# 配置tomcat

[返回主页](http://www.cnblogs.com/qixiaoyizhan/)

随笔 - 24  文章 - 2  评论 - 121

# [SpringMVC框架入门配置 IDEA下搭建Maven项目](http://www.cnblogs.com/qixiaoyizhan/p/5819392.html)

,初衷：本人初学SpringMVC的时候遇到各种稀奇古怪的问题，网上各种技术论坛上的帖子又参差不齐，难以一步到位达到配置好的效果，这里我将我配置的总结写到这里供大家初学SpringMVC的同僚们共同学习使用！

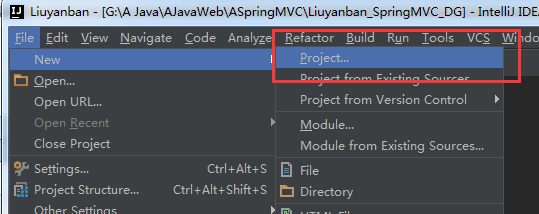
关于SpringMVC的介绍我就不在此赘述了，想必每一个配置Spring的人都已经或多或少了解过一些SpringMVC的基础知识，可能有的人用的myeclipse或者eclipse进行项目的搭建，我会告诉你这里步骤是很相似的，在myeclipse里面需要建立web项目，然后加入Spring支持，后续的jar包自己下载导入，其他配置和本文别无二致。但是本人推荐大家使用idea，这个编译器公认的java最好用的编译器，想必编写代码会更加的舒适快捷！

**在众多朋友的帮助指点下，本文不断完善，这里将修改完的完善版重新发表一遍，望广大网友雅正！！！**

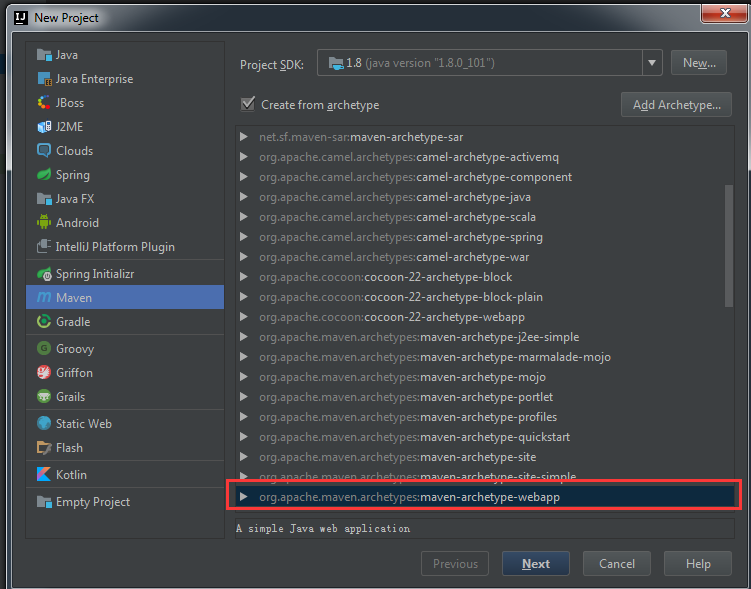
**这里只讲解如何在IntelliJ IDEA下搭建maven SpringMVC项目：**

**一、首先需要有一个IntelliJ IDEA 可以去官网下载，虽然是收费的，然而并不算贵。**

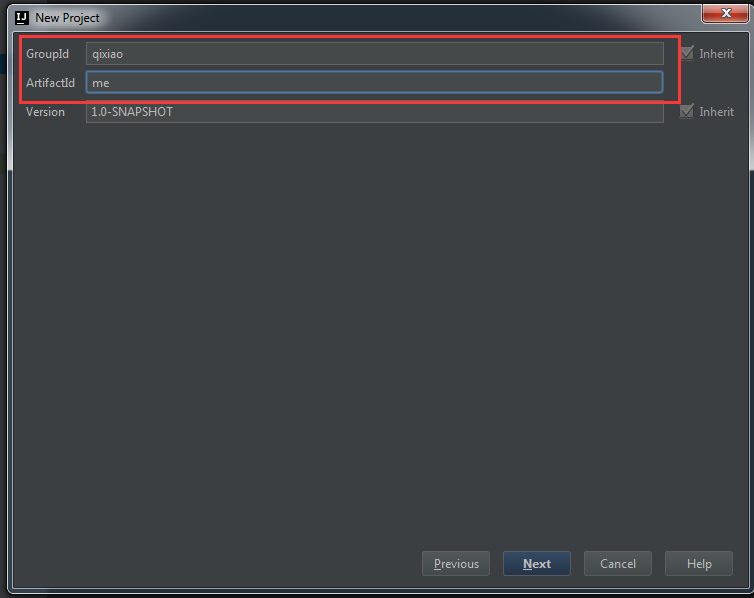
**二、运行编译器新建项目**



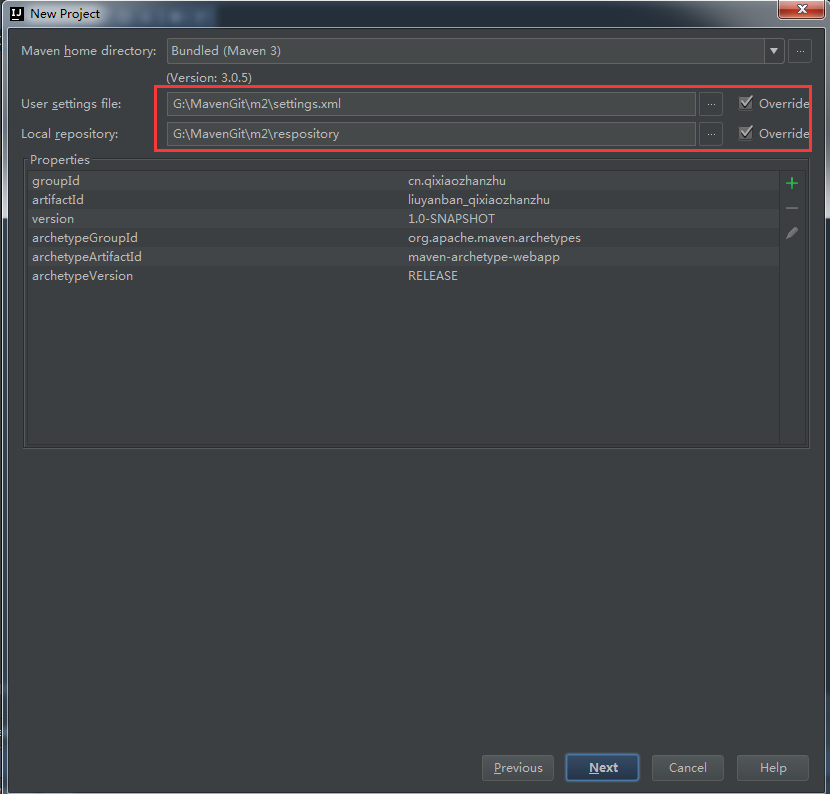
**在弹出的窗体中选择maven，然后勾选要建的maven模板--这里选webApp**



