Ant

Ant is a flexible, platform-independent build tool from Apache Ant Project ⊞. IntelliJ IDEA integrates with Ant to provide a comprehensive build process, that includes compilation, packaging with the documentation and source code, committing to version control and much more.

Ant integration is shipped with IntelliJ IDEA, and you do not need to perform any additional actions to install it. However, it is also possible to use the other Ant installations.

Ant support in IntelliJ IDEA imposes certain prerequisites, and includes the following features:

- Ant
  - Prerequisites
  - Dedicated tool window
  - Ant build files
  - Ant build targets
  - Coding assistance
  - Path-like structures
- Creating Ant Build File
- Generating Ant Build File
- Adding Build File to Project
- Controlling Behavior of Ant Script with Build File Properties
- Running the Build

Prerequisites

> Before you start working with Ant, make sure that **Ant support** plugin is enabled.
>
> The plugin is bundled with IntelliJ IDEA and activated by default. If not, enable the plugin as described in Enabling and Disabling Plugins.

Dedicated tool window

Ant Build tool window enables adding Ant build scripts to IntelliJ IDEA project, control behavior of the build, and execute Ant build targets.

Ant build files

Ant works with the XML build file. Normally, the name of the build file is `build.xml`. Build file describes the steps, or build targets, required to build a project. The root element of the build file is `<project>`. IntelliJ IDEA makes it possible to work with existing build files, create new build files from scratch, or generate them automatically.

IntelliJ IDEA is aware of specific Ant syntax. However, you have to let IntelliJ IDEA know that a certain XML file is in fact an Ant build file. To be recognized as a build file and enable all advanced editing features, an Ant build file should meet at least one of the following requirements:

- The file should be properly added to the project.
- The `<project>` root element should have `default` attribute.

Otherwise such files are treated as regular XML files with basic editing support. Once a build file is added to a project, it can be used to run the build and modify its properties.

Ant build targets

A *build target* is identified with a unique name and defines a procedure that should be executed to accomplish a certain task, for example, create a JAR file, or generate API documentation. A target specified in the `default` attribute of the `<project>` element is considered the default target, which is executed when no other target is specified. This target is called the *primary target*, and is marked with bold font in the Ant Build tool window.

Coding assistance

When editing Ant build files in IntelliJ IDEA, you can enjoy the following advanced editing features:

- Syntax highlighting.
- Code completion. In particular, code completion is provided for the properties of the File type.
- Navigating to declaration `Ctrl+B`, `Ctrl+Button1 Click` or `Button2 Click`.
- Using Structure view.
- Rename refactoring.
- Code folding.
- Reformatting.
- Validation.
- Viewing parameter information `Ctrl+P`.
- Viewing quick info `GuiDesigner.QuickJavadoc`. In particular, if classpath is defined as a path-like structures, the View Quick Info command for the `fileset` or `dirset` directives displays the actual files and directories on the disk, to which these directives are resolved.

Path-like structures

IntelliJ IDEA enables using path-like structures in the task definitions. If a classpath is defined as a path-like structure, the paths in the `fileset` and `dirset` directives are resolved into the actual files and directories on the disk. All JARs, required for performing the task, should be placed to the same place that contains the JAR with task definitions.

See Also

Concepts:

- Compiler and Builder

Procedures:

- Creating Ant Build File
- Generating Ant Build File

Reference:

- Ant Build Tool Window

Web Resources:

- Developer Community ⧉