The EEL Project

Table of contents

1. What's this?

This page is about tweaking Eclipse 3.1.1 or Eclipse 3.2 to produce JDK 1.4 (or lower) compliant bytecode, even when source Level is set to 5.0.

This is especially useful if on your platform there is no JDK 1.5 available yet (e.g. Mac OS X) or if the application server you have to work with is not supporting 1.5 yet (e.g. Websphere 5.x which is still based on JDK 1.3(!))

javac does not allow you to go ahead with source=1.5, target<1.5, (`javac: source release 1.5 requires target release 1.5`); neither does the eclipse UI. This page will teach you how to workaround the Eclipse UI and also overcome a few issues when setting this "illegal" target mode.

2. Why does this work at all?

Most features added in Java 5 (or JDK1.5, as you desire) are actually "just" preprocessor features. This means the features have no impact on the classfile format, or if they result int the classfile, earlier virtual machines than version 1.5 just ignore these properties. E.g. Generics information gets mostly lost through "Erasure". The Virtual machine was only slightly modified (ldc,ldc_w instructions) for Java 5 (and I hope we will just not run into any kind of problems because of this).

What's more, the eclipse compiler people did an excellent job in *internally* supporting other output formats than JDK1.5, even if source compatibility is set to JDK 1.5. On the GUI, you won't be able to do this change, but if you modify the concerned settings file (org.eclipse.jdt.core.prefs in the .settings folder of you Java Project), Eclipse will allow you to go ahead with Source=1.5 and target=1.2|1.3|1.4. The GUI is disabled for policitcal reasons.

3. What is supported

- varargs
- static imports
- Generics
  These work since the Parameter information gets (mostly) lost during compilation. The hard part is that you can't simply compile against 1.5 libraries and expect the thing to work on earlier virtual machine. For example, you run into problems when (accidentially) using a new class or method (e.g. Queue in the 1.5 collection classes).

  The workaround is like this: "Port" the {1.2|1.3|1.4} classes you require to a "parametrized" version, by only modifying their public interface (no method body etc.

required) to their JDK1.5 counterpart. Put this class on the **Build Path** and eclipse will happily compile against them.
For the runtime classpath, DO NOT include these classes there, so that at runtime, the java virtual machine will see the "original", non patched version of the classes.
For myself, the 1.4 Collection classes are most useful, so I started porting them. The current version can be downloaded here. Lots of classes are still missing, but the most common ones (Vector, List, ArrayList, HashMap, etc.) are already done. Tell me if you need anything else, and I'll do it.

Update: This jar now includes ALL Collection classes, i.e. all classes have been ported (January 2006)

- for/in statement (aka. foreach loop)
  This works for arrays and classes that implement `java.lang.Iterable`. (Default 1.5 behaviour). This interface is included in the "patched" 1.4 collection classes. See download section . Luckily, JDK1.4 (and 1.3) `java.util.Collection` already implements this interface, so for all "ported" Collection classes, for/in should work.
- Autoboxing and Unboxing
  Currently, this is supported by applying the following eclipse 3.1.1 jdt source patch . The patch makes sure that, unlike for a jdk1.5 target, the eclipse compiler does not emit e.g. Integer.valueOf(int) for a Autoboxing operation, but instead uses new Integer(int).

  Update: thanks to Olivier Thomann of eclipse, this patch will make it into version 3.2 of Eclipse. Look here for details.
- Annotations
  `@SuppressWarnings` is tested. I assume the other standard annotations will work too: `@Override` `@Deprecated` I would expect custom annotations to work as well.
- Running the eclipse compiler in "batch mode" e.g. in ant scripts when you are using

  ```
  <property name="build.compiler"
  value="org.eclipse.jdt.core.JDTCompilerAdapter"/>
  ```

  and want to use source=1.5 and target=1.4, e.g. inside a javac task

  ```
  <compilerarg
  compiler="org.eclipse.jdt.core.JDTCompilerAdapter"
  line="-1.5 -source 1.5 -target 1.4"/>
  ```

  Update: in Eclipse 3.2 this has to look differently forget about the <property name="build.compiler" etc. line, it has become much easier. In the javac line of your ant script, set source and target like this

  ```
  <javac (...) source="1.5" target="jsr14" (...)
  ```

4. What is NOT supported

- enumerations
- direct references to `java.util.Iterable`

5. Summary

To make it clear: You do not need any sourcecodetweaks anymore in Eclipse >=3.2

- Autoboxing has been fixed by Olivier Thomann in the compiler in November 2005.
- The batchcompiler supports the features by specifying the following compiler arguments

```
source="1.5" target="jsr14"
```