

使用 IntelliJ IDEA 开发 SpringMVC 网站

前言

由于近期一直在做学术方面的工作，项目开发相关工作并没有花太多的时间，导致这篇文章的更新停步了很长一段时间。现在应大家的要求，补上剩余部分，希望能给大家带来一些帮助。由于时间的原因，在开发环境上面有了一定的更新，但是并不造成太大的影响。

最近在做某在线教育平台网站的开发，按师兄的建议要用 SpringMVC 来搞。之前对 Spring MVC 的认知度为 0，网上查阅各种资料，发现五花八门的配置都有，文章写的那叫一个乱啊，我觉得有些文章还是不要发出来的比较好，简直误人子弟耽误时间。最近借着师兄网上搜集的一些开发经验，找到了 IntelliJ 网站上的这篇文章《[Getting Started with SpringMVC, Hibernate and J SON](#)》（该链接已失效，内容会在文中体现），外加看了孔老师的《[SpringMVC 视频教程](#)》，着实有一种醍醐灌顶的感觉，整个路子瞬间通了，开发速度指数型上涨。现在把开发过程中的一些相关经验贴出来。

访问 GitHub 下载最新源码：<https://github.com/gaussic/SpringMVCDemo>

一、相关环境

- IntelliJ IDEA 15.0.4 Ultimate
- Tomcat 7.0.68
- JDK 1.7.0_80
- Spring 3.2.0
- MySql 5.7
- Maven 3.3.9

- Bootstrap 3.3.5

以上是我要做的这个 demo 所需要的东西，当然有些是可选的，版本也是可控的。比如说如果你用不惯 Maven 的话可以自行去官网下载 jar 包然后导入自己的工程中，如果想要学习下 Maven 可以看看《[Maven 视频教程](#)》（偶然找到，这个老师做的视频都挺好，推荐以下），不用完全的去学习 Maven，懂大概意思后再去找找 IntelliJ IDEA 如何配置 maven 的相关文章就足够了。

还有 Bootstrap，纯粹是个人洁癖，不需要这可以去之。

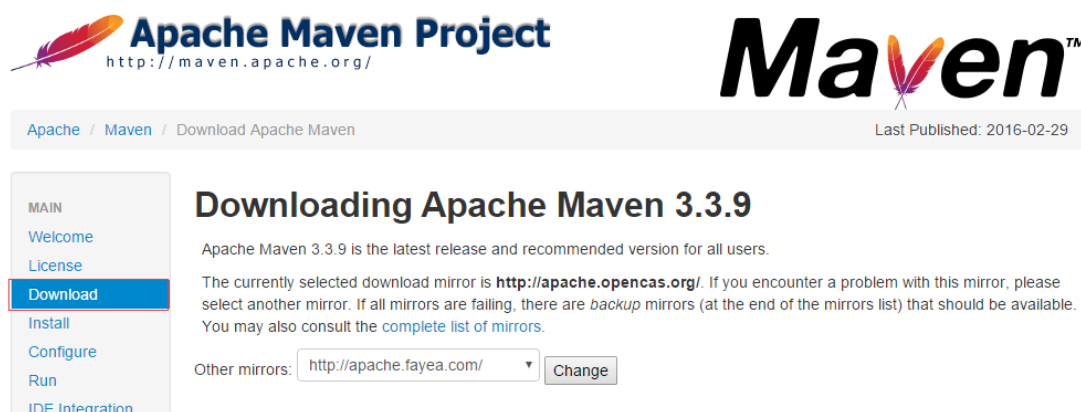
事先声明，请确保 IntelliJ IDEA、Tomcat、MySql 和 JDK 都已经安装好。Maven 和 Bootstrap 的话能有则有。前者为了让导包更容易，后者为了让页面更美观。此外，由于 jdk 以及 mysql 的安装网上已经有了很多教程，在此为节省篇幅不做介绍。废话不多说，正式开始。

二、本地 Maven 与 Tomcat 的安装

注：如果使用 IntelliJ IDEA 集成的 maven 3.0.5 的话，可以忽略此步安装。

1、下载并安装本地 maven

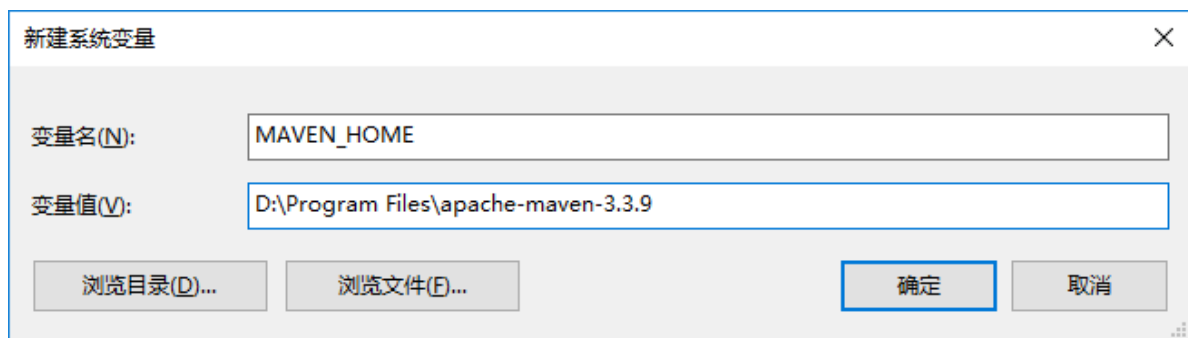
点击“[Apache-Maven 官方网站](#)”进入官网，点击左侧 Download 选项：



进入了下载页面，往下拉可发现当前版本是 3.3.3，点击下面红框中的 [apache-maven-3.3.9-bin.zip](#) 就可下载，下载后解压缩到相应目录下：

	Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.3.9-bin.tar.gz	apache-maven-3.3.9-bin.tar.gz.md5	apache-maven-3.3.9-bin.tar.gz.asc
Binary zip archive	apache-maven-3.3.9-bin.zip	apache-maven-3.3.9-bin.zip.md5	apache-maven-3.3.9-bin.zip.asc
Source tar.gz archive	apache-maven-3.3.9-src.tar.gz	apache-maven-3.3.9-src.tar.gz.md5	apache-maven-3.3.9-src.tar.gz.asc
Source zip archive	apache-maven-3.3.9-src.zip	apache-maven-3.3.9-src.zip.md5	apache-maven-3.3.9-src.zip.asc

新增系统变量 MAVEN_HOME : 即 MAVEN 安装目录 :



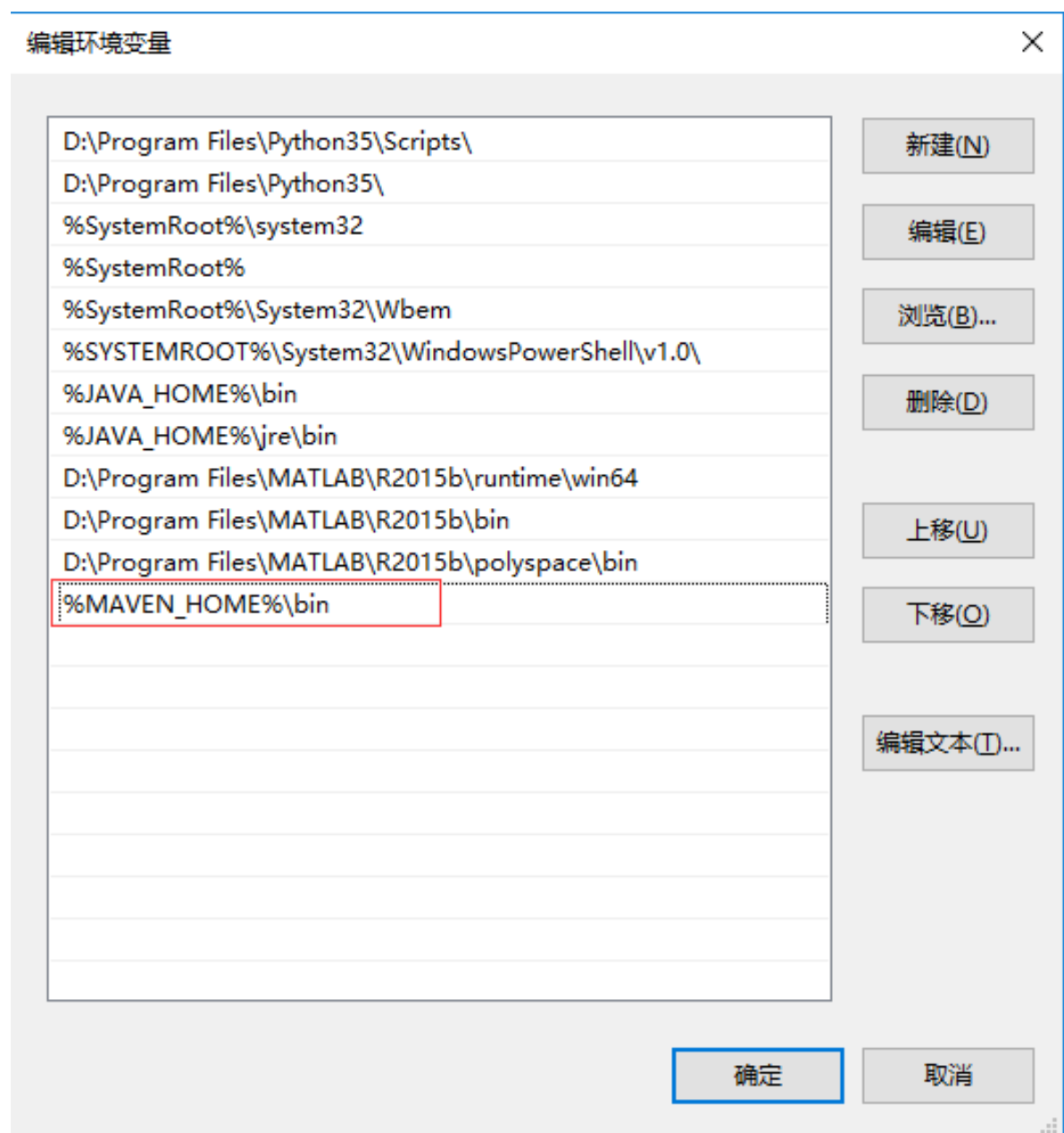
新建系统变量

变量名(N): MAVEN_HOME

变量值(V): D:\Program Files\apache-maven-3.3.9

浏览目录(D)... 浏览文件(F)... 确定 取消

在 Path 中加入 : %MAVEN_HOME%\bin;



在 cmd 中输入 `mvn -v`，若显示如下，则说明本地 maven 配置完成：

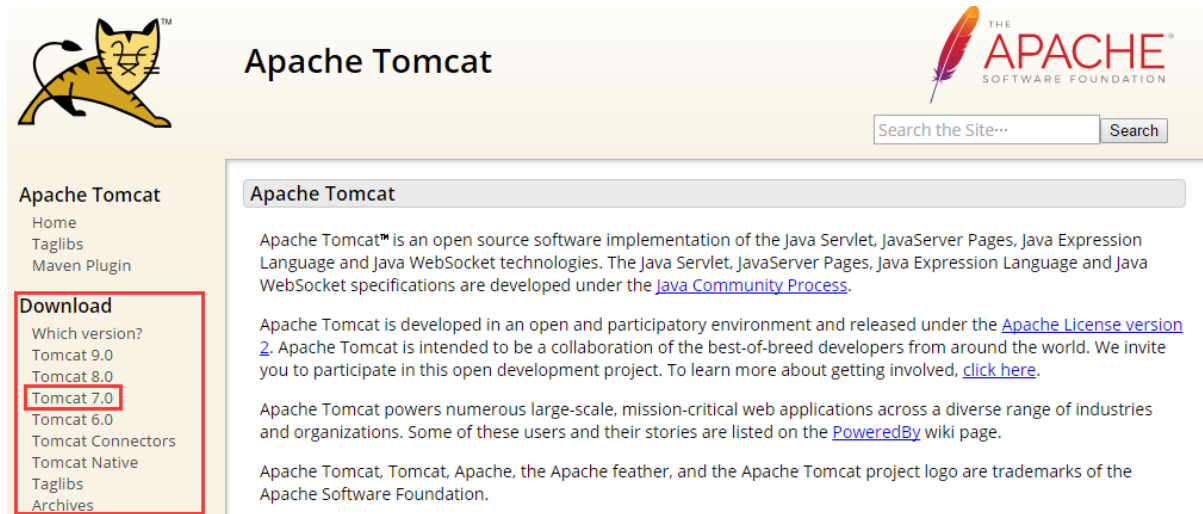
```
管理员: 命令提示符
Microsoft Windows [版本 10.0.10586]
(c) 2015 Microsoft Corporation。保留所有权利。

C:\Windows\system32>mvn -v
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-11T00:41:47+08:00)
Maven home: D:\Program Files\apache-maven-3.3.9\bin\..
Java version: 1.7.0_80, vendor: Oracle Corporation
Java home: D:\Program Files\Java\jdk1.7.0_80\jre
Default locale: zh_CN, platform encoding: GBK
OS name: "windows 8.1", version: "6.3", arch: "amd64", family: "windows"

C:\Windows\System32>
```

2、下载并安装本地 Tomcat

进入 [Tomcat 官网](#)，点击左侧 Download 的 Tomcat7.0，进入 Tomcat 的下载页面：



64 位 Windows 版本下载 [64-bit Windows zip \(pgp, md5, sha1\)](#)，解压到所需目录下：

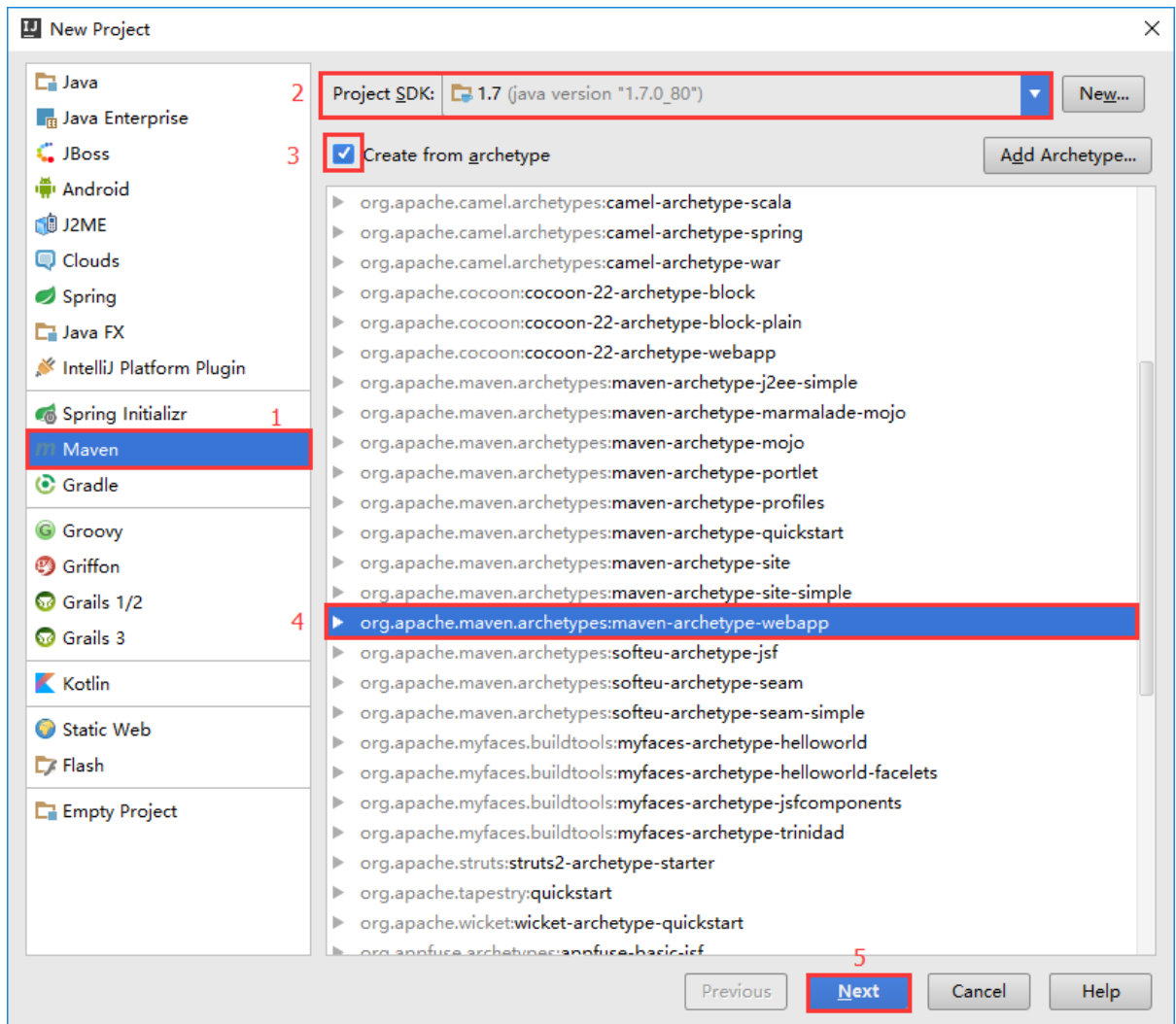
- Core:
 - [zip \(pgp, md5, sha1\)](#)
 - [tar.gz \(pgp, md5, sha1\)](#)
 - [32-bit Windows zip \(pgp, md5, sha1\)](#)
 - [64-bit Windows zip \(pgp, md5, sha1\)](#)
 - [64-bit Itanium Windows zip \(pgp, md5, sha1\)](#)
 - [32-bit/64-bit Windows Service Installer \(pgp, md5, sha1\)](#)

解压后到\bin\目录下运行 startup.bat，如图下所示，如果出现 Server startup in xxxx ms 说明 Tomcat 安装成功。

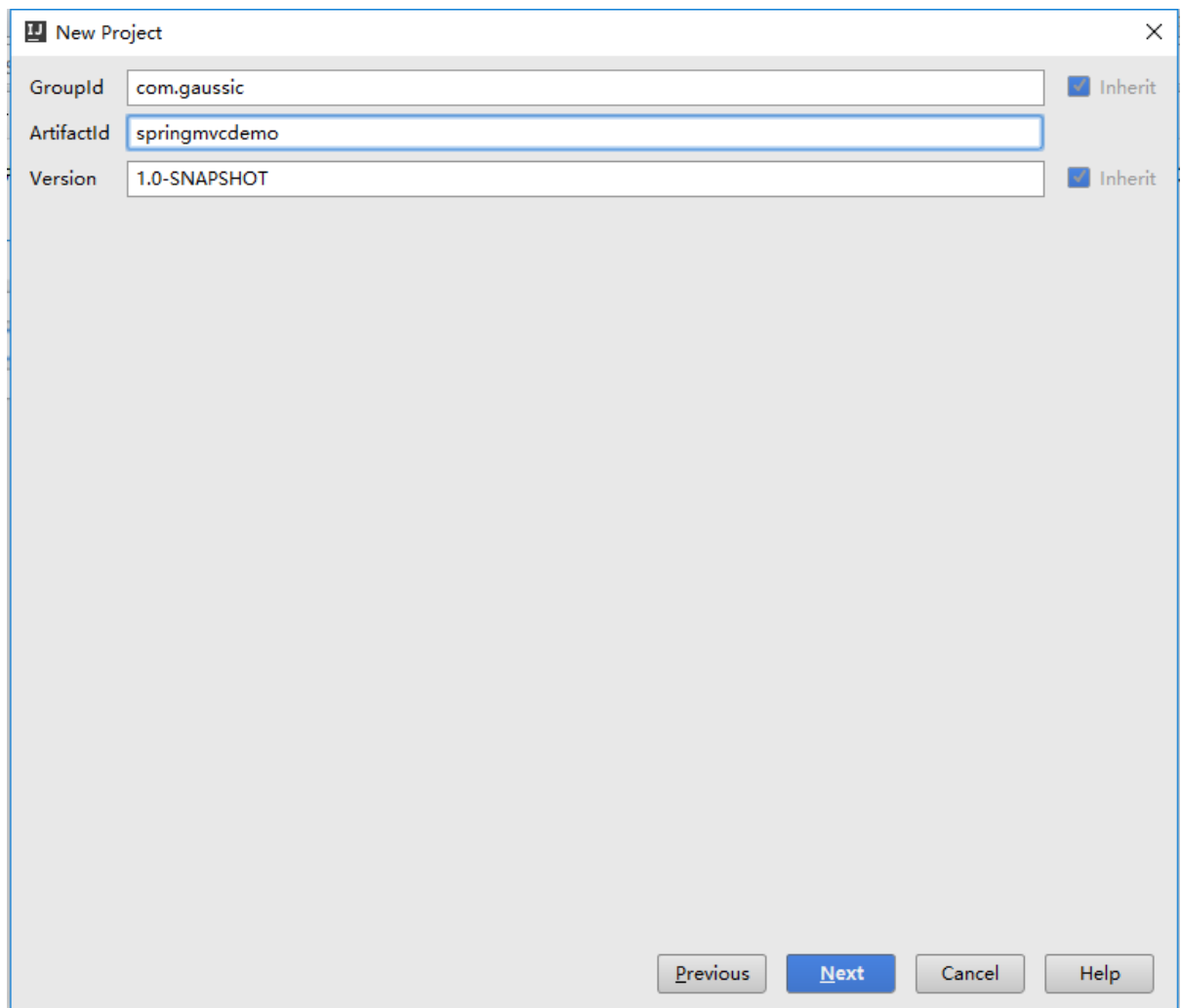
```
Tomcat
信息: Deploying web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\docs
三月 08, 2016 9:52:44 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deployment of web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\docs has finished in 685 ms
三月 08, 2016 9:52:44 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\examples
三月 08, 2016 9:52:45 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deployment of web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\examples has finished in 475 ms
三月 08, 2016 9:52:45 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\host-manager
三月 08, 2016 9:52:45 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deployment of web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\host-manager has finished in 97 ms
三月 08, 2016 9:52:45 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\manager
三月 08, 2016 9:52:45 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deployment of web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\manager has finished in 92 ms
三月 08, 2016 9:52:45 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\ROOT
三月 08, 2016 9:52:45 上午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deployment of web application directory D:\Program Files\apache-tomcat-7.0.68\webapps\ROOT has finished in 116 ms
三月 08, 2016 9:52:45 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-apr-8080"]
三月 08, 2016 9:52:45 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-apr-8009"]
三月 08, 2016 9:52:45 上午 org.apache.catalina.startup.Catalina start
信息: Server startup in 1569 ms
```

三、创建 Maven Web 项目

前面说了这么多，差不多基本的东西都保障了（前提保证你已经安装了 jdk）。现在进入正题，如何来创建一个 Web 项目。对于不使用 Maven 的开发者，可以直接建一个简单的 Web 项目。使用 Maven 的话，请按照图进行操作。



菜单 File->New Project 可进入上图界面，首先选择左边栏 Maven，再配置 JDK（一般如果之前添加了 JDK 的话会自动填充，如未添加的话点击旁边的 New 将 JDK 目录导入即可）。勾选 “Create from archetype”，然后选中 4 处蓝色位置 webapp，点 Next，进入如下界面：



New Project

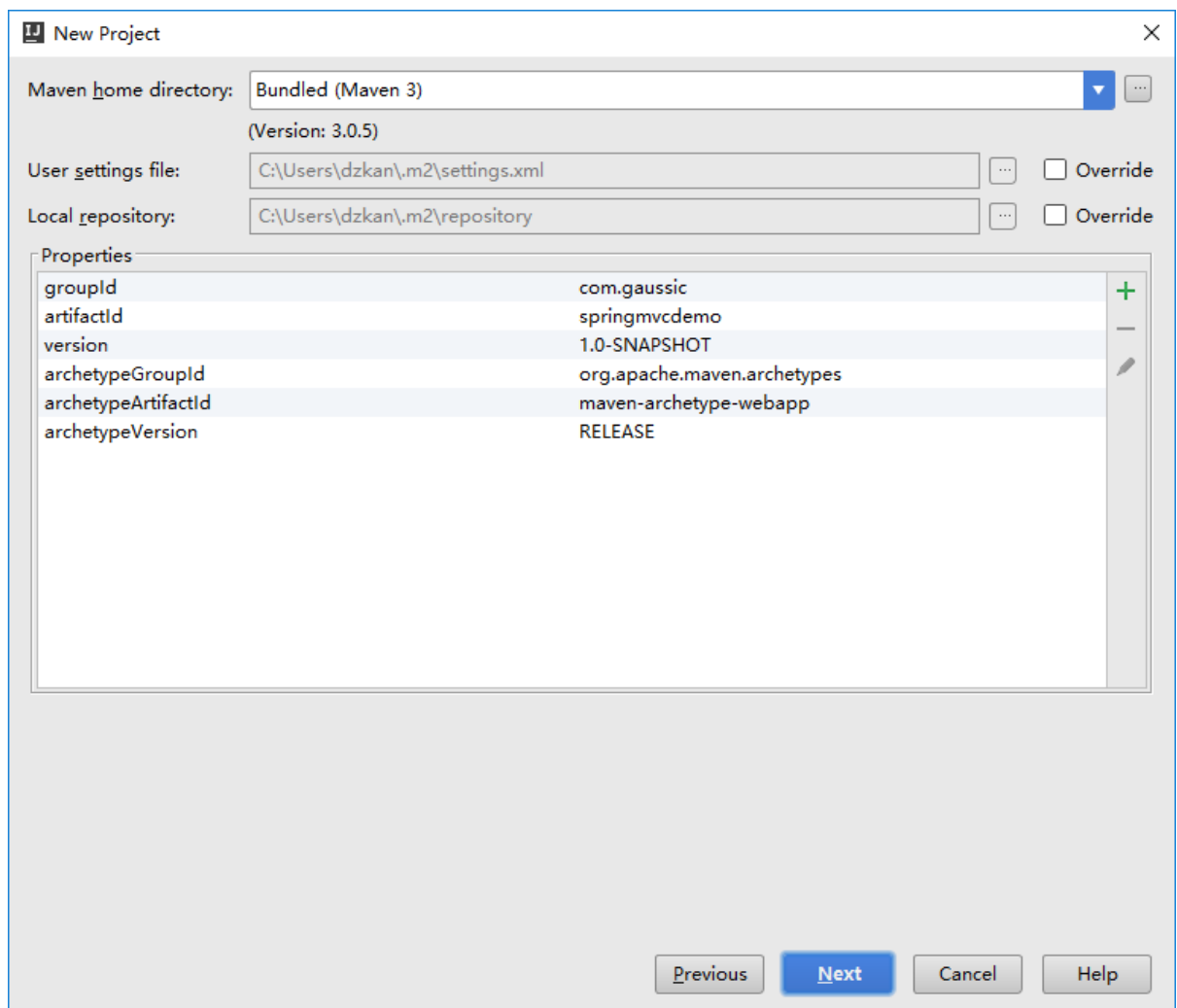
GroupId ☒ Inherit

ArtifactId ☒ Inherit

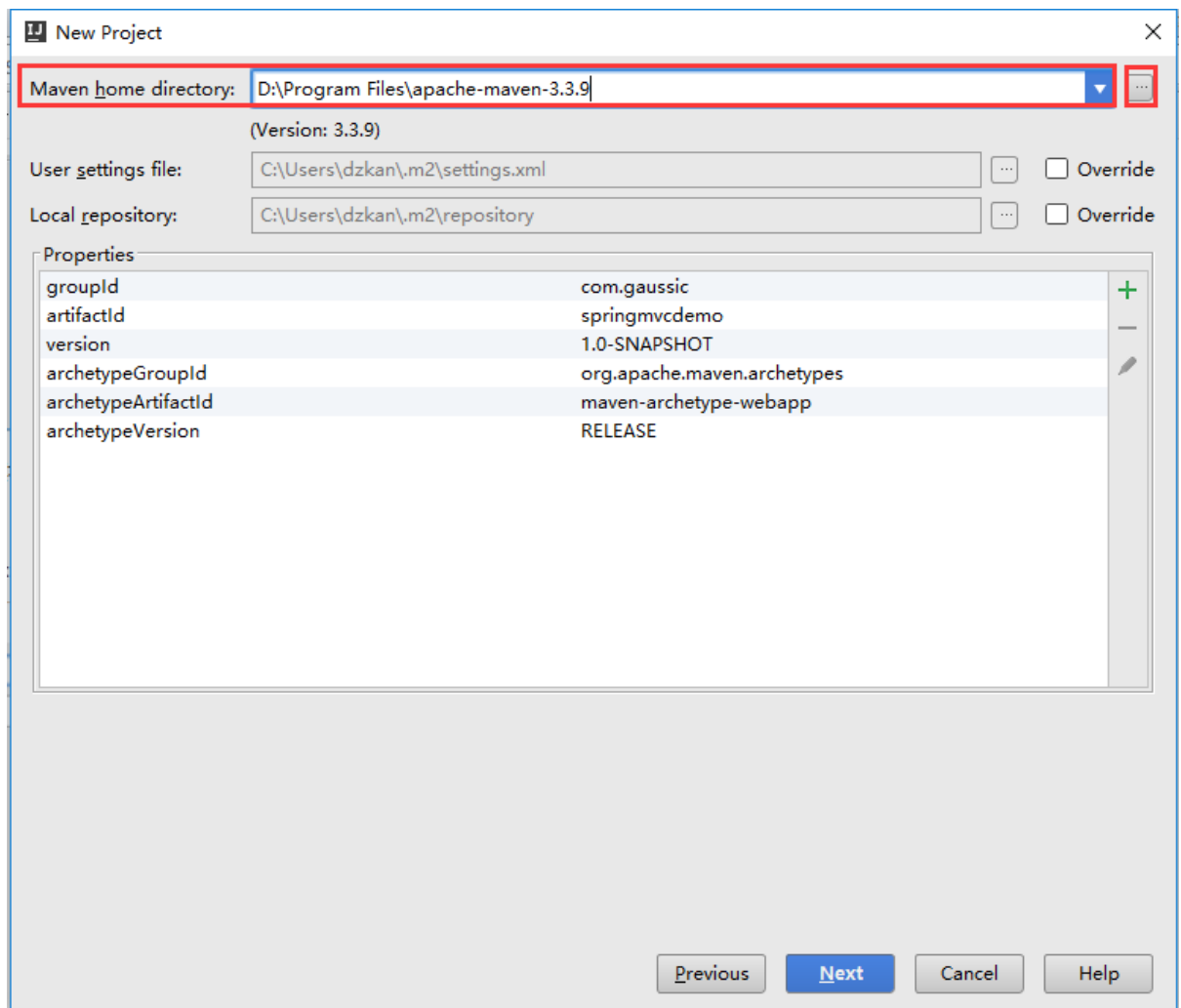
Version ☒ Inherit

这里需要填写 GroupId 和 ArtifactId 还有 Version，这三个属性目的是标识你的项目的唯一性，比如 Tomcat 的 GroupId 是 org.apache，即它是 apache 组织的项目，ArtifactId 是 tomcat，项目名为 tomcat，而我当前使用的 Version 是 7.0.68。这些只在发布时有用，在此可以随便填写，填好后点 Next，到如下界面。

