

Chapter 6

Version Control

A version control system tracks the changes made to files and who made the changes. Version control comes in handy on medium to large projects and projects with multiple developers. Each developer can take an individual file, add code to the file, and the version control program records the code each developer added to the file. Even if you're working on a project by yourself, version control can help you. If you've ever mistakenly saved a file and wished you could go back to the way the file was before you saved it, you'll appreciate version control. With version control you can go back to an older version of a file.

Xcode supports three version control systems: CVS, Perforce, and Subversion. I am focusing on Subversion in this book because Subversion ships with Mac OS X and because Subversion is used more than CVS by Mac and iPhone developers. Most of the material applies to all three version control systems so don't despair if you use CVS or Perforce; you can gain a lot by reading this chapter.

To start using version control in Xcode, you must perform the following tasks:

1. Create a repository.
2. Configure the repository so you can access it in Xcode.
3. Open Xcode's repositories window.
4. Import your Xcode project into the repository.
5. Check out your project's files, which gives you a local copy of the project for you to modify.
6. Turn on version control.

Once you get version control working in Xcode, you can use Xcode to perform the following version control tasks:

- See what files aren't up to date.
- Add files to repository.
- Remove files from repository.
- Compare two versions of a file.
- Commit changes to the repository.
- Discard changes you made to a file.
- See all the changes you've made to a file.
- Revert to an old version of a file.
- See who modified a line of code.

Creating a Repository

A repository is where Subversion stores the files you place under version control. There are two types of repositories you can create in Subversion: local and remote. A local repository resides on your computer and will be used only by you. A remote repository allows other people to access the repository and check out the files in the repository from their computers.

Creating a repository is the only basic version control task you cannot perform in Xcode. To create a local repository, launch the Terminal application and run the `svnadmin create` command. Supply a path to the repository. The following command creates a local repository named `MyRepository` on the startup disk

```
svnadmin create /MyRepository
```

Remote Repositories

In Subversion creating a remote repository is the same as creating a local repository: run the `svnadmin create` command. To let other people access the repository, you must setup a Subversion server. You have two options for Subversion servers: use Subversion's `svnserve` server or use a Web server like Apache. A Web server is the better choice if you want other people to access your repository. Setting up a Subversion Web server is beyond the scope of this book, but there are three options to simplify the process for you.

Option 1: If you currently have a website, your hosting company has already set up a Web server for you. All you have to do is create a Subversion repository by running the `svnadmin create` command. Most web hosts have instructions on setting up Subversion on their websites.

Option 2: If you have an open source project, have it hosted on SourceForge or Google Code. They have tools to set up a Subversion repository for your project.

Option 3: If you want to set up a Subversion Web server on your Mac, use MAS. MAS is a Mac open source program that does everything you need to create a Subversion server. You can find a link to MAS on the Xcode Tools Sensei site under Resources.

How Many Repositories Should I Make?

You can create one repository per project or place multiple projects in a repository. The way Subversion updates version numbers may affect your decision on how many projects to place in one repository. Subversion maintains one set of version numbers for the entire repository instead of giving each file its own version number. Subversion updates the version number every time you commit a file to the repository. When you add your first project to

the repository, the files in the project will have version number 1. If you add a file to the repository, that file will have version number 2. If you make changes to the newly added file and commit the changes, the new version will be at version 3. If you add a second project to the repository, the second project's files will have version number 4. Every time the repository changes, Subversion updates the version number.

If you have a lot of projects in one repository, the version numbers can start to get high. You may not want to deal with version number 69472 of a file. Having one project per repository makes keeping track of version numbers easier. Having one repository for all your projects makes maintaining repositories easier. The number of projects you place in one repository is up to you.

Configuring the Repository

After creating your repository, you must configure it so you can use it in Xcode. Launch Xcode, choose **SCM > Repositories**, and click the **Configure** button in the repository window toolbar. The repository configuration window, shown in Figure 6.1 will open. Click the **+** button to add a repository. You will be asked to name the repository and specify what version control system you're using. Choose Subversion unless you've decided to use CVS or Perforce.

