

SpringSource Tool Suite 2.7.2

- New and Noteworthy -

Martin Lippert

2.7.2

September 14, 2011

Updated for 2.7.2.RELEASE

ENHANCEMENTS – 2.7.2

General Updates

vFabric tc Server 2.6

STS now ships with the latest vFabric tc Server Developer Edition 2.6 and is compatible with those just released tc Server versions.

Spring Roo 1.2.0.M1

STS is now compatible with the latest Spring Roo 1.2.0 milestone builds. STS still ships the latest Roo release (which is 1.1.5 at the moment), but you can add Roo 1.2.0.M1 to the available Roo runtimes and use it directly from within STS.

Mylyn 3.6.2

The packaged Mylyn version of STS is now 3.6.2

ENHANCEMENTS – 2.7.1

General Updates

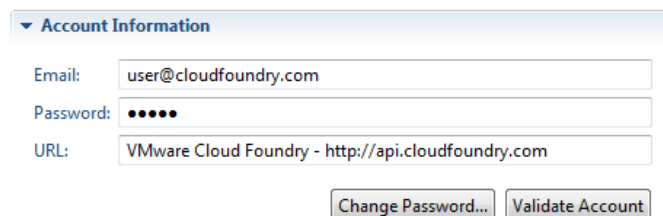
Spring Roo 1.1.5

STS now ships and works with the just released version 1.1.5 of Spring Roo.

Spring Development Tools

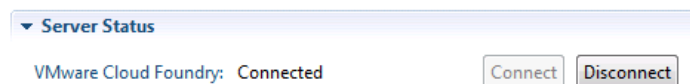
Cloud Foundry Support

Since the last release we have built on top of the Cloud Foundry integration by adding the functionality to change your Cloud Foundry account password from within STS.



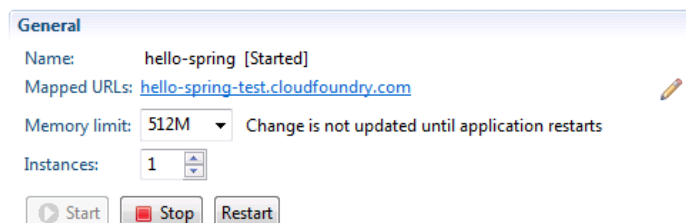
The screenshot shows the 'Account Information' dialog box. It contains three input fields: 'Email' with the value 'user@cloudfoundry.com', 'Password' with masked characters '•••••', and 'URL' with the value 'VMware Cloud Foundry - http://api.cloudfoundry.com'. Below the fields are two buttons: 'Change Password...' and 'Validate Account'.

We have also added a server status section in the Cloud Foundry server editor to display whether it is connected to your Cloud Foundry account and you can connect or disconnect from the editor.



The screenshot shows the 'Server Status' section. It displays 'VMware Cloud Foundry: Connected' and two buttons: 'Connect' and 'Disconnect'.

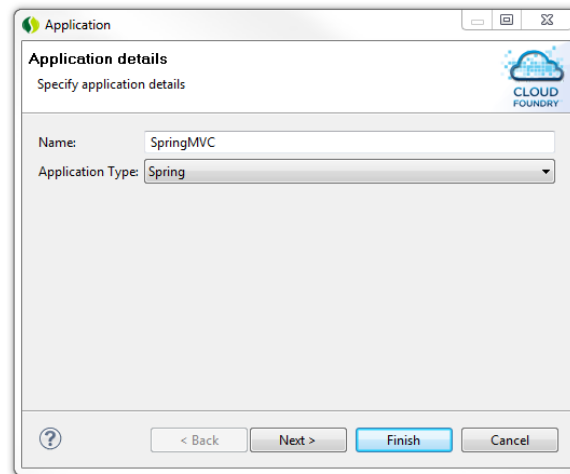
It is now also possible to change memory allocation to an application once it is deployed. Note that it does not take effect till the application is restarted.



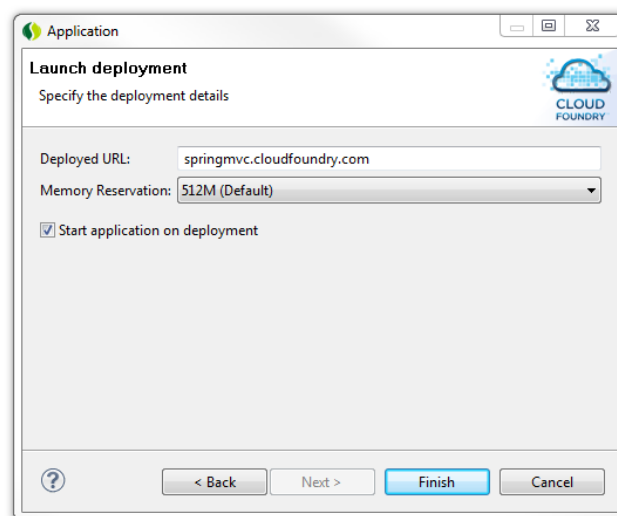
The screenshot shows the 'General' settings for an application. It includes fields for 'Name' (hello-spring [Started]), 'Mapped URLs' (hello-spring-test.cloudfoundry.com), 'Memory limit' (512M), and 'Instances' (1). A note states 'Change is not updated until application restarts'. At the bottom are three buttons: 'Start', 'Stop', and 'Restart'.

