

Geliştirme Ortamını Tanımak



Modül 1: Geliştirme Ortamını Tanımk

Geliştirme Ortamını Tanımk

- ◆ Visual C# .NET ile Proje Oluşturmak
- ◆ Proje Bileşenlerini Tanımk
- ◆ Uygulamalarda Hata Ayıklama
- ◆ Uygulamanın Derlenmesi

Bu modülde, Visual Studio .NET ortamını ile tanışacak ve içerisinde kullanılan temel proje bileşenleri hakkında genel bilgiler verilecektir. Ayrıca çalışma zamanı hatalarını yakalamayı ve uygulamayı derlemeyi öğreneceksiniz.

Bu modül tamamlandıktan sonra:

- Proje oluşturabilecek,
- Projeye referans ekleyebilecek,
- Projeye isim alanı ekleyebilecek,
- Proje özelliklerini değiştirebilecek,
- Dinamik yardım alabilecek,
- Proje içerişine görevler ekleyebilecek
- Çalışma zamanı hatalarını yakalayabilecek,
- Uygulamaları derleyebileceksiniz.

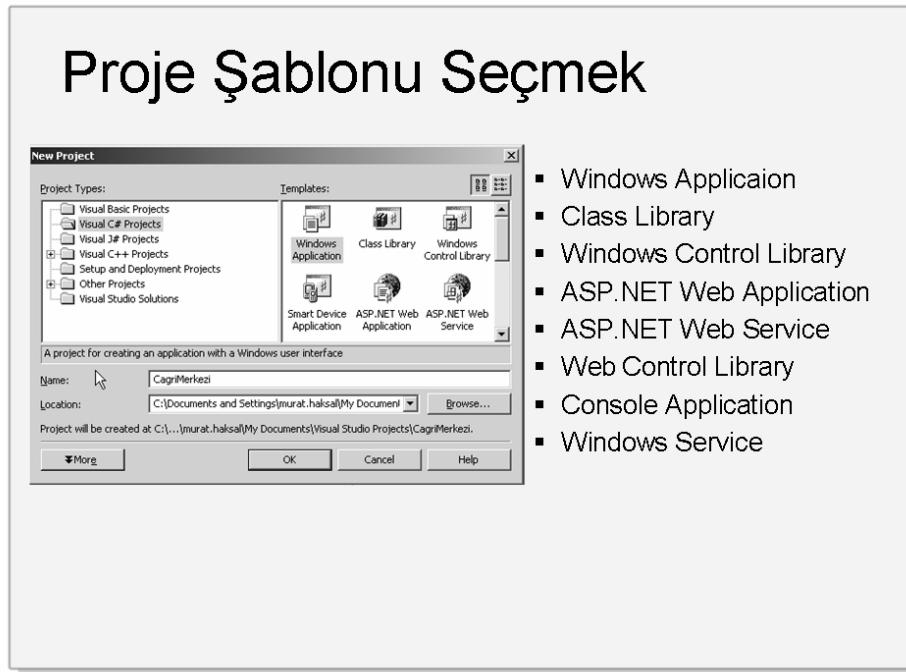
Konu 1: C# .NET ile Proje Oluşturmak

Visual C#.NET ile Proje Oluşturmak

- ◆ Proje Şablonu Seçmek
- ◆ Proje Dosyalarına Genel Bakış
- ◆ Assembly Nedir ?
- ◆ Projeye Referans EklemeK
- ◆ İsim Alanı Nedir ?
- ◆ Yeni İsim Alanı EklemeK
- ◆ Projeye İsim Alanı Dahil Etmek
- ◆ Proje Özelliklerini Ayarlamak

C# ortamı, C#.NET projelerini kolay bir şekilde oluşturma imkanı sağlar. Projenin çalışması için gereken dosyaları otomatik olarak ekler. Projenin geliştirilme aşamasında yeni bileşenlerin eklenmesi, menü ve araç çubukları ile kolay bir şekilde gerçekleştirilebilir.

Proje Şablonu Seçmek



C#.NET ile Windows tabanlı, Web tabanlı gibi çeşitli projeler geliştirilebilir. Bu projeler farklı platformlarda çalışacağı veya farklı amaçlara yönelik oluşturulacağı için, başlangıç bileşenleri farklılık gösterecektir. Örneğin Windows tabanlı projeler için Windows formlarının kullanılması ve bazı referansların eklenmesi gereklidir. Visual Studio ortamının sağladığı şablonlar, proje dosyalarının başlangıç kodlarını otomatik olarak yazıp gerekli referansları ekleyerek geliştiriciye hızlı bir başlangıç sağlar.

- **Windows Application**

Windows tabanlı uygulamalar geliştirmek için kullanılır.

- **Class Library**

Diğer projeler için **Class** kütüphaneleri sağlayan DLL (Dynamic Link Library) oluşturmak için kullanılır. Bu bileşenler projelere **Reference** olarak eklenerek tekrar kullanılır.

- **Windows Control Library**

Kullanıcı tanımlı Windows kontrolleri oluşturmak için kullanılır. Bu kontroller Windows uygulamalarında, birçok formda tekrar kullanılmak üzere tasarlanır.

- **Smart Device Application**

Mobil cihazlar üzerinde uygulama geliştirmek için kullanılır.

- **ASP.NET Web Application**

IIS (Internet Information Services) üzerinde çalışacak Web uygulamaları geliştirmek için kullanılır.

- **ASP.NET Web Service**

Web uygulamalarına XML Web Service sağlayan projeler geliştirmek için kullanılır. Oluşturulan bu projeler, diğer uygulamalara **Web Reference** olarak eklenir.

- **Web Control Library**

Web uygulamalarında, kullanıcı tanımlı kontroller oluşturmak için kullanılır.

- **Console Application**

Komut penceresinde çalışacak konsol uygulamaları geliştirmek için kullanılır.

- **Windows Service**

Windows altında sürekli çalışan uygulamalar için kullanılır. Bu uygulamalar, kullanıcıların sisteme giriş yapmadığı durumlarda da çalışmaya devam eder.

- **Other Projects**

Enterprise Applications (Şirket uygulamaları), Deployment Projects (Yükleme projeleri), Database Projects (Veritabanı projeleri) gibi değişik şablonlardır.

- **Empty Project**

Herhangi bir şablon uygulanmadan açılan Windows projelerdir. Başlangıç nesnesi ve referanslar eklenmez.

- **Empty Web Project**

Herhangi bir şablon uygulanmadan açılan Web projelerdir. Bu proje IIS üzerinde tanımlanır ancak form ve referans nesneleri eklenmez.

- **Blank Solution**

Başlangıç olarak bir proje açılmaz. Boş bir **solution** dosyası açılır. İstenen projeler, **Add New Project** komutu ile bu **solution** içine dâhil edilir.

Visual Studio ile yeni bir proje birkaç adımda oluşturulabilir.

1. **File** menüsünden **New** alt menüsüne işaret edin ve **Project** komutunu seçin.
2. "New Project" penceresinden **Visual C# Projects** tipini ve çalışmak istediğiniz şablonunu seçin.
3. **Name** özelliğine projeye vereceğiniz ismi yazın.
4. **Location** özelliği projenin dosyalarının bulunacağı yeri belirler. **Browse** düğmesine basarak Windows dizinine ulaşın ve projenin yerini seçin.
5. **More** düğmesine basıldığı zaman, **solution** dosyası için yeni bir isim kullanılması ve ayrı bir klasör açılmasını sağlayan panel görüntülenir. **Solution** için farklı bir isim vermek için **Create directory for Solution** seçeneğini işaretleyin ve metin kutusuna **solution** için yeni bir isim yazın.
6. **OK** tuşuna basıldığı zaman proje açılır. **solution** için ayrı bir klasör seçilmemişse, proje dosyaları proje ismi ile oluşturulan klasör altında oluşturulur.

Proje Dosyalarına Genel Bakış

Proje Dosyalarına Genel Bakış

- Solution Dosyaları (.sln, .suo)
- Project Dosyaları (.csproj, .csproj.user)
- Yerel Proje Dosyaları (.cs)
- Web Projeleri Dosyaları (.aspx, .asmx, .asax)

Visual C# .NET ile oluşturululan bir projenin çalışması için gereken bazı dosyalar vardır. Bu dosyaların birçoğu, projenin tipine göre farklılık gösterir. Yeni bir proje açıldığında, projeye verilen isim ile bir klasör açılır ve proje dosyaları bu klasör altına yerleştirilir.

- **Solution** Dosyaları (.sln, .suo)

Visual C# .NET projeleri bir **solution** dosyası (.sln) altında oluşturulur. **solution** dosyası farklı projeleri bir arada tutar ve birden fazla projeyi içerisinde barındırır. Visual Studio ile proje oluşturulurken **solution** dosyası otomatik olarak eklenir.

Solution User Option (.suo) dosyaları, kullanıcının **solution** ile çalışırken yaptığı ayarları tutar ve proje tekrar açıldığı zaman bu ayarları getirir.

- **Project** Dosyaları (.csproj, .csproj.user)

Bir projenin içinde bulunan bileşenlerin, eklenen referansların tutulduğu proje dosyasıdır. Visual C# projeleri .csproj uzantılı dosya ile oluşturulur. Bu dosya aynı zamanda, bir **solution** içinde farklı dilde ve tipteki projeleri ayırt etmek için kullanılır. Projeye özgü ayarlar ise .csproj.user dosyasında tutulur.

- Yerel Proje Dosyaları (.cs)

Form, **class** gibi bileşenlerin tutulduğu dosyalardır. Bir .cs uzantılı dosya içinde birden **class** tutulabilir. Ancak projedeki her form için ayrı bir .cs dosyası oluşturulur.

- Web Projeleri Dosyaları (.aspx, .asmx, .asax)

Web uygulamalarında oluşturulan dosyalar Web sunucusunda (ISS) tutulur. Bu dosyalar web formları için .aspx, Web Service için .asmx, global sınıfı için .asax uzantısına sahiptir.

Proje oluşturuluktan sonra yeni nesnelerin eklenmesi **Project** menüsü ile ya da **Solution Explorer** paneli kullanımı ile gerçekleşir. Project menüsünden yeni bir **form**, **module**, **class**, **component**, **user control** eklemek için ilgili menü komutu seçilebilir. **Add New Item** komutu ile farklı tipte birçok dosya projeye dahil edilebilir.

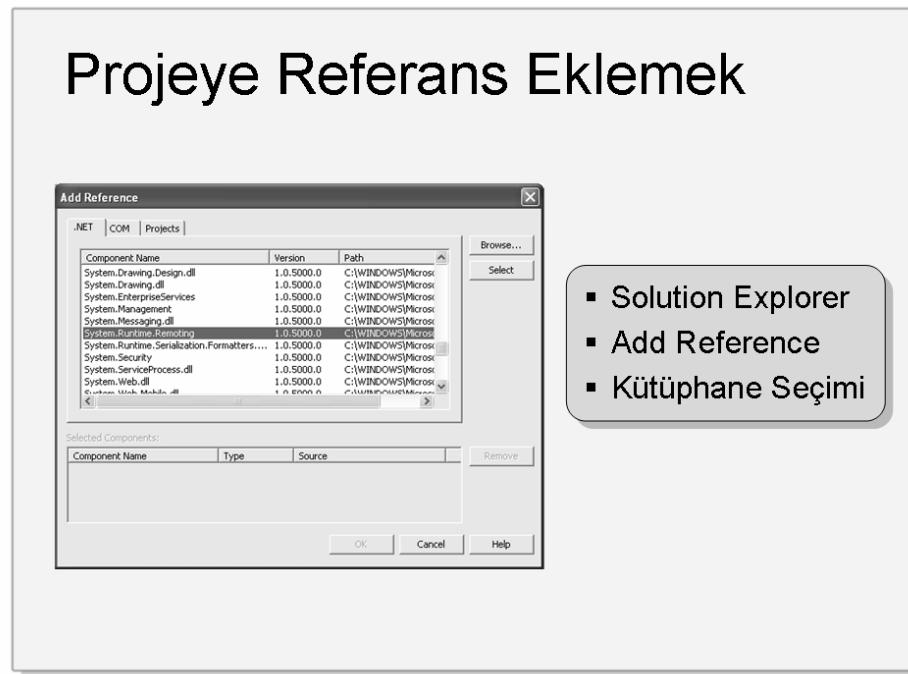
Assembly Nedir?

- ### Assembly Nedir ?
- Dosyaya ait başlık, açıklama ve telif hakkı gibi kritik bilgiler.
 - Farklı kişiler tarafından geliştirilmiş Assembly'ler farklı projelere eklenebilir.

Visual Studio .NET ortamında geliştirilen uygulamalar derlendiğinde, .exe veya .dll uzantılı dosyalar oluşur. .NET'in otomatik olarak oluşturduğu bu dosyalara assembly denir. Assembly içerisinde dosyaya ait başlık, açıklama ve telif hakkı gibi kritik bilgiler tutulur.

Visual Studio .Net içerisinde geliştirilen bir projeye, farklı kişiler tarafından geliştirilmiş assembly'ler eklenebilir. Özellikle gelişmiş projelerde assembly'ler ayrı programcılar tarafından yazılarak ortak bir proje altında toplanabilir.

Projeye Referans Eklemek



Herhangi bir projenin içerisinde, bileşen kütüphanelerinin eklenmesi için kullanılır. Bu bileşen kütüphaneleri, .NET ve COM bileşenlerinden oluşur.

Projeye referans eklemek için belirtilen adımlar takip edin.

1. Solution Explorer penceresinden **References** menüsü seçin. **References** menüsüne sağ tıklayın.
2. Açılan menüden **Add Reference** komutunu verin.
3. Açılan pencere üzerinden .NET, COM, Projects sekmelerinden herhangi birini seçin.
 - .NET, projeye NET bileşen kütüphanelerini eklemek için kullanılır.
 - COM, projeye COM bileşen kütüphanelerini eklemek için kullanılır.
 - Projects, proje ile aynı **solution** içerisinde yer alan bileşen kütüphanelerini eklemek için kullanılır.
4. Eklenecek assembly nesnesini seçin ve **Select** düğmesini tıklayın. Birden fazla assembly seçmek için aynı işlemi tekrar edin.
5. Referans ekleme işlemini tamamlamak için **OK** düğmesine tıklayın.

Proje şablonları içerisinde en çok kullanılan referans'lar şunlardır:

Projeye Referans Eklemek

- System
- System.Data
- System.Drawing
- System.Windows.Forms
- System.Xml

- **System:** Programın çalışması için gerekli en temel referanstır. **System.dll** kütüphanesi içerisinde tutulur.
- **System.Data:** Veritabanı bağlantılarının yapılması için gerekli referanstır. **System.Data.dll** kütüphanesi içerisinde tutulur.
- **System.Drawing, System.Windows.Forms:** Windows form ve kontrollerini içeren referanstır. **System.Drawing.dll** ve **System.Windows.Forms.dll** kütüphaneleri içerisinde tutulur.
- **System.XML:** XML teknolojisinin kullanılmasını sağlayan referanstır. **System.XML.dll** kütüphanesi içerisinde tutulur.

İsim Alanı (Namespace) Nedir?

İsim Alanı Nedir ?

Sınıflar, Arayüzler ve Modüller İsim Alanları kullanılarak gruplandırılır.

System.Data.dll içerisinde;

- System.Data
- System.Data.Common
- System.Data.SqlClient
- System.Data.OleDb
- System.Data.SqlTypes
- System.Xml

.NET içerisindeki tüm kütüphaneler, .NET Framework ismi verilen ortak çatı altında toplanır. Bu çatı altındaki tüm kütüphaneler amaçlarına göre namespace denilen isim alanı altında gruplandırılır. Bu isim alanı içerisinde sınıflar, ara yüzler ve modüller bulunur.

.NET içerisinde veritabanı uygulamaları geliştirmek için **System.Data.dll** kütüphanesine ihtiyaç duyulur. Bu kütüphane Visual Studio .NET içerisindeki tüm proje şablonlarında otomatik olarak yer alır. **System.Data.dll** kütüphanesi içerisinde;

- **System.Data**
- **System.Data.Common**
- **System.Data.SqlClient**
- **System.Data.OleDb**
- **System.Data.SqlTypes**
- **System.Xml**

isim alanları yer alır.

Yeni İsim Alanı EklemeK

Yeni İsim Alanı EklemeK

Yeni isim alanı oluşturmak için;

```
namespace Isimalani_ismi
{
    ...
}
```

söz dizimi kullanılır.

```
namespace Egitim
{
    public class Grup
    {
        ...
    }
}
```

Yeni isim alanı oluşturmak için **namespace** anahtar kelimesi kullanılır.

```
namespace Isimalani_ismi
{
    ...
}
```

Örnekte **NSBilgeAdam** isminde bir isim alanı tanımlanmıştır. Bu isim alanı içerisinde **Egitim** ve **Ogrenci** isminde sınıflar eklenmiştir.

```
namespace NSBilgeAdam
{
    // BilgeAdam isim alanında kullanılacak
    //Sınıf, Modül ve Arayüzler tanımlanır

    class Egitim
    {
        //...
    }
    class Ogrenci
    {
        //...
    }
}

// vs...
}
```

BilgeAdam isim alanı içindeki **Ogrenci** sınıfını kullanmak için, sınıf ismini, isim alanı ile birlikte belirtilmelidir.

```
bilgeadam.Nsbilgeadam.Ogrenci yeniogrenci;
yeniogrenci = new bilgeadam.Nsbilgeadam.Ogrenci() ;
```

UYARI: Proje ile aynı isimdeki bir isim alanı .NET derleyicisi tarafından yeni oluşturululan tüm projelere eklenir. Bu genel isim alanına kök isim alanı (root namespace) denir. Dolayısıyla kendi oluşturulan isim alanlarını, kök isim alanını ile birlikte belirtilmelidir.

Herhangi bir isim alanı içerisinde birden fazla isim alanı tanımlanabilir. Örnekte **NSBilgeAdam** isim alanı içerisinde **Idari**, **Egitim** ve **Ogrenci** adında üç ayrı isim alanı eklenmiştir.

```
namespace NSBilgeAdam
{
    // BilgeAdam isim alanında kullanılacak
    // Class, Module ve Interface'ler tanımlanır
    namespace Idari
    {
        class Personel
        {
        }
    }
    namespace Egitim
    {
        class Grup
        {

        }
    }
    namespace Ogrenci
    {
        class Bilgi
        {

        }
    }
    // vs...
}
```

Projeye İsim Alanı Dâhil Etmek

Projeye İsim Alanı Dahil Etmek

```
using System;
namespace Egitim
{
    public class Grup
    {
        ...
    }
}

using Egitim;

Egitim.Grup egtGrup = new Egitim.Grup();
```

Bir isim alanı içerisinde yer alan sınıfları tanımlamak için, sınıfın bulunduğu kütüphanenin yolunu eksiksiz olarak belirtmek gerekir. Ancak bu şekilde kullanımlar, kodun okunmasını oldukça zorlaştırır. Örnekte sınıflar bu yöntemle tanımlanmıştır.

```
bilgeadam.Nsbilgeadam.Idari.Personel kisi1;
kisi1 = new bilgeadam.Nsbilgeadam.Idari.Personel();
bilgeadam.Nsbilgeadam.Ogrenci.Bilgi ogrencibilgi;
ogrencibilgi = new bilgeadam.Nsbilgeadam.Ogrenci.Bilgi();
```

Her sınıf için kütüphane yolunun tekrarını ortadan kaldırmak için, **using** anahtar sözcüğü kullanılır. **using** sözcüğü ile eklenen isim alanlarının nesnelerine, proje içerisinde doğrudan erişilebilir.

Örnekte **NSBilgeAdam** isim alanının projeye dâhil edilmesi gösterilmektedir.

```
using bilgeadam.Nsbilgeadam;
```

NSBilgeAdam isim alanında bulunan bir sınıfı kullanmak için sadece ismini yazmak yeterli olacaktır.

```
Ogrenci.Bilgi ogrencibilgi = new ogrenci.Bilgi();
```

İç içe isim alanının kullanımında, içteki isim alanına kolayca erişmek için kısaltmalar kullanılabilir. Örnekte, **NSBilgeAdam** isim alanı içerisindeki **Ogrenci** isim alanına erişim gösterilmektedir

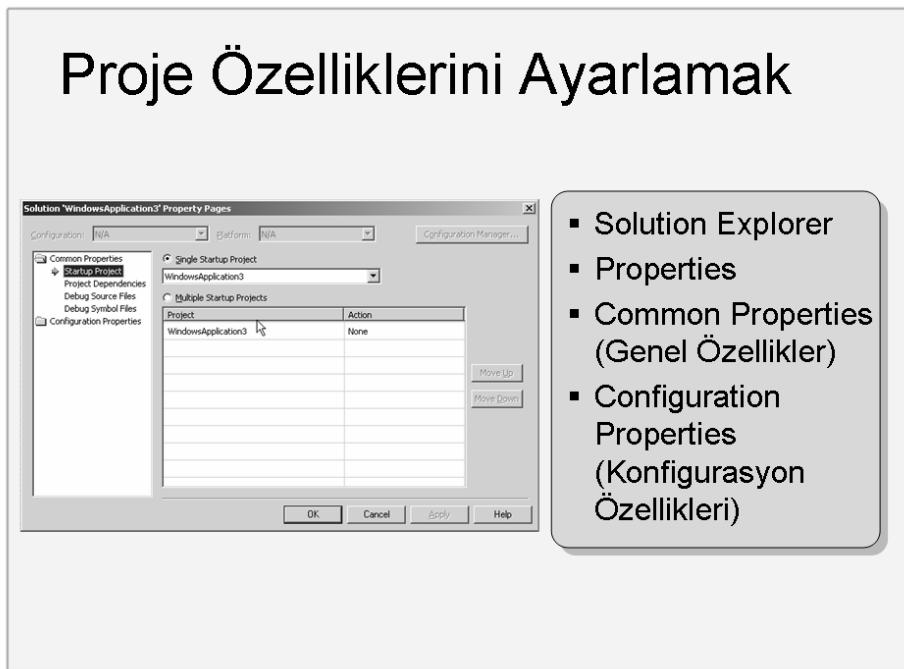
```

using ogr = bilgeadam.NSBilgeadam.Ogrenci;

Public Class Form1:System.Windows.Forms.Form
{
    // ...
    ogr.BilgiogrBilgi = new ogr.Bilgi();
}

```

Proje Özelliklerini Ayarlamak



- Solution Explorer
- Properties
- Common Properties (Genel Özellikler)
- Configuration Properties (Konfigürasyon Özellikleri)

Projenin genel davranışlarını ve konfigürasyon özelliklerini değiştirmek için **Property Page** penceresi kullanılır.

Proje özelliklerini değiştirmek için belirtilen adımları takip edin.

1. Proje isminin üzerinde farenin sağ butonunu tıklayın.
2. Açılan menüden **Properties** komutunu verin.
3. Açılan **Property Page** penceresi üzerinde **Common Properties** (Genel Özellikler) ve **Configuration Properties** (Konfigürasyon Özellikleri) sekmelarından herhangi birini seçin.
4. Genel Özellikler, projenin genel davranışlarını değiştirmek için kullanılır.
5. Konfigürasyon Özellikleri, Hata ayıklama ve Derleme seçeneklerinin değiştirilmesi için kullanılır.
6. Proje özelliğini değiştirdikten sonra **OK** butonunu tıklayın.

En çok kullanılan proje özellikleri şunlardır:

Proje Özelliklerini Ayırmak

Genel Özellikler

- Assembly Name - Dosya İsmi
- Root Namespace - Kök İsim Alanı
- Project Output Type - Uygulama Türü
- Startup Object - Başlangıç Nesnesi

Konfigurasyon Özellikleri

- Debug, Build - Hata Ayıklama ve Derleme

- **Assembly Name:** Derlenen uygulamanın exe veya .dll uzantılı çıktı dosyasının adını belirler.
- **Root Namespace:** Kök isim alanını belirler. Varsayılan olarak projenin ismi gelir.
- **Project Output Type:** Derlenen uygulamanın hangi tipte assembly oluşturacağını belirler. Bu tipler Windows, konsol uygulamaları ya da sınıf kütüphaneleri (.dll) olabilir.
- **Startup Object:** Uygulamanın hangi formdan veya modülüden çalışmaya başlayacağını belirtir.

Konu 2 : Proje Bileşenlerini Tanımk

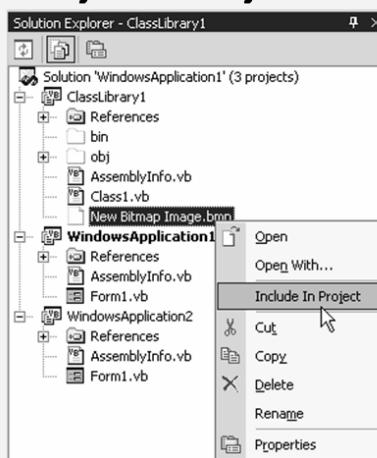
Proje Bileşenlerini Tanımk

- Solution Explorer Kullanmak
- Object Browser Kullanmak
- Server Explorer Kullanmak
- Dinamik Yardım Almak
- Görev Listesini Kullanmak

Solution Explorer Kullanmak

Solution Explorer Kullanmak

Proje ve Proje Elemanlarını yönetme



Solution Explorer paneli, bir **solution** içindeki tüm dosyaları görüntüler. **Solution** içinde birden fazla proje bulunıldığı için, bu projeler sıralı bir

şekilde listelenir. Koyu renkle gösterilen proje, **solution** içinde ilk çalıştırılacak projedir.

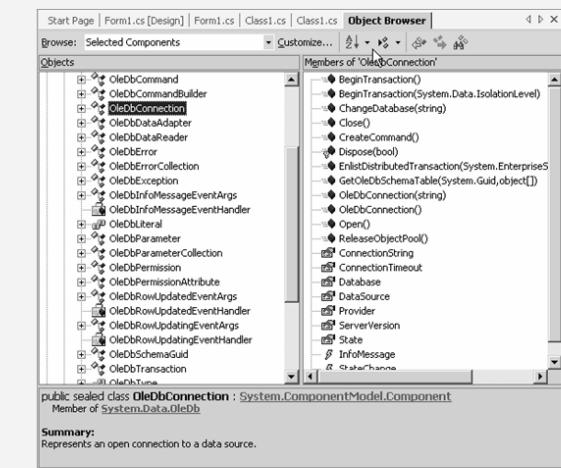
Bu panel ile **solution** içine proje ekleme – silme, projelere yeni nesne ekleme – silme işlemleri gerçekleştirilebilir.

Panelin üst tarafında bulunan araç çubuğu, dosyalar üzerinde bazı işlemlerin gerçekleştirilmesi için kısa yollar sunar. Örneğin araç çubuğundan **Show All Files** komutu seçildiği zaman, projelerin bulunduğu klasördeki tüm dosyalar gösterilir. **Solution Explorer** panelinde beyaz ile gösterilen nesneler projeye dahil edilmemiştir. Örneğin, proje klasöründe bulunan bir resim dosyasını projeye dahil etmek için, resme sağ tıklayıp **Include In Project** komutu verilmelidir.

Solution Explorer panelini görüntülemek için **View** menüsünden **Solution Explorer** komutunu verin.

Object Browser Kullanmak

Object Browser Kullanmak Nesneleri İzlemek



Object Browser, Visual Studio .NET içerisindeki kütüphane ve isim alanlarını tüm alt öğeleriyle beraber hiyerarşik şekilde listeler.

Object Browser’ı görüntülemek için, **View** Penceresinden **Object Browser** komutunu verin.

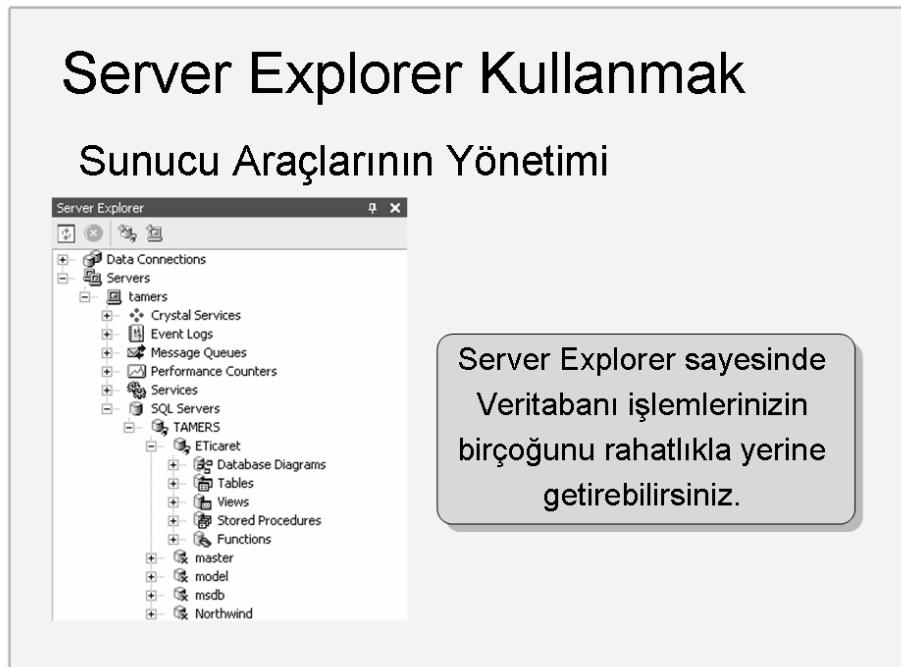
Object Browser pencerenin sol üst köşesinde **Browse** alanı **Selected Components** seçeneği ile birlikte varsayılan olarak görünür. Bu seçenek ile

projeye dâhil edilen referanslar ve bu referanslarla ilişkili isim alanları hiyerarşik bir şekilde listelenir.

Objects paneli içerisindeki seçenek herhangi bir isim alanı genişletilirse, içindeki tüm öğeler hiyerarşik şekilde listelenir. Bu öğelerin herhangi biri seçildiğinde, o öğeye ait tüm alt öğeler Members penceresinde listelenir.

