**SVN使用**

<http://www.jianshu.com/p/94498251c7c8>

[源代码管理工具（上）-SVN](http://www.cnblogs.com/wsnb/p/4765597.html)

------------------------------------------------------SVN简介和搭建 ---------------------------------------------------------------

**1.源代码管理工具的起源：**

不进行管理可能存在的问题

无法后悔：做错了一个操作后，没有后悔药可以吃

版本备份：费空间、费时间

版本混乱：因版本备份过多造成混乱，难于找回正确的想要的版本

代码冲突：多人操作同一个文件（团队开发中的常见问题）

权限控制：无法对源代码进行精确的权限控制

追究责任：出现了严重的BUG，无法得知是谁干的，容易耍赖

**2.源代码管理工具的作用：**

能追踪一个项目从诞生一直到定案的过程

记录一个项目的所有内容变化

方便地查阅特定版本的修订情况

**3.现在开始使用源代码管理工具：**

如果是团队开发，使用源代码管理工具是强制性的！

如果是单人开发，也强烈建议现在就开始使用源代码管理工具

由于使用简单，不会增加工作量

不会对现有工作造成任何损害（坏的影响）

是一位合格的软件开发人员必须掌握的技术

**4.常见的源代码管理工具：**

CVS：开启版本控制之门，1990年诞生，“远古时代”的主流源代码管理工具

SVN：集中式版本控制。是CVS的接班人，速度比CVS快，功能比CVS多且强大。在国内使用率非常高（70%~90%）。

GIT：分布式版本控制。目前被越来越多的开源项目使用，不过在国内企业尚未大范围普及。gitHub上的代码就是用git管理的。

**5.基本操作：**

下载服务器代码

修改本地代码

提交本地代码

常见操作：

svn checkout：将服务器代码完整的下载到本地

svn commit：将本地修改的内容提交到服务器

svn update：将服务器最新代码下载到本地

注意：

1.Checkout只需要做一次！

提醒：

每天下班前：commit“可运行版本”

每天上班前：update前一天所有代码

**6.使用环境：**

要想利用SVN管理源代码，必须得有2套环境：

服务器：

用于存储客户端上传的源代码

可以在Windows上安装Visual SVN Server

大部分情况下，公司的开发人员不必亲自搭建SVN服务器

客户端：

上传本地的源代码到服务器，或者更新服务器的代码到本地，保持同步

可以在Mac上使用命令行、Versions、Cornerstone、Xcode

开发人员就属于客户端这个角色

**7.SVN服务器搭建：**

Visual SVN Server下载地址：

http://www.visualsvn.com/server/download/

笔者用的安装包名为VisualSVN-Server-2.7.7.msi，可以看出该安装包的后缀名为.msi的格式，所以该文件必须在windows操作系统上运行。如果你的计算机是Mac的话，那么还需要安装一个虚拟机或者使你的Mac拥有双系统。安装虚拟机的步骤此处就不在赘述。

