IT OBSERVATIONS AND INSIGHTS

Information Technology commentary from an interested party.

HOME · TOP 100 PRODUCTIVITY TOOLS · MORE…

September 09, 2015

WHY FOSSIL-SCM IS AN EXCELLENT CHOICE FOR INTRODUCTORY PROGRAMMING COURSES

Fossil SCM for introductory programming courses

The use of source control management (or version control - take your pick) is an important skill for new programmers to adopt. It is expected that all programmers use SCM in their daily jobs, in order to coordinate changes among team members. Thus, getting beginners to adopt good habits early should be a goal.

While GIT (git-scm.com) is certainly the dominant source control system of today, I believe instructors of introductory classes in programming should consider an alternative called Fossil (fossil-scm.org).

Fossil has several compelling advantages in education over GIT. You will see that I value the practical aspects of Fossil even more than its technical capabilities. After all, an instructor has a limited amount of time to have an impact and they don't want to waste time doing technical support on a tool that is too complex. Helping one or two people is fine but helping 30 can be a real burden.

Simple installation and no dependencies

Fossil is a multi-platform program that is distributed as a single executable file. Nothing else is needed. This means that installation couldn't be easier and instructors won't have to fix broken installations. It also means it is easy to download and try new versions without having to uninstall old ones. Fossil uses a repository based on the battle-tested SQLite engine (and was written by the same principal author, D. Richard Hipp.) Out of the box, it is designed to use an HTTP based protocol without any add on programs. It also has a built-in graphical diff (both desktop and web).

Meanwhile, I was shocked to see that Git on Windows is 410 MB, with 286 MB of Mingw64, and git.exe is just 136kb. Git is very configurable but that also means it is dependent on a lot of external tools, like ssh (or putty on windows). Fossil is self-contained which means less hassle but also less flexibility. For this use case, it makes sense to me.

Simple consistent syntax

Fossil has a simple syntax that is consist and easy to remember. Git has several syntaxes available and while this provides flexibility, it can also be confusing for beginners. With fossil, just type:

```
fossil help
```

and you'll get immediate help.

```
fossil help command
```

will give you help on a given command. Built in, without the need for access to the Internet.

Distributed Version Control Systems (DVCS) are all very similar

Like GIT, Mercurial (https://mercurial.selenic.com/), and GNU Bazaar (http://bazaar.canonical.com/) , Fossil is a DVCS. This is now the most popular form of SCM. As a result, students are learning to think in terms of DVCS and many of the concepts in Fossil translate to other systems, like GIT.

Another advantage is that a DVCS doesn't require a central server. From a practical perspective, this means instructors don't have to teach students how to connect and authenticate to those servers. On the other hand, Fossil has workflows (autosync in particular) that can help teams that are working on a server. So you can have it both ways.

Fossil does not need any add on programs or plugins to work. So, you can diff, merge, branch, and edit to your heart's content. This also means less troubleshooting headaches for the instructor.

Fossil also does not allow rebase, like GIT. While the rebase capability has arguable advantages, it is possible to get into deep trouble. By avoiding this, Fossil retains an immutable history and simplity. If and when students are ready for rebase, they can always move to GIT. But by then, they are no longer beginners. With the popularity of Github, students will learn to use GIT eventually. Fossil is just another tool in their growing arsenal and can live comfortably besides GIT in their toolchest.

Fossil also supports importing from SVN and GIT, as well as exporting to GIT. So student code is not locked into Fossil, if that was a concern.

Built-in Tickets (Tracker) and Wiki

Unlike the other DVCS systems mentioned above, Fossil has a built in trouble ticketing system (also known as a "tracker" in Internet parlance.) *This is another good habit for students to learn* (not every student is an open source expert.) Because it is built in, instructors don't have to obtain accounts on a shared ticketing system and distributed credentials to the class. Each student can manage their own tickets. Instructors can incorporate good ticket tracking and writing into the class easily.

There is also a built-in Wiki. This makes writing documentation a snap and again, it is accessed by a simple command:

```
fossil ui
```

The repository can be hosted on a server and be serving files with a single command:

```
fossil server
```

SSL is supported as well (using your web server to act as the SSL/TLS listener, to be clear.) You can also serve requests via CGI or WSGI.

The UI can be used to review repository objects as well as trouble tickets, wikis, documentation, and commit histories (along with a graphical view of the branches).

One nice feature about Fossil is that you can have un-versioned resources in addition to versioned resources in the same repository.

Built-in Visual Diff

Comparing files and studying differences is a key practice in software development. Having a graphical diff program built-into Fossil means every student has the same program installed, regardless of platform, with integration into the repository. This is a time saver. Again, there are probably nicer, more functional diff programs but we're talking about the advantages of having the same tool everywhere without wasting time explaining how to install on 3 different platforms.

Single file repository

Fossil stores all files in a database that is a single file on disk. This makes distributing the repository a piece of cake. No need to zip up a repository. Just send the Fossil file which contains all source files, wiki pages, and trouble tickets! Instructors can see the full history of the homework assignment, including trouble tickets and wiki pages.

Adequate Performance

The one feature that GIT proponents will always raise is the speed of GIT. While it is true that GIT was optimized for performance and GIT is probably faster than Fossil, that is unlikely to be the primary reason for choosing GIT over Fossil for beginning programmers. For some performance numbers, check out this page over at the fossil site. I figure if they can store SQLite and NetBSD in Fossil, it's good enough for my needs.

Why not use Git?

At this point, you might conclude that all of this could be done with git and a combination of add on products. While this is true, fossil is just quicker and easier to get started with. Once you've learned Fossil, learning git is just not that big a deal. You will have learned the essentials of distributed version control and the differences between fossil and git are details. Git does have lots of tutorials and some of your more advanced students may already be using it. That's fine - they too will benefit from learning another DVCS.

I have heard another idea that teaching GIT prepares students for the "real world". Yes, this is undeniable. Except that most students won't get to see the depth of git during their coursework nor are their instructors likely to have encountered the full range of issues with git. They will stay on the well lit paths of adds, commits, and may be a merge with another student. But you can learn that in 10 minutes if you already know Fossil.

Conclusion

In conclusion, I feel that Fossil has all the right features for education. It is practical and easy to learn. While GIT is great, one size rarely fits all.

Give it a try at http://fossil-scm.org

Share **Email Post**

Labels: education, fossil-scm, professionalism