

**KOCAELİ ÜNİVERSİTESİ   
KOCAELİ MESLEK YÜKSEKOKULU  
BİLGİSAYAR TEKNOLOJİLERİ BÖLÜMÜ BİLGİSAYAR PROGRAMCILIĞI**

**PROJE**

**DERGİ ABONE TAKİP SİSTEMİ**

**MART 2023**



**KOCAELİ ÜNİVERSİTESİ**

**KOCAELİ MESLEK YÜKSEKOKULU**

**BİLGİSAYAR TEKNOLOJİLERİ BÖLÜMÜ BİLGİSAYAR PROGRAMCILIĞI**

**PROJE**

**DERGİ ABONE TAKİP SİSTEMİ**

**Öğr. Görevlisi Mustafa Of ................................................**

**Tezin Savunulduğu Tarih: 11.06.2023**

**ÖNSÖZ VE TEŞEKKÜR**

Bu tez çalışması, Kocaeli Üniversitesi'nde geçirdiğim eğitim süreci boyunca edindiğim bilgi ve becerilerin bir ürünüdür. Bu çalışma, yazılım dünyasında derinlemesine araştırmalar yapmayı hedefleyen bir yazılımcı adayı olarak bana sağlanan fırsatları yansıtmaktadır.

Tez çalışması, yazılım geliştirme alanında daha derin bir anlayış sağlamayı ve ilgili akademik çevrelerde tartışmalara katkıda bulunmayı hedeflemektedir. Aynı zamanda, gelecekteki yazılım geliştiricilere ilham kaynağı olmayı ve onlara yol göstermeyi amaçlamaktadır. Bu tez, benim için yazılım dünyasında yeni bir başlangıç noktasıdır.

Öncelikle, Kocaeli Üniversitesi'ne, sağladığı eğitim olanakları ve kaliteli akademik kadrosu için teşekkür etmek istiyorum. Üniversite, yazılım alanında sağladığı güçlü temel ve ileri düzey dersleriyle beni profesyonel bir yazılım geliştirici olma yolunda destekledi. Özellikle, bu tez çalışması sürecinde değerli öğretim üyesi Mustafa hocamın önderliğindeki rehberlik ve önerileri büyük önem taşıdı.

Mustafa hocam, tezimin her aşamasında büyük bir özveriyle beni yönlendirdi. Yazılım konusundaki derin bilgisi, deneyimi ve uzmanlığı, tezimin kalitesini artırmama yardımcı oldu. Kendisinin cesaretlendirici ve destekleyici yaklaşımı, zorlu sorunlarla karşılaştığım zamanlarda bile motivasyonumu yüksek tutmamı sağladı.

Ayrıca, tezimin hazırlık sürecindeki danışmanlık desteğinden yararlandığım diğer öğretim üyelerine de teşekkür etmek istiyorum. Onların değerli önerileri ve eleştirileri, yazılım alanındaki anlayışımı ve yeteneklerimi geliştirmeme yardımcı oldu.

Mehmet Mert COMBAŞ

Haziran-2023

**İçindekiler**

**Önsöz ve Teşekkür…………………………………………………………………2**

**Özet (Türkçe)……………………………………………………………………….3**

**Abstract……………………………………………………………………………..4**

**Genel Bilgiler …………………………………………………………………...6-70**

**Sonuç ve Öneriler ………………………………………………………………. 71**

**Kaynaklar ……………………………………………………………………….. 71**

**Kişisel Yayın ve Eserler …………………………………………………………71**

**Özgeçmiş …………………………………………………………………………71**

**Ekler …………………………………………………………………………… 72-74**

# DERGİ ABONE TAKİP SİSTEMİ

# ÖZET

Dergi abone takip sisteminin amacı dergi satıcılarının onlara aylık abone olan kişileri bilgilerini görme sisteme ekleme ve sistemden silme veya güncelleme için amaçlanmış bir yazılım projesidir.

Bu sistem Asp.Net teknolojisi ile Model-Views-Controller yapısı kullanılarak yazılmıştır. Dergi abone takip sistemi, Asp.Net teknolojisiyle geliştirilmiş ve Model-Görünüm-Kontrolcü yapısını kullanmıştır. Satıcılar, aylık abonelerin bilgilerini görüntüleyebilir, ekleme, silme veya güncelleme işlemlerini gerçekleştirebilir.

**Anahtar Kelimeler**: Abone, Asp.Net, Controller, Dergi, Model, Views

**MAGAZİNE SUBSCRİOPTİON TRACKİNG SYSTEM**

**ABSTRACT**

The purpose of the magazine subscription tracking system is to provide magazine vendors with a software project that allows them to view, add, delete, or update the information of monthly subscribers. This system has been developed using Asp.Net technology and follows the Model-View-Controller architectural pattern. Magazine vendors can view subscriber information and perform operations such as adding, deleting, or updating records.

**Keywords:** Asp.Net, Controller, Magazine, Model, Subscriber, Views

**GİRİŞ**

Bu tez çalışması, dergi abone takip sisteminin tasarımı ve geliştirilmesi üzerine bir araştırmayı temsil etmektedir. Günümüzde dergi yayıncılığı sektöründe, abonelik yönetimi ve takibi önemli bir rol oynamaktadır. Bu bağlamda, dergi satıcılarının abonelerin bilgilerini etkin bir şekilde yönetebilecekleri bir yazılım projesi ihtiyacı ortaya çıkmaktadır.

Bu tez çalışması, Asp.Net teknolojisi ve Model-Görünüm-Kontrolcü (Model-View-Controller) yapısı kullanılarak bir dergi abone takip sistemi geliştirmeyi amaçlamaktadır. Sistem, dergi satıcılarına, aylık abone olan kişilerin bilgilerini görme, sisteme ekleme, silme ve güncelleme gibi işlemleri gerçekleştirme imkanı sunmaktadır.

Tez çalışmasının amacı, dergi satıcılarının abonelik yönetim süreçlerini kolaylaştırmak, veri güncellemelerini hızlandırmak ve etkin bir şekilde takip edebilmelerini sağlamaktır. Bu tez kapsamında, dergi abone takip sistemi için gereksinim analizi, tasarım, uygulama ve test aşamaları ele alınacak ve sistemin başarıyla tamamlanması hedeflenecektir.

Sonuç olarak, bu tez çalışması dergi yayıncılığı sektöründe abonelik yönetimine yönelik bir yazılım projesinin tasarım ve geliştirme sürecini içermektedir. Umarım bu çalışma, dergi satıcılarına etkili bir abone takip sistemi sağlamak ve sektördeki abonelik yönetimi süreçlerini iyileştirmek adına önemli bir katkı sunar.

# Genel Bilgiler

Dergi abone takip sistemi projesini yazarken kullanılan teknolojiler:

Mvc(Model-View-Controller) yapısı kullanılarak asp.net yazılım teknolojisi ile ve diğer linq, entity framwork, migrations, resposityory parrent , unit of work pattern yazılım teknolojileri kullanılmıştır.  
Sistemde Data ve Data.Model adında 2 farklı veri tabanı işlemleri ve bilgileri tutacak arabirim (interface) kullanılmıştır.

Controller içerisinde AlinanDergi , Dergi , Kategori , Uye , Uyelik , Yazar isimleri ile kontrolcüler vardır.

HelperClass adlı dosyamda şifreleme işlemi vardır. Şifreleme adlı sınıf içerisindedir. SHA\_1 ve MD5 formatında şifreleme işlemi parola verileri için kullanılmak üzere yazıldı. Script adlı dosyada tasarımda kullanılan javascript dosyaları bulunmaktadır.

Views klasöründe kontrolcüden gelen bilgilere göre oluşturulmuş ekle, güncelle, ındex gibi görünüm sayfaları bulunmaktadır.

Content isimli dosyalar da css , resimler bulunmaktadır.

Sisteme “Uye” kontrolcüsü sayesinde abone eklenebilir , silinebilir , güncellenebilir. Bunlar içinde Data.Modelden dönen sorgu sonucunda kontrolcüden geçerek görünüm sayfasındaki action bağlantıları ile gerçekleşir.

Uye modelini içeren kodlar:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.Model

{

public class Uye : BaseEntity

{

[Required]

[Column(TypeName = "varchar")]

[MaxLength(70)]

public string Ad { get; set; }

[Required]

[Column(TypeName = "varchar")]

[MaxLength(70)]

public string Soyad { get; set; }

[Column(TypeName = "char")]

[MaxLength(400), MinLength(10)]

public string Adres { get; set; }

[Column(TypeName = "char")]

[MaxLength(11), MinLength(11)]

public string Tel { get; set; }

[Required]

public DateTime KayitTarihi { get; set; }

[Column(TypeName = "nvarchar")]

[MaxLength(250)]

public string Mail { get; set; }

[Column(TypeName = "char")]

[MaxLength(32), MinLength(32)]

public string Sifre { get; set; }

[Required]

public int Ceza { get; set; }

[Column(TypeName = "char")]

[MaxLength(1), MinLength(1)]

public string Yetki { get; set; }

public virtual List<AlinanDergi> AlinanDergiler { get; set; }

}

}

Bu kodlar, "Uye" adında bir model sınıfını tanımlar. Bu sınıf, "BaseEntity" adlı başka bir temel sınıftan miras alır. `Ad`, `Soyad`, `Adres`, `Tel`, `KayitTarihi`, `Mail`, `Sifre`, `Ceza`, `Yetki` ve `AlinanDergiler` gibi özelliklere sahiptir. Bazı özelliklerin üzerinde `[Required]`, `[Column]`, `[MaxLength]`, `[MinLength]` gibi öznitelikler kullanılmıştır. Bu öznitelikler, özelliklerin gereklilik, sütun türü, maksimum uzunluk, minimum uzunluk gibi özelliklerini belirtir.

Aşağıda, UyeController ve Index views kodlarını da inceleyelim:

UyeController kodları:

using Dergi\_Abone\_Takip\_ASP.NET.Data.Model;

using Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork;

using System;

using System.Web.Mvc;

namespace Dergi\_Abone\_Takip\_ASP.NET.Controllers

{

public class UyeController : Controller

{

private UnitOfWork unitOfWork;

public UyeController()

{

unitOfWork = new UnitOfWork();

}

// GET: Uye

public ActionResult Index()

{

var uyeler = unitOfWork.GetRepository<Uye>().GetAll();

return View(uyeler);

}

public ActionResult Ekle()

{

return View();

}

[HttpPost]

public JsonResult EkleJson(string uyeAd, string uyeSoyad, string uyeAdres, string uyeTel)

{

if (!string.IsNullOrEmpty(uyeAd) && !string.IsNullOrEmpty(uyeSoyad))

{

Uye uye = new Uye();

uye.Ad = uyeAd;

uye.Soyad = uyeSoyad;

uye.Adres = uyeAdres;

uye.Tel = uyeTel;

uye.Ceza = 0;

uye.KayitTarihi = DateTime.Now;

unitOfWork.GetRepository<Uye>().Add(uye);

var durum = unitOfWork.SaveChanges();

if (durum > 0)

{

return Json("1");

}

else

{

return Json("0");

}

}

else

{

return Json("bosOlamaz");

}

}

[HttpPost]

public JsonResult SilJson(int uyeId)

{

var uye = unitOfWork.GetRepository<Uye>().GetById(uyeId);

if (uye != null)

{

unitOfWork.GetRepository<Uye>().Delete(uye);

unitOfWork.SaveChanges();

return Json("1");

}

return Json("0");

}

public ActionResult Guncelle(int uyeId)

{

var uye = unitOfWork.GetRepository<Uye>().GetById(uyeId);

return View(uye);

}

[HttpPost]

public JsonResult GuncelleJson(int uyeId, string uyeAd, string uyeSoyad, string uyeAdres, string uyeTel)

{

if (!string.IsNullOrEmpty(uyeAd) && !string.IsNullOrEmpty(uyeSoyad))

{

var uye = unitOfWork.GetRepository<Uye>().GetById(uyeId);

uye.Ad = uyeAd;

uye.Soyad = uyeSoyad;

uye.Adres = uyeAdres;

uye.Tel = uyeTel;

unitOfWork.GetRepository<Uye>().Update(uye);

var durum = unitOfWork.SaveChanges();

if (durum > 0)

{

return Json("1");

}

else

{

return Json("0");

}

}

else

{

return Json("bosOlamaz");

}

}

}

}

Index.cshtml kodları:

@{

ViewBag.Title = "Aboneler";

}

@using Dergi\_Abone\_Takip\_ASP.NET.Data.Model

@model List<Uye>

<h2>ABONELER</h2>

<table id="example" class="display" style="width:100%">

<thead>

<tr>

<th>Ad</th>

<th>Soyad</th>

<th>Adres</th>

<th>Telefon</th>

<th>Kayıt Tarihi</th>

<th>Ödeme Geciktirme Cezası</th>

<th>Güncelle</th>

<th>Sil</th>

</tr>

</thead>

<tbody>

@foreach (var uye in Model)

{

<tr>

<td>@uye.Ad</td>

<td>@uye.Soyad</td>

<td>@uye.Adres</td>

<td>@uye.Tel</td>

<td>@uye.KayitTarihi</td>

<td>@uye.Ceza TL</td>

<td><a href="@Url.Action("Guncelle","Uye",new { uyeId = uye.Id })" class="btn btn-custom">Güncelle</a></td>

<td><button class="uyeSil btn btn-danger" value="@uye.Id">Sil</button></td>

</tr>

}

</tbody>

<tfoot>

<tr>

<th>Ad</th>

<th>Soyad</th>

<th>Adres</th>

<th>Telefon</th>

<th>Kayıt Tarihi</th>

<th>Ödeme Geciktirme Cezası</th>

<th>Güncelle</th>

<th>Sil</th>

</tr>

</tfoot>

</table>

```

Guncelle.cshtml kodları:

```html

@{

ViewBag.Title = "Abone Güncelle";

}

@using Dergi\_Abone\_Takip\_ASP.NET.Data.Model

@model Uye

<h2>ABONE GÜNCELLE</h2>

<form class="form-horizontal">

<div class="form-group">

<label class="col-md-12">Ad</label>

<div class="col-md-12">

<input id="uyeAd" value="@Model.Ad" type="text" class="form-control" placeholder="Abone adını giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Soyad</label>

<div class="col-md-12">

<input id="uyeSoyad" value="@Model.Soyad" type="text" class="form-control" placeholder="Abone soyadını giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Adres</label>

<div class="col-md-12">

<input id="uyeAdres" value="@Model.Adres" type="text" class="form-control" placeholder="Adresinizi giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Tel</label>

<div class="col-md-12">

<input id="uyeTel" value="@Model.Tel" type="text" class="form-control" placeholder="Tel noyu giriniz.">

</div>

</div>

<div class="form-group">

<div class="col-md-12">

<input type="button" data-id="@Model.Id" id="uyeGuncelle" class="btn btn-block btn-info" value="Güncelle" />

</div>

</div>

</form>

```

Ekle.cshtml kodları:

```html

@{

ViewBag.Title = "Abone Ekle";

}

<h2>ABONE EKLE</h2>

<form class="form-horizontal">

<div class="form-group">

<label class="col-md-12">Ad</label>

<div class="col-md-12">

<input id="uyeAd" type="text" class="form-control" placeholder="Abone adını giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Soyad</label>

<div class="col-md-12">

<input id="uyeSoyad" type="text" class="form-control" placeholder="Abone soyadını giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Adres</label>

<div class="col-md-12">

<input id="uyeAdres" type="text" class="form

-control" placeholder="Adresinizi giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Tel</label>

<div class="col-md-12">

<input id="uyeTel" type="text" class="form-control" placeholder="Tel noyu giriniz.">

</div>

</div>

<div class="form-group">

<div class="col-md-12">

<input type="button" id="uyeEkle" class="btn btn-block btn-info" value="Ekle" />

</div>

</div>

</form>

Bu kodlar, bir MVC (Model-View-Controller) uygulamasında Uye modeline dayalı bir veritabanı tablosu ve bu tabloya CRUD (Create, Read, Update, Delete) işlemlerini gerçekleştirmek için kullanılan bir controller ve view'ları içerir.

UyeController, Uye modeline bağlı olarak çalışır ve Index() yöntemi, veritabanındaki tüm üyeleri alır ve Index view'ına gönderir. Ekle() yöntemi, üye ekleme işlemi için bir view döndürür. EkleJson() yöntemi, Ajax isteği alır ve gelen verileri kullanarak bir üye oluşturur ve veritabanına ekler. SilJson() yöntemi, Ajax isteği alır ve gelen üye kimliği ile eşleşen bir üyeyi veritabanından siler. Guncelle() yöntemi, güncelleme işlemi için bir view döndürür. GuncelleJson() yöntemi, Ajax isteği alır ve gelen verileri kullanarak bir üyenin bilgilerini günceller.

Index view'ı, üyelerin listelendiği bir tablo gösterir. Her üyenin yanında güncelle ve sil düğmeleri bulunur. Guncelle view'ı, seçilen bir üyenin bilgilerini güncellemek için bir form sunar. Ekle view'ı, yeni bir üye eklemek için bir form sunar. Bu view'lar, Bootstrap stil kullanarak tasarlanmıştır ve JavaScript koduyla etkileşimli hale getirilmiştir.