1.直接双击运行VisualSVN-Server-2.7.7.msi文件：

2.勾选同意，下一步：

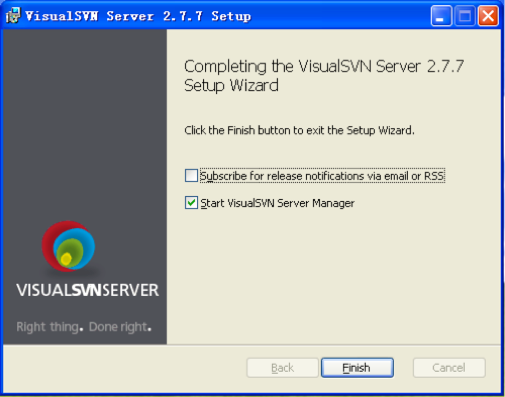
3.如图操作，不要问我为什么：

4.选择标准版：

5.选择服务器初始化路径等信息，并指定连接方式为http而非https，端口号80：

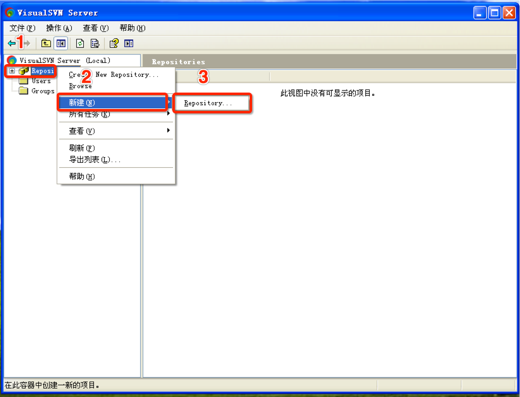
6.点击安装：

7.完成：

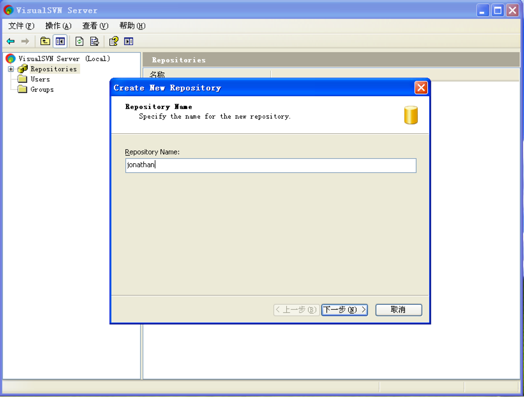


8.SVN服务器解决面简介：

9.创建代码仓库：



10.指定仓库的名称：

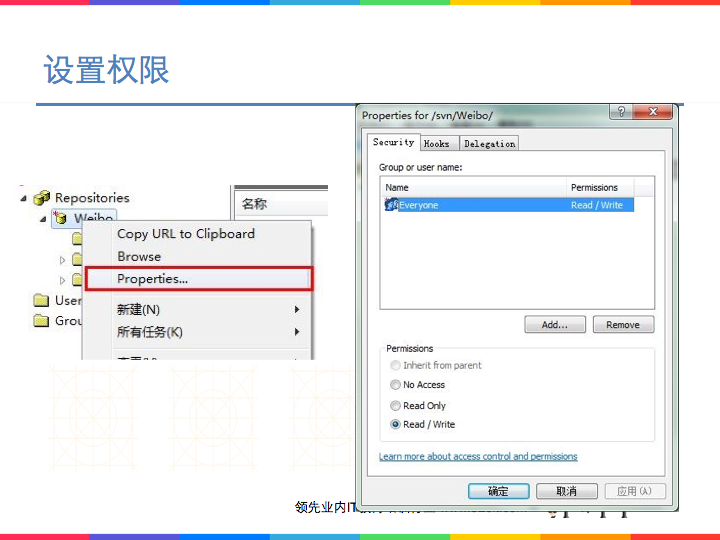


11.指定创建的仓库为最纯洁的仓库：

12.暂时选择第一项，不允许任何人访问：

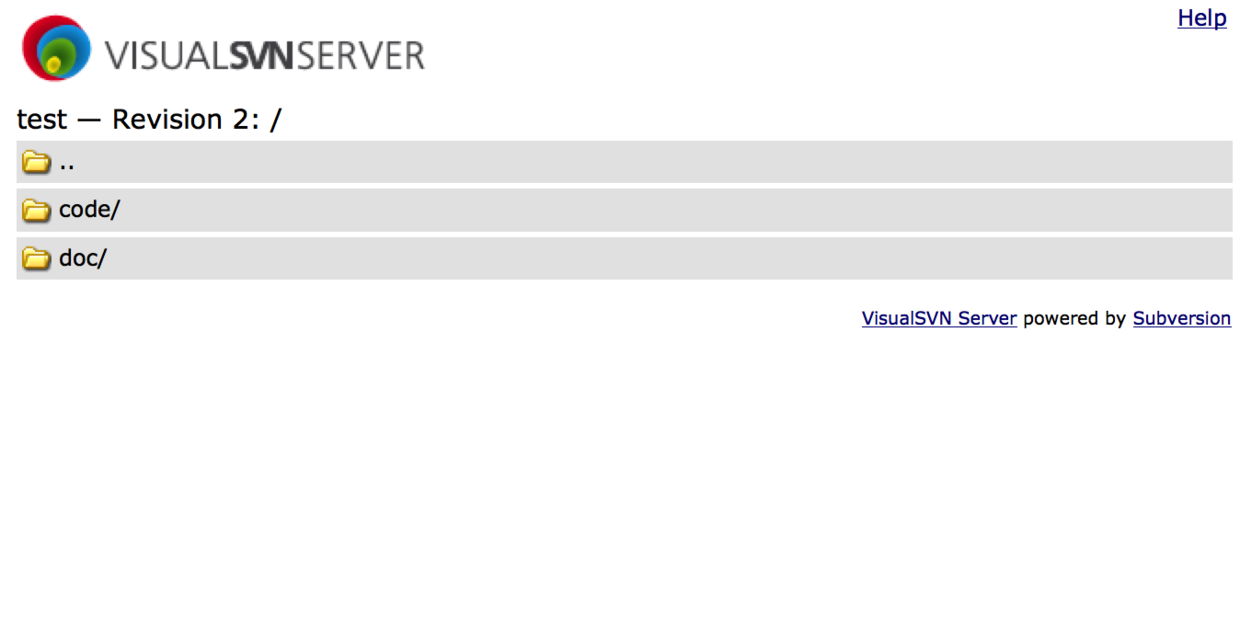
13.创建用户：

14.添加用户，并且指定权限：



15.访问SVN服务器（需要计算机名称换成IP地址才可以在其他计算机访问）：

16.本地访问SVN服务器：



-------------------------------------------------项目经理初始化项目-------------------------------------------------------------

一般情况下，一个新的项目的初始化工作都是由项目经理完成的，项目的初始化工作多数人是不需要做的。但是考虑到大家都是IT界的精英，一不小心就会成为公司的核心公干或者技术总监，所以项目的初始化工作还需要再次介绍一番。

1.将服务器已有的内容下载到本地（比如，在SVN某个项目的仓库里已经创建好的文件夹结构等内容需要下载到本地）

1.svn checkout 服务器地址 --username=mgr --password= mgr（把服务器所有的东西都下载下来）

svn checkout http://192.168.155.128/svn/test/ --username=mgr --password= mgr 把svn服务器的test仓库下载到本地

// 终端执行命令：

swdeMacBook-Pro:svn sw$ svn checkout http://192.168.155.128/svn/test/ --username=ws --password=wsA    test/doc

A    test/code

Checked out revision2.

swdeMacBook-Pro:svn sw$

之后，客户端计算机的文件夹下除了上述两个文件夹外，还会有一个.svn文件。.svn是记录代码所有的变化的。

不能够修改或者删除.svn文件，如果删除，则本地代码不在svn管理之下，不能从服务器更新代码，也不能上传代码。

2.项目经理初始化项目

svn add 文件名/项目名 把文件/项目添加到svn管理之下

svn commit -m “注释” -文件名 例如： svn commit  -m"提交文件" main.m注意：-m是注释的意思

3.查看文件/项目状态。文件是否在svn管理下/文件是否修改了而没有上传（命令：svn status）

“?”代表：文件没有被添加到本地版本库中。

“A”代表：该文件已经在svn管理之下，但是该文件没有被提交到svn服务器

“m”代表：修改了本地文件，但是还没有上传到svn服务器 所以需要再次commit一次

注意：本地文件每被commit到svn服务器上一次，版本号就会加一。

swdeMacBook-Pro:test sw$ cd code

swdeMacBook-Pro:code sw$ svn status?项目经理初始化项目

swdeMacBook-Pro:code sw$

添加到svn管理之下：

// 部分截图：

swdeMacBook-Pro:code sw$ svn add 项目经理初始化项目/A        项目经理初始化项目

A        项目经理初始化项目/项目经理初始化项目.xcodeproj

如何避免多人开发产生冲突：

每次修改代码或者提交代码之前，先update，然后再commit。

常用到的两个命令：

svn update和svn commit

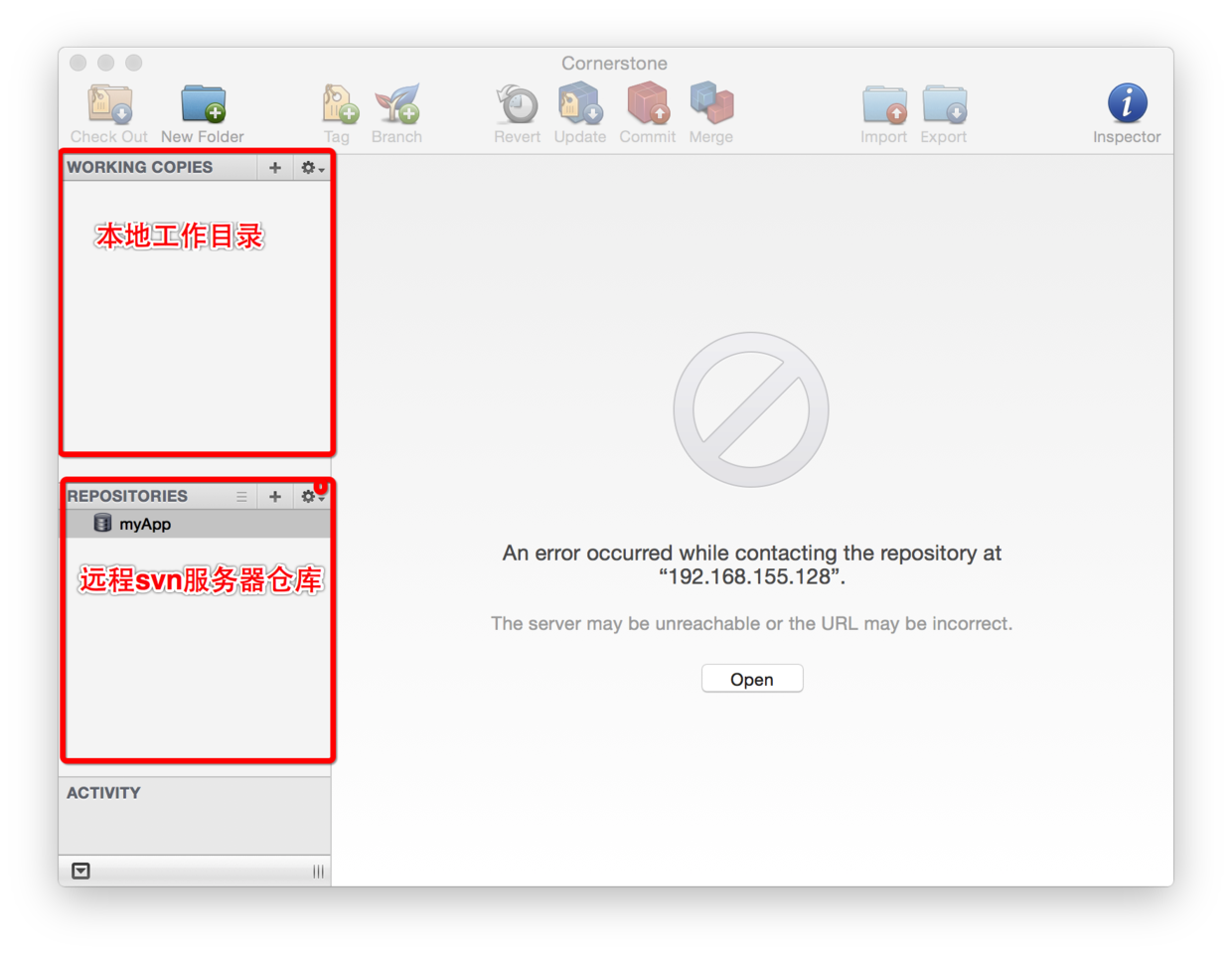
---------------------------------------------------cornerstone简介和使用 --------------------------------------------------

其实，用命令行进行项目的初始化存在很多弊端，不能够忽略一些文件（至少现在我还不会用命令行去忽略文件，即便能忽略的话，那也会是很麻烦的，至于为什么要忽略文件，稍后就会明了），所以，开发中几乎所有的开发人员都是用图形化界面工具cornerstone去管理源代码。下面就针对于cornerstone的安装和使用进行讲解。

