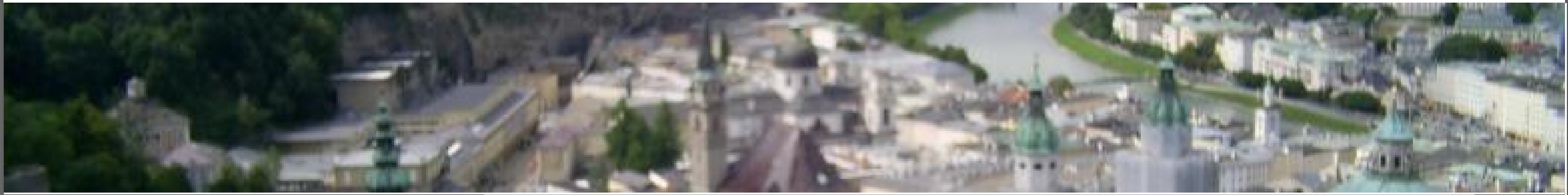


Guide to Versioning a Visual Studio Solution with Subversion, TortoiseSVN and AnkhSVN

★★★★½ 2 Votes

27
04
2008

1. Introduction
2. Resources
3. Prerequisites – Install Software
4. Setup a Subversion Repository with the Help of TortoiseSVN
    4.1. Create a Repository
    4.2. Set a Security Policy
    4.3. Run Subversion Server Process
5. Checkout a Working Copy
6. Add content and Commit
    6.1. Add a Text File to Versioning
    6.2. Add a Visual Studio Solution to Versioning (TortoiseSVN)
    6.3. Add a Visual Studio Solution to Versioning (AnkhSVN)

RECENT POSTS

ASP.NET MVC: Panel HtmlHelper extension methods with using syntax
CruiseControl.Net Tutorial – Part 2
CruiseControl.Net Tutorial – Part 1
Line Continuation Character
Guide to Versioning a Visual Studio Solution with Subversion, TortoiseSVN and AnkhSVN

ARCHIVES

November 2010 (1)
June 2008 (2)
May 2008 (1)
April 2008 (1)
March 2008 (1)
February 2008 (3)
January 2008 (3)

7. Install Subversion as a Service

1. Introduction

In this article I will describe the process of setting up a working Subversion server, creating a repository and add to versioning a Visual Studio solution to work with.

2. Resources

During this article I will guide you through the installation process of the following software:

Subversion
TortoiseSVN
AnkhSVN

→ top of post
→ top of paragraph

3. Prerequisites – Install Software

- Install Subversion
- Install TortoiseSVN
- Install AnkhSVN

Install Subversion

Download the latest Subversion release from the download page and launch the installer. At the moment I'm writing the last available version is: svn-1.4.6-setup.exe.
Subversion will be installed under: C:\Program Files\Subversion.

Make sure that C:\Program Files\Subversion\bin is included in the %PATH% environment variable, otherwise add it following these steps:

SEARCH

TOPICS

.NET AnkhSVN ASP.NET MVC C# Command line CruiseControl.NET Extension Methods Music Project Management RhinoMocks Scripting Subversion TortoiseSVN Tutorials Visual Studio 2005 Wordpress

VISITORS

HITS

CURRENTLY ONLINE

right-click MyComputer and choose **Properties** –> **Advanced**, then click the button: **Environment Variables**.
Look for the name: *Path* between variable names (left column) in the bottom list.
Once found double click it, add a semi-colon to the '*variable value*' field and paste the path to Subversion bin directory (C:\Program Files\Subversion\bin).
Click ok three times to accept the changes.

Now open a command prompt and digit **svn help**. You should be able to see svn syntax information.
The installation succeeded. Now go on to the next step: Install TortoiseSVN

→ top of post
→ top of paragraph

Install TortoiseSVN

Download the latest TortoiseSVN release from the download page: TortoiseSVN-1.4.8.12137-win32-svn-1.4.6.msi and launch the installer.

→ top of post
→ top of paragraph

Install AnkhSVN

Download the latest AnkhSVN release: AnkhSetup-1.0.2.2778-Final.msi from the download page and launch the installer.

Now that you have installed it, forget it for a while and step through the next paragraphs.

→ top of post
→ top of paragraph

4. Setup a Subversion Repository with the Help of TortoiseSVN

4.1. Create a Repository
4.2. Setup a Security Policy

4.3. Run Subversion Server Process

Now you have a very cool GUI for Subversion provided by TortoiseSVN.
In the following paragraphs we will setup a Subversion repository on the local machine and we will start working with it.

4.1. Create a Repository

Create a new directory, e.g.: C:\develop\test\repo, right click it and choose: **TortoiseSVN** –> **Create repository here…** (see: Figure 1).
Accept the preselected option: *Native filesystem (FSFS)* at the prompt and click ok (see: Figure 2).
The new repository will be created and an informational message will tell you that '*The Repository was successfully created*'.

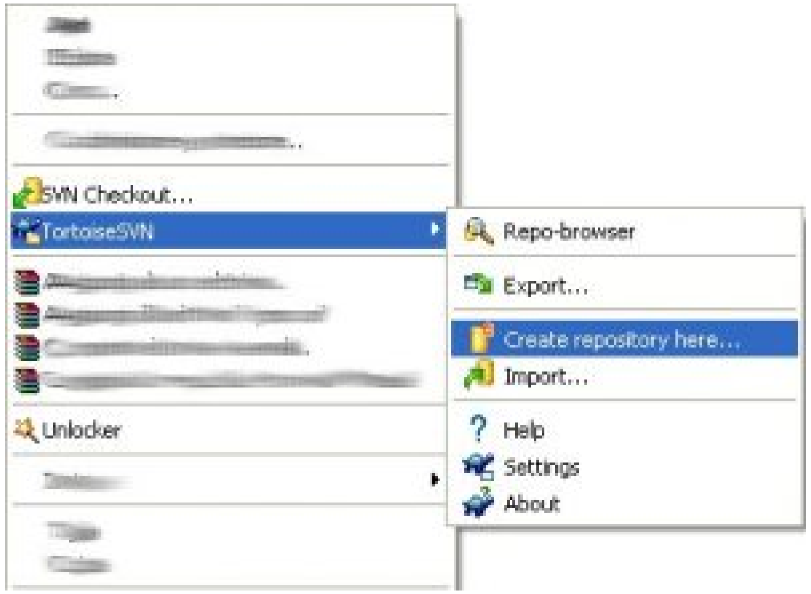


Figure 1. Context menu for the 'repo' directory

FRIENDS

David Keaveny

SOFTWARE

CruiseControl.NET
Gimp
MediaWiki
Nunit
Rhino Mocks
Subversion
TortoiseSVN

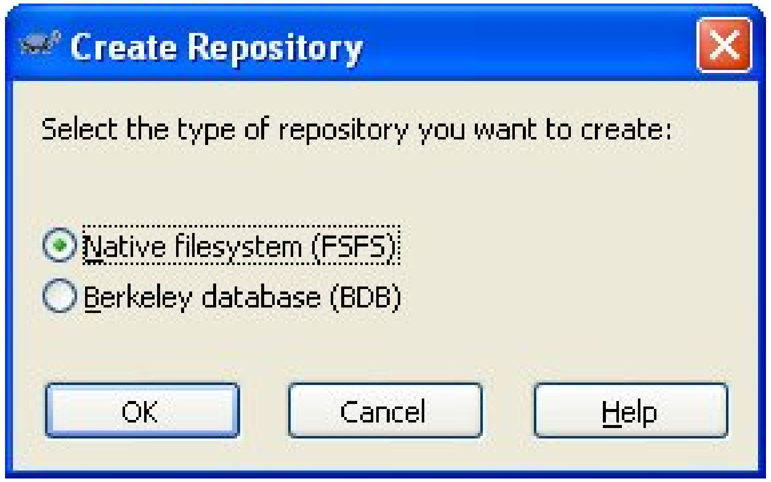
ADMIN

Register
Log in
Entries RSS
Comments RSS
WordPress.com

FEEDS

Subscribe in a reader

NOW READING



Select the type of repository you want to create:

[✓ Native filesystem (FSFS)]
Berkeley database (BDB)

Figure 2. Filesystem choice for the repository

Now let's add some content. First of all prepare the standard layout: create a temporary directory next to the *repo* directory and call it *temp*: C:\develop\test\temp. Add three subdirectories inside it, named: **trunk**, **tags**, **branches**.
The **trunk** directory will become the main directory of our project and it will contain all the versioned data.
Right click on C:\develop\test\temp and choose: **TortoiseSVN –> Import…** and the Import window will open. Click the button on the right of the combobox: '*URL of repository*' (as shown in Figure 3).