Bu kodlar, temel bir üye yönetimi sistemi sağlar.

Yazar Modeli (Yazar.cs):

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.Model

{

public class Yazar:BaseEntity

{

[Required]

[Column(TypeName = "varchar")]

[MaxLength(200)]

public string Ad { get; set; }

public virtual List<Dergi> Dergiler { get; set; }

}

}

- `Yazar` adında bir sınıf tanımlanmıştır.

- `BaseEntity` sınıfını miras almaktadır.

- `Ad` adında bir string özelliği bulunmaktadır.

- `Dergiler` adında bir `List<Dergi>` özelliği bulunmaktadır.

Yazar Kontrolcüsü (YazarController.cs):

using Dergi\_Abone\_Takip\_ASP.NET.Data.Model;

using Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Dergi\_Abone\_Takip\_ASP.NET.Controllers

{

public class YazarController : Controller

{

UnitOfWork unitOfWork;

public YazarController()

{

unitOfWork = new UnitOfWork();

}

// GET: Yazar

public ActionResult Index()

{

var yazarlar = unitOfWork.GetRepository<Yazar>().GetAll();

return View(yazarlar);

}

[HttpPost]

public JsonResult EkleJson(string yzrAd)

{

Yazar yazar = new Yazar();

yazar.Ad = yzrAd;

var eklenenYazar = unitOfWork.GetRepository<Yazar>().Add(yazar);

unitOfWork.SaveChanges();

return Json(

new

{

Result = new

{

Id = eklenenYazar.Id,

Ad = eklenenYazar.Ad,

},

JsonRequestBehavior.AllowGet

}

);

}

[HttpPost]

public JsonResult GuncelleJson(int yzrId, string yzrAd)

{

var yazar = unitOfWork.GetRepository<Yazar>().GetById(yzrId);

yazar.Ad = yzrAd;

var durum = unitOfWork.SaveChanges();

if (durum > 0)

{

return Json("1");

}

return Json("0");

}

[HttpPost]

public JsonResult SilJson(int yazarId)

{

var yazar = unitOfWork.GetRepository<Yazar>().GetById(yazarId);

if (yazar != null)

{

unitOfWork.GetRepository<Yazar>().Delete(yazar);

unitOfWork.SaveChanges();

return Json("1");

}

return Json("0");

}

}

}

- `YazarController` adında bir sınıf tanımlanmıştır.

- `UnitOfWork` sınıfının bir örneği alınmıştır.

- `Index` adında bir eylem (action) bulunmaktadır. Bu eylem, tüm yazarları veritabanından alır ve bu verileri bir görünüm (view) ile döndürür.

- `EkleJson` adında bir eylem (action) bulunmaktadır. Bu eylem, bir yazarın JSON formatında eklenmesini sağlar.

- `GuncelleJson` adında bir eylem (action) bulunmaktadır. Bu eylem, bir yazarın JSON formatında güncellenmesini sağlar.

- `SilJson` adında bir eylem (action) bulunmaktadır. Bu eylem, bir yazarın JSON formatında silinmesini sağlar.

Yazar Index Görünümü (Index.cshtml):

@{

ViewBag.Title = "YAYIN EVLERİ";

}

@using Dergi\_Abone\_Takip\_ASP.NET.Data.Model

@model List<Yazar>

<h2>YAYIN EVLERİ</h2>

<button class="btn btn-primary yazarEkle">Ekle</button>

<table id="example" class="display" style="width:100%">

<thead>

<tr>

<th>Ad</th>

<th>Dergi Adeti</th>

<th>Güncelle</th>

<th>Sil</th>

</tr>

</thead>

<tbody>

@foreach (var yzr in Model)

{

<tr>

<td>@yzr.Ad</td>

<td>@yzr.Dergiler.Count()</td>

<td><button class="yazarGuncelle btn btn-custom" data-id="@yzr.Id">Güncelle</button></td>

<td><button class="yazarSil btn btn-danger" value="@yzr.Id">Sil</button></td>

</tr>

}

</tbody>

<tfoot>

<tr>

<th>Ad</th>

<th>Dergi Adeti</th>

<th>Güncelle</th>

<th>Sil</th>

</tr>

</tfoot>

</table>

- Bu görünüm, tüm yazarları listelemektedir.

- `yazarEkle` adında bir düğme, yeni bir yazar eklemek için kullanılır.

- Bir tablo oluşturulmuştur ve tablonun başlıkları tanımlanmıştır.

- Bir döngü kullanılarak tüm yazarlar tabloya eklenir.

- Her bir yazar için güncelleme ve silme düğmeleri bulunmaktadır.

Yukarıdaki kodlar, bir abonelik takip sistemi için kullanılan Uye ve Yazar modellerini, kontrolcülerini ve görünümlerini temsil etmektedir. Bu kodlar, aboneleri ve yazarları listelemek, ekleme, güncelleme ve silme gibi temel işlemleri gerçekleştirmek için kullanılabilir.

Yukarıdaki kodlar, `Kategori` modeli ve `KategoriController` sınıfını içermektedir.

`Kategori` modeli aşağıdaki özelliklere sahiptir:

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.Model

{

public class Kategori: BaseEntity

{

[Required]

[Column(TypeName ="varchar")]

[MaxLength(50)]

public string Ad { get; set; }

public virtual List<Dergi> Dergiler { get; set; }

}

}

Bu model, `BaseEntity` sınıfından türetilmiştir ve bir kategori öğesini temsil etmektedir. `Ad` adında bir string özelliğe sahiptir ve `Dergiler` adında bir `Dergi` listesini içermektedir.

`KategoriController` sınıfı, `Controller` sınıfından türetilmiştir ve aşağıdaki eylemleri içermektedir:

- `Index`: Kategorilerin listelendiği Index sayfasını döndüren eylem.

- `Ekle`: Kategori eklemek için kullanılan sayfayı döndüren eylem.

- `EkleJson`: Ajax isteği ile çağrılan ve yeni bir kategori eklemek için kullanılan eylem.

- `GuncelleJson`: Ajax isteği ile çağrılan ve bir kategoriyi güncellemek için kullanılan eylem.

- `SilJson`: Ajax isteği ile çağrılan ve bir kategoriyi silmek için kullanılan eylem.

Index view'ı aşağıdaki gibi görünmektedir:

@{

ViewBag.Title = "Kategoriler";

}

@using Dergi\_Abone\_Takip\_ASP.NET.Data.Model

@model List<Kategori>

<h2>KATEGORİLER</h2>

<button class="btn btn-primary ktgEkle">Ekle</button>

<table id="example" class="display" style="width:100%">

<thead>

<tr>

<th>Ad</th>

<th>Dergi Adeti</th>

<th>Güncelle</th>

<th>Sil</th>

</tr>

</thead>

<tbody>

@foreach (var ktg in Model)

{

<tr>

<td>@ktg.Ad</td>

<td>@ktg.Dergiler.Count()</td>

<td><button class="guncelle btn btn-custom" value="@ktg.Id">Güncelle</button></td>

<td><button class="sil btn btn-danger" value="@ktg.Id">Sil</button></td>

</tr>

}

</tbody>

<tfoot>

<tr>

<th>Ad</th>

<th>Dergi Adeti</th>

<th>Güncelle</th>

<th>Sil</th>

</tr>

</tfoot>

</table>

Bu görünüm, `Kategori` modelinin listesini alır ve tablo şeklinde görüntüler. Her bir

kategori için `Ad` ve `Dergi Adeti` sütunlarına sahip bir satır oluşturulur. Ayrıca, her satırda güncelleme ve silme işlemlerini gerçekleştirmek için ilgili düğmeler de bulunur.

Bu şekilde, `Kategori` modeline ait verileri görüntüleyebilir, ekleyebilir, güncelleyebilir ve silebilirsiniz.

