IDEA 学习笔记

水滴石穿，不断学习，不断耕耘，总会有收获的一天。

笔者：范亚南

**目录**

[“Import from external model” 与 “Create from existing source”的区别 2](#_Toc27861382)

[IntelliJ IDEA创建maven web项目（IDEA新手适用） 2](#_Toc27861383)

[Tomcat的安装与环境变量配置 12](#_Toc27861384)

[Maven的安装与配置 16](#_Toc27861385)

[IDEA中Maven依赖下载失败解决方案 18](#_Toc27861386)

[使用IDEA建立SpringBoot工程 20](#_Toc27861387)

# “Import from external model” 与 “Create from existing source”的区别

IDEA在导入第一个Spring的helloworld的demo时，提示 “Import from external model” 与 “Create from existing source”。不知道这两个选项的区别，最后得到如下结论。

“Import from external model”选项允许你导入一个已经存在的固定model的项目。例如Eclipse、gradle和Maven等。IDEA会根据你选择的model自动转换项目文件，换句话说就是把项目成功从原来的平台迁移到IDEA。

“Create from existing source”选项会在已有的代码基础上创建新的IDEA项目，换句话说就是用IDEA自己的model。如果你只有一部分代码而没有完整的构建model的话，建议用这个选项。

Demo的链接为：git clone <https://github.com/spring-guides/gs-rest-service.git>

# IntelliJ IDEA创建maven web项目（IDEA新手适用）

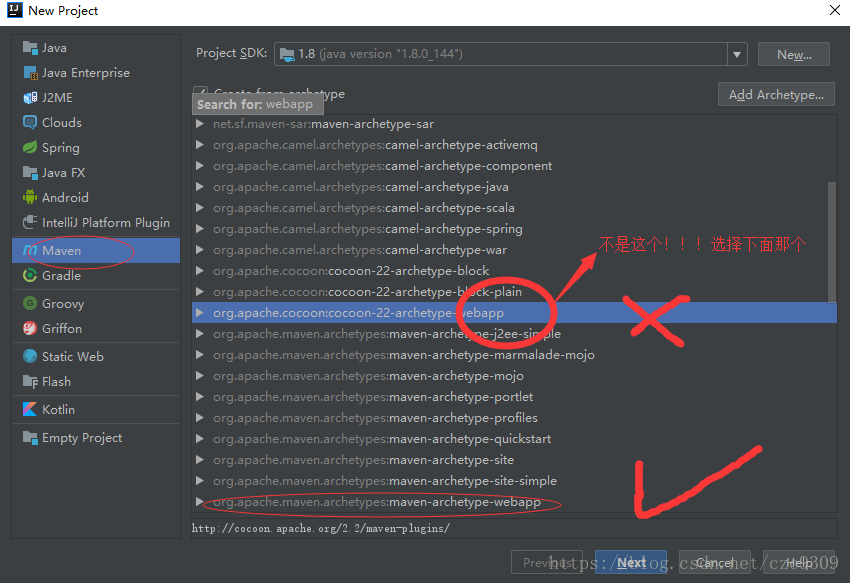
步骤一：首先先创建一个project,在这里就是创建一个maven的工作空间

步骤二：按照下面的步骤操作就可以了，最后next

首先，选择左边的maven

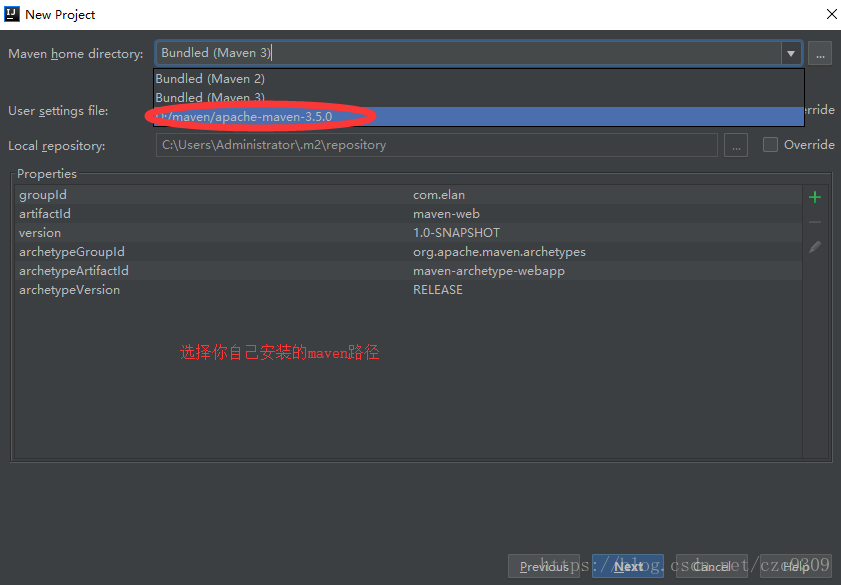
然后在右边Creater from archetype选项前面打个勾，

最后选择maven-archetype-webapp（这里不知道是我IDEA的问题还是什么原因，我打webapp搜索它只给出第一个，所以在这个坑上耗了几乎1个小时，都选择了第一个）

