SVN的使用

# SVN的使用

在实际开发中的我们会经常遇到以下问题：

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！

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梦的时光机啊？

——需求之二：代码还原！

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30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！

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！

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在应用初始化时赋值！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！

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！

## SVN的概述

### 项目管理中版本控制的问题

通常软件开发由多人协作开发，如果对代码文件、配置文件、文档等没有进行版本控制，将会出现很多问题：

* 代码管理混乱
* 备份多个版本，占用磁盘空间大
* 解决代码冲突困难
* 容易引发BUG
* 难于追溯问题代码的修改人和修改时间
* 难于恢复至以前正确版本
* 无法进行权限控制
* 项目版本发布困难

### 什么是版本控制

版本控制(Revision control)是维护工程蓝图的标准做法，能追踪工程蓝图从诞生到定案的过程。是一种记录多个文件内容变化，以便将来查阅特定版本修订情况的系统。

### 主流的版本控制

VSS:Visual Source Safe(Microsoft Visual Studio成员)主要任务是负责项目文件的管理

CVS:march-hare出品的一套用于进行文件版本控制软件

SVN:Apache软件基金会名下的一套用于进行文件版本控制软件

在2000年初，开发人员要写一个CVS的自由软件代替品，它保留CVS的基本思想，但没有它的错误和局限，保留CVS的基本特性但去除CVS的bug和不好的特性。

在2000年2月，他们联系《使用CVS开发开源项目》（Open Source Development with CVS）(Coriolis, 1999)的作者Karl Fogel，并征求了他是否愿意在这个新的项目中担任一个角色。巧合的是，当时Karl已经和他的朋友Jim Blandy讨论了一个关于新的版本控制系统的设计。在1995年，这两人就成立了Cyclic Software，一个提供CVS的商业支持的软件公司。虽然他们经营商业服务，但是仍然在每天都在工作中使用CVS。使用CVS的挫折感使得Jim认真思考更好的方法来管理数据，不但确定名字为“Subversion”，而且完成了Subversion档案库的基础设计。

当CollabNet的电话到来时，Karl立即答应了加入项目中，而且Jim让他的雇主RedHat Software同意让他在这个项目中不定期工作。CollabNet雇用了Karl和Ben Collins-Sussman，并在5月开始了详细设计工作。在得到了来自CollabNet的Brian Behlendorf、Jason Robbins和Greg Stein（当时是一名活跃在WebDAV/DeltaV规范过程的自由程序员）很多创意的帮助下，Subversion很快地引起了一个活跃开发者社区的注意。它找出并欢迎很多同样在CVS上受到挫折的社员能来为这个项目做点什么。

Subversion 最初的设计Team定下了几个简单的目标。 它必须在功能上可取代 CVS，也就是说, 所有 CVS 可做到的事, 它都要能够作到。 在修正最明显的瑕疵的同时, 还要保留相同的开发模式。 还有, Subversion 应该要和 CVS 很相像, 任何 CVS 使用者只要花费少许的力气, 就可以很快地上手。

经过十四个月的编码后, Subversion 于2001年8月31日开始实现 “自行管理”。 也就是说, 开发人员不再使用 CVS 来管理 Subversion 的代码, 而以 Subversion 自己来管理。

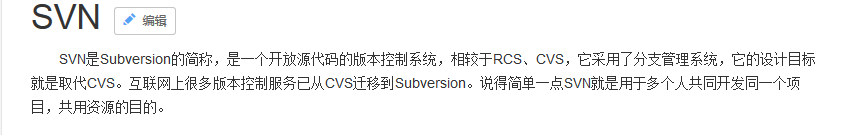
2009年11月，Subversion被Apache Incubator专案所接收。

2010年1月，正式成为Apache软件基金会的一个顶级专案，所以为Apache Subversion.

目前Apache Subversion的主席为Greg Stein, 项目领导者Release manager为Wandisco公司。

### 什么是SVN

SVN（Subversion）是近年来崛起的版本管理工具，在当前的开源项目里(J2EE)，几乎95%以上的项目都用到了 SVN。Subversion 项目的初衷是为了替换当年开源社区最为流行的版本控制软件 CVS，在 CVS的功能的基础上有很多的提升同时也能较好的解决 CVS 系统的一些不足。



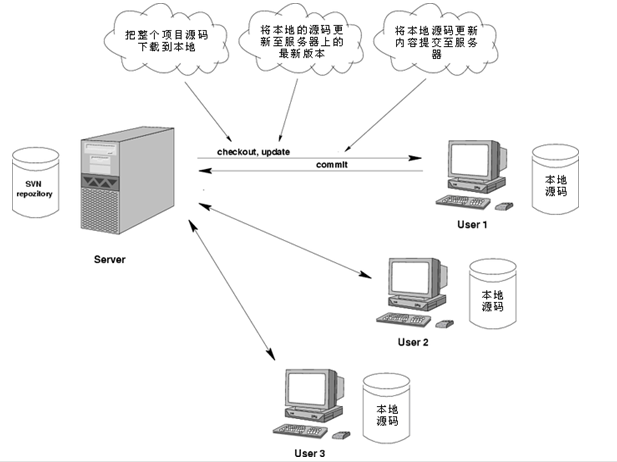
### SVN的作用

针对软件研发企业的软件生产过程而言，SVN用于管理整个开发过程中的源码，进行版本控制。

## SVN的使用

### SVN的使用方法

svn是基于客户/服务器模式：



* 复制-修改-合并方案(Subversion默认的模式)

在这种模型里，每一个客户读取项目配置库建立一个私有工作副本——版本库中文件和目录的本地映射。用户并行工作，修改各自的工作副本，最终，各个私有的副本合并在一起，成为最终的版本，这种系统通常可以辅助合并操作，但是最终要靠人工去确定正误。

* 锁定-修改-解锁方案

在这样的模型里，在一个时间段里配置库的一个文件只允许被一个人修改。

### SVN的工作方式

**客户端接入方式**

**命令行操作**

**图形界面操作**

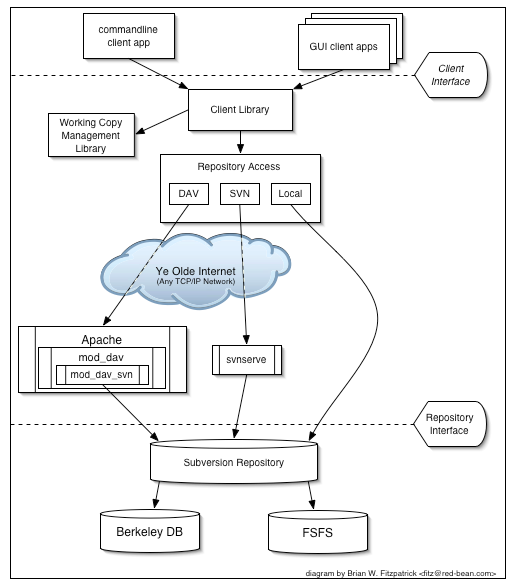
**数据库存储**

**服务端接入方式**

**文件系统存储**

**SVN服务**

**第三方服务器**



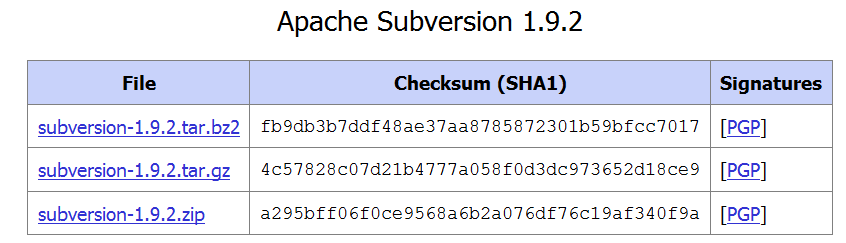
**SVN仓库**

## SVN服务的安装

### 官方安装包

官方网站：http://subversion.apache.org/

下载：http://subversion.apache.org/download.cgi

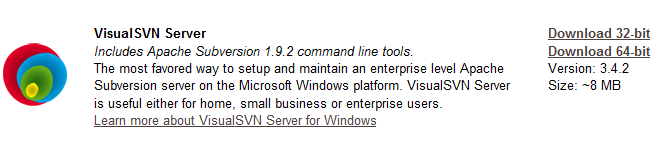


官方提供的服务端安装包，安装后需要通过命令行操作，适用于专业配置管理员使用。

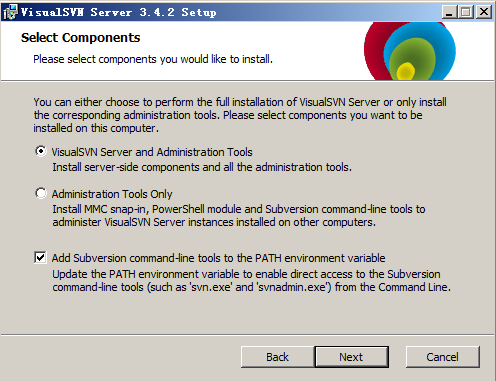
### 图形化服务端

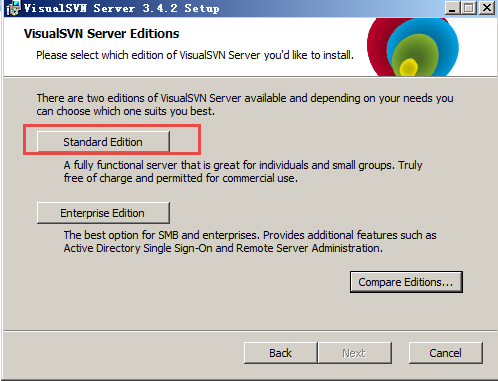
志愿者开发的图形化操作界面的svn服务端，它适用于普通软件开发人员使用。

下载地址：https://www.visualsvn.com/downloads/

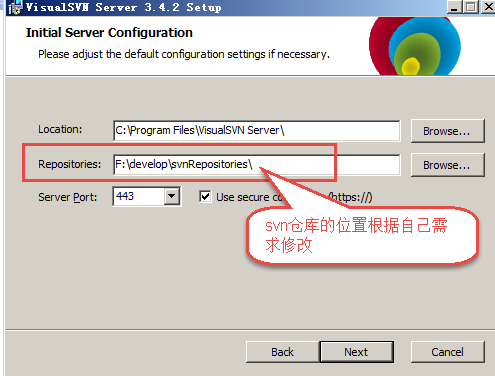


### 安装图形化服务端

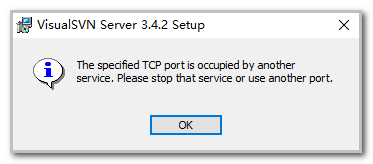




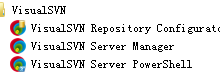
注意：如果端口被占用，请更换其他端口并记住



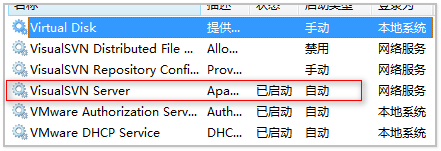
端口被占用的提示：



查看程序菜单：



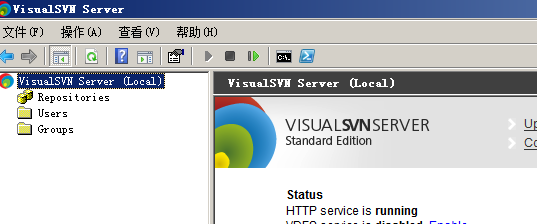
查看服务，VisualSVN成功启动：



### 创建仓库

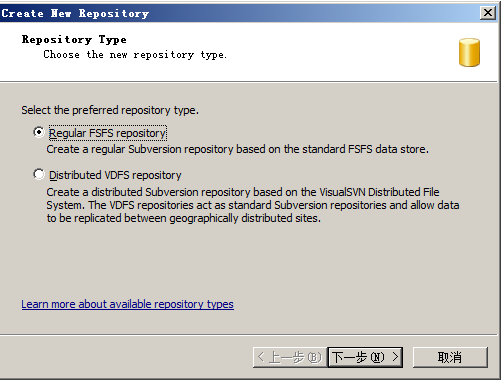
svn服务端创建完成需要创建仓库，仓库中存放要版本控制的文件。

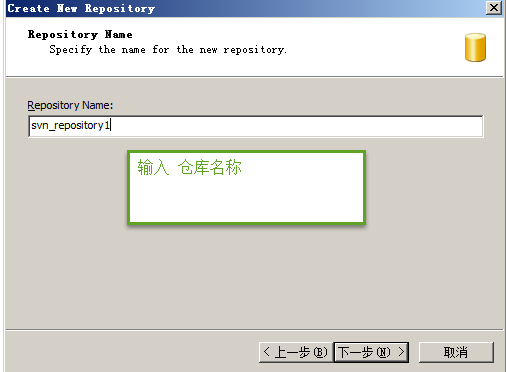
打开svn:



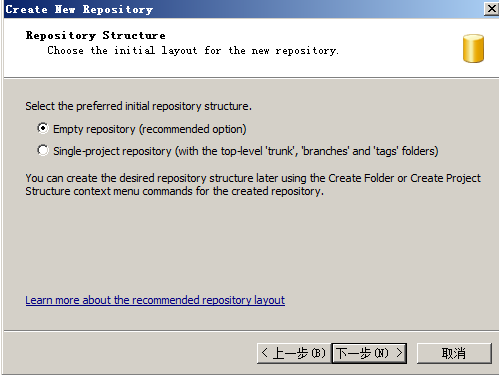


选择文件存储方式：常规和分布式

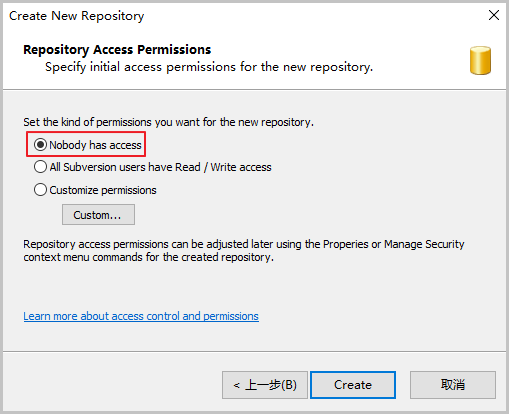




创建一个空的仓库：

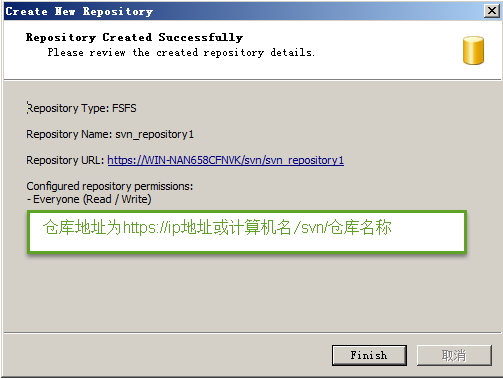


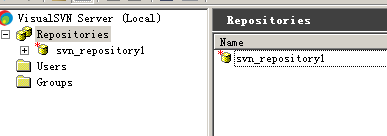
设置用户访问仓库权限：都设置没有权限



仓库创建成功：

仓库地址为https://ip地址或计算机名/svn/仓库名称





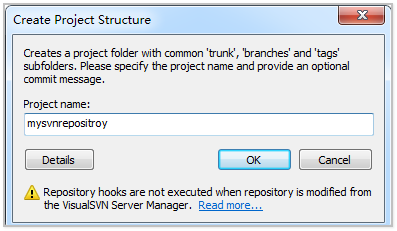
### 创建工程目录

仓库中存放开发项目代码、文档等，需要创建一个工程目录。

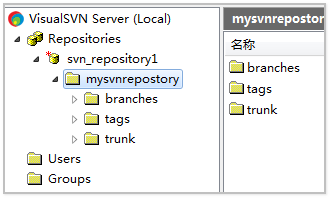
创建一个项目存放的目录：



给该目录取一个名字：



最终创建如下：



* trunk为主开发目录:

一般的，我们的所有的开发都是基于trunk进行开发

* branches为分支开发目录

如果发现了上一个已发行版本（Released Version）有一些bug，或者一些很急迫的功能要求以及添加一些新的技术，而正在开发的版本（Developing Version）无法满足时间要求，这时候就需要在上一个版本上进行修改了。应该基于发行版对应的tag，做相应的分支（branch）进行开发，最终可以和trunk进行合并。

* tags为tag存档目录（不允许修改）：

用于记录各种开发版本。

## 权限管理(了解)

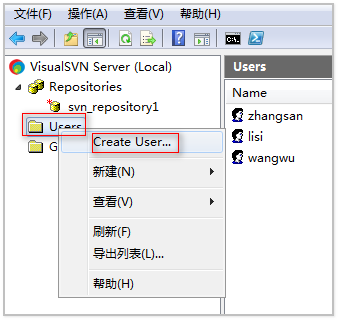
### 认证授权机制

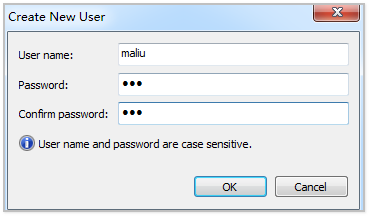
在企业开发中会为每位程序员、测试人员等相关人员分配一个账号，用户通过使用svn客户端连接svn服务时需要输入账号和密码，svn服务对账号和密码进行校验，输入正确可以继续访问，当用户访问仓库下某个目录时，svn服务对用户进行授权，如果用户拥有该目录的访问权限方可访问。

判断账号和密码输入是否正确的过程即认证过程。

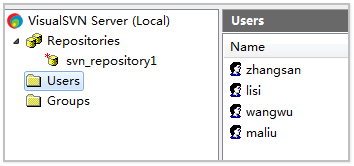
判断用户是否拥有目录的读/写权限时即授权过程。

### 创建用户





查看已创建的用户：



修改用户：



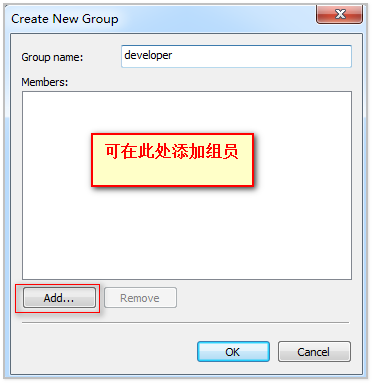
### 创建组

可以给这些帐号进行分组，这样就可以方便统一给一个组的帐号设置权限。

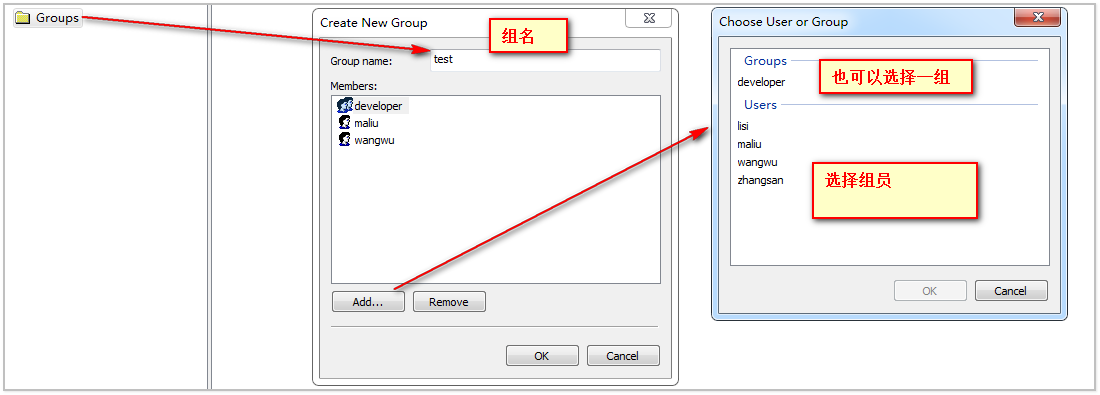
创建组：



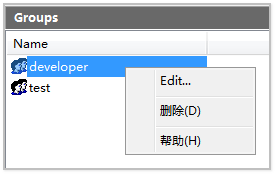
设置组名：



查看创建的组：



修改组：

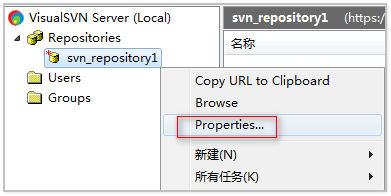


## 分配权限

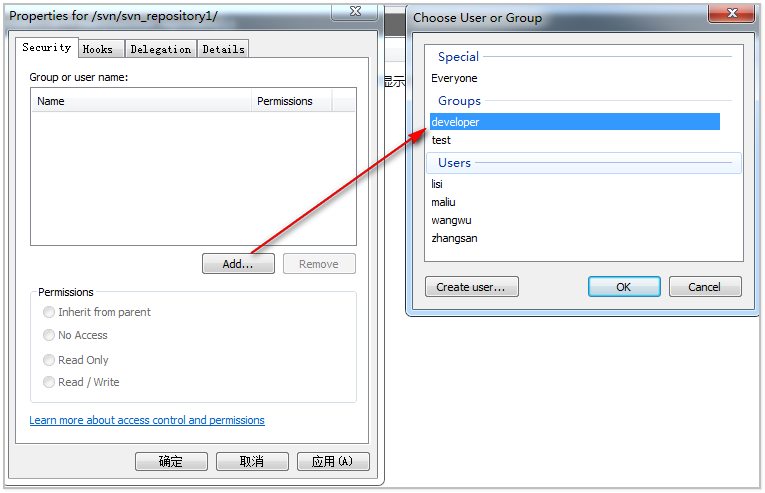
给仓库下的每个目录分配权限对访问进行控制。

### 示例一：开发人员拥有读写权限

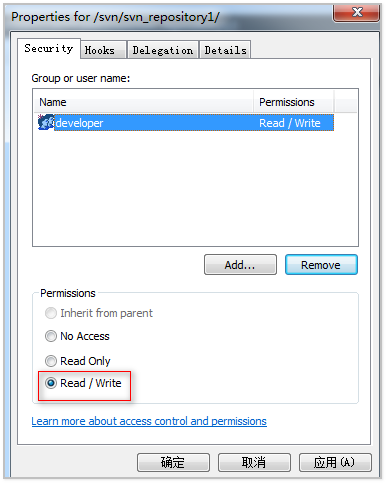
进入权限分配界面：



选择开发组：

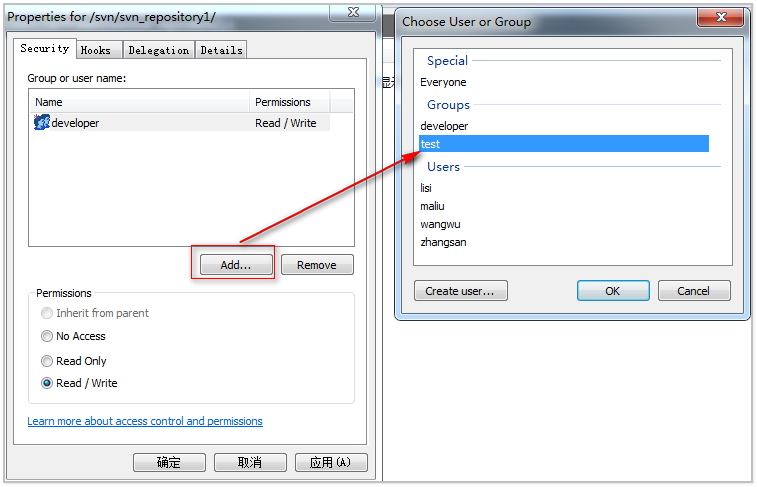


设置开发组权限：

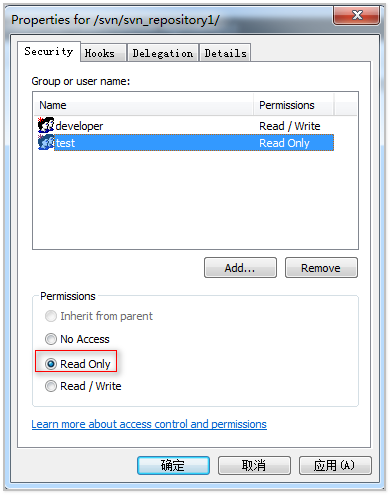


### 示例二：测试人员拥有读权限

选择测试组



设置只读权限：



## TortoiseSVN客户端

### svn客户端类型

svn客户端需要通过网络访问svn服务端提交文件、查询文件等，可通过以下客户端类型访问svn服务端：

1、使用Subversion提供的客户端命令

使用方式：在命令行下输入命令操作。

2、使用Torotise图形化界面操作（推荐）

3、使用Eclipse等开发工具插件操作（推荐）

### 下载安装

TortoiseSVN是Subversion版本控制系统的一个免费开源客户端，不需要为使用它而付费。

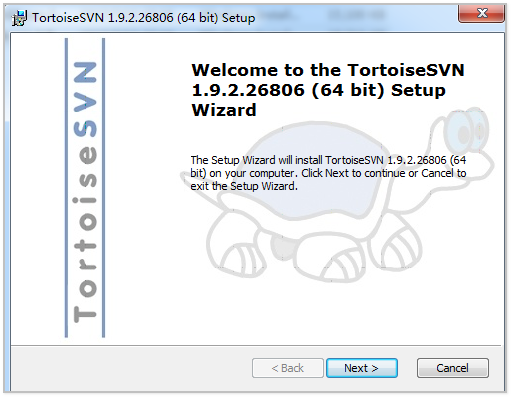
TortoiseSVN是 Subversion 的 Windows 扩展。它使你避免接触 Subversion 枯燥而且不方便的 Command Line。它完全嵌入 Windows Explorer，使用时只需在正常的窗口里右键操作就可以了



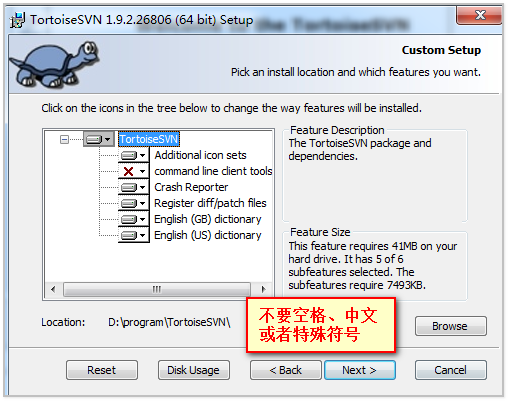
下载：http://tortoisesvn.net/downloads.html 提供 32位和64位不同版本，安装tortoiseSVN 需要修改客户端电脑右键菜单，安装后需要重启电脑。



按照自己的电脑的情况选择32位或者64位进行安装。



选择客户端安装目录：



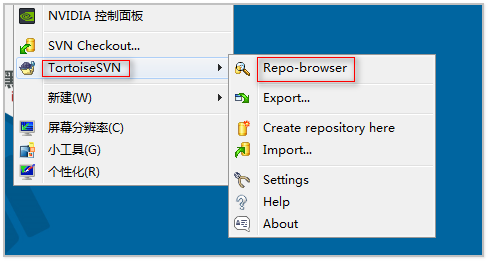
### 浏览仓库

使用Tortoise浏览svn服务端的仓库的内容。

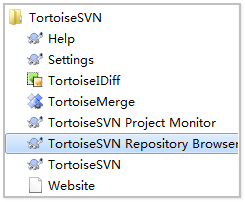
如果右键报错，需要重启电脑，一般是由于之前安装过，二次安装的时候更换了安装路径的原因。

第一步：打开svn客户端

右键鼠标：

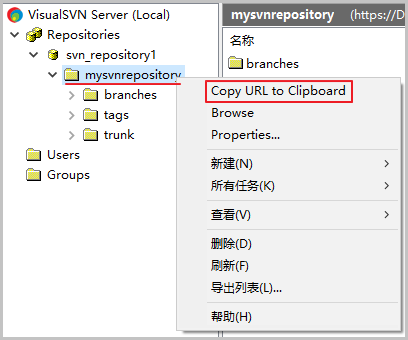


或者在开始菜单中选择客户端：

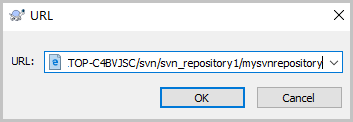


第二步：复制仓库地址

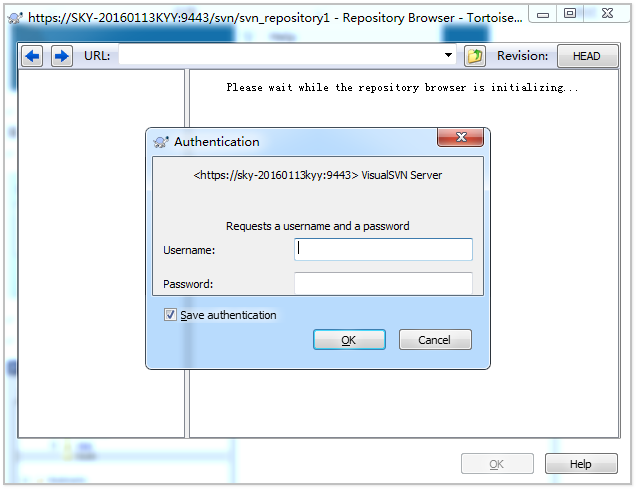
从服务器获取仓库的url



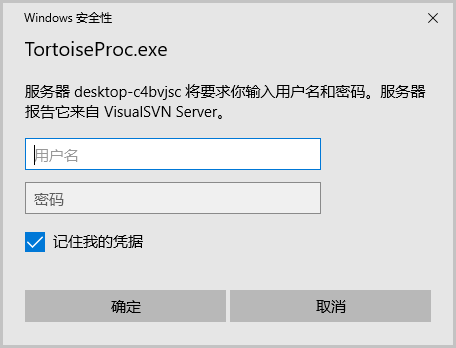
粘贴到客户端：



第三步：输入用户名和密码



注意，如果弹窗如下：请查看用户权限是否设置



### 更换登录帐号

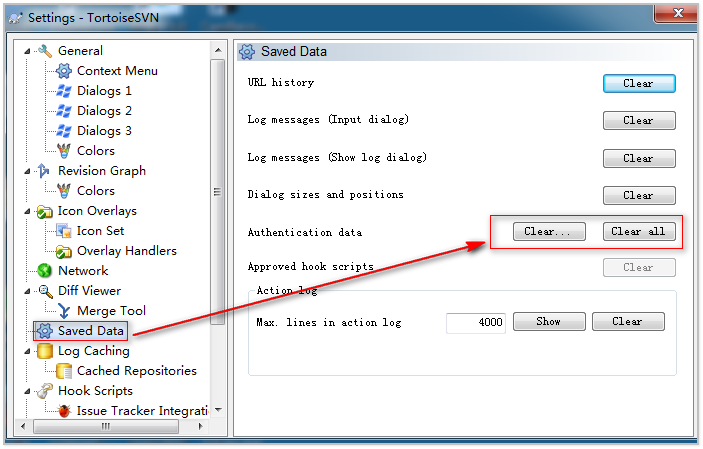
如果之前已经登陆过，会记录之前的帐号和密码并直接登录，而不需要填写用户名和密码，如果要更换帐号，可以将之前的缓存先清除掉。

清除之后可以选择其他帐号进行登录了。

有几种情况需要清除认证缓存：

1、本地使用多个账号登陆，每次输入的账号和密码都不一样

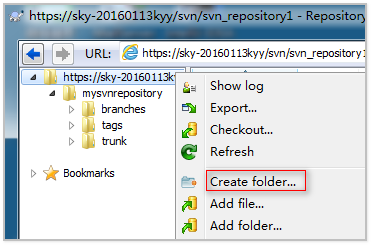
2、当账号密码修改后（建议清理）



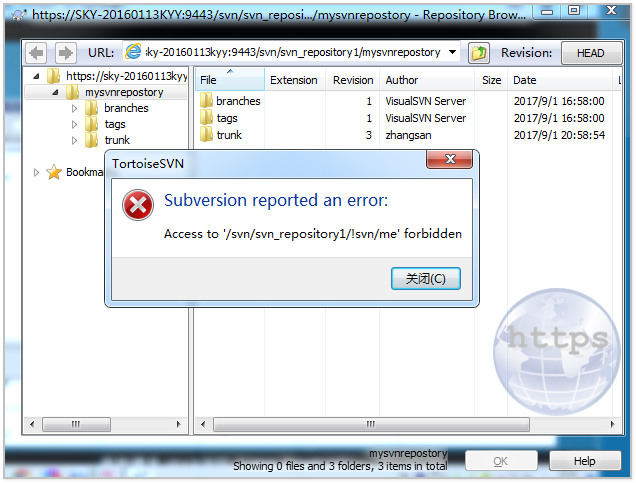
## TortoiseSVN日常使用

### 操作仓库

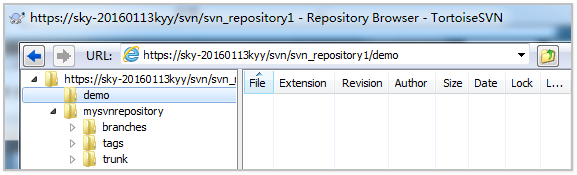
如果登录之后创建文件夹(即创建一个目录)



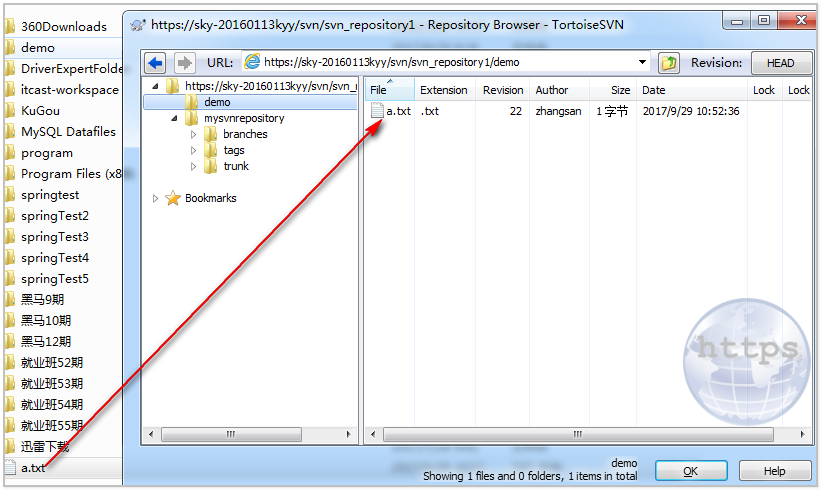
报错如下：可能是帐号没有写的权限导致，更换一个有读写权限的帐号即可。



更换一个有读写权限的帐号即可：



随便在哪个位置创建一个a.txt文件，然后拖入到demo目录中即可：上传成功



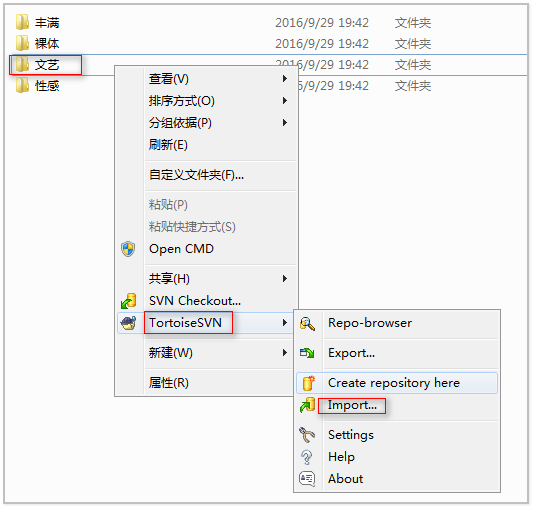
### 导入导出

#### 导入

* import 将本地目录中的所有数据导入到svn 服务器

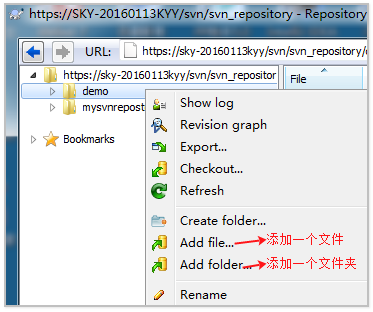
如果想要将一个文件夹中的所有数据上传到服务器，可以如下操作：

选中需要导入的文件夹右键



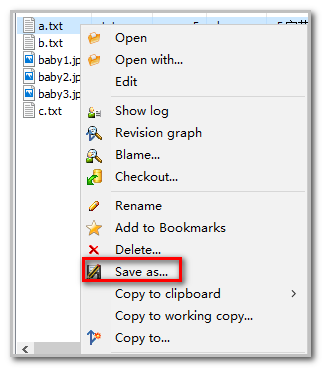
注意：一般这种导入导出的方式用于操作文档，如果要操作代码显然就很麻烦了，因此如果要操作代码，可以使用eclipse的svn插件。

* 客户端添加一个文件或文件夹

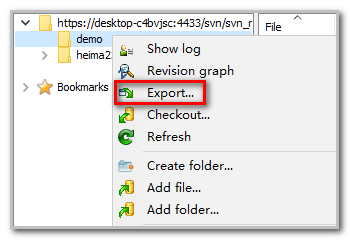


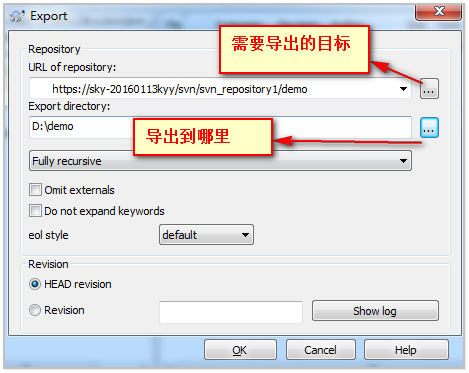
#### 导出和检出

* 导出 ：和checkout区别 （checkout检出后文件，含有.svn隐藏文件夹， 会和SVN仓库交互， export导出，没有.svn隐藏文件夹）
* 导出单个文件：

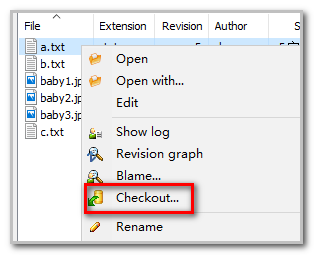


* 导出整个目录：

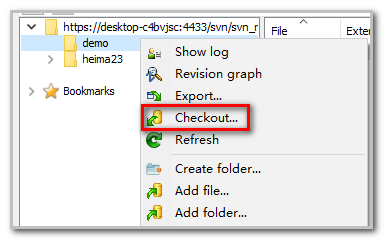




* 检出
* 检出单个文件：



* 检出目录：



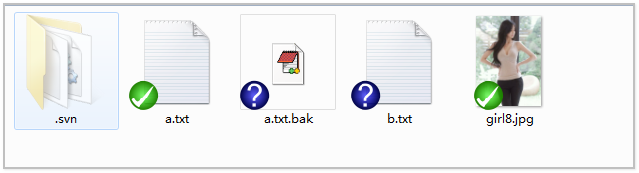
区别如下：检出的文件或目录受svn的版本控制，而导出的文件或目录不受svn版本控制



注意：如果在检出后的目录上传文件，那么就只能上传到检出的目录中。

### ※版本控制图标

如果使用checkout之后发现没有版本控制的图标



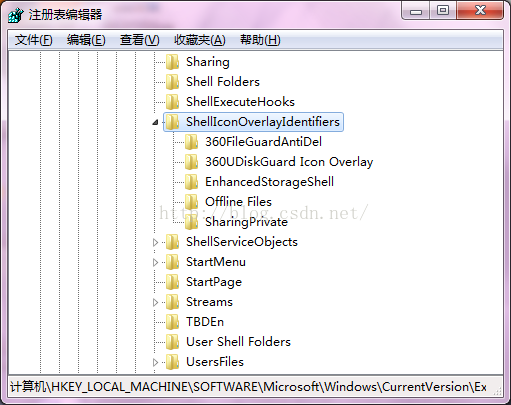
需要重新设置显示版本控制的图标

#### 显示版本控制图标方式一

在运行里输入regedit  进入【注册表】界面：

依次打开HKEY\_LOCAL\_MACHINE->SOFTWARE->Microsoft->Windows->CurrentVersion->Explorer->ShellIconOverlayIdentifiers

发现Tortoise 系列内容不显示，如下图.



**问题原因**：svn的文件状态图标插件禁止启动了

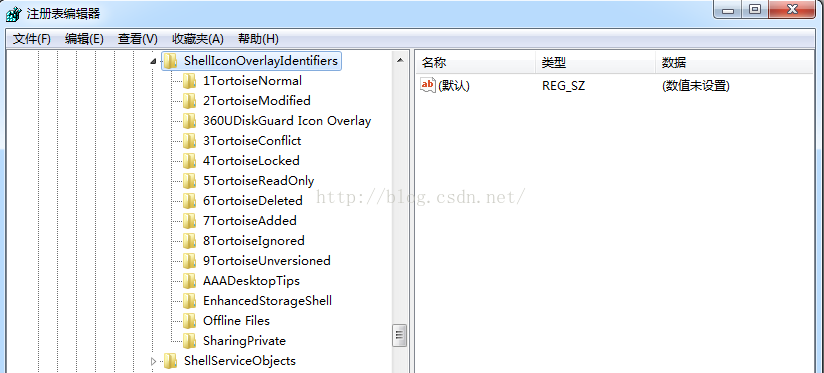


启动项：



**解决方法四**（成功）：恢复svn的文件状态图标插件启动项。

        操作步骤：360安全卫士首页》点击“优化加速”》启动项》自启动插件：将TortoiseSVN的文件状态插件 恢复启动 ，重新驱动计算机，svn图标就显示了。

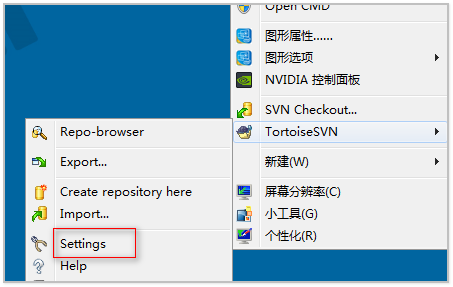


**问题总结**：检查注册表相应节点是否有tortoiseSVN的项目，如果无，应该是svn的文件状态图标插件启动项禁用了，如果有且svn图片不显示了，可能是Tortoise图标的显示顺序排在15位之后。

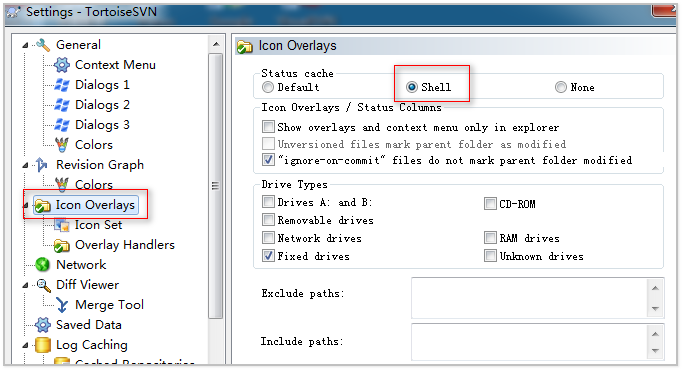
#### 显示版本控制图标方式二

如果用以上方式还是无法显示图标，建议重新修复。

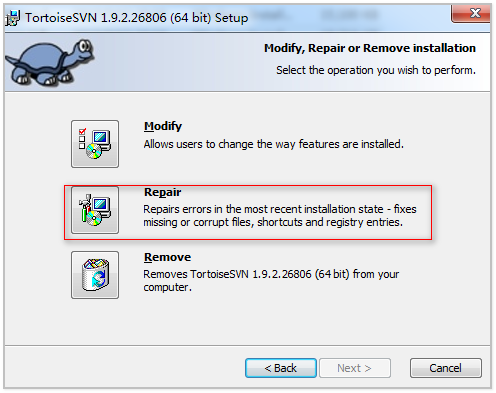
第一步：



第二步：

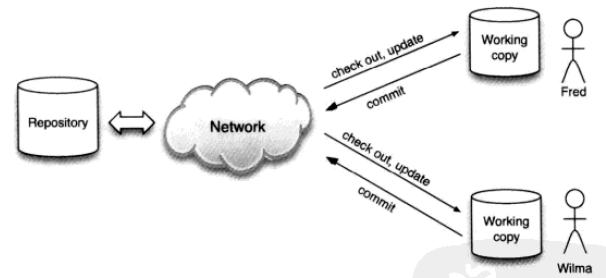


第三步：找到客户端重新修复一下即可。



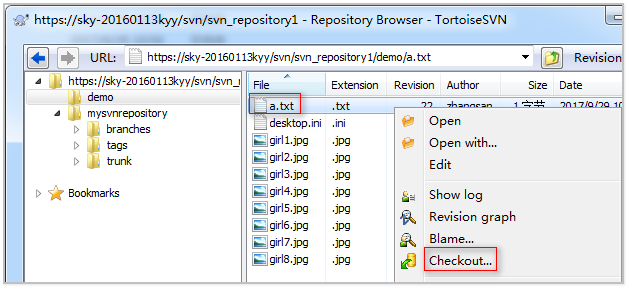
## 修改提交

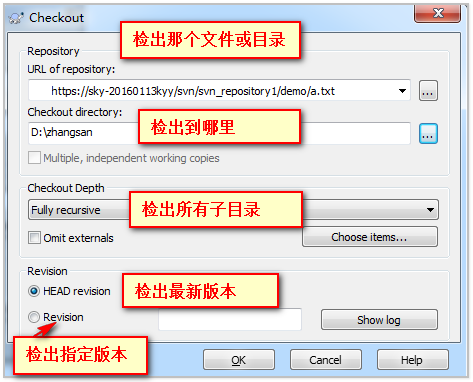
可以分别使用两个帐号来登录客户端去连接服务器，然后分别将服务器上的数据检出到两个文件夹中，用来测试版本控制。



### Checkout

选择要检查的文件或者目录右键：





可以分别使用zhangsan和lisi的帐号检出到对应的目录中：



内容如下：



### add

在lisi的目录中添加文件：

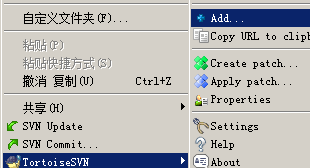
创建一个b.txt



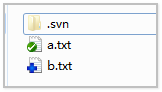
图标： 这是一个新文件

* Add ： 标记这个文件添加到服务器

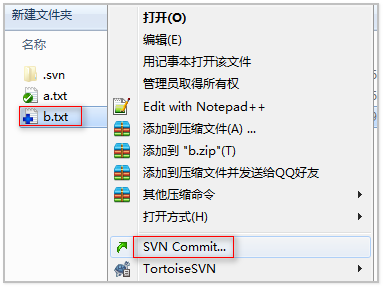
标记完后可以将该文件上传到服务器



刷新如下：

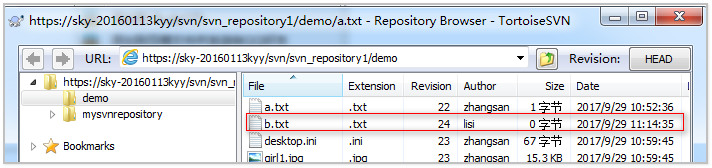


可以将b.txt上传到服务器：



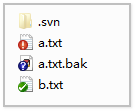
一旦提交之后，无论哪个帐号登录，都可以看到仓库中多了一个b.txt:

其中可以看到谁上传以及上传的时间等信息



### Commit

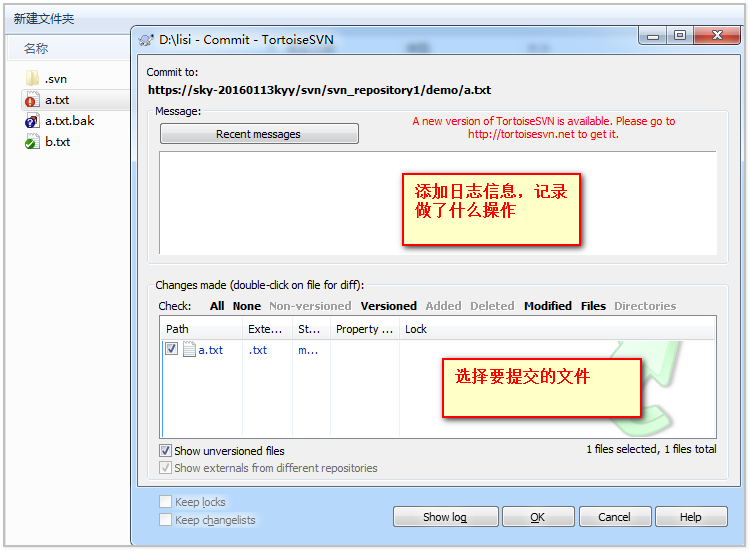
在lisi的目录中修改a.txt的内容后，由于本地文件和服务器中的数据不同，显示标记如下：

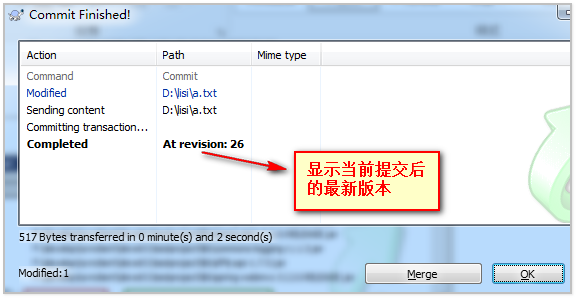


右键提交Commit 提交本地修改至svn服务器：

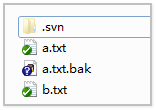
在检出目录或要提交修改的目录右键：





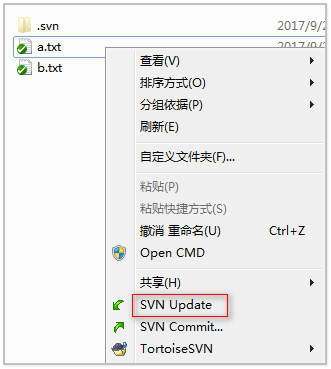


提交后目录中的内容与svn服务同步，目录图标变为：



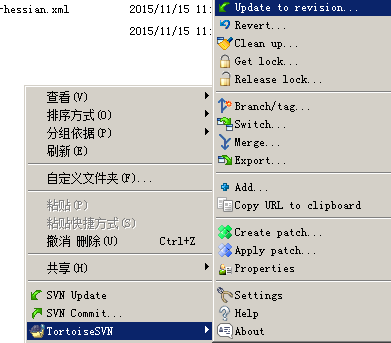
### update

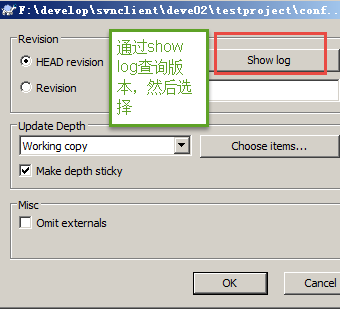
进入zhangsan的目录，在检出目录或子目标或文件上右键：



会将lisi更新的最新的数据下载到本地。

### 更新到指定版本

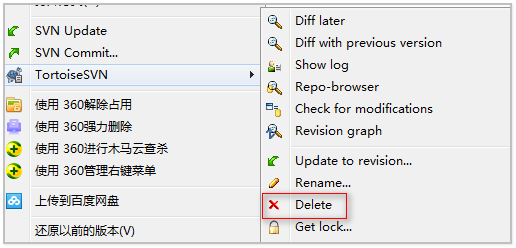




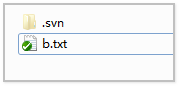
### Delete

Delete ：删除版本库文件

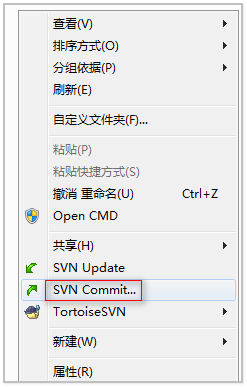
右键a.txt,选择Delete



发现本地文件被删除，没有a.txt了



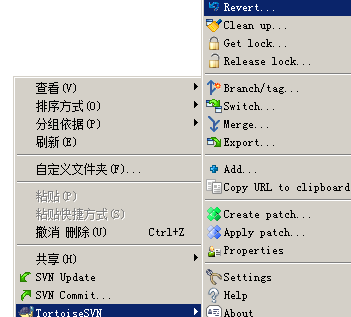
此时如果选择提交，那么会将服务器中的a.txt也删除掉。如果还没有commit，那么还可以恢复，见1.8.7恢复

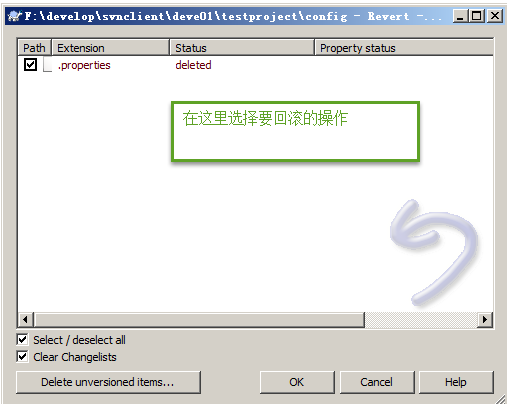


### ※恢复

在检出目录或子目录操作会记录操作日志，提交前可以回滚操作。

在要回滚的检出目录或子目录中右键：





## 冲突处理

两个客户端同时修改同一个文件， 改动同一个位置，发生冲突情况

如果当commit 遇到文件已经过时，说明另一个人可能改动过

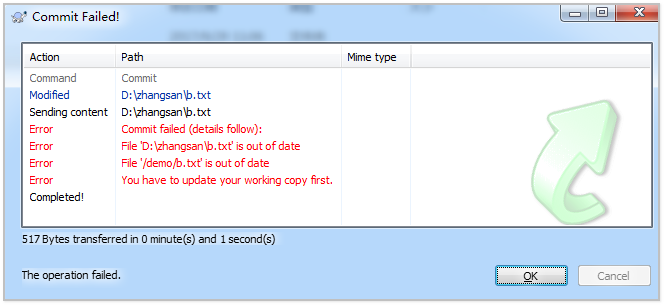
### 制造冲突

步骤一：进入lisi的目录

修改b.txt的内容后commit

步骤二：进入zhangsan的目录

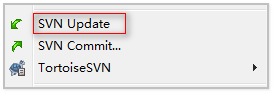
修改b.txt的内容后commit,报错如下：



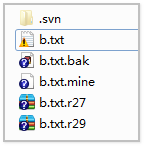
### 解决冲突

步骤一：查看哪里冲突

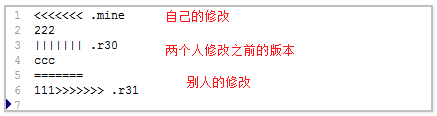
右键b.txt,选择update:



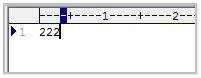
更新后如下：



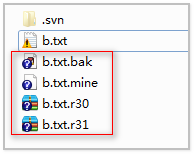
打开b.txt内容显示冲突如下：



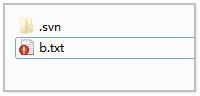
步骤二：将b.txt的内容重新修改成自己想要的内容



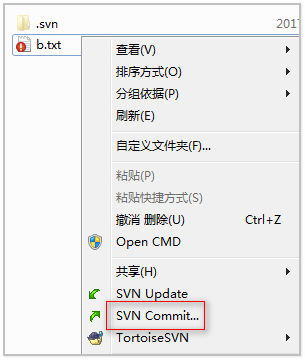
步骤三：删除b.txt以外的文件，见红框



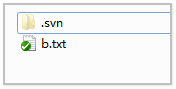
删除后，b.txt标记修改如下：



步骤四：右键提交即可



提交之后刷新：服务端的数据也会随之更新。



这样就把别人的修改也给覆盖掉了。

这种情况产生的原因是，双方都是直接进行修改然后进行commit提交导致的问题。因此我们在编写代码之前都应该先从服务器将最新的代码更新（update）到本地之后再去进行修改。

## eclipse的SVN插件使用

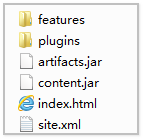
### svn插件安装

#### 简单方式

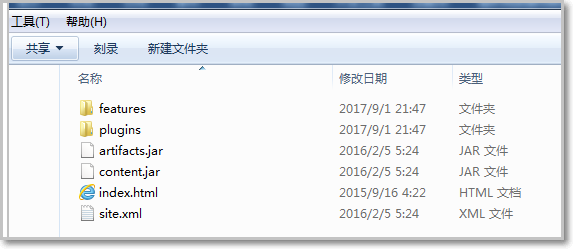
解压插件



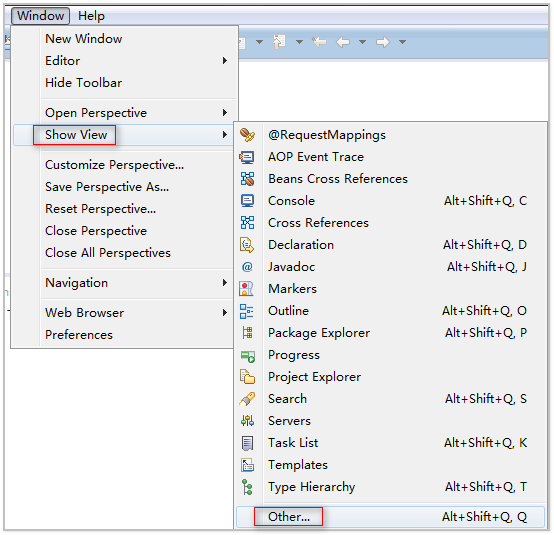
得到如下：



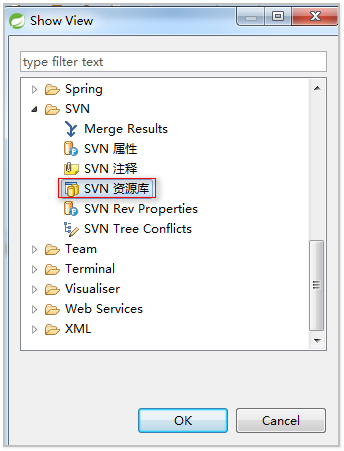
将上面解压出来的所有文件复制到eclipse下的dropins目录下：



安装完成，查看STS视图有SVN选项说明安装成功：



如果出现下图证明安装成功



#### 复杂方式

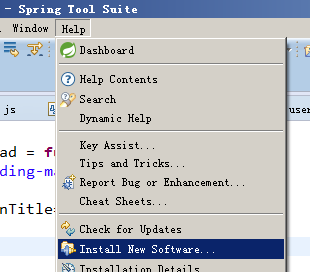
下载Subversion的eclipse插件

<http://subclipse.tigris.org/servlets/ProjectProcess?pageID=p4wYuA>

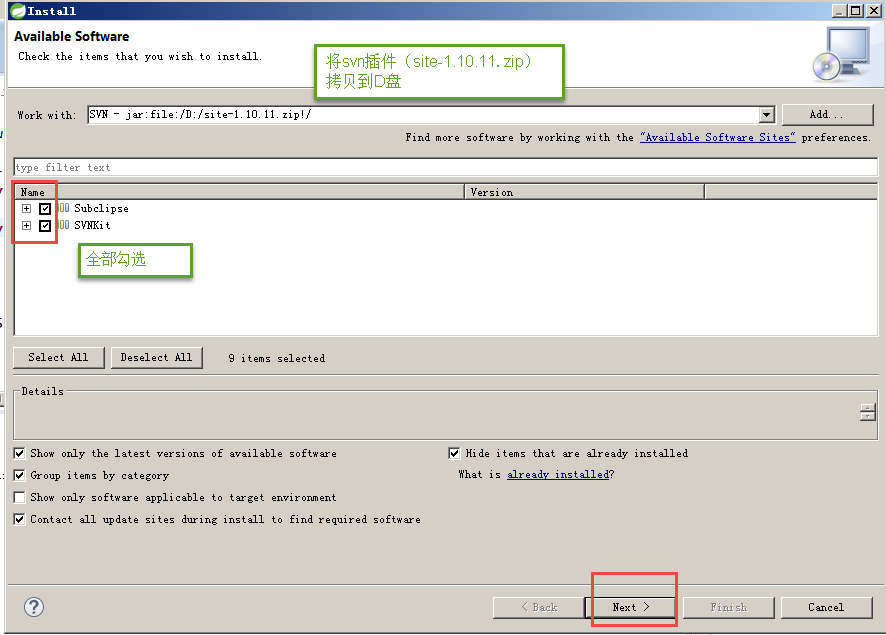
将site-1.10.11.zip放在D盘。

下边是在STS上安装svn插件步骤：

1、进入STS软件安装界面



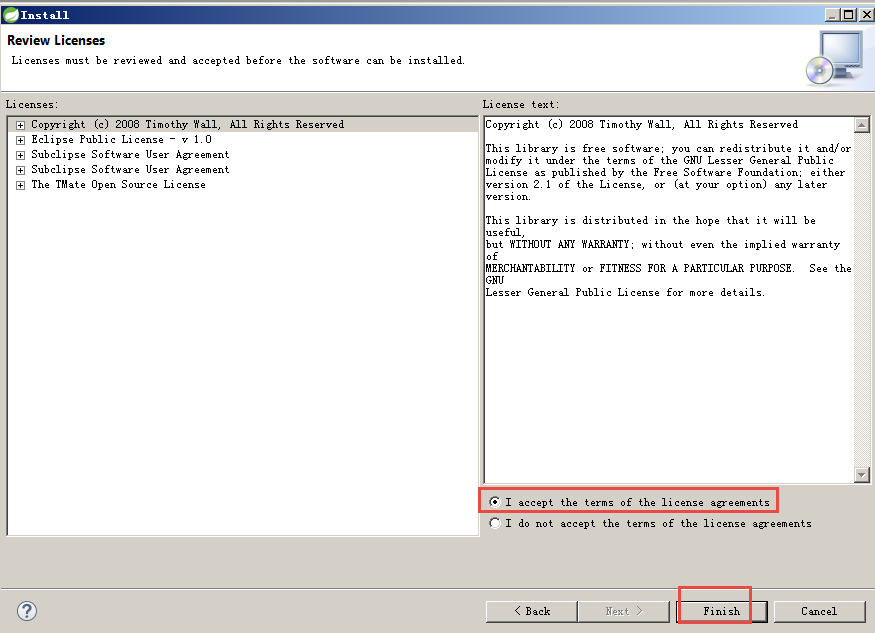
2、安装svn插件



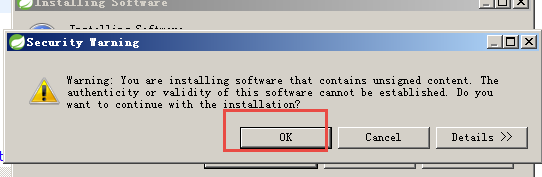
上边命令行输入：SVN - jar:file:/d:/site-1.10.11.zip!/

点击下一步。

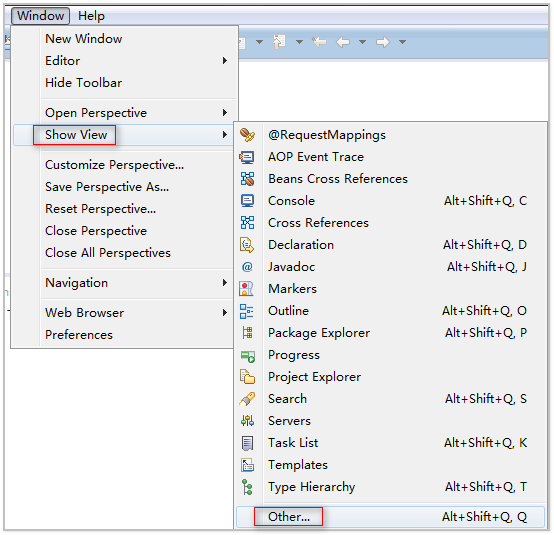
接受协议，完成：



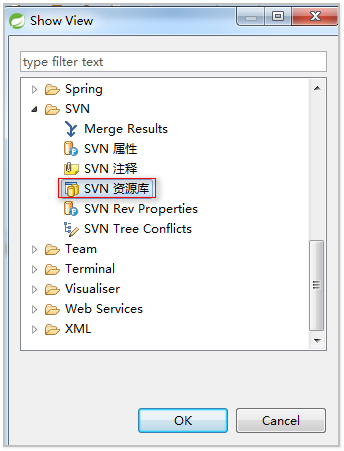
出现提示，选择继续安装：



安装完成，查看STS视图有SVN选项说明安装成功：



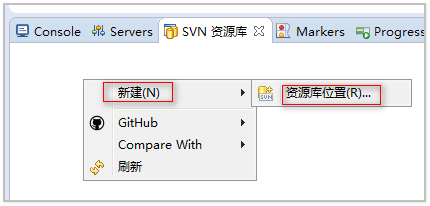
如果出现下图证明安装成功



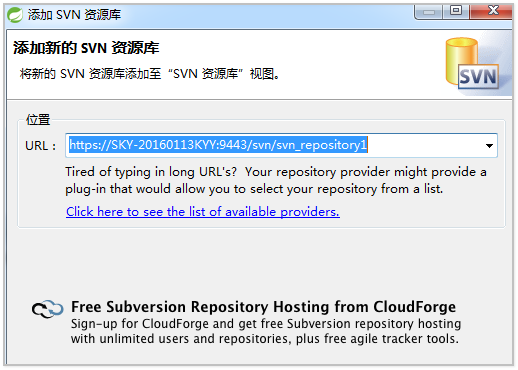
### 将项目共享到SVN

#### 步骤一：调出svn视图

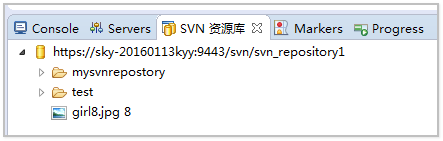
点击SVN资源库，调出SVN视图：



选择资源库的url：



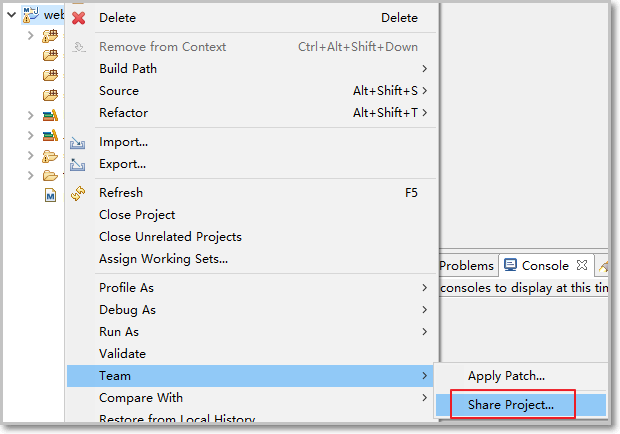
完成后如下：



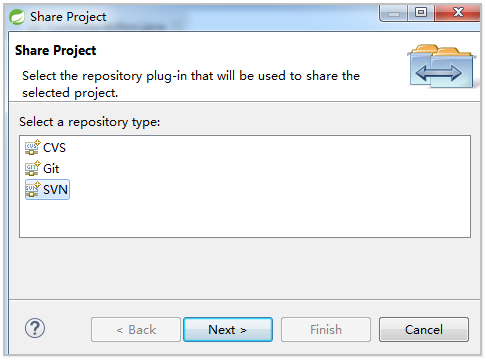
注意:如果之前用客户端连接过服务器并记住了密码，那么此时就不需要输入用户名和密码就能登陆。

