OmniVus formulär & DBkoppling

En lathund och exempel på C R U D från webben m.hj.a. .NET ASP MVC

---------- Code First DB ---------

med MVC i sidoprojekt (01_Forms)

Skapa DB-filen

Skapa nytt projekt (i samma solution) ASP .. MVC
Ta fram Server explorer
- välj data connection - add connection
- skapa db-file
- döp den och lägg den i en 'data'-mapp
- Kopiera 'connection-string' (välj properties)

appsettings.json

lägg till connectionsträngen:

```
"Logging": {
        "LogLevel": {
        "Default": "Information",
        "Microsoft . AspNetCore": "Warning"
        }
},
"AllowedHosts": "*"  ,    ← OBS! Glöm inte kommat
        "ConnectionStrings": {
         "Sql" : "<connectionsträngen här>"
        }
}
```

OBS: connectionsträngen får inte innehålla mellanslag i sökvägen!

Installera Entity framework

Starta Packet Manager
välj **rätt project** i VS titlebar
starta NuGet packet Manager Console
välj **rätt project** i 'Default project' i PM

kör:

```
install-package microsoft.entityframeworkcore.sqlserver
install-package microsoft.entityframeworkcore.tools
```

modellen Entities

skapa ny mapp i models : 'Entities'
Skapa klass: ContactFormEntity
lägg in:

```
using System.ComponentModel.DataAnnotations;

public class ContactFormEntity
{
        [Key]
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        [Required]
        public string Email { get; set; }
        [Required]
        public string Message { get; set; }
}
```

Quick action med markören på [Key] och välj Using System … (se ovan)

SqlContext

skapa klass 'SqlContext' i databasmappen

public SqlContext ska ärva DbContext (QuickAction!)
skapa en tom constructor (ctor)
skapa en constructor med options (markera klassnamnet och kör QuickAction - SqlContext)
infoga <sqlContext> (se nedan)
skapa metoden DbSet

Ska bli som så här:

```
using _01_Forms.Models.Entities;
using Microsoft.EntityFrameworkCore;

public class SqlContext : DbContext
{
public SqlContext()
{
}
public SqlContext (DbContextOptions<SqlContext> options) : base(options)
{
}
public DbSet<ContactFormEntity > ContactForms { get; set; }
}
```

Detta knyter Formuläret till DB'n

Spara

Program.cs

Lägg till följande:

```
using _01_Forms;
using Microsoft. EntityFrameworkCore;

var builder = WebApplication. CreateBuilder(args) ;
// Add services to the container.
builder . Services . AddControllersWithViews () ;
builder.Services.AddDbContext<SqlContext>(x=>x.UseSqlServer(
builder.Configuration.GetConnectionString("Sql")));
```

QuickAction på AddDbcontext och UseSqlServer för att skapa using .. raderna.

Spara

Add & Update med Packet Manager

Kör kommando:

`Add-Migration "init"`

(skapar xxxx_init.cs som innehåller det som behövs för skapa DB'n)

Kontrollera ovanstående fil om den verka stämma.

`update-database`

(kör cs-filen som skapar DB'n)

Controller med Entity Framework

Skapa en MVC-controller med views

(H-klicka i controllermappen - add controller - with …views)

Välj modellen du skapade tidigare (ContactFormEntity)

Välj context SqlContext (01_Forms)

[ADD]

(Controllern skapas med färdiga metoder för C R U D, create, read, update & delete.
Om problem uppstår kan man starta om hela VS och köra skapa controller igen)

Test

I webläsaren, prova med att lägga till /ContactFormEntities till URL'en

---------- Kodjusteringar i filerna ----------

Formuläret

html - Formulär. Taggar som är lämpliga att ha med

```
<form action="Index" id=contactform> (gör så att knappen pekar på index-sidan)
      <input type="text" placeholder="Enter your name">
      <textarea cols="30" rows="20"  placeholder="Write your message here>
      <button type="submit" name="submitnow" class="btnBlue">Submit</button>
</form>
```

css: `#contactform`

```
width: 90%
```

DefaultController:

```
public IActionResult Index()
{
      ViewData ["Title"] = "Kontakta Oss";
      ViewData [ "ControllerName"] = "Contacts";
      return View();
}
```

```
[HttpPost]
public IActionResult Index(ContactForm model)
{
}
```

