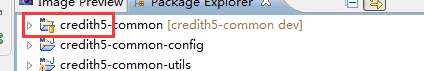
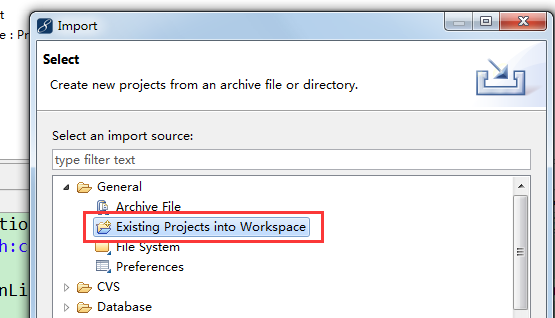
1.Myeclipse导入已存在的git仓库中的项目，使其关联

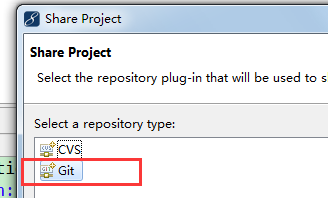
1. 本地已有的git仓库中的项目，想把它导入MyEclipse中，并使其关联（就是有仓库的小图标）

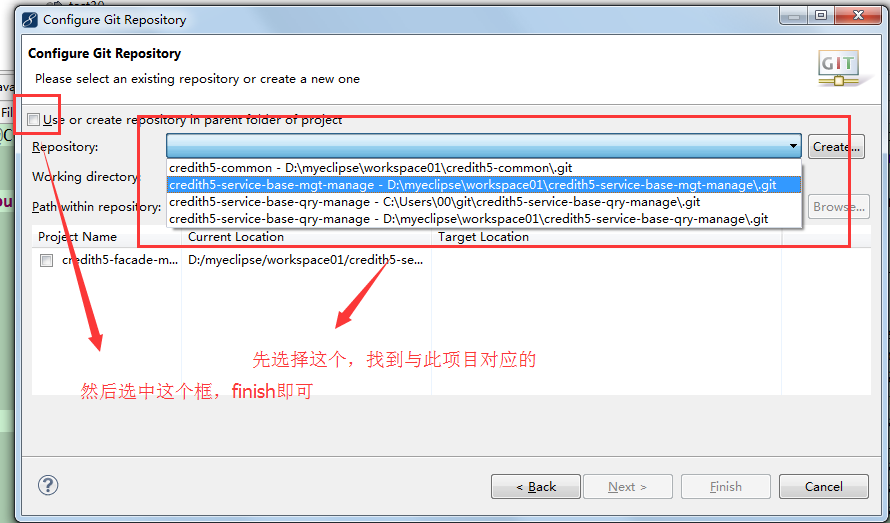


1. 先这样直接导入项目即可（最普通的方式导入）



1. 然后右键导入的项目 Team🡪Share Project..-->选git 点击next （如下图）



1. 按照图中步骤操作即可
2. 