在Mac上，也可以利用SVN图形界面工具来管理源代码，可以大大减小使用命令行的痛苦（有些操作使用命令行会比较繁琐，比如解决冲突）

Cornerstone / Versions / Xcode

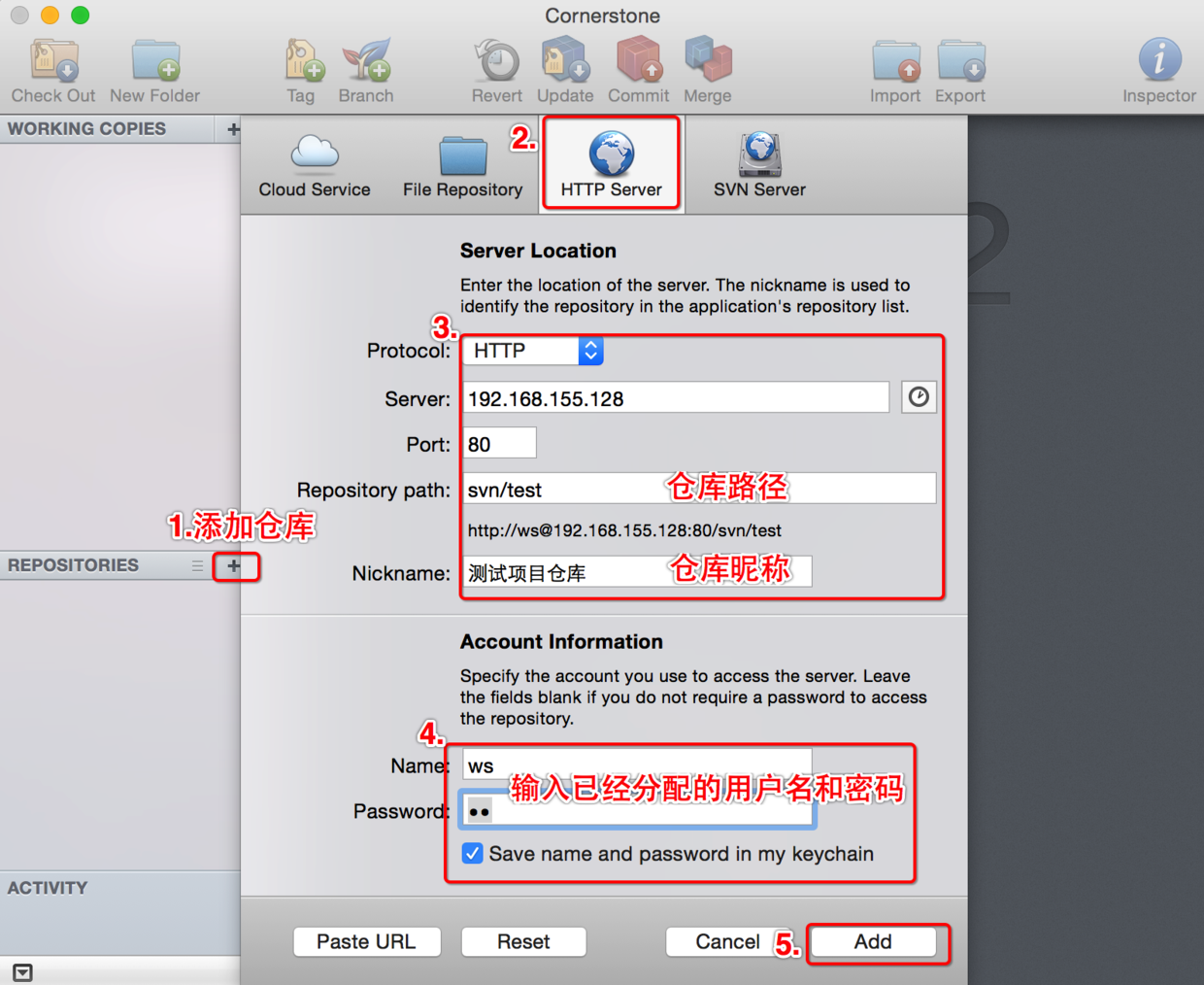
安装完cornerstone后，双击进入cornerstone图形化界面：



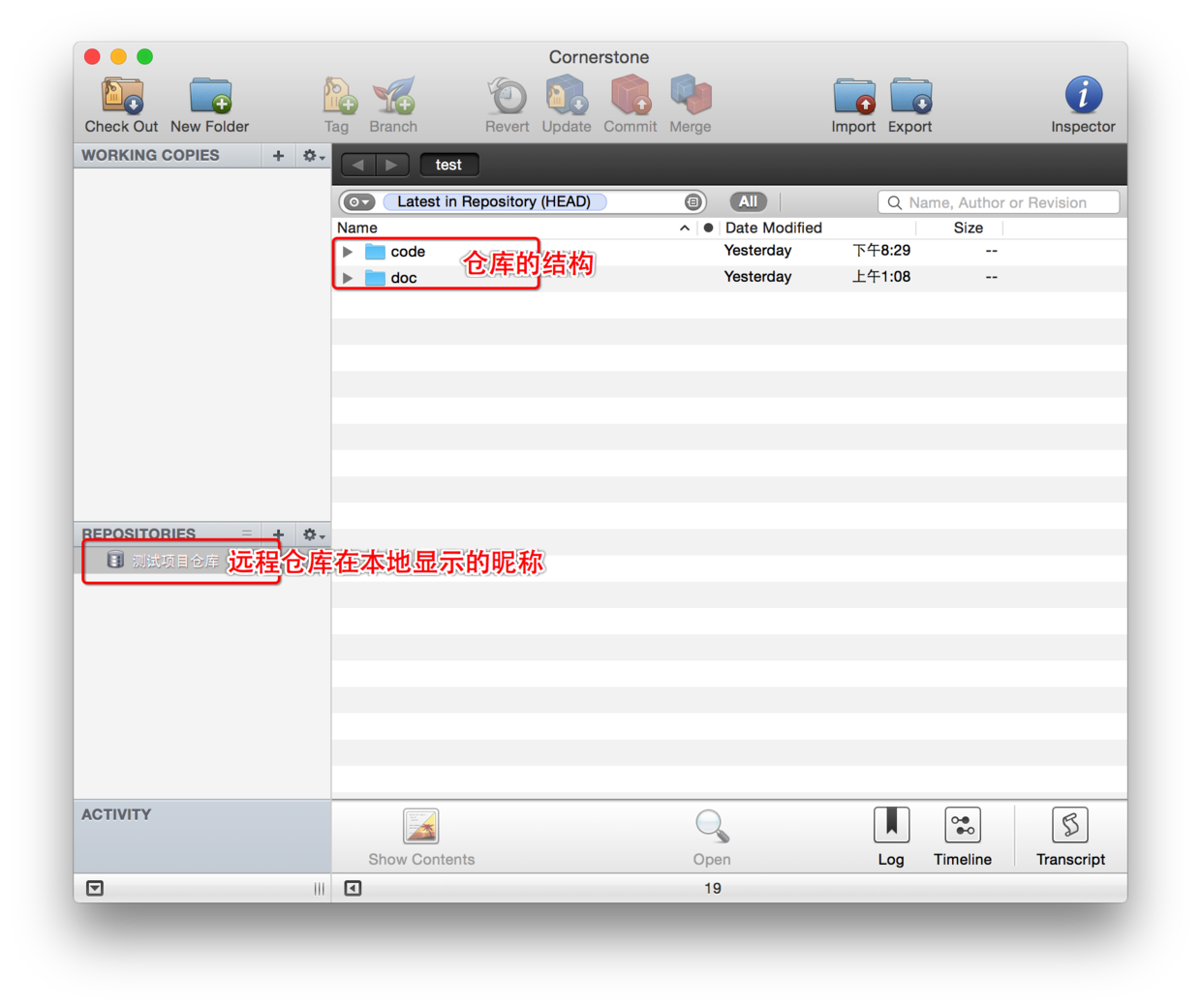
上图中“远程svn服务器仓库”其实是和svn服务器建立连接后的一个映射。

接下来，项目经理需要先把服务器完整的目录下载到本地，然后初始化项目。

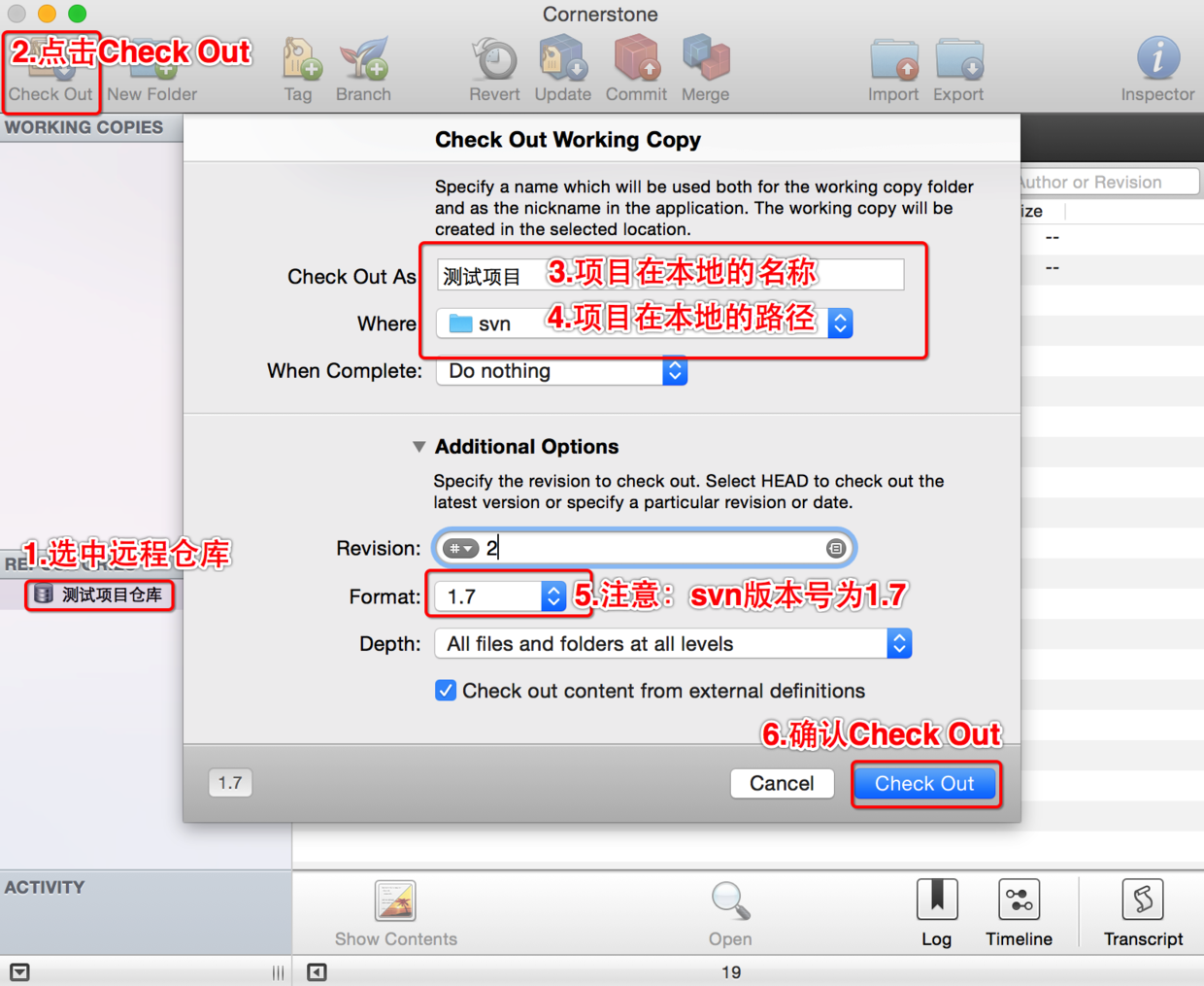
