USING SUBVERSION (SVN) VIA TORTOISESVN

Subversion (SVN for short) is a source code versioning system. TortoiseSVN is a GUI to access SVN. As the first step, download TortoiseSVN and install it on your machine. If you are using a 64-bit Windows, make sure you are download the 64-bit version of TortoiseSVN.

Each team has been assigned space on our SOC SVN server. For example, the following URL https://subversion.comp.nus.edu.sg/svn/cs3201/cs3201g99/trunk/ is for group 99 (**Note**: other groups simply replace the name "cs3201g99" in the URL by your own group number, for example cs3201g02). This space, called a *repository*, is simply a central location where we keep shared files. However, this is different from standard file sharing because SVN also tracks changes made to file over time by different authors. You can browse the repository content by accessing the given URL through your web browser. Simply login with your SOC **UNIX ID**. For those accessing the repository from outside, please use SOC VPN (**not** NUS VPN).

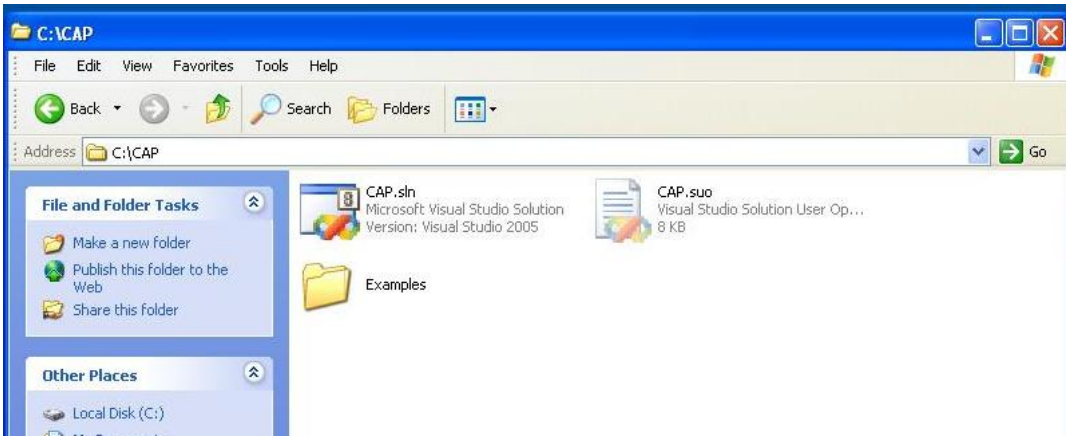
The repository set up for you is created with the SVN-recommended directory structure, this includes three directories namely: 'trunk', 'branches' and 'tags'. 'trunk' directory is used as the location to store files corresponding to the main line of development. 'branch' directory is used to start a new branch of development, without affecting the main line of development. 'tags' directory is used to save special snapshots of your system (e.g., intermediate releases). Our course usually needs only the 'trunk' and 'tags' directories.

Step 1: Initializing the repository (performed by one member for each team)

CREATE DIRECTORY STRUCTURE

The repository given to each team is initially empty. Assign **one** team member to initialize it with the basic directory structure. Create a new solution, say CAP, used as the project name in this guide, from the VS.Net 2005. Add several projects to the new solution (corresponding to sub-parts of the CAP system, e.g., Query Evaluator, CKB, etc.). Organize your solution well, it is harder to modify this directory structure once uploaded to the SVN. Save the solution at the desired location (let's assume it is C:\CAP) . VS.Net 2005 will create the solution file and a list of folders corresponding to projects under C:\CAP. Let's call this directory the temporary *project directory*.

