

ASP.NET MVC Labo Week 8

Doelstelling:

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

Oefening 1:

We gaan een online dropbox systeem schrijven waar je bestanden kan opladen en andere mensen toegang kan geven tot die bestanden om ze te downloaden.

Stap 1: Connectionstring

Maak een nieuw ASP.NET MVC project aan (geen empty) en vink zowel MVC als Web API. Zorg ervoor dat authentication op “Individual User Accounts” staat. Voor deze oefening gaan we gebruik maken van een full SQL Server. Wijzig de connectionstring uit de web.config zodat deze verwijst naar jouw SQL Server lokaal. De connectionstring bestaat uit:

- Data Source = naam van de SQL Server instantie
- Initial Catalog = naam van de database
- Integrated Security mag je op true plaatsen

De naam van de connectionstring mag je laten staan op “DefaultConnection”

```
<connectionStrings>
  <add name="DefaultConnection" connectionString="Data Source=.;Initial Catalog=DropBox;Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Stap 2: Uitbreiden profiel

Standaard is er reeds een registratie systeem aanwezig binnen de applicatie. We gaan dit een stuk uitbreiden zodat we wat extra informatie kunnen vragen bij het registreren. Open de file “IdentityModel.cs” uit de map models. Daarin zitten 2 klassen:

- ApplicationUser: stelt de ingelogde gebruiker voor
- ApplicationDbContext: zorgt voor connectie met de database

Het is de klasse “ApplicationUser” die we gaan uitbreiden met volgende profiel velden:

- Name, FirstName, Address, City, Zipcode, TwitterName

```

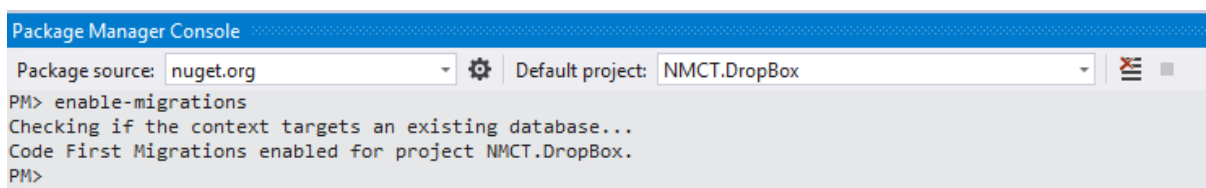
15 references
public class ApplicationUser : IdentityUser
{
    3 references
    public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)...

    0 references
    public string Name { get; set; }
    0 references
    public string FirstName { get; set; }
    0 references
    public string Address { get; set; }
    0 references
    public string City { get; set; }
    0 references
    public string Zipcode { get; set; }
    0 references
    public string TwitterName { get; set; }
}

```

Stap 3: Aanmaken database

ASP.NET Identity maakt gebruik van Entity Framework voor de aanmaak van de databases. In semester 4 gaan we hier dieper op ingaan. We moeten echter een aantal basis concepten gebruiken om toch verder te kunnen met deze oefening. Het eerste wat we gaan doen is het activeren van “Code First Migrations”. We doen dit door de “Package Manager Console” te activeren via “Tools → Nuget Package Manager → Package Manager Console”. In dez console kunnen we nuget commando’s tikken. Activeer migraties door het commando “enable-migrations” te gebruiken. Via migraties kunnen we de database aanmaken maar achteraf ook makkelijk veranderingen aanbrengen. Meer info , semester 4.

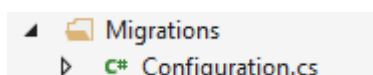


```

Package Manager Console
Package source: nuget.org
Default project: NMCT.DropBox
PM> enable-migrations
Checking if the context targets an existing database...
Code First Migrations enabled for project NMCT.DropBox.
PM>

```

Dit commando activeert migraties. Als dit lukt zal er een map “Migrations” aangemaakt worden in het project met daarin de file “Configuration.cs”



Deze file bevat configuratie info over de migratie waaronder de “Seed” methode. Deze methode zal aangeroepen worden als we een migratie uitvoeren. Open deze file en ga naar de “Seed” methode. In deze methode zullen we de twee rollen aanmaken in de database en één administrator. We zullen het e-mail adres ook gebruiken als username aangezien dit uniek is. De seed methode ziet er als volgt uit:

0 references

