5. Spring Boot MyBatis配置Druid多数据源

回顾在Spring中配置MyBatis SqlSessionFactory的配置：

```xml
<!-- mybatis 的SqlSessionFactory -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean" scope="prototype">
  <property name="dataSource" ref="dataSource"/>
  <property name="configLocation" value="classpath:mybatis-config.xml"/>
</bean>
```

所以实际上在Spring Boot中配置MyBatis多数据源的关键在于创建SqlSessionFactory的时候为其分配不同的数据源。

引入依赖

先根据https://mrbird.cc/%E5%BC%80%E5%90%AFSpring-Boot.html开启一个最简单的Spring Boot应用，然后引入如下依赖：

```xml
<dependency>
  <groupId>org.mybatis.spring.boot</groupId>
  <artifactId>mybatis-spring-boot-starter</artifactId>
  <version>1.3.1</version>
</dependency>

<!-- oracle驱动 -->
<dependency>
  <groupId>com.oracle</groupId>
  <artifactId>ojdbc6</artifactId>
  <version>6.0</version>
</dependency>

<!-- mysql驱动 -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
</dependency>

<!-- druid数据源驱动 -->
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid-spring-boot-starter</artifactId>
  <version>1.1.6</version>
</dependency>
```

多数据源配置

在Spring Boot配置文件application.yml中配置多数据源和Spring Boot JdbcTemplate配置Druid多数据源一致。

然后根据application.yml创建两个数据源配置类MysqlDatasourceConfig和OracleDatasourceConfig：

MysqlDatasourceConfig：

```java
@Configuration
@MapperScan(basePackages = MysqlDatasourceConfig.PACKAGE, sqlSessionFactoryRef = "mysqlSqlSessionFactory")
public class MysqlDatasourceConfig {

    // mysqldao扫描路径
    static final String PACKAGE = "com.springboot.mysqldao";
    // mybatis mapper扫描路径
    static final String MAPPER_LOCATION = "classpath:mapper/mysql/*.xml";

    @Primary
    @Bean(name = "mysqldatasource")
    @ConfigurationProperties("spring.datasource.druid.mysql")
    public DataSource mysqlDataSource() {
        return DruidDataSourceBuilder.create().build();
    }

    @Bean(name = "mysqlTransactionManager")
    @Primary
    public DataSourceTransactionManager mysqlTransactionManager() {
        return new DataSourceTransactionManager(mysqlDataSource());
    }

    @Bean(name = "mysqlSqlSessionFactory")
    @Primary
    public SqlSessionFactory mysqlSqlSessionFactory(@Qualifier("mysqldatasource") DataSource dataSource)
    throws Exception {
        final SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();
        sessionFactory.setDataSource(dataSource);
        //如果不使用xml的方式配置mapper，则可以省去下面这行mapper location的配置。
        sessionFactory.setMapperLocations(new PathMatchingResourcePatternResolver()
                                          .getResources(MysqlDatasourceConfig.MAPPER_LOCATION));
        return sessionFactory.getObject();
    }
}
```

上面代码配置了一个名为mysqldatasource的数据源，对应application.yml中 `spring.datasource.druid.mysql` 前缀配置的数据库。然后创建了一个名为mysqlSqlSessionFactory的Bean，并且注入了mysqldatasource。与此同时，还分别定了两个扫描路径PACKAGE和MAPPER_LOCATION，前者为Mysql数据库对应的mapper接口地址，后者为对应的mapper xml文件路径。

`@Primary` 标志这个Bean如果在多个同类Bean候选时，该Bean优先被考虑。多数据源配置的时候，必须要有一个主数据源，用 `@Primary` 标志该Bean。

同理，接着配置Oracle数据库对应的配置类：

OracleDatasourceConfig：

```java
@Configuration
@MapperScan(basePackages = OracleDatasourceConfig.PACKAGE,
            sqlSessionFactoryRef = "oracleSqlSessionFactory")
public class OracleDatasourceConfig {

    // oracledao扫描路径
    static final String PACKAGE = "com.springboot.oracledao";
    // mybatis mapper扫描路径
    static final String MAPPER_LOCATION = "classpath:mapper/oracle/*.xml";

    @Bean(name = "oracledatasource")
    @ConfigurationProperties("spring.datasource.druid.oracle")
    public DataSource oracleDataSource() {
        return DruidDataSourceBuilder.create().build();
    }

    @Bean(name = "oracleTransactionManager")
    public DataSourceTransactionManager oracleTransactionManager() {
        return new DataSourceTransactionManager(oracleDataSource());
    }

    @Bean(name = "oracleSqlSessionFactory")
    public SqlSessionFactory oracleSqlSessionFactory(@Qualifier("oracledatasource") DataSource dataSource)
    throws Exception {
        final SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean();
        sessionFactory.setDataSource(dataSource);
        //如果不使用xml的方式配置mapper，则可以省去下面这行mapper location的配置。
        sessionFactory.setMapperLocations(new PathMatchingResourcePatternResolver()
                                          .getResources(OracleDatasourceConfig.MAPPER_LOCATION));
        return sessionFactory.getObject();
    }
}
```

