XQuery Language Aware Plugin for IntelliJ

Ron Hitchens

Principal Consultant, OverStory Ltd

ron@overstory.co.uk

Licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.
© 2012 OverStory Ltd

Who Am I?

Former Lead Engineer at MarkLogic (5 years)

Created XCC, way back when

Written lots of XQuery over the years

Even wrote an XQuery book (Pragmatic)

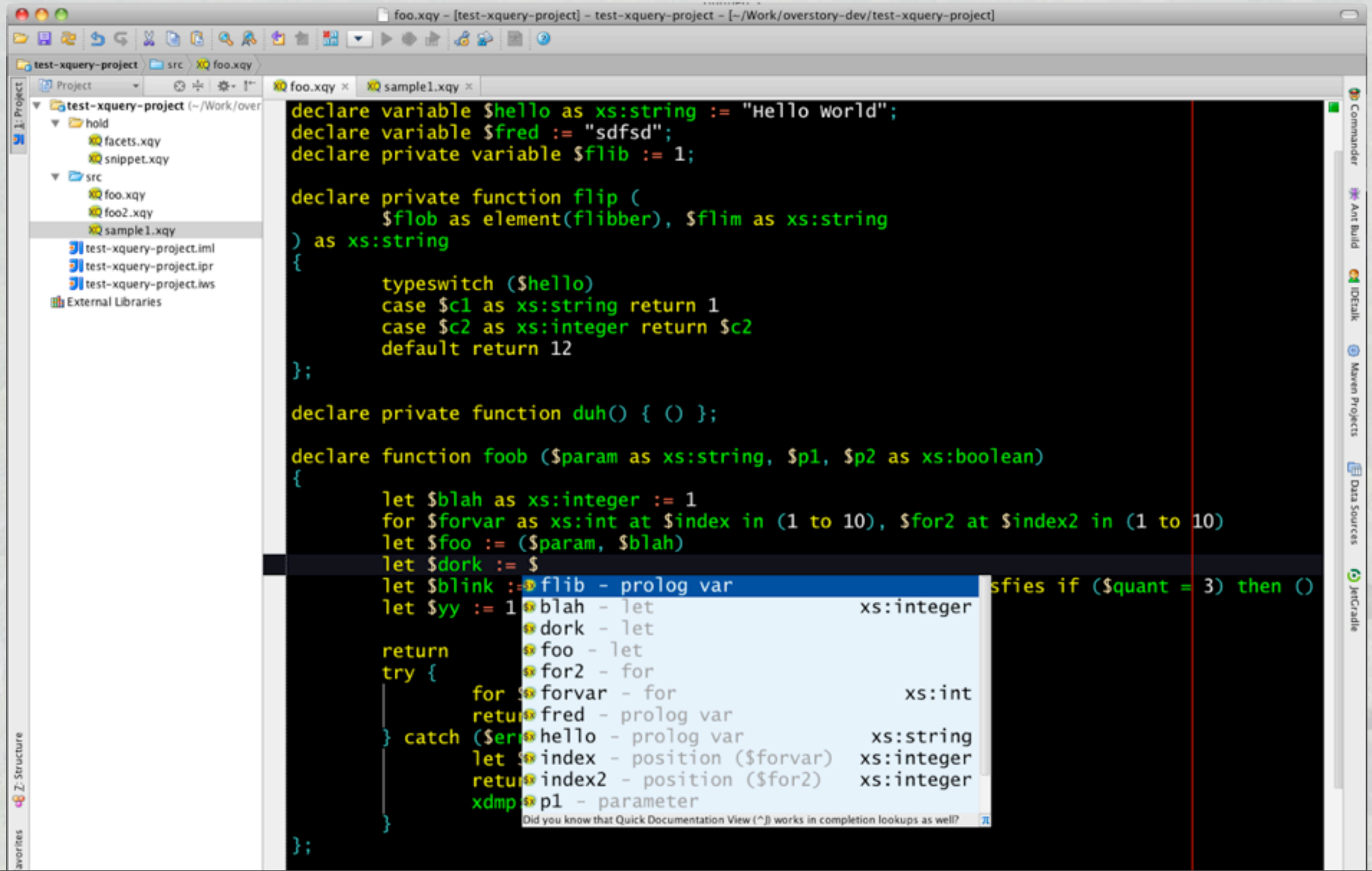
IntelliJ has always been my favorite Java IDE

