Groovy/Grails tooling in NetBeans

Matthias Schmidt
Software Engineer
Sun Microsystems, Inc.

Agenda

- Some Background.
- Structure of the Groovy/Grails support modules.
- What's possible at the moment.
- The Frameworks/API's used and dependencies.
- How to plug the Groovy compiler into NetBeans.
- Support for Grails with new project type.
- Future/Roadmap.

Background

- More and more Languages on the VM: Ruby, Groovy, Scala, Python, etc
- NetBeans > 6.5 : Ruby
- Retouche, GSF or how to stop re-inventing the wheel.
- NB 6.5: Groovy, JavaScript, PHP

Module structure

groovy.support - Collection of helper- and utility classes alongside the Customizers.

groovy.editor - Editing Groovy sourcecode files. All the bells and whistles like syntax-highlighting, folding resides here.

groovy.gsp - Support for editing Groovy server pages files (GSP).

groovy.grails - Here's all the code interfacing with an ex-process grails server. Architecture-specific execution of the "grails <target>".

groovy.grailsproject - Support for the new NetBeans project type "Grails-Project" together with all visual components like Wizards, Actions, Node-renderers and the like.

groovy.kit - Wrapper module for the ones above to show-up as a single entity in NetBeans PluginManager.

groovy.samples - Collection of Sample Applications.

groovy.refactoring - All high-level refactoring code goes here.

What's possible I - Groovy Editor

- Syntax Highlighting.
- Code Folding.
- Mark occurrences, instant rename.
- Goto declaration.
- Code completion.
- Navigator View.
- Error-marking
- Hints: Fix imports, surround with ...

What's possible II - Grails

- Seamlessly open existing Grails projects.
- Create new Grails projects using Wizards.
- Start/Stop/Monitor running Grails server.
- Grails-Server output redirected into output-pane.
- Creating Artifacts and run various grails commands according to context in tree-control.

Frameworks used

- Schliemann could not be used.
- Groovy Editor uses pretty much all of NetBeans infrastructure. Dependency to some 32 Libraries (Nodes, Filesystem, Lookup, Projects, etc.).
- GSF guinea pig. (General Scripting Framework)
- Besides dependency to Groovy, no other external dependencies.

How to plugin into Groovy compiler

- Ship Groovy as wrapped library in groovy.editor module: groovy-all-1.5.7.jar
- Run Groovy compiler to CLASS_GENERATION phase.
- Fetch resulting AST: compilationUnit.getAST();
- Try to sanitize the source, if it's broken. (in fact it's broken all the time)
- Use getErrorCollector() to mark errors.
- Work with Groovy-AST to provide all editor features.

Creating a new (Grails) project type

- No import of existing Grails apps, we simply open them.
- We store nothing in the Grails projects. We leave no traces. People can move on using other tools or IDE's.
- Netbeans "recognizes" Grails projects and displays it's contents using a new project type.
- Special LogicalViewProvider to have a customized tree-control and context-aware actions.
- Running servers are shown in the "Services" tab.
- We basically map NB's actions to "grails <cmd>"

The future

- Migrating to CSL and parsing & indexing API.
- Debugging.
- Better CodeCompletion incl. grails CC.
- Refactoring.

Links

- http://martin.adamek.sk
- http://www.netbeans.org
- http://blogs.sun.com/tor/
- http://hg.netbeans.org/main/summary
- http://hg.netbeans.org/main/contrib/summary
- http://wiki.netbeans.org/WorkingWithNetBeansSources
- http://grails.org/
- http://groovy.codehaus.org/

Thanks

Matthias Schmidt
schmidtm@sun.com