Objects penceresinin sağ alt köşesinde ise, seçilen öğenin tanımını ve hangi isim alanının altında olduğu gösterilir.

Server Explorer Kullanmak



Server Explorer, Visual Studio .NET ortamı içerisinde veri sağlayıcılarla çalışmayı kolaylaştırmak için tasarlanmış bir araçtır. Ayrıca **Server Explorer** sunucu makine bileşenlerinin yönetimi ve kullanımını sağlar.

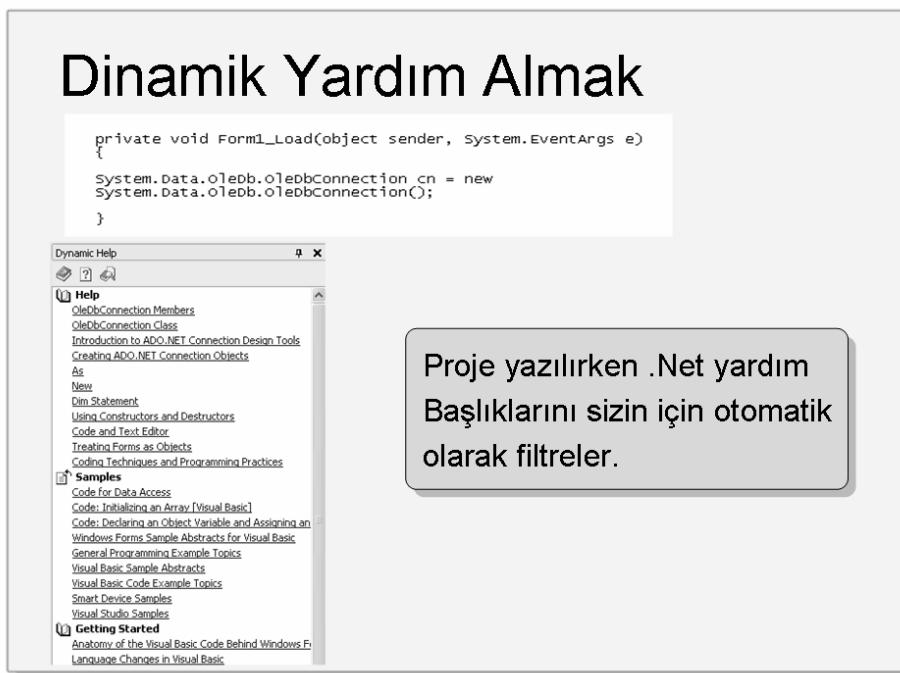
Server Explorer, **Data Connections** ve **Servers** olmak üzere iki sekmeden oluşur. Veri sağlayıcıları ile çalışmak için **Data Connections** seçenekleri kullanılır.

Yeni bir veri sağlayıcı oluşturmak için belirtilen adımları takip edin.

1. **Server Explorer** üzerinden **Data Connections** seçenekini seçin.
2. **Data Connections** seçenekü üzerinde farenin sağ butonunu tıklayın. Açılan menüden **Add Connection** komutunu verin.
3. Açılan **Data Link Properties** penceresinden bağlantı oluşturulur.

Servers sekmesinin altındaki **SQL Servers** menüsünü kullanarak, veritabanı işlemleri yerine getirilebilir ve veritabanı nesneleri, sürekle bırak metodu ile form üzerine sürüklenebilir.

Dinamik Yardım Almak



Visual Studio .NET, içerisinde çok fazla konuyu barındırduğu için tümüne hakim olmak neredeyse imkânsızdır. Bu nedenle yazılım geliştiricilerin işini kolaylaştırmak için, Visual Studio .NET içerisinde dinamik yardım kütüphanesi oluşturulmuştur. Dinamik yardım, uygulama geliştirirken yazılan koda göre tüm yardım konularını listeler.

İPUCU: Dinamik yardımı aktif hale getirmek için, "Help" menüsünden "Dynamic Help" komutunu tıklayın.

Görev Listesini Kullanmak

Görev Listesini Kullanmak

- Proje görevlerini yönetmek
- Yeni görev eklemek
- Tamamlanan görevi işaretlemek

Görev Listesi, aktif proje içeresine görev eklemek için kullanılır. Bu görevler uygulama gelişiminin takip edilmesini sağlar. Görev Listesi içersine eklenen tüm görevleri önem sırasına göre sıralanabilir.

Görev Listesi aracını proje ortamında aktif hale getirmek için **View** menüsünün **Other windows** alt menüsünden **Task List** komutunu tıklayın.

Görev Listesi aracı üzerinde **Click here to add a new task** alanı tıklanarak yeni görev eklenebilir. Biten görevin önündeki onay kutusu tıklanarak, görev sonlandırılabilir.

Konu 3: Uygulamalarda Hata Ayıklama

Uygulamalarda Hata Ayıklama

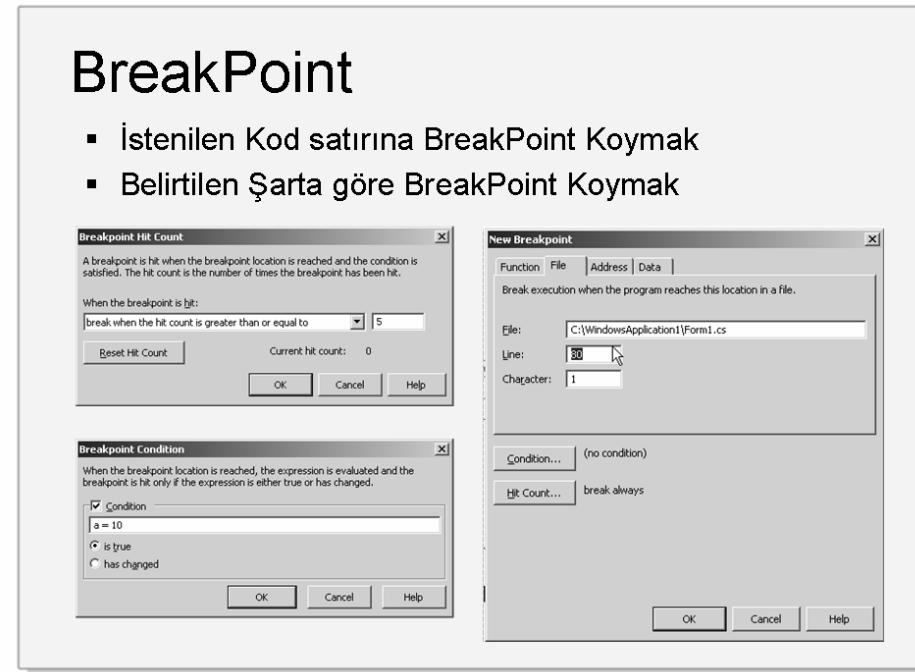
- BreakPoint
- Debug Panelleri
- Command Panelini Kullanmak

Uygulamaların geliştirme sırasında birçok hata ile karşılaşılır. Bu hataların çoğu çalışma zamanında ortaya çıktıgı için, kodun yazılması sırasında hatanın kaynağının anlaşılması zordur. Hata üreten kod satırını, hataların nedenini anlamak için Visual Studio **Debug** (Hata ayıklama) aracı kullanılır.

Visual Studio **Debug** aracı

- Kodlar arasına **BreakPoint** konarak, çalışmanın istenen satırda durmasını,
- Kodlar arasında ilerlerken **Debug** panelleri ile değişkenlerin değerlerinin gözlenmesini,
- **Command** paneli ile çalışma anında komut çalıştırılmasını, değişkenlerin değerlerinin değiştirilmesini sağlar.

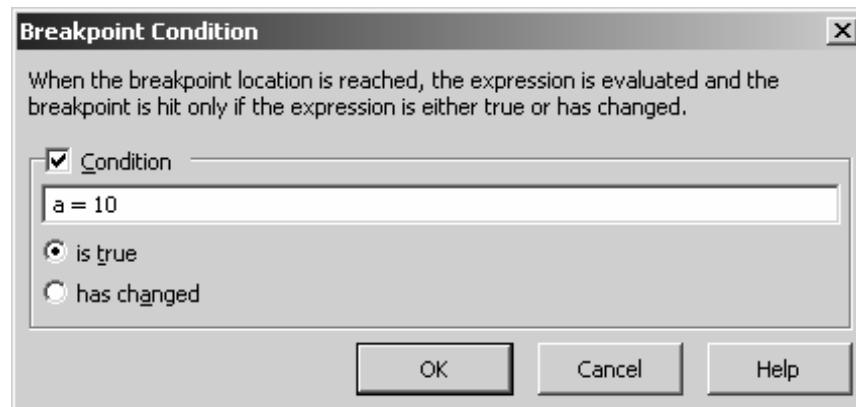
BreakPoint



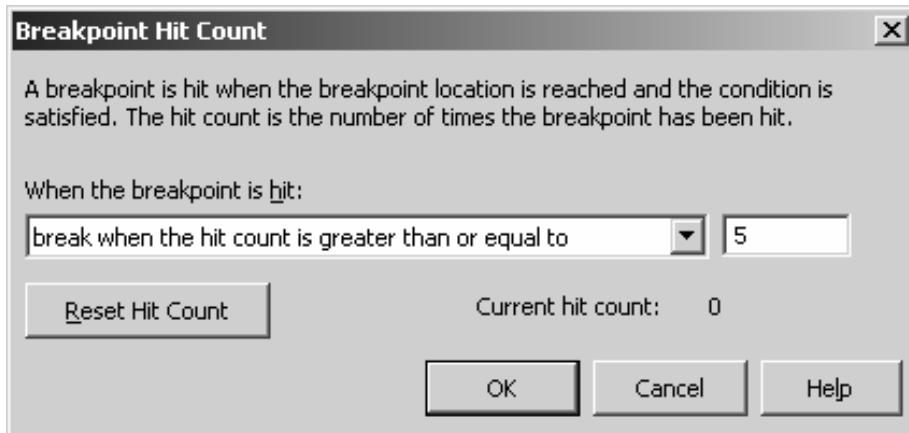
BreakPoint kullanımı, uygulamanın çalışmasının istenen kod satırında durdurulmasını sağlar. Çalışma, bir şartın gerçekleştiği durumda da durdurulabilir. Örneğin bir değişkenin, belli bir değeri aldığı kod satırında uygulamanın durması istenebilir.

İstenen bir kod satırına **BreakPoint** koymak için, kod sayfasının sol tarafında bulunan panele tıklanır ya da **F9** tuşuna basılır.

Belirtilen bir şart gerçekleştiğinden sonra çalışmanın durması isteniyorsa, **Breakpoint** üzerine sağ tıklanıp **BreakPoint Properties** komutu verilmelidir. Çıkan pencerede **Condition** düğmesine basılarak **BreakPoint Condition** penceresi açılır. Bu pencerede bir değişkenin istenen bir değeri aldıktan sonra çalışmanın durması belirtilir.



Çalışmanın, şartın belli bir sayı kadar sağlandığı zaman durdurulması için, **BreakPoint Properties** penceresinde **Hit Count** düğmesine basılır. **BreakPoint Hit Count** penceresinde, şartın gerçekleşme sayısı girilir. Örnekte, **BreakPoint** beş defa veya daha fazla ulaşıldığı zaman durulması belirtilir.



Debug Panelleri

Debug Panelleri

Hata ayıklamak için kullanılır

- Autos
- Local
- Watch

Çalışma durdurulduktan sonra, değişkenlerin o andaki durumları **Debug** panelleri ile gözlemlenir. Bu paneller ancak hata ayıklama sırasında kullanılabilir. **Debug** panelleri, **Debug** menüsü altında **Windows** menüsünden seçilebilir.

- **Autos**

Çalışmakta olan satırla, bir önceki ve bir sonraki arasında kalan değişkenleri listeler.

- **Locals**

Çalışılan kapsam içindeki tüm değişkenleri listeler. Bu kapsam bir modül, yordam veya döngü olabilir.

- **Watch**

Değeri incelenmek istenen değişken veya özellikler, bu panele yazılarak eklenir.

Çalışma durdurulduktan sonra kodlar arasında ilerlemek gereklidir. Kodlar arasında ilerlemenin, yordamların içine girilmesi, üzerinden atlanması gibi birçok yol vardır.

1. **Step Into**

Çalıştırılan kod eğer bir yordam veya fonksiyon ise bu yordam veya fonksiyonun içine girilir ve hata ayıklamaya devam edilir.

2. **Step Over**

Bir yordam veya fonksiyon içine girilmeden ilerlenir.

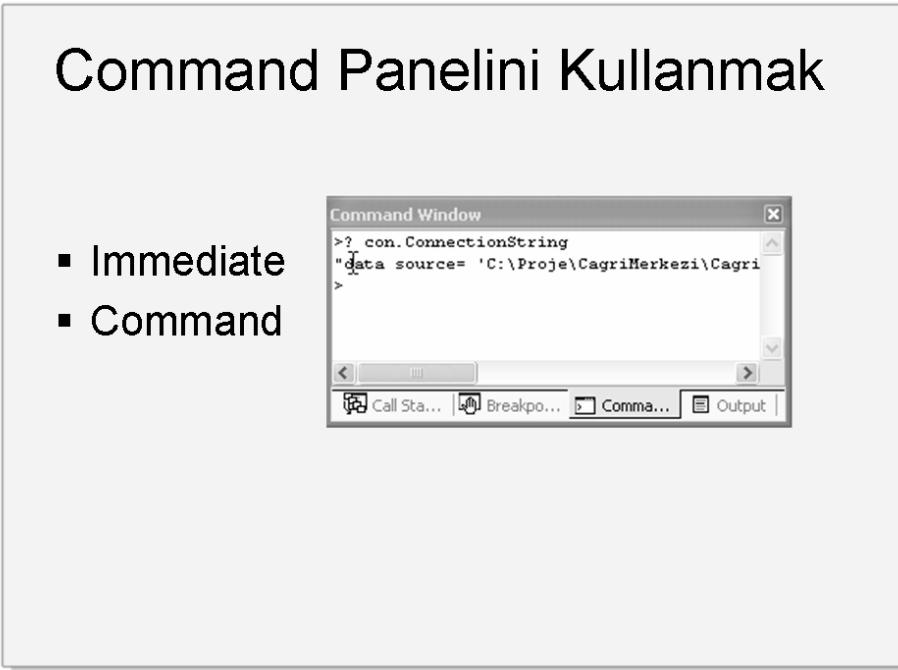
3. **Step Out**

Bir yordam veya fonksiyon içinde ilerleniyorsa, buradan çıkışarak yordam veya fonksiyonun çağrııldığı yere dönülür.

4. **Continue**

Bir sonraki **BreakPoint** satırına gidilir. Eğer başka bir **BreakPoint** konmamışsa, uygulama normal çalışmasına devam eder.

Command Panelini Kullanmak



Command paneli iki farklı modda kullanılır.

- Immediate

Bu modda, değişken değerleri değiştirebilir, .NET Framework sınıflarındaki metodlar veya kullanıcı tanımlı metodlar çalıştırılabilir. **Immediate** moduna geçmek için **immed** komutu kullanılır.

Immediate modunda bir değişkenin değeri **?** ile öğrenebilir ve yeni değer atanabilir.

```
?sayi
40
sayi = 50
?sayi
50
```

- Command

Bu modda, Visual Studio ortamında tanımlı veya kullanıcı tanımlı makroları, menü öğeleri kullanılabilir. **Command** moduna geçmek için **>cmd** komutu kullanılmalıdır.

```
>cmd
>Debug.StepOver
>Help.About
```

Konu 4: Uygulamanın Derlenmesi

Uygulamanın Derlenmesi

- Derleme Seçeneklerine Bakış

Derleme Seçeneklerine Bakış

Derleme Seçeneklerine Bakış

- Build Solution
- Rebuild Solution
- Build “Proje İsmi”
- Rebuild “Proje İsmi”

C# .NET ile geliştirilen uygulamalar çalıştırılmadan önce derleme işleminden geçer. Derleme işlemi ile kodun C# söz dizimine uygun yazılmışlığı kontrol edilir ve kod çalıştırılmak üzere makine diline çevrilir.

Uygulamaları derlenmesi **Build** menüsünden yapılır.

- Build Solution

Solution içindeki, son derleme işleminden sonra değişen projelerin derlenmesini sağlar.

- Rebuild Solution

Solution içindeki tüm projelerin tekrar derlenmesini sağlar.

- Build “Proje İsmi”

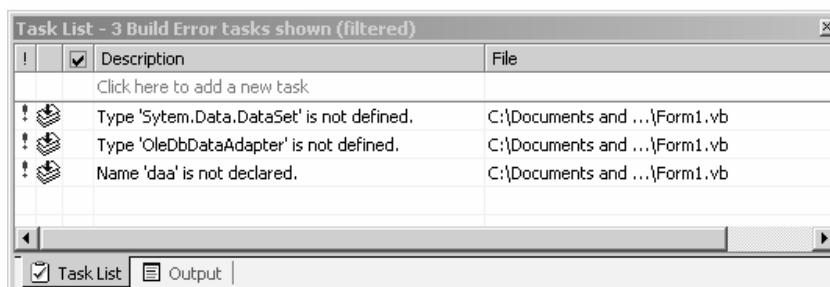
Belirtilen projenin, son derleme işleminden sonra değişen bileşenlerinin derlenmesini sağlar.

- Rebuild “Proje İsmi”

Belirtilen projenin tüm bileşenlerinin tekrar derlenmesini sağlar.

Uygulama derlendikten sonra bulunan hatalar **Task List** panelinde görüntülenir. **Task List** panelinde görüntülenen hatalara çift tıklanarak, hatanın yapıldığı satır ulaşılır.

```
Dim ds As New System.Data.DataSet  
Dim da As New OleDbDataAdapter  
  
daa.Fill(ds)
```

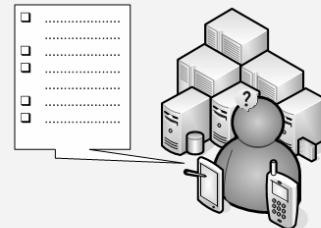


NOT: C# .NET ile uygulama geliştirirken yapılan söz dizimi hataları hemen **Task List** paneeline yansır. Buna **Background Compiling** denir.

Modül Özeti

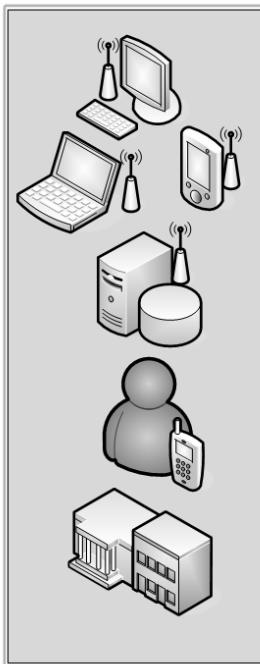
Modül Özeti

- ◆ Assembly nedir?
- ◆ İsim Alanı nedir?
- ◆ Object Browser niçin kullanılır?
- ◆ Server Explorer ne işe yarar?
- ◆ Çalışma zamanı hatalarını yakalamak için neler yapılır?
- ◆ Proje nasıl derlenir?



1. Assembly nedir?
2. İsim Alanı nedir?
3. Object Browser niçin kullanılır?
4. Server Explorer ne işe yarar?
5. Çalışma zamanı hatalarını yakalamak için neler yapılır?
6. Proje nasıl derlenir?

LAB1 : Geliştirme Ortamını Tanımk



Uygulamalar

Geliştirme Ortamını Tanımk

- ◆ Windows Uygulaması Oluşturmak
- ◆ Object Browser Kullanmak
- ◆ Debug Aracını Kullanmak

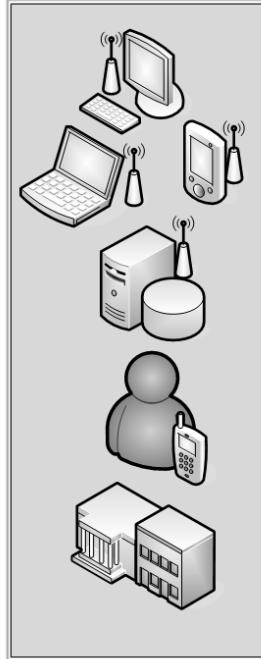
Uygulama1

Windows Uygulaması Oluşturmak

Bu uygulamada “Windows Application” kullanarak Çağrı Merkezi (Call Center) isminde bir uygulaması oluşturacağız.

Çağrı Merkezi uygulasını oluşturmak

1. Visual Studio .Net'i kullanarak Çağrı Merkezi isminde uygulama oluşturmak.
 - **File** menüsü altından **New** alt menüsünü işaret edin ve **Project** komutunu tıklayın.
 - **New Project** iletisi kutusundan “Windows Application” şablonunu seçin.
 - **Name** metin kutusuna “CagriMerkezi” yazın.
 - **Location** metin kutusuna “C:\Proje” yazın ve **OK** butonunu tıklayın
2. Uygulamanın **Assembly Name** “Cagri” adı ile değiştirmek.
 - Proje isminin üzerinde farenin sağ butonunu tıklayın.
 - Açılan menüden **Properties** komutunu verin.
 - Açılan pencere üzerinden **Common Properties** (Genel Özellikler) klasörünü seçin
 - **Common Properties** klasörü altındaki **General** sekmesi içerisinde **Assembly Name** metin kutusuna “Cagri” yazın ve **OK** butonunu tıklayın.



Uygulamalar

Geliştirme Ortamını Tanımak

- ◆ Windows Uygulaması Oluşturmak
- ◆ Object Browser Kullanmak
- ◆ Debug Aracını Kullanmak

Uygulama 2

Object Browser Kullanmak

Bu uygulamada Object Browser’ı kullanarak **System.Data** kütüphanesini inceleyeceğiz.

“**System.Data**” kütüphanesini açmak

1. **View** menüsü içerisinde Object Browser alt menüsünü seçin.
2. **Object** paneli içerisindeki **System.Data** kütüphanesini genişletin.
3. **System.Data** Kütüphanesi içerisindeki **System.Data.OleDb** isim alanını genişletin.
4. **System.Data.OleDb** isim alanını içerisindeki **OleDbConnection**, **OleDbCommand**, **OleDbDataReader**, **OleDbDataAdapter** sınıflarını inceleyin.

Uygulama 3

Debug Aracını Kullanmak

Bu uygulamada çağrı merkezi veritabanına bağlantı açılan kodlar **Debug** kullanarak incelenir.

Kodların yazılması

1. Form1 nesnesinin kod sayfasına geçin ve **OleDbConnection** oluşturan bir fonksiyon yazın.

```
private System.Data.OleDb.OleDbConnection ConnectionOlustur
(string connectionString)
{
    return new
System.Data.OleDb.OleDbConnection(connectionString);
}
```

2. Formun üzerine çift tıklayarak **Load** olayına gelin. **Load** olayında, veri tabanına bağlantı açan kodları yazın. Bu veri tabanı "C:\Proje\CagriMerkezi" klasörü altında bulunacaktır.

```
System.Data.OleDb.OleDbConnection con = new
System.Data.OleDb.OleDbConnection();

// Connection oluşturan fonksiyon çağrıılır

con =
ConnectionOlustur(@"data
source=C:\Proje\CagriMerkezi\CagriMerkezi.mdb;Provider=Micro
soft.Jet.OleDB.4.0");

con.Open();

// Veritabanı işlemleri ilerleyen modüllerde
// anlatılacaktır. Bu kısmı boş bırakın.

con.Close();
```

Hata ayıklama

1. Formun **Load** metoduna bir **BreakPoint** yerleştirin.
2. Projeyi **F5** ile çalıştırın. Başlangıç formu yükleniği zaman **Load** olayı çalışacağı için, çalışma belirtilen noktada durur.
3. **Debug** menüsünden **Step Into** komutunu seçerek ya da **F11** tuşuna basarak kod içersinde ilerleyin. **ConnectionOlustur** isimli metodun içine girildiği görülür.
4. **ConnectionOlustur** metodundan çıkışından sonra, **Locals** panelini açın ve **con** isimli değişkeni inceleyin.
5. **Command** panelini açın ve **immed** komutunu yazarak, **Immediate** moduna geçin.
6. **Command** paneline **con.State** yazarak **Connection** nesnesinin **State** özelliğini öğrenin.
7. **Debug** menüsünden **Continue** komutunu vererek ya da **F5** tuşuna basarak çalışmanın ilerlemesini sağlayın.
8. Formu kapatarak uygulamayı sonlandırın.
9. **con.Open()** kodunun bulunduğu satırda **BreakPoint** koyun ve sağ tıklayarak **BreakPoint Properties** komutunu verin.
10. **Condition** düğmesine tikayın ve metin kutusuna **con.State = 1** yazın. **con** nesnesinin **State** özelliği 1 (bağlantı açık) olduğu zaman çalışmanın durması ayarlanır.

11. **OK** tuşuna basın ve projeyi çalıştırın. Çalışmanın belirtilen noktada durmadığı gözükür. Bunun nedeni, **BreakPoint** içinde verilen şartın sağlanmamasıdır.
12. Formu kapatarak uygulamayı sonlandırın ve Visual Studio ortamından çıkışın.

Modul 1:

Veri Merkezli Uygulamalar ve ADO.NET' e Giriş



Modül 2: Veri Merkezli Uygulamalar ve ADO.NET'e Giriş

Veri Merkezli Uygulamalar ve ADO.NET' e Giriş

- ◆ Veri Merkezli Uygulamalar
- ◆ ADO.NET'e GİRİŞ

Bu modülde verilerin hangi ortamlarda depolandığını öğreneceksiniz. Ayrıca depolanan veriye erişmek için kullanılan yöntemleri öğrenecek ve ADO.NET teknolojisi hakkında bilgi sahibi olacaksınız.

Bu modülün sonunda:

- Veri depolama yöntemlerini öğrenecek,
- Bağlantılı ve Bağlantısız veri ortamlarını öğrenecek,
- Veri erişim yöntemlerini öğrenecek,
- ADO.NET nesne modelini öğrenecek,
- ADO.NET nesne modelinde veri sağlayıcılarını seçebileceksiniz.

Konu 1: Veri Merkezli Uygulamalar

Veri Merkezli Uygulamalar

- ◆ Veri Depolama
- ◆ Bağlantılı (Connected) Veri Ortamları
- ◆ Bağlantısız (Disconnected) Veri Ortamları
- ◆ Veri Erişim Yöntemleri

Veri Depolama

Veri Depolama

Veriye erişmek için çeşitli veri depolama yöntemleri geliştirilmiştir.

- Yapısal Olmayan Yöntem
- Yapısal Yöntem
- Hiyerarşik Yöntem
- İlişkisel Veritabanı Yöntemi
- Nesne Yönetimli Veritabanı Yöntemi

Günümüzde verileri saklamak için çeşitli teknikler kullanılır. Örneğin bir emlakçı emlak alım, satım bilgilerini dosya kâğıtları üzerinde depolayabilir. Bu yöntem veri arama ve listeleme işlemlerinin karmaşık hale gelmesine ve arama

süresinin uzamasına sebep olur. Hatta daha büyük organizasyonlarda işlemlerin yavaşlamasına ve durmasına sebep olabilir.

Artan ihtiyaçlar doğrultusunda veri depolamak ve depolanan veriye erişmek için çeşitli veri depolama yöntemleri geliştirilmiştir. Bu yöntemler:

- **Yapısal Olmayan:** Bu yöntem ile depolanan veriler için belirli bir sınıflandırma ve sıralama yoktur. Veriler düz bir şekilde kaydedilir. Örneğin basit not dosyaları.
- **Yapısal:** Bu yöntem ile depolanan veriler çeşitli grplara ayrılarak saklanır fakat bu gruplar arasında bir alt-üst ayrimı yapılmaz. Örneğin virgülle ayrılmış dosyalar (csv), Excel belgeleri.
- **Hiyerarşik:** Hiyerarşik depolama yöntemini ağaç yapısına benzetur. Bu yöntemde veriler çeşitli kategorilere bölünerek depolanır. Her bir kategorinin içerisinde alt kategorilerde olabilir. Örneğin XML (eXtensible Markup Language) dosyaları.
- **İlişkisel Veritabanı:** İlişkisel veritabanlarında veriler tablolar üzerinde depolanır. Tablo içerisindeki her bir satır kaydı, her bir sütun ise veriyi ifade eder. Örneğin SQL Server, Oracle, Access.
- **Nesne Yönelimli Veritabanı:** En gelişmiş veri depolama yöntemidir. Bu yöntemde veriler; ihtiyaca göre gruplandırılarak, nesneler içerisinde saklanır. Örneğin Versant, AOL

ADO.NET bu depolama tekniklerinin tümünü destekler.

Bağlantılı (Connected) Veri Ortamları

Bağlantılı (Connected) Veri Ortamları

Veri işlemleri gerçekleştiği sürece bağlantı açıktır.

▪ Avantajları

- En güvenli veri ortamı
- Erişimler eş zamanlı

▪ Dezavantajları

- Sabit bir ağ bağlantısı gerektirir.
- Ağ trafiğinin yoğunluğunu arttırmır.

Bağlantılı veri ortamları, uygulamaların veri kaynağına sürekli bağlı kaldığı ortamlardır. Bu ortamlarda veri alma ve değiştirme işlemleri uygulama ile veri kaynağı arasında bağlantı kurulduktan sonra gerçekleştirilir. Bağlantılı veri ortamlarında, veri işlemleri gerçekleştiği sürece bağlantı açık kalır.

İlk bilgisayar üretiminden bugüne en çok tercih edilen yöntem bağlantılı veri ortamları olmuştur. Bağlantılı ortamlar veriye erişmek için birçok avantaj sağlar.

Avantajları:

- En güvenli veri ortamıdır.
- Veri kaynağına yapılan eş zamanlı erişimlerde, veri kaynağının kontrolünü kolaylaştırır.

Dezavantajları:

- Uygulama ile veri kaynağı arasında gerçekleşen bağlantıyı koruyabilmek için sabit bir ağ bağlantısının olması gereklidir.
- Uygulama ile veri kaynağı arasındaki bağlantı ağ üzerinden gerçekleştiği için, ağ trafiğinin yoğunluğunu artırır.