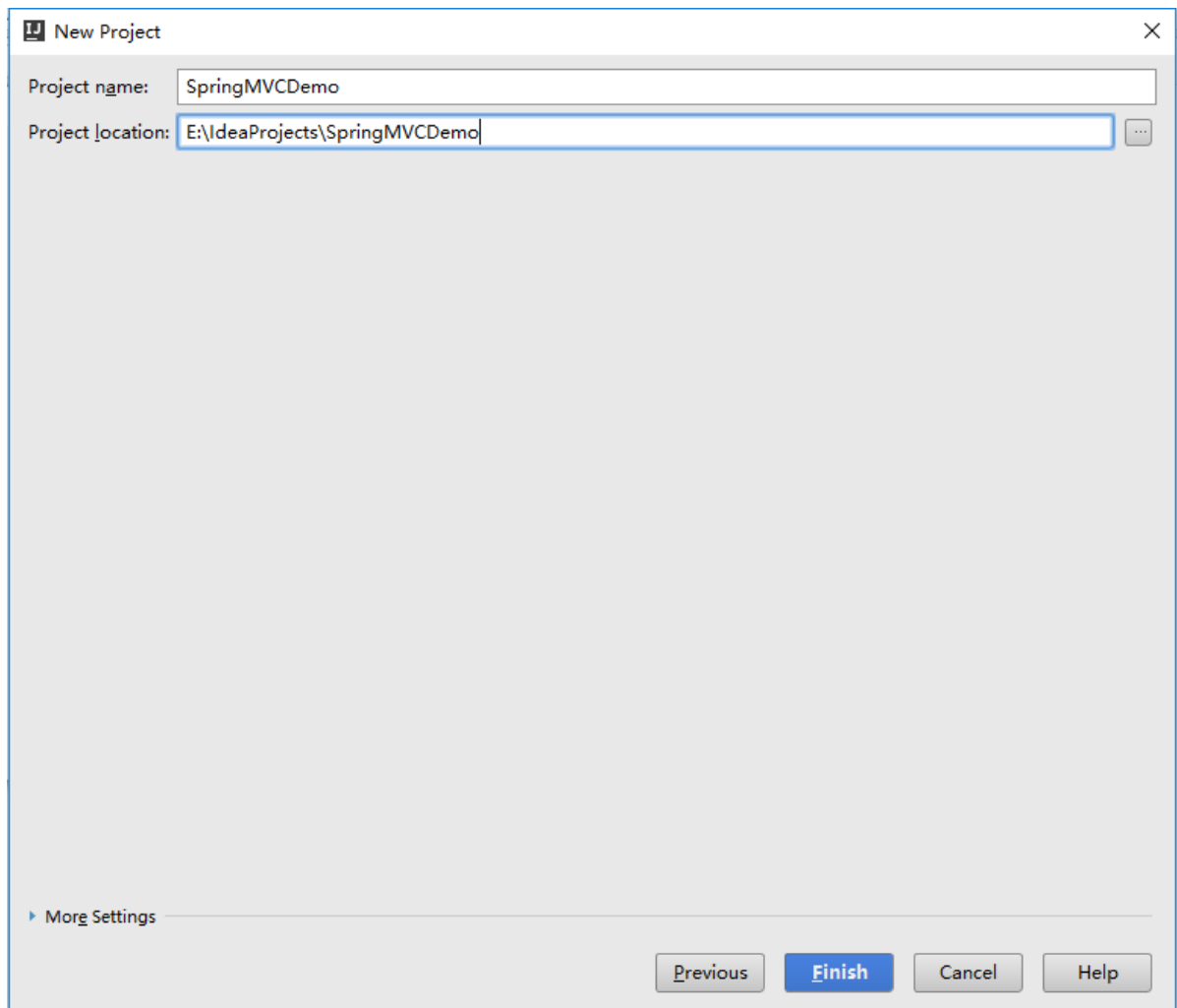
打开 Maven home directory，可以发现 IntelliJ IDEA 已经集成了 Maven 2 和 Maven 3 两个版本，如果使用默认集成的 maven 的话，选择 Buldled(Maven 3)，直接点击 Next。



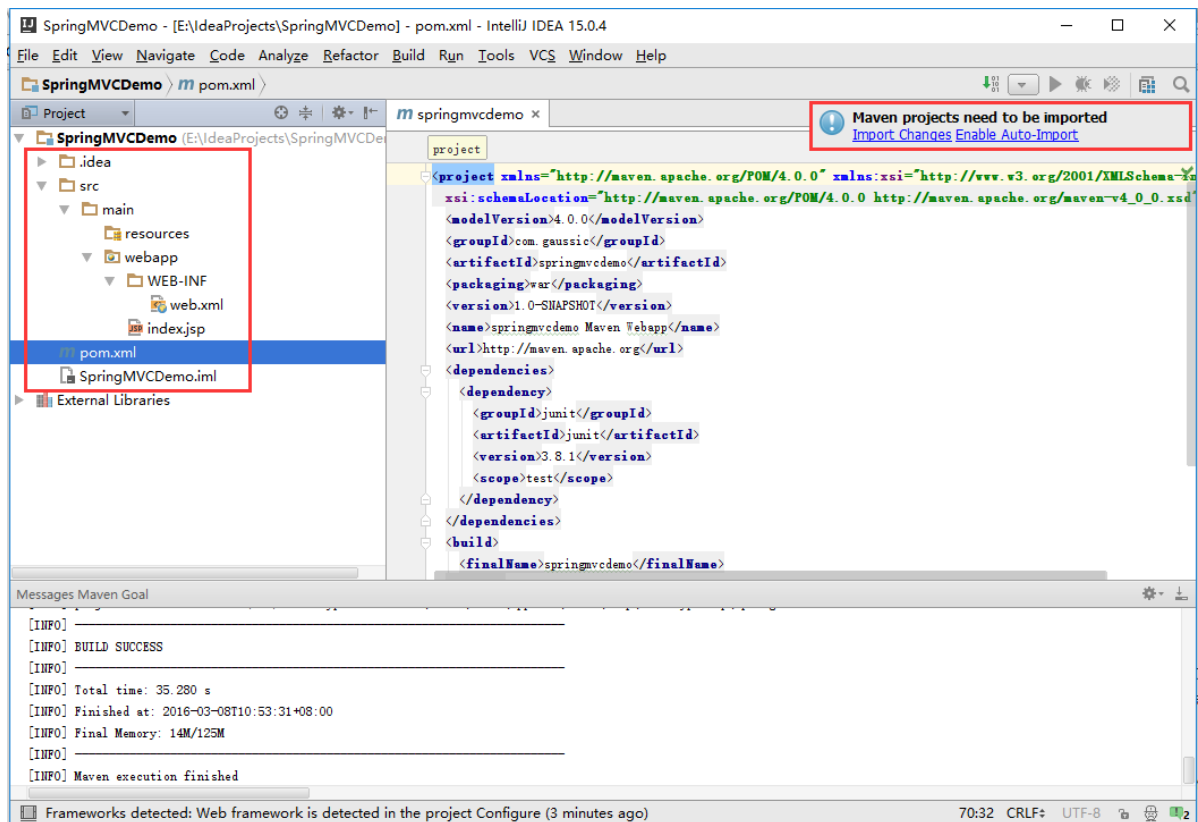
我们也可以导入本地新安装的较新的 Maven 版本，点击蓝色箭头右边的 ... 按钮将 Maven 路径导入即可，点击 Next：



填写项目名，选择项目保存路径，点击 Finish：



进入如下界面，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，经验发现用较新版本的 maven 项目生成更快，使用 IDEA 集成的 maven 可能会等待很长一段实践。



左边红框中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 `resources` 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，这一点在后面详述。

四、Maven 自动导入 jar 包

既然我们要用 `SpringMVC` 开发，那肯定少不了 `SpringMVC` 的相关 `jar` 包。如果不使用 `Maven` 的话，那就需要去官网下载相关的 `jar` 包，然后导入到项目中。现在使用 `maven` 的话，就不需要上网找 `jar` 包了。具体容我一一道来。

`Maven` 所做的工作其实很简单，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](#)，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大家面前），maven 就是靠它来定义需求的，代码如下：

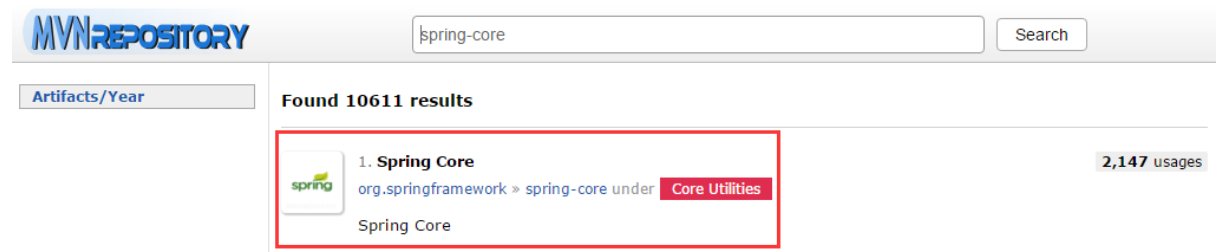
```
?  
5 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http:  
6 //www.w3.org/2001/XMLSchema-instance"  
7 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http:  
8 p://maven.apache.org/maven-v4_0_0.xsd">  
9 <modelVersion>4.0.0</modelVersion>  
10 <groupId>com.gaussic</groupId>  
11 <artifactId>springmvcdemo</artifactId>  
12 <packaging>war</packaging>  
13 <version>1.0-SNAPSHOT</version>  
14 <name>springmvcdemo Maven Webapp</name>  
15 <url>http://maven.apache.org</url>  
16 <dependencies>  
17 <dependency>  
18 <groupId>junit</groupId>  
19 <artifactId>junit</artifactId>  
20 <version>3.8.1</version>  
21 <scope>test</scope>  
22 </dependency>  
23 </dependencies>  
24 <build>  
25 <finalName>springmvcdemo</finalName>  
26 </build>  
27 </project>
```

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 <dependency>，定义在 <dependencies> 中。在每个 <dependency> 中，你需要提供的是所需 jar 包的 groupId、artifactId、version 这三个必要信息。比如上面我们看到引入可一个 junit 包，格式如下：

?

```
3         <dependency>
4             <groupId>junit</groupId>
5             <artifactId>junit</artifactId>
6             <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
```

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。那么如何查看这些 jar 包的 3 个信息呢，可能刚接触是开发者还不是很熟悉，这个时候就需要查阅仓库了。比如我们需要引入 Spring 核心 jar 包 spring-core，打开 [Maven Repository](#)，搜索 spring-core，进入如下界面：



点击进入红框选中的 Spring Core，如下所示，可以看到各版本的使用情况：

Spring Core



Spring Core

org.springframework » spring-core under

Core Utilities

2,147 usages

Spring Core

Tags: [framework](#) [spring](#)

	Version	Usages	Type	Date
4.2.x	4.2.5.RELEASE	25	release	(Feb, 2016)
	4.2.4.RELEASE	57	release	(Dec, 2015)
	4.2.3.RELEASE	82	release	(Nov, 2015)
	4.2.2.RELEASE	64	release	(Oct, 2015)
	4.2.1.RELEASE	59	release	(Sep, 2015)
	4.2.0.RELEASE	49	release	(Jul, 2015)
	4.1.9.RELEASE	62	release	(Dec, 2015)

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

Spring Core » 4.2.5.RELEASE



Spring Core

[org.springframework](#) » [spring-core](#) » 4.2.5.RELEASE under **Core Utilities**

Spring Core

Artifact	Download (JAR) (1.1 MB)
POM File	View
Date	(Feb 25, 2016)
HomePage	https://github.com/spring-projects/spring-framework
Organization	Spring IO
Issue Tracker	https://jira.springsource.org/browse/SPR

[Maven](#)

[Ivy](#)

[Grape](#)

[Gradle](#)

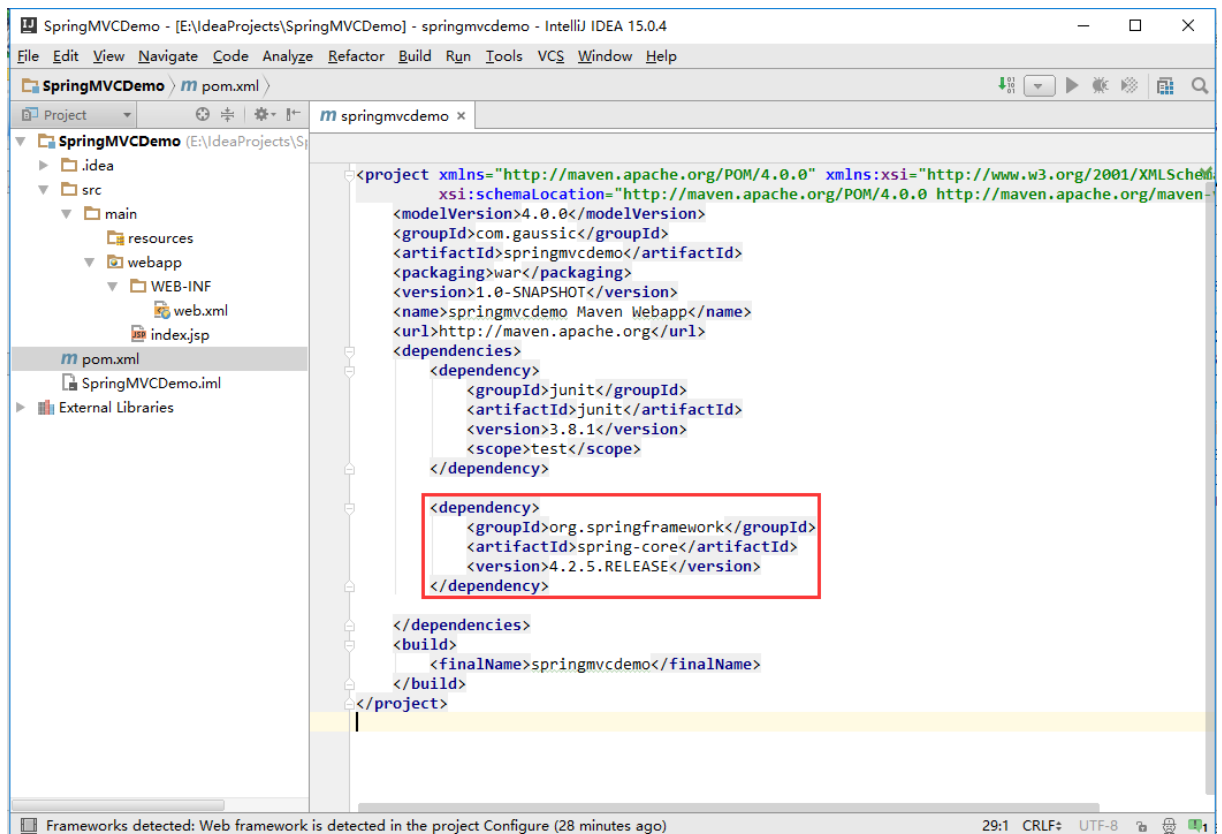
[Buildr](#)

[SBT](#)

[Leiningen](#)

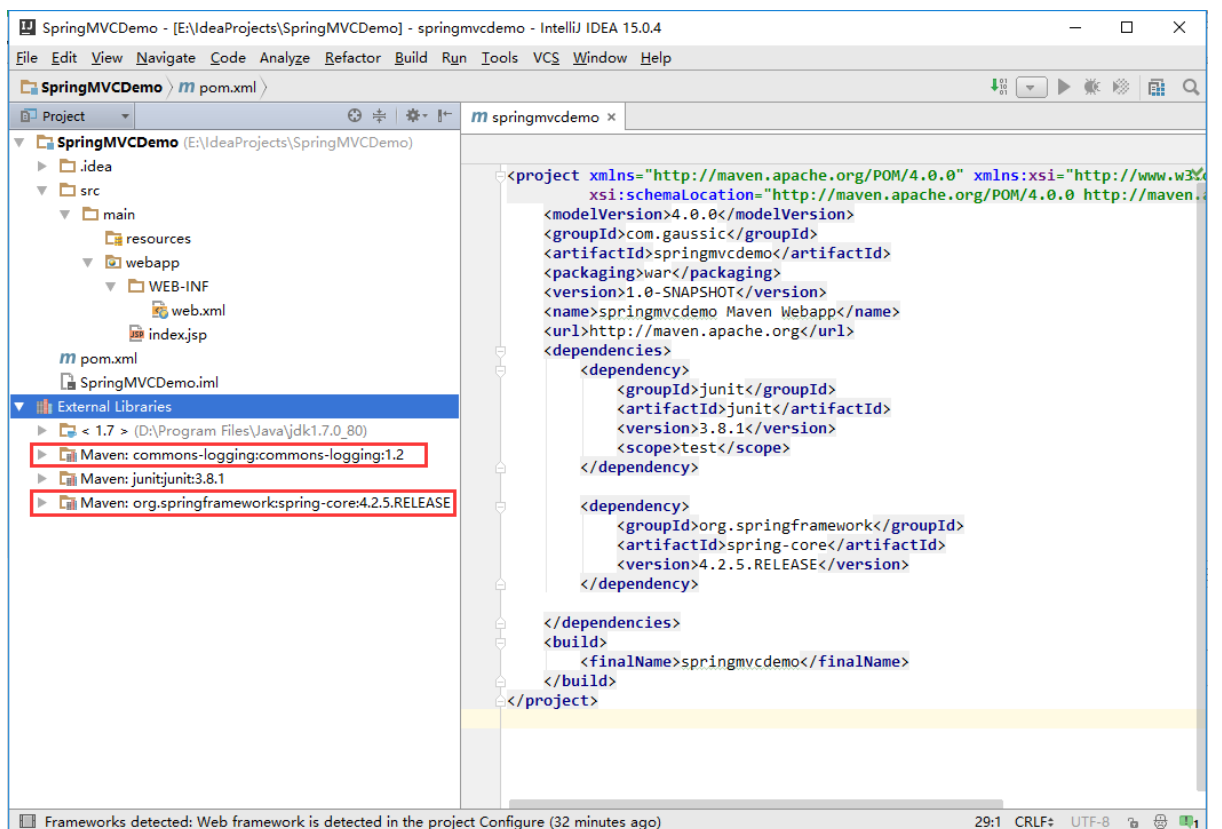
```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>4.2.5.RELEASE</version>
</dependency>
```

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，但是它导入了两个 jar 包，也就是说，导入某个 jar 包时，与它密切相关的 jar 包也会同时被导入进来。

除了 spring-core，我还要 spring-context，复制 spring-core 的<dependency>，将 spring-core 改为 spring-context，如下：

?

1

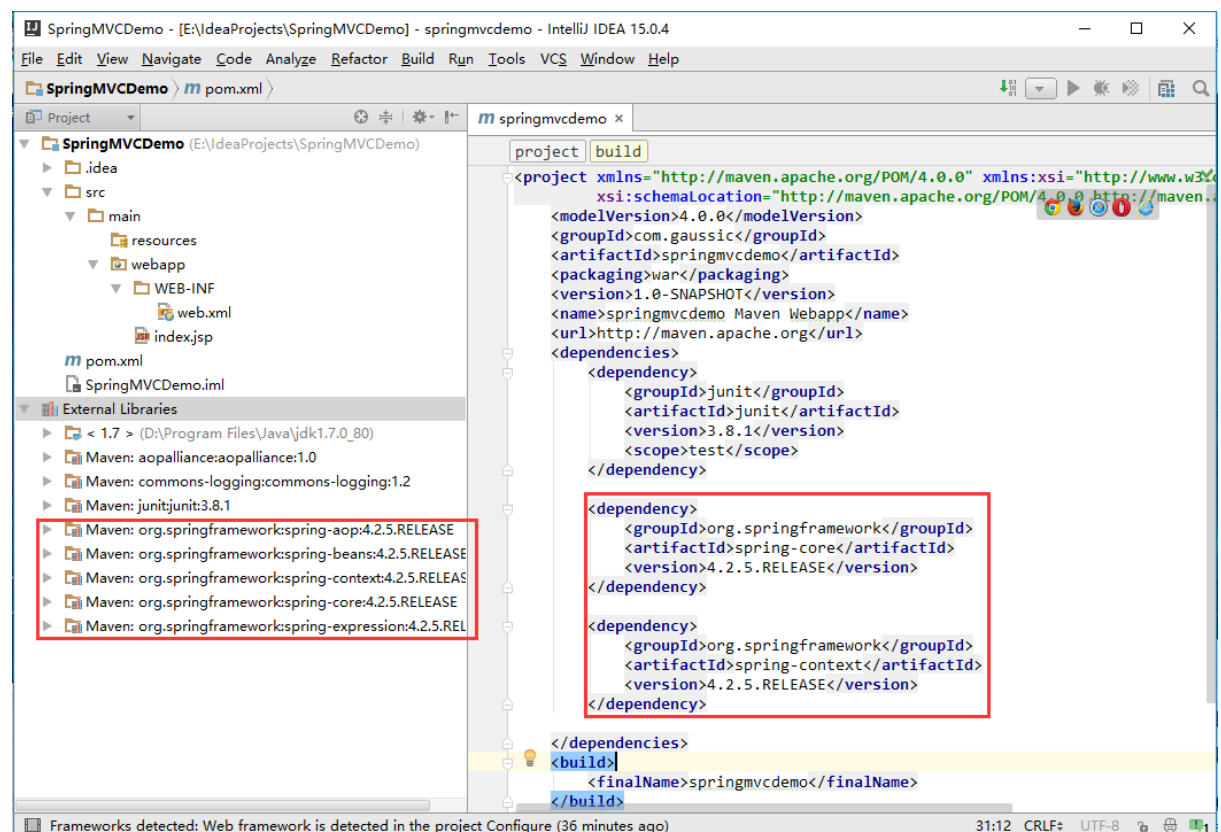
2

3

4

5

下载完成后，查看 External Libraries，会不会发现，瞬间导入了好多 jar 包（当然不是瞬间，这得看你的网速了）呢：



这就是 Maven 的强大之处，如果你需要使用 SpringMVC 开发网站的话，只需记住几个重要的包的名字，就可以轻松将所有包导入项目中。

长话短说，现在我们要进行 SpringMVC 的开发，请把你的 pom.xml 变成下面的样子，当然不要改你的 groupId 等信息（从 modelVersion 到 url 都不要动）：

```
2           <properties>
3       <spring.version>3.2.0.RELEASE</spring.version>
4       <spring-data.version>1.2.0.RELEASE</spring-data.version>
           </properties>
```

请在 <dependencies> 中加入以下依赖：

```
?
1
2           <dependency>
3       <groupId>org.springframework</groupId>
4       <artifactId>spring-web</artifactId>
5       <version>${spring.version}</version>
6       </dependency>
7
8           <dependency>
9       <groupId>javax.servlet</groupId>
10      <artifactId>servlet-api</artifactId>
11      <version>2.5</version>
12      </dependency>
13
14      <dependency>
15      <groupId>javax.servlet.jsp</groupId>
16      <artifactId>jsp-api</artifactId>
17      <version>2.1</version>
18      <scope>provided</scope>
19      </dependency>
20
21      <dependency>
22      <groupId>org.springframework</groupId>
```

```

14         <artifactId>spring-webmvc</artifactId>
15         <version>${spring.version}</version>
16         </dependency>
17
18         <dependency>
19         <groupId>org.springframework</groupId>
20         <artifactId>spring-test</artifactId>
21         <version>${spring.version}</version>
22         <scope>test</scope>
23         </dependency>
24
25         <dependency>
26         <groupId>jstl</groupId>
27         <artifactId>jstl</artifactId>
28         <version>1.2</version>
29         </dependency>
30
31         <dependency>
32         <groupId>org.springframework.data</groupId>
33         <artifactId>spring-data-jpa</artifactId>
34         <version>${spring-data.version}</version>
35         </dependency>
36
37         <dependency>
38         <groupId>org.hibernate.javax.persistence</groupId>
39         <artifactId>hibernate-jpa-2.0-api</artifactId>
40         <version>1.0.0.Final</version>
41         </dependency>
42
43         <dependency>
44         <groupId>org.hibernate</groupId>
45         <artifactId>hibernate-entitymanager</artifactId>
46         <version>3.6.10.Final</version>
47         </dependency>
48
49         <dependency>
50         <groupId>mysql</groupId>
51         <artifactId>mysql-connector-java</artifactId>
52         <version>5.1.34</version>