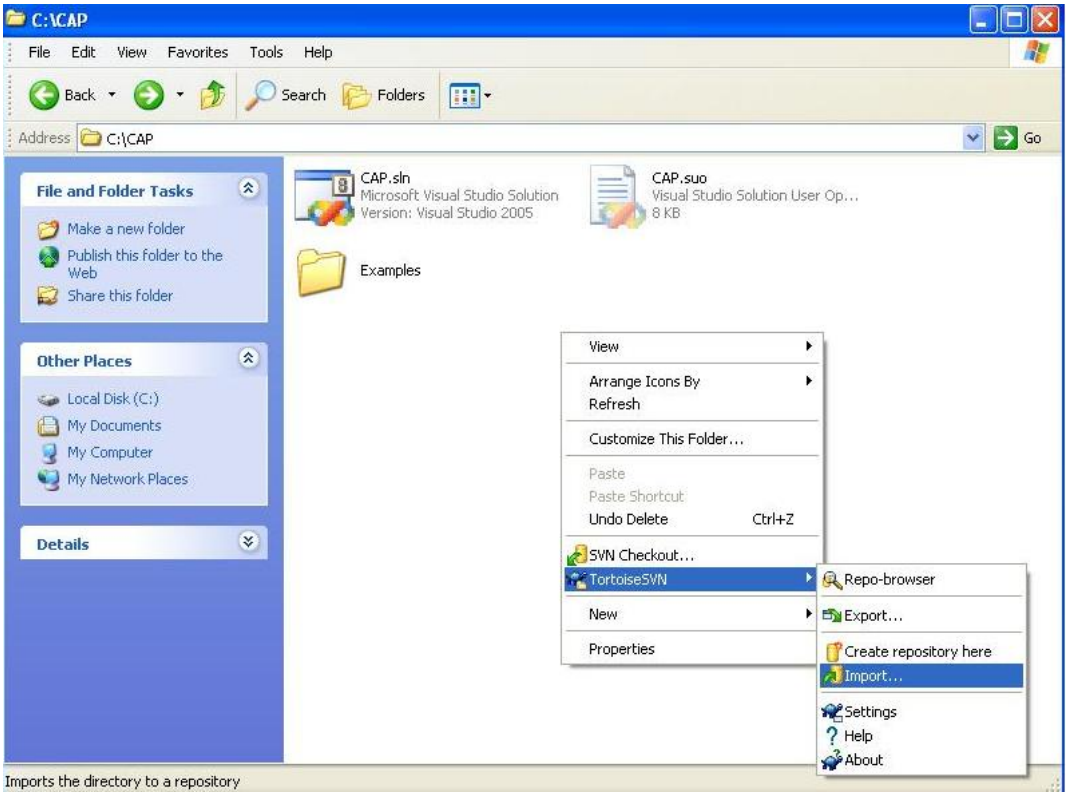
For illustration purpose, let's assume that at this step we have created a solution named *CAP* at C:/CAP with a project named *Examples* having one C# file named *Form1.cs*. Refer to the following snapshots of the directory structure.



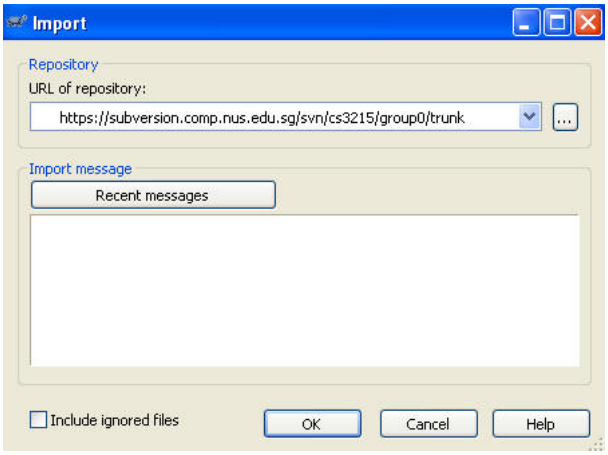
IMPORT TO REPOSITORY

Remove all unnecessary files to build CAP (temporary files, compiler-generated files, e.g. exe, obj, dll) from the project directory and its subdirectories. However, keep the solution (.sln) and project files (.csproj). Alternatively, you can set Tortoise SVN to ignore such files in the following manner: Right click an empty area of the desktop, choose **TortoiseSVN > settings**. Enter the string "*.exe *.dll *.user" (without double quotes) into the **global ignore pattern** box.

Let's now import to the repository (save the directory structure to the repository). Start the file explorer, go to the project directory (C:/CAP) right click and choose `TortoiseSVN>Import' as shown below.



