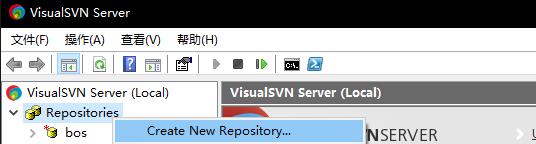
## 一 visualSVN的下载和安装



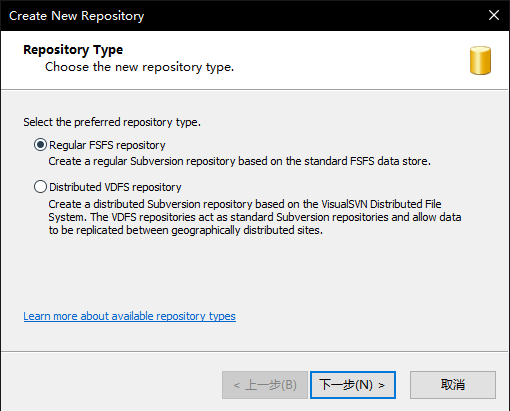
双击后一直下一步，其中两个路径设置分别是svn服务器的安装位置以及服务器文件的仓库位置！

## 二 SVN服务器仓库的创建以及操作

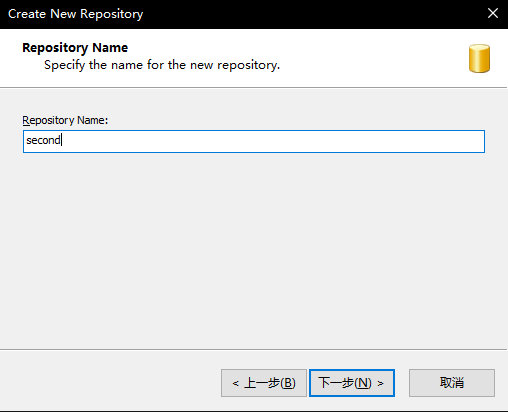


进入windows下的svn界面点击右键create一个新的仓库：

选择仓库的类型：



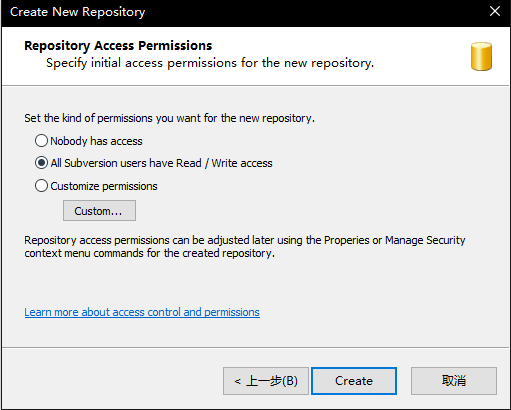
编辑仓库的名字



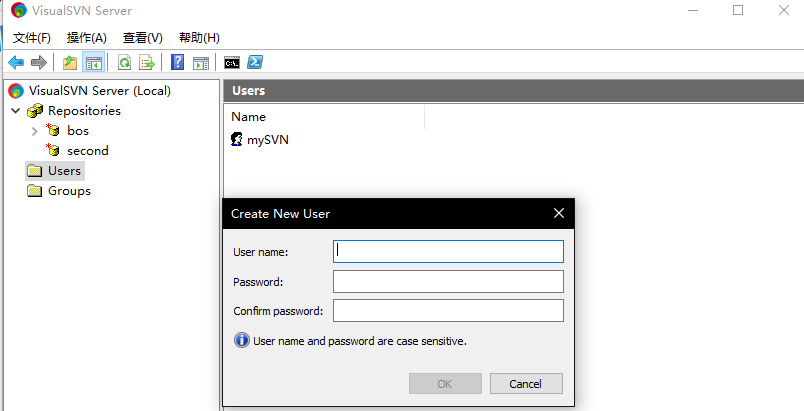
选择创建空仓库：



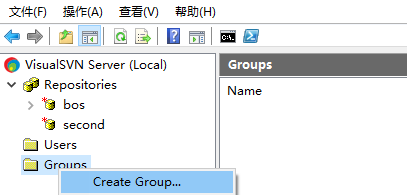
选择访问的权限：



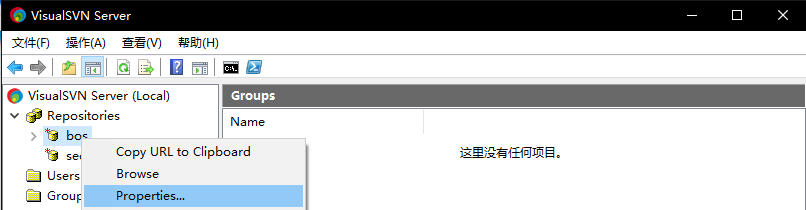
在user下右键选择创建一个新的用户，这里填写用户名和密码



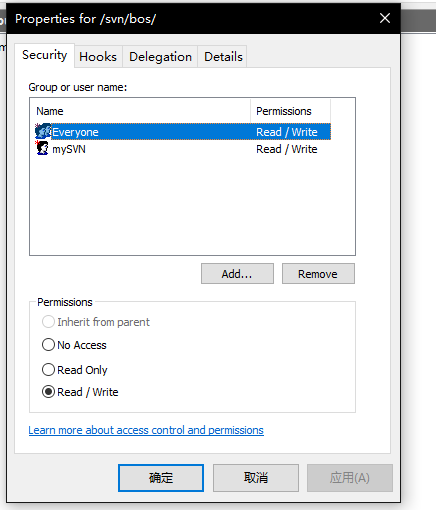
创建一个组，多个用户可以放在一个组内进行权限管理！



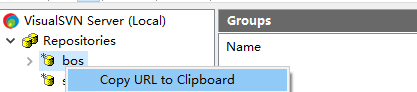
点击仓库右键，选择权限



在弹出的窗口中更改我们仓库的访问权限：



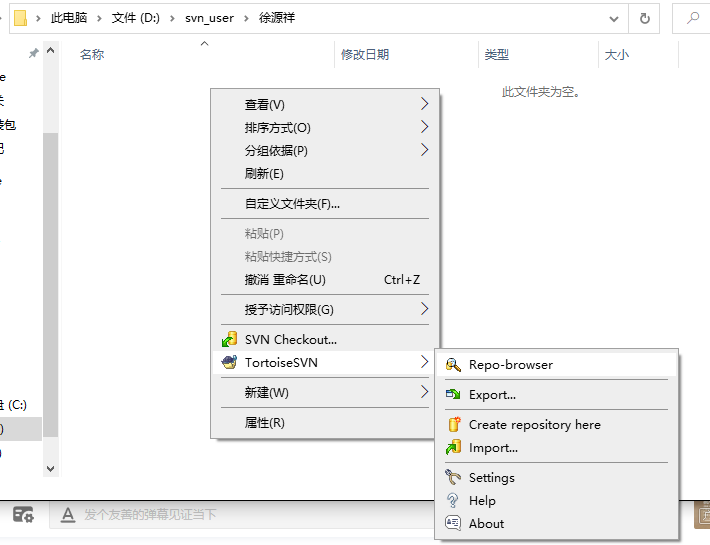
另外：这里的copy url可以将网址复制，方便再浏览器中产看svn服务器！



## 三 TortoiseSVN客户端操作

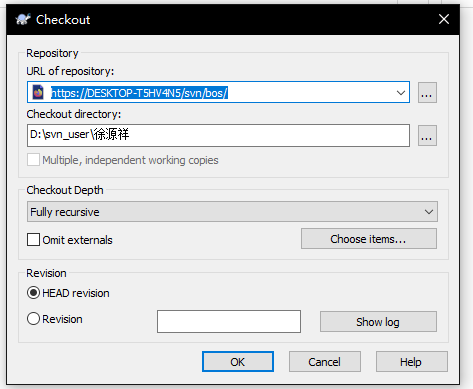
首先安装svn客户端，安装完成后创建一个文件夹用来存放代码，如下在文件夹中右键选择连接好我们的svn服务器的仓库！

第一次登陆会填写用户名和密码，再次登录就不会填写了，想要删除用户的信息 我们可以在setting中进行修改！

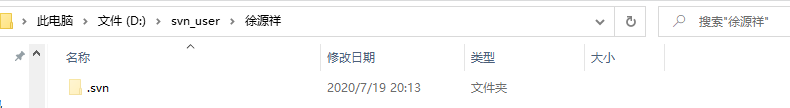


然后选择checkout将我们的svn 仓库中的代码拉取下来！

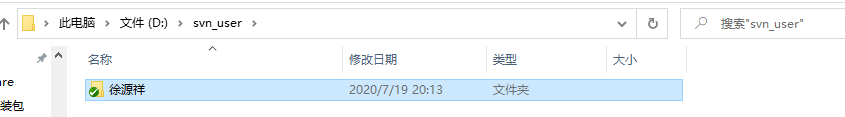




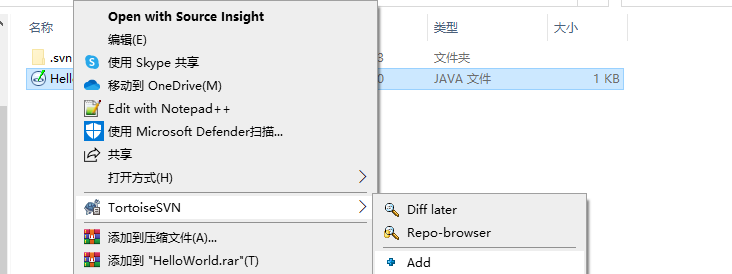
由于我们现在仓库中没有软件的任何版本，所以这里拉取下来除了一个隐藏文件什么都没有！



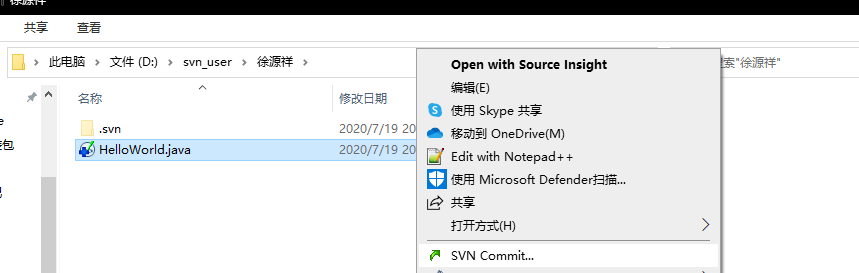
此时再次观察我们的文件夹发现已经有了变化，前面的绿色对勾表示此文件夹已经纳入了svn的管理范围！



