What I know about SVN

Juan Delgado – Zárate
zarate.tv – dandolachapa.com – loqueyosede.com

Table of contents

Disclaimer

This document is ~~not finished~~ work in progress. I'll be more than happy to receive suggestions, improvements and corrections. Please keep in mind that I'm far from being an expert on source control and I might be wrong. You should see this more as a colleague telling you what he knows after some years using source control systems, no more, no less. If you see something wrong please do let me know. You can contact me at cuentame at zarate . tv.

What I know about SVN is intended as an entry point for starters although it might also help average SVN users. I highly doubt it's going to teach anything to advanced users.

Thanks

Thanks to anyone that shares knowledge, the Internet is f***ing cool. Keep up the good work.

License

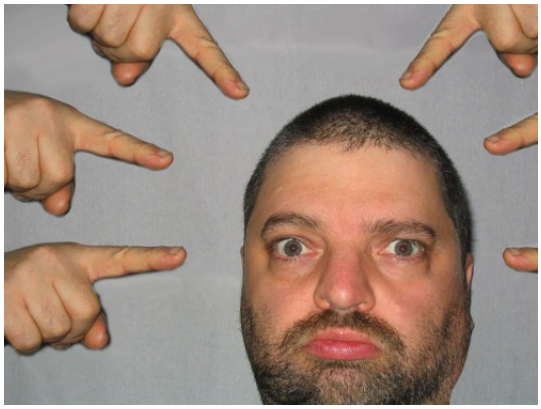
Introduction

From Wikipedia:

> Version Control Systems (VCS) or Source Code Management (SCM) is the management of multiple revisions of the same unit of information.

And by "unit of information" we understand .as files, .fla files, images, sounds, etc. Any type of file can be set up under source control.

Source control sounds nice, but it's all about BLAME:



(Photo credits)

Using source control you can find out WHO did WHAT, WHEN and WHY. This is specially helpful fox example when fixing bugs. Many times code doesn't speak for itself and something not obvious might be there leaving you staring at the screen clueless about why ***that call*** is being made. Well, fire up your source control client and find out who added it. Now you can send an email or IM whoever did it and ask.

It's also a sort of TIME MACHINE that you can use to go back in time. Actually, here's a picture of the one of the first version control systems:

Source control lets you jump and go back to old versions. Not only to versions where ***the annoying bug*** wasn't present, but also for example to the exact set of files you sent to your client (if you have tagged them, of course, we will come back to that later).

There are many VCS systems available and it's outside the scope of this document reviewing them all. We are going to focus on Subversion or SVN because it is one of the simplest and very well known VCS. If you want more information about other systems and differences between them, please go to the Links section.

Although source control shows all its goodness when working within a team it's also extremely helpful if you work on your own whether you are a developer, a designer or a marketing person. If you have ever seen in your folder something like this.

- Final_Proposal.doc
- Final_Proposal_modifications.doc
- Final_Proposal_modifications_b.doc
- Final_Proposal_sent_to_client.doc
- ...

Then you are going to love source control because it's going to do this nasty for you. And it's going to do it better. Some office suits like Google Docs are starting to integrate some sort of built-in source control, so I guess we can expect some of this into Office or OpenOffice in the not so long future.

How SVN works

SVN is a centralized VCS (as opposed to distributed) which means that there's a central place (called repository or repo for shorten) which you can see as the “master copy”. The repository then is accessed by people to get a fresh copy of whatever is in it. Then people make some modifications

and commit those modifications back to the repo so everyone else can get them.

You access the repo with a ***client***. There are tons of different clients and it's up to you picking up the one you like the most. If you are working on Windows, chances are you are going to use TortoiseSVN. If you are on a Mac maybe Versions. And if you are on Linux, probably the command line, although there are graphical clients too like RapidSVN. Just try some of them and pick up the one that suits you the most.

As a centralized VCS Subversion needs access to the repository to work. This means that if you repository is online you need internet access to commit, update, etc.

Glossary

Mastering one of the clients is ok, but it's much more important getting to know how SVN really works. Every client has its pros and cons and some clients are better than others. You should see them as browsers, what really matters is learning HTML not learning IE (well, specially not learning IE!).

So please take a moment to learn the basic terminology of SVN which, by the way, has a lot in common with other source control systems:

- Checkout: Get a fresh copy of the repo. Typically you do this once per project.
- Commit: Push your changes to the repo. Changes here means anything including adding/removing/renaming files or folders, modifying files, etc. **Whatever you do with your local copy will NOT reach the repository if you do not commit**.
- Add: If you want to put **new** files in the repo, you have to add them through SVN first, then commit.
- Update: Refresh your local copy with latest changes from the repo. SVN will try to merge whenever possible, although there might be conflicts (more information later).
- Revert: When you just want to get rid of your local changes and get a fresh copy from the repo.
- trunk, tags and branches. Typically a SVN repository has 3 different folders/sections:
  - The **trunk** is where most of the changes are done, where the daily job happens. It usually considered as “unstable” in the sense that the code hasn't usually gone through testing or not all the features are implemented.
  - **Tags**. When the trunk is stable or an important milestone is reached, typically you create a tag. You can see tagging as taking a picture of the project. Once it's taken, it's fixed, you shouldn't commit changes to that tag. Typical tags are "0.3", "RC 1", etc, but you can go creative as well with tags such as "GreatWhite" or "Short Circuit".
  - **Branches**. When a developer feels the need of implementing a rather big feature that might interfere too much with the trunk but don't want to keep everything local (or the feature is going to be implemented by a team) he/she can create a branch. Opposed to tags, developers keep updating/improving branches until the original problem is solved. Once solved, the branch is merged back with the trunk. SVN is well know for ***not*** merging branches very well so it's something usually avoided. This is not the case of other VC systems.

Working with SVN

Installing Subversion

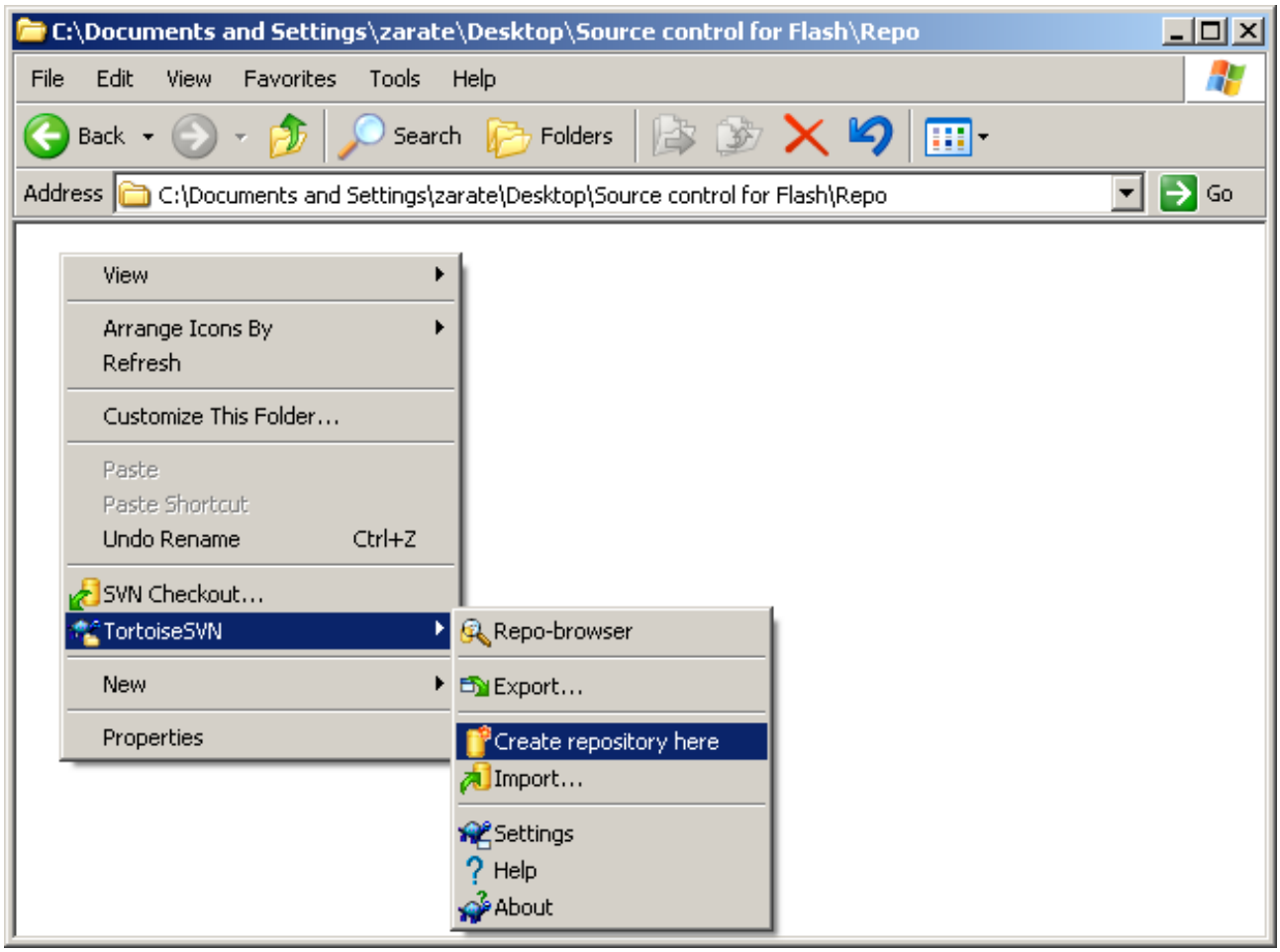
In Windows just install TortoiseSVN and you will be good to go.

Most Linux distributions come with the command line client installed, same as MacOSX. However, if you still prefer a GUI client, you can find some in the Links section.

Creating your repository

Setting up a simple SVN repository is very, very easy. SVN repositories do NOT need a server, just some space on the disk. They work pretty well on shared folders on the network too.