```

36

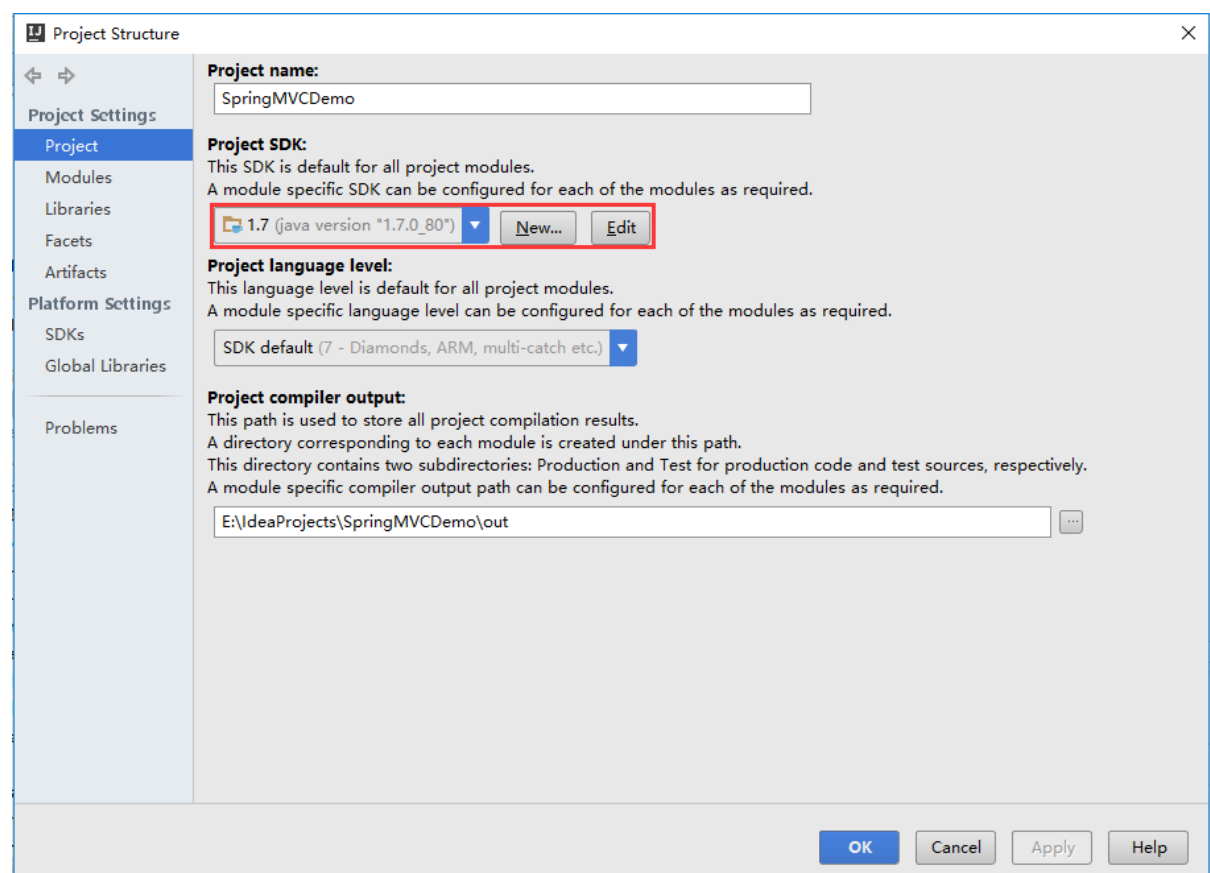
37

38

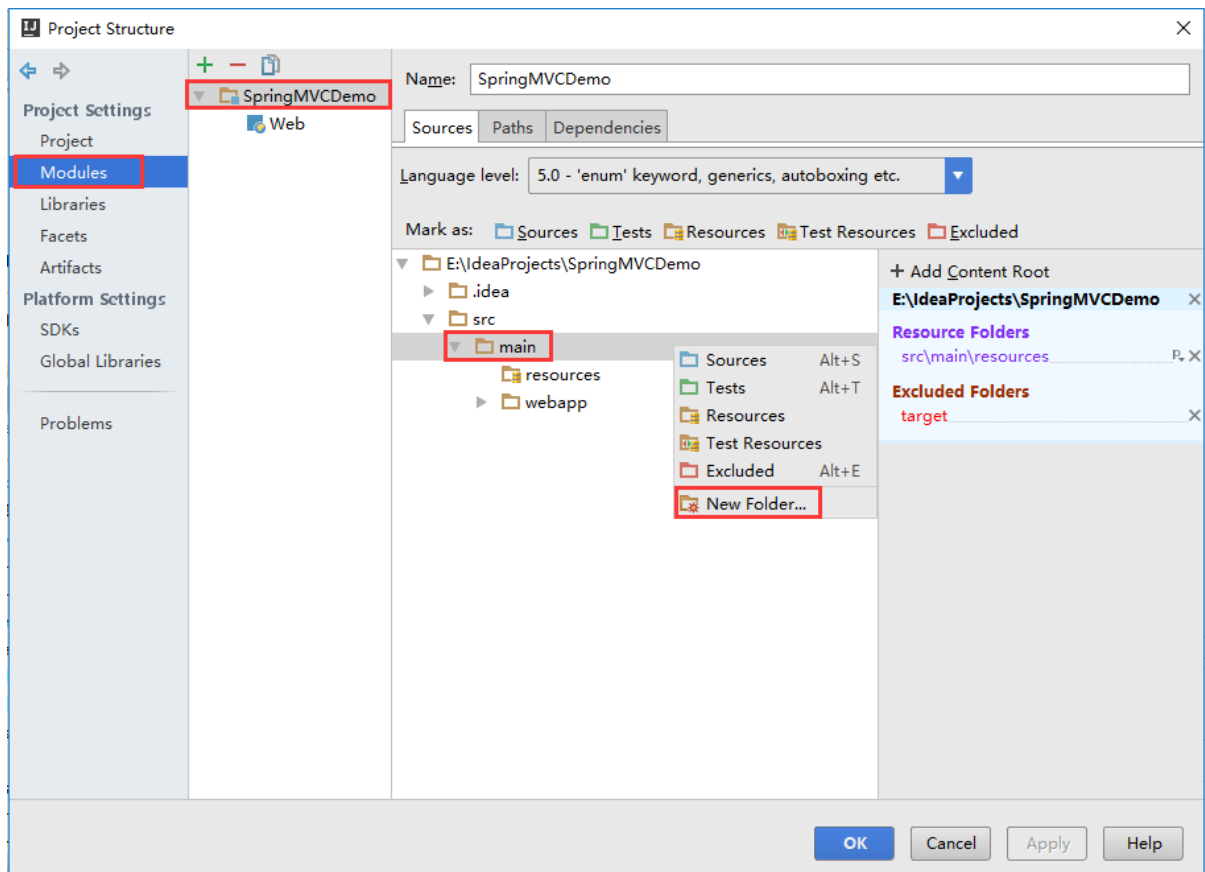
39

40

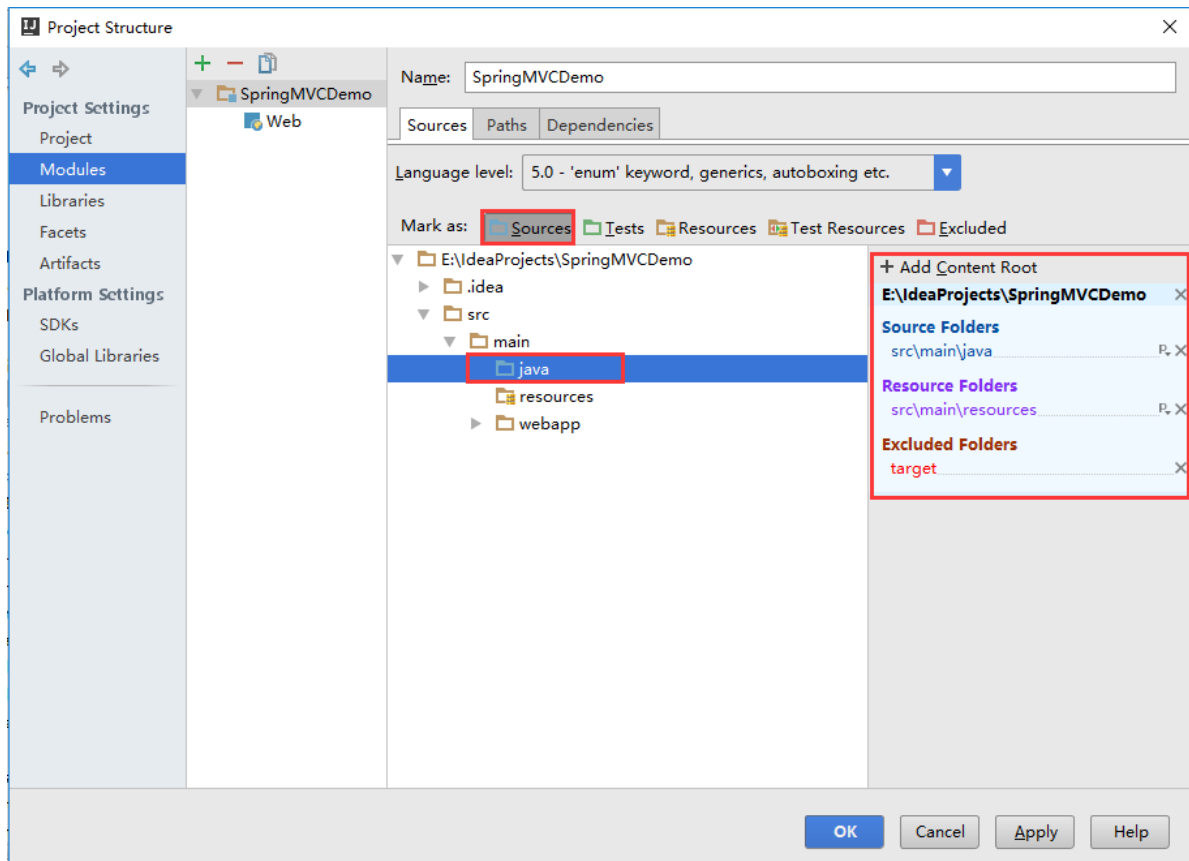
们后面慢慢说。如果不使用 Maven 请自行下载 spring、hibernate、mysql、jstl、javax-servlet、json 等相关 jar 包然后导入到工程中。至此，jar 包的导入就完成了，我们按 `ctrl+alt+shift+s`，或者 `File->Project Structure` 查看一下项目结构，看看有什么问题：



由于之后我们要开始写代码了，先做一些配置，选择 Modules，在 SpringMVCDemo 的 src\main 文件夹中新建一个文件夹，取名为 java：



选中java 文件夹，点击上面的 Make as : Sources，该文件夹就会变成蓝色，用以保存 java 代码，按 OK，结束配置。



五、SpringMVC 框架配置

进行完上面的配置，那就说明现在基本的开发环境已经搭建好了，现在要开始进行 SpringMVC 的网站开发。

1、web.xml 配置

打开 src/main/webapp/WEB-INF 下的 web.xml 文件，稍微更新一下 web.xml 的版本，可以支持更高级的一些语法，如下：

```
2 <?xml version="1.0" encoding="UTF-8"?>
3 <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
4         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
6         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
7         version="3.1">
```

```

7      <display-name>SpringMVCDemo Web Application</display-name>
8
9  </web-app>

```

在<web-app>中加入一个 servlet :

?

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4          xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5          http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6          version="3.1">
7      <display-name>SpringMVCDemo Web Application</display-name>
8
9      <servlet>
10         <servlet-name>mvc-dispatcher</servlet-name>
11         <servlet-
12            class>org.springframework.web.servlet.DispatcherServlet</servl
13            et-class>
14         <load-on-startup>1</load-on-startup>
15     </servlet>
16
17     <servlet-mapping>
18         <servlet-name>mvc-dispatcher</servlet-name>
19         <url-pattern>/</url-pattern>
20     </servlet-mapping>
21 </web-app>

```

该 servlet 名为 mvc-dispatcher (名称可修改) , 用于拦截请求 (url-pattern 为 / , 说明拦截所有请求) , 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：

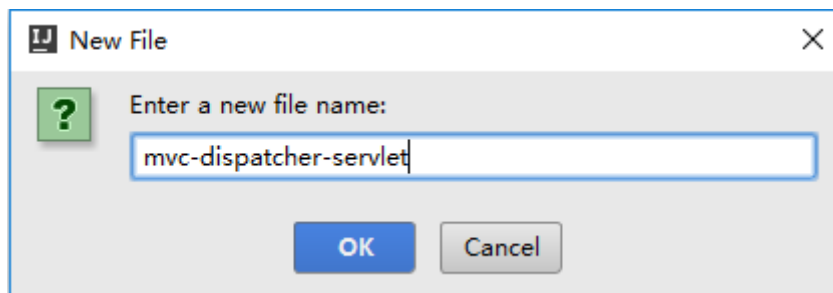
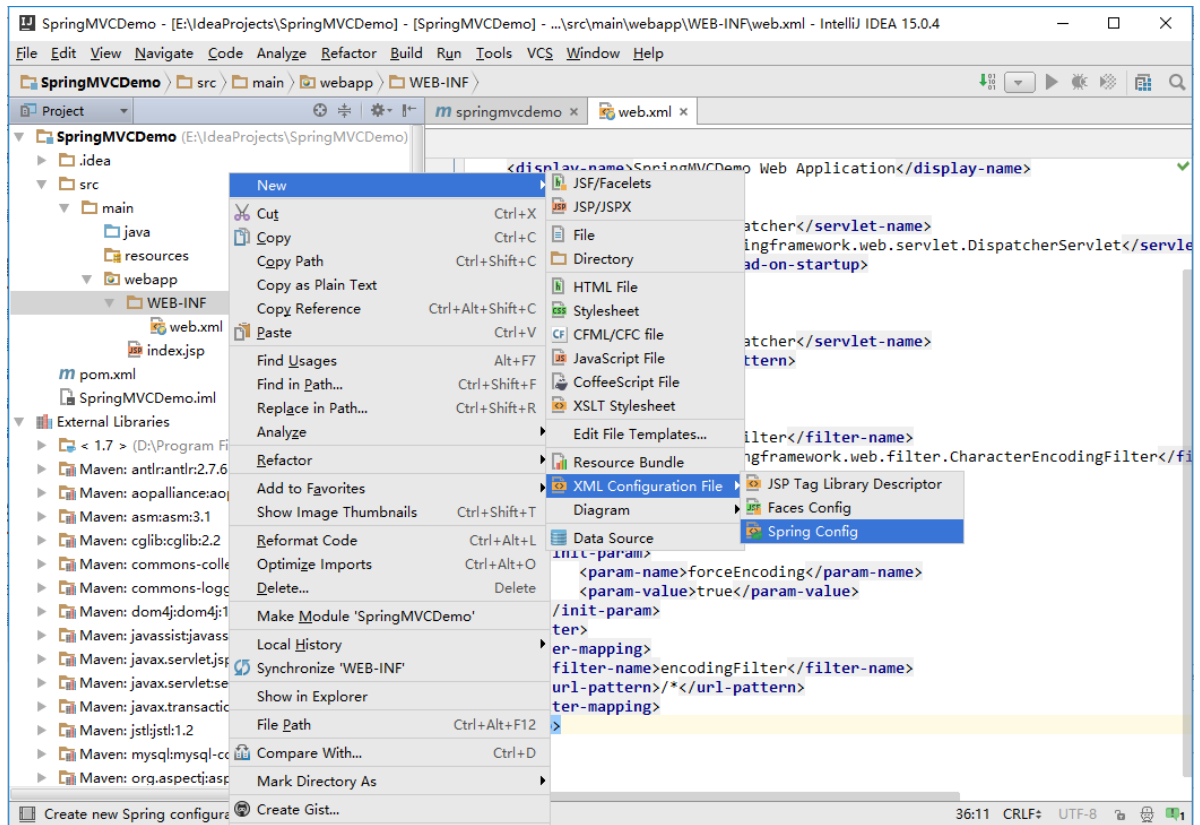
况：

```
1  <filter>
2      <filter-name>encodingFilter</filter-name>
3      <filter-
4      class>org.springframework.web.filter.CharacterEncodingFilter</f
5      ilter-class>
6      <init-param>
7          <param-name>encoding</param-name>
8          <param-value>UTF-8</param-value>
9      </init-param>
10     <init-param>
11         <param-name>forceEncoding</param-name>
12         <param-value>true</param-value>
13     </init-param>
14 </filter>
15 <filter-mapping>
16     <filter-name>encodingFilter</filter-name>
17     <url-pattern>/*</url-pattern>
18 </filter-mapping>
```

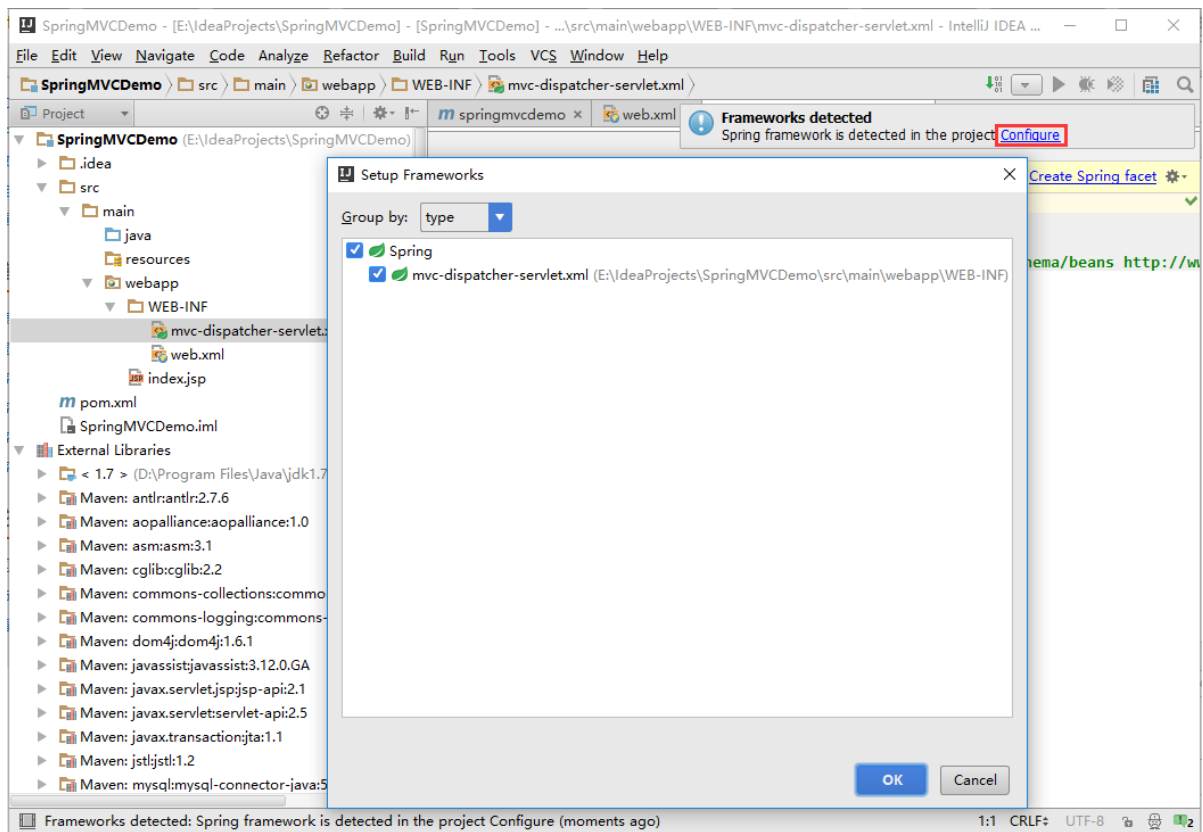
至此，web.xml 配置完毕。

2、xxx-servlet.xml 配置

在配置完 web.xml 后，需在 web.xml 同级目录下新建 mvc-dispatcher-servlet.xml（-servlet 前面是在 servlet 里面定义的 servlet 名）：



新建该 xml 文件后，点击右上角的 configure，出现 Setup Frameworks 界面，点击 OK，这样，IntelliJ IDEA 就识别了 SpringMVC 的配置文件：



mvc-dispatcher-servlet.xml 文件如下：

?

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

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

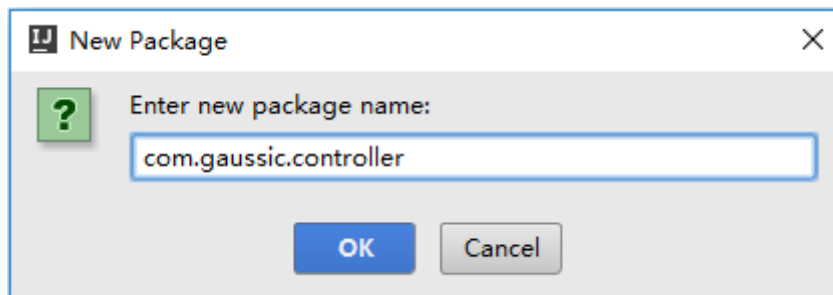
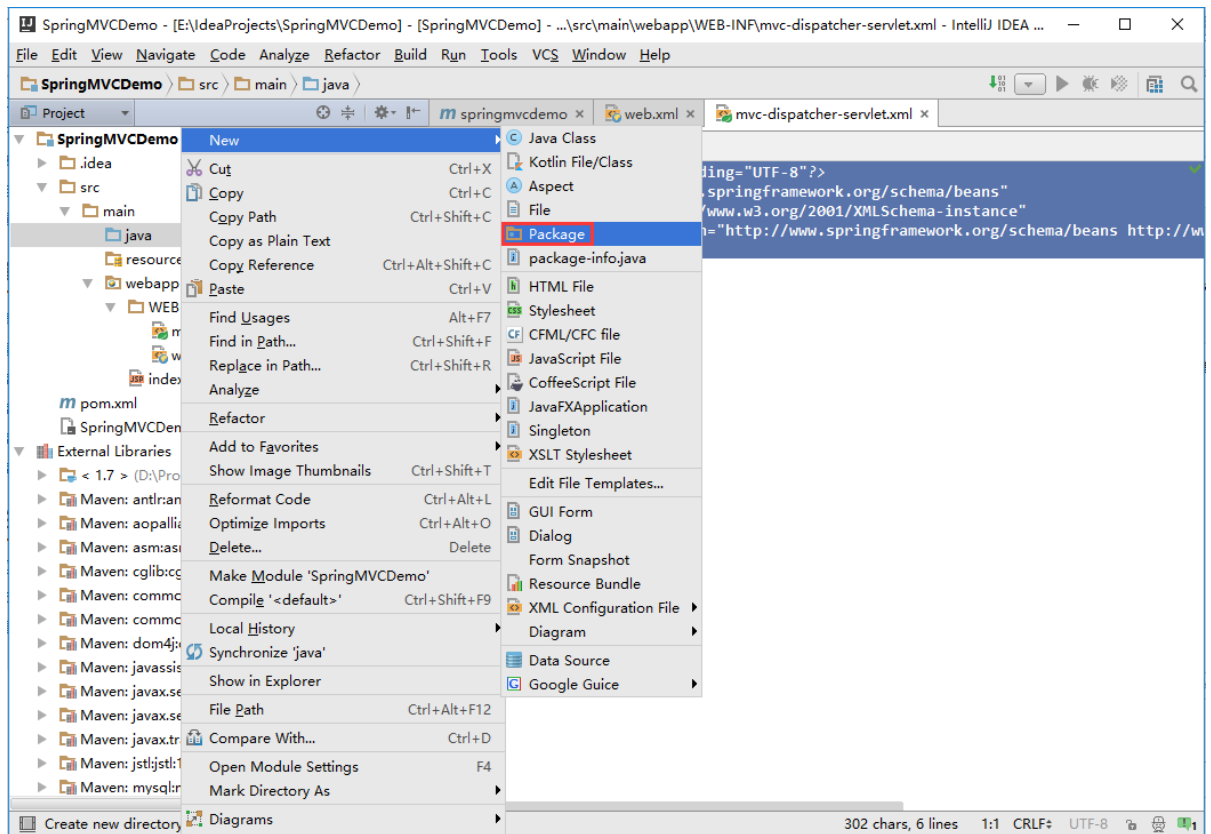
5

```
</beans>
```

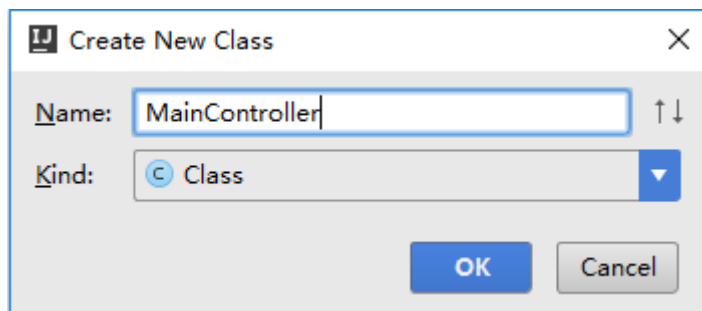
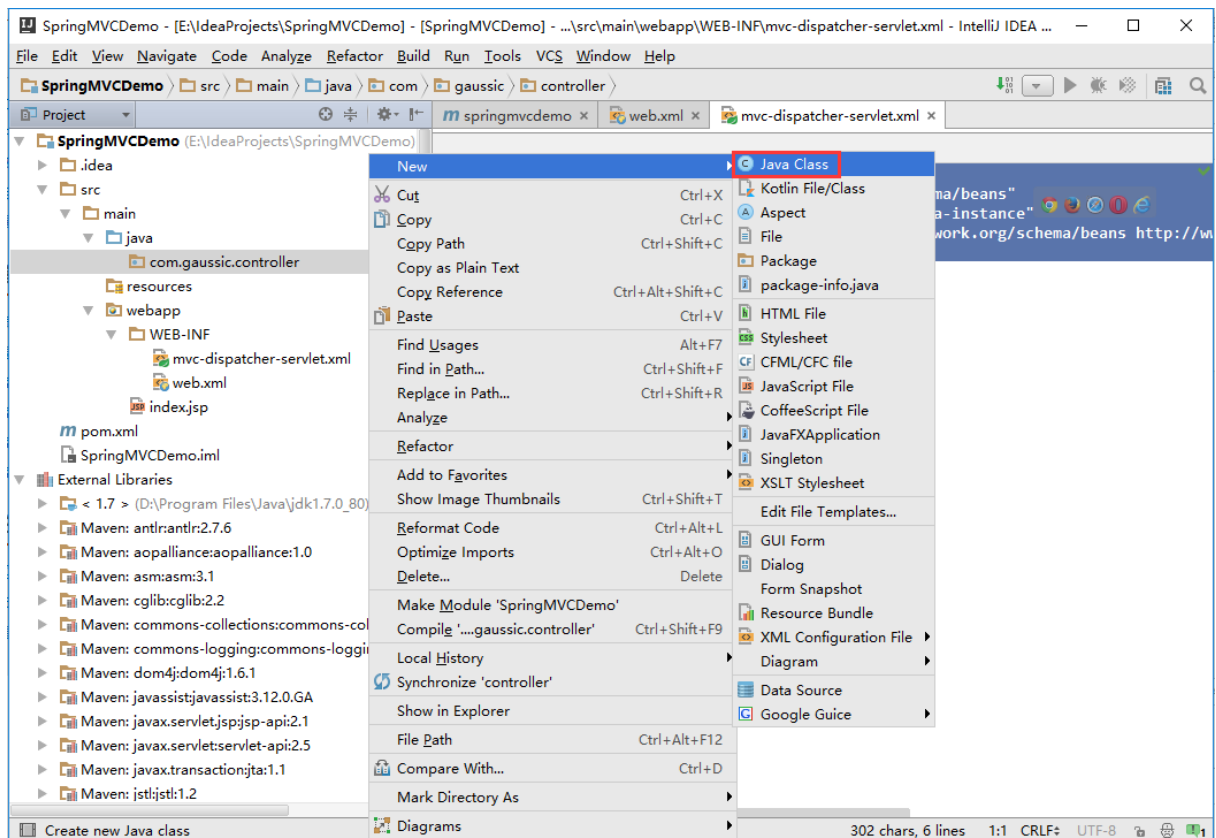
6

MVC 框架有 model、view、controller 三部分组成。model 一般为一些基本的 Java Bean，view 用于进行相应的页面显示，controller 用于处理网站的请求。

在 src\main\java 中新建一个用于保存 controller 的 package：



在 controller 包中新建 java 类 MainController (名称并不固定 , 可任意取) , 并修改如下 :



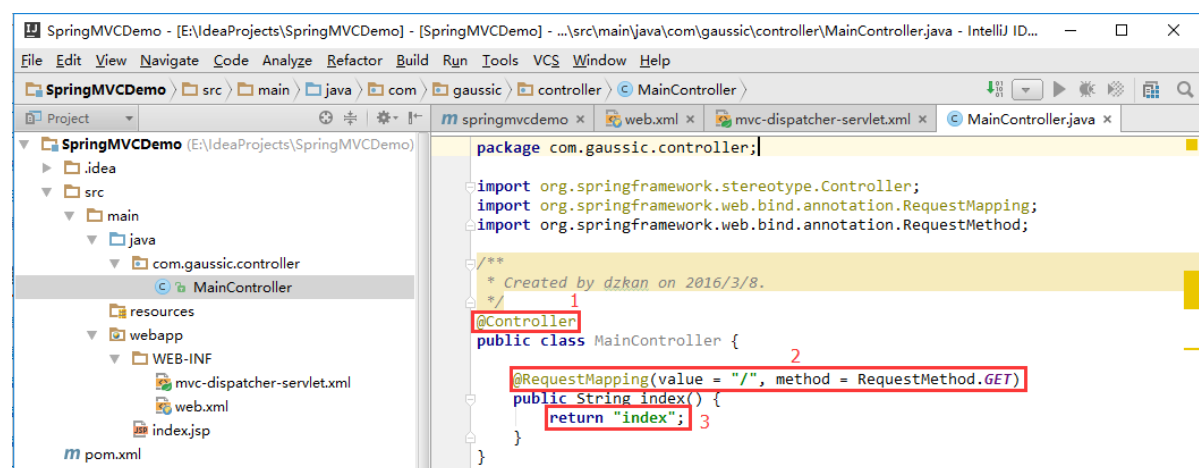
?

