# 太极培训

## Git的使用

Repo.taiji.com.cn

Git是版本控制工具

Git的原理

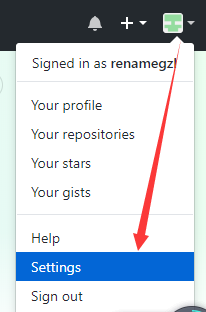
Git的出现&发明

GitHub托管代码

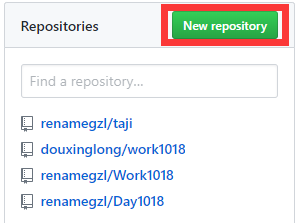
Git和svn的区别

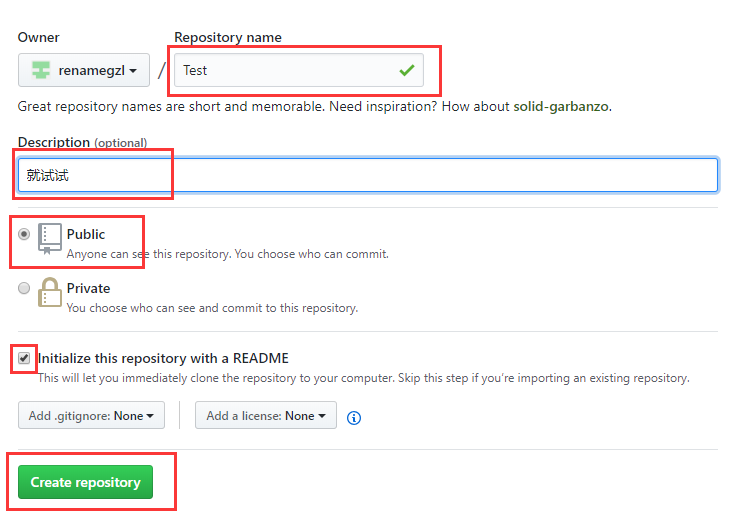
Git分支

设置加密什么乱七八糟的

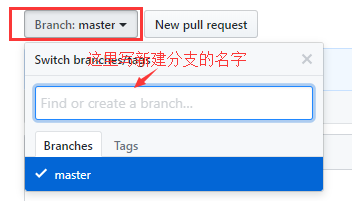


1.创建项目

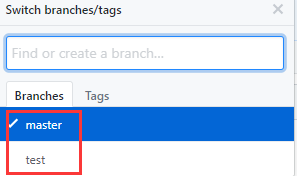




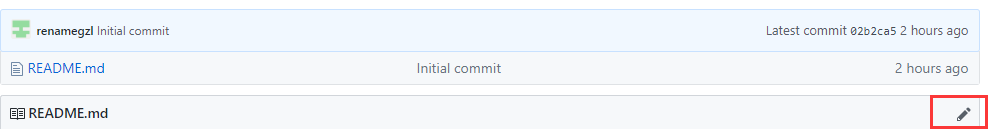
2.创建分支



3.创建完成后的样子

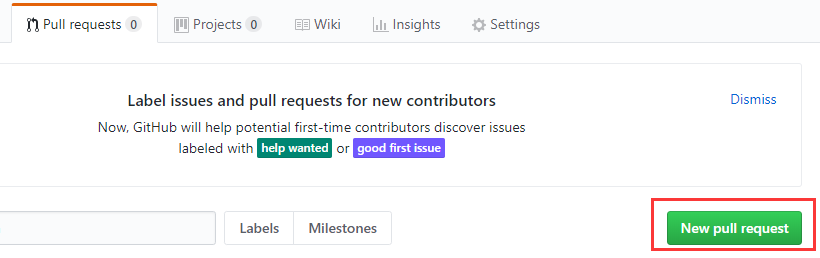


