

32

4

81

手记 / 后端开发

自定义注解完成数据库切库

2018.01.06 15:23 6666浏览

前提

这几天，学员们反馈希望学习一下自定义注解，正好准备高并发课程内容里有一块涉及到使用自定义注解完成数据库切库的内容。这里单独写一篇文章记录说明一下。

为什么会有数据库切库一说

首先，许多项目都有主库与从库，有的主库后面甚至会有很多个从库，主从库之间的通常同步也很快，这为数据库切库提供了一个基础，因为可以去不同的数据库查询，得到相同的结果（如果不同的数据库是完全不同的，这个不在我们这篇文章讨论的范围之内，那个属于让项目支持多个数据源）

其次，随着项目越来越大、操作的用户越来越多，对数据库的请求操作越来越多，很容易想到的是将读写请求分开，将写请求交给主库处理，读请求直接从某个从库读取。这样可以极大的减少大量对主库的请求，提升主库的性能。

接下来具体说一下如何通过自定义注解完成切库（代码使用springboot实现）：

第一步、定义我们自己的切库注解类

自定义注解有几点需要注意：

- 1) @Target 是作用的目标，接口、方法、类、字段、包等等，具体看：ElementType
- 2) @Retention 是注解存在的范围，RUNTIME代表的是注解会在class字节码文件中存在，在运行时可以通过反射获取到，具体看：RetentionPolicy
- 3) 允许的变量，通常都要给定默认值，比如我们使用一个service时，可以@Service，也可以@Service("xxxx")

```
@Retention(RetentionPolicy.RUNTIME)
@Target({
    ElementType.METHOD
})
public @interface RoutingDataSource {

    String value() default DataSources.MASTER_DB;
}
```

第二步、定义需要使用的数据库及配置

- 1、数据库配置：application.properties，这里要注意不同db的前缀区别

实战Java并发编程入门与高并发... 登录 / ¥ 299 · 初级 · 82977

实战Java开发企业级权限管理系统 ¥ 388 · 中级 · 81383

实战学习算法思想，修炼编程内功 ¥ 166 · 中级 · 87729

Java入门第一季 入门 · 8951354

Java入门第二季 入门 · 8453019

就业班SSM到Spring Boot入门与... ¥ 966 · 4步骤/22门课 · 829

Redis在股票分时K线图计算的

数据库设计技巧：一个字符串:

```
## datasource master #
spring.datasource.type=com.alibaba.druid.pool.DruidDataSource
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/master?characterEncoding=utf8
spring.datasource.username=root
spring.datasource.password=466420182
```

```
## datasource slave #
spring.datasourceSlave.type=com.alibaba.druid.pool.DruidDataSource
spring.datasourceSlave.driver-class-name=com.mysql.jdbc.Driver
spring.datasourceSlave.url=jdbc:mysql://localhost:3306/slave?characterEncoding=utf8
spring.datasourceSlave.username=root
spring.datasourceSlave.password=466420182
```

2、定义支持的数据源id:

```
public interface DataSources {

    String MASTER_DB = "masterDB";

    String SLAVE_DB = "slaveDB";

}
```

3、定义数据库实体类并配置为多数据源的形式

这里不要忽略了通过 MapperScan 指定需要扫描的mybatis的接口类

```
@Configuration
public class DataSourceConfig {

    //destroy-method="close"的作用是当数据库连接不使用的時候,就把该连接关闭
    @Bean(destroyMethod = "close", name = DataSources.MASTER_DB)
    @ConfigurationProperties(prefix = "spring.datasource")
    public DataSource dataSource() {
        return DataSourceBuilder.create().type(DruidDataSource.class).build();
    }

    @Bean(destroyMethod = "close", name = DataSources.SLAVE_DB)
    @ConfigurationProperties(prefix = "spring.datasourceSlave")
    public DataSource dataSourceSlave() {
        return DataSourceBuilder.create().type(DruidDataSource.class).build();
    }

}
```