Örneğin araba üreten bir fabrikada yapılan üretim bilgilerinin diğer birimlere ulaştırılması ve bu kayıtların depolanması için eşzamanlı bir bağlantı kurulması gereklidir. Ya da bir emlak firmasında emlakçının, mülk ve menkul bilgilerini güncel tutabilmesi için sabit bir bağlantı kurması gereklidir.

Bağlantısız (Disconnected) Veri Ortamları

Bağlantısız (Disconnected) Veri Ortamları

Bağlantı, veri alış verisi yapılmırken açılır, işlem bittikten sonra kapatılır .

▪ Avantajları

- Taşınabilir aygıtlarla girilen veriler, istenilen zamanda veri ortamlarına aktarılabilir.
- Uygulama performansını arttırr.

▪ Dezavantajları

- Verinin güncellliği sağlanmalıdır.
- Veri çakışmaları önlenmelidir.

Bağlantısız veri ortamı, uygulamanın veri kaynağına sürekli bağlı kalmadığı veri ortamıdır. Uygulama ile veri kaynağı arasında bağlantı, veri alış verisi yapılmırken açılır ve işlem bittikten sonra kapatılır. Bu veri ortamları çevrimdışı çalışmak için kullanılır.

Teknolojinin ilerlemesi ve veri depolayan araçların taşınabilirliğinin sağlanması ile tüm dünyada çevrimdışı ortamlara duyulan ihtiyaç artmıştır. Laptop, Notebook ve Pocket PC gibi araçların yaygınlaşması ile günümüzde uygulamanın veri kaynağına bağlı olmadığı durumlarda bile veri girişi yapılabilir.

Uygulamada sadece çevrimiçi veya çevrimdışı ortamlardan birini seçmek yeterli olmayabilir. Gelişmiş uygulamalarda her iki ortamın avantajlarını birleştiren bir çözüm tercih edilebilir.

Avantajları:

- Laptop, Notebook ve Pocket PC gibi araçlarla girilen veriler, istenilen zamanda veri ortamlarına aktarılabilir.
- Çevrimdışı ortamlar sayesinde, verilerin depolandığı uygulama üzerindeki yük hafifletilir. Bu durum performans artışını sağlar.

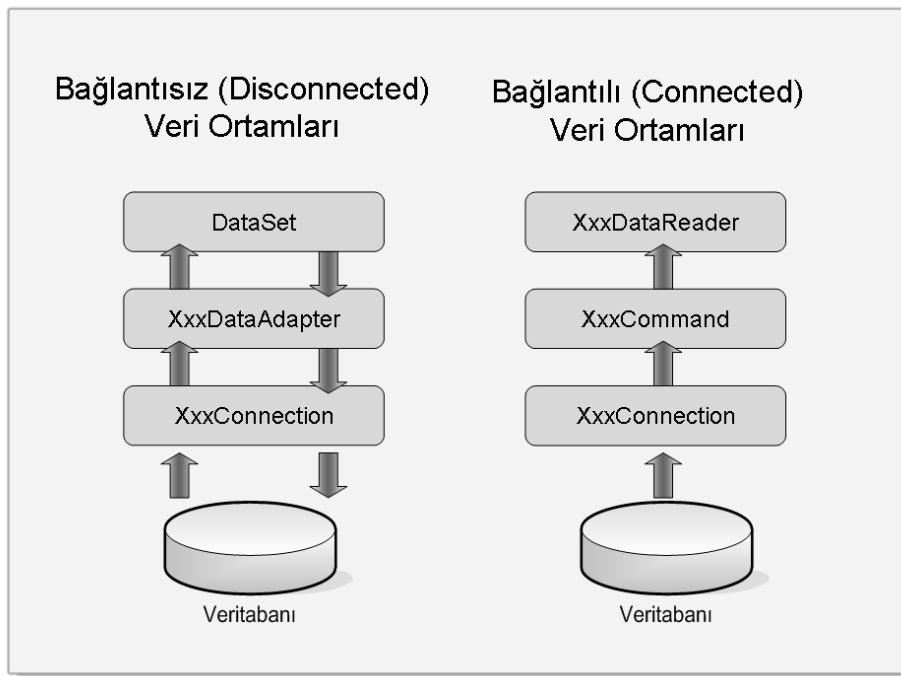
Dezavantajları:

- Bağlantısız veri ortamlarında, verilerin güncel kalmasına dikkat edilmelidir. Bu ortamlarda veri güncelleme işlemleri farklı zamanlarda

gerçekleştirilebilir. Veri üzerinde yapılan bu değişimlerin, diğer kullanıcılarla gösterilebilmesi için çeşitli çözümler geliştirilmelidir.

- Bağlantısız veri ortamları içerisinde farklı kullanıcılar eşzamanlı güncelleme işlemleri gerçekleştirebilir. Bu durumda oluşacak veri çakışmalarının engellenmesi gereklidir.

Örneğin bir toptancı firmasında, firma çalışanları farklı konumdaki bayilerinin tüm siparişlerini bir el bilgisayarına kaydedebilir. Bu veriler el bilgisayarında geçici bir süre için depolanır. Bu süre çalışanların sahada kaldığı süredir. Süre sonunda veriler sunucu bilgisayara aktarılır.



Veri Erişim Yöntemleri

Veri Erişim Yöntemleri

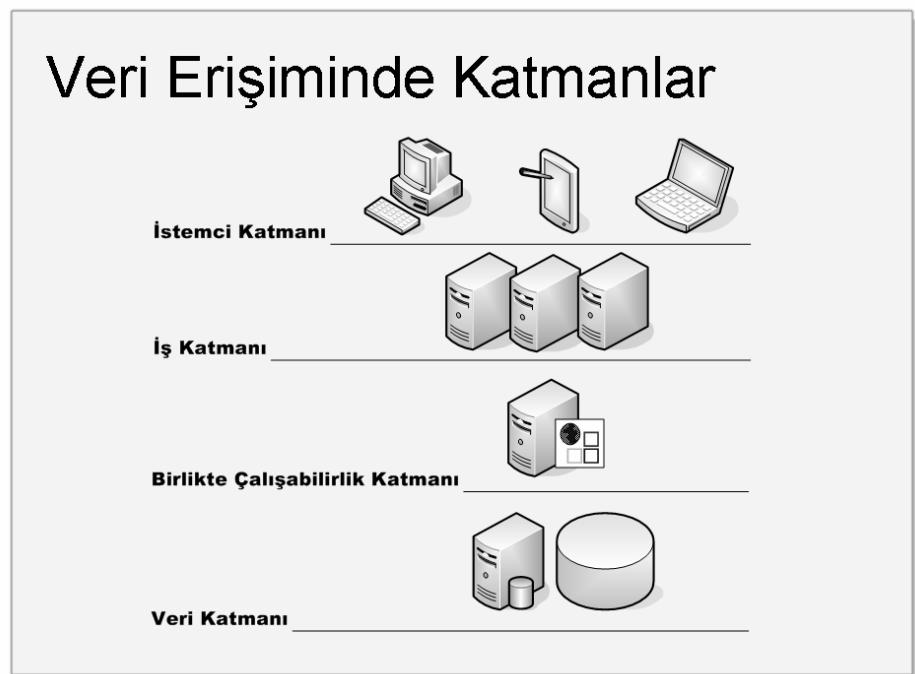
Bir veri erişim modelinde ki mantıksal her birime **Katman (Tier)** denir.

- İstemci Katmanı (Client tier)
- İş Katmanı (Business tier)
- Veri Katmanı (Data tier)
- Birlikte çalışabilirlik Katmanı (Interoperability tier)

İlk bilgisayardan bugüne veriye erişmek için pek çok yöntem geliştirilmiştir. Bu yöntemlerin bazlarında amaç yerleşim, bazlarında ise paylaşım olmuştur. Amacın veriyi saklamak olduğu durumlarda paylaşım konusunda çözüm aranmış, amacın veriyi birçok kullanıcı arasında paylaşmak olduğu durumda ise ana verinin nerede saklanacağı konusunda çözüm yolları aranmıştır.

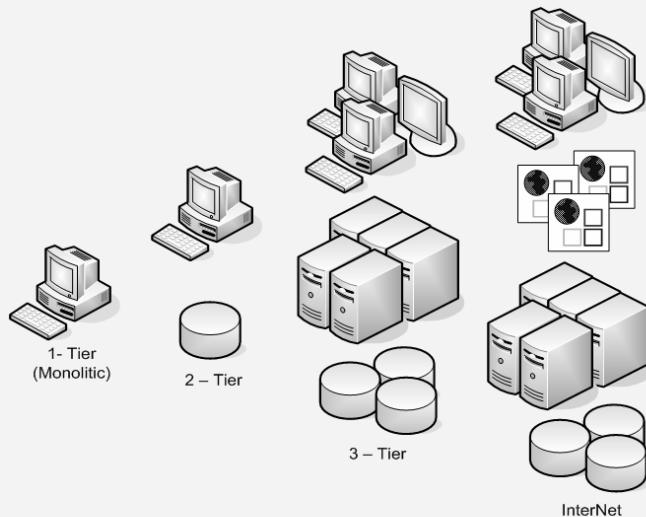
Kullanıcı sayısının ve verinin boyutunun artmasıyla, veri erişimi için bilinen modeller de oldukça gelişmiştir. Birebir veri paylaşımı yerine, internet üzerinden çoklu kullanıcı destekine açık veri erişim modelleri geliştirilmiştir. Günümüzde gelinen son nokta ise, her an her yerden veriye kolayca erişmemizi sağlayan XML Web Servis modelidir.

Veri merkezli uygulamalar geliştirmek için veri erişim modelleri kullanılır. Bir veri erişim modelindeki mantıksal her birime **katman (tier)** denir. Veri merkezli bir uygulamada katman sayısı makine sayısına bağlı değildir. Katman sayısını veri erişim modelindeki düzeyler belirler.



- **İstemci Katmanı (Client tier)**: Sunum ya da kullanıcı servis katmanı olarak da bilinir. Bu katman kullanıcı ara yüzünü içermektedir.
- **İş Katmanı (Business tier)**: Bu katman, uygulamanın veri kaynağı ile etkileşen bölümündür.
- **Veri Katmanı (Data tier)**: Veriyi içeren katmandır.
- **Birlikte çalışabilirlik Katmanı (Interoperability tier)**: Platform ve dilden bağımsız, her tür veriye etkileşim sağlayan katmandır. Bu katmana herhangi bir işletim sistemi üzerinde bulundurulabilen XML Web Servislerini örnek verebiliriz.

Veri Erişiminde Katmanlar



Uygulamalar, katmanlara bölünerek ölçeklenebilirliği artırılır.

NOT: Katman sayısı arttıkça, veri erişim modelinin ölçeklendirilebilirliği ve karmaşıklığı da artar.

Konu 2: ADO.NET'e GİRİŞ

ADO.NET'e GİRİŞ

- ♦ ADO.NET Nedir?
- ♦ ADO.NET Nesne Modeli
- ♦ ADO.NET Veri Sağlayıcıları

ADO.NET Nedir?

ADO.NET Nedir?

Veri kaynaklarına hızlı ve güvenli erişim için Microsoft tarafından geliştirilen nesne modelidir.

- ADO.NET' in Üstünlükleri
 - Verinin yerel bir kopyasını belleğe aktarmak için XML formatının kullanır.
 - N-tier uygulamaları kullanarak uygulamaların farklı katmanlara dağıtımlı.
 - Web uygulamaları için sistem kaynakları korunur.

ADO (Activex Data Objects), farklı veri kaynaklarına hızlı ve güvenli erişim için Microsoft tarafından geliştirilen nesne modelidir. ADO.NET ise ADO teknolojisinin en yeni versiyonudur. ADO ile aynı programlama modelini

kullanmamakla birlikte, ADO modelinden gelen pek çok çözüm yolunu da beraberinde getirir.

Uygulama gelişim ihtiyacı arttıkça, yeni uygulamalarda Web uygulama modeline olan bağlılık gittikçe azalmaktadır. Şimdilerde ise ağ bağlantıları üzerinden veriyi rahatça aktarabilmek için XML kullanımına olan yönelik artmaktadır. İşte ADO.NET, XML ve ADO.NET'in .NET Framework içinde en uygun şekilde programlama ortamı oluşturmamızı sağlar.

ADO.NET modelinin diğer veri erişim modellerine göre üstünlüklerini şöyle sıralayabiliriz:

- ADO.NET, veritabanından çekilen verilerin kopyasını XML formatını kullanarak belleğe aktarır.
- Uygulamanın kullanıcı sayısı arttıkça kaynak kullanımı da artmaktadır. N-Katmanlı (N-tier) uygulama yapısı kullanılarak, uygulamaların katmanlar üzerinden dağıtilması sağlanır. Böylece uygulamaların ölçeklenirliği artar.
- ADO.NET ile bağlantısız veri ortamları için uygulama geliştirilebilir.
- ADO.NET gelişmiş XML desteği verir.

ADO.NET Nesne Modeli:

ADO.NET Nesne Modeli

- **ADO.NET Nesne Modeli**
 - **DataSet Sınıfları**
 - Verinin, çevrimdışı ortamda tutularak, kolayca yönetilmesini sağlar.
 - **.NET Veri Sağlayıcı Sınıfları**
 - Hangi veri kaynağı kullanılacaksa, ona uygun veri sağlayıcı sınıfı kullanılır.
 - **SQL Server .NET Veri Sağlayıcısı**
 - **OLE DB .NET Veri Sağlayıcısı**
 - **Diğer .NET Veri Sağlayıcıları**

ADO.NET nesne modeli iki ana bölümden oluşmaktadır.

- **DataSet** Sınıfları
- **.NET Veri Sağlayıcı Sınıfları**

DataSet sınıfları, çevrimdışı ortamlar için veri depolama ve yönetme işlemlerini sağlar. **DataSet** sınıfları veri kaynağından bağımsız her tür uygulama ve veritabanı için kullanılabilir. Özellikle İlişkisel Veritabanı, XML ve XML Web Servisleri üzerinden veri çekmek için kullanılır.

.NET veri sağlayıcı sınıfları, farklı türdeki veritabanlarına bağlanmak için kullanılır. Bu sınıflar sayesinde istenilen türdeki veri kaynağına kolayca bağlantı kurulabilir, veri çekilebilir ve gerekli güncelleme işlemleri yapılabilir. ADO.NET nesne modeli, aşağıdaki veri sağlayıcı sınıflarını içerir:

- SQL Server .NET Veri Sağlayıcısı
- OLE DB .NET Veri Sağlayıcısı
- Diğer .NET Veri Sağlayıcıları

DİKKAT: Hangi veri kaynağı kullanılacaksa, sadece ona uygun veri sağlayıcı sınıfı kullanılmalıdır.

ADO.NET Veri Sağlayıcıları:

ADO.NET Veri Sağlayıcıları

Tüm veri sağlayıcıları, **System.Data** isim alanı içinde tanımlanmışlardır.

- SQL Server .NET
 - SQL Server 7.0 ve SQL Server 2000
- OLE DB .NET
 - SQL Server 6.5, Oracle, Sybase ve Access
- ORACLE .NET
 - ORACLE
- ODBC .NET
 - Diğer tüm veri tabanlarını

NET veri sağlayıcıları, ADO.NET mimarisinin veritabanı ile uygulama (Windows, Web) veya XML Web Servis arasında bağlantı kurmak için her tür alt yapıyı barındıran çekirdek bileşendir. Tüm veri sağlayıcıları, **System.Data** isim alanı içinde tanımlanmıştır.

NET Framework 1.0 sürümü ile birlikte SQL Server .NET ve OLE DB .NET veri sağlayıcı sınıfları gelmiştir.

SQL Server .NET: SQL Server 7.0 ve SQL Server 2000 veritabanlarına hızlı bağlantı sağlar. SQL Server bağlantı nesneleri **System.Data.SqlClient** isim alanında bulunur.

OLE DB .NET: SQL Server 6.5 ve daha öncesi sürümlerine, Oracle, Sybase, DB2/400 ve Microsoft Access veri tabanlarına bağlantı kurmayı sağlar. OLE DB bağlantı nesneleri **System.Data.OleDb** isim alanında bulunur.

NET Framework 1.1 sürümü ile birlikte SQL Server .NET ve OLE DB .NET veri sağlayıcılarına Oracle .NET ve ODBC .NET veri sağlayıcıları da eklenmiştir.

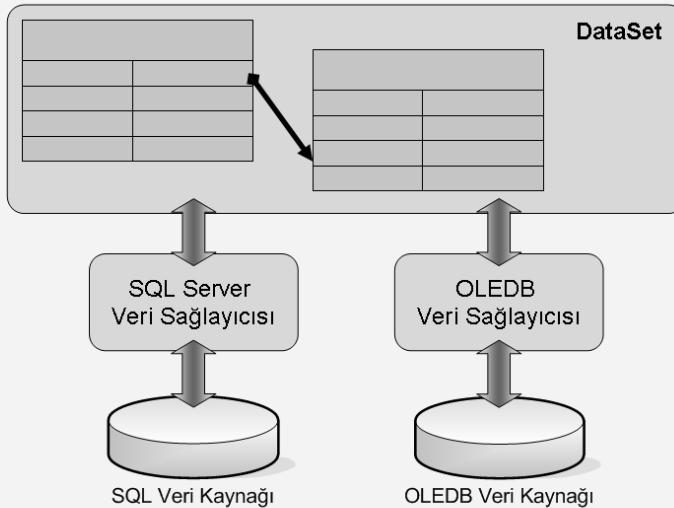
ORACLE .NET: Oracle veritabanlarına bağlantı için tasarlanmış veri sağlayıcısıdır. Oracle bağlantı nesneleri **System.Data.OracleClient** isim alanında bulunur.

NOT: `System.Data.OracleClient` isim alanını kullanmak için, projeye “`System.Data.OracleClient.dll`” referansı eklenmelidir.

ODBC .NET: Diğer veritabanlarını destekleyen genel bir veri sağlayıcıdır. ODBC bağlantı nesneleri **System.Data.ODBC** isim alanında bulunur.

Öğrenim ve kullanım kolaylığı olması amacıyla ADO.NET veri sağlayıcılarının isimlendirilmesinde genelleştirmeye gidilmiştir. SQL Server .NET veri sağlayıcılarının sınıf isimleri “Sql” ön eki ile, OLE DB .NET veri sağlayıcılarının sınıf isimleri ise “OleDb” ön eki ile başlar. Bu genellemeye “SqlConnection” ve “OleDbConnection” örnekleri verilebilir.

ADO.NET Veri Sağlayıcıları



Her bir veri sağlayıcısı içerisinde, birçok bağlantı nesnesi bulunur.

- **Connection**
- **Command**
- **DataReader**
- **DataAdapter**

ADO.NET Veri Sağlayıcıları

Her bir veri sağlayıcısı aşağıdaki nesneleri içerir.

- **XxxConnection**
 - Bağlantı kurmak için kullanılır.
- **XxxCommand**
 - Veritabanına sorğu yollamak için kullanılır.
- **XxxDataReader**
 - Çevrimiçi bağlantı ile sadece veri okuma.
- **XxxDataAdapter**
 - Çevrimdışı bağlantıarda veri işleme nesnesi.

XxxConnection: : Veri kaynağuna bağlantı için kullanılan sınıfıdır.

XxxCommand: Veri kaynağı üzerinde sorgu çalıştırmak için kullanılır. Veri kaynağından dönen kayıtlar XxxDataReader veya DataSet kullanılarak veri bağlantılı kontrollere aktarılır.

XxxDataReader: Çevrimiçi bağlantıarda sadece veri okumak için kullanılan sınıftır.

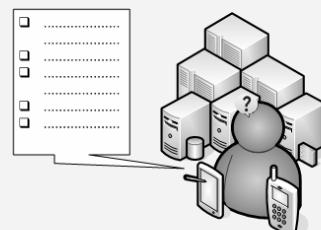
XxxDataAdapter: Çevrimdışı bağlantıarda kullanılan veri işleme nesnesidir.

NOT: Xxx yerine seçilen veri sağlayıcısına göre SQL, OLEDB, Oracle ve ODBC eklerinden biri kullanılır.

Modül Özeti

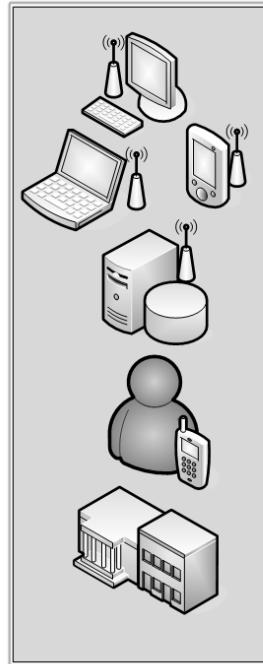
Modül Özeti

- ◆ Veri depolama yöntemleri nelerdir?
- ◆ Bağlantılı ve Bağlantısız veri ortamları nelerdir ?
- ◆ Veri erişim yöntemleri nelerdir?
- ◆ ADO.NET veri sağlayıcıları nelerdir?



1. Veri depolama yöntemleri nelerdir?
2. Bağlantılı ve Bağlantısız Veri ortamları nelerdir?
3. Veri erişim yöntemleri nelerdir?
4. ADO.NET veri sağlayıcıları nelerdir?

Lab 2: Veri Merkezli Uygulamalar ve ADO.NET'e Giriş



Uygulamalar

Veri Merkezli Uygulamalar ve ADO.NET'e Giriş

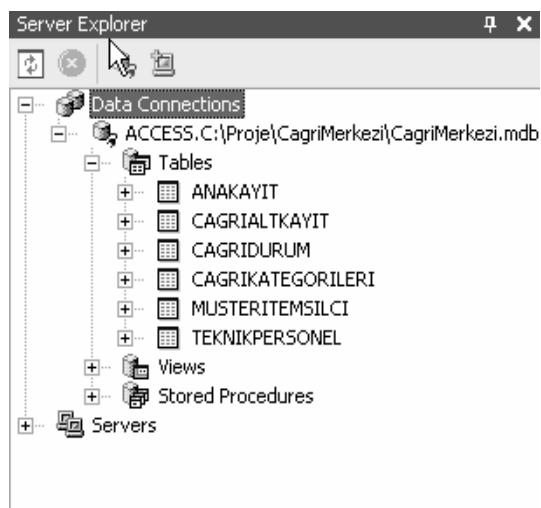
- ◆ Çağrı Merkezi Uygulaması için yeni bağlantı oluşturmak.

Uygulama 1

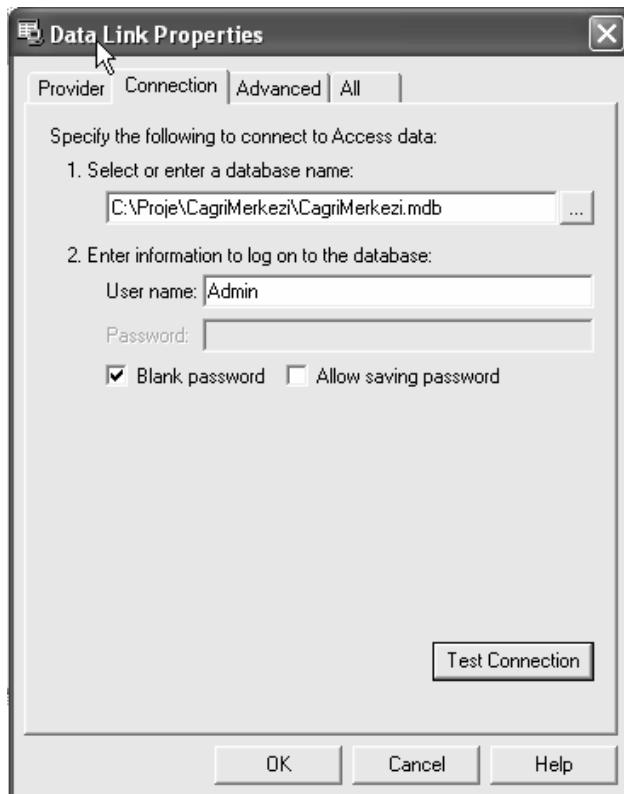
Yeni bağlantı oluşturmak.

Bu uygulamada Server Explorer'ı kullanarak Çağrı Merkezi uygulaması için yeni bir bağlantı oluşturulur.

Çağrı Merkezi Uygulaması için yeni bağlantı oluşturmak.



3. Visual Studio .Net'i kullanarak Çağrı Merkezi uygulamasını açmak.
 - File menüsü altından **Open** alt menüsü içerisinde **Project** komutunu tıklayın.
 - **Look in** açılan kutusundan “C:\Proje\ CagriMerkezi” klasörünü seçin.
 - Açılan **Open Project** penceresinden “CagriMerkezi.sln” dosyasını seçerek **Open** butonunu tıklayın.
4. Uygulamaya CagriMerkezi veritabanını eklemek.
 - Proje isminin üzerinde farenin sağ butonunu tıklayın.
 - Açılan menüden **Add** menüsünden **Add Existing Item** komutunu verin.
 - Açılan pencere üzerindeki **Look in** açılan kutusundan “CD Sürücüsü\Veritabani” klasörünü seçin.
 - Açılan **Add Existing Item** penceresinden “CagriMerkezi.mdb” veritabanını seçerek **Open** butonunu tıklayın.
5. “CagriMerkezi” uygulaması için yeni bağlantı oluşturmak.
 - **Server Explorer** penceresi üzerinde farenin sağ butonunu tıklayın. Açılan menüden **Add Connection** komutunu tıklayın.
 - Açılan **Data Link Properties** penceresinin **Provider** sekmesini tıklayın.
 - Provider sekmesinden **Microsoft.Jet.OLEDB.4.0 Provider**’i seçerek **Next** butonunu tıklayın.



- Açılan **Connection** sekmesinin görüntüsünü resimdeki gibi düzenleyerek **OK** butonunu tıklayın.

Veri Kaynaklarına Bağlanmak



Modül 3: Veri Kaynaklarına Bağlanmak

Veri Kaynaklarına Bağlanmak

- ◆ Veri Sağlayıcı Seçmek
- ◆ Bağlantı Oluşturmak
- ◆ Bağlantı Yönetimi

Veriyi yöneten uygulamalar, bu verilerin bulunduğu kaynağa bağlanma ihtiyacı duyar. Visual Basic .NET ile veri kaynağının bağlanması için, kaynağın tipine, yapısına göre farklı nesneler ve farklı veri sağlayıcıları kullanılır.

Bu modülün sonunda:

- Farklı veritabanlarına göre veri sağlayıcıları seçebilecek,
- Bağlantı cümlesi oluşturabilecek,
- Farklı veritabanları için **Connection** nesnelerini yönetebileceksiniz.

Konu 1: Veri Sağlayıcı Seçmek

Veri Sağlayıcı Seçmek

- ◆ Veri Sağlayıcı Nedir?
- ◆ Veri Sağlayıcı Sınıfları

Veri Sağlayıcı Nedir?

Veri Sağlayıcı Nedir ?

Uygulama ile veritabanı arasında bağlantı kurmak ve kurulan bağlantı üzerinden kayıtları almak, değiştirmek ve silmek için veri sağlayıcıları kullanılır.

Microsoft .Net Framework ;

- SQL Server .NET
- OLEDB .NET
- ODBC .NET

ADO.NET mimarisi, uygulama ile veritabanı arasında bağlantı kurmak ve kurulan bağlantı üzerinden kayıtları almak, değiştirmek ve silmek için veri sağlayıcılarını kullanır. Farklı veritabanları için farklı veri sağlayıcıları kullanılır.

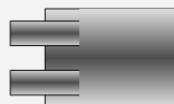
Uygun veri sağlayıcı seçiminde en önemli kriter “Hangi sağlayıcı en iyi performansı verir?” sorusunun cevabıdır. Çünkü Sql Server, Oracle, Access gibi veritabanlarına farklı veri sağlayıcıları ile erişilebilir.

Microsoft .NET Framework, veritabanları ile bağlantı kurmak için farklı veri sağlayıcılarını destekler.

- SQL Server .NET
- OLEDB .NET
- ODBC .NET

Veri Sağlayıcı Sınıfları

Veri Sağlayıcı Sınıfları



▪ System.Data.dll



▪ System.Data

.NET Framework içindeki veri sağlayıcıları, **System.Data.dll** içerisindeki **System.Data** isim alanında yer alır. Tablo 1.1 de hangi sağlayıcı isim alanı ile hangi veritabanına bağlanabileceğinin gösterilmektedir.

Veri Sağlayıcı Sınıfları

Uygulamalarımız için hangi veri sağlayıcısı en iyi performansı verir ?