4.进行文件修改

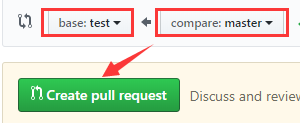


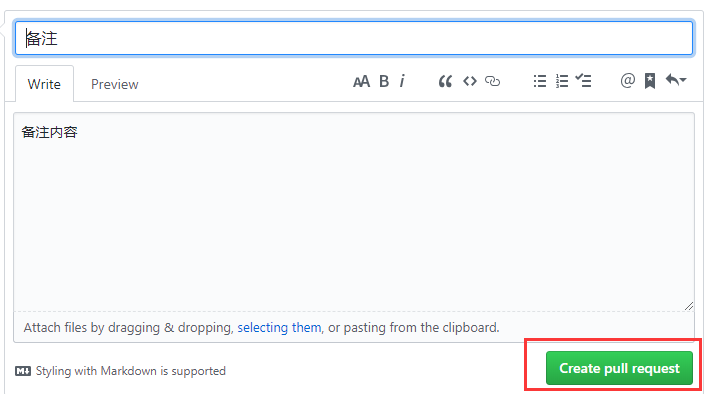
5.分支融合

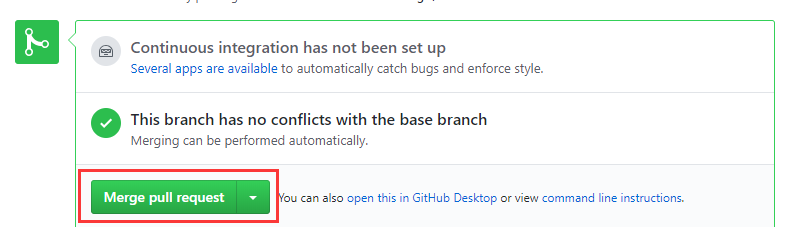


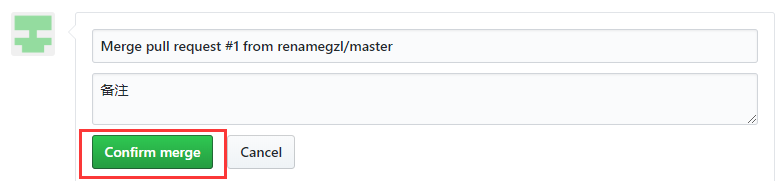


由xxx到yyy





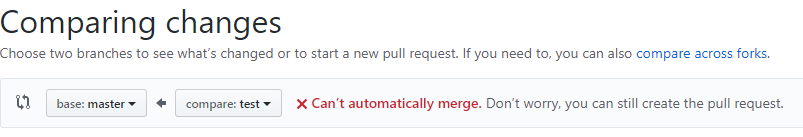




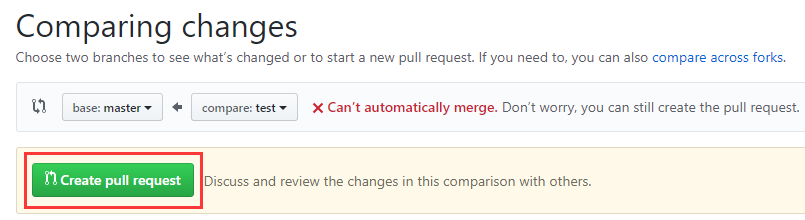
整合完成

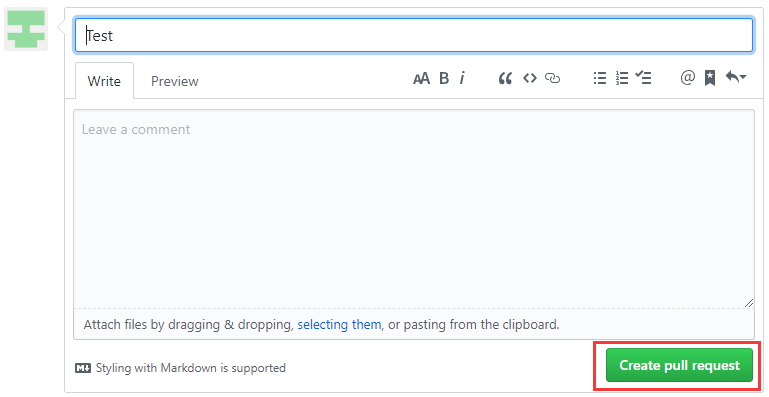
解决分支冲突

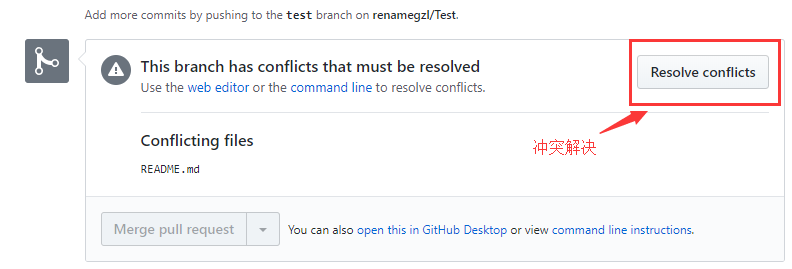
如果遇到下图这种情况，说明两个分支冲突，有不一样的东西。



选择这个