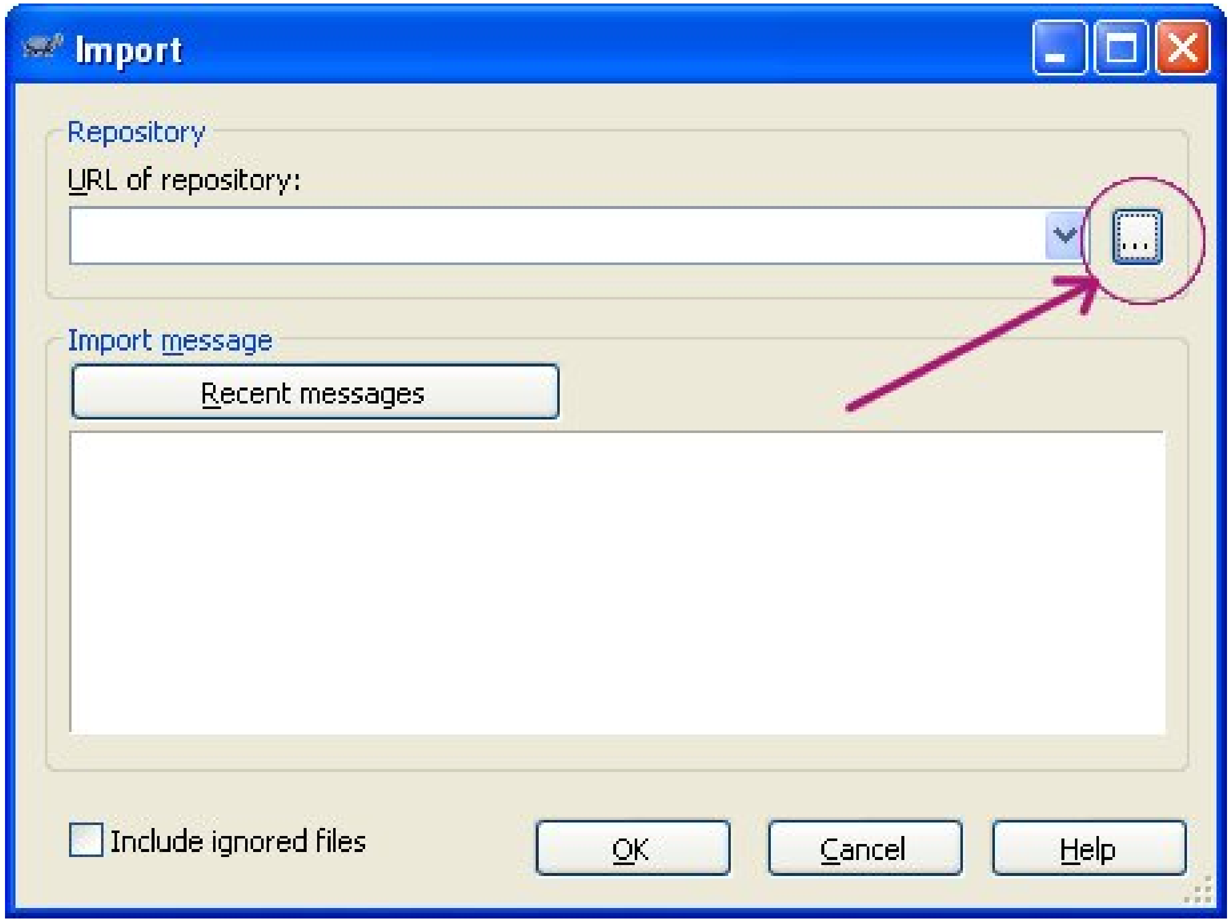
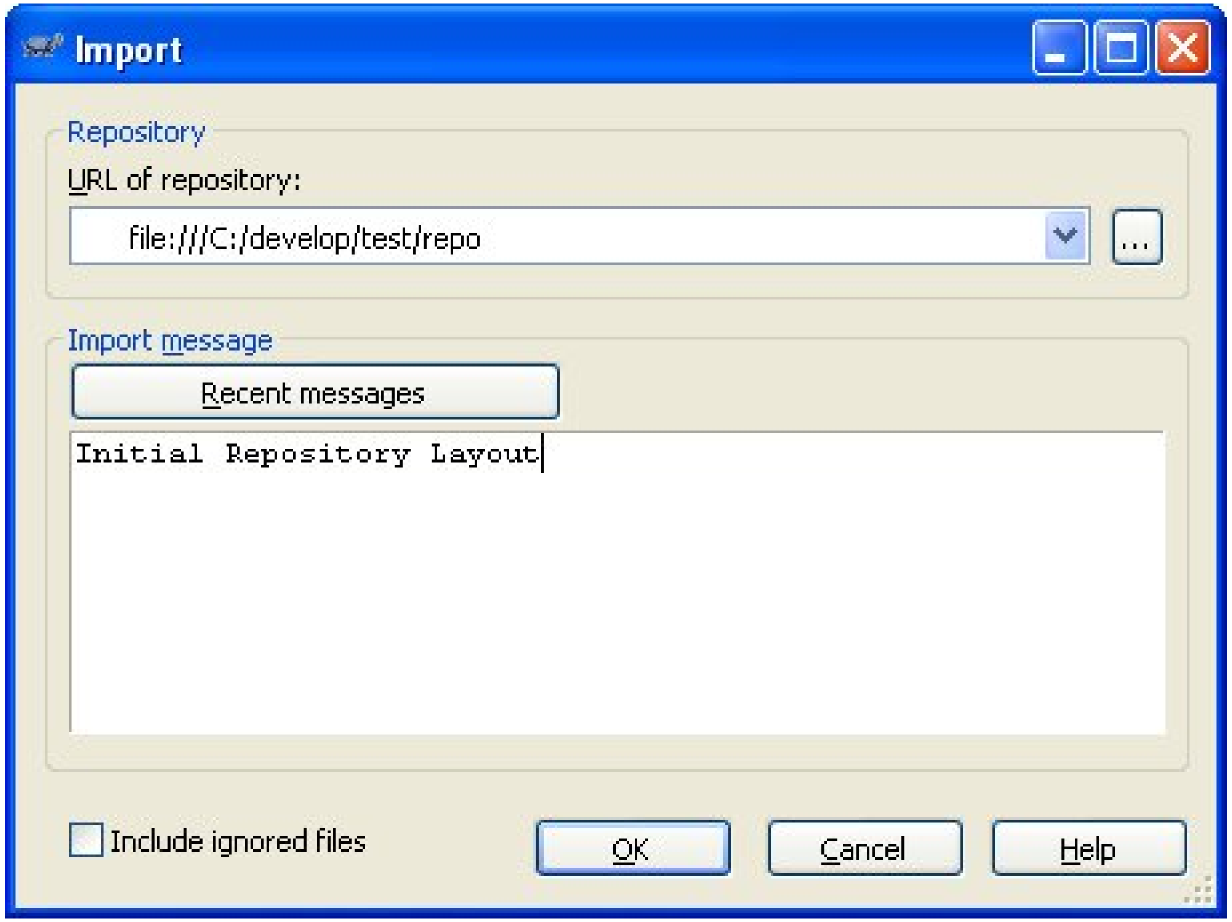


Figure 3. Import window.

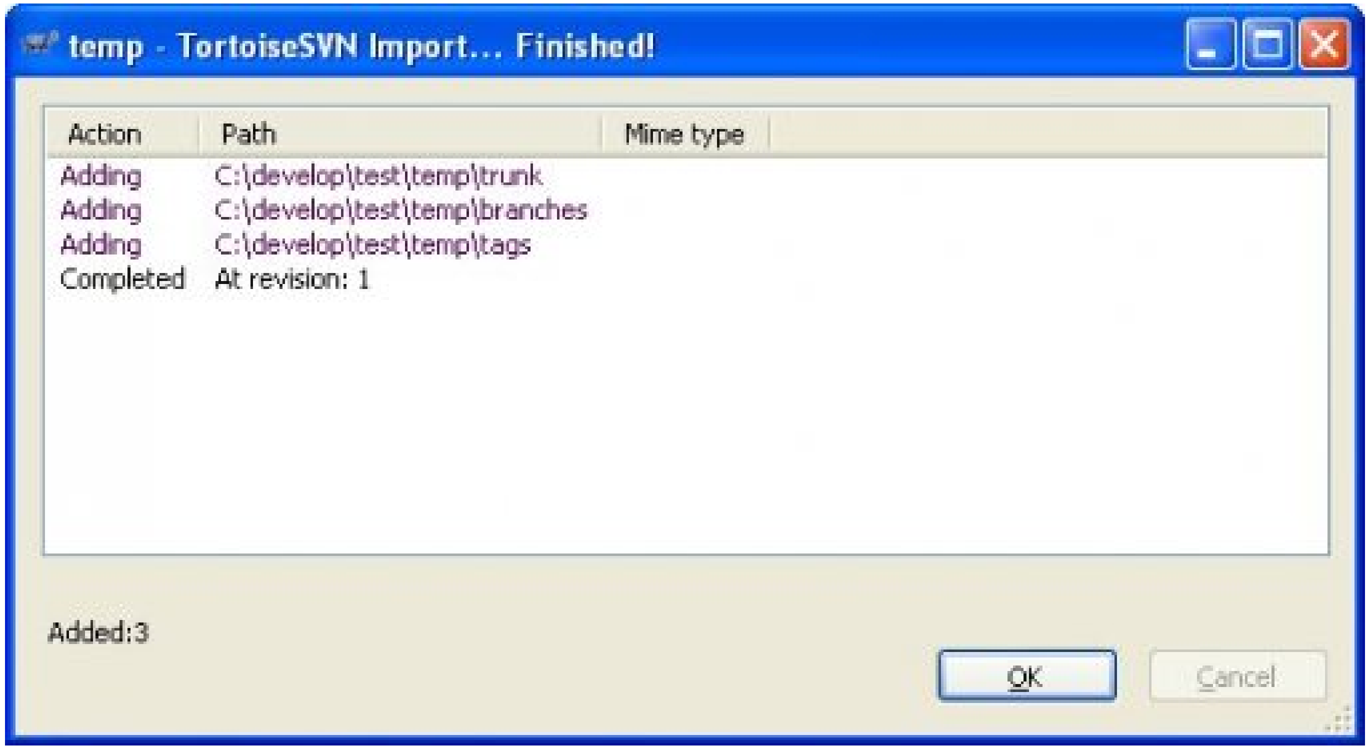
A **browse** pop up window will open. Browse to the repository directory: C:\develop\test\repo and click **OK**.
The combobox will be filled with: file:///C:/develop/Test/repo.
Write the comment '*Initial Repository Layout*' in the '*Import Message*' text area at the bottom of the Import window and click **OK** (see: Figure 4).
Everything should be alright and the informational window shown in Figure 5 should appear.
You can click **OK** and the repository layout is ready.



URL of repository: file:///C:/develop/test/repo

Import message: Initial Repository Layout

Figure 4. Import window with with repository path set.


temp - TortoiseSVN Import... Finished!

Action	Path	Mime type
Adding	C:\develop\test\temp\trunk	
Adding	C:\develop\test\temp\branches	
Adding	C:\develop\test\temp\tags	
Completed	At revision: 1	

Added:3

Figure 5. Import operation succeeded.

Now you can delete the temporary directory: C:\develop\test\temp.

→ top of post
→ top of paragraph

4.2. Setup a Security Policy

Before starting to use your freshly created repository you must set a security policy, that is, you must create user accounts for the repository.
First of all open the file: C:\develop\test\repo\conf\svnserve.conf.
Have a read to the file and notice that all lines are commented (the pound character: # is the comment character).
Uncomment rows 12 and 13, remove any leading spaces at the beginning of the rows and change them to fit your security policy.
You can, for instance, decide to deny access to the repository to anonymous users and to unauthorized ones while giving full access to authorized users.
To obtain this result set those two lines as follows:

```
anon-access = none
auth-access = write
```

Then uncomment row 18 to tell Subversion where the user accounts information will be stored:

```
password-db = passwd
```

where ‘passwd‘ is the name of a file in the same directory as ‘svnserve.conf‘ (C:\develop\test\repo\conf\passwd).
Open the ‘passwd‘ file and you will find two example rows, at rows 7 and 8, showing the syntax to define users:

```
# harry = harryssecret
# sally = sallyssecret
```

The two lines are commented but they clearly show that the syntax is:

```
user = password
```

Add information about the account that you will use to access the repository (e.g.: test = test).

→ top of post
→ top of paragraph

4.3. Run Subversion Server Process

You can simply start svn server process from the command line.
Open a command prompt and digit:

```
svnserve -d -r “C:\develop\test\repo”
```

and you’re done!
If you prefer to run Subversion in order to let it manage different repositories placed under a common directory, you can run the Subversion server process in that directory, e.g.:

```
svnserve -d -r “C:\develop”.
```

The row above will let Subversion be able to expose repositories placed in the paths:

```
C:\develop\TestRepo1
```

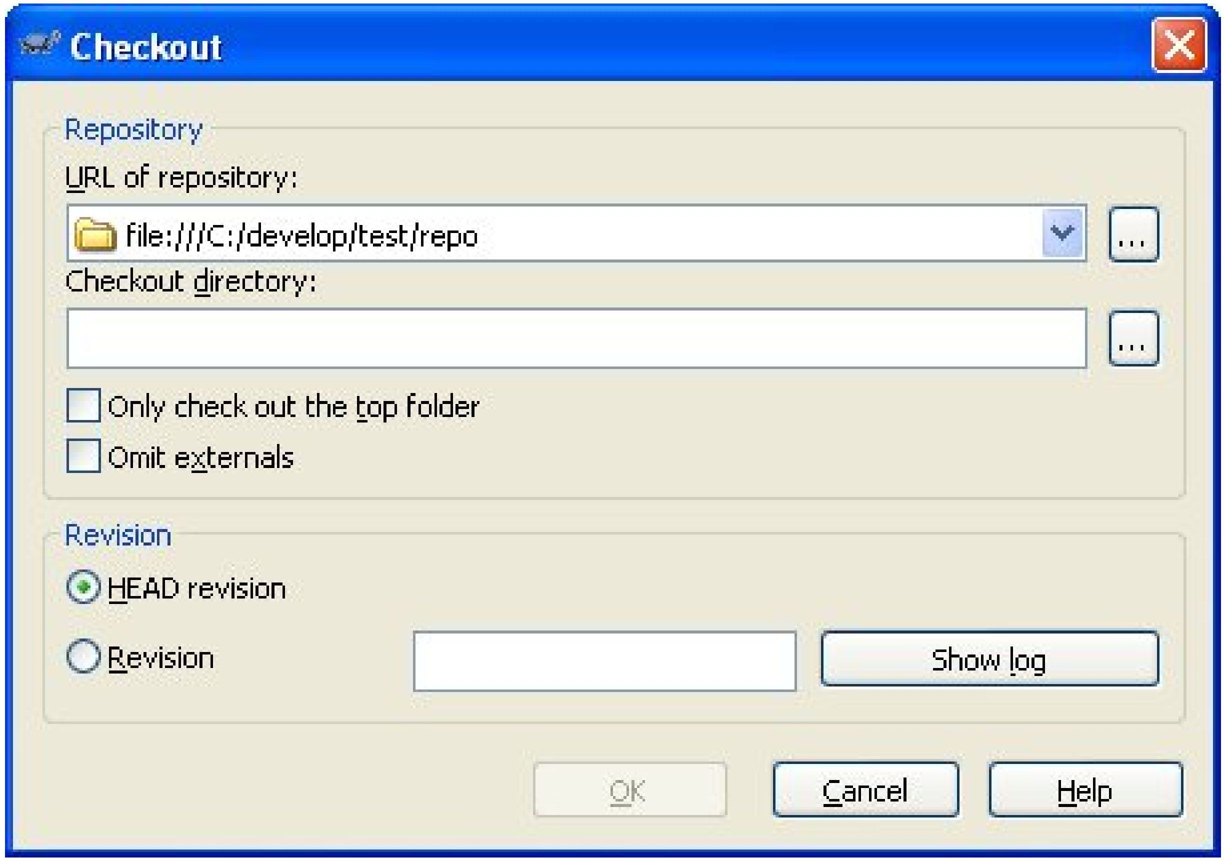
C:\develop\TestRepo2
…

If you prefer to run Subversion as a Windows service, have a look at the paragraph: Install Subversion as a Service.