Cloud Foundry deployment mode

We made the Cloud Foundry server more consistent with Cloud Foundry by showing always the applications that are deployed in Cloud Foundry within STS. Therefore we deploy an application right after it is added to the Cloud Foundry server. Previously an application is not deployed until the first time it is started. In this release the application deployment wizard pops up after an application is added.



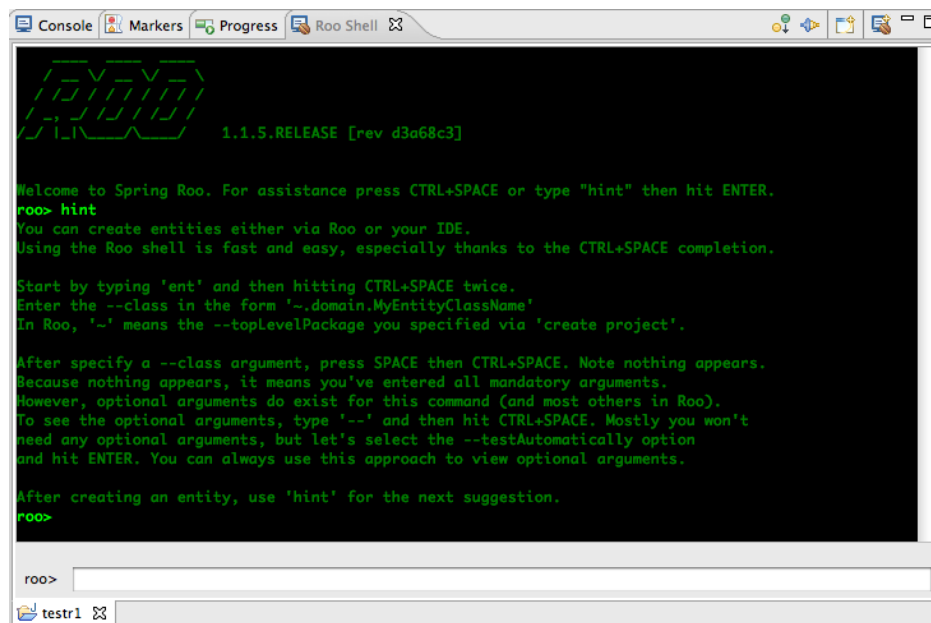
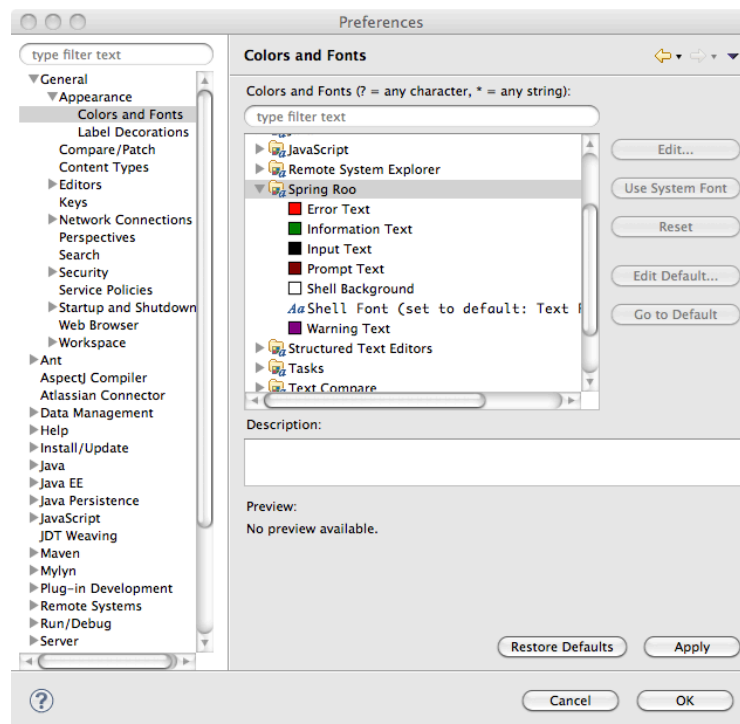
By default an application is started on deployment but you can deselect this option in the deployment wizard. This can be useful if services need to be configured before an application can run properly. You can deselect the "Start application on deployment" and configure services in the Cloud Foundry server editor before starting the application.



Another way we are making Cloud Foundry integration in STS more consistent with Cloud Foundry is applications are now removed from the Cloud Foundry server if they are deleted outside of STS. Previously applications that are deployed within STS are still shown (with a status of undeployed) under the Cloud Foundry server even if they are deleted from outside STS. This seems to lead to a bit of confusion as the list of applications under the server do not match the ones that are actually deployed. In this release such an application will be removed from Cloud Foundry server. If you wish to deploy this application again, simply drag and drop onto the Cloud Foundry server or into the server editor to deploy to Cloud Foundry again.

Roo Shell colors now customizable

We added support to customize the colors and fonts that are used by the Roo Shell.



ENHANCEMENTS – 2.7.0

General Updates

Eclipse Indigo (3.7)

STS now ships by default on top of the just released Eclipse Indigo (3.7.0) release. Eclipse Helios is still supported via the appropriate update site.

With this release, we drop support for Eclipse 3.5, aka Galileo.

