

## 版本控制器：SVN

### 1 开发中的实际问题

- 1.1 小明负责的模块就要完成了，就在即将 Release 之前的一瞬间，电脑突然蓝屏，硬盘光荣牺牲！几个月来的努力付之东流——需求之一：**备份**！
- 1.2 这个项目中需要一个很复杂的功能，老王摸索了一个星期终于有眉目了，可是这被改得面目全非的代码已经回不到从前了。什么地方能买到哆啦 A 梦的时光机？需求之二：**代码还原**！
- 1.3 小刚和小强先后从文件服务器上下载了同一个文件：Analysis.java。小刚在 Analysis.java 文件中的第 30 行声明了一个方法，叫 count()，先保存到了文件服务器上；小强在 Analysis.java 文件中的第 50 行声明了一个方法，叫 sum()，也随后保存到了文件服务器上，于是，count()方法就只存在于小刚的记忆中了——需求之三：**协同修改**！
- 1.4 老许是一位项目经理，我会告诉你他把每一个版本都保存一份吗？我会告诉你这些工程里其实有很多文件都是重复的吗？我会告诉你老许为这事删了很多电影吗？——需求之四：**多版本项目文件管理**！
- 1.5 老王是另一位项目经理，每次因为项目进度挨骂之后，他都不知道该扣哪个程序员的工资！就拿这次来说吧，有个该死的 Bug 调试了 30 多个小时才知道是因为相关属性没有应用初始化时赋值！可是二胖、王东、刘流和正经牛都不承认是自己干的！——需求之五：**追溯问题代码的编写人和编写时间**！
- 1.6 小温这两天幸福的如同掉进了蜜罐里，因为他成功的得到了前台 MM 丽丽的芳心，可他郁闷的是这几天总是收到 QA 小组的邮件，要求他修正程序中存在的 Bug，可他自己本地电脑上是没有这些 Bug 的，“难道我的代码被哪个孙子给改了？”。是的，小温没来的时候，丽丽是 QA 小组小郑的女朋友啊！——需求之六：**权限控制**！

### 2 版本控制简介

- 2.1 **版本控制**[Revision control]，最初来源于工程设计领域，是维护工程蓝图的标准做法，能追踪工程蓝图从诞生一直到定案的过程。是一种记录若干文件内容变化，以便将来查阅特定版本修订情况的系统。
- 2.2 Subversion 就是一款实现版本控制的工具软件，通常也称为版本控制器，简称 SVN。Subversion 是 Apache 软件基金会组织下的一个项目。
- 2.3 Subversion 的优良特性
  - ①目录版本控制  
CVS 只能追踪单个文件的历史，但是 Subversion 实现了一个“虚拟”文件系统，可以追踪整个目录树的修改，文件和目录都是版本控制的，结果就是可以在客户端对文件和目录执行移动和复制命令。
  - ②原子提交  
提交要么完全进入版本库，要么一点都没有，这允许开发者以一个逻辑块提交修改。
  - ③版本控制的元数据  
每个文件和目录都有一组附加的“属性”，你可以发明和保存任意的键/值对，属性也会像文件内容一样被纳入版本控制。

## ④可选的网络层

Subversion 在版本库访问方面有一个抽象概念，利于人们去实现新的网络机制，Subversion 的“高级”服务器是 Apache 网络服务器的一个模块，使用 HTTP 的变种协议 WebDAV/DeltaV 通讯，这给了 Subversion 在稳定性和交互性方面很大的好处，可以直接使用服务器的特性，例如认证、授权、传输压缩和版本库浏览等等。也有一个轻型的，单独运行的 Subversion 服务器，这个服务器使用自己的协议，可以轻松的用 SSH 封装。

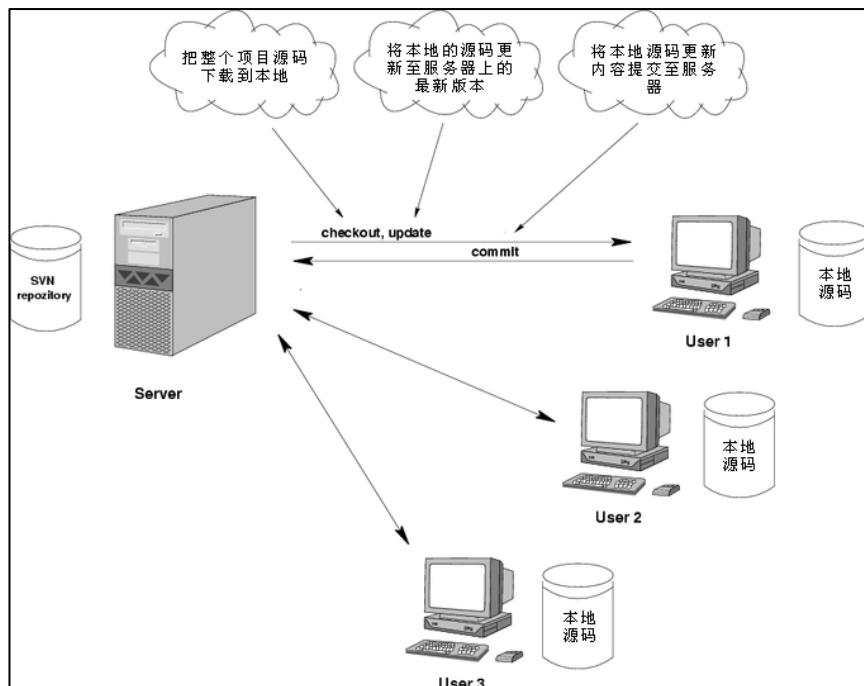
## ⑤一致的数据处理

Subversion 使用二进制文件差异算法展现文件的区别，对于文本(人类可读)和二进制(人类不可读)文件具备一致的操作方式，两种类型的文件都压缩存放在版本库中，差异在网络上双向传递。

## ⑥高效的分支和标签

分支与标签的代价不与工程的大小成比例，Subversion 建立分支与标签时只是复制项目，使用了一种类似于硬链接的机制，因而这类操作通常只会花费很少并且相对固定的时间，以及很小的版本库空间。

- 2.4 SVN 的工作原理：采取**客户端/服务器**模式——在服务器的版本库中保存项目文件的各个版本，所有参与协同开发的程序员在自己本地电脑上保存一个工作副本。SVN 支持程序员将本地副本更新到服务器端的最新版本，也支持将本地副本的最新改变更新到服务器端，而且后面的更新不会覆盖前面的更新，而是作为一个新的版本被保存下来——SVN 甚至支持将本地工作副本恢复为服务器端保存的某一个历史版本。



## 2.5 SVN 基本操作

①**检出**（checkout）：将一个服务器端创建好的项目整个下载到本地，这是到项目组后参与开发的第一步，只需执行一次。

②**更新**（update）：将本地文件更新为服务器端的最新版本，通常为每天上班时

或修改公共文件之前执行一次。

③**提交** (commit): 将本地修改提交到服务器端。通常每天下班前或每实现一个功能、完成一个模块时执行一次。

### 3 Subversion 安装与配置

#### 3.1 安装服务器端程序

##### ①服务器端程序版本

目前 Subversion 的最新版本是 1.9.0-alpha2, 这是一个测试版。官方网站推荐使用的版本是 1.8.9, 原话是: **The best available version of Apache Subversion is: 1.8.9**

##### ②下载源码包

Apache 组织自己维护更新的只是 Subversion 的源码, 各个版本的源码包的下载地址是: <http://subversion.apache.org/download/>

Subversion 源码是使用 C 语言开发的。

##### ③下载二进制安装包

Subversion 在不同平台下的二进制包是由不同组织构建实现的, Windows 平台下的二进制包实现情况如下:

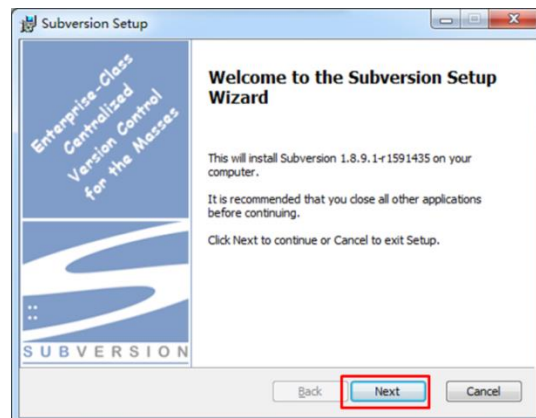


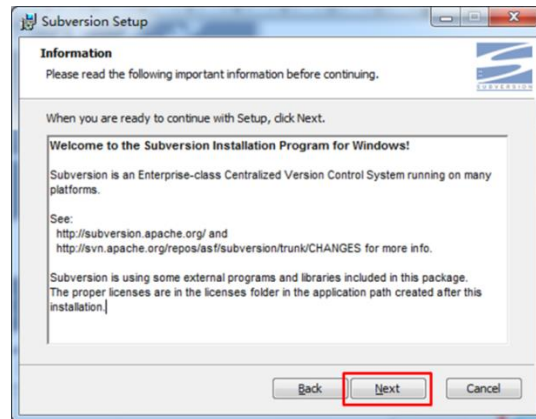
所以, 服务器端程序的下载地址可以使用:

<http://sourceforge.net/projects/win32svn/files/latest/download>

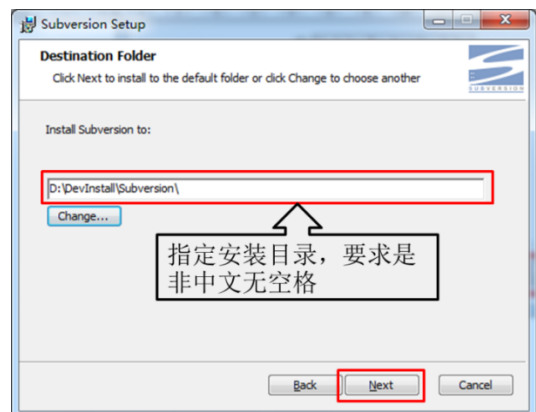
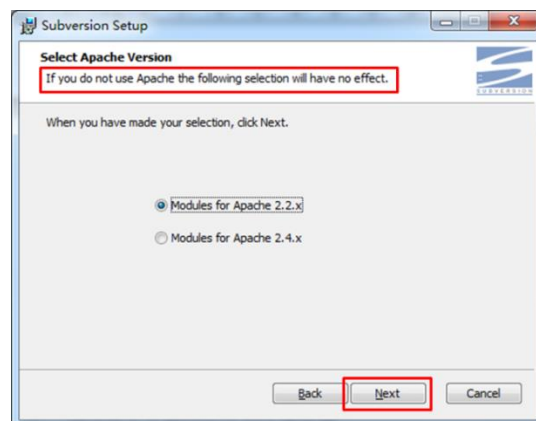
下载到的文件是: Setup-Subversion-1.8.9-1.msi

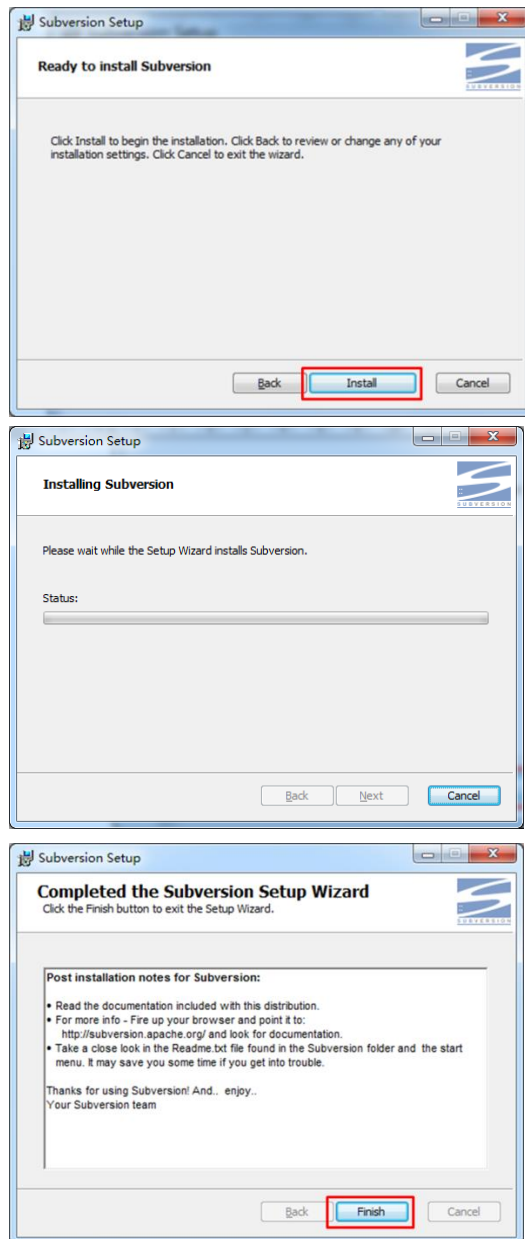
##### ④双击运行 Setup-Subversion-1.8.9-1.msi





不整合 Apache 服务器可以忽略此选项





⑤安装程序会自动配置 Path 环境变量

D:\DevInstall\Subversion\bin

所以 bin 目录下的可执行文件可以在任意目录下运行。

⑥验证是否安装成功

在命令行输入：svn --version

看到如下信息就表示服务器端程序安装成功

```
C:\Users\Phenix>svn --version
svn, 版本 1.8.9 (r1591380)
编译于 May 8 2014, 13:53:01 在 x86-microsoft-windows

Copyright (C) 2014 The Apache Software Foundation.
This software consists of contributions made by many people;
see the NOTICE file for more information.
Subversion is open source software, see http://subversion.apache.org/

可使用以下的版本库访问模块:

* ra_svn : 使用 svn 网络协议访问版本库的模块。 - 使用 Cyrus SASL 认证
  - 处理 "svn" 方案
* ra_local : 访问本地磁盘的版本库模块。
  - 处理 "file" 方案
* ra_serf : Module for accessing a repository via WebDAV protocol using serf.
  - using serf 1.3.5
  - 处理 "http" 方案
  - 处理 "https" 方案
```

### 3.2 配置版本库

#### ①为什么要配置版本库？

Subversion 是将文件数据信息保存到版本库中进行管理的，为了满足用户的不同需求，Subversion 允许用户对版本库目录进行定制。

#### ②在一个非中文无空格目录下创建一个文件夹，作为版本库的根目录。

例如：D:\DevRepository\Subversion

③在版本库根目录下创建与具体项目对应的子目录——这样做的目的是使一个 SVN 服务器能够同时管理多个项目，而不是为每一个项目搭建一个 SVN 服务器——这显然太浪费资源了。

例如：D:\DevRepository\Subversion\CRM

D:\DevRepository\Subversion\ERP

D:\DevRepository\Subversion\OA

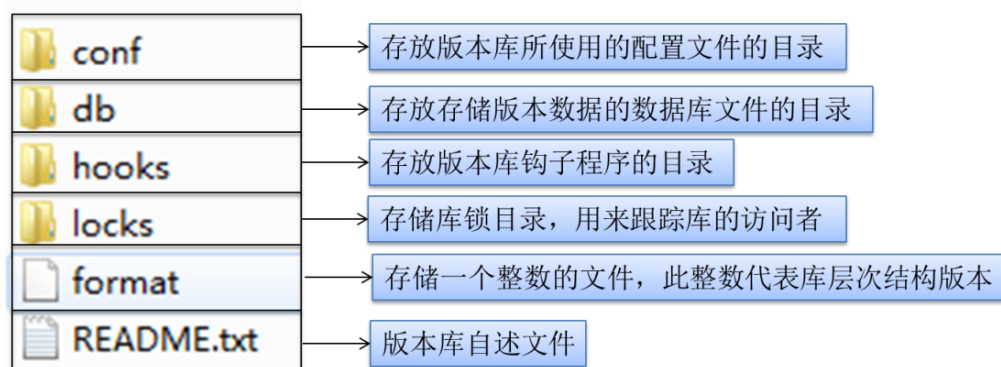
#### ④创建版本库

命令格式

主命令	子命令	参数 1
svnadmin	create	仓库路径
举例	svnadmin create D:\DevRepository\Subversion\StuSys	

#### ⑤版本库目录结构

版本库创建成功后会在指定目录下产生如下的目录结构



### 3.3 启动服务器端程序

①SVN 服务器必须处于运行状态才能响应客户端请求，帮助我们管理项目文件。所以我们将 SVN 服务器启动起来。启动 SVN 服务器有两种方法，一个是命令行方式，一个是注册 Windows 服务。



## ②命令行方式

### [1]命令格式

主命令	参数 1	参数 2	参数 3
svnserve	-d 表示后台执行	-r 表示版本库根目录	D:\DevRepository\Subversion
举例	svnserve -d -r D:\DevRepository\Subversion		

### [2]验证服务是否启动

SVN 服务监听 3690 端口, 打开一个新的 cmd 窗口, 使用 netstat -an 命令查看 3690 端口是否被监听

```
C:\Users\Phenix>netstat -an

活动连接

 协议 本地地址           外部地址           状态
TCP    0.0.0.0:135         0.0.0.0:0          LISTENING
TCP    0.0.0.0:445         0.0.0.0:0          LISTENING
TCP    0.0.0.0:623         0.0.0.0:0          LISTENING
TCP    0.0.0.0:1158        0.0.0.0:0          LISTENING
TCP    0.0.0.0:2401        0.0.0.0:0          LISTENING
TCP    0.0.0.0:3306        0.0.0.0:0          LISTENING
TCP    0.0.0.0:3690        0.0.0.0:0          LISTENING
TCP    0.0.0.0:5520        0.0.0.0:0          LISTENING
TCP    0.0.0.0:10108       0.0.0.0:0          LISTENING
```

[3]命令行方式的缺陷是：只要运行服务器端程序的命令行窗口一关闭，服务就停止了，很不方便，而且每次开机都需要手动启动。

### ③注册 Windows 服务

[1]将 SVN 服务端程序注册为 Windows 服务, 就可以让 SVN 服务随系统一起启动, 克服了命令行方式的不足。

[2]注册 Windows 服务需要利用 XP、2000 以上系统自带工具 Service Control, 执行文件是 sc.exe, 注意这个命令不是 SVN 的命令。

### [3]命令格式

主命令	子命令	参数 1	参数 2	参数 3	参数 4
sc	create	服务名	binpath= “运行服务所需要的二进制文件路径以及运行该二进制文件的命令行参数”	start= auto 表示自动启动	depend= Tcpip 表示依赖 Tcpip 协议

**[注意：在这个命令中，等号左边都没有空格，右边都有一个空格！]**

### [4]binpath 组成结构说明

svnserve.exe 路径	svnserve 命令参数 1	svnserve 命令参数 2	svnserve 命令参数 3
SVN 安装目录\bin\svnserve.exe	--service 表示以服务方式启动 Subversion	-r 表示版本库根目录	版本库目录

### [5]关于“版本库目录”

单仓库	指定与具体项目对应的仓库目录	例如：D:\DevRepository\Subversion\CRM	只能为一个项目服务
多仓库	指定版本库的根目录	例如：D:\DevRepository\Subversion	可以为多个项目服务

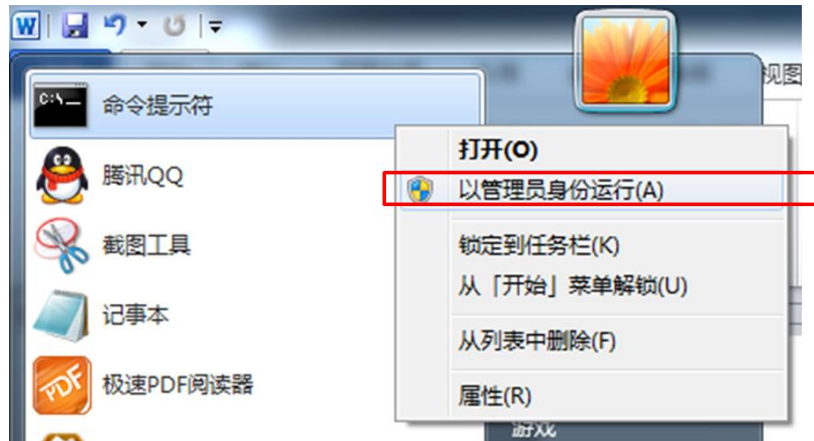
### [6]最终命令举例

```
sc create MySVNService binpath= "D:\DevInstall\Subversion\bin\svnserve.exe --service -r D:\DevRepository\Subversion"  
start= auto depend= Tcpip
```

