IDEA与GIT的使用

[1. Git配置 2](#_Toc25466)

[1.1. 安装GitBash 2](#_Toc12752)

[1.2. Git初始化 2](#_Toc28445)

[2. Git命令 3](#_Toc16250)

[2.1. Git 起步 3](#_Toc29190)

[2.1.1. 三种状态 3](#_Toc31272)

[2.1.2. Git 工作流程 3](#_Toc8797)

[2.2. Git基础 4](#_Toc18464)

[2.2.1. Git 基础 - 获取 Git 仓库 4](#_Toc6947)

[2.2.2. Git 基础 - 修改更新到仓库 4](#_Toc14054)

[2.2.3. Git 基础 - 查看提交历史 6](#_Toc16431)

[2.2.4. Git 基础 - 撤消操作 6](#_Toc32205)

[2.3. Git分支 7](#_Toc10995)

[2.3.1. Git 分支 - 分支的查看、新建与合并 7](#_Toc23083)

[2.4. submodule 命令 8](#_Toc11898)

[3. 安装IntelliJ IDEA 8](#_Toc14811)

[3.1. 运行IntelliJ IDEA安装程序 8](#_Toc13372)

[3.2. 同意协议 9](#_Toc3901)

[3.3. 选择安装路径 10](#_Toc192)

[3.4. 把桌面快捷方式和java关联选项都勾选起来 10](#_Toc16855)

[3.5. 默认 11](#_Toc9763)

[3.6. 开始安装 11](#_Toc2359)

[3.7. 完成安装并运行IntelliJ IDEA 12](#_Toc23550)

[3.8. 不需要导入先前版本 13](#_Toc11222)

[3.9. IntelliJIDEA Activation界面 13](#_Toc28472)

[3.10. 注册 14](#_Toc10751)

[3.11. 接受License条款 15](#_Toc20947)

[3.12. 设置界面风格，选择自己喜欢的风格 16](#_Toc23820)

[3.13. 按默认，下一步 17](#_Toc20915)

[3.14. 按默认，开始使用Intellij IDEA 18](#_Toc5497)

[4. IDEA操作 19](#_Toc30437)

[4.1. 导入Git项目 19](#_Toc22170)

[4.2. 修改Maven配置 20](#_Toc19709)

[4.3. 配置Maven命令 20](#_Toc1274)

[4.4. 配置Artifact 22](#_Toc14465)

[4.5. 配置tomcat 24](#_Toc17576)

# Git配置

## 安装GitBash

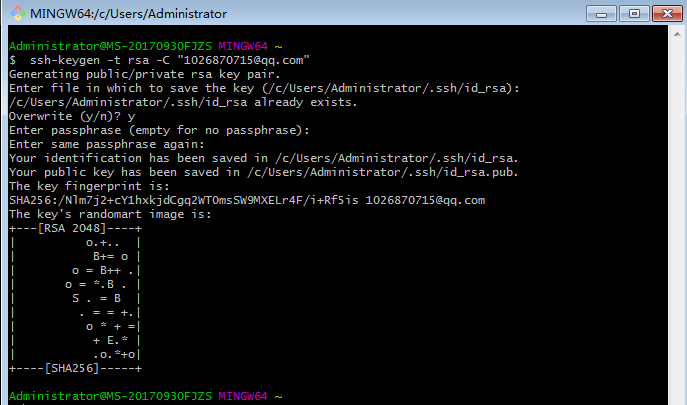
(已下载Git-2.13.3-64-bit.exe)下载地址<https://git-for-windows.github.io/>

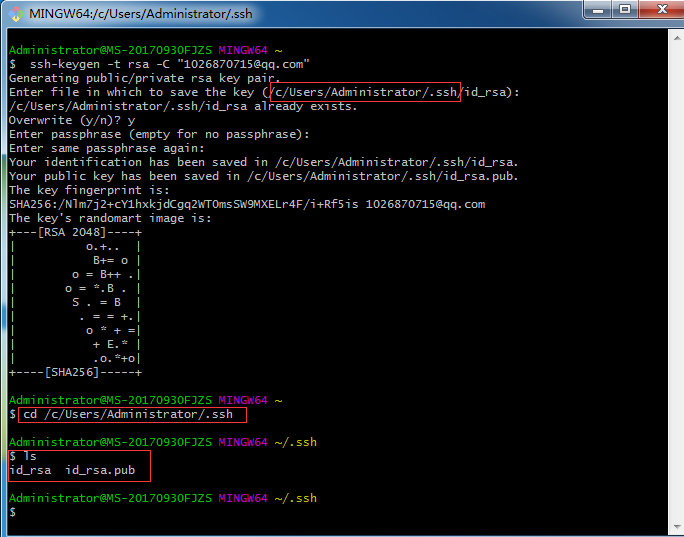
GitBash安装成功，右击鼠标会出现git操作，选择git bash here可以打开git命令行（一个类linux操作界面）

