ASP.NET Active Dynamic Web Pages

חזרה על השלבים בעבודה עם Entity Framework

פעולות CRUD עם Entity Framework
ו ASP.NET MVC

בקשות GET/POST
ולידציות צד שרת וצד לקוח

TomerBu

Entity Framework:
מיפוי של אובייקטים לדטה-בייס רלציוני:

Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write.

ארכיטקטורה:

דטה-בייס

קוד צד שרת
API/JSON
שרת יכול לשמור על סודות

קוד צד לקוח
ראקט
WPF
אייפון
מובייל

יושב אצל הלקוח

דטה-בייס

צד שרת משולב צד לקוח
לדוגמא ASP MVC

שרת בונה עמודים מוכנים
ושולח אותם ללקוח

הלקוח מקבל Html

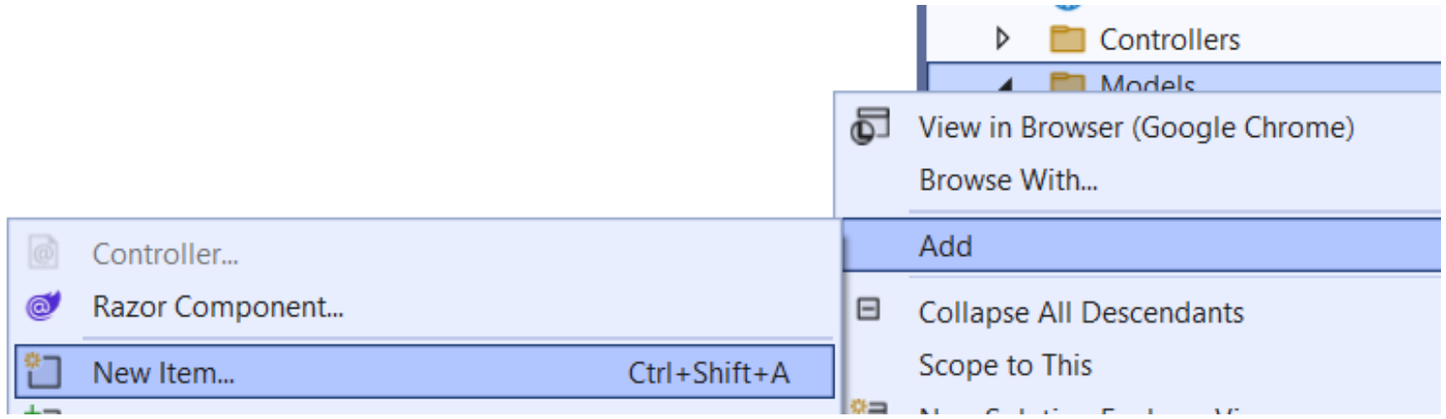
HTML
CSS
JS

יושב אצל הלקוח

אפליקציה
STAND ALONE
DESKTOP

פחות מאובטח
פחות יכולות שינוי מרחוק

חזרה קצרה על נושאים משיעור קודם:



1

יצירת מודל

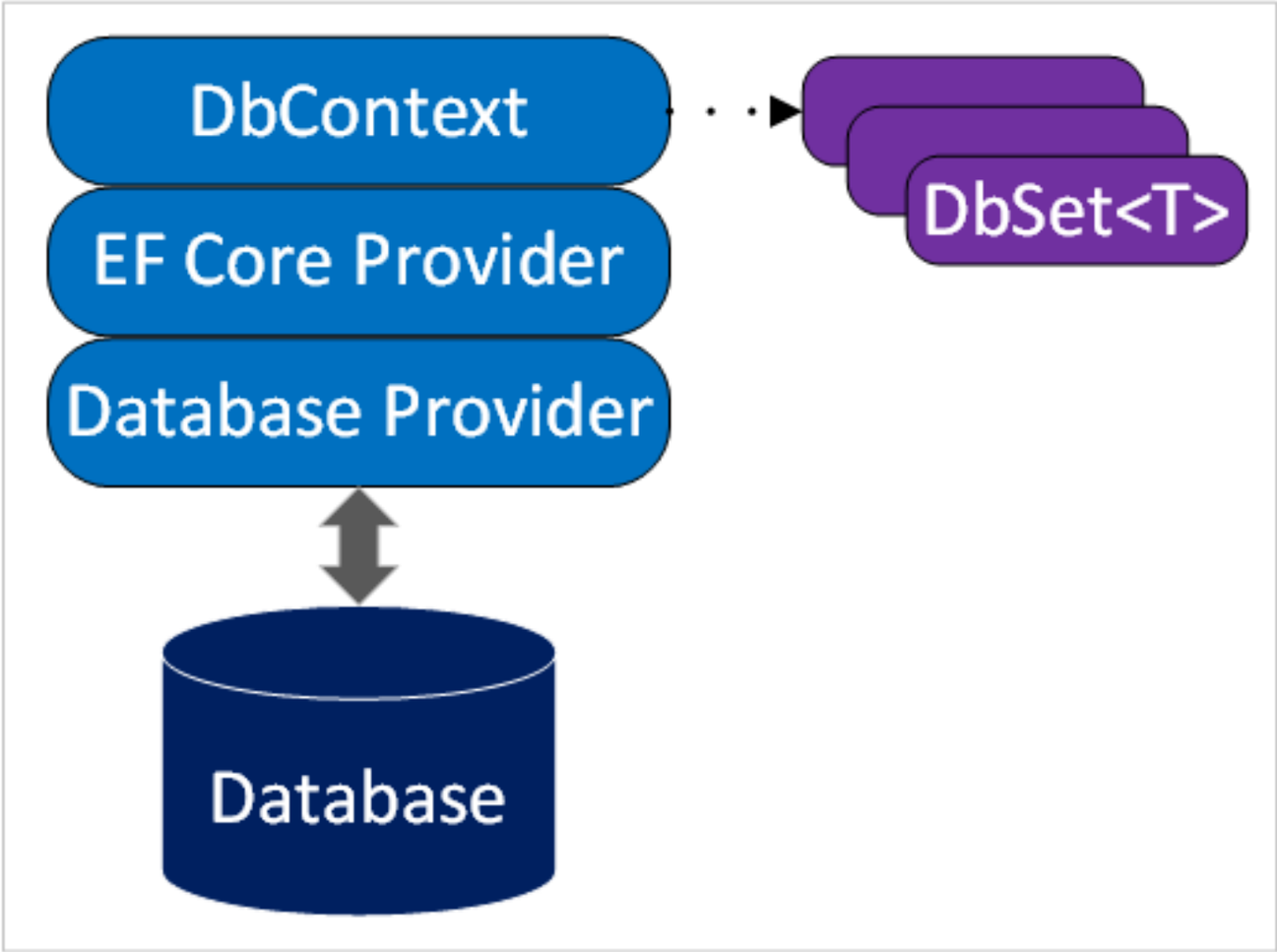
```
using System.ComponentModel.DataAnnotations;

namespace Lec2.Models;

public class Dog
{
    [Key]
    public int Id { get; set; }

    [Required(ErrorMessage ="Foo!"), MinLength(2), MaxLength(20)]
    public required string Breed { get; set; }
}
```

