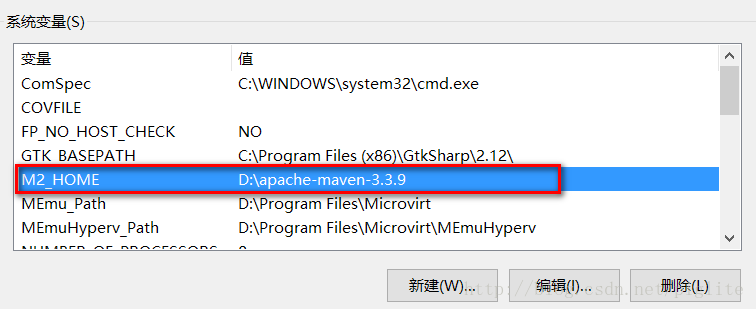
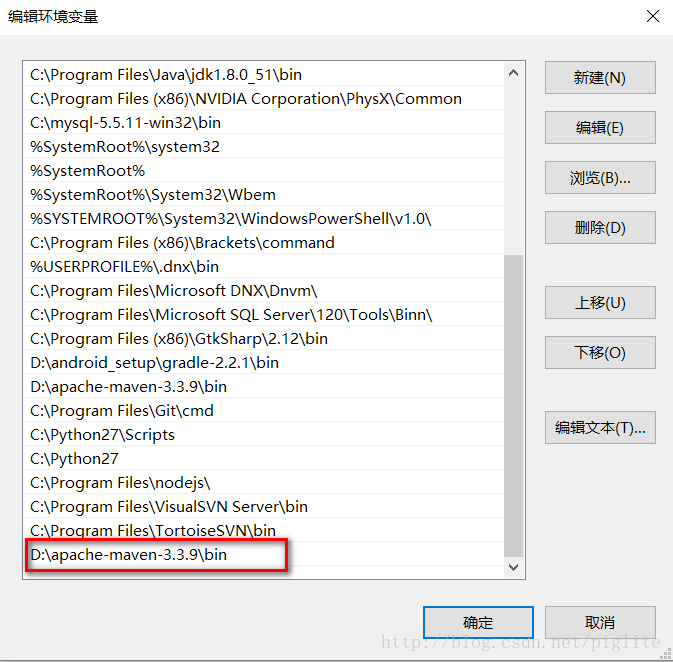
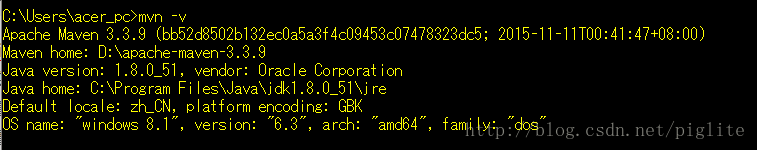
<http://blog.csdn.net/piglite/article/details/54412710>

版权声明：本文为博主原创文章，未经博主允许不得转载。

1. 使用Intellij Idea 2016.3开发工具，开发环境是JDK 1.8 + Maven 3.3.9，服务器为Tomcat 8
2. 配置环境变量：   
   Java的配置略去   
   Maven配置如下：   
   添加环境变量   
     
   在Path中加入Maven的相关内容   
   

配置成功后，在cmd中输入mvn -v 可以看见版本信息   


1. 在Windows的当前用户文件夹下创建一个目录.m2，然后从maven安装目录的conf子目录中复制过去settings.xml，并打开编辑。   
   主要编辑该文件中的两个地方：   
   添加localRepository，指明本地仓库

<localRepository>d:\mvnrepo\repository</localRepository>

* 1
* 1

添加网络中央仓库的镜像

<mirrors>

<!-- mirror

| Specifies a repository mirror site to use instead of a given repository. The repository that

| this mirror serves has an ID that matches the mirrorOf element of this mirror. IDs are used

| for inheritance and direct lookup purposes, and must be unique across the set of mirrors.

