Basic SVN Guide

Paul Conner

Preface

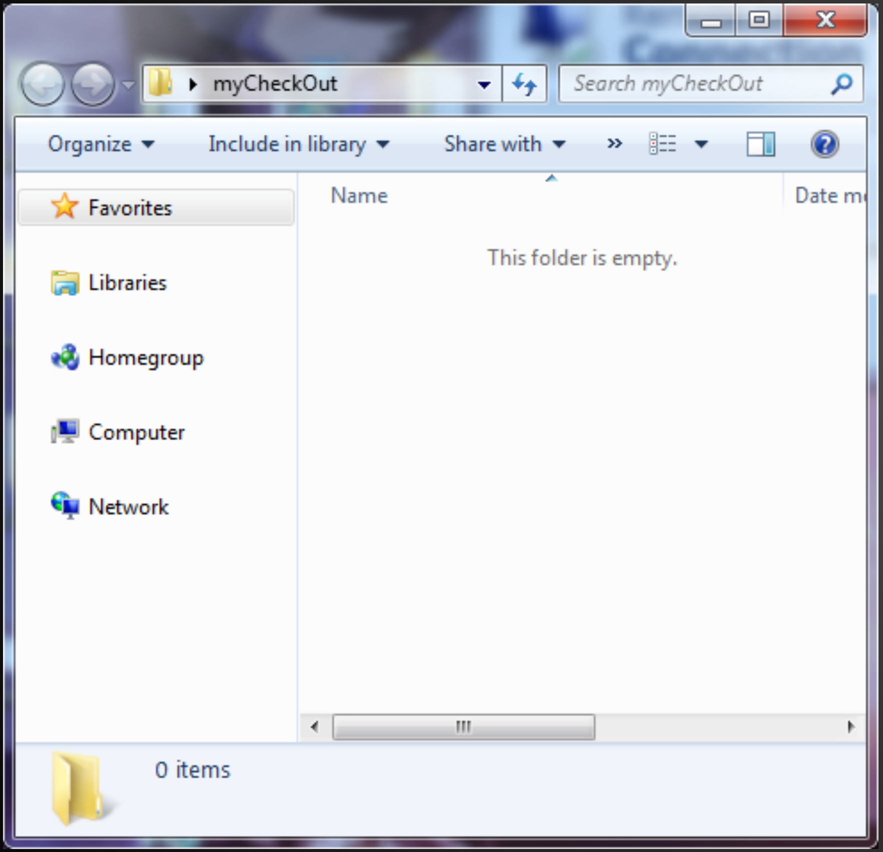
This is made to help people use SVN correctly. Not knowing the in's and out's of SVN is a common problem. This can lead to difficulties in project development later down the line.
Some of this guide may seem extremely basic, but I want it to be thorough.

Although this guide is made using a game's repository, the content discussed here applies to any software development.

This guide assumes that you have already set up a repository. However, there is a section that talks about repository organization. Aside from that, be sure to set up things, like your Ignore List, before anyone starts development using the repository - such things will not be discussed here.

Contents

Checking Out

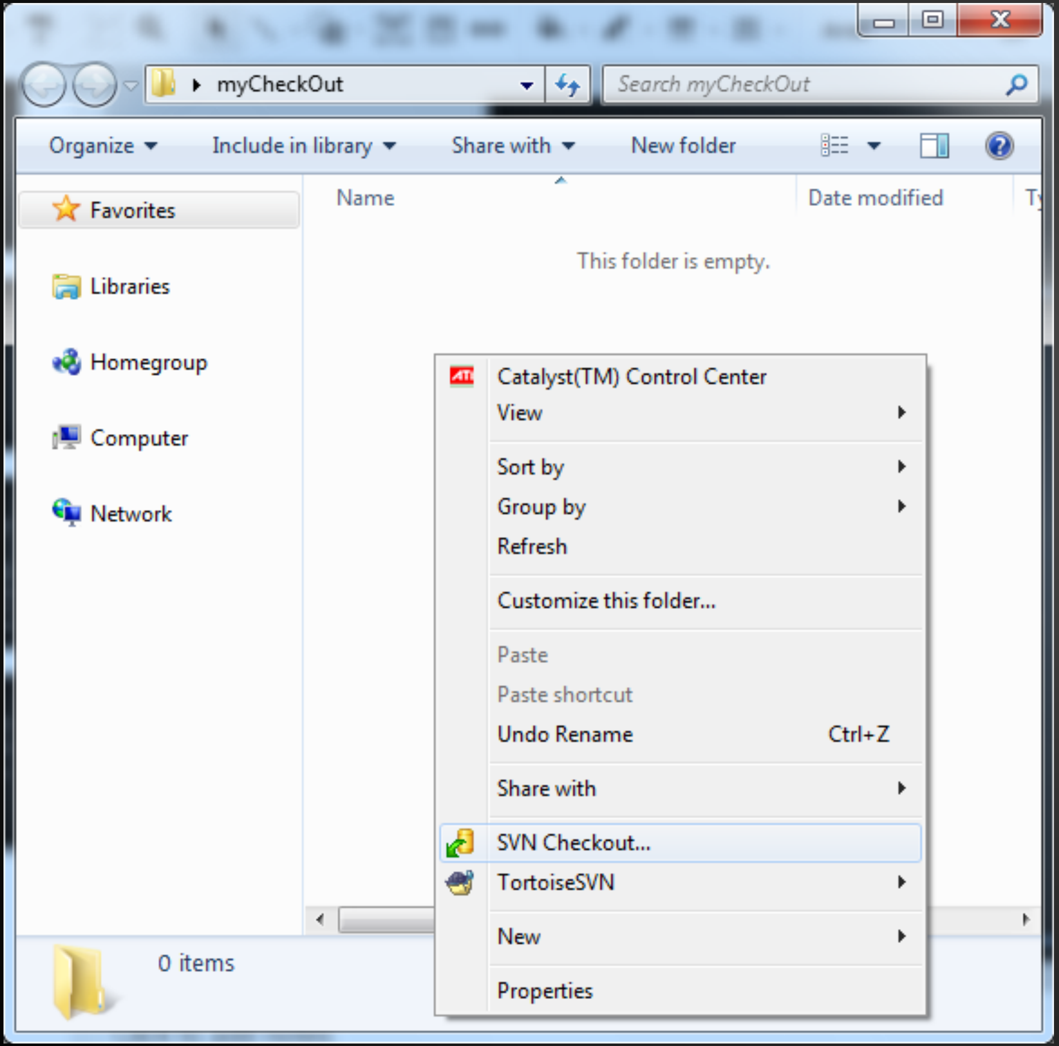


Checking out a project is where it all shall begin.
First, create a new folder wherever you please.

A word of caution: Do not use the desktop. Checking out often uses hidden folder(s) - when placed on the desktop, you may delete some folders, but miss the hidden folder(s) - causing any future checkouts to not work properly, with no clear reason why.

Checking Out

(cont)

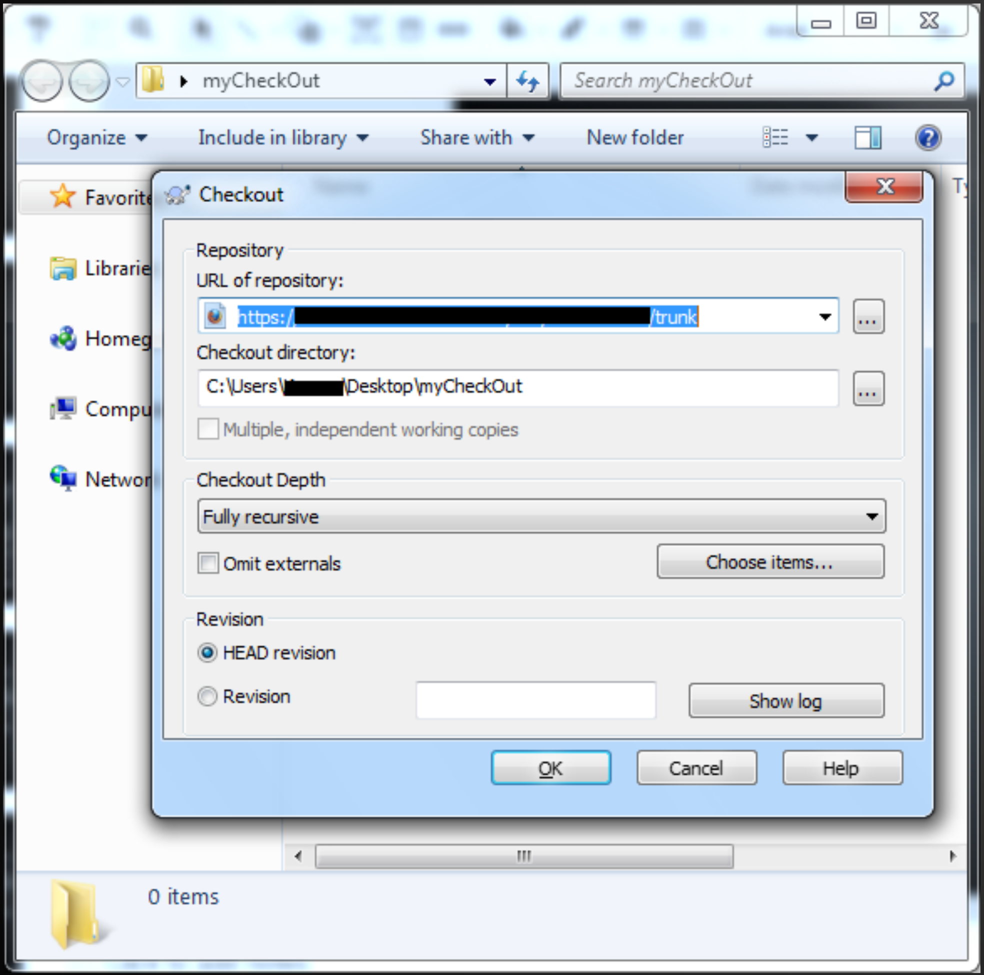


Throughout this guide, I will be using TortoiseSVN. This is a great tool, and I'd strongly recommend using it, if you are not already.

