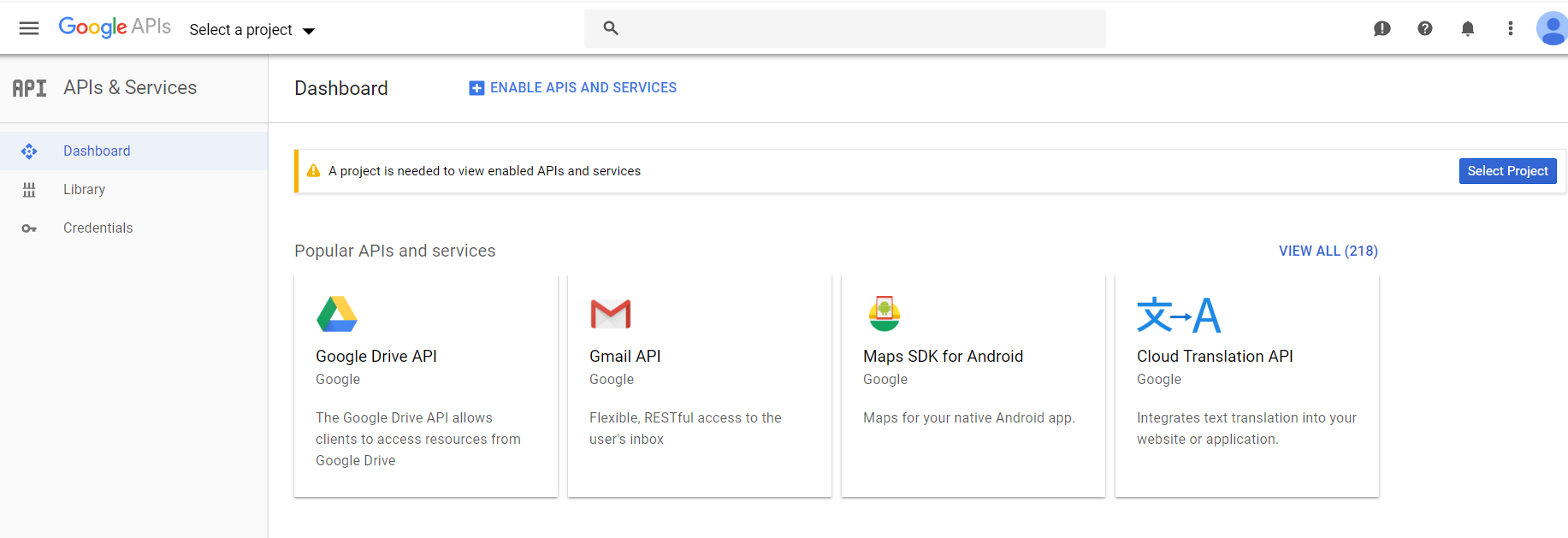
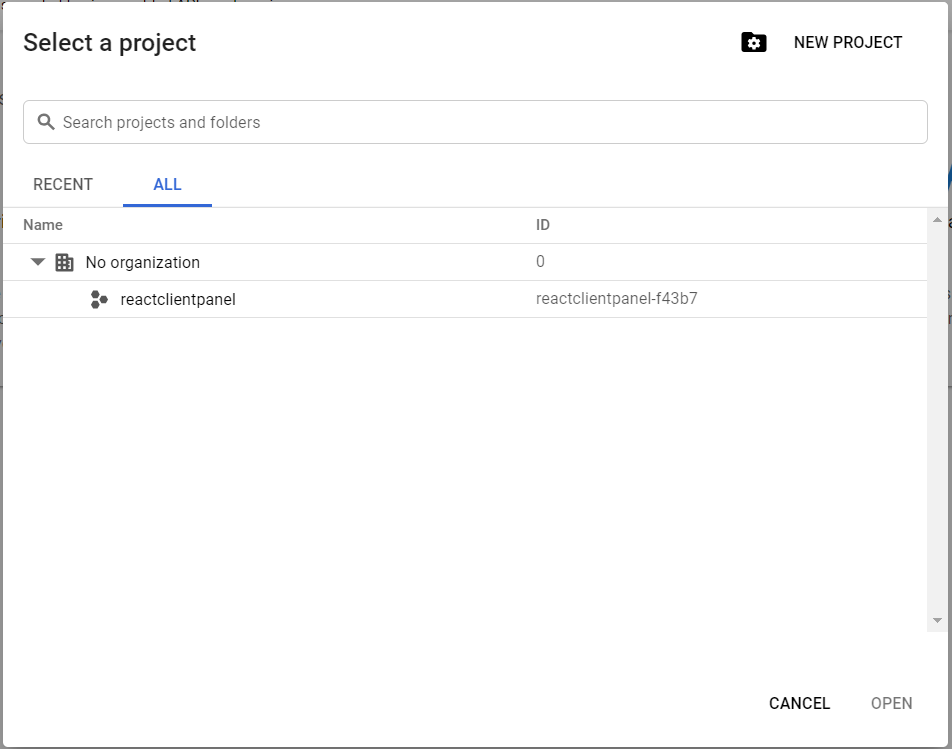
**LECTURE 28 – GOOGLE AUTHENTICATION**

