极客时间

课程答疑

答疑内容清单

- 开发环境相关的说明
- 一些 Spring 常用注解简介
- 关于 Actuator Endpoints 访问不到的说明
- 多数据源、分库分表、读写分离的关系
- 与内部方法调用与事务的课后问题
- REQUIRES_NEW 与 NESTED 事务传播特性的说明
- Alibaba Druid 的一些展开说明

关于开发环境的说明

开发环境

- Java 8 / Java 11 （由于会用到Lambda，必须是Java 8+）
- IntelliJ IDEA 社区版（安装 Lombok 插件）
- Apache Maven（如果不用命令行编译，也可以用IDE自带的）
- Mac OS Mojave （大家不用纠结操作系统，没什么差别）
- Docker（用于在本地启动一些演示用的依赖设施）

一些 Spring 常用注解简介

一些常用注解

Java Config 相关注解

- @Configuration
- @ImportResource
- @ComponentScan
- @Bean
- @ConfigurationProperties

一些常用注解

定义相关注解

- @Component / @Repository / @Service
- @Controller / @RestController
- @RequestMapping

注入相关注解

- @Autowired / @Qualifier / @Resource
- @Value

关于 Actuator Endpoints 访问不到的说明

Actuator 提供的一些好用的 Endpoint

URL	作用
/actuator/health	健康检查
/actuator/beans	查看容器中的所有 Bean
/actuator/mappings	查看 Web 的 URL 映射
/actuator/env	查看环境信息

如何解禁 Endpoint

默认

- /actuator/health 和 /actuator/info 可 Web 访问

解禁所有 Endpoint

- application.properties / application.yml
  - management.endpoints.web.exposure.include=*

生产环境需谨慎

“Talk is cheap, show me the code.”