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.Model

{

public class Dergi:BaseEntity

{

[Required]

[Column(TypeName = "varchar")]

[MaxLength(50)]

public string Ad { get; set; }

[Required]

[Column(TypeName = "varchar")]

[MaxLength(500)]

public string SiraNo { get; set; }

[Required]

public int Adet { get; set; }

[Required]

public DateTime EklenmeTarihi { get; set; } = DateTime.Now;

[Required]

public int YazarId { get; set; }

public virtual Yazar Yazar { get; set; }

public virtual List<Kategori> Kategoriler { get; set; }

}

}

yukarıdaki kodlar Dergi model kodları

using Dergi\_Abone\_Takip\_ASP.NET.Data.Model;

using Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork;

using System;

using System.Collections.Generic;

using System.Web.Mvc;

namespace Dergi\_Abone\_Takip\_ASP.NET.Controllers

{

public class DergiController : Controller

{

UnitOfWork unitOfWork;

public DergiController()

{

unitOfWork = new UnitOfWork();

}

// GET: Dergi

public ActionResult Index()

{

var dergiler = unitOfWork.GetRepository<Dergi>().GetAll();

return View(dergiler);

}

public ActionResult Ekle()

{

ViewBag.Kategoriler = unitOfWork.GetRepository<Kategori>().GetAll();

ViewBag.Yazarlar = unitOfWork.GetRepository<Yazar>().GetAll();

return View();

}

[HttpPost]

public JsonResult EkleJson(string[] kategoriler, string yazar, string dergiAd, string dergiAdet, string siraNo)

{

if (kategoriler != null && !string.IsNullOrEmpty(yazar) && !string.IsNullOrEmpty(dergiAd) && !string.IsNullOrEmpty(dergiAdet) && !string.IsNullOrEmpty(siraNo))

{

List<Kategori> k = new List<Kategori>();

foreach (var kategoriId in kategoriler)

{

var kategoriID = Convert.ToInt32(kategoriId);

var kategori = unitOfWork.GetRepository<Kategori>().GetById(kategoriID);

k.Add(kategori);

}

Dergi dergi = new Dergi();

dergi.Ad = dergiAd;

dergi.Adet = Convert.ToInt32(dergiAdet);

dergi.YazarId = Convert.ToInt32(yazar);

dergi.SiraNo = siraNo;

dergi.Kategoriler = k;

dergi.EklenmeTarihi = DateTime.Now;

unitOfWork.GetRepository<Dergi>().Add(dergi);

var durum = unitOfWork.SaveChanges();

if (durum > 0)

{

return Json("1");

}

else

{

return Json("0");

}

}

else

{

return Json("bosOlamaz");

}

}

[HttpPost]

public JsonResult SilJson(int dergiId)

{

Dergi dergi = unitOfWork.GetRepository<Dergi>().GetById(dergiId);

if (dergi != null)

{

unitOfWork.GetRepository<Dergi>().Delete(dergi);

var durum = unitOfWork.SaveChanges();

if (durum > 0)

{

return Json("1");

}

else

{

return Json("0");

}

}

else

{

return Json("dergiBulunamadi");

}

}

public ActionResult Guncelle(int dergiId)

{

ViewBag.Kategoriler = unitOfWork.GetRepository<Kategori>().GetAll();

ViewBag.Yazarlar = unitOfWork.GetRepository<Yazar>().GetAll();

var dergi = unitOfWork.GetRepository<Dergi>().GetById(dergiId);

return View(dergi);

}

[HttpPost]

public JsonResult GuncelleJson(int dergiId, string[] kategoriler, string yazar, string dergiAd, string dergiAdet, string siraNo)

{

if (kategoriler != null && !string.IsNullOrEmpty(yazar) && !string.IsNullOrEmpty(dergiAd) && !string.IsNullOrEmpty(dergiAdet) && !string.IsNullOrEmpty(siraNo))

{

var dergi = unitOfWork.GetRepository<Dergi>().GetById(dergiId);

if (dergi != null)

{

dergi.Ad = dergiAd;

dergi.Adet = Convert.ToInt32(dergiAdet);

dergi.YazarId = Convert.ToInt32(yazar);

dergi.SiraNo = siraNo;

dergi.Kategoriler.Clear();

foreach (var kategoriId in kategoriler)

{

var kategoriID = Convert.ToInt32(kategoriId);

var kategori = unitOfWork.GetRepository<Kategori>().GetById(kategoriID);

dergi.Kategoriler.Add(kategori);

}

unitOfWork.GetRepository<Dergi>().Update(dergi);

var durum = unitOfWork.SaveChanges();

if (durum > 0)

{

return Json("1");

}

else

{

return Json("0");

}

}

else

{

return Json("dergiBulunamadi");

}

}

else

{

return Json("bosOlamaz");

}

}

}

}

yukarıdaki kodlar Dergi kontrolcüsü

@{

ViewBag.Title = "Dergiler";

Layout = Layout;

}

<h2>DERGİLER</h2>

<table id="example" class="display" style="width:100%">

<thead>

<tr>

<th>Sıra No</th>

<th>Kategori</th>

<th>Dergi Adı</th>

<th>Adeti</th>

<th>Yayın</th>

<th>Güncelle</th>

<th>Sil</th>

</tr>

</thead>

<tbody>

@foreach (var dergi in Model)

{

<tr>

<td>@dergi.SiraNo</td>

<td>

@foreach (var kategori in dergi.Kategoriler)

{

<text>@kategori.Ad,</text>

}

</td>

<td>@dergi.Ad</td>

<td>@dergi.Adet</td>

<td>@dergi.Yazar.Ad</td>

<td><a href="@Url.Action("Guncelle", "Dergi", new {dergiId = dergi.Id})" class="btn btn-custom">Güncelle</a></td>

<td><button class="dergiSil btn btn-danger" value="@dergi.Id">Sil</button></td>

</tr>

}

</tbody>

<tfoot>

<tr>

<th>Sıra No</th>

<th>Kategori</th>

<th>Dergi Adı</th>

<th>Adeti</th>

<th>Yayın</th>

<th>Güncelle</th>

<th>Sil</th>

</tr>

</tfoot>

</table>

yukarıdaki kodlar Imdex Views kodları

@{

ViewBag.Title = "Dergi Ekle";

}

<h2>DERGİ EKLE</h2>

<form class="form-horizontal">

<div class="form-group">

<label class="col-sm-12">Kategori Ekle</label>

<div class="col-sm-10">

<select id="kategoriler" class="form-control">

@foreach (var kategori in ViewBag.Kategoriler)

{

<option data-id="@kategori.Id">@kategori.Ad</option>

}

</select>

</div>

<div class="col-sm-2">

<input type="button" id="kategoriEkle" class="btn btn-block btn-info" value="Ekle">

</div>

</div>

<div id="eklenenKategoriler" class="form-group">

<label class="col-md-12">Kategoriler</label>

</div>

<div class="form-group">

<label class="col-sm-12">Yayın Ekle</label>

<div class="col-sm-12">

<select id="yazar" class="form-control">

@foreach (var yazar in ViewBag.Yazarlar)

{

<option data-id="@yazar.Id">@yazar.Ad</option>

}

</select>

</div>

</div>

<div class="form-group">

<label class="col-md-12">Dergi Adı</label>

<div class="col-md-12">

<input id="dergiAd" type="text" class="form-control" placeholder="Dergi adını giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Dergi Adeti</label>

<div class="col-md-12">

<input id="dergiAdet" type="text" class="form-control" placeholder="Dergi adetini giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Sıra No</label>

<div class="col-md-12">

<input id="siraNo" type="text" class="form-control" placeholder="Sırayı giriniz.">

</div>

</div>

<div class="form-group">

<div class="col-md-12">

<input type="button" id="dergiKaydet" class="btn btn-block btn-info" value="Kaydet" />

</div>

</div>

</form>

yukarıdaki kodlar Ekle Views kodları

@{

ViewBag.Title = "Dergi Güncelle";

}

@using Dergi\_Abone\_Takip\_ASP.NET.Data.Model;

@using Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork;

@model Dergi

<h2>DERGİ GÜNCELLE</h2>

<form class="form-horizontal">

<div class="form-group">

<label class="col-sm-12">Kategori Ekle</label>

<div class="col-sm-10">

<select id="kategoriler" class="form-control">

@foreach (var kategori in ViewBag.Kategoriler)

{

bool kategoriEklendi = false;

foreach (var eklenmisKategori in Model.Kategoriler)

{

if (kategori.Id == eklenmisKategori.Id)

{

kategoriEklendi = true;

break;

}

}

if (!kategoriEklendi)

{

<option data-id="@kategori.Id">@kategori.Ad</option>

}

}

</select>

</div>

<div class="col-sm-2">

<input type="button" id="kategoriEkle" class="btn btn-block btn-info" value="Ekle">

</div>

</div>

<div id="eklenenKategoriler" class="form-group">

<label class="col-md-12">Kategoriler</label>

@foreach (var kategori in Model.Kategoriler)

{

<div id="@kategori.Id" class="col-md-1 bg-primary kategoriSil" style="margin-right:2px;margin-bottom:2px">@kategori.Ad</div>

}

</div>

<div class="form-group">

<label class="col-sm-12">Yayın Ekle</label>

<div class="col-sm-12">

<select id="yazar" class="form-control">

@foreach (var yazar in ViewBag.Yazarlar)

{

if (Model.Yazar.Id == yazar.Id)

{

<option data-id="@yazar.Id" selected>@yazar.Ad</option>

}

else

{

<option data-id="@yazar.Id">@yazar.Ad</option>

}

}

</select>

</div>

</div>

<div class="form-group">

<label class="col-md-12">Dergi Adı</label>

<div class="col-md-12">

<input id="dergiAd" value="@Model.Ad" type="text" class="form-control" placeholder="Dergi adını giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Dergi Adeti</label>

<div class="col-md-12">

<input id="dergiAdet" value="@Model.Adet" type="text" class="form-control" placeholder="Dergi adetini giriniz.">

</div>

</div>

<div class="form-group">

<label class="col-md-12">Sıra No</label>

<div class="col-md-12">

<input id="siraNo" value="@Model.SiraNo" type="text" class="form-control" placeholder="Sırayı giriniz.">

</div>

</div>

<div class="form-group">

<div class="col-md-12">

<input type="button" id="dergiGuncelle" data-id="@Model.Id" class="btn btn-block btn-info" value="Güncelle" />

</div>

</div>

</form>

yukarıdaki kodlar Guncelle Views kodları

"Dergi" modeli, "Dergi\_Abone\_Takip\_ASP.NET.Data.Model" adında bir namespace içerisinde yer almaktadır. Bu model, "BaseEntity" adında bir temel sınıftan miras alır ve aşağıdaki özellikleri içerir:

- "Ad" adında bir dergi adı (string tipinde)

- "SiraNo" adında bir sıra numarası (string tipinde)

- "Adet" adında bir adet sayısı (int tipinde)

- "EklenmeTarihi" adında bir eklenme tarihi (DateTime tipinde)

- "YazarId" adında bir yazar kimliği (int tipinde)

- "Yazar" adında bir Yazar nesnesi (Yazar modeline ait ilişkili bir özellik)

- "Kategoriler" adında bir Kategori listesi (Kategori modeline ait ilişkili bir özellik)

"DergiController" ise "Dergi" modeline ait işlemleri gerçekleştiren kontrolcü sınıfıdır. Bu sınıf, "Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork" adlı bir Unit of Work sınıfını kullanır.

İşlevleri şunlardır:

- "Index" işlevi, tüm dergileri getirir ve bunları bir görünüme (view) aktarır.

- "Ekle" işlevi, yeni bir dergi eklemek için gerekli verileri görünüme (view) aktarır.

- "EkleJson" işlevi, AJAX isteğiyle dergi ekleme işlemini gerçekleştirir.

- "SilJson" işlevi, AJAX isteğiyle belirtilen dergiyi siler.

- "Guncelle" işlevi, belirtilen bir dergiyi güncellemek için gerekli verileri görünüme (view) aktarır.

- "GuncelleJson" işlevi, AJAX isteğiyle belirtilen bir dergiyi günceller.

Ayrıca, "Index.cshtml" ve "Ekle.cshtml" adlı görünüm dosyaları, dergi verilerini göstermek ve dergi eklemek için kullanıcı arayüzünü oluşturmak için kullanılır.

Veri tabanı işlemlerini yönetmek için kullanılan bazı sınıfları içeriyor. İşte bu sınıfların detaylı açıklamaları:

using System.ComponentModel.DataAnnotations;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.Model

{

public class BaseEntity

{

[Key]

public int Id { get; set; }

}

}

yukarıdaki kodlar Base Entity modeli

using Dergi\_Abone\_Takip\_ASP.NET.Data.Migrations;

using Dergi\_Abone\_Takip\_ASP.NET.Data.Model;

using System.Data.Entity;

using System.Data.Entity.ModelConfiguration.Conventions;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data

{

public class Context:DbContext

{

public Context() : base("Context")

{

Database.SetInitializer(new MigrateDatabaseToLatestVersion<Context,Configuration>("Context"));//Migrataino database bağlayıp dahil ettik.

}

public DbSet<Kategori> Kategoriler { get; set; }

public DbSet<Dergi> Dergiler { get; set; }

public DbSet<AlinanDergi> AlinanDergiler { get; set; }

public DbSet<Uye> Uyeler { get; set; }

public DbSet<Yazar> Yazarlar { get; set; }

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

//Mssql de otomatik s takısı ekliyor bunu kaldırmanın yolu

modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

base.OnModelCreating(modelBuilder);

}

}

}

yukarıdaki kodlar Context kodları

using Dergi\_Abone\_Takip\_ASP.NET.Data.Repositories;

using System;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork

{

public class UnitOfWork : IUnitOfWork

{

private readonly Context \_context;

public UnitOfWork()

{

\_context = new Context();

}

public IRepository<T> GetRepository<T>() where T : class

{

return new Repository<T>(\_context);

}

public int SaveChanges()

{

return \_context.SaveChanges();

}

private bool disposed = false;

protected void Dispose(bool disposing)

{

if (!this.disposed)

{

if (disposed) { \_context.Dispose(); }

}

this.disposed = true;

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

}

}

yukarıdaki kodlar UnitOfWork kodları

using Dergi\_Abone\_Takip\_ASP.NET.Data.Repositories;

using System;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork

{

public interface IUnitOfWork : IDisposable

{

IRepository<T> GetRepository<T>() where T : class;

int SaveChanges();

}

}

yukarıdaki kodlar IUnitOfWork kodları

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Linq.Expressions;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.Repositories

{

public class Repository<T> : IRepository<T> where T : class

{

protected readonly Context \_context;

protected readonly DbSet<T> \_dbSet;

public Repository(Context context)

{

\_context = context;

\_dbSet = \_context.Set<T>();

}

public List<T> GetAll()

{

return \_dbSet.ToList();

}

public List<T> GetAll(Expression<Func<T, bool>> predicate)

{

return \_dbSet.Where(predicate).ToList();

}

public T GetById(int id)

{

return \_dbSet.Find(id);

}

public T Get(Expression<Func<T, bool>> predicate)

{

return \_dbSet.Where(predicate).SingleOrDefault();

}

public T Add(T entity)

{

return \_dbSet.Add(entity);

}

public T Update(T entity)

{

\_dbSet.Attach(entity);

\_context.Entry(entity).State = EntityState.Modified;

return entity;

}

public T Delete(T entity)

{

return \_dbSet.Remove(entity);

}

public void Delete(int id)

{

var entity = GetById(id);

if (entity != null) return;

Delete(entity);

}

}

}

yukarıdaki kodlar Repository kodları

using System;

using System.Collections.Generic;

using System.Linq;

using System.Linq.Expressions;

using System.Runtime.InteropServices;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.Repositories

{

public interface IRepository<T> where T:class

{

List<T> GetAll();

List<T> GetAll(Expression<Func<T, bool>> predicate);

T GetById(int id);

T Get(Expression<Func<T, bool>> predicate);

T Add(T entity);

T Update(T entity);

T Delete(T entity);

void Delete(int id);

}

}

yukarıdaki kodlar IRepository kodları

1. BaseEntity: Diğer model sınıflarının temelini oluşturan bir temel sınıftır. İçinde sadece bir "Id" özelliği bulunur ve bu özellik, veritabanı tablosunda birincil anahtar (primary key) olarak kullanılır.

2. Context: Veritabanı bağlantısını ve tabloları temsil eden DbSet nesnelerini içeren bir DbContext sınıfıdır. Veritabanı bağlantısı ve tablo tanımları bu sınıfta yapılandırılır. Ayrıca, migrasyon işlemlerini yürütmek için bir yapılandırıcı (initializer) ayarlanır.

3. UnitOfWork: Veritabanı işlemlerinin gerçekleştirildiği birim işlem sınıfıdır. Context sınıfını kullanarak Repository sınıflarına erişim sağlar ve değişiklikleri kaydetmek için SaveChanges yöntemini kullanır.

4. IRepository: Genel bir arabirim (interface) tanımlayan bir kod parçasıdır. Repository sınıfları tarafından uygulanır ve temel veritabanı işlemlerini (ekleme, güncelleme, silme vb.) gerçekleştirmek için yöntemler sağlar.

5. Repository: Veritabanı işlemlerini gerçekleştiren bir Repository sınıfıdır. Context sınıfına erişim sağlar ve LINQ ifadeleri kullanarak veritabanında sorgular yapar. GetAll, GetById, Add, Update, Delete vb. gibi yöntemler içerir.

