

# The Scala IDE for Eclipse Retrospect and Prospect for 2.8.0

Miles Sabin, Chuusai Ltd.  
<http://www.chuusai.com/>

# Outline

- History
- Current Development
- Roadmap for 2.8 Preview
- Roadmap for 2.8 Final
- How to Help
- Roadmap Beyond 2.8 Final
- Speeding up Development

# History 1

- Started very early on in Scala's timeline (early 2005)
- Very simple IDE plugin
- Little functionality beyond basic syntax highlighting and build invocation
- Written in Java
- There was a least one other similar offering at the time: Scaliptor

# History 2

- Announced December 2005, first release February 2006 (2.1.0)
- Rewritten in Scala
- Some semantic features acquired (eg. limited auto-completion)
- Even at this early stage requests for the ability to mix Java and Scala and JDT interop were coming in

# History 3

- Announced June 2007, first release February 2008
- Attempted much deeper integration with the Scala compiler,
  - Interactive error reporting
  - Semantic highlighting
  - Incremental compilation
  - Dependency management
- Many hooks added to scalac

# History 4

- Start of my involvement (May 2008)
- Prompted by very generous sponsorship by EDF Trading
- Primary goals
  - Ease Java/Scala migration
    - Mixed Scala/Java compilation in scalac
    - Mixed Scala/Java projects in Eclipse
  - Improved Eclipse stability and release process

# History 4

- First commit to trunk in July 2008
- Results of the work first visible in 2.7.2.RC1 in August 2008
- Final 2.7.2 release in November
- Goals somewhat met,
  - Mixed Scala/Java enabled in scalac and Eclipse
  - Release process dramatically improved
  - However JDT integration limited and stability issues remain

# History 5

- Serious roadblocks to JDT integration
- 2.7.3 contained only very minor bugfixes and enhancements
- Long standing patch to open up the JDT for extension updated but rejected
- AspectJ and Equinox Aspects provided a Plan B

# History 5

- JDT Roadblocks removed enabling rapid progress,
  - Many bugs fixed
  - Many features enhanced
- First appeared in 2.7.4 in April 2009
- Some drawbacks however,
  - Some installation issues
  - Some JDT features magically started to almost work

# History 6

- Solution to preceeding: full speed ahead on JDT integration, recruit contributors
- However, much of the codebase complex over-abstracted and forbidding
  - Learning curve too steep for casual contribs
  - Fragility in presentation compiler
    - Spurious pink squiggles, minor errors destroy all highlighting, unreliable dependency management
  - Negative effect on scalac
    - Many fragile “if (inIDE) ...” blocks

# History 6

- Complete reimplementations of all semantic and build related features
- Work done in collaboration with Martin Odersky and Iulian Dragos at EPFL in May

# History 6

- New interactive compiler
  - Completely replaces old presentation compiler
  - Now the unit of (re)compilation is a whole source file.
    - Performance is on a par if not better than before, old approach a “premature optimization”
  - All “inIDE” conditions removed from scalac
  - Accurate position information to AST nodes
    - Enables new features (folding)
    - Supports new tools (refactoring, formatting)

# History 6

- New incremental build manager
- Far more reliable than previous Eclipse-specific code
- Makes use of scalac's filesystem abstraction enabling,
  - Use by other IDEs
  - Use by other tools (eg. refactoring)
- Supports all JDT build path features
  - Multiple source and output directories

# History 6

- Drawbacks
  - Big bang ... there's no going back,
    - All support for previous integration has been removed from scalac
    - All semantic features in Eclipse have to be reworked
    - No feasible backport of new features or bugfixes to 2.7.x branch
  - This is in stark contrast to the incremental improvement policy I had intended to adopt from 2.7.2 onwards.

