Slice2Java Ant Task

On this page:

- Execution Environment
- Dependencies
- Parameters
- Nested Elements
- Using the Task

Ice for Java includes an ant task named Slice2JavaTask that automates the execution of the Slice-to-Java compiler. The task and its supporting classes reside in the JAR file named ant-ice.jar, which normally can be found in the lib subdirectory of your Ice installation.

Execution Environment

The Slice2JavaTask must be able to locate and spawn the slice2java executable. You can specify the directory of your Ice installation by defining the ice.home ant property or the ICE_HOME environment variable, in which case the task assumes that the Slice compiler's executable is located in the bin subdirectory of the specified installation directory. For example, if ICE_HOME is set to /opt/Ice on Linux, the task assumes that the executable path name is /opt/Ice/bin/slice2java. Furthermore, the task also configures its shared library search path (if necessary for your platform) to ensure the executable can resolve its library dependencies.

If both ice.home and ICE_HOME are defined, ice.home takes precedence. If neither are defined, the task assumes that the executable can already be found in your PATH and that your shared library search path is configured correctly.

Finally, you can use a task parameter to specify the full path name of the Slice compiler. Again, the task assumes that your shared library search path is configured correctly.

Dependencies

The task minimizes recompilation by maintaining dependencies between Slice files. The task stores this information in a file named .depend in the output directory and updates these dependencies after each invocation. (You can specify a different name for this file using a task parameter.)

Note that the task does not maintain dependencies between a Slice file and its generated Java source files. Consequently, removing the generated Java source files does not cause the task to recompile a Slice file. In fact, the task only compiles a Slice file when any of the following conditions are true:

- no dependency file exists
- no dependency information is found for the Slice file
- the modification time of the Slice file is later than the modification time of the dependency file
- the Slice file includes another Slice file that is eligible for compilation

The simplest way to force the task to recompile all of your Slice files is to remove the dependency file.

Parameters

The task supports the parameters listed in the following table:

Attribute	Description	Required
checksum	Specifies the name of a class to contain the Slice checksums.	No
dependen cyfile	Specifies an alternate name for the dependency file. If you specify a relative filename, it is relative to ant's current working directory. If not specified, the task uses the name .depend by default. If you do not define this attribute and o utputdir is defined, the task creates the .depend file in the designated output directory (see outputdir).	No
ice	Instructs the Slice compiler to permit symbols that have the reserved prefix Ice. This parameter is used in the Ice build system and is not normally required by applications.	No
outputdir	Specifies the directory in which the Slice compiler generates Java source files. If not specified, the task uses ant's current working directory.	No
stream	Indicates whether to generate streaming support. If not specified, streaming support is not generated.	No
tie	Indicates whether to generate tie classes. If not specified, tie classes are not generated.	No
translat or	Specifies the path name of the Slice compiler. If not specified, the task locates the Slice compiler in its execution environment.	No

For the flag parameters (ice, stream, and tie), legal positive values are on, true, or yes; negative values are off, false, or no.

Nested Elements

Several Slice compiler options must be defined as nested elements of the task:

• define

Defines a preprocessor macro. The element supports the attributes name and (optionally) value, as shown below: <define name="FOO"> <define name="BAR" value="5"> <

These definitions are equivalent to the command-line options -DFOO and -DBAR=5, respectively.

• fileset

Specifies the set of Slice files to be compiled. Refer to the ant documentation of its FileSet type for more information.

• includepath

Specifies the include file search path for Slice files. In ant terminology, includepath is a *path-like structure*. Refer to the ant documentation of its Path type for more information.

• meta

Defines a global metadata directive in each Slice file as well as in each included Slice file. The element supports name and value attributes.

Using the Task

Define the following taskdef element in your project's build file to enable the task:

```
Ant
<taskdef name="slice2java" classname="Slice2JavaTask"/>
```

This configuration assumes that ant-ice.jar is already present in ant's class path. Alternatively, you can specify the JAR explicitly as follows:

```
Ant

<taskdef name="slice2java" classpath="/opt/Ice/lib/ant-ice.jar"
    classname="Slice2JavaTask"/>
```

Once activated, you can invoke the task to translate your Slice files. The example shown below is a simplified version of the ant project for the hello demo:

Ant

```
<target name="generate" depends="init">
    <mkdir dir="generated"/>
   <slice2java outputdir="generated">
       <fileset dir="." includes="Hello.ice"/>
   </slice2java>
</target>
<target name="compile" depends="generate">
   <mkdir dir="classes"/>
   <javac srcdir=".:generated" destdir="classes">
        <exclude name="generated/**"/>
   </javac>
</target>
<target name="all" depends="compile"/>
<target name="clean">
   <delete dir="generated"/>
   <delete dir="classes"/>
</target>
```

This project demonstrates some practices that we encourage you to adopt in your own projects. First, it is helpful to keep the source files generated by the Slice compiler separate from your application's source files by dedicating an output directory for the exclusive use of the Slice compiler. Doing so helps to minimize confusion and makes it easier to configure a source-code management system to ignore generated files.

Next, we also recommend that you include a clean target in your ant project that removes this output directory. Assuming that the dependency file (.depend) is also stored in this directory, removing the output directory is an efficient way to clean up your project's source tree and guarantees that all of your Slice files are recompiled in the next build.

Finally, after seeing the exclude element in the invocation of javac you might infer that the generated code was not being compiled, but the presence of the output directory in the srcdir attribute ensures that the generated code is included in the build. The purpose of the exclude element is to prevent ant from including the generated files twice in its target list.

See Also

- slice2java Command-Line Options
- Using the Slice Compilers
- Using Slice Checksums in Java
- Tie Classes in Java
- Streaming Interfaces