Subject: Re: Coyote Library Updates
Posted by Paul Van Delst[1] on Tue, 09 Aug 2011 19:47:31 GMT
View Forum Message <> Reply to Message

Hello,

David Fanning wrote:
> Paul van Delst writes:
>
>> Are you planning on creating svn tags of these releases? And, with apologies for the preaching, but how come you don't
>> create the zip files from said tagged release? That should avoid wrong files being copied.
>
> I've experimented (because of your nagging!) with
> tagged releases. It's probably because I don't fully
> understand what I am doing, but I don't like tagged
> releases. Mostly because two minutes after I tag
> a release someone reports a bug, and then the tagged
> release is obsolete

Well, don't blame subversion tags for that. :o)

> and SVN seems to not allow me to
> change it. I am embarrassed to create a NEW tagged
> release two minutes after the old one. It makes it
> seem (duh!) like I don't know what I am doing!

I don't think this applies to you, but I've found that to be a common feeling amongst people that resist using version
control: it immortalises all their mistakes.

My response to that is: meh. I tend to learn much more from my mistakes and use them as teaching moments (i.e. "What not
to do" slides in seminar presentations). Anyway...

>
>> Additionally, that would allow outside users of your package to update their own source code histories with the tag name
>> of your release. Or, in my case, export your tagged release into my own repo -- I'm not allowed to use svn:externals :o(
>>
>> Traceability and all that, you know?
>
> OK, here is how I work. I have two folders, Coyote and Zip_Files, under
> my IDL directory. When I make a change to a file that I think is a
> "significant" change (meaning, in practice, any change at all,
> because I am so damn anal!) I add the change to the coyoteprograms.zip
> file and I commit both the zip file and the program files to the Goggle

> SVN repository. Then, I "update" the same two folders that exist
> in my copy of my web page on my computer. After the update, I
> copy the updated files from my local web page to my real web page.
>
> I update Coyote Library routines frequently, probably at least once
> a week, and occasionally (when I'm not completely on my game)
> several times a day. The SVN repository includes the latest
> files. My web page contains the latest files. I would think that
> would be enough, but...

It is enough. But it really only makes sense to you because you use it all the time. Outside users just want a single,
reliable place to go to to get updates.
>
> What exactly do you want me to do? I'm interested in doing
> whatever works for people. Tell me how you would use these
> tagged releases, given what you know about my complete
> inability to serve up files that I know are out of date.
>
> Is it permissible to create tagged releases several times
> a day, or would I run out of release numbers. :-)

I guess it depends on your release, and numbering, strategy. I use 3 numbers: X.Y.Z

From my Coding Review and Acceptance policy document (cribbed from the WRF UCAR document) for the CRTM package I work on:

* A “major release” contains important new scientific and/or software engineering advancements that significantly
alter the behavior and/or performance of the CRTM. Numbering for major releases increments the first element of the version number. Examples: “CRTM 2.0.0”, “CRTM 3.0.0”.
Major releases will be
synchronized across all development subgroups and scheduled well in advance. There will be no more than
one major release per year.

* A “minor release” contains minor feature additions and enhancements. Model behavior is not fundamentally
altered. Numbering for minor releases increments the second element of the version number. Examples: “CRTM 2.1.0”, “CRTM 3.2.0”. Minor releases will be scheduled well in advance and will
be coordinated across all developers. However it is not required that every subgroup schedule minor releases
at the same time. This allows for quicker turnaround of minor feature enhancements by individual development subgroups.

* A “patch release” contains bug fixes. No new features are added. Coordination and scheduling are identical

to minor releases, except that less advance notice is required. This allows for quick turnaround of bug fixes
by individual development subgroups. Numbering for patch releases increments the third element of the
version number. Examples: “CRTM 2.1.1”, “CRTM 2.1.2”

Some package also tack on a patch number to a release, e.g. the current ruby release version is 1.9.2-p290. Yeah, 290
patches to release 1.9.2. No one thinks those folks don't know what they're doing. Have a look at:
  http://www.ruby-lang.org/en/

> And, how would these tagged releases be any different from
> the files that are already in the SVN repository? I mean, isn't
> this just extra work?

Yes. But a tag name (e.g. "release_2.1.6") tends to be easier to remember than a particular revision in the repository,
and it (generally) has more meaning to a development team. Additionally, the tag itself contains all the history
associated with the files.

As for the tagged releases being the same as the files in the repository, you can create mixed revision tags (I never
do, but some people like that feature) so even a fixed repository revision number might not be a good anchor for a release.

An anonymous zipfile name has no traceability and, more importantly, no associated history. You don't know what version
of your package users have when they ask about problems they're having. Even a date has no meaning if they downloaded
the zipfile before you fixed it.

Maybe you should look into using some sort of continuous integration tool to manage all that for you. There are plenty
third party ones out there. I would assume the google thing you use has something available. (I think I can hear you
groaning all the way across the country :o)

If that's too much for you right now, you could try creating generic tags, e.g. "nightly", or "latest", etc, by hand
(well, script) for potentially unstable code that you're not yet ready to release as a package. Basically you just
recreate these as needed. People know that if they go to the "latest" tag in your repo they're going to get the latest
release of code that you're mostly happy with. How to do it? Well, probably the simplest method is to simply recreate
the tag each time, e.g.

```
  svn move http://svn.repo.com/tags/latest http://svn.repo.com/tags/latest_YYYY-MM-DD
```

--message "Moving latest tag"
  svn copy http://svn.repo.com/trunk http://svn.repo.com/tags/latest --message "Creating latest tag"
After a sufficient period of time (depending on your schedule for "latest" updates) you could delete the oldest
"YYYY-MM-DD" tags (references to which still survive in your repo). Remember than a subversion move or copy doesn't
replicate all your source code - it just creates a database reference (effectively) to a particular snapshot.

Apologies for the monologue.

And the nagging.

cheers,

paulv

p.s. How do other IDLers out there handle their code updates?