

Using the Swordfish Git repository with Eclipse

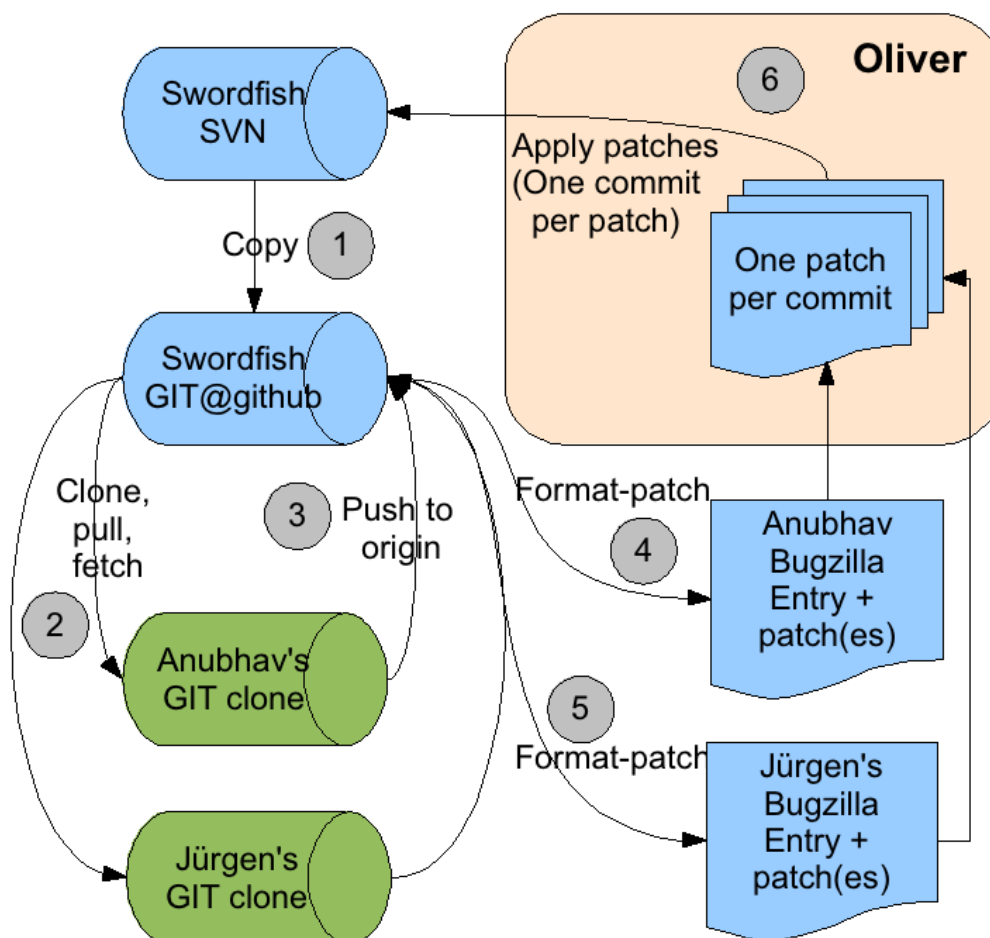
Motivation

Quite a few of the current contributors to the Swordfish project do not have committer status in the Swordfish Eclipse project. It was important to find a way to let them contribute in a controlled way in a repository that is available to the public. This repository should be

- writable for anyone who want to collaborate
- easy to synchronize with the official Eclipse SVN repository
- reliably about the creation of patches (we had problems with patches created / applied with Eclipse based on the SVN repository)
- allow for a CI build to ensure consistent & stable patches

Solution

The idea is to use a free public repository hosted at github.com that is mirrored from the Eclipse SVN. Collaborators can either fork this and create patches or (in case of Sopera employees and partners) get direct access to the mirror repository.



Prerequisites

To give a short checklist, this is what you need to use Git:

- The Git command line tools
- The EGit Eclipse plugin
- An account with a public key at <http://www.github.com>
- To be added as collaborator at Olivers repository at <http://github.com/owolf/eclipse-swordfish.git>

How to use Git

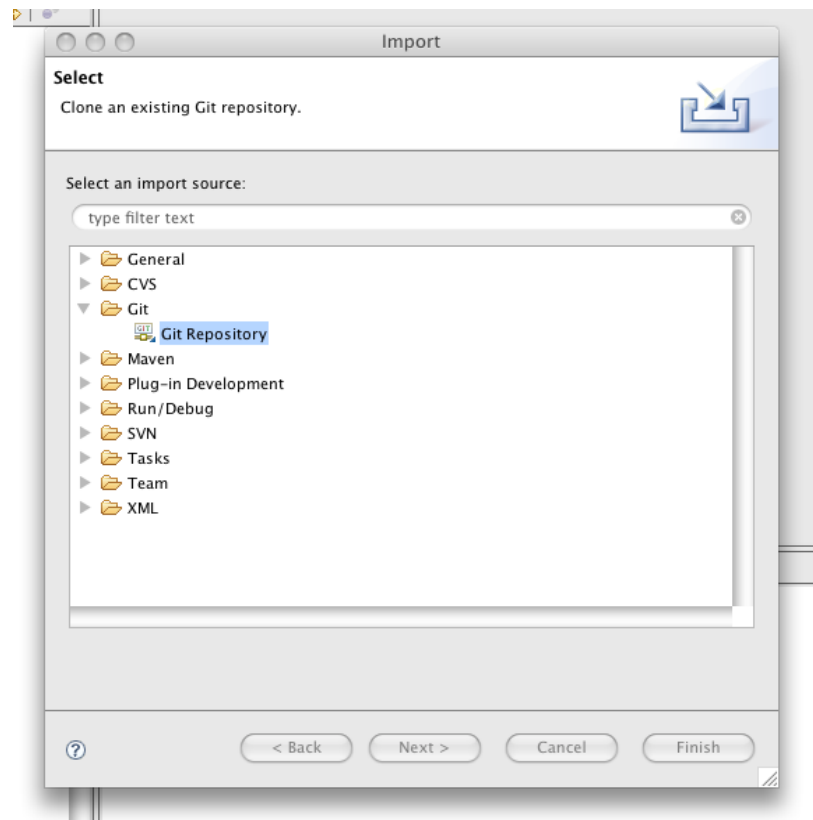
In a blog post on <http://swordfishing.wordpress.com> we saw how to access the Swordfish Git repository using Git from command line. Now that may not be everybody's cup of tea. Some people (including me) are not so happy when forced to do **everything** from command line. I don't like it with CVS or SVN either.

Fortunately there is the “EGit” plugin for Eclipse (<http://github.com/guides/using-the-egit-eclipse-plugin-with-github>) – including a nice tutorial. The plugin is still in an early stage, so we'll need to install the command line tools anyway.

The next thing we need is an account on Github.com. Most probably we'll want to add a SSH public key to our account.

Finally Oliver to add we as collaborator (you'll need that to be able to push our updates to Olivers repository).

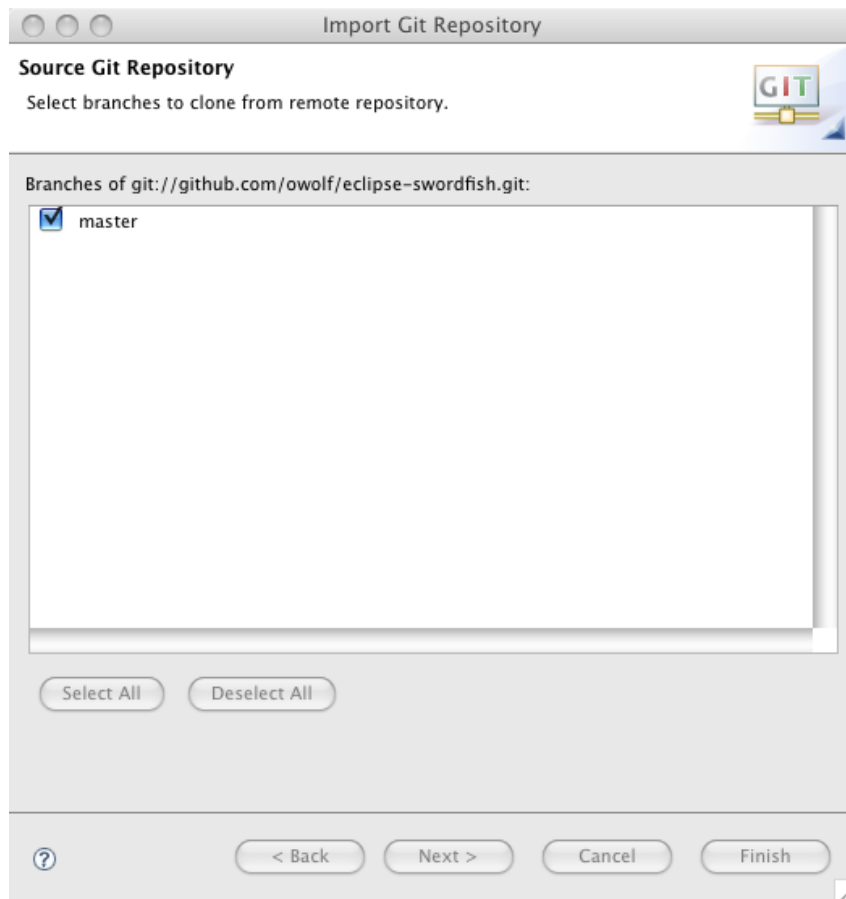
Now we can import the Swordfish Git (located at <git://github.com/owolf/eclipse-swordfish.git>), we can just import it into Eclipse using “Import ...”



