Packaging and Distributing Java Desktop Applications

One question that a lot of beginning programmers have is: "Now that I've created my application in the IDE, how do I get it to work from the command line outside of the IDE." Similarly, someone might ask, "How do I distribute this application to other users without having to give them the whole IDE as well?"

The answers to these questions are relatively simple, but not necessarily obvious. This document addresses those questions by taking you through the basics of using the IDE to prepare your applications for distribution and deployment. In addition, this document provides information that you might need to configure your system (or which you might need to pass on to the users of your application). We will show a few different approaches for deploying an application, so that users can access the application by:

- Double-clicking the application's Java Archive (JAR) file.
- Calling the application from the command line.
- Calling the application from a script file.
- Using Java Web Start.

Contents

REQUIRES NetBeans 7.0/7.1/7.2

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

Software or Resource	Version Required
NetBeans IDE	version 6.9, 7.0, 7.1, or 7.2
Java Development Kit (JDK)	version 6 or 7
Deployment Tutorial source files	

Creating Executable JAR File

This part of the tutorial shows how you can create a distributable application in the IDE and then run that application from outside of the IDE. We will package the application in the form of an executable JAR file.

A JAR file is an archive file that can contain multiple files and folders. JAR files are similar to zip files, but JAR files can have additional attributes that are useful for distributing Java applications. These attributes include digitally signing JAR files, additional compression, multiplatform compatibility, etc.

In this exercise, you create an IDE project and then place two pre-written Java source files into that project. Then you will compile the classes and build an executable JAR file. Afterwards, you will learn how to run the JAR file from outside of the IDE.

The classes used in this tutorial implement features of the GNU grep utility, which can be used for searching text or regular expression patterns inside text files. The project contains both command-line and GUI versions of the application, so that you can see different ways of running the application.

Creating a Project with Existing Sources

Download NetBeans IDE

Training

Java Programming Language

Support

Oracle Development Tools Support Offering for NetBeans IDE

Documentation

General Java Development
External Tools and Services
Java GUI Applications
Java EE & Java Web Development
Web Services Applications
NetBeans Platform (RCP) and Module Development
PHP and HTML5 Applications
C/C++ Applications
Mobile Applications

Sample Applications
Demos and Screencasts

More

FAQs
Contribute Documentation!
Docs for Earlier Releases

1. Download the DeploymentTutorial.zip file and extract its contents on your system.
   This zip archive contains source files for the application plus a few other files that will be used in the tutorial.
2. In NetBeans IDE, choose File > New Project.
3. In the Choose Category page, select Java Project With Existing Sources in the Java category and click Next.
4. On the Name and Location page of the wizard, type `AnotherGrep` as the project name and specify the project's location. Leave the Set as Main Project checkbox selected and click Next.
   The project folder does *not* have to be in the same location as the source files that you are importing into the project.
5. On the Existing Sources page of the wizard, specify the sources that will be in the project.
   Click the Add Folder button that is to the right of the Source Package Folders field. Navigate to the `DeploymentTutorial` folder that you have just unzipped on your system, expand the folder, select the `src` folder, and click Open. The `src` folder is added to your Source Package Folders field.
6. Click Finish.
   Note: If, for example, you want to exclude some source files from importing into the project, click Next to open the last Includes & Excludes window. In our case, we want to use all the source files in the `src` folder, so we click Finish to finish working in the New Project wizard.

The project opens in the IDE and becomes visibile in the Projects window. You can explore the contents of the project by expanding the project's Source Packages node, where you should see classes called `Grep` and `xGrep`. `Grep.java` is a console version of the application. `xGrep.java` is a GUI version of the application and uses methods defined in `Grep.java`.

Configuring the Project

There are a few configuration steps you need to do, such as:

- Choose the Java platform that will be used to compile the sources.
- Set the project's main class. By doing this, you ensure that the JAR file that you create when you build the project is executable.

Verifying the Java Platform

Our project needs to be compiled and run on Java 6 platform. Therefore, you need to make sure that Java 6 is used as the platform for this project.

1. Right-click the project's node and choose Properties.
2. On the Libraries tab, ensure that the Java Platform is JDK 6.
3. On the Sources tab, choose JDK 6 in the Source/Binary format.
4. Click OK to close the Properties window.

Setting the Main Class

In order for a user to easily run your JAR file (by double-clicking the JAR file or by typing `java -jar AnotherGrep.jar` at the command line), a main class has to be specified inside the JAR's *manifest* file. (The manifest is a standard part of the JAR file that contains information about the JAR file that is useful for the `java` launcher when you want to run the application.) The main class serves as an entry point from which the `java` launcher runs your application.