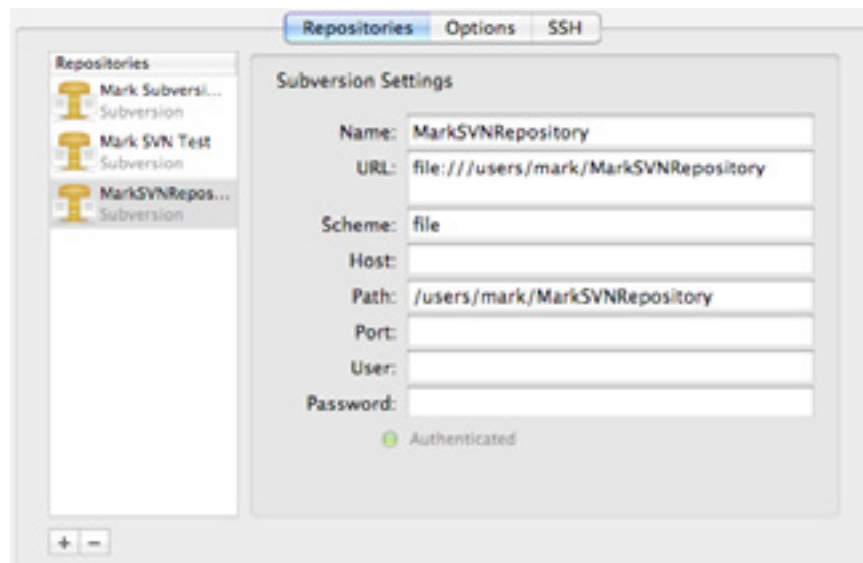


Figure 6.1

Repository configuration window

Specify the URL. The URL depends on the type of repository you want to access. A local repository uses `file://` as the start of the URL.

```
file:///path/to/your/repository
```

A repository hosted on a Web server uses either `http://` or `https://` as the start of the URL.

```
http://svn.example.com
```

A repository set up with the svnserve server uses `svn://` as the start of the URL.

```
svn:///path/to/repository
```

To access a svnserve repository using `ssh`, you would use `svn+ssh://` as the start of the URL.

```
svn+ssh:///path/to/repository
```

Entering the URL will cause Xcode fill in the Scheme, Host, and Path fields for you. There are three additional fields you may have to fill in: Port, User, and Password. The most common case where you would need to enter a port number is when the repository is on a svnserve server. The default port number for svnserve is 3690. If the repository is password-protected, you'll need to enter a username and password.

Options

When you configured the repository, you noticed two other tabs in the configuration window: Options and SSH. The main thing the Options tab lets you control is how Xcode compares two versions of a file. You can also tell Xcode to automatically configure SCM and automatically save files when performing SCM operations on them.

SSH

Clicking the SSH tab shows a list of SSH keys you have on your Mac. You would use this tab if the repository you're accessing uses a svnserve server. If you're using a local repository or a repository hosted on a Web server, you shouldn't have to worry about SSH.

Opening the Repositories Window

Choose **SCM > Repositories** to open Xcode's repositories window, which you can see in Figure 6.2. The repositories window lets you examine the directories inside your version control repositories. Select a repository from the list on the left to examine that repository's directories. Some things you can do from the repositories window is import projects, check files out of the repository, create directories, move directories, and delete directories.

If you plan on storing more than one project in a repository, I recommend creating a directory inside the repository and storing your projects inside that directory. In doing research for this chapter, some versions of Xcode let me add only one directory to the root of the repository. Any additional directories I tried to create were created inside the first directory I created. This directory behavior is why I recommend creating a directory, naming it something like **Projects**, and importing your Xcode projects into the **Projects** directory. Click the **Create Directory** button in the repositories window's toolbar to add a directory to the repository.

Importing Your Project to the Repository

Importing a project adds the project's files to the repository. To import a project in Xcode, you need an Xcode project. Create a project if you haven't already done so. Move the project's files to a folder. If you follow Subversion conventions, you'll go to your project folder in the Finder and add three folders named **branches**, **tags**, and **trunk**. Adding the **branches**, **tags**, and **trunk** folders is not a mandatory step to add a project to a Subversion repository. But doing so will make your life easier if you need to branch your code in the future. Suppose you have a Mac OS X application, and you decide to create Linux and Windows versions. You could use the same Subversion repository, but have three branches: one for the Mac version of your code, one for the Linux version, and one for the Windows version. I recommend adding the **branches**, **tags**, and **trunk** folders just to be safe.

