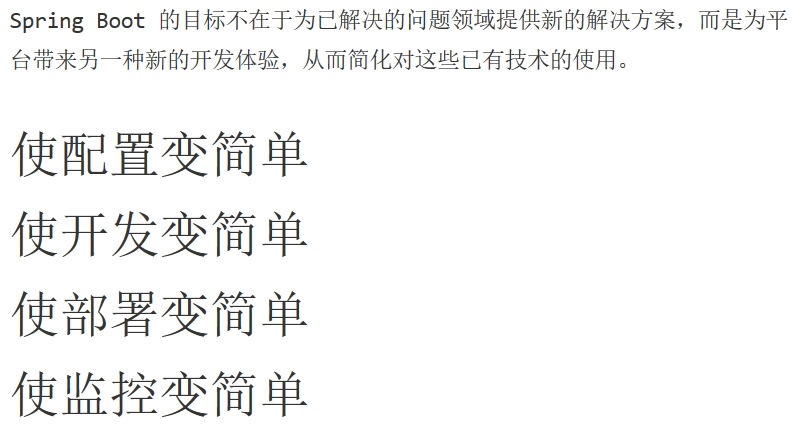
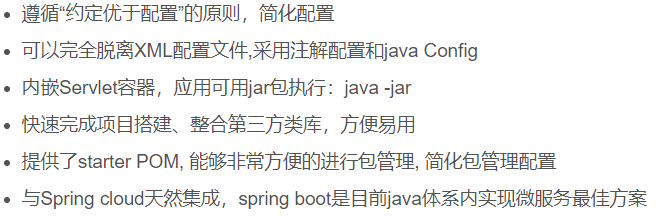
<https://www.kancloud.cn/hanxt/springboot2/1177580>

1. Springboot诞生的目的

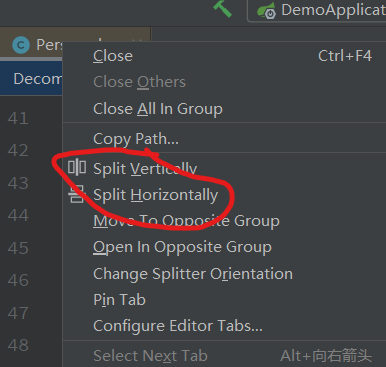


2.主要特性



3.提高开发效率的常用插件

在右侧maven->lifecycle->双击compile 可以完成编译

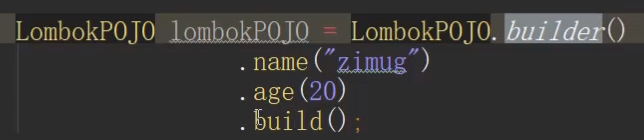
 ->可以实现分屏

lombok：通过注解可以自动生成getset、构造器等方法以及打日志

@Data =>生成getter setter方法,无参构造,toString方法,hashcode方法..

@Sl4j => 代替打日志代码

@Builder => 实现链式构造对象



@AllArgsConstructor

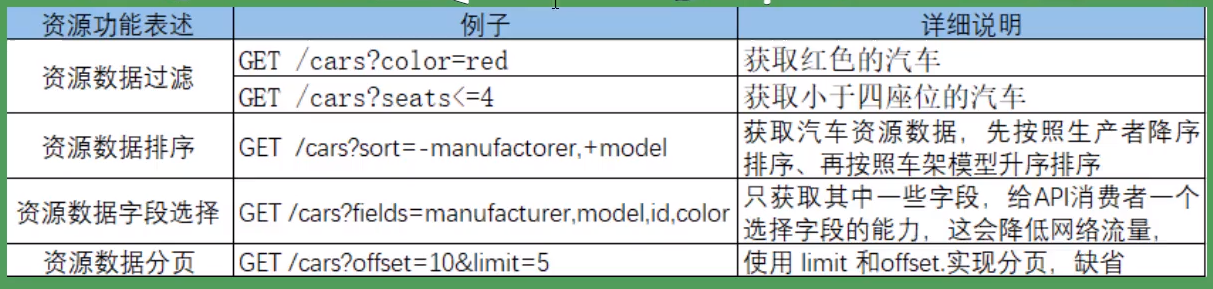
全参构造函数

Gsonformat：快速的将json转变为实体类

Maven Helper：高效管理配置文件

4.RESTful风格

URI体现资源，而对资源的操作用http方法去体现（get、post、put、delete）



版本化你的api

api/v1/person

http状态码

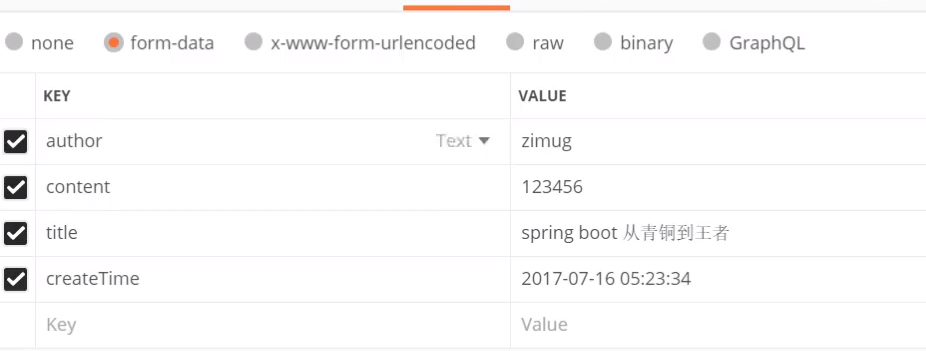
200 ok ； 400 错误请求 ； 500 服务端错误；

5.常用注解

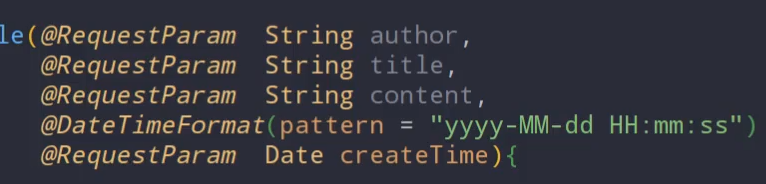
@RequestMapping

@RestController = 在类上@Controller+在方法上@ResponseBody

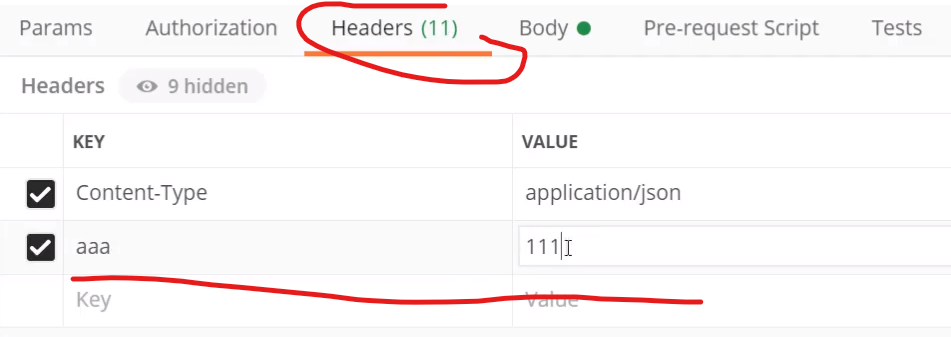
@PathVariable表示URI路径上的变量



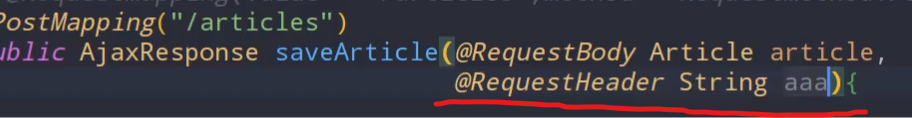
模拟前端表单提交对应后端接口就是



模拟前端请求头传参



