CollabNet Subversion 1.9 for Developers

Enterprise Lab Exercises

Eclipse

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

Setting up the Lab

Steps	Comments
1. Make sure a repository has been created or get an empty one created	Check with the instructor for a path to your specific repository
2. Run the script svnlabprep.bat giving it the path to your repository	This script will load the repository with directories and files for these exercises

1. Lab 1 – Essential Concepts 1

1.1. Standard Work Cycle – Part 1

Steps	Comments
1. Create 2 projects based on the repository's trunk named *JoeDev* and *JillDev*	
2. Using *JoeDev*, create and add a new folder under the folder *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 change	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 it 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* project	Content change
2. Create and add a new folder under *EasySVN* by name *SecondLessons* and add a file within it named *Second.txt*	Structural change
3. Move the folder *MoveMe* to the folder *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 and save the file	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 OS properties	
3.	Commit your change	Committing a property
4.	Update your project	
5.	Check that the Read only attribute is now set on the file in the OS 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 OS 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.	Examine the file to see that your changes are gone	
3.	Update the folder *EasySVN* to revision 1	
4.	Note the changes made by the update	
5.	Update your project to point to the HEAD on the trunk	

2.2. Merging

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

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 project 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. Record/execute a manual merge between the trunk and the branch	
6. Commit	
7. Examine *First.txt* to see that the changes were not made	
8. Examine the mergeinfo data on the *JillDev* folder	

3.2. Sparse Checkouts

Steps	Comments
1. Create a new project by checking out only the files beneath the trunk	
2. Update the project folder to get all files and folders, 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	Explanation
1. Assuming you have an Eclipse workspace, use the navigator view, right click in that panel and select Project under New Under SVN select "Checkout Projects from SVN" to create each project and click on Next Select "Create a new repository location" and click on Next Enter the URL that you have been given and click on Next Select trunk and click on Next Select "Check out as a project in the workspace", enter the name of JoeDev and click on Finish You may be prompted for authentication Repeat the process to create JillDEV, but select the existing repository location (and URL) you entered previously rather than creating a new one.	Behind the scenes, the SVN checkout fetches all the contents from the specified URL into the local project. An example of a URL is http://localhost/svn/LearningLab/trunk Use the defaults in the dialog except where told otherwise
2. Navigate to the *EasySVN* folder in the project, and create new folder, *FirstLessons*, using the Eclipse New command Right click on *FirstLessons* folder and use the Eclipse New command to create a file, *First.txt* Right click on *FirstLessons* and click "Add to version control" under Team	Note that *First.txt* will also be added
3. Select *DeleteMe.txt* in the working copy and click on Delete	Eclipse allows for a tight integration with Subversion so that structural changes done from Eclipse are reflected in Subversion
4. Select *RenameMe.txt* and choose Rename to rename the file to *ChangedName.txt*	You use the normal Eclipse Rename for the same reason as discussed above for delete
5. Double click on the file *EditMe.txt* to edit it changing "anything" on line 2 to "something" and save the file	Content change

6.	At the top of your project, click on “Synchronize with Repository” under Team to see all the changes you have made so far	High level checking
7.	Within the file *EditMe.tx*t, click on “Compare with” and select “Base Revision” to see the difference between what you checked out and what you’ve now changed Close the comparison window	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
8.	Click on the file *ChangedName.txt* and select Revert Click on OK You should also probably go ahead and delete the now unversioned *ChangedName.txt* file	Remember a Rename/Copy is implemented as a combination of a delete and an add operation though that is hidden to you in Eclipse The Revert command will display only the new filename, but it will revert both operations You can choose Revert to undo any operation which has not been committed
9.	Under Window choose to Close Perspective Select the top of the project and click on Update under Team	This operation will report the changes made to your project after updating it with the latest changes from the repository
10.	At this point there are no conflicts, if there were the steps would be the same as defined later for merge	
11.	Select the folder *JillDev* and click on Commit under Team to commit all the changes done up until now Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 1.1) 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
12.	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	Explanation
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 and create a new folder, *SecondLessons*, using the Eclipse New command Right click on *SecondLessons* and create a new file, *Second.txt* Right click on *SecondLessons* and click Add under Team	
3. Select the *MoveMe* folder, right click on Move Select the AddMe folder as the destination and click OK	
4. Select the file *EditMe.txt* and click on Lock under Team Enter a reason for getting the lock and click on OK	Locking should normally be used for file types that can't be merged, but it may be used for other purposes
5. Open the file "EditMe.txt" and edit it to change "line" to "sentence" on the 2nd line and save the file	
6. At the top of your project, click on "Synchronize with Repository" under Team to see all the changes you have made so far	High level checking
7. Under Window choose to Close Perspective. Select the top of the project and click on Update under Team	
8. At this point there are no conflicts, if there were the steps would be the same as defined for the merge process later	
9. Select the top of the project and click on Commit under Team to commit all the changes done up until now Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 1.2) before hitting OK	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 Note the incremented global revision number after a successful commit
10. Select the top of the project and click on Update under Team	This is to ensure that your project does not have mixed revisions