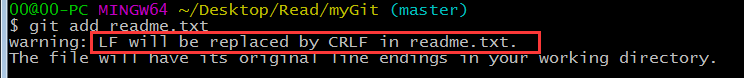
# 2.git 命令

git stash –提交的时候先隐藏本地文件

git pull

git stash pop –将隐藏的显示出来然后看文件

通过git init命令把这个目录变成Git可以管理的仓库：

1. git status --查看仓库状态，看看那些文件修改了
2. git log --查看操作日志
3. git reflog --查看所有的操作日志，可以用于回到未来的版本
4. git commit -m “对提交的描述”
5. git push --推送到远程
6. git diff --查看不同
7. git reset --hard 后面跟commit id 前几位即可 --重置为某个版本
8. git merge命令用于合并指定分支到当前分支
9. 查看分支：git branch
10. 创建分支：git branch <name>
11. 切换分支：git checkout <name>
12. 创建+切换分支：git checkout -b <name>
13. 合并某分支到当前分支：git merge <name>
14. 删除分支：git branch -d <name>
15. 当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。
16. 用git log --graph命令可以看到分支合并图。
17. 合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。
18. 如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除。
19. 现在，你的小伙伴要在dev分支上开发，就必须创建远程origin的dev分支到本地，于是他用这个命令创建本地dev分支：
20. $ git checkout -b dev origin/dev
21. git pull也失败了，原因是没有指定本地dev分支与远程origin/dev分支的链接，根据提示，设置dev和origin/dev的链接：
22. $ git branch *--set-upstream dev origin/dev*
23. 如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream branch-name origin/branch-name。这就是多人协作的工作模式，一旦熟悉了，就非常简单
24. 要关联一个远程库，使用命令git remote add origin [git@server-name:path/repo-name.git](mailto:git@server-name:path/repo-name.git)；
25. 关联后，使用命令git push -u origin master第一次推送master分支的所有内容
26. 此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改
27. git push origin local\_branch:remote\_branch 例：将本地新建的分支dev推送到远程，如果远程没有此分支就新建一个dev（用此让远程建立分支）
28. 这个操作，local\_branch必须为你本地存在的分支，remote\_branch为远程分支，如果remote\_branch不存在则会自动创建分支。
29. 
30. 

**以上问题解决办法：git config core.autocrlf false**

1. 删除远程分支
2. git branch -r -d origin/dev
3. git push origin :dev

&lt; 是< [less](javascript:void(0);) [than](javascript:void(0);)

&gt; 是> [greater](javascript:void(0);) [than](javascript:void(0);)

**小技巧： 创建.gitignore文件**



## 2.1自己测试git心得：

本地新建一个git仓库，想把远程git库的内容pull到本地的git库，不使用clone

首先本地新建一个git仓库，然后使用命令 git remote add origin [git@server-name:path/repo-name.git](mailto:git@server-name:path/repo-name.git) 连接远程库，此时如果git pull的话还是不行的，还要再指定一下分支的连接使用命令git branch --set-upstream branch-name origin/branch-name 然后就可以了

# 3.github账号

guomaoyangeheh

[229406891@qq.com](mailto:229406891@qq.com)

886xiaolaji

# 4.开发问题

1. <http://localhost:8080/credith5-web-boss/swagger/index.html> **swagger地址**

**2.** <http://blog.csdn.net/linlzk/article/details/50728264> **swagger 整合springmvc 教程**

**Git release 111**

**3.配置tomcat配置文件解决get乱码问题**



4.单元测试时要使用这个东西判断，（涨涨姿势）

# 5.当git识别不了新增文件时

方法：

git add 1.cpp

返回无法添加，说1.cpp已经处于.gitignore中。但是iganore中明明没有忽略这些cpp文件。

解决方法：

git add -f 1.cpp

这样就可以将1.cpp添加到git的版本控制中。

如果直接git add -f 1.cpp

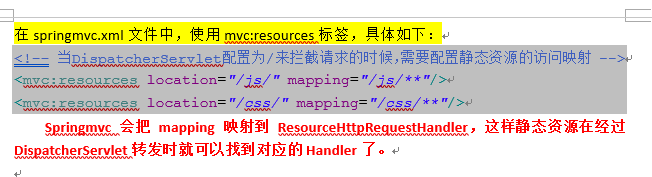
报错，无法找到文件1.cpp。

那么就将1.cpp的完整路径输入即可。比如D:/jingmai\_git/TradePlugin/src/Trade/1.cpp就可以了。

1. 首先先看一下文件中是否忽略掉了该文件，或者用以上方法解决
2. 异常处理，在service层处理异常，抛至controller层，在controller层配置了ExceptionHandler捕获到异常，在web层设置了一个

