Eclipse Plug-in

The Slice2Java Eclipse plug-in manages all aspects of code generation, including automatically recompiling Slice files that have changed, removing obsolete generated classes, and tracking dependencies.

On this page:

Configuring the Slice2Java Eclipse Plug-in

Choose *Window* -> *Preferences*, select *Slice2Java*, and review the default setting for the location of your Ice installation. The property pane will display an error message if the plug-in considers the specified location to be invalid. If necessary, click *Browse...* to pick the top-level directory of your Ice installation and apply your changes.

The Slice2Java plug-in automatically configures a workspace classpath variable named `ICE_HOME` that refers to the Ice installation directory you specified in the *Preferences* dialog. This variable is primarily intended for use in Android projects.

Activating the Slice2Java Plug-in for a Project

You can activate the Slice2Java plug-in for your project by right-clicking on the project, choosing *Slice2Java* and clicking *Add Slice2Java builder*. The plug-in immediately makes several additions to your project:

- Creates a `slice` subdirectory to contain your Slice files. The plug-in automatically compiles any Slice file that you add to this directory.
- Creates a `generated` subdirectory to hold the Java source files that the slice2java translator generates from your Slice files.
- Adds a library reference to the Ice run time JAR file (`Ice.jar`). The plug-in assumes that the JAR file resides in the `lib` subdirectory of your Ice installation.

Configuring Slice2Java Project Settings

To configure the project-specific Slice2Java settings, select *Properties* from the *Project* menu or right-click on the name of your project and choose *Pr operties*. Click on *Slice2Java Properties* to view the plug-in's configuration settings, which are presented in two tabs: Source and Options.

Settings in the Source Tab

This tab configures the directories of your Slice files and generated code. The plug-in includes the `slice` subdirectory by default, but you can remove this directory and add other directories if necessary. The plug-in only compiles Slice files that are located in the configured subdirectories.

For the generated code, the plug-in uses the default name `generated` for the subdirectory. If you want to store your generated code in a different directory, you must first create the directory and then click *Browse* to select it. The new directory must be empty otherwise the plug-in will reject your change. The plug-in also requires exclusive use of this directory, therefore you must not place other project resources in it.

Settings in the Options Tab

This tab is where you configure additional plug-in settings. You can enter a list of include directories corresponding to the compiler's `-I` option. You can also specify preprocessor macros and metadata definitions in the fields provided. Finally, checkboxes offer additional control over certain features of the plug-in and the Slice compiler. When enabled, the checkboxes have the following semantics:

- **Enable streaming** generates code to support the dynamic streaming API
- **Enable tie** generates TIE classes
- **Enable ice** instructs the compiler to accept Slice symbols that use the `ice` prefix
- **Enable console** causes the plug-in to emit diagnostic information about its activities to Eclipse's console
- **Enable underscore** determines whether underscores are permitted in Slice identifiers (this feature is only supported in Ice 3.4.1 or later)

For non-Android projects, an option is also provided for adding a reference to the Freeze library.

Configuring Slice2Java File Settings

The project settings described above serve as the default compiler settings for all Slice files in the project. You may also override the compiler settings on a per-file basis by selecting a Slice file in *Package Explorer* and choosing *Properties* from the *File* menu, or by right-clicking on the file and choosing *Properties*. Select *Slice2Java Properties* to configure the Slice compiler settings, which have the same semantics as those in the *Options* tab described earlier.

See Also

- Using the Slice Compiler for Java