

SQL SERVER MANAGEMENT OBJETS'İ KULLANMAK

21

Bu bölüme kadar, SQL Server'ın T-SQL sorgu geliştirme yöntem ve yetenekleriyle, bazı veritabanı yöneticiliği konularını inceledik. Hem yazılım geliştirici, hem de veritabanı yöneticilerinin ortak ihtiyacı olan bir diğer özel konu ise, özel veritabanı işlem ve yönetimlerini gerçekleştirmek için, **Management Studio** gibi bir veritabanı yönetim ve geliştirme istemci aracı tasarlamak ve programsal olarak geliştirmektir.

Management Studio gibi bir istemci aracının olmadığı ortamlarda ya da yapılan bazı işlemlerin farklı bir kullanıcı arayüzü ile yönetilmek istendiğinde, .NET alt yapısını kullanarak veritabanına doğrudan ulaşılabilir. Veritabanı motoruna doğrudan bağlanarak, nesne oluşturma, veri yönetimi, sunucu yönetimi gibi veritabanı programlama tarafında gerçekleştirilecek birçok önemli işlemin yönetiminin yapılmasını sağlar. Bunları sağlayan nesne modelinin adı SMO, yani SQL Server Management Objects'dir.

Neler yapılabilir?

- Prosedür, trigger, view gibi nesneler oluşturmak.
- Veritabanı yedeği (*backup*) alma ve yedekten geri dönme işlemi gerçekleştirmek.
- Veritabanı script'i elde etmek.
- Bir veritabanı oluşturmak.
- Bir tranaction oluşturmak.

- Bir job ve agent ile ilişkili diğer görevler yerine getirmek.
- LinkedServer oluşturmak.
- Index oluşturmak ve yönetmek.
- Kullanıcı tanımlı fonksiyon oluşturmak.
- SQL Server'da sunucusunda meydana gelen olayların yakalanması ve olay durumunu yönetmek.
- Sunucudaki nesne tiplerine koleksiyon olarak referans verme yeteneği. (T-SQL'de koleksiyon yoktur).
- Veritabanı şeması oluşturmak.
- SQL Server mail işlemlerini yönetmek.
- Veritabanı özellikleri ve ayarları hakkında bilgi almak.
- FullTextServices yönetimi.

Bu ve bunlar gibi birçok nesne SMO ile geliştirilebilir ve yönetilebilir.

SQL SERVER MANAGEMENT OBJECTS UYGULAMALARI

SMO nesne modelini anlamanın en iyi yolu, birçok yeteneğini kullanarak, ihtiyaçlar doğrultusunda neler yapılabileceğini görmektir.

SQL Server'da, SMO ile uygulama geliştirebilmek için Visual Studio.NET editörü kullanılır. Bu kitaptaki örnek uygulamalarda, Visual Studio.NET editörü ve C# programlama dilini kullanacağız.

Bu bölüm, belirli seviyede programlama becerisi ve C# programlama dili bilgisi gerektirir. .NET tabanlı bir programlama dili kullanıyorsanız C# yerine, geliştiricisi olduğunuz programlama dilini de kullanabilirsiniz.

Programlama ortamında SMO nesne modelini kullanabilmek için bir Windows ya da konsol (Console) uygulaması oluşturun. Örneklerimizde Windows uygulaması geliştireceğiz. Ancak, bazı kodları değiştirerek konsol uygulaması ile daha hızlı sonuç elde edilebilir.

SMO nesnelerinin kullanılabilmesi için projeye aşağıdaki DLL'lerin referans olarak eklenmesi gerekir.

```

Microsoft.SqlServer.ConnectionInfo
Microsoft.SqlServer.Management.Sdk.Sfc
Microsoft.SqlServer.Smo
Microsoft.SqlServer.SmoExtended
Microsoft.SqlServer.SqlEnum
Microsoft.SqlServer.SmoEnum

```

Belirtilen bu DLL'ler ile kitaptaki tüm örnekler gerçekleştirilebilir.

Eklenen DLL'lerin proje formlarında kullanılabilmesi için form içerisindeki, yukarıda bulunan **using** bölümünde bazı namespace tanımlamaları yapılması gerekecektir. Namespace tanımlamaları yapılmazsa ya da yukarıdaki DLL'ler eklenmezse SMO nesneleri aşağıdaki gibi altlarında kırmızı çizgi ve siyah yazı fontu ile görünecektir.

```
conn = new ServerConnection("dijibil-pc", "sa", "cihan..");
```

The type or namespace name 'ServerConnection' could not be found (are you missing a using directive or an assembly reference?)

Böyle bir görüntü oluştuğunda, projeye ilgili DLL'lerin referans edilip edilmediği kontrol edilmelidir. Sorun halen devam ediyorsa, formun **using** kısmındaki **namespace** tanımlamalarına bakılmalıdır.

Bir nesnenin namespace'ini kolay yoldan eklemenin iki yolu vardır.

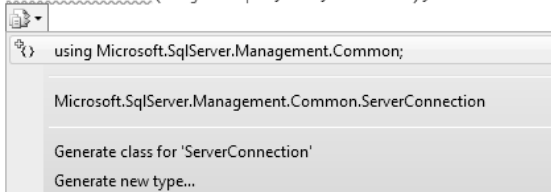
Klavye kısa yolları ile namespace eklemek;

- Altı kırmızı çizgili olan nesnenin üzerine fare ile tıklayın.
- **CTRL+ALT+F10** tuş kombinasyonunu kullanarak namespace ekleme penceresini açın ve ilgili namespace'i using kısmına ekleyin.

Fare ile namespace eklemek;

- Altı kırmızı çizgili olan nesnenin üzerine fare ile tıklayın.
- Nesne isminin sol alt köşesinde beliren mavi çizginin üzerine tıklayın ve açılan namespace ekleme penceresinde namespace eklemesini gerçekleştirin.

```
conn = new ServerConnection("dijibil-pc", "sa", "cihan..");
```





Kitabın bu bölümünde, SMO konusunu detaylarıyla kavrayabilmek için bir çok uygulama gerçekleştireceğiz. Bu uygulamaların kaynak kodlarını kitabın CD ekinde bulabilirsiniz.

SMO ile ilgili hazırlanan uygulamaların ana ekran görüntüsü aşağıdaki gibidir.



Resimde görünen tüm örnekleri sırasıyla, parçalar halinde yapacağız.

SMO İLE SUNUCU BAĞLANTISI OLUŞTURMAK

SMO ile uygulama geliştirebilmek için her uygulamada gerekli olan ortak özellik, sunucuya bir bağlantı ile bağlı olmaktır. Bu bağlantıyı oluşturabilmek için sunucu nesnesinin bir örneğine ve SQL Server erişim bilgilerine sahip olunmalıdır.