- 在 Win7 及以上系统中，运行该命令需要管理员权限，否则会得到如下错误提示

[SC] OpenSCManager 失败 5:  
拒绝访问。

- 解决的办法是以管理员身份运行 cmd 命令行窗口即可



- 在防火墙或电脑卫士提示阻止时，选择允许



- 此时查看当前系统中的服务，可以看到我们刚刚创建的服务，但此时它还没有启动，如果创建失败，需检查 sc 命令是否正确

MySQL	已启动	自动	本地系统
MySVNService		自动	本地系统
Net.Msmq Listen...	通过 ...	禁用	网络服务

- 启动此服务



启动服务的命令格式如下：

格式	sc start 服务名
举例	sc start MySVNService

MySQL	已启动	自动	本地系统
MySVNService	已启动	自动	本地系统
Net.Msmq Listen...	通过 ...	禁用	网络服务

- 打开命令行窗口运行 `netstat -an` 查看 3690 端口是否被监听
- 如果启动失败，那很有可能是 `binpath` 中的内容有错误，此时只能将已经创建的服务删除，重新创建。
- 删除服务之前，最好先停止服务。停止服务的命令格式如下：

格式	sc stop 服务名
举例	sc stop MySVNService

- 删除服务的命令格式如下：

格式	sc delete 服务名
举例	sc delete MySVNService

- 删除、启动、停止服务同样需要管理员权限

#### 4 使用命令行模式访问 SVN 服务器

##### 4.1 检出

- ① 首先进入自己的工作目录，例如：D:\DevWorkSpace\SVNSpace
- ② 运行 `svn checkout` 命令，命令格式如下

格式	svn checkout svn://SVN 服务器主机地址/具体仓库目录 保存检出内容的目录
举例	svn checkout svn://localhost/ERP MyERP
运行结果	取出版本 0。

##### ③ 工作副本

运行 `checkout` 命令后进入 `MyERP` 目录，看到里面什么都没有。真的什么都没有吗？不是的。检出命令会在这一目录下创建一个隐藏目录 `.svn`，用来保存与服务器交互的重要信息，其中包括从服务器端取回的最新版本信息、文件状态、更新时间等。`SVN` 正是以此为依据判断当前目录中文件的状态。所以这个隐藏目录 **千万不要删除或修改其中的内容**——完全无视它的存在吧。如果服务器端保存的文件可以视为一个“正本”，那么每个开发人员检出到本地目录的文件可以视为“副本”，通常称为工作副本。

##### 4.2 提交

- ① 进入 D:\DevWorkSpace\SVNSpace\MyERP 目录
- ② 创建一个文件 `test.txt`
- ③ 执行 `svn commit` 命令，运行结果是

```
D:\DevWorkSpace\SVNSpace\MyERP>svn commit test.txt
svn: E200009: 提交失败(细节如下):
svn: E200009: “D:\DevWorkSpace\SVNSpace\MyERP\test.txt” 尚未纳入版本控制
```

说明一个文件必须纳入版本控制才可以提交到服务器端。

- ④ 执行 `svn add` 命令，将 `test.txt` 纳入版本控制

```
D:\DevWorkSpace\SVNSpace\MyERP>svn add test.txt
A      test.txt
```

⑤再次执行 `svn commit` 命令

```
D:\DevWorkSpace\SVNSpace\MyERP>svn commit test.txt
svn: E205007: 提交失败(细节如下):
svn: E205007: 无法使用外部编辑器获得日志信息; 考虑设置环境变量 $SVN_EDITOR, 或者
使用 --message (-m) 或 --file (-F) 选项
svn: E205007: 没有设置 SVN_EDITOR, VISUAL 或 EDITOR 环境变量, 运行时的配置参数中
也没有 “editor-cmd” 选项
```

此时要求附加日志信息

⑥使用 `-m` 参数附加日志信息

```
D:\DevWorkSpace\SVNSpace\MyERP>svn commit -m "My first commit" test.txt
svn: E170001: 提交失败(细节如下):
svn: E170001: 认证失败
```

原因是没有权限

⑦暂时先开启匿名访问权限

[1]进入对应的版本库目录下的 `conf` 目录: `D:\DevRepository\Subversion\ERP\conf`

[2]打开 `svnserve.conf`

[3]将第 19 行的 `# anon-access = read` 改为 `anon-access = write`, 也就是去掉 “#”, 将 `read` 改为 `write`。注意前面不要留空格, 一定要顶格写。

[4]不需要重启 SVN 服务, 甚至命令行窗口都不需要重新打开。

⑧重新执行提交命令

```
D:\DevWorkSpace\SVNSpace\MyERP>svn commit -m "My first commit" test.txt
正在增加      test.txt
传输文件数据.
提交后的版本为 1。
```

说明提交成功了。

⑨其实 `svn commit` 命令最后可以不指定具体文件, 此时表示提交当前工作副本中的所有修改。

#### 4.3 更新

①将服务器端文件检出到一个新的目录, 模拟另外一个终端

```
D:\DevWorkSpace\SVNSpace>svn checkout svn://localhost/ERP TomERP
A      TomERP\test.txt
取出版本 1。
```

②回到 `MyERP` 目录, 对 `test.txt` 文件修改后提交。

③进入 `TomERP` 目录

④执行 `svn update` 命令

```
D:\DevWorkSpace\SVNSpace\TomERP>svn update
正在升级 '!':
U      test.txt
更新到版本 2。
```

这样我们就可以在 TomERP 目录下看到 MyERP 目录下提交的修改。

⑤思考：更新和检出的相同点和不同点分别是什么？

	检出	更新
相同点	从服务器端下载最新内容	
不同点 1	下载整个项目	下载与本地工作副本不同的内容
不同点 2	创建.svn 目录,使检出目录成为工作副本	依赖.svn 目录
不同点 3	只能操作 1 次	可以操作多次

#### 4.4 工作副本中文件的几种状态

①没有修改, 现行版本

本档案在工作目录中没有被修改, 而且自当前版本之后, 其他终端也没有任何该文件的修改被提交到服务器, 即当前工作副本的版本和服务器端最新版本是一致的。对它执行 `svn commit` 和 `svn update` 都不会发生任何事。

②本地修改, 现行版本

这个文件被修改过, 但这个修改还没有提交到服务器, 而且自当前版本之后, 其他终端也没有任何该文件的修改被提交到服务器, 所以当前工作副本的版本和服务器端最新版本仍然是一致的。由于有尚未送交回去的本地修改, 所以对它的 `svn commit` 会成功提交你的修改, 而 `svn update` 则不会作任何事。

③没有修改, 过时版本

这个文件没有修改, 但是版本库中有其他终端提交的修改。此时当前工作副本的版本比服务器端的版本落后了, 我们称之为“过时”。对当前文件的 `svn commit` 不会发生任何事, 而 `svn update` 会让工作目录中的文件更新至最新版本。

④本地修改, 过时版本

服务器端存在没有更新到本地的修改, 导致当前版本过时。如果这个文件在本地有未提交的修改, 则无法提交, 对它执行 `svn commit` 会产生“out-of-date”错误。

此时应该先尝试更新本地文件。更新时 `SVN` 会尝试将服务器端的更新与本地文件进行合并, 合并的结果有两种可能: 一个是服务器端和本地修改位于文件的不同位置, 合并成功; 另一个是服务器端的修改正好和本地修改位于同一个位置, 发生冲突。

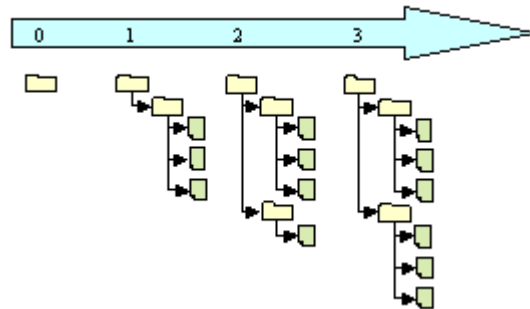
#### 4.5 将工作副本整体回复到某一个历史版本

①假设当前版本为 12, 想要取回版本 9

②执行 `svn update` 命令

格式	<code>svn update --revision 想要取回的版本号</code>
举例	<code>svn update --revision 1</code>
运行结果	正在升级 '': D pp.txt U test.txt 更新到版本 1。

③这里需要注意的是, `SVN` 版本号并不是对某一个文件进行编号, 而是对应整个版本库总体状态的一个“快照”, 取回某个版本不是取回版本号对应的某个文件, 而是整个项目的一个快照。



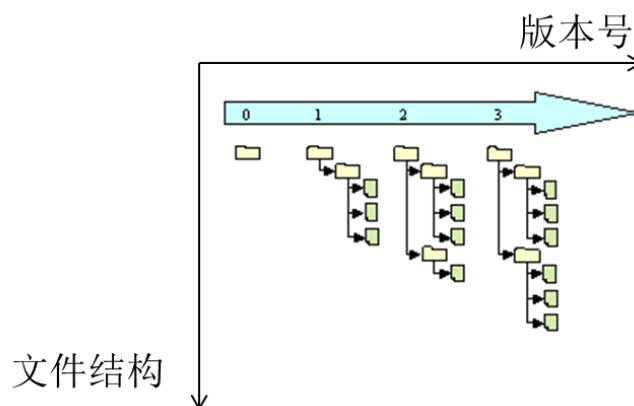
4.6 将某个文件恢复到某个版本中的状态，同时不涉及其他文件

①假设想要取回 pp.txt 在版本 10 时的状态

②执行 svn update 命令

格式	svn update 文件名 --revision 想要取回的版本号
举例	svn update pp.txt --revision 10
运行结果	正在升级 'pp.txt': U pp.txt 更新到版本 10。

③综合这两个例子，我们可以认为版本号和文件名构成了一个横纵坐标系，通过文件路径和版本号定位其在某一个时刻的状态。



## 5 单一版本库权限配置

5.1 匿名访问：前已述及。

5.2 授权访问

①要设置授权访问就需要创建用户，并为用户设定权限

②打开授权访问的配置

[1]打开 D:\DevRepository\Subversion\ERP\conf\svnserve.conf

[2]将第 19 行 anon-access = write 注释掉：# anon-access = write

表明该版本库不接受匿名访问

[3]将第 20 行# auth-access = write 注释打开：auth-access = write

表明该版本库使用授权访问

[4]将第 27 行注释打开：password-db = passwd

表明使用同目录下的 passwd 文件保存用户信息

[5]将第 36 行注释打开：authz-db = authz  
表明使用同目录下的 authz 文件保存权限信息

[6]打开 passwd 文件创建用户

```
userWrite01 = 123456
userWrite02 = 123456
userRead = 123456
userOther = 123456
```

[7]打开 authz 文件：#后面注释的是例子

<1>创建用户组

```
[groups]
# harry_and_sally = harry,sally
# harry_sally_and_joe = harry,sally,&joe
canWrite = userWrite01,userWrite02
```

<2>指定路径，给用户和用户组授权

```
# [/foo/bar]
# harry = rw
# &joe = r
# * =屏蔽那些未设定的用户，让它们没有任何权限
[/]
@canWrite = rw
userRead = r
* =
```

<3>权限的继承性：父目录设置的权限，对子目录同样有效——除非子目录进行了更为具体的设定

```
[/subDir]
userOther = rw
* =
```

这个例子表示当前版本库下的 subDir 目录只有 userOther 有读写权限，其它用户无任何权限

## 6 多版本库共享配置

6.1 在版本库根目录 D:\DevRepository\Subversion 下创建 commConf 目录

6.2 将未修改的 authz 和 passwd 文件拷贝到 commConf 目录下

6.3 修改需要设置权限的版本库的 svnserve.conf 文件

①password-db = ../../commConf/passwd

②authz-db = ../../commConf/authz

6.4 在 password 中创建用户

```
[users]
# harry = harryssecret
# sally = sallyssecret

userERP = 123456
```

```
userOA = 123456
userCRM = 123456
```

#### 6.5 在 authz 中针对不同版本库为不同用户授予权限

```
# [repository:/baz/fuz]
# @harry_and_sally = rw
# * = r

[ERP:/]
userERP = rw
* =

[OA:/]
userOA = rw
* =

[CRM:/]
userCRM = rw
* =
```

#### 7 查看工作副本信息

①使用 `svn info` 命令

②执行效果如下

```
D:\DevWorkSpace\SVNSpace\MyCRM>svn info
路径: .
工作副本根目录: D:\DevWorkSpace\SVNSpace\MyCRM
URL: svn://localhost/CRM
正确的相对 URL: ^/
版本库根: svn://localhost/CRM
版本库 UUID: d5768329-0587-f54a-b44e-72a9a02ddf1b
版本: 12
节点种类: 目录
调度: 正常
最后修改的作者: userCRM
最后修改的版本: 12
最后修改的时间: 2014-08-08 00:30:38 +0800 (周五, 2014-08-08)
```

③对某一个文件使用 `svn info` 命令

```
D:\DevWorkSpace\SVNSpace\MyCRM\src\com\atguigu\crm>svn info MyCRM.java
路径: MyCRM.java
名称: MyCRM.java
工作副本根目录: D:\DevWorkSpace\SVNSpace\MyCRM
URL: svn://localhost/CRM/src/com/atguigu/crm/MyCRM.java
正确的相对 URL: ^/src/com/atguigu/crm/MyCRM.java
版本库根: svn://localhost/CRM
```



```
版本库 UUID: d5768329-0587-f54a-b44e-72a9a02ddf1b
版本: 14
节点种类: 文件
调度: 正常
最后修改的作者: userCRM
最后修改的版本: 14
最后修改的时间: 2014-08-08 00:36:06 +0800 (周五, 2014-08-08)
文本最后更新: 2014-08-08 00:35:59 +0800 (周五, 2014-08-08)
校验和: dbea85a617fb59cb739c3d18968a941d32858372
```

## 8 查看目录或文件日志信息

### ①使用 svn log 命令

[注意: 使用这个命令的前提是设置匿名访问为 none, 即: anon-access = none, 否则会出现 “**svn: E220001: 条目不可读**” 错误]

### ②执行效果如下

```
D:\DevWorkSpace\SVNSpace\MyCRM\src\com\atguigu\crm>svn log MyCRM.java
-----
r18 | userCRM | 2014-08-08 09:43:00 +0800 (周五, 2014-08-08) | 4 行

aaa
bbb
ccc
ddd
-----
r17 | userCRM | 2014-08-08 09:36:46 +0800 (周五, 2014-08-08) | 1 行

aaaa\nbbb\ntt
-----
r16 | userCRM | 2014-08-08 09:30:26 +0800 (周五, 2014-08-08) | 1 行

-----
r15 | userCRM | 2014-08-08 00:58:30 +0800 (周五, 2014-08-08) | 1 行

-----
r14 | userCRM | 2014-08-08 00:36:06 +0800 (周五, 2014-08-08) | 1 行

tt
-----
r13 | userCRM | 2014-08-08 00:33:12 +0800 (周五, 2014-08-08) | 1 行

tt
```

```
-----  
r4 | userCRM | 2014-08-07 22:19:16 +0800 (周四, 2014-08-07) | 1 行  
  
-----  
r2 | (没有作者信息) | 2014-08-05 11:37:29 +0800 (周二, 2014-08-05) | 1 行  
  
tt  
  
-----  
r1 | (没有作者信息) | 2014-08-05 11:34:40 +0800 (周二, 2014-08-05) | 1 行  
  
tt  
  
-----
```

## 9 在 Eclipse 中安装 SVN 客户端插件

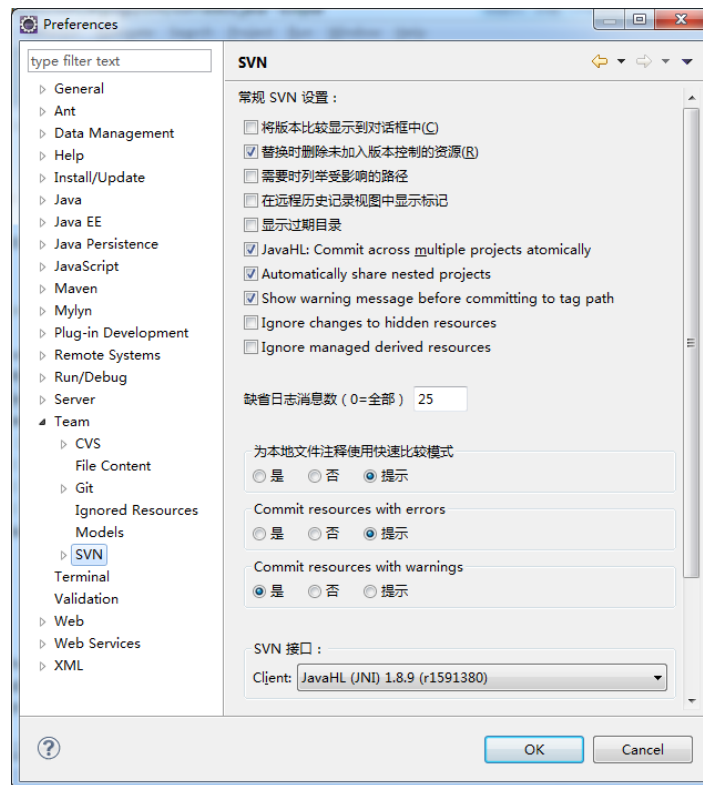
### 9.1 Eclipse 插件应用市场

在 Eclipse 中访问 Eclipse Marketplace Client 可以搜索 Subversion，下载插件，按提示安装即可。

### 9.2 使用压缩包

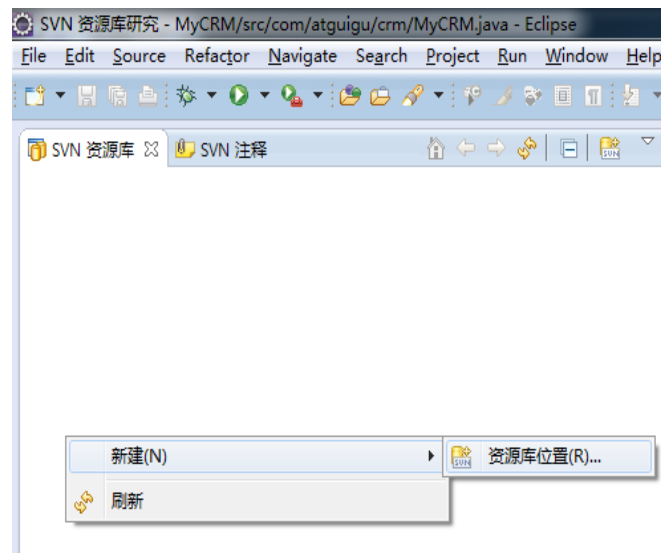
如果不能联网可以使用下载好的插件压缩包 eclipse\_svn\_site-1.10.5.zip，这个压缩包是从 <http://subclipse.tigris.org/> 网站（subclipse 是这款 Eclipse 插件的名称）上下载的。安装方法是：

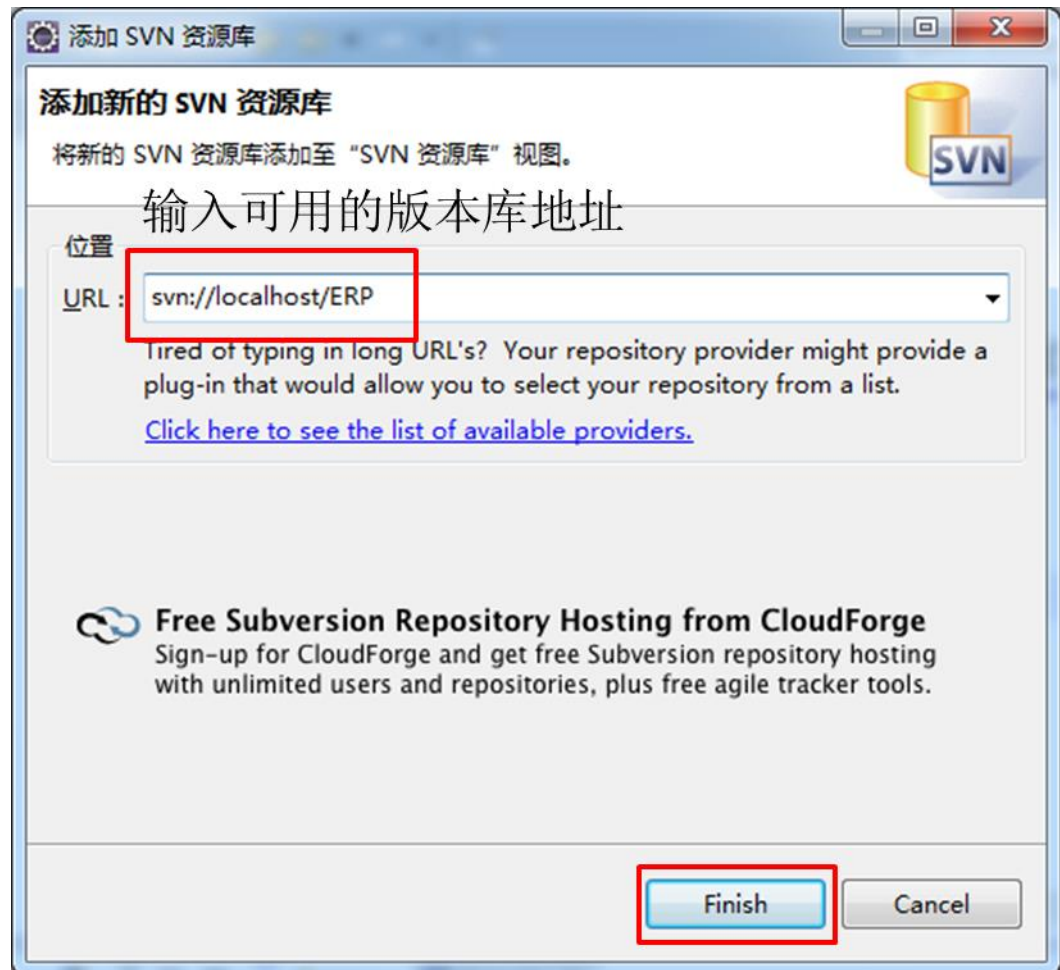
- ①解压 eclipse\_svn\_site-1.10.5.zip 中的 features 和 plugins 这两个目录
- ②将 features 和 plugins 复制到 Eclipse 安装目录/dropins/eclipse\_svn\_site-1.10.5 下
- ③重启 Eclipse
- ④进入 Eclipse 依次打开 Window→Preferences→Team→SVN，看到如下界面即说明 SVN 插件安装成功



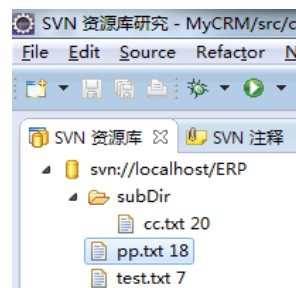
### 9.3 创建资源库位置

- ①切换到透视图 SVN 资源库研究
- ②创建资源库位置

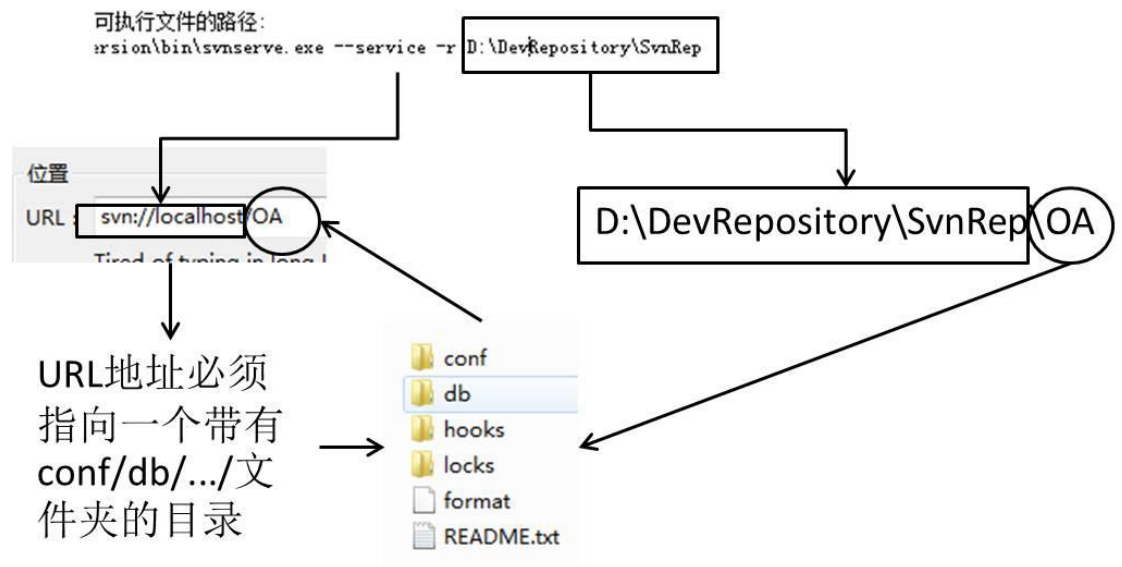




此时可以查看版本库中的文件及目录结构



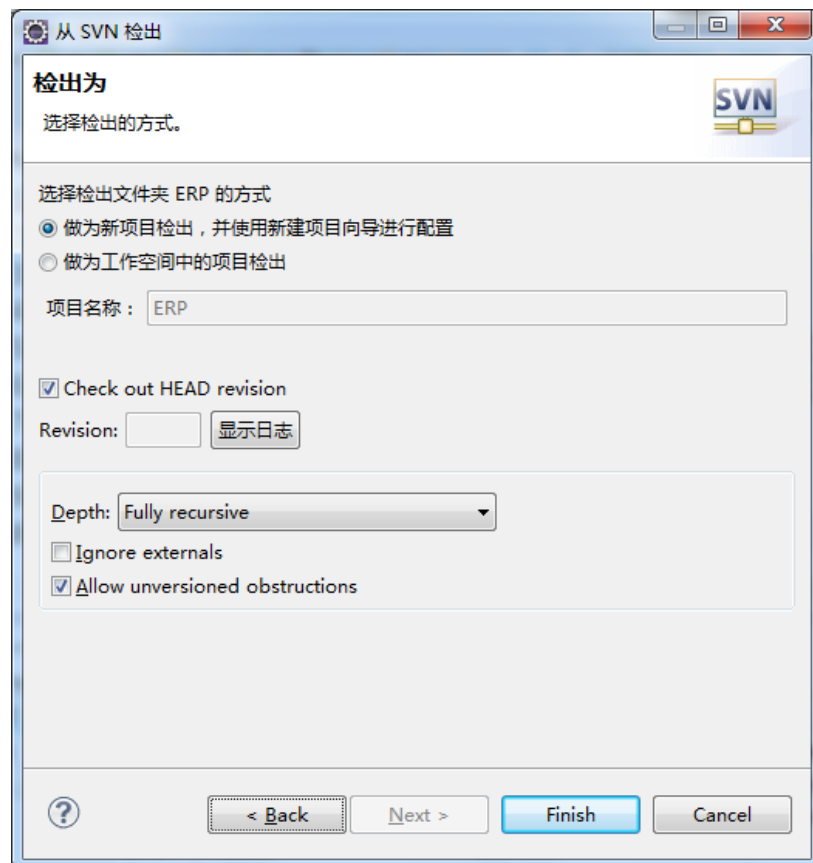
补充：如何确定版本库地址？



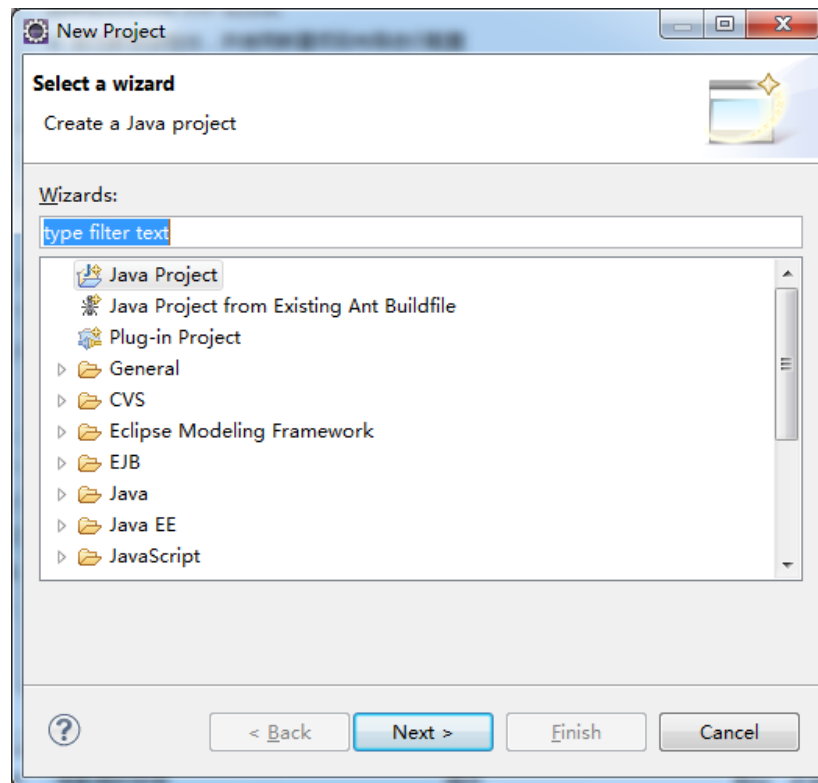
#### 9.4 检出

##### ①检出分两种情况

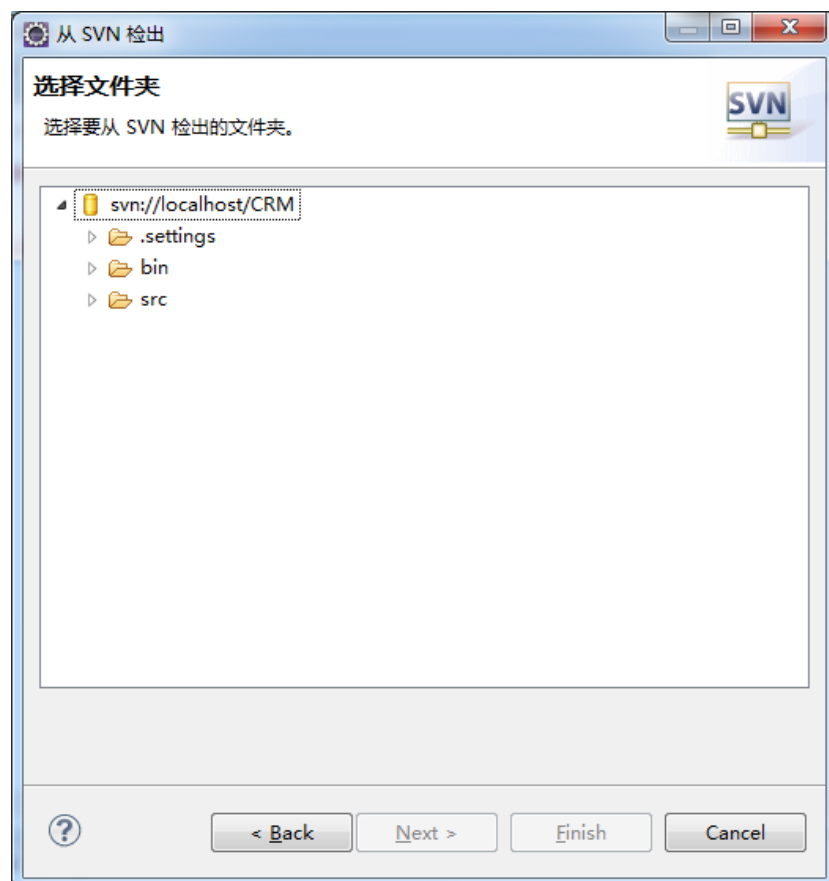
<1>从服务器端获取到的一些零散的文件，不能作为项目检出



此时会弹出一个新建项目向导，之所以会这样是因为我们需要创建一个项目来保存从服务器端取回的文件

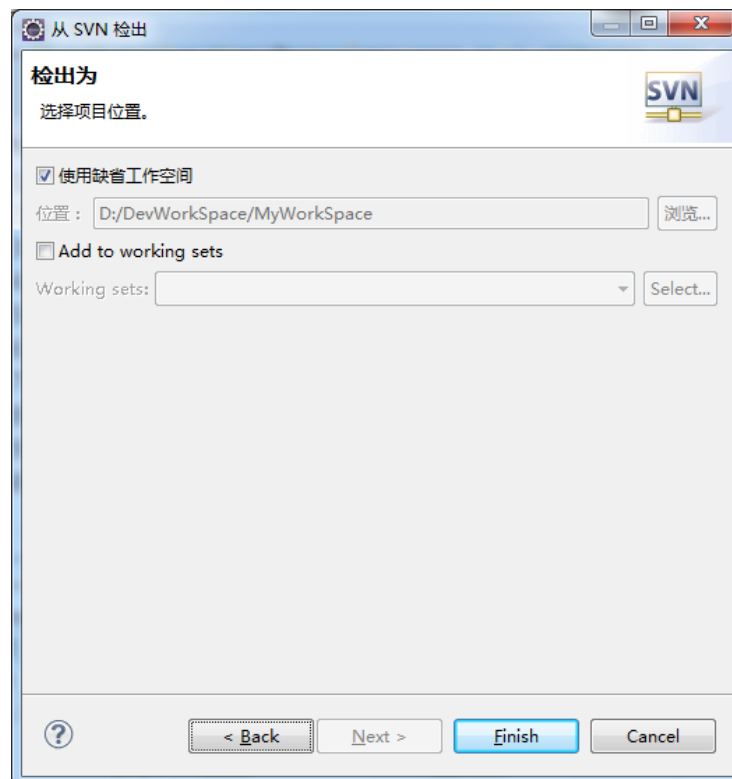
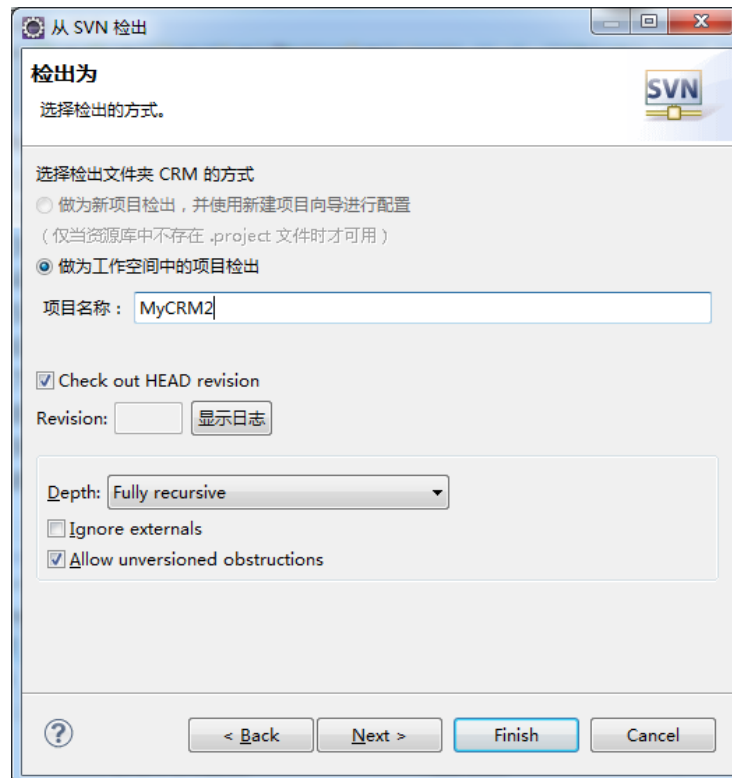


<2>从服务器端获取到的是一个完整的项目，例如





此时必须作为项目检出



项目从服务器检出后，会成为一个工作副本，根目录下会自动创建.svn 隐藏目录  
提交

①新创建文件后，文件图标上会以“?”标识，表示该文件尚未纳入版本控制

②在新创建的文件上点右键→Team→添加至版本控制，这样文件图标上会显示“+”，表示当前文件已纳入版本控制，但还未提交至服务器。

③在要提交的文件上点右键→Team→提交...会提交文件，在弹出的对话框中可以不填写日志。文件提交后，图标会变为“金色的圆柱体”表示当前文件的版本和服务端一致。

④文件修改后图标会变为“\*”，表示当前文件或目录包含未提交的修改。

#### 9.6 更新

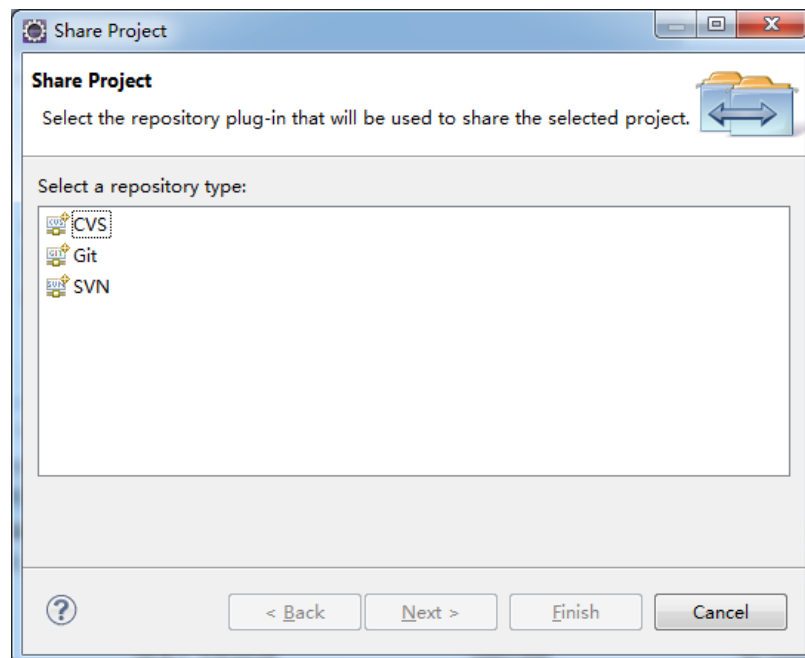
①更新整个项目时可以在项目上点右键→Team→更新

②更新某个具体的文件时，可以在文件上点右键→Team→更新

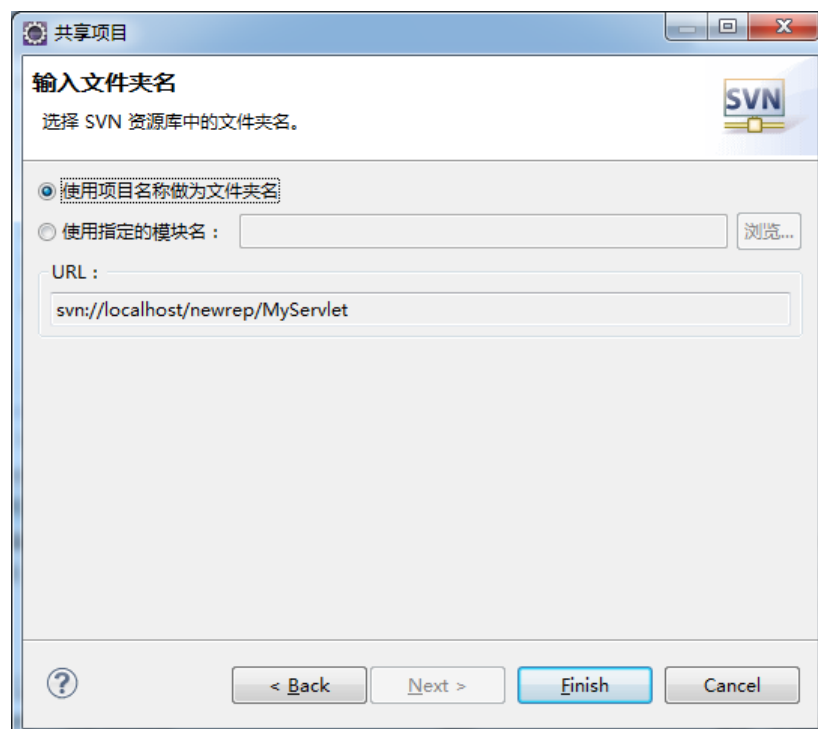
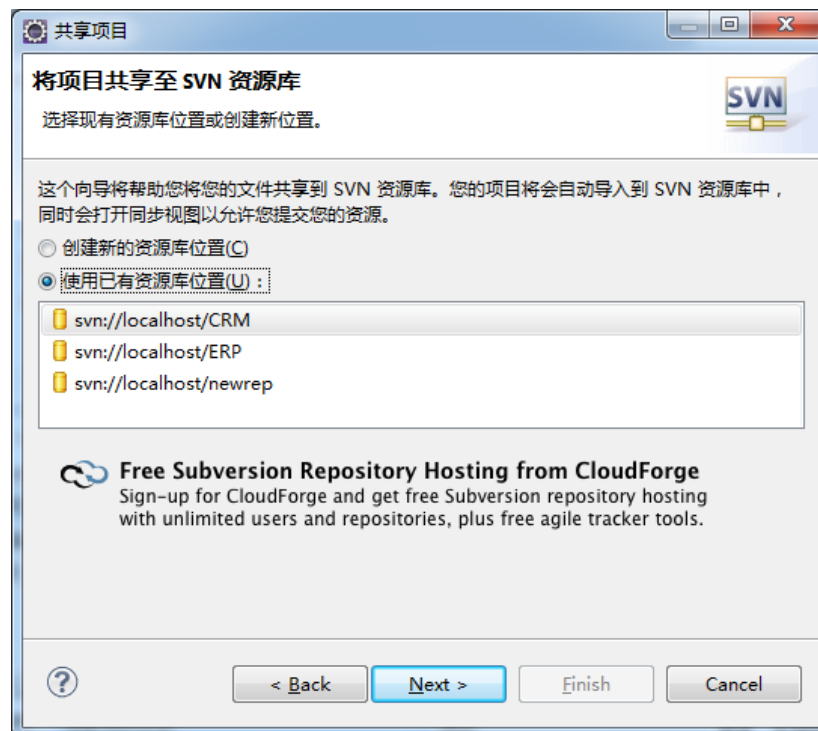
#### 9.7 共享项目

①在 Eclipse 中创建的新项目想要发布到 SVN 服务器端，可以通过“共享”项目实现

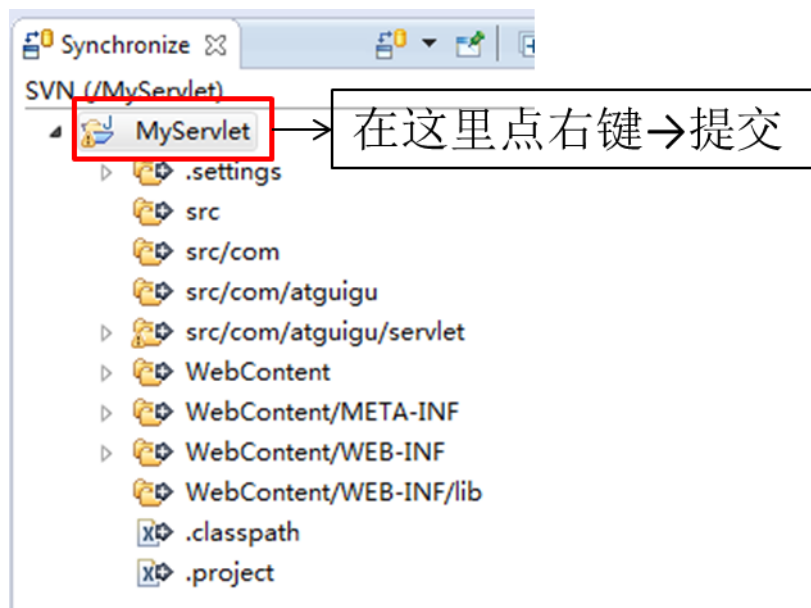
②在项目上点右键→Team→Share Project...→选择一种版本控制工具



选择一个资源库位置



切换到 Team Synchronizing 透视图，选择项目中要提交的内容，通常是项目中的全部内容



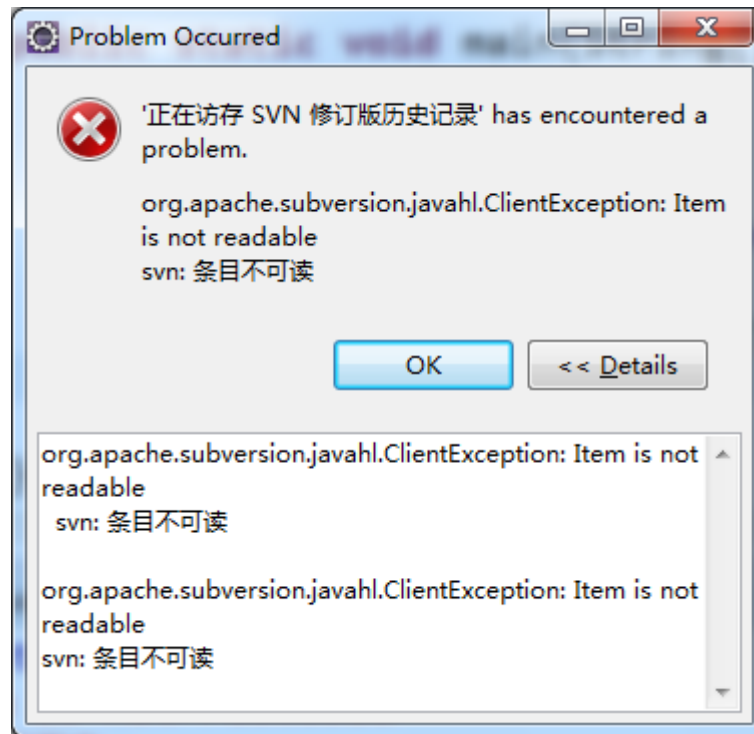
#### 9.8 回复历史版本

①在需要回复的文件上点右键→Team→显示资源历史记录→得到如下界面

/src/com/atguigu/crm/SvnTest01.java in svn://localhost/CRM			
修订	日期	作者	注释
*11	14-8-7 下午10:43	userCRM	by command
10	14-8-7 下午10:42	userCRM	by command
9	14-8-7 下午10:37	userCRM	by command
8	14-8-7 下午10:36	userCRM	
7	14-8-7 下午10:28	userCRM	by command
6	14-8-7 下午10:23	userCRM	
5	14-8-7 下午10:21	userCRM	
动作	受影响的目录	描述	
M	/src/com/atguigu/crm/SvnTest01.java		by command

②选择某一个历史记录点右键→获取内容。文件就会恢复到指定版本的状态，同时图标变为“\*”。

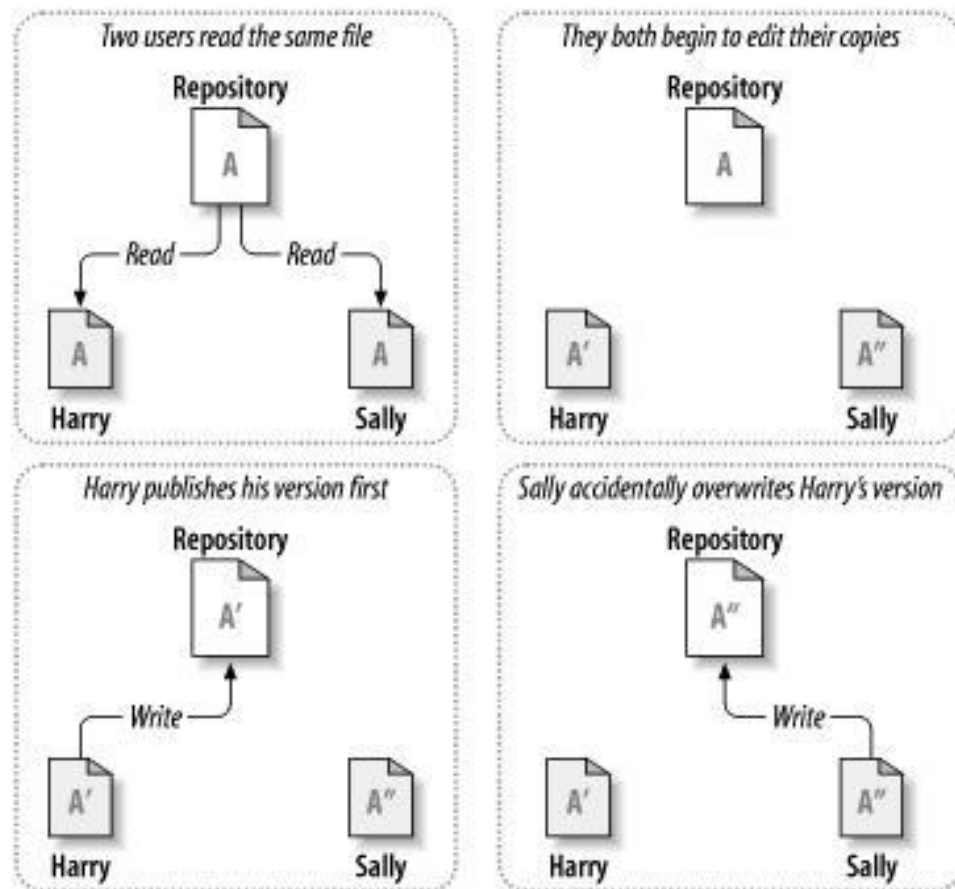
③获取历史记录时，如果出现如下错误提示



可以通过将对应版本库中的 `svnserve.conf` 文件中的 `anon-access` 设置为 `none` 解决。

## 9.9 解决冲突

### ① 什么情况下会发生冲突



<1>两个开发人员，Harry 和 Sally，分别从服务器端下载了文件 A。

<2>Harry 修改之后，A 变成了 A'，Sally 修改之后，A 变成了 A''。

<3>Harry 先一步提交，使服务器端文件的版本也变成了 A'

<4>Sally 本地的文件 A'' 已经过时了，此时她已无法提交文件，服务器会要求她先进行一次更新操作。

<5>此时 Sally 的更新操作有两种可能

(1) Sally 所做的修改与 Harry 不是同一个位置，更新操作尝试合并文件成功。

(2) Sally 所做的修改与 Harry 恰好是同一个位置，更新操作尝试合并文件失败，发生冲突。

<6>发生冲突后，本地工作副本会发生如下变化

(1) 文件 A 中的内容发生如下改变

```
public static void main(String[] args) {  
    System.out.println("Edit By Command!");  
    System.out.println("Edit By Command!");  
    <<<<<<< .mine  
        System.out.println("Edit By Eclipse!");  
    =====  
        System.out.println("Edit By Command!New Edit");  
    >>>>>>> .r14
```



```
System.out.println("Edit By Command!");  
System.out.println("Edit By Command!");  
}
```

其中，从<<<<<<< .mine 到=====之间是发生冲突时本地副本的内容。从=====到>>>>>>> .r14 是发生冲突时服务器端的最新内容。注意这里 r 后面的数字是发生冲突时服务器端的版本号，有可能是任何整数值，r14 只是一个例子。

同时文件图标变成一个“黄色的！”。

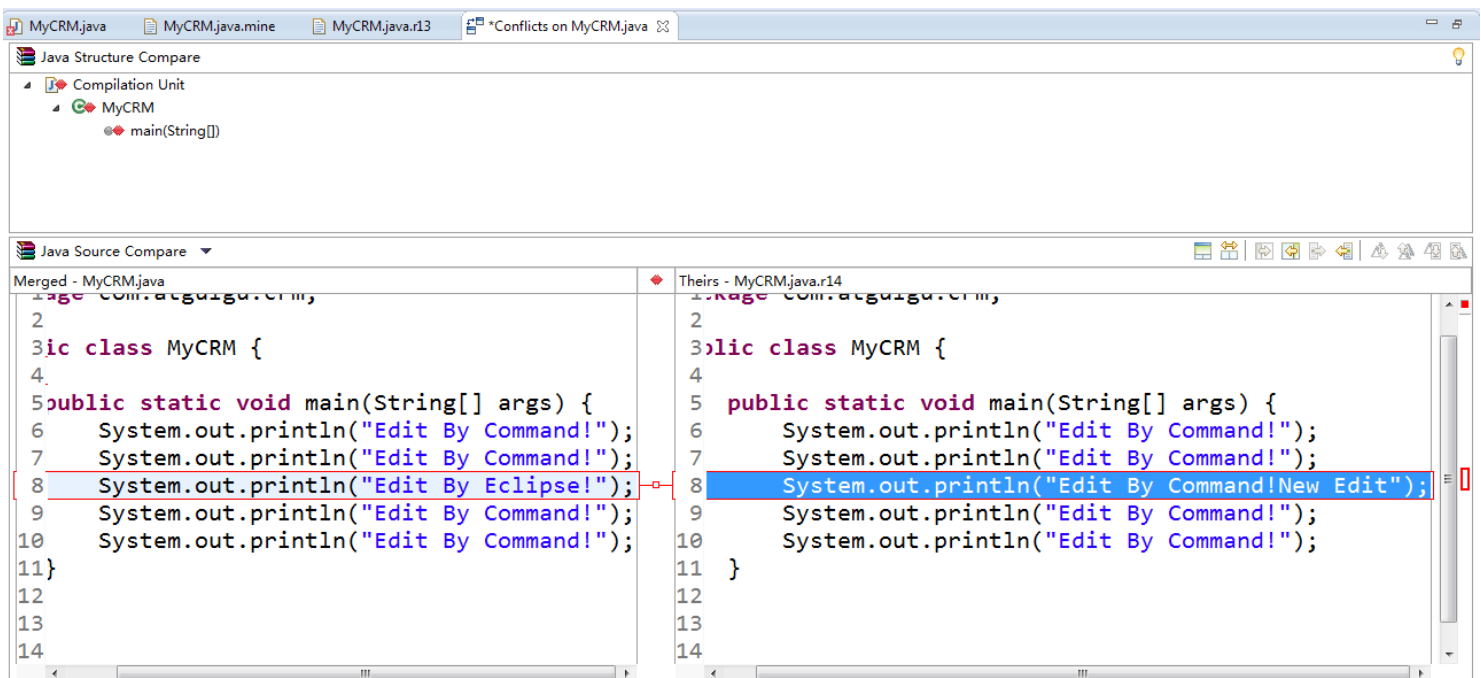
(2)与冲突文件同目录下新增文件，扩展名为.mine，其内容是发生冲突时本地副本的文件内容。

(3)与冲突文件同目录下新增文件，扩展名为.r 小版本号，例如 MyCRM.java.r13，其内容是冲突发生之前，服务器端的文件内容，可以作为解决冲突的参照。

(4)与冲突文件同目录下新增文件，扩展名为.r 大版本号，例如 MyCRM.java.r14，其内容是冲突发生时，服务器端的文件内容。

## ②解决冲突

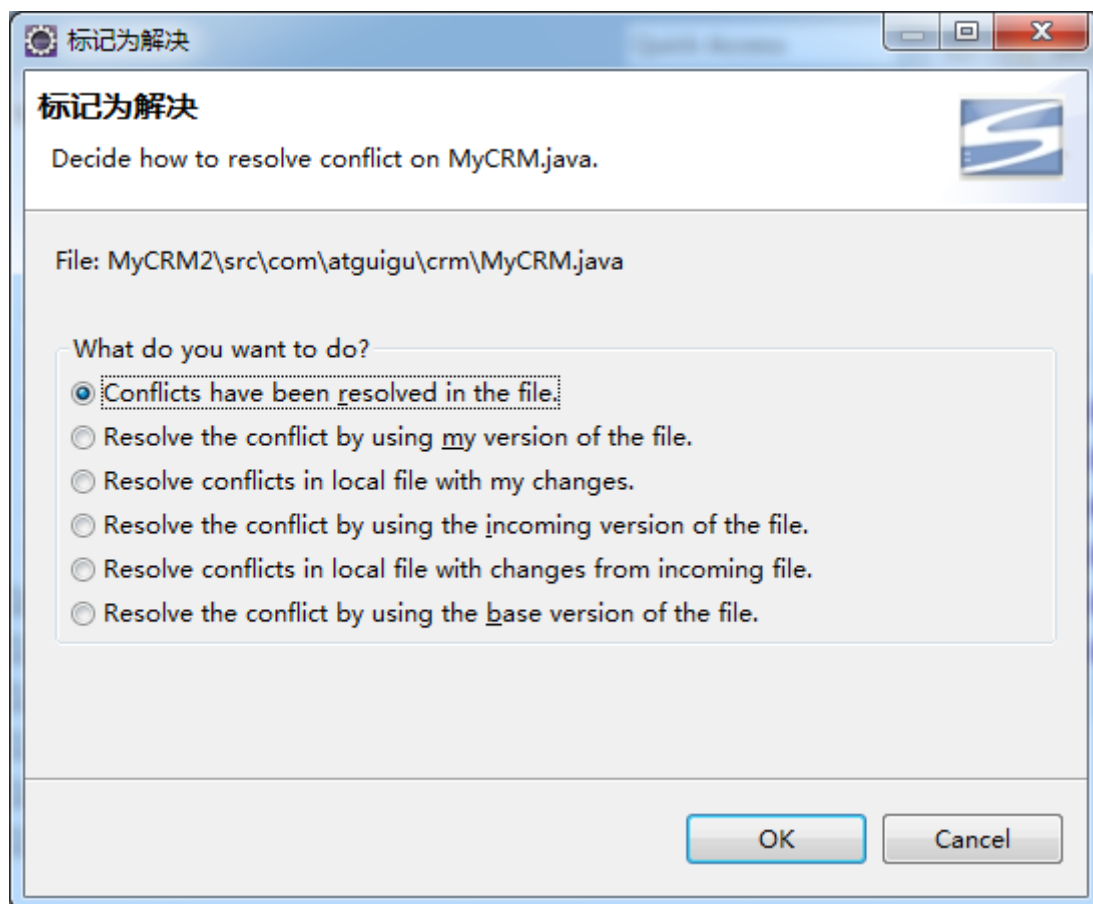
(1)在冲突文件上点右键→Team→编辑冲突...→出现如下界面



以对比的方式将本地内容与冲突内容显示出来，其中左侧为本地内容，右侧为冲突内容。其中本地内容是可以修改的。

(2)根据需求和实际情况将本地内容更正——这个过程很可能需要牵涉冲突的两位开发人员进行必要的沟通——机器与程序目前还不能完全取代人工智能。更正后文件图标会变成一个“四角形”，同时冲突文件内的<<<<<<< .mine、=====以及>>>>>>> .r14 等标记都会被去掉。

(3)在冲突文件上点右键→Team→标记为解决



此时.mine 文件和.r 版本号文件都会被自动删除，冲突文件的图标变为“\*”，表示可以提交。

(4)提交文件，文件图标变为“金色圆柱体”。

## 10 使用 SVN 独立客户端：TortoiseSVN

### 10.1 TortoiseSVN 简介

TortoiseSVN 是一个 Windows 下的版本控制系统 Apache™ Subversion®的客户端工具。



# TortoiseSVN

### 10.2 TortoiseSVN 的优良特性

#### ①外壳集成

TortoiseSVN 无缝地整合进 Windows 的外壳(例如资源管理器)。

#### ②重载图标

每个版本控制的文件和目录的状态使用小的重载图标表示，可以让你立刻看出工作副本的状态。

#### ③图形用户界面

当你列出文件或文件夹的更改时，你可以点击任意版本查看提交注释。也可以看到更改过的文件列表 - 只要双击文件就可以查看更改内容。

提交对话框列出了本次提交将要包括的条目，每一个条目有一个复选框，所以你可以选择包括哪些条目。未版本控制的文件也会被列出，以防你忘记添加新文件。

#### ④Subversion 命令的简便访问

所有的 Subversion 命令存在于资源管理器的右键菜单，TortoiseSVN 在那里添加子菜单。

### 10.3 TortoiseSVN 的历史

2002 年，Tim Kemp 发现 Subversion 是一个非常好的版本管理系统，但是缺乏一个好的图形界面客户端程序。做一个与 Windows 外壳整合的 Subversion 客户端程序的想法是受一个叫 TortoiseCVS 的 CVS 客户端程序所启发的。Tim 研究了 TortoiseCVS 的源码并以此为 TortoiseSVN 的基础。他开始运作这个项目，注册了域名 tortoisefvn.org 并且将源码放在了网上。

就在同时，Stefan K ng 正在寻找一个好用的并且免费的版本控制系统。他找到了 Subversion 和 TortoiseSVN 的源码。因为 TortoiseSVN 还不能使用，他加入了项目并开始编码。很快，他就重写了现有的大部分代码并开始添加命令和功能，到了某个时段，最初的代码已经都被改写了。

由于 Subversion 变得越来越稳定，它吸引了越来越多用户，他们同时也开始使用 TortoiseSVN 作为 Subversion 的客户端程序。用户数量快速增长(并且每天还在增长)。这时候，L bbe Onken 提出帮助项目提供精美的图标和 TortoiseSVN 的标志。现在他负责照看网站和管理多语言翻译。

### 10.4 TortoiseSVN 安装

①下载安装程序：<http://tortoisefvn.net/downloads.html>

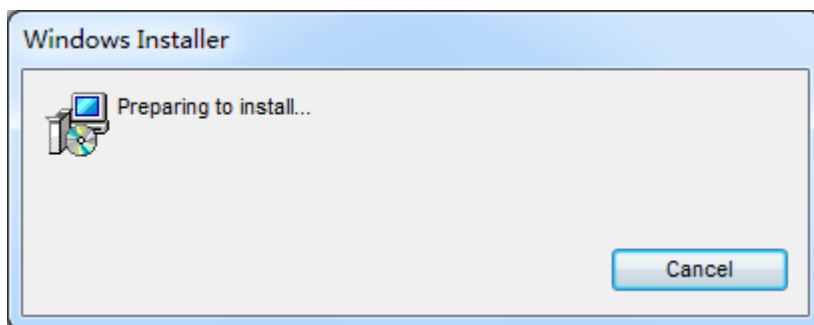
Win32 位：TortoiseSVN-1.8.7.25475-win32-svn-1.8.9.msi