4.3. Suggested Solution for Examine History

Steps	Explanation
1. Select the top of the project and click on "Show History" under Team	The entire trunk folder structure's (any revision where something changed) history will be displayed
2. Click on *EditMe.txt* and select "Show History" under Team	Show log will display only the history of the selected file
3. Within *EditMe.txt*, right click and select "Revision …" under "Compare With" Select the previous revision which is likely revision 2 and double click Close the comparison	Diff will show the modifications made in the project with respect to the two previous revisions
4. Select the file *EditMe.txt* and right click to select "Show Annotation" under Team Close the annotation window	Annotation (or blame) shows what revision and author last changed each line of a text file

4.4. Suggested Solution for Property Changes

Steps	Explanation
1. Right click on the file *Subversionlogo.jpg* and choose "Set property …" under Team Choose svn : needs-lock from the drop down and click OK to set the property	After you click OK a new property is added to the file type displaying svn:needslock with a value of *
2. Right click on the file and select Properties to check the Windows properties under Resource	The read only attribute will not be set at this point, since the property is not yet committed
3. Select the file and click on Commit under Team Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 1.4) before hitting OK	Note the incremented global revision number after a successful commit
4. Select the top of the project and click on Update under Team	This ensures that your project does not have mixed revisions
5. Right click on the file and select Properties to check the Windows properties. Under resource, the read only attribute is set since the property has been committed	
6. Select the file *SubversionLogo.jpg* and click on Lock under Team	This informs the Subversion server that you reserve the right to create the next revision of this file on the trunk

	Enter a reason for getting the lock and click on OK	
7.	Right click on the file and select Properties to check the Windows properties. Under resource, the read only attribute is no longer set since the lock has been obtained	
8.	Select the file *SubversionLogo.jpg* and click on Unlock under Team	

5. Solutions for Lab 2 – Essential Concepts 2

5.1. Suggested Solution for Traversing

Steps		Explanation
1.	Make sure your JoeDev project is pointing to the trunk by noting what is in parentheses after that folder name Select *EditMe.txt* and click "Switch to another Branch/Tag/Revision …" under Team Unselect "Switch to HEAD revision" and then enter Revision 2 before clicking on OK	
2.	Open the file and check the contents to observe that the changes you made after revision 2 are not included	
3.	Select *EasySVN* and click "Switch to another Branch/Tag/Revision …" under Team Unselect "Switch to HEAD revision" and then enter Revision 1 before clicking on OK	
4.	Observe the changes in the folder's contents	
5.	Select the JoeDev project and click "Switch to another Branch/Tag/Revision …" under Team Accept the default to "Switch to HEAD revision" and click on OK	

