Day-to-day with Subversion

Wednesday, May 7, 2008

.NET (/bsimser/Tags/.NET)

General Software Development (/bsimser/Tags/General%20Software%20Development)

How many times have you said "What version is in production?" or "Can we rebuild productio a bug and release an update?"

Better yet my favourite:

> "We're working on Feature Y so we can't fix the bug for Feature X. Doing so would mean w deploy part of Feature X and Y with the patch!"

These are typical problems with source control, patching, and keeping your working flowing. C hard to keep track of what's being worked on vs. what was already deployed. Sometimes you deploying something not tested or "ready for primetime". For example, at one point I was dep screens and we had to pass along explicit instructions to the QA folks to "not touch that butto because we hadn't finished the backend or our own testing. Of course, they touched it and log bug. Still, we often run into the problem of working on one set of features while testing anoth

Recently we've switched over (not fully yet, but most of the projects are going there) from TFS Subversion. TFS is just a bloody nightmare when it comes to trying to keep the trunk revision s while performing updates on branches and not getting into a merge from hell scenario, which sometimes typical when you have branches.

In doing the switch, we landed on a solution around branching code for new features and kee trunk clean. Branching is a hot topic in source control circles and has been known to start holy my past life (like a month ago) I avoided branches like the plague. This is probably due to the branching (and more importantly the merge back) in TFS and VSS was like a live enema. Not something you want to do every day.

However in working through the process in a few projects and experiencing the daily merge ro first-hand, it's become my friend and makes for building systems along a feature driven develo stream much easier. Here's how the process goes and all the details on each step.

Revision 1

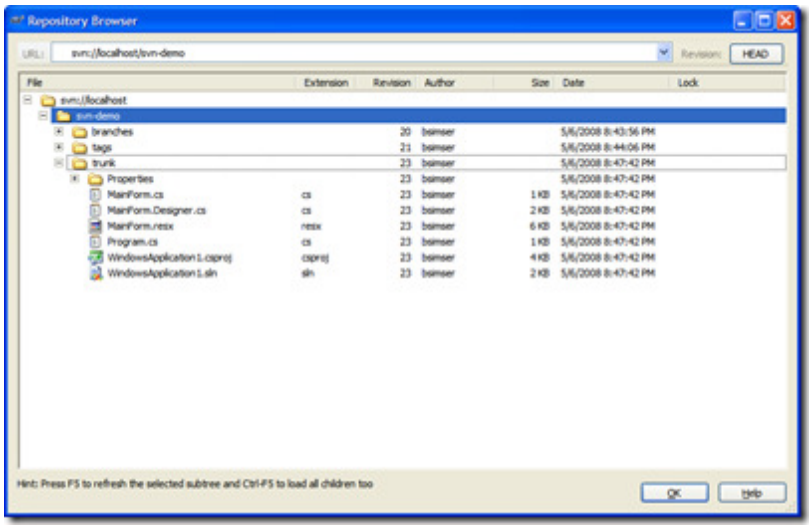
Code and screenshots are always the best way to work through a process. While the code here trivial (just a WinForms app with a few custom forms and dialogs) the principles are the same matter how big your project is.

First we setup our subversion repository for the project. The typical setup is to create three fol the repository; branches, tags, and trunk. Branches hold any branches you work on for new fea development; Tags contains named copies of revisions representing some point in time (perha deployment); Trunk contains the main codebase and is always stable. These become vital to organizing your code and not clobbering other work going on as we'll see as go along.

Here's our sample repository in TortoiseSVN:

image (https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_40.png)

We'll start with revision 1, the basic application (the proverbial WinForms "Hello World!"). A sir application with a single form. Check this in to Subversion into the trunk path. This gives us an updated repository:



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_41.png)

Now your day to day work begins. The trunk revision is the most important (aka "The King"). A other work being done will happen in branches and are known as servants. Servants are impor they take less priority than The King. The most important and highest priority work being don King (and there is only one king, viva Las Vegas baby!).

Fast forward to day 10 of our development cycle. We've been adding forms and code (all com to the trunk by various people) and it's time to do a release. A release is cut (using whatever p you use here, the details are not important) and deployed. At that point we want to tag the re

Tag and Deploy

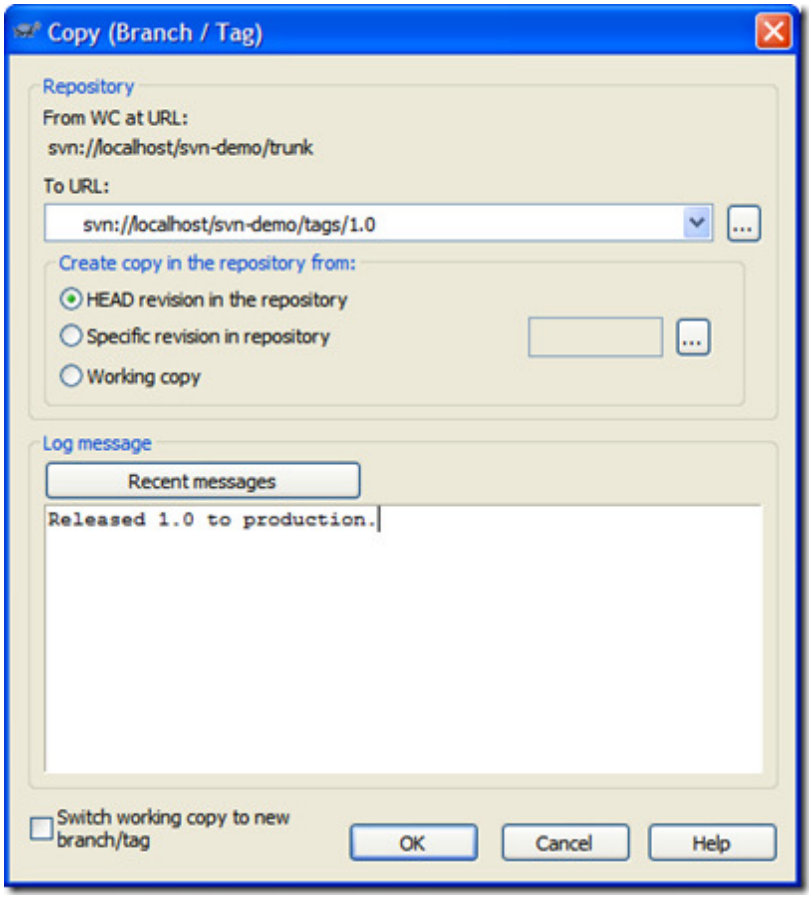
Tagging is a way to identify a set of code, a snapshot, so you can retrieve it later. Once tagged

go back to the revision and all files from that point in time to rebuild the system. This is mainly deployment thing. For example, you tag the release "1.0" and then continue on. At some point future you can check the code out using that tag, rebuild it, and it will be the same as the day deployed it.

We'll tag our release as "1.0". This creates what looks like an entire copy of the code in the "tag folder, but in reality it's all virtual. Unlike "other" source control systems, this doesn't actually m copy and the magic of Subversion will let us pull this tag out and all the code associated with later.

