

IDEA

整理自互联网，仅供学习交流。



2018-10-30

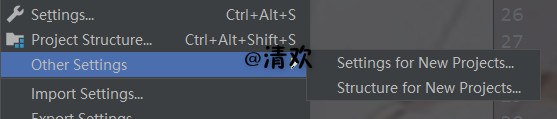
[公司名称]

[公司地址]

目录

# 基础设置

## 全局设置和临时设置



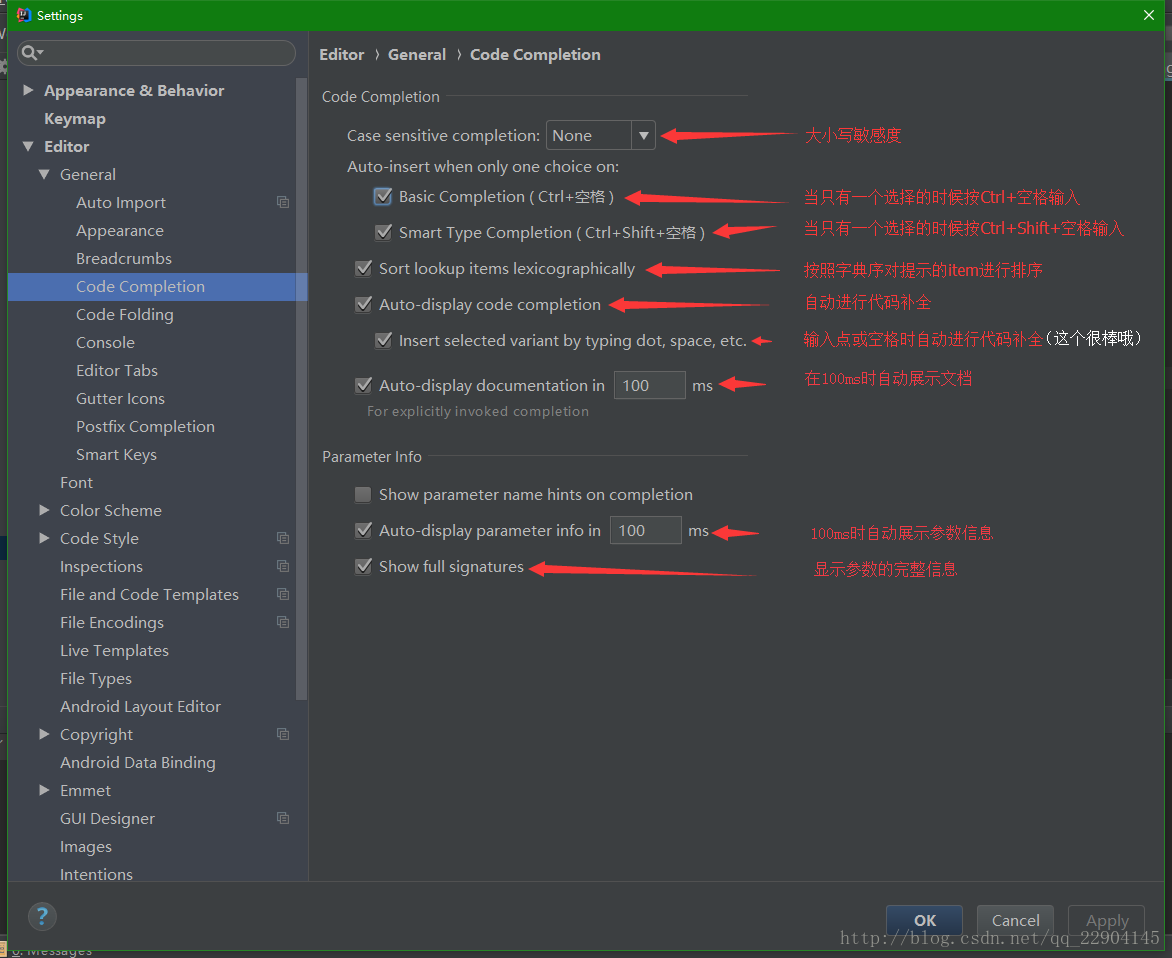
设置的时候根据需求设置全局和临时。临时设置只对当前项目有效。

## 自动提示

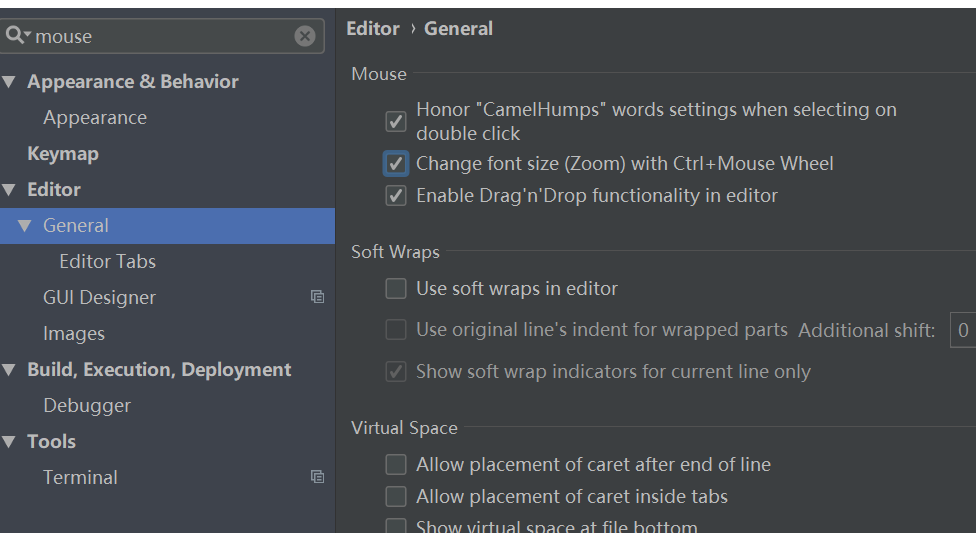
在IDEA中，默认的代码自动提示不够智能，现在配置成更加智能的方式。

File-Settings-Editor-General-Code Completion中

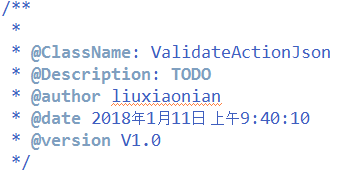
把最上面的大小写敏感度改成none，下面的钩全打。



## 滚动缩放代码字体大小



## 自动注释



一、首先我们来设置IDEA中类的模板：（IDEA中在创建类时会自动给添加注释）

1、File-->settings-->Editor-->File and Code Templates-->Files

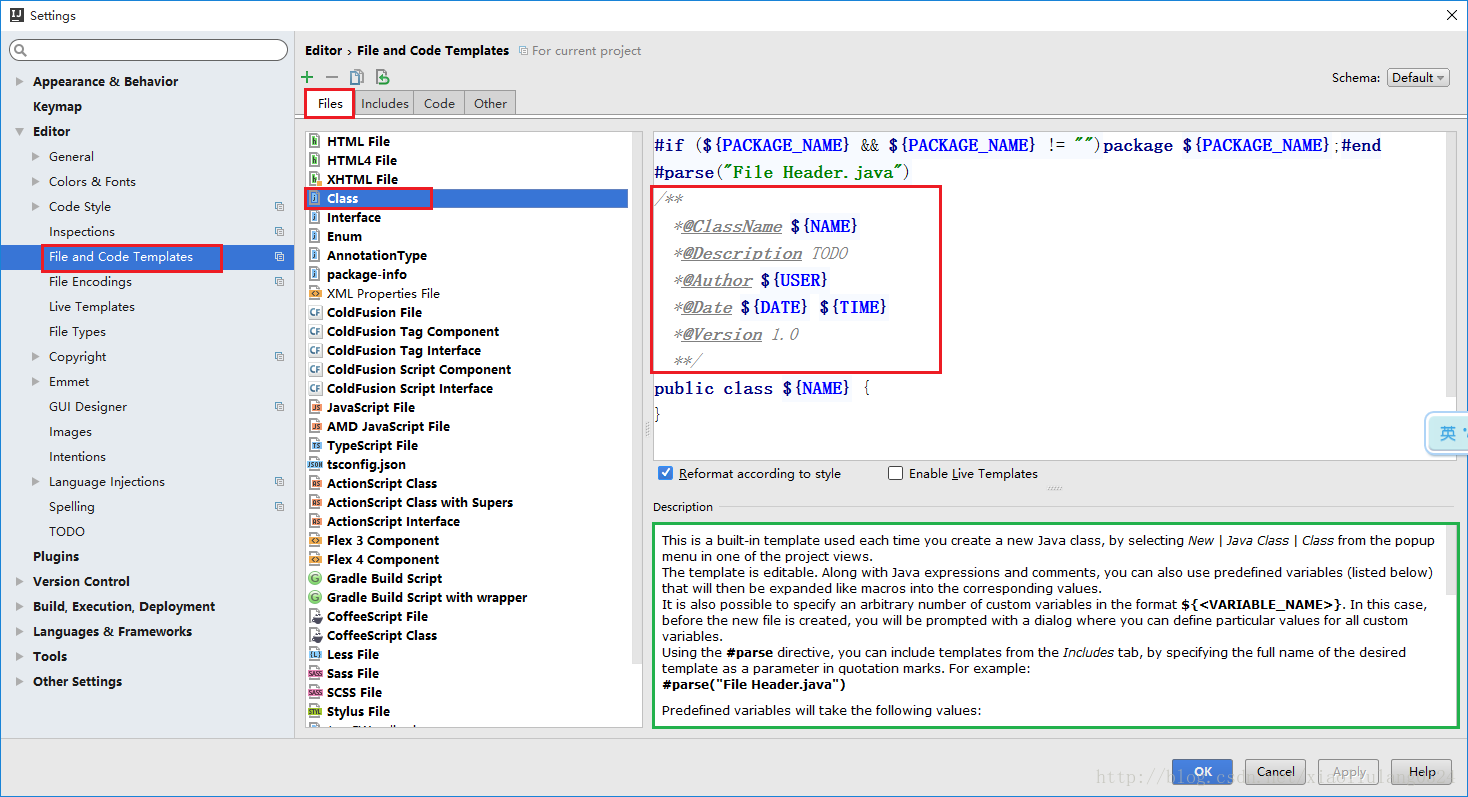
我们选择Class文件（当然你要设置接口的还也可以选择Interface文件）

（1）${NAME}：设置类名，与下面的${NAME}一样才能获取到创建的类名

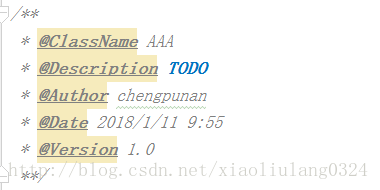
（2）TODO：代办事项的标记，一般生成类或方法都需要添加描述

（3）${USER}、${DATE}、${TIME}：设置创建类的用户、创建的日期和时间，这些事IDEA内置的方法，还有一些其他的方法在绿色框标注的位置，比如你想添加项目名则可以使用${PROJECT\_NAME}

（4）1.0：设置版本号，一般新创建的类都是1.0版本，这里写死就可以了



2、效果图展示

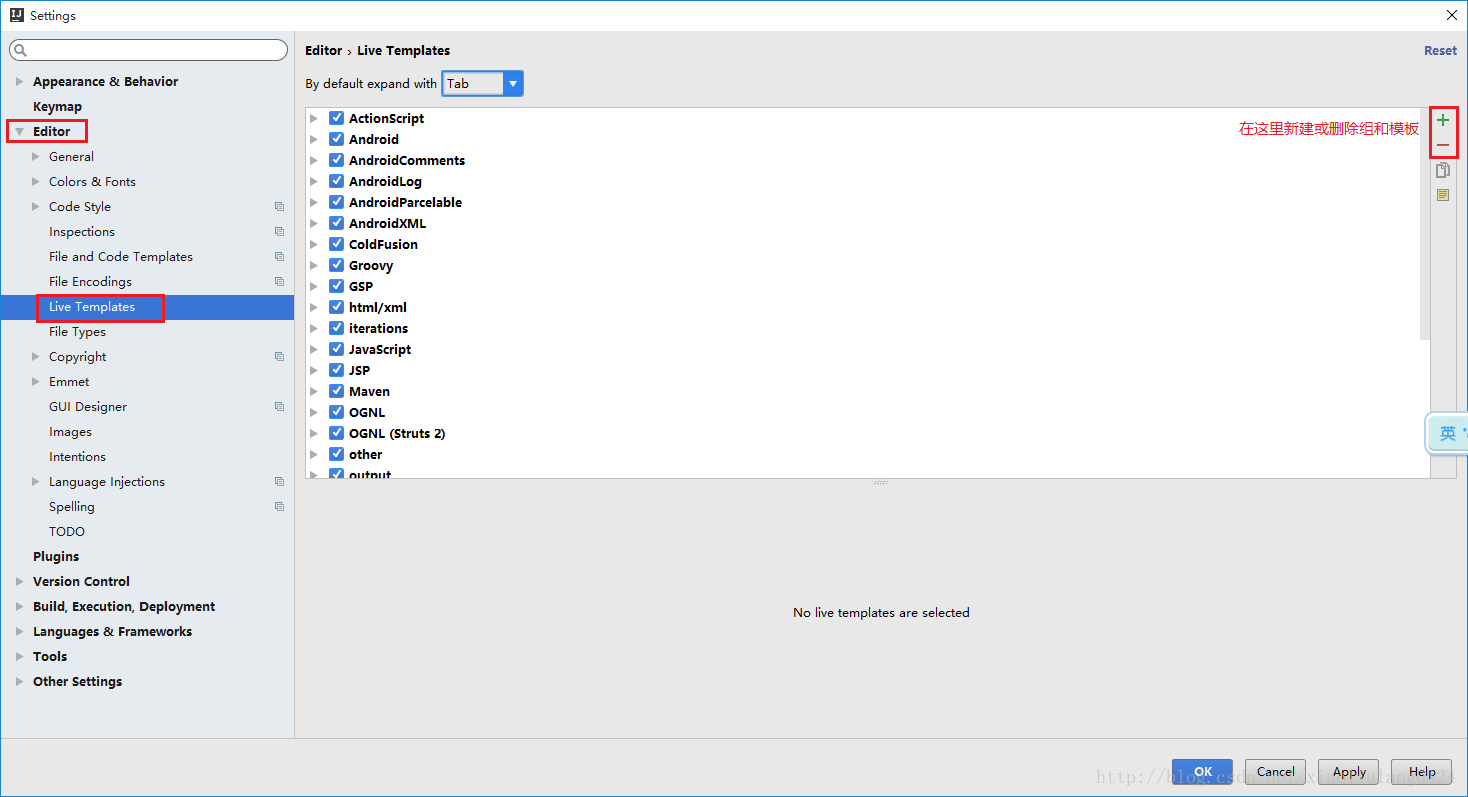


### 设置方法注释模板

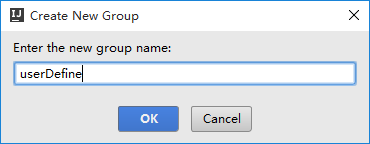
IDEA还没有智能到自动为我们创建方法注释，这就是要我们手动为方法添加注释，使用Eclipse时我们生成注释的习惯是

/\*\*+Enter，这里我们也按照这种习惯来设置IDEA的方法注释

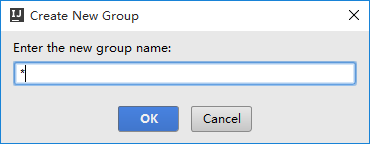
1、File-->Settings-->Editor-->Live Templates



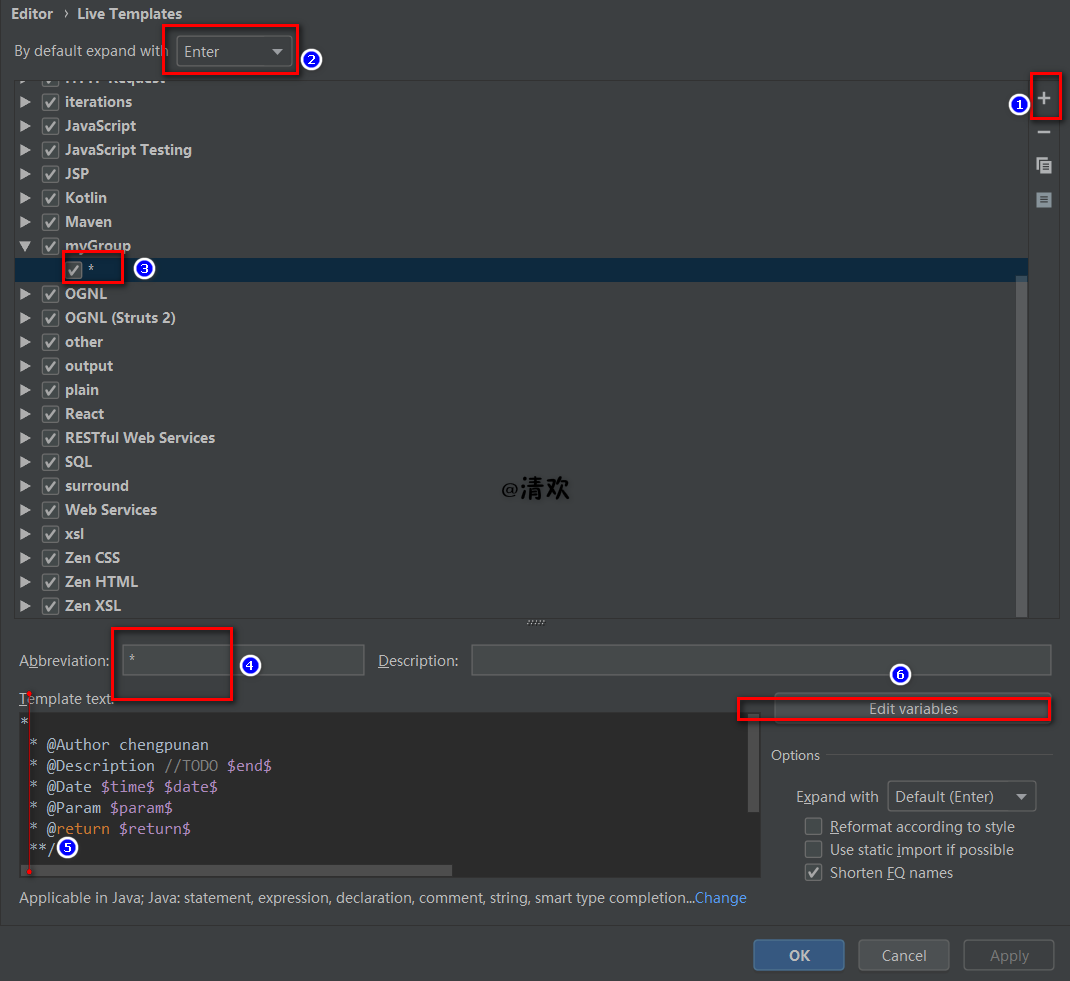
（1）新建组：命名为userDefine



（2）新建模板：命名为\*因为IDEA生成注释的默认方式是：/\*+模板名+快捷键（比如若设置模板名为add快捷键用Tab，则生成方式为/\*add+Tab），如果不采用这样的生成方式IDEA中没有内容的方法将不可用，例如获取方法参数的methodParameters(）、获取方法返回值的methodReturnType(）



（3）设置生成注释的快捷键



注意text中的内容第一行定格，第二行到最后一行开头空一个空格。如果复制乱格式，手打上去。保持对齐。

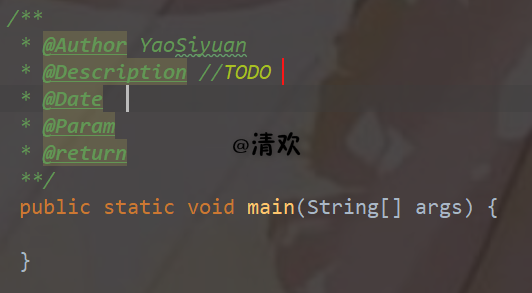
（4）设置模板：模板内容如下

注意第一行，只有一个\*而不是/\*

在设置参数名时必须用${参数名}$的方式，否则第五步中读取不到你设置的参数名

|  |
| --- |
| \*  \* @Author chengpunan  \* @Description //TODO $end$  \* @Date $time$ $date$  \* @Param $param$  \* @return $return$  \*\*/ |

如果使用/\*生成的模板注释将会是如下效果：所以我们要去掉最前面的/\*

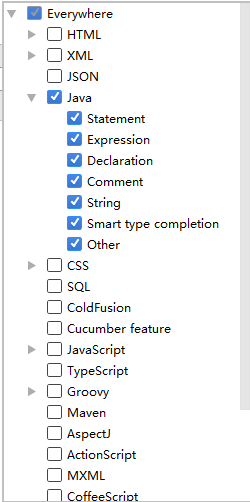


（5）设置模板的应用场景

点击模板页面最下方的警告，来设置将模板应用于那些场景，一般选择EveryWhere-->Java即可

（如果曾经修改过，则显示为change而不是define）

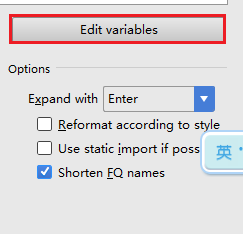
https://img-blog.csdn.net/20180111103636249?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQveGlhb2xpdWxhbmcwMzI0/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/SouthEast



（6）设置参数的获取方式

选择右侧的Edit variables按钮

PS:第五步和第六步顺序不可颠倒，否则第六步将获取不到方法



选择每个参数对应的获取方法（在下拉选择框中选择即可），网上有很多教程说获取param时使用脚本的方式，我试过使用脚本

的方式不仅麻烦而且只能在方法内部使用注释时才能获取到参数



（7）效果图

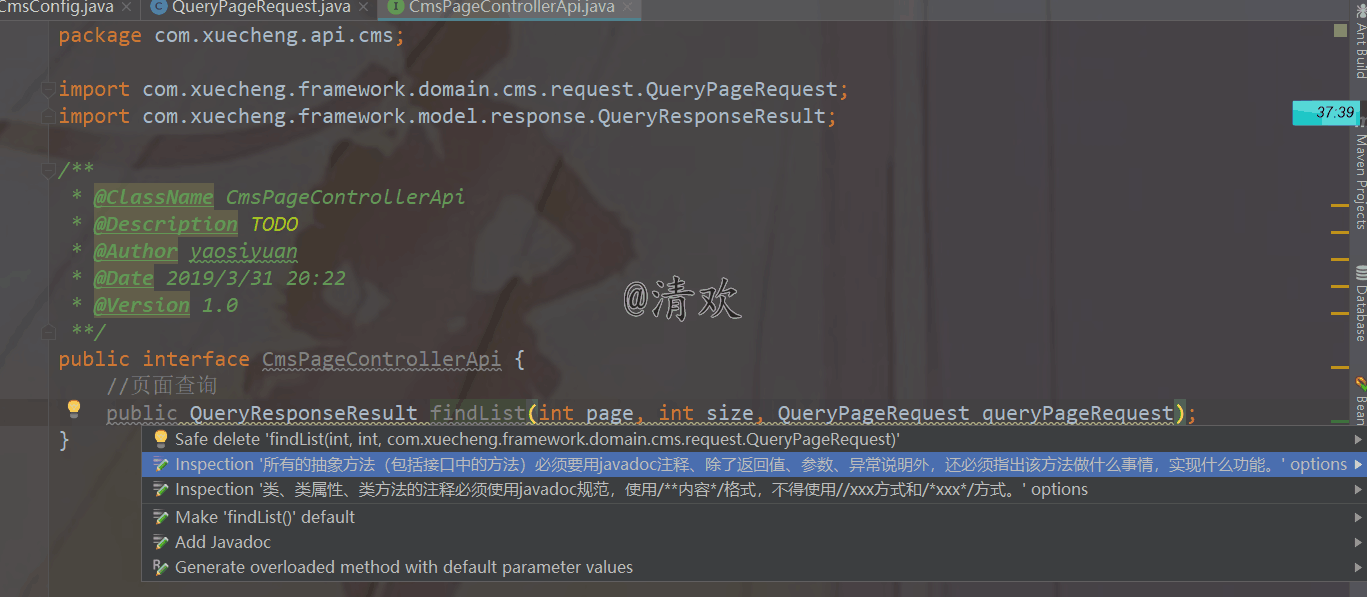
创建方法，在方法上面写：/\*+模板名+Enter-->/\*\*+Enter



## 插件安装

### Alibaba Java Coding Guidelines（必备）

https://github.com/alibaba/p3c



### ctivate-power-mode for IDEA.

### ViceFantasyPlace

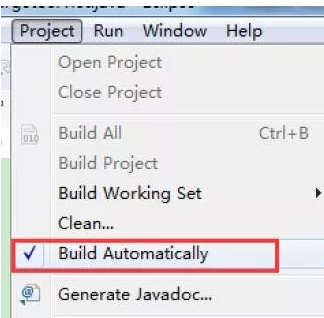
### Power Mode II

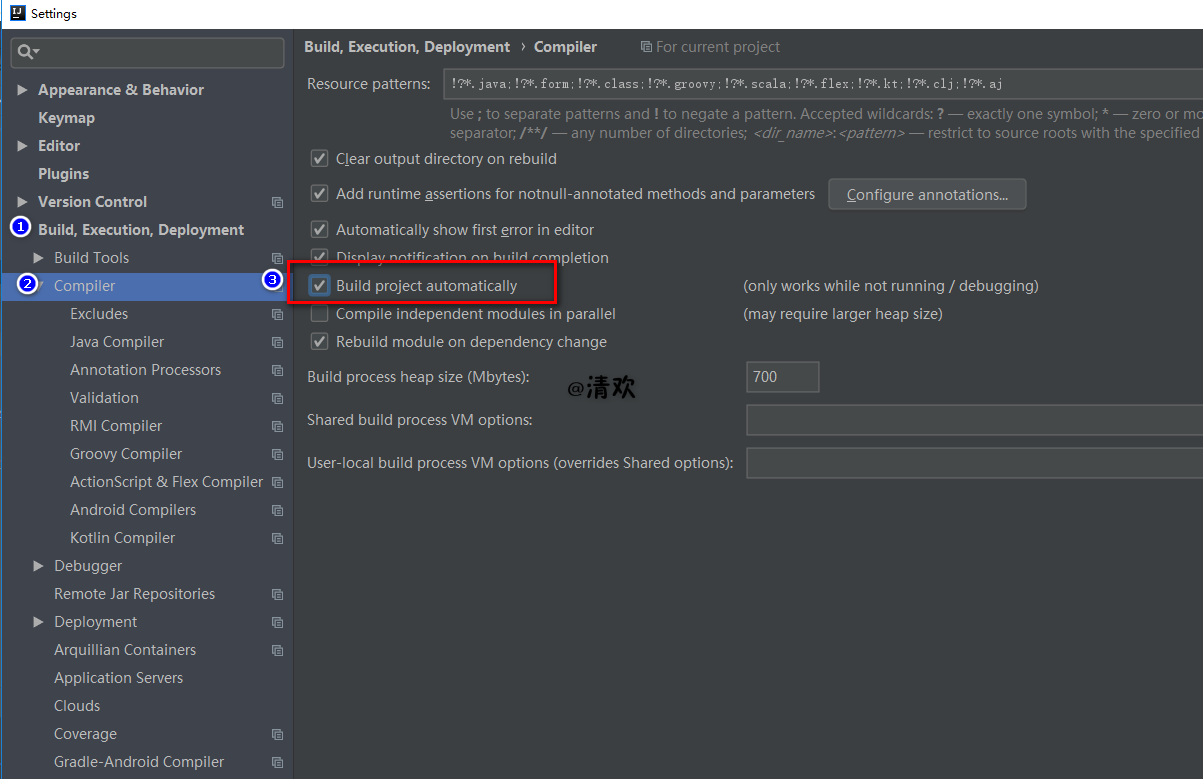
### Activate-power-mode

### BackgroundImagePlus

## 自动编译开关

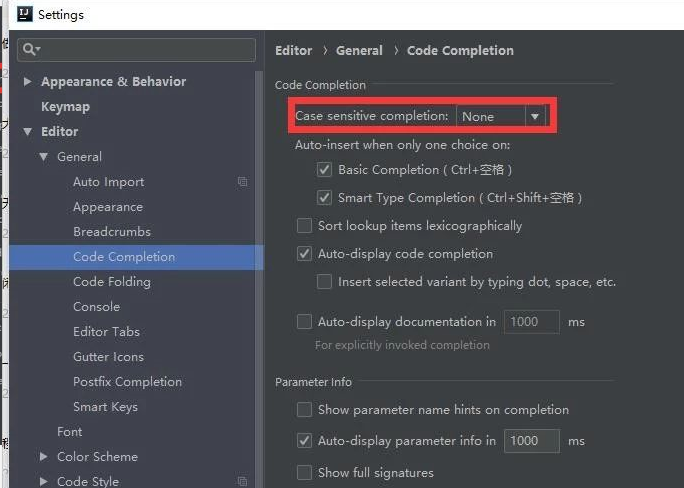
在Eclipse中自动编译开关是开着的，如下所示

  
那么，在IDEA中，务必要手动将其打开，非常重要！如下所示



## 忽略大小写开关

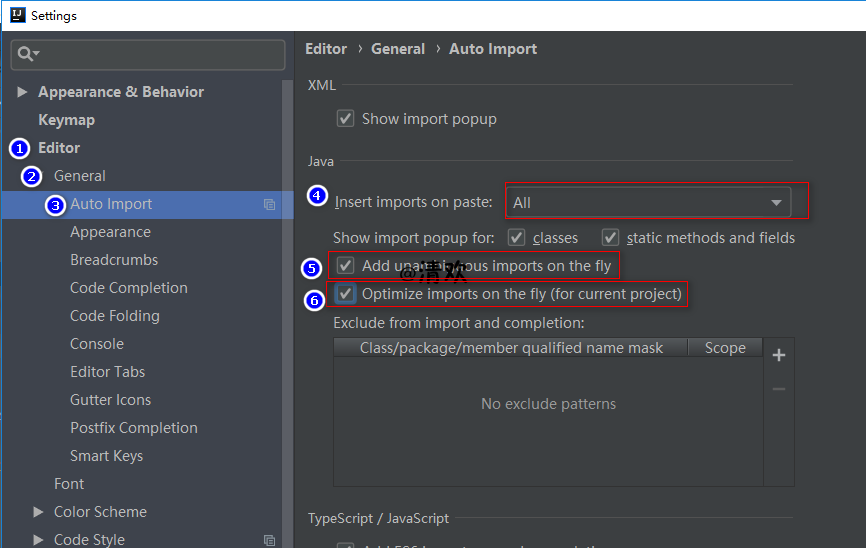
IDEA默认是匹配大小写，此开关如果未关。你输入字符一定要符合大小写。比如你敲string是不会出现代码提示或智能补充。  
但是，如果你开了这个开关，你无论输入String或者string都会出现代码提示或者智能补充！



## 智能导包开关

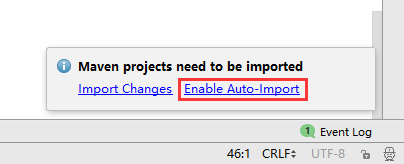
如下图所示，将自动导入不明确的结构智能优化包

这两个选项勾上。那么有什么效果呢？  
你在代码中，只要敲list，就会出现提示，自动导入java.util.List这个类。而这个特性，在eclipse中，是无法做到的。如下图所示

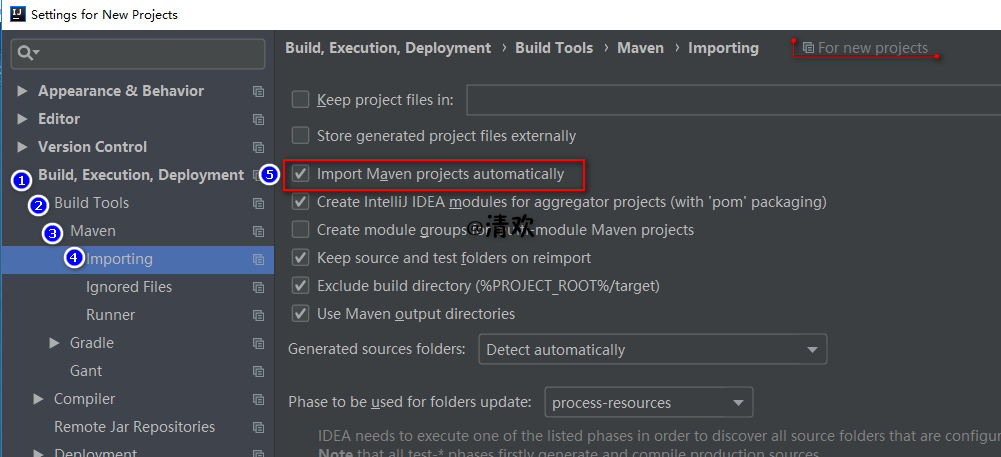


## maven自动导入依赖

第一种方式：首次在pom文件中添加依赖时，IDEA右下角会弹框提示是否开启自动导入，点击开启即可。



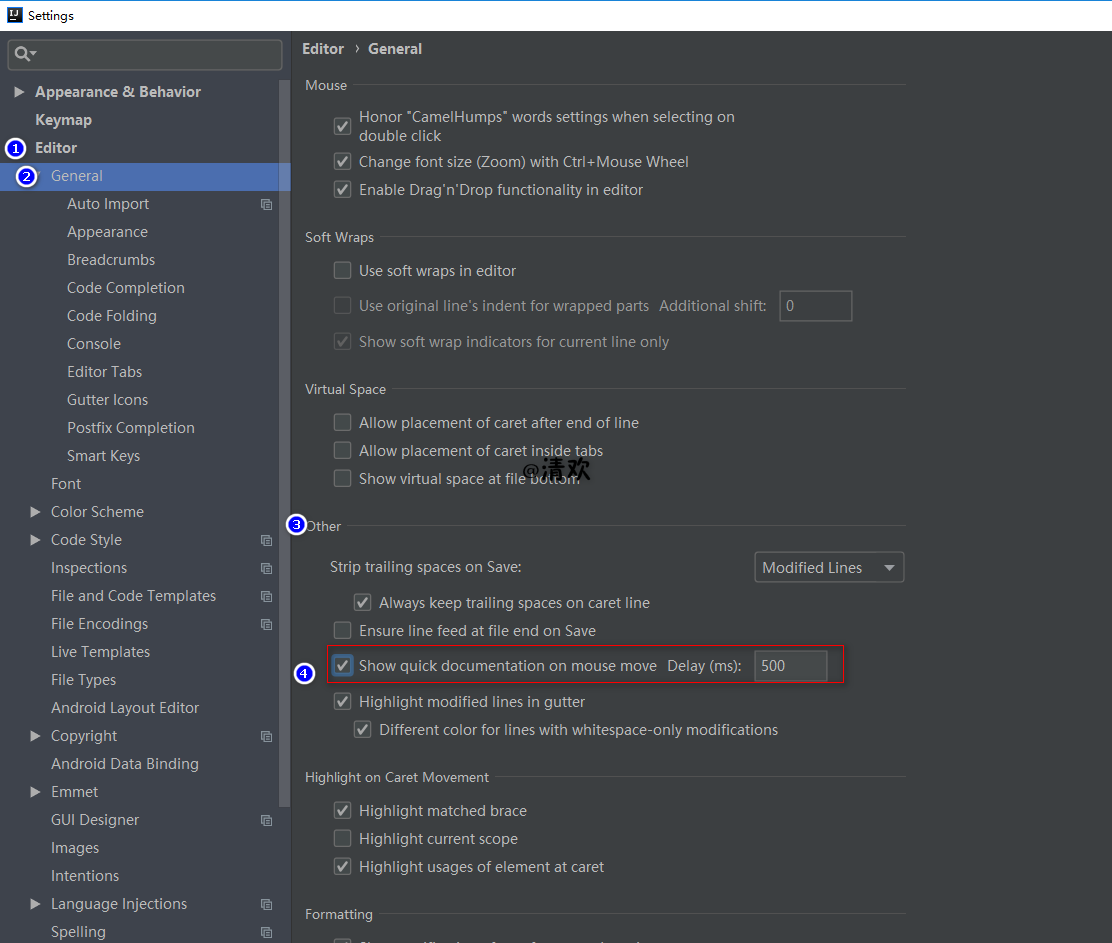
* 第二种方式：手动设置  
  File → Other Settings → Default Settings → Importing → Import Maven projects automatically → OK