Mylyn 3.6

The Mylyn version within STS is upgraded to the latest Mylyn release 3.6.

EGit 1.0

STS now comes with EGit being installed by default. No need to install additional support, just start using Git right from within STS.

Scala IDE 2.0.0 beta

We added the latest beta builds of the Scala IDE to the list of possible extensions to STS to ease the use and installation of Scala for your projects. This plays nicely together with the Cloud Foundry integration within STS now supporting the direct deployment of Lift applications into Cloud Foundry.

Since the Scala IDE is still in beta status, we marked this extension as “experimental”.

Memory Settings

We raised the default memory settings for the PermGen when running STS slightly from a max. of 256m to 384m. If you would like to use a different value or increase the heap size, you can easily tweak them in your sts.ini file of your STS installation.

Spring Development Tools

Updated template projects

STS 2.7.0 comes with updated Spring Template Projects (go to the File > New > Spring Template Project) that you can use to bootstrap your development with the various features of the Spring frameworks. These templates have been refreshed to reflect newer versions of the Spring projects, making it a snap to build current, correctly configured applications.

In addition to update the Spring templates, we cleaned up the list of template projects and removed the OSGi related ones from that list. The OSGi tooling support continues as part of the Eclipse Virgo project.

dm Server Tools

Contribution to Eclipse Virgo

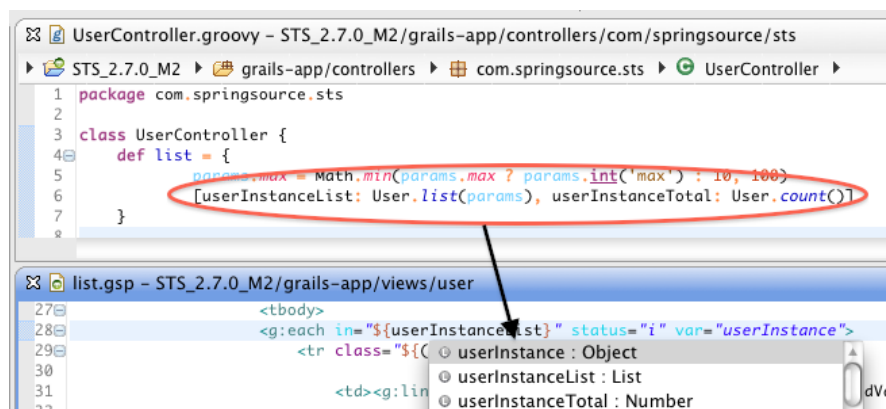
This release of STS is the last one that ships a new build of the dm Server tooling. If you are already using Eclipse Virgo as your runtime environment for enterprise OSGi development, you should consider to move to the Eclipse Virgo IDE tooling (which basically is the contributed version of the dm Server tooling from STS) in the future.

If you use the tooling together with dm Server 1.0 or dm Server 2.0, you will be able to install this last version of the dm Server tooling as an optional component.

Grails Development Tools

GSP content assist now recognizes arguments coming from controller actions

In the screenshot below, you can see that the controller action (list) returns a map of two elements. The map values are available as variables in the associated GSP. You can see that in the content assist list. Also, STS makes a best effort to infer the types of all of these arguments.



GSP content assist now recognizes empty tags

As described here (<http://grails.org/Doc+Comments+for+GSP+tags>), it is now possible to specify that your custom GSP tags should be 'empty' when being applied in content assist. This means that your tag will be filled in like this:

```
<g:myTag />
```

instead of like this:

```
<g:myTag></g:myTag>
```

To do so, simply create a javadoc for your custom tag and ensure that the `@emptyTag` attribute is somewhere in the Javadoc on its own line.

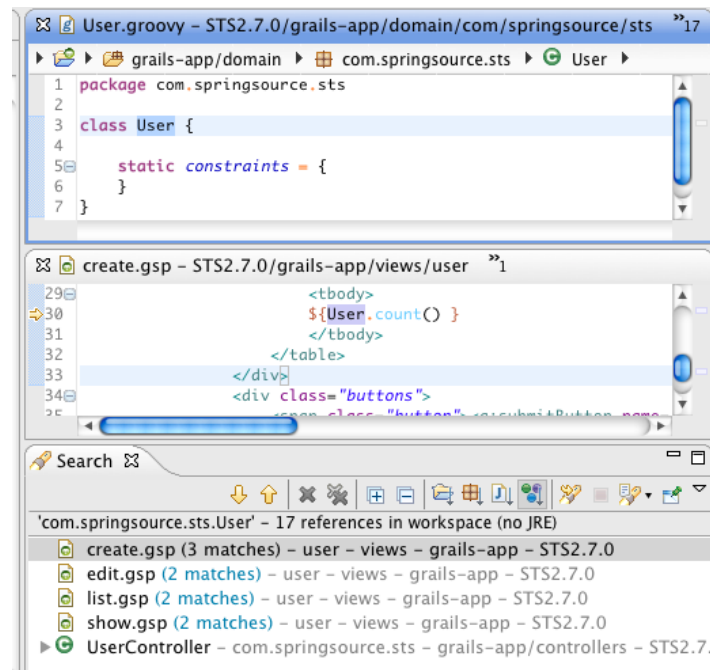
Grails 1.4-M2 will be taking advantage of this for all of its standard tags (eg- `def`, `set`, etc).