4、配置成动态数据源:

```
@Configuration
@MapperScan(basePackages = {"com.xxx.dao"}) // 这里需要替换为实际的包名
public class MybatisConfig {

    @Autowired
    @Qualifier(DataSources.MASTER_DB)
    private DataSource masterDB;

    @Autowired
    @Qualifier(DataSources.SLAVE_DB)
    private DataSource slaveDB;

    /**
     * 动态数据源
     */
    @Bean(name = "dynamicDataSource")
    public DataSource dynamicDataSource() {
        DynamicDataSource dynamicDataSource = new DynamicDataSource();
        // 默认数据源
        dynamicDataSource.setDefaultTargetDataSource(masterDB);

        // 配置多数据源
        Map<Object, Object> dsMap = Maps.newHashMap();
        dsMap.put(DataSources.MASTER_DB, masterDB);
        dsMap.put(DataSources.SLAVE_DB, slaveDB);
        dynamicDataSource.setTargetDataSources(dsMap);
    }
}
```

相关课程



Java并发编程入门与高并发...
¥ 299 · 初级 · 82977



Java开发企业级权限管理系统
¥ 388 · 中级 · 81383



学习算法思想, 修炼编程内功
¥ 166 · 中级 · 87729



Java入门第一季
入门 · 8951354



Java入门第二季
入门 · 8453019



SSM到Spring Boot入门与...
¥ 966 · 4步骤/22门课 · 829

×



```

        return dynamicDataSource;
    }

    @Bean
    @ConfigurationProperties(prefix = "mybatis")
    public SqlSessionFactoryBean sqlSessionFactoryBean() {
        SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();
        // 配置数据源, 此处配置为关键配置, 如果没有将 dynamicDataSource
        sqlSessionFactoryBean.setDataSource(dynamicDataSource());
        return sqlSessionFactoryBean;
    }
}

```

第三步、使用ThreadLocal安全的管理当前进程使用的数据源连接

```

@Slf4j
public class DataSourceContextHolder {

    /**
     * 默认数据源
     */
    public static final String DEFAULT_DATASOURCE = DataSources.DEFAULT_DATASOURCE;

    private static final ThreadLocal<String> contextHolder = new ThreadLocal<>();

    // 设置数据源名
    public static void setDB(String dbType) {
        log.debug("切换到{}数据源", dbType);
        contextHolder.set(dbType);
    }

    // 获取数据源名
    public static String getDB() {
        return (contextHolder.get());
    }

    // 清除数据源名
    public static void clearDB() {
        contextHolder.remove();
    }
}

```

第四步、通过编写切面, 对所有我们自定义切库注解的方法进行拦截, 动态的选择数据源

这里是为下一步提供铺垫, 动态调整DataSourceContextHolder里存储的值, 使用threadLocal来管理是为了避免多线程之间互相影响。

自定义注解, 核心的处理就是写处理这个注解的逻辑, 然后通过指定的拦截方案根据当前的数据做一些动态的处理。比如Spring提供的@Controller、@Service等注解, 都是需要我们在配置文件里配置好需要扫描的路径, 然后项目启动时, spring根据配置去指定路径读取这些配置, 然后这些类才可以被spring进行管理。

这里不要忽略了默认数据源要选择主库, 如果切库出现什么问题, 比如配置错误等, 可以保证访问主库来得到正确的结果; 另外, 请求完了不要忘记调用提供的clearDB的操作, 防止threadLocal误用带来的内存泄露。

```

@Aspect
@Component
@Slf4j
public class DynamicDataSourceAspect {

    @Before("@annotation(RoutingDataSource)")
    public void beforeSwitchDS(JoinPoint point){

        // 获得当前访问的class
    }
}

```

相关课程



Java并发编程入门与高并发...
¥ 299 · 初级 · 82977



Java开发企业级权限管理系统
¥ 388 · 中级 · 81383



学习算法思想, 修炼编程内功
¥ 166 · 中级 · 87729



Java入门第一季
入门 · 8951354



Java入门第二季
入门 · 8453019



SSM到Spring Boot入门与...
¥ 966 · 4步骤/22门课 · 829

✕





32



4



81



```
Class<?> className = point.getTarget().getClass();

// 获得访问的方法名
String methodName = point.getSignature().getName();
// 得到方法的参数的类型
Class[] argClass = ((MethodSignature)point.getSignature
String dataSource = DataSourceContextHolder.DEFAULT_DAT
try {
    // 得到访问的方法对象
    Method method = className.getMethod(methodName, arg

    // 判断是否存在@DS注解
    if (method.isAnnotationPresent(RoutingDataSource.cl
        RoutingDataSource annotation = method.getAnnota
        // 取出注解中的数据源名
        dataSource = annotation.value();
    }
} catch (Exception e) {
    log.error("routing datasource exception, " + method
}
// 切换数据源
DataSourceContextHolder.setDB(dataSource);
}

@After("@annotation(RoutingDataSource)")
public void afterSwitchDS(JoinPoint point){
    DataSourceContextHolder.clearDB();
}
}
```

第五步、动态的取出我们在切面里设置的数据源的字符串即可

这里需要把原理介绍一下，在连接数据库时其实是先选择一个配置好的spring管理的datasource的id，就是我们之前在 DataSourceConfig 类里定义的DataSource 实体类的id：masterDB 和 slaveDB。然后根据id去spring的上下文选择配置，进行数据库连接。有兴趣的可以看一下源码。

```
@Slf4j
public class DynamicDataSource extends AbstractRoutingDataSource

    @Override
    protected Object determineCurrentLookupKey() {
        log.debug("数据源为{}", DataSourceContextHolder.getDB());
        return DataSourceContextHolder.getDB();
    }
}
```

第六步、取消自动配置数据源，使用我们这里定义的数据源配置

在SpringBoot启动类上通常直接使用@SpringBootApplication就可以了，这里需要调整为：

```
@SpringBootApplication(exclude = {
    DataSourceAutoConfiguration.class
})
```

使用

如何使用呢，我们简单演示一下：

```
@Service
public class DataSourceRoutingService {

    @Resource
```

相关课程



Java并发编程入门与高并发...
¥ 299 · 初级 · 82977



Java开发企业级权限管理系统
¥ 388 · 中级 · 81383



学习算法思想，修炼编程内功
¥ 166 · 中级 · 87729



Java入门第一季
入门 · 8951354



Java入门第二季
入门 · 8453019



SSM到Spring Boot入门与...
¥ 966 · 4步骤/22门课 · 829



```
private SysUserMapper sysUserMapper;

@RoutingDataSource(DataSources.MASTER_DB) // 这个注解这时是可以
public SysUser test1(int id) {
    return sysUserMapper.selectByPrimaryKey(id);
}

@RoutingDataSource(DataSources.SLAVE_DB)
public SysUser test2(int id) {
    return sysUserMapper.selectByPrimaryKey(id);
}
}
```

如此，数据库切库就OK了。如果你的系统已经有主库、从库之分了，那么赶紧在你的系统里利用起来吧。

扩展

这里呢，还可以支持多个扩展。比如现在一个主库后面有多个从库，在切面拿到需要切换从库时，还可以选择随机选择一个，或者根据类名、方法名或业务配置等选择某一个从库，这样不但可以分担每个从库的压力，也可以有针对性的让指定的读请求打到指定的从库上。如果有多个主库，也可以有更多的选择~

本文转载自：<http://url.cn/5ewMEc1>

JAVA MySQL



32人点赞

相关课程



Java并发编程入门与高并发...
¥ 299 · 初级 · 82977



Java开发企业级权限管理系统
¥ 388 · 中级 · 81383



学习算法思想，修炼编程内功
¥ 166 · 中级 · 87729



Java入门第一季
入门 · 8951354



Java入门第二季
入门 · 8453019



SSM到Spring Boot入门与...
¥ 966 · 4步骤/22门课 · 829



4 评论

评论 共同学习，写下你的评论



不悔1919

4楼

这是我参照老师手记构建的项目，有需要的可以下载
https://gitee.com/pgs/mybatis_dynamic.git



3

回复

2019.01.08



行望星烁

老师 请问一下mapper-locations不需要在配置类中配置吗，我的程序报java.io.FileNotFoundException: Could not open ServletContext resource [/classpath*/mapper/*.xml]的错误



