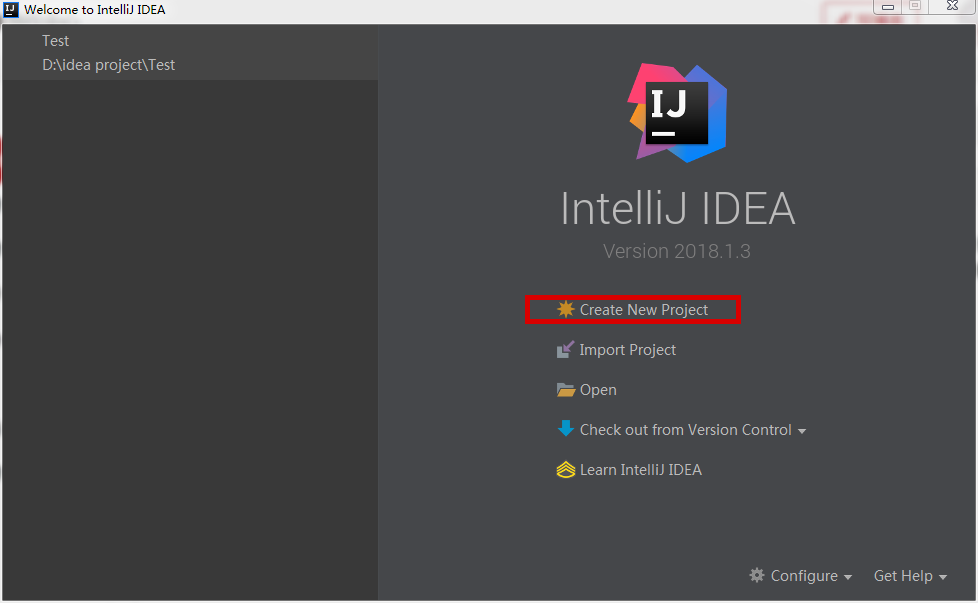
工具：Intellij IDEA （我用的是2018.1.3版本）

友情提示： 这笔记太干了，边喝水边看！！！！！！！

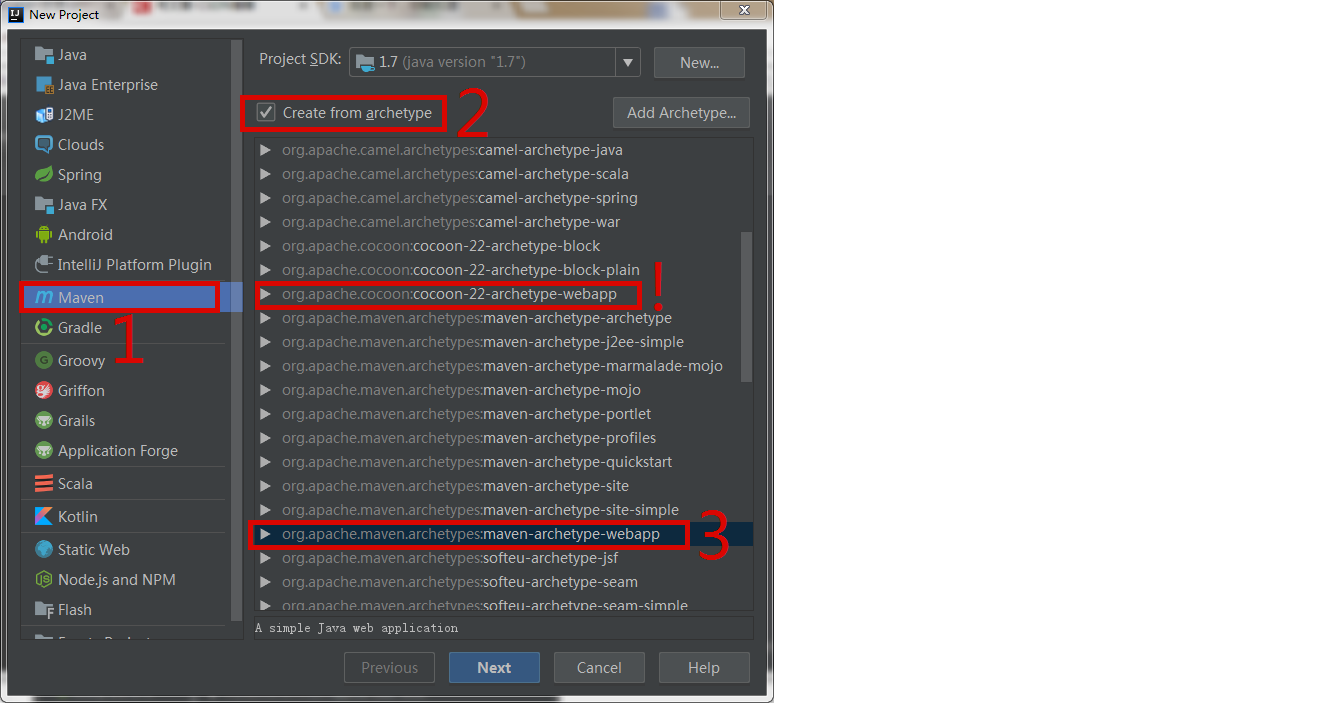
1. 打开IDEA，Create New Project



 Project SDK 不要用自带的JDK，点击New，选择自己电脑上配置好的JDK

 选择Maven，将Create from archetype勾上，选择图中的3，这里有两个webapp，不要选标感叹号的，会报错！选3选3！

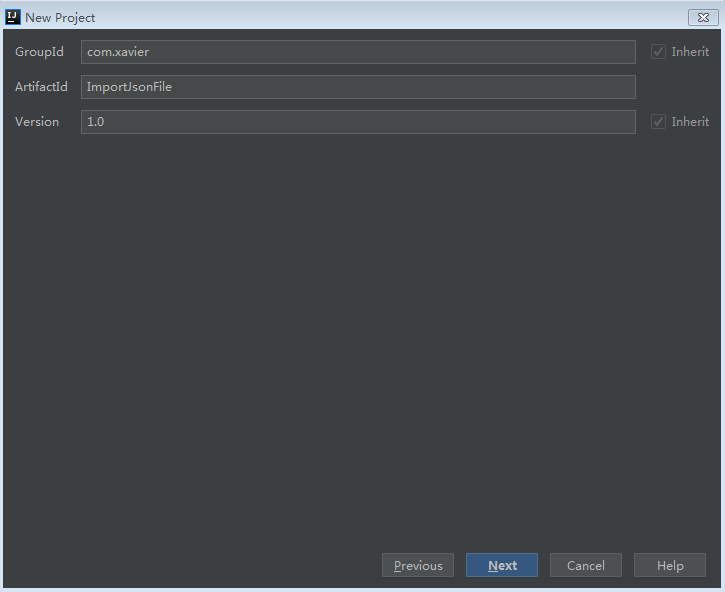
next



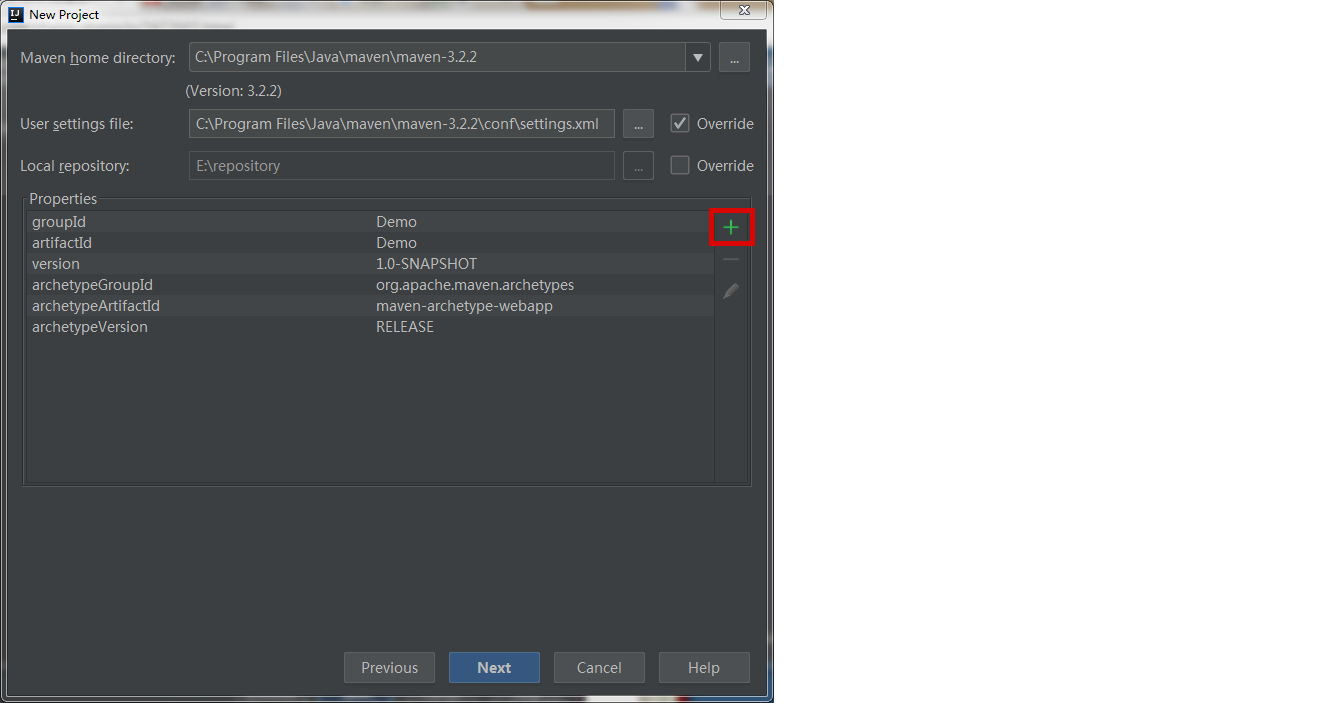
这里的GroupId和ArtifactId命名规则

GroupId:com.[name]

ArtifactId:[projectName]

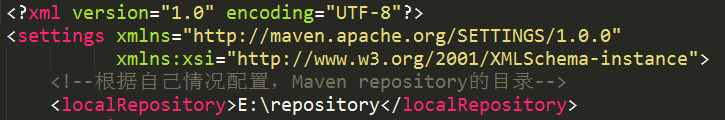


选择自己电脑上配置好的maven和settings.xml配置文件，如果settings.xml文件配置好了的话，Local repository会自动改成你配置好的仓库地址，不需要override。如果创建速度比较慢，可以点击绿色的加号添加一个属性（我没加，加不加自己看着办咯），name: archetypeCatalog , value: internal。然后next。（注意：settings.xml文件配置有几个需要注意的地方，关乎于此项目的存亡问题，有问题的参考这张图的下方，没问题的直接忽略吧）



