Spring+MyBatis多数据源配置实现

最近用到了MyBatis配置多数据源，原以为简单配置下就行了，实际操作后发现还是要费些事的，这里记录下，以作备忘

不多废话，直接上代码，后面会有简单的实现介绍

jdbc和log4j的配置

```
#定义输出格式
ConversionPattern=%d %-5p [%t] %c - %m%n

log4j.rootLogger=DEBUG,Console
log4j.logger.com.cnblogs.lzrabbit=DEBUG
log4j.logger.org.springframework=ERROR
log4j.logger.org.mybatis=ERROR
log4j.logger.org.apache.ibatis=ERROR
log4j.logger.org.quartz=ERROR
log4j.logger.org.apache.axis2=ERROR
log4j.logger.org.apache.axiom=ERROR
log4j.logger.org.apache=ERROR
log4j.logger.httpclient=ERROR
#log4j.additivity.org.springframework=false
#Console
log4j.appender.Console=org.apache.log4j.ConsoleAppender
log4j.appender.Console.Threshold=DEBUG
log4j.appender.Console.Target=System.out
log4j.appender.Console.layout=org.apache.log4j.PatternLayout
log4j.appender.Console.layout.ConversionPattern=${ConversionPattern}
#log4j.appender.Console.encoding=UTF-8

#org.apache.log4j.DailyRollingFileAppender
log4j.appender.DailyFile=org.apache.log4j.DailyRollingFileAppender
log4j.appender.DailyFile.DatePattern='.'yyyy-MM-dd'.log'
log4j.appender.DailyFile.File=${myApp.root}/logs/daily.log
log4j.appender.DailyFile.Append=true
log4j.appender.DailyFile.Threshold=DEBUG
log4j.appender.DailyFile.layout=org.apache.log4j.PatternLayout
log4j.appender.DailyFile.layout.ConversionPattern=${ConversionPattern}
log4j.appender.DailyFile.encoding=UTF-8

#    %c 输出日志信息所属的类的全名
#    %d 输出日志时间点的日期或时间，默认格式为ISO8601，也可以在其后指定格式，比如：%d{yyy-MM-dd HH:m
#    %f 输出日志信息所属的类的类名
#    %l 输出日志事件的发生位置，即输出日志信息的语句处于它所在的类的第几行
#    %m 输出代码中指定的信息，如log(message)中的message
#    %n 输出一个回车换行符，Windows平台为“rn”，Unix平台为“n”
#    %p 输出优先级，即DEBUG，INFO，WARN，ERROR，FATAL。如果是调用debug()输出的，则为DEBUG，依此类
#    %r 输出自应用启动到输出该日志信息所耗费的毫秒数
#    %t 输出产生该日志事件的线程名
```

log4j.properties

```
#===============================================================================
# MySQL
#===============================================================================
jdbc.mysql.driver=com.mysql.jdbc.Driver
jdbc.mysql.url=jdbc:mysql://127.0.0.1:3306/test?useUnicode=true&characterEncoding=UTF-8&a
jdbc.mysql.username=root
```

```
jdbc.mysql.password=root

#===============================================================================
# MS SQL Server
#===============================================================================
#jdbc.sqlserver.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
#jdbc.sqlserver.url=jdbc:sqlserver://127.0.0.1:1433;database=test;
#jdbc.sqlserver.username=sa
#jdbc.sqlserver.password=sa

#===============================================================================
# MS SQL Server (JTDS)
#===============================================================================
jdbc.sqlserver.driver=net.sourceforge.jtds.jdbc.Driver
jdbc.sqlserver.url=jdbc:jtds:sqlserver://127.0.0.1:1433/test
jdbc.sqlserver.username=sa
jdbc.sqlserver.password=sa



#===============================================================================
# 通用配置
#===============================================================================
jdbc.initialSize=5
jdbc.minIdle=5
jdbc.maxIdle=20
jdbc.maxActive=100
jdbc.maxWait=100000
jdbc.defaultAutoCommit=false
jdbc.removeAbandoned=true
jdbc.removeAbandonedTimeout=600
jdbc.testWhileIdle=true
jdbc.timeBetweenEvictionRunsMillis=60000
jdbc.numTestsPerEvictionRun=20
jdbc.minEvictableIdleTimeMillis=300000
```