Modellen ContactForm:

I mappen models:

Add class 'ContactForm' (Eller varför inte 'modelContactForm')
Högerklicka och välj [using WebApp.Models;] på 'ContactForm'
lägg till return View(); inuti metoden

Contact form - Error messages

För att få svenskt felmeddelanden kan man ändra/lägga till i några filer

Enkel variant

Öppna modellen ContactForm.cs
I Data annotation .ex. [Required] ovanför Name)

- Lägg till "(ErrorMessage= "Du måste ange ett namn")

Mer komplett exempel med placeholders:

```
using System. ComponentModel. DataAnnotations;
namespace WebApp. Models
}
      public class ContactForm
      {
            [Display (Name = "Namn")]
            [Required(ErrorMessage = "Du måste ange ett namn")]
            [StringLength (256, ErrorMessage = "{0}let måste bestå av minst {2}
            tecken", MinimumLength = 2)]
            public string Name { get; set;
      }
}
```

Filen _ContactFormPartial

lägg till:

@model ContacForm
QuickAction: välj 'Using WebApp.Models'
Klipp ur Using WebApp.Models
flytta / klistra in i _VIewPorts.cshtml

Detta för att göra den tillgänglig överallt (globalt i projektet ?).

Lägg till asp-for="xxxx" till input-raderna för att koppla till modellens fält:

Kopiera validationrader från 01_Forms / Create.cshtml
och lägg in under input / textarea-raderna.

Modifiera så du knyter till rätt namn i din modell

Kopiera asp-validation-summary raden också.

```
<h1>Estimate for your projects. </hl>
<div asp-validation-summary="ModelOnly" class="text-danger"></div>
<input asp-for="Name" class="form-control" type="text" placeholder="Enter your name">
<small><span asp-validation-for="Name" class="text-danger"></span></small>
```

Ändra (Om du vill gå till första sidan efter Submit)
så att:

```
[HttpPost]

public IActionResult Index(ContactForm model)
{
      return RedirectToAction("Index", "Default") ;
}
```

Contact Confirm

Alternativt, kan man skapa en kvittensfunktion som aktiveras vid Submit om man byter ut lite kod:

```
[HttpPost]

public IActionResult Index(ContactForm model)
{
      return RedirectToAction("ContactConfirm");
}

public IActionResult ContactConfirm()
{
      return View();
}
```

Kopiera >ContactConfirm<
Högerklicka på View och
Välj: Add View + [ADD]
Klistra in ContactConfirm som namn + [ADD]

Skriv lämplig HTML som t.ex:

```
<h1>Tack för att du valde oss!</h1>

<p>Vi återkommer till dig inom 24 timmar.</p>
```

Modell till databas i (Web) projektet

Skapa en klass ContactFormEntity

Skapa Entity-mappen
kopiera innehållet i klassen i ContactForm.cs
Ta bort alla Data-annotations utom [Required]
QuickAction på Required
Lägg till [Key] public int ID {get; set}
Skapa en constructor:
markera Name, Email & Message
QuickAction: Generate constructor

Installera entityframeworks inkl. tools i detta projekt (se Packet manager ovan)

Skapa klass (SqlContext)
QuickAction: using MS.EntityFrameworkCore
QuickActor på SqlContextklassen: Gen constructor DbContextOptions
ovanför den: ctor<tab><tab>
Sist: public DbSet<ContactFormEntity >
QuickAction på den: Using WebApp…
fyll på med: ContactRequests {get; set;}

i programs.cs
lägg till builder..<SqlContext …. (återigen precis som i 01_forms)

Add-migration.
Kontrollera den genererade filen om den verkar göra det man förväntar sig.
Update-Database

Kod för att skriva informationen från formuläret i DB'n

HttpPost i ContactsController.cs

Lägg till kod för att **inte** skicka data vid formulärfel , utan istället återsända datan till formuläret.

```
[HttpPost]
public async Task<IActionResult> Index(ContactForm model)
{
	If (ModelState. IsValid)
	{
		_context. ContactRequests . Add(new ContactFormEntity (model. Name,
				model. Email, model. Message) ) ;
		await _context. SaveChangesAsync() ;
		return RedirectToAction("ContactConfirm");
	}
	return View(model) ;
}
```

```
public IActionResult ContactConfirm()
return View();
```