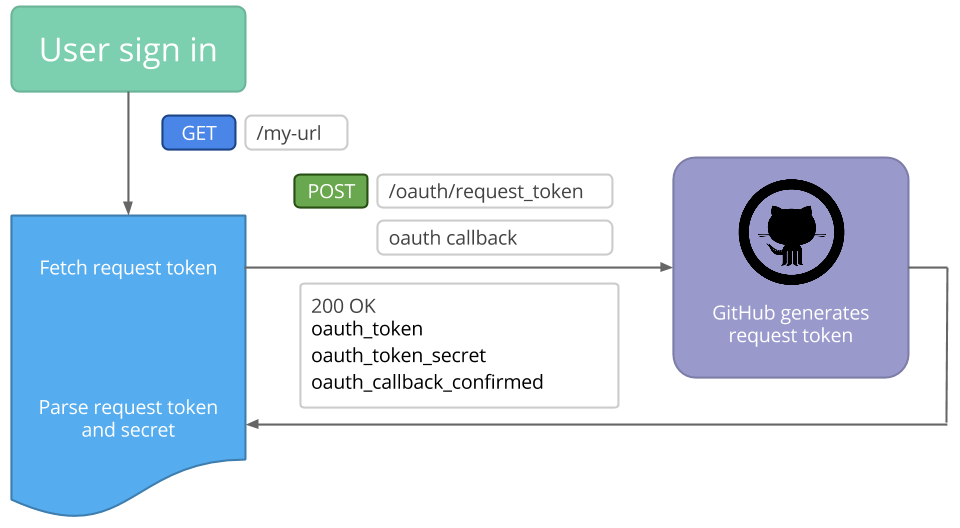
**Handling oAuth callbacks in AngularJS SPA**

In this tutorial I will show you how I solved one of the mayor issues when trying to add third party **oAuth** services to our **SPA** *"Single Page Application"*.

**oAuth** is great ! Though some users love it others not so much. Above all, the service is not only quite convenient it's also nesesary if we want use the oAuth provider API on behalf our users.

*Good examples are: Post a tweet, store a file in Dropbox or create a GIST in GitHub*

**Usual oAuth flow:**



oAuth implmentation is quite easy nowdays, but as you probably know already, It presents a serious problem for SPA's.   
oAuth relies on a post to a **callback** route in our server, this undoubtedly will break our client app state.

*Though there are a some workarounds like client side oAuth these wont really help in terms of reliablitity and security.*

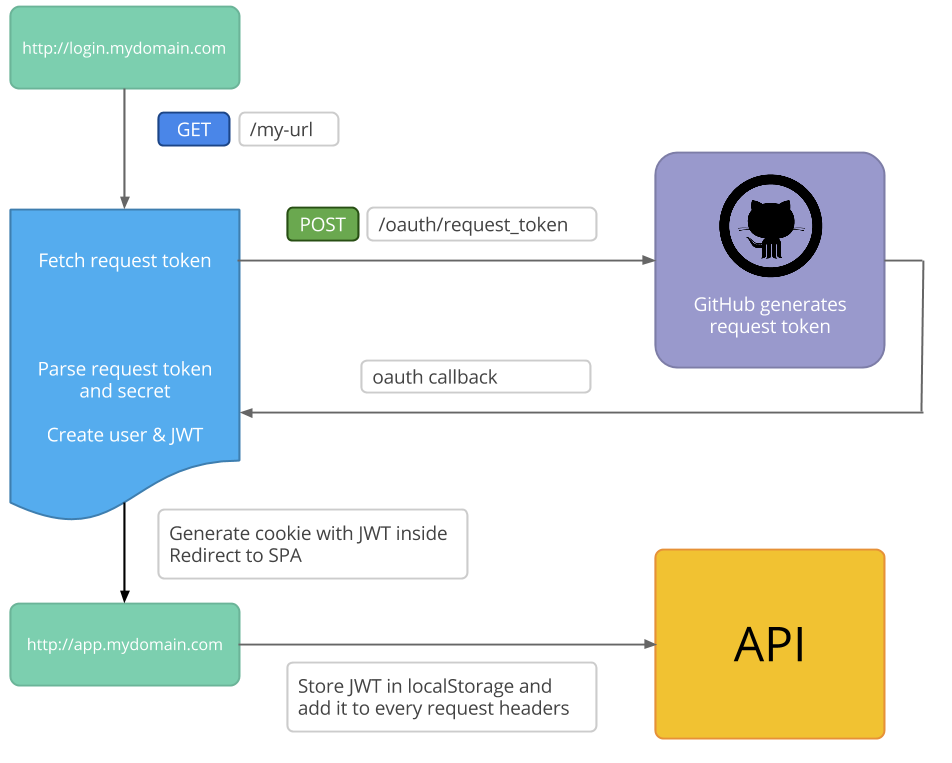
**Tokens cookies and beyond...**

Now that we know our SPA can't handle the oAuth callback is time I show you what I've done.   
First off I think is very important that I describe how my API handles trusted communications with my client.   
Im using **JWT** ( or JSON Web Tokens for short ) they are incredible easy to use and implement.   
JWT's can handle **delegation** quite easely, contrary to cookies and other session specific, JWT's work reliably across **CORS**.