Right-click in the folder, and select SVN Checkout...

Checking Out

(cont)



Repository

URL of repository: https:/[redacted]/trunk

Checkout directory: C:\Users\[redacted]\Desktop\myCheckOut

☐ Multiple, independent working copies

Checkout Depth: Fully recursive

☐ Omit externals

Revision

● HEAD revision

○ Revision

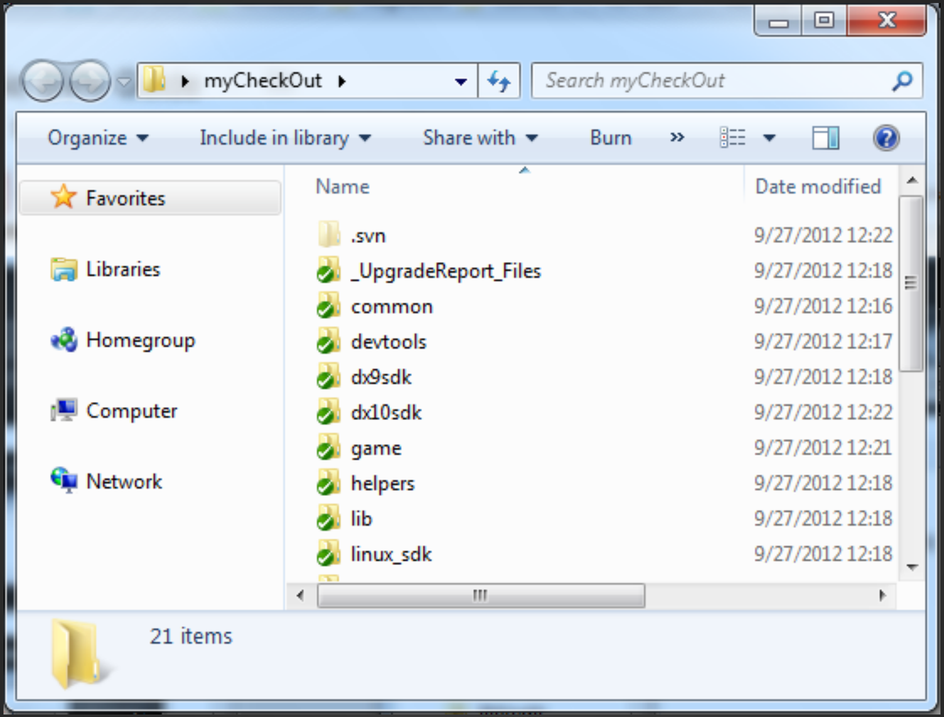
Put the URL of the repository into the designated location. Be sure to select the trunk (for now!) as the location to checkout.

After this Check Out section, I will cover my preferred SVN folder hierarchy.

If you do not have a repository setup, that is outside the scope of this guide. But, I can recommend VisualSVN Server. It is a great tool.

Checking Out

(cont)

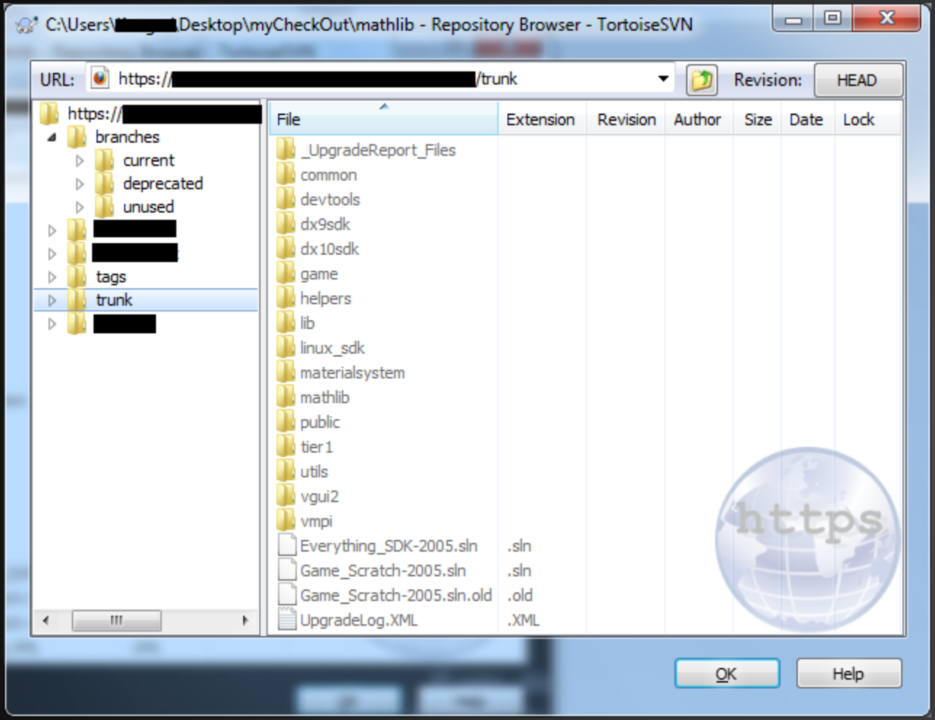


Name	Date modified
.svn	9/27/2012 12:22
_UpgradeReport_Files	9/27/2012 12:18
common	9/27/2012 12:16
devtools	9/27/2012 12:17
dx9sdk	9/27/2012 12:18
dx10sdk	9/27/2012 12:22
game	9/27/2012 12:21
helpers	9/27/2012 12:18
lib	9/27/2012 12:18
linux_sdk	9/27/2012 12:18

21 items

Now, our checkout is complete. Congrats.
(Our code files/folders may - and will - differ).
Note: Do not checkout the root folder directly.
It will prevent us from using SVN Switch (discussed later).

Repository Organization



C:\Users\[redacted]Desktop\myCheckOut\mathlib - Repository Browser - TortoiseSVN

URL: https://[redacted]/trunk   Revision: HEAD

- https://[redacted]
  - branches
    - current
    - deprecated
    - unused
  - [redacted]
  - [redacted]
  - tags
  - trunk
  - [redacted]

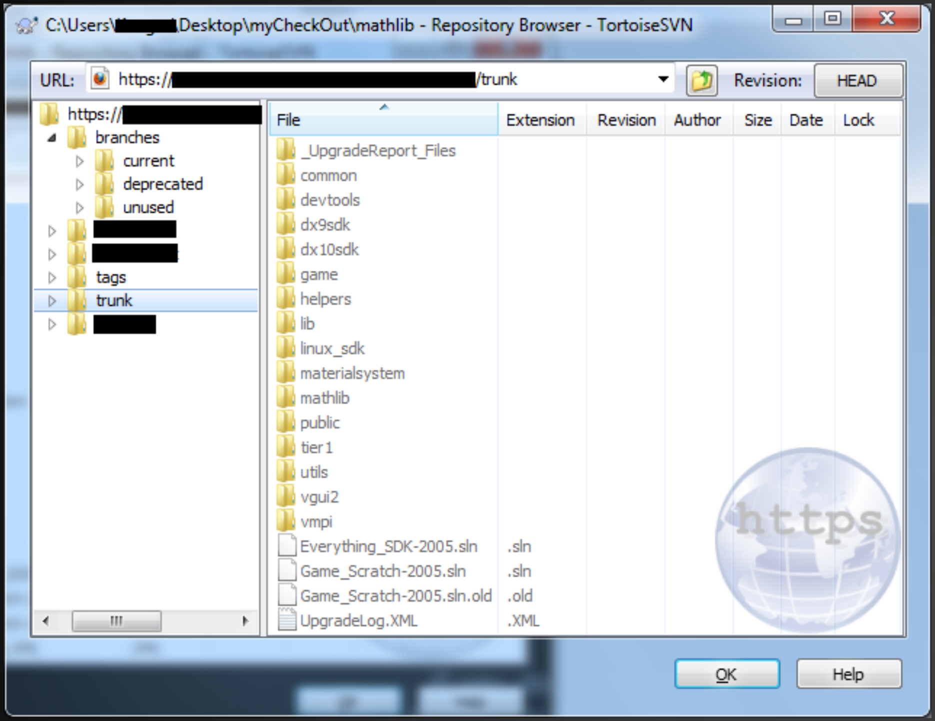
File	Extension	Revision	Author	Size	Date	Lock
_UpgradeReport_Files						
common						
devtools						
dx9sdk						
dx10sdk						
game						
helpers						
lib						
linux_sdk						
materialsystem						
mathlib						
public						
tier1						
utils						
vgui2						
vmpi						
Everything_SDK-2005.sln	.sln					
Game_Scratch-2005.sln	.sln					
Game_Scratch-2005.sln.old	.old					
UpgradeLog.XML	.XML					

The main three folders are trunk (the centerpoint of all development), tags (saved snapshots of the trunk), and branches (copies of the trunk that are currently being worked on).
The trunk should "never" be worked out from directly.

The trunk should be a runnable version of the software at all times. When implementing something new, use branches to preserve the runnable nature of the trunk.
A clean branch system, that I employ, is using current, deprecated, and unused folders to order branches.