5.2. Suggested Solution for Merging

Steps	Explanation
1. Right click on “Branch/Tag …” under Team Create a branch named Demo under the branches directory and click on Next Click on Next in the following window Enter a useful commit message (perhaps noting this is for lab 2.2 part 1) and click on Finish	Option of switching to the new branch / tag is available. However, to remain on the trunk do not check the checkbox saying “ Switch working copy to new branch/tag”
2. Edit the *EditMe.txt* file (on the trunk) by changing the last word from “Subversion” to “SVN” and saving the file	
3. Select the top of the project and click on Commit under Team to commit this change Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 2.2 part 2) before hitting OK	
4. Use Switch to another “Branch/Tag /Revision …” under Team to switch to the Demo branch Note the change is reflected in the path in parentheses after JoeDev	Note that the only reported action is an update of the *EditMe.txt* file
5. Edit the *EditMe.txt* file (on the Demo branch) by adding the word “Smart” before the last word “Subversion” and saving the file	
6. Select the top of the project and click on Commit under Team to commit this change Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 2.2 part 3) before hitting OK	
7. Select the top of the project and click on Update under Team	
8. Select the top of the project and click on Merge under Team Of the 6 options of Merge presented select the first one “Merge a range of revisions” and click on Next “Merge from” should already be set to the trunk so click on Finish	The trunk changes have to be brought into the branch.

There will be a conflict reported here since the same line has been changed so choose to launch a graphical conflict resolution editor

The conflict resolution editor will show 2 different windows. Select “Next Change” (the left window should have “Smart Subversion” highlighted and the right window should have “SVN” highlighted.

Choose to “Copy the Current Change from Right to Left” either from the tool bar in the editor window or right clicking in the center column

Choose “Yes. I resolved all of the conflicts in the file. Mark conflict resolved” and click on OK

Note you’ll get a dialog showing the merge results summary with just 1 Updated and 1 “Resolved conflicts” noted click on OK

---

9. Select the top of the project and click on Commit under Team to commit this change

   Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 2.2 part 4) before hitting OK

6. Solutions for Lab 3 – Enterprise Features

6.1. Suggested Solution for Advanced Merge

Steps	Explanation
1. Right click on “Branch/Tag …” under Team Create a branch named Feature1 under the branches directory and click on Next Click on Next in the following window Enter a useful commit message (perhaps noting this is for lab 3.1 part 1) and click on Finish	
2. Edit *First.txt* however you wish and save your changes	
3. Select the top of the project and click on Commit under Team to commit this change Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 3.1 part 2) before hitting OK	
4. Use Switch to another “Branch/Tag /Revision …” under Team to switch to the Feature1 branch Note the change is reflected in the path in parentheses after JillDev	
5. Select the top of the project and click on Merge under Team Of the 6 options of Merge presented select the fifth one “Manually record merge information (block one or more revisions)” and click on Next “Merge from” should already be set to the trunk so click on Next Choose the only revision provided in the list and click on Finish Click on Yes in the “Merge Operations Completed” dialog	
6. Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 3.1 part 3) before hitting OK	

7.	Examine *First.txt* to see that changes from the trunk were not applied	
8.	Click on JillDev and choose “Show Properties” under Team	Note the value of the svn:mergeinfo property showing the manual merge

6.2. Suggested Solution for Sparse Checkouts

Steps		Explanation
1.	Create a new project from either the trunk or one of the branches. On the Checkout from SVN “Checkout As” dialog choose Depth (at the bottom of the dialog) as “Only File children”	This option specifies to selectively checkout only the files beneath this folder (in this case trunk) Verify that you only get your new project with a .project file
2.	At the top of the project, click on “Switch to another Branch/Tag/ Revision …” under Team Select “Immediate children, including folders” under Depth and click on “Change working copy to specified 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 by confirming the presence of the EasySVN folder
3.	Select the *EasySVN* folder, click on “Switch to another Branch/Tag/ Revision …” under Team Select “Immediate children, including folders” under Depth and click on “Change working copy to specified depth” Select the *Background* folder, click on “Switch to another Branch/Tag/ Revision …” under Team Select “Fully recursive” under Depth and click on “Change working copy to specified Depth”	This option specifies to include the full file and folder structure beneath this folder Verify that you got the expected results
4.	Select the *SecondLessons* folder, click on “Switch to another Branch/Tag/ Revision …” under Team Select “Fully recursive” under Depth and click on “Change working copy to specified Depth”	This option specifies to add this folder and the full file and folder structure beneath it Verify that you got the expected results