APPENDIX A

Introducing Xcode Source Control

What You'll Learn in This Appendix:

- The source control features offered in Xcode
- The language of source control systems
- How to connect to remote Subversion and Git repositories
- The tools available for working with Subversion and Git

As projects grow larger, it becomes more and more important to keep track of the changes that you make and to be able to identify modifications to your code. A change can sometimes cause unintended behaviors that, without a good way to audit your coding, might be difficult to track down. This point holds especially true for projects where one or more people need to access and modify the same project.

The answer to this problem is to use *source control*—a term that refers to a system that keeps track of changes to source code files. This appendix walks through the basics of source control and helps you get started using the tools Xcode provides.

A Brief Introduction to Source Control Systems

Xcode includes support for two popular version control systems: Subversion, also known as SVN (http://subversion.apache.org/), and Git (http://git-scm.com/). Like any version control system, the purpose of these products is the same, but the implementation in Xcode differs enough to be confusing. Before trying to use Xcode with SVN or Git, you need a bit of background in the philosophy and terminology of each system.

Let's Cut to the Chase Already!

Let's face it. Source control isn't a very exciting topic. You want to program, and your book told you that you could get some additional information from this online supplemental material. So, where's the payoff?

Here you go:

I've placed the book's tutorials in a source control repository that you can connect to with Xcode. This means that you can access the project files with no manual downloading, unzipping, and so forth. Just add the repository, and poof, instant access to the *very latest* versions of the projects! If this is what you're interested in, skip ahead to the section at the very end: "Accessing the *Teach Yourself iOS 9* Tutorials Through Subversion." (Don't worry, you can come back and read the rest of this appendix later.)

Repositories and Working Copies

Both Git and Subversion can provide server-based repositories. These act as very lightweight file servers that include the files for a project, along with a log of all the changes that have been made over the life of the project.

To edit code that is stored in a repository, you create what is called a *working copy* from all, or a portion of, the repository. The working copy is a local copy of a project that is edited on your computer (often pulled from a Git or SVN repository). The initial creation of a working copy is called a *checkout* in SVN terminology and a *clone* in Git terminology.

NOTE

A checkout creates a copy of the code in an SVN repository along with the information SVN needs to track your changes. A clone of a Git repository, however, creates a full local Git repository that is no longer reliant on a server at all. The local repository contains a "remote" entry that links back to the originating server, but you do not have to use it if you don't want to.

Committing Changes

One of the biggest differences between your use of Git and SVN in Xcode is saving changes to your code. You edit your projects exactly as you always have, editing the working copies of your code. When you decide you want to save a change to the repository, you execute a *commit* action. The commit notes the change, and gives you an opportunity to annotate the change, as well.

In Subversion, a commit stores the updated file back to the repository immediately. In Git, however, your local copy *is* the repository, so nothing is updated on the remote repository. To push your changes back to a network repository, you must execute a push command, as discussed later in this hour.

Downloading Changes

As you commit and, in the case of Git, push changes to a central repository, you'll also want to download changes that other users have added to the repository. To do this, you execute an update command on the SVN repositories and a pull on Git repositories. In either operation, if conflicts with the code occur in your working copy, you are given the opportunity to reconcile the conflicting code.

Branching and Merging

Developers often need to maintain a release version of a product while working on new features. The base/release version of a project in Git/SVN is the *trunk*. New versions of the trunk are developed in branches off of the trunk or off of another branch. In Subversion, you work on branches in a new working copy of your code. Git maintains a single working copy and enables you to switch branches at will.

When changes are complete, and a branched version is ready to become the release version, it is merged with another branch (or the trunk) to create a unified code base. Conflicts are dealt with, and then the finished code is committed to the repository and pushed (with Git) to the remote server, if desired.

NOTE

You might, in your SVN/Git travels, encounter references to the term *tags*. Tags are simply named copies of a repository. You might maintain a tagged copy of each release of your software (version 1.0, version 2.0, and so on).

These are the basic operations that you learn about in the rest of this appendix. We cover just the tools necessary to make this work in Xcode; there are entire books written about using Git and Subversion, and the feature set exposed in Xcode is much, much smaller than what is available from the command-line interface (CLI). If you want to learn more about these tools, I highly recommend reading the resources on their respective websites.

TIP

If you want to experiment with server-hosted Git/Subversion repositories, I highly recommend signing up for an account at Beanstalk (http://beanstalkapp.com/), Assembla (https://www.assembla.com/), or Github (https://github.com/). The first two sites offer low-cost (and trial) accounts with access to both Subversion and Git. The third option (Github) provides free Git repositories for open source projects and low-cost options for private Git repositories.

Working with Subversion and Git Repositories

Xcode's integration of Subversion and Git features directly into the user interface (UI) make working with these systems quite simple, even if you have never touched a version control system before. In fact, Xcode even enables you to create local Git repositories that you can use to track your own changes on a project or share, via a file server, with other individuals working on a project.

In this part of the appendix, we create a local repository, and then cover the process needed to connect to remote SVN or Git servers. Because these topics do not necessarily flow together as well as I would like, do not feel bad about jumping around to find the information relevant to you.

TIP

You can find nearly all of Xcode's source control features under the Source Control menu, although new repositories can also be added within the Xcode Preferences, Accounts area.

Creating Local Git Repositories

If you're a small developer with a good backup system and only a few people working on a project, you'll probably do just fine with a locally hosted Git repository that Xcode can create for you when you start your project.

To create a project that includes its own Git repository, follow these steps:

1. Begin the project-creation process, described in Hour 2.
2. When prompted to save the project, be sure that the Source Control check box is checked, as shown in Figure A.1.
3. Click Create. The project is created and opened.

