

QuickStart Tool

Eclipse

By **Bill Dudney**

Getting a good start in any new technology or programming language often depends on finding the best available information. The Builder.com QuickStart Tools give you the information you need to quickly grasp the fundamentals of developing in a new IDE, using a new programming language, or working with a new development tool.

Besides explaining the basics, the Builder.com QuickStart Tools show common tasks, expose strengths and weaknesses, demonstrate some of the best uses of the technology, and list a variety of other online and offline resources that can help you build a solid foundation of practical knowledge.

Table of Contents

Fundamentals.....3

Creating and running a Java project5

Strengths7

Weaknesses7

Online resources8

Other resources8

Additional articles8

About Builder.com.....9

Fundamentals

Eclipse.org was founded in November 2001, principally by IBM in the form of a donation of software valued at \$40 million. The organization is led by a “board of stewards,” which is responsible for setting the overall direction of the Eclipse platform. The main focus of Eclipse, in contrast to other open source consortiums, is to build a robust platform for creating commercial tools. Many of the tool vendors that are part of the Eclipse board of stewards have commercial products based on Eclipse (IBM’s WSAD and Metanology’s MDE for J2EE, for example). Although other open source communities have no problem with commercial entities using their software, they were not formed with this as a main goal, as Eclipse.org was.

The question then becomes, What does this mean to us as consumers of Eclipse? The biggest gain we get is a managed release cycle that more or less meets the advertised dates. Many of the contributors to Eclipse get to work on the code as a full-time job instead of having to make time outside of normal working hours.

In addition, because many of the members have products that depend on a robust infrastructure, they are financially motivated to have their employees work on Eclipse. All in all, this is a very beneficial setup for consumers of Eclipse—not only is there a great group of volunteers, but there is also a lot of financial clout behind the ongoing development of the platform.

Getting Eclipse

Let’s get started using Eclipse to develop Java code. The first step, of course, is to download and install the default IDE. You will find all of the currently available versions of Eclipse on the Eclipse download page. Once you have selected a build from the main download page, you will be taken to that build’s specific download page. This page provides many different options for related packages to download. The typical option is to download the Eclipse SDK. The SDK is the most common way to use Eclipse; this package has the Java Development Tools (JDT) and the Eclipse Plug-in Development Environment (PDE) plug-ins already loaded.

The download options are listed in a table. The first column indicates the status of the build (the definition of the status depends on which version you are downloading, but typically a green check is good and a red X is bad). The second column is the platform/OS for which the build is intended. The third column lets you select between using HTTP or FTP to download. The final column is the name of the file that you will download.

Installing and running Eclipse

Now that you have your desired version of Eclipse, install it by unzipping the download into the directory of your choice. In Windows, the typical install is into `C:\DevTools\Eclipse-version`, and on the Macintosh, it is into `/Users/Shared/Application/Eclipse-version`. You can, of course, install it in any directory you’d like.

In Windows, you need to have a `java.exe` somewhere in your system path for Eclipse to start up properly. If you don’t, Eclipse will fail to start, and a message will appear stating that it can’t find a JRE to run within. If that happens, just click OK and add the JRE and JDK bin directories to your system path.

Now that the IDE is installed, let’s take a quick look at some platform-specific options for adjusting the runtime of Eclipse.

Platform	Process
Windows	Create a shortcut for <code>eclipse.exe</code> , select Properties from the context menu (usually a right-click of your mouse), and add the command line arguments to the end of the Target: text field. If you are going to change the location of the workspace directory, specify the Start In: directory as the parent directory of your workspace.
Macintosh OS X	Go to the <code>Eclipse.app</code> bundle in the finder, select Show Package Contents, and then open the file <code>Content/Info.plist</code> .
Linux	Create an alias in your shell’s startup script (i.e., for bash, <code>.bash_profile</code>) that contains the Eclipse executable and the options.

Once you have identified the OS-specific place to put the options, they are the same on every OS. Generally, you specify the options with the following format: *eclipse [platform options] [-vmargs VMOptions]*. The *platform options* are passed to Eclipse, and the *VMOptions* (everything past *-vmargs*) are passed to the Java Virtual Machine (JVM) that is running Eclipse. The options for the JVM obviously depend on the JVM that you are running.

The one you will most likely have to change is the amount of memory allocated to the JVM. You do that with the *-Xms<size>* and *-Xmx<size>* options available on most JVMs. Eclipse 3.0 is shipping with a default setting of 150 MB max memory used, which is fine for just getting started and for small projects, but for doing serious development you should increase the memory to 256 MB. To do so, specify *-Xmx256MB* on the command line.

