

---

## 爱旅行项目—Token 的添加与置换

Web 前端项目的开发使用分布式设计，使得 session 不能正常使用，如何检测用户是在登录的状态，什么时间登录，自己私有信息如何获取和设置呢？针对于此类问题，我们给每个用户在登录的时候生成一个 Token，每一个用户有一个唯一 Token 值，这个值可以使当前的用户获取自己私有的信息和设置私有的信息。如何设置 Token 和置换 Token 呢？

### 1.1 Token 设计与使用

在我们开始实现 Token 的添加和置换的时候我们需要先考虑这样几点：

1. 什么时间添加 Token，是否需要添加多次；
2. 什么时间置换 Token，置换周期是多长时间；
3. Token 存在什么位置，如何存取 Token 值；
4. Token 值如何传递到后端 server 端

针对上述几个问题，我们在添加 Token 的时候主要是在用户登录时进行 Token 的添加，因为经过登录才能获取 Token，所以我们可以将 Token 添加放到登录成功之后，同时调用登录接口获取到登录的 Token 和超时时间信息；那 Token 存在什么位置那？我们这里将 Token 存在 Cookie 里面，使用 js-cookie 插件对 Token 进行添加置换和删除；那 Token 多长时间进行置换了，这里我们在什么位置添加置换的信息呢？这个跟我们的设置有关，在我们设计界面结构的时候，我们只有一个 html 界面（单页面设计），这样设计的好处在于可以提高前端界面的加载速度和响应速度，极大的提高了用户的体验；同时我们引入路由的机制控制显示页面，因此我们只需要在主 index.js 中增加 Token 置换的程序即可；关于 Token 置换我们只设计了两个主函数（addToken、changeToken）和三个方法（remveCookie、getCookie、setCookie）；Token 设定完成之后我们将 Token 设置的请求的头信息中在每次发送请求的时候携带 Token 值；

#### setCookie

当我们拿到 Token 值后，我们使用 js-cookie 进行 Token 的设定，将 Token 设置到 cookie 中，使用 Token 的时候使用 getCookie 进行获取；setCookie 设计如下：

```
function setCookie(data) {  
  // expTime 使用本地时间，加 2 个小时的时间  
  const days = moment(data.data.expTime - 0).diff(moment(data.data.genTime - 0), 'days',  
true)  
  Cookie.set('token', data.data.token, { expires: days })  
  Cookie.set('user', data.user, { expires: days })  
  Cookie.set('expTime', data.data.expTime, { expires: days })  
}
```

---

## getCookie

此函数主要是用于获取 Token 信息的，主要设计如下：

```
function getCookie() {  
  return {  
    token: Cookie.get('token'),  
    expTime: Cookie.get('expTime'),  
    user: Cookie.get('user')  
  }  
}
```

} 此函数是用来获取 Token 的对象信息的；

## remveCookie

此函数是用来删除 Token 信息的，主函数体如下：

```
function remveCookie() {  
  Cookie.remove('token'),  
  Cookie.remove('expTime'),  
  Cookie.remove('user')  
}
```

## addToken

主要通过发送 HTTP 请求到后台验证用户名和密码是否正确，正确返回 Token 信息和超时信息，addToken 方法设计如下：

```
export function addToken(param) {  
  postRequestForm(loginUrl, param).then(  
    res => {  
      res["user"] = param.name;  
      setCookie(res);  
    }  
  )  
}
```

参数 loginUrl: 验证登录的 url 地址，可参照接口文档进行添加和使用；

参数 param 为登录输入的用户名和密码格式为{name:'name',password:'password'}

在代码中我们给 res 添加了一个 user 参数，此参数是添加当前的登录的用户信息；

**此函数使用的位置在登录时回调函数调用，实现添加功能；**

## ChangeToken

主要是在 index.js 文件中加载完成后触发，其代码结构设计为：

```
export function changeToken() {
```


```

const tokenPojo = getCookie()
const milliseconds = 25 * 60 * 1000
clearTimeout(timeout)
if (tokenPojo.token && tokenPojo.user) {
  console.log(`${milliseconds} 毫秒后自动转换 Token.`)
  timeout = setTimeout(() => {
    postRequest(chTokenUrl).then(data => {
      data["user"] = tokenPojo.user
      setCookie(data);
    }).catch(requestError)
  }, milliseconds)
}
}

