

ASP.NET MVC Labo Week 9

Doelstelling:

- Opzetten ASP.NET Identity binnen web applicaties
- Opzetten basis Web API Security
- ADO.NET leren gebruiken vanuit ASP.NET MVC
- Herhaling van de reeds gekende concepten zoals Presentation Model of ViewBag

Inleiding

Deze oefening zal verder bouwen op de oefening van vorige week (8) de dropbox oefening. Zorg ervoor dat deze volledig af is voor u start met dit labo. De bedoeling van dit labo is een WPF client te schrijven die via ASP.NET Web API de logs kan opvragen en weergeven in een datagrid. Het is natuurlijk niet de bedoeling dat iedereen zomaar deze Web API kan gebruiken. We gaan deze dan ook beveiligen met een login en paswoord systeem.

Stap 1

Voor dag gebruikers kunnen inloggen via de WPF applicatie moeten ze eerst bestaan in de database. Zoals reeds gezegd in de theorie gaan we GEEN gebruik maken van ASP.NET Identity maar gaan we een tabel "ExternalUsers" aanmaken in de SQL server. Daarin gaan we gebruikers bijhouden die mogen inloggen via de WPF toepassing. Het scherm om externe gebruikers te maken is enkel toegankelijk voor gebruikers die "Administrator" zijn in de website. De klasse die we hiervoor gaan gebruiken is:

```
11 references
public class ExternalUser
{
    0 references
    public int Id { get; set; }
    4 references
    public string Username { get; set; }
    4 references
    public string Password { get; set; }
    1 reference
    public bool Blocked { get; set; }
}
```

Voeg deze klasse toe aan een nieuw project zodat we deze kunnen hergebruiken in ons WPF project. Daarna moet u instaat zijn om het volgende scherm zelfstandig te ontwikkelen:

Nieuwe externe gebruiker toevoegen

ExternalUser

UserName

Password

Blocked ☐

Create

© 2014 - My DropBox

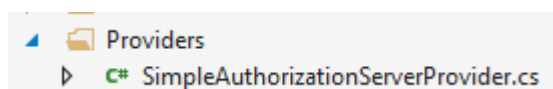
UserName	Blocked	
test	<input type="checkbox"/>	Edit
login	<input type="checkbox"/>	Edit

Als het wegschrijven lukt zou u de gebruikers moeten zien in de database:

	Id	UserName	Password	Blocked
1	1	test	test	0
2	2	login	login	0

Stap 2

In deze stap gaan we de OAuth security opzetten zodat enkel gekende gebruikers kunnen inloggen in de WebAPI. Voeg een map "Providers" toe met daarin volgende klasse:



Ga naar de file Startup.Auth.cs in de map App_Start en voeg onderstaande code toe:

```
app.UseOAuthAuthorizationServer(new OAuthAuthorizationServerOptions()
{
    //Demo stuff
    AllowInsecureHttp = true,
    TokenEndpointPath = new PathString("/token"),
    AccessTokenExpireTimeSpan = TimeSpan.FromHours(8),
    Provider = new SimpleAuthorizationServerProvider()
});

app.UseOAuthBearerAuthentication(new OAuthBearerAuthenticationOptions());
```

Ga nu naar de file “SimpleAuthorizationServerProvider.cs” en voorzie volgende code.

```
1 reference
public class SimpleAuthorizationServerProvider : OAuthAuthorizationServerProvider
{
    0 references
    public override async Task ValidateClientAuthentication(OAuthValidateClientAuthenticationContext context)
    {
        context.Validated();
    }

    0 references
    public override Task GrantResourceOwnerCredentials(OAuthGrantResourceOwnerCredentialsContext context)
    {
    }
}
```

Het is in deze klasse dat we moeten controleren of de gebruiker al dan niet bestaat in de database. Indien gebruiker NIET bestaat moet je weigeren via volgende code:

```
context.Rejected();
return Task.FromResult(0);
```

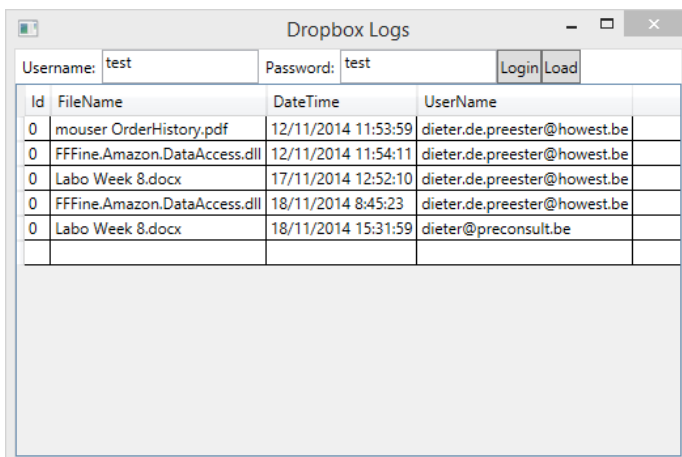
Indien de gebruiker wel bestaat moeten we een nieuw “ClaimsIdentity” opgeven en terugkeren. De code die je daarvoor nodig hebt ziet er als volgt uit:

```
var id = new ClaimsIdentity(context.Options.AuthenticationType);
id.AddClaim(new Claim("username", context.UserName));
id.AddClaim(new Claim("role", "user"));
context.Validated(id);
return Task.FromResult(0);
```

Als laatste stap aan de server kant moeten we een nieuwe API Controller toevoegen met als naam LogController. Deze controller moet je beveiligen met het [Authorize] attribuut en zal één Get() bevatten. Deze Get() zal een lijst van ALLE logs terugkeren. U zou zelf instaat moeten zijn om deze methode te schrijven. Verplaats ook de klasse “FileLog.cs” naar het aparte project zodat we deze klasse kunne hergebruiken binnen onze WPF Applicatie.