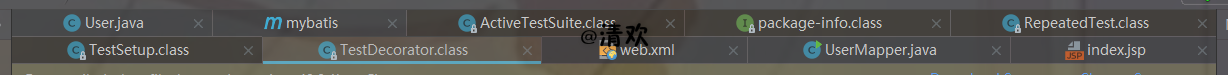
备注：设置的时候右上角一定要是for new projects，也就是全局设置否则只为当前项目，临时设置。新建项目配置无效，需要重新配置。

## 悬浮提示开关

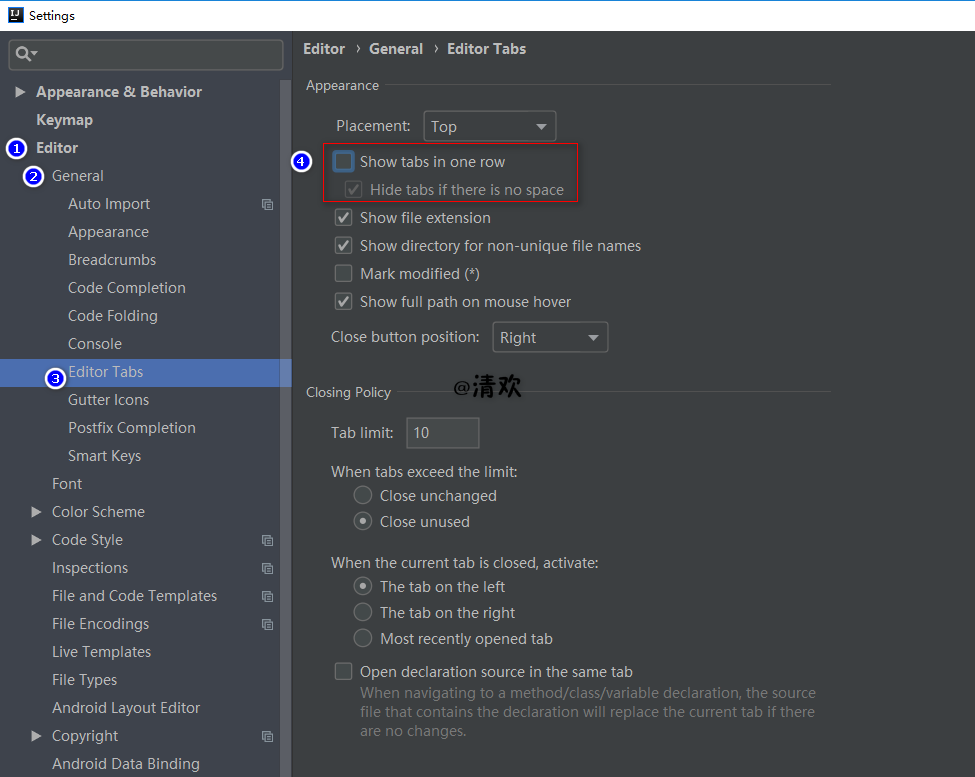
这个容易理解，打开这个开关后。只要把鼠标放在相应的类上，就会出现提示，如下图所示



## 取消单行显示tabs的操作



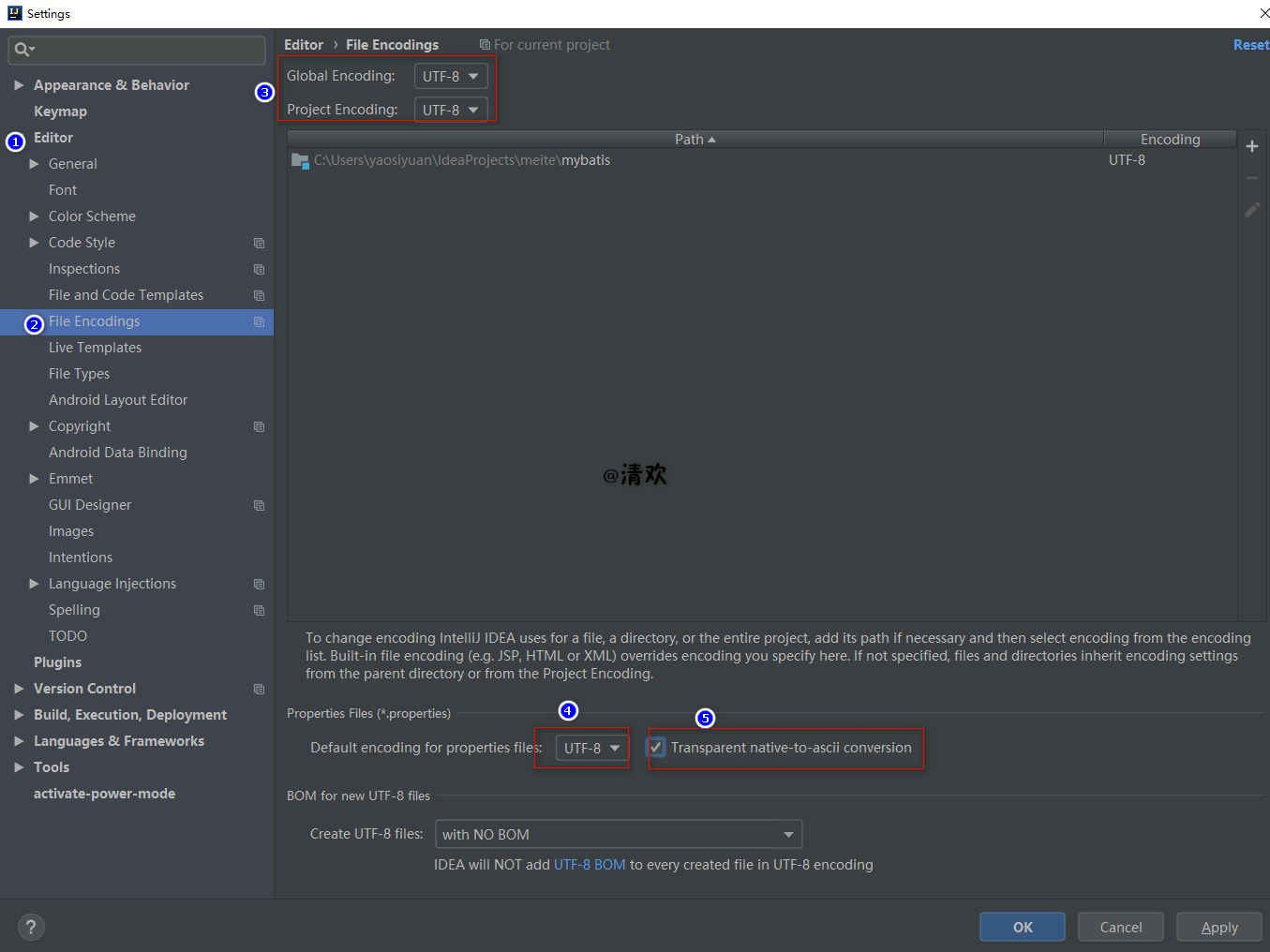
当打开多个文件的时候，会换行显示，非常直观。大大提升效率！



## 项目文件编码

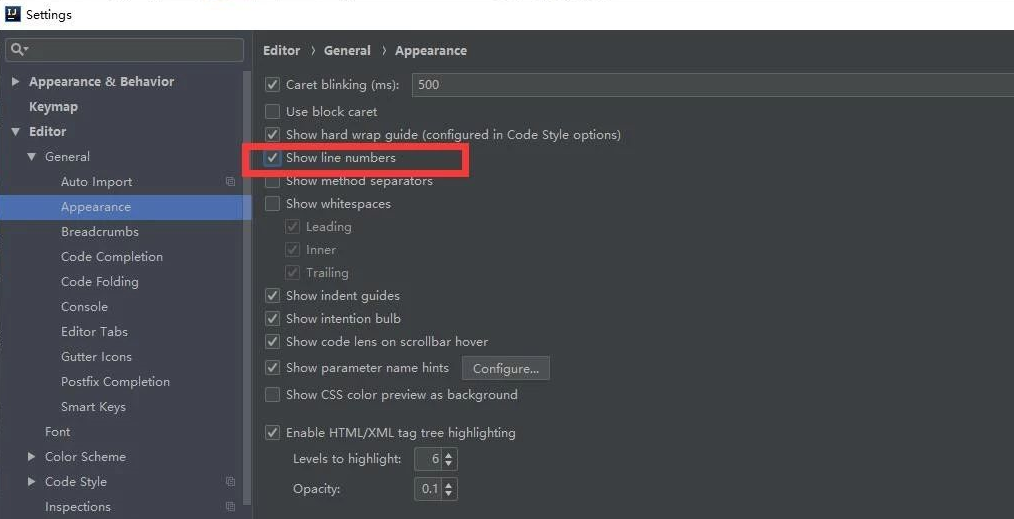
如下图所示进行设置

Transparent native-to-ascii conversion的意思是：自动转换ASCII编码。  
他的工作原理是：在文件中输入文字时他会自动的转换为Unicode编码，然后在idea中发开文件时他会自动转回文字来显示。  
这样做是为了防止文件乱码。  
这样你的properties文件，一般都不会出现中文乱码！



## 设置行号显示

这个的重要性就不用多说了，勾上后代码中，会显示行数!



