链滴

微信扫码登录实现原理

作者：guobing

原文链接：https://ld246.com/article/1470883150755

来源网站：链滴

我最近的很多工作围绕着微信公众号的开发进行的。对微信公众号开发的很多细节都比较清楚。扫码录也基本是首选的登录方式。这里我就记录一下我们的微信扫码登录时怎么实现的。
我们是用websocket来实现的。大体思路就是
>* 前端请求登录二维码的时候传token参数，后端保存token到数据库
>* 手机扫码。后台判断扫码事件的参数是否合法，然后通过websocket把验证结果发送给客户端
>* 客户端如果收到成功消息。则向服务端发送一条登录成功的确认消息。
>* 服务端收到之后，进行用户登录状态的处理，session的保存,把处理结果返回给客户端
>* 客户端再次收到成功标志之后，做登录跳转。标志着登录成功。
所有异常结果都由前端提示操作失败。
##1. 进入登录页面，前端向后台请求二维码
在扫码登录页面，前端向后台发个Ajax请求，请求扫描登录二维码。这里需要一个传一个请求参数token，因为后面要用它进行验证。
token可用时间戳+随机数获取
```
var token = '' + new Date().getTime() + '_' + Math.floor(Math.random() * 1000);
```

后台接收到token后，要保存到数据库。后面验证需要。二维码的接口写法当然有很多种。这是我的法(方法细节就不列出来了)，groovy代码：
```
// 场景二维码 当用户未关注时弹出
handler.get('/ih/bpv2/get-login-qrcode'){req,resp->
    String token = req.params.token
    byte[] b = BpAddService.getFocusQr(token)
    if(!b){
        log.info('ih bp qr generate error')
        resp.statusCode = 404
        resp.end ''
        return
    }

    AuthService.addToken(token)
    resp.headers.set('Content-type', 'image/jpg')
    resp.end(new Buffer(b))
}

```
返回的直接是一张图片，所以替换img src即可。
前端调用二维码的方式为
```
$('#qr').attr('src', '/ih/bpv2/get-login-qrcode?token=' + token);
```
##2. 通过websocket向后台接口发送token
用sockjs实现,前端写法为
```
sock = new SockJS(sockServerUrl);
        sock.onopen = function(){
            console.log('open');
            send({action: 'wx-auth-login-mapping-token-4-bpv2', token: tokenLast});
        };
```
后端接口接收到之后的处理方式就是记录登录方的`ip`，`token`和`sockId`信息,放在缓存`Redis`中
详细代码如下：
```
```

```
// client id binding with token，现在是一台机器
handlerSockjs.add('wx-auth-login-mapping-token-4-bpv2'){sock, json ->
    if(!json.token)
        return [flag: false, msg: 'no token']

    String ipThis = IPUtils.getFirstNoLoopbackAddress()
    String sp = '--'

    def ins = JedisStore.getInstance(config.redisHost)
    ins.set('WX_AUTH_TOKEN_SOCKID', json.token, ipThis + sp + sock.writeHandlerID)
    [flag: true]
}
```

##3. 手机扫码登录
这个二维码是微信的场景二维码，手机扫码事件会被微信接收到，然后推送到我们的服务器。推送过的参数就有用户的openid，我们再从场景二位码的场景值中用正则匹配出token。拿这个token，和据库的对比，看是不是合法的。如已经登录过。伪造token等情况。不合法的时候直接返回错误信息如果合法则用websocket向客户端发送一条消息，告诉客户端扫码验证成功了。客户端收到之后就是码真正成功的标志。客户端再需要向服务端发送一条消息确认成功。准备去登录成功的页面。服务端做一次判断。添加用户登录的cookie等信息。返回给客户端成功消息的时候，客户端才算是真正成功做跳转。