settings.xml文件配置需要注意如下几个地方：

要将<localRepository>标签里的地址改成自己的本地仓库，这是我们自己创建的文件夹



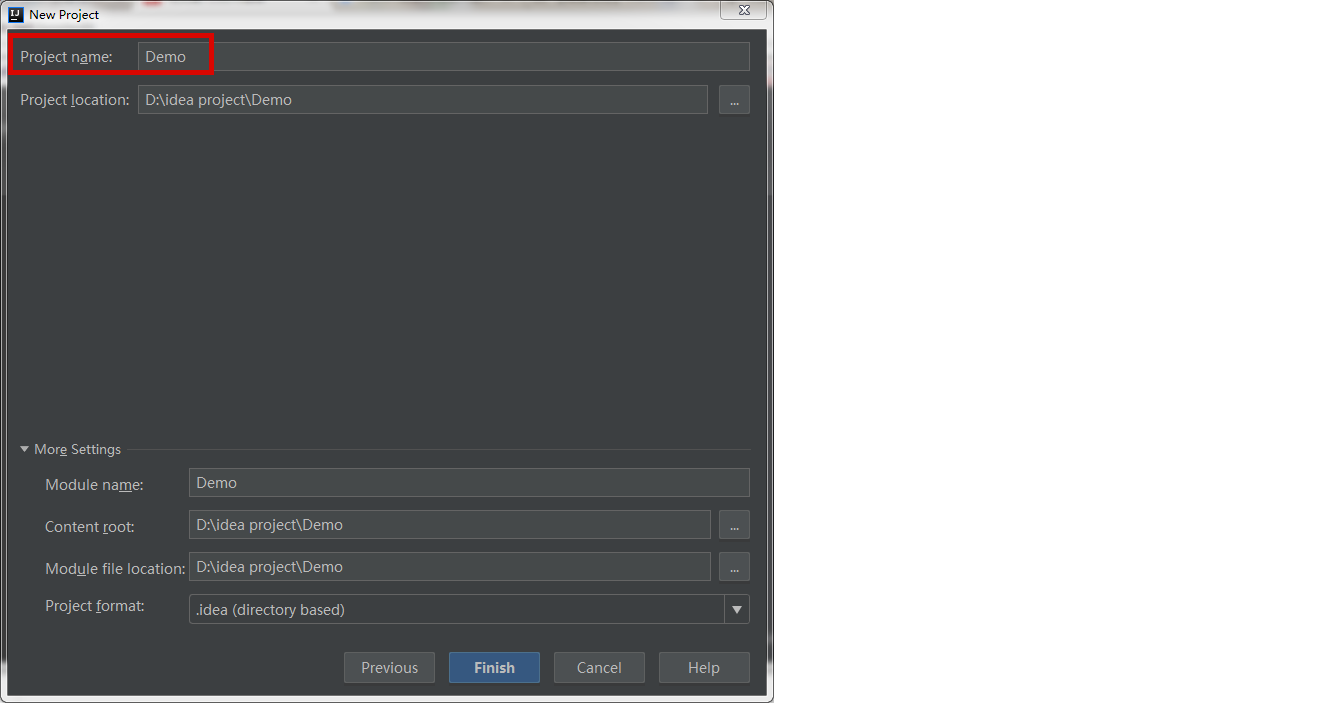
为了之后能正确得下载各种依赖包，需要加一个官方库的镜像，https://www.cnblogs.com/sunny3096/p/7510641.html 这个地址里有一些国内OSChina提供的镜像。在上图中代码下面添加一个<mirrors></mirrors>标签，然后将上面地址中给出的镜像复制到这个标签中（个数自己看着办，只要能用，一个也行）。注意注意：当时我试了很多镜像，都下载不了依赖包，后来才知道原来是公司的网设了限制，需要添加代理才行（这一步我浪费了很多时间），代码如下：



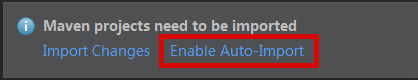
<host>和<port>分别填上地址和端口，如果不知道是多少的话也没关系，打开浏览器的设置，找到代理设置，看局域网设置中的代理服务器，就是那个地址和端口啦。

settings.xml配置结束啦

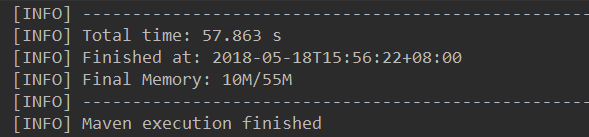
然后回到IDEA，next，这里的项目名跟前面设置的ArtifactId名一样，Project location是你的项目地址，都配置好就Finish啦



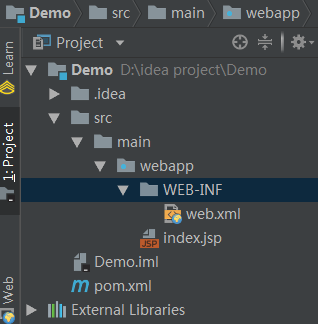
IDEA开始创建项目，这里可能要等一会儿，或许会有点慢。右下角会出现如下图所示的提示框，需要把Enable Auto-Import点上，这样maven就会开启自动导入了。



然后就等着maven创建成功吧，出现如下图所示就说明创建成功啦



2. 下图是项目创建好之后的目录



打开pom.xml文件，添加各类依赖，将下面的代码复制到<dependencies>标签中（这些是我添加的依赖，做个参考），添加完成之后，等待maven将这些jar下载完毕，下载完成后，可以看到External Library下多了很多文件

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<!--日志-->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.10</version>

</dependency>

<!--J2EE-->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>jsp-api</artifactId>

<version>2.2</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!--mysql驱动包-->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.35</version>

</dependency>

<!--springframework-->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.2.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.2.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.2.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>4.2.6.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.2.6.RELEASE</version>

</dependency>

<dependency>

<groupId>com.github.stefanbirkner</groupId>

<artifactId>system-rules</artifactId>

<version>1.16.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.8.9</version>

</dependency>

<!--其他需要的包-->

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.4</version>

</dependency>

<dependency>

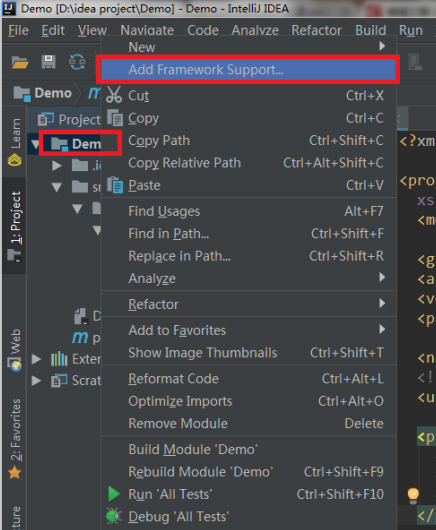
<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

