Subversion Tutorial

based on a tutorial written by David Wolff that is available at
http://www.cs.plu.edu/~dwolff/svn-tutorial/svn-tutorial.html

Subversion (SVN) is a free/open source version control system. It manages a tree of files and directories over time, keeping track of changes to all files under its control. It is a valuable tool for developing large scale projects with a group or individually. A tree of files is placed in a repository. Subversion remembers every change ever made to your files and directories. This allows you to recover older versions of your data, or examine the history of how your data changed.

Other features:

1. Works over a network, so that your code can be kept on a central server. Team members can download *working copies* of the code, make their changes locally, and *commit* their changes to the server.
2. Keeps track of all previous versions of your code. If you make a mistake, you can revert back to any previous version.
3. Helps with concurrent editing of the same source file. If you and your partner make changes to the same part of a file at the same time, Subversion will alert you and ask you which version to use.

Most (if not all) software is developed using a versioning system of some sort. It is an extremely valuable tool when working on a code base of a non-trivial size.

Requirements

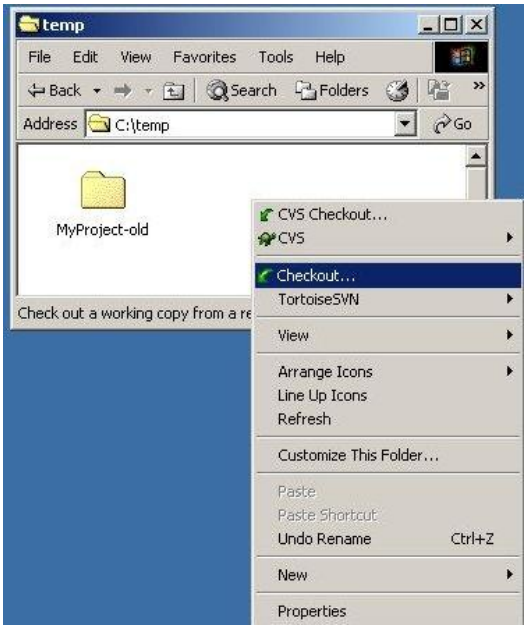
1. A SVN client
    1. If you are using Windows, try "Tortoise SVN" from http://tortoisesvn.tigris.org. This tutorial focuses on this client.
    2. For Linux/Unix the command line SVN client can be installed for your distribution of choice. See: http://subversion.tigris.org .
    3. For MacOS X users, subversion is available through Fink, or see http://subversion.tigris.org.
2. A Subversion repository.
    1. Is already created for you.

Checking out a local Working Copy

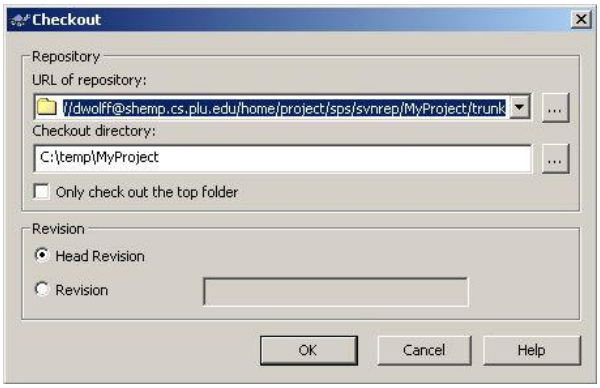
Now that you have installed the software, we need to "check out" a working copy of our repository.

Start Explorer and create a folder where you want to checkout the repository. Eg. Create a folder on the Desktop or on your local hard disk. “C:\project”

Right click in the parent folder and choose "Checkout..":



The following dialog will appear:



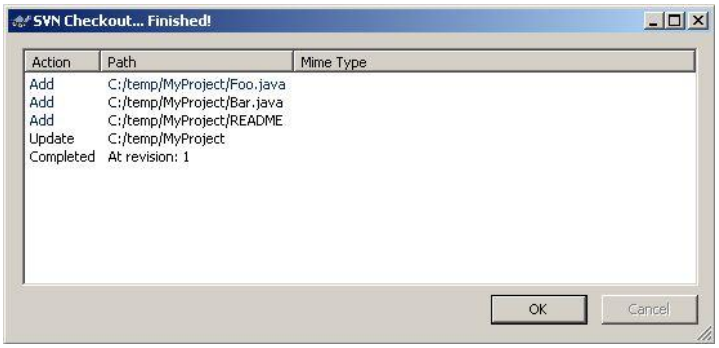
Use the url that was given via e-mail to you for your repository e.g.