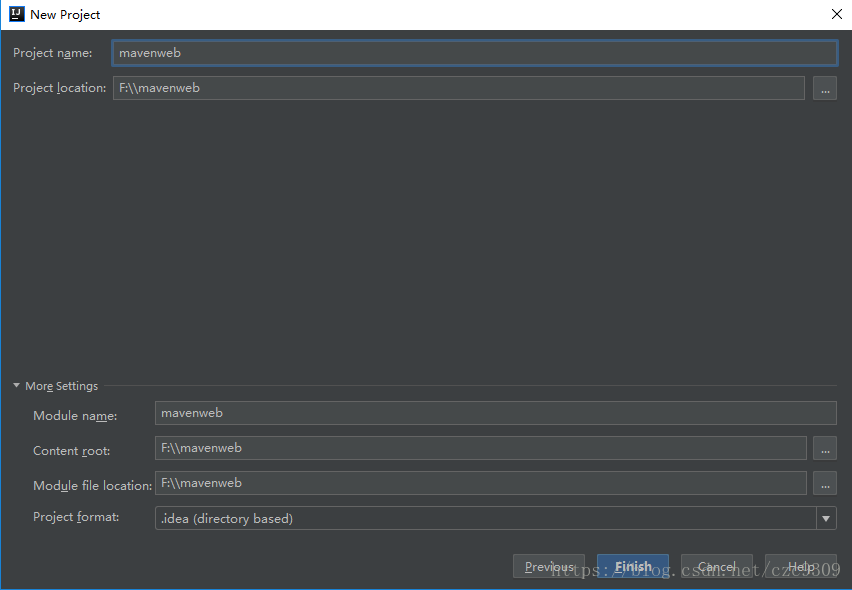


步骤三：就简单了，填写你的GroupId跟ArtifictId（瞎写都没问题，当然不建议哈，这里GroupId一般写公司名，ArtifictId一般我写项目名）

步骤四：配置自己的maven，然后next

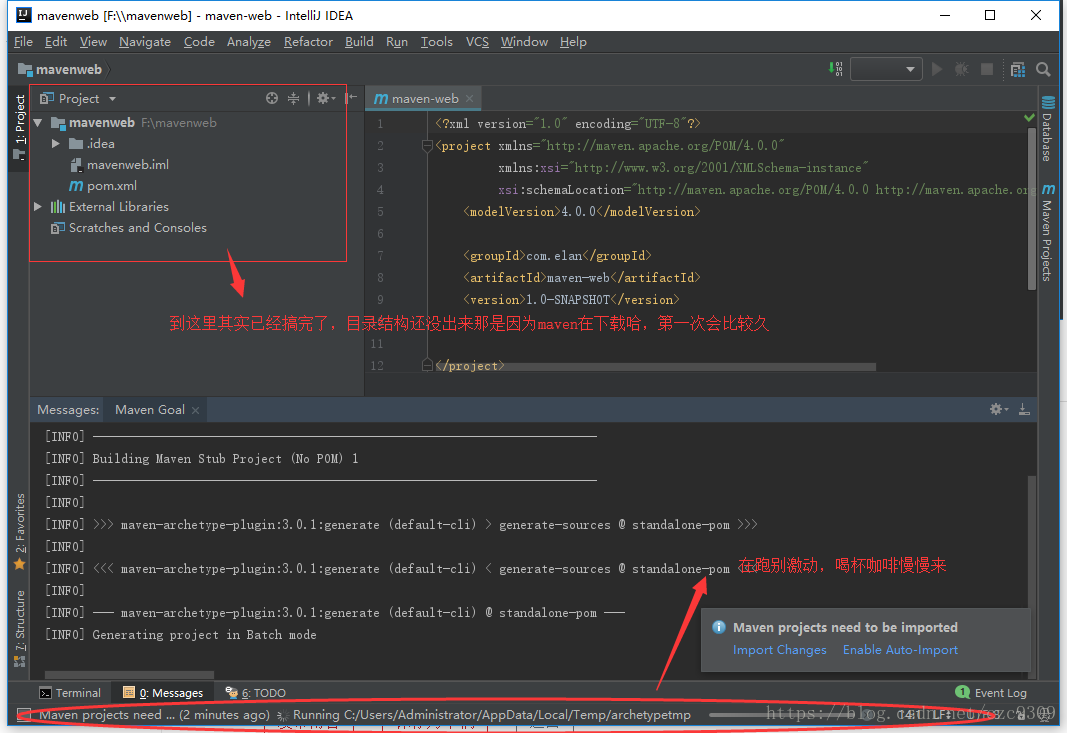


步骤五：直接finish，没什么坑哈

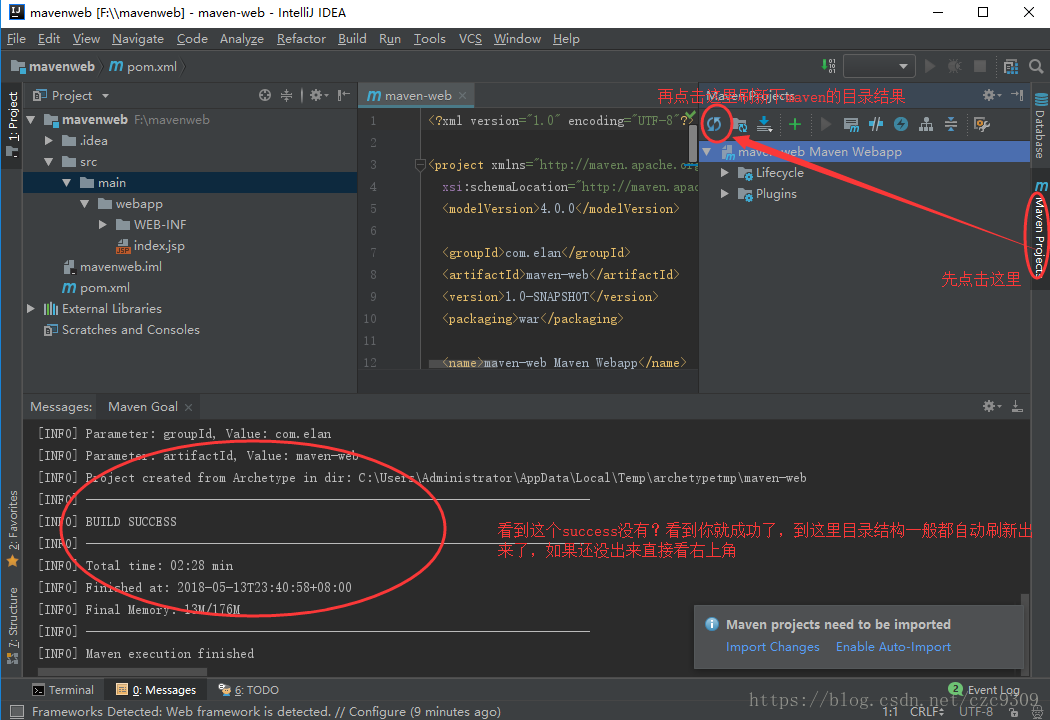


步骤六：喝杯咖啡，泡杯茶稍安勿躁，等一等（此时目录结构还没出来，我好几次这里没等就百度乱整，直接整坏了）

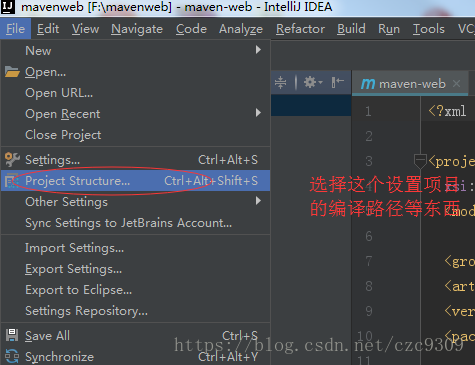
ps:由于有些idea设置问题可能不会自动导入jar包，这个时候点击右下角的提示Import Changes就可以导包拉



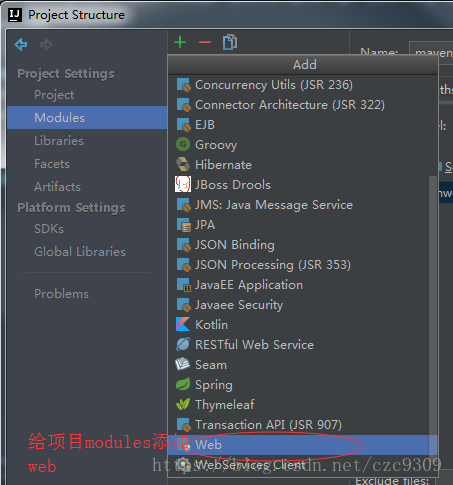
步骤七：等出结果，如果目录结构还不出来直接按图操作一波

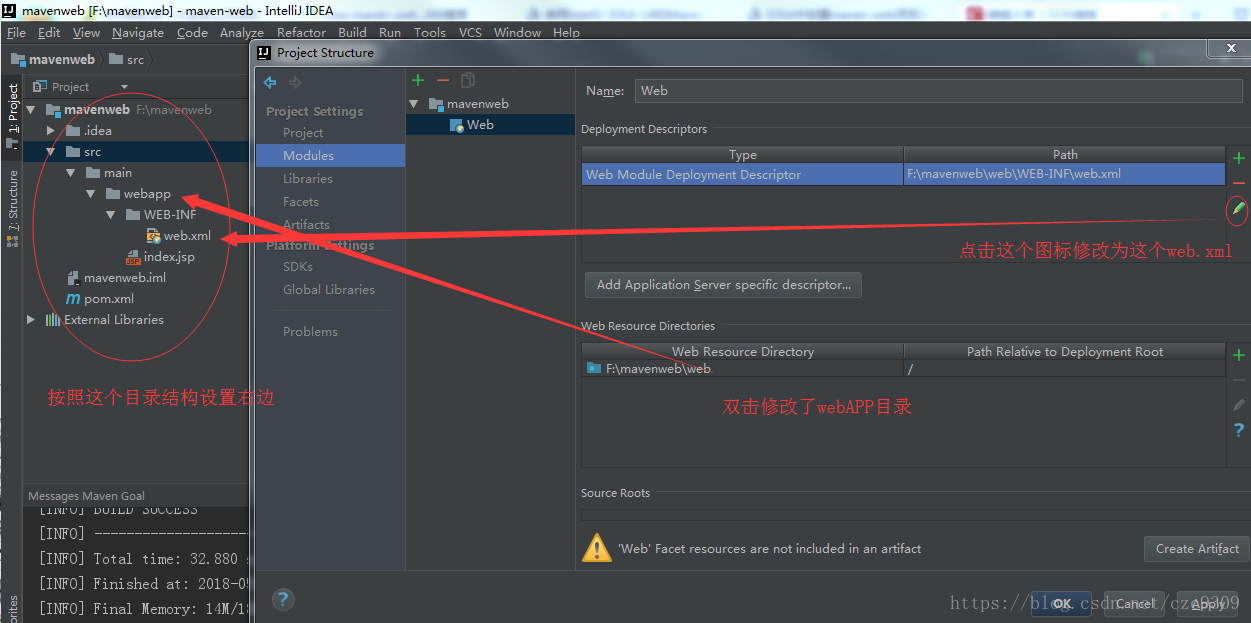


步骤八：项目的基本目录结构出来后我们最后还需要为项目配置下编译路径还有artifact，按图操作把

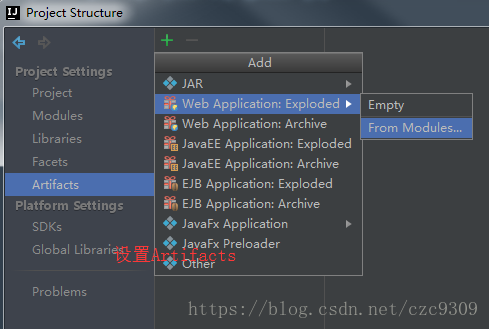


出现下图

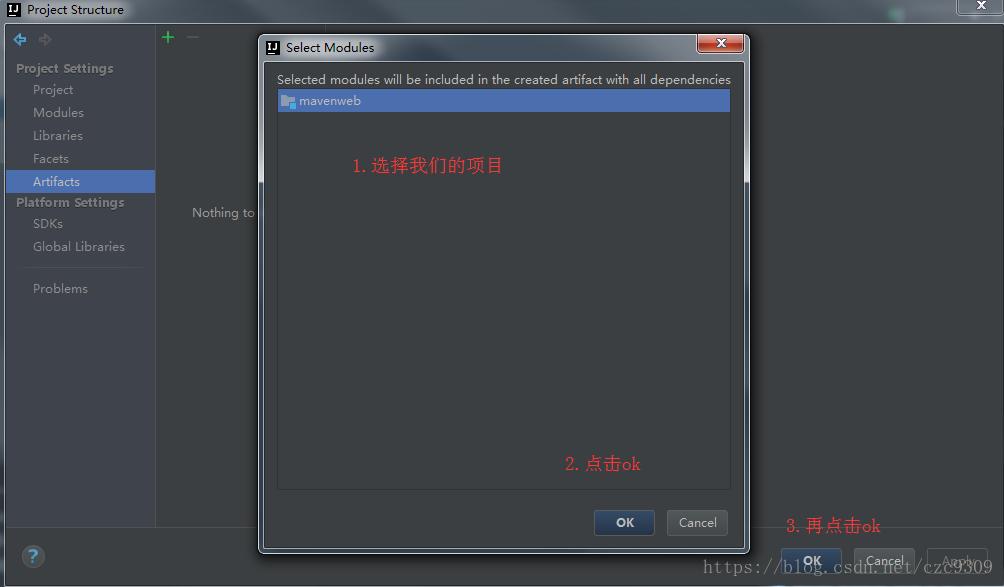




接着设置Artifacts

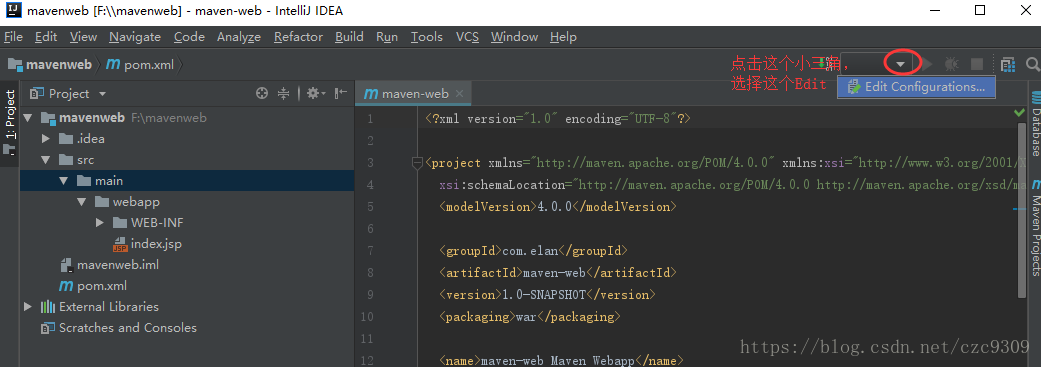


选择我们的项目

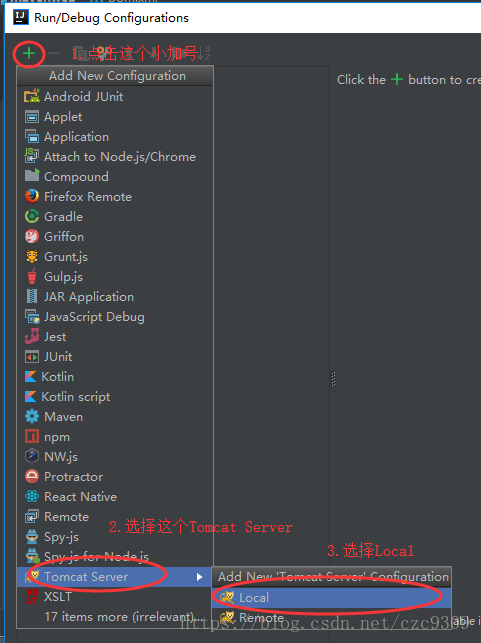


到了这里再配置tomcat跑下基本就ok了！

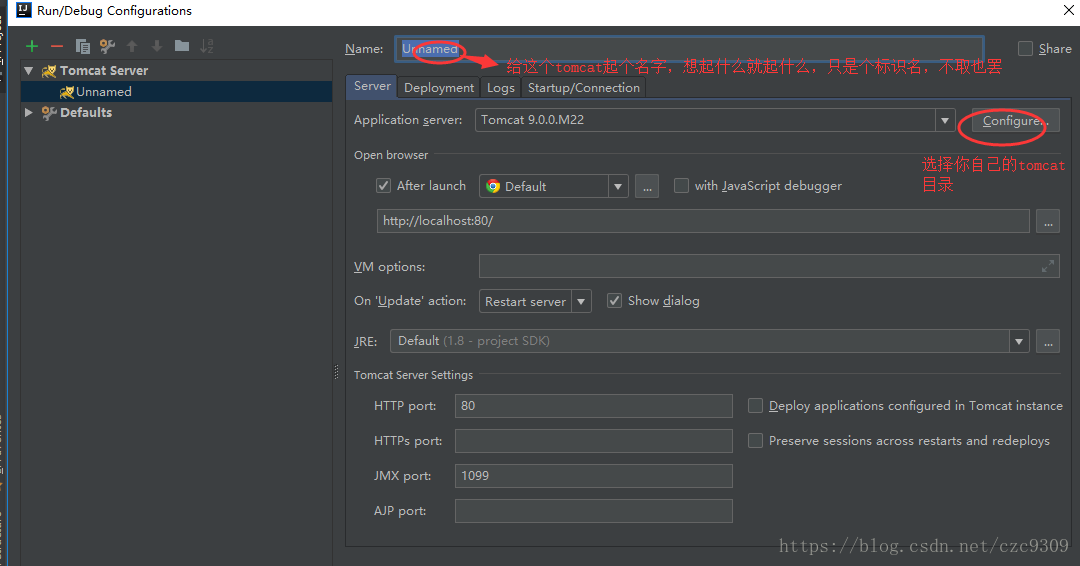
步骤九：配置Tomcat，上面目录结构都出来了基本就没有坑了，现在我们给它配置个tomcat---按图操作吧



