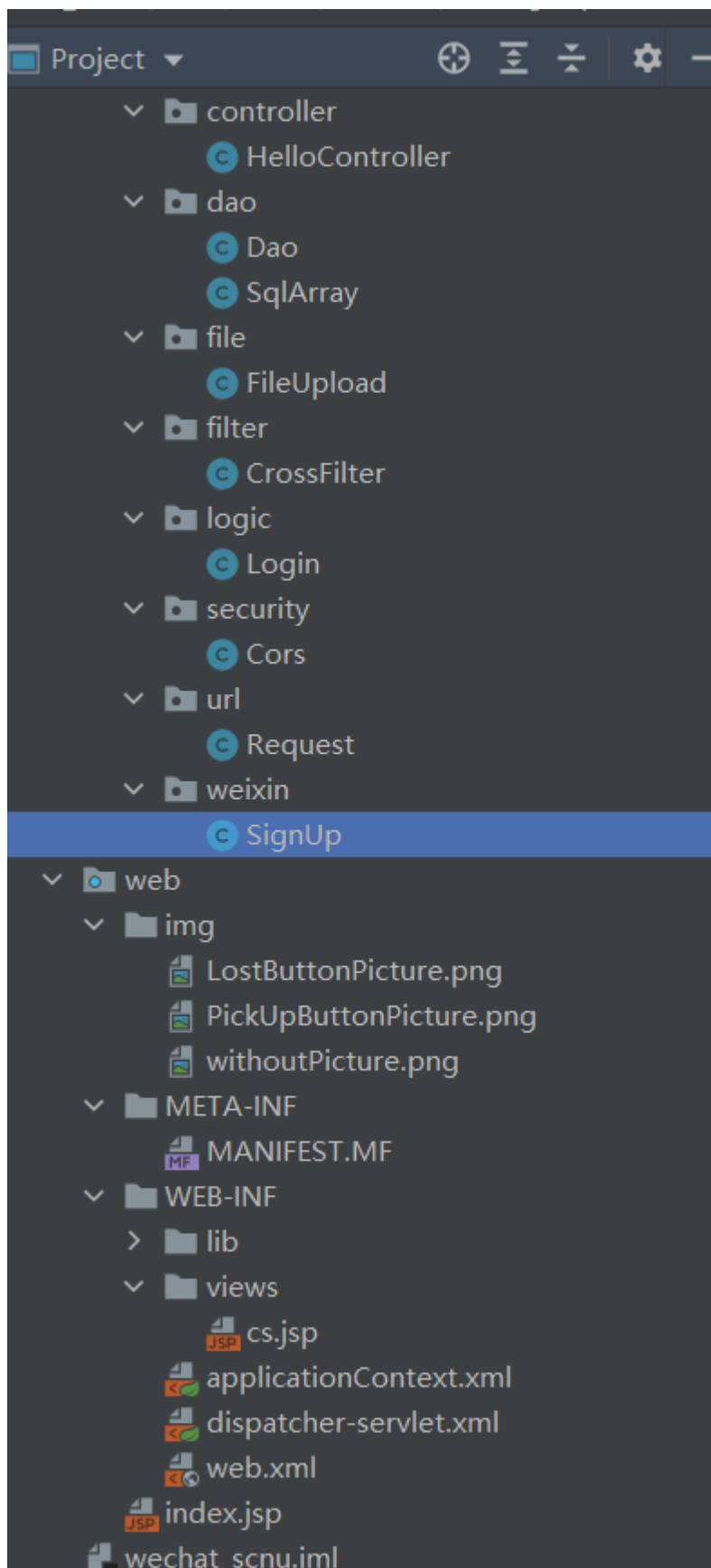
































(2022.4.15)

终版1.0

与小程序前端进行了测试，修改了部分错误
项目结构



包（一些可能没用上）

>  activation.jar
>  aopalliance-1.0.jar
>  commons-beanutils-1.4.jar
>  commons-collections-3.2.1-1.0.0.jar
>  commons-fileupload-1.4.jar
>  commons-io-2.0.jar
>  commons-io-2.11.0.jar
>  commons-lang-1_0.jar
>  commons-logging-1.1.1-1.0.0.jar
>  commons-logging-1.2.jar
>  ezmorph-0.8.1.jar
>  fastjson-1.1.30.jar
>  gson-1.4.jar
>  imap.jar
>  jackson-annotations-2.9.0.jar
>  jackson-core-2.9.0.jar
>  jackson-databind-2.9.0.jar
>  javax.annotation.jar
>  javax.ejb.jar
>  javax.jms.jar
>  javax.persistence.jar
>  javax.resource.jar
>  javax.servlet.jar
>  javax.servlet.jsp.jar
>  javax.servlet.jsp.jstl.jar
>  javax.servlet-api-4.0.1.jar
>  javax.transaction.jar
>  json-lib-2.1-jdk13.jar
>  jstl.jar
>  mail.jar

```
> json-lib-2.1-jdk13.jar
> jstl.jar
> mail.jar
> mysql-connector-java-8.0.25.jar
> smtp.jar
> spring-aop-5.2.3.RELEASE.jar
> spring-aspects-5.2.3.RELEASE.jar
> spring-beans-5.2.3.RELEASE.jar
> spring-context-5.2.3.RELEASE.jar
> spring-context-support-5.2.3.RELEASE.jar
> spring-core-5.2.3.RELEASE.jar
> spring-expression-5.2.3.RELEASE.jar
> spring-instrument-5.2.3.RELEASE.jar
> spring-jdbc-5.2.3.RELEASE.jar
> spring-jms-5.2.3.RELEASE.jar
> spring-messaging-5.2.3.RELEASE.jar
> spring-orm-5.2.3.RELEASE.jar
> spring-oxm-5.2.3.RELEASE.jar
> spring-test-5.2.3.RELEASE.jar
> spring-tx-5.2.3.RELEASE.jar
> spring-web-5.2.3.RELEASE.jar
> spring-webmvc-5.2.3.RELEASE.jar
> standard-2.2.0.jar
> taglibs-standard-impl-1.2.5.jar
> taglibs-standard-spec-1.2.5.jar
```

HelloController.java

```
package com.controller;

import com.dao.SqlArray;
import com.file.FileUpload;
import com.logic.Login;
import com.url.Request;
import com.weixin.SignUp;
import org.springframework.http.*;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
```

```

import com.google.gson.Gson;
//import net.sf.json.JSONObject;

import com.alibaba.fastjson.JSONObject;
import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.*;

//使用Controller来标识它是一个控制器
@Controller
//@CrossOrigin
@RequestMapping(value = "/api")           //使链接还有一个 /api/
public class HelloController {
    //全局变量
    public static String appid = "wx7d006a32416eba16";
    public static String secret = "bc9aea068d709089bf97c56cf7f3ea0f" ;

    //测试springmvc是否创建成功
    @RequestMapping(value = "/helloworld")
    public String Hello() {
        return "cs";           //返回测试页面
    }

    /* 测试链接
    * 请求: Get
    * 链接: 地址/api/cs
    * 参数: 无
    * 返回: json {"msg":"HelloWorld"}
    */
    @RequestMapping(value = "/cs")           //请求链接是cs
    public @ResponseBody Map<String,String> cs() {
        Map<String,String> data = new HashMap<String,String>();           //创建map
        格式的数据
        data.put("msg","HelloWorld" );
        return data;           //返回后会被前端解析为json格式的数据
    }

    /*
    * 获取两张图片
    * 请求: Get
    * 链接: 地址/api/getpicture

```

```

    * 参数: 无
    * 返回: json
    * {
    *     "msg":"ok",
    *     "picture1":"/img/LostButtonPicture.png",
    *     "picture2":"/img/PickUpButtonPicture.png"
    * }
    */
@RequestMapping(value = "/getpicture")
public @ResponseBody
Map<String,String> getPicture() {
    Map<String,String> data = new HashMap<String,String>();
    data.put("msg","ok" );
    //向前端返回两张图片的地址
    data.put("picture1","/img/LostButtonPicture.png");
    data.put("picture2","/img/PickUpButtonPicture.png");
    return data;
}

/* 注册
 * 请求: Get
 * 链接: 地址/api/register
 * 参数: 无
 * 返回: json {"msg":"HelloWorld"}
 */
// 没用到
/*@RequestMapping(value = "/register",method = RequestMethod.POST)
public @ResponseBody
Map<String,String> register(HttpServletRequest request) {
    String code = request.getParameter("code");
    Map<String,String> data = new HashMap<String,String>();
    data.put("msg","ok" );
    return data;
}*/

/* 获取手机号码
 * 请求: POST
 * 链接: 地址/api/getphone
 * 参数: 手机号码获取的code
 * 返回: json {"msg":"HelloWorld"}
 */
@RequestMapping(value = "/getphone",method = RequestMethod.POST)
public @ResponseBody Map<String,String> getPhone(HttpServletRequest
request) {
    Map<String,String> data = new HashMap<String,String>();
    try{
        //获取前端传过来的code
        String code = request.getParameter("code");

```

```

        //System.out.println(code);

        //获取获取小程序全局唯一后台接口调用凭据（access_token）
        String getTokenUrl = "https://api.weixin.qq.com/cgi-bin/token?
grant_type=client_credential&appid="+appid+"&secret="+secret;
        String jsonStringA = Request.doGet(getTokenUrl);           //调用get
请求去访问微信小程序自带的链接，将返回结果存储到jsonStringA中
        //.out.println(jsonStringA);

        //String转JSON
        JSONObject jsonObject = JSONObject.parseObject(jsonStringA);
        String access_token = jsonObject.getString("access_token");
//获取JSON数据中的access_token

        //提交参数
        String getPhoneUrl =
"https://api.weixin.qq.com/wxa/business/getuserphonenumber?
access_token="+access_token;

        Map<String, Object> map = new HashMap<String, Object>();
        //map.put("access_token", access_token);
        map.put("code", code);           //将code放到map中
        JSONObject json = new JSONObject(map);           //Map格式转化成JSON格式
        //System.out.println(map);
        //System.out.println(json);
        String jsonStringB = Request.doPostForm(getPhoneUrl, json);
//向微信小程序接口提交Post请求得到结果
        //System.out.println(jsonStringB);

        //String转JSON
        JSONObject jsonObject2 = JSONObject.parseObject(jsonStringB);
        HashMap hashMap =
JSONObject.parseObject(jsonObject2.toJSONString(), HashMap.class);
        if( 0 == (int)hashMap.get("errcode") ){           //请求成功
            data.put("msg", "ok" );
            JSONObject tmp2 = (JSONObject)hashMap.get("phone_info");
            data.put("phone", (String)tmp2.get("phoneNumber"));
//将结果存储下来
        }else{
            data.put("msg", "fail" );
            data.put("error", (int)hashMap.get("errcode")+"" );
        }
    }catch (Exception e){
        System.out.println(e);
        data.put("msg", "fail" );
    }

    return data;

```

```

}

/* 登录
 * 请求: POST
 * 链接: 地址/api/login
 * 参数: 登录获取的code
 * 返回: json {"skey":""}
 */
@RequestMapping(value = "/login",method = RequestMethod.POST)
public @ResponseBody Map<String,String> login(HttpServletRequest request)
{
    Map<String, String> data = new HashMap<String, String>();
    try {
        //获取前端传过来的code
        String code = request.getParameter("code");
        //Get请求（登录凭证校验）
        String getAuthUrl =
"https://api.weixin.qq.com/sns/jscode2session?appid=" + appid + "&secret=" +
secret + "&js_code=" + code + "&grant_type=authorization_code";
        String jsonString = Request.doGet(getAuthUrl);           //进行get
请求

        //System.out.println(jsonString);
        //String转JSON, 再json转为map
        JSONObject jsonObject = JSONObject.parseObject(jsonString);
        HashMap hashMap =
JSONObject.parseObject(jsonObject.toJSONString(), HashMap.class);

        //注意这里要加上 hashMap.get("errcode") == null
        if ( hashMap.get("errcode") == null || 0 == (int)
hashMap.get("errcode")) {           //请求成功
            data.put("msg", "ok");
            //得到openid和session_key去生成3rd_session
            //这个生成3rd_session的方式自己决定即可，比如使用SHA或Base64算法都可
以。例如：将session_key或openid+session_key作为SHA或Base64算法的输入，输出结果做为
3rd_session来使用，同时要将openid, session_key, 3rd_session三者关联存储到数据库中，
方便下次拿3rd_session获取session_key或openid做其他处理。
            String openid = (String) hashMap.get("openid");
            String session_key = (String) hashMap.get("session_key");

            //uuid生成唯一
key(https://blog.csdn.net/weixin_38169886/article/details/99820453?
utm_medium=distribute.pc_relevant.none-task-blog-
2~default~baidujs_baidulandingword~default-
5.pc_relevant_default&spm=1001.2101.3001.4242.4&utm_relevant_index=8)
            String skey = UUID.randomUUID().toString();           //用UUID来
生成唯一的skey

            //判断是否注册过

```



```

        boolean tmp = Login.checkUser(openid);
        if (tmp == false) { //没有注册过
            Login.register(openid, skey);
            data.put("msg", "ok");
            data.put("skey", skey);
        } else { //注册过，更新新的skey
            Login.updatekey(openid, skey);
            data.put("skey", skey);
        }
    } else {
        data.put("msg", "fail");
        data.put("error", (int) hashMap.get("errcode") + "");
    }
}
catch(Exception e){
    e.printStackTrace();
    data.put("msg", "fail");
    data.put("error", e.toString());
}
return data;
}

```

//文件上传

/*

* HTTP请求: POST

* 参数: file (可以是一个文件, 也可以是多个文件)

* 请求链接: 地址/api/upload

* {"msg":"ok","filename":""}

* {"msg":"fail","error":""}

* */

@RequestMapping(value = "/upload", method = RequestMethod.POST)

//@RequestParam(value = "file", required = false) MultipartFile[] file 注意

这里的写法, 参数名必须和前端提交上来的名字一致

```

public @ResponseBody Map<String,String> Upload(@RequestParam(value =
"file",required = false) MultipartFile[] file, HttpServletResponse response,
HttpServletRequest request) throws IOException {

```

```

    Map<String, String> data = new HashMap<String, String>();

```

```

    try {

```

```

        //System.out.println("总共有"+file.length+"个文件");

```

```

        // 文件上传到服务器的位置"/files"

```

```

        //System.out.println("正在上传文件");

```

```

        List<String> fileList = new ArrayList<>(); //全部的数组

```

```

        for( MultipartFile f:file){ //for each将文件数组一

```

个个取出

```

            String filelocation = FileUpload.SaveServer(f,request);

```

//将文件保存下来

```

            if( filelocation != null ){ //说明文件保存成功

```

```

        fileList.add(filelocation); //将文件位置加入到链表中
    }else{ //说明文件保存失败，直接向前
端返回错误信息

        data.put("msg","fail");
        data.put("error","文件上传失败");
        return data;
    }
}
//合并
String all_file = SqlArray.Merge(fileList); //将前面存储起来的文
件位置一个个连接起来，变成一个字符串（用|分割）
//System.out.println(all_file);
data.put("msg","ok");
data.put("filepath",all_file); //向前端返回成功的数据

}catch (Exception e){
    e.printStackTrace();
}
return data;
}

//失主提交页面
/*
HTTP请求：POST
请求链接：地址/api/ownerEnrol
参数： 7
category_value 失物类别
LostTime 丢失时间
LostPosition 丢失地点
LostPerson 联系人
LostTelNumber 联系方式
LostThingsPicture 失物照片
FilePath 图片位置
LostMessage 备注
返回JSON
提交成功
{
    "msg":"ok"
}
提交失败
{
    "msg":"fail",
    "error":错误原因
}
*/
//失主提交和拾主的函数基本一致，只有一些传数据的不同
@RequestMapping(value = "/ownerEnrol", method = RequestMethod.POST)
public @ResponseBody Map<String,String> Enrol(HttpServletResponse

```

```

response, HttpServletRequest request) throws IOException {
    Map<String, String> data = new HashMap<String, String>();
    try {
        //获取前端传过来的数据
        String category_value = request.getParameter("category_value");
        String LostTime = request.getParameter("Time");
        String LostPosition = request.getParameter("Position");
        String LostPerson = request.getParameter("Person");
        String LostTelNumber = request.getParameter("TelNumber");
        String LostMessage = request.getParameter("Message");
        String FilePath = request.getParameter("FilePath");

        //如果传文件位置为空的话,说明没有图片,文件位置变成默认“无图片”的图片
        if( FilePath == null || "".equals(FilePath)){
            FilePath = "\\img\\withoutPicture.png";
        }

        //判断关键信息是否存储,若不存在,则直接返回
        if ( category_value == null
            || LostTime == null || LostTelNumber == null ||
LostMessage == null
            || "".equals(category_value) || "".equals(LostTime) ||
"".equals(LostTelNumber) || "".equals(LostMessage) ) {
            data.put("msg", "fail");
            data.put("error", "失物类型,失物丢失时间,失物情况,联系方式为必填项且
不能为空");
            throw new NullPointerException("xx");           //抛出异常,方便
直接跳到最后一步
        }

        //进行登记,写入数据库
        if
(SignUp.Enrol(category_value,LostTime,LostPosition,LostPerson,LostTelNumber,F
ilePath,LostMessage,"owner")){
            data.put("msg", "ok");
        }else{
            data.put("msg","fail");
            data.put("error","写入数据库失败");
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    return data;
}

//拾主提交页面
/*

```

HTTP请求: POST

请求链接: 地址/api/pickerEnrol

参数: 7

category_value	失物类别
LostTime	丢失时间
LostPosition	丢失地点
LostPerson	联系人
LostTelNumber	联系方式
FilePath	图片位置
LostMessage	备注