https://miller@svn.fh-luebeck.de/svn/repos/thesis/miller

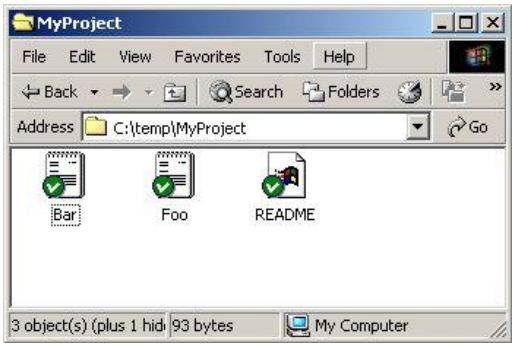
Press OK

Here you will use the same URL that we used in the import step above. In addition, you should choose the destination folder for your working copy. In this example, all files contained in the "trunk" will be placed in the destination folder.

After entering your password (twice) you should see something like the following:



The "Add" indicates that the file is "Added" to your local working copy. After clicking Ok, navigate to your local Project folder. You should see something like the following:



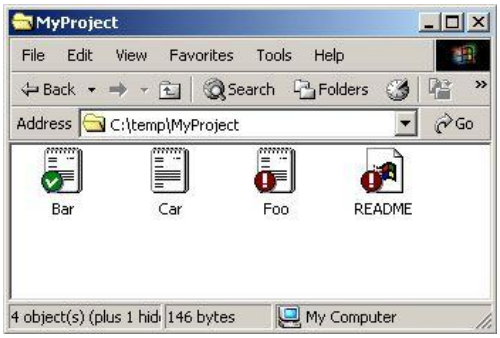
The green check marks indicate two things: (1) the file is under subversion "control", and (2) that the file is equivalent to the most recently checked out version of the file (called the BASE).

After that you can move the directories on your hard disk to the right position if necessary. E.g. Source code where needed. SVN will recognize that the files are still under revision control.

Working with your Working copy

Now that you have a SVN controlled working copy, you can start coding away. It is important to remember that if you make any changes to the file name, file location, add files, or delete files you need to let subversion know about it. In this example, we add a file, and modify a few files. In general, you might have changes that involve file name changes or file location changes. In those situations, you must "tell" subversion about those changes using one of the options in the TortoiseSVN menu like "Relocate.." or "Delete". You should have subversion make these kinds of changes rather than doing them yourself.

For example, suppose that during the course of our work, we have made changes to the files Foo.java and README. Also, we have created a new file to add to our project called Car.java. Then the working copy would look like this:



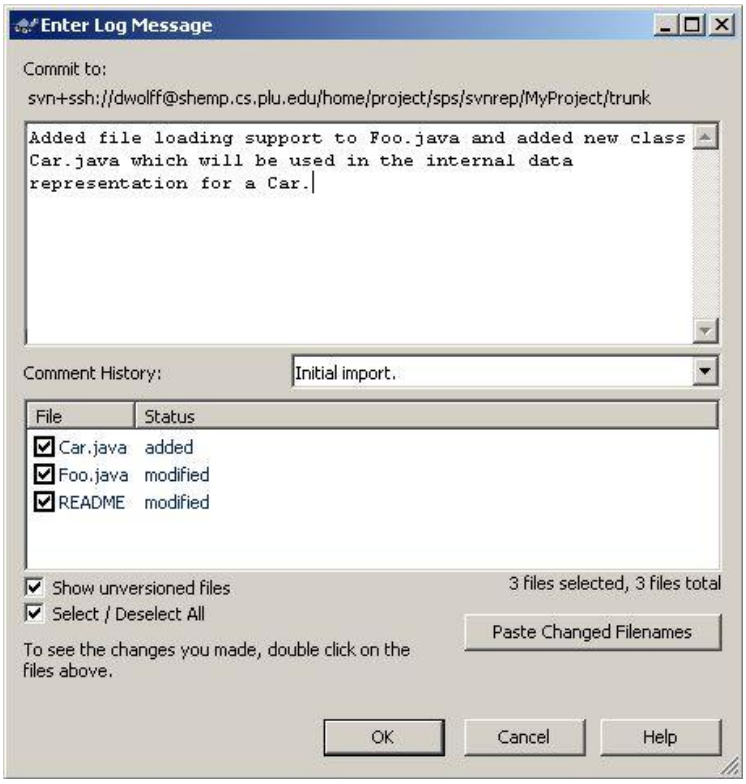
The exclamation marks in the red circles mean that the files have been changed since they were checked out. The file Car.java does not have any symbol on it because SVN does not know about it yet. To add it to our SVN project, right click on the file and choose TortoiseSVN -> Add.

