# 

T.C.

CUMHURİYET ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ÇALIŞAN KAYIT İŞLEMLERİ

## BARIŞ CAN AKDAĞ

## 2015141056

### *Özet:*

Projem CRUD işlemlerini servis tabanlı olarak yapabilmektedir.Projem 3 tane tane temel yapı üzerine kuruludur.Bunlar client(istemci) , server(sunucu), Web APİ’den oluşmaktatır.Client’de calisan ekleme,düzenleme ve silme işlemleri yapılmaktatır.Web APİ burada bir köprü görevi üstlenmektedir.Yani bu köprüden geçmek isteyen herşeye izin verir.Örneğin windows,lunix,android vb. Cihazların client-server haberleşmesine izin verir.Client’den server tarafına bir bilgi gönderirken(POST (Ekleme,Silme,Düzenleme)) bunu Web APİ üzerindende yollar.Web APİ kendisine gelen verileri XML formatında server tarafına yollar.Client Server’dan bir veri talep ettiğinde talep edilen veri(GET) XML formatında cliente gönderilir.Bu sayede birbirinden farklı platformlar arası veri alışverişi sağlanır.Kodlanması C# SQL Server(ISS) EntitiyFramework kullanarak yapılmıştır.Tüm bu anlattıklarım Back-end ‘de gerçekleşen olaylardır.Front-End de ise uygulamamız kullanıcı arayüzüne sunulması için bootstrap kütüphanesi kullandım. Basit sade ve kullanışlı bir arayüzdür.

***Anahtar Kelimeler:***  Web APİ,Client,Server,EntitiyFramework,XML,CRUD

L

*Abstract*

My project is able to perform CRUD operations on a service basis. The project is based on 3 basic structures. These are composed of client, server, web API. Adding, editing and deleting works in the client are done here. The bridge permits everything that wants to pass through this bridge. The windows, lunix, android and so on. Allows devices to communicate with client-server. When the server sends information to the server side (POST (Add, Delete, Edit)) it sends it to the web API. The web API sends the incoming data to the server in XML format. (GET) is sent to the client in XML format. The data exchange between the different platforms is provided between them. The encoding was done using C # SQL Server (ISS) EntitiyFramework. This is what happens in the back-end. I used the bootstrap library to present it to the user interface. Simple simple and useful interface.***.***

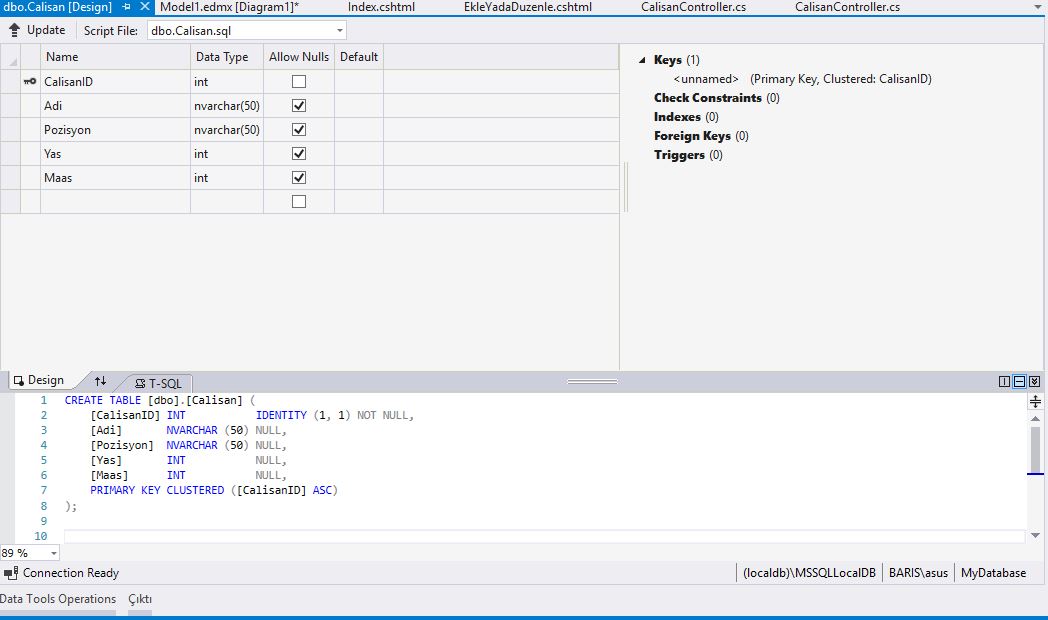
***Keywords:*** Web APİ,Client,Server,EntitiyFramework,XML,CRUD

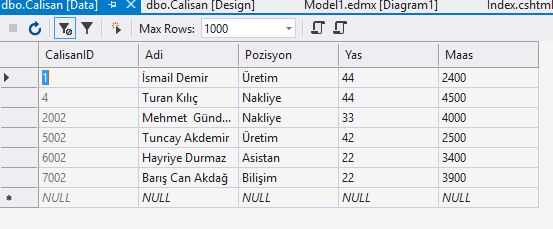
## Giriş

Uygulamanın amacı bir şirketteki çalışanların kayıtlarının tutulması üzerinedir.Adısoyadı,yaşı,maaşı ve pozisyon bilgileri veritabanında tutulmalıdır.Bir client ve server olmalıdır.Client tarafında çalışanların bilgileri gözükmeli istenildiği zaman çalışan silinebilmesi,güncellenebilmeli yada yeni bir çalışan eklenebilmelidir(Kısaca CRUD işlemlerini yapmalıdır.).Server tarafında tüm bu bilgiler bir veritabanında tutulmalıdır.

## UYGULAMA

Database : Veritabanın nasıl oluşturulduğu ve tutulan nitelikler.





barisCrudService (CalisanControllers):Web servis metodları ve servis ile veritabanı ilişkileri(EntitiyFramework) yapılıyor.Gelen CRUD işlemlerine karşılık bu controllerde bulunan hazır metotlar ile servere gönderiyor. HTTP istekleri bu controller içinde bulunuyor.(HttpActionResult).