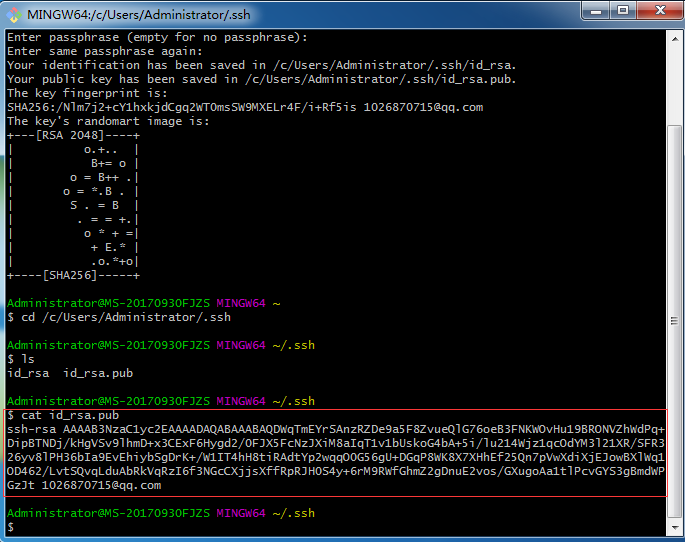


## Git初始化

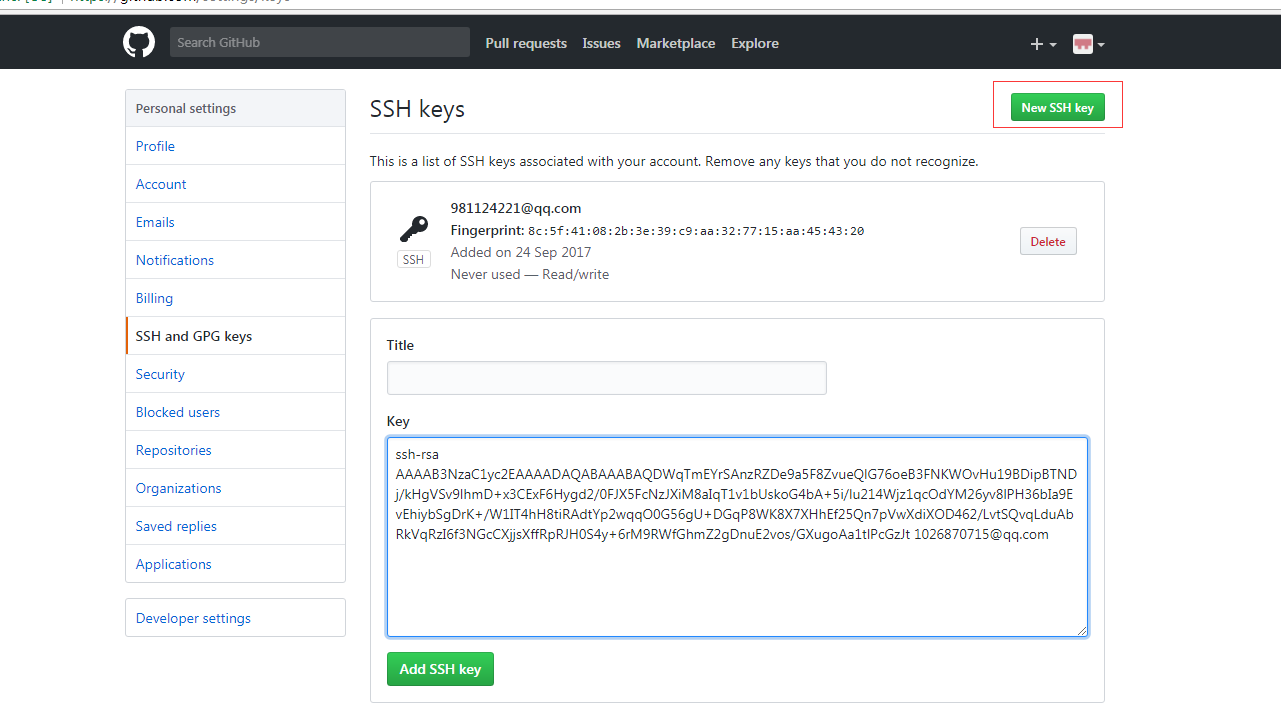
右击打开Git Bash Here命令窗，输入ssh-keygen -t rsa -C "你的邮箱账号"，一路回车。







生成的公钥拷贝到github的账号settings中，点击NEW SSH key按钮



# Git命令

## Git 起步

### 三种状态

Git 项目的三个工作区域的概念：Git 仓库、工作目录以及暂存区域。



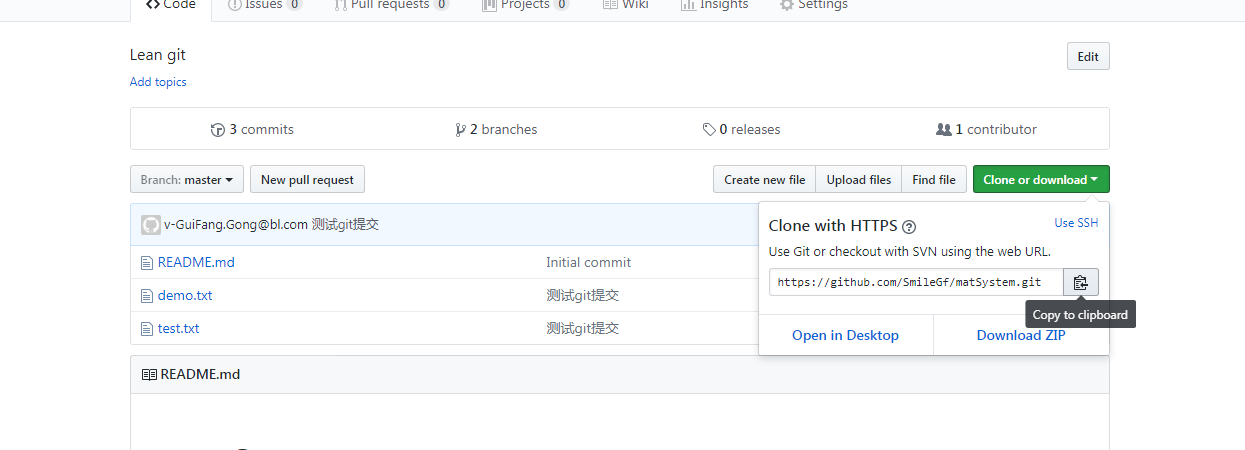
### Git 工作流程

1. 在工作目录中修改文件。
2. 暂存文件，将文件的快照放入暂存区域。
3. 提交更新，找到暂存区域的文件，将快照永久性存储到 Git 仓库目录。

如果 Git 目录中保存着的特定版本文件，就属于已提交状态。 如果作了修改并已放入暂存区域，就属于已暂存状态。 如果自上次取出后，作了修改但还没有放到暂存区域，就是已修改状态。

## Git基础

### Git 基础 - 获取 Git 仓库

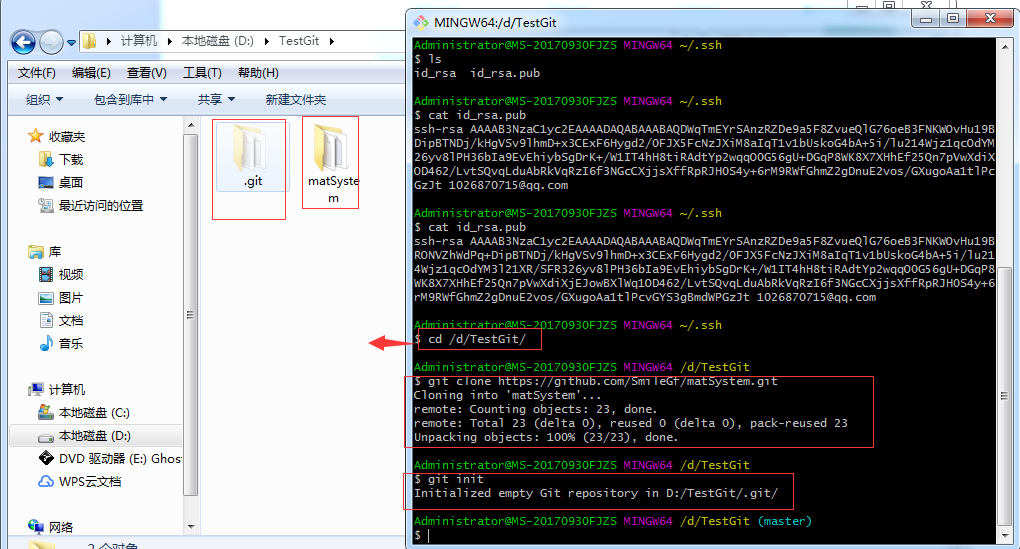


克隆git项目git clone [url]，比如：

$ git clone https://github.com/SmileGf/matSystem.git

克隆后本地TestGit是没有.git的子目录的，你需要执行以下命令初始化仓库：

$ git init



然后更新下来的iokblweb是没有子模块内容的，执行以下命令更新子模块内容：(没有子模块的项目不考虑)

$ git submodule update --recursive

### Git 基础 - 修改更新到仓库

检查当前文件状态

$ git status

暂存已修改文件

$ git add [. or 文件名]

在.gitignore中列出要忽略的文件

文件 .gitignore 的格式规范如下：

所有空行或者以 ＃ 开头的行都会被 Git 忽略。

可以使用标准的 glob 模式匹配。

匹配模式可以以（/）开头防止递归。

匹配模式可以以（/）结尾指定目录。

要忽略指定模式以外的文件或目录，可以在模式前加上惊叹号（!）取反。

$ cat .gitignore

提交更新（只提交暂存区的文件也就是git add过的文件，并且只提交到本地仓库而已）

$ git commit [-m ‘备注信息’]

当你忘记添加 .gitignore文件，不小心把一个很大的日志文件或一堆 .a 这样的编译生成文件添加到暂存区时，这一做法尤其有用。 为达到这一目的，使用 --cached 选项：

$ git rm --cached README

git rm 命令后面可以列出文件或者目录的名字，也可以使用 glob 模式。 比方说：

$ git rm log/\\*.log

注意到星号 \* 之前的反斜杠 \， 因为 Git 有它自己的文件模式扩展匹配方式，所以我们不用 shell 来帮忙展开。 此命令删除 log/ 目录下扩展名为 .log 的所有文件。 类似的比如：

$ git rm \\*~

该命令为删除以 ~ 结尾的所有文件。

推送到远程仓库

$ git push origin master

### Git 基础 - 查看提交历史

查看提交历史

$ git log

### Git 基础 - 撤消操作

撤销Git add操作

$ git reset HEAD <file> # 取消add操作并保留修改

撤销Git commit操作

$ git reset --soft <commit\_id> #可以回退到某个commit并保存之前的修改

$ git reset --hard <commit\_id> #回退到某个commit不保留之前的修改

撤销Git push操作

$ git revert <commit\_id>

## Git分支

### Git 分支 - 分支的查看、新建与合并

 查看远程分支

$ git branch -a

查看本地分支

$ git branch

创建分支

$ git branch test

把分支推到远程分支

$ git push origin test

切换分支到test

$ git checkout test

删除本地分支

$ git branch -d test

删除远程分支

$ git branch -r -d origin/test

$ git push origin :test

分支的合并

这个命令把分支"branchname"合并到了当前分支里面。

$ git merge branchname

## submodule 命令

submodule 添加

$ git submodule add [-b master ] [URL to Git repo] [local path]