Repository Organization

(cont)



C:\Users\[redacted]Desktop\myCheckOut\mathlib - Repository Browser - TortoiseSVN

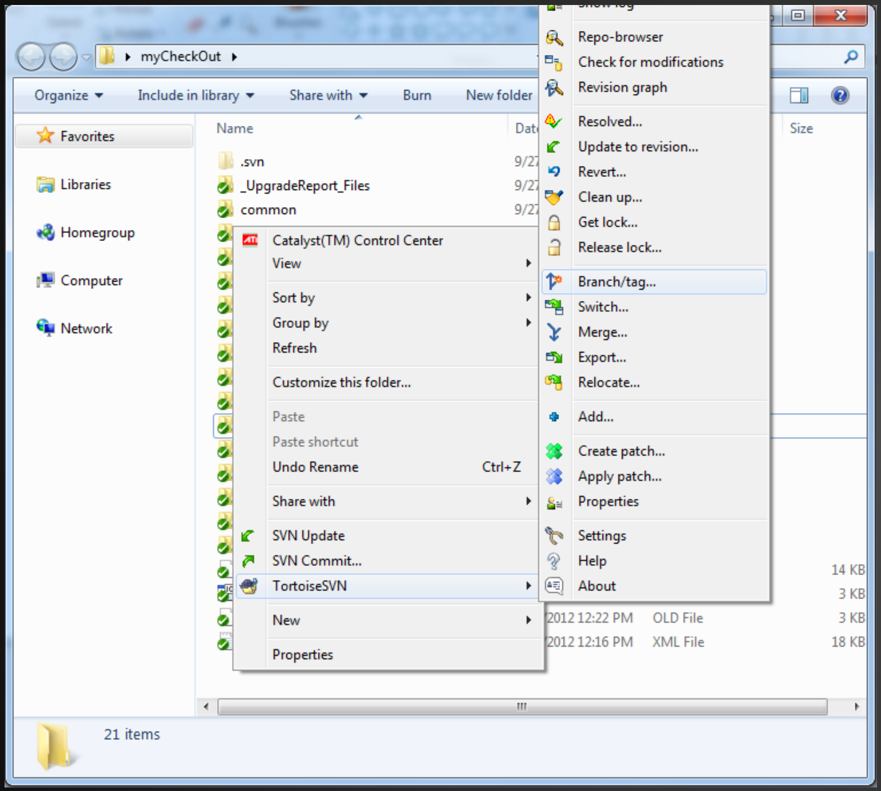
URL: https://[redacted]/trunk   Revision: HEAD

- https://[redacted]
  - branches
    - current
    - deprecated
    - unused
  - [redacted]
  - [redacted]
  - tags
  - trunk
  - [redacted]

File	Extension	Revision	Author	Size	Date	Lock
_UpgradeReport_Files						
common						
devtools						
dx9sdk						
dx10sdk						
game						
helpers						
lib						
linux_sdk						
materialsystem						
mathlib						
public						
tier1						
utils						
vgui2						
vmpi						
Everything_SDK-2005.sln	.sln					
Game_Scratch-2005.sln	.sln					
Game_Scratch-2005.sln.old	.old					
UpgradeLog.XML	.XML					

The current folder is for branches that are currently being worked on. The 'deprecated' folder is for branches that have been finished and re-merged back into the trunk (see Merging). And 'unused' is for branches that have been decided they no longer apply - they are *done* being worked on, but will **not** be re-merged into the trunk. If your design is strong, you may (and hopefully will) find your 'unused' folder to be empty. Although, it can allow for sandbox branches (used for learning the code base/platform) to find a home out of everyones' way once actual development has begun.

Branching

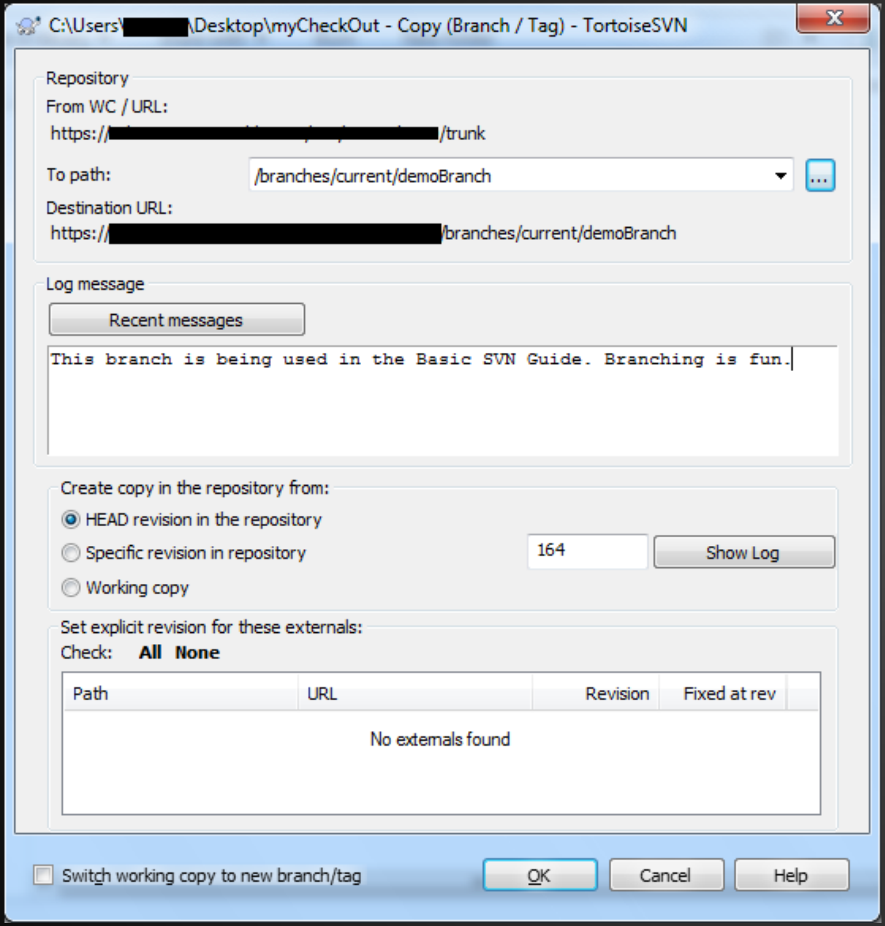


Select TortoiseSVN > Branch/tag... This is how you start the branching process (the more astute ones reading this will also gather that this is how we start tagging as well. We'll get to that later.)

Be sure to Right-Click in the white-space, at the bottom of the folder - as to not accidentally Right-Click a file or folder that we are looking at. You want to branch/tag the folder you checked out, not any inner folders.

Branching

(cont)


C:\Users\[redacted]\Desktop\myCheckOut - Copy (Branch / Tag) - TortoiseSVN

Repository

From WC / URL:
https://[redacted]/trunk

To path: /branches/current/demoBranch

Destination URL:
https://[redacted]/branches/current/demoBranch

Log message

```
This branch is being used in the Basic SVN Guide. Branching is fun.
```

Create copy in the repository from:
- [✓ HEAD revision in the repository]
- Specific revision in repository: 164
- Working copy

Set explicit revision for these externals:
Check: **All** **None**

Path	URL	Revision	Fixed at rev

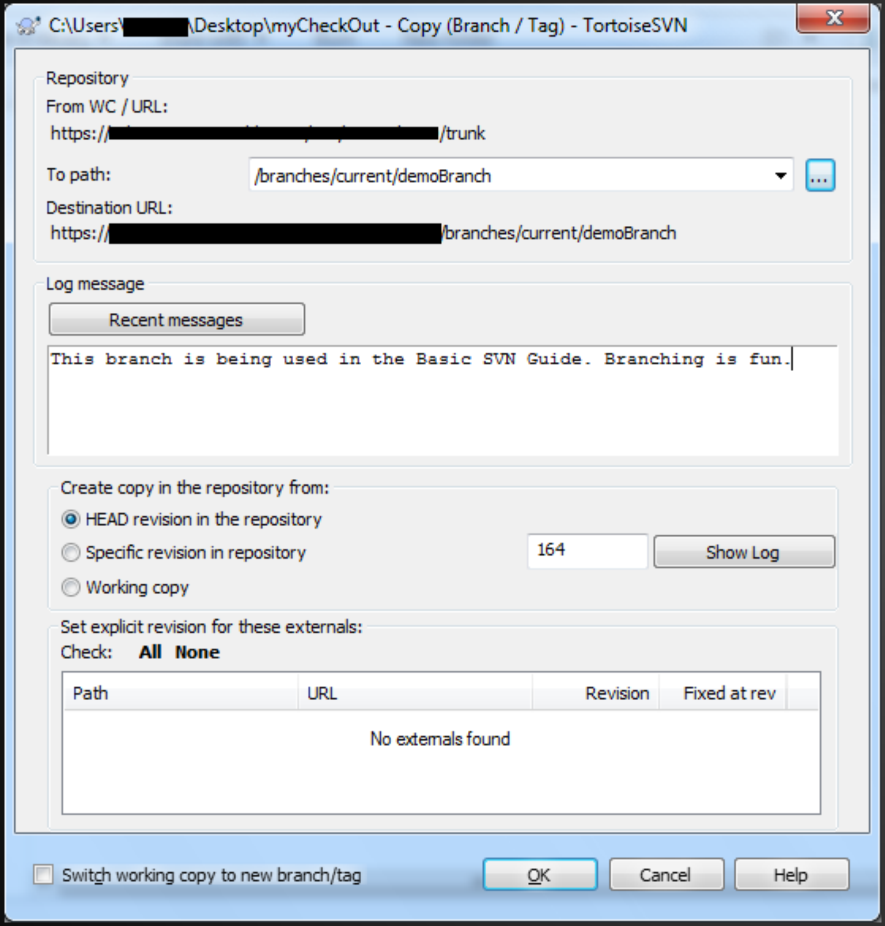
No externals found

Choose your desired branch location. You can use the [...] to browse the repository and find your 'current' folder. You will then have to type in the rest of the directory manually. This is because the directory that you want to branch into must not exist (to protect against any conflicts of what may possibly exist in an existing folder.