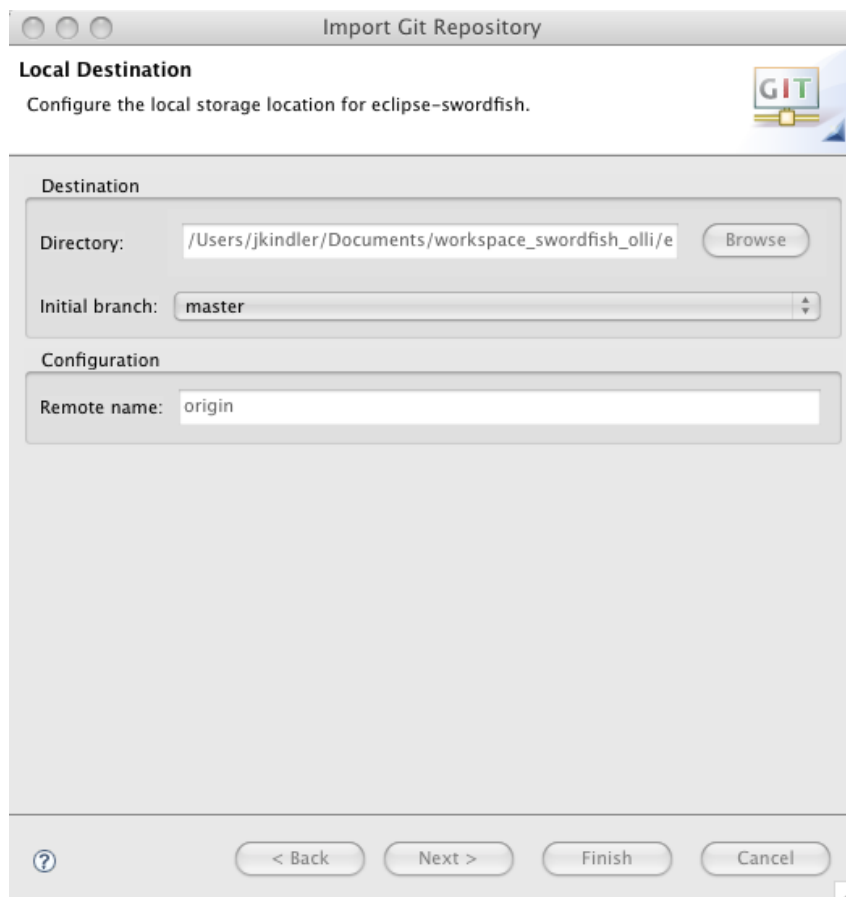
Then we select Git as source and enter the Swordfish Git URI (located at `git://github.com/owolf/eclipse-swordfish.git`) as source.

The screenshot shows the 'Import Git Repository' dialog box. The title bar reads 'Import Git Repository'. Below the title bar, there is a 'Source Git Repository' section with the instruction 'Enter the location of the source repository.' and a small Git logo. The dialog is divided into three main sections: 'Location', 'Connection', and 'Authentication'.
- In the 'Location' section, the 'URI' field contains 'git://github.com/owolf/eclipse-swordfish.git', the 'Host' field contains 'github.com', and the 'Repository path' field contains '/owolf/eclipse-swordfish.git'.
- In the 'Connection' section, the 'Protocol' dropdown menu is set to 'git', and the 'Port' field is empty.
- In the 'Authentication' section, both the 'User' and 'Password' fields are empty.
At the bottom of the dialog, there are four buttons: a help button (question mark), '< Back', 'Next >', 'Cancel', and 'Finish'.

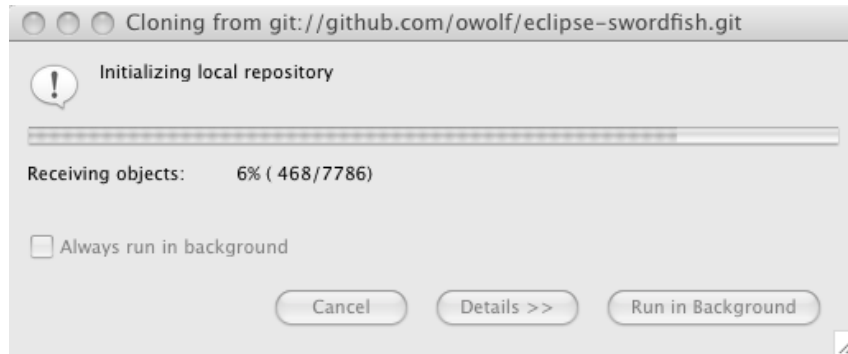
We press the “Next >” button.



Select destination and initial branch and press finish.

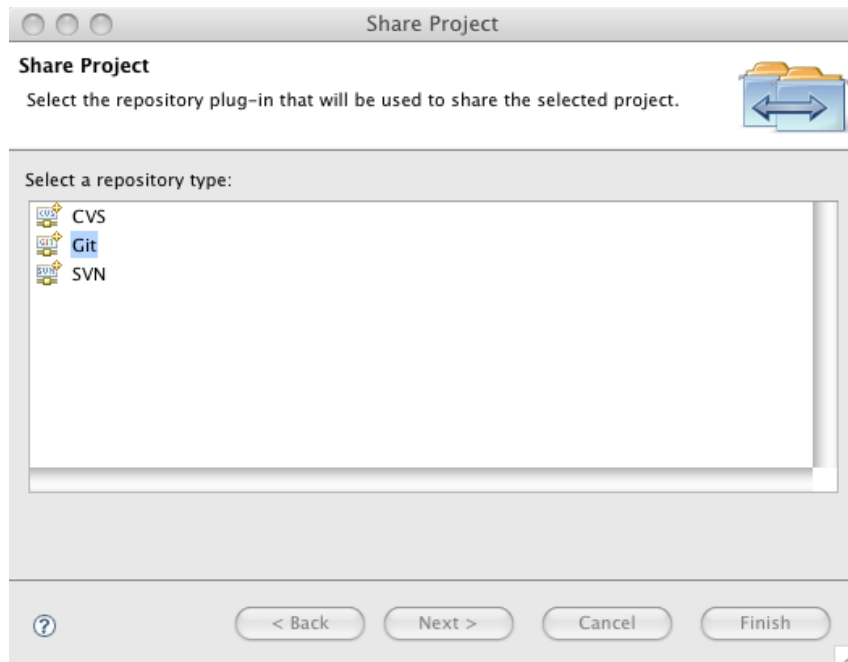


Sooner or later the repository cloning will finish ...



Then we need to go to a terminal window to create the Eclipse projects and import them as existing projects (as shown in Renats post at <http://swordfishing.wordpress.com/2009/02/03/first-steps/>).

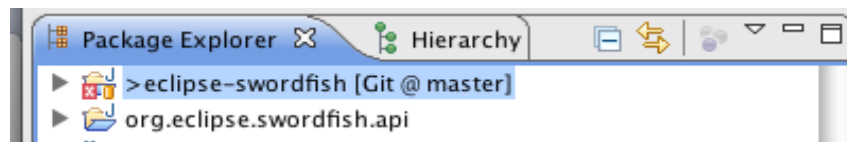
Afterwards it's easiest to create an Eclipse project file in the root project (just copy the one from the API project, change the name inside), import it and share it as a Git repository. To do that you right-click on the eclipse-swordfish project and choose "Team / Share Project..." :



Press "Finish" ...



Now the package explorer indicates that we are using Git:



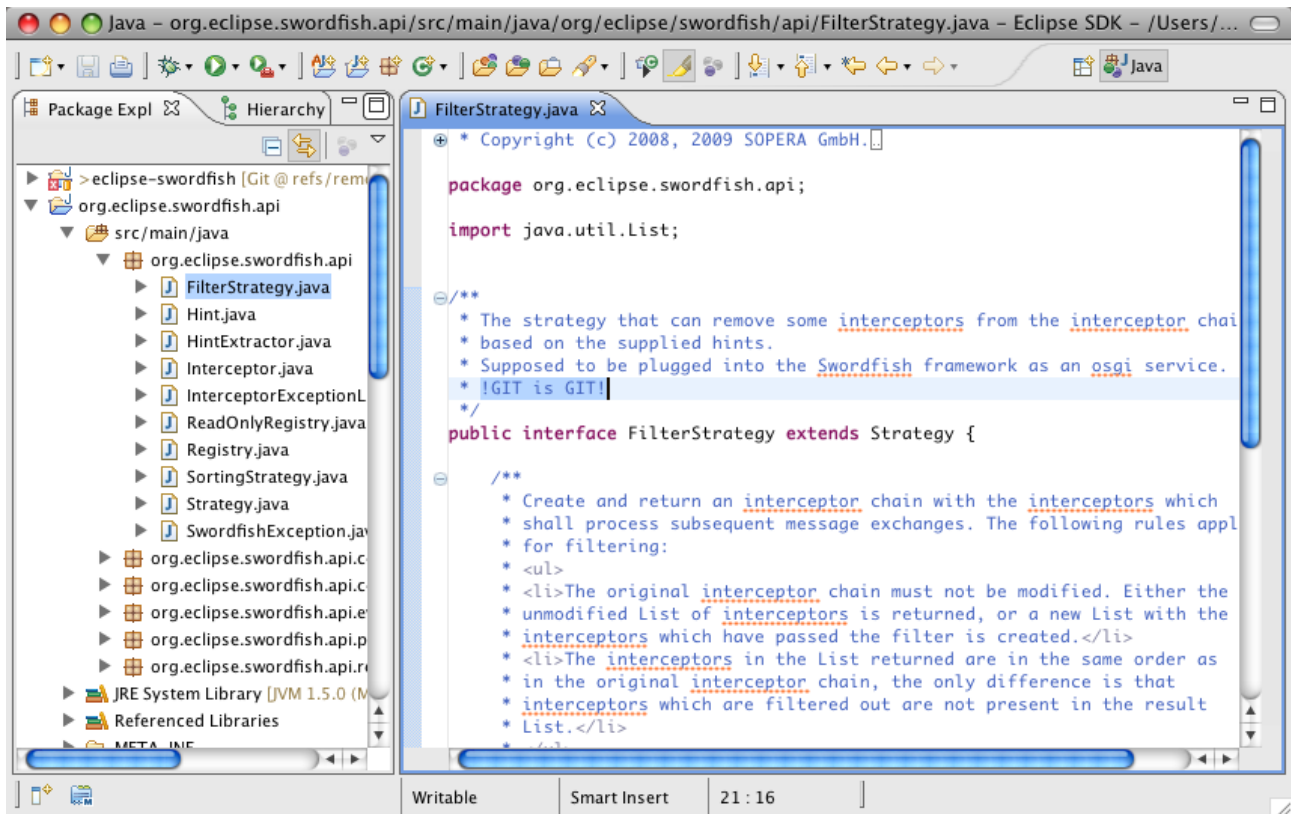
Some things will be a bit different from working with CVS or SVN. In these systems we are clearly a client who just received a working copy of the central repository. When we imported from the remote repository, we have created a standalone local repository of Oliver's remote repository residing on github.com. So working with that requires slightly different steps.

