

Gears Click-SSO

场景

非Atlassian的应用系统，在点击Atlassian产品上的链接，如JIRA, Confluence的具体页面，通过在没有Crowd或者其它的SSO产品支持下，一般都会转到登录页面，由用户再次录入Atlassian产品的中的账户名和密码再次登录，在使用体验上比较麻烦，同时也浪费了时间。

同时，如果操作的人在Atlassian产品没有账户，此时可以设置一个专用账户，帮忙用户能够获取他们需要查看的信息

Gears Click-SSO的目标是让非Atlassian的产品能够通过点击Atlassian的产品链接直接登录到如JIRA, Confluence等，而不用跳转一次由用户再次登录账户和密码来进行登录。

应用产品

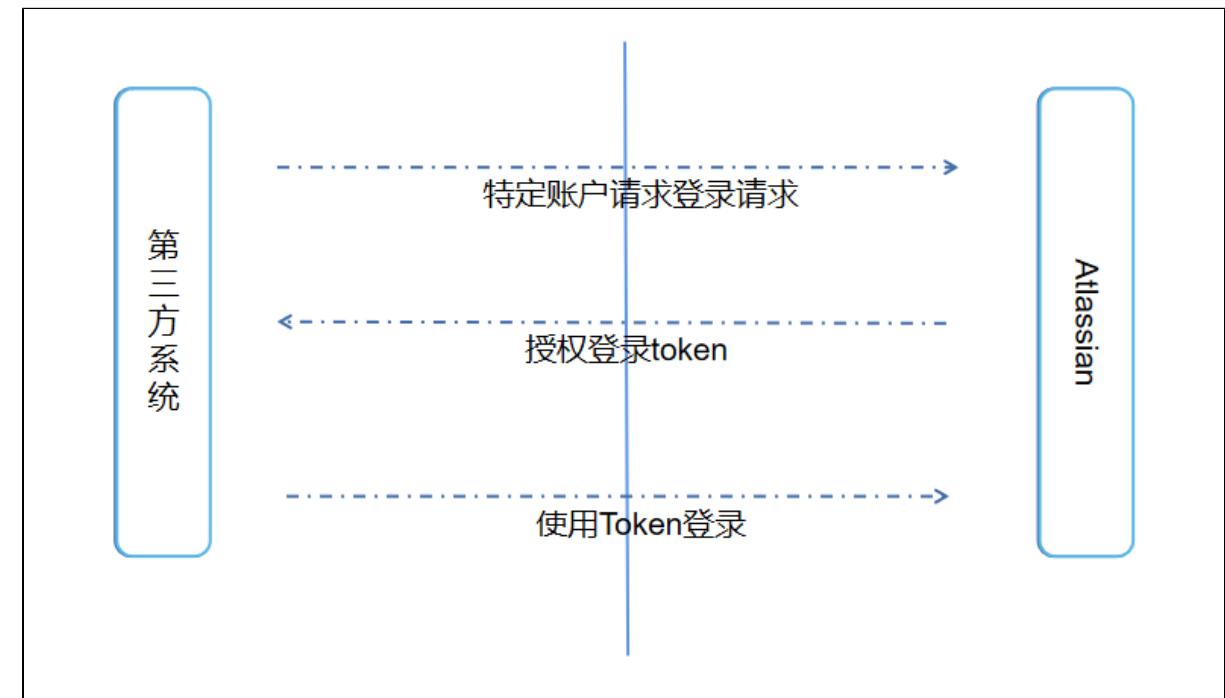
- 1. JIRA
- 2. Confluence

作用

- 1. 能够实现不需要账户和密码登录
- 2. 能够以指定账户（第三方系统的账户在Atlassian中不存在，使用Atlassian中的账户）登录

方案

登录示例图



- 1. 第三方登录，首先调用获得特定用户的token接口
- 2. Atlassian根据用户特定请求，返回token(专码专用，短时间内失效)到第三系统，
- 3. 第三方系统使用Token，放在请求地址中或者head请求头中来登录Atlassian产品