```

1 package com.gaussic.controller;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RequestMethod;
6
7 /**
   * Created by dzkan on 2016/3/8.

```

```
8          */
          @Controller
9      public class MainController {
10
11      @RequestMapping(value = "/", method = RequestMethod.GET)
12          public String index() {
13
14              return "index";
15          }
16      }
17  }
```



(1) @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；

(2) @RequestMapping()注解：用于定义一个请求映射，value 为请求的 url，值为 / 说明，该请求首页请求，method 用以指定该请求类型，一般为 get 和 post；

(3) return "index"：处理完该请求后返回的页面，此请求返回 index.jsp 页面。

回到 mvc-dispatcher-servlet.xml，进行相关配置。首先加入 component-scan 标签，指明 controller 所在的包，并扫描其中的注解（最好不要复制，输入时按 IDEA 会在 beans xmlns 中添加相关内容）：

```
?
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             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">
6
7             <!--指明 controller 所在包，并扫描其中的注解-->
8             <context:component-scan base-package="com.gaussic.controller"/>
9
10            </beans>
```

再进行 js、image、css 等静态资源访问的相关配置，这样，SpringMVC 才能访问网站内的静态资源：

```
?
1      <!-- 静态资源(js、image 等)的访问 -->
2      <mvc:default-servlet-handler/>
```

再开启 springmvc 注解模式，由于我们利用注解方法来进行相关定义，可以省去很多的配置：

```
?
1      <!-- 开启注解 -->
2      <mvc:annotation-driven/>
```

再进行视图解析器的相关配置：

?

```
1      <!--ViewResolver 视图解析器-->
      <!--用于支持 Servlet、JSP 视图解析-->
2  <bean id="jspViewResolver" class="org.springframework.web.servlet.vi
3      ew.InternalResourceViewResolver">
4      <property name="viewClass" value="org.springframework.web.servle
      t.view.JstlView"/>
5      <property name="prefix" value="/pages/" />
6      <property name="suffix" value=".jsp" />
7  </bean>
```

关于 controller 如何找到视图文件，这里需要详细的说明。在 `controller` 的一个方法中，返回的字符串定义了所需访问的 jsp 的名字（如上面的 `index`）。在 `jspViewResolver` 中，有两个属性，一个是 `prefix`，定义了所需访问的文件路径前缀，另一是 `suffix`，表示要访问的文件的后缀，这里为 `.jsp`。那么，如果返回字符串是 `xxx`，SpringMVC 就会找到 `/WEB-INF/pages/xxx.jsp` 文件。

完成以上配置后，`mvc-dispatcher-servlet.xml` 文件如下图所示：

?

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



```

7 context.xml http://www.springframework.org/schema/mvc http://www.sp
ringframework.org/schema/mvc/spring-mvc.xsd">
8
9      <!--指明 controller 所在包，并扫描其中的注解-->
10     <context:component-scan base-package="com.gaussic.controller"/>
11
12     <!-- 静态资源(js、image 等)的访问 -->
13     <mvc:default-servlet-handler/>
14
15     <!-- 开启注解 -->
16     <mvc:annotation-driven/>
17
18     <!--ViewResolver 视图解析器-->
19     <!--用于支持 Servlet、JSP 视图解析-->
20     <bean id="jspViewResolver" class="org.springframework.web.servle
3    t.view.InternalResourceViewResolver">
1      <property name="viewClass" value="org.springframework.web.se
4      rvlet.view.JstlView"/>
1      <property name="prefix" value="/WEB-INF/pages/" />
1      <property name="suffix" value=".jsp" />
5      </bean>
1      </beans>

```

我们删除 webapp 目录下的 index.jsp 文件，在 WEB-INF 目录下新建文件夹 pages，再在 pages 目录下新建 index.jsp，并修改为如下所示：

```

?
1 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2     <!DOCTYPE html>
3     <html lang="zh-CN">
4     <head>

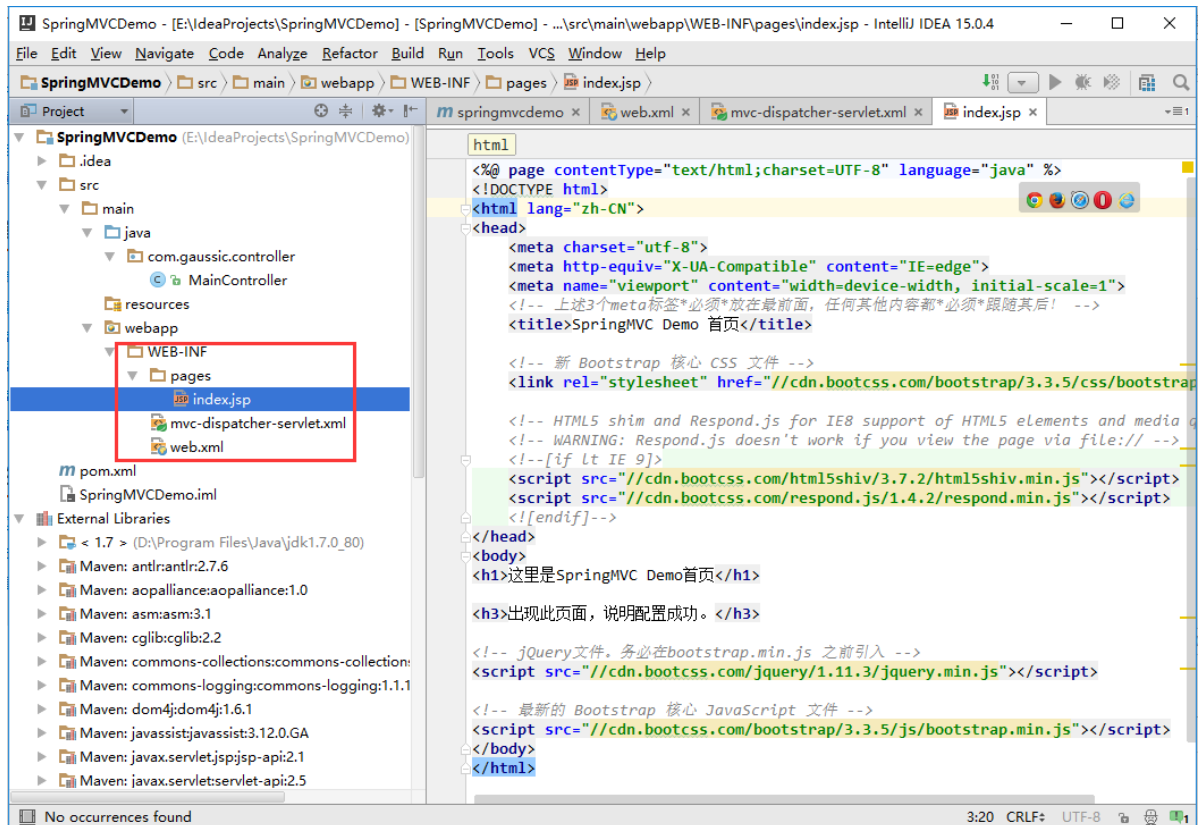
```

```

4         <meta charset="utf-8">
5
6         <meta http-equiv="X-UA-Compatible" content="IE=edge">
7
8         <meta name="viewport" content="width=device-width, initial-
9             scale=1">
10        <!-- 上述 3 个 meta 标签*必须*放在最前面，任何其他内容都*必须*跟随其
11            后! -->
12        <title>SpringMVC Demo 首页</title>
13
14
15        <!-- 新 Bootstrap 核心 CSS 文件 -->
16        <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css
17            /bootstrap.min.css">
18
19
20        <!--
21        - HTML5 shim and Respond.js for IE8 support of HTML5 elements and
22            media queries -->
23        <!--
24        - WARNING: Respond.js doesn't work if you view the page via file:/
25            / -->
26        <!--[if lt IE 9]>
27        <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"
28            s"></script>
29        <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"
30            "></script>
31        <![endif]-->
32        </head>
33        <body>
34        <h1>这里是 SpringMVC Demo 首页</h1>
35
36
37        <h3>出现此页面，说明配置成功。</h3>
38
39
40        <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->
41        <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
42            t>
43
44        <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->

```