→ top of post
→ top of paragraph

5. Checkout a Working Copy

Now you can prepare your workin copy or, in Subversion language, checkout your working copy.
Create a new directory named ‘wk’: C:\develop\test\wk.
Right-click the directory: C:\develop\test\repo and choose: **SVN Checkout…**. The Checkout panel will open.
Notice that the box: ‘*URL of repository*‘ is already filled with the path: file:///C:/develop/test/repo (see: Figure 6).


Checkout

Repository

URL of repository: file:///C:/develop/test/repo

Checkout directory:

[ ] Only check out the top folder

[ ] Omit externals

Revision

(●) HEAD revision

( ) Revision

Figure 6. Checkout window.

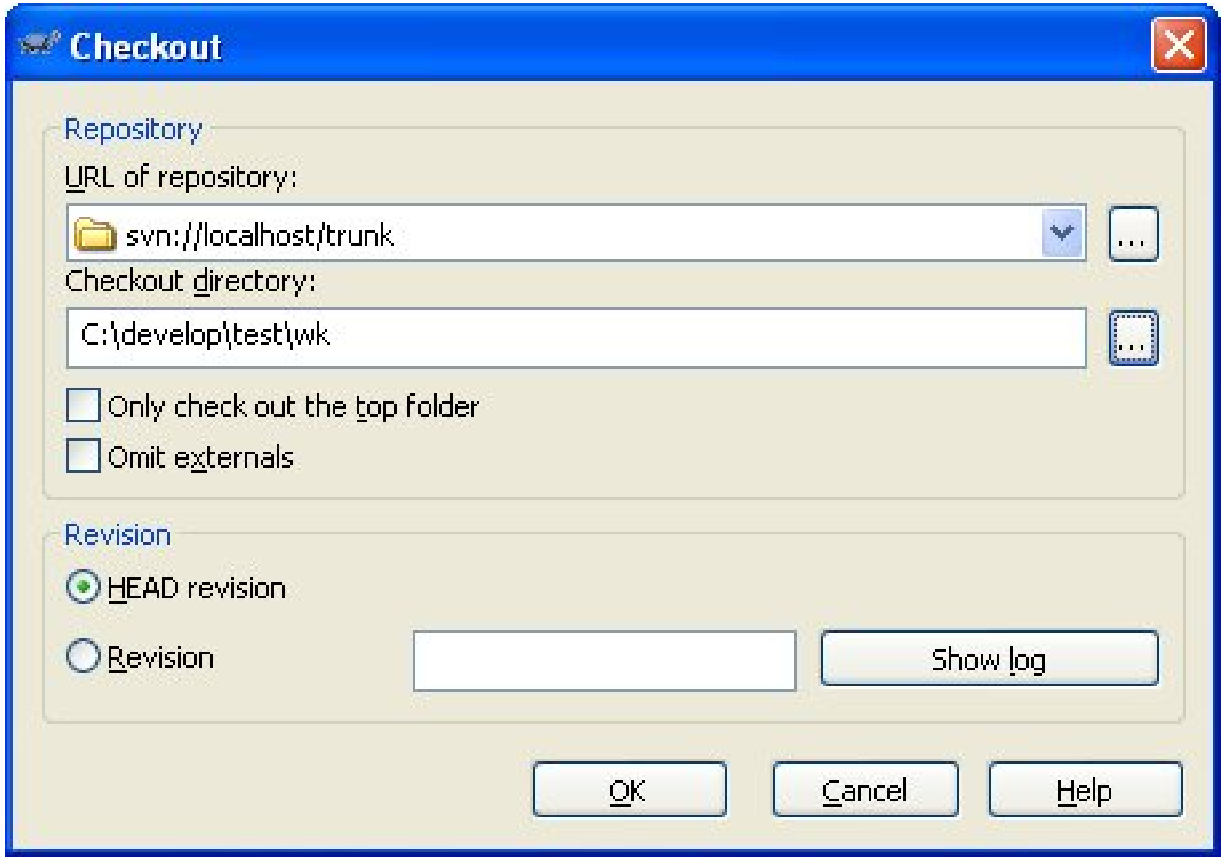
You need to modify that path to match the exact path to the repository and you can do it in one of two ways:

- Just add '*trunk*' to the path, because we want that only the content under that directory is copied in our working copy.
  The '*URL of repository*' will then be: file:///C:/develop/test/repo/trunk.
  This choice doesn't need to have svnserve running and it's good for a local test environment.

- If you want to use Subversion through your network (i.e. in a real environment) and if you want to take advantage of the Subversion security policy that we have set up in the previous paragraph, you'd better use this alternative option:
  Let's assume that you have run svnserve with the following row: svnserve -d -r "C:\develop\test\repo" as explained in paragraph 4.3.
  Delete the content of the '*URL of repository*' box and replace it with: '**svn://localhost/trunk**'.

The example is provided assuming Subversion server installed on the local machine but you can replace *localhost* with the server machine name or IP address.
We need to specify the path of the repository relative to the working directory of Subversion (svnserve), which is: C:\develop\test\repo, so the path that we specify is simply: trunk.

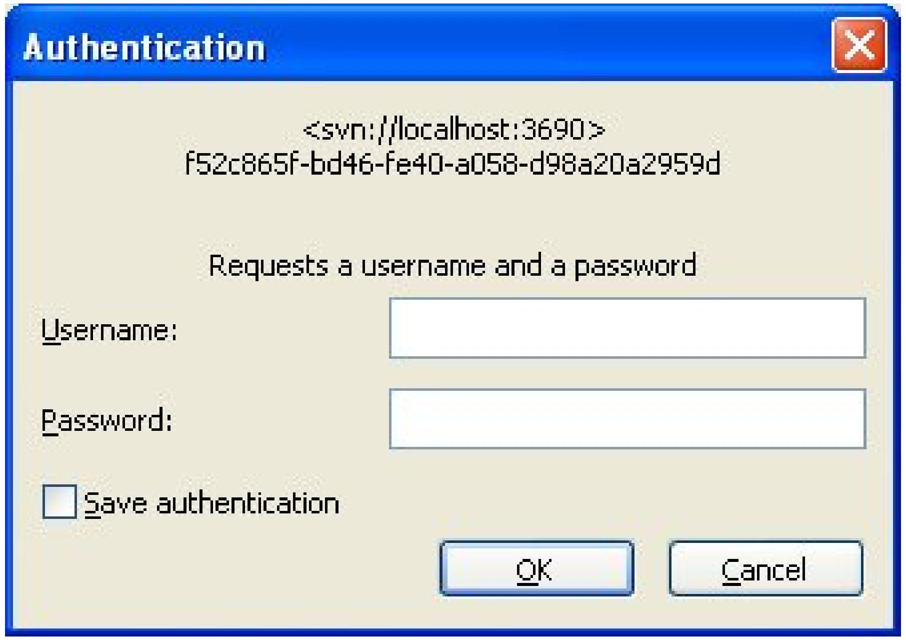
In the rest of the article we will assume the second choice.
After having set the '*URL of repository*' we must fill the '*Checkout directory*' box and we can do it browsing to C:\develop\wk by means of the adjacent button (see: Figure 7).
Make sure that '*HEAD revision*' is checked and click **OK**.



URL of repository: svn://localhost/trunk
Checkout directory: C:\develop\test\wk
HEAD revision [✓]

Figure 7. Checkout window filled.

If you have set the '*URL of repository*' to svn://localhost/trunk you will be prompted for username and password (see Figure 8).

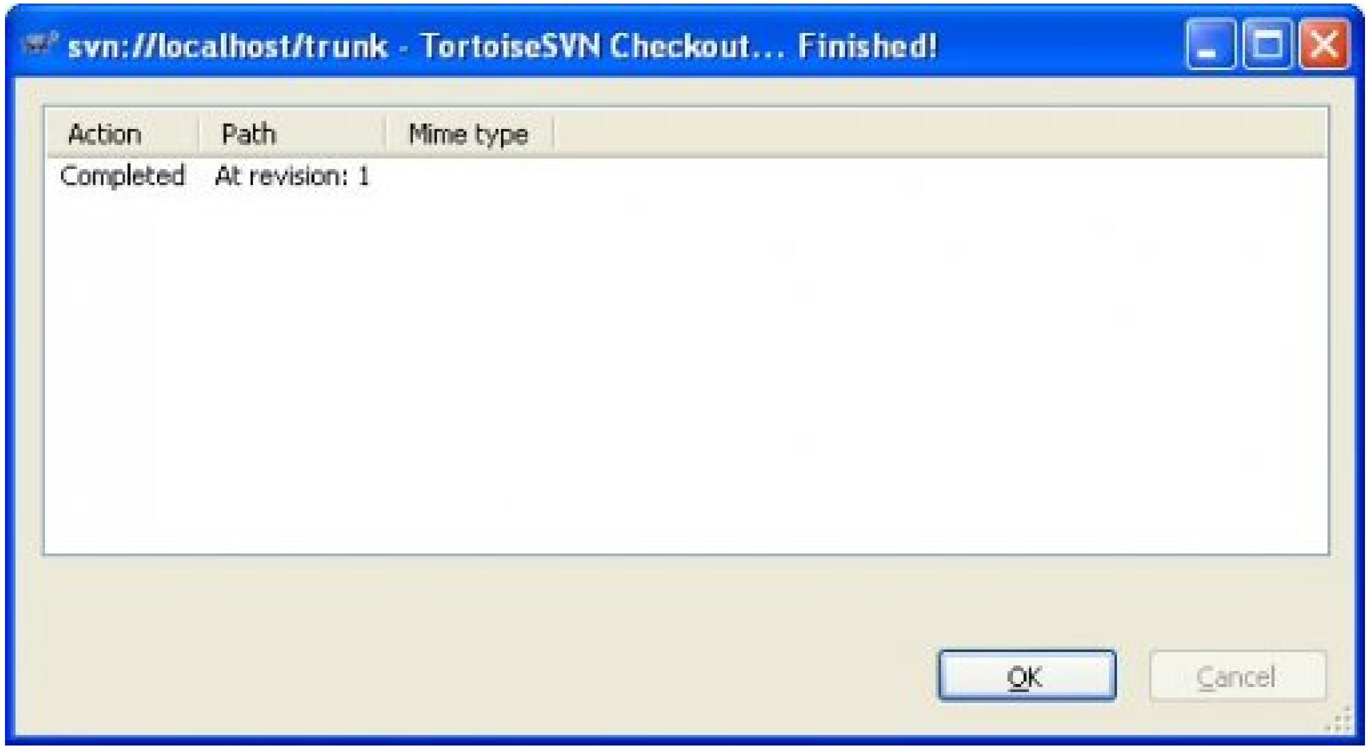

Authentication

<svn://localhost:3690>
f52c865f-bd46-fe40-a058-d98a20a2959d

Requests a username and a password

Figure 8. Prompt for username and password.