jdbc.properties

单数据源时的Spring配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd
      http://www.springframework.org/schema/aop
          http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">
    <bean id="propertyConfigurer" class="org.springframework.beans.factory.config.Properf
        <property name="location" value="classpath:jdbc.properties"/>
    </bean>
       <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-metl
        <property name="driverClassName" value="${jdbc.mysql.driver}"/>
        <property name="url" value="${jdbc.mysql.url}"/>
        <property name="username" value="${jdbc.mysql.username}"/>
        <property name="password" value="${jdbc.mysql.password}"/>
        <property name="initialSize" value="${jdbc.initialSize}"/>
        <property name="minIdle" value="${jdbc.minIdle}"/>
        <property name="maxIdle" value="${jdbc.maxIdle}"/>
        <property name="maxActive" value="${jdbc.maxActive}"/>
        <property name="maxWait" value="${jdbc.maxWait}"/>
        <property name="defaultAutoCommit" value="${jdbc.defaultAutoCommit}"/>
        <property name="removeAbandoned" value="${jdbc.removeAbandoned}"/>
```

```
        <property name="removeAbandonedTimeout" value="${jdbc.removeAbandonedTimeout}"/>
        <property name="testWhileIdle" value="${jdbc.testWhileIdle}"/>
        <property name="timeBetweenEvictionRunsMillis" value="${jdbc.timeBetweenEvictionI
        <property name="numTestsPerEvictionRun" value="${jdbc.numTestsPerEvictionRun}"/>
        <property name="minEvictableIdleTimeMillis" value="${jdbc.minEvictableIdleTimeMi
    </bean>
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
    </bean>

    <!-- mybatis.spring自动映射 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.cnblogs.lzrabbit"/>
    </bean>

    <!-- 自动扫描，多个包以 逗号分隔 -->
    <context:component-scan base-package="com.cnblogs.lzrabbit"/>
    <aop:aspectj-autoproxy/>
</beans>
```

applicationContext.xml

多数据源时Spring配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd
      http://www.springframework.org/schema/aop
          http://www.springframework.org/schema/aop/spring-aop-3.0.xsd">
    <bean id="propertyConfigurer" class="org.springframework.beans.factory.config.Propert
        <property name="location" value="classpath:jdbc.properties"/>
    </bean>
    <bean id="sqlServerDataSource" class="org.apache.commons.dbcp.BasicDataSource" destro
        <property name="driverClassName" value="${jdbc.sqlserver.driver}"/>
        <property name="url" value="${jdbc.sqlserver.url}"/>
        <property name="username" value="${jdbc.sqlserver.username}"/>
        <property name="password" value="${jdbc.sqlserver.password}"/>
        <property name="initialSize" value="${jdbc.initialSize}"/>
        <property name="minIdle" value="${jdbc.minIdle}"/>
        <property name="maxIdle" value="${jdbc.maxIdle}"/>
        <property name="maxActive" value="${jdbc.maxActive}"/>
        <property name="maxWait" value="${jdbc.maxWait}"/>
        <property name="defaultAutoCommit" value="${jdbc.defaultAutoCommit}"/>
        <property name="removeAbandoned" value="${jdbc.removeAbandoned}"/>
        <property name="removeAbandonedTimeout" value="${jdbc.removeAbandonedTimeout}"/>
        <property name="testWhileIdle" value="${jdbc.testWhileIdle}"/>
        <property name="timeBetweenEvictionRunsMillis" value="${jdbc.timeBetweenEvictionI
        <property name="numTestsPerEvictionRun" value="${jdbc.numTestsPerEvictionRun}"/>
        <property name="minEvictableIdleTimeMillis" value="${jdbc.minEvictableIdleTimeMi
    </bean>
    <bean id="mySqlDataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-me
        <property name="driverClassName" value="${jdbc.mysql.driver}"/>
        <property name="url" value="${jdbc.mysql.url}"/>
        <property name="username" value="${jdbc.mysql.username}"/>
        <property name="password" value="${jdbc.mysql.password}"/>
        <property name="initialSize" value="${jdbc.initialSize}"/>
        <property name="minIdle" value="${jdbc.minIdle}"/>
        <property name="maxIdle" value="${jdbc.maxIdle}"/>
        <property name="maxActive" value="${jdbc.maxActive}"/>
```

```
        <property name="maxWait" value="${jdbc.maxWait}"/>
        <property name="defaultAutoCommit" value="${jdbc.defaultAutoCommit}"/>
        <property name="removeAbandoned" value="${jdbc.removeAbandoned}"/>
        <property name="removeAbandonedTimeout" value="${jdbc.removeAbandonedTimeout}"/>
        <property name="testWhileIdle" value="${jdbc.testWhileIdle}"/>
        <property name="timeBetweenEvictionRunsMillis" value="${jdbc.timeBetweenEviction
        <property name="numTestsPerEvictionRun" value="${jdbc.numTestsPerEvictionRun}"/>
        <property name="minEvictableIdleTimeMillis" value="${jdbc.minEvictableIdleTimeMi
    </bean>
    <bean id="multipleDataSource" class="com.cnblogs.lzrabbit.MultipleDataSource">
        <property name="defaultTargetDataSource" ref="mySqlDataSource"/>
        <property name="targetDataSources">
            <map>
                <entry key="mySqlDataSource" value-ref="mySqlDataSource"/>
                <entry key="sqlServerDataSource" value-ref="sqlServerDataSource"/>
            </map>
        </property>
    </bean>
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="multipleDataSource"/>
    </bean>

    <!-- mybatis.spring自动映射 -->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.cnblogs.lzrabbit"/>
    </bean>

    <!-- 自动扫描，多个包以 逗号分隔 -->
    <context:component-scan base-package="com.cnblogs.lzrabbit"/>
    <aop:aspectj-autoproxy/>
</beans>
```

