



Extjs 布局篇-上下左右布局

Spring对JSON请求加解密

线上服务CPU100%问题快速定位

网页Web上调用本地应用程序 (C#)

mysql主从复制

extjs4

jquery

j2ee

oracle

ajax

jsp/servlet

2018年4月

2018年3月

2018年1月

2017年12月

2017年8月

2017年7月

ajax执行先后顺序
4544

Extjs 中id与itemId的区别
2761

easyUI 获得对象中的对象属性，
对象的属性
2602

... 多个... 维数组进行组合... 度...

Spring+MyBatis实现读写分离四种实现方案整理

转载2017年04月24日 20:04:58

2461

Spring+MyBatis实现读写分离四种实现方案整理

方案1

通过MyBatis配置文件创建读写分离两个DataSource，每个SqlSessionFactoryBean对象的mapperLocations属性制定两个读写数据源的配置文件。将所有读的操作配置在读文件中，所有写的操作配置在写文件中。

- 优点：实现简单
- 缺点：维护麻烦，需要对原有的xml文件进行重新修改，不支持多读，不易扩展
- 实现方式

```
<bean id="abstractDataSource" abstract="true" class="com.alibaba.druid.pool.DruidDataSource"
    destroy-method="close">
    <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>
    <!-- 配置获取连接等待超时的时间 -->
    <property name="maxWait" value="60000"/>
    <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
    <property name="timeBetweenEvictionRunsMillis" value="60000"/>
    <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->
    <property name="minEvictableIdleTimeMillis" value="300000"/>
    <property name="validationQuery" value="SELECT 'x'"/>
    <property name="testWhileIdle" value="true"/>
    <property name="testOnBorrow" value="false"/>
    <property name="testOnReturn" value="false"/>
    <!-- 打开PSCache，并且指定每个连接上PSCache的大小 -->
    <property name="poolPreparedStatements" value="true"/>
    <property name="maxPoolPreparedStatementPerConnectionSize" value="20"/>
    <property name="filters" value="config"/>
    <property name="connectionProperties" value="config.decrypt=true" />
</bean>

<bean id="readDataSource" parent="abstractDataSource">
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${read.jdbc.url}"/>
    <property name="username" value="${read.jdbc.user}"/>
    <property name="password" value="${read.jdbc.password}"/>
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="${read.jdbc.initPoolSize}"/>
    <property name="minIdle" value="10"/>
    <property name="maxActive" value="${read.jdbc.maxPoolSize}"/>
</bean>

<bean id="writeDataSource" parent="abstractDataSource">
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${write.jdbc.url}"/>
    <property name="username" value="${write.jdbc.user}"/>
    <property name="password" value="${write.jdbc.password}"/>
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="${write.jdbc.initPoolSize}"/>
    <property name="minIdle" value="10"/>
    <property name="maxActive" value="${write.jdbc.maxPoolSize}"/>
</bean>

<bean id="readSqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 实例化sqlSessionFactory时需要使用上述配置好的数据源以及SQL映射文件 -->
    <property name="dataSource" ref="readDataSource"/>
    <property name="mapperLocations" value="classpath:mapper/read/*.xml"/>
</bean>
```

```
<!-- 实例化sqlSessionFactory时需要使用上述配置好的数据源以及SQL映射文件 -->
<property name="dataSource" ref="writeDataSource"/>
<property name="mapperLocations" value="classpath:mapper/write/*.xml"/>
</bean>
```

方案2

通过Spring AOP在业务层实现读写分离，在DAO层调用前定义切面，利用Spring的AbstractRoutingDataSource解决多数据源的问题，实现动态选择数据源

- 优点：通过注解的方法在DAO每个方法上配置数据源，原有代码改动量少，易扩展，支持多读
- 缺点：需要在DAO每个方法上配置注解，人工管理，容易出错
- 实现方式

```
//定义枚举类型，读写
public enum DynamicDataSourceGlobal {
    READ, WRITE;
}

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

/**
 * RUNTIME
 * 定义注解
 * 编译器将把注释记录在类文件中，在运行时 VM 将保留注释，因此可以反射性地读取。
 * @author shmal664
 *
 */
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface DataSource {

    public DynamicDataSourceGlobal value() default DynamicDataSourceGlobal.READ;

}

/**
 * Created by IDEA
 * 本地线程设置和获取数据源信息
 * User: mashaohua
 * Date: 2016-07-07 13:35
 * Desc:
 */
public class DynamicDataSourceHolder {

    private static final ThreadLocal<DynamicDataSourceGlobal> holder = new ThreadLocal<DynamicDataSourceGlobal>();

    public static void putDataSource(DynamicDataSourceGlobal dataSource){
        holder.set(dataSource);
    }

    public static DynamicDataSourceGlobal getDataSource(){
        return holder.get();
    }

    public static void clearDataSource() {
        holder.remove();
    }

}

import org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;
```

联系我们



请扫描二维码联系
webmaster@
400-660-0
QQ客服

关于 招聘 广告服务
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

```
import java.util.Map;
import java.util.concurrent.ThreadLocalRandom;
import java.util.concurrent.atomic.AtomicLong;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

/**
 * Created by IDEA
 * User: mashaohua
 * Date: 2016-07-14 10:56
 * Desc: 动态数据源实现读写分离
 */
public class DynamicDataSource extends AbstractRoutingDataSource {

    private Object writeDataSource; //写数据源

    private List<Object> readDataSources; //多个读数据源

    private int readDataSourceSize; //读数据源个数

    private int readDataSourcePollPattern = 0; //获取读数据源方式, 0: 随机, 1: 轮询

    private AtomicLong counter = new AtomicLong(0);

    private static final Long MAX_POOL = Long.MAX_VALUE;

    private final Lock lock = new ReentrantLock();

    @Override
    public void afterPropertiesSet() {
        if (this.writeDataSource == null) {
            throw new IllegalArgumentException("Property 'writeDataSource' is required");
        }
        setDefaultTargetDataSource(writeDataSource);
        Map<Object, Object> targetDataSources = new HashMap<>();
        targetDataSources.put(DynamicDataSourceGlobal.WRITE.name(), writeDataSource);
        if (this.readDataSources == null) {
            readDataSourceSize = 0;
        } else {
            for(int i=0; i<readDataSources.size(); i++) {
                targetDataSources.put(DynamicDataSourceGlobal.READ.name() + i, readDataSource
            }
            readDataSourceSize = readDataSources.size();
        }
        setTargetDataSources(targetDataSources);
        super.afterPropertiesSet();
    }

    @Override
    protected Object determineCurrentLookupKey() {

        DynamicDataSourceGlobal dynamicDataSourceGlobal = DynamicDataSourceHolder.getDataSou

        if(dynamicDataSourceGlobal == null
            || dynamicDataSourceGlobal == DynamicDataSourceGlobal.WRITE
            || readDataSourceSize <= 0) {
            return DynamicDataSourceGlobal.WRITE.name();
        }

        int index = 1;

        if(readDataSourcePollPattern == 1) {
            //轮询方式
            long currValue = counter.incrementAndGet();
            if((currValue + 1) >= MAX_POOL) {
                try {
                    lock.lock();
                    if((currValue + 1) >= MAX_POOL) {
                        counter.set(0);
                    }
                }
            }
        }
    }
}
```

联系我们



请扫描二维码联系

✉ webmaster@

☎ 400-660-0

🗣 QQ客服 🗣

[关于](#) [招聘](#) [广告服务](#) [🐾](#)

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



```

        lock.unlock();
    }
}
index = (int) (currValue % readDataSourceSize);
} else {
    //随机方式
    index = ThreadLocalRandom.current().nextInt(0, readDataSourceSize);
}
return dynamicDataSourceGlobal.name() + index;
}

public void setWriteDataSource(Object writeDataSource) {
    this.writeDataSource = writeDataSource;
}

public void setReadDataSources(List<Object> readDataSources) {
    this.readDataSources = readDataSources;
}

public void setReadDataSourcePollPattern(int readDataSourcePollPattern) {
    this.readDataSourcePollPattern = readDataSourcePollPattern;
}
}

import org.apache.log4j.Logger;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.reflect.MethodSignature;

import java.lang.reflect.Method;

/**
 * Created by IDEA
 * User: mashaohua
 * Date: 2016-07-07 13:39
 * Desc: 定义选择数据源切面
 */
public class DynamicDataSourceAspect {

    private static final Logger logger = Logger.getLogger(DynamicDataSourceAspect.class);

    public void pointCut();

    public void before(JoinPoint point)
    {
        Object target = point.getTarget();
        String methodName = point.getSignature().getName();
        Class<?>[] clazz = target.getClass().getInterfaces();
        Class<?>[] parameterTypes = ((MethodSignature) point.getSignature()).getMethod().getParameterTypes();
        try {
            Method method = clazz[0].getMethod(methodName, parameterTypes);
            if (method != null && method.isAnnotationPresent(DataSource.class)) {
                DataSource data = method.getAnnotation(DataSource.class);
                DynamicDataSourceHolder.putDataSource(data.value());
            }
        } catch (Exception e) {
            logger.error(String.format("Choose DataSource error, method:%s, msg:%s", methodName, e.getMessage()));
        }
    }

    public void after(JoinPoint point) {
        DynamicDataSourceHolder.clearDataSource();
    }
}

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.1.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.1.xsd">
```



0



```
<bean id="abstractDataSource" abstract="true" class="com.alibaba.druid.pool.DruidDataSour
    <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"
    <!-- 配置获取连接等待超时的时间 -->
    <property name="maxWait" value="60000"/>
    <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
    <property name="timeBetweenEvictionRunsMillis" value="60000"/>
    <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->
    <property name="minEvictableIdleTimeMillis" value="300000"/>
    <property name="validationQuery" value="SELECT 'x'"/>
    <property name="testWhileIdle" value="true"/>
    <property name="testOnBorrow" value="false"/>
    <property name="testOnReturn" value="false"/>
    <!-- 打开PSCache，并且指定每个连接上PSCache的大小 -->
    <property name="poolPreparedStatements" value="true"/>
    <property name="maxPoolPreparedStatementPerConnectionSize" value="20"/>
    <property name="filters" value="config"/>
    <property name="connectionProperties" value="config.decrypt=true" />
</bean>
```

```
<bean id="dataSourceRead1" parent="abstractDataSource">
    <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${read1.jdbc.url}"/>
    <property name="username" value="${read1.jdbc.user}"/>
    <property name="password" value="${read1.jdbc.password}"/>
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="${read1.jdbc.initPoolSize}"/>
    <property name="minIdle" value="${read1.jdbc.minPoolSize}"/>
    <property name="maxActive" value="${read1.jdbc.maxPoolSize}"/>
</bean>
```

```
<bean id="dataSourceRead2" parent="abstractDataSource">
    <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${read2.jdbc.url}"/>
    <property name="username" value="${read2.jdbc.user}"/>
    <property name="password" value="${read2.jdbc.password}"/>
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="${read2.jdbc.initPoolSize}"/>
    <property name="minIdle" value="${read2.jdbc.minPoolSize}"/>
    <property name="maxActive" value="${read2.jdbc.maxPoolSize}"/>
</bean>
```

```
<bean id="dataSourceWrite" parent="abstractDataSource">
    <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${write.jdbc.url}"/>
    <property name="username" value="${write.jdbc.user}"/>
    <property name="password" value="${write.jdbc.password}"/>
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="${write.jdbc.initPoolSize}"/>
    <property name="minIdle" value="${write.jdbc.minPoolSize}"/>
    <property name="maxActive" value="${write.jdbc.maxPoolSize}"/>
</bean>
```

```
<bean id="dataSource" class="com.test.api.dao.datasource.DynamicDataSource">
    <property name="writeDataSource" ref="dataSourceWrite" />
    <property name="readDataSources">
        <list>
            <ref bean="dataSourceRead1" />
            <ref bean="dataSourceRead2" />
        </list>
    </property>
</bean>
```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



```

<property name="readDataSourcePollPattern" value="1" />
<property name="defaultTargetDataSource" ref="dataSourceWrite"/>
</bean>

<tx:annotation-driven transaction-manager="transactionManager"/>

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransa
    <property name="dataSource" ref="dataSource"/>
</bean>

<!-- 针对myBatis的配置项 -->
<!-- 配置sqlSessionFactory -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 实例化sqlSessionFactory时需要使用上述配置好的数据源以及SQL映射文件 -->
    <property name="dataSource" ref="dataSource"/>
    <property name="mapperLocations" value="classpath:mapper/*.xml"/>
</bean>

<!-- 配置扫描器 -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <!-- 扫描包以及它的子包下的所有映射接口类 -->
    <property name="basePackage" value="com.test.api.dao.inte"/>
    <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>
</bean>

<!-- 配置数据库注解aop -->
<bean id="dynamicDataSourceAspect" class="com.test.api.dao.datasource.DynamicDataSourceAs
<aop:config>
    <aop:aspect id="c" ref="dynamicDataSourceAspect">
        <aop:pointcut id="tx" expression="execution(* com.test.api.dao.inte..*(..))"/>
        <aop:before pointcut-ref="tx" method="before"/>
        <aop:after pointcut-ref="tx" method="after"/>
    </aop:aspect>
</aop:config>
<!-- 配置数据库注解aop -->
</beans>