返回JSON

提交成功

```
{
    "msg": "ok"
}
```

提交失败

```
{
    "msg": "fail",
    "error": "错误原因"
}
```

*/

//拾主

```
@RequestMapping(value = "/pickerEnrol", method = RequestMethod.POST)
public @ResponseBody Map<String,String> pickerEnrol(HttpServletRequestResponse
response, HttpServletRequest request) throws IOException {
    Map<String, String> data = new HashMap<String, String>();
    try {
        String category_value = request.getParameter("category_value");
        String LostTime = request.getParameter("Time");
        String LostPosition = request.getParameter("Position");
        String LostPerson = request.getParameter("Person");
        String LostTelNumber = request.getParameter("TelNumber");
        String LostMessage = request.getParameter("Message");
        String FilePath = request.getParameter("FilePath");

        if( FilePath == null || "".equals(FilePath)){
            FilePath = "\\img\\withoutPicture.png";
        }

        if ( category_value == null
            || LostTime == null || LostTelNumber == null ||
LostMessage == null
            || "".equals(category_value) || "".equals(LostTime) ||
"".equals(LostTelNumber) || "".equals(LostMessage) ) {
            data.put("msg", "fail");
            data.put("error", "失物类型,失物丢失时间,失物情况,联系方式为必填项且
不能为空");
            throw new NullPointerException("xx"); //抛出异常
```

```

    }

    //进行登记（与失主的区别是，这里传参的最后一个参数是“picker”）
    if
(SignUp.Enrol(category_value, LostTime, LostPosition, LostPerson, LostTelNumber, F
ilePath, LostMessage, "picker")) {
        data.put("msg", "ok");
    }else{
        data.put("msg", "fail");
        data.put("error", "写入数据库失败");
    }
}catch (Exception e){
    e.printStackTrace();
}
return data;
}

//显示失物数据
/*
    HTTP请求: POST
    请求链接: 地址/api/ownershowdata
    参数:      page(字符串)
    返回JSON
    获取成功
    {
        "msg": "ok",
        "num": "总数量"
        "data": {
            (数组类型的数据)
        }
    }
    获取失败
    {
        "msg": "fail",
        "num": "总数量",
        "error": "错误原因"
    }
*/
@RequestMapping(value = "/ownershowdata", method = RequestMethod.POST)
//嵌套的JSON格式可以用Map<String, Object>, 不过需要相关的包, 可能是gson吧
public @ResponseBody Map<String, Object> ownershowData(HttpServletRequest response,
    HttpServletRequest request) throws IOException {
    Map<String, Object> data = new HashMap<String, Object>();
    try {
        int num = SignUp.queryNum("owner"); //返回失物总数量
        String page = request.getParameter("page"); //得到前端请求
        的页数
        List list = SignUp.signupList(Integer.parseInt(page), 10, "owner");
    }
}

```

```

//返回数据， 每页限制10条数据          //每页显示10条数据
    data.put("num",""+num);              //将数据加入map中
    if( list == null ){                  //如果为空，说明没有获取到数据
        data.put("msg","fail");
        data.put("error","当前页面已经无数据");
    }else{
        data.put("msg","ok");
        data.put("data",list);
    }
} catch (Exception e){
    e.printStackTrace();
}
return data;
}

//显示失物搜索数据
/*
    HTTP请求： POST
    请求链接： 地址/api/ownersearch
    参数：
        page          页数          必填
        search         必填
        category_value 非必填
        begintime       非必填
        endtime         非必填
    返回JSON
    获取成功
    {
        "msg":"ok",
        "data":{
            (数组类型的数据)
        }
    }
    获取失败
    {
        "msg":"fail",
        "error":错误原因
    }
*/
@RequestMapping(value = "/ownersearch", method = RequestMethod.POST)
public @ResponseBody Map<String,Object> ownersearch(HttpServletRequest response,
    HttpServletRequest request) throws IOException {
    Map<String, Object> data = new HashMap<String, Object>();
    try {
        int num = SignUp.queryNum("owner");          //获取数量
        //获取前端得到的值，注意存在一些值为空的情况
        String page = request.getParameter("page");
        String search = request.getParameter("search");

```

```

String category_value = request.getParameter("category_value");
String begintime = request.getParameter("begintime");
String endtime = request.getParameter("endtime");
if( page == null || "".equals(page) ){           //必填项判定
    data.put("msg","fail");
    data.put("error","page为必填项");
    return data;
}
//返回搜索的结果
List list =
SignUp.searchList(Integer.parseInt(page),10,category_value,begintime,endtime,
search,"owner");           //每页显示10条数据

    if( list == null ){                               //同上
        data.put("msg","fail");
        data.put("error","当前页面已经无数据");
    }else{
        data.put("msg","ok");
        System.out.println(list.toString());
        data.put("data",list);
    }
} catch (Exception e){
    e.printStackTrace();
}
return data;
}
//显示拾物数据
/*
HTTP请求: POST
请求链接: 地址/api/pickershowdata
参数:      page(字符串)
返回JSON
    获取成功
    {
        "msg":"ok",
        "num":“总数量”
        "data":{
            (数组类型的数据)
        }
    }
    获取失败
    {
        "msg":"fail",
        "num":“总数量”,
        "error":错误原因
    }
*/
//与失物函数基本一致

```

```

@RequestMapping(value = "/pickershowdata", method = RequestMethod.POST)
public @ResponseBody Map<String, Object>
pickershowData(HttpServletResponse response, HttpServletRequest request)
throws IOException {
    Map<String, Object> data = new HashMap<String, Object>();
    try {
        int num = SignUp.queryNum("picker");
        String page = request.getParameter("page");
        List list =
SignUp.signupList(Integer.parseInt(page), 10, "picker"); //每页显示10条数据

        data.put("num", ""+num);
        if( list == null ){
            data.put("msg", "fail");
            data.put("error", "当前页面已经无数据");
        }else{
            data.put("msg", "ok");
            data.put("data", list);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return data;
}

```

//显示拾物搜索数据

/*

HTTP请求: POST

请求链接: 地址/api/pickersearch

参数:

page	页数	必填
search		必填
category_value		非必填
begintime		非必填
endtime		非必填

返回JSON

获取成功

```

{
    "msg": "ok",
    "data": {
        (数组类型的数据)
    }
}

```

}

获取失败

```

{
    "msg": "fail",
    "error": 错误原因
}

```



```

*/
//与失物函数基本一致
@RequestMapping(value = "/pickersearch", method = RequestMethod.POST)
public @ResponseBody Map<String, Object> pickersearch(HttpServletRequest response,
    HttpServletRequest request) throws IOException {
    Map<String, Object> data = new HashMap<String, Object>();
    try {
        int num = SignUp.queryNum("picker");
        String page = request.getParameter("page");
        String search = request.getParameter("search");
        String category_value = request.getParameter("category_value");
        String begintime = request.getParameter("begintime");
        String endtime = request.getParameter("endtime");
        if( page == null || "".equals(page) ){
            data.put("msg", "fail");
            data.put("error", "page为必填项");
            return data;
        }
        List list =
SignUp.searchList(Integer.parseInt(page), 10, category_value, begintime, endtime,
search, "picker");          //每页显示10条数据

        if( list == null ){
            data.put("msg", "fail");
            data.put("error", "当前页面已经无数据");
        }else{
            data.put("msg", "ok");
            System.out.println(list.toString());
            data.put("data", list);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return data;
}
}

```

Dao.java

```

package com.dao;

import java.sql.*;

//作为辅助连接数据库的工具
public class Dao {
    // 获取数据库连接

```

```

//scnu_wechat      scnu_wechat      YJ8DhK5miHaYsx44
public static Connection getConnection(){
    Connection conn = null;
    //注意这里后面加参数避免中文乱码，而且在web.xml加部分内容避免乱码
    String url = "jdbc:mysql://****:3306/****?
useUnicode=true&characterEncoding=utf8";
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        //Class.forName("com.mysql.jdbc.Driver");
        //数据库名字和密码在这里改!!!!
        conn = DriverManager.getConnection(url, "****", "****");
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
        System.out.println("数据库驱动加载出错");
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("数据库出错");
    }
    return conn;
}
//关闭相关通道
public static void close(ResultSet rs,PreparedStatement p,Connection
conn)
{
    try
    {
        if(!rs.isClosed()){
            rs.close();
        }
        if(!p.isClosed()){
            p.close();
        }
        if(!conn.isClosed()){
            conn.close();
        }
    }
    catch(SQLException e)
    {
        e.printStackTrace();
        System.out.println("数据关闭出错");
    }
}

//关闭相关通道
public static void close(PreparedStatement p,Connection conn)
{
    try
    {

```

```

        if(!p.isClosed()){
            p.close();
        }
        if(!conn.isClosed()){
            conn.close();
        }
    }
    catch(SQLException e)
    {
        e.printStackTrace();
        System.out.println("数据关闭出错");
    }
}
}

```

SqlArray.java

```

package com.dao;

import java.util.ArrayList;
import java.util.List;

public class SqlArray {
    /*
    * 存储数组进数据库
    * 数据库用一个字段表示数组，字段类型为文本类型。
    * 程序存入数组到数据库的时候，利用join方法把数组转换为分隔符分隔的字符串，比如你的例子数组a[1]="第一步";a[2]="第二步";合并后为"第一步|第二步"，把这个合并后的字符串存入数据库你是会的。
    * 从数据库里面取出合并后的字符串"第一步|第二步"以后，利用split方法可以转换为数组。
    * 这个方法的最大的优点是可以保存个数不确定的数组，程序编写相当简单。
    * */

    //将数组中的内容合并成一个字符串，并用|分割
    public static String Merge(String[] nameArr){
        String nameAll = "";
        for( int i = 0 ; i < nameArr.length ; i++){
            nameAll += (nameArr[i] + "|");
            if( i == nameArr.length - 1){
                nameAll += nameArr[i];
            }
        }
        return nameAll;
    }
}

```

```

//将链表中的内容合并成一个字符串，并用|分割
public static String Merge(List nameArr){
    if( nameArr.size() == 1 )    return String.valueOf(nameArr.get(0));
    String nameAll = "";
    for( int i = 0 ; i < nameArr.size() ; i++ ){
        nameAll += String.valueOf((nameArr.get(i) + "|"));
        if( i == nameArr.size() - 1){
            nameAll += String.valueOf(nameArr.get(i));
        }
    }
    return nameAll;
}

//根据合并的规则进行分离，分离成数组形式
public static String[] SeparationArray(String nameAll){
    String[] nameArr = nameAll.split("\\|");           //split分割
    return nameArr;
}

//根据合并的规则进行分离，分离成链表形式
public static List SeparationList(String nameAll){
    String[] nameArr = nameAll.split("\\|");           //分割成数组
    //再一个个存储到链表中
    List list = new ArrayList();
    for(String name:nameArr){
        list.add(name);
    }
    return list;
}
}

```

FileUpload.java

```

package com.file;

import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;
import java.io.File;
import java.util.UUID;

//文件上传相关的工具包
public class FileUpload {

```

```

// UUID可以避免重命名
// 传入原始文件名可以得到  uuid+"_"+文件的原始名称 的返回值
public static String getUuidFileName(String fileName){
    //文件名以: uuid+"_"+文件的原始名称
    return UUID.randomUUID().toString()+ "_" +fileName;
}

//保存到服务器, 返回存储的位置
public static String SaveServer(MultipartFile file, HttpServletRequest
request){
    System.out.println("正在上传文件");
    //得到上传文件的保存目录, 将上传的文件存放于WEB-INF目录下, 不允许外界直接访问,
    保证上传文件的安全
    //String realpath = request.getServletContext().getRealPath("/WEB-
INF/files");
    //暂时就不保存在WEB-INF那里, 先放在files
    //这里保存在根目录中的file文件中
    String realpath = request.getServletContext().getRealPath("files");

    //获取文件名字, 进行UUID重命名
    String fileName = getUuidFileName(file.getOriginalFilename());

    //文件上传
    File targetFile = new File(realpath, fileName);

    //如果不存在, 创建文件
    if (!targetFile.exists()) {
        targetFile.mkdirs();
    }
    // 上传
    try {
        file.transferTo(targetFile);                //保存下来
        System.out.println("上传成功");
        //return realpath + '\\ ' + fileName;
        //返回相对路径就行
        return "\\files\\" + fileName;
    } catch (Exception e) {
        e.printStackTrace();
        return null;                                //说明保存不成功
    }
}
}

```

```

package com.filter;

import com.alibaba.fastjson.JSONObject;
import com.logic.Login;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

// https://blog.csdn.net/qq_32363305/article/details/82469451
// 过滤器，可以保证每个部分请求提交必须是已经登陆的状态
// 不过这里加在头部貌似没用，后期可以考虑参数直接传递
public class CrossFilter implements Filter{
    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain
chain)
        throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;
        response.addHeader("Access-Control-Allow-Origin",
request.getHeader("Origin"));
        response.addHeader("Access-Control-Allow-Methods", "POST, GET,
OPTIONS, PUT, DELETE, HEAD");
        response.addHeader("Access-Control-Max-Age", "27000000");
        response.addHeader("Access-Control-Allow-Headers", "X-PINGOTHER,
Origin, X-Requested-With, Content-Type, Accept, Cookie");
        response.addHeader("Access-Control-Allow-Credentials", "true");
        //获取请求时头部的sessionId
        String skey = request.getHeader("skey");
        if( skey == null || "".equals(skey) ){           //如果没有说明不需要登录的权
限

            //该请求不需要验证session,直接通过
            System.out.println("该请求不需要过滤，通过");
            chain.doFilter(request,response);
            return;
        }else{
            //只有在缓存中存在该sessionId才能进行请求
            if ( Login.checkskey(skey) == false ) {
                // 登录信息已过期，请重新登录
            }
        }
    }
}

```

```

        System.out.println("登录信息失效，请重新登录");
        //response.getWriter().write("登录信息失效，请重新登录");
        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/json; charset=utf-8");
        PrintWriter out = response.getWriter();
        JSONObject tmp = new JSONObject();
        tmp.put("msg", "fail");
        tmp.put("error", "登录信息失效，请重新登录");
        out.append(tmp.toString());
        return;
    }
    System.out.println("session验证成功");
    chain.doFilter(request, response);
}
}

@Override
public void init(FilterConfig filterConfig) throws ServletException {
    //system.out.println("开始初始化");
}

@Override
public void destroy() {
    // system.out.println("销毁完成");
}
}

```

Login.java

```

package com.logic;

import com.dao.Dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Login {
    //检测是否有该账户,有则返回真，没有则返回假
    public static boolean checkUser(String username) {
        try {
            Connection conn = Dao.getConnection(); //连接数据库
            //构造sql语句
            PreparedStatement p = conn.prepareStatement("select * from
user_data where open_id=?"); //查询

```

```

        //将? 替代成具体的值
        p.setString(1, username);
        //执行sql语句
        ResultSet rs = p.executeQuery();
        //读取sql的值
        if(rs.next()){
            String user_name = rs.getString("open_id");
            Dao.close(rs, p, conn);
            return true;
        }
        Dao.close(rs, p, conn);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

//检测是否有该账户,有则返回真, 没有则返回假
public static boolean checkskey(String skey) {
    try {
        Connection conn = Dao.getConnection(); //连接数据库
        PreparedStatement p = conn.prepareStatement("select * from
user_data where skey=?;"); //查询
        p.setString(1, skey);
        ResultSet rs = p.executeQuery();
        if(rs.next()){
            String user_name = rs.getString("skey");
            Dao.close(rs, p, conn);
            return true;
        }
        Dao.close(rs, p, conn);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

//注册
public static boolean register(String open_id,String skey){
    //ticket就是准考证号, 准考证号就是登录名
    if(Login.checkUser(open_id) == false ){ //已经注册的就不能再注册

        try {
            Connection conn = Dao.getConnection();
            /* 对账户密码的数据库进行补充 */

```



```

        PreparedStatement p = conn.prepareStatement("insert into
user_data(open_id,skey) VALUES (?,?);");
        //对占位符进行补充
        p.setString(1, open_id);
        p.setString(2, skey);
        p.executeUpdate();
        System.out.println("注册成功");
        Dao.close(p, conn);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    }
} else {
    System.out.println("该账户已经注册过, 请不要重复注册!");
    return false;
}
return false;
}

//更新skey
public static boolean updateskey(String open_id, String newskey) {
    try {
        Connection conn = Dao.getConnection(); //连接数据库
        //
        PreparedStatement p = conn.prepareStatement("update user_data set
skey=? where open_id=?;");
        p.setString(1, newskey);
        p.setString(2, open_id);
        p.executeUpdate(); //执行SQL语句(注意这里是execute, 因为进
行的是修改操作)
        Dao.close(p, conn);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

}

```

Cors.java

```

package com.security;

import org.springframework.context.annotation.Bean;

```

```

import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

//避免跨域问题，本项目暂时没用到
public class Cors {
    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurerAdapter() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/**")
                    // 设置允许跨域请求的域名
                    // 是否允许证书 配置是否允许发送Cookie，用于凭证请求， 默认
                    // 不发送cookie。
                    .allowCredentials(true)
                    // 设置允许的方法
                    .allowedMethods("GET", "POST", "DELETE", "PUT")
                    .allowedHeaders("Content-Type", "X-Requested-With",
"accept", "Origin", "Access-Control-Request-Method",
                    "Access-Control-Request-Headers")
                    .exposedHeaders("Access-Control-Allow-Origin",
"Access-Control-Allow-Credentials")
                    // 设置允许的header属性
                    .allowedHeaders("*")
                    //.allowedOrigins("*")
                    // 跨域允许时间
                    .maxAge(3600);
            }
        };
    }
}

```

Request.java

```

package com.url;

import com.alibaba.fastjson.JSONObject;
import com.google.gson.Gson;
import com.google.gson.Gson;
import java.io.*;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

```

```

import java.net.URLConnection;
import java.text.ParseException;
import java.util.*;

//https://www.cnblogs.com/mufengforward/p/10510337.html

public class Request {
    /**
     * 向指定URL发送GET方法的请求
     *
     * @param httpurl
     *      请求参数用?拼接在url后边，请求参数应该是
name1=value1&name2=value2 的形式。
     * @return result 所代表远程资源的响应结果
     */
    //发送get请求
    public static String doGet(String httpurl) {
        HttpURLConnection connection = null;
        InputStream is = null;
        BufferedReader br = null;
        String result = null;// 返回结果字符串
        try {
            // 创建远程url连接对象
            URL url = new URL(httpurl);
            // 通过远程url连接对象打开一个连接，强转成URLConnection类
            connection = (HttpURLConnection) url.openConnection();
            // 设置连接方式: get
            connection.setRequestMethod("GET");
            // 设置连接主机服务器的超时时间: 15000毫秒
            connection.setConnectTimeout(15000);
            // 设置读取远程返回的数据时间: 60000毫秒
            connection.setReadTimeout(60000);
            // 发送请求
            connection.connect();
            // 通过connection连接，获取输入流
            if (connection.getResponseCode() == 200) {
                is = connection.getInputStream();
                // 封装输入流is，并指定字符集
                br = new BufferedReader(new InputStreamReader(is, "UTF-8"));
                // 存放数据
                StringBuffer sbf = new StringBuffer();
                String temp = null;
                while ((temp = br.readLine()) != null) {
                    sbf.append(temp);
                    sbf.append("\r\n");
                }
                result = sbf.toString();
            }
        }
    }
}

