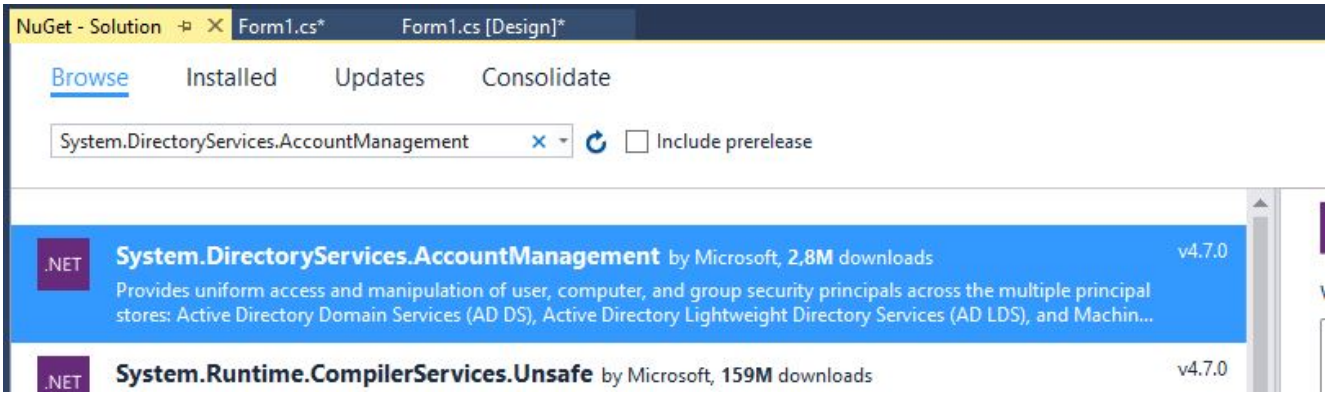
Sistem Yöneticileri İçin C# Kodları

Bilgisayar biliminin yaygın alanlarından ikisi sistem yönetimi ve yazılım alanlarıdır. Her ne kadar bu iki alanda çalışan arkadaşlar bir birine karışmamaya çalışsa da tecrübelerimiz bize farklı şeyler söylüyor. Tecrübelerimiz, her sistem yöneticisi arkadaşın az da olsa yazılım, her yazılımcı arkadaşın da az da olsa sistem yönetimi konusunda tecrübe edinmesi gerektiğini söylüyor. Bu yazımızda sistem yönetimi için kullanılan bazı kodları paylaşıyor olacağız.

Active Directory Kullanıcı ve Grup İşlemleri

AD işlemleri için .Net Framework 4.5 ile gelen System.DirectoryServices.AccountManagement isim uzayında (Namespace) yer alan sınıfları kullanacağız. İlgili paketi Nuget'ten projemize ekledikten sonra işlemlere başlayabiliriz. Peket linki: https://www.nuget.org/packages/System.DirectoryServices.AccountManagement/



Active Directory işlemlerinin tamamında sunucu ile bağlantıyı sağlayan PrincipalContext sınıfından nesne türetilmelidir. PrincipalContext sınıfının yedi farklı oluşturucu metodu olsa da en sık kullanılan üç metodu şunlardır;

- PrincipalContext(ContextType): Sadece bağlanacak sunucun türünün alındığı oluşturucu metottur. ContextType enum değerinin alabileceği değerler:
  * ApplicationDirectory: Uygulama dizin sunucusu
  * Domain: Domain controllerdan verileri okumak için okunur.
  * Machine: Local makine SAM veri tabanına bağlantıda kullanılır.
- PrincipalContext(ContextType, String): İlkinin yanında ikinci parametre olarak bağlanacak domain controller adını alır. cozumpark.local gibi…
- PrincipalContext(ContextType, String, String, String): İlk iki metodun parametrelerinin yanında üçüncü parametre olarak bağlantıda kullanılacak kullanıcı adı ve dördüncü parametre olarak da girilen kullanıcı adının şifresini alır.

AD Kullanıcılarının Sistemden Alınması

AD kullanıcılarının sistemden alınabilmesi için oluşturulan PrincipalContext nesnesini parametre alan bir UserPrincipal nesnesi oluşturulur. Daha sonra bu nesne üzerinden istenen filtre değerleri girilir. Örneğin aşağıdaki kodda sadece aktif kullanıcıları istediğimizden “userPrincipal.Enabled = true;” ataması ile bunun filtreye ekliyoruz. Daha sonra oluşturulan bu UserPrincipal nesnesini parametre alan bir PrincipalSearcher nesnesi oluşturulur. Son adım olarak PrincipalSearcher nesnesinin FindAll metodu ile UserPrincipal üzerinden girilen değerlere uygun kullanıcılar sistemden

alınır. Metotdan dönen değerler PrincipalSearchResult türünde olduğundan “Cast<UserPrincipal>()” metodu ile dönen sonuç UserPrincipal türüne çevrilir.

```
PrincipalContext principalContext = new PrincipalContext(ContextType.Domain, "saitorhan.local", "Administrator", "123456qaZ.");
UserPrincipal userPrincipal = new UserPrincipal(principalContext);
userPrincipal.Enabled = true;
PrincipalSearcher principalSearcher = new PrincipalSearcher(userPrincipal);
List<UserPrincipal> principalSearchResult = principalSearcher.FindAll().Cast<UserPrincipal>().OrderBy(u => u.SamAccountName).ToList();
```