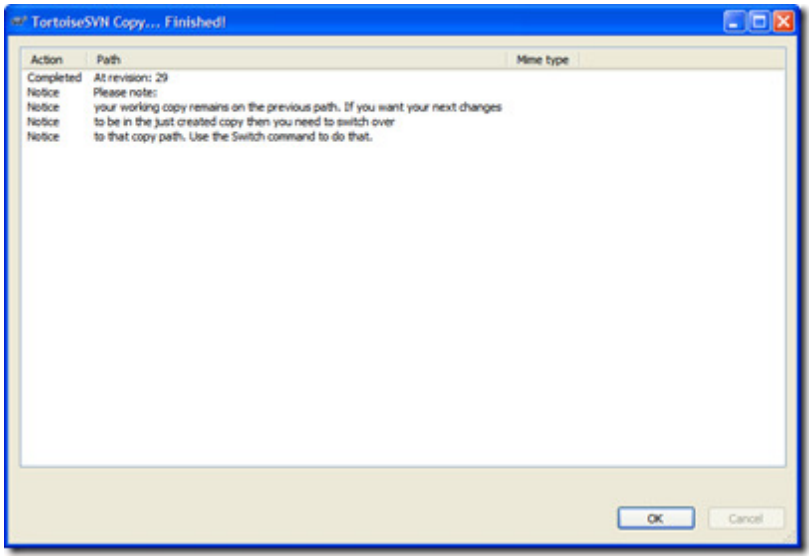
To tagging and creating branches is essentially the same act (it's the same dialog box) but will where you put the tag. Subversion does not have special commands for branching or tagging, uses so-called cheap copies instead. Cheap copies are similar to hard links in Unix, which mean instead of making a complete copy in the repository, an internal link is created, pointing to a s tree/revision. As a result branches and tags are very quick to create, and take up almost no ext space in the repository.

So while the dialog box says "Copy" you're creating this cheap copy. Don't get miffed if you're is huge, tagging takes next to nothing. Here's our tag ready to go:



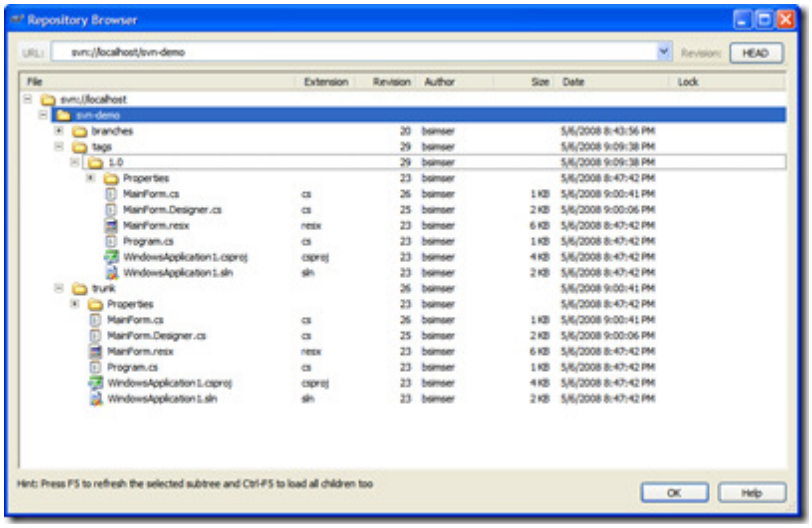
(https://mscblogs.blob.core.window /media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_6.png)

For tagging, you generally won't want to click on the "Switch working copy to new branch/tag checkbox. Tags are just snapshots in time and you go along your merry way in the trunk. For b we'll be doing something different. So after you create the tag, don't be alarmed when you se message in TortoiseSVN:



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_8.png)

And here's the repository tree after the tag. Note the tags folder has a new entry, "1.0" which c
an exact copy of what's in the "trunk", our King.



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_10.png)

Now comes the fun. We've tagged the work and deployed. At any point in time we can go bac
redeploy this version by pulling out the "1.0" tag and building/deploying from there. At this p
where we branch. We want to work in a new feature set. This is going to involve new dialogs a
code.

Branching New Features

Why do we branch? Isn't branching bad?

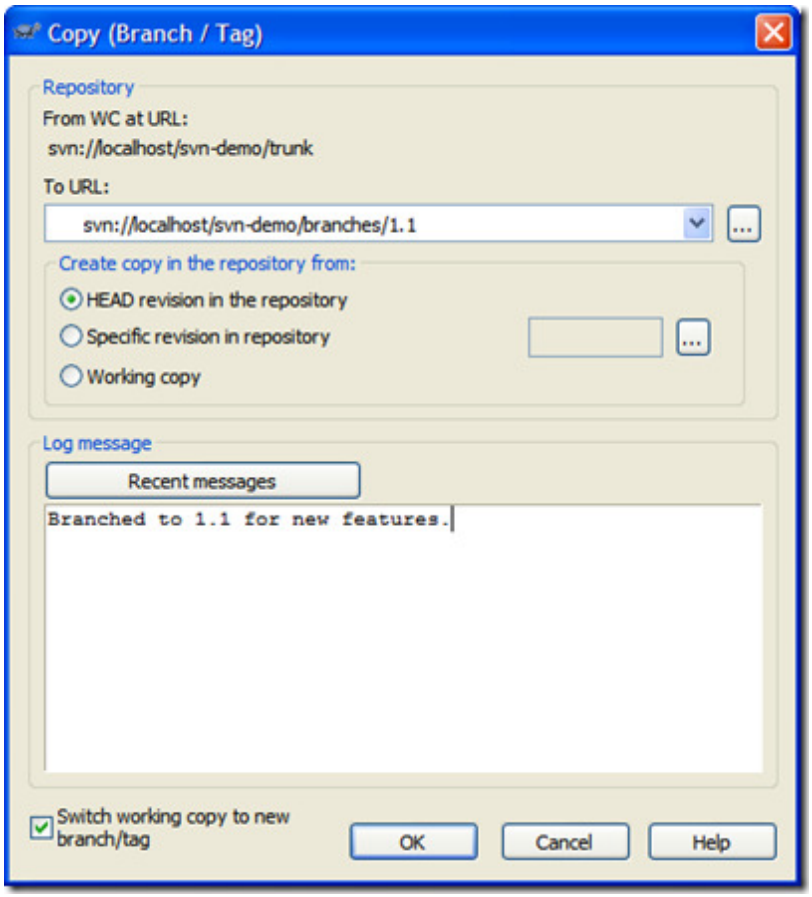
No. Branching, when used this way keeps your trunk clean. Remember, there can only be one
(trunk). Any other work is a servant and will eventually go into the trunk.

Why again do we branch? Imagine if we didn't branch. So right after you apply the "1.0" tag st
modifying trunk. Sure, we can go back to "1.0" but how are we going to get any changes merg
together when we're on a single line? We're also violating the "One King" rule. Who's the King
Our new branch becomes a servant. The King still takes priority (for example to fix bugs) but w
continue on in the servant branch.

Walk with me on this, by the end you'll see what the branch is for and why we want it.

