

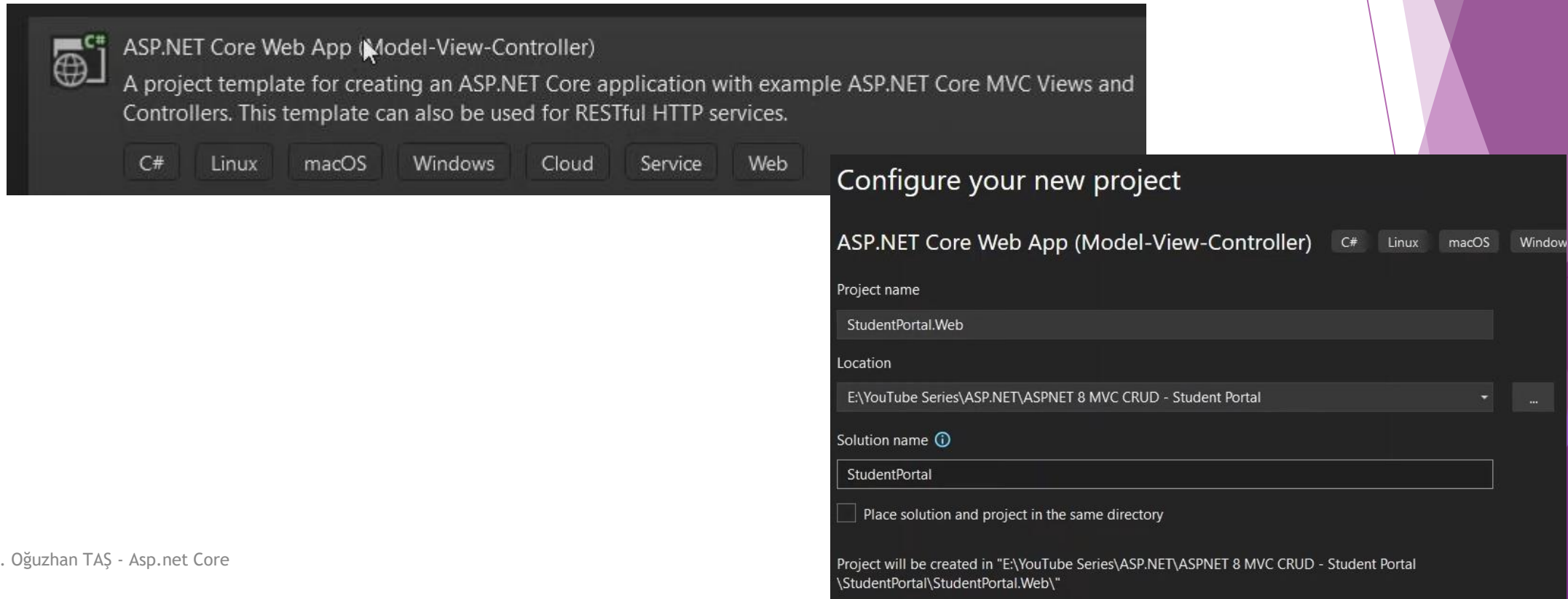
Asp.net Core ile veritabanı

Oğuzhan TAŞ

10.05.2024

Veritabanı İşlemleri

- Yeni bir ASP.net MVC(Model-View-Controller) projesi oluşturup **Project Name**(StudentPortal.Web) ve **Solution Name**(StudentPortal) olarak aşağıdaki gibi ekliyoruz.



ASP.NET Core Web App (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

C# Linux macOS Windows Cloud Service Web

Configure your new project

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows

Project name

StudentPortal.Web

Location

E:\YouTube Series\ASP.NET\ASPNET 8 MVC CRUD - Student Portal

Solution name ⓘ

StudentPortal

☐ Place solution and project in the same directory

Project will be created in "E:\YouTube Series\ASP.NET\ASPNET 8 MVC CRUD - Student Portal\StudentPortal\StudentPortal.Web\"

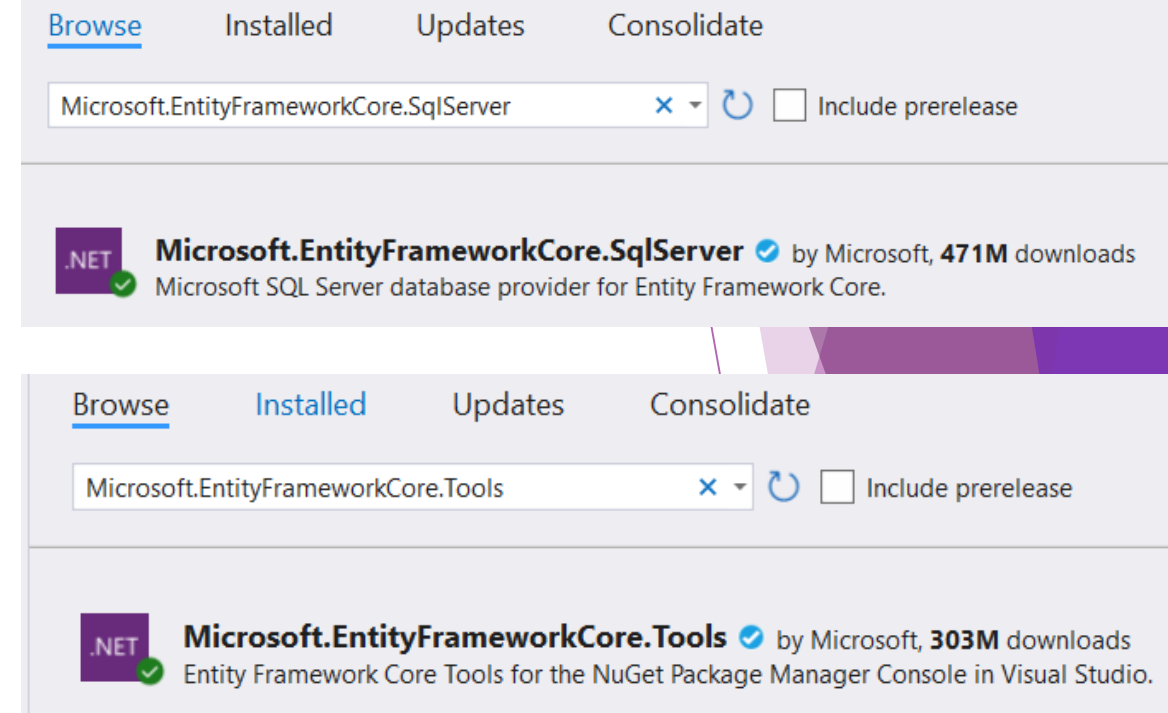
Nuget Paketlerini Yükleme

- Projenin **Dependencies** üzerine sağ tıklayıp **Manage Nuget Packages** seçip aşağıdaki SQL Server ile ilgili paketleri yükleyiniz. MySQL veya SQLite veritabanlarına bağlanacaksanız ilgili paketleri aratıp yükleyebilirsiniz.

[Microsoft.EntityFrameworkCore.SqlServer](#)

[Microsoft.EntityFrameworkCore.Tools](#)

- İkinci bir alternatif olarak nuget.org sitesine girip paketleri aratıp bulduktan sonra Nuget Paket Manager(NM) konsolundan yükleyebilirsiniz. **Visual Studio-Tools- Nuget Package-Manager- Packager Manager Console** ekranını açıp aşağıdaki gibi yapıştırabilirsiniz.




Nuget Paketlerini Yükleme (Önemli)

Önemli NOT

- ▶ EntityFramework Core ile EntityFramework farklı kütüphanelerdir ve ikisi kurulmuş ise yazılımda çakışmalar meydana gelmektedir.
- ▶ **Aşağıdaki kütüphaneyi kesinlikle kurmayınız, kurduysanız kaldırınız.**



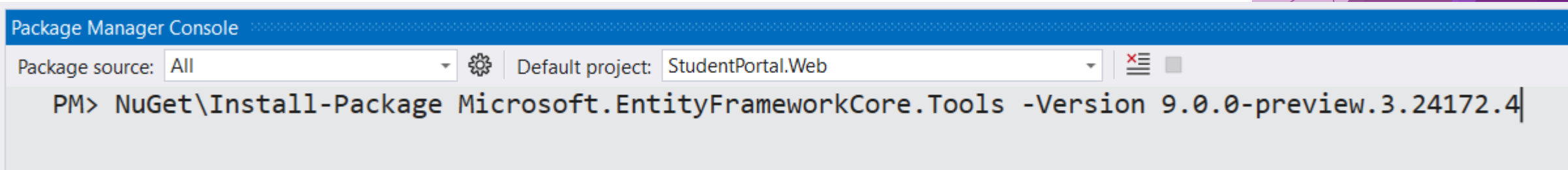
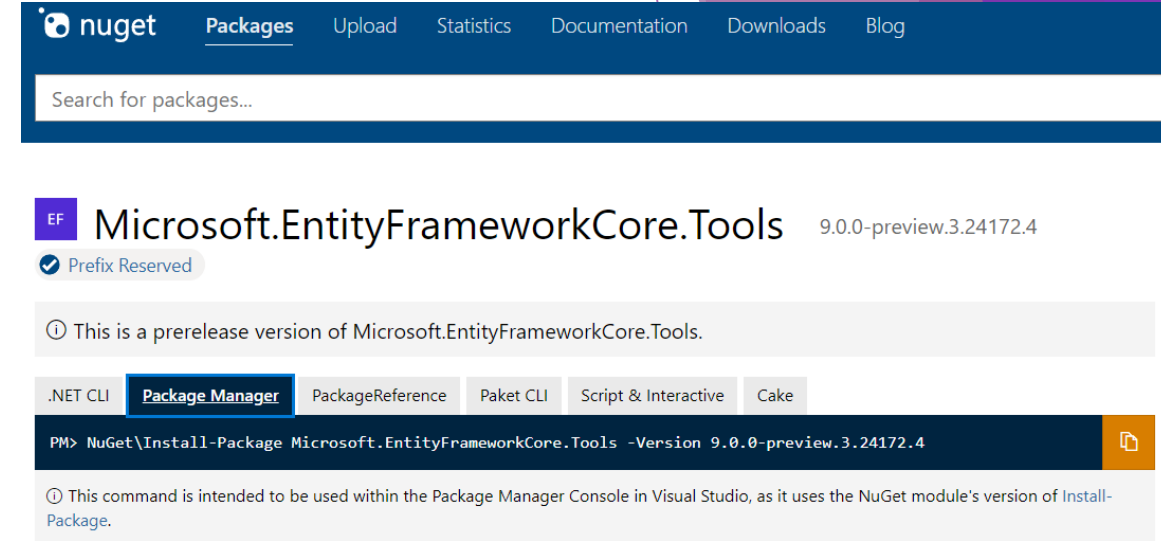
EntityFramework  by Microsoft, **267M** downloads
Entity Framework 6 (EF6) is a tried and tested object-relational
development and stabilization.

Nuget Paketlerini Yükleme

- İkinci bir alternatif olarak nuget.org sitesine girip paketleri aratıp bulduktan sonra Nuget Paket Manager(NM) konsolundan yükleyebilirsiniz. Visual Studio-Tools- Nuget Package-Manager- Packager Manager Console ekranını açıp aşağıdaki gibi yapıştırabilirsiniz.

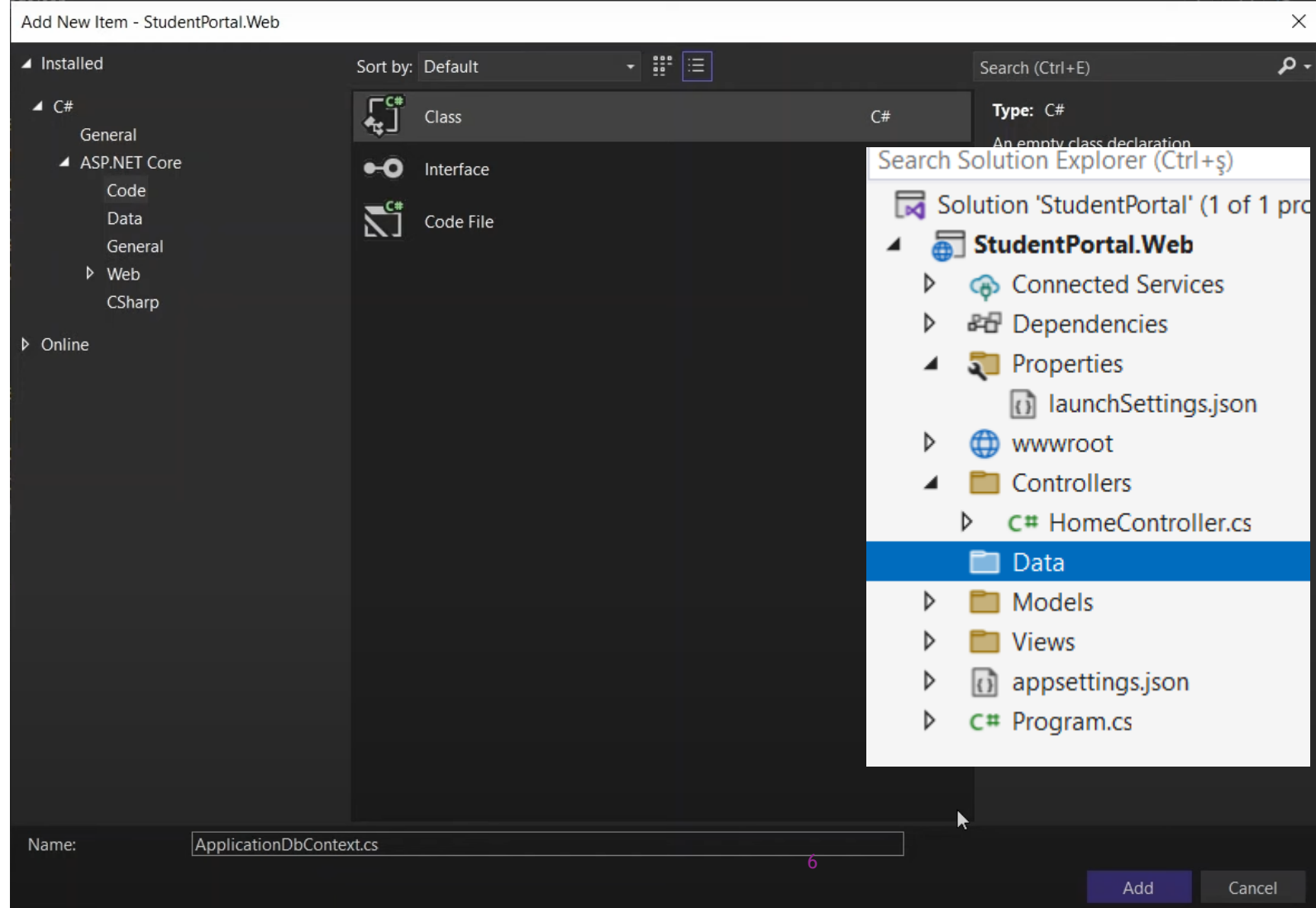
`Microsoft.EntityFrameworkCore.SqlServer`

`Microsoft.EntityFrameworkCore.Tools`



Veritabanı İşlemleri-DbContext

- Projeler içinde Model, View, Controller klasörleriyle aynı yolda **Data** klasörü oluşturup içine **ApplicationDbContext.cs** isimli class ekleyelim.



Veritabanı İşlemleri- ApplicationDbContext.cs

- ▶ ApplicationDbContext sınıfı oluşturduktan sonra, bu sınıfı DbContext sınıfından kalıtımla türeyecek şekilde ayarlayalım.
- ▶ Bu sınıf içinde Constructor(Yapıcı metod) oluşturmak için **ctor** yazıp TAB'a basalım ve aşağıdaki parametreleri ekleyelim.

```
ApplicationDbContext.cs*  launchSettings.json  HomeController.cs  Program.cs  StudentPortal.Web: Overview
StudentPortal.Web
  StudentPortal.Web.Data.ApplicationDbContext
    ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
1  using Microsoft.EntityFrameworkCore;
2
3  namespace StudentPortal.Web.Data
4
5      2 references
6      public class ApplicationDbContext: DbContext
7
8          0 references
9          public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options): base(options)
10
11
12
13
14
```

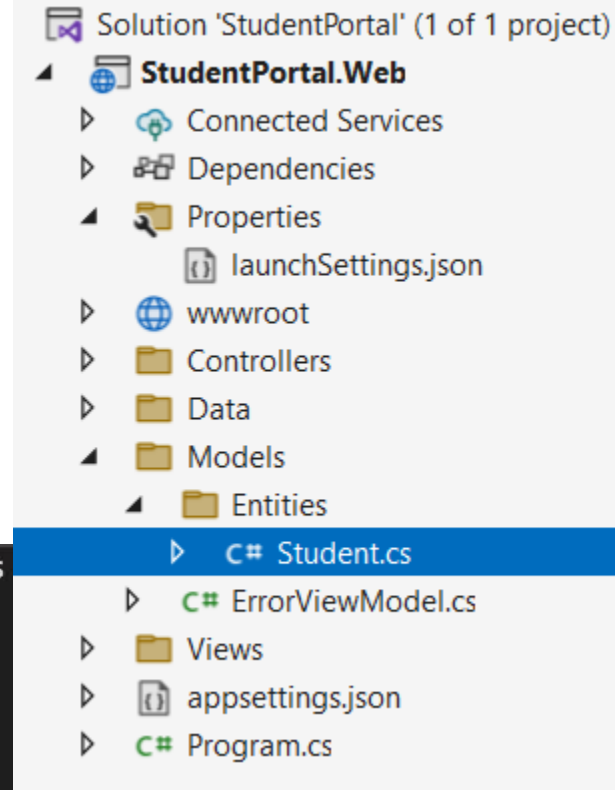
Dr. Oğuzhan TAŞ - Asp.net Core

Veritabanı İşlemleri

- Models klasöründe **Entities** klasörü oluşturalım
- Burda yeni bir class ekleyip ismine **Student.cs** verelim.

```
public class Student{  
    public Guid Id {get;set;}  
    public string Name{get;set;}  
    public string Email{get;set;}  
    public string Phone {get;set;}  
    public bool Subscribed {get;set;}  
}
```