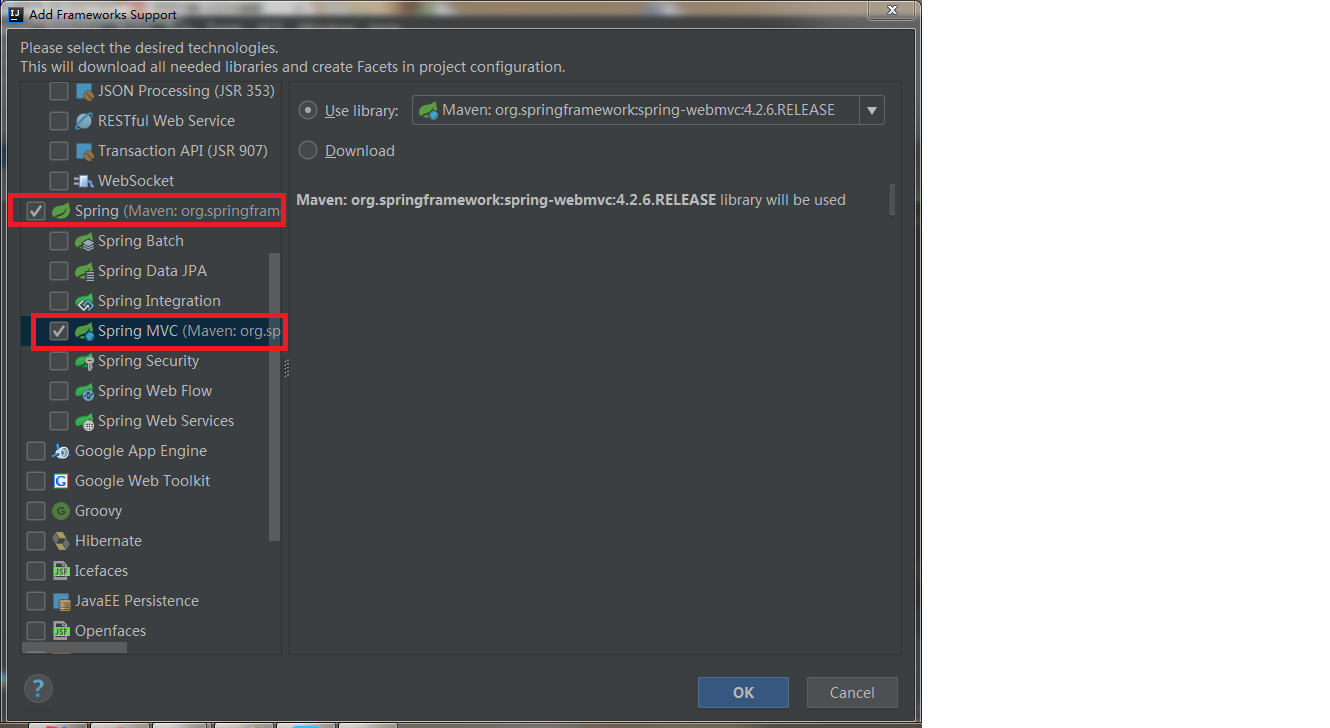
<version>1.3.1</version>

</dependency>

3. 添加springMVC框架，右击项目文件夹Demo，选择Add framework support

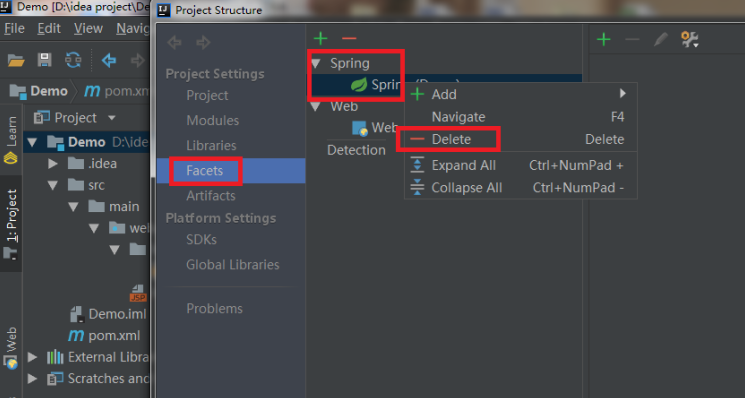


将下图中的Spring和Spring下的Spring MVC都勾上，之前配置pom.xml文件时，已经自动下载了spring相关文件，所以这里就直接用之前下载好的就可以了，OK。（注意：点了Add framework support之后，在下图中有可能会找不到Spring，解决办法在下图的下方）

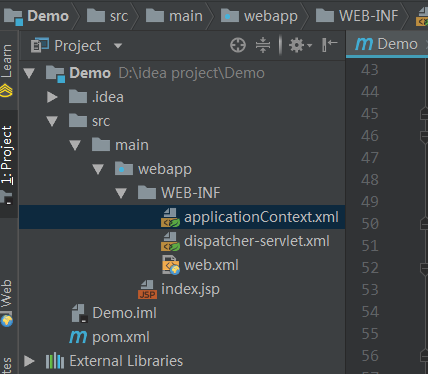


如果在Add framework support中找不到Spring，那是因为项目中可能已经存在Spring相关文件，但不一定是完善的。因此我们要将已经存在的Spring给删掉，重新添加，方法如下：

点击File，选择Project Structure，选择Facets，就会看到有一个Spring啦，右击它，点删除就行啦，然后再回到上面第3步重新Add framework support，Spring就会出现啦。



