引入mybatis依赖、数据库连接包、*使用阿里的Druid连接池*

<dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>2.0.0</version>  
</dependency>

<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>8.0.15</version>  
</dependency>

<dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid-spring-boot-starter</artifactId>  
 <version>1.1.13</version>  
</dependency>

配置文件（数据库配置，连接池配置，mybatis配置）：

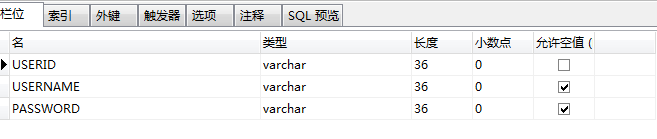
**spring**:  
 **profiles**:  
 **active**: dev  
 **datasource**:  
 **driver-class-name**: com.mysql.cj.jdbc.Driver  
 **type**: com.alibaba.druid.pool.DruidDataSource *# 使用阿里的Druid连接池* **url**: jdbc:mysql://localhost:3306/wems?serverTimezone=UTC&useSSL=true *# 填写你数据库的url和数据库名* **username**: root *#登录名* **password**: 123456 *#密码* **druid**:  
 **initial-size**: 3 *#初始化活跃连接数* **min-idle**: 5 *#最小连接池数* **max-active**: 20 *#最大连接池数* **max-wait**: 60000 *#获取连接等待超时的时间,单位是毫秒* **time-between-eviction-runs-millis**: 60000 *#间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒;testWhileIdle的判断依据，详细看testWhileIdle属性的说明* **time-between-connect-error-millis**: 18000 *#一旦重试多次失败后等待多久再继续重试连接，单位是毫秒* **min-evictable-idle-time-millis**: 300000 *#一个连接在池中最小生存的时间，单位是毫秒* **validationQuery**: SELECT 1 FROM DUAL *#检测连接是否有效的sql* **test-on-borrow**: **false** *#申请连接时执行validationQuery检测连接是否有效，做了这个配置会降低性能。* **test-on-return**: **false** *#归还连接时执行validationQuery检测连接是否有效，做了这个配置会降低性能* **test-while-idle**: **true** *#建议配置为true，不影响性能，并且保证安全性。申请连接的时候检测，如果空闲时间大于timeBetweenEvictionRunsMillis，执行validationQuery检测连接是否有效。* **pool-prepared-statements**: **true** *# 打开PSCache，* **max-pool-prepared-statement-per-connection-size**: 20 *# 为PSCache指定每个连接上PSCache的大小* **filters**: stat,wall,log4j2 *# 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* **connect-properties**: druid.stat.mergeSql\=true;druid.stat.slowSqlMillis\=5000 *# 通过connectProperties属性来打开mergeSql功能；慢SQL记录* **web-stat-filter**: *# 配置DruidStatFilter* **enabled**: **true  
 url-pattern**: "/\*"  
 **exclusions**: "\*.js,\*.gif,\*.jpg,\*.bmp,\*.png,\*.css,\*.ico,/druid/\*" *#排除一些不必要的url  
 # 配置DruidStatViewServlet* **stat-view-servlet**:

**enabled**: **true**  
 **url-pattern**: "/druid/\*"  
 *# IP白名单(没有配置或者为空，则允许所有访问)* **allow**: 127.0.0.1,192.168.163.1  
 *# IP黑名单 (存在共同时，deny优先于allow)* **deny**: 192.168.1.73  
 *# 禁用HTML页面上的“Reset All”功能* **reset-enable**: false  
 *# 登录名* **login-username**: admin  
 *# 登录密码* **login-password**: 123456

**mybatis**:  
 **type-aliases-package**: com.example.BACKEND.Entity *#实体类路径，用于xml文件中对实体类的映射* **mapper-locations**: classpath:Mapper/\*Mapper.xml *#Dao层与xml的映射路径*

mybatis：

1.数据库建表



2.新建实体映射类

public class Employee {  
 private String userid;  
 private String username;  
 private String password;  
 public String getUsername() {  
 return username;  
 }  
 public void setUsername(String username) {  
 this.username = username;  
 }  
 public String getUserid() {  
 return userid;  
 }  
 public void setUserid(String userid) {  
 this.userid = userid;  
 }  
 public String getPassword() {  
 return password;  
 }  
 public void setPassword(String password) {  
 this.password = password;  
 }  
   
}

3.Dao层

public interface EmployeeDao {  
 List<Employee> getAllEmployee();  
 List<Employee> getEmployeeByUsername(@Param("username") String username,@Param("password") String password);  
}

4.Mapper映射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.example.BACKEND.Dao.EmployeeDao">  
 <resultMap id="EmployeeResultMap" type="com.example.BACKEND.Entity.Employee">  
 <result column="USERID" jdbcType="VARCHAR" property="userid"/>  
 <result column="USERNAME" jdbcType="VARCHAR" property="username" />  
 <result column="PASSWORD" jdbcType="VARCHAR" property="password" />  
 </resultMap>  
 <select id="getAllEmployee" parameterType="map" resultType="com.example.BACKEND.Entity.Employee">  
 select t.\* from EMPLOYEE t  
 </select>  
 <select id="getEmployeeByUsername" parameterType="map" resultMap="EmployeeResultMap">  
 select t.\* from EMPLOYEE t where t.USERNAME=#{username} and t.PASSWORD=#{password}  
 </select>  
</mapper>

5.在启动类上添加Dao层扫描路径

@SpringBootApplication

@MapperScan("com.example.BACKEND.Dao")  
public class BackendApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(BackendApplication.class, args);  
 }  
}

6.localhost:8080/druid 监控界面