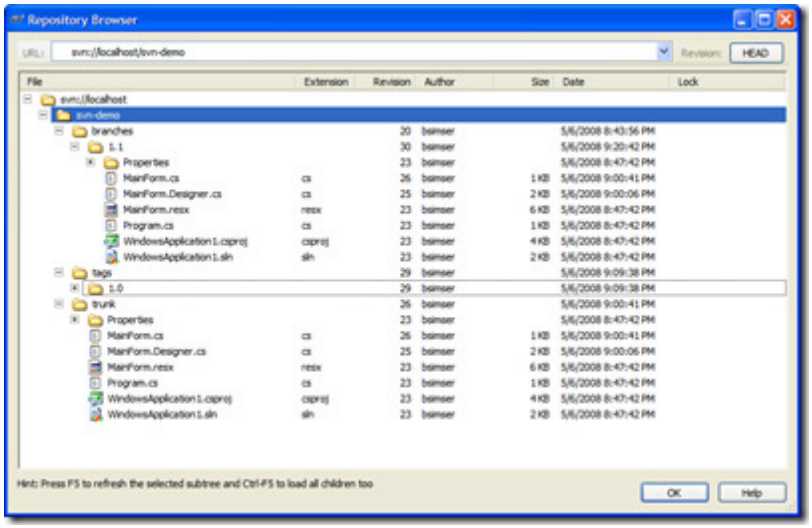
We'll create a new branch just like creating a tag. Call the branch "1.1" except in this case, we'r

to switch to the branch as our working copy. Here's the branch dialog:



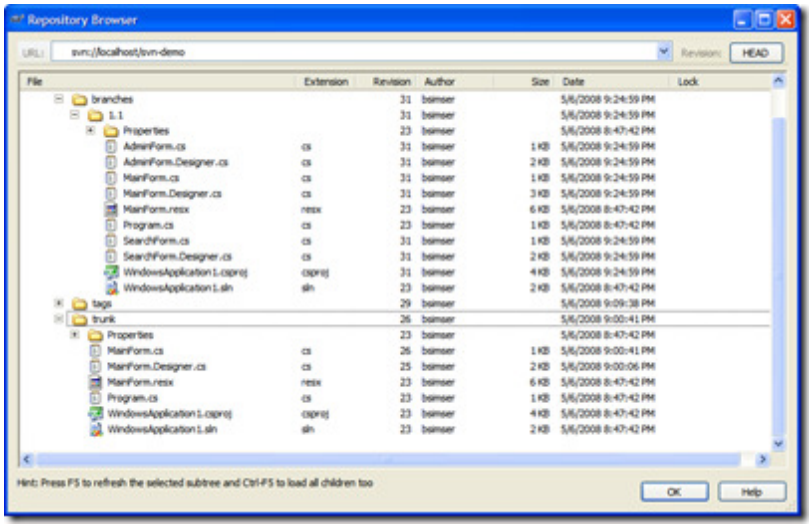
(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_12.png)

And here's the repository after the branch. Our work is now all going to be committed to the "svn-demo/branches/1.1" branch, keeping the trunk clean.



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_14.png)

Work in the 1.1 branch is underway with new features being added. We've created a few new f
modified the main form, and generally added new functionality. The 1.1 branch is quite differe
the original trunk it came from now:



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_16.png)

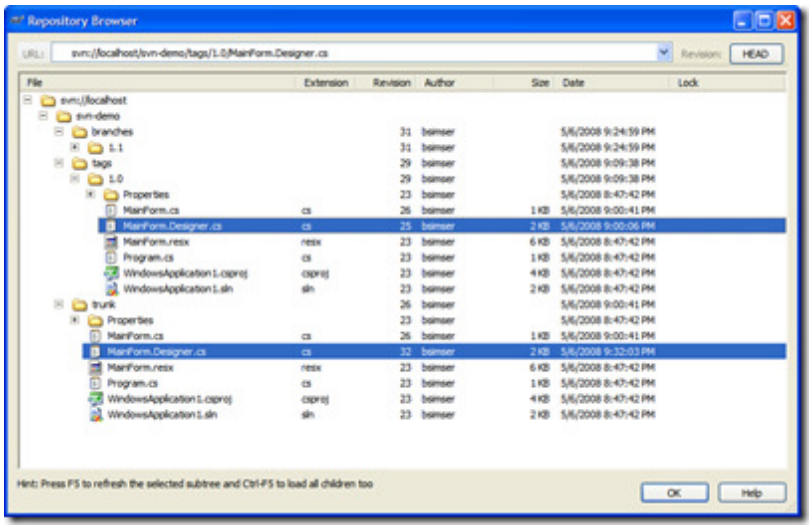
A couple of scenarios will arise out of this. For example, if there's a bug found in the 1.0 versio deployed what do you do? You don't want to dirty the 1.0 tag. That's why trunk is King (and th only one King). "trunk" is still the most important thing being worked on (at this point its in te production or whatever). Until it's verified, everyone else is a servant. Any problems found in " can be resolved on trunk. So we'll explore that scenario.

Waiter, There's a Bug in my Trunk!

There's a problem with 1.0. The window title is wrong. It reads "Hello World!" but it should rea World?".

Huge problem! Stop the presses. Halt the line. We need to fix this now!

You may be tempted to create a branch, fix it, then merge the branch back into trunk. This mig normal, but our trunk is clean so we can just work with it directly. Check out a copy of trunk to directory and we'll do the fix. Then commit it back. Now here's the updated repository:

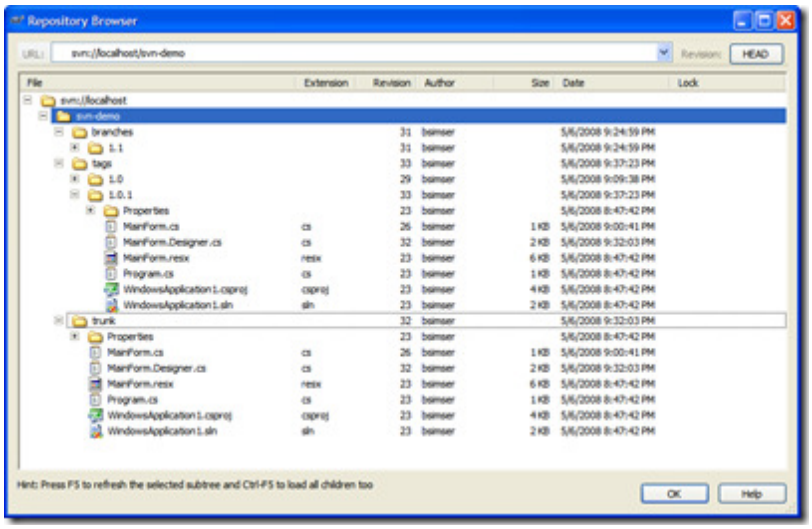


(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_18.png)

I've highlighted the file that changed in both versions. "/tags/1.0" is our deployed version (revi 25), "/trunk" is our bug fix update (revision 32). We can still, at any point, re-deploy "1.0" withc problems.