Windows Authentication ile sunucu bağlantısı gerçekleştirmek için aşağıdaki yöntem kullanılır.

```
ServerConnection conn = new ServerConnection();
conn.LoginSecure = true;
Server svr = new Server(conn);
svr.ConnectionContext.Connect();
```

SQL Server Authentication ile sunucu bağlantısı gerçekleştirmek için aşağıdaki yöntem kullanılır.

```
conn = new ServerConnection(
```

▲ 5 of 6 ▼ ServerConnection.ServerConnection(string serverInstance, string userName, string password)

Initializes a new instance of the Microsoft.SqlServer.Management.Common.ServerConnection class with the specified server instance and logon credentials.
serverInstance: A System.String value that specifies the name of the instance of the server with which the connection is established.

```

ServerConnection conn = new ServerConnection("dijibil-pc","sa","cihan..");
conn.LoginSecure = true;
Server svr = new Server(conn);
svr.ConnectionContext.Connect();

```

ADO.NET ya da farklı veri erişim katmanları ile veritabanı uygulamaları geliştirenler için bu kodun ne işe yaradığını anlamak zor değildir. Bu sorguda, sunucuya bağlantı oluşturabilmek için **ServerConnection** nesnesinin örneği oluşturuldu. Daha sonra bu bağlantı, sunucu işlemlerini yapmak için kullanılacak **Server** nesnesinin örneğine parametre olarak verildi.

Artık **conn** isimli bağlantı örneği kullanılarak, Server nesnesi yardımıyla sunucu iletişimi sağlanabilir.

SUNUCU ÖZELLİKLERİNİ ELDE ETMEK

Sunucu ile ilgili özellikler birçok programsal durumda gerekli olabilir. Sunucunun versiyon güncellemeleri, uyumluluk vb. bir çok durumda farklı özelliklerin elde edilmesi gerekir. Bu işlem için SMO kullanılabilir.

Basit bir form oluşturup, içerisine bir ListBox kontrolü yerleştirdikten sonra aşağıdaki kodları yazalım.

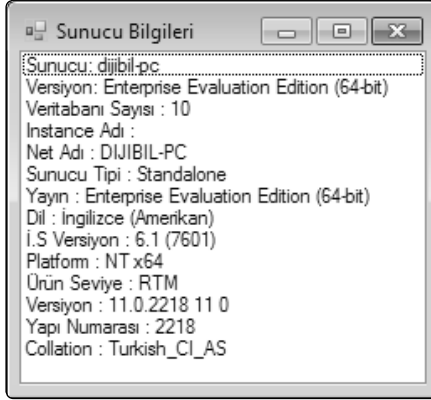
```

Server srv = new Server(conn);
listBox1.Items.Add("Sunucu: " + srv.Name);
listBox1.Items.Add("Versiyon: " + srv.Information.Edition);
listBox1.Items.Add("Veritabanı Sayısı : " + srv.Databases.Count);
listBox1.Items.Add("Instance Adı : " + srv.InstanceName);
listBox1.Items.Add("Net Adı : " + srv.NetName);
listBox1.Items.Add("Sunucu Tipi : " + srv.ServerType);
listBox1.Items.Add("Yayın : " + srv.Information.Edition);
listBox1.Items.Add("Dil : " + srv.Information.Language);
listBox1.Items.Add("İ.S Versiyon : " + srv.Information.OSVersion);
listBox1.Items.Add("Platform : " + srv.Information.Platform);
listBox1.Items.Add("Ürün Seviye : " + srv.Information.ProductLevel);
listBox1.Items.Add("Versiyon : " + srv.Information.Version + " "
    + srv.Information.VersionMajor + " "
    + srv.Information.VersionMinor);
listBox1.Items.Add("Yapı Numarası : " + srv.Information.BuildNumber);
listBox1.Items.Add("Collation : " + srv.Information.Collation);

```

Server nesnesinin örneği oluşturulurken **conn** parametresi gönderildiğine dikkat edin. Bu parametre, veritabanı bağlantısı oluşturmak için hazırladığımız bir **ServerConnection** nesnesi örneğidir. Sorgu yoğunluğu ve kitapta fazladan yer kaplamaması için bu ve bundan sonraki hiç bir SMO uygulamasında ek bir bağlantı tanımlama sorgusu yazmayacağız.

Proje çalıştırıldığında aşağıdaki gibi bir görüntü elde edilecektir.



Veritabanı ile ilgili sorguda istenen teknik özellikler listelenmektedir.

VERİTABANLARI, DOSYA GRUPLARI VE DOSYALARIN LİSTESİNİ ALMAK

SMO ile sunucudaki veritabanları, dosya grupları ve dosyalar ile ilgili bilgileri elde etmek mümkündür.

Form üzerine üç adet **ListBox** yerleştirdik. Kod sayfasındaki kodlar ise aşağıdaki gibidir.

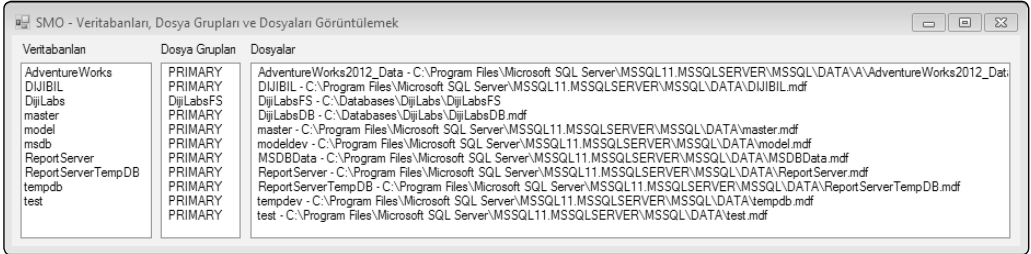
```
Server srv = new Server(conn);
foreach (Database db in srv.Databases)
{
    listBox1.Items.Add(db.Name);
    foreach (FileGroup fg in db.FileGroups)
    {
        listBox2.Items.Add(" " + fg.Name);
        foreach (DataFile df in fg.Files)
```

```

{
    listBox3.Items.Add(" " + df.Name + " - " + df.FileName);
}
}
}

```

Proje çalıştırıldığında aşağıdaki gibi görünecektir.



Solda bulunan `ListBox`'da sunucudaki veritabanları, ortadaki `ListBox`'da dosya grupları ve sağdaki `ListBox`'da ise veritabanı dosyalarının isimleriyle sistemdeki dosya yolları yer almaktadır.

VERİTABANI ÖZELLİKLERİNİ ALMAK

Bir veritabanının özelliklerini ve bu özelliklerin değerlerini elde etmek için SMO kullanılabilir.