```

方案3

通过Mybatis的Plugin在业务层实现数据库读写分离，在MyBatis创建Statement对象前通过拦截器选择真正的数据源，在拦截器中根据方法名称不同（select、update、insert、delete）选择数据源。

- 优点：原有代码不变，支持多读，易扩展
- 缺点：
- 实现方式

```

/**
 * Created by IDEA
 * User: mashaohua
 * Date: 2016-07-19 15:40
 * Desc: 创建Connection代理接口
 */
public interface ConnectionProxy extends Connection {

    /**
     * 根据传入的读写分离需要的key路由到正确的connection
     * @param key 数据源标识
     * @return
     */
    Connection getTargetConnection(String key);
}

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.lang.reflect.Proxy;

```

联系我们



请扫描二维码联系

✉ webmaster@

☎ 400-660-0

🗣 QQ客服 📞

关于 招聘 广告服务 🐾

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```

import java.util.List;
import java.util.logging.Logger;

import javax.sql.DataSource;

import org.springframework.beans.factory.InitializingBean;
import org.springframework.jdbc.datasource.AbstractDataSource;
import org.springframework.jdbc.datasource.lookup.DataSourceLookup;
import org.springframework.jdbc.datasource.lookup.JndiDataSourceLookup;
import org.springframework.util.Assert;

public abstract class AbstractDynamicDataSourceProxy extends AbstractDataSource implements InitializingBean {

    private List<Object> readDataSources;
    private List<DataSource> resolvedReadDataSources;

    private Object writeDataSource;
    private DataSource resolvedWriteDataSource;

    private int readDataSourcePollPattern = 0;

    private int readDsSize;

    private boolean defaultAutoCommit = true;
    private int defaultTransactionIsolation = Connection.TRANSACTION_READ_COMMITTED;

    public static final String READ = "read";

    public static final String WRITE = "write";

    private DataSourceLookup dataSourceLookup = new JndiDataSourceLookup();

    @Override
    public Connection getConnection() throws SQLException {
        return (Connection) Proxy.newProxyInstance(
            com.autohome.api.dealer.tuan.dao.rwmybatis.ConnectionProxy.class.getClassLoader().loadClass(
                "com.autohome.api.dealer.tuan.dao.rwmybatis.ConnectionProxy.class"
            ).newInstance(),
            new RWConnectionInvocationHandler());
    }

    @Override
    public Connection getConnection(String username, String password) throws SQLException {
        return (Connection) Proxy.newProxyInstance(
            com.autohome.api.dealer.tuan.dao.rwmybatis.ConnectionProxy.class.getClassLoader().loadClass(
                "com.autohome.api.dealer.tuan.dao.rwmybatis.ConnectionProxy.class"
            ).newInstance(),
            new RWConnectionInvocationHandler(username, password));
    }

    public int getReadDsSize(){
        return readDsSize;
    }

    public List<DataSource> getResolvedReadDataSources() {
        return resolvedReadDataSources;
    }

    public void afterPropertiesSet() throws Exception {

        if(writeDataSource == null){
            throw new IllegalArgumentException("Property 'writeDataSource' is required");
        }
        this.resolvedWriteDataSource = resolveSpecifiedDataSource(writeDataSource);

        resolvedReadDataSources = new ArrayList<DataSource>(readDataSources.size());
        for(Object item : readDataSources){
            resolvedReadDataSources.add(resolveSpecifiedDataSource(item));
        }
        readDsSize = readDataSources.size();
    }
}

```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



```
protected DataSource determineTargetDataSource(String key) {
    Assert.notNull(this.resolvedReadDataSources, "DataSource router not initialized");
    if(WRITE.equals(key)){
        return resolvedWriteDataSource;
    }else{
        return loadReadDataSource();
    }
}

public Logger getParentLogger() {
    // NOOP Just ignore
    return null;
}

/**
 * 获取真实的数据 source
 * @param dataSource (jndi | real data source)
 * @return
 * @throws IllegalArgumentException
 */
protected DataSource resolveSpecifiedDataSource(Object dataSource) throws IllegalArgumentException {
    if (dataSource instanceof DataSource) {
        return (DataSource) dataSource;
    }
    else if (dataSource instanceof String) {
        return this.dataSourceLookup.getDataSource((String) dataSource);
    }
    else {
        throw new IllegalArgumentException(
            "Illegal data source value - only [javax.sql.DataSource] and String suppo
    )
}

protected abstract DataSource loadReadDataSource();

public void setReadDsSize(int readDsSize) {
    this.readDsSize = readDsSize;
}

public List<Object> getReadDataSources() {
    return readDataSources;
}

public void setReadDataSources(List<Object> readDataSources) {
    this.readDataSources = readDataSources;
}

public Object getWriteDataSource() {
    return writeDataSource;
}

public void setWriteDataSource(Object writeDataSource) {
    this.writeDataSource = writeDataSource;
}

public void setResolvedReadDataSources(List<DataSource> resolvedReadDataSources) {
    this.resolvedReadDataSources = resolvedReadDataSources;
}

public DataSource getResolvedWriteDataSource() {
    return resolvedWriteDataSource;
}

public void setResolvedWriteDataSource(DataSource resolvedWriteDataSource) {
    this.resolvedWriteDataSource = resolvedWriteDataSource;
}

public int getReadDataSourcePollPattern() {
    return readDataSourcePollPattern;
}
```

联系我们



请扫描二维码联系

✉ webmaster@

☎ 400-660-0

🗣 QQ客服 📞

关于 招聘 广告服务 🐾

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心


```

public void setReadDataSourcePollPattern(int readDataSourcePollPattern) {
    this.readDataSourcePollPattern = readDataSourcePollPattern;
}

/**
 * Invocation handler that defers fetching an actual JDBC Connection
 * until first creation of a Statement.
 */
private class RWConnectionInvocationHandler implements InvocationHandler {

    private String username;

    private String password;

    private Boolean readOnly = Boolean.FALSE;

    private Integer transactionIsolation;

    private Boolean autoCommit;

    private boolean closed = false;

    private Connection target;

    public RWConnectionInvocationHandler() {

    }

    public RWConnectionInvocationHandler(String username, String password) {
        this();
        this.username = username;
        this.password = password;
    }