We'll do a deploy of our new trunk (which we'll call "1.0.1") and a series of exhaustive and inte tests beings. Weeks pass testing our massive change and finally QA accepts the version and al be deployed to production. This will replace "1.0" in production with "1.0.1" and the updated t

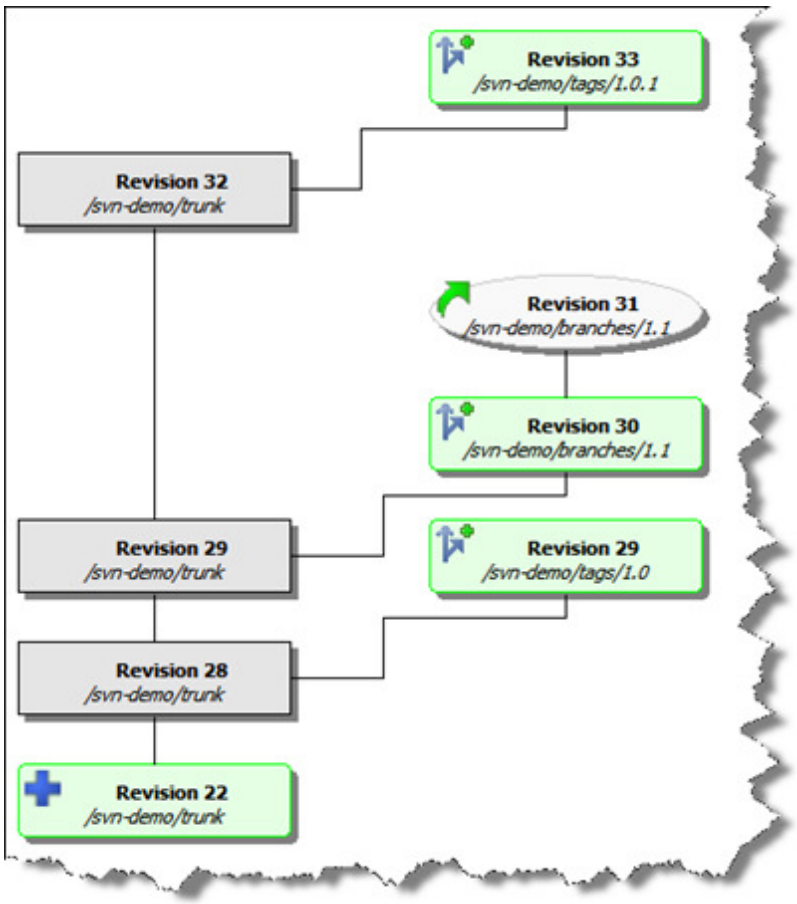
Tag trunk as "1.0.1" like we did "1.0" above and we'll now have this in our repository:



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_23.png)

The Graph is your Friend

TortoiseSVN has a wonderful feature called "Revision Graph" which gives you a visual tree of y
branches and tags and revisions. You will live and die by this tool. Here's ours so far:



From this visual we can assess:

- A tag called "1.0" was created from the trunk at revision 28, creating revision 29
- A branch called "1.1" was created from the trunk at revision 29
- Work continues on the "1.1" branch with daily commits (so far at revision 31)
- A tag called "1.0.1" was created from the trunk (after a bug fix) at revision 32, creating re
33

At this point I want to point out some major advantages with this approach:

- You can rebuild any deployed release easily (well, as long as you tagged it in the first pla
- Fixes can be done to the trunk and deployed quickly
- Work can continue on separate features without disturbing the main work

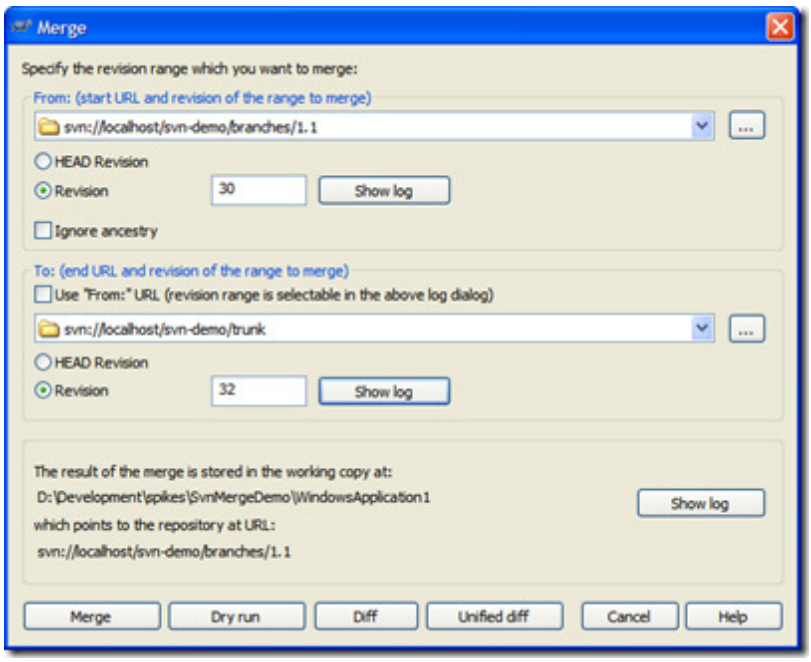
Day to Day Merges

So now we have a bit of a disconnect don't we? The trunk (revision 32) and the re-deployed ta version (1.0.1, revision 33) contains the fix we need however we're working on Feature X in the branch. We don't have that fix. If we were to merge our code back to the trunk (which we will do at some point) we might miss this fix, or worse yet clobber it.

To avoid this problem, anyone working in a branch follows one simple rule. Each day (say at th of the day) you update your branch from the trunk. In other words, you pick up any changes tl been applied to the trunk into your little branched world. Doing this will avoid any merge issu you commit your branch back to the trunk.

We do this with a merge. It's a simple merge but one that has to happen, and merges can get complicated and ugly. In your working directory where you're commits are happening on the l you won't see changes to trunk.

Here's the merge dialog that we perform on a daily basis. We'll merge changes from the trunk the 1.1 branch:



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_28.png)

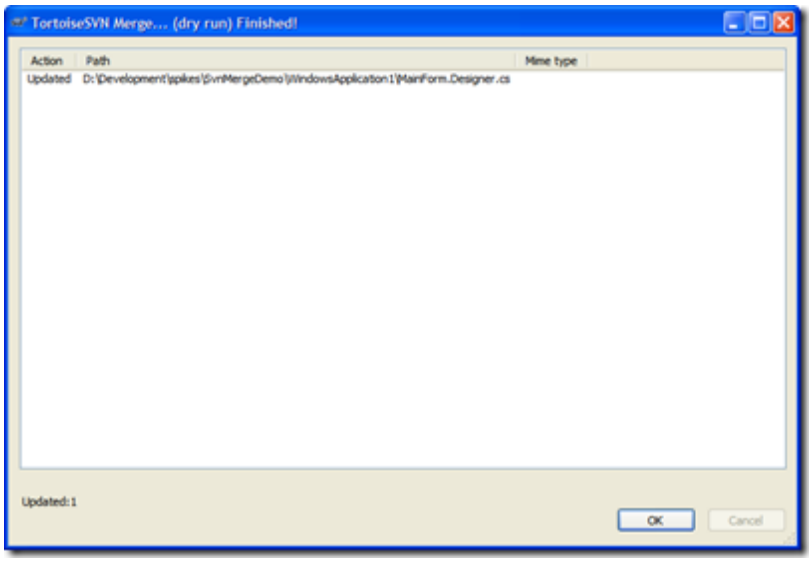
A few notes about this merge:

- We merge from the branch and specify the branch in the top section. This seems backwa we're merging "from a point in time" which needs to be the last revision when the two tr (trunk and branch) were synchronized. Remember, we're looking for all the changes "fro to branch" since we branched. The revision graph is essential in determining this. In our this is our first sync and is when we created the branch (revision 30).
- By default the merge to uses the "From" value but we want to merge into our branch so uncheck this and pick the trunk in the "To" section. For the trunk we're going to pick the revision but this happens to be revision 32. Picking either HEAD or revision 32 here resul same merge.