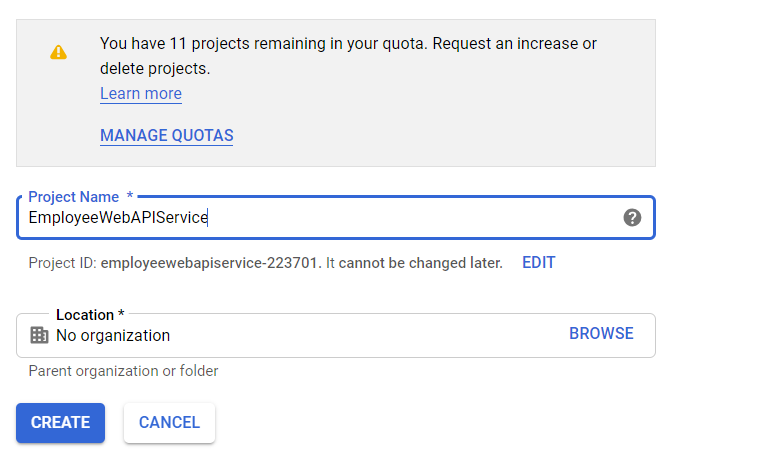
* What we want to do is have a button on the login page that when the User clicks on it they will be directed to google. The User then signs in using their Google account and once authenticated they are redirected back to the application.
* The first thing we need to do is register our application with Google.
* In a web browser navigate to **console.developers.google.com** and log in with your google account. You should be presented with the following screen once you log in.



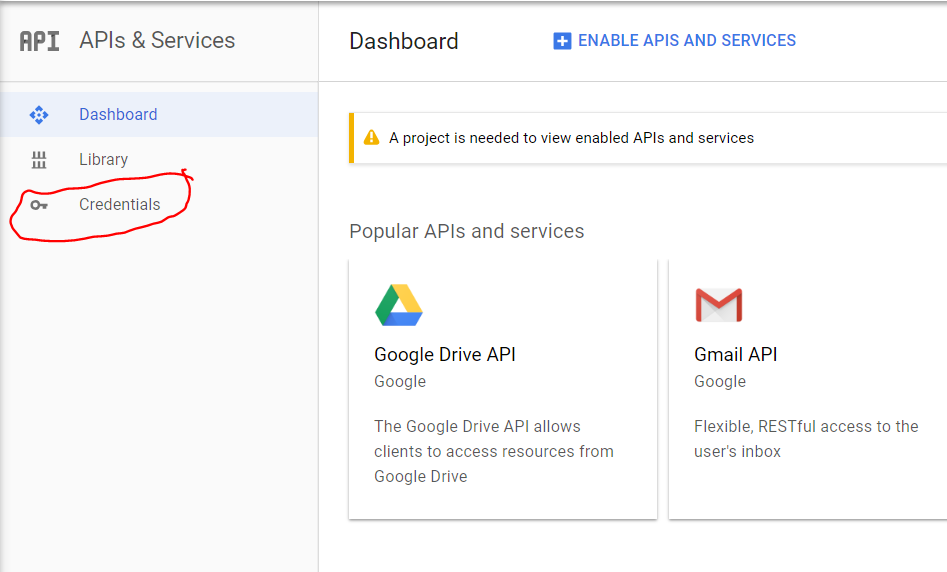
* Click on **Select a project** on the top left-hand corner.
* Click on **NEW PROJECT**.



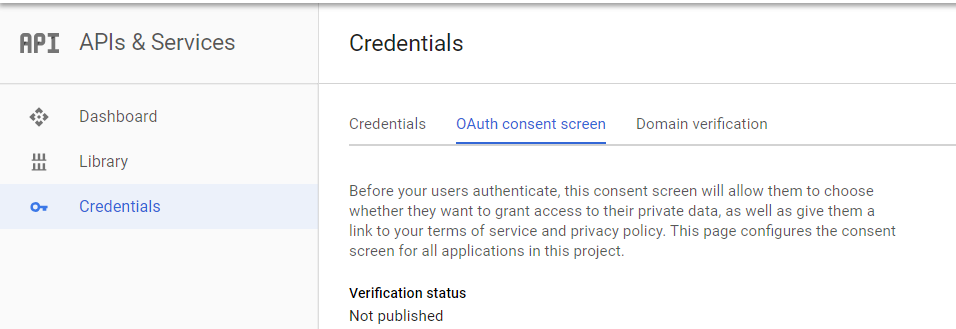
* Name the project **EmployeeWebAPIService** and then click on **CREATE**.



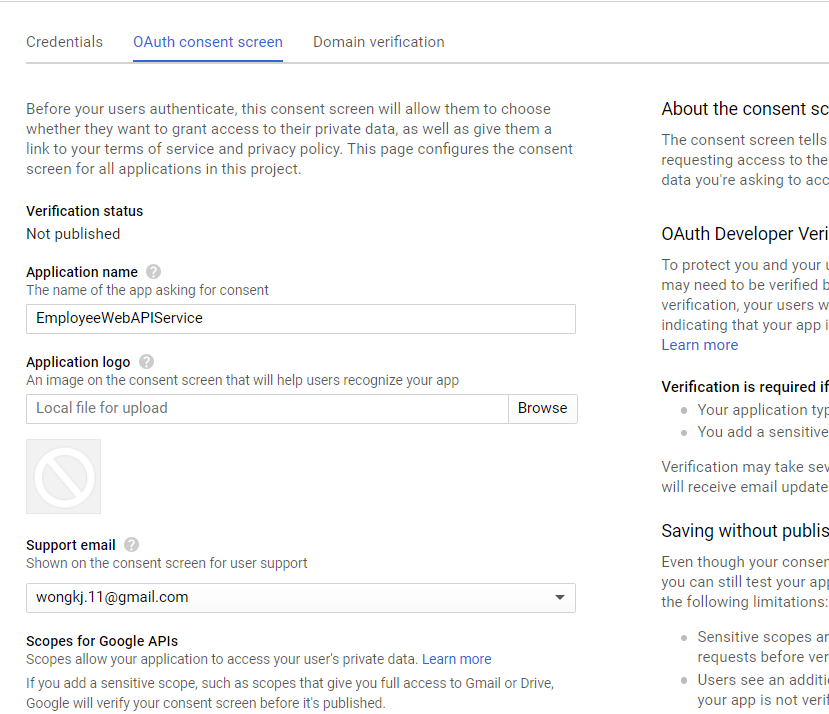
* Once the project is created click on **Credentials** which is located on the left hand of the screen.



