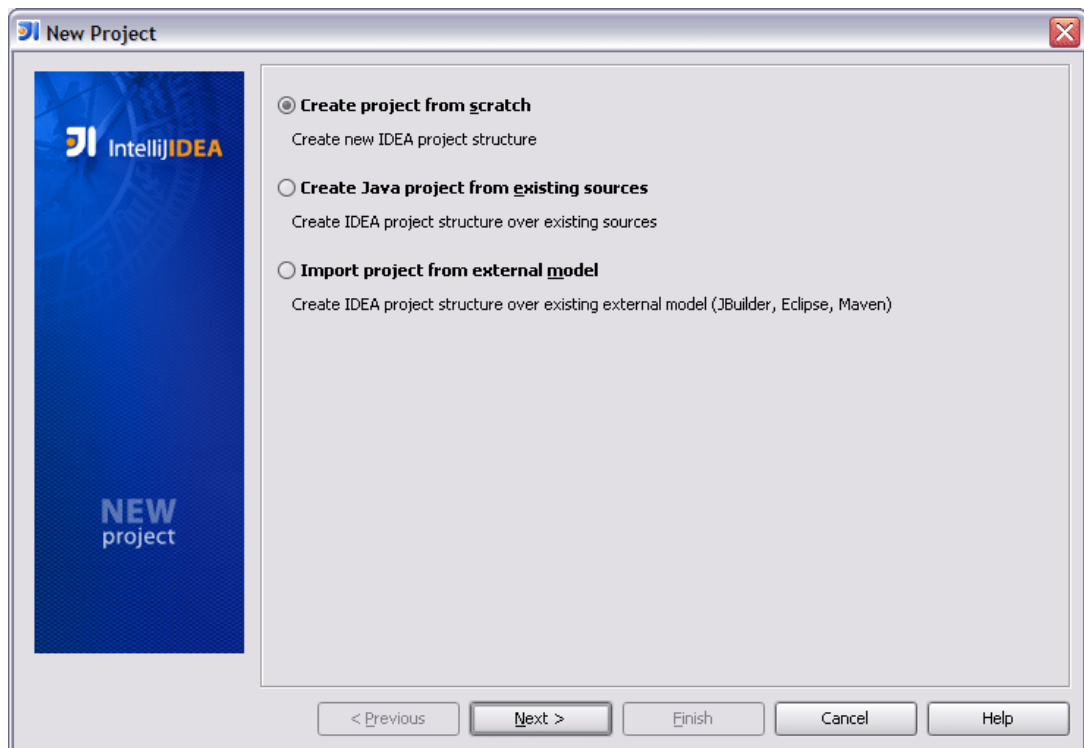
Creating Project

Prior to proceeding with the tutorial steps, make sure that an application server is installed on your machine. After that, launch IntelliJ IDEA and begin with creating a project from scratch.

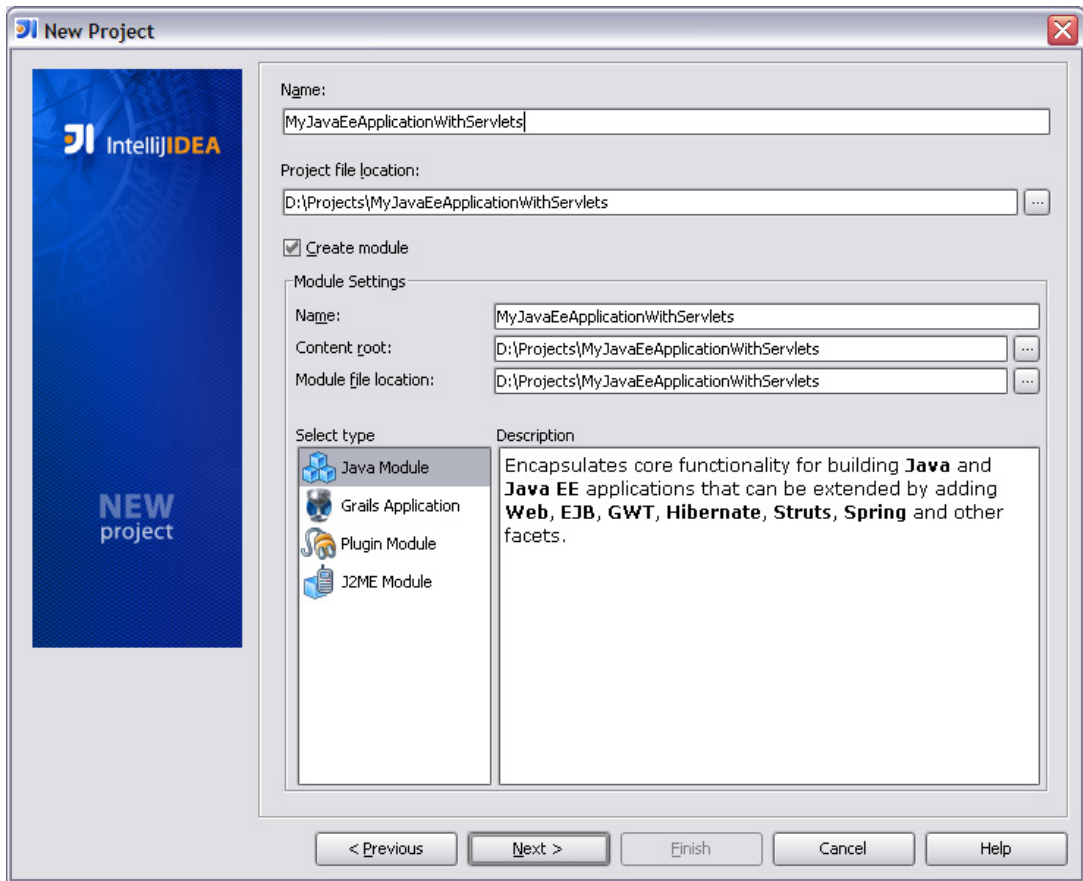
1. Run **IntelliJ IDEA**.



