SqlSessionFactoryBean

在基础的 MyBatis 用法中，是通过 `SqlSessionFactoryBuilder` 来创建 `SqlSessionFactory` 的。而在 MyBatis-Spring 中，则使用 `SqlSessionFactoryBean` 来创建。

设置

要创建工厂 bean，将下面的代码放到 Spring 的 XML 配置文件中：

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
</bean>
```

需要注意的是 `SqlSessionFactoryBean` 实现了 Spring 的 `FactoryBean` 接口（参见 Spring 官方文档 3.8 节 通过工厂 bean 自定义实例化逻辑🎈）。这意味着由 Spring 最终创建的 bean **并不是** `SqlSessionFactoryBean` 本身，而是工厂类（`SqlSessionFactoryBean`）的 getObject() 方法的返回结果。这种情况下，Spring 将会在应用启动时为你创建 `SqlSessionFactory`，并使用 `sqlSessionFactory` 这个名字存储起来。

等效的 Java 代码如下：

```
@Configuration
public class MyBatisConfig {
  @Bean
  public SqlSessionFactory sqlSessionFactory() {
    SqlSessionFactoryBean factoryBean = new SqlSessionFactoryBean();
    factoryBean.setDataSource(dataSource());
    return factoryBean.getObject();
  }
}
```

通常，在 MyBatis-Spring 中，你不需要直接使用 `SqlSessionFactoryBean` 或对应的 `SqlSessionFactory`。相反，session 的工厂 bean 将会被注入到 `MapperFactoryBean` 或其它继承于 `SqlSessionDaoSupport` 的 DAO（Data Access Object，数据访问对象）中。

属性

`SqlSessionFactory` 有一个唯一的必要属性：用于 JDBC 的 `DataSource`。这可以是任意的 `DataSource` 对象，它的配置方法和其它 Spring 数据库连接是一样的。

一个常用的属性是 `configLocation`，它用来指定 MyBatis 的 XML 配置文件路径。它在需要修改 MyBatis 的基础配置非常有用。通常，基础配置指的是 `<settings>` 或 `<typeAliases>` 元素。

需要注意的是，这个配置文件**并不需要**是一个完整的 MyBatis 配置。确切地说，任何环境配置（`<environments>`），数据源（`<dataSource>`）和 MyBatis 的事务管理器（`<transactionManager>`）都会被忽略。`SqlSessionFactoryBean` 会创建它自有的 MyBatis 环境配置（`Environment`），并按要求设置自定义环境的值。

如果 MyBatis 在映射器类对应的路径下找不到与之相对应的映射器 XML 文件，那么也需要配置文件。这时有两种解决办法：第一种是手动在 MyBatis 的 XML 配置文件中的 `<mappers>` 部分中指定 XML 文件的类路径；第二种是设置工厂 bean 的 `mapperLocations` 属性。

`mapperLocations` 属性接受多个资源位置。这个属性可以用来指定 MyBatis 的映射器 XML 配置文件的位置。属性的值是一个 Ant 风格的字符串，可以指定加载一个目录中的所有文件，或者从一个目录开始递归搜索所有目录。比如：

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="mapperLocations" value="classpath*:sample/config/mappers/**/*.xml" />
</bean>
```

这会从类路径下加载所有在 `sample.config.mappers` 包和它的子包中的 MyBatis 映射器 XML 配置文件。

在容器管理事务的时候，你可能需要的一个属性是 `transactionFactoryClass`。请参考事务一章的相关章节。

如果你使用了多个数据库，那么需要设置 `databaseIdProvider` 属性：

```
<bean id="databaseIdProvider" class="org.apache.ibatis.mapping.VendorDatabaseIdProvider">
  <property name="properties">
    <props>
      <prop key="SQL Server">sqlserver</prop>
      <prop key="DB2">db2</prop>
      <prop key="Oracle">oracle</prop>
      <prop key="MySQL">mysql</prop>
    </props>
  </property>
</bean>
```

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="mapperLocations" value="classpath*:sample/config/mappers/**/*.xml" />
  <property name="databaseIdProvider" ref="databaseIdProvider"/>
</bean>
```

仅限 自 1.3.0 版本开始，新增的 `configuration` 属性能够在没有对应的 MyBatis XML 配置文件的情况下，直接设置 `Configuration` 实例。例如：

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="configuration">
    <bean class="org.apache.ibatis.session.Configuration">
      <property name="mapUnderscoreToCamelCase" value="true"/>
    </bean>
  </property>
</bean>
```