0

回复

2018.10.08



宝慕林8331477 回复 行望星烁

目前我也遇到这个问题

回复

2018-11-07





宝慕林8331477 回复 行望星烁
mybatis.mapper-locations=classpath:/mapper/*.xml 加上这个路径配置，还是提示读取不到
回复 2018-11-07



宝慕林8331477 回复 行望星烁
似乎读取配置property文件的方式，spring解析带有通配符的路径时，有异常；临时解决方案是硬编码指定解析路径：sqlSessionFactoryBean.setMapperLocations((new PathMatchingResourcePatternResolver().getResources("classpath:/mapper/*.xml"))); 这样可以实现遍历多个文件
回复 2018-11-08



防腐基
动态切库竟然这么复杂的吗
👍 2 回复 2018.07.19



Oak_miss
好厉害啊，我也要试试，做个demo出来
👍 2 回复 2018.04.09

相关课程



Java并发编程入门与高并发...
¥ 299 · 初级 · 82977



Java开发企业级权限管理系统
¥ 388 · 中级 · 81383



学习算法思想，修炼编程内功
¥ 166 · 中级 · 87729



Java入门第一季
入门 · 8951354



Java入门第二季
入门 · 8453019



SSM到Spring Boot入门与...
¥ 966 · 4步骤/22门课 · 829

相关文章推荐



自定义注解完成参数校验
👁 248 慕灵守护 JAVA 11.14



在mysql数据库——自定义函数
👁 524 php小白兔秀秀 MySQL 02.06



php自定义函数库
👁 2 狐的传说 PHP 06.06



数据库系列：mysql存储过程与自定义函数
👁 25 素小暖 MySQL 05.08



数据库架构和对象、定义数据完整性-SQL Server
👁 55 DIEA SQL Server 04.08



最完美的数据库连接代码 (+完整注释与解析)
👁 3116 玄鉴 JAVA · MySQL · Oracle 08.19



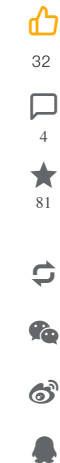
SQL Server-数据库架构和对象、定义数据完整性 (二)
👁 15 蛊毒传说 SQL Server 04.20



切面+自定义注解的一些玩法
👁 93 vizard JAVA · SpringBoot · 架构 08.12



PHP 自定义session储存 数据库 方式类
👁 2 慕码人2483693 PHP 05.22




32


4



81











JAVA里自定义注解来进行数据验证

👁 5 炎炎设计 JAVA 09.11

NoSql

Ruby

Flask

C#

Yii


爬虫

SSM

Go


C

Python




深入RxEasyHttp网络库教你3分钟学会自定义数据结构

👁 5 手掌心 Android 09.13




重新定义数据库的时刻，阿里云数据库专家带你了解POLARDB

👁 4 慕神8447489 大数据 12.03




Java自定义注解

👁 78 蝴蝶刀刀 JAVA 09.24



基于注解的Spring AOP开发--定义切入点函数

👁 3 慕的地10843 SpringBoot 11.19



Java中自定义注解

👁 4 蝴蝶刀刀 JAVA 09.04

相关课程



Java并发编程入门与高并发...

¥ 299 · 初级 · 82977



Java开发企业级权限管理系统

¥ 388 · 中级 · 81383



学习算法思想，修炼编程内功

¥ 166 · 中级 · 87729



Java入门第一季

入门 · 8951354



Java入门第二季

入门 · 8453019



SSM到Spring Boot入门与...

¥ 966 · 4步骤/22门课 · 829









