Janino

A super-small, super-fast Java Compiler

Arno Unkrig 2014-02-18

Agenda

- Quick Start
  - Setup
  - ExpressionEvaluator
  - ScriptEvaluator
- The Details
- Inside the Project

Setup

- Download `janino-`*x*`.`*y*`.`*z*`.zip` from

  `http://janino.net`

- Extract

  `commons-compiler.jar`
  `janino.jar`

- Put them on the class path

ExpressionEvaluator

```
IExpressionEvaluator ee = new ExpressionEvaluator();
ee.setExpressionType(int.class);
ee.setParameters(
    new String[] { "c", "d" },            // parameterNames
    new Class[] { int.class, int.class } // parameterTypes
);

// Compile the expression once; relatively slow.
ee.cook("c > d ? c : d");

// Evaluate it with varying parameter values; very fast.
Integer res = (Integer) ee.evaluate(
    new Object[] { 10, 11 }               // arguments
);
System.out.println("res = " + res);
```

ScriptEvaluator

```
// Create "ScriptEvaluator" object.
IScriptEvaluator se = new ScriptEvaluator();

se.setReturnType(boolean.class);
se.cook(
    "System.out.println(\"Hello world\");\n"
    + "return true;\n"
);

// Evaluate script with actual parameter values.
Object res = se.evaluate(
    new Object[0]           // arguments
);
System.out.println("res = " + res);
```

Agenda

- Quick Start
- The Details
  - Compiler
  - JavaSourceClassLoader
  - SimpleCompiler
  - ClassBodyEvaluator
  - Debugging Janino Scripts
  - Alternative Implementations
- Inside the Project

org.codehaus.janino.Compiler

- Provides all features of JAVAC:
  Source path, class path, extension directories, boot class path, destination directory, verbose flag, debug options
- Additional features: Warning handle patterns, rebuild flag, source finder
- Wrapper shell script "janinoc" is a drop-in replacement for JAVAC
- Bindings for ANT and TOMCAT exist

org.codehaus.janino.JavaSourceClassLoader

- Scans, parses, compiles and executes Java source files on-the-fly
- No class files created on the file system
- Wrapper shell script „janino“ implements a combination of JAVAC and JAVA

One bizarre example:

Run the JANINO compiler from source to compile itself:

```
$ janino -sourcepath src org.codehaus.janino.Compiler \
> -sourcepath src -d bin \
> src/org/codehaus/janino/Compiler.java
$
```

org.codehaus.janino.SimpleCompiler

- Scans, parses, and compiles a single compilation unit from a `String`/`Reader`/`InputStream` into a `ClassLoader`
- No class files created on the file system

org.codehaus.janino.ClassBodyEvaluator

- Scans, parses and compiles the body of a Java class
- Loads it as a `java.lang.Class`

Debugging JANINO scripts

In order to be able to single-step through JANINO-generated classes:

- Set
  `-Dorg.codehaus.janino.source_debugging.enable=true`
- Optionally set
  `-Dorg.codehaus.janino.source_debugging.dir=C:\tmp`
  to an existing directory

Alternative implementations

Instead of

```
import org.codehaus.janino.*;
new ExpressionEvaluator()
```

, call

```
import org.codehaus.commons.compiler.jdk.*;
new ExpressionEvaluator()
```

, or, even better:

```
import org.codehaus.commons.compiler.*;
CompilerFactoryFactory
    .getDefaultCompilerFactory()
    .newExpressionEvaluator()
```

Agenda

- Quick Start
- The Details
- Inside the Project
  - How does JANINO work?
  - History
  - Project Structure
  - Projects using Janino
  - FAQ

How does JANINO work?

- Create an abstract syntax tree (AST) programmatically and/or by parsing Java tokens, rooted at a `CompilationUnit` object
- Compile the AST into `ClassFile` objects
- During compilation, load referenced classes through the source path and/or the class path
- Store the `ClassFile` objects to disk (`Compiler`), or
- Load the generated class files through a custom `ClassLoader`

History 1/2

- Q4/2000: Development of an interpretative expression evaluator for INTERSHOP ENFINITY at Gauss AG
- Q3/2001: Re-implementation with byte code generation
- Q4/2001: Initial LGPL Release
- Continuous addition of language elements
- Q4/2003: Version 1.0 implements all of Java 1.1

History 2/2

- Q2/2004: Version 2.0 adds inner classes an consorts (Java 1.2)
- Q1/2005: Janino moved to CODEHAUS
- 2006/2007: Add part of the Java 5 language features
- 2008-2009: Bug fixing and minor improvements
- 2009: More committers join the project
- Q1/2010: Interfacification of API; add `javax.tools`-based alternative implementation
- 2012-2014: Addition of most Java 5 features

Project Structure

- Project web site
  `http://janino.net`
- Development: Four committers
- Version control (Subversion)
  `http://svn.codehaus.org/janino`
- Issue tracking (JIRA)
  `http://jira.codehaus.org/browse/JANINO`

Projects using Janino

Open source projects that use JANINO:

- Groovy -- an agile dynamic language for the JVM combining lots of great features from languages like Python, Ruby and Smalltalk and making them available to the Java developers using a Java-like syntax
- Drools -- an augmented implementation of Forgy's Rete algorithm tailored for the Java language
- Kataba Dynamics -- less verbose, more powerful, more consistent core libraries for Java
- JINX -- Java multi-user Unix-like system

FAQ

- License – New BSD
- ???