The following dialog box will appear:

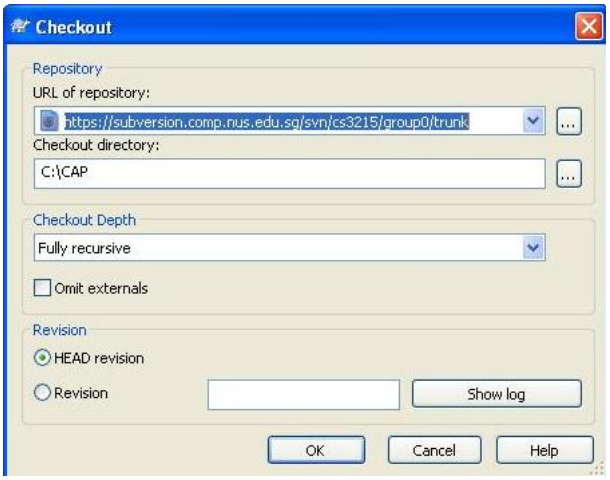


Type in the URL of the repository. Again, simply replace the "group0" in the above URL by your own group number. Remember that 'trunk' directory store files corresponding to the main development line. Great ! you have initiated the directory structure in your SVN project account.

Delete the content of C:/CAP to prepare for the next step. (yes delete it :-))

Step 2: Creating Working Copy (performed by every member of each team)

'Working copy' holds a copy of the repository in your local machine. This is the main directory where you access, modify and update source files for your project. The SVN client will maintain and manage files stored in the working copy. To create a working copy, go to your project directory (C:\CAP), right click and choose 'SVN Checkout'. The following dialog box will appear.



Put in the detail of your SVN repository URL (as mentioned in Step 1) and click OK.

'C:\CAP' will be initialized as your working copy. The SVN client will manage or 'remember' the link between files in the working copy and files stored in the repository. You will see the files and directories corresponding to the SPA solution your team member has initialized in Step 1. You will find that all the contents are the same as the original except for the inclusion of a SVN subdirectory (a hidden folder with suffix ".svn")in each directory. Congrats ! You have just retrieved your first working copy.

Note: 'import' command invoked in step 1 doesn't create a working copy, only a 'checkout' will create a working copy.

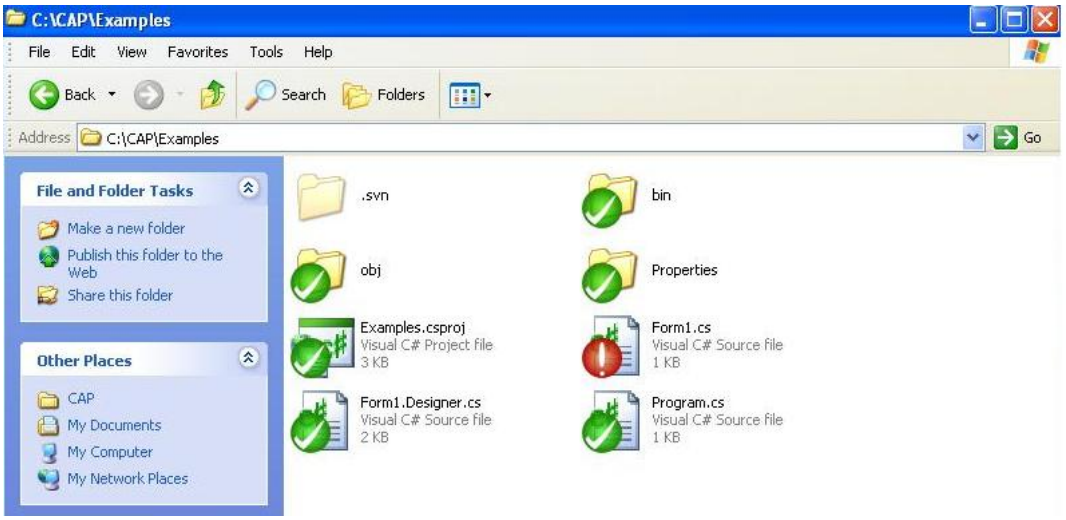
Step 3: Modifying your Working Copy

Assume the content of the file, Play.cs is as follows.