[1]	Name ("Sait ORHAN")	System.DirectoryServices.AccountManageme
AccountExpirationDate	null	System.DateTime?
AccountLockoutTime	null	System.DateTime?
AdvancedSearchFilter	{System.DirectoryServices.AccountManagement.AdvancedFilters}	System.DirectoryServices.AccountManageme
AllowReversiblePasswordEncryption	false	bool
BadLogonCount	0	int
Certificates	{System.Security.Cryptography.X509Certificates.X509Certificate2Collectio	System.Security.Cryptography.X509Certificat
Context	{System.DirectoryServices.AccountManagement.PrincipalContext}	System.DirectoryServices.AccountManageme
ContextRaw	{System.DirectoryServices.AccountManagement.PrincipalContext}	System.DirectoryServices.AccountManageme
ContextType	Domain	System.DirectoryServices.AccountManageme
DelegationPermitted	true	bool
Description	null	string
DisplayName	"Sait ORHAN"	string
DistinguishedName	"CN=Sait ORHAN,OU=BilgiTeknolojileri,OU=Users,OU=[redacted]	string
EmailAddress	"saitorhan@[redacted]"	string
EmployeeId	null	string
Enabled	true	bool?
GivenName	"Sait"	string
Guid	{ac788b01-d395-4208-8e00-1bedcfa0db2d}	System.Guid?
HomeDirectory	null	string
HomeDrive	null	string
LastBadPasswordAttempt	{5.02.2020 09:13:02}	System.DateTime?
LastLogon	{5.02.2020 00:17:12}	System.DateTime?
LastPasswordSet	{20.01.2020 06:38:33}	System.DateTime?
MiddleName	null	string
Name	"Sait ORHAN"	string
PasswordNeverExpires	true	bool
PasswordNotRequired	false	bool
PermittedLogonTimes	null	byte[]
PermittedWorkstations	{System.DirectoryServices.AccountManagement.PrincipalValueCollection	System.DirectoryServices.AccountManageme
SamAccountName	[redacted]	string
ScriptPath	null	string
Sid	{S-1-5-21-2023297446-2633530542-46019157-4110}	System.Security.Principal.SecurityIdentifier
SmartcardLogonRequired	false	bool
StructuralObjectClass	"user"	string
Surname	"ORHAN"	string
UserCannotChangePassword	false	bool
UserPrincipalName	"[redacted]"	string
VoiceTelephoneNumber	null	string
Static members		
Non-Public members		

Ekran görüntüsünde de görüldüğü üzere bu şekilde kullanıcı bilgileri AD Sunucusu üzerinden alındığında hesap ile ilgili her türlü bilgi elde edilebiliyor.

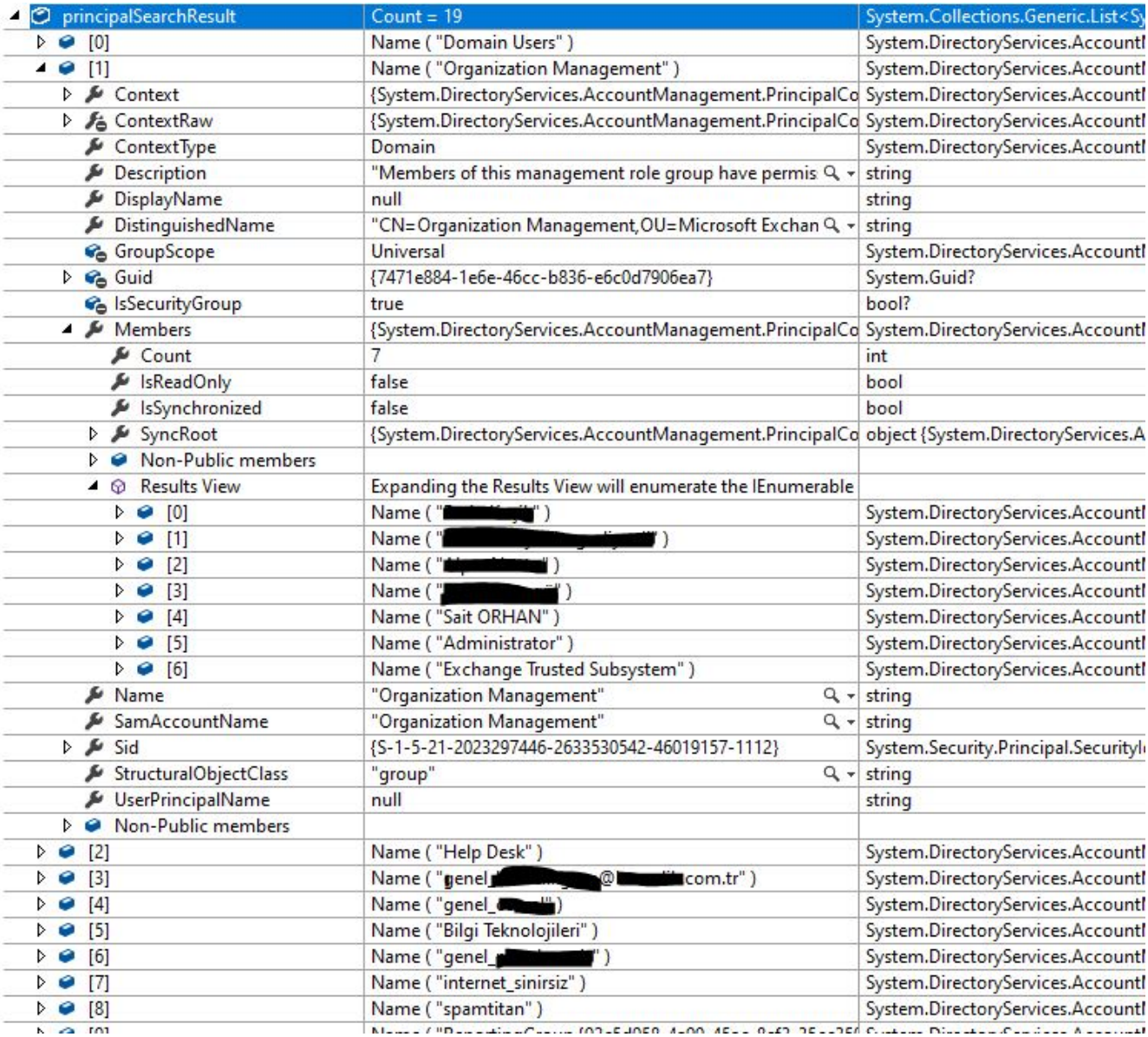
Kullanıcının Üyesi Olduğu Grupları Bulma

