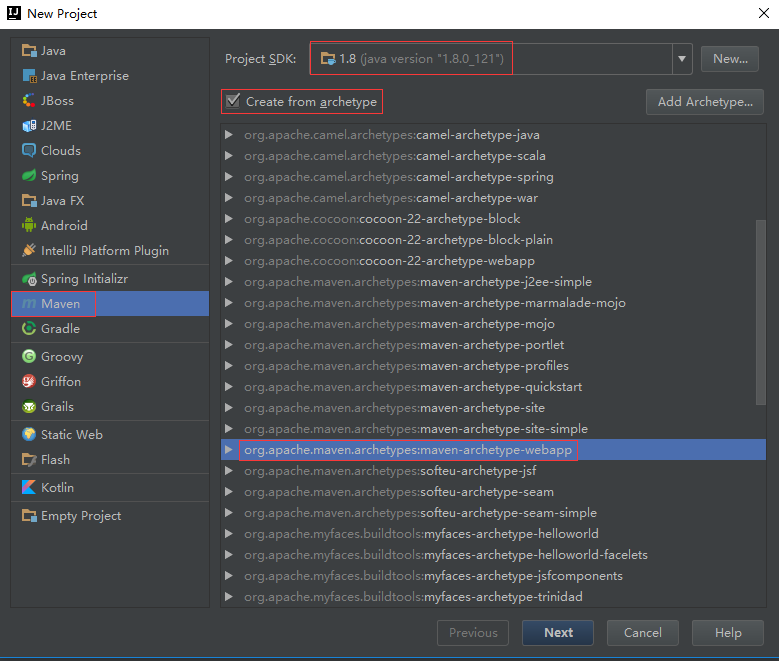
<http://www.cnblogs.com/winner-0715/p/5294917.html>

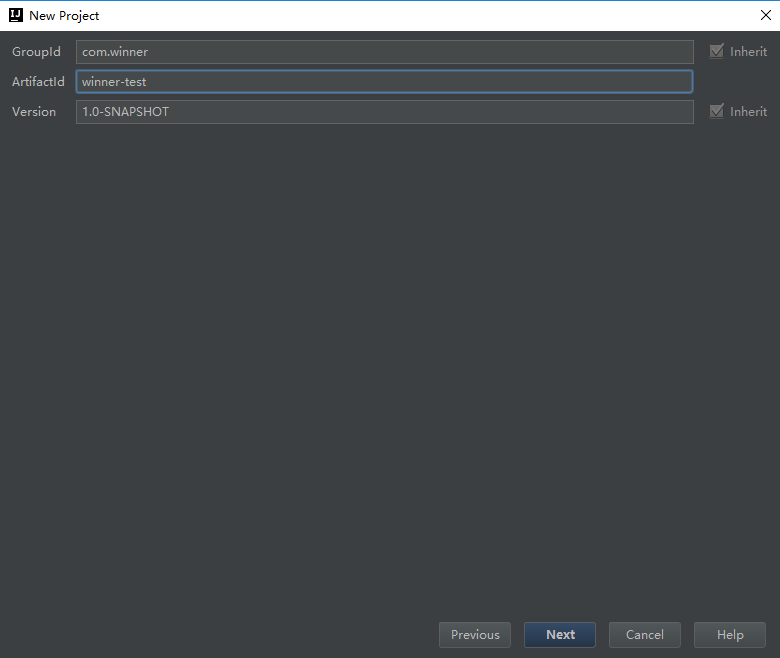
(本示例使用的IntelliJ idea版本是16，其实各个版本大同小异~)

**创建Maven Web项目**

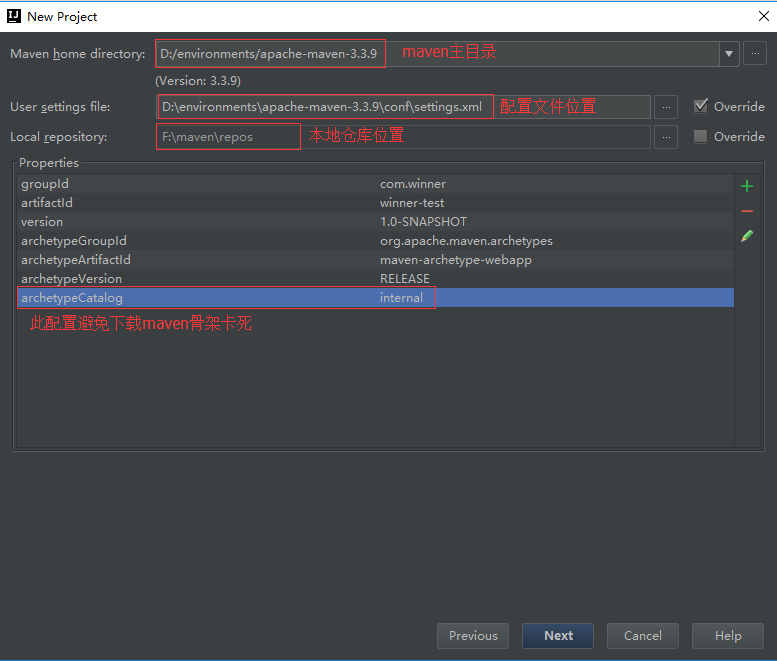
菜单File->New Project可进入如图界面，首先选择左边栏Maven，再配置JDK(如果之前添加了JDK的话会自动填充，如未添加点击旁边的New将JDK目录导入即可)。勾选"Create from archetype"，然后选中maven-archetype-webapp，点Next，进入如下界面：



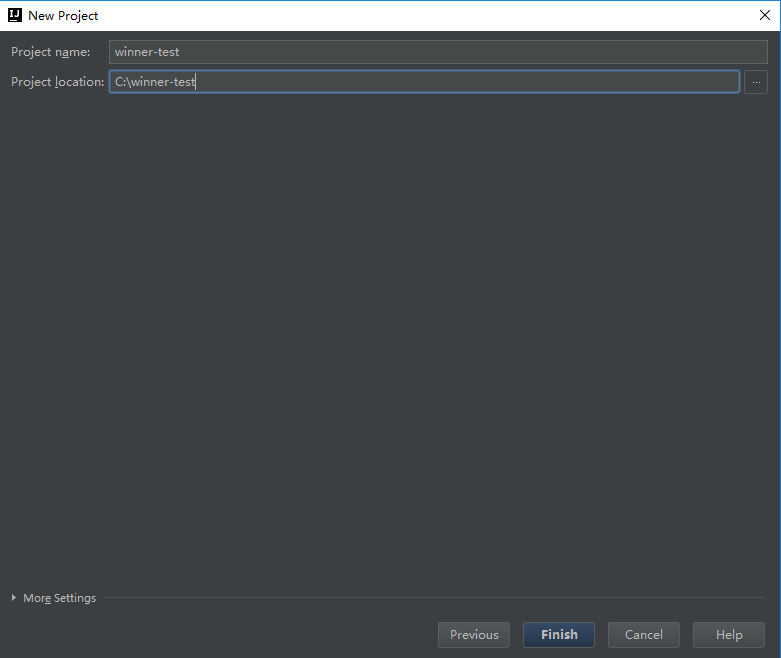
这里需要填写GroupId和ArtifactId,Version默认即可，这三个属性可以唯一标识你的项目。



我自己的maven配置

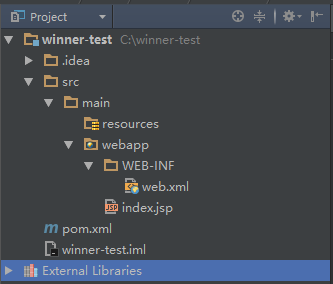


填写项目名，选择项目保存路径，点击Finish：



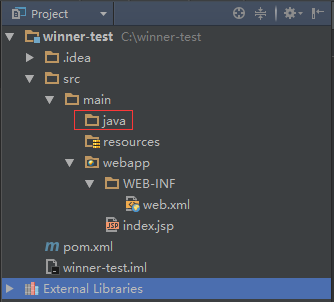
maven会在后台生成web项目，这需要等待一定的时间，视网络环境而定.

下图展示了该项目的文件结构。可以发现，它在src/main下创建了一个recources文件夹，该文件夹一般用来存放一些资源文件，还有一个webapp文件夹，用来存放web配置文件以及jsp页面等，这已经组成了一个原始的web应用。选择右边红框的Enable-Auto- Import，可以在每次修改pom.xml后，自动的下载并导入jar包。

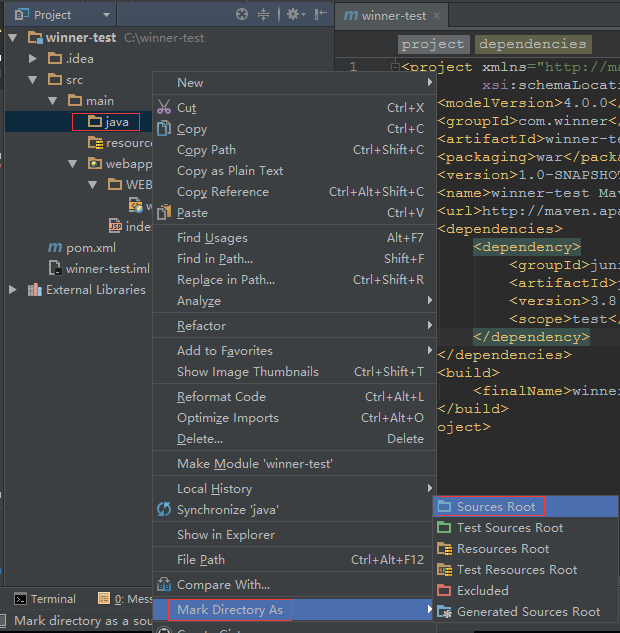


我们可以看到，目录结构并不是严格的maven格式,因为少了java源码文件夹

首先在main文件夹下创建一个文件夹，名称为java



将java文件夹标识为Source Root



可以看到文件夹的颜色变了，设置完成~

**Maven自动导入jar包**  
我们使用maven管理依赖~  
Maven所做的工作就是自动把你需要的jar包下载到本地，然后关联到项目中来。maven的所有jar包都是保存在几个中央仓库里面的，其中一个最常用的是[Maven Repository](http://mvnrepository.com/" \o "http://mvnrepository.com/" \t "_blank)，即，你需要什么jar包，它就会从仓库中拿给你。那么如何告诉maven需要什么jar包呢？我们看看工程目录，能找到一个pom.xml文件（这个文件在刚创建好项目时就已经展现在了大家面前），maven就是靠它来定义需求的：

[复制代码](javascript:void(0);)

<project xmlns="http://maven.apache.org/POM/4.0.0"

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

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.winner</groupId>

<artifactId>winner-test</artifactId>

<packaging>war</packaging>

<version>1.0-SNAPSHOT</version>

<name>winner-test Maven Webapp</name>

<url>http://maven.apache.org</url>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>winner-test</finalName>

</build>

</project>

[复制代码](javascript:void(0);)

我们可以看到这个文件包含了我们之前定义的本项目的groupId等信息，这些信息是该项目的标识，我们不要去改动它们。重点看<dependencies>标签，翻译过来是"依赖"的意思，也就是说把对每个包的需求都称为一个依赖<depedency>，定义在<dependencies>中。在每个<depedency>中，你需要提供的是所需jar包的groupId、artifactId、version(g,a,v)这三个必要信息(坐标)。比如上面我们看到引入可一个junit包，格式如下：

[复制代码](javascript:void(0);)

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

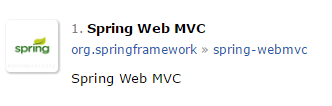
<scope>test</scope>

</dependency>

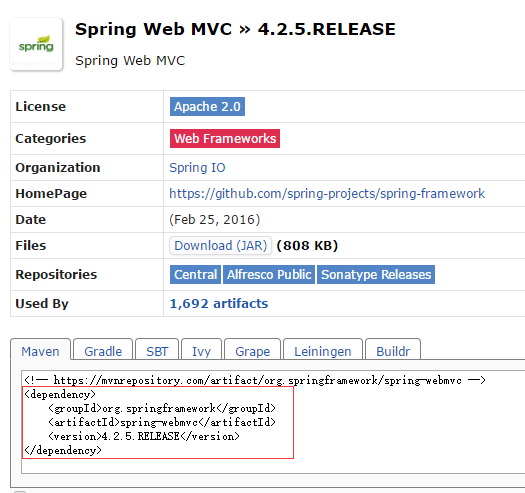
</dependencies>

[复制代码](javascript:void(0);)

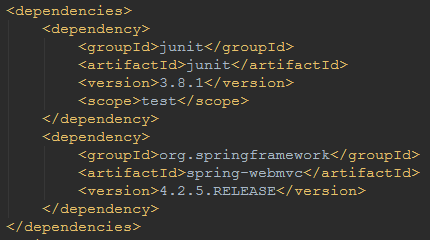
这是单元测试包，提供了三个基本信息，第4个scope对其他包来说是非必需的。所有jar包的引入都要满足这个格式。要查看这些jar包的3个信息呢，需要查阅仓库。比如我们需要引入spring-webmvc包，打开**[Maven Repository](http://mvnrepository.com/" \t "_blank)，**搜索spring-webmvc，进入如下界面：



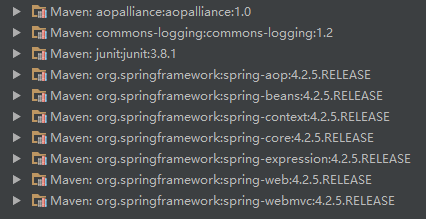
选择最新版本4.2.5.RELEASE，可以看到其dependency写法如下红框所示：



我们将其复制到pom.xml中的<dependencies>中：



这样，Maven就会开始自动下载jar包到本地仓库，然后关联到你的项目中，下载完成后，我们展开工程目录中External Libraries：



可以发现，虽然我们只写了一个依赖，但是会把与它密切相关的jar包同时导入进来。  
自此演示一个简单的maven项目所需的jar包其实已经够了~

**SpringMVC框架配置**  
现在进行项目的配置  
1、web.xml配置，默认情况下生成的web.xml的配置如下：

[复制代码](javascript:void(0);)

<!DOCTYPE web-app PUBLIC

"-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"

"http://java.sun.com/dtd/web-app\_2\_3.dtd" >

<web-app>

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

</web-app>

[复制代码](javascript:void(0);)

这并是不我们需要的，暂未找到更改生成web.xml配置的方法,知道的麻烦告知一下，谢谢~

--->

[复制代码](javascript:void(0);)

<?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>winner-test Web Application</display-name>

</web-app>

[复制代码](javascript:void(0);)

在web.xml中创建Spring MVC的中央控制器

[复制代码](javascript:void(0);)

<?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>winner-test Web Application</display-name>

<servlet>

<servlet-name>mvc-dispatcher</servlet-name>

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

<init-param>

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

<param-value>classpath:mvc-dispatcher.xml</param-value>

</init-param>

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

</servlet>

<servlet-mapping>

<servlet-name>mvc-dispatcher</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

</web-app>

[复制代码](javascript:void(0);)

该servlet名为mvc-dispatcher(名称可修改)，用于拦截请求（url-pattern为\*.do），并交由Spring MVC的后台控制器来处理。这一项配置是必须的

为了能够处理中文的post请求，再配置一个encodingFilter，以避免post请求中文出现乱码情况：

[复制代码](javascript:void(0);)

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<init-param>

<param-name>forceEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

[复制代码](javascript:void(0);)

至此，web.xml配置完毕。

2、mvc-dispatcher.xml配置  
在配置完web.xml后，在resources文件夹下创建spring配置文件mvc-dispatcher.xml：

mvc-dispatcher.xml文件如下：

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

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

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

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

</beans>

代码如下：

[复制代码](javascript:void(0);)

@Controller

public class MainController {

@RequestMapping(value = "/test.do",method = RequestMethod.GET)

public String index() {

return "index";

}

}

[复制代码](javascript:void(0);)

--@Controller注解：采用注解的方式，可以明确地定义该类为处理请求的Controller类；  
--@RequestMapping()注解：用于定义一个请求映射，value为请求的url；  
--return "index"：处理完该请求后返回的逻辑视图。  
回到mvc-dispatcher.xml，进行相关配置。

[复制代码](javascript:void(0);)

<?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:p="http://www.springframework.org/schema/p"

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/mvc

http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd

http://www.springframework.org/schema/context

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

<!-- 配置包扫描器 -->

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

<!-- 配置注解驱动 -->

<mvc:annotation-driven/>

<!-- 视图解析器 -->

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

<property name="prefix" value="/WEB-INF/views/"/>

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

</bean>

</beans>

[复制代码](javascript:void(0);)

现在，需要配置Tomcat来运行该项目。

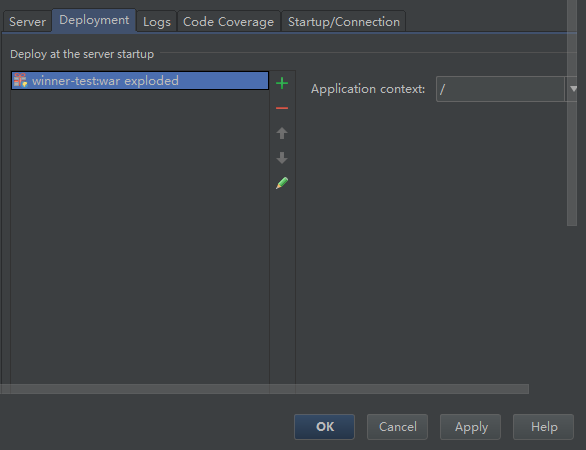
Run->Edit Configurations

点击左上角的"+"号，选择Tomcat Server，(如果没有请选择最下方的33 items more，找到Tomcat Server)，再选择Local：

点击 Application server 右边的 Configure，导入Tomcat 目录：

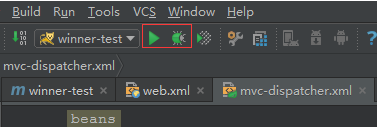
切换到Deployment标签，再点击右边的"+"号，添加一个Artifact.

选择第二个：war exploded，点击OK，这样，该项目就已经部署到了tomcat中.



再点击OK，整个Tomcat配置结束.

运行tomcat



项目启动加载完毕后访问http://127.0.0.1:8080/test.do



分类: [Spring MVC](http://www.cnblogs.com/winner-0715/category/755399.html)