## 自动根据屏幕代码换行

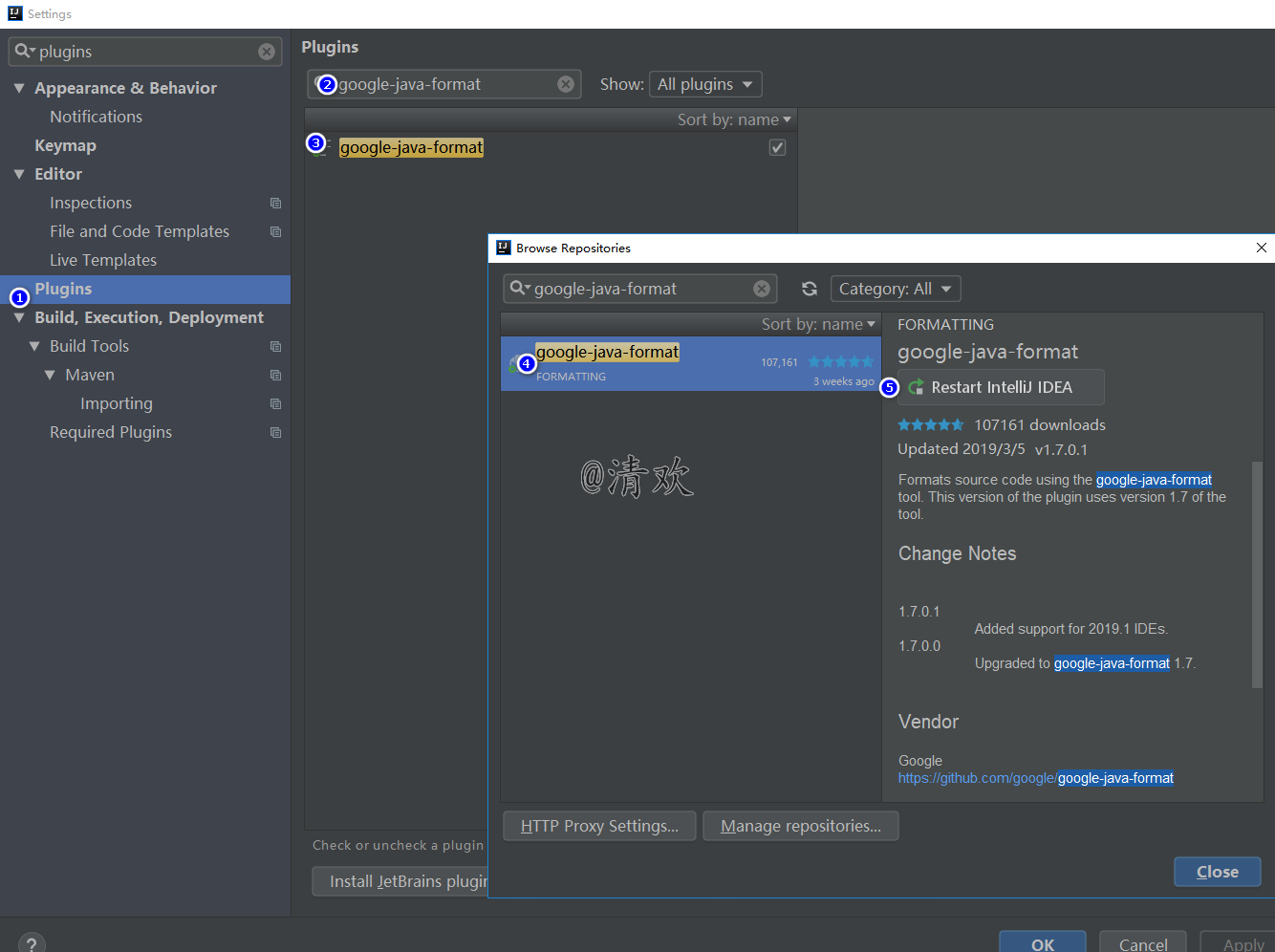


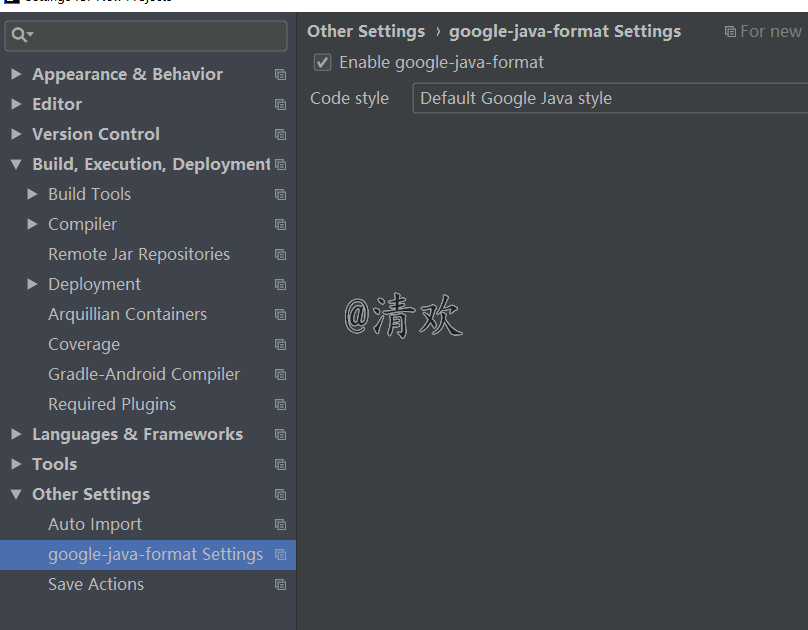
## 自动整理

因为自动整理会导致自动注释格式丢失，目前此自动整理已弃用。

1. 安装google-java-format

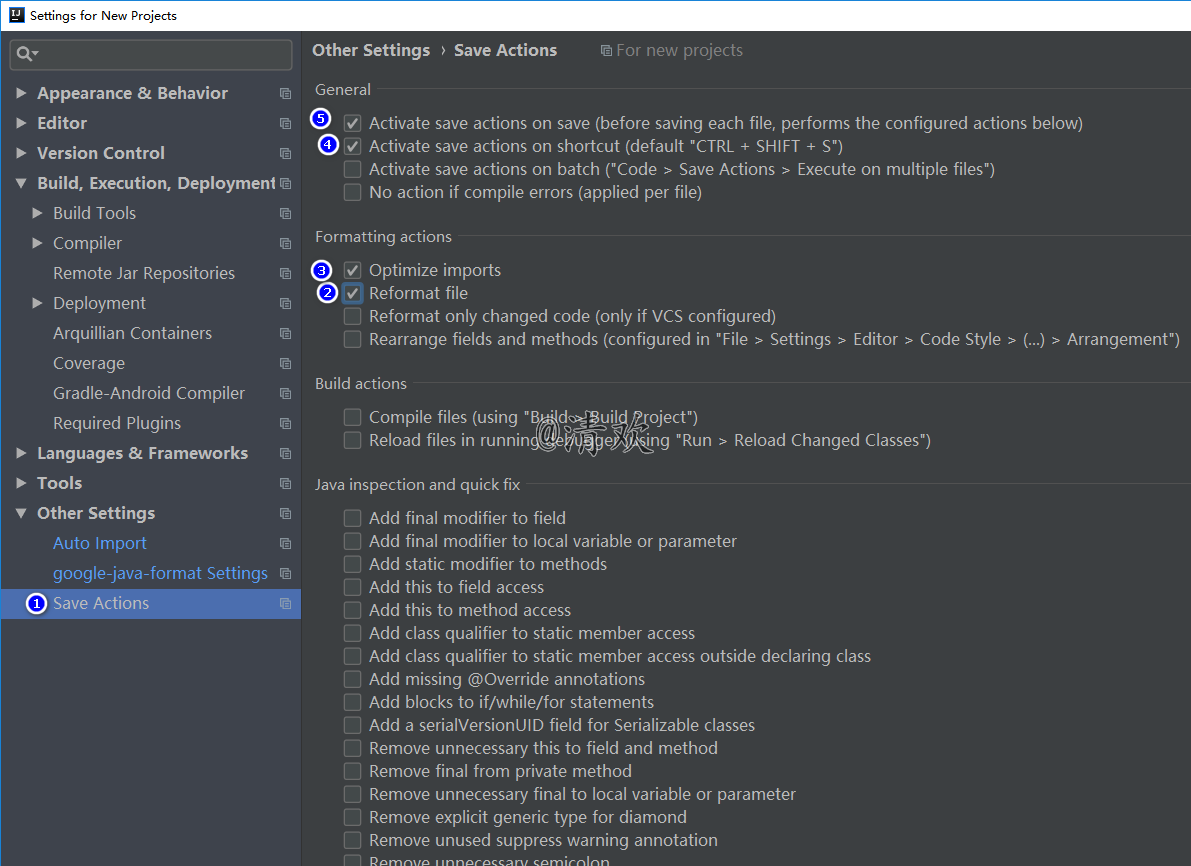
搜索google-java-format



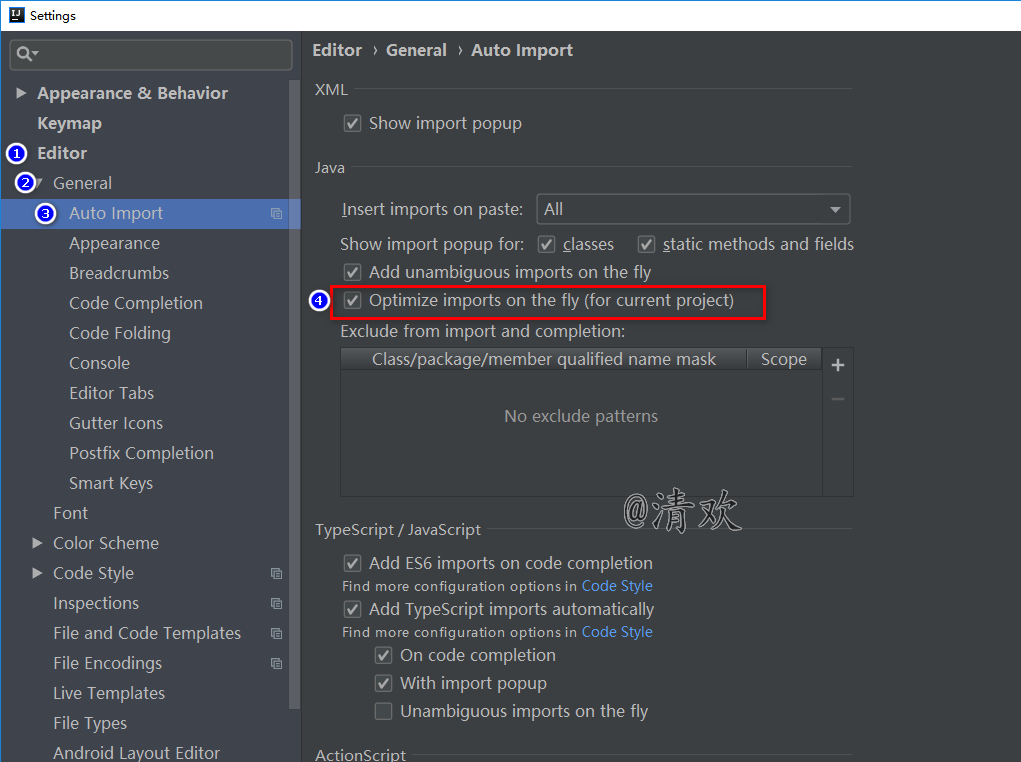
重启idea后启用google-java-format   


1. 安装save actions

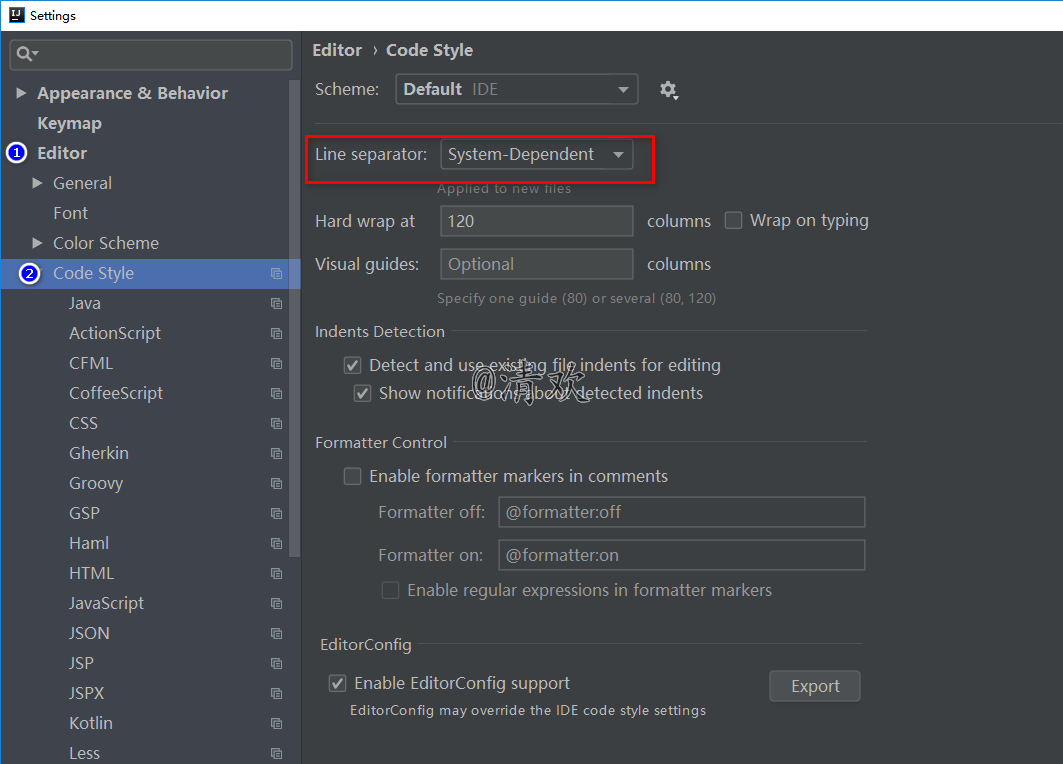
搜索save actions  
重启idea后启用save actions

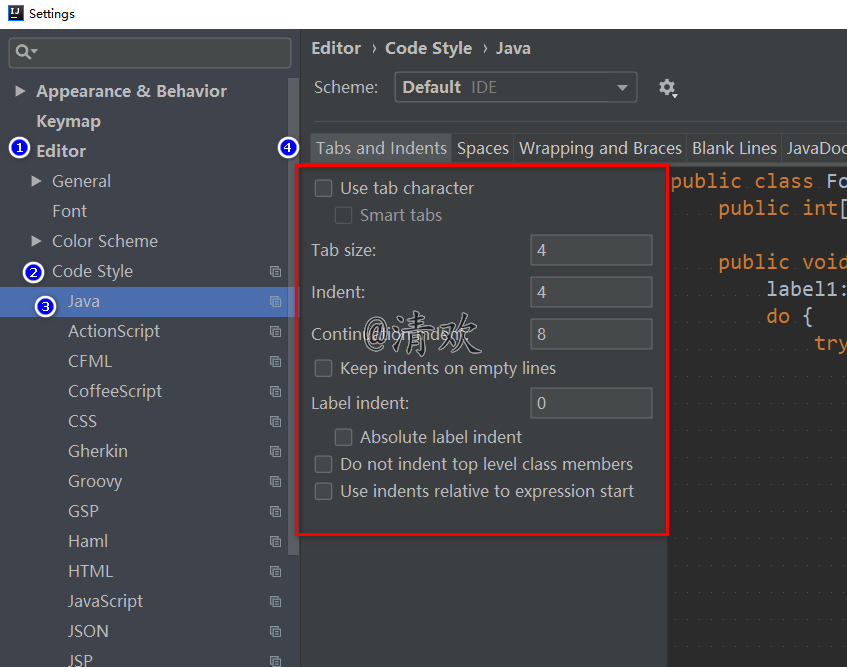


保存时自动格式化   
将没用的import 语句删除

三、自动优化导包选项   


四、检查行分隔符/缩进大小

行分隔符需要统一，不得使用windows   


缩进大小需要统一   


# 快捷键

ctrl + alt + v 自动填充变量

idea中选中一个类名,然后右键 copy reference

在需要包名+类名的地方,右键 paste simple

CTRL+ alt + v 自动创建变量

IntelliJ Idea 常用快捷键列表

Ctrl+Alt+L，格式化代码

Ctrl+Shift + Enter，语句完成

“！”，否定完成，输入表达式时按 “！”键

Ctrl+E，最近的文件

Ctrl+Shift+E，最近更改的文件

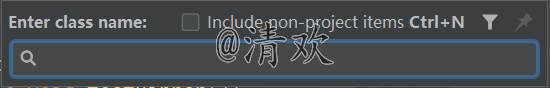
Shift+Click，可以关闭文件

Ctrl+[ OR ]，可以跑到大括号的开头与结尾

Ctrl+F12，可以显示当前文件的结构

Ctrl+F7，可以查询当前元素在当前文件中的引用，然后按 F3 可以选择

Ctrl+N，可以快速打开类



Ctrl+Shift+N，可以快速打开文件

Alt+Q，可以看到当前方法的声明

Ctrl+P，可以显示参数信息

Ctrl+Shift+Insert，可以选择剪贴板内容并插入

Alt+Insert，可以生成构造器/Getter/Setter等

Ctrl+Alt+V，可以引入变量。例如：new String(); 自动导入变量定义

Ctrl+Alt+T，可以把代码包在一个块内，例如：try/catch

Ctrl+Enter，导入包，自动修正

Ctrl+Alt+I，将选中的代码进行自动缩进编排，这个功能在编辑 JSP 文件时也可以工作

Ctrl+Alt+O，优化导入的类和包

Ctrl+R，替换文本

Ctrl+F，查找文本

Ctrl+Shift+Space，自动补全代码

Ctrl+空格，代码提示（与系统输入法快捷键冲突）

Ctrl+Shift+Alt+N，查找类中的方法或变量

Alt+Shift+C，最近的更改

Alt+Shift+Up/Down，上/下移一行

Shift+F6，重构 – 重命名

Ctrl+X，删除行

Ctrl+D，复制行

Ctrl+/或Ctrl+Shift+/，注释（//或者/\*\*/）

Ctrl+J，自动代码（例如：serr）

Ctrl+Alt+J，用动态模板环绕

Ctrl+H，显示类结构图（类的继承层次）

Ctrl+Q，显示注释文档

Alt+F1，查找代码所在位置

Alt+1，快速打开或隐藏工程面板

Ctrl+Alt+left/right，返回至上次浏览的位置

Alt+left/right，切换代码视图

Alt+Up/Down，在方法间快速移动定位

Ctrl+Shift+Up/Down，向上/下移动语句

F2 或 Shift+F2，高亮错误或警告快速定位

Tab，代码标签输入完成后，按 Tab，生成代码

Ctrl+Shift+F7，高亮显示所有该文本，按 Esc 高亮消失

Alt+F3，逐个往下查找相同文本，并高亮显示

Ctrl+Up/Down，光标中转到第一行或最后一行下

Ctrl+B/Ctrl+Click，快速打开光标处的类或方法（跳转到定义处）

Ctrl+Alt+B，跳转到方法实现处

Ctrl+Shift+Backspace，跳转到上次编辑的地方

Ctrl+O，重写方法

Ctrl+Alt+Space，类名自动完成

Ctrl+Alt+Up/Down，快速跳转搜索结果

Ctrl+Shift+J，整合两行

Alt+F8，计算变量值

Ctrl+Shift+V，可以将最近使用的剪贴板内容选择插入到文本

Ctrl+Alt+Shift+V，简单粘贴

Shift+Esc，不仅可以把焦点移到编辑器上，而且还可以隐藏当前（或最后活动的）工具窗口

F12，把焦点从编辑器移到最近使用的工具窗口

Shift+F1，要打开编辑器光标字符处使用的类或者方法 Java 文档的浏览器

Ctrl+W，可以选择单词继而语句继而行继而函数

Ctrl+Shift+W，取消选择光标所在词

Alt+F7，查找整个工程中使用地某一个类、方法或者变量的位置

Ctrl+I，实现方法

Ctrl+Shift+U，大小写转化

Ctrl+Y，删除当前行

Shift+Enter，向下插入新行

psvm/sout，main/System.out.println(); Ctrl+J，查看更多

Ctrl+Shift+F，全局查找

Ctrl+F，查找/Shift+F3，向上查找/F3，向下查找

Ctrl+Shift+S，高级搜索

Ctrl+U，转到父类

Ctrl+Alt+S，打开设置对话框

Alt+Shift+Inert，开启/关闭列选择模式

Ctrl+Alt+Shift+S，打开当前项目/模块属性

Ctrl+G，定位行

Alt+Home，跳转到导航栏

Ctrl+Enter，上插一行

Ctrl+Backspace，按单词删除

Ctrl+”+/-”，当前方法展开、折叠

Ctrl+Shift+”+/-”，全部展开、折叠

【调试部分、编译】

Ctrl+F2，停止

Alt+Shift+F9，选择 Debug

Alt+Shift+F10，选择 Run

Ctrl+Shift+F9，编译

Ctrl+Shift+F10，运行

Ctrl+Shift+F8，查看断点

F8，步过

F7，步入

Shift+F7，智能步入

Shift+F8，步出

Alt+Shift+F8，强制步过

Alt+Shift+F7，强制步入

Alt+F9，运行至光标处

Ctrl+Alt+F9，强制运行至光标处

F9，恢复程序

Alt+F10，定位到断点

Ctrl+F8，切换行断点

Ctrl+F9，生成项目

Alt+1，项目

Alt+2，收藏

Alt+6，TODO

Alt+7，结构

Ctrl+Shift+C，复制路径

Ctrl+Alt+Shift+C，复制引用，必须选择类名

Ctrl+Alt+Y，同步

Ctrl+~，快速切换方案（界面外观、代码风格、快捷键映射等菜单）

Shift+F12，还原默认布局

Ctrl+Shift+F12，隐藏/恢复所有窗口

Ctrl+F4，关闭

Ctrl+Shift+F4，关闭活动选项卡

Ctrl+Tab，转到下一个拆分器

Ctrl+Shift+Tab，转到上一个拆分器

【重构】

Ctrl+Alt+Shift+T，弹出重构菜单

Shift+F6，重命名

F6，移动

F5，复制

Alt+Delete，安全删除

Ctrl+Alt+N，内联

【查找】

Ctrl+F，查找

Ctrl+R，替换

F3，查找下一个

Shift+F3，查找上一个

Ctrl+Shift+F，在路径中查找

Ctrl+Shift+R，在路径中替换 全局替换

Ctrl+Shift+S，搜索结构

Ctrl+Shift+M，替换结构

Alt+F7，查找用法

Ctrl+Alt+F7，显示用法

Ctrl+F7，在文件中查找用法

Ctrl+Shift+F7，在文件中高亮显示用法

# 代码

## [Idea＋maven＋tomcat部署第一个tomcat项目](https://www.cnblogs.com/esther-qing/p/7411984.html)

IDEA创建Maven项目及部署发布，IDEA配置Tomcat，创建java源文件夹。

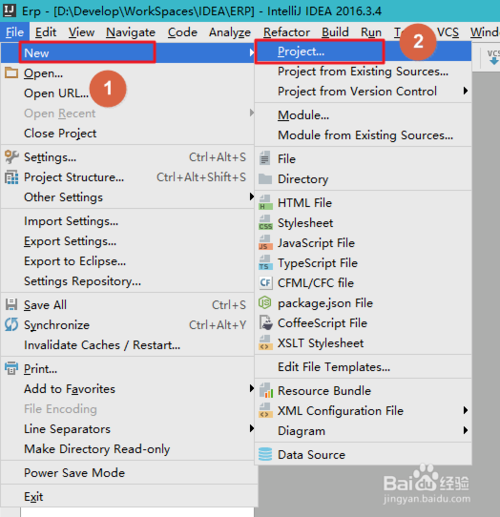
此教程适合刚刚使用IDEA的新手。

### 工具/原料

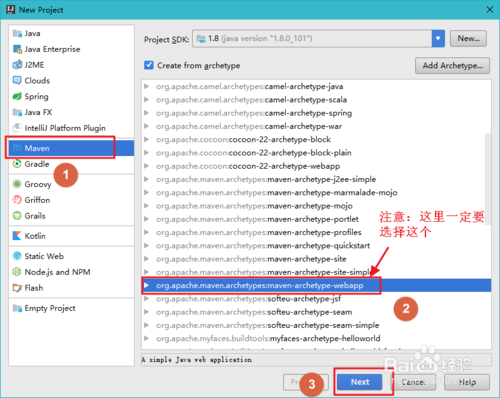
* IntelliJ IDEA 2016.3.4
* apache-maven-3.3.9
* apache-tomcat-8.5.11

### 创建Maven项目

如下图，打开idea之后，file -> new -> project

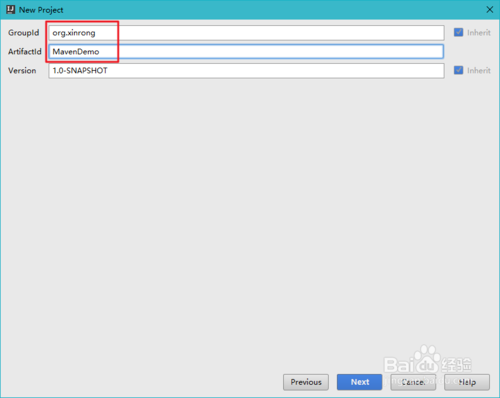


如下图，在弹出的new project 页面，选择maven -> 勾选Create from artifactype -> 选中webapp（注意：这个地方有两个app，选择maven-archetype-webapp）, next