Kullanıcının gruplarını bulmak için öncelikle "UserPrincipal.FindByIdentity" metodu ile üyesi olduğu grupların sorgulanacağı kullanıcıyı sistemden alıyoruz. UserPrincipal.FindByIdentity metodu üç parametre alır. Bu parametreler sırasıyla;

1. Yazının başında tanımını yaptığımız PrincipalContext nesnesi
2. Kullanıcı hesabı sorgulanırken kullanılacak özellik. Biz örneğimizde SamAccountName özelliği ile sorgulama yapacağımızdan IdentityType.SamAccountName değerini kullanıyoruz.
3. Sorgulama yapılacak özelliğe ait değer

```
PrincipalContext principalContext = new PrincipalContext(ContextType.Domain, "saitorhan.local", "Administrator", "123456qaZ.");
UserPrincipal userPrincipal = UserPrincipal.FindByIdentity(principalContext, IdentityType.SamAccountName, "saitorhan");
List<GroupPrincipal> principalSearchResult = userPrincipal.GetGroups().Cast<GroupPrincipal>().ToList();
```

Dönen UserPrincipal nesnesinin GetGroups metodu ile kullanıcının üyesi olduğu gruplar alınır ve gene Cast<GroupPrincipal>() metodu ile dönen sonucu grupların temsil edildiği GroupPrincipal türüne çeviriyoruz.



principalSearchResult	Count = 19	System.Collections.Generic.List<Sy
[0]	Name ("Domain Users")	System.DirectoryServices.AccountI
[1]	Name ("Organization Management")	System.DirectoryServices.AccountI
Context	{System.DirectoryServices.AccountManagement.PrincipalCo	System.DirectoryServices.AccountI
ContextRaw	{System.DirectoryServices.AccountManagement.PrincipalCo	System.DirectoryServices.AccountI
ContextType	Domain	System.DirectoryServices.AccountI
Description	"Members of this management role group have permis	string
DisplayName	null	string
DistinguishedName	"CN=Organization Management,OU=Microsoft Exchan	string
GroupScope	Universal	System.DirectoryServices.AccountI
Guid	{7471e884-1e6e-46cc-b836-e6c0d7906ea7}	System.Guid?
IsSecurityGroup	true	bool?
Members	{System.DirectoryServices.AccountManagement.PrincipalCo	System.DirectoryServices.AccountI
Count	7	int
IsReadOnly	false	bool
IsSynchronized	false	bool
SyncRoot	{System.DirectoryServices.AccountManagement.PrincipalCo	object {System.DirectoryServices.A
Non-Public members		
Results View	Expanding the Results View will enumerate the IEnumerable	
[0]	Name ("[redacted]")	System.DirectoryServices.AccountI
[1]	Name ("[redacted]")	System.DirectoryServices.AccountI
[2]	Name ("[redacted]")	System.DirectoryServices.AccountI
[3]	Name ("[redacted]")	System.DirectoryServices.AccountI
[4]	Name ("Sait ORHAN")	System.DirectoryServices.AccountI
[5]	Name ("Administrator")	System.DirectoryServices.AccountI
[6]	Name ("Exchange Trusted Subsystem")	System.DirectoryServices.AccountI
Name	"Organization Management"	string
SamAccountName	"Organization Management"	string
Sid	{S-1-5-21-2023297446-2633530542-46019157-1112}	System.Security.Principal.Securityl
StructuralObjectClass	"group"	string
UserPrincipalName	null	string
Non-Public members		
[2]	Name ("Help Desk")	System.DirectoryServices.AccountI
[3]	Name ("genel[redacted]com.tr")	System.DirectoryServices.AccountI
[4]	Name ("genel_[redacted]")	System.DirectoryServices.AccountI
[5]	Name ("Bilgi Teknolojileri")	System.DirectoryServices.AccountI
[6]	Name ("genel_[redacted]")	System.DirectoryServices.AccountI
[7]	Name ("internet_sinirsiz")	System.DirectoryServices.AccountI
[8]	Name ("spamtitan")	System.DirectoryServices.AccountI