Fill both fields with: ‘*test*‘ or whatever account information you configured in the paragraph: 4.2. Set a Security Policy and flag the ‘*Save authentication*‘ checkbox, then click **OK**.
An informational window will appear stating that the checkout was succesfully created (see: Figure 9).



Action	Path	Mime type
Completed	At revision: 1	

Figure 9. Checkout completed.

Close that window and look inside the C:\develop\wk directory.
The **wk** directory has been filled with the content from the repository's trunk directory, that is the directory is empty.
Actually it's not quite empty: there's a directory named **.svn** inside it.
Such directory contains all the information needed by Subversion to keep track of the history of all the files that will be added to versioning inside this directory.
Now let's go to the following step and start adding content.

→ top of post
→ top of paragraph

6. Add content and Commit

6.1. Add a Text File to Versioning
6.2. Add a Visual Studio Solution to Versioning (TortoiseSVN)
6.3. Add a Visual Studio Solution to Versioning (AnkhSVN)

6.1. Add a Text File to Versioning

Let's add a simple text file to versioning: create a file named '*test.txt*' inside C:\develop\wk.
Right-click it and choose: **TortoiseSVN –> Add…** (see: Figure 10), then click **OK** at the following two prompts and you will see an overlay blue plus icon above your file meaning that the file has been scheduled for inclusion in the repository upon the next commit.

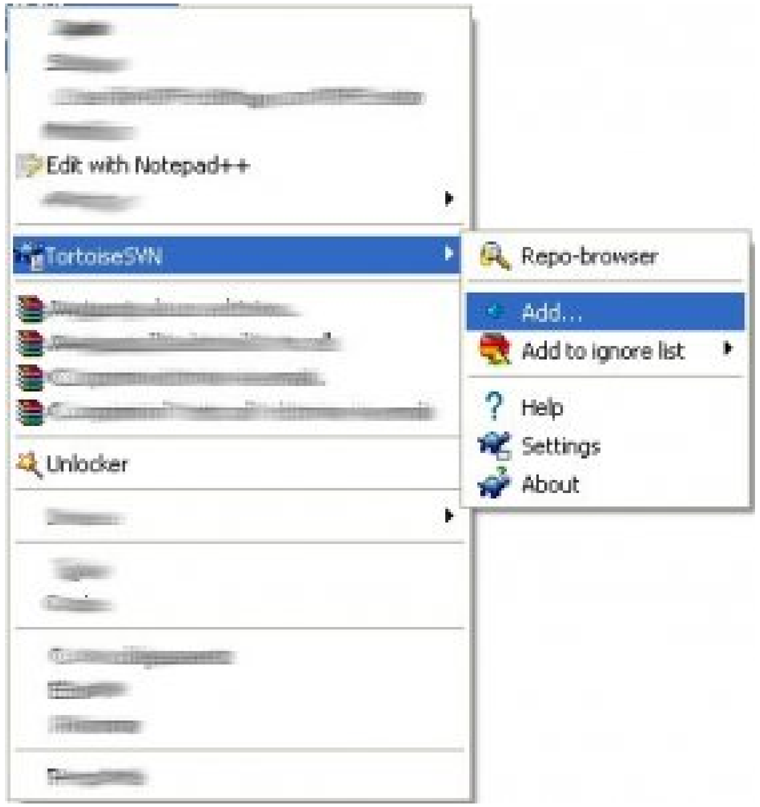


Figure 10. Add file to Versioning.

If you can't see the overlay icon, just press F5 to refresh the screen.
Now execute your first commit thus sending the new file to the repository. From now on, its changes will be tracked by the Subversion versioning system.
If you want to commit the content of the directory all at once (only one file at the moment) just go up one level, right click on the directory: C:\develop\wk and choose: **SVN –> Commit…** (see: Figure 11).
If you checked the '*Save authentication*' checkbox when checking out the working copy (as

shown in 5. Checkout a Working Copy) you won’t be prompted, now, for username and password.

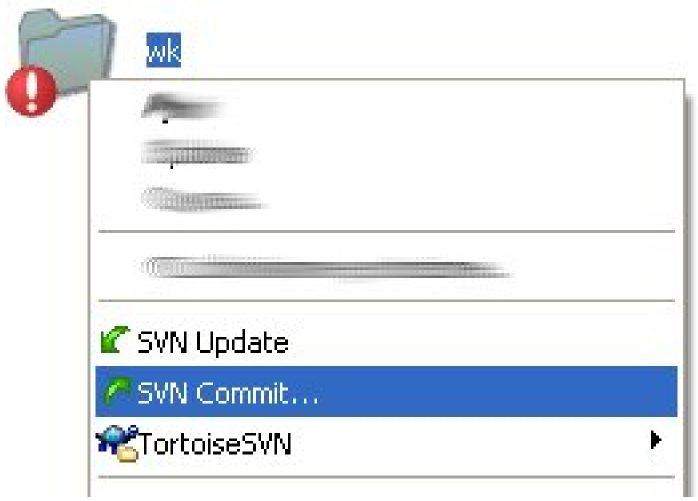
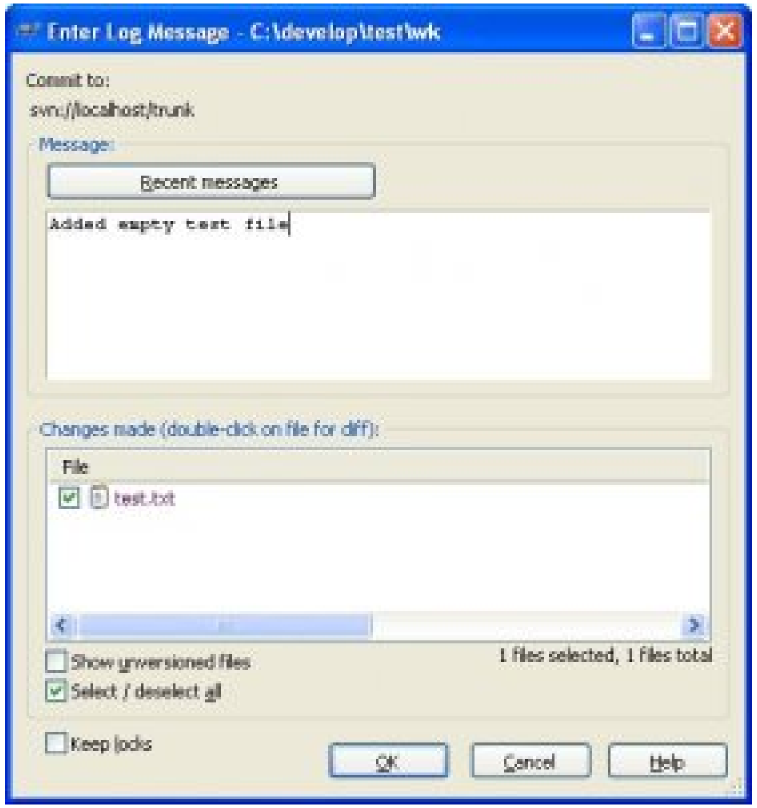


Figure 11. Commit working copy content with TortoiseSVN.

The Log Message window will open (see: Figure 12), waiting for you to provide a meaningful message. Write: “Added empty test file” and click **OK**.


Enter Log Message - C:\develop\test\wk

Commit to:
svn://localhost/trunk

Message:

```
Added empty test file
```

Changes made (double-click on file for diff):

File
✓ test.txt

1 files selected, 1 files total

Show unversioned files
✓ Select / deselect all
Keep locks

Figure 12. Provide a meaningful message before committing.

Make sure that everything worked fine when the informational window titled: '**wk – TortoiseSVN Commit… Finished!**' will appear.
It should state something like: '*Completed At revision: 1*'.
If everything is alright then click **OK**.

Now you can start adding content to the file and commit the changes to the repository.

→ top of post
→ top of paragraph

6.2. Add a Visual Studio Solution to Versioning (TortoiseSVN)

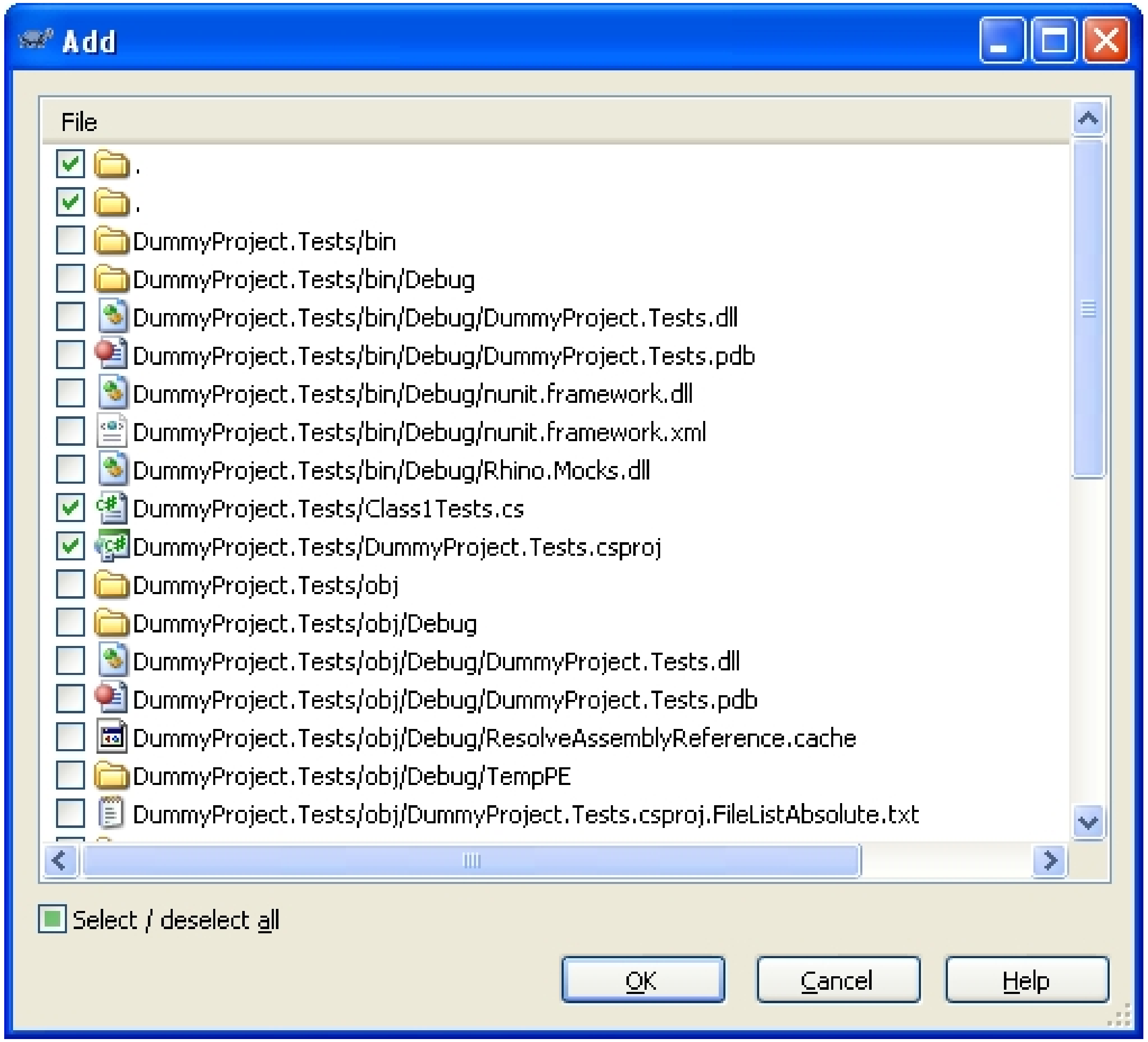
Let's talk about versioning a Visual Studio solution (I won't talk here about how to create Visual Studio solutions and projects).
Open Visual Studio (I'm working with Visual Studio 2005) and create a new blank solution inside C:\develop\wk.

