SpringSource Tool Suite 2.8.0.M1

- New and Noteworthy -

Martin Lippert 2.8.0.M1 August 11, 2011 Updated for 2.8.0.M1

www.springsource.com

springsource A division of vmware

ENHANCEMENTS – 2.8.0.M1

General Updates

vFabric tc Server 2.5.1

STS ships now with the latest vFabric tc Server Developer Edition 2.5.1.

Mylyn 3.6.1

The Mylyn version within STS is upgraded to the latest Mylyn release 3.6.1

Maven 3.0.3

The bundled Maven version got updated to Maven 3.0.3.

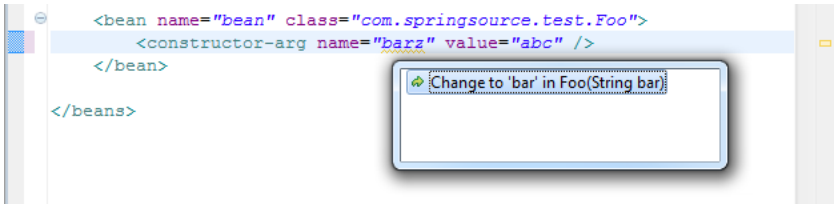
Spring Development Tools

JDK 1.7.0 and Spring Roo

There was a problem running Spring Roo on top of JDK 1.7.0. While this got fixed in Roo for the upcoming 1.2.0.M1 version of Roo, we managed to fix this within STS as well, so that you can now not only run Spring Roo 1.2.0.M1 on top of JDK 1.7.0, but also the latest Spring Roo 1.1.5 release.

As-you-type validation and quick fixes for constructor-arg

In this milestone we added as-you-type validation and quick fixes for the name attribute in a constructor-arg element. A warning (yellow) line appears under a name attribute that does not match with any constructor parameter names, and Ctrl+1 brings up a list of possible parameter names to rename to.



Grails Development Tools

Support for Grails 2.0.0.M1

The STS Grails tooling now supports Grails 2.0.0.M1. Note that Grails 1.4.M1 is no longer supported - users on Grails 1.4.M1 should upgrade to 2.0.0.M1. Note that in STS 2.8.0.M1 the option to "keep grails running" does not work for Grails 2.0 projects. If this option is turned

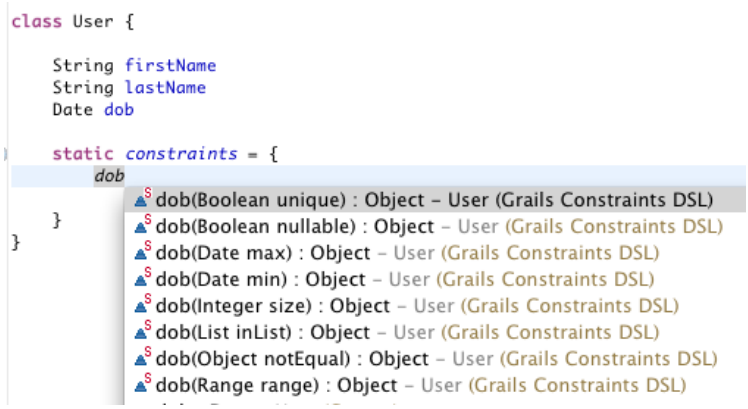
on it will be ignored when working with 2.0 projects. If this is something you have been using and would like to continue using, please vote on STS-1867.

Enhanced DSL support for Grails 2.0.0.M1

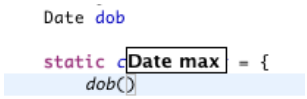
There is now a DSL descriptor shipped with Grails 2.0.0.M1. This provides extra editing support when working with Grails artifacts. For example:

Grails constraints DSL (STS-1184)

Full content assist for the standard constraints block:

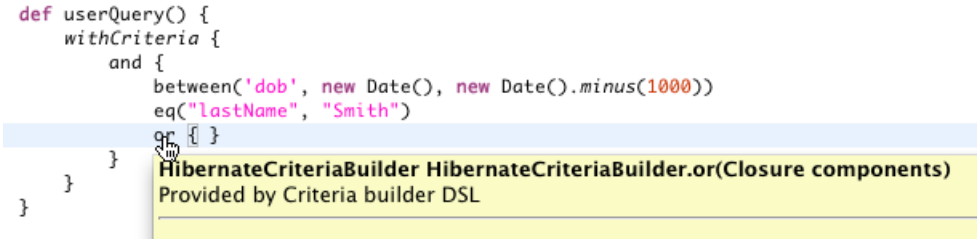


Invoking content assist inside of a constraint method, you can see the method context information displayed:



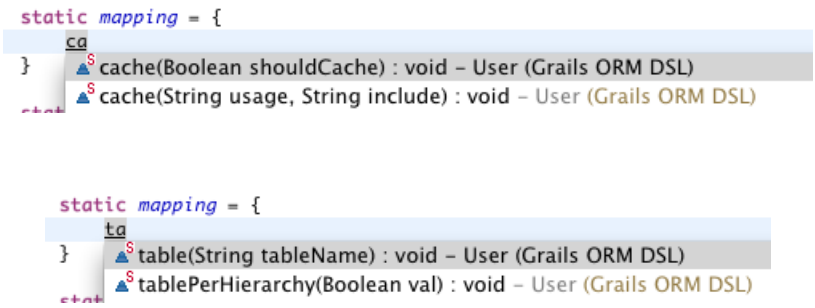
GORM criteria query support (STS-580)

There is now editing support for the criteria query builder DSL in the Groovy Editor:



Grails ORM DSL support

There is now full support for the Grails ORM DSL in the static mapping field of a domain class:



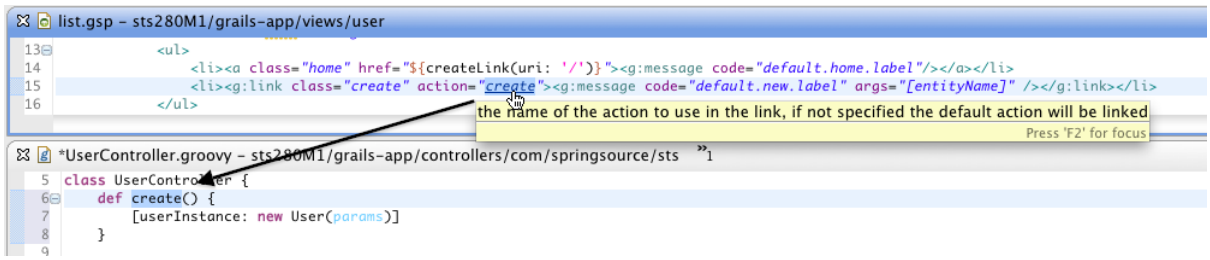
Enhanced dynamic finder support (STS-1982)

STS now has more complete support for dynamic finders.

GSP support (STS-1498)

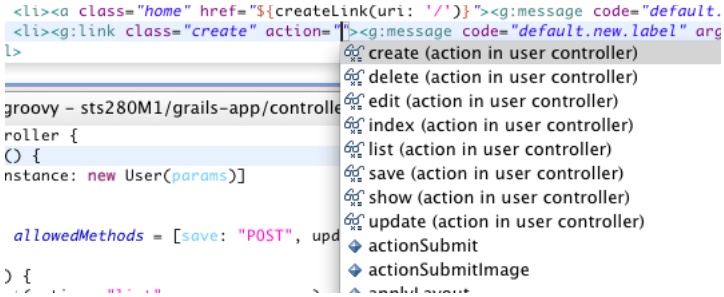
Navigation to referenced controller or action in g:link tag

Pressing F3 or Ctrl-hover over an action/controller in a g:link tag will navigate you to the definition of that action/controller:



Content assist in g:link tags

There is now content assist for relevant controllers and actions in g:link tags:



Groovy-Eclipse

Better DGM inferencing

Default Groovy Methods now have better inferencing inside of associated closure blocks. For example, the unique DGM method now supports inferencing of the parameter on its closure:

```
class Simple { int val }
[new Simple(), new Simple()].unique {
    it.val
}
```

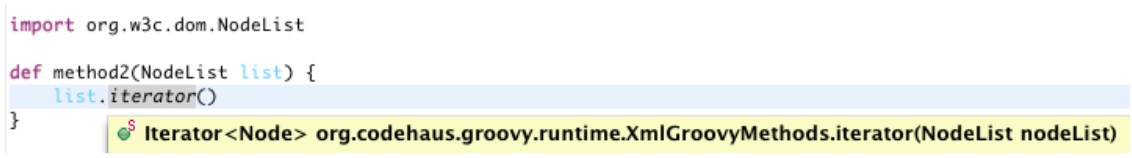
In addition to unique all methods inside the DefaultGroovyMethods class now support inferencing on parameters in closures where possible. A complete list is available in:

http://jira.codehaus.org/browse/GRECLIPSE-1143.