applicationContext.xml

MultipleDataSource实现

```
package com.cnblogs.lzrabbit;

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

/**
 * Created by rabbit on 14-5-25.
 */
public class MultipleDataSource extends AbstractRoutingDataSource {
    private static final ThreadLocal<String> dataSourceKey = new InheritableThreadLocal<

    public static void setDataSourceKey(String dataSource) {
        dataSourceKey.set(dataSource);
    }

    @Override
    protected Object determineCurrentLookupKey() {
        return dataSourceKey.get();
    }
}
```

MyBatis接口Mapper定义，直接使用注解方式实现

```
public interface MySqlMapper {
    @Select("select * from MyTable")
    List<Map<String,Object>> getList();
}

public interface SqlServerMapper {
        @Select("select * from MyTable")
```

```
        List<Map<String,Object>> getList();
}
```

手动数据源切换调用

```
package com.cnblogs.lzrabbit;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 * Created by rabbit on 14-5-25.
 */
public class Main {
    public static void main(String[] args) {
        //初始化ApplicationContext
        ApplicationContext applicationContext = new ClassPathXmlApplicationContext("appli

        MySqlMapper mySqlMapper = applicationContext.getBean(MySqlMapper.class);

        SqlServerMapper sqlServerMapper = applicationContext.getBean(SqlServerMapper.clas

        //设置数据源为MySql,使用了AOP测试时请将下面这行注释
        MultipleDataSource.setDataSourceKey("mySqlDataSource");
        mySqlMapper.getList();
        //设置数据源为SqlServer,使用AOP测试时请将下面这行注释
        MultipleDataSource.setDataSourceKey("sqlServerDataSource");
        sqlServerMapper.getList();
    }
}
```

使用SpringAOP方式实现自动切换

```
package com.cnblogs.lzrabbit;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Component
@Aspect
public class MultipleDataSourceAspectAdvice {

    @Around("execution(* com.cnblogs.lzrabbit.*.*(..))")
    public Object doAround(ProceedingJoinPoint jp) throws Throwable {
        if (jp.getTarget() instanceof MySqlMapper) {
            MultipleDataSource.setDataSourceKey("mySqlDataSource");
        } else if (jp.getTarget() instanceof SqlServerMapper) {
            MultipleDataSource.setDataSourceKey("sqlServerDataSource");
        }
        return jp.proceed();
    }
}
```

调用日志

```
2014-05-25 20:02:04,319 DEBUG [main] com.cnblogs.lzrabbit.MySqlMapper.getList - ooo Using
2014-05-25 20:02:04,333 DEBUG [main] com.cnblogs.lzrabbit.MySqlMapper.getList - ==>  Prep
2014-05-25 20:02:04,371 DEBUG [main] com.cnblogs.lzrabbit.MySqlMapper.getList - ==> Param
2014-05-25 20:02:04,396 DEBUG [main] com.cnblogs.lzrabbit.MySqlMapper.getList - <==
2014-05-25 20:02:04,620 DEBUG [main] com.cnblogs.lzrabbit.SqlServerMapper.getList - ooo U
2014-05-25 20:02:04,620 DEBUG [main] com.cnblogs.lzrabbit.SqlServerMapper.getList - ==>
2014-05-25 20:02:04,621 DEBUG [main] com.cnblogs.lzrabbit.SqlServerMapper.getList - ==> P
2014-05-25 20:02:04,681 DEBUG [main] com.cnblogs.lzrabbit.SqlServerMapper.getList - <==
```

这里就上面的实现做个简单解释，在我们配置单数据源时可以看到数据源类型使用了org.apache.commons.dbcp.BasicDataSource，而这个代码实现了javax.sql.DataSource接口

配置sqlSessionFactory时org.mybatis.spring.SqlSessionFactoryBean注入参数dataSource类型就是javax.sql.DataSource