Improved GSP content assist inside of scriptlets

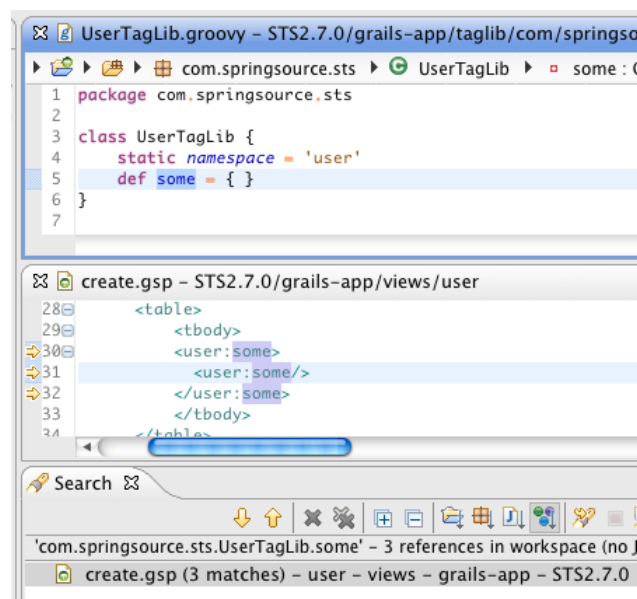
We have made many small enhancements to content assist in GSPs, especially when inside of groovy scriptlets. You will have a much smoother experience when editing GSP files.

GSP Search

The Eclipse search engine will now look inside GSP files for search results. References inside of scriptlets will be found:

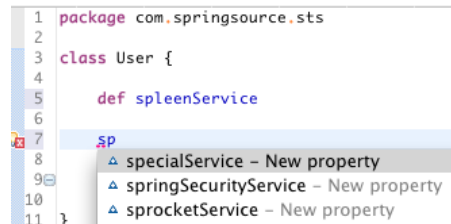


Also, references to tags will be found:



Service field declaration content assist

Content assist inside of Grails artifacts at a location where a new field can be declared will now display all available services as proposals:



```

1 package com.springsource.sts
2
3 class User {
4
5     def spleenService
6
7     .sp
8     ▲ specialService – New property
9     ▲ springSecurityService – New property
10    ▲ sprocketService – New property
11 }
  
```

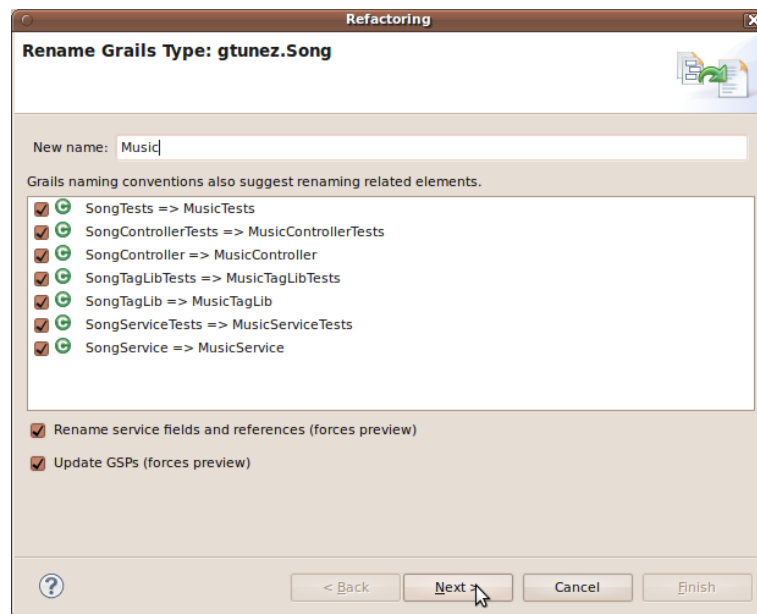
Grails 1.4 Support

Greclipse and STS Grails tooling can now work with Grails 1.4.M1.