Entity Framework



https://learn.microsoft.com/en-us/ef/core/miscellaneous/connection-strings

Entity Framework
מחלקה עבור DBContext

הוספת אוסף/טבלה לDBContext

Data/Lec2DbContext.cs

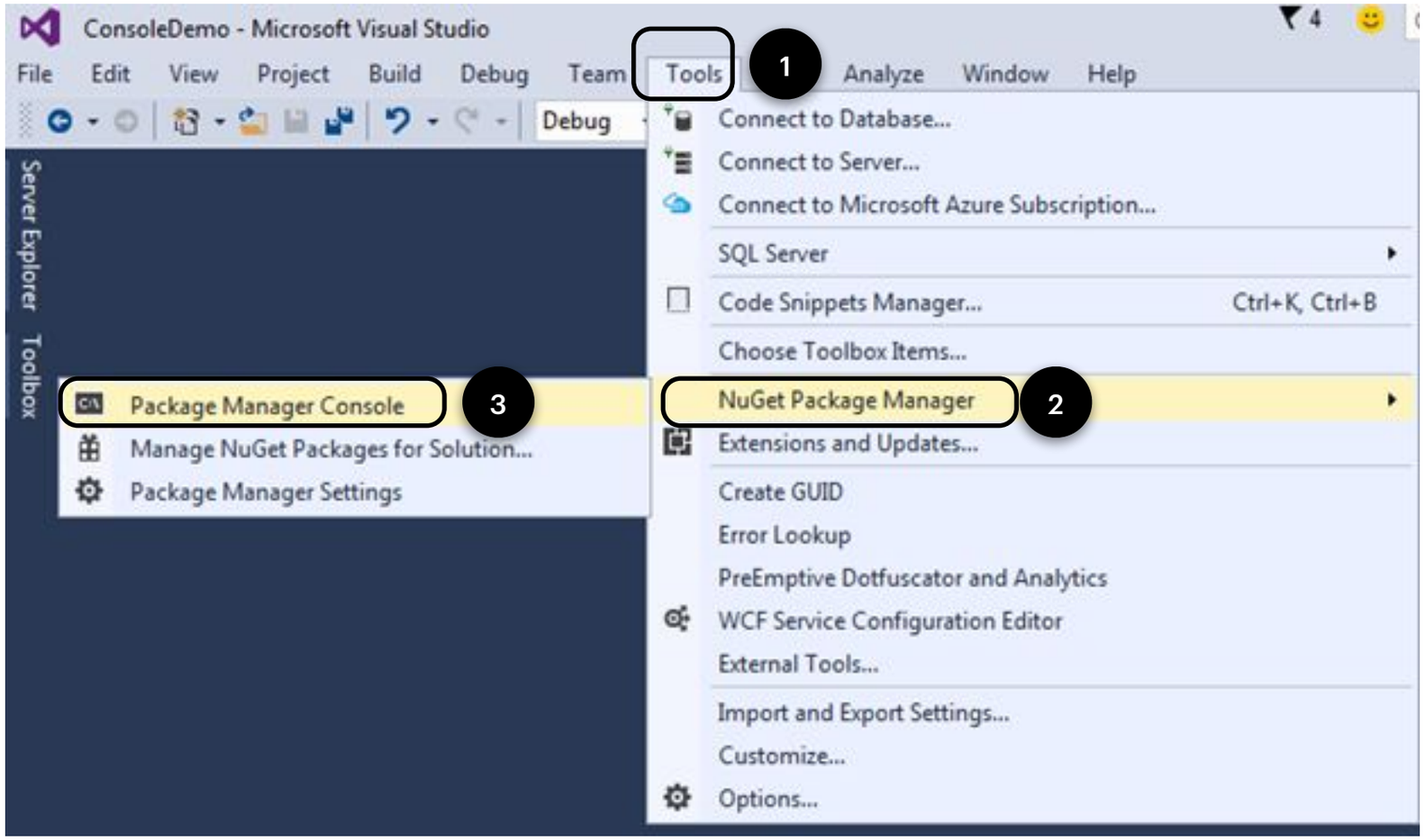
```
namespace Lec2.Data;

using Lec2.Models;
using Microsoft.EntityFrameworkCore;
public class Lec2DbContext(DbContextOptions<Lec2DbContext> options) :
DbContext(options)
{
    //ORM = Object Relational Mapper
    public DbSet<Person> People { get; set; }
    public DbSet<Dog> Dogs { get; set; }
}
```

2

https://learn.microsoft.com/en-us/training/modules/persist-data-ef-core/3-migrations#scaffold-models-and-dbcontext

הרצה של סקריפט לעדכון דטה-בייס בNuget Package Manager Console



הוספנו מודל - ביצענו שינוי במבנה של הOOP שלנו ולכן עלינו לעדכן את המבנה של מסד הנתונים

הרצה של סקריפט ליצירת דטה-בייס בNuget Package Manager Console

```
PM> add-migration AddDogsTable
```

1

Package Manager Console

Package source: All

```
PM> update-database
```

2

כדי ליצור את הטבלה – ניצור מיגרציה
ואז נעדכן את הדטה-בייס:

עכשיו נוצרו הטבלאות בדטה-בייס!

