ASP.NET CORE 8.0 MVC DERS NOTLARI-(9. HAFTA)

İçindekiler

ASP.NET CORE 8.0 MVC DERS NOTLARI-(6. HAFTA) 1
GÖRÜNTÜLEME (VIEW) SAYFALARINA VERİLERİN TAŞINMASI ve SİLİNMESİ 1
A-BİLGİLERİ LİSTELEME 1
a) Tek Tablodan Bilgileri Getirme 1
b) Bağlantılı Olarak Tablolardan Bilgileri Getirme (Navigation Property) 6
B-YENİ KAYIT 7
a) Fakülte Bilgileri Sayfa İçinde Sabit Olarak Yeni Kayıt Ekleme 7
b) Fakülte Bilgileri Veritabanından Getirilerek Yeni Kayıt Ekleme 10
C-KAYIT GÜNCELLEME 11
D-KAYIT SİLME 14
a) Direk Kaydı Silme 14
b) Onay Alarak Kaydı Silme 15
c) Birbirine Bağlantılı Tablolardan Kayıt Silme 16

GÖRÜNTÜLEME (VIEW) SAYFALARINA VERİLERİN TAŞINMASI ve SİLİNMESİ

Genel bir Veritabanı işleminde 4 temel uygulama vardır. Bunlar;

a) Bilgileri Listeleme
b) Yeni Kayıt Ekleme
c) Kayıt Güncelleme
d) Kayıt Silme

Şimdi bu işlemlerin hepsini basit düzeyde Bolumler tablosu üzerinde deneyelim.

A-BİLGİLERİ LİSTELEME

a) Tek Tablodan Bilgileri Getirme

Veritabanımızı oluşturduğumuza göre, bir tane tabloya verileri elle girelim ve bu verileri oluşturacağımız bir View sayfasında görüntüleyelim.

Bunun için önce Bolumler tablosuna 3 tane deneme verisi girelim. Bu verileri Bölümlerin listesinin gösterildiği sayfada yayınlayalım.

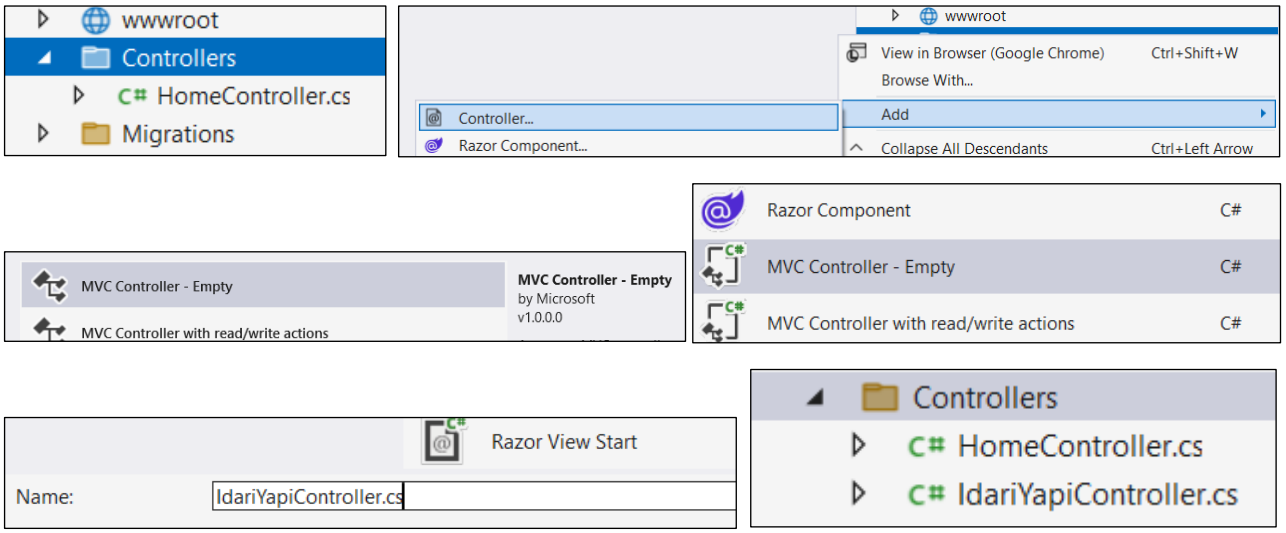
icayiroglu\SQLEXP...dek - dbo.Bolumler

	BolumID	FakulteID	BolumAdi
	1	1	Mekatronik Bölümü
	2	1	Bilgisayar Bölümü
	3	1	Makine Bölümü
▶*	NULL	NULL	NULL

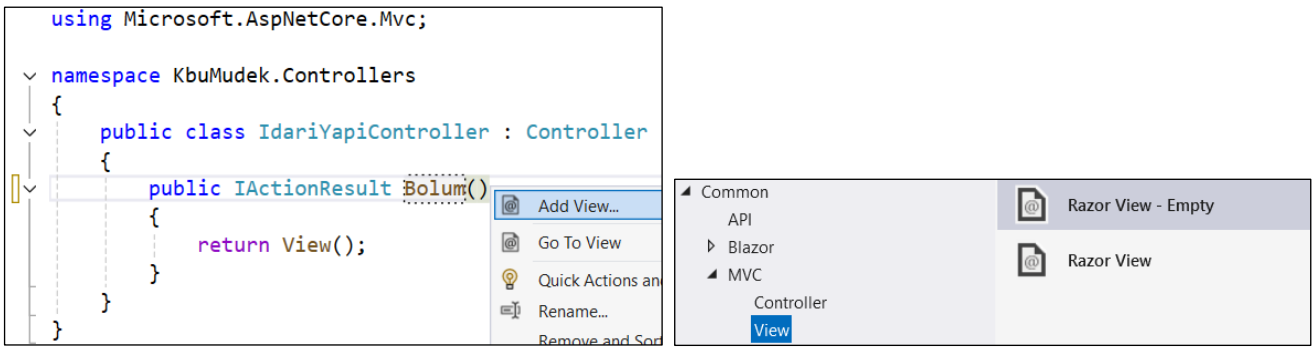
Öncelikle View sayfalarına veri gönderecek ve geri dönüşleri alacak olan Controller sınıfımızı oluşturmalıyız. Controller sınıfları kendisine bağlanan View sayfalarına veri girişi çıkışını yönetir. Bunlar C# kodlarının olduğu kod sayfalardır. Bir Controller sınıfı ile çok sayıda View sayfası yönetilebilir. Daha çok benzer görevleri sahip sayfaları aynı Controller altında toplamak gerekir. Aynı Controller'a bağlı View sayfaları yine o controller'a ait View klasörünün içinde olmalıdır.

Fakülteler, Bölümler, Programlar ile ilgili kayıt ve güncelleme işlemlerinin hepsini IdariContoller altında toplayalım.

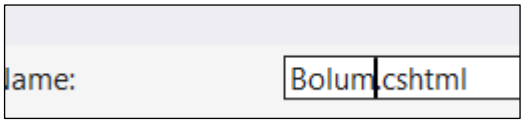
Sitemizdeki Controller klasörünün üzerine sağ tuşa tıklayalım. Buradan Controller'ımızı ekleyelim. Boş bir controller olsun. Not oluşturulan her Controller sınıfının sonuna mutlaka Controller adı eklenmelidir. Yani tek başına IdariYapi.cs olmaz. IdariYapiController.cs olmak zorundadır. MVC (model,view,controller) yapı mantığında isim ve klasörler belli bir hiyerarşi ile oluşturulur.



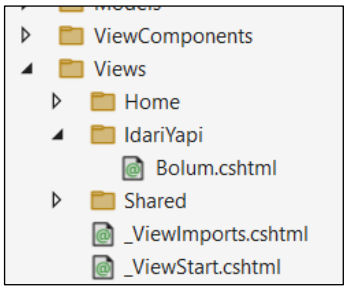
Idari yapının içerisinde sınıf ve metod yapımız oluştu. View sayfalarını yönetecek metodlar IActionResult tipindedir. Burada Index olarak varsayılan metodun adını Bolum olarak değiştirelim. Üzerine sağ tuşa tıklayıp Boş bir View sayfası ekleyelim.



Açılan pencerede bizden oluşturulacak View sayfasının adını isteyecektir. Bu adı da Bolum.cshtml olarak oluşturalım.



Dikkat edersek Controller'ımızın ismi ile aynı Views klasörü içerisinde IdariYapi kalasörü oluştu. Dolayısı ile bu kontrollera ait tüm View sayfalarımızı bu klasörün içinde toplayacağız. Böylece sayfalar Controller seviyesinde bir arada gruplandırılmış olacak.



Bolum.cshtml sayfasını açalım. Boş bir sayfa olduğunu göreceğiz. Bu sayfada bilgileri doldurmak için içerisini dolduracağız fakat öncelikle Veritabanından bilgileri alıp bu sayfaya gönderecek Controller sınıfımızın içerisindeki bilgileri hazırlayalım.

Controller sınıfımızın en üstüne önce Model yapılarımızı tanıması için onun referans satırını ekleyelim.

```
using KbuMudek.Models;
```

Sonra Bolum metodumuza girmeden önce üst satırda Veritabanında bilgileri getirecek olan ve Context sınıfından bir nesne türetelim. Bunun adına kısaca c diyelim. Yani oluşturduğumuz c sınıfı Context sınıfının özelliklerine sahip olacak ve veri taşıma kabiliyeti de olmuş olacak.

```
Context c = new Context();
```

Sonra Bolum metodumuzun içerisine Bolumler tablosundan tüm verileri ToList() fonksiyonu ile getirecek ve bunu "bolumler" adı ile belirttiğimiz bir taşıyıcı arabaya yükleyeceğiz. Dikkat edersek veritabanından taşınan bilgilerin yapısı ile onu sayfaya taşıyacak arabanın yapısı aynı olmak zorundadır. Yoksa bir kaptan alınıp başka bir kaba konulan bilgiler farklı kaplarla taşınamaz. Bu nedenle bilgileri taşıyacak olan "bolumler" arabasının yapısı da "Bolum" model yapısı ile tanımladığımız yapıda olmak zorundadır. Bu nedenle başına Bolum yazmalıyız. Yani aşağıdaki gibi yazmalıyız. Fakat bu satır hata verir. Neden? Çünkü veritabanından tek bir bölüm gelmiyor. Bir sürü bölüm geliyor. Dolayısı ile sol taraftaki Bolum yapısına sahip bolumler arabası çok sayıda bölümü alamayacaktır. Onun yerine eşittirin sağ tarafında nasıl bir yapı varsa sol tarafı da aynı yapı ile tanımlıyorum manasına gelen "var" kelimesi kullanılır. Böylece bolumler arabasının hangi tipte olacağını çok fazla düşünmemiş oluruz.

```
Bolum bolumler = c.Bolumler.ToList();
```

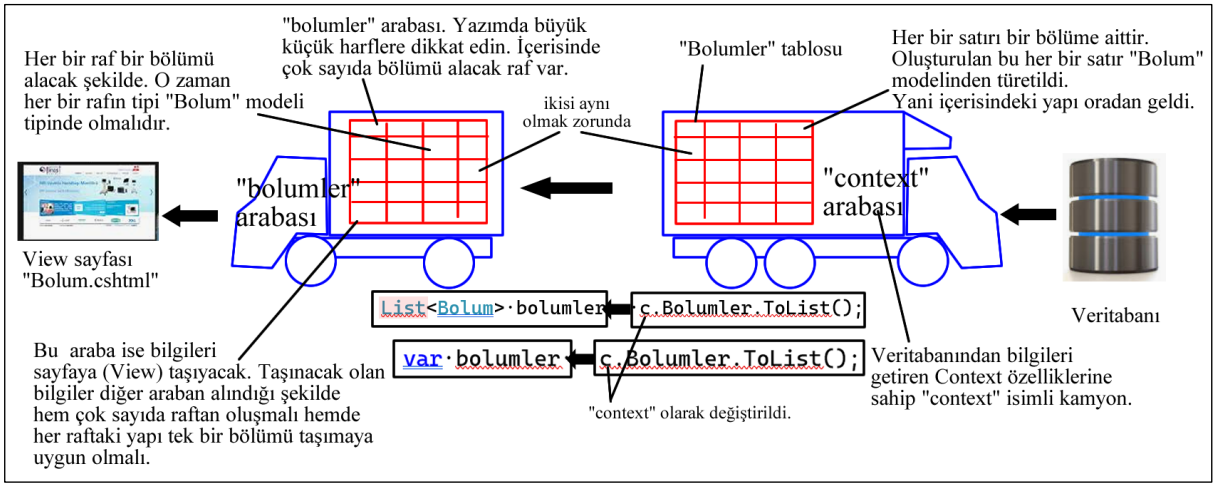
Doğrusu aşağıdaki şekilde olacaktır.

```
var bolumler = c.Bolumler.ToList();
```

Yada illa sol tarafın tipini de bilinçli olarak yazmak istiyorsak Bolum metodu bilgisini List<> şeklinde kullanmamız gerekir. Yani aşağıdaki gibi yazarsak hata almamış oluruz.

```
List<Bolum> bolumler = c.Bolumler.ToList();
```

Bu ifadeyi görsel bir anlatımla sunacak olursak aşağıdaki gibi bir görsel tanımlayabiliriz.

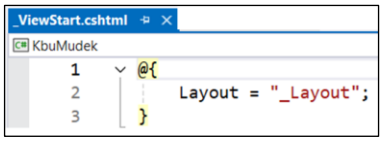


Sayfamıza bilgileri taşıyacak olan Controller içindeki kodlarımız şu şekilde oldu.