Stap 3

In deze stap maken we de WPF applicatie die de logs file zal weergeven. U moet zelf instaat zijn volgende opmaak te kunnen ontwikkelen:



Id	FileName	DateTime	UserName
0	mouser OrderHistory.pdf	12/11/2014 11:53:59	dieter.de.preester@howest.be
0	FFFine.Amazon.DataAccess.dll	12/11/2014 11:54:11	dieter.de.preester@howest.be
0	Labo Week 8.docx	17/11/2014 12:52:10	dieter.de.preester@howest.be
0	FFFine.Amazon.DataAccess.dll	18/11/2014 8:45:23	dieter.de.preester@howest.be
0	Labo Week 8.docx	18/11/2014 15:31:59	dieter@preconsult.be

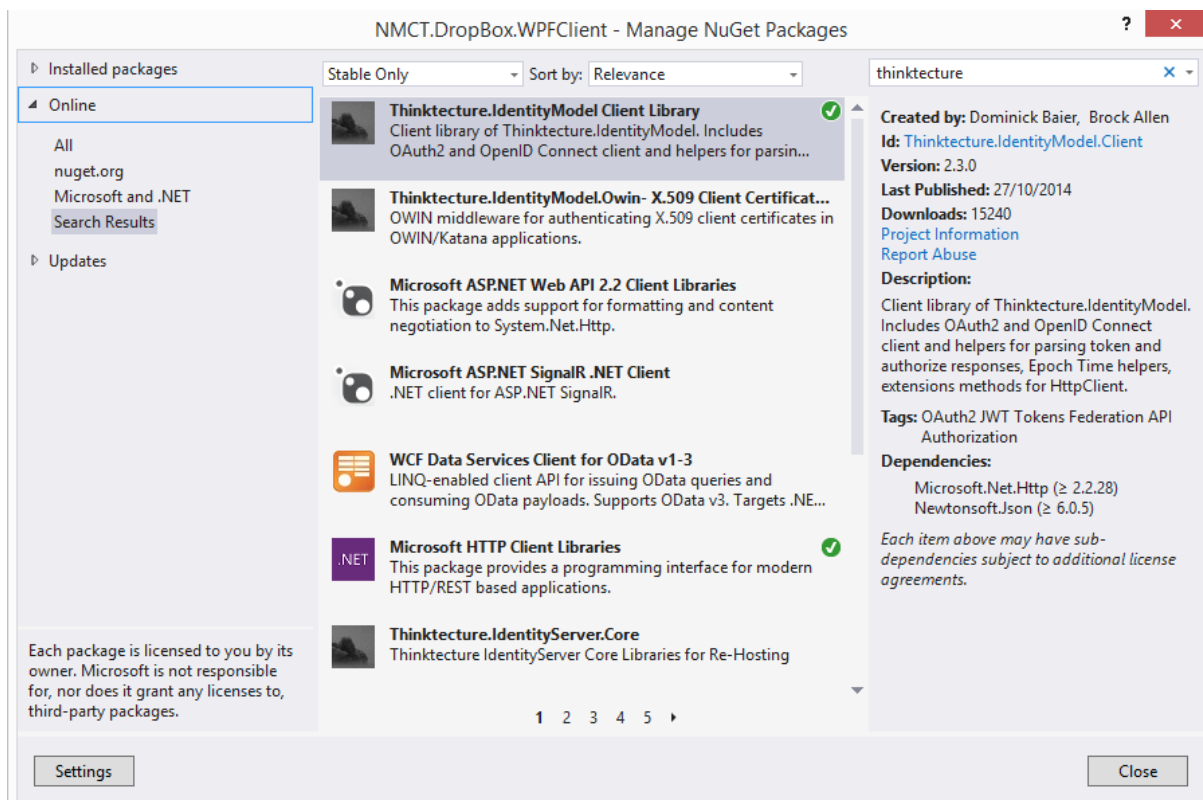
Na het maken van de UI voegen we een file Webaccess.cs toe. Deze file zal de aanroepen naar de webservice bevatten. Maak alvast een constante string URL aan met daarin de host name van je website + de poort. Voeg ook onderstaande methodes toe:

```
2 references
public class WebserviceAccess
{
    private const string URL = "http://localhost:8143/";

    1 reference
    public TokenResponse GetToken(string userName, string password)
    {
    }

    1 reference
    public async Task<List<FileLog>> GetLogs(string token)
    {
    }
}
```

Voor we de methode GetToken() uitwerken moeten we eerst een oAuth2 client library toevoegen via nuget. Voeg Thinktecture.IdentityModel Client library toe aan het WPF project.



U zou nu instaat moeten zijn om beide methode zelf verder uit te werken, maak gebruik van de theorie demo indien nodig. Zorg er ook voor dat de knop voor het inladen van de logs enkel zichtbaar is als we een geldige token hebben. Maak hiervoor gebruik van de "IsError" property van het token object.