```
protected override void Seed(NMCT.DropBox.Models.ApplicationDbContext context)
{
    string roleAdmin = "Administrator";
    string roleNormalUser = "User";
    IdentityResult roleResult;

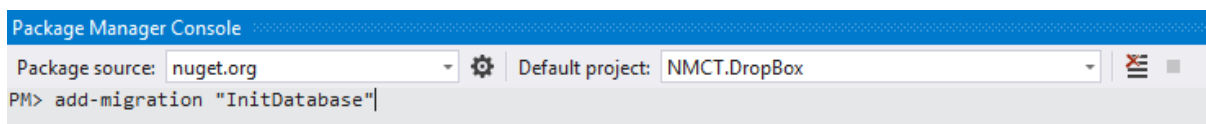
    var RoleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

    if (!RoleManager.RoleExists(roleNormalUser))
    {
        roleResult = RoleManager.Create(new IdentityRole(roleNormalUser));
    }

    if (!RoleManager.RoleExists(roleAdmin))
    {
        roleResult = RoleManager.Create(new IdentityRole(roleAdmin));
    }

    if (!context.Users.Any(u => u.Email.Equals("dieter.dp@gmail.com")))
    {
        var store = new UserStore<ApplicationUser>(context);
        var manager = new UserManager<ApplicationUser>(store);
        var user = new ApplicationUser()
        {
            Name = "De Preester",
            FirstName = "Dieter",
            Email = "dieter.de.preester@howest.be",
            UserName = "dieter.de.preester@howest.be",
            Address = "Graaf Karel De Goedelaan 1",
            City = "Kortrijk",
            Zipcode = "8500",
            TwitterName = "@nmct"
        };
        manager.Create(user, "-Password1");
        manager.AddToRole(user.Id, roleAdmin);
    }
}
```

Voer daarna het commando “add-migration” gevolgd door de naam van de migratie.



Er zal een migratie file verschijnen in de map “Migrations”. Open deze file eens en bekijk de code. Het is deze code die de database zal aanmaken binnen de SQL Server. Het enige wat we nu nog moeten doen is de migratie uitvoeren. Dit doen we door het commando “update-database” uit te voeren.

```
Package Manager Console
Package source: nuget.org
Default project: NMCT.DropBox
PM> update-database
Specify the '-Verbose' flag to view the SQL statements being applied to the target database.
Applying explicit migrations: [201411070839044_InitDatabase].
Applying explicit migration: 201411070839044_InitDatabase.
Running Seed method.
PM>
```

Als je nu kijkt in de SQL Server dan zou er een database moeten staan met daarin de ASP.NET Identity tabellen met daarin de administrator (AspNetUsers) en de twee rollen (AspNetRoles). Controleer dit !!

Stap 4: Aanpassen registratie scherm

Nieuwe gebruikers moeten wat extra profiel informatie opgeven bij het registreren. Hiervoor moeten we het registratiescherm aanpassen. We starten met het aanpassen van het ViewModel die we gebruiken binnen dit registratiescherm. Open de file AccountViewModel.cs in de map models en ga naar de klasse RegisterViewModel. Pas deze aan zodat we de profiel data opvragen bij registratie:

```
1 reference
public class RegisterViewModel
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    2 references
    public string Email { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    1 reference
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    0 references
    public string ConfirmPassword { get; set; }

    [Required]
    0 references
    public string Name { get; set; }
    [Required]
    0 references
    public string FirstName { get; set; }
    [Required]
    0 references
    public string Address { get; set; }
    [Required]
    0 references
    public string City { get; set; }
    [Required]
    0 references
    public string Zipcode { get; set; }
    0 references
    public string TwitterName { get; set; }
}
```

Na het aanpassen van het gebruikte viewmodel moeten we ook nog de HTML code aanpassen zodat de juiste controls verschijnen. Open de file "Register.cshtml" uit de map Views/Account. U zou dit nu zelf moeten kunnen aanpassen zodat de juiste controls verschijnen in het registreer scherm. Het scherm moet er als volgt uitzien:

Register.

Create a new account.

- The Name field is required.
- The FirstName field is required.
- The Address field is required.
- The City field is required.
- The Zipcode field is required.
- The Email field is required.
- The Password field is required.

Name	<input type="text"/>
FirstName	<input type="text"/>
Address	<input type="text"/>
City	<input type="text"/>
Zipcode	<input type="text"/>
Email	<input type="text"/>
Password	<input type="password"/>
Confirm password	<input type="password"/>
TwitterName	<input type="text"/>
<input type="button" value="Register"/>	

De laatste stap in het registreren is het aanpassen van de “Register” (POST) methode in de accountcontroller. Deze methode zal de gebruiker effectief gaan registreren in het systeem. Wijzig deze methode zodat de profiel velden ook opgeslagen worden. Voeg ook de nieuwe aangemaakte gebruiker toe aan een role “User”.