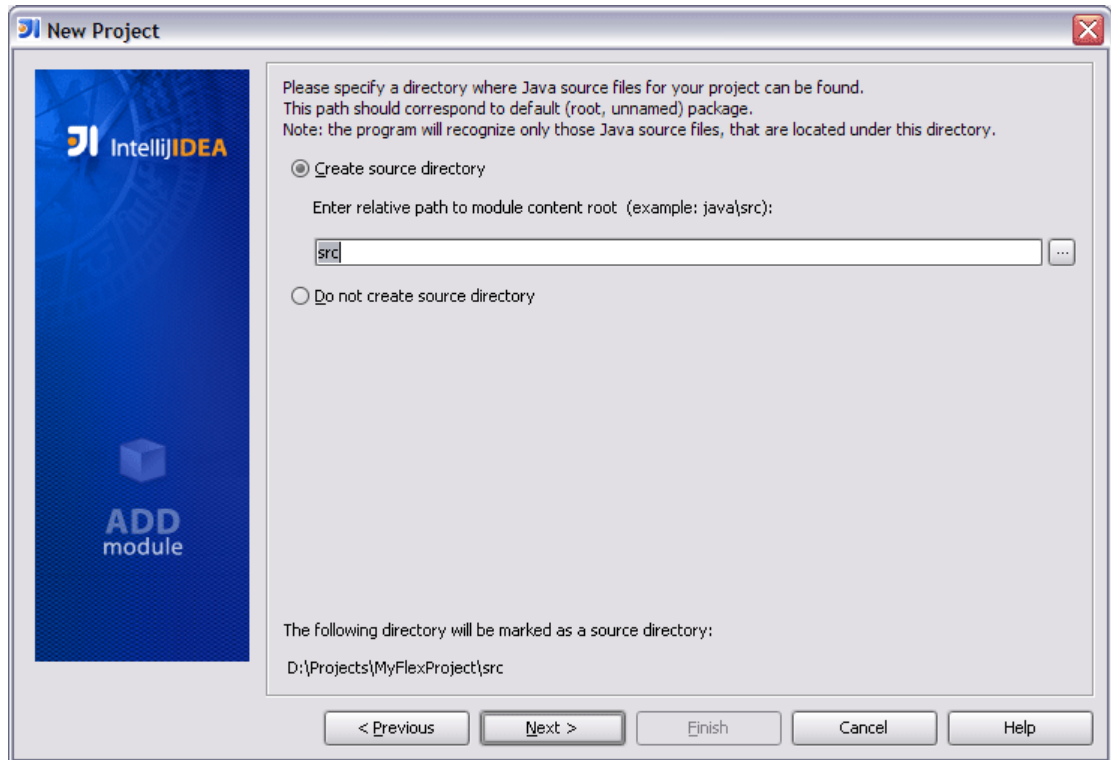
2. On the **Quick Start** page, click **Create New Project**. The **New Project** wizard appears. Click **Next**.



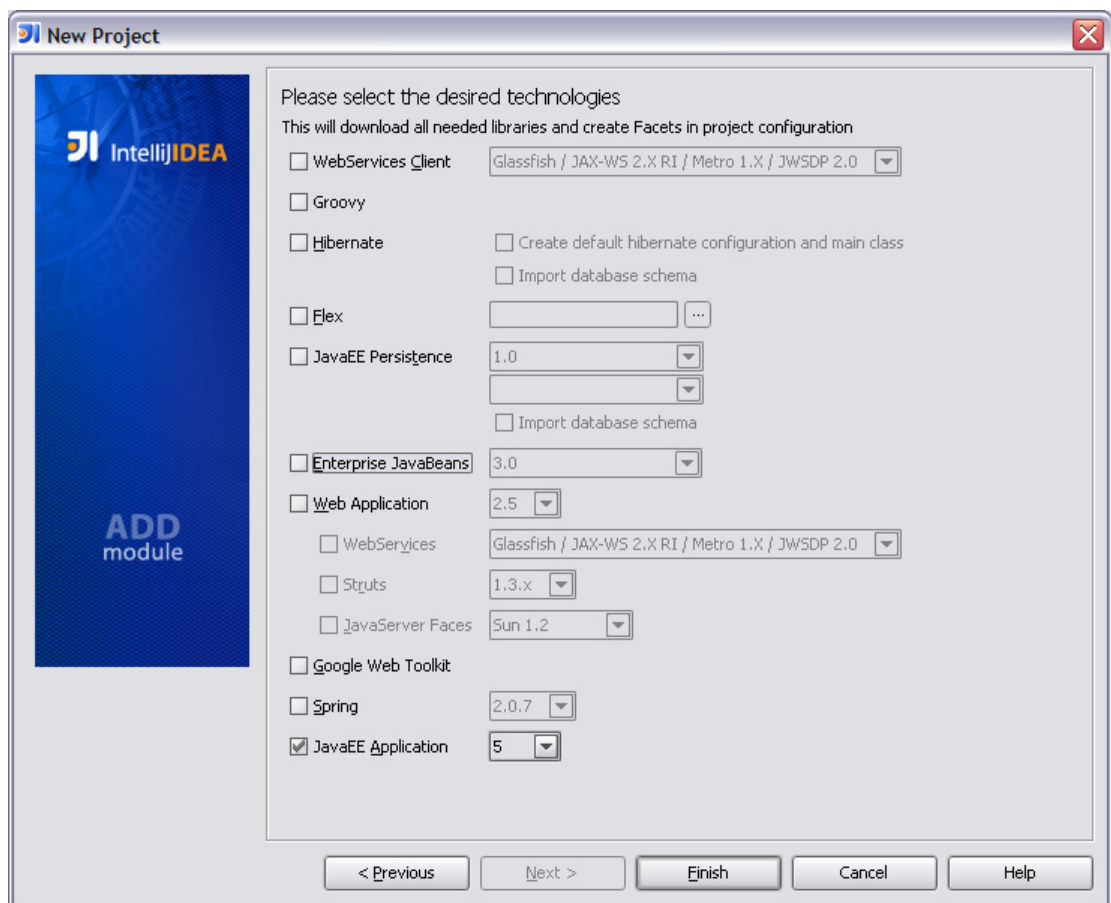
3. Specify the project name, for example, **MyJavaEeApplicationWithServlets**.



4. Leave the option to create source directory selected and click **Next**.



5. From the list of technologies, select **JavaEE Application**. Click **Finish**.



IntelliJ IDEA generates the project with Java EE Application facet that also includes the stub application descriptor, which will be used when deploying the application to a server.

Creating Servlet

Let's start creating the Servlet. First off, we need a Java class that encapsulates the Servlet functionality.

In the **Project** tree, right-click the **src** folder, select **New**, then **Class**. Type the class name – **HelloWorld**, and click **OK**. After IntelliJ IDEA creates the Java file and opens it in the code editor, type the code listed below.

```
public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {

        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        String userName = "";
        Enumeration paramNames = request.getParameterNames();

        if (paramNames != null) {

            while (paramNames.hasMoreElements()) {

                String paramName = (String) paramNames.nextElement();

                if (paramName.equals("userName")) {

                    String[] paramValues =
request.getParameterValues(paramName);

                    if (paramValues != null) {

                        if (paramValues.length > 0) userName =
paramValues[0];

                    }

                }

            }

            userName = (userName.length() > 0) ? userName : "Anonymous";
            Formatter UserNameFormatter = new Formatter();
            String greetingString = "Hello, %s!";

            out.println(UserNameFormatter.format(greetingString,
userName).toString());

        }

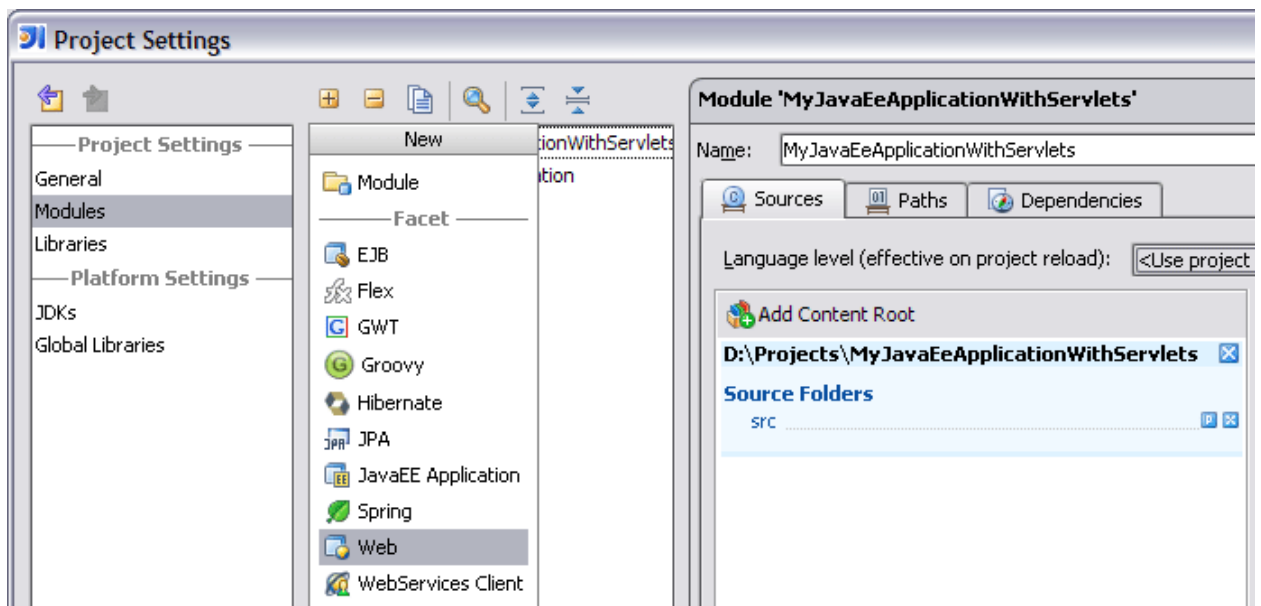
        public void doPost
            (HttpServletRequest
             request,
             HttpServletResponse
             response) throws IOException, ServletException {
            doGet(request, response);
        }

    }
}
```

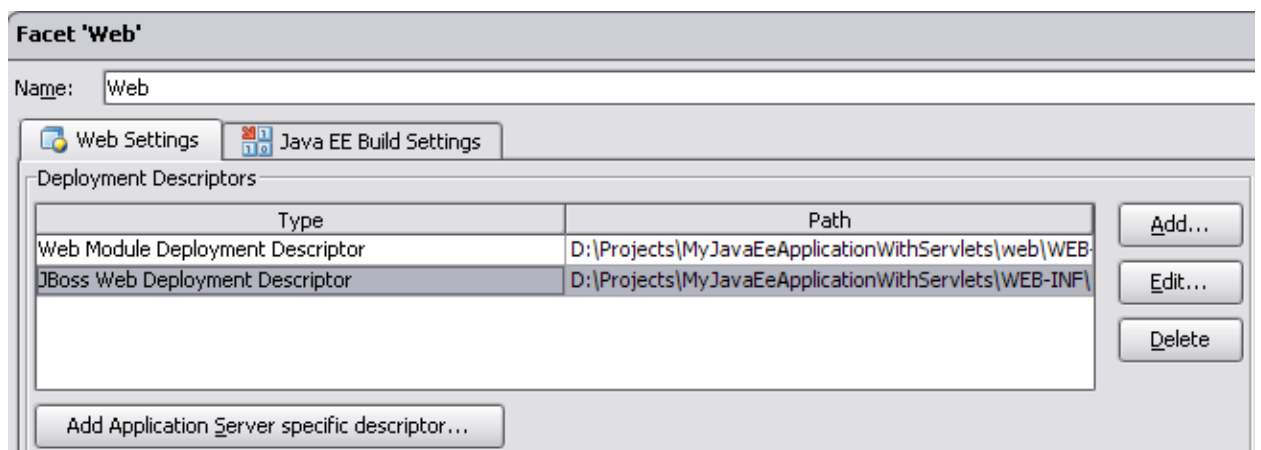
This class extends the **HttpServlet** class and overrides **doGet** and **doPost** methods that are called when Servlet is invoked through **GET** and **POST**, respectively. We're using enumeration through the parameters list instead of directly calling `request.getParameter("userName")` to avoid possible NPE in case there's no such parameter passed to Servlet.

It's recommended that you don't copy and paste the code, but type it live instead – it's really good way of getting experience with advanced IntelliJ IDEA code editing features. The import statements from this code are intentionally omitted to let you see how IntelliJ IDEA automatically detects and imports the required classes and packages.

Now, create a Servlet based on the class we just added. For that, we need to enhance our module with the Web Facet. Press **CTRL+ALT+S**, then **1** to bring up the **Project Settings** dialog. Then, select the **MyJavaEeApplicationWithServlets** module. On the dialog box toolbar, click **+** and select **Web** from the list of Facets. This creates the **Web** Facet, that will host the Servlet and all the required descriptors for the Web part of our Java EE application.

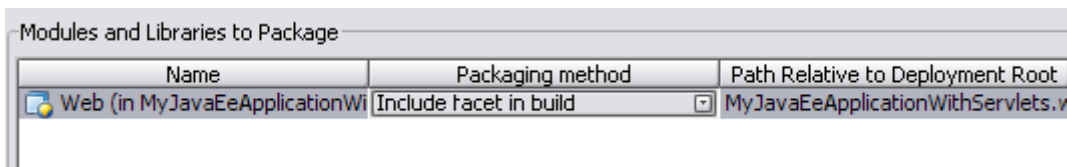


With the **Web** Facet selected, the Project Settings dialog box shows the Facet-specific page. In the Web Settings tab, click **Add Application Server Specific Descriptor**. In the appeared dialog box, select the desired server and its version, and click **OK**. In this case this is **JBoss**.



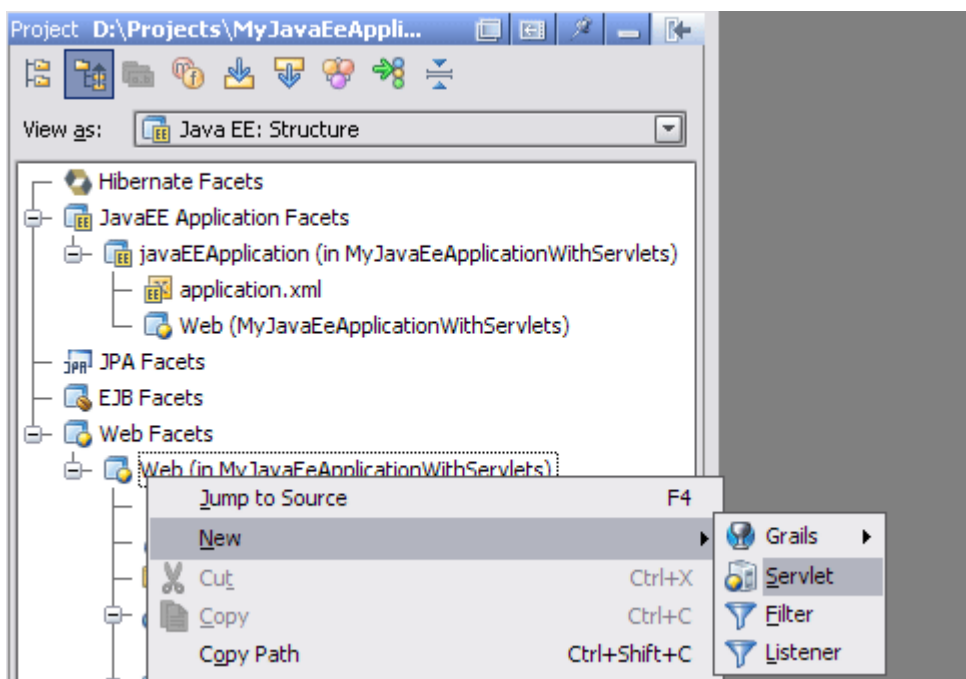
On the **Java EE Build Settings** tab, select **Create web module war file** – we will need this file later for deployment. Note that **Java EE Build settings** vary from the server you're using. In this example, **JBoss** server requires **WAR** file, other servers' requirements may include an exploded directory, or both.

Web Facet configuration is now completed. The last thing left here is to include the Web part in our Java EE application. For that, select **javaEEApplication** Facet, then under **Modules and Libraries to Package**, select packaging method **Include facet in build** for the module **Web (in MyJavaEeApplicationWithServlets)**.

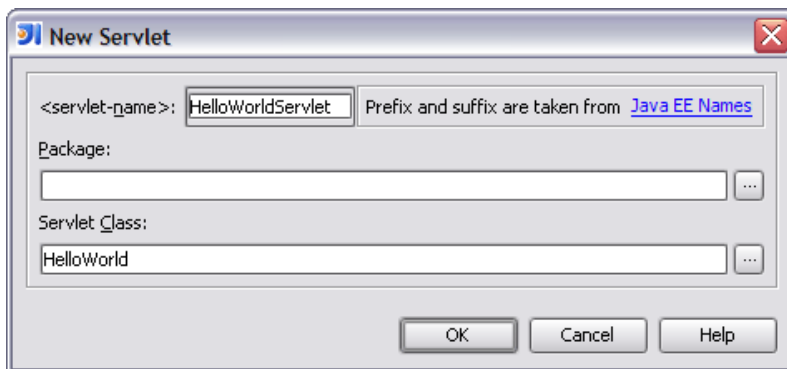


Now, click **OK** to save the project settings and add Servlet itself to the project.

In the Project tool window, switch to **Java EE: Structure** view. Right-click **Web (in MyJavaEeApplicationWithServlets)**, select **New** and then click **Servlet**.



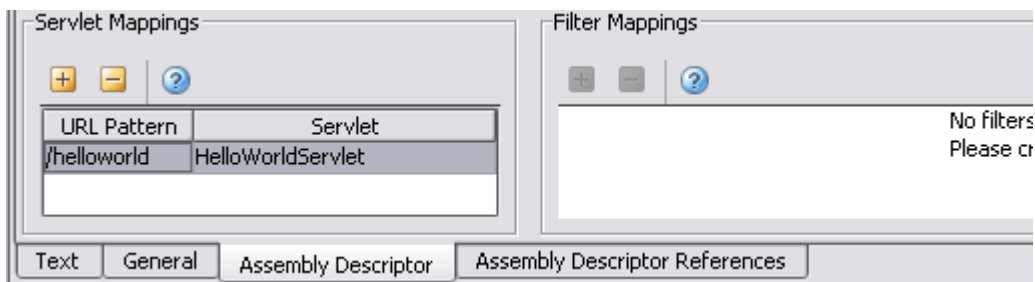
New Servlet dialog box appears.



Type the **Servlet name**, for example, **HelloWorldServlet**. Package can be left empty, click ... near **Servlet class** edit box. IntelliJ IDEA automatically filters the list of appropriate classes you can use here. In this sample project, there will be a single entry – **HelloWorld**.

Servlet is now created and we need to configure its deployment settings so that it'd be accessible when deployed to the server. For that, in the **Project** tool window, expand the **web** folder under **Web (in MyJavaEeApplicationWithServlets)**, expand **WEB-INF** and double-click **web.xml** – the Web deployment descriptor to open it in the editor.

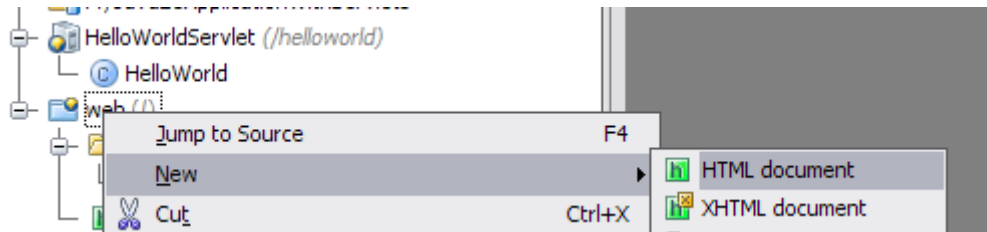
In the bottom of editor, select **Assembly Descriptor** tab. This opens the dedicated visual assembly editor. Click + under **Servlet Mappings**. IntelliJ IDEA adds new mapping for **HelloWorldServlet** (because it's the only Servlet in the project, otherwise there'd be a combo box with the list of all available Servlets). Type the **URL Pattern** – the address that will be used to access the Servlet. Don't forget the leading "/" delimiter. In this example, the pattern is **/helloworld**.



We have just created the Servlet, ready for being deployed and used. Let's make use of it through a simple HTML page.

Invoking Servlet from a Simple HTML Page

To create an HTML page, right-click the **web** folder under **Web (in MyJavaEeApplicationWithServlets)**, select **New** and click **HTML document**. In the appeared dialog box, type **index.html** and click **OK**.



After IntelliJ IDEA creates the blank HTML file and opens it in the editor, type the following code.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

    <title>IntelliJ IDEA Servlet Example</title>

</head>

<body>

<h2>Hello World Using GET</h2>

<form action="/MyJavaEeApplicationWithServletsWeb/helloworld" method="get">
    Type your name:&nbsp;<input type="text" name="userName"
width="100"/>&nbsp;<input type="submit" value="Say Hello"/>
</form>

<h2>Hello World Using POST</h2>

<form action="/MyJavaEeApplicationWithServletsWeb/helloworld" method="post">
    Type your name:&nbsp;<input type="text" name="userName"
width="100"/>&nbsp;<input type="submit" value="Say Hello"/>
</form>

</body>
</html>
```

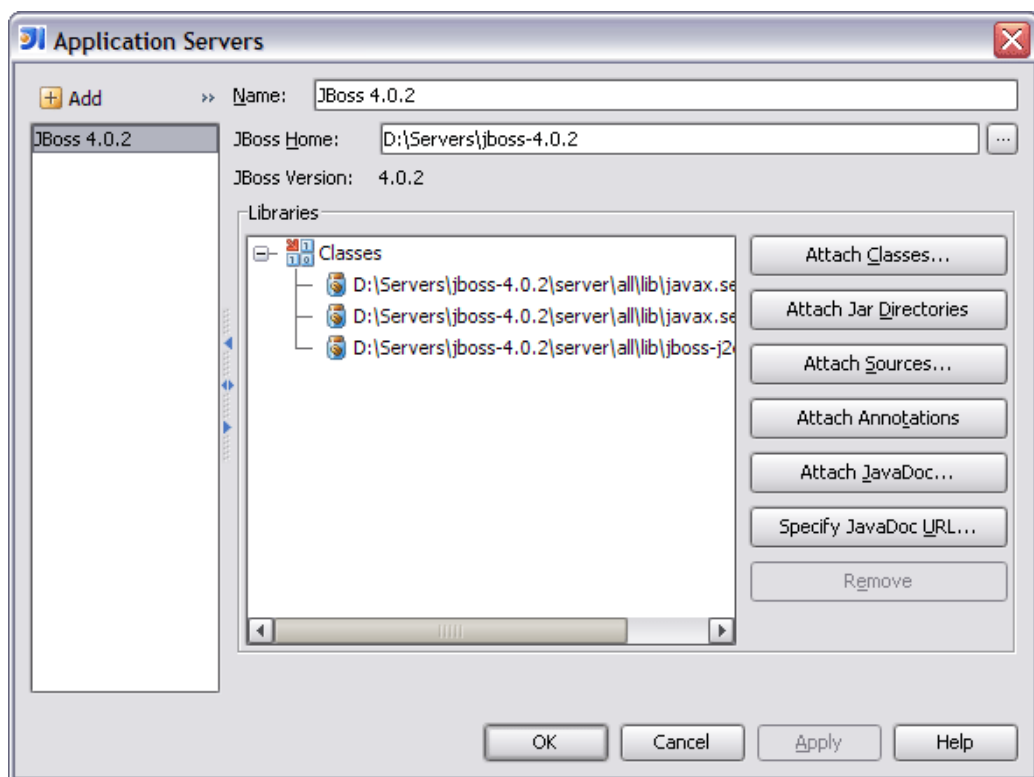
Notice that here we're creating two forms, one for **GET** method and another for **POST**, and that **MyJavaEeApplicationWithServletsWeb** is added to **form action**. This is the application context where the Servlet will be running after deployment. **/helloworld** is the URL to which we mapped Servlet, so these two parts combine the complete URL where the you are redirected after submitting the form.

As with the Java code, it's recommended that you type it instead of pasting to experience the coding assistance available for HTML code: completion for tag and attribute names, their values, etc.

Creating Run Configuration

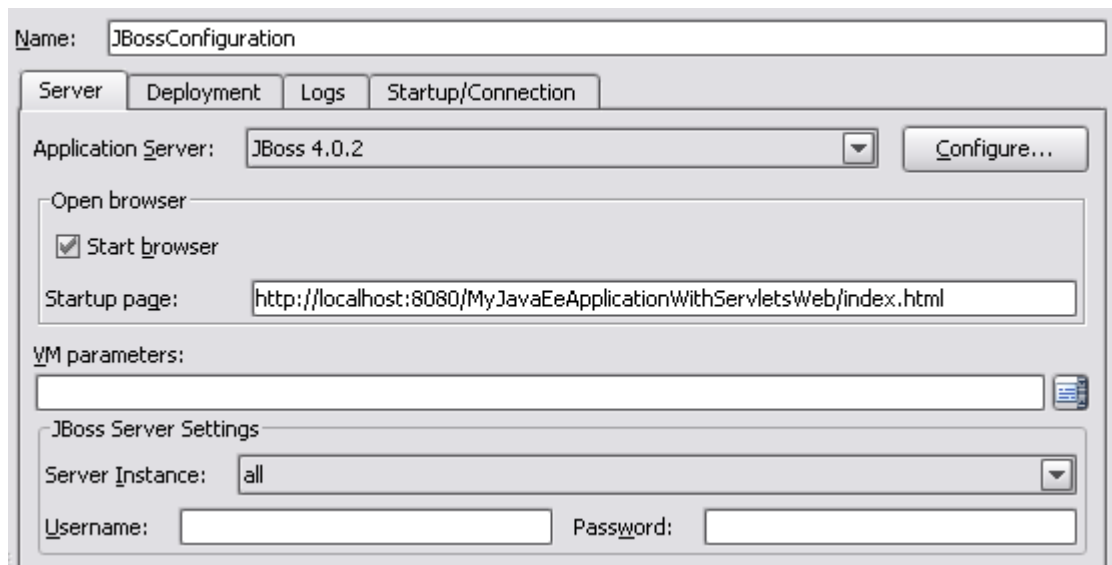
Let's create the run configuration to see our application in action.

1. On the main menu, select **Run** and then click **Edit Configurations**.
2. Click **plus** button to add a configuration. As it's already mentioned, we're using a local installation of JBoss server, so select **JBoss**, then click **Local**.
3. Specify the configuration name, for example, **JBossConfiguration**.
4. In the **Server** tab, click **Configure**. **Application Servers** dialog appears. Click + button to add the server configuration. In the **JBoss home** field specify the folder where you have installed the server.

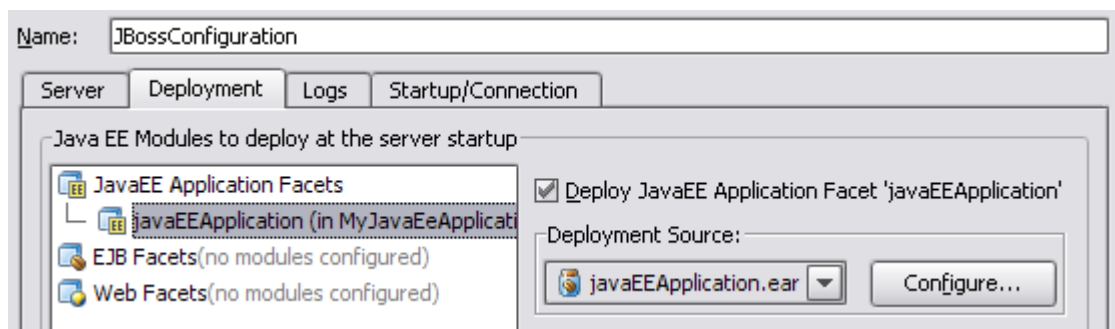


5. Click **OK**. Back in the **Run/Debug Configuration** dialog box, select the configured server from the **Application Server** list.

6. Make sure the **Start browser** option is selected. Also notice that IntelliJ IDEA has automatically added the application context to the **Startup page**. We only need to append the **/index.html** so that the HTML file we created is open right after the application starts.



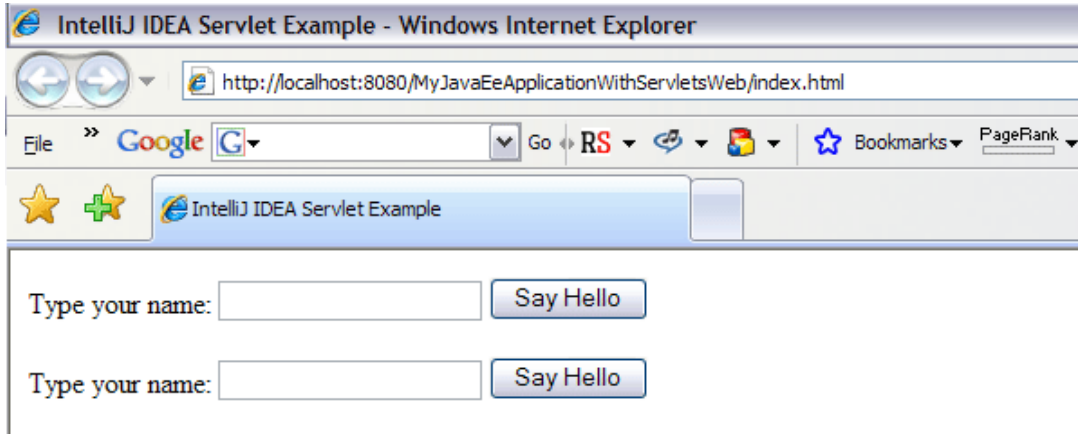
7. From the **Server Instance** list, select **all**.
8. Click **Deployment** tab. Select the **javaEEApplication** under **Java EE Application Facets** and then select **Deploy** option.



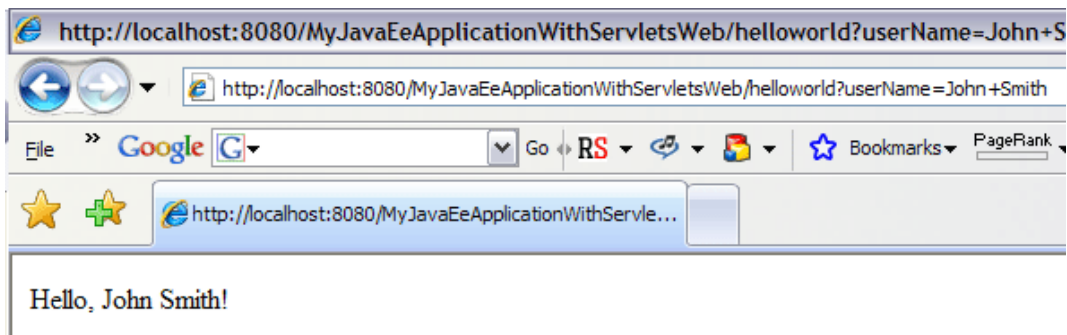
9. Click **OK** to save the configuration.

Run the Application

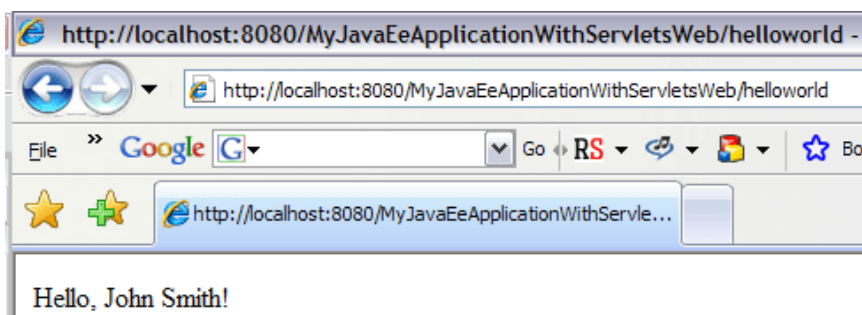
Now everything is ready to run, so just make sure the configuration is selected in the toolbar and press **SHIFT+F10**. IntelliJ IDEA compiles, deploys and runs the application.



Now you can try typing various names and see the response, both with GET...



... and POST methods.



That's all for now. Of course, this tutorial only covers the essential topics you will need to build a Java EE Application with a Servlet, using IntelliJ IDEA.

There's a lot more of things to explore and features that can help you enhance your application: integration with Hibernate, Spring, support for EJB, JSP, AJAX, integration with many popular application servers, and a whole lot more. Check out <http://www.jetbrains.com/idea> for additional details and other tutorials.