- Confirm the location and behaviour you expect in the bottom section. The working copy be your current working folder, and it should end up pointing at your current branch
- Always (always) do a Dry run first and confirm the updates your going to do are correct.

So in this merge we expect to get the changes to MainForm.Designer.cs (that title change bug had selected the HEAD revision for our branch version rather than the time where the branch from trunk, we would be comparing all the changes. This would result in the dry run telling us have new forms. This is incorrect because a) we only want the changes from trunk and b) trunk doesn't know (or need to know) about any new forms we created. We're only interested in the changes made on trunk that we don't have yet.

Here's the dry run dialog with the proper response (based on the last merge dialog):



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_30.png)

Perfect! We just want the changes to MainForm.Designer.cs (or whatever files changed since w sync' d) and we got them. Execute this to get the new changes from trunk into your branch.

When you do a merge, you're merging those changes into your working copy but you're still c more step away. This will modify your working code but now you have to commit it back to th repository. If you check your updates you'll see that the MainForm.Designer.cs file has change Here's the unified diff of the changes:

```
Index: D:/Development/spikes/SvnMergeDemo/WindowsApplication1/MainForm.Designer.cs
===================================================================
--- D:/Development/spikes/SvnMergeDemo/WindowsApplication1/MainForm.Designer.cs    (revision 31)
+++ D:/Development/spikes/SvnMergeDemo/WindowsApplication1/MainForm.Designer.cs    (working copy)
@@ -61,7 +61,7 @@
             this.Controls.Add(this.button1);
             this.Name = "MainForm";
             this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
-            this.Text = "Hello World!";
+            this.Text = "Hello World?";
             this.Load += new System.EventHandler(this.MainForm_Load);
             this.ResumeLayout(false);
```

As you can see, the title bar change is here and replaces our old (buggy) version.

Commit this to the repository. Your branch now has the changes from trunk and you can cont with your new feature work.

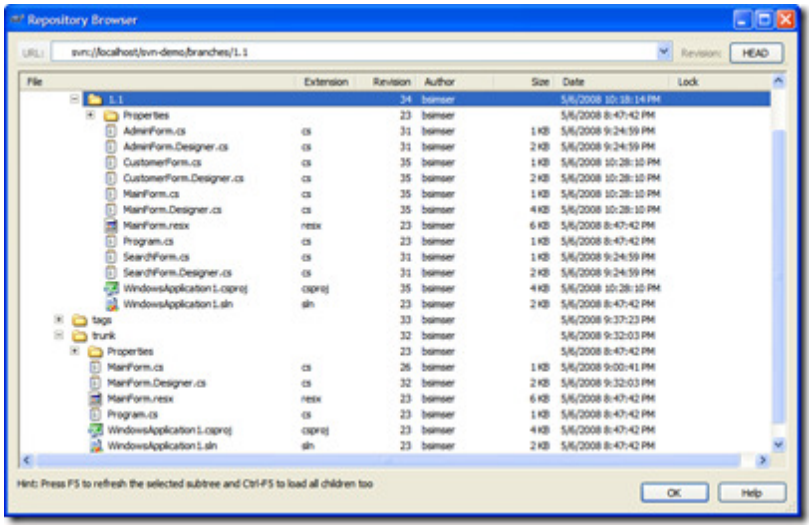
Remember, the key point of working in the branch is we don't pollute the trunk with our new or code, yet doing this daily merge (which will take all of 5 minutes on any codebase, trust me your branch up to date with any changes that may have happened.

Getting back to trunk

As we continue with our day to day work in the 1.1 branch, more changes might happen with More bug fixes, etc. However we don't introduce new features. We only add things on our bra the rare instance we're building a new feature while another feature is in play, we might create another branch with another team. I would however keep the number of active branches goin a minimum. It'll just get ugly later in life.

In any case, we continue with our branch until we're ready to deploy. At this point we probabl stable trunk (we should always have a stable trunk) with a number of tags. All changes in the t in our branch and the team has decided it's time to deploy a new version to replace 1.0.1. This 1.1 branch and we need to merge all the new stuff in 1.1 back into trunk.

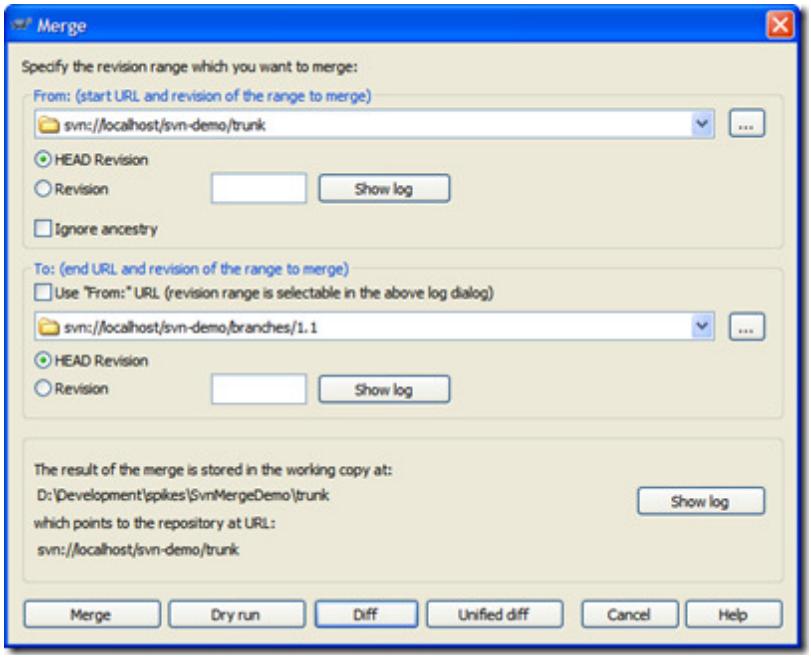
Here's our repository as it stands:



(https://mscblogs.blob.core.window /media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_32.png)

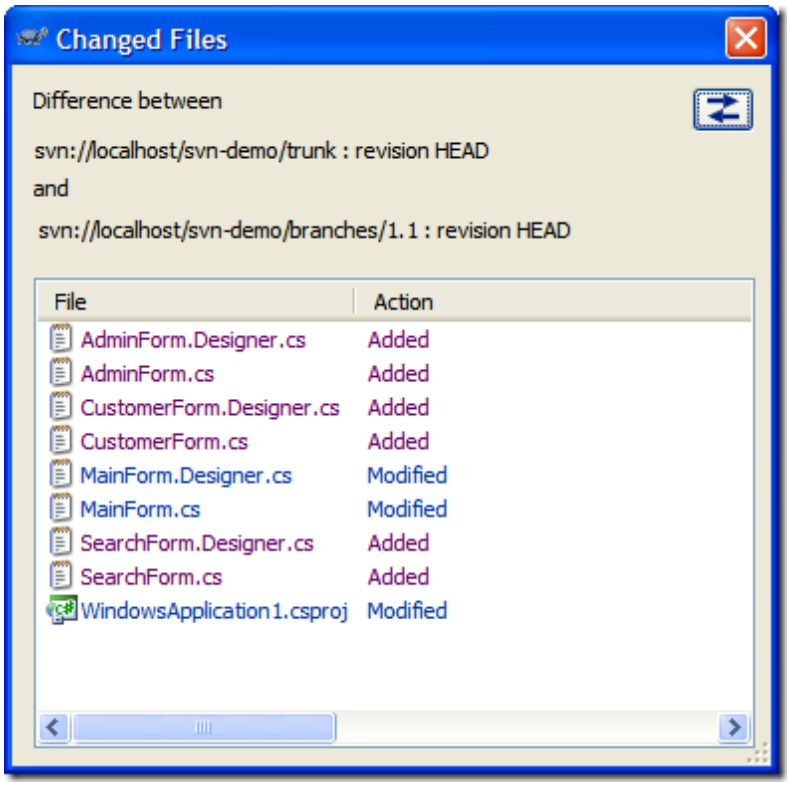
- The 1.1 branch contains all of our new work, 3 additional forms and some changes to the form to invoke our new forms
- As a result of our daily "merge from trunk" routine, we have any bug fixes or changes tha done in trunk
- Our trunk is clean and the version that was deployed (with various tags in our tags folde

To merge back into the trunk it's the opposite of what we do on a daily basis. Rather than mer into the branch, we reverse it and merge into trunk. Also, you'll need a working copy of trunk merge into. Check out trunk into a folder and invoke the merge. Again, the key point here is to the right revision. For the branch it'll be the HEAD revision. For trunk, it's the last point of synchronization which in this case is revision 32. Here's the merge dialog to commit our 1.1. fe to the trunk.



(https://mscblogs.blob.core.window
/media/bsimser/WindowsLiveWriter/SubversionDaytoday_12266/image_34.png)

In this case, we're committing to a working folder with a copy of trunk checked out to it. Click to see what changes are going to be applied:



Here we've added our new forms and there's changes to the MainForm.cs and MainForm.Desi (we've added buttons to invoke the new dialogs). Here's the unified diff of MainForm.Designe (with some lines removed for brevity):

```
Index: MainForm.Designer.cs
===================================================================
--- MainForm.Designer.cs    (.../trunk)    (revision 35)
+++ MainForm.Designer.cs    (.../branches/1.1)    (revision 35)
@@ -28,13 +28,49 @@
        /// </summary>
        private void InitializeComponent()
        {
+           this.button1 = new System.Windows.Forms.Button();
+           this.button2 = new System.Windows.Forms.Button();
+           this.button3 = new System.Windows.Forms.Button();
            this.SuspendLayout();
            //
+           // button1
+           //
+           this.button1.Location = new System.Drawing.Point(12, 12);
+           this.button1.Text = "Search";
+           //
+           // button2
+           //
+           this.button2.Location = new System.Drawing.Point(12, 41);
+           this.button2.Text = "Admin";
+           //
+           // button3
+           //
+           this.button3.Location = new System.Drawing.Point(12, 70);
+           this.button3.Text = "Customers";
+           //
            // MainForm
            //
+           this.Controls.Add(this.button3);
+           this.Controls.Add(this.button2);
+           this.Controls.Add(this.button1);
            this.Name = "MainForm";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Hello World?";
@@ -44,6 +80,10 @@
        }
        #endregion
+
+       private System.Windows.Forms.Button button1;
+       private System.Windows.Forms.Button button2;
+       private System.Windows.Forms.Button button3;
    }
```

```
}
```

Note towards the bottom of this diff, this.Text = "Hello World?". This was the result of our dail
so there's nothing to be applied back to trunk. We're in sync here. Only the changes/additions
/deletions are applied which will bring "trunk" up to par with the 1.1 branch work. Again, do y
run. You should see only the new work done in the branch as being applied to trunk. If not; st
drop, and roll and recheck your revisions.

Again, the trunk now is merged together with the 1.1 branch. At this point you'll want to load
solution up, build it, run unit tests, etc. and do a sanity check that everything works as expecte
would probably do your deployment and tag the new trunk as "1.1".

You can just simply ditch the branch folder or leave it there in the repository. After all, it's just
symbolic link and doesn't take up much space (we have a new tag created in our repository or
CruiseControl.NET build so there are hundreds of tags, no big deal).

Lather, Rinse, Repeat

Now you're back on the trunk. Trunk is King, there is only one King, and your day to day work
continues with whatever feature you're working on. You have the option to "always live in the
which might be an idea but this requires that daily merge from trunk and could cause problem
There's no problem "running on trunk" and building from it. The point at which you branch sh
when you do a release and want to continue on with new (different) work, otherwise daily com
trunk by the entire team is fine.

When a new feature comes along, branch, move part of the team (or the entire team) to that b
and keep trunk clean, doing any bug fixes as they come up. Then merge back from the feature
back into trunk at the appropriate time. Keep doing this as often as necessary, until you run ou
money or the team quits. Sit back, relax, and enjoy the simplicity of life.

Conclusion

It may seem complicated but it's really pretty basic when you boil it down. Just follow a few sir
rules:

- On a daily basis, developers in branches merge changes from the trunk into their branch
- Merge branch features back into trunk when you're ready to deploy
- Bug fixes are performed on the trunk then tagged and re-deployed

Give it a shot, email me if you're stuck or lost, or let me know what your experiences are.

Enjoy!

40 comments

Phew, nice job. I understand the basic mechanics, but this takes it a lot further than I eve
and makes it understandable!

— **Garrett Fitzgerald (http://blog.donnael.com/)** - Tuesday, May 6, 2008 11:55:57 PM (/bsimser/day-to-day-with-subversion#comment-1506)

Hi Bill,

Great article !! one thing to add is the notation of VSTS style shelfs, this is easily done in S
using the following - http://wiki.littlebluefroglabs.com
/index.php?title=Subversion_and_Shelving.

I suppose one thing that is also worth adding is what is SVN and why use SVN for those th
have not seen it before.

Andy

— **Andy Stopford** - Wednesday, May 7, 2008 12:52:53 AM (/bsimser/day-to-day-with-subversion#comment-1507)

That's really good stuff.
I used it for my own day to day development and now the company where I work uses it

Now we do not chase each other for updated or missing files because of that.

— **ferrymeidianto** - Wednesday, May 7, 2008 1:18:15 AM (/bsimser/day-to-day-with-subversion#comment-1508)

Good content but a little confusing.
You do cite alternative models for branching but you might want to make it a little bit mor explicit.

For example, here are two models I'm familiar with:

A. Release Branches
- New work goes in trunk
- Create a branch when freezing a release candidate
- Perform maintenance work on the frozen release branches (and copy back to trunk)
- Release issued from branch
- Continue new work in trunk

B. Work branches
- Trunk is kept clean and release-worthy at all times
- New features developed in branches
- Branches merged into trunk when features are ready
- Release issued from trunk

We use option A which works quite well and there's never any confusion about where new should go since it always goes in trunk. Only maintenance work on pending releases go in branches.

— **Jeff Brown (http://blog.bits-in-motion.com/)** - Wednesday, May 7, 2008 1:20:14 AM (/bsimser/day-to-day-with-subversion#comment-1509)

Truly a great post.

— **David Evans (http://www.contract5.com/BusinessSolutions/Pages/DavidEvans.aspx)** - Wednesday, May 7, 2008 2:12:28 AM (/bsimser/day-to-day-with-subversion#comment-1510)

+1 for release branches, not feature branches.