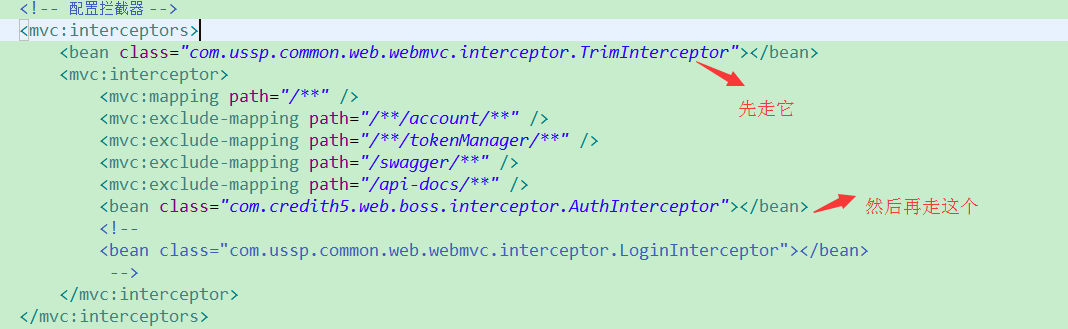
**异常处理思路，dao层抛至service层，service层抛至web层在web层设置异常处理器，所有的处理器都会继承它，通过注解@ExceptionHandler 可以抓取到抛至处理器中的异常，日志输出异常信息，并将错误信息返还至前端。**

**Spring MVC使用：**



**Spring MVC中的拦截器是实现HandlerIntercept接口实现的**





**Public class** HandlerInterceptor1 **implements** HandlerInterceptor{

/\*\*

\* controller执行前调用此方法

\* 返回true表示继续执行，返回false中止执行

\* 这里可以加入登录校验、权限拦截等

\*/

@Override

**Public boolean** preHandle(HttpServletRequest request,

HttpServletResponse response, Object handler) **throws** Exception {

**O** Auto-generated method stub

**Return false**;

}

/\*\*

\* controller执行后但未返回视图前调用此方法

\* 这里可在返回用户前对模型数据进行加工处理，比如这里加入公用信息以便页面显示

\*/

@Override

**Public void** postHandle(HttpServletRequest request,

HttpServletResponse response, Object handler,

ModelAndView modelAndView) **throws** Exception {

}

/\*\*

\* controller执行后且视图返回后调用此方法

\* 这里可得到执行controller时的异常信息

\* 这里可记录操作日志，资源清理等

\*/

@Override

**Public void** afterCompletion(HttpServletRequest request,

HttpServletResponse response, Object handler, Exception ex)

**throws** Exception {

}

}

# Checked异常一Unchecked异常(CreditH5项目中异常问题)

除了Error与RuntimeException，其他剩下的异常都是你需要关心的，而这些异常类统称为Checked Exception,至于Error与RuntimeException则被统称为Unchecked Exception.

Error和RuntimeException及其子类是unchecked exception.其他exception是checked exception

Error和RuntimeException属于unchecked（不受检查）异常，其余继承Exception的异常属于checked（受检查）异常

CreditH5项目中遇到的在service层抛异常被Dubbo的ExceptionFilter拦截Controller层抓取到的是时候该异常被包了一层RuntimeException

**详细见：**

<http://blog.csdn.net/mj158518/article/details/51228649>

**御魂**

<http://www.18183.com/yys/201610/725955.html>

# 7.Http协议详解

<http://www.ruanyifeng.com/blog/2016/08/http.html>

## 1. Tomcat Server处理一个http请求的过程

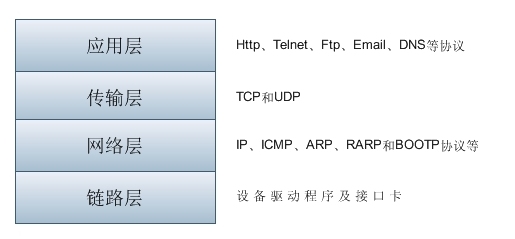
假设来自客户的请求为：  
http://localhost:8080/wsota/wsota\_index.jsp