```

此函数中有一个等待时间 `milliseconds` 常量，用于设定定时更新 `Token` 的时间，这个时间的设定是有要求的，必须大于 10 分钟且小于 60 分钟，当设定的时间比较小，前端会不断向后台发送数据请求，会占用我们服务器的带宽，影响界面显示的效率，如果设定的时间过大则会导致 `Token` 过期，需要我们重新登录，因此在一个小时之内请求 2 次 `Token` 置换是比较合理的，为避免出现特殊情况，我们这里的时间设定为 25 分钟请求置换一次 `Token`；

**changeToken 函数的使用我们主要放在 index.js 文件进行调用，示例如下（图 1）：**



```

ReactDOM.render(
  <Router
    history={hashHistory}
    routes={routes} />,
  document.getElementById('root'),
  changeToken
);

```

图 1

## 1.2 第三方登录

### 微信登录

关于微信登录，这里简单进行介绍一下，微信登录需要请求和改变的参数比较多，需要有几个不同的 `Token`：`access_token`、`refresh_token`、`openid` 等，这些参数是验证微信登录是否成功，是否能持续获得数据的必要参数，关于这些参数我们应该怎样进行修改才能保证之前的登录方式能正常使用，且不需要增加太多的代码那？

主要从以下几点进行修改：添加微信 `Token` 信息，获取 `Token` 信息，删除 `Token` 信息，定时更新 `Token` 信息；

**添加 Token 信息：**在登录时我们点击微信登录按钮切换到微信登录界面，通过二维码扫描，页面会重定向到我们的登录界面（[http://localhost:3000/#/login?user\\_type=1&token=](http://localhost:3000/#/login?user_type=1&token=)

---

access\_token=&expires\_in=&refresh\_token=&openid=)，通过获取 url 上的参数，得到微信登录的信息，判断登录的类型 user\_type 是否等于 1（user\_type=1 为微信登录）将 Token 信息添加到 Cookie 上，并且调用后端接口获取当前登录人信息，并一起设置到 Cookie 中；

**获取 Token 信息：**只要是在 getCookie() 函数中添加以下几行代码：

```
function getCookie() {  
    return {  
        ...  
        access_token: Cookie.get('access_token'),  
        expires_in: Cookie.get('expires_in'),  
        refresh_token: Cookie.get('refresh_token'),  
        openid: Cookie.get('openid'),  
        userType: Cookie.get('userType')  
    }  
}
```

此处不需要判断值是否存在；

**删除 Token 信息：**跟 getCookie()函数相同，只需要在 remveCookie()函数中增加以下函数即可：

```
function remveCookie() {  
    ...  
    Cookie.remove('access_token');  
    Cookie.remove('expires_in');  
    Cookie.remove('refresh_token');  
    Cookie.remove('openid');  
    Cookie.remove('userType');  
}
```

此处不需要判断值是否存在；

**定时更新 Token 函数信息：**因为在更新 Token 函数的时候，需要先判断登录的类型是什么？如果'userType'为空，则是正常登陆，否则需要判断是否是第三方登录，登录需要调用第三方的登录函数，此处暂时使用 if 进行判断，后期可使用工厂模式进行设计；具体实现如下：

使用：if (tokenPojo.token && tokenPojo.user && !tokenPojo.userType)判断是否是第三方登录，如果检测!tokenPojo.userType 为 false 则为第三方，再判断 tokenPojo.userType 值为什么，如果等于 1 则为微信登录，请求更新的函数代码为：

```
postRequest('/auth/vendors/wechat/token/refresh ', {}, {  
    headers: {  
        'token': tokenPojo.token,  
        'expTime': tokenPojo.expTime,  
        'access_token': tokenPojo.access_token,  
        'expires_in': tokenPojo.expires_in,  
        'refresh_token': tokenPojo.refresh_token,  
        'openid': tokenPojo.openid  
    }  
}).then(res => {  
    res["user"] = tokenPojo.user
```

---

```
let expires_in = res.get("expires_in");
Cookie.set('access_token', res.get('accessToken'), expires_in)
Cookie.set('expires_in', res.get('expiresIn'), expires_in)
Cookie.set('refresh_token', res.get('refreshToken'), expires_in)
Cookie.set('openid', res.get('openid'), expires_in)
setCookie(res);
}).catch(requestError)
```

我们在请求获取时需要给请求函数增加 `headers` 信息，主要包含微信登录的关键参数和系统使用的 `Token` 参数信息；

注：

其他第三方登录也可参照微信登录的方式去设定；