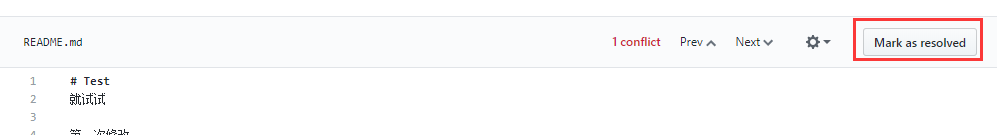






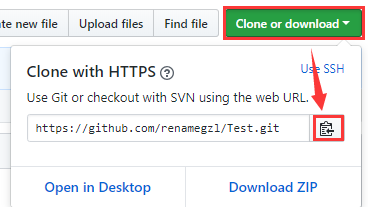
显示矛盾的代码，如果要都保存，就只删除注释

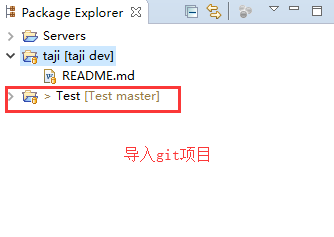




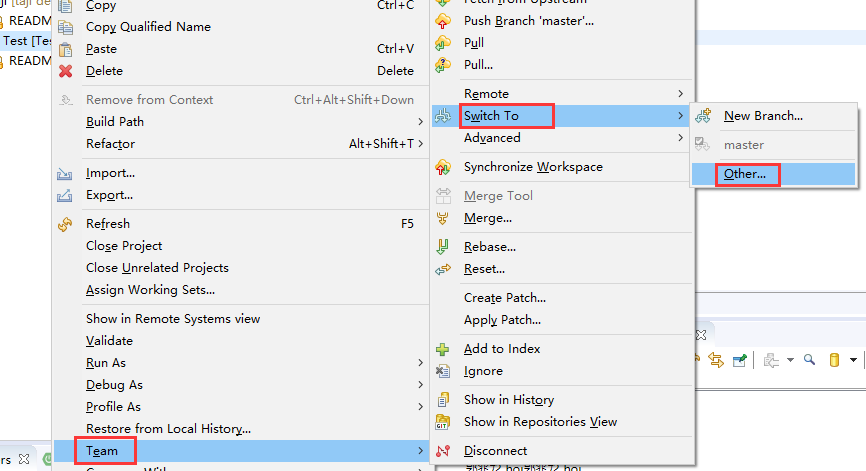


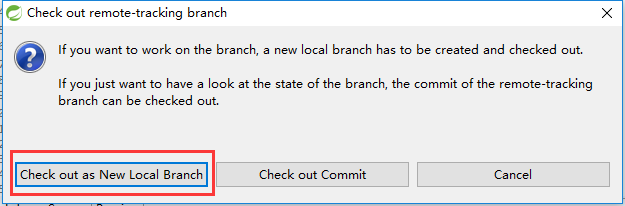
线上线下冲突，获取git地址

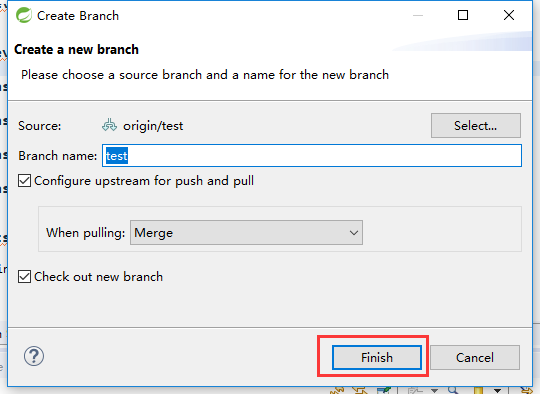




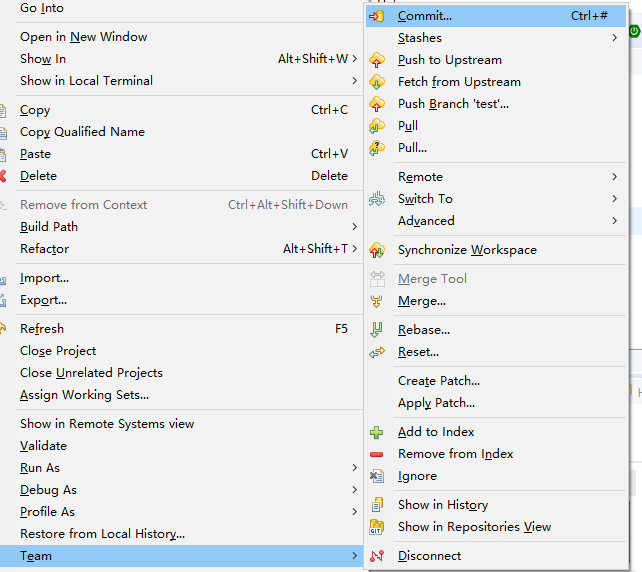
切换分支

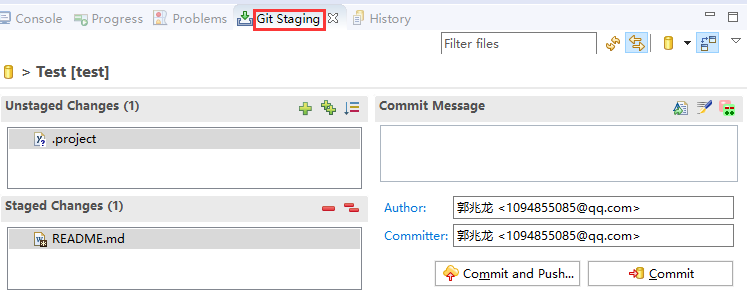




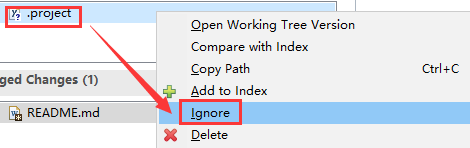


添加以后的上传

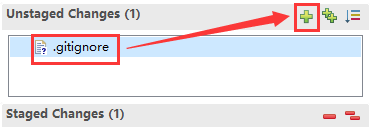




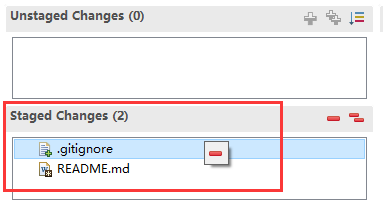
这个文件要忽视，不上传



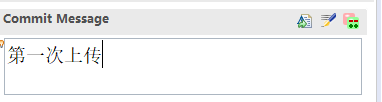
出来的这个是配置文件，要上传

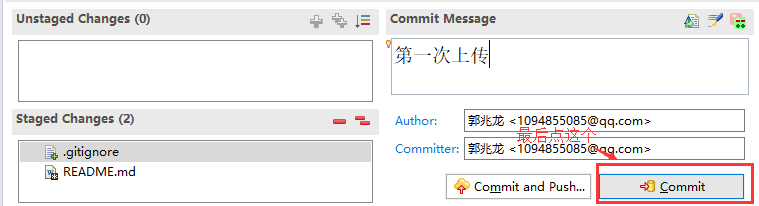


添加后增加到缓存区中

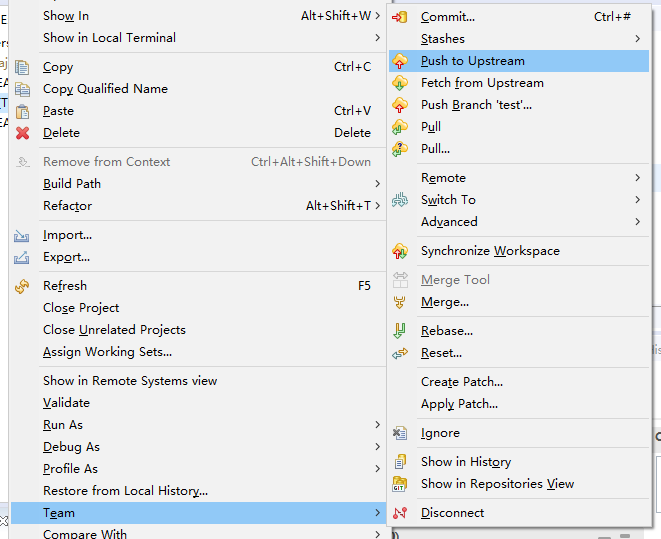


这个信息一定要写



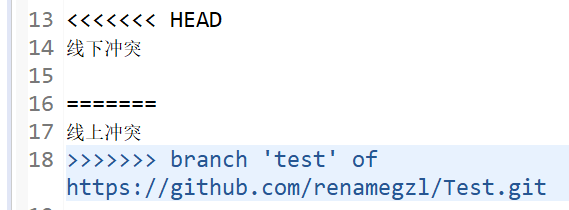


此时项目有向上的箭头，这时要push

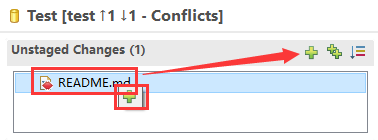
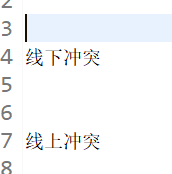


关于冲突的问题

如果有冲突，提交以后不会显示错误，但是仍会有向上的箭头提示没有上传，之后选择pull会报错并进行修改



修改完成后重新提交



此时项目变为：

## Git回档和maven使用

git reset

git reset的作用是还原Index的状态或修改本地分支HEAD的位置。这个命令适合用于从某个提交点重新开新的分支。比如，如果某个提交之后的代码我们都不要了，就可以在本地reset至指定commit，然后开一个新的branch继续新的开发（原来分支在远程上没有发生任何变化的）。

使用git reset是不影响远程分支的，一切都在本地发生。如果回退需要很快影响远程分支的，应该使用git revert。

Git Revert原理：根据你要回退的提交所做的改动做相反的改动，然后重新提交代码，使代码达到没有这些旧提交所能达到的状态。

如果对于git revert的回退策略还没理解，这样讲也许你就能理解了：回退旧的提交必然会导致当前最新代码发生变化，比如之前某个提交加了一行代码，那么回退就是在相同位置减一行代码。Git不会真的把旧提交抛弃，如果直接抛弃，历史记录就追踪不到了。因此，旧提交其实是没动的，Git只是根据旧提交反着做了一遍，这才导致最新的代码发生了改变。只有把revert命令回退导致的改动重新提交，revert命令才算真的完成并生效，否则效果只相当于修改了工作树的文件而已。

注意：revert之后需要push到远程分支上其他用户才能看到回退发生的改动。

使用git revert会出现的问题

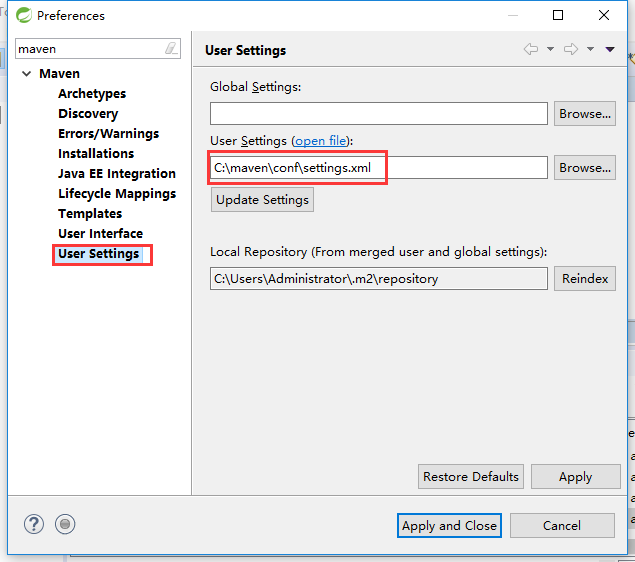
因为git revert是用新提交覆盖旧提交，因此，被覆盖的提交等于不会被采用了。如果两个分支（假设是master和A分支）先合并再用revert回滚，之后又合并（A合并到master），就会发现在master分支上，A分支第一次合并之前的修改大部分不见了。这是因为从时间的发生顺序来看，A分支第一次合并之前的修改发生在revert之前，revert发生在后，而 revert抛弃了A第一合并之前的修改，那么再合并Git就认为你永远抛弃了A第一次之前的修改。要解决这个问题，需要把revert产生的提交再revert一次。

MAVEN

下载、安装、配置环境变量（MAVEN\_HOME和PATH）这些不写了

检查Maven是否成功配置：cmd    mvn –version

STS配置maven



命令行进入项目名，然后测试maven的可行性



这里pom文件中必须有profile标签

若要更改本地仓库，更改位置



在.m2文件内的settings文件为项目配置，权限最高，在这里的xml文件决定了项目的配置，如果这里没有则是全局配置



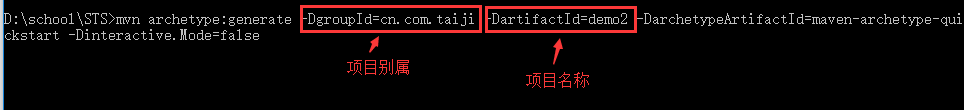
GAV三坐标

G : Group Id （公司或者组织名称，不准确），应该是项目名称

A : Artifact Id 模块名称

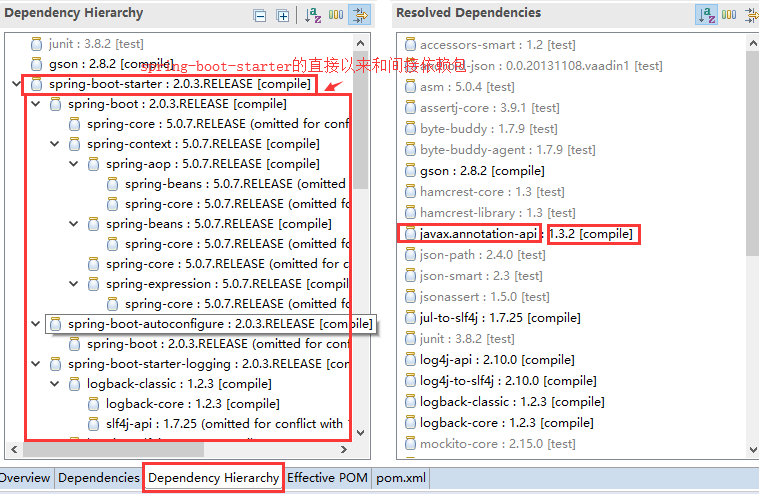
V : Version

使用命令行创建Maven项目

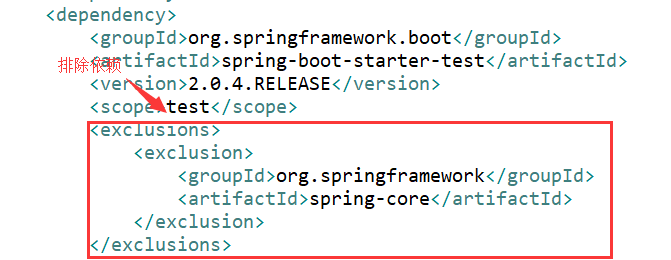


Maven命令行更多命令在此不加赘述

Maven直接依赖和间接依赖，在项目中也许会存在依赖冲突



解决依赖冲突的方法



最重要的：Maven继承和聚合，见网址：

https://www.cnblogs.com/xdp-gacl/p/4058008.html

https://www.cnblogs.com/an5211/p/6618902.html

mvn -Pnexus dependency:resolve 打印出已解决依赖的列表

## 第三天 Spring讲解

阅读开发者文档

三种方式

方法一Xml配置

Bean标签



构造方法注入



属性注入

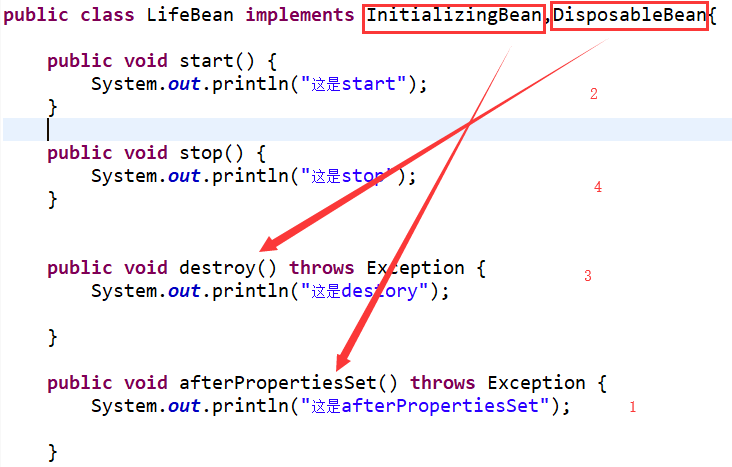


两种方式的依赖注入，构造器是先执行的后执行的是set，尽量少用构造器，在初始化的时候就会启动构造器，避免在初始化的时候进行依赖注入。Set方式在容器一开始的时候不会实例化，

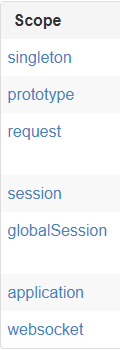
生命周期



这种写法看起来也许很吊，但是会造成耦合，并不推荐



Bean的作用范围



ApplicationContextAware

bean可以通过ApplicationContextAware以编程方式操作创建它们的方法，或者通过将引用转换为此接口的已知子类（例如ConfigurableApplicationContext，公开其他功能）来操作。有时这种能力很有用; 但是，通常你应该避免它，因为它将代码耦合到Spring并且不遵循Inversion of Control样式，[1见开发者文档]

@Required 注释适用于bean属性setter方法但必须填充，否则会报错，并且@option注解可以不填充。Aware 接口是容器提供访问资源的渠道Bean 父类容器的扩展

方法二注解

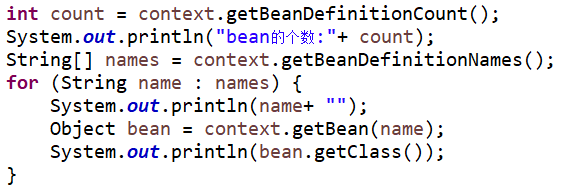
相当于在spring容器中写入一个bean标签

自动装配

按文件目录获取配置文件



打印容器中bean的个数



@Autowired注解可以应用于构造器，set方法，私有属性。

@Primary注解，Autowired按类型加载时如果有多个相同类型，会优先primary

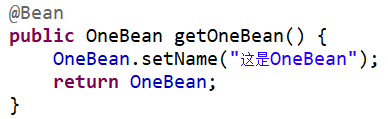
@Resource=@Autowired+@Qualifer

@postConstruct初始化时预先填充

@PreDestory销毁时行动

方法三配置类



设置bean标签

容器一

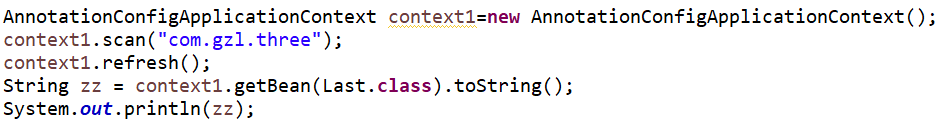


容器二



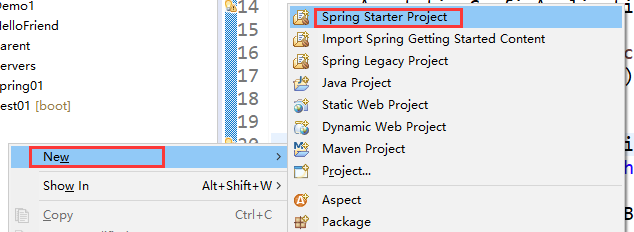
输出：

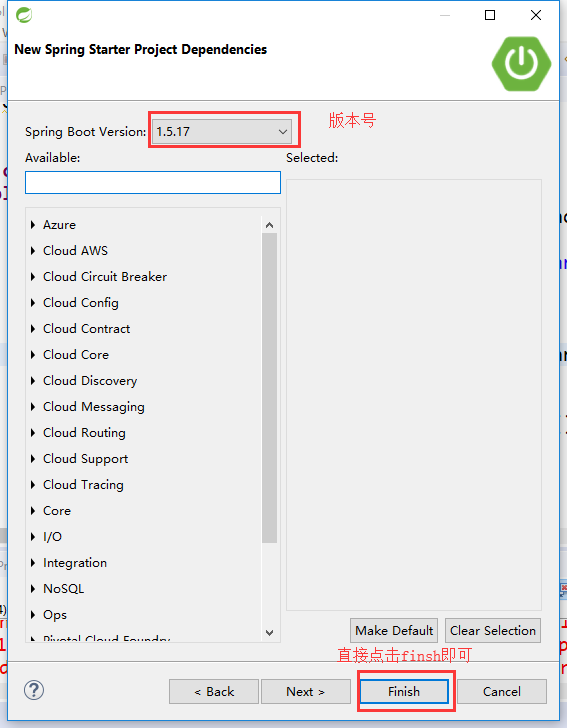
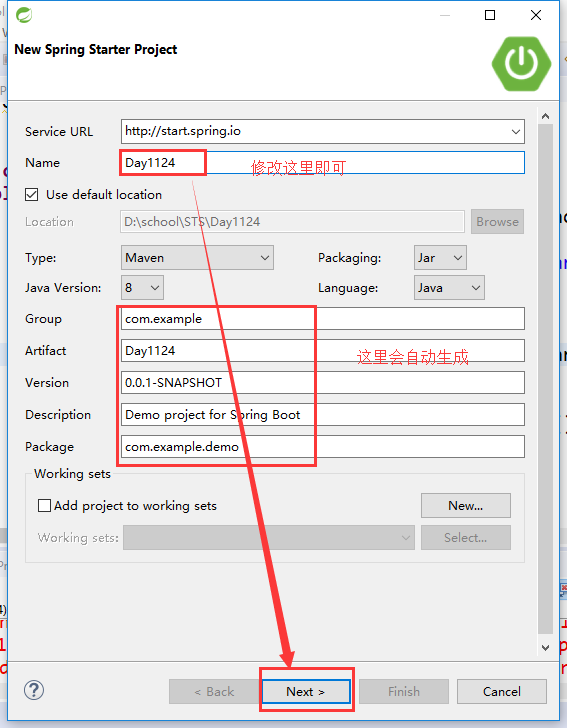




## 第四天 SpringBoot初识

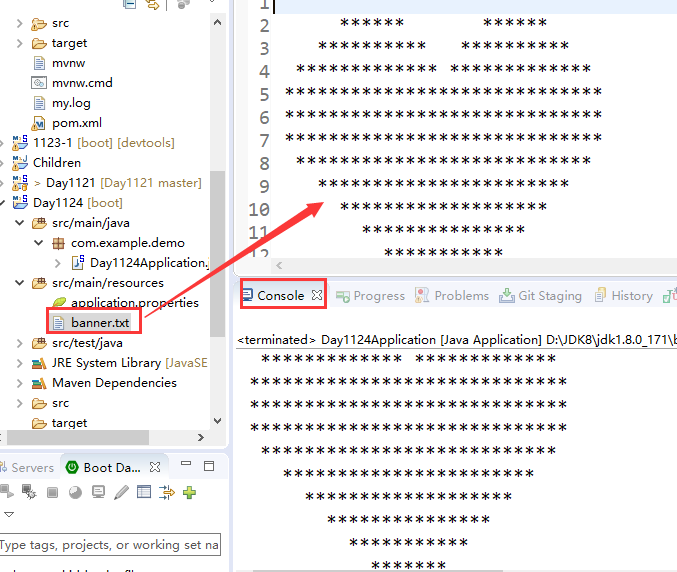
1. 建立项目





注意：application.java扫描本目录及其子目录下的其他文件所有其他文件应在其包下建立。

欢迎横幅



关于@SpringBootApplication

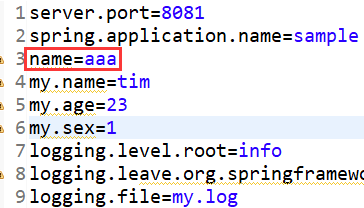


使用springb-boot-devtools可以实现热部署

使用ApplicationRunner或CommandLineRunner



通过配置文件为属性赋值



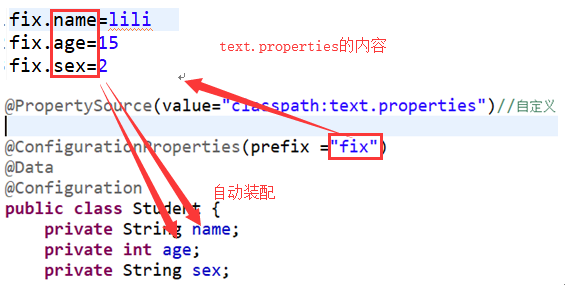


.yml配置高于.propertise

关于占位符的更多扩展：

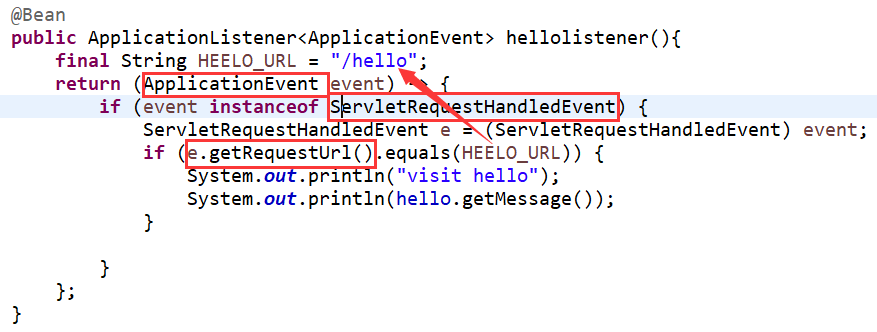


@ConfigurationPropertie



逻辑建立地址

监听器demo，感觉非常有用



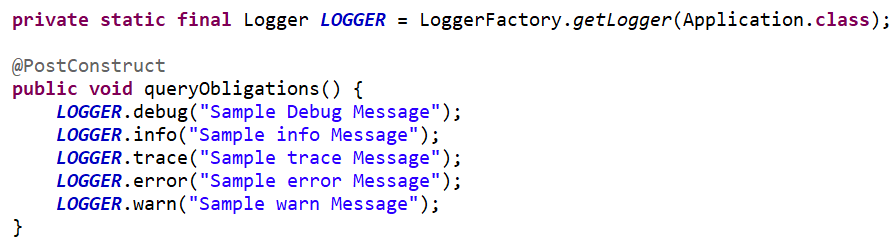
## 第五天 springboot深入





如果没有active，会执行默认配置文件，但现在执行的是hello配置文件

配置日志



在配置文件中：