|

<mirror>

<id>mirrorId</id>

<mirrorOf>repositoryId</mirrorOf>

<name>Human Readable Name for this Mirror.</name>

<url>http://my.repository.com/repo/path</url>

</mirror>

-->

<mirror>

<id>alimaven</id>

<name>aliyun maven</name>

<url>http://maven.aliyun.com/nexus/content/groups/public/</url>

<mirrorOf>central</mirrorOf>

</mirror>

<mirror>

<id>UK</id>

<name>UK Central</name>

<url>http://uk.maven.org/maven2</url>

<mirrorOf>central</mirrorOf>

</mirror>

<mirror>

<id>ibiblio.org</id>

<name>ibiblio Mirror of http://repo1.maven.org/maven2</name>

<url>http://mirrors.ibiblio.org/pub/mirrors/maven2</url>

<mirrorOf>central</mirrorOf>

</mirror>

<mirror>

<id>jboss-public-repository-group</id>

<name>JBoss Public Repository Group</name>

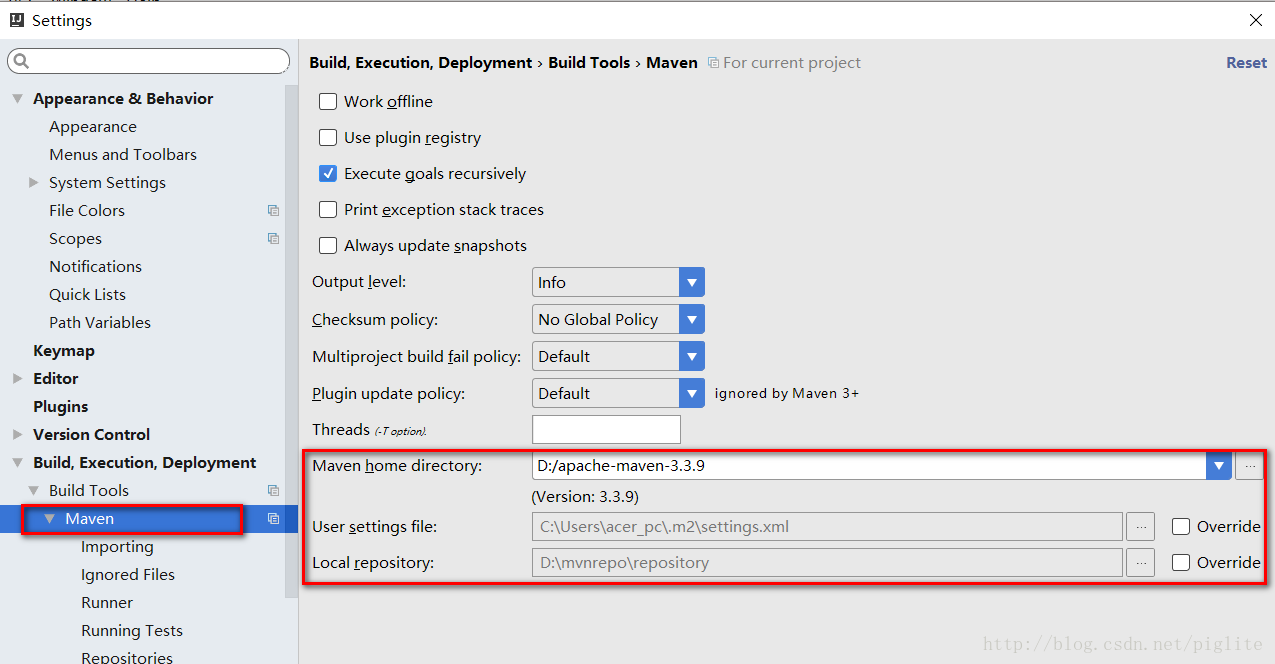
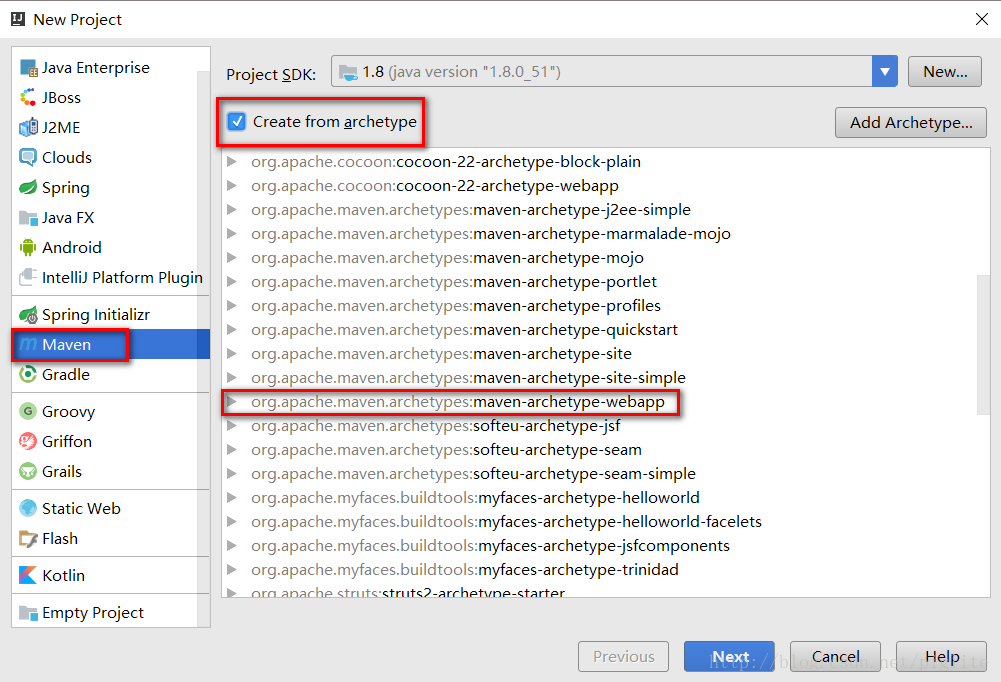
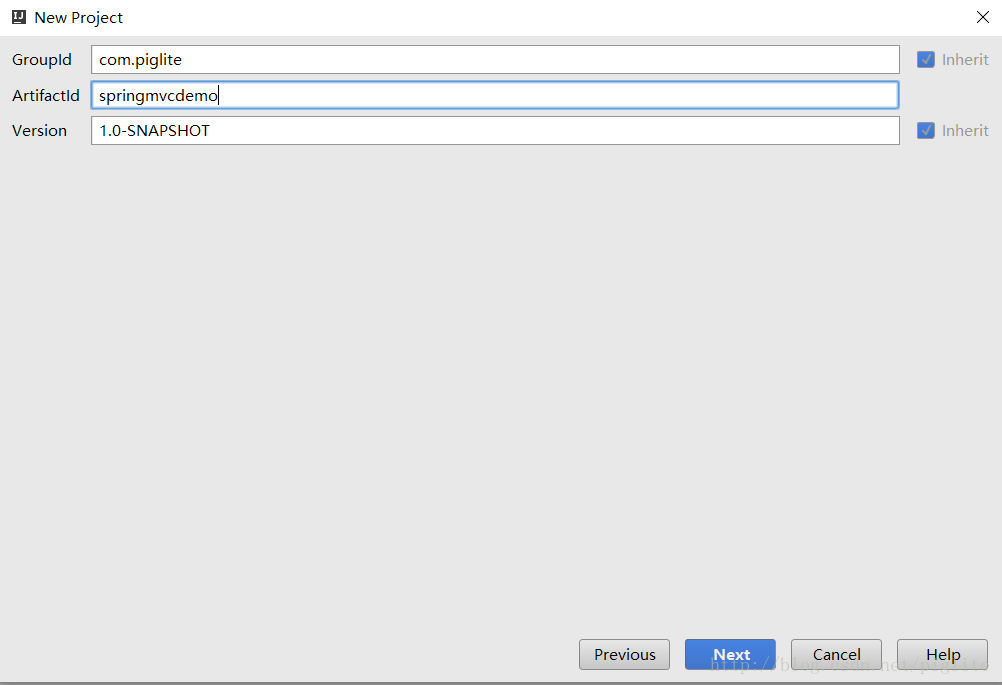
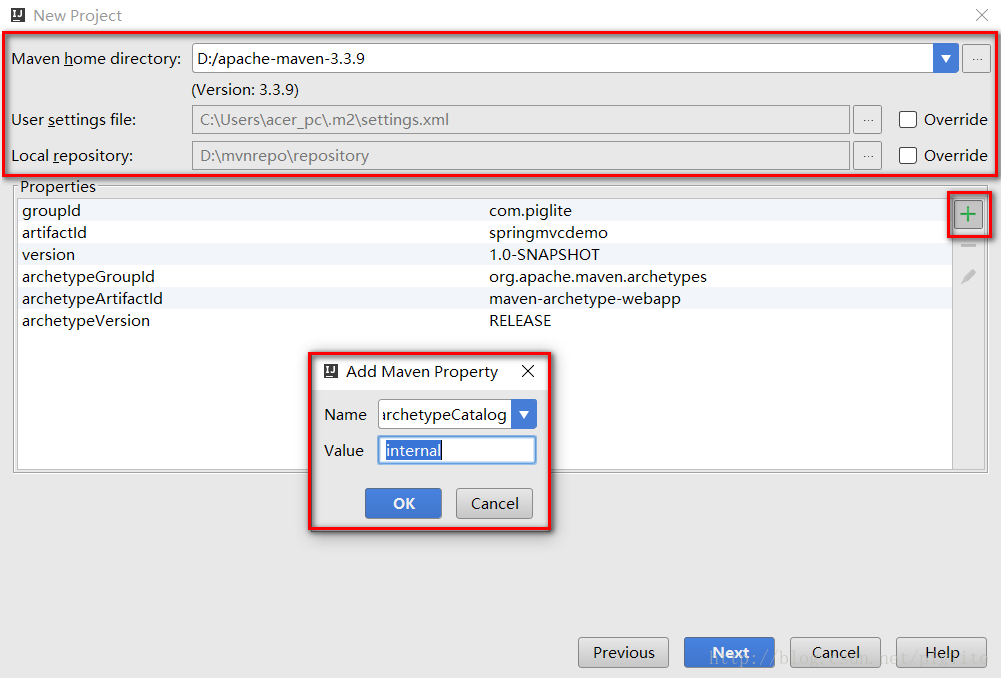
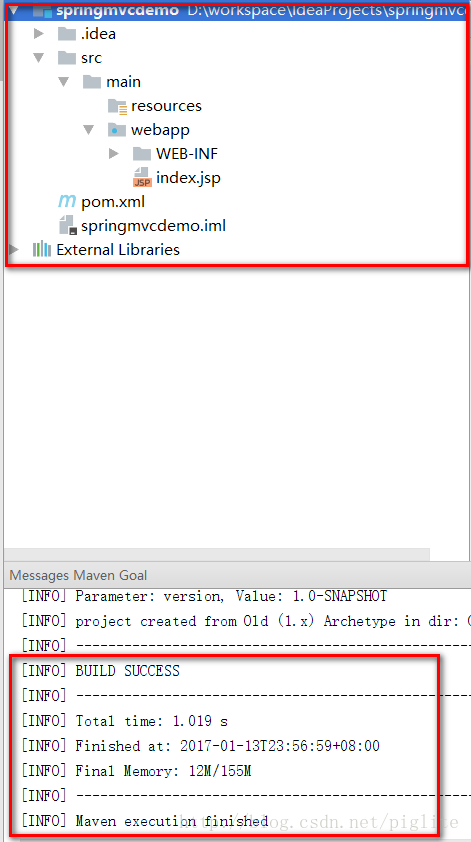
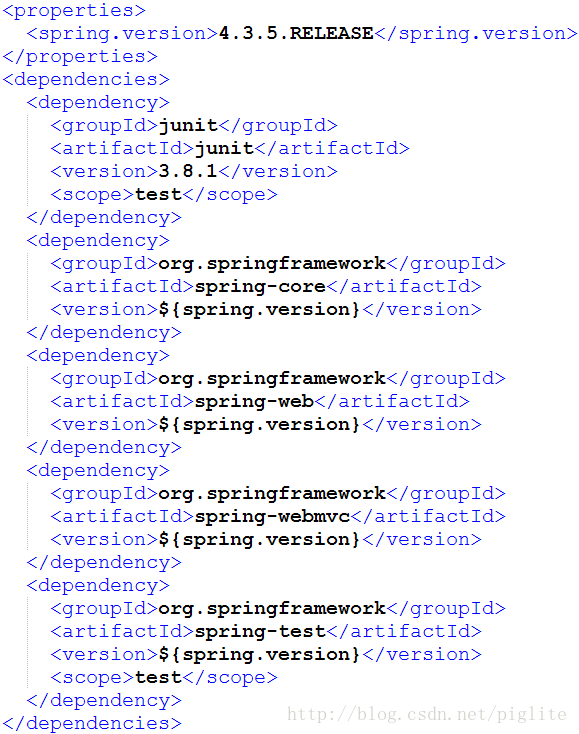
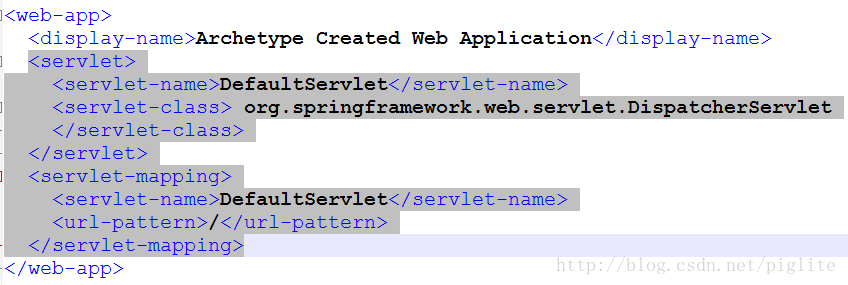
<url>http://repository.jboss.org/nexus/content/groups/pubic</url>