```

```

    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        // 关闭资源
        if (null != br) {
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        if (null != is) {
            try {
                is.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }

        connection.disconnect();// 关闭远程连接
    }

    return result;
}

/**
 *
 * @param httpUrl 请求的url
 * @param param form表单的参数 (key,value形式)
 * @return
 */
//发送post请求, 注意请求微信小程序接口的格式是"Content-Type":"application/json"
public static String doPostForm(String httpUrl, Map param) {

    HttpURLConnection connection = null;
    InputStream is = null;
    OutputStream os = null;
    BufferedReader br = null;
    String result = null;
    try {
        URL url = new URL(httpUrl);
        // 通过远程url连接对象打开连接
        connection = (HttpURLConnection) url.openConnection();
        // 设置连接请求方式
        connection.setRequestMethod("POST");
    }

```

```

// 设置连接主机服务器超时时间: 15000毫秒
connection.setConnectTimeout(15000);
// 设置读取主机服务器返回数据超时时间: 60000毫秒
connection.setReadTimeout(60000);

// 默认值为: false, 当向远程服务器传送数据/写数据时, 需要设置为true
connection.setDoOutput(true);
// 默认值为: true, 当前向远程服务读取数据时, 设置为true, 该参数可有可无
connection.setDoInput(true);
// 设置传入参数的格式:请求参数应该是 name1=value1&name2=value2 的形式。
connection.setRequestProperty("Content-Type",
"application/json");
// 设置鉴权信息: Authorization: Bearer da3efcbf-0845-4fe3-8aba-
ee040be542c0
//connection.setRequestProperty("Authorization", "Bearer
da3efcbf-0845-4fe3-8aba-ee040be542c0");
// 通过连接对象获取一个输出流
os = connection.getOutputStream();
// 通过输出流对象将参数写出去/传出去,它是通过字节数组写出的(form表单形式
的参数实质也是key,value值的拼接,类似于get请求参数的拼接)
JSONObject json = new JSONObject(param);
os.write( createLinkString(param).getBytes() );
// 通过连接对象获取一个输入流, 向远程读取
if (connection.getResponseCode() == 200) {

    is = connection.getInputStream();
    // 对输入流对象进行包装:charset根据工作项目组的要求来设置
    br = new BufferedReader(new InputStreamReader(is, "UTF-8"));

    StringBuffer sbf = new StringBuffer();
    String temp = null;
    // 循环遍历一行一行读取数据
    while ((temp = br.readLine()) != null) {
        sbf.append(temp);
        sbf.append("\r\n");
    }
    result = sbf.toString();
}
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // 关闭资源
    if (null != br) {
        try {
            br.close();
        } catch (IOException e) {

```

```

        e.printStackTrace();
    }
}
if (null != os) {
    try {
        os.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
if (null != is) {
    try {
        is.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
// 断开与远程地址url的连接
connection.disconnect();
}
return result;
}

```

//发送"application/json"格式的POST请求

```

public static String doPostForm(String httpUrl, JSONObject param) {

    HttpURLConnection connection = null;
    InputStream is = null;
    OutputStream os = null;
    BufferedReader br = null;
    String result = null;
    try {
        URL url = new URL(httpUrl);
        // 通过远程url连接对象打开连接
        connection = (HttpURLConnection) url.openConnection();
        // 设置连接请求方式
        connection.setRequestMethod("POST");
        // 设置连接主机服务器超时时间: 15000毫秒
        connection.setConnectTimeout(15000);
        // 设置读取主机服务器返回数据超时时间: 60000毫秒
        connection.setReadTimeout(60000);

        // 默认值为: false, 当向远程服务器传送数据/写数据时, 需要设置为true
        connection.setDoOutput(true);
        // 默认值为: true, 当前向远程服务读取数据时, 设置为true, 该参数可有可无
        connection.setDoInput(true);
        // 设置传入参数的格式:请求参数应该是 name1=value1&name2=value2 的形式。
        connection.setRequestProperty("Content-Type",

```

```

"application/json");
    // 设置鉴权信息: Authorization: Bearer da3efcbf-0845-4fe3-8aba-
ee040be542c0
    //connection.setRequestProperty("Authorization", "Bearer
da3efcbf-0845-4fe3-8aba-ee040be542c0");
    // 通过连接对象获取一个输出流
    os = connection.getOutputStream();
    // 通过输出流对象将参数写出去/传出去,它是通过字节数组写出的(form表单形式
的参数实质也是key,value值的拼接,类似于get请求参数的拼接)

    os.write( param.toString().getBytes() );
    // 通过连接对象获取一个输入流,向远程读取
    if (connection.getResponseCode() == 200) {

        is = connection.getInputStream();
        // 对输入流对象进行包装:charset根据工作项目组的要求来设置
        br = new BufferedReader(new InputStreamReader(is, "UTF-8"));

        StringBuffer sbf = new StringBuffer();
        String temp = null;
        // 循环遍历一行一行读取数据
        while ((temp = br.readLine()) != null) {
            sbf.append(temp);
            sbf.append("\r\n");
        }
        result = sbf.toString();
    }
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // 关闭资源
    if (null != br) {
        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != os) {
        try {
            os.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != is) {

```

```

        try {
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // 断开与远程地址url的连接
    connection.disconnect();
}
return result;
}

/**
 * 把数组所有元素排序，并按照“参数=参数值”的模式用“&”字符拼接成字符串
 * @param params 需要排序并参与字符拼接的参数组
 * @return 拼接后字符串
 */
public static String createLinkString(Map<String, String> params) {

    List<String> keys = new ArrayList<String>(params.keySet());
    Collections.sort(keys);

    StringBuilder prestr = new StringBuilder();
    for (int i = 0; i < keys.size(); i++) {
        String key = keys.get(i);
        String value = params.get(key);
        if (i == keys.size() - 1) { // 拼接时，不包括最后一个&字符
            prestr.append(key).append("=").append(value);
        } else {
            prestr.append(key).append("=").append(value).append("&");
        }
    }

    return prestr.toString();
}

//String 转 Map
public static Map<String, Object> strToJson(String jsonString) {
    Gson gson = new Gson();
    Map<String, Object> map = new HashMap<String, Object>();
    map = gson.fromJson(jsonString, map.getClass());
    return map;
}

```



```
}
```

SignUp.java

```
package com.weixin;

import com.dao.Dao;
import com.dao.SqlArray;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.*;

public class SignUp {

    /*
        category_value      失物类别
        LostTime            丢失时间
        LostPosition        丢失地点
        LostPerson          联系人
        LostTelNumber       联系方式
        LostThingsPicture   失物照片
        LostMessage         备注
    */
    //create table data_enrol(category_value char(50),LostTime
    char(100),LostPosition char(100),LostPerson char(50),LostTelNumber
    char(50),LostThingsPicture longtext,LostMessage longtext,model char(10));
    //进行报名
    public static boolean Enrol(String category_value, String LostTime,
    String LostPosition, String LostPerson, String LostTelNumber, String
    LostThingsPicture,String LostMessage,String model) {
        //ticket就是准考证号，准考证号就是登录名
        try {
            Connection conn = Dao.getConnection();           //连接数据库
            /* 对报名的数据库进行补充 */
            //sql语句预编译
            PreparedStatement p = conn.prepareStatement("insert into
            data_enrol(category_value,LostTime,LostPosition,LostPerson,LostTelNumber,Lost
            ThingsPicture,LostMessage,model) VALUES (?,?,?,?,?,?,?,?,?);");
            //对占位符进行补充
            p.setString(1, category_value);
            p.setString(2, LostTime);
            p.setString(3, LostPosition);
```

```

        p.setString(4, LostPerson);
        p.setString(5, LostTelNumber);
        p.setString(6, LostThingsPicture);
        p.setString(7, LostMessage);
        p.setString(8, model);
        System.out.println(p);
        p.executeUpdate();           //运行sql语句
        System.out.println("登记成功");
        Dao.close(p, conn);          //关闭数据库连接
        return true;
    } catch (Exception e) {
        System.out.println("登记失败");
        e.printStackTrace();
        return false;
    }
}

```

//查询总的数目

```

public static int queryNum(String which){
    int res = 0;
    try {
        Connection conn = Dao.getConnection();           //连接数据库
        PreparedStatement p = conn.prepareStatement("select count(*) from
data_enrol where model=?;");           //查看报名的人数
        p.setString(1,which);
        ResultSet rs = p.executeQuery();
        if(rs.next()){
            res = rs.getInt(1);
            Dao.close(rs, p, conn);
            return res;
        }
        Dao.close(rs, p, conn);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return res;
}

```

//返回全部报名人及信息

```

public static List signupList(String which){
    int sign = 0;           //标记一下当前页数是否有数据
    //参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
    List list = new ArrayList();
    int res = 0;
    try {
        Connection conn = Dao.getConnection();           //连接数据库
        PreparedStatement p = conn.prepareStatement("select * from

```

```

data_enrol where model=?;");          //查看报名信息
    p.setString(1,which);
    ResultSet rs = p.executeQuery();

    while(rs.next()){                  //不断获取读到的数据
        sign = 1;
        //category_value char(50),LostTime char(100),LostPosition
char(100),LostPerson char(50),LostTelNumber char(50),LostThingsPicture
longtext,LostMessage longtext,model char(10));
        Map<String,String> data = new HashMap();
        //获取到对应列的数据保存的map中
        data.put("category_value", rs.getString("category_value"));
        data.put("LostTime", rs.getString("LostTime"));
        data.put("LostPosition", rs.getString("LostPosition"));
        data.put("LostPerson", rs.getString("LostPerson"));
        data.put("LostTelNumber", rs.getString("LostTelNumber"));

        String s = rs.getString("LostThingsPicture");
        if( s == null ){
            data.put("LostThingsPicture",null);
            data.put("FirstPicture",null);
        }else{                        //返回一个全部图片的地址和第一个图片的地址
            s = s.replaceAll("\\\\", "/");
            data.put("LostThingsPicture",s);
            String[] tmpa = SqlArray SeparationArray(s);
            data.put("FirstPicture",tmpa[0]);
        }

        data.put("LostMessage", rs.getString("LostMessage"));
        //返回类型
        if( "picker".equals(rs.getString("model")) ){ //说明是拾物
            data.put("model","拾物");
        }else{
            data.put("model","失物");
        }
        list.add(data);                //加本次的map进入到链表中
    }
    Dao.close(rs, p, conn);
} catch (Exception e) {
    e.printStackTrace();
}
if( sign == 1 ){
    return list;
}else{
    return null;
}
}

```

```

//返回部分报名人及信息
public static List signupList(int page,int limit,String which){
    int sign = 0;           //标记一下当前页数是否有数据
    //参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
    List list = new ArrayList();
    try {
        Connection conn = Dao.getConnection();           //连接数据库
        // 进行分页
        // page 页码 limit 每页数量
        // select * from TABLE_NAME where ... order by ... limit (page
-1) * limit, limit
        PreparedStatement p = conn.prepareStatement("select * from
data_enrol where model=? limit ?,?");           //查看报名信息
        p.setString(1,which);
        p.setInt(2,(page -1) * limit);
        p.setInt(3, limit);
        ResultSet rs = p.executeQuery();

        while(rs.next()){
            sign = 1;
            Map<String,String> data = new HashMap();

            data.put("category_value", rs.getString("category_value"));
            data.put("LostTime", rs.getString("LostTime"));
            data.put("LostPosition", rs.getString("LostPosition"));
            data.put("LostPerson", rs.getString("LostPerson"));
            data.put("LostTelNumber", rs.getString("LostTelNumber"));
            String s = rs.getString("LostThingsPicture");
            if( s == null ){
                data.put("LostThingsPicture",null);
                data.put("FirstPicture",null);
            }else{
                s = s.replaceAll("\\\\", "/");
                data.put("LostThingsPicture",s);
                String[] tmpa = SqlArray.SeparationArray(s);
                data.put("FirstPicture",tmpa[0]);
            }

            data.put("LostMessage", rs.getString("LostMessage"));
            if( "picker".equals(rs.getString("model")) ){ //说明是拾物
                data.put("model","拾物");
            }else{
                data.put("model","失物");
            }
            list.add(data);
        }
    }
}

```

```

    }
    Dao.close(rs, p, conn);
} catch (Exception e) {
    e.printStackTrace();
}
//System.out.println(list);
if( sign == 1 ){
    return list;
}else{
    return null;
}
}
}

```

//进行查询返回部分报名人及信息

```

public static List searchList(int page,int limit,String
category_value,String begintime,String endtime,String find,String which){
    int sign = 0;          //标记一下当前页数是否有数据

    //limit ?,?;
    //进行拼接
    String sql = "select * from data_enrol where model='"+which+"'";
    if( category_value != null && !"".equals(category_value)){
//说明有筛选类别
        sql += (" and category_value='" + category_value+"'");
        if( find != null && !"".equals(find)){
            sql += (" and (LostPosition like '%" + find + "%' or
LostMessage like '%" + find+"%' )");
        }
    }else{
        if( find != null && !"".equals(find)){
            sql += (" and LostPosition like '%" + find + "%' or LostMessage
like '%" + find+"%'");
        }
    }

    sql += " limit ?,?;";

    //参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
    List list = new ArrayList();
    int res = 0;
    try {
        Connection conn = Dao.getConnection();          //连接数据库
        // page 页码 limit 每页数量
        // select * from TABLE_NAME where ... order by ... limit (page
-1) * limit, limit
        PreparedStatement p = conn.prepareStatement(sql);          //查看报名

```

```

p.setInt(1,(page -1) * limit);
p.setInt(2, limit);
System.out.println(p);
ResultSet rs = p.executeQuery();
//从前端或者自己模拟一个日期格式，转为String即可
SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
//将字符串日期转为日期
Date datebegin = null,dateend = null;
int time = 0;
if( begintime != null && !"".equals(begintime) && endtime != null
&& !"".equals(endtime)){
    datebegin = format.parse(begintime);
    dateend = format.parse(endtime);
    time = 1;
}
while(rs.next()){
    Map<String,String> data = new HashMap();
    Date when = format.parse(rs.getString("LostTime"));
    if( time == 1){
        //判断现在时间是否在选择的时间范围内
        if(!(when.after(datebegin) && when.before(dateend)) ){
            continue;
        }
    }

    sign = 1;
    data.put("category_value", rs.getString("category_value"));
    data.put("LostTime", rs.getString("LostTime"));
    data.put("LostPosition", rs.getString("LostPosition"));
    data.put("LostPerson", rs.getString("LostPerson"));
    data.put("LostTelNumber", rs.getString("LostTelNumber"));

    String s = rs.getString("LostThingsPicture");
    if( s == null){
        data.put("LostThingsPicture",null);
        data.put("FirstPicture",null);
    }else{
        s = s.replaceAll("\\\\", "/");
        //字符串替代
        data.put("LostThingsPicture",s);
        String[] tmpa = SqlArray.SeparationArray(s);
        data.put("FirstPicture",tmpa[0]);
    }

    data.put("LostMessage", rs.getString("LostMessage"));
    if( "picker".equals(rs.getString("model")) ){ //说明是拾物
        data.put("model","拾物");
    }else{

```

```

        data.put("model", "失物");
    }
    list.add(data);
}
Dao.close(rs, p, conn);
} catch (Exception e) {
    e.printStackTrace();
}
//System.out.println(list);
if( sign == 1 ){
    return list;
}else{
    return null;
}
}

}

```

applicationContext.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>

```

dispatcher-servlet.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.2.xsd">

    <!--dispatcher-servlet.xml文件，这个文件主要用来配置HandlerMapping-->

```

```

<!-- 配置自动扫描的包 -->
<context:component-scan base-package="com.controller">
</context:component-scan>
<!-- 静态资源(js、image等)的访问 -->
<mvc:default-servlet-handler/>
<!-- 开启注解 -->
<mvc:annotation-driven/>
<!-- HandlerMapper和HandlerAdapter-->

<!--<bean class="com.test.controller.WelcomeController"></bean>-->
<!-- 配置视图解析器 如何把handler 方法返回值解析为实际的物理视图 -->
<bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
    <property name = "prefix" value="/WEB-INF/views/"></property>
    <property name = "suffix" value = ".jsp"></property>
</bean>
<!-- 配置MultipartResolver，用于上传文件，使用spring的
CommonsMultipartResolver -->
<bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="5000000" />
    <property name="defaultEncoding" value="UTF-8" />
</bean>

</beans>

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd"
    version="5.0">
    <context-param> <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext.xml</param-value>
    </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<!-- 编码过滤器 ， 保证数据库中文不乱码 -->
<filter>
    <filter-name>characterEncodingFileter</filter-name>
    <filter-

```



```

class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
        <param-name>forceEncoding</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>

<filter-mapping>
    <filter-name>characterEncodingFileter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<!-- 自己用的过滤器 -->
<filter>
    <filter-name>crossFilter</filter-name>
    <filter-class>com.filter.CrossFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>crossFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<!--web.xml是用来启动DispatcherServlet -->

<servlet>
    <!--这里写成对应名字-->
    <servlet-name>dispatcher</servlet-name>

    <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

    <!-- 启动顺序，数字越小，启动越早 -->
    <load-on-startup>1</load-on-startup>
</servlet>
<!--所有请求都会被dispatcher拦截 -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>

```

```
<welcome-file>default.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

(2022.4.4)

完善版2.0

完善:

- 1.简化部分接口参数名称
- 2.解决数据库中文乱码 【web.xml加代码和数据库加链接】
- 3.部分接口一个改成两个

修正部分

HelloController.java

```
package com.controller;

import com.dao.SqlArray;
import com.file.FileUpload;
import com.logic.Login;
import com.url.Request;
import com.weixin.SignUp;
import org.springframework.http.*;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import com.google.gson.Gson;
//import net.sf.json.JSONObject;

import com.alibaba.fastjson.JSONObject;
import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.*;

//使用Controller来标识它是一个控制器
@Controller
```

```
//@CrossOrigin
@RequestMapping(value = "/api")
public class HelloController {
    //全局变量
    public static String appid = "wxf58558547a7b352a";
    public static String secret = "c2ca1a3ee37256c970a0d6c4f4e475fc" ;

