Implement other OAuth Providers

Expand the OAuth Module version 1.2

Implementation steps

So you setup the OAuth module in your project and want to add a new OAuth provider?

Here is how:

In the document we'll implement a fictitious OAuth provider called myOAuth.

1. Create a copy of the following constants in your module
    a. CallbackURI_Google → CallbackURI_myOAuth
    b. ClientId_Google → ClientId_myOAuth
    c. ClientSecret_Google → ClientSecret_myOAuth
    d. OAuthTokenURI_Google → OAuthTokenURI_myOAuth
    e. OAuthURI_Google → OAuthURI_myOAuth
    f. UserInfoURI_Google → UserInfoURI_myOAuth
2. Register your application with myOAuth and make sure to setup your callback URL as https://<appname>/callback/myOAuth
3. Copy the callback URL into your CallbackURI_myOAuth constant
4. Copy your myOAuth client id into your ClientId_myOAuth constant
5. Copy your myOAuth client secret into your ClientSecret_myOAuth constant
6. Set the value for the OAuthURI_myOAuth constant to the OAuth URL provided by myOAuth
7. Set the value for the OAuthTokenURI_myOAuth constant to the URL for retrieving the access token provided by myOAuth
8. Set the value for the UserInfoURI_myOAuth constant to the URL that will provide the Mendix app with detail of the user provided by myOAuth
9. Import your project into Eclipse
10. In the class OAuthSigninMultiIdP add a new if statement like below:

```
else if(pathParameters[1].equals("myOAuth")){
                    new GetAccessCodeMyOAuth().getCode(UUIDstate, ServletResponse);
    }
```

11. Create a new class called GetAccessCodeMyOAuth copy and paste the code from the GetAccessCodeGoogle class and change the class name to GetAccessCodeMyOAuth.
12. Change the three final fields of the class to use your MyOAuth constants
13. Change the logging to represent MyOAuth
14. This results in the following class:

```
package oauthmodule.actions.custom;
import java.io.IOException;
import javax.servlet.http.HttpServletResponse;
import oauthmodule.proxies.constants.Constants;
import com.mendix.core.Core;
/**
 * HelperClass for retrieving a accesscode from MyOAuth
 *
 * @Author: <your name>
 * @version: 1.0
 * @since: <date>
```

```
 */
public class GetAccessCodeMyOAuth {

private final String OAUTHURI = Constants.getOAuthURI_MyOAuth();
private final String CLIENTID = Constants.getClientId_MyOAuth();
private final String CALLBACKURI = Constants.getCallbackURI_MyOAuth();

protected void getCode(String UUIDstate, HttpServletResponse servletResponse) throws IOException{
Core.getLogger("OAuthSignin").trace("Get token from MyOauth");
StringBuilder oauthUrl = new StringBuilder()
          .append(OAUTHURI)
          .append("?client_id=").append(CLIENTID) // the client id from the api console registration
          .append("&response_type=code")
          .append("&scope=openid%20email") // scope is the api permissions we are requesting
          .append("&redirect_uri=").append(CALLBACKURI) // the servlet that google redirects to after authorization
          .append("&state="+UUIDstate)
          .append("&access_type=online") // here we are asking to access to user's data while they are not signed in
          .append("&approval_prompt=auto"); //this requires them to verify which account to use, if they are already signed in
Core.getLogger("OAuthSignin").trace("Url used for myOAuth: \n"+oauthUrl.toString());
servletResponse.sendRedirect(oauthUrl.toString());
}
}
```

15. Depending on the requirements from the myOAuth provider change the append actions so that the correct url is constructed
16. In the class OAuthCallback add the following field:

```
private final String USERINFOURI_MYOAUTH = Constants.getUserInfoURI_MyOAuth();
```

17. In the class OAuthCallback add the following else statement after the statement for Facebook:

```
else if(pathParameters[1].equals("myoauth")){
                                        GetAccessTokenMyOAuth accessTokenMyOAuth = new GetAccessTokenMyOAuth (code);
                                        body = accessTokenMyOAuth.getResult();
                    }
```

18. In the class OAuthCallback add the following else statement after the statement for Facebook:

```
else if(pathParameters[1].equals("myoauth")){
          Core.getLogger("OauthCallback").trace("Get userinfo from MyOAuth");
          userInfoString = geturi.get(new StringBuilder(USERINFOURI_MYOAUTH).append(accessToken).toString());
}
```

19. In the class OAuthCallback add the following else statement after the statement for Facebook:

```
else if(pathParameters[1].equals("myoauth")){
       user =resolveUser(context, JsonObj.get(configuration.getUserId_MyOAuth()).toString(),configuration);
}
```

20. Add a new class GetAccessTokenMyOAuth and copy the code form the GetAccessCodeGoogle into this new class
21. Change the name of the class to GetAccessTokenMyOAuth
22. Change the log message to represent myOAuth
23. Change the final fields to retrieve your MyOAuth constant values
24. This will result in a class as below

```
package oauthmodule.actions.custom;
import java.io.IOException;
import org.apache.http.client.ClientProtocolException;
import oauthmodule.proxies.constants.Constants;
import com.google.common.collect.ImmutableMap;
import com.mendix.core.Core;
/**
 * HelperClass for retrieving a accesstoken from MyOAuth
 *
 * @Author: Erwin 't Hoen
 * @version: 1.0
 * @since: 2014-10-02
 */
public class GetAccessTokenMyOAuth extends GetAccessToken {
private final String CLIENTID = Constants.getClientId_MyOAuth ();
private final String CLIENTSECRET = Constants.getClientSecret_MyOAuth ();
private final String CALLBACKURI =  Constants.getCallbackURI_MyOAuth ();
private final String OAUTHTOKENURI = Constants.getOAuthTokenURI_MyOAuth ();
private ImmutableMap<String, String> map =ImmutableMap.<String,String>builder()
.put("code", code)
.put("client_id", CLIENTID)
.put("client_secret", CLIENTSECRET)
.put("redirect_uri", CALLBACKURI)
.put("grant_type", "authorization_code").build();

public GetAccessTokenGoogle(String code) {
          super(code);
}

protected String getResult() throws ClientProtocolException, IOException{
          Core.getLogger("OAuthCallback").debug("Get access token from MyOAuth");
          return PostHttpRequest.post(OAUTHTOKENURI, map);
}}
```

26. In your OAuthModule in the Mendix modeler add the attribute UserId_MyOAuth to the entity OAuthConfig with as default value the value used for identifying your Mendix users (e.g. email)
27. Add the field to the configuration page as well
28. Change the index.html in the theme folder to allow either a standard redirect to the new request handler or add a new button for the MyOAuth provider.
29. Deploy your app and test the new OAuth provider login!

Note: by defining a callback url as http://localhost:8080/callback/myoauth you are able to test the new provider locally first.

If you have any remaining questions please post them on the Mendix forum.