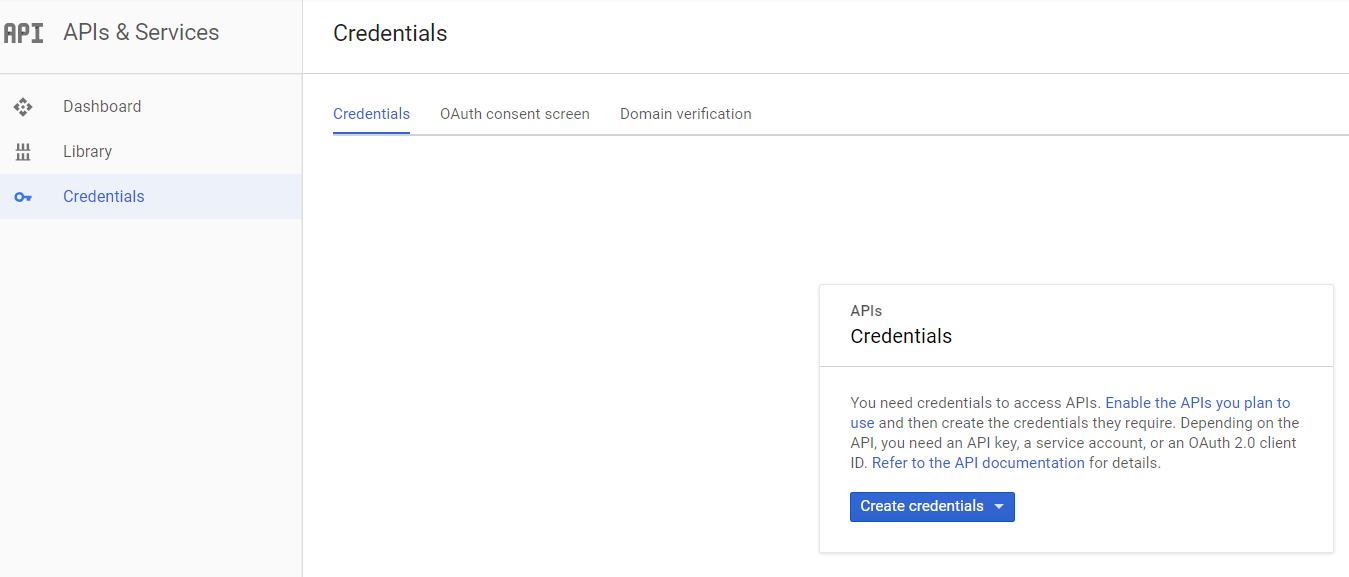
* Click on the **OAuth consent screen** tab.



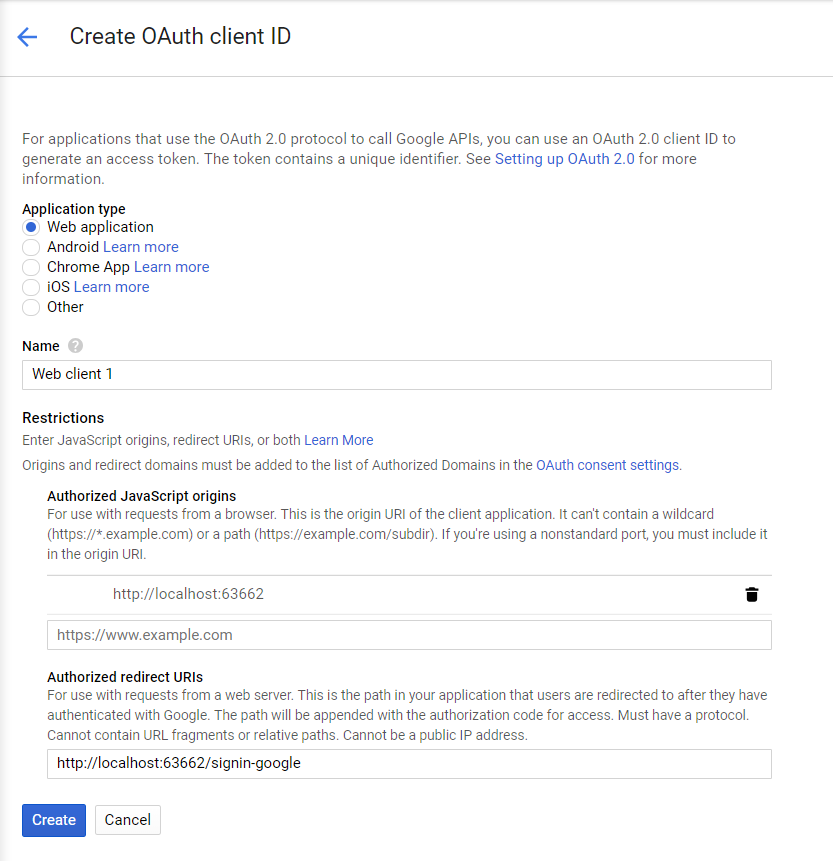
* Type in the Application name then click on **Save**.