自己写的代码文件我们通过右键add添加到本地仓库（也就是我们的徐源祥文件夹？）。

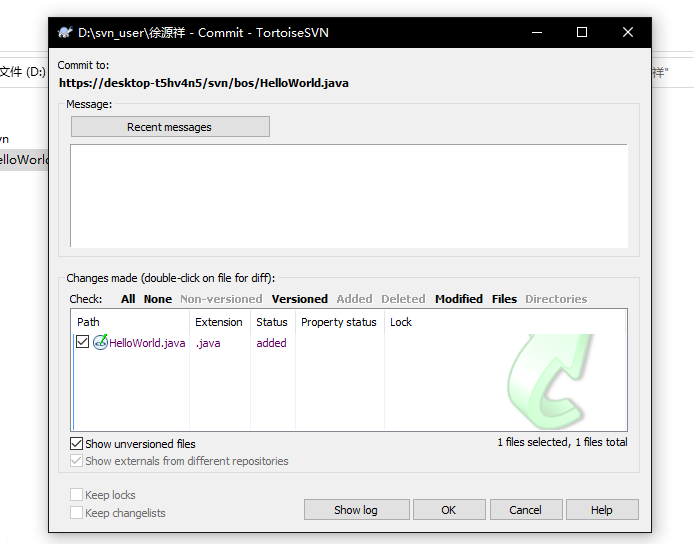


此时文件图标发生了改变：再通过右键svn commmt操作，这里就可以将我们的代码文件添加到svn的服务器！

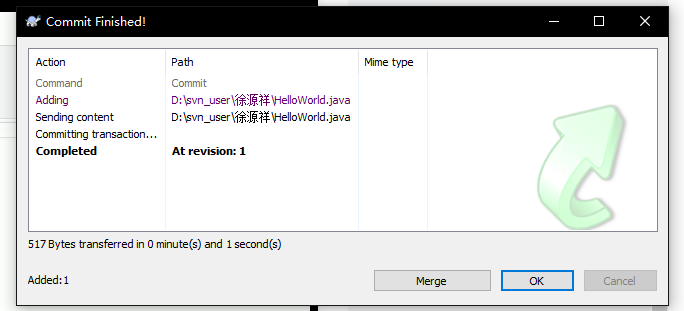


这里可以填写一些提交的日志（例如第几次提交，此次的版本信息等等）。

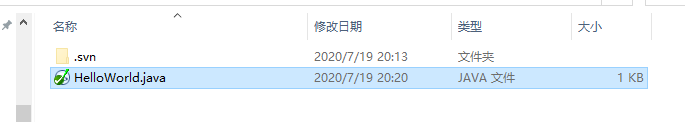
也可以选择提交的文件！



提交完成后弹出如下界面：

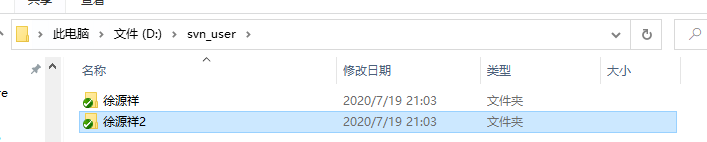


此时在观察我们的代码文件，图标又发生了改变：

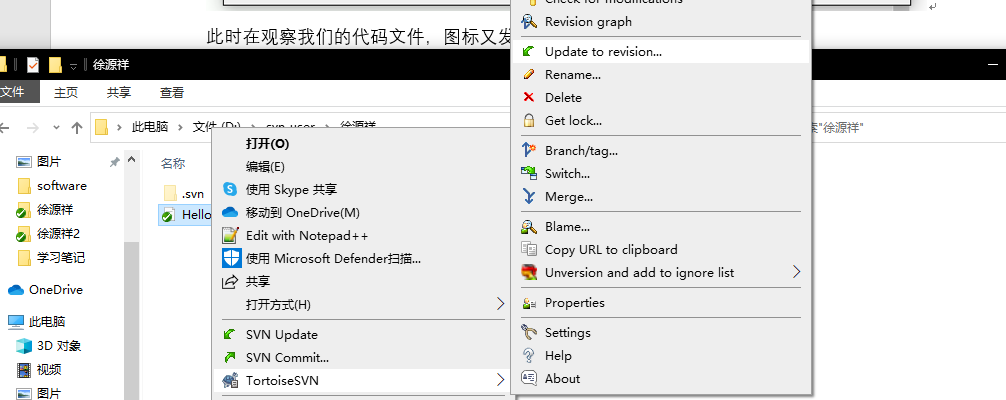


此时创建另一个文件夹，对代码进行拉取，就可以得到我们刚才上传的文件！

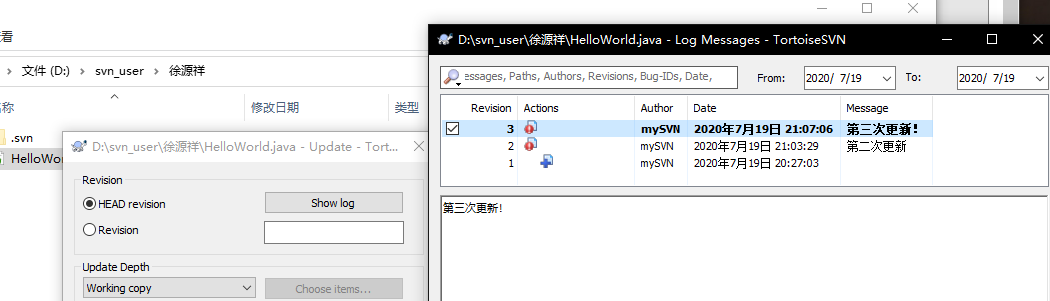
但是当我们在徐源祥中将文件进行修改，徐源祥2中的文件是不会发生变化的。需要我们进行更新（update）。