Veri Tabanı	Veri Sağlayıcısı İsim Alanı
Sql Server 7.0 ve sonrası sürümler	System.Data.SqlClient
Sql Server 6.5 ve öncesi sürümler	System.Data.OleDb
Microsoft Access veri tabanı	System.Data.OleDb
Oracle Server	System.Data.OracleClient
Diğer veri tabanları	System.Data.OleDb

Class	Veri Kaynağı
System.Data.SqlClient.SqlConnection	SQL Server
System.Data.OleDb.OleDbConnection	OLE DB veri sağlayıcısı
System.Data.Odbc.OdbcConnection	ODBC veri sağlayıcısı
System.Data.OracleClient.OracleConnection	Oracle

Veri Tabanı	Veri Sağlayıcısı İsim Alanı
Sql Server 7.0 ve sonraki sürümler	System.Data.SqlClient
Sql Server 6.5 ve önceki sürümler	System.Data.OleDb
Microsoft Access veri tabanı	System.Data.OleDb
Oracle Server	System.Data.OracleClient
Diğer veri tabanları(Oracle, Sybase, DB2/400)	System.Data.OleDb

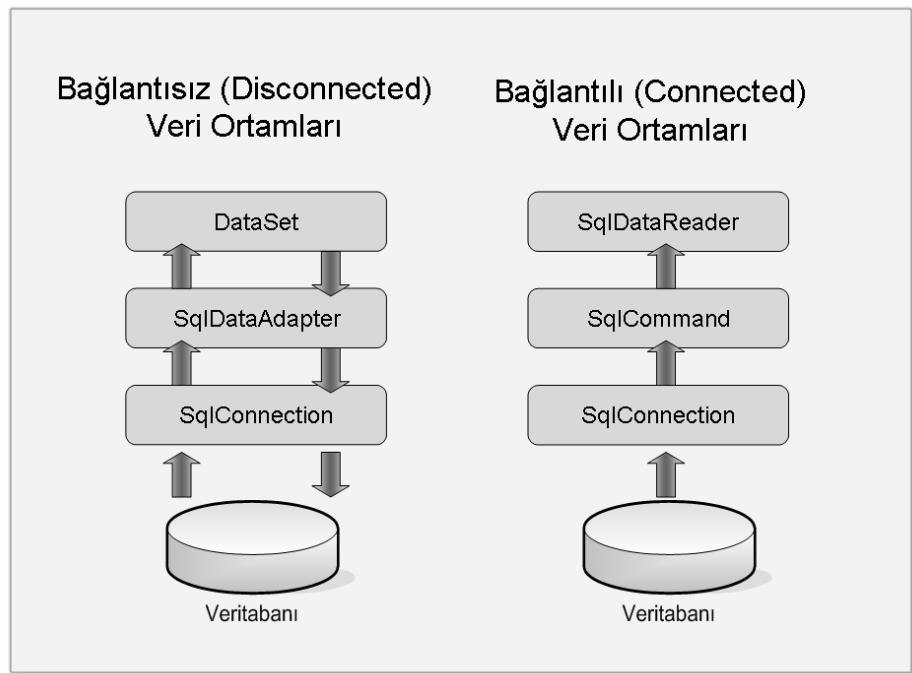
Tablo 1.1: Veri Tabanları ve Veri Sağlayıcı İsim Alanları

ODBC .NET veri sağlayıcıları, diğer veri sağlayıcılarından farklı olarak, veri kaynağına bağlanırken hiçbir ara katman kullanmaz. Bunun yerine, bağlantı için ODBC API'leri kullanır.

.NET Framework veri sağlayıcıları Tablo 1.2 de belirtilen sınıfları kullanmaktadır. Sınıf isimlerinin önündeki XXX ön eki kullanılan veri sağlayıcı ismini simgeler. Eğer veritabanına OLEDB veri sağlayıcısı ile bağlanılırsa OLEDB ön ekini, eğer SQL Server veri sağlayıcısı ile bağlıyorsa SQL ön ekini alır.

Sınıf	Açıklama
XXXConnection	Bağlantı açmak ve kapatmak için kullanılan sınıfır.
XXXCommand	Veritabanı üzerinde Stored Procedure (Saklı Yordamlar) veya SQL Cümleleri çalışırmak için kullanılan sınıfır.
XXXDataReader	Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılan sınıfır.
XXXDataAdapter	Veritabanından çekilen verileri DataSet içerisinde veya DataSet 'e çevrimdışı eklenmiş verileri veritabanına aktarmak için kullanılan sınıfır

Tablo 1.2: Veri Sağlayıcı Sınıf İsimleri



System.Data.SqlClient isim alanı içerisinde çevrimiçi bağlantılar geliştirmek için **SqlConnection**, **SqlCommand**, **SqlDataReader** sınıfları kullanılır.

SqlConnection; MS SQL Server üzerinde bağlantı açmak ve kapatmak için kullanılan sınıfıdır.

SqlCommand; MS SQL Server üzerinde Stored Procedure (Saklı Yordamlar) veya SQL Cümleleri çalışıtmak için kullanılan sınıfır.

SqlDataReader; MS SQL Server üzerinde **SqlCommand** ile çalıştırılan SELECT sorguların sonuçlarını geri döndürmek için kullanılan sınıfır.

System.Data.SqlClient isim alanı içerisinde çevrimdışı bağlantılar geliştirmek için **SqlConnection**, **SqlDataAdapter**, **DataSet** sınıfları kullanılır.

SqlConnection; MS SQL Server üzerinde bağlantı açmak ve kapatmak için kullanılan sınıfır.

SqlDataAdapter; MS SQL Server'dan çekilen verileri DataSet içerisinde ve DataSet'e çevrimdışı eklenmiş verileri MS SQL Server'a aktarmak için kullanılan sınıfır.

DataSet; SQLDataAdapter nesnesinden gelen kayıtları çevrimdışı depolamak ve yönetmek için kullanılan sınıfır. DataSet tüm veri sağlayıcı sınıflar için ortaktır.

NOT: **DataSet**, **System.Data** isim alanı içerisinde yer alır.

System.Data.OleDb isim alanı içerisinde çevrimiçi bağlantılar geliştirmek için **OleDbConnection**, **OleDbCommand**, **OleDbDataReader** sınıfları kullanılır.

OleDbConnection; Access veya diğer veritabanları üzerinde bağlantı açmak ve kapatmak için kullanılan sınıfır.

OleDbCommand; Access veya diğer veritabanları üzerinde Stored Procedure (Saklı Yordamlar) veya SQL Cümleleri çalışıtmak için kullanılan sınıfır.

OleDbDataReader; Access veya diğer veritabanları üzerinde **OleDbCommand** ile çalıştırılan SELECT sorguların sonuçlarını geri döndürmek için kullanılan sınıfır.

System.Data.OleDb isim alanı içerisinde çevrimdışı bağlantılar geliştirmek için **OleDbConnection**, **OleDbDataAdapter** sınıfları kullanılır.

OleDbConnection; Access veya diğer veritabanları üzerinde bağlantı açmak ve kapatmak için kullanılan sınıfır.

OleDbDataAdapter; Access veya diğer veritabanlarından çekilen verileri DataSet içerisinde ve DataSet'e çevrimdışı eklenmiş verileri ilgili veritabanına aktarmak için kullanılan sınıfır.

Konu 2: Bağlantı Oluşturmak

Bağlantı Oluşturmak

- ◆ Bağlantı Cümlesi (Connection String) Oluşturmak
- ◆ Bağlantı Cümlesini (Connection String) Kullanmak
- ◆ Bağlantı Cümlesi(ConnectionString) Örnekleri

Bağlantı Cümlesi (Connection String) Oluşturmak

Bağlantı Cümlesi Oluşturmak

Bağlantı cümlesi, bir veri kaynağına bağlanmak için gerekli olan en temel ögedir.

Parametre	Tanımı
Provider	Bağlantı sağlayıcısının ismini tutar.
ConnectionTimeout Connect Timeout	bağlantı için beklenmesi gereken maksimum saniye sayısıdır.
Initial Catalog	Veri tabanı ismi
Data Source	Veri tabanı için dosya ismi
Password (pwd)	Hesap bağlantı şifresi
User Id (uid)	Hesap kullanıcı ismi
Integrated Security Trusted Connection	Bağlantının güvenli olup olmadığını belirten parametredir.
Persist Security Info	Default durumda güvenlik için hassas bilgileri geri döndürmez.
WorkstationID (wid)	Workstation veya client adını belirtir.
Packet Size	Veri transferinde kullanılan paketlerin boyutunu belirtir.
Mode	Read-only yada Write modunu belirtir.

Bağlantı cümlesi, veri kaynağına bağlanmak için gerekli bilgileri tutar.. Bu cümle, veri kaynağına bağlantı kurmak için gerekli bağlantı parametrelerin birleşiminden oluşur. Bu parametrelerin listesi Tablo 2.1'de gösterilmiştir.

Parametre	Tanımı
Provider	Sadece OLEDBConnection nesnelerinde kullanılır. Bağlantı sağlayıcısının ismini tutar. Sağlayıcı isimleri Tablo 2.2 de belirtilmiştir.
ConnectionTimeout veya Connect Timeout	Veritabanı bağlantı için beklenmesi gereken maksimum saniye sayısıdır. Varsayılan değer 15 saniye dir.
Initial Catalog	Veri tabanı adı
Data Source	SQL Server adı, veya MS Access veri tabanı için dosya adı
Password (pwd)	SQL Server login(giriş) parolası
User Id (uid)	SQL Server login(giriş) adı
Integrated Security veya Trusted Connection	SQL sunucusuna Windows hesabı ile bağlantı yapılacağını belirtir. True , False veya SSPI girilebilir. SSPI , True ile eş anlamlıdır ve bu durumda Windows hesabı kullanılır.
Persist Security Info	Varsayılan değeri False olur. Bu durumda güvenlik için hassas bilgileri geri döndürmez. True olduğunda ise güvenlik risk taşıymaya başlar.
WorkstationID (wid)	Workstation veya client(istemci) adını belirtir.
Packet Size	Client(istemci)-server(sunucu) arası veri transferinde kullanılan paketlerin boyutunu belirtir.
Mode	Veritabanını Read-only(Sadece okunur) ya da Write(Yazılabilir) modunu belirtir. SQL Server bağlantılarında kullanılmaz.

Tablo 2.1: Bağlantı cümlesinin parametreleri

Provider parametresinin Access, SQL Server ve Oracle veri tabanlarına göre alacağı değerler Tablo 2.2 de gösterilmiştir

Tür	Açıklama
SQLOLEDB	SQL Server için Microsoft OLE DB Provider
MSDAORA	ORACLE için Microsoft OLE DB Provider
Microsoft.Jet.OLEDB.4.0	Microsoft Jet için OLE DB Provider

Tablo 2.2: Bağlantı cümlesinin parametreleri

Bağlantı Cümlesini (Connection String) Kullanmak

Bağlantı Cümlesini Kullanmak

```
System.Data.OleDb.OleDbConnection cnNorthwind = new
    System.Data.OleDb.OleDbConnection();
cnNorthwind.ConnectionString="Provider=Microsoft.Jet.OLEDB.4.0;" +
    "Data Source=C:\\Samples\\Northwind.mdb;";
cnNorthwind.Open();
```

Yeni bağlantı oluşturmak ve yönetmek için **OleDbConnection**, **SqlConnection** gibi **XXXConnection** sınıfları kullanılır. Veri kaynağına bağlanmak için oluşturulan Bağlantı Cümlesi, **XXXConnection** sınıfının **ConnectionString** özelliğine atanır.

Örnekte SQL Server veritabanı için bağlantı cümlesi oluşturulmuştur. London isimli sunucuda bulunan Northwind veritabanına, sa kullanıcı ismi ve 2389 parolası ile bağlanılıyor. Eğer veritabanı sunucusundan 60 saniye içinde cevap alınamazsa bağlantı iptal ediliyor.

```
System.Data.SqlClient.SqlConnection cnNorthwind;
cnNorthwind = new System.Data.SqlClient.SqlConnection();

cnNorthwind.ConnectionString ="Data Source=London;Initial
Catalog=Northwind;User ID=sa;Password=2389;Connection
Timeout=60";
```

Örnekte Microsoft Access veritabanı için bağlantı cümlesi oluşturulmuştur. OleDb bağlantısı yapıldığı için **Provider** özelliğinin Microsoft.Jet.OleDB.4.0 olarak belirtilmesi gereklidir. Bağlantının yapılacak olduğu Northwind veritabanının local makinede C:\Samples dizini altında bulunduğu belirtiliyor.

```
System.Data.OleDb.OleDbConnection cnNorthwind;
cnNorthwind = new System.Data.OleDb.OleDbConnection();
```

```
cnNorthwind.ConnectionString=@"Provider=Microsoft.Jet.OLEDB.  
4.0;Data Source=C:\Samples\Northwind.mdb";
```

Örnekte Sql Server 6.5 veri tabanı için bağlantı cümlesi oluşturulmuştur. SQL Server 7.0 sürümünden eski bir veritabanı sunucuna bağlantı yapıldığı için **Provider** özelliği SQLOLEDB olarak belirtiliyor. ProdServ isimli sunucudaki Pubs veritabanına, Windows hesabı (SSPI) ile bağlanılıyor.

```
System.Data.OleDb.OleDbConnection cnNorthwind;  
cnNorthwind = new System.Data.OleDb.OleDbConnection();
```

```
cnNorthwind.ConnectionString = "Provider=SQLOLEDB;Data  
Source=ProdServ;Initial Catalog=Pubs;Integrated  
security=SSPI";
```

DİKKAT: Microsoft Access veri kaynağı, tek veri tabanından oluşur. SQL Server veri kaynağı ise birden fazla veri tabanından oluşur. Bu yüzden SQL Server veritabanı bağlantı cümlesinde Initial Catalog parametresi kullanılır.

Bağlantı Cümlesi(ConnectionString) Örnekleri

Bağlantı Cümlesi Örnekleri

- ◆ Ms Access ile OLEDB Bağlantı Cümleleri
- ◆ SQL Server ile ODBC Bağlantı Cümleleri
- ◆ SQL Server ile OLEDB Bağlantı Cümleleri
- ◆ SQL Server ile Sql Server Bağlantı Cümleleri

Ms Access ile OLEDB Bağlantı Cümleleri

Ms Access ile OLEDB Bağlantı Cümleleri

Access'e Bağlantı	"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=DB_Name.mdb; "
Access'e Çalışma Grubu dosyası üzerinden Bağlantı	"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db_Name.mdb; Jet OLEDB:System Database=Db_Name.mdw"
Access'e Parola Korumalı Bağlantı	"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db_Name.mdb; Jet OLEDB:Database Password=sifreniz"
Network'teki Access'e Bağlantı	"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=\\Server_Name\Share_Name\Share_Path\Db_Name.mdb"
Remote Server(Uzak Server) üzerindeki bir Access'e Bağlantı	"Provider=MS Remote; Remote Server=http://Your-Remote-Server-IP; Remote Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db_Name.mdb"

Tablo 2.3 de OLEDB ile Access'e bağlanmak için gerekli, örnek Bağlantı Cümleleri gösterilmektedir.

Access'e Bağlantı	"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=DB_Name.mdb; "
Access'e Çalışma Grubu dosyası üzerinden Bağlantı	"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db_Name.mdb; Jet OLEDB:System Database=Db_Name.mdw"
Access'e Parola Korumalı Bağlantı	"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db_Name.mdb; Jet OLEDB:Database Password=sifreniz"
Network'teki Access'e Bağlantı	"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=\\Server_Name\Share_Name\Share_Path\Db_Name.mdb"
Remote Server(Uzak Server) üzerindeki bir Access'e Bağlantı	"Provider=MS Remote; Remote Server=http://Your-Remote-Server-IP; Remote Provider=Microsoft.Jet.OLEDB.4.0; Data Source=Db_Name.mdb"

Tablo 2.3: Ms Access ile OLEDB Bağlantı Cümleleri

SQL Server ile ODBC Bağlantı Cümleleri

SQL Server ile ODBC Bağlantı Cümleleri

SQL Server'a SQL Authentication ile bağlanmak	"Driver={SQL Server};Server=Server_Name;Database=Db_Name;Uid=Username;Pwd=sifreniz;"
SQL Server'a Windows Authentication ile bağlanmak	"Driver={SQL Server}; Server= Server_Name; Database=DB_Name; Trusted_Connection=yes;"

Tablo 2.4 de ODBC ile SQL Server'a bağlanmak için gerekli, örnek Bağlantı Cümleleri gösterilmektedir.

SQL Server sunucusuna SQL Authentication ile bağlanmak	"Driver={SQL Server};Server=Server_Name;Database=Db_Name;Uid=Username;Pwd=sifreniz;"
SQL Server sunucusuna Windows Authentication ile bağlanmak	"Driver={SQL Server}; Server= Server_Name; Database=DB_Name; Trusted_Connection=yes;"

Tablo 2.4: SQL Server ile ODBC Bağlantı Cümleleri

SQL Server ile OLEDB Bağlantı Cümleleri

SQL Server ile OLEDB Bağlantı Cümleleri

SQL Server'a SQL Authentication ile bağlanmak	"Provider=SQLOLEDB;Data Source= Server_Name; Initial Catalog=Db_Name;User Id= Username; Password=sifreniz;"
SQL Server'a Windows Authentication ile bağlanmak	"Provider=SQLOLEDB;Data Source= Server_Name; Initial Catalog=DB_Name; Integrated Security=SSPI;"

Tablo 2.5 de OLEDB ile SQL Server'a bağlanmak için gerekli, örnek Bağlantı Cümleleri gösterilmektedir.

SQL Server sunucusuna SQL Authentication ile bağlanmak	"Provider=SQLOLEDB;Data Source= Server_Name;Initial Catalog=Db_Name;User Id= Username;Password=sifreniz;"
SQL Server sunucusuna Windows Authentication ile bağlanmak	"Provider=SQLOLEDB;Data Source= Server_Name;Initial Catalog=DB_Name;Integrated Security=SSPI;"

Tablo 2.5: SQL Server ile OLEDB Bağlantı Cümleleri

SQL Server ile Sql Server Bağlantı Cümleleri

SQL Server ile Sql Server Bağlantı Cümleleri

SQL Server'a SQL Authentication ile bağlanmak	"Data Source= _Server_Name;Initial Catalog=Db _Name; User Id= Username;Password=sifreniz;"
SQL Server'a SQL Authentication ile bağlanmak	"Server= Server_Name;Database=Db_Name; UserID=Username; Password=sifreniz; Trusted_Connection=False"
SQL Server'a Windows Authentication ile bağlanmak	"Data Source= Server_Name;Initial Catalog=Db_Name;Integrated Security=SSPI;"
SQL Server'a SQL Authentication ile bağlanmak	"Server=Server_Name;Database=Db_Name; Trusted_Connection=True;"

Tablo 2.6 de SQLClient ile SQL Server'a bağlanmak için gerekli, örnek Bağlantı Cümleleri gösterilmektedir.

SQL Server sunucusuna SQL Authentication ile bağlanmak	"Data Source= _Server_Name;Initial Catalog=Db _Name;User Id= Username;Password=sifreniz;"
SQL Server sunucusuna SQL Authentication ile bağlanmak	"Server= Server_Name;Database=Db_Name;User ID= Username;Password=sifreniz;Trusted_Connection=False"
SQL Server sunucusuna Windows Authentication ile bağlanmak	"Data Source= Server_Name;Initial Catalog=Db_Name;Integrated Security=SSPI;"
SQL Server sunucusuna SQL Authentication ile bağlanmak	"Server=Server_Name;Database=Db_Name;Trusted_Connection=True;"

Tablo 2.6: SQL Server ile SQL Server Bağlantı Cümleleri

UYARI : Bağlantı cümle paramerelerinin benzer eşdeğerleri vardır. Bu eşdeğerler hem OLEDB hemde SQLClient veri sağlayıcılarda kullanılabilir. Tablo 2.7. de bu eşdeğerler gösterilmektedir.

OLEDDB ve SqlServer Parametreleri	Eşdeğerleri
Data Source	Server
User ID	UID
Password	PWD
Initial Catalog	Database

Tablo 2.7: OleDb ve Sql Server Parametre Eşdeğerleri

Konu 3: Bağlantı Yönetimi

Bağlantı Yönetimi

- ♦ Bağlantıyı Açımak ve Kapatmak
- ♦ Bağlantı Durumlarını Kontrol Etmek

Bağlantıyı Açımak ve Kapatmak

Bağlantıyı Açımak ve Kapatmak

Veri tabanına bağlantı kurmak için, **Connection** nesnesinin **Open** metodu, bağlantıyi kapatmak için ise **Close** metodu kullanılır.

- **Open()**
 - Timeout
 - Varsayılan değer 15 sn
- **Close()**
- **Try, Catch, Finally**

Bağlantı cümlesini oluşturduktan sonra, bağlantıyi açmak ve kapamak için **Connection** sınıfının iki önemli metodu kullanılır.

- Open

- Close

Open metodu, bağlantı cümlesinde belirtilen veri kaynağını açmak için kullanılır.

Close metodu, açılan bağlantıyı kapatmak için kullanılır. **Close** metodu ile kullanılmayan bağlantıları kapatmak, kaynak tüketimini azaltır.

Open metodu; uygulama ile veri kaynağı arasındaki bağlantıyı, bağlantı cümlesinin **Timeout** parametresinde belirtilen süre içerisinde kurmaya çalışır. Eğer belirtilen süre içerisinde bağlantı gerçekleşmeyorsa, uygulama hata üretir. Bu süre için herhangi bir değer belirtilmemişse, varsayılan değer 15 saniyedir.

Daha önceden açılmış bir bağlantı; kapatılmadan tekrar açılmaya çalışılırsa, uygulama yine hata üretecektir. Kapatılan bağlantının yeniden kapatılması hataya yol açmaz.

Open metodu ile açılan bağlantının kapatılmaması durumunda, "Garbage Collector" adı verilen çöp toplayıcı devreye girerek bağlantının kapatılmasını sağlar. Bu durum bağlantı değişkeninin geçici bir süre bellekte yer tutmasına neden olur.

NOT: Bağlantı nesnesinin **Dispose** metodu da bağlantıyı kapatmak için kullanılabilir.

Örnekte Northwind.mdb isimli Access veritabanı üzerinde, **Open** ve **Close** metodlarının kullanımı gösterilmektedir.

```
cnNorthwind.ConnectionString = @"Provider=Microsoft.Jet.  
OLEDB.4.0;Data Source=C:\Samples\Northwind.mdb";  
  
//Bağlantıyı açmak  
cnNorthwind.Open();  
  
//Veritabanı işlemleri bu arada gerçekleştirillir.  
//Bağlantıyı kapatmak  
cnNorthwind.Close();
```

Open metodu ile veri kaynağı açılırken, çeşitli çalışma zamanı hatalarından dolayı bağlantı açılmayabilir ve uygulama hata üretelebilir. Bu çalışma zamanı hataları

- Sunucunun bulunamamasından,
- Veritabanının bulunamamasından,
- Hatalı kullanıcı adı veya parola girilmesinden,
- Donanım veya yazılımdan kaynaklanabilir.

Try, **Catch**, **Finally** deyimlerini kullanarak bir bloğu oluşturabilecek potansiyel hatalardan korunur.

Try bloğu içinde, hata üretebilecek kodlar yazılır. Örneğin tüm veritabanı işlemleri bu blok içersine yazılmalıdır.

Catch blokları, uygulamanın ürettiği hataları, tiplerine göre sıralı bir şekilde işler. Bu blok içersine, yakalanan hataya göre yapılacak işlemler yazılmalıdır. Örneğin bağlantının açılmadığı bir durumda veritabanı işlemleri gerçekleştirilmeye çalışıldığı zaman, kullanıcıya bağlantının açılmasını bildiren mesaj kutusu çıkarılabilir.

Finally blogunda, **Try** ve **Catch** bloklarından herhangi biri işlendiğten sonra çalışır. Bu blokta, hatanın üretildiği veya üretilmediği iki durumda da yapılması gereken işlemler yazılır. Örneğin, bağlantının kapatılması her iki durumda da yapılması gereken bir işlemdir.

```
System.Data.OleDb.OleDbConnection cnNorthwind;
try
{
    cnNorthwind = new System.Data.OleDb.OleDbConnection();
    cnNorthwind.ConnectionString = @"Provider=Microsoft.Jet.
OLEDB.4.0;Data Source=C:\Samples\Northwind.mdb";
    cnNorthwind.Open();
    // Veritabanı işlemleri gerçekleştiriliyor.
}
catch (InvalidOperationException xcpInvOp)
{
    // İlk önce bu tipte hata yakalanır.
    MessageBox.Show("Önce veri tabanı bağlantısını kapatın");
    // Hata Mesajının içeriğini görmek için kullanılır.
    MessageBox.Show(xcpInvOp.ToString());

}
catch (Exception xcp)
{
    // Diğer hatadan farklı bir tipte hata burda yakalanır.
    MessageBox.Show(xcp.ToString());
}
finally
{
    cnNorthwind.Close();
    // ya da
    cnNorthwind.Dispose();
}
```

Bağlantı Durumlarını Kontrol Etmek

Bağlantı Durumlarını Kontrol Etmek

Bağlantı nesnesinin **State** property'si, bir bağlantı nesnesinin durumu hakkında bilgi verir.

İsim	Açıklama	Değeri
Broken	Yalnızca, açık bir bağlantının kopup tekrar bağlanıldığı durum	16
Closed	Bağlantı kapalı	0
Connecting	Veri kaynağına bağlanma aşamasında	2
Executing	Bağlantı üzerinden bir komutu çalıştırılıyor	4
Fetching	Bağlantı üzerinden veri çekiliyor	8
Open	Bağlantı açık	1

Bağlantı sınıfının durumu hakkında bilgi almak için, bağlantı sınıfının **State** özelliği kullanılır.

State özelliğinin alabileceği değerler tablo 3.1.1'de belirtilmiştir.

İsim	Açıklama	Değeri
Broken	Yalnızca, açık bir bağlantının kopup tekrar bağlanıldığı durum	16
Closed	Bağlantı kapalı	0
Connecting	Veri kaynağına bağlanma aşamasında	2
Executing	Bağlantı üzerinden bir komutu çalıştırılıyor	4
Fetching	Bağlantı üzerinden veri çekiliyor	8
Open	Bağlantı açık	1

Tablo 3.1.1: Connection nesnesinin State Property değerleri

```
private void ConnectionAc(OleDb.OleDbConnection con)
{
    //Connection, sadece kapalı ise açılacak
    if (con.State == ConnectionState.Closed)
    {
        con.Open();
    }
}
```

Bağlantı nesnelerinin durumu değiştiği zaman **StateChange** olayı tetiklenir. Bu olay ile bağlantının hangi durumlarda açılıp kapandığı öğrenilebilir.

Bağlantının eski ve yeni durumları **StateChangeEventArgs** parametresi ile öğrenilir. Tablo 3.1.2'de bu parametrenin **CurrentState** ve **OriginalState** özellikleri görülür.

Property	Açıklama
CurrentState	Bağlantının yeni durumu hakkında bilgi verir.
OriginalState	Bağlantının değişmeden önceki durumu hakkında bilgi verir.

Tablo 3.1.2: StateChangeEventArgs parametresinin özellikleri

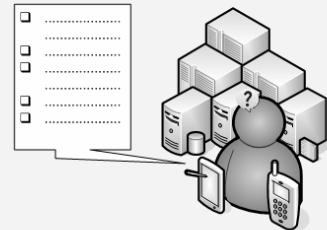
```
public void ConnectionOlustur()
{
    System.Data.OleDb.OleDbConnection conn;
    conn = new System.Data.OleDb.OleDbConnection();
    // Bağlantı ayarları yapılır.
    // ...
    // Bağlantının StateChange olayı gerçekleştiği zaman
    // DurumRapor yordamının çağırılması ayarlanır
    conn.StateChange += new
    StateChangeEventHandler(DurumRapor);
}

public void DurumRapor(Object sender, StateChangeEventArgs e)
{
    MessageBox.Show("Bağlantı " + e.OriginalState.ToString() +
    " durumundan " + e.CurrentState.ToString() + " durumu olarak
    değişti.");
}
```

Modül Özeti:

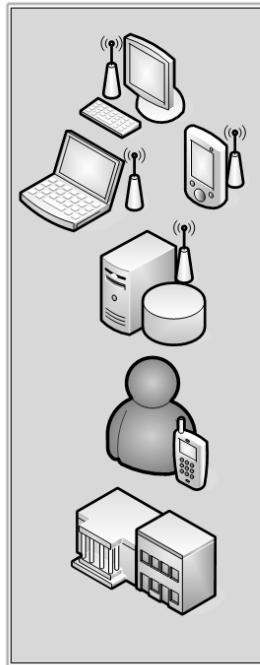
Modül Özeti

- ◆ Veri sağlayıcılar ne işe yarar? Connection nesnelerinin kullanılması için gereken temel özellik hangisidir?
- ◆ Bağlantı açıkken tekrar açılmaya çalışıldığında ne olur?
- ◆ Try Catch bloğu ne için kullanılır?
- ◆ Bağlantı durumu Connection nesnesinin hangi özelliği ile anlaşılır?



1. Veri sağlayıcılar ne işe yarar? SQL Server 6.5 veri tabanına bağlanmak için hangi veri sağlayıcının kullanılması gereklidir.
2. **Connection** nesnelerinin kullanılması için gereken temel özellik hangisidir?
3. Bağlantı açıkken tekrar açılmaya çalışıldığında ne olur? Bu durum nasıl engellenir?
4. **Try Catch** bloğu ne için kullanılır? **Finally** bloğunda hangi kodlar yazılır?
5. Bağlantı durumu **Connection** nesnesinin hangi özelliği ile anlaşılır? Durum değiştiği zaman hangi olay tetiklenir?

Lab 1: Bağlantı Oluşturmak



Uygulamalar

Veri Kaynaklarına Bağlanmak

- ◆ Bağlantı Oluşturmak

Bu labda, kullanıcını girdiği değerlere göre **Connection String** oluşturulur. Bu bağlantı cümlesi ile yeni bir **Connection** nesnesi oluşturularak, bağlantının durumu incelenir.

Bu lab tamamlandıktan sonra:

- Farklı veritabanlarına göre bağlantı cümlesi oluşturabilecek,
- Veritabanına bağlantı açıp kapayabilecek,
- Bağlantıların **State** özelliği ile durumunu gözlemleyebileceksiniz.