    //测试springmvc是否创建成功
    @RequestMapping(value = "/helloworld")
    public String Hello() {
        return "cs";
    }

    /* 测试链接
    * 请求: Get
    * 链接: 地址/api/cs
    * 参数: 无
    * 返回: json {"msg":"HelloWorld"}
    */
    @RequestMapping(value = "/cs")
    public @ResponseBody Map<String,String> cs() {
        Map<String,String> data = new HashMap<String,String>();
        data.put("msg","HelloWorld" );
        return data;
    }

    /*
    * 获取两张图片
    * 请求: Get
    * 链接: 地址/api/getpicture
    * 参数: 无
    * 返回: json
    * {
    *     "msg":"ok",
    *     "picture1":"/img/LostButtonPicture.png",
    *     "picture2":"/img/PickUpButtonPicture.png"
    * }
    */
    @RequestMapping(value = "/getpicture")
    public @ResponseBody
    Map<String,String> getPicture() {
        Map<String,String> data = new HashMap<String,String>();

        data.put("msg","ok" );
        data.put("picture1","/img/LostButtonPicture.png");
        data.put("picture2","/img/PickUpButtonPicture.png");
        return data;
    }
}
```

```

/* 注册
 * 请求: Get
 * 链接: 地址/api/register
 * 参数: 无
 * 返回: json {"msg":"HelloWorld"}
 */
@RequestMapping(value = "/register",method = RequestMethod.POST)
public @ResponseBody
Map<String,String> register(HttpServletRequest request) {
    String code = request.getParameter("code");
    Map<String,String> data = new HashMap<String,String>();

    data.put("msg","ok" );
    return data;
}

/* 获取手机号码
 * 请求: POST
 * 链接: 地址/api/getphone
 * 参数: 手机号码获取的code
 * 返回: json {"msg":"HelloWorld"}
 */
@RequestMapping(value = "/getphone",method = RequestMethod.POST)
public @ResponseBody Map<String,String> getPhone(HttpServletRequest
request) {
    Map<String,String> data = new HashMap<String,String>();
    try{
        //获取前端传过来的code
        String code = request.getParameter("code");
        System.out.println(code);

        //获取获取小程序全局唯一后台接口调用凭据 (access_token)
        String getTokenUrl = "https://api.weixin.qq.com/cgi-bin/token?
grant_type=client_credential&appid="+appid+"&secret="+secret;
        String jsonString = Request.doGet(getTokenUrl);
        System.out.println(jsonString);

        //String转JSON
        JSONObject jsonObject = JSONObject.parseObject(jsonString);
        String access_token = jsonObject.getString("access_token");

        //提交参数
        String getPhoneUrl =
"https://api.weixin.qq.com/wxa/business/getuserphonenumber?
access_token="+access_token;
        Map<String,Object> map = new HashMap<String,Object>();
        //map.put("access_token",access_token);

```

```

        map.put("code", code);
        JSONObject json = new JSONObject(map);
        System.out.println(map);
        System.out.println(json);
        String jsonStringb = Request.doPostForm(getPhoneUrl, json);
        System.out.println(jsonStringb);

        //String转JSON
        JSONObject jsonObject2 = JSONObject.parseObject(jsonStringb);
        HashMap hashMap =
JSONObject.parseObject(jsonObject2.toJSONString(), HashMap.class);
        if( 0 == (int)hashMap.get("errcode") ){           //请求成功
            data.put("msg", "ok" );
            JSONObject tmp2 = (JSONObject)hashMap.get("phone_info");
            data.put("phone", (String)tmp2.get("phoneNumber"));
        }else{
            data.put("msg", "fail" );
            data.put("error", (int)hashMap.get("errcode")+" " );
        }

    }catch (Exception e){
        System.out.println(e);
        data.put("msg", "fail" );
    }

    return data;
}

/* 登录
 * 请求: POST
 * 链接: 地址/api/login
 * 参数: 登录获取的code
 * 返回: json {"skey":""}
 */
@RequestMapping(value = "/login", method = RequestMethod.POST)
public @ResponseBody Map<String, String> login(HttpServletRequest request)
{
    Map<String, String> data = new HashMap<String, String>();
    try {
        //获取前端传过来的code
        String code = request.getParameter("code");
        //Get请求（登录凭证校验）
        String getAuthUrl =
"https://api.weixin.qq.com/sns/jscode2session?appid=" + appid + "&secret=" +
secret + "&js_code=" + code + "&grant_type=authorization_code";
        String jsonString = Request.doGet(getAuthUrl);
        System.out.println(jsonString);
    }
}

```

```

        //String转JSON, 再json转为map
        JSONObject jsonObject = JSONObject.parseObject(jsonString);
        HashMap hashMap =
        JSONObject.parseObject(jsonObject.toJSONString(), HashMap.class);

        //注意这里要加上 hashMap.get("errcode") == null
        if ( hashMap.get("errcode") == null || 0 == (int)
hashMap.get("errcode")) { //请求成功
            data.put("msg", "ok");
            //得到openid和session_key去生成3rd_session
            //这个生成3rd_session的方式自己决定即可, 比如使用SHA或Base64算法都可以。例如: 将session_key或openid+session_key作为SHA或Base64算法的输入, 输出结果做为3rd_session来使用, 同时要将openid, session_key, 3rd_session三者关联存储到数据库中, 方便下次拿3rd_session获取session_key或openid做其他处理。
            String openid = (String) hashMap.get("openid");
            String session_key = (String) hashMap.get("session_key");

            //uuid生成唯一
key(https://blog.csdn.net/weixin\_38169886/article/details/99820453?utm\_medium=distribute.pc\_relevant.none-task-blog-2~default~baidujs\_baidulandingword~default-5.pc\_relevant\_default&spm=1001.2101.3001.4242.4&utm\_relevant\_index=8)
            String skey = UUID.randomUUID().toString();

            //判断是否注册过
            boolean tmp = Login.checkUser(openid);
            if (tmp == false) { //没有注册过
                Login.register(openid, skey);
                data.put("msg", "ok");
                data.put("skey", skey);
            } else { //注册过, 更新新的skey
                Login.updateKey(openid, skey);
                data.put("skey", skey);
            }
        } else {
            data.put("msg", "fail");
            data.put("error", (int) hashMap.get("errcode") + "");
        }
    }
    catch(Exception e){
        e.printStackTrace();
        data.put("msg", "fail");
        data.put("error", e.toString());
    }
    return data;
}

```

//文件上传

```

/*
* HTTP请求: POST
* 参数: file    (可以是一个文件, 也可以是多个文件)
* 请求链接: 地址/api/upload
* {"msg":"ok","filename":""}
* {"msg":"fail","error":""}
* */
@RequestMapping(value = "/upload", method = RequestMethod.POST)
public @ResponseBody Map<String,String> Upload(@RequestParam(value =
"file",required = false) MultipartFile[] file, HttpServletResponse response,
HttpServletRequest request) throws IOException {
    Map<String, String> data = new HashMap<String, String>();
    try {
        System.out.println("总共有"+file.length+"个文件");
        // 文件上传到服务器的位置"/files"
        System.out.println("正在上传文件");

        List<String> fileList = new ArrayList<>();           //全部的数组
        for( MultipartFile f:file){
            String filelocation = FileUpload.SaveServer(f,request);
            if( filelocation != null ){
                fileList.add(filelocation);
            }else{
                data.put("msg","fail");
                data.put("error","文件上传失败");
                return data;
            }
        }
        //合并
        String all_file = SqlArray.Merge(fileList);           //全部的
        System.out.println(all_file);
        data.put("msg","ok");
        data.put("filepath",all_file);

    }catch (Exception e){
        e.printStackTrace();
    }
    return data;
}

```

//失主提交页面

/*

HTTP请求: POST

请求链接: 地址/api/ownerEnrol

参数: 7

category_value 失物类别

LostTime 丢失时间

LostPosition 丢失地点

```

        LostPerson      联系人
        LostTelNumber   联系方式
        LostThingsPicture 失物照片
        FilePath        图片位置
        LostMessage     备注
    返回JSON
    提交成功
    {
        "msg":"ok"
    }
    提交失败
    {
        "msg":"fail",
        "error":错误原因
    }

    */
    //失主
    @RequestMapping(value = "/ownerEnrol", method = RequestMethod.POST)
    public @ResponseBody Map<String,String> Enrol(HttpServletRequest response,
    response, HttpServletRequest request) throws IOException {
        Map<String, String> data = new HashMap<String, String>();
        try {
            /*
            category_value 失物类别
            LostTime        丢失时间
            LostPosition     丢失地点
            LostPerson       联系人
            LostTelNumber    联系方式
            LostThingsPicture 失物照片
            LostMessage      备注
            * */
            String category_value = request.getParameter("category_value");
            String LostTime = request.getParameter("Time");
            String LostPosition = request.getParameter("Position");
            String LostPerson = request.getParameter("Person");
            String LostTelNumber = request.getParameter("TelNumber");
            String LostMessage = request.getParameter("Message");
            String FilePath = request.getParameter("FilePath");

            if ( category_value == null
                || LostTime == null || LostTelNumber == null ||
                LostMessage == null
                || "".equals(category_value) || "".equals(LostTime) ||
                "".equals(LostTelNumber) || "".equals(LostMessage) ) {
                data.put("msg", "fail");
                data.put("error", "失物类型,失物丢失时间,失物情况,联系方式为必填项且
                不能为空");
                throw new NullPointerException("xx");                //抛出异常
            }
        }
    }

```



```

    }

    //进行登记
    if
(SignUp.Enrol(category_value, LostTime, LostPosition, LostPerson, LostTelNumber, F
ilePath, LostMessage, "owner")) {
        data.put("msg", "ok");
    }else{
        data.put("msg", "fail");
        data.put("error", "写入数据库失败");
    }
}catch (Exception e){
    e.printStackTrace();
}
return data;
}

```

//拾主提交页面

/*

HTTP请求: POST

请求链接: 地址/api/pickerEnrol

参数: 7

category_value	失物类别
LostTime	丢失时间
LostPosition	丢失地点
LostPerson	联系人
LostTelNumber	联系方式
FilePath	图片位置
LostMessage	备注

返回JSON

提交成功

```

{
    "msg": "ok"
}

```

提交失败

```

{
    "msg": "fail",
    "error": "错误原因"
}

```

*/

//拾主

@RequestMapping(value = "/pickerEnrol", method = RequestMethod.POST)

```

public @ResponseBody Map<String, String> pickerEnrol(HttpServletRequest response
response, HttpServletRequest request) throws IOException {
    Map<String, String> data = new HashMap<String, String>();
    try {
        /*

```

```

        category_value      失物类别
        LostTime            丢失时间
        LostPosition        丢失地点
        LostPerson          联系人
        LostTelNumber       联系方式
        LostThingsPicture   失物照片
        LostMessage         备注
    * */
    String category_value = request.getParameter("category_value");
    String LostTime = request.getParameter("Time");
    String LostPosition = request.getParameter("Position");
    String LostPerson = request.getParameter("Person");
    String LostTelNumber = request.getParameter("TelNumber");
    String LostMessage = request.getParameter("Message");
    String FilePath = request.getParameter("FilePath");

    if ( category_value == null
        || LostTime == null || LostTelNumber == null ||
        LostMessage == null
        || "".equals(category_value) || "".equals(LostTime) ||
        "".equals(LostTelNumber) || "".equals(LostMessage) ) {
        data.put("msg", "fail");
        data.put("error", "失物类型,失物丢失时间,失物情况,联系方式为必填项且
        不能为空");
        throw new NullPointerException("xx");          //抛出异常
    }

    //进行登记
    if
    (SignUp.Enrol(category_value,LostTime,LostPosition,LostPerson,LostTelNumber,F
    ilePath,LostMessage,"picker")) {
        data.put("msg", "ok");
    }else{
        data.put("msg","fail");
        data.put("error","写入数据库失败");
    }
    }catch (Exception e){
        e.printStackTrace();
    }
    return data;
}

//显示失物数据
/*
HTTP请求: POST
请求链接: 地址/api/ownershowdata
参数:      page(字符串)

```

返回JSON

获取成功

```
{
    "msg": "ok",
    "num": "总数量"
    "data": {
        (数组类型的数据)
    }
}
```

获取失败

```
{
    "msg": "fail",
    "num": "总数量",
    "error": "错误原因"
}
```

*/

```
@RequestMapping(value = "/ownershowdata", method = RequestMethod.POST)
public @ResponseBody Map<String, Object> ownershowData(HttpServletResponse
response, HttpServletRequest request) throws IOException {
    Map<String, Object> data = new HashMap<String, Object>();
    try {
        int num = SignUp.queryNum("owner");
        String page = request.getParameter("page");
        List list = SignUp.signupList(Integer.parseInt(page), 10, "owner");
//每页显示10条数据
        data.put("num", ""+num);
        if( list == null ){
            data.put("msg", "fail");
            data.put("error", "当前页面已经无数据");
        }else{
            data.put("msg", "ok");
            data.put("data", list);
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    return data;
}
```

//显示失物搜索数据

/*

HTTP请求: POST

请求链接: 地址/api/ownersearch

参数:

page	页数	必填
search		必填
category_value		非必填
begintime		非必填

```

        endtime                非必填
    返回JSON
    获取成功
    {
        "msg":"ok",
        "data":{
            (数组类型的数据)
        }
    }
    获取失败
    {
        "msg":"fail",
        "error":错误原因
    }

    */
    @RequestMapping(value = "/ownersearch", method = RequestMethod.POST)
    public @ResponseBody Map<String, Object> ownersearch(HttpServletRequest response,
        HttpServletRequest request) throws IOException {
        Map<String, Object> data = new HashMap<String, Object>();
        try {
            int num = SignUp.queryNum("owner");
            String page = request.getParameter("page");
            String search = request.getParameter("search");
            String category_value = request.getParameter("category_value");
            String begintime = request.getParameter("begintime");
            String endtime = request.getParameter("endtime");
            if( page == null || "".equals(page) || search == null ||
            "".equals(search) ){
                data.put("msg", "fail");
                data.put("error", "page和search为必填项");
                return data;
            }
            List list =
            SignUp.searchList(Integer.parseInt(page), 10, category_value, begintime, endtime,
            search, "owner");           //每页显示10条数据

            if( list == null ){
                data.put("msg", "fail");
                data.put("error", "当前页面已经无数据");
            }else{
                data.put("msg", "ok");
                System.out.println(list.toString());
                data.put("data", list);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return data;
    }

```

```

}
//显示拾物数据
/*
    HTTP请求: POST
    请求链接: 地址/api/pickershowdata
    参数:      page(字符串)
    返回JSON
        获取成功
        {
            "msg":"ok",
            "num":“总数量”
            "data":{
                (数组类型的数据)
            }
        }
        获取失败
        {
            "msg":"fail",
            "num":“总数量”,
            "error":错误原因
        }
*/
@RequestMapping(value = "/pickershowdata", method = RequestMethod.POST)
public @ResponseBody Map<String,Object>
pickershowData(HttpServletResponse response, HttpServletRequest request)
throws IOException {
    Map<String, Object> data = new HashMap<String, Object>();
    try {
        int num = SignUp.queryNum("picker");
        String page = request.getParameter("page");
        List list =
SignUp.signupList(Integer.parseInt(page),10,"picker");           //每页显示10条数据

        data.put("num",""+num);
        if( list == null ){
            data.put("msg","fail");
            data.put("error","当前页面已经无数据");
        }else{
            data.put("msg","ok");
            data.put("data",list);
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    return data;
}

//显示拾物搜索数据

```

```

/*
    HTTP请求: POST
    请求链接: 地址/api/pickersearch
    参数:
        page          页数          必填
        search         必填
        category_value 非必填
        begintime      非必填
        endtime        非必填
    返回JSON
    获取成功
    {
        "msg":"ok",
        "data":{
            (数组类型的数据)
        }
    }
    获取失败
    {
        "msg":"fail",
        "error":错误原因
    }
*/
@RequestMapping(value = "/pickersearch", method = RequestMethod.POST)
public @ResponseBody Map<String, Object> pickersearch(HttpServletRequest response,
    HttpServletRequest request) throws IOException {
    Map<String, Object> data = new HashMap<String, Object>();
    try {
        int num = SignUp.queryNum("picker");
        String page = request.getParameter("page");
        String search = request.getParameter("search");
        String category_value = request.getParameter("category_value");
        String begintime = request.getParameter("begintime");
        String endtime = request.getParameter("endtime");
        if( page == null || "".equals(page) || search == null ||
        "".equals(search) ){
            data.put("msg", "fail");
            data.put("error", "page和search为必填项");
            return data;
        }
        List list =
        SignUp.searchList(Integer.parseInt(page), 10, category_value, begintime, endtime,
        search, "owner"); //每页显示10条数据

        if( list == null ){
            data.put("msg", "fail");
            data.put("error", "当前页面已经无数据");
        }else{

```

```

        data.put("msg","ok");
        System.out.println(list.toString());
        data.put("data",list);
    }
} catch (Exception e){
    e.printStackTrace();
}
return data;
}
}

```

SignUp.java

```

package com.weixin;

import com.dao.Dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.*;

public class SignUp {

    /*
        category_value      失物类别
        LostTime             丢失时间
        LostPosition         丢失地点
        LostPerson           联系人
        LostTelNumber        联系方式
        LostThingsPicture    失物照片
        LostMessage          备注
    */
    //create table data_enrol(category_value char(50),LostTime
    char(100),LostPosition char(100),LostPerson char(50),LostTelNumber
    char(50),LostThingsPicture longtext,LostMessage longtext,model char(10));
    //进行报名
    public static boolean Enrol(String category_value, String LostTime,
    String LostPosition, String LostPerson, String LostTelNumber, String
    LostThingsPicture,String LostMessage,String model) {
        //ticket就是准考证号，准考证号就是登录名
        try {
            Connection conn = Dao.getConnection();

```

```

        /* 对报名的数据库进行补充 */
        PreparedStatement p = conn.prepareStatement("insert into
data_enrol(category_value,LostTime,LostPosition,LostPerson,LostTelNumber,Lost
ThingsPicture,LostMessage,model) VALUES (?, ?, ?, ?, ?, ?, ?, ?);");
        //对占位符进行补充
        p.setString(1, category_value);
        p.setString(2, LostTime);
        p.setString(3, LostPosition);
        p.setString(4, LostPerson);
        p.setString(5, LostTelNumber);
        p.setString(6, LostThingsPicture);
        p.setString(7, LostMessage);
        p.setString(8, model);
        System.out.println(p);
        p.executeUpdate();
        System.out.println("登记成功");
        Dao.close(p, conn);
        return true;
    } catch (Exception e) {
        System.out.println("登记失败");
        e.printStackTrace();
        return false;
    }
}

```

