

1. User send request to web server with user name and password
2. Web server verify using DB
3. If valide then Generate Cookies & serialize in Security Context inside Http Response Context
4. Send Response to Client
5. Client send request to User for some resource /page

Then In Header it check cookies attached , then deserialize that and check it authentic user or not

Then Check user is authorized to access page /resource or not

1. Then send response back to client.

In .net even though we are not creating any authentication , it create an anonymous identity .

Security Context Object :

Claim Name

Claim Email

Identity2

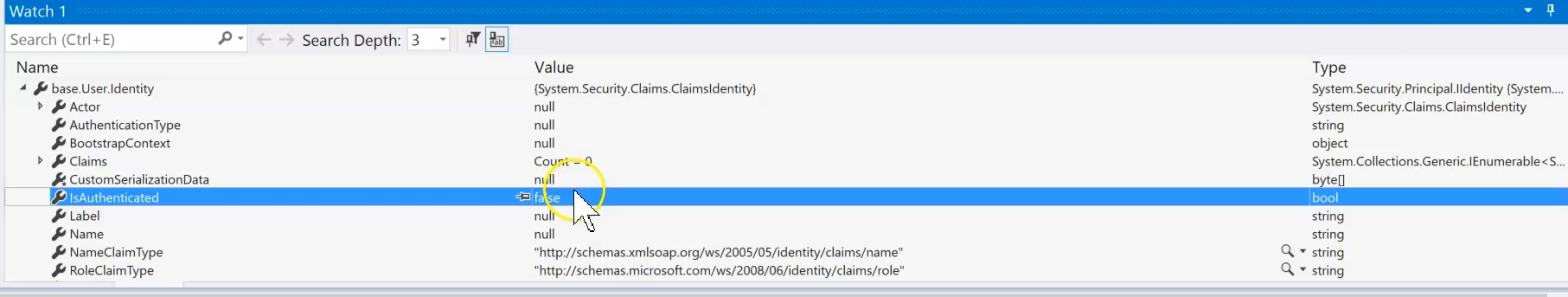
Identity1

Default Identity -Anomyous

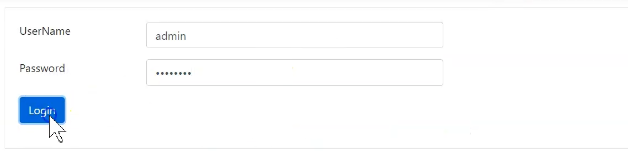
User/principal

Identity3

Default Identity : IsAuthenticated =false ( In Security Context)



Then we have login page



Now On server , Generate Cookies using Cookies Handler

If(Username==”admin” && password=”password”)

{

// Generate Security context for Header in which will have Cookies as serialize object

var claims = new List<Claim>()

{

new Claim( ClaimTypes.Name, "admin"),

new Claim( ClaimTypes.Email, "admin@admin.com")

};

var identity = new ClaimsIdentity(claims,"MyCookiesAuth");

var claimPrincipal = new ClaimsPrincipal(identity);

HttpContext.SigninAsync(“MyCookiesAuth”, claimPrincipal) //

// This line will serialize the principal as string , then encrypt that string and set as cookies and attach in header and send back to client .

}

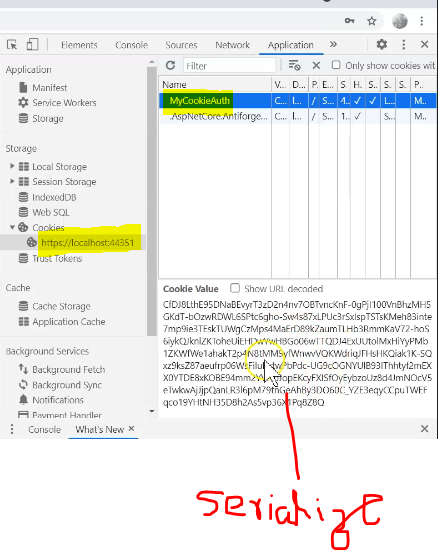
But this line can give us error “ No-Authentication handler is registered to call HttpContext.SigninAsync method, so to do that we have to Inject DI