1. 请求被发送到本机端口8080，被在那里侦听的Coyote HTTP/1.1 Connector获得  
   2) Connector把该请求交给它所在的Service的Engine来处理，并等待来自Engine的回应  
   3) Engine获得请求localhost/wsota/wsota\_index.jsp，匹配它所拥有的所有虚拟主机Host  
   4) Engine匹配到名为localhost的Host（即使匹配不到也把请求交给该Host处理，因为该Host被定义为该Engine的默认主机）  
   5) localhost Host获得请求/wsota/wsota\_index.jsp，匹配它所拥有的所有Context  
   6) Host匹配到路径为/wsota的Context（如果匹配不到就把该请求交给路径名为""的Context去处理）  
   7) path="/wsota"的Context获得请求/wsota\_index.jsp，在它的mapping table中寻找对应的servlet  
   8) Context匹配到URL PATTERN为\*.jsp的servlet，对应于JspServlet类  
   9) 构造HttpServletRequest对象和HttpServletResponse对象，作为参数调用JspServlet的doGet或doPost方法  
   10)Context把执行完了之后的HttpServletResponse对象返回给Host  
   11)Host把HttpServletResponse对象返回给Engine  
   12)Engine把HttpServletResponse对象返回给Connector  
   13)Connector把HttpServletResponse对象返回给客户browser

**Tomcat的web.xml文件是所有的WEB APP共用的部署配置文件，**

**每当一个WEB APP被DEPLOY，该文件都将先被处理，然后才是WEB APP自己的/WEB-INF/web.xml-->**

## 2网络协议总结：



<https://my.oschina.net/liting/blog/402795>

1. 物理层
2. 数据链路层
3. 网络层
4. 传输层‘
5. 会话层
6. 标示层
7. 应用层’

## TCP协议的3次握手和4次握手：

<http://blog.csdn.net/whuslei/article/details/6667471/>

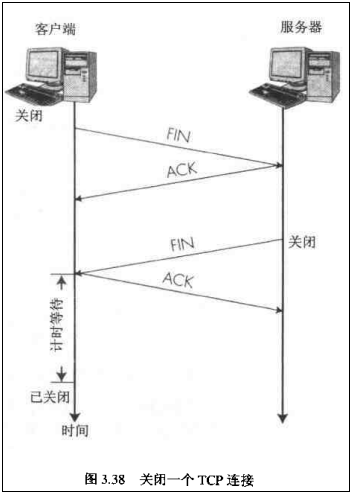
TCP建立连接需要三次握手

Client---》请求服务器建立连接 第一次握手

Server----》授予连接分配资源 第二次握手

Client-----》确认 （TCP连接建立）第三次握手

TCP关闭连接的4次握手(以下):



假设Client端发起中断连接请求，也就是发送FIN报文。Server端接到FIN报文后，意思是说"我Client端没有数据要发给你了"，但是如果你还有数据没有发送完成，则不必急着关闭Socket，可以继续发送数据。所以你先发送ACK，"告诉Client端，你的请求我收到了，但是我还没准备好，请继续你等我的消息"。这个时候Client端就进入FIN\_WAIT状态，继续等待Server端的FIN报文。当Server端确定数据已发送完成，则向Client端发送FIN报文，"告诉Client端，好了，我这边数据发完了，准备好关闭连接了"。Client端收到FIN报文后，"就知道可以关闭连接了，但是他还是不相信网络，怕Server端不知道要关闭，所以发送ACK后进入TIME\_WAIT状态，如果Server端没有收到ACK则可以重传。“，Server端收到ACK后，"就知道可以断开连接了"。Client端等待了2MSL后依然没有收到回复，则证明Server端已正常关闭，那好，我Client端也可以关闭连接了。Ok，TCP连接就这样关闭了！