```

//查询总的数目
public static int queryNum(String which){
    int res = 0;
    try {
        Connection conn = Dao.getConnection(); //连接数据库
        PreparedStatement p = conn.prepareStatement("select count(*) from
data_enrol where model=?;"); //查看报名人数
        p.setString(1, which);
        ResultSet rs = p.executeQuery();
        if(rs.next()){
            res = rs.getInt(1);
            Dao.close(rs, p, conn);
            return res;
        }
        Dao.close(rs, p, conn);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return res;
}

```

//返回全部报名人及信息

```

public static List signupList(String which){

```



```

        int sign = 0;                //标记一下当前页数是否有数据
        //参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
        List list = new ArrayList();
        int res = 0;
        try {
            Connection conn = Dao.getConnection();                //连接数据库
            PreparedStatement p = conn.prepareStatement("select * from
data_enrol where model=?");                //查看报名信息
            p.setString(1,which);
            ResultSet rs = p.executeQuery();

            while(rs.next()){
                sign = 1;
                //category_value char(50),LostTime char(100),LostPosition
char(100),LostPerson char(50),LostTelNumber char(50),LostThingsPicture
longtext,LostMessage longtext,model char(10));
                Map<String,String> data = new HashMap();

                data.put("category_value", rs.getString("category_value"));
                data.put("LostTime", rs.getString("LostTime"));
                data.put("LostPosition", rs.getString("LostPosition"));
                data.put("LostPerson", rs.getString("LostPerson"));
                data.put("LostTelNumber", rs.getString("LostTelNumber"));
                data.put("LostThingsPicture",
rs.getString("LostThingsPicture"));
                data.put("LostMessage", rs.getString("LostMessage"));
                if( "picker".equals(rs.getString("model")) ){ //说明是拾物
                    data.put("model","拾物");
                }else{
                    data.put("model","失物");
                }
                list.add(data);
            }
            Dao.close(rs, p, conn);
        } catch (Exception e) {
            e.printStackTrace();
        }
        if( sign == 1 ){
            return list;
        }else{
            return null;
        }
    }

    //返回部分报名人及信息
    public static List signupList(int page,int limit,String which){

```

```

        int sign = 0;                //标记一下当前页数是否有数据
        //参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
        List list = new ArrayList();
        try {
            Connection conn = Dao.getConnection();                //连接数据库
            // page 页码 limit 每页数量
            // select * from TABLE_NAME where ... order by ... limit (page
            -1) * limit, limit
            PreparedStatement p = conn.prepareStatement("select * from
            data_enrol where model=? limit ?,?");                //查看报名信息
            p.setString(1, which);
            p.setInt(2, (page -1) * limit);
            p.setInt(3, limit);
            ResultSet rs = p.executeQuery();

            while(rs.next()){
                sign = 1;
                Map<String,String> data = new HashMap();

                data.put("category_value", rs.getString("category_value"));
                data.put("LostTime", rs.getString("LostTime"));
                data.put("LostPosition", rs.getString("LostPosition"));
                data.put("LostPerson", rs.getString("LostPerson"));
                data.put("LostTelNumber", rs.getString("LostTelNumber"));
                data.put("LostThingsPicture",
            rs.getString("LostThingsPicture"));
                data.put("LostMessage", rs.getString("LostMessage"));
                if( "picker".equals(rs.getString("model")) ){ //说明是拾物
                    data.put("model", "拾物");
                }else{
                    data.put("model", "失物");
                }
                list.add(data);
            }
            Dao.close(rs, p, conn);
        } catch (Exception e) {
            e.printStackTrace();
        }
        //System.out.println(list);
        if( sign == 1 ){
            return list;
        }else{
            return null;
        }
    }
}

```

```

//进行查询返回部分报名人及信息
public static List searchList(int page,int limit,String
category_value,String begintime,String endtime,String find,String which){
    int sign = 0;           //标记一下当前页数是否有数据

    //limit ?,?;
    String sql = "select * from data_enrol where model='"+which+"'";
    if( category_value != null && !"".equals(category_value)){
//说明有筛选类别
        sql += (" and category_value='" + category_value+"'");
        if( find != null && !"".equals(find)){
            sql += (" and (LostPosition like '%" + find + "%' or
LostMessage like '%" + find+"%'"));
        }
    }else{
        if( find != null && !"".equals(find)){
            sql += (" where LostPosition like '%" + find + "%' or
LostMessage like '%" + find+"%'");
        }
    }

    sql += " limit ?,?";

    //参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
    List list = new ArrayList();
    int res = 0;
    try {
        Connection conn = Dao.getConnection();           //连接数据库
        // page 页码 limit 每页数量
        // select * from TABLE_NAME where ... order by ... limit (page
-1) * limit, limit
        PreparedStatement p = conn.prepareStatement(sql);           //查看报名
信息

        p.setInt(1,(page -1) * limit);
        p.setInt(2, limit);
        System.out.println(p);
        ResultSet rs = p.executeQuery();
        //从前端或者自己模拟一个日期格式, 转为String即可
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        //将字符串日期转为日期
        Date datebegin = null,dateend = null;
        int time = 0;
        if( begintime != null && !"".equals(begintime) && endtime != null
&& !"".equals(endtime)){
            datebegin = format.parse(begintime);
            dateend = format.parse(endtime);
            time = 1;

```

```

    }
    while(rs.next()){
        Map<String,String> data = new HashMap();
        Date when = format.parse(rs.getString("LostTime"));
        if( time == 1){
            if(!(when.after(datebegin) && when.before(dateend)) ){
                continue;
            }
        }

        sign = 1;
        data.put("category_value", rs.getString("category_value"));
        data.put("LostTime", rs.getString("LostTime"));
        data.put("LostPosition", rs.getString("LostPosition"));
        data.put("LostPerson", rs.getString("LostPerson"));
        data.put("LostTelNumber", rs.getString("LostTelNumber"));
        data.put("LostThingsPicture",
rs.getString("LostThingsPicture"));
        data.put("LostMessage", rs.getString("LostMessage"));
        if( "picker".equals(rs.getString("model")) ){ //说明是拾物
            data.put("model", "拾物");
        }else{
            data.put("model", "失物");
        }
        list.add(data);
    }
    Dao.close(rs, p, conn);
} catch (Exception e) {
    e.printStackTrace();
}
//System.out.println(list);
if( sign == 1 ){
    return list;
}else{
    return null;
}
}

}
}

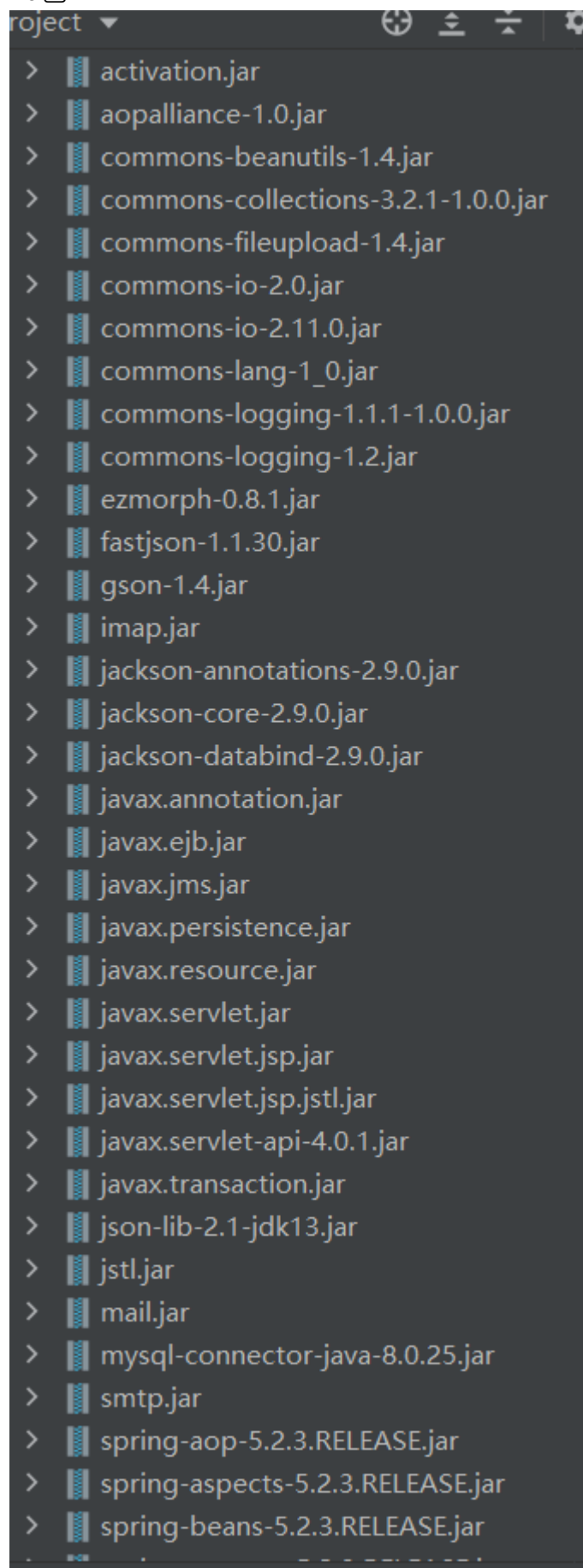
```

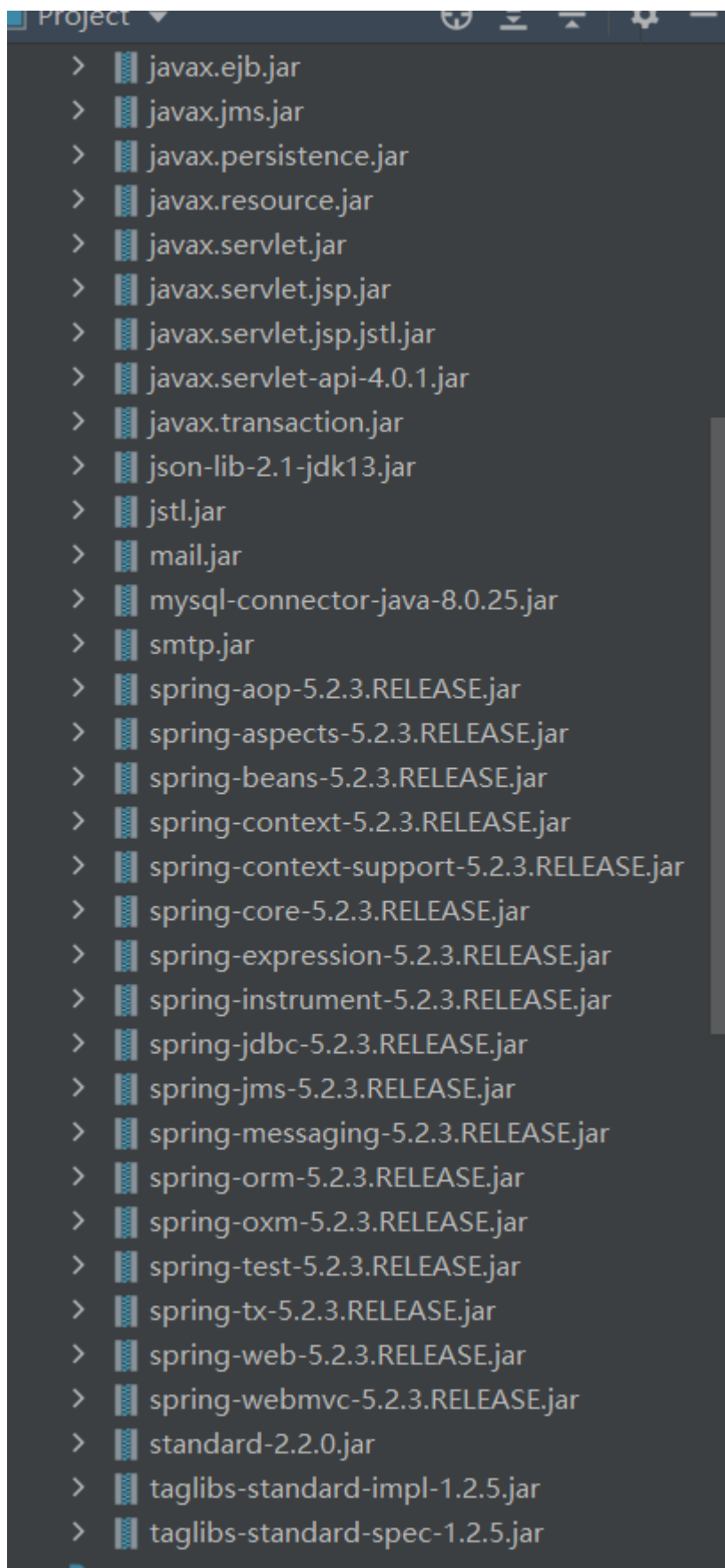
(2022.3.28)

基本功能版1.0

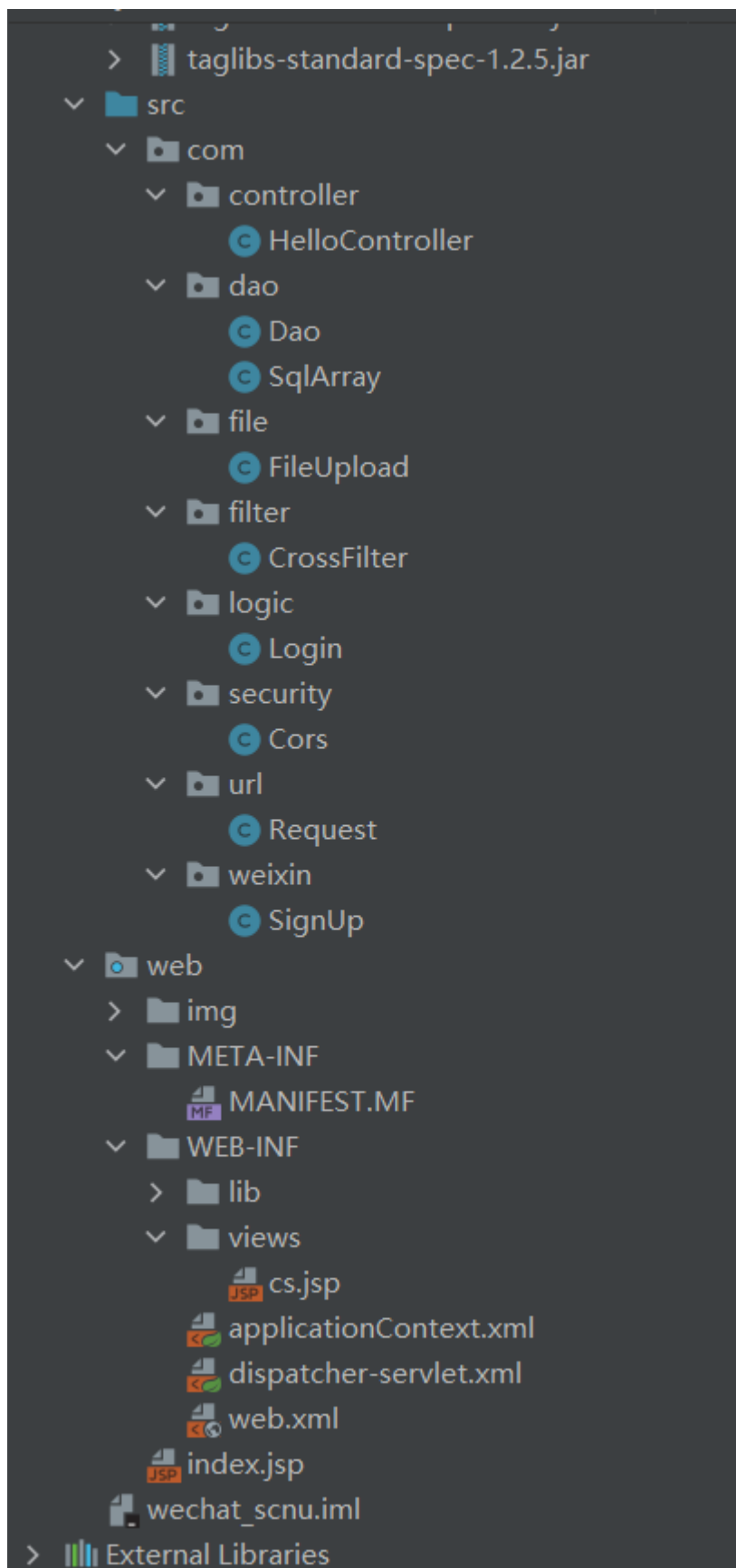
项目结构

lib包

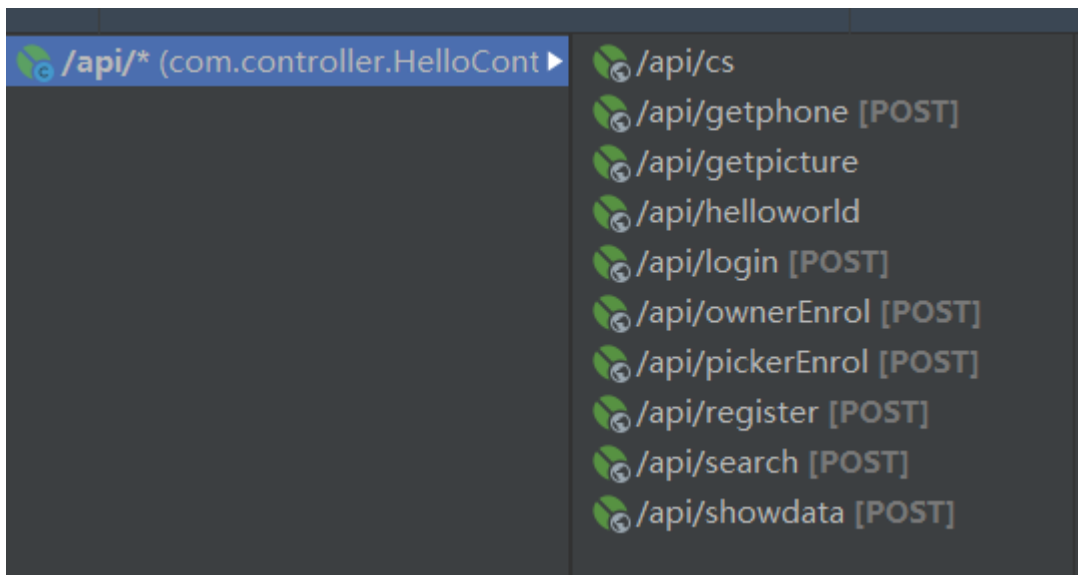




项目结构



接口



新增了：

- 1.登录接口
- 2.失物登记接口
- 3.拾物登记接口
- 4.显示数据接口
- 5.搜索筛选接口

上个版本功能：

- 1.返回两张图片
- 2.得到用户手机号码

HelloController.java