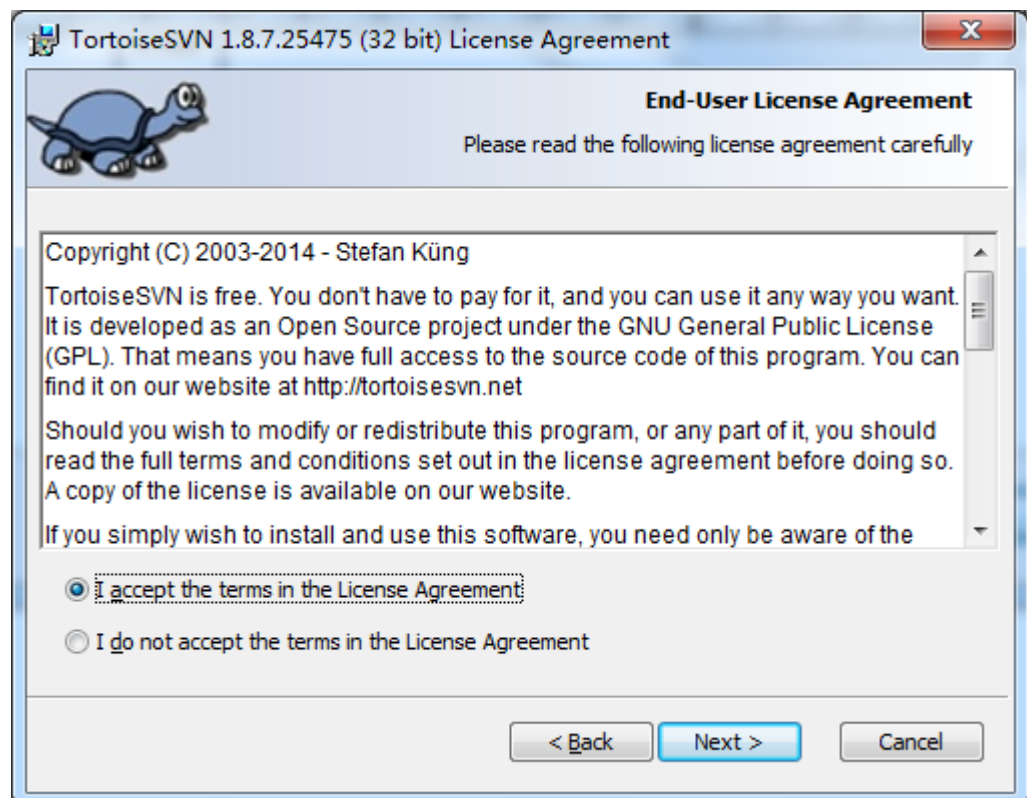
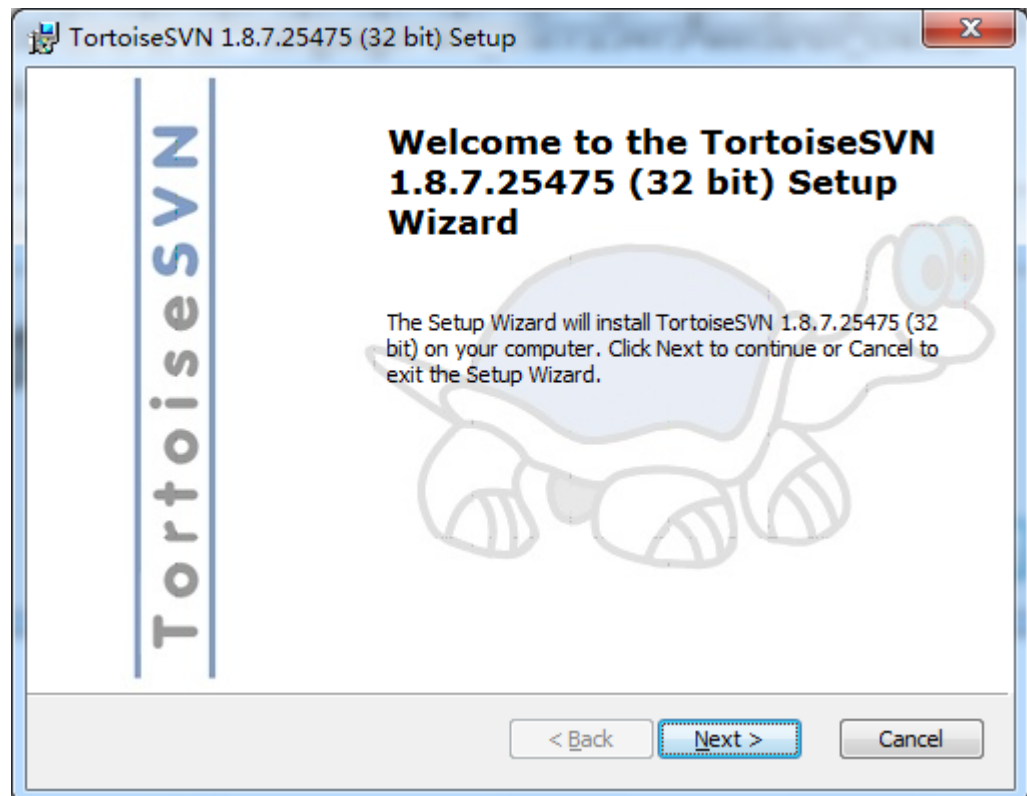
Win64 位：TortoiseSVN-1.8.7.25475-x64-svn-1.8.9.msi

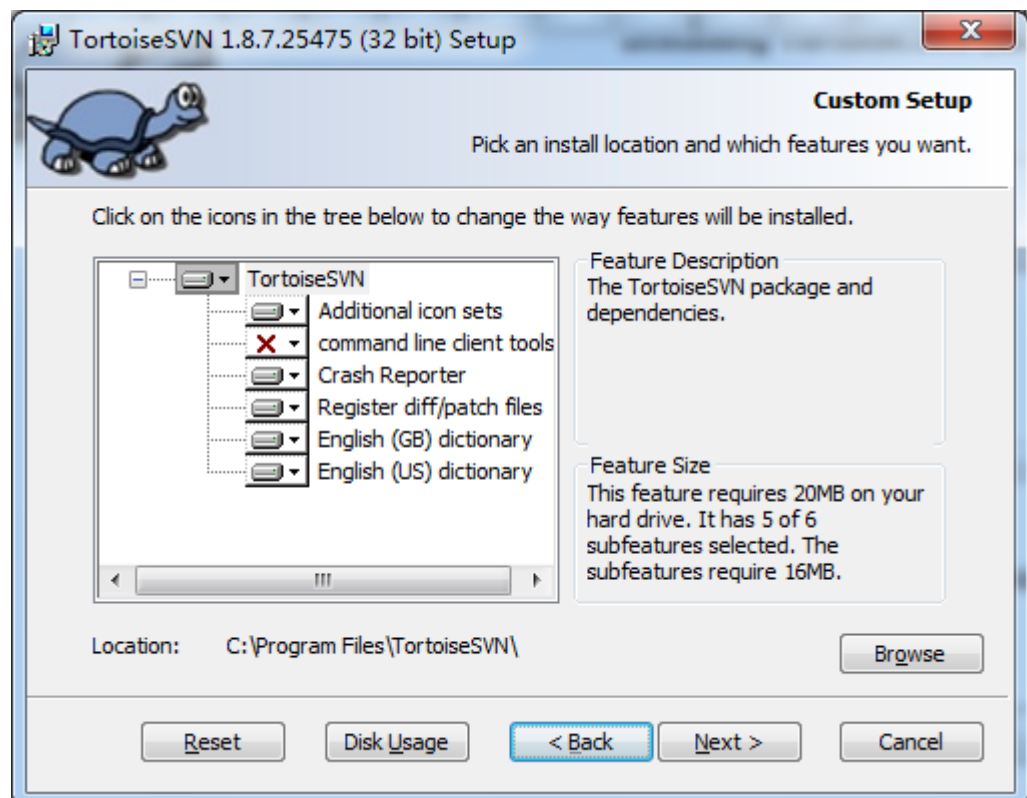
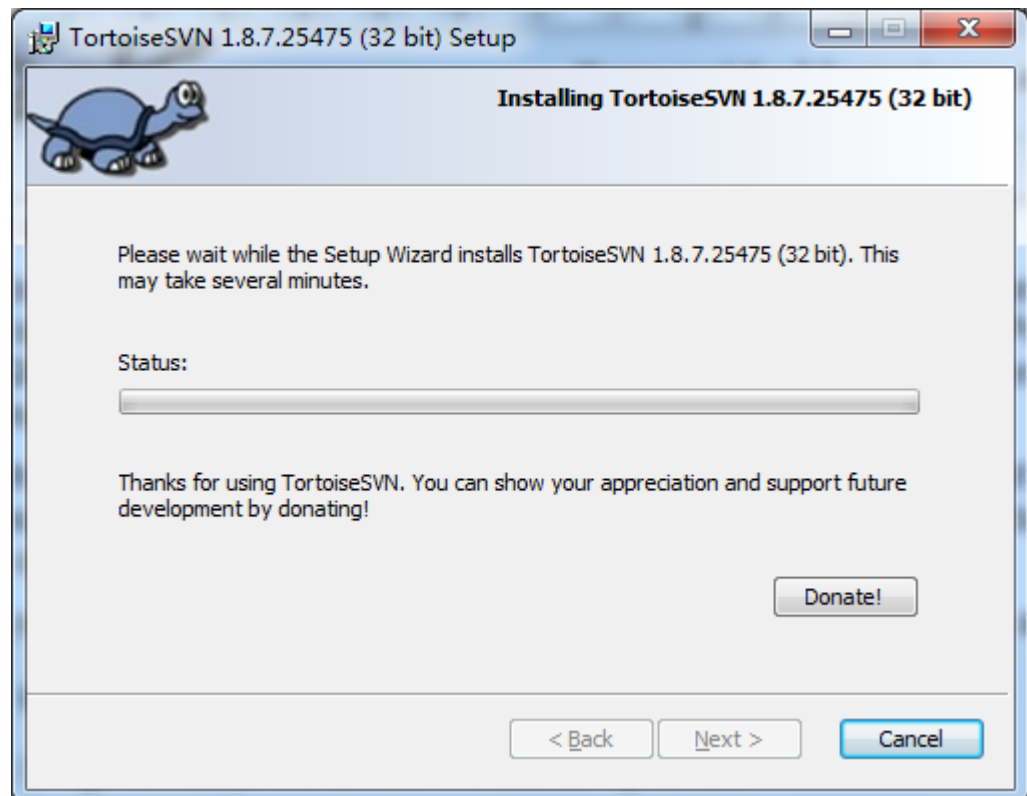
Win32 位语言包：LanguagePack\_1.8.7.25475-win32-zh\_CN.msi

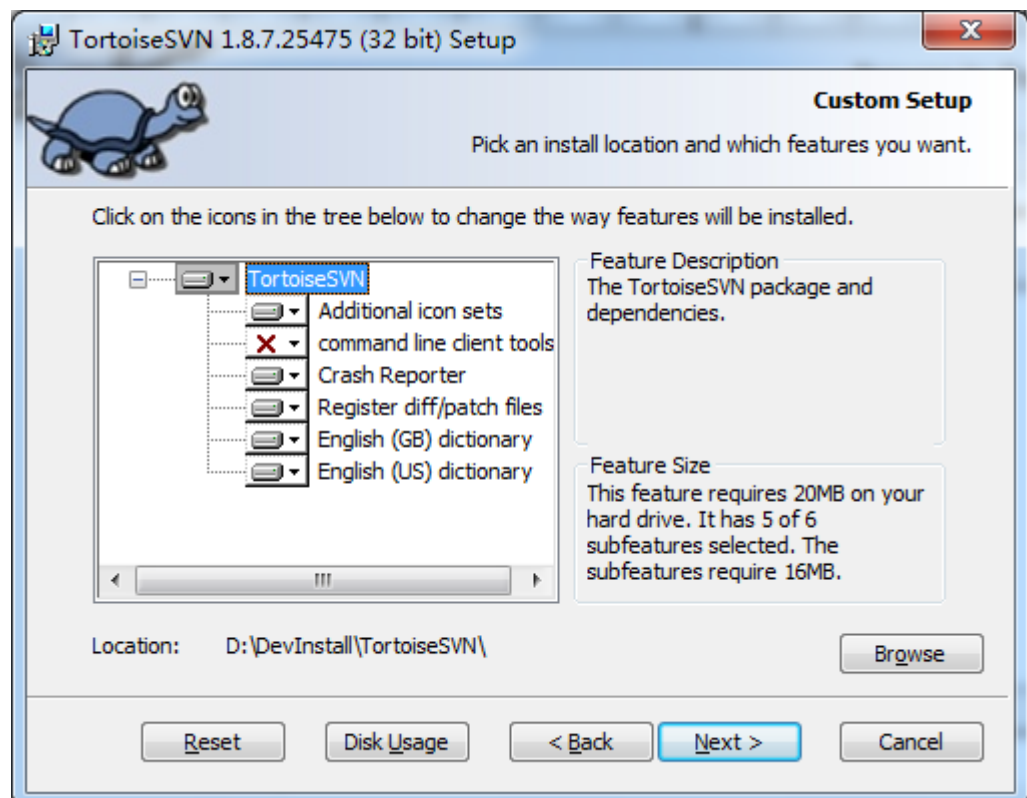
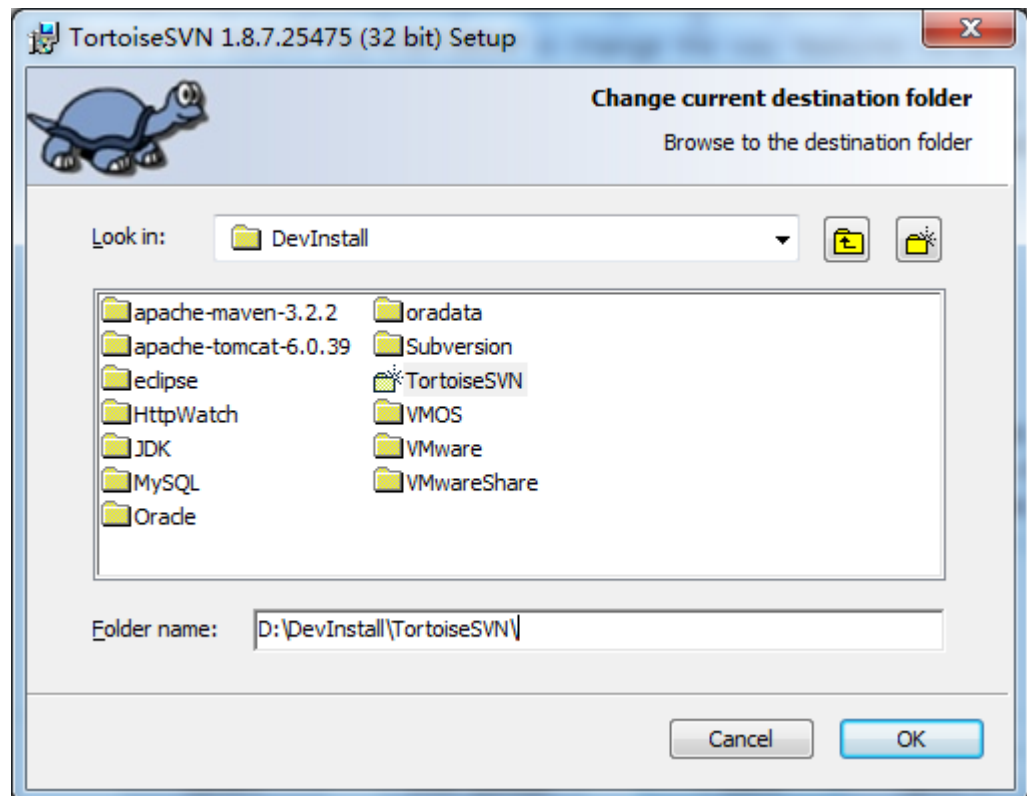
Win64 位语言包：LanguagePack\_1.8.7.25475-x64-zh\_CN.msi

②以 Win32 位为例

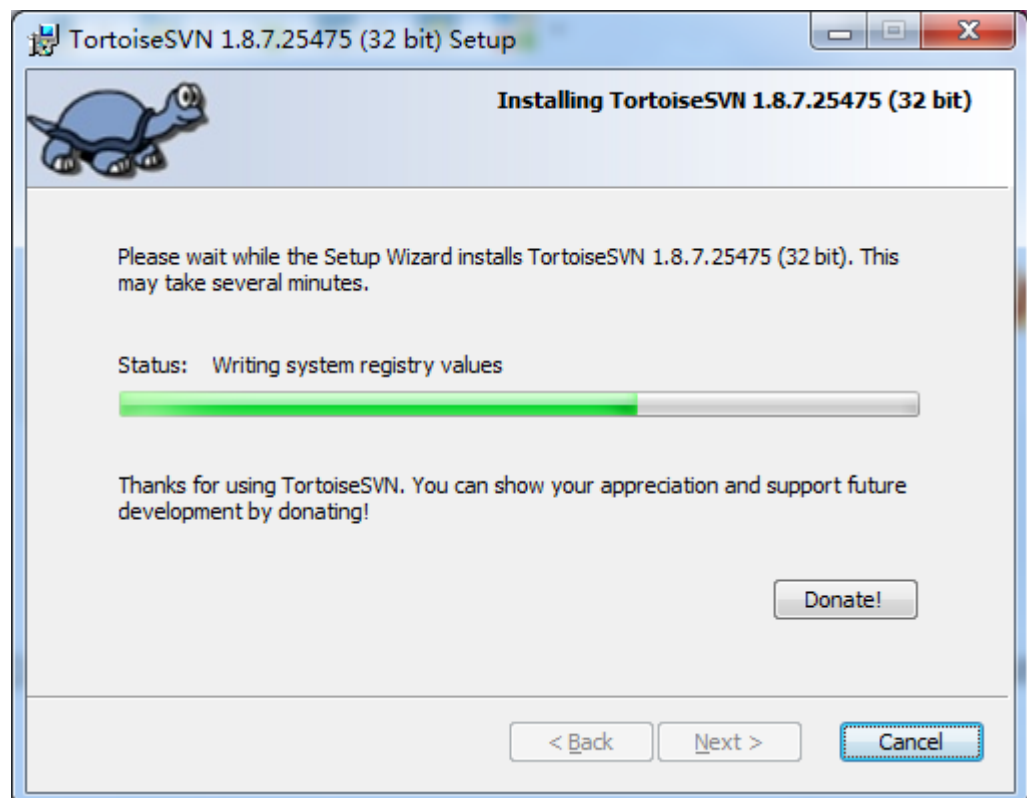
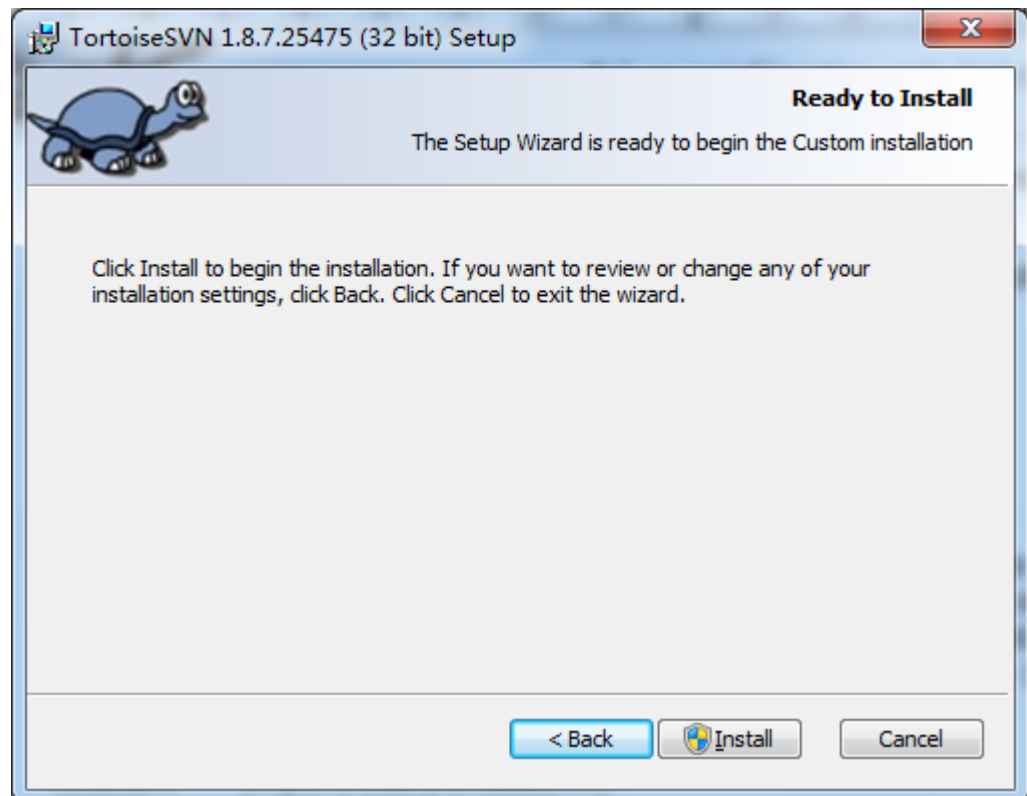


