利用 Spring 的 AbstractRoutingDataSource 解决多数据源的问题

作者：huihui

原文链接：https://ld246.com/article/1480062270806

来源网站：链滴

多数据源问题很常见，例如读写分离数据库配置。

原来的项目出现了新需求，局方要求新增某服务器用以提供某代码，涉及到多数据源的问题。

研究成果如下：

1、首先配置多个datasource

```
1.
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
2.
    <property name="driverClassName" value="net.sourceforge.jtds.jdbc.Driver">
3.
    </property>
4.
    <property name="url" value="jdbc:jtds:sqlserver://10.82.81.51:1433;databaseName=standrds">
5.
    </property>
6.
    <property name="username" value="youguess"></property>
7.
    <property name="password" value="youguess"></property>
8.
 </bean>
9.
 <bean id="dataSource2" class="org.apache.commons.dbcp.BasicDataSource">
10.
    <property name="driverClassName" value="net.sourceforge.jtds.jdbc.Driver">
11.
```

```
    </property>
12.

    <property name="url" value="jdbc:jtds:sqlserver://10.82.81.52:1433;databaseName=standards">
13.

    </property>
14.

    <property name="username" value="youguess"></property>
15.

    <property name="password" value="youguess"></property>
16.

</bean>
```

2、写一个DynamicDataSource类继承AbstractRoutingDataSource，并实现determineCurrentLookupKey方法

```
1. package com.standard.core.util;
2. import org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;
3. public class DynamicDataSource extends AbstractRoutingDataSource {
4.

 @Override
5.

 protected Object determineCurrentLookupKey() {
6.

    return CustomerContextHolder.getCustomerType();
7.

 }
8. }
```

3、利用ThreadLocal解决线程安全问题

```
1. package com.standard.core.util;
```

```
2. public class CustomerContextHolder {
3.
 public static final String DATA_SOURCE_A = "dataSource";
4.
 public static final String DATA_SOURCE_B = "dataSource2";
5.
 private static final ThreadLocal contextHolder = new ThreadLocal();
6.
 public static void setCustomerType(String customerType) {
7.
    contextHolder.set(customerType);
8.
 }
9.
 public static String getCustomerType() {
10.
    return contextHolder.get();
11.
 }
12.
 public static void clearCustomerType() {
13.
    contextHolder.remove();
14.
 }
15. }
```

4、数据源配置

```
1.
<bean id="dynamicDataSource" class="com.standard.core.util.DynamicDataSource" >
2.
    <property name="targetDataSources">
3.
        <map key-type="java.lang.String">
4.
            <entry value-ref="dataSource" key="dataSource"></entry>
5.
            <entry value-ref="dataSource2" key="dataSource2"></entry>
6.
        </map>
7.
    </property>
8.
    <property name="defaultTargetDataSource" ref="dataSource" >
9.
    </property>
10.
 </bean>
```

5、在DAOImpl中切换数据源

```
1. CustomerContextHolder.setCustomerType(CustomerContextHolder.DATA_SOURCE_B);
```

搞定!