IdariYapiController.cs

```
using Microsoft.AspNetCore.Mvc;
using KbuMudek.Models;

namespace KbuMudek.Controllers
{
  public class IdariYapiController : Controller
  {
    Context c = new Context();

    public IActionResult Bolumler()
    {
      List<Bolum> bolumler = c.Bolumler.ToList();

      return View(bolumler);
    }
  }
}
```

Bu kodlar View sayfasına bilgileri gönderecektir. View sayfasının yapısı ise şu şekilde oldu. Tasarımdan gelen bazı farklı kodlar vardır ama esas ilgileneceğimiz kısım <table> kısmıdır. Sayfa aynı zamanda _layout sayfasıyla çalıştığından html yapısını gösteren bir çok kod burada gözükmemektedir. Hangi Layout sayfası ile çalıştığı burada gözükmemektedir. Default _layout sayfasının adresi ViewStart dosyasının içinde verilir. Dolayısı bir alt sayfada bildirim yapılmazsa bu sayfa ile birlikte çalışır.

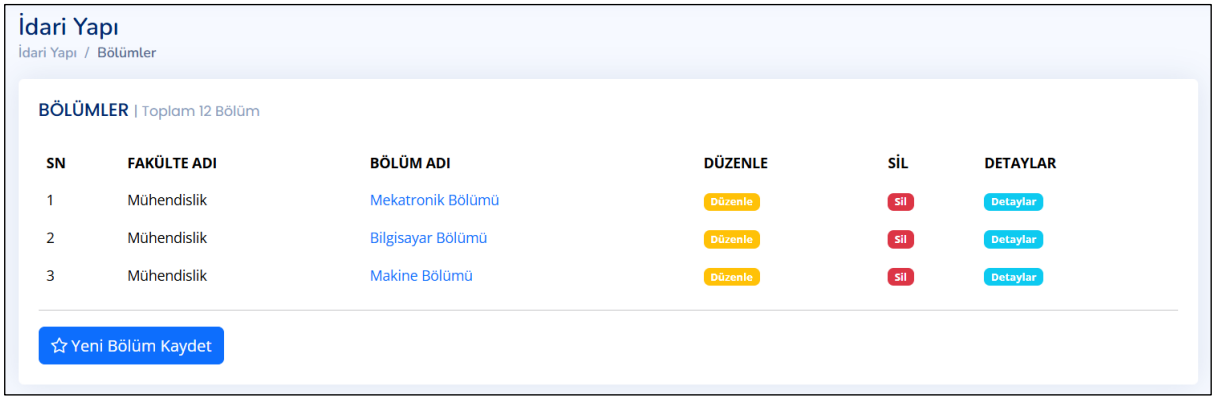


Bolumler.cshtml

```
@using KbuMudek.Models
@model List<Bolum>

<!-- ======= MAIN_BOLUM ======= -->
<main id="main" class="main">

  <div class="pagetitle">
    <h1>İdari Yapı</h1>
    <nav>
      <ol class="breadcrumb">
        <li class="breadcrumb-item"><a href="index.html">İdari Yapı</a></li>
        <li class="breadcrumb-item active">Bölümler</li>
```

```
      </ol>
    </nav>
  </div><!-- End Page Title -->

  <section class="section dashboard">
    <div class="row">

      <!-- BÖLÜMLER -->
      <div class="col-12">
        <div class="card recent-sales overflow-auto">

          <div class="card-body">
            <h5 class="card-title">BÖLÜMLER <span>| Toplam 12 Bölüm</span></h5>

            <table class="table table-borderless "> @* datatable eklenince arama çubuğu da
çıkıyor. *@
              <thead>
                <tr>

                  <th scope="col">SN</th>
                  <th scope="col">FAKÜLTE ADI</th>
                  <th scope="col">BÖLÜM ADI</th>
                  <th scope="col">GÜNCELLE</th>
                  <th scope="col">SİL</th>
                  <th scope="col">DETAYLAR</th>
                </tr>
              </thead>
              <tbody>

                @foreach (var bolum in Model)
                {
                  <tr>
                    <td scope="row"> @bolum.BolumID</td>
                    <td>Mühendislik</td>
                    <td><a href="@bolum.BolumLink" target="_blank" class="text-
primary">@bolum.BolumAdi</a></td>
                    <td><a href="#" class="badge bg-warning">Güncelle</a></td>
                    <td><a href="#" class="badge bg-danger">Sil</a></td>
                    <td><a href="#" class="badge bg-info">Detaylar</a></td>
                  </tr>
                }
              </tbody>
            </table>
            <hr />
            <a href="#" class="btn btn-primary"><i class="bi bi-star me-1"></i>Yeni Bölüm
Kaydet</a>


          </div>

        </div>
      </div><!-- End BÖLÜMLER -->

    </div><!-- End row -->
  </section>

</main><!-- End #main -->
```

Sayfanın görünümü ise şu şekildedir. _layout kısmı konulmadı.



b) Bağlantılı Olarak Tablolardan Bilgileri Getirme (Navigation Property)

Dikkat ettiyseniz yukarıdaki uygulama da sadece Bolumler tablosundaki bilgileri getirdik. Bu esnada yapılan listelemede Bölümün bağlı olduğu Fakülte adı sayfada göstermelik olarak sabit bir metin olarak gösterildi. Oysa Bolumler tablosunda Fakülte bilgisi sadece FakulteID şeklindedir. Bunu göstermekte çok bir anlama ifade etmeyecektir. Oysa olması gereken FakulteID bilgisini alıp Fakulteler tablosuna oradan Fakultenin adını getirmektir. Şimdi bu işlemi nasıl yapacağız onu görelim.

Öncelikle Bolum model yapısı içerisine Fakulte tablosuna bağlantı yapmak için Navigation Property (tablolar arası geçiş için gerekli özellik) satırını ekleyelim.

Bolum.cs model yapımız şu şekilde oldu.

Bolum.cs

```
    public class Bolum
    {
        [Key]
        public int BolumID { get; set; }
        public int FakulteID { get; set; }
        public string BolumAdi { get; set; }
        public string? BolumWebAdresi { get; set; } //Dikkat:? işareti bu null olmasına
izin verir.

        //Navigation Property (tablolar arası gezinme özelliği)
        public Fakulte Fakulte { get; set; } //Yazılar aynı oldu ama birincisi Fakültenin
model yapısı. İkincisi içerisinde bir tane fakülte bilgisini taşıyacak nesne..
    }
```

Şimdi Controller içerisinde bolumler tablosunu getirirken içerisine Fakülte bilgilerini de çekelim.

IdariYapiController.cs

```
public IActionResult Bolumler() //Bölümleri listeler
{
    //bolum bilgilerini çekerken içerisine bağlı olduğu fakülte bilgilerini de çekiyor.
    var bolumler = context.Bolumler.Include(bolum => bolum.Fakulte).ToList();
    return View(bolumler);
}
```

View sayfası tarafında artık bolum bilgisi altında bağlı olduğu fakülteye ait tüm bilgileri de görüntüleyebiliriz.