Submodule内容更新

在父项目的目录下运行

$ git submodule foreach git checkout master

$ git submodule foreach git pull

在Submodule的目录下面更新

$ git pull

父工程指定的submodule版本更新

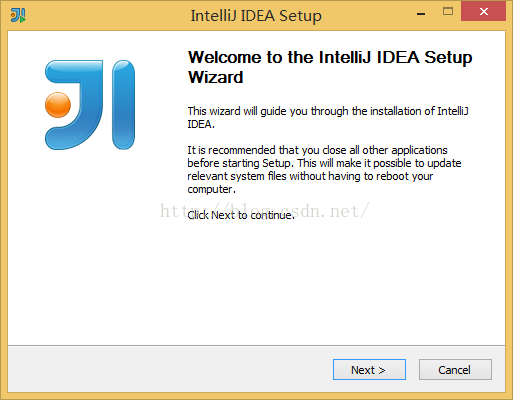
$ git submodule update --recursive

父工程更新submodule版本

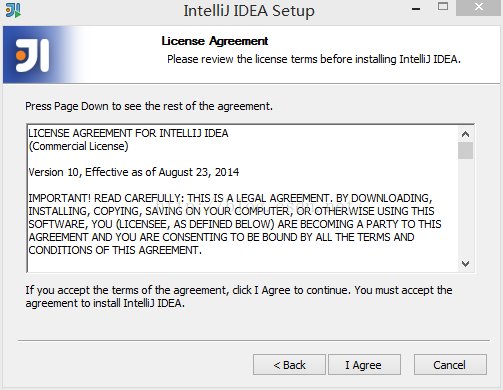
$ git submodule update --remote

# 安装IntelliJ IDEA

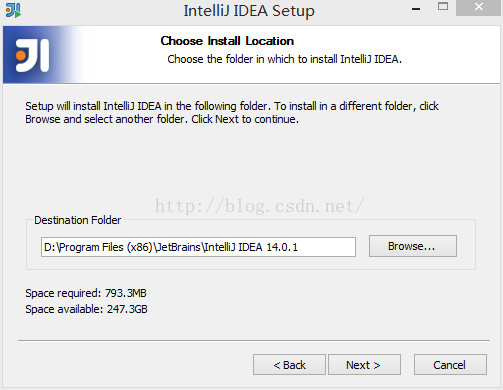
## 运行IntelliJ IDEA安装程序



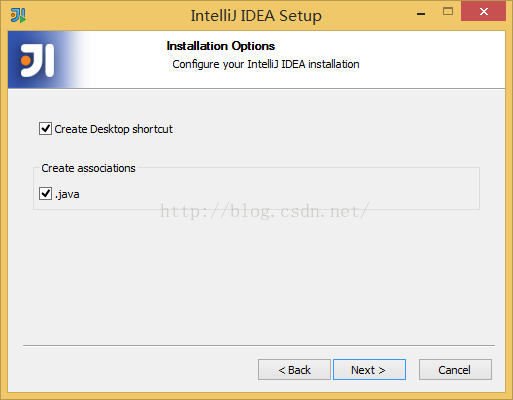
## 同意协议



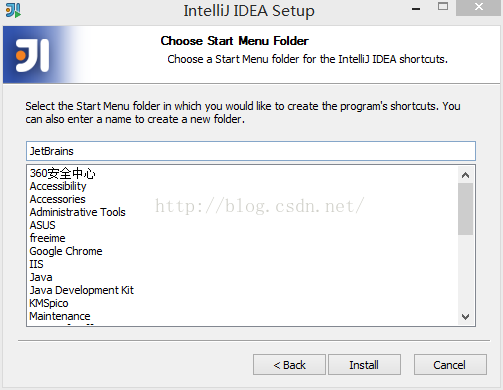
## 选择安装路径



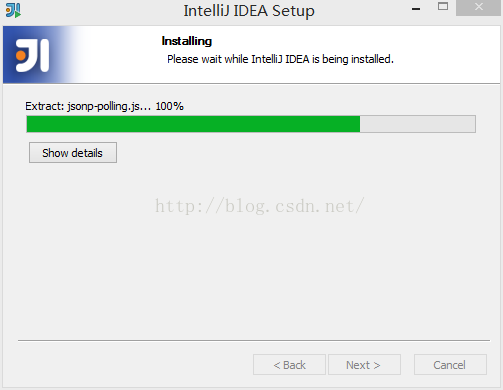
## 把桌面快捷方式和java关联选项都勾选起来



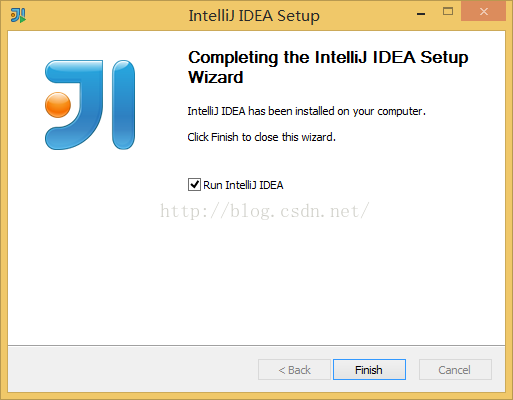
## 默认



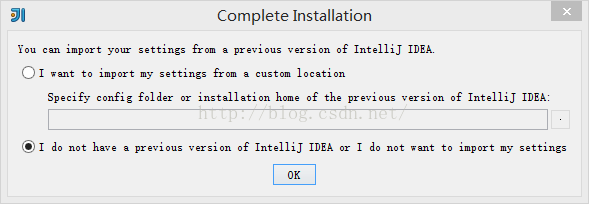
## 开始安装



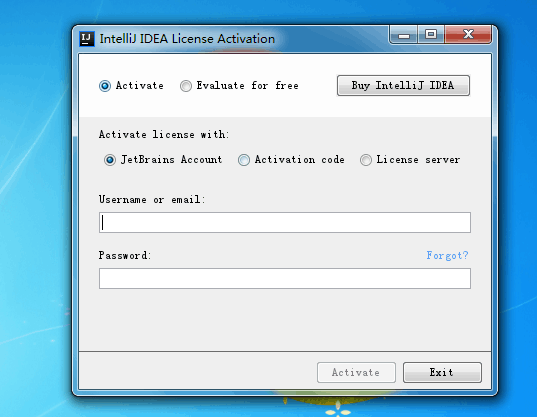
## 完成安装并运行IntelliJ IDEA



## 不需要导入先前版本

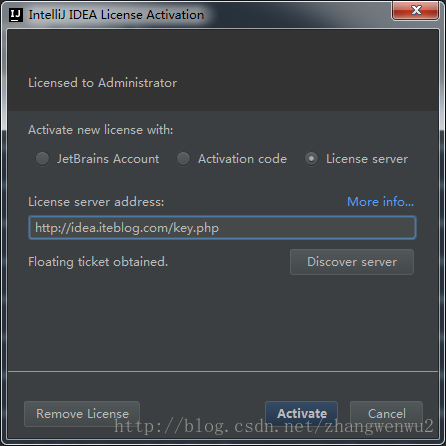


## IntelliJIDEA Activation界面



## 注册

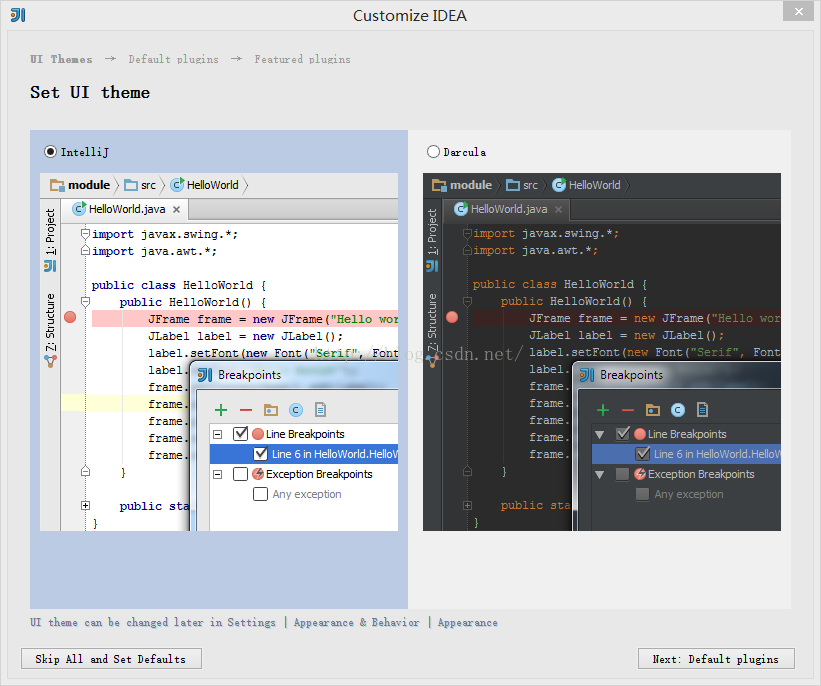
点击License server然后输入 http://idea.iteblog.com/key.php（注意：php要小写）即可~，如下图，