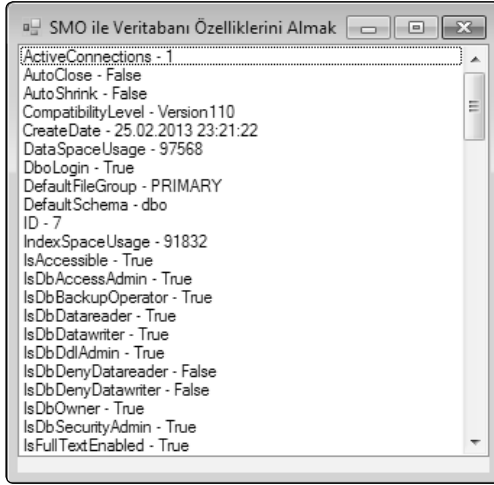
Form üzerine bir `ListBox` yerleştirerek formun `Load` event'ine aşağıdaki kodları yazalım.

```

Server srv = new Server(conn);
Database database = srv.Databases["AdventureWorks"];
foreach (Property prop in database.Properties)
{
    listBox1.Items.Add(prop.Name + " - " + prop.Value);
}

```

Programın çalıştırıldıktan sonraki görünümü aşağıdaki gibi olacaktır.



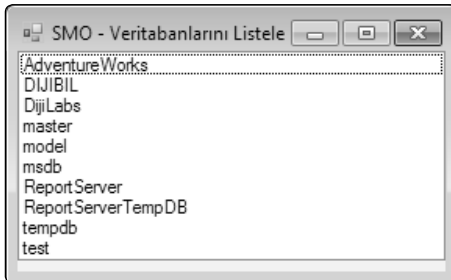
SUNUCUDAKİ TÜM VERİTABANLARINI LİSTELEMELER

Sunucuda bulunan tüm veritabanlarını listeleme işlemi aşağıdaki gibi yapılabilir.

```
Server srv = new Server(conn);
srv.ConnectionContext.Connect();
Database db = srv.Databases["AdventureWorks"];

foreach (Database vt in srv.Databases)
{
    ListBox1.Items.Add(vt.Name.ToString());
}
```

Programın çalıştırdıktan sonraki görünümü aşağıdaki gibi olacaktır.



VERİTABANI OLUŞTURMAK

SMO ile yeni bir veritabanı oluşturulabilir.

Yeni bir veritabanı oluşturmak için kullanılacak bir metot geliştirelim.

```
void VeritabaniOlustur(string vt_ad)
{
    Server srv = new Server(conn);
    Database database = new Database(srv, "" + vt_ad + "");
    database.FileGroups.Add(new FileGroup(database, "PRIMARY"));
    DataFile dtPrimary = new DataFile(database.FileGroups["PRIMARY"],
        "Data", @"C:\Databases\"
        + vt_ad + ".mdf");
    dtPrimary.Size = 77.0 * 1024.0;
    dtPrimary.GrowthType = FileGrowthType.KB;
    dtPrimary.Growth = 1.0 * 1024.0;
    database.FileGroups["PRIMARY"].Files.Add(dtPrimary);

    LogFile logFile = new LogFile(database, "Log",
        @"C:\Databases\"
        + vt_ad + ".ldf");
    logFile.Size = 7.0 * 1024.0;
    logFile.GrowthType = FileGrowthType.Percent;
    logFile.Growth = 10.0;

    database.LogFiles.Add(logFile);
    database.Create();
    database.Refresh();
}
```



Yeni veritabanının GrowthType değerini belirtmek için kullanılan FileGrowthType enum değerinin hata üretmemesi için aşağıdaki DLL projeye eklenmelidir.

Microsoft.SqlServer.SqlEnum

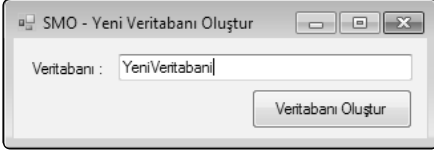
Projeye DLL eklemek için aşağıdaki yolu izleyin.

- **Solution Explorer** panelindeki **References** kısmına sağ tıklayın.
- **Add Reference** butonunu tıklayın. Açılan formda **.NET** tab menüsünden ilgili **DLL** seçimini yapıp, **OK** butonuna tıklayın.

Metot, program içerisinde aşağıdaki gibi çağrılabilir.

```
VeritabaniOlustur(txtVeritabani.Text);
```

KodLab adında yeni bir veritabanı oluşturalım.



Programın çalışabilmesi için, bilgisayarınızın **C:** dizini içerisinde, **Databases** adında bir klasör bulunmalıdır.

İşlem herhangi bir hata üretmeden tamamlandıktan sonra, **C:** dizinindeki **Databases** klasörüne giderek, **KodLab** ismindeki **MDF** ve **LDF** dosyalarını görelim.

	YeniVeritabanı.ldf	25.02.2013 18:58	SQL Server Database Transaction Log File	7.168 KB
	YeniVeritabanı.mdf	25.02.2013 18:58	SQL Server Database Primary Data File	78.848 KB

YeniVeritabanı isimli veritabanını **Management Studio** ile görüntüleyelim.

YeniVeritabanı

VERİTABANI YEDEKLEMEK

SMO nesnelerinin yoğun kullanıldığı işlemlerden biri de veritabanı yedekleme işlemleridir. Veritabanı yedekleme işlemini gerçekleştirecek bir metot yazalım.

```
void Yedekle(string vt_ad)
{
    listBox1.Items.Clear();
    Random rnd = new Random();
    Server srv = new Server(conn);
    Database database = srv.Databases["" + vt_ad + ""];
    Backup backup = new Backup();
    backup.Action = BackupActionType.Database;
    backup.Database = database.Name;
    backup.Devices.AddDevice(@"C:\Backups\Backup"
        + rnd.Next(1, 10000)
```

```

        + ".bak",
        DeviceType.File);
    backup.PercentCompleteNotification = 10;
    backup.PercentComplete += new PercentCompleteEventHandler(backup_
    PercentComplete);
    backup.SqlBackup(srv);
}

```

Veritabanı yedeklenirken, yedekleme işleminin % olarak tamamlanma oranı da öğrenilebilir. Bunun için, **backup** nesnesinin **PercentComplete** olayı kullanılmalıdır.

```

void backup_PercentComplete(object sender, PercentCompleteEventArgs e)
{
    listBox1.Items.Add("%" + e.Percent + " tamamlandı.");
}

```

Yedekle isimli metod, bir butonun **Click** olayı içerisinde şu şekilde kullanılabilir.

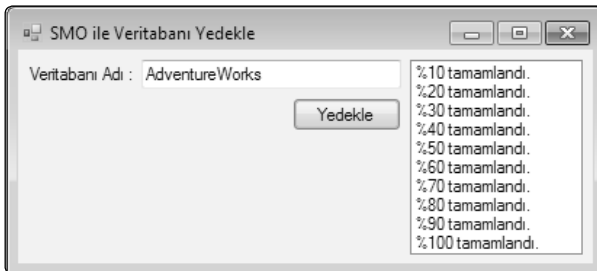
```

Yedekle(txtVeritabaniAd.Text);

```

Yedekleme işlemi için **C:** dizini içerisinde **Backups** klasörünü kullandık. Klasör içerisinde yedek isimleri **Backup** ile başlayacak ve devamında Random ile rastgele bir sayı oluşturularak, program kapatılmadan birden fazla yedekleme işlemi için farklı yedek dosyası isimleri oluşturulmasını sağladık.

