1. **Spring Boot环境搭建**
2. 访问<http://start.spring.io/>创建简单项目
3. 下载开发工具Spring Tools Suite
4. 将1中创建的项目使用2中下载的工具导入，maven导入
5. 启动简单测试web功能
6. **推荐使用application.yml，而不使用自带的application.properties配置文件**
7. **注解使用**

3.1、@Value使用

Application.yml写法

cupSize: B

Controller写法

@Value("${cupSize}")

**private** String cupSize;

3.2、@ConfigurationProperties使用

Application.yml写法

gril:

cupSize: D

age: 38

Controller写法

@Autowired

**private** GrilProperties grilProperties;

类GrilProperties.java



3.3、参数配置dev环境和生产环境

主配置文件

application.yml



dev配置文件

application-dev.yml



prod配置文件

application-prod.yml

****

3.4、启动项目时，使用带参数配置：

java -jar firstDemo-0.0.1-SNAPSHOT.jar --spring.profiles.active=dev

java -jar firstDemo-0.0.1-SNAPSHOT.jar --spring.profiles.active=prod

多环境配置还有更优雅的方式解决，日后再说

1. **Controller使用**

@Controller 🡪处理http请求

@RestController ->Spring4z之后新加的注解，原来返回json需要@ResponseBody配合 @Controller

@RequestMapping->配置URL参数

1. **了解thymeleaf使用，了解就行？为什么呢？**

因为现在的开发一般是前后端分离的。作为后端只需要提供一些rest接口，返回一些json格式，然后给前段就行了，不用再去使用模板的这种形式，使用模板的话会给性能想带来很大的损耗。所以这块仅仅了解即可。

1. **多映射使用**

@RequestMapping(value= {"/hello","/hi"},method=RequestMethod.***GET***)

1. **使用postman工具测试post方式发送请求到后台**

@RequestMapping(value= {"/hello","/hi"},method=RequestMethod.***POST***)

**8、@PathVariable->获取URL中的数据**

@RequestMapping(value={"/hello/{id}","/hi"},method=RequestMethod.***GET***)

**public** String index(@PathVariable("id") Integer id) {…}

**9、@RequestParam->获取请求参数的值**

不传?id=100参数报错

@RequestMapping(value= {"/hello","/hi"},method=RequestMethod.***GET***)

**public** String index(@RequestParam("id") Integer myId) {…}

不传?id=100参数不报错,采用默认值

@RequestMapping(value= {"/hello","/hi"},method=RequestMethod.***GET***)

**Public** String index(

@RequestParam(value="id",required=**false**,defaultValue="0") Integer myId) {

… 非必填 默认值为0

}

**10、@GetMapping->组合注解**

@GetMapping(value="/hello")

**public** String index(

@RequestParam(value="id",required=**false**,defaultValue="0") Integer myId) {… 类似的有@PostMapping, @PutMapping,组合注解，节省时间

}

**11、幽默趣味语句**

从删库到跑路，从入门到放弃。

**12、Spring-DATA-JPA**

JPA定义了一系列对象持久化的标准，目前实现这一规范的产品有Hibernate，TopLink 等。



12.1、引入包

<!-- jpa -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- oracle相关包 -->

<dependency>

<groupId>ojdbc</groupId>

<artifactId>ojdbc6</artifactId>

<version>11.2.0.4</version>

</dependency>

</dependencies>

12.2、application.yml中配置(在总配置中配置，那么dev和prod都是能用到的)

spring:

profiles:

active: dev

datasource:

driver-class-name: oracle.jdbc.driver.OracleDriver

url: jdbc:oracle:thin:@localhost:1521:orcl

username: c##zhouwq

password: oracle

jpa:

hibernate:

ddl-auto: create 删除表，删除序列，创建表，创建序列

update 不删除表

create-drop 应用停止时，会把表删掉

none 默认什么都不做

validate 验证类里面的属性跟表结构是否一致。不一 致就报错

show-sql: **true**

3、通过RESTful API实现一个女生的增删改查，见代码

Github:

**13、事务管理，service方法上加上注解**@Transactional

**import** org.springframework.transaction.annotation.Transactional;

@Transactional

@Override

**public** **void** addTwoGrils() {…}

总结：

配置文件application.yml使用，程序怎样使用@Value注解从配置文件取值。

不同环境不同配置，及其启动项目时带参数，可以很容易的打不同环境的包。

请求路径映射和参数怎样接受，以及**thymeleaf简单使用。**

Spring-data-JAP实现增删改查。

事务管理，只需在service层方法上加上@Transactional。