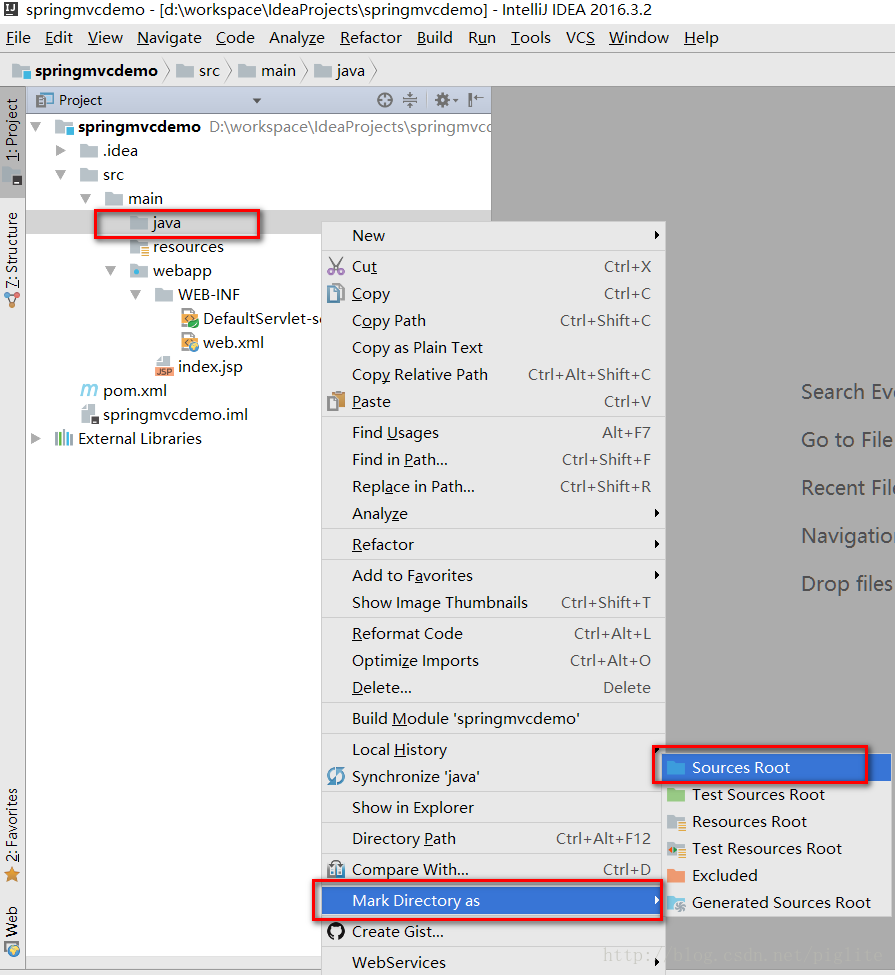
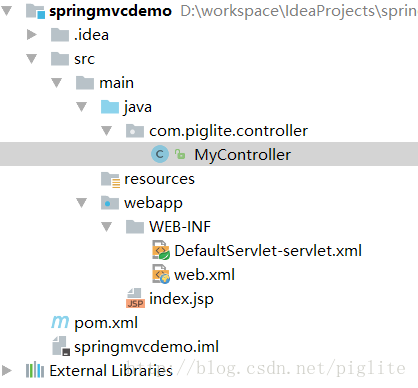
<mirrorOf>central</mirrorOf>

</mirror>

</mirrors>

鉴于网络的实际情况，可能下载速度会比较慢，而且也可能不会一次就成功。

1. 开启Intellij Idea做一些设置。选择File / Setting…   
   然后选择Build,Execution,Deployment下的Build Tools下的Maven进行设置   
   
2. 设置完毕后，利用Intellig Idea创建项目。   
   创建一个Maven项目，选择Create from archetype，然后选择创建maven-archetype-webapp项目   
     