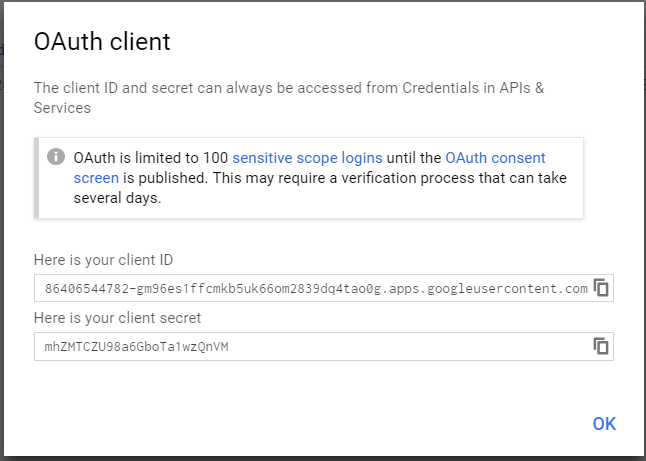
* Go to the **Credentials** tab and then click on **Create credentials** on the popup.



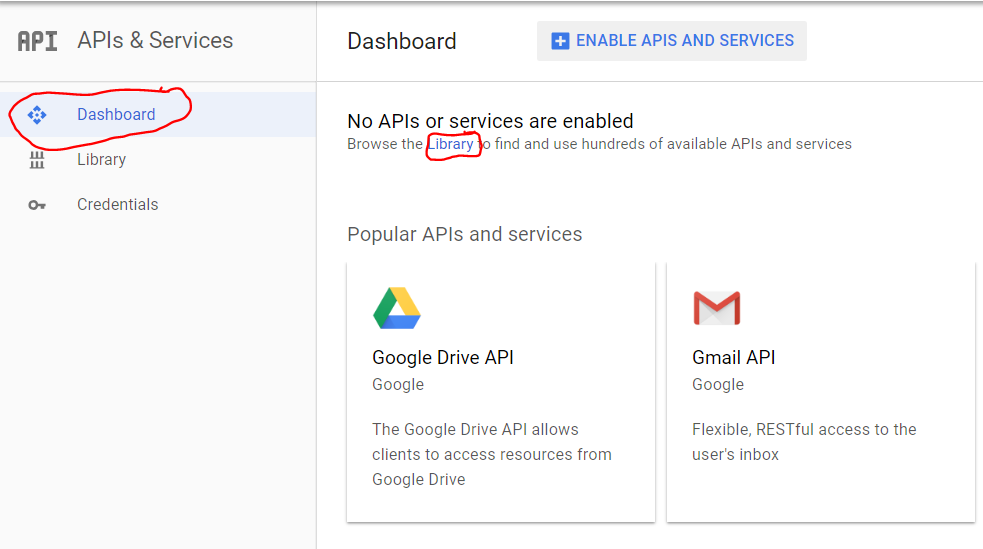
* Select **OAuth Client ID**.
* Select **Web Application.**
* Leave the Name as the default one.
* For Authorized JavaScript Origins, this is the root domain of your application when you run it. In this case it is [**http://localhost:63662**](http://localhost:63662).
* Add the Authorized Redirect URI which is the URI that the User gets redirected to once they are authenticated. In this case we chose <http://localhost:63662/signin-google>.



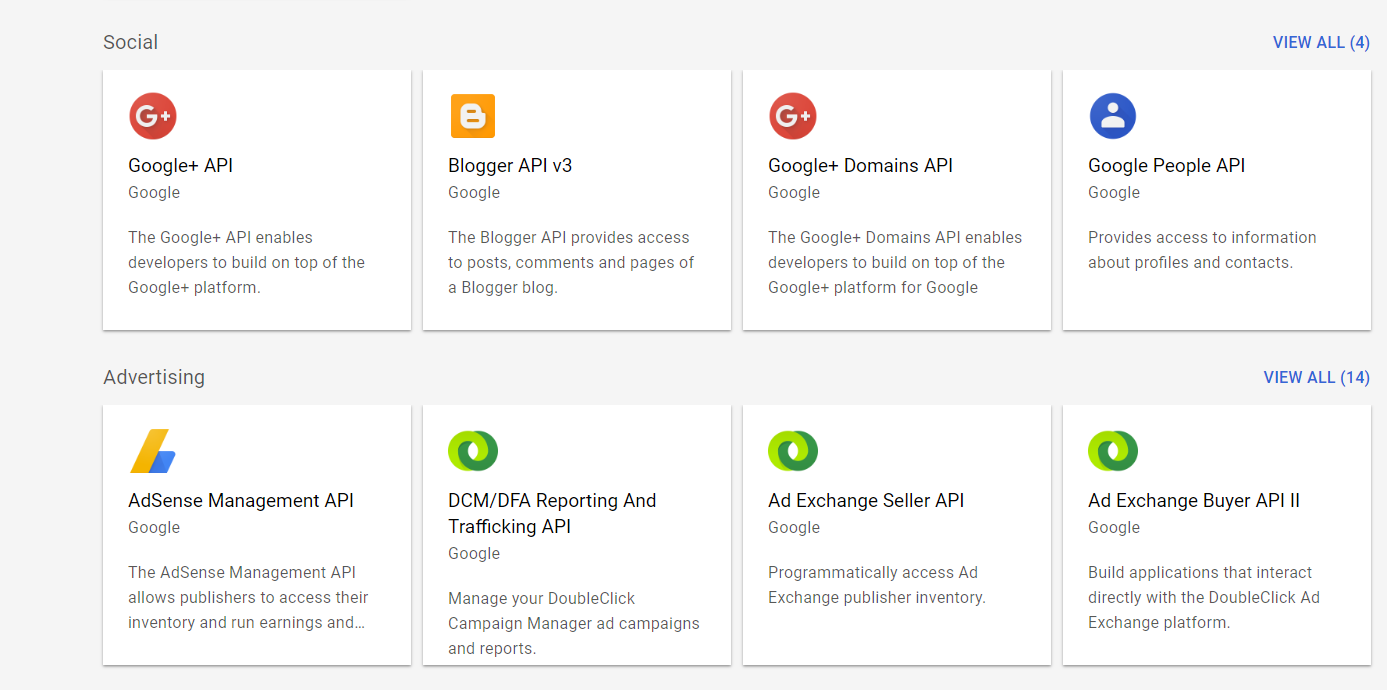
* The following screen should popup once you click **Create**. Click **OK** once done.