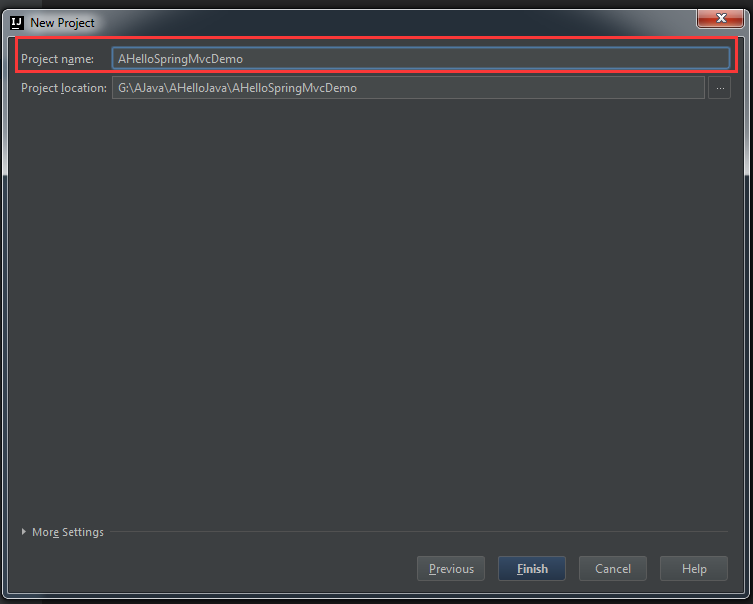
**然后填入相应的maven项目组信息（这个是比较随意的）**



**这里填写自己的maven本地仓库路径（这个不用过多解释吧，用maven的基础）；**



**这里填写项目名称以及项目生成路径，然后在下方检查填写是否正确：**

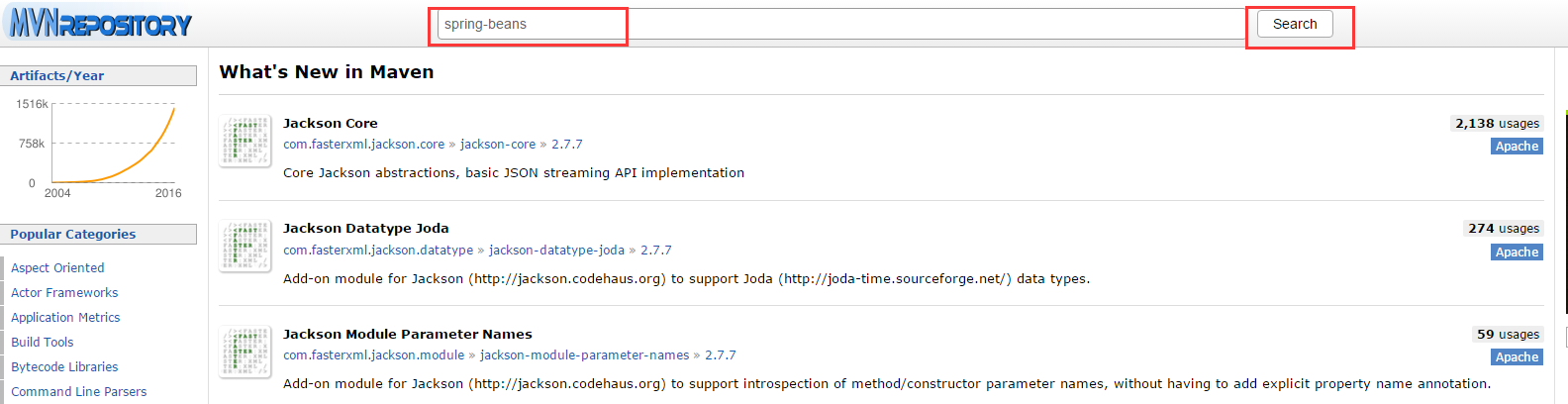


**如果正确就点击Finish->**

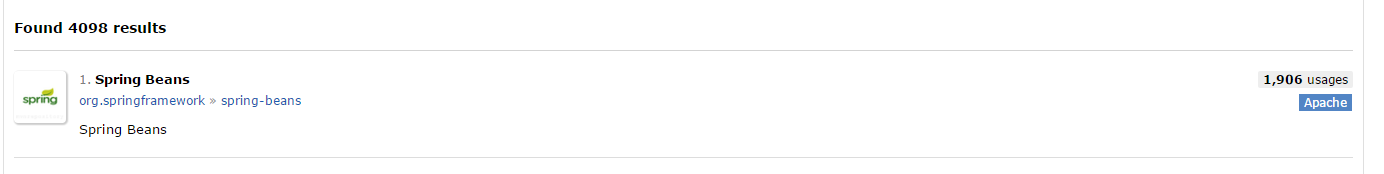
**maven会自动创建需要的一些配置信息以及目录结构，在这段时间里我们可以查找需要的jar包并在maven配置文件pom.xml里面进行配置，见下面步骤：**

**这里如果不知道要用到什么jar包就去百度SpringMVC需要的jar包，然后在maven的官方链接单独查找jar包来配置pom.xml，实例如下（这里我示范去maven官网查找xml配置的部分，具体需要的jar包去我后面展示的pom.xml里面查找）：**

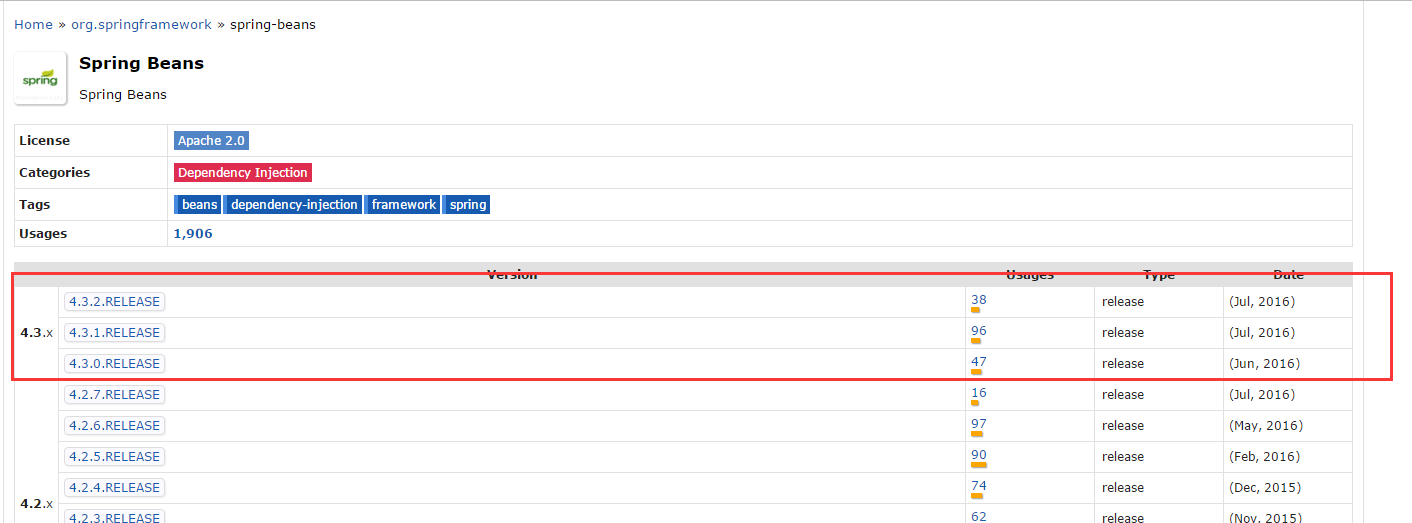
**登录**<http://mvnrepository.com/> **示例查找spring-beans**



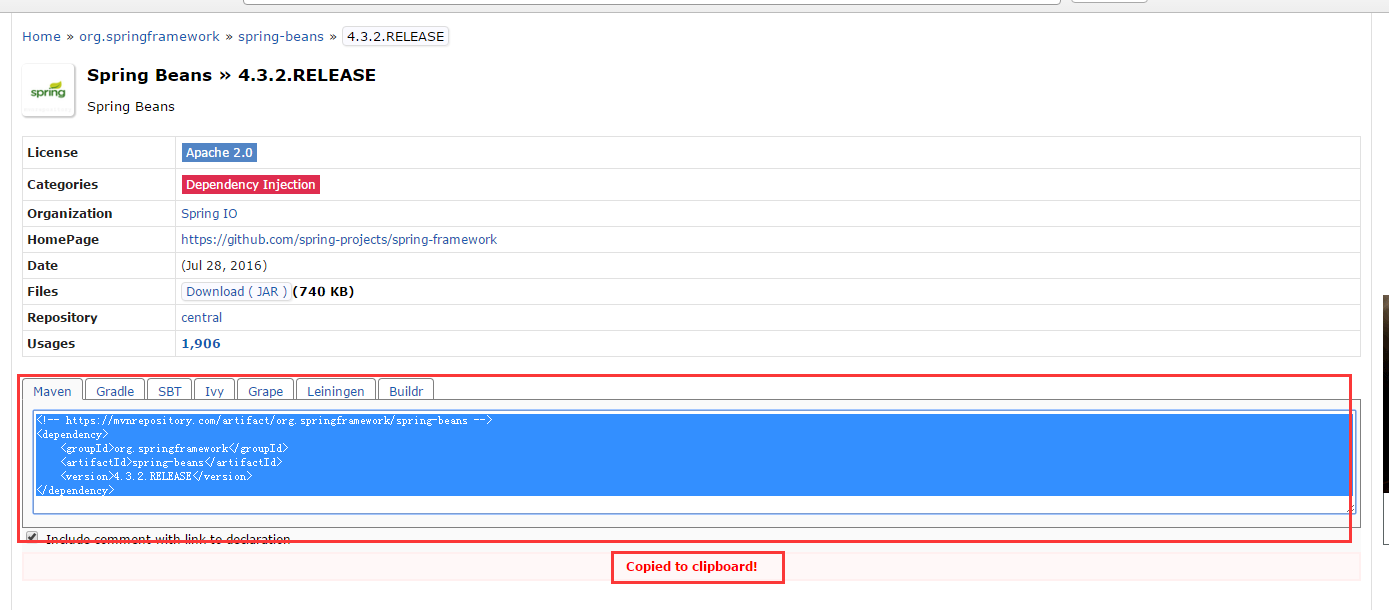
**点击查找结果**

****

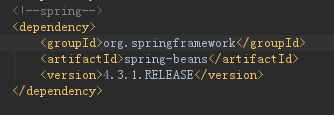
**这里可以看到最新版以及使用人数最多的版本，自己选择--配置文件里面的jar包版本最好选择同一个版本避免版本冲突；**