#### 步骤二：将项目共享到SVN

将自己的项目上传到SVN中



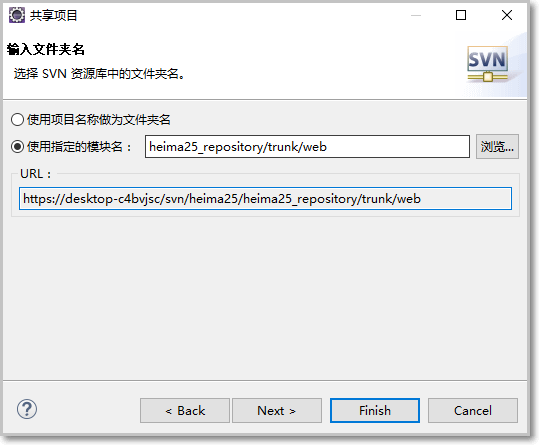
选择SVN:



选择共享的资源库地址：



选择项目上传到哪个目录：

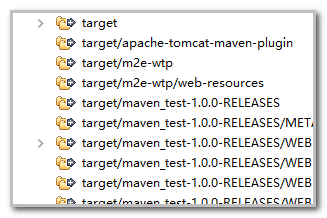


提交代码：

在将maven项目上传到svn服务器的时候，有几类文件是不需要上传的：

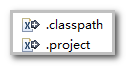
1. 所有含target的文件
2. 环境文件：.classpath、.settings、project

target文件:



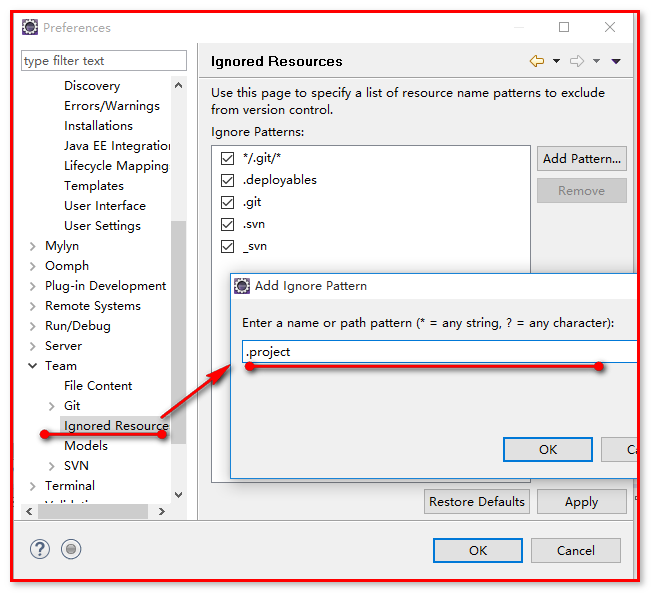
环境文件：





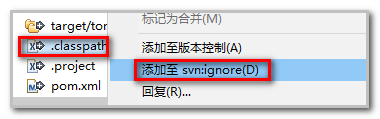
##### 全局指定（用的多）

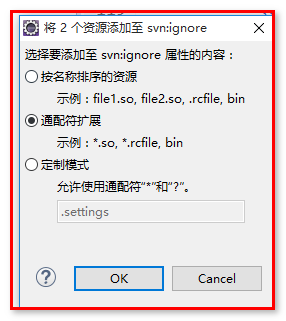
对所有的项目有效window-->属性



##### 单个指定

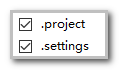
只对当前项目有效





* \* 匹配任何字符串，包括空串(没有字符)
* ? 匹配任何单字符
* [...] 匹配任何单在方括号[]内的单字符，在方括号内，一对字符被“-”分隔，匹配任何词汇表(lexically)上在他们中间的字符。例如[AGm-p]匹配任何单个的A,G,m,n,o或者p。

这里我们选择使用全局忽略指定资源：

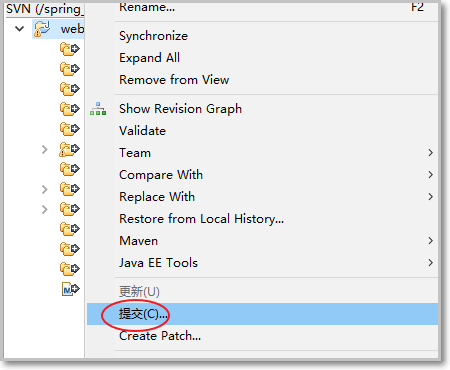






#### 步骤三：提交

忽略文件完成后，提交代码：

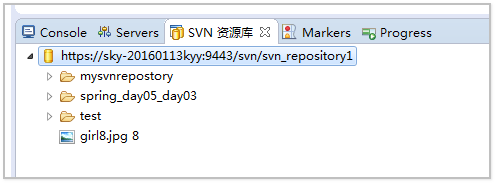


上传成功：此时其他人就可以从svn上检出你上传的项目继续进行其他功能的开发。

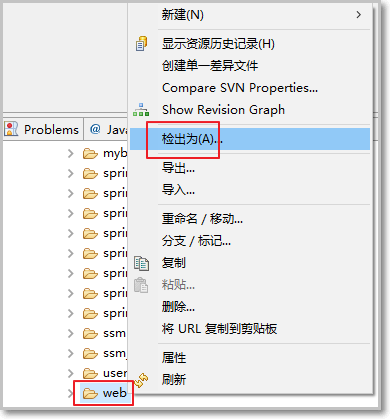
### 从svn检出

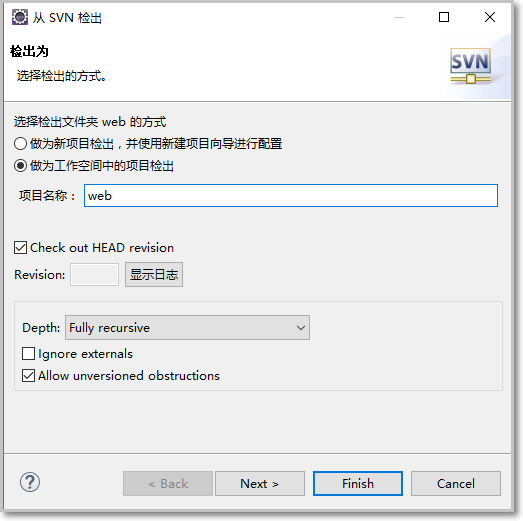
再打开一个eclipse(简称ec02),可在将eclipse的工作空间换一个，然后重新配置。

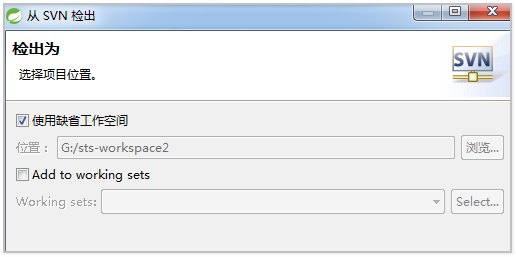
#### 步骤一：重新搭建svn资源库



#### 步骤二：检出工程





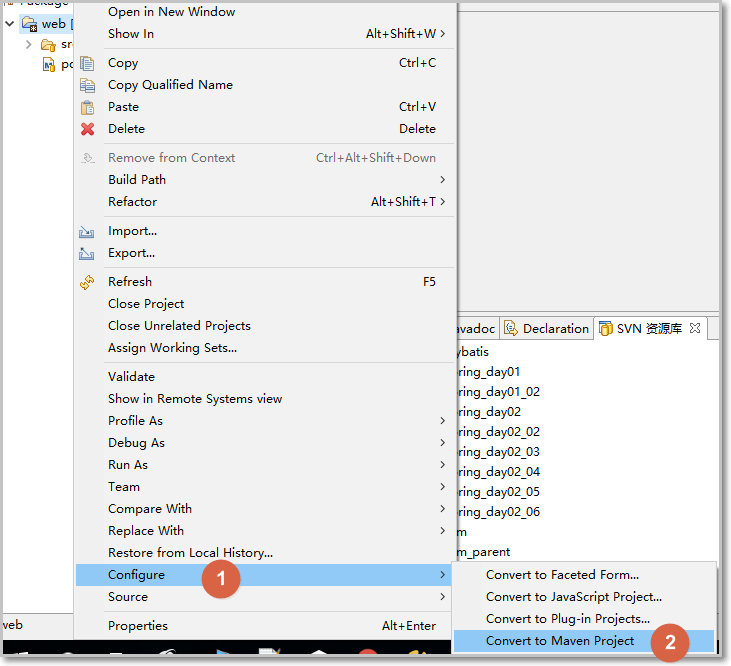


#### 步骤三：转换maven工程

检出后工程如下:



需要手动转换成maven工程:



#### 步骤四：修改代码并提交

在代码中添加一行注释：

**public** **class** Demo {

**public** **static** **void** main(String[] args) {

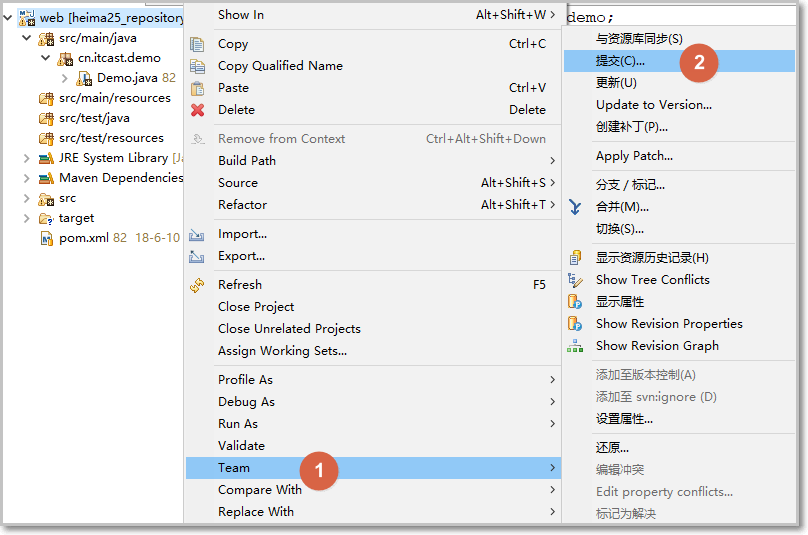
//初始化user对象

User user = **new** User();