1.连接仓库：



2.连接成功：



3.点击“Checkout”，下载仓库所有的内容（本例中是指下载远程仓库的目录结构）：

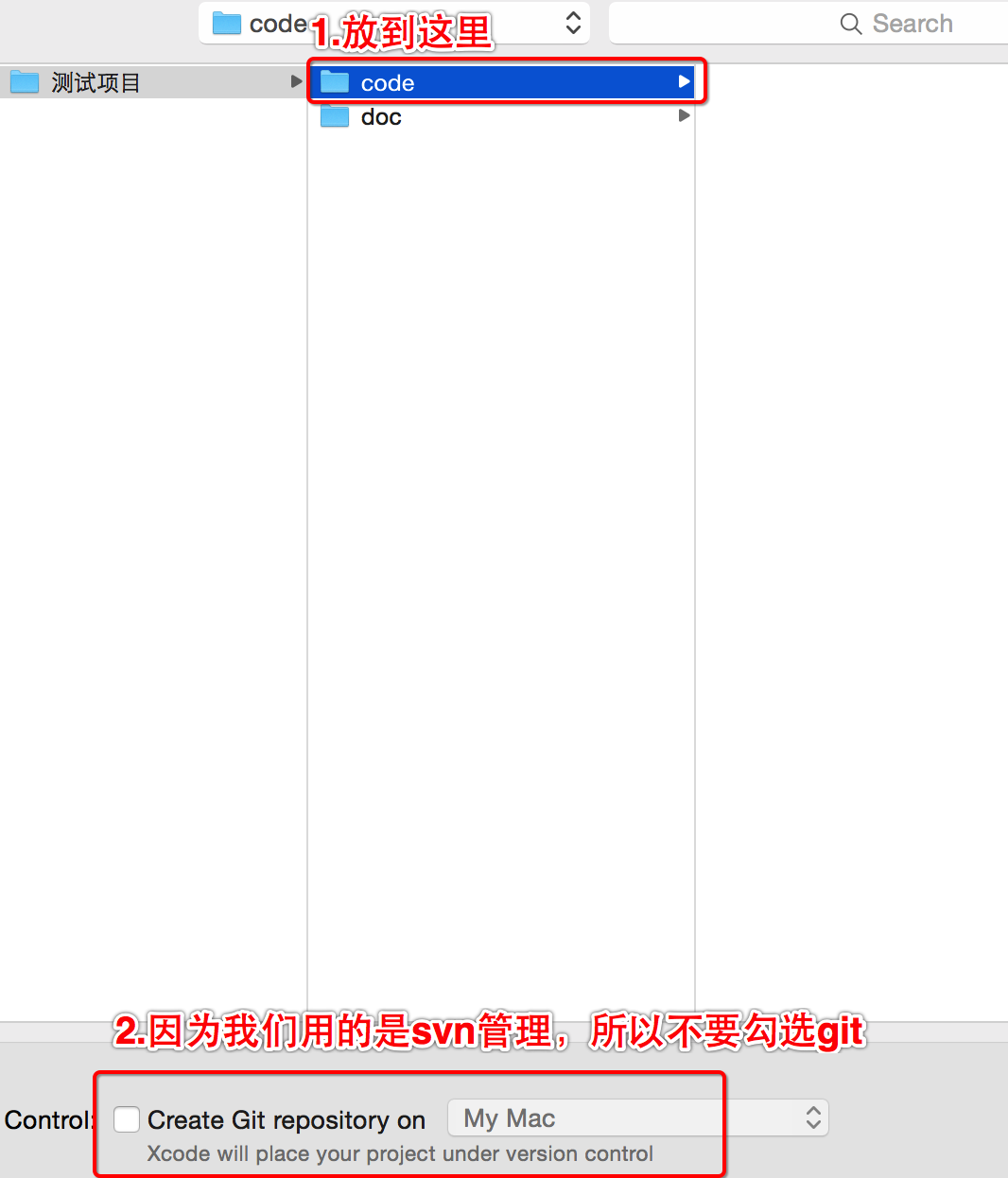


4.此时，“本地svn目录”下已经有了远程svn服务器test仓库的所有内容（code文件夹和doc文件夹）：

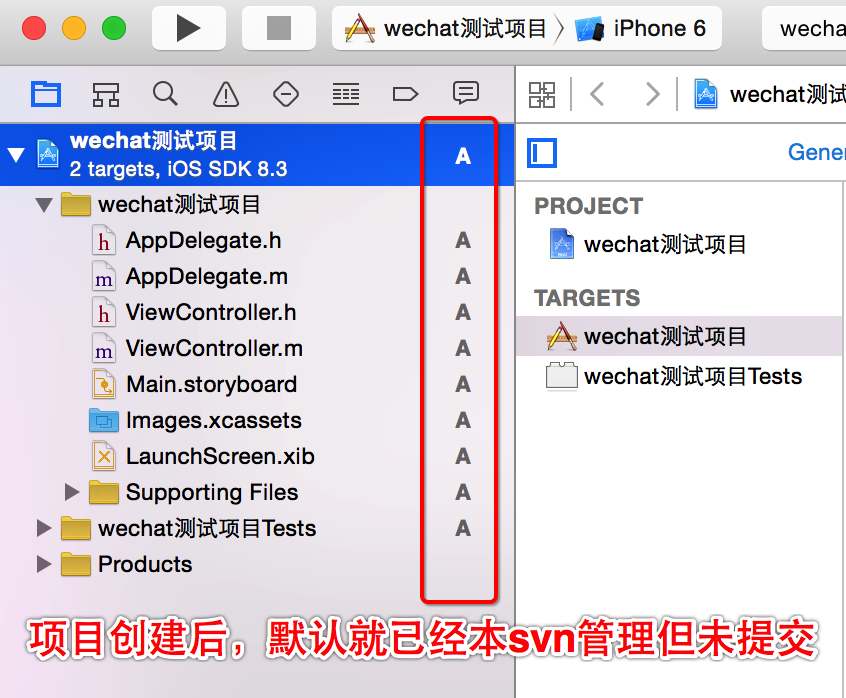


---------------------------------------------------项目经理初始化项目 --------------------------------------------------