   然后设置项目的group id， artifactid， version等信息   
     
   点击next后，在下一个界面再次确认一下maven的相关内容。点击右侧绿色加号按钮，然后添加一对创建项目时的键值对archetypeCatalog = internal 可以显著的加快项目的建立   
   
3. 当点击next，然后finish，就进入项目的创建阶段，可能时间会比较长，要耐心等待。   
   当必要的内容下载完毕后，控制台会有相应的提示信息。我们会获得一个具备基本Web结构的项目   
   
4. 编辑pom.xml文件，将SpringMVC需要的相关内容引入   
   编辑的内容包括建立标签，统一后面引入时涉及到的版本号   
   编辑标签，将需要的SpringMVC内容通过groupid artifactid以及version这些“坐标”信息告诉maven，由maven负责导入   
      
   pom.xml文件发生改变，会在Intellij idea的右下角出现一个提示窗口，可以选中import changes，将上述添加在中的内容真正引入进来。同样引入时间可能会比较长
5. 此时SpringMVC的基本jar文件已经都引入到了项目中。接下来要配置容器的相关内容。即配置WEB-INF下的web.xml文件。   
   

根据web.xml文件，要在WEB-INF文件夹下创建一个名字为DefaultServlet-servlet.xml的配置文件。（注意：该文件名字的命名有要求。默认时，web.xml文件中指定的是什么，则配置文件的名字就是该名字-servlet.xml。当然也可以进行额外的配置改变这种默认的文件命名方式以及文件的位置）该配置文件的简单内容如下：   
  
这里的 指定了一会儿对组件的扫描位置   
而bean标签指定了相应的视图映射规则。我们会把控制器的响应视图均放到WEB-INF的jsp路径下，且作为响应的视图的均为.jsp文件   
9. 创建控制器和一个响应视图   
在左侧的main下创建一个文件夹[**Java**](http://lib.csdn.net/base/java)。这里的java就是一个普通的目录，我们需要将其指定为存放java源文件的src型文件夹：   
  
选择后java文件夹的颜色会变成蓝色，此时它已经是src类型的文件夹。然后就可以创建包结构。根据我们刚才在DefaultServlet-servlet.xml文件中写的内容，基本包结构为com.piglite.xxx   
创建com.piglite.controller包，并创建一个MyController的class   
   
com.piglite.controller.MyController.java内容为：

@Controller

@RequestMapping("/demo")

public class MyController {

@RequestMapping("/welcome")

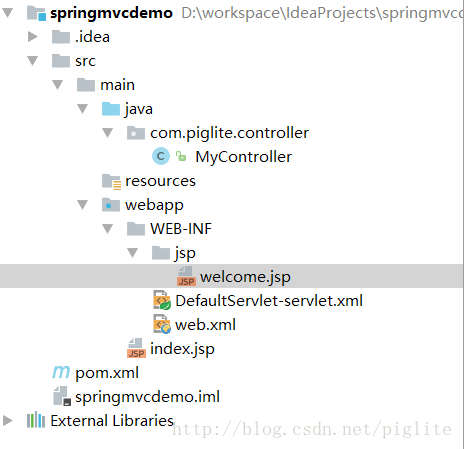
public String welcome(){

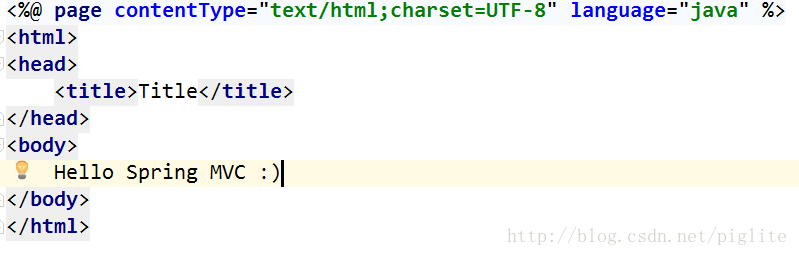
return "welcome";