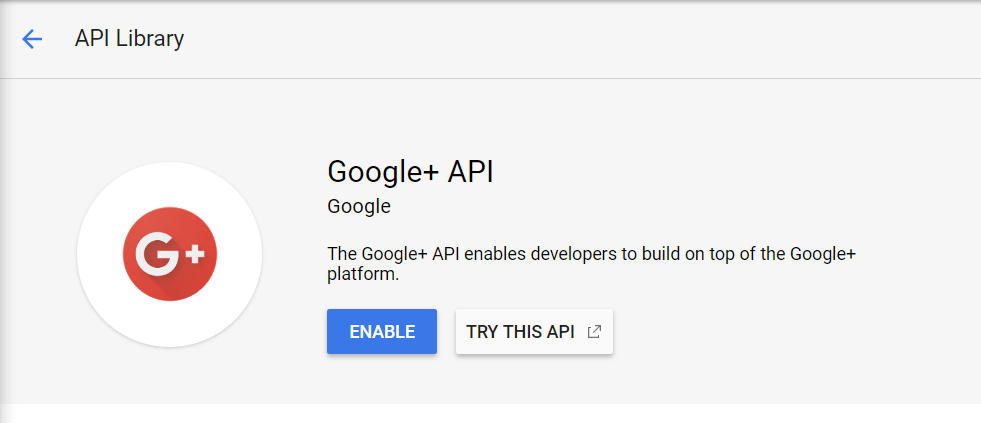
* There’s one more thing to do. We must enable Google+ API Service.
* On the main screen go to **Dashboard** then click on **Library**.



* Scroll down and choose **Google+ API**.



* Click on **Enable**.



* Now we need to enable Google Connection on our API.
* Go back to Visual Studio and go to **EmployeeService2/App\_Start/Startup.Auth.cs**.
* Uncomment the **UseGoogleAuthentication** method.

// Enable the application to use bearer tokens to authenticate users

app.UseOAuthBearerTokens(OAuthOptions);

// Uncomment the following lines to enable logging in with third party login providers

//app.UseMicrosoftAccountAuthentication(

// clientId: "",

// clientSecret: "");

//app.UseTwitterAuthentication(

// consumerKey: "",

// consumerSecret: "");

//app.UseFacebookAuthentication(

// appId: "",

// appSecret: "");

app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()

{

ClientId = "",

ClientSecret = ""

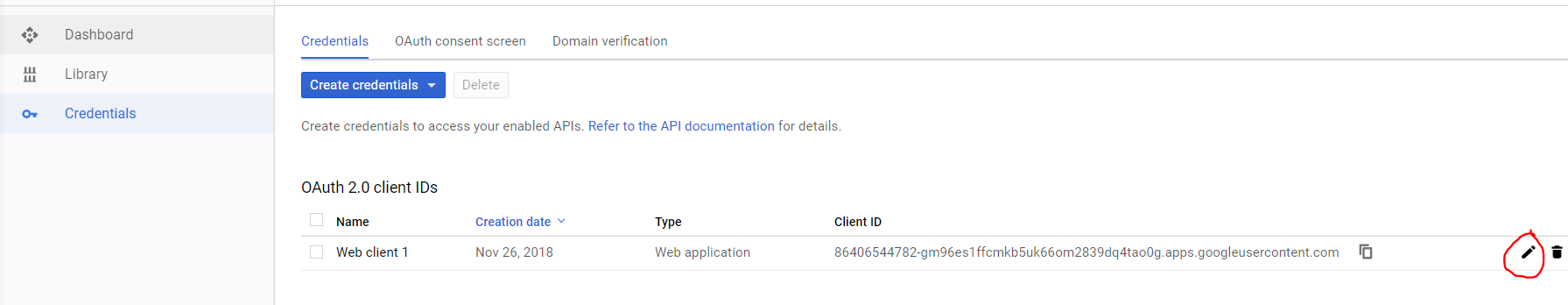
});

}

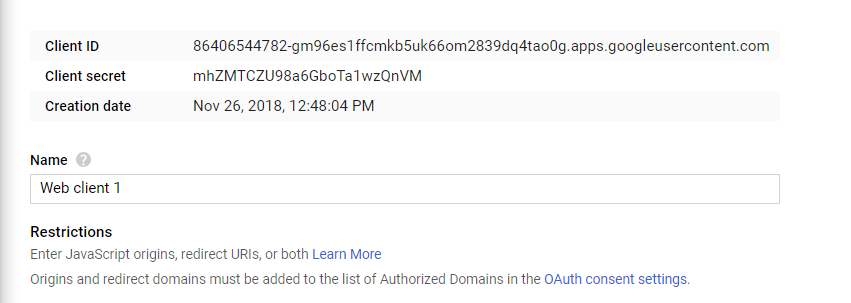
}

}

* Now to enable Google Authentication we need the Client ID and the Client secret.
* Go back to Google Developer and click on **Credentials**.
* Go to the **Credentials** tab and then click on the pencil icon (Edit Item) for the application you want to enable Google Authentication for.