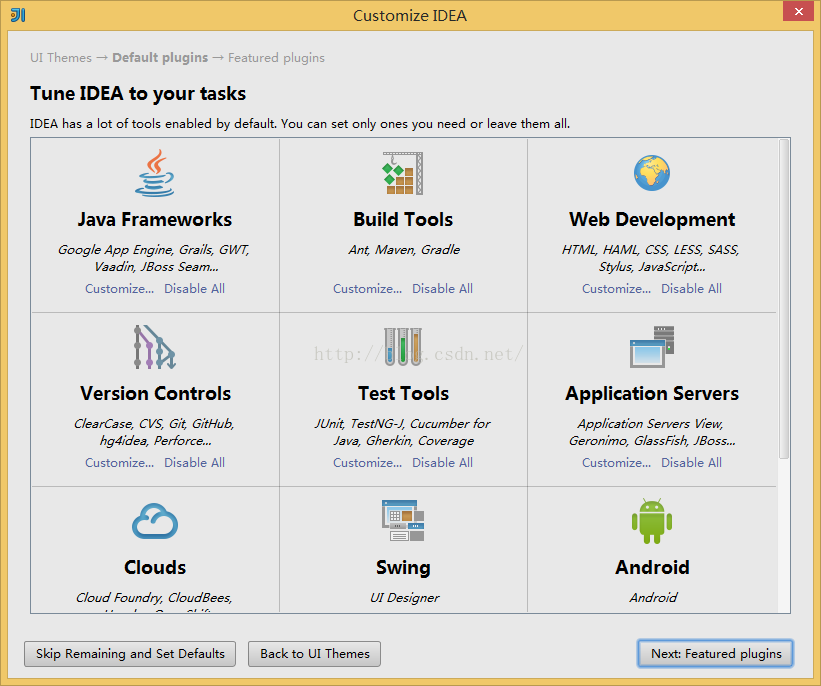
## 接受License条款



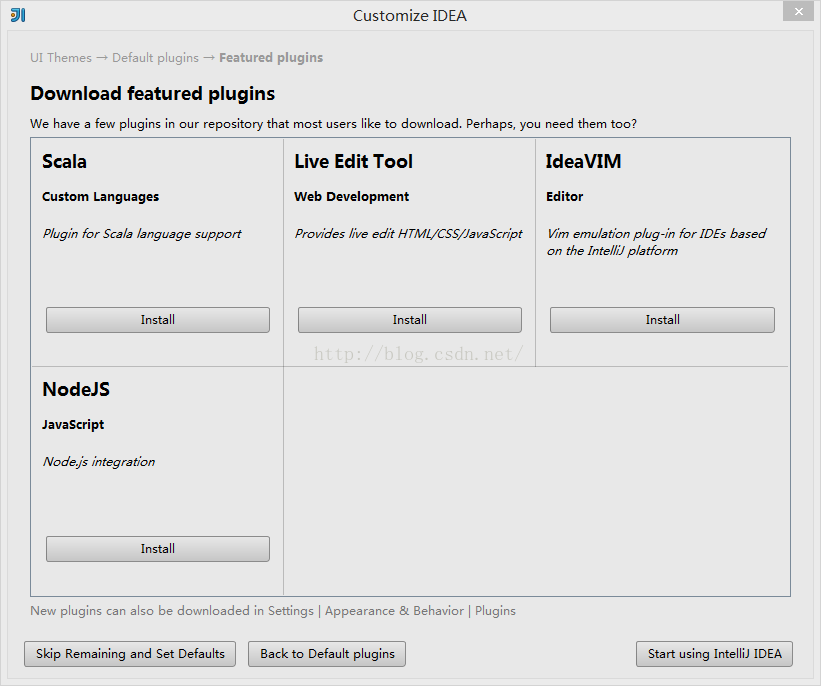
## 设置界面风格，选择自己喜欢的风格



## 按默认，下一步



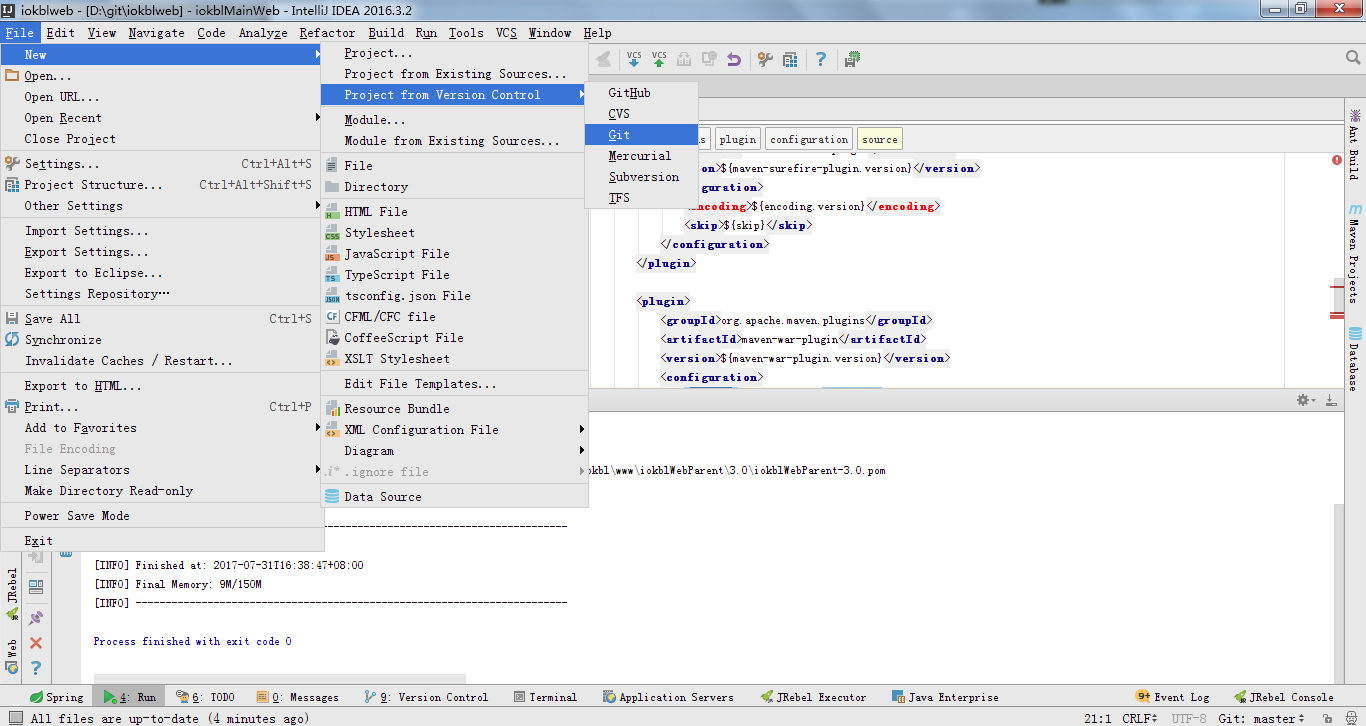
## 按默认，开始使用Intellij IDEA



# IDEA操作

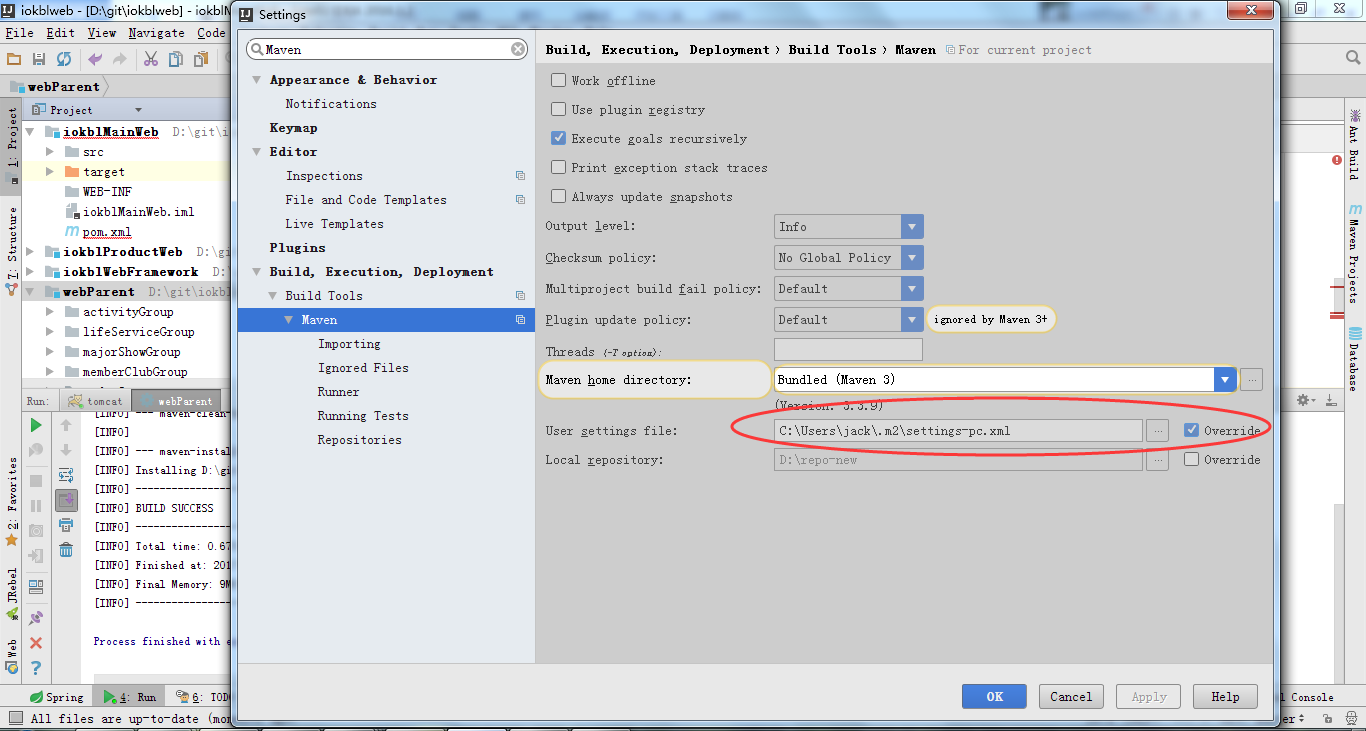
## 导入Git项目

点击File-->New-->Project form Version Control（project来自版本管理） 或者 Project form Existing Sources，如下图所示：