```
1 <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
8
      ></script>
      </body>
1    </html>
9
```

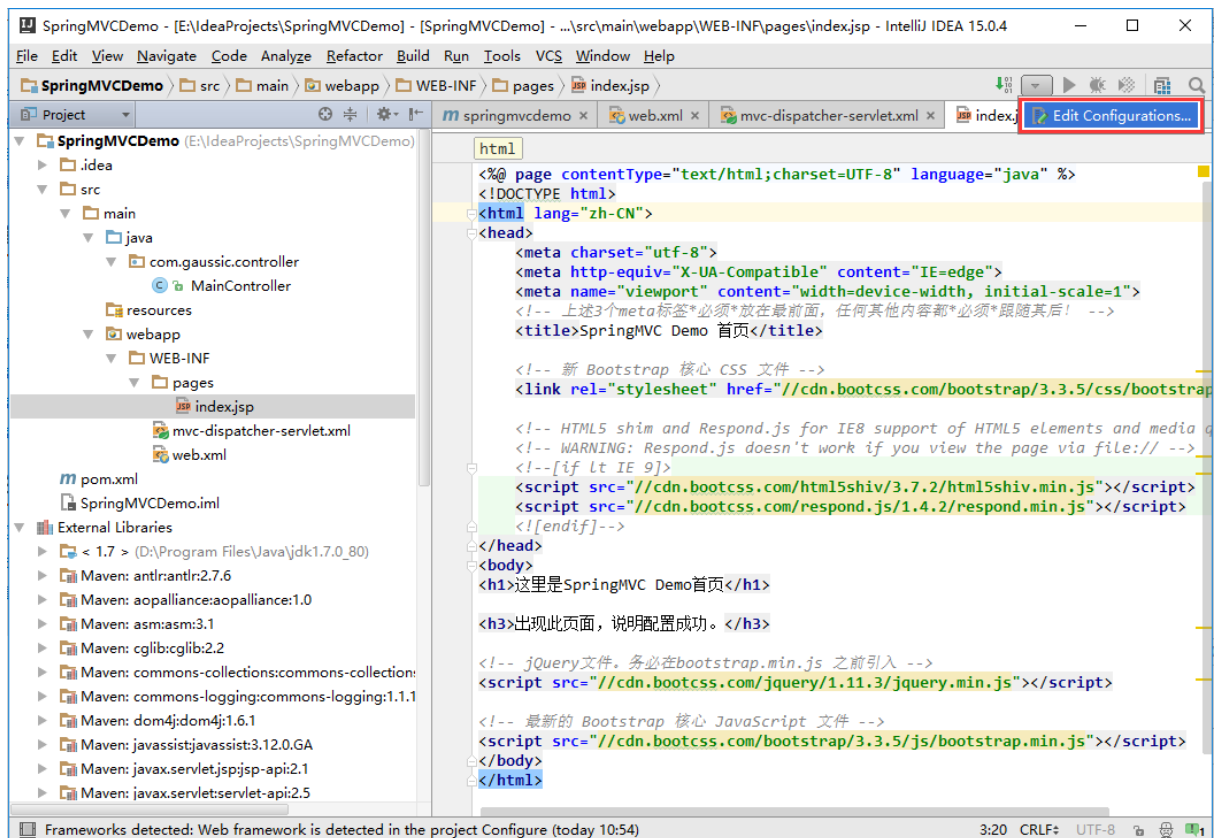


这里使用了 Bootstrap 的 CDN 加速服务，如果要使用本地的 Bootstrap，请前往 [Bootstrap 官网](#)

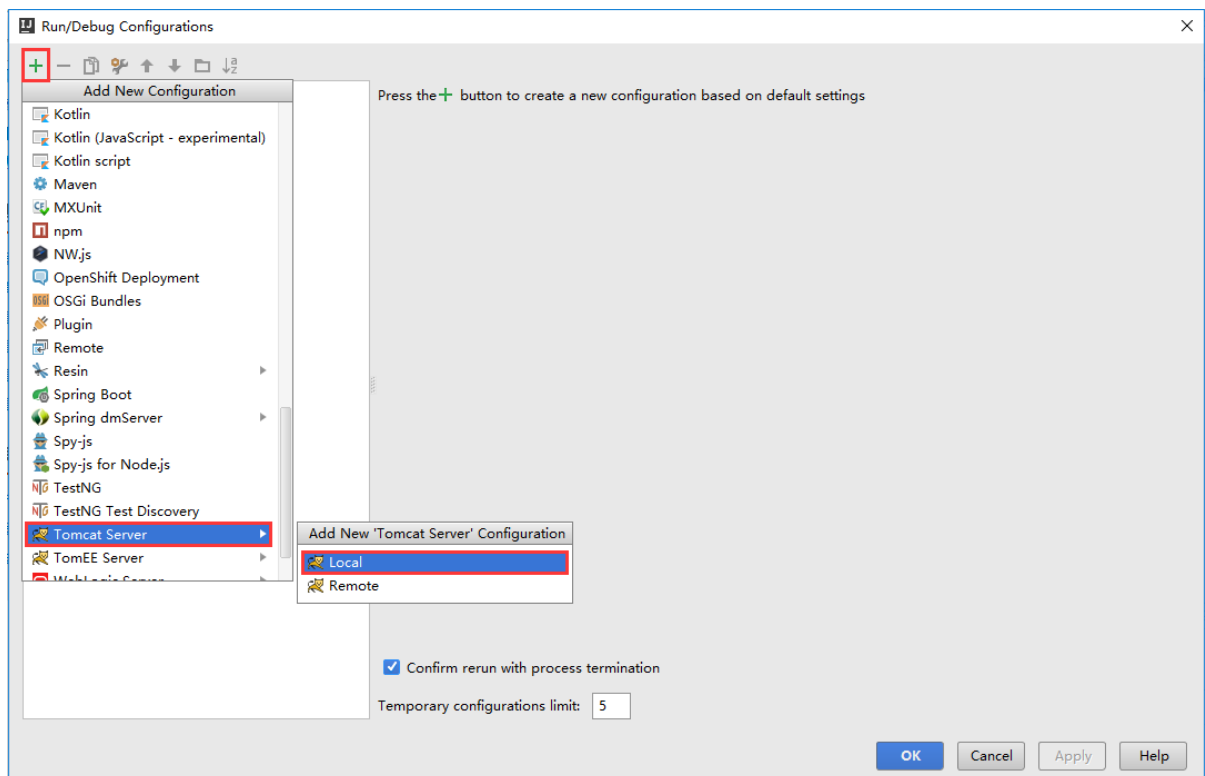
下载，并放在 webapp 目录下，然后引入到 index.jsp 中，这里不做详细介绍。

现在，需要配置 Tomcat 来运行该项目。点击界面右上角的向下箭头，选择 Edit Configuration

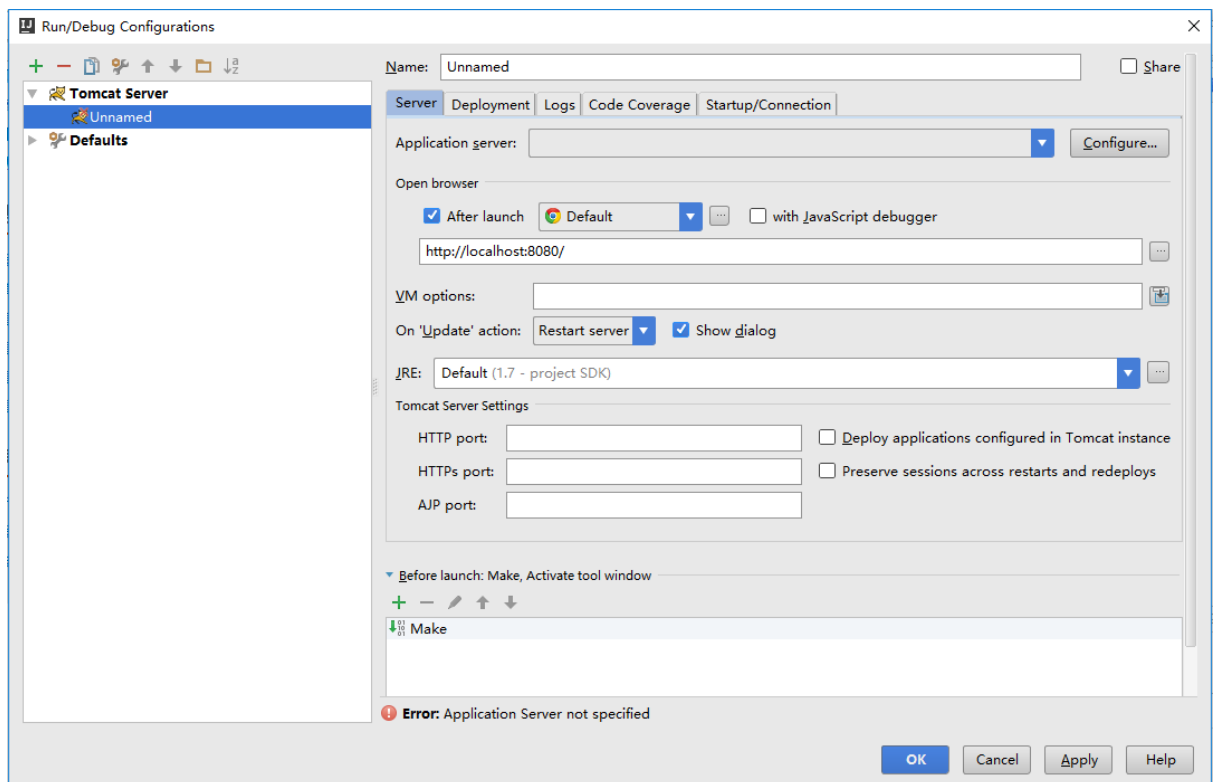
S：



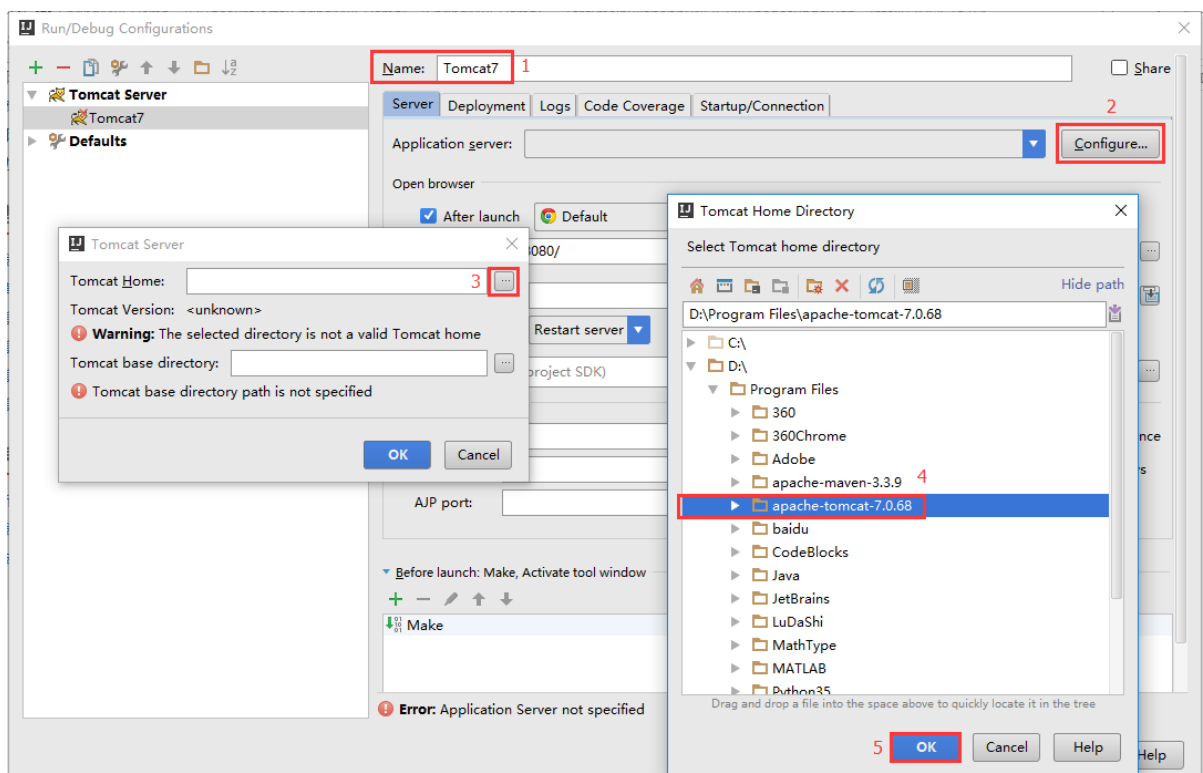
点击左上角的“+”号，选择 Tomcat Server，（如果没有请选择最下方的 33 items more，找到 Tomcat Server），再选择 Local：



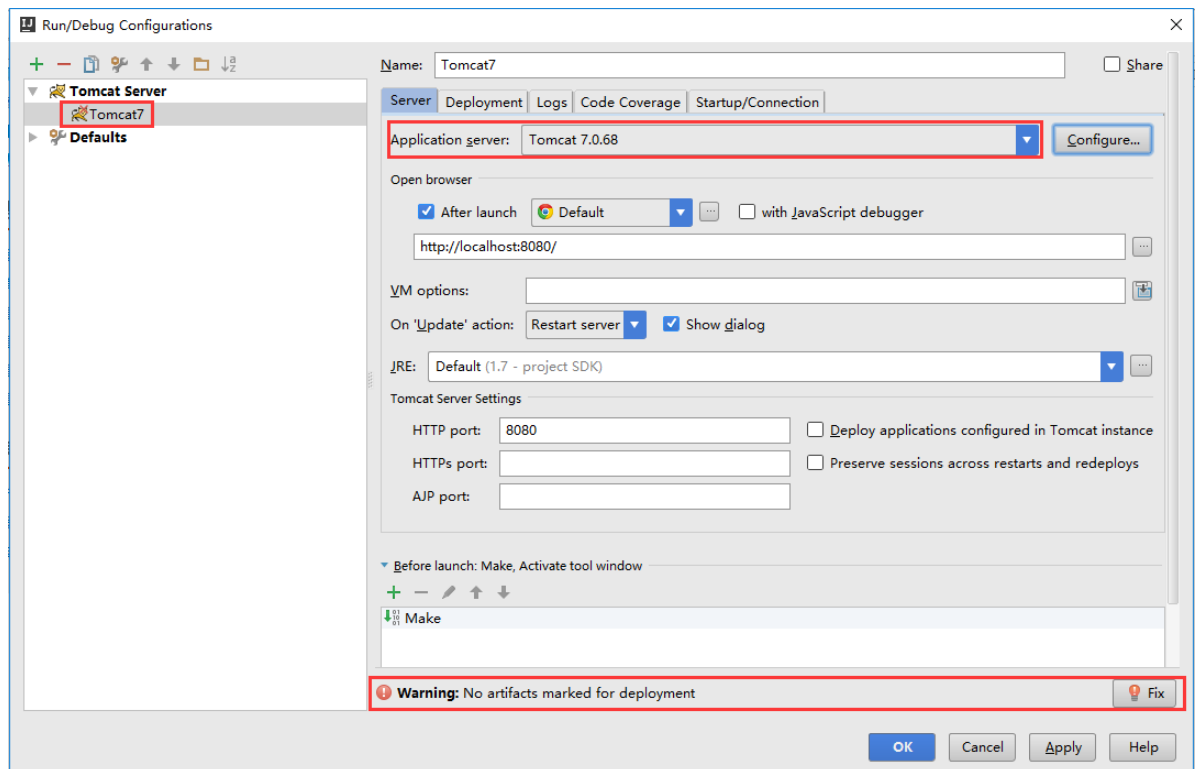
进入如下界面：



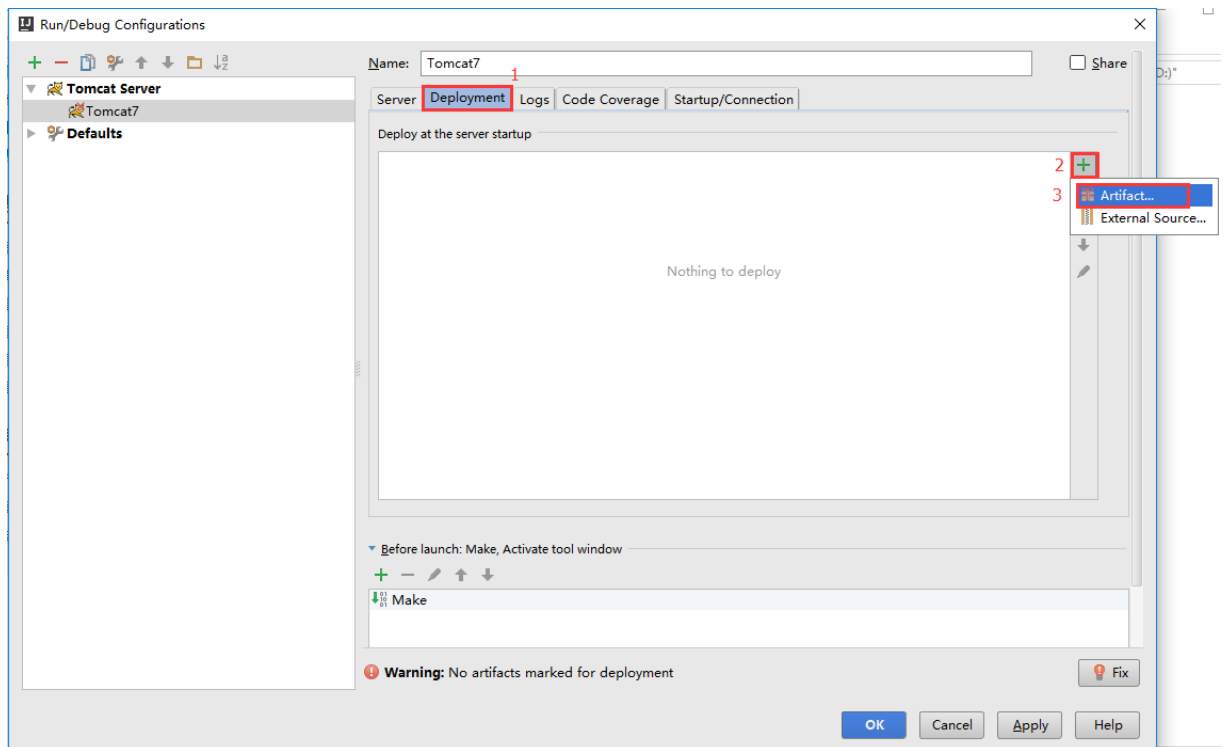
点击 Application server 右边的 Configure，导入 Tomcat 目录：



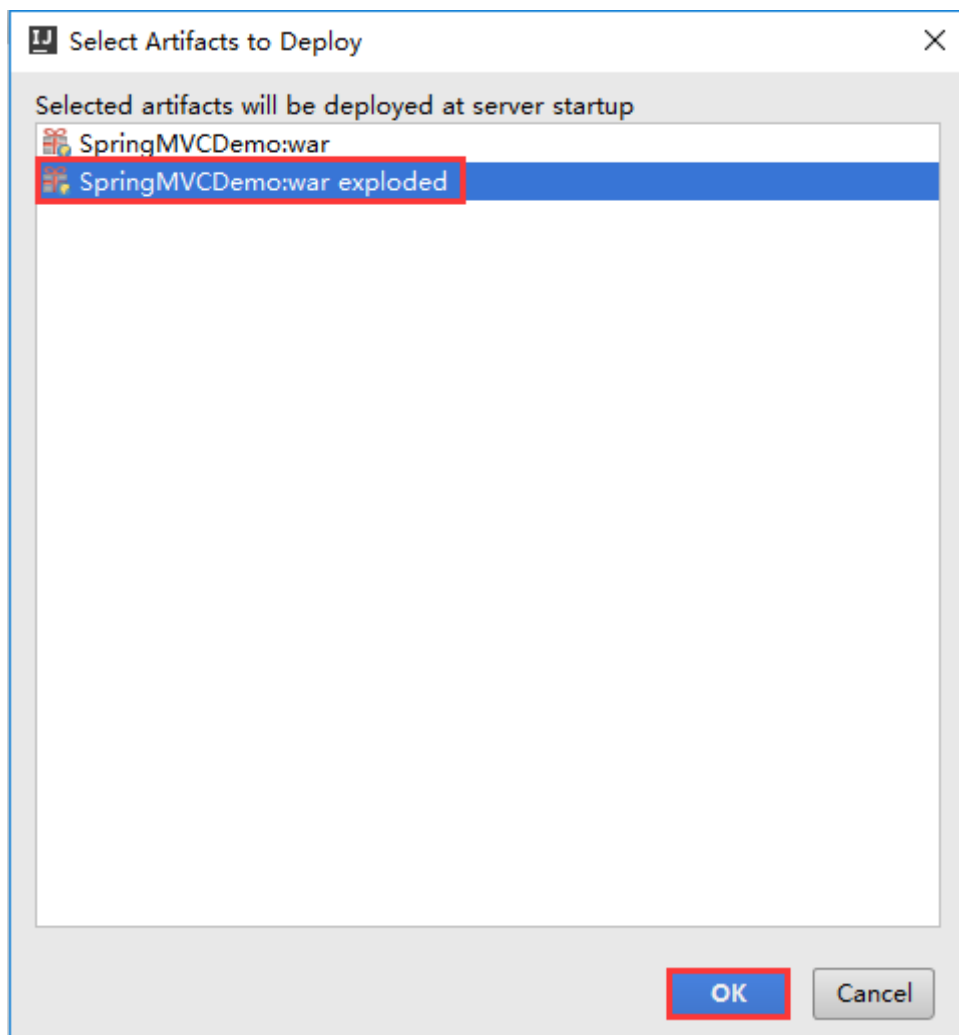
在配置好 tomcat 的路径后，如下图所示，发现依然存在警告，且左方的 Tomcat7 图标上有一个错误标记，说明还没有配置完全：



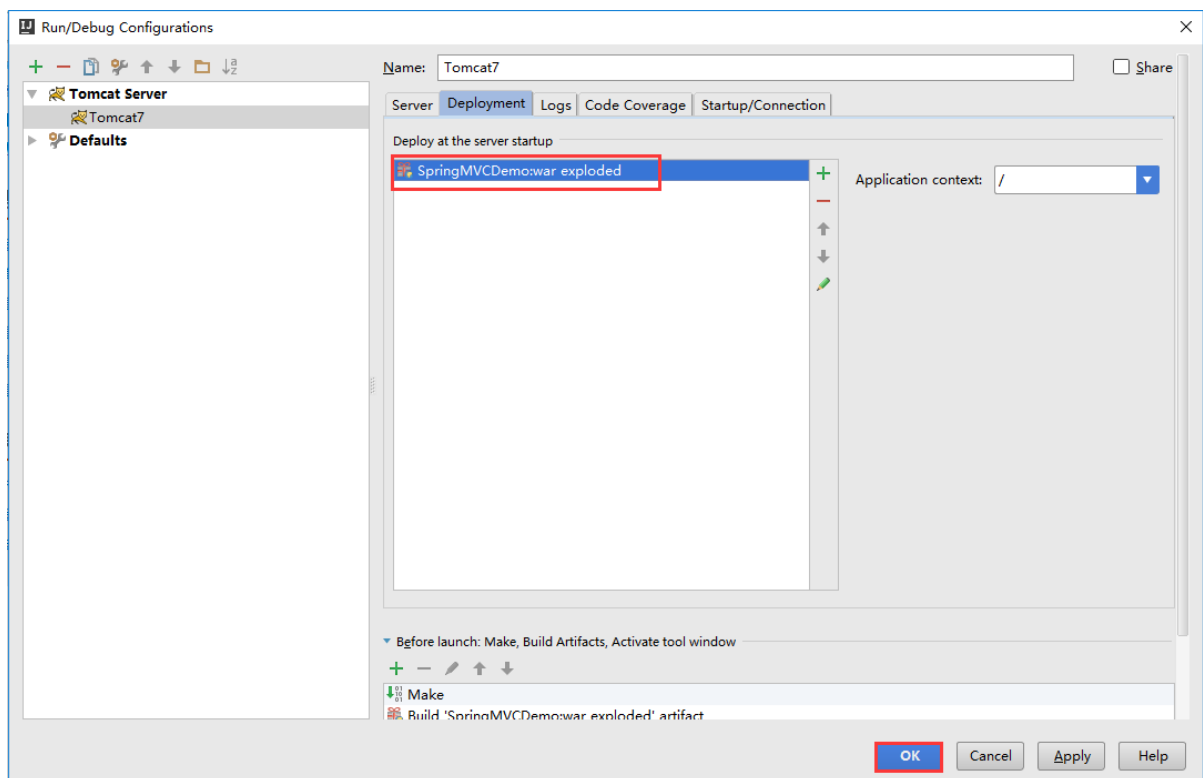
我们还需要将项目部署到 Tomcat 服务器中。点击 Deployment，再点击右边的“+”号，添加一个 Artifact：



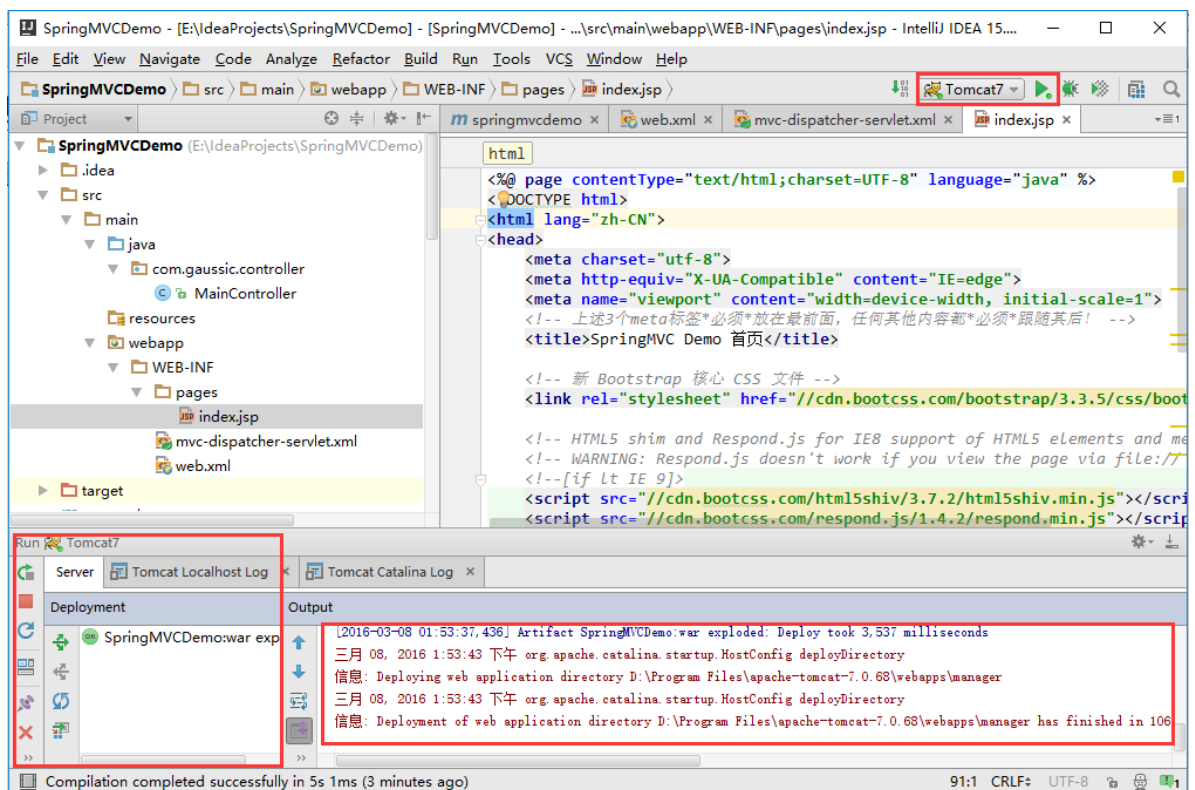
选择第二个：war exploded，点击 OK，这样，该项目就已经部署到了 tomcat 中：



再点击 OK，整个 Tomcat 配置结束：



点击界面右上角的红框中的绿色箭头，就可以启动 Tomcat 了，其控制台输出将在 IDEA 下方显示



启动后，浏览器将自动弹出项目首页：



这里是SpringMVC Demo首页

出现此页面，说明配置成功。

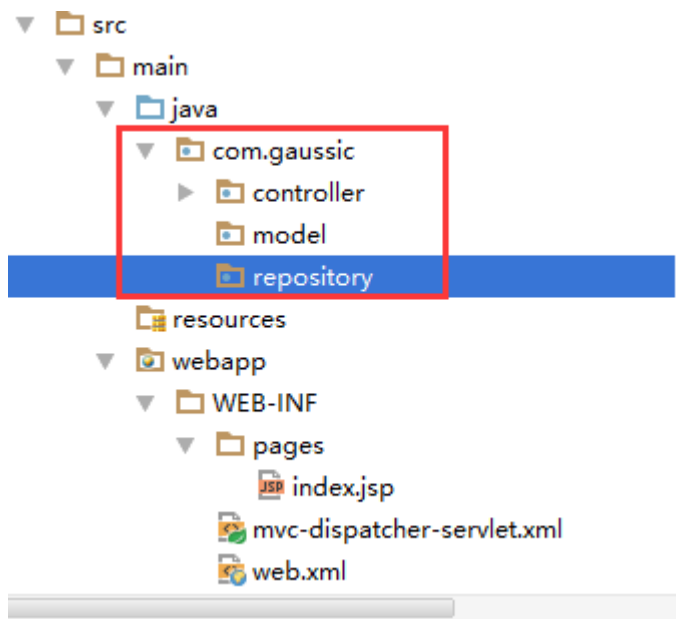
这样，说明配置完成。这里总结一下其相关机制：首先，浏览器访问 `localhost:8080`，后台 `controller` 拦截该请求，进行相应的处理（此处无），在跳转到视图 `index.jsp` 进行显示。此后，将会进行详细的介绍。

六、数据库配置

下面，就要通过一个简单的例子，来介绍 SpringMVC 如何集成 Spring Data JPA（由 Hibernate JPA 提供），来进行强大的数据库访问，并通过本章节的讲解，更加深刻地认识 Controller 是如何进行请求处理的，相信看完这一章节，你就可以开始你的开发工作了。

准备工作：

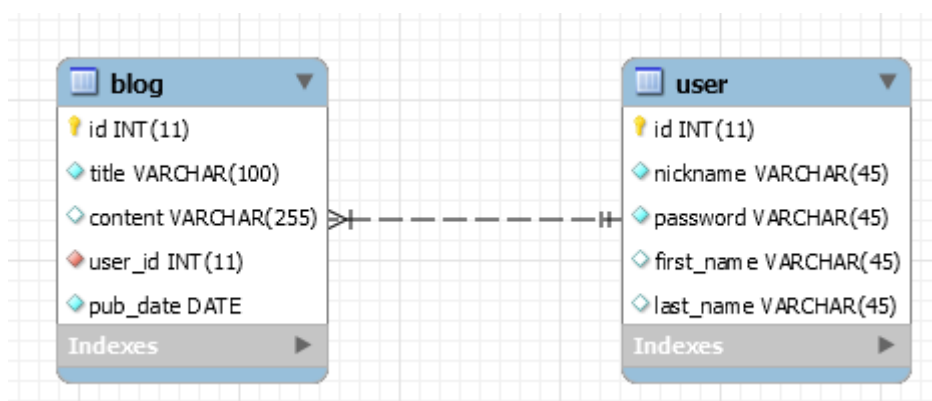
在 `src/main/java` 中新建两个包：`com.gaussic.model`、`com.gaussic.repository`，将在后面用上，如下图所示：



1、创建 Mysql 数据库

本文的讲解使用 Mysql 数据库，如果使用其它数据库的读者，可以去网上参考其他的配置教程，在此不做太多的叙述。数据库是一个底层的东西，底层的细节对上层的抽象并没有太大的影响，因此，只要配置好数据库，本章的内容仍然是适用于所有数据库的（貌似如此）。

假设我们现在要建立一个小小的博客系统，其数据库 ER 图如下所示（当然这只是一个小小的例子，真实的博客系统比这要复杂的多）：



新建一个数据库 springdemo，在数据库中，有两张表：

（1）用户表 user：用户登录信息，主键 id 设为自增；

(2) 博文表 blog：储存用户发表的博文，主键 id 设为自增，其中有一个外键 user_id 链接到 user 表。

详细表结构如下图所示：




Table Name:

Schema: **springdemo**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
nickname	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
password	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
first_name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
last_name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	




Table Name:

Schema: **springdemo**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
title	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
content	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
user_id	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
pub_date	DATE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

使用 MySQL Workbench 添加外键流程：

Query 1

user - Table

blog - Table

Table Name: blog

Schema: springdemo

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
title	VARCHAR(100)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
content	VARCHAR(255)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
user_id	INT(11)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:

Collation: Table Default

Comments:

Data Type:

Default:

Storage: ☐ Virtual ☐ Stored

☐ Primary Key ☐ Not Null ☐ Unique

☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

Columns

Indexes

Foreign Keys

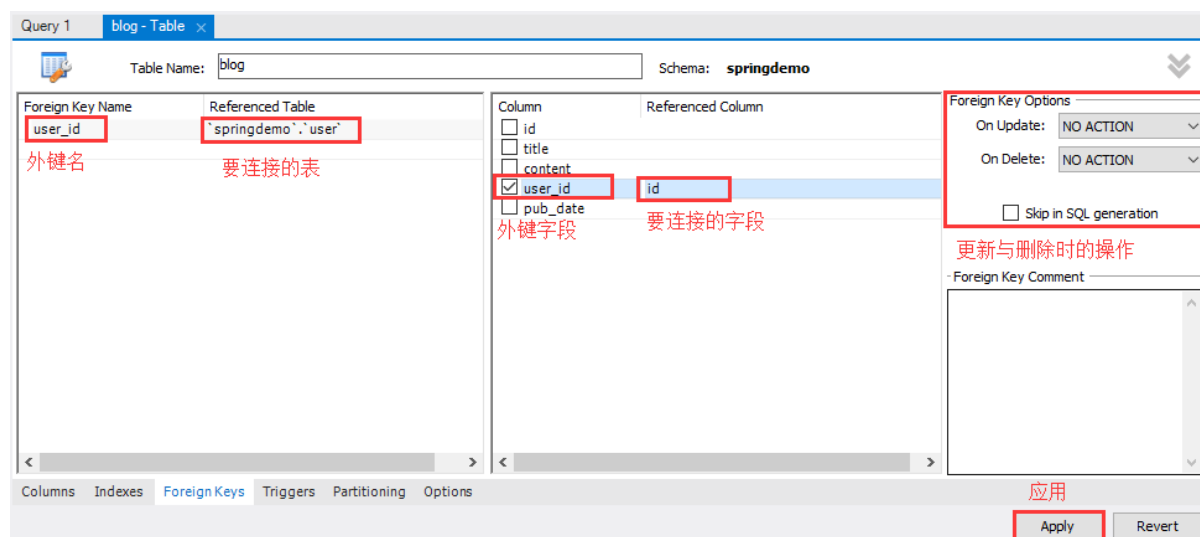
Triggers

Partitioning

Options

Apply

Revert

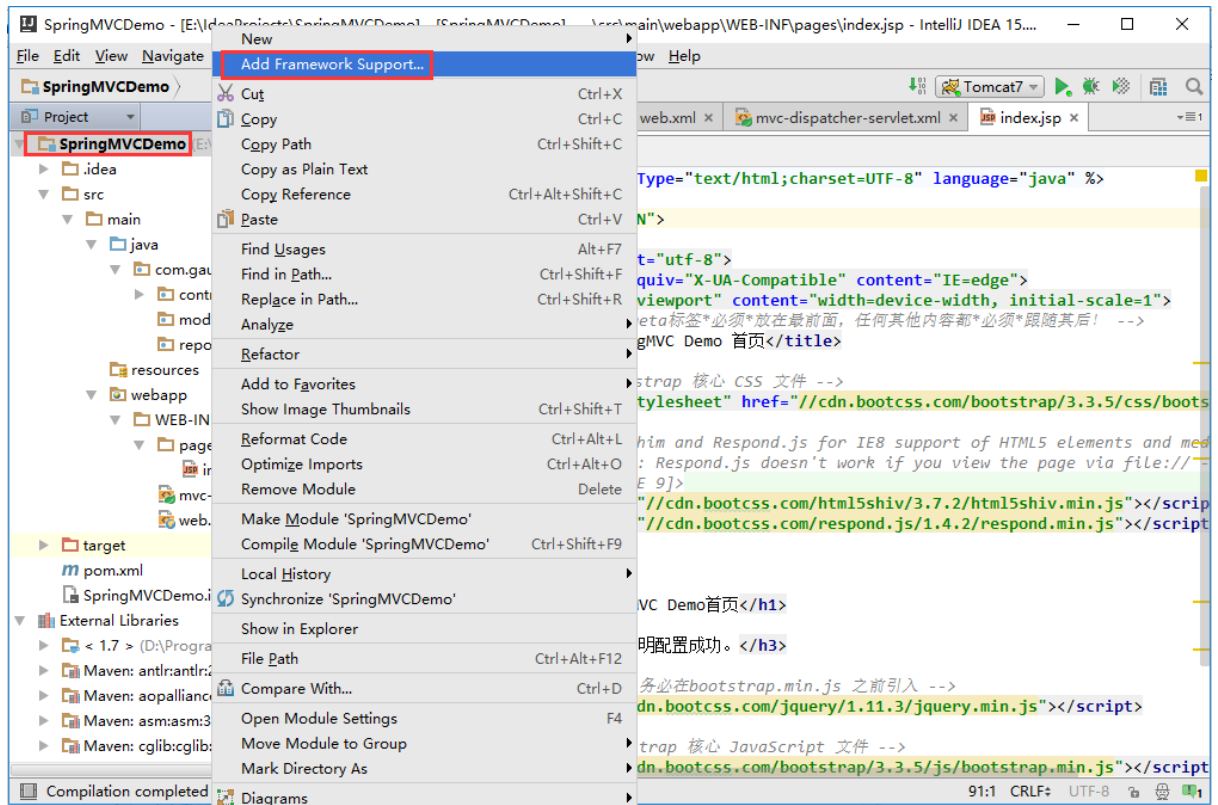


注意：在添加外键时，应该根据需求设置，例如右边红框中的 Foreign Key Options，默认在 Delete 时是 NO ACTION，说明在删除一个用户时，如果数据库中存在该用户的文章，那么就无法删除该用户，也无法删除该用户的所有文章，而如果将该选项改为 CASCADE，那么删除该用户，就会同时删除该用户所有的文章。通常后者是不太可取的，因为如果发生了删除用户的误操作，很有可能该用户的内容被连带删除，且不可逆，这也是实现真实系统时需要考虑的原因之一。

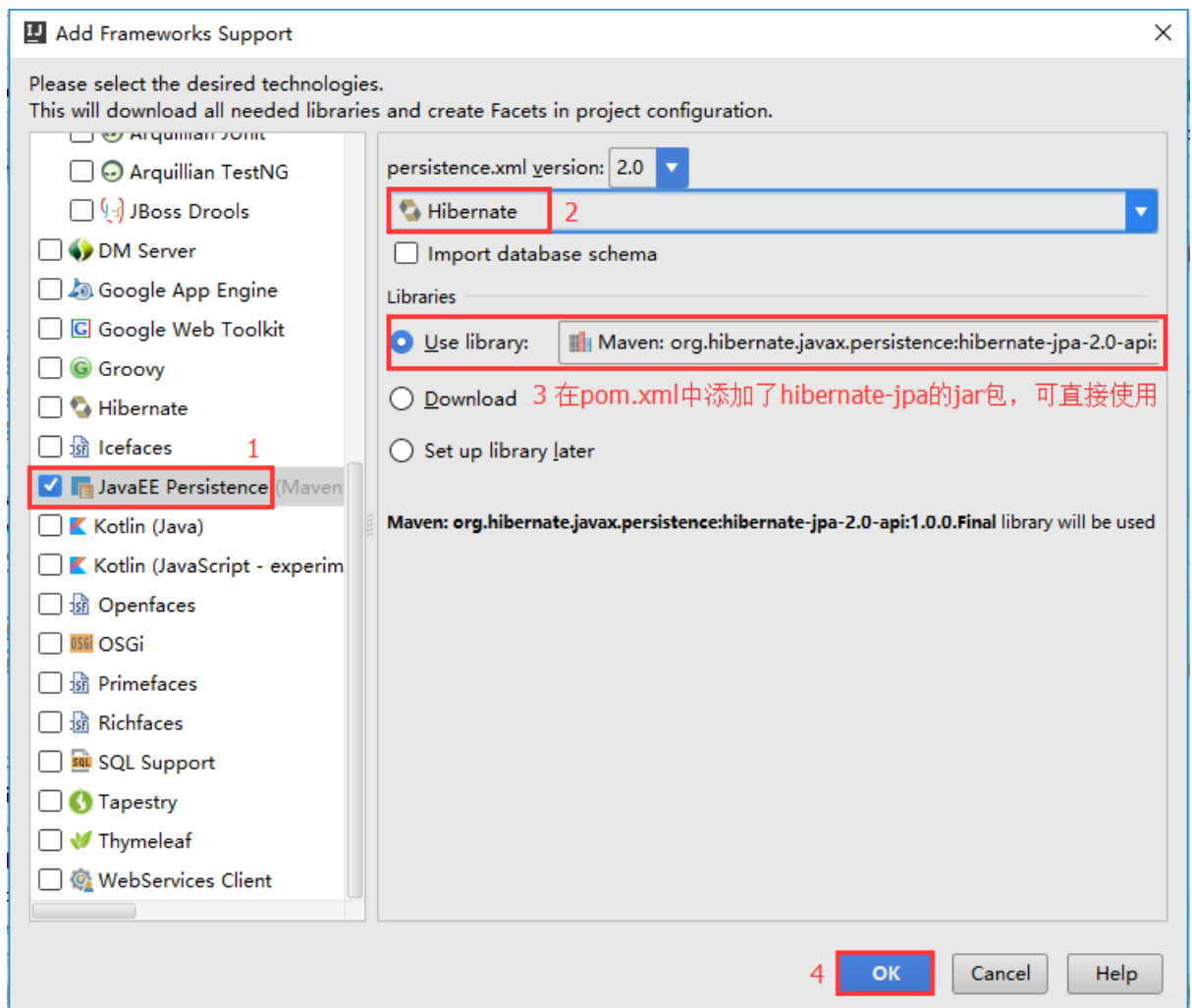
2、IntelliJ IDEA 导入数据库

对于此前所接触的一些常用的框架中，一张数据表往往对应一个 Java Bean。在 SpringMVC 中，这个 Java Bean 相当于 model。那么，这个类是否需要自己来写呢？不需要，利用 IntelliJ IDEA 可以帮助我们自动的生成这些 JavaBean。

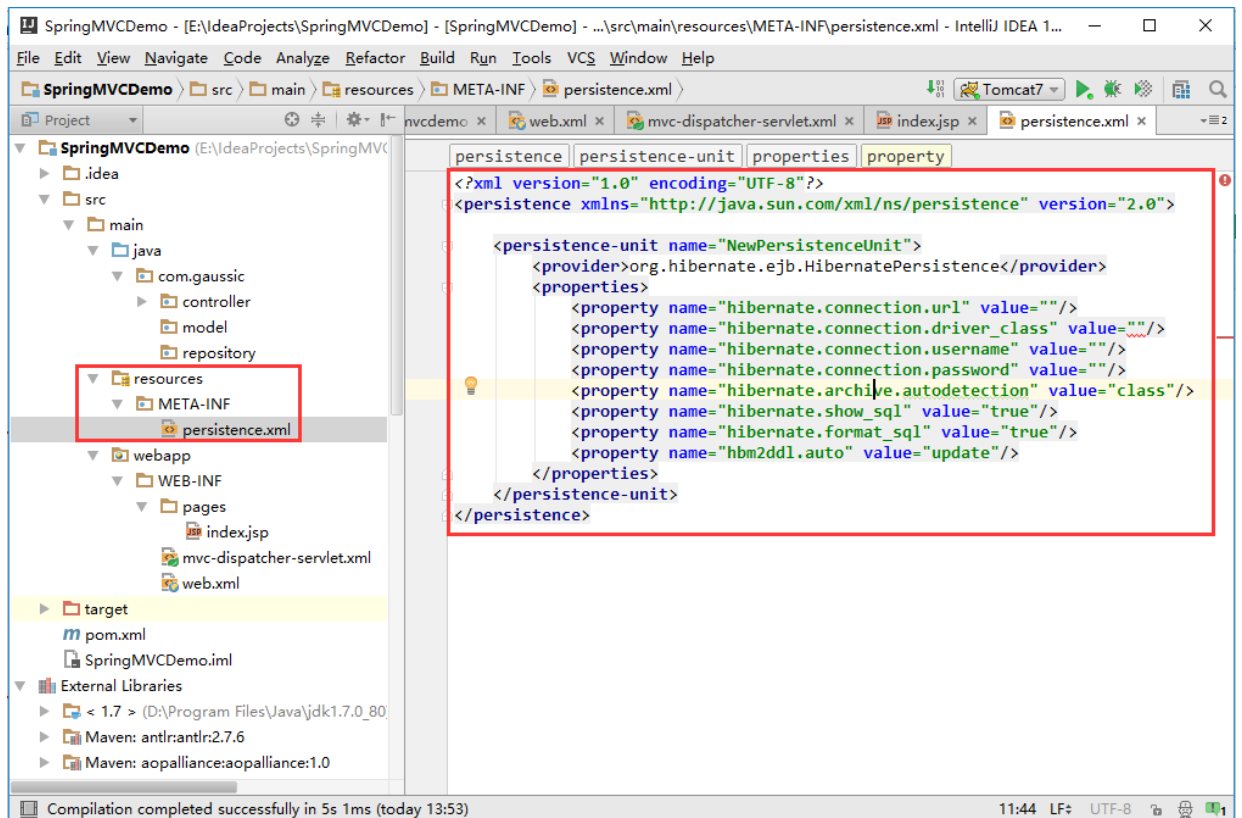
首先，右键项目，选择 Add Framework Support：



下拉选择 JavaEE Persistence , 右边 provider 选择 Hibernate :



在这一步结束后，我们可以发现，在 resources 里面生成了 persistence.xml 配置文件，左边栏出现了一个 Persistence 标题（若没有请点击左下角那个灰框）：



persistence.xml 具体如下：

?