* Here is the location of the Client Id and Client Secret.



* Copy and paste this information into the Google Authentication function in Startup.Auth.cs.

// Enable the application to use bearer tokens to authenticate users

app.UseOAuthBearerTokens(OAuthOptions);

// Uncomment the following lines to enable logging in with third party login providers

//app.UseMicrosoftAccountAuthentication(

// clientId: "",

// clientSecret: "");

//app.UseTwitterAuthentication(

// consumerKey: "",

// consumerSecret: "");

//app.UseFacebookAuthentication(

// appId: "",

// appSecret: "");

app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()

{

ClientId = "86406544782-gm96es1ffcmkb5uk66om2839dq4tao0g.apps.googleusercontent.com",

ClientSecret = "mhZMTCZU98a6GboTa1wzQnVM"

});

}

}

}

* Now we are going to create the Google Login button on the Login.html page.
* Go to **EmployeeService2/Login.html** and make the following changes.

<table class="table table-bordered">

<thead>

<tr class="success">

<th>

Social Logins

</th>

</tr>

</thead>

<tbody>

<tr>

<td>

<input id=” btnGoogleLogin” type="button" value="Login with Google" class="btn btn-success" />

</td>

</tr>

</tbody>

</table>

<div id="divError" class="alert alert-danger collapse">

<a id="linkClose" href="#" class="close">&times;</a>

<div id="divErrorText">

</div>

</div>

</div>

</div>

<script src="Scripts/jquery-3.3.1.min.js"></script>

<script src="Scripts/bootstrap.min.js"></script>

* We are now going to connect the Google connect button to the Google Authentication method.
* In Login.html write the following jQuery code.

<script type="text/javascript">

$(document).ready(function () {

$('#btnGoogleLogin').click(function () {

window.location.href = "";

});

$('#linkClose').click(function () {

$('#divError').hide('fade');

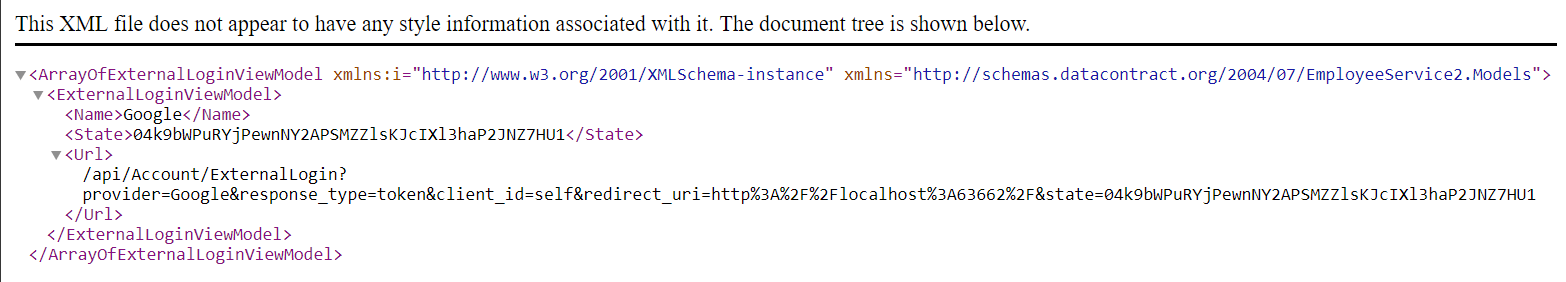
});

* We need to follow a few steps to get the href to connect to.
* In a Web browser type in the following:

**\*\*NOTE:** 63662 is the port number where your local application is running.

http://localhost:63662/api/Account/ExternalLogins?returnUrl=%2F&generateState=true

* You should get the following screen.



* Copy the Url and paste that into the href in the Google Button jQuery.

<script type="text/javascript">

$(document).ready(function () {

$('#btnGoogleLogin').click(function () {

window.location.href = "/api/Account/ExternalLogin?provider=Google&response\_type=token&client\_id=self&redirect\_uri=http%3A%2F%2Flocalhost%3A63662%2F&state=04k9bWPuRYjPewnNY2APSMZZlsKJcIXl3haP2JNZ7HU1";

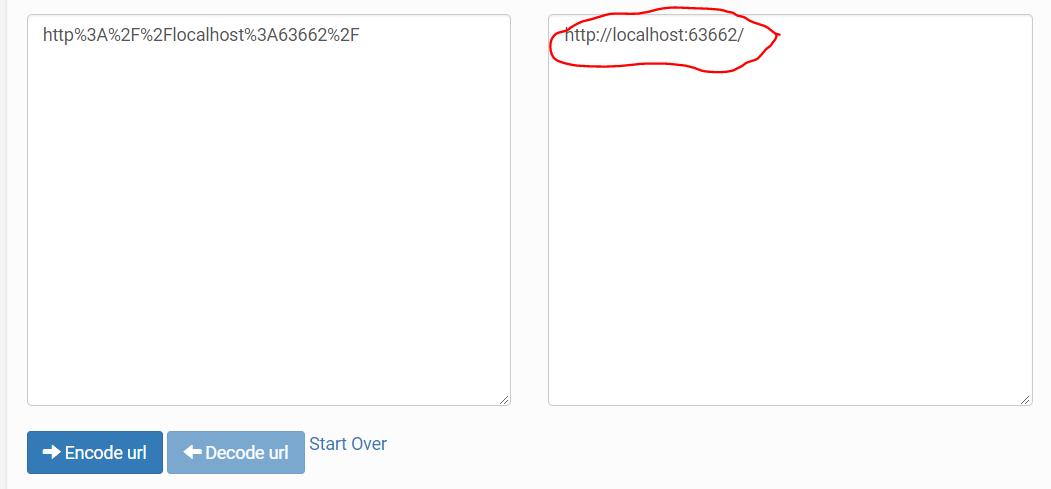
});