AdventureWorks veritabanını yedekleyelim.



Yedeklenen veritabanı dosyası, **C:\Backups** klasöründe bulunmaktadır.

 Backup4660.bak 25.02.2013 19:30 BAK Dosyası 198.772 KB

VERİTABANI GERİ YÜKLEMEK

Yedeklenen veritabanını geri yükleme işlemi de, yedekleme işlemine benzer şekilde SMO ile programlanabilir.

Yedekten geri dönme işlemini gerçekleştirecek bir metod geliştirelim. Bu metod, ekranda bulunan **ProgressBar** ve **ListBox** kontrolleriyle, yükleme işleminin yüzde olarak tamamlanma oranını gösterecektir.

```
void VeritabaniGeriYukle(string vt_ad, string yedek_ad)
{
    Server srv = new Server(conn);
    Restore restore = new Restore();
    string fileName = @"C:\Backups\" + yedek_ad + ".bak";
    string databaseName = "" + vt_ad + "";

    restore.Database = databaseName;
    restore.Action = RestoreActionType.Database;
    restore.Devices.AddDevice(fileName, DeviceType.File);

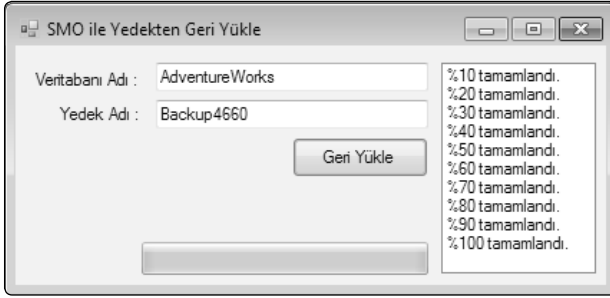
    this.progressBar1.Value = 0;
    this.progressBar1.Maximum = 100;
    this.progressBar1.Value = 10;

    restore.PercentCompleteNotification = 10;
    restore.ReplaceDatabase = true;
    restore.PercentComplete += new
    PercentCompleteEventHandler(res_PercentComplete);
    restore.SqlRestore(srv);
}
```

Yedeğin geri yüklenme işleminin % olarak tamamlanma oranını bir **ListBox'a** yazdıralım.

```
void restore_PercentComplete(object sender, PercentCompleteEventArgs e)
{
    progressBar1.Value = e.Percent;
    listBox1.Items.Add("%" + e.Percent + " tamamlandı.");
}
```

Program arayüzü aşağıdaki gibi görünmektedir.



Bu örnekte **C:** dizinindeki **Backups** klasöründe bulunan **Backup4660.bak** isimli veritabanı yedek dosyasını SQL Server'a geri yükleyerek, **AdventureWorks** isimli yeni bir veritabanı oluşturduk.

VERİTABANINI SİLMEK

SMO ile mevcut veritabanlarını silmek mümkündür.

Veritabanı silmek için, SMO nesneleriyle bir metod geliştirelim.

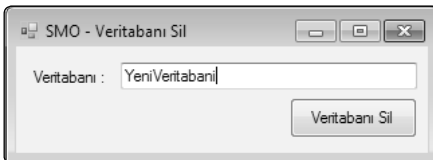
```
void VeritabanıSil(string vt_ad)
{
    Server srv = new Server(conn);
    Database db = srv.Databases["" + vt_ad + ""];
    db.Drop();
}
```

Metod, program içerisinde şu şekilde çağrılabilir.

```
VeritabanıSil(txtVeritabanı.Txt);
```

Veritabanı silmek kritik bir işlem olduğu için, SMO ile geliştirilen programlarda bazı kontrol ve kullanıcı onaylarına tabi tutulmalıdır. Örneğin; veritabanını silme butonuna basıldığında ekrana bir onay kutusu getirerek, kullanıcının bu işlemi gerçekten yapmak isteyip istemediği sorularak, ek bir onay alınabilir.

Daha önce oluşturduğumuz **YeniVeritabanı** isimli veritabanını silelim.



TABLO VE SÜTUNLARI LİSTELEMEK

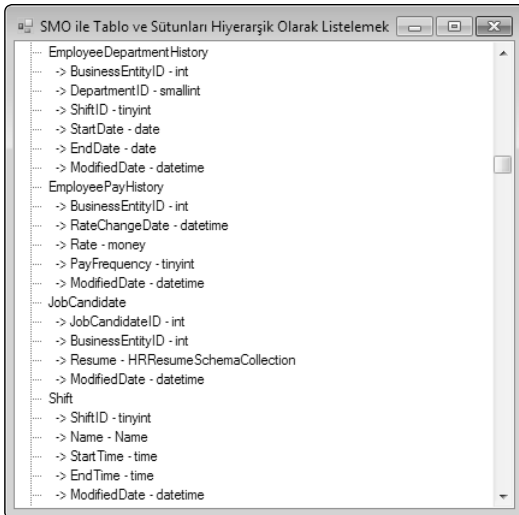
SMO kullanarak, bir veritabanındaki tablo ve her tablo içerisindeki sütunları hiyerarşik bir düzen içerisinde listelemek mümkündür.

Bu işlemi **AdventureWorks** veritabanı için yapalım. **AdventureWorks** veritabanındaki tüm tablo ve tabloların içerisindeki sütunların listesini getirecek **Getir()** adında bir metod oluşturalım.

```
void Getir()
{
    Server srv = new Server(conn);
    Database db = srv.Databases["AdventureWorks"];

    foreach (Table table in db.Tables)
    {
        treeView1.Nodes.Add(" " + table.Name);
        foreach (Column col in table.Columns)
        {
            treeView1.Nodes.Add(" -> " + col.Name + " - "
                                + col.DataType.Name);
        }
    }
}
```

Formun **Load** olayında **Getir()** metodunu çağırdığımızda aşağıdaki gibi, **TreeView** kontrolü içerisinde hiyerarşik bir liste döndürecektir.



VIEW OLUŞTURMAK

SMO nesne modeliyle view nesnesi programlanabilir.

View nesnesi oluşturmak için bir metod geliştirelim.

```
void ViewOlustur(string vt_ad, string view_ad,
    string view_header, string view_body)
{
    try
    {
        Server srv = new Server(conn);
        Database database = srv.Databases["" + vt_ad + ""];

        Microsoft.SqlServer.Management.Smo.View myview = new
        Microsoft.SqlServer.Management.Smo.View(database, "" + view_ad + "");
        myview.TextHeader = "" + view_header + "";
        myview.TextBody = "" + view_body + "";
        myview.Create();
        MessageBox.Show(view_ad + " adındaki view başarıyla oluşturuldu");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```


ViewOlustur isimli bu metod, bir butonun **click** olayında aşağıdaki gibi çağrılabilir.