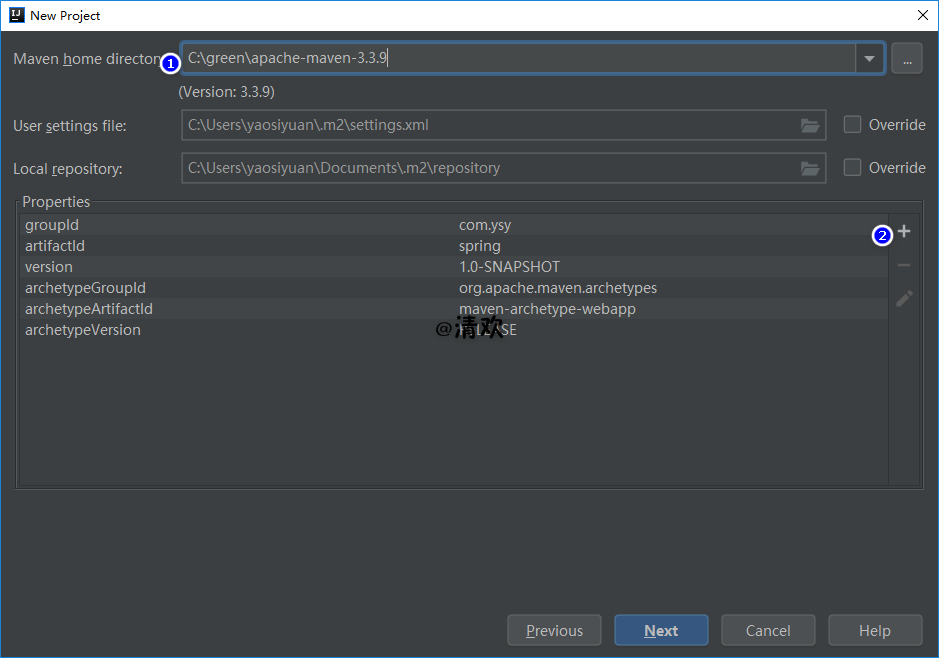


如下图，接下来就是填写groupId 和archetypeId 了，点击next。

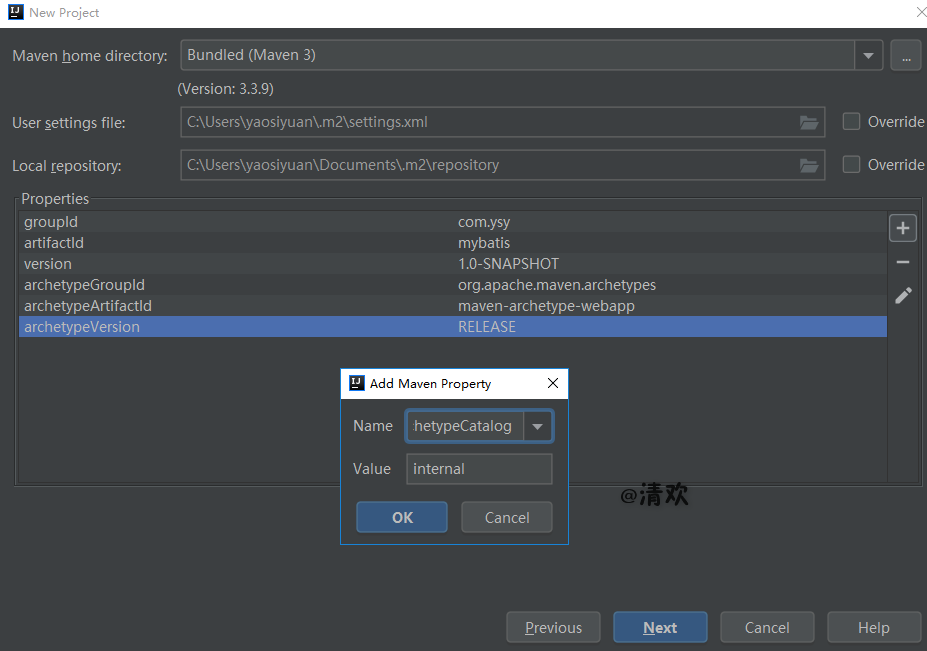
GroupID实际对应JAVA的包的结构，是main目录里java的目录结构。 ArtifactID就是项目的唯一的标识符，实际对应项目的名称，就是项目根目录的名称。



如下图，这一步，直接按照默认，点击next也可以的，但是可能创建完成之后会下载一些东西，然后导致比较慢，所以我个人建议。先配置maven路径后再点击右侧的那个加号

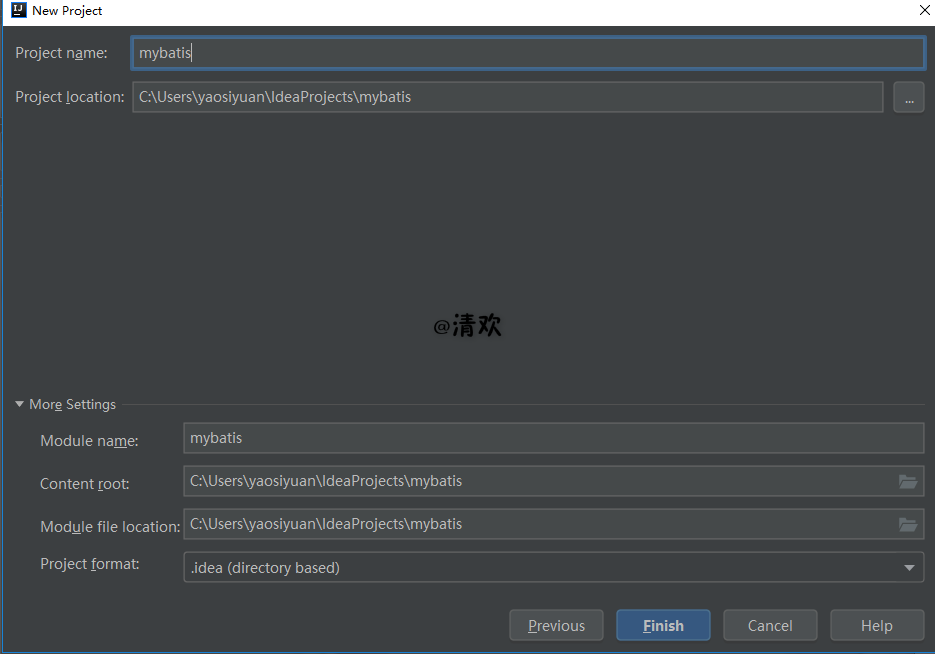


输入对应的name和value，然后点击OK。点击Next。archetypeCatalog表示插件使用的archetype元数据，不加这个参数时默认为remote，local，即中央仓库archetype元数据，由于中央仓库的archetype太多了，所以导致很慢，指定internal来表示仅使用内部元数据。



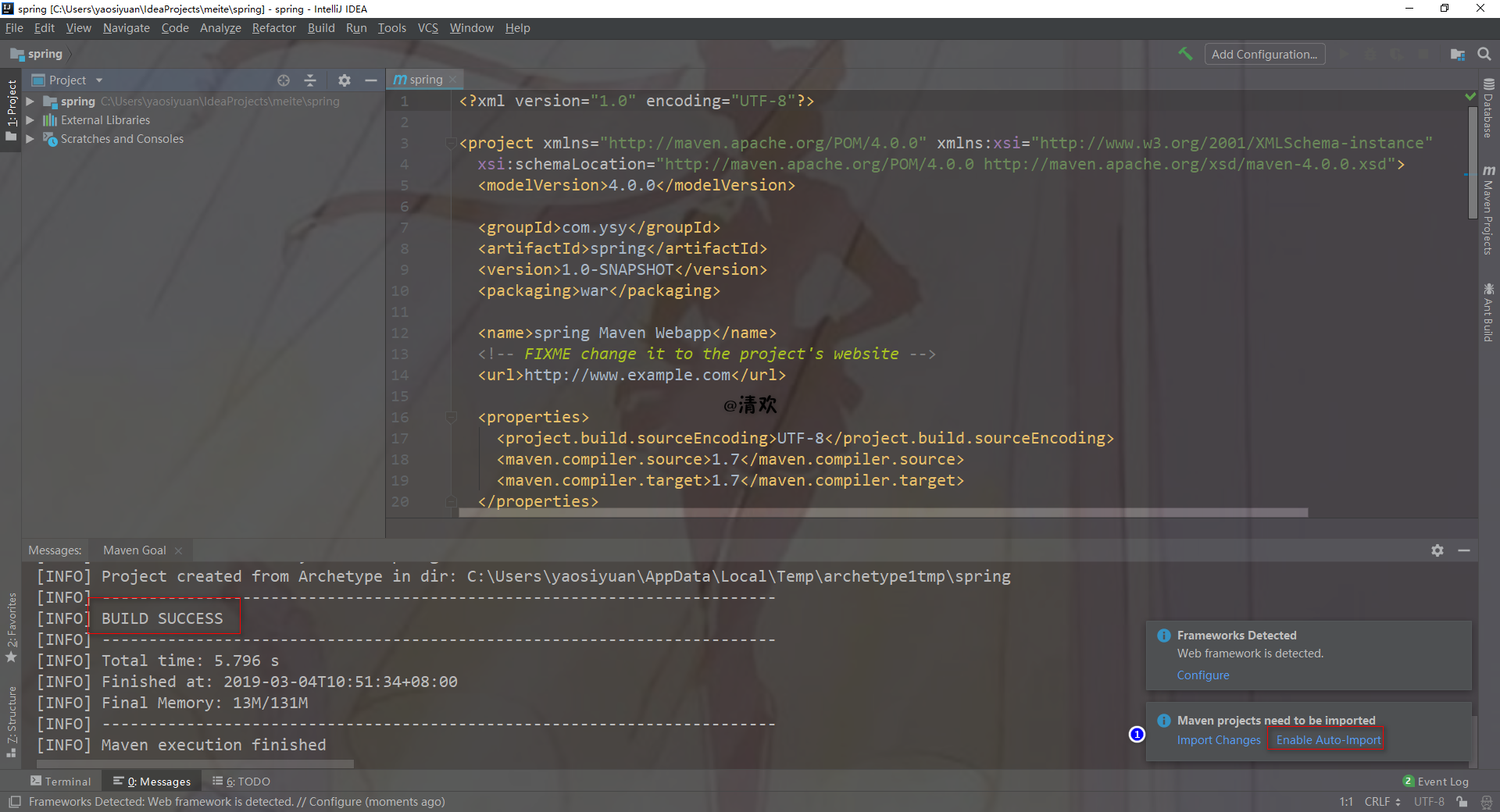
如下图，填写project name和module name，并选择项目所在的路径。

module name默认和project name 一样的，但是如果这个项目只有这一个module，不改无所谓的，如果有很多的module，那肯定是修改一下。 finish.



如下图，别急，稍等片刻，待目录结构变成图示的结构，说明项目创建完成。

项目创建完了还不一定能运行，要看有没有配置好tomcat和jdk了。 具体见步骤二。

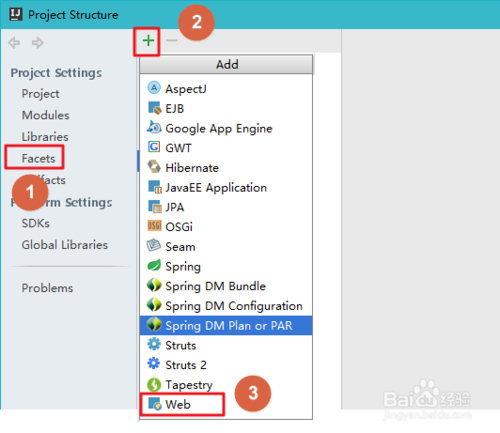


### 配置Tomcat

在主页面打开ProjectStructure，点击图示的按钮或是按ctrl+shift+alt+s快捷键。

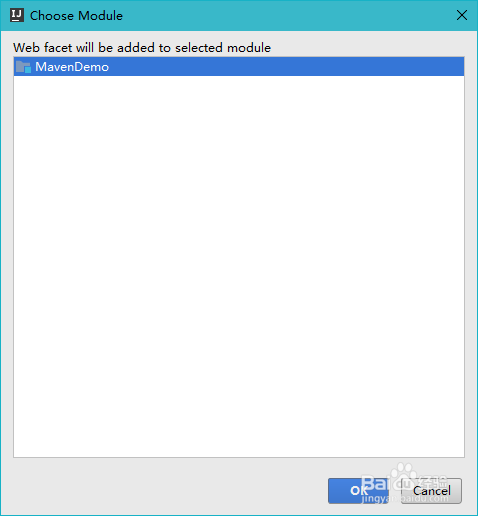


打开ProjectStructure后，按照图示依次点击Facets->+号,在弹出的对话框个中选择Web.

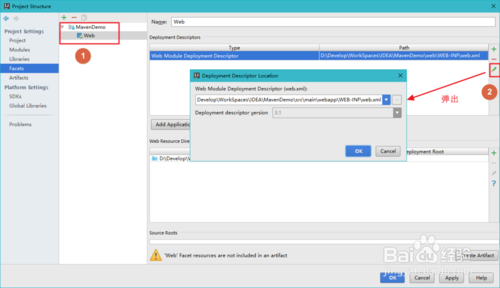


1. 3

在弹出的对话框选择刚才创建的Maven项目（MavenDemo），点击ok.

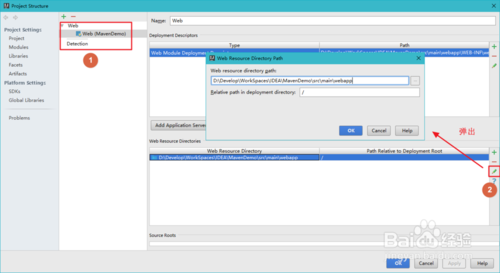


接着会出现图示的结构，点击Deployment Descriptors 对话框右侧的加号，会弹出一个图示的小对话框，选择web.xml所在的路径，点击Ok。



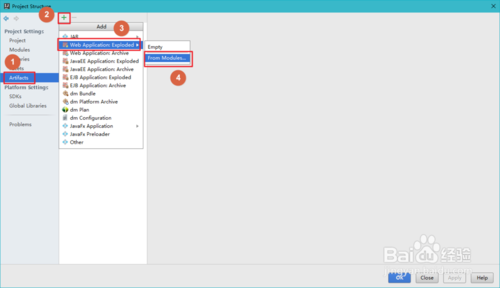
1. 5

再次点击Web Resource Directories对话框右侧的加号，会弹出一个图示的小对话框，选择webapp所在的路径，点击Ok。



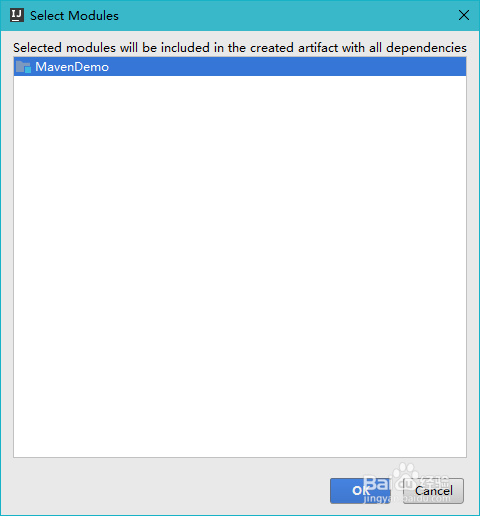
1. 6

再次回到Project Structure,点击左侧 Artifacts，依次点击+号->Web Application:Exploded->From Modules.



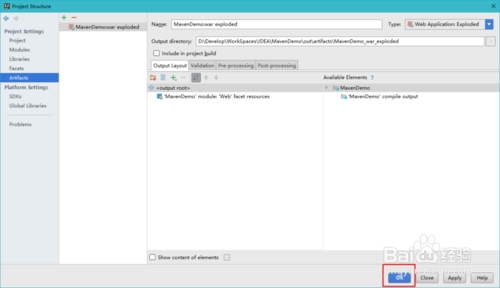
1. 7

在弹出的框中选择刚才创建的Maven项目（MavenDemo），Ok.



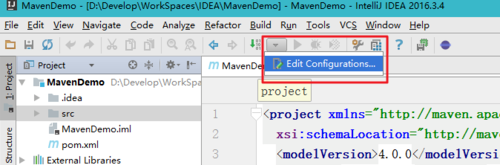
1. 8

出现下图的对话框，点击Ok即可。



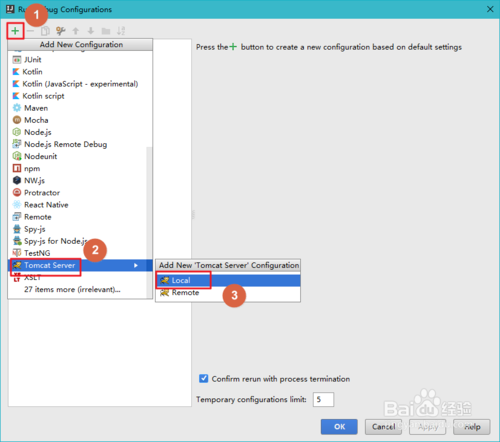
1. 9

接下来开始配置Tomcat。然后在右上角点击那个下拉框，再点击Edit Configurations



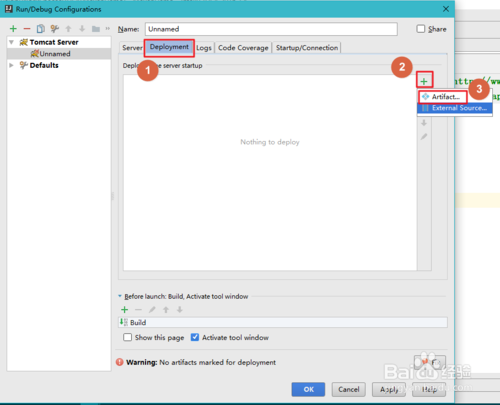
1. 10

在弹出的对话框中点击+号，找到Tomcat Server ，选择Local.