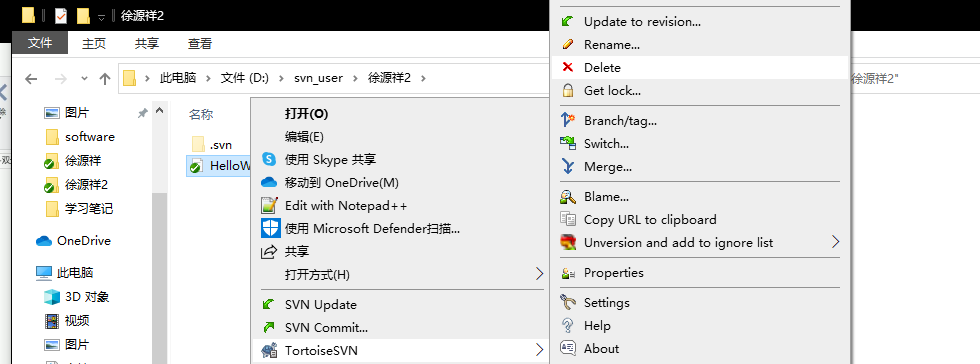
如果想要退回原先的版本，需要选择update to revision。



然后选择show log 选择版本！



删除文件的时候选择delete，此时删除的文件仅仅是本地删除了，我们通过update依然可以拉取回来。

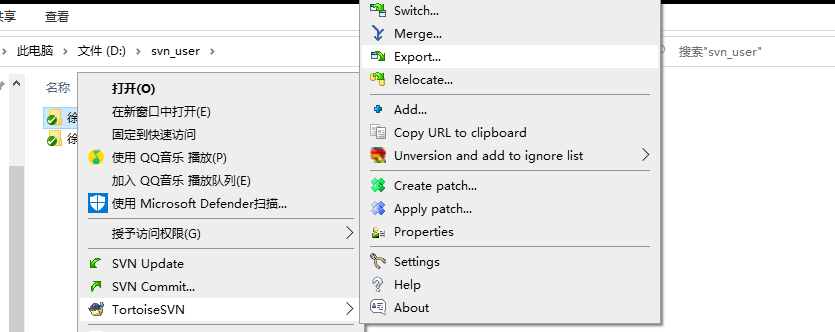


如果想要从服务器中也删除文件，需要进行commit提交！

此时其他用户进行update就可以更新版本了！

想要取回文件只需要进行版本回退就可以了！

另外svn也支持导入以及导出操作，不过导出的时候并不会将隐藏文件.svn导出，因为导出到的文件夹不受svn控制！



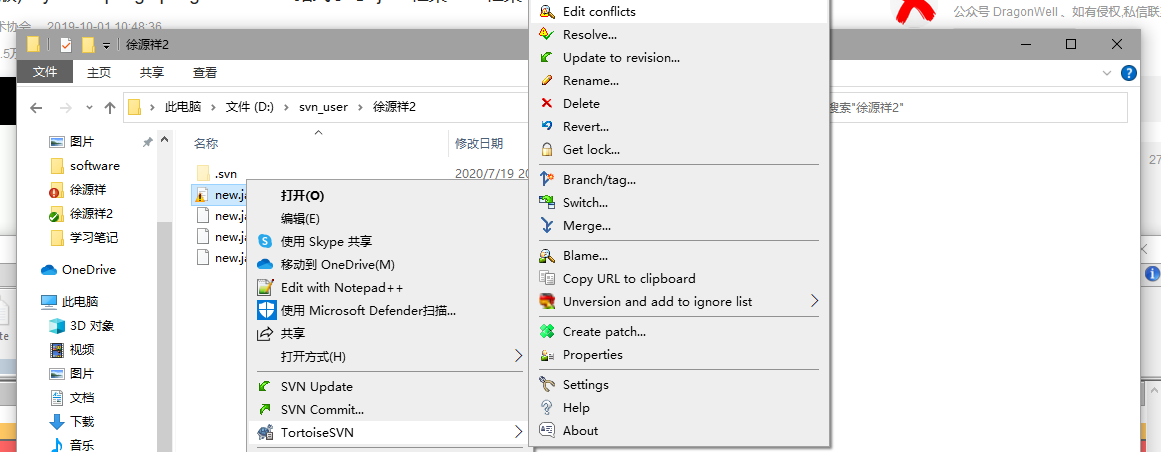
## 四 TortoiseSVN客户端操作冲突

当多个用户同时操作svn的同一个文件的时候就不可避免的会产生冲突！

例如，小王和小李同时修改文件，他们打开的版本都是5版本，小王修改完成后进行commit，此时版本已经更变为6了，过一段时间小李修改完毕也想将代码上传，就会报错，因为版本已经是11了，但是小李修改的版本依然是5版本，就发成了冲突，同时系统会在小李的文件夹下给出两个文件包，一个是小李打开的版本代码信息，另一个是当前svn的代码版本信息。