```
[HttpPost]
[AllowAnonymous]
[ValidateAntiForgeryToken]
0 references
public async Task<ActionResult> Register(RegisterViewModel model)
{
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser
        {
            Name = model.Name,
            FirstName = model.FirstName,
            Address = model.Address,
            City = model.City,
            Zipcode = model.Zipcode,
            TwitterName = model.TwitterName,
            UserName = model.Email,
            Email = model.Email
        };

        var result = await UserManager.CreateAsync(user, model.Password);
        if (result.Succeeded)
        {
            UserManager.AddToRole(user.Id, "User");
            await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
            return RedirectToAction("Index", "Home");
        }
        AddErrors(result);
    }
    return View(model);
}
```

Registreer nu een nieuwe gebruiker en controleer in de database of de nieuwe gebruiker aanwezig is en gekoppeld is aan de role. Als dit lukt registreer je nog minstens één gewone gebruiker.

Stap 5: Opladen van bestanden

Het scherm om bestande op te laden ziet er als volgt uit:

Dropbox
Files
Upload

Selecteer een bestand

Omschrijving voor bestand:

Bestand:

Choose file
No file chosen

Wie mag het bestand downloaden

Vannieuwenhuyse - Johan
De Preester - Dieter
Gaillez - Michael

Opladen...

De gebruiker moet een beschrijving opgeven voor het bestand, het bestand kiezen en één of meerdere gebruiker opgeven (multiselect) die toegang hebben tot dit bestand.

Voeg een nieuwe controller toe met als naam FilesController. Deze controller zal gebruikt worden voor zowel het opladen als downloaden van files. Zorg ervoor dat he ingelogd moet zijn om deze controller te gebruiken. Hoe doe je dit?

Maak volgende methode aan in deze controller. Deze methode zal het scherm voor het opladen terugkeren. In dit scherm moet je één of meerdere gebruikers kunnen kiezen (zie bovenstaande screenshot). Deze gebruikers kan je ophalen via de ApplicationDbContext. Na de aanmaak van dit object kan je via de property "Users" een lijst van gebruikers opvragen. Zoek nu zelf hoe je deze kan doorgeven naar de view zodat je ze daar kan weergeven in een multiselect listbox.

```
[HttpGet]
0 references
public ActionResult Upload()
{
    var context = new ApplicationDbContext();
    ViewBag.Users = context.Users;
    return View();
}
```

Schrijf nu zelf de view zodat deze er uit ziet als in het bovenstaande scherm. Het kiezen van de file die u wenst op te laden kan u doen via

```
@using (Html.BeginForm("Upload", "Files", FormMethod.Post, new { enctype = "multipart/form-data" }))
```

.....

```
<input type="file" name="file" />
```

Als id voor de gebruikers in de multiselect gebruiken we de UserName

We moeten nu ook een action maken op de server die aangeroepen zal worden als we de view terugsturen POSTEN. Maak een nieuwe methode Upload met volgende parameters (de namen kunnen anders zijn bij, dit hangt af van de gekozen namen in de view):

```
[HttpPost]
0 references
public ActionResult Upload(HttpPostedFileBase file, string description, string[] wie)
```

Bovenstaande methode heeft 3 parameters:

- ⇒ HttpPostedFileBase: dit is de file die je wenst op te laden
- ⇒ De omschrijving
- ⇒ Een array van geselecteerde gebruikers.

De files die we opladen slaan we NIET op in de database maar in de map "~/app_data/uploads". Het wegschrijven ziet er als volgt uit: Kijken of de file niet null is en of er content aanwezig is. Daarna halen we de naam van de file op. Als laatste moeten we de locatie bepalen waar we de file wensen op te slaan. Webserver werken met relatieve paden. We moeten deze omzetten naar absolute paden. Dit doen we via Server.MapPath. Daarna kunnen we de file op de goede locatie opslaan.