Bu kodlar, ASP.NET projelerinde genellikle Entity Framework kullanılarak veritabanı işlemlerini kolaylaştırmak amacıyla kullanılır. Veritabanı bağlantısı ve CRUD (Create, Read, Update, Delete) işlemleri bu yapılar üzerinden gerçekleştirilir.

Uygulama, alınan dergi aboneliklerini yönetmek için kullanılan bir veritabanı modeli ve kontrolcüleri içermektedir.

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Dergi\_Abone\_Takip\_ASP.NET.Data.Model

{

public class AlinanDergi:BaseEntity

{

[Required]

public int DergiId { get; set; }

[Required]

public int UyeId { get; set; }

[Required]

public DateTime AboneBaslangic { get; set;}

[Required]

public DateTime AboneBitis { get; set; }

public DateTime? OdemeTarihi { get; set; }

public virtual Uye Uye { get; set; }

public virtual Dergi Dergi { get; set; }

}

}

yukarıdaki kodlar AlinanDergi Model kodları

using Dergi\_Abone\_Takip\_ASP.NET.Data.Model;

using Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Dergi\_Abone\_Takip\_ASP.NET.Controllers

{

public class AlinanDergiController : Controller

{

private readonly UnitOfWork unitOfWork;

public AlinanDergiController()

{

unitOfWork = new UnitOfWork();

}

// GET: AlinanDergi/AboneSuresi

public ActionResult AboneSuresi()

{

var alinanDergi = unitOfWork.GetRepository<AlinanDergi>().GetAll(x => x.OdemeTarihi == null);

return View(alinanDergi);

}

public ActionResult Odendi()

{

var alinanDergi = unitOfWork.GetRepository<AlinanDergi>().GetAll(x=>x.OdemeTarihi != null);

return View(alinanDergi);

}

public ActionResult OdemeVer()

{

ViewBag.Dergiler = unitOfWork.GetRepository<Dergi>().GetAll(x => x.Adet > 0);

ViewBag.Uyeler = unitOfWork.GetRepository<Uye>().GetAll();

return View();

}

[HttpPost]

public JsonResult OdemeVerJson(int uyeId, int dergiId, DateTime aboneBitisTarih)

{

AlinanDergi alinanDergi = new AlinanDergi();

alinanDergi.AboneBaslangic = DateTime.Now;

alinanDergi.AboneBitis = aboneBitisTarih;

alinanDergi.DergiId = dergiId;

alinanDergi.UyeId = uyeId;

unitOfWork.GetRepository<AlinanDergi>().Add(alinanDergi);

var durum = unitOfWork.SaveChanges();

if (durum > 0)

{

return Json(new { success = true });

}

else

{

return Json(new { success = false, message = "Ödeme işlemi gerçekleştirilemedi." });

}

}

public ActionResult OdemeDergiGuncelle(int alinanDergiId)

{

ViewBag.Dergiler = unitOfWork.GetRepository<Dergi>().GetAll(x => x.Adet > 0);

ViewBag.Uyeler = unitOfWork.GetRepository<Uye>().GetAll();

var alinanDergi = unitOfWork.GetRepository<AlinanDergi>().GetById(alinanDergiId);

return View(alinanDergi);

}

[HttpPost]

public JsonResult OdemeDergiGuncelleJson(int alinanDergiId, int uyeId, int dergiId, DateTime aboneBitisTarih)

{

var alinanDergi = unitOfWork.GetRepository<AlinanDergi>().GetById(alinanDergiId);

alinanDergi.AboneBitis = aboneBitisTarih;

alinanDergi.DergiId = dergiId;

alinanDergi.UyeId = uyeId;

unitOfWork.GetRepository<AlinanDergi>().Update(alinanDergi);

var durum = unitOfWork.SaveChanges();

if (durum > 0)

{

return Json(new { success = true });

}

else

{

return Json(new { success = false, message = "Güncelleme işlemi gerçekleştirilemedi." });

}

}

[HttpPost]

public JsonResult OdemeOnayla(int alinanDergiId)

{

var alinanDergi = unitOfWork.GetRepository<AlinanDergi>().GetById(alinanDergiId);

alinanDergi.OdemeTarihi = DateTime.Now;

unitOfWork.GetRepository<AlinanDergi>().Update(alinanDergi);

var durum = unitOfWork.SaveChanges();

if(durum > 0)

{

return Json("1");

}

else

{

return Json("0");

}

}

}

}

yukarıdaki kodlar AlinanDergi Kontrolcüsü

@{

ViewBag.Title = "Abonelik Süresi";

}

@using Dergi\_Abone\_Takip\_ASP.NET.Data.Model

@model List<AlinanDergi>

<h2 style="display:inline-block">ABONELİK SÜRESİ</h2>

<a href="@Url.Action("OdemeVer","AlinanDergi")" class="btn btn-success" style="float:right;">Abonelik Ödeme</a>

<table id="example" class="display" style="width:100%">

<thead>

<tr>

<th>Dergi Adı</th>

<th>Abone Olan Üye</th>

<th>Abone Olduğu Tarih</th>

<th>Abonelik Bitiş Tarihi</th>

<th>Güncelle</th>

<th>Ödendi Olarak İşaretle</th>

</tr>

</thead>

<tbody>

@foreach (var alinanDergi in Model)

{

<tr>

<td>@alinanDergi.Dergi.Ad</td>

<td>@alinanDergi.Uye.Ad</td>

<td>@alinanDergi.AboneBaslangic.ToString("dddd, dd MMMM yyyy")</td>

<td>@alinanDergi.AboneBitis.ToString("dddd, dd MMMM yyyy")</td>

<td><a href="@Url.Action("OdemeDergiGuncelle","AlinanDergi", new { alinanDergiId= alinanDergi.Id})" class="btn btn-success">Güncelle</a></td>

<td><button class="odendiOlarakIsaretle btn btn-danger" value="@alinanDergi.Id">Teslim Edildiği İse İşaretle</button></td>

</tr>

}

</tbody>

<tfoot>

<tr>

<th>Dergi Adı</th>

<th>Abone Olan Üye</th>

<th>Abone Olduğu Tarih</th>

<th>Abonelik Bitiş Tarihi</th>

<th>Güncelle</th>

<th>Ödendi Olarak İşaretle</th>

</tr>

</tfoot>

</table>

yukarıdaki kodlar AboneSuresi Views kodları

@model Dergi\_Abone\_Takip\_ASP.NET.Data.Model.AlinanDergi

@{

ViewBag.Title = "Ödeme Güncelleme Sayfası";

}

<h2>ÖDEME GÜNCELLEME</h2>

<form class="form-horizontal">

<div class="form-group">

<label class="col-sm-12">Aboneler</label>

<div class="col-sm-10">

<select id="uyeId" class="form-control">

@foreach (var uye in ViewBag.Uyeler)

{

if (uye.Id == Model.UyeId)

{

<option value="@uye.Id" selected>@uye.Ad</option>

}

else

{

<option value="@uye.Id">@uye.Ad</option>

}

}

</select>

</div>

</div>

<div class="form-group">

<label class="col-sm-12">Dergiler</label>

<div class="col-sm-12">

<select id="dergiId" class="form-control">

@foreach (var dergi in ViewBag.Dergiler)

{

if (dergi.Id == Model.DergiId)

{

<option value="@dergi.Id" selected>@dergi.Ad</option>

}

else

{

<option value="@dergi.Id">@dergi.Ad</option>

}

}

</select>

</div>

</div>

<div class="form-group">

<label class="col-md-12">Abonelik Bitiş Tarihi</label>

<div class="col-md-12">

<input id="aboneBitisTarih" type="date" min="@DateTime.Now.ToString("yyyy-MM-dd")" value="@Model.AboneBitis.ToString("yyyy-MM-dd")" class="form-control">

</div>

</div>

<div class="form-group">

<div class="col-md-12">

<input type="button" id="odemeVerGuncelle" data-id="@Model.Id" class="btn btn-block btn-info" value="Güncelle" />

</div>

</div>

</form>

yukarıdaki kodlar OdemeDergiGuncelle Views kodları

@{

ViewBag.Title = "Ödeme Onaylama Sayfası";

}

<h2>ÖDEME ONAYLA</h2>

<form class="form-horizontal">

<div class="form-group">

<label class="col-sm-12">Aboneler</label>

<div class="col-sm-10">

<select id="uyeId" class="form-control">

@foreach (var uye in ViewBag.Uyeler)

{

<option data-id="@uye.Id">@uye.Ad</option>

}

</select>

</div>

</div>

<div class="form-group">

<label class="col-sm-12">Dergiler</label>

<div class="col-sm-12">

<select id="dergiId" class="form-control">

@foreach (var dergi in ViewBag.Dergiler)