נבצע את 2 השלבים האלה בכל שינוי
לדטה-בייס

DogsController
עבודה עם DBContext - נזריק בבנאי ונשתמש במתודות המובנות לעבודה עם הדטה-בייס

Controllers
DogsController.cs

3 יצירת Controller להגשת התכנים בנושא Dogs

```
using Lec2.Data;
using Lec2.Models;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.ChangeTracking;

namespace Lec2.Controllers;

public class DogsController(Lec2DbContext dbContext) : Controller
{
    //GET /dogs/index
    [HttpGet]
    public async Task<IActionResult> Index()
    {
        var dogs = await dbContext.Dogs.ToListAsync()
        return View(dogs); //Views/Dogs/Index.cshtml
    }
}
```

הזרקה של הDBContext

שליפת המידע מטבלת Dogs
~(SELECT * FROM Dogs)

החזרה של View - קובץ cshtml שיציג את המידע בcshtml

DogsController
עבודה עם DBContext - נזריק בבנאי ונשתמש במתודות המובנות לעבודה עם הדטה-בייס

- Views
  - Dogs
    - Index.cshtml

4

יצירת View - קובץ cshtml שיציג את המידע שהcontroller מושך ממסד הנתונים

```
@model List<Dog>

<h1>Dogs</h1>

<ul>
    @foreach (var dog in Model)
    {
        <li>@dog.Breed</li>
    }
</ul>

<hr />

<a asp-controller="Dogs" asp-action="Create">Add a dog</a>
```

הכרזה על הטיפוס של המודל

שימוש במודל להצגת הנתונים

קישור למתודה Create בController שלנו שנקרא Dogs

המתודה עדיין לא קיימת - ניצור אותה בשקף/שלב הבא

מתודה להצגת טופס Create
הלקוח יגיע למתודה הזאת אחרי שילחץ על הקישור בעמוד הקודם
לחיצה על קישור היא בקשה מסוג HttpGet

```
public class DogsController(Lec2DbContext dbContext) : Controller
{
    //GET /dogs/index
    [HttpGet]
    public async Task<IActionResult> Index()
    {
        var dogs = await dbContext.Dogs.ToListAsync();
        return View(dogs); //Views/Dogs/Index.cshtml
    }

    //GET /dogs/create
    [HttpGet]
    public IActionResult Create()
    {
        return View(); // Views/Dogs/Create.cshtml
    }
}
```

עכשיו עלינו ליצור קובץ .cshtml
בשם Create בתיקיית הViews של Dogs

View להצגת טופס Create
הלקוח יגיע לטופס אחרי שילחץ על הקישור בעמוד Index

טופס מכיל inputs
וטופס מכיל כפתור submit

המודל הוא Dog
בcshtml אם נכריז על מודל
שלא קיבלנו מהcontroller

ברגע שלוחצים על הכפתור - הטופס שולח בקשת POST
לקונטרולר Dogs

המשמעות היא יצירת new Dog

שם המתודה בקונטרולר
(ניצור אותה בעמוד/שלב הבא)

```
@model Dog

<h1>Add your dog</h1>

<form method="post" asp-controller="Dogs" asp-action="Create">
    <div class="form-group mb-4">
        <label asp-for="Breed" class="form-label">Dog Breed</label>
        <input asp-for="Breed"  placeholder="Breed" class="form-control" />
    </div>

    <button class="btn btn-primary">Add</button>
</form>
```

כל הקלדה בטופס תמלא את השדה Breed באובייקט Dog שיצרנו
בלחיצה על כפתור submit
ישלח האובייקט מסוג Dog למתודה Create בController - נכתוב אותה בעמוד הבא