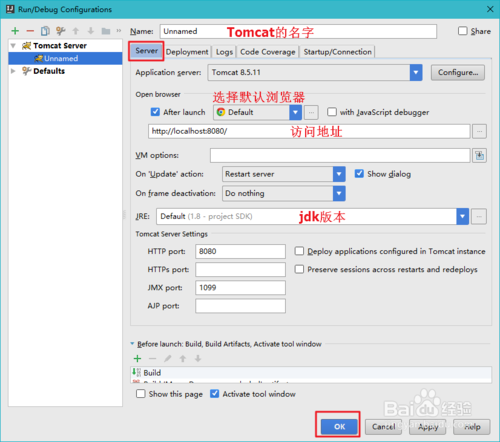
1. 11

在弹出的对话框中，选择Deployment,点击右侧的+号，选择Artifact。



1. 12

接着回到Server选项卡，配置一下Tomcat的基本参数，最后点击Ok即可



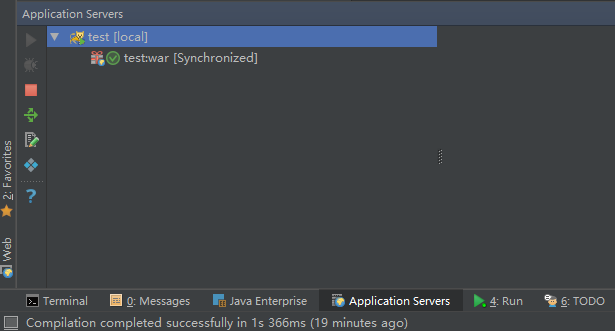
 这分别是在两种命令下的触发的事件。这个是 IntelliJ IDEA 特有的，也是重点

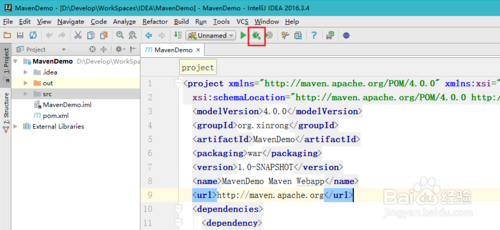
**On Update action：** 当我们按 Ctrl + F10 进行容器更新的时候，可以根据我们配置的这个事件内容进行容器更新。其中我选择的 Update classes and resources 事件是最常用的，表示我们在按 Ctrl + F10 进行容器更新的时候，我们触发更新编译的类和资源文件到容器中。在 Debug 模式下，这个也就是所谓的 HotSwap。这是这种热部署没有 JRebel 插件那样好用

**On frame deactivation：** 当我们切换 IntelliJ IDEA 到浏览器的时候进行指定事件更新，这个一般是因为 Web 开发的时候，我们需要经常在 IntelliJ IDEA 和各个浏览器之间来回切换测试，所以才有了这种需求。IntelliJ IDEA 是为了帮我们在做这种无聊切换的时候做一些指定事情。当然了，如果切换过于频繁，这个功能还是很耗资源的，所以我设置的是 Do nothing 表示切换的时候什么都不做

13.13

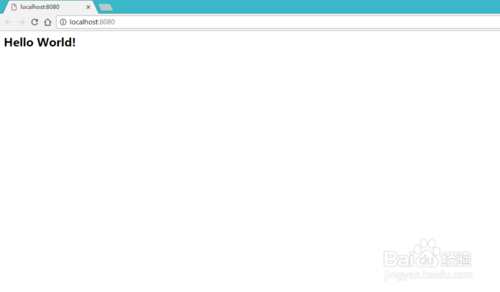
在主界面中 Application Servers 中就可以看到部署的工程，点击 Run/Connect 就可以运行 Tomcat 服务器



最后点击工具栏上的小虫子按钮，以Debug方式启动Tomcat服务器。

1. 14

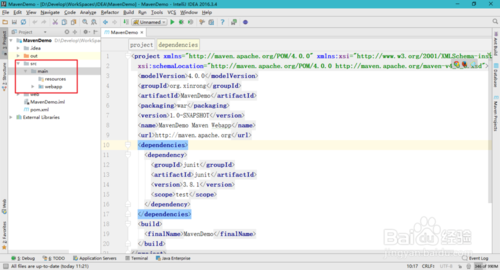
如果看到下图的页面，恭喜你，Maven项目的创建及运行已经成功。



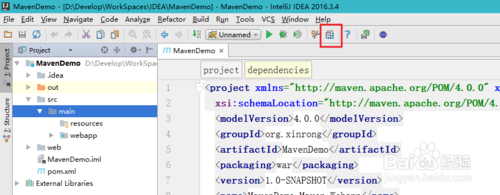
END

### 创建Java文件目录

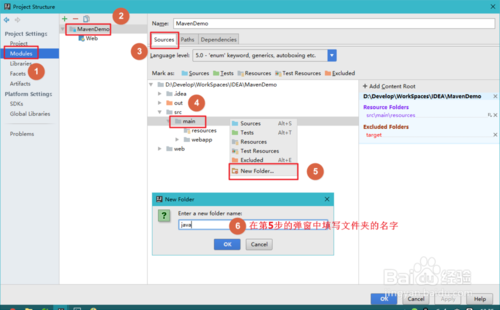
创建及部署完Maven项目会发现，main目录下没有java源文件夹。接下来我会教大家怎么创建这个java源文件夹。



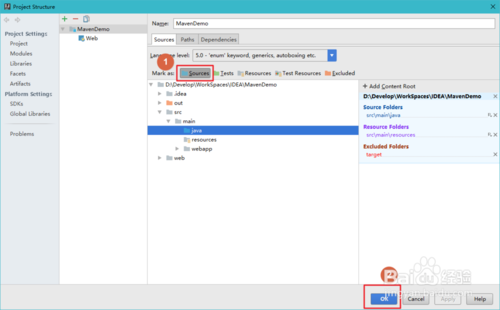
在主页面打开ProjectStructure，点击图示的按钮或是按ctrl+shift+alt+s快捷键



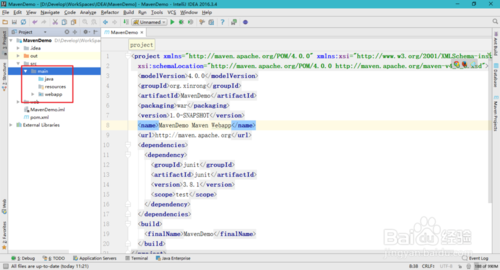
在弹出的Project Structure窗口中，依次从左侧选择Modules->MavenDemo->Sources,打开项目的main目录，在main上单击右键选择New Folder,在弹出的对话框中填写文件的名字后，点击Ok。



然后选中刚才创建的文件夹，点击上面的 Mark as： 中的sources按钮，该文件夹就会变成蓝色用于保存java代码，之后点击OK，



最后主页面的main文件夹下就会出现java代码的源文件夹。



## **使用IntelliJ IDEA 配置Maven（入门）**

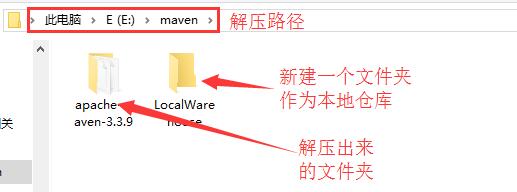
https://blog.csdn.net/qq\_32588349/article/details/51461182

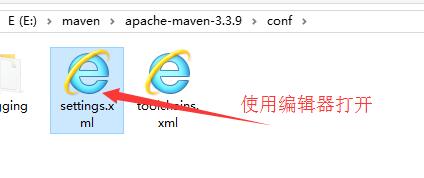
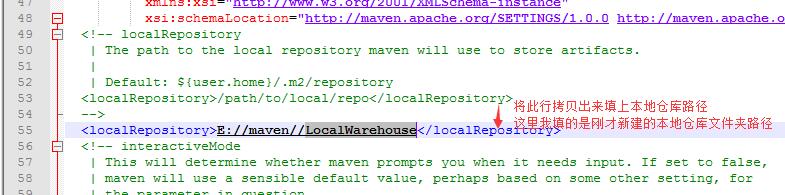
### **下载Maven**

官方地址：<http://maven.apache.org/download.cgi>

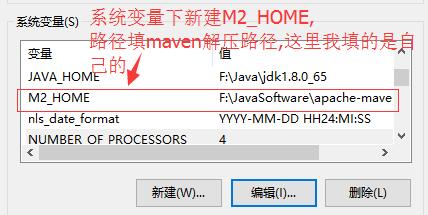
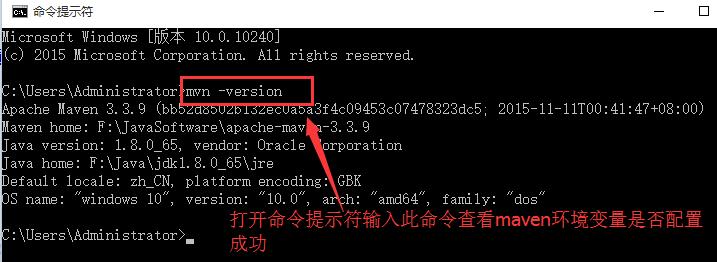


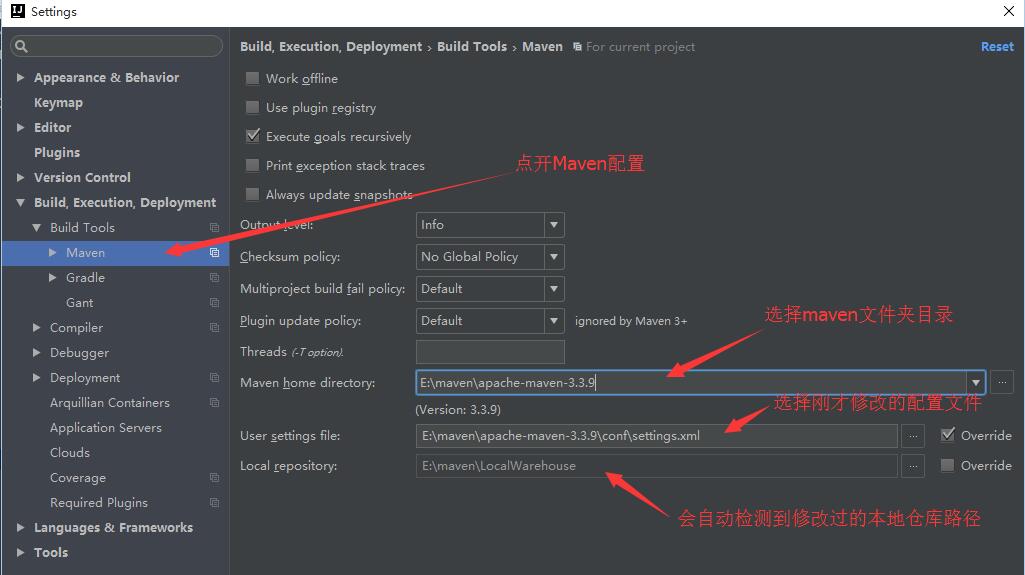
解压并新建一个本地仓库文件夹

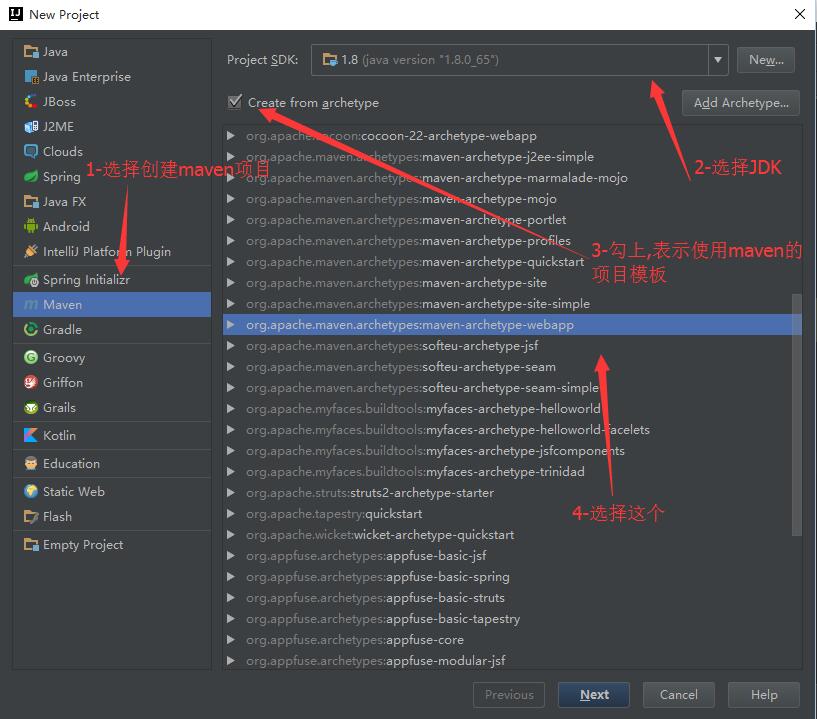
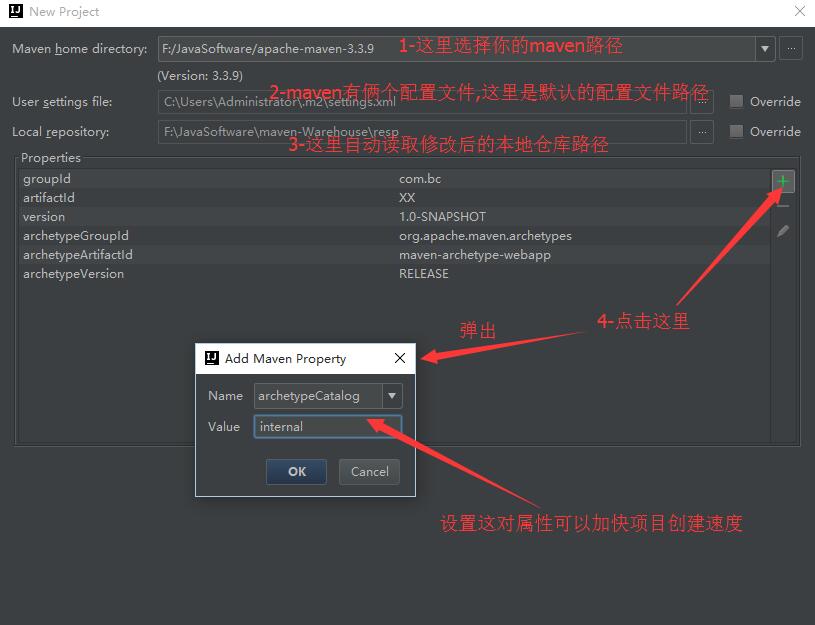
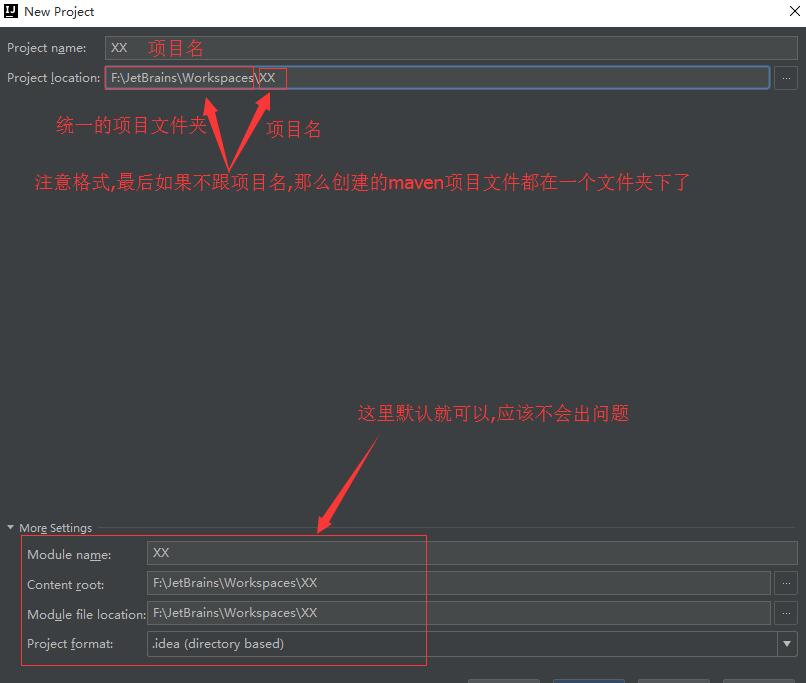
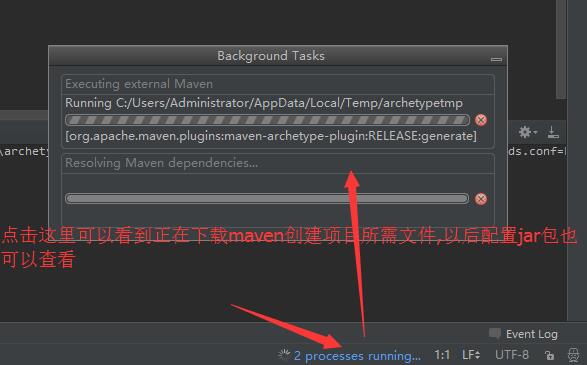


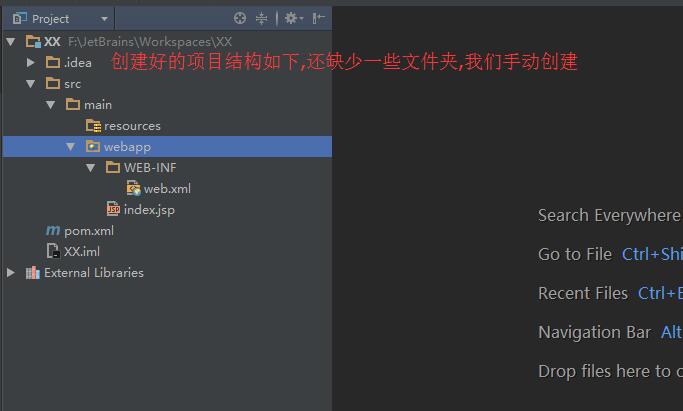
**2.配置本地仓库路径**   
   


**3.配置maven环境变量**

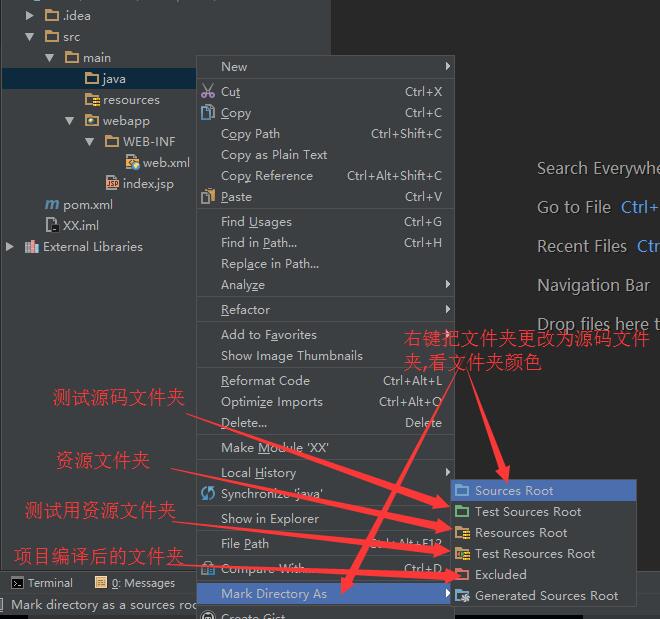
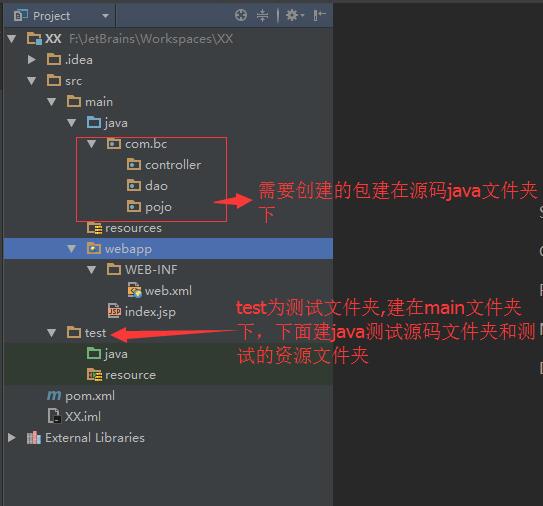
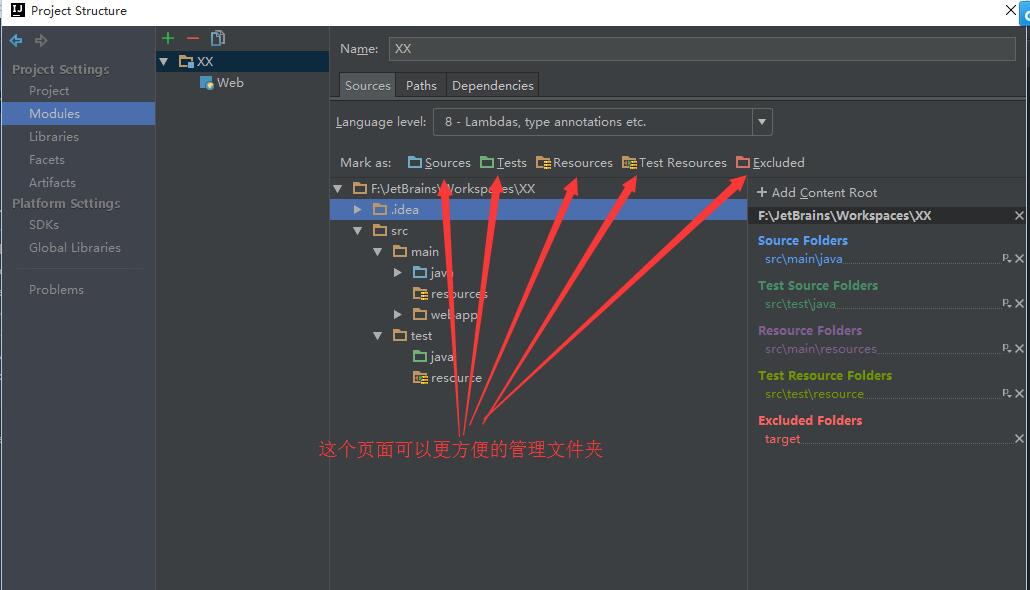
   