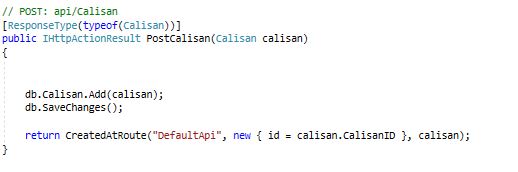
GetCalisan Metodu : Calisan bilgilerini Servere yollar.



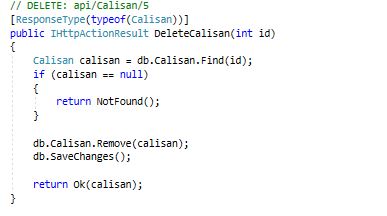
PutCalisan Metodu: Veritabanındaki bir çalışanın id’sine bakarak güncelleme yapılmasını sağlar.



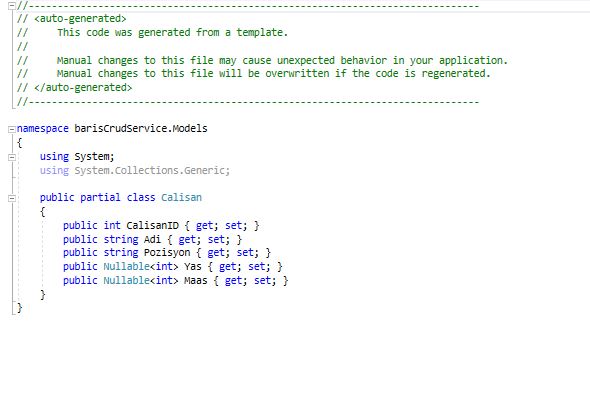
PostCalisan Metodu: Veritabanına eklenmek çalışanı Server’a post ederek eklenmesini saglar.



DeleteCalisan Metodu:Veritabanından silinmek istenen çalışını bulur .Bulduktan sonra siler ve veritabanını günceller.



Veritabanında oluşturduğumuz çalışan sınıfında niteliklerimiz veri tipi tutuluyor.

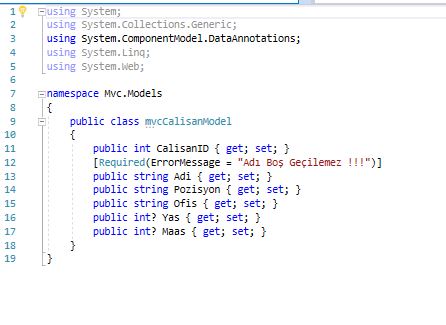


Mvc Kısmı:

GenelDegiskenler Sınıfı:Mvc adlı projemde HTTP metodlarını tanıması için Web API Client ‘i projemize ekliyoruz.Bu sayede barisCrudService projemize bağlantımızı sağlamış olduk.barisCrudService bulunan CalisanController ‘ e bağlantımızı sağlıyoruz.Bu işlemlerden sonra Client ve barisCrudService birbirlerine bağlanmış oluyor



mvcCalisanModel(Veritabanı Sınıfı) : Mvc kısmına barisCrudService bulunan veritabanını tanıtmak amacıyla bir model kullanıp veritabanında bulunan niteliklerimizi giriyoruz.)



Mvc Kısmı CalisanController :Controller kullanarak Servis üzerinden aldğımız çalışan bilgilerini bir html sayfasında üzerinde kullanıcılarıa gösteriyoruz.Aynı zamanda bu html sayfası üzerinden kullanıcının çalışan ekleme cıkarma duzenleme gibi işlemlerini yapabilmesi bu kısım ile sağlanır.

ActionResult Index Metodu: Veritabanından çalışan bilgilerini html sayfasına aktarılmak amacıyla yazılıyor(READ).barisCrudService’den aldığı calisan bilgilerini bir listeye atar ve listeyi görüntüler.

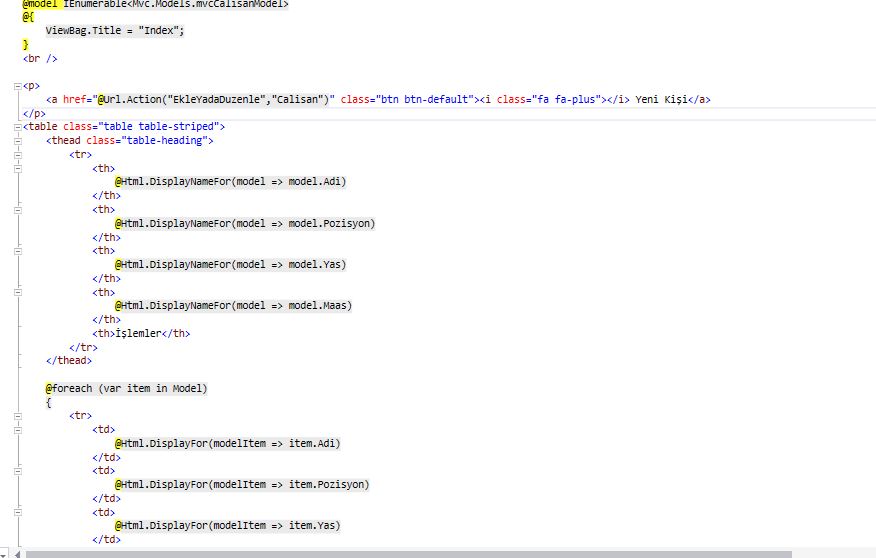


EkleYadaDuzenle ve Sil Metodları:EkleYadaDuzenle metodu Create(Ekleme) ve Uptade(Duzenleme)işlemlerini yapar.Yapılan işlemlerden sonra güncel tablo geri döndürülür.Sil metodu çalışan id’sine göre silme işlemi yapar.



View :Kullanıcının göreceği Client tarafı yazılmıştır..

Index View: Veritabanından alınan veriler kullancıya gösterilir.aKullanıcının çalışan ekleme, düzenleme ve silme yapabileceği operasyonlar için buton yapılandırmaları oluşturulmuştur.



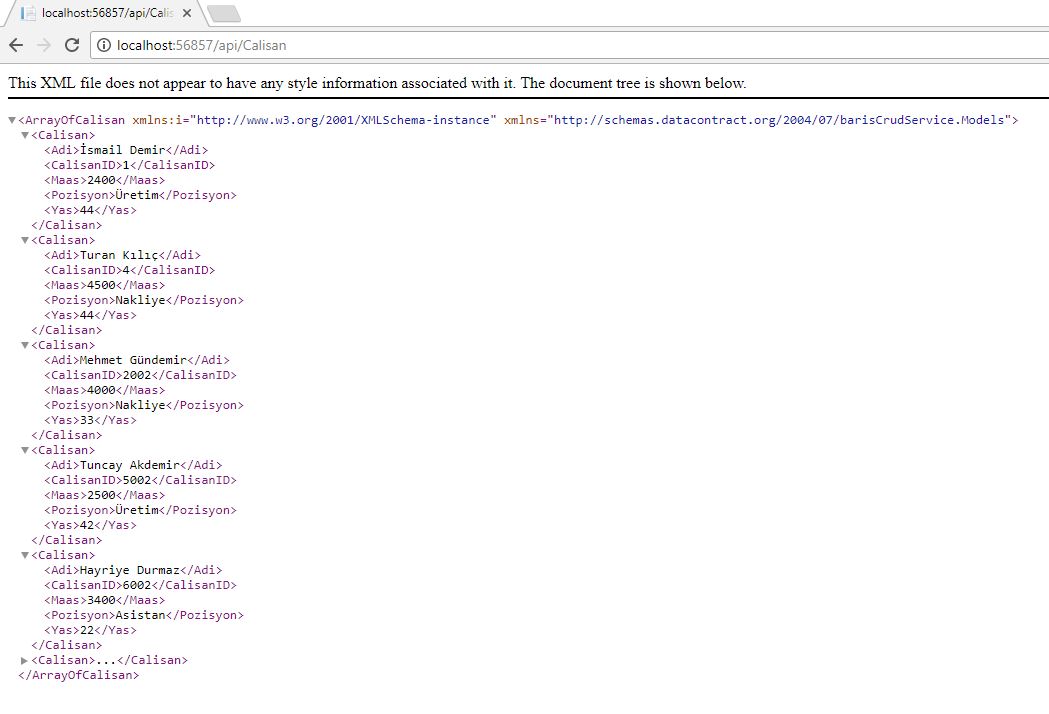


EkleYadaDuzenle View : Ekleme veya Düzenleme butonuna basıldığında açılması gereken HTML sayfasına yönlendirme yapılmıştır.Bu sayfa bir Form sayfasıdır.Veritabanına formda yazılı olan bilgiler gönderilir ve Index sayfasında görüntülenir.



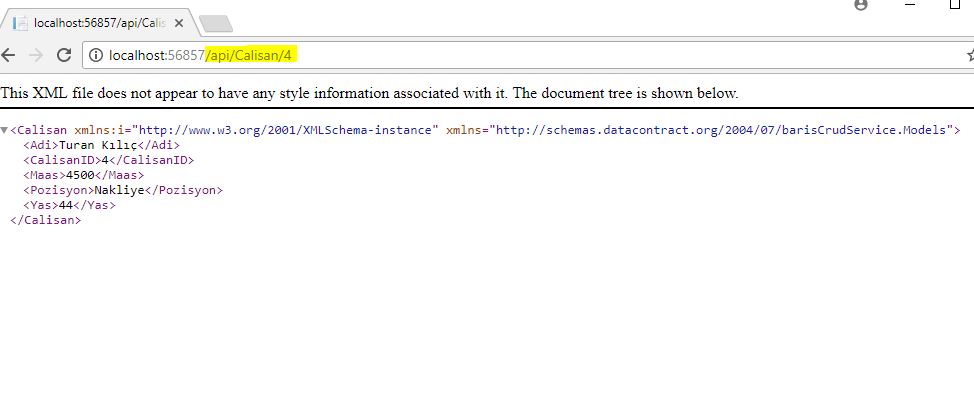
SONUÇLAR

barisCrudService Cıktısı : Veritabanındaki bilgiler Web API’de XML biçimde tutulur.Buradaki önemli nokta clientten-servere veya serverden-cliente giden ve gelen veriler XML biçimdedir.

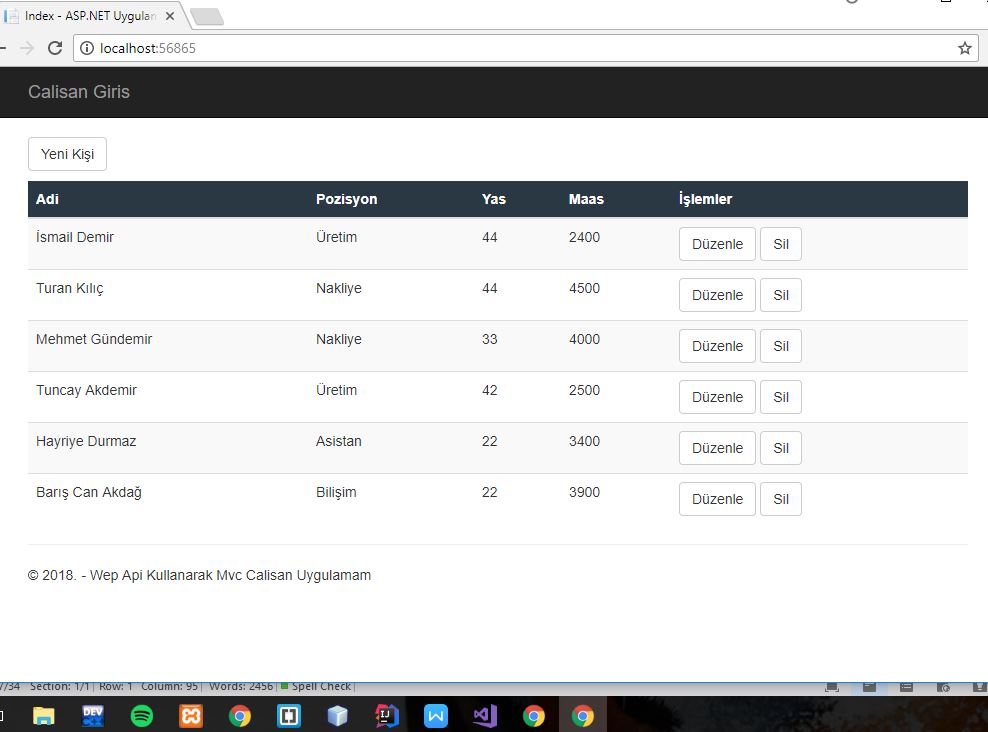


RouteConfig : HTML sayfalarımızda nerelerin gözükeceğini tanımlar.url : “{controller}/{action}/{id} ” yönlendirmesi ile veritabındaki çalışan varlıklıklarımızı web servis üzerinde görüntülememizi sağlamaktadır