    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        // Invocation on ConnectionProxy interface coming in...

        if (method.getName().equals("equals")) {
            // We must avoid fetching a target Connection for "equals".
            // Only consider equal when proxies are identical.
            return (proxy == args[0] ? Boolean.TRUE : Boolean.FALSE);
        }
        else if (method.getName().equals("hashCode")) {
            // We must avoid fetching a target Connection for "hashCode",
            // and we must return the same hash code even when the target
            // Connection has been fetched: use hashCode of Connection proxy.
            return new Integer(System.identityHashCode(proxy));
        }
        else if (method.getName().equals("getTargetConnection")) {
            // Handle getTargetConnection method: return underlying connection.
            return getTargetConnection(method, args);
        }
        else if (method.getName().equals("toString")) {
            return "RW Routing DataSource Proxy";
        }
        else if (method.getName().equals("isReadOnly")) {
            return this.readOnly;
        }
        else if (method.getName().equals("setReadOnly")) {
            this.readOnly = (Boolean) args[0];
            return null;
        }
        else if (method.getName().equals("getTransactionIsolation")) {
            return this.transactionIsolation;
        }
        else if (method.getName().equals("commit")) {
            return this.target.commit();
        }
        else if (method.getName().equals("rollback")) {
            return this.target.rollback();
        }
        else if (method.getName().equals("close")) {
            this.closed = true;
            return null;
        }
        else {
            // Default to target connection.
            return this.target.invoke(this, method, args);
        }
    }

    private Connection getTargetConnection(Method method, Object[] args) {
        // Only fetch target connection if we have not already.
        if (this.target == null) {
            // Create target connection.
            this.target = getConnection();
        }
        return this.target;
    }

    private Connection getConnection() {
        // Create connection.
        return DriverManager.getConnection(
            "jdbc:rw://" + username + ":" + password + "@localhost:3306/" +
            "databaseName", username, password);
    }
}

```



联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



```

    }
    return defaultTransactionIsolation;
    // Else fetch actual Connection and check there,
    // because we didn't have a default specified.
}
else if (method.getName().equals("setTransactionIsolation")) {
    this.transactionIsolation = (Integer) args[0];
    return null;
}
else if (method.getName().equals("getAutoCommit")) {
    if (this.autoCommit != null)
        return this.autoCommit;
    return defaultAutoCommit;
    // Else fetch actual Connection and check there,
    // because we didn't have a default specified.
}
else if (method.getName().equals("setAutoCommit")) {
    this.autoCommit = (Boolean) args[0];
    return null;
}
else if (method.getName().equals("commit")) {
    // Ignore: no statements created yet.
    return null;
}
else if (method.getName().equals("rollback")) {
    // Ignore: no statements created yet.
    return null;
}
else if (method.getName().equals("getWarnings")) {
    return null;
}
else if (method.getName().equals("clearWarnings")) {
    return null;
}
else if (method.getName().equals("isClosed")) {
    return (this.closed ? Boolean.TRUE : Boolean.FALSE);
}
else if (method.getName().equals("close")) {
    // Ignore: no target connection yet.
    this.closed = true;
    return null;
}
else if (this.closed) {
    // Connection proxy closed, without ever having fetched a
    // physical JDBC Connection: throw corresponding SQLException.
    throw new SQLException("Illegal operation: connection is closed");
}
}

// Target Connection already fetched,
// or target Connection necessary for current operation ->
// invoke method on target connection.
try {
    return method.invoke(target, args);
}
catch (InvocationTargetException ex) {
    throw ex.getTargetException();
}
}

/**
 * Return whether the proxy currently holds a target Connection.
 */
private boolean hasTargetConnection() {
    return (this.target != null);
}

/**
 * Return the target Connection, fetching it and initializing it if necessary.
 */

```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



```

if (this.target == null) {
    String key = (String) args[0];
    // No target Connection held -> fetch one.
    if (logger.isDebugEnabled()) {
        logger.debug("Connecting to database for operation '" + operation.getName()
    }

    // Fetch physical Connection from DataSource.
    this.target = (this.username != null) ?
        determineTargetDataSource(key).getConnection(this.username, this.password) :
        determineTargetDataSource(key).getConnection();

    // If we still lack default connection properties, check them now.
    //checkDefaultConnectionProperties(this.target);

    // Apply kept transaction settings, if any.
    if (this.readOnly.booleanValue()) {
        this.target.setReadOnly(this.readOnly.booleanValue());
    }
    if (this.transactionIsolation != null) {
        this.target.setTransactionIsolation(this.transactionIsolation.intValue())
    }
    if (this.autoCommit != null && this.autoCommit.booleanValue() != this.target.
        this.target.setAutoCommit(this.autoCommit.booleanValue());
    }
}

else {
    // Target Connection already held -> return it.
    if (logger.isDebugEnabled()) {
        logger.debug("Using existing database connection for operation '" + operation.getName()
    }
}

return this.target;
}
}
}

```

```
import javax.sql.DataSource;
import java.util.concurrent.ThreadLocalRandom;
import java.util.concurrent.atomic.AtomicLong;
import java.util.concurrent.locks.Lock;
import java.util.concurrent.locks.ReentrantLock;

/**
 * Created by IDEA
 * User: mashaohua
 * Date: 2016-07-19 16:04
 * Desc:
 */
public class DynamicRoutingDataSourceProxy extends AbstractDynamicDataSourceProxy {

    private AtomicLong counter = new AtomicLong(0);

    private static final Long MAX_POOL = Long.MAX_VALUE;

    private final Lock lock = new ReentrantLock();

    @Override
    protected DataSource loadReadDataSource() {
        int index = 1;


        if(getReadDataSourcePollPattern() == 1) {
            //轮询方式
            long currValue = counter.incrementAndGet();
        }
    }
}
```

联系我们



请扫描二维码联系

webmaster@

 400-660-0

QQ客服

关于 招聘 广告服务 

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



```

        lock.lock();
        if((currValue + 1) >= MAX_POOL) {
            counter.set(0);
        }
    } finally {
        lock.unlock();
    }
}

index = (int) (currValue % getReadDsSize());
} else {
    //随机方式
    index = ThreadLocalRandom.current().nextInt(0, getReadDsSize());
}
return getResolvedReadDataSources().get(index);
}
}

import org.apache.ibatis.executor.statement.RoutingStatementHandler;
import org.apache.ibatis.executor.statement.StatementHandler;
import org.apache.ibatis.mapping.MappedStatement;
import org.apache.ibatis.mapping.SqlCommandType;
import org.apache.ibatis.plugin.*;

import java.sql.Connection;
import java.util.Properties;

