CollabNet Subversion 1.9 for Developers

Enterprise Lab Exercises

AnkhSVN

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. AnkhSVN will not allow you to create a local repository for this lab. You will need to install a client integration that will allow you to create a local repository.	TortoiseSVN is a good application to install to complete this lab.
2. Create a local repository and import the lab source files into that repository.	This can be performed via another client tool, such as TortoiseSVN or Command line.
3. Obtain the newly created repository URL address, which will be used later. Also, verify that the Subdirectories branches, tags, and trunk are created.	The TortoiseSVN Repository Browser for this newly created repository will identify the correct URL address, and repository structure.
4. Launch Visual Studio and verify AnkhSVN is integrated.	Using the help menu option under “About Microsoft Visual Studio” confirm one of the install projects is AnkhSVN.
5. Bring up the Repository Explorer view and add the URL obtained in step 3.	Use the view menu option.
6. Select the trunk folder and use the icon to checkout from Subversion. Create two new folders called JoeDev, and JillDev.	This will create a working copy for the explicit use of AnkhSVN project JoeDev and JillDev. If you use a commonly used working copy instead you will have a lot of miscellaneous files in that location.
7. Use the Working Copy Explorer view to confirm the working copies are setup correctly.	Use the View menu option and select working Copy Explorer.
8. Create a new project from existing code for each working copy (JoeDev and JillDev) and commit all pending changes. Exclude the .svn folder via the Solution Explorer view, and then save all via the file menu option.	For this lab use the Visual C# project type and use JoeDevProject, and JillDevProject for the project names. Each solution will have one project in it. You may receive a warning concerning the .svn files should not be included, which is fine.
9. Close each project and then reopen each to allow each project to be easily switched using the start page.	The start page view is normal default, if it is not available use the view menu option to select it.
10. Select each project in the solution Explorer and ensure the show all hidden icon is engaged.	Repository changes will not automatically be visible if you switch between solutions after a Subversion commit.
11. During the lab if you switch from one project to the other, ensure you update and then refresh the view in the Solution Explorer. If you see a blank file that should be included in the project select Include in project option.	It is imperative that you refresh and then include all newly created files and folders. Also, if a file has been deleted you will see a question mark after an update. You will then have the option to delete this manual when you desire to do so. This is not an automatic action.

1. Lab 1 – Essential Concepts 1

1.1. Standard Work Cycle – Part 1

Steps	Comments
1. Create a lab repository, import the lab source files into that repository, and create two working copies named *JoeDev* and *JillDev.*	Initial setup requires using a tool that allows you to create a repository before you can recreate the solutions.
2. Create a solutions with one project in each based on the repository's trunk named JoeDevProject.	This can be done using TortoiseSVN or in Visual Studio.
3. Create a solutions with one project in each based on the repository's trunk named JillDevProject.	This can be done using TortoiseSVN or in Visual Studio.
4. Set up the Start Page so you can easily switch between projects.	Using the start page with both projects listed will make it easier to switch back and forth.
5. 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.
6. Delete an existing file called *DeleteMe.*	Structural change.
7. Rename the existing file *RenameMe.txt* to *ChangedName.txt.*	Structural change.
8. Choose an existing file *EditMe.txt* and change "*anything*" to "*something*" on the 2nd line and save your change.	Content change.
9. Use Subversion to examine all your changes done from step 2 to step 5.	(High level checking).
10. Choose the file *EditMe.txt* whose content was changed and compare it with the previous revision.	Examine changes (Low level checking).
11. Undo the rename you did earlier.	Reverting changes.
12. Bring in changes made and committed by others.	Potential content change.
13. Resolve any conflict detected.	Resolve conflicts.
14. Commit your changes to the repository.	Commit changes.
15. 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.	

4. Solutions for Lab 1 – Essential Concepts 1

4.1. Suggested Solution for Standard Work Cycle – Part 1

Steps	Explanation
1. Use Command line or another client integration such as TortoiseSVN to create a local repository.	AnkhSVN does not have the functionality to create a repository for this lab. We recommend using TortoiseSVN.
Import the lab source directory labeled "src" into the newly created local repository.	There should be three subdirectories listed named branches, tags, and trunk. An example of a URL is http://localhost/svn/LearningLab/trunk.
Bring up the repository browser for the newly created repository copy the URL and confirm the repository structure.	
Create a directory locally for AnkhSVN working copies.	AnkhSVN will generate many additional solution and project files into any working copy it will use as a source for creating a project. Isolating this by creating a working copy use for AnkhSVN makes it easier to manage.
Create a subdirectory under your working copies directory called JoeDev, and create another subdirectory under your working copies directory called JillDev.	
Perform a checkout on each working copy directory (JoeDev and JillDev) of the newly create repository subdirectory called trunk.	Behind the scenes, the SVN checkout fetches all the contents from the specified URL into the local project.
Launch Visual Studio and bring up the repository Explorer via the view menu option.	
Click the "Add a new URL to the Repository Explorer" icon, and enter the URL address obtained in step 3.	
Highlight the repository Trunk subdirectory then click the "Checkout from Subversion" icon to perform a checkout.	
Add the URL address obtained in step 3 in the "Check Out URL" section of the popup.	

Ensure the “To path:” is the JoeDev working copy directory created in steps 5 then repeat for JillDev working copy created in step 6.	
2. Create a Solution for the JoeDev project.	The Visual Studio solution will hold the work space information for each project.
Use the File menu -> New -> Project From Existing Code options.	
Select the project type Visual C# in the Project from Existing Code Files Wizard.	
Use the three ellipse icon and navigate to the working copy directory.	
Highlight the JoeDev working copy subdirectory and then click ok.	
Click the “Include subfolders” box	
Enter JoeDevProject into the Name: section.	
Use the defaulted Output type of “Console Application”. Click OK on any warning messages.	You will get a Microsoft Visual Studio warning indicating that the .svn directory is a Subversion administrative area and should not be added. This is expected and not a problem.
Click the Commit button in “Pending Changes Panel” in the lower left corner of Visual Studio.	
Right click the .svn subdirectory in the Solution Explorer Panel in the upper right corner and select “Exclude From Project” option.	This file should not be monitored by Subversion since it is just an administrative directory.
Click the Commit button in “Pending Changes Panel” in the lower left corner of Visual Studio.	AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option.
Highlight the file JoeDevProject in the Solution Explorer panel in the upper right corner.	
Click the Show all Files icon and then the refresh icon	
Use the File menu and Save all option.	

Click the Commit button in “Pending Changes Panel” in the lower left corner.	
3. Create a Solution for the JillDev project. Use the File menu -> New -> Project From Existing Code options. Select the project type Visual C# in the Project from Existing Code Files Wizard. Use the three ellipse icon and navigate to the working copy directory. Highlight the JillDev working copy subdirectory and then click ok. Click the “Include subfolders” box Enter JillDevProject into the Name: section. Use the defaulted Output type of “Console Application”. Click OK on any warning messages. Click the Commit button in “Pending Changes Panel” in the lower left corner of Visual Studio. Right click the .svn subdirectory in the Solution Explorer Panel in the upper right corner and select “Exclude From Project” option. Click the Commit button in “Pending Changes Panel” in the lower left corner of Visual Studio. Highlight the file JillDevProject in the Solution Explorer panel in the upper right corner. Click the Show all Files icon and then the refresh icon. Use the File menu and Save all option.	The Visual Studio solution will hold the work space information for each project. You will get a Microsoft Visual Studio warning indicating that the .svn directory is a Subversion administrative area and should not be added. This is expected and not a problem. This file should not be monitored by Subversion since it is just an administrative directory. AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option.

	Click the Commit button in “Pending Changes Panel” in the lower left.	
4.	Set up the Start Page so you can easily switch between projects. Select File -> Close Solution option. Select the View Start page icon above the Solution Explorer Panel in the upper right section of Visual Studio. Use the File -> Open -> Project/Solution option and navigate to the JoeDev working copy subdirectory in the Open Project popup. Highlight the file JoeDevProject.sln and click the open button. Highlight the file JoeDevProject in the Solution Explorer panel in the upper right corner. Click the Show all Files icon and then the refresh icon. Select File -> Close Solution option. Use the File -> Open -> Project/Solution option and navigate to the JoeDev working copy subdirectory in the Open Project popup. Highlight the file JillDevProject.sln and click the open button. Highlight the file JoeDevProject in the Solution Explorer panel in the upper right corner. Click the Show all Files icon and then the refresh icon.	During many of the exercises, you will be switching from each project frequently. Setting up the start page so it has both projects listed will make it easier to switch back and forth. AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option.

5.	Add a new folder called *FirstLessons* and a file under it called *First.txt.* Select the Start Page tab in the upper left section of Visual Studio, and click onto the JoeDevProject under the Recent Projects Panel. Right click the *EasySVN* folder in the project and select the Add -> New Folder option. Type FirstLessons for the folder name. Right click on the FirstLessons folder and select Add -> New Item option. Select the text file Visual Studio template, and name the file First.txt. Click the Commit button in "Pending Changes Panel" in the lower left.	Use the Solution Explorer Panel in the upper right section of Visual Studio to make modification to this project.
6.	Select *DeleteMe.txt* in the Solution Explorer and click on Delete. Hit the OK button on the Warning message.	You will get a Warning Popup that states that the file will be deleted permanently, which is fine.
7.	Select *RenameMe.txt* and choose Rename to rename the file to *ChangedName.txt.*	You use the normal Visual Studio Rename option.
8.	Double click on the file *EditMe.txt* to edit it changing "anything" on line 2 to "something" and save the file.	Content change.
9.	Click the Repository Explorer tab in the upper left section to see all the changes you have made so far.	High level checking.
10.	Right click the file *EditMe.tx*t, click on the Subversion -> Compare options. Select Base for the "From:" area and the Working type for the "To:" area then click the OK button. View the Base and the Working Revisions to see the difference between what you checked out and what you've now changed. Close the comparison window.	In order to see a graphically representation of the two you have to set up the Source Control environment. Select the Tools -> options from the menu options. Expand Source Control options. Highlight the Subversion User Tools. Use the down arrow to change the Diff, Merge, and Patch tools to TortoiseSVN. 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. (One can also use the Show Changes option when you right click on the EditMe.txt file.)
11.	Click on the file *ChangedName.txt* and select Revert.	Remember a Rename/Copy is implemented as a combination of a delete and an add operation

Click on OK. You should also probably go ahead and delete the now unversioned *ChangedName.txt* file.	though that is hidden to you in Visual Studio. 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.
12. Click the Update option in the Pending Changes are in the lower left section.	This operation will report the changes made to your project after updating it with the latest changes from the repository.
13. At this point there are no conflicts, if there were the steps would be the same as defined later for merge.	
14. Click the Commit option in the Pending Changes area in the lower left section 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) in the Message section 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.
15. Click the Update option in the Pending Changes are in the lower left section to bring in changes made and committed by others.	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. Click on to the Start Page tab in the upper right section of Visual Studio, and the JillDevProject. Click on to the Update icon in the Pending Changes area in the lower left section of Visual Studio. Highlight the JillDevProject in the Solution Explorer in the upper right, and ensure the Show All Files icon is selected. Click the Refresh icon to update this view. Right Click on the white folder FirstLessons and select the “Include In Project” option. Delete any files with question marks. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	You should always begin work with the latest revision from the repository. AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option. These are files that are not part of the current repository, but are still defined in this view. Some client applications do this automatically, but in Visual Studio one had to manually perform that task. This provides the developer with more control on when and what files are automatically added or deleted out of a project.
2. Add a new folder called SecondLessons and a file under it called Second.txt. Right click the EasySVN folder in the project and select the Add -> New Folder option. Type SecondLessons for the folder name. Right click on the SeondLessons folder and select Add -> New Item option. Select the text file Visual Studio template, and name the file Second.txt. Click the Commit button in “Pending Changes Panel” in the lower left.	Use the Solution Explorer Panel in the upper right section of Visual Studio to make modification to this project.
3. Select the *MoveMe* folder in the Solution Explorer, and right click drag the directory to the AddMe Folder.	
4. Right click on the file *EditMe.txt* and select the Subversion Lock option.	Locking should normally be used for file types that can’t be merged, but it may be used for other

	Enter a reason in the message area of the popup for getting the lock and click on OK.	purposes.
5.	Double click the file “EditMe.txt” and edit it to change “line” to “sentence” on the 2nd line and save the file.	
6.	Click the Repository Explorer tab in the upper left section to see all the changes you have made so far.	High level checking.
7.	Click the Update option in the Pending Changes are in the lower left section.	This operation will report the changes made to your project after updating it with the latest changes from the repository.
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.	Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 1.2) in the Pending Changes Message area. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes done up until now.	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.
10.	Click the Update option in the Pending Changes are in the lower left section to bring in changes made and committed by others.	This is to ensure that your working copy does not have mixed revisions.

4.3. Suggested Solution for Examine History

Steps		Explanation
1.	Right click onto JillDevProject in the Solution Explorer and select “View Project History”. You can also perform this via the Repository Explorer by right clicking the trunk subdirectory and selecting the view history option.	The entire trunk folder structure’s (any revision where something changed) history will be displayed.
2.	Right click on *EditMe.txt* and select “View History” option.	Show log will display only the history of the selected file.
3.	Compare different versions of EditMe.txt file.	Diff will show the modifications made in the project with respect to the two previous revisions.

	Right click the file EditMe.txt, click on the Subversion -> Compare options. Select Working type for the “From:” area and the Revision for the “To:” area. Select the specific revision in the Selection box that just appeared to the right. Select the previous revision which is likely revision 2 and then double click that revision or hit the OK button. Close the comparison.	One can also use the ellipse button to the right of the revision selection box to determine which revisions have been changed for that specific file. One can also use the View History Popup and a Control-click on two revisions to compare differences.
4.	Right click on the file *EditMe.txt* and right click to select Subversion -> “Show Annotation” option. (Use the defaulted values) 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 Subversion -> “Subversion Property” option Click the Add button and 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 Subversionlogo.jpg file and choose Subversion -> “Subversion Property” option 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.	Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 1.4) in the Pending Changes message area in the lower left corner. Click the Update option in the Pending Changes are in the lower left section.	Note the incremented global revision number after a successful commit.
4.	Click the Update option in the Pending Changes are in the lower left section to bring in changes made and committed by others.	This ensures that your project does not have mixed revisions.
5.	Bring up the Windows Explorer and verify the only attribute is set.	This verification requires some type of Windows file manager application to see this attribute. Recommend using Window Explorer

		for this step.
	Left click on the Window Start button at the bottom of the Windows Task bar area, and select the Window Explorer option. Navigate to the JillDev working copy and select the EasySVN folder. 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.	Right click on the file *SubversionLogo.jpg* and select the Subversion -> Lock option. Enter a reason for getting the lock and click on OK.	This informs the Subversion server that you reserve the right to create the next revision of this file on the trunk.
7.	Bring up the Windows Explorer and verify the only attribute is not set. Left click on the Window Start button at the bottom of the Windows Task bar area, and select the Window Explorer option. Navigate to the JillDev working copy and select the EasySVN folder. 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.	Right click on the file *SubversionLogo.jpg* and select the Subversion -> Unlock option and click on OK.	

5. Solutions for Lab 2 – Essential Concepts 2

5.1. Suggested Solution for Traversing

Steps	Explanation
1. Use the JoeDevProject and update EditMe.txt to revision 2.	You should always begin work with the latest revision from the repository.
Click on to the Start Page tab in the upper right section of Visual Studio, and the JoeDevProject.	
Click on to the Update icon in the Pending Changes area in the lower left section of Visual Studio.	AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option.
Highlight the JoeDevProject in the Solution Explorer in the upper right, and ensure the Show All Files icon is selected.	
Click the Refresh icon to update this view.	These are files that are not part of the current repository, but are still defined in this view. Some client applications do this automatically, but in Visual Studio one had to manually perform that task. This provides the developer with more control on when and what files are automatically added or deleted out of a project.
Right click on the MoveMe folder, and select the “Include In Project” option	
Right click on the SecondLessons folder, and select the “Include In Project” option.	
Right click on the file Moved-file.txt and select the delete option.	
Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	
Make sure your JoeDev project is pointing to the trunk reviewing the URL location in the Property Window in the lower right.	If the Property Window is not selected use the view menu option and select it.
Scroll down to the URL line, and click into the actual URL address and use the arrow key to move to the far right of the box.	
Verify that the file is in the trunk subdirectory looks something like the following: file:///C://lab_repo/trunk/EasySVN/EditMe.txt	

	Right click on the *JoeDevProject* folder and select Subversion -> “Switch Project” option. Use the drop down arrow and change the “Type:” to Revision, Hit the up arrow twice in the revision number section on the far right of the Switch popup window, and then click OK.	
2.	Open the file and check the contents to observe that the changes you made after revision 2 are not included.	
3.	Right click on the JoeDevProject folder and select Subversion -> Switch option. Use the drop down arrow and change the “Type:” to Revision, Hit the up arrow once in the revision number section on the far right of the Switch popup window, and then click OK.	
4.	Open the file and check the contents to observe that the changes.	
5.	Right click on the JoeDevProject folder and select Subversion -> Switch option. Accept the default to “Latest Version” and click on OK. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	

5.2. Suggested Solution for Tree Conflict

Steps	Explanation
1. Click on to the Start Page tab in the upper right section of Visual Studio, and the JillDevProject.	You should always begin work with the latest revision from the repository.
Click the Update down arrow icon in the Pending Changes area in the lower left, and select “Update to Latest Version” option.	AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option.
Highlight the JillDevProject in the Solution Explorer in the upper right, and ensure the Show All Files icon is selected.	
Click the Refresh icon to update this view.	These are files that are not part of the current repository, but are still defined in this view. Some client applications do this automatically, but in Visual Studio one had to manually perform that task. This provides the developer with more control on when and what files are automatically added or deleted out of a project.
Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	
2. Click on to the Start Page tab in the upper right section of Visual Studio, and the JoeDevProject.	You should always begin work with the latest revision from the repository.
Click the Update down arrow icon in the Pending Changes area in the lower left, and select “Update to Latest Version” option.	AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option.
Highlight the JoeDevProject in the Solution Explorer in the upper right, and ensure the Show All Files icon is selected.	
Click the Refresh icon to update this view.	These are files that are not part of the current repository, but are still defined in this view. Some client applications do this automatically, but in Visual Studio one had to manually perform that task. This provides the developer with more control on when and what files are automatically added or deleted out of a project.
Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	
Select the SecondLessons directory in the Solution Explorer, and right click drag the directory to the AddMe Folder.	
3. Click the Commit button in “Pending Changes Panel” in the lower left.	

4. Click on to the Start Page tab in the upper right section of Visual Studio, and the JillDevProject. Click the Update down arrow icon in the Pending Changes area in the lower left, and select “Update to Latest Version” option. Highlight the JillDevProject in the Solution Explorer in the upper right, and ensure the Show All Files icon is selected. Click the Refresh icon to update this view. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution. Double click to edit the Second.txt file under the SecondLessons, and enter the text “Tree Conflict Looming”.	You should always begin work with the latest revision from the repository. AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option. These are files that are not part of the current repository, but are still defined in this view. Some client applications do this automatically, but in Visual Studio one had to manually perform that task. This provides the developer with more control on when and what files are automatically added or deleted out of a project.
5. Click the Update icon in the Pending Changes area in the lower left.	Notice that you have a tree conflict.
6. Fix the tree conflict by using the Subversion -> Include and Exclude Project options. Highlight the JillDevProject in the Solution Explorer in the upper right, and ensure the Show All Files icon is selected. Click the Refresh icon to update this view. Right click on the Yellow SecondLessons folder and select the “Exclude From Project” option. Right click on the White SecondLessons folder and select the “Include In Project” option. Update and Commit changes via the Pending changes section in the lower left.	After an updated is perform selecting the project and refreshing will show all additions and deletions. All new files and folders will be shown as a blank or white. All new deletions will be shown as a question mark before the file name. You then have they ability to add or delete these items when you choose do so. Although you can see what these files are and decide if you desire to place them in your project, it is not an automatic actions like others tools. If a Tree conflict Pending Changes item does not go away after the fix use highlight it and use the remove from Pending Changes option.

5.3. Suggested Solution for Merging

Steps	Explanation
1. Using JoeDev, create a branch “Demo” (from HEAD of the trunk) in the branches subdirectory. Choose to stay on the trunk.	You should always begin work with the latest revision from the repository.
Click on to the Start Page tab in the upper right section of Visual Studio, and the JoeDevProject.	AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option.
Click on to the Update icon in the Pending Changes area in the lower left section of Visual Studio.	These are files that are not part of the current repository, but are still defined in this view. Some client applications do this automatically, but in Visual Studio one had to manually perform that task. This provides the developer with more control on when and what files are automatically added or deleted out of a project.
Highlight the JoeDevProject in the Solution Explorer in the upper right, and ensure the Show All Files icon is selected.	
Click the Refresh icon to update this view.	
Use the Repository Explorer view in the upper left section of Visual Studio, and navigate to the branches directory.	
Click the branches folder and click the “Copy URL to Clipboard” icon above.	
Right click the JoeDevProject in the Solution Explorer in the upper right, and select the Subversion -> “Branch Project” option.	Option of switching to the new branch / tag is available. However, to remain on the trunk do not check the box labeled “Switch working copy to new branch/tag”.
Paste the repository address in the To: URL area, and append the name Demo right after the trailing slash. It should like something like the following: file:///C:/ /Labs/lab_repo/branches/Demo	
Enter a useful commit message (perhaps noting this is for lab 2.2 part 1) and click on OK.	Highlight the Repository icon above the branches directory and click the refresh icon. Navigate to the branches directory and validate the Demo branch was created.
2. Edit the *EditMe.txt* file (on the trunk) by changing the last word from “Subversion” to “SVN” and saving the file.	

3.	Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 2.2 part 2) in the Pending Changes Message area. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	
4.	Right click the JoeDevProject in the Solution Explorer in the upper right, and select the Subversion -> "Switch Project" option. Click the ellipse box to the right of the URL text and navigate the repository to the branches/Demo directory and left click it to highlight it. Hit OK.	Note that the only reported action is an update of the *EditMe.txt* file. Note the change is reflected in the path using the Property Window or by viewing the URL in the Pending changes area.
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.	Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 2.2 part 3) in the Pending Changes Message area. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	
7.	Click on to the Update icon in the Pending Changes area in the lower left section.	
8.	Perform a merge of the trunk and branch. Right click the JoeDevProject in the Solution Explorer in the upper right, and select the Subversion -> "Merge Project" option. Select the first one "Merge a range of revisions", uncheck the "Perform pre-merge best practices check box", and click on Next. Select the only Revision option in the Merge Wizard, and hit Next until Finished.	The trunk changes have to be brought into the branch. Normally one would check the Perform pre-merge best practices check, but we want to simulate a case where there is a problem. This is a Dry run option to test the merge in the Merge Summary view of the Merge Wizard.
9.	There will be a conflict reported here since the same line has been changed so choose to launch a graphical conflict resolution editor.	We have changed the options to ensure that we use the TortoiseMerge GUI to resolve any conflict. This option can be changed to via the Tools ->