Note: The folder to branch into **must not** exist, but all folders above it in the directory **must** exist.

Branching

(cont)



C:\Users\[redacted]\Desktop\myCheckOut - Copy (Branch / Tag) - TortoiseSVN

Repository

From WC / URL:

https://[redacted]/trunk

To path: /branches/current/demoBranch

Destination URL:

https://[redacted]/branches/current/demoBranch

Log message

```
This branch is being used in the Basic SVN Guide. Branching is fun.
```

Create copy in the repository from:

[✓ HEAD revision in the repository]

Specific revision in repository: 164

Working copy

Set explicit revision for these externals:

Check: **All** **None**

Path	URL	Revision	Fixed at rev

No externals found

Switch working copy to new branch/tag

Be sure to add a message. This message should describe why you are making this branch. The reason is almost always to implement a specific new feature. Try to be descriptive as possible. This will allow future repository traversal to be much easier and, hopefully, headache free.

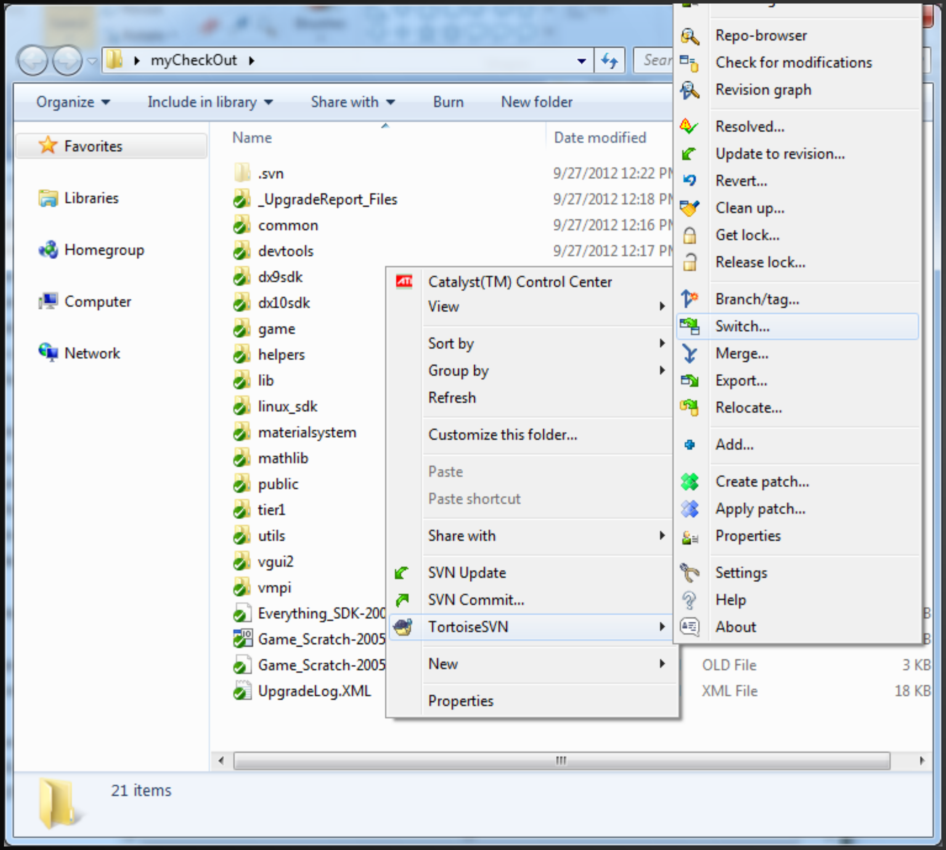
A good rule of thumb is to always write a message when changing anything on the repository. This is often considered mandatory - and rightfully so.

Branching
(cont)

If you were super astute, you may have noticed a check box at the bottom of the Branch/tag window, it would auto-switch to the new branch. I left that unchecked so we can discuss switching manually, and why it's important.

Now, you will notice the branch window talks about being in the old working copy, and you will now need to use switch. This brings us to the next segment.

Switching



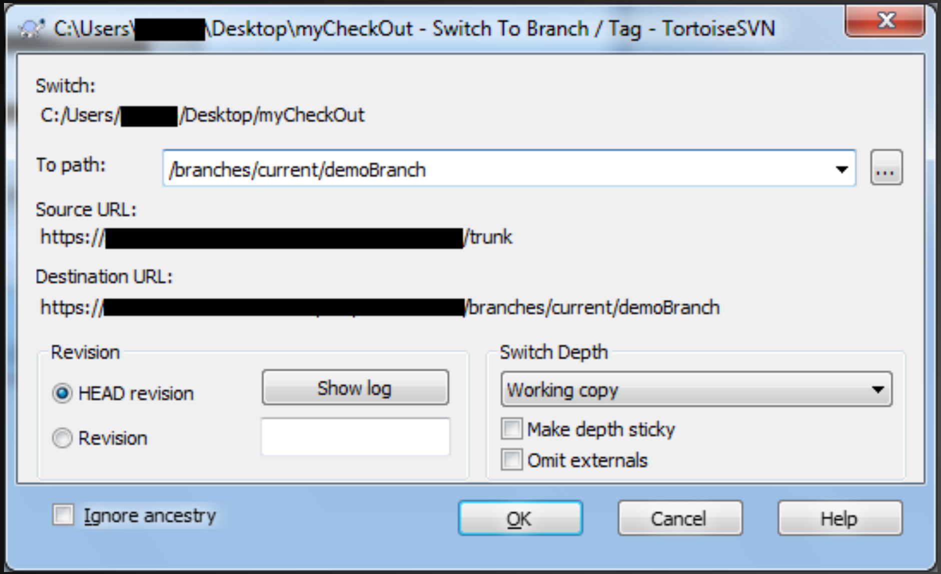
Select TortoiseSVN > Switch...

What this allows us to quickly change between repository sections (branches, the trunk, tags, etc.) with ease and little downtime.

The reason we use SVN Switch is because it will only change our local files (if changes exist) to match the location where we are switching to. For all files that are the same, nothing happens. This is ridiculously faster than re-checking everything out - especially on large projects.

Switching

(cont)



C:\Users\[redacted]\Desktop\myCheckOut - Switch To Branch / Tag - TortoiseSVN

Switch:
C:/Users/[redacted]/Desktop/myCheckOut

To path: /branches/current/demoBranch

Source URL:
https://[redacted]/trunk

Destination URL:
https://[redacted]/branches/current/demoBranch

Revision: [✓ HEAD revision] Revision

Switch Depth: Working copy

☐ Make depth sticky
☐ Omit externals
☐ Ignore ancestry

Choose the directory of your new branch (that you had created in the Branching section of this guide). You can use the [...] button to browse the repository and find the correct branch.

Note 1: As stated in the Branching section, you can tell TortoiseSVN to automatically switch after creating a branch.
Note 2: As mentioned in the Checking Out section, this is why you want to checkout specific sections of the repository. Checking out the whole root directory prevents you from using SVN Switch - it costs you the ability of efficiently navigating the entire repository.

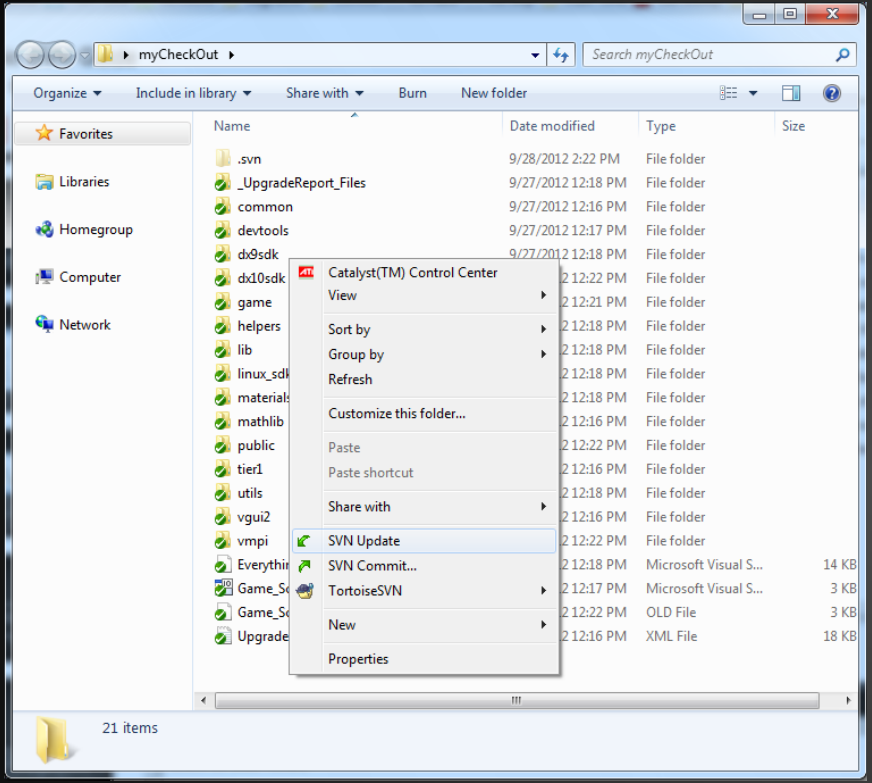
Working

Now has come your moment of glory.

This is where you work, implement your new feature and do all the lovely testing required to ensure that your feature works properly.

The next four sections will talk about using SVN mid-feature development.