/**
 * 拦截器
 */
@Intercepts({ @Signature(type = StatementHandler.class, method = "prepare", args = { ConnectionProxy.class, MappedStatement.class, Object[] }) })
public class DynamicPlugin implements Interceptor {

    public Object intercept(Invocation invocation) throws Throwable {

        Connection conn = (Connection)invocation.getArgs()[0];
        //如果是采用了我们代理,则路由数据源
        if(conn instanceof com.autohome.api.dealer.tuan.dao.rwmybatis.ConnectionProxy){
            StatementHandler statementHandler = (StatementHandler) invocation
                .getTarget();

            MappedStatement mappedStatement = null;
            if (statementHandler instanceof RoutingStatementHandler) {
                StatementHandler delegate = (StatementHandler) ReflectionUtils
                    .getFieldValue(statementHandler, "delegate");
                mappedStatement = (MappedStatement) ReflectionUtils.getFieldValue(
                    delegate, "mappedStatement");
            } else {
                mappedStatement = (MappedStatement) ReflectionUtils.getFieldValue(
                    statementHandler, "mappedStatement");
            }
            String key = AbstractDynamicDataSourceProxy.WRITE;

            if(mappedStatement.getSqlCommandType() == SqlCommandType.SELECT){
                key = AbstractDynamicDataSourceProxy.READ;
            }else{
                key = AbstractDynamicDataSourceProxy.WRITE;
            }

            ConnectionProxy connectionProxy = (ConnectionProxy)conn;
            connectionProxy.getTargetConnection(key);

        }

        return invocation.proceed();
    }
}

```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```
    }

    public void setProperties(Properties properties) {
        //NOOP
    }
}

import org.apache.ibatis.logging.Log;
import org.apache.ibatis.logging.LogFactory;

import java.lang.reflect.*;

public class ReflectionUtils {

    private static final Log logger = LogFactory.getLog(ReflectionUtils.class);

    /**
     * 直接设置对象属性值,无视private/protected修饰符,不经过setter函数.
     */
    public static void setFieldValue(final Object object, final String fieldName, final Object value) {
        Field field = getDeclaredField(object, fieldName);

        if (field == null)
            throw new IllegalArgumentException("Could not find field [" + fieldName + "] on target object " + object);

        makeAccessible(field);

        try {
            field.set(object, value);
        } catch (IllegalAccessException e) {
            logger.error("Exception setting " + fieldName + " on " + object, e);
        }
    }

    /**
     * 直接读取对象属性值,无视private/protected修饰符,不经过getter函数.
     */
    public static Object getFieldValue(final Object object, final String fieldName) {
        Field field = getDeclaredField(object, fieldName);

        if (field == null)
            throw new IllegalArgumentException("Could not find field [" + fieldName + "] on target object " + object);

        makeAccessible(field);

        Object result = null;
        try {
            result = field.get(object);
        } catch (IllegalAccessException e) {
            logger.error("Exception getting " + fieldName + " on " + object, e);
        }

        return result;
    }

    /**
     * 直接调用对象方法,无视private/protected修饰符.
     */
    public static Object invokeMethod(final Object object, final String methodName, final Class[] parameterTypes, final Object[] parameters) throws InvocationTargetException {
        Method method = getDeclaredMethod(object, methodName, parameterTypes);
        if (method == null)
            throw new IllegalArgumentException("Could not find method [" + methodName + "] on target object " + object);

        method.setAccessible(true);

        return method.invoke(object, parameters);
    }
}
```



联系我们



请扫描二维码联系

✉ webmaster@

☎ 400-660-0

🗣 QQ客服 📞 电话

关于 招聘 广告服务 🐾

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



0



```

    }

    return null;
}

/**
 * 循环向上转型,获取对象的DeclaredField.
 */
protected static Field getDeclaredField(final Object object, final String fieldName) {
    for (Class<?> superClass = object.getClass(); superClass != Object.class; superClass
        .getSuperclass()) {
        try {
            return superClass.getDeclaredField(fieldName);
        } catch (NoSuchFieldException e) {
        }
    }
    return null;
}

/**
 * 循环向上转型,获取对象的DeclaredField.
 */
protected static void makeAccessible(final Field field) {
    if (!Modifier.isPublic(field.getModifiers()) || !Modifier.isPublic(field.getDeclaring
        field.setAccessible(true);
    }
}

/**
 * 循环向上转型,获取对象的DeclaredMethod.
 */
protected static Method getDeclaredMethod(Object object, String methodName, Class<?>[] pa
    for (Class<?> superClass = object.getClass(); superClass != Object.class; superClass
        .getSuperclass()) {
        try {
            return superClass.getDeclaredMethod(methodName, parameterTypes);
        } catch (NoSuchMethodException e) {
        }
    }
    return null;
}

/**
 * 通过反射,获得Class定义中声明的父类的泛型参数的类型.
 * eg.
 * public UserDao extends HibernateDao<User>
 *
 * @param clazz The class to introspect
 * @return the first generic declaration, or Object.class if cannot be determined
 */
@SuppressWarnings("unchecked")
public static <T> Class<T> getSuperClassGenericType(final Class clazz) {
    return getSuperClassGenericType(clazz, 0);
}

/**
 * 通过反射,获得Class定义中声明的父类的泛型参数的类型.
 * eg.
 * public UserDao extends HibernateDao<User>
 *
 * @param clazz The class to introspect
 * @return the first generic declaration, or Object.class if cannot be determined
 */
@SuppressWarnings("unchecked")
public static Class getSuperClassGenericType(final Class clazz, final int index) {
    Type genType = clazz.getGenericSuperclass();

```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```

        logger.warn(clazz.getSimpleName() + "'s superclass not ParameterizedType");
        return Object.class;
    }