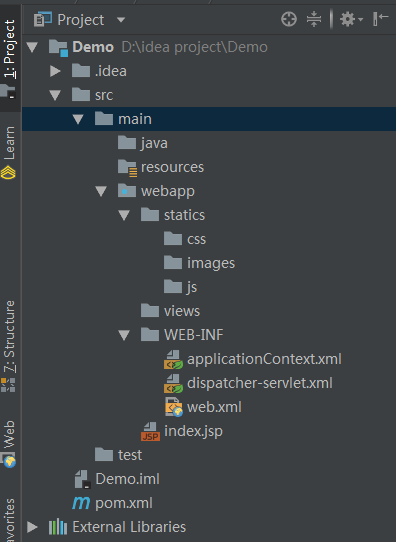
Spring框架添加完之后，会看到目录下多了两个xml文件，之后需要在里面配置的，暂时不管。



4. 到目前为止，我们的目录结构还不完善，先把结构完善一下。

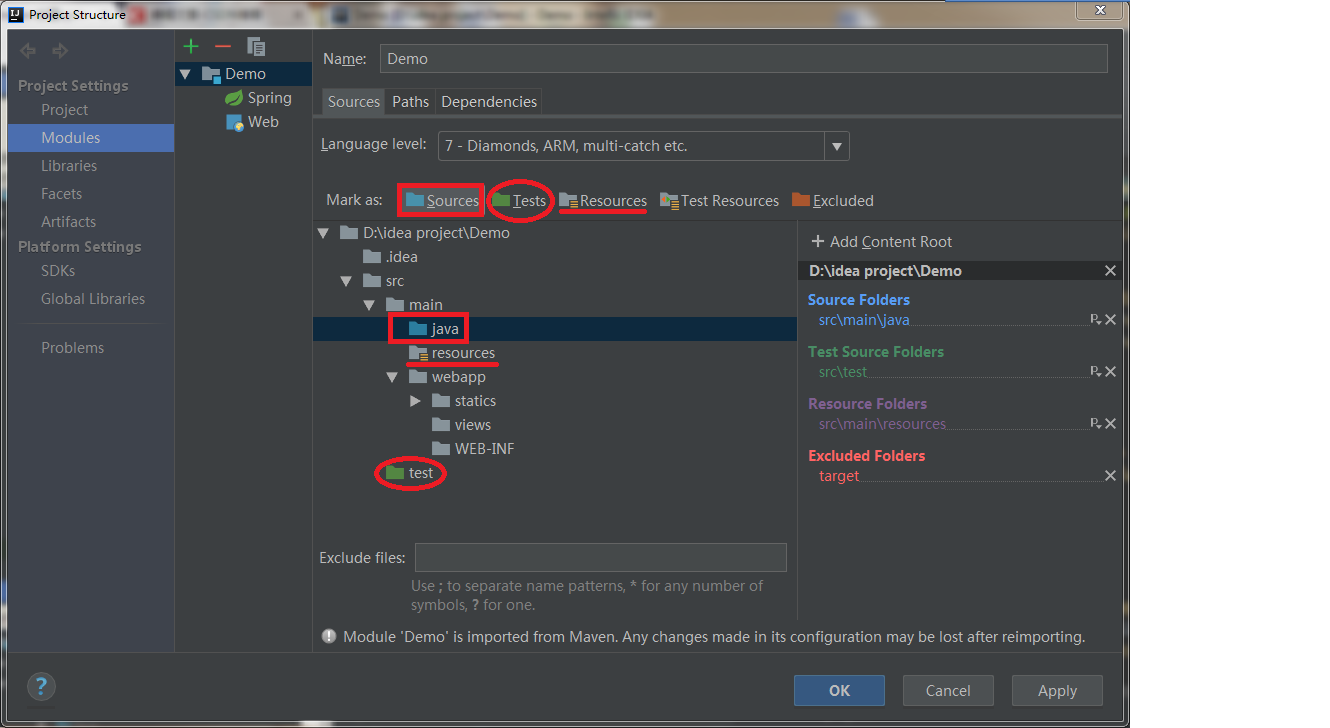
创建java、resources、test文件夹。

    先在src/main文件夹下创建java文件夹，这个文件夹暂时不能创建java的类文件和package，不过没关系，之后再来设置，resources是和java一样的文件夹，建好之后会自动变成有黄色长方形的样子，如果建好之后没有黄色的长方形，这个也之后再设置，先把需要的文件夹都创建好，如下图所示：（test文件夹是src下的）

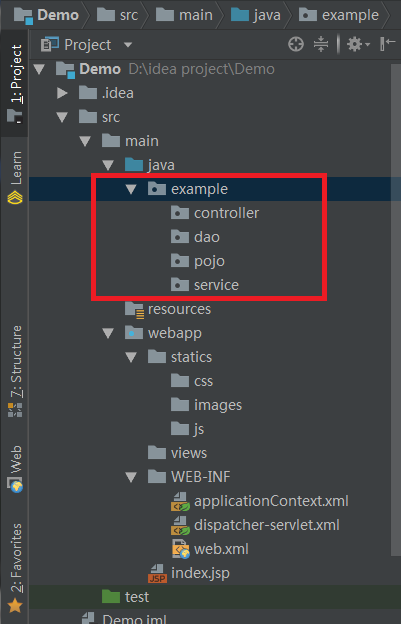


创建好之后，File-->Project Structure或者Ctrl+Alt+Shift+S，选择Modules，给java，resources，test文件夹标记。

单击选中java文件夹，再单击Sources，resources文件夹对应Resources，test文件夹对应Tests，OK，这时候被标记的文件夹就变色了，说明标记成功。