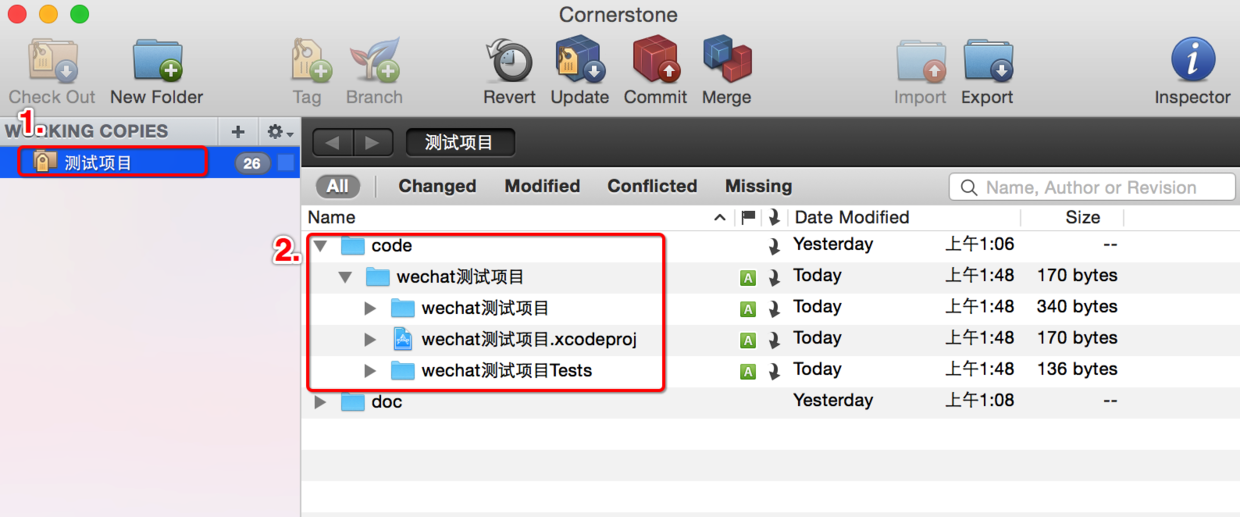
1.用Xcode创建一个项目，把项目添加到code文件夹下：



2.默认项目创建好后，所有相关文件的状态都变为了A(Add)，代表已经在svn管理之下，但是未提交到svn服务器。



cornerstone上的本地目录（working copies）已经有了项目文件：

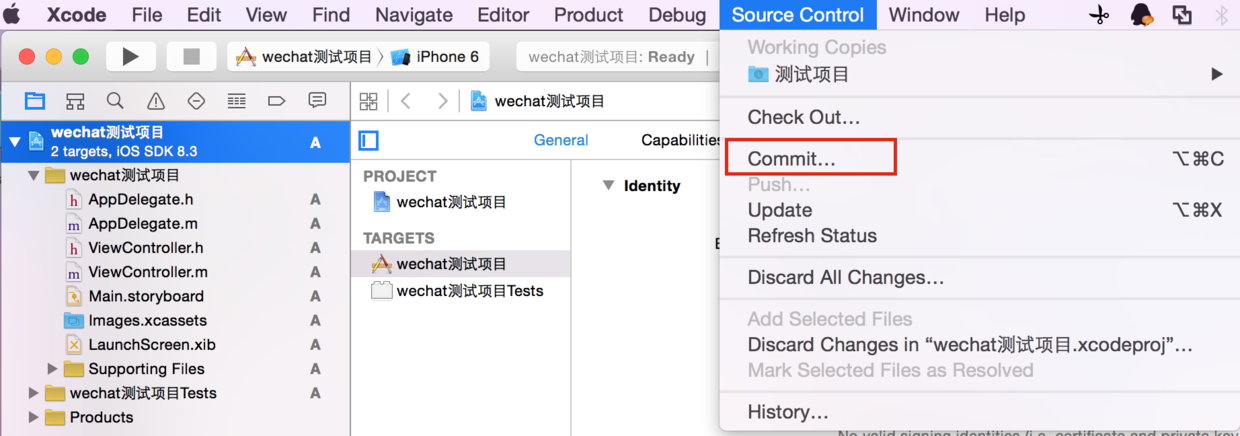


然而，此时远程代码仓库仍然没有代码，下一步需要commit本地代码到远程代码仓库。

cocornerstone界面刷新服务器代码仓库快捷键：command + r

3.提交本地代码到远程仓库：

方法一：用Xcode commit：

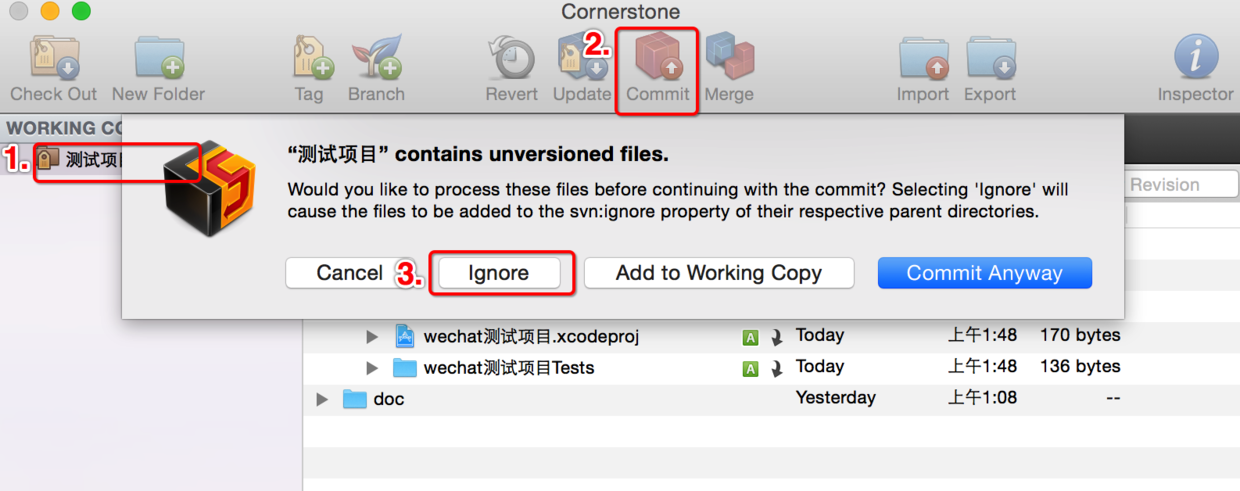


方法二：用cornerstone commit：

一般，初始化项目的时候，第一次commit项目，不在Xcode中提交项目，因为，

除了默认的需要忽略的文件，还需要手动的忽略一些不需要上传到svn服务器代码仓库共享的文件，Xcode做不到这一点。cornerstone能够帮助项目经理手动忽略一些文件，所以，此处我们用cornerstone演示怎么手动忽略不需要上传的文件。