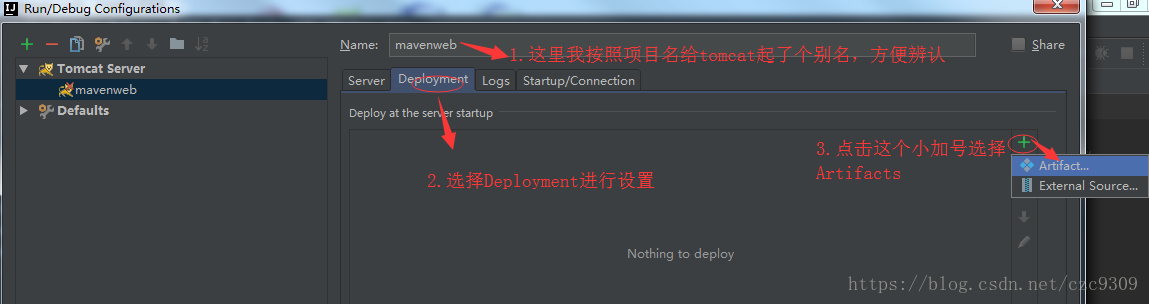
出现这个画面，然后按图再操作一波



接着出现这个画面，继续看图操作

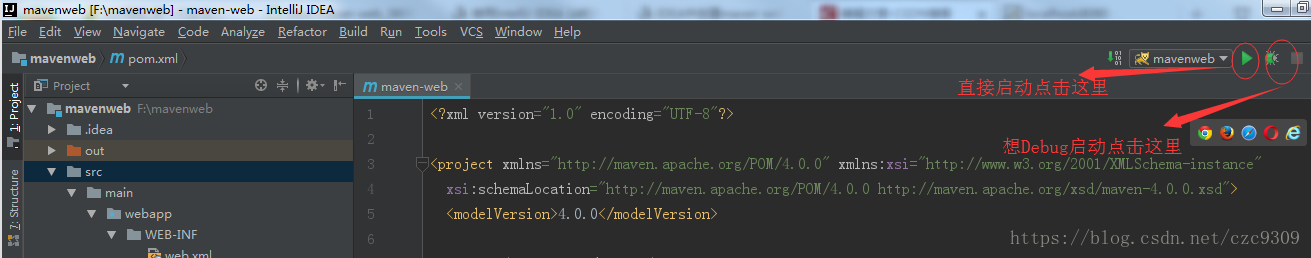


然后你选择右边第二个选项，Deployment，还是看图操作。。。

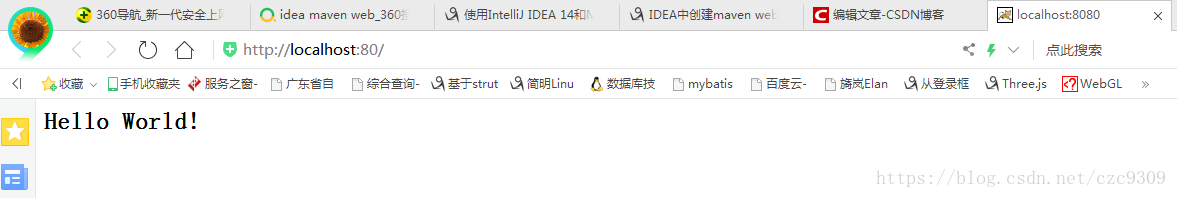


最后点击ok完成Tomcat配置，现在就可以启动tomcat跑跑我们的helloWorld了！

步骤十：启动tomcat，看效果



启动完idea一般会自动通过默认浏览器打开http://localhost:80/,如果没有就直接在直接的浏览器打这个地址，注意我这里tomcat自己设置了80端口

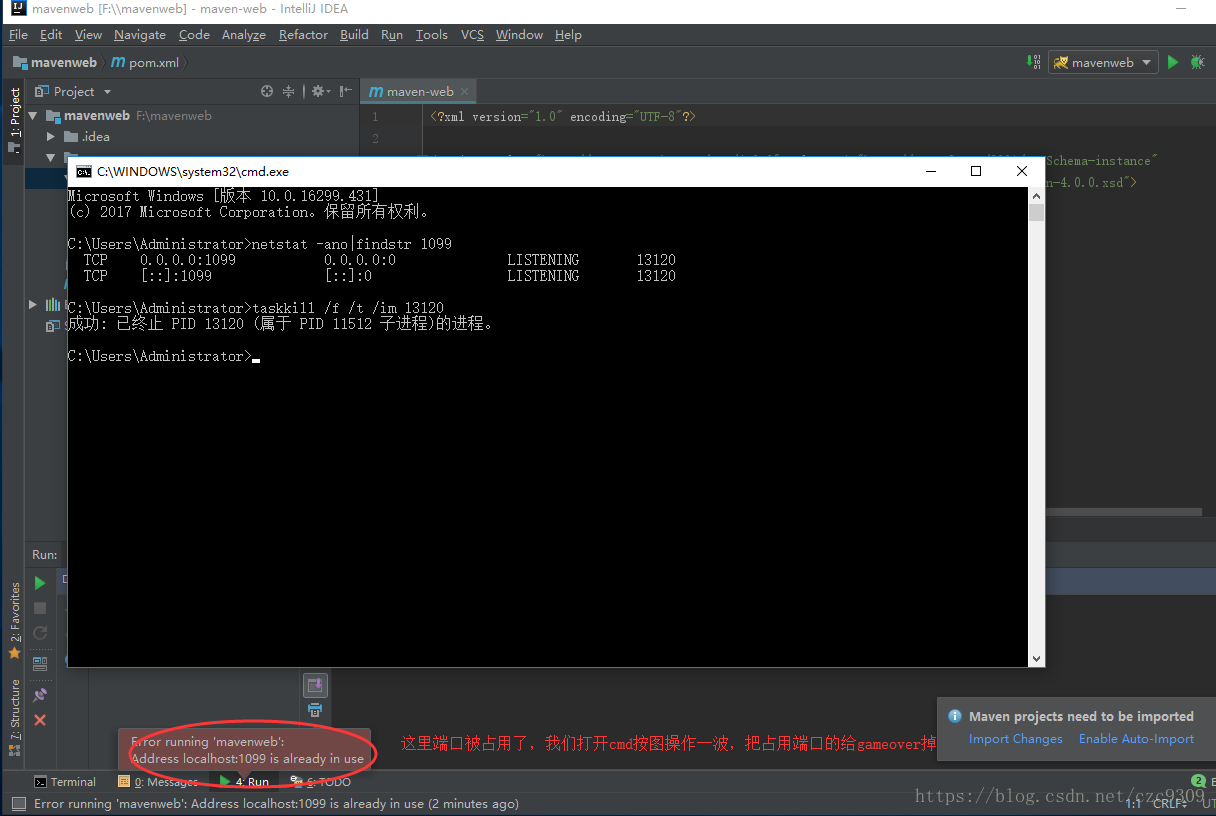


ps:我第二次搞的时候遇到端口被占用的情况也顺带贴出来，自己看图操作把

1.调出系统cmd窗口

2.通过netstat -ano|findstr 1099查询是哪个pid占用了端口

3.通过taskkill /f /t /im 13120直接结束掉占用端口的进程，搞定...