解决的方法其实很简单，我们在修改带代码前尽量先进行update将本地代码和svn的版本进行同步。

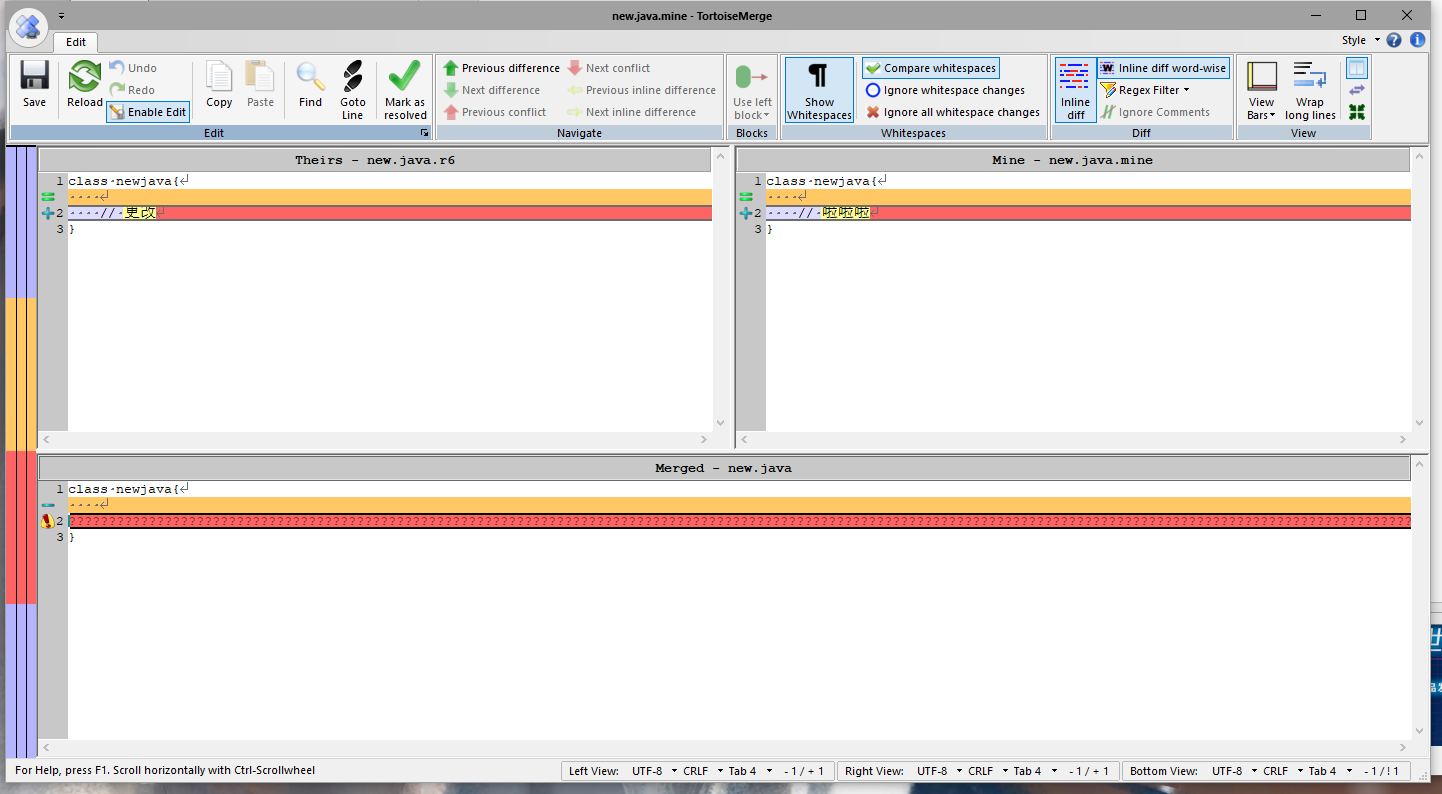
在svn中选中编辑错误：



这里给出了当前svn版本的代码信息和要提交的版本代码信息！

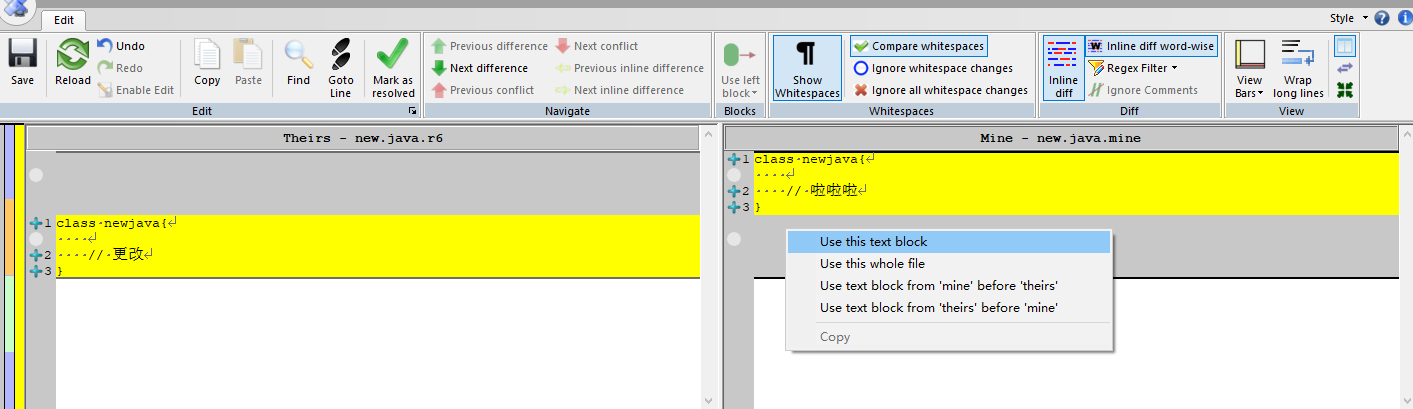
左边是服务器的代码，右边是我们自己的代码

下面是合并后的代码

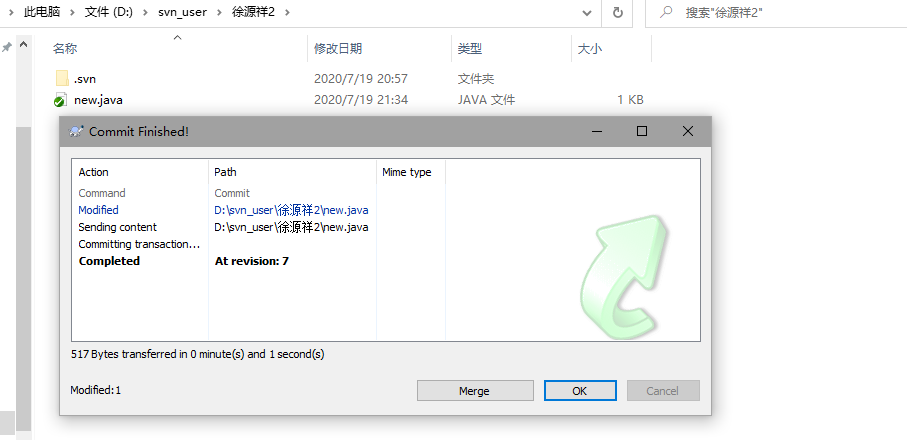


我们可以选择将服务器代码上传或者将我们的代码上传或者两者合并上传都可以！

具体情况看自己选择！解决完成后选择绿色对勾标记我们已经解决冲突了。

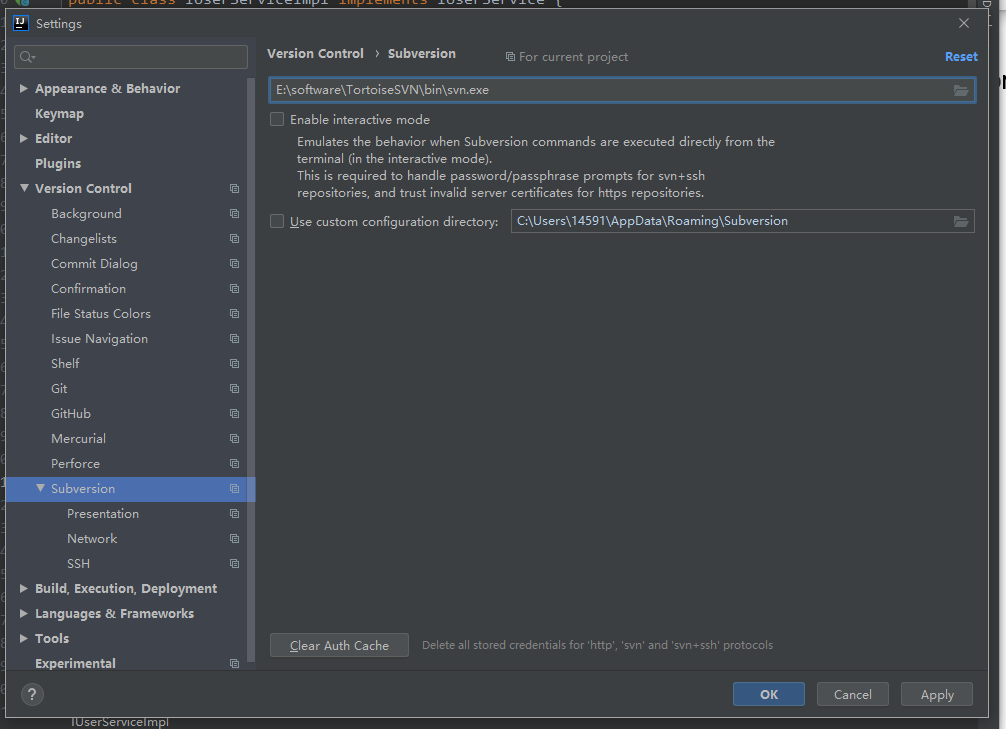


此时在进行上传就可以了：这里上传成功，且版本号自动进行了增长直接跳到了7！

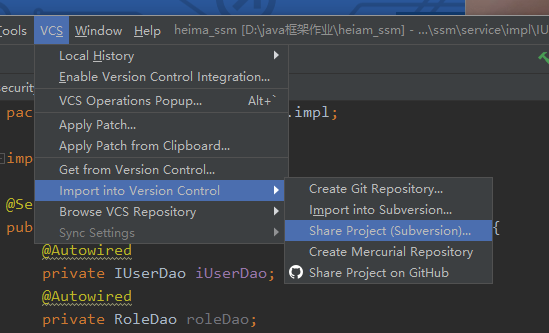


## 五 IDEA下的SVN配置以及share project

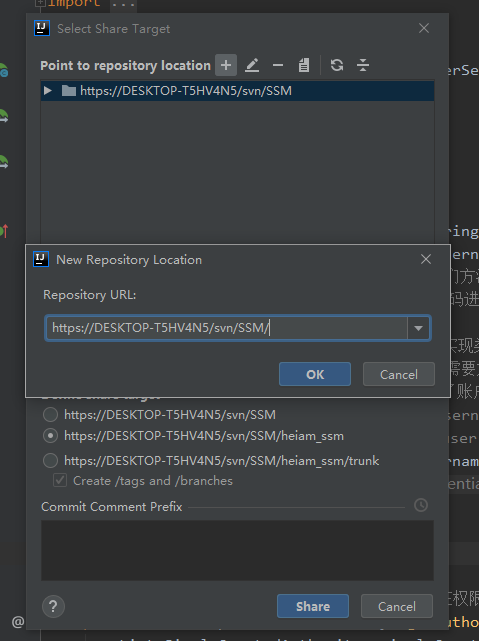
IDEA中做出如下设置：



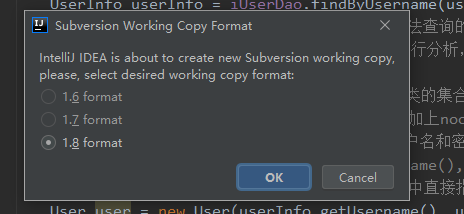
选中vcs进行import这里选择version或者git（git以后学习）



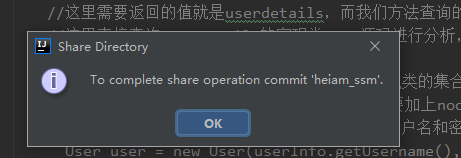