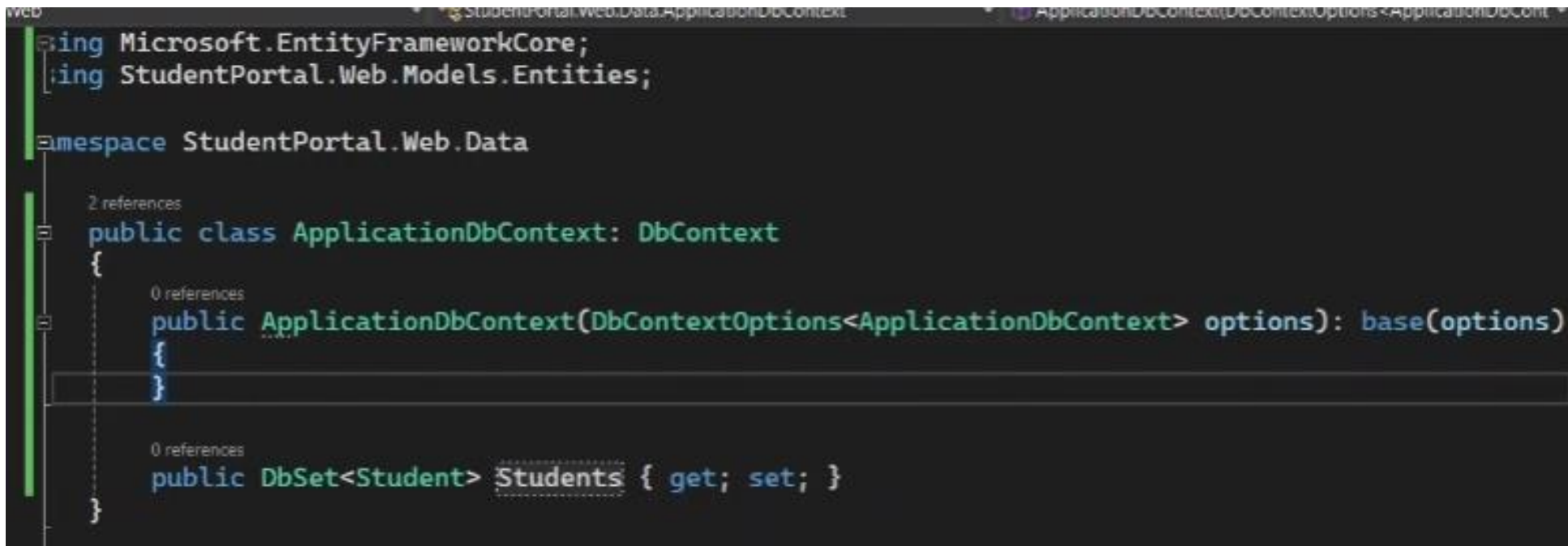
```
namespace StudentPortal.Web.Models.Entities  
{  
    0 references  
    public class Student  
    {  
        0 references  
        public Guid Id { get; set; }  
  
        0 references  
        public string Name { get; set; }  
        0 references  
        public string Email { get; set; }  
        0 references  
        public string Phone { get; set; }  
        0 references  
        public bool Subscribed { get; set; }  
    }  
}
```



Veritabanı İşlemleri

//Aşağıdaki satırı daha önce oluşturduğumuz
ApplicationDbContext.cs dosyasına ekliyoruz

public DbSet<Student> Students {get; set;}



```
using Microsoft.EntityFrameworkCore;
using StudentPortal.Web.Models.Entities;

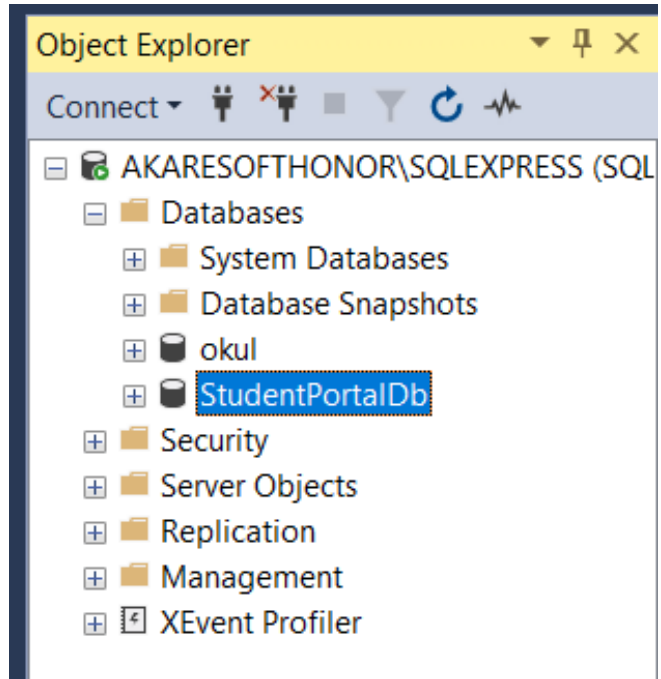
namespace StudentPortal.Web.Data

{
    2 references
    public class ApplicationDbContext: DbContext
    {
        0 references
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options): base(options)
        {
        }

        0 references
        public DbSet<Student> Students { get; set; }
    }
}
```

Veritabanı İşlemleri

Daha sonra SQL Server Management Studio'yu açarak aşağıdaki gibi **StudentPortalDb** veritabanını oluşturuyoruz.



Veritabanı İşlemleri

- ▶ **Program.cs** içine aşağıdaki satırı ekliyoruz.
- ▶ `builder.Services.AddDbContext<ApplicationDbContext>(options=>options.UseSqlServer());`

```
builder.Services.AddDbContext<ApplicationDbContext>(options => options.UseSqlServer());
```

- ▶ `appsettings.json` dosyasına Microsoft SQL Server veritabanı bağlantısı için aşağıdaki **ConnectionString** satırını ekliyoruz. Burada veritabanı adı(Database) olarak **StudentPortalDb** verilmiştir. Bu veritabanını SQL Server'da oluşturmuştuk. Server(Sunucu adı) için dikkat ederseniz . (nokta) yazıldı, bu yerel sunucu olduğu içindir, farklı bilgisayarda olsa IP adresi yazılırdı.

```
"AllowedHosts": "*",  
"ConnectionStrings": {  
  "StudentPortal": "Server=.;Database=StudentPortalDb;Trusted_Connection=True;TrustServerCertificate=True"  
}
```

Veritabanı İşlemleri

Sonra dosyamıza veritabanına nasıl bağlanacağını belirtmek için aşağıdaki gibi yeni eklediğimiz **ConnectionString** değerine göre değiştiriyoruz. Burada **StudentPortal** ismi verildi.

```
builder.Services.AddDbContext<ApplicationDbContext>(options =>  
options.UseSqlServer(builder.Configuration.GetConnectionString("StudentPortal")));
```

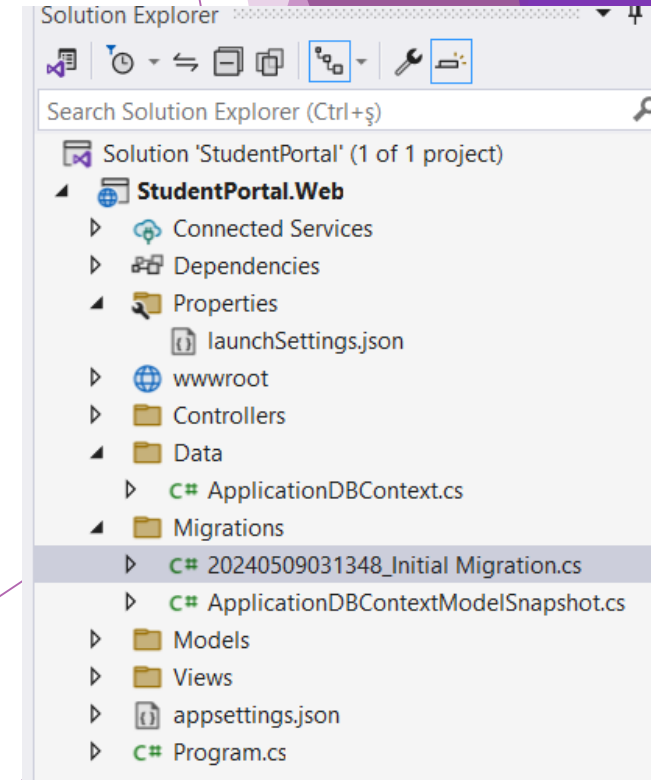
```
builder.Services.AddDbContext<ApplicationDbContext>(options =>  
options.UseSqlServer(builder.Configuration.GetConnectionString("StudentPortal")));
```

Veritabanı İşlemleri - Migration

Migration işlemleri ile yapılan ayarları veritabanına yansıtıyoruz. ApplicationDbContext.cs içinde yazdığımız Students tablosunun veritabanı içinde oluşmasını sağlayacağız.