מתודה לטיפול בבקשת POST
המתודה מקבלת אובייקט מסוג Dog ושומרת אותו למסד הנתונים - לאחר שמירה - נעביר את הלקוח לעמוד הבית - שם יוכל לצפות ברשימה העדכנית

```
public class DogsController(Lec2DbContext dbContext) : Controller
{
    //GET /dogs/index
    [HttpGet]
    public async Task<IActionResult> Index()
    {
        var dogs = await dbContext.Dogs.ToListAsync();
        return View(dogs); //Views/Dogs/Index.cshtml
    }

    //GET /dogs/create
    [HttpGet]
    public IActionResult Create()
    {
        return View(); // Views/Dogs/Create.cshtml
    }

    //POST /dogs/create
    [HttpPost]
    public async Task<IActionResult> Create(Dog d)
    {
        //add the dog to the database
        var result = await dbContext.Dogs.AddAsync(d);
        await dbContext.SaveChangesAsync();

        //now we got an id
        //if all is good -> goto dogs list
        return Redirect("/dogs/index");
    }
}
```

המתודה מגיבה לבקשת HTTP POST

המתודה מקבלת Dog

שמירה לדטהבייס

העברה לעמוד הרצוי

ולידציות: בדיקת תקינות של הקלט:

1 הוספת שדה לטופס להצגת הודעות שגיאה

```
@model Dog

<h1>Add your dog</h1>

<form method="post" asp-controller="Dogs" asp-action="Create">
    <div class="form-group mb-4">
        <label asp-for="Breed" class="form-label">Dog Breed</label>
        <input asp-for="Breed"  placeholder="Breed" class="form-control" />
        <span asp-validation-for="Breed" class="text-danger"/>
    </div>

    <button class="btn btn-primary">Add</button>
</form>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

2 הוספת script מוכן שקיים בפרוייקט - זוהי ספרית JS מפורסמת בשם Jquery validation
ואנחנו מוסיפים את הסקריפט רק בעמודים שבהם צריכים ולידציות

ולידציות: בדיקת תקינות של הקלט:

בדיקת תקינות של האובייקט שמתקבל בController
לא נרצה לשמור בדטה-בייס מידע שאינו תקין / לא עבר ולידציה

3

```
//POST /dogs/create
[HttpPost]
public async Task<IActionResult> Create(Dog d)
{
    if (ModelState.IsValid)
    {
        //add the dog to the database
        var result = await dbContext.Dogs.AddAsync(d);
        await dbContext.SaveChangesAsync();

        //now we got an id
        //if all is good -> goto dogs list
        return Redirect("/dogs/index");
    }

    return View(d);
}
```

אם המידע עובר ולידציה - נשמור אותו

אחרת נחזיר את המשתמש לView
שיראה את הודעת השגיאה

שיעורי בית:

צרו מודל חדש בשם Product ובצעו עליו את הפעולות שלמדנו בכיתה:
הצגת כל המוצרים
הצגת פרטים למוצג בודד
הוספת מוצר

פירוט של שיעורי הבית בשלבים:

1. הוסיפו מודל חדש בשם Product

לכל מוצר יש: Id, Name, Price, ImageURL

2. הוסיפו DBSet של מוצרים למחלקה DBContext

3. בצעו מיגרציה לדטה-בייס (עדכון של הדטה-בייס כי הוספתם מודל בOOP)
4. צרו Controller עם מתודה להצגת מוצרים בשם Index
5. צרו View להצגת המוצרים - עם קישור לעמוד הוספת מוצר חדש
6. הוסיפו מתודה בController בשם Create שתחזיר View
7. צרו את הView בתיקיית הViews בשם Create.cshtml - הView יכיל form להוספה של Product
8. ממשו את המתודה לשמירה של Product במסד הנתונים.
9. הוסיפו ולידציות.

(בדיוק מה שעשינו בכיתה - אז יכולים לעבוד עם המצגת בזמן הפתרון)