Getting the Photran 7.0 Sources from CVS

and Creating Launch Configurations

Revision: $Id: app-cvs.ltx-inc,v 1.16 2010/06/24 13:21:41 joverbey Exp

Before you begin…

1. You will need to be running **Eclipse 3.6 (Helios)** on a Java 5 or later JVM. You should install *Eclipse Classic 3.6* (also called the *Eclipse SDK),* which includes the Eclipse Platform, Java development tools, and Plug-in Development Environment. (Since Photran is a set of Eclipse plug-ins, you need a version of Eclipse that includes the Plug-in Development Environment.)
2. *(Optional)* We also recommend that you install the following Eclipse plug-ins:
   (a) **Subclipse** – provides Subversion support
   (b) **FindBugs** – a static analysis tool for Java
   (c) **EclEmma** – a Java code coverage tool
   (d) **Metrics** – provides code metrics for Java code

Part I. Check out the CDT 7.0 sources from CVS

1. In Eclipse, switch to the CVS Repository Exploring perspective.
2. Right-click the CVS Repositories view; choose New, Repository Location
3. In the dialog box, enter the following information, then click Finish.

Host name:	dev.eclipse.org
Repository path:	/cvsroot/tools
Username:	anonymous
Password:	(no password)
Connection type:	pserver

4. In the CVS Repositories view
   - Expand ":pserver:anonymous@dev.eclipse.org:/cvsroot/tools"
   - Then expand "HEAD"
   - Then expand "org.eclipse.cdt"
   - Then expand "all"
5. Click on the first entry under "all" (it should be org.eclipse.cdt), then shift-click on the last entry under "all" (it should be org.eclipse.cdt.ui.tests). All of the intervening plug-ins should now be selected. Right-click on any of the selected plug-ins, and select Check Out from the pop-up menu. (Check out will take several minutes.)
6. You now have the CDT source code. Make sure it compiles successfully (lots of warnings, but no errors).

Part II. Check out the Photran sources from CVS

7. Under ":pserver:anonymous@dev.eclipse.org:/cvsroot/tools," expand HEAD, then expand org.eclipse.ptp, then expand photran
8. Click on the first entry under "photran" (it should be org.eclipse.photran-dev-docs), then shift-click on the last entry under "photran" (it should be org.eclipse.rephraserengine.ui.vpg). All of the intervening plug-ins should now be selected. Right-click on any of the selected plug-ins, and select Check Out from the pop-up menu. (Check out will take several minutes.)
9. *(Optional)* You may also wish to switch back to the Java perspective and...
   (a) Close or delete the org.eclipse.photran.cmdline project. This project contains code demonstrating how to use Photran's parser outside of Eclipse; most developers won't need to do this, and since it contains many classes copied from the "real" Photran projects, it can be confusing having it in your workspace.
   (b) Delete the following projects (they are used for release engineering, so most developers will never need to modify them):
      - org.eclipse.photran-feature
      - org.eclipse.photran.intel-feature
      - org.eclipse.photran.master
      - org.eclipse.photran.releng
      - org.eclipse.photran.vpg-feature
      - org.eclipse.photran.xlf-feature
      - org.eclipse.rephraserengine-feature

The sources should all compile (albeit with lots of warnings).

Revision: $Id: app-tests.ltx-inc,v 1.5 2010/05/17 20:28:54 joverbey Exp

Part III. Running the test cases

10. In the Package Explorer view, select the `org.eclipse.photran.core.vpg.tests` project.
11. Right-click on that project and select Run As > Run Configurations.... A dialog will appear.
12. In that dialog, create a new **JUnit Plug-in Test** launch configuration. Call it "Photran-Tests".
13. For the configuration that you have just created, switch to the "Arguments" tab.
14. Change the "VM arguments" field to `-ea -Xms40m -Xmx512m`
15. Switch to the "Environment" tab.
16. *(Optional)* If you are running Linux or Mac OS X and have gfortran installed, some of Photran's refactoring unit tests can attempt to compile and run the Fortran test programs before and after the refactoring in order to ensure that the refactoring actually preserves behavior (and produces code that compiles). The following steps will enable this behavior. Note, however, that if the path to gfortran is incorrect, or if gfortran cannot be run successfully, it will cause the test suite to fail... so you might not want to do this the very first time you attempt to run the test suite.
   (a) Create a new environment variable called `COMPILER` with the full path to gfortran. This will be something like `/usr/local/bin/gfortran`
   (b) Create a new environment variable called `EXECUTABLE` with a path to some non-existent file in your home directory, e.g., `/Users/joverbey/a.out`. When gfortran is run, it will write the executable to this path.

17. Click the "Run" button to run the tests. It will take at least a minute to run the test suite. When it finishes, you should get a green bar in the JUnit view. If you get a red bar, some of the tests failed; the JUnit view will have details.

18. To run the tests again later, just launch the "Photran-Tests" configuration from the Eclipse Run menu.

    Note. *UIUC personnel: See the appendix "Additional Information for UIUC Personnel" in the Photran Developer's Guide for information on additional unit test cases.*

Revision: $Id: app-launch.ltx-inc,v 1.2 2010/06/24 13:21:32 joverbey Exp

Part IV. Launching Photran in the debugger

19. In the Package Explorer view, select the `org.eclipse.photran.core` project (or any other plug-in project).

20. Right-click on that project and select Debug As > Debug Configurations.... A dialog will appear.

21. In that dialog, create a new **Eclipse Application** launch configuration. Call it "Photran".

22. For the configuration that you have just created, switch to the "Arguments" tab.

23. Change the "VM arguments" field to:

    -ea -XX:PermSize=64m -XX:MaxPermSize=128m -Xms64m -Xmx768m

    (These arguments will enable assertions, increase the amount of PermGen space, and increate the amount of heap space available to Eclipse.)

24. *(Optional)* If you will be developing fixed form refactorings, or if you need fixed form refactoring enabled...

    (a) Switch to the "Environment" tab.

    (b) Create a new environment variable called `ENABLE_FIXED_FORM_REFACTORING` with a value of 1.

25. Click the "Debug" button. A new instance of Eclipse will open with the CDT and Photran plug-ins compiled from the code in your workspace.

26. To run it again later, just launch the "Photran" configuration from the Eclipse Run menu. Debug > Debug History > Photran will launch it in the debugger again (this will allow you to set breakpoints, watch expressions, etc.), while Run > Run History > Photran will launch it in a normal JVM (with debugging disabled).