When you build a project, the IDE builds the JAR file and includes a manifest. When you set the project's main class, you ensure that the main class is be designated in the manifest.

To set the project's main class:

1. Right-click the project's node and choose Properties.
2. Select the Run panel and enter `anothergrep.xGrep` in the Main Class field.
3. Click OK to close the Project Properties dialog box.

When you build the project later in this tutorial, the manifest will be generated and include the following entry:

```
Main-Class: anothergrep.xGRep
```

Building the Project and Creating the JAR File

Now that you have your sources ready and your project configured, it is time to build your project.

To build the project:

- Choose Run > Build Main Project.
  Alternatively, right-click the project's node in the Projects window and choose Build.

When you build your project:

- `build` and `dist` folders are added to your project folder (hereafter referred to as the *PROJECT_HOME* folder).

- All of the sources are compiled into `.class` files, which are placed into the *PROJECT_HOME*/`build` folder.
- A JAR file containing your project is created inside the *PROJECT_HOME*/`dist` folder.
- If you have specified any libraries for the project (in addition to the JDK), a `lib` folder is created in the `dist` folder. The libraries are copied into `dist/lib`.
- The manifest file in the JAR is updated to include entries that designate main class and any libraries that are on the project's classpath.

Note: You can view the contents of the manifest in the IDE's Files window. After you have built your project, switch to the Files window and navigate to `dist/AnotherGrep.jar`. Expand the node for the JAR file, expand the `META-INF` folder, and double-click `MANIFEST.MF` to display the manifest in the Source Editor.

```
Main-Class: anothergrep.xGrep
```

(To find more about manifest files, you can read this chapter from the Java Tutorial.)

Running and Distributing the JAR File

Running the Application Inside of the IDE

When developing applications in the IDE, typically you will need to test and refine them before distributing. You can easily test an application that you are working on by running the application from the IDE.

To run the `AnotherGrep` project in the IDE, right-click the project's node in the Projects window and choose Run.

The xGrep window should open. You can click the Browse button to choose a file in which to search for a text pattern. In the Search Pattern field, type text or a regular expression pattern that you would like to match, and click Search. The results of each match will appear in the xGrep window's Output area.

Information on regular expressions that you can use in this application are available here and in many other places.

Running the Application Outside of the IDE

Once you have finished developing the application and before you distribute it, you will probably want to make sure that the application also works outside of the IDE.

You can run the application outside of the IDE by following these steps:

- In your system's file manager (for example, in the My Computer window on Windows XP systems), navigate to *PROJECT_HOME*/`dist` and double-click the `AnotherGrep.jar` file.

You will know that the application has started successfully when the xGrep window opens.

If the xGrep window does not open, your system probably does not have a file association between JAR files and the Java Runtime Environment. See Troubleshooting JAR File Associations below.

Distributing the Application to Other Users

Now that you have verified that the application works outside of the IDE, you are ready to distribute it.

- Send the application's JAR file to the people who will use the application. The users of your application should be able to run it by double-clicking the JAR file. If this does not work for them, show them the information in the Troubleshooting JAR File Associations section below.

Note: If your application depends on additional libraries other than those included in JDK, you need to also include them in your distribution (not the case in our example). The relative paths to these libraries are added in the `classpath` entry of the JAR's manifest file when you are developing your applicaiton in the IDE. If these additional libraries will not be found at the specified classpath (i.e., relative path) at launch, the application will not start.
Create a zip archive that contains the application JAR file and the library and provide this zip file to users. Instruct the users to unpack the zip file making sure that the JAR file and libraries JAR files are in the same folder. Run the application JAR file.

Starting Your Java Application

The goal of this exercise is to show you some ways that you can start your application from the command line.

This exercise shows you how you can start a Java application in the following two ways:

- Running the `java` command from the command line.
- Using a script to a call a class in the JAR file.

Launching Applications From the Command Line

You can launch an application from the command line by using the `java` command. If you want to run an executable JAR file, use the `-jar` option of the command.