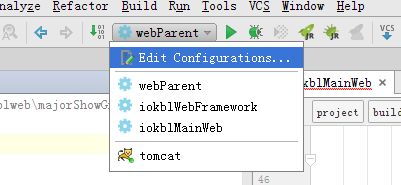
## 修改Maven配置

File-->settings-->Maven，如下图所示：

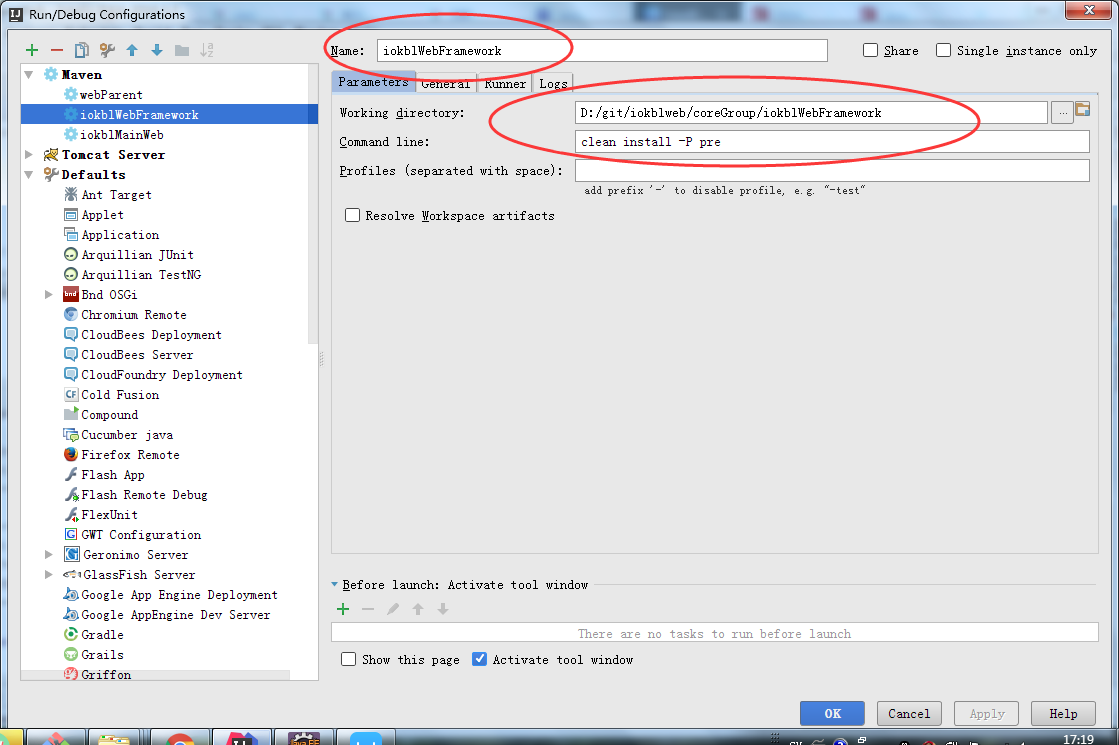


## 配置Maven命令

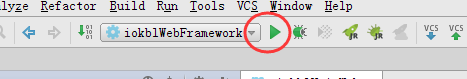
1 点击"Edit Configurations"



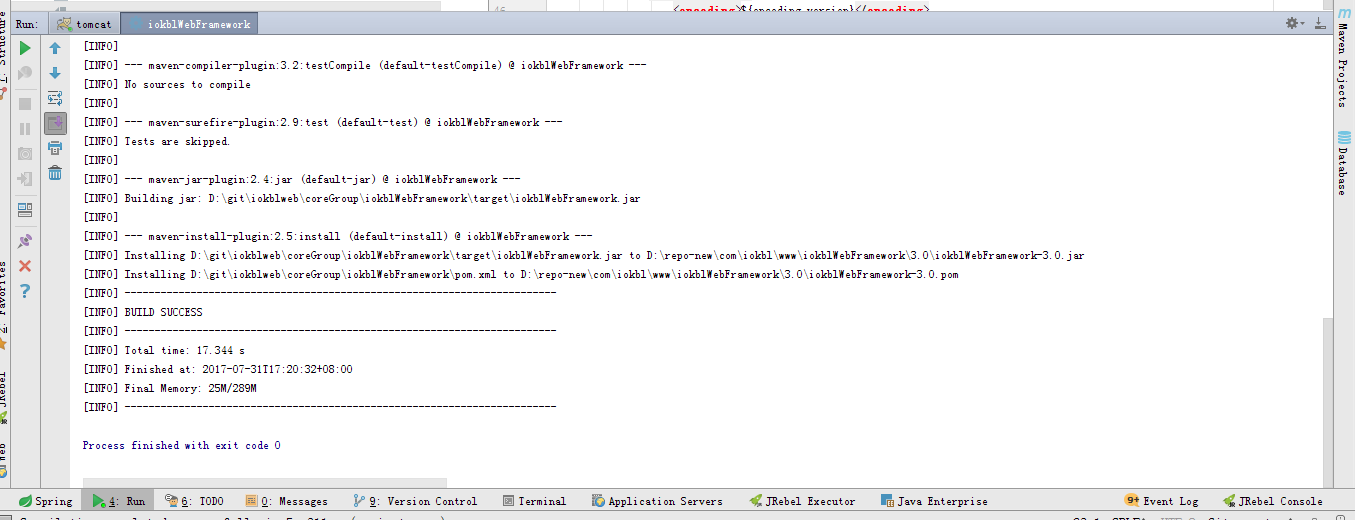
 2 进入Run/Debug Configurations窗口,点击左上角的"+",在弹出的"Add New Configuration"列表中选择"maven".