Once that is done, the file is marked for addition. There is no change to the image of the file, because it has not been "officially" added to the server's repository. The file is not truly added to the repository until you commit your changes to the repository by doing a "Commit." The commit operation sends all local changes (file adds, file changes, deletes, moves, renames etc.) to the server. Generally, you will execute a commit when you are done working for the day.

Commiting your Changes to the Repository

When you have finished your work for the day, it is time to commit your changes to the repository. However, before you do so, it is a good idea to make sure that you have merged any changes that might have been made to the repository after you checked out your working copy. To do so right click in your working copy and choose Update. If there were any changes to the repository since your last checkout, SVN will try to merge those changes with your working copy. If there are conflicting changes, SVN will notify you and you may need to manually resolve those conflicts. For more information on this, please see the links below.

Now that you have updated your working copy, you are ready to commit your work to the repository. To commit your changes (and add the files marked to be added) don't highlight any files, just right click in the blank space and choose Commit. You should see a screen like the following:



Every time you commit, you will be asked to write a comment. This is a very useful way of keeping a record of the changes you make to your project. Every time you do a commit, write a brief, but detailed description of the changes you made in the text area.

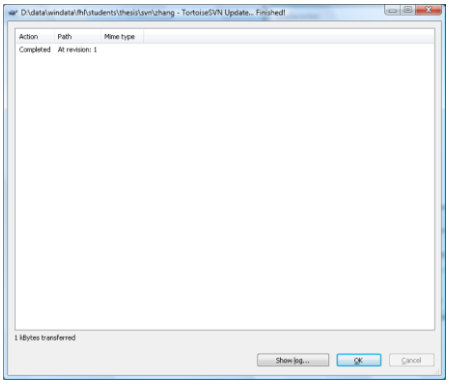
Click Ok, and your changes will be committed to the repository. When you are done for the day, and after you have committed your changes, you can safely delete the local copy of your module. In fact, it is a good idea to do so. The reason for this is that the next time you sit down to work on your project you will check out a new copy of the module, along with any changes that may have been made by your group partners in the meantime. The other option is rather than deleting your working copy you can just make sure that you do an "Update" before you start working.

Update your Working Copy

If the instructor another team member has also access to the repository you need to make regular updates especially **BEFORE** you make changes to documents

Right click on the local working copy and choose SVN update.

You hopefully see the following which means that your working copy is uptodate.



Common Work Flow

For best results, you should use the following steps as a guideline when working with SVN. On a given day when you want to work on your project,

1. Check out a working copy from the repository. (Or, if you already have a working copy, run an "update" on your working copy.)
2. Make your changes, additions, etc. (write code!) Make sure that notify SVN of any new files or name/location changes of files.
3. Do an "Update" in your working copy. If there are others working on the code at the same time, and they commit changes to the repository, this will download those changes to your local copy. If there are conflicts (i.e. someone has changed something that conflicts with your local changes) you will be alerted and asked to fix those conflicts before you commit.
4. Commit your changes to the repository. Make sure you write an informative comment about your changes.

More Information

Of course, there are many other features to SVN that we have not covered here. For more information on SVN check out the following:

- The Subversion Book. This is an excellent reference for Subversion.
- Tortoise SVN home page.