Name the solution: **DummySolution.sln** and add two projects to the solution: **DummyProject.csproj** and **DummyProject.Tests.csproj**.
The first is a Class Library project and the second is a **Nunit** Tests project referencing **DummyProject**.
If you're using Nunit and RhinoMocks you can take advantage of the project Template I provide with the TDDTemplates.vsi (see article: Visual Studio Nunit & Rhino Mocks Templates) installer to easily create the unit tests project.
Now you can add the solution to versioning in one of two ways:

1. adding the solution file and the projects directories with TortoiseSVN;
2. installing AnkhSVN and versioning the solution directly from Visual Studio;

In this paragraph we'll see the first option while in the next paragraph we will repeat the same actions with AnkhSVN.

At first select the solution file: *DummySolution.sln* and the two project directories: *DummyProject* and *DummyProject.Tests*.
Be careful not to select the file: *DummySolution.suo*, which contains solution-wide configuration data specific to every developer's machine.
Then right-click one of the selected items and choose: **TortoiseSVN –> Add….**.
In the **Add** window (see: Figure 13) uncheck the directories: **bin** and **obj** of each project and the files: *DummyProject.csproj.user* and *DummyProject.Tests.csproj.user* if you find any of them in the list of checked items.
Click **OK** and a report window, titled: **TortoiseSVN Add… Finished!**, should appear if everything worked well. Click **OK** again as you did for the 'test.txt' file.
Now all the content needed is scheduled to be added to versioning upon the next commit action.



- [✓] .
- [✓] .
- [ ] DummyProject.Tests/bin
- [ ] DummyProject.Tests/bin/Debug
- [ ] DummyProject.Tests/bin/Debug/DummyProject.Tests.dll
- [ ] DummyProject.Tests/bin/Debug/DummyProject.Tests.pdb
- [ ] DummyProject.Tests/bin/Debug/nunit.framework.dll
- [ ] DummyProject.Tests/bin/Debug/nunit.framework.xml
- [ ] DummyProject.Tests/bin/Debug/Rhino.Mocks.dll
- [✓] DummyProject.Tests/Class1Tests.cs
- [✓] DummyProject.Tests/DummyProject.Tests.csproj
- [ ] DummyProject.Tests/obj
- [ ] DummyProject.Tests/obj/Debug
- [ ] DummyProject.Tests/obj/Debug/DummyProject.Tests.dll
- [ ] DummyProject.Tests/obj/Debug/DummyProject.Tests.pdb
- [ ] DummyProject.Tests/obj/Debug/ResolveAssemblyReference.cache
- [ ] DummyProject.Tests/obj/Debug/TempPE
- [ ] DummyProject.Tests/obj/DummyProject.Tests.csproj.FileListAbsolute.txt

Figure 13. Add window.

Now you can commit the freshly added content to the repository: right click on the directory C:\develop\wk and choose: **SVN –> Commit…** as you did when you added the *test.txt* file.

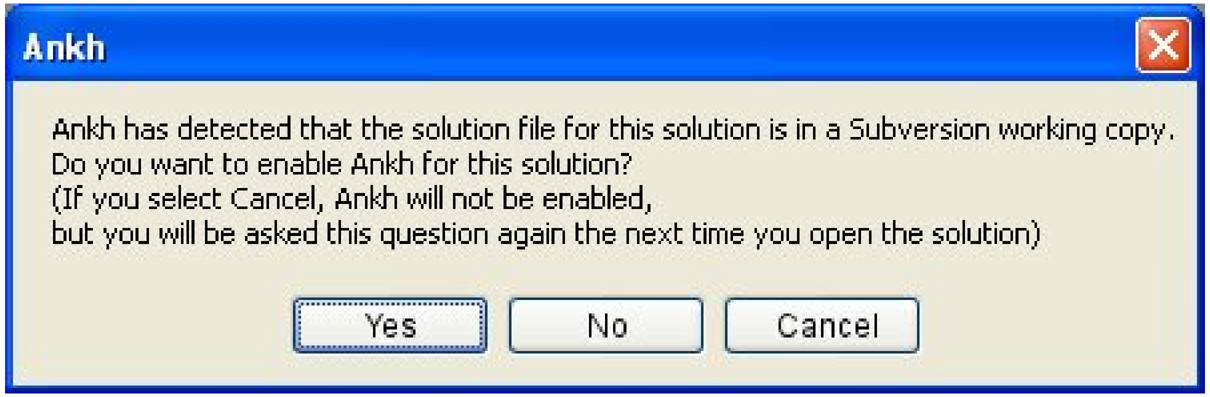
→ top of post
→ top of paragraph

6.3. Add a Visual Studio Solution to Versioning (AnkhSVN)

As an alternative to previous paragraph, we can add the Visual Studio solution to versioning directly from the Visual Studio GUI with the help of AnkhSVN.

Before reading the current paragraph you need to install AnkhSVN as in Install AnkhSVN.

Open the solution file: *DummySolution.sln* with Visual Studio 2005.
A little prompt window (see: Figure 14) will ask if you want to enable AnkhSVN for the current solution:

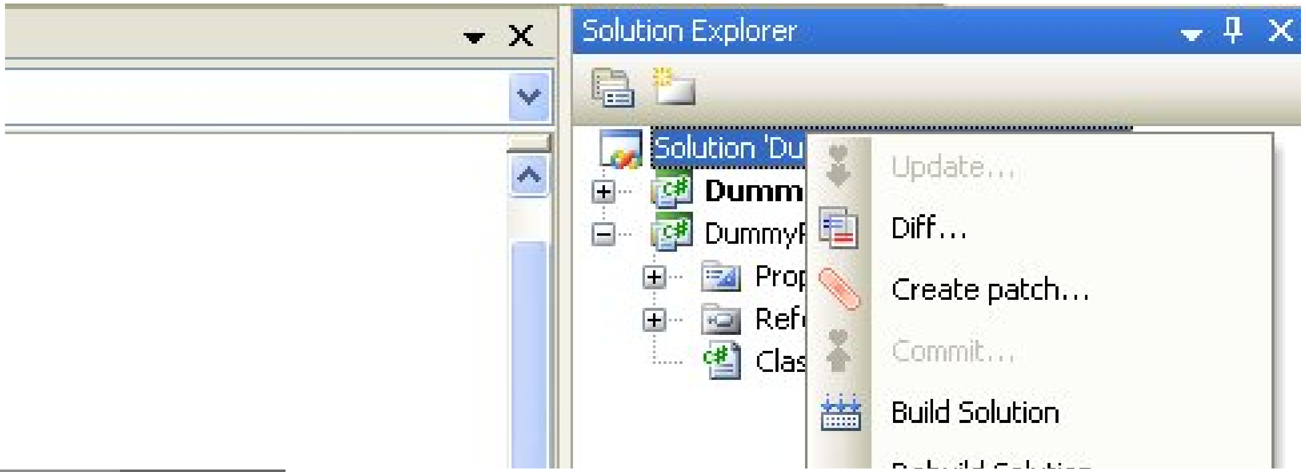


Ankh

Ankh has detected that the solution file for this solution is in a Subversion working copy.
Do you want to enable Ankh for this solution?
(If you select Cancel, Ankh will not be enabled,
but you will be asked this question again the next time you open the solution)

Figure 14. Enable AnkhSVN for current solution.

Click **No** and go to solution explorer. Right-click the solution item and choose:
Ankh –> Add solution to Subversion repository... as shown in Figure 15:



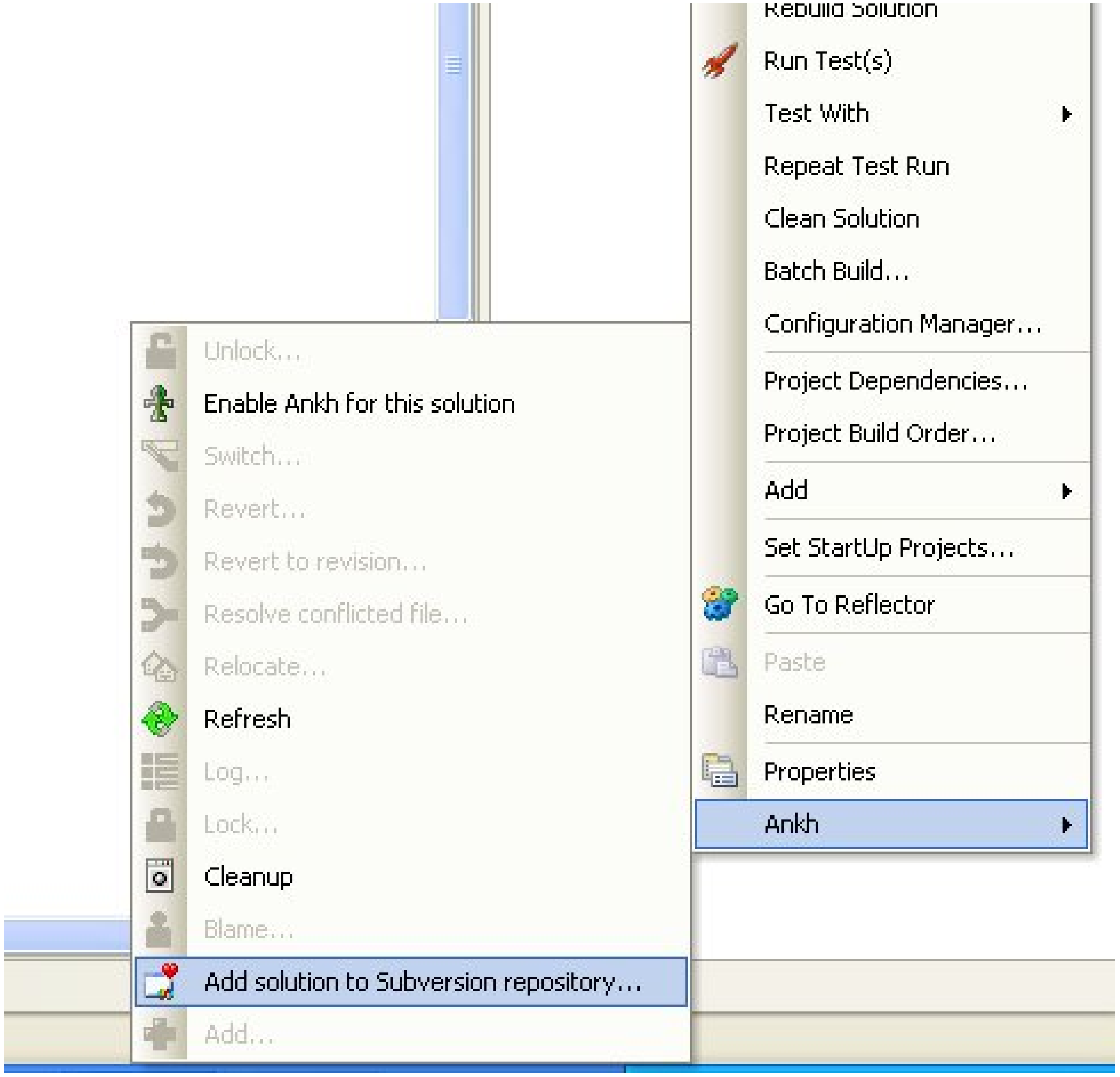
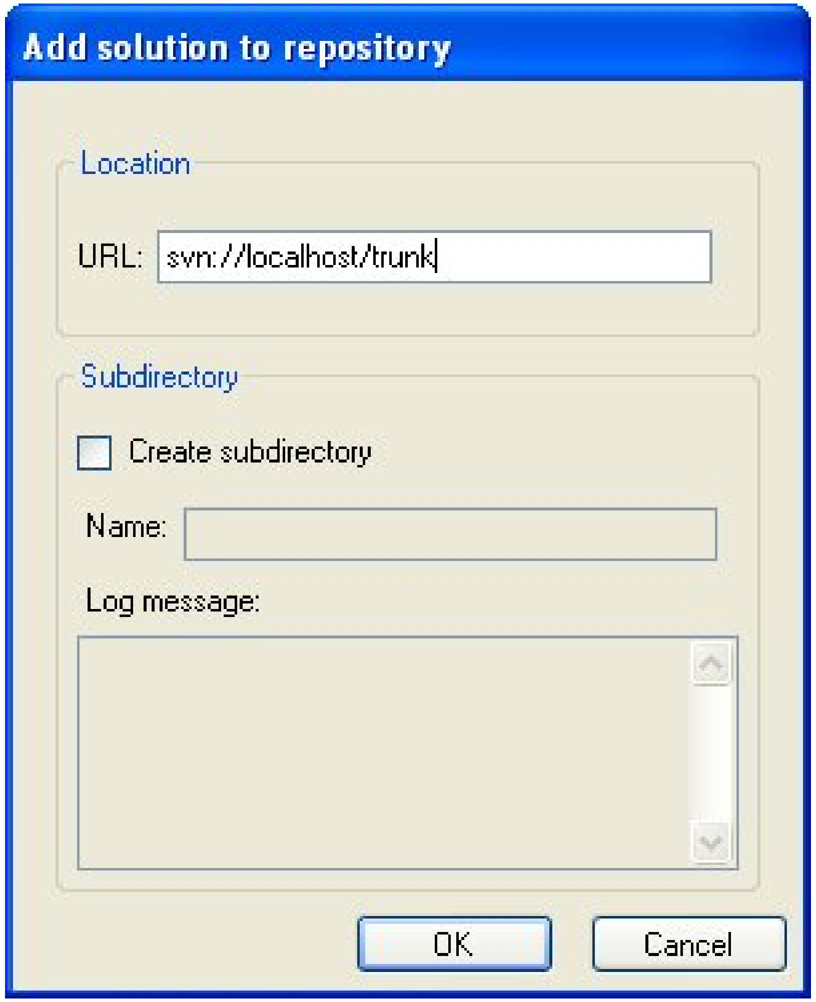


