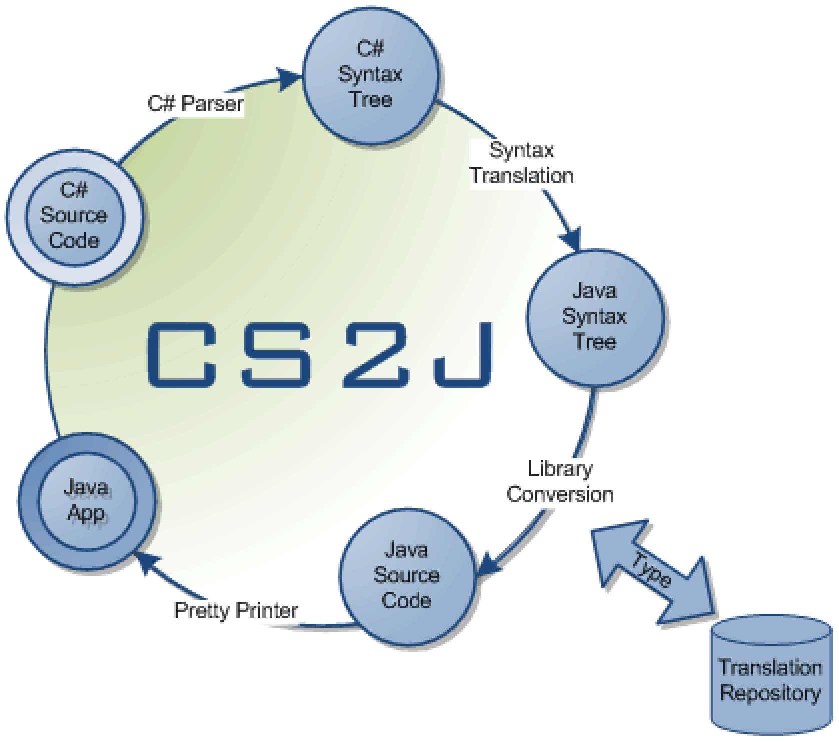
CS2J: The User Guide



Trial version

The trial version of CS2J may be used for evaluation purposes only. It has some usage restrictions compared to the full product:

1. Java classes are truncated at 120 lines.
2. The XML translation files are signed. You can make modifications, or add new translation files, and these will be used by CS2J as long as there are 5 or less translation files without valid signatures. If you hit this limit, then just restore some of the original translation files and try again.

Overview

CS2J is a C# application that converts C# types (classes, structs, enums, delegates) to Java types (classes and enums).

The translator first crawls over the whole of the C# application and builds up an internal data structure, called the translation repository, that stores translation metadata for all the application's classes, structs, enums, etc. It then extends this repository from XML files that add translation metadata for .NET Framework system calls and third party libraries used by the application. Using this translation repository it then takes each class, struct, enum, and so on, from the application and translates it to Java:

1. Translate the C# source into a C# parse tree.
2. Translate the C# parse tree into a Java(ish) parse tree. This converts C# syntax into Java syntax, it doesn't translate method calls or do any translations that depend on types.
3. Generate types for the nodes in the Java(ish) parse tree and use the translation repository to translate types and method calls into their Java equivalent.
4. Pretty print the Java parse tree to Java source files (one per top level type in the C# source file).

Running the translator

CS2J is a Windows executable that can be run from the command line. (There is also a GUI launcher which is not yet described in this document, ask for details).

To run the translator there are three required arguments:

- The directory where the XML .Net Framework translation files are held. e.g NetFramework.

- The directory that is the root of the C# application to be translated.
- The directory where the java classes will be written (e.g. JavaProject/src).

There are many, many more options too, `cs2j --help`, describes them.

A minimal command line would be:

```
CS2jTranslator\bin\cs2j.exe -netdir NetFramework -odir <java project source> <cs application root>
```

This will translate all cs files below <cs application root> and place the resultant java files below <java project source>. (The directory structure of the java files will not match the directory structure of the C# files, instead it will match the java namespaces). To translate calls to the .NET libraries the translator will use the translation templates found below NetFramework.

A slightly more complicated command line would be:

```
CS2jTranslator\bin\cs2j.exe -debug 1 -netdir NetFramework -odir <java project source> -appdir <cs application root> -csdir <cs tx root>
```

This will add all cs files below <cs application root> to the translation repository, and translate the files below <cs tx root> (for example, <cs tx root> could be a sub-part of <cs application root>).