原文链接：<https://blog.csdn.net/czc9309/article/details/80304074>

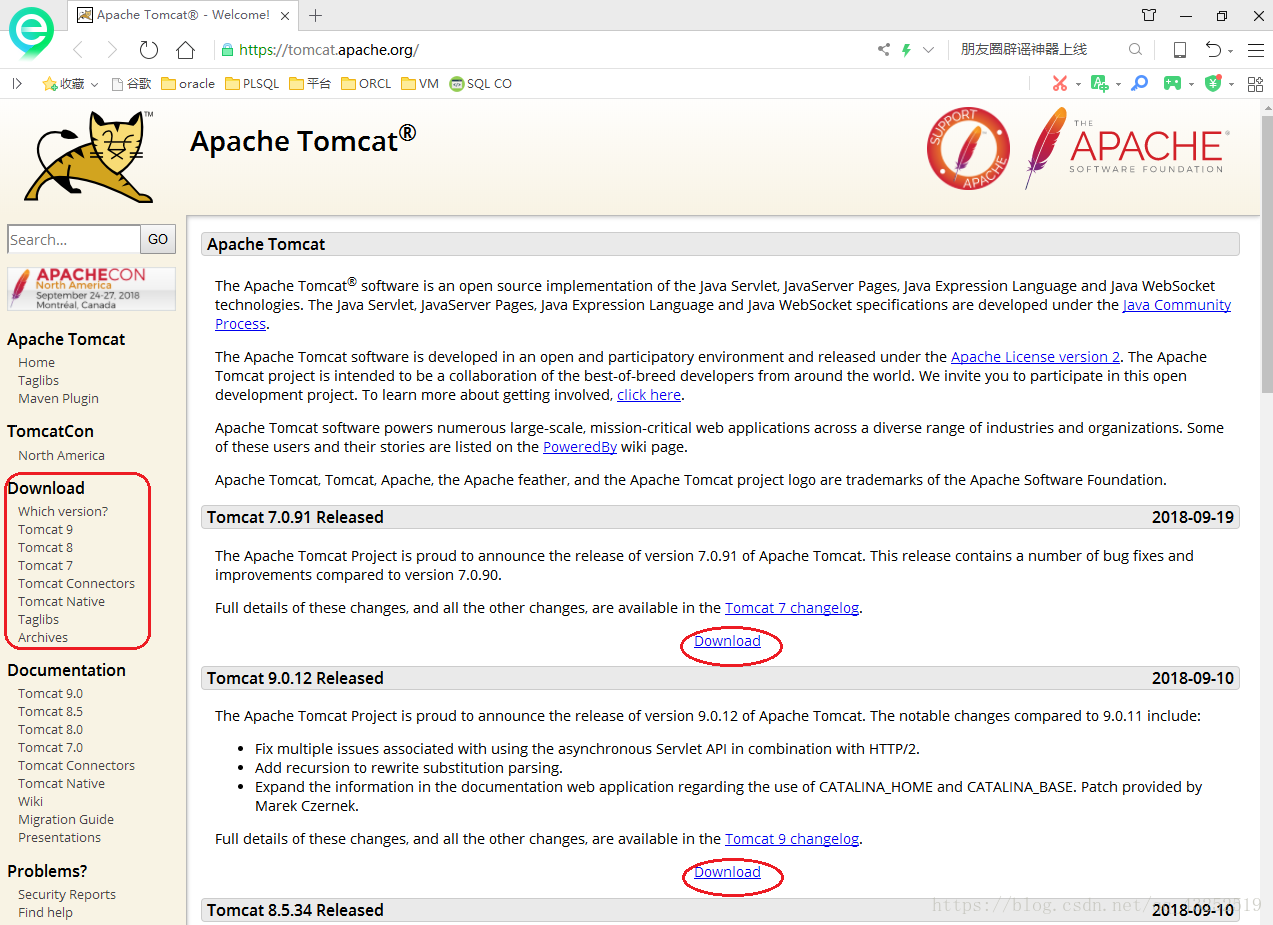
# Tomcat的安装与环境变量配置

Tomcat 用于deploy程序，建立服务器。但是在机器人后台，用的是spring boot 并不需要tomcat。

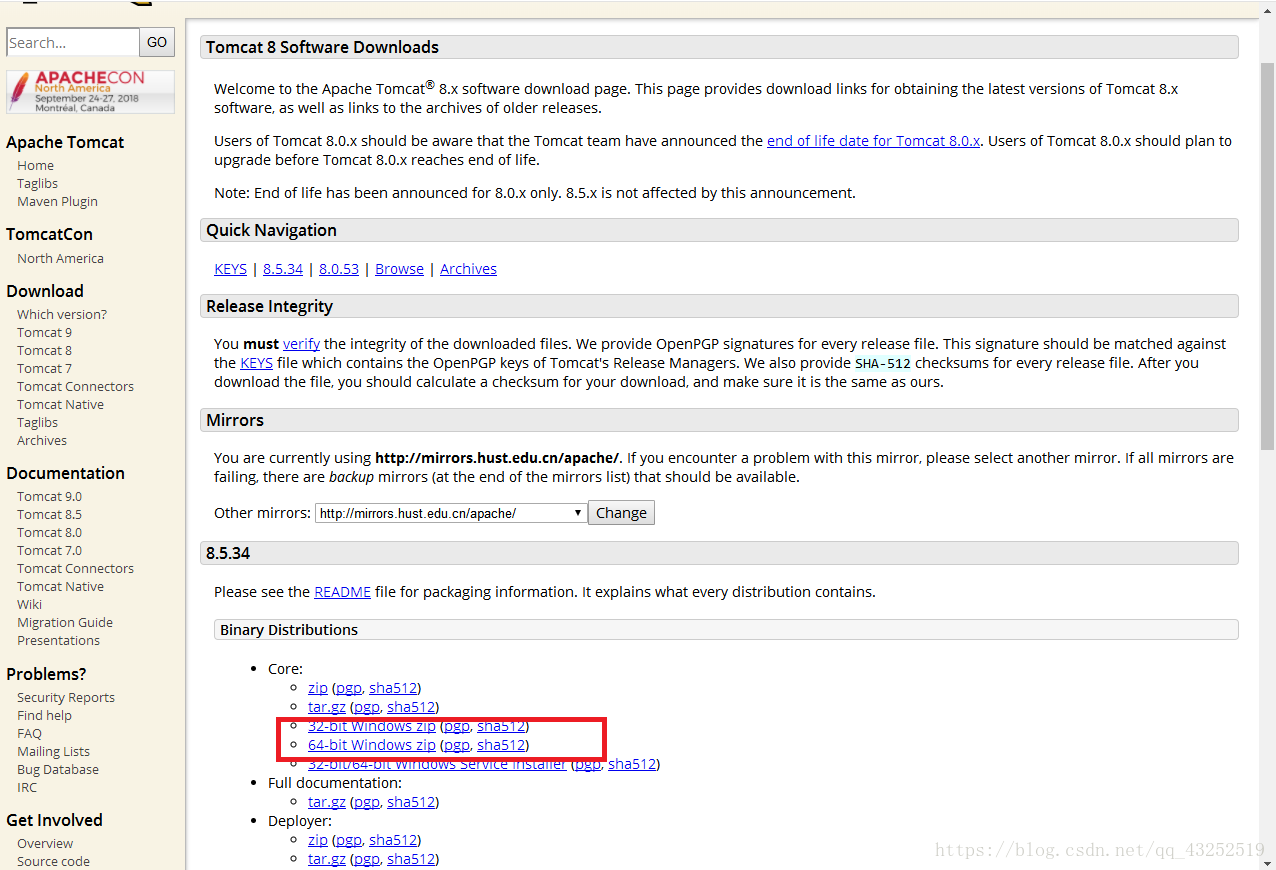
一、Tomcat的下载与安装

1.进入Tomcat官网： https://tomcat.apache.org/

2.在该网页左侧栏红框标记的download下，选择自己要下载的Tomcat的版本，点进去；也可以在右边红框标记处，点击下载。

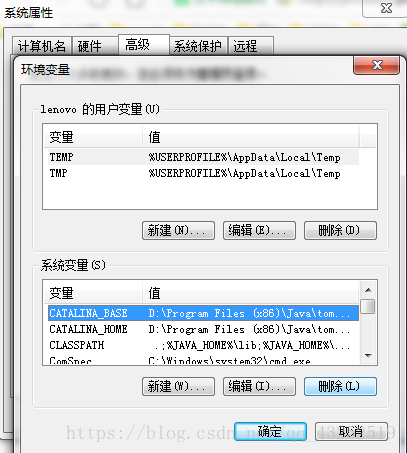


3.根据你电脑的位数，选择合适的进行下载。红框标记是免安装版本的，下载下来进行解压，就可以了。 这个解压位置也就是Tomcat的安装位置，记住这个位置，在配置环境变量时会用到。



二、环境变量的配置

右击“我的电脑”>>选择“属性”>>单击左侧栏“高级系统设置”>>“高级”>>“环境变量”



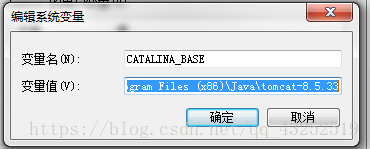
下面就要在系统变量这一栏里，配置Tomcat的环境变量了：

1.新建CATALINA\_BASE变量

变量名：CATALINA\_BASE

变量值：D:\Program Files (x86)\Java\tomcat-8.5.33

（填Tomcat的安装位置,免安装版的就是其解压位置）

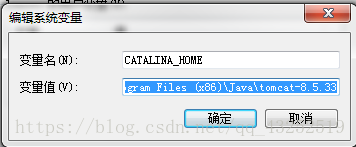


2.新建CATALINA\_HOME变量

变量名：CATALINA\_HOME

变量值：D:\Program Files (x86)\Java\tomcat-8.5.33

（填Tomcat的安装位置,免安装版的就是其解压位置）



3.找到Path变量，点击编辑

变量名：Path

变量值：直接复制，添加在原变量值的末尾 ;%CATALINA\_HOME%\bin;%CATALINA\_HOME%\lib

（注意，前面的那个分号是用来和原有的变量值隔开的，如果原来有的话，就不用写了）



到这里，所有的安装配置工作就都完成了。下面，我们来启动Tomcat验证一下，是否配置成功.