For example, to run the AnotherGrep application, you would take the following steps:

1. Open a terminal window. On Microsoft Windows systems, you do this by choosing Start > Run, typing `cmd` in the Open field, and clicking OK.
2. Change directories to the *PROJECT_HOME*/`dist` folder (using the `cd` command).
3. Type the following line to run the application's main class:

   ```
   java -jar AnotherGrep.jar
   ```

If you follow these steps and the application does not run, you probably need to do one of the following things:

- Include the full path to the `java` binary in the third step of the procedure. For example, you would type something like the following, depending on where your JDK or JRE is located:

  ```
  C:\Program Files\Java\jdk1.6.0_23\bin\java -jar AnotherGrep.jar
  ```

- Add the Java binaries to your PATH environment variable, so that you never have to specify the path to the `java` binary from the command line. See Setting the PATH Environment Variable.

Launching Applications From a Script

If the application that you want to distribute is a console application, you might find that it is convenient to start the application from a a script, particularly if the application takes long and complex arguments to run. In this section, you will use a console version of the Grep program, where you need to pass the arguments (search pattern and file list) to the JAR file, which will be invoked in our script. To reduce typing at the command line, you will use a simple script suitable to run the test application.

First you need to change the main class in the application to be the console version of the class and rebuild the JAR file:

1. In the IDE's Projects window, right-click the project's node (`AnotherGrep`) and choose Properties.
2. Select the Run node and change the Main Class property to `anothergrep.Grep` (from `anothergrep.xGrep`). Click OK to close the Project Properties window.
3. Right-click the project's node again and choose Clean and Build Project.

After completing these steps, the JAR file is rebuilt, and the `Main-Class` attribute of the JAR file's manifest is changed to point to `anothergrep.Grep`.

BASH script -- for UNIX and Linux machines

Inside the folder on your system where you extracted the contents of the DeploymentTutorial.zip file, there is a `grep.sh` bash script. Have a look at it:

```
#!/bin/bash
                    java -jar dist/AnotherGrep.jar $@
```

The first line states which shell should be used to interpret this. The second one executes your JAR file, created by the IDE inside *PROJECT_HOME*/`dist` folder. `$@` just copies all given arguments, enclosing each inside quotes.

This script presumes that the Java binaries are part of your PATH environment variable. If the script does not work for you, see Setting the PATH Environment Variable.

More about bash scripting can be found here.

.bat script for Windows machines

On Microsoft Windows systems, you can only pass nine arguments at once to a batch file. If there were more than nine arguments, you would need to execute the JAR file multiple times.

A script handling this might look like the following:

```
@echo off
set jarpath="dist/AnotherGrep.jar"
set pattern="%1"
shift
:loop
  if "%1" == "" goto :allprocessed
  set files=%1 %2 %3 %4 %5 %6 %7 %8 %9
  java -jar %jarpath% %pattern% %files%
  for %%i in (0 1 2 3 4 5 6 7 8) do shift
goto :loop

:allprocessed
```

This script is included as `grep.bat` inside the folder on your system where you extracted the contents of the DeploymentTutorial.zip file so you can try it out.

The nine arguments are represented inside the batch file by `%<ARG_NUMBER>`, where `<ARG_NUMBER>` has to be inside `<0-9>`. `%0` is reserved for the script name.

You can see that only nine arguments are passed to the program at a time (in one loop). The `for` statement just shifts the arguments by nine, to prepare it for next loop. Once an empty file argument is detected by the `if` statement (there are no further files to process), the loop is ended.

More about batch scripting can be found on this page.

Packaging the Application for Java Web Start

Java Web Start is a technology that is used to run Java applications from a web browser with a single click. For detailed information on packaging applications for deployment through Java Web Start, see Enabling Java Web Start in the NetBeans IDE. Here we provide only quick steps you need to follow to make you application deployable by using Java Web Start.

1. Right-click the project's node in the Projects window and choose Properties.
2. On the Web Start tab of the Project Properties window, select the Enable Web Start checkbox.
3. Choose Local Execution from the Codebase drop-down list (as we test only the local execution).
   Leave all other settings at their default values and click OK.
4. Right-click the project's node and choose Clean and Build.
   This IDE command deletes all previously compiled files and build outputs, recompiles your application, and builds the project with new settings.
5. Outside of the IDE, open the `*PROJECT_HOME*/dist` folder and open the `launch.html` file in your browser.
   The test HTML page with the Launch button opens.
6. Click the Launch button to open the application.
   You can see that Java is loaded and the application starts.

   Note: Some browsers redirect you to Java download page first.

Troubleshooting Tips

Specifying JAR File Associations

On most systems, you can execute an executable JAR file by simply double-clicking the JAR file. If nothing happens when you double-click the JAR file, it might be because of either of the following two reasons:

- The JAR file type is probably not associated with a Java Runtime Environment (JRE) on that system.
  If the JAR file type is associated with a JRE, the icon that represents that file should include a Java logo.
- The JAR file type is associated with the JRE, but the `-jar` option is not included in the command that is passed to the JRE when you double-click the icon.

