Model ASP.NET MVC Core 3.1 Web App -Deel 2

Eindresultaat: <https://github.com/SyntraWestKortrijk/PittigRestoMVC-DEEL2>

# Start project

In deel 1 zijn we gekomen tot deze code: <https://github.com/SyntraWestKortrijk/PittigRestoMVC-DEEL1>

We starten vanaf deze code

**Dit hebben we in Deel 1 reeds gedaan tijdens de virtuele les:**

* Aanmaken van ASP.Net Core Web applicatie met MVC architectuur design pattern
* opzetten configuratie Sql Server database , gebruik van ORM Entity FrameWork Core

en migraties opzetten

* Opzetten Authenticatie en Authorisatie met ASP.Net Identity en beheren van Individuele User Accounts in SQL Server.
* Gebruik van LINQ to Entities om gegevens op te halen uit de database
* Asynchroon aangroepen van methoden
* Aanmaken van models voor Category, Subcategory, menuitem en ApplicationUser
* Aanmaken van controllers voor CRUD operaties op Category
* Aanmaken van Views voor Create/Read/Update/Delete (CRUD) van een Category in de database

In dit deel (Deel 2) gaan we het eerste deel snel herhalen en

* modellen, controllers en views toevoegen voor de Coupon, Subcategory, MenuItem, ApplicationUser
* Areas gebruiken en de routing configuratie aanpassen

# Aanmaken models voor SubCategory,MenuItem, Coupon en ApplicationUser

**Onder de folder models, voeg een nieuwe class toe voor SubCategory, MenuItem, Coupon en ApplicationUser**

public class SubCategory

{

[Key]

public int Id { get; set; }

[Display(Name = "Naam Sub categorie")]

[Required]

public string Name { get; set; }

[Required]

[Display(Name ="Categorie")]

public int CategoryId { get; set; }

[ForeignKey("CategoryId")]

public virtual Category Category { get; set; }

}

public class MenuItem

{

public int Id { get; set; }

[Required]

[Display(Name = "Naam")]

public string Name { get; set; }

[Display(Name = "Beschrijving")]

public string Description { get; set; }

[Display(Name = "Pittigheid")]

public string Spicyness { get; set; }

public enum ESpicy { NA=0, Geen=1, Weinig=2, Sterk=3}

[Display(Name = "Foto")]

public string Image { get; set; }

[Display(Name="Categorie")]

public int CategoryId { get; set; }

[ForeignKey("CategoryId")]

public virtual Category Category { get; set; }

[Display(Name = "Sub categorie")]

public int SubCategoryId { get; set; }

[ForeignKey("SubCategoryId")]

public virtual SubCategory SubCategory { get; set; }

[Range(1,int.MaxValue, ErrorMessage = " Prijs moet hoger zijn dan €{1}")]

[Display(Name = "Prijs (EUR)")]

public double Price { get; set; }

}

public class Coupon

{

[Key]

public int Id { get; set; }

[Required]

[Display(Name = "Naam")]

public string Name { get; set; }

[Required]

[Display(Name = "Type Korting")]

public string CouponType { get; set; }

public enum ECouponType { Percent = 0, Euro = 1 }

[Required]

[Display(Name = "Korting")]

public double Discount { get; set; }

[Required]

[Display(Name = "Minimum bedrag")]

public double MinimumAmount { get; set; }

[Display(Name = "Afbeelding")]

public byte[] Picture { get; set; }

[Display(Name = "Is Actief")]

public bool IsActive { get; set; }

}

public class ApplicationUser : **IdentityUser**

{

public string Name { get; set; }

public string StreetAddress { get; set; }

public string City { get; set; }

public string State { get; set; }

public string PostalCode { get; set; }

}

## Aanpassen ApplicationDbContext

**Voeg aan de class ApplicationDbContext DbSet public properties toe voor SubCategory, MenuItem, Coupon en ApplicationUser**

public DbSet<SubCategory> SubCategory { get; set; }

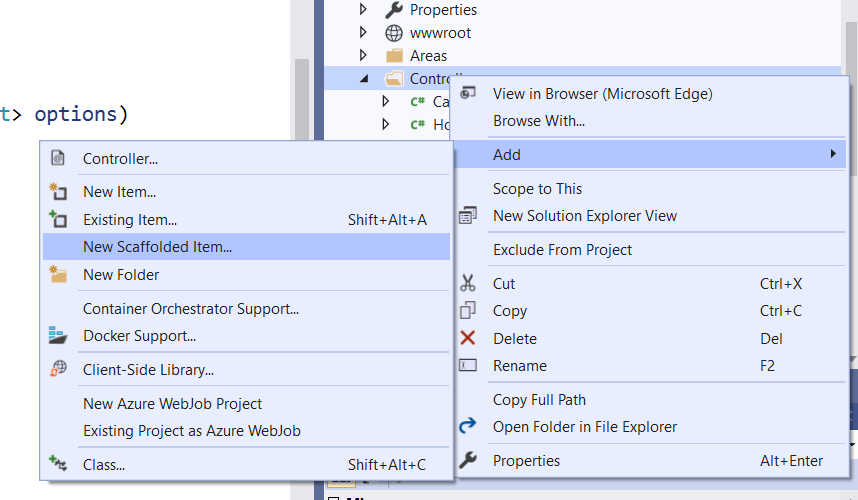
public DbSet<MenuItem> MenuItem { get; set; }

public DbSet<Coupon> Coupon { get; set; }

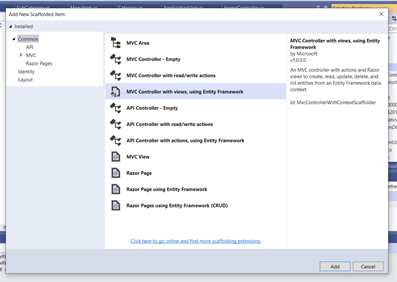
public DbSet<ApplicationUser> ApplicationUser { get; set; }

# Aanmaken Controllers en Views voor CRUD operaties voor SubCategory, MenuItem en Coupon

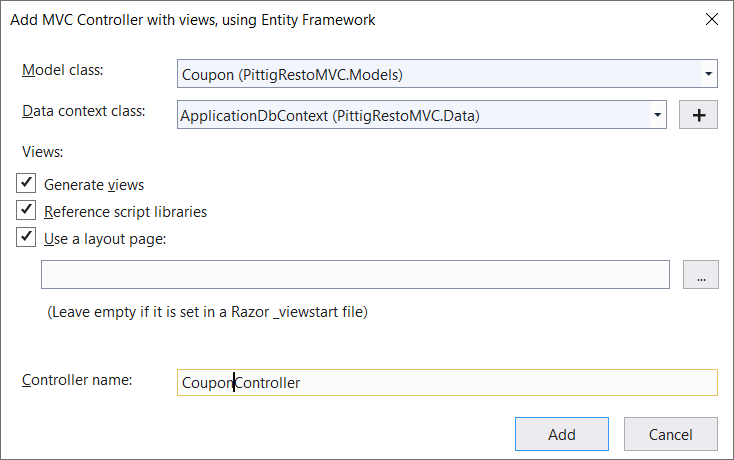
Rechtsklik op de folder Controller en kies Add/New Scaffolded Item…



En daarna Mvc Controller with views, using Entity Framework



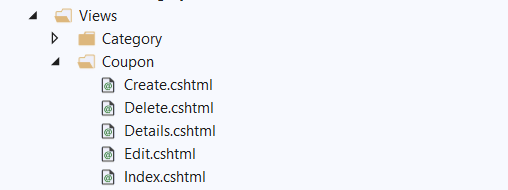
Kies in het volgende popup-venster bv als **model class Coupon**



**Data class ApplicationDbContext**

**Controller name CouponController**

Zorg dat de **Generate views aangevinkt** staat, deze zal dan onder de Views/Coupon folder Razor views genereren voor een Coupon (Create.cshtml, Delete.cshtml, Edit.cshtml enIndex.cshtml,Details.cshtml) die een standaard UI voorzien waarin een gebruiker CRUD operaties kan verrichten voor Coupons.



Onder de folder controller is er eveneens een CouponController.cs toegevoegd

**Doe nu hetzelfde voor de SubCategory en MenuItem: Maak een**

**Controller CategoryController en MenuItemController aan en views via Add/New scaffolded item**

**Test de applicatie via deze urls:**

[**https://localhost:44365/Category/index**](https://localhost:44365/Category/index)

[**https://localhost:44365/Coupon/index**](https://localhost:44365/Coupon/index)

[**https://localhost:44365/SubCategory/index**](https://localhost:44365/SubCategory/index)

[**https://localhost:44365/MenuItem/index**](https://localhost:44365/MenuItem/index)

# Database migraties toevoegen voor de nieuwe tabellen en database updaten

Indien je applicatie goed werkt, kan je nu de tabellen in de database creëren.

We doen dit via de **NuGet Package Manager Console** door eerst data-migraties op te zetten en daarna update-database commando

PM> add-migration AddedSubCatMenuCouponAppUser

PM> update-database

**Test de applicatie nogmaals en test de CRUD operaties op coupon,category,subcategory en MenuItem**

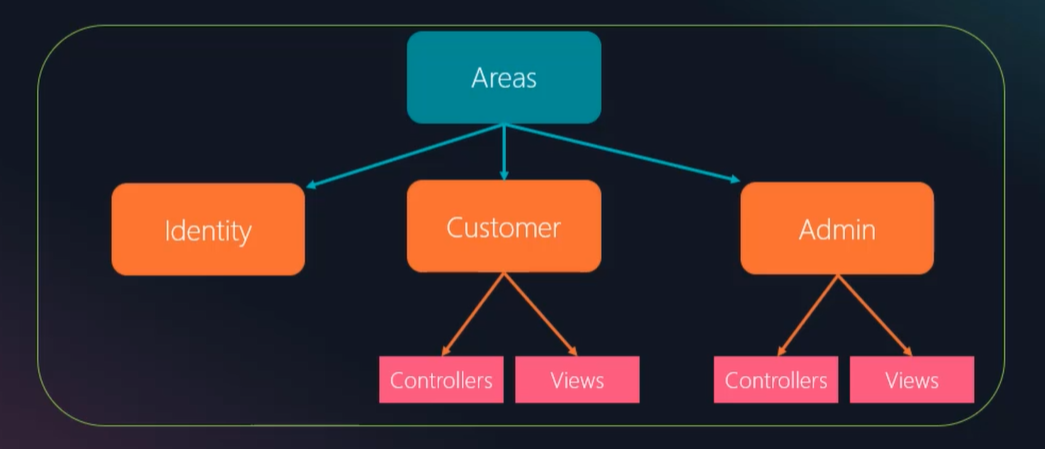
# Herorganiseren van Controllers en Views in Areas

Bij grotere projecten worden de controllers en views ondergebracht onder apparte Area-folders.

Dit is om bv het gedeelte voor de content management die enkel door Admin mag worden gedaan in een Admin-area folder onder te brengen en de customer controllers en views onder te brengen in een andere Customer-area folder.

We gaan dus ons folder structuur hiervoor en ook routing systeem moeten aanpassen:

Onder de **folder Areas** vind je nu enkel Identity (Deze is automatisch gegenereerd bij creatie van het project wanneer we Authenticatie hebben aangezet). Dit zijn Razor pages en deze hebben geen corresponderende Controller class.

We gaan nu een nieuwe folder structuur maken, zodat we deze stuctuur krijgen:

**Maak een nieuwe folder Customer en Admin onder Areas**

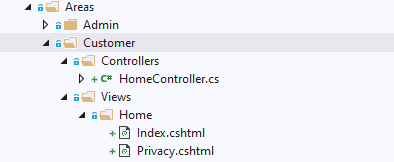
**Maak** **onder de Customer en Admin folder de folders** **Controllers en Views**

# Verplaatsen van controllers en views

Verplaats **alle controllers en vie**ws voor de Category, Coupon,MenuItem en SubCategory onder de **Areas folder Admin**

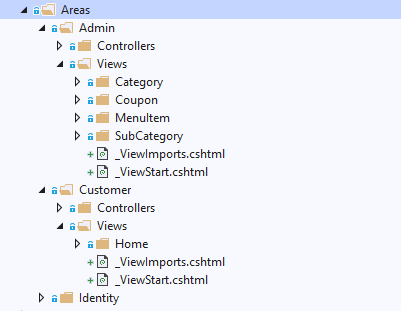
## 

Verplaats eveneens de HomeController en bijhorende views naar de folder **Areas/Customer**

****

**Maak een kopie van de bestanden Views/\_ViewsImports.cshtml en Views/\_ViewStart.cshtml**

**Naar de folders Admin/Views en Controller/Views**

****

**Wijzig de inhoud van Areas/Admin/Views/\_ViewStart.cshtml en Areas/Customer/Views/\_ViewStart.cshtml naar deze inhoud:**

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

# Controller classes onderverdelen in areas

Open elke controller class en zet het attribuut [Area("Admin")]

Boven elke Controller die onder de folder Admin staat: de Categorycontrolller, couponController, MenuItemController en SubCategoryController

[Area("Admin")]

public class CategoryController : Controller

….

Open de HomeController.cs file en zet eveneens het Area attribuut boven de class

[Area("Customer")]

public class HomeController : Controller

{….

## Aanpassen routing bij Areas

In het routing systeem zal nu een extra niveau bijkomen, namelijk de locatie van de area folder, deze komt voor de controller folder

**Open de startup.cs file en verander het default pattern bij de endpoints in de methode Configure**

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "areas",

pattern: **"{area=Customer}/{**controller=Home}/{action=Index}/{id?}");

endpoints.MapRazorPages();

});

**Start de applicatie en test nu de applicatie met deze urls:**

[**https://localhost:44365/Customer/Home/index**](https://localhost:44365/Customer/Home/index)

[**https://localhost:44365/Admin/Category/index**](https://localhost:44365/Admin/Category/index)

[**https://localhost:44365/Admin/SubCategory/index**](https://localhost:44365/Admin/SubCategory/index)

[**https://localhost:44365/Admin/Coupon/index**](https://localhost:44365/Admin/Coupon/index)

[**https://localhost:44365/Admin/MenuItem/index**](https://localhost:44365/Admin/MenuItem/index)

# Toevoegen navigatie in shared Layout

**Open** de Shared razor layout file van de home-pagina: **Views/Shared/\_Layout.cshtml**

Wijz**ig deze code**

<partial name="\_LoginPartial" />

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Home</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</a>

</li>

</ul>

**Door**

<**partial** **name**="\_LoginPartial" />

<ul class="navbar-nav flex-grow-1">

**<li class="nav-item dropdown text-white-50">**

**<a class="nav-link dropdown-toggle" href="#" id="navbarDropDownMenuLink" role="button"**

**data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">**

**Inhoud beheren**

**</a>**

**<div class="dropdown-menu" aria-labelledby="navbarDropDownMenuLink">**

**<a class="dropdown-item" asp-action="Index" asp-controller="Category" asp-area="Admin">categorie</a>**

**<a class="dropdown-item" asp-action="Index" asp-controller="SubCategory" asp-area="Admin">subcategorie</a>**

**<a class="dropdown-item" asp-action="Index" asp-controller="MenuItem" asp-area="Admin">Menu</a>**

**<a class="dropdown-item" asp-action="Index" asp-controller="Coupon" asp-area="Admin">Kortingen</a>**

**<a class="dropdown-item" asp-action="Index" asp-controller="User" asp-area="Admin">Personeel</a>**

**</div>**

**</li>**

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Home</a>

</li>

<li class="nav-item">

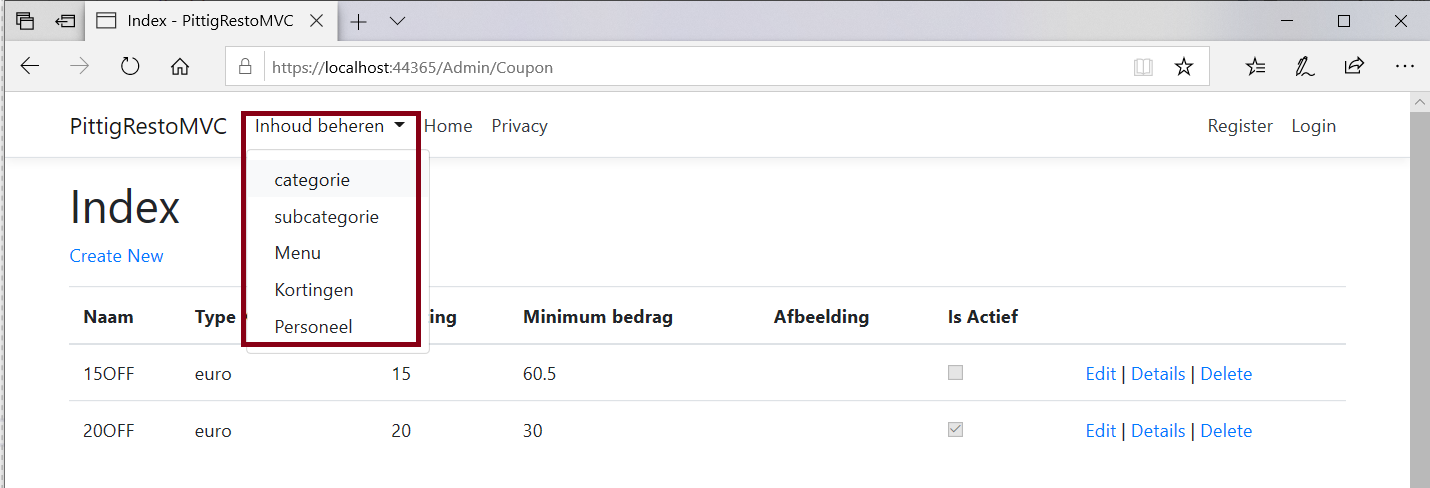
<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</a>

</li>

</ul>

**Deze code zal een dropdown list met navigatie links in de header sectie tonen van elke pagina die deze shared layout gebruikt**

**Start de app op en test de dropdown list met navigatie links**

****

# Tonen van Categorieën,coupons en menuitems op de homepagina

We moeten op de homepagina gegevens uit verschillende entiteiten en tonen en dus uit tabellen van de database halen.

Een goede manier om dit te doen is om een ViewModel class te maken die de gegevens uit verschillende tabellen zal bijhouden. In de HomeController vullen we de properties van een object van deze ViewModel in en halen de gegevens op vanuit. Ten slotte geeft de Controller dit ViewModel object door aan de View.

We maken een nieuwe folder Models/ViewModels om onze ViewModel class te plaatsen.

**Maak een folder ViewModels onder de folder Models**

**Maak onder ViewModels folder** een nieuwe class **IndexViewModel**

public class IndexViewModel

{

public IEnumerable<MenuItem> MenuItem { get; set; }

public IEnumerable<Category> Category { get; set; }

public IEnumerable<Coupon> Coupon { get; set; }

}

**Open de HomeController.cs**

**Verwijder eerst deze code:**

~~private readonly ILogger<HomeController> \_logger;~~

~~public HomeController(ILogger<HomeController> logger)~~

~~{~~

~~\_logger = logger;~~

~~}~~

We hebben **gegevens nodig uit de ApplicationDbContext**. Deze is reeds in de ConfigureServices in startup.cs geregistreerd als service in de DI Container van ASP.Net Core

We kunnen deze service **nu injecteren in onze HomeController en we gaan dit doen via Constructor-injection:**

**Maak hiervoor een private field aan waarin we een ApplictionDbContext object kunnen bijhouden.**

private readonly ApplicationDbContext \_db;

**En een constructor die deze als parameter aanneemt aan de private field initialiseert:**

public HomeController(ApplicationDbContext db)

{

\_db = db;

}

**Pas nu de Index() methode aan. Deze wordt uitgevoerd bij een HttpGet Request naar de Home-pagina**

public async Task<IActionResult> Index()

{

**IndexViewModel IndexVM = new IndexViewModel()**

**{**

**MenuItem = await \_db.MenuItem.Include(m =>**

**m.Category).Include(m =>m.SubCategory).ToListAsync(),**

**Category = await \_db.Category.ToListAsync(),**

**Coupon = await \_db.Coupon.Where(c => c.IsActive == true).ToListAsync()**

**};**

return View(IndexVM);

}

**Opmerking: Aangezien we nu asynchrone calls maken in de Index methode, moet deze ook asynchroon worden**

public async Task<IActionResult> Index()

Hier wordt een object van IndexViewModel aangemaakt en de Menus, Category en Coupons worden door asynchrone aanroepen uit de database gehaald en ingevuld in de public properties van het object.

**Ten slotte wordt het object meegegeven aan de View (standaard heeft de view dezelfde naam als de action method, dus in dit geval zal dit dus de view Index.cshtml zijn onder de folder Area/Customer/Home/**

**Opmerking:** De .Include() in LINQ to Entities bij bv\_db.MenuItem, Category en SubCategory

Is nodig omdat **EF Core Lazy Loading** gebruikt en de gegevens uit gerefereerde tabellen niet automatisch zal opladen. Dus indien je eveneens gegevens wil ophalen van de tabel die gelinkt is met een FK relatie, moet je dit expliciet vermelden in de LINQ Query. Indien je dit niet dus, kan je null waarden terugkrijgen voor de gelinkte data.

# Pas View van homepagina aan

**Open de Index.cshtml onder** **Area/Customer/Home/**

De View ontvangt een object van Type IndexViewModel van de HomeController’s Index() action method. We specifiëren bovenaan deze pagina het type van het object dat binnenkomt dat ons Model zal zijn in deze view:

@model Pittig.Models.ViewModels.IndexViewModel

In de View gaan we een lijst van Categorieën tonen:

<ul id="menu-filters" class="menu-filter-list list-inline text-center">

<li class="active btn btn-secondary ml-1 mr-1" data-filter=".menu-restaurant">Alles</li>

@foreach (var item in Model.Category)

{

<li class="ml-1 mr-1" data-filter=".@item.Name.Replace(" ",string.Empty)">@item.Name</li>

}

</ul>