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 Oliver's 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 us as collaborator (you'll need that to be able to push our updates to Oliver's 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.

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 Olivers 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:

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:

Commit Message:

Added a GIT test comment!

Author:

Commit Changes

Status       File
---          -------
Mod, not staged  eclipse-swordfish: org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java

Select All   Deselect All   Cancel   Commit
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):

**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.
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 Olivers 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
```

```
commit b0a2f56f9b70a25c0c82626c70f650c265:
Author: Juergen Kindler <juergen.kindler@espera.de>
Date: Wed Apr 1 14:44:44 2009 +0200

    Added a GIT test comment.
```

```
commit 914b7b6b62b9eff4d5ee0d98b395f476cb56f6ce:
Author: renat.zubirov <renat.zubirov@espera.de>
Date: Tue Mar 31 15:42:11 2009 +0200

    GIT HOW Test.
```

```
commit 96ef19600a7e2a78e4b76b9f359eff7cbb8b:
Author: renat.zubirov <renat.zubirov@espera.de>
Date: Tue Mar 31 15:39:51 2009 +0200

```

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

```
git format-patch 914b7b6b62b9eff4d5ee0d98b395f476cb56f6ce
```

```
  juergen-kindler@wackoock-pro-eclipse-swordfish:skindler$ git format-patch 914b7b6b62b9eff4d5ee0d98b395f476cb56f6ce
0001-Added a GIT test comment. patch
  juergen-kindler@wackoock-pro-eclipse-swordfish:skindler$ 
```

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

```
--- ./org/eclipse/swordfish/aps/FilterStrategy.java  1.2
1 file changed, 1 insertions(+), 1 deletions(-)
```

```
diff --git a/org/eclipse/swordfish/aps/FilterStrategy.java b/org/eclipse/swordfish/aps/FilterStrategy.java
index 8d7f5e01..919df0f0 100644
--- a/org/eclipse/swordfish/aps/FilterStrategy.java
+++ b/org/eclipse/swordfish/aps/FilterStrategy.java
@@ -19,7 +19,7 @@ import java.util.List;
 * The strategy that can receive 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 {

```

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 Olivers 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):

![Local Clone of Olivers Repository](image)

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

![Push Results: origin](image)

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:
We can also located them outside Eclipse using `git diff origin`:

```
 diff --git a/org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java b/org.eclipse.swordfish.api/src/
index 956d587..62ed0d5 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
@@ -15,7 +16,11 @@ 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 orgi service.
-<<<<<<< HEAD:org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java
+<<<<<<< HEAD:org.eclipse.swordfish.api/src/main/java/org/eclipse/swordfish/api/FilterStrategy.java
+    * GIT is GIT
+=======
+    */
+    public interface FilterStrategy extends Strategy {

```
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:

```
error: path not found
```

Now contents in our Eclipse editor change:
We resolve the conflict by keeping both changes and save our file:

That done we do not have to commit, but have to (re-) add 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:

```
$ git log
commit b6f255.6bc2976
Author: Juergen Kindler <juergen.kindler@aspers.de>
Date: Wed Apr 1 15:44:29 2009 +0000

  A conflicting change!

commit b6f255.6bc2976
Author: Juergen Kindler <juergen.kindler@aspers.de>
Date: Wed Apr 1 15:44:29 2009 +0000

  Added a Git test commit.
```

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).
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 ...”
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!):

![Reset Type](image1)

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

![Really reset?](image2)
So when the operation is finished, we see that our local master branch and the remote master branch are at the same state: