# 1、idea下载及安装，并永久激活

备注：2019.6.20

第一种方法：<http://www.downcc.com/soft/400202.html>（推荐）

第二种方法

下载路径：<https://www.jetbrains.com/idea/download/#section=windows>

自己的强迫症导致自己想破解idea为永久激活的，目前在网上能查找到永久激活的版本为Intellij Idea 2018.3.4及以下

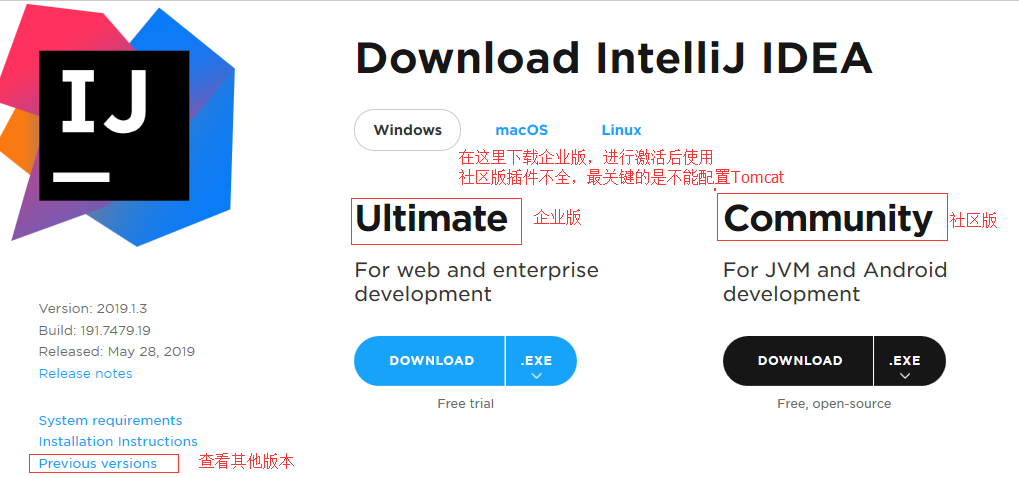


在下载补丁前，最好先将电脑上的IDEA卸载，否则可能出现key无效，破解失败的情况。

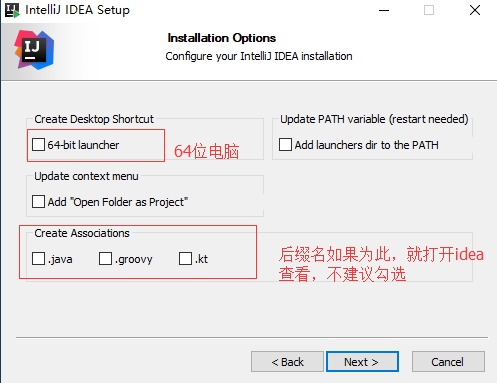
下载补丁JetbrainsIdesCrack-4.2.jar

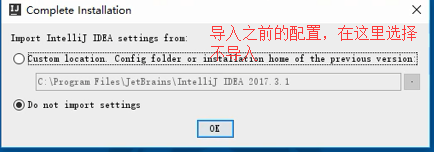
我的云盘‘软件 -- 破解idea方法和步骤明细.zip‘

下载idea的版本需要与激活补丁匹配



安装为机械式安装 不过有一点稍微说明一下：如图



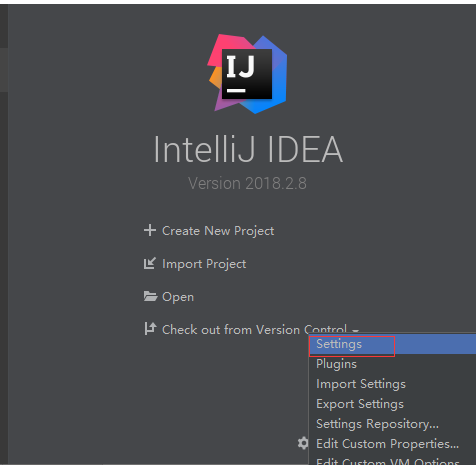


激活方法：<http://www.itzoo.net/other/idea/822.html>

# 2、idea的基础设置

全局配置的两种方式:

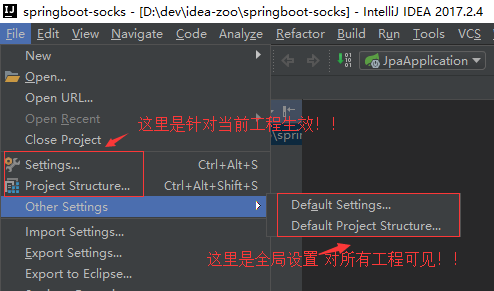
第一种：



第二种

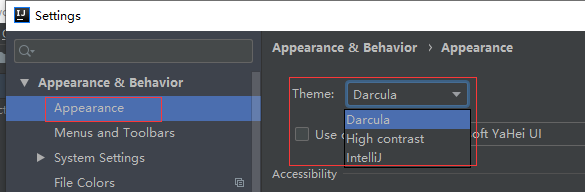
File > Other Settings > Settings for New Projects 这是全局设置

注意如果 File > Settings…只针对的是当前项目的设置，



## 2.1.修改主题背景

Settings > Appearance & Behavior



## 2.2.修改主题字体（默认不修改）

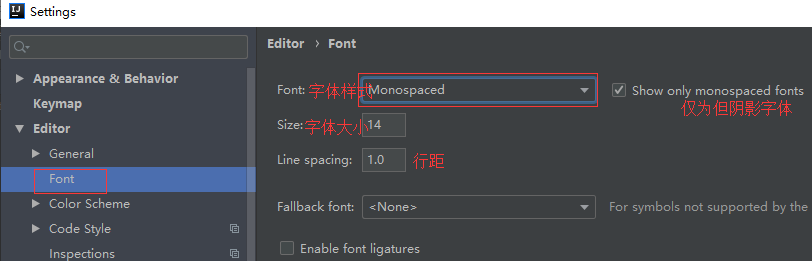
建议这个不修改，选择不对，会出现中英文显示乱码问题

Settings > Appearance & Behavior



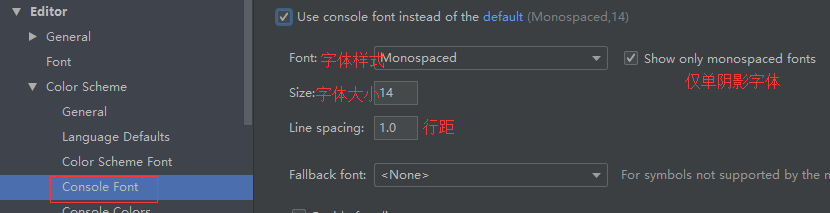
## 2.3.修改编辑字体

Settings > Editor



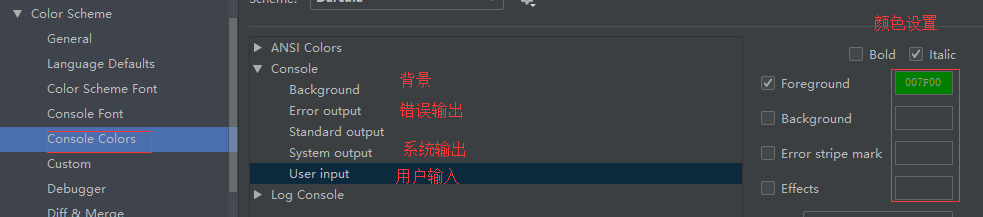
## 2.4.修改控制台字体

Settings > Editor > Color Scheme



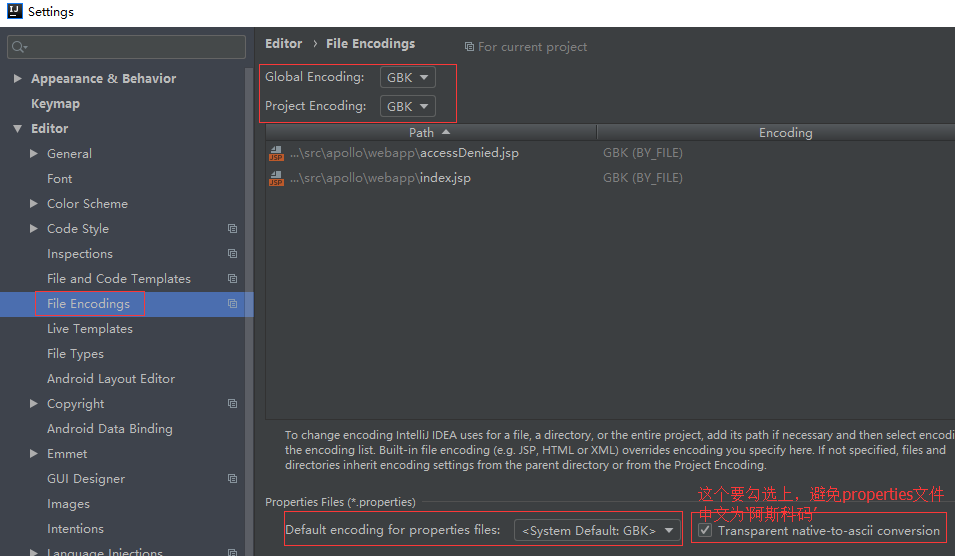
## 2.5.修改控制台不同情况（错误，输入，背景）打印字体颜色

Settings > Editor > Color Scheme



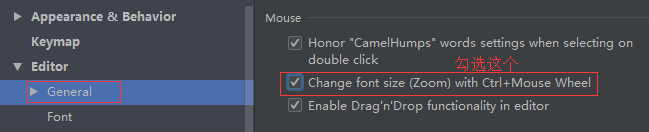
## 2.6.修改编码格式

Settings > Editor



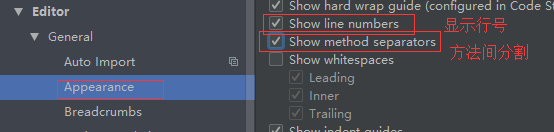
## 2.7.Ctrl + 滚轮 动态设置字体大小

Settings > Editor



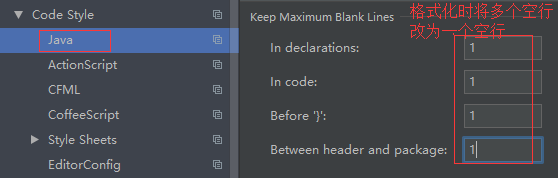
## 2.8.显示行号与方法间分割

Settings > Editor > General



## 2.9.格式化时出现多个空行的统一修改为一个空行

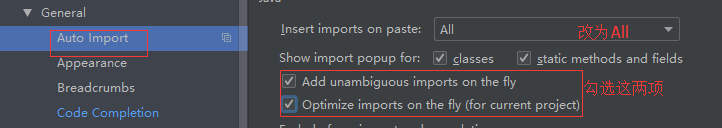
Settings > Editor > Code Style



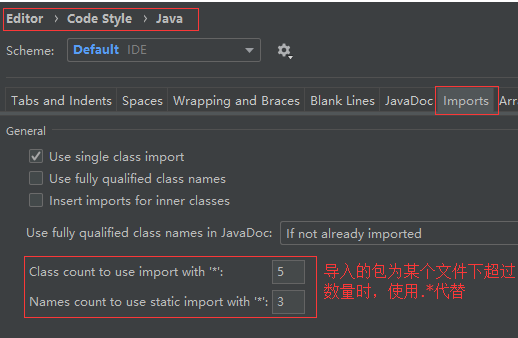
## 2.10.自动导包

自动导入包，并自动去掉一些没有用到的包

Settings > Editor > General

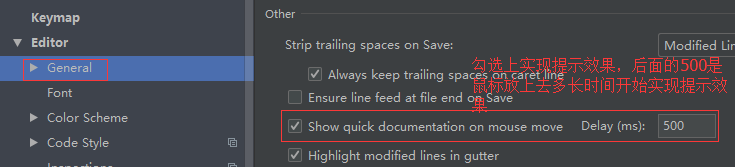


还有就是**优化引入**，假设你引入了一个包下的很多个文件，他就会给你优化成xxx包名.\*。也就是下面这个图的配置啦。



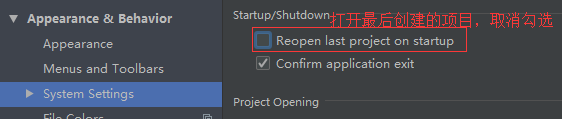
## 2.11.鼠标放在类上或方法名上源码提示

Settings > Editor



## 2.12. 取消打开最后创建的项目

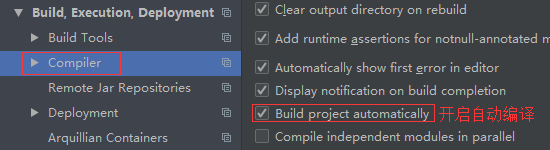
Settings > Appearance & Behavior



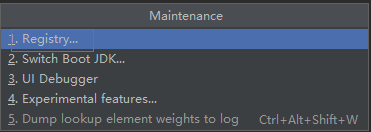
## 2.13.自动编译

开启自动编译之后，结合Ctrl+Shift+F9 会有热更新效果。

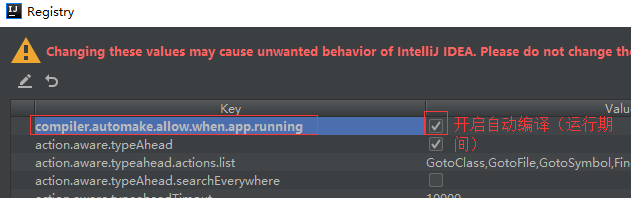
Settings > Build,Execution,Deployment



Ctrl + Shift + Alt + /  出现如图 点击Registry…

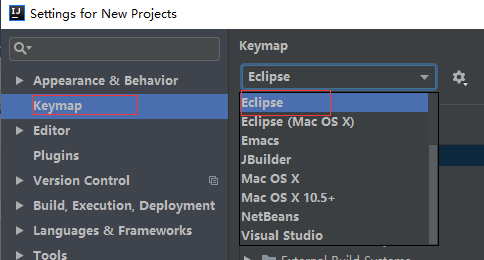


进入后，如图操作

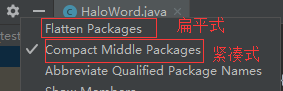


## 2.14. 将快捷键设置为跟Eclipse一样

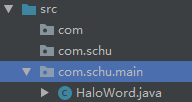
Settings > keymap 选择Eclipse



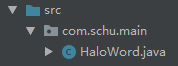
## 2.15.设置项目目录的展开结构



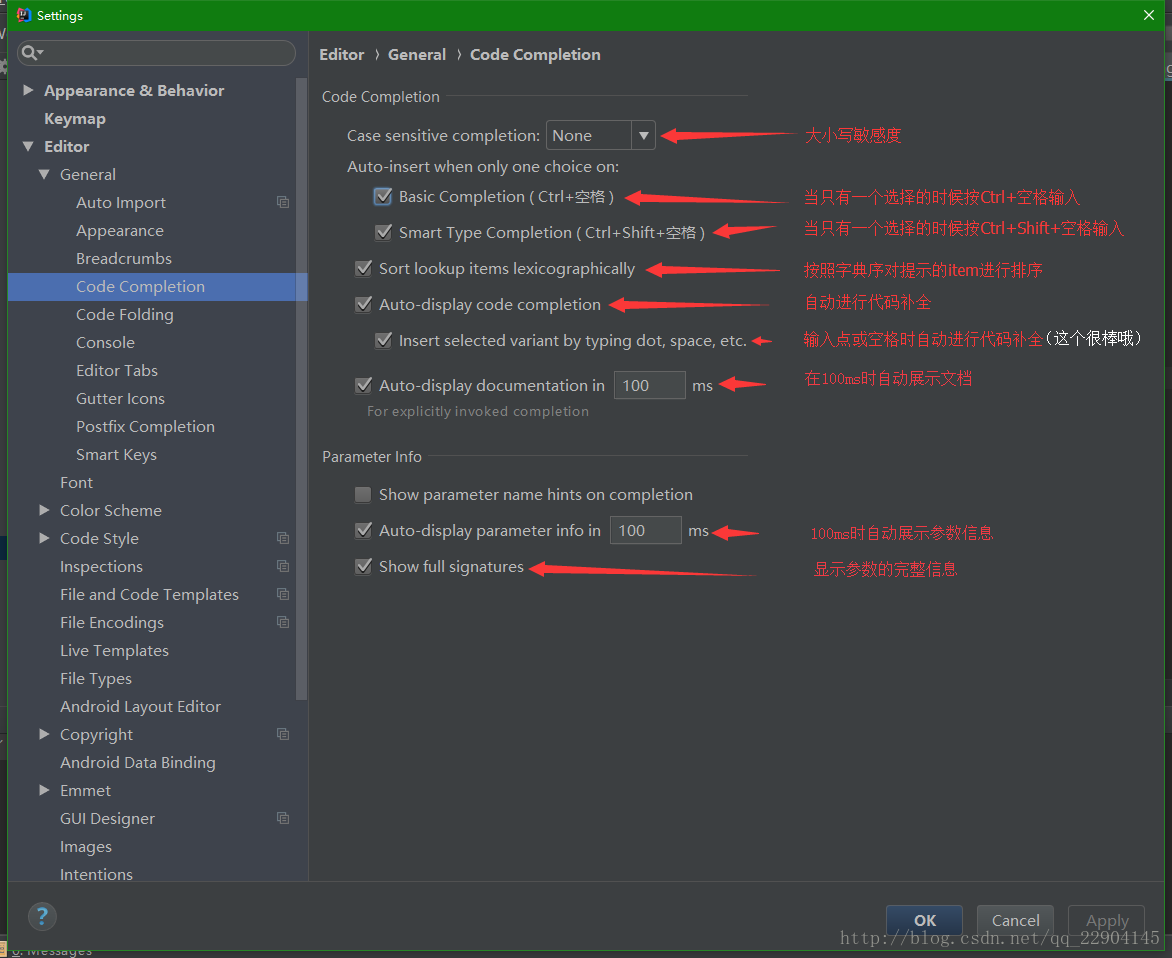
扁平式



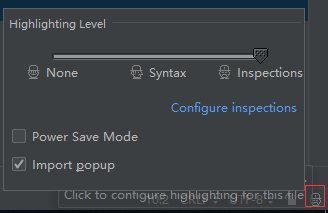
紧凑式



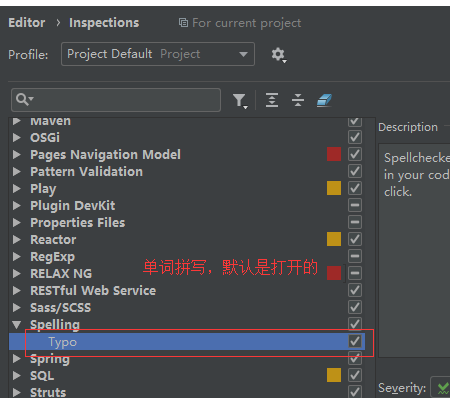
## 2.16.自动提示



## 2.17. 代码检测警告提示等级设置

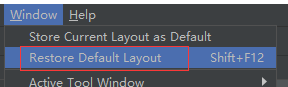


## 2.18.单词拼写设置

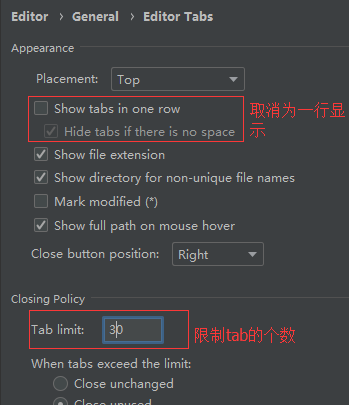


## 2.19.窗口复位

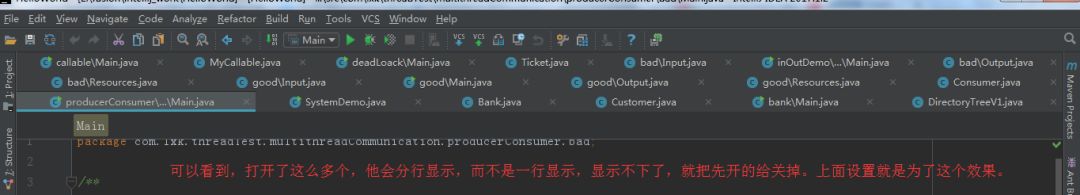
这个就是当你把窗口忽然间搞得乱七八糟的时候，还可以挽回，就是直接restore一下，就好啦。



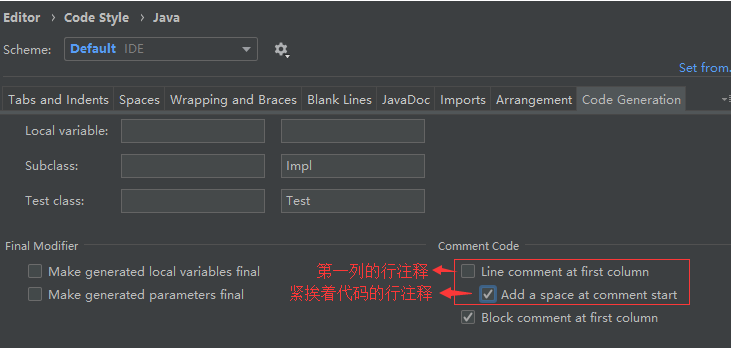
## 2.20. tab页面多行显示的设置



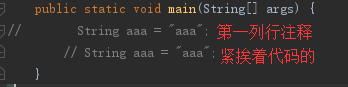
之后如图



## 2.21. 设置Java代码的注释风格

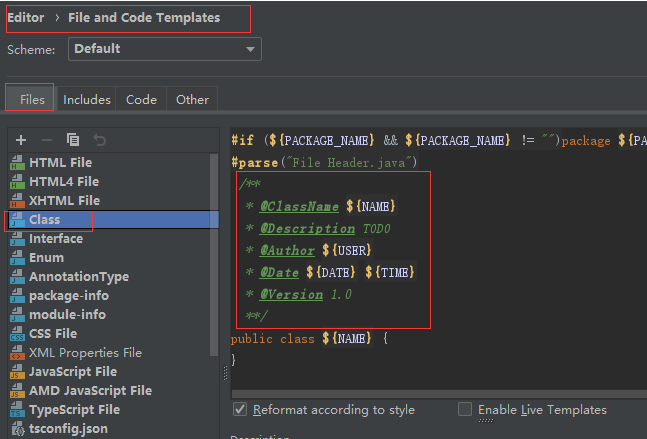


设置效果



## 2.22.设置类、方法的注释模板（待完成）

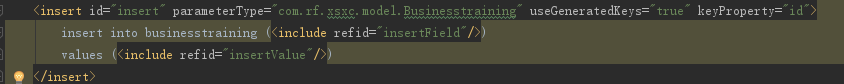
类的注释模板设置



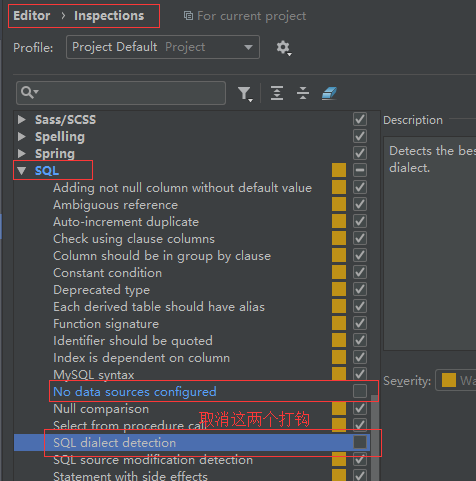
方法的注释模板设置

## 2.23. 去掉关于SQL的警告

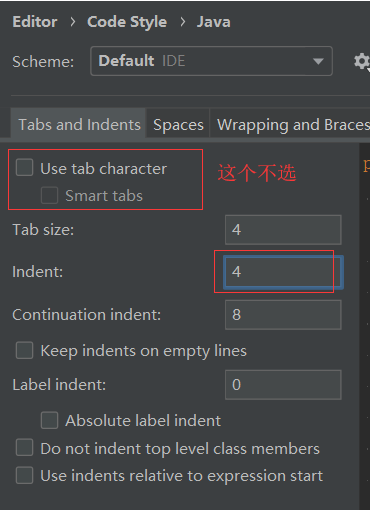
如图:出现sql的黄色警告



看着特别别扭，如何取消呢？

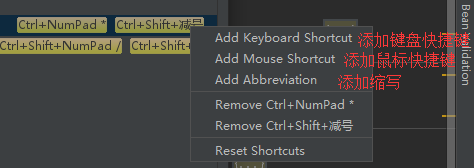


## 2.24.tab键为4个空格



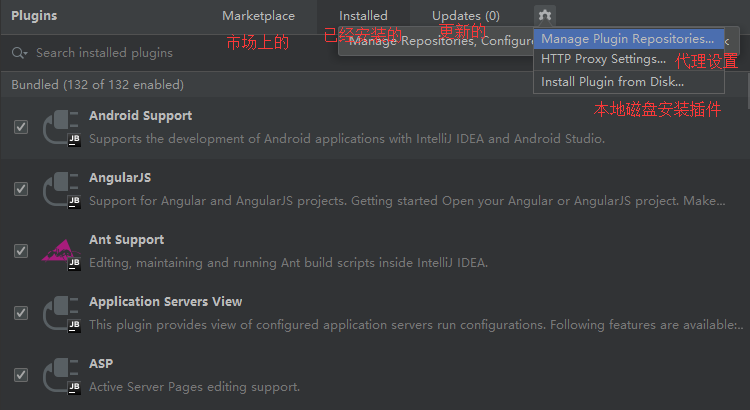
## 2.25.快捷键修改

找个某个功能的快捷键，右击 进行修改



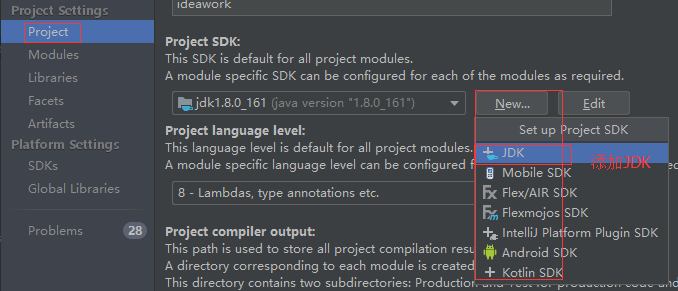
# 3、安装插件，扩展功能

File > Settings… > Plugins



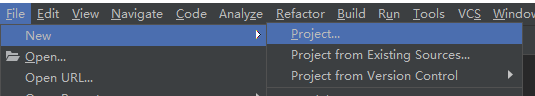
# 4、配置JDK

file -- Project Structure... – Project

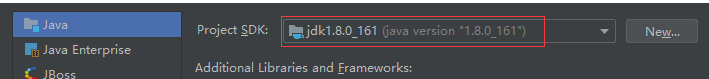


# 5、创建JavaSE项目

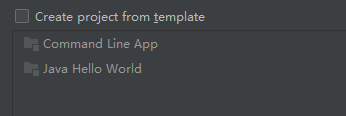
File > New > Project



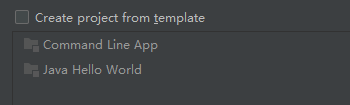
选择JDK



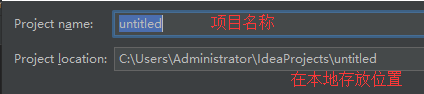
Next



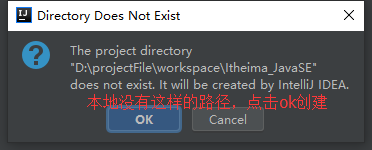
不用操作，直接next



填写项目名称及设置项目在本地存放位置



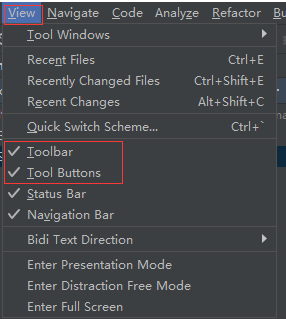
Finish

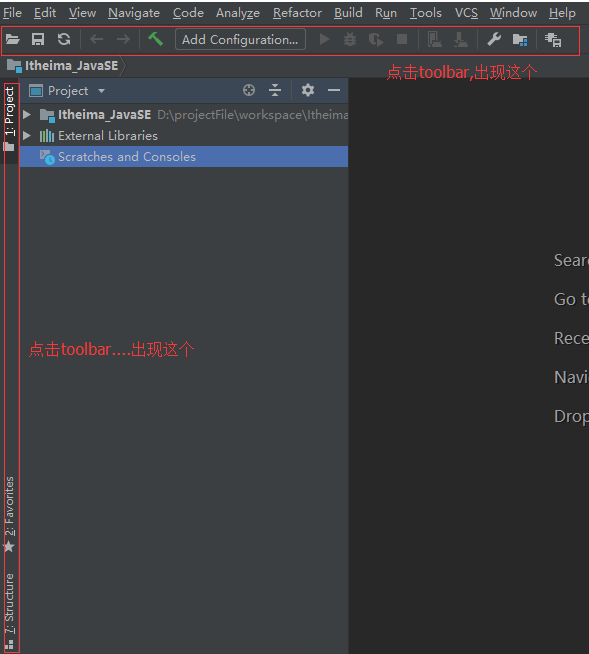


这样项目就创建好了

# 6、菜单栏显示

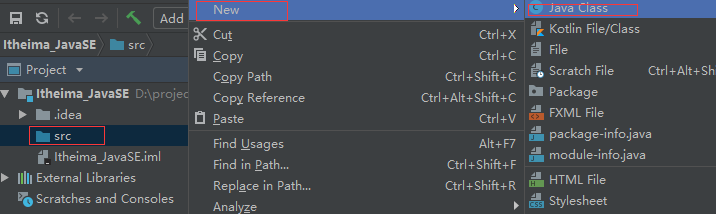
点击View 勾选Toolbar 与Tool Buttons 会出现上菜单栏与左侧菜单栏



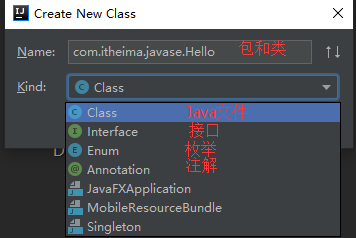


# 7、创建文件

Src > New > Java Class

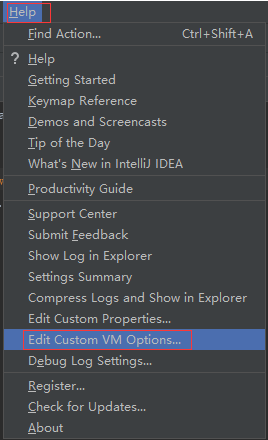


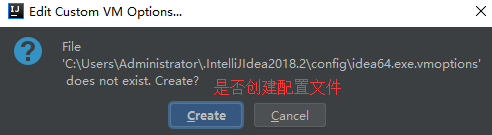
选择创建的文件类型



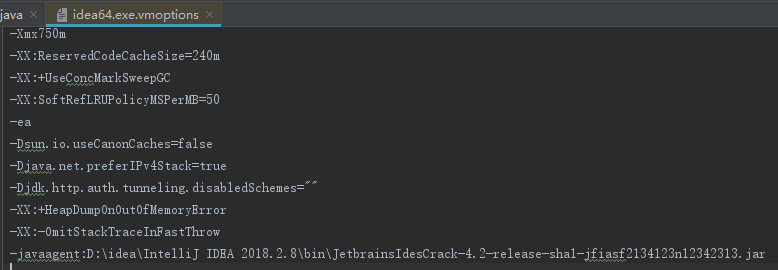
# 9、配置JVM参数

Help > Edit Custom VM Options…

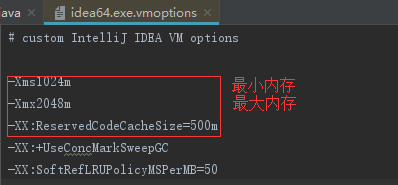




点击创建Create



修改内存配置



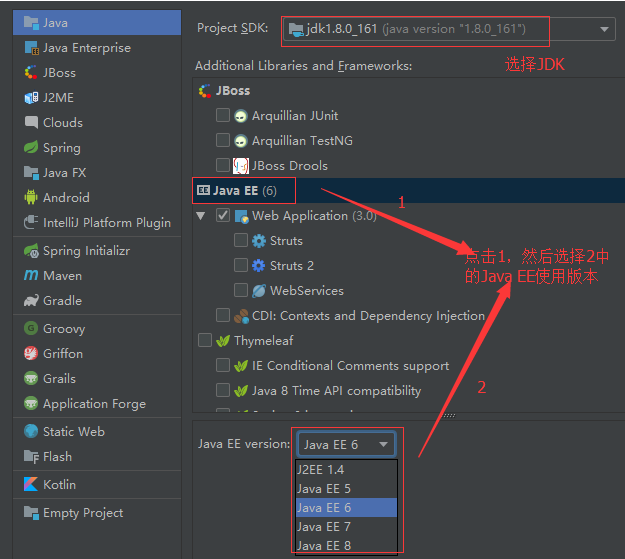
# 10、编辑页面跳转到Welcome 页面

File > Close Project



# 11、创建Javaweb项目

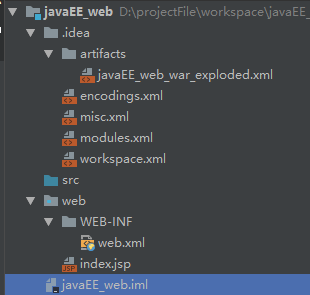
Create New Project



点击Next 填写项目名称及保存本地路径



Finish 创建完成

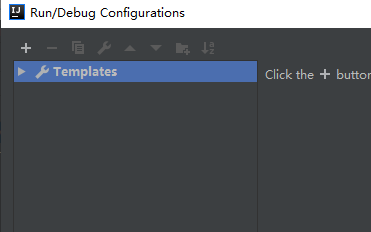


# 12、配置tomcat

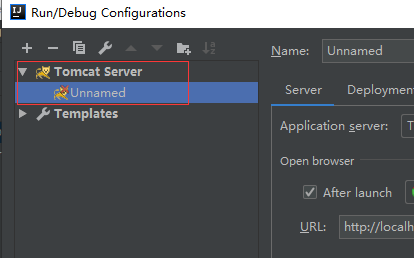
点击Add Configuration…



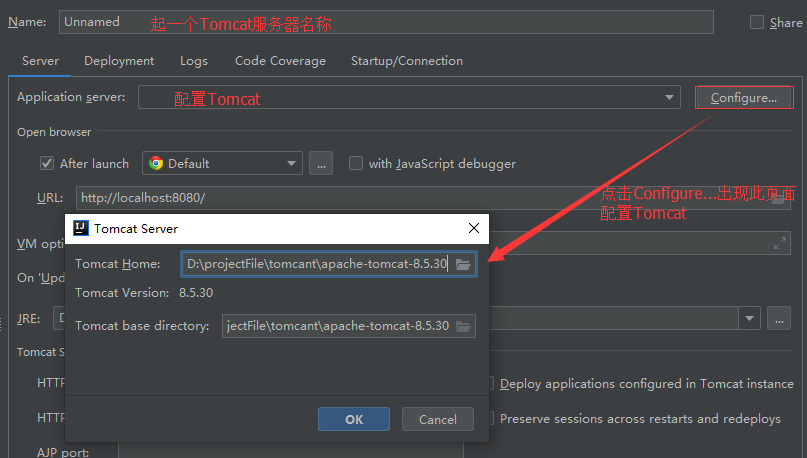
出现如下页面



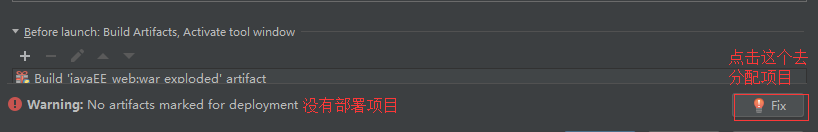
点击“+”号 找到Tomcat Server – local 会出现如图：



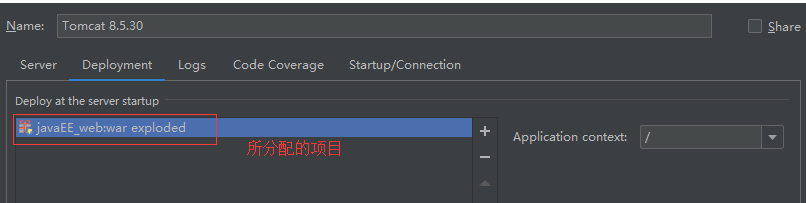
配置Tomcat



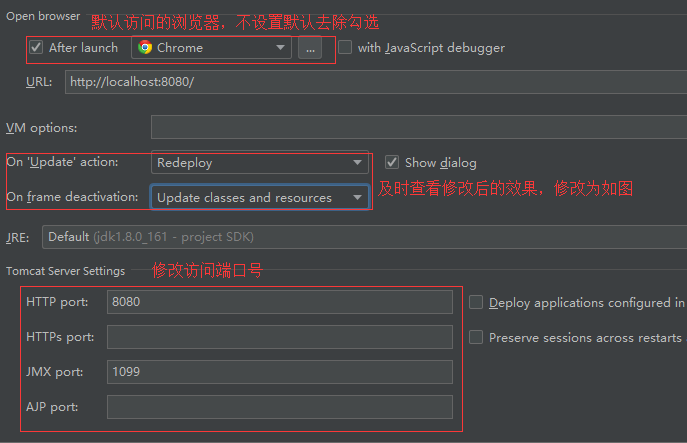
将tomcat分配给某个项目



出现如图：

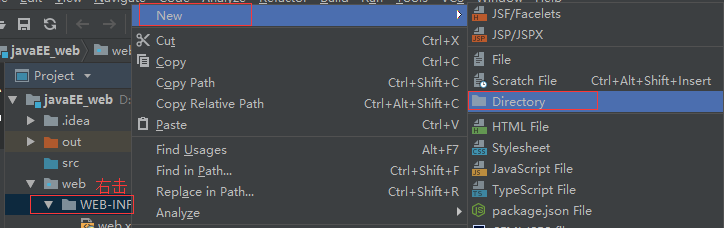


修改tomcat的其他设置

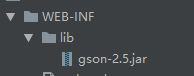


# 13、配置第三方依赖

在WEB-INF 下创建一个lib

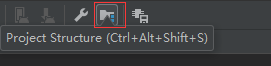


将jar包放进lib中

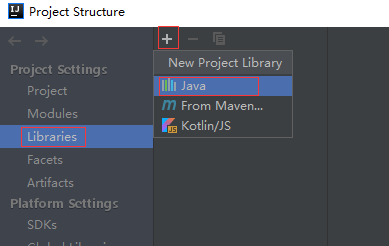


配置第三方依赖

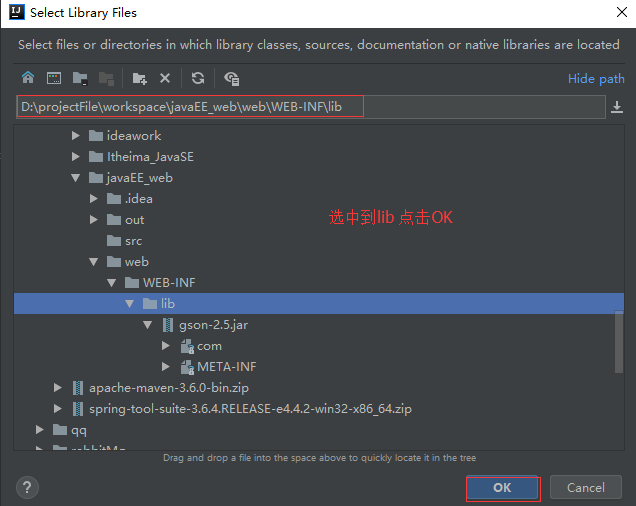
1. 点击 如图：



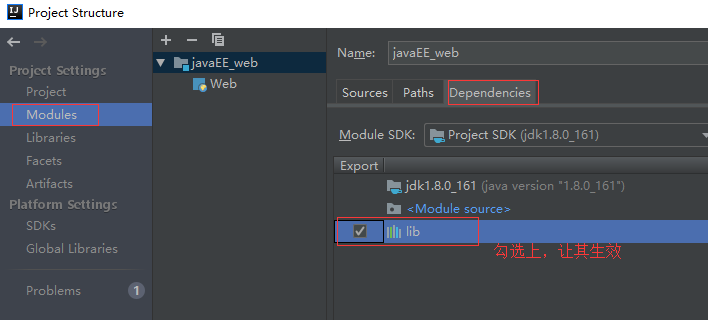
1. 找到Libraries 点击“+” 选择Java



出现如图：选中lib 点击OK



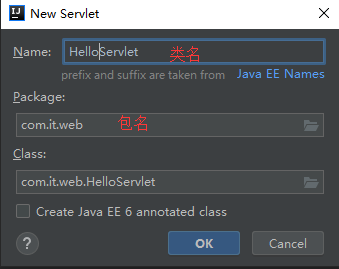
然后在选择Modules，使其lib生效



这样就配置完成了，可以在代码中调用试试

# 14、创建servlet

src右击new – servlet 输入类名、包名后点击OK



这时候发现类中有错，需要加载父类servlet



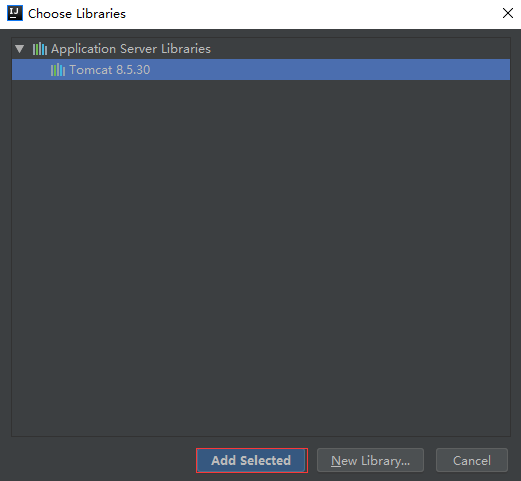
加载父类servlet配置 点击



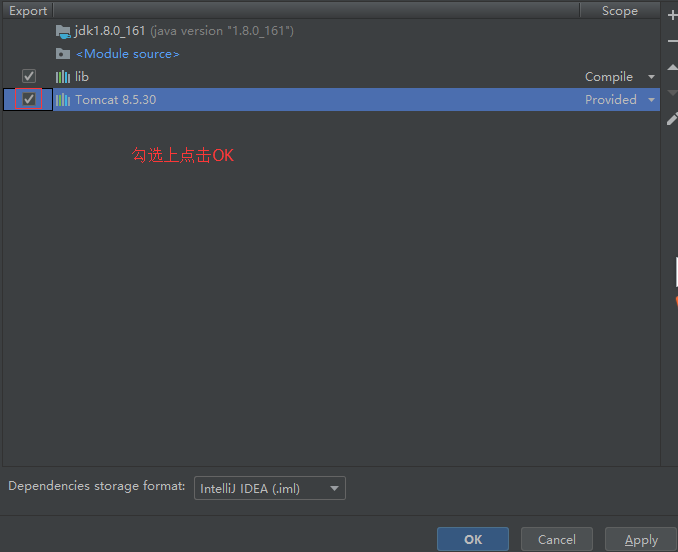
选择Modules 找到Dependencies 点击“+”选择Library



出现如图，选择tomcat 点击Add selected



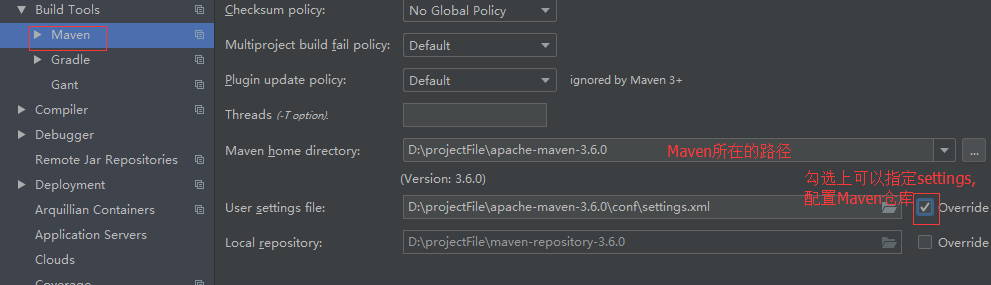
使其Tomcat servlet生效



这样就可以了

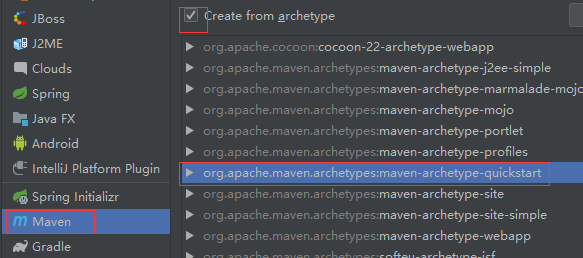
# 15、Maven的配置

Settings… -- build,Execution,Deployment – Build Tools



# 16、创建Maven工程

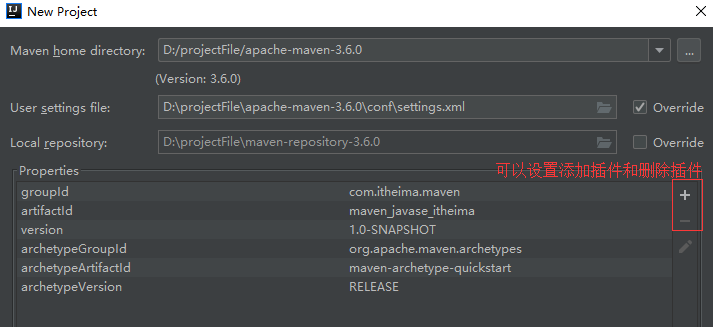
选中Maven，archetype（原型）打上勾，并选择原型



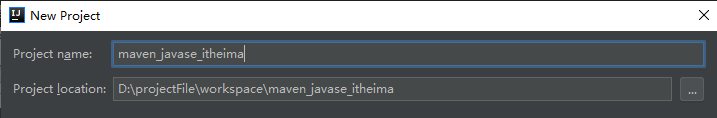
Next 创建GroupId 和 ArtifactId



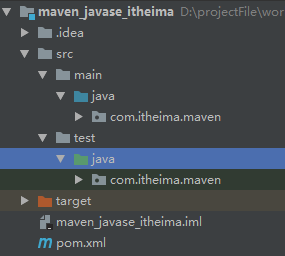
Next



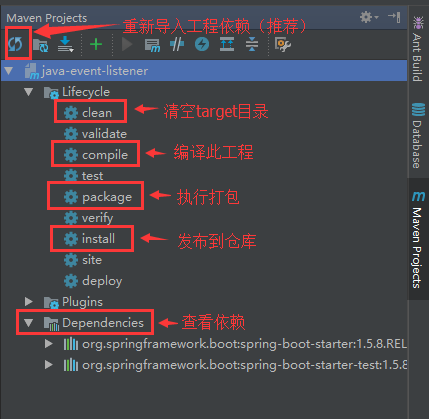
Next 创建工程的名称



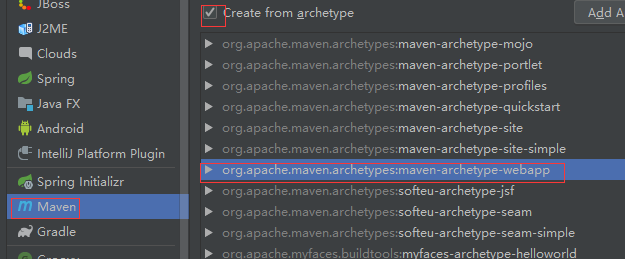
Finish 完成创建工程



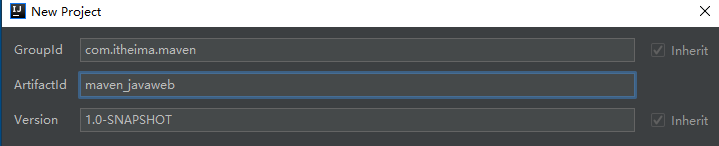
# 17、Maven的快捷菜单



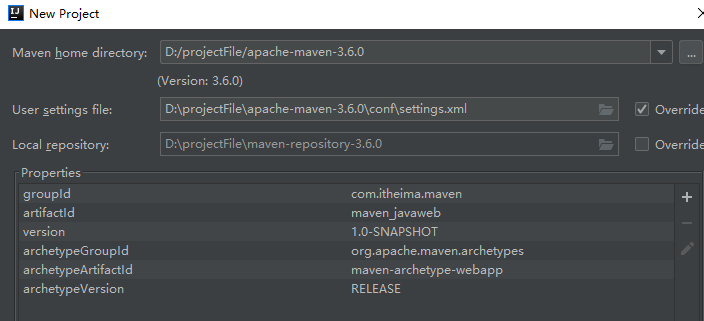
# 18、创建Maven的Javaweb工程



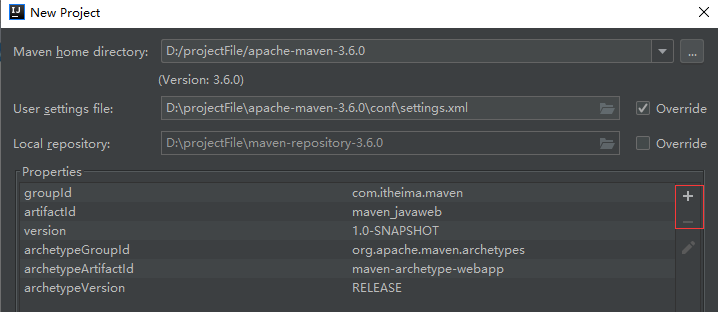
Next



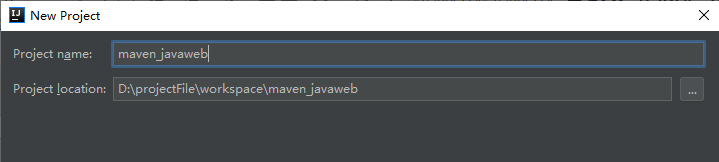
Next



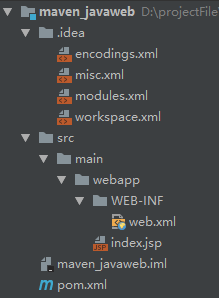
Next



Next



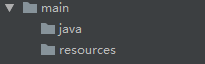
Finish 工程创建成功



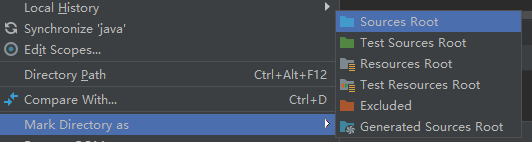
查看工程结构缺少Java、resources目录

创建Java目录，在main右击new – Directory

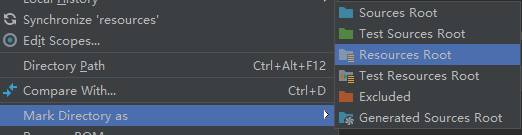
创建resources目录，在main右击new – Directory



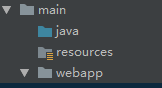
设置Java为根源目录，java右击 选择Mark Directory as 点击Sources Root



设置resources为资源根目录，java右击 选择Mark Directory as 点击Resources Root



样式成了这样

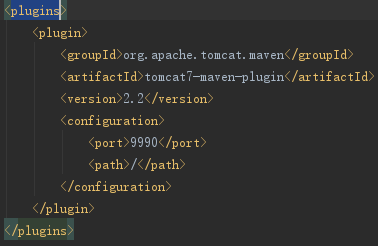


**为工程配置Tomcat**

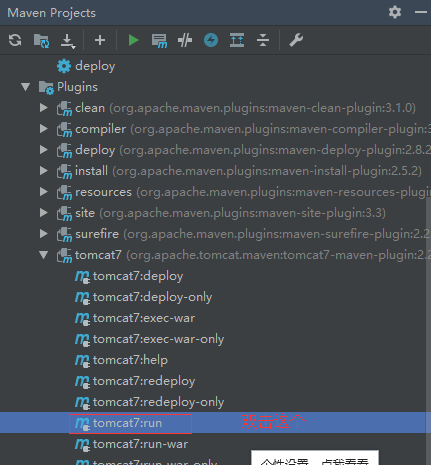
方法一：

方法二：配置Maven插件式Tomcat

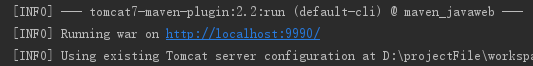
在pom.xml中



**使用插件式启动Tomcat，在Maven Projects中双击启动**

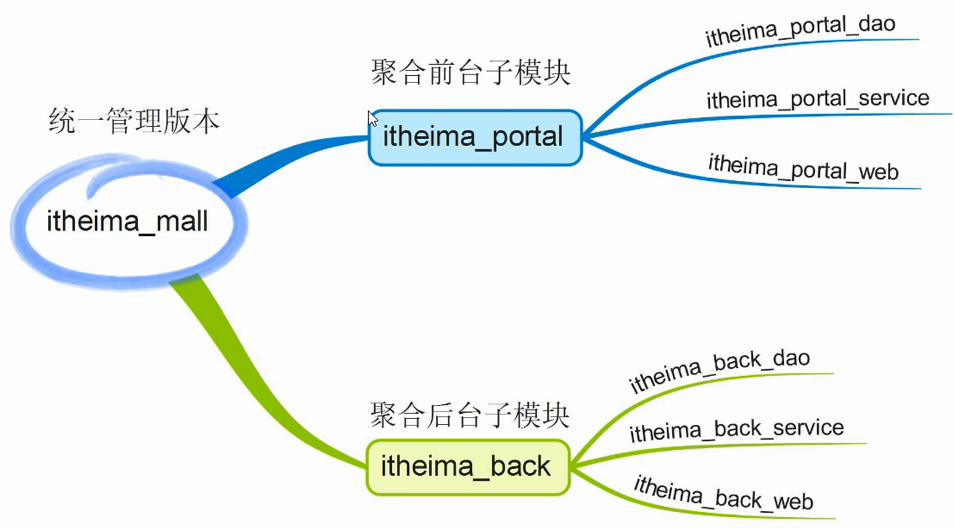


**访问路径为控制台中的路径**

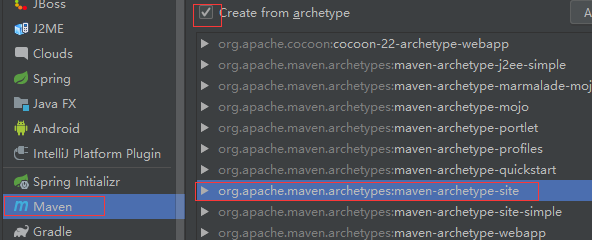


# 19、创建Maven聚合工程

聚合工程就如图这样式的一个工程



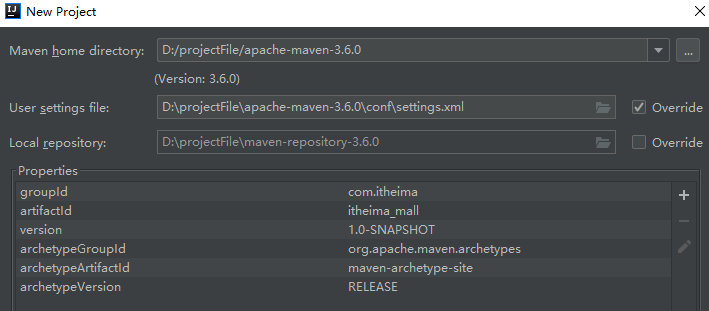
**1、首先创建itheima\_mall工程**



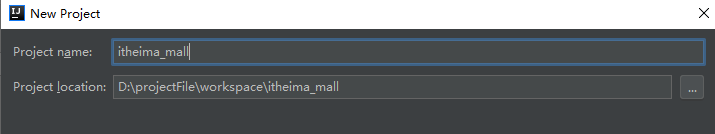
Next



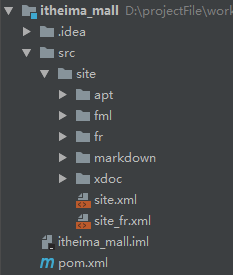
Next



Next

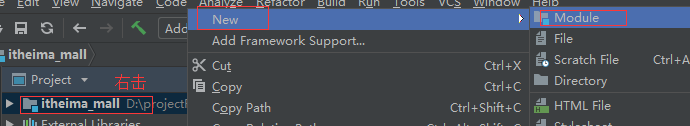


Finish

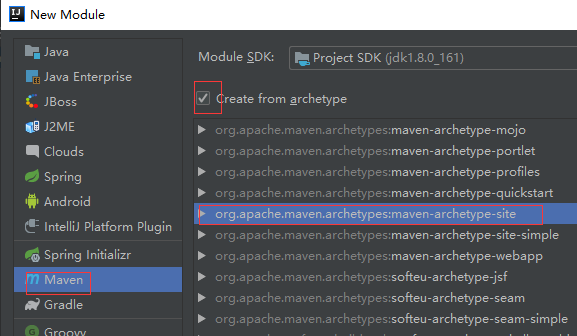


**2、在itheima\_mall工程的基础上创建itheima\_portal工程**

在itheima\_mall工程上右击 选择new -- Module



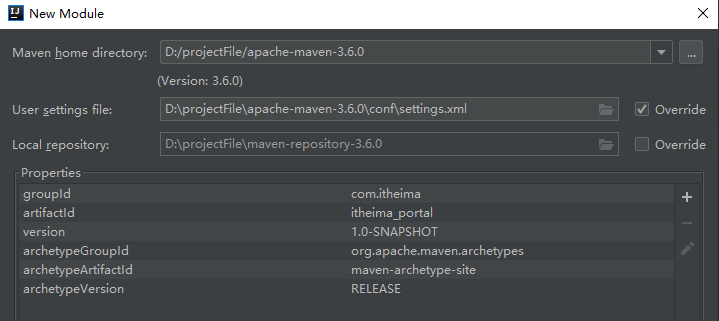
创建itheima\_portal工程



Next



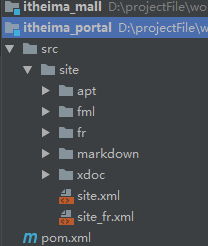
Next



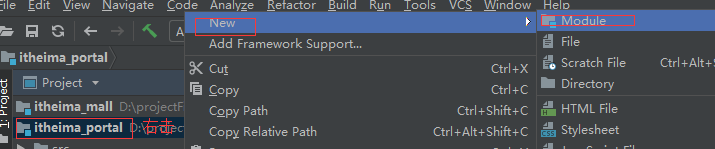
Next



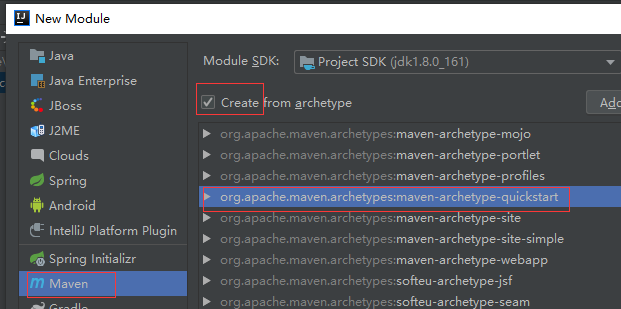
Finish



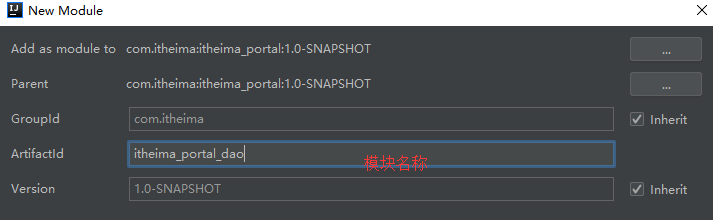
**3、在itheima\_portal工程里创建itheima\_portal\_dao子模块**



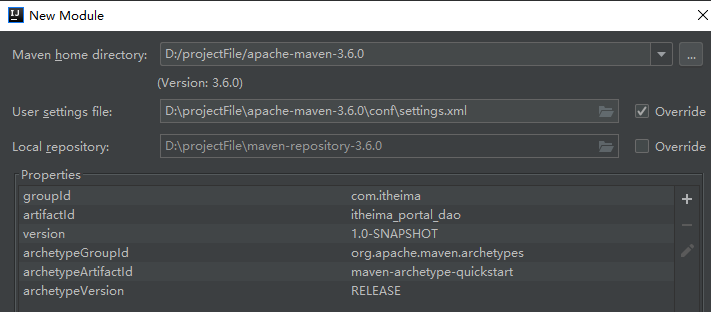
创建itheima\_portal\_dao子模块



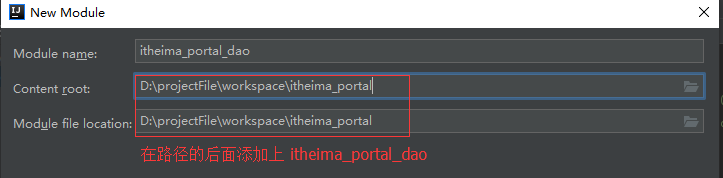
Next



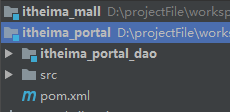
Next



Next



Finish



**4、在itheima\_portal工程里创建itheima\_portal\_service子模块**

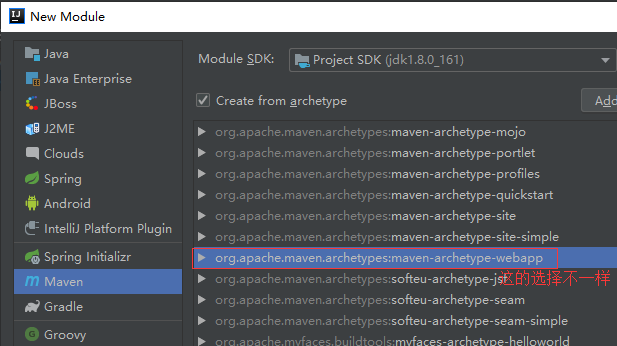
创建itheima\_portal\_service子模块与创建itheima\_portal\_dao步骤一样

不过创建完成后需要配置service依赖dao配置

在service的pom.xml中



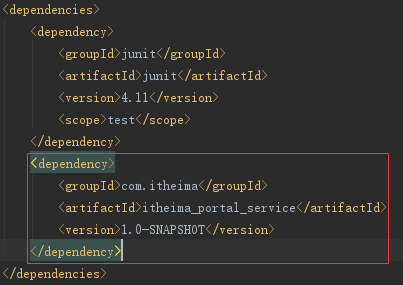
**5、在itheima\_portal工程里创建itheima\_portal\_web子模块**



除了上图的选择不一样，其他的与创建itheima\_portal\_dao步骤一样

不过创建完成后需要配置web依赖service配置

在web的pom.xml中



配置好后需要创建java与resource目录，参考“创建maven的javaweb工程”

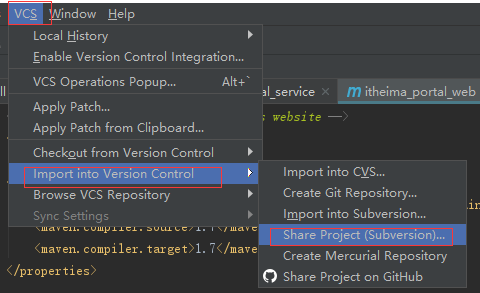
**6、在itheima\_mall工程的基础上创建itheima\_back工程**

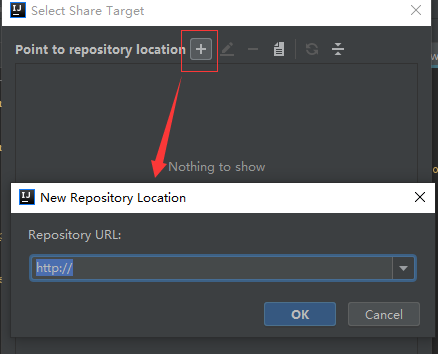
步骤参考2

**7、创建itheima\_back\_dao、itheima\_back\_service、itheima\_back\_web工程**

步骤参考3、4、5

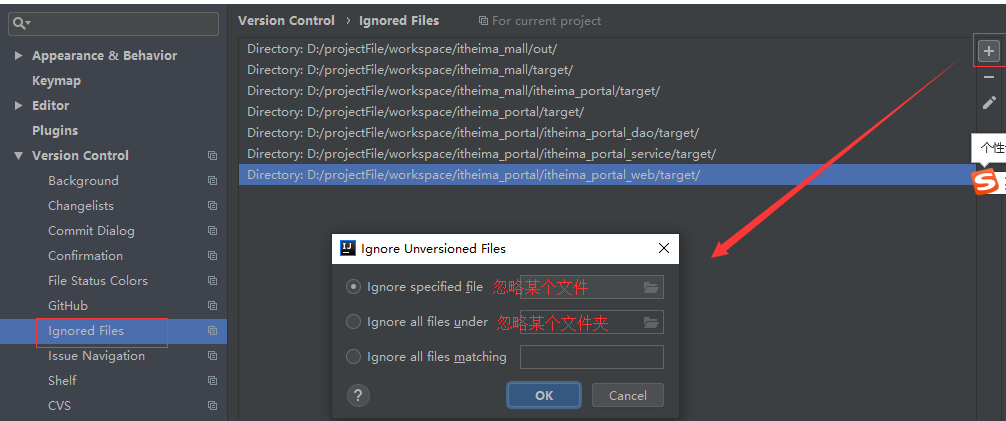
# 20、使用SVN管理代码





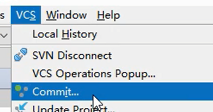
配置提交代码时的忽略

File – settings… -- version Control



提交代码

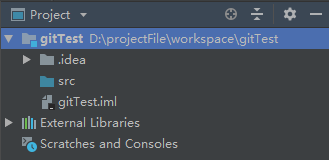
VCS – Commit…



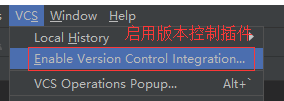
# 21、使用GIT管理代码

## 21.1.将代码交由git管理

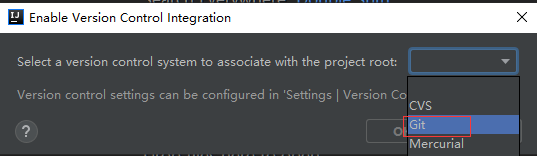
创建一个测试项目



VCS > Enable Version Control Integration...



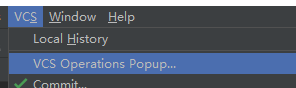
选择要使用的版本控制系统，选择Git 点击OK



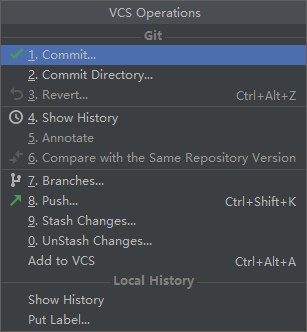
到此，已将本项目与Git进行关联，即已将本项目交由Git管理。

## 21.2.将代码提交到本地仓库（commit）

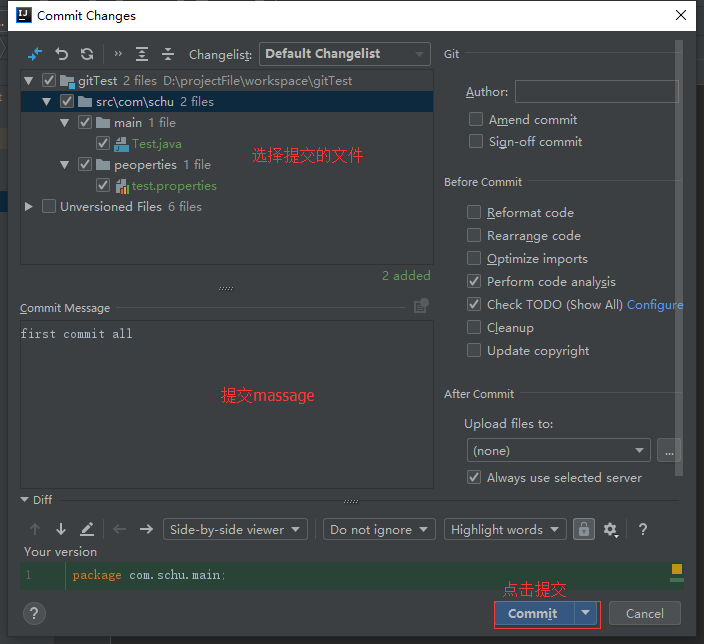
VCS  >  VCS Operations Popup...



出现Git所能进行的操作，点击commit



选择提交文件，并编写提交massage，最后点击commit提交



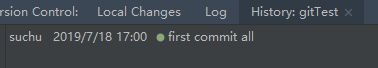
然后ide下方会出现一条绿色提示



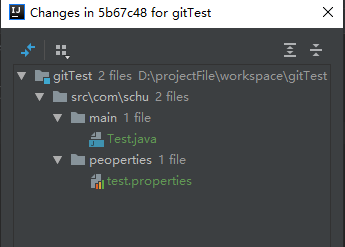
到此已将代码提交到本地仓库。

## 21.3.查看代码的提交历史

右击项目  ——>  Git  ——>  Show History

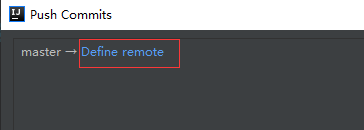


双击其中选项，会详细展示每一次的提交内容



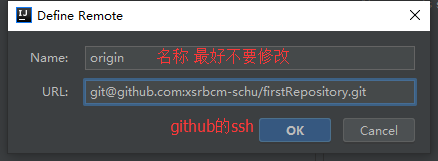
## **21.4.将代码提交到远程仓库（push）**

 VCS  ——>  VCS Operations Popup...  ——>  Push...

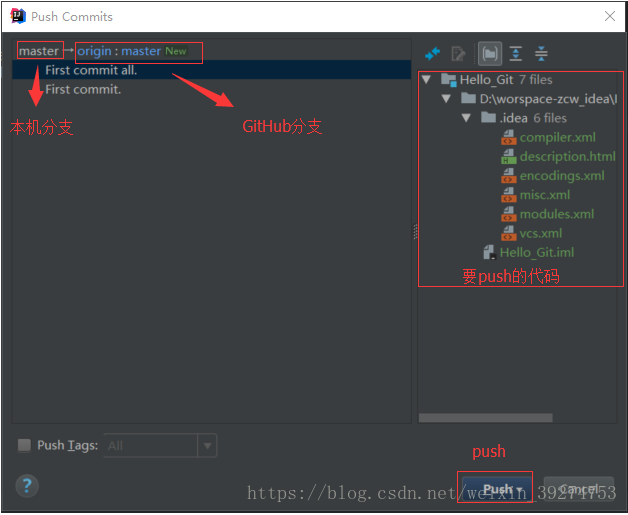


出现上述窗口，因为还没选择要连接的远程仓库，因此需要明确远程仓库

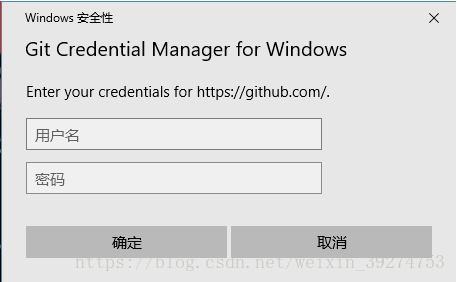
点击 Define remote 定义远程，再点击OK



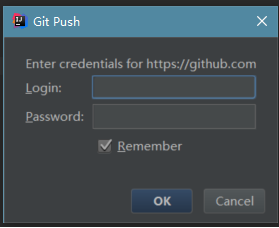
点击push， 将代码推送到GitHub



Git的凭证管理，输入GitHub的帐密



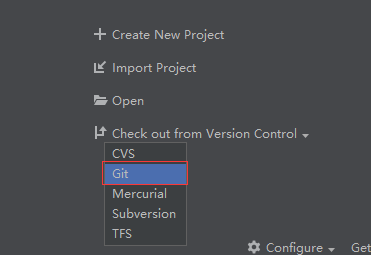
IDEA上也要输入一次，那就输入吧



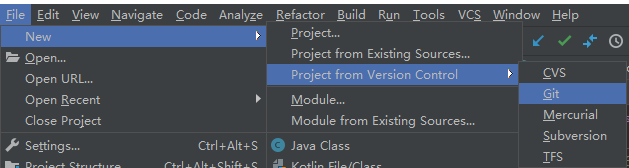
这里可以配置ssh免登陆，具体的见git文档

## **21.5.从远程仓库克隆项目到本地（Clone）**

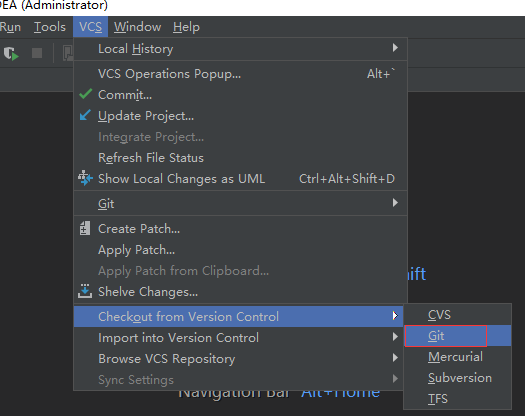
方式一：



方式二：



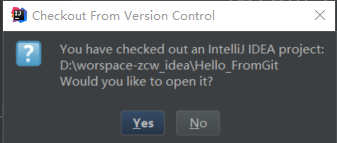
方法三：



连接远程仓库，点击clone，后面next

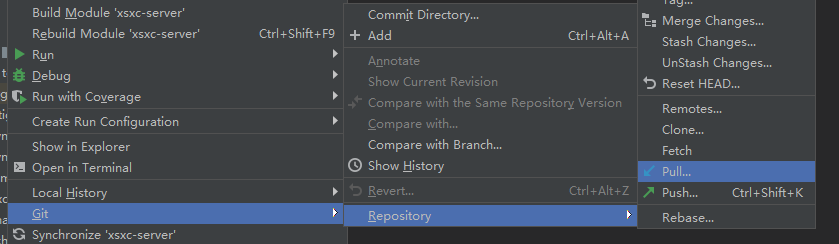


克隆完成后会询问你是否打开项目，点击yes

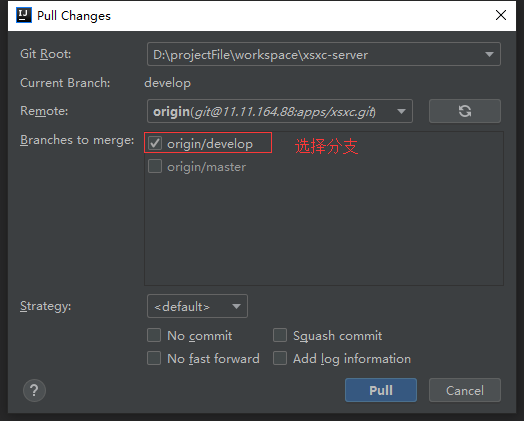


## 21.6.**从远程仓库中获取其他用户对项目的修改（pull）**

 右击项目  ——>  Git  ——>  Repository  ——>  Pull...



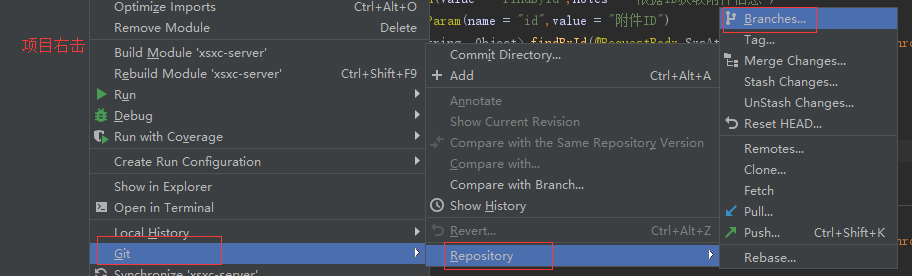
选择分支，点击pull，进行提取

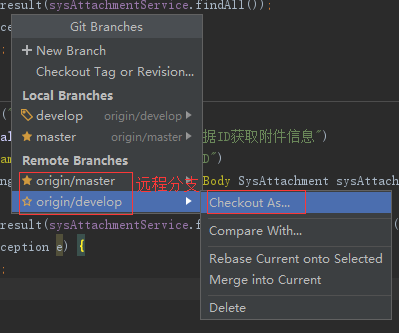


## 21.7.拉去远程分支

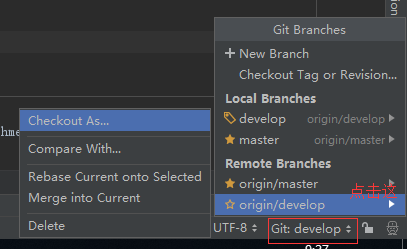
方法一：

项目右击，选择git > repository >Branches.. 弹出分支信息有本地的分支也有远程的分支，选择所有提取的远程 checkout As.. 创建分支。



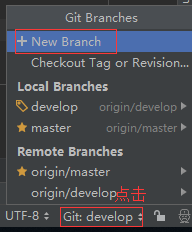


方法二：

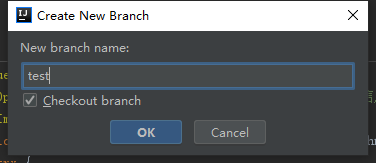


## 21.8.新建本地个人分支

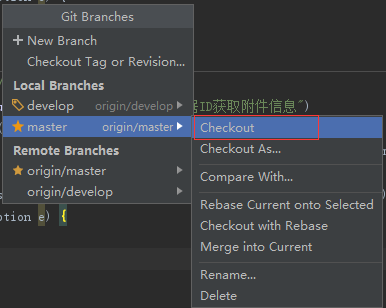
点击如图，选择New Branch



填写分支名称，点击OK



## 21.9.切换本地分支

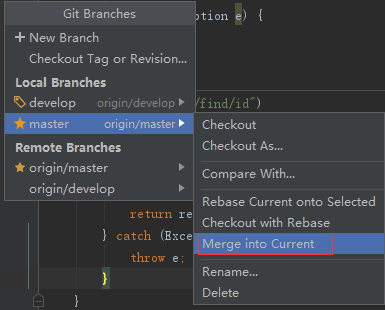


## 21.10.本地个人分支提交，以及push到远程

和提交代码，推送代码一样

## 21.11.本地合并分支

切换到主分支，选择次分支的merge，将次分支合并到主分支上

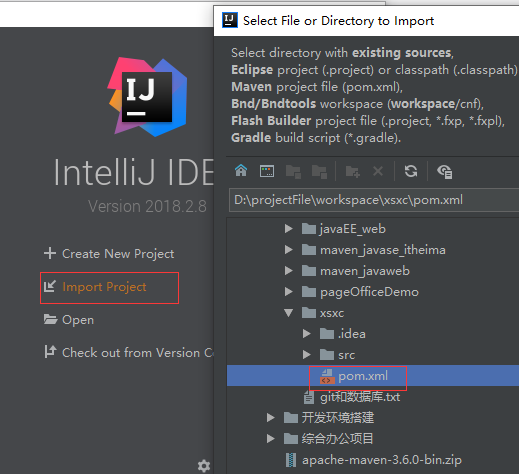


# 22、导入Maven项目

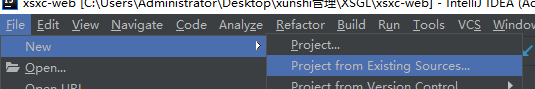
Maven项目第一次在本地打开时，最好使用导入，如果用open打开的话，好多jar包是不会加载的 比如jdk、等