The translator will place the resultant java files below <java project source>. To translate calls to the .NET libraries the translator will use the translation templates found below NetFramework and it will write some additional diagnostics to the terminal. Increasing amounts of diagnostics are output as the debug parameter is increased from 1 to 10.

We now briefly discuss some of the other options to the translator.

Visualizing the translation

The -showXXXX options will show the internal data structure during processing. There are options to display the parse tree at each stage: CSharp, Java Syntax, and Java.

Excluding paths

The -exXXX options allow you to exclude files and whole sub-trees (by giving the root of the excluded directory) from consideration. You can block parts of the XML translation area; parts of the application when generating the translation repository; and part of the source to be translated. For these options you can specify multiple exclusion paths separated by semi-colons.

Dumping the translation repository

The translation database generated from the application can be dumped to a set of XML files with the -dumpxml option. This produces a directory structure matching the application and XML translation namespaces. Leaf XML files show the translation for each top level C# type. These translation files are discussed in more detail in the next section.

Guiding the translation process (adding Cheats)

The -cheatdir option points to a directory hierarchy that matches the target java output directory structure. You can put two types of file here:

- files with extension .none: If file nothankyou.none exists in the cheats area then the translator won't produce a class file for nothankyou.
- files with extension .java: If file manualisbetter.java exists in the cheats area then the translator will copy manualisbetter.java instead of its own translation.

You should consider carefully before using the cheats facility. We do not use it for our main translated product. For code that we can't translate we move it into a separate (untranslated) namespace (e.g. Application.Utils) and write separate versions for .NET and Java.

CS2J Parameters

argument	***default***	***meaning***
-version		output CS2J version
-help		output help message, listing most common options
-v		increase verbosity
-debug	1	set debug level for diagnostic messages (0..10)
-debug-template-extraction	as -debug	set a different debug level for template extraction phase
-warnings	true	output CS2J warnings
-warnings-resolve-failures	false	output warnings for failure to resolve external references
-showcsharp	false	output representation of C# parse tree
-showjavasyntax	false	output representation of parse tree after C# to Java syntax pass
-showjava	false	output representation of final parse tree before pretty print pass
-D		set a C# preprocessor token (can be repeated)

-dumpxmls	false	dump translation templates (including those for the C# application being processed)
-xmldir	tmpXMLs sub directory	directory to dump translation templates
-odir	current directory	directory to write java output files
-cheatdir		directories containing cheat files
-netdir		directories/files containing .NET translation templates
-exnetdir		directories/files to exclude from -netdir
-appdir	as -csdir	directories/files containing C# application
-exappdir		directories/files to exclude from -appdir
-csdir		directories/files containing C# code to translate
-excsdir		directories/files to exclude from -csdir
-alt-translations		list of translation template variants that should have priority
-translator-keep-parens	true	keep (redundant) parenthesis from C# source
-translator-timestamp-files	true	add timestamp comment to head ofl Java files
-translator-blanket-throw	true	add “throws Exception” to all methods
-translator-make-javadoc-comments	true	convert C# XML documentation comments to Javadoc comments
-translator-make-java-naming-conventions	true	rename method names to conform to the usual Java conventions (ExecutePool() becomes executePool()). This option implies “LCC” is added to alt-translations.
-experimental-enums-numericconsts	false	convert enums to integer constants
-experimental-unsigned-to-signed	false	convert unsigned C# data types to signed Java data types
-experimental-unsigned-to-	false	convert unsigned C# data types to bigger

bigger-signed		Java signed types (e.g., ushort to int)

.NET Framework translations

The provided .NET framework translations are in the NetFramework\System sub-folder. This is structured in the same way as the .NET framework namespace, i.e., the translation for "System.Collections.ArrayList" is the file System\Collections\ArrayList.xml. The translation file's location must match the position in the .NET namespace or the translator won't find it.

These files are XML and the translator is a bit finicky about their structure and if it sees something it doesn't recognise it often fails silently. There is not yet a schema file to check correctness.

A useful trick when a translation doesn't seem to be picked up is to ask the translator to dump the translation database (option -dumpxml) and look at the resultant xml files to see if it recognized the translation template.

Translation files

The format of the translation file deserves another document, unfortunately it isn't written (yet), so you will have to divine it from the provided translations, and asking questions.