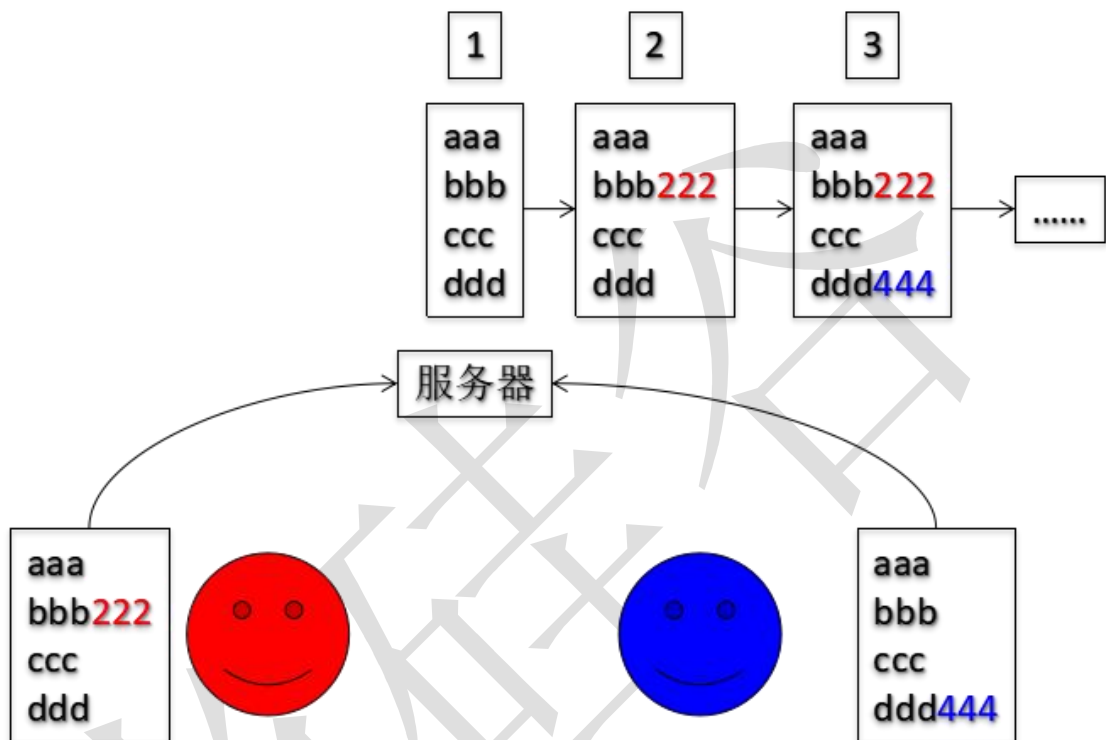


SVN 笔记

1 从个人开发到团队协作



2 版本控制工具的功能

- 协同修改
 - 多人并行不悖的修改服务器端的同一个文件。
- 数据备份
 - 如果本地文件发生丢失可以服务器端文件进行恢复。
- 增量式的版本管理
 - 服务器端保存每一个版本信息时只保存有修改的局部内容, 节约服务器端资源。
- 权限控制
 - 对团队中参与开发的人员进行权限控制。
- 历史记录
 - 查看修改人、修改时间、修改内容、日志信息。
 - 将本地文件恢复到某一个历史状态。

3 版本控制简介

3.1 版本控制

工程设计领域中使用版本控制管理工程蓝图的设计过程。在 IT 开发过程中也可以使用版本控制思想管理代码的版本迭代。

3.2 版本控制工具

思想：版本控制

实现：版本控制工具

集中式版本控制工具：

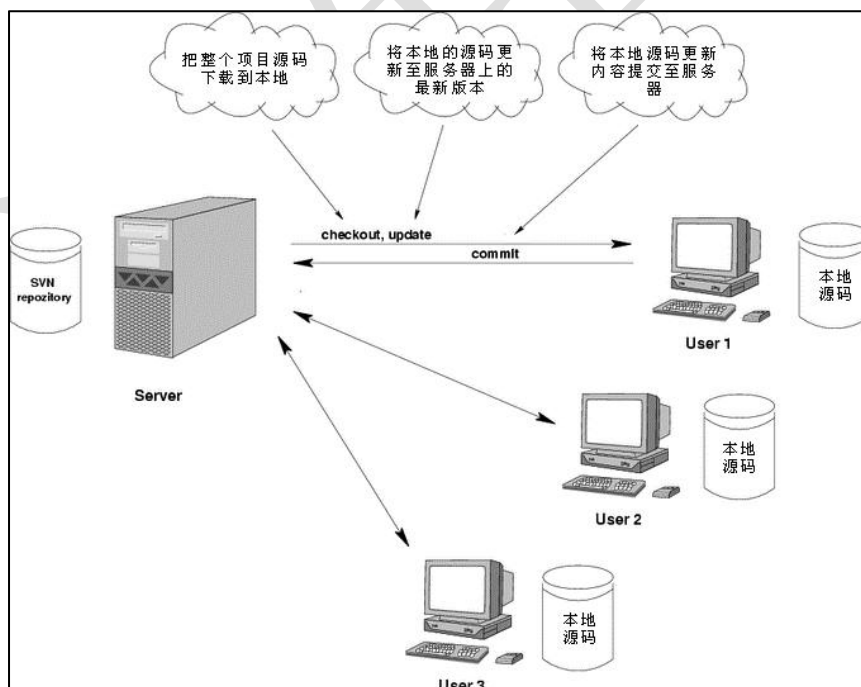
CVS、SVN、VSS……

分布式版本控制工具：

Git

4 SVN 的工作机制

4.1 C/S 结构



4.2 基本操作

➤ 检出 (Checkout)

■ 把服务器端版本库内容完整下载到本地。

- 在整个开发过程中只做一次。
- 更新（Update）
 - 把服务器端相对于本地的新的修改下载到本地。
- 提交（Commit）
 - 把本地修改上传到服务器。

5 服务器端环境搭建步骤

5.1 安装服务器端程序

```
yum install -y subversion
```

验证
<pre>[root@rich ~]# svn --version svn, 版本 1.6.11 (r934486) 编译于 Aug 17 2015, 08:37:43 版权所有 (C) 2000-2009 CollabNet。 Subversion 是开放源代码软件，请参阅 http://subversion.tigris.org/ 站点。 此产品包含由 CollabNet(http://www.Collab.Net/) 开发的软件。 可使用以下的版本库访问模块: * ra_neon: 通过 WebDAV 协议使用 neon 访问版本库的模块。 - 处理“http”方案 - 处理“https”方案 * 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</pre>

5.2 创建并配置版本库

- 创建版本库目录

```
mkdir -p /var/svn/repository
```
- 在版本库目录下创建具体项目目录

```
mkdir pro_oa
```
- 创建 SVN 版本库

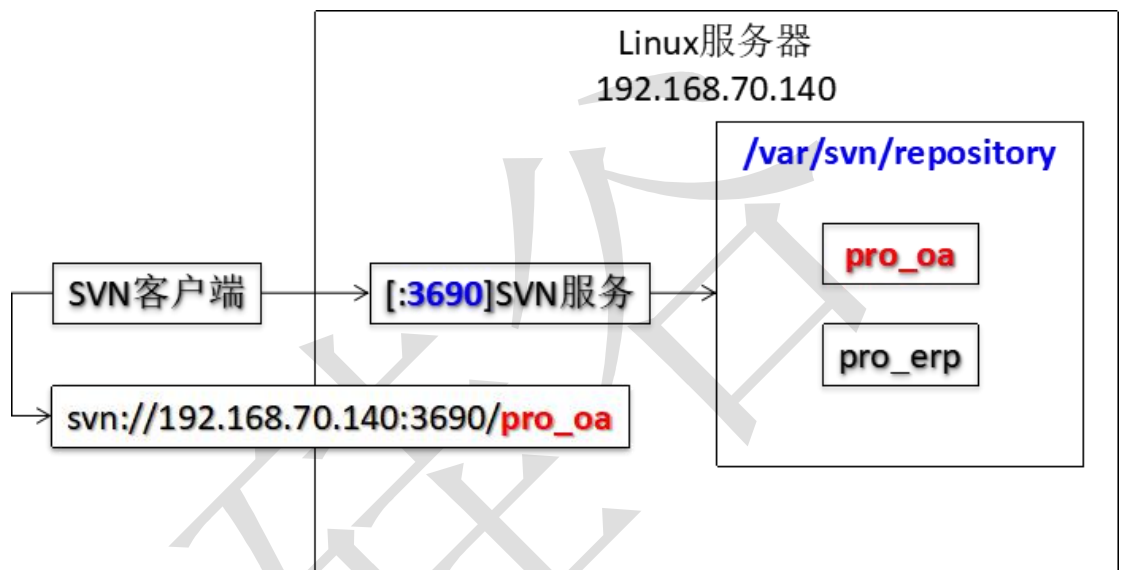
```
svnadmin create /var/svn/repository/pro_oa
```
- 版本库内容

```
[root@rich pro_oa]# ll
总用量 24
drwxr-xr-x. 2 root root 4096 3月 30 17:17 conf
drwxr-sr-x. 6 root root 4096 3月 30 17:17 db
-r--r--r--. 1 root root 2 3月 30 17:17 format
drwxr-xr-x. 2 root root 4096 3月 30 17:17 hooks
drwxr-xr-x. 2 root root 4096 3月 30 17:17 locks
-rw-r--r--. 1 root root 229 3月 30 17:17 README.txt
```

版本库配置文件
数据库目录
钩子程序

5.3 配置 SVN 对应的服务

➤ 思路



➤ SVN 服务

- 名称: svnserve
- 默认情况下不是开机自动启动

```
[root@rich repository]# chkconfig|grep svn
svnserve 0:关闭 1:关闭 2:关闭 3:关闭 4:关闭 5:关闭 6:关闭
[root@rich repository]# chkconfig svnserve on
[root@rich repository]# chkconfig|grep svn
svnserve 0:关闭 1:关闭 2:启用 3:启用 4:启用 5:启用 6:关闭
```

- 对应可执行脚本文件路径

```
/etc/rc.d/init.d/svnserve
```

注意备份!

➤ SVN 服务具体配置

原版
args="--daemon --pid-file=\${pidfile} \$OPTIONS"
修改版
args="--daemon --root 版本库根目录 --listen-port 指定端口号 --pid-file=\${pidfile} \$OPTIONS"
args="--daemon --root /var/svn/repository --listen-port 2255 --pid-file=\${pidfile} \$OPTIONS"

使用默认端口号的配置方式
args="--daemon --root /var/svn/repository --pid-file=\${pidfile} \$OPTIONS"

4

更多 Java - 大数据 - 前端 - python 人工智能资料下载, 可访问百度: 尚硅谷官网

5.4 启动 SVN 服务

```
[root@rich init.d]# service svnserve start
正在启动 svnserve: [确定]
[root@rich init.d]# service svnserve status
svnserve (pid 3443) 正在运行...
[root@rich init.d]# netstat -anp | grep :2255
tcp        0      0 0.0.0.0:2255          0.0.0.0:*            LISTEN
3443/svnserve
[root@rich init.d]# ps -ef | grep svnserve | grep -v grep
root      3443      1   0 11:41 ?                00:00:00 /usr/bin/svnserve --daemon --root
/var/svn/repository --listen-port 2255 --pid-file=/var/run/svnserve.pid
[root@rich init.d]# cat /var/run/svnserve.pid
3443
```

6 命令行客户端

6.1 创建两个工作区目录模拟两个开发人员

```
mkdir -p /root/workspace/harry
mkdir -p /root/workspace/sally
```

6.2 检出

- 作用：完整下载版本库中的全部内容。
- 命令：
 - `svn checkout svn://192.168.70.140/pro_oa ./`
- 附加效果
 - 在指定目录下创建.svn 目录
 - 保存本地目录和文件状态信息，用来和 SVN 服务器进行交互
- 工作副本
 - .svn 所在的目录
 - 版本控制相关操作都需要在工作副本目录下执行。例如：提交、更新等等这样的操作。
 - 为了保证工作副本能够正常和服务器进行交互，请不要删除或修改.svn 目录中的内容。

6.3 添加

- SVN 要求提交一个新建的文件前先把这个文件添加到版本控制体系中。
- `svn add 文件名`

6.4 提交

- 要求 1: 附加日志信息
 - 日志信息相当于写 Java 代码时的注释, 用来标记本次操作所做的修改。
 - `svn commit -m "xxx" [文件名]`
- 要求 2: 必须具备相应的权限
 - 使用文本编辑器打开版本库根目录/conf/svnserve.conf 文件

```
8 [general]
9 ### These options cont
10 ### and authenticated
11 ### and "none". The s
12 anon-access = write
13 # auth-access = write
```

把匿名访问配置项的注释打开。注意: 行的开头不能有空格

```
[root@rich harry]# svn commit hello.txt
svn: 提交失败(细节如下):
svn: "/root/workspace/harry/hello.txt" 尚未纳入版本控制
[root@rich harry]# svn add hello.txt
A      hello.txt
[root@rich harry]# svn commit hello.txt
svn: 提交失败(细节如下):
svn: 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: 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
[root@rich harry]# svn commit -m "My first commit" hello.txt
svn: 提交失败(细节如下):
svn: 认证失败
[root@rich harry]# svn commit -m "My first commit" hello.txt
增加      hello.txt
传输文件数据.
提交后的版本为 1。
```

6.5 查看服务器端文件内容

```
[root@rich harry]# svn list svn://192.168.70.140/pro_oa
good.log
hello.txt
```

6.6 更新操作

- 作用: 把服务器端文件所产生的所有修改下载到本地
- 命令: `svn update [文件名]`

7 冲突

7.1 过时的文件

- 概念：在一个相对服务器端版本来说是旧版本的基础上进行了修改的文件。
- 要求：所有过时的文件都必须先执行更新操作，更新后在最新版基础上修改的文件才允许提交。

7.2 冲突的产生

- 条件 1：本地当前编辑的文件已经过时。
- 条件 2：从服务器端更新下来的修改和本地的修改在“同文件同位置”不一致。

7.3 冲突的表现

- 文件内

发生冲突时我们本地自己的内容

```
[root@rich pro_oa]# cat hello.txt
aaaaaaaa
bbbbbbbbb2222222
<<<<<<< .mine
cccccccc% % % %
=====
cccccccc3333333
>>>>>>> .r6
ddddddddd444444444
```

发生冲突时服务器端内容

- 目录内

由于发生了冲突而产生的三个文件

```
good.log
hello.txt
hello.txt.mine
hello.txt.r5
hello.txt.r6
```

```
[root@rich pro_oa]# cat hello.txt.mine
aaaaaaaa
bbbbbbbbb2222222
cccccccc% % % %
ddddddddd444444444
[root@rich pro_oa]# cat hello.txt.r5
aaaaaaaa
bbbbbbbbb2222222
cccccccc
ddddddddd444444444
[root@rich pro_oa]# cat hello.txt.r6
aaaaaaaa
bbbbbbbbb2222222
cccccccc3333333
ddddddddd444444444
```

xxx.mine 文件：发生冲突时本地文件内容

xxx.r[小版本号]文件：发生冲突前文件内容

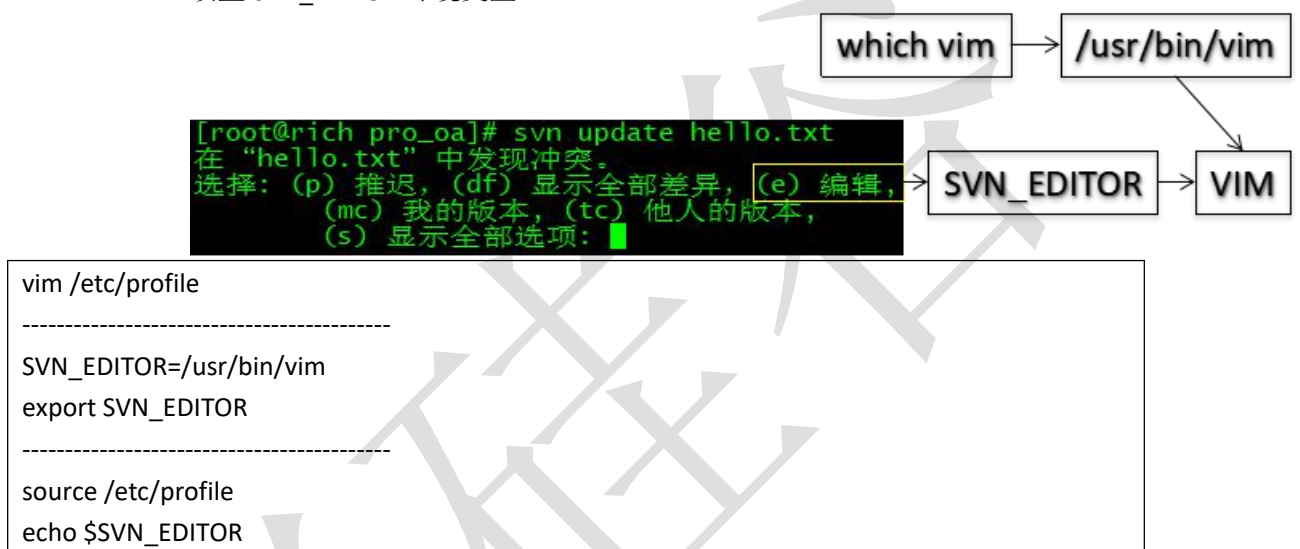
xxx.r[大版本号]文件：发生冲突时服务器端文件内容

7.4 冲突的手动解决

- 第一步：删除冲突发生时产生的三个多余文件
- 第二步：删除冲突文件内多余的符号
- 第三步：把文件编辑到满意的状态
- 第四步：提交

7.5 冲突的半自动解决

- 设置 SVN_EDITOR 环境变量



- 解决的过程
 - 使用 e 选项进入文件内容编辑界面

```

[root@rich pro_oa]# svn update good.log
在 "good.log" 中发现冲突。
选择：(p) 推迟，(df) 显示全部差异，(e) 编辑，
      (mc) 我的版本，(tc) 他人的版本，
      (s) 显示全部选项： e
  
```

- 进入 vim 编辑器编辑文件内容

```

1 1111111
2 2222222@@@@@@@@
3 3333333
4 4444444 edit by sally
5 resolve conflict on this file
6 4444444 edit by harry
7 5555555
  
```

- 编辑完成后使用 r 选项标记为已解决

```

选择：(p) 推迟，(df) 显示全部差异，(e) 编辑，(r) 已解决，
      (mc) 我的版本，(tc) 他人的版本，
      (s) 显示全部选项：
  
```


7.6 减少冲突的发生

- 尽可能在修改文件前先进行更新操作，尽量在最新版基础上修改文件内容。
- 尽量减少多人修改同一个文件的可能性。
- 加强团队成员之间的沟通。

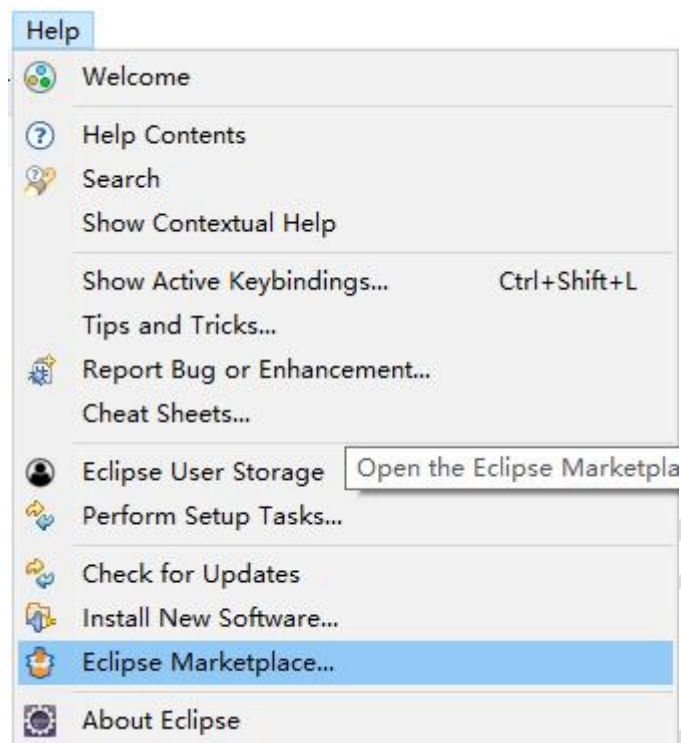
8 Eclipse 的 SVN 插件

8.1 简介

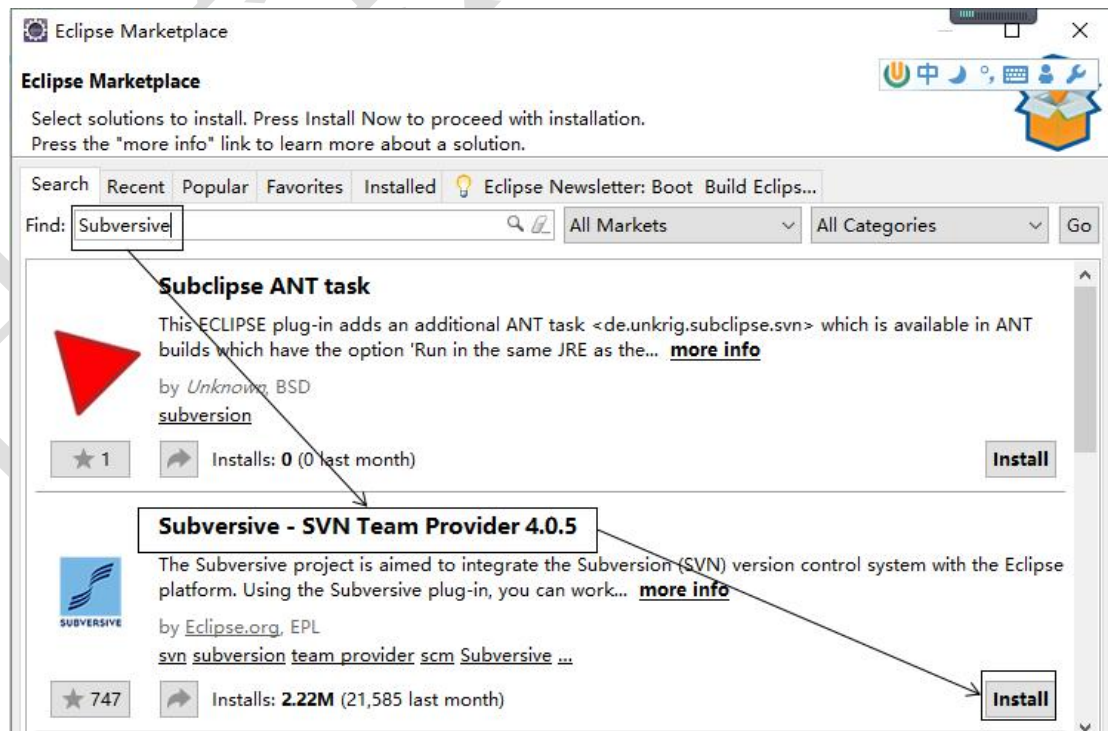
- Subversive
Eclipse 团队开发的 SVN 插件。
- Subclipse
Apache 的 SVN 团队开发的 Eclipse 插件。

8.2 Subversive 的安装

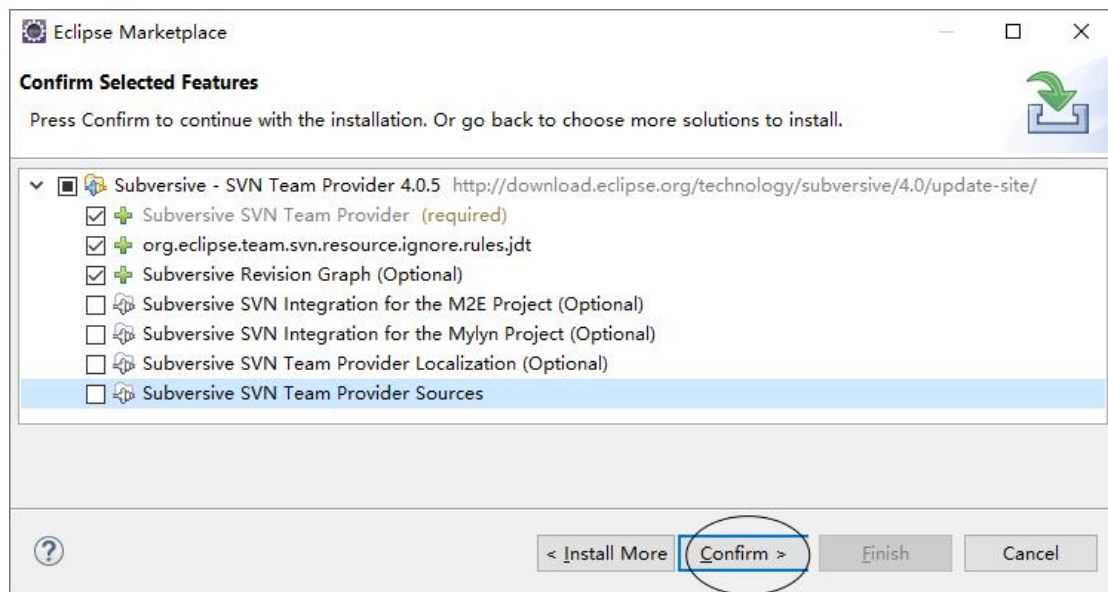
- 安装前
 - Eclipse→Window→Preferences→Team
 - ▼ Team
 - File Content
 - > Git
 - Ignored Resources
 - Models
 - 用户家目录下没有 Subversion 目录
- 安装过程
 - 打开 Eclipse 应用市场



■ 搜索 Subversive

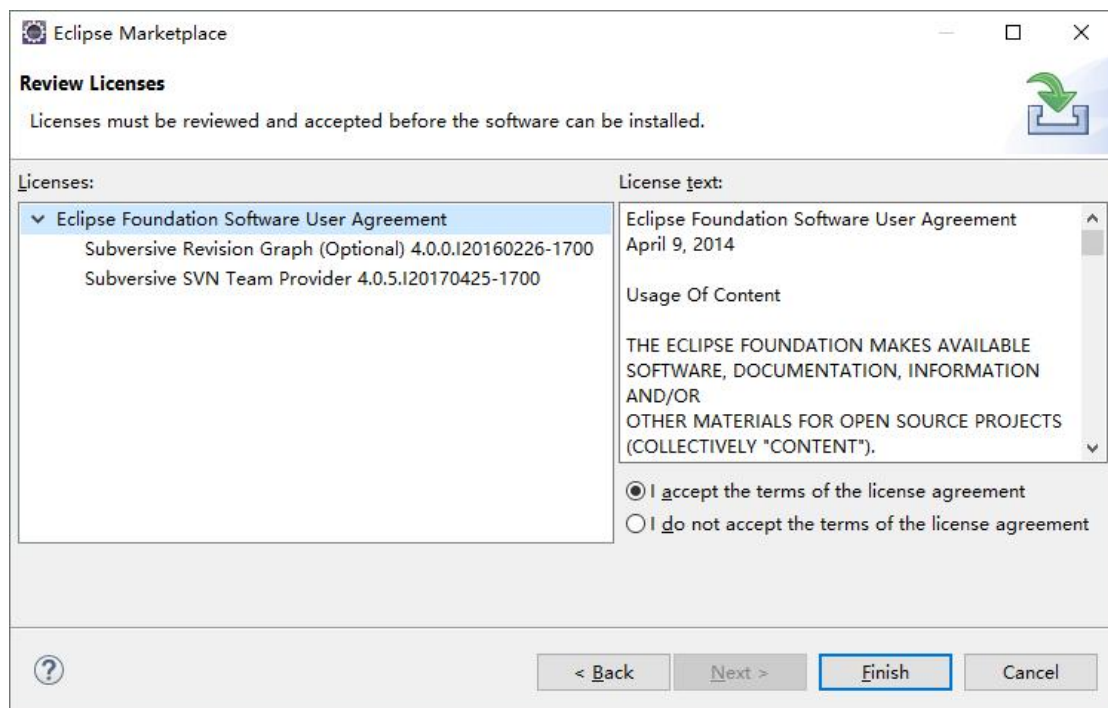


■ 确认安装项目

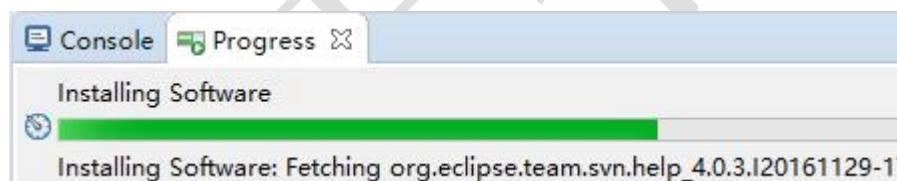


点 yes 即可

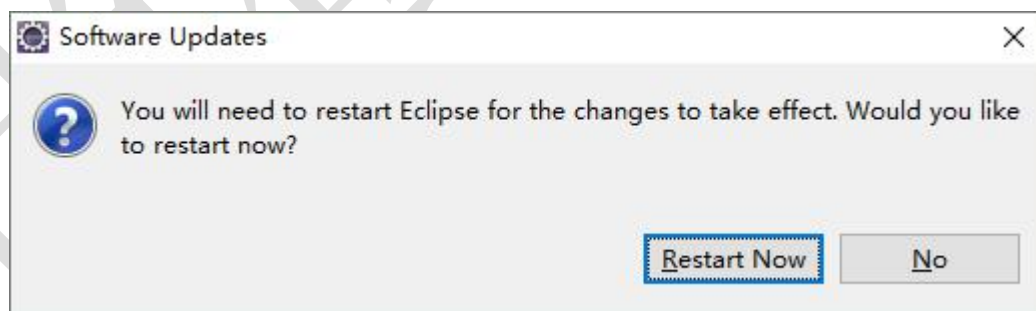
- 同意协议，点 Finish



■ 安装过程

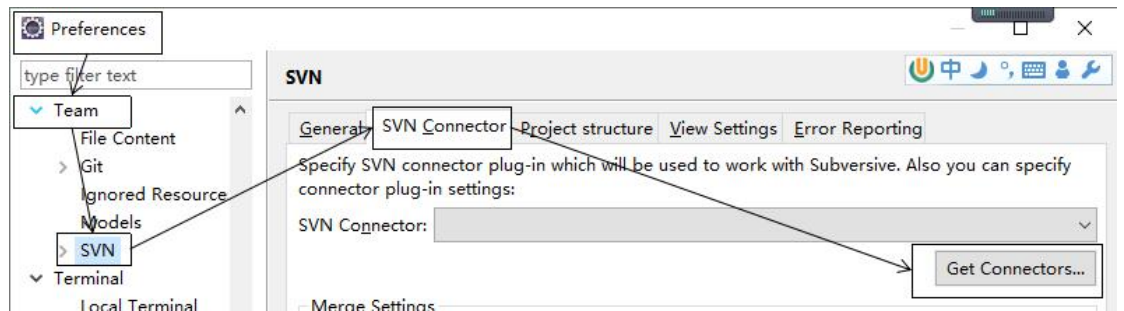


■ 确认重启

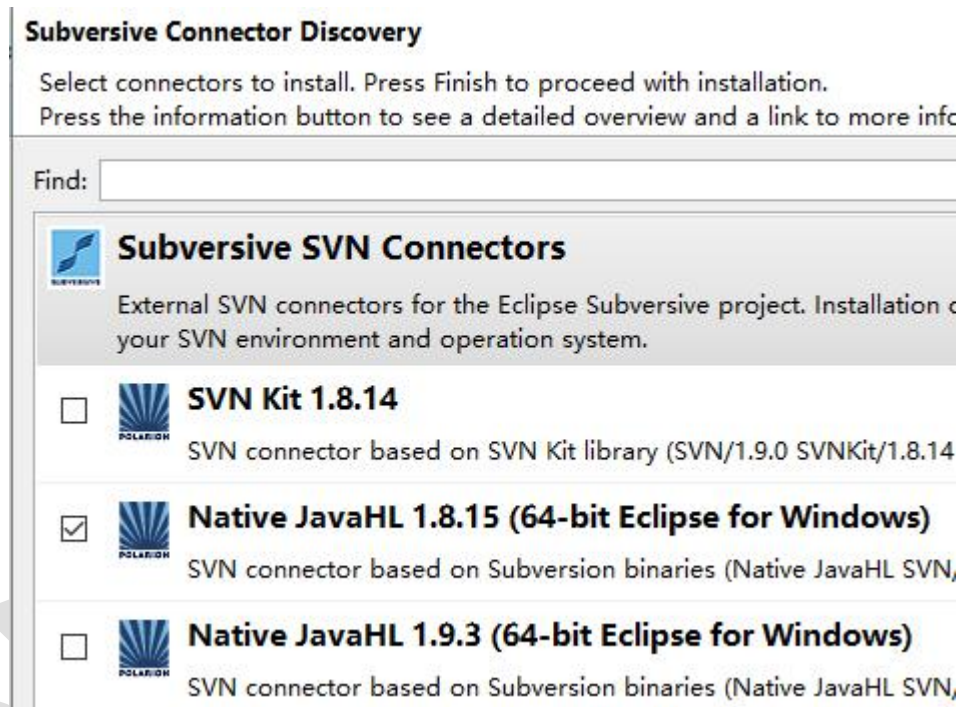


8.3 SVN Connector 安装

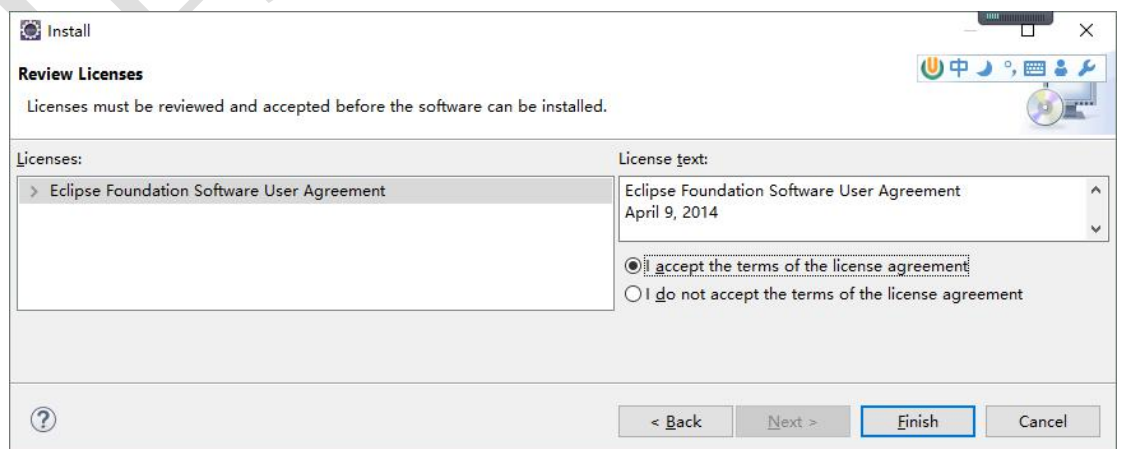
➤ Get Connectors



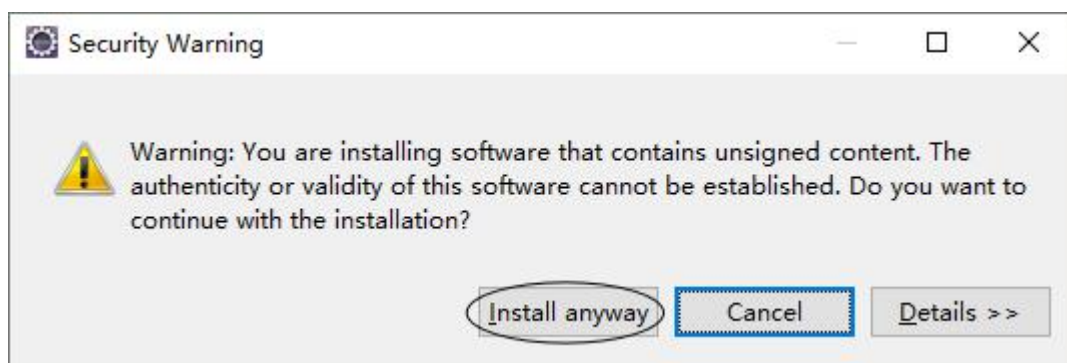
- 选择要安装的 Connector



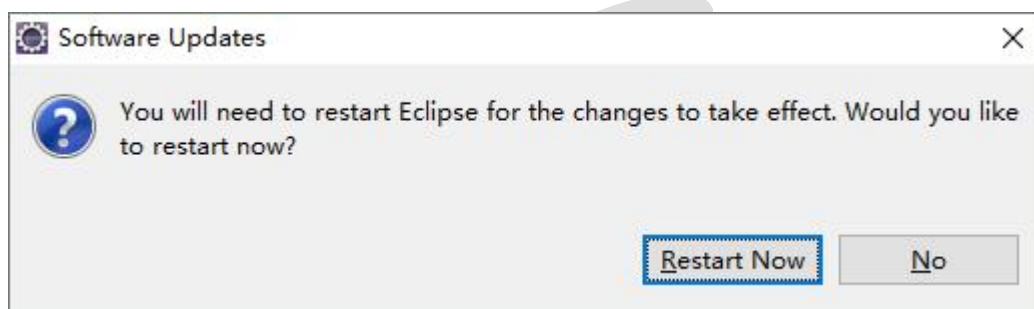
- 下一步、下一步……
- 同意协议，Finish



- 点击 Install anyway

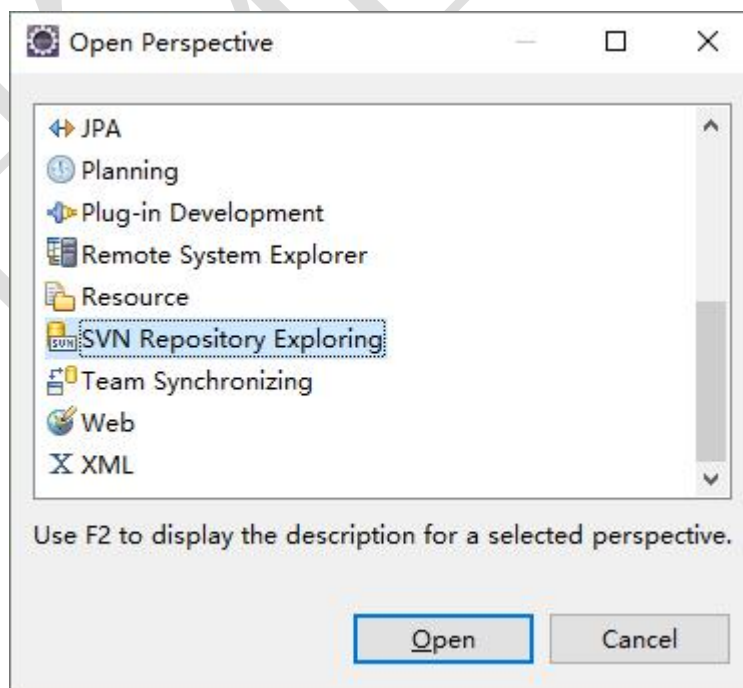


- 确认重启



8.4 创建资源库位置

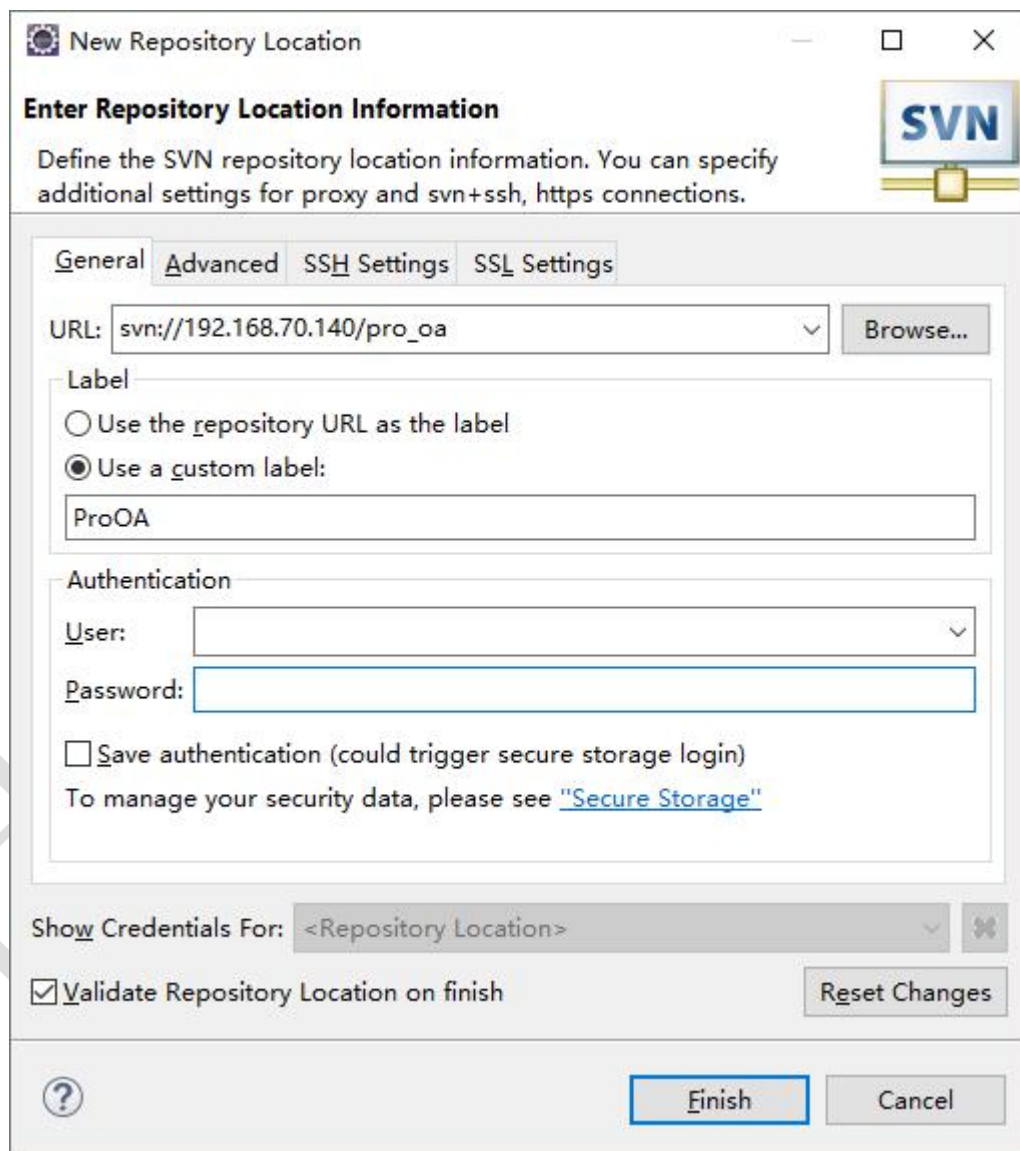
- 目的：让本地 Eclipse SVN 插件知道 SVN 服务器的位置
- 操作步骤
 - 第一步：切换透视图



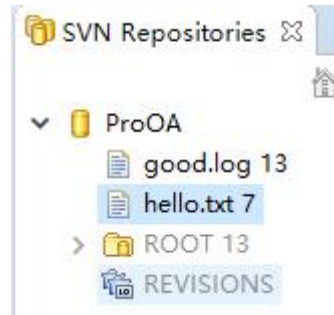
- 第二步：创建资源库位置



- 输入 SVN 服务器的 URL 地址

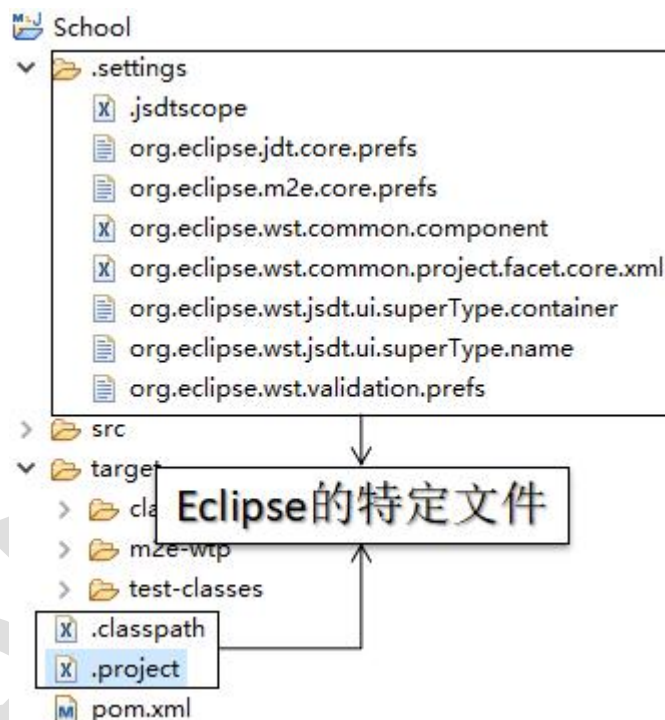


- 效果



8.5 Eclipse 工程中忽略文件

- Eclipse 特定文件



- 忽略特定文件的原因
 - 在服务器上最终运行工程完全没有关系
 - 开发团队中，并不是所有参与开发的成员都使用相同的 IDE，所以代码文件之外的 IDE 特定文件有可能有区别。如果这些文件也都上传到 SVN 服务器，那么很可能产生冲突。不同 IDE 之间可以基于 Maven 的标准目录结构识别工程。
- 配置全局范围忽略文件的操作方式
 - 配置文件位置

~\AppData\Roaming\Subversion\config

例如：C:\Users\Lenovo\AppData\Roaming\Subversion\config

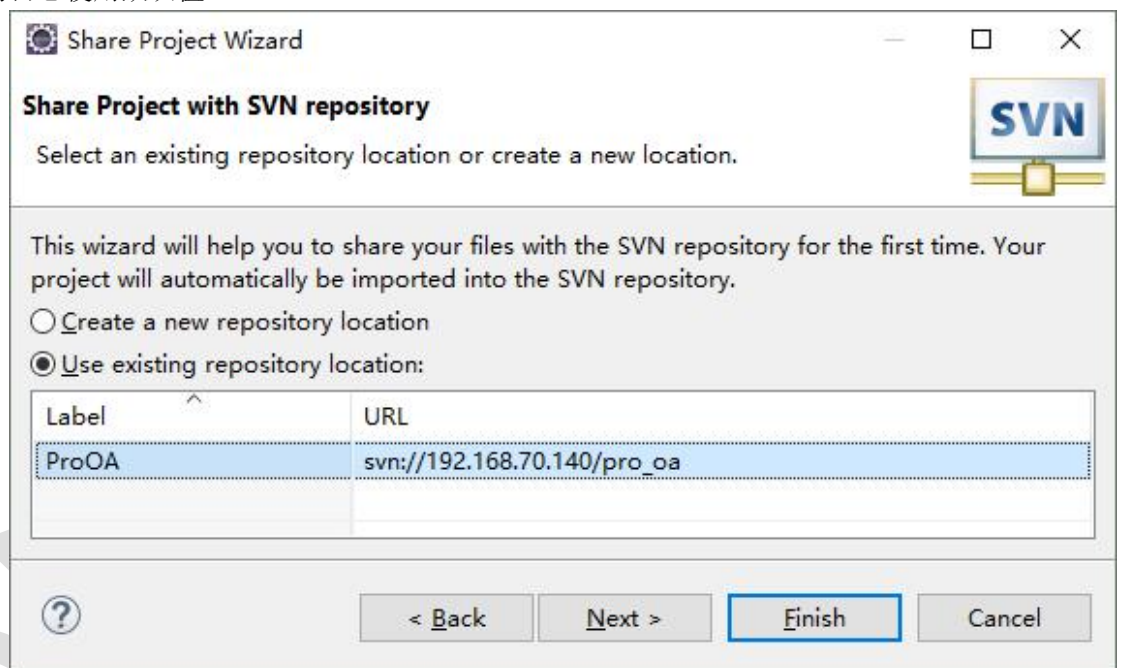
- 要修改的配置项
把 global-ignores 的注释打开

```
global-ignores = *.o *.lo *.la *.al .libs *.so *.so.[0-9]* *.a *.pyc *.pyo __pycache__ *.rej *~  
### .* *.swp .DS_Store .settings */.settings/* .classpath .project target */target/*
```

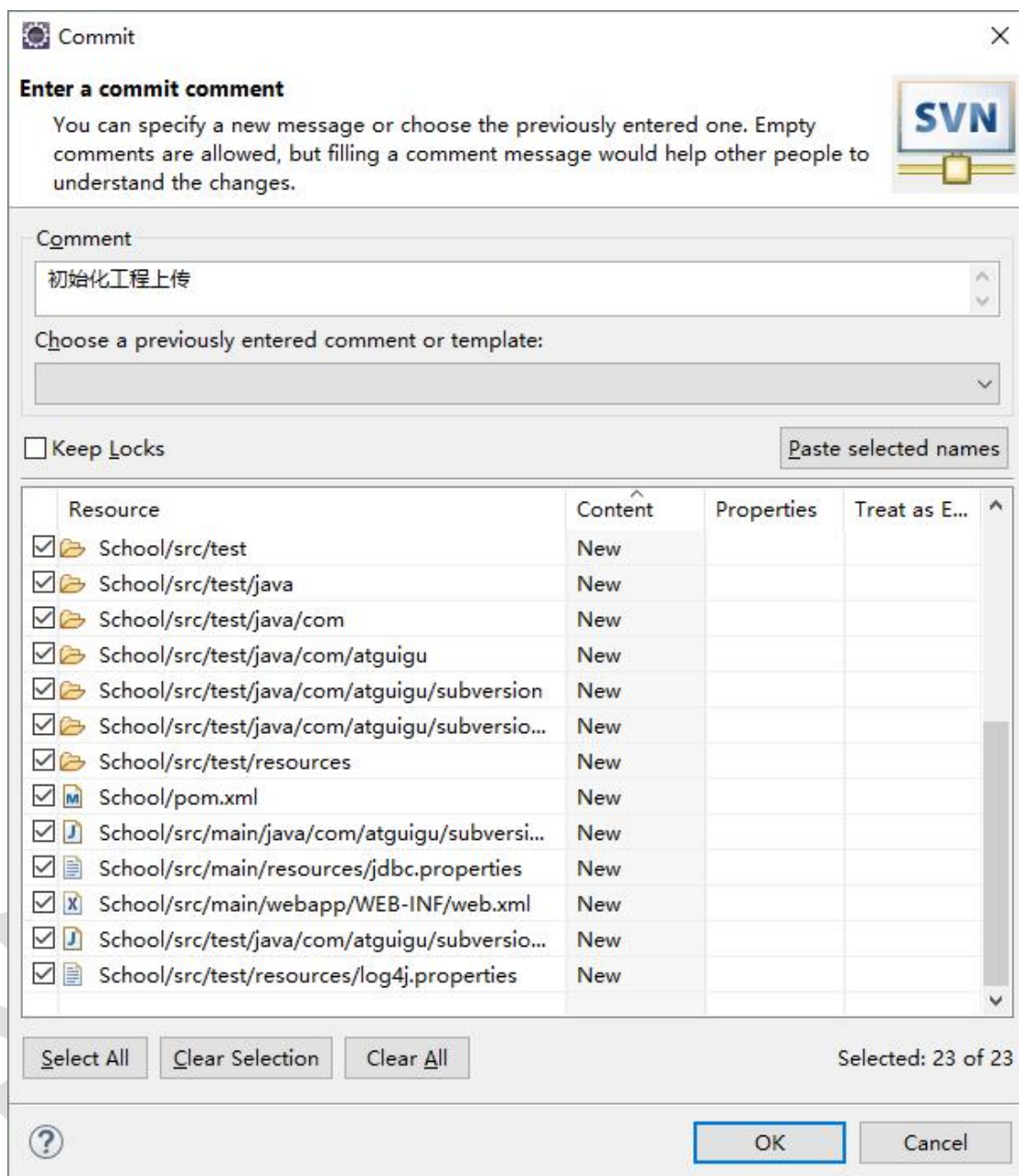
※如果在 Eclipse 中操作，target 目录会自动忽略；如果使用 TortoiseSVN 则需要追加 target 目录设置。

8.6 分享工程

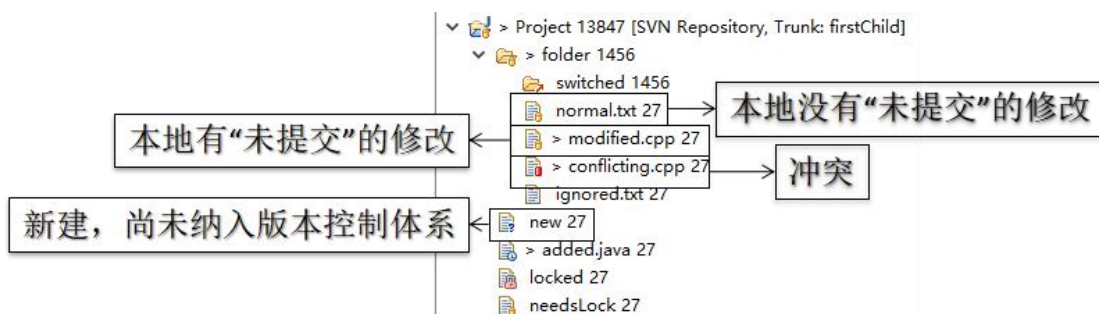
- 第一步：工程→右键→Team→Share Project...
- 第二步：版本控制工具中选择 SVN
- 第三步：选择一个已经存在的资源库位置或新建一个
可以直接点 Finish（工程在 SVN 服务器端的目录名和工程名一致；上传工程目录的日志使用默认值）



- 第四步：确认工程根目录下子目录和文件是否全部上传

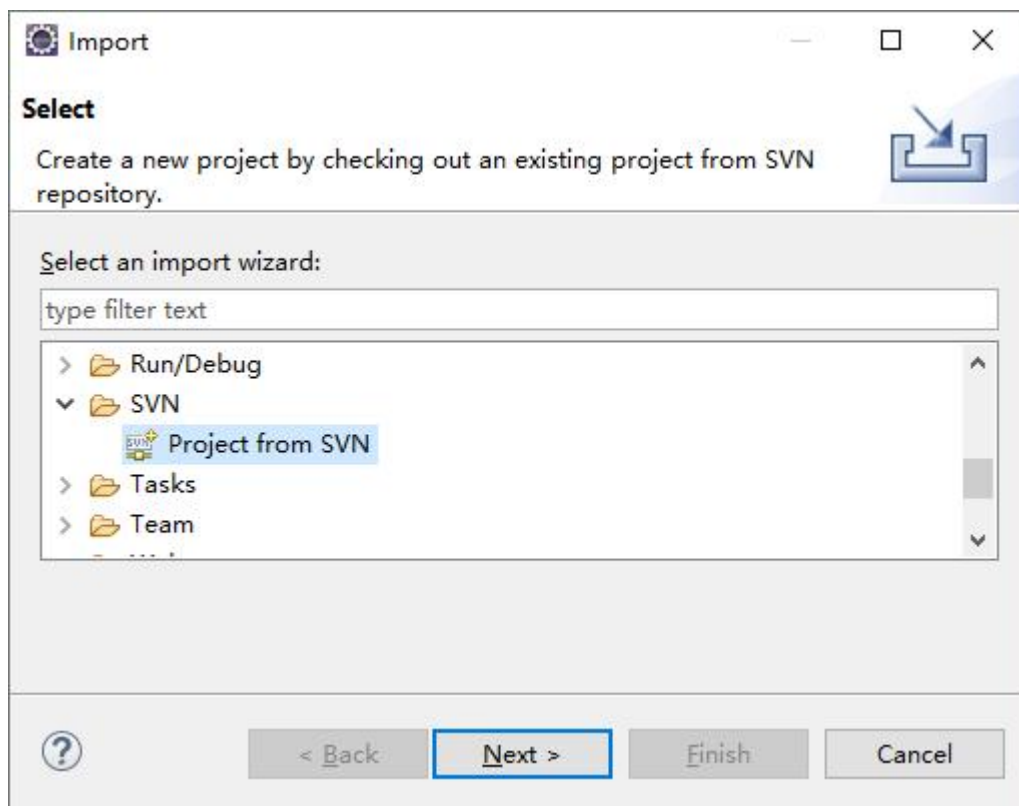


8.7 常见图标含义

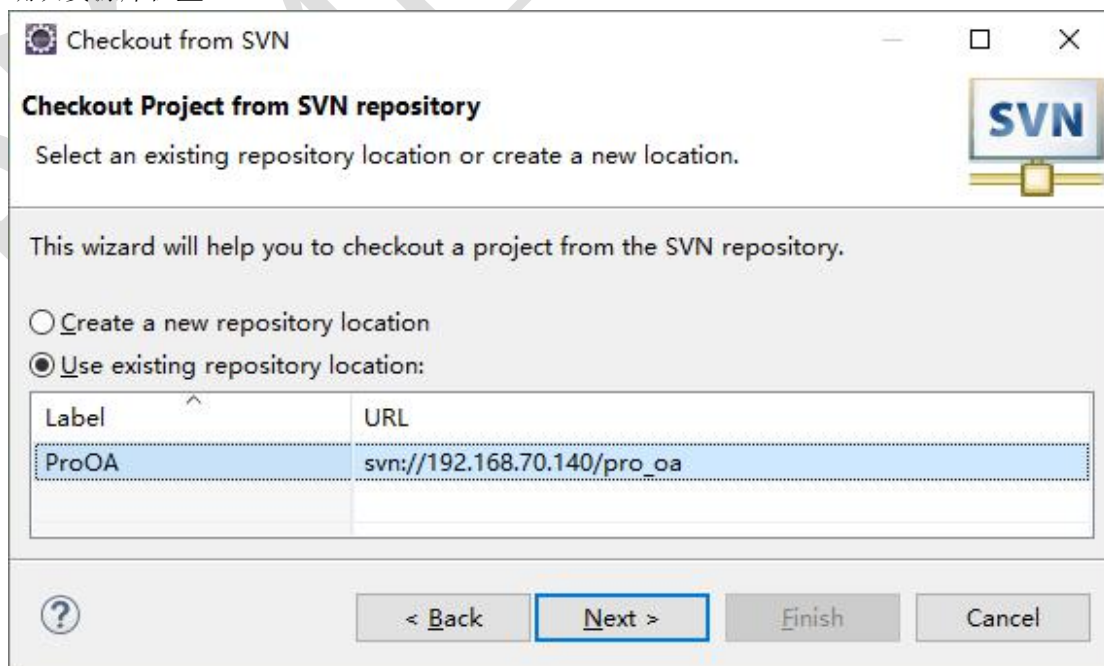


8.8 检出操作

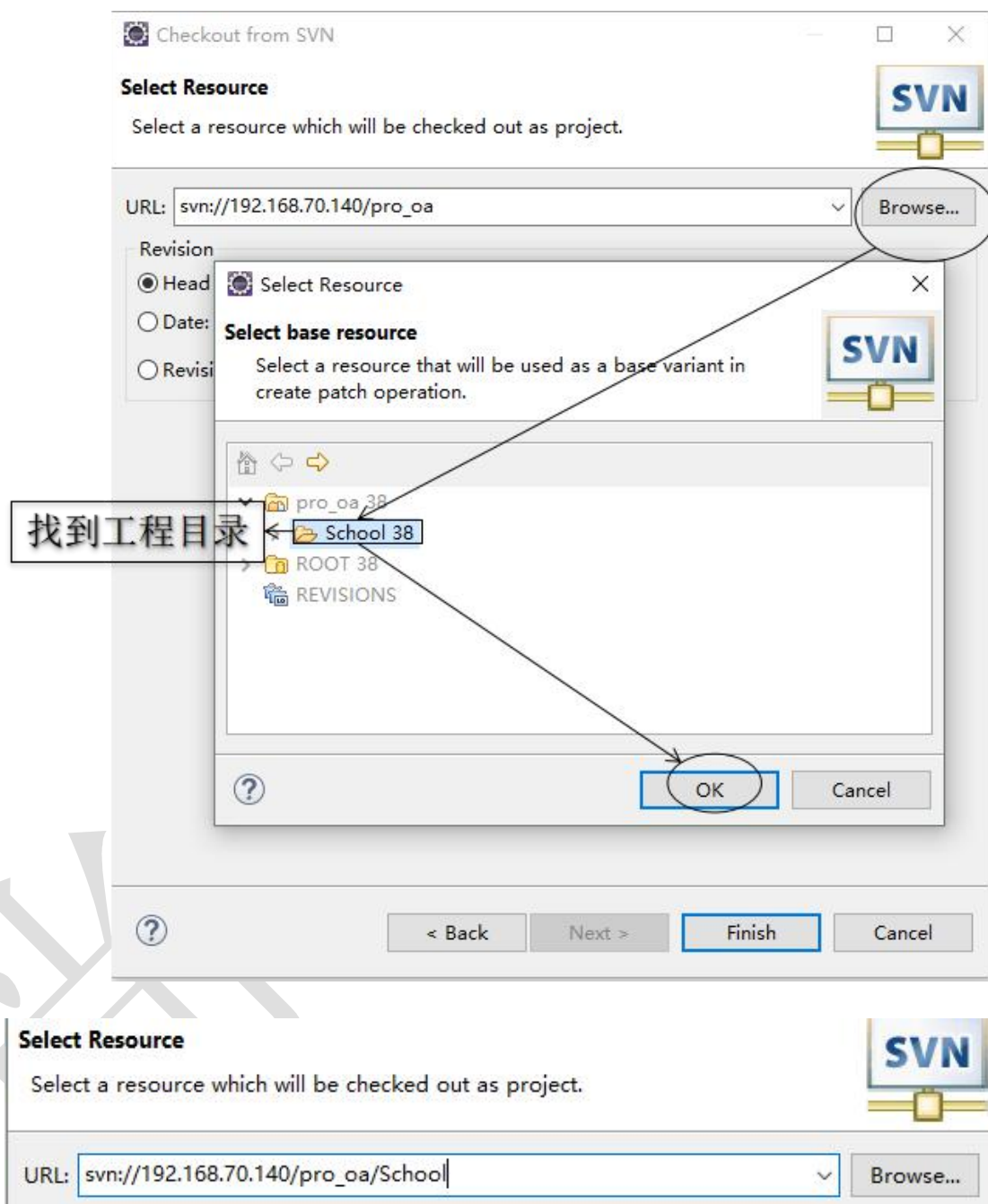
- 在 Eclipse 中执行 Import 操作



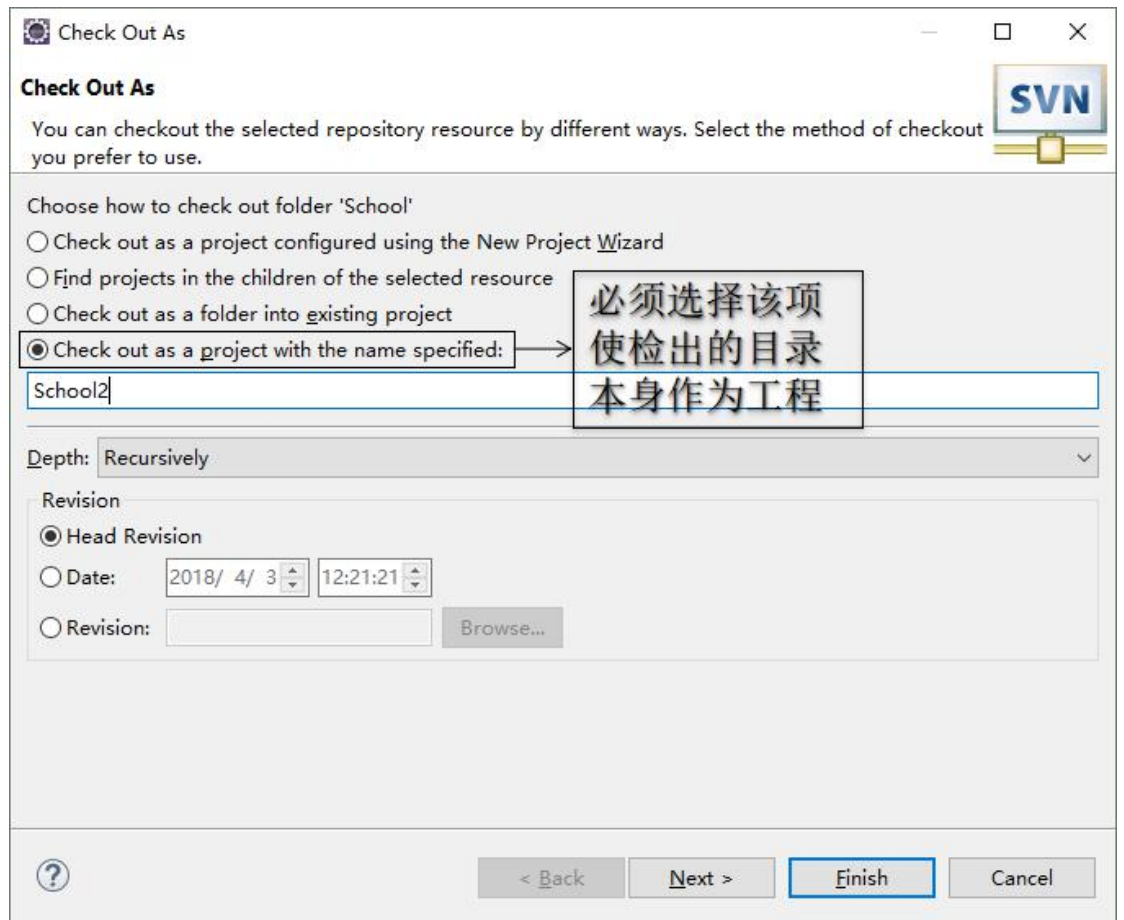
- 确认资源库位置



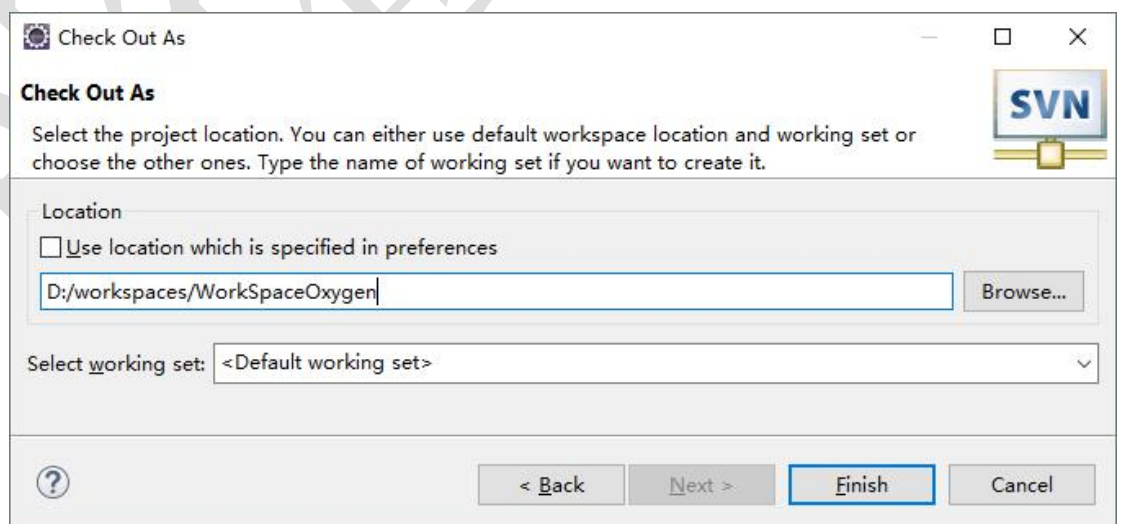
- 找到 SVN 服务器端工程对应的目录



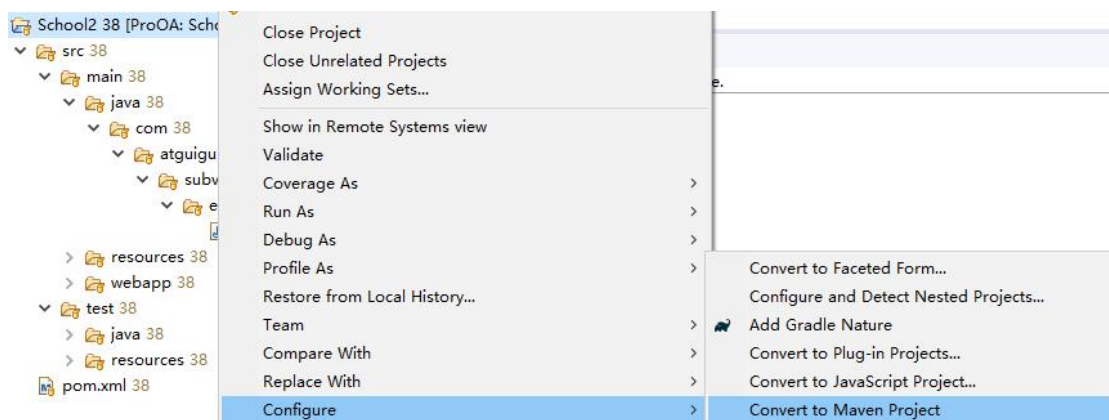
➤ 选择检出方式



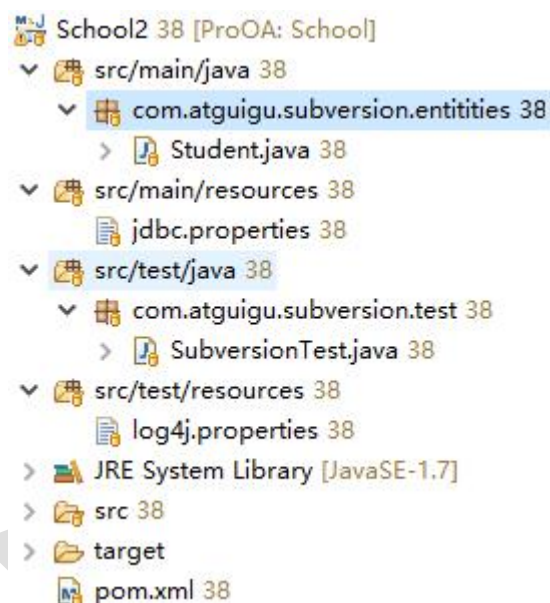
➤ Finish



➤ 转换工程类型



➤ 最终效果

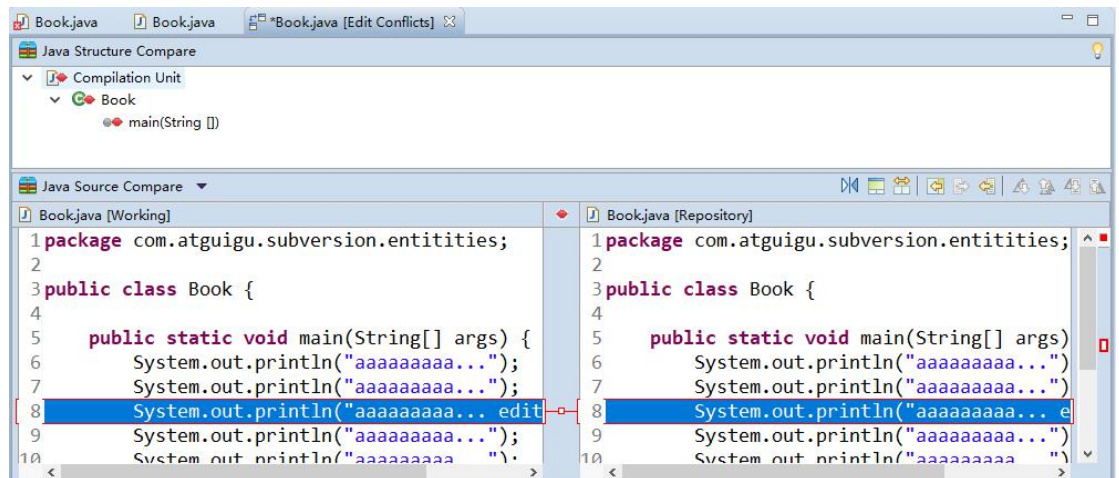


8.9 基本操作

资源→右键→Team→相关菜单项

8.10 解决冲突

- 第一步
冲突文件→右键→Team→Edit Conflicts
- 第二步



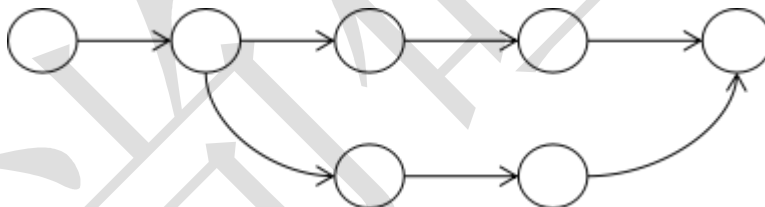
在这个界面中修改至满意，关闭界面。

- 标记为“已合并”
冲突文件→右键→Team→Mark as merged
- 提交

9 分支

9.1 概念

在版本控制过程中，使用多个分支同时推进多个不同功能开发。



不使用分支开发：人与人之间协作

使用分支开发：小组和小组之间协作

9.2 应用场景举例

蓝色皮肤界面功能：小组 1

用户账号管理功能：小组 2

支付功能：小组 3

.....

9.3 作用

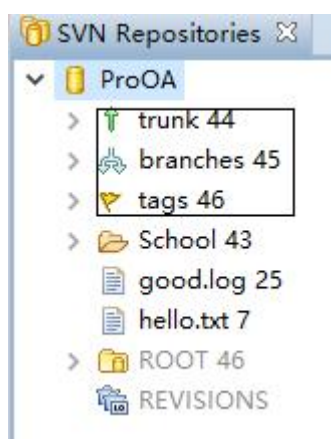
- 多个功能开发齐头并进同时进行
- 任何一个分支上功能开发失败，删除即可，不会对其他分支造成影响

9.4 相关目录

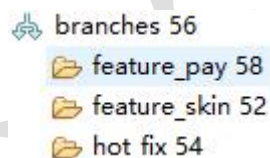
- trunk
主干
- branches
分支
- tags
存放项目开发过程中各个里程碑式的代码