```

1      <?xml version="1.0" encoding="UTF-8"?>
2      <persistence xmlns="http://java.sun.com/xml/ns/persistence" version
3          ="2.0">
4          <persistence-unit name="NewPersistenceUnit">
5              <provider>org.hibernate.ejb.HibernatePersistence</provider>
6              <properties>
7                  <property name="hibernate.connection.url" value=""/>
8                  <property name="hibernate.connection.driver_class" valu
9                      e=""/>
10                 <property name="hibernate.connection.username" value=""
11                     />

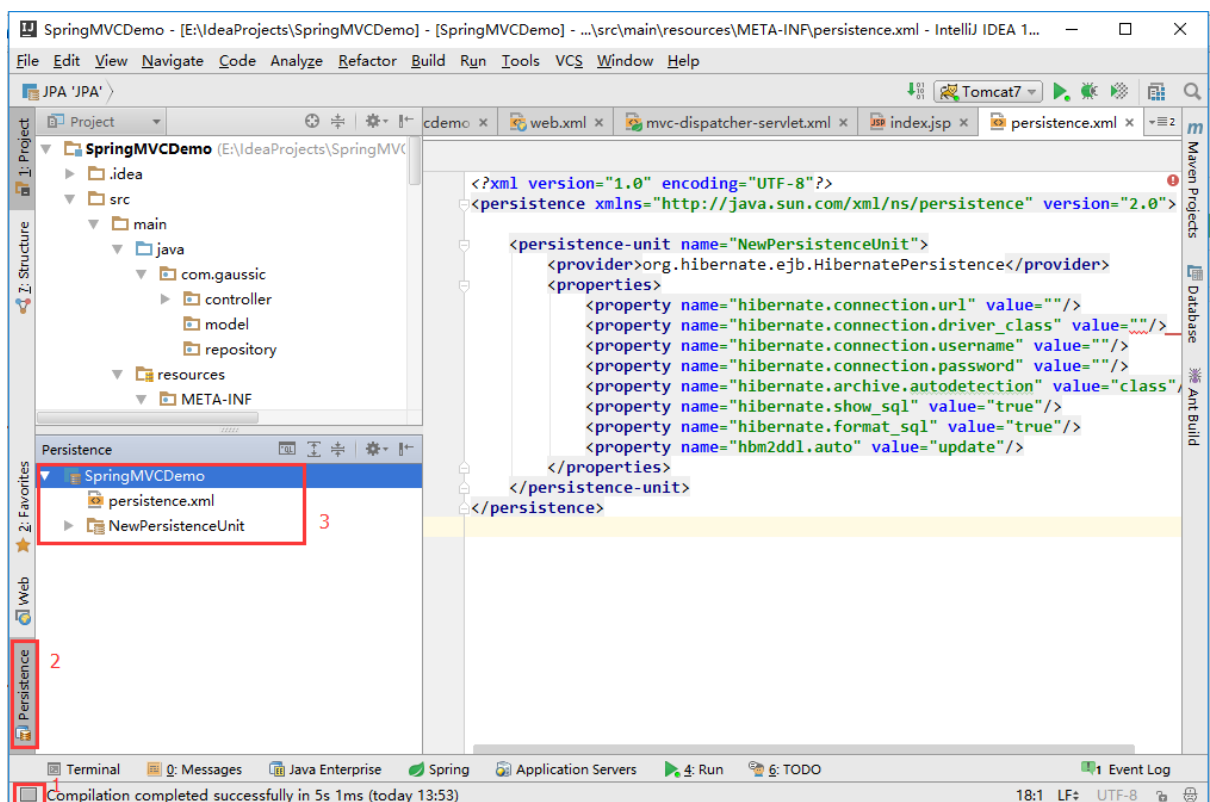
```

```

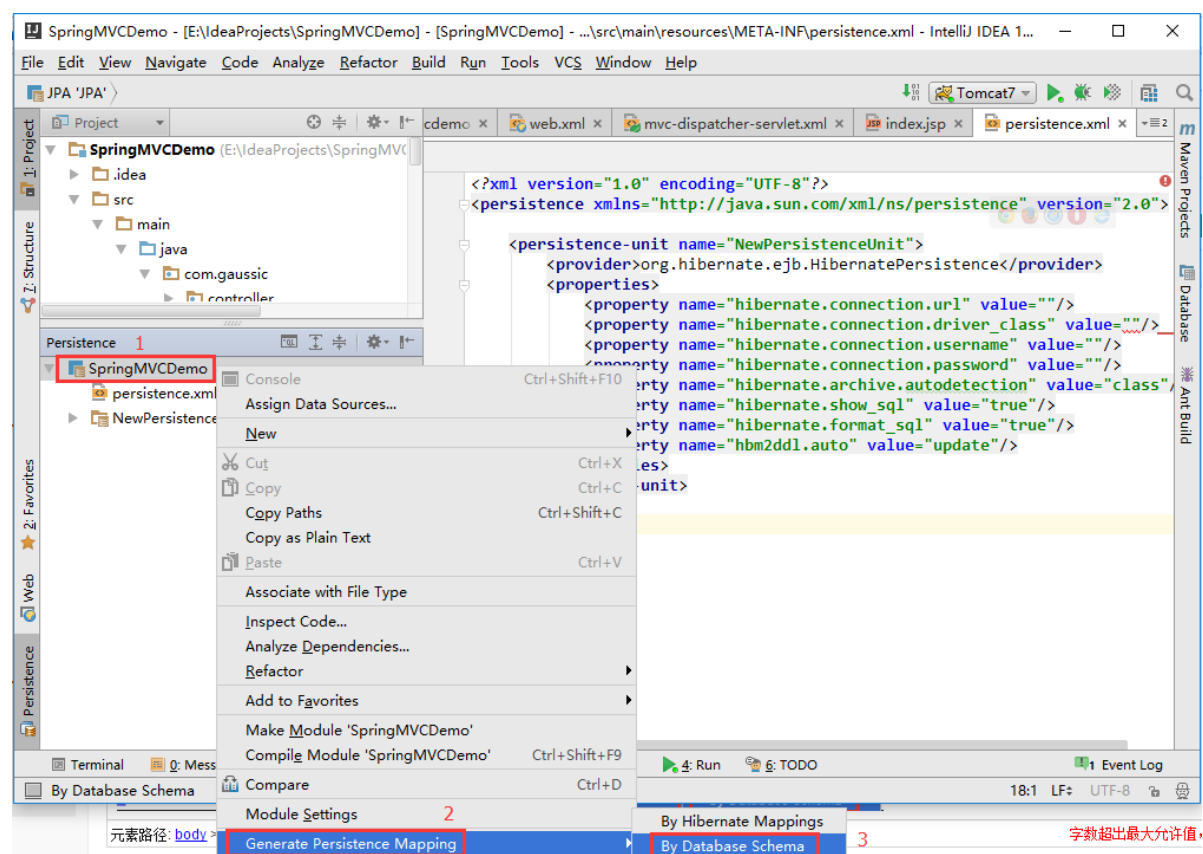
1      <property name="hibernate.connection.password" value=""
0          />
1      <property name="hibernate.archive.autodetection" value=
1          "class"/>
1      <property name="hibernate.show_sql" value="true"/>
1      <property name="hibernate.format_sql" value="true"/>
2      <property name="hbm2ddl.auto" value="update"/>
1          </properties>
1      </persistence-unit>
3      </persistence>

```

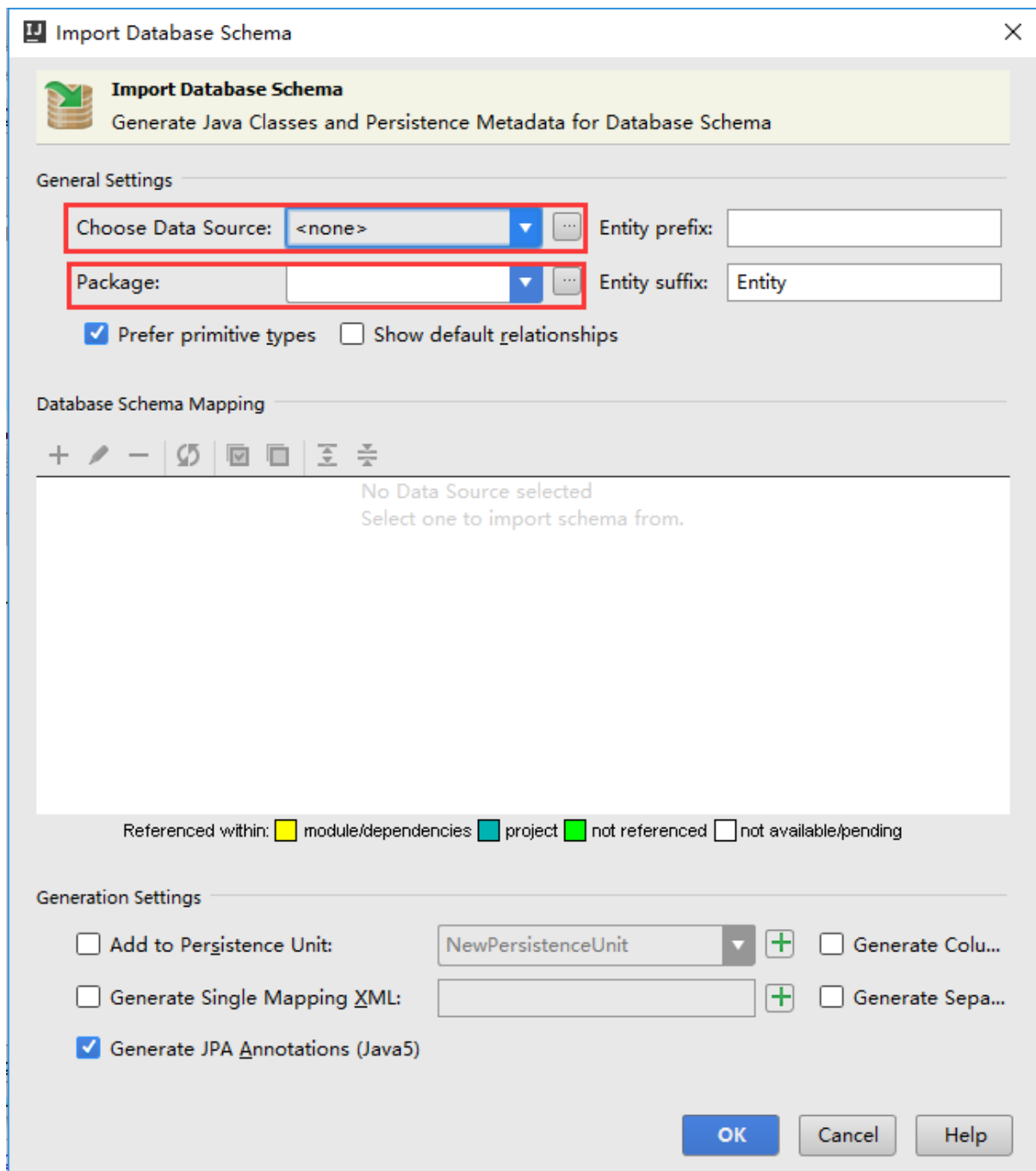
我们先不着急填写这个配置文件。点开左边栏的 Persistence，显示如下图所示：



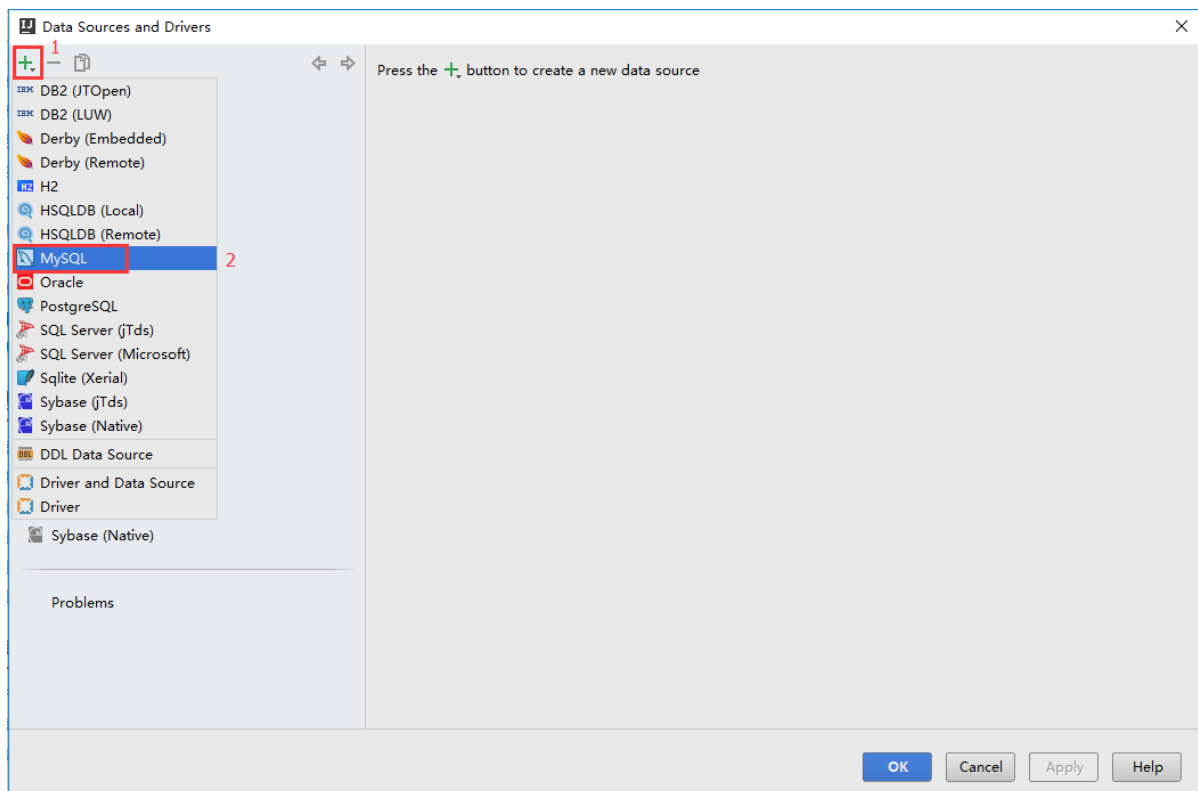
右键项目名，选择 Generate Persistence Mapping，再选择 By Database Schema：



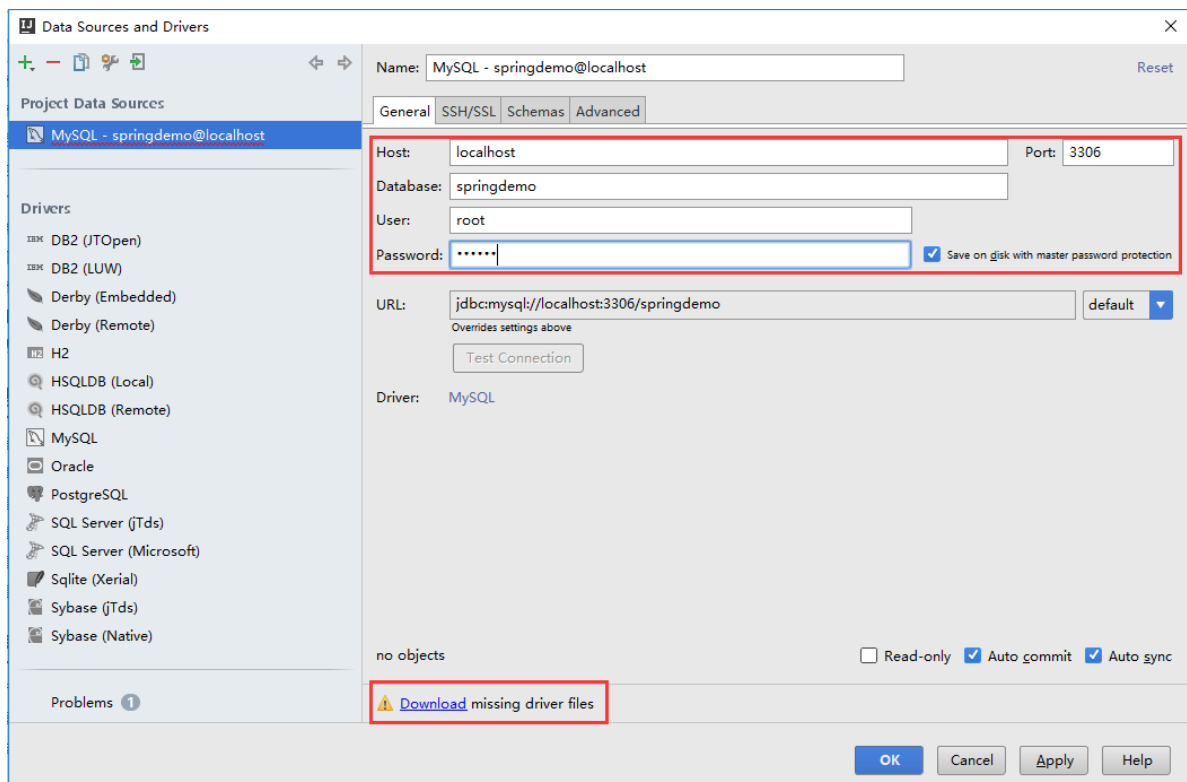
出现如下界面，其主要需要配置的地方如下图红框所示：



点击 Choose Data Source 右边的三个点选择数据源，在弹出的界面左上角选择“+”，选择
Mysql：



在如下界面填写主机、端口号、数据库名、用户名、密码，如果驱动丢失点击下面的 Download 可以下载驱动，点击 Test Connection 可以测试数据库是否连接成功：



在以上界面配置完成后，点 OK，第一次使用需要 Setup Master Password：



回到如下页面，package 填写 model 包（1），勾选 Prefer primitive type 使用原始数据类型（2），勾选 Show default relationships 以显示所有数据库关系（3），再点击刷新按钮（4），将会找到数据库中的两个表，勾选两个数据表（5），再勾选 Generate Column Definition 以生成每一列的描述信息（6）。选中 blog 表然后点击“+”号按钮，添加外键关系（7）。

Import Database Schema

Generate Java Classes and Persistence Metadata for Database Schema

General Settings

Choose Data Source:
MySQL - springdemo@localhost

Entity prefix:

1
Package:
com.gaussic.model

Entity suffix:
Entity

2
☒
Prefer primitive types

☒
3
Show default relationships

Database Schema Mapping

7
+

4

	Database Schema	Map As	Mapped Type
<input checked="" type="checkbox"/>	blog	BlogEntity	BlogEntity
<input checked="" type="checkbox"/>	id	id	int
<input checked="" type="checkbox"/>	user_id	userId	int
<input checked="" type="checkbox"/>	content	content	java.lang.String
<input checked="" type="checkbox"/>	pub_date	pubDate	java.sql.Date
<input checked="" type="checkbox"/>	title	title	java.lang.String
<input checked="" type="checkbox"/>	user	UserEntity	UserEntity
<input checked="" type="checkbox"/>	id	id	int
<input checked="" type="checkbox"/>	first_name	firstName	java.lang.String
<input checked="" type="checkbox"/>	last_name	lastName	java.lang.String
<input checked="" type="checkbox"/>	nickname	nickname	java.lang.String

Referenced within:
 module/dependencies
 project
 not referenced
☐ not available/pending

Generation Settings

☐
Add to Persistence Unit:
defaultPersistenceUnit

☒
6
Generate Column Prope...

☐
Generate Single Mapping XML:

☐
Generate Separate XML ...

☒
Generate JPA Annotations (Java5)

OK

Cancel

Help

Add Relationship

Source: BlogEntity

Table: springdemo.blog

Type: Single

Attribute Name: userByUserId

Map Key Column:

☐ Join Table

博客作者

返回单个对象

Target: UserEntity

Table: springdemo.user

Type: java.util.Collection

Attribute Name: blogsById

Map Key Column:

用户所有博客

返回一个集合

Join Columns

+ - 添加连接的字段，即外键

Source Column	Target Column
user_id	id

外键字段 要连接的字段

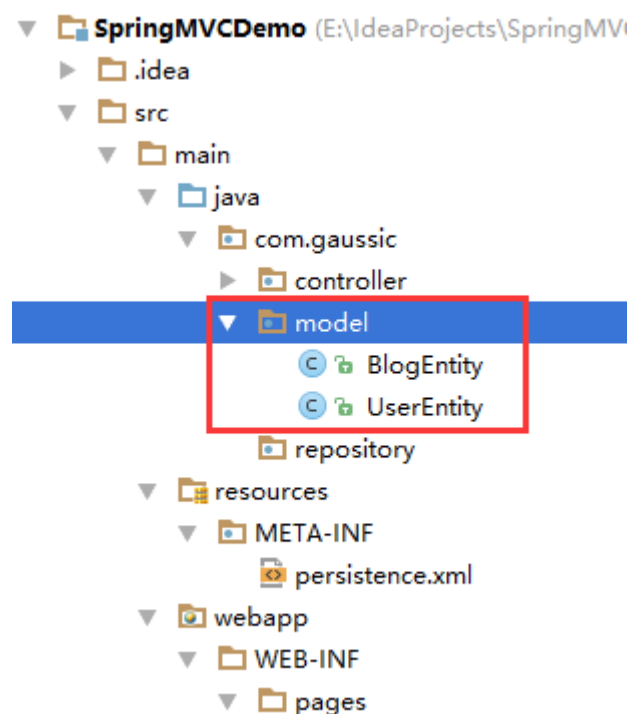
OK Cancel Help

点击 OK 后，在 Database Schema Mapping 中可以发现多出了两个关系，如图所示：

	Database Schema	Map As	Mapped Type
✓	blog	BlogEntity	BlogEntity
✓	user.blogsById	userByUserId	UserEntity
✓	id	id	int
✓	user_id	userId	int
✓	content	content	java.lang.String
✓	pub_date	pubDate	java.sql.Date
✓	title	title	java.lang.String
✓	user	UserEntity	UserEntity
✓	blog.userById.blogsById		java.util.Collection<BlogEntity>
✓	id	id	int
✓	first_name	firstName	java.lang.String

Referenced within: module/dependencies project not referenced not available/pending

再点击 OK，稍后，打开 model 包，可以看到生成了两个 Java Bean，在 SpringMVC 中称为两个实体，它们对应了数据库的两张表：



BlogEntity 如下所示（注意把 `java.sql.Date` 改为 `java.util.Date`）：

```

1 package com.gaussic.model;
6 import javax.persistence.*;
7 import java.util.Date;

```

```
8
9          /**
10             * Created by dzkan on 2016/3/8.
11             */
12             @Entity
13             @Table(name = "blog", schema = "springdemo", catalog = "")
14             public class BlogEntity {
15
16                 private int id;
17
18                 private String title;
19
20                 private String content;
21
22                 private Date pubDate;
23
24                 private UserEntity userByUserId;
25
26                 @Id
27                 @Column(name = "id", nullable = false)
28                 public int getId() {
29
30                     return id;
31                 }
32
33                 public void setId(int id) {
34                     this.id = id;
35                 }
36
37                 @Basic
38                 @Column(name = "title", nullable = false, length = 100)
39                 public String getTitle() {
40
41                     return title;
42                 }
43             }
```



```
30         public void setTitle(String title) {
31             this.title = title;
32         }
33
34         @Basic
35         @Column(name = "content", nullable = true, length = 255)
36         public String getContent() {
37             return content;
38         }
39
40         public void setContent(String content) {
41             this.content = content;
42         }
43
44         @Basic
45         @Column(name = "pub_date", nullable = false)
46         public Date getPubDate() {
47             return pubDate;
48         }
49
50         public void setPubDate(Date pubDate) {
51             this.pubDate = pubDate;
52         }
53
54         @Override
55         public boolean equals(Object o) {
56             if (this == o) return true;
57             if (o == null || getClass() != o.getClass()) return false;
58
```

```

59         BlogEntity that = (BlogEntity) o;
60
61         if (id != that.id) return false;
62
63         if (title != null ? !title.equals(that.title) : that.title
64             != null) return false;
65
66         if (content != null ? !content.equals(that.content) : that.
67             content != null) return false;
68
69         if (pubDate != null ? !pubDate.equals(that.pubDate) : that.
70             pubDate != null) return false;
71
72         return true;
73     }
74
75     @Override
76     public int hashCode() {
77         int result = id;
78
79         result = 31 * result + (title != null ? title.hashCode() :
80             0);
81
82         result = 31 * result + (content != null ? content.hashCode(
83             ) : 0);
84
85         result = 31 * result + (pubDate != null ? pubDate.hashCode(
86             ) : 0);
87
88         return result;
89     }
90
91     @ManyToOne

```

```

81     @JoinColumn(name = "user_id", referencedColumnName = "id", nullable = false)
82     public UserEntity getUserByUserId() {
83         return userByUserId;
84     }
85
86     public void setUserByUserId(UserEntity userByUserId) {
87         this.userByUserId = userByUserId;
88     }
89 }

```

再看 UserEntity :

```

?
package com.gaussic.model;

1
2
3
4
5     /**
6     * Created by dzkan on 2016/3/8.
7     */
8     @Entity
9     @Table(name = "user", schema = "springdemo", catalog = "")
10    public class UserEntity {
11
12        private int id;
13
14        private String nickname;
15
16        private String password;
17
18        private String firstName;

```

```

1         private String lastName;

1         private Collection<BlogEntity> blogsById;

1
2         @Id
   @Column(name = "id", nullable = false)
1         public int getId() {
3             return id;
1         }

4         public void setId(int id) {
1             this.id = id;
5         }

1         @Basic
6         @Column(name = "nickname", nullable = false, length = 45)
1         public String getNickname() {
7             return nickname;
1         }

8         public void setNickname(String nickname) {
1             this.nickname = nickname;
9         }

2         @Basic
   @Column(name = "password", nullable = false, length = 45)
0         public String getPassword() {
2             return password;
1         }

```

```

2         public void setPassword(String password) {
2             this.password = password;
2             }
2
2             @Basic
3         @Column(name = "first_name", nullable = true, length = 45)
2         public String getFirstName() {
4             return firstName;
2             }
2
5         public void setFirstName(String firstName) {
2             this.firstName = firstName;
6             }
2
2             @Basic
7         @Column(name = "last_name", nullable = true, length = 45)
2         public String getLastName() {
2             return lastName;
8             }
2
9         public void setLastName(String lastName) {
3             this.lastName = lastName;
0             }
3
3             @Override
1         public boolean equals(Object o) {
3             if (this == o) return true;
3             if (o == null || getClass() != o.getClass()) return false;
2

```

```

3         UserEntity that = (UserEntity) o;

3
3         if (id != that.id) return false;
3
4         if (nickname != null? !nickname.equals(that.nickname) : th
4             at.nickname != null) return false;
3
5         if (password != null? !password.equals(that.password) : th
5             at.password != null) return false;
3
6         if (firstName != null? !firstName.equals(that.firstName) :
6             that.firstName != null) return false;
3
7         if (lastName != null? !lastName.equals(that.lastName) : th
7             at.lastName != null) return false;
3
8
8         return true;
3
        }

9
        @Override
4
        public int hashCode() {
0
            int result = id;
4
            result = 31 * result + (nickname != null? nickname.hashCod
1                e() : 0);
4
            result = 31 * result + (password != null? password.hashCod
2                e() : 0);
4
            result = 31 * result + (firstName != null? firstName.hashC
4                ode() : 0);
3
            result = 31 * result + (lastName != null? lastName.hashCod
                e() : 0);

```

```

4         return result;
4     }

4     @OneToMany(mappedBy = "userByUserId")
5     public Collection<BlogEntity> getBlogsById() {
4         return blogsById;
6     }

4     public void setBlogsById(Collection<BlogEntity> blogsById) {
7         this.blogsById = blogsById;
4     }
}

```

3、配置数据库

既然数据库已经导入了，那么前期准备工作基本完成，还需要进行最终的配置。

首先，打开 `mvc-dispatcher-servlet.xml`，添加下列配置（如果某些地方报错，请选中并按 `Alt`

+ Insert 补全配置）：

```

?
1     <!-- 表示 JPA Repository 所在的包 -->
2     <jpa:repositories base-package="com.gaussic.repository"/>
3
4     <!-- 链接到 persistence.xml -->
5     <bean id="entityManagerFactory" class="org.springframework.orm.jpa.
6         LocalEntityManagerFactoryBean">
7         <property name="persistenceUnitName" value="defaultPersistenceU
            nit"/>
        </bean>

```

```

8
9         <!-- 事务管理 -->
10    <bean id="transactionManager" class="org.springframework.orm.jpa.Jp
11        aTransactionManager">
12        <property name="entityManagerFactory" ref="entityManagerFactory
13            "/>
14        </bean>
15
16        <!-- 开启事务管理注解 -->
17    <tx:annotation-driven transaction-manager="transactionManager"/>
18
19

```

讲解：

- (1) jpa:repositories：这一部分涉及到数据库的接口，将在后面详解；
- (2) entityManagerFactory：实体管理器工厂，读取 persistence.xml 配置；
- (3) transactionManager：事务管理器，利用 entityManager 进行事务管理；
- (4) tx:annotation-driven：打开事务管理器的注解驱动，可以使用注解的方法操纵数据库。

整体如下所示：

```

?
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        xmlns:jpa="http://www.springframework.org/schema/data/jpa"
7        xmlns:tx="http://www.springframework.org/schema/tx"

```



