CollabNet Subversion 1.9 for Developers

Enterprise Lab Exercises

TortoiseSVN

COLLABNET

4000 Shoreline Court, Suite 300
South San Francisco, California 94080 U.S.A.

888.778.9793 toll free
650.228.2500 voice
650.228.2501 fax
www.collab.net
E-mail info@collab.net

Table of Contents

1. Lab 1 – Essential Concepts 1

1.1. Standard Work Cycle – Part 1

Steps	Comments
1. Create 2 working copies based on the repository's trunk in directories named *JoeDev* and *JillDev*	
2. Using *JoeDev*, create and add a new directory under the directory *EasySVN* by the name *FirstLessons* and a file within it named First.txt	Structural change
3. Delete an existing file called *DeleteMe*	Structural change
4. Rename the existing file RenameMe.txt to *ChangedName.txt*	Structural change
5. Choose an existing file EditMe.txt and change *"anything"* to *"something"* on the 2nd line and save your changes	Content change
6. Use Subversion to examine all your changes done from step 2 to step 5	(High level checking)
7. Choose the file *EditMe.txt* whose content was changed and compare with the previous revision	Examine changes (Low level checking)
8. Undo the rename you did earlier	Reverting changes
9. Bring in changes made and committed by others	Potential content change
10. Resolve any conflict detected	Resolve conflicts
11. Commit your changes to the repository	Commit changes
12. Bring in changes made and committed by others	Remove mixed revisions state

1.2. Standard Work Cycle – Part 2

Steps	Comments
1. Update the *JillDev* working copy	Content change
2. Create and add a new directory under *EasySVN* by name *SecondLessons* and add a file within it named Second.txt	Structural change
3. Move the directory *MoveMe* to the directory *AddMe*	Structural change
4. Get a lock on the existing file *EditMe.txt*.	Lock
5. Edit the existing file *EditMe.txt* and change *"line"* to *"sentence"* on the 2nd line.	Content change
6. Examine all your changes done from step 2 to 4	(High level checking)
7. Bring in changes made and committed by others	Same as step 1
8. Resolve any conflict detected	Resolve conflicts
9. Commit your changes to the repository	Commit changes
10. Bring in changes made and committed by others	Remove mixed revision state

1.3. Examine History

Steps	Comments
1. Using *JillDev*, show the history for the trunk	
2. Show the history for the specific file – *EditMe.txt*	
3. Check the difference between any two sequential revisions of the file, *EditMe.txt*, which was modified in Exercises 1.1 and 1.2	
4. Check *EditMe.txt* to find who last changed line number 3	

1.4. Property Changes

Steps	Comments
1. Using JillDev, add a property to enforce locking on the image file, “Subversion Logo.jpg”	Adding a property
2. Check that the Read only attribute is not set on the file in the Windows properties	
3. Commit your changes	Committing a property
4. Update your working copy	
5. Check that the Read only attribute is now set on the file in the Windows properties	
6. Get a lock on the image file, “Subversion Logo.jpg”	
7. Check that the Read only attribute is not set on the file in the Windows properties	
8. Release the lock	

2. Lab 2 – Essential Concepts 2

2.1. Traversing

Steps	Comments
1. Using JoeDev, update EditMe.txt to revision 2	Moving back in time
2. Note that the update records a modification made to that file. Examine the file to see that your changes are gone	
3. Update the directory EasySVN to revision 1	
4. Note the changes recorded by the update	
5. Update your working copy to point to the HEAD on the trunk	

2.2. Tree Conflict

Steps	Comments
1. Using *JillDev*, update your working copy to point to the HEAD on the trunk.	Notice the speed of creating a branch
2. Using *JoeDev*, move the directory *SecondLessons* to the directory *AddMe* changes are gone	
3. Commit your changes	
4. Using *JillDev*, edit the file *Second.txt* in the directory *SecondLessons* and enter the text “Tree conflict looming”	
5. Update your working copy	Notice that you have a tree conflict
6. Use a Subversion operation to identify the type of tree conflict	
7. Fix the conflict by getting your edited content of the file *Second.txt* into that file in its new location.	

8.	Inform Subversion that you resolved the conflict	
9.	Commit your changes.	

2.3. Merging

Steps		Comments
1.	Using *JoeDe*v, create a branch *"Demo"* (from HEAD of the trunk) in the branches subdirectory. Choose to stay on the trunk	Notice the speed of creating a branch
2.	Update your working copy	
3.	Modify *EditMe.txt* on the trunk. Change the last word from *"Subversion"* to *"SVN"*	
4.	Commit your changes	
5.	Switch your working copy to the newly created branch, *Demo*	Notice that the only change is the modified file
6.	Modify *EditMe.txt* changing the last word from *'Subversion"* to *"Smart Subversion"*	
7.	Commit your changes.	
8.	Update your working copy	
9.	Perform a merge between the trunk and the branch. Choose to resolve the resulting conflict by accepting the trunk revision.	Interactive conflict resolution.
10.	Commit your changes.	

3. Lab 3 – Enterprise Features

3.1. Advanced Merge – Manual Merge

Steps	Comments
1. Using *JillDev*, create a new branch by the name of *Feature1* but keep your working copy referencing the trunk	Creating a branch for a new feature
2. Edit *First.txt* and change its contents however you wish	
3. Commit your changes	
4. Switch to the branch, *Feature1*	
5. Execute a manual merge between the trunk and the branch	
6. Examine *First.txt* to see that the changes were not made	
7. Commit	
8. Examine the mergeinfo data on the top directory	

3.2. Sparse Checkouts

Steps	Comments
1. Create a new working copy by checking out only the files beneath the trunk	Creating a branch for a new feature
2. Update the root directory to get all files and directories, but nothing below that	
3. Update to get the full structure beneath of the *Background* folder	
4. Add the full structure of the SecondLessons folder	

4. Solutions for Lab 1 – Essential Concepts 1

4.1. Suggested Solution for Standard Work Cycle – Part 1

Steps	Comments
1. Use SVN Checkout to create each working copy 2. Specify the URL of the repository you to checkout from and name the top level directory as directed 3. You may be prompted for authentication	• SVN checkout fetches all the contents from the specified URL in to the local area. An example of URL is http://localhost/svn/LearningLab/trunk • Use the defaults in the dialog
4. Navigate to the *EasySVN* folder in the JoeDev working copy, right click to create new folder, *FirstLessons*, using the Windows New command 5. Right click in the *FirstLessons* folder and use the Windows New command to create a new text document by the name *First.txt* 6. Right click on *FirstLessons* and click Add under TortoiseSVN	Note that First.txt is also listed and selected, click on OK
7. Select *DeleteMe.txt* in the working copy and click on Delete under Tortoise SVN	Do not use Windows Delete as Windows doesn't allow for as tight an integration with Subversion as experienced with Eclipse for example. Structural changes in Windows do not get reflected in Subversion
8. Select *RenameMe.txt* and choose Rename under TortoiseSVN to rename the file to *ChangedName.txt*	Do not use Windows Rename for the same reasons as discussed above for delete
9. Choose the file *EditMe.txt* and edit it using your preferred editor to change "anything" to "something" on line 2 and save the file	Content change
10. At the top of your working copy, click on "Check for modifications" under Tortoise SVN to see all the changes you have made so far	High level checking
11. Choose the file *EditMe.txt* and click on Diff to see the difference between what you checked out and what you've now changed	The result will display two windows one with the current revision and the other with the previous revision with both highlighting the differences/changes between the two

12. Go to the parent directory of the file *ChangedName.txt* and click on Revert under TortoiseSVN 13. Select *ChangedName.txt* and *RenameMe.txt* before clicking on OK 14. You should also probably go ahead and delete the now unversioned *ChangedName.txt* file	• Remember a Rename or copy is implemented as a combination of a delete operation and an add operation • The Revert command will display all changes not yet committed to the repository, choose the Add and Delete options corresponding to the rename • You can choose Revert to undo any operation which has not been committed
15. Select the top of the working copy and click on “SVN Update”	This operation will report the changes made to your working copy after updating it with the latest changes from the repository
16. At this point there are no conflicts, if there were the steps would be the same as defined for the merge process later	
17. Select the top of the working copy and click on “SVN Commit” to commit all the changes done up until now 18. Enter a useful and concise log message before hitting OK	• The commit will show all the changes that can be committed to the repository • If there are any files which have been left out they could be selected from the unversioned files to be added • Note the incremented global revision number after a successful commit
19. Select the top of the working copy and click on “SVN Update”	This is to ensure that your working copy does not have mixed revisions.

4.2. Suggested Solution for Standard Work Cycle – Part 2

Steps	Comments
1. Select the top of the working copy and click on “SVN Update”	You should always begin work with the latest revision from the repository
2. Right click on the *EasySVN* folder to create a new folder, *SecondLessons*, using the Windows New command 3. Right click on *SecondLessons* to create a new text document, *Second.txt* 4. Right click on *SecondLessons* and click Add under TortoiseSVN	
5. Select the *MoveMe* directory, right click and hold, then drag and drop it on the *AddMe* directory 6. Select the “SVN move versioned files here” option	Upon dropping it at the destination you will get various options
7. Select the file *EditMe.txt* and click on “Get Lock” under TortoiseSVN 8. Enter a reason for getting the lock	Locking should normally be used for file types that can’t be merged, but it may be used for other purposes
9. Choose the file “EditMe.txt” and edit it using your preferred editor to change “line” to “sentence” on the 2nd line and save the file	
10. At the top of your working copy, click on “Check for modifications” under TortoiseSVN to see all the changes you have made Tortoise SVN to see all the changes you have made so far	High level checking
11. Select the top of the working copy and click on “SVN Update”	
12. Now there are no conflicts, if there were the same process would be used as defined for merge later	

