

# IDEA 3.0 and Ant

## Introduction

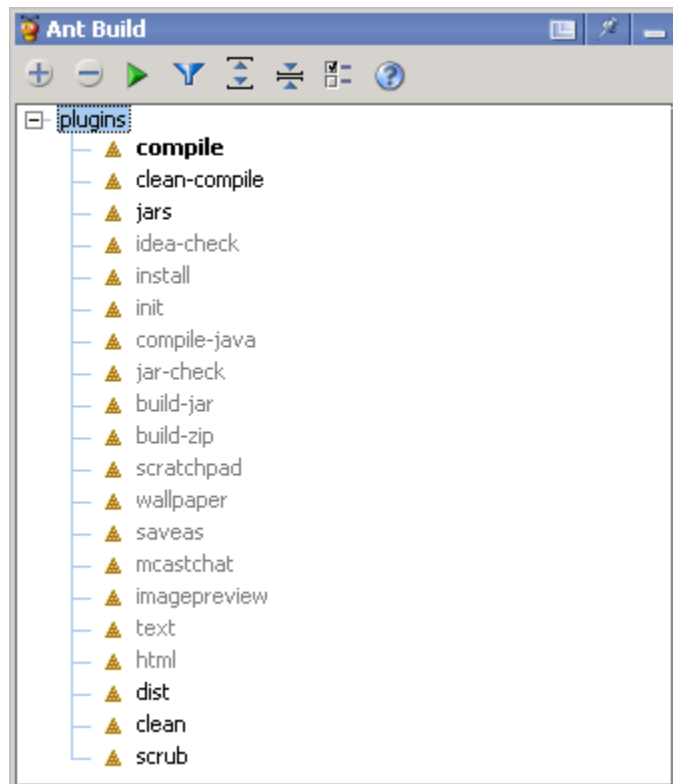
In recent years the Open Source tool Ant has become the de facto standard for building Java projects. Like its venerable ancestor make, Ant controls the steps necessary to compile, package, and arrange the files that make up your development project. Built with Java, Ant has been designed to be easy to use, flexible, and platform independent.

IDEA 3.0 provides tight integration with Ant, blending it seamlessly into the development process. From within IDEA 3.0 you can not only create and edit Ant build scripts, but also execute build targets, review the results of compilation, and even jump directly to any errors encountered. You can even automatically trigger Ant build targets upon using launching or debugging an application from within IDEA. Unlike proprietary build environments, this system provides the flexibility to build your project outside of the IDE, as required by nightly builds for example.

In this article I will discuss the Ant related features of IDEA 3.0, and explain how to integrate them into your development process. I will not however, attempt to cover the syntax or rules behind the Ant tool itself. If you are unfamiliar with Ant, suffice it to say that a build file (typically `build.xml`) defines a number of named targets (such as `compile`, `clean`, `dist`) which encapsulate a series of actions to be applied to the development tree. For details on Ant itself I direct you to the ample material at the Jakarta web site, <http://jakarata.apache.org>.

## The Ant Build Tool Window

The Ant Build Tool Window allows you to visit and execute Ant build scripts from within the IDE. To expose it, click the **Ant Build** icon on the right margin of IDEA, or select it from the **Window** menu. When you initially create a project it will be empty until you select a build script from your project path by clicking on the plus sign and selecting it in the file requestor. Understand that you must select an existing build script; one will not be created for you. After selecting a build script, the window will look something like that shown here.



You can have many build scripts defined in a single project. Each one will be listed in the Ant Build Tool Window in the tree structure shown above. Under each build script will be a list of the build targets defined in the Ant script.