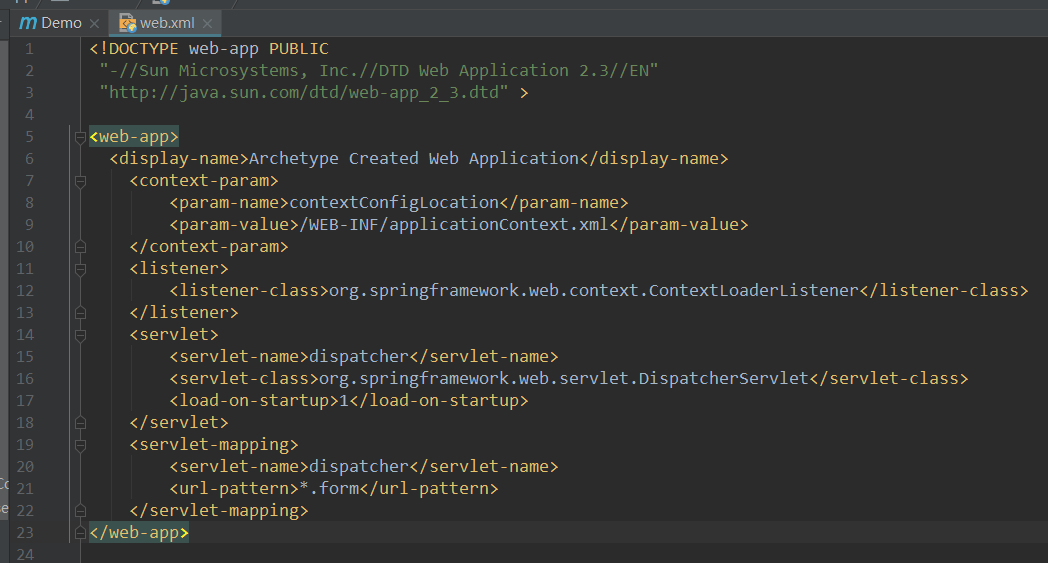
接下来在java文件夹里建立需要的包，结构如下:



上图里webapp下面也要建立包!!!!!!!!

5. 现在可以对SpringMVC进行设置了，首先配置web.xml

下图是web.xml自动生成的配置，但是这个版本比较低，我们来换个新的



新配置代码如下，直接复制就好（全替换掉）

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"

version="3.1">

<display-name>Archetype Created Web Application</display-name>

<!--welcome pages-->

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

<!--配置springmvc DispatcherServlet-->

<servlet>

<servlet-name>springMVC</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<!--配置dispatcher.xml作为mvc的配置文件-->

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/dispatcher-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

<async-supported>true</async-supported>

</servlet>

<servlet-mapping>

<servlet-name>springMVC</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<!--把applicationContext.xml加入到配置文件中-->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/applicationContext.xml</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

</web-app>

6. 配置dispatcher-servlet.xml，负责mvc的配置（把原先代码全部替换成下面代码，直接复制粘贴！！！！！）

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd">

<!--此文件负责整个mvc中的配置-->

<!--启用spring的一些annotation -->

<context:annotation-config/>

<!-- 配置注解驱动 可以将request参数与绑定到controller参数上 -->

<mvc:annotation-driven/>

<!--静态资源映射-->

<!--本项目把静态资源放在了webapp的statics目录下，资源映射如下-->

<mvc:resources mapping="/css/\*\*" location="/statics/css/"/>

<mvc:resources mapping="/js/\*\*" location="/statics/js/"/>

<mvc:resources mapping="/image/\*\*" location="/statics/images/"/>

<mvc:default-servlet-handler /> <!--这句要加上，要不然可能会访问不到静态资源，具体作用自行百度-->

<!-- 对模型视图名称的解析，即在模型视图名称添加前后缀(如果最后一个还是表示文件夹,则最后的斜杠不要漏了) 使用JSP-->

<!-- 默认的视图解析器 在上边的解析错误时使用 (默认使用html)- -->

<bean id="defaultViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>

<property name="prefix" value="/views/"/><!--设置JSP文件的目录位置-->

<property name="suffix" value=".jsp"/>

<property name="exposeContextBeansAsAttributes" value="true"/>

</bean>

<!-- 自动扫描装配 -->

<context:component-scan base-package="example.controller"/>

</beans>

7. 配置applicationContext.xml，负责一些非MVC组件的配置，暂时没有所以是空的，但也可以扫描一下（也是！全部替换掉！！！直接复制下面的代码，粘就完了铁子！）

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">

<context:component-scan base-package="example"/>

</beans>

8. 以上全都配置好，就可以开始测试了，来试一个很简单的例子

在controller包里新建一个DemoController.java，代码如下：

package example.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

@Controller

@RequestMapping("/home")

public class DemoController {

@RequestMapping("/index")

public String index(){

return "index";

}

}

9，在views文件夹下新建一个index.jsp，statics/css下新建一个index.css（建的时候名字后缀也要加上，.jsp不要落下，.css也是！！！！！！）

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Index</title>

<link rel="stylesheet" type="text/css" href="../statics/css/index.css">

</head>

<body>

<h2>Test success!</h2>

</body>

</html>

h2{

color: darkred;

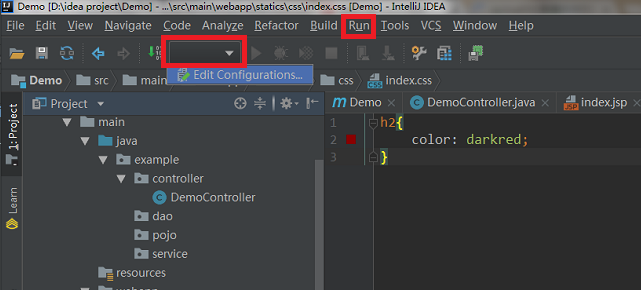
}

提示，上面这部分代码是自己写在index.jsp里的，如果不想写这个代码idea会在index.jsp里自动生成一个helloword的代码 ， 代码如下（这两段代码写哪个都行，就是网页里最终会出现的字，当然，程序猿第一句必须是helloword，哈哈哈哈哈哈）