导入Maven项目，包含：springboot

第一种：点击Import project



第二种：



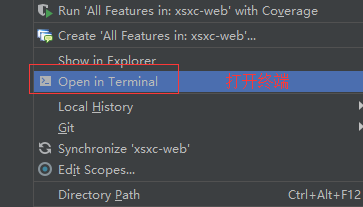
选择项目到pom.xml，点击OK，idea会自动识别为Maven文件

# 23、打开一些功能

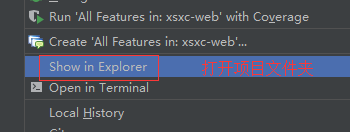
## 23.1.打开命令窗口

Alt+f12

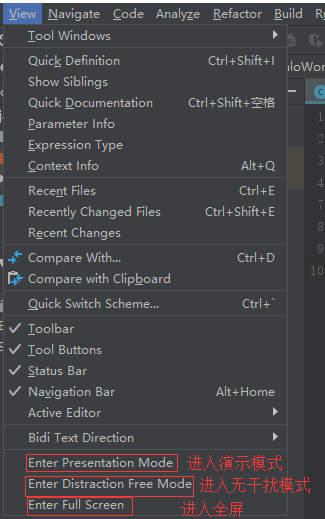
或者



## 23.2.打开项目文件夹



## 23.3.打开全屏模式

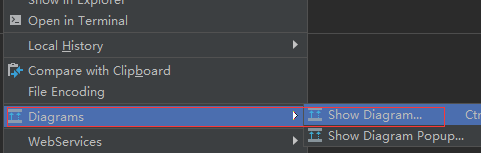


## 23.4.查看文件的修改记录

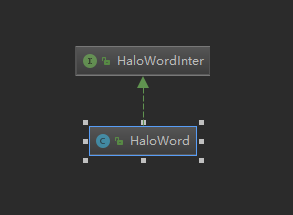


## 23.5. 查看类的结构图

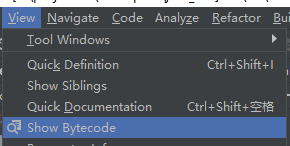
在类中任意一个地方右击



后出现



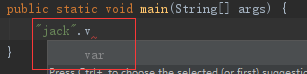
23.6.查看类的字节码



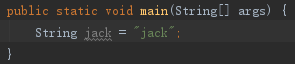
# 24、代码骚操作

## 24.1. var 声明

1、spring变量



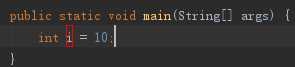
生成后



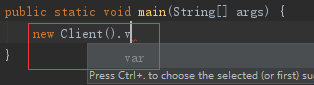
2、int变量



生成后



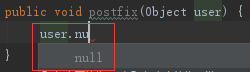
1. 对象变量



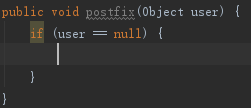
生成后



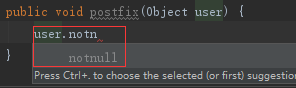
## 24.2. null 判空



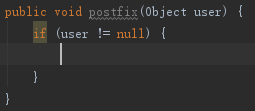
生成后



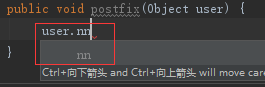
## 24.3. notnull 判非空



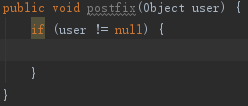
生成后



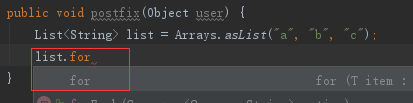
## 24.4. nn 判非空



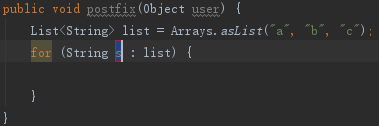
生成后



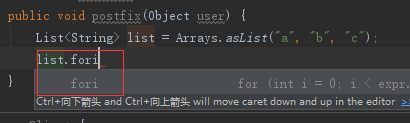
## 24.5. for 遍历



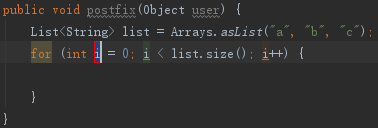
生成后



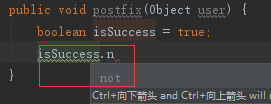
## 24.6. fori 带索引的遍历



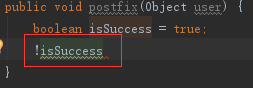
生成后



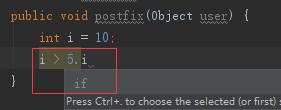
## 24.7. not 取反



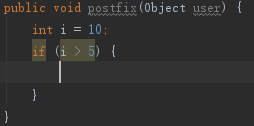
生成后



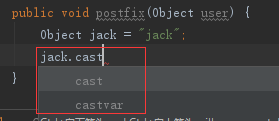
## 24.8. if 条件判断



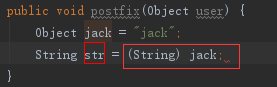
生成后



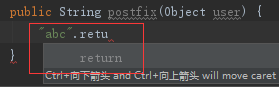
## 24.9. cast 强转



生成后



## 24.10. return 返回值



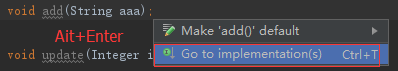
生成后



## 24.11. Alt + Enter操作

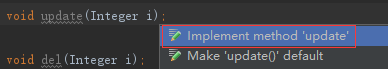
### ●接口类跳转实现类

在 **接口类** 中，如果光标当前所在的方法，已经在 **接口实现类** 中生成了，则此快捷键的效果是跳转。



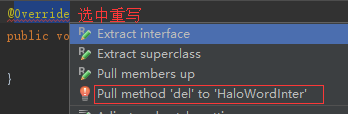
### ●接口类方法实现类生成

在 **接口类** 中添加一个方法后，让该 **接口实现类** 也跟着生成



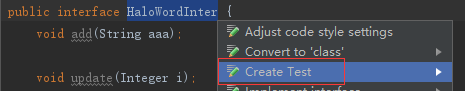
### ●实现类方法接口类生成

在 **接口实现类** 中添加一个方法后，让该 **接口类** 也跟着生成



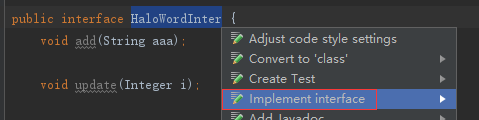
### ●对该类生成单元测试类

对当前光标所在类，生成单元测试类

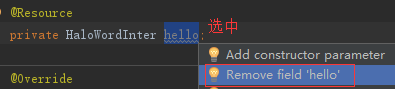


### ●类创建子类或接口实现类

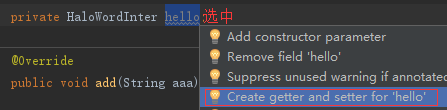
对当前光标所在类，创建子类，常用在对接口生成接口实现类



### ●移除未使用的变量、对象等元素

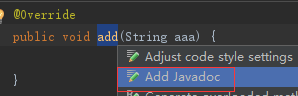


### ●对属性创建 set、get 方法



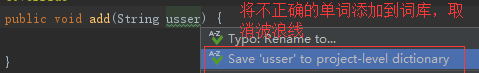
### ●对方法或变量添加doc

添加 doc，只能把光标放在方法名或是变量名等这类元素上才会有

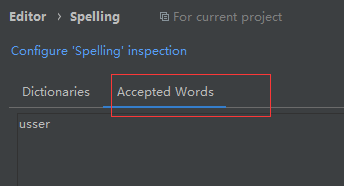


### ●消除单词检查错误的波浪线

把自己造的单词加入词库中，让拼写单词检查错误的波浪线效果消失。



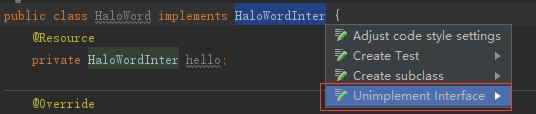
自己的词库在



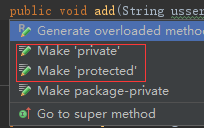
### ●快速移除实现类关系

快速移除当前类所继承的接口，并且同时清空已经写好的该接口所有的 Override 方法。

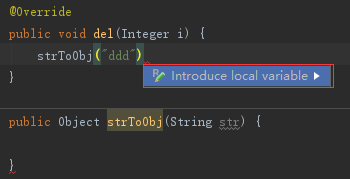
光标只能方式 **接口实现类** 上的 **接口对象单词** 上才可以实现。



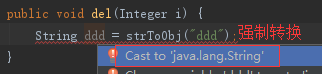
### ●修改光标当前元素的作用域



### ●给调用的方法生成返回值

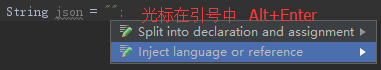


根据返回值自动强转

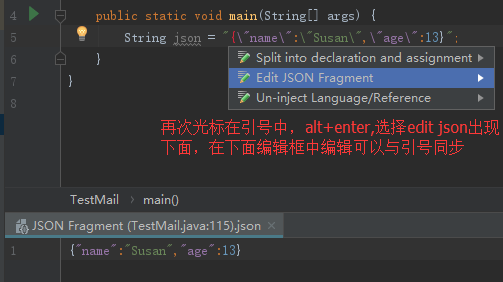


### ●语言注入

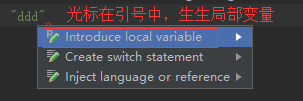
**这里注入json**



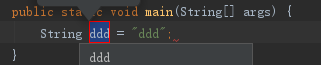




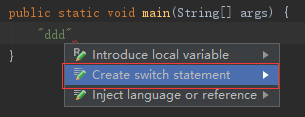
### ●引入局部变量



执行后



### ●创建switch语句



执行后



## 24.12. 快捷操作（eclips）

●CTRL + SHIFT + ENTER 在当前行任何地方可以快速在末尾生成分号；

●ALT+F7 查看此变量的引用

●CTRL + F 可以快速查找在文件中的内容并显示。

●按两个SHIFT，输入文件或文件夹名，可以快速跳转到具体的文件或文件夹中。

●Ctrl+Alt+L 格式化代码

●Shift+Alt+Z try-cache

●Alt+Insert(Fn+Alt+inset) 右击—Generate.. get/set生成

●Alt+Shift+R 文件重命名

## 24.13. main方法，控制台打印语句的创建

psvm

sout

## 24.14.dubug模式

F6 debug下一步

F7 debug进入此方法，当为变量时会在后面输出值

F8 跳到下一个debug

Ctrl+F8 在光标所在的行打上/取消断点



从左往右一一说明

1> Show Execution Point (Alt + F10)： 可跳转到当前代码执行的行。

2> Step Over (F6)：步过，一行一行地往下走，如果这一行上有方法不会进入方法。

3> Step Into (F5)：步入，如果当前行有方法，可以进入方法内部，一般用于进入自定义方法内，不会进入官方类库的方法

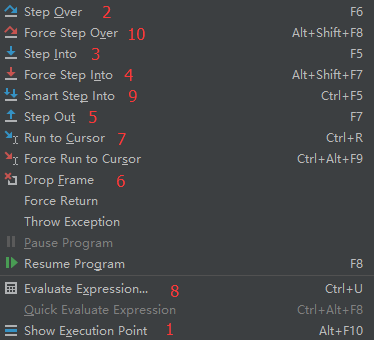
4> Force Step Into (Alt + Shift + F7)：强制步入，能进入任何方法，查看底层源码的时候可以用这个进入官方类库的方法。

5> Step Out (F7)：步出，从步入的方法内退出到方法调用处，此时方法已执行完毕，只是还没有完成赋值。

6> Drop Frame (默认无)：回退断点，后面章节详细说明。

7> Run to Cursor (Ctrl+R)：运行到光标处，而不需要打断点。

8> Evaluate Expression (Ctrl+U)：计算表达式，输入代码查询某部分的值。



在菜单栏Run中

9> Smart Step Into (Ctrl + F5) 智能步入，自动定位到当前断点行，并列出需要进入的方法



从上往下一一说明

1> Rerun 'xxxx'：重新运行程序，会关闭服务后重新启动程序。

2> Update 'tech' application (Ctrl + F5)：执行更新操作时所做的事情，一般选择'Update classes and resources'，即更新类和资源文件。配合热部署实现更改代码不重启

3> Resume Program (F8)：到下一个断点

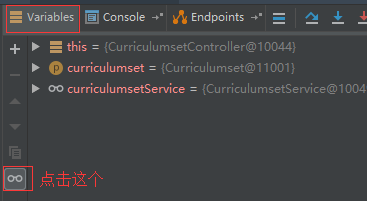
4> Pause Program：暂停程序，启用Debug。目前没发现具体用法。

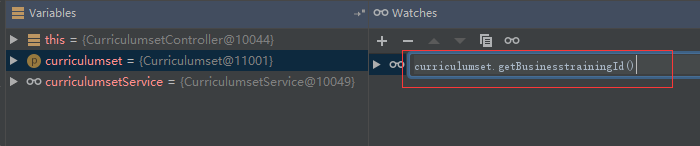
5> Stop 'xxx' (Ctrl + F2)：连续按两下，关闭程序

6> View Breakpoints (Ctrl + Shift + F8)：查看所有断点，后面章节会涉及到。

7> Mute Breakpoints：断点失效

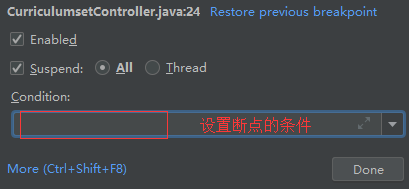
▲Watches的使用





▲条件断点设置

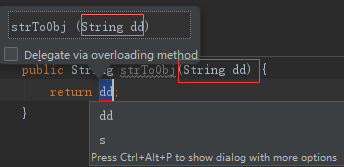
在断点上右键直接设置当前断点的条件



## 24.15. 提取写死的参数到方法参数

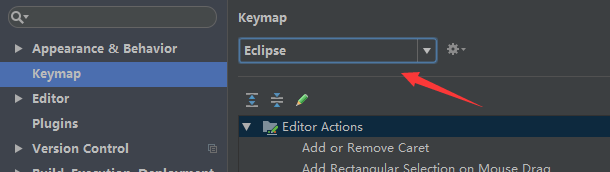


产生



## 24.16. Ctrl+Alt+↓ 在win10不好使

在Keymap里直接选择Eclipse可以直接使用Ctrl+Alt+向下箭头 复制当前行至下一行

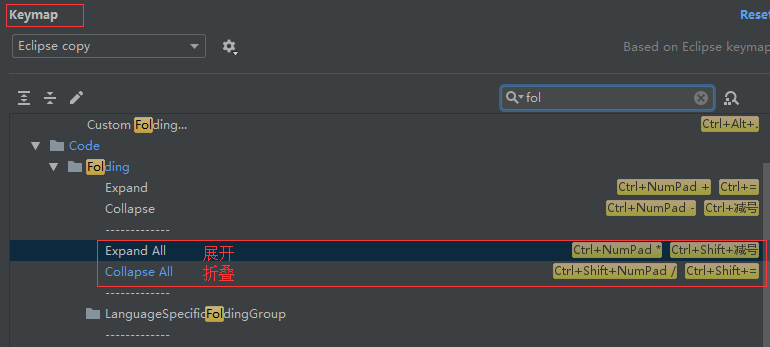


如果你的电脑使用Ctrl+Alt+向下箭头 的效果是翻转屏幕，那么在桌面右键将Inter显卡控制的快捷键禁用，如下图：



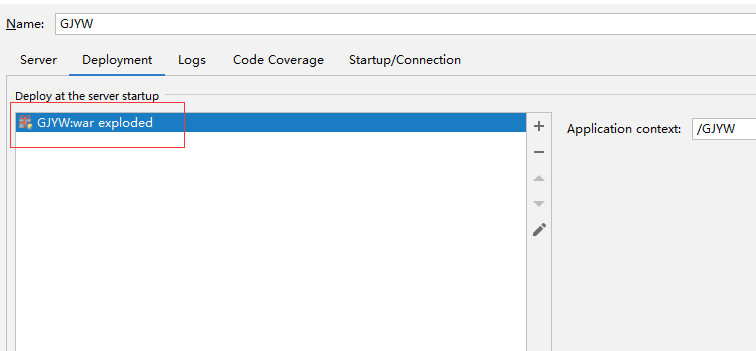
这样Ctrl+Alt+向下箭头 就起作用了

# 25.、设置折叠代码，展开代码快捷键



# 26、Tomcat加载web项目，在webapps下找不到问题

项目在tomucat部署成war exploded而不是war



Tips

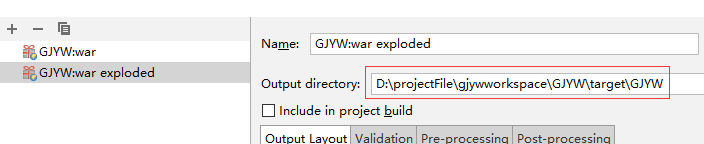
War模式：将web工程以包的形式上传到服务器

War exploded模式：将web工程以当前文件夹的位置关系上传到服务器

（1）war模式这种可以称之为是发布模式，看名字也知道，这是先打成war包，再发布。

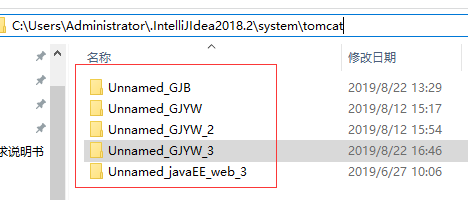
（2）war exploded模式是直接把文件夹、jsp页面 、classes等等移到Tomcat 部署文件夹里面，进行加载部署。因此这种方式支持热部署，一般在开发的时候也是用这种方式。

在idea中配置的tomcat，在运行时idea不会把项目放到自己的webapps路径下，而是在：artificts中war exploded设置的output 目录下



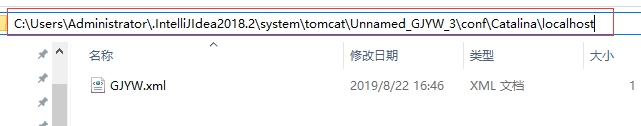
那么tomcat是如何加载到这个项目的，这里就要提到tomcat部署web应用的4中方法（见Tomcat.docx），这里和第三种方法自定义web部署文件有点相似。

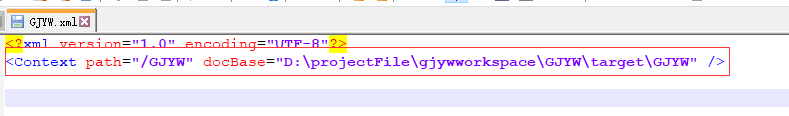
在idea中配置的tomcat，在运行时idea不会把项目放到该路径下，而是复制一份足够的配置文件，到 ${user.home}/.IntelliJIdea/system/tomcat 目录下，并会为每个web项目建立一个单独的文件夹，以“Unnamed\_项目名”命名（这个命名可在项目所在workspace/.idea/workspace.xml中修改）



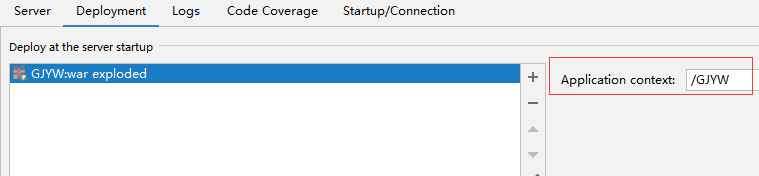
也就是说每个项目都有属于自己的一份tomcat配置，互不干扰。

每个项目的配置文件夹中有一个 /conf/Catalina/localhost/GJYW.xml 文件，内容如下：





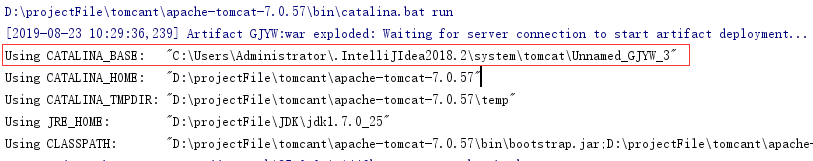
其中，path是指在访问此项目时，是否需要添加额外的路径，如果为空，则直接使用域名或者ip就可以访问到该项目：127.0.0.1。这个值在ieda中的Run/Debug Configurations中可以配置：



docBase是指要运行的项目的部署位置

我们来梳理一下逻辑

首先maven build项目，将构建结果写到项目的target目录下，然后idea复制一份tomcat的配置文件到${user.home}/.IntelliJIdea/system/tomcat 中，之后启动tomcat安装目录下的catalina.bat文件，tomcat读取配置文件，找到项目位置，然后就运行起来了。以下为Tomcat启动时打印的日志：



CATALINA\_BASE：指向${user.home}/.IntelliJIdea/system/tomcat下对应项目的文件夹，这里放着tomcat启动需要的而idea分配给每个启动项目特有的conf、work、log等文件夹

CATALINA\_HOME：指向使用的tomcat的文件夹，使用tomcat公用的bin，jar等文件夹

CATALINA\_TMPDIR：指向tomcat的文件夹下的temp文件夹，一般是使用CATALINA\_BASE下的temp文件夹，idea这里用的是tomcat自己的

JRE\_HOME：指向idea配置的jdk

CLASSPATH：tomcat文件夹下bin目录下的bootstrap.jar和tomcat-juli.jar