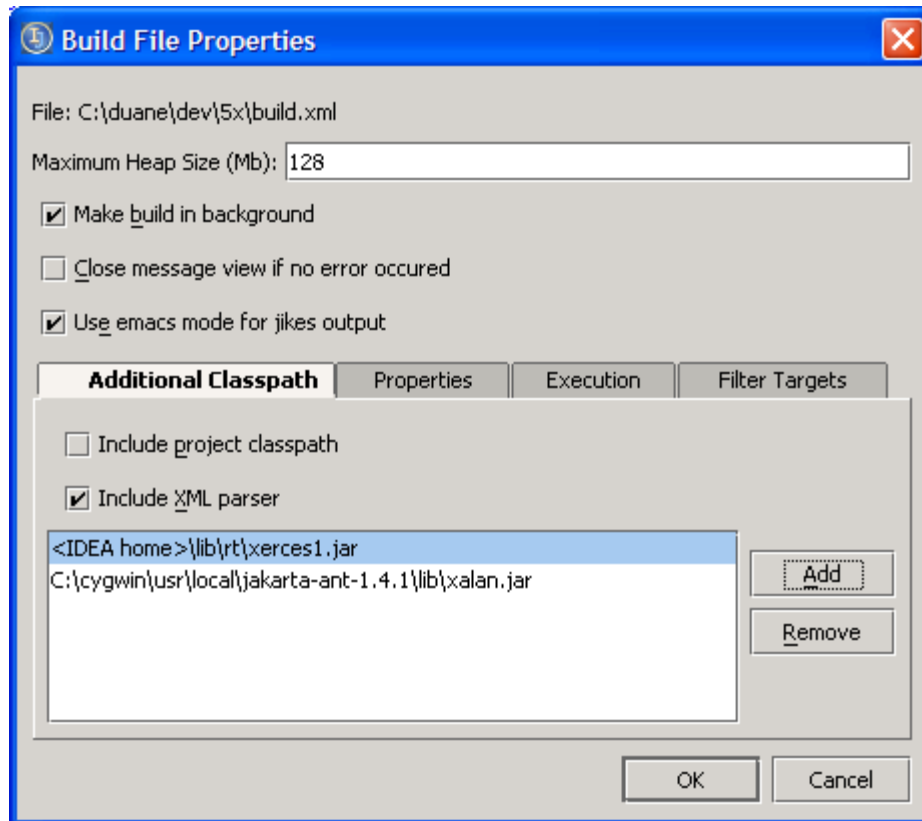
## ***Build Targets***

The default target (as defined in the build script itself) will be listed in bold. This is the target that will be executed if you click the Run icon with no target selected or double click on the name of the build file. In the above example, the default target is `compile`. The targets listed in the normal weight text are the primary targets – that is any target in the build script which has a description specified. Passing your mouse pointer over any of the targets will display its description in the form of a tool tip. This is handy for when you forget what does what.

The targets in the lighter color text are the secondary targets. Typical Ant scripts contain many targets that are not meant to be called directly, and are used like subroutines by the primary build targets. If you don't need to access these, you can hide them from view by enabling the **Filter Targets** option, the funnel icon. By default all targets without descriptions are considered secondary targets. You can control exactly which targets are displayed through the **Build File Properties** dialog.

## The Build File Properties Dialog

Each build file in your project can be customized through the **Build File Properties** dialog. You can access it by clicking on the properties icon in the Ant Build Tool Window, or by right-clicking on the build file and selecting the **properties** option. A typical example is shown below.



### Maximum Heap Size

This option allows you to control the amount of memory allocated to the Ant build process. Increase this only if your Ant process fails after running out of memory.

### Make Build in Background

By default, executing an Ant target will display a modal progress dialog, preventing you from doing anything else while the build is taking place. Selecting this option (or the option in the progress dialog) will push the build to the background, enabling you to work on other things. Be warned that compiling is often processor intensive; preventing you from accomplishing much while a build is in progress. You will want to experiment with this option in your environment to see if it is useful for you.

## Close Message View If No Error Occurred

This option tells IDEA that you don't care to see the message window if the build completes without any errors. If you get tired of closing that window after each successful build, enable this option.

## Use Emacs Mode for Jikes Output

This option applies only for users using the Jikes compiler instead of javac. If enabled, it causes Jikes to display its errors and warnings in a format readable by the IDEA build process, enabling you to hot link straight to the location of any errors encountered. If you are using Jikes, you will almost certainly want to enable this option.

## The Additional Classpath Tab

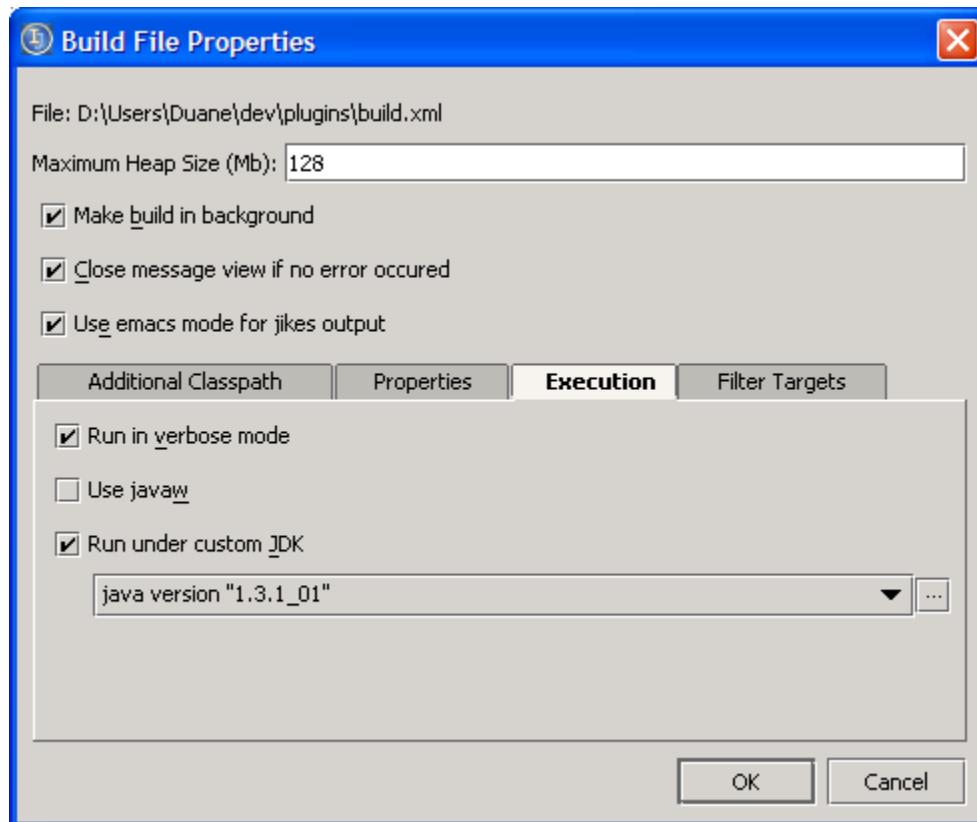
This tab allows you to control the classpath passed to the Ant build process. I have unchecked the option to **Include project classpath** in this example because my Ant script is designed to run outside of IDEA as well, and takes care of the classpath itself. The option to **Include XML parser** will add IDEA's own copy of the `xerces` library. If you are using your own Ant extensions, you will need to add them to the classpath as well. In the example above, I have included the `xalan` library because it is required by one of my build targets that use XSLT transformations to generate my tag libraries.

## The Properties Tab

The **Properties** tab allows you to define system properties to pass into the Ant build. These are the equivalent of adding `-Dproperty=value` options to the Ant command line, if you were executing it in a shell window. Unless your build script is looking for special runtime properties, you generally won't need to add anything here.

## The Execution Tab

The **Execution** tab specifies how IDEA is to launch the Ant build process when invoked. By default, IDEA will run Ant builds within its native JDK, version 1.4.1. One common point of confusion occurs when you have selected a different target JVM for your project, such as 1.3. Unless you enable the option to **Run under custom JDK** with your target JDK selected, (as shown below) your Ant targets will be built under 1.4.1, which is probably not what you want.



The **Run in verbose mode** option controls the amount of information displayed in the Message View Tool Window during the build. This information is generally hidden in the window, but is collected none the less, unless this option is unchecked. If you are finding Ant builds to be too slow in your environment you might wish to disable this option.

The option to **Use javaw** is only applicable if you are launching IDEA by manually executing its main class file, instead of the supplied *LaunchAnywhere* executable (available on most platforms). Selecting this option causes the Ant process to spawn in a windowless command shell, suppressing an extra popup window.

## The Filter Targets Tab

This tab allows you to customize the list of targets displayed in the Ant Build Tool Window by selecting and deselecting them from the list of available targets. It is a good way to reduce clutter and narrow down the number of options to pick from.

## Building with Ant

We've spent a lot of time looking at the configuration aspects up to now, but the whole point to all of this enterprise is to enable you to launch a build something! There are several ways to accomplish this in IDEA 3.0.

### *Executing a Build Target*

The most straight forward way to execute an Ant target is by double clicking it in the **Ant Build Tool Window**, or selecting it and clicking on the **Run** icon. This will execute the target, and display the results in the **Message View Window**, which we will discuss shortly.

### Running Targets from the Build Menu

One alternative way to execute Ant build targets is by selecting them from the **Build** menu. Each Ant script you have selected for your project will be listed at the bottom of the selections in the **Build** menu, with all of the primary targets listed as child selections. Selecting one of these targets is the same as selecting it from the Ant Build Tool Window.

### Assigning Shortcuts

You also have the option of assigning a shortcut key to any Ant target. Shortcuts are assigned in the **Keymaps** panel of the **Options** menu. An Ant entry, with your build files and targets below it, will be listed, enabling you to map any unassigned key combinations to your selected target. Alternatively, you can right click on the target in the Ant window, and select the option **Assign shortcut** to jump straight to the selected target in the shortcut window.