— **Codex** - Wednesday, May 7, 2008 4:58:07 AM (/bsimser/day-to-day-with-subversion#comment-1511)

Awesome post on how to use SVN. This is definitely a favorites page for SVN noobs like m

— **Ira (http://blog.reamped.net/)** - Wednesday, May 7, 2008 7:13:32 AM (/bsimser/day-to-day-with-subversion#comment-

I've found it hard to have people let go of their financial investment in TFS...

Also the perception of a tool that costs money vs. a free tool. The costly one _must_ be be

Good writeup though, I've forwarded it to my team (small jabs, ya know)

— **subdigital (http://www.flux88.com)** - Wednesday, May 7, 2008 7:51:46 AM (/bsimser/day-to-day-with-subversion#comment-1514)

Thanks Bill. Great article.

— **Alper** - Wednesday, May 7, 2008 8:08:01 AM (/bsimser/day-to-day-with-subversion#comment-1515)

+1 for release branches...

— **Simone (http://codeclimber.net.nz)** - Wednesday, May 7, 2008 9:29:30 AM (/bsimser/day-to-day-with-subversion#comment-1516)

+1 for release branches...

— **Shoaib** - Wednesday, May 7, 2008 10:27:27 AM (/bsimser/day-to-day-with-subversion#comment-1517)

Great post. Much thanks.

— **Ted Jardine (http://www.ovalsquare.com)** - Wednesday, May 7, 2008 12:15:28 PM (/bsimser/day-to-day-with-subversion#comment-1518)

Great writeup - well done in explaining the "day-to-day" working with Subversion.

I've been using SVN for a year now for one of my projects, only doing work in trunk. Have recently created a branch for adding in some unit tests (a more recent adventure). Have a made changes to the trunk in that time, so will need to merge trunk to the branch. Now I a good idea of how that will be done!

And I like the discussion over branching models, and I agree that release branches sound the better way to go. I will try to do that in the future. I think my current unit tests branch more of a feature branch!

— **Grant Palin (http://grantpalin.com)** - Wednesday, May 7, 2008 12:16:56 PM (/bsimser/day-to-day-with-subversion#comment-1519)

Thanks for perfect article!!!

p.s. I think you should swap merge screenshots :)

— **andrex** - Wednesday, May 7, 2008 2:55:37 PM (/bsimser/day-to-day-with-subversion#comment-1520)

That clarified and highlighted some ways of working with version control that I hadn't prop considered before.

-dave

— **David Gardiner (http://davidgardiner.blogspot.com)** - Wednesday, May 7, 2008 5:40:33 PM (/bsimser/day-to-day-witl subversion#comment-1521)

Release branches are the way to go and are far less work for the developer to maintain th feature branches. I only recommend feature branches if the project is extremely large and danger of becoming broken (or has had a history of being broken) by someone's trunk up Also, you can start with release branches and switch to feature branches later on in the project's lifecycle if the project size warrants it and it becomes a problem for your team.

— **John** - Wednesday, May 7, 2008 6:42:11 PM (/bsimser/day-to-day-with-subversion#comment-1522)

Bill,

What happens if you deleted/renamed a file in the branch in a refactoring cycle? How can merge any changes done to that file in the trunk?

Similarly, let us say you modified the namespace names that are referenced (via using statements) in a source file in trunk. Now you are trying to merge some other changes in same file from trunk. When you run the merge operation, SVN overrides the using stateme with old ones. It would take more than 5 minutes to fix all these breaking changes.

I am interested to know whether you have any tips in addressing these issues.

— **Kiran** - Wednesday, May 7, 2008 7:56:20 PM (/bsimser/day-to-day-with-subversion#comment-1523)

Very nice article on branching. Doesn't matter if you are using Subversion or not, the conc are the same (and just as powerful in other tools as well, just not TFS).

— **Jason Short (http://www.VistaDB.Net)** - Wednesday, May 7, 2008 8:36:54 PM (/bsimser/day-to-day-with-subversion#comment-1524)

@Kiran: You'll pick up the deletion (or any other changes) in your daily merge from trunk. problems there. If you're doing a *big* refactoring (like a domain shift or a massive namechange) you might want to co-ordinate with the team. Tell them the intent and talk it (most times something like this is needed and with a good refactoring tool and unit tests shouldn't worry). Get them all to merge up then do the change and get them to re-merge the trunk.

— **Bil Simser (http://weblogs.asp.net/bsimser)** - Thursday, May 8, 2008 7:48:52 AM (/bsimser/day-to-day-with-subversion#comment-1525)

@Cory: Moving from SVN to TFS is IMHO bad. I won't get into a holy war here about the b source control (as I don't feel is SVN is the "best" but think it's "better" than TFS). WIth Tl have this crazy workspace/local drive mapping issue which has caused me pain and suffer With SVN I just move a directory somewhere. It doesn't care. TFS requires me to tell the s to map my local drive to some location it knows about. And when a person leaves the com the workspace lingers and is a PITA to get rid of. Enough on that (for now).

As for SVN and CC.NET, not sure if that's worthy of an entire blog post. It's just a task in y ccnet.config file and is supported OOTB. How to track bugs and issues might be interestin maybe I'll look at a post on Developer->Bug Tool->SVN->CC.NET or something like that.

Thanks.

— **Bil Simser (http://weblogs.asp.net/bsimser)** - Thursday, May 8, 2008 7:53:33 AM (/bsimser/day-to-day-with-subversion#comment-1526)

Thank you very much for this great article that explains feature branching very clearly! An together with the comments you get a really good picture of the two most common branc strategies. One of the best SVN tutorials I've read so far.

— **Florian Potschka (http://www.potschka-it.de)** - Thursday, May 8, 2008 12:50:57 PM (/bsimser/day-to-day-with-subversion#comment-1527)

Hello.

Nice work Bil.

I have one question that keeps bothering me with this approach.

for instance, suppose we've release v1.0 and we've also release v2.0, which has lots of ne features. This means you'll have something like this:

- tags
- v1.0
- v2.0
- trunk
- v2.0 files
- branches
- v2.0 branch code

the trunk is the king and we've already updated it to 2.0 and have already put it on the we So, now the latest version is 2.0 but the client has to pay to update to it or else he must k using 1.0.

the problem: how do we solve a 1.0 error that is found after publishing the 2.0 version? W with the model you're using and when there was only a 1.0 version, solving it would be ea just correct the trunk and publish it to a new tag (like you've shown on your post).

but now that the trunk has been updated to 2.0, how do you solve this problem withouth changing the existing 1.0 snapshop that exists on the tags folder?

thanks.

— **Luis Abreu (http://msmvps.com/blogs/luisabreu)** - Wednesday, May 28, 2008 8:13:28 AM (/bsimser/day-to-day-with-subversion#comment-1528)

Luis:

My boss and I had a discussion about this, and we concluded that the crux of the matter i
By choosing to support two separate major versions (whether for backwards compatibility,
stingy customers, or dire security holes), you are ultimately maintaining two distinct produ
with code bases that may well be wildly divergent. This is likely to be an issue no matter w
method you use.

So you can either maintain a separate branch for version 1.0 just for bug fixes (i.e., withou
feature branches of its own -- those are reserved for 2.0) or create a separate trunk for w
essentially a separate project. It runs slightly counter to the original intent, but it's more o
necessary in any case. The end result may require you to fix the same bugs twice, but tha
basically what you sign up for when offering fixes for older versions.