9.5 创建相关目录

资源库位置→右键→New...→Folder

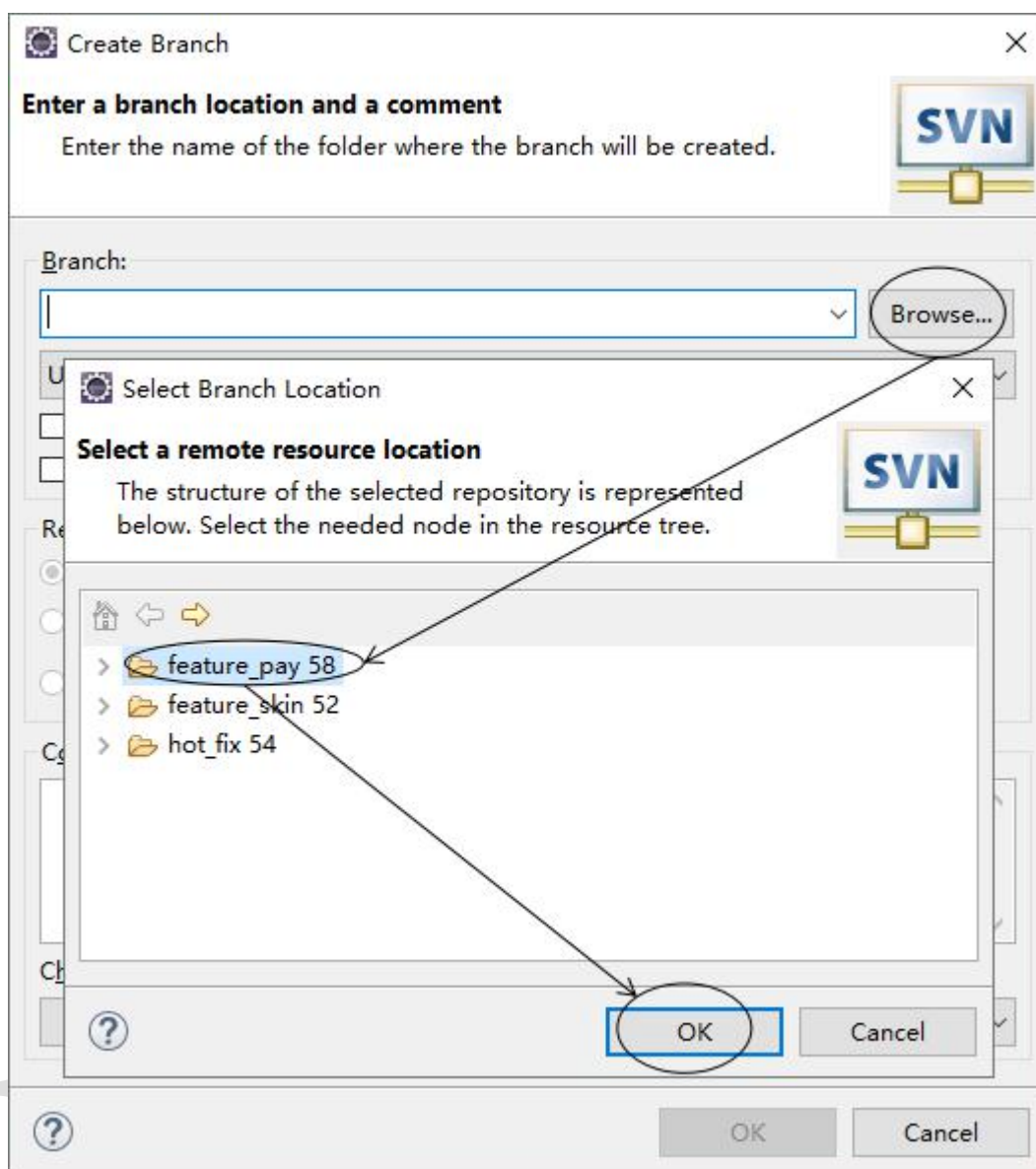


9.6 创建各个具体分支的目录



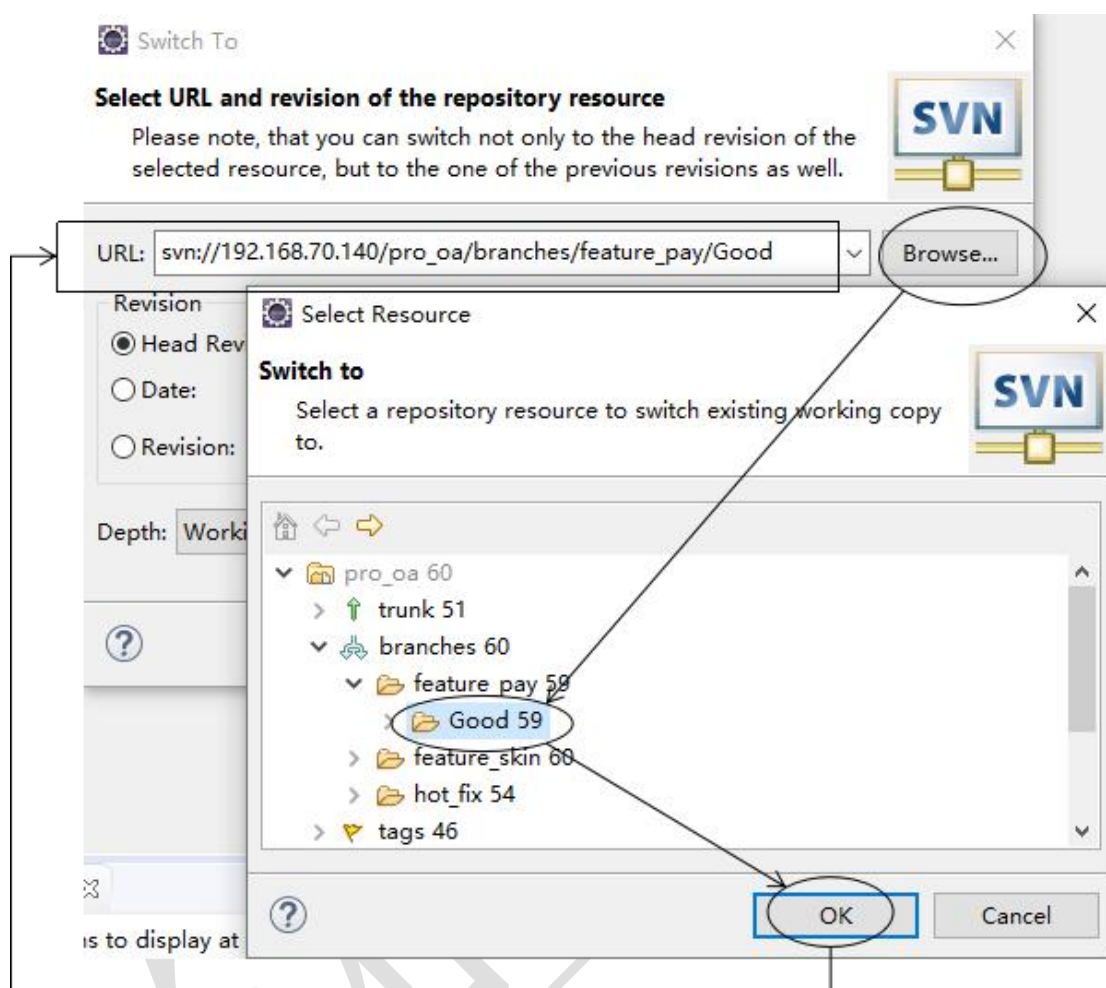
9.7 创建代码分支

项目→右键→Team→branch...




9.8 切换分支

项目 → 右键 → Team → Switch...

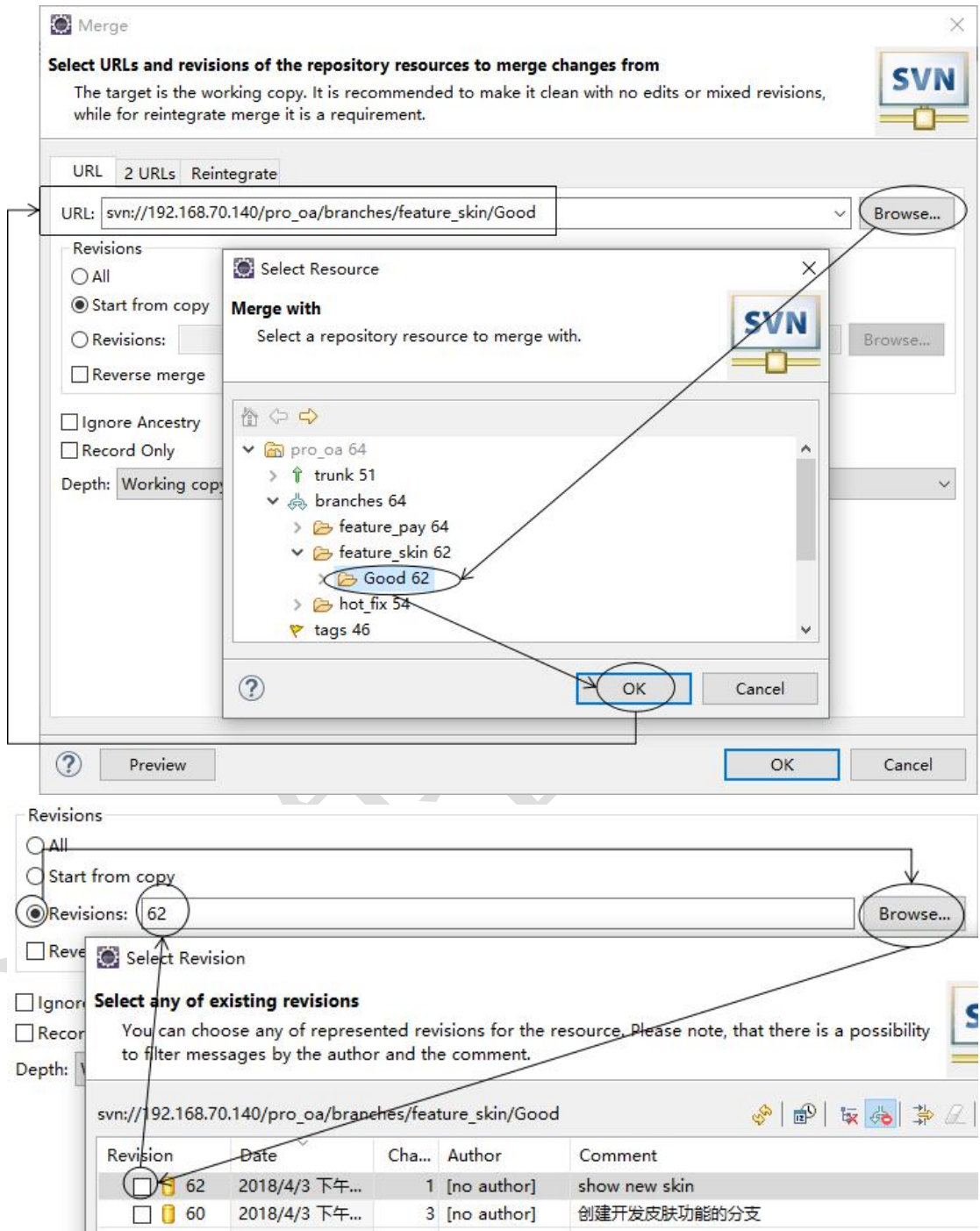


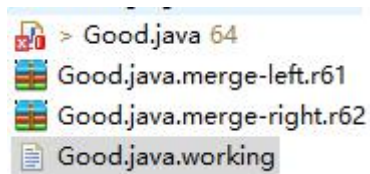
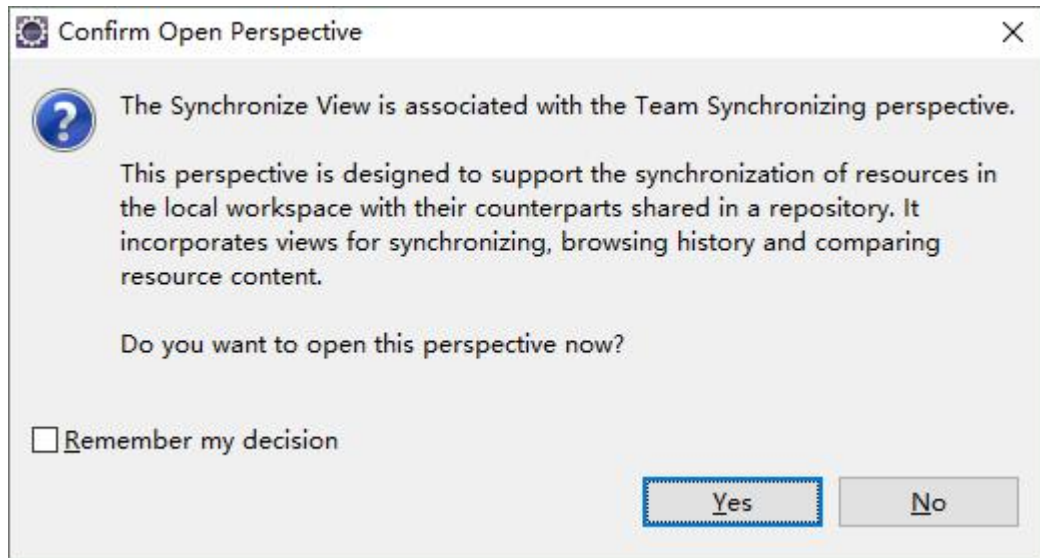
效果:

 Good 59 [ProOA, Branch: feature_pay]

9.9 合并分支

工程 → 右键 → Team → Merge...





※说明：如果两个分支各自都有新内容，需要合并两次才能够让他们内容一致

分支 1: ☆☆☆★★★

分支 2: ☆☆☆○○○

分支 1→分支 2:

分支 1: ☆☆☆★★★

分支 2: ☆☆☆○○○★★★★

分支 2→分支 1:

分支 1: ☆☆☆★★★○○○

分支 2: ☆☆☆○○○★★★★

10 SVN 权限管理

10.1 版本库中三个对应的配置文件

- 版本库配置文件目录
/var/svn/repository/pro_oa/conf
- svnserve.conf 文件

12 # anon-access = write	匿名访问
13 auth-access = write	授权访问

20 password-db = passwd	指定设置用户名密码的配置文件
27 authz-db = authz	分配权限的配置文件

➤ passwd 文件

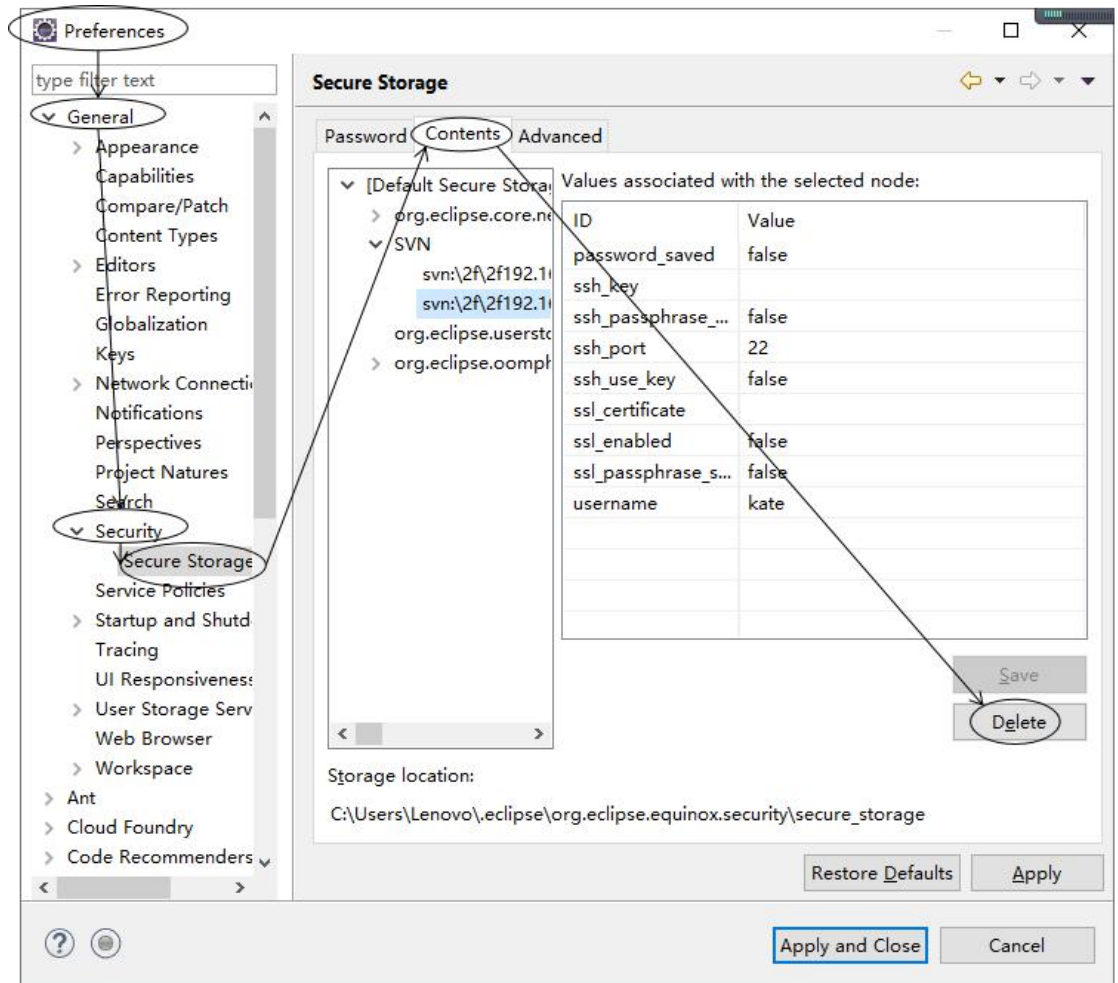
<pre>[users] # harry = harrysecret # sally = sallysecret tom = 123123 jerry = 123123 kate = 123123</pre>	<p>例子</p> <p>用户名 = 密码</p>
--	---------------------------

