@Component

**public** **class** AutoReport {

**public** **static** String *LOGIN\_URL* = "http://106.12.195.29/yqsb/login/in";

**public** **static** String *USER\_INFO\_URL* = "http://106.12.195.29/yqsb/user/info";

**public** **static** String *SAVE\_URL* = "http://106.12.195.29/yqsb/jkdj/save";

**private** **static** MediaType *JSON* = MediaType.*parse*("application/json; charset=utf-8");

@Autowired

UserDao userDao;

@Autowired

ReportLogDao reportLogDao;

**private** String token;

@Scheduled(cron = "0 0 0 \* \* ?")

**public** **void** clean() {

System.***out***.println("用户状态已刷新");

userDao.findAll().forEach((user) -> {

user.setState("待填报");

userDao.save(user);

});

}

@Scheduled(cron = "0 0,30 1-9 \* \* ?")

**public** **void** report() {

List<User> userList = userDao.findAll(); // 获取所有用户

OkHttpClient client = **new** OkHttpClient.Builder().build(); // 建立一个http客户端

userList.forEach((user) -> { // 遍历所有的用户

**if** (user.getState().equals("成功")||user.getState().equals("疑似多次填报")) {

**return**;

}

**long** sleepTime = (**long**) (1000 \* (Math.*random*() \* 60 \* 5));

System.***out***.println(**new** Date() + "用户：" + user.getName() + " 正在等待，时长：" + sleepTime / 1000 + "秒");

**try** {

Thread.*sleep*(sleepTime);

} **catch** (InterruptedException e1) {

e1.printStackTrace();

}

Request request = **new** Request.Builder() // 新建一个请求

.url(*LOGIN\_URL* + "?zh=" + user.getName() + "&&" + "mm=" + user.getPassword()).get().build(); // 设置请求地址与请求参数

client.newCall(request).enqueue(**new** Callback() { // 新会话

@Override

**public** **void** onResponse(Call call, Response response) **throws** IOException { // 会话成功回调

System.***out***.println(**new** Date() + "用户：" + user.getName() + " 已登录");

String jsonString = response.body().string(); // 获取json字符串

token = **new** ObjectMapper().readTree(jsonString).get("token").asText(); // 获取结果中token的值

Request request = **new** Request.Builder().url(*USER\_INFO\_URL*).get().addHeader("Authorization", token)

.addHeader("Cookie", "token=" + token).addHeader("token", token).build(); // 用获取到的token发起新的请求

client.newCall(request).enqueue(**new** Callback() { // 新会话

@Override

**public** **void** onResponse(Call call, Response response) **throws** IOException { // 获取用户信息成功

System.***out***.println(**new** Date() + "用户：" + user.getName() + " 成功获取个人信息");

Map readValue = **new** ObjectMapper().readValue(response.body().string(), Map.**class**);

Float randValue = (**float**) (user.getTempreature()

+ (Math.*random*() \* user.getRandomValue() \* 2) - user.getRandomValue());

Map reportData = autoUpdateForm(readValue, **new** DecimalFormat(".0").format(randValue));

String reportString = **new** ObjectMapper().writeValueAsString(reportData);

Request request = **new** Request.Builder().url(*SAVE\_URL*)

.post(RequestBody.*create*(*JSON*, reportString)).addHeader("Authorization", token)

.addHeader("Cookie", "token=" + token).addHeader("token", token).build(); // 用获取到的token发起新的请求

client.newCall(request).enqueue(**new** Callback() {

@Override

**public** **void** onResponse(Call call, Response response) **throws** IOException {

System.***out***.println(**new** Date() + "用户：" + user.getName() + " 报表填报完成");

String respString = response.body().string();

System.***out***.println("userName:" + user.getName() + "\treport:" + respString);

Boolean result = (Boolean) **new** ObjectMapper().readValue(respString, Map.**class**)

.get("success");

**if** (result) {

user.setState("成功");

userDao.save(user);

ReportLog log = **new** ReportLog();

log.setTime(**new** java.sql.Date(System.*currentTimeMillis*()));

log.setReportTemp(randValue);

log.setUserName((String) reportData.get("xm"));

reportLogDao.save(log);

} **else** {

user.setState("疑似多次填报");

userDao.save(user);

}

}

@Override

**public** **void** onFailure(Call call, IOException e) {

System.***out***.println(**new** Date() + "用户：" + user.getName() + " 报表填报异常");

user.setState("异常");

}

});

}

@Override

**public** **void** onFailure(Call call, IOException e) {

System.***out***.println(**new** Date() + "用户：" + user.getName() + " 获取个人信息异常");

user.setState("异常");

}

});

}

@Override

**public** **void** onFailure(Call call, IOException e) {

System.***out***.println(**new** Date() + "用户：" + user.getName() + " 登录异常");

user.setState("异常");

}

});

});

}

**public** Map autoUpdateForm(Map map, String tempreature) {

String[] removeItem = { "jbmc", "ssh", "fdyxm", "smc", "fymc", "sqmc", "sfzh", "sfgz", "bjmc", "jchbrsj", "nl",

"sfzy", "sfsw", "szyy", "kngrtj", "jtxx", "tjsj", "sfjcwhr", "jcwhrsj", "bszz" };

String[][] renameItem = { { "fdyh", "fdygh" }, { "sfmc", "jtzz" } };

String[][] reValueItem = { { "qtstyc", "" }, { "fxrq", "" }, { "jtgj", "" }, { "cc", "" }, { "mqsxdz", "" },

{ "fxstzk", "" }, { "dzsj", "" }, { "ecrctw", "" }, { "ectjsj", "" }, { "dttjcs", "1" },

{ "rctw", tempreature } };

String[][] addItem = { { "mqsxzd", "" }, { "jtdz", "" }, { "sfjz", "0" }, { "sfjcwhry", "0" } };

LinkedHashMap data = (LinkedHashMap) map.get("data");

**for** (**int** i = 0; i < removeItem.length; i++) {

data.remove(removeItem[i]);

}

**for** (**int** i = 0; i < renameItem.length; i++) {

Object temp = data.get(renameItem[i][0]);

data.remove(renameItem[i][0]);

data.put(renameItem[i][1], temp);

}

**for** (**int** i = 0; i < reValueItem.length; i++) {

data.replace(reValueItem[i][0], reValueItem[i][1]);

}

**for** (**int** i = 0; i < addItem.length; i++) {

data.put(addItem[i][0], addItem[i][1]);

}

**return** data;

}

}