Updating

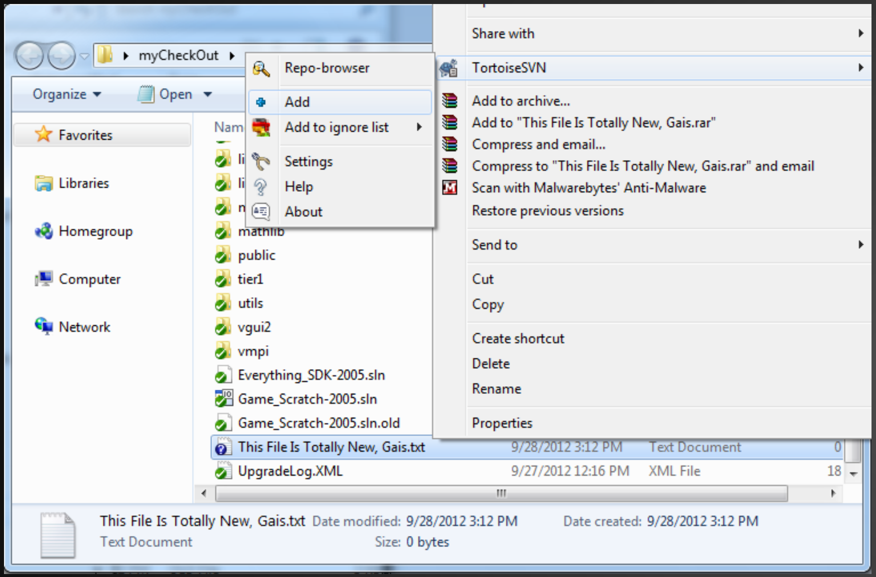


Name	Date modified	Type	Size
.svn	9/28/2012 2:22 PM	File folder	
_UpgradeReport_Files	9/27/2012 12:18 PM	File folder	
common	9/27/2012 12:16 PM	File folder	
devtools	9/27/2012 12:17 PM	File folder	
dx9sdk	9/27/2012 12:18 PM	File folder	
dx10sdk	2 12:22 PM	File folder	
game	2 12:21 PM	File folder	
helpers	2 12:18 PM	File folder	
lib	2 12:18 PM	File folder	
linux_sd	2 12:18 PM	File folder	
material	2 12:18 PM	File folder	
mathlib	2 12:16 PM	File folder	
public	2 12:22 PM	File folder	
tier1	2 12:16 PM	File folder	
utils	2 12:18 PM	File folder	
vgui2	2 12:16 PM	File folder	
vmpi	2 12:22 PM	File folder	
Everythi	2 12:18 PM	Microsoft Visual S...	14 KB
Game_S	2 12:17 PM	Microsoft Visual S...	3 KB
Game_S	2 12:22 PM	OLD File	3 KB
Upgrade	2 12:16 PM	XML File	18 KB

To update, Select SVN Update. This will pull any changes from the repository to your local copy. This is very important if you are working with someone else on the same branch. If you aren't, I'd still say Updating is good practice.

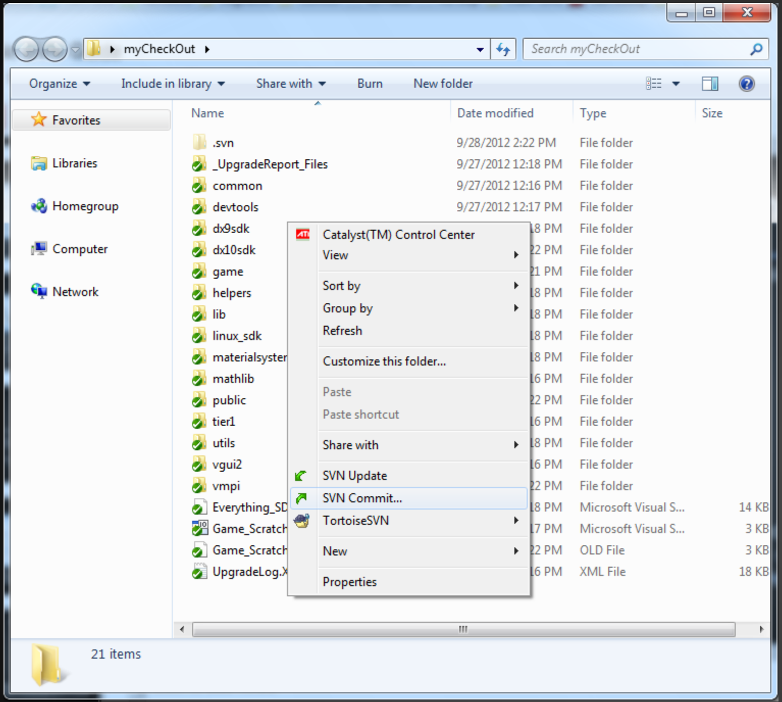
SVN is quite good at comparing files and combining them. Sometimes Conflicts can arise between versions of the file(s). Handling such issues will be discussed later on in the guide...

Adding To Version Control


myCheckOut
Organize
Open
Favorites
Libraries
Homegroup
Computer
Network
Repo-browser
Add
Add to ignore list
Settings
Help
About
Share with
TortoiseSVN
Add to archive...
Add to "This File Is Totally New, Gais.rar"
Compress and email...
Compress to "This File Is Totally New, Gais.rar" and email
Scan with Malwarebytes' Anti-Malware
Restore previous versions
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
mathlib
public
tier1
utils
vgui2
vmpi
Everything_SDK-2005.sln
Game_Scratch-2005.sln
Game_Scratch-2005.sln.old
This File Is Totally New, Gais.txt 9/28/2012 3:12 PM Text Document 0
UpgradeLog.XML 9/27/2012 12:16 PM XML File 18
This File Is Totally New, Gais.txt Date modified: 9/28/2012 3:12 PM Date created: 9/28/2012 3:12 PM
Text Document Size: 0 bytes

When we create a new file, SVN does not know of it magically (or at least, SVN does not know what you want to do with it magically.)
If this is a new code file, we will most likely want to add it to be kept track of in the repository - like the rest of the code.
Files that are not added to Version Control (or not added to the Ignore List - not discussed in this guide) are shown with a blue, question mark icon.
Right-Click on the file(s) we want to add and select TortoiseSVN > Add. It will now have a '+' sign, showing it has been added but not yet Committed (discussed later).

Deleting



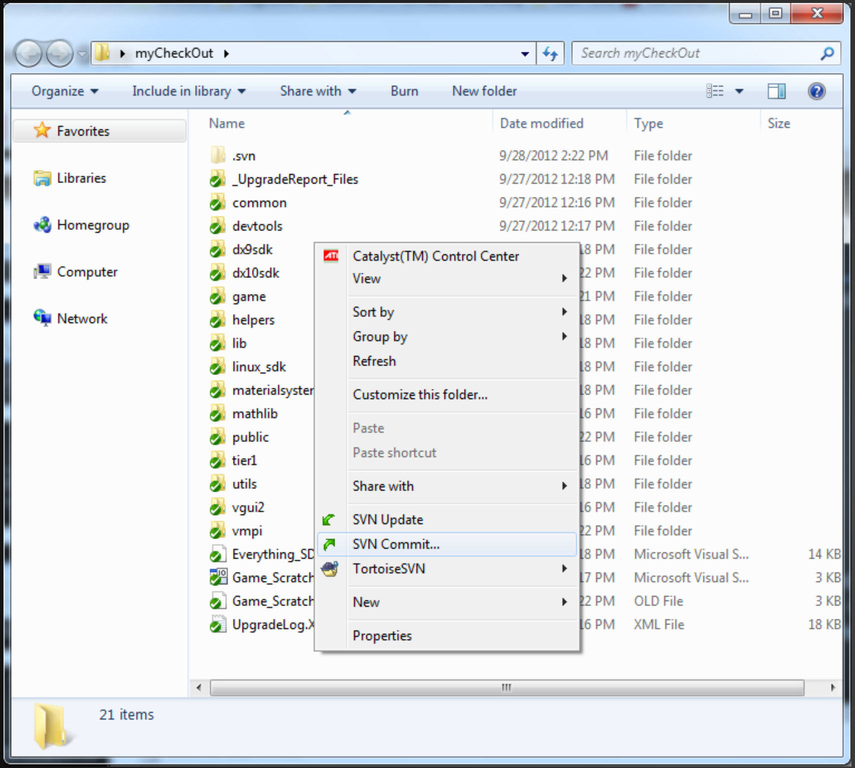
Name	Date modified	Type	Size
.svn	9/28/2012 2:22 PM	File folder	
_UpgradeReport_Files	9/27/2012 12:18 PM	File folder	
common	9/27/2012 12:16 PM	File folder	
devtools	9/27/2012 12:17 PM	File folder	
dx9sdk	18 PM	File folder	
dx10sdk	22 PM	File folder	
game	21 PM	File folder	
helpers	18 PM	File folder	
lib	18 PM	File folder	
linux_sdk	18 PM	File folder	
materialsyster	18 PM	File folder	
mathlib	16 PM	File folder	
public	22 PM	File folder	
tier1	16 PM	File folder	
utils	18 PM	File folder	
vgui2	16 PM	File folder	
vmpi	22 PM	File folder	
Everything_SD	18 PM	Microsoft Visual S...	14 KB
Game_Scratch	17 PM	Microsoft Visual S...	3 KB
Game_Scratch	22 PM	OLD File	3 KB
UpgradeLog.X	16 PM	XML File	18 KB

If you are not used to SVN, deleting a file may prove to be frustrating and confusing.

If we simply delete a file, it will come back the next time we SVN Update.
The answer is to simply use the SVN Delete command.

