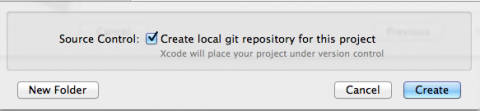
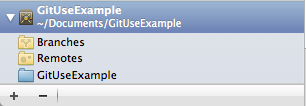
**This is inspired from** [**http://www.raywenderlich.com/13771/how-to-use-git-source-control-with-xcode-in-ios-6**](http://www.raywenderlich.com/13771/how-to-use-git-source-control-with-xcode-in-ios-6) **and Malek Trabelsi and** [**http://learnaholic.me/2012/11/29/xcode-4-dot-5-source-version-control-wit-git-and-github/**](http://learnaholic.me/2012/11/29/xcode-4-dot-5-source-version-control-wit-git-and-github/)

**Step 1: Create a new Single View Application project, make sure ARC and Storyboards are checked.**

**Step 2: Click Next and select a check box**

This creates a new repository and makes the first “commit” of your project

**Step 3: Select the “Organizer” and Click on Repositories**



**The letter “A” means new file and the letter “M” means modified file**

**Click on it and you will see all the details about the new files added to your blank project and a typical commit put by Xcode to indicate that a commit action was performed**

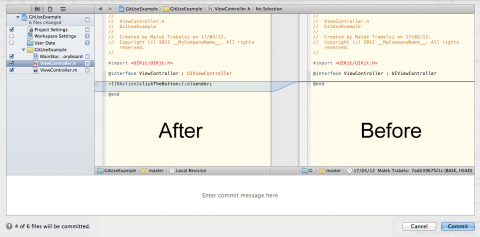
**Now let’s look at the version editor and its options**

**Blame View: Who changed what line and allows you to catch every commit on your file easily, in real time, so that you can distinguish which commit deals with each revision.**

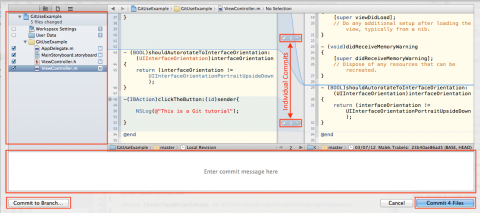
**Log View: When the line was changed**

**Comparison View:**

**The left pane shows the file in its current state with all changes made since the last commit. The right pane shows the file before you made your changes.**

****

**Commits are listed in the middle**

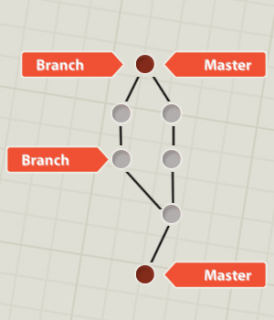
****

**So let’s make some changes and experiment with commits.**

**Remember, all of these commits were made locally.**

**Branches Explained**

**You can commit your changes to a branch**

****

**The master branch is your main copy. Other branches are for experimentation.**

**So, it you are adding a class that you just want to experiment or learn with, but not alter the main project, you’d add a branch.**

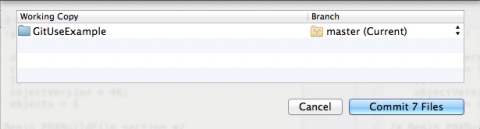
**Trying it:   
Step 1: Add a class**

**Step 2: Now, go to File/Source Control/Commit to switch to the version editor.**

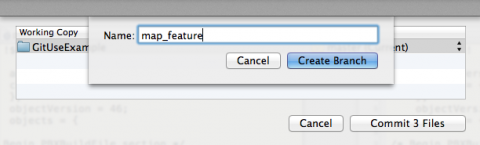
**If you select one of the files with the A status, you may notice that Xcode didn’t provide any earlier version to compare. This is because the file hasn’t been committed to the repository yet, so there is nothing to compare it with.**

**Ok, so you added 2 files (whatever your class was) to the project – Xcode detected that, and is ready to commit. However, you may need to assign these new changes to another specific branch to deal with. That will help to isolate the risk in case there are some problems with your new code.**