# Current Development

- The new build manager has been fully integrated. “clean build, open/close hell” is largely a thing of the past
- Previously project-specific compiler instances were inextricably linked to the project itself, so “rebooting” the compiler required the project to closed and opened
- Now fully decoupled and compiler is reinstantiated on a clean

# Current Development

- Removing the old presentation compiler has had some direct benefits,
  - The Scala editor is now effectively identical to the Java editor
    - This gives us features for free: comment and code folding
  - New lightweight syntax highlighting based on the JDT's for Java
    - Driven by a low-level source scanner, so very robust in the face of syntax and semantic errors

# Current Development

- Hyperlink navigation (“Jump to definition”) close to completion
  - Completely new implementation based on the JDT's mechanisms
  - Scala source code selections mapped to Scala AST nodes and Eclipse JDT model handles
  - Some “re-sugaring” required to map AST nodes to appropriate JDT construct
  - Supports “Jump to implementation” in many cases

# Current Development

- The underlying mechanism supports additional JDT features,
  - Javadoc hovers
  - Source hovers
  - Java search actions (eg. find references)
- Currently this is only fully functional for Java symbols appearing in Scala source
  - Aiming to have this uniform across Scala and Java before release

# Current Development

- Reusing the JDTs navigation required the extension of AST mapping to binaries (eg. class files in the Scala standard library Jars)
  - Currently only supported for binaries with source attachments
- This provides dramatic improvement in navigability for the standard library
  - Library components now open in Scala editor
  - Outline view and outlines in package explorer now available

# Current Development

- The full mapping of Scala source and binary components enables comprehensive JDT-wide indexing of Scala elements
  - The JDT indexer now understands Scala traits and objects
  - Indexing supports uniform “Open Type” behaviour across Scala and Java projects
  - Supports “Find references” to both Scala & Java in both Scala & Java sources
    - Still work in progress, but already useful

# Current Development

- The new presentation compiler is much more reliable
  - No longer used for syntax highlighting, so doesn't have to run synchronous with keystrokes and whole CU update is fast enough
- Currently supports,
  - Error reporting as-you-type
  - Live outline update in the outline view and the package explorer
  - Code completion (work in progress)

# Roadmap for 2.8 Preview

- Completion of all in-progress major features
- Reinstatement of semantic highlighting
- Support for Eclipse tasks framework
- Resolution of most Trac bugs,
  - REPL issues on Windows and Mac OS X)
- Fit and finish.
  - Disabling of partially functional JDT features

# Roadmap for 2.8 Preview

- Notable absences,
  - Refactoring
    - Organize imports
    - Rename/move element
  - Code formatting
- New work by Mirko Stoker would make any effort here redundant
- The aim is to integrate his work before 2.8.0 final, but impossible to make a commitment at this point

# How to Help

- A call for testers will go out once trunk is feature complete (sometime in the next few weeks)
- At that point please kick the tires of nightly builds, report bugs and partial features which are candidates for suppression
- Review (and hopefully close) old bugs which have been fixed as side-effects of other work

# Roadmap for 2.8 Final

- Integrate the basic refactorings if they're ready.
- Focus on testing and stability
  - Other than refactoring, there will be no additional features beyond those already planned for the preview release

# Roadmap Beyond 2.8 Final

- Minor features, more of the same
  - Additional semantic annotations
    - Markers/hovers for in-scope and applied implicits
  - More refactorings and formatter options
    - IDE independent API lowers the barrier to entry for external contributors, effort shared across tools
  - Support for binary elements without source
  - Maven support (eg. M2eclipse)
  - Gradual re-enabling of suppressed JDT features

# Roadmap Beyond 2.8 Final

- Major features
  - Support for Scala-specific tools
    - Lift-specific tooling – “New Lift project”
  - Paul Phillips Scalify
    - “Refactor” from Java to Scala
  - Debugger enhancements
    - Execution of Scala fragments, value hovers
  - New REPL features
    - Support for Paul Phillips new completion behaviour
    - Semantic highlighting

# Roadmap Beyond 2.8 Final

- Compatibility with other common Eclipse extensions,
  - Google App Engine
  - Android
  - AJDT and STS
  - Other JVM languages which take a similar approach to the JDT: Groovy, JavaFX
- We need a compatibility matrix for Eclipse extensions – will need community help

# Speeding up Development

- **Contribute**
  - Code
  - Bug reports
  - End-user and developer documentation
  - Howtos and experience reports
- **Help build a user community based on constructive criticism**
  - Contributors encouraged by positive vibe
- **Sponsorship**

# The Scala IDE for Eclipse Retrospect and Prospect for 2.8.0

Miles Sabin, Chuusai Ltd.  
<http://www.chuusai.com/>