Subversion Repository Layout

I see a lot of questions asked about "What is the recommended repository layout?", "What does trunk mean?", or: "What is the significance of trunk?". This post will try to answer those questions and more.

A Subversion repository implements the metaphor of a versioned filesystem. The repository is just a filesystem with folders and files. It so happens that modifications to this filesystem are versioned and there are implementation enhancements like "cheap" copies that make certain operations less expensive than they are in a traditional filesystem, but the repository itself still behaves like a filesystem: there are no special folders or names and Subversion itself has no knowledge of trunk or branches, they are just folders within the filesystem. It is up to you as the user to give those folders names and structure that are meaningful to you.

That said, there are several common layouts that have been adopted by the community as best practices and therefore one could think of these as recommendations. If your repository is accessible to the public, following these conventions might make it easier for users that have accessed other Subversion repositories to find what they are looking for.

There are two commonly used layouts:

```
trunk
branches
tags
```

This first layout is the best option for a repository that contains a single project or a set of projects that are tightly related to each other. This layout is useful because it is simple to branch or tag the entire project or a set of projects with a single command:

```
svn copy url://repos/trunk url://repos/tags/tagname -m "Create tagname"
```

This is probably the most commonly used repository layout and is used by many open source projects, like Subversion itself and Subclipse. This is the layout that most hosting sites like Tigris.org, SourceForge.net and Google Code follow as each project at these sites is given its own repository.

The next layout is the best option for a repository that contains unrelated or loosely related projects.

```
ProjectA
   trunk
   branches
   tags
ProjectB
   trunk
   branches
   tags
```

In this layout, each project receives a top-level folder and then the trunk/branches/tags folders are created beneath it. This is really the same layout as the first layout, it is just that instead of putting each project in its own repository, they are all in a single repository. The Apache Software Foundation uses this layout for their repository which contains all of their projects in one single

repository.

With this layout, each project has its own branches and tags and it is easy to create them for the files in that project using one command, similar to the one previously shown:

```
svn copy url://repos/ProjectA/trunk
url://repos/ProjectA/tags/tagname -m "Create tagname"
```

What you cannot easily do in this layout is create a branch or tag that contains files from both ProjectA and ProjectB. You can still do it, but it requires multiple commands and you also have to decide if you are going to make a special folder for the branches and tags that involve multiple projects. If you are going to need to do this a lot, you might want to consider the first layout.

As for the names of the folders within the repository, again: they are just a convention. They have no special meaning to Subversion.

"trunk" is supposed to signify the main development line for the project. You could call this "main" or "mainline" or "production" or whatever you like.

"branches" is obviously supposed to be a place to create branches. People use branches for a lot of purposes. You might want to separate your release or maintenance branches from your feature branches or your customer modification branches etc. In this case, you could create a layer of folders beneath branches, or just create multiple branch folders at the top-level.

"tags" are not treated as special by Subversion either. They are a convention, perhaps enforced by hook script or authz rules, that indicate you are creating a point in time snapshot. Typically the difference between tags and branches is that the former are not modified once they are created. You might call your tag folders "releases" or "snapshots" or "baselines" or whatever term you prefer.

Remember, the significance of the name is for your benefit, not for Subversion. Finally, the architecture of Subversion, with its global revision number can often make the need for tags unnecessary. I do not think there is any reason to create tags just for the sake of creating them. If you find a need to recreate the software at a specific point in time, you can always do so by using svn log to determine the relevant revision number. Tags are best when there are "external" consumers of the repository. Maybe it is a QA/Release team that needs to perform builds, maybe it is an internal development team that wants to use releases of your code in another product, or maybe it is literally external users or customers who need to grab release snapshots from your repository. In these scenarios, creating tags is both a convenient way to be sure they get the right code, as well as a good communication mechanism to indicate the presence of release snapshots.

Hopefully this post clarifies some of these issues for you and makes it easier for you to understand how Subversion works.

I would like to finish by pointing out that a Subversion repository layout can be changed. You can always reorganize and restructure your layout after the fact. At worst, it just might create some short term pain as users adjust their working copies. It is not like you need to start over though. Just change names, move folders or do whatever to get the filesystem looking the way you want it.