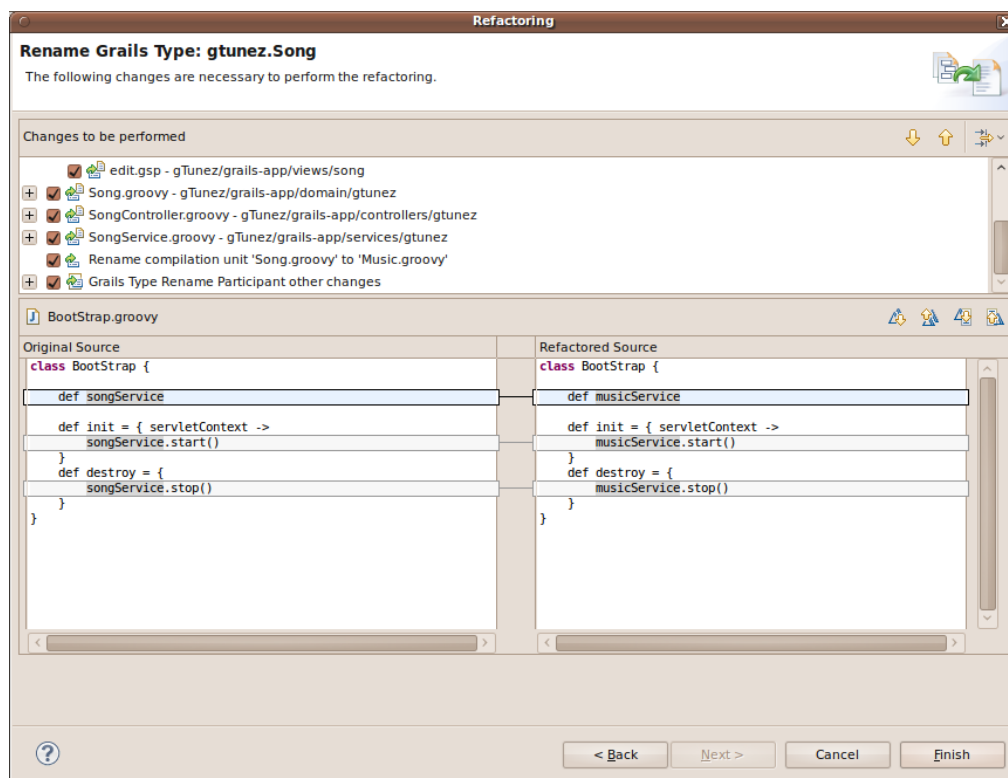
Grails Aware Rename Type Refactoring

When a rename type refactoring is invoked from the Groovy Editor or the Project Explorer in the Grails perspective the refactoring will be 'Grails Aware'. A Grails specific rename type wizard will popup allowing the user to enable or disable Grails specific support:

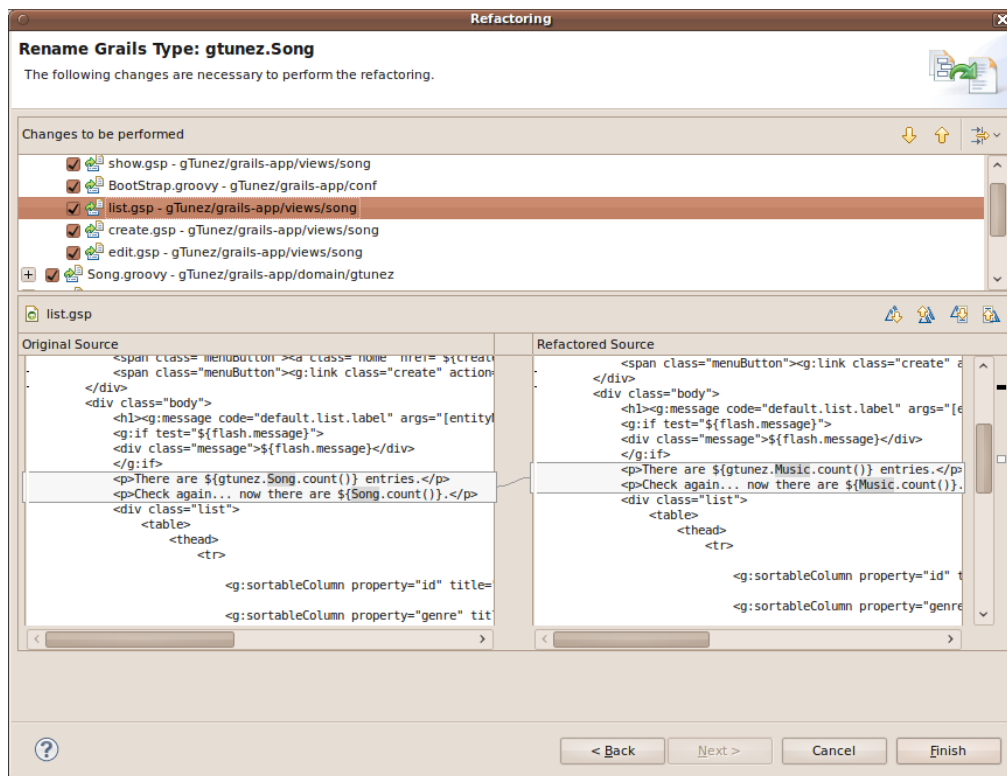
- Grails Type Awareness:
 - Rename related types (for example when renaming a Domain class also rename associated Controller, Test and Service classes).
- Service Awareness:
 - Updating autowired service fields when a Service class is being renamed
- GSP Awareness:
 - Updating references to the renamed type(s) inside of GSP files.
 - Renaming the associated views folder when a Controller class is being renamed.



Grails Rename Type Wizard Page One



Grails Rename Type – Preview



Grails Rename Type – GSPs

Groovy-Eclipse

Groovy-Eclipse 2.5.1 is now available and is installable from the extensions page on the dashboard. See:

<http://groovy.codehaus.org/Groovy-Eclipse+2.5.1+New+and+Noteworthy>

for the 2.5.1 New & Noteworthy and

<http://groovy.codehaus.org/Groovy-Eclipse+2.5.0+New+and+Noteworthy>

for the 2.5.0 New & Noteworthy.

Groovy-Eclipse Compiler

The 2.5.1 release of the groovy-eclipse-compiler plugin for Maven is now available. This allows you to compile your groovy-maven projects using Groovy-Eclipse outside of the IDE. Using this compiler plugin avoids many of the stub generation issues of GMaven and also compiles your projects more quickly.