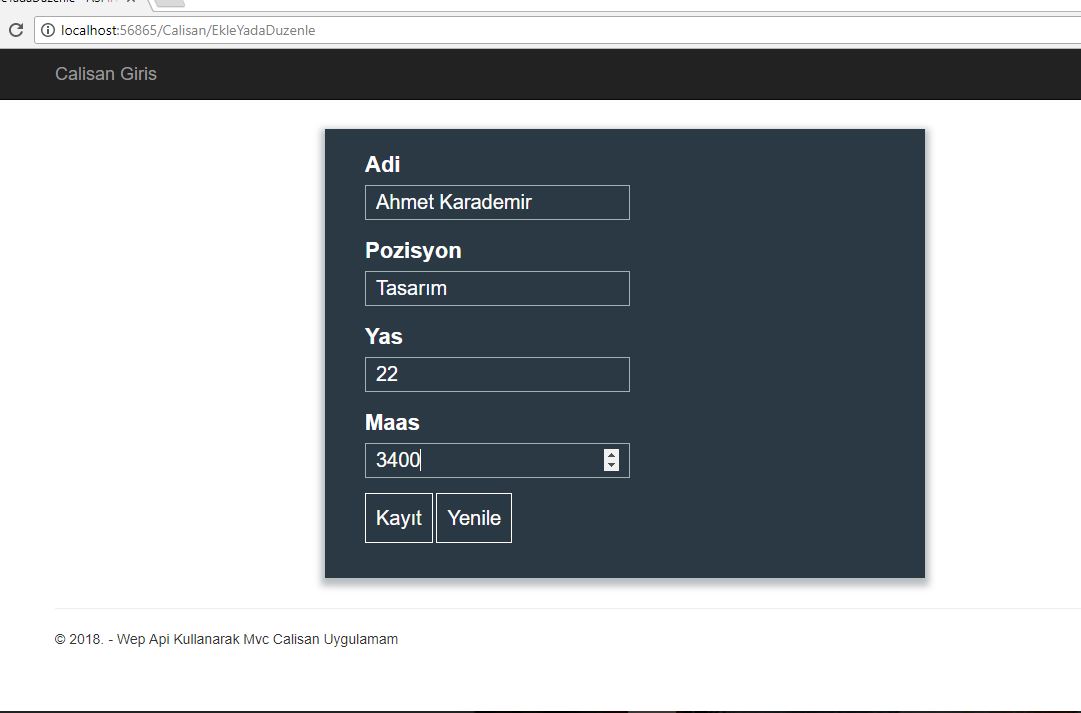
Örneğin id’si 4 olan çalışanımızı görüntüleyelim.

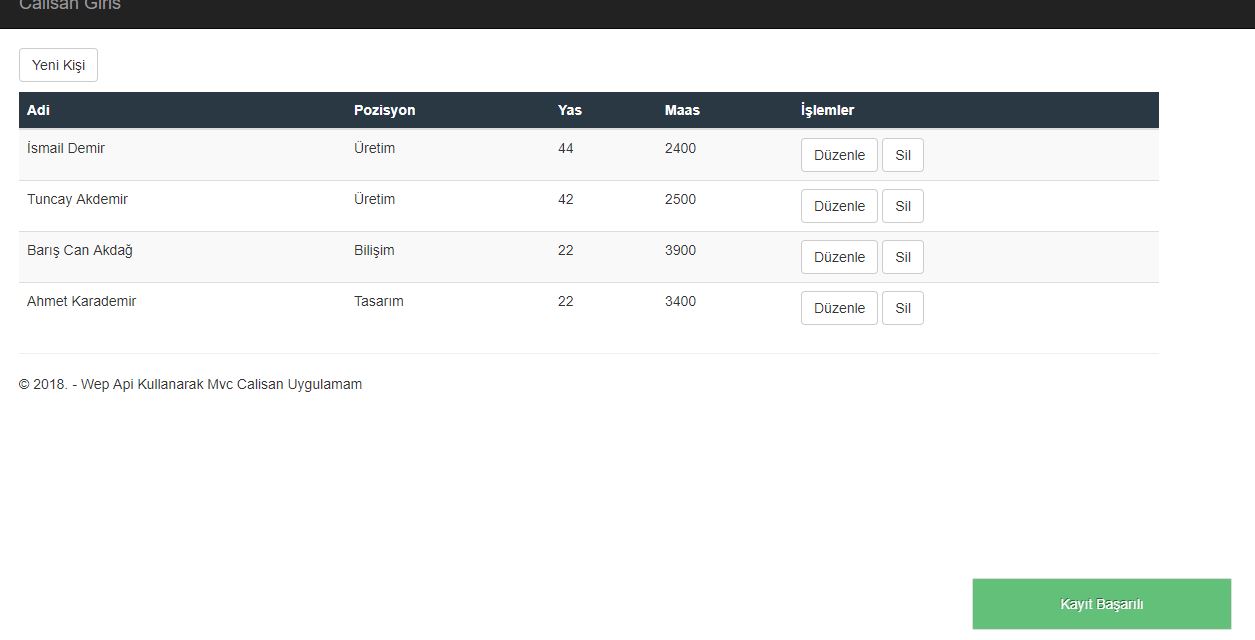


Index View Çıktısı: Olusturdugumuz tablo sayesinde veritabından alınan calisan varlıkları HTML sayfasında görüntüleniyor.Tanımladığımız butonlarda aynı sayfa içerisinde gözüküyor.