Figure 15. AnkhSVN: add solution to subversion repository.

The appropriate window will pop up (see: Figure 16) asking you the repository url. Fill the **URL** textbox with: svn://localhost/trunk , leave '*Create subdirectory*' unchecked and click **OK**.


Add solution to repository

Location

URL: svn://localhost/trunk

Subdirectory

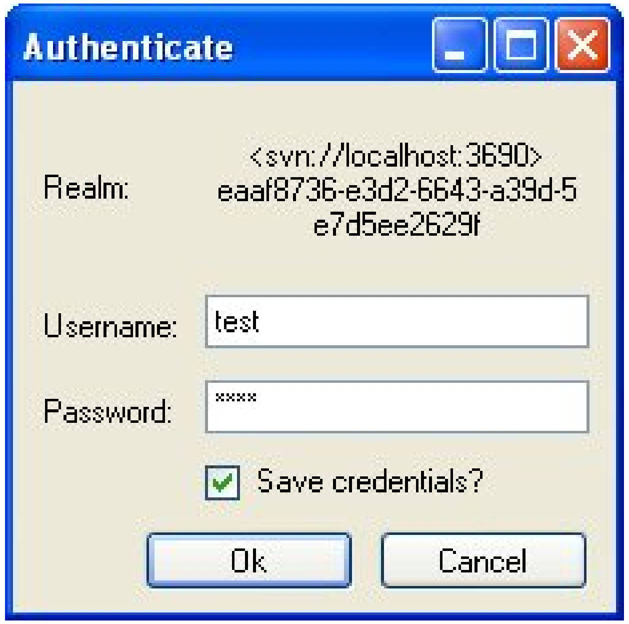
Create subdirectory

Name:

Log message:

Figure 16. AnkhSVN: add solution prompt window.

You will be prompted for username and password (they are managed separately by **TortoiseSVN** and by **AnkhSVN**) as shown in Figure 17.
Fill both the fields and check '*Save credentials*'.



Realm: <svn://localhost:3690> eaaf8736-e3d2-6643-a39d-5e7d5ee2629f

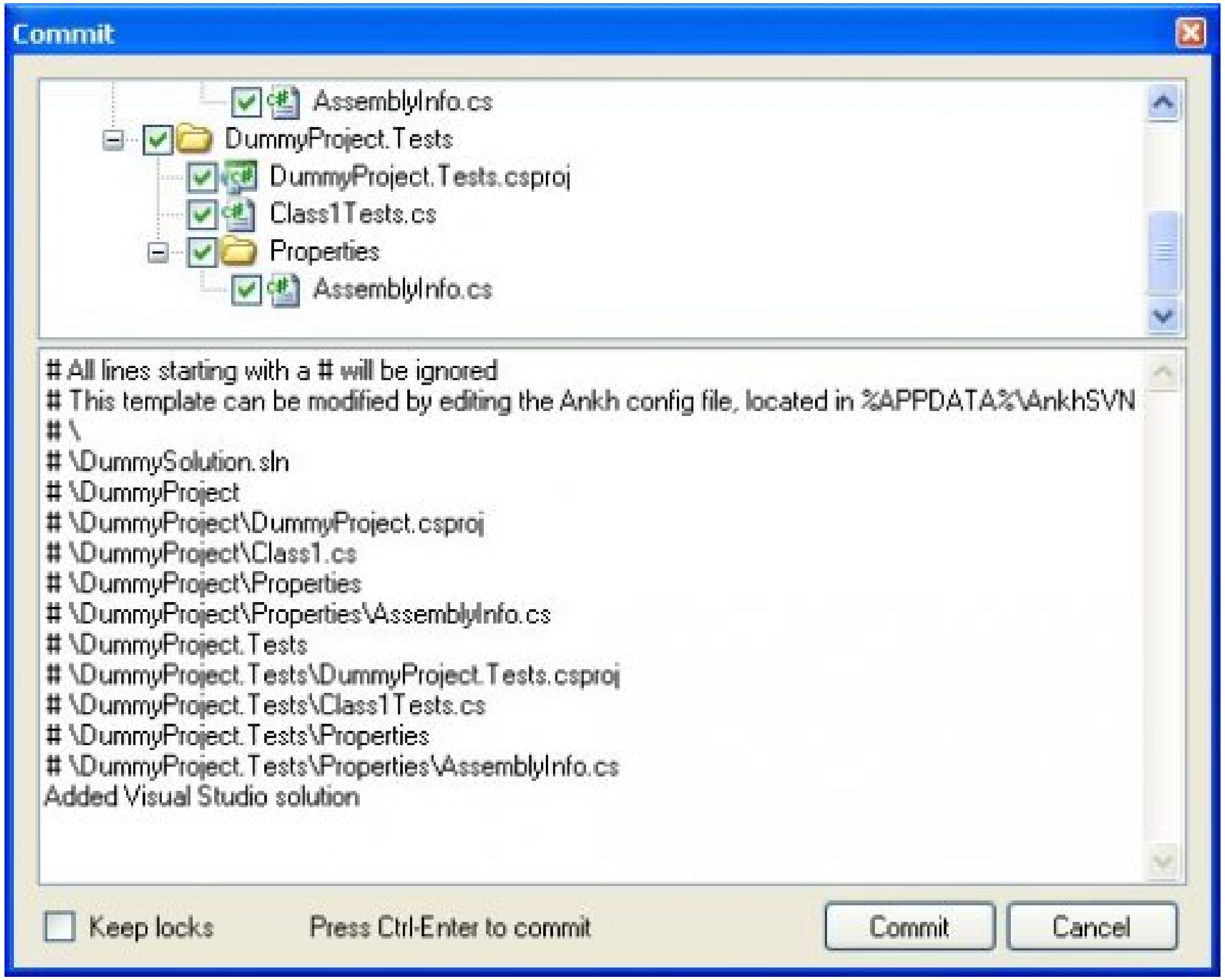
Username: test

Password: ****

☑ Save credentials?

Figure 17. AnkhSVN: provide username and password.

Upon clicking the **OK** button one more window will popup showing you the files that AnkhSVN is going to add to the repository and waiting for you to add a meaningful log message for the commit (see: Figure 18).
AnkhSVN already left out the directories: **bin**, **obj** and their content as well as files: **DummyProject.csproj .user** and **DummyProject.Tests.csproj.user** that you had to manually uncheck when using TorotoiseSVN.
Write something like: ‘*Added Visual Studio solution*‘ and click the **Commit** button.



```
# All lines starting with a # will be ignored
# This template can be modified by editing the Ankh config file, located in %APPDATA%\AnkhSVN
# \
# \DummySolution.sln
# \DummyProject
# \DummyProject\DummyProject.csproj
# \DummyProject\Class1.cs
# \DummyProject\Properties
# \DummyProject\Properties\AssemblyInfo.cs
# \DummyProject.Tests
# \DummyProject.Tests\DummyProject.Tests.csproj
# \DummyProject.Tests\Class1Tests.cs
# \DummyProject.Tests\Properties
# \DummyProject.Tests\Properties\AssemblyInfo.cs
Added Visual Studio solution
```

Figure 18. AnkhSVN: provide meaningful log message.

A *Committing* progress bar will appear and, upon success, the solution explorer will be updated with green icons before the names of versioned items (see: Figure 19).

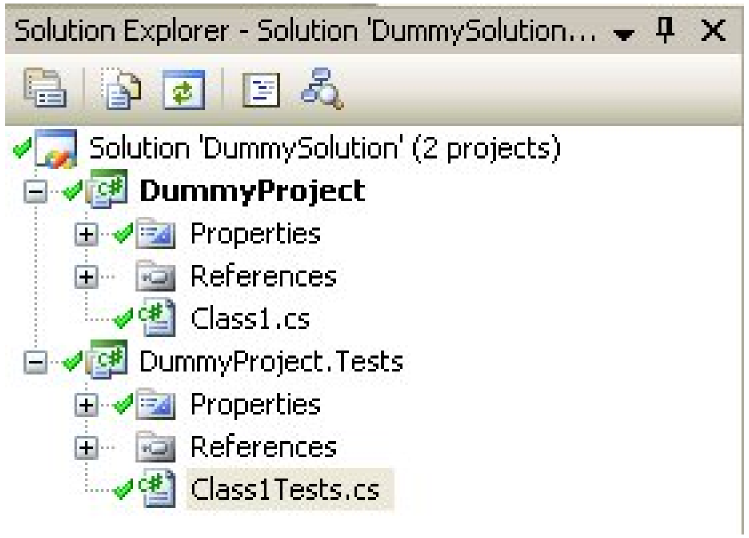


Figure 19. AnkhSVN: Solution explorer.

Now that AnkhSVN is enabled for this solution, each time you add a new item to a project it will be marked with a question mark icon to let you know that the file is not added to versioning (see: Figure 20).
Let’s see it in action by adding a new item in solution explorer:
right-click on **DummyProject** and choose: **Add –> New Item…**, select the **Class** item template, leave the default name (should be *Class2.cs*) and click the **Add** button.