Note: Sometimes JAR file associations are switched by software that you install, such as software to handle zip files.

The way how you associate the JAR file type with the `java` launcher depends on your operating system.

Make sure that there is a version of the JRE installed on your system. You should use version 1.4.2 or later. You cannot launch a Java application if no Java is installed. (If you have the JDK installed, you also get the JRE. However, if you are distributing the program to a non-programmer, that person does not necessarily have either the JRE or the JDK.)

- On Windows XP, you can check for installed versions of Java by choosing Start > Control Panel > Add or Remove Software (you should see, for example, Java(TM) 6 Update 33).
- On Windows Vista or 7, you can check for installed versions of Java by choosing Start > Control Panel > Programs and Components (you should see, for example, Java(TM) 6 Update 33).

If there is no Java on the system, you can get the JRE one from the Java SE download site.

If you have Java installed on your system, but the file association is not working, continue with the steps for adding the JAR file association on Microsoft Windows:

1. Choose Start > Control Panel.
2. (Applicable to Windows Vista only.) Click Control Panel Home > Programs.
3. For Windows XP, double-click Folder Options and select the File Types tab.
   For Windows Vista or 7, click Default Programs and select Associate a file type or protocol with a program.
4. In the Registered File Types list, select JAR File.
5. (On Windows XP, in the Details section of the dialog box), click Change Program.
6. In the Open With dialog box, select Java Platform SE Binary.
7. Click OK to exit the Open With dialog box.

8. Click Close to exit the Folder Options dialog box (on Windows XP) or the Associate a file type or protocol with a specific program dialog box (on Windows 7).

Note: If JAR files are associated with the Java Platform SE Binary on your system but double-clicking still does not execute the file JAR file, you might need to specify the `-jar` option in the file association.

To specify the `-jar` option in the file association on Microsoft Windows XP:

1. Choose Start > Control Panel.
2. For Windows XP, double-click Folder Options and select the File Types tab.
3. In the Registered File Types list, select JAR File.
4. In the Details section of the dialog box, click Advanced.
5. In the Edit File Type dialog box, click Edit.
6. In the Application Used to Perform Action text field, add the following at the end of the path to the JRE:

   ```
   -jar "%1" %*
   ```

   Afterwards, the field should contain text similar to the following:

   ```
   "C:\Program Files\Java\jre1.6.0_33\bin\javaw.exe" -jar "%1" %*
   ```

7. Click OK to exit the Editing Action for Type dialog box.
8. Click OK to exit the Edit File Type dialog box.
9. Click Close to exit the Folder Options dialog box.

Note: Starting with Windows Vista advanced file associations can be set via RegEdit. See the What Happened to the File Types Dialog? article for details.

For UNIX and Linux systems, the procedure for changing file associations depends on which desktop environment (such as GNOME or KDE) that you are using. Look in your desktop environment's preference settings or consult the documentation for the desktop environment.

Setting the PATH Environment Variable

If you can not run a Java class or JAR file on your system without pointing to the location of the JDK or JRE on your system, you might need to modify the value of your system's `PATH` variable.
If you are running on a Microsoft Windows system, the procedure for setting the PATH variable depends the version of Windows you are using.

The following are the steps for setting the `PATH` variable on a Windows XP system:

1. Choose Start > Control Panel and double-click System.
2. In the System Properties dialog box, click the Advanced tab.
3. Click the Environment Variables tab.
4. In the list of user variables, select `PATH` and click Edit.
5. Add the location of the JRE to the end of the list of paths. The locations in this list are separated by semicolons (;).
   For example, if your JRE is located at `C:\Program Files\Java\jdk1.6.0_23` you would add the following to the end of the PATH variable:

   ```
   C:\Program Files\Java\jdk1.6.0_23\bin
   ```

6. Click OK to exit the Environment Variables dialog box, and click OK to exit the System Properties dialog box.

If you are running on a UNIX or Linux system, the instructions for modifying your PATH variable depends on the shell program you are using. Consult the documentation of the shell that you are using for more information.

Send Feedback on This Tutorial

Next Steps

For more information on working with NetBeans IDE, see the Support and Docs page on the NetBeans website.

To learn more about the IDE workflow for developing Java applications, including classpath management, see Developing General Java Applications.

SiteMap About Us Contact Legal & Licences