To make it a bit more colourful, let's have a look at some typical scenarios:

- committing a non-conflicting fix
- committing a fix that causes conflicts
- resetting the local repository clone to the state of its origin

Committing a non-conflicting fix

Let's make a tiny change on the FilterStrategy interface:

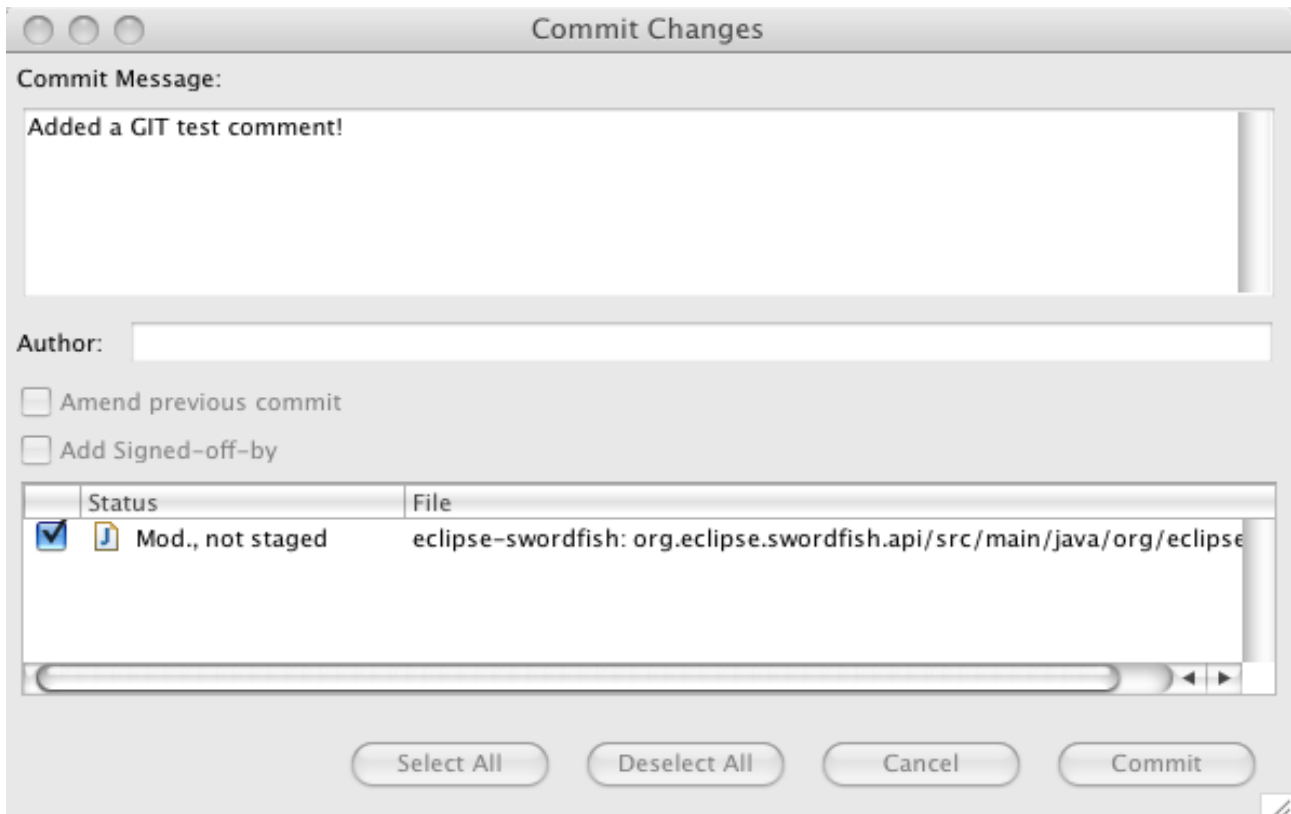


```
Java - org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java - Eclipse SDK - /Users/...
Package Expl | Hierarchy | FilterStrategy.java
+ * Copyright (c) 2008, 2009 SOPERA GmbH.
package org.eclipse.swordfish.api;
import java.util.List;

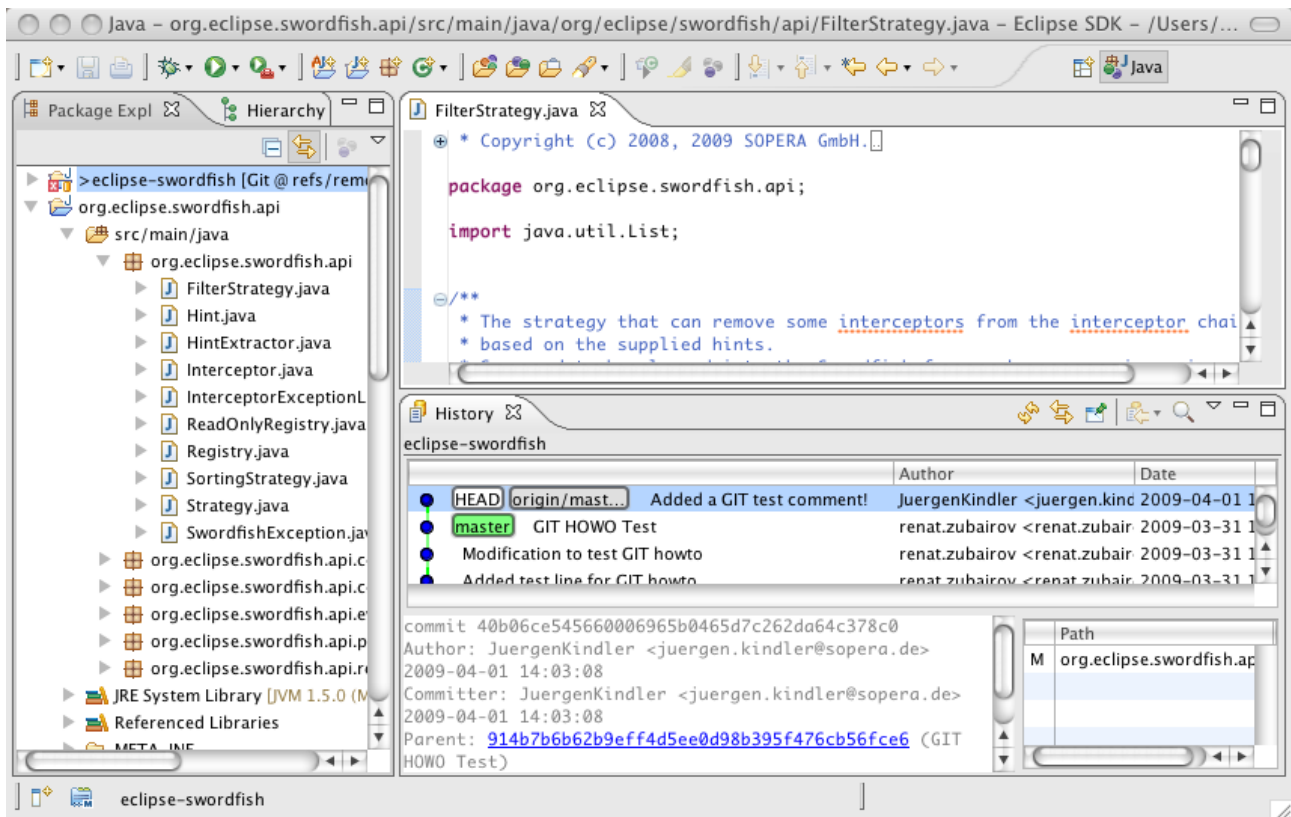
/**
 * The strategy that can remove some interceptors from the interceptor chain
 * based on the supplied hints.
 * Supposed to be plugged into the Swordfish framework as an osgi service.
 * !GIT is GIT!
 */
public interface FilterStrategy extends Strategy {

    /**
     * Create and return an interceptor chain with the interceptors which
     * shall process subsequent message exchanges. The following rules apply
     * for filtering:
     * <ul>
     * <li>The original interceptor chain must not be modified. Either the
     * unmodified List of interceptors is returned, or a new List with the
     * interceptors which have passed the filter is created.</li>
     * <li>The interceptors in the List returned are in the same order as
     * in the original interceptor chain, the only difference is that
     * interceptors which are filtered out are not present in the result
     * List.</li>
     * </ul>
     */
}
```

After saving the change it has to be committed to the *local* Git repository first. So we choose the Team/Commit on the eclipse-swordfish root project:



We press “Commit”, afterwards we right-click on the root again and select “Team / Show in Resource History”:



Ok, the change is done locally. Now our remote master has to be updated. In Git the command to do this is “PUSH”. So we select “Push to ...” from the Team context menu (Note that on the following screenshots I am using a clone of a local Git repository – if you push to Olivers repo, you'll certainly see a real remote URI):

Push To Another Repositories

Destination Git Repository
Enter the location of the destination repository.

Configured remote repository:
origin: /Users/jkindler/Documents/workspace_swordfish_git2/../../workspace_swordfish_git/eclipse-swordfish/

Custom URI:

Location

URI:

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

Now we'll have fun with this dialog. For a first-timer it's hard to get.

Push Ref Specifications

Select refs to push.



Add create/update specification

Source ref: Destination ref:

Add delete ref specification

Remote ref to delete:

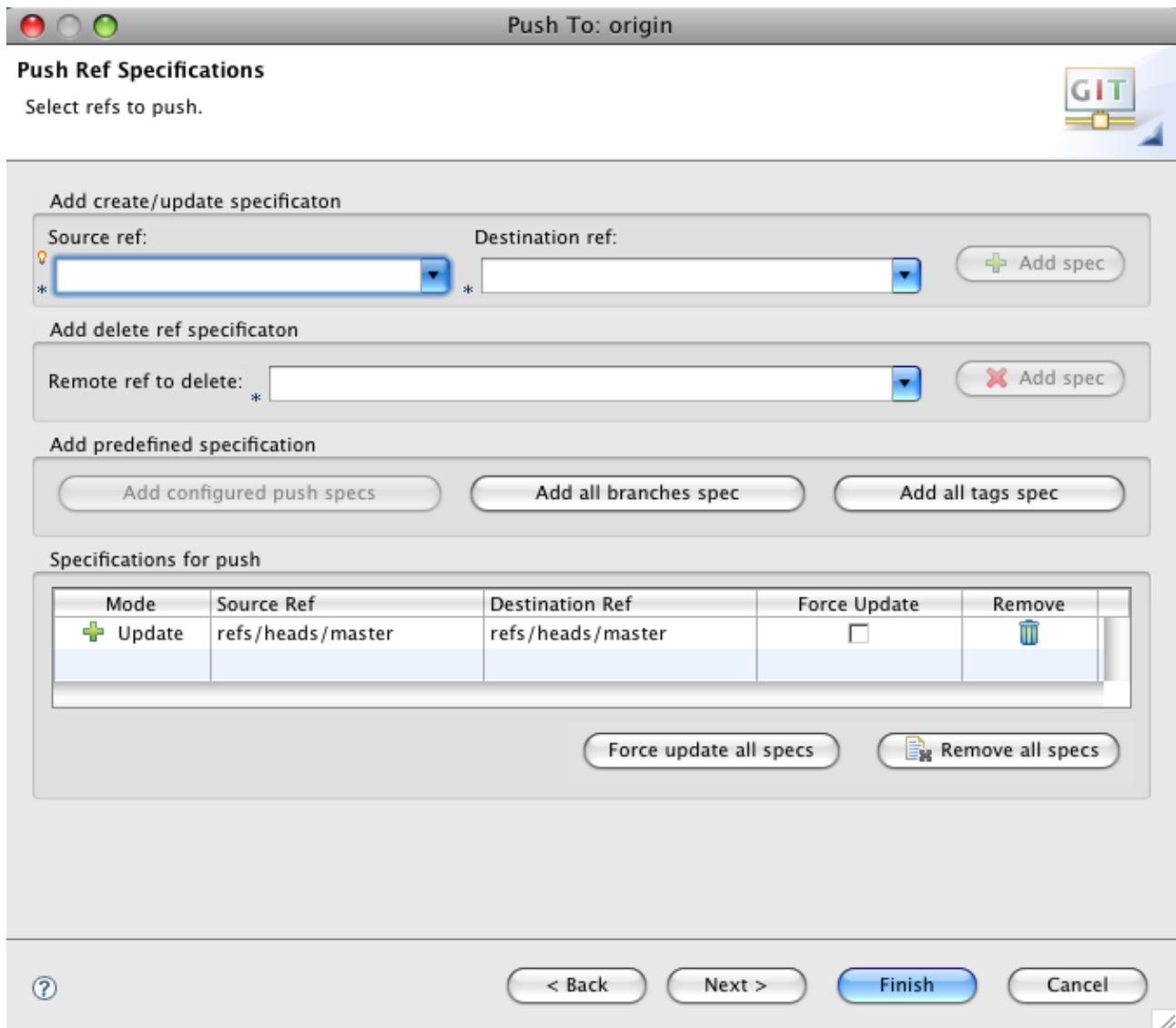
Add predefined specification

Specifications for push

| Mode | Source Ref | Destination Ref | Force Update | Remove |
|------|------------|-----------------|--------------|--------|
| | | | | |

