Angular.js Quickstart

- NodeJS 4.3
- Angular version 1.5.0-rc.0

Before Starting

Please remember that for security purposes, you have to register the URL of your app on the Application Settings section on Auth0 Admin app as the callbackURL.

1. Add the Auth0 scripts and set the viewport

Add the code below to the `index.html` file to include Auth0's angular module and its dependencies and set the viewport:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
<!-- We use client cookies to save the user credentials -->
<script src="//code.angularjs.org/1.2.16/angular-cookies.min.js"></script>

<!-- Auth0 Lock script and AngularJS module -->
<script src="//cdn.auth0.com/js/lock-9.0.min.js"></script>
<!-- angular-jwt and angular-storage -->
<script type="text/javascript" src="//cdn.rawgit.com/auth0/angular-
storage/master/dist/angular-storage.js"></script>
<script type="text/javascript" src="//cdn.rawgit.com/auth0/angular-
jwt/master/dist/angular-jwt.js"></script>

<script src="//cdn.auth0.com/w2/auth0-angular-4.js"> </script>
```

```
<!-- Setting the right viewport -->
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-
scale=1.0, user-scalable=no" />
```

2. Add the module dependency and configure the service

Add the `auth0`, `angular-storage` and `angular-jwt` module dependencies to your angular app definition and configure `auth0` by calling the `init` method of the `authProvider`:

```
1
2
3
4
5
6
7
8
9
10
11
12
// app.js
angular.module('YOUR-APP-NAME', ['auth0', 'angular-storage', 'angular-jwt'])
.config(function (authProvider) {
  authProvider.init({
    domain: 'myapplaud.auth0.com',
    clientID: 'TisoMUTDnyiQao6HDXNnmRS03ZAHA9i1'
  });
})
.run(function(auth) {
  // This hooks al auth events to check everything as soon as the app starts
  auth.hookEvents();
});
```

3. Implement the login

To implement the login, inject the `auth` service into any controller and call the `signin` method to show the Login / SignUp popup.

In the following code, a call is added to the `login` method of the `LoginCtrl` controller. On login

success, the user's profile and token are saved to `localStorage`:

```
// LoginCtrl.js
angular.module('YOUR-APP-NAME').controller('LoginCtrl', ['$scope', '$http', 'auth',
'store', '$location',
function ($scope, $http, auth, store, $location) {
  $scope.login = function () {
    auth.signin({}, function (profile, token) {
      // Success callback
      store.set('profile', profile);
      store.set('token', token);
      $location.path('/');
    }, function () {
      // Error callback
    });
  }
}]);
```

```
<!-- login.tpl.html -->
<!-- ... -->
<input type="submit" ng-click="login()" />
<!-- ... -->
```

This is how it will look on a browser...

https://your-domain.com/login

Note: There are multiple ways of implementing a login. The example above displays the Login Widget. However you may implement your own login UI by changing the line `<script src="//cdn.auth0.com/js/lock-7.5.min.js"></script>` to `<script src="//cdn.auth0.com/w2/auth0-6.7.js"></script>`. For more details, see the auth0-angular repo.

4. Add a logout button

To add a logout button, call the `auth.signout` method to log out the user. Also remove the profile and token information saved in `localStorage`:

```
1
2
3
4
5
$scope.logout = function() {
  auth.signout();
  store.remove('profile');
  store.remove('token');
}
```

```
<input type="submit" ng-click="logout()" value="Log out" />
```

5. Configure secure calls to your API

To configure secure calls to the API you are creating on no-api, return on each request the JWT token received on the login by adding `jwtInterceptor` to the list of `$http` interceptors:

```
1
2
3
4
5
6
7
8
9
10
```

```
11
12
13
// app.js
myApp.config(function (authProvider, $routeProvider, $httpProvider,
jwtInterceptorProvider) {
  // ...

  // We're annotating this function so that the `store` is injected correctly when
this file is minified
  jwtInterceptorProvider.tokenGetter = ['store', function(store) {
    // Return the saved token
    return store.get('token');
  }];

  $httpProvider.interceptors.push('jwtInterceptor');
  // ...
});
```

Now you can regularly call this API with `$http`, `$resource` or any rest client as you would normally and the JWT token will be sent on each request.

6. Show the user's information

After a user has logged in, retrieve from the `auth` service the `profile` property, which has all of the user's information:

```
<span>His name is {{auth.profile.nickname}}</span>
```

```
1
2
3
4
// UserInfoCtrl.js
function UserInfoCtrl($scope, auth) {
  $scope.auth = auth;
}
```

To discover all the available properties of a user's profile, see Auth0 Normalized User Profile. Note that the properties available depend on the social provider used.

7. Keep the user logged in after a page refresh

The user's profile and tokens are already saved to `localStorage`. To keep the user logged in, retrieve the token from `localStorage` on each page refresh and let `auth0-angular` know the user is already authenticated:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
angular.module('myApp', ['auth0', 'angular-storage', 'angular-jwt'])
.run(function($rootScope, auth, store, jwtHelper, $location) {
  // This events gets triggered on refresh or URL change
  $rootScope.$on('$locationChangeStart', function() {
    var token = store.get('token');
    if (token) {
      if (!jwtHelper.isTokenExpired(token)) {
        if (!auth.isAuthenticated) {
          auth.authenticate(store.get('profile'), token);
        }
      } else {
        // Either show the login page or use the refresh token to get a new idToken
        $location.path('/');
      }
    }
  });
});
```

8. All done!

You have completed the implementation of Login and Signup with Auth0 and AngularJS.

Optional Steps

Add routing

Most apps will need to authenticate users to enable access certain routes.

To enable access to a route:

1. Set the `requiresLogin` property to `true`.
2. Add the `$routeProvider` configuration in the `config` method of our app.
3. Specify a login page to which users will be redirected if trying to access a route when not authenticated.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
// app.js
.config(function (authProvider, $routeProvider, $locationProvider) {
  $routeProvider.when('/login', {
```

```
    templateUrl: 'login.tpl.html',
    controller: 'LoginCtrl'
  });
  // Logged in route
  $routeProvider.when('/user-info', {
    templateUrl: 'userInfo.tpl.html',
    controller: 'UserInfoCtrl',
    requiresLogin: true
  });

  authProvider.init({
    domain: 'myapplaud.auth0.com',
    clientID: 'TisoMUTDnyiQao6HDXNnmRS03ZAHA9i1',
    callbackURL: location.href,
    // Here include the URL to redirect to if the user tries to access a resource
when not authenticated.
    loginUrl: '/login'
  });
});
```

Note: If you are using a UI router, see UI Router.

Additional information

For additional information on how to use this SDK, see Auth0 and AngularJS.

Product