```
ViewOlustur(txtVeritabaniAd.Text, txtViewAd.Text,
    txtViewHeader.Text, txtViewBody.Text);
```

Programı test etmek için **vw_Urunler** isimli bir view oluşturalım.

The screenshot shows a Windows-style dialog box titled "SMO ile View Oluştur". It contains four text input fields with labels on the left: "Veritabanı Ad:" (Database Name) containing "AdventureWorks", "View Ad:" (View Name) containing "vw_Urunisimleri", "View Başlık:" (View Header) containing "CREATE VIEW vw_Urunisimleri AS", and "View Gövde:" (View Body) containing "SELECT Name FROM Production.Product;". At the bottom right, there is a button labeled "View Oluştur".

vw_Urunler isimli view'i SSMS içerisinde görüntüleyip sorgulayalım.

 **dbo.vw_UrunIsimleri**

```
SELECT * FROM vw_Urunler;
```

STORED PROCEDURE OLUŞTURMAK

SMO nesneleri kullanılarak Stored Procedure geliştirilebilir.

Programsal olarak prosedür oluşturmak için **ProsedurOlustur** isminde bir metod programlayalım.

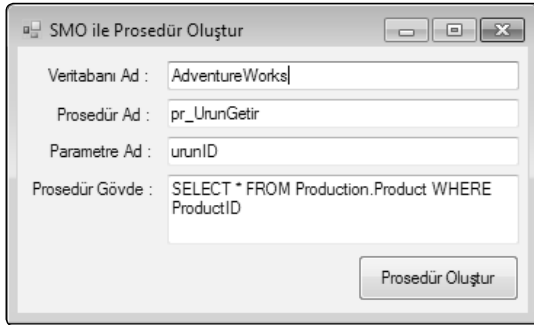
```
void ProsedurOlustur(string vt_ad, string pr_ad, string pr_govde,
string param_ad)
{
    try
    {
        Server srv = new Server(conn);
        Database database = srv.Databases["" + vt_ad + ""];

        StoredProcedure sp = new StoredProcedure(database, "" + pr_ad + "");
        sp.TextMode = false;
        sp.AnsiNullsStatus = false;
        sp.QuotedIdentifierStatus = false;
        StoredProcedureParameter param = new StoredProcedureParameter
            (sp, "@" + param_ad + "", DataType.Int);
        sp.Parameters.Add(param);
        string spBody = "" + pr_govde + " = @" + param_ad + "";
        sp.TextBody = spBody;
        sp.Create();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Hata : " + ex.Message);
    }
}
```

Bu metod, bir butonun **click** olayında aşağıdaki gibi kullanılabilir.

```
ProsedurOlustur(txtVeritabaniAd.Text, txtProsedurAd.Text,
txtProsedurGovde.Text, txtParametreAd.Text);
```

ProductID bilgisine göre ürün bulup getiren bir prosedür oluşturalım.



pr_UrunGetir prosedürünü SSMS ile SQL Server içerisinde test edelim.

```
pr_UrunGetir 1;
```

	ProductID	Name	ProductNumber	MakeFlag	FinishedGoodsFlag	Color	SafetyStockLevel	ReorderPoint	StandardCost	ListPrice
1	1	Adjustable Race	AR-5381	0	0	NULL	1000	750	0,00	0,00

BİR STORED PROCEDURE'Ü OLUŞTURMAK, DEĞİŞTİRMEK VE SİLMEK

Bir prosedürü tek bir transaction içerisinde oluşturma, değiştirme ve silme işlemini programlanabilir.

Bu işlemi gerçekleştirmek için gerekli metodu geliştirelim.

```
void ProsedurDegistir(string vt_ad, string prosedur_ad,
    string prosedur_govde, string param_ad1,
    string param_ad2)
{
    Server srv = new Server(conn);
    srv.ConnectionContext.Connect();
    Database db = srv.Databases["" + vt_ad + ""];

    StoredProcedure sp = new StoredProcedure(db, "" + prosedur_ad + "");

    sp.TextMode = false;
    sp.AnsiNullsStatus = false;
    sp.QuotedIdentifierStatus = false;

    StoredProcedureParameter param = new StoredProcedureParameter(sp,
        "@" + param_ad1 + "", DataType.Int);
```

```

sp.Parameters.Add(param);

StoredProcedureParameter param2 = new StoredProcedureParameter(sp,
    "@" + param_ad2 + "", DataType.NVarChar(50));
param2.IsOutputParameter = true;
sp.Parameters.Add(param2);

string sql = prosedur_govde;
sp.TextBody = sql;

sp.Create();
sp.QuotedIdentifierStatus = true;
listBox1.Items.Add("Prosedür oluşturuldu.");

sp.Alter();
listBox1.Items.Add("Prosedür değiştirildi.");

sp.Drop();
listBox1.Items.Add("Prosedür silindi.");
}

```

Metod, program içerisinde aşağıdaki şekilde kullanılabilir.

```

ProsedurDegistir(txtVeritabaniAd.Text,
    txtProsedurAd.Text,
    txtProsedurGovde.Text,
    txtParametre1Ad.Text,
    txtParametre2Ad.Text);

```

Program çalıştırıldıktan sonraki görünümü aşağıdaki gibi olacaktır.

SMO - Prosedür Oluştur, Değiştir ve Sil

Veritabanı Ad: AdventureWorks

Prosedür Ad: GetLastNameByBusinessEntityID

Parametre 1 Ad: Ret

Parametre 2 Ad: Emp

Prosedür Gövde: SELECT @Ret = (SELECT LastName FROM Person.Person, HumanResources.Employee WHERE Person.Person.BusinessEntityID = HumanResources.Employee.BusinessEntityID AND HumanResources.Employee.BusinessEntityID = @Emp)

Prosedür oluşturuldu.
Prosedür değiştirildi.
Prosedür silindi.

Uygula

VERİTABANINDAKİ TÜM STORED PROCEDURE'LERİ ŞİFRELEMEK

Veritabanı yöneticisinin görevlerinden birisi de, kaynak kod güvenliğini sağlamaktır. Kitabın **Stored Procedure**'ler isimli bölümünde tüm detaylarıyla incelediğimiz güvenlik ve şifreleme işlemlerini SMO ile programsal olarak da yapmak mümkündür.

Bir veritabanındaki tüm prosedürleri tek seferde şifrelemek ya da tek bir prosedürü şifrelemek için SMO kullanılabilir.

Belirtilen veritabanındaki tüm prosedürleri şifreleyecek bir metod geliştirelim.