$('#linkClose').click(function () {

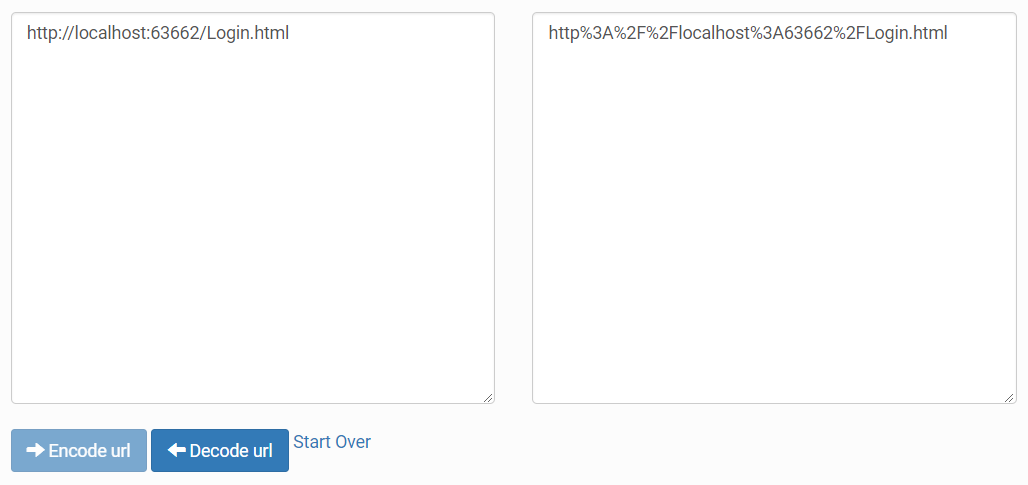
$('#divError').hide('fade');

});

* Cut out the section highlighted in yellow above.
* In a web browser go to [**https://www.url-encode-decode.com**](https://www.url-encode-decode.com) and paste the above encoded text in the left hand TextArea and decode it.



* On the decoded string, append the words **Login.html**, copy the whole thing and paste it back in to the left Text Area, but this time hit **Encode url**.



* Copy the encoded string on the right and replace uri in the href jQuery with this newly encoded string.

<script type="text/javascript">

$(document).ready(function () {

$('#btnGoogleLogin').click(function () {

window.location.href = "/api/Account/ExternalLogin?provider=Google&response\_type=token&client\_id=self&redirect\_uri=http%3A%2F%2Flocalhost%3A63662%2FLogin.html&state=04k9bWPuRYjPewnNY2APSMZZlsKJcIXl3haP2JNZ7HU1";

});

$('#linkClose').click(function () {

$('#divError').hide('fade');

});

* Next, go to **Providers/ApplicationOAuthProvider.cs** and make the following changes to the **ValidateClientRedirectUri** method.

public override Task ValidateClientRedirectUri(OAuthValidateClientRedirectUriContext context)

{

if (context.ClientId == \_publicClientId)

{

Uri expectedRootUri = new Uri(context.Request.Uri, "/Login.html");

if (expectedRootUri.AbsoluteUri == context.RedirectUri)

{

context.Validated();

}

}

return Task.FromResult<object>(null);

}

public static AuthenticationProperties CreateProperties(string userName)

{

IDictionary<string, string> data = new Dictionary<string, string>

{

{ "userName", userName }

};

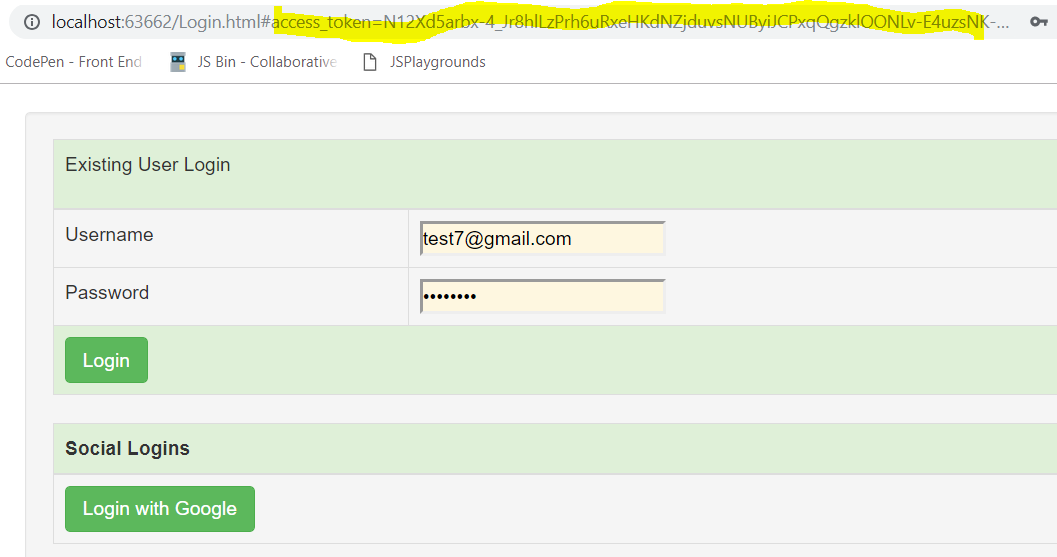
return new AuthenticationProperties(data);

}

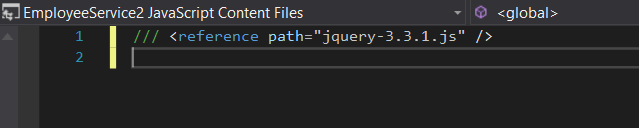
}

}

* If you are logged in to Google in a web browser, log out of it.
* Run the Login.html screen and click on **Login with Google**.
* Sign with your Google Account.
* You will be redirected back to the Login screen, but you should see in the Uri that you now have the access token included in there.