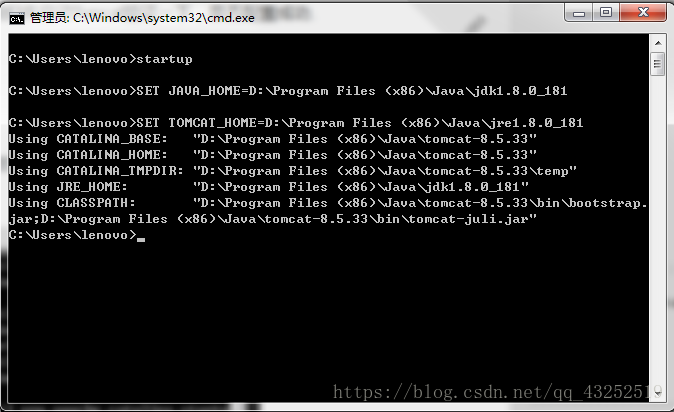
三、测试是否安装配置成功

启动Tomcat服务的方式有很多种，这里只列举了两种：

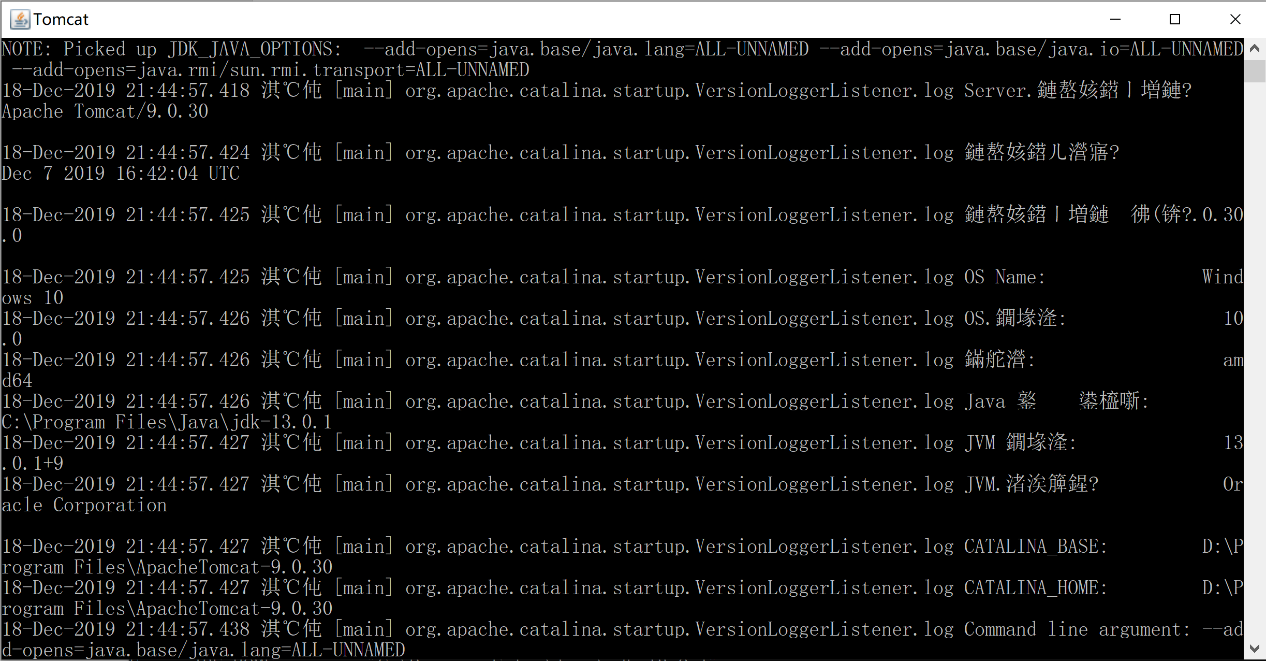
方法一

1.打开命令提示符窗口：

2.输入 startup ,按下回车键，会弹出一个标题栏为“Tomcat”的窗口，并出现如下图所示的信息。

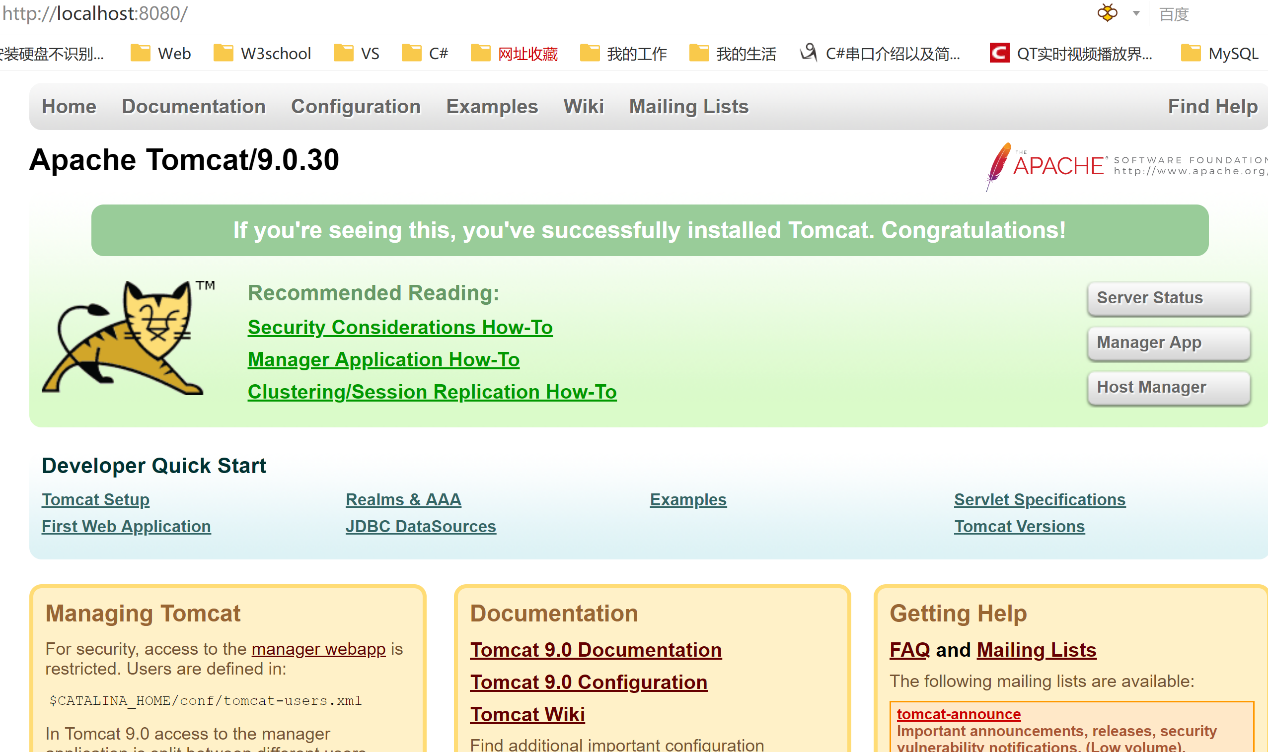


并且，弹出tomcat 程序启动的界面。弹窗 最后一句提示语句为[main] org.apache.catalina.startup.Catalina.start Server startup in [1,360] milliseconds。



3.在浏览器的地址栏输入：http://127.0.0.1:8080 或者 http://localhost:8080

如果能够显示下图所示界面，那么就说明Tomcat安装配置成功啦！！



方法二：

1.找到Tomcat的安装目录下的bin文件，我的地址是：D:\Program Files (x86)\Java\tomcat-8.5.33\bin;

2.打开bin文件，找到startup.bat，双击该文件，会弹出一个跟方法一中相同的tomcat窗口。

（弹窗显示的最后一句话，Server startup in 533 ms, 就是说服务启动耗时533毫秒）

如果想要关闭Tomcat，那么只要双击bin文件下的shutdown.bat，就可以了。

原文链接：<https://blog.csdn.net/qq_43252519/article/details/82883708>

# Maven的安装与配置

**一、什么是maven**

　　Maven基于项目对象模型(POM Project Object Model)，可以通过一小段描述信息（配置文件）来管理项目的构建、报告和文档的软件项目管理工具。

　　同时也是跨平台的项目管理工具。主要服务于基于Java平台的项目构建，依赖管理和项目信息管理。在项目构建中，maven实现高度自动化，跨平台，可重用的组件，标准化。　　在项目依赖管理中，maven提供pom.xml配置，实现自动下载，统一依赖管理。项目信息管理，可以有效记录项目名称描述，开发人员信息等。

**二、maven的优势**

　　1.传统方式管理jar依赖的问题：

　　　　jar冲突

　　　　jar依赖

　　　　jar体积过大

　　　　jar在不同阶段无法个性化配置

　　2.使用maven方式管理jar依赖的好处：

　　　　解决jar冲突

　　　　解决jar依赖问题

　　　　jar文件不用在每个项目保存，只需要放在仓库即可

　　　　maven可以指定jar的依赖范围

**三、maven安装**

1.maven安装环境要求：Maven 3.6.3+ 需要使用jdk 1.7+

2.下载Maven

进入maven官网：http://maven.apache.org/download.cgi

下载 Maven 的 zip 文件，例如：apache-maven-3.6.3-bin.zip。

3.解压文件

把压缩包“apache-maven-3.6.3.bin.zip”，解压到想要的路径，如D:\java\apache-maven，注意路径中不要包含中文。maven不用安装，解压就能用，但要配置环境信息。

Maven 包含的文件夹如下：

* bin：含有mvn运行的脚本
* boot：含有plexus-classworlds类加载器框架
* lib：含有Maven运行时所需要的java类库
* conf：含有settings.xml配置文件。在配置文件settings.xml中，需要将mirrors中的内容更改为阿里云镜像，不然下载速度很慢，而且大概率失败。settings.xml 中默认的用户库: ${user.home}/.m2/repository[通过maven下载的jar包都会存储到此仓库中]

4.配置环境变量

　　在环境变量中，新增系统变量名：MAVEN\_HOME，变量值：D:\Program Files\ApacheMaven-3.6.3; (即maven解压路径)

　　MAVEN\_OPTS : -Xms256m -Xmx512m（注意：可以不配置）

　　在Path中追加：%MAVEN\_HOME%\bin;

　5.验证是否安装成功

　　 在命令行(cmd)窗口中输入：mvn –v，如果看见下图则说明maven安装成功。

Maven安装成功后，第一次命令行输入mvn help:system会在C:\Users\用户\下生成.m2文件，里面会定义默认的镜像和默认的maven仓库，及存放下载jar包的文件夹。 IDEA会自动配置maven的setting.xml关联到这个文件。

实际使用时，一般是将maven安装目录下的.\conf\settings.xml进行修改。并且IDEA工程配置为此settings.xml文件。改动如下：新建 .\maven-repository文件夹，然后修改setting.xml中的在<localRepository>,将其改为刚刚新建的maven-repositor文件夹。

<localRepository>

D:\Program Files\ApacheMaven-3.6.3\maven-repository </localRepository>。

这样新下载的jar包就可以下载到D:\Program Files\ApacheMaven-3.6.3\maven-repository这个路径下，不用每次都下载到C盘下了。