### Triggering Ant Builds On Run or Debug

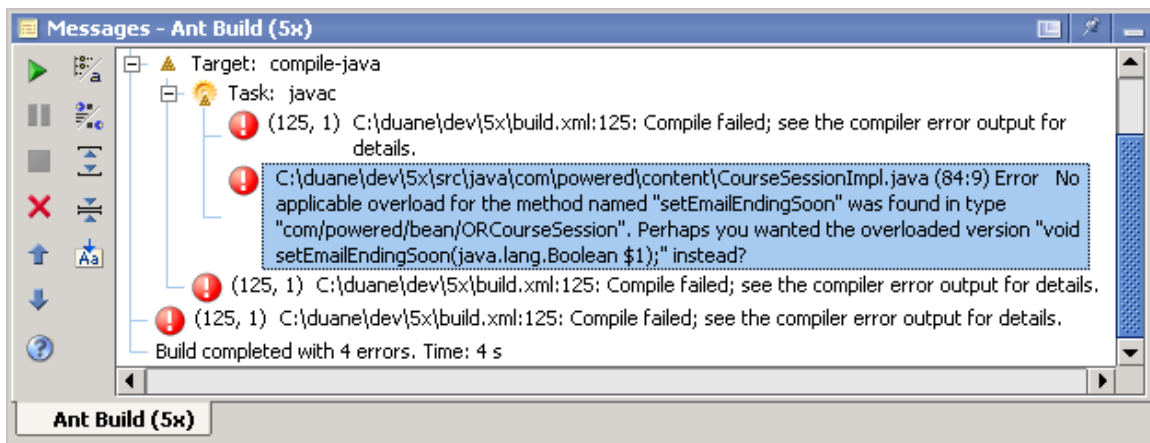
If you are taking advantage of IDEAs ability to launch, debug, or test applications you may wish to trigger an Ant build before doing so, to ensure your classes are built correctly. This can be accomplished by right clicking the target in the Ant window and selecting one of the options presented under the **Execute on** menu: **Before Compilation**, **After Compilation**, or **Before Run/Debug**. In this case the term compilation corresponds to IDEA's built-in project builder, and not any compilation your Ant script might perform.

## Message View Window

The **Message View Window** is where all of the action takes place. When you execute an Ant target, no matter how, the results will be displayed here. These results are parsed by IDEA, making them for more useful than the straight text results you might see running Ant in a command window. You have your choice of two message display formats: Tree mode and Text mode. The mode is selected via the icon in the upper right corner of the window's toolbar.

### Tree Mode

In tree mode the results of the Ant build are displayed in a hierarchy of nodes. By default, nodes are expanded only if they have produced some meaningful output (such as a compilation error). Of course you have the option of collapsing or expanding individual nodes or the whole tree en masse. A typical example of an Ant build in tree mode is shown below.



In this example there was a problem encountered with the Ant target `compile-java`. As you can see, there was an invalid method call. Shown is the source file reference and compiler error message, along with the location within the Ant script that caused the error. Clicking on either of these nodes will take you right to the offending spot.

If you are using the `jikes` compiler (as I am in the above example) you must have enabled the **Use Emacs mode for jikes output** option in the Ant properties dialog in order for this hot linking to activate.

### Text Mode

An alternative display is available through text mode. In this mode output is shown as more or less raw output, with little formatting. You do however, still retain the ability to link straight to the source of any compilation problems encountered, as shown in the example below, which is the text mode version of the build operation we looked out earlier.



## Verbose Mode

The **Verbose Mode** button (the second one from the top, in the second column) controls the amount of data Ant displays in the message view window. If you have ever played Zork or the other classic text games from Infocom, you are probably already familiar with the term *maximum verbosity*. This button enables Ant's full blown verbose output, which list the status and result of about every step of the build process. It is great for debugging your Ant scripts, but you generally will want to leave this off.

## Previous and Next Errors

These buttons work just like those in the native IDEA compiler window. They jump back and forth to the point of the error in the source file, just like clicking on the hot links in the message window. Even handier is their keyboard shortcuts CTRL+ALT+Up and CTRL+ALT+Down, which operate anywhere in the IDEA. This lets you step through (and hopefully fix) compilation errors without having to jump back to the message view window between each edit. When you have made your edits, click the **Run** icon to try again.

## Conclusions

Well that's about it. One thing left to mention is that IDEA can even help you build and edit your Ant build scripts. It understands most of the Ant tags, and will help you with completion and validation of their attributes. You can right click any target in the Ant window and jump straight into the build file for modifications.

If you are an Ant veteran, I hope this article has gotten you up and running with Ant and IDEA. If you have not yet begun using Ant in your own projects, I hope this encourages you to give it a try. Ant and IDEA are both powerful tools for streamlining your Java development process, and now with IDEA 3.0 they work wonderfully together.