services.AddAuthentication().AddCookies(“MyCookiesAuth”, options =>{

options.Cookie.Name=”MyCookiesAuth”; // most important

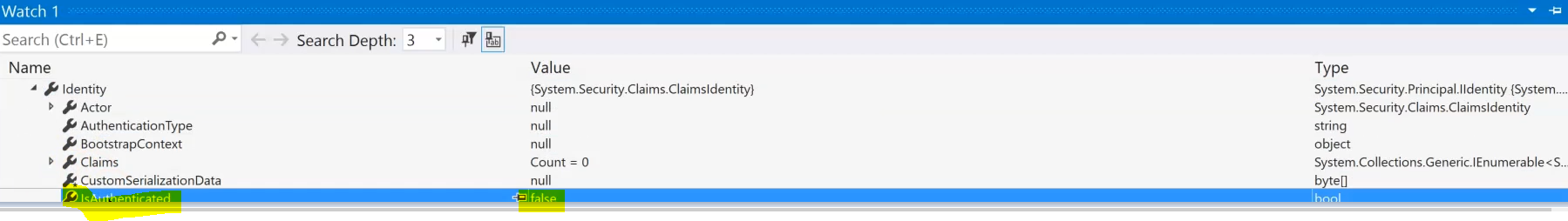
});

This will generate Http Cookie in our Web browser storage , this service also deserialize cookies into ClaimPrincipal Object



Now will send request with Cookie & deserialize cookie .

When we send request to server with cookies , & if we have not added any middleware in request pipeline then application will not read identity and “IsAuthenticated”= false will still be anonymous



So need to attach middle ware to read the cookie.

App.UseAuthentication();

Below line is still there as service and it convert Cookie into ClaimPrinicipal , but still don’t know which cookie have to validate

services.AddAuthentication().AddCookies(“MyCookiesAuth”, options =>{

options.Cookie.Name=”MyCookiesAuth”; // most important

});

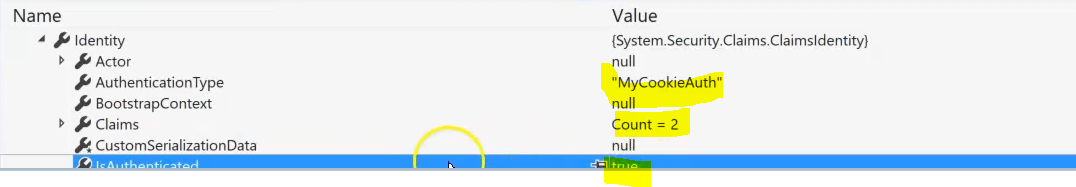
So will have to do as below

services.AddAuthentication(“**MyCookiesAuth**”).AddCookies(“MyCookiesAuth”, options =>{