对应后端接口注解





6.如何将yml中的配置注入到对象中

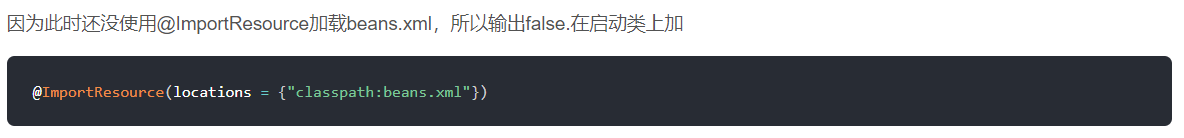
在对应的java类上注解@Component

在对应简单属性上注解@Value并声明yml中对应的配置名称

在父级类上注解@ConfigurationProperties可以实现复杂类型的装配

@ConfigurationProperties配合@Validated并将校验规则加到对应属性上可以实现对注入属性值的校验（支持嵌套校验）

7.如何引入旧项目的xml文件



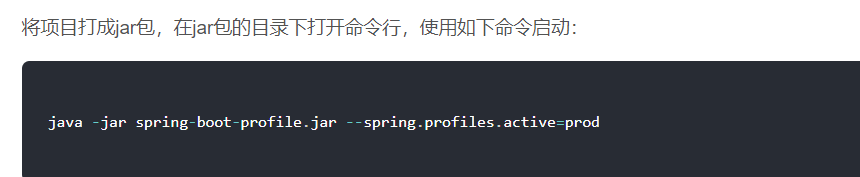
8.如何配置不同的生产环境（开发、测试、生产环境）

首先要配置多个yml文件对应不同环境

然后在application.yml中配置使用响应环境的yml



或者在项目已经打包之后，使用命令行传参的方式

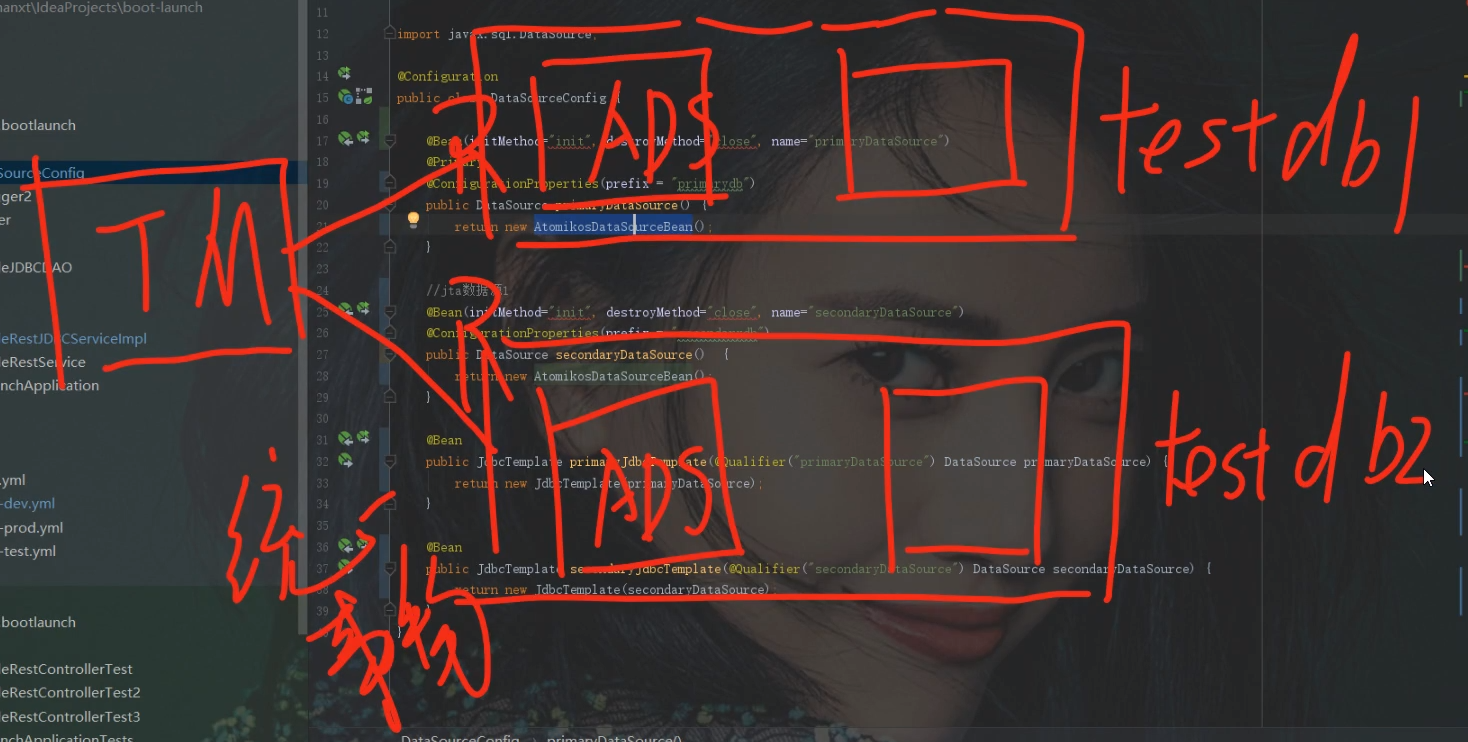


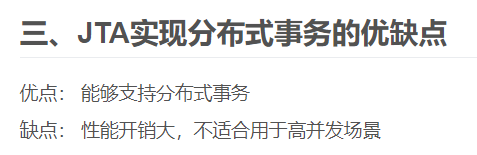
9.JTA实现分布式事务

一般单数据库所使用的数据源是javax.sql.DataSource，此数据源无法保证分布式事务的准确性。

多数据库所使用的数据源是其子类AtomikosDataSourceBean

