SubversionTutorialPart2

From Erh-itproject

This tutorial is designed to show a basic work flow using Subversion and to demonstrate the use of some of the svn commands. Please keep the SVN Book handy for reference, especially Chapter 2, for more details and descriptions of how to interpret the output of the commands.

Part 2

Part 2 covers the basics of multiple developers modifying an application.

This tutorial presents the commands you should run but in general does not show the results (to encourage you to actually perform the commands). A sample log of actually performing this tutorial is at SubversionTutorialLogPart2. Commands you should enter are prefaced with ==> and are in bold.

Here we will simulate another developer named Joe making changes to myApp. First Joe has to check out the application:

```
==> mkdir -p $HOME/svnwork/joe
==> cd $HOME/svnwork/joe
==> svn checkout file://$HOME/svnrepo/test/trunk/myApp
==> cd myApp
```

Joe tries to run the application and sees it doesn't print out the site as it is supposed to:

```
==> ./myscript.sh
Resource file myApp.config missing
My config=
```

Joe looks at the file and decides he is going to fix it. He edits myscript.sh and inserts an "exit 1" after the echo statement for the resource file. Edit myscript.sh to look like:

```
#!/bin/sh
# Source in configuration file
if [ -f myApp.config ]
then
  . myApp.config
else
  echo "Resource file myApp.config missing"
  exit 1
fi
echo "My config=$site"
```

After editing, test running the script to check for errors and commit.

```
==> svn commit -m "Added exit 1 if resource file not found."
```

Now lets go back to being the primary developer:

```
==> cd $HOME/svnwork/test/myApp
==> svn status -uv
```

which results in:

```
$ svn status -uv
       *        7        3 paul         myscript.sh
                7        6 paul         config/myApp.config
                7        6 paul         config
                7        7 paul         .
Status against revision:      8
```

The "*" with myscript.sh indicates that myscript.sh has been updated in the repository and is out of date in your working copy. Normally, an svn update would be run to update the working copy. However, don't do this now because we are simulating when two developers are working on the same file. Edit myscript.sh to add "exit 0" at the end to return a successful return code:

```
#!/bin/sh
# Source in configuration file
if [ -f myApp.config ]
then
  . myApp.config
else
  echo "Resource file myApp.config missing"
fi
echo "My config=$site"
exit 0
```

And now check the status before we commit:

```
==> svn status -uv
M      *        7        3 paul         myscript.sh
                7        6 paul         config/myApp.config
                7        6 paul         config
                7        7 paul         .
Status against revision:      8
```

Note in the status that myscript.sh is both modified (M) and out of date (*) with the repository. This means there could be changes already committed to the repository that aren't reflected in our working copy. Lets try to do a commit:

```
==> svn commit -m "Added exit code for success"
Sending        myscript.sh
svn: Commit failed (details follow):
svn: Out of date: '/trunk/myApp/myscript.sh' in transaction '8-1'
```

The commit fails. Going back to the SVN book, Chapter 2, Basic Work Cycle, it shows we need to check and merge others' changes into our working copy. To get changes in myscript.sh, you must run svn update:

```
==> svn update
```

Which results in:

```
$ svn update
G    myscript.sh
Updated to revision 8.
```

The G indicates that changes in our version and in the repository (i.e., Joe's changes) were automatically merged together. Look at myscript.sh and you will see the exit 0 at the end you added, and the exit 1 Joe added. If your changes had overlapped the changes Joe made, then svn status would have returned:

```
$ svn update
C    myscript.sh
Updated to revision 8.
```

The C means the files have conflicting changes and could not be automatically merged. A file listing would show some extra files svn creates to help in manually merging any needed changes:

```
$ ls myscript*
myscript.sh*  myscript.sh.mine*  myscript.sh.r7  myscript.sh.r8
```

See the SVN book for information on how to use these files and what needs to be done to resolve the conflict.

The most important thing to remember when using svn update, is to always watch for files flagged G or C. These files must be manually reviewed before committing your changes.

Review the automatic merge and commit:

```
==> svn diff myscript.sh
==> svn commit -m "Added exit code for success"
```

The rest of this part of the tutorial will demonstrate the most common way svn update is used.

Joe has emailed you to ask how to run the program. He has run it, but still gets the message about the Resource file missing. So now you have to add the documentation. Perform the following steps on your own (see the log if you still need help with the svn commands):

1. Create a doc directory
2. Create doc/README.txt with instructions that in order to run, to copy config/myApp.config to the main myApp directory, and edit myApp.config to set your site.
3. Add doc and doc/README.txt to Subversion.
4. Commit the changes.

Now we will pretend we are Joe again:

```
==> cd $HOME/svnwork/joe/myApp
```

Joe knows you have been working on the application, so the first thing he does is svn status and sees there are updates to the repository and he promptly does an update:

```
==> svn status -uv
==> svn update
```

In actual use, there is no need to do the status first (in this case, I'm just demonstrating how to check if your working copy is out of date). It should become a habit to always perform an svn update before starting to make changes.

Joe reads the documentation, copies the config file, changes the value of site, and tests running the program:

```
==> cp config/myApp.config .
==> vi myApp.config
==> ./myscript.sh
```

Joe's test works OK. He does svn status once more and now sees his config file is not under Subversion control (i.e., marked with "?"):

```
==> svn status -uv
                9        8 paul         myscript.sh
                9        6 paul         config/myApp.config
                9        6 paul         config
                9        9 paul         doc/README.txt
                9        9 paul         doc
?                                       myApp.config
                9        9 paul         .
Status against revision:      9
```

Is this a problem and should Joe add myApp.config to the repository? The answer to both is no. It is absolutely OK to have extra files in your working copy. This would normally be for testing (as in this case) or other development tools. For the myApp.config file, if Joe added it, then anyone else who checked out the code would have Joe's file overwrite their file.

This completes Part 3 of the tutorial. Please leave the working copy and repository as is in your home directory for Part 3.