- ▶ Visual Studio da menülerden **Tools- Nuget Package Manager - Package Manager Console** seçiyoruz. Ardından
- ▶ **Add-Migration «Initial Migration»** yazıyoruz.

```
PM> EntityFrameworkCore\Add-Migration "Initial Migration"  
Both Entity Framework Core and Entity Framework 6 are installed.  
Entity Framework 6.  
Build started...|  
Build succeeded.  
To undo this action, use Remove-Migration.
```



Veritabanı İşlemleri - Migration

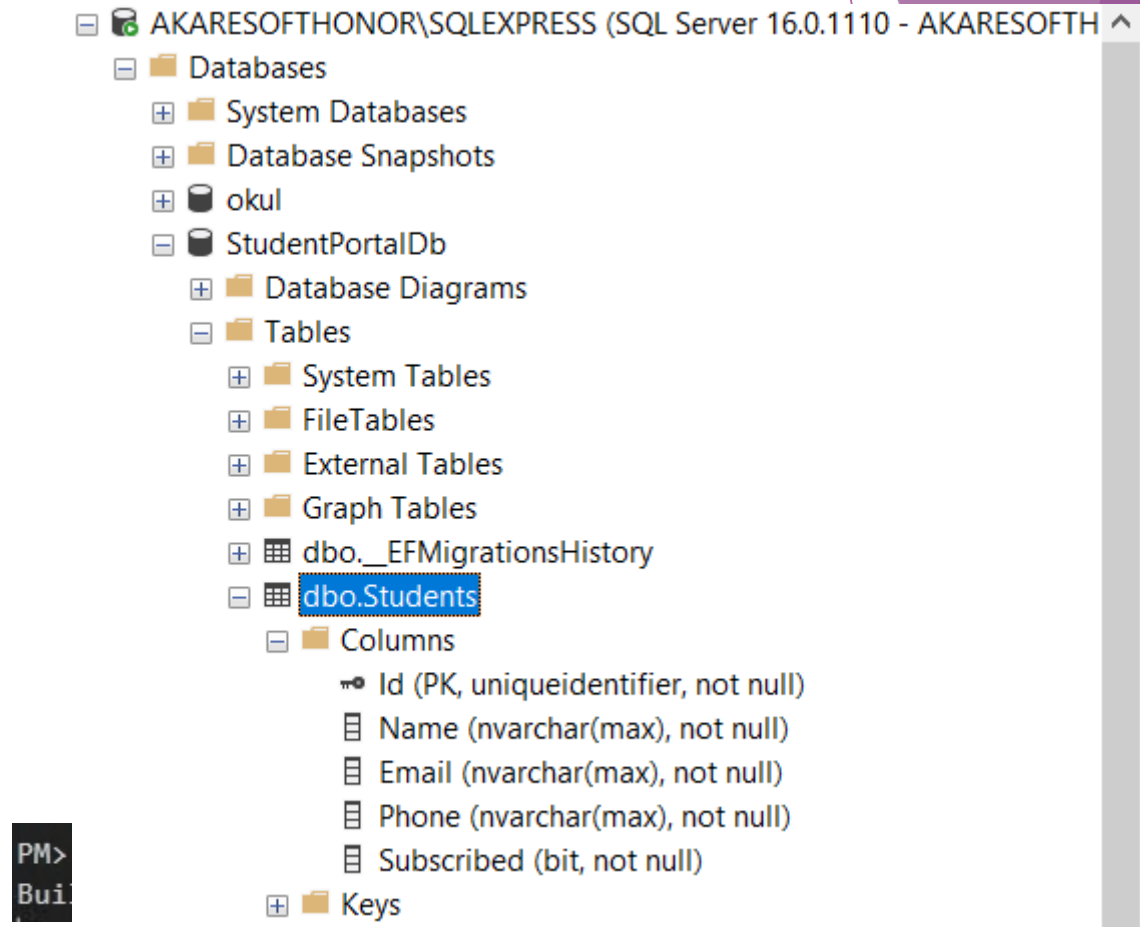
- Migrations klasörü oluşacak ve içinde class lar tanımlanacak, bunlar daha önce içeriğini belirlediğimiz Students tablosunun alanları.

```
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.CreateTable(
        name: "Students",
        columns: table => new
        {
            Id = table.Column<Guid>(type: "uniqueidentifier", nullable: false),
            Name = table.Column<string>(type: "nvarchar(max)", nullable: false),
            Email = table.Column<string>(type: "nvarchar(max)", nullable: false),
            Phone = table.Column<string>(type: "nvarchar(max)", nullable: false),
            Subscribed = table.Column<bool>(type: "bit", nullable: false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_Students", x => x.Id);
        });
}
```

Veritabanında güncelleme yaptıktan sonra yansıtmak için PM konsolunda **Update-Database** yazıyoruz.

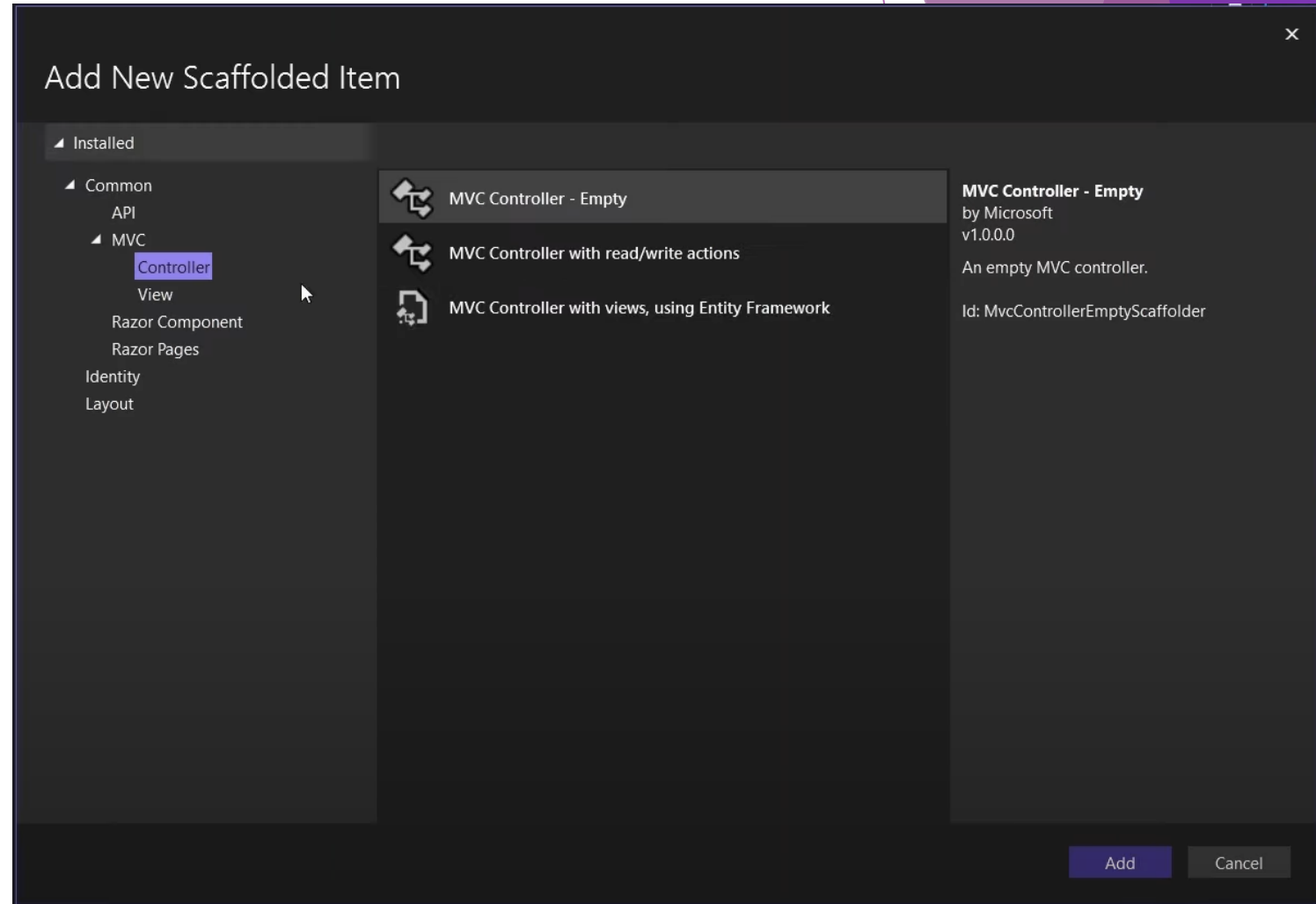
Veritabanı İşlemleri - Migration

- Yandaki gibi SQL Server içinde StudentPortalDb veritabanı içinde **Students** tablosu oluşmuş olacak.



Veritabanı İşlemleri - StudentController.cs

- ▶ Ardından yeni bir controller oluşturuyoruz.
- ▶ MVC Controller-empty **StudentController.cs** ismini veriyoruz.



Veritabanı İşlemleri - Add Metodu

- ▶ StudentController.cs içinde Varsayılan olarak IActionResult Index() gelir, onu değiştiriyoruz, **Add()** metodunu yandaki gibi ekliyoruz.
- ▶ return View(); satırındaki View üzerine sağ tıklayıp Add View seçeneğini seçiyoruz. Razor View empty seçiyoruz.
- ▶ [HttpGet] ifadesinin metod üzerine eklendiğine dikkat ediniz. GET action metodunu kullanacağız.

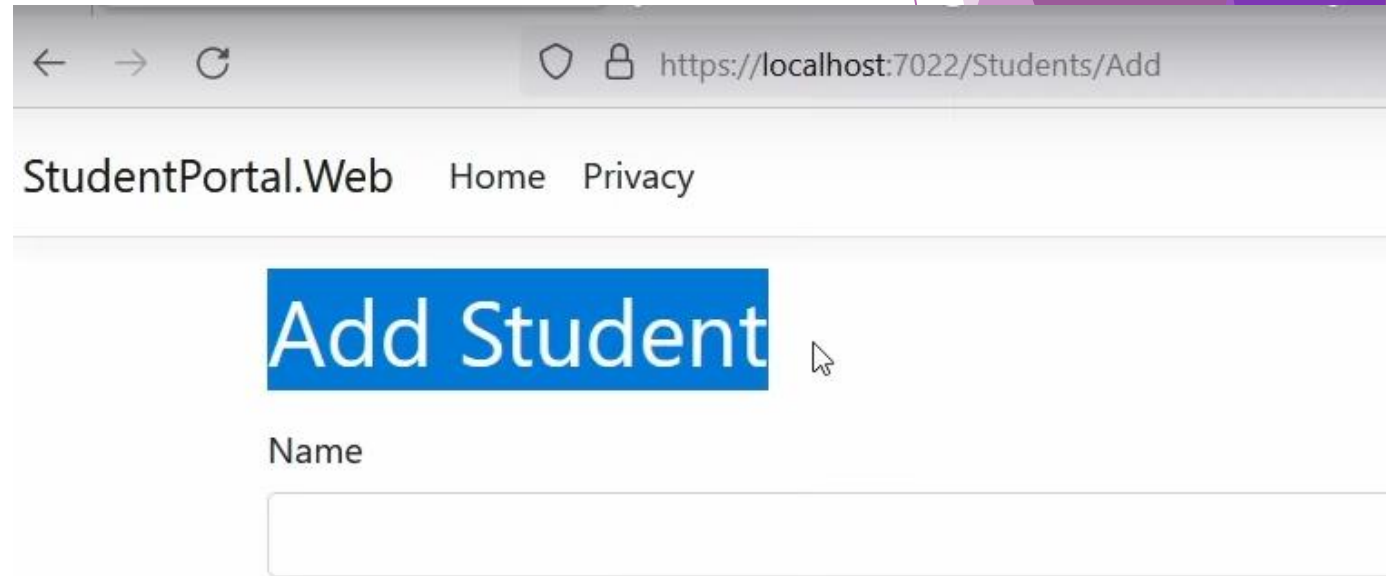
```
using Microsoft.AspNetCore.Mvc;

namespace StudentPortal.Web.Controllers
{
    0 references
    public class StudentsController : Controller
    {
        [HttpGet]
        0 references
        public IActionResult Add()
        {
            return View();
        }
    }
}
```

Add.cshtml View oluşturma

Tarayıcı adres çubuğuna **/Students/Add** yazdığımızda soldaki View dosyası sağdaki şekilde görünecektir.

```
Add.cshtml  StudentsController.cs
1  @*
2  For more information on enabling MVC for em
3  *@
4  @{
5  }
6
7
8  <h1>Add Student</h1>
9
10 <div class="mt-3">
11     <label class="form-label">Name</label>
12     <input type="text" class="form-control" />
13 </div>
```



Add.cshtml View oluşturma

View dosyasının tamamı yandaki gibidir Name, Email, Phone, Subscribed alanları ve Save butonu eklenmiştir.

```
<h1>Add Student</h1>

<div class="mt-3">
  <label class="form-label">Name</label>
  <input type="text" class="form-control" />
</div>

<div class="mt-3">
  <label class="form-label">Email</label>
  <input type="email" class="form-control" />
</div>

<div class="mt-3">
  <label class="form-label">Phone</label>
  <input type="text" class="form-control" />
</div>

<div class="mt-3">
  <input type="checkbox" class="form-check-input" id="Subscribed" />
  <label class="form-check-label" for="Subscribed">Subscribed</label>
</div>

<div class="mt-3">
  <button type="submit" class="btn btn-primary">Save</button>
</div>
```

Add.cshtml View oluşturma

View dosyasının tamamı yandaki gibidir Name, Email, Phone, Subscribed alanları ve Save butonu eklenmiştir. Form tag'leri eklenmiştir.

```
@model StudentPortal.Web.Models.AddStudentViewModel;
<form method="post">
<h1>Add New </h1>

<div class="mt-3">
    <label class="form-label">Name</label>
    <input type="text" class="form-control" asp-for="Name"/>
</div>

<div class="mt-3">
    <label class="form-label">E-mail</label>
    <input type="email" class="form-control" asp-for="Email" />
</div>

<div class="mt-3">
    <label class="form-label">Phone</label>
    <input type="text" class="form-control" asp-for="Phone" />
</div>

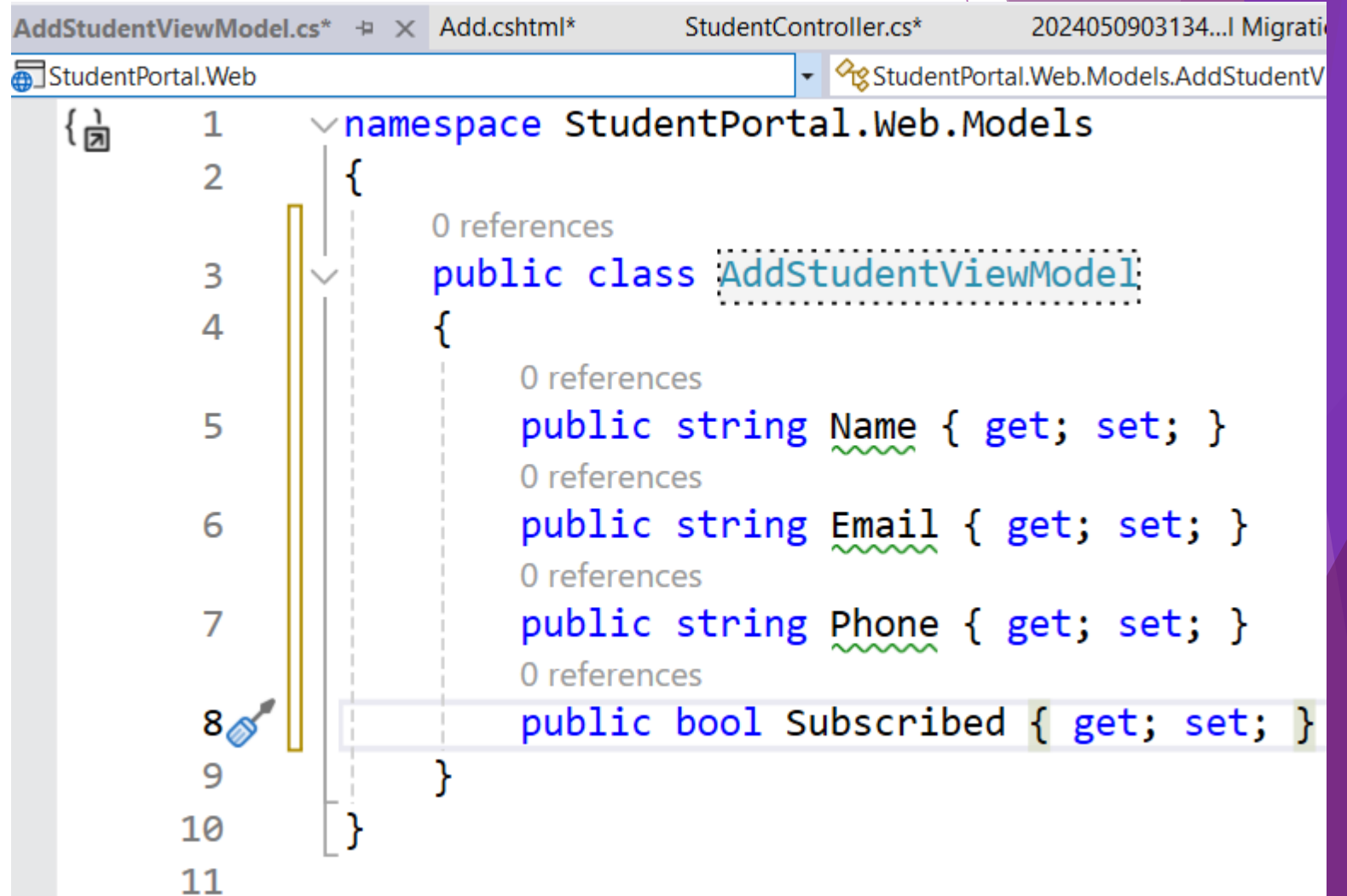
<div class="mt-3">
    <input type="checkbox" class="form-check-input" id="Subscribed" asp-for="Subscribed" />
    <label class="form-check-label" for="Subscribed">Subscribed</label>
</div>

<div class="mt-3">
    <button type="submit" class="btn btn-primary">Save</button>
</div>

</form>
```

AddStudentViewModel Oluşturma

Model klasörü üzerine sağ tıklayıp **Add Class** seçeneğini seçip, **AddStudentViewModel** ismini veriyoruz. Yandaki gibi içeri Name, Email, Phone, Subscribed alanları ile dolduruyoruz.



```
1 namespace StudentPortal.Web.Models
2 {
3     0 references
4     public class AddStudentViewModel:
5     {
6         0 references
7         public string Name { get; set; }
8         0 references
9         public string Email { get; set; }
10        0 references
11        public string Phone { get; set; }
12        0 references
13        public bool Subscribed { get; set; }
14    }
15 }
```

Add.cshtml View

Yandaki model bağlantı satırını ekliyoruz ve input alanlarını **asp-for** şekline çeviriyoruz.

Aşağıdaki satırı en üste ekleyerek formun model ile bağlantı kurmasını sağlıyoruz.

```
<h1>Add Student</h1>

<form method="post">
  <div class="mt-3">
    <label class="form-label">Name</label>
    <input type="text" class="form-control" asp-for="Name" />
  </div>

  <div class="mt-3">
    <label class="form-label">Email</label>
    <input type="email" class="form-control" asp-for="Email" />
  </div>

  <div class="mt-3">
    <label class="form-label">Phone</label>
    <input type="text" class="form-control" asp-for="Phone" />
  </div>

  <div class="mt-3">
    <input type="checkbox" class="form-check-input" id="Subscribed" asp-for="Subscribed" />
    <label class="form-check-label" for="Subscribed">Subscribed</label>
  </div>

  <div class="mt-3">
    <button type="submit" class="btn btn-primary">Save</button>
  </div>
</form>
```

```
@model StudentPortal.Web.Models.AddStudentViewModel;
```

```
@model StudentPortal.Web.Models.AddStudentViewModel

<h1>Add Student</h1>

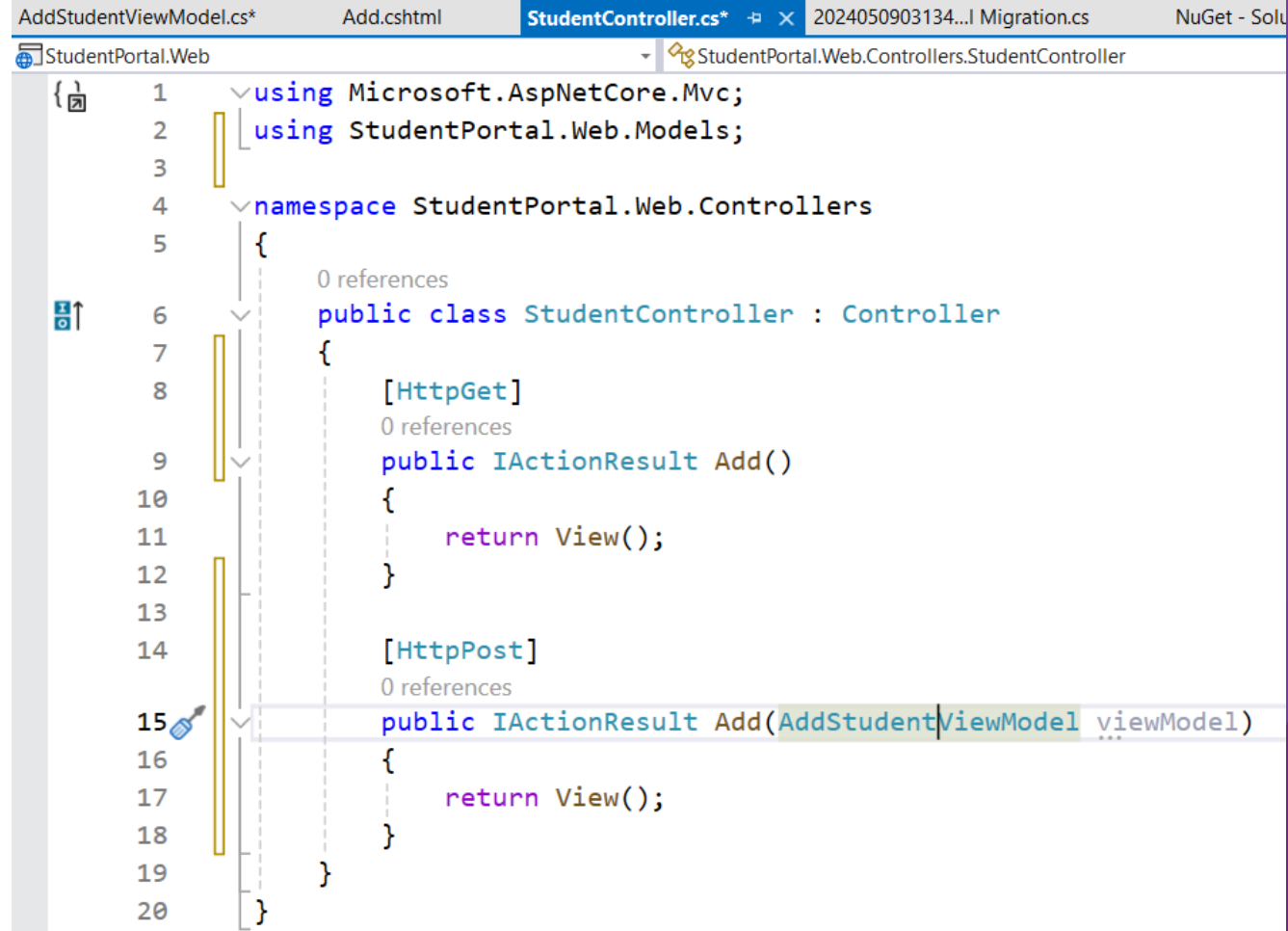
<div class="mt-3">
  <label class="form-label">Name</label>
  <input type="text" class="form-control" />
</div>
```


Add.cshtml View

StudentController.cs içini yandaki şekilde düzenliyoruz.

Üst kısma `using StudentPortal.Web.Models;` satırı ile kontrolün hangi modelle bağlantı kuracağını belirtiyoruz. Form post için kullanacağımız Add metodunu aşağıdaki gibi değiştiriyoruz.

```
[HttpPost]
public IActionResult
Add(AddStudentViewModel viewModel)
{
    return View();
}
```



```
1  using Microsoft.AspNetCore.Mvc;
2  using StudentPortal.Web.Models;
3
4  namespace StudentPortal.Web.Controllers
5  {
6      public class StudentController : Controller
7      {
8          [HttpGet]
9          public IActionResult Add()
10         {
11             return View();
12         }
13
14         [HttpPost]
15         public IActionResult Add(AddStudentViewModel viewModel)
16         {
17             return View();
18         }
19     }
20 }
```

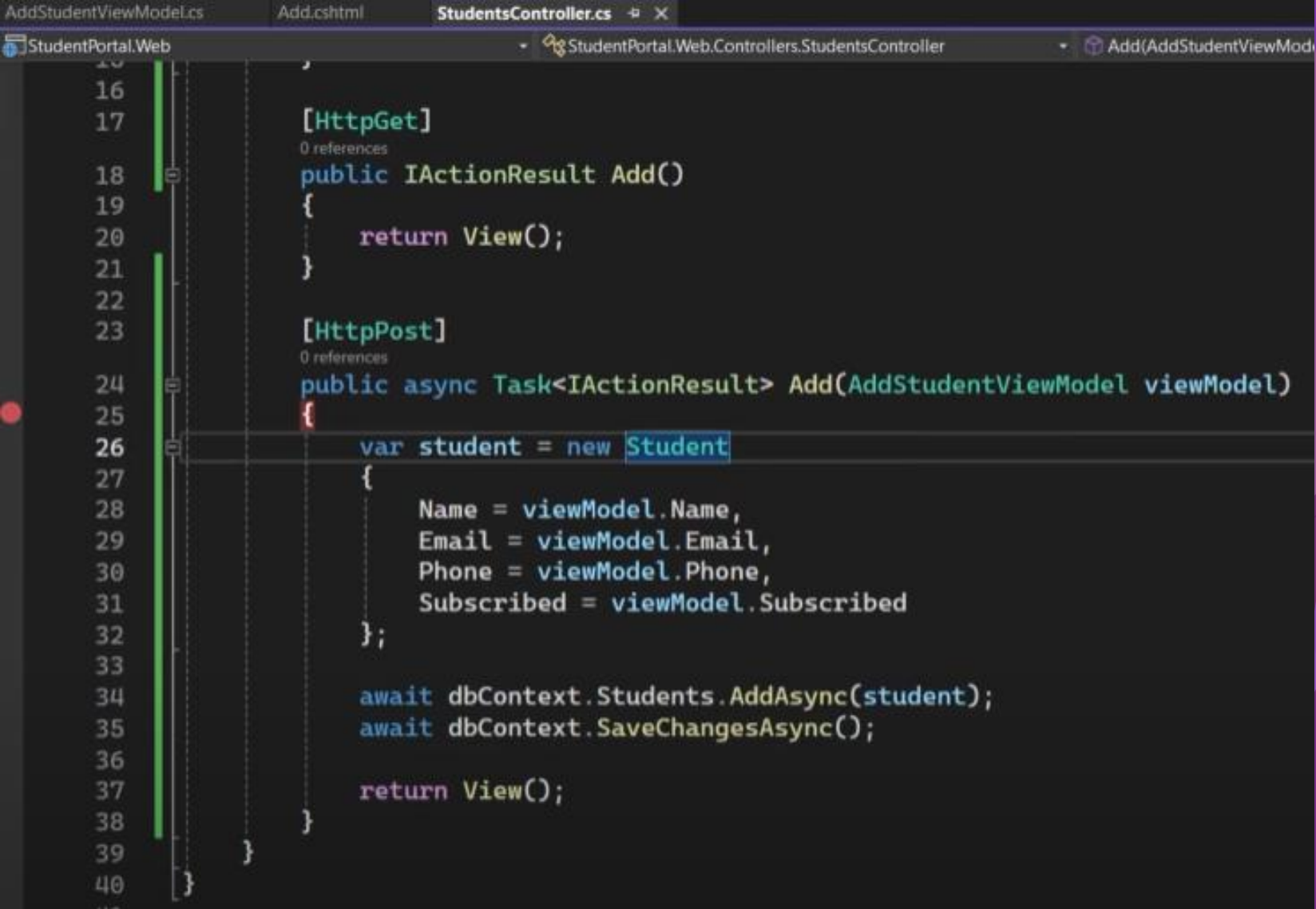
StudentController içeriği düzenleme

Yandaki gibi
dbContext isimli
sadece okunabilir bir
nesne oluşturuyoruz
ve yandaki gibi
StudentController(A
pplicationDbContext
dbContext)
metodunu
oluşturuyoruz

```
1 reference
public class StudentController : Controller
{
    private readonly ApplicationDbContext dbContext;
    0 references
    public StudentController(ApplicationDbContext dbContext)
    {
        this.dbContext = dbContext;
    }
}
```