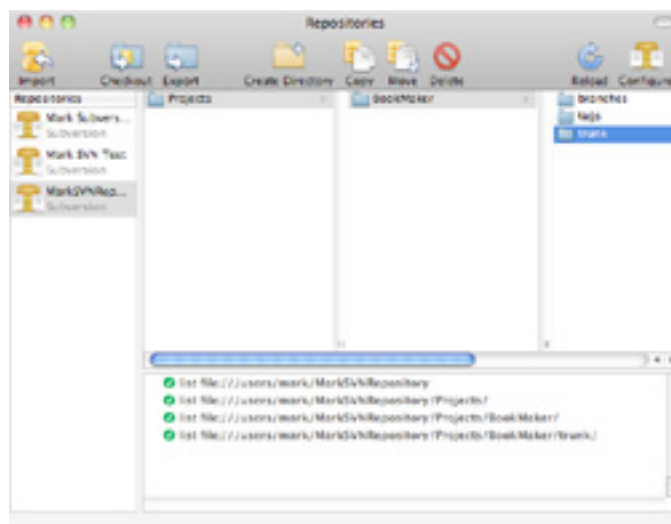


Figure 6.2

Repositories window

If you decide to add the branches, tags, and trunk folders to your project folder, you must move the files in the project folder to the trunk folder. You should move the project's build folder out of the project folder when adding the project to the repository. The build folder contains files Xcode creates when it builds your project, such as object and executable files. The contents of the build folder can become large as you work on your project, and the build folder's contents are files you don't want to place under version control because the files' contents are going to change every time you build your project. A project named MyProject should have the following directory structure prior to import:

```
> MyProject
  > branches
  > tags
  > trunk
    > Everything in your project except the build
      folder. Move the build folder out of MyProject.
```

To import a project, select the directory in the repositories window where you want to import the project. If you read the previous section, "Open the Repositories Window", and followed my advice, this directory will be the Projects directory you created in Xcode's repositories window. Click the Import button. A sheet opens, asking for a directory to import. Navigate to your project folder, type a comment, and click the Import button.

After clicking the Import button, there should be a directory inside the Projects directory with the name of your project. Inside the project name directory should be the branches, tags, and trunk folders. The trunk folder should contain your project's files. If these folders don't show up, click the Reload button to refresh the repositories window.

Checking Out Files

After importing a project, check out the files in your project so you can use them in Xcode. Select your trunk folder in the repositories window and click the Checkout button. A dialog box opens asking you where you want to store the checked out files on your hard drive. Navigate to where you want to store the checked out files. I recommend creating a folder specifically for storing the working copies of your Xcode projects. Click the Checkout button. You'll be asked if you want to open the Xcode project. Open it if you want.

Remember where you checked out the files after performing the checkout. The folder where you checked out the files contains the files that are under version control. The files that are under version control are the files you want to use in Xcode. Why is it so important to remember where you checked out the files?

It's important because after the checkout, you have two copies of your Xcode project and source code files, the one you initially created (Copy A) and the one you checked out (Copy B). Copy B is the one that is under version control. Copy B's project file is the one you want to open in Xcode. If you open Copy A's project file, you won't be able to do any version control operations because that copy isn't under version control. You'll want to delete Copy A to avoid confusion.

Turning on Version Control

When you imported your project and checked out its files, the project should be under version control. But you should make sure version control is turned on before you do any work. You don't want to make a bunch of changes to a source code file and find out version control wasn't turned on for the project.

To turn on version control, open the Xcode project (the project file you checked out, not the original one you created in Xcode). Select the name of the project from the Groups and Files list and click the Info button on the project window toolbar. An inspector will open. Click the General button in the inspector. Click the Configure Roots and SCM button.

When you click the Configure Roots and SCM button, a sheet will open. There will be an entry with the name <Project File Directory> in the Root column. The repository for the Project File Directory should be set to your repository. If the Project File Directory is set to your repository, you're ready to use version control in Xcode.