```

6      xsi:schemaLocation="http://www.springframework.org/schema/be
ans http://www.springframework.org/schema/beans/spring-beans.xsd
7      http://www.springframework.org/schema/context http://www.spr
ingframework.org/schema/context/spring-context.xsd
8      http://www.springframework.org/schema/mvc http://www.springf
ramework.org/schema/mvc/spring-mvc.xsd
9      http://www.springframework.org/schema/data/jpa http://www.sp
ringframework.org/schema/data/jpa/spring-jpa.xsd
1     http://www.springframework.org/schema/tx http://www.springfr
amework.org/schema/tx/spring-tx.xsd">
0
1         <!--指明 controller 所在包，并扫描其中的注解-->
2         <context:component-scan base-package="com.gaussic.controller"/>
1
1
2         <!-- 静态资源(js、image 等)的访问 -->
3         <mvc:default-servlet-handler/>
1
2
3         <!-- 开启注解 -->
4         <mvc:annotation-driven/>
1
2
3         <!--ViewResolver 视图解析器-->
4         <!--用于支持 Servlet、JSP 视图解析-->
5         <bean id="jspViewResolver" class="org.springframework.web.servle
1         t.view.InternalResourceViewResolver">
2         <property name="viewClass" value="org.springframework.web.se
3         rvlet.view.JstlView"/>
4         <property name="prefix" value="/WEB-INF/pages/" />
5         <property name="suffix" value=".jsp"/>
6         </bean>
7
8         <!-- 表示 JPA Repository 所在的包 -->
9         <jpa:repositories base-package="com.gaussic.repository"/>
0

```

```

1          <!-- 链接到 persistence.xml -->
2      <bean id="entityManagerFactory" class="org.springframework.orm.jp
9          pa.LocalEntityManagerFactoryBean">
2          <property name="persistenceUnitName" value="defaultPersisten
0              ceUnit"/>
1              </bean>
2
1          <!-- 事务管理 -->
2      <bean id="transactionManager" class="org.springframework.orm.jp
2          a.JpaTransactionManager">
2          <property name="entityManagerFactory" ref="entityManagerFact
2              ory"/>
1              </bean>
3
2          <!-- 开启事务管理注解 -->
4      <tx:annotation-driven transaction-
2          manager="transactionManager"/>
1      </beans>

```

下面，填充 persistence.xml，**将 persistence-unit 的 name 改为 defaultPersistenceUnit**

t. 在下面的文件中，我添加了一些更为详细的配置：

```

1      <?xml version="1.0" encoding="UTF-8"?>
2      <persistence xmlns="http://java.sun.com/xml/ns/persistence" version
3          ="2.0">
4
5          <persistence-unit name="defaultPersistenceUnit" transaction-
6              type="RESOURCE_LOCAL">
1              <provider>org.hibernate.ejb.HibernatePersistence</provider>
2              <properties>

```

```

7          <!-- 使用 MySQL 方言 -->
8      <property name="hibernate.dialect" value="org.hibernate
9          .dialect.MySQL5Dialect"/>
10      <!-- 数据库连接的 URL 地址 -->
11      <property name="hibernate.connection.url"
12          value="jdbc:mysql://localhost:3306/springdem
13          o"/>
14      <!-- 数据库连接的驱动 -->
15      <property name="hibernate.connection.driver_class" valu
16          e="com.mysql.jdbc.Driver"/>
17      <!-- 数据库连接的用户名 -->
18      <property name="hibernate.connection.username" value="r
19          oot"/>
20      <!-- 数据库连接的密码 -->
21      <property name="hibernate.connection.password" value="1
22          11111"/>
23      <!-- 显示 SQL 语句 -->
24      <property name="hibernate.show_sql" value="true"/>
25
26
27
28
29      <property name="hibernate.connection.useUnicode" value=
30          "true"/>
31      <property name="hibernate.connection.characterEncoding"
32          value="UTF-8"/>
33
34
35
36
37      <!-- 在显示 SQL 语句时格式化语句 -->
38      <property name="hibernate.format_sql" value="true"/>
39
40      <property name="hibernate.use_sql_comments" value="fals
41          e"/>
42      <!-- 自动输出 schema 创建 DDL 语句 -->
43      <property name="hibernate.hbm2ddl.auto" value="update"/
44          >

```

```

1
9      <!-- 数据库连接超时后自动重连 -->
      <property name="hibernate.connection.autoReconnect" val
2          ue="true"/>
0      <property name="connection.autoReconnectForPools" value
2          ="true"/>
      <property name="connection.is-connection-validation-
1          required" value="true"/>
2      </properties>
2      </persistence-unit>
      </persistence>

```

现在，重新启动 tomcat，如果没有报错，说明数据库已经配置完成了，接下来就要讲解数据库的相关开发工作。

更新：

阅读评论发现许多同学的 persistence.xml 出现了问题，因为出现问题的原因可能有很多，如果没有完全的报错以及代码的话，我这边很难解决问题，一个办法就是在 [GitHub Issues](#) 上面提问并贴出代码，我这边尽量解答。另一个办法就是下载最新的代码运行看有没有什么问题。

最后一个办法，尝试另外一种配置方法，无需 persistence.xml，直接在 mvc-dispatcher-servlet.xml 中配置数据库，如下所示：

```

?
1 <bean id="entityManagerFactory" class="org.springframework.orm.jpa.
2     LocalContainerEntityManagerFactoryBean">

```

```

3      <property name="persistenceUnitName" value="defaultPersistenceU
4          nit"/>
5      <property name="packagesToScan" value="com.gaussic.model" />
6          <property name="jpaVendorAdapter">
7              <bean class="org.springframework.orm.jpa.vendor.HibernateJp
8                  aVendorAdapter"/>
9              </property>
10              <property name="jpaProperties">
11                  <props>
12                      <prop key="hibernate.connection.driver_class">com.mysql
13                          .jdbc.Driver</prop>
14                      <prop key="hibernate.connection.url">jdbc:mysql://local
15                          host:3306/springdemo</prop>
16                      <prop key="hibernate.connection.username">root</prop>
17                      <prop key="hibernate.connection.password">111111</prop>
18                      <prop key="hibernate.show_sql">>false</prop>
19                      <prop key="hibernate.connection.useUnicode">>true</prop>
20                      <prop key="hibernate.connection.characterEncoding">UTF-
21                          8</prop>
22                      <prop key="hibernate.format_sql">>true</prop>
23                      <prop key="hibernate.use_sql_comments">>true</prop>
24                      <prop key="hibernate.hbm2ddl.auto">validate</prop>
25                      <prop key="hibernate.connection.autoReconnect">>true</pr
26                          op>
27                      <prop key="hibernate.dialect">org.hibernate.dialect.MyS
28                          QL5Dialect</prop>

```

```

1      <prop key="connection.autoReconnectForPools">true</prop>
7
      <prop key="connection.is-connection-validation-
1      required">true</prop>
      </props>
8    </property>
    </bean>
1

```

删除 persistence.xml，直接修改 entityManagerFactory bean 为如上图所示。这个方法可以拜托 persistence.xml 的困扰，但是有一个小小的问题，如果之前没有添加 Java EE Persistence 这个框架的，文中的 Persistence 工具栏将不会显示。一个解决办法就是，先修改 mvc-dispatcher-servlet，然后再添加 Java EE Persistence 框架，等能够看到 Persistence 工具栏后，删除 persistence.xml，余下的步骤可以继续操作。

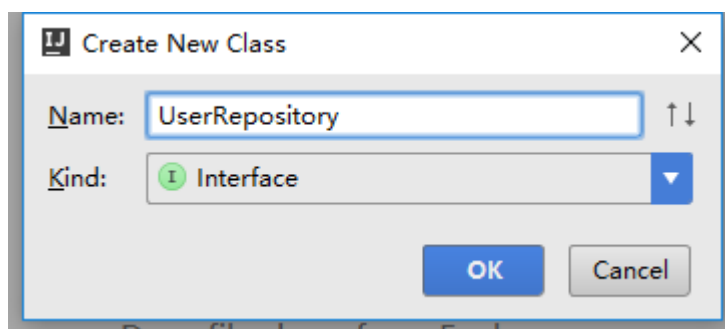
七、用户管理

既然我们要做一个博客管理系统，当然要首先实现我们的用户管理。在上一文中，我们已经配置好了数据库。接下来，就要实现网站的一些业务逻辑。

1、JPA 操作定义

在实现用户管理操作之前，需要讲解一下 JPA 的开发工作。

首先，在 com.gaussic.repository 包内新建一个 UserRepository 接口：



让该接口继承 JpaRepository：

```

?
5         package com.gaussic.repository;
6
7         import com.gaussic.model.UserEntity;
8         import org.springframework.data.jpa.repository.JpaRepository;
9         import org.springframework.stereotype.Repository;
1
10         /**
11         * Created by dzkan on 2016/3/8.
12         */
13         @Repository
14 public interface UserRepository extends JpaRepository<UserEntity, In
15         teger> {
16         }

```

在 JpaRepository 中，定义了几个简化的操作数据库的方法：

- (1) findAll()：查找表中所有记录；
- (2) findOne(Integer id)：按 id 来查找某一条记录；
- (3) findByXXX(Object xxx)：在这里 XXX 是一个字段名，根据该字段的值来查找所有记录；
- (4) save()和 delete()：添加一条记录以及删除一条记录。

除此之外，我们还可以在该 repository 中自定义新的方法，这将在稍后实际开发中提及。

2、后台管理

为了尽可能的在省去篇幅的情况下，在此省去管理员操作的开发。默认在访问 `/admin` 时，进入后台管理。

(1) 查看所有用户

将 MainController 补充为如下形式

```
1         package com.gaussic.controller;
2
3         import com.gaussic.model.UserEntity;
4         import com.gaussic.repository.UserRepository;
5         import org.springframework.beans.factory.annotation.Autowired;
6         import org.springframework.stereotype.Controller;
7         import org.springframework.ui.ModelMap;
8         import org.springframework.web.bind.annotation.RequestMapping;
9         import org.springframework.web.bind.annotation.RequestMethod;
10
11         import java.util.List;
12
13         /**
14          * Created by dzkan on 2016/3/8.
15          */
16         @Controller
17         public class MainController {
18
19             // 自动装配数据库接口，不需要再写原始的 Connection 来操作数据库
20             @Autowired
21             UserRepository userRepository;
22
23             @RequestMapping(value = "/", method = RequestMethod.GET)
24             public String index() {
25
26                 return "index";
27             }
28         }
```



```

1
6   @RequestMapping(value = "/admin/users", method = RequestMethod
      .GET)
1       public String getUsers(ModelMap modelMap) {
7           // 查询 user 表中所有记录
      List<UserEntity> userList = userRepository.findAll();
1
8           // 将所有记录传递给要返回的 jsp 页面，放在 userList 当中
      modelMap.addAttribute("userList", userList);
1
9           // 返回 pages 目录下的 admin/users.jsp 页面
      return "admin/users";
2       }
    }

```

讲解：

1. 自动装配：相当于数据库操作的极简化，只要定义了就可以直接进行数据操作，不用再去管开启连接、关闭连接等问题
2. 找到所有记录：使用 JpaRepository 的默认方法 findAll()。
3. modelMap：用于将 controller 方法里面的参数传递给所需的 jsp 页面，以进行相关显示。

现在，需要在 pages 下新建目录 admin，并新建 users.jsp 页面，以进行用户的管理：

```

?
1  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2      <!DOCTYPE html>
3      <html lang="zh-CN">
        <head>

```

```

4         <meta charset="utf-8">
5
6         <meta http-equiv="X-UA-Compatible" content="IE=edge">
7
8         <meta name="viewport" content="width=device-width, initial-
9             scale=1">
10        <!-- 上述 3 个 meta 标签*必须*放在最前面，任何其他内容都*必须*跟随其
11            后! -->
12        <title>SpringMVC 用户管理</title>
13
14        <!-- 新 Bootstrap 核心 CSS 文件 -->
15        <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css
16            /bootstrap.min.css">
17
18        <!--
19        - HTML5 shim and Respond.js for IE8 support of HTML5 elements and
20            media queries -->
21        <!--
22        - WARNING: Respond.js doesn't work if you view the page via file:/
23            / -->
24        <!--[if lt IE 9]>
25        <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"
26            ></script>
27        <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"
28            ></script>
29        <![endif]-->
30        </head>
31        <body>
32
33            <div class="container">
34
35                <h1>SpringMVC 博客系统-用户管理</h1>
36                <hr/>
37
38                <h3>所有用
39                    户 <a href="/admin/users/add" type="button" class="btn btn-
40                        primary btn-sm">添加</a></h3>
41
42                <!-- 如果用户列表为空 -->

```

```

1         <c:if test="${empty userList}">
8         <div class="alert alert-warning" role="alert">
1        <span class="glyphicon glyphicon-info-sign" aria-
9        hidden="true"></span>User 表为空，请
1        <a href="/admin/users/add" type="button" class="btn btn-primary btn-
2        sm">添加</a>
0        </div>
        </c:if>
2
2        <!-- 如果用户列表非空 -->
1        <c:if test="${!empty userList}">
2        <table class="table table-bordered table-striped">
2            <tr>
2                <th>ID</th>
3                <th>昵称</th>
3                <th>姓名</th>
2                <th>密码</th>
2                <th>操作</th>
4                </tr>
2
2        <c:forEach items="${userList}" var="user">
5            <tr>
2                <td>${user.id}</td>
2                <td>${user.nickname}</td>
2                <td>${user.firstName} ${user.lastName}</td>
6                <td>${user.password}</td>
                <td>
2                <a href="/admin/users/show/${user.id}" type
7                ="button" class="btn btn-sm btn-success">详情</a>
2                <a href="/admin/users/update/${user.id}" ty
8                pe="button" class="btn btn-sm btn-warning">修改</a>

```

```

2             <a href="/admin/users/delete/${user.id}" ty
9             pe="button" class="btn btn-sm btn-danger">删除</a>
3                 </td>
3                 </tr>
0                 </c:forEach>
0                 </table>
3                 </c:if>
3             </div>
1
3             <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->
3             <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
2             t>
3
3             <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
3             <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
3             ></script>
4             </body>
4             </html>

```

讲解：

1. <c>标签：在 jsp 中使用了 jstl 语法，可以方便地进行一些判断<c:if>与遍历操作<c:forEach>；


2. 页面使用了 Bootstrap，部分功能将在之后实现。

运行 Tomcat，在浏览器中输入 <http://localhost:8080/admin/users>，进入用户管理界面，显示

如下：

SpringMVC 博客系统-用户管理

所有用户

 User表为空, 请

由于目前数据库中没有数据，因而显示为空，现在需要向数据库中添加用户。

(2) 添加用户

在 MainController 中增加两个方法：

```
?  
  
        // get 请求，访问添加用户 页面  
1 @RequestMapping(value = "/admin/users/add", method = RequestMethod  
        .GET)  
2  
        public String addUser() {  
3  
        // 转到 admin/addUser.jsp 页面  
        return "admin/addUser";  
4  
        }  
5  
        // post 请求，处理添加用户请求，并重定向到用户管理页面  
6 @RequestMapping(value = "/admin/users/addP", method = RequestMetho  
        d.POST)  
7  
        public String addUserPost(@ModelAttribute("user") UserEntity userE  
8  
        ntity) {  
9        // 注意此处，post 请求传递过来的是一个 UserEntity 对象，里面包含了该  
        用户的信息  
1       // 通过@ModelAttribute()注解可以获取传递过来的'user'，并创建这个对  
        象  
0  
  
        // 数据库中添加一个用户，该步暂时不会刷新缓存
```

```

1          //userRepository.save(userEntity);

1          // 数据库中添加一个用户，并立即刷新缓存
1          userRepository.saveAndFlush(userEntity);

2          // 重定向到用户管理页面，方法为 redirect:url
1          return "redirect:/admin/users";

          }

```

讲解：

1. /admin/users/add 请求：get 请求转到添加用户页面
2. /admin/users/addP 请求：post 请求收集数据并存库
3. @ModelAttribute 注解：收集 post 过来的数据（在此，相当于 post 过来了一整个 userEntity，不用一个一个地取）
4. save()和 saveAndFlush()：save()方法处理完毕后，数据依然在缓冲区未写入数据库，使用 saveAndFlush()可以立即刷新缓冲区，写入数据库
5. redirect:/admin/users：这里使用重定向，可以让该方法重定向访问一个请求，return 之后，将跳转到 :/admin/users 所访问的页面。

现在，在 pages 目录下新建一个 addUser.jsp：

```

?
1 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
  <%@ taglib prefix="form" uri="http://www.springframework.org/tags/
2      form" %>

3      <!DOCTYPE html>

```

```

4         <html lang="zh-CN">
5             <head>
6                 <meta charset="utf-8">
7
8                 <meta http-equiv="X-UA-Compatible" content="IE=edge">
9
10                <meta name="viewport" content="width=device-width, initial-
11                    scale=1">
12                <!-- 上述 3 个 meta 标签*必须*放在最前面, 任何其他内容都*必须*跟随其
13                    后! -->
14                <title>SpringMVC 添加用户</title>
15
16                <!-- 新 Bootstrap 核心 CSS 文件 -->
17
18                <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css
19                    /bootstrap.min.css">
20
21                <!--
22                - HTML5 shim and Respond.js for IE8 support of HTML5 elements and
23                    media queries -->
24                <!--
25                - WARNING: Respond.js doesn't work if you view the page via file:/
26                    / -->
27                <!--[if lt IE 9]>
28                <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"></script>
29                <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>
30                <![endif]-->
31            </head>
32            <body>
33
34                <div class="container">
35
36                <h1>SpringMVC 添加用户</h1>
37                <hr/>
38
39                <form:form action="/admin/users/addP" method="post" commandName=
40
41                    "user" role="form">
42
43                    <div class="form-group">

```

```

1          <label for="firstName">Nickname:</label>
8          <input type="text" class="form-
1 control" id="nickname" name="nickname" placeholder="Enter Nickname:"
9          />
          </div>
2          <div class="form-group">
0          <label for="firstName">First Name:</label>
2          <input type="text" class="form-
1 control" id="firstName" name="firstName" placeholder="Enter FirstNam
2          e:"/>
          </div>
2          <div class="form-group">
2          <label for="lastName">Last Name:</label>
3          <input type="text" class="form-
2 control" id="lastName" name="lastName" placeholder="Enter LastName:"
4          />
          </div>
2          <div class="form-group">
5          <label for="password">Password:</label>
2          <input type="text" class="form-
6 control" id="password" name="password" placeholder="Enter Password:"
2          />
          </div>
7          <div class="form-group">
2          <button type="submit" class="btn btn-sm btn-success">提
8          交</button>
          </div>
          </form:form>

```



```

2         </div>

9         <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->
3 <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
0         t>

3         <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
1 <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
3         ></script>
3         </body>
2         </html>

```

讲解：

1. <form:form>标签：使用 Spring 的 form 标签，可以方便的收集整块数据，commandName="user" 说明 form 内的内容都保存在这个 user 实例中，然后将整个 user 实例传递给 controller 处理。在所有的 input 标签中，name 一定要与 User Entity 中的成员相同，不然无法找到。
2. 在提交之后，后台将会处理 /admin/users/add 请求。

现在，重新启动服务器，访问 <http://localhost:8080/admin/users/add> 页面，如下图所示：

← → ↻ 📄 localhost:8080/admin/users/add

SpringMVC 添加用户

Nickname:

Enter Nickname:

First Name:

Enter FirstName:

Last Name:

Enter LastName:

Password:

Enter Password:

提交

输入数据，点击提交，数据库中将会写入新的用户，重新跳转到用户管理页面：

← → ↻ 📄 localhost:8080/admin/users

SpringMVC 博客系统-用户管理

所有用户 [添加](#)

ID	昵称	姓名	密码	操作
1	Steve	Steve Jobs	123456	详情 修改 删除

(3) 查看用户详情

在 MainController 中加入查看详情操作：

```
?  
  
1 // 查看用户详情  
  // @PathVariable 可以收集 url 中的变量，需匹配的变量用{}括起来  
2 // 例如：访问 localhost:8080/admin/users/show/1，将匹配 id = 1  
  @RequestMapping(value = "/admin/users/show/{id}", method = RequestM  
3                      ethod.GET)
```

```

4 public String showUser(@PathVariable("id") Integer userId, ModelMap
5
6         modelMap) {
7
8         // 找到 userId 所表示的用户
9         UserEntity userEntity = userRepository.findOne(userId);
10
11        // 传递给请求页面
12        modelMap.addAttribute("user", userEntity);
13        return "admin/userDetail";
14    }

```

在 pages 目录下新建 userDetail.jsp :

```

?
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
1
2         <!DOCTYPE html>
3
4         <html lang="zh-CN">
5
6         <head>
7
8         <meta charset="utf-8">
9
10        <meta http-equiv="X-UA-Compatible" content="IE=edge">
11
12        <meta name="viewport" content="width=device-width, initial-
13                scale=1">
14        <!-- 上述 3 个 meta 标签*必须*放在最前面, 任何其他内容都*必须*跟随其
15                后! -->
16        <title>SpringMVC 用户详情</title>
17
18
19        <!-- 新 Bootstrap 核心 CSS 文件 -->
20        <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css
21                /bootstrap.min.css">
22
23
24        <!--
25        1 - HTML5 shim and Respond.js for IE8 support of HTML5 elements and
26                media queries -->

```

```

1         <!--
- WARNING: Respond.js doesn't work if you view the page via file:/
/ -->
2         <!--[if lt IE 9]>
1         <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"></script>
3         <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>
1         <![endif]-->
4         </head>
5         <body>
6         <div class="container">
1         <h1>SpringMVC 用户详情</h1>
2         <hr/>
3
4         <table class="table table-bordered table-striped">
5
6             <tr>
7                 <th>ID</th>
8                 <td>${user.id}</td>
9             </tr>
10            <tr>
11                <th>Nickname</th>
12                <td>${user.nickname}</td>
13            </tr>
14            <tr>
15                <th>First Name</th>
16                <td>${user.firstName}</td>
17            </tr>
18            <tr>
19                <th>Last Name</th>
20                <td>${user.lastName}</td>
21            </tr>
22            <tr>
23                <th>Password</th>
24                <td>${user.password}</td>
25            </tr>
26        </table>
27        </div>
28
29        <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->

```

```

2 <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
3
4 <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
5
2 <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
3
4 </script>
5 </body>
6 </html>

```

讲解：如何访问一个实例内的数据？

使用`${}`语法，在`{}`内可以使用类似 Java 的方法方便地访问数据。

重启服务器，进入 <http://localhost:8080/admin/users>，点击 ID = 1 的用户的 详情 按钮，可以查看用户详情：



SpringMVC 用户详情	
ID	1
Nickname	Steve
First Name	Steve
Last Name	Jobs
Password	123456

从 url 可以看出，访问的是 ID=1 的用户的详细情况，这样的 URL 采用了 REST 风格设计，看起来更加简便。

(4) 修改用户信息

现在我们要对用户信息做一定的修改，那该如何做呢。假设我们要能够修改全部的数据（除了 id），JpaRepository 未定义 update 方法，需要我们自己定义。

打开 UserRepository，添加 updateUser()接口方法：

?

```
1 package com.gaussic.repository;

2

3 import com.gaussic.model.UserEntity;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.data.jpa.repository.Modifying;
6 import org.springframework.data.jpa.repository.Query;
7 import org.springframework.data.repository.query.Param;
8 import org.springframework.stereotype.Repository;
9 import org.springframework.transaction.annotation.Transactional;

1 /**
0  * Created by dzkan on 2016/3/8.
1  */
1 @Repository
1 public interface UserRepository extends JpaRepository<UserEntity, Integer> {
1
2     @Modifying // 说明该方法是修改操作
1     @Transactional // 说明该方法是事务性操作
3         // 定义查询
4         // @Param 注解用于提取参数
1     @Query("update UserEntity us set us.nickname=:qNickname, us.firstName=:qFirstName, us.lastName=:qLastName, us.password=:qPassword
4         where us.id=:qId")
1     public void updateUser(@Param("qNickname") String nickname, @Param("qFirstName") String firstName,
5         @Param("qLastName") String qLastName, @Param("qPassword") String password, @Param("qId") Integer id);
```

```
1                                     }
```

在 MainController 中定义 update 操作方法：

```
?  
_
```

```
1                                     // 更新用户信息 页面  
2 @RequestMapping(value = "/admin/users/update/{id}", method = Reque  
    stMethod.GET)  
3 public String updateUser(@PathVariable("id") Integer userId, ModelM  
4     ap modelMap) {  
5  
6     // 找到 userId 所表示的用户  
7     UserEntity userEntity = userRepository.findOne(userId);  
8  
9     // 传递给请求页面  
10    modelMap.addAttribute("user", userEntity);  
11    return "admin/updateUser";  
12    }  
13  
14    // 更新用户信息 操作  
15 @RequestMapping(value = "/admin/users/updateP", method = RequestMe  
16     thod.POST)  
17 public String updateUserPost(@ModelAttribute("userP") UserEntity us  
18     er) {  
19  
20    // 更新用户信息  
21    userRepository.updateUser(user.getNickname(), user.getFirstNam  
22        e(),  
23        user.getLastName(), user.getPassword(), user.getId());  
24    userRepository.flush(); // 刷新缓冲区  
25  
26    return "redirect:/admin/users";  
27    }  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100
```

然后，在 pages 目录下，新建 updateUser.jsp 文件：

```
?
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/
1      form" %>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
2      <!DOCTYPE html>
3      <html lang="zh-CN">
4      <head>
5      <meta charset="utf-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-
      scale=1">
8      <!-- 上述 3 个 meta 标签*必须*放在最前面，任何其他内容都*必须*跟随其
      后! -->
9      <title>SpringMVC Demo 更新用户</title>
1
      <!-- 新 Bootstrap 核心 CSS 文件 -->
0 <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css
1      /bootstrap.min.css">
1
      <!--
1 - HTML5 shim and Respond.js for IE8 support of HTML5 elements and
      media queries -->
2      <!--
- WARNING: Respond.js doesn't work if you view the page via file:/
1      / -->
      <!--[if lt IE 9]>
3      <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"
      ></script>
1      <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"
      ></script>
4      <![endif]-->
      </head>
```



```

1          <body>
2
3          <div class="container">
4
5          <h1>SpringMVC 更新用户信息</h1>
6          <hr/>
7
8          <form:form action="/admin/users/updateP" method="post" commandName="userP" role="form">
9
10         <div class="form-group">
11
12         <label for="firstName">Nickname:</label>
13
14         <input type="text" class="form-control" id="nickname" name="nickname" placeholder="Enter Nickname:"
15         value="${user.nickname}"/>
16         </div>
17
18         <div class="form-group">
19
20         <label for="firstName">First Name:</label>
21
22         <input type="text" class="form-control" id="firstName" name="firstName" placeholder="Enter FirstName:"
23         value="${user.firstName}"/>
24         </div>
25
26         <div class="form-group">
27
28         <label for="lastName">Last Name:</label>
29
30         <input type="text" class="form-control" id="lastName" name="lastName" placeholder="Enter LastName:"
31         value="${user.lastName}"/>
32         </div>
33
34         <div class="form-group">

```

```

2          <label for="password">Password:</label>
6          <input type="text" class="form-
2 control" id="password" name="password" placeholder="Enter Password:"
          value="{user.password}"/>
7          </div>
2          <!-- 把 id 一并写入 userP 中 -->
          <input type="hidden" id="id" name="id" value="{user.id}"/>
8
2
          <div class="form-group">
9          <button type="submit" class="btn btn-sm btn-success">提
3          交</button>
          </div>
0          </form:form>
3          </div>
1
3          <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->
2<script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
3          t>
3
          <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
3<script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
4          ></script>
          </body>
3          </html>

```

重启服务器，进入 <http://localhost:8080/admin/users>，点击第一个用户的 修改 按钮，做如下修改：

← → ↻

localhost:8080/admin/users/update/1

☆ ≡

SpringMVC 更新用户信息

Nickname:

Steve

First Name:

Steve

Last Name:

NoJob

Password:

gaussic

提交

提交后，重新跳转回用户管理页面，可发现修改完成：

← → ↻

localhost:8080/admin/users

☆ ≡

SpringMVC 博客系统-用户管理

所有用户

添加

ID	昵称	姓名	密码	操作
1	Steve	Steve NoJob	gaussic	<div>详情</div> <div>修改</div> <div>删除</div>

(5) 删除用户

现在，新添加一个用户：

← → ↻

localhost:8080/admin/users

☆ ≡

SpringMVC 博客系统-用户管理

所有用户

添加

ID	昵称	姓名	密码	操作
1	Steve	Steve NoJob	gaussic	<div>详情</div> <div>修改</div> <div>删除</div>
2	microsoft11	Bill Gates	gaussic	<div>详情</div> <div>修改</div> <div>删除</div>

现在我们需要删掉新加入的用户，打开 MainController，加入以下方法：

?

1

// 删除用户

```

2 @RequestMapping(value = "/admin/users/delete/{id}", method = RequestMethod.GET)
3     public String deleteUser(@PathVariable("id") Integer userId) {
4
5         // 删除 id 为 userId 的用户
6         userRepository.delete(userId);
7         // 立即刷新
8         userRepository.flush();
9         return "redirect:/admin/users";
10    }
11
12
13
14
15
16
17
18
19
20

```

重启服务器，进入 <http://localhost:8080/admin/users>，点击 ID=2 的用户的删除按钮，在 controller 中处理完之后，将跳转回用户管理界面：



这样，增删该查基本完成了。

其实，更到这里，基本就可以开始开发工作了，还有一些其他的功能，都需要通过平时的积累以及多查资料来完成。例如 JSON 数据的处理，异步请求的处理，以及相关外键等操作。

八、博客文章管理

博客的管理与用户的管理有许多的相似之处，但是另外多了外键的操作，下面做简单的说明。

1、查看文章

查看文章的操作相对简单。首先在 com.gaussic.repository 中添加 BlogRepository，方法与 UserRepository 类似：

```
?  
1 package com.gaussic.repository;  
2  
3 import com.gaussic.model.BlogEntity;  
4 import org.springframework.data.jpa.repository.JpaRepository;  
5 import org.springframework.stereotype.Repository;  
6  
7 @Repository  
8 public interface BlogRepository extends JpaRepository<BlogEntity, Integer> {  
9     }  
}
```

在 com.gaussic.controller 中添加一个 BlogController 类，并添加以下方法（当然也可以写在 MainController 中，在较大型的项目开发中，最好对各类的操作进行一个区分，以增强代码的可读性）：

```
?  
1 package com.gaussic.controller;  
2  
3 import com.gaussic.model.BlogEntity;
```

```

4         import com.gaussic.repository.BlogRepository;
5     import org.springframework.beans.factory.annotation.Autowired;
6         import org.springframework.stereotype.Controller;
7         import org.springframework.ui.ModelMap;
8     import org.springframework.web.bind.annotation.RequestMapping;
9     import org.springframework.web.bind.annotation.RequestMethod;
10
11         import java.util.List;
12
13         @Controller
14         public class BlogController {
15
16             @Autowired
17             BlogRepository blogRepository;
18
19             // 查看所有博文
20             @RequestMapping(value = "/admin/blogs", method = RequestMethod.GET)
21             public String showBlogs(ModelMap modelMap) {
22                 List<BlogEntity> blogList = blogRepository.findAll();
23                 modelMap.addAttribute("blogList", blogList);
24                 return "admin/blogs";
25             }
26         }

```

接下来，在 pages/admin 目录下新建 blogs.jsp 文件：

?