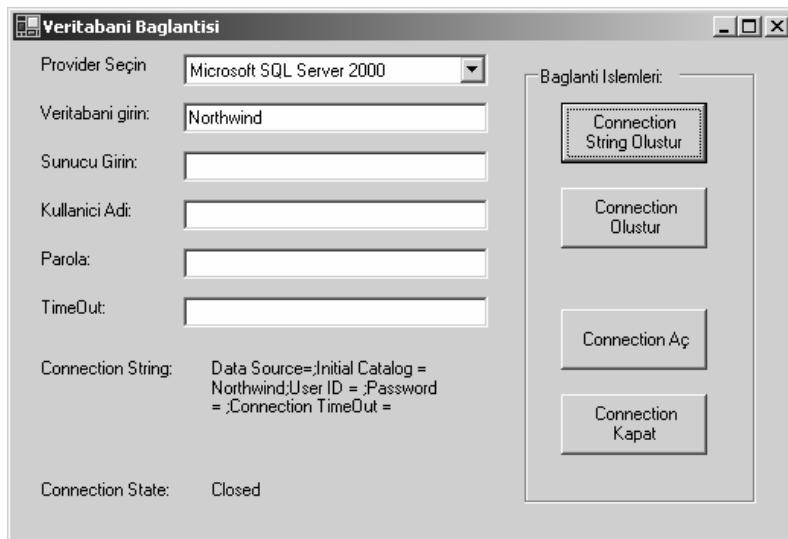
Kontrollerin eklenmesi

VeriTabaniBaglantisi isminde yeni bir Windows projesi açın.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
TextBox – txtVeriTabani	Text	
TextBox – txtSunucu	Text	
TextBox – txtKullaniciAdi	Text	
TextBox – txtParola	Text	
TextBox – txtTimeOut	Text	
Label – lblConnectionString	Text	
Label – lblConnectionState	Text	Closed

GroupBox	Text	Bağlantı İşlemleri:
ComboBox – ComboBox1	DropDownStyle	DropDownList
Button – Button1	Text	Connection String Oluştur
Button – Button2	Text	Connection Oluştur
Button – Button3	Text	Connection Aç
Button – Button4	Text	Connection Kapat



Kodların yazılması

Veritabanı bağlantısı oluşturmak için öncelikle bağlantı cümlesi oluşturulması gereklidir. Bu bağlantı cümlesi, kullanıcının gireceği bilgilere göre oluşturulur.

1. Bağlantı cümlesi için gerekli bilgileri tutan değişkenleri tanımlayın.

```
private string ConnectionString;
private string Provider;
private string Database;
private string Server;
private string KullaniciAdi;
private string Parola;
private String TimeOut;

// Sql veritabanına bağlanmak için
private System.Data.SqlClient.SqlConnection sqlCon;
// Access veritabanına bağlanmak için
private System.Data.OleDb.OleDbConnection oleDbCon;
```

2. Formun Load anında, **ComboBox** kontrolüne veritabanı seçeneklerini ekleyin.

```
private void Form1_Load(System.Object sender,
System.EventArgs e)
{
    comboBox1.Items.Add("Microsoft Access");
```

```

        comboBox1.Items.Add("Microsoft SQL Server 2000");
        comboBox1.Items.Add("Microsoft SQL Server 6.5");

        // Varsayılan olarak Sql Server 2000 seçili olur
        comboBox1.SelectedIndex = 1;
    }

```

3. **ComboBox** kontrolünden veritabanı seçildiği zaman, **Provider** ismini değiştirein. Seçilen veritabanı Access ise, kullanıcı adı, parola, sunucu ve timeout **TextBox** kontrollerini pasif duruma getirin.

```

private void comboBox1_SelectedIndexChanged(System.Object
sender, System.EventArgs e)
{
    // Farklı veritabanı seçildiği zaman tetiklenir.
    switch (comboBox1.SelectedIndex)
    {
        case 0:
            Provider = "Microsoft.Jet.OleDb.4.0";
            EnableTextBoxes(false);
            break;

        case 1:
            Provider = "";
            EnableTextBoxes(true);
            break;

        case 2:
            Provider = "SQLOLEDB";
            EnableTextBoxes(true);
            break;
    }
}

```

```

private void EnableTextBoxes(bool SQLVeriTabanıSecili)
{
    txtKullaniciAdı.Enabled = SQLVeriTabanıSecili;
    txtParola.Enabled = SQLVeriTabanıSecili;
    txtTimeOut.Enabled = SQLVeriTabanıSecili;
    txtSunucu.Enabled = SQLVeriTabanıSecili;

    // Access veritabanı seçili ise
    if (!SQLVeriTabanıSecili)
    {
        txtKullaniciAdı.Text = "";
        txtParola.Text = "";
        txtTimeOut.Text = "";
        txtSunucu.Text = "";
    }
}

```

4. Connection String oluşturma düğmesine basıldığı zaman, girilen değerleri alın ve bağlantı cümlesi oluşturun.

```

private void Button1_Click(System.Object
sender, System.EventArgs e)
{
    ConnectionString = "";

    // Access Veritabanı seçili ise
    if (comboBox1.SelectedIndex == 0)
    {
        Database = txtVeritabani.Text;
    }
}

```

```

        ConnectionString = "Provider= " + Provider +
                           ";Data Source= " + Database ;
    }
    else
    {
        KullaniciAdi = txtKullaniciAdi.Text;
        Parola = txtParola.Text;
        TimeOut = txtTimeOut.Text;
        Server = txtSunucu.Text;
        Database = txtVeritabani.Text;

        if (Provider != "")
        {
            ConnectionString = "Provider=" + Provider + ";";
        }

        ConnectionString += "Data Source=" + Server;
        ConnectionString += ";Initial Catalog=" + Database;
        ConnectionString += ";User ID=" + KullaniciAdi;
        ConnectionString += ";Password=" + Parola;
        ConnectionString += ";Connection Timeout=" + TimeOut;
    }

    lblConnectionString.Text = ConnectionString;
}

```

5. Bağlantı cümlesi oluşturulduktan sonra, bu cümleyi kullanarak **Connection** nesnesi oluşturun. Burada dikkat edilmesi gereken nokta, kullanıcının seçtiği veritabanı tipine göre farklı bağlantı nesnesi oluşturulmasıdır.

```

private void Button2_Click(System.Object sender,
System.EventArgs e)
{
    // Access veritabanı seçili ise
    if (comboBox1.SelectedIndex == 0)
    {
        oleDbCon = new
System.Data.OleDb.OleDbConnection(ConnectionString);
        oleDbCon.StateChange+= new
StateChangeEventHandler(DurumDegisti);
    }
    else
    {
        sqlCon = new
System.Data.SqlClient.SqlConnection(ConnectionString);
        sqlCon.StateChange += new
StateChangeEventHandler(DurumDegisti);
    }
}

// Bağlantı durumu değiştiği zaman, DurumDegisti
// yordamındaki kodlar çalışır.

private void DurumDegisti(Object sender,
System.Data.StateChangedEventArgs e)
{
    // Bağlantının durumu kullanıcıya bildirilir.
    lblConnectionState.Text = e.CurrentState.ToString();
}

```

6. Bağlantı nesnesi oluşturulduktan sonra, Connection Aç düğmesine basılıncı bağlantı **open** metodu ile açın. Ancak kullanıcının bazı değerleri yanlış girmesi durumu göz önüne alınarak **Try Catch** blokları kullanılmalıdır.

```
private void Button3_Click(System.Object sender,
System.EventArgs e)
{
    if (oleDbCon != null)
    {
        // OleDb bağlantısı oluşturulduysa
        try
        {
            oleDbCon.Open();
        }

        catch (InvalidOperationException ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
        catch (System.Data.OleDb.OleDbException ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
    }
    else
    {
        // Sql bağlantısı oluşturulduysa
        try
        {
            sqlCon.Open();
        }
        catch (InvalidOperationException ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
        catch (System.Data.SqlClient.SqlException ex)
        {
            MessageBox.Show(ex.Message.ToString());
        }
    }
}
```

7. Connection Kapat düğmesine basılıncı oluşturulan bağlantıyı bularak kapatın.

```
private void Button4_Click(System.Object sender,
System.EventArgs e)
{
    if (oleDbCon != null)
    {
        // OleDb bağlantısı oluşturulduysa
        oleDbCon.Close();
    }
    else
    {
        // Sql bağlantısı oluşturulduysa
        sqlCon.Close();
    }
}
```

Connected Veritabanı İşlemleri



Modül 4: Bağlantılı (Connected) Veritabanı İşlemleri

Connected Veritabanı İşlemleri

- ◆ Bağlantılı Veri Ortamlarıyla Çalışmak
- ◆ Command ile Çalışmak
- ◆ Command ile Geriye Değer Döndürmek
- ◆ Command ile Geriye Kayıt Döndürmek
- ◆ Command ile Kayıt Döndürmeyen Sorgular Çalıştırmak

Bu modülde ADO.NET ile bağlantılı veritabanı işlemlerin nasıl yapıldığını öğreneceksiniz.

Bu modülün sonunda:

- Bağlantılı veri ortamlarıyla çalışmayı öğrenecek,
- Command oluşturabilecek,
- Command ile geriye tek değer veya kayıt kumesi döndürebilecek,
- Command ile INSERT, UPDATE ve DELETE sorgularını çalıştırabilecek,

Konu 1: Bağlantılı Veri Ortamlarıyla Çalışmak

Bağlantılı Veri Ortamlarıyla Çalışmak

- ♦ Bağlantılı Uygulamalar İçin Veritabanı Mimarisi

Bağlantılı Uygulamalar İçin Veritabanı Mimarisi

Bağlantılı Uygulamalar İçin Veritabanı Mimarisi

- Uygulamaların veri kaynağına sürekli bağlı kaldığı ortamlardır.
- Veritabanı üzerinde yapılabilecek işlemler;
 - Veritabanından tek değer çekme
 - Sadece okunabilir kayıt kümeleri döndürme
 - Kayıt ekleme
 - Kayıt silme
 - Kayıt güncelleme

Bağlantılı veri ortamları, uygulamaların veri kaynağına sürekli bağlı kaldığı ortamlardır. Bu ortamlarda veri alma ve değiştirme işlemleri uygulama ile veri kaynağı arasında bağlantı kurulduktan sonra gerçekleştirilir.

Bağlantılı veri ortamları ile veritabanı üzerinde, gerekli tüm veritabanı işlemleri yapılabilir.

- Veritabanından tek değer çekme
- Sadece okunabilir kayıt kümeleri döndürme
- Kayıt ekleme
- Kayıt silme
- Kayıt güncelleme

Bağlantılı Veri Ortamı Sınıfları

Sınıf	Açıklama
XXXConnection	Bağlantı açmak ve kapatmak için kullanılan nesnedir.
XXXCommand	Veritabanı üzerinde Stored Procedure (Saklı Yordam) veya SQL Cümleleri çalıştırılmak için kullanılan nesnedir.
XXXDataReader	Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılan nesnedir. Kayıtlar XXXCommand nesnesinin ExecuteReader metodu ile XXXDataReader içerisine aktarılır.

Bağlantılı veri ortamları içerisinde kullanılan sınıflar Tablo 4.1 de belirtilmiştir.

Sınıf	Açıklama
XXXConnection	Bağlantı açmak ve kapatmak için kullanılan nesnedir.
XXXCommand	Veritabanı üzerinde Stored Procedure (Saklı Yordam) veya SQL Cümleleri çalıştırılmak için kullanılan nesnedir.
XXXDataReader	Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılan nesnedir. Kayıtlar XXXCommand nesnesinin ExecuteReader metodu ile XXXDataReader içerisine aktarılır.

Tablo 4.1. Bağlantılı Veri Ortamı Sınıfları

Konu 2: Command ile Çalışmak

Command ile Çalışmak

- ♦ Command Nedir?
- ♦ Command Oluşturmak
- ♦ Parametre Kullanmak

Command Nedir?

Command Nedir?

Command Nesneleri ile veritabanı tablolarında; sorgu, ekleme, silme ve güncelleme işlemleri yapılabilir.

Nesne	Veri Sağlayıcıları
System.Data.SqlClient.SqlCommand	SQL Server .NET Veri Sağlayıcısı
System.Data.OleDb.OleDbCommand	OLE DB .NET Veri Sağlayıcısı
System.Data.OleDb.ODBCCommand	ODBC .NET Veri Sağlayıcısı

Command, veritabanı üzerinde Stored Procedure (Saklı Yordam) ve Sorgu

çalıştırmak için kullanılır. Command Nesneleri ile veritabanı tablolarında; sorgu, ekleme, silme ve güncelleme işlemleri yapılabilir.

Tablo 4.2 de hangi veri sağlayıcı için hangi Command Nesnesinin kullanıldığı gösterilmektedir.

Tablo 4.2.

Command Nesneleri	Nesne	Veri Sağlayıcıları
	System.Data.SqlClient.SqlCommand	SQL Server .NET Veri Sağlayıcısı
	System.Data.OleDb.OleDbCommand	OLE DB .NET Veri Sağlayıcısı
Veritabanı Üzerinde	System.Data.OleDb.ODBCCommand	ODBC .NET Veri Sağlayıcısı

Stored Procedure ve Sorgu çalıştırmak için Command Nesnelerinin belirli özelliklerini kullanmak gereklidir. Command Nesnelerinin bu özellikleri aşağıda belirtilmiştir.

Command Nesnesinin Özellikleri

- **Connection**
 - Bağlantı nesnesi seçimi.
- **CommandType**
 - Text, Stored Procedure ve TableDirect
 - Yapılacak sorgunun türü.
- **CommandText**
 - Sorgu cümlesi veya Stored Procedure adı.
- **Parameters**
 - İsteğe bağlı parametrelerin kullanımı.

- **Name:** Command nesnesinin kod içerisindeki ismidir. Bu isim Command nesnesine başvurmak için kullanılabilir.
- **Connection:** Command nesnesinin hangi Connection üzerinde çalışacağını belirler.
- **CommandType:** Çalıştırılacak komutun türünü belirtir. **Text**, **Stored Procedure** ve **TableDirect** olmak üzere üç değeri vardır. **TableDirect** SQL Server tarafından desteklenmez.
- **CommandText:** Stored Procedure adını veya Sorgu cümlesini tutar.

- Parameters:** Command içerisinde çalıştırılacak Stored Procedure veya Sorgu cümlesine, dışardan değer almak ve dışarıya değer göndermek için kullanılır. Her bir Command Nesnesi için bir veya birden çok parametre tanımlanabilir.

Command Sınıfı Metotları

Command Sınıfının Metodu	Açıklama
ExecuteScalar	Tek bir değer döndürecek sorguları çalıştırır.
ExecuteReader	Birkaç satır bilgiyi döndürecek sorguları çalıştırır.
ExecuteNonQuery	Veri güncelleme yapılacağında çalıştırılır ve bu güncellemeden etkilenen satırların sayısını geri döndürür.
ExecuteXmlReader	Sorgu sonucunu XML olarak verir. Sadece SqlCommand nesnesinde bulunur.

Command özelliklerine değer girildikten sonra, Command’ı çalıştmak için Tablo 4.3 deki metotlardan uygun olan seçilir.

Command Metodu	Açıklama
ExecuteScalar	Çalıştırılan Command nesnesinden geriye tek değer döndürmek için kullanılır.
ExecuteReader	Çalıştırılan Command nesnesinden geriye kayıt kümesi döndürmek için kullanılır.
ExecuteNonQuery	Command Nesnesi üzerinde veri güncelleme, değiştirme ve silme işlemleri yapmak için kullanılır. Bu işlemin sonucunda etkilenen kayıt sayısı geriye döndürür.
ExecuteXmlReader	Çalıştırılan Command Nesnesinden geriye XML döndürmek için kullanılır. Sadece SQL Server 7.0 ve sonraki versiyonları için kullanılır.

Tablo 4.3: Command Metotları

Command Oluşturmak

- ### Command Oluşturmak
- Command, kod içerisinde veya ToolBox üzerinden oluşturulabilir.
 - Adımlar;
 - Veri kaynağuna bağlantı kurmak için Connection tanımlanır.
 - ToolBox'dan Data paneli seçilir.
 - Data panelindeki OleDbCommand veya SqlCommand nesnesi form üzerine sürüklendir.

Command, kod içerisinde veya ToolBox üzerinden oluşturulabilir. Bu yöntemler ile kullanılan veritabanına göre, **OleDbCommand** veya **SqlCommand** nesneleri oluşturulur.

Örnekte Access veritabanına bağlanmak için, **OleDbCommand** sınıfı tanımlanmış ve bu Command sınıfının gerekli özelliklerine değer atanmıştır.

```
//Access Veritabanına bağlanmak için Command tanımlanır.  
System.Data.OleDb.OleDbCommand cmd = new  
System.Data.OleDb.OleDbCommand();  
//Command Sınıfının CommandText özelliğine universiteler  
cmd.CommandText="select * from universiteler";  
  
//Command Sınıfının Connection özelliğine aktif connection  
//aktarılır  
cmd.Connection=conn;  
//Command Sınıfına, sorgu cümlesi yazılacağını belirler.  
cmd.CommandType=CommandType.Text;
```

ToolBox üzerinden Command nesnesi oluşturmak için belirtilen adımları takip edin.

- 1- Veri kaynağuna bağlantı kurmak için Connection tanımlanır.
- 2- ToolBox'dan Data paneli seçilir.
- 3- Data panelindeki **OleDbCommand** veya **SqlCommand** nesnesi form üzerine sürüklendir.

4- Eklenen Command nesnesinin özellikleri Tablo 4.4'e göre ayarlanır.

Özellik	Açıklama
Name	Command nesnesinin kod içerisinde kullanılan ismidir.
Connection	Command nesnesinin hangi Connection üzerinde çalışacağını belirler. Bu özellik ile yeni Connection oluşturabilir veya var olan Connection nesnesine bağlanılabilir.
 CommandType	Command nesnesinin tipini belirler. Çalıştırılacak Command'a göre Text , StoredProcedure ve TableDirect seçilir. Text: SQL Cümlesi StoredProcedure: Kayıtlı Yordam TableDirect: Tablo kayıtları TableDirect sadece OleDbCommand nesnesi tarafından kullanılır.
CommandText	Command nesnesinin çalıştırılacak komutudur. Seçilen CommandType'a göre CommandText belirlenir. Text: Çalıştırılacak SQL Cümlesi. StoredProcedure: Çalıştırılacak Stored Procedure adı. TableDirect: Veritabanındaki Tablo adı
Parameters	Command nesnesinin CommandText komutuna dışardan değer almak veya komuttan geriye değer döndürmek için kullanılır. Parameters özelliği Collection olduğu için bir veya birden çok değer alabilir veya gönderebilir.

Tablo 4.4: Command Özellikleri

Parametre Kullanmak

Parametre Kullanmak

Stored Procedure ve SQL Cümlelerinin parametre değerlerini saklamak için, **XXXCommand** nesnesinin **Parameters** özelliği kullanılır.

- **XxxParameter** nesneleri **XxxCommand'** in **Parameters** koleksiyonuna eklenir.
 - Yeni bir **XxxParameter** nesnesi oluşturulur.
 - **XxxParameter** nesnesinin özelliklerine değerler atanır.
 - **XxxCommand** Nesnesinin **Parameters** koleksiyonunun **Add** metodu kullanılır.

Stored Procedure ve SQL Cümleleri parametre alabilir veya gönderebilir. Ayrıca Stored Procedure geriye tek değer bile döndürebilir.

Stored Procedure ve SQL Cümlelerinin parametre değerlerini saklamak için, **XXXCommand** nesnesinin **Parameters** özelliği kullanılır. Aynı zamanda bu özellik **XXXParameters** nesnesinin koleksiyonudur.

Command nesnesi çalıştırılmadan önce, komuttaki her giriş parametresi için bir değer girilmelidir. Ayrıca Command nesnesi çalıştırıldıktan sonra, sonuç parametrelerinin değerleri geriye döndürülebilir.

Command nesnesine parametre eklemek için aşağıdaki yöntemler kullanılır.

- **XxxParameter** nesneleri oluşturulur ve Command nesnesinin **Parameters** koleksiyonuna bu nesneler eklenir.
- **Properties** penceresi kullanılarak Command nesnesinin **Parameters** özelliğine tasarım aşamasında parametreler eklenir.

XXXParameter nesnesini kullanarak, parametre eklemek için aşağıdaki adımlar takip edilir.

- 1- Yeni bir **OleDbParameter** veya **SqLParameter** nesnesi oluşturulur.
- 2- Eklenen **Parameter** nesnesinin özellikleri Tablo 4.5 göre ayarlanır.

Property	Açıklama
ParameterName	Parametrenin ismi, @Ad gibi
DbType ,SqlDbType, OleDbDbType	Parametrenin veri türü. Kullanılan veri tabanına göre SqlDbType veya OleDbType enumeratorlerinden seçilir.
Size	Parametredeki verinin byte olarak maksimum boyutu.
Direction	Parametrenin türü. ParameterDirection değerlerinden biri ile belirtilir. Bu özelliğin alabileceği değerler: ParameterDirection.Input (varsayılan değer) ParameterDirection.InputOutput ParameterDirection.Output ParameterDirection.ReturnValue

Tablo 4.5: XxxParameter Özellikleri

- 3- XxxParameter nesnelerini Command nesnesine eklemek için, Command nesnesinin **Parameters** koleksiyonunu içerisindeki **Add** metodu kullanılır. Eğer bu komut sonuç döndürecek bir stored procedure çağrırsrsa, herhangi bir parametre eklenmeden önce **ParameterDirection.ReturnValue** parametresini eklenmelidir. Parametrelerin eklenme sırası önemli değildir.

Parametre Kullanmak

XxxParameter nesnesinin özellikleri;

Property	Açıklama
ParameterName	Parametrenin ismi, @Ad gibi
DbType ,SqlDbType, OleDbDbType	Parametrenin veri türü. Kullanılan veri tabanına göre SqlDbType veya OleDbType enumeratorlerinden seçilir.
Size	Parametredeki verinin byte olarak maksimum boyutu.
Direction	Parametrenin türü. ParameterDirection enumeratorlerinden biri ile belirtilir. Bu özelliğin alabileceği değerler: ParameterDirection.Input (varsayılan değer) ParameterDirection.InputOutput ParameterDirection.Output ParameterDirection.ReturnValue

Tablo 4.6 da **Direction** özelliğinin değerleri listelenmiştir.

Enumeratör	Özellik
Input	Girdi parametresidir. Varsayılan değerdir.
Output	Çıktı parametresidir.
InputOutput	Girdi ve çıktı parametresi olarak kullanılır.
ReturnValue	Bir fonksiyon sonucunu geri döndürmek için kullanılır.

Tablo 4.6 Direction Özelliğinin Enumeratörleri

Örnekte EmployeeLogin isminde bir stored procedure için, paramUser, ve paramPass isminden iki **OleDbParameter** tanımlanmış ve bu parametreler Command nesnesinin **Parameters** koleksiyonuna eklenmiştir.

```
System.Data.OleDb.OleDbParameter paramUser = new
System.Data.OleDb.OleDbParameter("@userName",
System.Data.OleDb.OleDbType.Char, 50);
paramUser.Direction = ParameterDirection.Input;

System.Data.OleDb.OleDbParameter paramPass = new
System.Data.OleDb.OleDbParameter("@userPass",
System.Data.OleDb.OleDbType.Char, 20);
paramPass= ParameterDirection.Input;
cmd.Parameters.Add(paramUser);
cmd.Parameters.Add(paramPass);
```

ToolBox kontrollerini kullanarak parametre eklemek için, aşağıdaki adımlar takip edilir.

- 1- Toolbox üzerinden **OleDbCommand** veya **SqlCommand** nesnesi seçilir ve forma eklenir.
- 2- **Properties** penceresinden, bu Command nesnesinin **Connection**, **CommandType** ve **CommandText** özelliklerine değerler atanır.
- 3- **CommandText** özelliğine değer girildikten sonra, ekrana çıkan “Bu komut nesnesi için parametre eklemek ister misiniz?” ileti kutusuna Evet denir.
- 4- Visual Studio .NET ortamı Command nesnesi için parametre oluşturacak kodları otomatik olarak ekler.

Konu 3: Command ile Geriye Değer Döndürmek

Command ile Geriye Değer Döndürmek

Çalıştırılan stored procedure yada SQL cümlesinden geriye tek değer döndürülebilir.

Tek değer döndüren senaryolar :

- Stok tablosundaki toplam stok miktarını geriye döndürmek için.
- Ürün tablosundaki toplam kayıt sayısını geriye döndürmek için.
- Ürün kategorisine göre toplam ürün sayısını geriye döndürmek için.

Çalıştırılan stored procedure ya da SQL cümlesinden geriye tek değer döndürülebilir. Bu tür durumlar için DataSet yerine Command nesnesi kullanılmalıdır.

Command ile geriye tek değer döndüren senaryolara, aşağıdaki örnekler verilebilir.

- Stok tablosundaki toplam stok miktarını geriye döndürmek için
- Ürün tablosundaki toplam kayıt sayısını geriye döndürmek için
- Ürün kategorisine göre toplam ürün sayısını geriye döndürmek için

OleDbCommand veya **SqlCommand** nesnesi ile geriye değer döndürmek için, **ExecuteScalar** metodu kullanılır.

Örnekte **OleDbCommand** nesnesinin **ExecuteScalar** metodu ile Üniversiteler tablosundaki toplam kayıt sayısını geri döndürülmemektedir.

```
System.Data.OleDb.OleDbConnection conn = new  
System.Data.OleDb.OleDbConnection(@"provider =  
Microsoft.JET.OLEDB.4.0; Data source=..\universiteler.mdb");  
System.Data.OleDb.OleDbCommand cmd = new  
System.Data.OleDb.OleDbCommand("select count(*) from"  
+ "universiteler", conn);  
conn.Open();  
MessageBox.Show(cmd.ExecuteScalar.ToString());
```

ExecuteScalar metodu ile geriye değer döndürmek için, sadece **Sum** veya **Count** gibi fonksiyonlar kullanılmaz. Aynı zamanda Select cümlesi veya Stored

Procedure ile geriye tek değer döndürülebilir. Örnekte Ürün Tablosundaki Stok Miktarı **SqlCommand** nesnesi ile geriye döndürülmektedir.

```
string sql = "SELECT StokMiktarı FROM Urun WHERE UrunID=" +
    "@UrunID";
System.Data.SqlClient.SqlCommand cmUrun = new
System.Data.SqlClient.SqlCommand(sql,cnAlisveris);

System.Data.SqlClient.SqlParameter prmID = new
System.Data.SqlClient.SqlParameter("@UrunID",System.Data.Sql
DbType.Int, 4);
cmUrun.Parameters["@UrunID"].value = 42;

cnAlisveris.Open();
int adet = Convert.ToInt32(cmUrun.ExecuteScalar());
cnAlisveris.Close();

MessageBox.Show("Quantity in stock: " + adet.ToString());
```

Konu 4: Command ile Geriye Kayıt Döndürmek

Command ile Geriye Kayıt Döndürmek

Çalıştırılan stored procedure yada SQL cümlesi, geriye birden çok değer veya kayıt kümesi döndürülebilir.

Kayıt kümesi döndüren senaryolar :

- Müşteri tablosundan MüsteriID ye göre kayıt döndürmek
- Web Form üzerinde aranan ürünlerin, sonuçlarını döndürmek

Çalıştırılan stored procedure ya da SQL cümlesi, geriye birden çok değer veya kayıt kümesi döndürülebilir. Bu tür durumlar için **DataAdapter** veya **DataReader** nesneleri kullanılır. Bu nesnelerin genel farkı, **DataReader** bağlantılı, **DataAdapter** Bağlantısız veri ortamları için kullanılır.

DataReader nesnesinin kullanıldığı senaryolara, aşağıdaki örnekler verilebilir.

- Tablodan tek kayıt döndürmek.(Müşteri tablosundan MüsterİD ye göre kayıt döndürmek)
- Sadece okunur sonuçlar döndürmek. (Web Form üzerinde aranan ürünlerin, sonuçlarını döndürmek)

`OleDbCommand` veya `SqlCommand` nesnesi ile geriye kayıt döndürmek için, `ExecuteReader` metodu kullanılır. `ExecuteReader` ile dönen kayıtlar `DataReader` nesnesine aktarılır.

DataReader Özellik ve Metotları

`DataReader` dönen kayıtlar üzerinde işlem yapmayı sağlayan metod ve özelliklere sahiptir. `DataReader` nesnesinin bu özellik ve metotları aşağıda işlemleri yapar.

- Kayıt kümesi içindeki kayıtları tek tek okur.
- Kaydın belirli bir kolonunu veya tüm kolonlarını okur.
- Kolonların içerisinde değer olup olmadığını kontrol eder.
- Kolonların şema bilgilerini okur. (`ColumnName`, `ColumnOrdinal`, `ColumnSize`, `NumericPrecision`, `NumericScale`, `Datatype`, `ProviderType`, `IsLong`, `AllowDBNull`)

NOT: DataReader, verilere tek yönlü(forward-only) ve okunabilir (read-only) eriştiği için oldukça hızlıdır.

DataReader Nesnesinin Metotları

Metot	Açıklama
<code>Close()</code>	DataReader kapatılır ve hafızada yer tutmaz.
<code>Get(Veritürü)</code>	Belirli bir kolondaki değeri istenen veritüründe geri döndürür.
<code>GetString()</code>	Belirli bir kolondaki değeri string olarak geri döndürür.
<code>GetValue()</code>	Belirli bir kolondaki verinin doğal değerini alır.
<code>GetValues()</code>	Bulunan satirdaki tüm attribute kolonları alır.
<code>NextResult</code>	Komut metninde birden fazla SELECT ifadesi varsa, sonuçlar bu metotla iki ayrı küme gibi alınabilir.
<code>Read()</code>	DataReader'da okunacak kayıt olduğu sürece okuma yapar. Kayıt varsa True, yoksa False değerine geri döner.

`DataReader` nesnesinin metotları Tablo 4.7 de gösterilmiştir.

Metot	Açıklama
Close	DataReader nesnesini kapatılır ve hafızadan kaldırır.
GetBoolean	Belirli bir kolonun değerini bool olarak geri döndürür.
GetByte	Belirli bir kolonun değerini byte olarak geri döndürür.
GetBytes	Belirli bir kolonun değerini byte dizisi olarak geri döndürür.
GetChar	Belirli bir kolonun değerini char olarak geri döndürür.
GetChars	Belirli bir kolonun değerini karakter dizisi olarak geri döndürür.
GetDataTypeName	Belirli bir kolonun veri türünü verir.
GetDateTime	Belirli bir kolonun değerini DateTime olarak geri döndürür.
GetDecimal	Belirli bir kolonun değerini Decimal olarak geri döndürür.
GetDouble	Belirli bir kolonun değerini Double olarak geri döndürür.
GetFieldType	Belirli bir kolonun veri türünü geri döndürür.
GetFloat	Belirli bir kolonun değerini Float olarak geri döndürür.
GetGuid	Belirli bir kolonun değerini Globally-unique identifier(GUID) olarak geri döndürür.
GetInt16	Belirli bir kolonun değerini 16-bit tamsayı(Short) olarak geri döndürür.
GetInt32	Belirli bir kolonun değerini 32-bit tamsayı(Integer) olarak geri döndürür.
GetInt64	Belirli bir kolonun değerini 64-bit tamsayı(Long) olarak geri döndürür.
GetName	Belirli bir kolonun ismini geri döndürür.
GetOrdinal	Belirli bir kolonun sıra numarasını geri döndürür.
GetSchemaTable	DataReader nesnesinin şema bilgilerini gösterir. Tablolarındaki detay bilgilerini gösterir.
GetString	Belirli bir kolonun değerini string olarak geri döndürür.
GetTimeSpan	Belirli bir kolonun değerini TimeSpan nesnesi olarak geri döndürür.
GetValue	Belirli bir kolonun değerini geri döndürür.
GetValues	Belirli bir kaydın tüm kolon değerlerini geri döndürür.
NextResult	Komut metninde birden fazla SELECT ifade varsa, sonuçlar bu metot kullanılarak farklı veri kümeleri

	gibi alınabilir.
Read	DataReader nesnesinde okunacak kayıt olduğu sürece okuma yapar. Kayıt varsa True , yoksa False değeri geri döndürür.