As you work on your project, you'll see status icons (letters) appear beside your files in the project navigator (most often *M*, as shown in Figure A.2). This shows the status of the project's source control and demonstrates that your new project is within a repository. You'll learn more about the source control status icons in the section "Managing a Project in Source Control," later in this appendix.

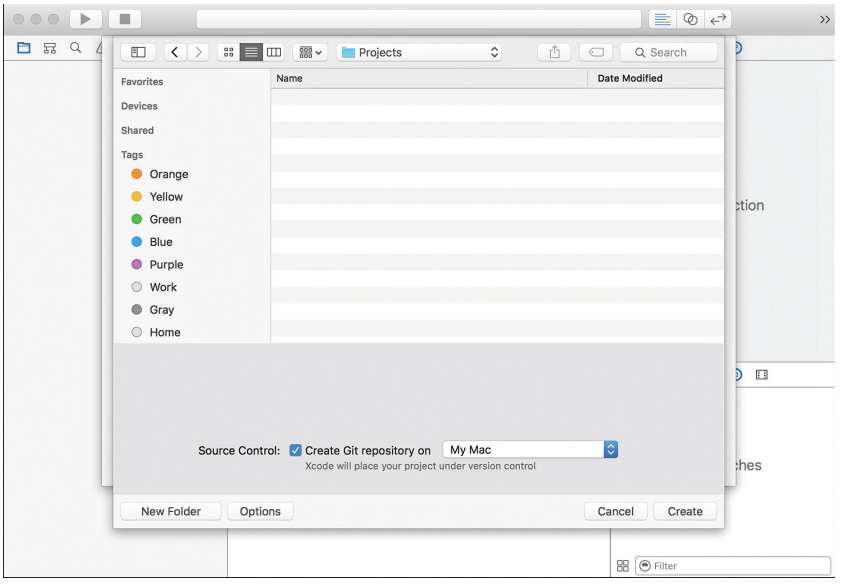

FIGURE A.1
Make sure that you have checked the option to use source control.

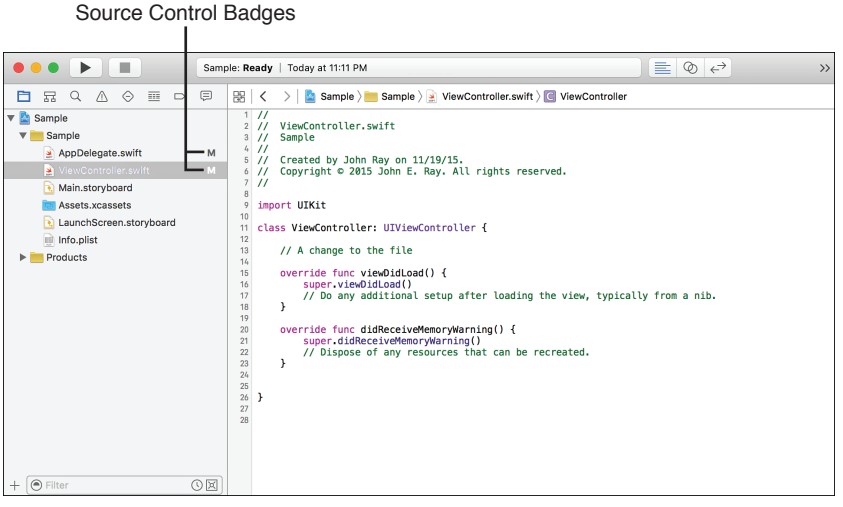

FIGURE A.2
You'll see a source control badges appear beside your project files as you make changes.

If you want to start working with files in your repository, skip ahead to the "Managing a Project in Source Control" section.

TIP

If you establish a local Git repository that you later want to connect to a remote repository, you can do so by choosing Source Control, <Project Name>, Configure <Project Name> from the menu bar. Within these settings, you can click the Remotes button to add a remote Git repository connect.

Connecting to Remote Repositories

If you have already established a hosted repository outside of Xcode, you can connect Xcode to the repository by walking through a simple setup assistant. This enables Xcode to download and work with copies of the stored code—or upload new code to the server. You manage access to your repositories through the Xcode preferences.

To add a new repository, follow these steps:

1. Open the Xcode preferences and click the Accounts icon.
2. Click the + button at the bottom of the window and choose Add Repository, as shown in Figure A.3.

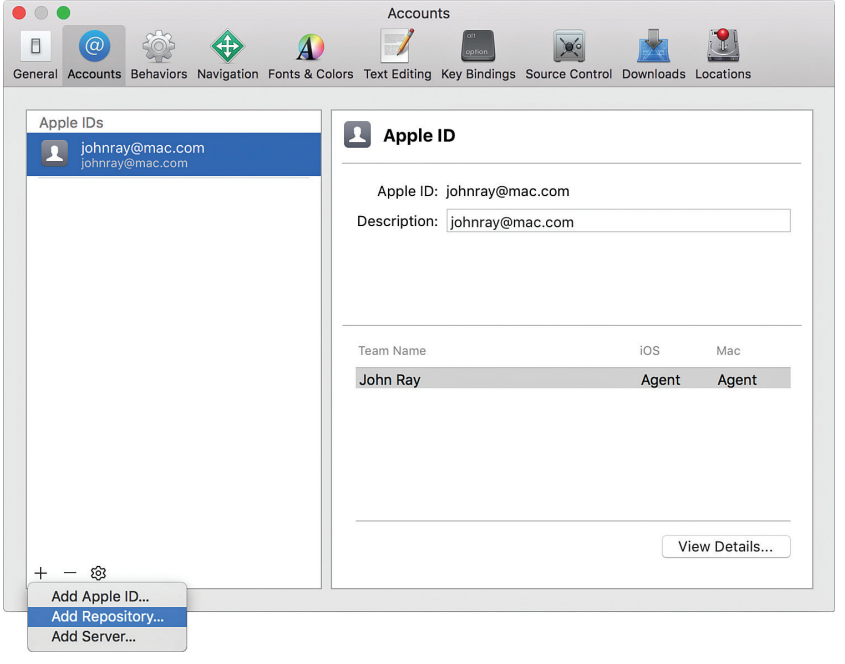

FIGURE A.3
Add repositories through the Xcode preferences.

3. Enter the URL for the repository, such as http://svn.placeforstuff.com/teachyourselfios9.
4. If you have authentication information, enter your credentials. These are established with your repository provider.
5. Subversion repositories may prompt for additional information—the paths to the Trunk, Branches, and Tags directories. Enter these if you have them; they are special folders within the repository and are not necessary for creating your connection.
6. The repository is added to the Accounts settings, as shown in Figure A.4. Selecting the repository enables you to change your authentication information and the repository description.

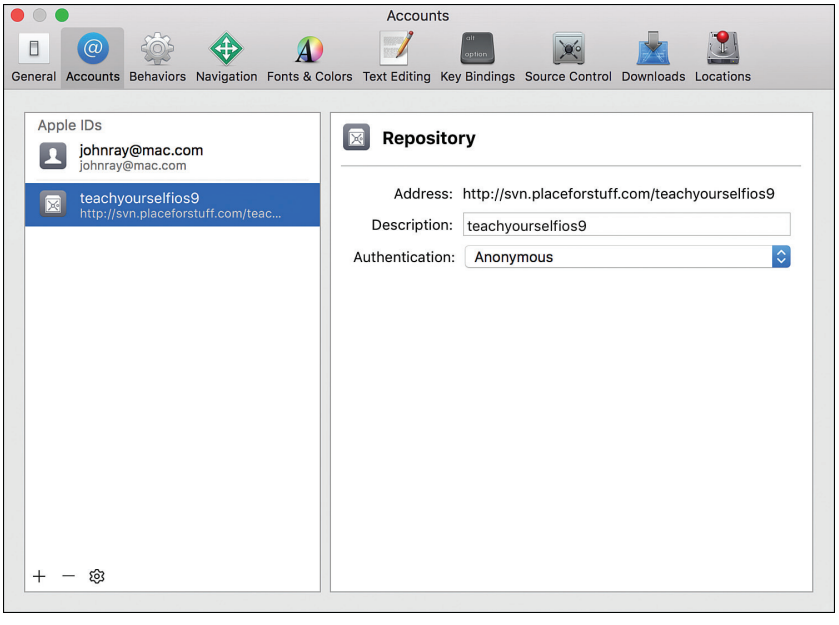

FIGURE A.4
Select repository entries to edit information about them.

Checking Out a Working Copy

After creating a remote repository and connecting to it, you'll likely want to add your project to it. To do this, you download a working copy of the repository, add your Xcode project, and then commit those changes back to the repository, as follows:

1. From the Xcode Source Control menu, choose Checkout.
2. Select the repository that you want to use, and then click Next, as shown in Figure A.5

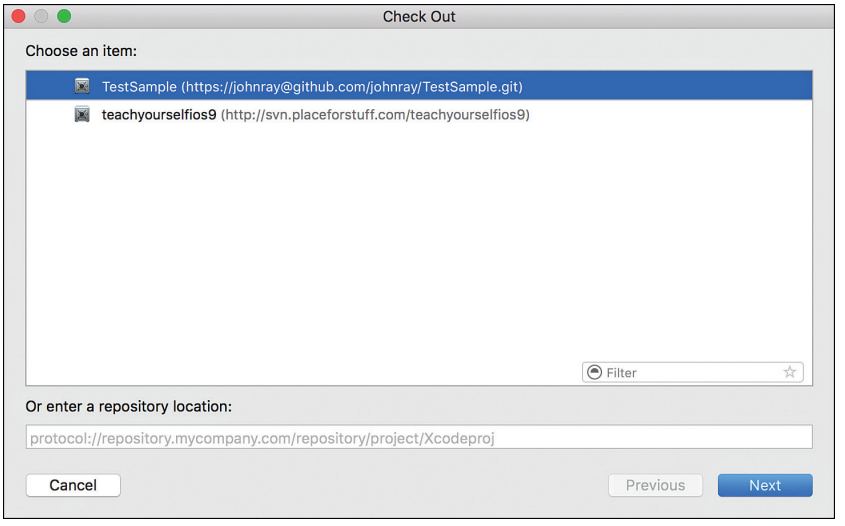

FIGURE A.5
Choose the repository you want to access.

3. Select the directory where you want to save the files, and click the Download button, as shown in Figure A.6
4. Xcode downloads the code into the directory you specified. If the repository is empty, this is just an empty folder.
5. Navigate to the folder in the Finder and copy your project to the folder.

Your project is now under source control with a remote repository connection. Your next step is to develop the world's greatest app while working within the source control system.

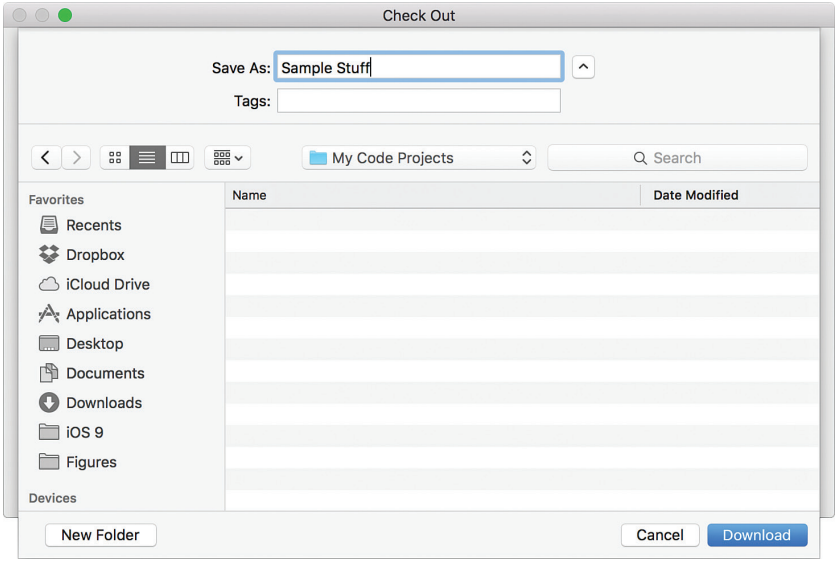

FIGURE A.6
Pick where you want to save the working copy.

Managing a Project in Source Control

Once you have a project under source control, you work just as you normally would, but you now have some additional options at your disposal. You'll also notice some changes in how the files display in the project navigator. In this, the last part of the appendix, we review many of the common activities you will perform with your Subversion or Git repository.

Status Codes

When working with a project that is under source control (that is, your working copy), you'll notice that a number of badges appear beside the files listed in your project navigator, as shown in Figure A.7.

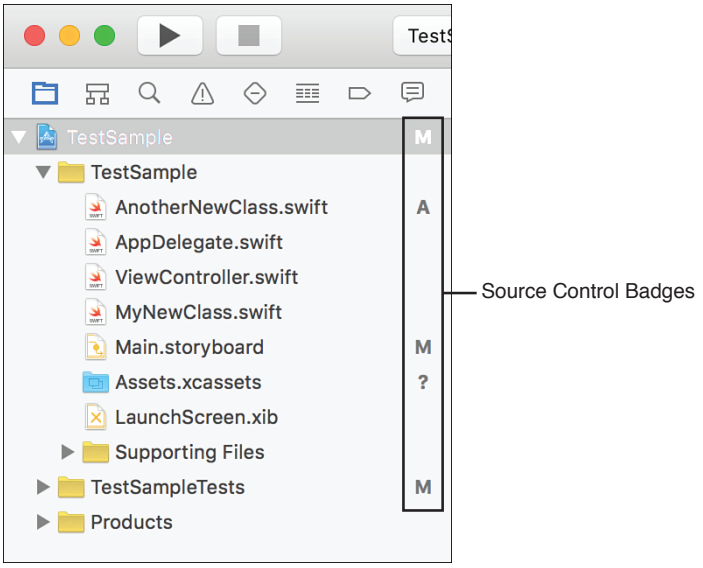

FIGURE A.7
The project navigator now contains badges indicating source control status.

Table A.1 lists the badges you might encounter, as provided by Apple.

TABLE A.1 Badges You Might Encounter

Symbol	Meaning
M	Locally modified file
U	Updated in repository
A	Locally added
D	Locally deleted
I	Ignored
R	Replaced in the repository
–	The contents of the folder have mixed status; display the contents to see individual status
?	Not under source control

TIP

You can click the tiny repository icon (shaped like a filing cabinet drawer) at the bottom of the project navigator to filter your project navigator files to show only the files that have an updated source control status.

Commits and Pushes

The most common type of change that you need to make when working with source control is a commit. A commit is used to add a finished collection of code to the repository. You might, for example, perform a commit at the end of every day, or commit any changes you have made after you have tested them thoroughly. You can also use a commit to add new files (denoted by ?) to the repository.

To perform a commit, follow these steps:

1. Choose Source Control, Commit from the menu. The Commit dialog appears, as shown in Figure A.8.
2. Check the check boxes beside each of the modified or added files that you want to commit.
3. Enter a message to describe your changes in the text area at the bottom of the dialog.
4. If working with a Git repository, check the Push to Remote check box to push all the changes to the remote server. (This will always happen on a commit with Subversion.)
5. Click Commit (or Commit and Push) to commit your changes to the repository.

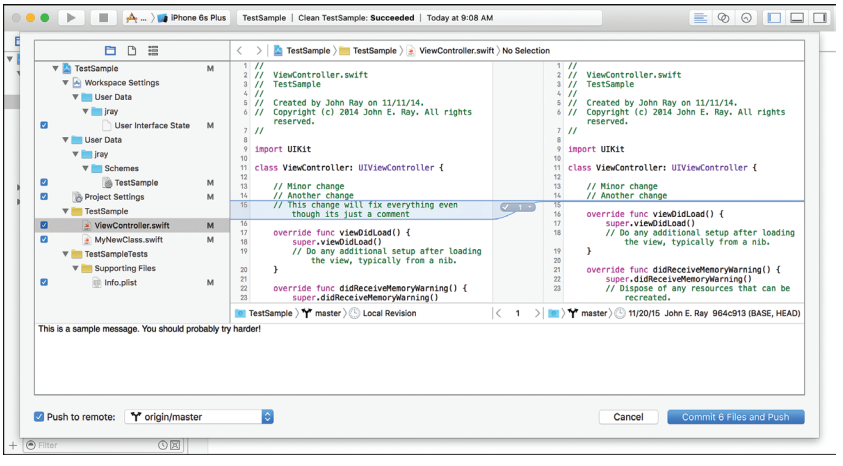

FIGURE A.8
Commit the checked files to the repository.