13. Select the top of the working copy and click on “SVN Commit” to commit all the changes	• The commit will show all changes that can be committed • If there are any files which have been left out they could be selected from the unversioned files to be added • Enter a useful and concise log message before hitting OK • Note the incremented global revision number after a successful commit
14. Select the top of the working copy and click on “SVN Update”	This is to ensure that your working copy does not have mixed revisions

4.3. Suggested Solution for Examine History

Steps	Comments
1. Select the top of the working copy and click on “Show log” under TortoiseSVN	The entire trunk directory structure’s (any revision where something changed) history will be displayed
2. Click on *EditMe.txt* and select “Show log” under TortoiseSVN	Show log will display only the history of the selected file
3. Select *EditMe.txt*, right click and select “Diff with previous revision” under TortoiseSVN	Diff will show the modifications made with respect to the two previous revisions
4. Click the file *EditMe.txt* and right click to select Blame under TortoiseSVN	Blame (or annotation) shows what revision and author last changed each line of a text file

4.4. Suggested Solution for Property Changes

Steps	Comments
1. Right click on the file Subversionlogo.jpg and choose Properties under TortoiseSVN 2. Click on new to add a new property 3. Choose svn : needs-lock from the drop down and click OK to set the property 4. Click on OK to apply the property	• After you click OK a new property is added to the file type displaying svn:needslock with a value of * • Alternatively, you select Windows Properties and the Subversion tab • The properties button will take you to the same dialog as choosing Properties under TortoiseSVN
5. Right click on the file and check the Windows properties	The read only attribute will not be set at this point, since the property is not yet committed
6. Commit the property by choosing SVN Commit	• The commit will show all changes that can be committed • Enter a useful and concise log message before hitting OK • Note the incremented global revision number after the commit
7. Select the top of the working copy and click on “SVN Update”	This ensures that your working copy does not have mixed revisions
8. Right click on the file and check the Windows properties. The read only attribute is set since the property has been committed	
9. Select the file SubversionLogo.jpg and click on “Get Lock” under TortoiseSVN 10. Enter a reason for getting the lock	This informs the Subversion server that you reserve the right to create the next revision of this file on the trunk
11. Right click on the file and check the Windows properties. The read only attribute is no longer set since the lock has been obtained	
12. Select the file SubversionLogo.jpg and click on “Release Lock” under TortoiseSVN	

Solutions for Lab 2 – Essential Concepts 2

4.5. Suggested Solution for Traversing

Steps	Comments
1. Make sure your working copy is pointing to the trunk of JoeDev by selecting the top level directory and clicking on Windows Properties. The Subversion tab should show the URL as ending with trunk 2. Select EditMe.txt and click “Show log” under TortoiseSVN 3. Select Revision 2 and right click to choose “Update item to revision”	• SVN checkout fetches all the contents from the specified URL in to the local area. An example of URL is http://localhost/svn/LearningLab/trunk • Use the defaults in the dialog
4. Check the contents of the file (via edit) to observe that the changes you made after revision 2 are not included	Note that First.txt is also listed and selected, click on OK
5. Navigate to the directory EasySVN, click “Update to revision …” under TortoiseSVN and enter revision 1	Do not use Windows Delete as Windows doesn’t allow for as tight an integration with Subversion as experienced with Eclipse for example. Structural changes in Windows do not get reflected in Subversion
6. Observe the changes listed by the update operation’s output	Do not use Windows Rename for the same reasons as discussed above for delete
7. Select the root of the working copy and click SVN Update	Content change

4.6. Suggested Solution for Tree Conflict

Steps	Comments
1. Make sure your working copy JillDev is pointing to the trunk by selecting the top level directory and clicking on Windows Properties. The Subversion tab should show the URL as ending with trunk 2. Select the root of the working copy and click SVN Update	
3. In your JoeDev working copy select the *SecondLessons* directory, right click and hold, then drag and drop it on the *AddMe* directory	
4. Select the top of the working copy and click on “SVN Commit” to commit the change	
5. In your JillDev working copy (do not update) select the *Second.txt* file and double click on it to open it up. 6. Type in “Tree conflict looming” and save the file.	
7. Select the root of the working copy and click SVN Update	
8. Select the top of the working copy and click on “Check for Modifications”. 9. Select and click on the directory with the tree conflict and select Edit Conflicts to get more information about your tree conflict, but select Cancel rather than either resolution. 10. Edit either by copying the file under the old directory on top of the one in the new directory or by manually typing in the same line.	