Tablo 4.7 DataReader Metodları

DataReader, Bağlılı veri ortamlarında kullanıldığı için veri kaynağına sürekli bağlıdır. Bundan dolayı veri alış işlemi bittikten sonra **Connection** ya da **DataReader** nesnesi kapatılarak, belleğin daha etkin kullanılması sağlanır.

DataReader Nesnesinin Özellikleri

- Kaydın belirli bir kolonunu veya tüm kolonlarını okur.
- Kolonların içerisinde değer olup olmadığını kontrol eder.
- Kolonların şema bilgilerini okur. (**ColumnName**, **ColumnOrdinal**, **ColumnSize**, **NumericPrecision**, **NumericScale**, **Datatype**, **ProviderType**, **Islong**, **AllowDBNull**)

Özellik	Açıklama
FieldCount	DataReader içinde tutulan sütun sayısını belirtir.
IsClosed	DataReader'ın bağlantısının açık-kapalı durumunu belirtir.
Item	DataReader ile gelen verilere erişim sağlar.
RecordAffected	Delete , Update , Insert komutları sonucu kaç satırın bu komuttan etkilendiğini belirtir.

DataReader nesnesinin özellikleri Tablo 4.8 de gösterilmiştir.

Özellik	Açıklama
FieldCount	DataReader içinde tutulan sütun sayısını belirtir.
IsClosed	DataReader bağlantısının durumunu belirtir. Bağlantı açık ise FALSE , kapalı ise TRUE döndürür.
Item	DataReader ile gelen verilere erişim sağlar.
RecordAffected	DataReader ile gelen kayıt sayısını verir.

Tablo 4.8 DataReader Özellikleri

Adım Adım DataReader

- Bağlanılacak veritabanına göre Connection nesnesi eklenir.
- Bağlanılacak veritabanına göre OleDbCommand veya SqlCommand nesnesi eklenir ve gerekli özellikleri ayarlanır.
- Veritabanı bağlantısı açılır.
- DataReader tanımlanır. Command nesnesinin ExecuteReader() metodu ile çağrılan kayıtlar DataReader'a atanır.
- DataReader nesnesinin Read metodu False oluncaya kadar, kayıtlar döngü ile okunur ve Form kontrollerine aktarılır.
- DataReader kapatılır.

DataReader kullanarak kayıt çekmek için aşağıdaki adımları takip edin.

- 1- Bağlanılacak veritabanına göre **Connection** nesnesi eklenir
- 2- Bağlanılacak veritabanına göre **OleDbCommand** veya **SqlCommand** nesnesi eklenir ve gerekli özellikleri ayarlanır.
- 3- Veritabanı bağlantısı açılır.
- 4- **DataReader** tanımlanır. Command nesnesinin **ExecuteReader** metodu ile çağrılan kayıtlar **DataReader** nesnesine atanır.
- 5- **DataReader** nesnesinin **Read** metodu **False** oluncaya kadar, kayıtlar döngü ile okunur ve Form kontrollerine aktarılır.
- 6- **DataReader** kapatılır.

Örnekte Ürün Tablosundaki tüm ürünler, **OleDbDataReader** ile form üzerindeki **ListBox** kontrolüne eklenir.

```
System.Data.OleDb.OleDbCommand cmUrun = new  
System.Data.OleDb.OleDbCommand("SELECT UrunAdi, StokMiktari"  
+"FROM Urun", cnAlisveris);  
cnAlisveris.Open();  
System.Data.OleDb.OleDbDataReader rdrUrun;  
  
rdrUrun =  
cmUrun.ExecuteReader(CommandBehavior.CloseConnection);  
  
while (rdrUrun.Read())  
{  
    listBox1.Items.Add(rdrUrun.GetString(0)+" - "+  
    rdrUrun.GetInt16(1));  
}  
rdrUrun.Close();
```

Konu 5: Command ile Kayıt Döndürmeyen Sorgular Çalıştırmak

Command ile Kayıt Döndürmeyen Sorgular Çalıştırmak

ExecuteNonQuery Metodu ile Select hariç SQL ve T-SQL komutları kullanılır.

ExecuteNonQuery metodu ile DDL ve DCL komutları çalıştırılmak için aşağıdaki adımları takip edin,

- Bağlanılacak veritabanına göre Connection nesnesi eklenir
- Bağlanılacak veritabanına göre OleDbCommand veya SqlCommand nesnesi eklenir ve gerekli özelliklerini ayarlanır.
- Veritabanı bağlantısı açılır.
- Command nesnesinin ExecuteNonQuery metodu kullanılır.
- Veritabanı bağlantısı kapatılır.

Command ile veritabanı yapısında değişiklik yapılabilir (Tablo, View ve Stored Procedure oluşturmak, değiştirmek ve silmek), güvenlik seçenekleri ayarlanabilir (Tablo ve View izinleri) ve veritabanı içerisindeki veri değiştirilebilir (Kayıt ekleme, silme ve güncelleme). **OleDbCommand** veya **SqlCommand** nesnesi ile bu tür işlemlerin yapılabilmesi için, **ExecuteNonQuery** metodu kullanılır.

ExecuteNonQuery Metodu ile Select hariç SQL (Structured Query Language) ve T-SQL (Transact- Structured Query Language) komutları kullanılır.

SQL "Structured Query Language" (Yapilandırılmış Sorgulama Dili) veritabanı sorgu dilidir. SQL ile veritabanına kayıt ekleme, silme, var olan kaydı düzenleme ve kayıt sorgulama işlemleri yapılabilir. SQL standart bir veritabanı sorgu dilidir ve Oracle, db2, Sybase, Informix, Microsoft SQL Server, MS Access gibi veritabanı yönetim sistemlerinin temelini oluşturur. En sık kullanılan SQL komutları SELECT, INSERT, UPDATE ve DELETE komutlarıdır.

SQL Server'ın sorgulama ve programlama diline T-SQL denir. Transact-SQL ile ilişkisel veritabanı sistemi yönetilebilir. Transact-SQL komutları kullanım amaçlarına göre üç genel kategoriye ayrılır.

Bunlar:

- **SQL Veri İşleme Dili (Data Manipulation Language-DML)**

SQL Veri İşleme Dili; veri girmek, değiştirmek, silmek ve verileri almak için kullanılır. En sık kullanılan DML komutları ve kullanım amaçları Tablo 4.9 gösterilmiştir.

DML Komutu	Açıklama
SELECT	Veri seçmek
DELETE	Veri silmek
UPDATE	Veri güncellemek
INSERT	Veri girmek

Tablo 4.9 DML Komutları

- **SQL Veri Tanımlama Dili (Data Definition Language-DDL)**

SQL Veri Tanımlama Dili; Veritabanı nesnelerini yaratmak, silmek ve bazı temel özelliklerinin düzenlemek için kullanılır. En sık kullanılan DDL komutları ve kullanım amaçları Tablo 4.10'da gösterilmiştir.

DDL Komutu	Açıklama
CREATE	Yeni bir veritabanı nesnesi yaratmak. Örnek CREATE TABLE, CREATE TRIGGER
ALTER	Veritabanı nesnelerinde değişiklik yapmak. Örnek ALTER TABLE, ALTER TRIGGER
DROP	Veritabanı nesnelerini silmek. Örnek DELETE TABLE, DELETE TRIGGER

Tablo 4.10 DDL Komutları

- **SQL Veri Kontrol Dili (Data Control Language-DCL)**

SQL Veri Kontrol Dili; bir veritabanı kullanıcısı veya rolü ile ilgili izinlerin düzenlenmesini sağlar. Aşağıdaki tablo DCL komutlarını ve fonksiyonlarını göstermektedir.

DCL Komutu	Açıklama
GRANT	Kullanıcıya yetki vermek
DENY	Kullanıcı, grup veya rolü herhangi bir eylem için engeller.
REVOKE	Daha atanmış olan yetki veya engeli kaldırır.

Tablo 4.11 DCL Komutları

ExecuteNonQuery metodu ile DDL ve DCL komutları çalıştırılmak için aşağıdaki adımları takip edin

- 1- Bağlanılacak veritabanına göre Connection nesnesi eklenir
- 2- Bağlanılacak veritabanına göre **OleDbCommand** veya **SqlCommand** nesnesi eklenir ve gerekli özellikleri ayarlanır.
- 3- Veritabanı bağlantısı açılır.
- 4- Command nesnesinin **ExecuteNonQuery** metodu kullanılır.
- 5- Veritabanı bağlantısı kapatılır.

Örnekte **SqlCommand** nesnesinin **ExecuteNonQuery** metodu çalıştırılarak, Ürün isminde tablo oluşturulmaktadır.

```
System.Data.SqlClient.SqlCommand cmUruntabloOlustur = new
System.Data.SqlClient.SqlCommand("CREATE TABLE Urun (UrunID"
+ "int, UrunAdi varchar(50), StokMiktari int)",
connAlisveris);

cmUruntabloOlustur.CommandType = CommandType.Text;
try
{
    connAlisveris.Open();
    int kayitSayisi;
    kayitSayisi=cmUruntabloOlustur.ExecuteNonQuery();
    connAlisveris.Close();
    MessageBox.Show(kayitSayisi+" kayıt eklendi.");
}
catch(Exception ex)
{
    MessageBox.Show(ex.Message.ToString());
}
```

ExecuteNonQuery metodu ile INSERT, UPDATE ve DELETE sorguları çalıştırılabilir.

Örnekte **ExecuteNonQuery** metodu çalıştırılarak, Ürün tablosuna yeni kayıt eklenmiştir.

```
System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand("INSERT INTO Urun"+
"(UrunID,UrunAdi,StokMiktari) VALUES" +
"(@UrunID,@UrunAdi,@StokMiktari)", connAlisveris);
cmd.Parameters.Add("@UrunID", 1);
cmd.Parameters.Add("@UrunAdi", "DVD");
cmd.Parameters.Add("@StokMiktari", 15);
try
{
    connAlisveris.Open();
    int kayitSayisi = cmd.ExecuteNonQuery();
    connAlisveris.Close();
    MessageBox.Show(kayitSayisi+" kayıt eklendi.");
}
catch(Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

Örnekte **ExecuteNonQuery** metodu çalıştırılarak, Ürün tablosunda kayıt güncellenmiştir.

```
System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand("Update Urun Set" +
"StokMiktari = @StokMiktari where UrunID = @UrunID",
connAlisveris);
cmd.Parameters.Add("@UrunID", 1);
cmd.Parameters.Add("@StokMiktari", 30);
try
{
    connAlisveris.Open();
    int kayitSayisi = cmd.ExecuteNonQuery();
    connAlisveris.Close();
    MessageBox.Show(kayitSayisi+" kayıt değiştirildi.");
```

```
        }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
```

Örnekte **ExecuteNonQuery** metodu çalıştırılarak, Ürün tablosundan kayıt silinmiştir.

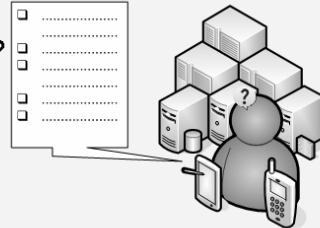
```
System.Data.SqlClient.SqlCommand cmd = new
System.Data.SqlClient.SqlCommand("Delete Urun Where"+
"UrunID= @UrunID", connAlisveris);
cmd.Parameters.Add("@UrunID", 1);

try
{
    connAlisveris.Open();
    int kayitSayisi=cmd.ExecuteNonQuery();
    connAlisveris.Close();
    MessageBox.Show(kayitSayisi+" kayıt silindi.");
}
catch(Exception ex)
{
    MessageBox.Show(ex.Message);
}
```

Modül Özeti

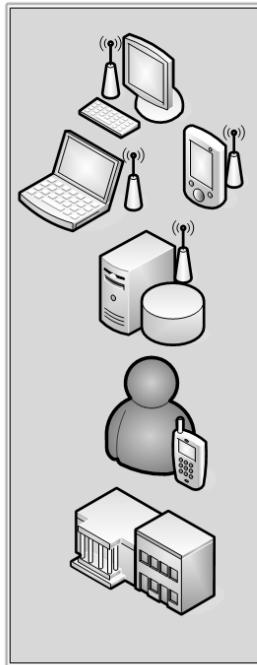
Modül Özeti

- ◆ Bağlantılı veri ortamı hangi .NET nesneleri ile gerçekleştirilir?
- ◆ Command nesnesinin kaç farklı çalışma biçimi vardır? Açıklayın.
- ◆ Command nesnesinin **Parameters** özelliği ne için kullanılır?
- ◆ DataReader nesnesinin çalışma modelini bir örnekle açıklayın.
- ◆ Kaç farklı SQL sorgu türü vardır?



1. Bağlantılı veri ortamı hangi .NET nesneleri ile gerçekleştirilir?
2. Command nesnesinin kaç farklı çalışma biçimi vardır? Açıklayın.
3. Command nesnesinin **Parameters** özelliği ne için kullanılır?
4. DataReader nesnesinin çalışma modelini bir örnekle açıklayın.
5. Kaç farklı SQL sorgu türü vardır?

Lab 1: Veritabanı İşlemleri



Uygulamalar

Veritabanı İşlemleri

- ◆ Veritabanının oluşturulması
- ◆ Kontrollerin eklenmesi
- ◆ Kodların yazılması
 - ExecuteNonQuery metodu
 - ExecuteReader ve DataReader
 - Form Kontrolleri işlemleri
 - Yordamların Formda kullanılması

Bu uygulamada, veritabanındaki Personel tablosu üzerinde kayıt işlemleri gerçekleştirilir. Personel kayıtlarının okunup **TextBox** kontrollerine doldurulması, yeni personel kaydının eklenmesi, bir personelin bilgilerinin güncellenmesi veya silinmesi işlemleri yapılır.

Bu lab tamamlandıktan sonra:

- Access veritabanına bağlantı oluşturabilecek,
- Command nesnesinin **ExecuteNonQuery** metodu ile INSERT, DELETE sorgusu çalıştırabilecek,
- Sorgulara parametre ekleyebilecek,
- Command nesnesinin **ExecuteReader** metodu ile **DataReader** oluşturabilecek,
- **DataReader** nesnesi ile kayıt okuyabileceksiniz.

Veritabanının oluşturulması

Bu uygulamada kullanılacak Personel tablosu için bir veritabanı oluşturulması gereklidir.

1. Microsoft Access ile “kisi” isminde bir veritabanı oluşturun.
2. Veritabanına Personel isminde bir tablo ekleyin ve tabloda belirtilen kolonları ekleyin.

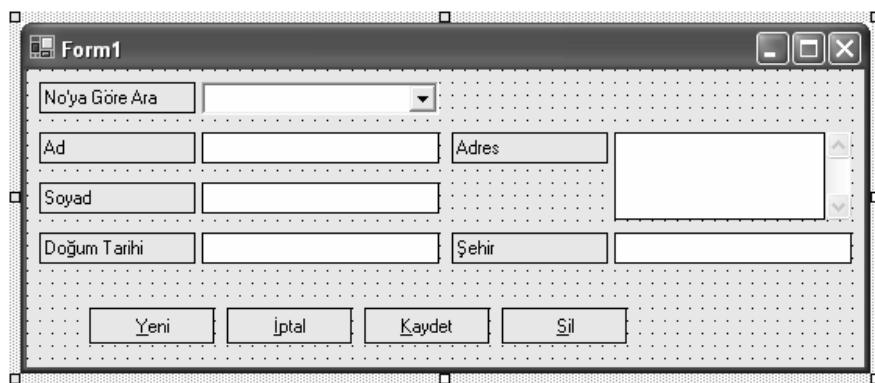
Alan Adı	Veri Türü
Numara	AutoNumber
Ad	Text
Soyad	Text
DogumTarihi	Date/Time
Adres	Text
Sehir	Text

Kontrollerin eklenmesi

Personel isminde yeni bir Windows projesi açın.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
TextBox – txtAd	BorderStyle	FixedSingle
TextBox – txtSoyad	BorderStyle	FixedSingle
TextBox – txtDTarihi	BorderStyle	FixedSingle
TextBox – txtSehir	BorderStyle	FixedSingle
TextBox – txtAdres	BorderStyle	FixedSingle
	Multiline	True
	ScrollBars	Vertical
ComboBox – cbNo	DropDownStyle	DropDownList
Button – btnYeni	Text	Yeni
Button – btnIptal	Text	Iptal
Button – btnKaydet	Text	Kaydet
Button – btnSil	Text	Sil



Kodların yazılması

Personel tablosu üzerinde işlem yapılması için veritabanına bağlantı yapılması gereklidir. Bu bağlantı için gereken Connection String ifadesinin merkezi bir yerden alınması, değişiklik durumunda kolaylık sağlayacaktır.

1. Projeye bir modül ekleyin ve bağlantı dizisini tanımlayın.

```
public string connstr=@"Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=C:\Samples\kisi.mdb";
```

DİKKAT: Bu aşamadan sonra kodlar, Formun kod tarafına yazılacaktır.

ExecuteNonQuery metodu

2. Veritabanına yeni bir Personel kaydı eklemek için gereken kodları bir yordam altında yazın.

```
public void Kaydet()
{
    System.Data.OleDb.OleDbConnection conn = new
    System.Data.OleDb.OleDbConnection();
    conn.ConnectionString = Module1.connStr;

    System.Data.OleDb.OleDbCommand comm = new
    System.Data.OleDb.OleDbCommand();
    comm.Connection = conn
    comm.CommandType = CommandType.Text;
    comm.CommandText = "INSERT INTO"+
        "Personel(Ad,Soyad,DogumTarihi,Adres,Sehir)+"+
        "values(@ad,@soyad,@tarih,@adres,@sehir)";
    comm.Parameters.Add("@ad", txtAd.Text);
    comm.Parameters.Add("@soyad", txtSoyad.Text);
    comm.Parameters.Add("@tarih", txtDTarihi.Text);
    comm.Parameters.Add("@adres", txtAdres.Text);
    comm.Parameters.Add("@sehir", txtSehir.Text);
    try
    {
        conn.Open();
        comm.ExecuteNonQuery();
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

```

finally
{
    conn.Close();
}
}

```

3. Verilen bir Personel numarasına göre tablodan kayıt silme işlemini gerçekleştiren kodları yazın.

```

public void Sil(int ID)
{
    System.Data.OleDb.OleDbConnection conn = new
    System.Data.OleDb.OleDbConnection();
    conn.ConnectionString = Module1.connStr;

    System.Data.OleDb.OleDbCommand comm = new
    System.Data.OleDb.OleDbCommand();
    comm.Connection = conn;
    comm.CommandType = CommandType.Text;
    comm.CommandText = "Delete from Personel Where"+
    "Numara=@No";
    comm.Parameters.Add("@No", ID);
    try
    {
        conn.Open();
        comm.ExecuteNonQuery();
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        conn.Close();
    }
}

```

ExecuteReader ve DataReader

4. **ComboBox** kontrolüne personel numaralarını dolduran kodları yazın. Bu **ComboBox**, personel kayıtlarını numaraya göre seçmek için kullanılacaktır.

```

public void IDDoldur()
{
    cbNo.Items.Clear();

    System.Data.OleDb.OleDbConnection conn = new
    System.Data.OleDb.OleDbConnection();
    conn.ConnectionString = Module1.connStr;
    System.Data.OleDb.OleDbCommand comm = new
    System.Data.OleDb.OleDbCommand();
    comm.Connection = conn;
    comm.CommandType = CommandType.Text;
    comm.CommandText = "Select Numara from Personel";
    System.Data.OleDb.OleDbDataReader dr;

    try
    {
        conn.Open();
        dr = comm.ExecuteReader();
    }
}

```

```

        while(dr.Read())
    {
        cbNo.Items.Add(dr.GetInt32(0));
    }
}
catch(Exception ex)
{
    MessageBox.Show(ex.Message);
}
finally
{
    dr.Close();
    conn.Close();
}

}
}

```

5. ComboBox kontrolünden seçilen personel numarasına göre formdaki kontrollerin doldurulmasını sağlayan kodları yazın.

```

public void IDyeGoreFormDoldur(int ID)
{
    System.Data.OleDb.OleDbConnection conn = new
System.Data.OleDb.OleDbConnection();
    conn.ConnectionString = Module1.connstr;
    System.Data.OleDb.OleDbCommand comm = new
System.Data.OleDb.OleDbCommand();
    comm.Connection = conn;
    comm.CommandType = CommandType.Text;
    comm.CommandText = "Select Ad,Soyad,Adres, Sehir, "+
"DogumTarihi from Personel where Numara=@No";
    comm.Parameters.Add("@No", ID);
    System.Data.OleDb.OleDbDataReader dr;
    try
    {
        conn.Open();
        dr = comm.ExecuteReader();
        if(dr.Read())
        {
            txtAd.Text = dr.GetString(0);
            txtSoyad.Text = dr.GetString(1);
            txtAdres.Text = dr.GetString(2);
            txtSehir.Text = dr.GetString(3);
            txtDTarihi.Text = dr.GetString(4);
        }
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        dr.Close();
        conn.Close();
    }
}

```

Form Kontrolleri İşlemleri

6. Formdaki TextBox kontrollerinin değerlerini sıfırlayan kodları yazın.

```

public void Temizle()
{
}

```

```

Control kontrol = new Control();
foreach(TextBox kontrol in this.Controls)
{
    kontrol.Text="";
}

txtAd.Focus();
}

```

7. Kayıt eklenmeden önce, tüm değerlerin doğru girilmesini kontrol eden kodları yazın.

```

public bool Kontrol()
{
    if(txtAd.Text == "")
    {
        MessageBox.Show("Adı Giriniz");
        txtAd.Focus();
        return false;
    }
    else if(txtSoyad.Text == "")
    {
        MessageBox.Show("Soyadı Giriniz");
        txtSoyad.Focus();
        return false;
    }
    else if(txtDTarihi.Text = "")
    {

        MessageBox.Show("Doğum Tarihini Giriniz");
        txtDTarihi.Focus();
        return false;
    }
    else if(txtAdres.Text = "")
    {
        MessageBox.Show("Adresi Giriniz");
        txtAdres.Focus();
        return false;
    }
    else if(txtSehir.Text = "")
    {
        MessageBox.Show("Şehiri Giriniz");
        txtSehir.Focus();
        return false;
    }
    else
    {
        return true;
    }
}

```

Yordamların Formda kullanılması

8. btnKaydet düğmesinin **Click** olayına Kaydet ve Kontrol yordamlarını kullanarak kaydetme işlemlerini yazın.

```

private void btnKaydet_Click(System.Object sender,
System.EventArgs e)
{
    if(Kontrol == true)

```

```

    {
        Kaydet();
        btnYeni.Enabled = true;
        btnKaydet.Enabled = false;
        btnIptal.Enabled = false;
        IDDoldur();
        cbNo.SelectedIndex = cbNo.Items.Count - 1;
    }

}

```

9. btnYeni düğmesinin **Click** olayında formu, yeni kayıt eklemek için hazırlayan Temizle yordamını kullanın.

```

private void btnYeni_Click(System.Object sender,
System.EventArgs e)
{
    Temizle();
    btnYeni.Enabled = false;
    btnKaydet.Enabled = true;
    btnIptal.Enabled = true;
    cbNo.SelectedIndex = - 1;
}

```

10. btnIptal düğmesine basıldığı zaman kontrolleri temizleyen ve ilk kayıta dönen kodları yazın.

```

private void btnIptal_Click(System.Object sender,
System.EventArgs e)
{
    Temizle();
    btnYeni.Enabled = true;
    btnKaydet.Enabled = false;
    btnIptal.Enabled = false;
    cbNo.SelectedIndex = 0;
}

```

11. btnSil düğmesine basıldığı zaman kayıt silme işlemleri gerçekleştiren yordamı kullanın.

```

private void btnSil_Click(System.Object sender,
System.EventArgs e)
{
    if(MessageBox.Show(cbNo.SelectedItem + " nolu kaydı silmek istiyor musunuz?", this.Text,
    MessageBoxButtons.YesNo, MessageBoxIcon.Question,
    MessageBoxDefaultButton.Button2) == DialogResult.Yes)
    {
        Sil(cbNo.SelectedItem);
        IDDoldur();
        cbNo.SelectedIndex = cbNo.Items.Count - 1;
    }
}

```

12. Formun **Load** olayında **ComboBox** kontrolünü personel numaraları ile dolduran yordamı kullanın.

```

private void frmPersonel_Load(System.Object sender,
System.EventArgs e)
{
    IDDoldur();
    cbNo.SelectedIndex = 0;
}

```

13. **ComboBox** kontrolünde bir personel numarası seçildiğinde, bu personele ait bilgileri forma yükleyen kodları yazın.

```
private void cbNo_SelectedIndexChanged(System.Object sender,
System.EventArgs e)
{
    IDyeGoreFormDoldur(cbNo.SelectedItem);
}
```

Bağlantısız (Disconnected) Veritabanı İşlemleri



Modül 5: Bağlantısız (Disconnected) Veritabanı İşlemleri

Bağlantısız (Disconnected) Veritabanı İşlemleri

- ◆ Disconnected Uygulamalar İçin Veritabanı Mimarisi
- ◆ DataSet ve DataTable Oluşturmak
- ◆ DataAdapter ile kayıtları Dataset'e doldurmak
- ◆ DataSet nesnesini Kontrollere bağlamak
- ◆ DataTable Üzerindeki Veriyi Düzenlemek
- ◆ Veri Arama ve Sıralama

Bağlantısız veri ortamları, uygulamaların veritabanından bağımsız çalıştığı ortamlardır. Veritabanı sunucusunun uzak olması, veri işlemlerinin uzun

sürmesi ve mobil çalışma ihtiyacı, bağlantısız veri ortamlarına olan ihtiyacı artırmıştır.

Bu modül tamamlandıktan sonra:

- Bağlantısız veritabanı mimarisini öğrenecek,
- **DataAdapter** nesnesinin yapısını tanıyacak
- **DataSet** nesne modelini öğrenecek
- **DataTable** nesne modelini öğrenecek
- Veri arama ve sıralama işlemlerini öğreneceksiniz.

Konu 1: Disconnected Uygulamalar İçin Veritabanı Mimarisi

Disconnected Uygulamalar İçin Veritabanı Mimarisi

- Veri kaynağının istenen bölümü çekilerek belleğe alınır.
- Veri üzerinde gerekli işlemler gerçekleştirildikten sonra, veri kaynağına aktarılarak güncelleme yapılır.

Bağlantısız veri ortamı, uygulamanın veri kaynağuna sürekli bağlı kalmadığı veri ortamıdır. Bu modelde, veri kaynağının istenen bölümü çekilerek belleğe alınır. Veri üzerinde gerekli işlemler gerçekleştirildikten sonra, veri kaynağına aktarılarak güncelleme yapılır.

Bağlantısız Veri Ortamı Sınıfları

Sınıf	Açıklama
XXXDataAdapter	Connection, Command ve DataReader sınıflarını kullanarak, verilerin DataSet'e doldurulmasını ve DataSet de yapılan değişikliklerin veri tabanına kaydedilmesini sağlar.
XXXConnection	Bağlantı açmak ve kapatmak için kullanılan nesnedir.
XXXCommand	Veritabanı üzerinde Stored Procedure (Saklı Yordam) veya SQL Cümleleri çalıştırılmak için kullanılan nesnedir.
XXXDataReader	Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılır.

Bağlantısız veri ortamları içerisinde kullanılan sınıflar Tablo 5.1 de belirtilmiştir.

Sınıf	Açıklama
XXXDataAdapter	Connection, Command ve DataReader sınıflarını kullanarak, verilerin DataSet'e doldurulmasını ve DataSet de yapılan değişikliklerin veri tabanına kaydedilmesini sağlar. Örneğin SqlDataAdapter sınıfı SQL Server ile DataSet arasındaki etkileşimi sağlar.
XXXConnection	Bağlantı açmak ve kapatmak için kullanılan nesnedir. Örneğin SqlConnection SQL Server a bağlantı sağlar.
XXXCommand	Veritabanı üzerinde Stored Procedure (Saklı Yordam) veya SQL Cümleleri çalıştırılmak için kullanılan nesnedir. Örneğin SqlCommand SQL Server üzerinde Stored Procedure veya SQL Cümleleri çalıştırmayı sağlar.
XXXDataReader	Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılır. Örneğin SqlDataReader ile SQL Server üzerinden kayıtlar okunur. Kayıtlar SqlCommand nesnesinin ExecuteReader metodu ile DataReader'a aktarılır.