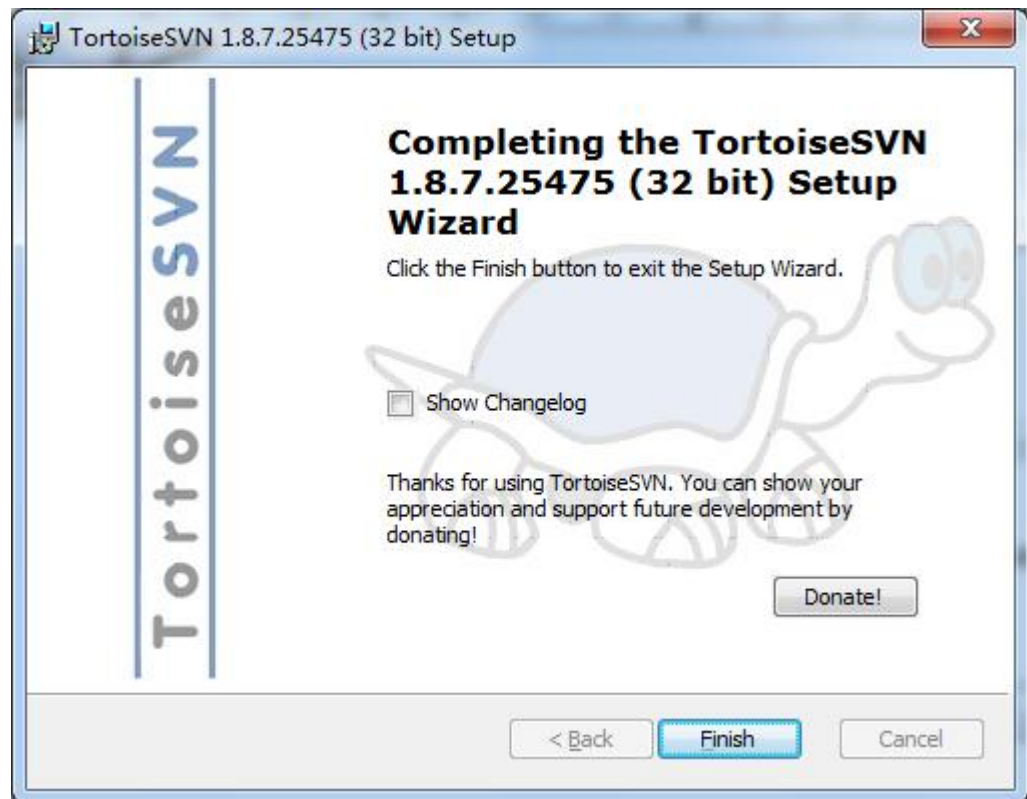




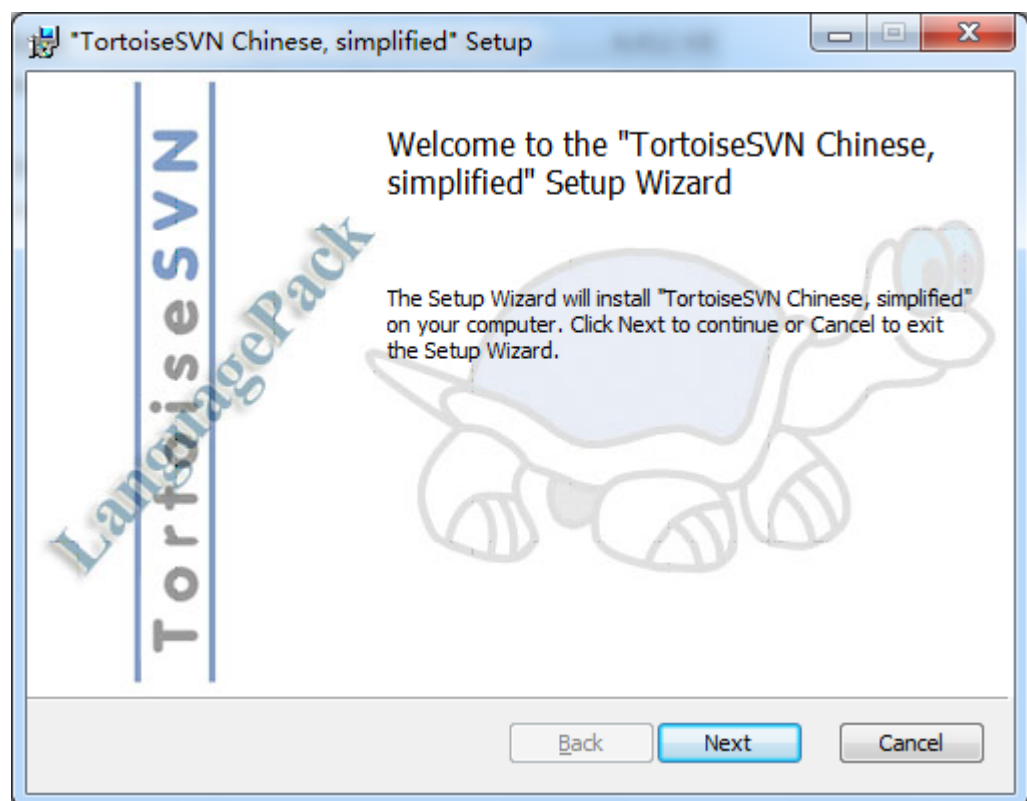


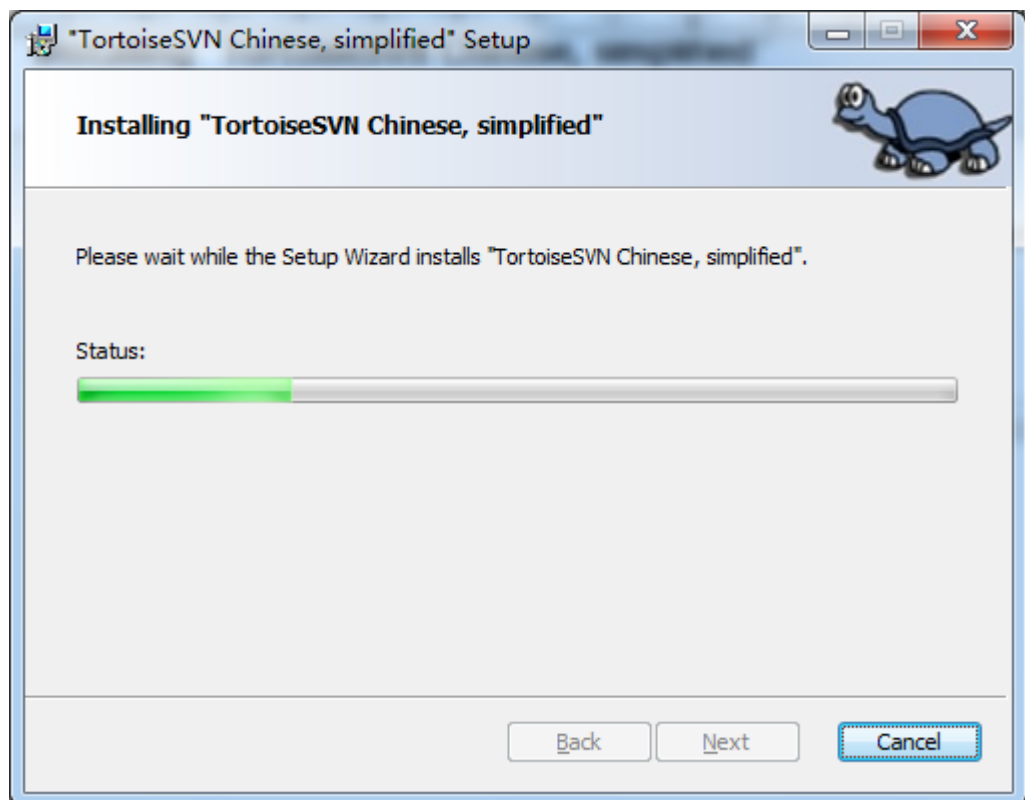
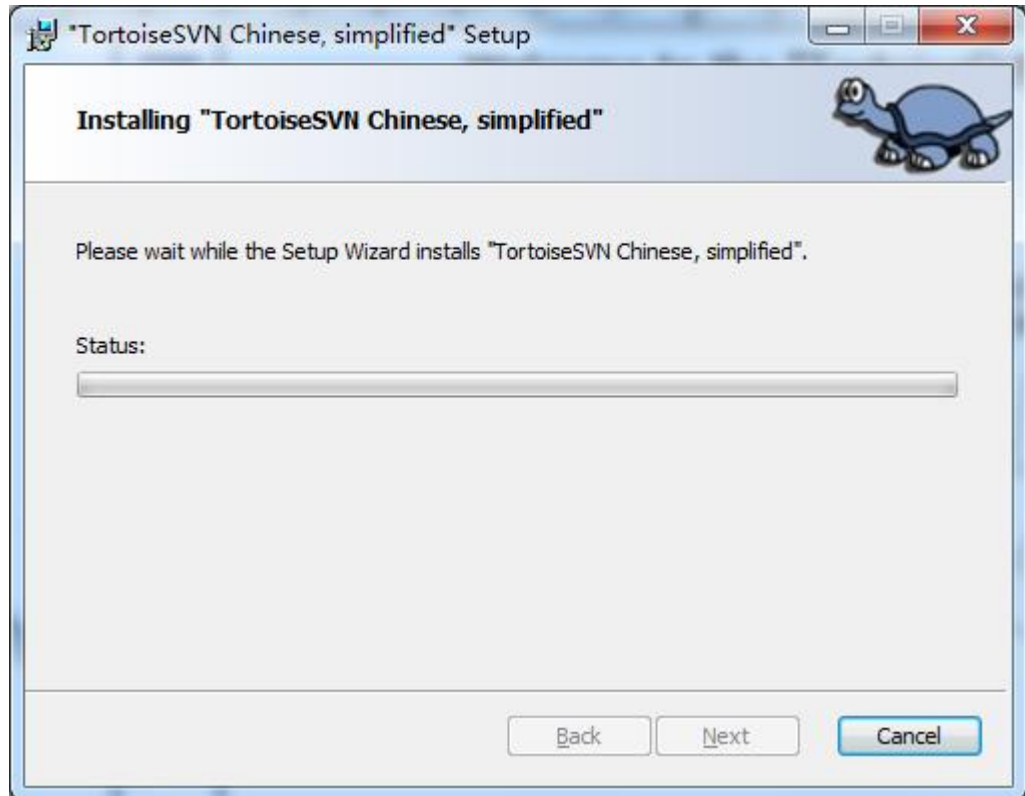


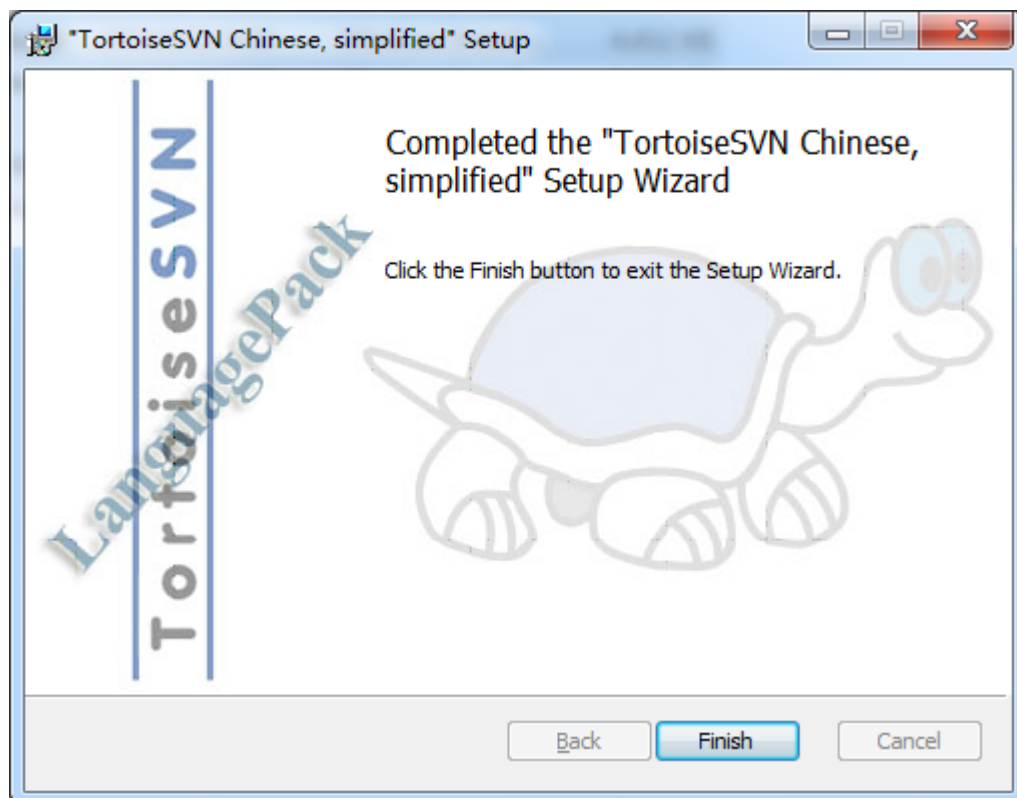




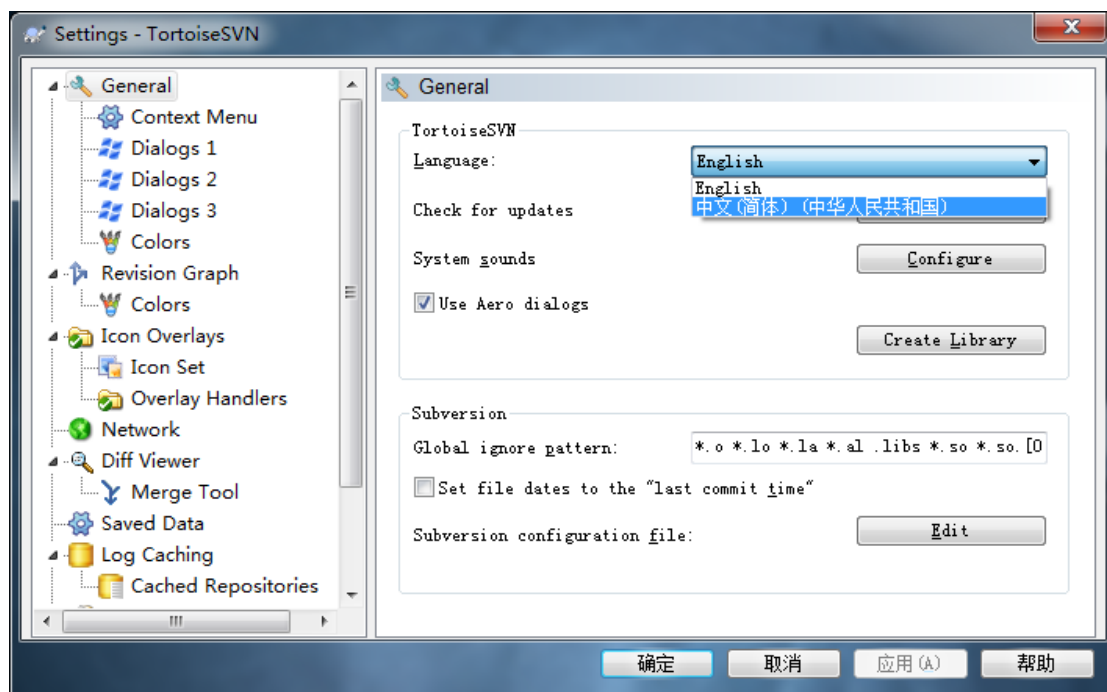
#### 10.5 中文语言包安装





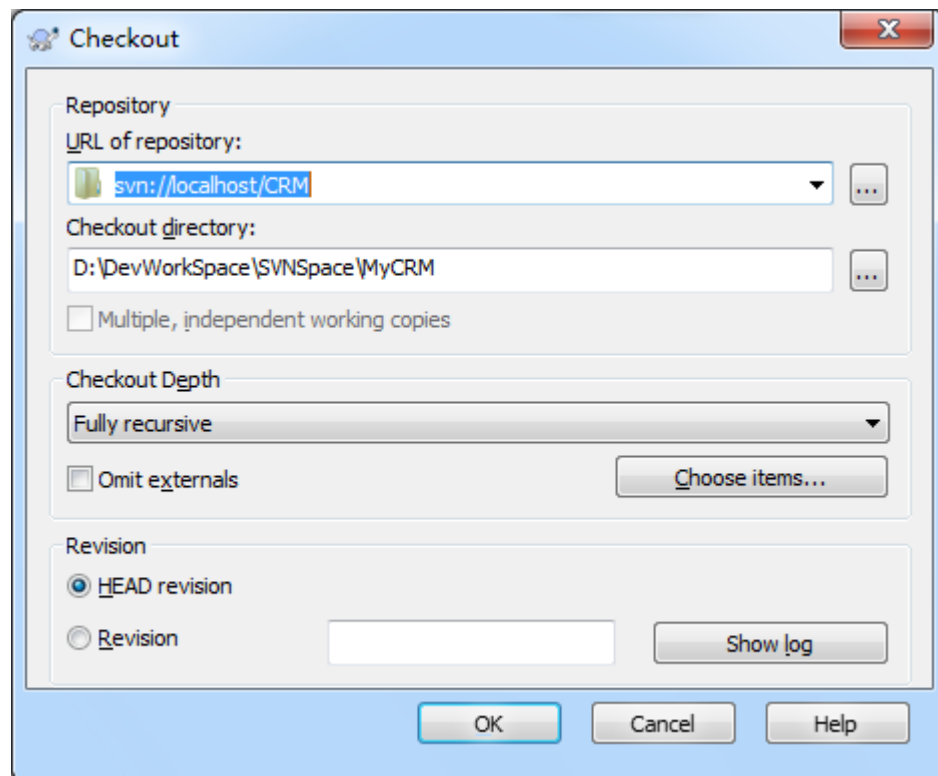
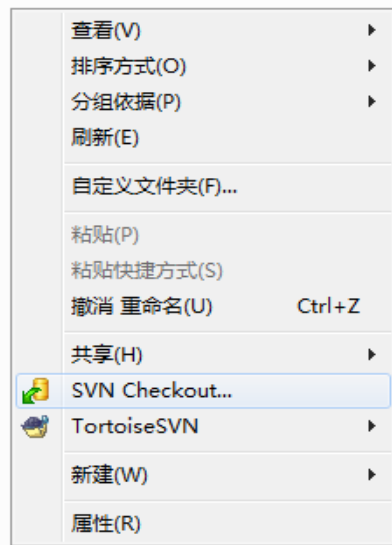