TIP

You might want to commit only related files at the same time—not everything simultaneously. This gives you an opportunity to document your changes more fully than applying a single commit comment to every changed file in your project.

After you have completed a commit on Subversion, you're done. Your file is sent back to the hosted repository. With Git, however, your changes are sent to the server only after you execute a push command. If you didn't choose Push to Remote while performing your commit with a Git repository, you'll need to choose Source Control, Push from the menu bar.

CAUTION

Don't Commit *Everything*!

You do not have to commit files just because they have been modified. You can discard your changes within the project navigator, or from the commit screen by right-clicking the file and choosing Discard Changes from the Source Control menu. Using this same menu, you can also choose to ignore files that you do not want to commit or manage in your repository or to commit a selection of files immediately.

Updates and Pulls

While you're making changes to your project in your local working copy, others might do the same in their copy, committing changes back to the repository. The end result: Your version of the project might become out of sync with the central repository. This is not a problem. In fact, it's a guaranteed outcome of a distributed source control system.

To update your working copy of a project to the latest version held in a repository, you use the Update command (Subversion) or Pull command (Git) from the Source Control menu.

NOTE

During a commit, update, or branch merge, you might get a warning about conflicting changes in your code. If this is the case, you are given a UI to correct any conflicts before proceeding. Conflicts occur when no clear path exists to merge heavy changes across a file—often when two people have modified the same method in the same file.

Viewing Revisions

Using source control enables you to go back through your code and see what changed and when. You can get a quick view of all the commits that have been made to a repository, working copy, or any directory of a repository by selecting History from the Source Control menu. The commit messages to the repository are shown, along with who made them, as shown in Figure A.9.

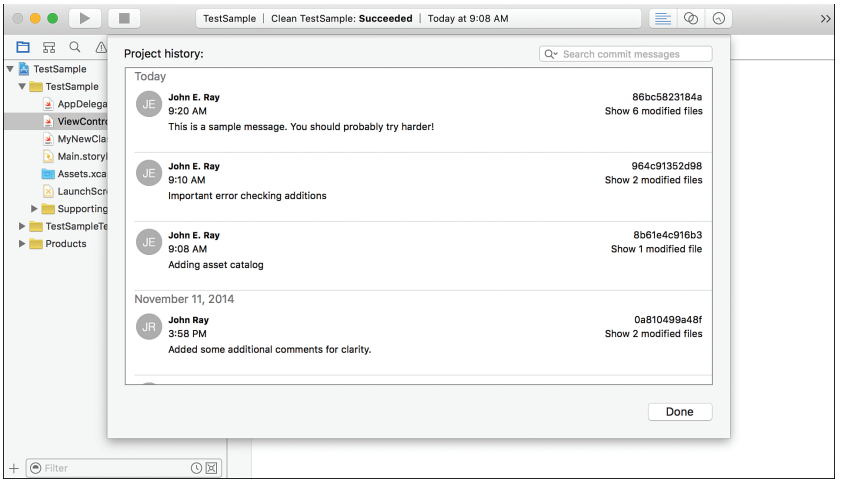

FIGURE A.9
Use the History to get a quick view of the history of your changes.

You can click the Show Modified File(s) links in the history to see the changes that have been made to a specific file.

Using the Version Editor

In addition to the Organizer, you can also view the changes that have been made to a file through the Version editor in the main Xcode interface. This works on the current file you have selected, so it is a bit easier to target and identify individual changes.

To display the Version editor, click and hold the third icon in the Editor section of the Xcode toolbar. You can choose between three types of view modes (Comparison, Blame, and Log), as shown in Figure A.10.

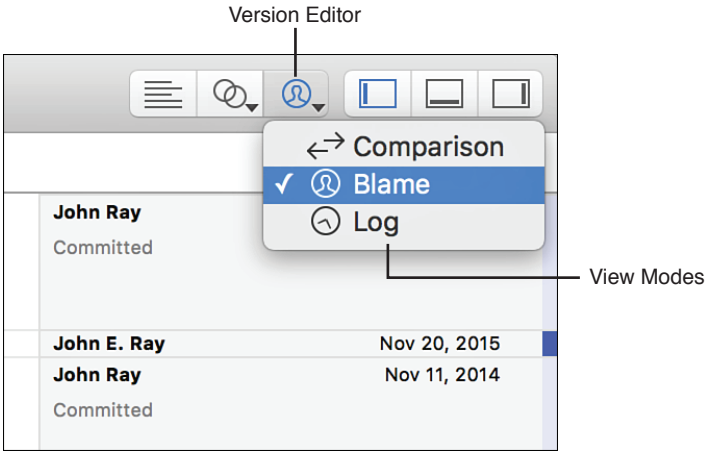

FIGURE A.10
Use the Version editor to target changes to a specific file.

In the Log mode, a chronological list of changes is shown to the right of the current file.

The Blame mode is similar to the Log mode, but displays the change entries next to the code that was changed, as shown in A.11. Within the Blame mode, click the *i* icon beside each name to show a popover with the full details about the change, and then click the Open Comparison button to view the changes.