测试

配置完多数据源，接下来分别在com.springboot.mysqldao路径和com.springboot.oracledao路径下创建两个mapper接口：

MysqlStudentMapper：

```java
package com.springboot.mysqldao;

import java.util.List;
import java.util.Map;
import org.apache.ibatis.annotations.Mapper;

@Mapper
public interface MysqlStudentMapper {
    List<Map<String, Object>> getAllStudents();
}
```

OracleStudentMapper：

```java
package com.springboot.oracledao;

import java.util.List;
import java.util.Map;
import org.apache.ibatis.annotations.Mapper;

@Mapper
public interface OracleStudentMapper {
    List<Map<String, Object>> getAllStudents();
}
```

接着创建mapper接口对应的实现：

在src/main/resource/mapper/mysql/路径下创建MysqlStudentMapper.xml：

XML

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.springboot.mysqldao.MysqlStudentMapper">
  <select id="getAllStudents" resultType="java.util.Map">
    select * from student
  </select>
</mapper>
```

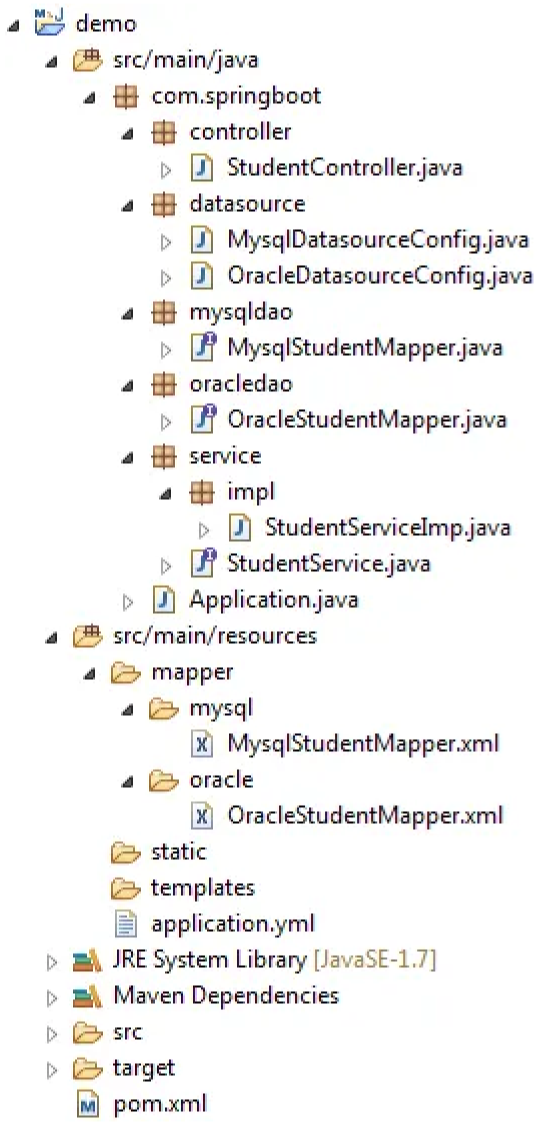
在src/main/resource/mapper/oracle/路径下创建OracleStudentMapper.xml：

XML

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.springboot.oracledao.OracleStudentMapper">
  <select id="getAllStudents" resultType="java.util.Map">
    select * from student
  </select>
</mapper>
```