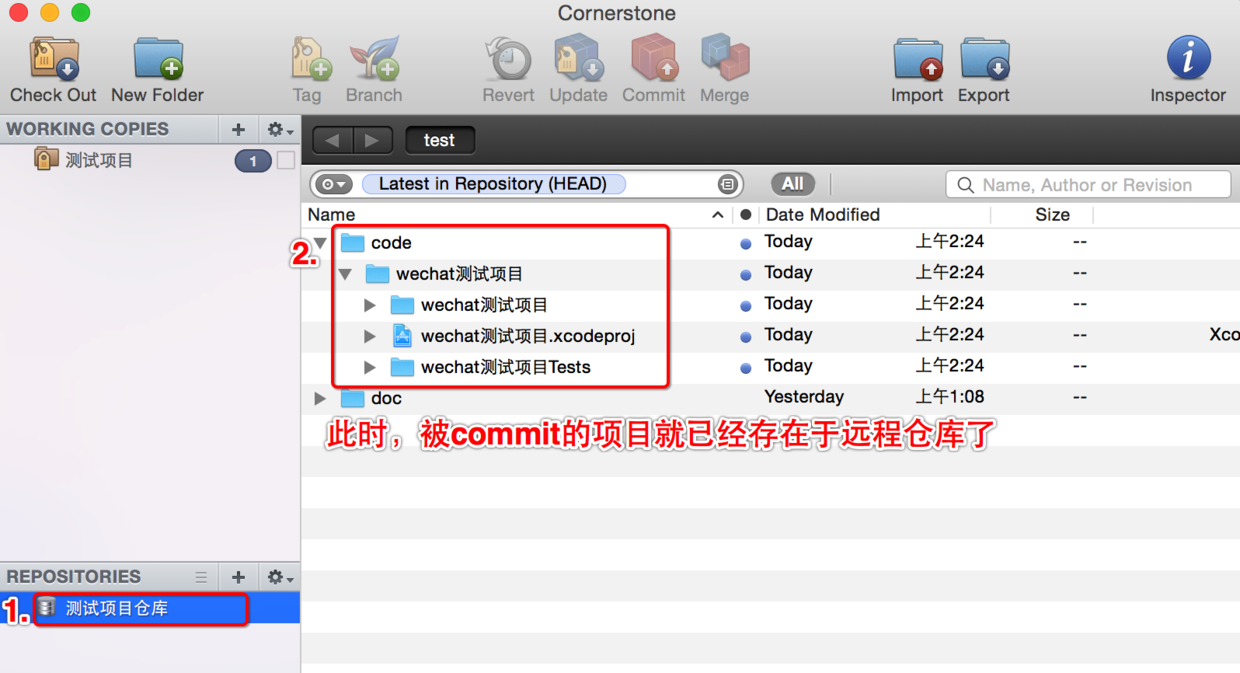
cornerstone会自动帮我们识别，哪些文件不需要提交，哪些需要提交。



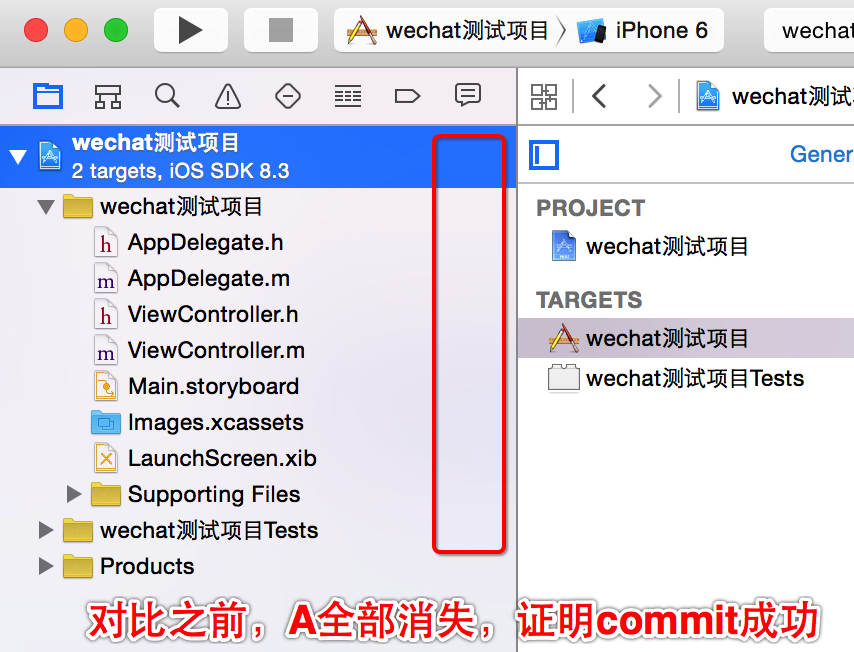
4.添加注释，提交初始化项目到服务器（不需要改变勾选项，cornerstone默认会忽略不需要上传的文件）