## 连接服务的ip地址和用户名密码，linux命令：

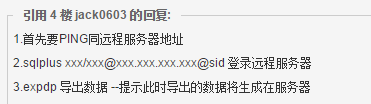
Ip: 172.16.4.200

用户名： user

密码：123456

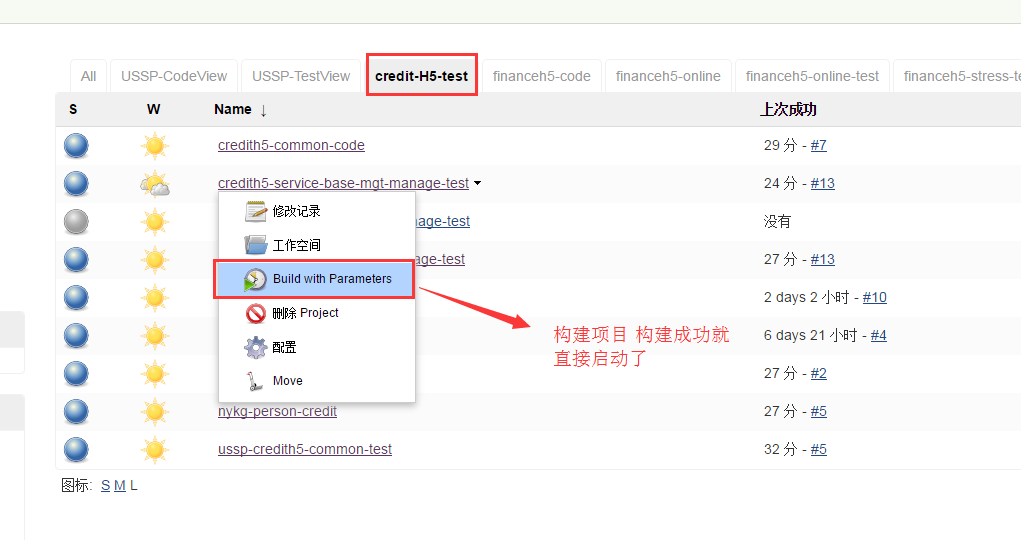
**View命令下的操作：**

Ctrl+d：向下翻半半页pgmb2  
Ctrl+u：向上翻版半页hxfRFiF  
Ctrl+f：向下翻一页E#XN5  
Ctrl+b：向上翻一页i\*KyVg