   
   
   


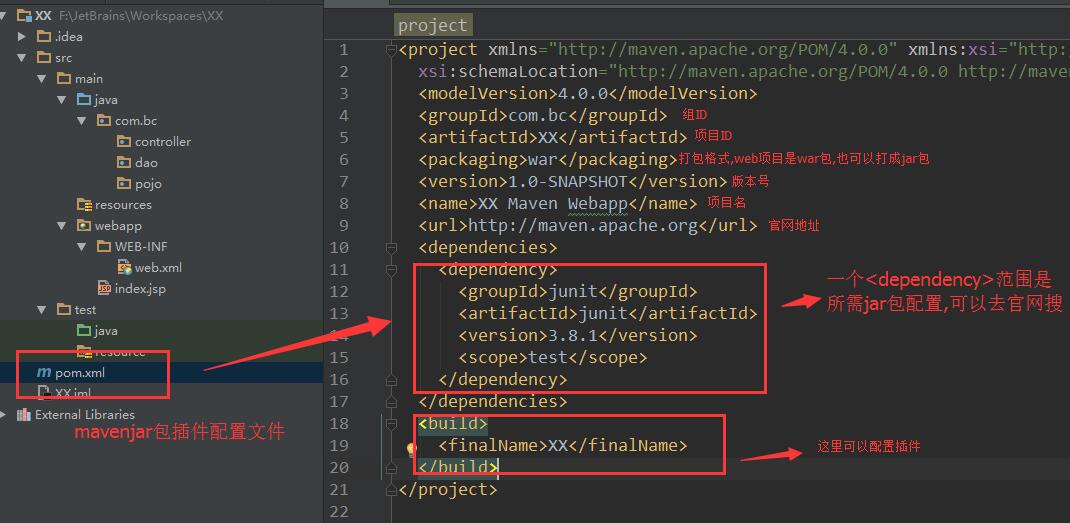
**4.在IntelliJ IDEA中配置maven**   
打开-File-Settings   


**5.新建maven WEB项目**   
打开-File-New-Project   
点击NEXT   
   
点击NEXT   
添加的配置为 archetypeCatalog=internal   
点击NEXT   
   
点击NEXT   
   
点击Finish后项目开始创建   
点击右下角查看进去   


**6.maven web模板项目结构**   




   
同样在main下新建test测试文件夹,再在此文件夹下新建java测试源码文件夹和resource测试资源文件夹   
   
也可以右键项目-选择Open Module Settings打开项目配置页面更改   


**7.配置依赖jar包**   


jar包配置搜索   
官方地址：<http://mvnrepository.com/>

### 创建java jar、pom项目

创建maven-archetype-quickstart

[**Maven的Archetype简介**](https://www.cnblogs.com/LinkinPark/p/5232995.html)

## Archetype，骨架的意思。

文章出处：http://m.blog.csdn.net/blog/FireOfStar/42526027

### Archetype是什么？

简单的说，Archetype是Maven工程的模板工具包。一个Archetype定义了要做的相同类型事情的初始样式或模型。这个名称给我们提供来了一个一致的生成Maven工程的方式。Archetype会帮助作者给用户创建Maven工程模板，并给用户提供生成相关工程模板版本的参数化方法。

使用Archetype提供的好的方法，是开发者能够使用最佳实践来快速的构建和组织一致化的工程。在Maven工程中，我们努力使用Archetype来尽可能快的给用户提供示例工程，同时也会把Maven的最佳实践介绍给新的用户。一个新的用户可以使用工作中的Maven工作作为跳板来研究更过的Maven中功能。我们也可以使用Archetype的添加机制，这样就意味着允许我们抓取Archetype中项目片段，并把它们添加到既存的工程中。Maven网站的Archetype就是很好的例子。例如，你可以使用“quick start archetype”来生成一个工程，然后就可以通过其中既存的“site archetype”来快速的创建一个网址工程。你能够使用Archetype来做很多这样的事情。

在你的团队中可能想要标准化的J2EE开发，这需要你提供EJBs、或者是WARs、或者是Web services的原型。一旦在你团队资源库中创建和部署这些原型，它们就可以在你团队内共享使用。

如何使用Archetype

要基于Archetype来创建一个新的工程，需要像下面示例这样来调用：

mvn archetype:generate

已有的Archetypes

|  |  |
| --- | --- |
| Archetype ID | 说明 |
| maven-archetype-archetype | 一个样例原型 |
| maven-archetype-j2ee-simple | 简单的J2EE应用程序样例 |
| maven-archetype-mojo | Maven插件样本的示例 |
| maven-archetype-plugin | Maven插件样本 |
| maven-archetype-plugin-site | Mave插件网站的样例 |
| maven-archetype-portlet | JSR-268门户样例 |
| maven-archetype-quickstart | Maven工程样例 |
| maven-archetype-simple | 一个简单的Maven工程 |
| maven-archetype-site | Maven网站的样例，它演示了对诸如APT、XDoc和FML等文档类型的支持，并演示了如果把网站国际化（i18n） |
| maven-archetype-site-simple | Maven网站样例 |
| maven-archetype-webapp | Maven的Webapp工程样例 |

### 常用Archetype

### maven-archetype-quickstart

默认的Archetype,基本内容包括：  
一个包含junit依赖声明的pom.xml  
src/main/java主代码目录及一个名为App的类  
src/test/java测试代码目录及一个名为AppTest的测试用例

### maven-archetype-webap

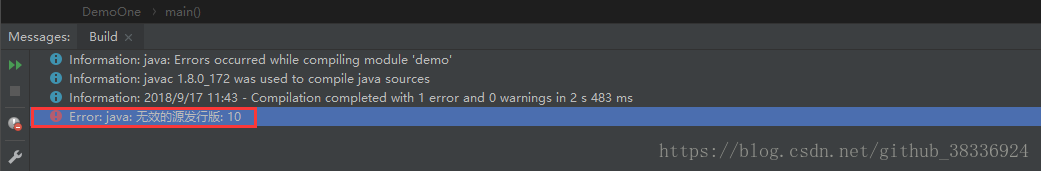
一个最简单的Maven war项目模板，当需要快速创建一个Web应用的时候可以使用它。生成的项目内容包括：  
一个packaging为war且带有junit依赖声明的pom.xml  
src/main/webapp/目录  
src/main/webapp/index.jsp文件  
src/main/webapp/WEB-INF/web.xml文件

# 常见问题

## 无效的源发行版本

问题描述

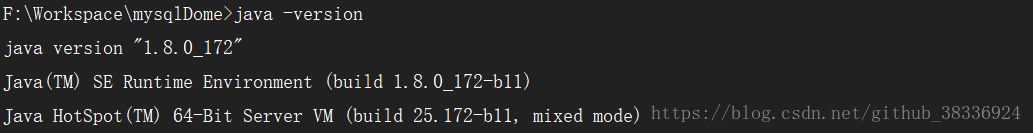
今天编译java程序时出现Error:java: 无效的源发行版: 10错误



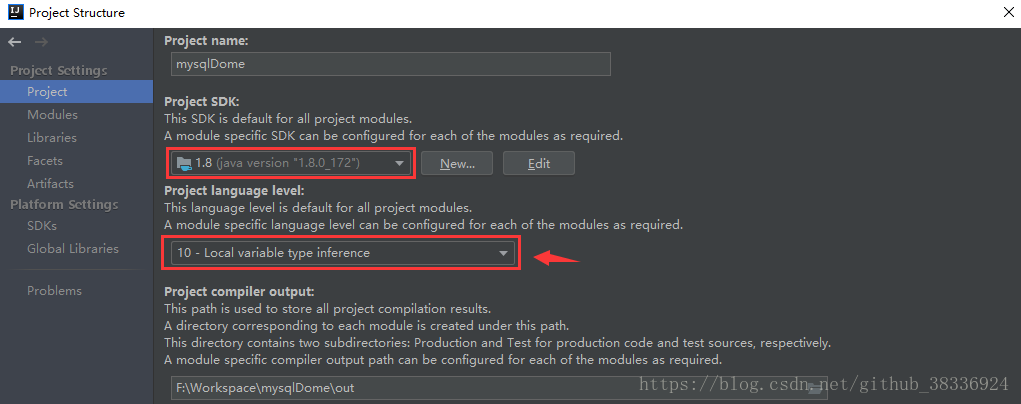
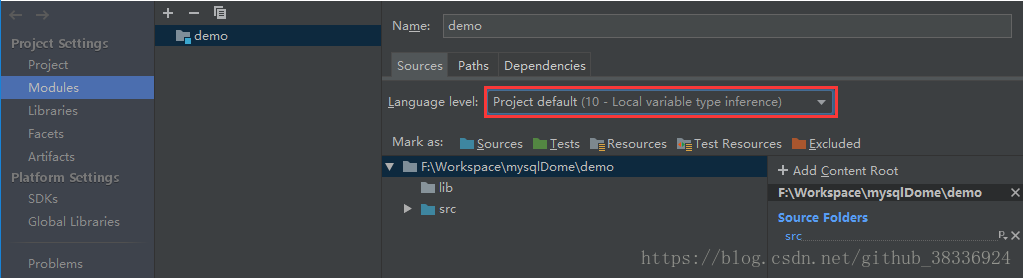
主要原因

本机的[JDK](https://www.baidu.com/s?wd=JDK&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)版本与项目的语言级别不同造成的，例如：

我本机的JDK版本

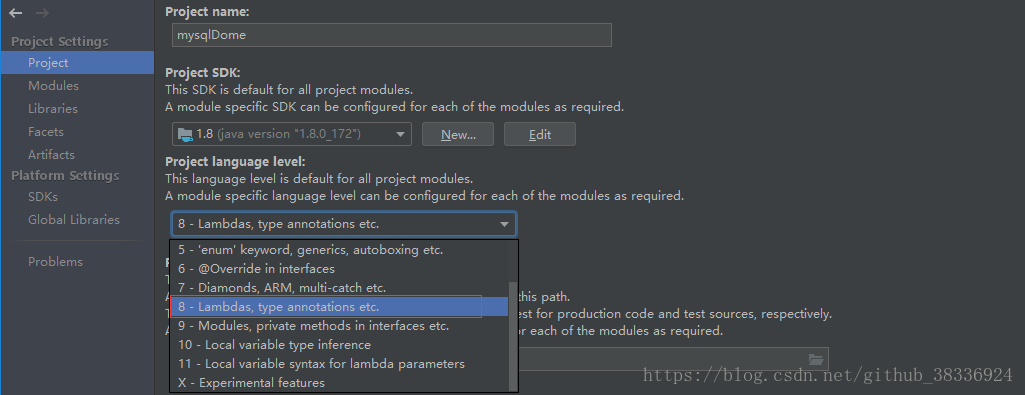


使用Ctrl+Alt+Shift+s打开Project Structure查看Project Language Level，发现是10

解决办法

打开Project Structure界面，选择Project，改变Project Language Level如下图所示



## 踩坑：Failed to create a Maven project: ‘xxx/pom.xml’ already exists in VFS

标签（空格分隔）： 踩坑

最近看hadoop，maven下载依赖包各种问题，无奈之下我只有频繁重新创建项目，不免有重名。

以上为背景。

根据错误提示，我琢磨着，应该把这个VFS里面的内容清理一下，因为毕竟我已经在文件中将project删除了。但是我不知道VFS是啥

遂上stackoverflow找方法，遇到一哥们，问题如下：

When I would like to re-create a maven project with a name had been but have been deleted already, I receive the following error message:

Failed to create maven project: ‘…/pom.xml’ already exists in VFS

下面靠谱回答：

See Cleaning System Cache page.

Open any project in IntelliJ IDEA

Select “File > Invalidate Caches / Restart …”

Click on “Invalidate and Restart”

**WARNING:**

Cleaning out the system caches, keep in mind that:

It results in clearing the local history.

To avoid losing data, check in the changes to your version control system before invalidating caches. Causes a complete rebuild of all the projects ever run in the current version of IntelliJ IDEA.

大体的意思是让我们清除缓存。还提醒我们在清缓存之前注意版本控制问题。

删掉这个[残缺不全](https://www.baidu.com/s?wd=%E6%AE%8B%E7%BC%BA%E4%B8%8D%E5%85%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)的项目，然后 Invalidate and Restart 重新建立。OK

异常信息

Library Maven: com.fasterxml.jackson.core:jackson-databind:unknown has broken classes path:   C:\Users\yaosiyuan\Documents\.m2\repository\com\fasterxml\jackson\core\jackson-databind\unknown\jackson-databind-unknown.jar [Fix]

Library Maven: commons-fileupload:commons-fileupload:unknown has broken classes path:   C:\Users\yaosiyuan\Documents\.m2\repository\commons-fileupload\commons-fileupload\unknown\commons-fileupload-unknown.jar [Fix]

Library Maven: javax.servlet:jsp-api:unknown has broken classes path:   C:\Users\yaosiyuan\Documents\.m2\repository\javax\servlet\jsp-api\unknown\jsp-api-unknown.jar [Fix]

Library Maven: javax.servlet:servlet-api:unknown has broken classes path:   C:\Users\yaosiyuan\Documents\.m2\repository\javax\servlet\servlet-api\unknown\servlet-api-unknown.jar [Fix]

Library Maven: jstl:jstl:unknown has broken classes path:   C:\Users\yaosiyuan\Documents\.m2\repository\jstl\jstl\unknown\jstl-unknown.jar [Fix]

Library Maven: org.springframework:spring-webmvc:unknown has broken classes path:   C:\Users\yaosiyuan\Documents\.m2\repository\org\springframework\spring-webmvc\unknown\spring-webmvc-unknown.jar [Fix]

场景描述

idea 创建maven工程，

异常原因

1、idea没有及时刷新工程，导致找到依赖。

2、代码问题。

解决方案

1、把红线依赖剪切，再粘贴进去，刷新maven工程。

2、检查代码。



在<dependencies>外添加<dependencyManagement>

最终目录

<dependencyManagement>

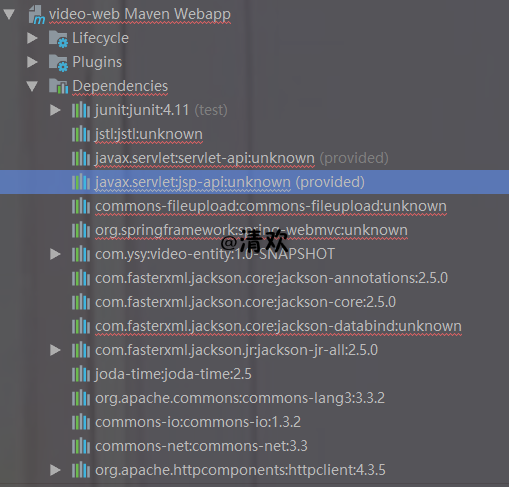
<dependencies>

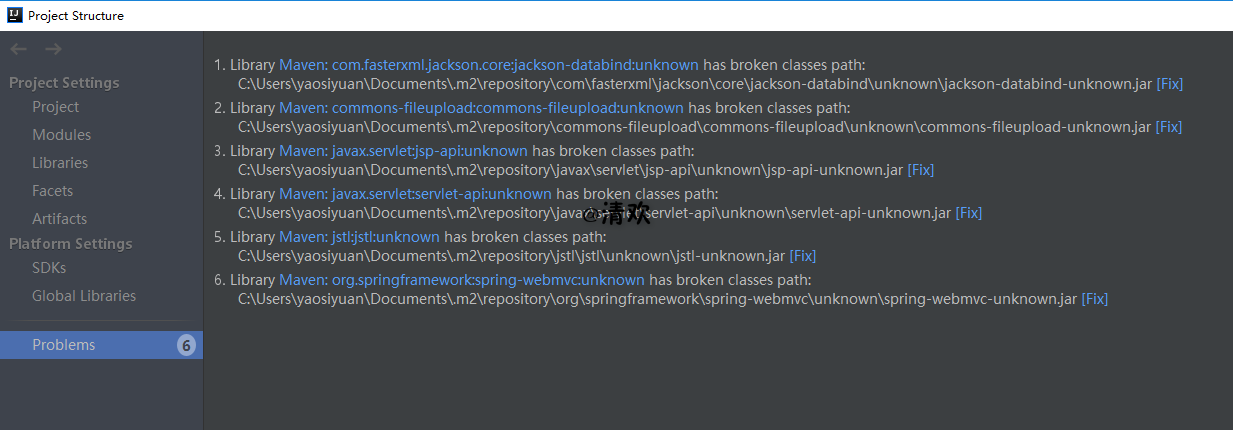
….