* In the **scripts** folder create a javascript file called **GoogleAuthentication.js**.
* In GoogleAuthentication.js we are going to make use of jQuery so drag and drop the jquery-3.3.1.js file into the GoogleAuthentication.js file.



* In GoogleAuthentication.js, write the following code.

/// <reference path="jquery-3.3.1.js" />

function getAccessToken() {

if (location.hash) {

if (location.hash.split('access\_token=')) {

var accessToken = location.hash.split('access\_token=')[1].split('&')[0];

if (accessToken) {

isUserRegistered(accessToken);

}

}

}

}

function isUserRegistered(accessToken) {

$.ajax({

url: '/api/Account/UserInfo',

method: 'GET',

headers: {

'content-type': 'application/JSON',

'Authorization': 'Bearer ' + accessToken

},

success: function (response) {

if (response.HasRegistered) {

localStorage.setItem('accessToken', accessToken);

localStorage.setItem('userName', response.Email);

window.location.href = 'Data.html';

}

else {

signupExternalUser(accessToken);

}

}

})

}

function signupExternalUser(accessToken) {

$.ajax({

url: '/api/Account/RegisterExternal',

method: 'POST',

headers: {

'content-type': 'application/JSON',

'Authorization': 'Bearer ' + accessToken

},

success: function (response) {

window.location.href = "/api/Account/ExternalLogin?provider=Google&response\_type=token&client\_id=self&redirect\_uri=http%3A%2F%2Flocalhost%3A63662%2FLogin.html&state=d2BVMaKRy4a1D\_rG9JkvpprBxJ3m0Klwm\_X8JhVxCe01";

}

})

}

* The RegisterExternal function is located in the AccountController.
* Go to the AccountController and make the following changes.

// POST api/Account/RegisterExternal

[OverrideAuthentication]

[HostAuthentication(DefaultAuthenticationTypes.ExternalBearer)]

[Route("RegisterExternal")]

public async Task<IHttpActionResult> RegisterExternal(RegisterExternalBindingModel model)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

var info = await Authentication.GetExternalLoginInfoAsync();

if (info == null)

{

return InternalServerError();

}

var user = new ApplicationUser() { UserName = info.Email, Email = info.Email };

IdentityResult result = await UserManager.CreateAsync(user);

if (!result.Succeeded)

{

return GetErrorResult(result);

}

result = await UserManager.AddLoginAsync(user.Id, info.Login);

if (!result.Succeeded)

{

return GetErrorResult(result);

}

return Ok();

}

* Go to **Login.html** and drag and drop the **GoogleAuthentication.js** file into the following spot.

<script src="Scripts/jquery-3.3.1.min.js"></script>

<script src="Scripts/bootstrap.min.js"></script>

<script src="Scripts/GoogleAuthentication.js"></script>

<script type="text/javascript">

$(document).ready(function () {

* Add the **getAccessToken()** function in the Login.html jquery code.

<script type="text/javascript">

$(document).ready(function () {

getAccessToken();

$('#btnGoogleLogin').click(function () {

window.location.href = "/api/Account/ExternalLogin?provider=Google&response\_type=token&client\_id=self&redirect\_uri=http%3A%2F%2Flocalhost%3A63662%2FLogin.html&state=d2BVMaKRy4a1D\_rG9JkvpprBxJ3m0Klwm\_X8JhVxCe01";

});

$('#linkClose').click(function () {

$('#divError').hide('fade');

});

* Now if you login to the application with Google successfully it will redirect you back to the Login page, but if you click on the Login with Google again you will be directed to the Data Page.

**LECTURE 29 – FACEBOOK AUTHENTICATION**

* We must first register our application with Facebook.
* Navigate to **https://developers.facebook.com**.
* Fa
* Fa
* Dsf
* Asdf
* Af
* Asd
* Fa
* As
* Dfa
* Sf
* Adf
* Asd
* Af
* As
* Dfa
* Fads
* Fas
* Dfa
* Fd
* Afa
* Sf
* Ads
* fsadf
* Ads
* Fa
* Fa
* Dsf
* Asdf
* Af
* Asd
* Fa
* As
* Dfa
* Sf
* Adf
* Asd
* Af
* As
* Dfa
* Fads
* Fas
* Dfa
* Fd
* Afa
* Sf
* Ads
* fsadf
* Ads
* Fa
* Fa
* Dsf
* Asdf
* Af
* Asd
* Fa
* As
* Dfa
* Sf
* Adf
* Asd
* Af
* As
* Dfa
* Fads
* Fas
* Dfa
* Fd
* Afa
* Sf
* Ads
* fsadf
* Ads
* Fa
* Fa
* Dsf
* Asdf
* Af
* Asd
* Fa
* As
* Dfa
* Sf
* Adf
* Asd
* Af
* As
* Dfa
* Fads
* Fas
* Dfa
* Fd
* Afa
* Sf
* Ads
* fsadf