For documentation on how to use the groovy-eclipse-compiler, see:

<http://groovy.codehaus.org/Groovy-Eclipse+compiler+plugin+for+Maven>

This page also includes a link to an archetype project that can get you started quickly with using the compiler plugin.

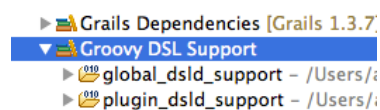
Global location for DSLDs

Groovy-Eclipse now supports a global location to place your DSLDs (see

<http://blog.springsource.com/2011/05/08/better-dsl-support-in-groovy-eclipse/>

for a description of DSLDs).

You will notice that there is a new classpath container on all of your Groovy/Grails projects:



This classpath container contains all of the global and plugin DSLDs for your projects.

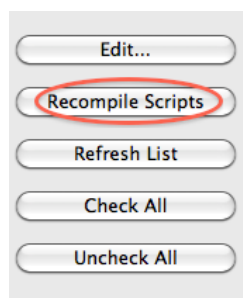
If you want a DSLD file to be applicable to all of your Groovy projects in the workspace, then place your global DSLDs in the `~/.groovy/greclipse/global_dsl_support/dsl` directory. A package statement of `'package dsl'` is required. These DSLDs will appear in the `global_dsl_support` class folder.

DSLDS that are specific to a single version of Groovy-Eclipse are located in the `plugin_dsl_support` class folder.

Note: Eclipse 3.6 and earlier suffer from

https://bugs.eclipse.org/bugs/show_bug.cgi?id=305172

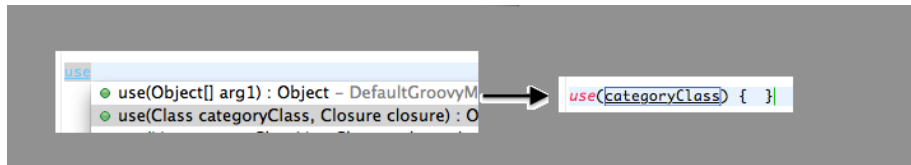
where these external classfolders will not be updated automatically. Instead, every time that you make a change to the `global_dsl_support` folder, you should go to the DSLD preferences page (Preferences -> Groovy -> DSLD) and click "Recompile Scripts":



In the coming weeks, we will be supplying several default DSLD files to handle the common builders and AST transforms inside of Groovy-Eclipse (such as Swing Builder, and HTML builder). These DSLDs will appear in the `plugin_dsl_support` folder.

Groovier way of handling closure arguments in content assist

When applying a content assist method proposal that has a closure as its last argument, Groovy-Eclipse will now automatically use the standard Groovy idiom of placing the closure expression outside of the parens. See, for example this screenshot:

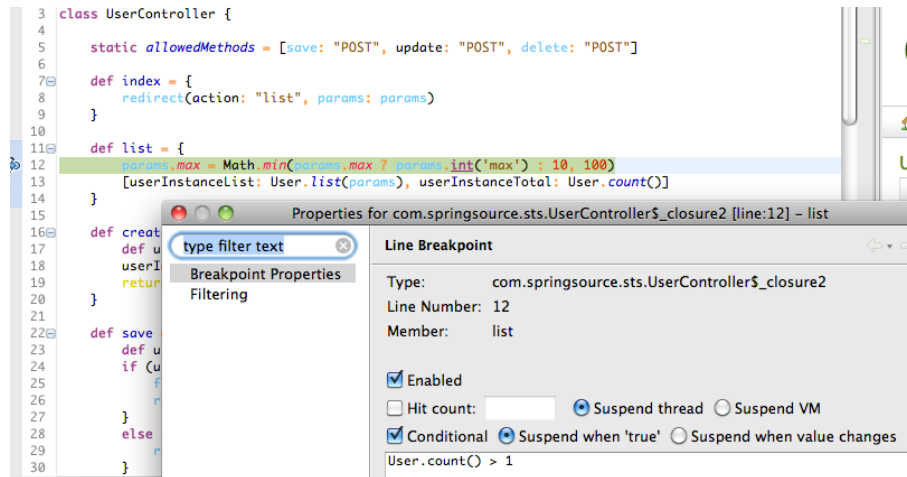


Support for `$/ /$` multiline strings

Groovy-eclipse now recognizes `$/ /$` style strings that have been introduced in Groovy 1.8.

Conditional breakpoints for Groovy code

Conditional breakpoints are now supported in Groovy files using groovy syntax:



This is built on top of the same infrastructure that the Groovy debug support uses. So, enhanced debug support must be activated and JDT Weaving must be enabled. See the blog post about Groovy Debug support in STS for more information:

<http://blog.springsource.com/2010/11/30/new-groovy-debug-support-in-sts-2-5-1/>.

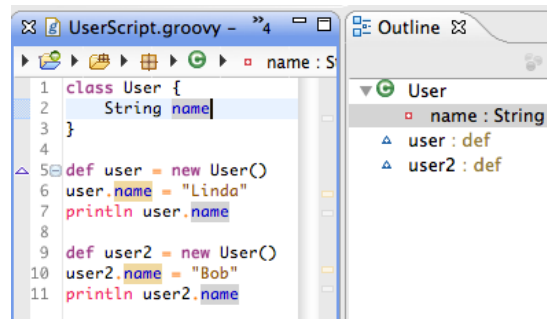
Groovy 1.8 support

Support for the Groovy 1.8 compiler is available. It will not be installed by default with the rest of Groovy-Eclipse. Instead, it is available as a separate feature that can be installed from the Groovy-Eclipse update site. For information on how to do this, see [Compiler Switching in Groovy-Eclipse](#):

<http://groovy.codehaus.org/Compiler+Switching+within+Groovy-Eclipse>

Script outline view

When working with scripts, the outline view displays contents in a more Groovy-like way. The run, and main methods are hidden. Instead, script variable declarations are displayed.