**Instead of clicking the “Commit Files” button, click the “Commit to Branch” button instead. This will appear:**

****

**Now click on the Master folder and click create new branch**

****

**Now go back to the organizer and check out your branch**

**RECOVERY!**

**In case you want to revert to a point where the version last worked…**

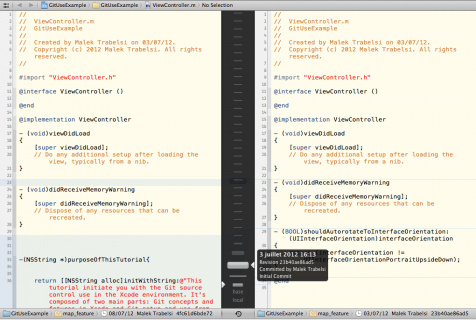
**When you make a mistake and want to revert, the first thing you should do is “discard changes” in your commit screen:**

**File -> Source Control -> Discard Changes.**

**Don’t Commit lets you skip the change so that it won’t be committed with other changes, but it will remain in the local source code.**

**Discard Changes not only skips the change, but also deletes it from the local source code.**

**The timeline helps us see which versions were saved over time:**

****

**More on branching:**

**It’s good coding practice to always work on a copy of your project – the development branch, if you will. And once you complete a development milestone, you normally merge the development branch back to the trunk.**

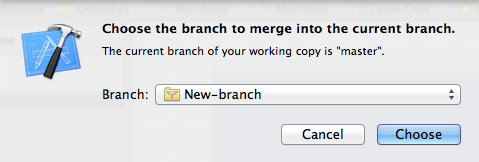
**If you need to remove a branch, you can go to the Repositories view.**

**Select your project from the left sidebar and make sure that the Branches folder/group is selected.**

**You should now see the “Add Branch” and “Remove Branch” buttons at the bottom of the window.**

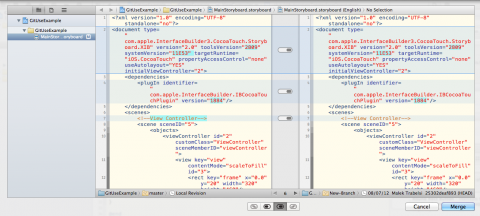
****

**Once everything good in your experimental branch, you can merge it with the working copy.**



**A new window will show up where you can use the left and right buttons on the bottom menu to specify the direction of the merge.**

**In our case, you want to merge the new branch into the master branch which is the current one. Since the current branch is on the left and the new branch is on the right, you should select the “Right” button to transfer the changes from the right pane to the left pane.**

****

**What is .gitignore?**

**Git uses a special file called .gitignore, which holds information about files you don’t want to commit to the Git repository.**

**But, why should you think about ignoring some of your files since they depend on your project?**

**Some files need to be ignored like your setting file .DS\_Store, or your private shemes files like .xcuserdatad.**

**SO ADD a .gitignore text file that contains the following:**

build/\*

.DS\_Store

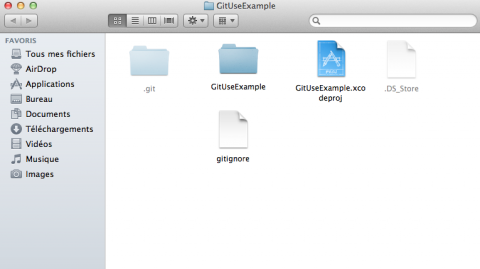
\*.xcuserdatad

**build/\* : This will exclude the whole build directory which is created by Xcode when creating the application binary.**

**DS\_Store : This rule will exclude the OS X folder attributes.**

**xcuserdata : Exclude user data files.**

**ALWAYS SAVE .gitignore files to the root directory of your project:**

****

**If you have git and run**

|  |
| --- |
| xcrun git config --global core.excludesfile ~/.gitignore\_global |

**It will ignore the same file always**

**LAST THING…GITHUB for team stuff**

**Set up git: https://help.github.com/articles/set-up-git**

[**Go to github and create an account.**](https://help.github.com/articles/set-up-git)

**Then create a new repo!**

**Push your project and view it on the github website**