5.查看远程仓库是否已经有了被commit的项目：

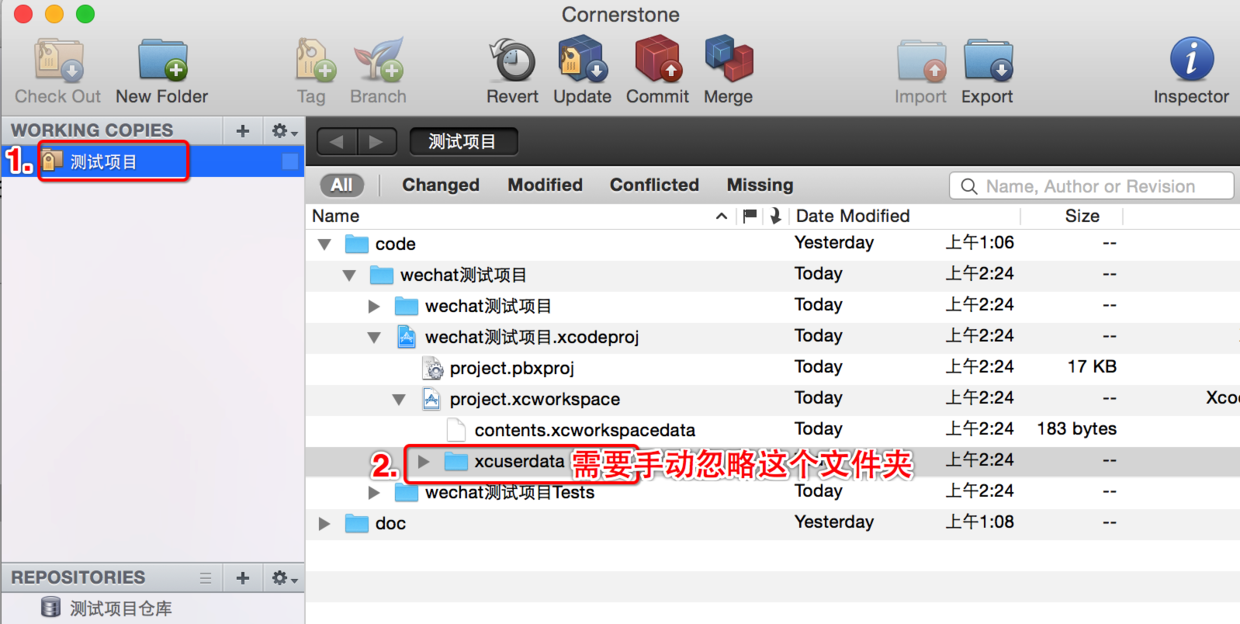


当然，被commit成功的文件，其在Xcode中的A(Add)标记也会消失：

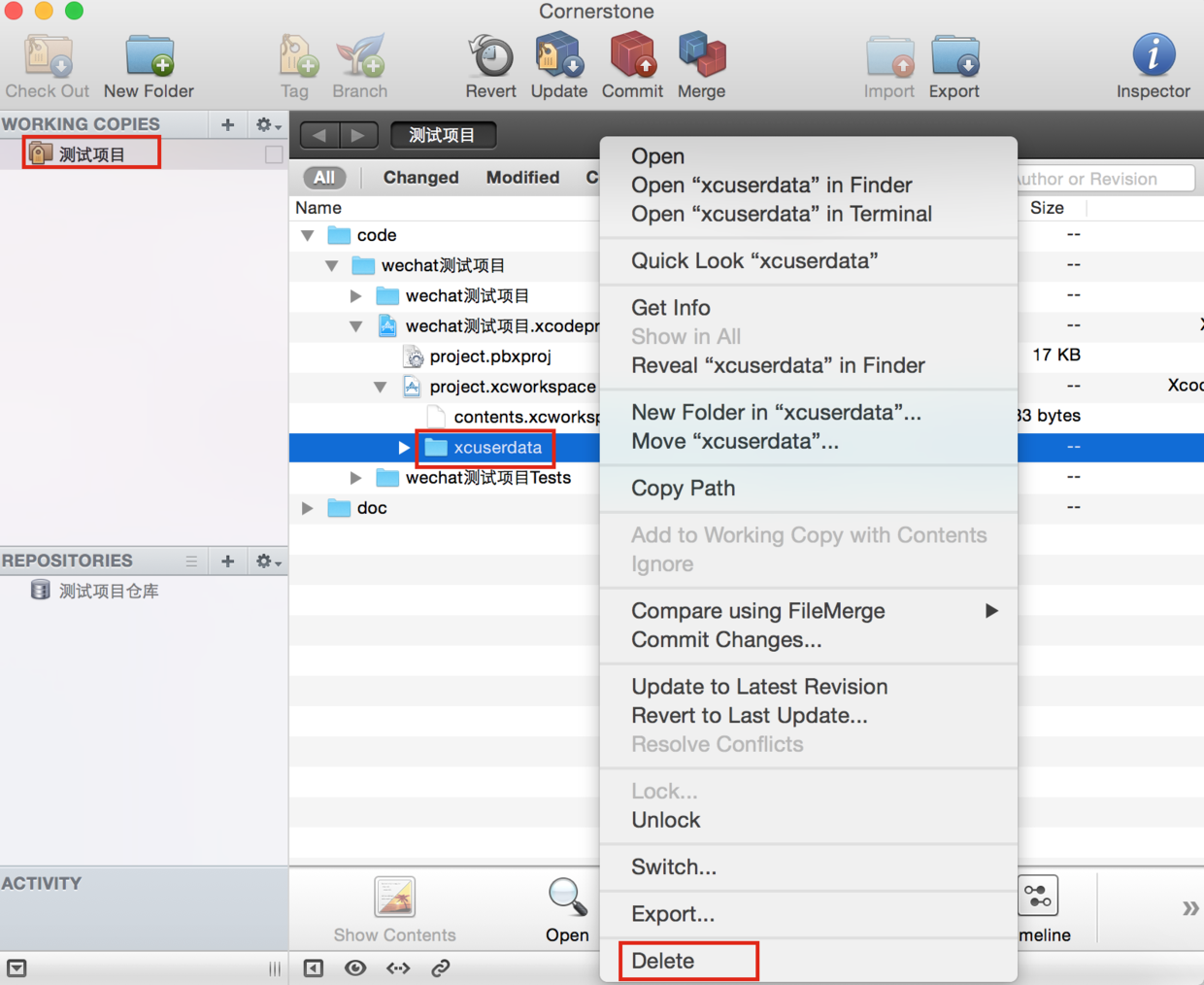


6.此时，初始化项目工作还没有完成，还需要手动忽略一些cornerstone么有帮助我们忽略的文件

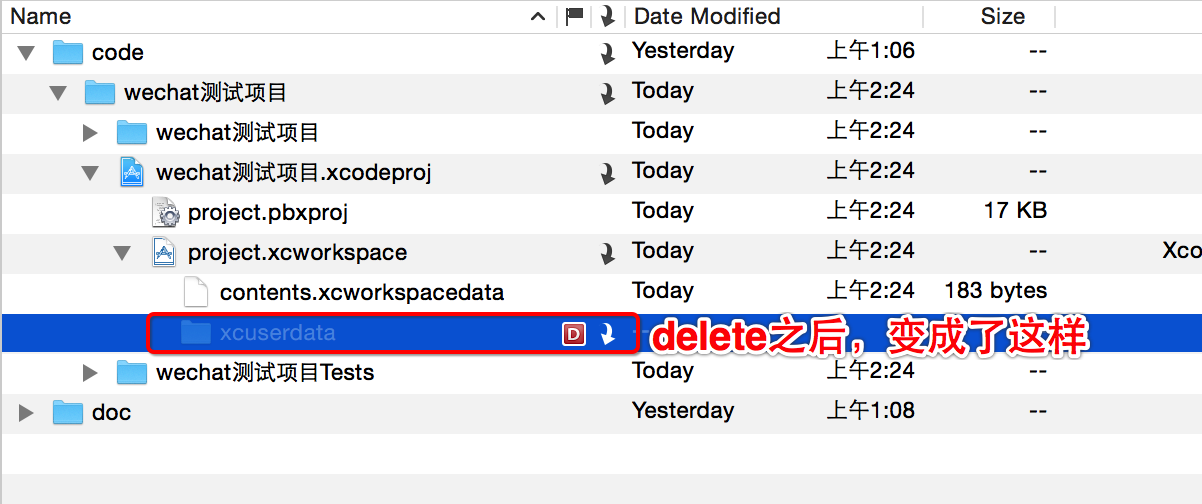
需要忽略的文件如下：



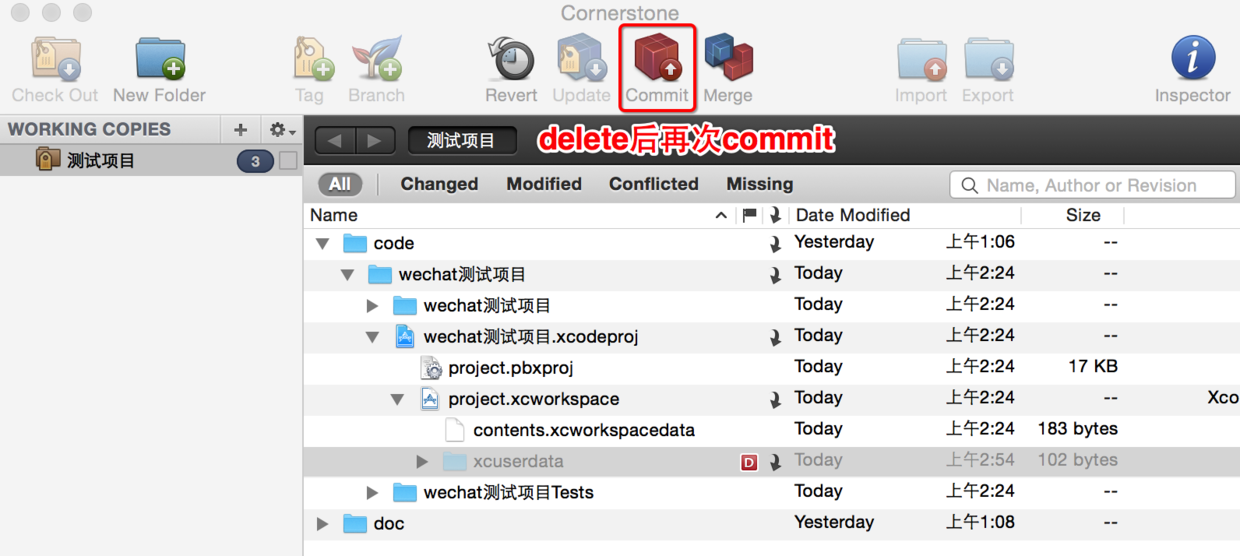
7.先delete需要忽略的文件夹：



delete之后：



8.再commit一次该项目（因为在delete之前，我们已经commit过一次项目，所以需要再次commit删除了文件夹后的项目，更新远程仓库，使之保持同步）。PS：一般commit不成功，会报错，这时，需要先update，然后再commit。



