OAUTH

The OAuth process works as follows:

- A login link with the applications client ID, redirect URL and state parameters redirects the user to the OAuth application
- The user see the login form and enters his/her credentials
- Then the user is redirected back to the application with an auth code
- This code is exchanges for an access token, which is used to identify and log in the user

This example also illustrates the steps, which the API implementation described below follows, with code examples:
https://www.oauth.com/oauth2-servers/server-side-apps/example-flow/

1. Config

The backend needs to have following parameters defined in the config .yml file. (Important is the proper indentation in the yml files!)

```
oauth:
   enabled: true
   authorizeUrl: "https://test.com/oauth/authorize"
   tokenUrl: "https://test.com/oauth/token"
   userDetailsUrl: "https://test.com/oauth/getuserinfo"
   clientId: "invitepro_sso"
   clientSecret: "xxx"
   scope: openid
   registerUrl: "https://test.com/oauth/registration"
   acceptHeader: "application/json"
```

2. API-Call

When the user clicks the login button in the web frontend the following API-call is triggered and he/she is redirected to the oauth login page of the provider:

```
GET {{authorizeUrl - path, e.g. /oauth/authorize}}
?client_id={{clientId}}&response_type=code&redirect_uri={{baseUrl}}/oauth-ca
llback&scope=openid HTTP/1.1
Host: {{authorizeUrl - base url e.g. test.com}}
```

Where `{{authorizeUrl}}` and `{{clientId}}` are from above defined parameters in the config file and `{{baseUrl}}` is the already in the .yml config file defined base url of the application.

On the OAuth page the user logs in and once finished a call to the `redirect-uri` will be executed. Upon success this call returns an access code. With this code an OAuth token will be requested with following call:

```
POST {{tokenUrl - path, e.g. /oauth/token}} HTTP/1.1
Host: {{tokenUrl - base url e.g. test.com}}
Authorization: Basic {{"clientId:clientSecret" - base64-encoded}}
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code={{access
code}}&redirect_uri={{baseUrl}}/oauth-callback&scope=openid
```

Where `{{tokenUrl}}`, `{{clientId}}`, `{{clientSecret}}`, `{{baseUrl}}` are defined in the config .yml file and `{{access code}}` is the code returned by the authorize API-call. `{{clientId}}` and `{{clientSecret}}` need to be base64 encoded in following format `"clientId:clientSecret"`.

Then the user details will be queried with this call:

```
GET {{userDetailsUrl - path e.g. /oauth/getuserinfo}} HTTP/1.1
Host: {{userDetailsUrl - base url e.g. test.com}}
Accept: {{acceptHeader}}
Authorization: Bearer {{auth token}}
```

where `{{userDetailsUrl}}` and `{{acceptHeader}}` is also defined in the config .yml file and `{{auth token}}` is the result of the prior token API-Call. `{{acceptHeader}}` is a HTTP accept header, such as “application/json”, for further documentation see: https://developer.mozilla.org/de/docs/Web/HTTP/Headers/Accept.

If this call is successful the user is logged in, redirected to `/dashboard` and can then navigate around the application.

2.1 Success Responses

The following table gives an overview over the success responses described above.

2.1.1. /oauth-callback

For this API-Call an access token is return in form of a url parameter

Example:
`/oauth-callback?code="hCL9tA4SKiKgFxvXvWnGn…..ngAM5NNC8riKrBHfQ2"`

2.1.2 /tokenUrl

If the correct token was sent to the API following access token with is returned.

Example:

```
{
	"access_token":  "hCL9tA4SKiKgFxvXvWnG...ngAM5NNC8riKrBHfQ2",
	"expires_in": 7000,
	"refresh_token": "..."
}
```

2.1.3 /userDetailsUrl

If result code 0 is returned the API-Call was successful and also return the following data, email and customer number of the logged in user.

Parameter:

- `"Data"`: object containing the following values:
  - `"CustomerNr"`: (String) customer number of logged in user; is later used for the statistics API
  - `"Email"`: (String) e-mail address of logged in user
- `"ResultCode"`: (Long) shows success or error code, more information see below
- `"ResultDescription"`: (String) contains relevant message set in case of an error

Example:

```
{
	"Data":  {
		"CustomerNr":  "000.000.000.000",
		"Email": "abc@test.at"
	},
	"ResultCode": 0,
	"ResultDescription": "..."
}
```

Result Codes:

- 0 Success
- 1000 Unknown error

2.2 Error Responses

These are the errors that can occur during the above described steps.

2.2.1 /oauth-callback

The oauth-callback call can throw following errors:

- 404 Not found
  Possible reasons:
  - OAuth is not enabled in the config.

- 400 Bad Request
  Possible reasons:
  - Something in the api call has the wrong format

2.2.2 /tokenUrl

- 403 Forbidden - Unable to request access token
  Possible reasons:
  - The access code is empty
  - Wrong parameters sent to API

2.2.3 /userDetailsUrl

- 500 Internal Server error
  - If the result code is 1002 see error description
  - else “Got non zero response while retrieving user details after oauth login, ResultCode: xxx”

  Possible reasons:
  - See error description, error stems from OAuth application

3. Logs

After deploying the application, the logs in the console should show any errors, warnings, infos and debug messages. The credentials for access to the server via SSH can be found in the admin dashboard of the host.
The level of detail of the logs can be adjusted in the config .yml file with following parameters (pay attention to the indent of the parameters in the .yml file):

```
logging:
  level:
    org.springframework: INFO
    com.lyconet.invite.backend: DEBUG
    com.lyconet.invite.backend.db: INFO
```

It can have one of the following values: ERROR, WARN, INFO, DEBUG or TRACE.

Also logs can be found by using following command in the console, grepping for “StopWatch”, “ERROR” or “WARN”, depending on what kind of information is of interest.

```
sudo cat /usr/local/apache-tomcat9/logs/catalina.out | grep StopWatch
```

An example for such logs, as follows: