数据源配置文件：

**spring:**

**datasource:**

**type:** com.alibaba.druid.pool.DruidDataSource

**driverClassName:** com.mysql.jdbc.Driver

**druid:**

**first:** *# 数据源1：配置中心表所在库*

**url:** jdbc:<mysql://localhost:3306/cc?allowMultiQueries=true&useUnicode=true&characterEncoding=UTF-8>

**username:** root

**password:** tiger

**second:** *# 数据源2：顾问云企业表所在库*

**url:** jdbc:<mysql://localhost:3306/gwy?allowMultiQueries=true&useUnicode=true&characterEncoding=UTF-8>

**username:** root

**password:** 123qwe

**initial-size:** 10

**max-active:** 100

**min-idle:** 10

**max-wait:** 60000

**pool-prepared-statements: true**

**max-pool-prepared-statement-per-connection-size:** 20

**time-between-eviction-runs-millis:** 60000

**min-evictable-idle-time-millis:** 300000

**validation-query:** SELECT 1 FROM DUAL

**test-while-idle: true**

**test-on-borrow: false**

**test-on-return: false**

**stat-view-servlet:**

**enabled: true**

**url-pattern:** /druid/\*

*#login-username: admin*

*#login-password: admin*

**filter:**

**stat:**

**log-slow-sql: true**

**slow-sql-millis:** 1000

**merge-sql: true**

**wall:**

**config:**

**multi-statement-allow: true**

**配置数据源**

**@Configuration**

**public class** **DynamicDataSourceConfig {**

**@Bean**

**@ConfigurationProperties("spring.datasource.druid.first")**

**public** **DataSource** **firstDataSource(){**

**return** **DruidDataSourceBuilder.***create***().build();**

**}**

**@Bean**

**@ConfigurationProperties("spring.datasource.druid.second")**

**public** **DataSource** **secondDataSource(){**

**return** **DruidDataSourceBuilder.***create***().build();**

**}**

**@Bean**

**@Primary**

**public** **DynamicDataSource** **dataSource(DataSource firstDataSource,** **DataSource secondDataSource) {**

**Map<String,** **DataSource> targetDataSources =** **new** **HashMap<>();**

**targetDataSources.put(DataSourceNames.***FIRST***,** **firstDataSource);**

**targetDataSources.put(DataSourceNames.***SECOND***,** **secondDataSource);**

**return new** **DynamicDataSource(firstDataSource,** **targetDataSources);**

**}**

**}**

**动态数据源：**

**public class** **DynamicDataSource** **extends** **AbstractRoutingDataSource {**

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

**public** **DynamicDataSource(DataSource defaultTargetDataSource,** **Map<String,** **DataSource> targetDataSources) {**

**super.setDefaultTargetDataSource(defaultTargetDataSource);**

**super.setTargetDataSources(new** **HashMap<>(targetDataSources));**

**super.afterPropertiesSet();**

**}**

**@Override**

**protected** **Object** **determineCurrentLookupKey() {**

**return** *getDataSource***();**

**}**

**public static void** **setDataSource(String dataSource) {**

*contextHolder***.set(dataSource);**

**}**

**public static** **String** **getDataSource() {**

**return** *contextHolder***.get();**

**}**

**public static void** **clearDataSource() {**

*contextHolder***.remove();**

**}**

**}**

**注解：**

**@Target(ElementType.***METHOD***)**

**@Retention(RetentionPolicy.***RUNTIME***)**

**@Documented**

**public** **@interface** **DataSource** **{**

**String** **name()** **default** **"";**

**}**

**切面处理类：**

**@Aspect**

**@Component**

**public class** **DataSourceAspect** **implements** **Ordered {**

**protected** **Logger** **logger** **= LoggerFactory.***getLogger***(getClass());**

**@Pointcut("@annotation(io.renren.datasources.annotation.DataSource)")**

**public void** **dataSourcePointCut() {**

**}**

**@Around("dataSourcePointCut()")**

**public** **Object** **around(ProceedingJoinPoint point)** **throws** **Throwable {**

**MethodSignature signature = (MethodSignature) point.getSignature();**

**Method method = signature.getMethod();**

**DataSource** **ds = method.getAnnotation(DataSource.class);**

**if(ds ==** **null){**

**DynamicDataSource.***setDataSource***(DataSourceNames.***FIRST***);**

**logger.debug("set datasource is "** **+ DataSourceNames.***FIRST***);**

**}else** **{**

**DynamicDataSource.***setDataSource***(ds.name());**

**logger.debug("set datasource is "** **+ ds.name());**

**}**

**try** **{**

**return** **point.proceed();**

**}** **finally** **{**

**DynamicDataSource.***clearDataSource***();**

**logger.debug("clean datasource");**

**}**

**}**

**@Override**

**public int** **getOrder() {**

**return** **1;**

**}**

**}**

**具体使用：**

**在service上加注解：@DataSource(name="")**

**其实原理就是先加载多数据源，然后通过extends** **AbstractRoutingDataSource 将默认数据源和目标多数据源装进去，通过切面实现数据源切换**