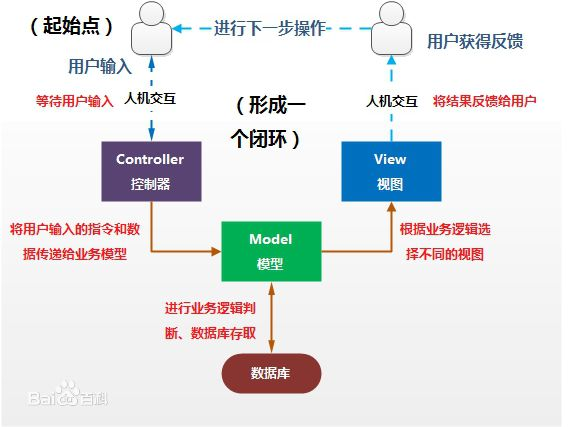
基于Springboot搭建及相关拓展

1. 关于MVC的介绍



描述：起点有一个用户，在客户端输入所需要提交的内容，然后进入到控制器；控制器根据用户输入的指令和数据传递给业务模型；在业务模型中，系统根据所需要的数据加载到模型中，然后提交到数据库进行查询操作，然后数据库将查询到的数据返回给系统，系统通过模型整合成一个视图，也就是用户所需要的数据，将这些数据通过客户端返回给用户得到反馈，这样就形成了整个MVC的闭环。

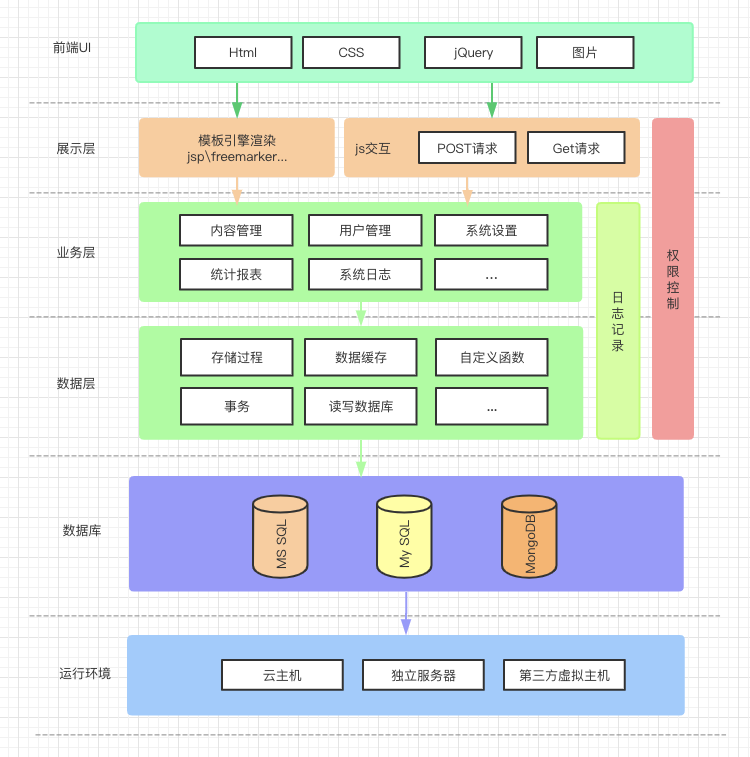
MVC各个部分的大致作用：

控制器：传递客户数据和请求

模型：接收客户和处理后的数据容器

视图：显示处理后数据的页面

要实现MVC则需要用到这样一些组件



首先大家看到的是前端Html，CSS，JQuery和图文；然后在展示层是通过jsp\freemaker等前端引擎渲染成Html展现给用户，然后有一些数据交互的逻辑放在js中，比如一些提交post请求、get请求，拿到数据后返回给前端；这些请求是要落在业务层，业务层会根据业务需求，分成内容管理，用户管理，系统，设置，统计报表，系统日志，每一个都可以理解为是平行的，就是需要什么样的业务，就在业务层写这么一个模块。然后这些业务模块统统在数据层，通过存储过程，数据缓存，自定义函数，读写数据库等各种各样的方式从数据库读取到数据，最终一层一层网上返回，返回到前端给用户。在这中间，为了便于管理还会生成日志，对权限做一些控制。最终底层的数据库是部署在服务器上的，服务器最简单的就是个人pc电脑，也可以是云上的主机（阿里云服务器，腾讯云服务器或者自己在机房搭建的独立服务器等），但是现在比较流行的是第三方虚拟机，这样就可以在一台物理主机内存CPU支持的范围内，虚拟出很多虚拟主机。