Eclipse options

Option	Description
<i>-application [App ID]</i>	<i>App ID</i> specifies the application to run and defaults to the Eclipse workbench. This option will become particularly important for development with and use of the Rich Client Platform (which we will talk briefly about later).
<i>-boot [boot code path]</i>	Allows you to change the location of Eclipse's startup classes, which are contained in <i>boot.jar</i> and <i>startup.jar</i> . The typical developer using Eclipse will not use this option.
<i>-consolelog</i>	Puts the error log onto the console in addition to writing it to Eclipse's internal log. If you are having a problem starting Eclipse, this is a good option for debugging what has gone wrong.
<i>-data [workspace path]</i>	The location of the workspace. This is where Eclipse keeps its metadata and is the most likely option to be specified by developers using Eclipse. (We will talk more about this option later.)
<i>-debug [options path]</i>	Specifying this option with a path to an options file will allow you to specify debugging options for the Eclipse platform. This options file is very useful if you are developing plug-ins for Eclipse.
<i>-dev [classpath entries]</i>	This option allows you to add classpath entries to the classpath of every plug-in.
<i>-nosplash</i>	This option will keep the Eclipse splash page from appearing on startup.
<i>-os [os-id]</i>	You only need to specify this option if Eclipse can't figure it out on its own. This usually happens when a new OS is added to the list of supported OSs. Most installations will not have to specify this option.
<i>-vm [vm-path]</i>	This option allows you to specify an alternate JVM to use for running Eclipse. In Windows, if you do not specify this, Eclipse will use the first <i>java.exe</i> it finds in the System PATH variable.
<i>-ws [ws-id]</i>	You only need to specify this option if Eclipse can't figure it out on its own. This usually happens when a new windowing system is added to the list of supported windowing systems. Most installations will not have to specify this option.

Users of Eclipse often use the *-data* option to specify an alternate place for Eclipse to store their projects. Changing this location allows you to put your projects into a directory that will be backed up on a regular basis. Whether or not you use the default location for your projects, make sure to back up the entire workspace directory on a regular basis. Eclipse keeps important metadata in the *.metadata* directory in your workspace. If something goes wrong with this data, Eclipse sometimes won't start, and you will have to set up all of your projects again to continue to use Eclipse.

Quick Java development tools UI

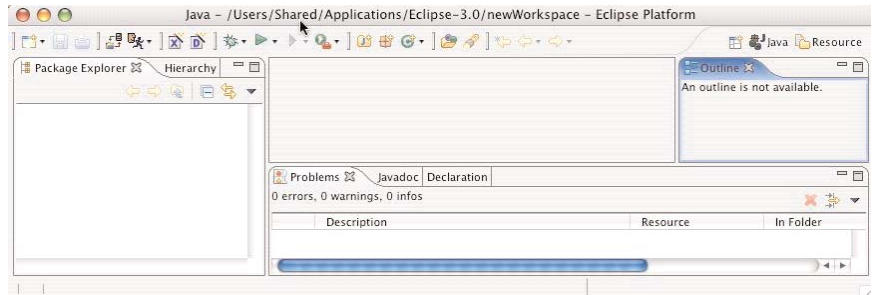
When you first open Eclipse, it will be in what is known as the Welcome View. This view has a group of buttons to help you to get to know Eclipse. Since we won't be discussing the tutorials or samples, etc., we'll skip straight to the Workbench.

Once in the Workbench, you'll be in what is known as the Resource Perspective. Perspectives are groups of views and tools that are arranged to help you do particular tasks. Since the Resource Perspective is not designed for doing

Java development, we won't spend any more time on it. Instead, once you have Eclipse in the Resource Perspective, switch to the Java perspective. Do this by clicking on Window | Open Perspective | Java. You should see a window that looks like **Figure A**.

This window is made up of several regions. The leftmost region, where the Package Explorer views can be seen, is referred to as the navigation region. The next region to the right is the editor region; this is where the Java editor will appear when we edit Java files. The next is the auxiliary view region; this region contains useful views, such as the Outline view shown in Figure A. The region along the bottom is the utility region and contains views that allow us to see interesting information about our projects.

Figure A



Creating and running a Java project

To create a new project in Eclipse, you need to invoke the New Project wizard from the Package Explorer or use the New Project button. We'll start with the context menu invocation in the Package Explorer. Within the Package Explorer, invoke the context menu (right-click in Windows; Control-click on the Mac), and then select New | Project. The first page of the new project wizard appears on the screen, as shown in **Figure B**.

If it's not already selected, choose Java Project, as shown in Figure B, then choose the Next > button. The second page of the wizard will appear. Enter the name of the project (*HelloWorld* in this simple case), as shown in **Figure C**.

In this page, you specify the name of the project and its location. The Create Project In Workspace checkbox is on by default because Eclipse expects that your typical development will reside in the workspace directory. For existing development where you will be creating a project with an existing code base, you will want to uncheck this box and select the directory that the existing code base is in. For now, leave the checkbox checked. Notice that the location is the path to the Eclipse workspace plus the project name. If you ever need to view your project using a program other than Eclipse, you will find it in this directory.

You can also specify the layout of the project on this page. The layout refers to the location of the source code and the generated class files (the compiled Java code). The default location is the root directory of the project. You can, however, edit the defaults via the Configure Defaults button. If you do edit the defaults, the next time you create a new Java project, the Create Separate Source And Output Folders radio button will be selected. It is typical for big projects to have at least one source code directory, and often there are several. For simple experimenting with Eclipse, you can leave the default of keeping the source in the root directory of the package. When you get serious about using Eclipse for real development, though, it is recommended that you create a directory to put your source code into.

Figure B

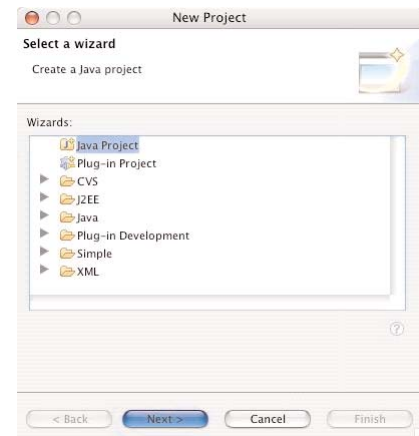


Figure C