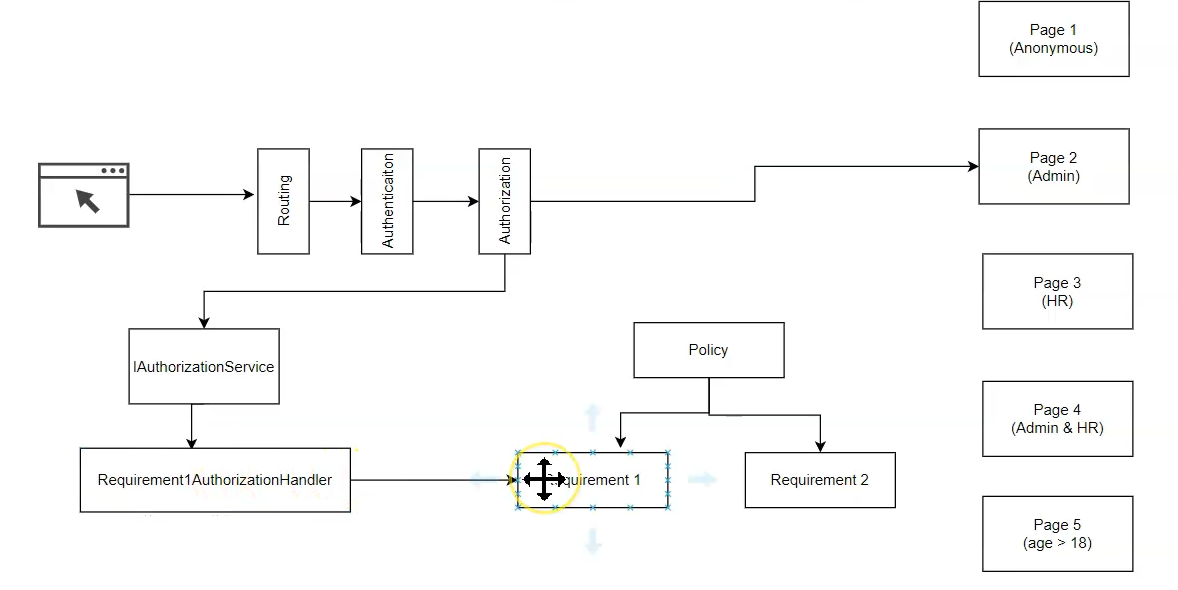
options.Cookie.Name=”MyCookiesAuth”; // most important

});

Then only isAuthenticated =true;



**How Authorization work now ?**

To 

Admin , HR, Admin & HR types of request can be handle by Claim , but for age>18 we have create custom Authorization as above

**Simple Policy based authorization :**

1. To deny anomyous access of all resource in Controller [Authorize] add
2. To redirect on login page

Services.AddAuthentication(“**MyCookieAuth**”).addCookie(“**MyCookieAuth**”,option => {

Option.Cookie.Name=” **MyCookieAuth**”,

Option.LoginPage=”/login”

});

If we remove **app.UseAuthorization()**.

Then throw error , when we try to access action of controller (because we have added [Authorize] attribute ) then “/index” containe Authorize metadata but no middleware found to support that .

In Startup.cs

For claim based Authorization

Service.AddAuthorization(options =>

{

Options.AddPolicy(“**MyClaimedPloicyHRshouldbeinDepartment**”,

Policy => Policy.RequiredClaim(“Department”,”HR”)

});

[Authorize(Policy=” **MyClaimedPloicyHRshouldbeinDepartment**”)]

Public void Index()

{

}

This tell that user must have Department->HR claim to satisfy this policy as defined in startup.cs

But in our claim , we have only Name and Email in Claim policy

var claims = new List<Claim>()

{

new Claim( ClaimTypes.Name, "admin"),

new Claim( ClaimTypes.Email, "admin@admin.com")

};

So when we try to access the page will get accessdened error

When user authenticated then will add claim from Db that ok user will have access of these no of pages , in our case “Department” and value is HR so , Now when user want to access HR page s/he can access that .

var claims = new List<Claim>()

{

new Claim( ClaimTypes.Name, "admin"),

new Claim( ClaimTypes.Email, "admin@admin.com"),

new Claim( “Department”, "HR"),

};

Same like that suppose we have setting page and Admin can access that page

Then in AddAuthorization in Startup.cs

Service.AddAuthorization(options =>

{

Optipons.addPolicy(“**OnlyAdmin**” option => {

Policy =>policy.RequiredClaim(“Role”,”Admin”)

}),

Options.AddPolicy(“**MyClaimedPloicyHRshouldbeinDepartment**”,

Policy => Policy.RequiredClaim(“Department”,”HR”)

});

And

var claims = new List<Claim>()

{

new Claim( ClaimTypes.Name, "admin"),

new Claim( ClaimTypes.Email, "admin@admin.com"),

new Claim( “Role”, "Admin"),

};