3 配置完成后，新的configuration会成为默认的configuration,直接点击运行按钮－－三角形绿色按钮

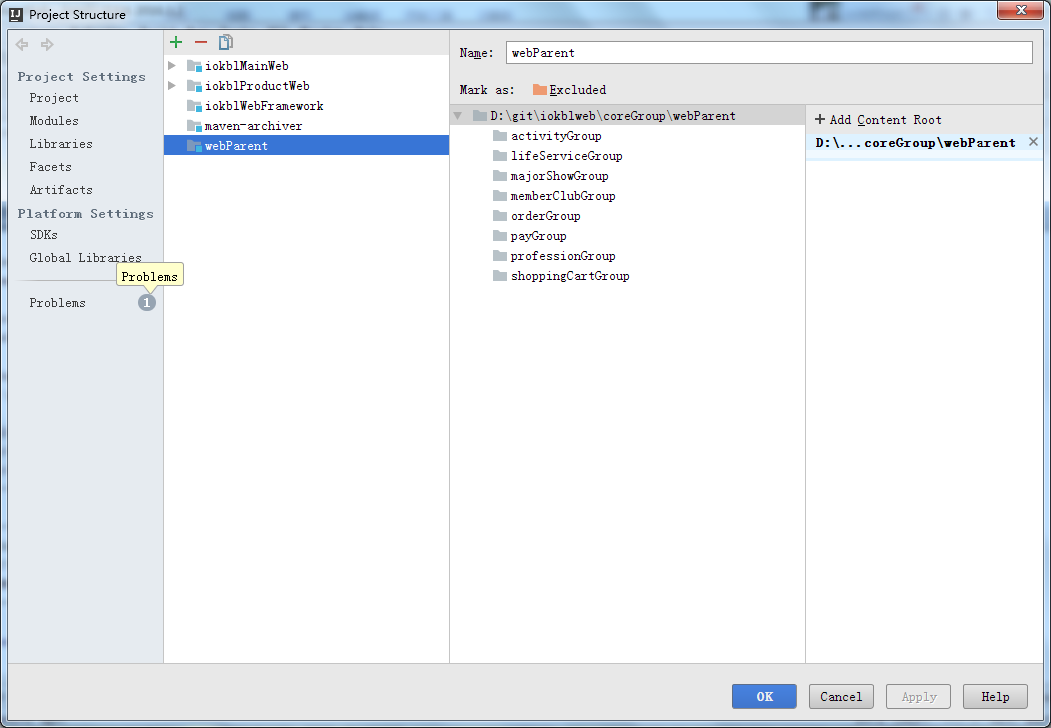


4 之后,就会执行配置的maven命令！

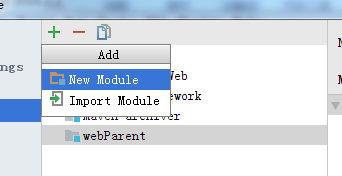


## 配置Artifact

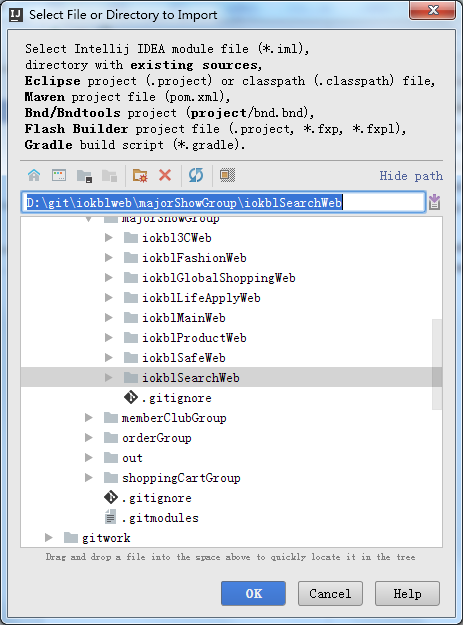
1. 点击File-->Project Structure...