The conflict resolution editor will show 2 different windows. Right click on the Red line within the Mine Panel and select the Use This Whole File" option. Use the TortoiseMerge Save the results and Exit out of it via File -> Exit option. Click the Yes button on the Visual Studio Resolve Conflict popup. Note you'll get a dialog showing the merge results summary with just 1 Updated.	options selection, and then expanding the Source Control to highlight the Subversion User Tools.
10. Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 2.2 part 4) in the Pending Changes Message area. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	

6. Solutions for Lab 3 – Enterprise Features

6.1. Suggested Solution for Advanced Merge

Steps	Explanation
1. Using JillDev, create a new branch by the name of Feature1 but keep your working copy referencing the trunk.	
Click on to the Start Page tab in the upper right section of Visual Studio, and the JillDevProject.	You should always begin work with the latest revision from the repository.
Click on to the Update icon in the Pending Changes area in the lower left section of Visual Studio.	AnkhSVN will not automatically show newly created file and folder. After each update ensure you refresh the Solution Explorer view. If you see any new files or folder you want in the project right click them and select “Include In Project” option.
Highlight the JillDevProject in the Solution Explorer in the upper right, and ensure the Show All Files icon is selected.	
Click the Refresh icon to update this view.	These are files that are not part of the current repository, but are still defined in this view. Some client applications do this automatically, but in Visual Studio one had to manually perform that task. This provides the developer with more control on when and what files are automatically added or deleted out of a project.
Use the Repository Explorer view in the upper left section of Visual Studio, and navigate to the tags directory.	
Click the branches folder and click the “Copy URL to Clipboard” icon above.	
Right click the JillDevProject in the Solution Explorer in the upper right, and select the Subversion -> “Branch Project” option.	
Paste the repository address in the To: URL area, and append the name Demo right after the trailing slash. It should like something like the following: file:///C:/Labs/lab_repo/tags/Feature1	
Enter a useful commit message (perhaps noting this is for lab 3.1 part 1) in the Pending Changes area and click on OK.	Highlight the Repository icon above the branches directory and click the refresh icon. Navigate to the tags directory and validate the Feature1 branch was created. You may need to refresh the view before that new tag is visible by using the refresh icon.

2.	Edit *First.txt* however you wish and save your changes.	
3.	Enter a useful and concise log message (perhaps including the fact that this is being done for Lab 3.1 part 2) in the Pending Changes Message area. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	
4.	Right click on the JillDevProject folder and select Subversion -> Switch option. Click on the ellipse box in the “To: URL” Section, and navigate to the tags/Feature1 directory in the popup window. Click OK.	Note the change is reflected in the Property Window where the URL is listed.
5.	Right click the JillDevProject in the Solution Explorer in the upper right, and select the Subversion -> Merge Project option. Select the first one “Manually record merge information”, uncheck the “Perform pre-merge best practices check box”, 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 Next until Finish.	Normally one would check the Perform pre-merge best practices check, but we want to simulate a case where there is a problem. This is a Dry run option to test the merge in the Merge Summary view of the Merge Wizard.
6.	Examine First.txt to see that changes from the trunk were not applied.	You can use the Repository Browser to Examine what existed before the modifications.
7.	Enter a useful and concise log message (perhaps including the fact that this is being done for 3.1 part 3) in the Pending Changes Message area. Click the Commit option in the Pending Changes area in the lower left section to commit all the changes to the solution.	
8.	Right click on JillDevProject and select Subversion -> “Project Properties” option.	Note the value of the svn:mergeinfo property showing the manual merge.