Bolumler.cshtml

```
@foreach (var bolum in Model)
{
    <tr>
        <td scope="row"> @bolum.BolumID</td>
```

```
        <td>@bolum.Fakulte.FakulteAdi</td>
        <td><a href="@bolum.BolumWebAdresi" target="_blank" class="text-
primary">@bolum.BolumAdi</a></td>
        <td><a href="/IdariYapi/BolumGuncelle/@bolum.BolumID" class="badge bg-
warning">Güncelle</a></td>
        <td><a href="#" class="badge bg-danger">Sil</a></td>
        <td><a href="#" class="badge bg-info">Detaylar</a></td>
    </tr>
}
```

B-YENİ KAYIT

a) Fakülte Bilgileri Sayfa İçinde Sabit Olarak Yeni Kayıt Ekleme

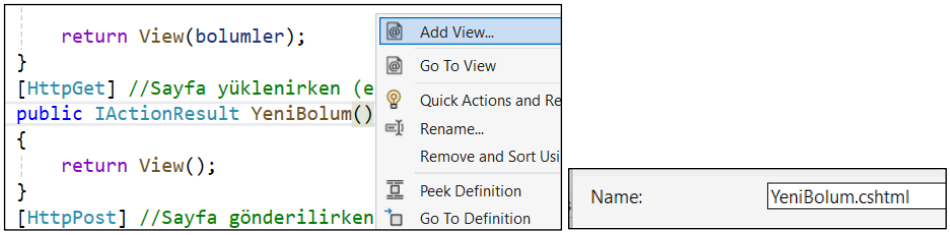
Şimdi sayfamızın altına koyduğumuz butona tıkladığımızda yeni bir bölümü kaydetmek üzere gerekli olacak sayfamızı hazırlayalım. Öncelikle sayfayı yönetecek olan kontroller içerisine iki tane ActionResult metodu ekleyeceğiz. Bu metodlardan birisi, gelen link bilgisini kullanarak sayfaya gitmemizi sağlayacak. Sayfaya bilgi götürün yada oraya yönlendiren bu tür metedların tipi [HttpGet] tipindedir. Sayfadan girilen herhangi bir bilgiyi Controller'a geri getiren ve bu bilgileri alan metodlar da [HttpPost] tipinde olacaktır. Nasıl yazıldığına kodlar içinde bakalım.

IdariYapiController.cs

```
using Microsoft.AspNetCore.Mvc;
using KbuMudek.Models;

namespace KbuMudek.Controllers
{
  public class IdariYapiController : Controller
  {
    Context context = new Context();

    public IActionResult Bolumler() //Bölümleri listeler
    {
      List<Bolum> bolumler = context.Bolumler.ToList();

      return View(bolumler);
    }
    [HttpGet] //Sayfa yüklenirken (elde edilirken-get) çalıştırılır
    public IActionResult YeniBolum()
    {
      return View();
    }
    [HttpPost] //Sayfa gönderilirken (post) çalıştırılır
    public IActionResult YeniBolum(Bolum bolum)
    {
      //Veritabanına bilgileri taşıyan context nesnemiz, oradaki Bolumler
      //tablosuna view sayfasından gelen bolum içindeki bilgileri kaydedecek.
      //ardından bilgileri tekrar görmek için yukarıdaki Bolumler Action metoduna gidecek.
      context.Bolumler.Add(bolum);
      context.SaveChanges();
      return RedirectToAction("Bolumler");
    }

  }
}
```

Yeni kayıt sayfamızı oluşturalım. YeniBolum metodumuzun üzerine sağ tuşa tıklayıp bu metodun View sayfasını oluşturalım. Açılan pencerede sayfamızın adını YeniBolum olarak verelim. Metodun adı ile aynı

olmak zorundadır. View sayfasını ilgili Views içindeki Controller klasörünün altında oluşturur. Burada controllerımız IdariYapi idi.



View sayfasının tasarımını aşağıdaki gibi yapalım. Bu sayfadan girilecek bilgileri Controllera taşımak bize bize bir araba lazım. Bu arabamız Models.Bolum olacaktır. Sayfanın en başına bu tanımlamayı yapalım.

```
@model KbuMudek.Models.Bolum
```

Sayfamızdan arabaya yüklenip götürülücek bilgileri mutlaka <form> etiketleri arasında olmalıdır. Ayrıca bu formun metodunun yapacağı işin Post (postala) türünde olması gerekir.

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

Butona basıldığında form elemanlarına girilen bilgilerin Controllera taşınabilmesi için hangi butona tıklanıldığında bu işlem yapılacak o butonun tipinin de submit (gönder) türünde olması gerekir.

```
<button type="submit" class="btn btn-primary">Bölümü Kaydet</button>
```

Form elemanlarının hangisinin veritabanındaki hangi alanla eşleşeceğini ise `asp-for=""` ifadesi ile veriyoruz. Böylece o alana girilen bilgi veritabanındaki o sutüna girilmiş olacak.

```
<input type="text" asp-for="BolumAdi" class="form-control">
```

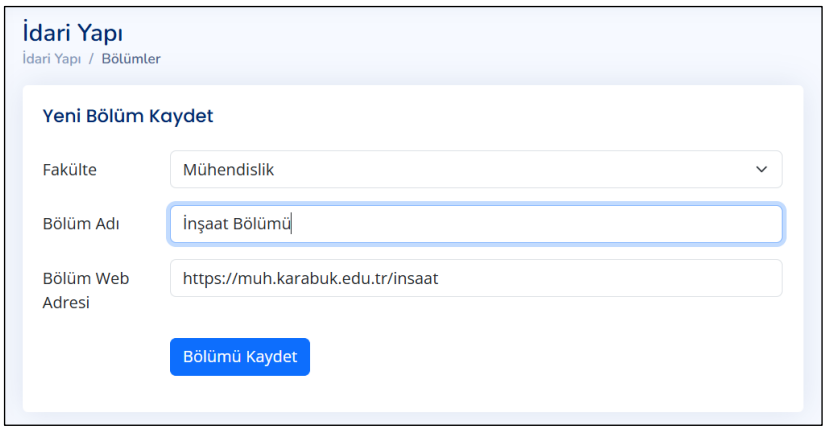
Kodların son hali şu şekilde olmaktadır.