****

**我们选择第一个进入网页，里面可以看到maven的配置pom.xml文件写法，点击代码直接复制（自动复制）；**



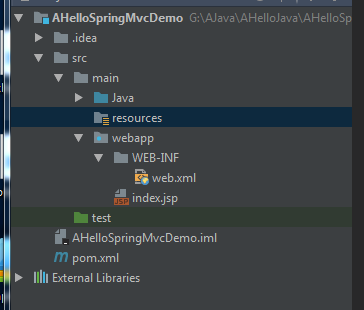
**然后将复制的代码拷贝到pom.xml文件中去即可，maven会自动下载所需要的jar包，我们不需要理会（如果有网络不顺畅情况，请自行想办法下载jar包或联系本人，然后将下载好的jar包拷贝到对应的maven本地仓库文件夹中也可以的）；**



**这里我展示我配置的maven pom.xml，maven会自动到库里面下载需要的jar包到maven仓库。**

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif pom.xml

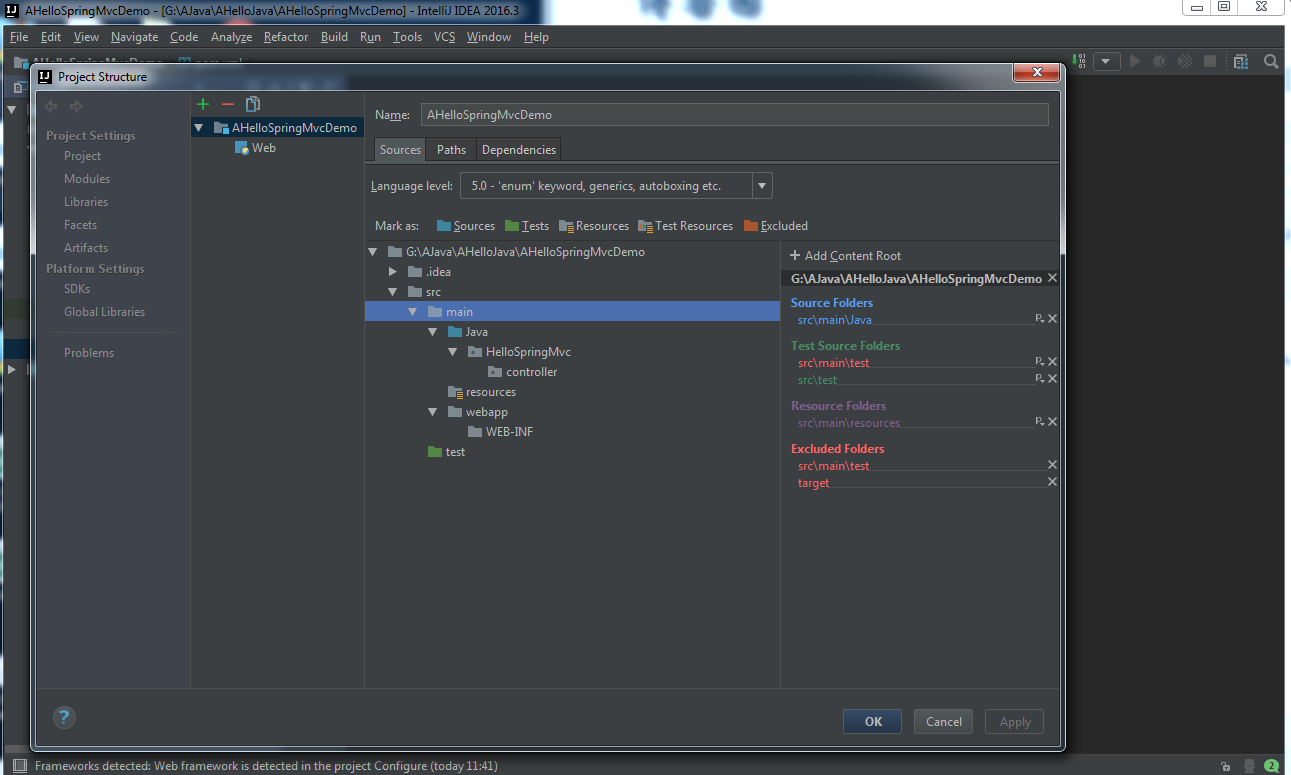
**等系统构建完成以后，我们就可以看到目录结构**



**不全的可以按照需求来补全文件结构，maven项目中的文件结构分为Sources，Tests，Resources，Test Resources，Excluded几种，我们需要详细区分各个文件夹的类型：**

http://images2015.cnblogs.com/blog/998608/201608/998608-20160829173514636-237517065.png

**配置文件夹类型方法如下：**



**可以选中文件夹点击上面的按钮来标注文件夹类型！**

**三、配置SpringMVC**

**一切就绪后接下来我们继续配置SpringMVC的具体信息：**

**首先需要配置Web.xml这个不必多说，网站项目运行第一个加载的就是web.xml，进入**

**src->main->webapp->WEB-INF->web.xml**

[复制代码](javascript:void(0);)

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

2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

3 xmlns="http://java.sun.com/xml/ns/javaee"

4 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"

5 version="3.0">

6 <welcome-file-list>

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

8 </welcome-file-list>

9 <!-- Spring MVC配置 -->

10 <servlet>

11 <servlet-name>spring</servlet-name>

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

13

14 <!-- 可以自定义servlet.xml配置文件的位置和名称，默认为WEB-INF目录下，名称为[<servlet-name>]-servlet.xml，如spring-servlet.xml

15 <init-param>

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

17 <param-value>/WEB-INF/spring-servlet.xml</param-value>

18 </init-param>-->

19

20 <!-- load-on-startup元素标记容器是否在启动的时候就加载这个servlet(实例化并调用其init()方法) -->

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

22 </servlet>

23

24 <servlet-mapping>

25 <servlet-name>spring</servlet-name>

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

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

28 </servlet-mapping>

29

30 <!-- Spring配置 -->

31 <listener>

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

33 </listener>

34

35 <!-- 指定Spring Bean的配置文件所在目录。默认配置在WEB-INF目录下 -->

36 <context-param>

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

38 <param-value>classpath:applicationContext.xml</param-value>

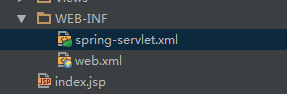
39 </context-param>

40

41 </web-app>

[复制代码](javascript:void(0);)

**然后在WEB-INF文件夹下新建 spring-servlet.xml**

****

**配置spring-servlet.xml这里采用注解的方式：**

[复制代码](javascript:void(0);)

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

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

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

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

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

6 xsi:schemaLocation="http://www.springframework.org/schema/beans

7 http://www.springframework.org/schema/beans/spring-beans-3.1.xsd

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

9 http://www.springframework.org/schema/context/spring-context-3.1.xsd

10 http://www.springframework.org/schema/mvc

11 http://www.springframework.org/schema/mvc/spring-mvc-3.1.xsd">

12

13 <!-- 启动注解驱动的Spring MVC功能，注册请求url和注解POJO类方法的映射-->

14 <mvc:annotation-driven >

15

16 </mvc:annotation-driven>

17

18 <!-- 启动包扫描功能，以便注册带有@Controller、@service、@repository、@Component等注解的类成为spring的bean -->

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

20 <!-- 对模型视图名称的解析，在请求时模型视图名称添加前后缀 -->

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

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

23 <property name="prefix" value="/"/> <!-- 前缀 -->

24 <property name="suffix" value=".jsp"/> <!-- 后缀 -->

25 </bean>

26 <!-- 访问静态文件（jpg,js,css）的方法 -->

27 <mvc:resources location="/files/" mapping="/files/\*\*" />

28 <mvc:resources location="/scripts/" mapping="/scripts/\*\*" />

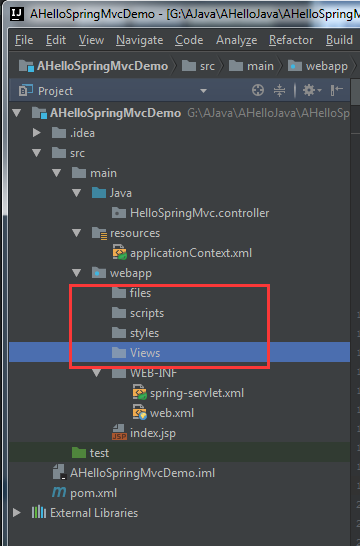
29 <mvc:resources location="/styles/" mapping="/styles/\*\*" />