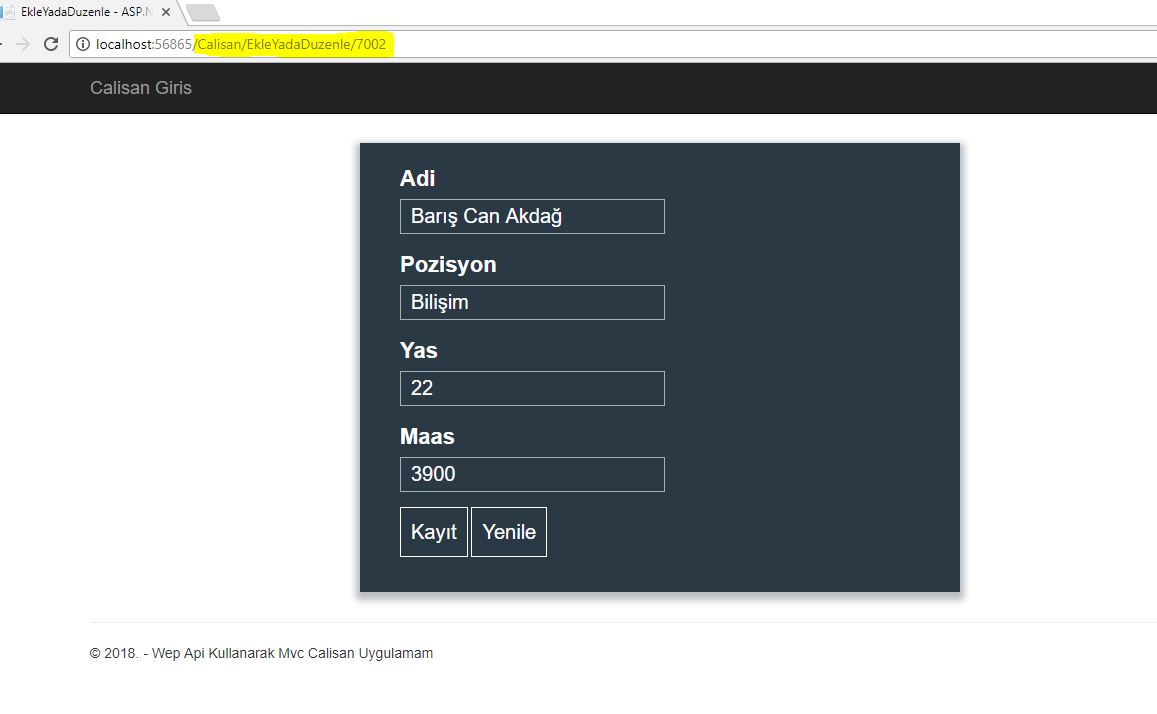


Yeni Kişi botununa tıkladığımızda boş bir form sayfası açılıyor ve verileri girmemiz bekleniyor.Burada sadece Adi boş geçilemez !!

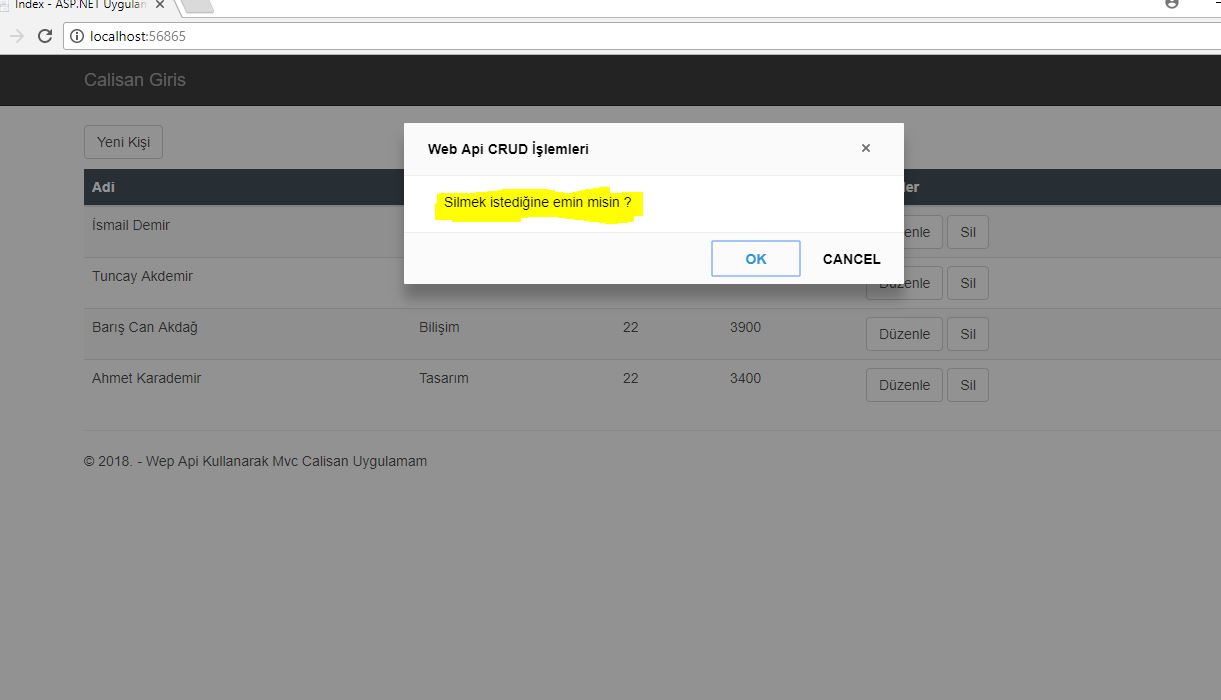
Kayıt botununa bastığımız zaman bir hata yok ise çalışan veritabanına kayıt olur.Bizi tekrar index sayfasına yönlendirir.Eklediğimiz çalışan tabloda görünür.