    Type[] params = ((ParameterizedType) genType).getActualTypeArguments();

    if (index >= params.length || index < 0) {
        logger.warn("Index: " + index + ", Size of " + clazz.getSimpleName() + "'s Paramete
            + params.length);
        return Object.class;
    }
    if (!(params[index] instanceof Class)) {
        logger.warn(clazz.getSimpleName() + " not set the actual class on superclass gene
            return Object.class;
    }

    return (Class) params[index];
}

/**
 * 将反射时的checked exception转换为unchecked exception.
 */
public static IllegalArgumentException convertToUncheckedException(Exception e) {
    if (e instanceof IllegalAccessException || e instanceof IllegalArgumentException
        || e instanceof NoSuchMethodException)
        return new IllegalArgumentException("Refelction Exception.", e);
    else
        return new IllegalArgumentException(e);
}
}
}

```

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD SQL Map Config 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <plugins>
        <plugin interceptor="com.test.api.dao.mybatis.DynamicPlugin">
        </plugin>
    </plugins>

</configuration>

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-4.1.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-4.1.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-4.1.xsd">

    <bean id="abstractDataSource" abstract="true" class="com.alibaba.druid.pool.DruidDataSour
        <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver"
        <!-- 配置获取连接等待超时的时间 -->
        <property name="maxWait" value="60000"/>
        <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
        <property name="timeBetweenEvictionRunsMillis" value="60000"/>
        <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->
        <property name="minEvictableIdleTimeMillis" value="300000"/>
        <property name="validationQuery" value="SELECT 'x'"/>
        <property name="testWhileIdle" value="true"/>
        <property name="testOnBorrow" value="false"/>
        <property name="testOnReturn" value="false"/>
        <!-- 打开PSCache，并且指定每个连接上PSCache的大小 -->
        <property name="poolPreparedStatements" value="true"/>
    </bean>

```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

```
<property name="connectionProperties" value="config.decrypt=true" />
</bean>

<bean id="dataSourceRead1" parent="abstractDataSource">
    <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${read1.jdbc.url}" />
    <property name="username" value="${read1.jdbc.user}" />
    <property name="password" value="${read1.jdbc.password}" />
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="${read1.jdbc.initPoolSize}" />
    <property name="minIdle" value="${read1.jdbc.minPoolSize}" />
    <property name="maxActive" value="${read1.jdbc.maxPoolSize}" />
</bean>

<bean id="dataSourceRead2" parent="abstractDataSource">
    <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${read2.jdbc.url}" />
    <property name="username" value="${read2.jdbc.user}" />
    <property name="password" value="${read2.jdbc.password}" />
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="${read2.jdbc.initPoolSize}" />
    <property name="minIdle" value="${read2.jdbc.minPoolSize}" />
    <property name="maxActive" value="${read2.jdbc.maxPoolSize}" />
</bean>

<bean id="dataSourceWrite" parent="abstractDataSource">
    <property name="driverClassName" value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
    <!-- 基本属性 url、user、password -->
    <property name="url" value="${write.jdbc.url}" />
    <property name="username" value="${write.jdbc.user}" />
    <property name="password" value="${write.jdbc.password}" />
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="${write.jdbc.initPoolSize}" />
    <property name="minIdle" value="${write.jdbc.minPoolSize}" />
    <property name="maxActive" value="${write.jdbc.maxPoolSize}" />
</bean>

<bean id="dataSource" class="com.test.api.dao.datasource.DynamicRoutingDataSourceProxy">
    <property name="writeDataSource" ref="dataSourceWrite" />
    <property name="readDataSources">
        <list>
            <ref bean="dataSourceRead1" />
            <ref bean="dataSourceRead2" />
        </list>
    </property>
    <!--轮询方式-->
    <property name="readDataSourcePollPattern" value="1" />
</bean>

<tx:annotation-driven transaction-manager="transactionManager"/>

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransa
    <property name="dataSource" ref="dataSource"/>
</bean>

<!-- 针对myBatis的配置项 -->
<!-- 配置sqlSessionFactory -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 实例化sqlSessionFactory时需要使用上述配置好的数据源以及SQL映射文件 -->
    <property name="dataSource" ref="dataSource" />
    <property name="mapperLocations" value="classpath:mapper/*.xml" />
    <property name="configLocation" value="classpath:mybatis-plugin-config.xml" />
</bean>

<bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg ref="sqlSessionFactory" />
</bean>
```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心


```
<property name="basePackage" value="com.test.api.dao.inte" />
<property name="sqlSessionTemplate" ref="sqlSessionTemplate"></property>
</bean>

</beans>
```

方案4

如果你的后台结构是spring+mybatis，可以通过spring的AbstractRoutingDataSource和mybatis Plugin拦截器实现非常友好的读写分离，原有代码不需要任何改变。推荐第四种方案

```
import org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;

import java.util.HashMap;
import java.util.Map;

/**
 * Created by IDEA
 * User: mashaohua
 * Date: 2016-07-14 10:56
 * Desc: 动态数据源实现读写分离
 */
public class DynamicDataSource extends AbstractRoutingDataSource {

    private Object writeDataSource; //写数据源

    private Object readDataSource; //读数据源

    @Override
    public void afterPropertiesSet() {
        if (this.writeDataSource == null) {
            throw new IllegalArgumentException("Property 'writeDataSource' is required");
        }
        setDefaultTargetDataSource(writeDataSource);
        Map<Object, Object> targetDataSources = new HashMap<>();
        targetDataSources.put(DynamicDataSourceGlobal.WRITE.name(), writeDataSource);
        if(readDataSource != null) {
            targetDataSources.put(DynamicDataSourceGlobal.READ.name(), readDataSource);
        }
        setTargetDataSources(targetDataSources);
        super.afterPropertiesSet();
    }

    @Override
    protected Object determineCurrentLookupKey() {

        DynamicDataSourceGlobal dynamicDataSourceGlobal = DynamicDataSourceHolder.getDataSou

        if(dynamicDataSourceGlobal == null
            || dynamicDataSourceGlobal == DynamicDataSourceGlobal.WRITE) {
            return DynamicDataSourceGlobal.WRITE.name();
        }

        return DynamicDataSourceGlobal.READ.name();
    }

    public void setWriteDataSource(Object writeDataSource) {
        this.writeDataSource = writeDataSource;
    }

    public Object getWriteDataSource() {
        return writeDataSource;
    }

    public Object getReadDataSource() {
        return readDataSource;
    }
}
```