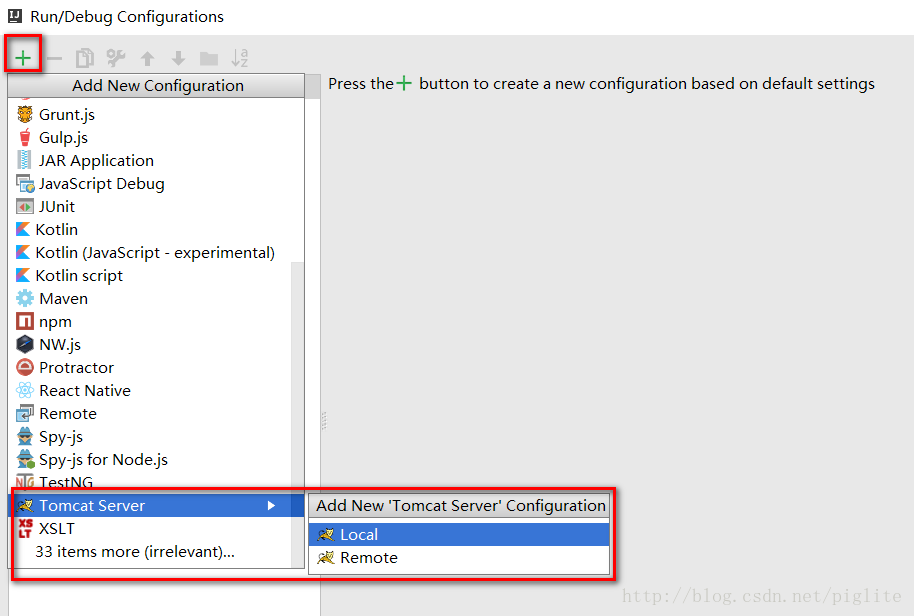
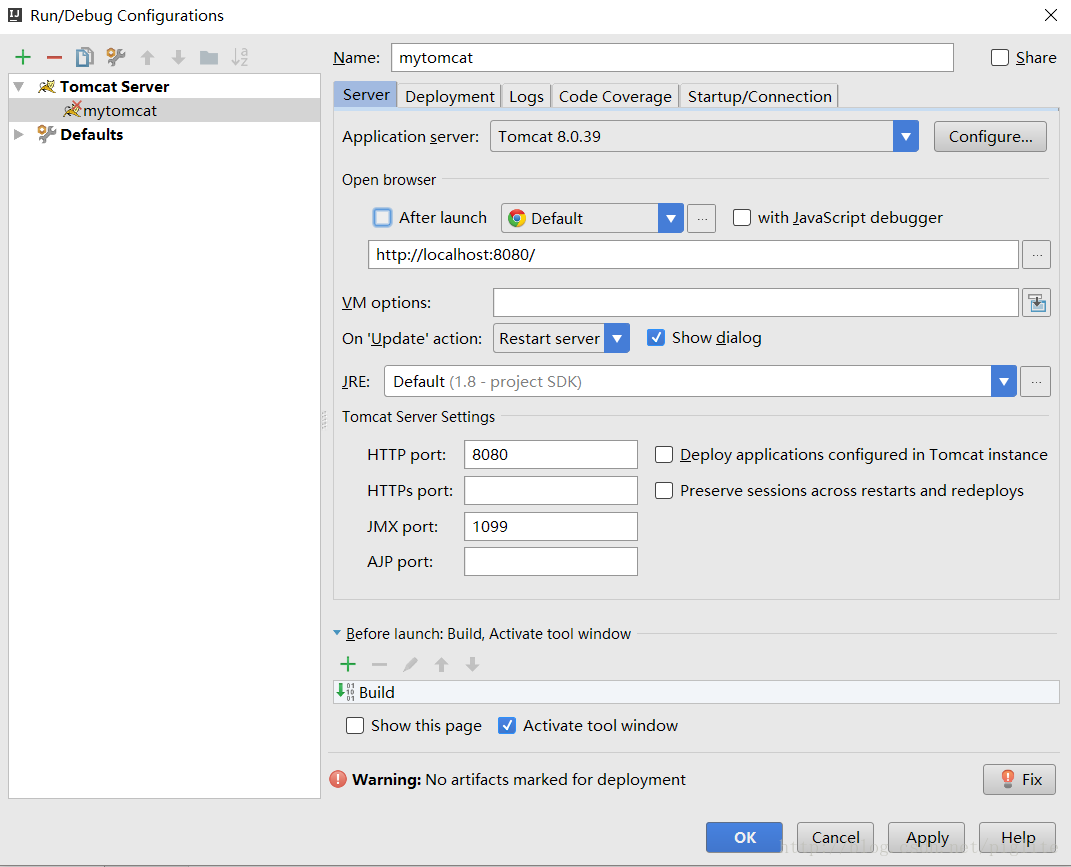
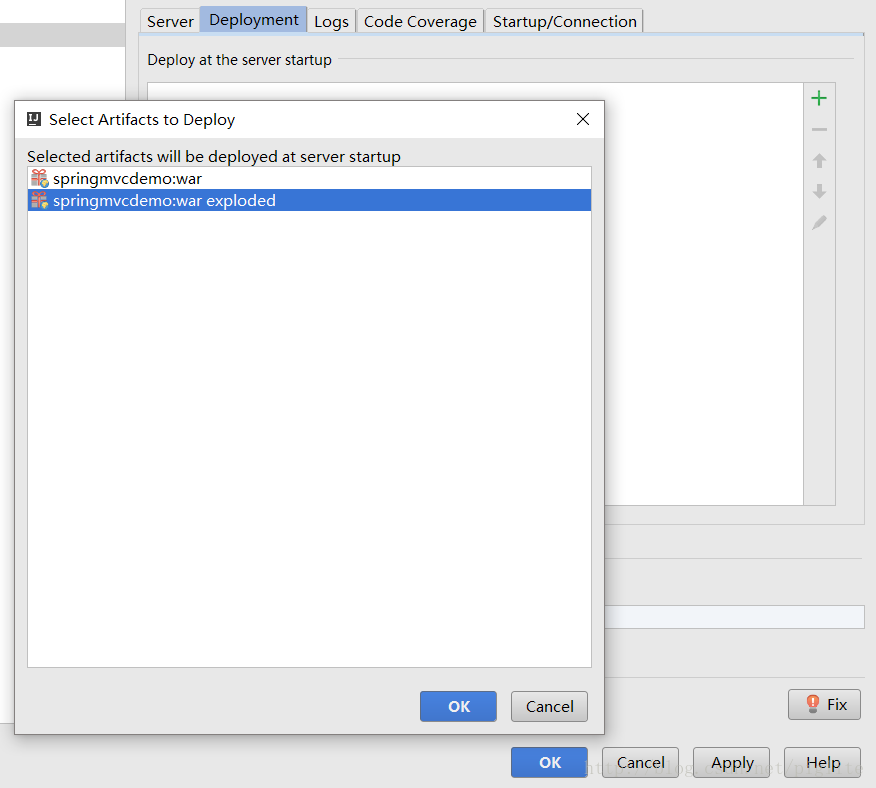
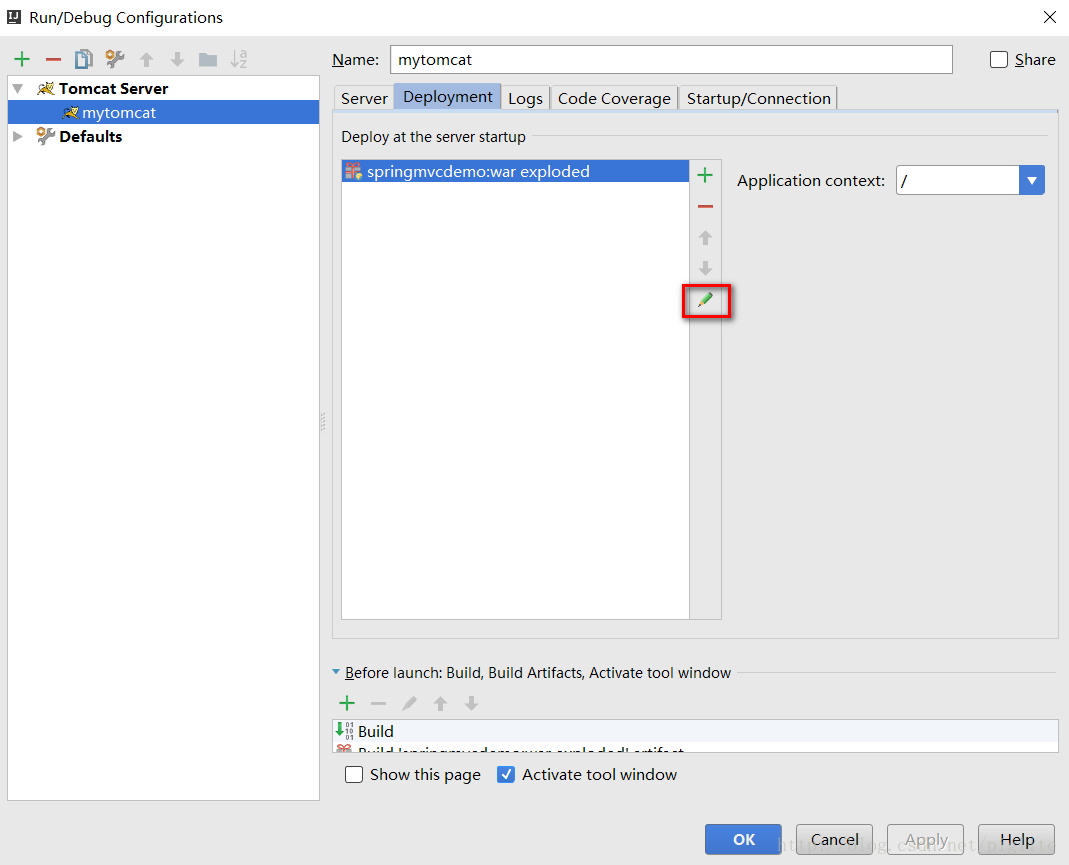