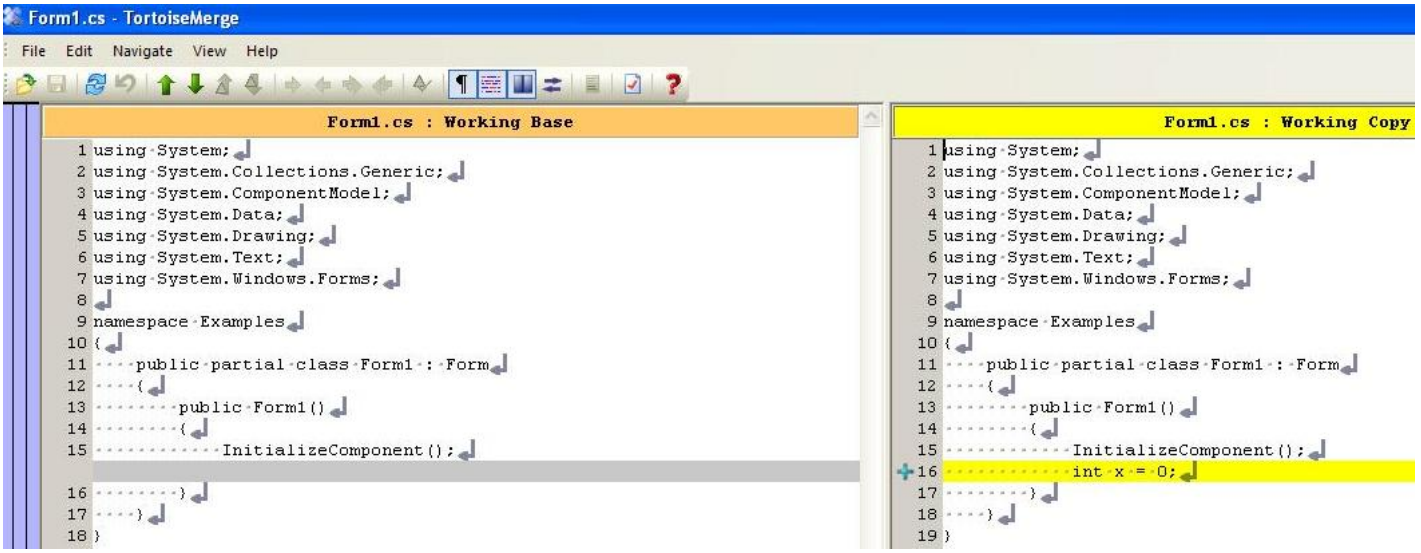
```
1  using System;
2  namespace Examples
3  {
4    public class Play
5    {
6     }
7  }
```

DIFF

Now, let us make some changes to the source files. Edit Form1.cs of your working copy (in C:\CAP). Add the statement "int x = 0;" to the body of Form1 class constructor and save the file. Note that the icon to Form1.cs has now changed. Form1.cs icon is marked (or overlaid) with a red exclamation mark. This indicates that the file has been changed or modified.



To check what changes has been made, right click Form1.cs and choose 'Diff'. The following dialog box will appear.



It highlights the modification that has been made to the original Form1.cs file.