联系我们



请扫描二维码联系

✉ webmaster@

☎ 400-660-0

🗣 QQ客服 🗣

[关于](#) [招聘](#) [广告服务](#) [🐾](#)©1999-2018 CSDN版权所有
京ICP证09002463号[经营性网站备案信息](#)[网络110报警服务](#)[中国互联网举报中心](#)[北京互联网违法和不良信息举报中心](#)

```
    }
}

/**
 * Created by IDEA
 * User: mashaohua
 * Date: 2016-07-14 10:58
 * Desc:
 */
public enum DynamicDataSourceGlobal {
    READ, WRITE;
}

public final class DynamicDataSourceHolder {

    private static final ThreadLocal<DynamicDataSourceGlobal> holder = new ThreadLocal<DynamicDataSourceGlobal>();

    private DynamicDataSourceHolder() {
        //
    }

    public static void putDataSource(DynamicDataSourceGlobal dataSource){
        holder.set(dataSource);
    }

    public static DynamicDataSourceGlobal getDataSource(){
        return holder.get();
    }

    public static void clearDataSource() {
        holder.remove();
    }

}

import org.springframework.jdbc.datasource.DataSourceTransactionManager;
import org.springframework.transaction.TransactionDefinition;

/**
 * Created by IDEA
 * User: mashaohua
 * Date: 2016-08-10 14:34
 * Desc:
 */
public class DynamicDataSourceTransactionManager extends DataSourceTransactionManager {

    /**
     * 只读事务到读库，读写事务到写库
     * @param transaction
     * @param definition
     */
    @Override
    protected void doBegin(Object transaction, TransactionDefinition definition) {

        //设置数据源
        boolean readOnly = definition.isReadOnly();
        if(readOnly) {
            DynamicDataSourceHolder.putDataSource(DynamicDataSourceGlobal.READ);
        } else {
            DynamicDataSourceHolder.putDataSource(DynamicDataSourceGlobal.WRITE);
        }
        super.doBegin(transaction, definition);
    }

    /**
     * 清理本地线程的数据源
     * @param transaction
     */
}
```

联系我们



请扫描二维码联系

✉ webmaster@

☎ 400-660-0

🗣 QQ客服 🗣

[关于](#) [招聘](#) [广告服务](#) [🐾](#)

©1999-2018 CSDN版权所有

京ICP证09002463号

[经营性网站备案信息](#)[网络110报警服务](#)[中国互联网举报中心](#)[北京互联网违法和不良信息举报中心](#)

```
protected void doCleanupAfterCompletion(Object transaction) {
    super.doCleanupAfterCompletion(transaction);
    DynamicDataSourceHolder.clearDataSource();
}

import org.apache.ibatis.executor.Executor;
import org.apache.ibatis.executor.keygen.SelectKeyGenerator;
import org.apache.ibatis.mapping.BoundSql;
import org.apache.ibatis.mapping.MappedStatement;
import org.apache.ibatis.mapping.SqlCommandType;
import org.apache.ibatis.plugin.*;
import org.apache.ibatis.session.ResultHandler;
import org.apache.ibatis.session.RowBounds;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.transaction.support.TransactionSynchronizationManager;

import java.util.Locale;
import java.util.Map;
import java.util.Properties;
import java.util.concurrent.ConcurrentHashMap;

/**
 * Created by IDEA
 * User: mashaohua
 * Date: 2016-08-10 11:09
 * Desc:
 */
@Intercepts({
    @Signature(type = Executor.class, method = "update", args = {
        MappedStatement.class, Object.class }),
    @Signature(type = Executor.class, method = "query", args = {
        MappedStatement.class, Object.class, RowBounds.class,
        ResultHandler.class }) })
public class DynamicPlugin implements Interceptor {

    protected static final Logger logger = LoggerFactory.getLogger(DynamicPlugin.class);

    private static final String REGEX = ".*insert\\u0020.*|.*delete\\u0020.*|.*update\\u0020.*";

    private static final Map<String, DynamicDataSourceGlobal> cacheMap = new ConcurrentHashMap<>();

    @Override
    public Object intercept(Invocation invocation) throws Throwable {

        boolean synchronizationActive = TransactionSynchronizationManager.isSynchronizationActive();
        if(!synchronizationActive) {
            Object[] objects = invocation.getArgs();
            MappedStatement ms = (MappedStatement) objects[0];

            DynamicDataSourceGlobal dynamicDataSourceGlobal = null;

            if((dynamicDataSourceGlobal = cacheMap.get(ms.getId())) == null) {
                //读方法
                if(ms.getSqlCommandType().equals(SqlCommandType.SELECT)) {
                    //!selectKey 为自增id查询主键(SELECT LAST_INSERT_ID() )方法, 使用主库
                    if(ms.getId().contains(SelectKeyGenerator.SELECT_KEY_SUFFIX)) {
                        dynamicDataSourceGlobal = DynamicDataSourceGlobal.WRITE;
                    } else {
                        BoundSql boundSql = ms.getSqlSource().getBoundSql(objects[1]);
                        String sql = boundSql.getSql().toLowerCase(Locale.CHINA).replaceAll(" ", "");
                        if(sql.matches(REGEX)) {
                            dynamicDataSourceGlobal = DynamicDataSourceGlobal.WRITE;
                        } else {
                            dynamicDataSourceGlobal = DynamicDataSourceGlobal.READ;
                        }
                    }
                } else {
                    dynamicDataSourceGlobal = DynamicDataSourceGlobal.WRITE;
                }
            }
        }
    }
}
```

联系我们



请扫描二维码联系

webmaster@

400-660-0

QQ客服

关于 招聘 广告服务

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心

👍

0

🔖

🔖

🔖

🔖

🔖

```
        logger.warn("设置方法[{}] use [{}] Strategy, SqlCommandType [{}].", ms.getId(
            cacheMap.put(ms.getId(), dynamicDataSourceGlobal);
        }
        DynamicDataSourceHolder.putDataSource(dynamicDataSourceGlobal);
    }

    return invocation.proceed();
}

@Override
public Object plugin(Object target) {
    if (target instanceof Executor) {
        return Plugin.wrap(target, this);
    } else {
        return target;
    }
}

@Override
public void setProperties(Properties properties) {
    //
}
}
```

联系我们



请扫描二维码联系
✉ webmaster@
☎ 400-660-0
🗣 QQ客服 📞

关于 招聘 广告服务 🐾
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

看 Python 如何诠释“薪”时代

Python全栈开发包含Python爬虫、前端、网站后台、Python机器学习与数据挖掘等，从0基础小白到Python 企业级web开发达人、自动化运维开发能手的进击，课程真实企业项目实战演练，全面系统学习python编程语言，从容应对企业中各式各样的.....

查看更多>>

29224

查看更多>>

👤 目前您尚未登录，请 [登录](#) 或 [注册](#) 后参与评论



u013051497 2018-04-10 15:00 #1楼

回复

您好，能否请教下方案四中的spring如何配置多个数据源，就是一个写库，多个读库

MyBatis多数据源配置(读写分离)



isea533 2015年07月09日 13:45 57682

MyBatis多数据源配置(读写分离)首先说明，本文的配置使用的最直接的方式，实际用起来可能会很麻烦。实际应用中可能存在多种结合的情况，你可以理解本文的含义，不要死板的使用。多数据源的可能情况1.主从...