弹出如下界面，见我们的svn服务器仓库url添加进来，选中，在下面算则一个仓库的路径版本！



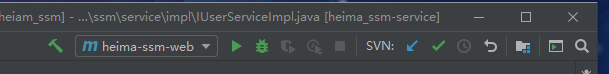
最后选择svn版本，ok进行上传



上传完成：



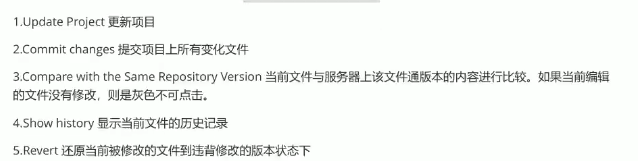
此时ieda上方出现了一系列图标进行checkout以及commit以及update等等操作

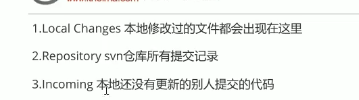


此时进行commit操作就可以将项目上传到svn服务器中了！



以下介绍一些添加svn后出现的功能：后续使用！

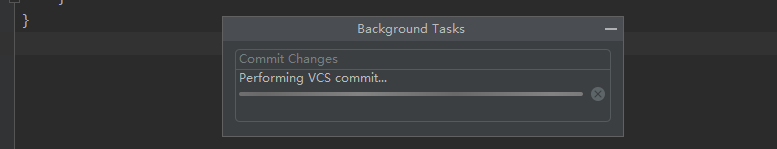




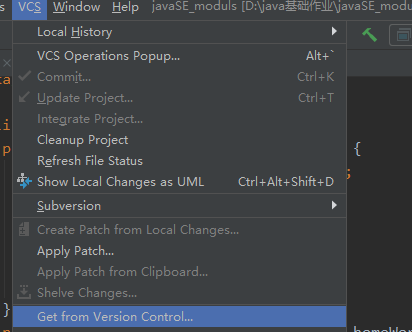
## 六 IDEA下的SVN操作

上面的share project并没有实际的上传我们的项目，只不过是在svn中创建了对应的项目结构。

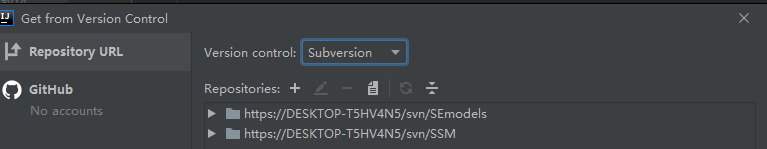
真正的上传项目需要我们手动的commit！弹出如下界面等待上传完成，过程非常慢



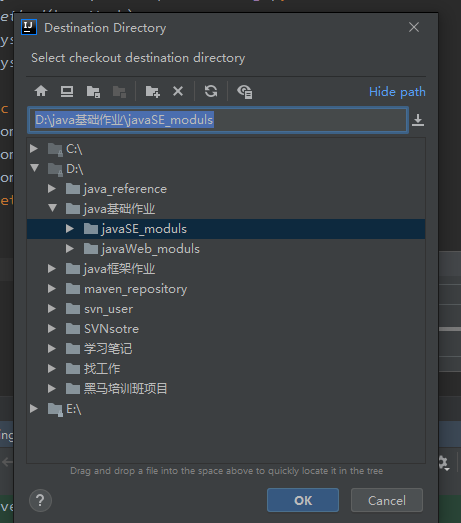
当我们需要从svn中下载项目到ieda中的时候可以选择get from version control，选择git 或者svn