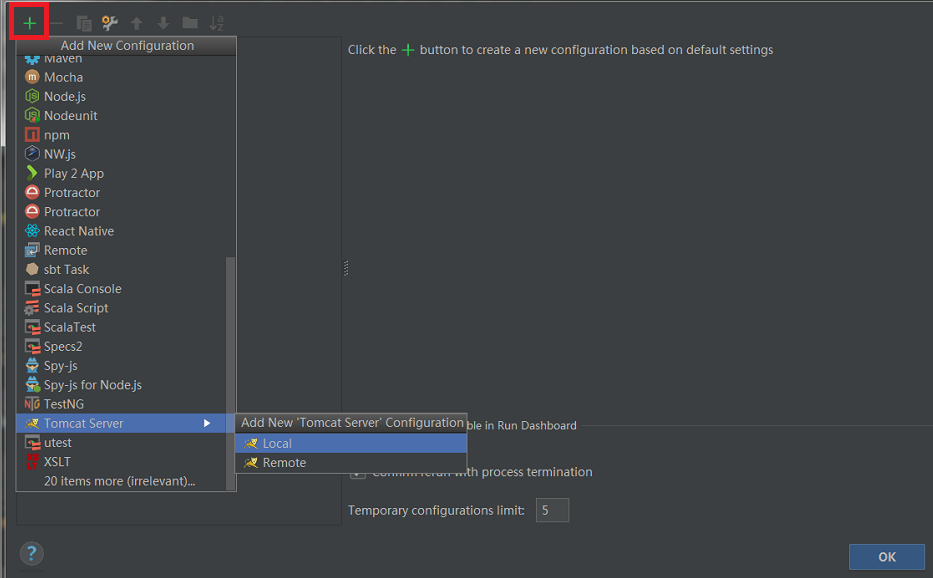
<**html**>  
<**body**>  
<**h2**>Hello World!</**h2**>  
</**body**>  
</**html**>