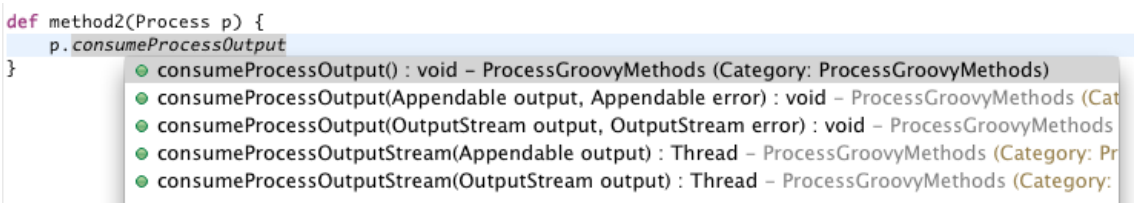
Support for DateGroovyMethods, SwingGroovyMethods, XmlGroovyMethods, and ProcessGroovyMethods

These classes are now handled just like DefaultGroovyMethods in terms of content assist, hovers, and navigation. For example:

XmlGroovyMethods:



ProcessGroovyMethods:



For more information, see GRECLIPSE-1131, GRECLIPSE-1143, GRECLIPSE-1145, GRECLIPSE-1153, GRECLIPSE-1154 and GRECLIPSE-1155.

Multiple assignment statements

The types of variables assigned in multi-assignment statements are discovered during inferencing. For example, when assigning a list to multiple variables, the static type of the list elements will be assigned to each variable:

```
class Simple { int val }
class Simple2 { int val2 }

def (x, y) = [new Simple(), new Simple2()]
x.val
y.val2
```

Also, when a method returns a list or array, the type parameter or component type is used as the type of the assigned variable:

```
class Simple { int val }

List<Simple> method () {
    return [new Simple(), new Simple()]
}

def (x, y) = method()
x.val
y.val
```

Match operator inferencing

Match operator expressions now support inferencing:

```
("" =~ /pattern/).hasGroup()
("" ==~ /pattern/).booleanValue()
```

For more information, see GRECLIPSE-1159.

Method context information

Groovy-Eclipse now follows the JDT model of displaying context information for methods when invoking content assist inside of a method call and there is no prefix.

For example, when invoking content assist just after an opening paren, you are only shown a list of known ways to invoke the target method:

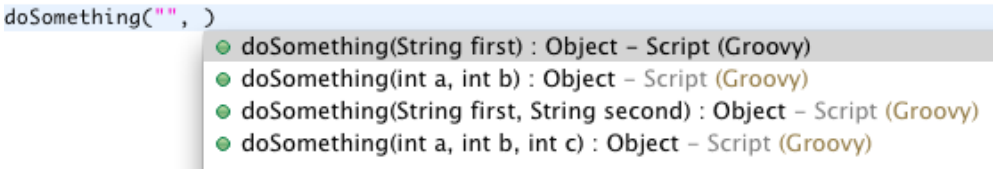
```
def doSomething(int a, int b, int c = 9) { }
def doSomething(String first, String second = "") { }

doSomething()
```

```
doSomething(String first) : Object - Script (Groovy)
doSomething(int a, int b) : Object - Script (Groovy)
doSomething(String first, String second) : Object - Script (Groovy)
doSomething(int a, int b, int c) : Object - Script (Groovy)
```

Selecting one of the proposals will show that methods information in tooltips above the caret location:

```
                String first, String second
doSomething()
```

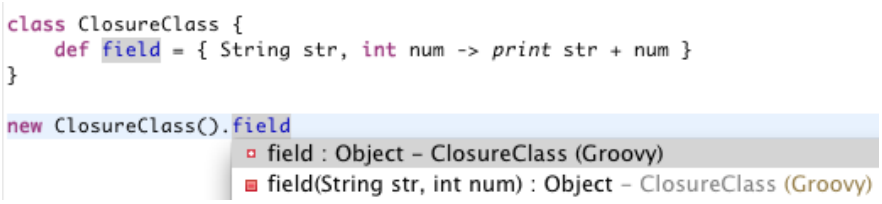
Similarly, invoking content assist after a comma will show context information as well:



For more information, see GRECLIPSE-674.