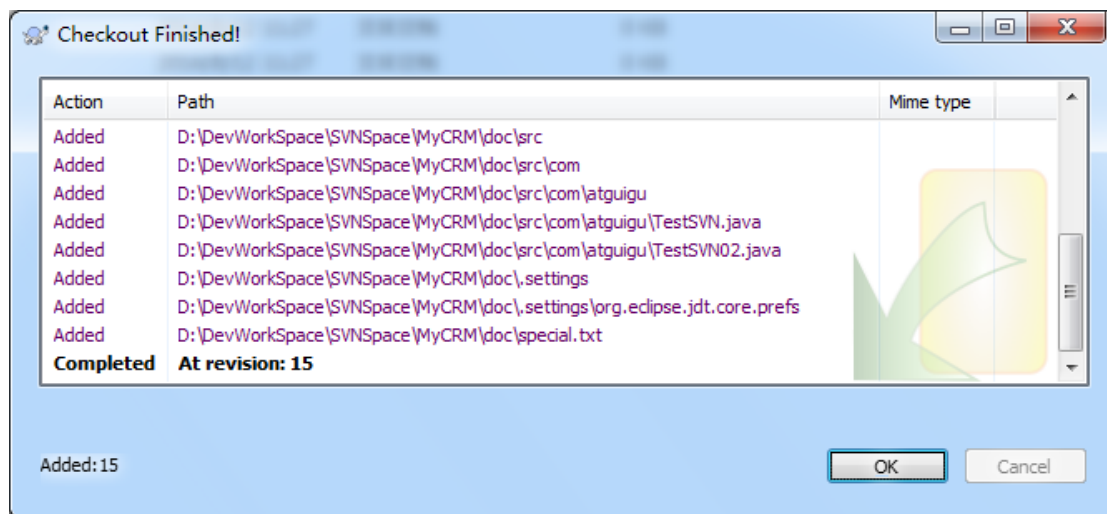
在任意目录下点右键→TortoiseSVN→settings



## 10.6 检出

- ① 创建一个目录用来存放检出得到的文件，例如 MyCRM
- ② 进入目录 MyCRM，点右键

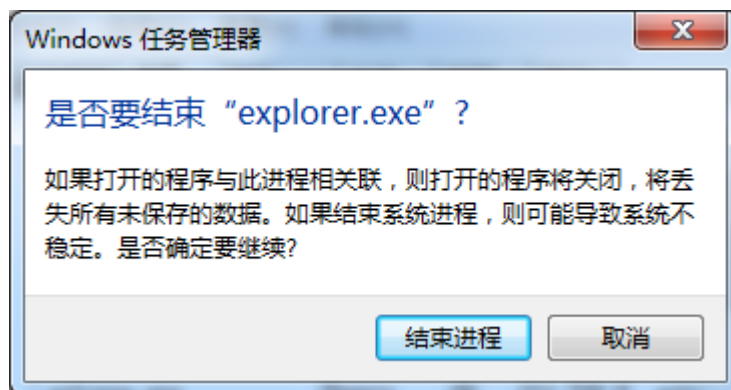
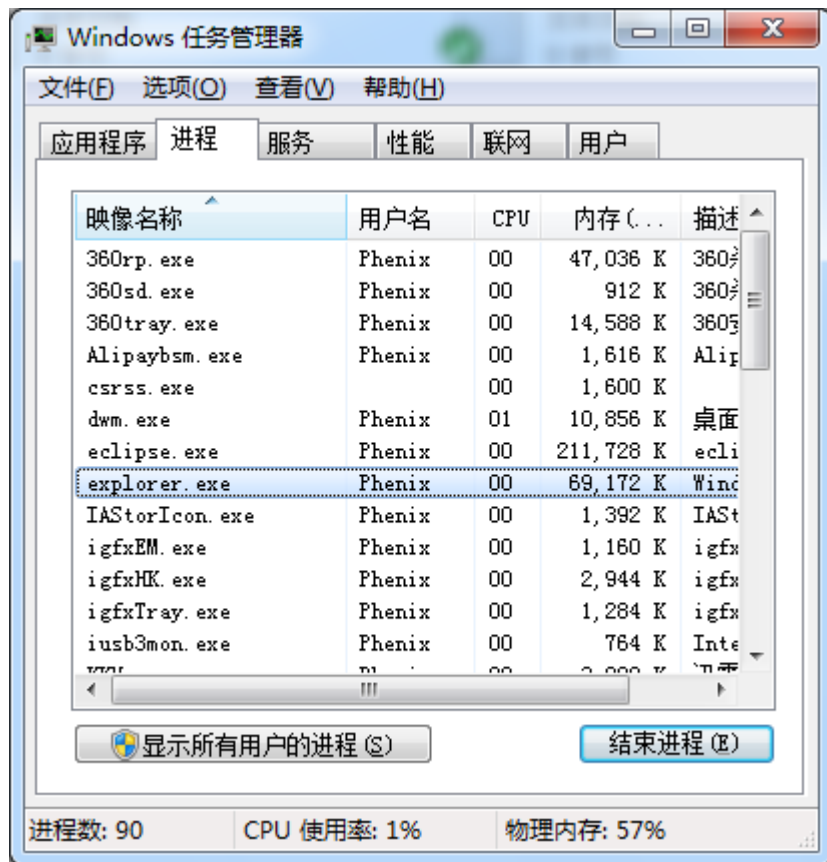




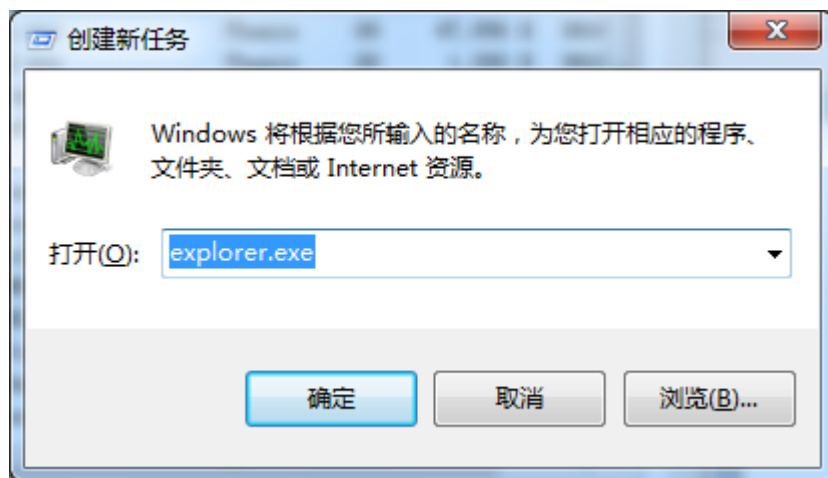
③可以看到检出得到的文件



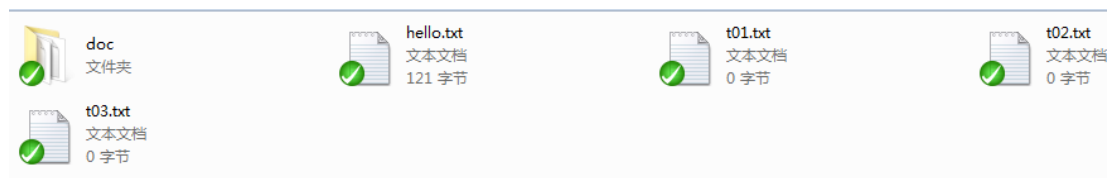
此时文件图标上没有任何标识。可能你会想到通过重启电脑的方式解决这一问题——其实不用这么麻烦。文件图标是受外壳程序控制的，我们只需要重启外壳程序——explorer.exe 就可以了。打开任务管理器，选中 explorer.exe 进程，结束进程，然后新建进程 explorer.exe 就可以了。



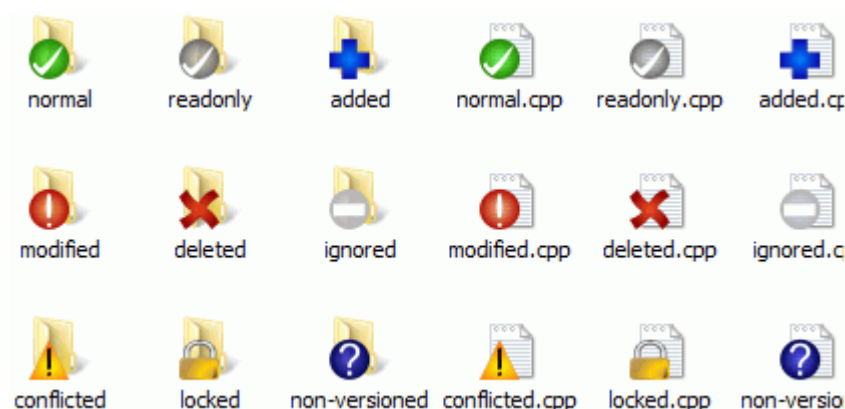






如果一切顺利的话，你会看到文件图标变成了这样：





#### ④ TortoiseSVN 图标含义





● 图标 1:  一个新检出的工作副本使用绿色的对勾做重载。表示 Subversion 状态正常。


● 图标 2:  在你开始编辑一个文件后，状态就变成了已修改，而图标重载变成了红色感叹号。通过这种方式，你可以很容易地看出哪些文件从你上次更新工作副本后被修改过，需要被提交。


● 图标 3:  如果在更新的过程中出现了冲突，图标会变成黄色感叹号。


● 图标 4:  如果你给一个文件设置了 `svn:needs-lock` 属性，Subversion 会让此文件只读，直到你获得文件锁。具有这个重载图标的文件来表示你必须在编辑之前先得到锁。

● 图标 5:  如果你拥有了一个文件的锁，并且 Subversion 状态是正常，这个重载图标就提醒你如果不使用该文件的话应该释放锁，允许别人提交对该文件的修改。

● 图标 6:  这个图标表示当前文件夹下的某些文件或文件夹已经被调度从版本控制中删除，或是该文件夹下某个受版本控制的文件丢失了。

● 图标 7:  加号告诉你有一个文件或目录已经被调度加入版本控制。

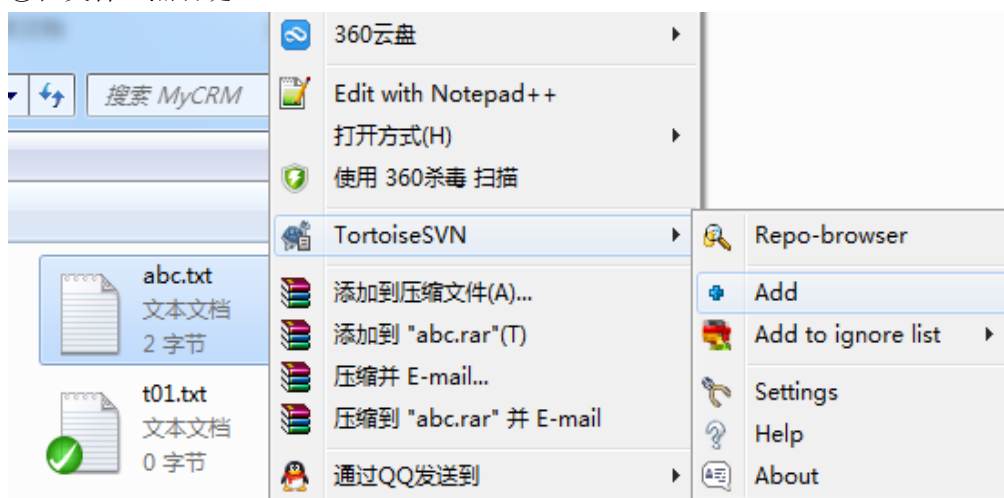
● 图标 8:  横条告诉你有一个文件或目录被版本控制系统所忽略。这个图标重载是可选的。

● 图标 9:  这个图标说明文件和目录未被版本控制，但是也没有被忽略。这个图标重载是可选的。

#### 10.7 纳入版本控制

① 新建文件 abc.txt

② 在文件上点右键

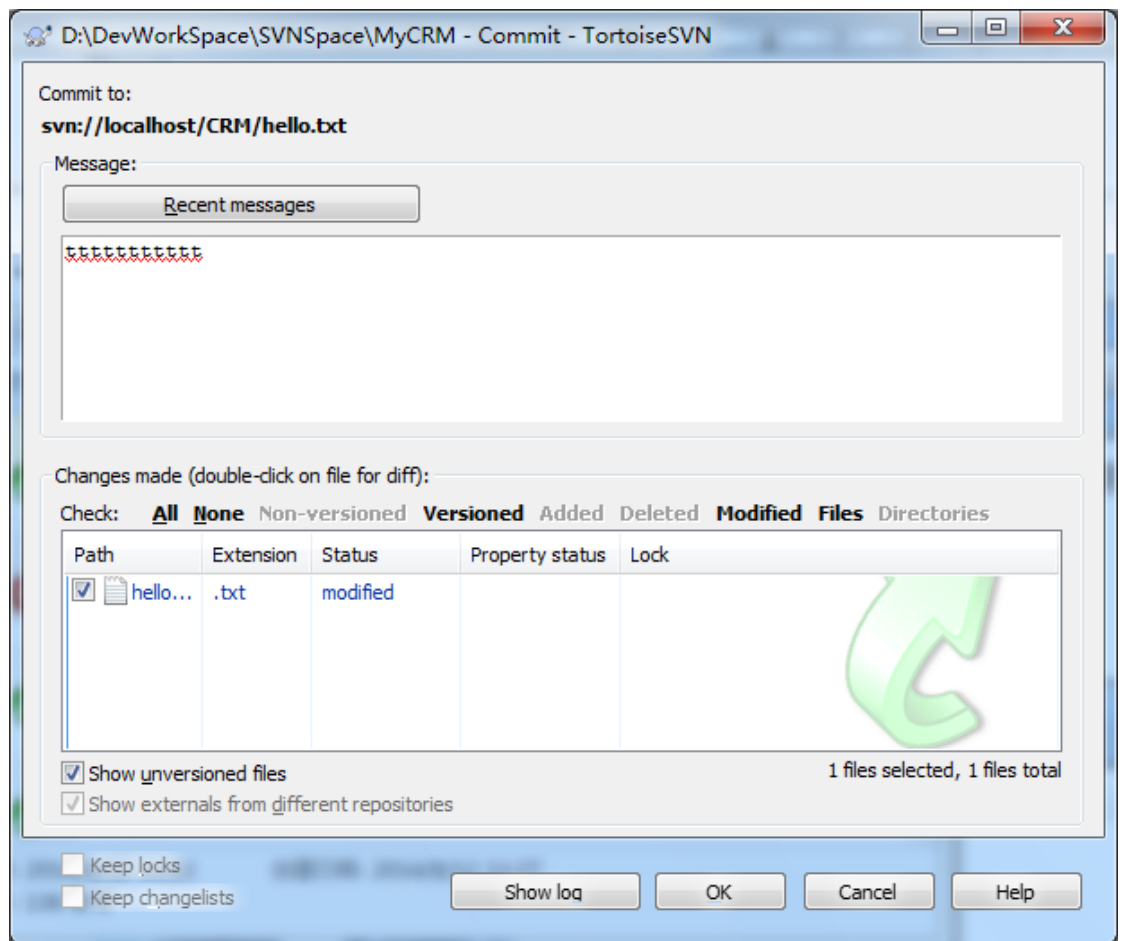


③ 添加后文件图标发生变化



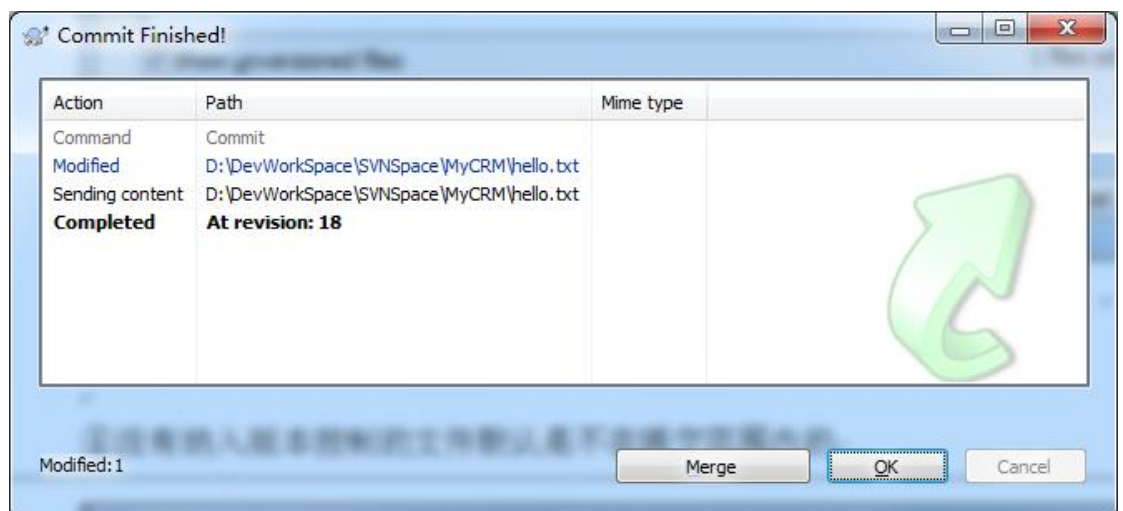
#### 10.8 提交

① 使用 TortoiseSVN 可以提交具体某一个文件，或某一个目录下的所有改变。方法就是在想要提交的项目下点右键，然后 SVN Commit...，就可以看到如下界面