{

<option data-id="@dergi.Id">@dergi.Ad</option>

}

</select>

</div>

</div>

<div class="form-group">

<label class="col-md-12">Abonelik Bitiş Tarihi</label>

<div class="col-md-12">

<input id="aboneBitisTarih" type="date" min="@DateTime.Now.ToString("yyyy-MM-dd")" class="form-control">

</div>

</div>

<div class="form-group">

<div class="col-md-12">

<input type="button" id="odemeVer" class="btn btn-block btn-info" value="Kaydet" />

</div>

</div>

</form>

yukarıdaki kodlar OdemeVer Views kodları

@{

ViewBag.Title = "Ödeme Yapanlar";

DateTime odemeTarih;

}

@using Dergi\_Abone\_Takip\_ASP.NET.Data.Model

@model List<AlinanDergi>

<h2>ÖDEME YAPMIŞLAR VE TESLİM EDİLENLER</h2>

<table id="example" class="display" style="width:100%">

<thead>

<tr>

<th>Dergi Adı</th>

<th>Abone Olan Üye</th>

<th>Abone Olduğu Tarih</th>

<th>Abonelik Bitiş Tarihi</th>

<th>Ödeme Tarihi</th>

</tr>

</thead>

<tbody>

@foreach (var alinanDergi in Model)

{

odemeTarih = Convert.ToDateTime(@alinanDergi.OdemeTarihi);

<tr>

<td>@alinanDergi.Dergi.Ad</td>

<td>@alinanDergi.Uye.Ad</td>

<td>@alinanDergi.AboneBaslangic.ToString("dddd, dd MMMM yyyy")</td>

<td>@alinanDergi.AboneBitis.ToString("dddd, dd MMMM yyyy")</td>

<td> @odemeTarih.ToString("dddd, dd MMMM yyyy")</td>

</tr>

}

</tbody>

<tfoot>

<tr>

<th>Dergi Adı</th>

<th>Abone Olan Üye</th>

<th>Abone Olduğu Tarih</th>

<th>Abonelik Bitiş Tarihi</th>

<th>Ödeme Tarihi</th>

</tr>

</tfoot>

</table>

yukarıdaki kodlar Odendi Views kodları

`AlinanDergi` sınıfı, abonelik kayıtlarını temsil etmektedir. Bu sınıf, bir dergiye ait abonelik bilgilerini ve ödeme durumunu tutmaktadır. İlgili dergi ve üye bilgilerine ilişkin ilişkileri içermektedir.

`AlinanDergiController` sınıfı, abonelik işlemlerini yöneten kontrolcüdür. Bu kontrolcü, abonelik süresi, ödeme yapılanlar ve ödeme güncelleme gibi işlemleri gerçekleştiren metotları içermektedir.

"AboneSuresi.cshtml" görünümü, abonelik süresini görüntülemek için kullanılmaktadır. Bu görünüm, abone bilgilerini ve güncelleme veya ödeme işlemlerini gerçekleştirmek için ilgili bağlantıları içeren bir tabloyu içermektedir.

"OdemeDergiGuncelle.cshtml" görünümü, ödeme güncelleme işlemi için kullanılmaktadır. Bu görünüm, bir aboneliğin üye, dergi ve abonelik bitiş tarihini düzenlemek için gerekli giriş alanlarını içermektedir.

"OdemeOnayla.cshtml" görünümü, ödeme onaylama işlemi için kullanılmaktadır. Bu görünüm, bir aboneliği ödendi olarak işaretlemek için gerekli giriş alanlarını içermektedir.

"OdemeVer.cshtml" görünümü, yeni bir aboneliği ödeme yaparak kaydetmek için kullanılmaktadır. Bu görünüm, bir üye, dergi ve abonelik bitiş tarihini seçmek için giriş alanları içermektedir.

"Odendi.cshtml" görünümü, ödeme yapılan abonelikleri görüntülemek için kullanılmaktadır. Bu görünüm, ödeme tarihiyle birlikte abone bilgilerini içeren bir tabloyu içermektedir.

Quartz.NET kütüphanesini kullanarak bir görev (job) oluşturmayı ve bu görevin belirli aralıklarla çalışmasını sağlamayı amaçlamaktadır. `CezaArttirmaDusurmeJob` adında bir sınıf tanımlanmıştır ve `IJob` arayüzünü uygular.

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations.Schema;

using Dergi\_Abone\_Takip\_ASP.NET.Data.Model;

using Dergi\_Abone\_Takip\_ASP.NET.Data.UnitOfWork;

using Quartz;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Web;

namespace Dergi\_Abone\_Takip\_ASP.NET.Tasks.Jobs

{

public class CezaArttirmaDusurmeJob : IJob

{

private readonly UnitOfWork unitOfWork;

public CezaArttirmaDusurmeJob(UnitOfWork unitOfWork)

{

this.unitOfWork = unitOfWork ?? throw new ArgumentNullException(nameof(unitOfWork));

}

public void Execute(IJobExecutionContext context)

{

try

{

CezaArttir();

CezaDusur();

unitOfWork.SaveChanges();

}

catch { }

}

void CezaArttir()

{

var alinanDergiler = unitOfWork.GetRepository<AlinanDergi>().GetAll(x => x.OdemeTarihi == null && DateTime.Now > x.OdemeTarihi);

foreach (var alinanDergi in alinanDergiler)

{

alinanDergi.Uye.Ceza += 1;

unitOfWork.GetRepository<Uye>().Update(alinanDergi.Uye);

}

}

void CezaDusur()

{

var alinanDergiler = unitOfWork.GetRepository<AlinanDergi>().GetAll(x => x.OdemeTarihi != null && x.Uye.Ceza > 0);

foreach (var alinanDergi in alinanDergiler)

{

alinanDergi.Uye.Ceza -= 1;

unitOfWork.GetRepository<Uye>().Update(alinanDergi.Uye);

}

}

}

}

Bu sınıfın yapısı aşağıdaki gibi özetlenebilir:

- `UnitOfWork` adında bir bağımlılık enjeksiyonu kullanarak, veritabanı işlemlerini gerçekleştirmek için gerekli olan `unitOfWork` nesnesi tanımlanmıştır.

- `Execute` metodu, `IJob` arayüzünün bir gereksinimidir ve bu metot Quartz.NET tarafından çağrılır. Bu metot, `CezaArttir` ve `CezaDusur` metotlarını çağırır ve ardından veritabanı değişikliklerini `unitOfWork.SaveChanges()` ile kaydeder.

- `CezaArttir` metodu, henüz ödeme yapılmamış dergilerin sahiplerine bir ceza puanı ekler. Bu metot, `unitOfWork` üzerinden `AlinanDergi` ve `Uye` tablolarına erişerek işlemler gerçekleştirir.

- `CezaDusur` metodu, ödeme yapılmış ancak hala ceza puanı olan dergi sahiplerinin ceza puanını azaltır. Yine, `unitOfWork` üzerinden `AlinanDergi` ve `Uye` tablolarına erişim sağlanır.

Quartz.NET kütüphanesini kullanarak bir tetikleyici (trigger) oluşturmayı ve belirli bir zaman aralığında belirli bir görevi çalıştırmayı amaçlamaktadır. `CezaArttirmaDusurmeTrigger` adında bir sınıf tanımlanmıştır.

using Dergi\_Abone\_Takip\_ASP.NET.Tasks.Jobs;

using Quartz;

using Quartz.Impl;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

namespace Dergi\_Abone\_Takip\_ASP.NET.Tasks.Triggers

{

public class CezaArttirmaDusurmeTrigger

{

public static void Baslat()

{

IScheduler zamanlayici = StdSchedulerFactory.GetDefaultScheduler();

if(!zamanlayici.IsStarted)

{

zamanlayici.Start();

}

IJobDetail gorev = JobBuilder.Create<CezaArttirmaDusurmeJob>().Build();

ICronTrigger tetikleyici = (ICronTrigger)TriggerBuilder.Create().WithIdentity("CezaArttirmaDusurmeJob", "null").WithCronSchedule("0 0 17 \* \* ? \*").Build();

zamanlayici.ScheduleJob(gorev, tetikleyici);

}

}

}

Bu sınıfın yapısı aşağıdaki gibi özetlenebilir:

- `Baslat` adında bir statik metot tanımlanmıştır. Bu metot, belirli bir zamanda `CezaArttirmaDusurmeJob` görevini çalıştırmak için kullanılır.

- `IScheduler` arayüzünden bir zamanlayıcı (scheduler) nesnesi oluşturulur. `StdSchedulerFactory.GetDefaultScheduler()` metodu kullanılarak varsayılan bir zamanlayıcı elde edilir.