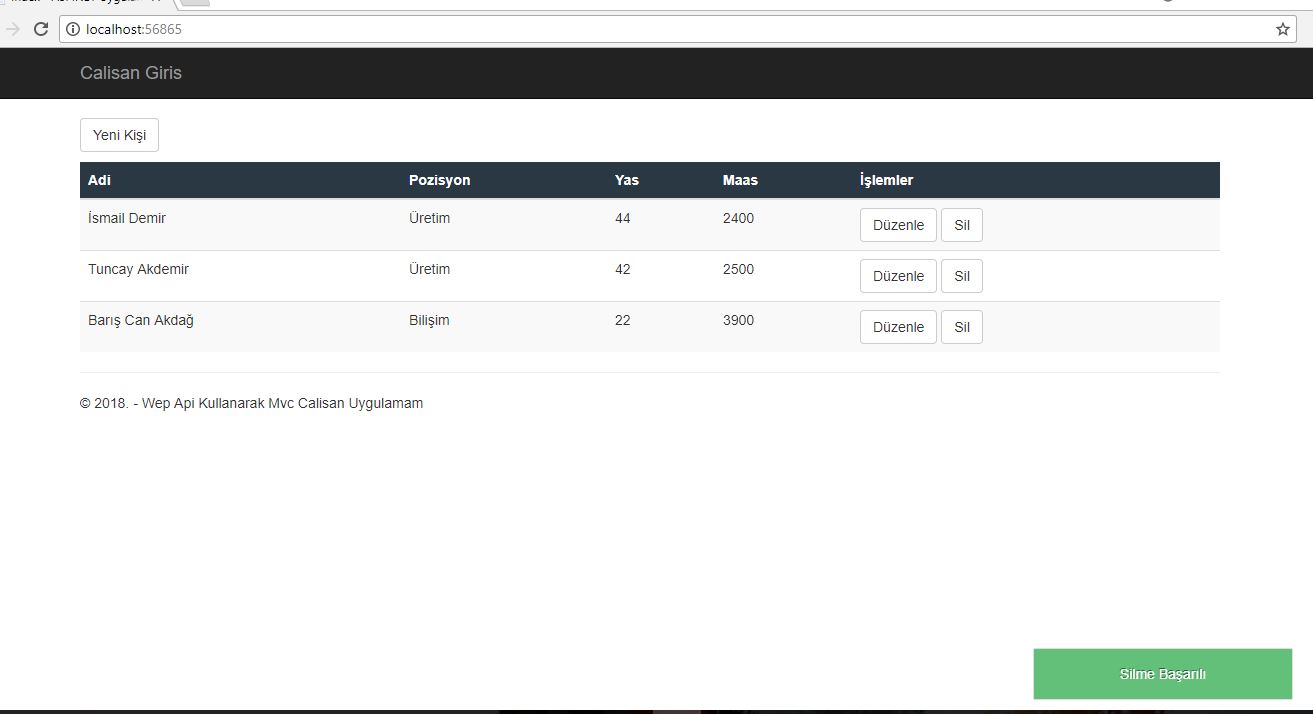
Anasayfada düzenle botununa bastığımız zaman çalışanın bilgilerini bize getirir ve forma yerleştirir.Düzenle botunu EkleYadaDuzenle dosyasını çalışının id’si ile birlikte getirir.Değişiklik yapar isek yapılan değişilik veritabanında güncellenir.Bizi tekrar anasayfaya atar.Güncel tablo görünür.



Sil botununa bastığımızda bize bir uyarı mesajı verir.



Ok’e bastığımızda çağırdığımız çalışan veritabanından silinir.Silindikten sonra tablo güncellenir.Görüldüğü gibi Ahmet Karademir’i sildik.Cancel dediğimizde uyarı kaybolur.



## kaynAK KODLAR

barisCrudService KISMI

Calisan.Cs Model sınıfı Web API kısmında tanımlandı.EntitiyFramework kullanırak verilerimizi nesnel olarak tanımlanıyoruz ve bize büyük bir kolaylık sağlıyor.EntitiyFramework’ün metotlarını kullanıp iş yükümüzü hafifletip güvenliği ön plana çıkartıyoruz.

namespace barisCrudService.Models

{

using System;

using System.Collections.Generic;

public partial class Calisan //Calisan sinifinde veritabanı niteliklerini tanımlıyoruz.(EntitiyFramework’u eklediğimiz zaman otomatik olarak dolduruluyor.)

{ // Tüm değişkenler public olarak tanıtılarak diğer sınıfların erişimine açık hala getiriliyor.

public int CalisanID { get; set; }

public string Adi { get; set; }

public string Pozisyon { get; set; }

public Nullable<int> Yas { get; set; }

public Nullable<int> Maas { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Web.Http;

using System.Web.Http.Description;

using barisCrudService.Models;

namespace barisCrudService.Controllers

{

public class CalisanController : ApiController

{

private MyDatabaseEntities1 db = new MyDatabaseEntities1();//Veritabanından bir nesne oluşturuluyor.

public IQueryable<Calisan> GetCalisan() //Çalışan tablosu çağırılıyor

{

return db.Calisan;

}

// GET: api/Calisan/5

[ResponseType(typeof(Calisan))]

public IHttpActionResult GetCalisan(int id) //Çalışan varlıkları id’lerine göre veritabanından okunuyor.Mvc’de bu bilgiler kullanırak kullanıcıya sunulması sağlanacak.XML olarak JSON tarafından.

{

Calisan calisan = db.Calisan.Find(id); //Calisan sınıfından calisan nesnesi üretilip Find metodu ile id’lerine göre calisan nesnesine atılıyor.

if (calisan == null)

{

return NotFound(); //nesne boş ise bulunamadı hatası ver.

}

return Ok(calisan); // nesne null değil ise calisan nesnesini yolla.Tüm tablo gezilir.

}

// PUT: api/Calisan/5

[ResponseType(typeof(void))]

public IHttpActionResult PutCalisan(int id, Calisan calisan )// Update yapmamızı sağlayan bir metottur.

{

db.Entry(calisan).State = EntityState.Modified;//Entry metodu Veritabanına giriş yapacak. State ile bildirecek.Sonrasında EntrySatete.Modefied ile Veritabanının degiştirilmesi gerektiğini söylerek değişiklikleri yapılıcaktır.

try

{

db.SaveChanges();//bir sıkıntı yok ise veritabanına değişiklik kayıt edilir.

}

catch (DbUpdateConcurrencyException)//Bir sorun meydana gelirse program catch bloğuna girer.

