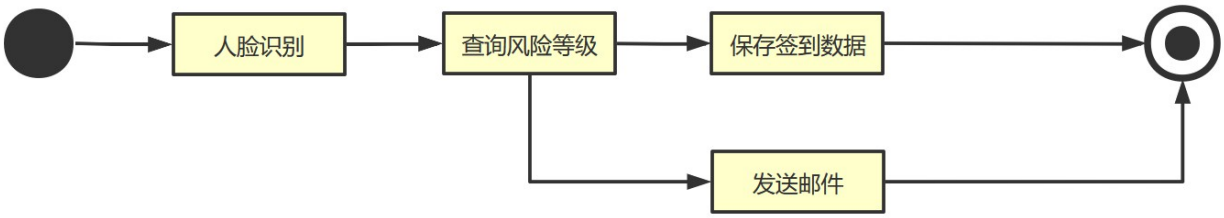
4-20 发送疫情高风险地区告警邮件

一、为什么要采用异步发送邮件？

因为在签到过程中，执行人脸识别和查询疫情风险等级，都比较消耗时间。如果发送邮件再做成同步执行的，势必导致签到执行时间过长，影响用户体验。由于要把签到结果保存到签到表，所以人脸识别和疫情风险等级查询必须是同步执行的。发送邮件跟保存签到数据没有直接关联，所以做成异步并行执行的程序更好一些，这样也能缩短用户签到时候等待的时间。



二、导入Email邮件库

编辑 `pom.xml` 文件，添加依赖库

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

三、设置SMTP服务器信息

发送邮件是通过SMTP服务器来完成的，所以我们要配置一下SMTP服务器的连接信息。这里我以163的SMTP服务器为例，并且提前已经开启了163邮箱的SMTP功能。

```
spring:
    ……
    mail:
        default-encoding: UTF-8
        host: smtp.163.com
        username: **************@163.com
        password: 此处是密码
```

接下来我们把系统内的常用邮箱声明一下，以后会用到这些邮箱往外发送邮件，或者给这些邮箱发送内部邮件。例如，员工签到地点是疫情高风险地区，那么就应该向HR邮箱发送邮件，告知人事总监有员工需要隔离。

```
emos:
    ……
    email:
        system: *********@163.com
        hr: **********@qq.com
```

二、实现异步发送邮件

在SpringBoot项目中开启异步多线程非常简单，只需要下面几个步骤即可。

在主类上面开启 `@EnableAsync` 注解

```
1. ……
2. @EnableAsync
3. public class EmosWxApiApplication {
4.     ……
5. }
```

在 `com.example.emos.wx.config` 中创建 `ThreadPoolConfig` 类，声明Java线程池

```
1. @Configuration
2. public class ThreadPoolConfig {
3.     @Bean("AsyncTaskExecutor")
4.     public AsyncTaskExecutor taskExecutor() {
5.         ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
6.         // 设置核心线程数
7.         executor.setCorePoolSize(8);
8.         // 设置最大线程数
9.         executor.setMaxPoolSize(16);
10.        // 设置队列容量
11.        executor.setQueueCapacity(32);
12.        // 设置线程活跃时间（秒）
13.        executor.setKeepAliveSeconds(60);
14.        // 设置默认线程名称
15.        executor.setThreadNamePrefix("task-");
16.        // 设置拒绝策略
17.        executor.setRejectedExecutionHandler(new ThreadPoolExecutor.CallerRunsPolicy());
18.        executor.initialize();
19.        return executor;
20.    }
21. }
```

在 `com.example.emos.wx.task` 中创建 `EmailTask` 类，定义线程任务

```
1. @Component
2. @Scope("prototype")
3. public class EmailTask implements Serializable {
4.     @Autowired
5.     private JavaMailSender javaMailSender;
6.     @Value("${emos.email.system}")
7.     private String mailbox;
8.     @Async
9.     public void sendAsync(SimpleMailMessage message){
10.        message.setFrom(mailbox);
11.        javaMailSender.send(message);
12.    }
13. }
```

查询员工的姓名和部门名称，在`TbUserDao.xml`文件中声明查询语句

```
<select id="searchNameAndDept" parameterType="int" resultType="HashMap">
    SELECT u.name, d.dept_name
    FROM tb_user u LEFT JOIN tb_dept d ON u.dept_id=d.id
    WHERE u.id = #{userId} AND u.status = 1
</select>
```

在`TbUserDao`接口中定义抽象方法

```
public HashMap searchNameAndDept(int userId);
```

定义值注入变量，用来接收人员隔离告警邮件

```
@Value("${emos.email.hr}")
private String hrEmail;
@Autowired
private EmailTask emailTask;
```

编写发送告警邮件的代码

```
HashMap<String, String> map = userDao.searchNameAndDept(userId);
String name = map.get("name");
String deptName = map.get("dept_name");
deptName = deptName != null ? deptName : "";
SimpleMailMessage message = new SimpleMailMessage();
message.setTo(hrEmail);
message.setSubject("员工" + name + "身处高风险疫情地区警告");
message.setText(deptName + "员工" + name + "，" + DateUtil.format(new Date(), "yyyy年MM月dd日") + "处于" + address + "，属于新冠疫情高风险地区，请及时与该员工联系，核实情况！");
emailTask.sendAsync(message);
```