在这里选择我们想要get的项目



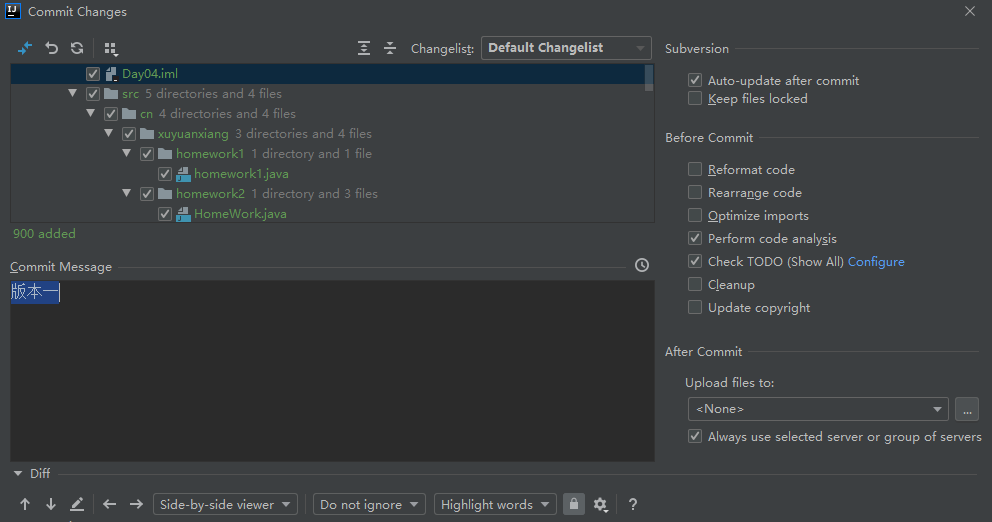
在最后选择get下来的项目的存放路径：



当然当我们svn中项目发生了改变，本地项目可以采用update和svn中的版本进行同步！

以上所有的提交，更新，get等操作在version control中的repository中都可以查看！

当然我们在提交项目的时候没必要将所有的文件全部提交，部分属于idea的功能性文件没有必要提交给svn，所以在提交的时候会有ignore选项帮助我们忽略部分文件：



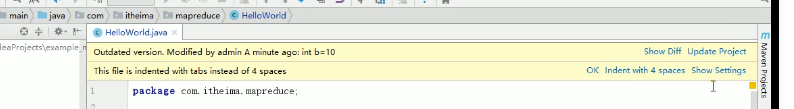
最后，idea中也存在我们之前描述的冲突，当多个人同时修改同一个版本的时候会发生异常！

Idea中帮我们提供了类似的解决方法：

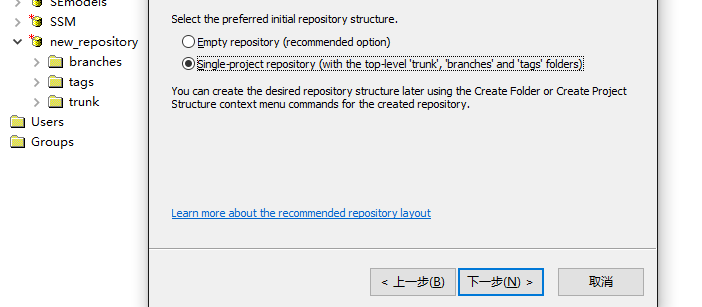
点击show diff查看区别

点击update project后选择merge来更新版本，至于如何更新，根据提示进行操作！

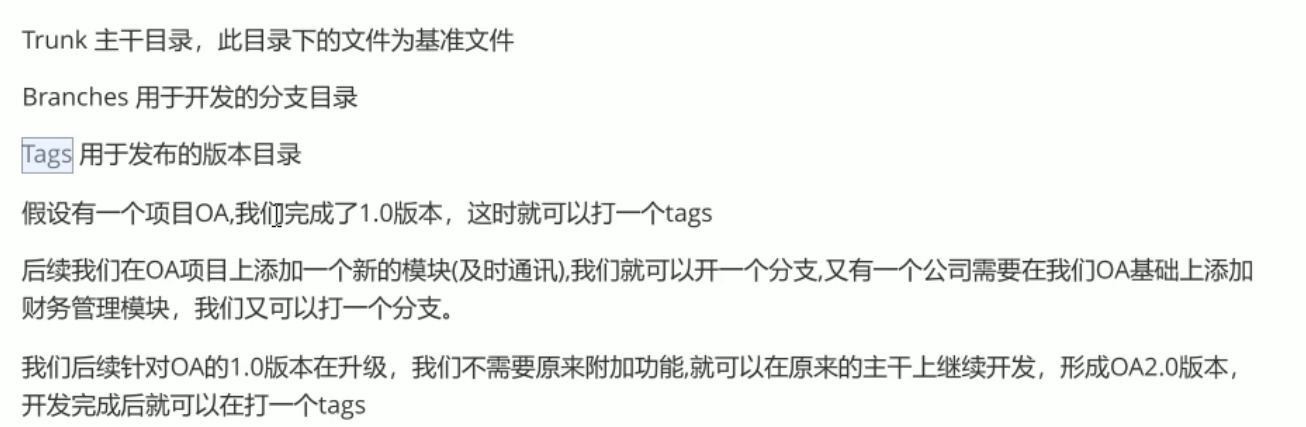
当然最好的方式就是避免产生这样的异常，我们每次进行更新之前进行update将代码进行同步！



## 七 SVN目录结构以及作用



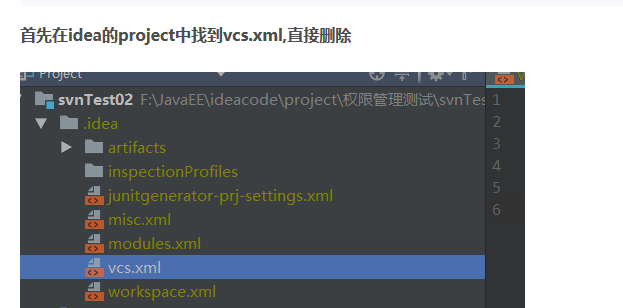
我们完成1.0版本后将代码打包到tags中形成一个版本，给客户进行使用，如果客户提出其他方面给的需求，我们就需要在分支中补充代码，然后合并到主干，再打一个包到tags中形成另一个版本。



Svn对分支以及版本命名有自己的规范！



这里发现一个问题：当我们创建的idea项目点击vcs没有发现share project的时候需要如下操作：

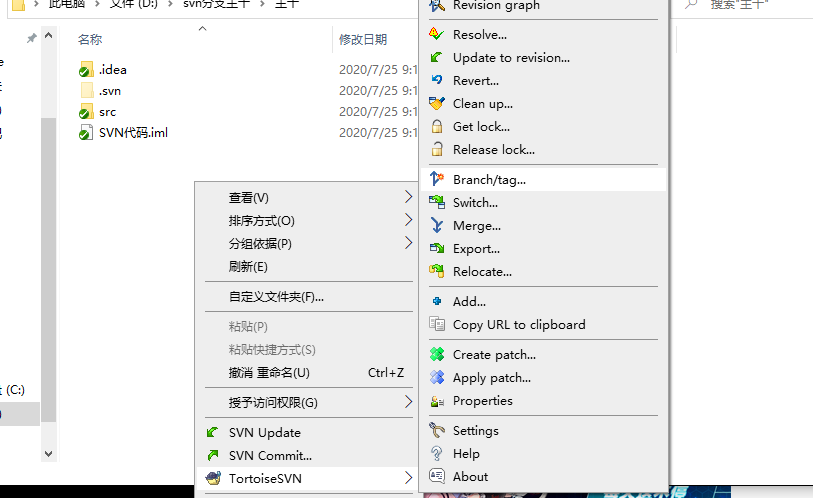


下面讲解如和操作项目的分支和主干：

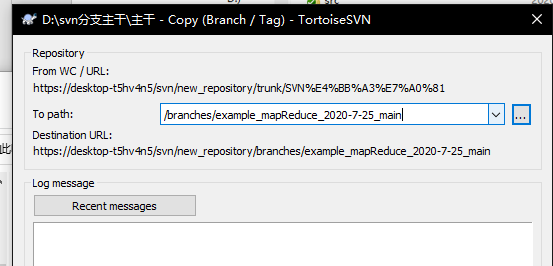
首先创建一个文件夹，命名主干，将我们项目中truck中的内容check out的让内容拉取下来！