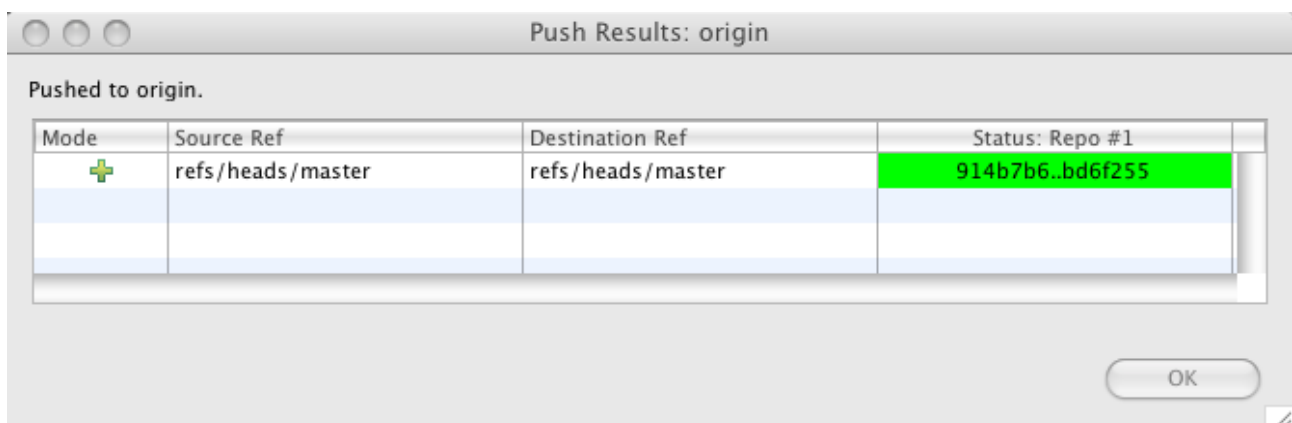


Currently we are on the local master (“master [branch]”), so we simply select this as source. The remote master will automatically selected and we press “Add spec”:



Note that we should not check the “Force Update” when pushing as it would overwrite the remote origin with our local one. It may be tempting if there is a conflict, but forcing is only OK for incoming changes (to overwrite local changes that do not exist in the remote origin).

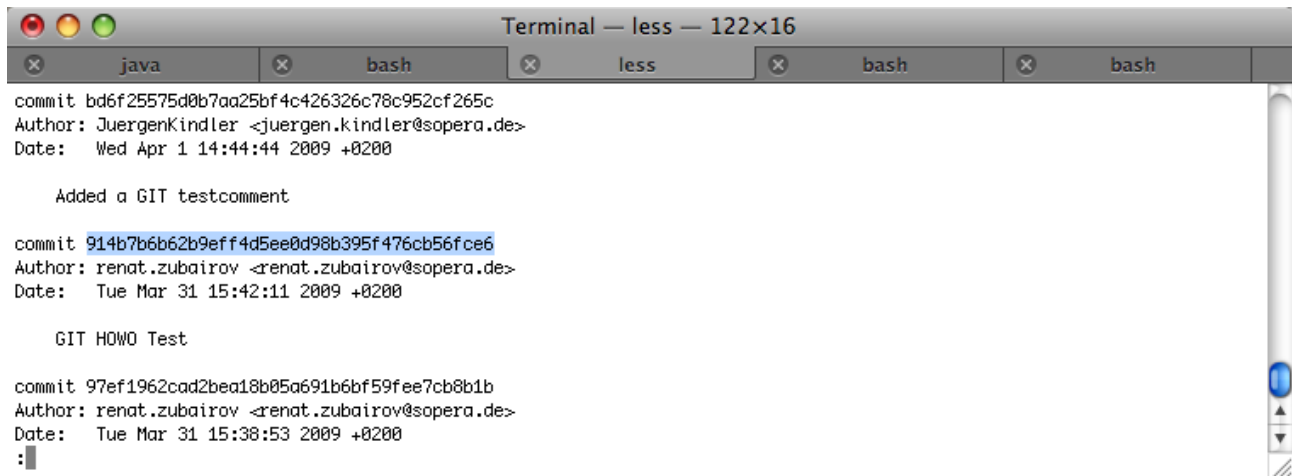
So when we press “Finish” we get this:



Congratulations – the remote repository has been updated! Now the CI build based on Oliver's Git will be triggered. In the mean time, we can create a patch file that includes our fix (and attach it to a

Bugzilla request. To do that we need a terminal window and use the command line tool. We enter the eclipse-swordfish directory of our workspace and enter

```
git log
```



```
Terminal — less — 122x16
commit bd6f25575d0b7aa25bf4c426326c78c952cf265c
Author: JuergenKindler <juergen.kindler@sopera.de>
Date: Wed Apr 1 14:44:44 2009 +0200

    Added a GIT testcomment

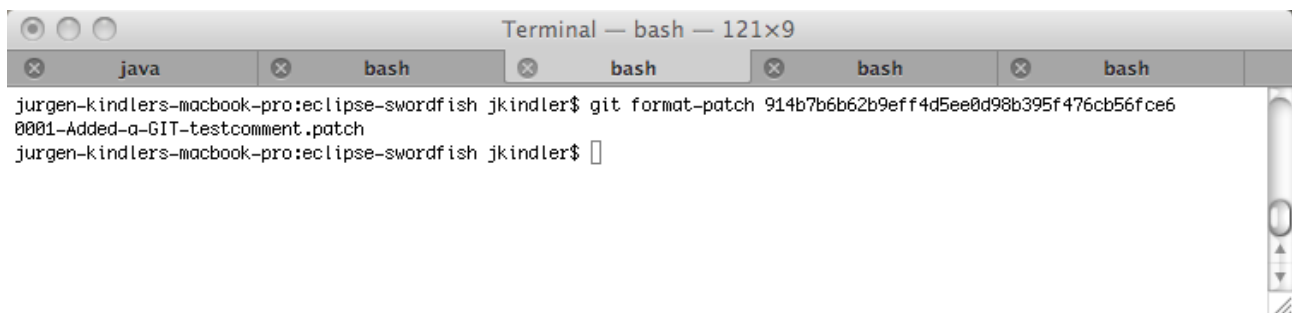
commit 914b7b6b62b9eff4d5ee0d98b395f476cb56fce6
Author: renat.zubairov <renat.zubairov@sopera.de>
Date: Tue Mar 31 15:42:11 2009 +0200

    GIT HOW0 Test

commit 97ef1962cad2bea18b05a691b6bf59fee7cb8b1b
Author: renat.zubairov <renat.zubairov@sopera.de>
Date: Tue Mar 31 15:38:53 2009 +0200
:
```

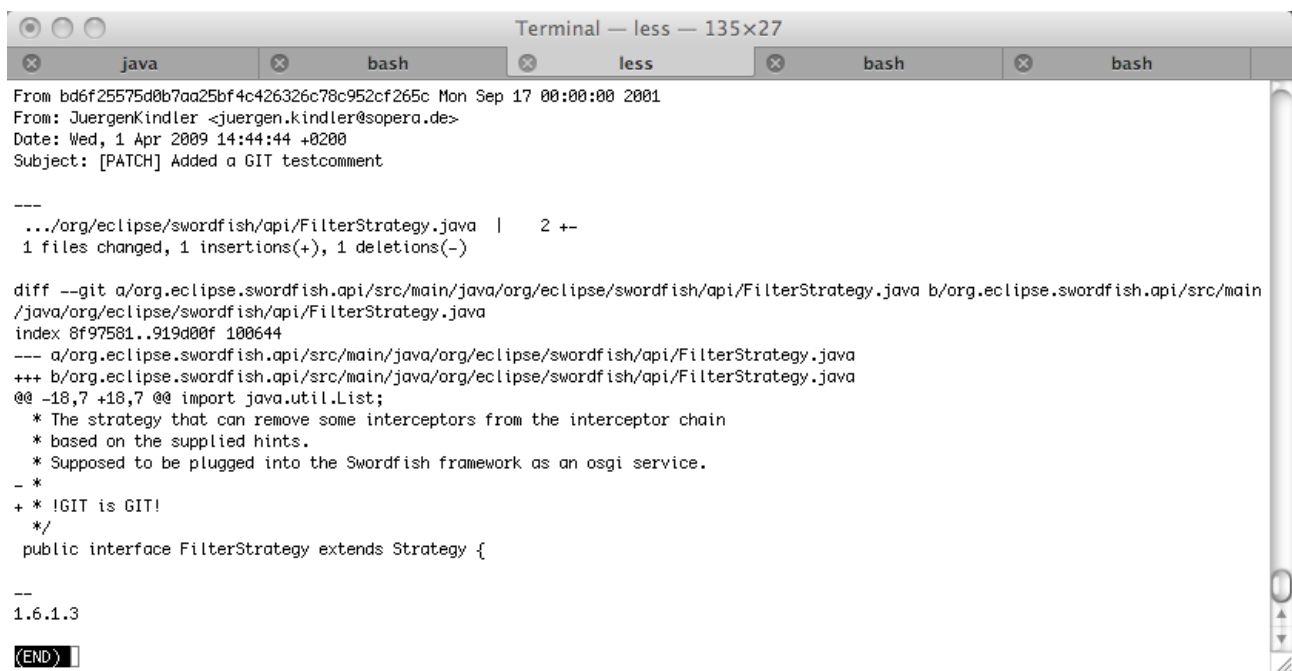
Now we copy the ID of the *previous* commit and create the patch file using

```
git format-patch 914b7b6b62b9eff4d5ee0d98b395f476cb56fce6
```



```
Terminal — bash — 121x9
jürgen-kindlers-macbook-pro:eclipse-swordfish jkindler$ git format-patch 914b7b6b62b9eff4d5ee0d98b395f476cb56fce6
0001-Added-a-GIT-testcomment.patch
jürgen-kindlers-macbook-pro:eclipse-swordfish jkindler$
```