}

}

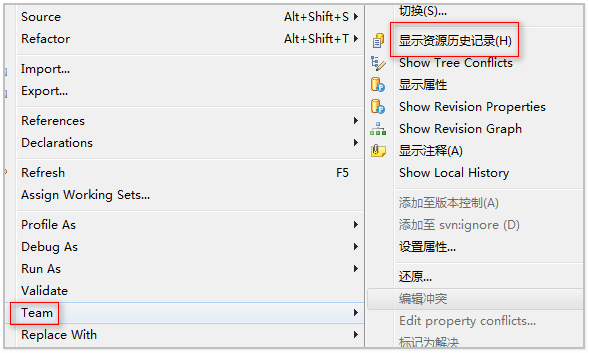
提交代码:



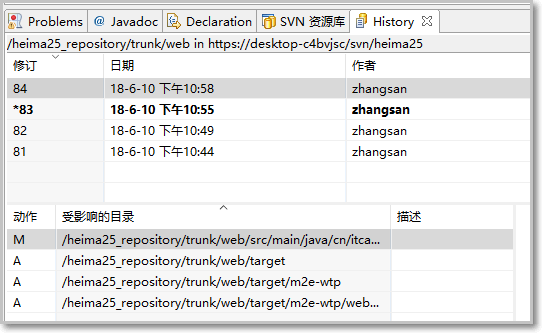
注意：一般在公司开发时，到公司第一件是先更新代码，将svn上的代码更新到本地然后再进行开发，避免到时候产生的各种冲突问题。

#### 查看历史记录

如果想要查看历史更新记录，如下操作：



查询历史记录：



## 提交冲突

解决冲突2

此时ec01和ec02在修改同一个方法。

注意：此种解决方式适用于两人修改了同一个文件相同的位置

### 解决版本冲突方式一

##### 第一步：ec01修改方法名称

ec01将方法名称进行了修改，如下：改成saveUser

/\*\*

\* 测试

\* **@author** zhang

\*

\*/

**public** **class** Demo {

**public** **static** **void** main(String[] args) {

//初始化user对象

User user = **new** User();

}

**public** **void** saveUser(){};

}

然后进行了提交。

##### 第二步：ec02也修改方法名称

ec01也在对该方法进行修改，觉得之前的方法名称不好,改一下，改成updateUICustomer:

/\*\*

\* 测试

\* **@author** zhang

\*

\*/

**public** **class** Demo {

**public** **static** **void** main(String[] args) {

//初始化user对象

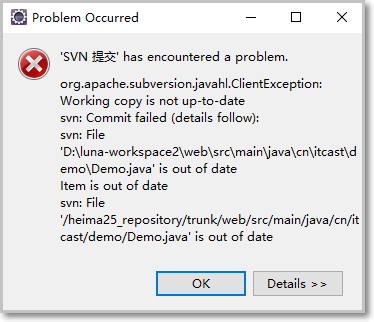
User user = **new** User();

}

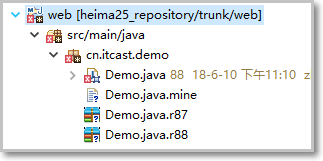
**public** **void** updateUser(){};

}

然后一提交，又报错了：



然后ec02选择了更新：发现如下



/\*\*

\* 测试

\* **@author** zhang

\*

\*/

**public** **class** Demo {

**public** **static** **void** main(String[] args) {

//初始化user对象

User user = **new** User();

}

<<<<<<< .mine

**public** **void** updateUser(){};

=======

**public** **void** saveUser(){};

>>>>>>> .r88

}

分析：svn服务器对每一个都有一个版本号，当提交之后版本号就会更新，比如ec01拿到的版本号是1，然后提交之后版本号就变成2了，然后ec02之前也是拿到的版本1，在进行提交的时候会先和之前的版本进行比较，发现提交之后也要变成2了，版本号不能重复，因此提交不成功。

Ec01和ec02协商后达成共识，改成某一个方法：

/\*\*

\* 测试

\* **@author** zhang

\*

\*/

**public** **class** Demo {

**public** **static** **void** main(String[] args) {

//初始化user对象

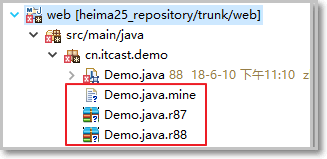
User user = **new** User();

}

**public** **void** updateUser(){};

}

删除比较的版本

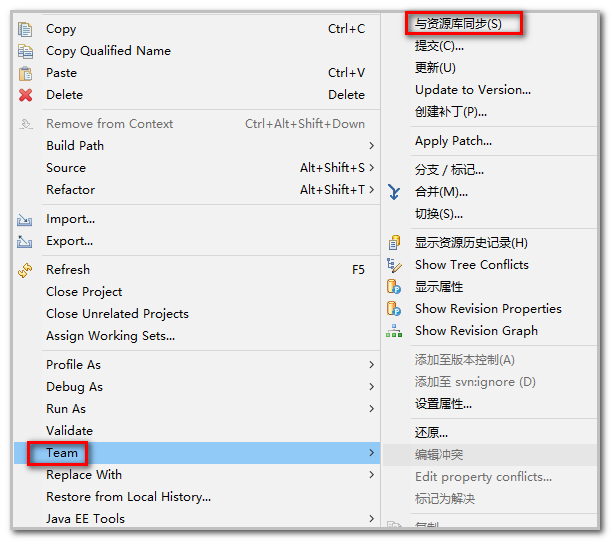


此时提交无误，不会再报错了。

### 解决版本冲突方式二

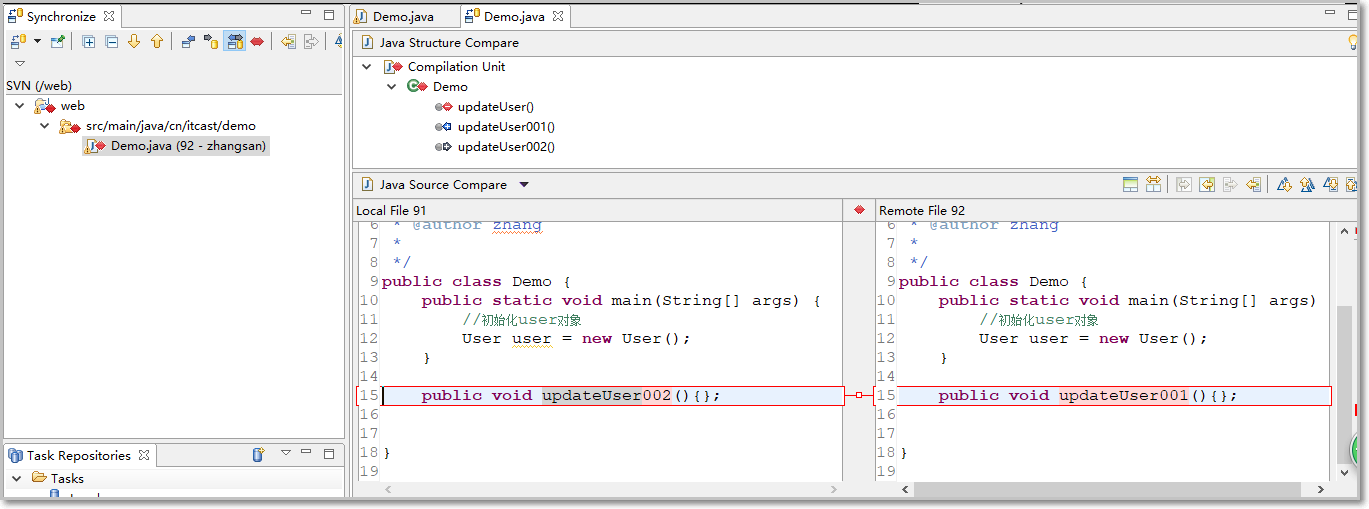
##### 第一步：与资源库同步

右键工程-Team-与资源库同步



##### 第二步：检查异常问题

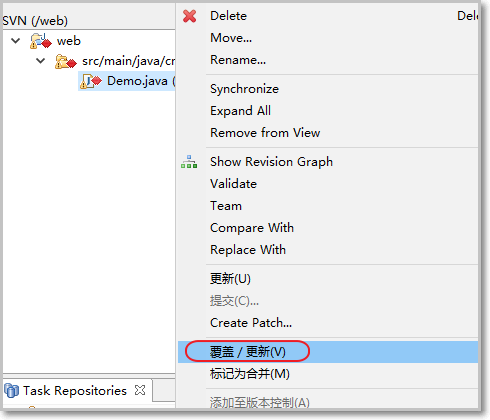
通过同步，查看代码有差异的地方：

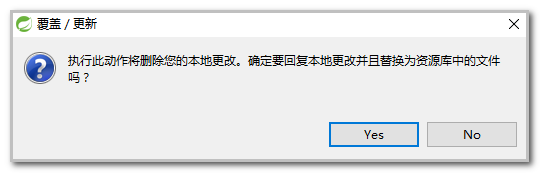


##### 第三步：解决异常

* 方式一：覆盖/更新

将服务器上的代码覆盖到本地



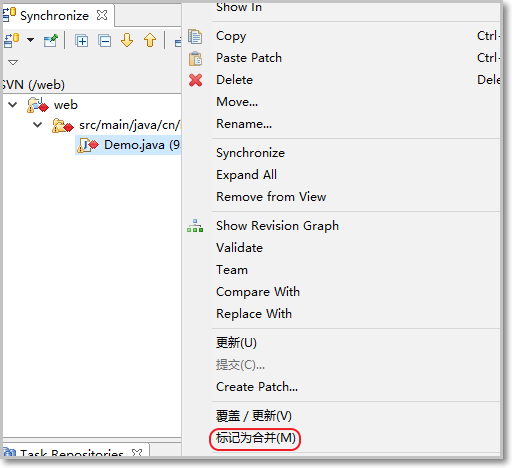


注意：当执行Yes后，会使用服务器上的代码覆盖自己本地的代码。

* 方式二：标记为合并

将自己的代码更新到服务器

这样会将其他用户提交的代码给覆盖掉



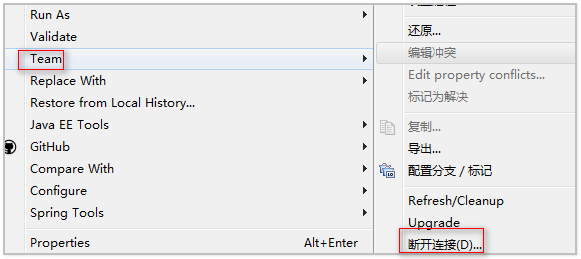
在该界面再次右键工程,进行提交：将本地代码更新到服务器，

注意：此操作会覆盖其他用户的更新。

## ※删除项目

如果有的时候感觉自己的项目已经被改的面目全非了，想要删除后重新检出，然后重新编写。

右键工程：



选择：第一个选项



两个选项的区别如下：

* 并且从文件系统中删除SVN元信息

不能重新连接到svn

* 不删除SVN元信息(例如：SVN子目录):

可以重新连接到svn

那就和svn彻底断开了连接关系。