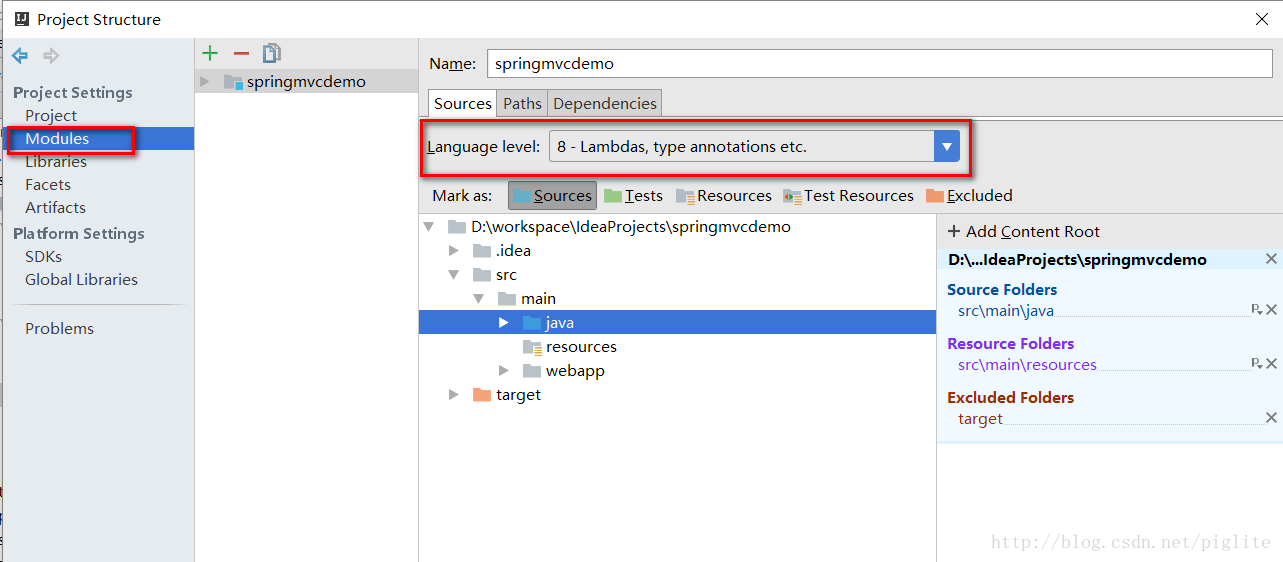
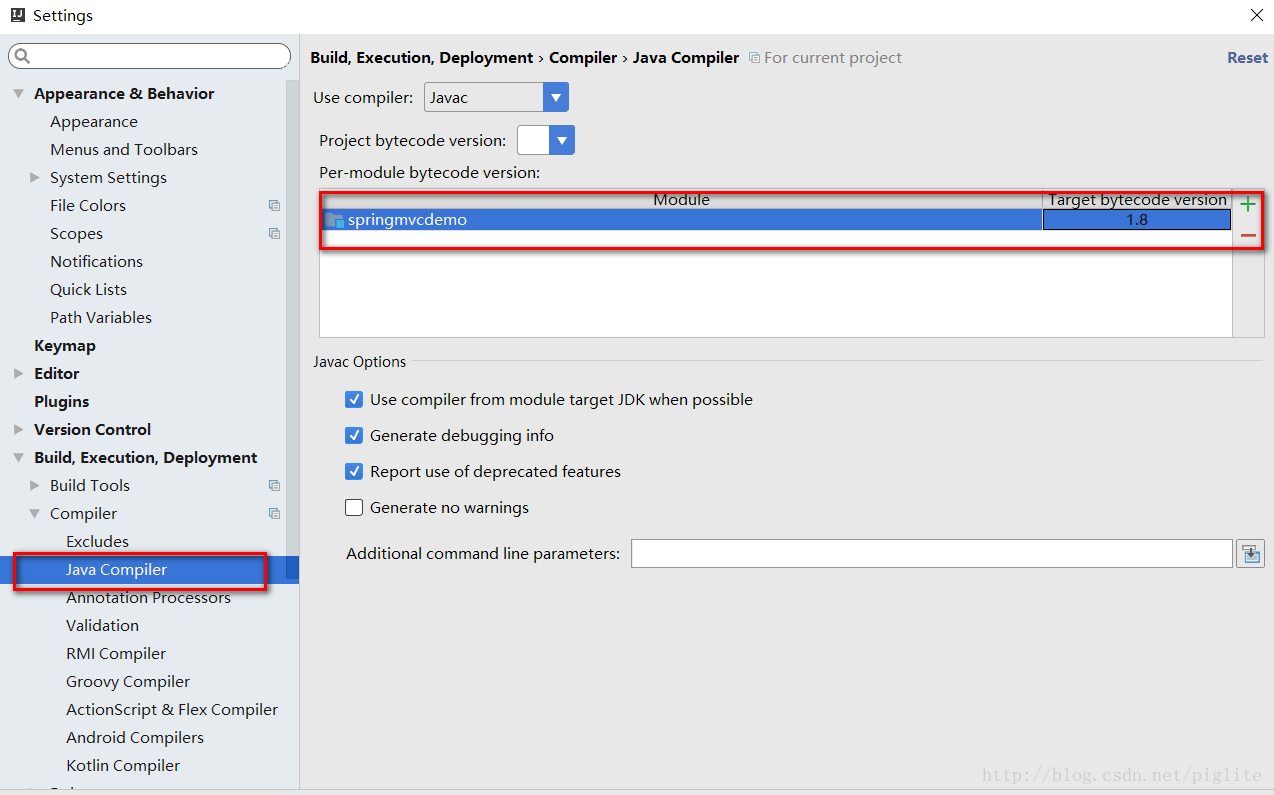
}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

如果一旦项目部署到服务器，对于host:8080/demo/welcome这样的请求，MyController会用一个名字为welcome的文件来作为响应视图。且根据DefaultServlet-servlet.xml文件中的内容，该welcome文件位于WEB-INF下的jsp文件夹，且扩展名为.jsp。   
我们接下来就创建该目录结构和文件：   




1. 至此代码部分已经完成，需要将项目部署到服务器中。   
   下载Tomcat并解压到磁盘。然后点击Untellij Idea的右上角的下拉箭头图片   
   这里写图片描述   
   在这里添加上Tomcat服务器。选择右上角的绿色加号，然后添加一个Tomcat local服务器   
     
   然后对Tomcat服务器进行一些基本的设置。包括指定一个名字，并指定Tomcat的解压路径   
     
   基本设置完成后，点击Deployment选项卡，将springmvcdemo部署到服务器。点击右侧的绿色加号按钮，然后选择artifact…然后选择springmvdemo:war exploeded，就将项目部署完成了。注意，可能点击加号后没有artifact…选项卡，就需要点击屏幕右下角的fix按钮，生成一个springmvcdemo对应的artifact。   
     
   此时部署已经完成，但是还可以做一些额外的简单设置。依然点击Department选项卡，然后点击一个铅笔按钮：   
     
   点击左侧Modules选项，然后将项目的编译环境从默认的1.5提高到1.8   
     
   此时还需要去修改另外一个地方的设置，否则在启动服务器后会报编译错误   
   点击File/Setting…/Build,Execution,Deployoment/[**Java**](http://lib.csdn.net/base/java)Compile，将项目从默认的1.5改为1.8   
   
2. 启动服务器。在服务器启动完成后，在浏览器中输入localhost:8080/demo/welcome   
   

总结：   
在利用maven配置SpringMVC环境时遇到的问题主要包括：   
1. 利用Maven下载相关内容时因为网络原因有时会时间很长，有时还不能一次成功，需要多次重试   
2. pom.xml文件中的依赖、版本、插件等内容根据需要会有非常多的设置。网上的资料根据自身项目的不同写的也都不尽相同。因此在不断的尝试下生成了一套基本的pom.xml文件。随着项目的延伸还会不但的扩展该文件。