**cd credith5-web-boss-tomcat/ cd credith5-web-boss**

## 5.Jenkins构建启动项目步骤：

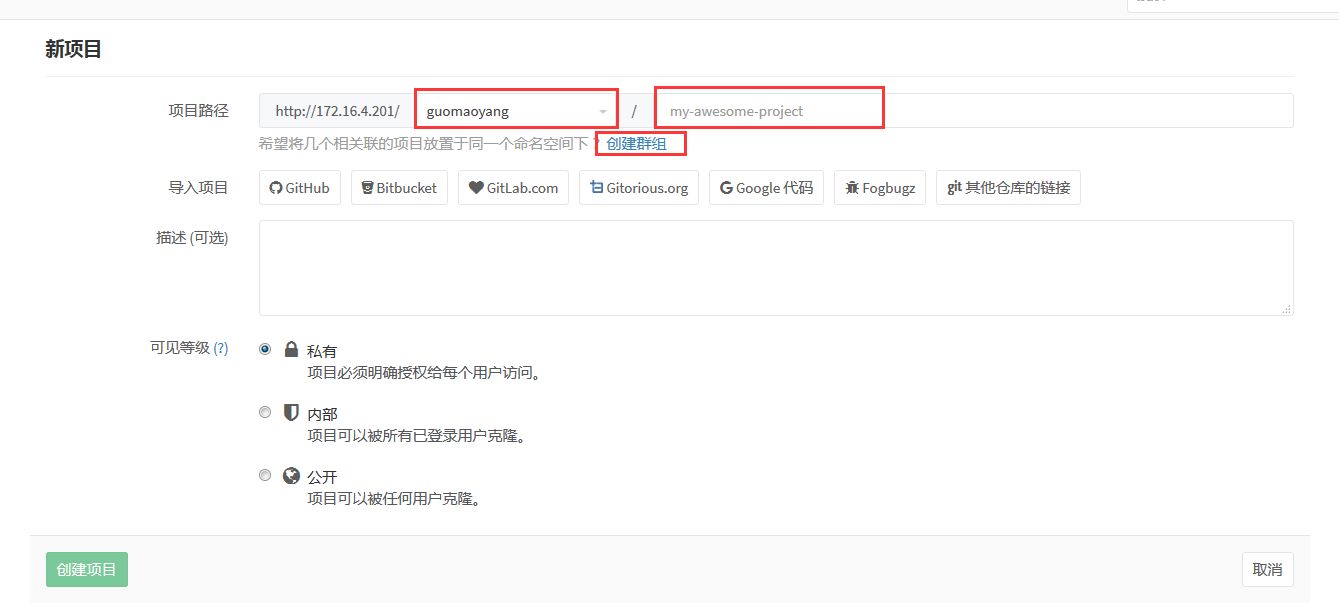
  
**注意：不要只单独起一个容易出错，重启的话要按顺序把项目都要重启完**

# 定时任务：

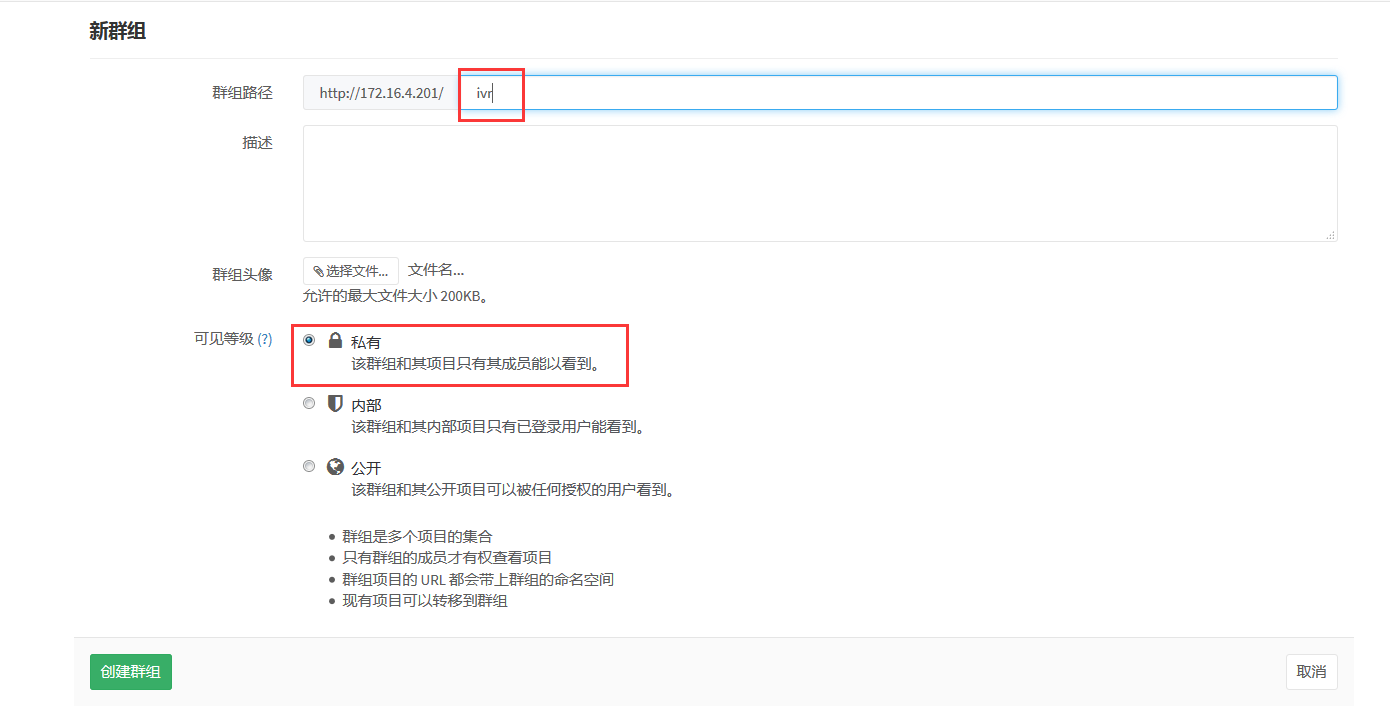
<http://gong1208.iteye.com/blog/1773177> 自动任务博客地址(介绍了3种实现)