30 <mvc:resources location="/Views/" mapping="/Views/\*\*" />

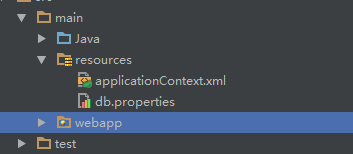
31 </beans>

[复制代码](javascript:void(0);)

**这里我们配置了静态文件路径，因此在目录结构新建几个文件夹用以放静态文件例如脚本，图片文件，视图之类的。**



**然后在Resources资源文件夹下新建applicationContext.xml**



**配置applicationContext.xml这里并没有进行配置什么信息，因为本人用的是mysql jdbc连接数据库，没有用到Hibernate之类的orm，这里可以将hibernate相关配置项放在里面。**

[复制代码](javascript:void(0);)

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

2 <beans

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

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

5 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.1.xsd">

6 <!-- 我们可以在其中添加我们所需要配置的bean，也可以添加相应的数据库连接和事务处理等等，方便后续拓展

7 -->

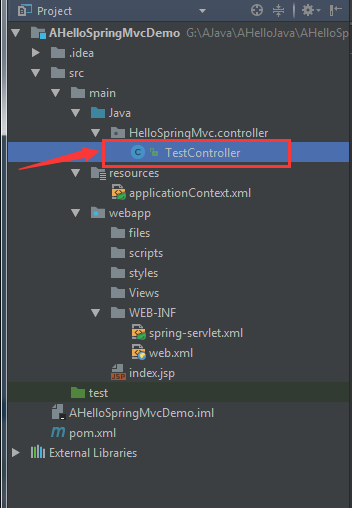
8 </beans>

[复制代码](javascript:void(0);)

**这里我们就将SpringMVC的基本XML配置文件设置完毕，接下来我们进行一个小实例进行测试配置好的SpringMVC。**

**四：测试SpringMVC配置：**

**1、在controller文件夹下新建一个类 名字叫TestController，代码如下：**

****

[复制代码](javascript:void(0);)

package HelloSpringMvc.controller;

import org.springframework.stereotype.Controller;

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

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

/\*\*

\* Created by Administrator on 2016/8/29.

\*/

@Controller

@RequestMapping(value = "/Test/\*") //访问的url地址前缀，可以不写，写了就必须在方法url前面先加上class url 进行区分控制器

public class TestController

{

//访问地址：http://localhost:8080/Test/returnSuccess

@RequestMapping(value = "returnSuccess") //实际访问的url地址

public String returnSuccess() {

return "/Views/success"; //返回Views文件夹下的success.jsp页面

}

//访问地址：http://localhost:8080/Test/returnString

@RequestMapping(value = "returnString", produces = {"text/plain;charset=UTF-8"})

//produces用于解决返回中文乱码问题，application/json;为json解决中文乱码

@ResponseBody //用于返回字符串,不写即返回视图

public String returnString() {

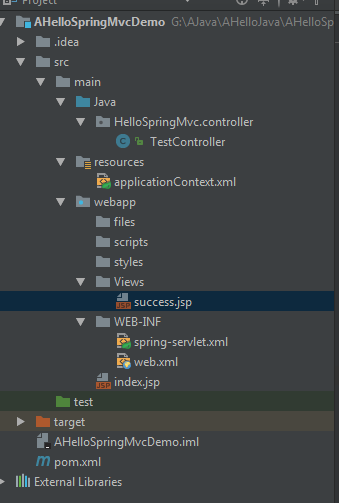
return "hello return string 这是中文，并没有乱码";

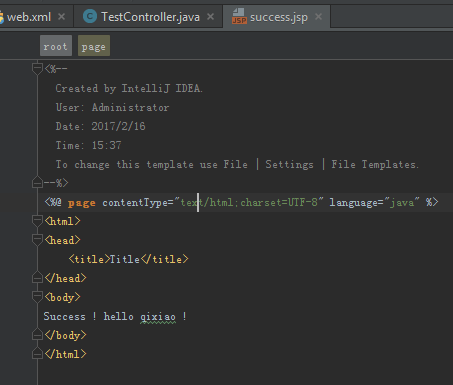
}

}

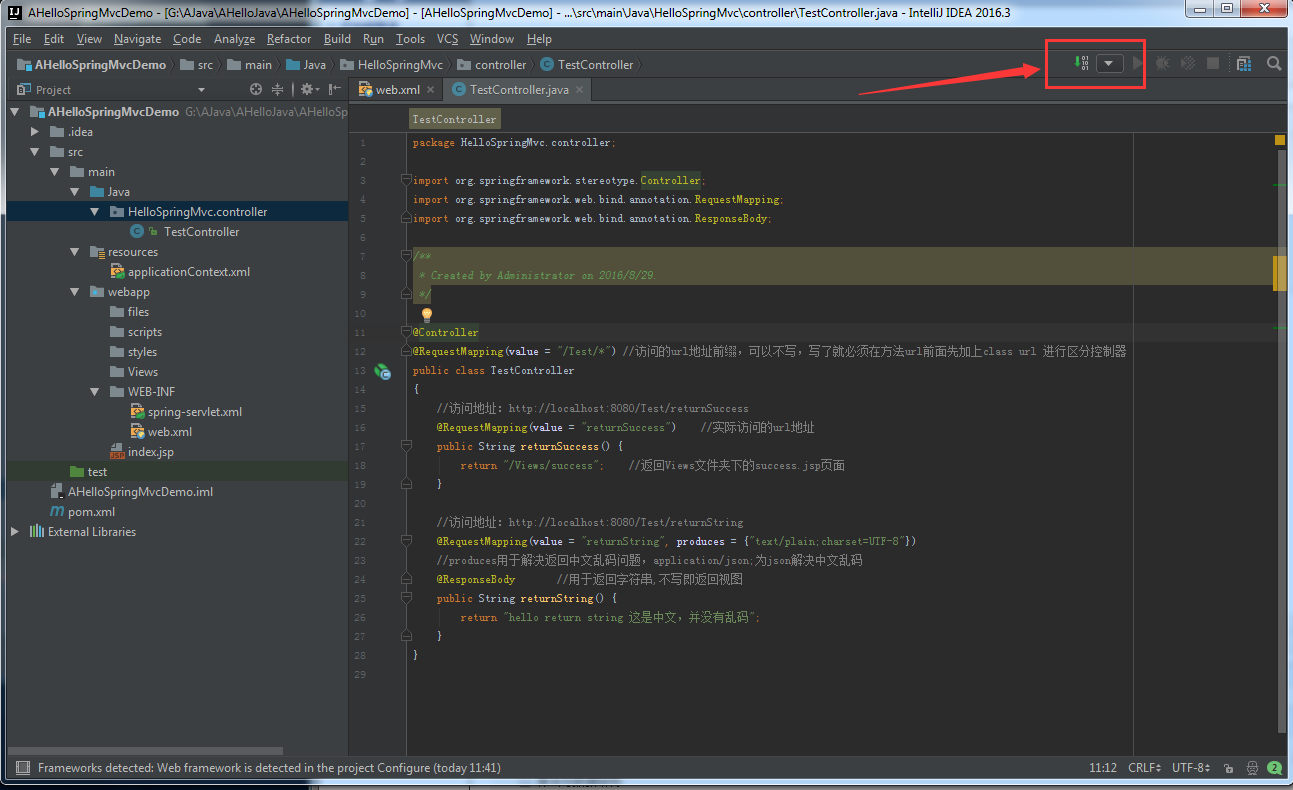
[复制代码](javascript:void(0);)