至此，maven安装完毕！

# IDEA中Maven依赖下载失败解决方案

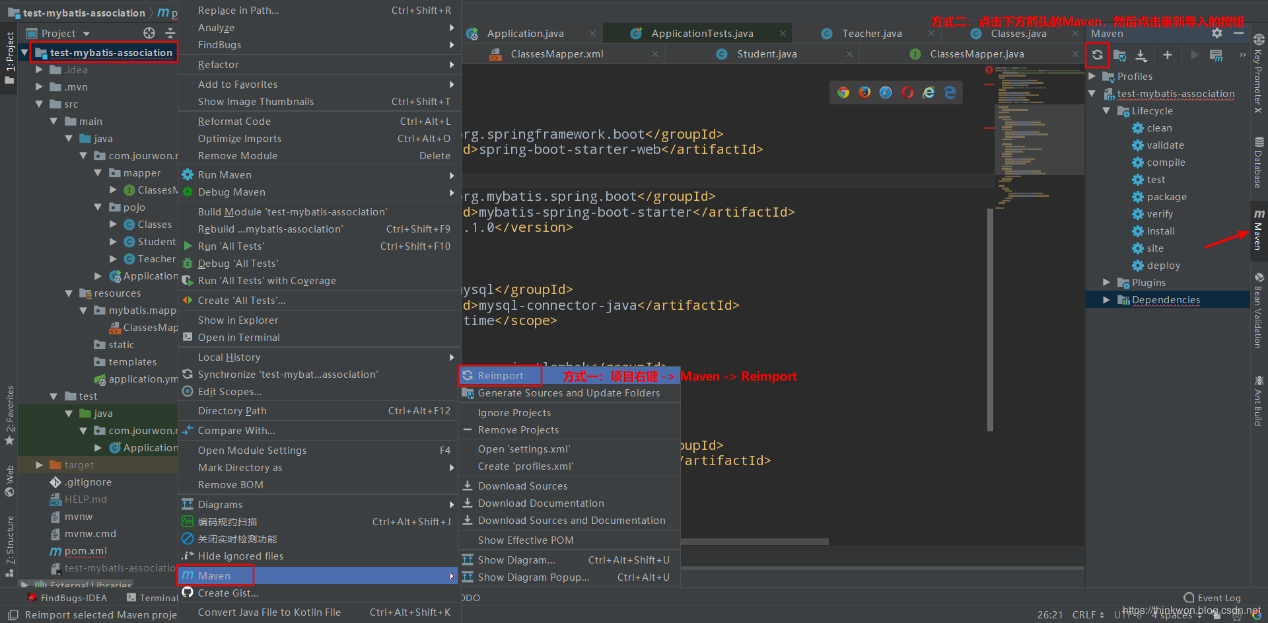
主要参考链接：

1. <https://blog.csdn.net/ThinkWon/article/details/101312918>

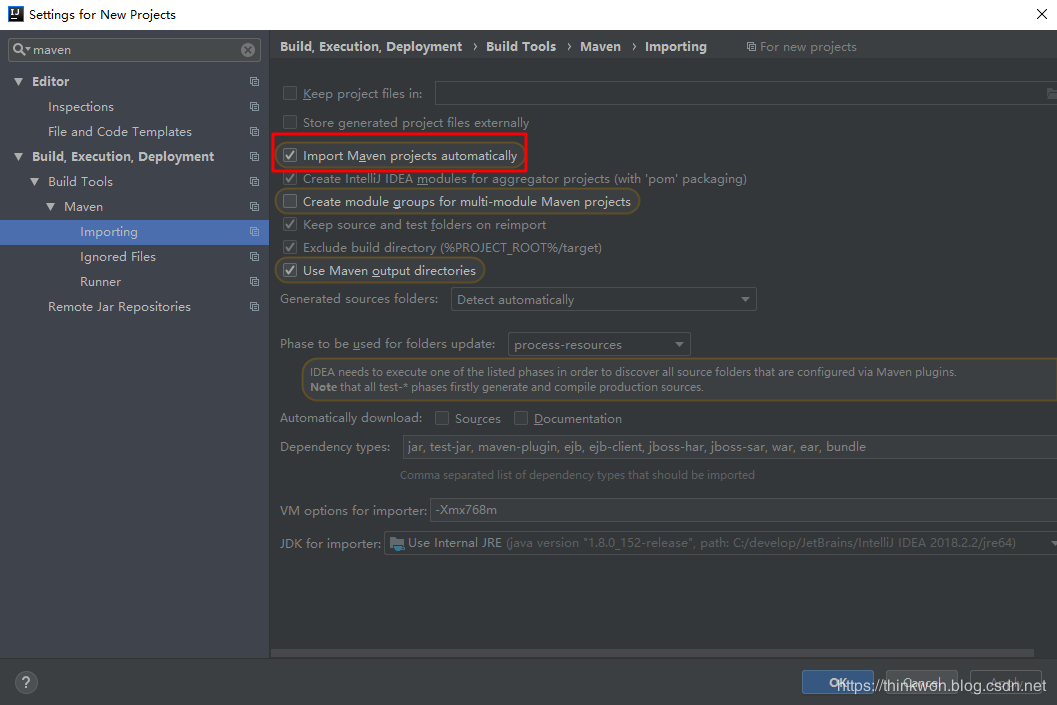
主要步骤：

1. 首先检查网络有没有问题，确定网络没有问题，请看下一步
2. 多次点击重新导入Maven依赖的按钮
3. 设置自动导入Maven依赖
4. 在IDEA中找到Maven的配置文件的地址，然后检查配置的远程仓库或者镜像有没有问题
5. 如果上面几步都没有解决问题，可以使用以下脚本删除Mvaen中的lastUpdated文件
6. Maven仓库依赖存在依旧报错
7. 首先检查网络有没有问题，确定网络没有问题，请看下一步
8. 多次点击重新导入Maven依赖的按钮

重新导入Maven依赖有两种方式，如上图所示。如果多次点击重新导入依赖按钮依然报错，请看下一步

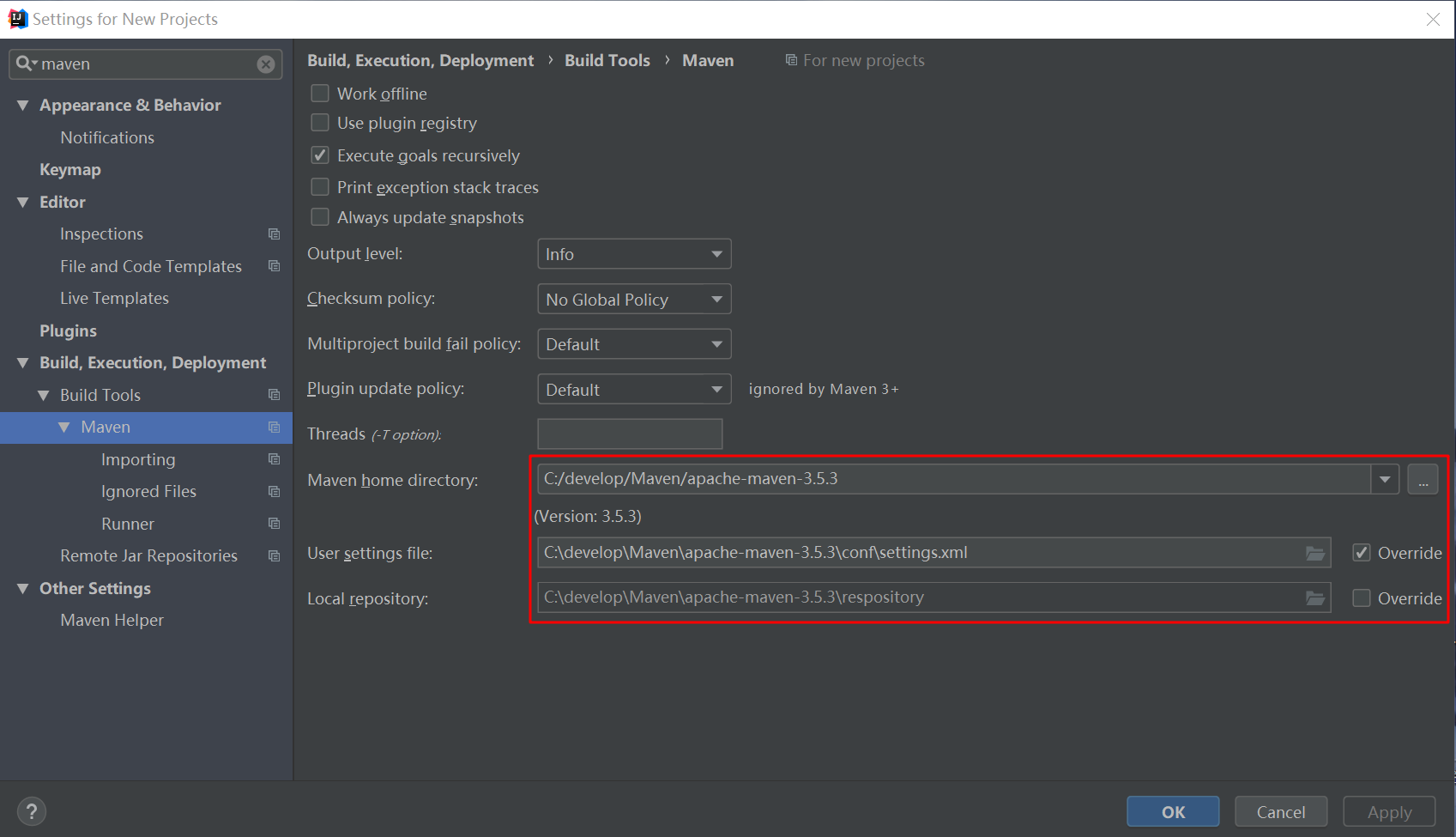


设置自动导入Maven依赖，Settings -> Build,Execution,Deployment -> Build Tools -> Maven -> Importing，如下图



这样设置后，如果Maven仓库存在依赖的话，IDEA会自动导入到项目中，如果没有用，看下一步

在IDEA中找到Maven的配置文件的地址，然后检查配置的远程仓库或者镜像有没有问题



如上图所示，我的配置在C:\develop\Maven\apache-maven-3.5.3\conf\settings.xml

我配置的是阿里云仓库，没有什么问题，如果配置的是Maven私服Nexus的话，需要检查配置的路径和私服网络有没有问题

<!-- 配置阿里云仓库 -->

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>

http://maven.aliyun.com/nexus/content/groups/public/

</url>

<mirrorOf>central</mirrorOf>

</mirror>

# 使用IDEA建立SpringBoot工程

主要参考网址：

<https://blog.csdn.net/chy555chy/article/details/84970042>

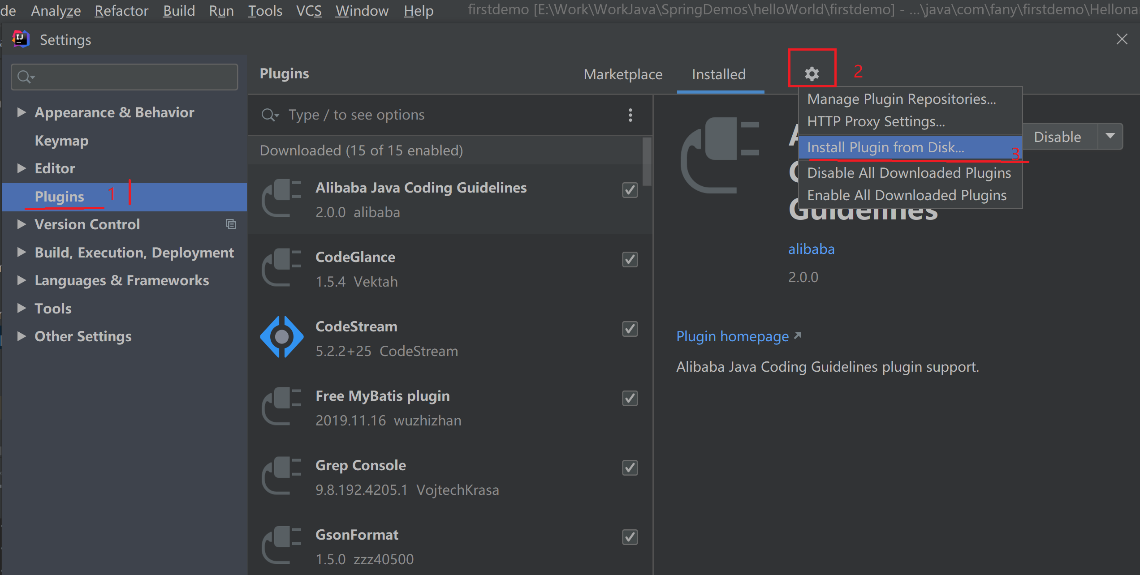
<https://blog.csdn.net/baidu_39298625/article/details/98102453>

最开始，使用IDEA创建工程时，由于工程种类真的太多。Web开发路线过多。因此一直找不到“项目后台”的正确创建方式。开始的时候遇到了没有maven选项的问题，后来遇到maven下载jar包失败，需要reimport的问题。最后终于发现是通过spring assistant 插件进行建立工程。详见以下记录。