I strongly suggest you read these how-to's first:

* [Cookies vs Tokens](http://blog.auth0.com/2014/01/07/angularjs-authentication-with-cookies-vs-token/)
* [10 Things You Should Know about Tokens](http://blog.auth0.com/2014/01/27/ten-things-you-should-know-about-tokens-and-cookies/)

**oAuth flow and JWT**

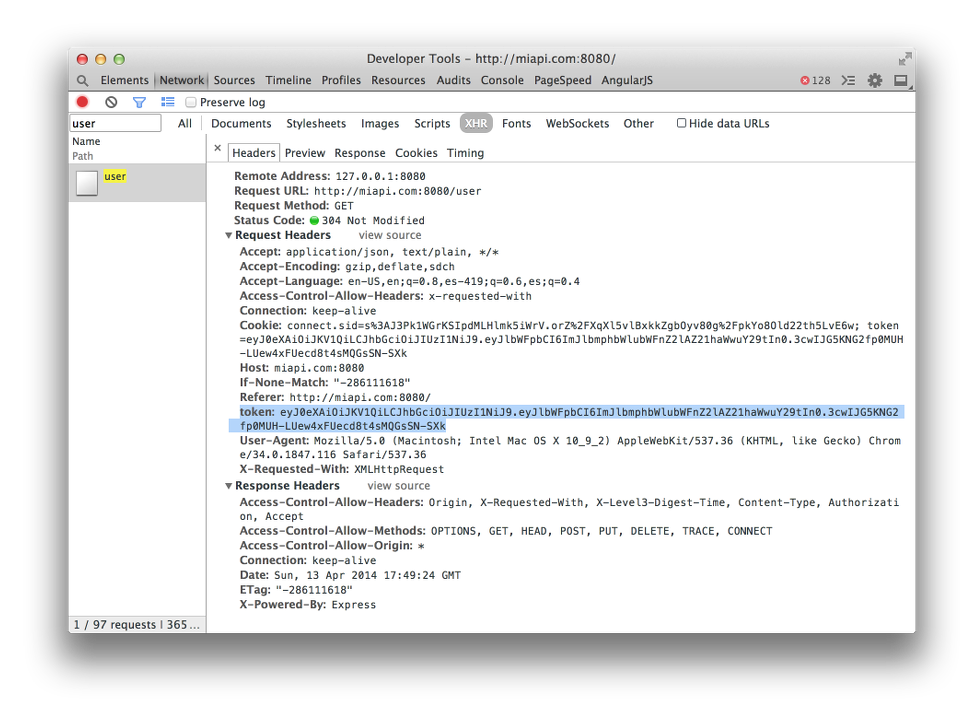


**Our client code**

Im using Angularjs as my client framework, the very first thing I do in my app is to check whether or not I've a trusted user to begin with.   
Calls to the API will be issued with a custom **header** named **token** that contains the JWT.   
Since I want my users to be remembered Im using localStorage to save and retreive the token.

*I could however use cookies here, that's ok too.*

**JWT in action**

**Tools used:**

* [AngularJS](http://angularjs.org/)
* [Angular-cookies](http://docs.angularjs.org/api/ngCookies)
* [Restangular](https://github.com/mgonto/restangular)
* [LocalStorageModule](https://github.com/grevory/angular-local-storage)

angular.module( 'App', [

'LocalStorageModule',

'ngCookies',

'restangular'

])

.controller( 'AppCtrl', function AppCtrl ( $scope, localStorageService, $cookies, Restangular ) {

// Check if we have a stored token

var token = localStorageService.get('token');

if (token) {

// Add it to our default headers

Restangular.configuration.defaultHeaders.token = token;

// Make an API call

Restangular.one('user').get().then(function (user) {

$scope.user = user;

}, function (error) {

// It's invalid remove it

localStorageService.remove('token');

// TODO: redirect to login

});

// Do we have a cookie with a token in it ?

} else if ($cookies.token) {

// cookies are escaped

token = unescape($cookies.token);

// Add it to out headers

Restangular.configuration.defaultHeaders.token = token;

// Make an API call

Restangular.one('user').get().then(function (user) {

// If ok add it to localStorage

localStorageService.add('token', token);

$scope.user = user;

},

function (error) {

// It's invalid remove it

$cookies.token = undefined;

// TODO: redirect to login

});

}

});

**How does it work ?**

From [http://login.mydomain.com](http://login.mydomain.com/) the user will issue an oAuth request.   
The oAuth provider will in turn post a **callback** to my server.   
My server in turn will check if this is a new user and create a JWT for it.   
Then It'll send a cookie that contains the newly generated **JWT** and issue a redirect to [http://app.mydomain.com](http://app.mydomain.com/) in here my client app will take control and check for the token inside that cookie.   
*Im not using localStorage here because it wont work across subdomains !*

*I could however send the token in the redirect url:*[*http://app.mydomain.com?token=blablabla*](http://app.mydomain.com/?token=blablabla)

Finally if anyone tries to access [http://app.mydomain.com](http://app.mydomain.com/) directly, the client will check for a valid token either inside a cookie or localStorage and if not present it will redirect users to [http://login.mydomain.com](http://login.mydomain.com/)

EOF