```
void ProsedurleriSifrele(string vt_ad)
{
    string dbRequested = vt_ad;

    var srv = new Server();
    try
    {
        srv = new Server(conn);
    }
    catch(Exception ex)
    {
        MessageBox.Show("Hata : " + ex.Message);
        Environment.Exit(Environment.ExitCode);
    }

    var db = new Database();
    try
    {
        db = srv.Databases[dbRequested];
        if (db == null)
            throw new Exception();
    }
    catch(Exception ex)
    {
        MessageBox.Show("Hata : " + ex.Message);
        Environment.Exit(Environment.ExitCode);
    }
}
```

```

var sp = new StoredProcedure();
for (int i = 0; i < db.StoredProcedures.Count; i++)
{
    sp = db.StoredProcedures[i];
    if (!sp.IsSystemObject)
    {
        if (!sp.IsEncrypted)
        {
            sp.TextMode = false;
            sp.IsEncrypted = true;
            sp.TextMode = true;
            sp.Alter();
            listBox1.Items.Add(" " + sp.Name + Environment.NewLine);
        }
    }
}
}
}
}

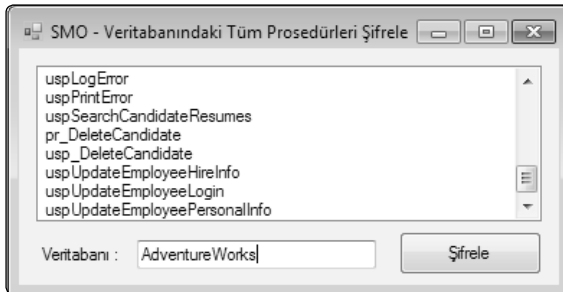
```

Bu metodu program içerisinde kullanmak için, aşağıdaki gibi sadece veritabanı adını vermek yeterlidir.

```
ProsedurleriSifrele("AdventureWorks");
```

Metot, yukarıdaki gibi çalıştırıldığında, **AdventureWorks** veritabanında bu metot ile şifrelenen tüm prosedürlerin isimlerini geri döndürecektir.

Programı, üzerinde bir çok prosedür oluşturarak değiştirdiğim AdventureWorks veritabanı ile test ettiğimde aşağıdaki gibi görünmektedir.



ŞEMA OLUŞTURMAK

SQL Server'da veritabanı yönetiminin önemli yardımcılardan birisi şemalardır. Veritabanındaki nesneleri şemalar içerisinde alarak mantıksal nesne yönetimi gerçekleştirilebilir. SMO ile de şema oluşturmak mümkündür.

Veritabanında yeni bir şema oluşturacak metod geliştirelim.

```

void SemaOlustur(string vt_ad, string sema_ad, string tablo_ad,
                string id_sutun, string sutun_adl)
{
    try
    {
        Server srv = new Server(conn);
        Database database = srv.Databases["" + vt_ad + ""];

        Schema schema = new Schema(database, "" + sema_ad + "");
        schema.Owner = "dbo";
        schema.Create();

        Table Kullanicilar = new Table(database, "" + tablo_ad + "", ""
+ sema_ad + "");

        DataType dt = new DataType(SqlDataType.Int);
        Column IDSutunu = new Column(Kullanicilar, "" + id_sutun + "", dt);
        IDSutunu.Nullable = false;
        IDSutunu.Identity = true;
        IDSutunu.IdentityIncrement = 1;
        IDSutunu.IdentitySeed = 1;
        Kullanicilar.Columns.Add(IDSutunu);
        dt = new DataType(SqlDataType.VarChar, 50);
        Column AdSutunu = new Column(Kullanicilar, "" + sutun_adl + "", dt);
        Kullanicilar.Columns.Add(AdSutunu);
        Kullanicilar.Create();
        MessageBox.Show("Şema ve tablo başarıyla oluşturuldu.");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Hata : " + ex.Message);
    }
}

```

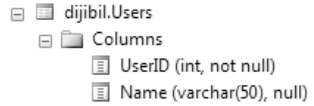
Bu metod program içerisinde aşağıdaki gibi kullanılabilir.

```
SemaOlustur(txtVeritabaniAd.Text,
            txtSemaAd.Text,
            txtTabloAd.Text,
            txtIDSutun.Text,
            txtNormalSutunAd.Text);
```

Programın örnek kullanımı aşağıdaki gibidir.



Şema oluşturulduktan sonra, **dijibil** şeması ile oluşturulan **Users** isimli tablo, **Object Explorer** panelinin **AdventureWorks** tabloları içerisinde görülebilir.



ŞEMALARI LİSTELEMEK

SMO ile bir veritabanındaki tüm şemalar listelenebilir.

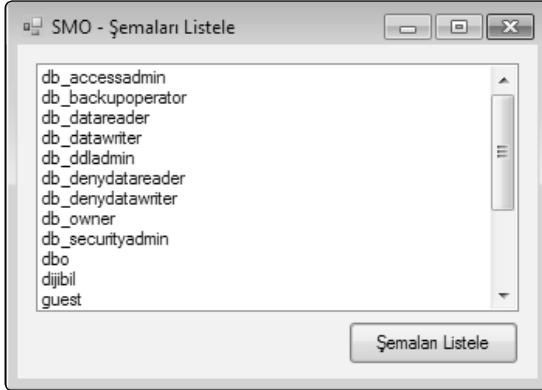
Veritabanındaki tüm şemaları listeleyecek bir metod geliştirelim.

```
void SemalariListele()
{
    Server srv = new Server(conn);
    Database database = srv.Databases["AdventureWorks2012"];
    foreach (Schema schema in database.Schemas)
    {
        lstSemalar.Items.Add(schema.Name);
    }
}
```

Metodun program içerisinde kullanımı aşağıdaki gibi basittir.

```
SemalariListele();
```

Program çalıştırıldıktan sonra aşağıdaki gibi tüm şemaları listeleyecektir.



LINKED SERVER OLUŞTURMAK

Bağlı sunucular ile çalışmak için SMO ile bir **LinkedServer** programlanabilir.

```
Server srv = new Server(@"Instance_A");
LinkedServer lsrv = new LinkedServer(srv, "Instance_B");
LinkedServerLogin login = new LinkedServerLogin();
login.Parent = lsrv;
login.Name = "Login_Instance_A";
login.RemoteUser = "Login_Instance_B";
login.SetRemotePassword("sifre");
login.Impersonate = false;
lsrv.ProductName = "SQL Server";
lsrv.Create();
login.Create();
```

OTURUM OLUŞTURMAK

SMO ile yeni bir oturum (*login*) oluşturulabilir.

Yeni oturum oluşturmak için gerekli metodu geliştirelim.