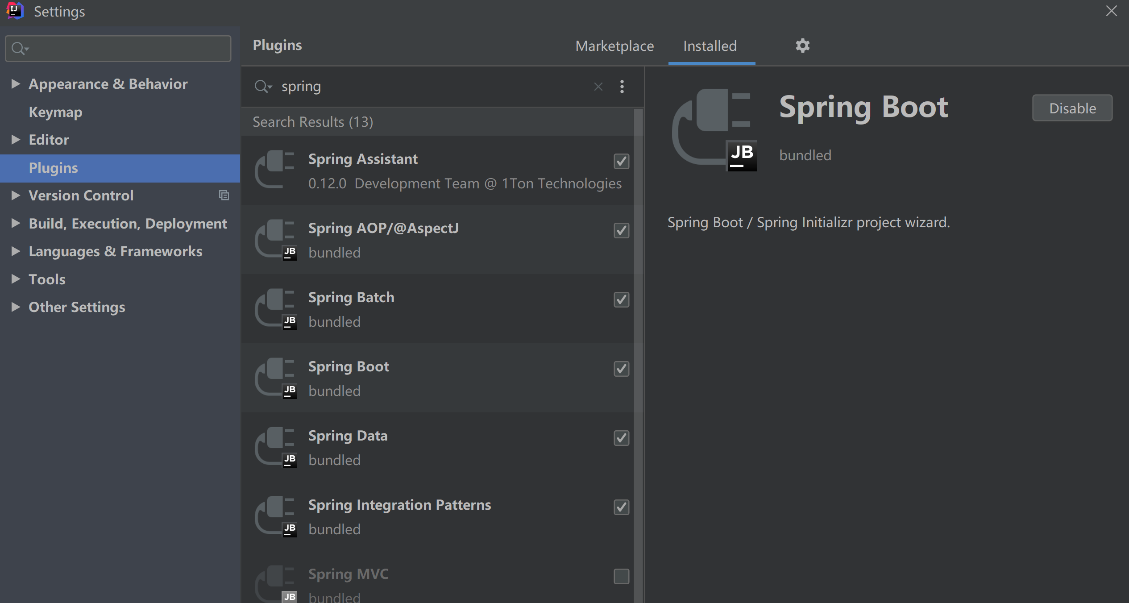
跳坑: 最大的坑，在于不要使用 IDEA建立maven工程后，添加spring框架。这样会有很多问题。而应该，使用IDEA通过 spring initializr 或者 spring assistant 建立 maven 工程。

1. **下载并安装spring assistant插件**

打开[http://plugins.jetbrains.com/plugin](http://plugins.jetbrains.com/plugin/8321-free-mybatis-plugin/versions)网址，搜索并下载spring assistant 插件。下载完毕后，在IDEA界面下，“Ctrl+Alt+S”或者“File/Settings” 找到对应的plugins。点击设置图标，选择”Install Plugin from Disk” , 如下图的所示步骤。选择需要安装的插件。插件安装后需要重新启动IDEA。



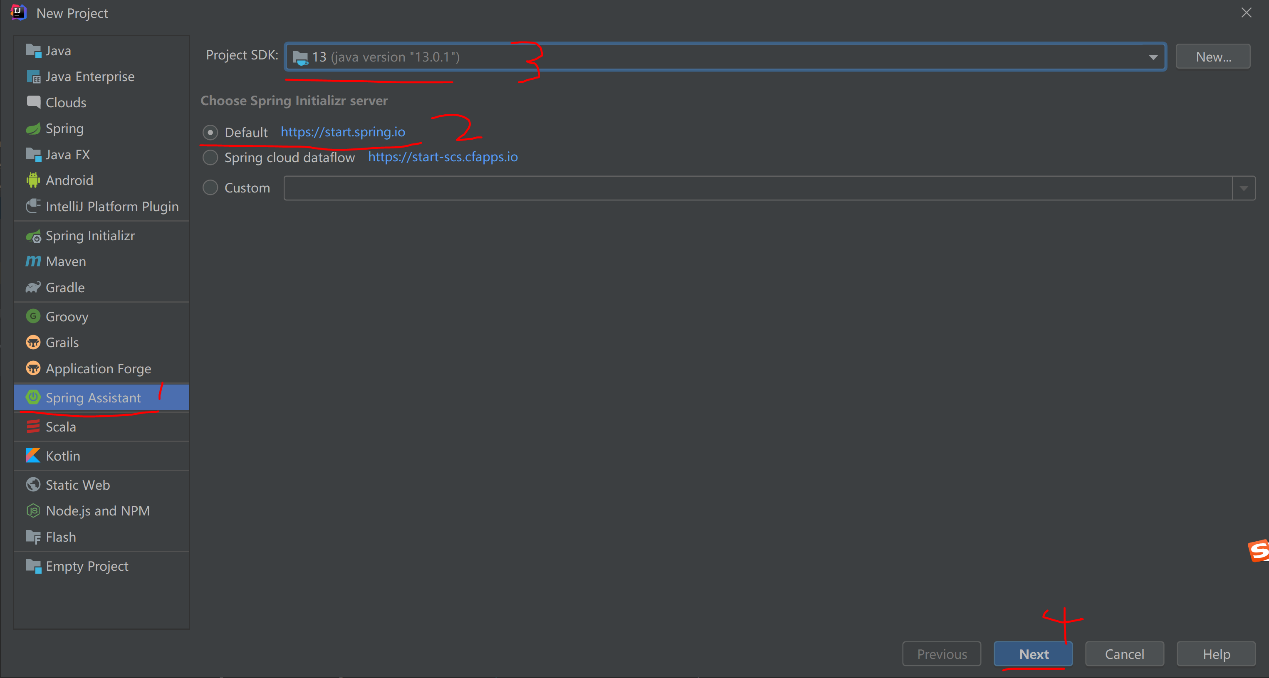
跳坑： springboot工程不能Debug/run configuration的问题。 一般来说，IDEA安装后，会自动打包安装spring。但有时候工程建立后，却不能进行debug/run。这是时则需要在plugins中勾选 Spring Boot，使其Enable。



1. **创建springboot 工程**

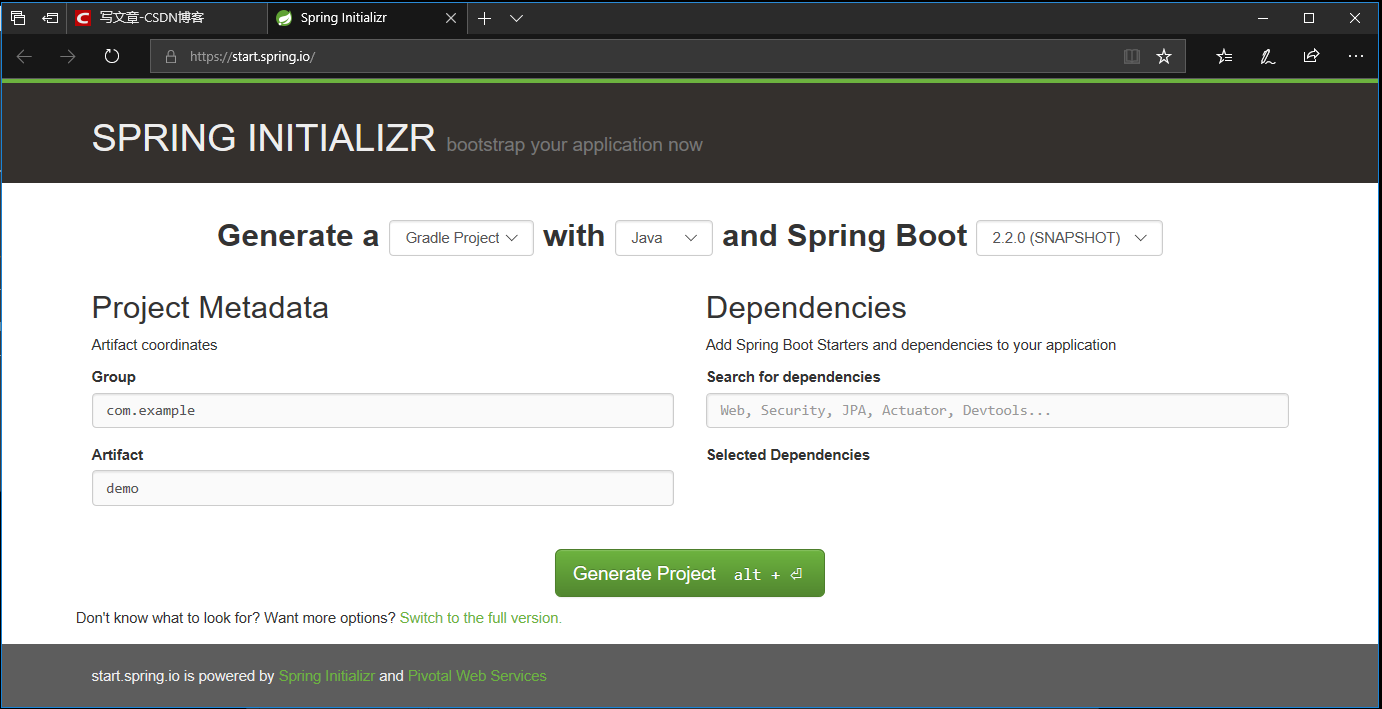
2.1 **创建项目**

File->New->Project->SpringInitializer或者Spring Assistant。选择 然后点Next。可以看到 <https://start.spring.io/> 也是一个填写项目配置信息的网页，它会直接生成一个压缩好的项目供我们下载。IntelliJ IDEA 其实也是用这个原理创建的项目，只不过这一切都在后台替我们完成了。