YeniBolum.cshtml

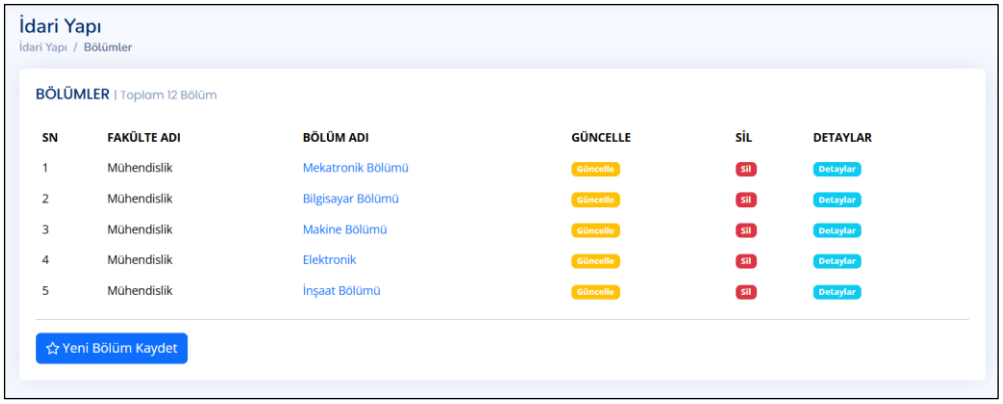
```
@model KbuMudek.Models.Bolum

<!-- ======= MAIN_BOLUM ======= -->
<main id="main" class="main">

 <div class="pagetitle">
  <h1>İdari Yapı</h1>
  <nav>
   <ol class="breadcrumb">
    <li class="breadcrumb-item"><a href="index.html">İdari Yapı</a></li>
    <li class="breadcrumb-item active">Bölümler</li>
   </ol>
  </nav>
 </div><!-- End Page Title -->

 <section class="section">
  <div class="row">
   <div class="col-lg-8">

    <div class="card">
     <div class="card-body">
      <h5 class="card-title">Yeni Bölüm Kaydet</h5>

      <!-- General Form Elements -->
      <form method="post">

       <div class="row mb-3">
        <label class="col-sm-2 col-form-label">Fakülte</label>
```

```
        <div class="col-sm-10">
         <select class="form-select" asp-for="FakulteID" aria-label="Default select
example">
          <option selected>Fakülte seç</option>
          <option value="1">Mühendislik</option>
          <option value="2">Lisansüstü Enstitüsü</option>
         </select>
        </div>
       </div>

       <div class="row mb-3">
        <label for="BolumAdi" class="col-sm-2 col-form-label">Bölüm Adı</label>
        <div class="col-sm-10">
         <input type="text" asp-for="BolumAdi" class="form-control">
        </div>
       </div>

       <div class="row mb-3">
        <label for="BolumWebAdresi" class="col-sm-2 col-form-label">Bölüm Web
Adresi</label>
        <div class="col-sm-10">
         <input type="text" asp-for="BolumWebAdresi" class="form-control">
        </div>
       </div>

       <div class="row mb-3">
        <label class="col-sm-2 col-form-label"></label>
        <div class="col-sm-10">
         <button type="submit" class="btn btn-primary">Bölümü Kaydet</button>
        </div>
       </div>

      </form><!-- End General Form Elements -->

     </div>
    </div>

   </div>
  </div>

 </section>
</main><!-- End #main -->
```

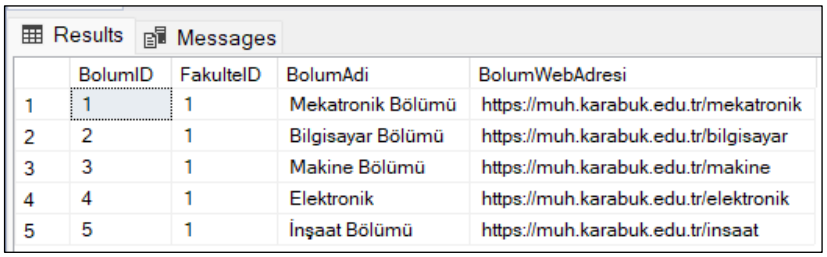
Kodların ekran görüntüsü şu şekilde olmaktadır. (Layout kısımları yoktur)



Kaydettikten sonra listeleme sayfasına göndermiştik. Bilgilerin eklendiğini görüyoruz.



Doğal olarak bilgiler veritabanına da eklenmiştir.



Results | Messages

	BolumID	FakulteID	BolumAdi	BolumWebAdresi
1	1	1	Mekatronik Bölümü	https://muh.karabuk.edu.tr/mekatronik
2	2	1	Bilgisayar Bölümü	https://muh.karabuk.edu.tr/bilgisayar
3	3	1	Makine Bölümü	https://muh.karabuk.edu.tr/makine
4	4	1	Elektronik	https://muh.karabuk.edu.tr/elektronik
5	5	1	İnşaat Bölümü	https://muh.karabuk.edu.tr/insaat

Not: Butona tıklanıldığında sayfa otomatik olarak bilgileri hangi Controller içindeki hangi Action metoduna götüreceğini anlamaktadır. Bu konuda farklı uygulamalarda bir sorun oluşursa aşağıdaki gibi controller ve action isimleri form etiketleri içinde ayrıca verilebilir.

```
<form asp-controller="IdariYapi" asp-action="YeniBolum" method="post">
```

b) Fakülte Bilgileri Veritabanından Getirilerek Yeni Kayıt Ekleme

Bir önceki uygulamada Fakülte bilgilerini sayfa içinde Option List içinde sabit olarak koyduk. Oysa bölümlerin bağlı olduğu fakülte isimleri veritabanından gelmelidir. Bu durumda Sayfadaki Controller ve sayfadaki Listeleme kısımları şu şekilde değiştirilmelidir.

Controller sayfasında değiştirilecek kısım.

IdariYapiController.cs

```
[HttpGet] //Sayfa yüklenirken (elde edilirken-get) çalıştırılır
public IActionResult YeniBolum()
{
    //Sayfaya giderken yanımızda ViewBag içerisinde Fakülte bilgilerinide götüreceğiz.
    ViewBag.Fakulteler = context.Fakulteler
        .Select(fakulte => new SelectListItem
        {
            Text = fakulte.FakulteAdi,    // veya sizin fakülte adı alanınız neyse
            Value = fakulte.FakulteID.ToString()  // Primary Key alanınız neyse
        }).ToList();

    return View();
}
```

View sayfasında değiştirilecek kısım

YeniBolum.cshtml

```
<div class="row mb-3">
<label class="col-sm-2 col-form-label">Fakülte</label>
<div class="col-sm-10">
    <select class="form-select" asp-for="FakulteID" aria-label="Default select example">
```

```
        @foreach (var fakulte in ViewBag.Fakulteler as List<SelectListItem>)
        {
            <option value="@fakulte.Value">@fakulte.Text</option>
        }
    </select>
</div>
</div>
```

C-KAYIT GÜNCELLEME

Listeleme içindeki Güncelle butonuna tıkladığımızda Linkler için temel olan 3 tane bilgiyi vermemiz gerekiyor. Bu bilgiler "Controller Bilgisi/ ActionMetod Bilgisi/ ID Bilgisi". Bu sıralamayı Adres linklerinde de sürekli olarak kullanmaktayız. Örneğin https://www.karabuk.edu.tr/bolum/guncelle/1 şeklinde yazdığımız bir adres içindeki bolum=controller adıdır, güncelle=Action metod adıdır, 1 rakamı ise güncellenecek bölümün Id sidir. Bu üçüncü bölüm adreslerde verilmesi zorunlu değildir. Ama diğer ikisini vermek zorunludur.