StudentController içeriği düzenleme

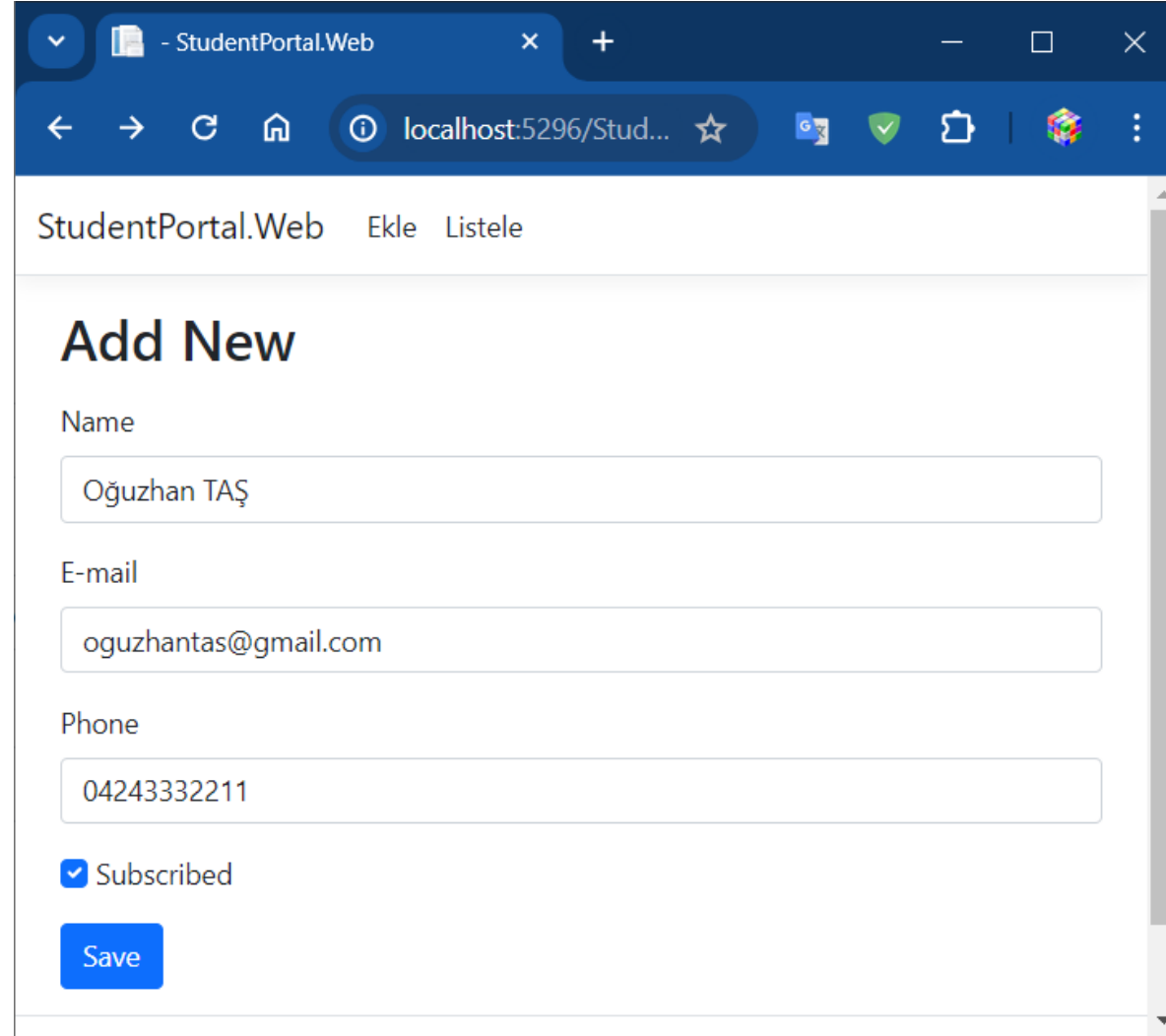
Yandaki gibi
Add() metodunu
değiştiriyoruz



```
16  
17 [HttpGet]  
18 0 references  
19 public IActionResult Add()  
20 {  
21     return View();  
22 }  
23  
24 [HttpPost]  
25 0 references  
26 public async Task<IActionResult> Add(AddStudentViewModel viewModel)  
27 {  
28     var student = new Student  
29     {  
30         Name = viewModel.Name,  
31         Email = viewModel.Email,  
32         Phone = viewModel.Phone,  
33         Subscribed = viewModel.Subscribed  
34     };  
35     await dbContext.Students.AddAsync(student);  
36     await dbContext.SaveChangesAsync();  
37     return View();  
38 }  
39 }  
40 }
```

Form Ekranı

Kayıt ekleme
ekranı



The screenshot shows a web browser window with the title 'StudentPortal.Web'. The address bar displays 'localhost:5296/Stud...'. The page content includes a header with 'StudentPortal.Web', 'Ekle', and 'Listele'. The main section is titled 'Add New' and contains three text input fields: 'Name' with the value 'Oğuzhan TAŞ', 'E-mail' with the value 'oguzhantas@gmail.com', and 'Phone' with the value '04243332211'. Below these fields is a checkbox labeled 'Subscribed' which is checked. At the bottom of the form is a blue 'Save' button.

Kayıt Listeleme

Listeleme için **StudentController** kontrolüne(denetçisine) aşağıdaki veritabanından bilgileri listeyen kodları ekleyiniz. </student/list> şeklinde ulaşacağız, ilgili view dosyasını da sonraki ekranda tanımlayalım.

```
[HttpGet]
```

```
0 references
```

```
public async Task<IActionResult> List()
```

```
{
```

```
    var students = await dbContext.Students.ToListAsync();
```

```
    return View(students);
```

```
}
```

Kayıt Listeleme

View/List.cshtml dosyası içeriği

```
@model List<StudentPortal.Web.Models.Entities.Student>
<h3>Student List</h3>
```

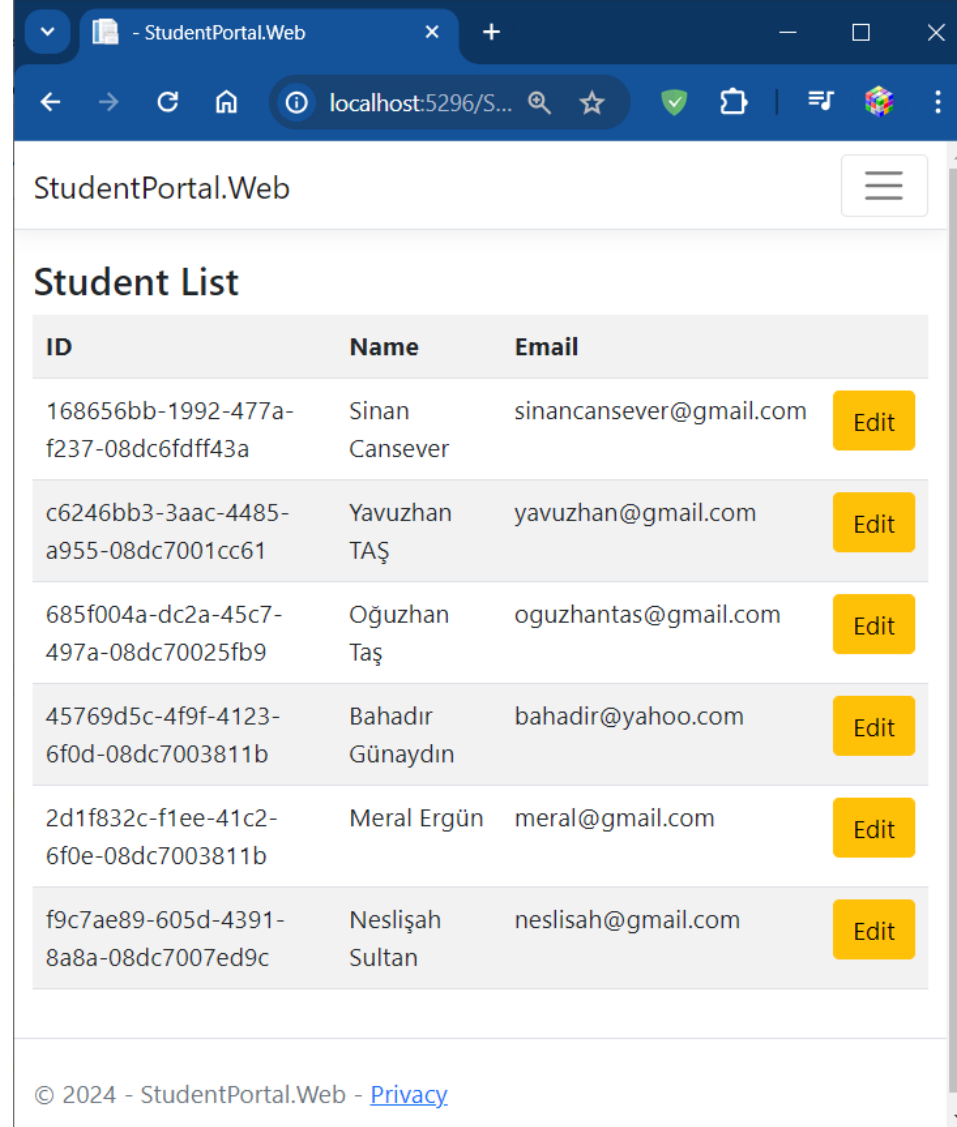
```
<table class="table table-striped">
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>Email</th>
    <th></th>
  </tr>
```

```
  @foreach (var student in Model){
    <tr>
      <td>@student.Id</td>
      <td>@student.Name</td>
      <td>@student.Email</td>
      <td><a class="btn btn-warning" asp-controller="student" asp-action="Edit"
asp-route-id="@student.Id">Edit</a></td>
    </tr>
  }
</table>
```

Kayıt Listeleme View/List.cshtml dosyası

[student/list](#)

Bağlantısından yandaki sayfaya ulaşabilirsiniz. Burada edit butonuna basınca kayıt düzenleme yapacağız. Get metodu ile kullanıcının hangi Id bilgisinden tıkladığını görüp, Post metodu ile kaydedeceğiz. Sonraki sayfada bu işlemi göreceğiz.