</dependencies>

</dependencyManagement>

参考资料

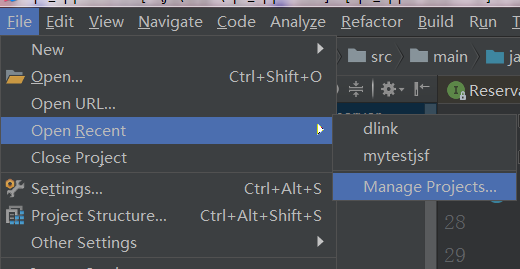




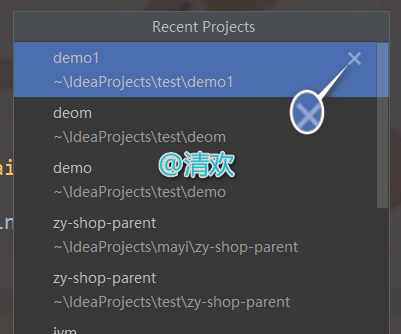
## 彻底删除项目

1. File -> Close Poject 关掉你需要删除的项目

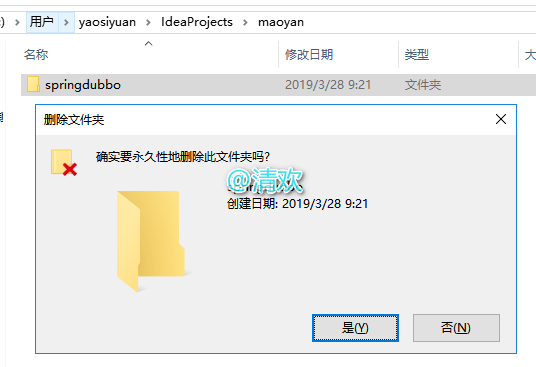
2. File -> Open Recent -> Manage Projects 如下图



3. 出现弹出框 鼠标移到你要删除的项目上面，点击右上角的叉号

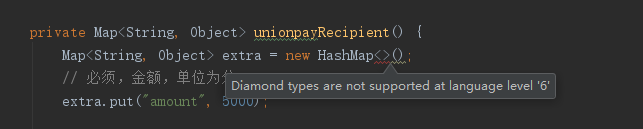


4. 找到你项目的存放位置（如果没有做其他设置就在workspace），然后删除文件即可

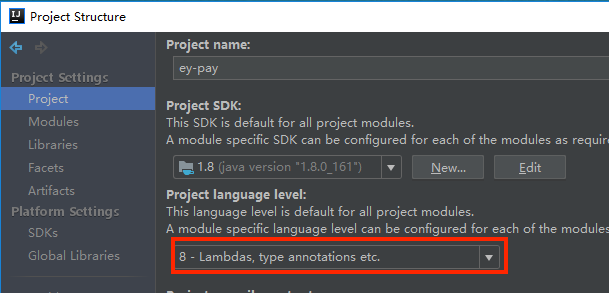


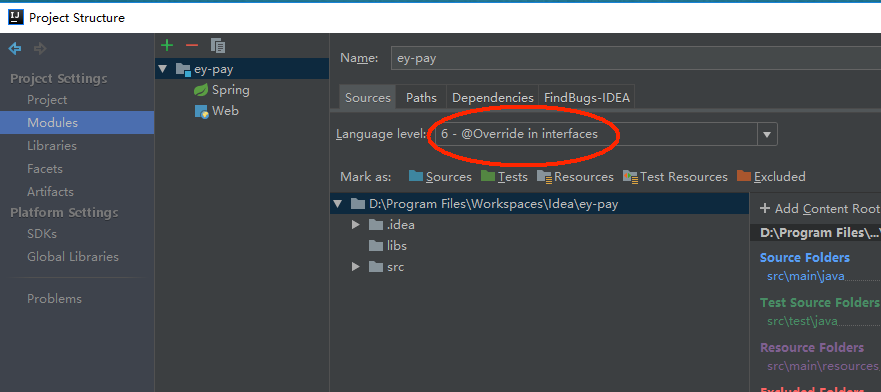
## 123

project的java level 已经核实确实为8，但是IDEA里面仍然会有如下图的提示:



通过查看项目设置，发现project的java level 也是8。





改为8就好了。

遇到类似问题重点排查的位置：

project structure中的project，module中的sources和setting中的java compiler

## 解决IntelliJ IDEA 创建Maven项目速度慢问题 DarchetypeCatalog

**原因**

IDEA根据maven archetype的本质，其实是执行mvn archetype:generate命令，该命令执行时，需要指定一个archetype-catalog.xml文件。

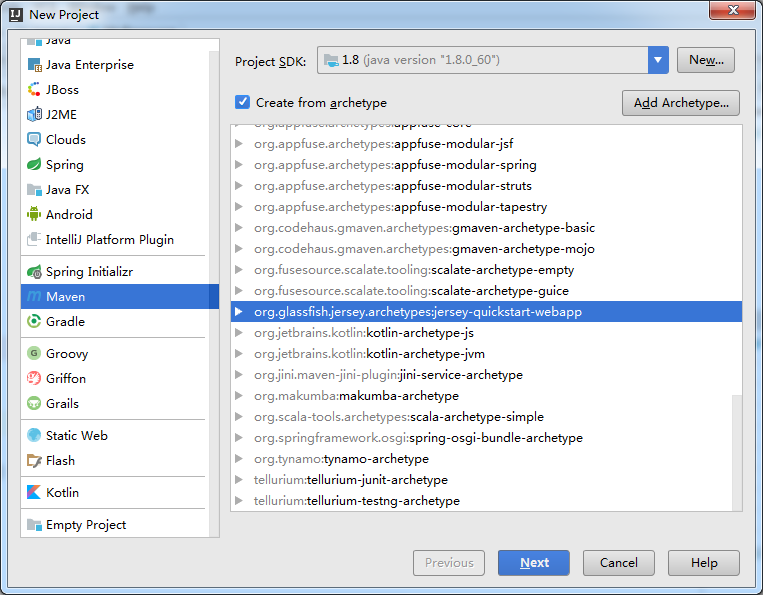
该命令的参数-DarchetypeCatalog，可选值为：remote，internal  ，local等，用来指定archetype-catalog.xml文件从哪里获取。

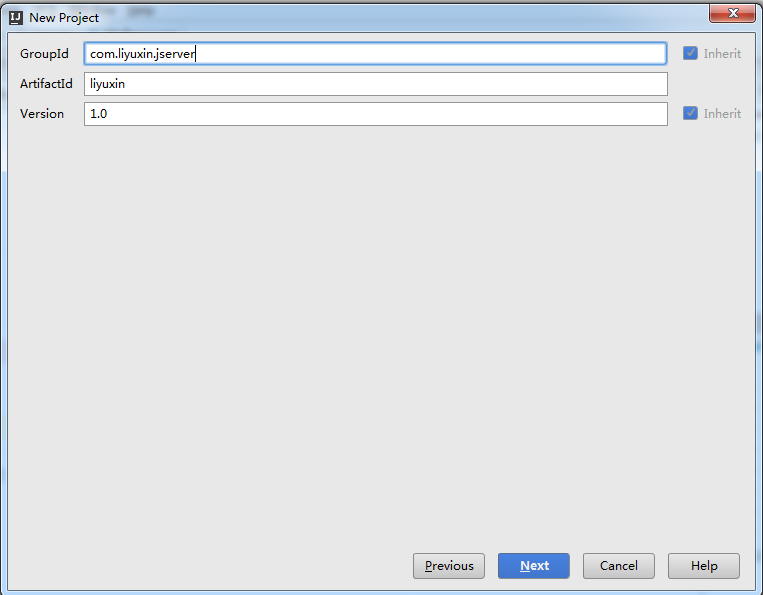
默认为remote，即从 http://repo1.maven.org/maven2/archetype-catalog.xml路径下载archetype-catalog.xml文件。

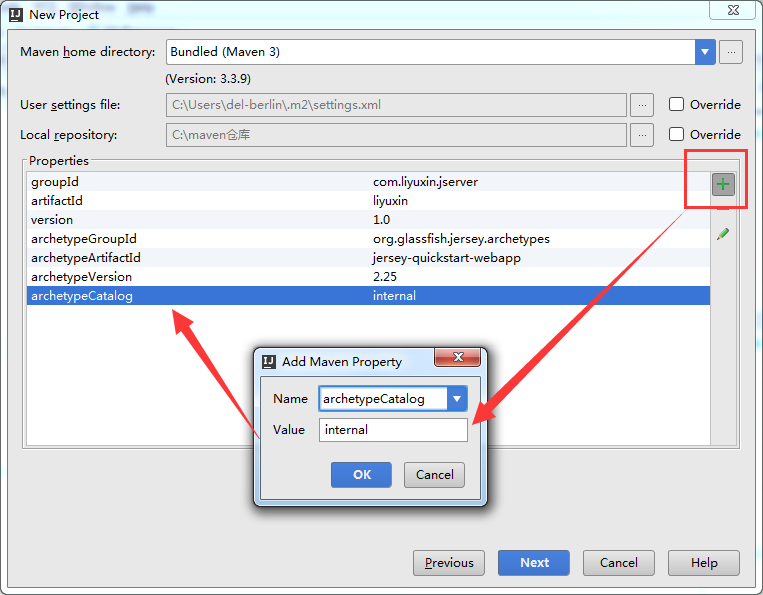
http://repo1.maven.org/maven2/archetype-catalog.xml 文件约为3-4M，下载速度很慢，导致创建过程卡住。

**解决方法**

**1.在下面界面添加一个属性,archetypeCatalog = internal**

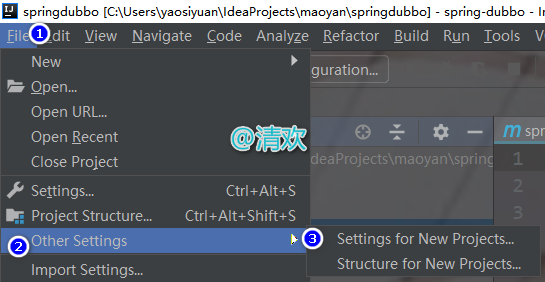
****



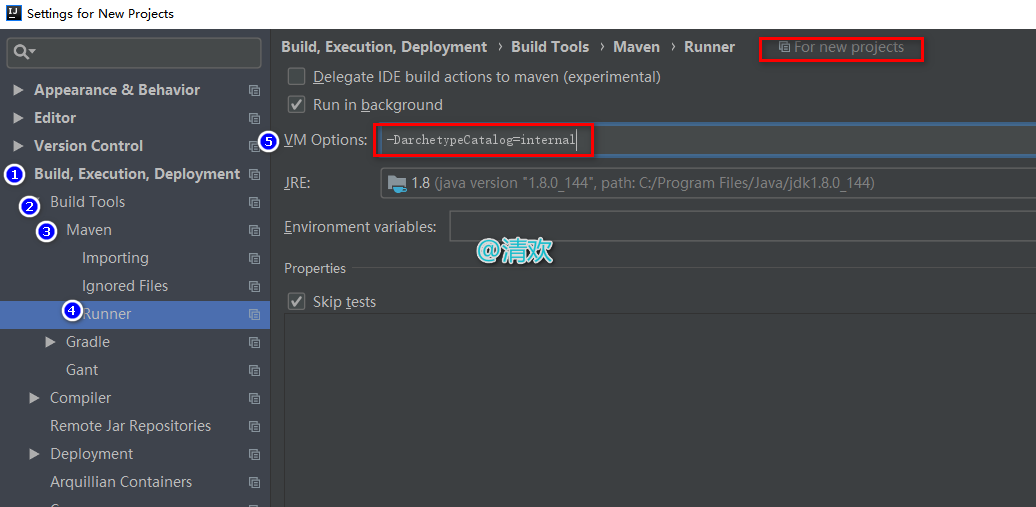


### **方法二**

**在maven的VM Options加上-DarchetypeCatalog=internal参数，如下：**



注意是defaultsettings 而不是settings。这样的话是全局设置，每次新建项目都会使用的设置。





**方法3: 用local （推荐这种方法 完美）**

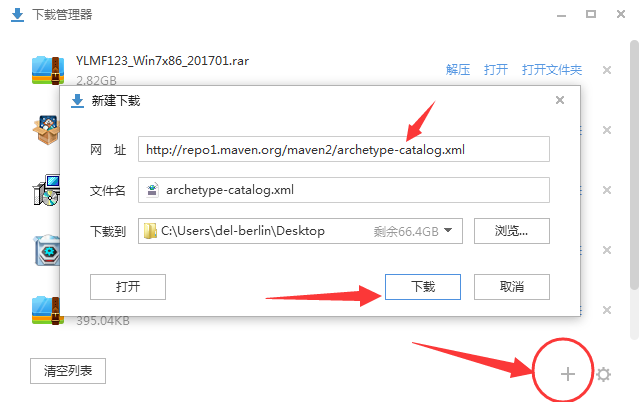
1.由于默认情况下，根据archetype创建maven项目会从网络下载catalog文件，导致创建maven项目缓慢

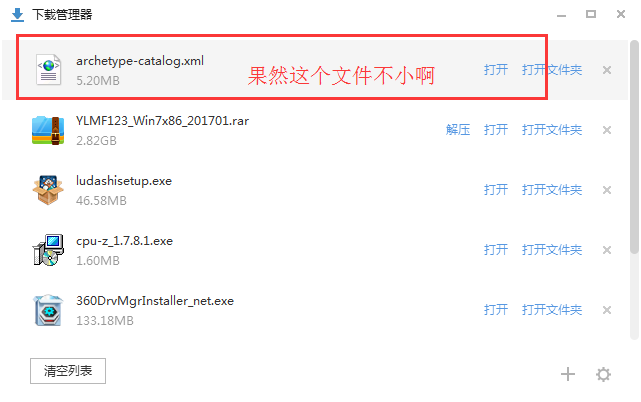
Searching for remote catalog: http://repo1.maven.org/maven2/archetype-catalog.xml

2.解决办法可以设置使用本地catalog文件，在IDEA中设置archetype的使用方式为local;

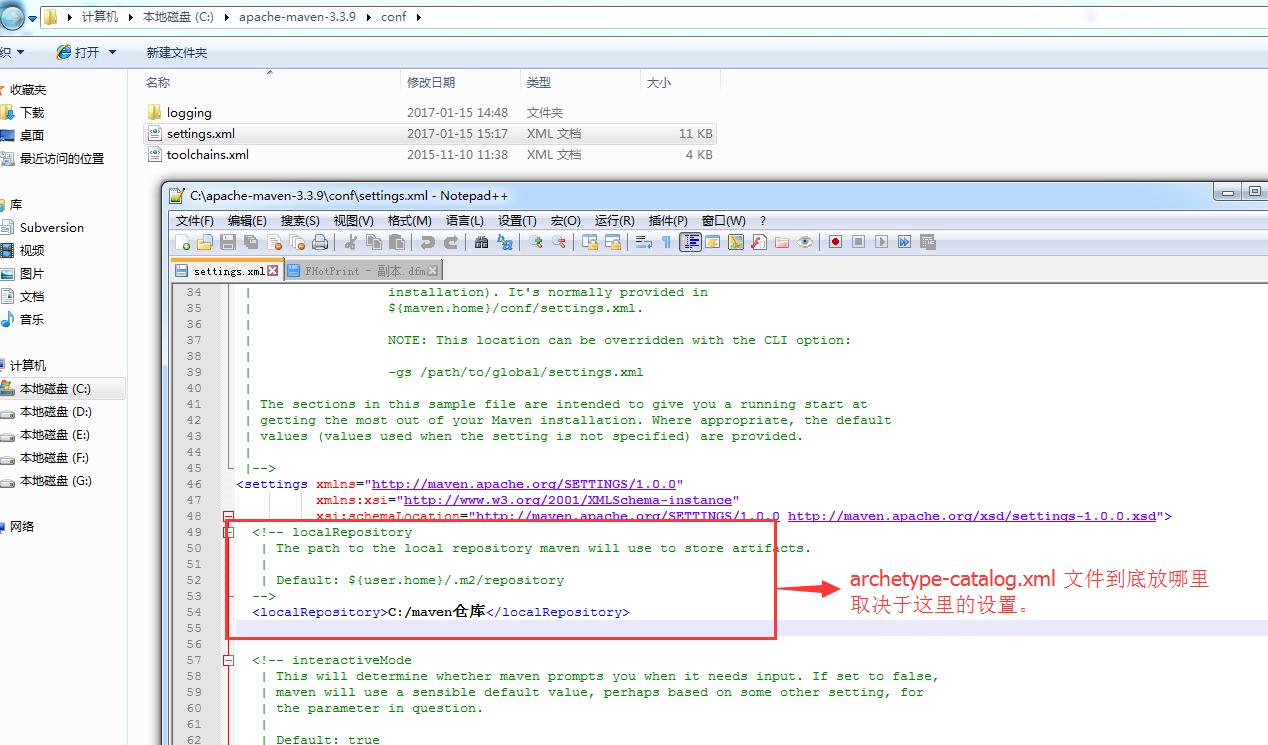
-DarchetypeCatalog=local

直接先把 文件用搜狗浏览器或其它下载工具下载下来。





**下载后放到哪里呢，这里与本地仓库的位置有关，假如本地仓库是maven默认的，并没有修改 那么就需要放到 C:\Users\del-berlin\.m2\repository\org\apache\maven\archetype\archetype-catalog\2.4\下  ，本地仓库的默认位置：Default: ${user.home}/.m2/repository**

****

****