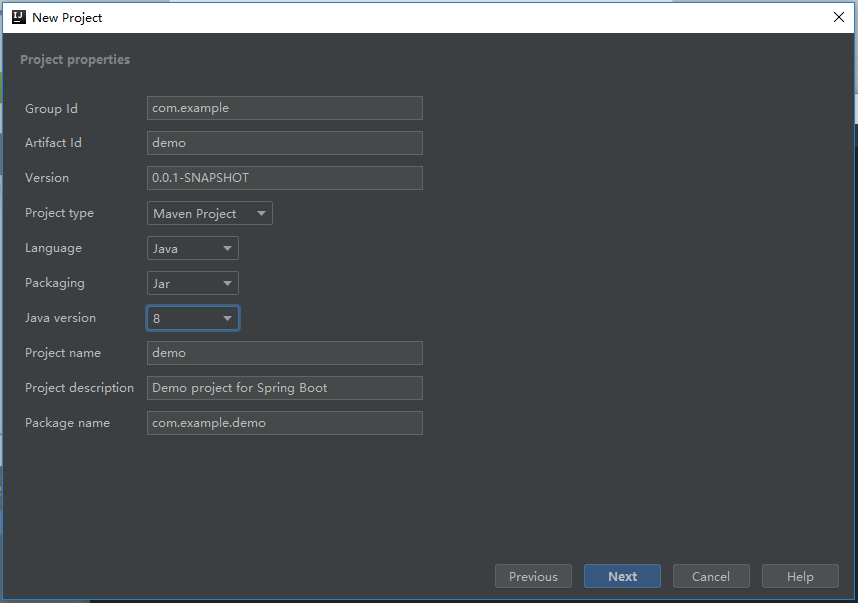


**2.2 设置项目参数**

设置工程的参数，包括GroupId（公司ID）, Artifact ID （项目ID），项目类型选为maven，语言选择java，输入工程名称等，然后Next。



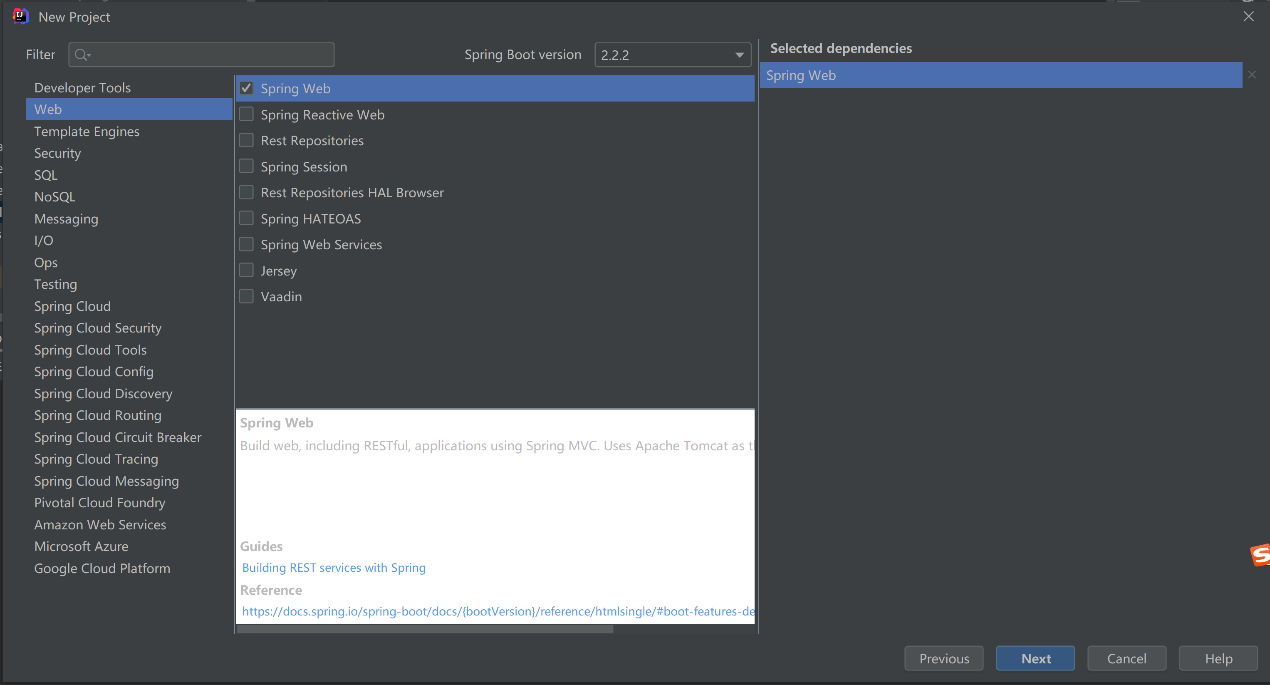
图x.  <https://start.spring.io/> 打开后的页面



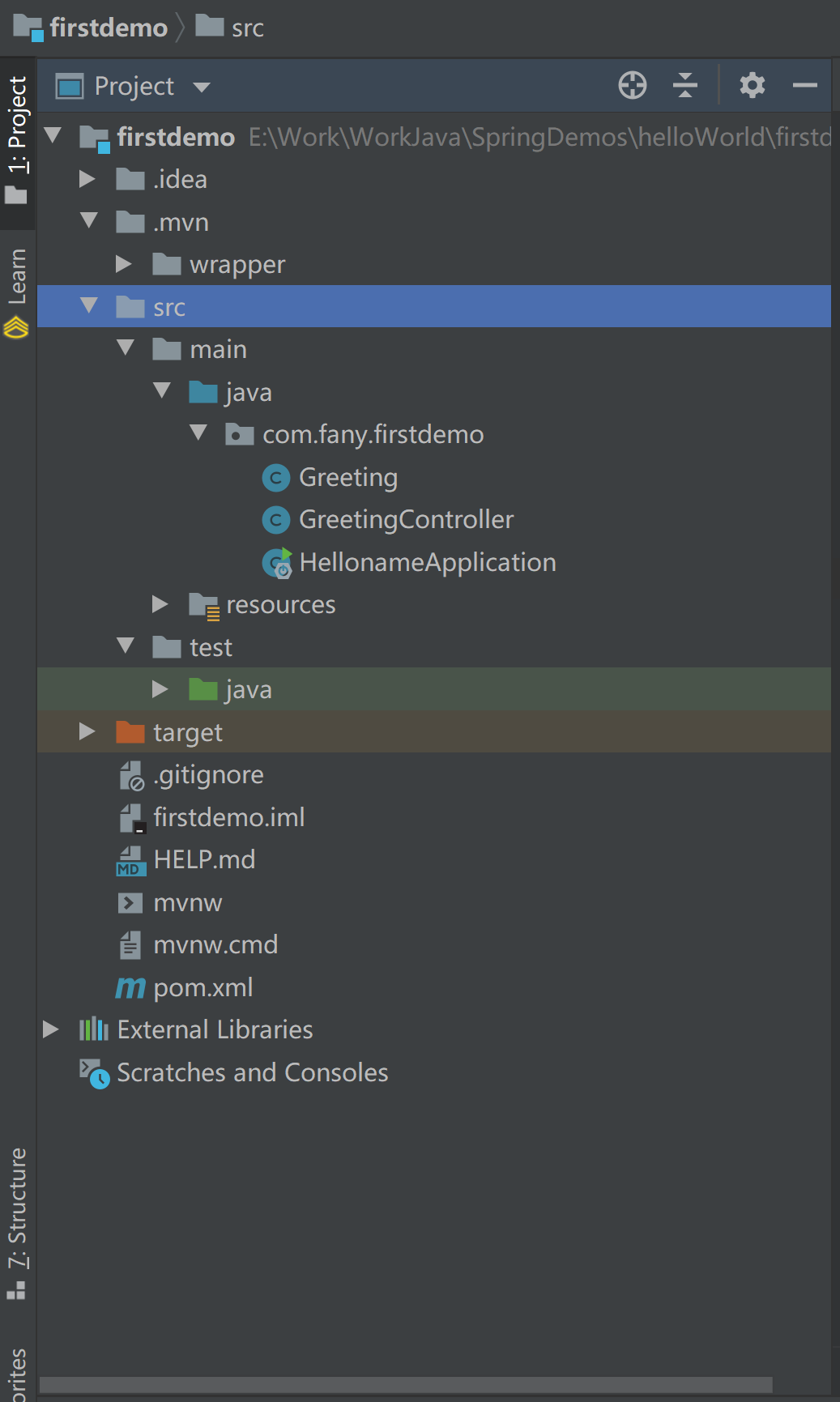
图x. IDEA 封装的<https://start.spring.io/>的配置界面

**2.3 选择Web 模块**

接着左边选择 Web, 右边只勾选 Web 即可，然后点击Next

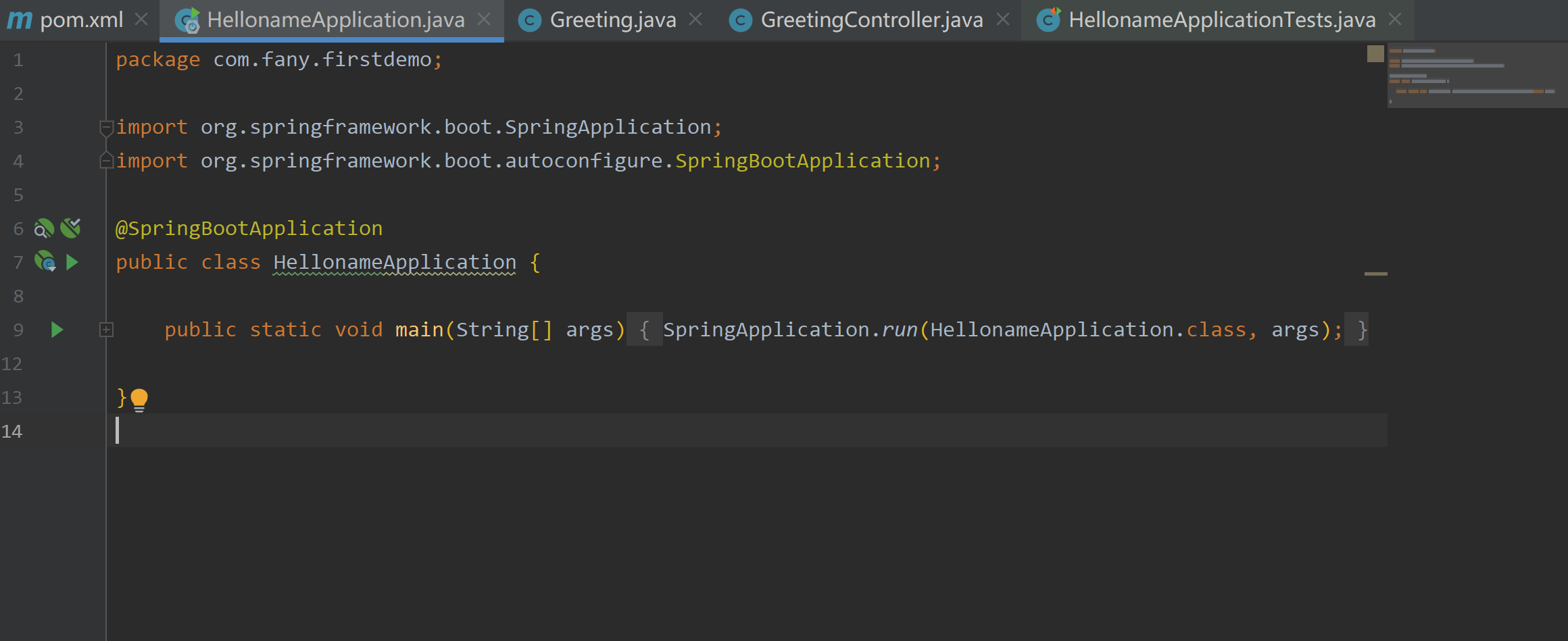
**2.4 指定项目的路径**

指定项目路径为 E:\project\springboot (其他位置也可以)。如此这般之后，项目就创建成功了，就可以看到项目结构了。



7.运行并调试

项目创建后，就自带一个XXXApplication, 其被@SpringBootApplication所标记，表示这个是一个Springboot 应用。



将 springframework 的官网demo添加到这个工程的中，代码略去。

**9. 运行并测试**

接下来就运行 SpringbootApplication.java， 然后访问地址<http://127.0.0.1:8080/greeting?name>, 就能看到测试效果了。