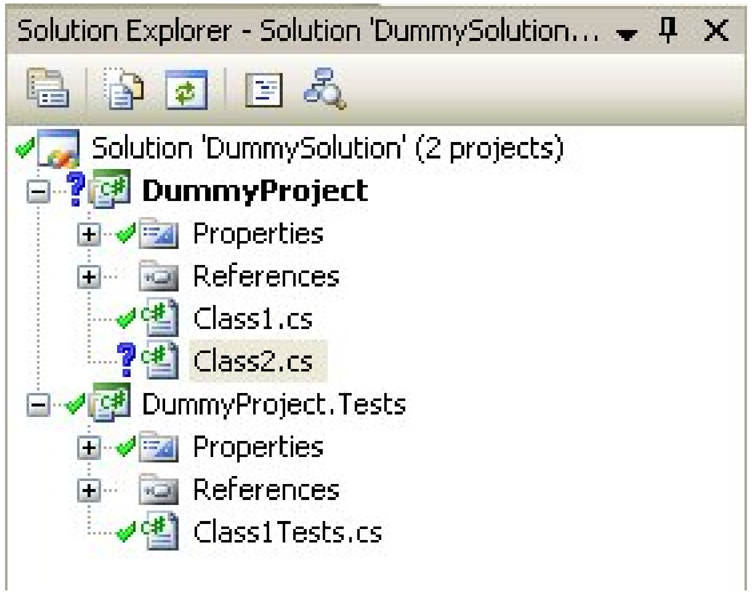
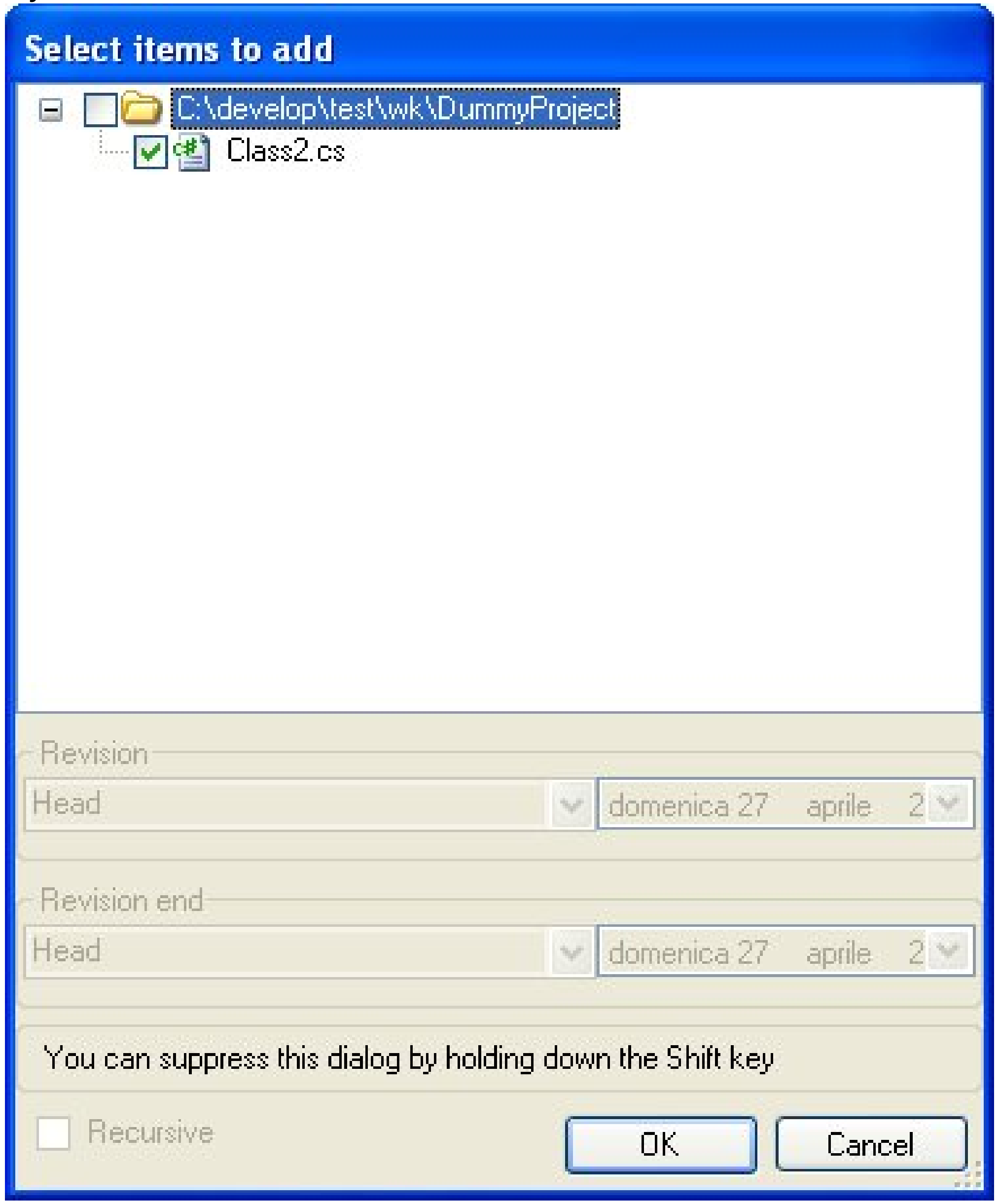


Figure 20. AnkhSVN: Solution explorer with new item.

If you right-click on the solution in solution explorer and choose: **Ankh –> Add…** a new window will open, whose title is: *Select items to add* (see: Figure 21).
It contains the list of non versioned files in the current solution, each prefixed by a checkbox.
They are all checked by default, meaning that they will be scheduled to be added to versioning if you will click **OK**.


Select items to add

C:\develop\test\wk\DummyProject

[✓] Class2.cs

Revision: Head — domenica 27 aprile 2

Revision end: Head — domenica 27 aprile 2

You can suppress this dialog by holding down the Shift key

Figure 21. AnkhSVN: select items to add to versioning.

After choosing **OK** the *Class2.cs* file in solution explorer will have a yellow plus icon beside (see: Figure 22), meaning that it is scheduled to be added at the next commit (the same is true for the project and solution items).

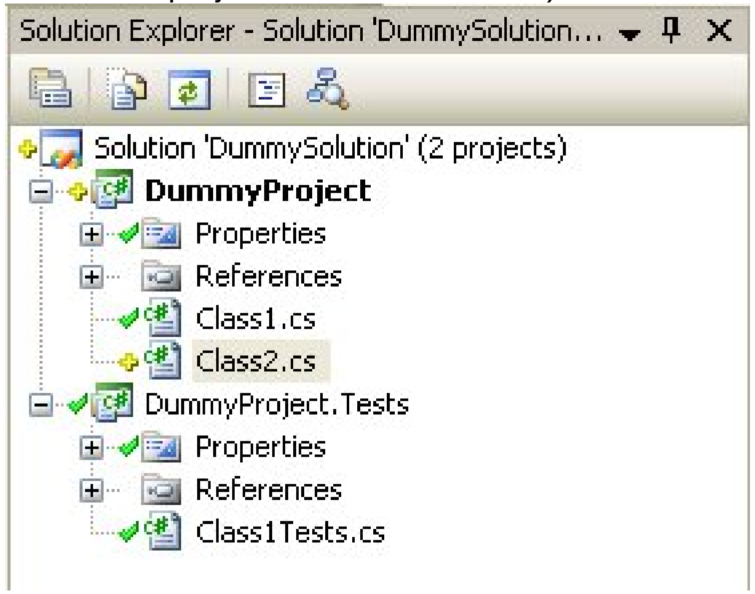


Figure 22. AnkhSVN: new item added to versioning.

Right-click the solution item and choose: **Commit…**. The *Commit* dialog will open.
At the same time the icons beside the project and solution items become red squares (see: Figure 23), meaning that there are local modifications that need to be committed to the repository.
Provide a meaningful message and click the **Commit** button (see: Figure 24).

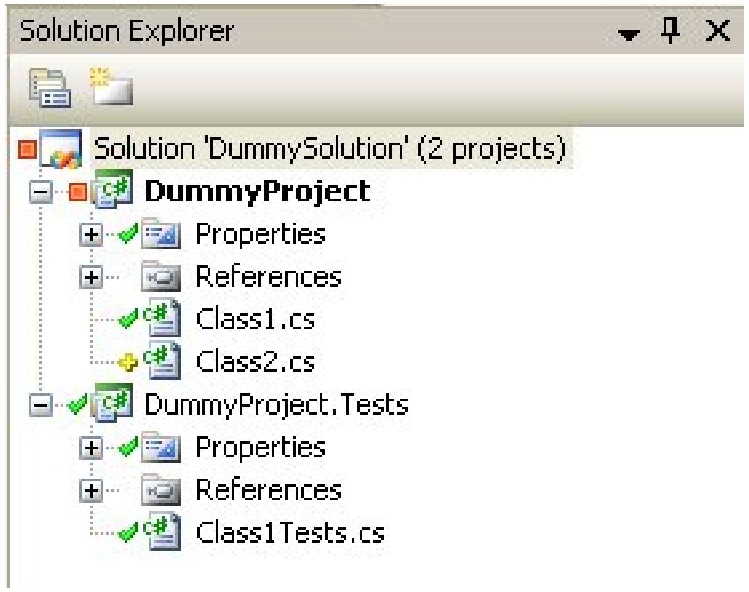
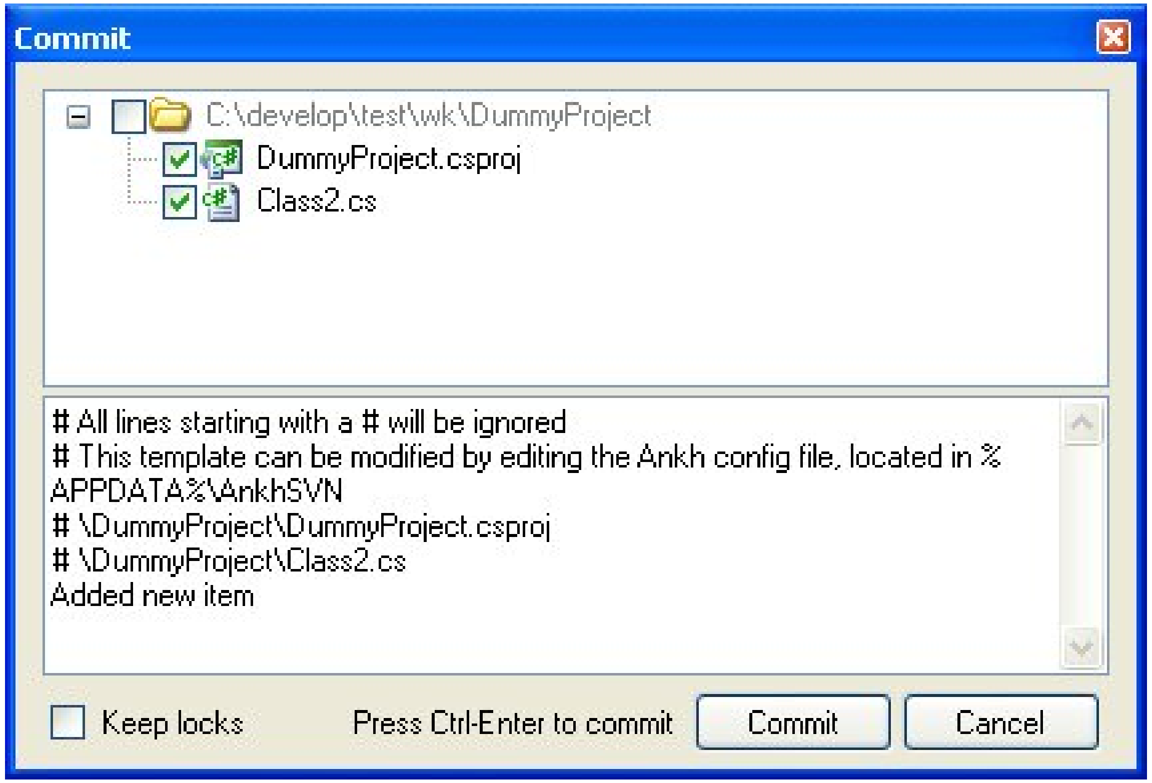


Figure 23. AnkhSVN: solution explorer upon commit.



```
# All lines starting with a # will be ignored
# This template can be modified by editing the Ankh config file, located in %
APPDATA%\AnkhSVN
# \DummyProject\DummyProject.csproj
# \DummyProject\Class2.cs
Added new item
```

Figure 24. AnkhSVN: commit dialog.

Whenever you want to quit using AnkhSVN you can disable it for the solution you are using.
Just right click the solution item and choose: **Ankh –> Disable Ankh for this solution**.
You can then work using TortoiseSVN or the Subversion command-line.
If you want to enable Ankh again just choose: **Ankh –> Enable Ankh for this solution**.

