Övning 11: MVC - Personalregister

Denna uppgift är relativt enkelt att följa, men ta god tid vid varje steg och se till att ni förstår! I denna applikation skall vi inte lägga stor vikt på effektivitet, optimering eller databasstruktur utan helt fokusera på MVC och hur det fungerar. Uppgiften går ut på att skapa ett personalregister för ett varuhus med olika avdelningar och personalansvar.

Projektet

Innan vi börjar måste vi ha något att arbeta i. Skapa ett nytt MVC5 projekt genom *File -> New -> Project... -> Installed > Visual C# > Web* -> Döp projektet: "EmployeesRegister"-> OK

- Välj *MVC* som template
- Change Authentication -> No Authentication
- "Host in the cloud" och "Add unit tests" skall **EJ** vara valda/kryssade.
- "OK"

Modellen

Det första ni skall göra är en modell för applikationen, den kommer vara relativt enkel och representera en anställd i varuhuset.

1. Skapa en klass i mappen *Models* som ni döper till "*Employee*"
2. Skapa nu följande *properties*:
    a. *int Id*
    b. *string FirstName*
    c. *string LastName*
    d. *int Salary*
    e. *string Position*
    f. *string Department*

Data access, Context

Nu när vi har vår modell skall vi skapa en data-access via en databaskontext.

1. Skapa mappen *DataAccessLayer* via *SolutionExplorer* i *Visual Studio*.
2. Skapa klassen *EmployeesContext* i den nya mappen.
3. I *EmployeesContext.cs* lägg till "*using System.Data.Entity;*"
   Om denna saknas så får du manuellt lägga till den i projektet genom att högerklicka *'References'* i er *solution explorer*. Om detta händer kan det även vara värt att kontrollera att ni har *Entity Framework* installerat i *Nuget Package Manager*
    a. Tools > Nuget Packade Manager > Manage Nuget Packages for solution…
    b. Finns "EntityFramework" med bland de installerade? Bra!
    c. Om inte: Browse > Välj "EntityFramework" > Markera employeeRegister > Install. Klart!
4. Låt *EmployeesContext* ärva från "*DbContext*" (detta går ej utan EntityFramework)
5. Gör en publik *konstruktor* för *EmployeesContext* som anropar *baskonstruktorn* med inputsträngen "*DefaultConnection*"
6. Skapa sedan propertyn "public DbSet<Models.Employee> Employees { get; set; }"

Generera Databasen med Entity Framework

Nu är det hög tid att generera databasen!

1. Öppna package-manager-console (PMC) via Tools -> Nuget/Library package manager -> Package Manager Console
2. Skriv därefter in "Enable-Migrations"
3. Gå in i den genererade Migrations.Configuration-klassen
4. I seed-metoden skall vi fylla på med lite grunddata som vi vill ska finnas i databasen redan från början. Gör detta genom att ta inspiration från den kod som finns i de genererade kommentarerna fast applicera det på er egna *klass* och *DbSet* (Glöm inte att lägga till er modell som *using statement*).
5. När vi är nöjda med seed-metoden skall vi använda oss av kommandot "*Add-Migration Namn*" där namn är ett beskrivande namn för migrationen. Förslagsvis använder ni namnet *Initial* eller *Init* vid er första *migration*.
6. Därefter kör vi direkt *Update-Database* kommandot för att uppdatera och skapa vår *databas*.

Skapa våra kontroller

Nu när vi har allt bakomliggande vi behöver så bör vi, innan vi skapar *front-end*, ha ett sätt att kommunicera mellan *back-end* och *front-end*, alltså en *kontroller*.

1. Högerklicka på *Controllers-mappen*
2. Välj *Add-Controller*
3. Välj "*MVC5 Controller with views using Entity Framework*"
4. Välj er *Employee* klass som *modellklass*
5. Välj er *EmployeesContext* som *Contextklass*
6. *Controllern* bör ha fått namnet *EmployeesController*.
7. "OK"

Funkar det?

När vi kommit såhär långt kan det vara dags att bygga och besöka sidan så att vi ser att det fungerar (om ni inte redan gjort det).

- Debugg > Start Debugging (F5)
- Lägg till "/Employees" på er URL för att se registret

Funkar det?

Lägg till navigation

För att enklare kunna navigera till vårt register vill vi ha en länk i huvudnavigationen.
Lägg till navigation i *headern* för *Employees (Se solution explorer: Views > Shared > _Layout)*.
När ni är där så skriv en bättre rubrik för applikationen också (typ "Employee Register" eller nått)

Views

När ni skapade Controllern i Wizarden så fick ni ett antal Views på köpet (Index,
Nu vill vi skapa en egen vy som endast visar anställda på sport-avdelningen. Det för vi på följande sätt:

1. Gå till er *Controller* i koden.
2. På valfri plats, efter “public ActionResult Index (){...}” lägger ni till en egen ActionResult. “public ActionResult Sport() { return View();} “
3. Högerclicka på Test och välj AddView
4. I wizzarden får ni en mängd val, välj det som passar, i detta fallet List.
5. Skriv in namnet på view:n och klicka på *Add*. En cshtml fil kommer genereras i views-mappen för controllern.
6. Skriv in en LINQ sats och skicka med till vyn som modell, exempelvis

```
var model = db.Employees.Where(i => i.Department == “Sport”).ToList();
return View(model);
```

- Skapa skapa en vy som endast visar avdelningschefer.
- Hitta på en egen vy som skulle vara praktisk i sammanhanget.

Nu då?

Fortsätt att utforska applikationen och gör de justeringar du tycker är passande. Övning ger färdighet och ju mer ni testar desto mer lär ni er och känner er bekväma.