2、Springboot

1．基本环境和工具

它是java j2e衍生出来的一个更为便利的部署框架

搭建环境有jdk1.8（java的编译环境），mysql5.5以上（数据库的存储）

开发工具使用IntelliJ IDEA（Eclipse、vscode），该文档是基于IDEA进行的Springboot的搭建

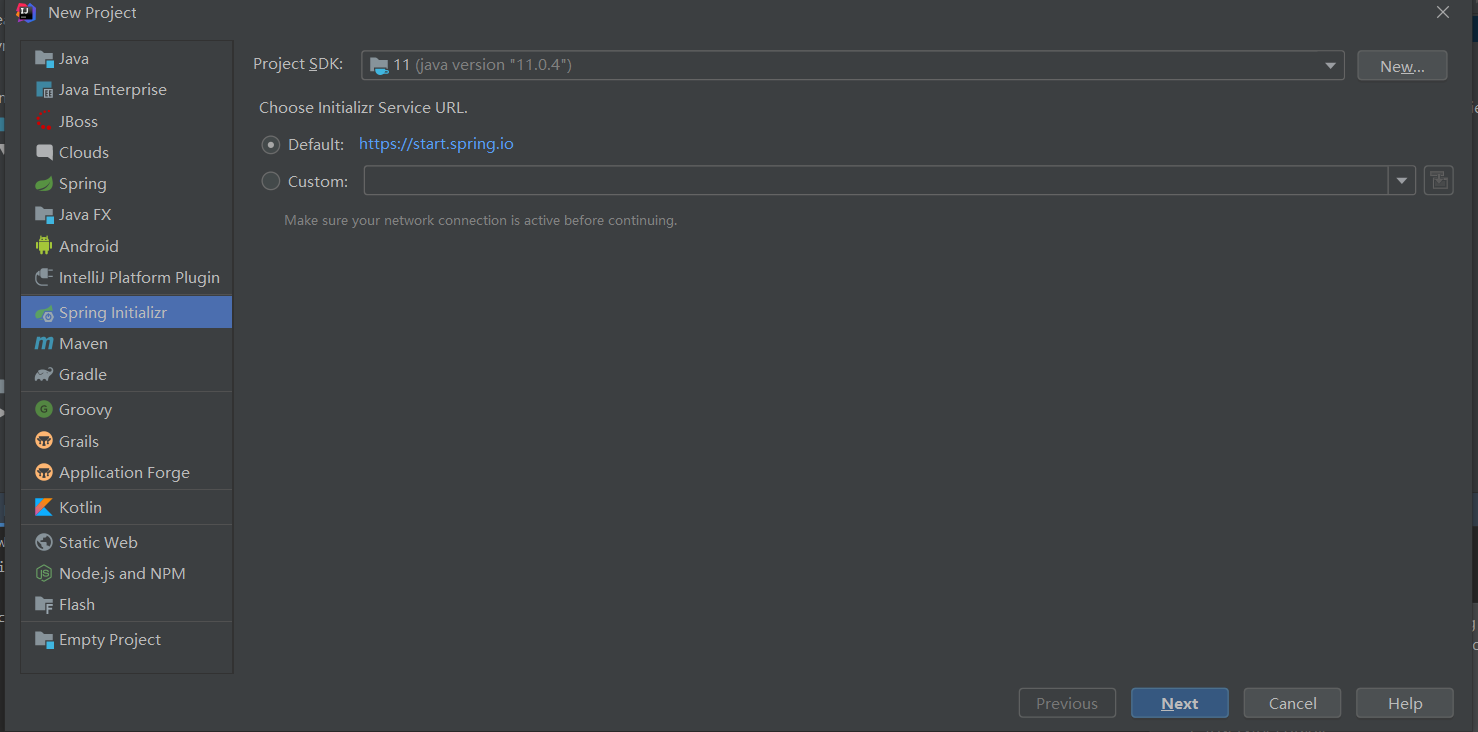
辅助工具postman，是一个简单的post请求、get请求的访问测试工具，用于调试接口。比浏览器访问的方式相比，浏览器只支持GET请求，而在postman中可以设置post请求需要的数据类型（JSON，JS等）以及上传。

Mysql的客户端（例如navicat），支持多种类型的数据库

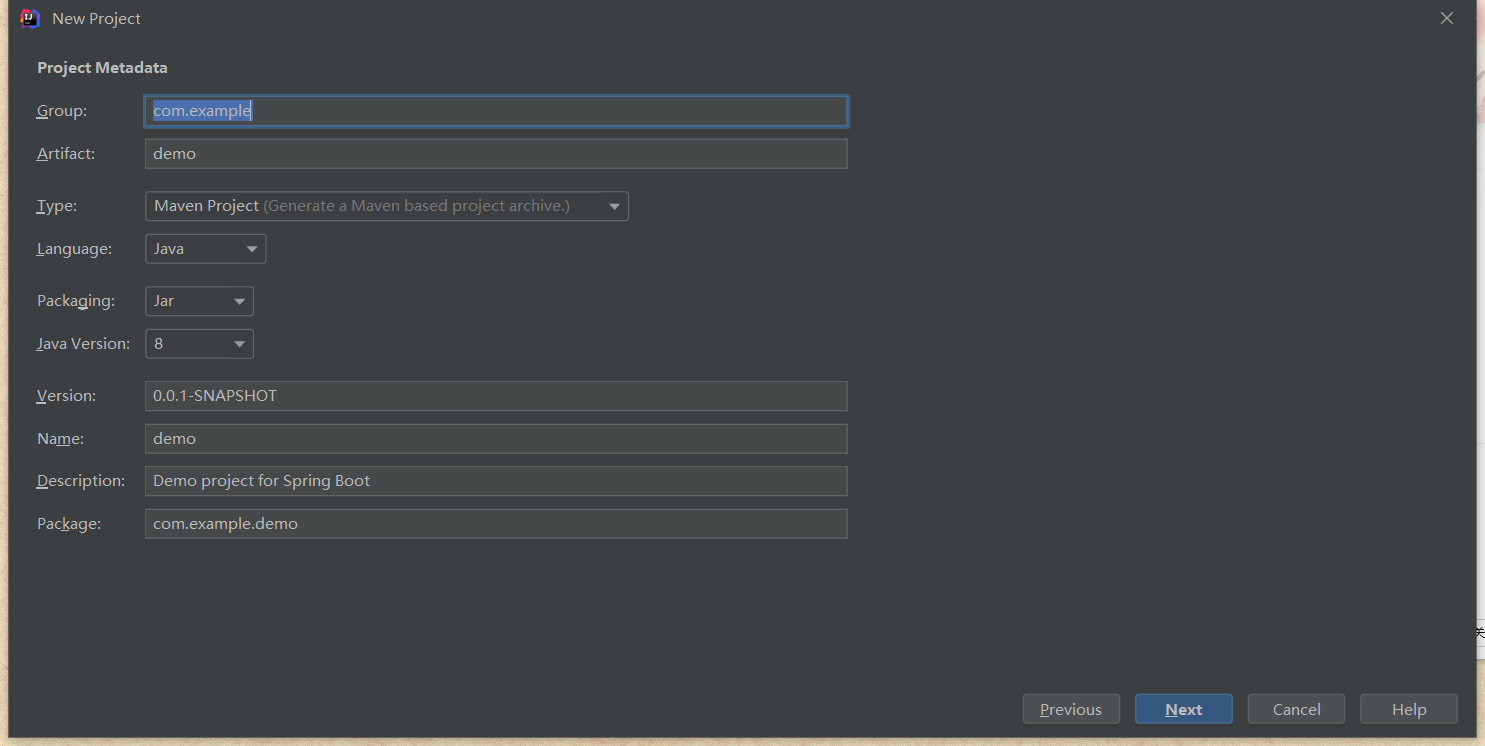
2．Springboot的环境搭建

a.打开IDEA，在初始化页面中点击Create New Project，IDEA会将它支持的所有类型的框架全部都展示出来，而本例子只需用到它提供的一个Spring Initializr这个初始化的工具。该初始化工具实际上是套用了Spring官网的一个初始化的脚手化工具。本例子是使用基于maven管理的开发

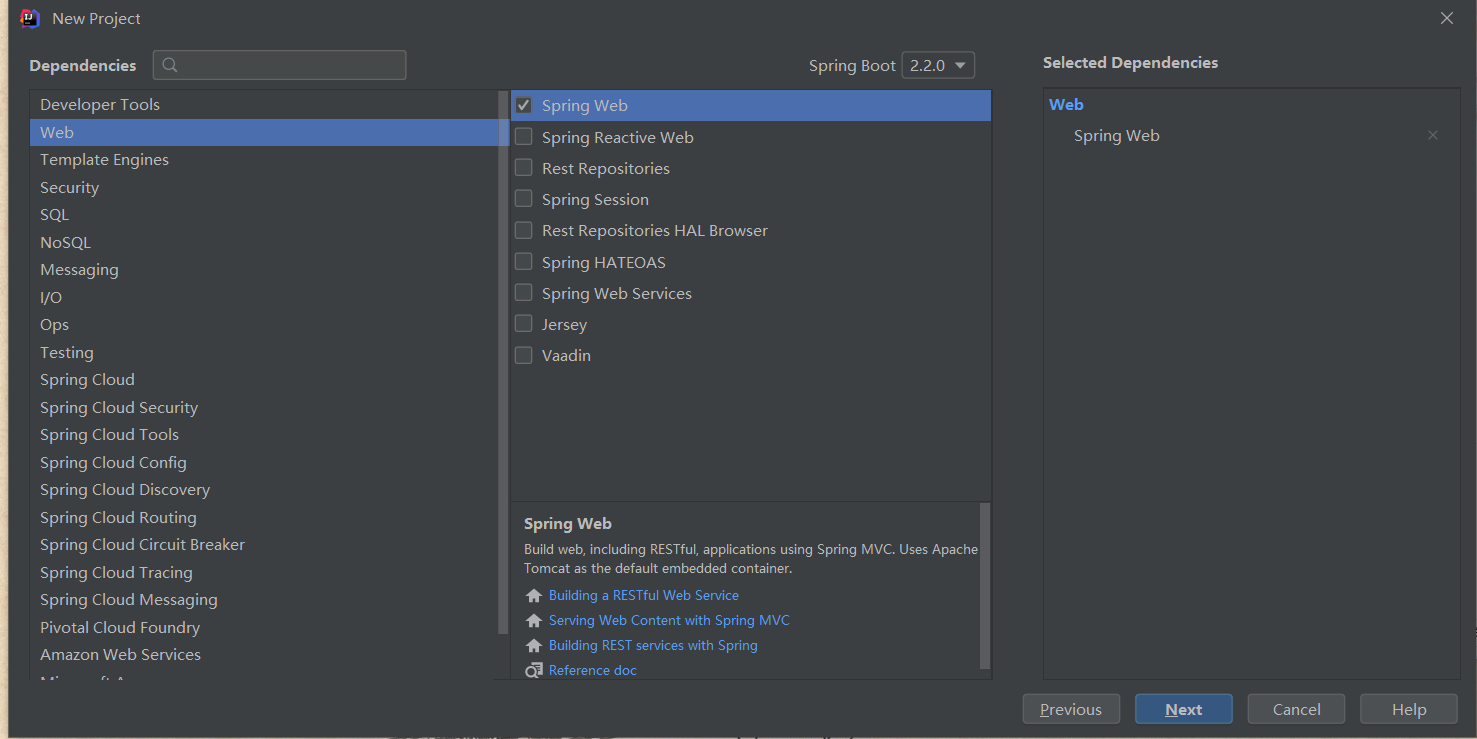
按照图中选中，默认点击next



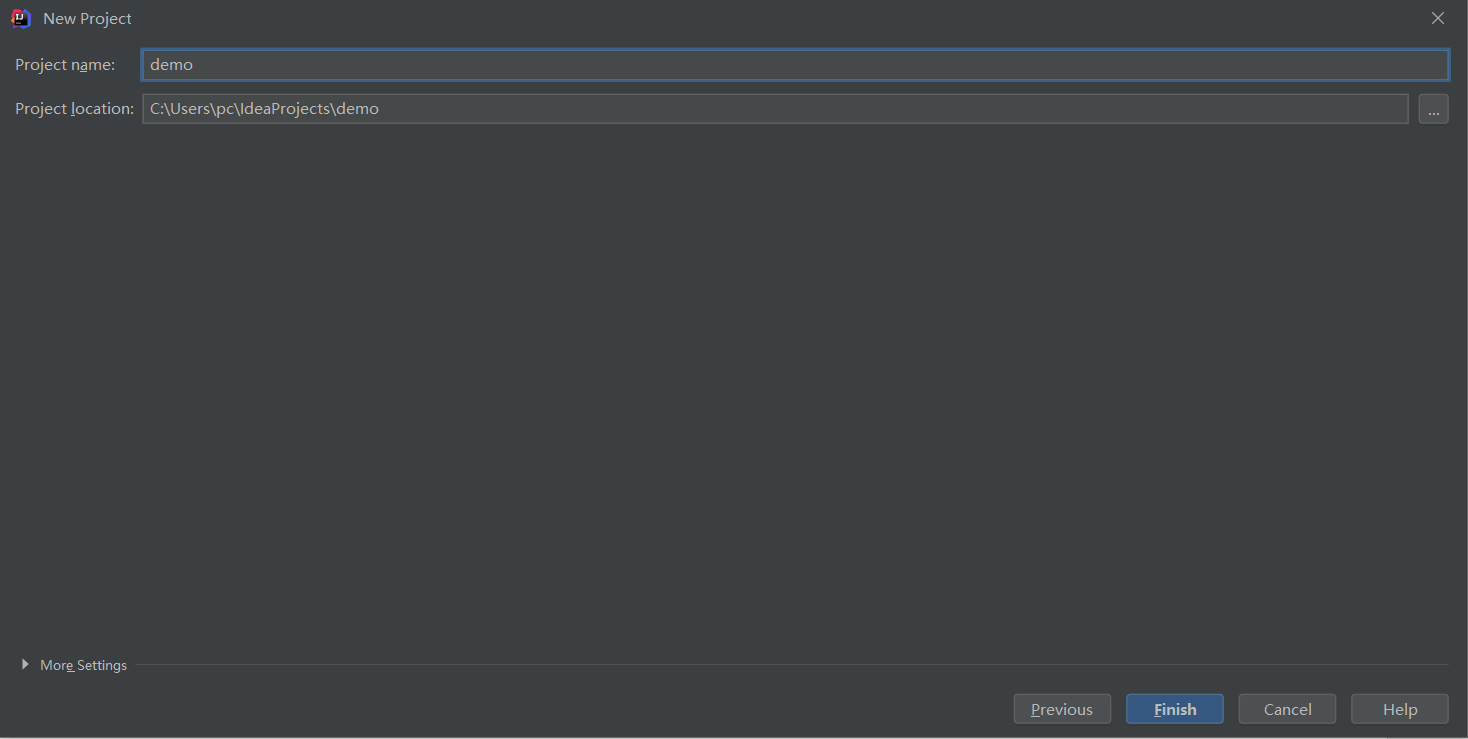
b.点击next后进入该页，选择默认，继续点击next



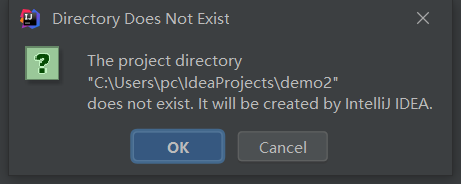
c.点击next后进入该页，可以选择开发需要的插件进行勾选，此时可根据需要，只勾选Web的Spring Web，然后继续点击next



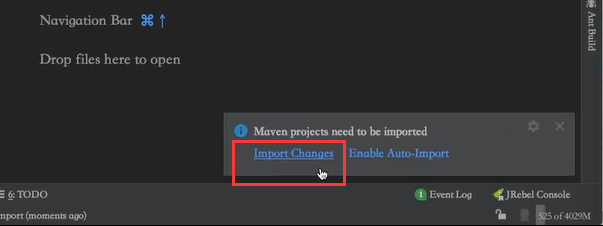
d.点击next后进入该页，可以选择修改名称和路径，完成后点击Finish

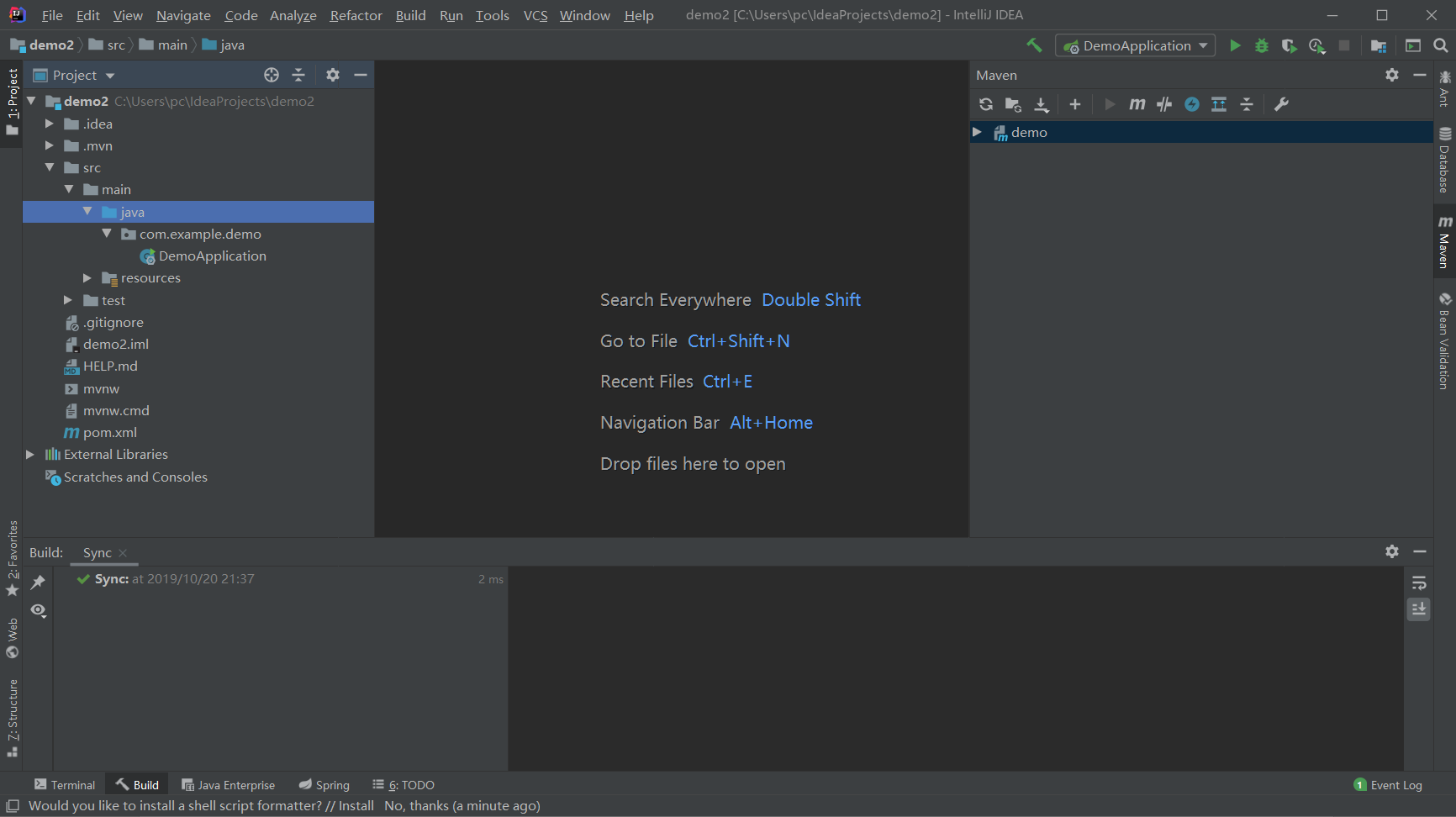


e.点击Finish后，会弹出该框提示该目录不存在，是否新建，选择OK即可

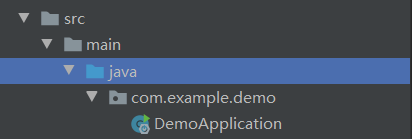


f.等待加载时间后，因为这是基于maven管理的开发，其中会有很多的依赖，过程中会在右下角提示需要加载，点击import Changes即可，然后就会加载出工程项目了，maven的依赖在pom.xml文件可以查看，页面如下

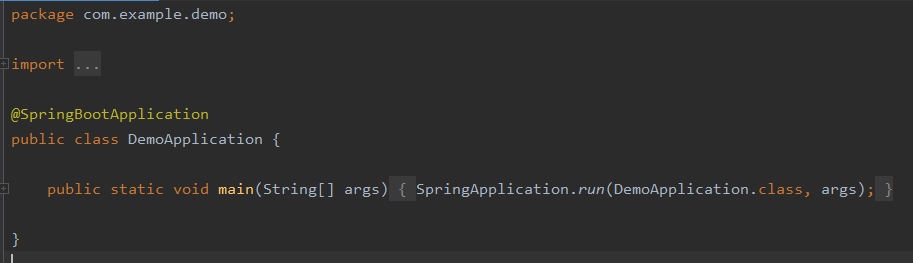




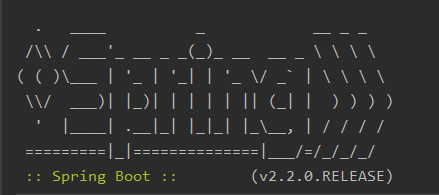
h.java文件都保存在src/main/java这个目录下面。这个DemoApplication.java文件就是工程项目的入口文件



里面有一个main方法是固定的，不需要去修改

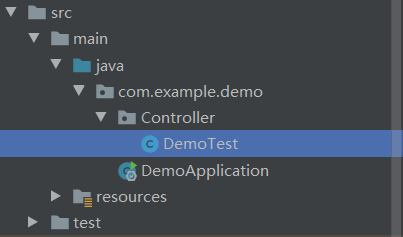


这样，一个Springboot的环境就算搭建好了，可以尝试运行一下试一下能否运行成功，若打印出这样一个图形，就表示运行成功，如若运行不成功，请根据报错提示更改。



2．Springboot的helloworld编写

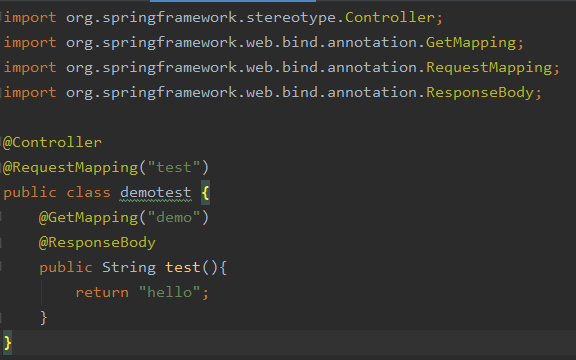
a.首先先在src/main/java目录下创建一个名叫Controller的包，并创建一个Controller的类，名叫DemoTest.java，创建完成后目录如下



b.接下来在该类中编写代码，先在class的外面写一个注解@Controller，标记这是一个Controller，然后再写一个路径注释@RequestMapping(path), 用于映射url到控制器类或一个特定的处理程序方法，可以指定GET、POST请求方式。@GetMapping也是路径注释，它是一个组合注解，是@RequestMapping(method = RequestMethod.GET)的缩写，等价于@RequestMapping的GET请求方式。

@ResponseBody 是作用在方法上的，表示该方法的返回结果直接写入 HTTP response body。

完成helloworld的代码编写，编写都是一些简单的java基础知识。代码如下，其中上面的import不需要手动输入，只需写好下面的注释敲回车后，上面的import会自动加载。



c.接下来可以在下方控制台中的terminal中进行测试代码。输入

curl localhost:8080/test/demo 即可测试，见图如下



其中因为默认端口是8080端口，所以要特别注意到端口冲突的问题。test是设置的路径，demo是我们设置的方法，根据代码，就可以很容易理解地址映射@RequestMapping(“test”) 以及 @GetMapping(“demo”) 这两句代码，可以尝试使用不同的名称进行测试学习。也可以尝试去掉@RequestMapping(“test”)中的test，这样在terminal中输入

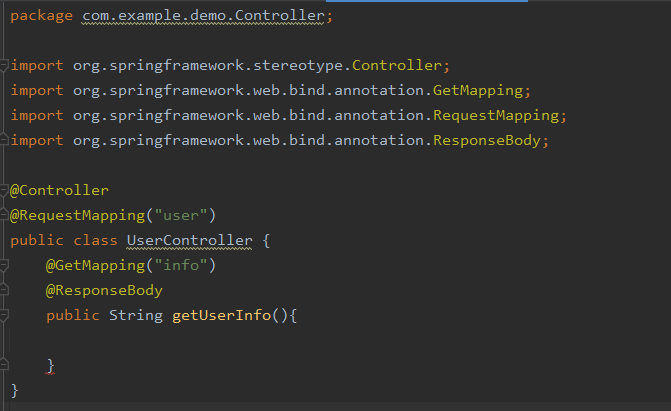
curl localhost:8080/test/demo 和curl localhost:8080/demo进行尝试一下会发生怎样的效果吧。路径管理是根据用户的业务实际需求进行判定是否需要写这个路径管理。

这样，helloworld例子就完成了。

3．Springboot数据库交互

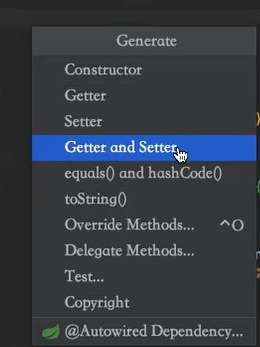
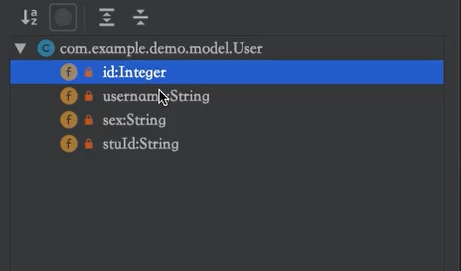
a.重新新建一个Controller，叫做UserController.java，同样的先写一个注解标记这是一个Controller。与helloworld样例相似。

此时代码如下

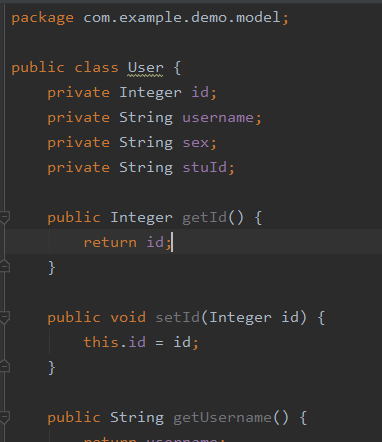


b.要准备一个model模块，在与Controller同级目录下创建一个model包，由于需要获取数据库中的信息，就需要写一个对象，命名为User，此时就可以在UserController中创建使用user了。同时需要给user对象定义字段。

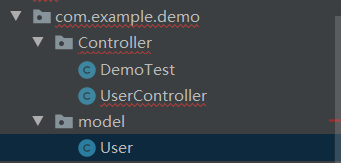
同时设置get和set方法，右键=>Getter and Setter=>选中需要设置的变量。

结果如下图所示：



此时目录如下

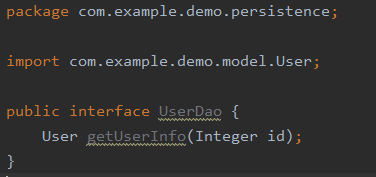


c.接下来需要使用一个插件mybatis进行与数据库进行连接，这是一个对jdbc进行封装过的插件，在使用时需要引入这个插件，我们将resources目录下的application.properties文件删掉，新建一个application.yml后缀的文件。然后内容复制上一定的通用代码。代码如下