然后选择branch/tag

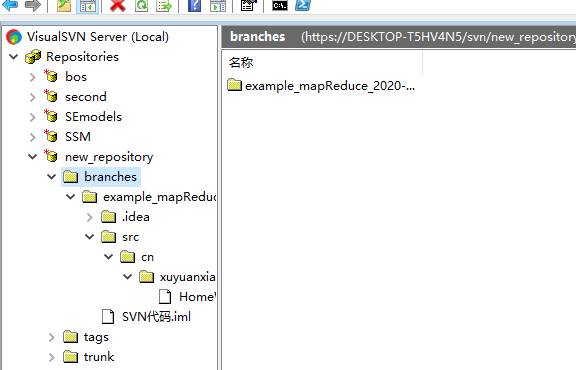


在这里填写好我们要打包到的分支，这里的命名需要规范！



这样就将我们的主干代码提交到了分支部分！

查看svn服务器发现的确成功了！



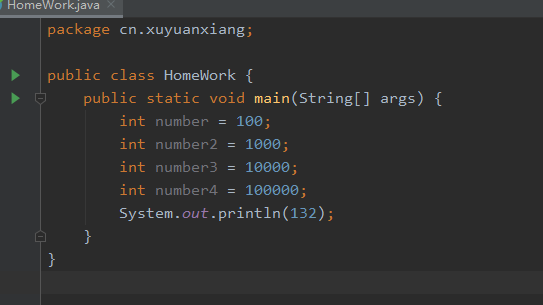
打完之后我们主干和分支可以合并：

首先要将我们的主干update保证最新！

因为现在我们主干分支代码一样，所以合并也没效果。

我们先修改代码：

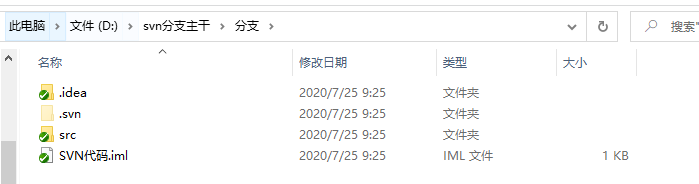
增加了sout方法:



Commit，在文件主干中点击update进行更新

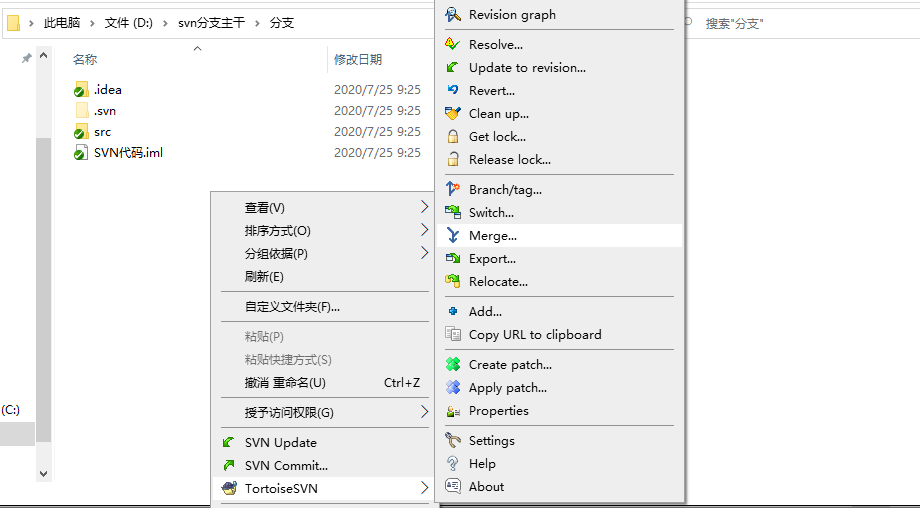
此时主干的代码已经修改

我们在创建一个分支文件夹，将我们svn服务器中分支中的代码拉去下来(这里是旧版本的代码)。

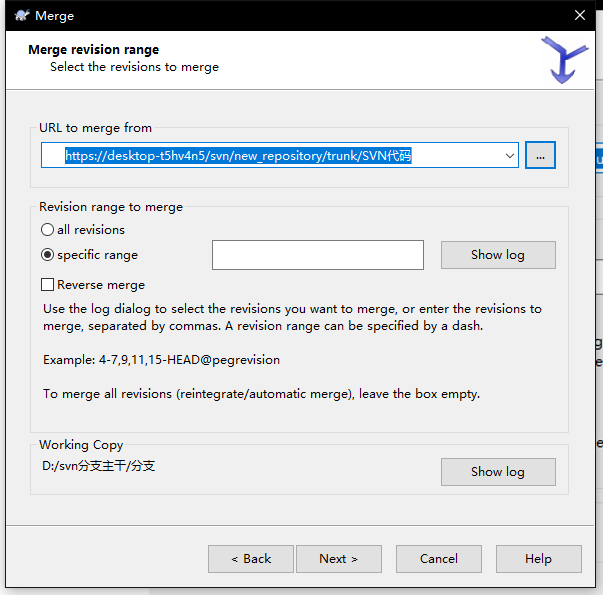


下面演示主干合并到分支：这里再次强调需要我们将主干进行update保证最新：

在分支部分进行merge

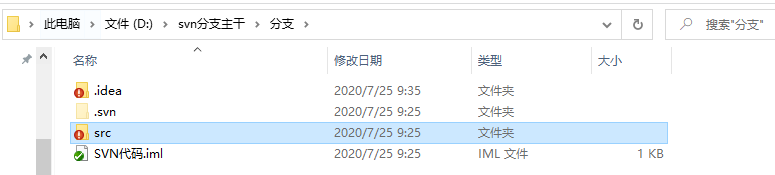


这里选择我们的主干和分支的路径：



此时在打开文件夹以及文件发生已经产生了变化：

文件夹有红色标记，打开发现代码中已经多了sout！



分支合并到主干同理。

需要注意的是我们在合并的时候不可避免的会产生一些冲突，这个时候需要我们进行手动解决！