Software Methods and Tools

Spring 2018
Instructor: Yongjie Zheng

Lab #7: Subversion

Description:

In this lab, we will go over the Subversion commands and operations that are frequently used in practice, including creating a repository, organizing the repository, creating a working copy, and the basic checkin and checkout operations. The purpose of this lab is to help students get familiar with the work cycle of Subversion, and understand its revision number assignment mechanism.

Instead of having instructions and exit exercise, we will be directly working on a list of specific tasks in this lab.

Tasks:

1. Running Subversion and setting up the environment.

The Subversion application can be started from Start > Specialized Programs > Subversion > Subversion Command Prompt on lab machines.

Once the command line window pops up, type in

set SVN_EDITOR=C:\Windows\System32\notepad.exe

This command specifies that we are going to use Windows' NotePad application to edit the Subversion messages.

2. Creating a Subversion Repository.

Run the command, svnadmin create Q:\Desktop\repos

Note that the folders before the last anti-slash must exist when you run the command, and Subversion will automatically create the last folder for you.

Open the "repos" folder in Windows browser, and check the content of a Subversion repository.

3. Creating a project in the repository.

First, create a folder on your machine. For example, Q:\Desktop \LunarLander.

Run the command, svn import LunarLander file:///Desktop/repos/ LunarLander

This will create a LunarLander folder in the “repos” repository.

4. Creating a Working Copy.

Create another folder on your desktop, WorkingCopyOne.

Run the command:

svn checkout file:///Desktop/repos/LunarLander WorkingCopyOne

This command will check out a working copy of the project “LunarLander” into your folder “WorkingCopyOne”.

5. Making changes to the file structure.

Change the directory to the WorkingCopyOne folder: cd WorkingCopyOne.

Run the following three commands successively:

svn mkdir trunk, svn mkdir tags, svn mkdir branches.

Go to the trunk folder, and put the three Java files (GameController.java, SpaceCraft.java, and LunarLander.java) you downloaded from the class website there.

Run the following three commands successively:

svn add GameController.java, svn add SpaceCraft.java, svn add LunarLander.java

Now all the files have been put into the protection of Subversion.

Before you commit your changes to the repository, run

svn status -v

This command will show you what changes you have made to the your working copy. To see what the repository looks like, run

svn list -v file:///Desktop/repos/LunarLander

The repository should be empty at this point since you have not committed your changes yet. Now let's do it:

svn commit

After commit is done, check your repository again:

svn list -v file:///Desktop/repos/LunarLander

6. Making changes to the file content.

Now, we are going to create another working copy of the LunarLander project.

First, we need to create a folder (WorkingCopyTwo) on your desktop.

Next, checkout the LunarLander repository into this folder by running:

svn checkout file:///Desktop/repos/LunarLander WorkingCopyTwo

This creates the second working copy of the LunarLander project.

Open a file (e.g GameController.java) and make some random changes to it. Please make sure that you remember what changes you have made.

Commit your changes: svn commit

Check the revision number of your working copy: svn status -v

Update the local working copy: svn update

Check the revision number again: svn status -v

What differences did you find in the revision numbers?

7. Resolving conflicts.

Now go back to the folder of your first working copy, WorkingCopyOne.

Again, make some random changes to the same file of the first working copy. Please make sure that you change the same line as you did with the second working copy. You can also make some other changes.

Try committing your changes, see what happens: svn commit

To get synchronized with the repository, run: svn update

Conflicts should be reported. Choose (e) to edit the changes. Once you are done modifying the file, select (r) to resolve the conflicts.

Commit again: svn commit.

Check the revision number: svn status -v

Update your working copy again: svn update

Check the revision number: svn status -v

Do you understand why the revision numbers are different before and after you update your working copy?

END.