```
[HttpPost]
0 references
public ActionResult Upload(HttpPostedFileBase file, string description, string[] wie)
{
    if (file != null && file.ContentLength > 0)
    {
        var fileName = Path.GetFileName(file.FileName);
        var path = Path.Combine(Server.MapPath("~/app_data/uploads"), fileName);
        file.SaveAs(path);
    }
    return RedirectToAction("Index");
}
```

We kunnen nu wel de file opslaan op de webserver maar we moeten dit ook ergens registreren. We gaan dit doen in SQL Server. Om zaken op te slaan en later weer op te vragen maken we gebruik van een klasse FileRegistration. Deze klasse ziet er als volgt uit:

```
19 references
public class FileRegistration
{
    4 references
    public int FileId { get; set; }
    4 references
    public string Description { get; set; }
    6 references
    public string FileName { get; set; }
    4 references
    public DateTime UploadTime { get; set; }
    2 references
    public string UserName { get; set; }
}
```

“FileId” is een autonummer veld in de SQL Server die zal oplopen. “Description” is tekst, “FileName” is ook tekst, “UploadTime” is het tijdstip van opladen en “UserName” is de persoon die deze file heeft opgeladen. Maak de juiste tabel aan in SQL Server met de juiste velden. Kies als tablenaam de naam van de klasse. Je maakt deze aan in dezelfde database waar de ASP.NET Identity Users reeds zitten.

Column Name	Data Type	Allow Nulls
FileId	int	<input type="checkbox"/>
Description	nvarchar(MAX)	<input checked="" type="checkbox"/>
FileName	nvarchar(MAX)	<input checked="" type="checkbox"/>
UploadTime	datetime	<input checked="" type="checkbox"/>
UserName	nvarchar(MAX)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Naast het registreren van de opgeladen file moeten we ook nog bijhouden wie toegang heeft tot deze file. Maak hier voor een tweede model aan, FileUser

0 references

```
public class FileUser
{
    0 references
    public int FileId { get; set; }
    0 references
    public string UserName { get; set; }
}
```

	Column Name	Data Type	Allow Nulls
►	FileId	int	<input checked="" type="checkbox"/>
	UserName	nvarchar(MAX)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Nu kunnen we starten met het schrijven van onze eigen database code.

Maak een mapje "DataAccess" toe aan je project en voeg de reeds gekende klasse Database.cs toe vanuit je Business Applications module. Voeg ook een nieuwe file DAFileRegistration.cs toe. Deze file zal de nodig methodes bevatten die we nodig hebben voor lezen/schrijven naar de database. Bovenaan plaatsen we een cons string die verwijst naar de naam van connectionstring uit de web.config. Daarna schrijven we de methode SaveFileRegistration met de parameters:

- fileName
- Description
- userName

Deze methode zal registeren die welke file heeft opgeladen en op welk tijdstip.

```
private const string CONNECTIONSTRING = "DefaultConnection";

1 reference
public static int SaveFileRegistration(string fileName, string description, string userName)
{
    string sql = "INSERT INTO FileRegistration VALUES(@Description,@FileName,@UploadTime,@UserName)";
    DbParameter par1 = Database.AddParameter(CONNECTIONSTRING, "@Description", description);
    DbParameter par2 = Database.AddParameter(CONNECTIONSTRING, "@FileName", fileName);
    DbParameter par3 = Database.AddParameter(CONNECTIONSTRING, "@UploadTime", DateTime.Now);
    DbParameter par4 = Database.AddParameter(CONNECTIONSTRING, "@UserName", userName);
    return Database.InsertData(CONNECTIONSTRING, sql, par1, par2, par3, par4);
}
```

Nu kunnen we deze methode aanroepen vanuit de controller na het opladen van de file. We slaan ook het ID op van de geregistreerde file. We hebben deze nodig om op te slaan welke gebruikers deze file mogen downloaden.