```
package com.controller;

import com.dao.SqlArray;
import com.file.FileUpload;
import com.logic.Login;
import com.url.Request;
import com.weixin.SignUp;
import org.springframework.http.*;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import com.google.gson.Gson;
//import net.sf.json.JSONObject;

import com.alibaba.fastjson.JSONObject;
import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;
```



```

import javax.servlet.http.HttpServletResponse;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.*;

//使用Controller来标识它是一个控制器
@Controller
//@CrossOrigin
@RequestMapping(value = "/api")
public class HelloController {
    //全局变量
    public static String appid = "wxf58558547a7b352a";
    public static String secret = "c2ca1a3ee37256c970a0d6c4f4e475fc" ;

    //测试springmvc是否创建成功
    @RequestMapping(value = "/helloworld")
    public String Hello() {
        return "cs";
    }

    /* 测试链接
    * 请求: Get
    * 链接: 地址/api/cs
    * 参数: 无
    * 返回: json {"msg":"HelloWorld"}
    */
    @RequestMapping(value = "/cs")
    public @ResponseBody Map<String,String> cs() {
        Map<String,String> data = new HashMap<String,String>();
        data.put("msg","HelloWorld" );
        return data;
    }

    /*
    * 获取两张图片
    * 请求: Get
    * 链接: 地址/api/getpicture
    * 参数: 无
    * 返回: json
    * {
    *     "msg":"ok",
    *     "picture1":"/img/LostButtonPicture.png",
    *     "picture2":"/img/PickUpButtonPicture.png"
    * }
    */

```

```

@RequestMapping(value = "/getpicture")
public @ResponseBody
Map<String,String> getPicture() {
    Map<String,String> data = new HashMap<String,String>();

    data.put("msg","ok" );
    data.put("picture1","/img/LostButtonPicture.png");
    data.put("picture2","/img/PickUpButtonPicture.png");
    return data;
}

/* 注册
 * 请求: Get
 * 链接: 地址/api/register
 * 参数: 无
 * 返回: json {"msg":"HelloWorld"}
 */
@RequestMapping(value = "/register",method = RequestMethod.POST)
public @ResponseBody
Map<String,String> register(HttpServletRequest request) {
    String code = request.getParameter("code");
    Map<String,String> data = new HashMap<String,String>();

    data.put("msg","ok" );
    return data;
}

/* 获取手机号码
 * 请求: POST
 * 链接: 地址/api/getphone
 * 参数: 手机号码获取的code
 * 返回: json {"msg":"HelloWorld"}
 */
@RequestMapping(value = "/getphone",method = RequestMethod.POST)
public @ResponseBody Map<String,String> getPhone(HttpServletRequest
request) {
    Map<String,String> data = new HashMap<String,String>();
    try{
        //获取前端传过来的code
        String code = request.getParameter("code");
        System.out.println(code);

        //获取获取小程序全局唯一后台接口调用凭据 (access_token)
        String getTokenUrl = "https://api.weixin.qq.com/cgi-bin/token?
grant_type=client_credential&appid="+appid+"&secret="+secret;
        String jsonString = Request.doGet(getTokenUrl);
        System.out.println(jsonString);
    }
}

```

```

        //String转JSON
        JSONObject jsonObject = JSONObject.parseObject(jsonStringa);
        String access_token = jsonObject.getString("access_token");

        //提交参数
        String getPhoneUrl =
"https://api.weixin.qq.com/wxa/business/getuserphonenumber?
access_token="+access_token;
        Map<String, Object> map = new HashMap<String, Object>();
        //map.put("access_token", access_token);
        map.put("code", code);
        JSONObject json = new JSONObject(map);
        System.out.println(map);
        System.out.println(json);
        String jsonStringb = Request.doPostForm(getPhoneUrl, json);
        System.out.println(jsonStringb);

        //String转JSON
        JSONObject jsonObject2 = JSONObject.parseObject(jsonStringb);
        HashMap hashMap =
JSONObject.parseObject(jsonObject2.toJSONString(), HashMap.class);
        if( 0 == (int)hashMap.get("errcode") ){ //请求成功
            data.put("msg", "ok" );
            JSONObject tmp2 = (JSONObject)hashMap.get("phone_info");
            data.put("phone", (String)tmp2.get("phoneNumber"));
        }else{
            data.put("msg", "fail" );
            data.put("error", (int)hashMap.get("errcode")+"");
        }

    }catch (Exception e){
        System.out.println(e);
        data.put("msg", "fail" );
    }

    return data;
}

/* 登录
 * 请求: POST
 * 链接: 地址/api/login
 * 参数: 登录获取的code
 * 返回: json {"skey":""}
 */
@RequestMapping(value = "/login", method = RequestMethod.POST)
public @ResponseBody Map<String, String> login(HttpServletRequest request)
{

```

```

//获取前端传过来的code
String code = request.getParameter("code");
//Get请求（登录凭证校验）
String getAuthUrl = "https://api.weixin.qq.com/sns/jscode2session?
appid="+appid+"&secret="+secret+"&js_code="+code+"&grant_type=authorization_c
ode";

String jsonString = Request.doGet(getAuthUrl);
System.out.println(jsonString);
//String转JSON，再json转为map
JSONObject jsonObject = JSONObject.parseObject(jsonString);
HashMap hashMap = JSONObject.parseObject(jsonObject.toJSONString(),
HashMap.class);

Map<String, String> data = new HashMap<String, String>();

if( 0 == (int)hashMap.get("errcode") ){                                //请求成功
    data.put("msg","ok" );
    //得到openid和session_key去生成3rd_session
    //这个生成3rd_session的方式自己决定即可，比如使用SHA或Base64算法都可以。
    例如：将session_key或openid+session_key作为SHA或Base64算法的输入，输出结果做为
    3rd_session来使用，同时要将openid, session_key, 3rd_session三者关联存储到数据库中，
    方便下次拿3rd_session获取session_key或openid做其他处理。
    String openid = (String)hashMap.get("openid");
    String session_key = (String)hashMap.get("session_key");

    //uuid生成唯一
    key(https://blog.csdn.net/weixin_38169886/article/details/99820453?
    utm_medium=distribute.pc_relevant.none-task-blog-
    2~default~baidujs_baidulandingword~default-
    5.pc_relevant_default&spm=1001.2101.3001.4242.4&utm_relevant_index=8)
    String skey = UUID.randomUUID().toString();

    //判断是否注册过
    boolean tmp = Login.checkUser(openid);
    if( tmp == false ){                                                //没有注册过
        Login.register(openid,skey);
        data.put("msg","ok" );
        data.put("skey",skey);
    }else{                                                            //注册过，更新新的skey
        Login.updatekey(openid,skey);
        data.put("skey",skey);
    }
}else{
    data.put("msg","fail" );
    data.put("error",(int)hashMap.get("errcode")+"" );
}

return data;

```

```

}

//失主提交页面
/*
    HTTP请求: POST
    请求链接: 地址/api/ownerEnrol
    参数:      7
        category_value 失物类别
        LostTime        丢失时间
        LostPosition    丢失地点
        LostPerson      联系人
        LostTelNumber   联系方式
        LostThingsPicture 失物照片
        LostMessage     备注
    返回JSON
        提交成功
        {
            "msg":"ok"
        }
        提交失败
        {
            "msg":"fail",
            "error":错误原因
        }
*/
//失主
@RequestMapping(value = "/ownerEnrol", method = RequestMethod.POST)
public @ResponseBody Map<String,String> Enrol(@RequestParam(value = "LostThingsPicture",required = false) MultipartFile[] LostThingsPicture,
HttpServletResponse response, HttpServletRequest request) throws IOException
{
    Map<String, String> data = new HashMap<String, String>();
    try {
        /*
            category_value      失物类别
            LostTime            丢失时间
            LostPosition        丢失地点
            LostPerson          联系人
            LostTelNumber       联系方式
            LostThingsPicture   失物照片
            LostMessage         备注
        */
        String category_value = request.getParameter("category_value");
        String LostTime = request.getParameter("LostTime");
        String LostPosition = request.getParameter("LostPosition");
        String LostPerson = request.getParameter("LostPerson");
        String LostTelNumber = request.getParameter("LostTelNumber");
        String LostMessage = request.getParameter("LostMessage");
    }
}

```

```

        if ( category_value == null
            || LostTime == null || LostTelNumber == null ||
LostMessage == null
            || "".equals(category_value) || "".equals(LostTime) ||
"".equals(LostTelNumber) || "".equals(LostMessage) ) {
            data.put("msg", "fail");
            data.put("error", "失物类型,失物丢失时间,失物情况,联系方式为必填项且
不能为空");
            throw new NullPointerException("xx");           //抛出异常
        }

        System.out.println("总共有"+LostThingsPicture.length+"个文件");
        // 文件上传到服务器的位置“/files”
        System.out.println("正在上传文件");

        List<String> fileList = new ArrayList<>();           //全部的数组
        for( MultipartFile f:LostThingsPicture){
            String filelocation = FileUpload.SaveServer(f,request);
            if( filelocation != null ){
                fileList.add(filelocation);
            }else{
                data.put("msg","fail");
                data.put("error","文件上传失败");
                return data;
            }
        }
        //合并
        String all_file = SqlArray.Merge(fileList);         //全部的
        System.out.println(all_file);

        //进行登记
        if
(SignUp.Enrol(category_value,LostTime,LostPosition,LostPerson,LostTelNumber,a
ll_file,LostMessage,"owner")) {
            data.put("msg", "ok");
        }else{
            data.put("msg","fail");
            data.put("error","写入数据库失败");
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    return data;
}
}

```

```

//拾主提交页面
/*
    HTTP请求: POST
    请求链接: 地址/api/pickerEnrol
    参数:      7
        category_value 失物类别
        LostTime        丢失时间
        LostPosition    丢失地点
        LostPerson      联系人
        LostTelNumber   联系方式
        LostThingsPicture 失物照片
        LostMessage     备注
    返回JSON
        提交成功
        {
            "msg": "ok"
        }
        提交失败
        {
            "msg": "fail",
            "error": 错误原因
        }
*/
//拾主
@RequestMapping(value = "/pickerEnrol", method = RequestMethod.POST)
public @ResponseBody Map<String,String> pickerEnrol(@RequestParam(value =
"LostThingsPicture",required = false) MultipartFile[] LostThingsPicture,
HttpServletRequest response, HttpServletRequest request) throws IOException
{
    Map<String, String> data = new HashMap<String, String>();
    try {
        /*
            category_value      失物类别
            LostTime            丢失时间
            LostPosition        丢失地点
            LostPerson          联系人
            LostTelNumber       联系方式
            LostThingsPicture   失物照片
            LostMessage         备注
        */
        String category_value = request.getParameter("category_value");
        String LostTime = request.getParameter("LostTime");
        String LostPosition = request.getParameter("LostPosition");
        String LostPerson = request.getParameter("LostPerson");
        String LostTelNumber = request.getParameter("LostTelNumber");
        String LostMessage = request.getParameter("LostMessage");
    }
}

```

```

        if ( category_value == null
            || LostTime == null || LostTelNumber == null ||
LostMessage == null
            || "".equals(category_value) || "".equals(LostTime) ||
"".equals(LostTelNumber) || "".equals(LostMessage) ) {
            data.put("msg", "fail");
            data.put("error", "失物类型,失物丢失时间,失物情况,联系方式为必填项且
不能为空");
            throw new NullPointerException("xx");          //抛出异常
        }

        System.out.println("总共有"+LostThingsPicture.length+"个文件");
        // 文件上传到服务器的位置"/files"
        System.out.println("正在上传文件");

        List<String> fileList = new ArrayList<>();          //全部的数组
        for( MultipartFile f:LostThingsPicture){
            String filelocation = FileUpload.SaveServer(f,request);
            if( filelocation != null ){
                fileList.add(filelocation);
            }else{
                data.put("msg","fail");
                data.put("error","文件上传失败");
                return data;
            }
        }
        //合并
        String all_file = SqlArray.Merge(fileList);          //全部的
        System.out.println(all_file);

        //进行登记
        if
(SignUp.Enrol(category_value,LostTime,LostPosition,LostPerson,LostTelNumber,a
ll_file,LostMessage,"picker")) {
            data.put("msg", "ok");
        }else{
            data.put("msg","fail");
            data.put("error","写入数据库失败");
        }
    }catch (Exception e){
        e.printStackTrace();
    }
    return data;
}

//显示数据
/*

```


HTTP请求: POST

请求链接: 地址/api/showdata

参数: page(字符串)

返回JSON

获取成功

```
{
    "msg": "ok",
    "num": "总数量"
    "data": {
        (数组类型的数据)
    }
}
```

获取失败

```
{
    "msg": "fail",
    "num": "总数量",
    "error": "错误原因"
}
```

*/

@RequestMapping(value = "/showdata", method = RequestMethod.POST)

public @ResponseBody Map<String, Object> showData(HttpServletRequest response, HttpServletRequest request) throws IOException {

Map<String, Object> data = new HashMap<String, Object>();

try {

int num = SignUp.queryNum();

String page = request.getParameter("page");

List list = SignUp.signupList(Integer.parseInt(page), 10);

//每页显示10条数据

data.put("num", ""+num);

if(list == null){

data.put("msg", "fail");

data.put("error", "当前页面已经无数据");

}else{

data.put("msg", "ok");

data.put("data", list);

}

}catch (Exception e){

e.printStackTrace();

}

return data;

}

//显示搜索数据

/*

HTTP请求: POST

请求链接: 地址/api/search

参数:

page

页数

必填

search	必填
category_value	非必填
begintime	非必填
endtime	非必填

返回JSON

获取成功

```
{
    "msg": "ok",
    "data": {
        (数组类型的数据)
    }
}
```

获取失败

```
{
    "msg": "fail",
    "error": "错误原因"
}
```

*/

```
@RequestMapping(value = "/search", method = RequestMethod.POST)
public @ResponseBody Map<String, Object> search(HttpServletResponse
response, HttpServletRequest request) throws IOException {
    Map<String, Object> data = new HashMap<String, Object>();
    try {
        int num = SignUp.queryNum();
        String page = request.getParameter("page");
        String search = request.getParameter("search");
        String category_value = request.getParameter("category_value");
        String begintime = request.getParameter("begintime");
        String endtime = request.getParameter("endtime");
        if( page == null || "".equals(page) || search == null ||
"".equals(search) ){
            data.put("msg", "fail");
            data.put("error", "page和search为必填项");
            return data;
        }
        List list =
SignUp.searchList(Integer.parseInt(page), 10, category_value, begintime, endtime,
search); //每页显示10条数据

        if( list == null ){
            data.put("msg", "fail");
            data.put("error", "当前页面已经无数据");
        }else{
            data.put("msg", "ok");
            System.out.println(list.toString());
            data.put("data", list);
        }
    } catch (Exception e) {
```

```

        e.printStackTrace();
    }
    return data;
}

}

```

Dao.java

```

package com.dao;

import java.sql.*;

public class Dao {
    // 获取数据库连接
    //scnu_wechat      scnu_wechat      YJ8DhK5miHaYsx44
    public static Connection getConnection(){
        Connection conn = null;
        String url = "jdbc:mysql://****:3306/****";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            //Class.forName("com.mysql.jdbc.Driver");
            //数据库名字和密码在这里改!!!!
            conn = DriverManager.getConnection(url, "****", "****");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            System.out.println("数据库驱动加载出错");
        } catch (SQLException e) {
            e.printStackTrace();
            System.out.println("数据库出错");
        }
        return conn;
    }
    //关闭相关通道
    public static void close(ResultSet rs,PreparedStatement p,Connection
conn)
    {
        try
        {
            if(!rs.isClosed()){
                rs.close();
            }
            if(!p.isClosed()){
                p.close();
            }
        }
    }
}

```

```

        if(!conn.isClosed()){
            conn.close();
        }
    }
    catch(SQLException e)
    {
        e.printStackTrace();
        System.out.println("数据关闭出错");
    }
}

//关闭相关通道
public static void close(PreparedStatement p,Connection conn)
{
    try
    {
        if(!p.isClosed()){
            p.close();
        }
        if(!conn.isClosed()){
            conn.close();
        }
    }
    catch(SQLException e)
    {
        e.printStackTrace();
        System.out.println("数据关闭出错");
    }
}
}

```

SqlArray.java

```

package com.dao;

import java.util.ArrayList;
import java.util.List;

public class SqlArray {
    /*
    * 存储数组进数据库
    * 数据库用一个字段表示数组，字段类型为文本类型。
    * 程序存入数组到数据库的时候，利用join方法把数组转换为分隔符分隔的字符串，比如你的例子数组a[1]="第一步";a[2]="第二步";合并后为"第一步|第二步"，把这个合并后的字符串存入数据
    */
}

```

库你是会的。

- * 从数据库里面取出合并后的字符串"第一步|第二步"以后，利用split方法可以转换为数组。
- * 这个方法的最大的优点是可以保存个数不确定的数组，程序编写相当简单。
- * */

//合并

```
public static String Merge(String[] nameArr){
    String nameAll = "";
    for( int i = 0 ; i < nameArr.length ; i++ ){
        nameAll += (nameArr[i] + "|");
        if( i == nameArr.length - 1){
            nameAll += nameArr[i];
        }
    }
    return nameAll;
}
```

//合并

```
public static String Merge(List nameArr){
    if( nameArr.size() == 1 )    return String.valueOf(nameArr.get(0));
    String nameAll = "";
    for( int i = 0 ; i < nameArr.size() ; i++ ){
        nameAll += String.valueOf((nameArr.get(i) + "|"));
        if( i == nameArr.size() - 1){
            nameAll += String.valueOf(nameArr.get(i));
        }
    }
    return nameAll;
}
```

//分离（数组形式）

```
public static String[] SeparationArray(String nameAll){
    String[] nameArr = nameAll.split("\\|");
    return nameArr;
}
```

//分离（列表形式）