**然后我们在Views文件夹下建一个success.jsp 随便写点什么：**

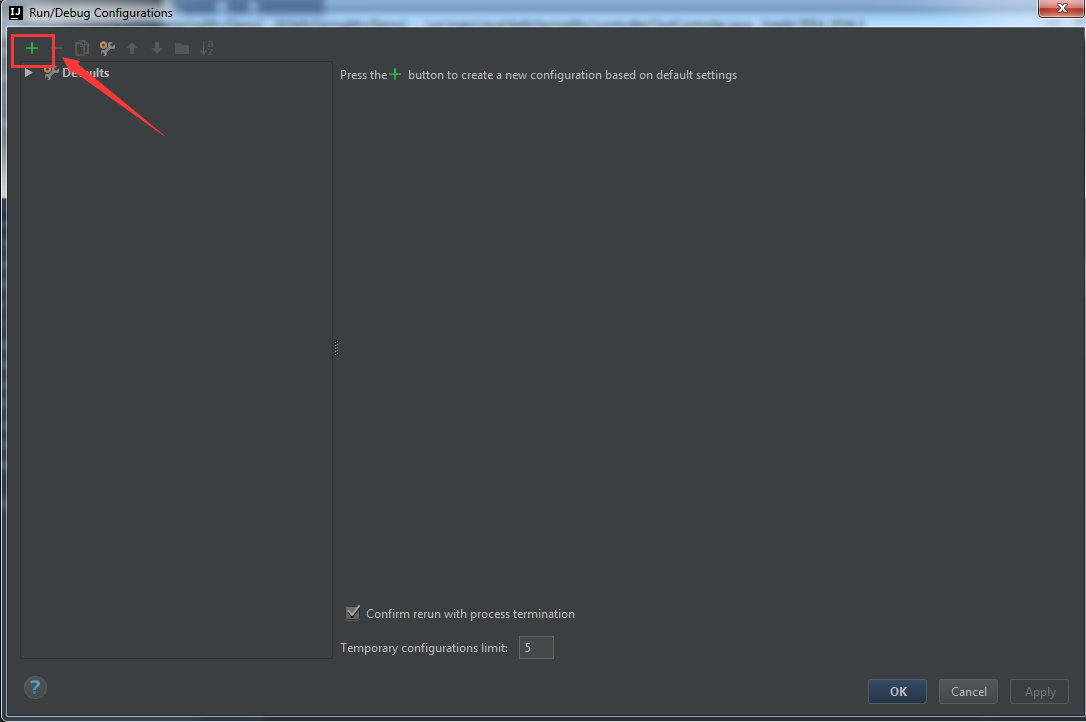
****



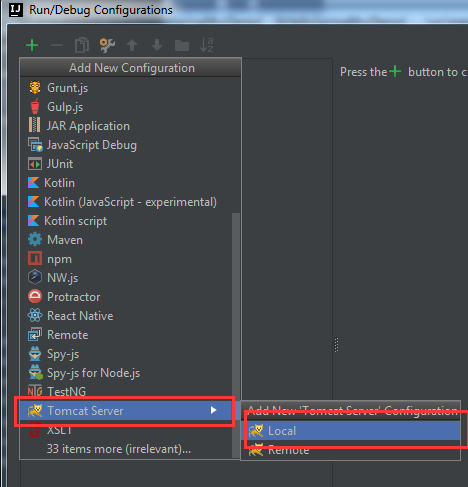
**然后试运行，IDEA需要先进行配置运行环境，配置如下->**



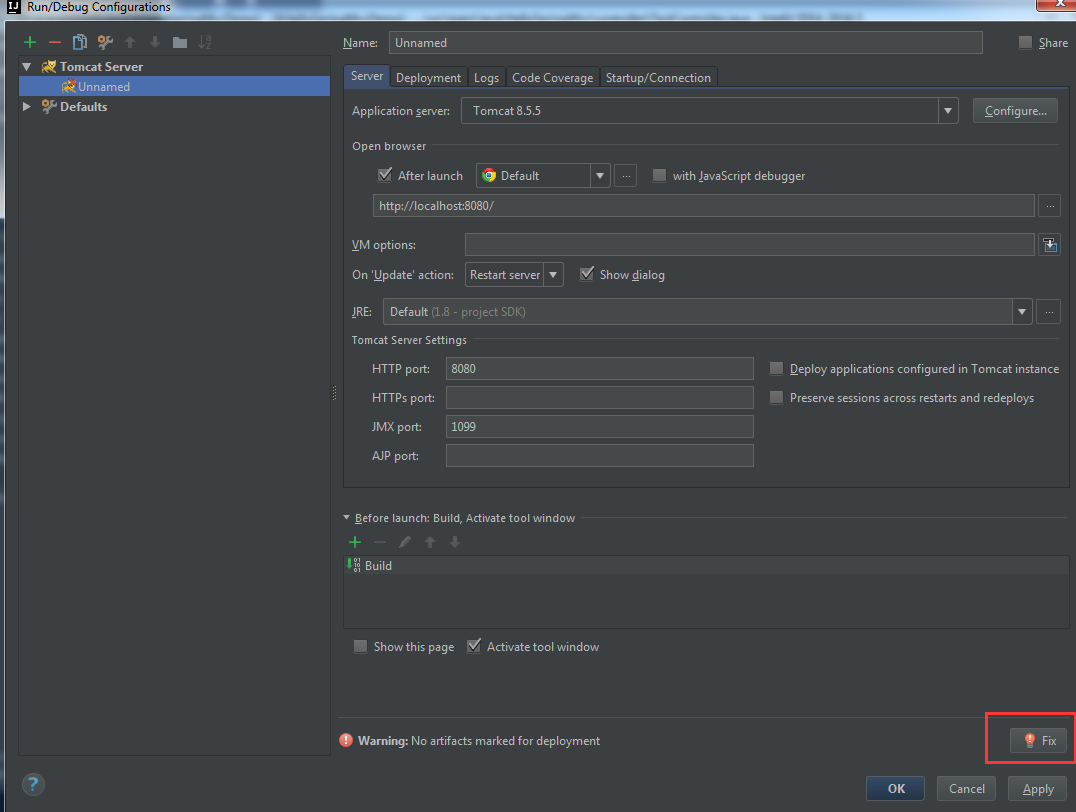
**在弹出的选项卡里面选择“+”号进行添加运行环境！**

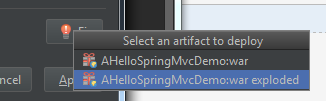


**本人这里选用TomCat启动项目:**

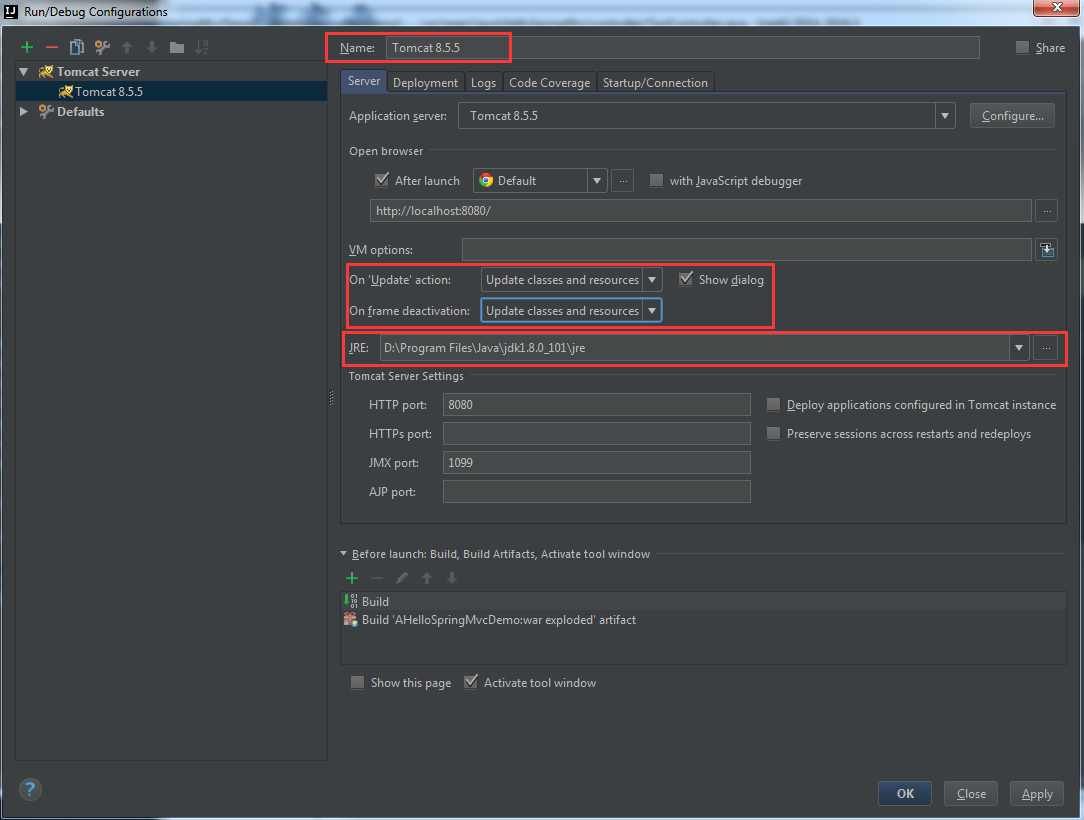


然后在弹出的选项卡选择Fix



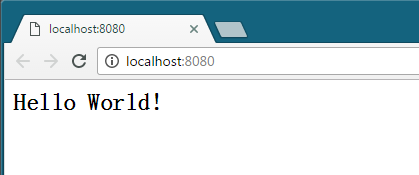


**然后应用即可，当然还有一些可选配置如下（这是我的配置）：**

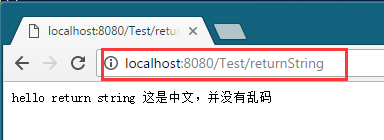


**接下来就可以运行了吧！！！**

# ****好,点击运行，google自动弹出了网页！****

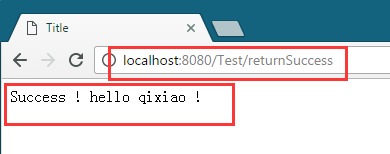
****

**运行后，在浏览器地址栏输入：http://localhost:8080/Test/returnString**



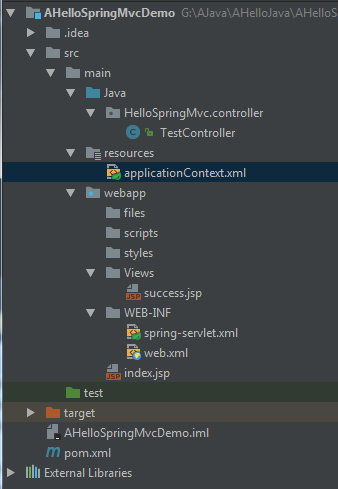
**可以看到我们运行第二个url地址可以成功返回字符串，而且并没有中文乱码！**