Spring+Mybatis透明实现读写分离



johnstrive 2016年09月20日 16:58 1835

背景网上有好多读写分离的实践，所应对的业务场景也不一样，本方法主要是应对中小型互联网产品的读写分离。数据库环境：1台master；2台slaver适用框架：spring+mybatis操作数据库的简单...

spring boot学习7之mybatis+mysql读写分离(一写多读)+事务

当业务的访问量（数据库的查询）非常大时，为了降低数据库的压力，希望有多个数据库进行负载均衡，避免所有的查询都集中在一台数据库，造成数据库压力过大。mysql支持一主多从，即在写库的数据库发生变动时，会...



加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录 注册

mysql+spring+mybatis实现数据库读写分离[代码配置]

mysql+spring+mybatis实现数据库读写分离[代码配置]

xtj332 2015年02月26日 16:50 38148

👍

🔖

💬

mybatis 读写分离

基于Mybatis的dao层读写分离组件。 实现原理： 1.实现spring的AbstractRoutingDataSource抽象类，AbstractRoutingDataSource本身实现了...

👤

xunwei0303

2016年08月23日 15:16

📖 499

💬

[技术分享]-Mybatis配置多个数据源（Java）

首先，使用Mybatis配置多个数据源需要用到两个工具类：SqlSessionFactory MapperFactory /** * 根据mybatis.xml中配置的不同environ...

👤

qiaoqiao0609

2018年01月31日 17:26

📖 294

🔖

mybatis 应用层读写分离

1.配置两个数据源到动态数据源 id="dataSource" class="com.che.rw.datasource.DynamicDataSource"> name="default...

👤

u010694931

2016年10月17日 16:11

📖 253

🔖

Spring和MyBatis实现数据的读写分离

1.Spring实现数据库的读写分离 现在大型的电子商务系统，在数据库层面大都采用读写分离技术，就是一个Master数据库，多个Slave数据库。Master库负责数据更新和实时数据查询，S...

👤

he90227

2016年04月26日 10:07

📖 10749

🔖

spring + mybatis + mysql(主从) 配置多个resource（读写分离）

配置多个resource 实现mysql读写分离准备工作： 1.需要一个能够运行的spring + mybatis的项目 2.mysql 主从数据库着手改造原来的项目 mysql 高可用...

👤

Linpk0315

2016年05月26日 10:20

📖 2450

🔖

Spring+MyBatis实现数据库读写分离方案

推荐第四种 方案1 通过MyBatis配置文件创建读写分离两个DataSource，每个SqlSessionFactoryBean对象的mapperLocations属性制定两个读写数据源的配...

👤

xwnxwn

2016年12月27日 11:16

📖 2069

🔖

spring MVC、mybatis配置读写分离

1.环境： 3台数据库机器，一个master，二台slave，分别为slave1，slave2 2.要实现的目标： ①使数据写入到master ②读数据时，从slave1和...

👤

lixiucheng005

2013年12月18日 11:34

📖 19340

🔖

使用mybatis +spring 插件实现读写分离

最近的项目中在数据库优化的时候需要用到读写分离，由于代码已经写好了，所以最优的方式就是在代码端不做任何修改，由于mybatis的灵活性，所以考虑用mybatis执行的时候通过插件的形式进行动态设置数据...

👤

yixiaogang109

2015年12月26日 11:34

📖 2108

🔖

spring+mybatis实现数据库读写分离

读写分离是为了减少数据库的负荷，当用户高并发访问时，绝大部分都是用户查询，少部分用户是写入到数据库的。这些我们把数据库拆分成主从两个数据库，主数据库用高性能 服务器承载高并发的用户访问并加...

👤

CSDNzhangtao5

2016年10月28日 15:36

📖 915

🔖

Spring+MyBatis实现数据库读写分离方案

转：http://www.jianshu.com/p/2222257f96d3 推荐第四种：https://github.com/shawntime/shawn-rwdb 方案1 通过MyBatis...

👤

lcathm

2017年05月08日 20:06

📖 1048

联系我们



请扫描二维码联系
webmaster@
400-660-0
QQ客服

关于 招聘 广告服务
©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息
网络110报警服务
中国互联网举报中心
北京互联网违法和不良信息举报中心

spring+mybatis利用interceptor(plugin)实现数据库读写分离

使用spring的动态路由实现数据库负载均衡 系统中存在的多台服务器是“地位相当”的，不过，同一时间他们都处于活动(Active)状态，处于负载均衡等因素考虑，数据访问请求需要在这几台数据库服务器...

 keda8997110

2013年11月19日 17:04

 25021



0





springboot+mybatis(读写分离)

 zjlzt1989

2017年06月12日 18:00

 981

pringboot+mybatis 整合有两种方式 即注解和配置文件。此demo中使用的是配置文件方式，个人觉得此种方式更便于维护和阅读。此文中只列出目录结构以及每个类的作用，详细代码见github...

spring+mybatis多数据源配置、读写分离

 guying4875

2016年08月25日 16:23

 1484

实现思路：在spring中配置多个数据源，然后在service层通过注解方式标明方法所要使用的数据源，利用springAOP在service方法执行前根据方法上的注解明确所要使用的数据源。如下图 ...

Spring和MyBatis实现数据的读写分离

 xyw591238

2016年04月28日 14:09

 2343

1.Spring实现数据库的读写分离 现在大型的电子商务系统，在数据库层面大都采用读写分离技术，就是一个Master数据库，多个Slave数据库。Master库负责数据更新和实时数据查询...

spring+spring mvc +mybatis+druid 实现数据库主从分离

本文是基于:spring+spring mvc +mybatis+druid为基础框架， 实现mysql数据库主从分离. 第一步:基于java annotation(注解)并通过spring aop...

 zhoushiwengang

2016年04月07日 18:32

 7163

Spring MVC+jdbcTemplate+MySql实现读写分离

 zbraccp

2016年12月29日 10:01


 391


现如今，数据库读写分离已经不再是新鲜的话题，很多数据统计网站、访问量高的网站都会使用这项技术。解决网站响应慢、服务器承载压力过大的方案其实有很多，而读写分离只是其中一种实现方案。有的时候读写分离并不...

联系我们



请扫描二维码联系

 webmaster@

 400-660-0

 QQ客服 

[关于](#) [招聘](#) [广告服务](#) 

©1999-2018 CSDN版权所有
京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心