→ top of post
→ top of paragraph

7. Install Subversion as a Service

If you want to Install Subversion as a Windows service you can use the utility: **sc.exe**, where sc stands for “Service Control”.
Below you find a sample batch script to automate the install process.
You can copy into a text file and save it, for example, as InstallSvnService.bat.
Open a command prompt in the directory in which you placed the .bat file and run it by digiting InstallSvnService and pressing return.
The service will be installed and configured to start automatically.

```

sc create svnserve ^
binpath= "\"C:\Program Files\Subversion\bin\svnserve.exe\" -r ^
\"C:\develop\test\repo\" --service" ^
displayname= "Subversion" ^
depend= Tcpip ^
start= auto
```

Note that the caret symbol (^) in the script code is the line continuation character for batch scripts. It allows us to split the long command in multiple lines in order to make it more readable.
The line continuation character tells to the command interpreter that the current command is not ending at the first newline but is going on in the next line.

It’s possible that you need to change the path:
“C:\Program Files\Subversion\bin\svnserve.exe” depending on where is your copy of Subversion installed.
The path: “C:\develop\test\repo\” in the example is the path to the location in which svnserve will find the repositories to host.

Once the batch file has run open the administrative panel for services to make sure that the Subversion service exists and is running:
go to: **Start –> Administrative Tools –> Services**
scroll down looking for a service named **Subversion** and if it is not running click on the **Start** link.

→ top of post
→ top of paragraph

kick it 4

« CruiseControl.NET and Subversion – SvnRevisionLabeller
Line Continuation Character »

Actions

- Comments RSS
- Trackback

Information

- Date : April 27, 2008
- Tags: AnkhSVN, svn, TortoiseSVN
- Categories : AnkhSVN, Subversion, TortoiseSVN, Tutorials, Visual Studio 2005

Like Be the first to like this post.

31 responses

David (17:41:58) :

20
05
2008

Thanks that helped me out.

Reply

Farlin (03:43:23) :

21
05
2008

What if you have accidentally published “DummyProject.Tests.csproj.user”.
How do you suggest one should deal with it? Should it be deleted?

Reply

farlin (03:50:28) :

21
05
2008

by the way, great post!

Reply

ilmatte (09:01:02) :

21
05
2008

Hello Farlin
As far as I know there’s no easy way to remove a file or directory from versioning without deleting it from disk.
What I suggest is to make a copy of the user file (say: DummyProject.Tests.csproj.user.temp), delete the original from versioning with TortoiseSVN and then rename the copy to: DummyProject.Tests.csproj.user.

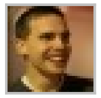
You should definitely remove it from versioning if you don’t want to version things like: showAllFiles (the option that lets you choose to see, in solution explorer, all the files in a project directory even if not included in the Visual Studio project).

If you have the same problem with a directory and you don't want to loose all its contents you can do a: TortoiseSVN –> export of such directory in a temporary folder before deleting the original one with TortoiseSVN.
The export command will provide you with a clean copy of the directory (with no .svn files)

Hope it is the answer you needed.

by the way thanks to you Farlin and to David for appreciating this post

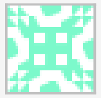
Reply


John S. (16:58:44) :

21
05
2008

Or just install VisualSVN server and VisualSVN.

Reply


M (18:11:19) :

21
05
2008

You can also have a more pleasant experience with Visual SVN and Visual SVN server

Reply

5

kory (15:30:57) :

06
2008

Thank you so much for this tutorial. I have just started monkeying around with SVN, and I was up and running in all of 15 minutes based on this post's information.

Reply

ilmatte (17:41:07) :

5
06
2008

You're welcome Kory,

It's very nice to know that it has been useful.
Thank you for the feedback.
Feel free to post or mail questions if you need.

Matteo

Reply

CMUZ (14:10:51) :

14
07
2008

Hi,

I cant seem to get authentication to work. It doesnt request for a password even after setting the conf files up. I didnt start the service though as it takes forever in the command prompt and I end up closing the window. Is this why the auth wont work

Reply

ilmatte (08:15:16) :

16
07
2008

Hello CMUZ, I’m sorry for the delay in the answer,

I guess that you got it: the problem is related to that command prompt.
Starting svnserve server from the command prompt has the purpose of having a console application running forever to let svnserve to be executed. When you stop the command prompt you stop svn server.
So the right thing is having the command prompt run forever. If you don’t want the command prompt you should install svn as a service.

As far as authentication is concerned:
in paragraph: 5. Checkout a Working Copy you had 2 choices to checkout the repository.
I guess you chose to let the ‘URL of repository‘ be:
file:///C:/develop/test/repo/trunk, because it was your only choice if not having a svnserve server instance running.
In that case (using file urls) TortoiseSvn (as well as Subversion command line client) acess directly the repository on disk and the only authorization rules are filesystem permissions (so they ignore the authentication rules you set up in svnserve.conf and passwd files).
You should run svnserve from the command prompt and checkout a working copy as described in the second option of paragraph 5. Checkout a Working Copy, that is with a ‘URL of repository like the following: ‘svn://localhost/trunk (this will work if you run svn with: svnserve -d -r “C:\develop\test\repo”). The svn:// protocol provides access to an svnserve server if there is one running. It is the svnserve server that checks the authentication and authorization rules set in the directory: “C:\develop\test\repo\conf”.

Hope it helps

Reply

sumant (03:46:47) :

22
07
2008

Thanks so much for such a great post! I had been searching on how to integrate AnkhSVN with VS2008 but didn't know that I also have to install Subversion & TortoiseSVN.

A great relief indeed! thnx!

Reply

ilmatte (08:08:50) :

22
07
2008

You're welcome, sumant,
I'm happy that it was useful and thank you for your feedback

Reply

ilmatte (12:49:36) :

25
08
2008

As far as VisualSvn is concerned…well I tried it and I liked it (even if it is not free).
But I think there's still a big problem with it:
once you versioned a visual studio solution you cannot disable visualsvn for that particular solution and you cannot even add a project to that solution

without adding it to versioning.
Hope this features will be added soon.

Reply

Jennifer Bonnett (20:20:57) :

14
10
2008

I'm having some issues creating my repository. I keep getting the following error message: Unable to open an ra_local session to URL; Unable to open repository 'file:///C:/test'

Any suggestions?

Note: I have IIS running on this machine, do I need to maybe change a port #?

Jennifer

Reply

ilmatte (12:33:03) :

24
10
2008

Hi Jennifer,

I guess that you received that message when trying to Checkout a Working Copy like in paragraph:

http://ilmatte.wordpress.com/2008/04/27/guide-to-versioning-a-visual-studio-solution-with-subversion-tortoisesvn-and-ankhsvn/#svntutwk

My guess is that you typed the incorrect repository path. In fact the error reports:

Unable to open repository 'file:///C:/test but probably in that directory you don't have a repository.
In the article, for instance, the repository is created under:
file:///C:/develop/test/repo
Try again checking out the path:
file:///C:/develop/test/repo/trunk
Hope this answers your question

Reply

Shehan (18:42:58) :

29
10
2008

Awesome guide, it kinda saved my life! 😉 Thanks!

Reply

ilmatte (09:35:40) :

30
10
2008

😃 You're welcome Shehan

Reply

Sergi (17:57:10) :

13
11
2008

Great article, it should help many to begin with these great tools!

Anyway, I have to say that I tried to use AnkhSVN, but at last we decided to drop it from our lives and survive just with TortoiseSVN… the main problem we had with Ankh is that when 2 users modify the .proj or .sln files, and we have a conflict, Ankh modifies the files themselves and introduces the conflict information with the “>>>>>mine” and “<<<<<theirs” stuff….

This makes VS not to open the project again… unless you edit manually the conflicts… let’s say with Notepad…. 😕 … and that’s not the best way.. I believe…

Anyway, VS + Tortoise is not a bad combination, as long as you have both installed if you follow your article.

Cheers!

Reply

ilmatte (08:44:12) :

14
11
2008

Hello Sergi,

I’m happy that you found my article useful.
I’m sorry about the problems you had with AnkhSVN and I’m not surprised at all.
My team experienced some problems as well in the last months.
Even the new version of Ankh still lacks some functionalities and behaves in an odd manner sometimes.
I stopped using it temporarily as well, but I keep supporting the AnksSVN team efforts while I’m waiting for their product to become mature enough to be used in a professional environment.

Reply

Jose (04:41:35) :

1
12
2008

Hi,
It’s the bes tutorial that I’ve seen.
I’m studying software development…so i have to do a report about Subversion, due to I have, well, we, our team have to develop a web application, using asp.net and c# with VS2005…
So I asked myself if you let me to copy your tutorial, but I’ll tell the source, the true author indeed.
Well, I’ll have to translate it to spanish.
Pd: please…

Reply

ilmatte (09:12:12) :

1
12
2008

Hi Josè,

it’s very kind of you to be so honest and ask to me the authorization.
Obviously you can copy the content of my blog posts whenever you want (providing the true source as you said hopefully 🙂).
I’m very happy about having been useful to your team and thank you very much for your feedback

Reply

Information Technology Draft » Blog Archive » Source Control (10:32:53) :

14
12
2008

[...] Versioning a Visual Studio Solution with Subversion, TortoiseSVN and AnkhSVN [...]

Reply

Dan (15:14:06) :

21
12
2008

Hi,

Great tutorial, but it seems a bit outdated.
The versions I downloaded and installed today seem to have a somewhat different interface, specifically the Ankh tool.

Other than that, extremely informative.
Now I can continue learning about source control with the more detailed manuals and applying that to my VS.

Thank you for spending the time and effort to write this down.

Reply

DIYS » Software tools og versionsstyring (11:18:12) :

17
02
2009

[...] Subversion i Visual Studio [...]

Reply

Eric (03:44:07) :

23
02
2009

Thank you so much for this article. I was frustrated by svn+ankh but your clear and straightforward steps helped me set it up. Keep up the good work.

Reply

GND (12:46:51) :

8
10
2009

This is a great post, it worked for me without any problem, usually i get all sorts of problems. This is just AMAZING. One more time THANK YOU….

Reply

Robert A. Green, MBA (22:37:43) :

17
11
2009

Thanks very much for assisting me in solving this problem. Once the breakthrough is manifested, I will be sure NOT to forget about how you assisted me in reaching my goal(s).

Thanks from my company and the entire world.

Regards,

Robert Green, MBA.

Reply

ravi (14:08:22) :

4
12
2009

Hello,

i am getting the below mentioned error when i start svn service from services.msc,please help me.

could not start the subversion service on local computer.
error 1053: the service did not respond to the start or control request in a timely fashion

Reply

David Bishop (13:01:39) :

31
12
2009

Hi,

This is a great post that I have referenced before but had a couple of issues which I think maybe to do with Vista or a new version of Tortoise.

Firstly when I created the repository via Tortoise it didn't ask me what file type I wanted to use. When trying to browse the repository it was unable to and I ended up deleting the repository and creating manually using the following command line.

svnadmin create –fs-type fsfs C:\Repository

(Note I moved the repository to a different location than the instruction)

I also had a problem creating the service, Vista didn't seem to like the ^ on some of my batch file lines, it also seemed to get confused with some of the escaping. I fixed this will the following command.

sc create svnserve binpath="C:\Program Files\Subversion\bin\svnserve.exe -r C:\Repository –service" displayname="Subversion" depend=Tcpip start=auto

Other than that everything worked a treat, thanks for posting!

Reply

kartika (21:09:16) :

6
08
2010

hi ,

I am using CVS instead of svn ..??but i have not found such good tutorials about using CVS with it ..can you help me out with it …

Reply

Alexia (07:46:17) :

16
09
2010

hello,

There is a new tortoise svn addin for Visual Studio 2005/2008. Its free GPL. You can get it from here: http://sourceforge.net/projects/tusvnaddin/

Reply

Leave a Reply

Your email address w ill not be published.

	Name
	Email

Website

POST COMMENT

☐ Notify me of follow-up comments via email.

☐ Send me site updates