If the repository says None, open the repositories window by choosing SCM Repositories. Select your repository. Navigate the directories in your repository and make sure you imported the project. If you imported the project, make sure you opened the Xcode project you checked out of the repository. When you check out a project, there are two copies of the project. If you open the wrong copy in Xcode, it won't be under version control.

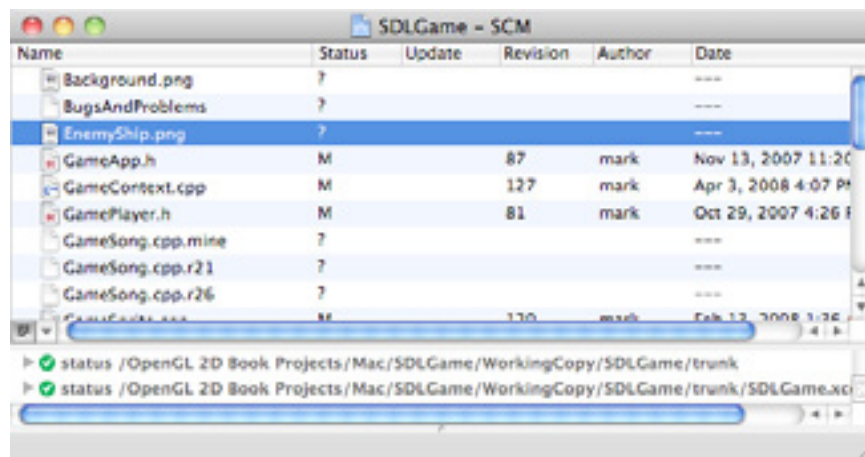
Seeing Which Files Aren't Up to Date

Initially the files in your project match the files in the repository. As you work on your project, editing source files and adding files to the project, some files no longer match. These files are the ones you must update in the repository. To see the list of files that don't match the saved version in the repository, select SCM from the Groups and Files list. The project window lists the files that don't match. The left column has a one-character code representing the file's version control status. Table 6.1 contains a list of status codes.

Table 6.1 Version Control Status Codes

Code	Description
?	The file is not in the repository. When you add a file to your project, it won't be in the repository initially.
-	The file is in a folder that is not in the repository. You must add the folder to the repository from Xcode's repositories window or from the command line.
M	You've modified the file. Choose SCM > Commit Changes to save your changes to the repository
A	To be added. The file will be added to the repository the next time you choose SCM > Commit Changes.
R, D	To be removed. The file will be removed from the repository the next time you choose SCM > Commit Changes.
C	The changes you made to the file may conflict with changes made in the latest version. You would get this code if another programmer modified a file while you had that file checked out.
U	The version of the file you're using is older than the latest version in the repository.
Blank	The file is up to date.

Choosing SCM > SCM Results brings up the SCM detail view, which you can see in Figure 6.3. The SCM detail view tells you the following information about each file in your project:

**Figure 6.3**

SCM detail view

- The status.
- Update. Only files with U for status will have a status in the Update column.
- Revision, the most recent version number in the repository.
- Author, the username of the person who committed the most recent version to the repository
- Date, when the file was last committed.

There are two buttons in the lower left corner of the SCM detail view. The first button toggles showing and hiding the SCM log. The second button filters the files that appear in the SCM detail view. There are three filters.

- All, which shows all files in the project, even the ones that are up to date.
- Interesting, which shows a hierarchical list of out of date files.
- Flat, which shows a non-hierarchical list of out of date files.

Adding Files to the Repository

If you create an Xcode project and check the project's files out, you can add files to the project and add them to the Subversion repository from Xcode. To add a file to the repository, perform the following steps:

1. Select the file you want to add from the project window.
2. Choose **SCM > Add to Repository**. The file you want to add has the status A, which means it's ready to be added to the repository.
3. Choose **SCM > Commit Changes** to add the file to the repository.

Removing Files from the Repository

Although it's more common to add files to a project than to remove them, you can remove files from a Subversion repository from Xcode. To remove a file from the repository, perform the following steps:

1. Select the file you want to remove from the Groups and Files list.
2. Press the Delete key.
3. An alert opens. Pressing the Delete key deletes the reference to the file from the project. The alert asks if you also want to move the file to the trash. If you want to remove the file from the repository, you should move the file to the trash. If you remove just the reference, you will have to run the `svn delete` command from the Terminal to remove the file from the repository.
4. If you told Xcode to move the file to the trash, a second alert opens, asking if you want to remove the file from the SCM repository. Click the Remove button.

5. Select SCM from the Groups and Files list. The file you want to remove should have an D or R next to it saying it's ready to be removed.
6. Select SCM > Commit Changes to remove the file from the repository.

Seeing the Changes You Made to a File

A common source code management task is comparing two revisions of a file. When you commit a change to the repository, you want to see what changed so you can write an informative message when you commit the change.

To see the changes you made to a file, select the file from the project window and choose SCM > Compare With. There will be another submenu. The submenu you will choose most is Latest, which will compare the file with the most recent revision in the repository. If you want to compare the file with an earlier revision, choose Revision instead of Latest. A window with all the revisions of that file will open. Select a revision and click the Compare button.

Committing Changes You Made

You've made changes to one of your source code files. It could be new code, a bug fix, a speed boost, or taking some ugly code that somehow works and cleaning it up. You tested the changes you made, and everything works perfectly. At this point you want to send your changes to the repository, creating a new version of the file in the repository.

To send your changes to the repository, choose SCM > Commit Changes. Xcode prompts you to type in a message describing the changes you've made. After typing the message, click the Commit button to send the changes. Now there's a new version of the file in the repository.

Discarding Changes

Suppose you made some changes in a file and found out the changes didn't work. How do you go back? Choose SCM > Discard Changes. Xcode erases all the changes you made and takes you back to the last version you sent to the repository. Discarding changes is the solution when you save a file and wish you hadn't.

Reverting to an Old Version of a File

If you need to go back to an older version of a file, choose **SCM > Update To > Revision**. A window with all the revisions of that file will open. Select a revision and click the **Update** button. If you revert a file, you will lose any changes you've made to the file that were not committed to the repository. To save the changes you made, commit the changes to the repository before reverting.

If you revert to an old version of a file and want to go back to the latest version, choose **SCM > Update To > Latest**.

Viewing Annotations

When multiple people make multiple changes to a file, it's nice to know who changed what in the file. Annotations perform this vital task, telling you the following information for each line in a source code file:

- The revision of the file that last modified the line of code.
- Who modified the line.
- When that person modified the line.

To view a file's annotations, select the file from the project window and choose **SCM > Get Annotations for > Latest**. To see annotations for an older version of the file, choose **SCM > Get Annotations for > Revision**. A dialog appears with a list of all the versions for the file. Select the version you want and click the **Annotate** button.

Seeing a File's SCM Information

If you want to see all the SCM information for an individual file, use the SCM inspector. To open a file's SCM inspector, select the file from the project window and choose **SCM > Get SCM Info**. Figure 6.4 shows an example of the SCM inspector. When you open the SCM inspector, lists every revision made to the file. Selecting a revision from the inspector fills the bottom area of the inspector with information about that revision of the file, including.

- The file name.
- The revision number.
- Who checked the revision into the repository.
- When the person checked the revision into the repository.
- A message describing the changes made in this revision.

In Subversion the revision number represents the version of the repository, not the version of the file. Subversion updates the revision number every time you make a change to the repository, such as importing a project, adding a file, or creating a new revision of a file.

Under the revision information of the file are four buttons that provide shortcuts to items in Xcode's SCM menu. Selecting a revision of the file and clicking the Update button reverts the file to that revision. Selecting a revision of the file and clicking the Annotate button shows the annotations for that revision.

Selecting a revision of the file and clicking the Compare button shows the differences between that revision of the file and the most recent revision. Selecting two revisions and clicking the Compare button shows the differences between those two revisions of the file. The Diff button works similarly to the Compare button, but when you click the Diff button, it uses the `diff` tool to compare two revisions of a file. What differentiates `diff` is it shows changes made per line in the file.

Most of you will find the Compare button's results easier to read. When you click the Compare button, Xcode highlights what changed in the file. What changed in the file is generally the most important information when comparing two files.

Snapshots

If version control is overkill for your needs, take a look at Xcode's project snapshots. When you take a snapshot of your project, Xcode saves a copy of all the files in the project. If you make a bunch of changes to your code that didn't work and you want to go back, you can revert back to the snapshot you took. You can see what files changed between snapshots and examine the changes made in the changed files, which makes snapshots a lightweight form of version control.