Used it to create and test XCC

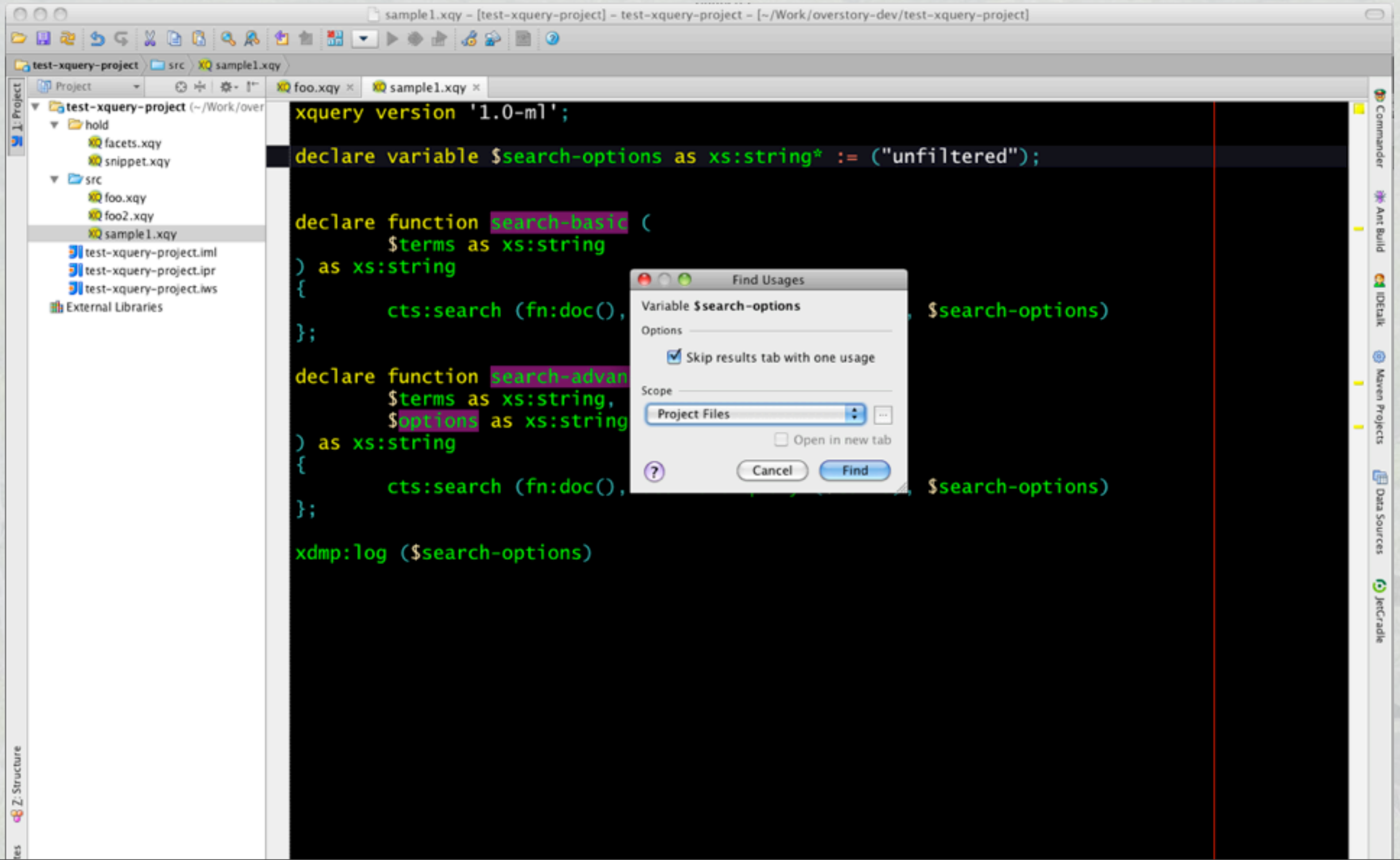
Always wished it could do for XQuery what it does for Java