```

1  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
4
5      <!DOCTYPE html>
6
7      <html lang="zh-CN">
8          <head>
9              <meta charset="utf-8">
10
11              <meta http-equiv="X-UA-Compatible" content="IE=edge">
12
13              <meta name="viewport" content="width=device-width, initial-
14                  scale=1">
15              <!-- 上述 3 个 meta 标签*必须*放在最前面, 任何其他内容都*必须*跟随其
16                  后! -->
17              <title>SpringMVC 博客管理</title>
18
19
20              <!-- 新 Bootstrap 核心 CSS 文件 -->
21
22              <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css/
23                  bootstrap.min.css">
24
25
26              <!--
27              - HTML5 shim and Respond.js for IE8 support of HTML5 elements and m
28                  edia queries -->
29              <!--
30              - WARNING: Respond.js doesn't work if you view the page via file://
31                  -->
32              <!--[if lt IE 9]>
33                  <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js
34                      "></script>
35                  <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"
36                      ></script>
37                  <![endif]-->
38              </head>
39              <body>
40
41                  <div class="container">
42
43                      <h1>SpringMVC 博客系统-博客管理</h1>
44                      <hr/>

```

```

1
6          <h3>所有博
客 <a href="/admin/blogs/add" type="button" class="btn btn-
1          primary btn-sm">添加</a></h3>
7
1          <!-- 如果用户列表为空 -->
1          <c:if test="${empty blogList}">
8
1          <div class="alert alert-warning" role="alert">
1          <span class="glyphicon glyphicon-info-sign" aria-
9          hidden="true"></span>Blog 表为空，请
2 <a href="/admin/blogs/add" type="button" class="btn btn-primary btn-
0          sm">添加</a>
          </div>
2          </c:if>
1
1          <!-- 如果用户列表非空 -->
2          <c:if test="${!empty blogList}">
2
2          <table class="table table-bordered table-striped">
2          <tr>
2          <th>ID</th>
3          <th>标题</th>
2          <th>作者</th>
2          <th>发布日期</th>
4          <th>操作</th>
2
2          </tr>
2
2          <c:forEach items="${blogList}" var="blog">
5          <tr>
2          <td>${blog.id}</td>
2          <td>${blog.title}</td>
6          <td>${blog.userByUserId.nickname}, ${blog.userB
yUserId.firstName} ${blog.userByUserId.lastName}</td>

```



```

2         <td><fmt:formatDate value="${blog.pubDate }" pat
7         tern="yyyy-MM-dd"/></td>
           <td>
2             <a href="/admin/blogs/show/${blog.id}" type=
8             "button" class="btn btn-sm btn-success">详情</a>
2             <a href="/admin/blogs/update/${blog.id}" typ
9             e="button" class="btn btn-sm btn-warning">修改</a>
3             <a href="/admin/blogs/delete/${blog.id}" typ
0             e="button" class="btn btn-sm btn-danger">删除</a>
           </td>
3       </tr>
         </c:forEach>
1       </table>
3       </c:if>
         </div>
2
3       <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->
       <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
3       t>
3
4       <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
       <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
3       ></script>
5       </body>
       </html>

```

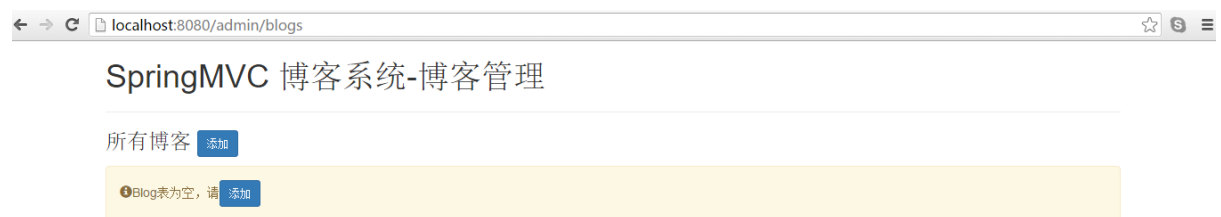
先不要急着运行代码，我们来看看 blogs.jsp 与 users.jsp 有何不同。

注意到，在查看博文作者的时候，使用了如下代码：

?

```
1 <td>${blog.userById.nickname}, ${blog.userById.firstName} ${blog.userById.lastName}</td>
```

也就是说，通过 blog 的 userById 对象，找到了博文的作者，并且输出了他的昵称以及姓名。在这里，外键起到了决定性的作用。下面我们运行 Tomcat，浏览器中输入 <http://localhost:8080/admin/blogs>，可以看到如下界面：



2、添加博客

当然，现在数据库中是没有数据的，不用着急，我们先实现博文的添加功能。回到 BlogController，添加 addBlog 的 GET 和 POST 操作：

```
?
package com.gaussic.controller;

import com.gaussic.model.BlogEntity;
import com.gaussic.model.UserEntity;
import com.gaussic.repository.BlogRepository;
import com.gaussic.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.ModelAttribute;
```

```

1  import org.springframework.web.bind.annotation.RequestMapping;
0  import org.springframework.web.bind.annotation.RequestMethod;

1

1          import java.util.List;

1

2          @Controller
          public class BlogController {

1

3          @Autowired
          BlogRepository blogRepository;

1

4          @Autowired
          UserRepository userRepository;

1

5          // 查看所有博文
1  @RequestMapping(value = "/admin/blogs", method = RequestMethod
          .GET)

6          public String showBlogs(ModelMap modelMap) {
1          List<BlogEntity> blogList = blogRepository.findAll();
          modelMap.addAttribute("blogList", blogList);

7          return "admin/blogs";

1          }

8

1          // 添加博文
9  @RequestMapping(value = "/admin/blogs/add", method = RequestMe
          thod.GET)

2          public String addBlog(ModelMap modelMap) {
0          List<UserEntity> userList = userRepository.findAll();
          // 向 jsp 注入用户列表
          modelMap.addAttribute("userList", userList);

```

```

2         return "admin/addBlog";
1     }

2         // 添加博文，POST 请求，重定向为查看博客页面
2     @RequestMapping(value = "/admin/blogs/addP", method = RequestMethod.POST)
2     public String addBlogPost(@ModelAttribute("blog") BlogEntity blogEntity) {
3         // 打印博客标题
2         System.out.println(blogEntity.getTitle());
2         // 打印博客作者
4         System.out.println(blogEntity.getUserByUserId().getNickname());
2         // 入库
2         blogRepository.saveAndFlush(blogEntity);
5         // 重定向地址
2         return "redirect:/admin/blogs";
2     }
6 }

```

接下来，在 pages/admin 目录下新建 addBlog.jsp 文件：

```

?
1 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2 <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
3 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4 <!DOCTYPE html>
5 <html lang="zh-CN">
6 <head>
7 <meta charset="utf-8">
8 <meta http-equiv="X-UA-Compatible" content="IE=edge">

```

```

8      <meta name="viewport" content="width=device-width, initial-
9          scale=1">
10      <!-- 上述 3 个 meta 标签*必须*放在最前面, 任何其他内容都*必须*跟随其
11          后! -->
12      <title>SpringMVC 添加博客</title>
13
14      <!-- 新 Bootstrap 核心 CSS 文件 -->
15      <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css/
16          bootstrap.min.css">
17
18      <!--
19      - HTML5 shim and Respond.js for IE8 support of HTML5 elements and
20          media queries -->
21      <!--
22      - WARNING: Respond.js doesn't work if you view the page via file:/
23          / -->
24      <!--[if lt IE 9]>
25      <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"
26          ></script>
27      <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"
28          ></script>
29      <![endif]-->
30      </head>
31      <body>
32
33          <div class="container">
34
35              <h1>SpringMVC 添加博客</h1>
36              <hr/>
37
38              <form:form action="/admin/blogs/addP" method="post" commandName=
39                  "blog" role="form">
40
41                  <div class="form-group">
42
43                      <label for="title">Title:</label>
44
45                      <input type="text" class="form-
46
47                      control" id="title" name="title" placeholder="Enter Title:"/>
48                  </div>

```

```

2         <div class="form-group">
0             <label for="userByUserId.id">Author:</label>
2                 <select class="form-
1 control" id="userByUserId.id" name="userByUserId.id">
2                     <c:forEach items="${userList}" var="user">
2                         <option value="${user.id}">${user.nickname}, ${
2 user.firstName} ${user.lastName}</option>
2                             </c:forEach>
3                             </select>
3                             </div>
2                     <div class="form-group">
4                         <label for="content">Content:</label>
2                             <textarea class="form-
5 control" id="content" name="content" rows="3" placeholder="Please Inp
2 ut Content"></textarea>
2                             </div>
6                             <div class="form-group">
2                                 <label for="pubDate">Publish Date:</label>
7                                     <input type="date" class="form-
2 control" id="pubDate" name="pubDate"/>
8                                         </div>
8                                         <div class="form-group">
2                                             <button type="submit" class="btn btn-sm btn-success">提
9 交</button>
3                                             </div>
0                                             </form:form>
0                                             </div>

```

```

3         <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->
        <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
1
            t>
3
2         <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
        <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
3
            ></script>
            </body>
            </html>

```

讲解：

（1）首先在作者一栏使用了选择框，通过 select 来选择该博文作者，注意到 select 标签的 id 和 name 都是 userByUserId.id（id 可以不同，但 name 必须如此），也就是说，需要通过 blog 的外键来定位到所需要选择的作者。而在其选项组中，使用 user.id 来进行赋值，这样，就能把 blog 和 user 表相关联，是不是很方便呢？

（2）Content 处使用了 textarea 标签，关于文中的一些标签的用法可以[参照 Bootstrap 中文官网](#)（没有 Bootstrap 实在不会写前端。。），注意由于数据表的限制，请将字数保存在 255 以下。当然也可以把数据表中的字段改为 TEXT，以支持更长的输入。

（3）发布日期的选取，采用了最简单的 H5 date 控件，有兴趣做成选择框的话，可以引入[Bootstrap Datetimepicker](#)，这是一个比较好的组件，但不是本文的重点，在此使用最简单的。点击其右方的下三角，可以选择日期，也可以直接输入：

说了真么多，我们来重启一下 Tomcat，点击博客管理界面的添加按钮，添加以下内容：

← → ↻ localhost:8080/admin/blogs/add ☆ Ⓢ ≡

SpringMVC 添加博客

Title:

Author:

Content:

Publish Date:

注意 Author 是一个 select 选框，如下图所示，（如果选项很少效果不太好的话，请自行到用户管理界面多添加几个用户再来）：

Title:

Author:

Steve, Steve NoJob

Steve, Steve NoJob

Bill, Bill Gates

Mark, Mark Zuckerberg

gauss, Gaussic D

Publish Date:

点击提交，系统重新跳转到了博客管理界面，这里已经显示出了所添加的博客列表：

← → ↻ localhost:8080/admin/blogs ☆ Ⓢ ≡

SpringMVC 博客系统-博客管理

所有博客

ID	标题	作者	发布日期	操作
1	SpringMVC教程	gauss, Gaussic D	2016-03-18 00:00:00.0	<input type="button" value="详情"/> <input type="button" value="修改"/> <input type="button" value="删除"/>

3、查看博文详情

有了前面的基础，这个就很好实现了，不多说，趁热打铁，在 BlogController 中添加如下方法：

?


```

1
2      // 查看博文详情，默认使用 GET 方法时，method 可以缺省
      @RequestMapping("/admin/blogs/show/{id}")
3 public String showBlog(@PathVariable("id") int id, ModelMap modelMap
4                        ) {
      BlogEntity blog = blogRepository.findOne(id);
      modelMap.addAttribute("blog", blog);
5
      return "admin/blogDetail";
6
      }
7

```

在 pages/admin 目录下新建文件 blogDetail.jsp :

```

?
1 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
2
      <!DOCTYPE html>
3
      <html lang="zh-CN">
4
          <head>
5
              <meta charset="utf-8">
6
              <meta http-equiv="X-UA-Compatible" content="IE=edge">
7
              <meta name="viewport" content="width=device-width, initial-
                  scale=1">
8      <!-- 上述 3 个 meta 标签*必须*放在最前面，任何其他内容都*必须*跟随其
                  后! -->
9      <title>SpringMVC 博文详情</title>
1
1
          <!-- 新 Bootstrap 核心 CSS 文件 -->
0 <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css/
1
          bootstrap.min.css">
1

```

```

1             <!--
- HTML5 shim and Respond.js for IE8 support of HTML5 elements and
2             media queries -->
3             <!--
- WARNING: Respond.js doesn't work if you view the page via file:/
1             / -->
2             <!--[if lt IE 9]>
3             <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js"
1             <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js
4             "></script>
5             <![endif]-->
6             </head>
7             <body>
8
9             <div class="container">
1            <h1>SpringMVC 博文详情</h1>
2            <hr/>
3
4            <table class="table table-bordered table-striped">
5
6                <tr>
7                    <th>ID</th>
8                    <td>${blog.id}</td>
9                </tr>
10               <tr>
11                   <th>Title</th>
12                   <td>${blog.title}</td>
13               </tr>
14               <tr>
15                   <th>Author</th>
16                   <td>${blog.userByUserId.nickname}, ${blog.userByUserId
17                   .firstName} ${blog.userByUserId.lastName}</td>
18               </tr>
19               <tr>
20                   <th>Content</th>
21                   <td>${blog.content}</td>
22               </tr>
23               <tr>
24                   <th>Publish Date</th>
25                   <td><fmt:formatDate value="${blog.pubDate}" pattern="yyy
26                   y 年 MM 月 dd 日"/></td>
27               </tr>

```

```

2         </table>
        </div>

3

2         <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->
        <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
4            t>

2

5         <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
        <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
2            ></script>
        </body>
6        </html>

2

```

注意：在输出日期的时候使用了 fmt 标签，请在顶部引入 fmt 标签。

现在重启服务器，进入博客管理页面，点击刚才添加的博文的详情按钮，查看该博文的详情：



4、修改博客内容

写完了增加和查看的操作，现在实现修改的操作，内容依旧与用户操作类似。首先，在 Blog Repository 中添加如下代码：

```

?
1 // 修改博文操作
    @Modifying
    @Transactional

```

```

1 @Query("update BlogEntity blog set blog.title=:qTitle, blog.userByU
2         serId.id=:qUserId," +
3         " blog.content=:qContent, blog.pubDate=:qPubDate where blog
4         .id=:qId")
5 void updateBlog(@Param("qTitle") String title, @Param("qUserId") int
6         userId, @Param("qContent") String content,
7         @Param("qPubDate") Date pubDate, @Param("qId") int i
            d);

```

接下来，在 BlogController 中添加修改博文的 GET 和 POST 方法：

```

1 // 修改博文内容，页面
2 @RequestMapping("/admin/blogs/update/{id}")
3 public String updateBlog(@PathVariable("id") int id, ModelMap model
4         Map) {
5     // 是不是和上面那个方法很像
6     BlogEntity blog = blogRepository.findOne(id);
7     List<UserEntity> userList = userRepository.findAll();
8     modelMap.addAttribute("blog", blog);
9     modelMap.addAttribute("userList", userList);
10    return "admin/updateBlog";
11 }
12
13 // 修改博客内容，POST 请求
14 @RequestMapping(value = "/admin/blogs/updateP", method = RequestMe
15         thod.POST)
16 public String updateBlogP(@ModelAttribute("blogP") BlogEntity blogE
17         ntity) {
18     // 更新博客信息
19     blogRepository.updateBlog(blogEntity.getTitle(), blogEntity.ge
20         tUserByUserId().getId(),
21     blogEntity.getContent(), blogEntity.getPubDate(), blog
22     Entity.getId());
23     blogRepository.flush();
24     return "redirect:/admin/blogs";

```

}

新建 updateBlog.jsp 页面：

?

```
1  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
2  <%@ taglib prefix="form" uri="http://www.springframework.org/tags/f
    orm" %>
3  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
4  <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
5
6  <!DOCTYPE html>
7
8  <html lang="zh-CN">
9  <head>
10 <meta charset="utf-8">
11
12 <meta http-equiv="X-UA-Compatible" content="IE=edge">
13
14 <meta name="viewport" content="width=device-width, initial-
15     scale=1">
16 <!-- 上述 3 个 meta 标签*必须*放在最前面，任何其他内容都*必须*跟随其
17     后! -->
18 <title>SpringMVC 修改博客</title>
19
20 <!-- 新 Bootstrap 核心 CSS 文件 -->
21 <link rel="stylesheet" href="//cdn.bootcss.com/bootstrap/3.3.5/css/
22     bootstrap.min.css">
23
24 <!--
25 - HTML5 shim and Respond.js for IE8 support of HTML5 elements and m
26     edia queries -->
27 <!--
28 - WARNING: Respond.js doesn't work if you view the page via file://
29     -->
30 <!--[if lt IE 9]>
31 <script src="//cdn.bootcss.com/html5shiv/3.7.2/html5shiv.min.js
32     "></script>
33 <script src="//cdn.bootcss.com/respond.js/1.4.2/respond.min.js"
34     ></script>
35 <![endif]-->
```

```

1         </head>
        <body>
5         <div class="container">
            <h1>SpringMVC 修改博客</h1>
1            <hr/>
6        <form:form action="/admin/blogs/updateP" method="post" commandNa
1            me="blogP" role="form">
7            <div class="form-group">
1                <label for="title">Title:</label>
8                <input type="text" class="form-
1control" id="title" name="title" placeholder="Enter Title:" value="{
9                    blog.title}"/>
                </div>
2                <div class="form-group">
0                    <label for="userByUserId.id">Author:</label>
2                    <select class="form-
1control" id="userByUserId.id" name="userByUserId.id">
2                        <c:forEach items="{userList}" var="user">
2                            <c:if test="{user.id==blog.userByUserId.id}">
2                                <option value="{user.id}" selected="selecte
3d">{user.nickname}, {user.firstName} {user.lastName}</option>
                                    </c:if>
2                            <c:if test="{user.id!=blog.userByUserId.id}">
4                                <option value="{user.id}">{user.nickname}
2                                , {user.firstName} {user.lastName}</option>
                                    </c:if>
5                                </c:forEach>
                                    </select>
                                </div>

```

```

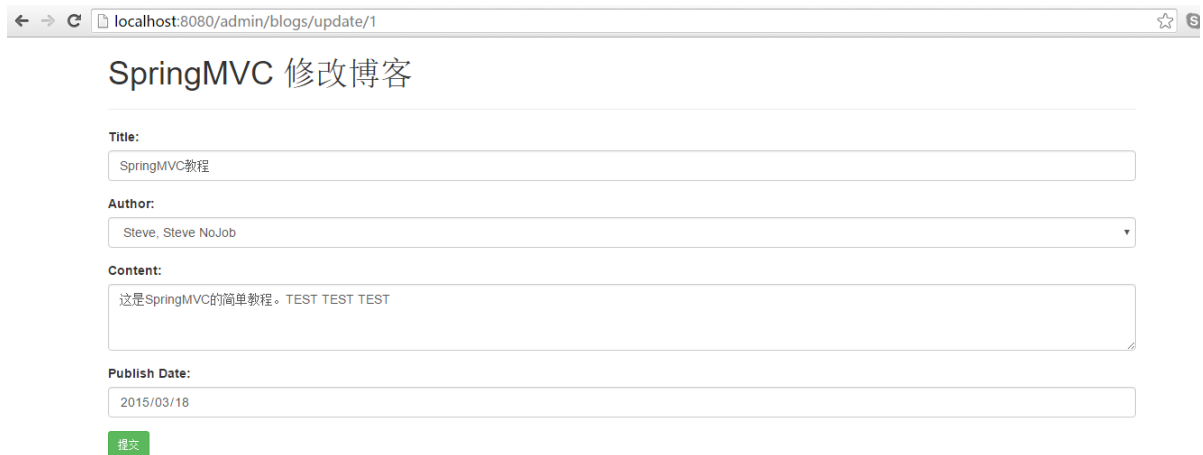
2         <div class="form-group">
6             <label for="content">Content:</label>
2
2             <textarea class="form-
7                 control" id="content" name="content" rows="3"
2                 placeholder="Please Input Content">${blog.content}</textarea>
2                 </div>
8             <div class="form-group">
2                 <label for="pubDate">Publish Date:</label>
9
2                 <input type="date" class="form-
3                     control" id="pubDate" name="pubDate"
0                     value="<fmt:formatDate value='${blog.pubDate }' pattern='yyyy-MM-
                        dd' />" />
3                     </div>
3                     <!-- 把 id 一并写入 blogP 中 -->
1                     <input type="hidden" id="id" name="id" value='${blog.id}' />
3
3                     <div class="form-group">
2                         <button type="submit" class="btn btn-sm btn-success">提
3                             交</button>
3                             </div>
3                             </form:form>
3                             </div>
3
4                     <!-- jQuery 文件。务必在 bootstrap.min.js 之前引入 -->
4                     <script src="//cdn.bootcss.com/jquery/1.11.3/jquery.min.js"></scrip
3                         t>
5
3                     <!-- 最新的 Bootstrap 核心 JavaScript 文件 -->
3                     <script src="//cdn.bootcss.com/bootstrap/3.3.5/js/bootstrap.min.js"
6                         ></script>
6                         </body>

```

</html>

注：感谢某位同学的指出，这里 pubDate 的输出如图下图所示：

重启服务器，进入修改博客页面，做出一定的修改：



点击提交，返回博客列表页面，可以查看修改。



ID	标题	作者	发布日期	操作
1	SpringMVC教程	Bill, Bill Gates	2015-03-18 00:00:00.0	详情 修改 删除

5、删除博客文章

删除博客的实现非常简单，在 BlogController 下加入以下方法：

```
?  
1 // 删除博客文章  
  @RequestMapping("/admin/blogs/delete/{id}")  
2 public String deleteBlog(@PathVariable("id") int id) {  
3     blogRepository.delete(id);  
    blogRepository.flush();  
4     return "redirect:/admin/blogs";  
5 }
```


6

7

重启服务器，随便添加一篇新的文章，然后删除之：

SpringMVC 博客系统-博客管理

所有博客 [添加](#)

ID	标题	作者	发布日期	操作
1	SpringMVC教程	Bill, Bill Gates	2015-03-18 00:00:00.0	详情 修改 删除
2	Spring MVC教程2	Steve, Steve NoJob	2016-03-20 00:00:00.0	详情 修改 删除

点击删除按钮，删除第二篇文章，将返回博客管理界面：



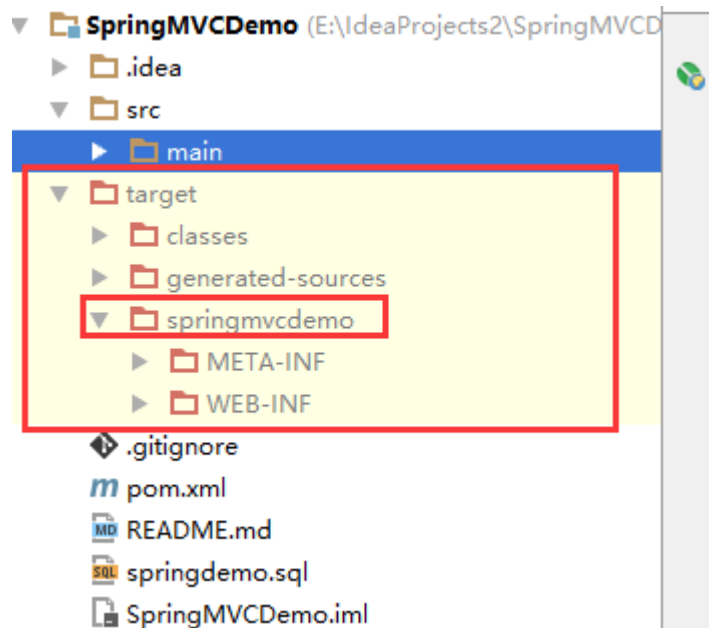
九、尾声

这样，整个博客的增删改查操作就完成了，而这一系列的文章也即将接近尾声。还有许多的细节是可以优化的，SpringMVC 还有许多优化代码的小技巧，能让你在开发时加省力，这一点是要在我们的学习和使用中去探索和思考的，特别作为一个 WEB 开发人员，探索和思考的能力是宝贵的。

在此，还有一些小的事情需要交代，让我一一道来。

1、如何部署

在项目的目录下，IntelliJ IDEA 生成了一个 target 文件夹，如下图所示：



而 springmvcdemo 就是生成好的项目，我们把这个项目放到 Tomcat 的 webapps 目录下，然后启动 Tomcat，访问 <http://localhost:8080/springmvcdemo> 就可以看到网站。

2、进一步的学习

如果本文满足不了你的需求，你还需要许多更高级的操作，那么就应该查查文档了，访问 <http://projects.spring.io/spring-framework/> 可以查看 springmvc 的详细文档，按照所使用的版本进行查阅。在 <http://spring.io/projects> 中还有许多其他的 Spring 框架，比如比较流行的 Spring BOOT 框架，以及本文中用到的 Spring Data 框架等。

关于 Bootstrap，在前端开发上面，离了 Bootstrap 我还真难写出了像样的前端来，当然为了成为一个出色的 Full Stack Developer，会一点 HTML+CSS+JS 那肯定是有必要的。我维护过 PHP 的项目，开发过 Django 的项目，SpringMVC 的项目也做了不少，甚至乎用 Node.js 搭建博客等都有一定的涉猎，这些项目无论哪一个都离不开前端知识的支持。如果对前端不太了解，又想速成的话，建议上 [Bootstrap 中文官网](#) 看看教程。

3、ENDING

如果你对这几篇文章有什么问题、意见、或者建议的话，欢迎访问我的 [GitHub 博客](#) 提出，
我会在空余时间帮助大家解答。

。