**下面测试第一个url地址：http://localhost:8080/Test/returnSuccess**



**可以看到，成功返回到success.jsp ! 说明我们的SpringMVC并没有配置错误。**

**再次将项目目录结构附上->**

****

**还有人习惯于使用Json，这里可以将json直接当string输出，前台就可以获取到，json的jar包配置也可以自行查询，这里不再赘述。**

2016-08-29 19:27:41

**本文为七小站主原创作品，转载请注明出处：http://www.cnblogs.com/qixiaoyizhan/**

# 项目源代码下载：

<http://files.cnblogs.com/files/qixiaoyizhan/AHelloSpringMvcDemo.zip>

另外安利博主个人博客，欢迎参观，另本站蓬荜生辉！！！

[http://qixiao.me](http://qixiao.me/)

 Thx and best regards~

本文为七小站主原创作品，转载请注明出处：[http://www.cnblogs.com/qixiaoyizhan/](http://www.cnblogs.com/qixiaoyizhan/" \t "_blank) 且在文章页面明显位置给出原文链接，否则保留追究法律责任的权利。 

http://mark-yang.github.io/blog/2015/04/24/[**spring**](http://lib.csdn.net/base/javaee)-springmvc-[**hibernate**](http://lib.csdn.net/base/javaee)/

# [Spring-SpringMVC-Hibernate在IntelliJ与Maven的环境下搭建](http://blog.csdn.net/adermxl/article/details/46011499)

本篇主要介绍的是JavaEE中的框架SpringMVC，在以前上JavaEE课程时，课程项目里使用的是比较“古老”的SSH（Struts2+Spring+Hibernate）框架，最近受某小朋友的影响着手了解SpringMVC，一种更轻量级，更方便使用的框架。老规矩，附Wikipedia链接——[Spring](https://en.wikipedia.org/wiki/Spring_Framework)，其中Model-view-controller framework部分的介绍就是我们常说的SpringMVC。

受以前项目的影响，感觉框架的合理使用可以省去不少的麻烦。所以此次介绍的是另一种多框架的混合使用，即，Spring + SpringMVC + Hibernate框架在InteliJ和Maven的环境下的搭建，并采用注释的写法。后续还会有在此框架的基框之上的[demo](http://mark-yang.github.io/blog/2015/04/24/spring-springmvc-hibernate/)，敬请期待。

### 搭建环境

[**操作系统**](http://lib.csdn.net/base/operatingsystem)：Mac OSX Yosemite Version 10.10.3

集成开发环境（IDE）：IntelliJ IDEA 14.1

本地服务器：Tomcat 8.0.21

Maven版本：Maven3 3.0.5

## 目录

一、SpringMVC框架的使用

二、SpringMVC与Spring结合

三、Hibernate框架的加入

四、总结

### 一、SpringMVC框架的使用

1.建立工程

首先我们在IntelliJ中建立一个JavaEE Project，选择Maven工程，选择org.apache.marmotta:marmotta-archetype-webapp，然后输入GroupId与ArtifactId，这里就不做详细的介绍了。

2.项目结构

工程建好了之后，如果想先看一下本文所展示的demo的工程目录，用来为后面的步骤过程中路径配置时作为参考，其次还有项目的源码都请戳：structure:<https://raw.githubusercontent.com/MarK-YANG/Spring-SpringMVC-Hibernate/master/structure.png>  
源码：<https://github.com/MarK-YANG/Spring-SpringMVC-Hibernate>

3.pom.xml

正如开始的时候介绍的一样，我们使用Maven来管理我们在项目使用到的各种包，而所有包的配置信息都在pom.xml文件中，所以我们首要的工作就是通过pom.xml文件来找并管理到我们想要的包。在pom.xml文件中加入我们所需要的dependence和properties，具体如下：

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223242526272829303132333435363738394041424344454647484950515253 | 4.0.4.RELEASEorg.springframeworkspring-webmvc${spring.version}javax.servlet.jsp.jstljavax.servlet.jsp.jstl-api1.2.1taglibsstandard1.1.2tomcatservlet-api5.5.23providedtomcatjsp-api5.5.23providedcommons-fileuploadcommons-fileupload1.3.1org.hibernatehibernate-validator5.1.2.Finalorg.codehaus.jacksonjackson-mapper-asl1.9.13 |

然后，简单的介绍一下这些我们导入的包， spring-webmvc是我们SpringMVC框架的核心包，spring-core,spring-web都是其依赖包。

其次，jstl、taglibs等包是为了在jsp页面中使用JSTL标签。

然后，jackson-mapper包是使springMVC可以返回json值，json和xml方法是我们常用的两种网络传值方式。在springMVC中仅提供了json的接口，并未提供其实现，所以我们要导入它的实现包json-mapper。

再之，commons-fileupload包是为了文件上传所需要的，在本文中的demo中并未涉及到文件的上传，可以选择不添加。

然后，hibernate-validator是为了正常的使用springMVC中的valid，这是因为springMVC中的valid只提供了接口的实现，如果想使用这个接口，就必须要导入它的实现——hibernate-validator，但这项在本文中也未涉及，可以选择不添加。

4.修改web.xml

springMVC中只有一个核心的Servlet，所以我们要在web.xml中注册这个springMVC的Servlet，web.xml文件如下，

|  |  |
| --- | --- |
| 12345678910111213141516 | xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns="http://java.sun.com/xml/ns/javaee"xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"metadata-complete="true"version="3.0"**>**spring-dispatcherorg.springframework.web.servlet.DispatcherServletspring-dispatcher/ |

这只是我们配置springMVC的开始，springMVC还有许多其他的配置项，但其他的配置项在另一个文件中，这个配置文件就是默认位置为web.xml同路径下的——-servlet.xml。如，我的web.xml中配置的是spring-dispatcher，对应的xml文件的名字应为：spring-dispatcher-servlet.xml。

5.开始配置spring-dispatcher-servlet.xml

我们在与web.xml同路径下创建spring-dispatcher-servlet.xml如下：

|  |  |
| --- | --- |
| 12345678910111213141516171819202122232425262728 | 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-3.0.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd"**>**base-package="com.mark.java.controller"**/>/>**id="viewResolver"class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>**name="prefix"value="/pages/"**/>**name="suffix"value=".jsp"**/>**id="multipartResolver"class="org.springframework.web.multipart.commons.CommonsMultipartResolver"**>**name="defaultEncoding"value="UTF-8"**/>**name="maxUploadSize"value="2000000"**/>**mapping="/static/\*\*"location="/"**/>/>** |

简单的解释一下，springMVC中的虚拟路径与处理函数的映射，是通过annotation（注释）来设置的，而这些类都在哪个包下面，就是通过component-scan属性配置的，并设置支持annotation支持。

springMVC的每一个处理函数都会返回一个String类型的变量（有的函数返回值就是 String，有的返回值是一个对象，对象里有一个viewResolver需要的字符串），我们这里设置了一个前缀和后缀，举个例子来说，对于函数返回的这个String，比如是fowafolo，那么需要的文件就是/pages/fowafolo.jsp。

multipartResolver是上传文件需要的配置，本文中并未涉及到。

最后是对静态资源的映射，经过这个映射之后，要访问/static/fowafolo.png时，实际访问的就是/resources/fowafolo.png。

6.MainController

下面我们开始写我们springMVC中的第一个controller, MainController，这个controller的位置就应该是我们刚才配置的spring-dispatcher-servlet.xml中component-scan配置的路径下，代码如下：

|  |  |
| --- | --- |
| 12345678910111213141516171819 | @Controller@RequestMapping**(**"/"**)**publicclassMainController**{**@RequestMapping**(**""**)**publicString**index(){**return"index"**;}**@RequestMapping**(**"/json"**)**@ResponseBodypublicMap**<**String**,**String**>json(){**Map**<**String**,**String**>**result**=**newHashMap**<**String**,**String**>();**result**.**put**(**"MarK"**,**"hello"**);**result**.**put**(**"Ken"**,**"Hehe"**);**result**.**put**(**"Fowafolo"**,**"fool"**);**returnresult**;}}** |