实现多数据源的方法就是我们自定义了一个MultipleDataSource，这个类继承自AbstractRoutingDataSource，而AbstractRoutingDataSource继承自AbstractDataSource，AbstractDataSource 实现了javax.sql.DataSource接口，所以我们的MultipleDataSource也实现了javax.sql.DataSource接口，可以赋值给sqlSessionFactory的dataSource属性

```
public abstract class AbstractRoutingDataSource extends AbstractDataSource implements In
}

public abstract class AbstractDataSource implements DataSource {
}
```

再来说下MultipleDataSource的实现原理，MultipleDataSource实现AbstractRoutingDataSource抽象类，然后实现了determineCurrentLookupKey方法，这个方法用于选择具体使用targetDataSources中的哪一个数据源

```
<bean id="multipleDataSource" class="com.cnblogs.lzrabbit.MultipleDataSource">
        <property name="defaultTargetDataSource" ref="mySqlDataSource"/>
        <property name="targetDataSources">
            <map>
                <entry key="mySqlDataSource" value-ref="mySqlDataSource"/>
                <entry key="sqlServerDataSource" value-ref="sqlServerDataSource"/>
            </map>
        </property>
    </bean>
```

可以看到Spring配置中multipleDataSource设置了两个属性defaultTargetDataSource和targetDataSources，这两个属性定义在AbstractRoutingDataSource，当MyBatis执行查询时会先选择数据源，选择顺序时现根据determineCurrentLookupKey方法返回的值到targetDataSources中去找，若能找到怎返回对应的数据源，若找不到返回默认的数据源defaultTargetDataSource，具体参考AbstractRoutingDataSource的源码

```
public abstract class AbstractRoutingDataSource extends AbstractDataSource implements In

    private Map<Object, Object> targetDataSources;

    private Object defaultTargetDataSource;


    /**
     * Retrieve the current target DataSource. Determines the
     * {@link #determineCurrentLookupKey() current lookup key}, performs
     * a lookup in the {@link #setTargetDataSources targetDataSources} map,
     * falls back to the specified
     * {@link #setDefaultTargetDataSource default target DataSource} if necessary.
     * @see #determineCurrentLookupKey()
     */
    protected DataSource determineTargetDataSource() {
        Assert.notNull(this.resolvedDataSources, "DataSource router not initialized");
        Object lookupKey = determineCurrentLookupKey();
        DataSource dataSource = this.resolvedDataSources.get(lookupKey);
        if (dataSource == null && (this.lenientFallback || lookupKey == null)) {
            dataSource = this.resolvedDefaultDataSource;
        }
        if (dataSource == null) {
            throw new IllegalStateException("Cannot determine target DataSource for looku
        }
        return dataSource;
    }

    /**
     * Determine the current lookup key. This will typically be
```

```
     * implemented to check a thread-bound transaction context.
     * <p>Allows for arbitrary keys. The returned key needs
     * to match the stored lookup key type, as resolved by the
     * {@link #resolveSpecifiedLookupKey} method.
     */
    protected abstract Object determineCurrentLookupKey();

    ............

}
```

在动态切换数据源方法时选择了AOP方式实现，这里实现的简单粗暴，具体应用时根据实际需要灵活变通吧

题外话，这里提下SqlServer驱动选择的问题，目前SqlServer的驱动主要有微软的官方驱动和JTDS驱动两种，关于这两个驱动我做过测试，批量更新，在小数据量(100以下)时，JTDS相对微软驱动性能稍微高一点点，在数据量增大时几万到上百万时，微软驱动有着明显优势，所以若对性能比较敏感，建议使用微软驱动，否则随意

微软驱动在Maven库找不到，这点比较郁闷，若使用maven的话还得先安装到本地，这点很不爽

JTDS使用比较方便Maven直接引用即可

相关jar maven引用

```
<properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <org.springframework.version>3.2.7.RELEASE</org.springframework.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.7.2</version>
        </dependency>
        <dependency>
            <groupId>commons-dbcp</groupId>
            <artifactId>commons-dbcp</artifactId>
            <version>1.4</version>
        </dependency>
        <dependency>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.1.3</version>
        </dependency>
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
```

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>${org.springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.2.4</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>1.2.2</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.6</version>
        </dependency>
        <dependency>
            <groupId>net.sourceforge.jtds</groupId>
            <artifactId>jtds</artifactId>
            <version>1.2.8</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.29</version>
        </dependency>
    </dependencies>
```

View Code

posted @ 2014-05-25 20:50 懒惰的肥兔 阅读(...) 评论(...) 编辑 收藏

刷新评论刷新页面返回顶部