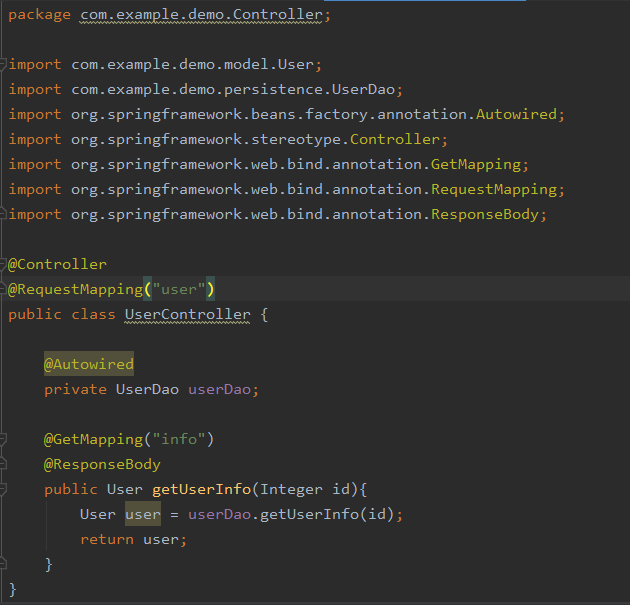
spring:  
 datasource:  
 url: jdbc:mysql://120.27.17.71:3306/hznu\_demo?useUnicode=true&characterEncoding=UTF-8&allowMultiQueries=true&autoReconnect=true&failOverReadOnly=false&useTimezone=True&serverTimezone=GMT%2B8  
 username: hznu\_stu  
 password: hznu123  
 driver-class-name: com.mysql.cj.jdbc.Driver  
  
mybatis:  
 type-aliases-package: com.example.demo.model  
 mapper-locations: classpath:mappers/\*Mapper.xml  
 configuration:  
 map-underscore-to-camel-case: true

同时我们需要在pom.xml文件中引入mysql的驱动，添加在依赖中，这时IDEA会提示是否加载，点击import即可。

d.我们再与controller同级目录下再新建一个包，叫做persistence。然后我们再创建一个方法，叫做UserDao，这个不是一个直接实现的方法，而是一个接口，在接口中定义了一个getUserInfo方法，代码如下



e.修改UserController代码，做UserDao的初始化和User对象创建时，根据提示导入对象即可。代码如下



f.因为现在接口是一个没有实现的方法，所以现在要对它进行一个实现，需要用到映射。我们在resources下创建一个Directory，命名为Mappers，然后创建一个UserMapper.xml文件，并贴上通用代码，代码如下

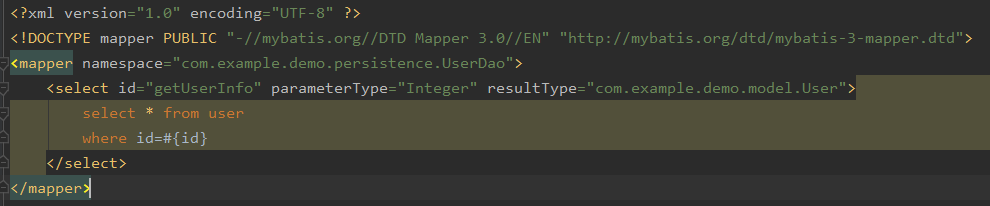
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="">

<sql id=”Base\_Colimn\_List”>

Id,articleId,appName,status,createdTime,updateTime

</sql>  
</mapper>

这里namespace命名空间要填写persistence的路径，并标记sql语句的id是getUserInfo的方法，然后就开始编写熟悉的sql语句了。我们还需要用一个模型装载返回的数据，我们在resultType属性中输入model中User对象的路径。完成后的代码如下



g.工程需要在使用时扫描mappers，所以我们需要在DemoApplication文件中头部写一段代码，同时根据提示加载mybatis依赖。

@MapperScan("com.example.demo.persistence")

最后还要新增一个依赖，这是一个mybatis和spring交互所要依赖的库，如下

<dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>1.1.1</version>  
</dependency>

这时我们运行，就可以运行成功了，在terminal中进行测试，输入

curl localhost:8080/user/info?id=107即可成功，当然如果数据源不同，请根据数据库进行不同的id值查询。

3、相关扩展：

①学习的网站推荐：

a.廖雪峰的官方网站：java和git基础知识建议去廖雪峰的官方网站上学习，上面分享了他的一些学习经验，对初学者非常友好。

b.极客时间：付费的课程，比如Springboot项目课程，上面的老师会详细地介绍原理。对初学者来讲，有个全面而完整的讲解，会让你有一个更高的起点。有需要的同学可以购买学习。

②扩展的辅助工具

a.ide插件：现在使用的东西都是ide自带的，可能以后会用到需要的插件，到时候自行下载。阿里巴巴代码规范插件，会提示出错的代码，减少项目的错误隐患。

b.postman：简化访问请求的操作，可以实现浏览器无法用post传递数据访问的缺点。

c.Nginx代理：在提交请求的时候，在不写端口的情况下，直接转发访问默认的端口号，实现接口的访问。

d.前端框架：jsp的学习，在正式工作中一般很少用到，jsp是java代码内嵌在html文件中的

e.缓存方法：减少对服务器的重复访问，延长服务器寿命。生成访问对象，把查询到对象数据放到缓存里，下一次请求相同的地址和参数时，会从缓存读取。

疑问的解答：

demo中的UserDao是什么？

答：生成的接口，里面有定义的方法，这个方法通过Mybatis框架做了映射的关联，映射到Mybatis的sql语句。访问数据库需要数据源（地址访问驱动），Mybatis框架的使用省略了繁琐的流程。