简单的解释一下，这个controller的RequestMapping是"/"，index()的RequestMapping也是"/"，而方法json的RequestMapping是"/json"，也就是说，当访问的URL是”/“时，我们会调用index()方法处理，它返回的字符串是index，再根据我们配置文件中的路径，它指向的就是我们/webapp/pages下的index.jsp，所以别忘了去在这个目当下新建一个index.jsp，给一个简单的例子。

|  |  |
| --- | --- |
| 123456789101112 | <%@ page contentType="text/html;charset=UTF-8" language="java" %>  Hello, Fowafolo  Fowafolo is a foolish ! |

部署工程，我们给这个工程加一个前缀，如/spring，那么我们访问：<http://localhost:8080/spring/>得到的就是我们新建的index.jsp，如果我们访问<http://localhost:8080/spring/json>得到的就是我们json的返回值:{"MarK":"hello","Ken":"Hehe","Fowafolo":"fool"}。

最后，我们再多说几句，因为我们想在/webapp/pages下管理我们所有jsp，所以/webapp下的index.jsp可以不保留，其次，在部署项目时，不建议勾选此项，手动的在browser中输入url更好。

### 二、SpringMVC与Spring结合

Spring是一个开源框架，Spring是于2003 年兴起的一个轻量级的[**Java**](http://lib.csdn.net/base/java)开发框架，由Rod Johnson在其著作Expert One-On-One J2EE Development and Design中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。框架的主要优势之一就是其分层[**架构**](http://lib.csdn.net/base/architecture)，分层架构允许使用者选择使用哪一个组件，同时为J2EE应用程序开发提供集成的框架。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可[**测试**](http://lib.csdn.net/base/softwaretest)性和松耦合的角度而言，任何Java应用都可以从Spring中受益。Spring的核心是控制反转（IoC）和面向切面（AOP）。简单来说，Spring是一个分层的JavaSE/EEfull-stack(一站式) 轻量级开源框架。

1. 再次配置web.xml

Spring要在程序调用某个对象的时候，就要首先把这个对象的实例注入进去。Spring自己对实例、程序运行的管理构成了Spring自己的容器，第一步就是要在web.xml中注册初始化这个容器。在此之前我们先在/resources文件夹下，创建子路径META-INF，然后在这个文件夹下，创建我们后面要用到的applicationContext.xml文件，如果用过spring框架的人对这个文件都不会陌生，这是spring的配置文件。最后web.xml文件如下：

|  |  |
| --- | --- |
| 123456789101112131415161718192021222324 | xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns="http://java.sun.com/xml/ns/javaee"xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"metadata-complete="true"version="3.0"**>**spring-dispatcherorg.springframework.web.servlet.DispatcherServletspring-dispatcher/contextConfigLocationclasspath:/META-INF/applicationContext.xmlorg.springframework.web.context.ContextLoaderListener |

2. 配置applicationContext.xml文件

在spring4之后，spring配置的bean不是在xml文件中配置了，而是采用固定扫描annotation的类，根据对象的类型或者名字自动加载，但此前提是我们仍然要在配置文件中配置好这些类所在包的路径。最后applicationContext.xml文件如下，

|  |  |
| --- | --- |
| 1234567891011121314 | xmlns="http://www.springframework.org/schema/beans"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns:aop="http://www.springframework.org/schema/aop"xmlns:context="http://www.springframework.org/schema/context"xmlns:jee="http://www.springframework.org/schema/jee"xmlns:p="http://www.springframework.org/schema/p"xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd"**>**base-package="com.mark.java.service"**/>** |

3. 创建一个Service类，UserService.java

Service的创建我们还是比较推荐具体的service实现相应的接口的方式，下面我们写一个服务类，用来获取注册的用户，在包com.mark.java.service下创建接口UserService。

|  |  |
| --- | --- |
| 123 | publicinterfaceUserService**{**publicList**<**String**>getAllUsernames();}** |

然后再创建一个UserServiceImpl来实现此接口，

|  |  |
| --- | --- |
| 123456789101112 | @ServicepublicclassUserServiceImplimplementsUserService**{**publicList**<**String**>getAllUsernames(){**List**<**String**>**users**=**newArrayList**<**String**>();**users**.**add**(**"MarK"**);**users**.**add**(**"Ken"**);**users**.**add**(**"Fowafolo"**);**returnusers**;}}** |

注意！！！，我们要对个实现加上注释@Service，这是在告诉spring，我需要你来管理我。

4. 修改MainController

在上一节中，我们通过json来返回一个json对象，现在我们修改这个方法，通过刚刚创建的service来返回用户列表给客户端。

首先，我们得先定义一个全局变量，usrService，

|  |  |
| --- | --- |
| 12 | @AutowiredprivateUserServiceuserService**;** |

这里要注意的是Autowire这个annotation，它是为了告诉spring，这个对象没有实例化，需要注入一个UserService的实例，可是问题是，UserSerivce是一个接口，如果不指定就不知道你想用哪个实现类，Spring会首先看自己的容器里有没有一个叫做userService的对象（刚才创建的UserServiceImpl的对象名字就叫做userServiceImpl），如果找不到就在配置文件里配置的路径下面寻找UserService的实现类，找到了就把它的对象拿过来.除此之外刚才Service那个annotation还可以指 定一个value：@Service("userService")。这样一来，对于UserServiceImpl这个类的实例，Spring给它起的名字就不是userServiceImpl了，而是userService，如果某个接口的实现类有多个，就可以使用这种方法来指定用哪个实现类，个人认为，如果每个接口都只有一个实现类，那么这么做确实很方便，但如果有多个实现类并且可能会更换的话，就不如配置文件明了了。

说了半天，最后修改json方法，

|  |  |
| --- | --- |
| 12345 | @RequestMapping**(**"/json"**)**@ResponseBodypublicList**<**String**>json(){**returnuserService**.**getAllUsernames**();}** |

Spring在这里的作用就是把MainController需要的userService引入进来，然后访问<http://localhost:8080/spring/json>会得到以下结果：

["MarK","Ken","Fawofolo"]

### 三、Hibernate框架的加入

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，使得Java程序员可以随心所欲的使用对象编程思维来操纵[**数据库**](http://lib.csdn.net/base/mysql)。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的J2EE架构中取代CMP，完成数据持久化的重任。

1. 导入Hibernate所需要的包

在pom.xml中，在properties标签下面定义一个版本，

|  |  |
| --- | --- |
| 1234 | 4.0.4.RELEASE4.3.5.Final |

然后加入hibernate的核心包，因为要让Spring管理一些对象，还要引入spring-orm的包：

|  |  |
| --- | --- |
| 12345678910 | org.springframeworkspring-orm${spring.version}org.hibernatehibernate-core${hibernate.version} |

除此之外，因为我们要用[**MySQL**](http://lib.csdn.net/base/mysql)数据库，并使用c3p0连接池，因此需要引入这两个对应的包：

|  |  |
| --- | --- |
| 12345678910 | c3p0c3p00.9.1.2mysqlmysql-connector-java5.1.31 |

2. 再次修改web.xml

Hibernate通过SessionFactory来获取Session,我们要在spring的配置文件中配置一个我们所需的SessionFactory，为了便于修改，我们新建一个配置文件，路径与spring的配置文件相同，我们新建一个infrastructure.xml, 最后我们的web.xml文件如下，其他的不做具体的介绍了。

|  |  |
| --- | --- |
| 12345678910111213141516171819202122232425262728293031323334353637383940414243444546474849505152535455 | xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns="http://java.sun.com/xml/ns/javaee"xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"metadata-complete="true"version="3.0"**>**spring-dispatcherorg.springframework.web.servlet.DispatcherServletspring-dispatcher/hibernateFilterorg.springframework.orm.hibernate4.support.OpenSessionInViewFilterhibernateFilter/\*SpringEncodingFilterorg.springframework.web.filter.CharacterEncodingFilterencodingUTF-8forceEncodingtrueSpringEncodingFilter/\*contextConfigLocationclasspath:/META-INF/applicationContext.xml, classpath:/META-INF/infrastructure.xmlorg.springframework.web.context.ContextLoaderListener |