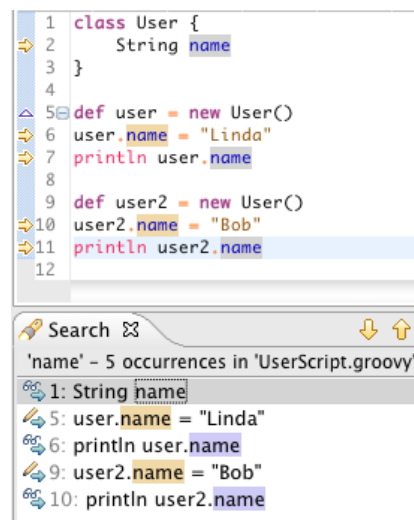


The enhanced outline view is part of Groovy-Eclipse and is available for third party plugins to use their own outline extender. For more information, see the feature request here:

GRECLIPSE-1029: <http://jira.codehaus.org/browse/GRECLIPSE-1029>

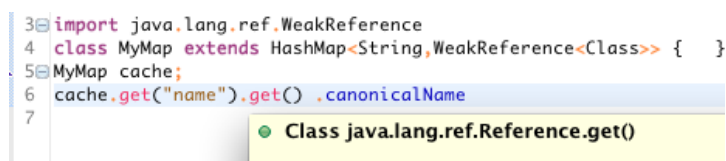
Read access and Write access differentiation when searching

When doing a find references in the Groovy editor (CTRL-Shift-U on Windows/Linux), read occurrences are highlighted differently from write occurrences:



Better inferencing for Generics

Groovy-Eclipse now recognizes more complex forms of generics to help seed type inferencing. For example, in the following code, Groovy-Eclipse can determine that the second call to "get()" is declared by the WeakReference class and returns a java.lang.Class object:



For more information, see GRECLIPSE-997: <http://jira.codehaus.org/browse/GRECLIPSE-997>

Gradle

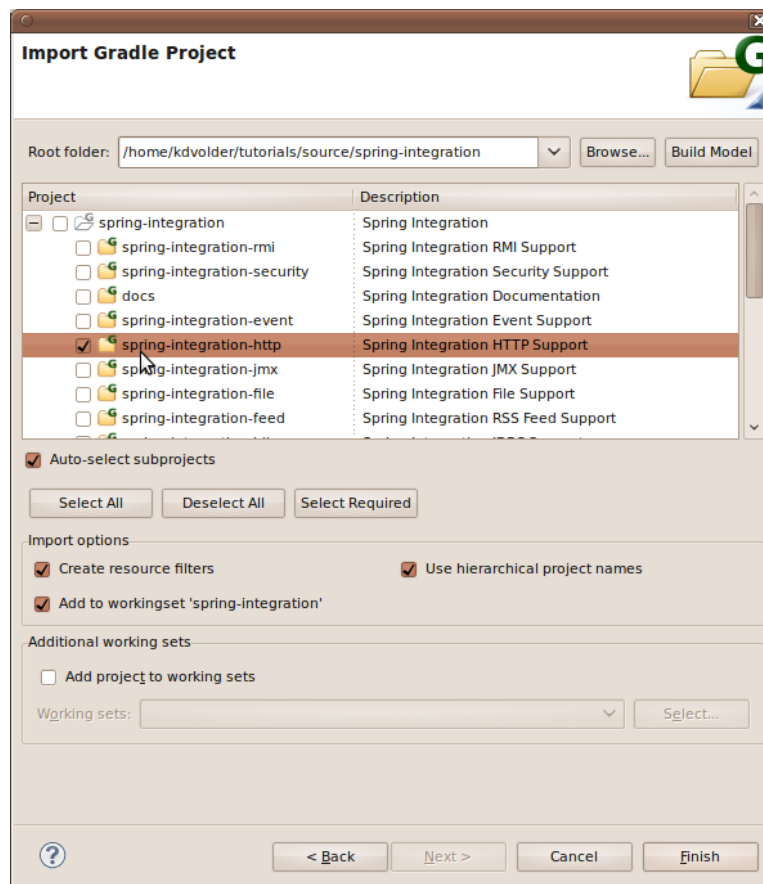
STS 2.7.0 includes an initial, basic set of features to support working with Gradle projects inside of STS. The following list of features are provided:

First set of features

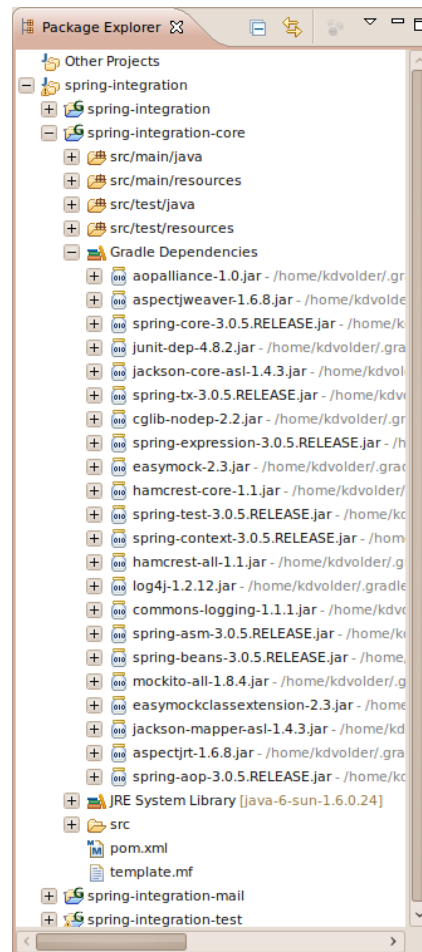
- Supports working with nested Gradle "multi projects":
 - mapping hierarchical structure onto a flat set of "linked" Eclipse projects with appropriate automatically configured resource filters.
 - flexible name mapping, allows user to adopt two different default name mapping schemes and also rename individual projects in the workspace without breaking the tools.
- Gradle Import Wizard:
 - ease importing existing Gradle (Multi) Projects with a number of subprojects.
 - ease the importing of a subset of "interesting projects", without breaking project dependencies.
 - allow incrementally importing additional projects after initially importing only a subset.
- Manage Gradle dependencies: (so projects can "build/compile" using the Eclipse JDT tools)
 - Classpath container that manages Gradle Jar dependencies on the Eclipse classpath
 - configure source folders automatically
 - configure project dependencies automatically
- Task Execution UI allows users to...
 - create, store, edit, manage and execute lists of Gradle tasks as Eclipse launch configurations from within STS.
 - Since this leverages the standard Eclipse launching framework, you can also
 - relaunch from a list of favorites
 - relaunch from a "recent history" list
 - store launch configurations in files that can be shared via a code repository like SVN, CVS or git.

For more information see the documentation:

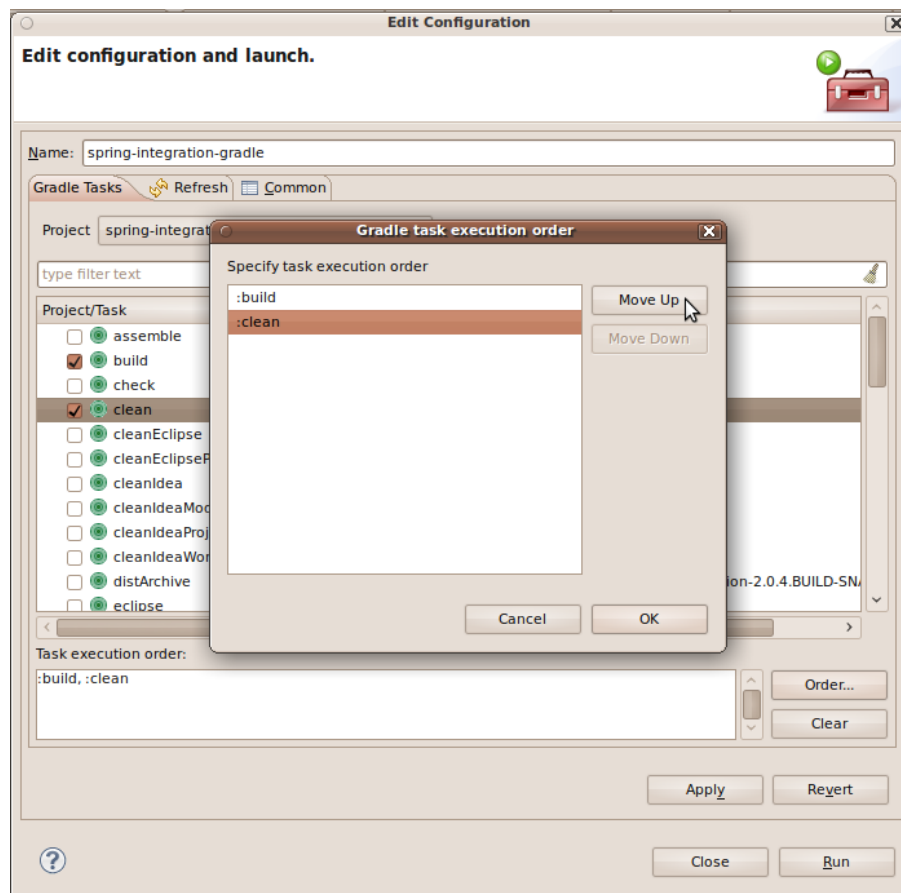
<http://static.springsource.org/sts/docs/2.7.0/reference/html/gradle/index.html>



Gradle Import Wizard



Gradle Classpath Container



Gradle Launch Configuration Editor

Fixed Bugs and Enhancement Requests

Here is a full list of resolved bugs and enhancement requests for the 2.7.0 release:

<https://issuetracker.springsource.com/secure/IssueNavigator.jspa?reset=true&jqlQuery=project+%3D+STS+AND+fixVersion+in+%2811205%2C+11001%2C+11000%2C+10999%2C+10998%29+AND+status+in+%28Resolved%2C+Closed%29>