Content assist now recognizes closure parameters

When a field has a closure for an initializer, content assist will now reflect this and show method proposal variants for the field:



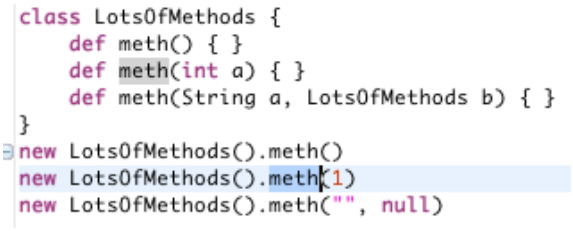
For more information, see GRECLIPSE-1139.

Fix import statements after move/copy

Import statements are now properly cleaned up after a move or copy of a Groovy file to a new location. See GRECLIPSE-682 for more information.

More precise searching for overloaded methods

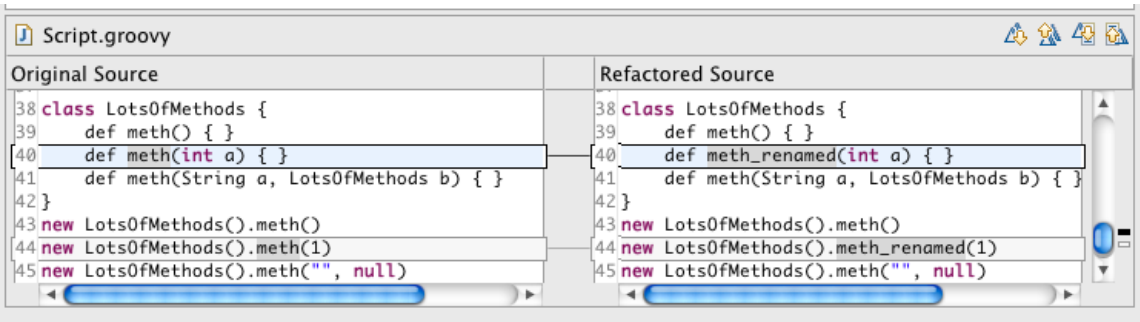
The number of parameters of method declarations are now used to help more precisely determine the actual declaration of a method reference:



See GRECLIPSE-1138 for more information.

More precise rename refactoring of overloaded methods

This new technique for method searching is used to ensure that overloaded methods do not interfere with rename refactoring:



Compiler switching through system property

There is an experimental way to use multiple Groovy compiler levels with one STS or Eclipse installation. You can add the groovy.compiler.level system property to the launch command to control which compiler level is started. Use:

```
-vmargs -Dgroovy.compiler.level=18
```

or

```
-vmargs -Dgroovy.compiler.level=17
```

To start your workspace with 1.8 or 1.7 respectively. A word of caution, however. specifying vmargs on the command line will cause the launcher to ignore vmargs specified in the eclipse.ini or STS.ini. So, if this option is used, be sure to use appropriate memory settings in your command line, such as this:

```
STS -data "/users/myself/Path/To/Workspace" -vmargs -Xmx1024M
-XX:PermSize=64M -XX:MaxPermSize=256M -Dgroovy.compiler.level=18
```

or

```
STS -data "/users/myself/Path/To/Workspace" -vmargs -Xmx1024M
-XX:PermSize=64M -XX:MaxPermSize=256M -Dgroovy.compiler.level=17
```

See STS-1844 for more information.

Editing support for Groovy files outside of the build path

Scripts that are not placed on the build path now provide many of the standard editing features that is used by scripts on the build path, such as content assist, type inferencing, and mark occurrences. The classpath for the script is assumed to be the same build path as the one for the rest of the project.

Note that missing import references will not be reported, and references in scripts will not show in search results, nor will refactoring work in these files.

AspectJ/AJDT

AJDT now includes a new development build of AspectJ 1.6.12. In this build two options have been turned to ON by default (minimalModel and typeDemotion). Previously these options defaulted to OFF. These options should lead to a reduction in memory footprint for AspectJ projects (e.g. Roo projects) in your workspace.

If you have any issues after upgrading, the settings can be turned off by specifying –Xset:minimalModel=false,typeDemotion=false in the project properties (in non-standard compiler options).

Fixed Bugs and Enhancement Requests

Here is a full list of resolved bugs and enhancement requests for the 2.8.0.M1 release:

https://issuetracker.springsource.com/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+STS+AND+fixVersion+%3D+11184+AND+status+in+%28Resolved%2C+Closed%29