## 1、添加依赖

<!--mybatis-->

<dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>2.1.0</version>  
</dependency>  
<!--mysql驱动-->  
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>5.1.46</version>  
</dependency>  
<!--druid数据库连接池-->  
<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid</artifactId>  
 <version>1.1.9</version>  
</dependency>

spring boot默认使用的是tomcat-jdbc数据源，为了方便，添加alibaba数据源。

mybatis-spring-boot-starter会做哪些事：

* 自动检测现有的DataSource
* 将创建并注册SqlSessionFactory的实例，该实例使用SqlSessionFactoryBean将该DataSource作为输入进行传递。
* 将创建并注册从SqlSessionFactory中获取的SqlSessionTemplate的实例。
* 自动扫描mappers，将它们链接到SqlSessionTemplate并将其注册到Spring上下文，以便将它们注入到您的bean中。

只要使用这个springboot mybatis starter 只需要DataSource的配置就可以使用了，它会自动创建使用该DataSource的SqlSessionFactoryBean以及SqlSessionTemplate。会自动扫描你的Mappers，连接到SqlSessionTemplate，并注册到Spring上下文中。

## 2、配置数据源

spring:

datasource:  
 *#使用阿里的Druid连接池* type: com.alibaba.druid.pool.DruidDataSource  
 *#数据库连接四要素* driver-class-name: com.mysql.jdbc.Driver  
 url: jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=utf-8&useSSL=true&serverTimezone=UTC  
 username: root  
 password: 539976  
 *#连接池的一些配置* druid:  
 initial-size: 10  
 max-active: 100  
 min-idle: 10  
 max-wait: 60000  
 pool-prepared-statements: true  
 max-pool-prepared-statement-per-connection-size: 20  
 time-between-eviction-runs-millis: 60000  
 min-evictable-idle-time-millis: 300000  
 validation-query: SELECT 1  
 test-while-idle: true  
 test-on-borrow: false  
 test-on-return: false  
 *#配置DruidStatViewServlet* stat-view-servlet:  
 enabled: true  
 url-pattern: /druid/\*  
 *#配置监控统计拦截的filters，去掉后监控界面sql无法统计，wall用于防火墙* filter:  
 stat:  
 log-slow-sql: true  
 slow-sql-millis: 1000  
 merge-sql: true  
 wall:  
 config:  
 multi-statement-allow: true

spring boot会自动加载spring.datasource.\*相关配置，数据源就会自动注入到sqlSessionFactory中，sqlSessionFactory会自动注入到Mapper中。

## 3、使用Mybatis

### 1）创建一个实体类

User.java

### 2）使用XML形式

创建User的映射类UserDao.java，也可以命名Mapper作为尾缀，编写curd接口，映射类上加上@Mapper注解。

@Mapper

public interface UserDao {  
   
 User getUser(Long id);  
  
 int insert(User user);  
  
 int delete(int id);  
  
 int update(User user);  
  
 List<User> getUsers();  
}

使用xml的时候需要注意的是Mybatis扫描mapper.xml并且装配，需要在系统的配置文件resources/application.yml加入：

*#Mybatis配置*

mybatis:  
 mapper-locations: classpath:mapper/\*.xml  
 type-aliases-package: com.chance.demo.pojo  
 type-handlers-package: com.chance.demo.typeHandler  
 *#将带有下划线的表字段映射为驼峰格式的实体类属性* configuration:  
 map-underscore-to-camel-case: true

根据自己的xml目录，进行配置。

例如：在resources/mapper/目录下加入UserMapper.xml文件，添加自己编写得SQL：

### 3）使用注解形式

这种形式没有UserMapper.xml文件，也不要配置它的文件路径的映射了，只要把xml中的SQL写到注解上就可以了。

根据数据库的操作不同，使用不同的注解：

* @Select是查询类的注解，所有的查询均使用这个。
* @Result修饰返回的结果集，关联实体类属性和数据库字段——一一对应，如果实体类属性和数据库属性名保持一致，就不需要这个属性来修饰。
* @Insert插入数据库使用，直接传入实体类会自动解析属性到对应的值。
* @Update负责修改，也可以直接传入对象。
* @Delete负责删除。

4）

如果没有在映射类上加@Mapper注解，也可以在运行的主类上添加@MapperScan(“com.chance.demo.dao”)注解包扫描。