AspectJ 1.7.0 Release Review

Planned Review Date: [Date]

Communication Channel: aspectj-users@eclipse.org, aspectj-dev@eclipse.org

Andy Clement, Andrew Eisenberg

Introduction

- AspectJ is a seamless extension to Java that adds the ability to capture cross-cutting concerns
- It adds a few new keywords and constructs (e.g. pointcut, aspect) to the Java language and provides a compiler that understands these extensions
  - The compiler is a modified form of the JDT core compiler
- It also includes a weaver that can be used to apply cross cutting concerns to code that has previously been compiled to bytecode
  - The weaver can be used as an offline post-compile step or as a load-time weaver.

Features

- AspectJ major/minor version numbers have traditionally tracked Java version numbers
  - AspectJ 1.7.0 is the first Java 1.7 version of AspectJ
  - AspectJ takes and modifies the JDT compiler. For 1.7.0 AspectJ has been rebased on the Eclipse 3.7 compiler (JDT build tag B79). This is a big move, AspectJ 1.6.X was based on Eclipse 3.3 JDT core.
  - 1.7.0.M1 readme: http://www.eclipse.org/aspectj/doc/released/README-170.ht
    - Basically showing ability to use 1.7 constructs in AspectJ code.

Features

- Declarative aspect construction.
  - The XML files, traditionally only used to 'fill-in' abstract aspects, can now be used to define entire aspects (no abstract aspect required). This provides more flexibility for users wishing to generate aspects on-the-fly as they can create XML more easily than generate/compile source code
    - https://bugs.eclipse.org/bugs/show_bug.cgi?id=375881
    - https://bugs.eclipse.org/bugs/show_bug.cgi?id=359159
- LTW caching
  - Faster restarts now possible by caching woven code
    - https://bugs.eclipse.org/bugs/show_bug.cgi?id=367673

Features

- Weaver upgrade for Java 1.7
  - On the back end the AspectJ weaver has been upgraded to understand the new bytecode changes in Java 1.7 (new constant pool entries, new invokedynamic instruction) – however it is merely tolerating these changes for now, not exploiting them
    - Toleration enables us to weave Java 1.7 code

Non-Code Aspects

- The readmes for each release continue to provide the most up to date documentation, some of the new features discussed in these do need folding into the main documentation.
- All the existing documentation (getting started, reference material, etc) remains valid and relevant to AspectJ 1.7.0
- Moved to git from cvs for 1.7.0 release
  - Ditched some unwanted code/modules in the move

APIs

- Primary API exposed for integration into AJDT
  - recent releases have increased the granularity in the API to enable finer grained interactions between AJ/AJDT → improving incremental compilation

Architectural Issues

- On the front end AspectJ continues to be based on a modified JDT core compiler, there is no real need for additional extensibility in this area
  - However, continuing to maintain a large 'patch' on JDT core does slow down the ability to keep up with Eclipse versions (hence skipping eclipse 3.4/3.5/3.6 !)

Tool Usability

- For the Eclipse UI, defer to the AJDT project
- As a pure compiler/weaver the project is currently actively (and successfully) used through:
  - Command line batch invocation
  - Loadtime weaving (-javaagent)
  - Maven AspectJ plugin
  - Gradle (no central plugin but a number of users building their own custom plugins pulling in AspectJ)
- The maven plugin does fall behind with supporting new options as it isn't the AspectJ team maintaining it – we may try to get more involved with it

End-of-Life

- AspectJ continues to maintain a high degree of backwards compatibility. Programs compiled with versions back to AspectJ 1.2 will work just fine with the latest AspectJ release
- Nothing is being end-of-lifed/removed in 1.7.0

Bugzilla

- Bugs/Enh opened in last 6months: 33
- Bugs/Enh resolved in last 6months: 23
- Total bugs/enh open against AJ: 332bugs 198enh
  - No P1 Bugs open
- Bugs resolved against 1.6.12: 60 (previous release: 18-Oct-2011)
- Bugs resolved against 1.7.0: 17
  - Fewer bugs closed against 1.7.0 as larger issues have been tackled, like moving to Java 7
- Bugzilla could do with a pass to close a number of the minor/niche problems that we just won't get to in the foreseeable future

Standards

- J2SE
  - AspectJ now utilizes generics in its source code
    - Requires Java 1.5 (this is a divergence from JDT core which only requires Java 1.4)
  - Code generated by AspectJ can run on Java 1.1 and later
  - AspectJ 1.7.0 can now cope with compiling Java 1.7 source code or weaving into previously compiled Java 1.7 class files.

UI Usability

- Defer to AJDT project for Eclipse UI usability

Schedule

- AspectJ 1.7 has been a long time coming as our version numbers follow Java version numbers (and Java 1.7 was heavily delayed).
- Upgrade to Java 1.7 was relatively easy as AspectJ could build upon the work done in JDT core
- AspectJ has been on a 3month release schedule through the 1.6 releases (1.6.0 → 1.6.12)
  - AspectJ 1.7 will likely follow the same model

Communities

- Mailing list continues to be the most active place for AspectJ discussions – 99% of posts getting a response within 24hours
- Most bugs triaged within 48hours
- Inclusion of AJDT in SpringSource Tool Suite drives some traffic on the STS forums related to AspectJ
- Blog on AspectJ and other eclipsey stuff: http://andrewclement.blogspot.ca/
  - Could do with a recent article. Plan to make a splash for the 1.7.0 release

IP Log

- Nothing unusual to report for 1.7.0
- Link to IPlog will be here once approved

Project Plan

- http://www.eclipse.org/projects/project-plan.php?project
- Work items on the horizon
  - persistent build state to avoid full builds being required on eclipse startup
  - More memory optimization work
- Future plans may include
  - adding new language constructs to support weaving of the invokedynamic instruction

IP Issues

- The EMO explicitly asks during the Release Review if any Member would like to assert that this release infringes their IP rights.
- If so, the EMO and the project will follow the Eclipse IP Policy in discussions with that Member.