Comparing Files from Different Branches with Git Difftool

Posted At : June 21, 2010 3:10 PM | Posted By : Bob Silverberg
Related Categories: Git, OS X

As a relative newcomer to Git one of the things I've struggled most with is how to compare files from different branches. The challenge comes from the fact that, from the perspective of the file system, two branches cannot exist at the same time. When you switch from one branch to another the new branch replaces the old branch, so you cannot use a native file compare tool to compare two sets of files, as there really is only one set of files at any point in time. Now I admit that I might be totally wrong about this, and I'm sure that there are other, perhaps better, solutions to the issue, but the one that works for me currently is *git difftool*.

What is Git Difftool?

According to the man page for git-difftool,

> *git difftool* is a git command that allows you to compare and edit files between revisions using common diff tools. *git difftool* is a frontend to *git diff* and accepts the same options and arguments.

I've tried using *git diff* in the past and, after spending years working with a wonderful tool like Subclipse's *Synchronize with Repository*, I just did not enjoy the output of *git diff* at all. Luckily, *git difftool* works with a file compare tool on your system, making the output much easier (for me at least) to deal with. On my system, which is OS X, because I have the Apple Developer Tools installed, when I issue the *git difftool* the output is sent to *opendiff*, which in turn uses *FileMerge* which is a nice, graphical file compare and merge tool. Other than installing the developer tools, which I did long before I started using Git, I didn't have to do any other setup. I honestly have no idea how easy it is to set up a graphical compare tool to work with *git difftool* on a Windows or Linux box, but I'm guessing it cannot be that difficult.

Using Git Difftool

To start a compare, you simply issue the *git difftool* command and pass it paths to two sets of files. The paths look like *branchName*:*path*. So if I wanted to compare the file *ValidationFactory.cfc* from the *master* branch to the same file in the *newStuff* branch, I'd type:

```
git difftool master:ValidationFactory.cfc newBranch:ValidationFactory.cfc
```

I'd see a prompt that says something like:

```
merge tool candidates: opendiff kdiff3 tkdiff xxdiff meld kompare gvimdiff diffuse ecmerge araxis emerge vimdiff

Viewing: 'master:ValidationFactory.cfc'

Hit return to launch 'opendiff':
```

And when I hit return FileMerge would open up with both files displayed. If I want to compare an entire folder, I can just type

```
git difftool master:ValidateThis/core/ newBranch:ValidateThis/core/
```

And then I receive that prompt for each individual file in turn.

I still don't think this is anywhere near as good as what I had with Subclipse, and I'm guessing there are ways to configure it to make it even friendlier, but for now it's much better than *git diff*.