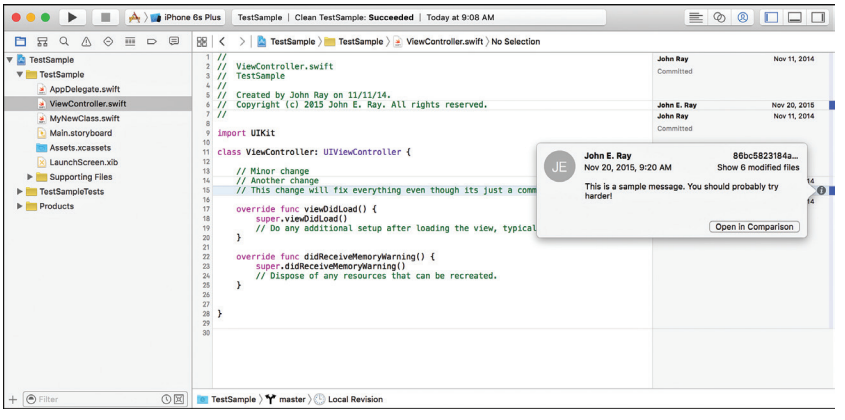

FIGURE A.11
The Blame mode makes it easy to find the team members who broke your app and yell at them.

The final view, Comparison mode, displays revisions side by side in the editor area. Use the pop-up file paths under each side of the view to choose the revision you want to see. Alternatively, you can click the clock icon to show a Time Machine-like timeline view of both files.

If you find a change you don't like, just click the drop-down menu that appears in the Comparison mode and choose Discard change, as shown in Figure A.12.

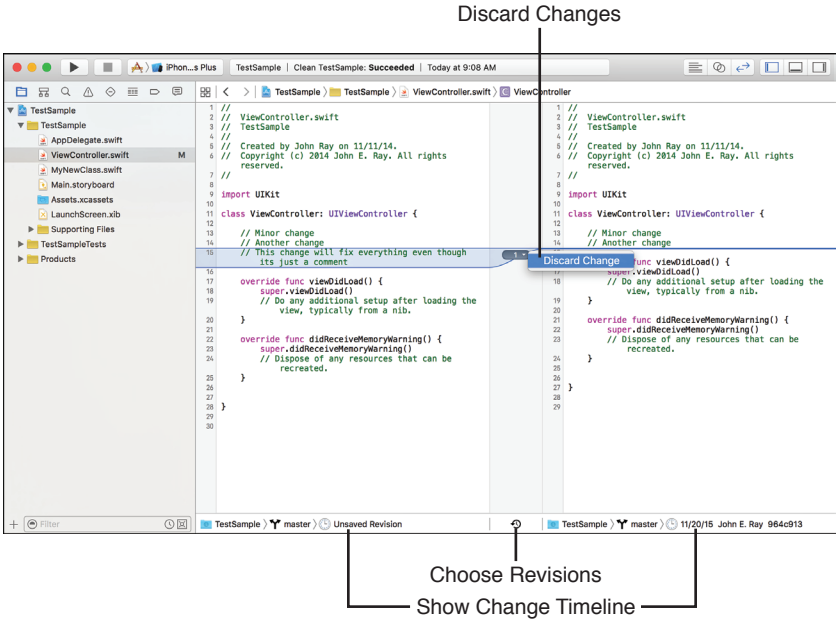

FIGURE A.12
Use Comparison mode to view changes to files and discard them if needed.

TIP

The Source Control menu also offers options for discarding all changes and discarding changes in a group of selected files within the project navigator.

Branches and Merges

When writing code, you'll finally reach a point (I hope) where you are ready to release a product. At that point, you will likely want to create a new branch of your code that will contain any future work (feature additions, for example). You can still continue to work on the existing copy for bug fixes and so on, but the new branch represents your next release.

Creating Branches

To create a branch in Subversion or Git, follow these steps:

1. Choose Source Control, <Project Name>, New Branch.
2. Provide a name for the branch, as shown in Figure A.13.
3. Click Create.

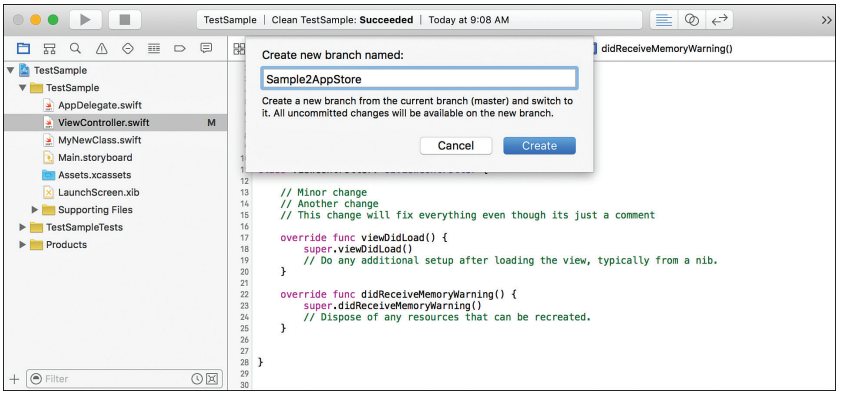

FIGURE A.13
Create a new branch.

Xcode now churns away for a while and then places you in the new branch, ready for editing. You then work within the branch until you reach a point where you are ready to merge it back to your main code base, such as a new release. At that point, you perform a merge.

TIP

You aren't stuck working within a single branch. You can switch between branches, if need be, by choosing Source Control, <Project Name>, Switch to Branch at any time.

Performing Merges

When merging, you can either merge your current working copy *into* a branch or merge a branch *with* your current copy. Assuming you want to merge a branch back to another branch or to the trunk of your repository, follow these steps:

1. Choose Source Control, <Project Name>, Merge from branch.
2. Pick the branch that you are merging with your code, as shown in Figure A.14.
3. Click Merge.