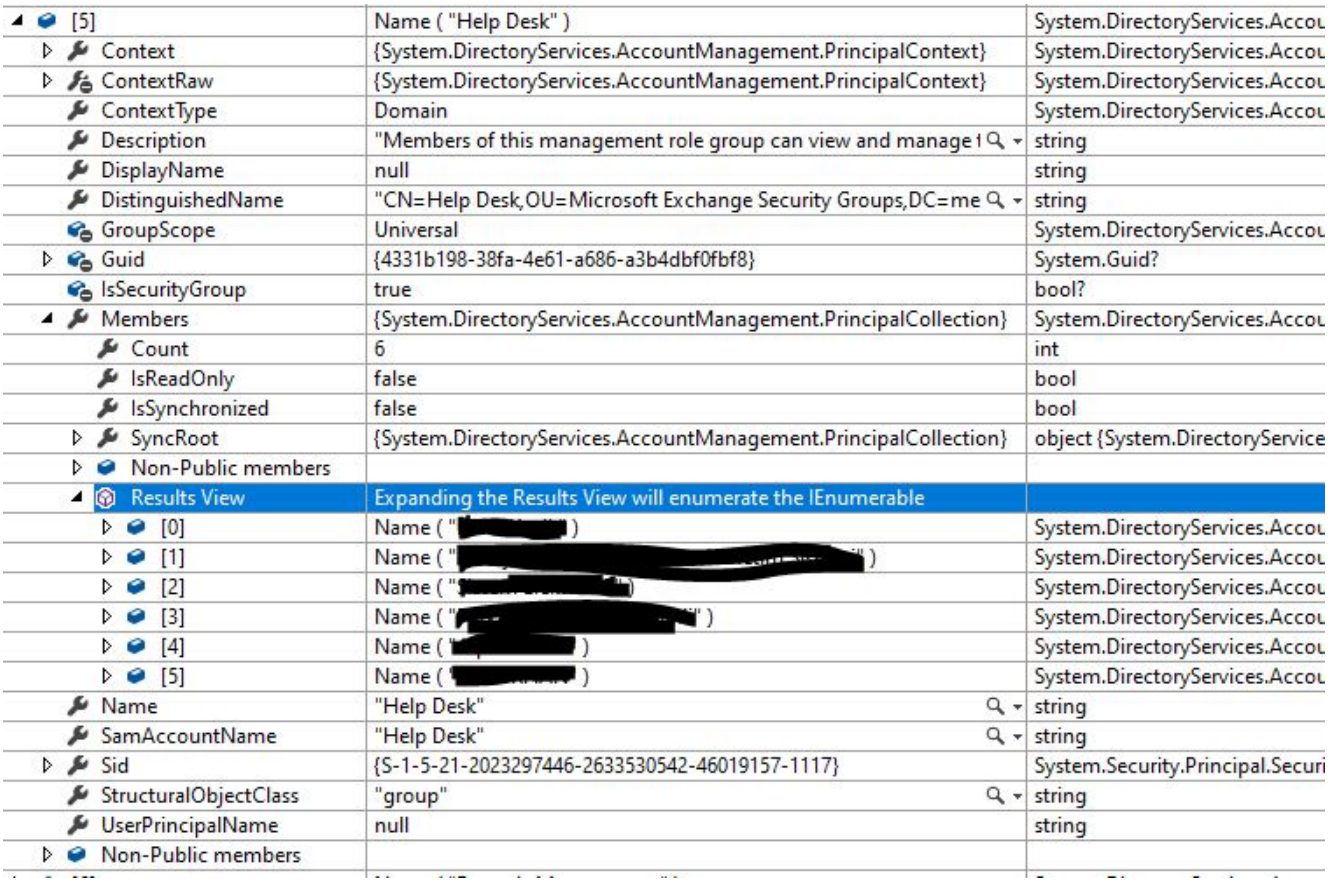
Ekran görüntüsünde de görünen enteresan bir duruma dikkatinizi çekmek istiyorum. Sonuç dönen grupların “Members” özelliğinin içinde o gruba üye olan kullanıcıları görebiliyoruz. Tabi bu kullanıcıların içerisinde de üye oldukları gruplar görünüyor. Bu iç içe sonuç kümeleri sayesinde aslında kullanıcının birinin hesabından girip aslında bütün kullanıcı ve grup bilgilerine ulaşılabilir.

AD Gruplarının Sistemden Alınması

AD sunucusundan grupların alınması işlemi de yazının ilk

maddesi olan kullanıcı bilgilerinin alınması ile aynı adımlardan oluşmaktadır. Aradaki fark burada UserPrincipal yerine GroupPrincipal sınıfı kullanılmaktadır.

```
PrincipalContext principalContext = new PrincipalContext(ContextType.Domain, "saitorhan.local", "Administrator", "123456qaZ.");
GroupPrincipal groupPrincipal = new GroupPrincipal(principalContext);
PrincipalSearcher principalSearcher = new PrincipalSearcher(groupPrincipal);
List<GroupPrincipal> groupPrincipals = principalSearcher.FindAll().Cast<GroupPrincipal>().ToList();
```



AD Grup Üyelerine Kullanıcı Ekleme

AD gruplarından birine kullanıcı eklemek için öncelikle

GroupPrincipal.FindByIdentity metodu ile kullanıcı eklenecek grup seçilir. Daha sonra dönen GroupPrincipal türündeki grup nesnesinin Members özelliğine Add metodu (group.Members.Add) ile kullanıcı eklenir. Add metodu üç parametre alır. Bu parametreler;

1. Sunucu ile bağlantıyı sağlayan PrincipalContext
2. Eklenecek kullanıcının hangi özelliği ile bulunup ekleneceği
3. Eklenecek kullanıcının ikinci parametrede verilen özelliğinin değeri

```
PrincipalContext principalContext = new PrincipalContext(ContextType.Domain, "saitorhan.local", "Administrator", "123456qaZ.");
GroupPrincipal group = GroupPrincipal.FindByIdentity(principalContext, "Help Desk");
group.Members.Add(principalContext, IdentityType.SamAccountName, "saitorhan");
group.Save();
```

AD Grup Üyelerinden Kullanıcı Silmek

Gruptan kullanıcı silinirken de ekleme ile aynı adımlar takip edilir. Aradaki tek fark Add metodu yerine Remove metodu kullanılır.

```
PrincipalContext principalContext = new PrincipalContext(ContextType.Domain, "saitorhan.local", "Administrator", "123456qaZ.");
GroupPrincipal group = GroupPrincipal.FindByIdentity(principalContext, "Help Desk");
group.Members.Remove(principalContext, IdentityType.SamAccountName, Username);
```

```
group.Save();
```

AD Kullanıcı Parolası Doğrulama

PrincipalContext.ValidateCredentials metodu, parametre olarak aldığı kullanıcı adı ve parolanın eşleşip eşleşmediğini kontrol eder. İkinci parametrede verilen parola ilk parametrede verilen kullanıcıya aitse “true” yanlış şifre ise de “false” değer döner.

```
PrincipalContext principalContext = new PrincipalContext(ContextType.Domain, "saitorhan.local", "Administrator", "123456qaZ.");
bool credentials = principalContext.ValidateCredentials("bilalorhan", "Parolam1234");
```

AD Kullanıcı Şifresi Değiştirme

AD kullanıcısına ait şifreyi değiştirmek için UserPrincipal sınıfının SetPassword metodu kullanılır.

```
PrincipalContext principalContext = new PrincipalContext(ContextType.Domain, "saitorhan.local", "Administrator", "123456qaZ.");
        UserPrincipal userPrincipal = UserPrincipal.FindByIdentity(principalContext, IdentityType.SamAccountName, "bilalorhan");
userPrincipal.SetPassword("Parolam1234");
userPrincipal.Save();
```