Using Tortoise, just right click on an empty folder and select TortoiseSVN > Create repository here:



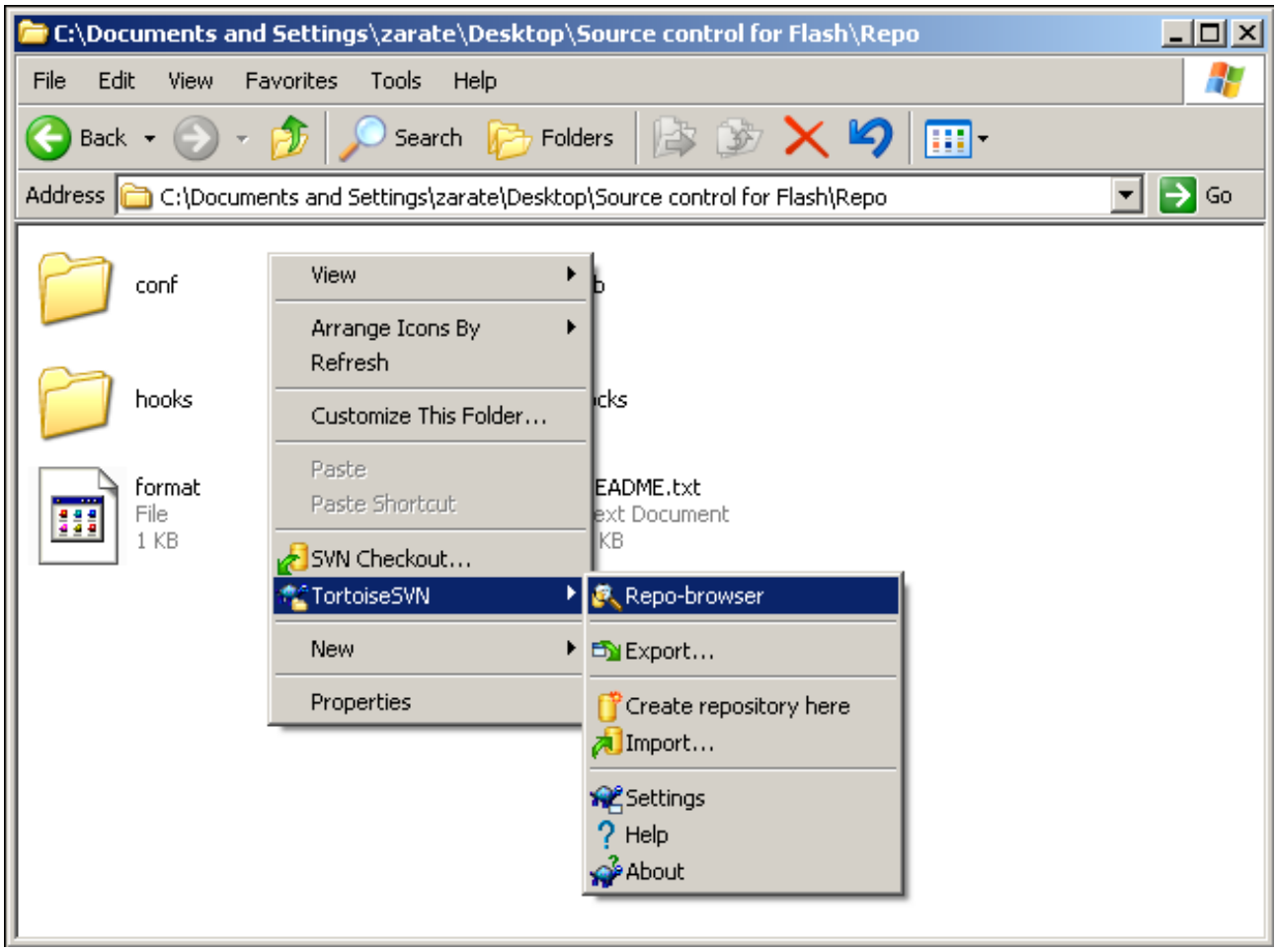
That's it, it's that simple. The actual SVN command is svnadmin create:

svnadmin create REPO_PATH

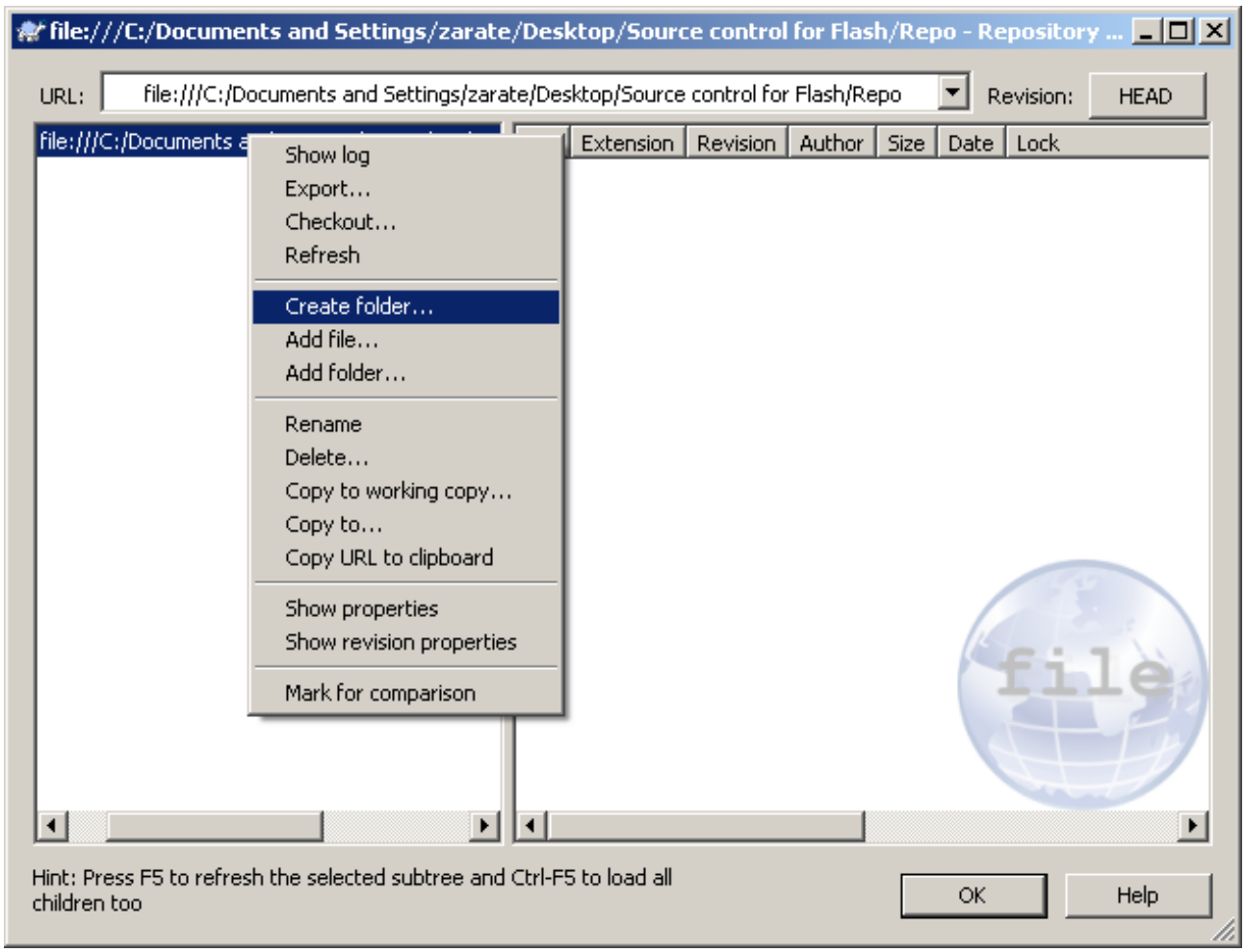
REPO_PATH must be an actual hard drive path, not an URL. Something like C:\repos\project1 or /home/user/repos/project1.

If you want to access your repo wherever you are without VPN connection, you need to put it under Apache. See Links section for more info.

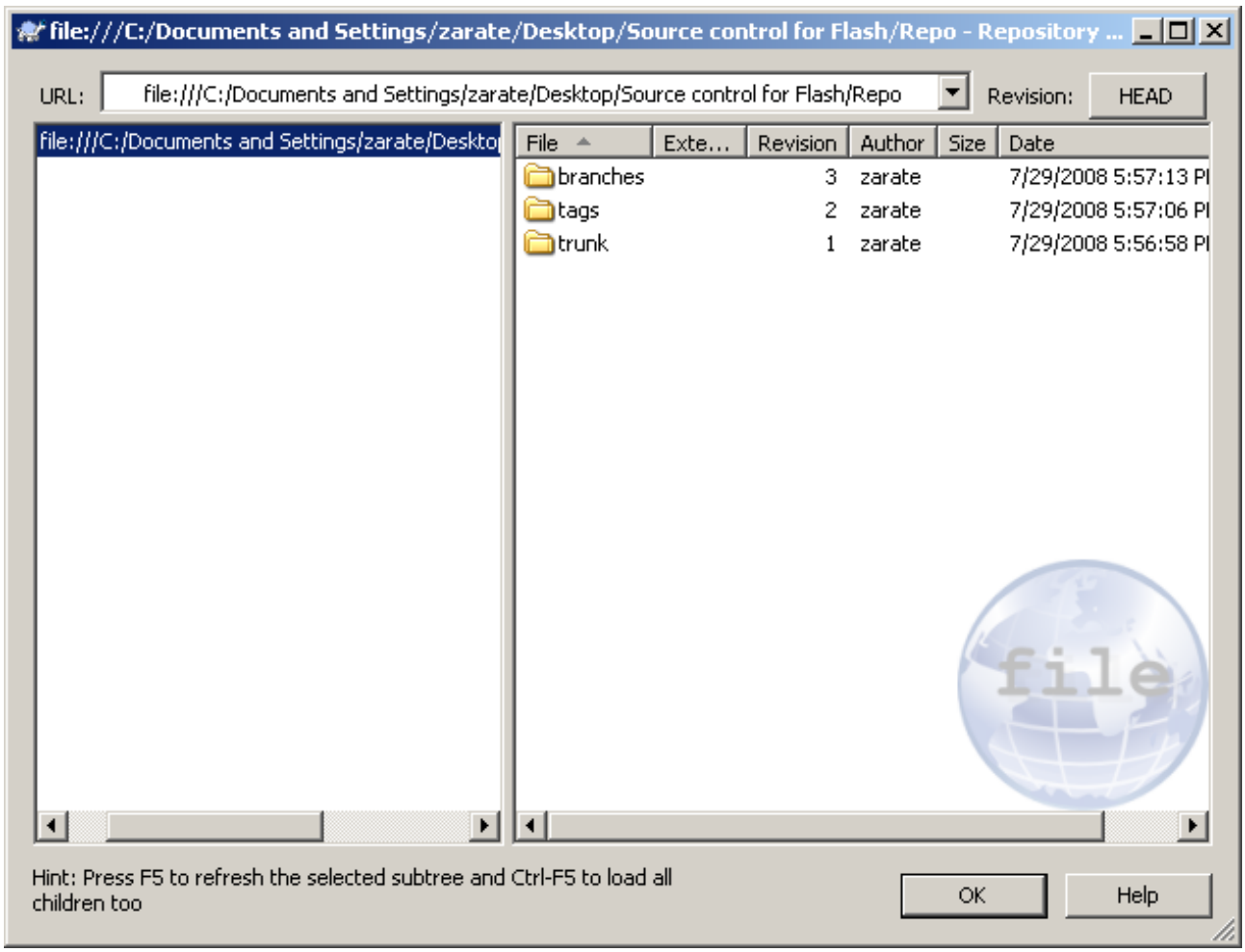
Once the repository has been created you have to manually create the trunk, tags and branches folders. Right click on the folder you just created that is now full of SVN stuff and selected "Repo-browser":



Now right click on the top address and select "Create folder":



Set "trunk" (no quotes) as name, enter a comment ("Initial set up" will do if you run out of ideas) and click enter. Repeat the same process for "tags" and "branches". At the end your repository should look like this:

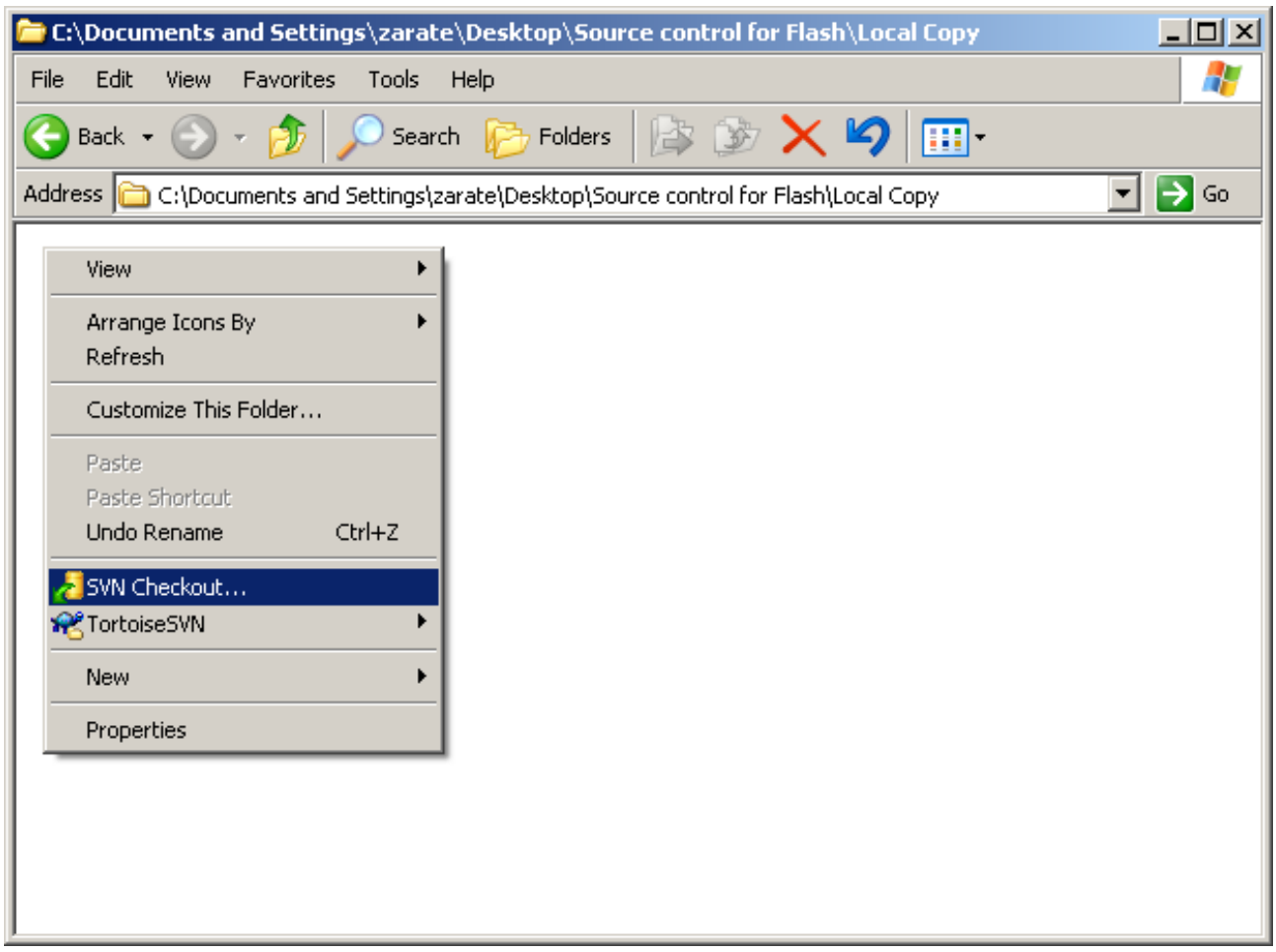


Now you repository is good to go.

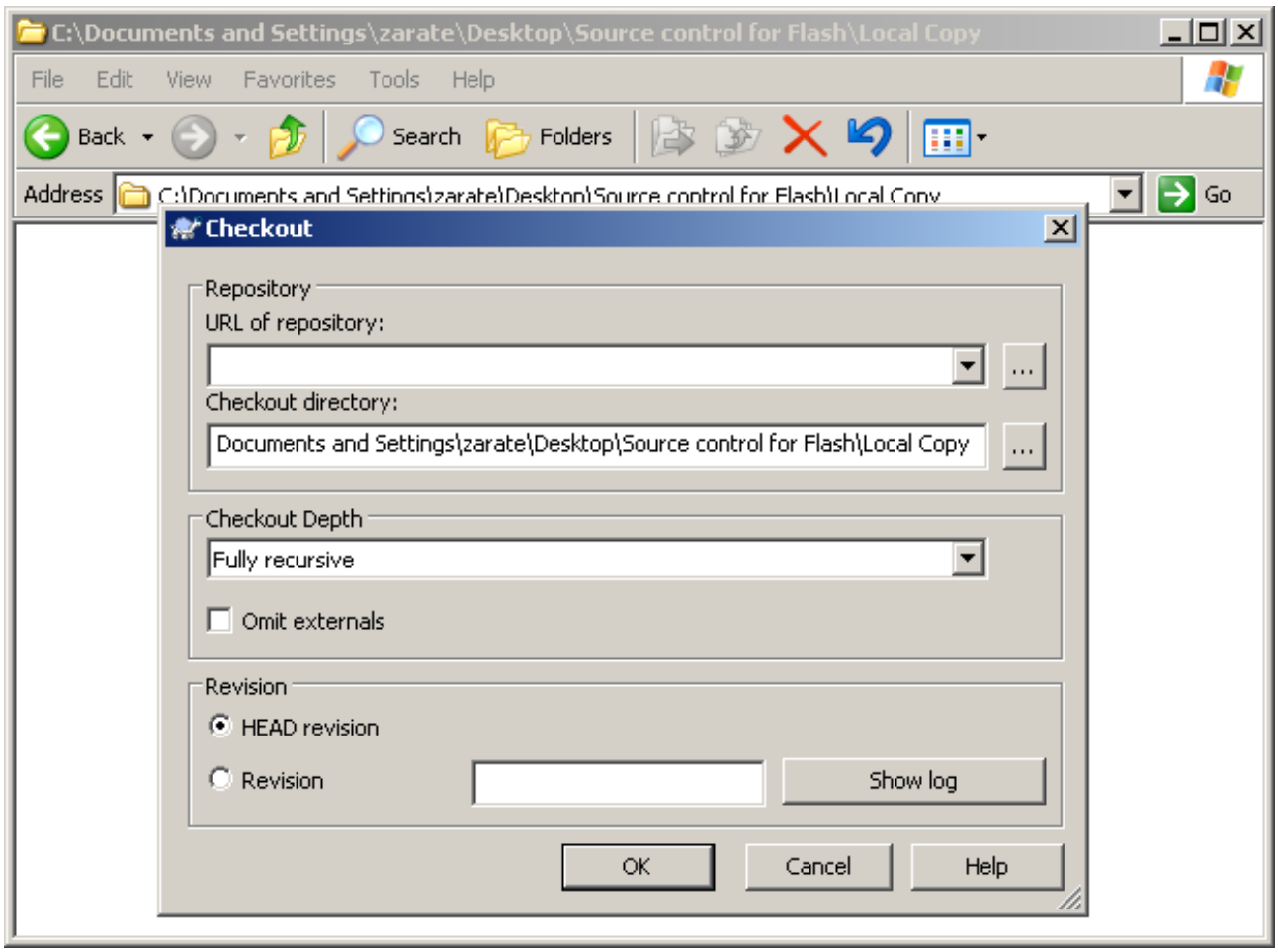
SVN daily routine

Check out

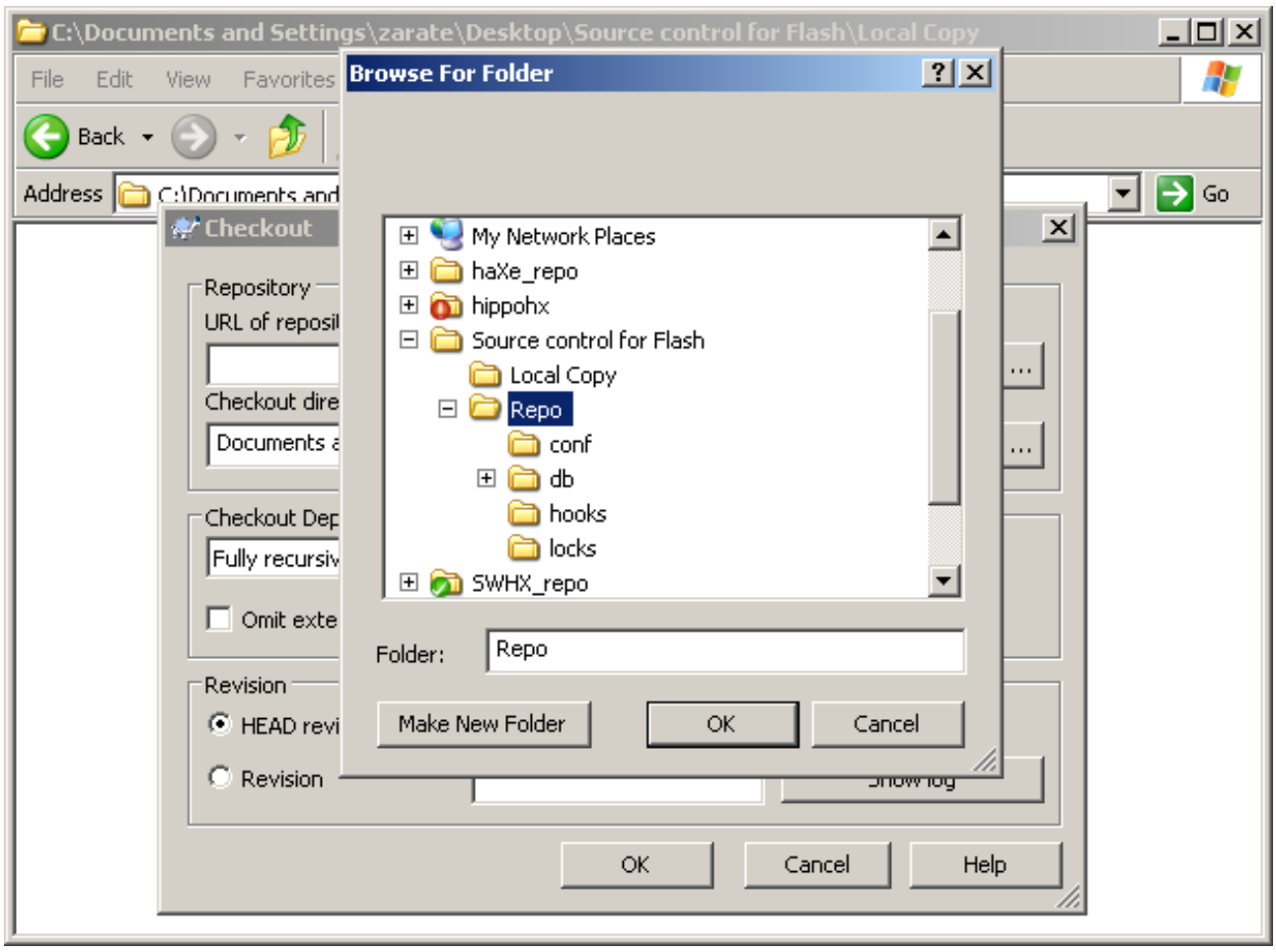
Once you have your repo up and running (or you got the URL from the project) first thing you have to do is check it out:



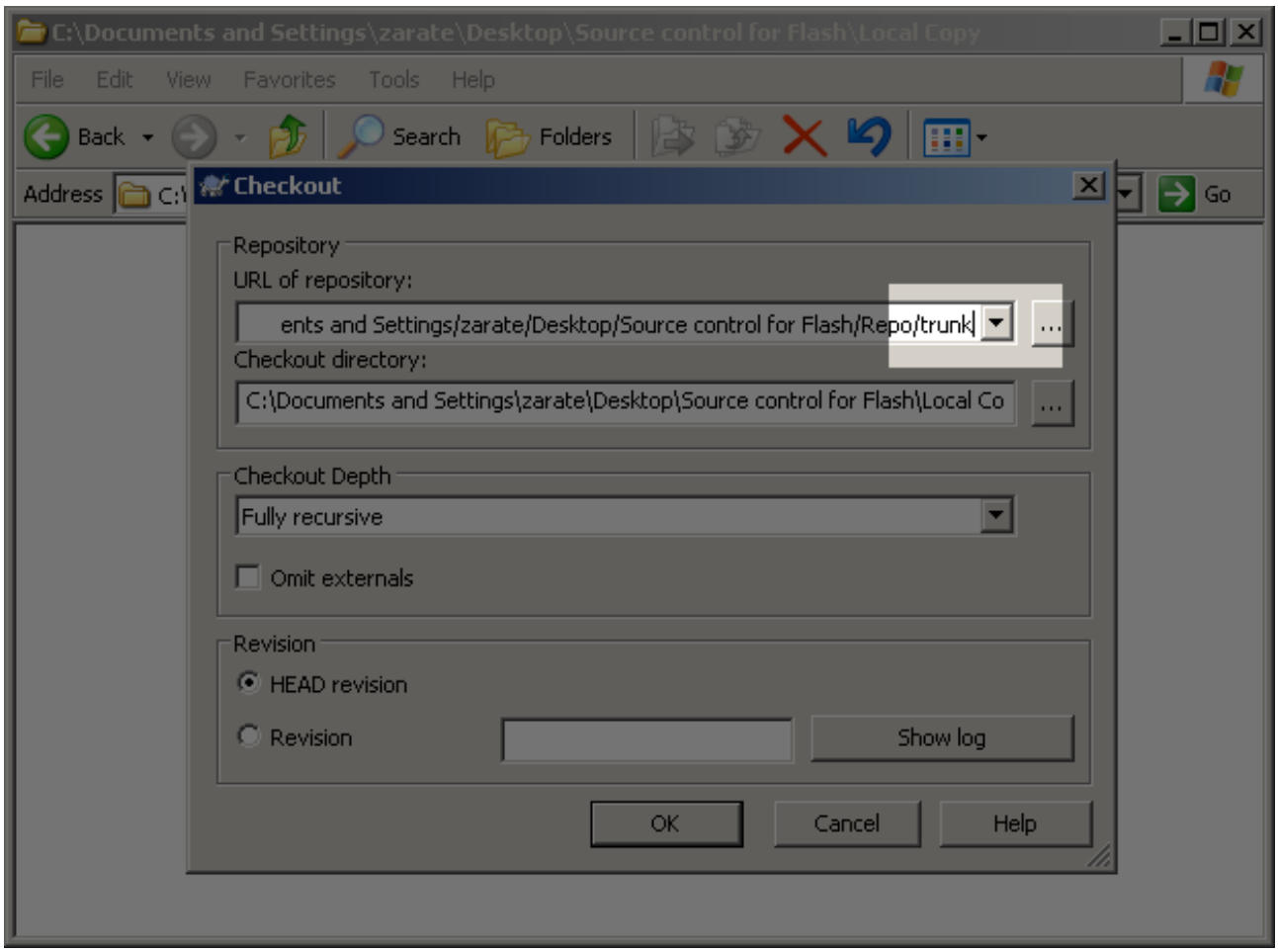
Then browse to the folder you created in the previous step using the "..." button close to the "URL of repository" field:



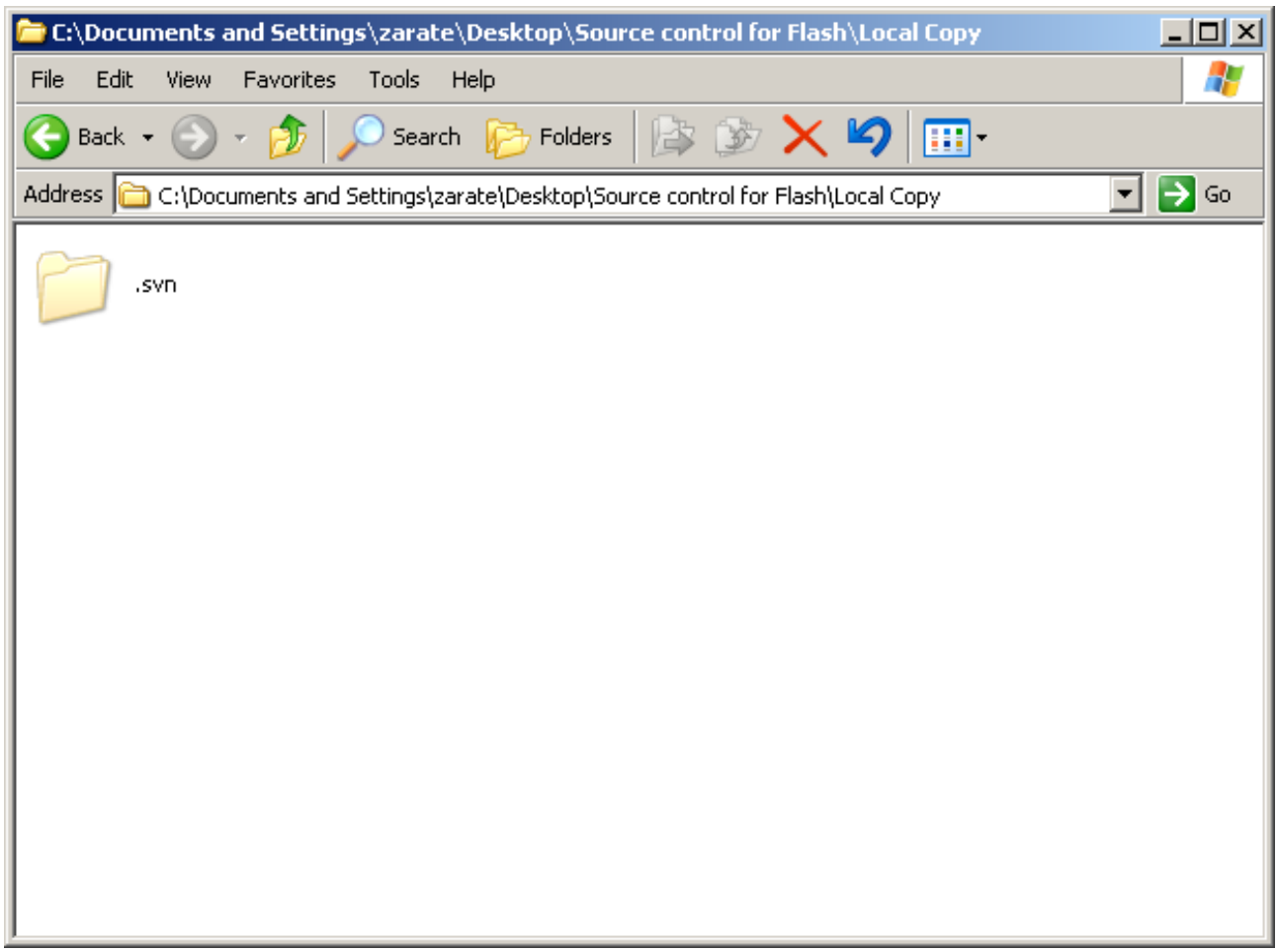
Select the folder and click OK:



Once selected, **double check that you are checking out the trunk**. If your client hasn't added that bit, just add it yourself:



That's it! Now you have a fresh copy of the project in your local folder to play with:



The actual SVN command is checkout:

svn co REPO_PATH/trunk

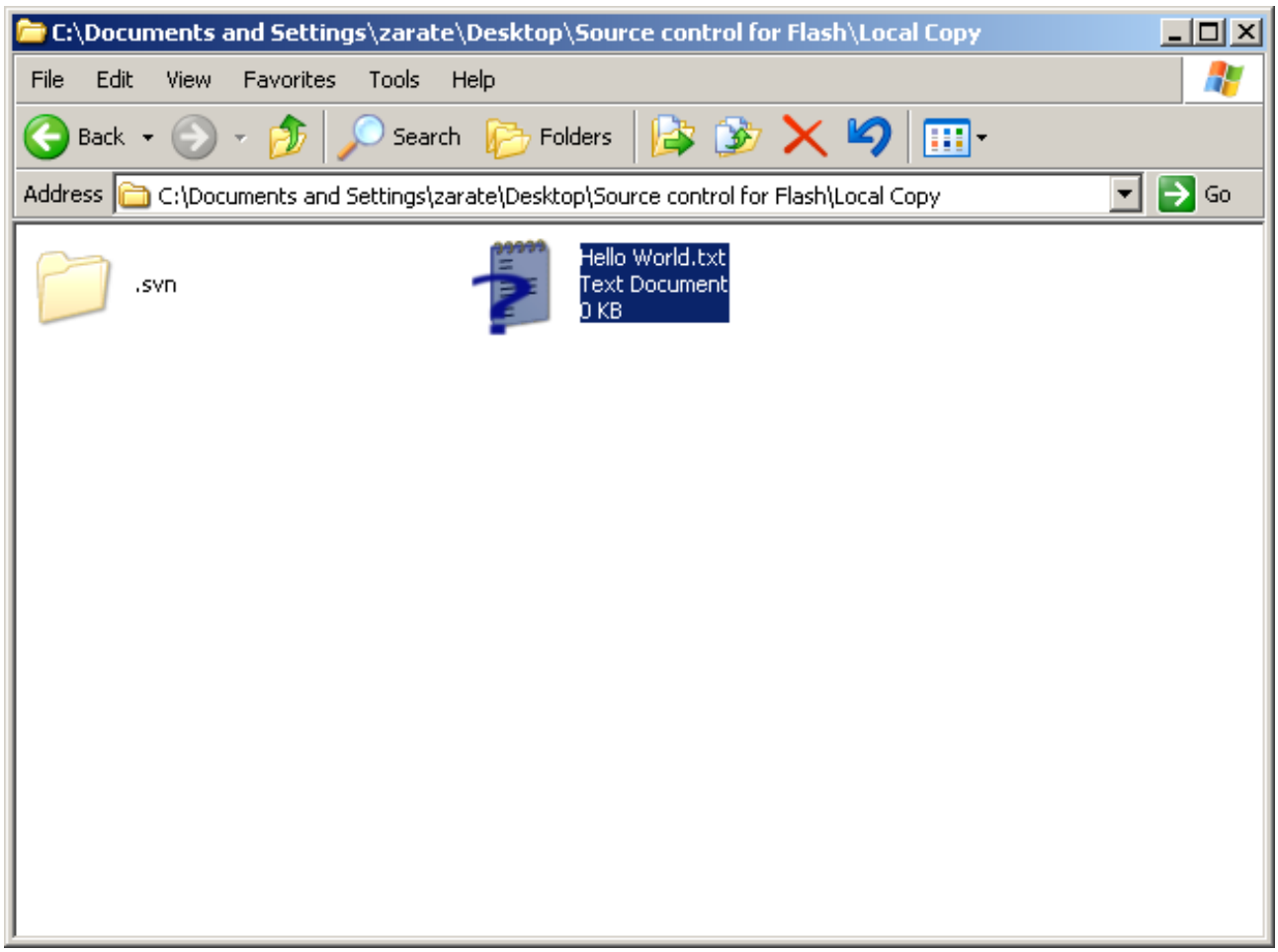
Because you have downloaded your own empty project there is only one hidden .svn folder (you might not see it depending on the configuration of your system). That folder is where SVN keeps track of changes and does its magic. Now, write this down as many times as you need to:

* THE FIRST RULE OF THE SVN CLUB IS YOU DO NOT MESS UP .SVN FOLDERS.
* THE SECOND RULE OF THE SVN CLUB IS YOU DO *NOT* MESS UP .SVN FOLDERS.

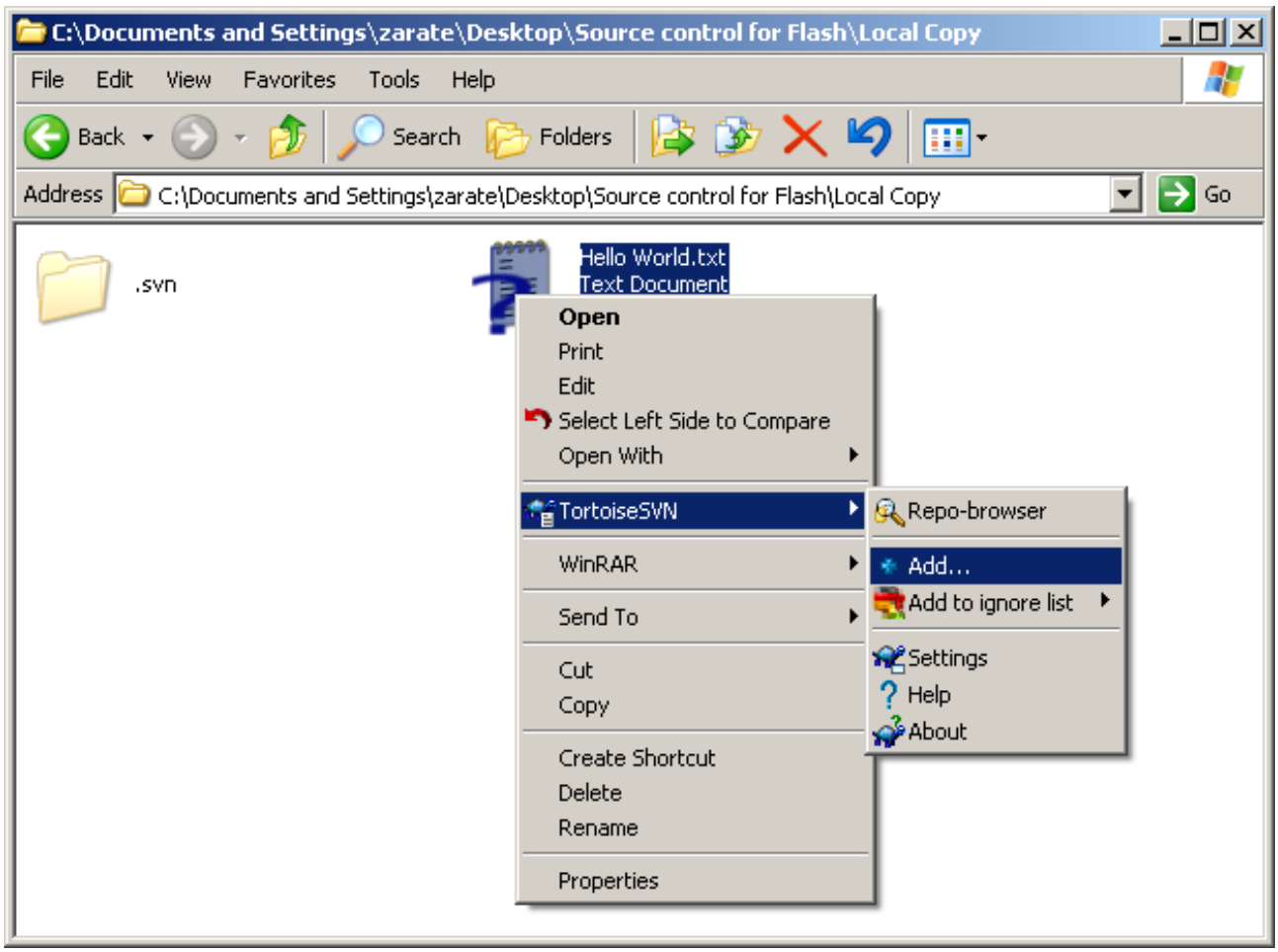
Really, you don't.

Add and commit

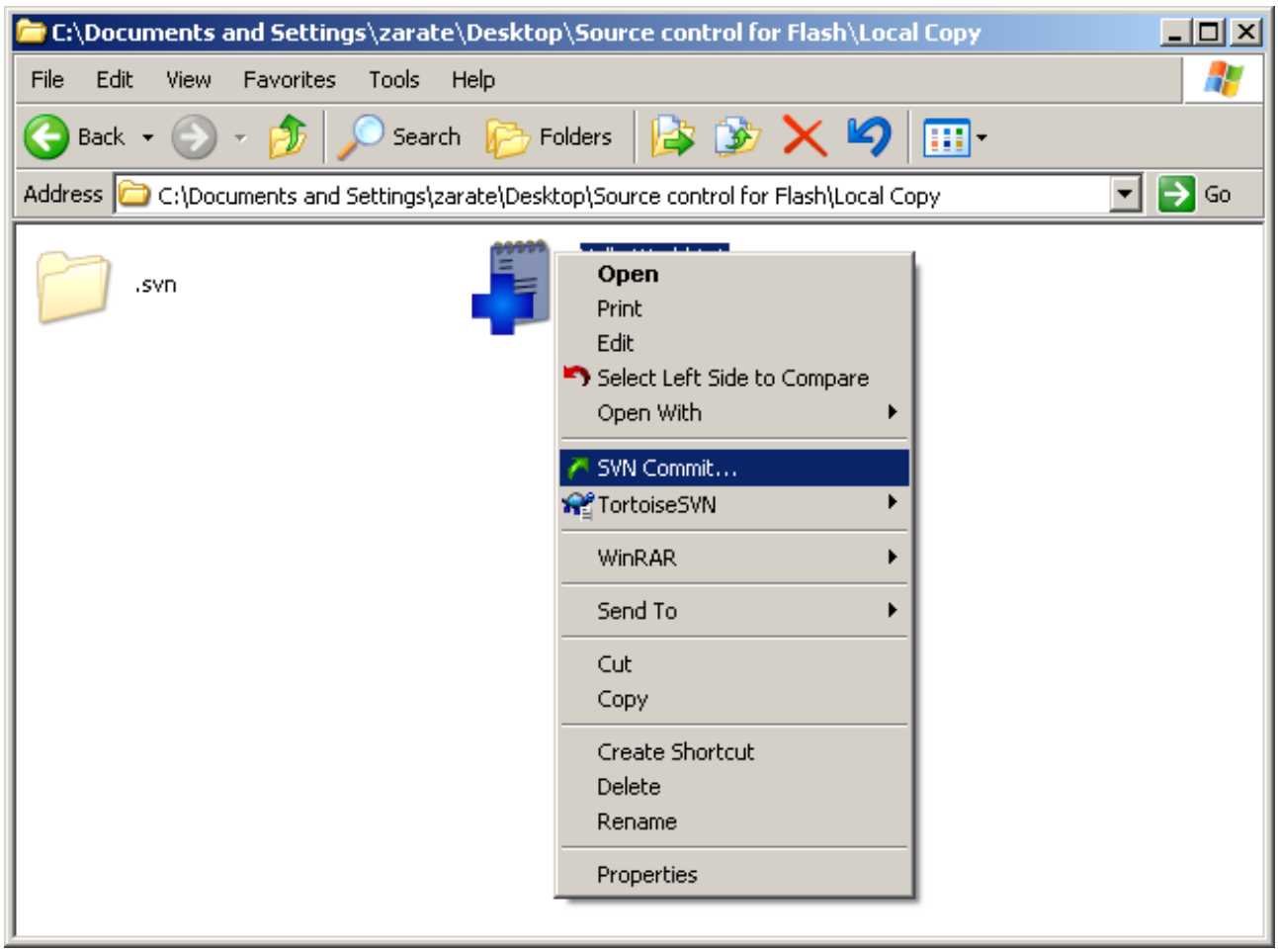
Now, create a simple text file with dummy content and save it. As SVN doesn't know it has to track it, it appears with an interrogation icon:



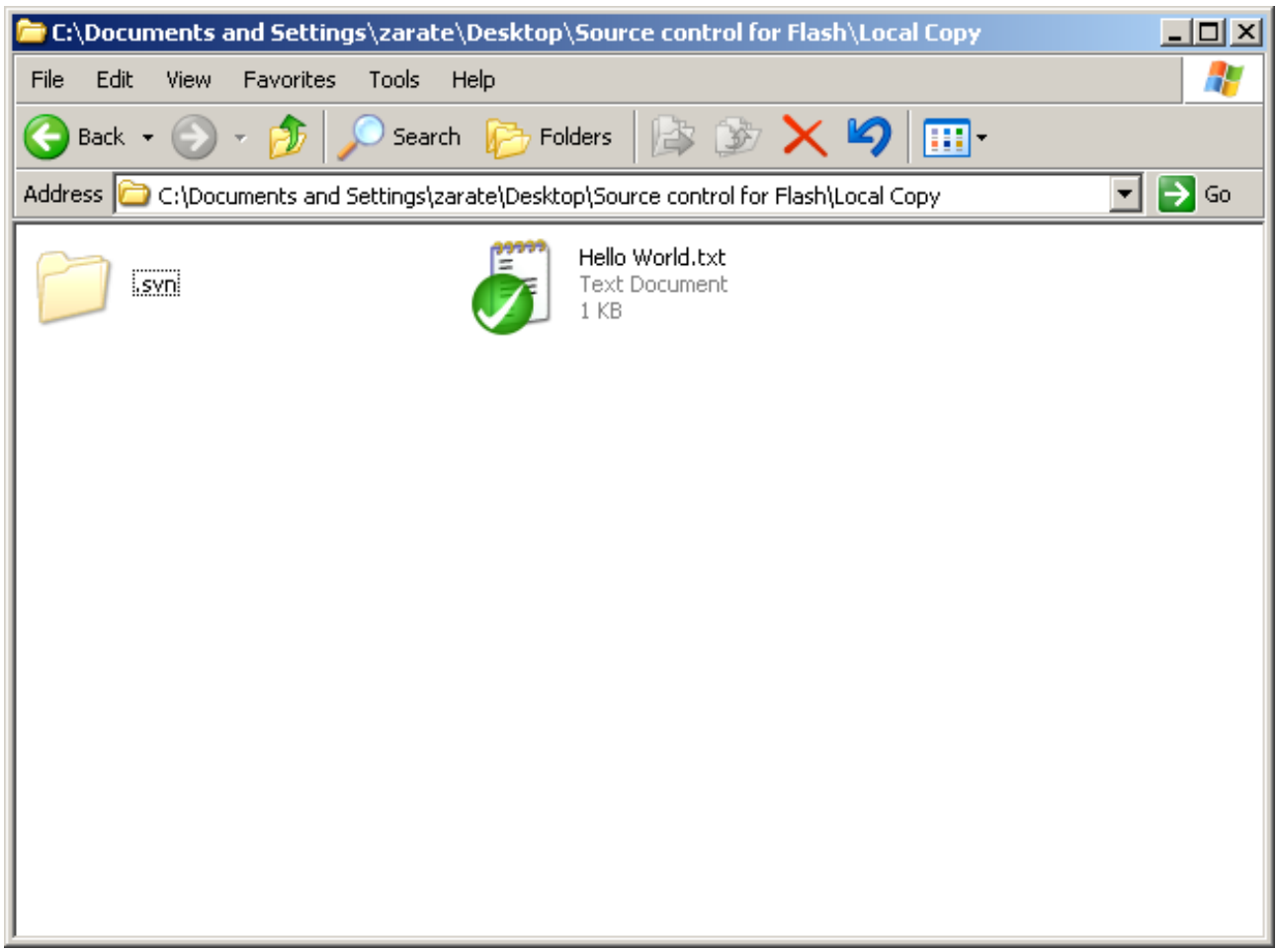
To add it to the repo you have to add it through SVN first so it knows it is a file it has to keep track of:



That will mark the file ready to be committed (icon changed to + sign) but so far has ***not*** made it to the repo, you **must** commit:



After committing the file has now a green icon that tells us our local copy is the same as the one in the repo:



That's it, done. This is the same as executing add and commit commands:

svn add Hello World.txt
svn ci -m "Adding my first file!" Hello World.txt

A note about what and when you have to commit:

> When you commit a change to the repository, make sure your change reflects a single purpose: the fixing of a specific bug, the addition of a new feature, or some particular task

From Subversion best practices. Also check the difference between a good and a bad log message:

BAD

> Fixed bug

GOOD

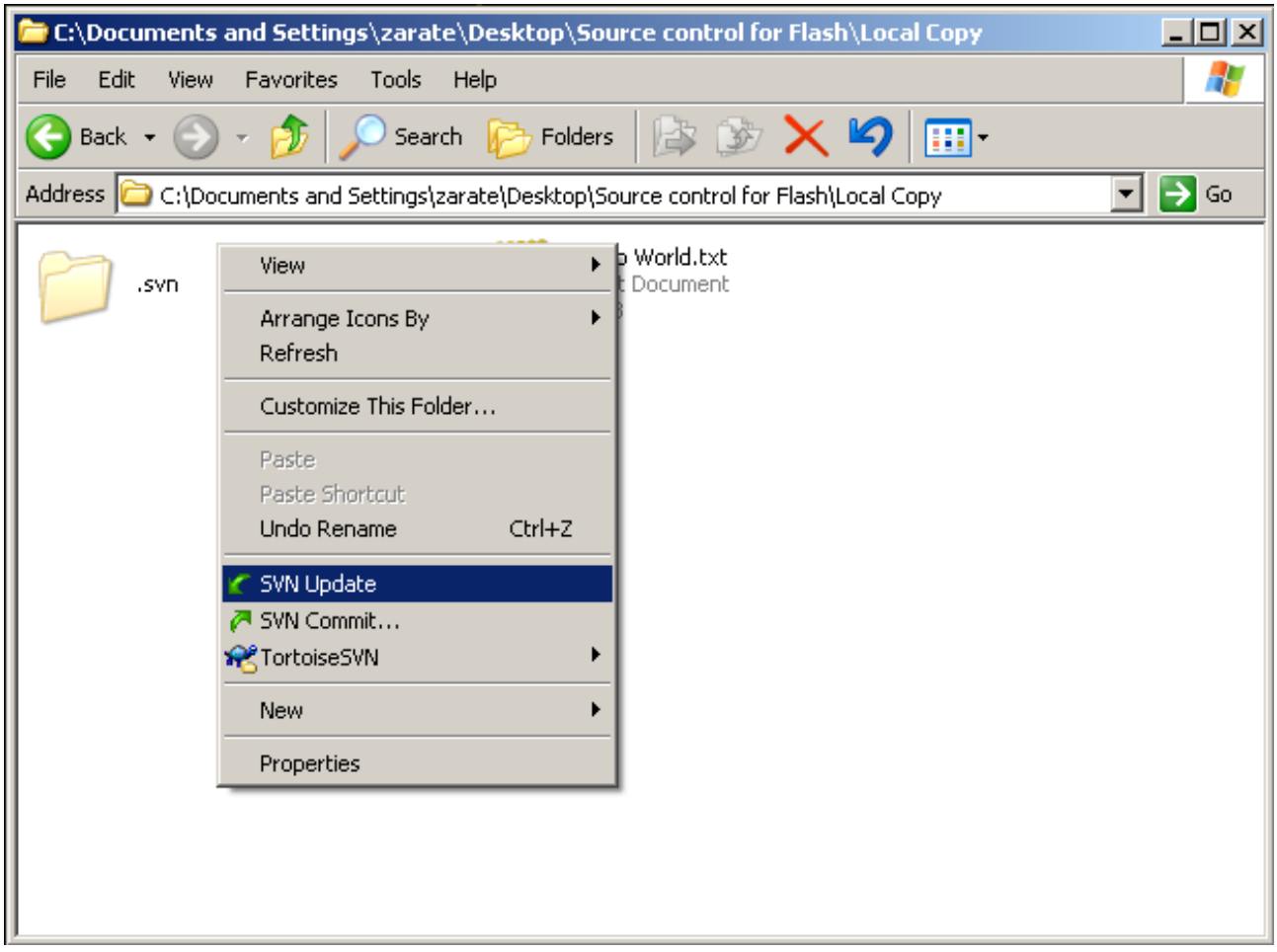
Fix issue #1729: Don't crash because of a missing file.

* get_editor.c
  (frobnicate_file): Check that file exists before frobnicating,
   Take an absolute path instead of an relative one.
* do_thing.c
  (frobnicate_things): Update calls to frobnicate_file.

From OSCON 2004, SVN best practices.

Update

If you are not working alone, you need to get the changes other members of the team have done:



That the same as using the update command:

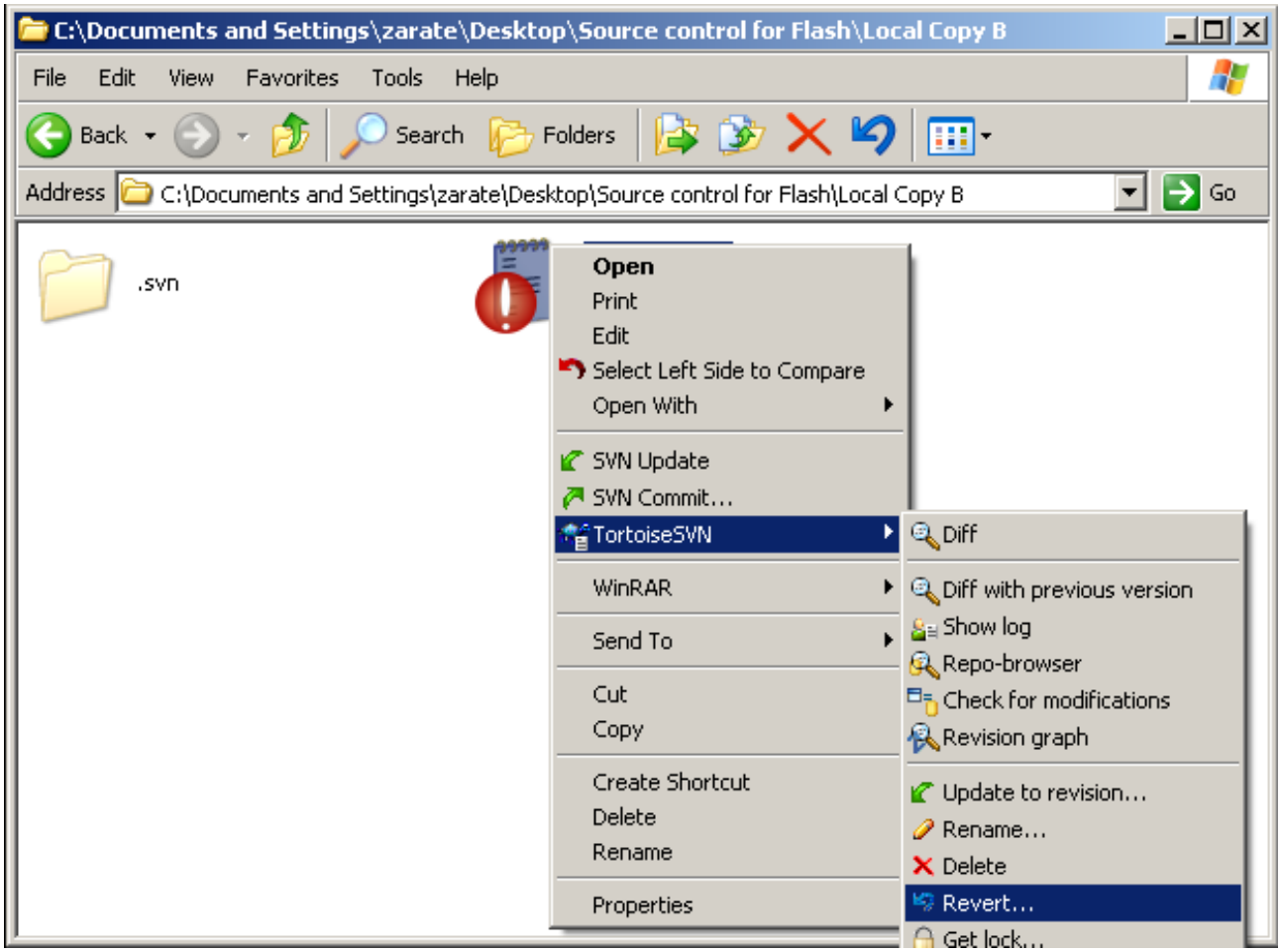
svn update

Note that you cannot commit your changes unless you have the latest copy from the repository That

means that if someone committed a new version of the file you are editing while you were doing your changes you are forced to update your copy **before** committing new changes.

Revert

Sometimes you have been modifying a file and at some point you decide that you have to go back to the latest version on the repo. To do that you use the revert command:



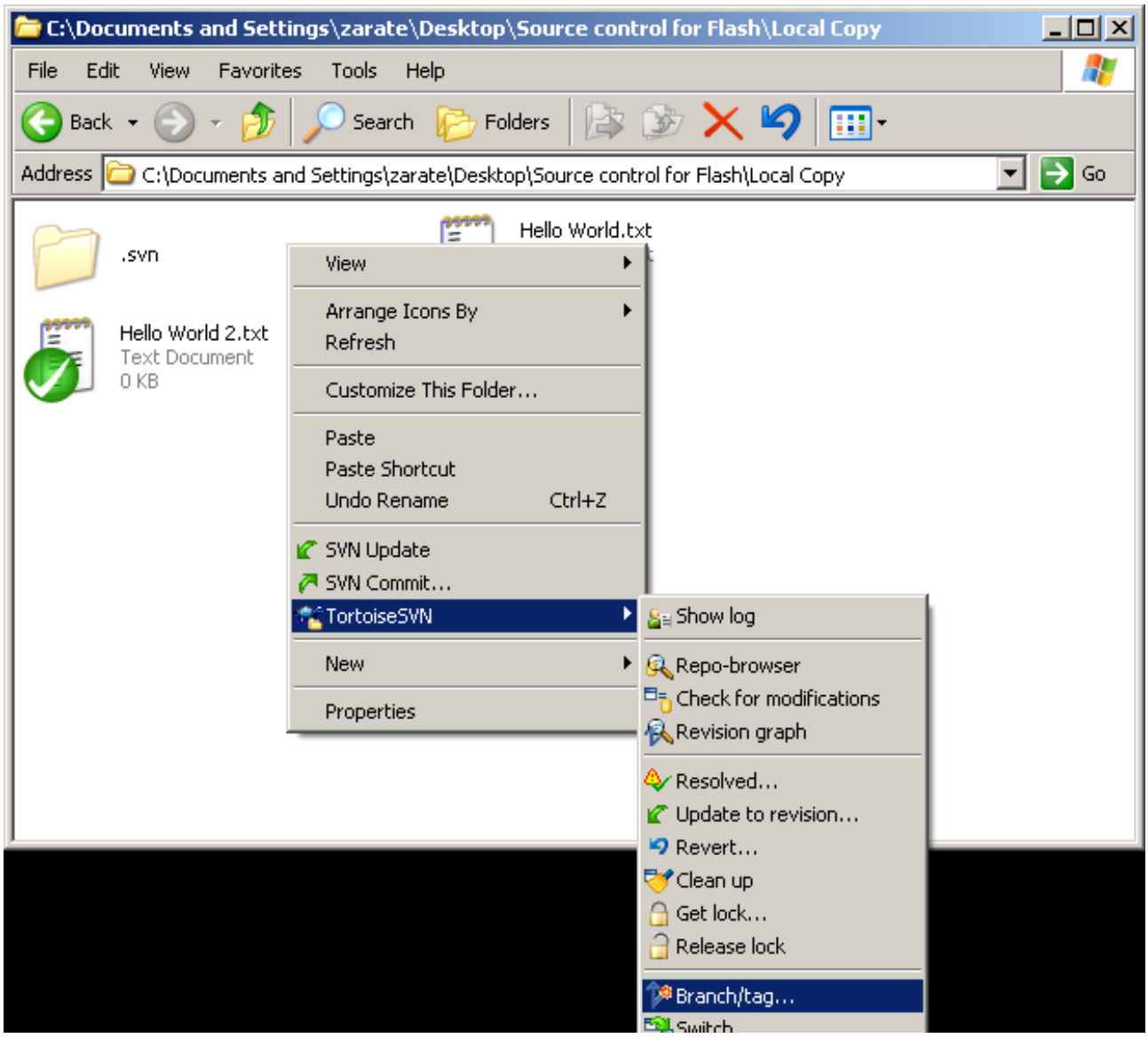
That will undo your local changes and bring back latest version from the repo. This is the equivalent of running the revert command:

svn revert FILE_PATH

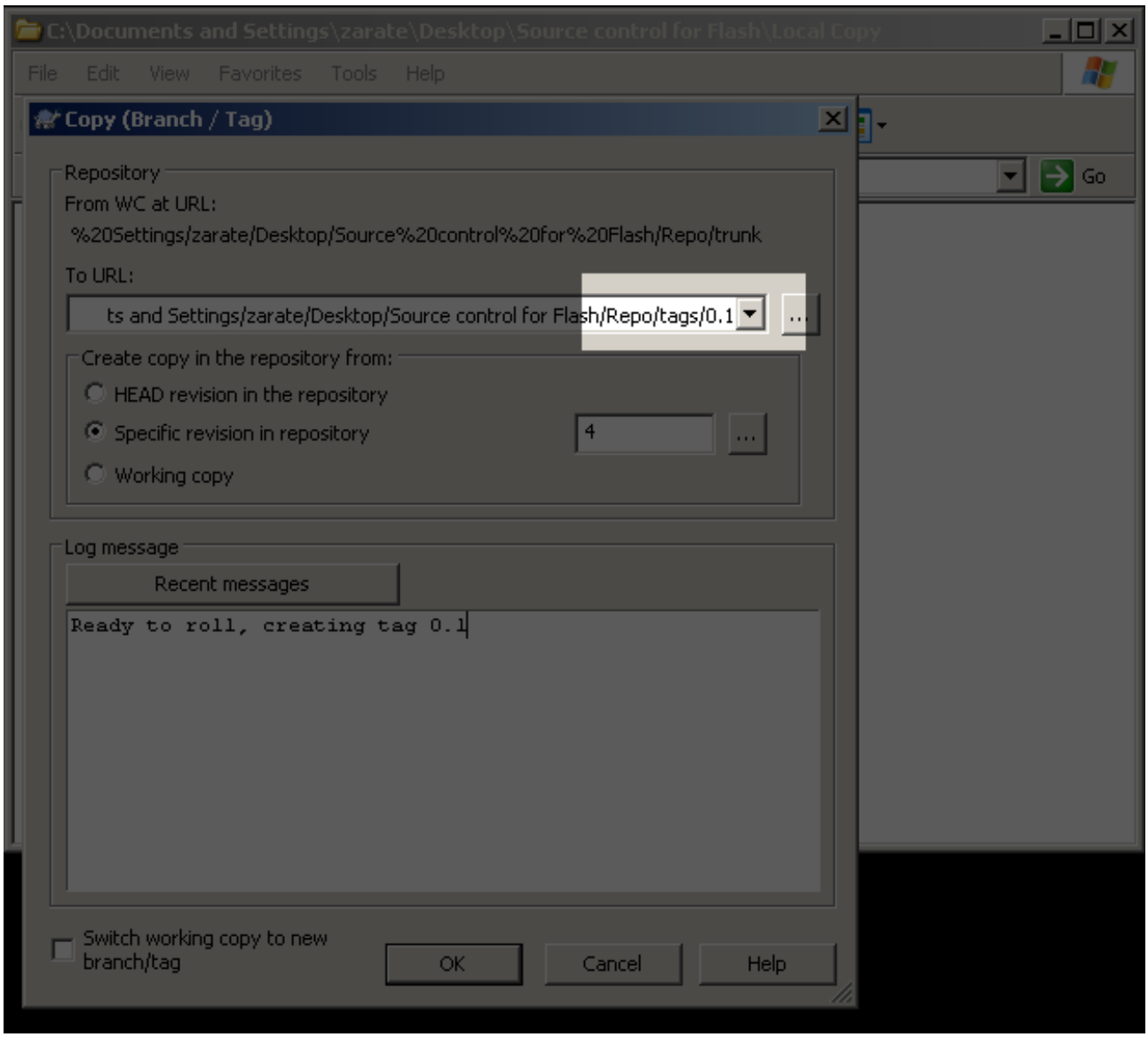
Tag

Tagging your repo is like taking a picture, a snapshot. Typically you tag your code when there is an important milestone or just when you want to create a returning point.

In Subversion tags (and branches) are just copies of the trunk in a specific moment. To create one, right click on the root of your local copy, then select Tortoise > Branch/Tag:

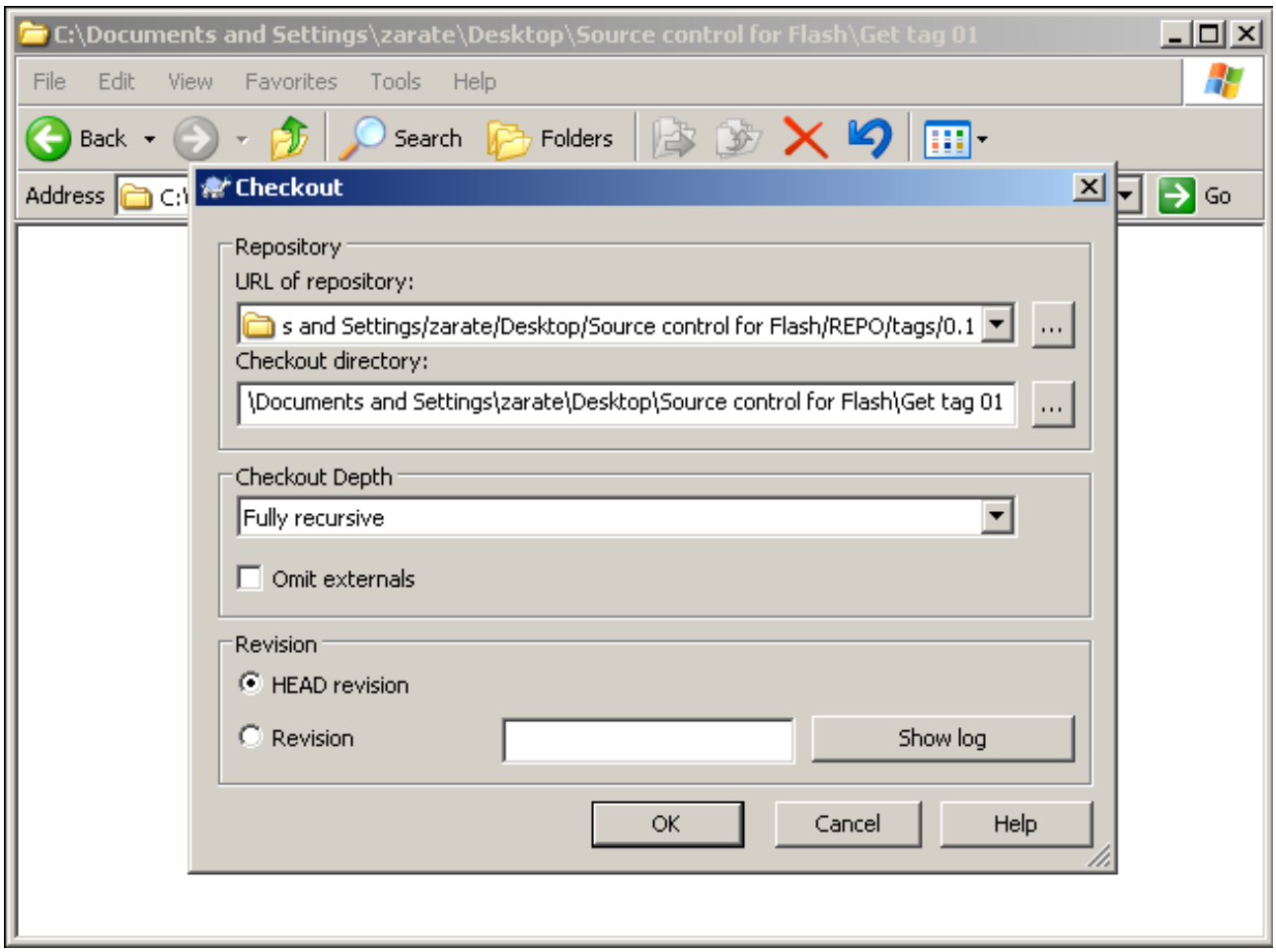


This is going to make a copy of your current local copy to wherever you tell it to. Let's go for creating a simple tag “0.1”:



Please note that you can select the “tag” folder using the “...” button or just tweak the “To URL” field by hand adding the name of the new tag at the end. Click ok and you are done.

Now to checkout a specific tag, just refer to it while checking out:



See? Instead of getting the trunk, we get an specific tag.

There's no “tag” command on SVN because tags and branches are just copies of the trunk. So, to create a tag use the copy command:

svn copy SOURCE_PATH DESTINATION_PATH

Something like this:

svn copy . REPO_PATH/tags/0.1

The “.” means “current directory”.

