

AntBuilder

AntBuilder helps you construct an Ant *build.xml* file for your project.

AntBuilder is a GUI tool and requires Java 1.5 (or later) to operate.

Ant is a project build tool published by the Apache Foundation. Although Ant is most often used for Java projects, it is perfectly capable of describing the build process for any other type of system, as well.

Introduction

If you are not using a heavy-weight development environment that obviates the need for constructing your own *build.xml* file for Ant, then AntBuilder may be the perfect little tool for you to construct and reconstruct a *build.xml* file for your project.

AntBuilder stores your choices in a file named *.antbuildrc* (note the leading dot) in the current directory. The *build.xml* that it constructs is written as *_build.xml* (note the leading underscore) so that your real *build.xml* is never touched until you are ready to put it into place. AntBuilder will never overwrite your *_build.xml* file.

AntBuilder organizes the available choices into four sections which it presents to you each on a different tab. In summary, these are:

- Main — The options most commonly of interest to you;
- Other — Some secondary options;
- JAXB — Experimental JAXB-related options;
- Paths — The paths that AntBuilder will manage for you.

File System Paths

Preferences probably vary wildly as to the names of directories where different things get stored; the defaults for AntBuilder may please or irritate you, but you can change them on a project-by-project basis:

- Source directory — This is where your source files should be located;
- Library directory — Library files should go here (this would include third-party .jar or .zip files);
- Images directory — Picture files that your application's GUI might rely on;
- Miscellaneous directory — Anything else that doesn't fit the above category;
- Build directory — The place where binaries get compiled; *this should **not** be your source directory!*

Note that the contents of the build directory are volatile; you should ever store anything there that you can't easily rebuild. Ideally you forget about that directory completely and simply let the build processes manage its contents.

AntBuilder will attempt to manage the creation and removal of these directories for you: When you accept the choices, AntBuilder creates any of the above directories that do not yet exist but will be needed according to the choices you made; likewise, if your choices indicate that one or more of them are no longer needed, then AntBuilder will remove them *so long as they are empty*. AntBuilder will never throw away or alter any files that don't belong to it.

The only files that AntBuilder will ever change are the *_build.xml* that it creates and the *.antbuildrc* into which it stores your choices so that you can easily change your mind later.

Main Options

The Project name option is quite optional, it simply gives your project a nice name, and if you ever get things mixed up and find yourself with a `.antbuilderrc` file out of place, you can see in that file what project it belonged to.

What Type of Project?

One of the options is to build a `.war` file instead of a `.jar` file for your project. If you are building a standard Java application, then you should not select that option; if you are going to deploy a `.war` file to an application server, then it's a `.war` file you want. Most of our work is focused on Java applications, not web services or application server based software, so AntBuilder is probably not as good at building `.war` files; we'd appreciate feedback.

And yes, building `.ear` files (enterprise archives) is not even part of our process at all, but if you're working on those kinds of projects without the help of much better tools, then you're probably the kind of person who drives without a seat belt, sky dives with parachutes packed by someone you don't know, and has sex without a condom. ;-)

But I digress ...

For the executable you would give only the name, not the extension, of the file that Ant should create. The filename extension (`.jar` or `.war`) will be added by AntBuilder as appropriate.

The main class is your program entry point. It is the fully qualified class name where execution begins; it's the class containing the `public static main(String[] args)` method.

What to Include

1. Do you have any external / third-party `.jar` or `.zip` files to include in your project?

These should be stored in the libraries directory (this defaults to `lib/`) and their names will be included in the `.manifest` file in such a way that your application has access to them provided that their relative paths do not change. After building your project, check out the `.manifest` file; do not be disturbed by the strange formatting of the `Class-path:` entry there, but you should be able to tell from that where those files should be stored relative to your main project's `.jar` file.

2. Do you have any images (pretty pictures, clever icons) to include in your project?

These should be stored in the images directory (this defaults to `img/`) and they will be included in the `.jar` inside an `img/` directory so that your project would need to look for “`img/myicon.png`”, for example.

3. Do you have miscellaneous files to include?

These should be stored in the miscellaneous directory (this default to `misc/`).

4. Do you wish for your project's source code to be included in the `.jar` file?

Those of us who work with Free Software and with Open Source Software will find it useful to ship the source code as part of the `.jar` file.

Other Options

You can change the default Java (source and target) version by typing a different version.

AntBuilder can include a macro that executes `git` to build a `ChangeLog` file with a little more information than a simple “`git log`” would provide. If nothing else it can serve to educate you on writing your own Ant macros!

AntBuilder can include commands that can create MD5 and SHA1 checksum files for your `.jar` file.

And lastly, you can have AntBuilder include commands to sign your application .jar file as well as third-party .jar files — Wait, you might be thinking right now, why would *I* need to sign someone else's .jar files? The issue lies with Java Web Start: It insists that all .jar files are signed by the same entity, otherwise it will refuse to start the application. This may have its root in a user interface limitation because Java Web Start only asks you to verify the identity of the first (application) .jar file and won't (or doesn't want to) ask you to acknowledge the identity of every other library from potentially numerous sources. It would be annoying, and it could start confusing you as to who was the originator of the actual application.

In short, AntBuilder can help you re-sign all libraries (if you tell it to) so that there is no deployment issue with your application software.

Changes

The original version of this document is dated 21-May-2010.