# SpringBoot + Mybatis 多数据源 注解版

# 基础环境

1. Springboot2.2.4
2. Mysql 8.+
3. Oracle11g

# 依赖导入

## web依赖

|  |
| --- |
| <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency> |

## MySQL驱动

|  |
| --- |
| <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <scope>runtime</scope>  </dependency> |

## Oracle 驱动+字符集支持包

|  |
| --- |
| <dependency>  <groupId>com.oracle.ojdbc</groupId>  <artifactId>ojdbc8</artifactId>  <scope>runtime</scope>  </dependency> |

|  |
| --- |
| <!-- https://mvnrepository.com/artifact/cn.easyproject/orai18n 添加如下依赖，否则会抱一个不支持字符集的错误！-->  <dependency>  <groupId>cn.easyproject</groupId>  <artifactId>orai18n</artifactId>  <version>12.1.0.2.0</version>  </dependency> |

## Mybatis-spring-boot依赖包

|  |
| --- |
| <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  <version>2.1.1</version>  </dependency> |

# 数据源信息配置

【此配置可以在 application.properties文件中进行设置】

## Mysql数据源

|  |
| --- |
| #1.配置 mysql 的数据源  spring.datasource.mysql.driver=com.mysql.cj.jdbc.Driver  spring.datasource.mysql.url=jdbc:mysql://localhost:3306/mybatis\_springboot?characterEncoding=utf8&useSSL=false&serverTimezone=UTC  spring.datasource.mysql.username=root  spring.datasource.mysql.password=123456 |

## Oracle数据源

|  |
| --- |
| #2.配置 oracle 的数据源  spring.datasource.oracle.driver=oracle.jdbc.driver.OracleDriver  spring.datasource.oracle.url=jdbc:oracle:thin:@localhost:1521/orcl  spring.datasource.oracle.username=scott  spring.datasource.oracle.password=tiger  spring.datasource.oracle.poolMaximumActiveConnections=30  spring.datasource.oracle.poolMaximumIdleConnections=10 |

# 数据源配置类

【这个配置类中，配置了所有的 数据源，根据3中设置的数据源的信息来的】

|  |
| --- |
| @Configuration |
|  |
| //0.通过内置的environment对象可以读取spring的配置文件中的内容  @Autowired  private Environment env; |

## Mysql数据源的配置bean

|  |
| --- |
| //1.1 配置第一个MySQL数据源作为主数据源  @Bean("dataSourceMysql")  @Primary // 表示 优先配置这个数据源，也就是默认的一个  public DataSource dataSourceMysql(){  PooledDataSource pooledDataSource = new PooledDataSource();  String driver = env.getProperty("spring.datasource.mysql.driver");  String url = env.getProperty("spring.datasource.mysql.url");  String username = env.getProperty("spring.datasource.mysql.username");  String password = env.getProperty("spring.datasource.mysql.password");  pooledDataSource.setDriver(driver);  pooledDataSource.setUrl(url);  pooledDataSource.setUsername(username);  pooledDataSource.setPassword(password);  pooledDataSource.setPoolPingEnabled(true);  pooledDataSource.setPoolPingQuery("select 1");  return pooledDataSource;  } |

## Oracle 数据源的配置bean

|  |
| --- |
| //1.2 配置第二个Oracle数据源作为次数据源  @Bean("dataSourceOracle")  public DataSource dataSourceOracle(){  PooledDataSource pooledDataSource = new PooledDataSource();  String driver = env.getProperty("spring.datasource.oracle.driver");  String url = env.getProperty("spring.datasource.oracle.url");  String username = env.getProperty("spring.datasource.oracle.username");  String password = env.getProperty("spring.datasource.oracle.password");  String maxActiveConn = env.getProperty("spring.datasource.oracle.poolMaximumActiveConnections");  String maxIdelConn = env.getProperty("spring.datasource.oracle.poolMaximumIdleConnections");  pooledDataSource.setDriver(driver);  pooledDataSource.setUrl(url);  pooledDataSource.setUsername(username);  pooledDataSource.setPassword(password);  pooledDataSource.setPoolMaximumActiveConnections(Integer.parseInt(maxActiveConn) );  pooledDataSource.setPoolMaximumIdleConnections(Integer.parseInt(maxIdelConn));  pooledDataSource.setPoolPingEnabled(true);  pooledDataSource.setPoolPingQuery("select 1 from dual");  return pooledDataSource;  } |