Tags are extremely useful as returning points and you shouldn't be afraid of using them. For example:

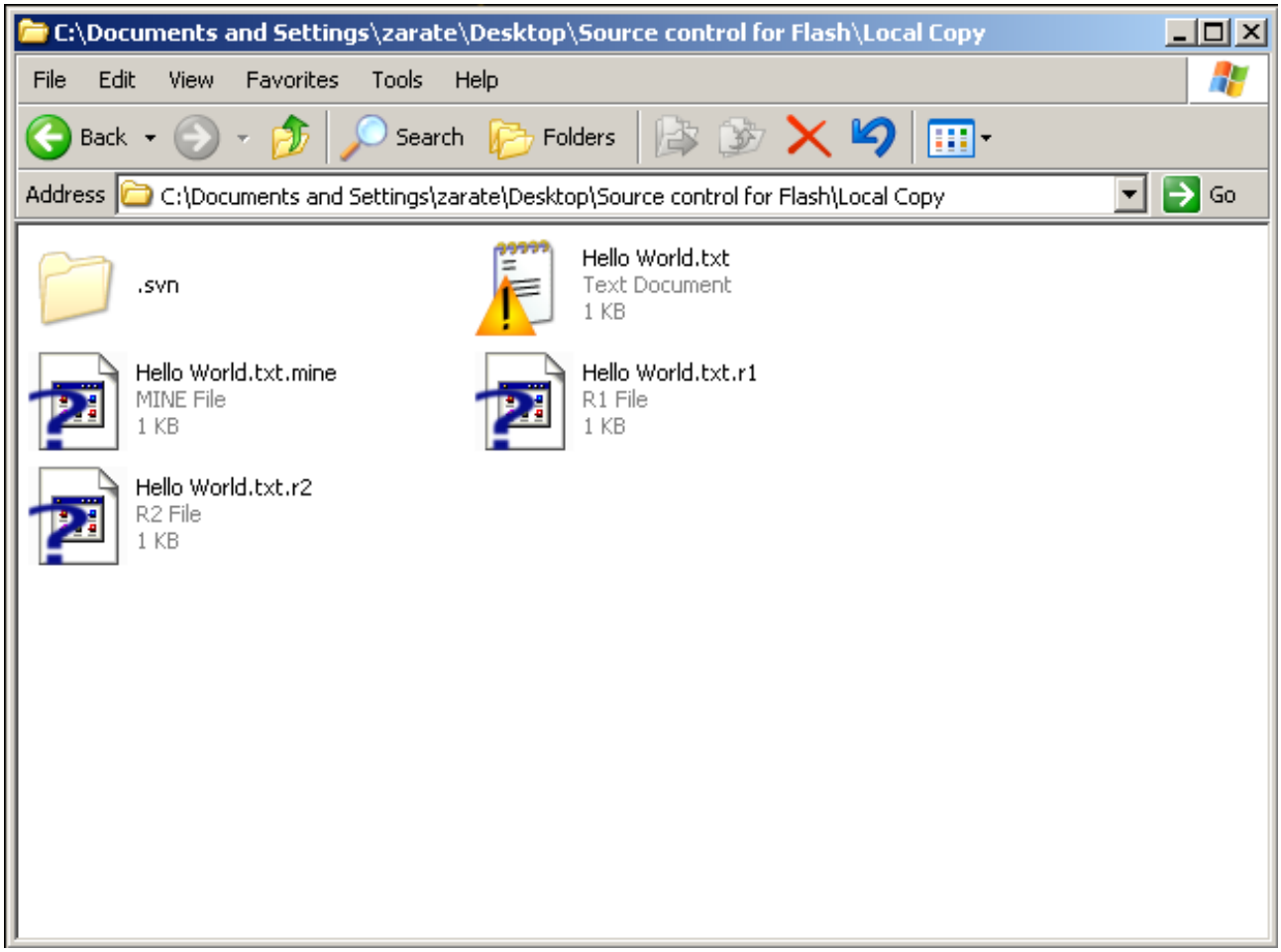
- Tag your repo when you send something to your client.
- Tag your repo when you make a release of your product.

Conflicts

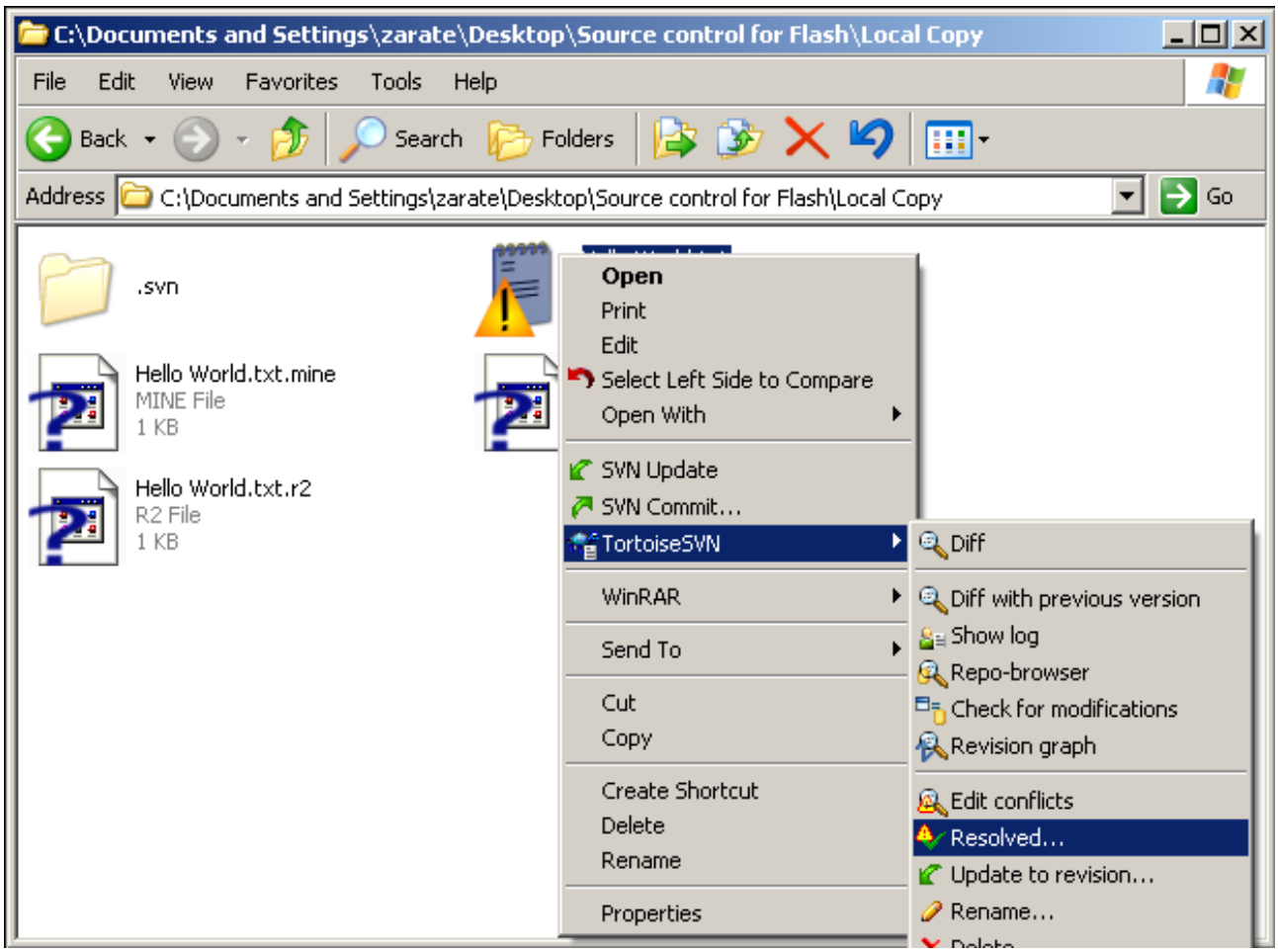
When updating SVN will get you any new files added. It will also update any existing files you

already have. To update existing files it needs to merge the repo and your local copy. If it finds no problems while merging then you are good to go. However, sometimes an automatic merge is not possible because there is conflict between the repo and your local copy. Most of the times these conflicts come because two people edited the same line with different information.

After an update with conflicts you will something like this:



Because SVN cannot resolve the conflict automatically you have to do it manually. You can either use Tortoise Merge (Tortoise > Edit conflicts) or do it by hand. You have to decide which change you keep, SVN cannot do it for you. Once done, resolve the conflict:



That would get rid of the additional files and you will be ready to commit again. This is the same as using the resolved command:

svn resolved FILE_PATH

Cool things to do with Subversion on a Sunday morning

This is a short list of really handy SVN topics, some easier than others, most of them around automatically link SVN with other development technologies. You can do all these things by hand but automating the process assures a) you don't forget to do it, b) you do it right and c) you save time.

(I might extend this section later either expanding these subjects or adding more, just don't have the time now!)

Painless WordPress updates

If you run a blog, you probably know that keeping you CMS updated is a real pain. WordPress guys know that, so they wrote a nice and easy tutorial about updating your WP using SVN:

http://codex.wordpress.org/Installing/Updating_WordPress_with_Subversion

Unless you are a developer willing to hack WP, choose Tracking Stable Versions.

Tweet your commits using SVN hooks

You know you can Tweet to the world every time you commit? I found this thanks to the vi.vu guys, thanks!

You need to use Subversion hooks (more info on the Links section) that are scripts automatically triggered by different events (such as committing, tagging, etc).

Using those hooks you can call a PHP script (or Java, or bat file, whatever is executable in the system hosting the repository) that connects to Tweeter and posts the tweet with the relevant information. The process looks like this:

geek a > commit > hook > script > tweet > other geeks get notified

Pretty sweet. You can find more info here:

http://remeseiro.com/metadata/index.php/2007/10/25/enviando-commits-de-subversion-a-twitter/
http://quarkblog.org/2007/02/06/subversion-hooks/

(Sorry for the Spanish links, haven't had time to find them in English, but will do. Until then, just Google “svn hooks” and you should be on track).

Add SVN revision information to your files

You can tell your SVN client to inject Subversion information (such as revision number, who did the

commit, etc.) to your files. This might be really useful to avoid firing up your client to find out such information, it will appear straight away in the code automatically.

See this:

http://svnbook.red-bean.com/en/1.0/ch07s02.html

Integrate SVN with bugtracking systems

Some SVN clients allow to link your repository and your bugtracking system automatically. Check this out:

http://blog.platinumsolutions.com/node/102

Links

Some of the links bellow cover advanced SVN and Source Control topics that you might not need now. My advice is: **read them anyway**.

These are just *some* of the links I've followed while researching for this tutorial, as I said at the beginning thanks to anybody that shares knowledge for free. You guys rock.

SVN clients

TortoiseSVN (Windows only): http://tortoisesvn.tigris.org
Rapid SVN (crossplatform): http://rapidsvn.tigris.org
SCPluing (Mac): http://scplugin.tigris.org
Cornerstone (Mac, commercial): http://www.zennaware.com/cornerstone
Syncro (crossplatform): http://www.syncrosvnclient.com
Versions (Mac, commercial): http://www.versionsapp.com

Documentation

Subversion official site:
http://subversion.tigris.org

Official Subversion book (might look big, but don't be afraid of consulting it):
http://svnbook.red-bean.com

Full list of SVN commands:
http://svnbook.red-bean.com/en/1.1/ch09.html

SVN hooks:
http://svnbook.red-bean.com/en/1.1/ch05s02.html

Adding SVN properties to your files:
http://svnbook.red-bean.com/en/1.0/ch07s02.html

Integrating SVN and Apache:
http://svnbook.red-bean.com/en/1.1/ch06s04.html

Wikipedia about Version Control Systems:
http://en.wikipedia.org/wiki/Source_control

Wikipedia about Subversion:
http://en.wikipedia.org/wiki/Subversion_(software)

Tutorials and other articles

SVN hooks tricks:
http://www.petefreitag.com/item/244.cfm
http://chicken.wiki.br/svn-post-commit-hooks

SVN integration with bugtracking systems (such as Bugzilla, Mantis, etc) :
http://blog.platinumsolutions.com/node/102
http://tortoisesvn.tigris.org/svn/tortoisesvn/trunk/doc/issuetrackers.txt (user: guest, password: empty)

Collaborate and connect with Subversion:
http://www.alistapart.com/articles/collaboratewithsubversion

List of SVN tutorials:
http://www.subversionary.org/taxonomy/term/14

Subversion best practices:
http://svn.collab.net/repos/svn/trunk/doc/user/svn-best-practices.html

5 SVN best practices:
http://www.iovene.com/5-svn-best-practices/

Command line Subversion for Windows (no Tortoise):
https://www.projects.dev2dev.bea.com/Subversion%20Clients/CommandLineSVN.html

Misc

Putty, free SSH client for Windows (use this to connect to your server via command line, might need to contact your ISP for details first):
http://www.chiark.greenend.org.uk/~sgtatham/putty/

List of companies offering free and commercial SVN hosting:
http://www.subversionary.org/hosting/hosting-services

And finally I need to ask Angus Blacksheep for permission to use his picture, but I don't have a stupid Yahoo! account to do it. Anyone can help here?

http://www.flickr.com/photos/43933977@N00/1455304619/