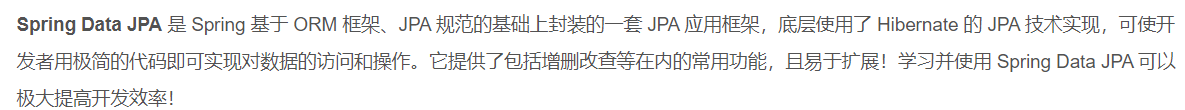
为了实现分布式事务还需要一个事务管理器

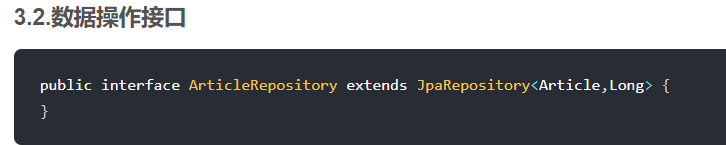




10.ORM主流框架选型

Spring Data JPA和Mybatis





11.实现对象之间相同属性的快速赋值

Spring自带的BeanUtil，缺点：不能对list等集合进行拷贝

Dozer包中的工具类DozerBeanMapper，自己写一个工具类就可以实现list转list

12.JPA多数据源的实现

yml里配置两套DataSource

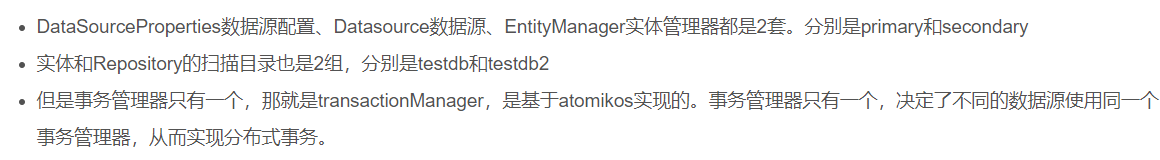
分两个包对应两套数据源，建立对应的数据操作接口，



创建对应数据源的bean对象，并注入到上下文中

配置实体扫描以及事务管理

13.JPA+atomikos实现分布式事务



14.JPA、JTA

JPA—java persistent API 持久层API

JTA—java transactional API 事务API

15.Mybatis实现多数据源

16.Mybatis整合atomikos实现分布式事务

驱动类是MysqlXADataSource（支持分布式事务）