Contents

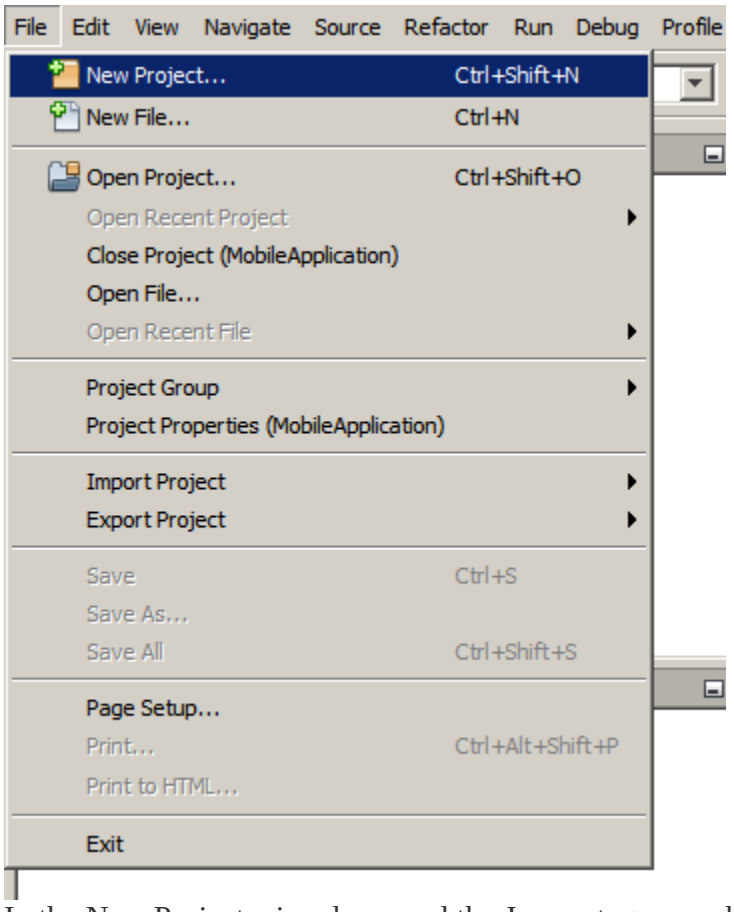
To complete this tutorial, you need the following software and resources.

Software or Resource	Version Required
NetBeans IDE	version 7.2, 7.3, 7.4, or 8.0
Java Development Kit (JDK)	version 6, 7, or 8

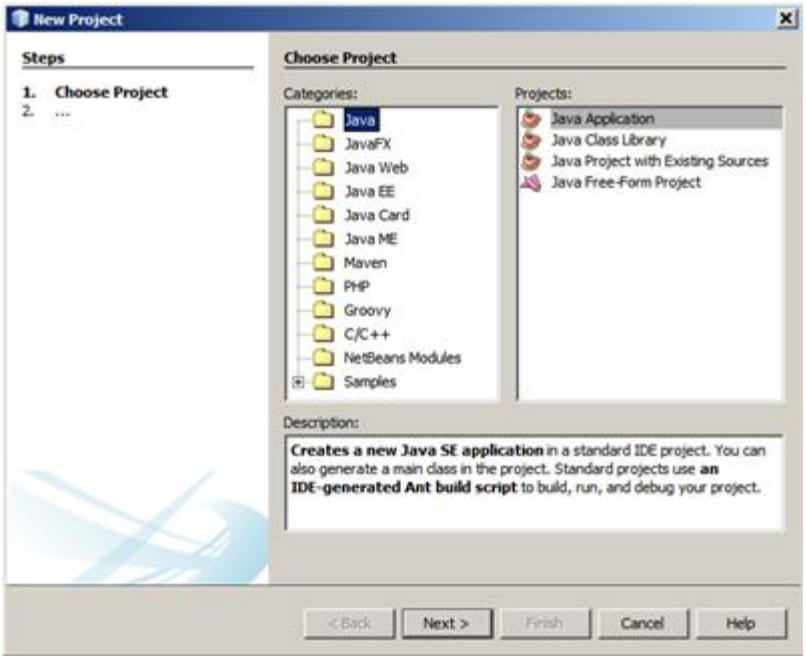
Setting Up the Project

To create an IDE project:

1. Start NetBeans IDE.
2. In the IDE, choose File > New Project, as shown in the figure below.

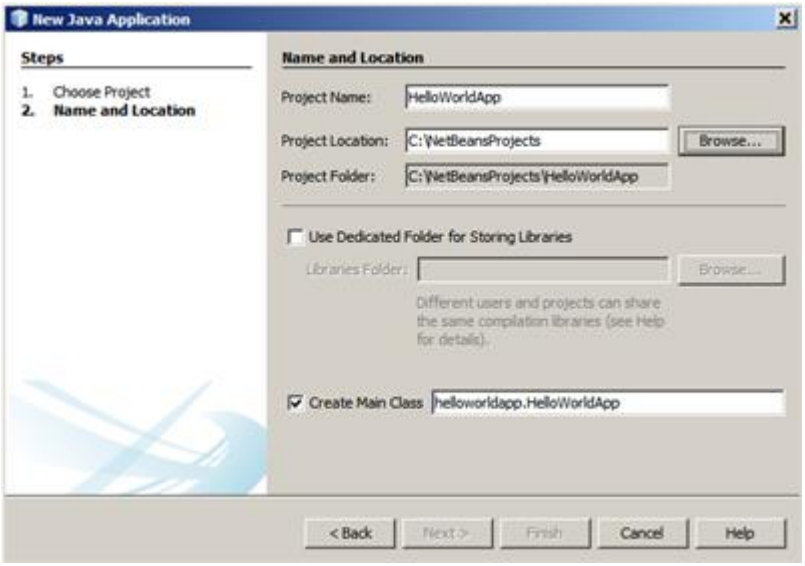


3. In the New Project wizard, expand the Java category and select Java Application as shown in the figure below. Then click Next.



4. In the Name and Location page of the wizard, do the following (as shown in the figure below):

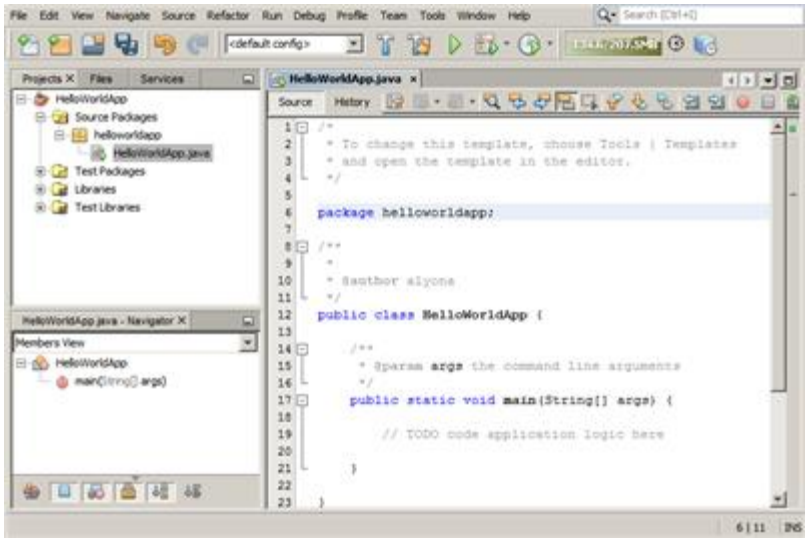
- In the Project Name field, type HelloWorldApp.
- Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
- In the Create Main Class field, type helloworldapp.HelloWorldApp.



5. Click Finish.

The project is created and opened in the IDE. You should see the following components:

- The Projects window, which contains a tree view of the components of the project, including source files, libraries that your code depends on, and so on.
- The Source Editor window with a file called HelloWorldApp open.
- The Navigator window, which you can use to quickly navigate between elements within the selected class.