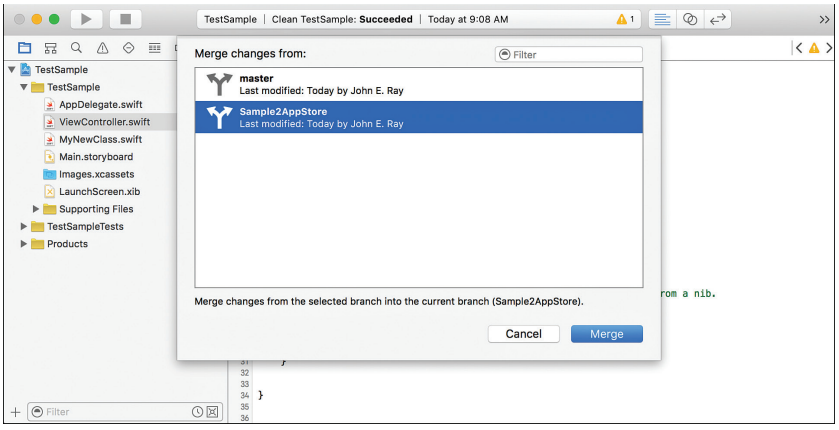

FIGURE A.14
Choose the branch to merge with your code.

Xcode merges the branches together, identifying any conflicting code along the way, as shown in Figure A.15. At this point, you must resolve any conflicts before proceeding.

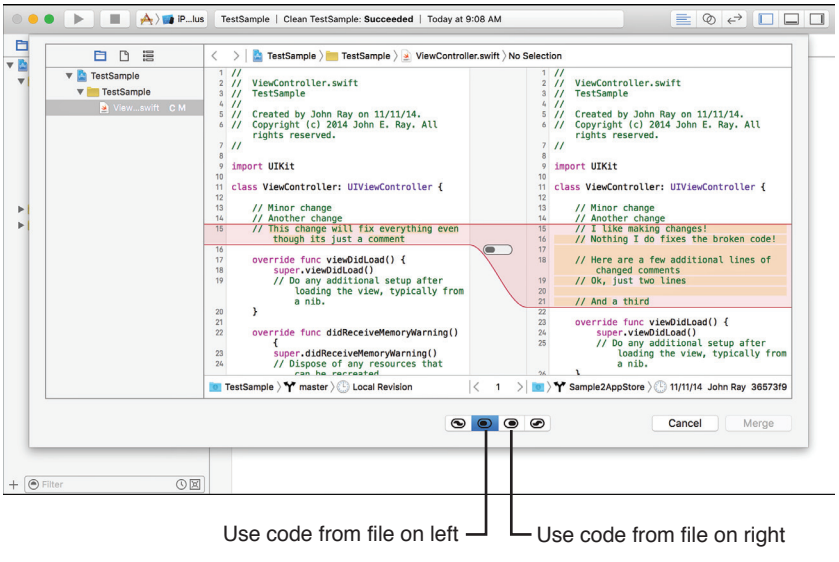

FIGURE A.15
Fix any conflicts in your files.

Work through each file listed on the left side of the display. Using the buttons at the bottom of the screen, choose whether to use the code from the file on the right or the file on the left. When satisfied with your selections, click the Merge button to merge the changes into your repository.

You should then perform a commit (and a push, in Git) to sync the merge to the remote source control servers.

Accessing the *Teach Yourself iOS 9* Tutorials Through Subversion

In the past, I've made the iOS tutorials available in a zip file, but Apple has finally made Xcode's version control friendly (and reliable) enough that even beginning users can access this book's tutorials without downloading a thing.

To do so, just follow these steps:

1. Start Xcode and click the Check Out an Existing Project button at the bottom of the Welcome screen, as shown in Figure A.16. Alternatively, choose Check Out from the Source Control menu.

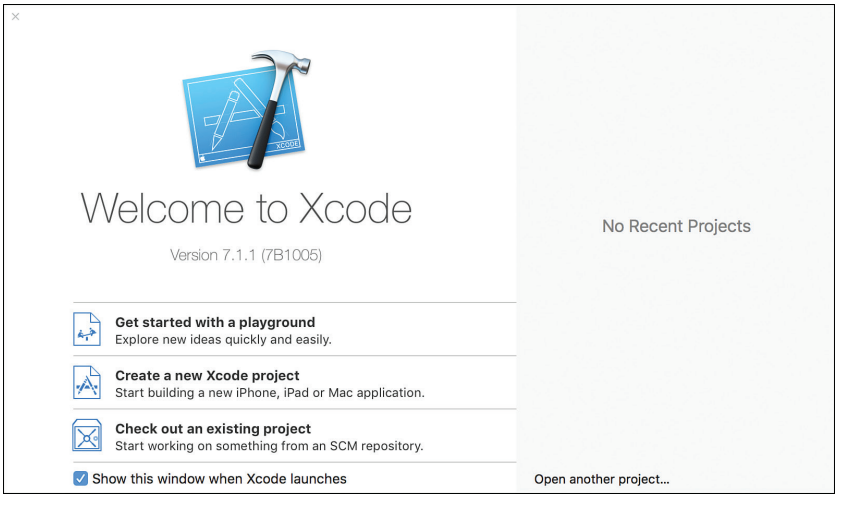

FIGURE A.16
Access the book's tutorials directly from Xcode.

2. If you've already done this before, choose the repository from the list that appears. Otherwise, enter **http://svn.placeforstuff.com/teachyourselfios9** in the Repository Location field, as shown in Figure A.17.
3. Click Next.
4. Click the hour you are interested in, and then click Next.
5. Choose where you want to save the tutorials, as shown in Figure A.18, and then click Download.