Started a plugin while at MarkLogic, in 2005

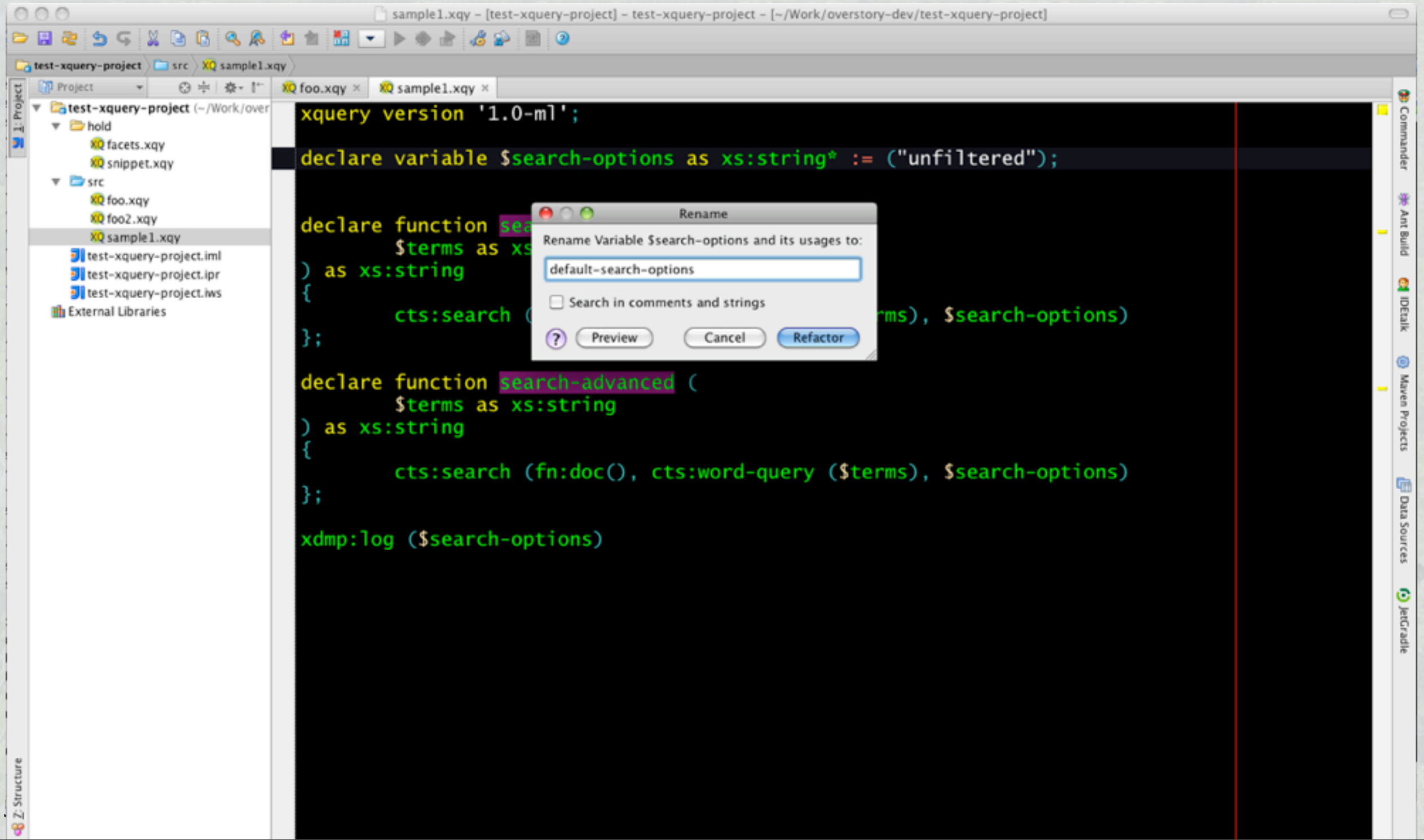
Wouldn’t This Be Nice?



