1 项目简介

- 项目引入了 `mybatis-plus-boot-starter` 方便操作数据库
- 配置了3个数据源，做动态数据源实验

```
 <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.5.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>
 <dependency>
     <groupId>com.baomidou</groupId>
     <artifactId>mybatis-plus-boot-starter</artifactId>
     <version>3.0.1</version>
 </dependency>
```

```
import com.alibaba.druid.spring.boot.autoconfigure.DruidDataSourceBuilder;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import javax.sql.DataSource;
import java.util.HashMap;
import java.util.Map;
@Configuration
public class DynamicDataSourceConfig {

    @Bean(name = "firstDataSource")
    @ConfigurationProperties("spring.datasource.druid.first")
    public DataSource firstDataSource(){
        return DruidDataSourceBuilder.create().build();
    }
    @Bean(name = "secondDataSource")
    @ConfigurationProperties("spring.datasource.druid.second")
    public DataSource secondDataSource(){
        return DruidDataSourceBuilder.create().build();
    }

    @Bean(name = "dataSource")
    //@Primary //这个是关键，如果没有这个注解，就会报标题中的错
```

```
    public DynamicDataSource dataSource(DataSource firstDataSource, DataSource
secondDataSource) {
        Map<Object, Object> targetDataSources = new HashMap<>();
        targetDataSources.put(DataSourceNames.FIRST, firstDataSource);
        targetDataSources.put(DataSourceNames.SECOND, secondDataSource);
        return new DynamicDataSource(firstDataSource, targetDataSources);
    }
}
```

2 本项目报错原因分析

- 字面上意思：需要 `sqlSessionFactory` 和 `sqlSessionTemplate` 。即，项目中需要用到这两个类，可是并没有注入到容器中。
- 根本原因：配置了3个数据源，并且没有使用 `@Primary` 注解注明主数据源，导致 `mybatis-plus-boot-starter` 中的配置文件不能初始化，代码如下。

```
@Configuration
@ConditionalOnClass({SqlSessionFactory.class, SqlSessionFactoryBean.class})
//关键是这个注解：因为容器中有3个数据源，且没有指定主数据源，这个条件不通过，就不会初始化这个配置类了
//这样容器中就没有报错中需要的两个类了
@ConditionalOnSingleCandidate(DataSource.class)
@EnableConfigurationProperties({MybatisPlusProperties.class})
@AutoConfigureAfter({DataSourceAutoConfiguration.class})
public class MybatisPlusAutoConfiguration {
  @Bean
  @ConditionalOnMissingBean
  public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception
{...}
  @Bean
  @ConditionalOnMissingBean
  public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory)
{...}
}
```

3 其它原因报这个错

- 项目中引入了 `mybatis-spring-boot-starter` ，但做了一下处理

```
//MybatisAutoConfiguration的条件就达不到了，就不会配置MybatisAutoConfiguration和其中的
'sqlSessionFactory' 和 'sqlSessionTemplate' 两个类
@SpringBootApplication(exclude={DataSourceAutoConfiguration.class})
//如果加了exclude={MybatisAutoConfiguration.class} 那更不行了...
@SpringBootApplication(exclude={MybatisAutoConfiguration.class})
public class SpringBootDatasourceApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringBootDatasourceApplication.class, args);
    }
}
```

4 总结

- 如果项目报某个资源找不到或需要某个资源，说明项目肯定需要这个资源，需要自己注入相关资源，或者看看是否因为没有引入相关包，或初始化这个资源需要别的条件
- 如果项目中有多个数据源，一定要使用 `@Primary` 注明主数据源，这样其它类需要依赖数据源时，能够匹配上主数据源

5 扩展-DataSourceAutoConfiguration的作用

1. 自动配置数据源——以配置文件以 `spring.datasource` 开头的配置项来配置
2. 注册 `dataSourceInitializerPostProcessor` 后置处理器，用来初始化数据源的时候自动执行 `schema.sql` 和 `data.sql`
3. 所以，一般情况下可以 `exclude={DataSourceAutoConfiguration.class}`，只要自定义了数据源即可。如果实在需要执行 `schema.sql`，可以抽取相关源码作为一个定时任务

参考

springboot如何运行项目中的sql文件 fescar多数据源出现的数据源循环依赖问题