{

if (!CalisanExists(id))

{

return NotFound(); //İd’ler uyuşmuyor ise bulunamadı hatası verir.

}

Else // sorun id değilse throw fırlatır.

{

throw;

}

}

return StatusCode(HttpStatusCode.NoContent); //başarı gönderilir.

}

// POST: api/Calisan //Calisan tablosuna yeni calisan eklememizi sağlar.

[ResponseType(typeof(Calisan))]

public IHttpActionResult PostCalisan(Calisan calisan)

{

db.Calisan.Add(calisan); //Gelen nesne Calisan tablosuna Add metodu ile eklenir.

db.SaveChanges(); //Eklendikten sonra değişiklikler kayıt edilir.

return CreatedAtRoute("DefaultApi", new { id = calisan.CalisanID }, calisan); //Kayıtların id’leri geri döndürülür.

}

// DELETE: api/Calisan/5

[ResponseType(typeof(Calisan))]

public IHttpActionResult DeleteCalisan(int id)//Gelen id’i veritabanından siler.

{

Calisan calisan = db.Calisan.Find(id); //calisan nesnesine veritanındaki gelen id’i aktarır.

if (calisan == null)

{

return NotFound(); //Öyle bir çalışan yoksa bulanamadı hatası döndürür.

}

db.Calisan.Remove(calisan); //İd var ise veritabanından gelen id’e ait çalışanı siler

db.SaveChanges(); //Veritabanındaki değişkiliği kayıt eder.

return Ok(calisan); //Geriye calisan nesnesi döndürülür.

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

private bool CalisanExists(int id)

{

return db.Calisan.Count(e => e.CalisanID == id) > 0;

}

}

}

using System.Web.Http;

using System.Web.Http.Description;

using WebApiKismi.Models;

MVC KISMI

Web API kısmında server tarafına veri tabanımızı ve yapılacak CRUD işlemleri ile alakalı kodları yazdık. API suan sadece server tarafını tanıyor.Şimdi Client tarafana tanıtmaya başlayalım.Amacımız Client veriyi API’ye yollayacak API’de Servere yollacak bağlantıları yazalım.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Web;

namespace Mvc

{

public static class GenelDegiskenler

{

public static HttpClient WepApiClient = new HttpClient();//HttpClient oluşturuluyor.Oluşturduğumuz nesne sayesinde API’ya bağlanabileceğiz.Sınıfımız static tanımlanıyor ve bu projede bu nesneye her yerden ulaşabiliriz.

static GenelDegiskenler()

{

WepApiClient.BaseAddress = new Uri("http://localhost:56857/api/Calisan");//API’dan alıcağımız bilgilerin tam adresini giriyoruz.XML kodlarının geldiği yer.

WepApiClient.DefaultRequestHeaders.Clear();//Bağlantımız temizleniyor.

WepApiClient.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json") );//Client tarafına verilerin JSON tipinde veri geliceğini bildiriyoruz.

}

}

}

mvcCalisanModel Sınıfı : Veri tabanımızı Mvc kısmına da tanımlıyoruz.

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

namespace Mvc.Models

{

public class mvcCalisanModel

{

public int CalisanID { get; set; }

[Required(ErrorMessage = "Adı Boş Geçilemez !!!")]

public string Adi { get; set; }

public string Pozisyon { get; set; }

public string Ofis { get; set; }

public int? Yas { get; set; }

public int? Maas { get; set; }

}

}

}

Mvc CalisanController : Sayfalarda hangi işlemler yapılacağını ve bu işlemlerin sayfalara arka planda yansıtılmasını sağlıyor.

using Mvc.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Web;

using System.Web.Mvc;

using Mvc.Controllers;

namespace Mvc.Controllers

{

public class CalisanController : Controller

{

// GET: Calisan

public ActionResult Index()//ActionResult metodu tanımlanıyor

{

IEnumerable<mvcCalisanModel> clsList;mvcCalisanModel modelindeki veritabanı listeye alınıyor.

HttpResponseMessage response = GenelDegiskenler.WepApiClient.GetAsync("Calisan").Result;//Web API ‘den veriler JSON tipinde Client kısmına GetAsync metodunun icine yazılan Calisan ile barisCrudService kısmındaki controller icindeki veriler XML formatında çağırılıyor.

clsList = response.Content.ReadAsAsync<IEnumerable<mvcCalisanModel>>().Result; //XML formatındaki veri okunur ve veritabanına yerleştirilir.ReadAsAsync JSON tipindeki veriyi okur.

return View(clsList);//Okunan veriler View de kullanıcıya gösterilir.

}

public ActionResult EkleYadaDuzenle(int id = 0)//EkleYadaDuzenleme kısmının GET yeridir.

{

if (id == 0)

return View(new mvcCalisanModel()); //İd sıfır ise veritabanı döndürülür.

else

{

HttpResponseMessage response = GenelDegiskenler.WepApiClient.GetAsync("Calisan/" + id.ToString()).Result;

return View(response.Content.ReadAsAsync<mvcCalisanModel>().Result);//Veri varsa olan verileri tekrardan Index Aciton'ınındaki işlemlerle servis tabanlı olarak dönecektir.

}

}

[HttpPost]

public ActionResult EkleYadaDuzenle(mvcCalisanModel cls) //Verileri Client'dan Server'a parametre olarak Veritabanı sınıfından bir nesne geçiliyor bu nesne ile yapılan veritabanı işlemleri kolaylıkla yapılacaktır.

{

if (cls.CalisanID == 0) //Calisan İd’si sıfır ise

{

HttpResponseMessage response = GenelDegiskenler.WepApiClient.PostAsJsonAsync("Calisan", cls).Result;//Kullanıcıdan alınan veriyi Web API kısmına gönderir ve PostAsJsonAsync metodu ile Web API

kısmındaki CalisanContollers kısmına giderek post metodunu calıstırır ve Json tipinde bu veriyi veritabanına ekleyip sonrasında kaydeder.

TempData["SuccessMessage"] = "Kayıt Başarılı";//Sayfanın sağ altında kayıt başarılı mesajı çıkar.