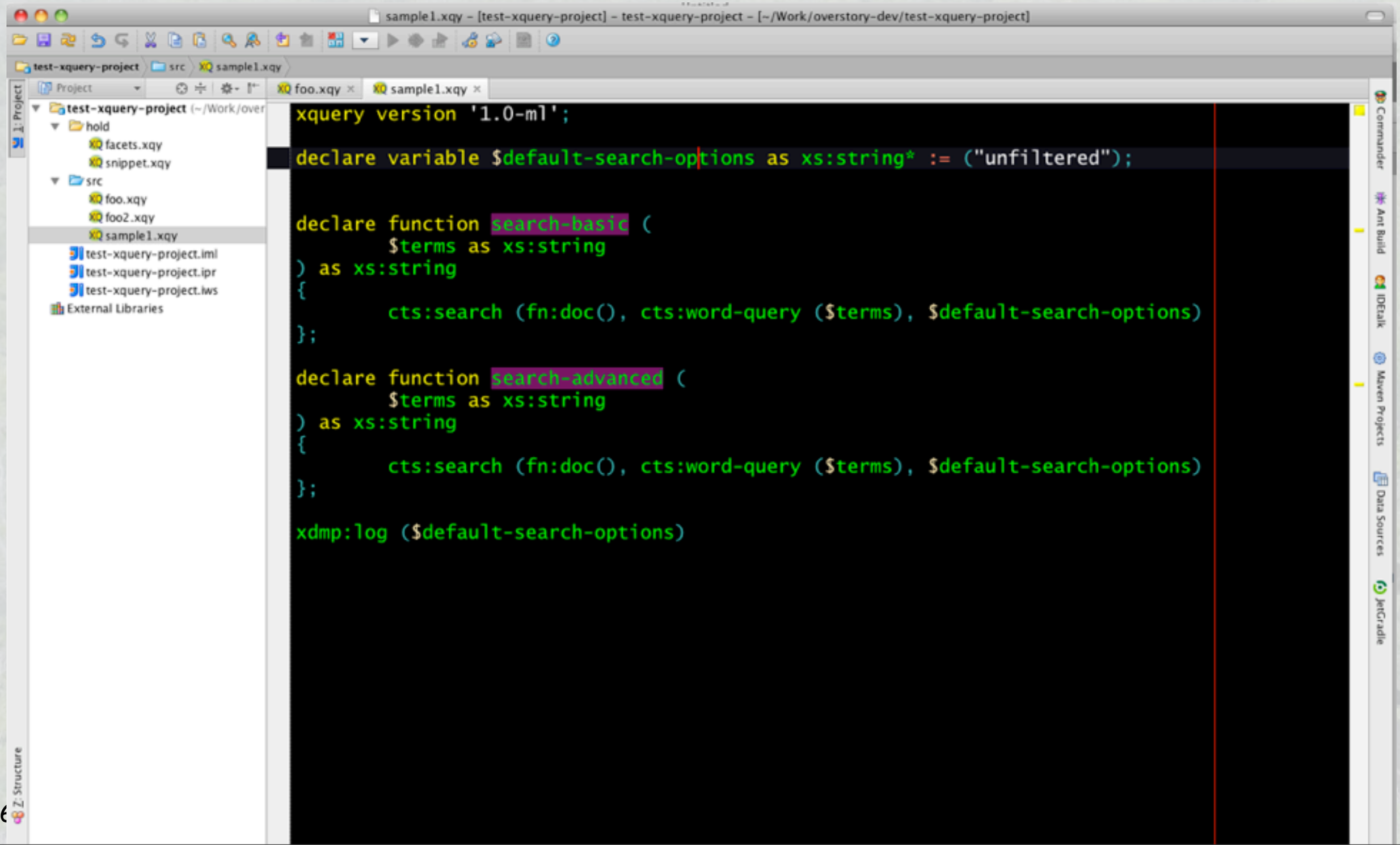
Where Is This Thing Used?



How About A Name Change



Variable Renamed



XQuery Language Plugin

- Hooks into IntelliJ's Program Structure Interface
  - XQuery lexer/parser builds an internal model
  - IntelliJ calls into the plugin to manipulate the model
  - The plugin also traverses the model
    - Validation, type-ahead suggest, usage checks, etc
    - Refactoring of various flavors
- Common editor, customized to your preferences
  - Switch easily between languages in one IDE

Current State

Basic Functionality is in Place

- XQuery-specific syntax highlighting
- XQuery & XML-specific bracket/brace matching
- Jump to definition, find usages for vars and functions
- Auto-suggest in-scope variables and functions
- Warn unused, error undefined vars and functions
- Rename variables and functions
- Comment/uncomment in the XQuery style

DEMO

Where It Stands Now

- Basic functionality is mostly there
  - Auto-suggest, API info, usages, rename, warnings, etc
  - Syntax-aware code traversal
- Parser/Lexer still rather brittle
  - Many features stop working with invalid XQuery
  - XML parsing still a bit flaky
- Not *quite* ready for prime time
  - Soon, I hope

Parser State

- Uses the Grammar Kit from JetBrains
  - Generates a parser from a BNF spec
  - Somewhat customizable, but poorly documented
  - Not very forgiving of malformed XQuery, yet
- Features that use the model stop working when the parser can’t build a complete model
  - Editor-based features, like brace matching, still work
- Parser improvement is the next priority

Intermediate Plans

- Contextual auto-complete, BNF-driven
  - Next legal XQuery construct at cursor, with look-ahead
- More and better annotations
  - Type mis-matches in assignments and parameters
  - Variables hiding other variables
  - Unused imports and namespace declarations,
  - Invalid values for options/parameters, where known
- Matching function parameters to signatures

Longer Term Plans

Resolve namespace values across modules

Contextual documentation

Code Intentions

Full refactoring

- Extract, introduce, etc

Warn about unresolvable import paths

Beyond Editing

- Execution from IntelliJ
  - Send request, capture result in an edit buffer
  - Jump to error source line for syntax exceptions
- Profiling
  - Capture history to plot code improvements
  - Correlate performance with code checkins
- Debugging
- Testing framework integration

Help Me!

If you want to contribute to the project, especially if you love tinkering with parsers & lexers, please contact me:

ron@overstory.co.uk

http://github.com/overstory

Questions?

Ron Hitchens

Principal Consultant, OverStory Ltd

ron@overstory.co.uk

Licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.
© 2012 OverStory Ltd