Authentication

Authentication occurs on the REST API backend.

However, the state is also sent to the node.js server when the user logs in or out.

This allows node.js to also keep track of the current user state e.g. to deny access to certain routes when user is not logged in.

Auth service

auth.logIn(credentials)

Log in with credentials:

```
var credentials = {
    username: 'demo',
    password: 'demo'
};

auth
    .logIn(credentials)      // Returns promise
    .then(...);              // Success and error handler
```

Behind the scenes, the auth service will:

1. POST credentials to `/login` REST API endpoint
2. POST response from REST API to `/session/create` of zc-owner-application so node.js can maintain state
3. Create new session using session service
4. Return promise

Parameters

Credentials should be a JavaScript object with the following properties:

- *username*: username

- *password*: password

Return value

Returns a promise.

auth.logOut()

Log out:

```
auth
    .logOut()               // Returns promise
    .then(...);             // Success and error handler
```

Behind the scenes, the auth service will:

1. GET `/logout` REST API endpoint
2. GET `/session/create` of zc-owner-application so node.js can maintain state
3. Destroy current session using session service
4. Return promise

Return value

Returns a promise.

auth.isLoggedIn()

Check if user is logged in:

```
if (auth.isLoggedIn()){
    ...
}
```

Return value

Returns a boolean.

Session service

The session service contains the session data and takes care of storing the session in the `$cookieStore`.

session.create(token, user, roles)

Create a new session and stores it in a cookie for later retrieval:

```
session.create(token, user, roles);
```

Return value

Void.

session.destroy()

Destroy the current session and removes the cookie(s):

```
session.destroy();
```

Return value

Void.