```
[HttpPost]
0 references
public ActionResult Upload(HttpPostedFileBase file, string description, string[] wie)
{
    if (file != null && file.ContentLength > 0)
    {
        var fileName = Path.GetFileName(file.FileName);
        var path = Path.Combine(Server.MapPath("~/app_data/uploads/"), fileName);
        file.SaveAs(path);
        int id = DAFileRegistration.SaveFileRegistration(fileName, description, User.Identity.Name);
    }
    return RedirectToAction("Index");
}
```

De volgende stap is opslaan welke van de geselecteerde gebruikers uit de view toegang hebben tot de file. De geselecteerde gebruikers bevinden zich in de `string[]` die we als parameter binnenkrijgen in de actionmethode. We moeten deze array overlopen en voor iedere persoon in de array slaan we zijn username op samen met het Id van de file. Hiervoor moet u nu zelf de methode `SaveDownloaders(user,id)` aanmaken.

```
[HttpPost]
0 references
public ActionResult Upload(HttpPostedFileBase file, string description, string[] wie)
{
    if (file != null && file.ContentLength > 0)
    {
        var fileName = Path.GetFileName(file.FileName);
        var path = Path.Combine(Server.MapPath("~/app_data/uploads"), fileName);
        file.SaveAs(path);
        int id = DAFileRegistration.SaveFileRegistration(fileName, description, User.Identity.Name);

        foreach (string user in wie)
            DAFileRegistration.SaveDownloaders(user, id);
    }
    return RedirectToAction("Index");
}
```

Stap 6: Weergeven van te downloaden files

Dit scherm ziet er als volgt uit:

DropboxFilesUploadHello dieter.de.preester@howest.beLog off

Mijn bestanden

Description	FileName	UploadTime	UserName
Test	backup startpage.txt	12/11/2014 11:19:33	Download Delete

Bestanden waar ik toegang tot heb

Description	FileName	UploadTime	UserName
test	FFFine.Amazon.DataAccess.dll	7/11/2014 13:47:16	Download
Test	backup startpage.txt	12/11/2014 11:19:33	Download

Het bovenste stuk “Mijn bestanden” zijn de bestanden de ingelogde gebruiker heeft opgeladen. Het stuk “Bestanden waar ik toegang tot heb” zijn de bestanden die niet van de ingelogde gebruiker zijn maar waar hij welk toegang tot heeft en kan downloaden. U moet nu instaat zijn om deze view volledig zelf te maken. Het downloaden van de file en het verwijderen zit in de volgende stappen.

Stap 7: Downloaden van een file

Het downloaden van de file doen we door als parameter mee te geven de file die we wensten te downloaden. Daarna halen we een `FileRegistration` object op zodat we weten over welk bestand het gaat. Daarna moeten we terug het volledig path opbouwen naar de file. De laatste stap is de file terugkeren naar de browser. Dit doen we door hem volledig in te lezen, een mimetype mee te geven en de naam.

[HttpGet]

0 references

```
public FileResult Download(int id)
{
    FileRegistration fileRegistration = DAFileRegistration.GetFileRegistrationById(id);
    string path = Server.MapPath("/app_data/uploads/");
    path += fileRegistration.FileName;
    return File(System.IO.File.ReadAllBytes(path),
        System.Net.Mime.MediaTypeNames.Application.Octet, fileRegistration.FileName);
}
```

Stap 8: Verwijderen van een file

Het is enkel mogelijk eigen opgeladen bestanden te verwijderen. De gebruik kiest delete in het overzichtsscherm en krijgt dan nog eens expliciet de vraag of het bestand verwijderd mag worden.

Dropbox Files Upload

Wenst u deze file te verwijderen ?

FileRegistration

FileId	12
Description	fdf
FileName	mouser OrderHistory.pdf
UploadTime	12/11/2014 11:32:46
UserName	

Delete

Daarna moet de record verdwijnen uit de database en de file van de webserver.

Stap 9: Administrator role mag alle files zien.

Zorg ervoor dat de gebruikers in de “Administrator” role files zien en deze ook kunnen downloaden.

Extra:

- Als iemand een file download moet je dit gaan loggen in de database. Denk zelf na welke je info zeker moet opslaan. Maak de klasse aan, de tabel en schrijf een log weg als iemand een file download.
- Zorg voor een derde menu item “Logs” die enkel zichtbaar is voor gebruikers in de Administrator role. Dit menu item zal een weergave zijn van log tabel.