Select TortoiseSVN > Delete. You will now see a red X next to the file. When you Commit (discussed next) next, it will tell the repository to remove the file, then it will automatically delete the item locally. Sometimes, it will delete the local item immediately. If you hit SVN Delete by accident, do not worry. Select TortoiseSVN > Revert, check the file in question, and hit ok.

Committing



Name	Date modified	Type	Size
.svn	9/28/2012 2:22 PM	File folder	
_UpgradeReport_Files	9/27/2012 12:18 PM	File folder	
common	9/27/2012 12:16 PM	File folder	
devtools	9/27/2012 12:17 PM	File folder	
dx9sdk	18 PM	File folder	
dx10sdk	22 PM	File folder	
game	21 PM	File folder	
helpers	18 PM	File folder	
lib	18 PM	File folder	
linux_sdk	18 PM	File folder	
materialsyster	18 PM	File folder	
mathlib	16 PM	File folder	
public	22 PM	File folder	
tier1	16 PM	File folder	
utils	18 PM	File folder	
vgui2	16 PM	File folder	
vmpi	22 PM	File folder	
Everything_SD	18 PM	Microsoft Visual S...	14 KB
Game_Scratch	17 PM	Microsoft Visual S...	3 KB
Game_Scratch	22 PM	OLD File	3 KB
UpgradeLog.X	16 PM	XML File	18 KB

Select SVN Commit...
This is how we put our changes, that exist on our local checkout, onto the repository.

We are 'committing' to our changes. We should have, of course, tested everything and make sure that our version is at least somewhat stable.

Important Note: Best practice is to **always** Update right before Committing. If we don't take changes from others into account when you commit, it may cause problems (ones that *may* not be able to be remedied.) Even if you're working alone. **Still do it. Get in the habit. Always do it!**

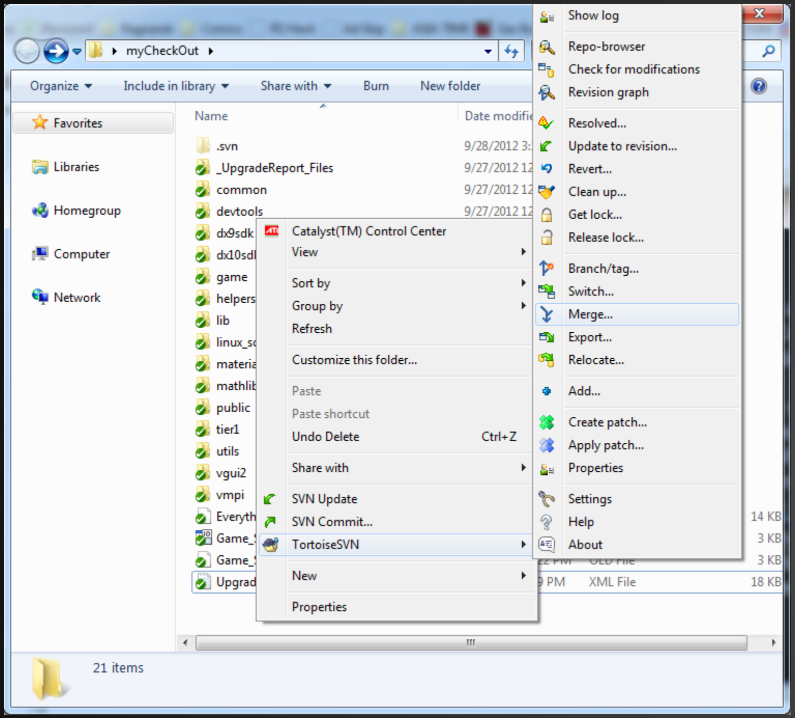
Reintegration

At this point, our new feature should be complete.
We have tested all our code.
Our feature should be stable - and, hopefully, bugless.

All changes on all local copies should be committed.

We are ready to reintegrate our branch back into the trunk.

Merge Range of Revisions
(MRoR)



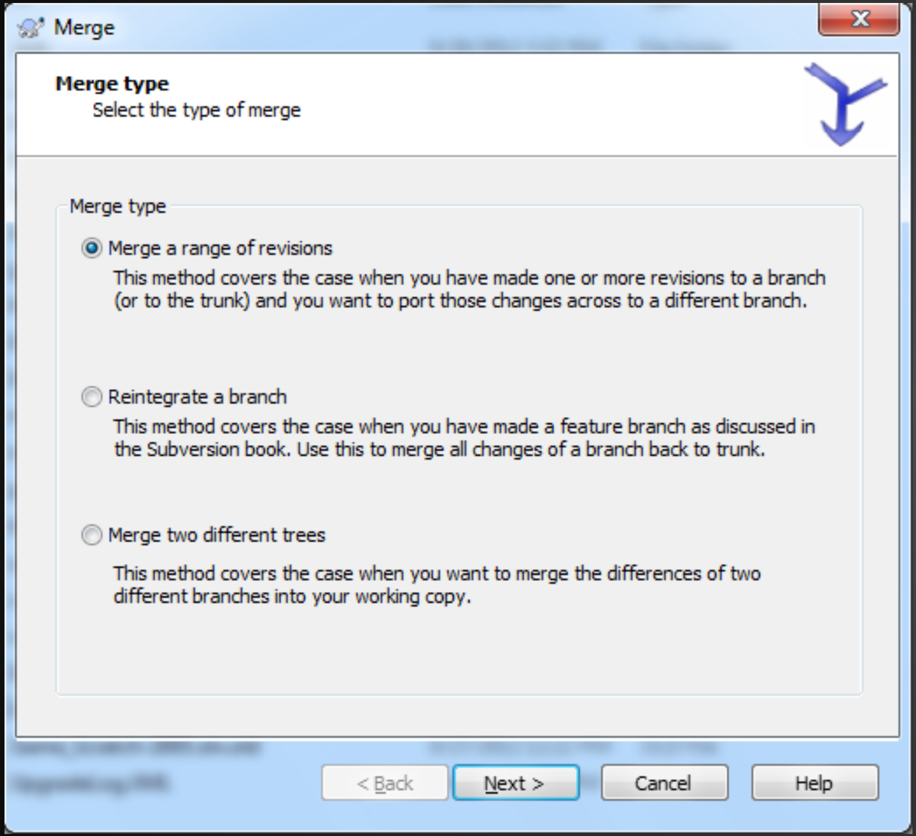
Just as when we are working, before we can Commit, we must Update.

Now, before we can Merge (bring our new feature to the trunk) we must MRoR (bring any changes that might exist in the trunk to our branch).

Remember, as we were working on this feature, another feature branch(es) may have been started and completed. We need to bring that into our branch and make sure all features mesh well and don't conflict.

Select TortoiseSVN > Merge...

Merge Range of Revisions
(MRoR)
(cont)



Merge type
Select the type of merge

Merge type

[✓ Merge a range of revisions]
This method covers the case when you have made one or more revisions to a branch (or to the trunk) and you want to port those changes across to a different branch.

Reintegrate a branch
This method covers the case when you have made a feature branch as discussed in the Subversion book. Use this to merge all changes of a branch back to trunk.

Merge two different trees
This method covers the case when you want to merge the differences of two different branches into your working copy.

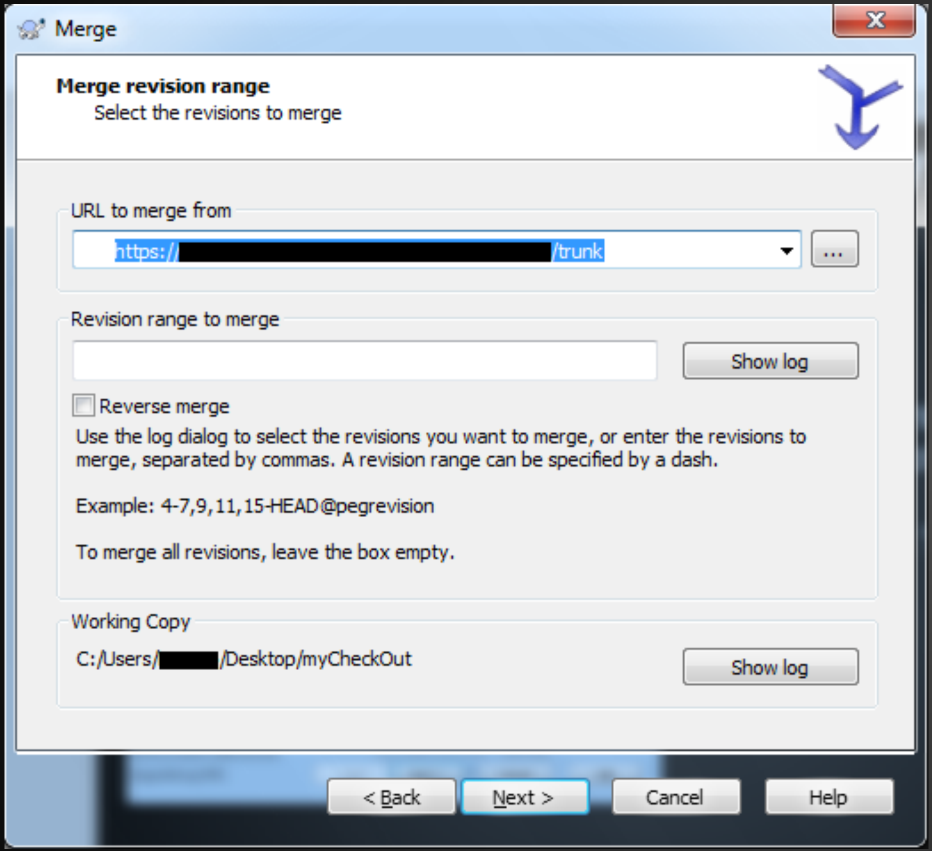
Choose the first option, "Merge a range of revisions"

Hit 'Next'

Merge Range of Revisions

(MRoR)

(cont)


Merge

Merge revision range

Select the revisions to merge

URL to merge from

https://[redacted]/trunk

Revision range to merge

Reverse merge

Use the log dialog to select the revisions you want to merge, or enter the revisions to merge, separated by commas. A revision range can be specified by a dash.