# 本地新建maven工程然后推送到远程步骤

9.1本地就直接正常建工程，不需要有git仓库，工程建完后



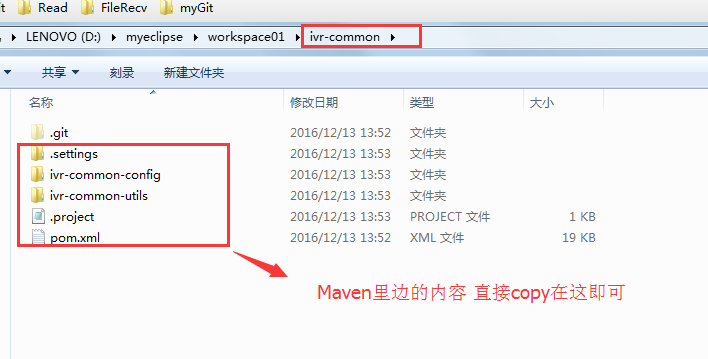
在git上建立远程的仓库，群组可自定义(看红框)



群组可以自定义授权（自己摸索即可）

9.2建立完远程库之后然后直接Clone到本地即可，然后只是一个空的git库

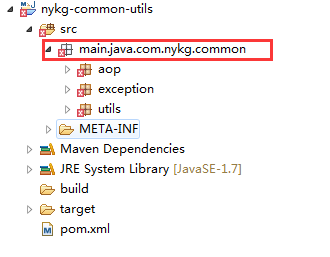
然后把maven工程里的红框标记的内容（应该是所有内容）copy于此即可，然后就直接提交到远程就行了！



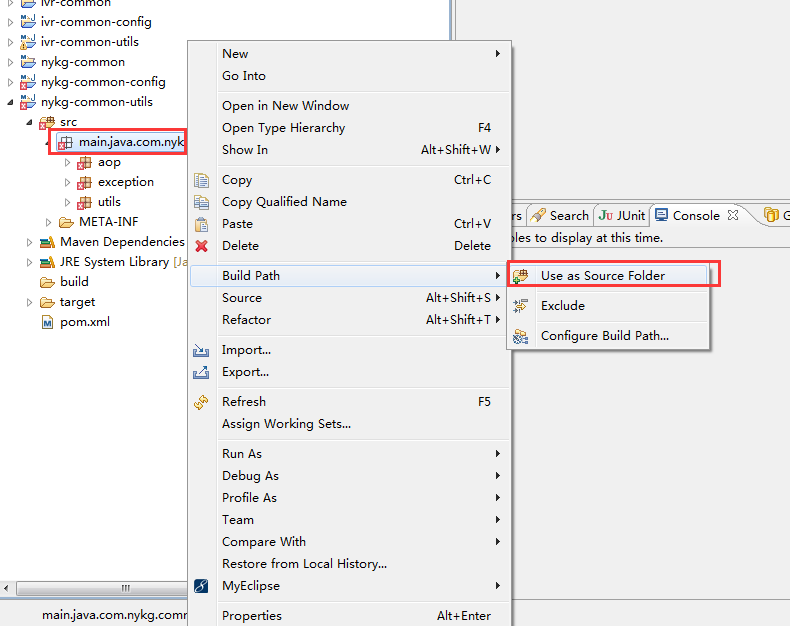
# 10,呼叫中心项目问题

10.1 户口所在地和现住宅地址信息是否是拼接的？

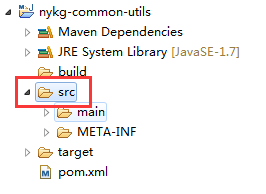
# 11.导入新建项目出现目录结构错误问题：



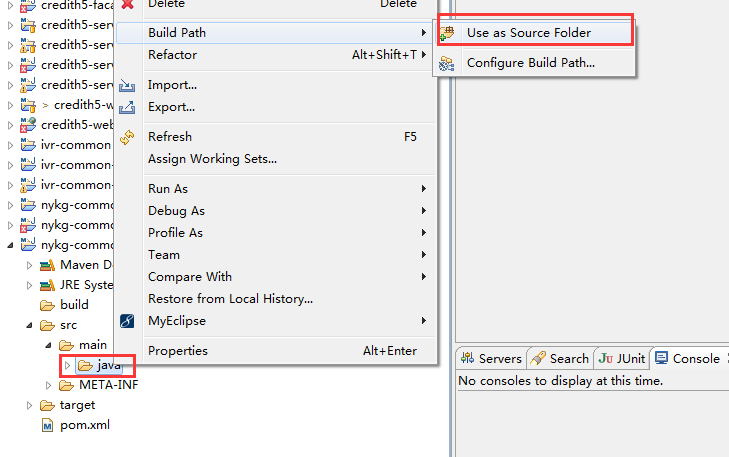
解决方法为：（**截图有点小毛病，应该右键src 注意一下**）



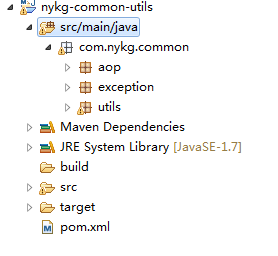
然后就变成了这个样子:



再然后：

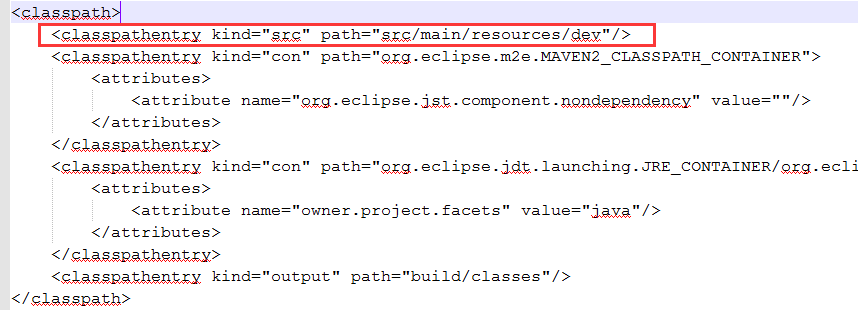


就ok了：



注：

其实这个sourceFolder 就是在.classpath中做了这样一个操作



也就是在构建项目的时候构建这个里边的东西.

# 12.在日志中搜索关键字使用”/”或者”?”：

不过没关系，你在命令模式下敲斜杆( / )这时在状态栏

（也就是屏幕左下脚）就出现了 “/” 然后输入你要查找的关键字敲回车就可以了。

如果你要继续查找此关键字，敲字符 n 就可以继续查找了。

**注：**

**也可以使用？加关键字，使用 ” / ”的时候是向下搜索，使用 ” ? ”的时候是向上搜索(个人使用得知)**

# 13.git无法提交解决办法：



# 14.[Spring中@Autowired注解、@Resource注解的区别](http://blog.csdn.net/gst6062825/article/details/8765157)

<http://blog.csdn.net/gst6062825/article/details/8765157> 博客url

# 15.单元测试相关，重点记住一个结论：

<http://blog.csdn.net/javaloveiphone/article/details/52243984> 单元测试博客url

***重点结论：***

**单元测试为什么还要单独加载spring容器，因为web主项目的spring容器对单元测试是隔离的！！！**