接口

获得Token接口

协议

rest

接口

本页内容

- 场景
- 应用产品
- 作用
- 方案
- 接口
 - 获得Token接口
 - 协议
 - 接口
 - 方法
 - 请求参数
 - 返回参数
 - Confluence登录方法
 - JIRA 登录方法
 - 加密方法

/rest/sso/1.0/user/createToken

方法

post

请求参数

```
{
  "username": "hktxcn" ## 以谁的身份登录JIRA/Confluence
  "password": "Cjza65RynLkbGABYKUxzw==" ## 分派给某个系统 的账户对应的密码
  "sys": "SI" #分派给的系统名称
}
```

参数说明

参数	说明
username	需要以Atlassian的中的哪个账户来登录
password	授权的第三方系统的账户对应的密码
sys	授权的第三方系统编码

返回参数

```
{
  "username": "hktxcn",
  "token": "+mrMuQmY7R6zpwlllFivlhFjyc/e8Z1PxeCk8MrLIRIKas2hKemfHCuo8FNJHIW",
  "created": 1590168016215,
  "expired": 60000
}
```

参数说明

参数	值	说明
username	用户在Atlassian的登录账户	用那个账户登录
token		用于访问Atlassian系统中的地址中增加的参数值，或者放到head头中放的内容
created	date.getTime()	生成token的服务器时间
expired	时间	多久过期，秒*1000

Confluence登录方法

直接将Confluence的页面增加参数即可，如

<http://wiki.jiracn.com/pages/viewpage.action?pageId=72056897&auth=mrMuQmY7R6zpwlllFivlhFjyc/e8Z1PxeCk8MrLIRIKas2hKemfHCuo8FNJHIW>

JIRA 登录方法

直接将JIRA的页面增加参数即可，如

<http://jira.jiracn.com/browse/DEMO-1&auth=0NxTQ99ejjTKrBr/gyWQrJEIJI7zqgbDVTZalwH/iRI dtgJTNxU3aV9Aqea8wB05>

加密方法

```

private String encrypt(String password, String sys) {
    String aesEncode = null;
    try {
        //1. 构造密钥生成器, 指定为AES算法, 不区分大小写
        KeyGenerator keygen=KeyGenerator.getInstance("AES");
        //2. 根据ecnodeRules规则初始化密钥生成器
        //生成一个256位的随机源, 根据传入的字节数组
        SecureRandom secureRandom = SecureRandom.getInstance("SHA1PRNG") ;
        secureRandom.setSeed(sys.getBytes());
        keygen.init(256, secureRandom);
        //3. 产生原始对称密钥
        SecretKey original_key=keygen.generateKey();
        //4. 获得原始对称密钥的字节数组
        byte [] raw=original_key.getEncoded();
        //5. 根据字节数组生成AES密钥
        SecretKey key=new SecretKeySpec(raw, "AES");
        //6. 根据指定算法AES自成密码器
        Cipher cipher=Cipher.getInstance("AES");
        //7. 初始化密码器, 第一个参数为加密(Encrypt_mode)或者解密解密
        (Decrypt_mode)操作, 第二个参数为使用的KEY
        cipher.init(Cipher.ENCRYPT_MODE, key);
        //8. 获取加密内容的字节数组(这里要设置为utf-8)不然内容中如果有中文和英
        文混合中文就会解密为乱码
        byte [] byte_encode=password.getBytes("utf-8");
        //9. 根据密码器的初始化方式--加密: 将数据加密
        byte [] byte_AES=cipher.doFinal(byte_encode);
        //10. 将加密后的数据转换为字符串
        aesEncode=new Base64().encodeToString(byte_AES);
        //11. 将字符串返回
    } catch (Exception e) {
        e.printStackTrace();
    }
    return aesEncode;
}

```