AD Kullanıcıyı İlk Oturum Açmada Parola Değiştirmeye Zorlama

Kullanıcıyı ilk oturumda parola değiştirmeye zorlamak için öncelikle “userPrincipal.PasswordNeverExpires = false” ifadesi ile kullanıcının şifresinin süresiz geçerli olması iptal edilir. İkinci adımda “userPrincipal.ExpirePasswordNow()” metodu ile kullanıcının parolasını süresi sona ermişe çekilir.

```
PrincipalContext principalContext = new PrincipalContext(ContextType.Domain, "saitorhan.local", "Administrator", "123456qaZ.");
UserPrincipal userPrincipal = UserPrincipal.FindByIdentity(principalContext, IdentityType.SamAccountName, userName);
userPrincipal.PasswordNeverExpires = false;
userPrincipal.ExpirePasswordNow();
userPrincipal.Save();
```

Kaynak Kodlar: https://github.com/saitorhan/CSharpForSystemAdmins

---

SQL Server ve MySQL Server Destekli Yazılım Geliştirme

Kurumsal olarak geliştirilen yazılımların özelliklerine

bakıldığında hemen hemen hepsinde en az iki veri tabanı sistemini desteklediği belirtilmiştir. Bu özellik ilk bakışta her ne kadar geliştirilmesi zor bir özellik olarak görünse de aslında sadece temel kalıtım özellikleri kullanılarak geliştirilebilir. Bu yazımızda biz de herhangi bir harici kütüphane kullanmadan bu işlemi nasıl yaptığımızı inceleyeceğiz.

Bu işle için öncelikle uygulamamızın destekleyeceği veri tabanı sistemlerinde eş yapılı bir veri tabanı yapısı kurulur. Biz örneğimizi tek tablo üzerinden götüreceğiz.

MySQL Tablo Yapısı:

```
CREATE TABLE `kisiler` (
  `id` int NOT NULL AUTO_INCREMENT,
  `isim` varchar(100) NOT NULL,
  `soyisim` varchar(100) NOT NULL,
  `telefon` char(10) NOT NULL,
  PRIMARY KEY (`id`)
)
```

SQL Server Tablo Yapısı

```
CREATE TABLE Kisiler2(
Id int PRIMARY KEY IDENTITY(1,1) NOT NULL,
Isim nvarchar(100) NOT NULL,
Soyisim nvarchar(100) NOT NULL,
Telefon char(10) NOT NULL
)
```

Veri tabanı yönetim sistemi üzerinde tablolar oluşturulduktan

sonra bu tablolara karşılık gelen C# sınıfları yazılır. Bu sınıfların yazılasının sebebi veri tabanı yönetim sistemlerimiz ile uygulamamızı ortak bir yapı altında konuşturabilmektir.

Tablolara Karşılık C# Sınıfı

```
class Kisi
   {
       public int Id { get; set; }
       public string Isim { get; set; }
       public string Soyisim { get; set; }
       public string Telefon { get; set; }
   }
```

Sınıfımızı da oluşturduktan sonra artık veri tabanı işlemleri için yapımızı kurmaya başlayabiliriz. Yapımızın temelinde interface (ara yüz) kavramı olduğundan burada interface kavramına değinmek lazım.

Interface

Interface tanımlanırken herhangi bir sınıf tanımlanır gibi tanımlanır ancak türüne class yerine interface yazılır.

Interface içerisinde neler bulunabilir neler bulunamaz:

- Interface içerisinde propertyler bulunabilir.
- Interface içerisinde tanımlanan metotların sadece başlık bildirimi yapılır, gövdesi tanımlanmaz.
- Interface içerisinde tanımlanan property ve metotlar

public olduğundan ayrıca erişim belirteci belirtilmez.

Bu bilgilerden sonra şimdi tanımlanan tablomuz üzerinde yapılacak olan işlemlerin metotlarını barındıran bir interface tanımlayalım.

```
interface IKisi
{
   int KisiEkle(Kisi yeniKisi);
   List<Kisi> KisileriGetir();
}
```

İncelememizi iki metot üzerinden ilerleteceğiz. Tanımlanan interface içerisinde de göründüğü üzere metotların sadece başlık bilgileri girilmiş ancak gövdeleri tanımlanmamış. Gövdeleri yani bu metotların yapacakları işleri bu interfaceden türetilen sınıflar yapacak.

Interface tanımı da yaptığımıza göre şimdi de sıra SQL Server ve MySQL Server işlemlerini yapacak sınıfları tanımlamaya.

SQL Server Veri Tabanı İşlemleri

```
    class MsSqlKisiIslemleri : IKisi
    {
        public int KisiEkle(Kisi yeniKisi)
        {
                    SqlConnection sqlConnection = new
SqlConnection("Data        Source=.\\egitim;Initial
Catalog=OgretmenBilgi;Integrated Security=True");
             SqlCommand sqlCommand = new SqlCommand("INSERT
INTO Kisiler([Isim], [Soyisim], [Telefon]) values(@i, @s,
@t)", sqlConnection);
```

```
                sqlCommand.Parameters.AddWithValue("@i",
yeniKisi.Isim);
                sqlCommand.Parameters.AddWithValue("@s",
yeniKisi.Soyisim);
                sqlCommand.Parameters.AddWithValue("@t",
yeniKisi.Telefon);


            sqlConnection.Open();
            int i = sqlCommand.ExecuteNonQuery();
            sqlConnection.Close();

            return i;
        }

        public List<Kisi> KisileriGetir()
        {
            List<Kisi> kisiler  = new List<Kisi>();
                    SqlConnection sqlConnection = new
SqlConnection("Data        Source=.\\egitim;Initial
Catalog=OgretmenBilgi;Integrated Security=True");
            SqlCommand sqlCommand = new SqlCommand("select Id,
Isim, Soyisim, Telefon from Kisiler", sqlConnection);
            sqlConnection.Open();
                        SqlDataReader sqlDataReader =
sqlCommand.ExecuteReader();

            while (sqlDataReader.Read())
            {
                Kisi kisi = new Kisi();
                kisi.Id = sqlDataReader.GetInt32(0);
                kisi.Isim = sqlDataReader.GetString(1);
                kisi.Soyisim = sqlDataReader.GetString(2);
                kisi.Telefon = sqlDataReader.GetString(3);
                kisiler.Add(kisi);
            }


            sqlConnection.Close();
            return kisiler;
        }
```

```
    }
```

MySQL Server Veri Tabanı İşlemleri

```
    class MySqlKisiIslemleri : IKisi
    {
        public int KisiEkle(Kisi yeniKisi)
        {
                  MySqlConnection sqlConnection = new
MySqlConnection("server=localhost;user
id=root;database=egitim;pwd=123456qaZ.");
            MySqlCommand sqlCommand = new MySqlCommand("INSERT
INTO Kisiler(isim, soyisim, telefon) values(@i, @s, @t)",
sqlConnection);

                sqlCommand.Parameters.AddWithValue("@i",
yeniKisi.Isim);
                sqlCommand.Parameters.AddWithValue("@s",
yeniKisi.Soyisim);
                sqlCommand.Parameters.AddWithValue("@t",
yeniKisi.Telefon);


            sqlConnection.Open();
            int i = sqlCommand.ExecuteNonQuery();
            sqlConnection.Close();

            return i;
        }

        public List<Kisi> KisileriGetir()
        {
            List<Kisi> kisiler = new List<Kisi>();
                  MySqlConnection sqlConnection = new
MySqlConnection("server=localhost;user
id=root;database=egitim;pwd=123456qaZ.");
            MySqlCommand sqlCommand = new MySqlCommand("select
id, isim, soyisim, telefon from Kisiler", sqlConnection);
```

```
            sqlConnection.Open();
                    MySqlDataReader sqlDataReader =
sqlCommand.ExecuteReader();

            while (sqlDataReader.Read())
            {
                Kisi kisi = new Kisi();
                kisi.Id = sqlDataReader.GetInt32(0);
                kisi.Isim = sqlDataReader.GetString(1);
                kisi.Soyisim = sqlDataReader.GetString(2);
                kisi.Telefon = sqlDataReader.GetString(3);
                kisiler.Add(kisi);
            }


            sqlConnection.Close();
            return kisiler;
        }
    }
```

İlgili işlem sınıflarımızı da tanımladıktan sonra şimdi de aralarında ki benzerlik ve farklılıklarına göz atalım.

Benzerlikler:

- Her iki sınıf da IKisi interfacesinden türetilmiştir ve interface özelliği gereği aynı metot tanımlarını barındırırlar
- Metotların aldıkları parametre türleri ve dönüş türleri aynıdır.

Farklılıklar:

- Metotların gövdelerinde yer alan veri tabanı işlemleri

sınıfların sorumlu oldukları veri tabanı sisteminin kodlarıdır.

Ve geldik son işleme: Hem SQL Server hem MySQL Server için gerekli sınıflar ve kodlar yazıldığına göre uygulamamızda ilgili veri tabanı yönetim sistemine nasıl ulaşacağız?

Öncelikle uygulamamızın hangi veri tabanı sisteminde çalışacağını bir sistem parametresi olarak kaydetmek gerekiyor. Bu kayıt işlemi Properties -> Settings altında veya app.config içerinde olabilir. Yazımız içerisinde bu parametreye “dbType” ismini vereceğiz.

İlk işlemimizi olan kişi ekleme metodunu çağırmak için gerekli kodları yazalım.

```
IKisi iKisi = null;

if (dbType == SQLServer)
{
    iKisi = new MsSqlKisiIslemleri();
}

else if(dbType == MySQL)
{
    iKisi = new MySqlKisiIslemleri();
}

Kisi kisi = new Kisi
{
    Isim = textBoxAd.Text,
    Soyisim = textBoxSoyad.Text,
    Telefon = textBoxTelefon.Text
};
```

```
int kisiEkle = iKisi.KisiEkle(kisi);
```

Öncelikle veri tabanı türüne göre ilgili sınıfı barındıracak olan IKisi interfacesinden bir tanımlama yapıyoruz. Daha sonra dbType isimli sistem parametresinin değerine göre bu interfaceye ilgili sınıfın bir örneği atanıyor. Bu adımdan sonra metotlar interface üzerinden çağrıldığında artık kendisine atanan sınıfın içeriğini çalıştırır. Dolayısı ile bu adımdan sonra uygulamamız artık çoklu veri tabanı yönetim sistemi desteğine sahip olmuş oluyor.

Kodlamada son olarak kişileri listeleme işlemi metotlarını çağıralım.

```
 IKisi iKisi = null;

if (comboBoxDb.SelectedIndex == 0)
{
    iKisi = new MsSqlKisiIslemleri();
}

else if (comboBoxDb.SelectedIndex == 1)
     {
          iKisi = new MySqlKisiIslemleri();
      }

List<Kisi> kisiler = iKisi.KisileriGetir();
```

Kişi ekleme işleminde olduğu gibi burada da oluşturulan interface ve ona atanan sınıflar üzerinden işlemler yapılıyor.

Yazımızın sonucunu mini bir öz eleştiri ile kapatacak olursak; Eğitimlerde, okulda gördüğümüz her yeni bilginin muhakkak

surette pratik hayatta profesyonel kullanımda nasıl kullanıldSQL Server ve MySQL Server Destekli Yazılım Geliştirmeığını sorgulayalım. Kendi adıma ilk olarak çoklu veri tabanı destekli yazılım geliştirme konusunu gördüğümde “çok zor bir işlem” demiştim ancak işlemi yaptığımda gördüm ki aslında çok iyi bildiğim bir konuymuş.

Hepinize bol ve bugsız bir kodlama yaşamı diliyorum

C# Form Çoklu Dil Desteği

Yerelleştirme diye tabir edilen uygulamanın birden çok dili desteklemesi, uygulamanın başarısını etkileyen faktörlerden biridir.

Aşağıdaki videomuzda C# form uygulamamıza çoklu dil desteğini nasıl sağlayacağımızı ve kullanıcının seçimine göre programın dilini ayarlayacağımızı işliyoruz.

C# Eklenti (Plugin) Destekli Yazılım Geliştirme

Yazılım geliştirilirken göz önünde bulundurulması gereken konulardan biri de uygulamanın geliştirilebilir olması ve bağımsız geliştiricilerin uygulamaya katkılarına destek sağlanması konusudur.

Bu tür uygulamalara en güzel örnekler internet tarayıcıların eklenti desteği ile vardıkları mükemmel özelliklerdir. C# dilinde de bu destek kolaylıkla, geliştirilen uygulamalara eklenebilir. Eklenti (Plugin) desteğinin uygulamaya nasıl eklendiğini öğrenmek için aşağıdaki videoya göz atabilirsiniz.

C# User Control Nedir ve Nasıl Oluşturulur

Windows form veya web form geliştirirken çoğu zaman .Net standart kütüphanesi ile gelen kontroller (buton, textbox vs.) yeterli olsa da özellikle projede geliştirilen sınıflara özgü görsel kontrol geliştirme gerekmektedir. Bu durumlarda User Control dediğimiz konu devreye girmektedir.

Örneğin bir sınav uygulaması geliştirilecek, bunun için en temel sınıfımız Soru sınıfı olacak. Soru sınıfına ait temel

özellikle ise şunlar olacaktır.

- Soru metni
- Cevap şıkları (A,B,C,D,E)
- Doğru cevap
- Kullanıcının soruya verdiği cevap

Bu durumda 20 soruluk bir test için 20 tane metin alanı, 5 şıktan 20 soru için 100 radiobutton vs. kontrol eklenmesi gerekecektir. Tabi bütün bunları kontrol edecek kodlar. Bir sürü iş yani Bütün bu kargaşa yerine yapılabilecek çözüm soru sınıfı için bir user control oluşturmak ve bir soru için gereken tasarımı yaptıktan ve ilgili kodları yazdıktan sonra her soru için bunu kullanmak.

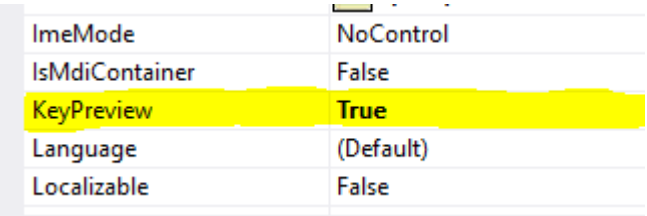
Aşağıdaki videomuzda user controlun nasıl tanımlandığını ve kullanımı ile ilgili ayrıntılı bilgi bulabilirsiniz.

C# Forma Kısayol Ekleme

Yazılım geliştirilirken göz önünde bulundurulması gereken noktalardan biri de kullanıcı deneyimidir. Kullanıcı deneyiminde de ilk sırada efektif olarak oluşturulmuş kısa yollardır.

Form üzerinde kısayol tanımlanabilmesi için formun KeyPreview

özelliğinin True olarak ayarlanmış olması gerekmektedir.



Bu özelliğin ayarlanmasından sonra forma ait KeyDown olayının (Event) ayarlanması gerekmektedir. Parametrede yer alan KeyEventArgs parametresi basılan tuşlara ait bilgileri tutmaktadır.

Örnek 1:

```
private void MyForm_KeyDown(object sender, KeyEventArgs e)
{
 if(e.Modifiers == Keys.Control && e.KeyCode == Keys.N)
 {
 // (Ctrl+N) tuşlarına basıldığında yapılması istenen işlemler
 }
}
```

Örnek 2:

```
private void MyForm_KeyDown(object sender, KeyEventArgs e)
{
 if (e.Control && e.Shift && e.KeyCode == Keys.O)
 {
  // Ctrl+Shft+O tuşlarına basıldığında yapılması istenen
işlemler
 }
}
```

Aşağıdaki videodan konuyu daha ayrıntılı inceleyebilirsiniz.

C# Çoklu Form Uygulamaları ve Formlar Arası Veri Alış Verişi

Geliştirilen uygulamaların nerede ise tamamında birden fazla ekran kullanılmaktadır. Bu yazımızda birden fazla ekran içeren uygulamalarda ekranların beraber kullanılması ve ekranlar arası veri alış verişini inceleyeceğiz.

Bir ekranda iken ikinci bir ekranı açmak için iki metot bulunmaktadır. Bu metotlar ve açıklamaları;

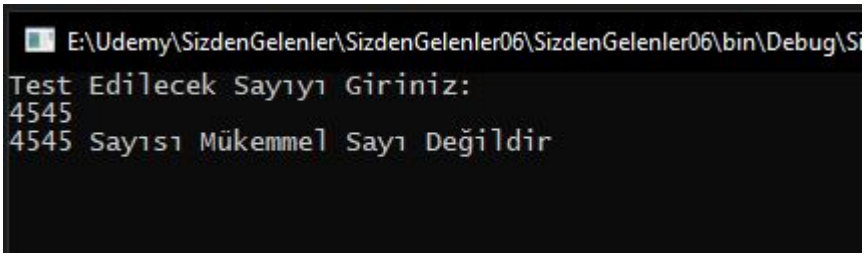
- [Form].Show() : Açılacak form açıldıktan sonra ana form kullanılmaya devam edilebilir şekilde açılır.
- [Form].ShowDialog(): Açılacak form, açıldıktan sonra ana form kullanılamaz. Açılan form kapandıktan sonra ana form kullanılabilir. Kullanıcıya soru sorulacağı zaman çoğunlukla kullanılır. Bununla ilgili MessageBox yazımızı inceleyebilirsiniz.

İki metotta da öncelikle açılacak formu bir örneği oluşturulduktan sonra ihtiyaca göre Show veya ShowDialog metotları ile form açılır.

```
Form form1 = new Form1();
form1.Show();
```

Aşağıdaki videomuzda çoklu form uygulamaları ve formlar arası veri alış veriş yöntemlerini ayrıntılı şekilde bulabilirsiniz.

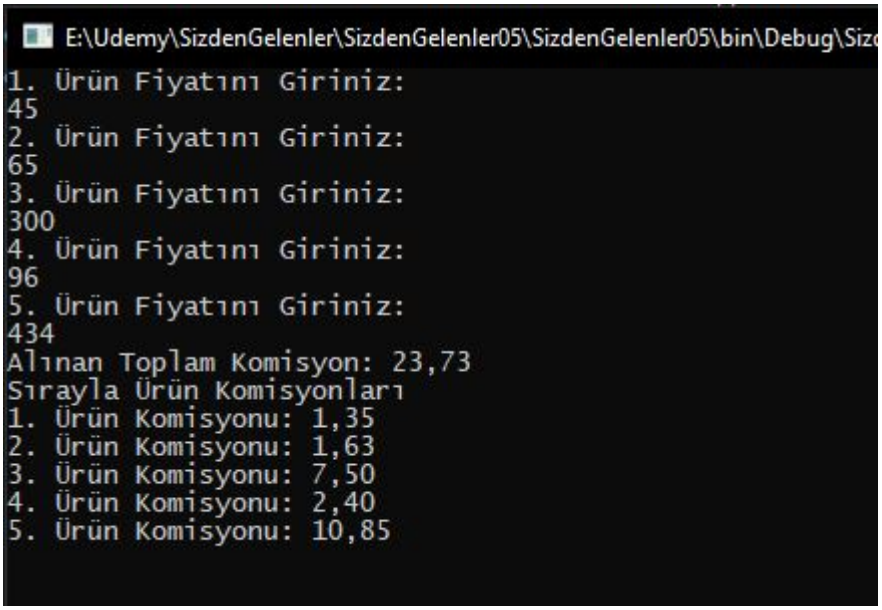
C# Mükemmel Sayı Testi



Kendisi hariç bütün pozitif çarpanları (tam bölenleri) toplamı, yine kendisine eşit olan sayılara ”mükemmel sayı” denir. Örneğin 6=1+2+3 ve 28=1+2+4+7+14 gibi. Buna göre klavyeden girilen bir tam sayının “mükemmel sayı” olup olmadığını kontrol eden C# programını kodlarını yazınız

https://github.com/saitorhan/SizdenGelenler06

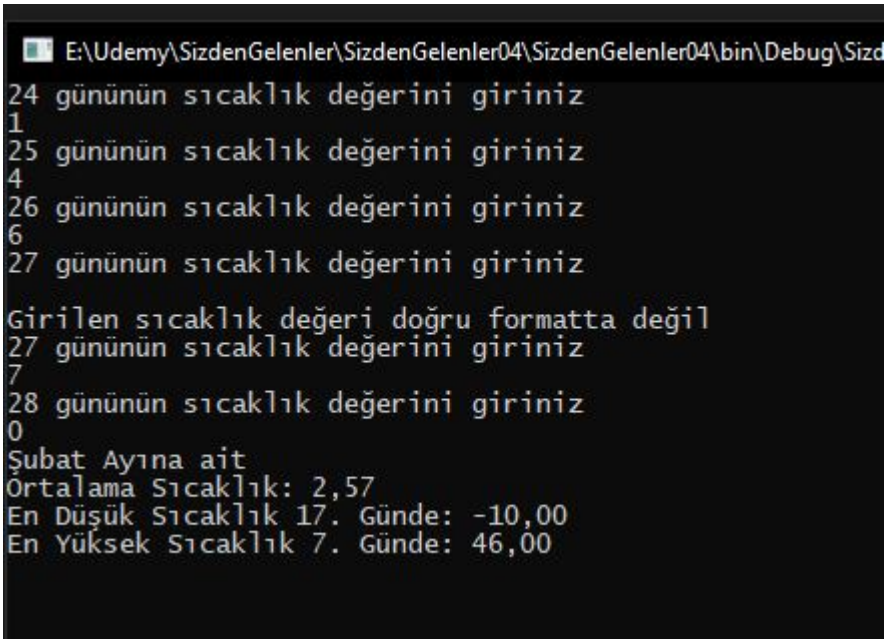
C# Console Komisyon Hesaplama



Bir komisyoncu sattığı mallardan fiyatı 50 TL kadar olanlardan %3, daha fazla olanlardan ise %2 komisyon almaktadır. Klavyeden girilen teker teker girilen 5 malın komisyonlarını bulup ekrana yazdıran ve en sonunda da toplam komisyonu ekrana yazdıran programını yazınız.

https://github.com/saitorhan/SizdenGelenler05

(C# Console, Diziler) Meteoroloji Sıcaklık Analizi



Meteoroloji merkezi için bir program tasarlanılması istenilmiştir. Programın çalışma şekli ise şöyle olmalıdır:

a. İlk önce hangi ay için sıcaklık bilgisi girileceği kullanıcıya sorulacaktır.

b. Girilen ay bilgisine uygun olarak o ayda kaç tane gün var ise kullanıcıdan gün sayısı kadar sıcaklık bilgisi girilmesi istenilecektir (şubat ayı için gün sayısını 28 alınız).

c. Sıcaklık veri girişi bittikten sonra o ayın sıcaklık ortalaması ve en düşük sıcaklık bilgisi ekrana yazdırılacaktır. Bu işlemden sonra program sonlanacaktır. Örnek Çıktı: Şubat Ayına ait Ortalama Sıcaklık=15,6 derecedir ve En düşük sıcaklık 6.Gün=10, 1 derecedir.

https://github.com/saitorhan/SizdenGelenler04