Now in Setting Contoller

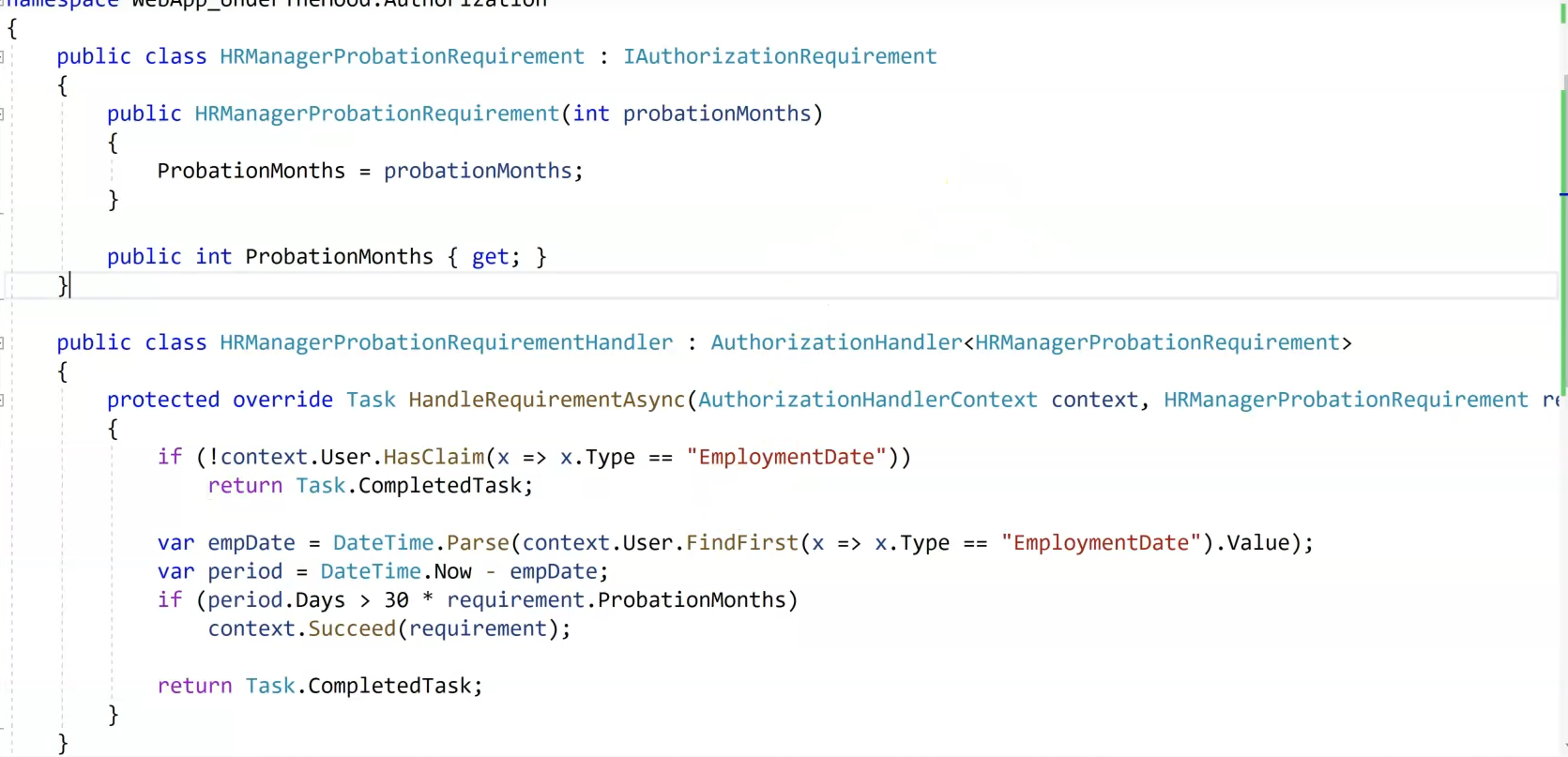
[Authorize(Policy=” **OnlyAdmin**”)]

Public void Index()

{

}

For Depart HR and Admin Role we want to give HR ‘s Some admin section but only when HR ‘s joining date > number of months



In Startup.cs ( need to a Custom class which will implement IAuthorizationRequirement)

Service.AddAuthorization(options =>

{

Options.AddPolicy(“**MyClaimedPloicyHRshouldbeinDepartment**”,

Policy => Policy.RequiredClaim(“Department”,”HR”)