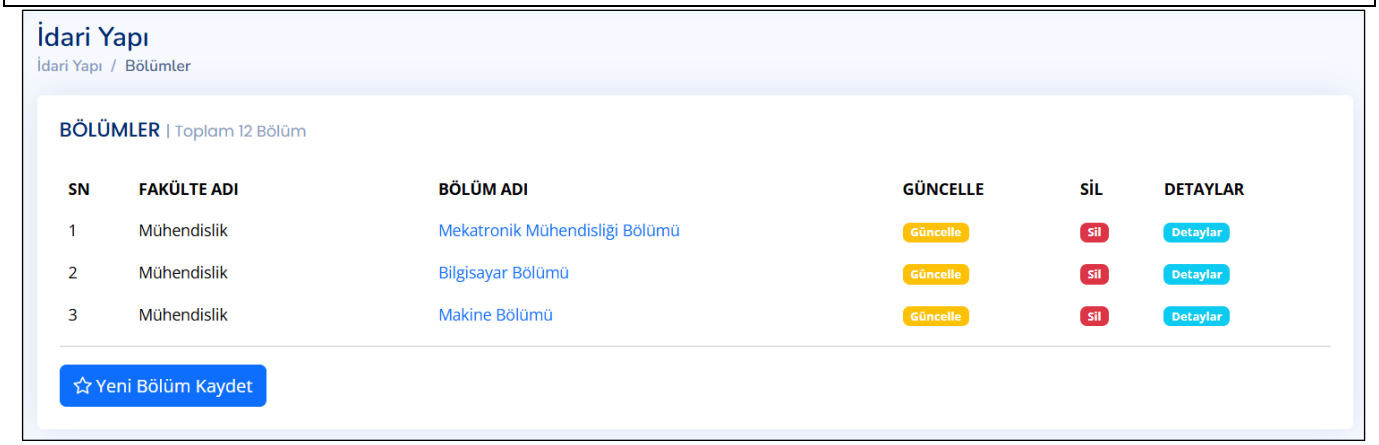
Şimdi Listeleme içindeki "Güncelle" butonlarının içinde adresimizi

```
href="/IdariYapi/BolumGuncelle/@bolum.BolumID"
```

şeklinde oluşturalım.

Bolumler.cshtml

```
@foreach (var bolum in Model)
{
    <tr>
        <td scope="row"> @bolum.BolumID</td>
        <td>Mühendislik</td>
        <td><a href="@bolum.BolumWebAdresi" target="_blank" class="text-
primary">@bolum.BolumAdi</a></td>
        <td><a href="/IdariYapi/BolumGuncelle/@bolum.BolumID" class="badge bg-
warning">Güncelle</a></td>
        <td><a href="#" class="badge bg-danger">Sil</a></td>
        <td><a href="#" class="badge bg-info">Detaylar</a></td>
    </tr>
}
```



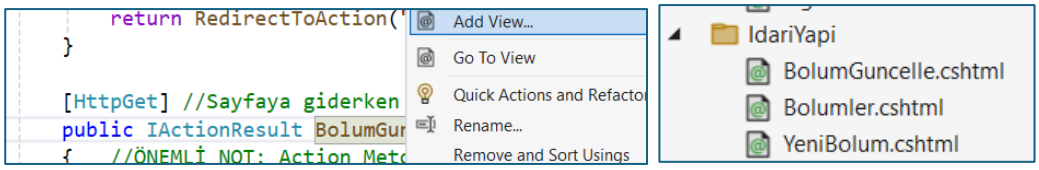
Bu linke tıklandığında gidilecek olan IdariYapi controller'ı içerisinde BolumGuncelle Action metod'unu oluşturalım. View sayfasına giderken Fakulte bilgilerini de taşıyalım. Bunun için viewBag nesnesini kullanalım.

IdariYapiController.cs

```
[HttpGet] //Sayfaya giderken çalıştırılan metod
public IActionResult BolumGuncelle(int ID)
{   //ÖNEMLİ NOT: Action Metod girişlerinde alınan Id bilgileri
    //Mutlaka "ID" yada "id" şeklinde yazılmalıdır.
```

```
    //BolumID şeklinde yazılırsa çalışmaz.

    //Sayfaya giderken yanımızda ViewBag içerisinde Fakülte bilgilerinide götüreceğiz.
    ViewBag.Fakulteler = context.Fakulteler
        .Select(fakulte => new SelectListItem
        {
            Text = fakulte.FakulteAdi,    // veya sizin fakülte adı alanınız neyse
            Value = fakulte.FakulteID.ToString()  // Primary Key alanınız neyse
        }).ToList();


    var bolum = context.Bolumler.Find(ID);
    return View(bolum);
}
```

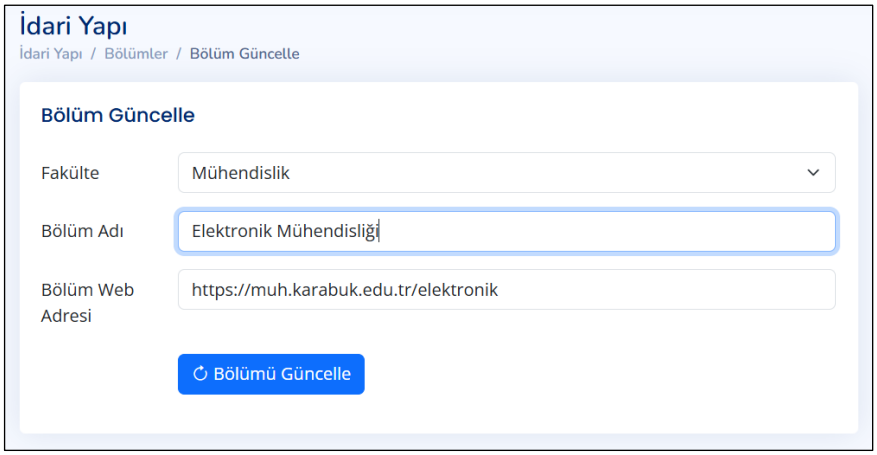
Action metod başlığına sağ tuşa tıklayıp buna ait View sayfamızı oluşturalım. View sayfamızın adını Action metod adıyla aynı yapalım (BolumGuncelle.cshtml).



View sayfamızın içi boştur. Öncelikle dışarıdan bölüme ait bilgileri alıp sayfa içinde değişik yerlerde kullanmak üzere bir model taşıma aracına ihtiyacımız vardır. Onu sayfanın en başında tanımlayalım.

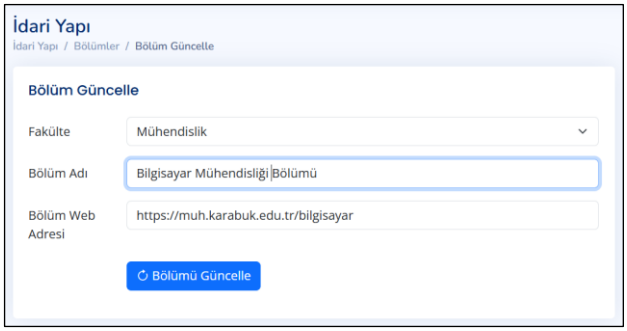
```
@model KbuMudek.Models.Bolum
```

Ardından sayfamızda Label ve Textbox'lardan oluşan bir form yapısı gerekli. Bunun için yeni kayıt kısmında oluşturduğumuz yapıyı buraya kopyalayalım.




```
@model KbuMudek.Models.Bolum

<!-- ======= MAIN_BOLUM ======= -->
<main id="main" class="main">

 <div class="pagetitle">
  <h1>İdari Yapı</h1>
  <nav>
   <ol class="breadcrumb">
    <li class="breadcrumb-item">İdari Yapı</li>
    <li class="breadcrumb-item"><a href="IdariYapi/Bolumler">Bölümler</a></li>
    <li class="breadcrumb-item active">Bölüm Güncelle</li>
   </ol>
  </nav>
 </div><!-- End Page Title -->
```

```
 <section class="section">
  <div class="row">
   <div class="col-lg-8">

    <div class="card">
     <div class="card-body">
      <h5 class="card-title">Bölüm Güncelle</h5>

      <!-- General Form Elements -->

      <form asp-controller="IdariYapi" asp-action="BolumGuncelle" method="post">
       <input type="hidden" name="BolumID" value="@Model.BolumID" />

       <div class="row mb-3">
        <label class="col-sm-2 col-form-label">Fakülte</label>
        <div class="col-sm-10">
         <select class="form-select" asp-for="FakulteID" aria-label="Default select
example">
          @foreach (var fakulte in ViewBag.Fakulteler as List<SelectListItem>)
          {
           <option value="@fakulte.Value">@fakulte.Text</option>
          }
         </select>
        </div>
       </div>

       <div class="row mb-3">
        <label for="BolumAdi" class="col-sm-2 col-form-label">Bölüm Adı</label>
        <div class="col-sm-10">
         <input type="text" asp-for="BolumAdi" class="form-control">
        </div>
       </div>

       <div class="row mb-3">
        <label for="BolumWebAdresi" class="col-sm-2 col-form-label">Bölüm Web
Adresi</label>
        <div class="col-sm-10">
         <input type="text" asp-for="BolumWebAdresi" class="form-control">
        </div>
       </div>

       <div class="row mb-3">
        <label class="col-sm-2 col-form-label"></label>
        <div class="col-sm-10">
         <button type="submit" class="btn btn-primary"><i class="bi bi-arrow-clockwise me-
1"></i>Bölümü Güncelle</button>
        </div>
       </div>

      </form><!-- End General Form Elements -->

     </div>
    </div>

   </div>
  </div>

 </section>
</main><!-- End #main -->
```



View sayfası açıldıktan sonra ilgili bölümde gerekli değişiklikleri yapıp Güncelle butonuna tıkladığımızda form etiketleri arasındaki bilgileri Post etmiş oluruz. Bu durumda buradan Controller içerisine geliriz ve burada post özelliğine sahip yani [HttpPost] özelliğine sahip aynı isimde ikinci bir Action metod bizi karşılar. Bu metodun içerisine sayfadan getirdiğimiz yeni bilgileri almamız ve veritabanına götürüp güncellememiz gerekir. Bu işlemler için ikinci action metodumuz şu şekilde olacaktır.

IdariYapiController.cs

```
[HttpPost] //Sayfadan dönüşte çalıştırılan metod.
public IActionResult BolumGuncelle(Bolum bolumEski)
{
    var bolumYeni = context.Bolumler.Find(bolumEski.BolumID);
    //Güncellenecek her eski ve yeni alanı birbirine eşitlememiz gerekir.
    bolumYeni.FakulteID = bolumEski.FakulteID;
    bolumYeni.BolumAdi = bolumEski.BolumAdi;
    bolumYeni.BolumWebAdresi = bolumEski.BolumWebAdresi;
    //Değişiklikleri kaydediyor.
    context.SaveChanges();
    //Tekrardan liste olarak görmek için Bolumler action üzerinden sayfaya gidiyor.
    return RedirectToAction("Bolumler");
}
```

Bolumler.cshtml

SN	FAKÜLTE ADI	BÖLÜM ADI	GÜNCELLE	SİL	DETAYLAR
1	Mühendislik	Mekatronik Mühendisliği Bölümü	Güncelle	Sil	Detaylar
2	Mühendislik	Bilgisayar Mühendisliği Bölümü	Güncelle	Sil	Detaylar

D-KAYIT SİLME

a) Direk Kaydı Silme

Kayıt silme işleminde herhangi bir View sayfasını açmamıza gerek yoktur. Yani silincek kayıtları öncesinde gösterip ardından silme işlemi gibi bir uygulamız yoktur. Sonuçta listemizde hangi kaydı silmek istediğimizi görüyoruz, butona tıkladığımızda sadece "emin misiniz?" şeklinde bir teyit alıp silme işlemini gerçekleştirebiliriz. Bu durumda butona tıkladığımızda direk Controller içindeki silme işlemini gerçekleştirecek Action metoduna gitmemiz ve orada silme olayı gerçekleştirmemiz yeterlidir.

Buna göre Bolumler sayfasındaki sil butonunun içine adresimizi yazalım.

```
@foreach (var bolum in Model)
{
 <tr>
  <td scope="row"> @bolum.BolumID</td>
  <td>@bolum.Fakulte.FakulteAdi</td>
  <td><a href="@bolum.BolumWebAdresi" target="_blank" class="text-
primary">@bolum.BolumAdi</a></td>
```

```
  <td><a href="/IdariYapi/BolumGuncelle/@bolum.BolumID" class="badge bg-
warning">Güncelle</a></td>
  <td><a href="/IdariYapi/BolumSil/@bolum.BolumID" class="badge bg-danger">Sil</a></td>
  <td><a href="#" class="badge bg-info">Detaylar</a></td>
 </tr>
}
```

Action metodu içerisini yazalım. Action metodlarda dışarıdan alınan Id lerin yazım şekli mutlaka Id yada ID şeklinde olmalıdır. Yani BolumID vs şeklinde yazarsak metod çalışmaz. ID bilgisiniz aldıktan sonra Find(ID) metodu ile Bolumler tablosu içindeki ilgili satırı yani Bolum bilgisinin tüm alanlarını alıp "var bolum" nesnesinin içerisine atıyoruz. Burada var tip tanışmalaması eşittirin sağ tarafından nasıl gelirse o şekilde tipini ayarlar manasındadır. İstersek tipi direk te yazabiiriz. Yani "Bolum bolum" şeklinde yazarsak buradaki "Bolum" modelimizin adı olmuş olur.

