# 微信小程序保持会话session

一般我们web网站都会有cookie来保存session ID，将用户和服务器保持在一次会话中，但是很遗憾，微信小程序不支持cookie，他的每一次请求就是一次会话，这样就会产生一个问题，每次请求都需要确定当前的用户是谁，但是我们又不能在每次请求的数据中携带用户的信息，这样是不安全的。今天就介绍两种方式来实现保持会话。

第一种：客户端保存session ID

1、通过wx.login接口获取 code，将code传递到后台（一般后台都有shiro或者Spring security这种过滤器，该方法作为第一个启动方法需要后台放过），后台通过code访问微信接口，返回当前登陆的微信信息。

```
//获取session
getSession:  function  ( )  {
    var that = this;
    wx.login({
        success: function(res){
            if(res.code){
                wx.request({
                    url: this.webUrl + '/',
                    data:{
                        code: res.code
                    },
                    success: function(res){
                        wx.setStorageSync( "sessionId", res.sessionId);
                    },
                    fail: function (errMsg) {
                        console.log(errMsg);
                    }
                })
            }else {
                console.log('登录失败!' + res.errMsg)
            }
        }
   });
},
```

2、后台通过微信的openID去数据库的微信信息表中查询该微信号是否与后台用户绑定过，如果绑定过，将该用户信息放入session中，并返回已绑定标记，为绑定过，返回为绑定标记。并且将会话ID返回。

```
@Override
public String  findOpenid() {
    if (StringUtils.isBlank(code)) return "";
    String url = "https://api.weixin.qq.cam/sns/jscade2sessian?appid=" + Constant.XCX_OPENID +
"&secret="+Constant.XCX_SECRET+"&js_code="+code+"&grant_type=authorization_code"
    String result = HttpRequestUtil.get(url);
    if (StringUtils.isBlank(result)) return "";
    JSONObject jsonObject = JSON.parseObject(result);
    String openid = jsonObject.getString("openid");
    if( StringUtils.isBlank(openid)) return "";
    return openid;
```

```
//通过openid判断用户是否绑定
Wxinfo wxinfo = wxinfoService.selectOne(new EntityWrapper<Wxinfo>
().eq("del_flag",0).eg("openid",openid));
if (wxinfo!=null){
    Person person = personService.selectOne(new EntitylVrapper<Person>
().eq("del_fag",0).eq("wxinfo_id",wxinfo.getId()));
    if (person!=null){
        json.put("bind", true);
    }
}
```

3、小程序拿到sessionID和是否绑定标记，将session ID存入缓存中或者在app.js中定一个全局变量用来存储。如果未绑定，跳转到绑定页面，如果绑定，去往首页。

4、因为后台的session都有过期时间，默认是半个小时，所以说为了确保小程序端的session ID不会过期，我们需要在他过期之前刷新sessionID。

```
//定时任务，每隔二十分钟刷新session
refresh:function(){
    var that=this;
    setInterval(that.getSession, 20*60*1000);
}
```

我们每过二十分钟刷新一次，也就是重新请求一次。

将该函数放到app.js的生命周期函数中，这样一旦打开小程序就会执行

```
/**
* 当小程序初始化完成时，会触发onLaunch(全局只触发一次)
*/
onLaunch:function(){
    this.getSession();
    this.refresh();
}
```

5、每次请求需要将sessionID放入cookie中，并将它设置头request header中

```
//通过将sessionID写到cookie中，请求数据
wx.request({
    url:app.webUrl + '/getinfo',
    header:{"Cookie":"JSESSIONID="+wx.getStorageSync("sessionId")},
    method:"GET",
    success:function(res){
        fail:function(error){
            console,log(error);
        }
    }
})
```

第二种：小程序保持唯一标识uuid（如果后台过滤器需要登陆验证的话，小程序端的请求接口都需要放行），这个就不贴代码了，基本和上面逻辑差不多，只不过这次存的是uuid。

1、小程序请求微信，拿到code，请求后台并传递code
2、后台根据code请求微信，拿到openID
3、后台根据openID去微信信息表查询该openID是否绑定用户，绑定了，返回一个唯一表示uuid以及绑定标识，没有绑定，创建一个uuid以及未绑定标识返回给小程序
4、小程序判断未绑定，跳转到绑定页面，根据输入信息+uuid返回给后台绑定用户，后台返回是否绑定成功
5、根据绑定是否成功，小程序操作

小程序只保存uuid（相当于accesstoken），这个uuid是小程序与后台连接的桥。

注意：这两种方法都是不安全的，试想一下如果sessionID被劫持到了，并且还处于当前会话中，黑客完全可以通过该session ID恶意获取后台数据（累屎CSRF攻击）。uuid的话，因为这种请求被后台过滤器放过，所以完全可以直接请求后台数据了。