```

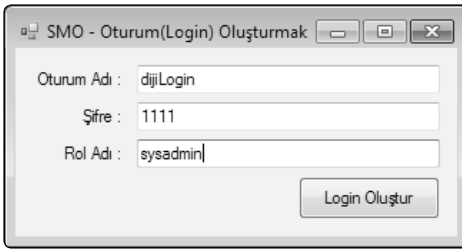
void LoginOlustur(string login_ad, string sifre, string rol_ad)
{
    Server srv = new Server(conn);
    Login login = new Login(srv, "" + login_ad + "");
    login.LoginType = LoginType.SqlLogin;
    login.Create("" + sifre + "");
    login.AddToRole("" + rol_ad + "");
}


```

Metot, program içerisinde aşağıdaki gibi çağrılabilir.

```
LoginOlustur(txtOturumAd.Text, txtSifre.Text, txtRol.Text);
```

dijiLogin adında, **sysadmin** rolüne sahip yeni bir oturum oluşturalım.



SQL Server'da **Logins** bölümü içerisinde **dijiLogin** oturumunu  **dijiLogin** görüntüleyelim.

Login üzerine iki kez tıklayarak özelliklerine girilip, **Server Roles** sekmesinden oturumun **sysadmin** rolüne sahip olduğu görülebilir.

OTURUMLARI LİSTELEMEK

SMO ile SQL Server'daki oturumlar listelenebilir.

SQL Server'daki oturumları ve ilişkiliği olan veritabanları ile kullanıcı adlarının tamamını hiyerarşik olarak listeleyecek metodu geliştirelim.

```

void LoginleriListele()
{
    Server srv = new Server(conn);
    foreach (Login login in srv.Logins)
    {

```



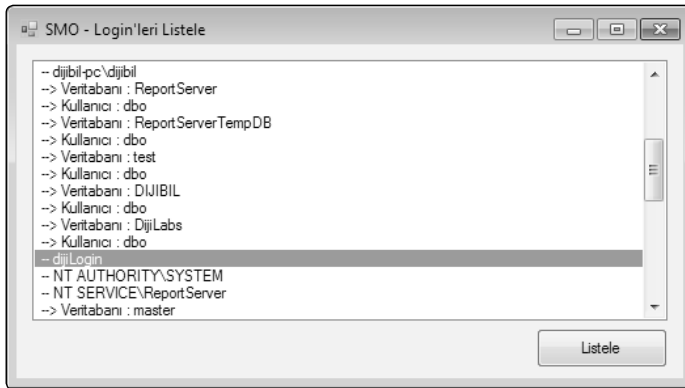
```

lstLoginler.Items.Add(" -- " + login.Name);
if (login.EnumDatabaseMappings() != null)
{
    foreach (DatabaseMapping map in login.EnumDatabaseMappings())
    {
        lstLoginler.Items.Add(" --> Veritabanı : " + map.DBName);
        lstLoginler.Items.Add(" --> Kullanıcı : " + map.UserName);
    }
}
}
}

```

Metot aşağıdaki gibi çağrılabilir.

```
LoginleriListele();
```



KULLANICI OLUŞTURMAK

Bir veritabanında kullanıcı oluşturmak için SMO nesne modeli kullanılabilir.

Kullanıcı oluşturmak için gerekli metodu geliştirelim.

```

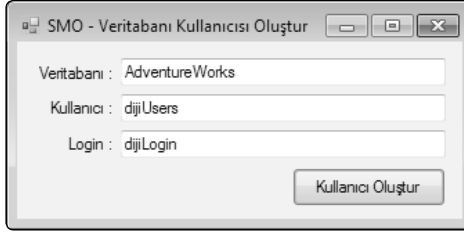
void KullaniciOlustur(string vt_ad, string kullanıcı_ad, string login_ad)
{
    Server srv = new Server(conn);
    Database db = srv.Databases["" + vt_ad + ""];
    User u = new User(db, "" + kullanıcı_ad + "");
    u.Login = "" + login_ad + "";
    u.Create();
}

```

Metot, program içerisinde aşağıdaki gibi çağrılabilir.

```
KullaniciOlustur(txtVeritabani.Text, txtKullanici.Text, txtLogin.Text);
```

AdventureWorks veritabanında yeni bir kullanıcı oluşturalım.



Oluşturduğumuz kullanıcıyı **Management Studio** ile görüntüleyelim.

KULLANICILARI LİSTELEMEK

SMO ile bir veritabanına ait kullanıcıların bilgileri elde edilebilir.

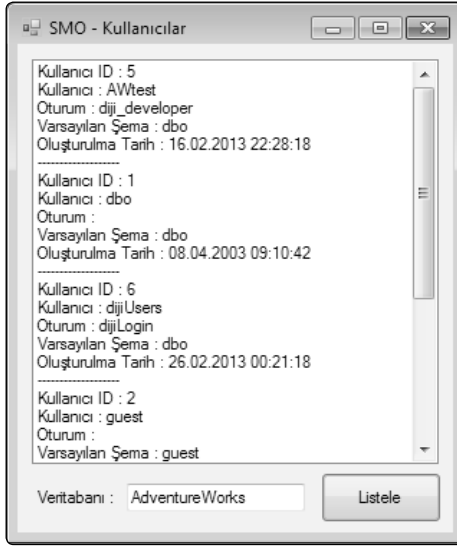
Kullanıcı bilgilerini elde etmek için gerekli metodu geliştirelim.

```
Server srv = new Server(conn);
Database db = srv.Databases["" + vt_ad + ""];
foreach (User user in db.Users)
{
    lstKullanicilar.Items.Add("Kullanıcı ID : " + user.ID);
    lstKullanicilar.Items.Add("Kullanıcı : " + user.Name);
    lstKullanicilar.Items.Add("Oturum : " + user.Login);
    lstKullanicilar.Items.Add("Varsayılan Şema : " + user.DefaultSchema);
    lstKullanicilar.Items.Add("Oluşturulma Tarih : " + user.CreateDate);
    lstKullanicilar.Items.Add("-----");
}
}
```

Metot program içerisinde aşağıdaki gibi çağrılabilir.

```
KullaniciListele(txtVeritabani.Text);
```

AdventureWorks veritabanındaki tüm kullanıcıların bilgilerini listeleyelim.



ROL OLUŞTURMAK

SMO ile veritabanında bir rol programlanabilir.

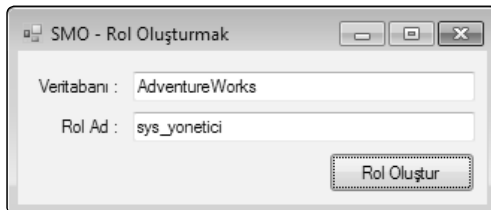
Rol oluşturmak için gerekli metodu geliştirelim.

```
void RolOlustur(string vt_ad, string rol_ad)
{
    Server srv = new Server(conn);
    Database db = srv.Databases["" + vt_ad + ""];
    DatabaseRole dbRole = new DatabaseRole(db, "" + rol_ad + "");
    dbRole.Create();
}
```

Metod, program içerisinde aşağıdaki gibi çağrılabilir.