Git has created a patch for the commit that happened since this previous one. Let's have a look at what is inside:



```
Terminal — less — 135x27
From bd6f25575d0b7aa25bf4c426326c78c952cf265c Mon Sep 17 00:00:00 2001
From: JuergenKindler <juergen.kindler@sopera.de>
Date: Wed, 1 Apr 2009 14:44:44 +0200
Subject: [PATCH] Added a GIT testcomment

---
.../org/eclipse/swordfish/api/FilterStrategy.java | 2 +-
1 files changed, 1 insertions(+), 1 deletions(-)

diff --git a/org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java b/org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java
index 8f97581..919d00f 100644
--- a/org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java
+++ b/org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java
@@ -18,7 +18,7 @@ import java.util.List;
 * The strategy that can remove some interceptors from the interceptor chain
 * based on the supplied hints.
 * Supposed to be plugged into the Swordfish framework as an osgi service.
- *
+ * !GIT is GIT!
 */
public interface FilterStrategy extends Strategy {

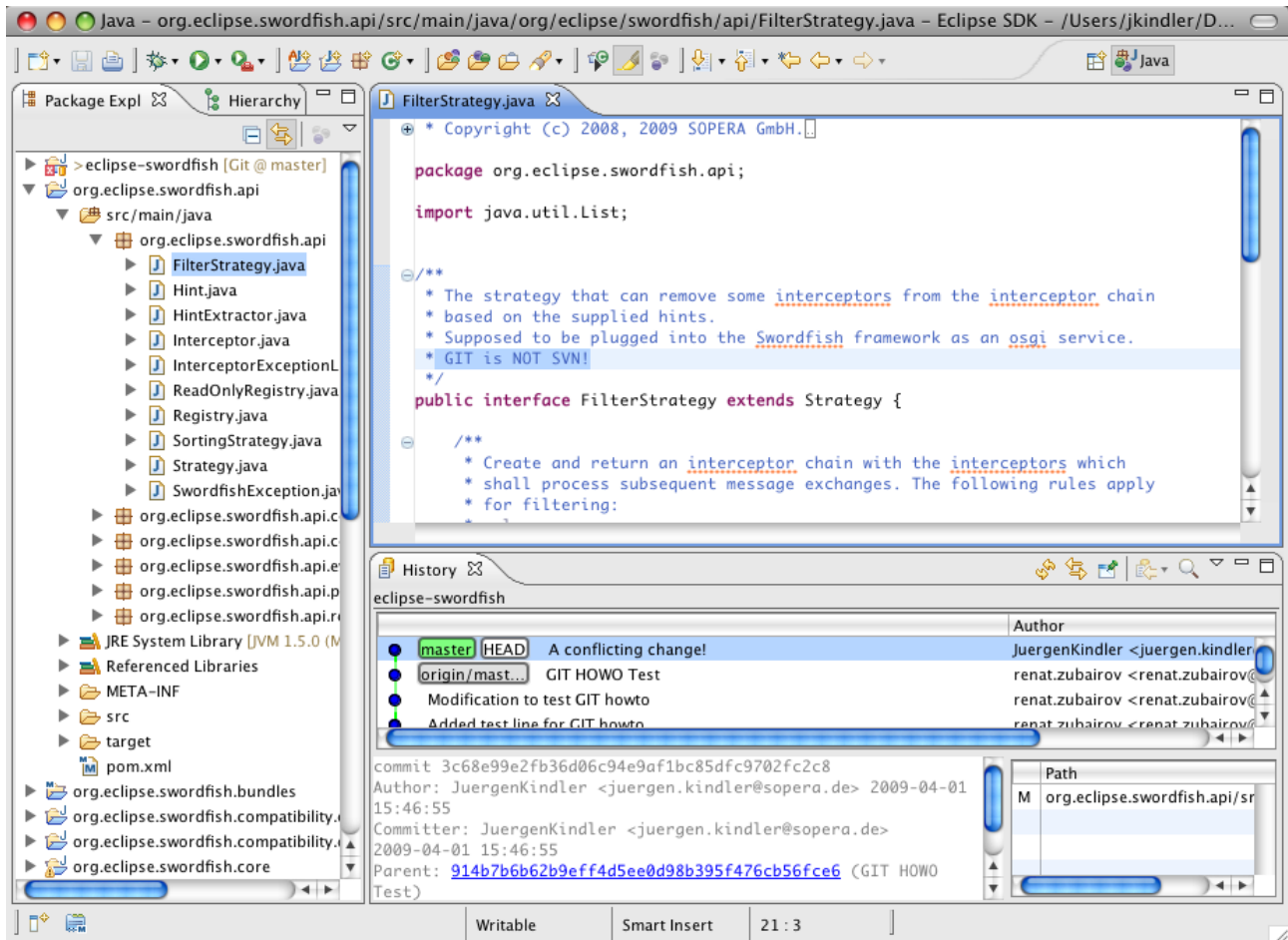
--
1.6.1.3

<END>
```

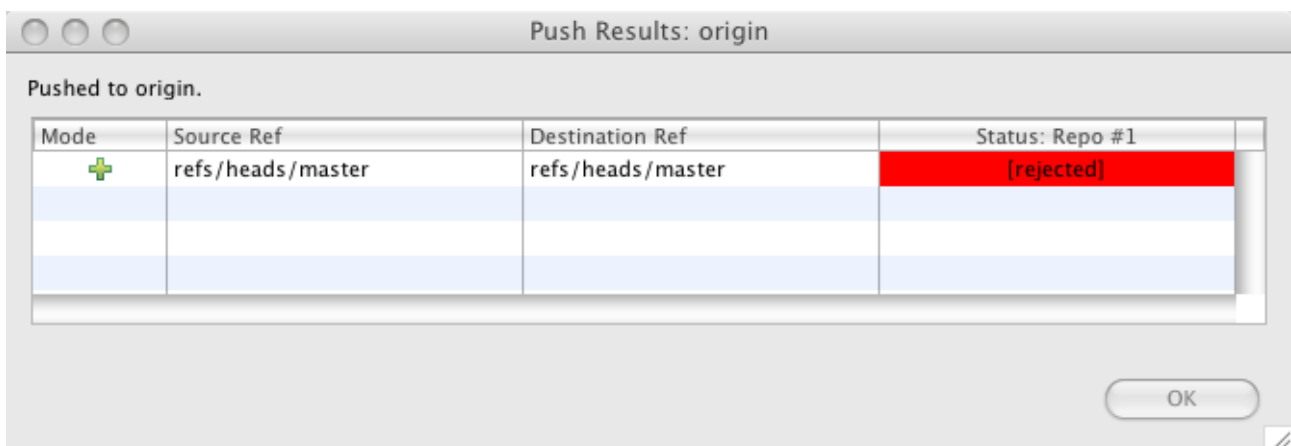
Note that you can go backwards to previous patches as well. Each commit will end up in a separate patch file.

Committing a fix that causes conflicts

OK, the previous change was pretty simple ... pretty unrealistic also ;-). Let's have a look what happens if somebody else created a local clone of Oliver's repository before we pushed our change to there. So the change is done and committed to the local Git (note that origin/master is one step behind us):

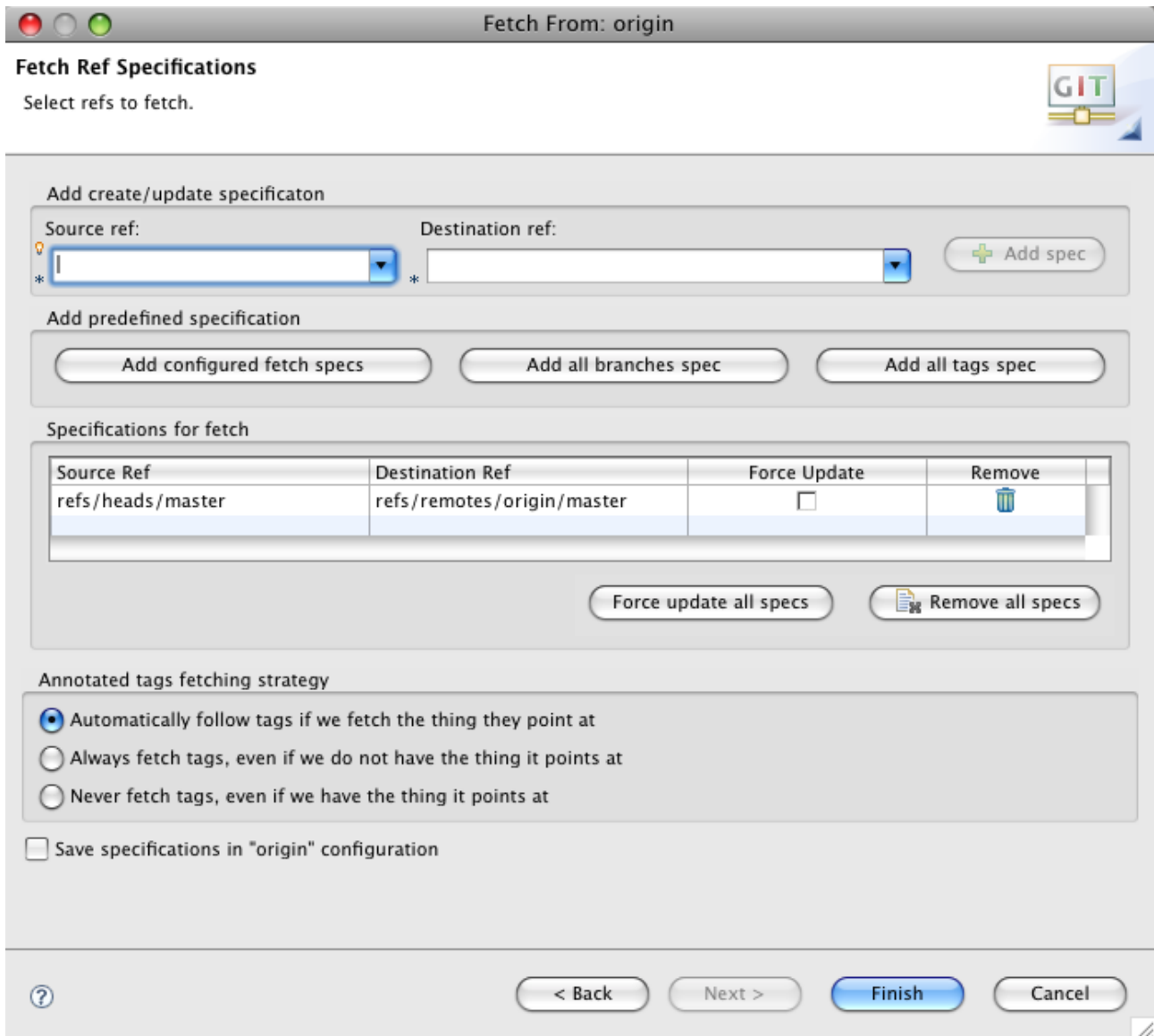


OK, so we want to be documentation heroes and push this change to the origin repository:

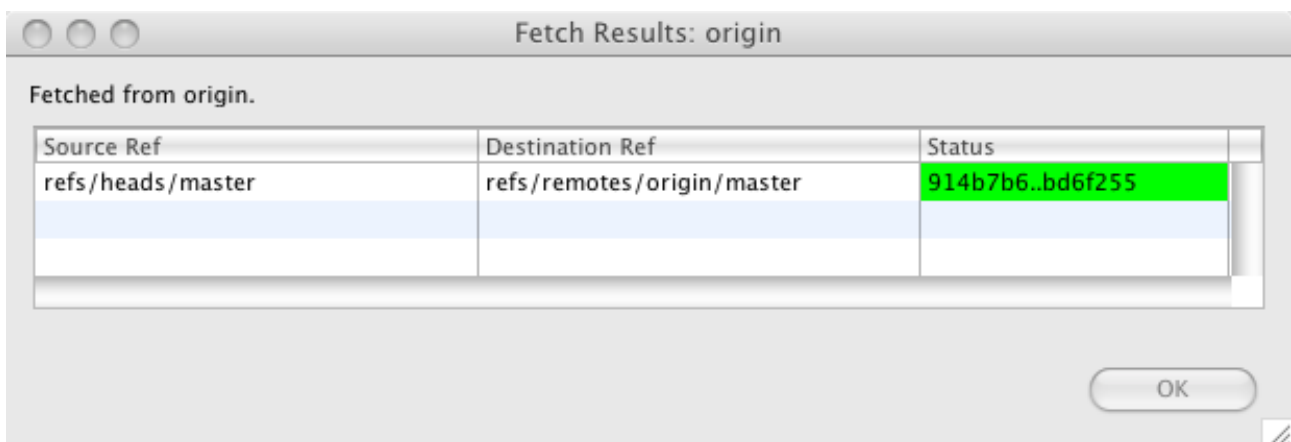


So somebody was faster – our change is rejected because of a merge conflict. :-)

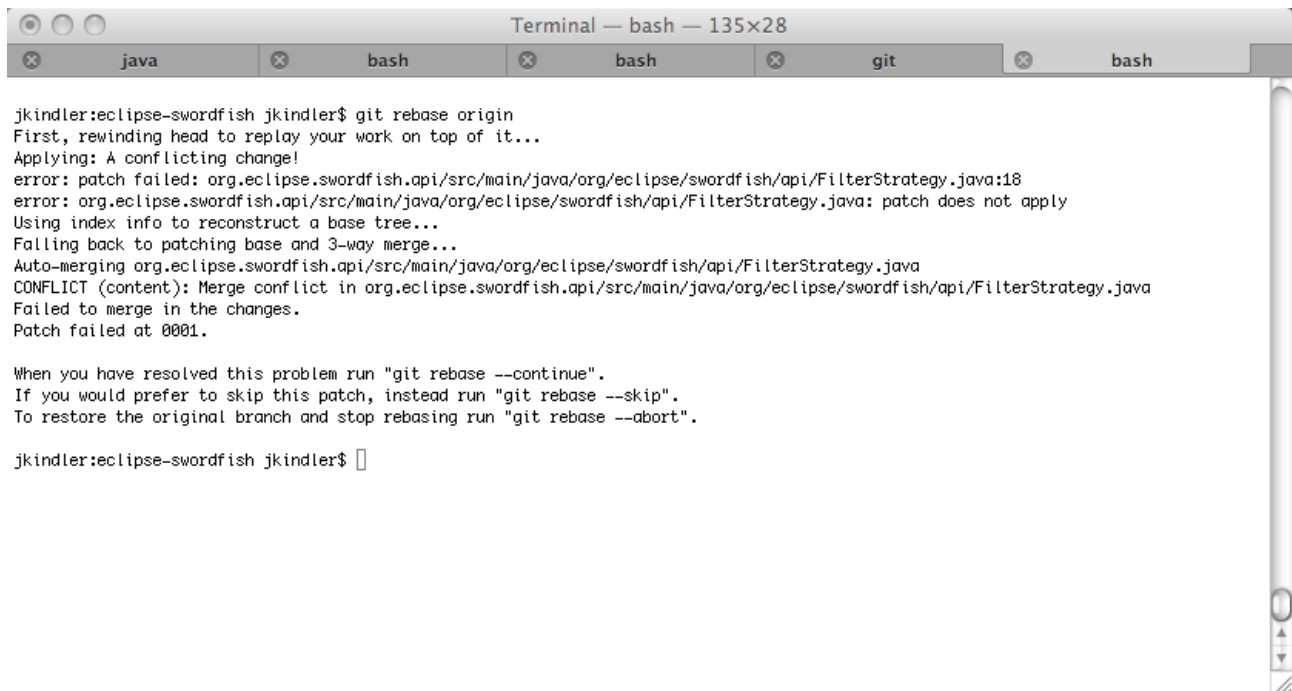
We need to know what happened on the origin repository and fetch its contents:



So we are able to get the changes:



But now we have to leave Eclipse and open a Terminal window to rebase with the origin repository, because that is not yet supported in the Egit plugin. We enter `git rebase origin` and Git tries to merge changes. In our case it fails, because of the conflicting line:

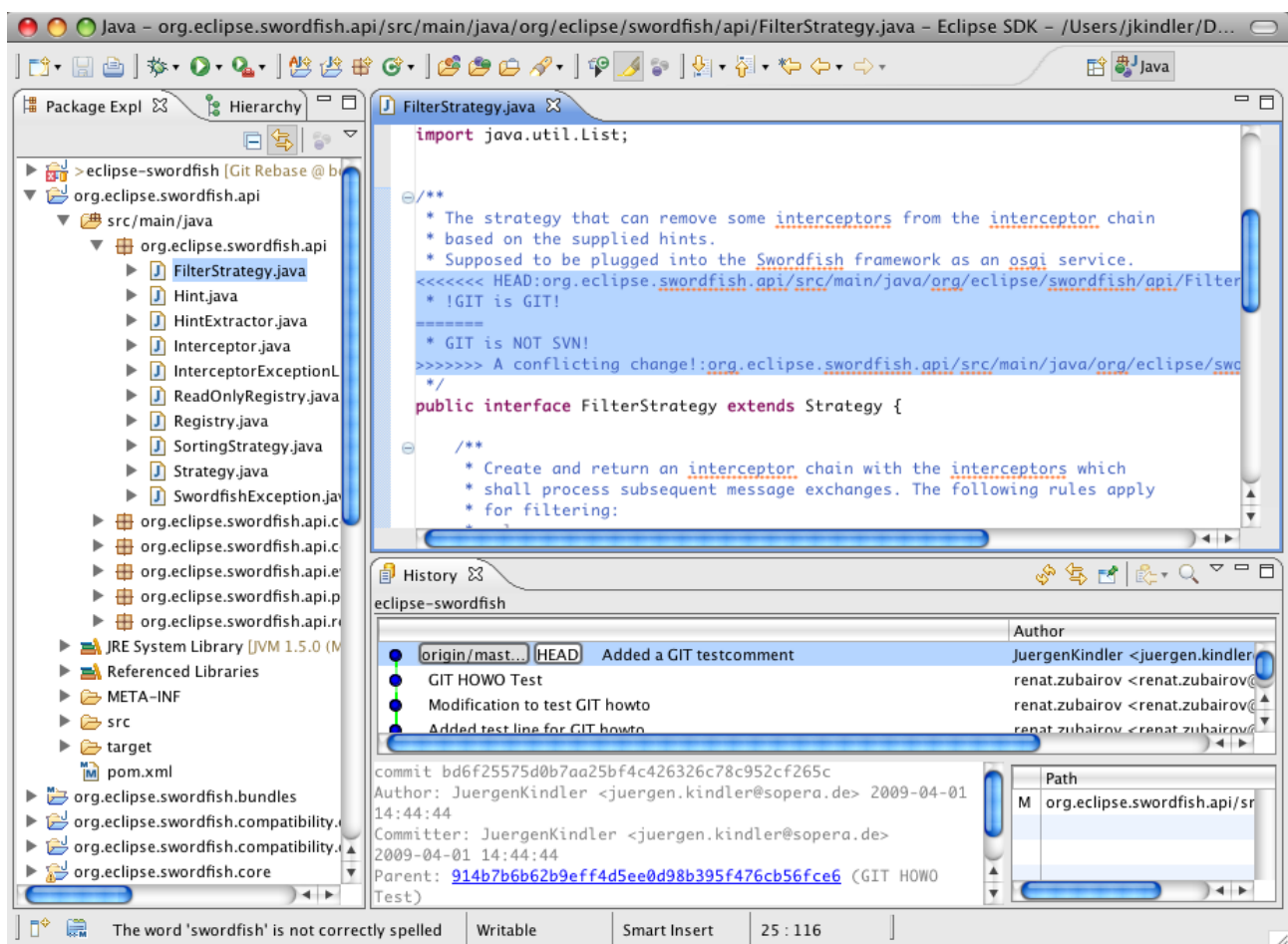


```
Terminal — bash — 135x28
jkindler:eclipse-swordfish jkindler$ git rebase origin
First, rewinding head to replay your work on top of it...
Applying: A conflicting change!
error: patch failed: org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java:18
error: org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java: patch does not apply
Using index info to reconstruct a base tree...
Falling back to patching base and 3-way merge...
Auto-merging org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java
CONFLICT (content): Merge conflict in org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java
Failed to merge in the changes.
Patch failed at 0001.

When you have resolved this problem run "git rebase --continue".
If you would prefer to skip this patch, instead run "git rebase --skip".
To restore the original branch and stop rebasing run "git rebase --abort".

jkindler:eclipse-swordfish jkindler$
```

Now contents in our Eclipse editor change:

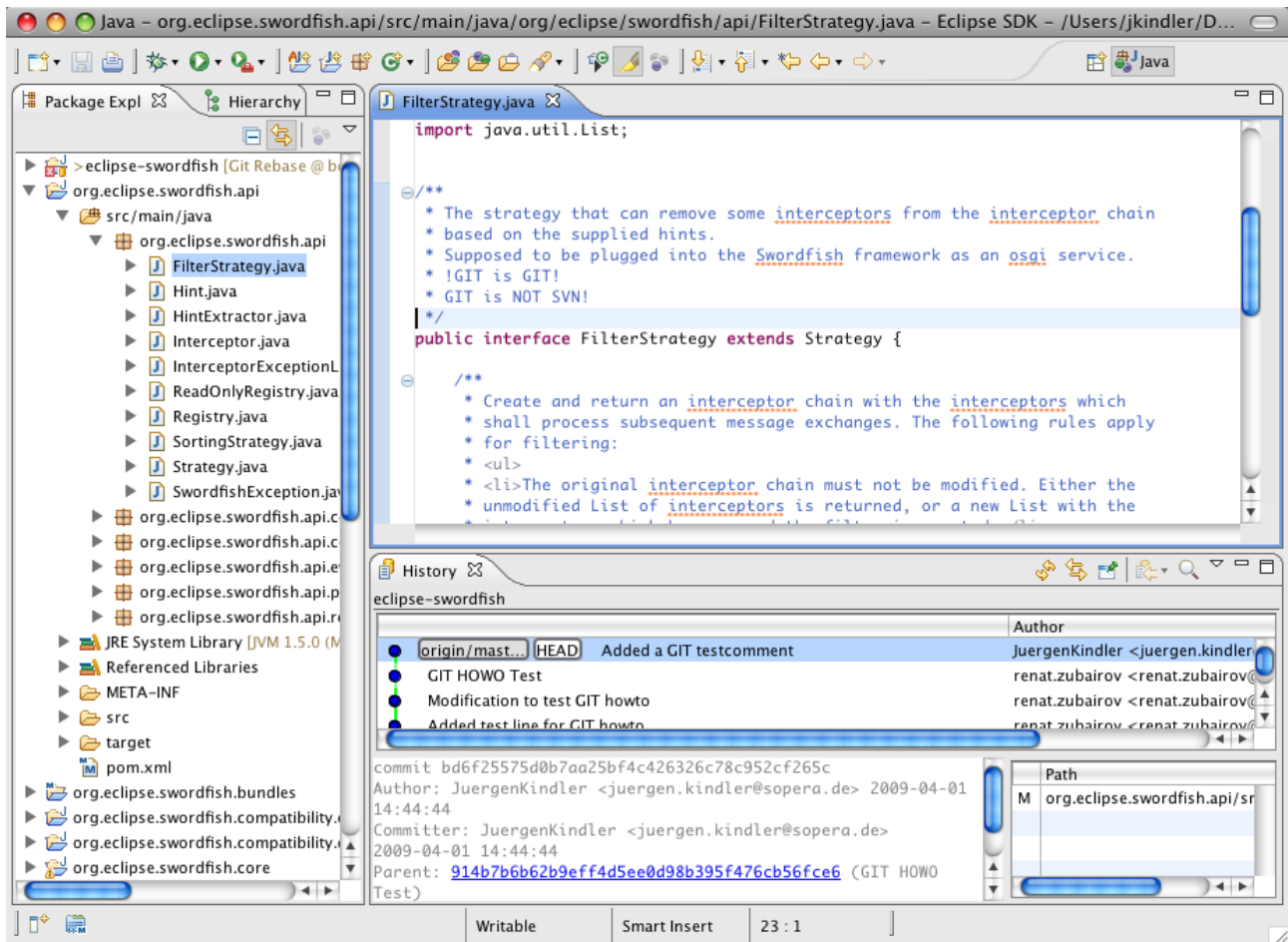


The screenshot shows the Eclipse IDE with the `FilterStrategy.java` file open. The code contains a conflict marker: `<<<<<< HEAD:org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java`. Below the code, the Git commit history window is visible, showing a list of commits. The current commit is highlighted, and its details are shown below.

| Commit | Author |
|--------------------------------|--|
| origin/mast... HEAD | JuergenKindler <juergen.kindler@sopera.de> |
| GIT HOWO Test | renat.zubairov <renat.zubairov@sopera.de> |
| Modification to test GIT howto | renat.zubairov <renat.zubairov@sopera.de> |
| Added test line for GIT howto | renat.zubairov <renat.zubairov@sopera.de> |

commit bd6f25575d0b7aa25bf4c426326c78c952cf265c
Author: JuergenKindler <juergen.kindler@sopera.de> 2009-04-01 14:44:44
Committer: JuergenKindler <juergen.kindler@sopera.de> 2009-04-01 14:44:44
Parent: 914b7b6b62b9eff4d5ee0d98b395f476cb56fce6 (GIT HOWO Test)

We resolve the conflict by keeping both changes and save our file:



That done we do not have to commit, but have to (re-) add¹ the changed file to indicate that we have solved the conflict using

```
git add
org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java
```

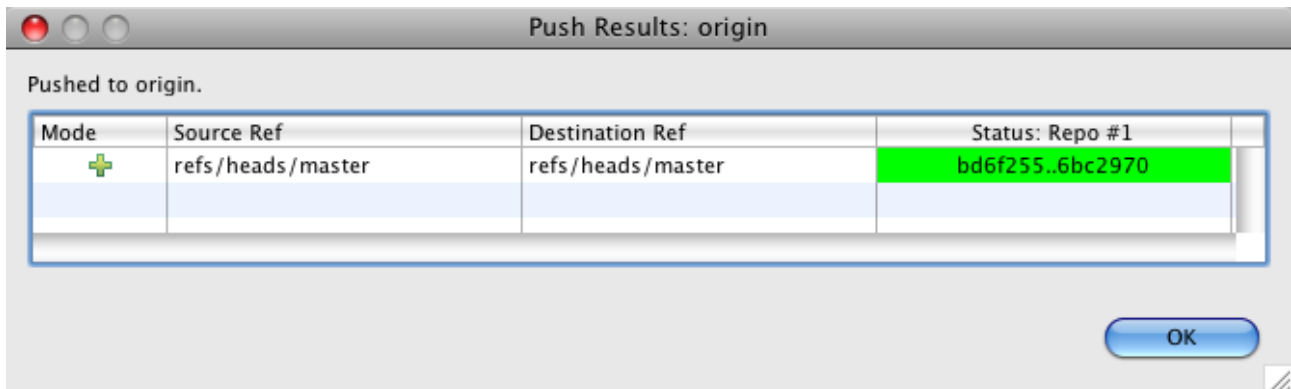
Now the rebase can continue:

```
git rebase --continue
```

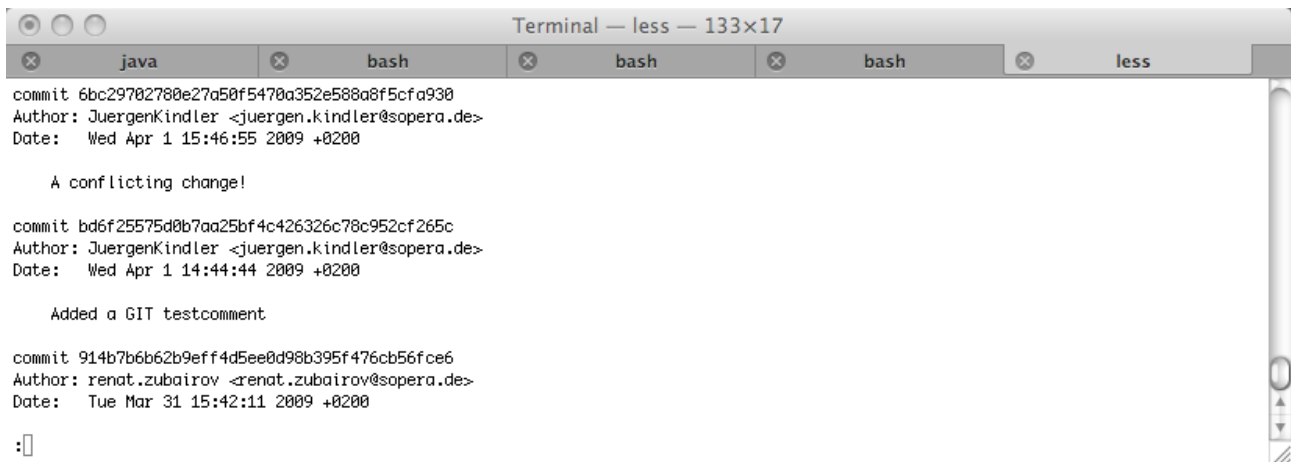


¹ Curious who implemented `git add`? Then type `git add -help` and go to the bottom of the help text ;-)

And having done that, our push will work:



Again we can see that on the terminal window as well using `git log`:



Resetting the local repository clone to the state of its origin

Suppose we have made some changes that we do not want to push to the origin repository. In that case we need to go back to the last state of the origin (as we can see below, our local master branch is 3 commits ahead of the last known state of the origin master branch).

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer shows the project structure for 'eclipse-swordfish'. The main editor displays the 'FilterStrategy.java' file with the following code:

```
/**
 * The strategy that can remove some interceptors from the interceptor chain
 * based on the supplied hints.
 * Supposed to be plugged into the Swordfish framework as an osgi service.
 * !GIT is GIT!
 * !GIT is NOT SVN! More serious info
 */
public interface FilterStrategy extends Strategy {

    /**
     * Create and return an interceptor chain with the interceptors which
     * shall process subsequent message exchanges. The following rules apply
     * for filtering:
     * <ul>
     * <li>The original interceptor chain must not be modified. Either the
     * unmodified List of interceptors is returned, or a new List with the
     * interceptors which have passed the filter is created. /i

```

The History view at the bottom shows the commit history for the 'eclipse-swordfish' repository. The local 'HEAD' branch is on 'master' and is 3 commits ahead of the 'origin/master' branch. The commit history is as follows:

| Commit | Author | Date |
|---------------------|--------------------------------------|---------------------|
| HEAD master | JuergenKindler <juergen.kindler@...> | 2009-04-02 07:44:20 |
| Improved even more! | JuergenKindler <juergen.kindler@...> | 2009-04-02 07:44:20 |
| Improved info | JuergenKindler <juergen.kindler@...> | 2009-04-02 07:44:20 |
| Added more info | JuergenKindler <juergen.kindler@...> | 2009-04-02 07:44:20 |
| origin/mast... | JuergenKindler <juergen.kindler@...> | 2009-04-02 07:44:20 |

The commit details for the selected commit (85be626bd7fdf83b31f87ea78fa5c983f13acf9a) are shown below the history view:

```
commit 85be626bd7fdf83b31f87ea78fa5c983f13acf9a
Author: JuergenKindler <juergen.kindler@sopera.de> 2009-04-02 07:44:20
Committer: JuergenKindler <juergen.kindler@sopera.de> 2009-04-02 07:44:20
Parent: fb19c1e86bd6fa60b0c0f8e5bdb1d9c3d31b5cdc (Improved info)
```

With EGit you will have to get the last changes of the origin and then reset you local branch to the state of the origin. So right-click on the eclipse-swordfish project and select “Team / Fetch from ...”

Fetch From Another Repository

Source Git Repository
Enter the location of the source repository.