- `zamanlayici.IsStarted` kontrol edilerek, zamanlayıcının başlatılıp başlatılmadığı kontrol edilir. Başlatılmamışsa `zamanlayici.Start()` metodu çağrılarak zamanlayıcı başlatılır.

- `IJobDetail` arayüzünden bir görev (job) nesnesi (`gorev`) oluşturulur. `JobBuilder.Create<CezaArttirmaDusurmeJob>().Build()` metodu kullanılarak `CezaArttirmaDusurmeJob` sınıfına ait bir görev nesnesi oluşturulur.

- `ICronTrigger` arayüzünden bir tetikleyici (trigger) nesnesi (`tetikleyici`) oluşturulur. `TriggerBuilder.Create().WithIdentity("CezaArttirmaDusurmeJob", "null").WithCronSchedule("0 0 17 \* \* ? \*").Build()` metodu kullanılarak cron tablosu formatında bir tetikleyici oluşturulur. Bu tetikleyici, her gün saat 17:00'de görevin çalıştırılmasını sağlar.

- `zamanlayici.ScheduleJob(gorev, tetikleyici)` metoduyla oluşturulan görev ve tetikleyici zamanlayıcıya eklenir ve görev belirtilen zaman aralığında çalıştırılmaya başlar.

`CezaArttirmaDusurmeTrigger.Baslat()` metodu çağrılarak kullanılabilir. Bu metot, belirli bir zaman aralığında `CezaArttirmaDusurmeJob` görevini çalıştırmak için Quartz.NET'i kullanır.

Şifreleme ve rastgele şifre oluşturma işlemlerini gerçekleştiren bir yardımcı sınıf olan `Sifreleme` sınıfını içermektedir.

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Web;

namespace Dergi\_Abone\_Takip\_ASP.NET.HelperClass

{

public static class Sifreleme

{

private static string MD5(this string parola)

{

MD5CryptoServiceProvider md5 = new MD5CryptoServiceProvider();

byte[] dizi = md5.ComputeHash(Encoding.UTF8.GetBytes(parola));

StringBuilder sb = new StringBuilder();

for(int i = 0; i < dizi.Length; i++)

{

sb.Append(dizi[i].ToString("x2"));

}

return sb.ToString();

}

private static string SHA\_1(this string parola)

{

SHA1 sha1Hasher = SHA1.Create();

byte[] dizi = sha1Hasher.ComputeHash(Encoding.Default.GetBytes(parola));

StringBuilder sb = new StringBuilder();

for (int i = 0; i < dizi.Length; i++)

{

sb.Append(dizi[i].ToString("x2"));

}

return sb.ToString();

}

public static string Sifrele(this string parola)

{

parola = parola.SHA\_1();

parola = parola.MD5();

parola = parola.SHA\_1();

parola = parola.MD5();

return parola;

}

public static string YeniSifreOlustur(int karakterSayisi)

{

var chars = "ABCDEFGHJKLMNOPQRSTUVWXYZ0123456789";

var random = new Random();

var result = new string(Enumerable.Repeat(chars, karakterSayisi).Select(s => s[random.Next(s.Length)]).ToArray());

return result;

}

}

}

Bu sınıfın yapısı aşağıdaki gibi özetlenebilir:

- `MD5` metodu, verilen bir parolayı MD5 algoritmasıyla şifreler. MD5CryptoServiceProvider sınıfı kullanılarak parola önce byte dizisine dönüştürülür, ardından bu dizi MD5 algoritmasıyla işlenir ve sonuç bir StringBuilder kullanılarak hex formatta bir dizeye dönüştürülür.

- `SHA\_1` metodu, verilen bir parolayı SHA-1 algoritmasıyla şifreler. SHA1 sınıfı kullanılarak parola önce byte dizisine dönüştürülür, ardından bu dizi SHA-1 algoritmasıyla işlenir ve sonuç bir StringBuilder kullanılarak hex formatta bir dizeye dönüştürülür.

- `Sifrele` metodu, verilen bir parolayı şifrelemek için önce `SHA\_1` metoduyla şifreler, ardından bu sonucu `MD5` metoduyla şifreler. İşlemi iki kez tekrarlayarak şifreleme güvenliğini artırır.

- `YeniSifreOlustur` metodu, belirli bir karakter sayısıyla rastgele bir şifre oluşturur. Belirtilen karakterlerden oluşan bir karakter dizisi (`chars`) ve bir rastgele sayı üreticisi (`random`) kullanılarak, karakter sayısı kadar rastgele karakter seçilir ve sonuç bir dize olarak döndürülür.

Bu `Sifreleme` sınıfı, şifreleri şifrelemek ve rastgele şifreler oluşturmak için kullanılabilir. Ancak, MD5 ve SHA-1 algoritmalarının günümüzde güvenlik açıklarına sahip olduğu ve daha güçlü şifreleme yöntemlerinin tercih edilmesi gerektiği unutulmamalıdır.

# 

# Sonuç ve Öneriler

Kategori Modeli ve Kontrolcüsü:

Kategori modeli, "Ad" adında bir özellik içermekte ve "Dergiler" adında bir Dergi listesine sahiptir.

Kategori kontrolcüsü, Index, Ekle, EkleJson, GuncelleJson ve SilJson gibi işlemleri gerçekleştiren metotları içermektedir.

Index görünümü, tüm kategorileri listelemektedir.

Dergi Modeli ve Kontrolcüsü:

Dergi modeli, "Ad", "SiraNo", "Adet", "EklenmeTarihi", "YazarId" ve "Kategoriler" gibi özelliklere sahiptir.

Dergi kontrolcüsü, Index, Ekle, EkleJson, GuncelleJson ve SilJson gibi işlemleri gerçekleştiren metotları içermektedir.

Index görünümü, tüm dergileri listelemektedir. Her dergi için güncelleme ve silme işlemleri gerçekleştirilebilir.

BaseEntity:

BaseEntity, diğer model sınıflarının temel sınıfıdır ve "Id" adında bir özelliğe sahiptir.

Context:

Context sınıfı, Entity Framework kullanarak veritabanı bağlantısını yönetmektedir. Kategoriler, Dergiler, AlinanDergiler, Uyeler ve Yazarlar gibi DbSet özelliklerine sahiptir.

Migration yapılandırması ile veritabanı güncelleme işlemleri otomatik olarak gerçekleştirilmektedir.

UnitOfWork ve IRepository:

UnitOfWork, veritabanı işlemlerini yöneten bir sınıftır. IRepository'yi kullanarak her bir model sınıfı için bir repository elde eder ve SaveChanges metoduyla veritabanına değişiklikleri kaydeder.

AlinanDergi Modeli ve Kontrolcüsü:

AlinanDergi modeli, "DergiId", "UyeId", "AboneBaslangic", "AboneBitis", "OdemeTarihi" gibi özelliklere sahiptir. Ayrıca Uye ve Dergi sınıflarına ilişki içermektedir.

AlinanDergi kontrolcüsü, AboneSuresi, Odendi, OdemeVer, OdemeVerJson, OdemeDergiGuncelle, OdemeDergiGuncelleJson ve OdemeOnayla gibi işlemleri gerçekleştiren metotları içermektedir.

AboneSuresi ve Odendi görünümleri, abonelik kayıtlarını listelemektedir. OdemeVer görünümü, yeni bir abonelik kaydı oluşturmak için kullanılmaktadır. OdemeDergiGuncelle görünümü, belirli bir abonelik kaydını güncellemek için kullanılmaktadır.

Öneriler:

Kodlar genel olarak iyi bir yapıya sahip gibi görünmektedir. Model sınıfları doğru ilişkilendirilmiş ve gerekli doğrulama özellikleri eklenmiştir.

Ancak, bazı metot ve değişken isimlerinde Türkçe karakterler kullanılmıştır. Genel olarak, İngilizce isimlendirme standartlarına uygun isimlendirme tercih edilmelidir.

# 

# Kaynaklar

https://dotnet.microsoft.com/apps/aspnet/mvc

https://www.yazilimkodlama.com/asp-net-mvc-tutorial/

https://docs.microsoft.com/en-us/ef/

https://docs.microsoft.com/en-us/dotnet/csharp/

https://getbootstrap.com

https://sweetalert2.github.io/

https://www.jqueryscript.net/table/

# Kişisel Yayın ve Eserler

# Şuan da herhangi bir yayınım veya eserim yoktur.

# 

# Özgeçmiş

2001 yılında İstanbul’da doğdu. İlk, orta ve lise öğrenimini İstanbul’da tamamladı.

2021 yılında Kocaeli Üniversitesi Meslek Yüksekokulu Bilgisayar Programcılığı

Bölümü’ne girdi.

# Ekler