.RequiredClaim(“Manager”,true)

.Requirements.Add(new *HRManagerProbationRequirement*(3))

});

And in Claim

var claims = new List<Claim>()

{

new Claim( ClaimTypes.Name, "admin"),

new Claim( ClaimTypes.Email, "admin@admin.com"),

new Claim( “Role”, "Admin"),

new Claim( “EmployementDate”, "2021-06-20"),

};

Now Cookie & Token

1. Limitation : Cookie can not go out of domain boundary , like if we want to implement

Single Signon we can not use that because SingalSingon implemented by other application /server so we have to implement Token based Authentication/authorization

Suppose we have created a web api and want to consume in our application

Services.AddHttpClient(“**mywebapi**”, client => {

Client.BaseAddress = new Uri(“url of web api”);

});

In controller class Dependecy Injection of IHttpClientFactory object

Then in action

Var httpclient= objectofDI.CreateClient(“**mywebapi**”);

httpclient.GetFromJsonAsync<List<DTOS>>(“ControllerActionName”);

**JWT (JSOn Web Token )**

**{ Header , Payload , Signature}**

**Claim -> Hash Algorimed + Key –> Generate Hashed Claimed Token**

User client(intruder) send request with token and don’t have “Self signed secret key “ s/he will not able able to create a valid token to send on server /api to get access.

Public string CreateToken(IEnumberale<Claim> claims , DateTime expiresat)

{

Var jwt = new JwtSecurityToken({

Claims: claims,

Expires:expiresAt,

SigningcRedential : new SiginingCredential(

New SymmetricSecurityKey(),

SecurityAlgorithm.HmacSha256

)

});

Return new JwtSecurityTokenHandler().WriteToken(jwt);

}

var claims = new List<Claim>()

{

new Claim( ClaimTypes.Name, "admin"),

new Claim( ClaimTypes.Email, "admin@admin.com"),

new Claim( “Role”, "Admin"),

new Claim( “EmployementDate”, "2021-06-20"),

};

DateTime expiersAt = DateTime.AddMinutes(1);

Return Ok(

new {

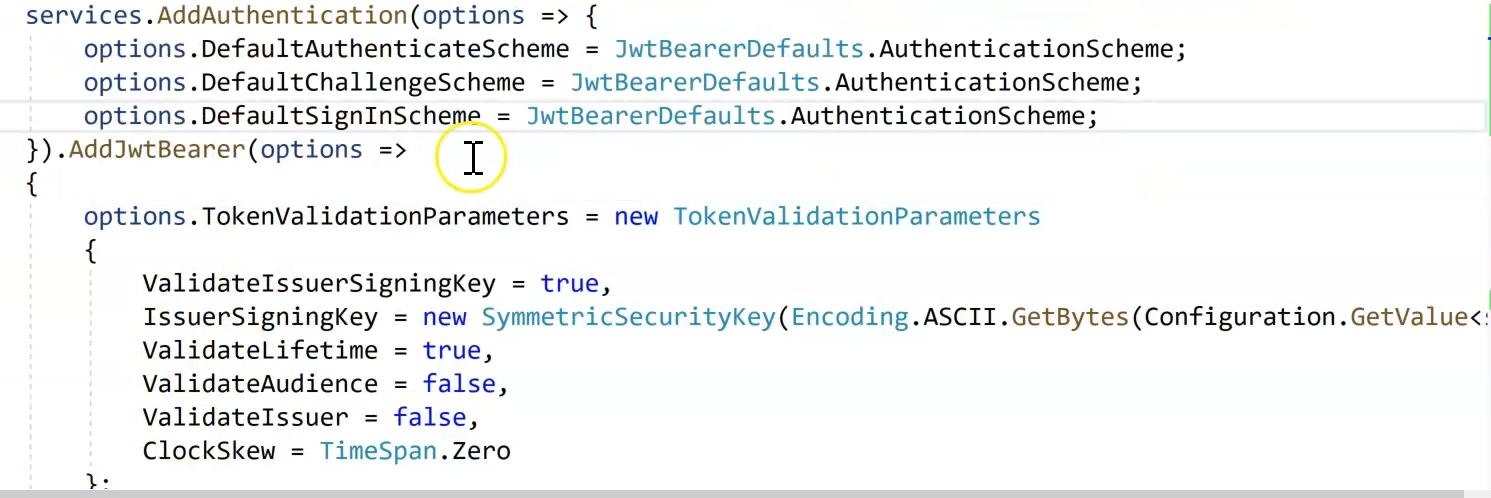
accesstoken= CreateToken(claims, expiersAt);