后端对微信扫码登录的处理逻辑如下：
```
// 扫描二维码,发送模板消息和生成名片的连接
WxMessager.inst.add('event', /ih\-bpv2\-focus\-qrcode\-.+/){Map<String, String> info ->
    String openid = info.FromUserName
    Map user = WxUserLocal.getUserByOpenid(openid)
    if(!user)
        return

    // 场景值
    String pre = 'ih-bpv2-focus-qrcode-'
    int start = pre.size()
    String token = info.EventKey[start..-1]

    log.info('ih bp login token - ' +  openid + ' - ' + token)

    // check token
    def tokenRow = AuthService.getTokenRow(token)
    if(!tokenRow || tokenRow.status != 0){
        return
    }
    int tokenRowId = tokenRow.id
    AuthService.updateTokenRow(tokenRowId, Constant.STATUS_YES, openid)

    // broadcast
    def ins = JedisStore.getInstance(config.redisHost)
    String ipAndSockid = ins.get('WX_AUTH_TOKEN_SOCKID', token)
    if(!ipAndSockid){
        log.info('broadcast not found sock id - ' + token)
        return
    }
```

```
    log.info('broadcast found sock id - ' + ipAndSockid)
    String sp = '--'
    String[] arr = ipAndSockid.split(sp)
    String targetIp = arr[0]
    String sockid = arr[1]

    String ipThis = IPUtils.getFirstNoLoopbackAddress()
    if(targetIp == ipThis){
        handlerSockjs.send(sockid, [action: 'loginok'])
    }else{
        int port = 80
        String url = 'http://' + targetIp + ':' + port + '/ih/bpv2/front/broadcast'
        log.info('front user login redirect -  ' + url)

        Map params = [:]
        params.sockid = sockid
        String body = JSON.toJSONString(params)
        try{
            def request = HttpRequest.post(url).connectTimeout(1000 * 2).send(body)
            String str = request.body()
            log.info('front user login redirect result ' + str)
            if(!str){
                log.info('front user login redirect by http return blank ' + sockid)
            }else{
                def resultSend = JSON.parse(str)
                if(resultSend && resultSend.flag){
                    log.info('front user login redirect by http return ok ' + sockid)
                }else{
                    log.info('front user login redirect by http return error ' + sockid)
                }
            }
        }catch(e){
            log.error('front user login redirect error - ' + sockid, e)
        }
    }


    String inner = ''
    String loginTime = new Date().format('yyyy-MM-dd HH:mm:ss')
    // 您与${loginTime}登录了xxxx平台

    Map cardSaved = new CardRefactorService().getCardInfoByOpenid(openid)
    boolean isCardFill = cardSaved && cardSaved.name
    if(!isCardFill) {
        String redirectUrl = '/ih/comm/index?hash=/card-edit'
        String url = WxApi.getLoginRedirectUrl(redirectUrl)
        inner = "<a href=\"${url}\">点击此处生成你的名片，方便BP审核通过后联系你</a>"
    }

    String c = """欢迎使用xxxxx功能
${inner}"""
    [MsgType: 'text', Content: c]
}
```

```

前端处理逻辑：
```
sock.onmessage = function(e){
            console.log(e.data);
            if(!e.data)
                return;

            var obj = JSON.parse(e.data);
            if(obj && 'reject' == obj.action){
                console.log('登陆失败');
                return;
            }

            if(obj && 'loginok' == obj.action){
                $.get('/ih/bpv2/front/login/done?token=' + tokenLast, function(data){
                    var errs = {
                        'token-invalid': '令牌时效'
                    };
                    if(data.error){
                        alert(errs[data.error]);
                        return;
                    }

                    document.location.href = '/bp_20/detail-list.html';
                /*   if(data.isAdmin){
                        document.location.href = '/bp/manage.html';
                    }else{
                    }*/
                });
                return;
            }
        };
```

后台最后一次验证的逻辑
```
// 用websocket登陆成功后
handler.get('/ih/bpv2/front/login/done'){req, resp ->
    String token = req.params.token
    if(!token){
        resp.json([error: 'token-invalid'])
        return
    }

    def tokenRow = AuthService.getTokenRow(token)
    if(!tokenRow || tokenRow.status != 1){
        resp.json([error: 'token-invalid'])
        return
    }

    String openid = tokenRow.openid
    resp.addCookie([name: 'user-login-openid', value: openid, path: '/'])
```
```

```
    resp.addCookie([name: 'user-login-id', value: '', path: '/'])
    BpAddService.addRole([openid: openid, roleId: BpConstant.BP_USER_ROLE_GENERAL_USER
)

    resp.json([flag: true, info: 'login success'])
}
```

至此，所有流程走完。