Hangi bölüm bilgisinin silineceğini belirledikten sonra artık onu Bolumler tablosundan kaldırabiliriz. Bu işlemi ise (`Bolumler.Remove(bolum);`). Daha sonra (`.SaveChanges();`) işlemi ile veritabanında olay gerçekleştirilmiş oluyor. Dikkat edersek bütün bu işleri yani Veritabanında gerçekleştirilecek işleri controllerın en üst kısmında ürettiğimiz context nesnesi yapmaktadır (kodların tamamı burada değildir). İşlem bittikten sonra kullanıcıya listenin son halini göstermek için yine Bolumler sayfasına gidiyoruz. Fakat dikkat edersek yönlendirme yaparken direk View sayfası yazılmıyor. View sayfasını girişi kontrol eden Controller içindeki action metodun adı yazılıyor. O ilgili metodun içine girdiğimizde ise "return View()" komutunu görünce ona bağlı View sayfasına gidecektir.

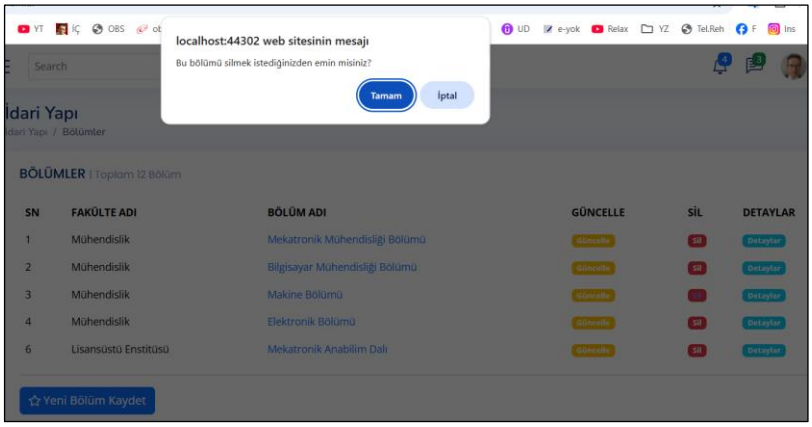
IdariYapiController.cs

```
[HttpGet] //Bunu yazmasakta olur. Varsayılan Action tipi zaten [HttpGet] tipindedir.
public IActionResult BolumSil(int ID)
{   //ÖNEMLİ NOT: Action Metod girişlerinde alınan Id bilgileri
    //Mutlaka "ID" yada "id" şeklinde yazılmalıdır.
    var bolum = context.Bolumler.Find(ID);
    context.Bolumler.Remove(bolum);
    context.SaveChanges();
    return RedirectToAction("Bolumler");
}
```

b) Onay Alarak Kaydı Silme

Fakat yaptığımız bu işlemde bir eksiğimiz vardır. Direk butona tıkladığımızda kayıt silinmektedir. Yani kullanıcı yanlışlıkla butona bassa kayıt silinecektir. Bunu önlemek için View sayfası içerisindeki ilgili linki şu şekilde düzenlememiz yeterli olacaktır.

```
<td><a href="/IdariYapi/BolumSil/@bolum.BolumID" onclick="return confirm('Bu bölümü silmek
istediğinizden emin misiniz?');" class="badge bg-danger">Sil</a></td>
```



c) Birbirine Bağlantılı Tablolardan Kayıt Silme

Şu ana kadar yaptığımız işlemde tek başına bir Tablodan kaydı sildik. Yani burada bir bölümü sildik ama eğer ona bağlı başka tablolar olsaydı o tablolardaki bilgiler boşa çıkardı ve sayfalar hata vermeye başlardır. Yani diyelim Sildiğimiz bölüme bağlı müfredat ve dersleri atamış olsaydık bu sefer o derslerin karşılığı olan Bölüm bulunmayacağından veriler hatalı duruma düşerdi. Bunu engellemek için bir tabloya başka tablolar bağlı ise o tablodan direk kaydın silinmesi hatalı olacaktır. Bunun yerine ilgili kayıtlara Active/Pasif (1/0) olarak kullanabileceğimiz bir bool sutunu ekleyip, kayıt silme işlemini sadece o satırı Pasif yaparak kullanmamız daha uygun olacaktır. Bu tür durumlarda aşağıdaki iki yaklaşım kullanılır:

Soft Delete (Yumuşak Silme) — Tavsiye Edilen Yöntem

Kaydı gerçekten silmek yerine, örneğin AktifMi gibi bir sütunla pasif hale getirirsiniz. Sistem sadece AktifMi = true olan kayıtlarla çalışır.

Avantajları:

- Veri kaybı olmaz.
- Geri alma (undo) mümkündür.
- İlişkisel bütünlük korunur.
- Loglama/audit trail için uygundur.

Kullanım:

```
// Kullanıcıyı pasif yapmak
var user = _context.Users.Find(id);
user.AktifMi = false;
_context.SaveChanges();
```

Filtreleme:

```
var aktifKullanicilar = _context.Users.Where(u => u.AktifMi).ToList();
```

Gelişmiş: EF Core 5+ ile Global Query Filter:

```
protected override void OnModelCreating(ModelBuilder builder)
{
    builder.Entity<AppUser>().HasQueryFilter(u => u.AktifMi);
}
```

Hard Delete (Gerçekten Silmek) — Dikkatli Kullanılmalı

Eğer ilişkili veriler varsa ve silmeye kalkarsanız, foreign key constraint nedeniyle hata alırsınız. Örnek hata:

```
“The DELETE statement conflicted with the REFERENCE constraint...”
```

Bu durumda seçenekler:

- Cascade Delete: İlişkili kayıtları da otomatik siler (Tehlikeli olabilir).
- Manuel Silme: İlişkili kayıtları önce elle silmeniz gerekir.

Cascade Delete Örneği:

```
modelBuilder.Entity<Parent>()
    .HasMany(p => p.Children)
```

```
    .WithOne(c => c.Parent)
    .OnDelete(DeleteBehavior.Cascade);
```

Ancak bu yöntem genellikle hassas sistemlerde (kullanıcılar, roller, loglar vb.) kullanılmamalıdır.

Sonuç: Hazırlayacağımız projelerde ilişkili tablolar çok olacağından ve kullanıcı girişi işlemleri bulunacağından kişilere bağlı verileri yada birbaşka tabloya bağlı verileri silmek yerine aktif-pasif yöntemini kullanalım. Buna veritabanı tablolarımızda gerekli değişiklikleri yapıp yeniden Migration güncellemesi yapalım.