expiresAt: expiersAt

});

So JwtToken , takes claim , use Algorithm to generate Token with then help of Symmetric key .

In Startup.cs at Web api Level



This is same Signing key which client have

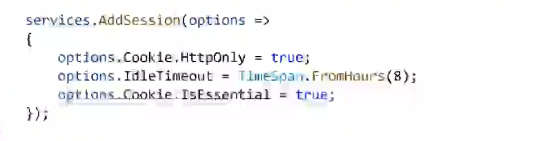
If we have multiple issue of JWt token , we can chaining to add with multiple scheme name.

Calling WebAPI with token



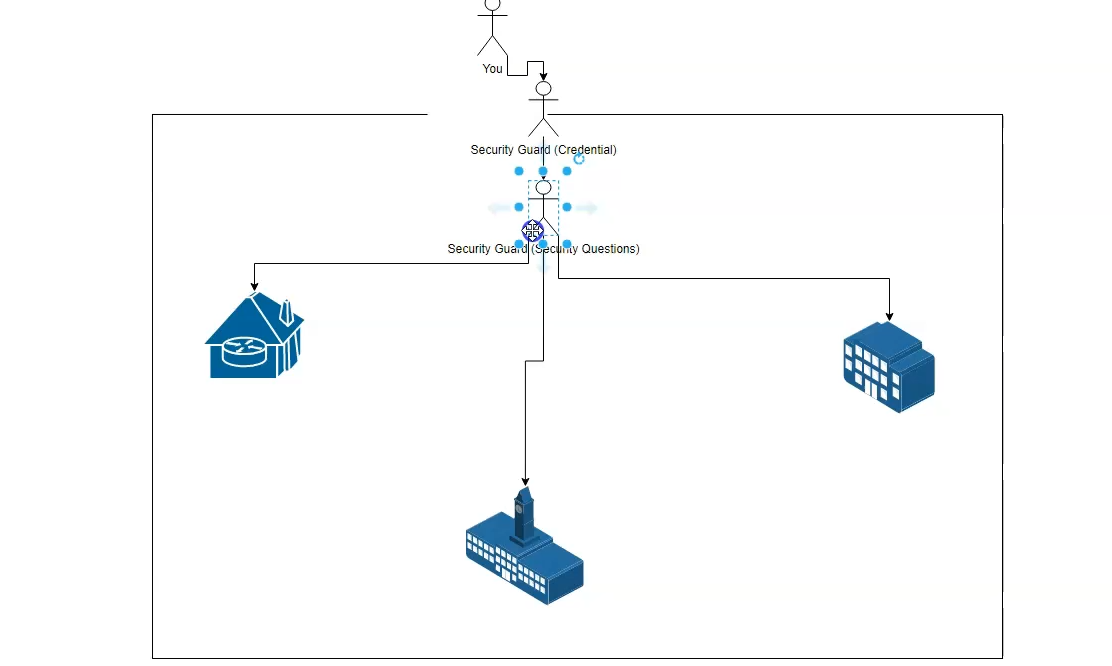
1. Here get WebAPI token’s Httpclient
2. Post with username and password to API “/auth” and get AccessToken as JWT
3. Then Read Response
4. Deserialize Response token and fetch AccessToken
5. Then Set in Authorization in header of Request for new Resource Request .

At WebApplication level we can save that token in session to use further to avoid round trip to



& app.UseSession();

**MFA ( MultiFactor Authentication )**



Login with SocialMedia (thirdparty using Oauth)