```
RolOlustur(txtVeritabani.Text, txtRolAd.Text);
```

AdventureWorks veritabanında, **sys_yonetici** adında yeni bir rol oluşturalım.



Oluşturduğumuz rolü **Mangement Studio** ile görüntüleyelim.

db_securityadmin
public
sys_yonetici

ROLLERİ LİSTELEMELİK

SMO ile veritabanındaki rolleri listelenebilir.

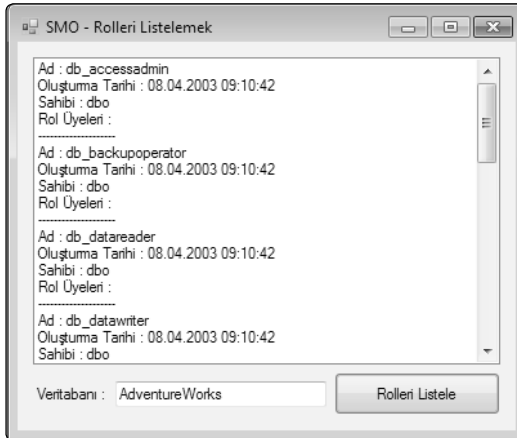
Belirtilen veritabanına ait rolleri ve role bağlı üyeleri listelemek için bir metod geliştirelim.

```
void RollerListele(string vt_ad)
{
    lstRoller.Items.Add("Ad : " + dr.Name);
    lstRoller.Items.Add("Oluşturma Tarihi : " + dr.CreateDate);
    lstRoller.Items.Add("Sahibi : " + dr.Owner);
    lstRoller.Items.Add("Rol Üyeleri :");
    foreach (string s in dr.EnumMembers())
        lstRoller.Items.Add("  " + s);
    lstRoller.Items.Add("-----");
}
```

Metot, program içerisinde aşağıdaki gibi çağrılabilir.

```
RollerListele(txtVeritabani.Text);
```

AdventureWorks veritabanındaki rollerin bilgilerini ve role bağlı üyeleri listeleyelim.



ROL ATAMAK

SMO ile bir kullanıcıya rol ataması yapılabilir.

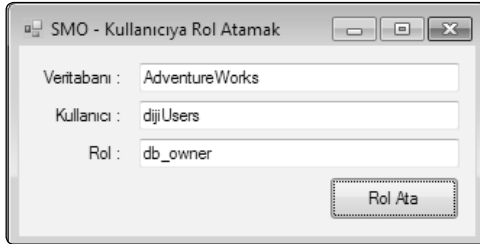
Bir rol ataması gerçekleştirecek metod geliştirelim.

```
void KullaniciRolAtama(string vt_ad, string kullanıcı_ad, string rol)
{
    Server srv = new Server(conn);
    Database db = srv.Databases["" + vt_ad + ""];
    User u = db.Users["" + kullanıcı_ad + ""];
    u.AddToRole("" + rol + "");
    u.Alter();
}
```

Metot, program içerisinde aşağıdaki gibi çağrılabilir.

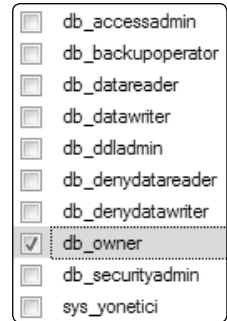
```
KullaniciRolAtama(txtVeritabani.Text, txtKullanici.Text, txtRol.Text);
```

Programı kullanarak bir rol ataması gerçekleştirelim.



Object Explorer panelinden aşağıdaki yol takip edilerek eklenen rol görülebilir.

- **AdventureWorks** veritabanı içerisinde **Security** klasörünü açın.
- Security klasöründe **Users** klasörünü açın.
- dijiUsers isimli kullanıcının üzerine iki kez tıklayarak, **Özellikler** penceresini açın.
- Açılan pencerede Membership bölümüne girin.



ASSEMBLY'LERİ LİSTELEMEK

Veritabanındaki tüm Assembly'leri listelemek için SMO nesne modeli kullanılabilir.

Belirtilen veritabanına ait Assembly'leri listelemek için bir metod geliştirelim.

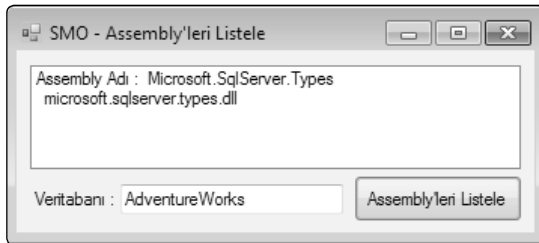
```
void Assemblyler(string vt_ad)
{
    Server srv = new Server(conn);
    Database db = srv.Databases["" + vt_ad + ""];

    foreach (SqlAssembly assembly in db.Assemblies)
    {
        lstAssemblyler.Items.Add("Assembly Adı : " + " " + assembly.Name);
        foreach (SqlAssemblyFile assemblyFile in assembly.SqlAssemblyFiles)
            lstAssemblyler.Items.Add("   " + assemblyFile.Name);
    }
}
```

Metot, program içerisinde aşağıdaki gibi çağrılabilir.

```
Assemblyler(txtVeritabani.Text);
```

AdventureWorks veritabanındaki Assembly'leri listeleyelim.



BİR TABLONUN SCRIPT'İNİ OLUŞTURMAK

Genellikle veritabanı yöneticisinin görevi olan, veritabanı tablolarının SQL script kodlarını saklama ihtiyacı, bazen yazılım geliştiriciler için de gerekli olur. Belirli bir şema içerisindeki belirtilen bir tablonun, SQL script kodlarını elde etmek için SMO nesne modeli kullanılabilir.

Bir tablonun SQL script kodlarını elde edecek metod geliştirilim.

```
string ScriptOlustur(string vt_ad, string tablo_ad, string sema_ad)
{
    Server srv = new Server(conn);
    srv.ConnectionContext.Connect();

    Database db = srv.Databases["" + vt_ad + ""];
    Table tablo = db.Tables["" + tablo_ad + "", "" + sema_ad + ""];
    StringCollection script = tablo.Script();
    string sql_script = string.Empty;

    foreach (string s in script)
    {
        sql_script = sql_script + s;
    }
    return sql_script;
}
```

Metot, program içerisinde aşağıdaki gibi çağrılabilir.

```
ScriptOlustur(txtVeritabani.Text, txtTablo.Text, txtSema.Text);
```

Dikkat ederseniz, **ScriptOlustur** metodu string geri dönüş tipine sahiptir. Metottan geri dönen değer, parametre ile belirtilen tablonun SQL script kodları olacaktır. Biz de bu geri dönen SQL kodlarını bir **TextBox**'a aktararak ekranda görüntüledik.