Chapter 2 / datasource-demo

多数据源、分库分表、读写分离的关系

几种常见情况

- 系统需要访问几个完全不同的数据库

不同的datasource

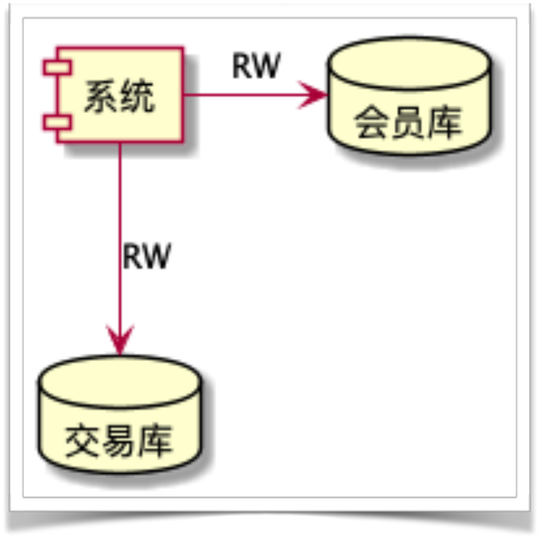
- 系统需要访问同一个库的主库与备库

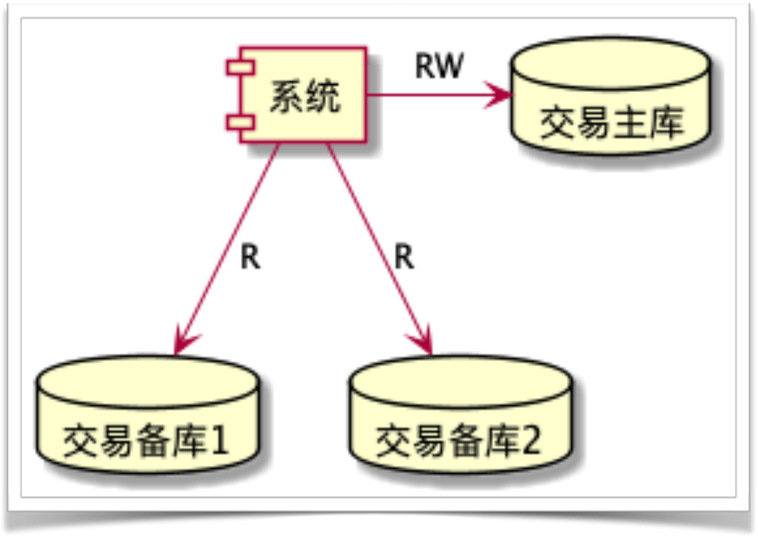
处于性能考虑

- 系统需要访问一组做了分库分表的数据库

垂直拆分

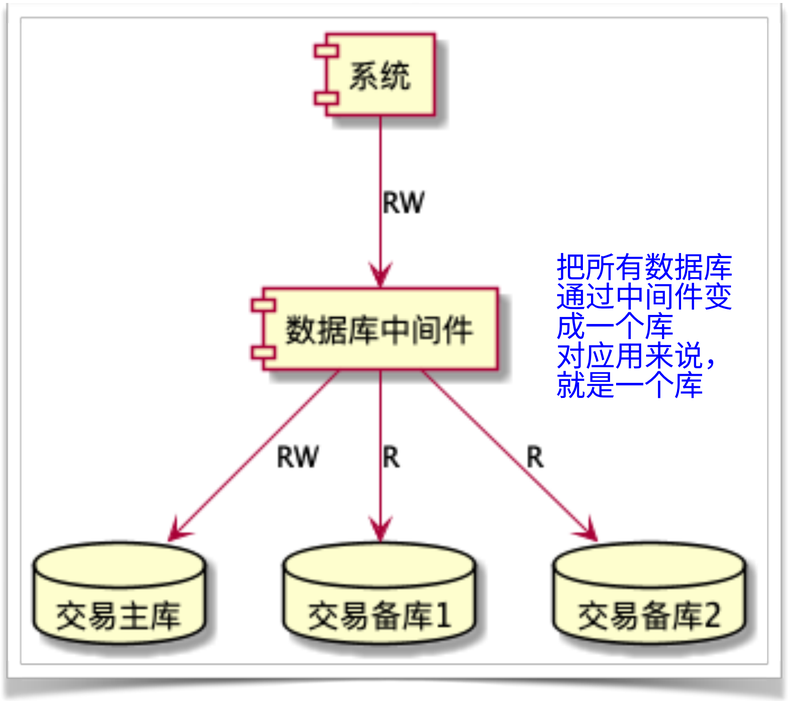
水平拆分

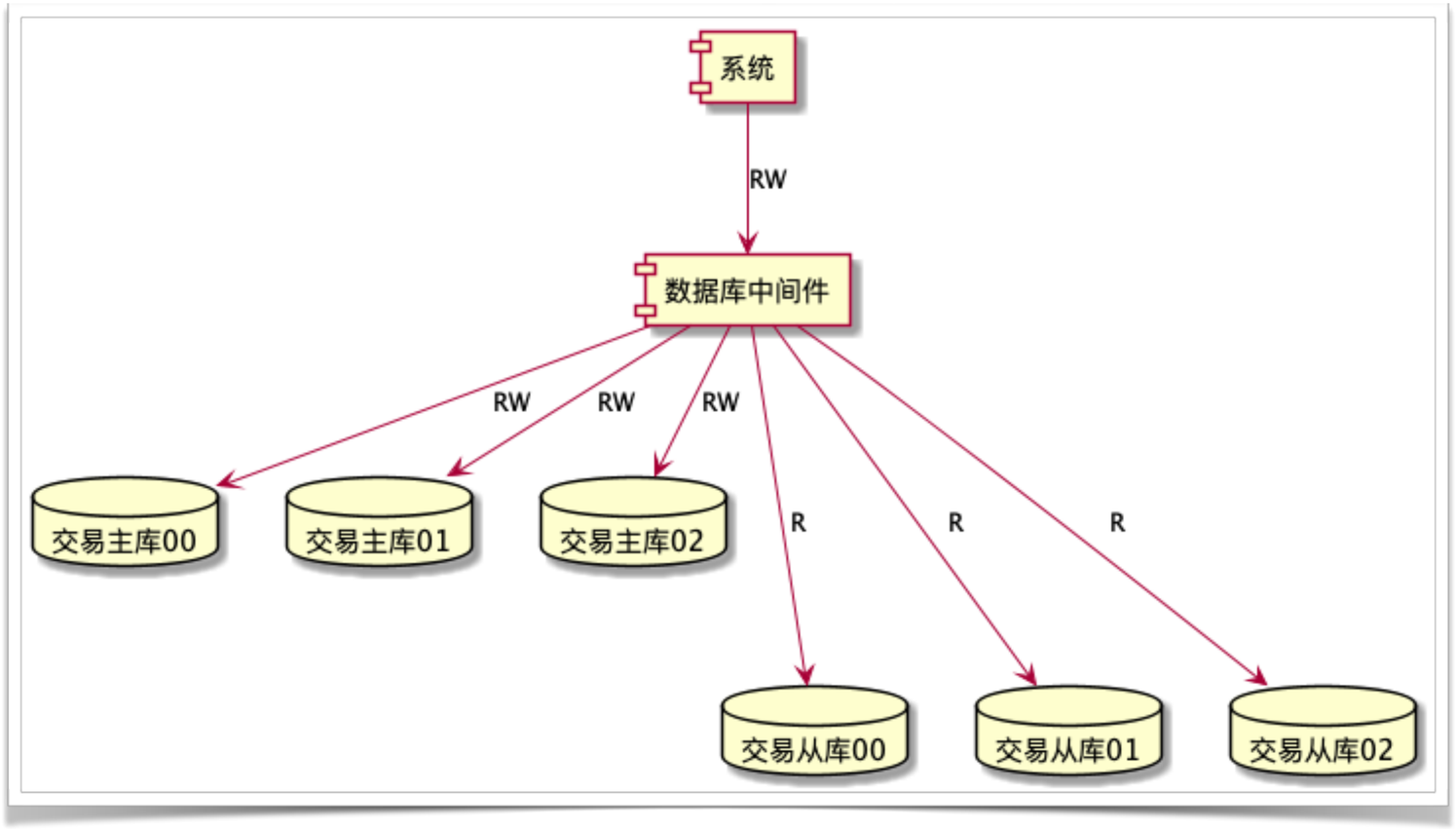




使用数据库中间件的情况

若有分库分表，使用中间件简化操作；
若只是实现读写分离，可以在Client端实现读写分离，也可以使用中间件实现读写分离。





内部方法调用与事务的课后问题

事务的本质

- **Spring 的声明式事务本质上是通过 AOP 来增强了类的功能**
- **Spring 的 AOP 本质上就是为类做了一个代理**
  - 看似在调用自己写的类，实际用的是增强后的代理类

如何增强呢？

事务之前，代理类开启事务；
事务执行；
事务之后，代理类进行提交或者回滚。

- **问题的解法**
  - 访问增强后的代理类的方法，而非直接访问自身的方法

“Talk is cheap, show me the code.”

Chapter 2 / declarative-transaction-demo

REQUIRES_NEW 与 NESTED 事务传播特性的说明

REQUIRES_NEW v.s. NESTED

REQUIRES_NEW，始终启动一个新事务