Example: 4-7,9,11,15-HEAD@pegrevision

To merge all revisions, leave the box empty.

Working Copy

C:/Users/[redacted]/Desktop/myCheckOut

Choose the trunk for the 'URL to merge from.' Leave the rest alone to pull all changes to your branch. Hit 'Next'.

On the next screen, you can leave all the options alone.

Press the 'Test merge' button, located near the bottom right.

Merging is a scary process.

It is, without a doubt, where the most things can go wrong.

That is why we test it first. This does not change any files, but tells you what would happen if you hit 'Merge'. If everything's ok, go ahead and Merge. (If not so smooth discussed later.)

Merge Range of Revisions
(MRoR)
(cont)

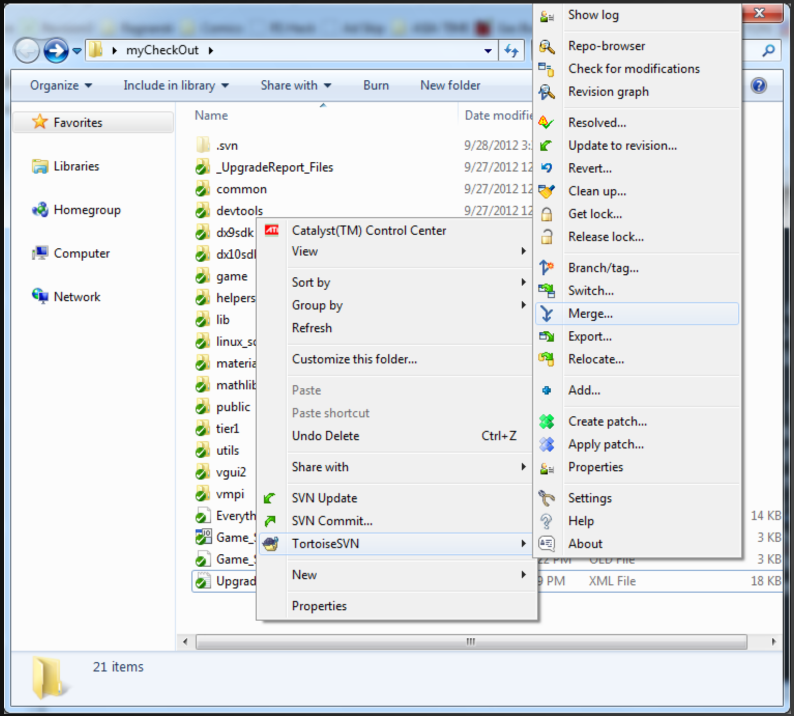
Now, we will probably have changes.

Go ahead and test our code once again, make sure all things work well together. Make sure it is stable.

Once we are sure it is ok, Update/Commit to the branch. This branch should now consist of the current trunk + the new feature.

Now use SVN Switch to go back to the trunk.
Note: It will likely say "Deleted: ..." for many files. Don't worry. They are being deleted locally because they do not exist in the trunk (yet!). They still exist on the repository, in your branch. This is why we made sure to commit just now.

Merging



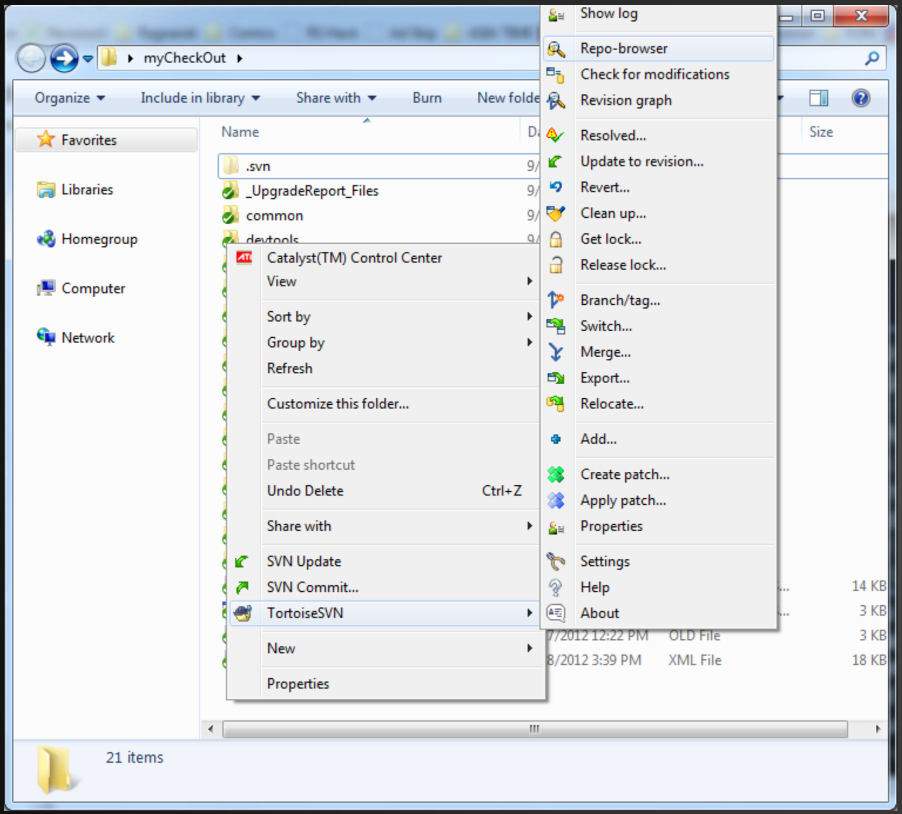
Like at the beginning of MRoRing, we again select TortoiseSVN > Merge...

This time, select 'Reintegrate a branch.'

Hit 'Next'. On the next screen, select your branch, you can use [...] to find it easily.

Hit 'Next'. On the last screen, make use of 'Test merge' to make sure that things will merge smoothly. If it says no conflicts, then merge for real and cross your fingers. Again, test that it works. **You do not want to break the trunk, ever.** If its perfect and good and wonderful and the best thing ever, Update/Commit. Note: Our branch is no longer work-on-able. You can only merge once.

Cleaning Up

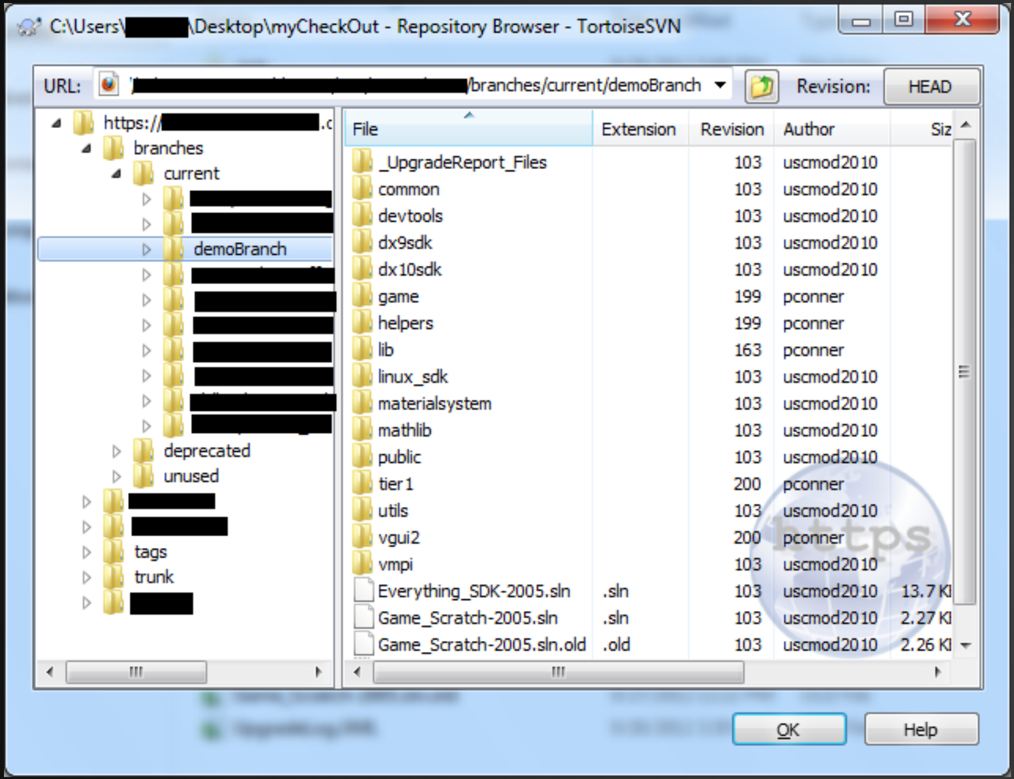


Select TortoiseSVN > Repo-browser.

Navigate to your branch.