11. Select the top of the working copy and click on “Check for Modifications”. 12. Select and click on the directory with the tree conflict and select Edit Conflicts and select “Remove the Directory”. 13. Refresh and you’ll see that resolved the conflict.	Resolution of tree conflicts isn’t fully automated as in this case you need to make a manual fix before utilizing the automated step provided for this type of tree conflict.
14. Commit your changes to the repository using SVN Commit	

4.7. Suggested Solution for Merging

Steps	Comments
1. Make sure your working copy JoeDev is pointing to the trunk by selecting the top level directory and clicking on Windows Properties. The Subversion tab should show the URL as ending with trunk 2. Click on Branch/Tag under TortoiseSVN 3. Create a branch named Demo under the branches directory, but choose to remain on the trunk after creating it	Option of switching to the new branch / tag is available. However, to remain on the trunk do not check the checkbox saying “ Switch to branch / tag”
4. Modify the *EditMe.txt* file on the trunk by changing the last word from “Subversion” to “SVN”	
5. Select the top of the working copy and click on “SVN Commit” to commit the change	
6. Use Switch under TortoiseSVN to switch to the Demo branch	Note that the only reported action is an update of the EditMe.txt file
7. Modify *EditMe.txt* file by adding the word “Smart” before the last word “Subversion”	
8. Select the top of the working copy and click on “SVN Commit” to commit the change	

9. Select the root of the working copy and click SVN Update	
10. Choose the Merge operation under TortoiseSVN 11. Of the 3 options of Merge presented select the first one “Merge a range of revisions” 12. Set the “URL to merge from” to be the trunk 13. On clicking “Next” you will be prompted to choose the merge options. Keep the default options in this case 14. Click the “Test Merge” to preview the result of the merge. 15. If the results of the test merge is as per your expectation click on the “Merge” 16. There will be a conflict reported here since the same line has been changed so choose to resolve the conflict now. 17. The conflict editor will show 3 different windows. Choose the version coming from the trunk to be reflected in the working copy. 18. Choose “Conflict resolved” after conflict has been resolved and continue with the Merge	• The trunk changes have to be brought into the branch. • This is a precautionary step to be sure that your setting are right and the impact is known before you do the actual merge
19. Commit the merge to the repository using SVN Commit	

5. Solutions for Lab 3 – Enterprise Features

5.1. Suggested Solution for Advanced Merge

Steps	Comments
1. Make sure your working copy JillDev is pointing to the trunk by selecting the top level directory and clicking on Windows Properties. The Subversion tab should show the URL as ending with trunk 2. Click on Branch/Tag under TortoiseSVN 3. Create a branch named *Feature1* under the branches directory, but choose to remain on the trunk after creating it	
4. Edit *First.txt* however you wish	
5. Commit the changes to the repository using SVN Commit	
6. Use Switch under TortoiseSVN to switch to the Feature1 branch	
7. Choose the Merge operation under TortoiseSVN 8. Of the 3 options of Merge presented select the first one “Merge a range of revisions” 9. Set the “URL to merge from” to be the trunk 10. On clicking “Next” you will be prompted to choose the merge options. Select to “Only record the merge” and click on Merge 11. Click on OK in the results dialog	
12. Examine *First.txt* to see that the changes from the trunk were not applied	
13. Commit the merge to the repository using SVN Commit	
14. Click on the top level directory and choose Properties under TortoiseSVN	Note the value of the svn:mergeinfo property showing the manual merge

5.2. Suggested Solution for Sparse Checkouts

Steps	Comments
1. Using SVN Checkout create a new working copy from either the trunk or one of the branches. 2. In the checkout option choose Checkout Depth as “Only File children”	• This option specifies to selectively checkout only the files beneath this directory (in this case trunk) • Verify that you only get your new working copy directory with a .svn directory
3. At the top of the working copy, click on “Update to revision …” under TortoiseSVN 4. Select “Immediate children, including folders” under “Update Depth”	This option specifies to selectively include files at the top level of the branch and empty (excluding the .svn directories) folders beneath Verify that you got the expected results
5. Select the *Background* folder and click on “update to revision …” under TortoiseSVN 6. Select “Fully recursive” under “Update Depth”	• This option specifies to include the full file and folder structure beneath this folder • Verify that you got the expected results
7. From the working copy, click and select Repo-browser under TortoiseSVN 8. Navigate to the *SecondLessons* folder under the trunk, click and select “Update item to revision” 9. Select “Fully recursive” under “Update Depth” and click on OK	• This option specifies to add this folder and the full file and folder structure beneath it • Verify that you got the expected results