10， 接下来该配置Tomcat了，最好用Tomcat 7以上的版本

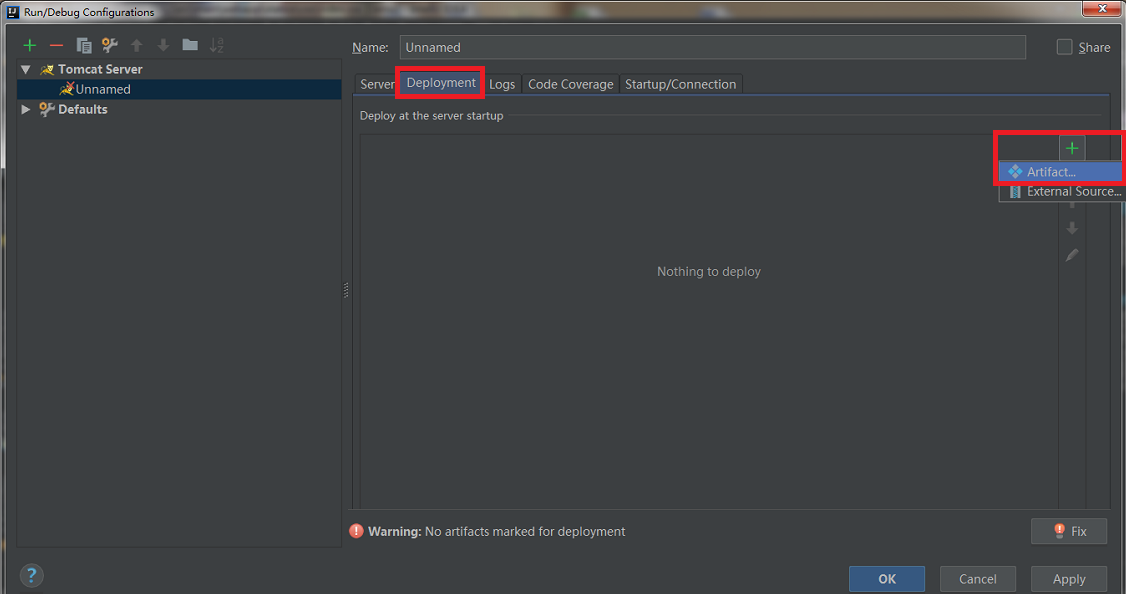
点击这个下拉框或者点击Run，选择Edit Configurations



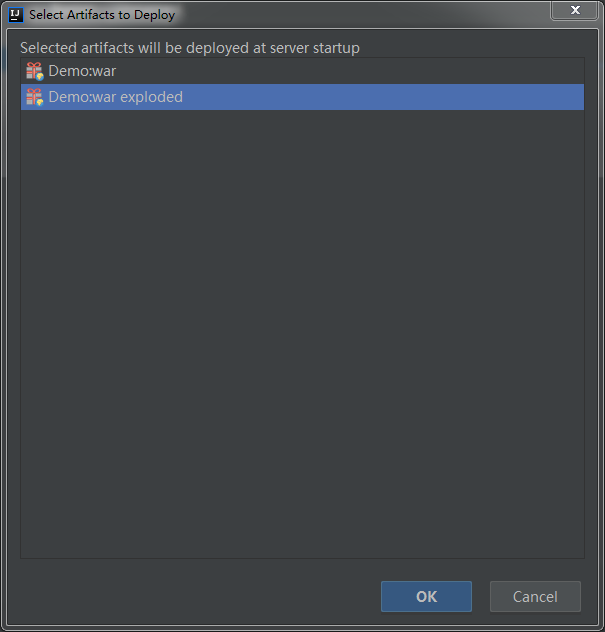
点击绿色加号选择Tomcat server --> Local



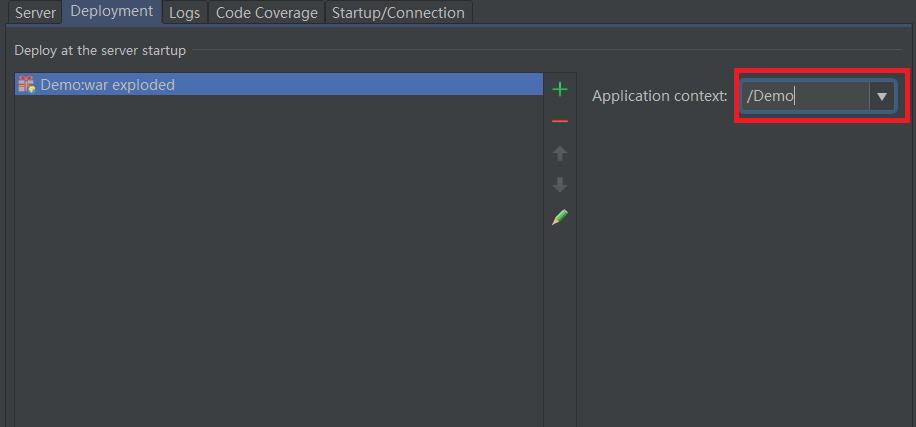
我们先配置Deployment吧，点击绿色加号，选择Artifact...



选择第二个，一定要选择war exploded，要不然会报错，OK

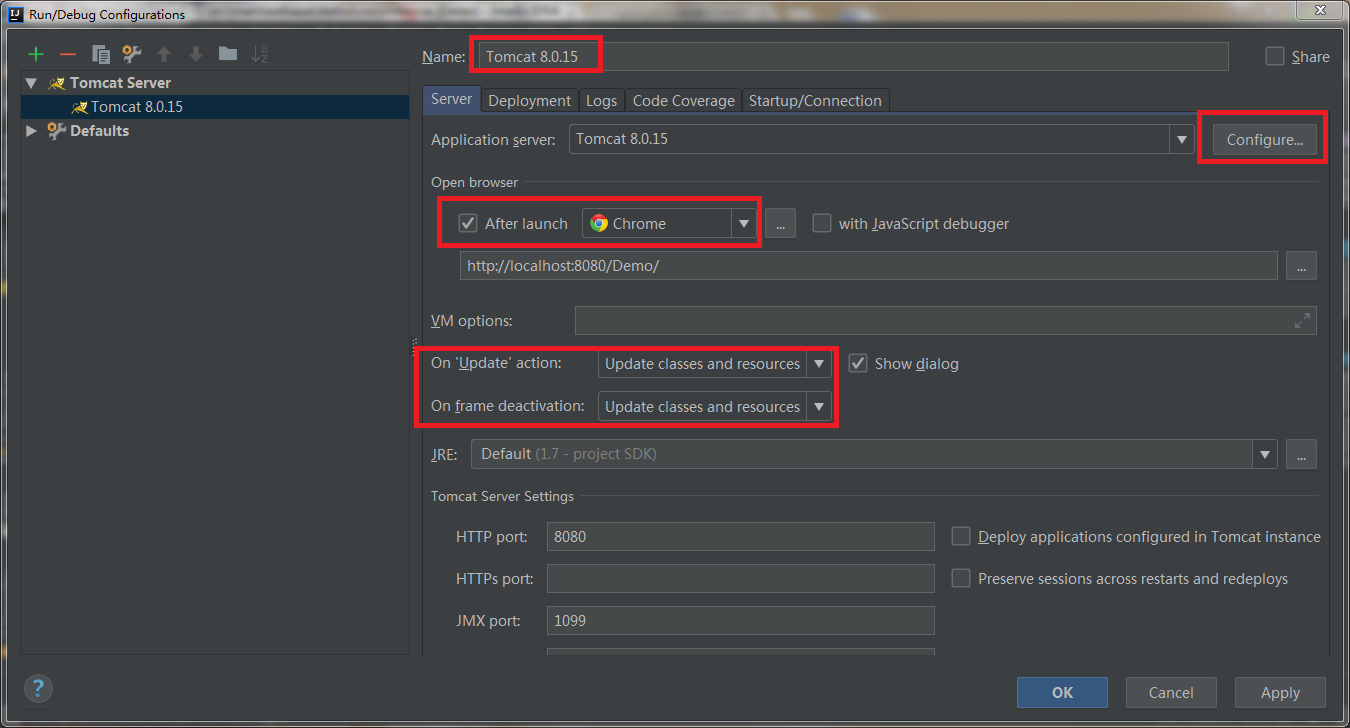


这里的名字和项目名一样，这里也就是Demo



然后来设置Server，可以给Tomcat取个名字，Configure 选择自己安装好的Tomcat，根据个人习惯设置一个默认的浏览器。

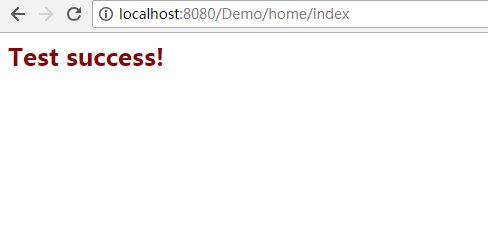
然后On 'Update' action 和 On frame deactivation两项都选择Update classes and resources，这是为了当我们修改了jsp、css、js等静态资源的时候，立即生效，不需要重启Tomcat，直接在页面上刷新就可以看到效果。之前之所以先配置Deployment，是因为要先配置里面的Artifact，这两项要依赖于exploded，要不然Update classes and resources是没有的，只是为了方便而已啦。设置完成后，OK



一定在上图要选JRE的路径!!!!!!!!!!!!!!!

11. 运行Tomcat试试吧，在地址栏输入 <http://localhost:8080/Demo/home/index>

或者直接在idea里启动项目也可以！



这是第九步里上面那段代码的图片 ，选择下面helloword的代码，最后图片里的字就是helloword！！！！