[Sublime Subversion](http://www.sublimesvn.com/blog) [Subscribe to RSS feed](http://www.sublimesvn.com/blog/feed/)

[What are branches, tags, and trunk in Subversion?](http://www.sublimesvn.com/blog/2009/11/what-are-branches-tags-and-trunk-in-subversion/)

Monday, November 23rd, 2009 | [Getting Started](http://www.sublimesvn.com/blog/category/sublime/getting-started/), [Subversion 101](http://www.sublimesvn.com/blog/category/subversion/subversion-101/) | [glenc](http://sublimesvn.com)

Even if you are new to Subversion you have likely heard terms like “trunk”, “branching” and “tags” thrown around.  If you are coming from a source control system like CVS you may already be familiar with these terms but if you are coming from a tool like Visual Source safe they may be new.  This post provides an introduction to these concepts in Subversion.  For more in-depth information please see [Chapter 4](http://svnbook.red-bean.com/en/1.5/svn.branchmerge.html) of the [Subversion Book](http://svnbook.red-bean.com/).

First things first – the **trunk**.  In Subversion, the term **trunk** refers to your main source code tree.  If you are starting out a new project, you will spend the majority of your time making changes and committing them to the **trunk** of your repository.  This is because early in a project there are fewer situations where you need branches or tags (there are certainly exceptions depending on your team and development practice, but this is an introductory article and we won’t get into that).

Now, let’s say your project is moving along and getting fairly stable, but you want to experiment with adding a new feature.  You could simply start making changes in your working copy and see how they work out without ever committing them, but this would be a mistake.  If this is a large set of changes your work could span multiple days – even weeks.  At some point your manager is going to wander over to your desk and inform you that there is a critical change that needs to be made **RIGHT NOW**. Now you’ll need to set aside those experimental changes, get a fresh copy of the trunk, commit those, somehow port those changes into your experimental code, etc etc.  It gets complicated fast.

Branching is specifically designed for situations like this.  A **branch** is a copy of your source code – typically a copy of the trunk – with a special name like “experimental feature”.  It sets up a space for you to work and make changes without committing to the main trunk.  Once the code in your branch is complete, you can **merge** your changes back to the trunk.  In addition, as new changes are committed to the main trunk, you can merge those changes into your branch so your branch is up to date with the latest changes from the trunk.  This is a great habit to get into by the way because it makes the process of merging your branch back to the trunk much much easier when the time comes.  Not to mention the fact that you are much more likely to catch potential integration issues early.

I’ll come back to branching in a bit, but now let’s move to tagging.

A **tag** is simply a named version of your code at a particular point in time.  For example, let’s say you’ve reached version 1.0 of your project.  Rather than having to remember that version 1.0 was revision 638, you can create a **tag** and give it the name of “1.0″.  Best practice is to create a tag for every major release or milestone (beta, 1.0, 1.1, 2.0, etc) but you can create tags for any point in time you think developers will care about in the future.

So we’ve covered the basic concepts, now I’m going to let you in on a little secret.  Ready?  In Subversion, there is literally nothing special about a branch, or a tag, or the trunk for that matter.  It’s just a directory in your repository which holds a version of your source code.  That’s it.  In most Subversion repositories the directory for storing branches is called, well “branches”, but that is just a naming convention.  You could call yours “superman” and Subversion would still behave exactly the same.

Here’s what I mean.  Most repositories will have the following structure at their root:  
/branches  
/tags  
/trunkEach path above is simply a directory in your repository.  Trunk is self explanatory – it contains your main development tree.  Branches is a folder which will contain sub-folders, one for each branch.  Tags is the same – a folder which will contain sub-folders for each tag.

Here’s what your repository might look like with several tags and branches:

/branches  
/branches/new\_feature <- branch for implementing a new feature  
/branches/1.0\_maint <- branch for doing maintenance on version 1.0  
/tags  
/tags/0.9 <- version 0.9 release  
/tags/1.0 <- version 1.0 release  
/trunkWhen you create a branch, or a tag, Subversion is simply copying your current code usually from the trunk (although not required) into a new folder in /branches or /tags. Once your code is copied, you can check out a new working copy from that location, make changes, commit, etc. Since your working copy points at the branch folder rather than the trunk, changes you make will only be committed to the branch and not to the trunk. The act of merging is essentially picking the changes you have made and moving them back into the trunk. Subversion gives us tools to make this easier, but it essentially boils down to just that.

Now, why is this important? Well for one it means you aren’t stuck with the standard naming convention of branches, tags, and trunk. If you prefer something like “variations”, “labels”, and “root” it is entirely up to you – Subversion won’t care. However, unless you have a compelling reason to change, it’s best to stick with branches, tags, and trunk because it is widely accepted as the standard naming convention.

Another reason this is important is that Subversion isn’t going to *limit* you from making mistakes like checking out a tag and making changes to it. Making changes to a tag goes against the entire purpose of a tag because a tag should be a static point in the history of your code – not something you make changes to. If you need to make changes to version 1.0 outside of your trunk (maintenance for example) you should make a new branch using your 1.0 tag as the source (step by step instructions is beyond the scope of this article).

So that’s it. **Branches**, **tags**, and **trunk** are simply standard folders in your repository set up to help you manage your development process. You can change these conventions if you wish, but you should have a solid reason for doing so. You can also add to this structure if you need additional controls for your development team or process.

Before closing I want to make one point about how your code is copied to a branch or a tag. Subversion is smart enough not to make an entire duplicate copy of your code each time you create a branch or tag. It simply creates a pointer to the revision your branch is based on. When you make a commit to the branch, Subversion stores the difference between the modified version and the original file. It does not create a full copy of the file.