ID	Name	Email	Edit
168656bb-1992-477a-f237-08dc6fdff43a	Sinan Cansever	sinancansever@gmail.com	Edit
c6246bb3-3aac-4485-a955-08dc7001cc61	Yavuzhan TAŞ	yavuzhan@gmail.com	Edit
685f004a-dc2a-45c7-497a-08dc70025fb9	Oğuzhan Taş	oguzhantas@gmail.com	Edit
45769d5c-4f9f-4123-6f0d-08dc7003811b	Bahadır Günaydın	bahadir@yahoo.com	Edit
2d1f832c-f1ee-41c2-6f0e-08dc7003811b	Meral Ergün	meral@gmail.com	Edit
f9c7ae89-605d-4391-8a8a-08dc7007ed9c	Neslişah Sultan	neslisah@gmail.com	Edit

© 2024 - StudentPortal.Web - [Privacy](#)

Kayıt Düzenleme

Düzenleme için

StudentController

kontrolüne(denetçisine)
aşağıdaki kodları ekleyiniz.

[/student/edit](#) şeklinde
ulaşacağız, ilgili view
dosyasını da sonraki
ekranda tanımlayalım.

[return.RedirectToAction\(\)](#)
metodu ile istediğimiz
kontrolün metoduna
yönlendirme yapabiliriz.

```
[HttpGet]
```

```
0 references
```

```
public async Task<IActionResult> Edit(Guid id)
{
    var student = await dbContext.Students.FindAsync(id);
    return View(student);
}
```

```
[HttpPost]
```

```
0 references
```

```
public async Task<IActionResult> Edit(Student viewModel)
{
    var student = await dbContext.Students.FindAsync(viewModel.Id);

    if(student is not null)
    {
        student.Name = viewModel.Name;
        student.Phone= viewModel.Phone;
        student.Email = viewModel.Email;
        student.Subscribed= viewModel.Subscribed;

        await dbContext.SaveChangesAsync();
    }

    return RedirectToAction("List","Student");
}
```

Kayıt Düzenleme

```
@model StudentPortal.Web.Models.Entities.Student;
```

```
<h3>Edit Student</h3>
```

```
@if (Model is null)
```

```
{
```

```
    <h3>No student has this ID</h3>
```

```
}
```

```
else
```

```
{
```

```
    <form method="post">
```

```
        <div class="mt-3">
```

```
            <label class="form-label">ID</label>
```

```
            <input type="text" class="form-control" asp-for="id" readonly />
```

```
        </div>
```

Kayıt Düzenleme -Devam

```
<div class="mt-3">
    <label class="form-label">Name</label>
    <input type="text" class="form-control" asp-for="Name" />
</div>

<div class="mt-3">
    <label class="form-label">Email</label>
    <input type="email" class="form-control" asp-for="Email" />
</div>
<div class="mt-3">
    <label class="form-label">Telephone</label>
    <input type="number" class="form-control" asp-for="Phone" />
</div>
<div class="mt-3">
    <input type="checkbox" class="form-check-input" id="Subscribed" asp-for="Subscribed" />
    <label class="form-check-label">Subscribed</label>
</div>

<div class="mt-3">
    <button type="submit" class="btn btn-primary">Save</button>

    <button type="submit" class="btn btn-danger"
    asp-action="Delete" asp-controller="Student"
    >Delete</button>

</div>
</form>
```


Kayıt Sil

StudentController içinde yandaki Delete metodu ile silme işlemini yapıp, silme sonrası listeleme sayfasına yönlendiriyoruz.

```
[HttpPost]
0 references
public async Task<IActionResult> Delete(Student viewModel)
{
    var student = await dbContext.Students
        .AsNoTracking()

        .FirstOrDefaultAsync(x => x.Id == viewModel.Id);

    if (student is not null)
    {
        dbContext.Students.Remove(viewModel);
        await dbContext.SaveChangesAsync();
    }

    return RedirectToAction("List", "Student");
}
```

Kayıt Silme

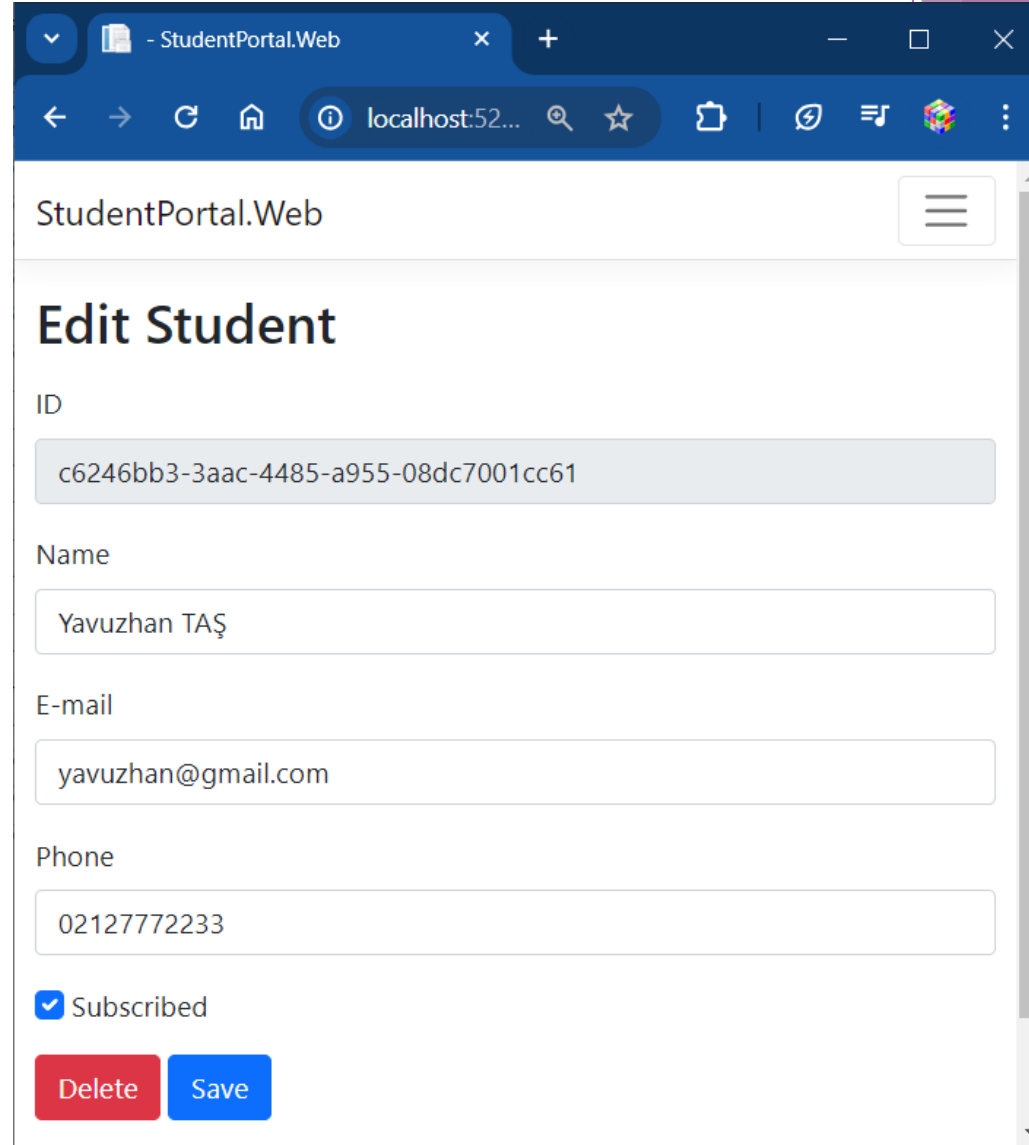
View/Edit.cshtml dosyası içeriği

Edit.cshtml dosyasına yandaki Sil butonunu ekliyoruz, bu butona tıklayınca StudentController içindeki Delete metoduna yönlendiriyor.

```
<div class="mt-3">  
  <button type="submit" class="btn btn-danger"  
    asp-action="Delete"  
    asp-controller="Student"  
  >Delete</button>  
  <button type="submit" class="btn btn-  
    primary">Save</button>  
</div>
```

Kayıt Sil

Listeden ilgili Edit butonuna basınca gelen sayfada DELETE butonunu aşağıdaki gibi görüyoruz, tıklayınca silme işlemi gerçekleşiyor



The screenshot shows a web browser window with the address bar displaying 'localhost:52...'. The page title is 'StudentPortal.Web'. The main content area is titled 'Edit Student' and contains the following fields:

- ID: c6246bb3-3aac-4485-a955-08dc7001cc61
- Name: Yavuzhan TAŞ
- E-mail: yavuzhan@gmail.com
- Phone: 02127772233
- Subscribed: ☒

At the bottom of the form, there are two buttons: 'Delete' (red) and 'Save' (blue).

Kaynaklar

- Microsoft ASP.net Core Genel Bakış

<https://learn.microsoft.com/tr-tr/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0>

- Sadık Turan Asp.netCore

<https://www.youtube.com/watch?v=YAgvVqh8pPA>

- ASP.NET Core MVC CRUD Operations using .NET 8 and Entity Framework Core - MVC For Beginners

https://www.youtube.com/watch?v=_uSw8sh7xKs