Spring 多数据源事务配置

作者：Veasion
原文链接：https://ld246.com/article/1504701312280
来源网站：链滴

简单的Spring多数据源配置事务（不支持分布式动态事务）：

直接贴代码(当做个笔记吧，毕竟也不会写博客)，spring配置：

```
    <!-- spring 配置 -->

    <!-- ================== 第一个数据源【db1】 ================== -->

    <bean id="dataSource_db1" class="com.alibaba.druid.pool.DruidDataSource" destroy-met
od="close" >
        <!-- 数据库基本信息配置 -->
        <property name="url" value="${jdbc.db1.url}" />
        <property name="username" value="${jdbc.db1.username}" />
        <property name="password" value="${jdbc.db1.password}"/>
        <property name="driverClassName" value="${driverClassName}" />
        <property name="filters" value="${filters}" />
        <!-- 最大并发连接数 -->
        <property name="maxActive" value="${maxActive}" />
        <!-- 初始化连接数量 -->
        <property name="initialSize" value="${initialSize}" />
        <!-- 配置获取连接等待超时的时间 -->
        <property name="maxWait" value="${maxWait}" />
        <!-- 最小空闲连接数 -->
        <property name="minIdle" value="${minIdle}" />
        <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
        <property name="timeBetweenEvictionRunsMillis" value="${timeBetweenEvictionRunsM
llis}" />
        <!-- 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="${minEvictableIdleTimeMillis}" />
        <property name="validationQuery" value="${validationQuery}" />
        <property name="testWhileIdle" value="${testWhileIdle}" />
        <property name="testOnBorrow" value="${testOnBorrow}" />
        <property name="testOnReturn" value="${testOnReturn}" />
        <property name="maxOpenPreparedStatements" value="${maxOpenPreparedStatemen
s}" />
        <!-- 打开 removeAbandoned 功能 -->
        <property name="removeAbandoned" value="${removeAbandoned}" />
        <!-- 1800 秒，也就是 30 分钟 -->
        <property name="removeAbandonedTimeout" value="${removeAbandonedTimeout}" /

        <!-- 关闭 abanded 连接时输出错误日志 -->
        <property name="logAbandoned" value="${logAbandoned}" />
    </bean>

    <bean id="sqlSessionFactory_db1" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="mapperLocations" value="classpath*:mybatis/mapper_db1/*.xml"></
roperty>
        <property name="dataSource" ref="dataSource_db1"></property>
        <property name="configLocation" value="classpath:spring/mybatis-config.xml"></prop
rty>
    </bean>

    <bean id="transactionManager_db1" class="org.springframework.jdbc.datasource.DataSou
```

```
ceTransactionManager">
        <property name="dataSource" ref="dataSource_db1" />
        <qualifier value="db1"/>
    </bean>

    <tx:annotation-driven transaction-manager="transactionManager_db1" />

    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.wheatek.dao.db1"></property>
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory_db1"></pro
erty>
    </bean>

    <aop:aspectj-autoproxy/>



    <!-- ================== 第一个数据源【db2】 ================== -->

    <bean id="dataSource_db2" class="com.alibaba.druid.pool.DruidDataSource" destroy-met
od="close" >
        <!-- 数据库基本信息配置 -->
        <property name="url" value="${jdbc.db2.url}" />
        <property name="username" value="${jdbc.db2.username}" />
        <property name="password" value="${jdbc.db2.password}"/>
        <property name="driverClassName" value="${driverClassName}" />
        <property name="filters" value="${filters}" />
        <!-- 最大并发连接数 -->
        <property name="maxActive" value="${maxActive}" />
        <!-- 初始化连接数量 -->
        <property name="initialSize" value="${initialSize}" />
        <!-- 配置获取连接等待超时的时间 -->
        <property name="maxWait" value="${maxWait}" />
        <!-- 最小空闲连接数 -->
        <property name="minIdle" value="${minIdle}" />
        <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
        <property name="timeBetweenEvictionRunsMillis" value="${timeBetweenEvictionRunsM
llis}" />
        <!-- 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="${minEvictableIdleTimeMillis}" />
        <property name="validationQuery" value="${validationQuery}" />
        <property name="testWhileIdle" value="${testWhileIdle}" />
        <property name="testOnBorrow" value="${testOnBorrow}" />
        <property name="testOnReturn" value="${testOnReturn}" />
        <property name="maxOpenPreparedStatements" value="${maxOpenPreparedStatemen
s}" />
        <!-- 打开 removeAbandoned 功能 -->
        <property name="removeAbandoned" value="${removeAbandoned}" />
        <!-- 1800 秒，也就是 30 分钟 -->
        <property name="removeAbandonedTimeout" value="${removeAbandonedTimeout}" /

        <!-- 关闭 abanded 连接时输出错误日志 -->
        <property name="logAbandoned" value="${logAbandoned}" />
    </bean>
```

```
    <bean id="sqlSessionFactory_db2" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="mapperLocations" value="classpath*:mybatis/mapper_db2/*.xml"></
roperty>
        <property name="dataSource" ref="dataSource_db2"></property>
        <property name="configLocation" value="classpath:spring/mybatis-config.xml"></prop
rty>
    </bean>

    <bean id="transactionManager_db2" class="org.springframework.jdbc.datasource.DataSou
ceTransactionManager">
        <property name="dataSource" ref="dataSource_db2" />
        <qualifier value="db2" />
    </bean>

    <tx:annotation-driven transaction-manager="transactionManager_db2" />

    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.wheatek.dao.db2"></property>
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory_db2"></pro
erty>
    </bean>
```

db1 Service Impl：

```
  @Service("testDb1Service")
  @Transactional(value="db1")
  public class TestDb1ServiceImpl implements TestService{

      @Resource
      private TestDb1Mapper testDb1Mapper;

      @Override
      public List<TestBean> select() {
          return testDb1Mapper.select();
      }

      @Override
      public int insert(TestBean testBean, boolean isBack) {
          int count=testDb1Mapper.insert(testBean);
          if(isBack){
              // 测试事物回滚
              System.out.println(1/0);
          }
          return count;
      }

  }
```

db2 Service Impl：

```
@Service("testDb2Service")
@Transactional(value="db2")
public class TestDb2ServiceImpl implements TestService{

    @Resource
    private TestDb2Mapper testDb2Mapper;

    @Override
    public List<TestBean> select() {
        return testDb2Mapper.select();
    }

    @Override
    public int insert(TestBean testBean, boolean isBack) {
        int count=testDb2Mapper.insert(testBean);
        if(isBack){
            // 测试事物回滚
            System.out.println(1/0);
        }
        return count;
    }

}
```