}

else

{

HttpResponseMessage response = GenelDegiskenler.WepApiClient.PutAsJsonAsync("Calisan/" + cls.CalisanID, cls).Result;

TempData["SuccessMessage"] = "Düzenleme Başarılı";//CalisanID sıfır değilse güncelleme yapılacaktır.Web API kısmındaki Calisan controllerına gidip PUT metodunu cağırır ve oraya CalisanID ile güncellenecek çalışan bulunur ve güncellenir. }

return RedirectToAction("Index"); //İşlemler bitince Index View’e gider.

}

public ActionResult Sil(int id)//Veritabanından çalışan siler.

{

HttpResponseMessage response = GenelDegiskenler.WepApiClient.DeleteAsync("Calisan/" + id.ToString()).Result;//id ile birlikte barisCrudService kısmındaki Delete Metoduna gidilir ve o metod calısarak silme işlemini gercekleştirir.Veri silindiğinde geriye değer döndürmeye gerek yoktur

TempData["SuccessMessage"] = "Silme Başarılı";//Kullanıcıyı bilgilendirir.

return RedirectToAction("Index"); //İşlem bittiğinde Index View’e gider.

}

}

}

Index View’i : Bu sayfada veritabanındaki kayıtlar gözükecektir. Veriler bir tablo şeklinde gözükecek.Alert kutusu için AlertifyS dll kurdum.

@model IEnumerable<Mvc.Models.mvcCalisanModel>

@{

ViewBag.Title = "Index";

}

<br />

<p>

<a href="@Url.Action("EkleYadaDuzenle","Calisan")" class="btn btn-default"><i class="fa fa-plus"></i> Yeni Kişi</a>

</p>

<table class="table table-striped">

<thead class="table-heading">

<tr>

<th>

@Html.DisplayNameFor(model => model.Adi)//Veri tabanı modelinden Adi niteliği çağırılarak tablo başlığı haline getiriliyor(Adiolarak).Diğer işlemlerde aynı şekild

</th>

<th>

@Html.DisplayNameFor(model => model.Pozisyon)

</th>

<th>

@Html.DisplayNameFor(model => model.Yas)

</th>

<th>

@Html.DisplayNameFor(model => model.Maas)

</th>

<th>İşlemler</th>

</tr>

</thead>

@foreach (var item in Model)//Döngü ile tanıttığımız model listesi yani veri tabanı listesindeki her eleman item değişkeni içerisine alınıyor

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Adi)

</td>

<td>

@Html.DisplayFor(modelItem => item.Pozisyon)

</td>

<td>

@Html.DisplayFor(modelItem => item.Yas)

</td>

<td>

@Html.DisplayFor(modelItem => item.Maas)

</td>

<td>

<a href="@Url.Action("EkleYadaDuzenle", "Calisan", new { id = item.CalisanID})" class="btn btn-default"><i class="fa fa-pencil"></i> Düzenle</a>//Duzenle Butonuna basıldığında Calisan Controller altındaki EkleYadaDuzenle ActionResult’u çağırılıyor ve aynı zamanda bu action’a id değişkeninee calisan id atılarak gidiliyor ki hangi id’yi düzenlemek istediğimizi bilelim.

<a onclick="Sil(@item.CalisanID)" class="btn btn-default"><i class="fa fa-trash"></i> Sil</a>//Sil butonuna basıldığında CalisanID ile Delete ActionResult’una gidiliyor fakat direk silinmiyor aşağıdaki alert kutusundan onay alındığında siliniyor.

</td>

</tr>

}

</table>

@section scripts{

<script>

$(function () {

var successMessage = '@TempData["SuccessMessage"]'

if (successMessage != '')

alertify.success(successMessage);

});

function Sil(id) {//Bir alert kutusu oluşturularak silme işleminin doğrulanması isteniyor eğer onaylanırsa Sil Action Resultuna id ile birlikte gidiliyor ve silme işlemi bu koddan sonra gerçekleştiriliyor.

alertify.confirm('Web Api CRUD İşlemleri','Silmek istediğine emin misin ?',function(){

window.location.href = '@Url.Action("Sil","Calisan")/'+id;

},null );

}

</script>

EkleYadaDuzenle View’i : Bu View Kayıt Ekleme veya düzenleme yapmak istediğimizde gözüküyor.

@model Mvc.Models.mvcCalisanModel//mvcCalisanModel Veri tabanı sınıfı liste halinde değil sınıf olarak tanıtılıyor.Cünkü burda bir görüntüleme işlemi yapmayacağız sadece veri ekleme veya değiştirme işlemi yapacağız.

@{

ViewBag.Title = "EkleYadaDuzenle";

}

<div class="form-body">//Veri girişi için form oluşturuluyor

@using (Html.BeginForm())

{

@Html.HiddenFor(model => model.CalisanID)

<div class="form-group"> //Niteliğinin verisinin girilmesi bekleniyor Adi olarak Adiiçine veri girişi olarak Adi ve onay mesajı Adi niteliği olduğunu belirtiyoruz.

@Html.LabelFor(model => model.Adi)

@Html.EditorFor(model => model.Adi)

@Html.ValidationMessageFor(model => model.Adi)

</div>

<div class="form-group">

@Html.LabelFor(model => model.Pozisyon)

@Html.EditorFor(model => model.Pozisyon)

@Html.ValidationMessageFor(model => model.Pozisyon)

</div>

<div class="form-group">

@Html.LabelFor(model => model.Yas)

@Html.EditorFor(model => model.Yas)

@Html.ValidationMessageFor(model => model.Yas)

</div>

<div class="form-group">

@Html.LabelFor(model => model.Maas)

@Html.EditorFor(model => model.Maas)

@Html.ValidationMessageFor(model => model.Maas)

</div>

<div class="form-group">

<input type="submit" value="Kayıt" class="btn button" />//Kaydet butonuna basarsak bunun tipinin bir submit olduğunu bilir ve kaydeder.

<input type="reset" value="Yenile" class="btn button" />//Reset butonu da tipinin reset olduğunu bilir ve formu yenileyerek boş halini geri getirir.

</div>

}

</div>

@section scripts{

@Scripts.Render("~/bundles/jqueryval");

}