|  |  |
| --- | --- |
| 开始创建项目之间，强烈建议将IDEA内置Maven镜像更换为阿里云中央仓库  网上的例子：  1.找到自带maven目录中的settings.xml，我的是C:\Program Files (x86)\JetBrains\IntelliJ IDEA 2016.3.2\plugins\maven\lib\maven3\conf\settings.xml  2. 在setting.xml中<mirrors></mirrors>中添加内容：   |  | | --- | | 1. **<mirror>** 2. **<id>**alimaven**</id>** 3. **<name>**aliyun maven**</name>** 4. **<url>**http://maven.aliyun.com/nexus/content/groups/public/**</url>** 5. **<mirrorOf>**central**</mirrorOf>** 6. **</mirror>** |   最后保存即可，（将默认的删掉） |

1.第一个springboot应用：

|  |  |  |
| --- | --- | --- |
| HelloController.java：  C:\Users\zj\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2345截图20180316111727.png   |  | | --- | | 另举个例子：  @RequestMapping(value={"/",""},method={RequestMethod.POST,RequestMethod.GET},produces = "application/json; charset=UTF-8")  返回类型的是一个 Class， value 是请求的映射集合， method是请求格式的集合，produces 是返回数据格式。  ==========================  C:\Users\zj\Desktop\2345截图20180316161825.png  可以访问/hello,或者/hi 来打开同一个网页 |  |  | | --- | | @RestController  @RestController是@ResponseBody和@Controller的组合注解。  @Controller (最好用controller,不用restcontroller)  首先扫描Classpath获取注解了@Controller的对象(扫面通过语句：<context:component-sacn/>进行扫描)。 之后通过反射机制进行绑定。  @RequestMapping(")  可以被标注类型申明和方法申明这两种方式上。  申明在类型定义上，那么@RequestMapping（） 括号里面的东西通常为请求的地址  申明在方法定义上，那么它用来表明当面方法定义是一个Web请求处理方法。（Ps：这是书上讲的，个人理解的是加上具体的映射条件，使映射不会出错） |   **@RequestMapping 配置url映射**  @RequestMapping此注解即可以作用在控制器的某个方法上，也可以作用在此控制器类上。  当控制器在类级别上添加@RequestMapping注解时，这个注解会应用到控制器的所有处理器方法上。处理器方法上的@RequestMapping注解会对类级别上的@RequestMapping的声明进行补充。  以上代码sayHello所响应的url=localhost:8080/hello。  GirlApplication.java:  C:\Users\zj\Desktop\2345截图20180316135916.png  @SpringBootApplication//必须标注该注解，才能扫描程序里面的注解组件 |

2.项目属性配置：

|  |
| --- |
| C:\Users\zj\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2345截图20180316142220.png  项目配置文件 |

application.yml:

(项目配置文件)

|  |
| --- |
| **server:  port:** 8080 //设置端口号8080（及默认）  **servlet:  context-path:** /girl //（要加一个servlet,不然不行）设定应用的context-path. 即http://localhost:8081/girl/hello |

1. 如何将配置注入的类中：

法一：@Value

通过注解将配置文件中的内容，注到cupSize中：

|  |
| --- |
| C:\Users\zj\Desktop\2345截图20180316153942.png  此时配置文件中有： |

法二： 加以下两个注解

@Component

@ConfigurationProperties

|  |
| --- |
| 配置文件： |

=======================

通过修改active:后的dev或者prod,来切换不同配置

|  |
| --- |
|  |

3.mysql数据库操作

（1）

修改配置文件为以下内容：

|  |
| --- |
| **server:  port:** 8080  **spring:  datasource:  driver-class-name:** com.mysql.jdbc.Driver  **url:** jdbc:mysql://127.0.0.1:3306/dbgirl  **username:** root  **password:** 123456  **jpa:  hibernate:  ddl-auto:** create  **show-sql:** true |

**driver-class-name:** com.mysql.jdbc.Driver 驱动

dbgirl 数据库名称  
**username:** root 用户名  
**password:** 123456 密码

|  |
| --- |
| **jpa:  hibernate:  ddl-auto:** create  1、选择create运行时会自动帮我们创建表，每次运行都会创建空表。如果存在数据，会先删掉，再创建空。  2、update 一开始会创建新表，之后不会删除已有数据  3、create-drop 运行结束时，会删除表  4、none 默认什么都不做  5.validate 会检测类的属性和表结构是否一致，如果不一致，会报错 |

**show-sql:** true 可以在控制台显示sql语句

（2）

实现以下5个接口：

|  |
| --- |
| C:\Users\zj\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2345截图20180317110756.png |

=====================

rest风格接口：

新的rest用法

在较新的spring版本中，出了几个新的注解，简化了上面的用法，如下

**[java]** [view plain](http://blog.csdn.net/wangb_java/article/details/71271490) [copy](http://blog.csdn.net/wangb_java/article/details/71271490)

1. //@Controller+@ResponseBody组合，相当于在每个方法都加上@ResponseBody。
2. @RestController
3. **public** **class** HelloController {
4. //直接指定Post请求，同样也有@GetMapping
5. @PostMapping("/url")
6. //@RequestBody是指请求来的参数是json格式，以json格式来转换到uer，现在用json传参已经用的越来越多
7. **public** String hello(@RequestBody User user){
8. **return** "hello world";
9. }
10. }

前端ajax请求rest

JS请求AJAX我就不说的，不同的JS框架有不同的用法。如果要请求上面的rest接口，只需要在请求时指定url为http://localhost:8080/url，请求方法为post，传入相应的参数，上面的方法就会向AJAX请求返回一个字符串"hello world"，返回值 也可以是一个java复合类型，@ResponseBody注解会自动将其转换成json格式。

==================

@Repository用于标注数据访问组件，即DAO组件

如果 Web 应用程序采用了经典的三层分层结构的话，最好在持久层、业务层和控制层分别采用上述注解对分层中的类进行注释。

@Service用于标注业务层组件

@Controller用于标注控制层组件（如struts中的action）

@Repository用于标注数据访问组件，即DAO组件

@Component泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注。

第一个项目学习到此为止，初步了解javaweb后端的知识，但仍有很多不足