```
public static List SeparationList(String nameAll){
    String[] nameArr = nameAll.split("\\|");
    List list = new ArrayList();
    for(String name:nameArr){
        list.add(name);
    }
    return list;
}
```

}

FileUpload.java

```
package com.file;

import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;
import java.io.File;
import java.util.UUID;

public class FileUpload {

    //避免重命名
    public static String getUuidFileName(String fileName){
        //文件名以: uuid+"_"+文件的原始名称
        return UUID.randomUUID().toString()+ "_" +fileName;
    }

    //获取本来的名字

    //保存到服务器
    //返回位置
    public static String SaveServer(MultipartFile file, HttpServletRequest
request){
        System.out.println("正在上传文件");
        //得到上传文件的保存目录, 将上传的文件存放于WEB-INF目录下, 不允许外界直接访问,
保证上传文件的安全
        //String realpath = request.getServletContext().getRealPath("/WEB-
INF/files");
        //暂时就不保存在WEB-INF那里, 先放在files
        String realpath = request.getServletContext().getRealPath("files");

        //获取文件名字
        String fileName = getUuidFileName(file.getOriginalFilename());

        //文件上传
        File targetFile = new File(realpath, fileName);

        //如果不存在, 创建文件
        if (!targetFile.exists()) {
            targetFile.mkdirs();
        }
        // 上传
        try {
            file.transferTo(targetFile);
        }
```

```

        System.out.println("上传成功");
        //return realpath + '\\' + fileName;
        //返回相对路径就行
        return "\\files\\" + fileName;
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
}
}

```

CrossFilter.java

```

package com.filter;

import com.alibaba.fastjson.JSONObject;
import com.logic.Login;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

// https://blog.csdn.net/qq_32363305/article/details/82469451
public class CrossFilter implements Filter{
    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain
chain)
        throws IOException, ServletException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;
        response.addHeader("Access-Control-Allow-Origin",
request.getHeader("Origin"));
        response.addHeader("Access-Control-Allow-Methods", "POST, GET,
OPTIONS, PUT, DELETE, HEAD");
        response.addHeader("Access-Control-Max-Age", "27000000");
        response.addHeader("Access-Control-Allow-Headers", "X-PINGOTHER,

```

```

Origin, X-Requested-With, Content-Type, Accept, Cookie");
response.addHeader("Access-Control-Allow-Credentials", "true");
//获取请求时的sessionId
String skey = request.getHeader("skey");
if( skey == null || "".equals(skey) ){           //如果没有说明不需要登录的权限

    //该请求不需要验证session,直接通过
    System.out.println("该请求不需要过滤, 通过");
    chain.doFilter(request,response);
    return;
}else{
    //只有在缓存中存在该sessionId才能进行请求
    if ( Login.checkskey(skey) == false ) {
        // 登录信息已过期, 请重新登录
        System.out.println("登录信息失效, 请重新登录");
        //response.getWriter().write("登录信息失效, 请重新登录");
        response.setCharacterEncoding("UTF-8");
        response.setContentType("application/json; charset=utf-8");
        PrintWriter out = response.getWriter();
        JSONObject tmp = new JSONObject();
        tmp.put("msg", "fail");
        tmp.put("error", "登录信息失效, 请重新登录");
        out.append(tmp.toString());
        return;
    }
    System.out.println("session验证成功");
    chain.doFilter(request, response);
}
}

@Override
public void init(FilterConfig filterConfig) throws ServletException {
    //system.out.println("开始初始化");
}

@Override
public void destroy() {
    // system.out.println("销毁完成");
}
}

```

Login.java

```

package com.logic;

```



```

import com.dao.Dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class Login {
    //检测是否有该账户,有则返回真, 没有则返回假
    public static boolean checkUser(String username) {
        try {
            Connection conn = Dao.getConnection(); //连接数据库
            PreparedStatement p = conn.prepareStatement("select * from
user_data where open_id=?;"); //查询
            p.setString(1, username);
            ResultSet rs = p.executeQuery();
            if(rs.next()){
                String user_name = rs.getString("open_id");
                Dao.close(rs, p, conn);
                return true;
            }
            Dao.close(rs, p, conn);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return false;
    }

    //检测是否有该账户,有则返回真, 没有则返回假
    public static boolean checkskey(String skey) {
        try {
            Connection conn = Dao.getConnection(); //连接数据库
            PreparedStatement p = conn.prepareStatement("select * from
user_data where skey=?;"); //查询
            p.setString(1, skey);
            ResultSet rs = p.executeQuery();
            if(rs.next()){
                String user_name = rs.getString("skey");
                Dao.close(rs, p, conn);
                return true;
            }
            Dao.close(rs, p, conn);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return false;
    }
}

```

```

//注册
public static boolean register(String open_id,String skey){
    //ticket就是准考证号，准考证号就是登录名
    if(Login.checkUser(open_id) == false ){           //已经注册的就不能再注
        try {
            Connection conn = Dao.getConnection();
            /* 对账户密码的数据库进行补充 */
            PreparedStatement p = conn.prepareStatement("insert into
user_data(open_id,skey) VALUES (?,?);");
            //对占位符进行补充
            p.setString(1, open_id);
            p.setString(2, skey);
            p.executeUpdate();
            System.out.println("注册成功");
            Dao.close(p, conn);
            return true;
        } catch (Exception e) {
            e.printStackTrace();
        }
    }else{
        System.out.println("该账户已经注册过，请不要重复注册！");
        return false;
    }
    return false;
}

//更新skey
public static boolean updateskey(String open_id, String newskey) {
    try {
        Connection conn = Dao.getConnection();           //连接数据库
        //
        PreparedStatement p = conn.prepareStatement("update user_data set
skey=? where open_id=?;");
        p.setString(1,newskey);
        p.setString(2,open_id);
        p.executeUpdate();           //执行SQL语句(注意这里是execute，因为进
行的的是修改操作)
        Dao.close(p, conn);
        return true;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return false;
}

```

```
}
```

Request.java

```
package com.url;

import com.alibaba.fastjson.JSONObject;
import com.google.gson.Gson;
import com.google.gson.Gson;
import java.io.*;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.text.ParseException;
import java.util.*;

//https://www.cnblogs.com/mufengforward/p/10510337.html

public class Request {
    /**
     * 向指定URL发送GET方法的请求
     *
     * @param httpurl
     *      请求参数用?拼接在url后边，请求参数应该是
     *      name1=value1&name2=value2 的形式。
     * @return result 所代表远程资源的响应结果
     */
    public static String doGet(String httpurl) {
        HttpURLConnection connection = null;
        InputStream is = null;
        BufferedReader br = null;
        String result = null; // 返回结果字符串
        try {
            // 创建远程url连接对象
            URL url = new URL(httpurl);
            // 通过远程url连接对象打开一个连接，强转成HttpURLConnection类
            connection = (HttpURLConnection) url.openConnection();
            // 设置连接方式: get
            connection.setRequestMethod("GET");
            // 设置连接主机服务器的超时时间: 15000毫秒
            connection.setConnectTimeout(15000);
            // 设置读取远程返回的数据时间: 60000毫秒
            connection.setReadTimeout(60000);
```

```

// 发送请求
connection.connect();
// 通过connection连接, 获取输入流
if (connection.getResponseCode() == 200) {
    is = connection.getInputStream();
    // 封装输入流is, 并指定字符集
    br = new BufferedReader(new InputStreamReader(is, "UTF-8"));
    // 存放数据
    StringBuffer sbf = new StringBuffer();
    String temp = null;
    while ((temp = br.readLine()) != null) {
        sbf.append(temp);
        sbf.append("\r\n");
    }
    result = sbf.toString();
}
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // 关闭资源
    if (null != br) {
        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    if (null != is) {
        try {
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    connection.disconnect();// 关闭远程连接
}

return result;
}

/**
 *
 * @param httpUrl 请求的url
 * @param param form表单的参数 (key,value形式)

```

```

* @return
*/
public static String doPostForm(String httpUrl, Map param) {

    HttpURLConnection connection = null;
    InputStream is = null;
    OutputStream os = null;
    BufferedReader br = null;
    String result = null;
    try {
        URL url = new URL(httpUrl);
        // 通过远程url连接对象打开连接
        connection = (HttpURLConnection) url.openConnection();
        // 设置连接请求方式
        connection.setRequestMethod("POST");
        // 设置连接主机服务器超时时间: 15000毫秒
        connection.setConnectTimeout(15000);
        // 设置读取主机服务器返回数据超时时间: 60000毫秒
        connection.setReadTimeout(60000);

        // 默认值为: false, 当向远程服务器传送数据/写数据时, 需要设置为true
        connection.setDoOutput(true);
        // 默认值为: true, 当前向远程服务读取数据时, 设置为true, 该参数可有可无
        connection.setDoInput(true);
        // 设置传入参数的格式:请求参数应该是 name1=value1&name2=value2 的形式。
        connection.setRequestProperty("Content-Type",
"application/json");
        // 设置鉴权信息: Authorization: Bearer da3efcbf-0845-4fe3-8aba-
ee040be542c0
        //connection.setRequestProperty("Authorization", "Bearer
da3efcbf-0845-4fe3-8aba-ee040be542c0");
        // 通过连接对象获取一个输出流
        os = connection.getOutputStream();
        // 通过输出流对象将参数写出去/传出去,它是通过字节数组写出的(form表单形式
的参数实质也是key,value值的拼接,类似于get请求参数的拼接)
        JSONObject json = new JSONObject(param);
        os.write( createLinkString(param).getBytes() );
        // 通过连接对象获取一个输入流, 向远程读取
        if (connection.getResponseCode() == 200) {

            is = connection.getInputStream();
            // 对输入流对象进行包装:charset根据工作项目组的要求来设置
            br = new BufferedReader(new InputStreamReader(is, "UTF-8"));

            StringBuffer sbf = new StringBuffer();
            String temp = null;
            // 循环遍历一行一行读取数据
            while ((temp = br.readLine()) != null) {

```

```

        sbf.append(temp);
        sbf.append("\r\n");
    }
    result = sbf.toString();
}
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // 关闭资源
    if (null != br) {
        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != os) {
        try {
            os.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != is) {
        try {
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // 断开与远程地址url的连接
    connection.disconnect();
}
return result;
}

```

```

public static String doPostForm(String httpUrl, JSONObject param) {

```

```

    HttpURLConnection connection = null;
    InputStream is = null;
    OutputStream os = null;
    BufferedReader br = null;
    String result = null;
    try {
        URL url = new URL(httpUrl);
        // 通过远程url连接对象打开连接

```

```

connection = (URLConnection) url.openConnection();
// 设置连接请求方式
connection.setRequestMethod("POST");
// 设置连接主机服务器超时时间: 15000毫秒
connection.setConnectTimeout(15000);
// 设置读取主机服务器返回数据超时时间: 60000毫秒
connection.setReadTimeout(60000);

// 默认值为: false, 当向远程服务器传送数据/写数据时, 需要设置为true
connection.setDoOutput(true);
// 默认值为: true, 当前向远程服务读取数据时, 设置为true, 该参数可有可无
connection.setDoInput(true);
// 设置传入参数的格式:请求参数应该是 name1=value1&name2=value2 的形式。
connection.setRequestProperty("Content-Type",
"application/json");
// 设置鉴权信息: Authorization: Bearer da3efcbf-0845-4fe3-8aba-
ee040be542c0
//connection.setRequestProperty("Authorization", "Bearer
da3efcbf-0845-4fe3-8aba-ee040be542c0");
// 通过连接对象获取一个输出流
os = connection.getOutputStream();
// 通过输出流对象将参数写出去/传出去,它是通过字节数组写出的(form表单形式
的参数实质也是key,value值的拼接,类似于get请求参数的拼接)

os.write( param.toString().getBytes() );
// 通过连接对象获取一个输入流, 向远程读取
if (connection.getResponseCode() == 200) {

    is = connection.getInputStream();
    // 对输入流对象进行包装:charset根据工作项目组的要求来设置
    br = new BufferedReader(new InputStreamReader(is, "UTF-8"));

    StringBuffer sbf = new StringBuffer();
    String temp = null;
    // 循环遍历一行一行读取数据
    while ((temp = br.readLine()) != null) {
        sbf.append(temp);
        sbf.append("\r\n");
    }
    result = sbf.toString();
}
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // 关闭资源
    if (null != br) {

```

```

        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != os) {
        try {
            os.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != is) {
        try {
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // 断开与远程地址url的连接
    connection.disconnect();
}
return result;
}

```

```

/**
 * 把数组所有元素排序，并按照“参数=参数值”的模式用“&”字符拼接成字符串
 * @param params 需要排序并参与字符拼接的参数组
 * @return 拼接后字符串
 */
public static String createLinkString(Map<String, String> params) {

    List<String> keys = new ArrayList<String>(params.keySet());
    Collections.sort(keys);

    StringBuilder prestr = new StringBuilder();
    for (int i = 0; i < keys.size(); i++) {
        String key = keys.get(i);
        String value = params.get(key);
        if (i == keys.size() - 1) { // 拼接时，不包括最后一个&字符
            prestr.append(key).append("=").append(value);
        } else {
            prestr.append(key).append("=").append(value).append("&");
        }
    }
}

```



```

        return prestr.toString();
    }

    //String 转 Map
    public static Map<String, Object> strToJson(String jsonString) {
        Gson gson = new Gson();
        Map<String, Object> map = new HashMap<String, Object>();
        map = gson.fromJson(jsonString, map.getClass());
        return map;
    }
}

```

SignUp.java

```

package com.weixin;

import com.dao.Dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.*;

public class SignUp {

    /*
        category_value      失物类别
        LostTime             丢失时间
        LostPosition         丢失地点
        LostPerson           联系人
        LostTelNumber        联系方式
        LostThingsPicture    失物照片
        LostMessage          备注
    */
    //create table data_enrol(category_value char(50),LostTime
    char(100),LostPosition char(100),LostPerson char(50),LostTelNumber
    char(50),LostThingsPicture longtext,LostMessage longtext,model char(10));
    //进行报名
    public static boolean Enrol(String category_value, String LostTime,
    String LostPosition, String LostPerson, String LostTelNumber, String
    LostThingsPicture,String LostMessage,String model) {
        //ticket就是准考证号，准考证号就是登录名
    }
}

```

```

try {
    Connection conn = Dao.getConnection();
    /* 对报名的数据库进行补充 */
    PreparedStatement p = conn.prepareStatement("insert into
data_enrol(category_value,LostTime,LostPosition,LostPerson,LostTelNumber,Lost
ThingsPicture,LostMessage,model) VALUES (?,?,?,?,?,?,?,?,?);");
    //对占位符进行补充
    p.setString(1, category_value);
    p.setString(2, LostTime);
    p.setString(3, LostPosition);
    p.setString(4, LostPerson);
    p.setString(5, LostTelNumber);
    p.setString(6, LostThingsPicture);
    p.setString(7, LostMessage);
    p.setString(8, model);
    System.out.println(p);
    p.executeUpdate();
    System.out.println("登记成功");
    Dao.close(p, conn);
    return true;
} catch (Exception e) {
    System.out.println("登记失败");
    e.printStackTrace();
    return false;
}
}

```

```

//查询总的数目
public static int queryNum(){
    int res = 0;
    try {
        Connection conn = Dao.getConnection(); //连接数据库
        PreparedStatement p = conn.prepareStatement("select count(*) from
data_enrol;"); //查看报名的人数
        ResultSet rs = p.executeQuery();
        if(rs.next()){
            res = rs.getInt(1);
            Dao.close(rs, p, conn);
            return res;
        }
        Dao.close(rs, p, conn);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return res;
}

```

//返回全部报名人及信息

```

public static List signupList(){
    int sign = 0;          //标记一下当前页数是否有数据
    //参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
    List list = new ArrayList();
    int res = 0;
    try {
        Connection conn = Dao.getConnection();          //连接数据库
        PreparedStatement p = conn.prepareStatement("select * from
data_enrol;");          //查看报名信息
        ResultSet rs = p.executeQuery();

        while(rs.next()){
            sign = 1;
            //category_value char(50),LostTime char(100),LostPosition
char(100),LostPerson char(50),LostTelNumber char(50),LostThingsPicture
longtext,LostMessage longtext,model char(10));
            Map<String,String> data = new HashMap();

            data.put("category_value", rs.getString("category_value"));
            data.put("LostTime", rs.getString("LostTime"));
            data.put("LostPosition", rs.getString("LostPosition"));
            data.put("LostPerson", rs.getString("LostPerson"));
            data.put("LostTelNumber", rs.getString("LostTelNumber"));
            data.put("LostThingsPicture",
rs.getString("LostThingsPicture"));
            data.put("LostMessage", rs.getString("LostMessage"));
            if( "picker".equals(rs.getString("model")) ){ //说明是拾物
                data.put("model","拾物");
            }else{
                data.put("model","失物");
            }
            list.add(data);
        }
        Dao.close(rs, p, conn);
    } catch (Exception e) {
        e.printStackTrace();
    }
    if( sign == 1 ){
        return list;
    }else{
        return null;
    }
}

//返回部分报名人及信息
public static List signupList(int page,int limit){

```

```

int sign = 0;           //标记一下当前页数是否有数据
//参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
List list = new ArrayList();
int res = 0;
try {
    Connection conn = Dao.getConnection();           //连接数据库
    // page 页码 limit 每页数量
    // select * from TABLE_NAME where ... order by ... limit (page
-1) * limit, limit
    PreparedStatement p = conn.prepareStatement("select * from
data_enrol limit ?,?;");           //查看报名信息
    p.setInt(1, (page -1) * limit);
    p.setInt(2, limit);
    ResultSet rs = p.executeQuery();

    while(rs.next()){
        sign = 1;
        Map<String,String> data = new HashMap();

        data.put("category_value", rs.getString("category_value"));
        data.put("LostTime", rs.getString("LostTime"));
        data.put("LostPosition", rs.getString("LostPosition"));
        data.put("LostPerson", rs.getString("LostPerson"));
        data.put("LostTelNumber", rs.getString("LostTelNumber"));
        data.put("LostThingsPicture",
rs.getString("LostThingsPicture"));
        data.put("LostMessage", rs.getString("LostMessage"));
        if( "picker".equals(rs.getString("model")) ){ //说明是拾物
            data.put("model", "拾物");
        }else{
            data.put("model", "失物");
        }
        list.add(data);
    }
    Dao.close(rs, p, conn);
} catch (Exception e) {
    e.printStackTrace();
}
//System.out.println(list);
if( sign == 1 ){
    return list;
}else{
    return null;
}
}

```

```

//进行查询返回部分报名人及信息
public static List searchList(int page,int limit,String
category_value,String begintime,String endtime,String find){
    int sign = 0;           //标记一下当前页数是否有数据

    //limit ?,?;
    String sql = "select * from data_enrol";
    if( category_value != null && !"".equals(category_value)){
//说明有筛选类别
        sql += (" where category_value='" + category_value+"'");
        if( find != null && !"".equals(find)){
            sql += (" and (LostPosition like '%" + find + "%' or
LostMessage like '%" + find+"%'"));
        }
    }else{
        if( find != null && !"".equals(find)){
            sql += (" where LostPosition like '%" + find + "%' or
LostMessage like '%" + find+"%'"));
        }
    }

    sql += " limit ?,?;";

    //参考链接:
https://blog.csdn.net/weixin\_28779183/article/details/117810091
    List list = new ArrayList();
    int res = 0;
    try {
        Connection conn = Dao.getConnection();           //连接数据库
        // page 页码 limit 每页数量
        // select * from TABLE_NAME where ... order by ... limit (page
-1) * limit, limit
        PreparedStatement p = conn.prepareStatement(sql);           //查看报名
信息

        p.setInt(1,(page -1) * limit);
        p.setInt(2, limit);
        System.out.println(p);
        ResultSet rs = p.executeQuery();
        //从前端或者自己模拟一个日期格式，转为String即可
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");
        //将字符串日期转为日期
        Date datebegin = null,dateend = null;
        int time = 0;
        if( begintime != null && !"".equals(begintime) && endtime != null
&& !"".equals(endtime)){
            datebegin = format.parse(begintime);

```

```

        dateend = format.parse(endtime);
        time = 1;
    }
    while(rs.next()){
        Map<String,String> data = new HashMap();
        Date when = format.parse(rs.getString("LostTime"));
        if( time == 1){
            if(!(when.after(datebegin) && when.before(dateend)) ){
                continue;
            }
        }

        sign = 1;
        data.put("category_value", rs.getString("category_value"));
        data.put("LostTime", rs.getString("LostTime"));
        data.put("LostPosition", rs.getString("LostPosition"));
        data.put("LostPerson", rs.getString("LostPerson"));
        data.put("LostTelNumber", rs.getString("LostTelNumber"));
        data.put("LostThingsPicture",
rs.getString("LostThingsPicture"));
        data.put("LostMessage", rs.getString("LostMessage"));
        if( "picker".equals(rs.getString("model")) ){ //说明是拾物
            data.put("model", "拾物");
        }else{
            data.put("model", "失物");
        }
        list.add(data);
    }
    Dao.close(rs, p, conn);
} catch (Exception e) {
    e.printStackTrace();
}
//System.out.println(list);
if( sign == 1 ){
    return list;
}else{
    return null;
}

}

}

```

applicationContext.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

</beans>

```

dispatcher-servlet.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.2.xsd">

    <!--dispatcher-servlet.xml文件，这个文件主要用来配置HandlerMapping-->
    <!-- 配置自动扫描的包 -->
    <context:component-scan base-package="com.controller">
</context:component-scan>
    <!-- 静态资源(js、image等)的访问 -->
    <mvc:default-servlet-handler/>
    <!-- 开启注解 -->
    <mvc:annotation-driven/>
    <!--      Handlermapper和HandlerAdapter-->

    <!--<bean class="com.test.controller.WelcomeController"></bean>-->
    <!-- 配置视图解析器 如何把handler 方法返回值解析为实际的物理视图 -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
        <property name = "prefix" value="/WEB-INF/views/"></property>
        <property name = "suffix" value = ".jsp"></property>
    </bean>
    <!-- 配置MultipartResolver，用于上传文件，使用spring的
CommonsMultipartResolver -->
    <bean id="multipartResolver"

```

```

class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="5000000" />
    <property name="defaultEncoding" value="UTF-8" />
</bean>

</beans>

```

web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd"
    version="5.0">
    <context-param> <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext.xml</param-value>
    </context-param> <listener> <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
    <filter>
        <filter-name>crossFilter</filter-name>
        <filter-class>com.filter.CrossFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>crossFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <!--web.xml是用来启动DispatcherServlet -->

    <servlet>
        <!--这里写成对应名字-->
        <servlet-name>dispatcher</servlet-name>

        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

        <!-- 启动顺序，数字越小，启动越早 -->
        <load-on-startup>1</load-on-startup>
    </servlet>
    <!--所有请求都会被dispatcher拦截 -->
    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

```

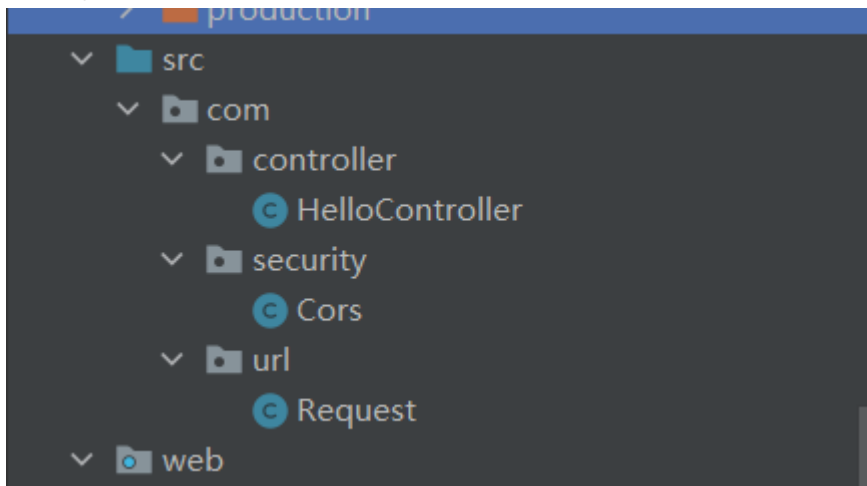


```
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

(2022.3.27)

测试接口版0.1

项目结构



当前功能：

- 1.返回两张图片
- 2.得到用户手机号码

HelloController.java

```
package com.controller;

import com.url.Request;
import org.springframework.http.*;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import com.google.gson.Gson;
```

```

//import net.sf.json.JSONObject;

import com.alibaba.fastjson.JSONObject;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Map;

//使用Controller来标识它是一个控制器
@Controller
//@CrossOrigin
@RequestMapping(value = "/api")
public class HelloController {
    //测试springmvc是否创建成功
    @RequestMapping(value = "/helloworld")
    public String Hello() {
        return "cs";
    }

    /* 测试链接
    * 请求: Get
    * 链接: 地址/api/cs
    * 参数: 无
    * 返回: json {"msg":"HelloWorld"}
    */
    @RequestMapping(value = "/cs")
    public @ResponseBody Map<String,String> cs() {
        Map<String,String> data = new HashMap<String,String>();
        data.put("msg","HelloWorld" );
        return data;
    }

    /*
    * 获取两张图片
    * 请求: Get
    * 链接: 地址/api/getpicture
    * 参数: 无
    * 返回: json
    * {
    *     "msg":"ok",
    *     "picture1":"/img/LostButtonPicture.png",
    *     "picture2":"/img/PickUpButtonPicture.png"
    * }
    */

```

```

    */
@RequestMapping(value = "/getpicture")
public @ResponseBody
Map<String,String> getPicture() {
    Map<String,String> data = new HashMap<String,String>();

    data.put("msg","ok" );
    data.put("picture1","/img/LostButtonPicture.png");
    data.put("picture2","/img/PickUpButtonPicture.png");
    return data;
}

/* 注册
 * 请求: Get
 * 链接: 地址/api/register
 * 参数: 无
 * 返回: json {"msg":"HelloWorld"}
 */
@RequestMapping(value = "/register",method = RequestMethod.POST)
public @ResponseBody
Map<String,String> register(HttpServletRequest request) {
    String code = request.getParameter("code");
    Map<String,String> data = new HashMap<String,String>();

    data.put("msg","ok" );
    return data;
}

/* 获取手机号码
 * 请求: Get
 * 链接: 地址/api/getphone
 * 参数: 无
 * 返回: json {"msg":"HelloWorld"}
 */
@RequestMapping(value = "/getphone",method = RequestMethod.POST)
public @ResponseBody Map<String,String> getPhone(HttpServletRequest
request) {
    Map<String,String> data = new HashMap<String,String>();
    try{
        String code = request.getParameter("code");
        System.out.println(code);

        //Map<String,String> params = new HashMap<String,String>();
        //params.put("chat","草莓");

        //System.out.println(Request.doPostForm("http://****:80/spring_app_war/educat
ion",params));

```

```

        //获取获取小程序全局唯一后台接口调用凭据（access_token）
/* data:
{ appid: "wxf58558547a7b352a", //需要开通企业账号权限 用测试号也行
secret: "c2ca1a3ee37256c970a0d6c4f4e475fc", // 这个测试号也有
grant_type: "client_credential", //这个写死的 },
* */

String getTokenUrl = "https://api.weixin.qq.com/cgi-bin/token?
grant_type=client_credential&appid=wxf58558547a7b352a&secret=c2ca1a3ee37256c9
70a0d6c4f4e475fc";

String jsonStringA = Request.doGet(getTokenUrl);
System.out.println(jsonStringA);

//String转JSON
JSONObject jsonObject = JSONObject.parseObject(jsonStringA);
String access_token = jsonObject.getString("access_token");

//提交参数
String getPhoneUrl =
"https://api.weixin.qq.com/wxa/business/getuserphonenumber?
access_token="+access_token;

Map<String,Object> map = new HashMap<String,Object>();
//map.put("access_token",access_token);
map.put("code",code);
JSONObject json = new JSONObject(map);
System.out.println(map);
System.out.println(json);
String jsonStringB = Request.doPostForm(getPhoneUrl,json);
System.out.println(jsonStringB);

//String转JSON
JSONObject jsonObject2 = JSONObject.parseObject(jsonStringB);

//String转Map
//Map<String,Object> tmp2 = Request.strToJson(jsonStringB);
//String phone_info =(String)tmp2.get("phone_info");
//System.out.println(phone_info);
//data.put("phone_info",phone_info);
HashMap hashMap =
JSONObject.parseObject(jsonObject2.toJSONString(), HashMap.class);
if( 0 == (int)hashMap.get("errcode") ){ //请求成功
    data.put("msg","ok" );
    JSONObject tmp2 = (JSONObject)hashMap.get("phone_info");
    data.put("phone",(String)tmp2.get("phoneNumber"));
}else{

```

```

        data.put("msg", "fail" );
        data.put("error", (int)hashMap.get("errcode")+" " );
    }

    }catch (Exception e){
        System.out.println(e);
        data.put("msg", "fail" );
    }

    return data;
}
}

```

Request.java

```

package com.url;

import com.alibaba.fastjson.JSONObject;
import com.google.gson.Gson;
import com.google.gson.Gson;
import java.io.*;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.text.ParseException;
import java.util.*;

//https://www.cnblogs.com/mufengforward/p/10510337.html

public class Request {
    /**
     * 向指定URL发送GET方法的请求
     *
     * @param httpurl
     *      请求参数用?拼接在url后边，请求参数应该是
name1=value1&name2=value2 的形式。
     * @return result 所代表远程资源的响应结果
     */
    public static String doGet(String httpurl) {
        HttpURLConnection connection = null;
        InputStream is = null;
        BufferedReader br = null;
    }
}

```

```

String result = null;// 返回结果字符串
try {
    // 创建远程url连接对象
    URL url = new URL(httpurl);
    // 通过远程url连接对象打开一个连接，强转成URLConnection类
    connection = (URLConnection) url.openConnection();
    // 设置连接方式: get
    connection.setRequestMethod("GET");
    // 设置连接主机服务器的超时时间: 15000毫秒
    connection.setConnectTimeout(15000);
    // 设置读取远程返回的数据时间: 60000毫秒
    connection.setReadTimeout(60000);
    // 发送请求
    connection.connect();
    // 通过connection连接，获取输入流
    if (connection.getResponseCode() == 200) {
        is = connection.getInputStream();
        // 封装输入流is，并指定字符集
        br = new BufferedReader(new InputStreamReader(is, "UTF-8"));
        // 存放数据
        StringBuffer sbf = new StringBuffer();
        String temp = null;
        while ((temp = br.readLine()) != null) {
            sbf.append(temp);
            sbf.append("\r\n");
        }
        result = sbf.toString();
    }
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // 关闭资源
    if (null != br) {
        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

if (null != is) {
    try {
        is.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

```

    }

    connection.disconnect();// 关闭远程连接
}

return result;
}

/**
 *
 * @param httpUrl 请求的url
 * @param param form表单的参数 (key,value形式)
 * @return
 */
public static String doPostForm(String httpUrl, Map param) {

    HttpURLConnection connection = null;
    InputStream is = null;
    OutputStream os = null;
    BufferedReader br = null;
    String result = null;
    try {
        URL url = new URL(httpUrl);
        // 通过远程url连接对象打开连接
        connection = (HttpURLConnection) url.openConnection();
        // 设置连接请求方式
        connection.setRequestMethod("POST");
        // 设置连接主机服务器超时时间: 15000毫秒
        connection.setConnectTimeout(15000);
        // 设置读取主机服务器返回数据超时时间: 60000毫秒
        connection.setReadTimeout(60000);

        // 默认值为: false, 当向远程服务器传送数据/写数据时, 需要设置为true
        connection.setDoOutput(true);
        // 默认值为: true, 当前向远程服务读取数据时, 设置为true, 该参数可有可无
        connection.setDoInput(true);
        // 设置传入参数的格式:请求参数应该是 name1=value1&name2=value2 的形式。
        connection.setRequestProperty("Content-Type",
"application/json");
        // 设置鉴权信息: Authorization: Bearer da3efcbf-0845-4fe3-8aba-
ee040be542c0
        //connection.setRequestProperty("Authorization", "Bearer
da3efcbf-0845-4fe3-8aba-ee040be542c0");
        // 通过连接对象获取一个输出流
        os = connection.getOutputStream();
        // 通过输出流对象将参数写出去/传输出去,它是通过字节数组写出的(form表单形式
的参数实质也是key,value值的拼接,类似于get请求参数的拼接)
        JSONObject json = new JSONObject(param);

```

```

os.write( createLinkString(param).getBytes() );
// 通过连接对象获取一个输入流，向远程读取
if (connection.getResponseCode() == 200) {

    is = connection.getInputStream();
    // 对输入流对象进行包装:charset根据工作项目组的要求来设置
    br = new BufferedReader(new InputStreamReader(is, "UTF-8"));

    StringBuffer sbf = new StringBuffer();
    String temp = null;
    // 循环遍历一行一行读取数据
    while ((temp = br.readLine()) != null) {
        sbf.append(temp);
        sbf.append("\r\n");
    }
    result = sbf.toString();
}
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // 关闭资源
    if (null != br) {
        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != os) {
        try {
            os.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != is) {
        try {
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // 断开与远程地址url的连接
    connection.disconnect();
}
return result;

```



```
}
```

```
public static String doPostForm(String httpUrl, JSONObject param) {

    HttpURLConnection connection = null;
    InputStream is = null;
    OutputStream os = null;
    BufferedReader br = null;
    String result = null;
    try {
        URL url = new URL(httpUrl);
        // 通过远程url连接对象打开连接
        connection = (HttpURLConnection) url.openConnection();
        // 设置连接请求方式
        connection.setRequestMethod("POST");
        // 设置连接主机服务器超时时间: 15000毫秒
        connection.setConnectTimeout(15000);
        // 设置读取主机服务器返回数据超时时间: 60000毫秒
        connection.setReadTimeout(60000);

        // 默认值为: false, 当向远程服务器传送数据/写数据时, 需要设置为true
        connection.setDoOutput(true);
        // 默认值为: true, 当前向远程服务读取数据时, 设置为true, 该参数可有可无
        connection.setDoInput(true);
        // 设置传入参数的格式:请求参数应该是 name1=value1&name2=value2 的形式。
        connection.setRequestProperty("Content-Type",
"application/json");
        // 设置鉴权信息: Authorization: Bearer da3efcbf-0845-4fe3-8aba-
ee040be542c0
        //connection.setRequestProperty("Authorization", "Bearer
da3efcbf-0845-4fe3-8aba-ee040be542c0");
        // 通过连接对象获取一个输出流
        os = connection.getOutputStream();
        // 通过输出流对象将参数写出去/传出去,它是通过字节数组写出的(form表单形式
的参数实质也是key,value值的拼接,类似于get请求参数的拼接)

        os.write( param.toString().getBytes() );
        // 通过连接对象获取一个输入流, 向远程读取
        if (connection.getResponseCode() == 200) {

            is = connection.getInputStream();
            // 对输入流对象进行包装:charset根据工作项目组的要求来设置
            br = new BufferedReader(new InputStreamReader(is, "UTF-8"));

            StringBuffer sbf = new StringBuffer();
            String temp = null;
            // 循环遍历一行一行读取数据
            while ((temp = br.readLine()) != null) {
```

```

        sbf.append(temp);
        sbf.append("\r\n");
    }
    result = sbf.toString();
}
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} finally {
    // 关闭资源
    if (null != br) {
        try {
            br.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != os) {
        try {
            os.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (null != is) {
        try {
            is.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // 断开与远程地址url的连接
    connection.disconnect();
}
return result;
}

```

```

/**
 * 把数组所有元素排序，并按照“参数=参数值”的模式用“&”字符拼接成字符串
 * @param params 需要排序并参与字符拼接的参数组
 * @return 拼接后字符串
 */
public static String createLinkString(Map<String, String> params) {

    List<String> keys = new ArrayList<String>(params.keySet());

```

```

        Collections.sort(keys);

        StringBuilder prestr = new StringBuilder();
        for (int i = 0; i < keys.size(); i++) {
            String key = keys.get(i);
            String value = params.get(key);
            if (i == keys.size() - 1) { // 拼接时, 不包括最后一个&字符
                prestr.append(key).append("=").append(value);
            } else {
                prestr.append(key).append("=").append(value).append("&");
            }
        }

        return prestr.toString();
    }

    //String 转 Map
    public static Map<String, Object> strToJson(String jsonString) {
        Gson gson = new Gson();
        Map<String, Object> map = new HashMap<String, Object>();
        map = gson.fromJson(jsonString, map.getClass());
        return map;
    }
}

```