Configured remote repository:

origin: /Users/jkindler/Documents/workspace_swordfish_git2/../../workspace_swordfish_git/eclipse-swordfish/

Custom URI:

Location

URI:

Host:

Repository path:

Connection

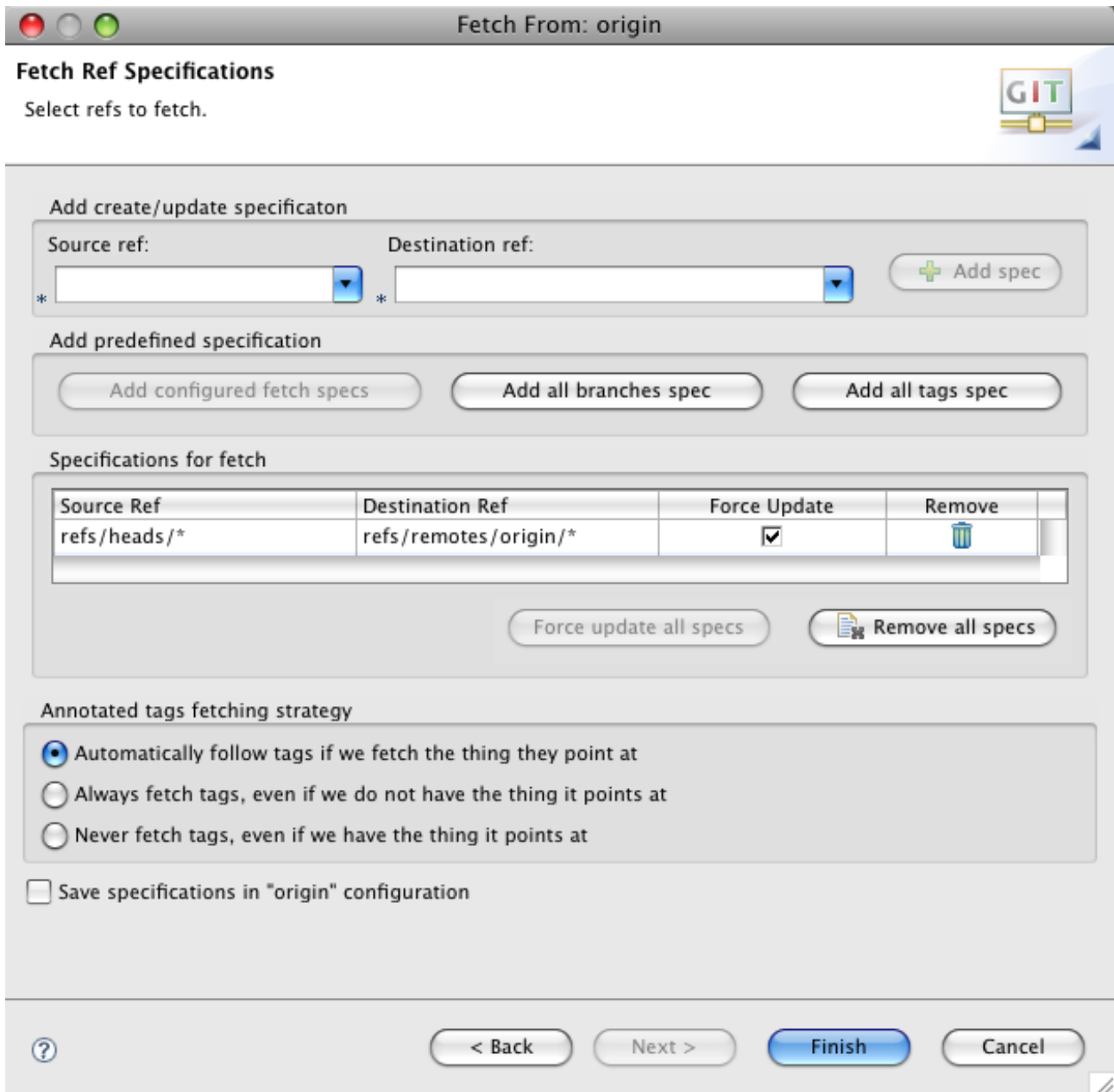
Protocol:

Port:

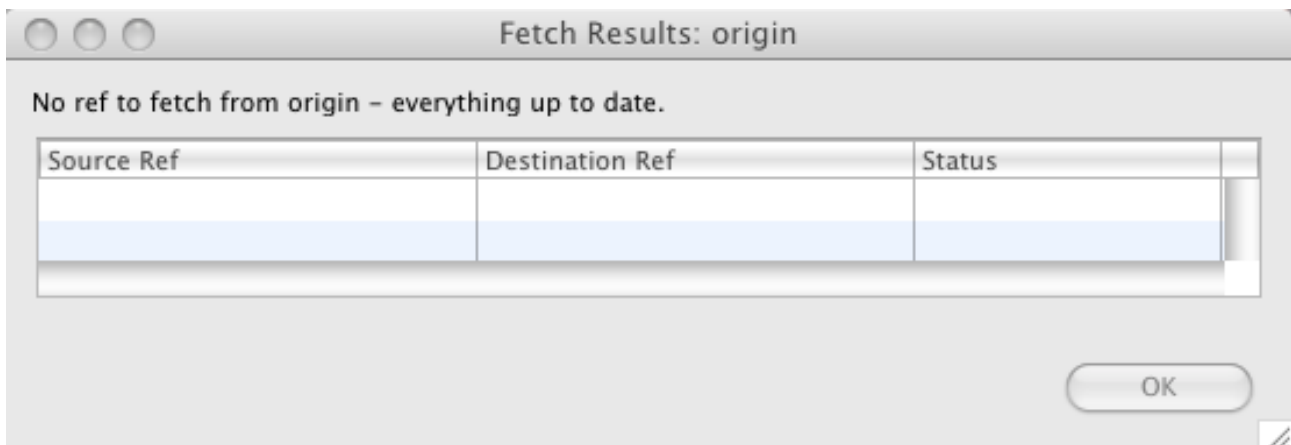
Authentication

User:

Press "Next >" ...

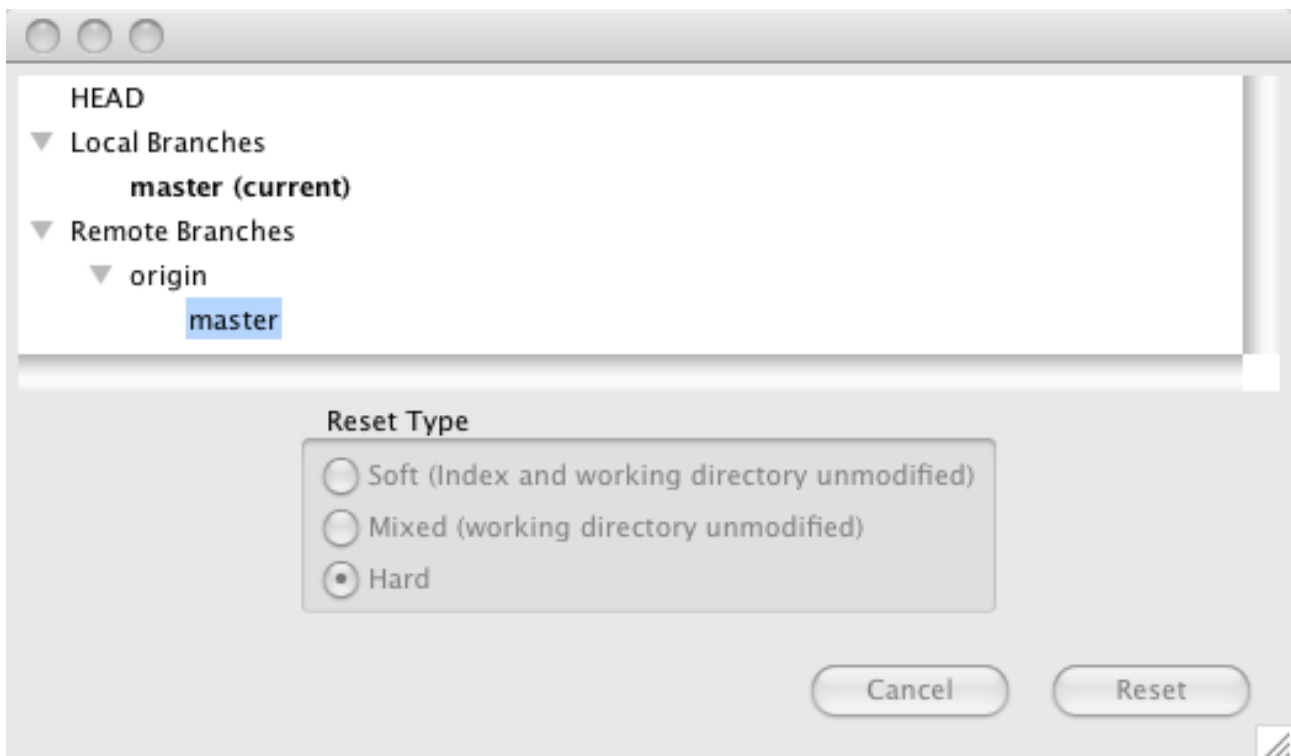


Then "Finish" it:

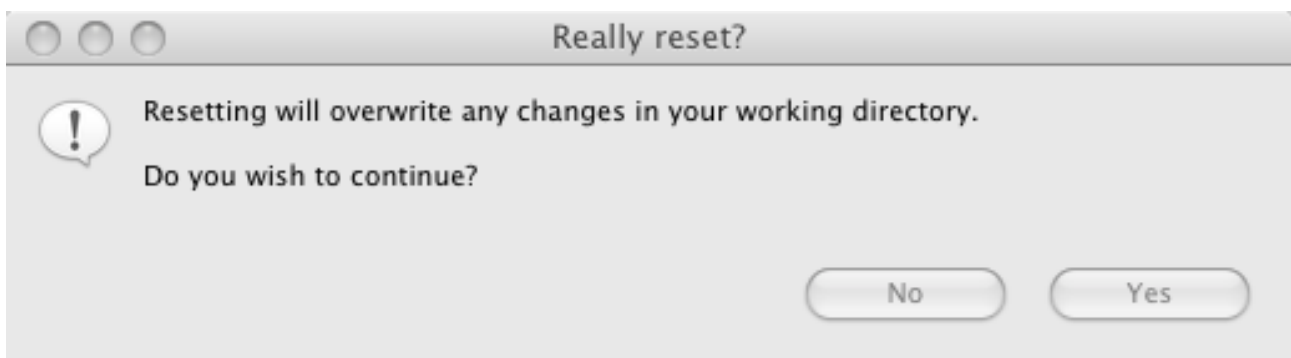


So there were no changes on the origin since our last fetch command.

Now we can reset our branch to the contents of the origin. We use “Team / Reset To ...”, select the master branch of the origin repository we just fetched and select the Reset Type “Hard” (overwrite everything locally!):



No we will have to confirm that we really want to wipe out all local changes (the answer is Yes!)



So when the operation is finished, we see that our local master branch and the remote master branch are at the same state:

