Springboot框架

依赖的引入

<**parent**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>2.2.2.RELEASE</**version**>  
</**parent**>//必须要的  
<**dependencies**>  
 *<!--Web依赖-->* <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
 </**dependency**>//必须要的  
  
 *<!--单元测试配置-->* <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
  
 *<!--MySQL数据库配置-->* <**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 </**dependency**>  
  
 *<!--热部署-->* <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-devtools</**artifactId**>  
 <**scope**>runtime</**scope**>  
 <**optional**>true</**optional**>  
 </**dependency**>  
  
 *<!--Mybatis Plus依赖-->* <**dependency**>  
 <**groupId**>com.baomidou</**groupId**>  
 <**artifactId**>mybatis-plus-boot-starter</**artifactId**>  
 <**version**>3.2.0</**version**>  
 </**dependency**>  
</**dependencies**>  
  
*<!--打包相关依赖-->*<**build**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>

<**configuration**>  
 <**includeSystemScope**>true</**includeSystemScope**>*<!--需要这个才能将jar引入打包-->* </**configuration**>

</**plugin**>  
 </**plugins**>  
</**build**>

Jar包案例：

*<!--语音转写 SDK-->*<**dependency**>  
 <**groupId**>com.iflytek.msp.lfasr</**groupId**>  
 <**artifactId**>lfasr-sdk</**artifactId**>  
 <**version**>3.0.1</**version**>  
 <**scope**>system</**scope**>  
 <**systemPath**>${project.basedir}/src/main/resources/lib/lfasr-sdk-3.0.1.jar</**systemPath**>  
</**dependency**>

Yml配置

**//端口的设置**

**server:  
 port:** 8080

**//数据库配置**  
**spring:  
 datasource:  
 username:** muke  
 **password: "123456"  
 url:** jdbc:mysql://192.168.127.1:3306/xnsj?useUnicode=true&characterEncoding=utf8&serverTimezone=GMT%2B8  
 **driver-class-name:** com.mysql.jdbc.Driver

//全局主键生成策略

mybatis-plus:

# 全局主键策略

global-config:

db-config:

id-type: auto # 设置为数据库ID自增

以下的依赖都在开头

启动类：  
**import** org.mybatis.spring.annotation.MapperScan;  
**import** org.springframework.boot.SpringApplication;  
**import** org.springframework.boot.autoconfigure.SpringBootApplication;  
@SpringBootApplication  
@MapperScan(**"cn.kql.dao"**)  
**public class** MainApplication {  
 **public static void** main(String[] args) {  
 SpringApplication.*run*(MainApplication.**class**, args);  
 }  
}

Controller:实现客户端与服务器的通信及操作

例：开启启动类，浏览器输入localhost:8080/Hello/underHello  
**package** cn.kql.controller;  
  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.web.bind.annotation.GetMapping;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.ResponseBody;  
  
@Controller  
@RequestMapping(**"/Hello"**)*//加一级***public class** HelloController {  
  
 @ResponseBody  
 *//@RequestMapping("/hello")* @GetMapping(**"/underHello"**)*//次级* **public** String hello() {  
 **return "欢迎来到烟田智能灌溉系统-hh"**;  
 }  
  
}

Bean层：需要在yml中配置全局主键生成策略

**package** cn.kql.been;  
**import** com.baomidou.mybatisplus.annotation.TableId;  
**import** com.baomidou.mybatisplus.annotation.TableName;  
@TableName(**"usertable"**)//对应的数据库  
**public class** User {  
 @TableId//主键  
 Integer **id**;  
 String **user**;  
 String **password**;  
 String **yz**;

无参构造

有参构造

Get和set  
 toString  
}

Dao层:

Mapper 是与数据库实现连接

例:

@Repository  
**public interface** UserMapper **extends** BaseMapper<User> {  
}

增删改查语句：Mybatis Plus

<https://mp.baomidou.com/>

Eq:相等

Like:像

增加：**userMapper**.insert(user)

修改：update(user,**new** UpdateWrapper<User>().set(**"password"**,user.getPassword()).eq(**"id"**,1))

删除：**userMapper**.delete(**new** QueryWrapper<User>().eq(**"id"**,1))

查询：①全部：**userMapper**.selectList(**new** QueryWrapper<User>())

②单个：**userMapper**.selectOne(**new** QueryWrapper<User>().eq(**"user"**,user.getUser()).eq(**"password"**,user.getPassword()))

Service层:包括接口和实现接口

接口：

**package** cn.kql.service.impl;  
**import** cn.kql.been.User;  
**import** java.util.List;  
**public interface** UserServiceImpl {  
 List<User> getAll();  
}

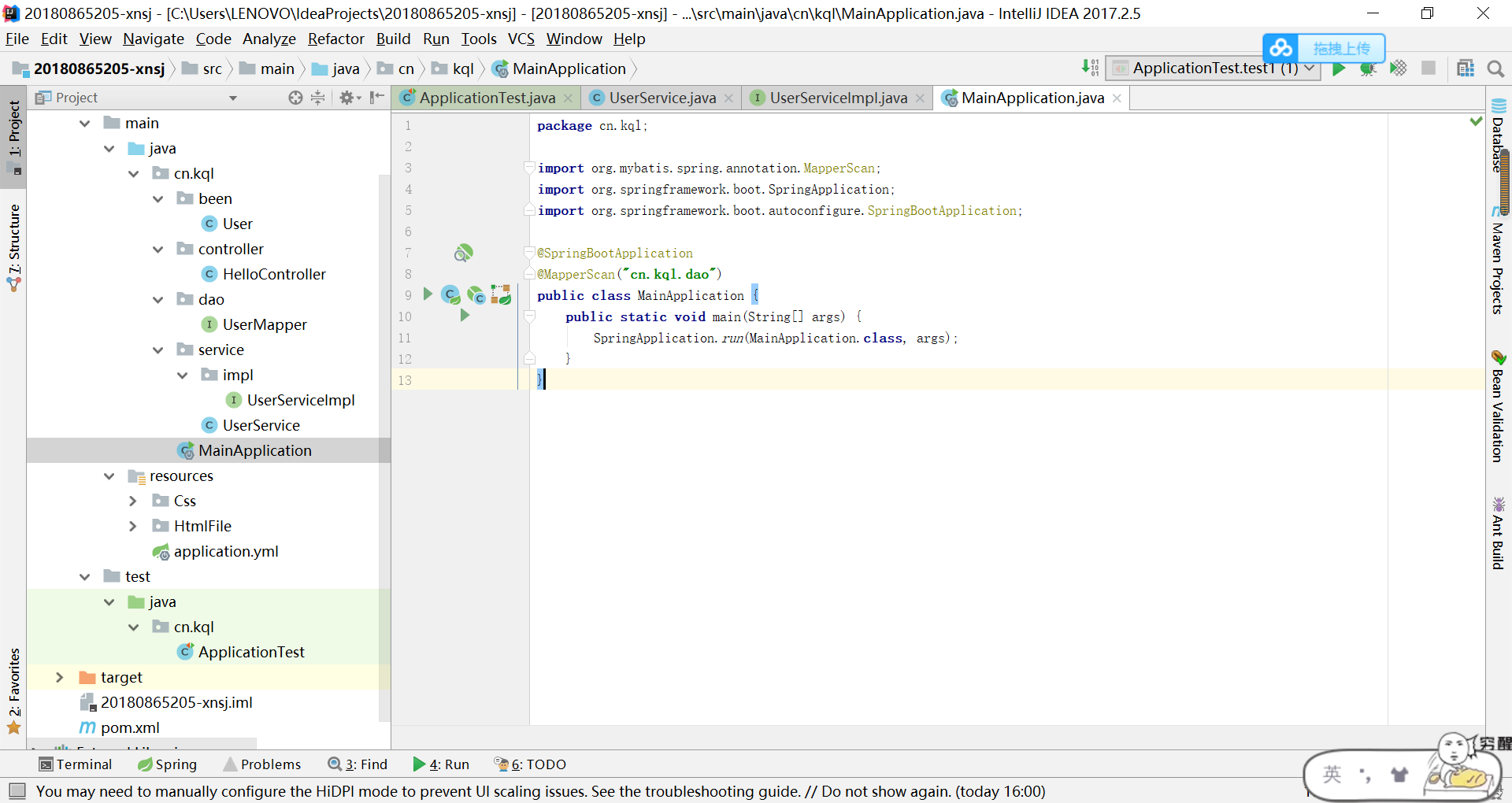
实现接口：  
**import** cn.kql.been.User;  
**import** cn.kql.service.impl.UserServiceImpl;  
**import** java.util.List;

@Repository  
**public class** UserService **implements** UserServiceImpl {  
 @Override  
 **public** List<User> getAll() {  
 **return null**;  
 }  
}

测试类：  
**import** org.junit.Test;  
**import** org.junit.runner.RunWith;  
**import** org.springframework.boot.test.context.SpringBootTest;  
**import** org.springframework.test.context.junit4.SpringRunner;  
  
@SpringBootTest  
@RunWith(SpringRunner.**class**)  
**public class** ApplicationTest {  
 @Test  
 **public void** test1(){  
 *System.out.println("欢迎来到烟田智能灌溉系统");* }

}

整体架构：



前端和后端打通：

解决跨域问题：

@Configuration

public class CorsConfiguration implements WebMvcConfigurer {

@Override

public void addCorsMappings(CorsRegistry registry) {

registry.addMapping("/\*\*")

.allowedOrigins("\*")

.allowCredentials(true)

.allowedMethods("GET", "POST", "DELETE", "PUT")

.maxAge(3600);

}

}

Get请求：

axios({  
 **method**: **'get'**,  
 **url**: **'http://localhost:8080/User/user'**,  
 **params**: { **username**: username,**pwd**: password},  
 }).then(**function** (response) {  
 **if**(response.**data**){  
 **location**.assign(**"main.html"**);  
 }**else**{  
 alert(**"你输入的账号或密码有误！"**);  
 }  
 **console**.log(response);  
 }).catch(**function** (error) {  
 **console**.log(error);  
});

Post请求：

axios({  
 **method**: **'post'**,  
 **url**: **'http://localhost:8080/User/user1'**,  
 **data**: {  
 **user**: username,  
 **password**: password  
 }  
}).then(**function** (response) {  
 **if**(response.**data**){  
 **location**.assign(**"main.html"**);  
 }**else**{  
 alert(**"你输入的账号或密码有误！"**);  
 }  
 **console**.log(response.**data**);  
}).catch(**function** (error) {  
 **console**.log(error);  
});

Post请求相比于Get请求更加的安全

线程

**new** Thread(() -> {  
 **try** {

} **catch** (Exception e) {  
 e.printStackTrace();  
 }  
}).start();

文件操作

File path = **new** File(ResourceUtils.*getURL*(**"classpath:"**).getPath());  
**if**(!path.exists()) {  
 path = **new** File(**""**);  
  
}  
File upload = **new** File(path.getAbsolutePath(),**"static/"**+fileName);  
**if** (!upload.exists()){  
 upload.mkdirs();  
}