- 两个事务没有关联

NESTED，在原事务内启动一个内嵌事务

- 两个事务有关联
- 外部事务回滚，内嵌事务也会回滚

* 内嵌事务回滚并不会影响外部事务的回滚

“Talk is cheap, show me the code.”

FAQ 2019-02 / declarative-transaction-demo

transaction-propagation-demo

Alibaba Druid 的一些展开说明

慢 SQL 日志

系统属性配置

- druid.stat.logSlowSql=true
- druid.stat.slowSqlMillis=3000 SQL执行超过3s,则输出

Spring Boot

- spring.datasource.druid.filter.stat.enabled=true
- spring.datasource.druid.filter.stat.log-slow-sql=true
- spring.datasource.druid.filter.stat.slow-sql-millis=3000

一些注意事项

- 没特殊情况，不要在生产环境打开监控的 Servlet　以日志输出为主
- 没有连接泄露可能的情况下，不要开启 removeAbandoned
- testXxx 的使用需要注意　取放的时候会有连接校验，开销挺大的

  如：

  testOnBorrow()

  testOnReturn()
- 务必配置合理的超时时间

  如，一旦数据库有问题，建立连接的时候，可能一直卡死在那边，所以建立连接的地方，配置一些像ConnectionTimeOut这样一些超时的设置

  每条SQL语句的执行，也最好设置超时的设置，避免长时间卡死