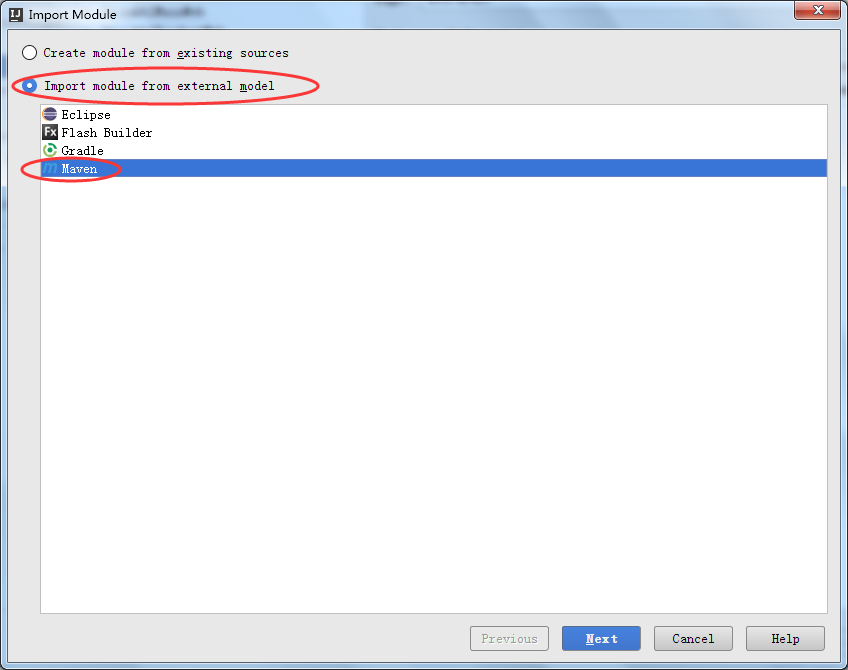
1. 选择Modules 点击“+”，选择Import Module



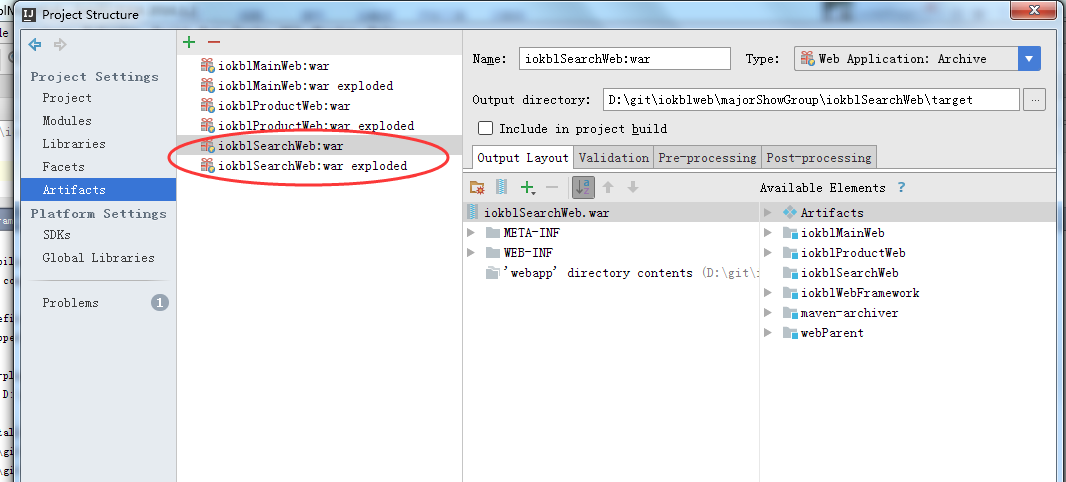
1. 选择添加的目录



1. 如图选择Maven，一直点击下一步

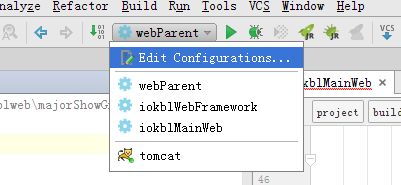


1. 条件Module后，在Artifact中自动添加了\*:war 和 \*:war exploded

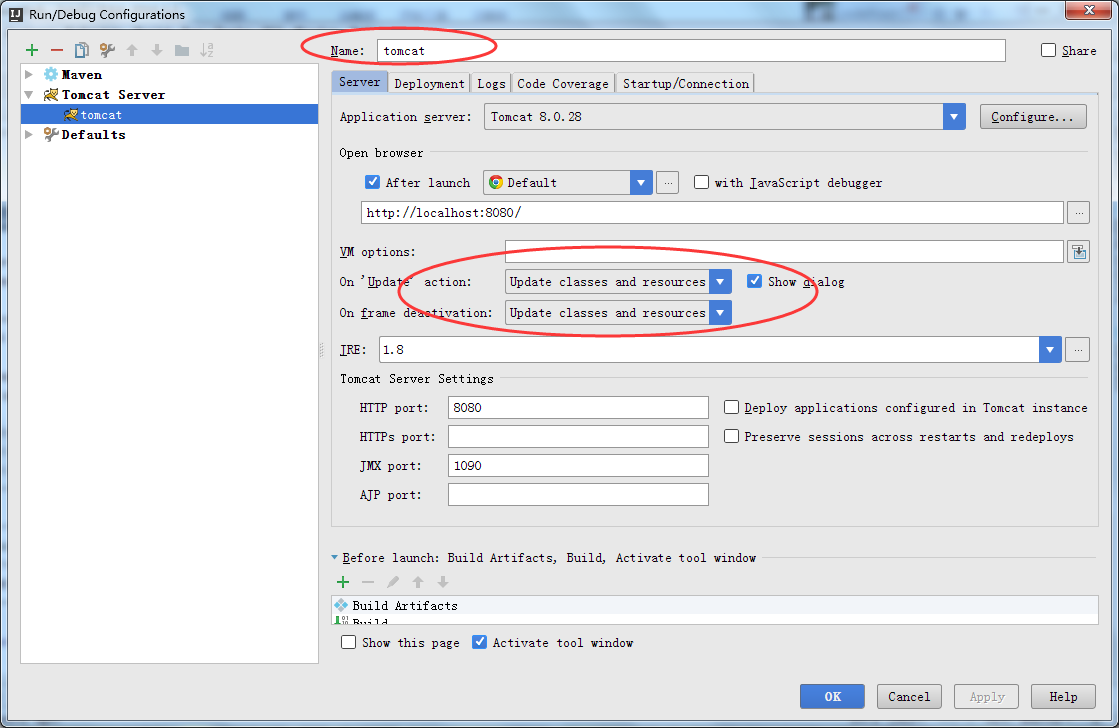


## 配置tomcat

1 点击"Edit Configurations"

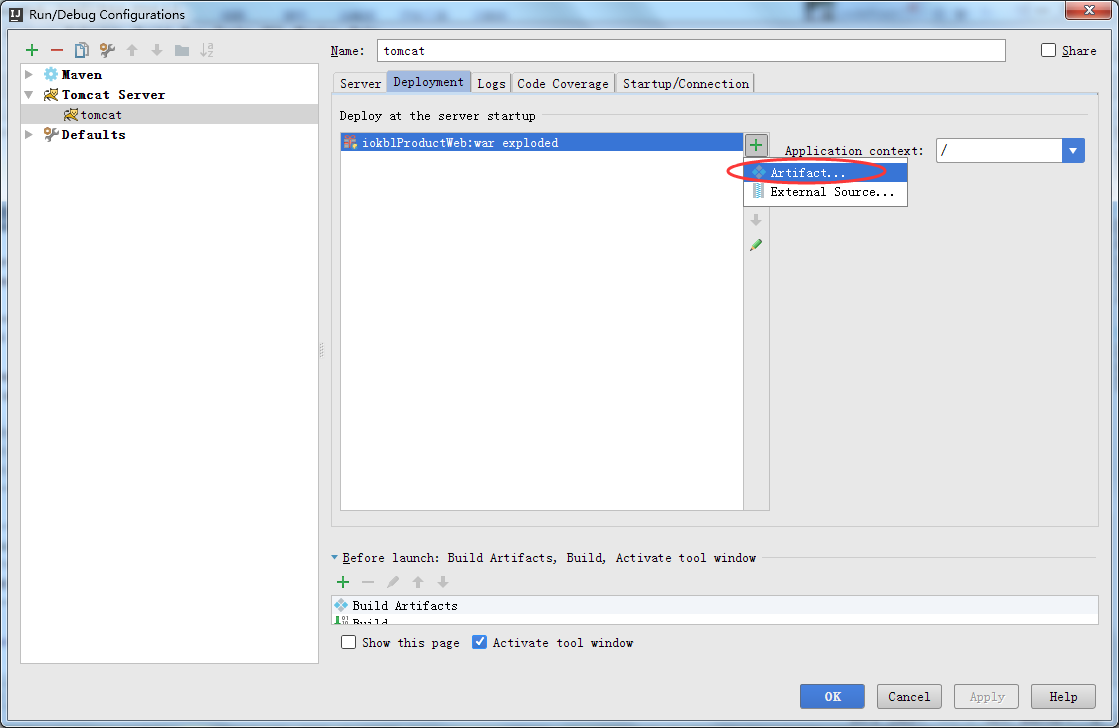


 2 进入Run/Debug Configurations窗口,点击左上角的"+",在弹出的"Add New Configuration"列表中选择"tomcat".

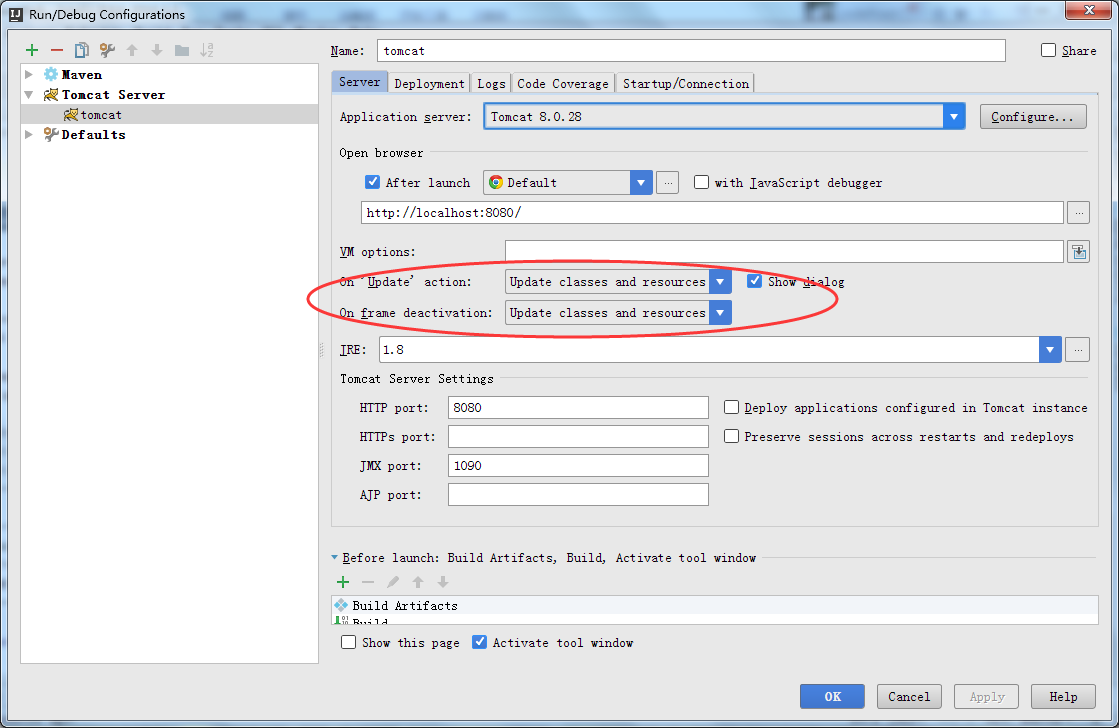


Tomcat 中修改 Server 、Deployment

1. 修改Deployment，点击“+”，选择Artifact



1. 修改Server , On ‘Update’ action和On frame deactivetion,但是必须上一步选择Artifact中的\*:war exploded，才会有Update classes and resouces选项



1. 启动Tomcat