Revision list

Revision
information

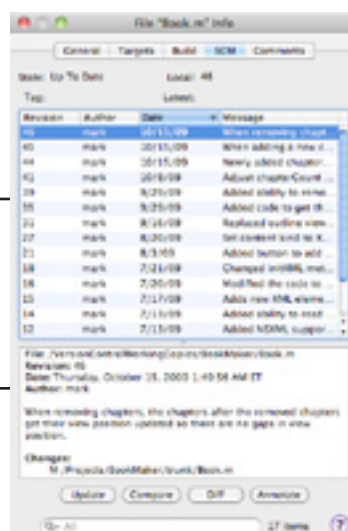


Figure 6.4

SCM inspector

Snapshots have two weaknesses that make them unsuitable as a replacement for version control. The first weakness is they can only be used by one person on their local Mac. If you have a team of developers, they can't use snapshots the way they would use version control. The second weakness is when you take a snapshot, Xcode saves a copy of all the files in the project, even if there were no changes in the files from the previous snapshot. If you take 25 snapshots of a project, there will be 25 copies of each file, which can take up a lot of space. Snapshots are good for solo developers working on small projects and for people to do quick experiments with code. Take a snapshot, experiment, and revert back if the experiment didn't work.

Taking a Snapshot

Taking a snapshot of a project is easy. Open the project and choose File > Make Snapshot.

Looking at a Project's Snapshots

To examine your project's snapshots, open the snapshots window, shown in Figure 6.5, by choosing File > Snapshots. Make sure the project is both open and the currently selected project (if you have multiple projects open).

The top of the window contains a list of snapshots. Selecting a snapshot fills the bottom of the window with information about the snapshot. You can change the snapshot's name and add a comment about the snapshot.

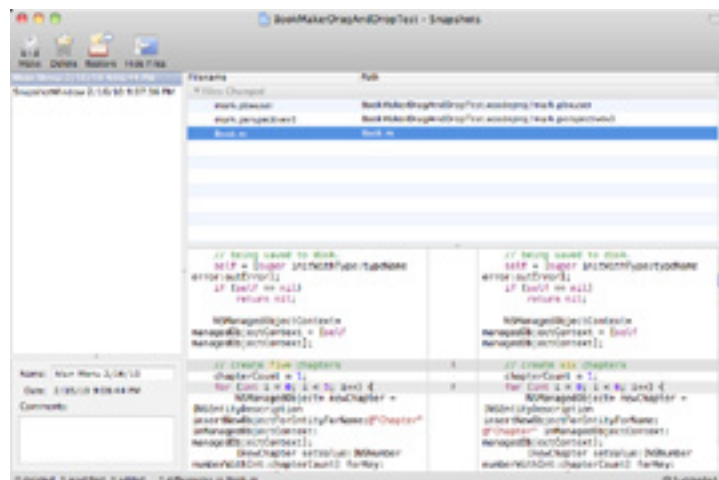


Figure 6.5

Snapshots window

If you want to see the differences between two snapshots, click the Show Files button in the toolbar. Clicking the Show Files button expands the snapshots window. When you select two snapshots, the expanded snapshots window lists the files that changed. Selecting a snapshot lists the files that changed from that snapshot to the most recent snapshot. When you select a changed file, the bottom of the window shows the two versions of the file side by side and highlights the changes made to the file.

To restore an old snapshot, select it from the list and click the Restore button. Xcode will create a snapshot that shows how the project was before you restored it. To restore a single file in a snapshot, select the file from the file list and click the Restore button.

To delete a snapshot, select the snapshot and click the Delete button in the toolbar.

Accessing the Snapshot Repository

The snapshot repository contains all the snapshots you've taken of your Xcode projects. You may need to access the snapshot repository so you can make a backup copy or delete it completely to free up disk space. Snapshots are stored in a file called `SnapshotRepository.sparseimage`. You can find it in the following directory:

```
/Users/YourUsername/Library/Application Support/Developer/  
Shared
```

To access the disk image, you must quit Xcode. Do not try to take snapshots if you're accessing the disk image.