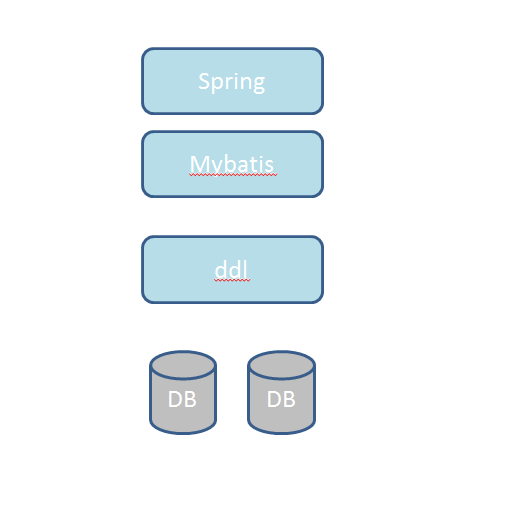
# 体系架构

该中间件是在Spring和Mybatis的基础上做的，主要解决分库分表场景下的访问路由，体系架构如下图：



# 二、主要优点

1、轻量化

2、支持自定义分库分表规则（热数据处理，多数据中心，读写分离）

2、支持Spring原生事务（单库事务）

3、分库分表规则支持多个维度

4、支持关联表shard（主表和关联表使用相同的分表规则）

# 三、缺点

1、只支持Spring和Mybatis

2、Service业务入口方法必须通过注解指定分库分表维度（支持多个shard维度）

3、Service业务入口方法必须配置spring事务（因为shard规则是在spring事务拦截器的基础上实现的）

4、提前预知数据源数量、不支持动态变更数据源

# 四、如何接入

1、中间件jar包maven引入。

<dependency>

<groupId>com.sonic</groupId>

<artifactId>sonic-ddl</artifactId>

<version>0.1</version>

</dependency>

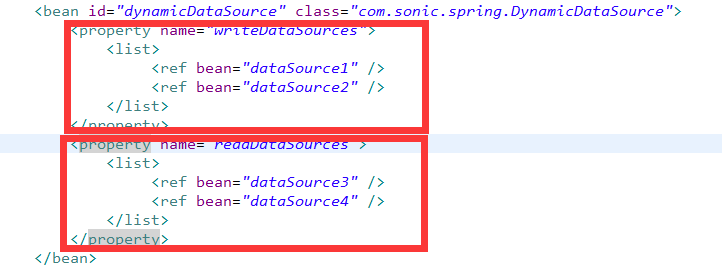
2、在spring中配置相关组件bean，如下图：

配置数据源





配置动态数据源



读数据源“writeDataSources”是有序的，分库规则就是根据dataSource在writeDataSources的位置计算的。

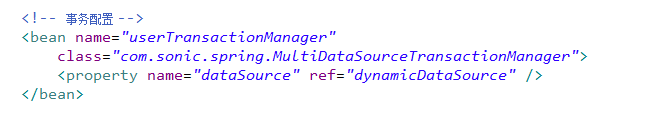
读数据源“readDataSources”也是有序的，规则和写是一样的，该配置是可选项，如果配置了readDataSources，那么数据源的数量应该和writeDataSources的数量是一致的，同时顺序应该也是一致的

配置分表



com.sonic.base.ShardStratetyImpl的tableNums属性是配置每个库有几张分表。

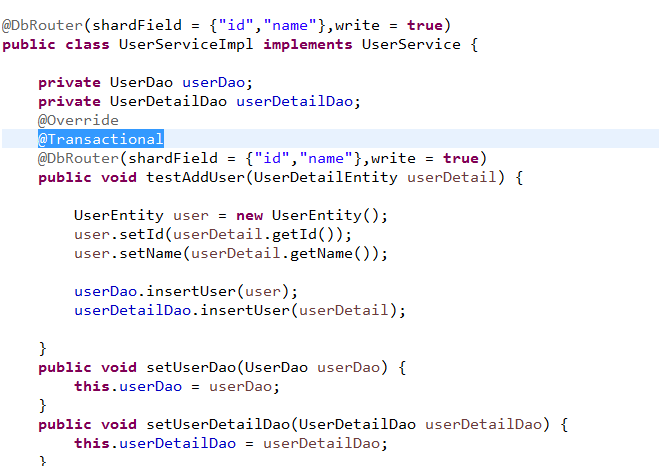
配置事务管理器和动态数据源



配置中间件注解扫描

C:\Users\X\AppData\Roaming\Tencent\Users\150607813\QQ\WinTemp\RichOle\_~5%U(@BJX})%UH%YZGDKCT.png

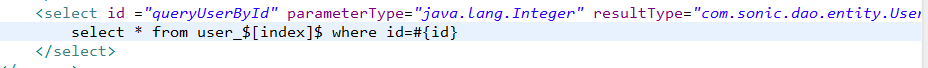
在业务层配置shard规则



分库分表规则可以配置在业务类上，也可以配置在方法上（方法上的配置覆盖类上的配置），shardFiled可以指定多个规则（多个对象属性值，原理是获取service方法的所有参数获取相关属性值），write =true表示使用写数据源，false表示读数据源，默认为true。

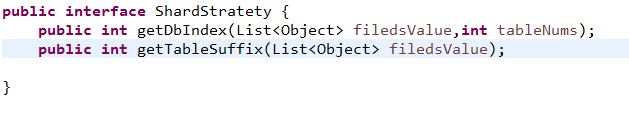
注意：Service上的事务是必须配置的，如果是读就配置读事务

3、mybatis mapping文件配置如下图：

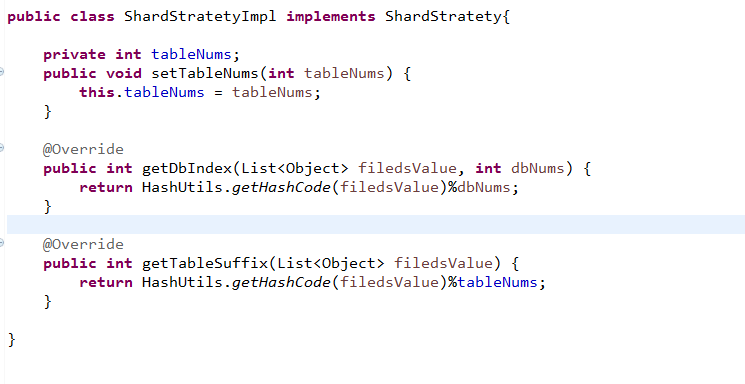


4、如何自定义分库分表规则

需要实现ShardStratety接口，如下图



默认实现类采用的是求模，如下图



上图的方法参数“filedsValue”是和DbRouter的shardFild值是一一对应的；dbNums是表示 “writeDataSources”和“readDataSources”分别包含的dataSource的size。

如果要实现多数据中心只需要在“writeDataSources”中配置多个数据中心的数据库地址信息，同时根据filedsValue中的参数实现不同的路由即可，如果要配置写原理同上。