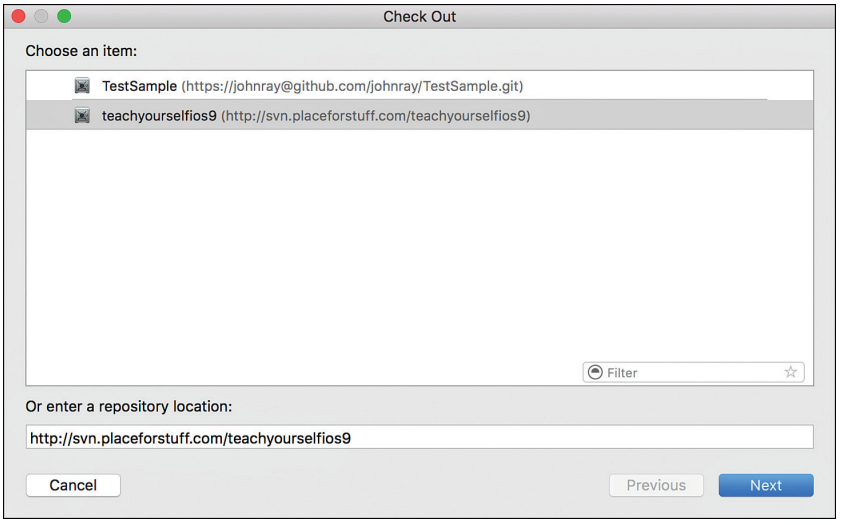

FIGURE A.17
Connect to the tutorial repository.

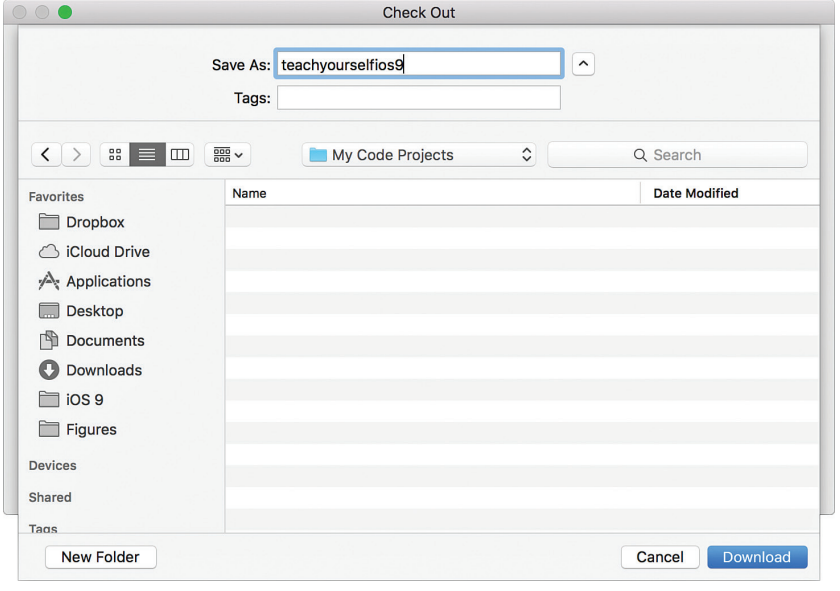

FIGURE A.18
Pick where you want to save the files.

6. Xcode downloads the files to the chosen location, as seen in Figure A.19. (This can take awhile.)

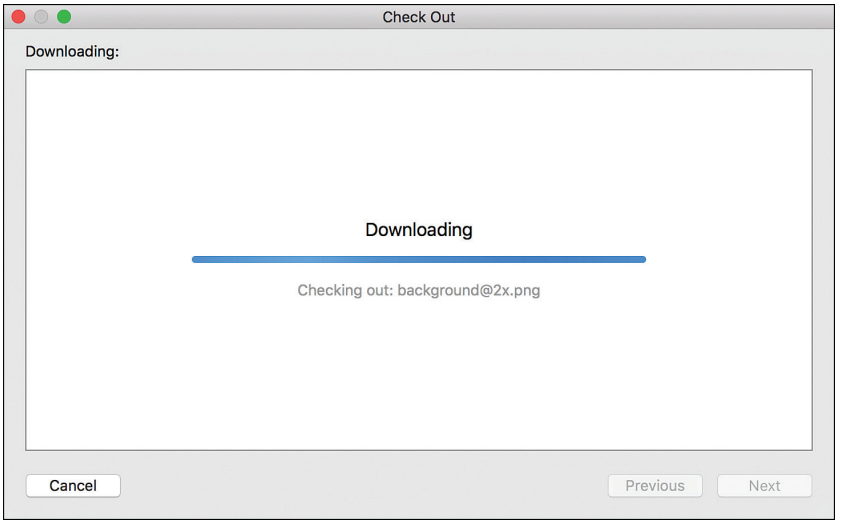

FIGURE A.19
Xcode downloads the project files from the repository.

When the download is finished, navigate in the Finder to the project you want to open, and open it like any other Xcode project.

If you're more comfortable with the zip file download, there's no reason you need to use this approach to access the *Teach Yourself iOS 9* tutorials. This is provided as an opportunity to get your feet wet with Xcode source control. In addition, if there are changes/fixes to a given project, you can update your copy of a single hour's code a bit more easily (use the Source Control, Update menu) than downloading the entire archive again.

Summary

In this appendix, you learned how to use the different source control options in Xcode. Xcode integrates access to Subversion and Git repositories, including local Git repositories that require no additional servers. Using these options, you can easily work with teams to collaboratively code. Although the interface is not as consistent as I would like, the Xcode source control tools do provide most of the features that you need to manage large, distributed projects.