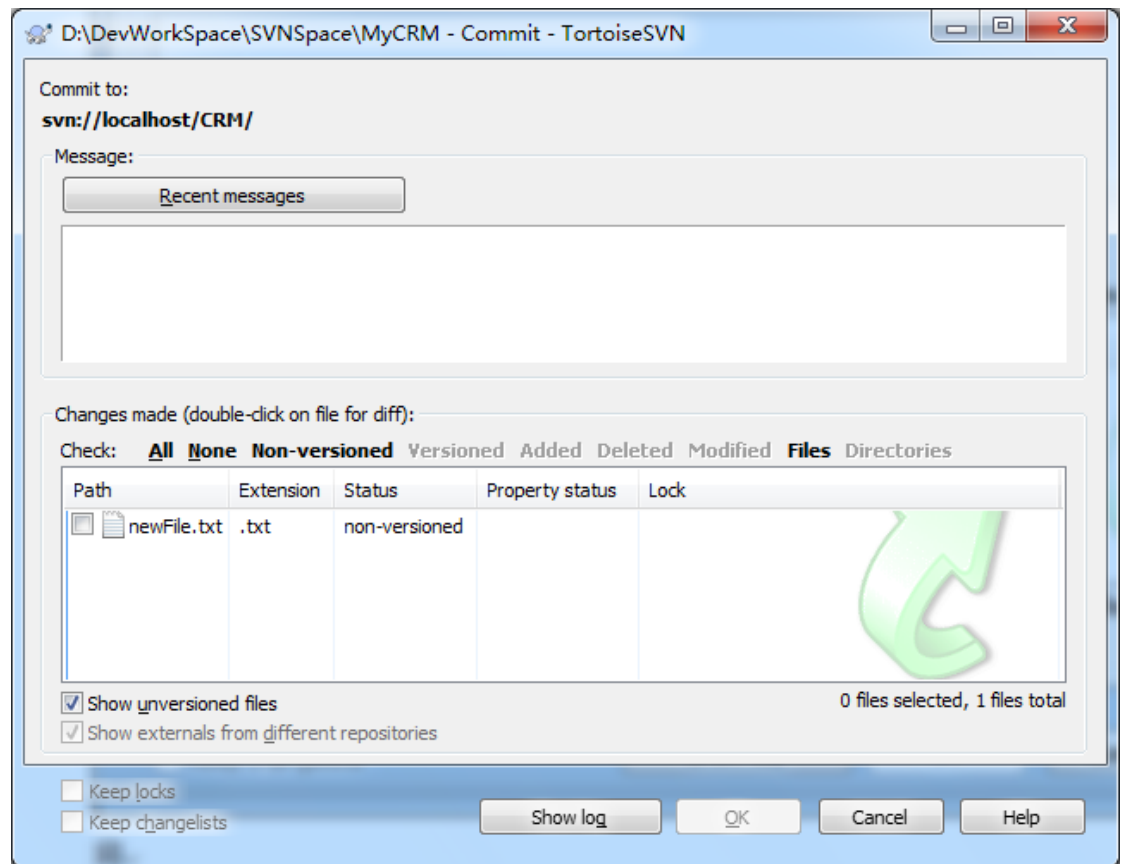


②日志内容如果不填，TortoiseSVN 会提交一个空字符串作为日志信息。

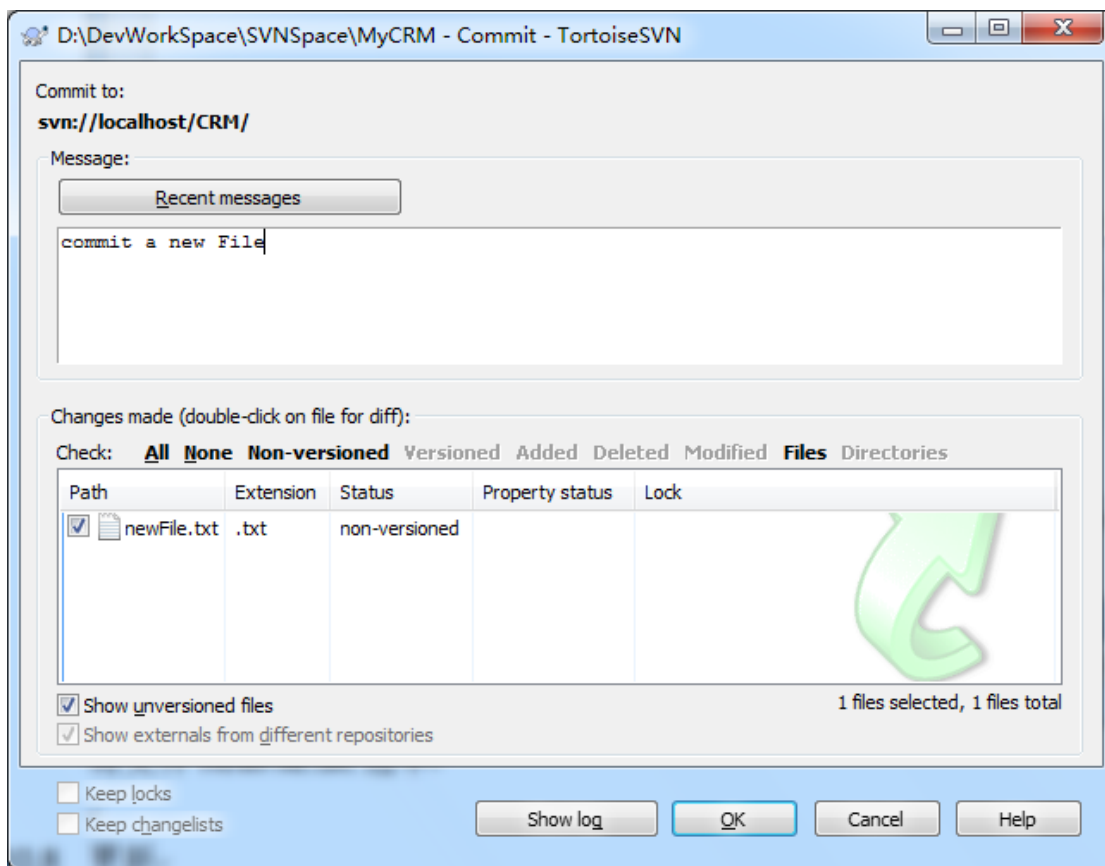
③提交后显示信息如下



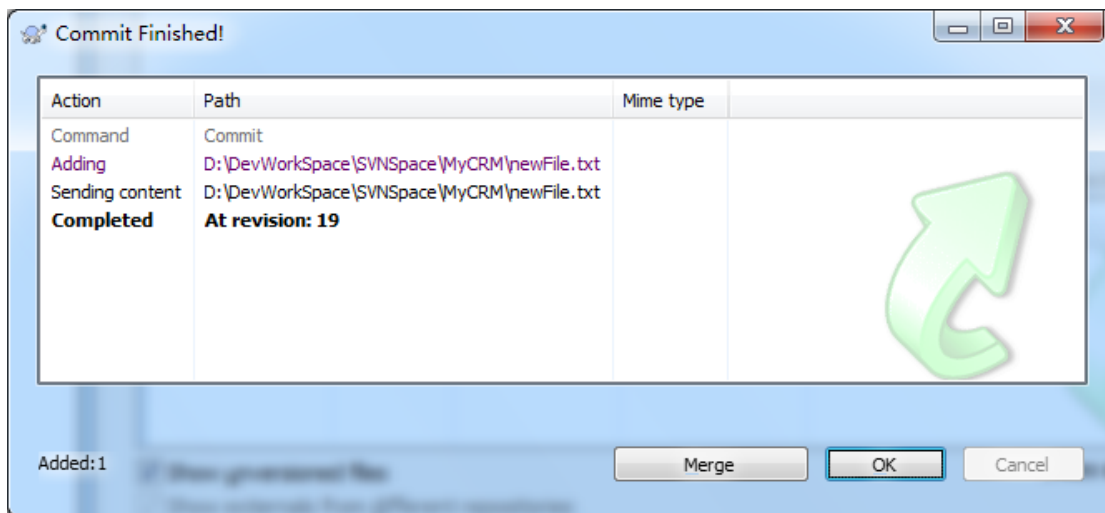
④没有纳入版本控制的文件默认是不在提交范围内的，直接在新创建的文件上点右键只能看到 add 操作的选项，如前所述。但在新创建的文件所在目录点右键选择 SVN commit...，可以看到如下界面



将文件 newFile.txt 选中

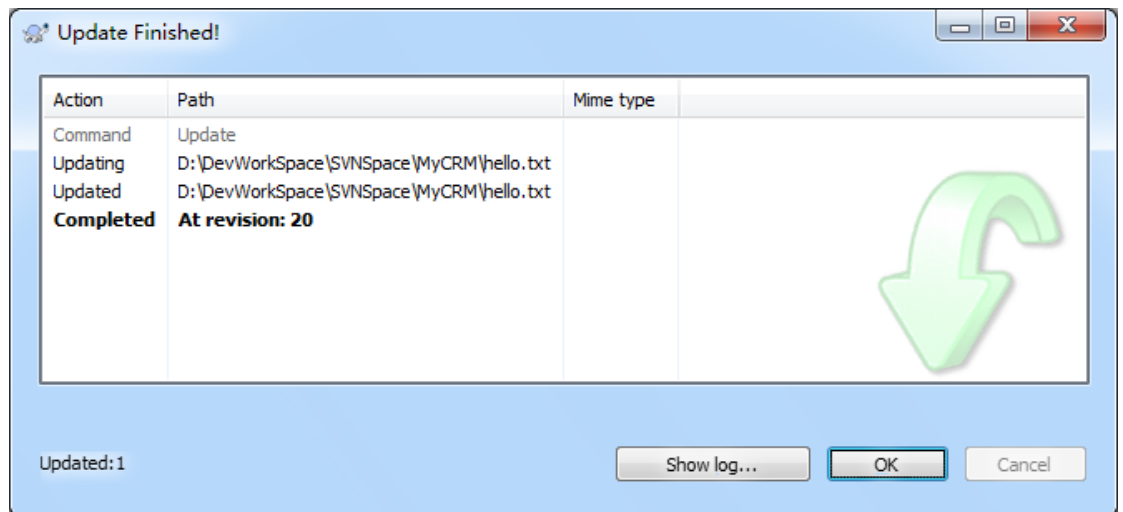


同样可以提交文件，TortoiseSVN 会帮我们自动将 newFile.txt 纳入版本控制



## 10.9 更新

在要更新的文件或目录上点右键→SVN Update

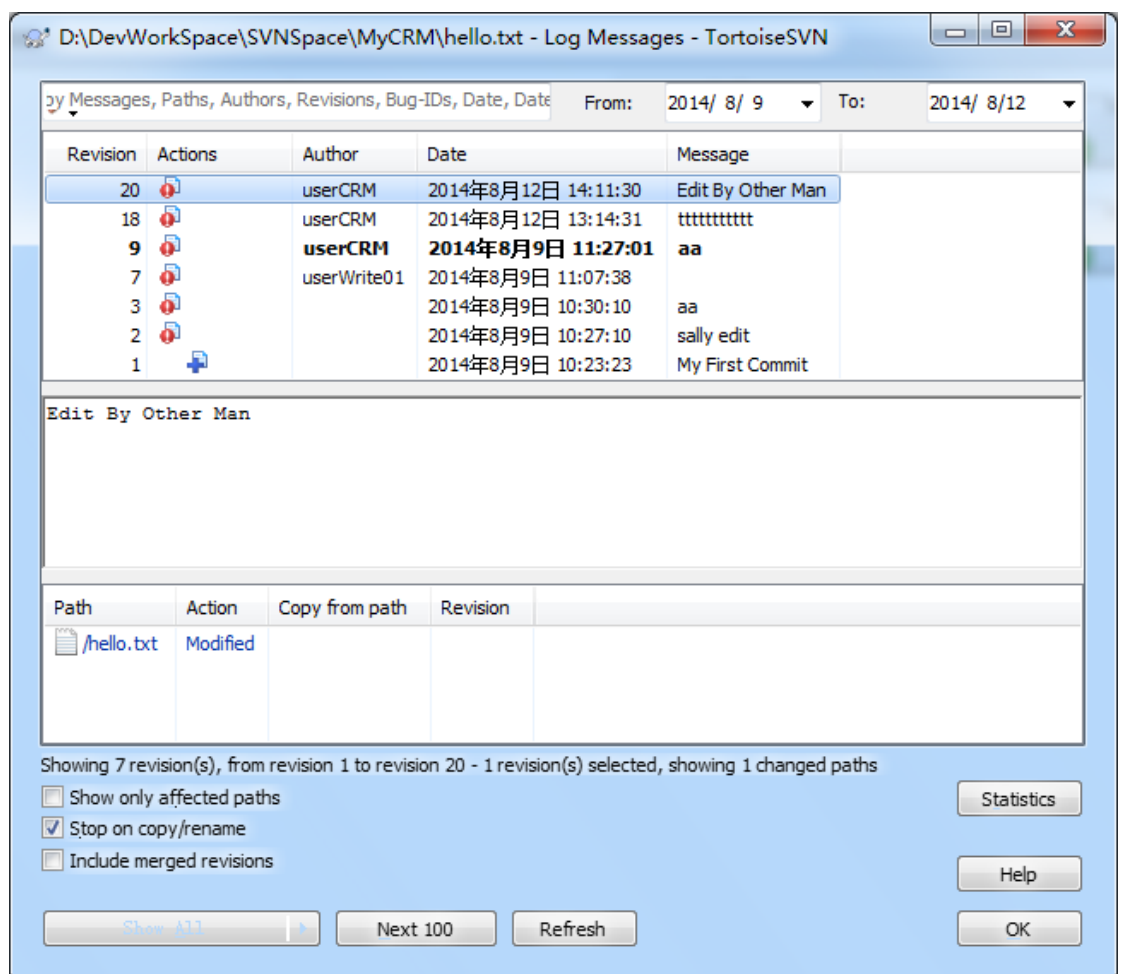


#### 10.10 回复历史版本

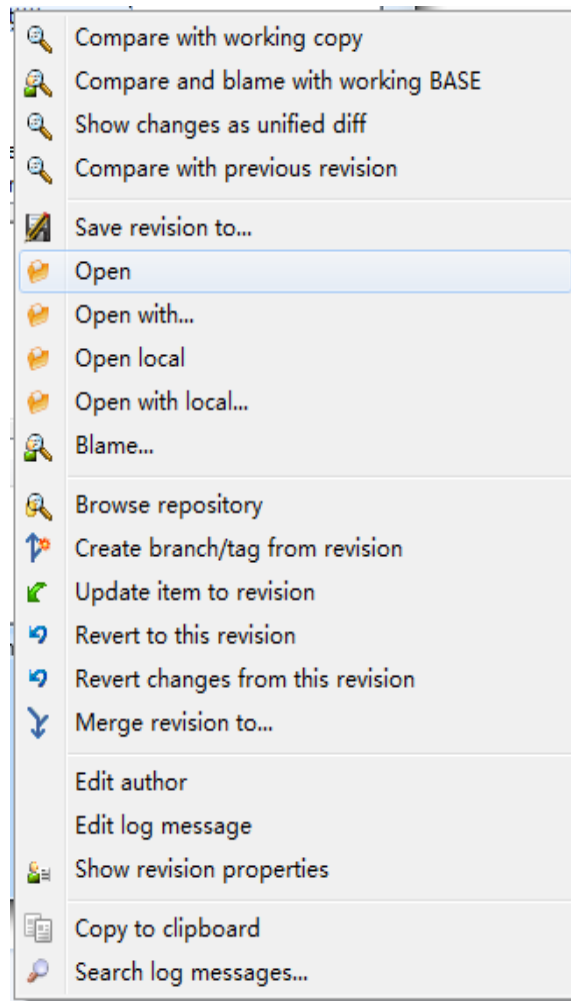
##### ①查看历史版本内容

[1]首先需要把对应版本库的匿名访问权限设置为 none: anon-access = none

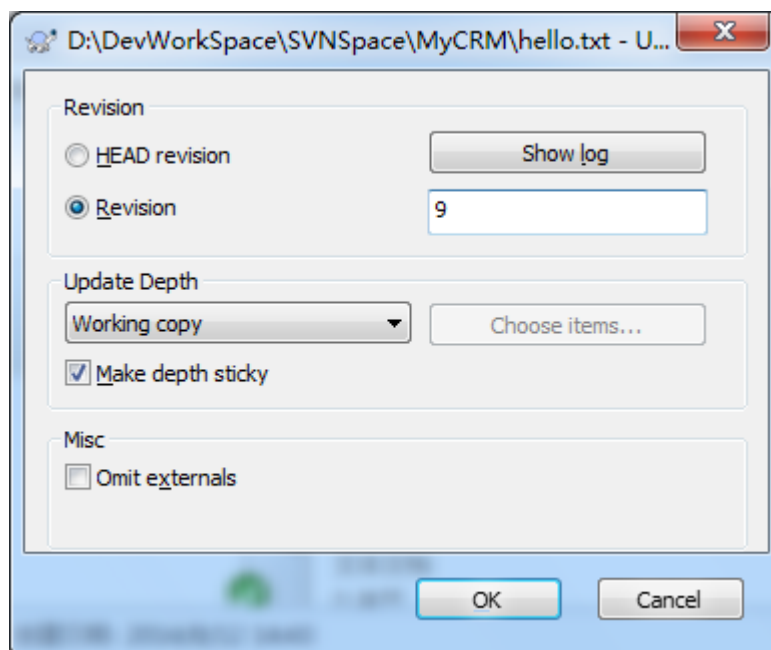
[2]在要查看历史版本的文件上点右键→TortoiseSVN→Show log



[3]在感兴趣的历史版本上点右键，可以与当前工作副本进行比较，或直接打开。

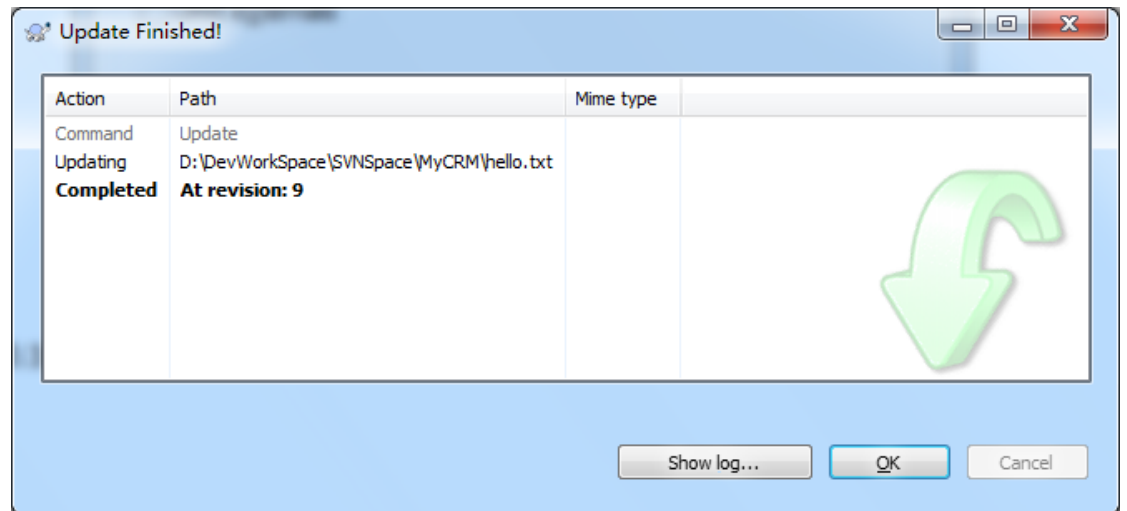


②在要回复历史版本的文件上点右键→Update to revision



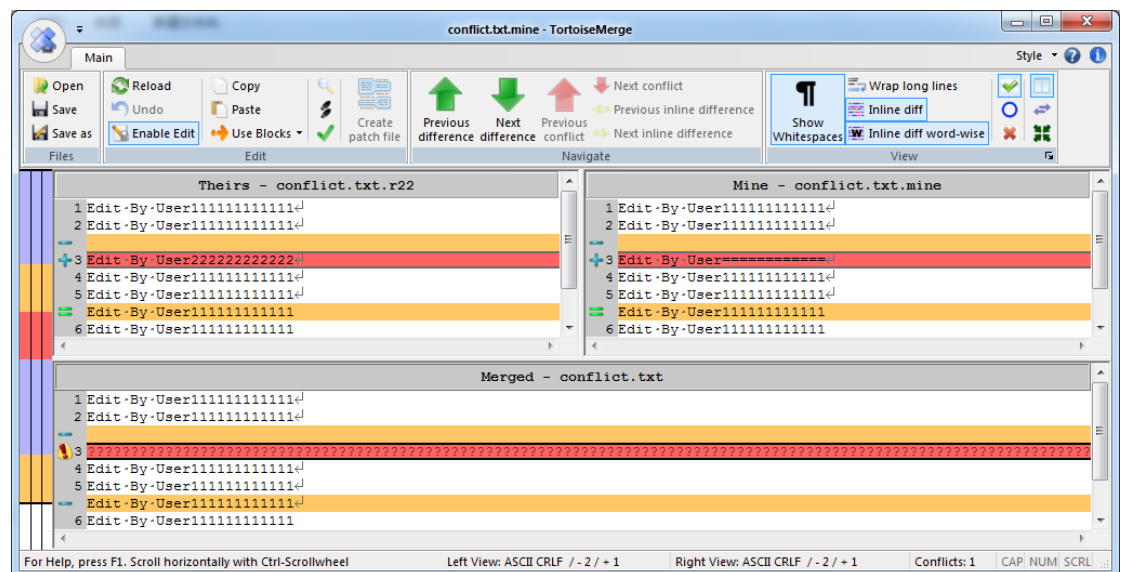


填上想要回到的版本即可



#### 10.11 解决冲突

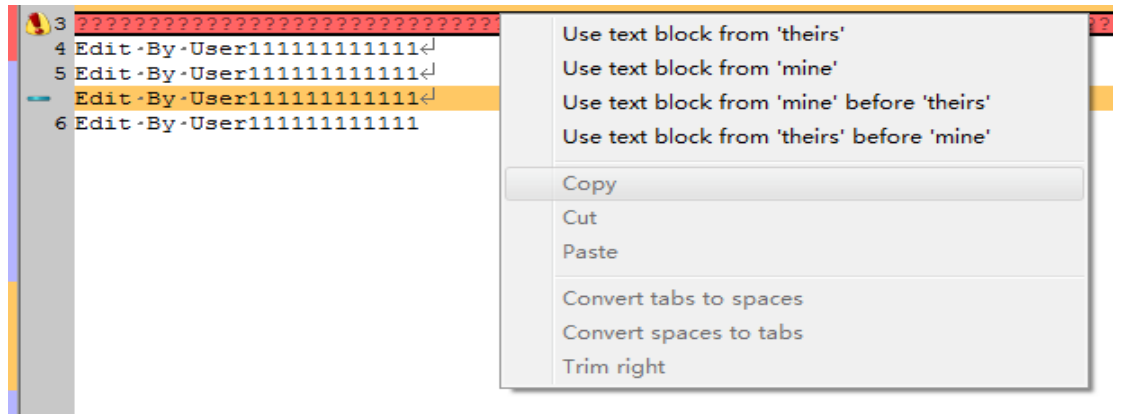
- ①文件发生冲突时的状态和在 Eclipse 中一样，这里就不赘述了。
- ②在冲突的文件上点右键→Edit Conflicts



- ③有“叹号”的行是发生冲突的行



- ④在冲突行点右键



可以选择四种操作：

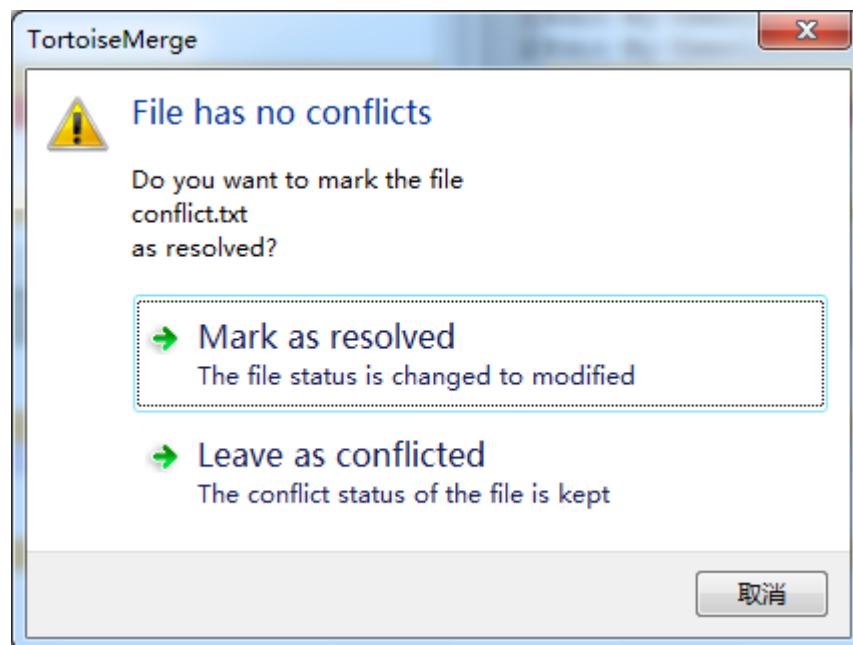
[1]使用我的

[2]使用他们的

[3]把我的放在他们的前面

[4]把他们的放在我的前面

⑤在冲突解决后，直接保存——这时 TortoiseSVN 自动弹出如下确认界面



⑥文件变为红色叹号标志，自动生成的三个文件被删除。提交修改即可。