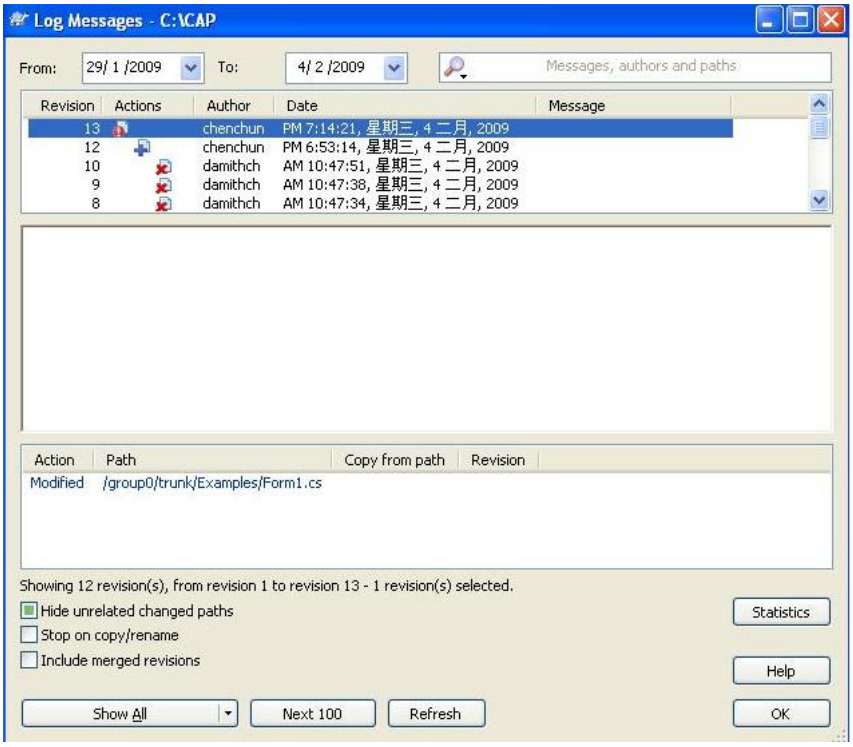
COMMITTING

Now, you will learn how to commit your working copy. The *commit* command will update the files in the repository with those in your working copy. Go to your working copy (C:/CAP), right click and choose 'SVN Commit'. The following dialog box will appear.

Add comment to the message box to indicate the changes that have been made (For example, "Addition to constructor body of Play class" in the above dialog box). The comment will be very useful in the future for change-tracking purpose. During the lifespan of the project, there will be multiple changes done by different developers at different time. Browsing the log messages will be useful to get an overview of what have changed.

Click OK.

To view a log of changes that have been made, right click, choose 'TortoiseSVN>Show Log'. The following dialog box will appear. It shows a history of changes that have been made to the repository.



Every commit will advance the 'Revision' number by 1. The 'Action' column in the log denotes the action that has been taken. The 'Author' column in the log denotes who had made the change (who to blame :-)). The 'Date' and 'Message' columns provide information on when changes were made and what was the nature of the changes.

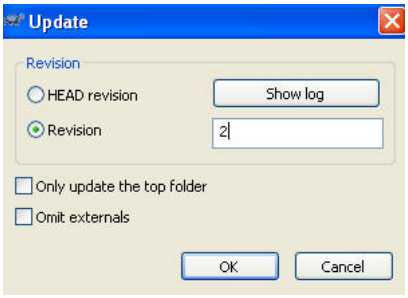
UPDATE

To update your working copy with the latest version in the repository, right click and choose 'TortoiseSVN>SVN Update'.

REVERTING CHANGES

There are two different cases. First, you might realize that you have made a wrong (**un-committed**) change and would like to revert to the latest version in the repository. To do this, right click (the directory or file to revert) and choose 'TortoiseSVN>Revert'.

Sometimes, you may discover that you need to revert back to an earlier version. Here's the way to do it. Right click, choose 'TortoiseSVN> Update to Revision'. The following dialog box will appear.



Enter the revision number to revert to and click OK.

Check Play.cs and you will see that it is reverted back to the original version. Note that the file is reverted only at your working copy. To propagate it to the repository, perform some changes to Play.cs and commit it.

ADDING/REMOVING DIRECTORIES/FILES

To add a file/directory:

- Create the file/directory in your working copy first
- Update the repository by right clicking the new file/directory, and choose 'TortoiseSVN>Add'
- Commit the changes

To remove a file/directory:

- Right click the file or directory
- Choose 'TortoiseSVN>Delete'
- Commit the changes

Tagging

Tagging is simply taking snapshots of the repository. We take a snapshot of the repository (i.e., create a tag) at important points of your project such as at the end of an iteration. This is because each iteration will signify a stable version. If things get too messy as you move to the next iteration, you will be happy to know that you can fall back on the set of stable code (from the last iteration) and start again.

To create a tag, select the folder in your working copy which you want to tag, then select the command TortoiseSVN ->Branch/Tag...., and give the URL which contains the new tag. For example, to create a tag called 'V1.1' you could give the URL as '[SVN_repository_URL]/tags/V1.1'. It looks like you are creating a copy of the source code and gave it a different name, but SVN internally stores it in a more efficient manner.

Special Note

Since we don't want to swamp you with too much information, we will end the guide here. However, we left out an important topics that we hope you all will read up on your own (get you team's SVN guru to help you here): resolving conflicts. A conflict arises when both you and another developer happen to modify the same portion of a source file. SVN does not resolve such conflicts for you. However, it will warn you of any conflicts detected. You need to communicate with the other developer to resolve the conflict.

References

1. Tortoise SVN Manual (in pdf), downloadable from http://tortoisesvn.net/downloads
2. SVN Red Book.