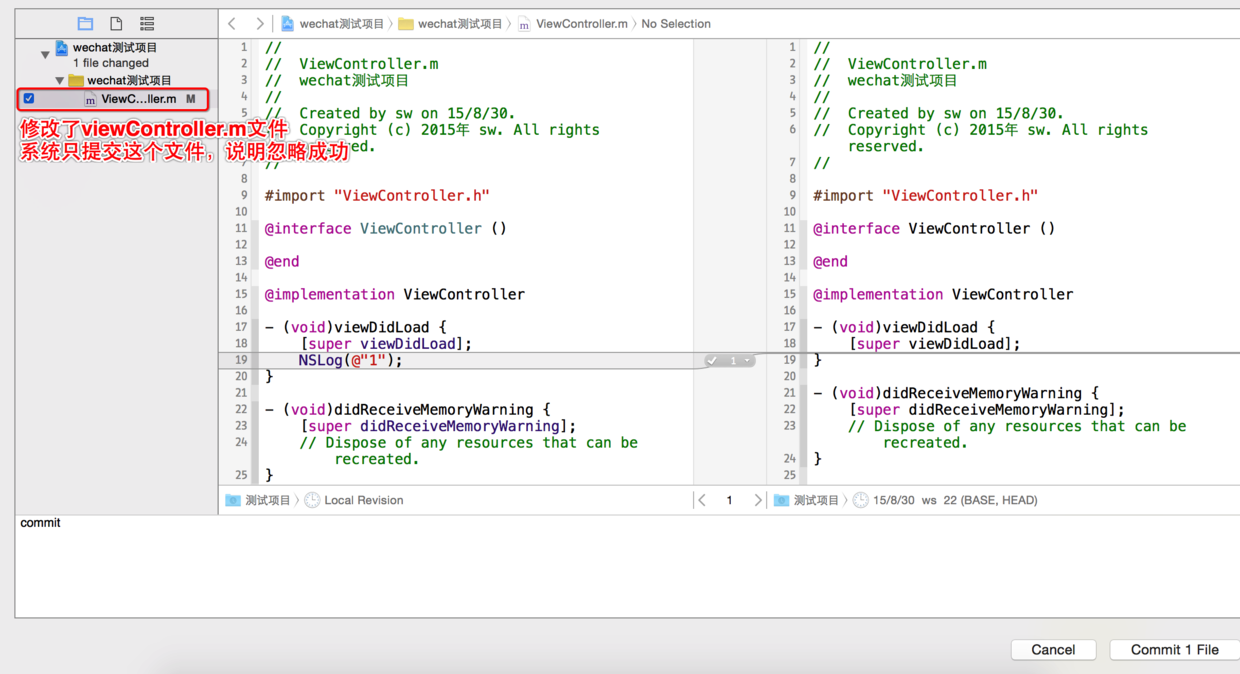
9.然后再右键需要忽略的文件（夹），点击ignore



10.最后再来一次commit（前后一共3次commit）

怎么验证所有需要忽略的文件都已经全部忽略掉？

只修改了一个文件，如果commit的时候系统也只是commit被修改的那一个文件，说明忽略成功（所有需要忽略的文件都已经被忽略），如果修改了一个文件，commit的时候系统却需要commit多个文件，说明还有一些文件没有被忽略。



总结：

需要忽略的文件：

1.Xcode默认会记住我们当前正在编辑的文件，也就是上次退出项目时停留的文件，下次打开项目会停留在该文件。我们不需要把这些信息共享给同事，所以需要忽略。

2.Xcode会记录目录的打开情况。同事不需要共享。

3.断点信息。同事不需要共享。

以上三个需要忽略的信息，都在xcuserdata 文件夹下，所以只需要忽略xcuserdata文件夹。

至此，项目经理用cornerstone初始化项目，忽略文件等操作已经完成。

—————————————————————cornerstone的多人开发——————————————————————————

因为cornerstone根Xcode相比，能够更好的忽略文件，所以除了项目经理第一次初始化项目的时候需要用到cornerstone，其他时候用Xcode的source control的命令就可以进行源码的update和commit。

新人加入：

1.新人需要做的第一件事情

和项目经理要svn的服务器地址/svn账号和密码

产品需求文档

接口文档

效果图

2.下载服务器的源代码

svn checkout

新人刚到公司，项目经理不会分配write代码的权限，所以只能先阅读公司的代码

拿到账号密码后，check out源码

如果新人修改了代码并且也commit了

其他员工需要update

新员工给项目增加了一个类，此时这个类文件变成了A（Add），此时需要commit，但是project setting也需要被提交。因为project setting记录了目录的结构情况，添加、删除、移动文件位置影响了目录结构，所以也需要提交project setting，不能忽略。

——————————————————————解决冲突—————————————————————————

两个人前后都改了同一个位置的代码，先改代码的人先commit。后改代码的没有update就commit，此时就会产生冲突。例如：

经理->update->修改了第99行代码->commit；张三->修改了第99行代码->commit。此时就会冲突。

——————————————————Xcode中svn使用注意---添加静态库—————————————————————

注意：静态库拖拽到项目中后，.a文件默认是I（ignore）的，但是.a文件是需要参与编译的，所以需要用命令行把.a文件添加到项目中，然后commit。

——————————————————Xcode中svn使用注意---storyBoard—————————————————————————

在原来的Xcode版本中，svn对storyBoard的支持非常不好，鼠标点击一下xib或者storyBoard，xib或者storyBoard就会变成M。现在6.0版本之后，可以点击，但是不可以移动，如果移动xib或者storyBoard也会变为M。因为storyBoard本质是XML。

多个人同时修改storyBoard的后果，storyBoard冲突，导致storyBoard打不开。

和代码冲突类似，如果先往storyBoard的同一个位置添加了一个控件，并且后者在不知道前者在同一个位置添加了控件的情况下也再该位置添加了一个控件，那么两个控件在storyBoard上的位置冲突，后者commit会报错。

建议：尽量使用xib，这样可以避免多个人同时操作storyBoard。

——————————————————Xcode的checkOut和svn使用技巧—————————————————————————

1.尽可能修改文件之前先update文件，写一些代码后就立即提交到服务器

2.尽可能在下班之前半小时就提交代码，这样可以预留出来半小时解决可能存在的冲突

3.修改公共文件之前尽可能和同事说一声，修改完成后让同事及时更新，不要做哑巴式程序员，多沟通才能避免一些不必要的冲突和误会，不仅是体现出来对工作的认真，也是对同事的尊重（哥们我之前工作就一直不爱说，工作很低调，吃过这亏）。

———————————————————————svn目录结构——————————————————————————————

正规项目的SVN目录结构一般有3个文件夹：

trunk：主干，当前开发项目的主目录

branches：分支目录，添加非主线功能时使用，开发测试之后，可以合并到主干项目中

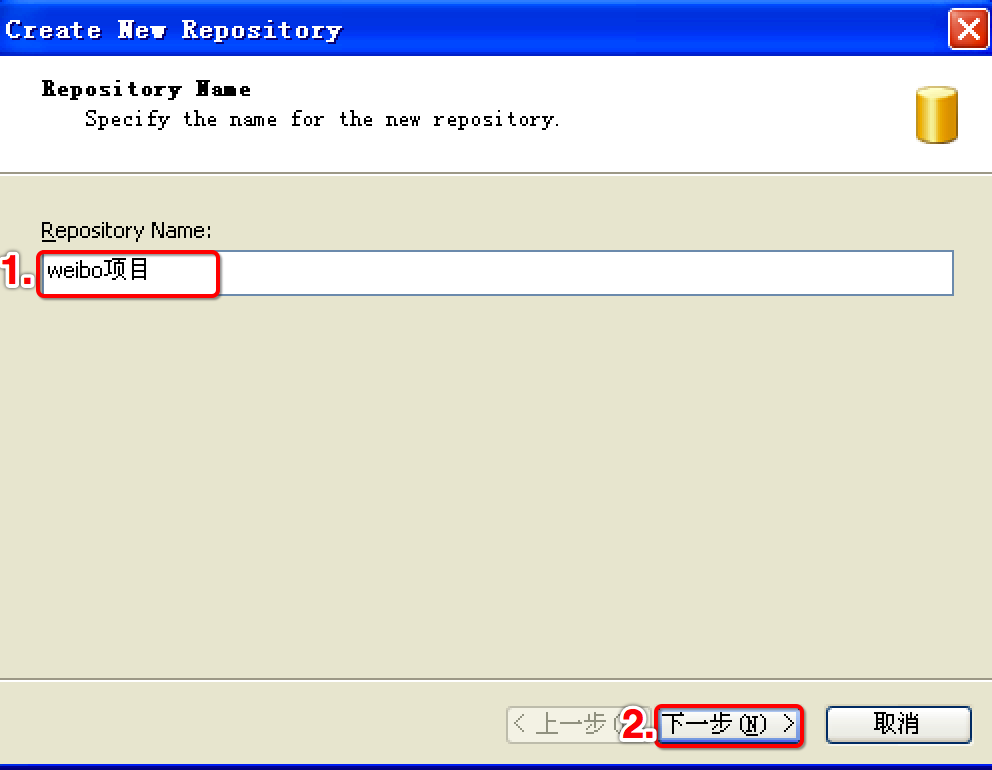
tags：标记目录，通常作为重大版本的备份

在svn服务器上再次创建一个仓库，这个仓库死真正的仓库，包含了trunk、branches、tags三个文件夹，模拟开发、修复bug、合并版本的流程。

1.创建仓库



2.命名仓库



作者：白屏  
链接：http://www.jianshu.com/p/94498251c7c8  
來源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。