chem-bla-ics

Bioclipse git experiences #2: Create patches for individual plugins/ features

Egon Willighagen

Published July 3, 2020

Citation

Willighagen, E. (2020, July 3). Bioclipse git experiences #2: Create patches for individual plugins/ features. *Chem-bla-ics*. https://doi.org/10.59350/7ts20-k7s71

Keywords

Bioclipse, Git

This is a series of two posts repeating some content I wrote up back in the Bioclipse days (see also this Scholia page). They both deal with something we were facing: restructuring of version control repositories, while actually keeping the history. For example, you may want to copy or move code from one repository to another. A second use case can be a file that must be removed (there are valid reasons for that). Because these posts are based on Bioclipse work, there will be some specific terminology, but the approach I regularly apply in other situations.

This second post talks about how to migrate code from one repository to another.

Create patches for individual plugins/features

While the above works pretty well, a good alternative in situations where you only need to get a repository-with-history for a few plugins, is to use patch sets.

First, initialize a new git repository, e.g. bioclipse.rdf:

```
mkdir bioclipse.rdf
cd bioclipse.rdf
git init
nano README
git commit -m "Added README with some basic info about the new repository"
README
```

Then, for each plugin discover you need what the commit was where the plugins was first commited, using the git-svn repository created earlier:

```
cd your.gitsvn.checkout
git log --pretty=oneline externals/com.hp.hpl.jena/ | tail -1
```

Then create patches for the last tree before that last patch by appending `^1` to the commit hash. For example, the first patch of the Jena libraries was 06d0eb0542377f958d06892860ea3363e3316389, so I type:

```
rm 00*.patch
git format-patch 06d0eb0542377f958d06892860ea3363e3316389^1 -- externals/
com.hp.hpl.jena
```

(tune the filter when removing old patches if there are more than 99!)

The previous two steps can be combined into a Perl script:

```
#!/usr/bin/perl
use diagnostics;
use strict;

my $plugin = $ARGV[0];
```

```
if (!$plugin) {
  print "Syntax: gfp <plugin|feature>\n";
  exit(0);
}

die "Cannot find plugin or feature $plugin !" if (!(-e $plugin));

`rm -f *.patch`;
my $hash = `git log --follow --pretty=oneline $plugin | tail -1 | cut -d' ' -
f1`;
$hash =~ s/\n|\r//g;

print "Plugin: $plugin \n";
print "Hash: $hash \n";
`git format-patch $hash^1 -- $plugin`;
```

Move these patches into your new repository:

```
mv 00*.patch ../bioclipse.rdf
```

(tune the filter when moving the patches if there are more than 99! Also customize the target folder name to match your situation)

Apply the new patches in your new git repository:

```
cd ../bioclipse.rdf
git am 00*.patch
```

(You're on your own if that fails... and you may have to default to the other alternative then)

Repeat those two steps for all plugins you want in your new repository