Adding Code to the Generated Source File

Because you have left the Create Main Class checkbox selected in the New Project wizard, the IDE has created a skeleton main class for you. You can add the "Hello World!" message to the skeleton code by replacing the line:

```
        // TODO code application logic here
```

with the line:

```
        System.out.println("Hello World!");
```

Save the change by choosing File > Save.
The file should look something like the following code sample.

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package helloworldapp;

/**
 *
 * @author <your name>
 */
public class HelloWorldApp {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}
```

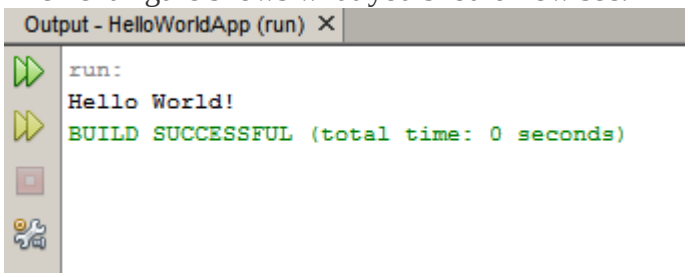
Compiling and Running the Program

Because of the IDE's Compile on Save feature, you do not have to manually compile your project in order to run it in the IDE. When you save a Java source file, the IDE automatically compiles it.
The Compile on Save feature can be turned off in the Project Properties window. Right-click your project, select Properties. In the Properties window, choose the Compiling tab. The Compile on Save checkbox is right at the top. Note that in the Project Properties window you can configure numerous settings for your project: project libraries, packaging, building, running, etc.

To run the program:

- Choose Run > Run Project.

The next figure shows what you should now see.



Congratulations! Your program works!
If there are compilation errors, they are marked with red glyphs in the left and right margins of the Source Editor. The glyphs in the left margin indicate errors for the corresponding lines. The glyphs in the right margin show all of the areas of the file that have errors, including errors in lines that are not visible. You can mouse over an error mark to get a description of the error. You can click a glyph in the right margin to jump to the line with the error.

Building and Deploying the Application

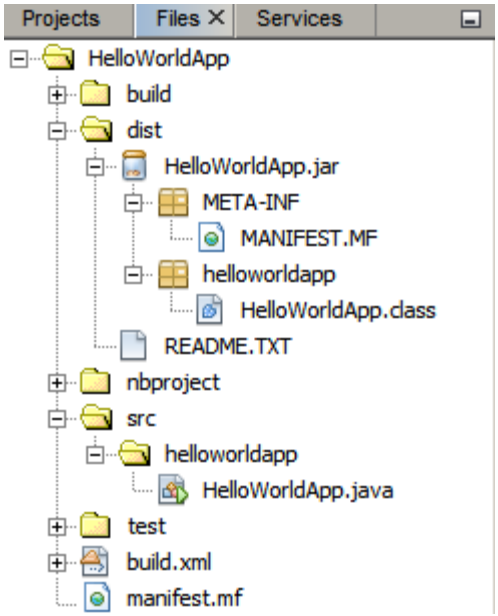
Once you have written and test run your application, you can use the Clean and Build command to build your application for deployment. When you use the Clean and Build command, the IDE runs a build script that performs the following tasks:

- Deletes any previously compiled files and other build outputs.
- Recompiles the application and builds a JAR file containing the compiled files.

To build your application:

- Choose Run > Clean and Build Project.

You can view the build outputs by opening the Files window and expanding the HelloWorldApp node. The compiled bytecode file HelloWorldApp.class is within the build/classes/helloworldapp subnode. A deployable JAR file that contains the HelloWorldApp.class is within the dist node.


Projects
Files
Services
HelloWorldApp
build
dist
HelloWorldApp.jar
META-INF
MANIFEST.MF
helloworldapp
HelloWorldApp.class
README.TXT
nbproject
src
helloworldapp
HelloWorldApp.java
test
build.xml
manifest.mf