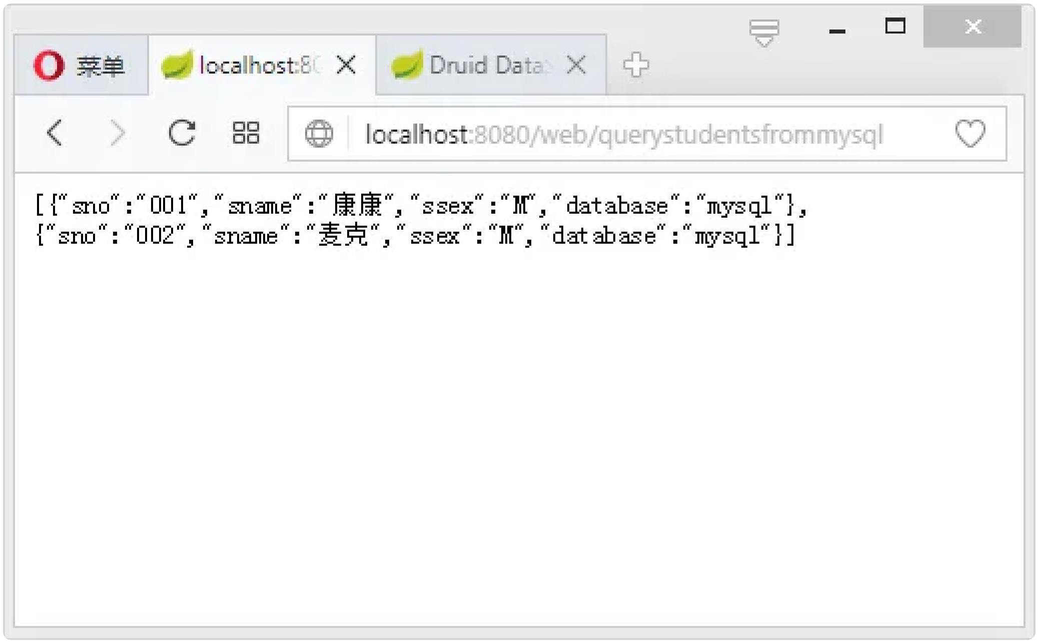
Service，Controller以及测试数据同Spring Boot JdbcTemplate配置Druid多数据源，这里不再赘述。

最终项目目录如下图所示：



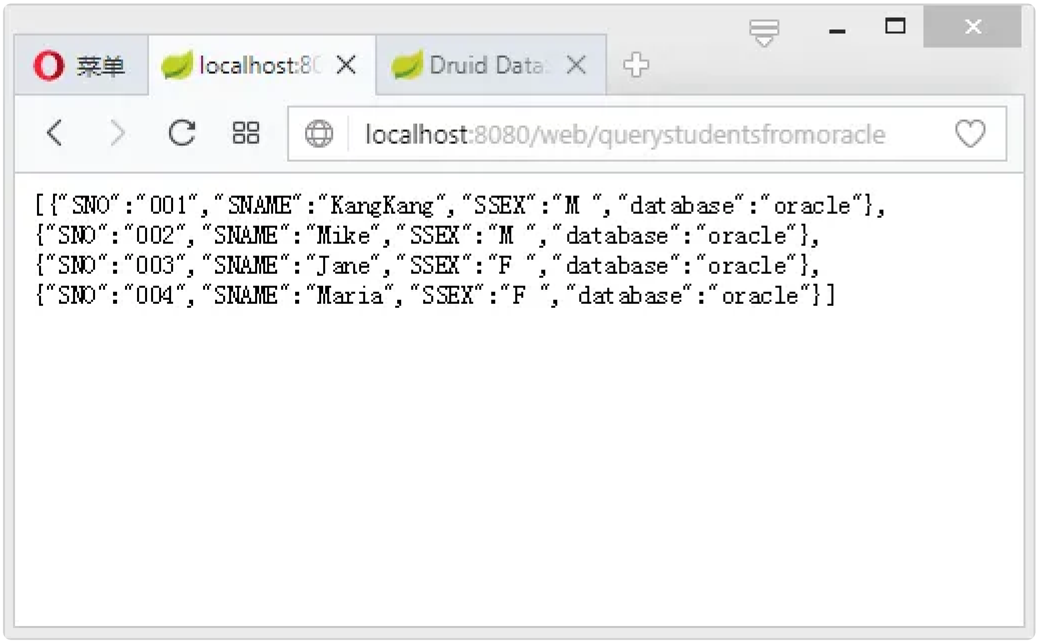
```
demo
  src/main/java
    com.springboot
      controller
        StudentController.java
      datasource
        MysqlDatasourceConfig.java
        OracleDatasourceConfig.java
      mysqldao
        MysqlStudentMapper.java
      oracledao
        OracleStudentMapper.java
      service
        impl
          StudentServiceImp.java
        StudentService.java
      Application.java
  src/main/resources
    mapper
      mysql
        MysqlStudentMapper.xml
      oracle
        OracleStudentMapper.xml
    static
    templates
    application.yml
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  src
  target
  pom.xml
```

启动项目，访问：http://localhost:8080/web/querystudentsfrommysql：



```
[{"sno":"001","sname":"康康","ssex":"M","database":"mysql"},
{"sno":"002","sname":"麦克","ssex":"M","database":"mysql"}]
```

http://localhost:8080/web/querystudentsfromoracle：



```
[{"SNO":"001","SNAME":"KangKang","SSEX":"M ","database":"oracle"},
{"SNO":"002","SNAME":"Mike","SSEX":"M ","database":"oracle"},
{"SNO":"003","SNAME":"Jane","SSEX":"F ","database":"oracle"},
{"SNO":"004","SNAME":"Maria","SSEX":"F ","database":"oracle"}]
```

source code