Cleaning Up

(cont)



C:\Users\[redacted]\Desktop\myCheckOut - Repository Browser - TortoiseSVN

URL: [redacted]/branches/current/demoBranch   Revision: HEAD

- https://[redacted].c
  - branches
    - current
      - [redacted]
      - demoBranch
      - [redacted]
    - deprecated
    - unused
  - [redacted]
  - tags
  - trunk
  - [redacted]

File	Extension	Revision	Author	Siz
_UpgradeReport_Files		103	uscmod2010	
common		103	uscmod2010	
devtools		103	uscmod2010	
dx9sdk		103	uscmod2010	
dx10sdk		103	uscmod2010	
game		199	pconner	
helpers		199	pconner	
lib		163	pconner	
linux_sdk		103	uscmod2010	
materialsystem		103	uscmod2010	
mathlib		103	uscmod2010	
public		103	uscmod2010	
tier1		200	pconner	
utils		103	uscmod2010	
vgui2		200	pconner	
vmpi		103	uscmod2010	
Everything_SDK-2005.sln	.sln	103	uscmod2010	13.7 K
Game_Scratch-2005.sln	.sln	103	uscmod2010	2.27 K
Game_Scratch-2005.sln.old	.old	103	uscmod2010	2.26 K

As stated at the end of the Merging section, this branch is no longer able to be worked from. Once we Merge, we're done with it.

To keep things nice and tidy, click and drag it into the 'deprecated' folder. A message log window will come up, simply explain that you are moving it because it is finished and Merged. We want it out of the way, but we would like to keep a record of it - so we don't just delete it.

We're Done!

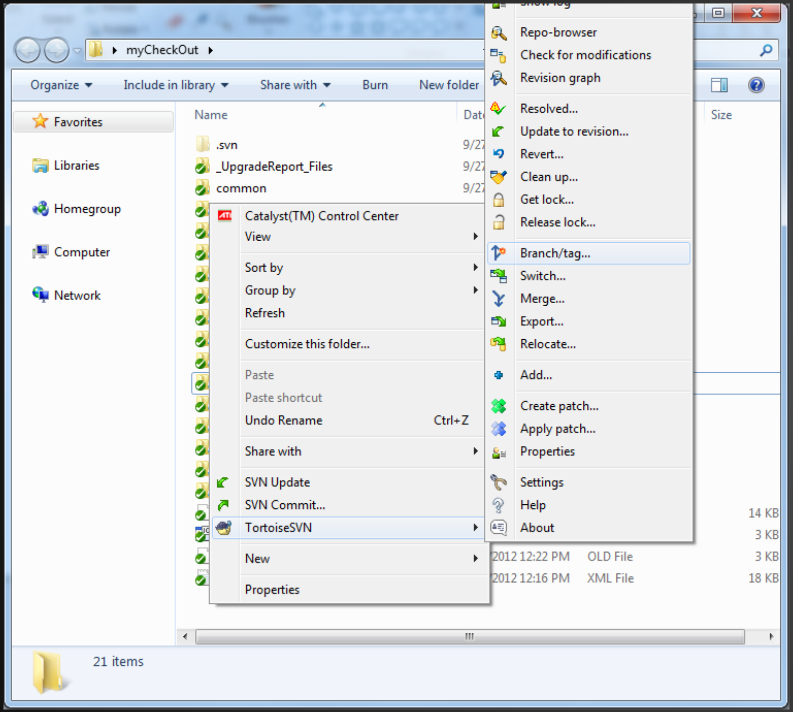
Congrats!

We have now finished a feature development/integration cycle.

If everything went well, give yourself a pat on the back.
If not, blame SVN and your co-workers.
You did everything right, it's everyone else.

Now we will go over a few extra SVN topics...
(Including resolving conflicts)

Tagging



Just like Branching, select TortoiseSVN > Branch/tag... Now, instead of choosing '/branches/current', choose '/tags'.

Name the Tag something meaningful.

What is a Tag? Essentially, it is identical to a Branch, it only differs in purpose. A Tag is a snapshot of the trunk. After reaching a milestone, or implementing a big feature, we may want to save a copy of the trunk. This allows a non-changing copy to exist (Tags are almost never changed, that are concrete snapshots.)

Using SVN Tools

As you most definitely noticed, there are many more items under the TortoiseSVN context submenu.
I'd like to talk about how some are used to aid you in working.

Repo-browser - Useful for looking through the whole repository (browsing it). It can also be nice to see what section (branch/tag/etc) you are in, as it selects where you are when you first open it. As in the Cleaning Up section, you can move stuff around in it.

Show log - Very helpful for keeping track of changes at any level. Shows revision numbers (good reference numbers for what version the software is in). This is where you can see who committed and their messages. Trust me, you will spend tons of time here - write good messages.

Using SVN Tools

(cont)

Resolved - this is for when you have Conflicts (we can go into more detail in the next section.)

Update to revision... - Nice for going to a specific version of the software. Extremely useful if the current version is broken.

Revert - The opposite of Commit. This will undo any changes that you have made, and change file(s) back to what is currently on the repository.

Clean up... - If you ever get an error about not being a working copy, or not having a lock... Try using this - will probably magically fix it all.

Using SVN Tools
(cont)

Get lock... / Release lock... - Like a semaphore, restricts/allows access to change areas in the repository. This could be helpful if you want to make sure nothing changes while you are doing something. But, it is dangerous. If you lock something.. then go on vacation, or delete that checkout, or your computer crashes, everyone is locked out... I try to stay away from this. It can easily be substituted with good team communication.

Relocate - Helpful if your repository has changed URLs. Allows you to update the URL with no need to re-check things out.

Using SVN Tools

(cont)

Diff - This will check what are your local changes of a specific file with respect to what is on the repository.

Diff with previous version - This will check what are your local changes with respect to the repository before your last commit.

Holding shift, as you Right-Click, you can access:
Diff with URL - This will check what are your local changes with respect to the given URL. This allows nice and easy comparison between, say for instance, your branch and the trunk.

Delete (keep local) - This is like what I described in the Deleting section, where it keeps the file (it just auto-unversions on commit.)

Dealing With Conflicts

Most of the conflicts that you will come across are just called 'Conflicts'. They arise when SVN cannot figure out how to combine two files. This can occur if two people change the same line of the same file. When you try and combine the files (via and Update or MRoR) the file(s) become marked as Conflict. You can use the 'Edit Conflicts' tool to fix this.

To get to this tool, you can either Right-Click the file from the Update/MRoR window and select 'Edit conflicts'; or you can Right-Click the file in the folder (windows explorer) and select TortoiseSVN > Edit Conflicts. Note: Conflicted files appear with a triangle warning sign on their icon. Once you're in the Edit conflicts tool, you can use the arrows at the top to fix the files as you see fit. Finally, Mark as Resolved, and save. Fix'd.

Dealing With Conflicts
(cont)

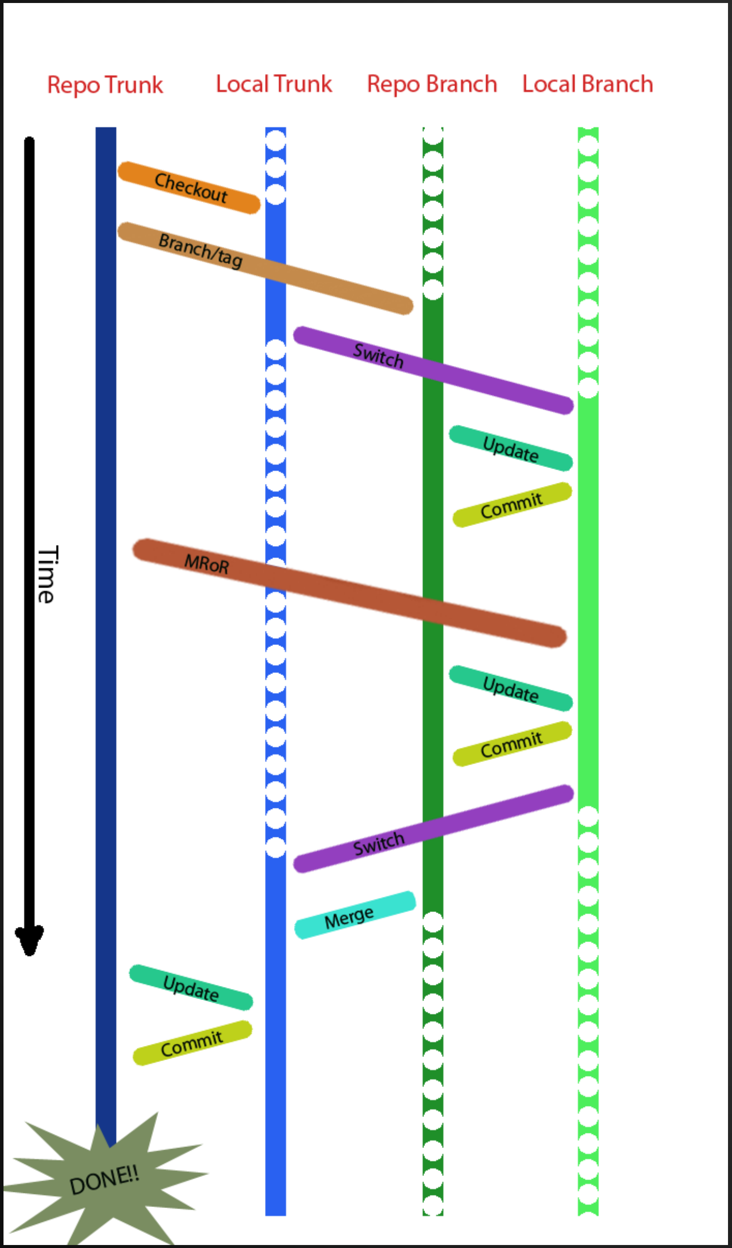
The far more problematic conflict is the dreaded 'Tree Conflict'.

This is a very complicated conflict, that may arise for a number of reasons. To properly explain them all, it would take too much space here (and probably wouldn't explain it as well.)

Instead, here is a link to TortoiseSVN's Resolving Conflicts page. This will give detailed fixed for both kinds of conflicts.

http://tortoisesvn.net/docs/nightly/TortoiseSVN_en/tsvn-dug-conflicts.html

Neato Diagram



This shows the simplified version of a feature branch's life cycle.