# Mybatis\_mysql配置类

## 注解使用

|  |
| --- |
| @Configuration  @MapperScan(basePackages = {"com.cidm.project02.mapper.mysql"},  sqlSessionFactoryRef = "sqlSessionFactoryMysql",  sqlSessionTemplateRef = "sqlSessionTemplateMysql") |

## sqlSessionFactory方法

|  |
| --- |
| @Bean("sqlSessionFactoryMysql")  @Primary // 表示默认用这个  public SqlSessionFactory sqlSessionFactoryMysql(  @Qualifier("dataSourceMysql")DataSource dataSource,  ApplicationContext applicationContext) throws Exception {  //0. 这里用到的是 SqlSessionFactoryBean 这个类，这个类就对应了mybatis原来的那个配置文件中的所有的内容  SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();  //1.设置数据源 ： 这个很关键  sqlSessionFactoryBean.setDataSource(dataSource);  //2.设置包的别名  sqlSessionFactoryBean.setTypeAliasesPackage("com.cidm.project02.model.mysql");  //3.设置 xml映射文件的路径地址 : 这个很关键,中间的模糊级别需要两个\*才可以，否则报错  sqlSessionFactoryBean.setMapperLocations(applicationContext.getResources("classpath:mybatis/mapper/mysql/\*\*/\*.xml"));  //4.设置是否开启驼峰命名法 ： 基本上没啥用  sqlSessionFactoryBean.setVfs(SpringBootVFS.class);  //5. 返回 sqlsessionfactory的对象  return sqlSessionFactoryBean.getObject();  } |
|  |

## sqlSessionTemplate方法

|  |
| --- |
| @Bean("sqlSessionTemplateMysql")  @Primary // 表示默认  public SqlSessionTemplate sqlSessionTemplateMysql(  @Qualifier("sqlSessionFactoryMysql") SqlSessionFactory sqlSessionFactory){  return new SqlSessionTemplate(sqlSessionFactory);  } |

# 6.Mybatis\_oracle 配置类

## 6.1 注解使用

|  |
| --- |
| @Configuration  @MapperScan(basePackages = {"com.cidm.project02.mapper.oracle"},  sqlSessionFactoryRef = "sqlSessionFactoryOracle",  sqlSessionTemplateRef = "sqlSessionTemplateOracle") |

## 6.2 sqlSessionFactory方法

|  |
| --- |
| @Bean("sqlSessionFactoryOracle")  public SqlSessionFactory sqlSessionFactoryOracle(  @Qualifier("dataSourceOracle")DataSource dataSource,  ApplicationContext applicationContext) throws Exception {  //0. 使用 SqlSessionFactoryBean 类进行配置  SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();  //1.设置数据源  sqlSessionFactoryBean.setDataSource(dataSource);  //2.设置包的别名  sqlSessionFactoryBean.setTypeAliasesPackage("com.cidm.project02.model.oracle");  //3.设置xml映射文件的位置  sqlSessionFactoryBean.setMapperLocations(applicationContext.getResources("classpath:mybatis/mapper/oracle/\*\*/\*.xml"));  //4.设置驼峰命名法  sqlSessionFactoryBean.setVfs(SpringBootVFS.class);  //5.返回对象  return sqlSessionFactoryBean.getObject();  } |

## 6.3 sqlSessionTemplate方法

|  |
| --- |
| @Bean("sqlSessionTemplateOracle")  public SqlSessionTemplate sqlSessionTemplateOracle(  @Qualifier("sqlSessionFactoryOracle") SqlSessionFactory sqlSessionFactory){  return new SqlSessionTemplate(sqlSessionFactory);  } |

# 7.正常开发

Model 层，mapper层，service接口，service实现类，controller类

每一个分开成 mysql 和 Oracle 两个部分的内容

# 8.项目结构图

