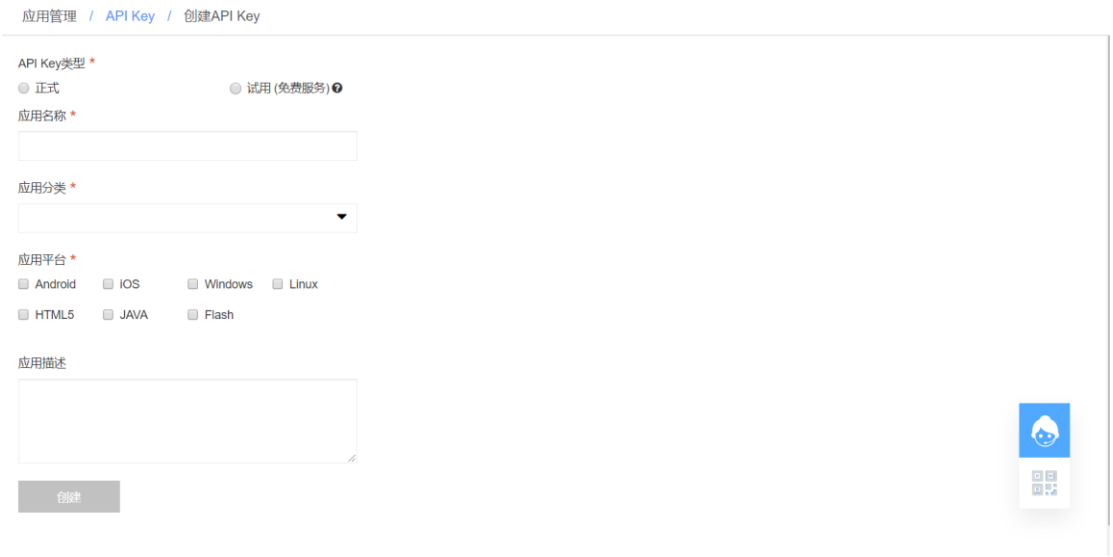
基于 Face++人脸检测以及人脸对比试验

Face++官网：https://www.faceplusplus.com.cn/

前言

我们需要在官网注册一个账号，注册完成后，我们在需要穿件一个 API_KEY



我创建的是一个试用版的。

应用管理 / API Key / HeloWXL的第一个应用

应用名称

HeloWXL的第一个应用

应用分类

应用 / 网站-其他-其他

应用平台

Android iOS Windows Linux

HTML5 JAVA Flash

应用描述

创建时间

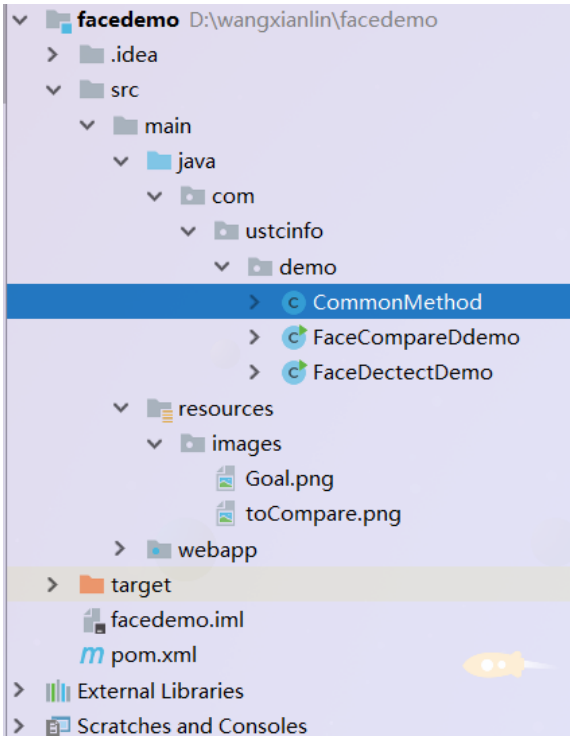
2019-08-31 11:47

API Key

L_mT-oglbWdX2uw20G3tGl5TJqBy9v3L 复制

有钱人的话，那就不多说。

代码示例（官网提供）



公共方法：CommonMethod

```
/**
 * @author wangxl
 * @ClassName CommonMethod
 * @Description TODO
 * @date 2019/8/31 15:08
 * @Version 1.0
 */
public class CommonMethod {
  private final static int CONNECT_TIME_OUT = 30000;
  private final static int READ_OUT_TIME = 50000;
  private static String boundaryString = getBoundary();
  protected static byte[] post(String url, HashMap<String, String> map,
HashMap<String, byte[]> fileMap) throws Exception {
    HttpURLConnection conne;
    URL url1 = new URL(url);
    conne = (HttpURLConnection) url1.openConnection();
    conne.setDoOutput(true);
    conne.setUseCaches(false);
    conne.setRequestMethod("POST");
    conne.setConnectTimeout(CONNECT_TIME_OUT);
    conne.setReadTimeout(READ_OUT_TIME);
```

```
    conne.setRequestProperty("accept", "*/*");
    conne.setRequestProperty("Content-Type", "multipart/form-data; boundary=" +
boundaryString);
    conne.setRequestProperty("connection", "Keep-Alive");
    conne.setRequestProperty("user-agent",     "Mozilla/4.0     (compatible;MSIE
6.0;Windows NT 5.1;SV1)");
    DataOutputStream obos = new DataOutputStream(conne.getOutputStream());
    Iterator iter = map.entrySet().iterator();
    while(iter.hasNext()){
      Map.Entry<String, String> entry = (Map.Entry) iter.next();
      String key = entry.getKey();
      String value = entry.getValue();
      obos.writeBytes("--" + boundaryString + "\r\n");
      obos.writeBytes("Content-Disposition: form-data; name=\"" + key
          + "\"\r\n");
      obos.writeBytes("\r\n");
      obos.writeBytes(value + "\r\n");
    }
    if(fileMap != null && fileMap.size() > 0){
      Iterator fileIter = fileMap.entrySet().iterator();
      while(fileIter.hasNext()){
        Map.Entry<String,   byte[]>   fileEntry   =   (Map.Entry<String,   byte[]>)
fileIter.next();
        obos.writeBytes("--" + boundaryString + "\r\n");
        obos.writeBytes("Content-Disposition:     form-data;     name=\""     +
fileEntry.getKey()
            + "\"; filename=\"" + encode(" ") + "\"\r\n");
        obos.writeBytes("\r\n");
        obos.write(fileEntry.getValue());
        obos.writeBytes("\r\n");
      }
    }
    obos.writeBytes("--" + boundaryString + "--" + "\r\n");
    obos.writeBytes("\r\n");
    obos.flush();
    obos.close();
    InputStream ins = null;
    int code = conne.getResponseCode();
    try{
      if(code == 200){
        ins = conne.getInputStream();
      }else{
        ins = conne.getErrorStream();
      }
```

```
    }catch (SSLException e){
      e.printStackTrace();
      return new byte[0];
    }
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    byte[] buff = new byte[4096];
    int len;
    while((len = ins.read(buff)) != -1){
      baos.write(buff, 0, len);
    }
    byte[] bytes = baos.toByteArray();
    ins.close();
    return bytes;
  }
  private static String getBoundary() {
    StringBuilder sb = new StringBuilder();
    Random random = new Random();
    for(int i = 0; i < 32; ++i) {
sb.append("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_-".ch
arAt(random.nextInt("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345
6789_".length())));
    }
    return sb.toString();
  }
  private static String encode(String value) throws Exception{
    return URLEncoder.encode(value, "UTF-8");
  }
  public static byte[] getBytesFromFile(File f) {
    if (f == null) {
      return null;
    }
    try {
      FileInputStream stream = new FileInputStream(f);
      ByteArrayOutputStream out = new ByteArrayOutputStream(1000);
      byte[] b = new byte[1000];
      int n;
      while ((n = stream.read(b)) != -1) {
        out.write(b, 0, n);
      }
      stream.close();
      out.close();
      return out.toByteArray();
    } catch (IOException e) {
    }
```

```
    return null;
  }
}
```

人脸检测(FaceDectectDemo)

```
/**
 * @author wangxl
 * @ClassName FaceDectectDemo
 * @Description TODO
 * @date 2019/8/31 14:05
 * @Version 1.0
 */
public class FaceDectectDemo {
  //  API 的API_KEY
  public static final String API_KEY = "L_mT-oglbWdX2uw20G3tGl5TJqBy9v3L";
  // API 的API_SECRET
  public static final String API_SECRET = "**************************";
  public static final String IMG_PATH =
"D:\\wangxianlin\\facedemo\\src\\main\\resources\\images\\Goal.png";
  public static void main(String[] args) throws Exception{
    File file = new File(IMG_PATH);
    byte[] buff = getBytesFromFile(file);
    /**
     * Url
     * 必须要传递的参数
     */
    String url = "https://api-cn.faceplusplus.com/facepp/v3/detect";
    /**
     * 存放  api_key ,  api_secret
     */
    HashMap<String, String> map = new HashMap<>();
    /**
     * 存放 image_file  因为它是一个二进制数组
     */
    HashMap<String, byte[]> byteMap = new HashMap<>();

    map.put("api_key", API_KEY);
    map.put("api_secret", API_SECRET);
    /**
     * return_landmark
     * 是否检测并返回人脸关键点
     * 参数:
```

```
     * 0 :  不检测
     * 1 : 检测 返回83 个人脸检测点
     * 2 : 检测 返回 106 个人脸监测点
     */
    map.put("return_landmark", "1");
    /**
     * return_attributes
     * 是否检测并返回根据人脸特征判断出的年龄、性别、情绪等属性。
     */
    map.put("return_attributes",
"gender,age,smiling,headpose,facequality,blur,eyestatus,emotion,ethnicity,beaut
y,mouthstatus,eyegaze,skinstatus");
    map.put("beauty_score_min","0");
    map.put("beauty_score_max","100");
    /***
     * image_file
     * 一个图片，二进制文件，需要用post multipart/form-data 的方式上传
     */
    byteMap.put("image_file", buff);
    try{
      byte[] bacd = post(url, map, byteMap);
      String str = new String(bacd);
      System.out.println(str);
    }catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

返回值

是一个比较长的字符串
以我的照片为例：只标注了一部分，具体的可以去官网进行查看

```
{
  //整个请求所花费的时间，单位为毫秒。
  "time_used": 215,
    //被检测出的人脸数组
    "faces": [{
  //人脸属性特征
  "attributes": {
    //情绪识别结果。
    "emotion": {
      "sadness": 0.002, //伤心
          "neutral": 99.935, //平静
```

```
        "disgust": 0.002, //厌恶
        "anger": 0.01, //愤怒
        "surprise": 0.02, //惊讶
        "fear": 0.003, //恐惧
        "happiness": 0.028 //高兴
    },
    //颜值识别结果
    "beauty": {
      "female_score": 63.866, //男性认为的此人脸颜值分数。值越大，颜值越高。
        "male_score": 61.117 //女性认为的此人脸颜值分数。值越大，颜值越高。
    },
    //性别分析结果
    "gender": {
      "value": "Male"
    },
    //年龄分析结果
    "age": {
      "value": 20
    },
    //嘴部状态信息
    "mouthstatus": {
      "close": 0.001, //嘴部没有遮挡且闭上的置信度
        "surgical_mask_or_respirator": 0.0, //嘴部被医用口罩或呼吸面罩遮挡的置信度
        "open": 0.0, //嘴部没有遮挡且张开的置信度
        "other_occlusion": 99.999 //嘴部被其他物体遮挡的置信度
    },
    //是否佩戴眼镜的分析结果
    "glass": {
      "value": "Normal" //佩戴普通眼镜
    },
    //面部特征识别结果  每个字段的值都是一个浮点数，范围 [0,100]
    "skinstatus": {
      "dark_circle": 1.038, //黑眼圈
        "stain": 3.417, //色斑
        "acne": 0.361, //青春痘
        "health": 1.772 //健康
    },
    //人脸姿势分析结果
    "headpose": {
      "yaw_angle": 0.6436087, //摇头
        "pitch_angle": -4.9861484, //抬头
        "roll_angle": 9.328366 //旋转（平面旋转）
    },
    //人脸模糊分析结果
```

```
"blur": {
  //人脸模糊分析结果
  "blurness": {
    "threshold": 50.0,
        "value": 10.062
  },
  "motionblur": {
    "threshold": 50.0,
        "value": 10.062
  },
  "gaussianblur": {
    "threshold": 50.0,
        "value": 10.062
  }
},
//笑容分析结果
"smile": {
  "threshold": 50.0, //代表笑容的阈值，超过该阈值认为有笑容。
     "value": 7.99 //数值越大表示笑程度高。
},
//眼睛状态信息
"eyestatus": {
  //左眼的状态
  "left_eye_status": {
    "normal_glass_eye_open": 99.945, //佩戴普通眼镜且睁眼的置信度
        "no_glass_eye_close": 0.0, //不戴眼镜且睁眼的置信度
        "occlusion": 0.0, //眼睛被遮挡的置信度
        "no_glass_eye_open": 0.055, //不戴眼镜且闭眼的置信度
        "normal_glass_eye_close": 0.0, //佩戴普通眼镜且闭眼的置信度
        "dark_glasses": 0.0 //佩戴墨镜的置信度
  },
  //右眼的状态
  "right_eye_status": {
    "normal_glass_eye_open": 99.622,
        "no_glass_eye_close": 0.0,
        "occlusion": 0.004,
        "no_glass_eye_open": 0.321,
        "normal_glass_eye_close": 0.048,
        "dark_glasses": 0.005
  }
},
//人脸质量判断结果
"facequality": {
  "threshold": 70.1, //表示人脸质量基本合格的一个阈值，超过该阈值的人脸适合用于人
```

```
脸比对。
          "value": 90.462 //值为人脸的质量判断的分数
    },
    //人种分析结果
    "ethnicity": {
      "value": "ASIAN" //亚洲人
    },
    //眼球位置与视线方向信息
    "eyegaze": {
      "right_eye_gaze": {
        "position_x_coordinate": 0.462,
            "vector_z_component": 0.947,
            "vector_x_component": -0.188,
            "vector_y_component": 0.26,
            "position_y_coordinate": 0.444
      },
      "left_eye_gaze": {
        "position_x_coordinate": 0.539,
            "vector_z_component": 0.93,
            "vector_x_component": 0.222,
            "vector_y_component": 0.292,
            "position_y_coordinate": 0.476
      }
    }
  },
  "face_rectangle": {
    "width": 79,
        "top": 53,
        "left": 28,
        "height": 79
  },
  "face_token": "546bdae807c8fc671e310785d9307401"
}],
  "image_id": "1FCyTJaK26tRfLMF26JL9Q==",
    "request_id": "1567233743,c23d1c89-b73c-4cd3-86aa-a71362d06402",
    "face_num": 1
}
```

人脸对比(FaceCompareDdemo)

```
/**
 * @author wangxl
```

```
 * @ClassName FaceCompareDdemo
 * @Description TODO
 * @date 2019/8/31 15:00
 * @Version 1.0
 */
public class FaceCompareDdemo {
  //  API 的API_KEY
  public static final String API_KEY = "L_mT-oglbWdX2uw20G3tGl5TJqBy9v3L";
  // API 的API_SECRET
  public static final String API_SECRET = "jaoj6KDENsq7JcgUL8mD-tMQVSMZRFlX";
  public static final String IMG_PATH =
"D:\\wangxianlin\\facedemo\\src\\main\\resources\\images\\Goal.png";
  public static final String IMG_PATH_TO_COMPARE =
"D:\\wangxianlin\\facedemo\\src\\main\\resources\\images\\toCompare.png";
  public static void main(String[] args) {
    File file1 = new File(IMG_PATH);
    byte[] buff1 = getBytesFromFile(file1);
    File file2 = new File(IMG_PATH_TO_COMPARE);
    byte[] buff2 = getBytesFromFile(file2);
    /**
     * Url
     * 必须要传递的参数
     */
    String url = "https://api-cn.faceplusplus.com/facepp/v3/compare";
    /**
     * 存放  api_key ，  api_secret
     */
    HashMap<String, String> map = new HashMap<>();
    /**
     * 存放 image_file  因为它是一个二进制数组
     */
    HashMap<String, byte[]> byteMap = new HashMap<>();

    map.put("api_key", API_KEY);
    map.put("api_secret", API_SECRET);
    /***
     * image_file
     * 一个图片，二进制文件，需要用post multipart/form-data 的方式上传
     */
    byteMap.put("image_file1", buff1);
    byteMap.put("image_file2", buff2);
    try{
      byte[] bacd = post(url, map, byteMap);
      String str = new String(bacd);
```

```
      System.out.println(str);
    }catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

返回值：

```
{
  //通过 image_url1、image_file1 或 image_base64_1 传入的图片中检测出的人脸数组，采用数组中的第一个人脸进行人脸比对。
  "faces1": [{
  //
  "face_rectangle": {
    "width": 79,
        "top": 53,
        "left": 28,
        "height": 79
  },
  "face_token": "ac2a0e45801f71f305a69d5457c1fdd2"  //人脸的标识
}],
  //通过 image_url2、image_file2 或 image_base64_2 传入的图片中检测出的人脸数组，采用数组中的第一个人脸进行人脸比对。
  "faces2": [{
  "face_rectangle": {
    "width": 47,
        "top": 46,
        "left": 76,
        "height": 47
    //人脸矩形框的位置，包括以下属性。每个属性的值都是整数：
    top：矩形框左上角像素点的纵坐标
    left：矩形框左上角像素点的横坐标
    width：矩形框的宽度
    height：矩形框的高度
  },
  "face_token": "c1e21e719d98d3752f2931dc391b2c0c" //人脸的标识
}],
  "time_used": 499,
    //一组用于参考的置信度阈值，包含以下三个字段。
    "thresholds": {
  "1e-3": 62.327, //误识率为千分之一的置信度阈值；
      "1e-5": 73.975, //误识率为万分之一的置信度阈值；
      "1e-4": 69.101  //误识率为十万分之一的置信度阈值；
```

```
  //如果置信值(confidence)低于“千分之一”阈值则不建议认为是同一个人；如果置信值超过“十万分之一”阈值，则是同一个人的几率非常高。

},
  "confidence": 83.324, //比对结果置信度，范围 [0,100]，小数点后 3 位有效数字，数字越大表示两个人脸越可能是同一个人。

    "image_id2": "UuDIPlFB8a6zLU7NMCsMbA==", //通过 image_url1、image_file1 或 image_base64_1 传入的图片在系统中的标识。
    "image_id1": "1FCyTJaK26tRfLMF26JL9Q==", //通过 image_url2、image_file2 或 image_base64_2 传入的图片在系统中的标识。
    "request_id": "1567235504,fba193bd-b9fd-4a6f-b434-5242d2840f30"
}
```