➤ authz 文件

<pre>21 [groups] 22 # harry_and_sally = harry,sally 23 # harry_sally_and_joe = harry,sally,&joe 24 kaifa = tom,jerry</pre>	<p>例子</p> <p>用户组 = 用户,用户</p>
<pre>30 [/] 31 @kaifa = rw 32 kate = r 33 * =</pre>	<p>针对版本库根目录进行权限设置</p> <p>@组名 = 权限值</p> <p>用户名 = 权限值</p> <p>上面已经授权的用户以外其他用户没有任何权限</p>

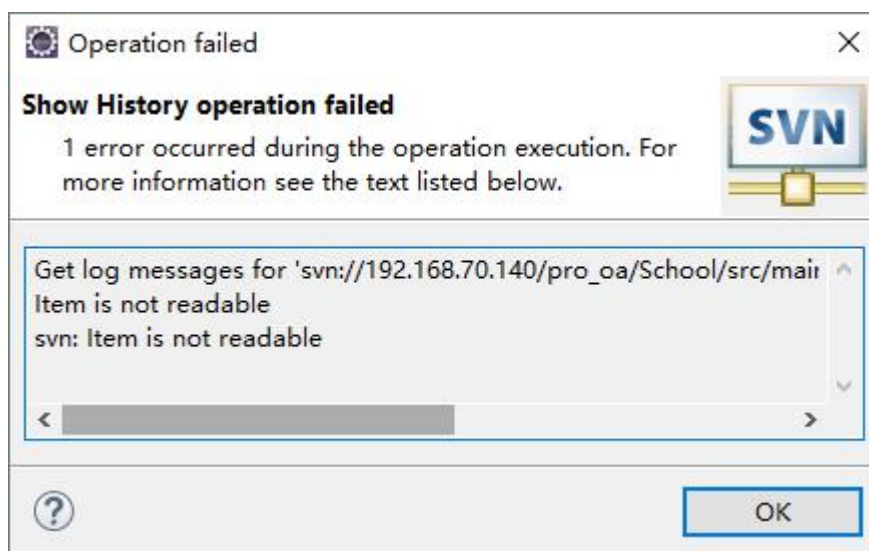
10.2 客户端测试

※Eclipse 中删除曾经登录过的用户名密码的操作方式



11 查看历史记录

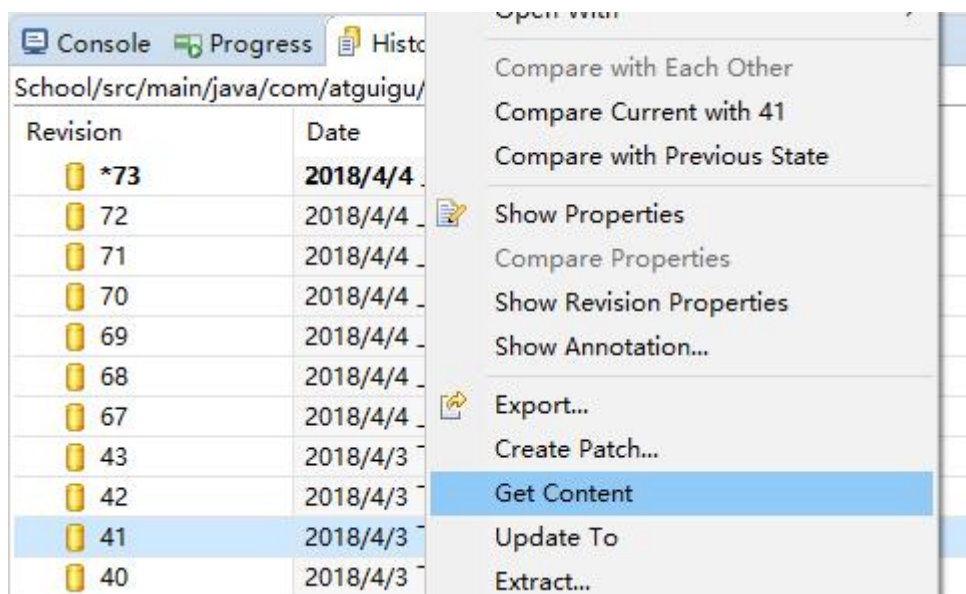
11.1 条目不可读问题解决



到 svnserve.conf 文件中把 anon-access 注释打开设置为 none

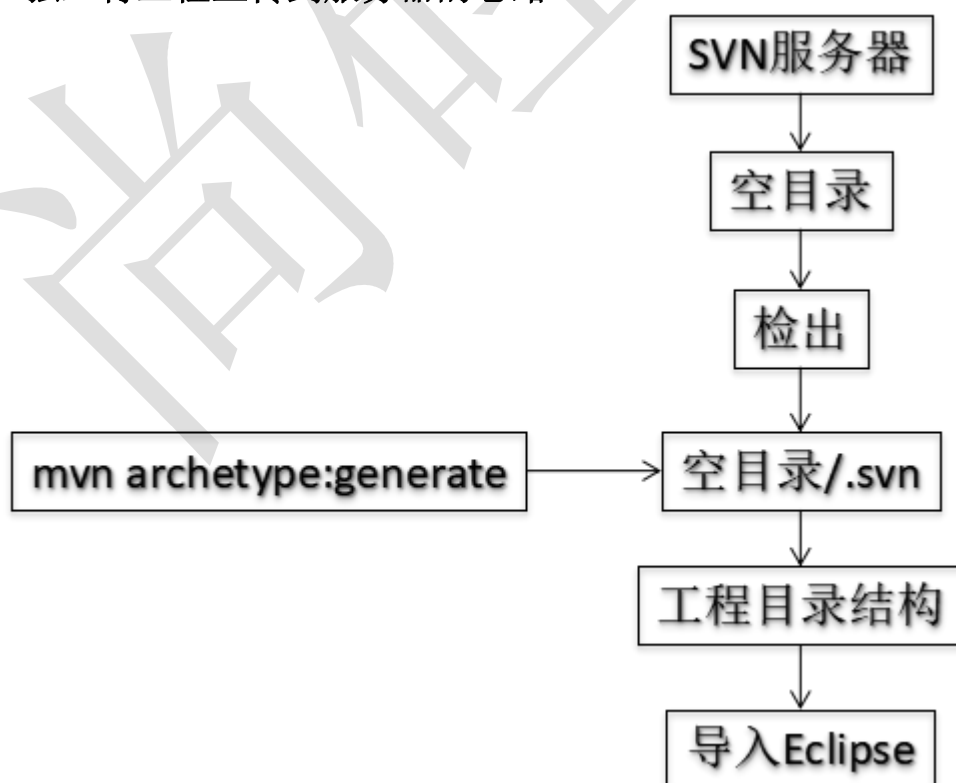
```
8 [general]
9 ### These options control access to the repository for unauthenticated
10 ### and authenticated users. Valid values are "write", "read",
11 ### and "none". The sample settings below are the defaults.
12 anon-access = none
13 auth-access = write
```

11.2 让文件回到某一个历史状态



12 TortoiseSVN

12.1 独立将工程上传到服务器的思路

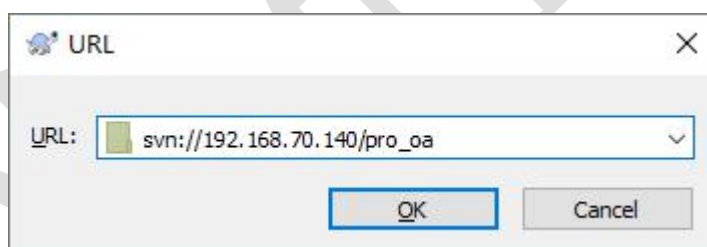
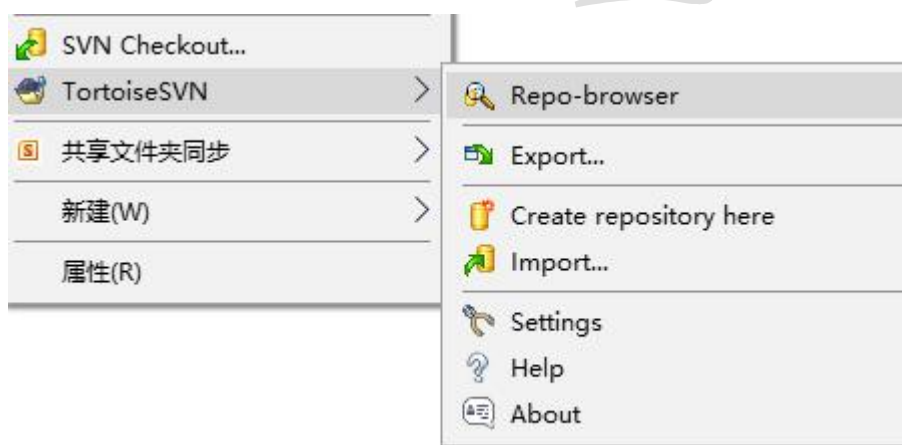


12.2 针对 archetype-catalog.xml 文件的准备工作

- 作用：Maven 生成工程目录结构过程中需要使用的配置文件
- 下载地址
<http://repo.maven.apache.org/maven2/archetype-catalog.xml>
- 复制到 Maven 的本地仓库
Maven 本地仓库根目录\org\apache\maven\archetype\archetype-catalog\[版本号目录]

12.3 操作步骤

- 打开资源库浏览器

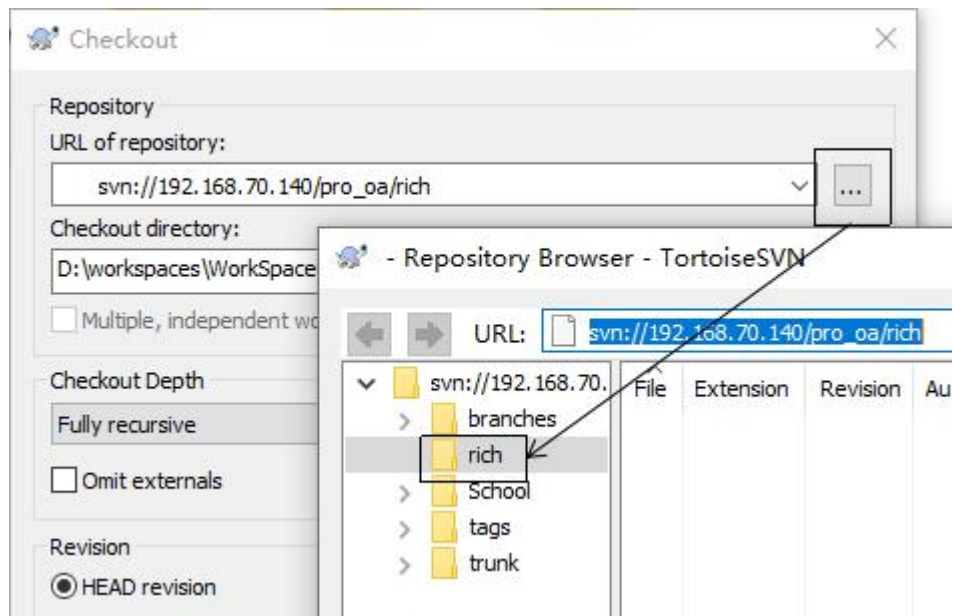


- 在 SVN 服务器上创建目录



- 检出新建的目录



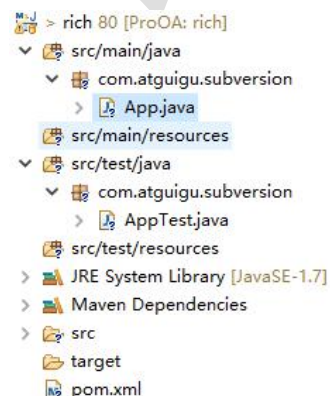


- 执行 Maven 命令
mvn archetype:generate 命令参数含义

参数名	作用/可选值
-DgroupId	生成工程坐标的 groupId 部分
-DartifactId	生成工程坐标的 artifactId 部分
-DarchetypeArtifactId	maven-archetype-quickstart 对应 jar 包工程 maven-archetype-webapp 对应 war 包工程
-DinteractiveMode	设置为 false 关闭用户交互模式
-DarchetypeCatalog	设置为 local 表示使用本地 archetype-catalog.xml 文件
-X	使用 DEBUG 级别打印日志

mvn	archetype:generate	-DgroupId=com.atguigu.subversion	-DartifactId=rich
	-DarchetypeArtifactId=maven-archetype-quickstart	-DinteractiveMode=false	-DarchetypeCatalog=local -X
mvn	archetype:generate	-DgroupId=com.atguigu.subversion	-DartifactId=rich_web
	-DarchetypeArtifactId=maven-archetype-webapp	-DinteractiveMode=false	-DarchetypeCatalog=local -X

- 导入 Eclipse 效果



微信号: creathinFeng

预告: SVN→Git/GitHub→Jenkins