3. 配置hibernate.properties, infrastructure.xml

infrastructure.xml之前，我们要在/META-INF下新建properties文件夹，在里在新建hibernate.properties文件用来加入我们数据库的参数，hibernate.properties的配置如下，

|  |  |
| --- | --- |
| 12345678910 | hibernate.dialect**=**org.hibernate.dialect.MySQLDialectdriverClassName**=**com.mysql.jdbc.DrivervalidationQuery**=**SELECT 1url**=**jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=UTF-8username**=**rootpassword**=**yanghibernate.hbm2ddl.auto**=**nonehibernate.show\_sql**=**truehibernate.format\_sql**=**true |

注意以下几点，

driverClassName是你所连数据库的驱动的jar包的名称，这里我用的**[mysql](http://lib.csdn.net/base/mysql)**，所以是com.mysql.jdbc.Driver，

url是连接数据库的url，这里是jdbc:mysql://localhost:3306/本机本地数据库，后面是数据库的名称，为test，然后后面的一串，是JDBC对数据库进行CRUD时的编码，我们采用UTF-8以支持中文。

username和password是本地数据库的用户名和密码。

其他项可无视。infrastructure.xml的配置如下：

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223242526272829303132333435363738394041424344454647484950515253 | xmlns="http://www.springframework.org/schema/beans"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns:tx="http://www.springframework.org/schema/tx"xmlns:aop="http://www.springframework.org/schema/aop"xmlns:context="http://www.springframework.org/schema/context"xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd "**>**location="classpath:/META-INF/properties/hibernate.properties"**/>**id="dataSource"class="com.mchange.v2.c3p0.ComboPooledDataSource"destroy-method="close"**>**name="driverClass"value="${driverClassName}"**>**name="jdbcUrl"value="${url}"**>**name="user"value="${username}"**>**name="password"value="${password}"**>**name="maxPoolSize"value="40"**>**name="minPoolSize"value="1"**>**name="initialPoolSize"value="1"**>**name="maxIdleTime"value="20"**>**id="sessionFactory"class="org.springframework.orm.hibernate4.LocalSessionFactoryBean"**>**name="dataSource"ref="dataSource"**/>**name="packagesToScan"value="com.mark.java.entity"**/>**name="hibernateProperties"**>**key="hibernate.hbm2ddl.auto"**>**${hibernate.hbm2ddl.auto}key="hibernate.dialect"**>**${hibernate.dialect}key="hibernate.show\_sql"**>**${hibernate.show\_sql}key="hibernate.format\_sql"**>**${hibernate.format\_sql}id="transactionManager"class="org.springframework.orm.hibernate4.HibernateTransactionManager"**>**name="sessionFactory"ref="sessionFactory"**>**transaction-manager="transactionManager"**/>** |

其所需的参数信息都是从刚才配置好的properties文件中读取。

4. 再次修改applicationContext.xml

我们还需要对这个配置文件做一点点小的修改，加入两行，

|  |  |
| --- | --- |
| 123 | base-package="com.mark.java.entity"**/>**base-package="com.mark.java.DAO"**/>** |

解释一下，其中com.mark.java.entity则是我们MVC模式中model或称为JavaBean的文件夹目录，其格式应与数据库中字段相同；com.mark.java.DAO是访问数据库要使用DAO，即数据访问对象的目录。我们把这些同service都交给spring管理。最后applicationContext.xml文件如下，

|  |  |
| --- | --- |
| 1234567891011121314151617 | xmlns="http://www.springframework.org/schema/beans"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns:aop="http://www.springframework.org/schema/aop"xmlns:context="http://www.springframework.org/schema/context"xmlns:jee="http://www.springframework.org/schema/jee"xmlns:p="http://www.springframework.org/schema/p"xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd"**>**base-package="com.mark.java.service"**/>**base-package="com.mark.java.entity"**/>**base-package="com.mark.java.DAO"**/>** |

5. 建立表User

我们在我们的本地数据库中建立一个表，名为User, 其有两个字段，一个为id，主键，int类型，自增，另一个为usrname, varchar。

具体请戳：<https://raw.githubusercontent.com/MarK-YANG/Spring-SpringMVC-Hibernate/master/table.png>

6. 创建JavaBean

在entity目录下创建User.java

|  |  |
| --- | --- |
| 1234567891011121314151617181920212223242526 | @Entity@TablepublicclassUserimplementsSerializable**{**@Id@GeneratedValueprivateintid**;**privateStringusername**;**publicint**getId(){**returnid**;}**publicvoid**setId(**intid**){**this**.**id**=**id**;}**publicString**getUsername(){**returnusername**;}**publicvoid**setUsername(**Stringusername**){**this**.**username**=**username**;}}** |

注：这里的annotation用的都是JPA的接口。

7. 创建UserDAO

首先创建UserDAO接口，

|  |  |
| --- | --- |
| 1234 | publicinterfaceUserDAO**{**publicint**save(**Useru**);**publicList**<**User**>findAll();}** |

创建UserDAOImpl实现此接口，

|  |  |
| --- | --- |
| 123456789101112131415 | @RepositorypublicclassUserDAOImplimplementsUserDAO**{**@AutowiredprivateSessionFactorysessionFactory**;**publicint**save(**Useru**){**return**(**Integer**)**sessionFactory**.**getCurrentSession**().**save**(**u**);}**publicList**<**User**>findAll(){**Criteriacriteria**=**sessionFactory**.**getCurrentSession**().**createCriteria**(**User**.**class**);**returncriteria**.**list**();}}** |

注：第一个annotation注明这是一个repository，需要被Spring管理，然后自动装载之前在配置文件中配置的 SessionFactory，获取当前session后获取所有的用户，下面修改之前的UserService代码，service封装业务逻辑层代 码，我把每个service方法封装为一个事务。（PS:上面的SessionFactory获取当前Session是依赖于事务的，如果不在某个事务之内， 会报错：No Session found for current thread。)

8.更新UserService

修改UserService接口，

|  |  |
| --- | --- |
| 1234 | publicinterfaceUserService**{**publicvoid**saveUsers(**List**<**User**>**us**);**publicList**<**User**>getAllUsernames();}** |

修改UserServiceImpl,

|  |  |
| --- | --- |
| 1234567891011121314151617 | @Service**(**"userService"**)**@TransactionalpublicclassUserServiceImplimplementsUserService**{**@AutowiredprivateUserDAOuserDao**;**publicvoid**saveUsers(**List**<**User**>**us**){**for**(**Useru**:**us**){**userDao**.**save**(**u**);}}**publicList**<**User**>getAllUsernames(){**returnuserDao**.**findAll**();}}** |

注：这个annotation为Transactional，就是指明每个方法是一个事务。

9. 修改MainController

|  |  |
| --- | --- |
| 123456789101112131415161718192021222324252627 | @Controller@RequestMapping**(**"/"**)**publicclassMainController**{**@AutowiredprivateUserServiceuserService**;**@RequestMapping**(**""**)**publicString**home(){**List**<**User**>**us**=**newArrayList**<**User**>();**Useru**=**new**User();**u**.**setUsername**(**"MarK"**);**us**.**add**(**u**);**u**=**new**User();**u**.**setUsername**(**"Fawofolo"**);**us**.**add**(**u**);**userService**.**saveUsers**(**us**);**return"index"**;}**@RequestMapping**(**"/json"**)**@ResponseBodypublicList**<**User**>json(){**returnuserService**.**getAllUsernames**();}}** |

部署项目，访问<http://localhost:8080/spring>加入两条新的数据，

访问<http://localhost:8080/spring/json>查看这两条数据，

结果如下：[{"id":1,"username":"MarK"},{"id":2,"username":"Fawofolo"}]

注：如果发现数据插入两遍，请回头看第一节关于项目部署的介绍。

至此，环境搭建完毕，后面还会有一篇在此环境基础之上做的springMVC的简单实例，敬请期待…

完。

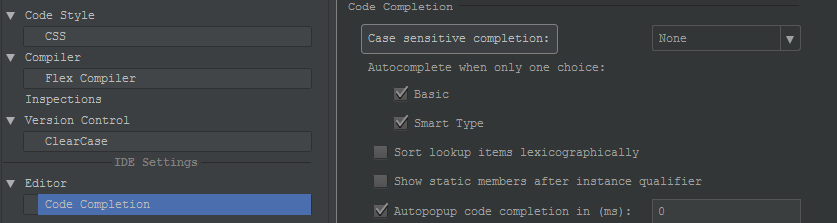
# idea使用

## 设置大小写不敏感提示

设置里可以找到啊！在：Editor-Code Completion页里有个Case sensitive completion，可以设置只第一个字母敏感、完全敏感或者不敏感。

评论(0)[引用此答案](https://www.oschina.net/question/answer?question=121766&answer=559959)举报

1



# spring-mvc

## 网页显示json字符串乱码