— **William Wedin (http://www.wcwedin.com)** - Tuesday, June 17, 2008 7:56:11 AM (/bsimser/day-to-day-with-subversion#comment-1529)

Great article! Our team is fairly new to SVN and this goes a long way in explaining the "ho
of branching and merging. The inclusion of screenshots was essential in that understandin
Thank you Bill!

— **Shirley** - Friday, May 28, 2010 9:49:42 AM (/bsimser/day-to-day-with-subversion#comment-1530)

This is great, it establishes perfectly why this is so confusing, why it will stay confused, an
how deeply broken TortoiseSVN's UI is. "We'll merge changes from the trunk into the 1.1
branch"... by specify the BRANCH in the FROM field, and the TRUNK in the TO field.

It's astounding, but there it is. No way to write a clear explanation of something that is
obfuscated by design.

I hope they don't use TortoiseSVN in the space program.

— **fortboise (http://fortboise.org/blog/)** - Monday, August 30, 2010 1:16:44 PM (/bsimser/day-to-day-with-subversion#comment-1531)

Been having some trouble with this stuff, sought help, and read through your account very carefully, tried to put it into practice. I'm good with the branching, and the day-to-day mer works as expected, with the important insight that it's FROM the branch revision that was common with the trunk.

But I've been unable to get TortoiseSVN to bring the branch changes back to the trunk. M project has one change in one file in the trunk, and one change to another file in the bran The trunk's change has been day-to-day merged to the branch, so the reintegration shoul comprise ONE change to ONE file in the trunk.

But the TEST merge (on the clean working copy of the trunk, FROM its HEAD, TO the HEA the branch) shows TWO changes coming, one to each file:
Updated Default.aspx.cs
Updated Default.aspx

But that's not the bad part. The bad part is that having it perform the merge after the test shows that, it does SOMETHING DIFFERENT:

Merging
Updated Default.aspx.cs
Updated Default.aspx
Merging r641 through r644
Updated Default.aspx.cs
Reverse merging 644 through 641
Updated Default.aspx.cs

It does bring in the branch's change to Default.aspx, but there are FOUR entries on a file i should not be touching, and looking at the resulting file, I see that it has reverted to a ver that is earlier than what's in EITHER the trunk or the branch in the repository.

Some things are profoundly broken here, and "it's not just me."

— **Tom von Alten (http://fortboise.org/blog/)** - Tuesday, August 31, 2010 9:19:47 AM (/bsimser/day-to-day-with-subversion#comment-1532)

Hey thanks for posting. From all my internet researches on the subject of TortoiseSVN / Subversion, this is the only one that makes any sense for me. This has saved me a LOT of damage. Great stuff.

Cheers

Andy

— **Andy (http://andyuk2010.blogspot.com)** - Wednesday, September 29, 2010 8:13:37 AM (/bsimser/day-to-day-with-subversion#comment-1533)

great article, wish images were more clear.

— **Tarun (http://tarunksblog.blogspot.com/)** - Saturday, November 13, 2010 5:23:15 AM (/bsimser/day-to-day-with-subversion#comment-1534)

Thanks for an excellent article! We're doing some initial R & D on switching from VSS over SVN, and this article really helped me understand the process of using SVN & Tortoise on daily basis. Thanks again!

— **Adam Belebczuk (http://www.twitter.com/nfdotcom)** - Tuesday, January 4, 2011 10:40:57 AM (/bsimser/day-to-day subversion#comment-1535)

Nice post!
This article clarified the most part of my questions.

— **Fred Wuerges** - Thursday, February 24, 2011 7:42:03 AM (/bsimser/day-to-day-with-subversion#comment-1536)

It does bring in the branch's change to Default.aspx, but there are FOUR entries on a file i should not be touching, and looking at the resulting file, I see that it has reverted to a ver that is earlier than what's in EITHER the trunk or the branch in the repository.

— **Electric Scooters (http://www.machscooters.com/electric-scooters.html)** - Saturday, June 4, 2011 12:23:59 PM (/ /day-to-day-with-subversion#comment-1537)

Hi,

I'm new to Subversion. I found your post very helpful, but there's one thing that actually confused me when I was reading along. When you say "We'll merge changes from the tru into the 1.1 branch:" the image right below it shows something different from what you're saying. It shows you're merging from the branch 1.1 to the trunk, wasn't it supposed to b done the other way around? Maybe I'm wrong, but that made me think whether it's a typ you uploaded the wrong image.

Anyways thank you for posting such info.

— **Mark** - Wednesday, December 14, 2011 2:29:56 PM (/bsimser/day-to-day-with-subversion#comment-1538)

Ohh,

I guess my question was already answered in the explanation you gave in the section «A f notes about this merge». I'm sorry, but maybe I had to continue reading to the end befor asking something, which was explained further down.

Great post!

— **Mark** - Thursday, December 15, 2011 12:07:28 PM (/bsimser/day-to-day-with-subversion#comment-1539)

Thank you so much, now I'm understand what is "stable trunk".
If I may suggest, please update on how to compile into download ready file (.zip) please.
And I can't find the donation button, where it is?

— **Amir** - Monday, January 16, 2012 8:01:57 AM (/bsimser/day-to-day-with-subversion#comment-1540)

Unexceptional branching is just a sign of poor planning of forthcoming features and lack o quality.

— **Arvind** - Tuesday, April 17, 2012 10:57:34 AM (/bsimser/day-to-day-with-subversion#comment-1541)

Awesome article. I don't care how old it is - still relevant. Subversion is easy setup (VisualSVN + TourtiseSVN) and is a Godsend when I discover clients still using VSS "because they don't have the money to move to TFS". This article saved me tc time and hair.

— **Mike (http://www.applied-health.com)** - Wednesday, June 20, 2012 11:28:45 PM (/bsimser/day-to-day-with-subversion#comment-1542)

Thanks so much for such a crisp explanation. Clarifies all remaining questions abou subversion.

— **IyerB** - Wednesday, July 25, 2012 3:59:02 PM (/bsimser/day-to-day-with-subversion#comment-1543)

Hmm.

Obfuscated a fairly simple topic, it seems to me, with gibberish such as:

"Why again do we branch? Imagine if we didn't branch. So right after you apply the "1.0" tag start modifying trunk. Sure, we can go back to "1.0" but are we going to get any changes merged together when we're on a single line? We're also violating the "One King" rule. Who's the King now? Our new branch becomes a servant. The King still takes priority (for example to fix bugs) but work will continue on in servant branch."

— **Ed Austin** - Monday, August 6, 2012 11:11:01 PM (/bsimser/day-to-day-with-subversion#comment-1544)

Wow! This is a very good article in 2008. But still it is applicable very much. Thank you ver much for sharing.

— **Ponam** - Wednesday, September 5, 2012 3:14:29 AM (/bsimser/day-to-day-with-subversion#comment-1545)

This is just what I was looking for!

Been using subversion for about an year and never created a branch, mainly because we l
been stabilizing and finishing our first "true" stable release.

Since we are almost releasing it, I was thinking to apply something like this strategy, but v
having trouble visualize it, and this article really helped.

thanks!

— **jfm** - Saturday, November 10, 2012 9:13:28 AM (/bsimser/day-to-day-with-subversion#comment-1546)

Comments have been disabled for this content.