Tablo 5.1. Bağlantısız Veri Ortamı Sınıfları

Konu 2: DataSet ve DataTable Oluşturmak

DataSet ve DataTable Oluşturmak

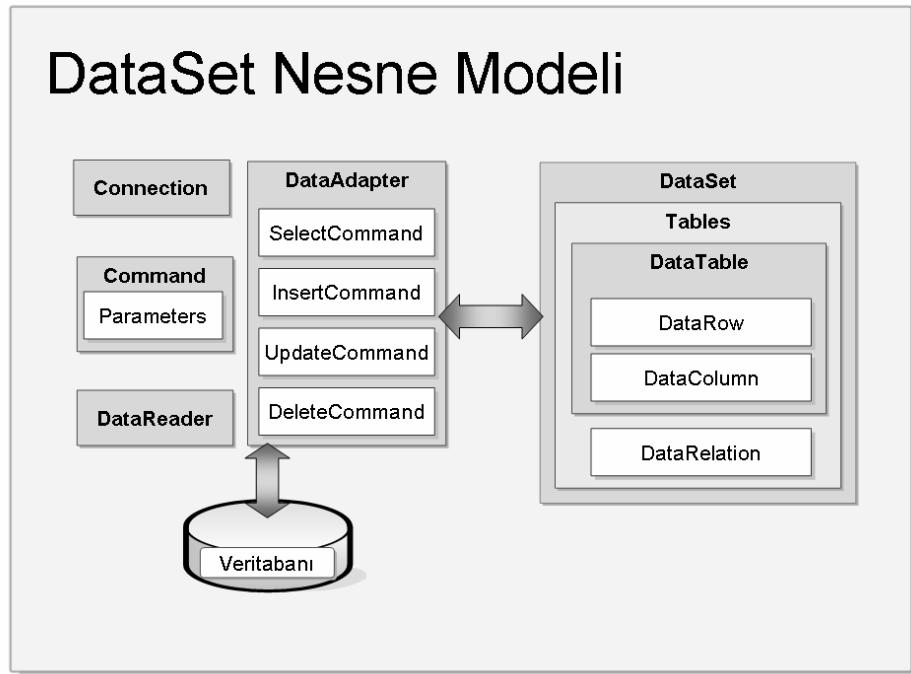
- Veri sağlayıcı türünden bağımsız çalışır.
 - DataSet tüm veri sağlayıcıları ile kullanılabilir.
- Sürekli çevrimidisidir.
 - DataAdapter nesnesi ile veriler DataSet içerisinde aktarılır ve bağlantı kapatılır.
- Değişikliklerin kaydını tutar.
 - DataSet içerisinde yapılan tüm değişiklikler, DataAdapter nesnesi ile veri kaynağına aktarılır.
- Birden fazla tablo bulundurabilir.
 - Birden fazla tablo ve ilişkileri hafızada tutmanın tek yolu DataSet kullanmaktadır.

Veri kaynağından DataAdapter ile çekilen verilerin, çekirdek belleğe atılan kopyası DataSet içerisinde saklanır. DataSet ile bu veriler üzerinde gerekli düzenlemeler yapıldıktan sonra, veriler aynı DataAdapter ile veritabanına aktarılır.

DataSet'in temel özellikleri aşağıda listelenmiştir:

- **Veri sağlayıcı türünden bağımsız çalışır:** DataSet tüm verisağlayıcıları ile kullanılabilir. Tamamen türden bağımsız çalışır.
- **Sürekli çevrimidisidir:** DataAdapter nesnesi ile veriler DataSet içerisinde aktarılır ve bağlantı kapatılır. Bağlantı kesildikten sonra yapılan tüm değişiklikler DataSet içerisinde kaydedilir. Bu durum uygulamanın çevrimidisi çalışmasını sağlar.
- **Değişikliklerin kaydını tutar:** DataSet içerisinde yapılan tüm değişiklikler, DataAdapter nesnesi ile veri kaynağına aktarılır.
- **Birden fazla tablo bulundurabilir:** İlişkisel veri tabanlarında olduğu gibi, birden fazla tablo ve ilişkileri hafızada tutmanın tek yolu DataSet kullanmaktadır.

DataSet Nesne Modeli



DataSet, Sanal bir veritabanı yapısını temsil eder. **DataTable** nesnelerinden oluşur. Bu tablolar arasında ilişkiler tanımlanabilir. DataSet'i oluşturan nesneler: DataTable, DataColumn, DataRow, DataRelation nesneleridir.

DataSet Nesne Modeli

▪ DataTable

- Veritabanı tablolarını temsil eder.
DataColumn, DataRow nesnelerinden oluşur.
Primary Key alanı tanımlanabilir.

▪ DataColumn

- DataTable nesnelerini oluşturmak için gereken kolonları temsil eder.

- DataTable

Veritabanı tablolarını temsil eder. **DataColumn**, **DataRow** nesnelerinden oluşur. **Primary Key** alanı tanımlanabilir.

- **DataColumn**

DataTable nesnelerini oluşturmak için gereken kolonları temsil eder.

DataSet Nesne Modeli

▪ **DataRow**

- **DataTable** nesneleri için veri satırlarını temsil eder.

▪ **DataRelationship**

- Tablolar arasındaki ilişkileri temsil eder.

▪ **DataView**

- **DataTable** nesneleri üzerinde filtreleme, veri güncellelemeleri işlemleri yapmak için kullanılır.

- **DataRow**

DataTable nesneleri için veri satırlarını temsil eder.

- **DataRelationship**

Tablolar arasındaki ilişkileri temsil eder.

- **DataView**

DataTable nesneleri üzerinde filtreleme, veri güncellelemeleri işlemleri yapmak için kullanılır.

Örnekte ds isminde yeni bir DataSet, **New** anahtar sözcüğü tanımlanmaktadır. Tanımlamada DataSet'e parametre olarak girilen "YeniDataSet" değeri, DataSet nesnesinin DataSetName argümanıdır. Eğer hiçbir isim verilmezse varsayılan olarak "NewDataSet" ismi verilir.

```
DataSet ds = new DataSet("Yeni Dataset");
```

DataSet, diğer bir DataSet nesnesinden kopyalanarak oluşturulabilir. DataSet kopyalamak için iki yöntem kullanılır. Birinci yöntem **Copy** metodu ile diğer bir DataSet nesnesinin, veri ve ilişkileri (şeme bilgileri) kopyalanarak yeni bir DataSet oluşturmak. İkinci yöntem **Clone** metodu ile diğer bir DataSet nesnesinin şema bilgilerini kopyalanarak, yeni bir DataSet oluşturmak. Bu yöntem şablon kopyalamak için kullanılır.

Örnekte ds ismindeki DataSet nesnesinin tüm tablo, ilişki ve verileri dsCopy ismindeki DataSet nesnesinin içerisine aktarılmıştır

```
DataSet dsCopy;  
dsCopy = ds.Copy();
```

Örnekte ds ismindeki DataSet nesnesinin tüm tablo ve ilişkileri, dsClone ismindeki DataSet nesnesinin içerisine aktarılmıştır.

```
DataSet dsClone;  
dsClone = cd.Clone();
```

Örnekte Stok veritabanını içerisindeki tüm kitaplar, DataAdapter nesnesi ile DataSet'e aktarılmıştır.

```
OleDbConnection conn = new OleDbConnection ("provider=" +  
"microsoft.jet.oledb.4.0; data source=../stok.mdb");  
  
OleDbDataAdapter da = new OleDbDataAdapter("select * from "  
+ "kitaplar", conn);  
  
DataSet ds = new DataSet("Set");  
  
da.Fill(ds,"Kitaplar")  
DataGrid1.DataSource = ds.Tables["Kitaplar"];
```

DataSet sınıfının Tables koleksiyonu ile DataSet içerisine bir veya birden çok DataTable eklenebilir. DataSet sınıfının Relations koleksiyonu ile DataSet içerisine bir veya birden çok DataRelation eklenebilir.

Örnekte dtKitaplar isminde yeni bir DataTable oluşturulmaktadır.

```
DataTable dtKitaplar = new DataTable("Kitaplar");
```

Oluşturulan tabloyu DataSet içerisine eklemek için DataSet nesnesinin Tables koleksiyonu kullanılır.

```
ds.Tables.Add(dtKitaplar);
```

DataTable nesnesinin içerisine kolon eklenebilir. Örnekte dtKitaplar ismindeki DataTable nesnesinin içerisine, yenid isminde yeni bir kolon eklenmektedir. Yeni kolon eklemek için, DataTable nesnesinin Columns koleksiyonu kullanılır.

```
 DataColumn colKitapID = dtKitaplar.Columns.Add("yeniId");
```

Örnekte DataTable nesnesi için Ucret, KDV ve Tutar isminde 3 adet kolon oluşturulmuştur. Örnekteki KDV kolonu, Ucret kolonun %17 değeri üzerinden hesaplanır. Tutar kolonu ise Ucret ve KDV değerinin toplamı ile hesaplanır.

```
DataTable dt = new DataTable();
dt.Columns.Add("Ucret");
dt.Columns.Add("KDV");
dt.Columns.Add("Tutar");
dt.Columns[0].Expression = "Ucret * 0.17";
dt.Columns[1].Expression = "Ucret + KDV";
```

Konu 3 : DataAdapter ile kayıtları Dataset'e doldurmak

DataAdapter ile kayıtları Dataset' e doldurmak

- DataAdapter sınıfı,
 - Verilerin DataSet nesnesine aktarılmasını sağlar.
 - DataSet üzerinde yapılan değişikliklerin veri kaynağına aktarılmasını sağlar.

DataAdapter sınıfı, DataSet ile veri kaynağı arasında köprü oluşturur. Veri kaynağına yapılan bağlantı ile verilerin DataSet nesnesine aktarılmasını sağlar. DataAdapter ayrıca DataSet üzerinde yapılan değişikliklerin veri kaynağına aktarılmasını sağlar.

Örnekte OleDbDataAdapter ile çekilen veriler, ds ismindeki DataSet nesnesine aktarılır. DataSet içerisindeki veriler, DataGrid ile ekranada gösterilir.

```
oledbConnection conn = new oledbConnection ("provider = "
+ "microsoft.jetoledb.4.0; data source=C:\Stok.mdb");

oledbDataAdapter da = new oledbDataAdapter("select * from
kitaplar", conn);

DataSet ds = new DataSet();

da.Fill(ds,"Kitaplar");
```

```
DataGrid1.DataSource= ds.Tables["Kitaplar"];
```

DataAdapter ile veri çekmek için DataAdapter nesnesinin başlangıç fonksiyonuna, SELECT sorgu ve bağlantı nesnesi parametre olarak gönderilir.

```
oleDbDataAdapter da = new OleDbDataAdapter("select * from  
" + "kitaplar", conn);
```

DataAdapter ile veri çekmenin diğer bir yöntemi SELECT sorgu ile Command nesnesi oluşturmaktır. Oluşturulan Command, DataAdapter nesnesinin SelectCommand özelliğine atanır. Örnekte Command ile DataAdapter nesnesinin beraber kullanımı gösterilmektedir.

```
OleDbConnection conn As = new OleDbConnection ("provider = "  
+ "microsoft.jet.oledb.4.0; data source=C:\Stok.mdb");  
  
OleDbCommand cmd = new OleDbCommand("select * from  
kitaplar");  
cmd.CommandType = CommandType.Text;  
cmd.Connection = conn;  
  
OleDbDataAdapter da = new OleDbDataAdapter(cmd);  
DataSet ds = new DataSet();  
  
da.Fill(ds, "kitaplar");  
DataGrid1.DataSource = ds.Tables["kitaplar"];
```

DataAdapter nesnesinin Fill metodu veri kaynağındaki veriyi DataSet veya DataTable nesnesini doldurmak için kullanılır. Örnekte da isimli DataAdapter ile çekilen veriler, Kitaplar tablosuna doldurulmaktadır.

```
da.Fill(ds, "kitaplar");
```

Bir DataSet içinde birden fazla tablo bulunabilir. Bu durumda DataAdapter nesnesinin Fill metodunu birden çok kez çağrılır.

Fill metodu ile belirli kayıt aralığı DataSet içerisinde aktarılabilir. Örnekte da isimli DataAdapter ile çekilen ilk altı kayıt, Kitaplar tablosuna aktarılır.

```
da.Fill(ds, 0, 5, "kitaplar");
```

DataSet üzerinde yapılan değişiklikleri veri kaynağına aktarmak için, DataAdapter sınıfının Update metodu kullanılır. DataAdapter nesnesinin DeleteCommand, UpdateCommand ve InsertCommand nesneleri içinde tutulan sorgular ile güncelleme işlemi gerçekleştirilir. Örnekte Sipariş tablosundaki tüm değişiklikler veri kaynağına aktarılmaktadır.

```
da.Update(ds, "siparisler");
```

Konu4: DataSet nesnesini Kontrollere bağlamak

DataSet nesnesini Kontrollere bağlamak

- ◆ DataSet İçerisideki Veriyi Windows Kontrollerine Bağlamak
- ◆ DataSet İçerisindeki Veriyi DataGrid'e Bağlamak

DataSet nesnesi ile veritabanının bir kopyası çekirdek belleğe atıldıktan sonra, bu veriler çeşitli Windows Kontroller ile gösterilebilir veya değiştirilebilir. Bu kontrollerin en önemlisi DataGrid bileşenidir.

DataSet İçerisideki Veriyi Windows Kontrollerine Bağlamak

DataSet İçerisideki Veriyi Windows Kontrollerine Bağlamak

- **Basit (simple data binding)**
 - Bir tek veri elemanını Windows kontrolerine bağlama
 - TextBox, Label, RadioButton....
- **Karmaşık (complex data binding)**
 - Birden fazla veri elemanını Windows kontrollere bağlama
 - DataGrid, ListBox, ErrorProvider....

DataSet nesnesin içeriği veri, Windows Form içerisindeki herhangi bir kontrolun herhangi bir özelliğine bağlanabilir. Örneğin DataSet içerisindeki bir tablonun ilk satır ve sütundaki veri, TextBox kontrolünün Text özelliğine bağlanabilir.

DataSet içerisindeki veriyi Windows kontrollere bağlamaının iki yöntemi vardır. Bu yöntemler basit (simple data binding) ve karmaşık (complex data binding) veri bağlama olarak adlandırılır.

Basit veri bağlama; DataSet içerisindeki bir veri elemanını (DataTable kolonunu) Windows kontrolere bağlama işlemidir. TextBox, Label, RadioButton gibi kontroller bu gruba girer. Örneğin DataSet tablosundaki herhangi bir kolonu TextBox, Label gibi Windows kontrollere bağlamak.

Karmaşık veri bağlama; DataSet içerisindeki birden fazla veri elemanını Windows kontrollere bağlama işlemidir. DataGrid, ListBox, ErrorProvider gibi kontroller bu gruba girer.

Örnekte Dataset içerisindeki kitap_baslik kolonun değeri, **TextBox** ve **Label** kontrollerin **Text** özelliğine aktarılır.

```
TextBox1.Text =  
    ds.Tables["kitaplar"].Rows[2].Item["kitap_baslik"];  
  
Label1.Text =  
    ds.Tables["kitaplar"].Rows[2].Item["kitap_baslik"];
```

ComboBox ve ListBox gibi kontrollere veri bağlamak için **DataSource** ve **DataMember** özelliği kullanılır. DataSouce özelliği DataSet içerisindeki tablo ismini DisplayMember ise Tablo kolonunu belirtir.

Örnekte ComboBox ve ListBox kontrolünün DataSource ve DisplayMember özellikleri kullanılmaktadır.

```
ComboBox1.DataSource = ds.Tables["kitaplar"];  
ComboBox1.DisplayMember =  
    ds.Tables["kitaplar"].Columns["kitap_baslik"].ToString();  
  
ListBox1.DataSource = ds.Tables["kitaplar"];  
ListBox1.DisplayMember =  
    ds.Tables["kitaplar"].Columns["kitap_baslik"].ToString();
```

TreeView kontrolüne veri bağlamak için, TreeNode nesnesinin Text özelliği kullanılır.

```
Treeview1.Nodes[0].Text =  
ds.Tables["kitaplar"].Rows[1].Item["kitap_baslik"];
```

Örnekte DataSet nesnesinden gelen veriler ListView ve CheckedListBox kontrollerine aktarılmıştır.

```
int count ;  
Count = ds.Tables["kitaplar"].Columns.Count;  
  
for (int i=0;i< count;i++)  
{  
    ListView1.Items.Add(ds.Tables["kitaplar"].Rows[i][0].  
ToString());  
}  
  
for (int i=0;i<count;i++)  
{  
    CheckedListBox1.Items.Add(ds.Tables["kitaplar"].Rows[i][0].  
ToString());  
}
```

DataSet İçerisindeki Veriyi DataGrid'e Bağlamak

DataSet İçerisindeki Veriyi DataGrid'e Bağlamak

- Grafiksel Yöntem
 - Araç kutusu üzerindeki DataGrid kontrolünü form üzerine sürükleyin.
 - DataGrid kontrolünün DataSource özelliğini, önceden oluşturulmuş DataSet nesnesine bağlayın.
 - DataGrid kontrolününDataMember özelliğini , DataSet tablolarının herhangi biri ile bağlayın.
- Programlama yöntemi
 - `DataGrid1.DataSource = ds.Tables["Kitaplar"]`

DataAdapter, veriyi satırlar ve sütunlar halinde görüntüler. DataGrid ile ilişkisiz DataSet Tabloları kolay bir şekilde görüntülenebilir. Bu görüntü excel tablolarına benzemektedir.

DataGrid ilişkili DataSet tablolarında gösterebilir. Bu durumda istenilen tabloya DataGrid üzerindeki gezinti köprülerinden erişilebilir.

DataSet tablolarını DataGrid kontrolune bağlamak için iki yöntem kullanılır. Bu yöntemler grafiksel ve programlama yöntemleridir.

Grafiksel yöntem ile bağlantı sağlamak için aşağıdaki adımlar takip edin.

- Araç kutusu üzerindeki DataGrid kontrolü form üzerine sürükleyin.
- DataGrid kontrolünün DataSource özelliğini, önceden oluşturulmuş DataSet nesnesine bağlayın.
- DataGrid kontrolününDataMember özelliğini , DataSet tablolarının herhangi biri ile bağlayın.

Programlama yöntemi ile bağlantı sağlamak için DataGrid nesnesinin DataSource özelliği kullanılır. Örnekte, DataSet içerisindeki Kitaplar tablosu DataGrid kontrolune bağlanır.

`DataGrid1.DataSource = ds.Tables["Kitaplar"];`

Konu : 5 DataTable Üzerindeki Veriyi Düzenlemek

DataTable Üzerindeki Veriyi Düzenlemek

- DataTable nesnesine, yeni bir satır eklemek için DataRow nesnesi kullanılır.
- DataTable nesnesinin NewRow metodu ile oluşturulan yeni satır, DataRow nesnesinin değişkenine atanır.
- Index veya kolon isimleri üzerinden kolonlara değerler girilir.
- DataRow nesnesi DataTable nesnesinin Rows koleksiyonuna eklenir.

DataTable, veritabanı tablolarını temsil eder. **DataColumn**, **DataRow** nesnelerinden oluşur. DataSet içerisinde yeni bir DataTable oluşturulduktan sonra, veritabanı üzerinde işlem yapıyormuş gibi veri üzerinde düzenlemeler yapılabilir. DataTable, bu çevrimdışı düzenlemeleri kabul veya iptal etme olanağı sunar.

DataTable nesnesine, yeni bir satır eklemek için **DataRow** nesnesi kullanılır. DataTable nesnesinin NewRow metodu ile oluşturulan yeni satır, DataRow nesnesinin değişkenine atanır. Örnekte dtKitaplar tablosuna drNew isminde yeni bir kolon eklenmiştir.

```
DataRow drNew = dtKitaplar.NewRow();
```

DataRow nesnesi tanımlandıktan sonra, index veya kolon isimleri üzerinden kolonlara değer girilir. Örnekte birinci kolona kitabı ISBN numarası, ikinci kolona ise yazar adı bilgileri girilmiştir.

```
drNew[0] = "975-8725-14-9";  
drNew[1] = "Tamer Sahiner";
```

veya

```
drNew["kitap_ISBN"] = "975-8725-14-9";
```

```
drNew["kitap_yazar"] = "Tamer Şahiner";
```

Kolanlara bilgi girildikten sonra, tanımlanan DataRow nesnesi DataTable nesnesinin Rows koleksiyonuna eklenir. Örnekte drNew nesnesi, dtKitaplar nesnesinin Rows koleksiyonuna eklenir.

```
dtKitaplar.Rows.Add(drNew);
```

DataTable nesnesine DataRow kullanmadan kayıt eklenebilir. Örnekte dtKitaplar ismindeki DataTable nesnesine bu yöntem ile kayıt eklenmiştir.

```
dtKitaplar.Rows.Add(New Object[] {"975-8725-14-9", "Tamer Şahiner"});
```

DataTable Üzerindeki Veriyi Düzenlemek

- DataRow nesnesinin metodları;
- BeginEdit()
 - Veriyi düzenlerken oluşabilecek olayları askıya alır.
- EndEdit()
 - Askıya alınan olaylar yeniden aktif edilir.
- CancelEdit()
 - Değişikliklerden ve askıya alınan olaylardan vazgeçilir.

DataRow ile DataTable içerisindeki kayıtlar değiştirilebilir. DataRow nesnesi ile satır düzenleme işlemleri için aşağıdaki metodlar kullanılır.

- BeginEdit
- EndEdit
- CancelEdit

BeginEdit, veriyi düzenlerken oluşabilecek olayları askıya alır. Veriyi düzenlemek için **Items** koleksiyonu kullanılır. **EndEdit** metodu ile, askıya alınan olaylar yeniden aktif edilir. **CancelEdit** metodu ile değişikliklerden ve askıya alınan olaylardan vazgeçilir. Örnekte DataTable içerisindeki dördüncü kayıt için güncelleme işlemi yapılmıştır.

```
DataRow drNew = dtKitaplar.Rows[3];
drNew.BeginEdit();
drNew["kitap_baslik"] = "yeni hayat";
drNew["kitap_yazar"] = "can dündar";
drNew.EndEdit();
```

DataRow ile DataTable içerisindeki belirli bir satır silinebilir. Örnekte DataTable içerisindeki dördüncü kayıt silinmiştir.

```
DataRow drsil = dtKitaplar.Rows[3];
dtKitaplar.Rows.Remove(drsil);
```

DataRow nesnesinin Delete metodu kullanılarak aktif kayıt silinebilir.

```
drsil.Delete();
```

Windows Form ile Kayıt Üzerinde Hareket Sağlamak

Windows Form ile Kayıt Üzerinde Hareket Sağlamak

- DataSet, DataTable veya DataView ile kayıtlar üzerinde hareket sağlayan nesneye CurrencyManager denir.
- Belirli bir satıra gidebilmek için, CurrencyManager nesnesinin Position özelliği kullanılır.

Verileri düzenlemeden önce, hangi veri üzerinde düzenleme yapılacağının tespit edilmesi gereklidir. Windows Form uygulamaları, veri içinde hareket sağlanan nesneler ile verilerin bağlı olduğu katmanı yönetebilir. DataSet, DataTable veya DataView ile kayıtlar üzerinde hareket sağlayan nesneye CurrencyManager denir.

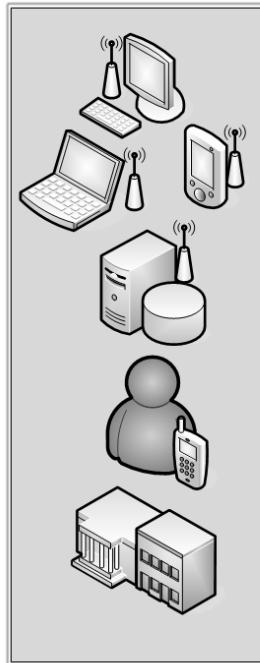
DataSet içinde çoklu veri kaynağı tutulabildiği için, birden fazla CurrencyManager nesnesi içerebilir.

Belirli bir satırı gidebilmek için, CurrencyManager nesnesinin Position özelliği kullanılır.

Örnekte dtKitaplar tablosunun kayıtları arasında ilk, son, önceki ve sonraki satırı hareket sağlanmıştır.

```
CurrencyManager cmKitaplar = new CurrencyManager();  
  
private void Form1_Load()  
{  
    txtKitapAdi.DataBindings.Add("Text", dtKitaplar, _  
        "kitap_baslik");  
    cmKitaplar = (CurrencyManager)BindingContext[dtKitaplar];  
    cmKitaplar.Position = 0;  
}  
  
private void btnMoveNext()  
{  
    if (cmKitaplar.Position != cmKitaplar.Count)  
    {  
        cmKitaplar.Position += 1;  
    }  
}  
private void btnMoveFirst()  
{  
    cmKitaplar.Position = 0;  
}  
  
private void btnMovePrevious()  
{  
    if (cmKitaplar.Position != 0)  
    {  
        cmKitaplar.Position -= 1;  
    }  
}  
  
private void btnMoveLast()  
{  
    cmKitaplar.Position = cmKitaplar.Count - 1;  
}
```

Lab 1: Bağlantısız Veritabanı İşlemleri



Uygulamalar

Bağlantısız (Disconnected) Veritabanı İşlemleri

- ◆ Veritabanının oluşturulması
- ◆ Kontrollerin eklenmesi
- ◆ DataSet verisinin kontrollere bağlanması
- ◆ Kodların yazılması

Bu uygulamada, veritabanındaki Personel tablosu üzerinde kayıt işlemleri gerçekleştirilir. Personel kayıtlarının okunup **DataGrid** kontrolüne doldurulması, kayıtlar arasında gezinti, yeni personel kaydının eklenmesi, bir personelin bilgilerinin güncellenmesi veya silinmesi işlemleri yapılır.

Bu lab tamamlandıktan sonra:

- Access veritabanına bağlantı oluşturabilecek,
- DataGrid kontrolüne kayıt doldurabilecek,
- DataGrid kontrolünü biçimlendirebilecek,
- Kayıtlar arasında dolaşabilecek,
- Kayıt ekleme, güncelleme ve silme işlemlerini DataSet içerisinde gerçekleştirebilecek.
- DataSet içerisindeki değişiklikleri veritabanına kaydedebileceksiniz.

Veritabanının oluşturulması

Bu uygulamada kullanılacak Personel tablosu için bir veritabanı oluşturulması gereklidir.

3. Microsoft Access ile "kisi" isminde bir veritabanı oluşturun.
4. Veritabanına Personel isminde bir tablo ekleyin ve tabloda belirtilen kolonları ekleyin.

Alan Adı	Veri Türü
Numara	AutoNumber
Ad	Text
Soyad	Text
DogumTarihi	Date/Time
Adres	Text
Sehir	Text

Kontrollerin eklenmesi

Baglantısız_Personel isminde yeni bir Windows projesi açın.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
DataGridView – dgPersonel	ReadOnly	True
TextBox – txtAd	BorderStyle	FixedSingle
TextBox – txtSoyad	BorderStyle	FixedSingle
TextBox – txtDogumTarihi	BorderStyle	FixedSingle
TextBox – txtSehir	BorderStyle	FixedSingle
TextBox – txtAdres	BorderStyle	FixedSingle
	Multiline	True
	ScrollBars	Vertical
Button – btnYeni	Text	Yeni
Button – btnIptal	Text	İptal
Button – btnKaydet	Text	Kaydet
Button – btnSil	Text	Sil
Button – btnVDoldur	Text	Veritabanından Getir
Button – btnVKaydet	Text	Veritabanına Kaydet
Button – btnCikis	Text	Çıkış

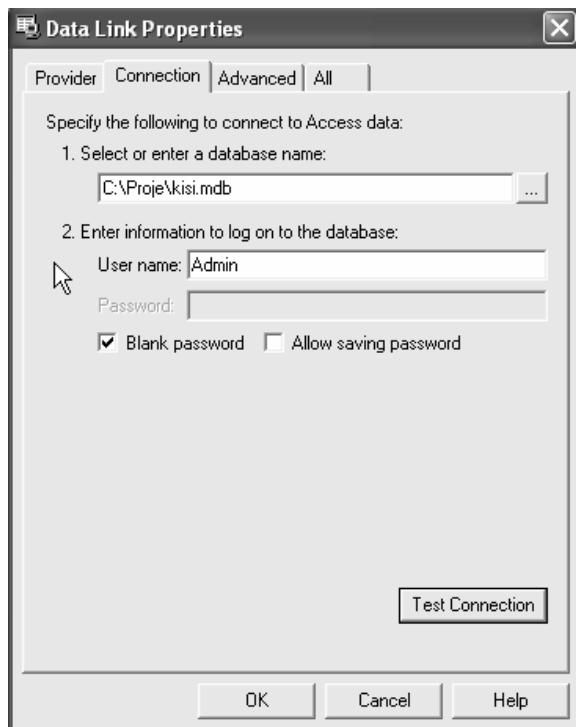


Bağlantı Cümlesinin oluşturulması

Personel tablosu üzerinde işlem yapılması için veritabanı bağlantısının kurulması gereklidir. Bu bağlantı için gereken Connection String cümlesini Server Explorer’ı kullanarak oluşturun.

Baglantısız_Personel uygulaması için yeni bağlantı oluşturmak.

1. **Server Explorer** penceresi üzerinde farenin sağ butonunu tıklayın. Açılan menüden **Add Connection komutunu** tıklayın.
2. Açılan **Data Link Properties** penceresinin **Provider** sekmesini tıklayın.
3. Provider sekmesinden **Microsoft.Jet.OLEDB.4.0 Provider** ‘i seçerek **Next** butonunu tıklayın.



4. Açılan **Connection** sekmesinin görüntüsünü resimdeki gibi düzenleyerek **OK** butonunu tıklayın.

Bağlantının Oluşturulması

Kisi veritabanına bağlantı sağlamak için **OleDbConnection** oluşturun.

1. Araç kutusu üzerindeki **OleDbConnection** kontrolü form üzerine sürükleyin.
2. **OleDbConnection** kontrolünün **ConnectionString** özelliği için oluşturduğunuz bağlantı cümlesini seçin.

DataAdapter nesnesinin Oluşturulması

Personel tablosunu DataSet içerisine aktarmak için OleDbDataAdapter oluşturun.

1. Araç kutusu üzerindeki **OleDbDataAdapter** kontrolü form üzerine sürükleyin.
2. Karşınıza çıkan **“Data Adapter Configuration Wizard”** penceresi üzerinde **Next** butonunu tıklayarak, bir sonraki adıma geçin.
3. **Choose Your Data Connection** penceresi üzerinde, oluşturduğunuz bağlantı cümlesini seçin ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
4. **Choose a Query Type** penceresinden **Use Sql statements** seçeneğini seçin ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
5. **Generate the Sql statements** penceresindeki metin kutusuna **SELECT Numara, Ad, Soyad, DogumTarihi, Adres, Sehir FROM Personel** yazın ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
6. **Finish** butonunu tıklayarak **“Data Adapter Configuration Wizard”** sihirbazını sonlandırın.

DataSet nesnesinin Oluşturulması

Personel kayıtları ile çevrimdışı çalışmak için DataSet oluşturun.

1. **da** üzerinde farenin sağ butonunu tıklayın.
2. Açılan kısayol menüsünden **Generate DataSet** menüsünü tıklayın.
3. **Choose a Dataset** menüsünden **New** seçeneğini seçin ve metin kutusuna ds yazın.
4. **OK** butonunu tıklayın

Dataset içerisindeki verinin DataGridViewe bağlanması

Personel tablosunu DataGridViewe bağlayın.

1. DgPersonel isimli DataGridView nesnesinin DataSource özelliğine ds1 isimli DataSet nesnesini seçin
2. DgPersonel isimli DataGridView nesnesininDataMember özelliğine Personel tablosunu seçin.

Dataset içerisindeki verinin TextBox kontrollerine bağlanması

Personel tablo içerisindeki Ad, Soyad, DTarihi, Adres ve Sehir kolonları sırayla txtAd, txtSoyad, txtDogumTarihi, txtAdres ve txtSehir metin kutularını bağlayın.

1. txtAd metin kutusunun **DataBindings** koleksiyonun **Text** özelliğine Ad kolonunu seçin.
2. txtSoyad, txtDogumTarihi, txtAdres ve txtSehir metin kutularını sırayla Soyad, DTarihi, Adres ve Sehir kolonlarına bağlayın.

Kodların Yazılması

14. btnDDoldur kontrolünün **Click** olayına kayıtları DataGridViewe dolduran kodları yazın.

```
try
{
    conn.Open();
    da.Fill(ds1, "Personel");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString());
}
finally
{
    conn.Close();
}
```

15. btnDKaydet kontrolünün **Click** olayına, DataSet kontrolundeki tüm değişiklikleri veritabanına kaydeden kodu yazın

```
try
{
    conn.Open();
    da.Update(Ds1, "Personel");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString());
}
finally
{
    conn.Close();
}
```

16. btnYeni kontrolunun **Click** olayına. DataSet tablosu için yeni satır oluşturan kodu yazın.

```
BindingContext(Ds1, "Personel").EndCurrentEdit();
BindingContext(Ds1, "Personel").AddNew();
```

17. btnIptal kontrolunun **Click** olayına. DataSet tablosu içinde eklenen yeni satırı iptal eden kodu yazın.

```
BindingContext(Ds1, "Personel").CancelCurrentEdit();
```

18. btnKaydet kontrolunun **Click** olayına. DataSet tablosuna yeni kayıt ekleyen kodu yazın.

```
BindingContext(Ds1, "Personel").EndCurrentEdit();
```

19. btnSil kontrolunun **Click** olayına. DataSet tablosundan aktif kaydı silen kodu yazın.

```
BindingContext(Ds1, "Personel").RemoveAt(BindingContext(Ds1, "Personel").Position);
```

20. btnIlk kontrolunun **Click** olayına. ilk kayda giden kodu yazın.

```
BindingContext(Ds1, "Personel").Position = 0;
```

21. btnOnceki kontrolunun **Click** olayına. önceki kayda giden kodu yazın.

```
BindingContext(Ds1, "Personel").Position -= 1;
```

22. btnSonraki kontrolunun **Click** olayına. sonraki kayda giden kodu yazın.

```
BindingContext(Ds1, "Personel").Position += 1;
```

23. btnSon kontrolunun **Click** olayına. son kayda giden kodu yazın.

```
BindingContext(Ds1, "Personel").Position =
BindingContext(Ds1, "Personel").Count-1;
```

Konu 6: Veri Arama ve Sıralama

Veri Arama ve Sıralama

- **DataTable.Rows.Find()**
 - Birincil anahtar değerine göre arama yapılmasını sağlar.
- **DataTable.Select()**
 - Belirli bir arama kriterine göre arama yapılmasını sağlar.
 - (FilterExpression, Sort, RecordStates)

DataSet içerisinde veri arama ve sıralama işlemleri yapmak için, DataTable ve DataView sınıfının arama ve sıralama metotları kullanılır.

DataTable sınıfının Find metodu, birincil anahtar değerine göre arama yapılmasını sağlar. Select metodu ise, belirli bir arama kriterine göre arama yapılmasını sağlar. Select metodu ile geriye satır koleksiyonları döndürülür.

Örnekte DataTable sınıfının Find metodu ile kitap barkod numarasına göre arama yapılmaktadır. Bu aramanın sonucunda yazar adı geriye döndürülür.

```
oleDbTypeConnection conn = new OleDbConnection("provider =" +  
    "microsoft.jet.oledb.4.0;data source=c:\Proje\stok.mdb");  
oleDbTypeAdapter da = new OleDbDataAdapter("select * from" +  
    "kitaplar", conn);  
  
DataTable tb1 = new DataTable;  
da.Fill(tb1);  
tb1.PrimaryKey = new DataColumn()  
{  
    tb1.Columns("kitap_ISBN");  
}  
  
DataRow row = tb1.Rows.Find("975-12-53-3");  
  
if (row==null)  
{  
    MessageBox.Show("Kayıt Bulunamadı!");  
}  
else
```

```

{
    MessageBox.Show(row["kitap_yazar"].ToString());
}

```

Örnekte Tbl ismindeki DataTable nesnesine, DataAdapter nesnesinden gelen kayıtlar doldurulur. Tbl tablosunun Kitap_ISBN kolonu birincil anahtar olarak atanır. Row isminden yeni bir DataRow tanımlanır. DataTable nesnesinin Find metoduna kitabıń barkod numarası girilir ve sonuç Row değişkenine aktarılır. Son olarak Row değişkeni içerisindeki sonuç ekrana yazılır.

Arama işlemlerinde özel karakter kullanılabilir. Örneğin yazar ismi 'E' harfi ile başlayan kayıtları sorgulamak için aşağıdaki komut kullanılır.

```
Select * from kitaplar where kitap_yazar Like 'E%'
```

DataTable sınıfının Select metodu ile DataRow nesnelerinden oluşan bir koleksiyon geri döndürür. Select metodu aslında orjinal DataSet içindeki satırları işaret eden işaretçiler kümesi olarak da algılanabilir. Veri kopyalama yapmaz ancak değişimleri görüntüler.

Örnekte DataTable sınıfının Select metodu kullanılarak, kitabıń fiyatı sekizden farklı kayıtlar gösterilmektedir. Bu kayıtlar Kitap isimlerine göre sıralanmıştır.

```

DataRow selRows = tbl.Select("kitap_fiyat != 8",
    "kitap_baslik ASC", DataViewRowState.CurrentRows);

foreach (DataRow row In selRows)
{
    MessageBox.Show("kitap: " + row["kitap_baslik",
        DataRowVersion.Original].ToString());
}

```

Select metodunun dört farklı kullanımı vardır. Bu kullanımlar aşağıda listelenmiştir.

```

public DataRow() Select()
public DataRow() Select(string filterExpression)
public DataRow() Select(string filterExpression,
    string sort)
public DataRow() Select(string filterExpression,
    string sort, recordStates as DataViewRowState)

```

Select metodunun **filterExpression**, **Sort** ve **recordStates** isminden üç parametresi bulunur.

filterExpression, filtreleme yapılacak ifadeyi içerir.

```
"Country = 'Turkey' AND City <> 'Ankara'"
```

Sort, sonuçların hangi sırada görüntüleneceğini belirtir.

```
"City DESC"
```

Veriler artan ve azalan olmak üzere iki şekilde sıralanabilir. Sıralanacak kolonun sonuna; azalan sıralama için DESC, artan sıralama için ASC anahtar sözcüğü yazılır.

recordStates ise, kayıtların durumuna göre, (Deleted, Modified gibi) seçim yapar.

DataView Özellik ve Metodları

DataView Özellik ve Metodları

▪ Grafiksel yöntem

- Araç kutusu üzerindeki DataView kontrolü form üzerine sürükleyin.
- DataView kontrolünün Table özelliği ile kullanılacak DataTable seçin.
- DataView kontrolünün Sort özelliğine, sıralanacak kolon bilgilerini girin.
- DataView kontrolünün RowFilter özelliğine arama sorgusunu girin.

ADO.NET ile veri kaynağından alınan bilgileri, sıralamak ve filtrelemek için DataView nesnesi kullanılır. DataView nesnesi ile DataTable nesneleri üzerinde arama veya sıralama işlemleri yapılabilir. DataView, diğer kontrollere bağlanabilen bir nesnedir.

DataView nesnesi oluşturmak için iki yöntem kullanılır. Bu yöntemler grafiksel ve programlama yöntemleridir.

Grafiksel yöntem ile bağlantı sağlamak için aşağıdaki adımlar takip edin.

- Araç kutusu üzerindeki DataView kontrolü form üzerine sürükleyin.

- DataView kontrolünün Table özelliği ile kullanılacak DataTable seçin.
- DataView kontrolünün Sort özelliğine, sıralanacak kolon bilgilerini girin.
- DataView kontrolünün RowFilter özelliğine arama sorgusunu girin.

DataView Özellik ve Metodları

▪ Programlama yöntemi

- DataView sınıfının Sort özelliğine, sıralanacak kolonun adı girilir.
- RowFilter özelliğine ise arama veya filtreleme sorgusu girilir.

```
DataTable dvProducts = new DataView(ds.Tables["Kitaplar"]);
dvProducts.Sort="kitap_yazar";
dvProducts.RowFilter="kitap_fiyat>8";
dataGridView1.DataSource=dvProducts;
```

Programlama yöntemi ile DataView kullanımı aşağıda gösterilmektedir. Örnekte fiyatı 8 YTL den büyük kayıtlar gösterilmektedir. Bu kayıtlar yazar adına göre sıralanarak gösterilir.

```
DataTable dvProducts = new DataView(ds.Tables["kitaplar"]);
dvProducts.Sort = "kitap_yazar";
dvProducts.RowFilter = "kitap_fiyat > 8";
dataGridView1.DataSource = dvProducts;
```

DataView sınıfının Sort özelliğine, sıralanacak kolonun adı girilir. RowFilter özelliğine ise arama veya filtreleme sorgusu girilir.

Örnekte Sipariş Numarası 10300 den büyük kayıtlar sorgulanmaktadır

```
dv.RowFilter = ("SiparisID >10300") ;
```

Örnekte Sipariş Tarihi 08/25/1996 tarihinden büyük olan kayıtlar sorgulanmaktadır.

```
dv.RowFilter = ("SiparisTarihi >#08/25/1996#");
```

Örnekte müşteri adı "V" ile başlayan kayıtlar sorgulanmaktadır.

```
dv.RowFilter = ("MusteriAdi like 'V*'");
```

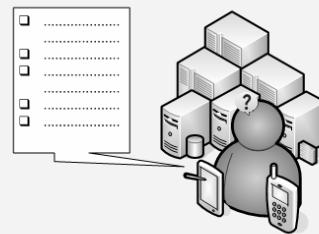
Örnekte müşteri adı "V" ile başlayan veya Sipariş Numarası 10300 den büyük kayıtlar sorgulanmaktadır.

```
dv.RowFilter = ("MusteriAdi like 'V*' or SiparisID >10300");
```

Modül Özeti

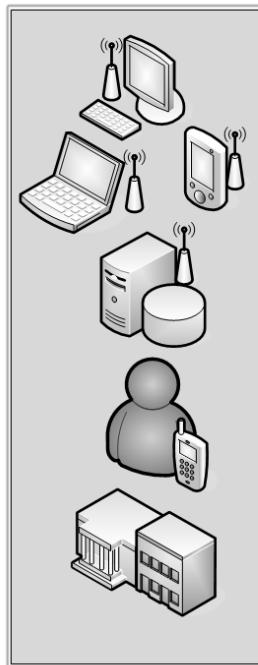
Modül Özeti

- Bağlantısız veri ortamı hangi .NET nesneleri ile gerçekleştirilir?
- DataSet nedir? DataSet hangi nesnelerden oluşur?
- DataTable nedir? Hangi durumlarda kullanılır ?
- DataColumn nedir? Hangi durumlarda kullanılır ?
- DataView Nedir ? Hangi durumlarda kullanılır ?



6. Bağlantısız veri ortamı hangi .NET nesneleri ile gerçekleştirilir?
7. DataSet nedir? DataSet hangi nesnelerden oluşur?
8. DataTable nedir? Hangi durumlarda kullanılır ?
9. DataColumn nedir? Hangi durumlarda kullanılır ?
10. DataView Nedir ? Hangi durumlarda kullanılır ?

Lab 2: Çoklu Tablolarla Çalışmak



Uygulamalar Çoklu Tablolarla Çalışmak

- ◆ Veritabanının oluşturulması
- ◆ Kontrollerin eklenmesi
- ◆ DataView nesnesinin Oluşturulması
- ◆ DataSet verisinin kontrollere bağlanması
- ◆ Kodların yazılması

Bu uygulamada aynı form üzerinde üç farklı DataTable ile çalışılacaktır. Bu uygulama ile Bolum tablosundaki kayıtların açılan kutuya doldurulması, seçilen bölüme göre öğrencilerin DataGrid doldurulması ve seçilen öğrenciye göre ders bilgilerinin DataGrid kontrolune doldurulması gerçekleştirilir. Form üzerindeki filtreleme işlemleri DataView kontrolü ile sağlanır.

Personel kayıtlarının okunup **DataGrid** kontrolune doldurulması, kayıtlar arasında gezinti, yeni personel kaydının eklenmesi, bir personelin bilgilerinin güncellenmesi veya silinmesi işlemleri yapılır

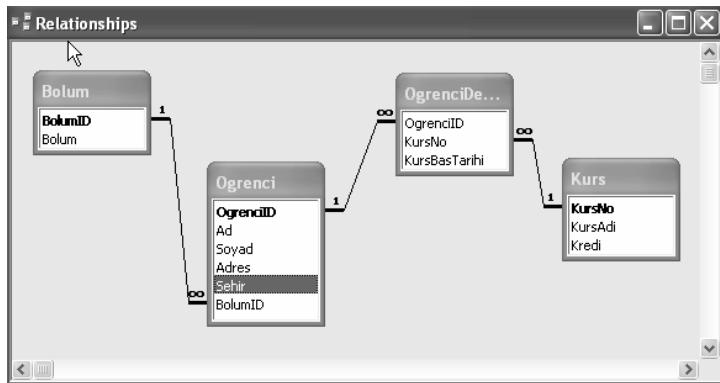
Bu lab tamamlandıktan sonra:

- Access veritabanına bağlantı oluşturabilecek,
- DataSet üzerinde birden fazla DataTable ile çalışabilecek,
- DataView ile filtreleme işlemleri yapabilecek,
- DataGrid kontrolüne kayıt doldurabileceksiniz.

Veritabanının projeye eklenmesi

Bu uygulamada kullanılacak Course veritabanı oluşturulur.

1. Microsoft Access ile “Dershane” isminde bir veritabanı oluşturun.
2. Veritabanın tablolarını aşağıdaki diyagrama göre oluşturun.

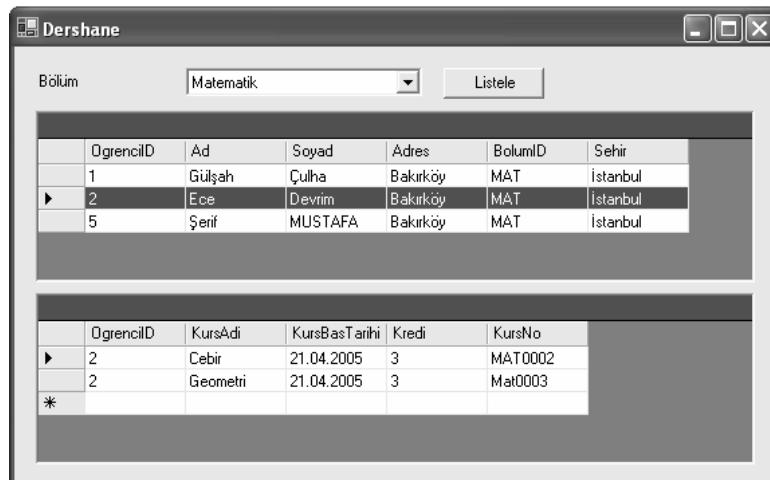


Kontrollerin eklenmesi

Dershane isminde yeni bir Windows projesi açın.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
DataGrid – dgOgrenci	ReadOnly	True
DataGrid – dgKurs	ReadOnly	True
ComboBox – cbBolum	DropDownStyle	DropDownList
Button – btnListele	Text	Listele
Label1 – Label1	Text	Bölüm

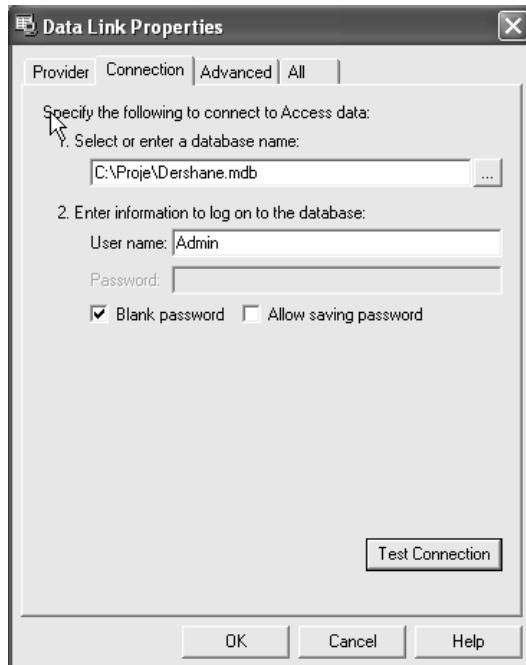


Bağlantı Cümlesinin oluşturulması

Dershane veritabanı üzerinde işlem yapılması için bağlantı kurulması gereklidir. Bu bağlantı için gerekli Connection String ifadesini Server Explorer’ı kullanarak oluşturun.

Dershane uygulaması için yeni bağlantı oluşturmak.

1. **Server Explorer** penceresi üzerinde farenin sağ butonunu tıklayın. Açılan menüden **Add Connection** komutunu tıklayın.
2. Açılan **Data Link Properties** penceresinin **Provider** sekmesini tıklayın.
3. Provider sekmesinden **Microsoft.Jet.OLEDB.4.0 Provider** 'i seçerek **Next** butonunu tıklayın.



4. Açılan **Connection** sekmesinin görüntüsünü resimdeki gibi düzenleyerek **OK** butonunu tıklayın.

Bağlantının Oluşturulması

Dershane veritabanına bağlantı sağlamak için **OleDbConnection** oluşturun.

3. Araç kutusu üzerindeki **OleDbConnection** kontrolü form üzerine sürükleyin.
4. **OleDbConnection** kontrolünün **ConnectionString** özelliği için oluşturduğunuz bağlantı cümlesini seçin.

DataAdapter nesnesinin Oluşturulması

Bolum tablosunu DataSet içerisine aktarmak için OleDbDataAdapter oluşturun.

1. Araç kutusu üzerindeki **OleDbDataAdapter** kontrolü form üzerine sürükleyin.
2. Karşınıza çıkan **"Data Adapter Configuration Wizard"** penceresi üzerinde **Next** butonunu tıklayarak, bir sonraki adıma geçin.
3. **Choose Your Data Connection** penceresi üzerinde, oluşturduğunuz bağlantı cümlesini seçin ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
4. **Choose a Query Type** penceresinden **Use SQL statements** seçeneğini seçin ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
5. **Generate the SQL statements** penceresindeki metin kutusuna **SELECT Bolum, BolumID FROM Bolum** yazın ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.

6. **Finish** butonunu tıklayarak “Data Adapter Configuration Wizard” sihirbazını sonlandırın.
7. Eklediğiniz OleDbDataAdapter kontrolunun ismini **daBolum** olarak değiştirin.

Ogrenci tablosunu DataSet içerisine aktarmak için OleDbDataAdapter oluşturun.

1. Araç kutusu üzerindeki **OleDbDataAdapter** kontrolü form üzerine sürükleyin.
2. Karşınıza çıkan “Data Adapter Configuration Wizard” penceresi üzerinde **Next** butonunu tıklayarak, bir sonraki adıma geçin.
3. **Choose Your Data Connection** penceresi üzerinde, oluşturduğunuz bağlantı cümlesini seçin ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
4. **Choose a Query Type** penceresinden **Use Sql statements** seçeneğini seçin ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
5. **Generate the Sql statements** penceresindeki metin kutusuna **SELECT OgrenciID, Ad, Soyad, Adres, BolumID, Sehir FROM Ogrenci** yazın ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
6. **Finish** butonunu tıklayarak “Data Adapter Configuration Wizard” sihirbazını sonlandırın.
7. Eklediğiniz OleDbDataAdapter kontrolunun ismini **daOgrenci** olarak değiştirin.

Kurs ve OgrenciDersKayit tablolarını DataSet içerisine aktarmak için OleDbDataAdapter oluşturun.

1. Araç kutusu üzerindeki **OleDbDataAdapter** kontrolü form üzerine sürükleyin.
2. Karşınıza çıkan “Data Adapter Configuration Wizard” penceresi üzerinde **Next** butonunu tıklayarak, bir sonraki adıma geçin.
3. **Choose Your Data Connection** penceresi üzerinde, oluşturduğunuz bağlantı cümlesini seçin ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
4. **Choose a Query Type** penceresinden **Use Sql statements** seçeneğini seçin ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
5. **Generate the Sql statements** penceresindeki metin kutusuna **SELECT OgrenciDersKayit.OgrenciID, Kurs.KursAdi, OgrenciDersKayit.KursBasTarihi, Kurs.Kredi, Kurs.KursNo FROM (Kurs INNER JOIN OgrenciDersKayit ON Kurs.KursNo = OgrenciDersKayit.KursNo)** yazın ve **Next** butonunu tıklayarak, bir sonraki adıma geçin.
6. **Finish** butonunu tıklayarak “Data Adapter Configuration Wizard” sihirbazını sonlandırın.
7. Eklediğiniz OleDbDataAdapter kontrolunun ismini **daKurs** olarak değiştirin.

DataSet nesnesinin Oluşturulması

Bolum, Öğrenci ve Kurs tabloları ile çevrimdışı çalışmak için DataSet oluşturun.

1. **daBolum** üzerinde farenin sağ butonunu tıklayın.
2. Açılan kısayol menüsünden **Generate DataSet** menüsünü tıklayın.
3. **Choose a Dataset** menüsünden **New** seçeneğini seçin ve metin kutusuna ds yazın.
4. **ok** butonunu tıklayın
5. **daOgrenci** üzerinde farenin sağ butonunu tıklayın.
6. Açılan kısayol menüsünden **Generate DataSet** menüsünü tıklayın
7. **Choose a Dataset** menüsünden **Existing** seçeneğini seçin ve açılan kutudan Dershane.ds seçeneğini seçin.

8. **Ok** butonunu tıklayın.
9. **dakurs** üzerinde farenin sağ butonunu tıklayın.
10. Açılan kısayol menüsünden **Generate DataSet** menüsünü tıklayın
11. **Choose a Dataset** menüsünden **Existing** seçeneğini seçin ve açılan kutudan Dershane.ds seçeneğini seçin.
12. **Ok** butonunu tıklayın.

DataView nesnesinin Oluşturulması

DataTable nesneleri üzerinde filtreleme işlemleri yapmak için DataView oluşturun.

Ogrenci tablosunu filtrelemek için aşağıdaki adımlar takip edin.

1. Araç kutusu üzerindeki DataView kontrolü form üzerine sürükleyin.
2. Eklediğiniz DataView kontrolünün ismini dvOgrenci olarak değiştirin.
3. DataView kontrolünün Table özelliği için Ogrenci tablosunu seçin.

Kurs tablosunu filtrelemek için aşağıdaki adımlar takip edin.

1. Araç kutusu üzerindeki DataView kontrolü form üzerine sürükleyin.
2. Eklediğiniz DataView kontrolünün ismini dvKurs olarak değiştirin.
3. DataView kontrolünün Table özelliği için Kurs tablosunu seçin

Dataset içerisindeki verinin ComboBox kontrolüne bağlanması

Bolum tablosunu cbBolum isimli açılan kutuya bağlayın.

4. CbBolum isimli açılan kutunun DataSource özelliğine Bolum tablosunu seçin.
5. CbBolum isimli açılan kutunun DisplayMember özelliğine Bolum kolonunu seçin.
6. CbBolum isimli açılan kutunun ValueMember özelliğine BolumID kolonunu seçin.

Kodların Yazılması

1. Form kontrolünün **Load** olayına kayıtları DataSet tablolarına dolduran kodları yazın.

```
try
{
    Conn.Open();
    dabolum.Fill(Ds1, "Bolum");
    daOgrenci.Fill(Ds1, "Ogrenci");
    dakurs.Fill(Ds1, "Kurs");
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString());
}
finally
{
    Conn.Close();
}
```

2. btnListele kontrolunun **Click** olayına dvOgrenci kontrolunu DataGridView kontrolune bağlayan kodu yazın.

```
dvOgrenci.RowFilter = "BolumID='"
    cbBolum.SelectedValue + "'";
dgOgrenci.DataSource = dvOgrenci;
```

3. dgOgrenci kontrolunun **CurrentCellChanged** olayına dvKurs kontrolunu DataGridView kontrolune bağlayan kodu yazın.

```
dvKurs.RowFilter = "OgrenciID='"
    dgOgrenci.Item(dgOgrenci.CurrentRow, 0) + "' ";
dgKurs.DataSource = dvKurs;
```

ASP.NET GİRİŞ



Modül 6: ASP.NET GİRİŞ

ASP.NET GİRİŞ

- ◆ ASP.NET Nedir?
- ◆ ASP Tarihçesi
- ◆ ASP.NET Uygulama Mimarisi
- ◆ ASP.Net Çalışma Modeli
- ◆ ASP.NET'in .NET Çatısındaki Yeri
- ◆ .NET Framework'un Asp.Net'teki Avantajları
- ◆ ASP.NET ile Uygulama Geliştirmek

ASP.NET, .Web sunucusu üzerinde çalışan ve .NET altyapısını kullanan geliştirme platformudur. ASP.NET Web Form nesneleri, dinamik Web uygulamaları geliştirmeyi kolaylaştırır.

Bu modülü tamamladıktan sonra:

- ASP.NET çalışma modelini öğrenecek,
- ASP.NET teknolojisinin .NET Framework çatısındaki yerini öğrenecek,
- IIS Web sunucusunun yapısını öğrenecek ve yönetebileceksiniz.

Konu 1: ASP.NET Nedir?

ASP.NET Nedir?

ASP



ASP.NET

- .NET çatısı ile yeni özellikler.
- Web Server tarafından çalışan, Web Form'lar.
- Web Server tarafından çalışan, HTML kodları ve ASP kontrolleri.
- Web Server tarafından çalışan, güvenlik bileşenleri.
- İstemci tarafından sadece Script'ler.

ASP.NET teknolojisinden önce, web üzerinde dinamik sayfalarla çalışabilmek için ASP teknolojisi kullanılırdı. ASP teknolojisi, .NET çatısı ile yeni özelliklere eklendi.

ASP.NET ile web uygulaması geliştirmek, Windows Form tabanlı uygulama geliştirmeye oldukça benzemektedir. Web Server tarafından çalışan, HTML kodlarını ve ASP kontrollerini içeren, temel ASP .NET bileşenine Web Form denir. Bir web uygulamasında birden fazla Web Form bulunabilir.

ASP.NET sayfaları ile yazılan kodlar sunucu tarafında çalışır, istemci tarafında çeşitli işlemleri gerçekleştirebilmek içinse Script adı verilen kodlar kullanılır. Web uygulamasının güvenliği ise yine sunucu tarafında çalışan .NET bileşenleri ile sağlanır.

ASP.NET Web Formları sunucu taraflı kodları çalıştırdığı için, kullanıcı tarafından tarayıcıya ve işletim sistemine bağlı değildir. Dolayısıyla ASP.NET ile yazılan uygulamalar, internet erişimi olan herhangi bir aygıtta çalışabilir.

Konu 2: Asp Tarihçesi

Asp Tarihçesi

- HTML (HyperText Markup Language)
- CGI (Common Gateway Interface)
- PERL (Practical Extraction and Reporting Language)
- ISAPI (Internet Server Application Programming Interface)
- ASP (Active Server Pages)

HTML(Hyper Text Markup Language) web sayfası hazırlamak için kullanılan temel web programlama dilidir.

HTML, kullanıcı ile sunucu arasında dinamik veri alışverişi sağlamaz. HTML'in bu açığını kapatmak için ilk olarak CGI arabirimini (Common Gateway Interface) geliştirilmiştir. CGI arabirimini C dilinde hazırlanan kodlar ile çalıştırılır. CGI arabiriminin dezavantajı, en ufak bir değişiklikte tüm kaynak kodun yeniden derlenmesidir. Bu durum zaman ve kaynak kullanımını olumsuz yönde artırır.

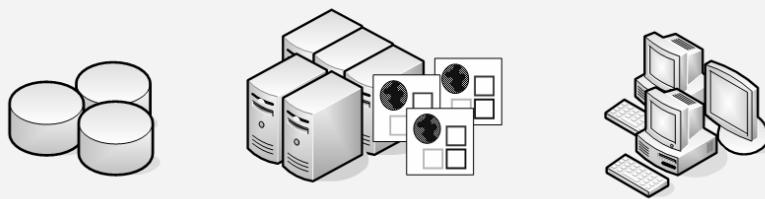
CGI arabiriminden sonra, sununu ile haberleşen ilk dil olan PERL(Practical Extraction and Reporting Language) geliştirilmiştir. Bu dil C ve C++ ile yazılan scriptler içerir. PERL, CGI'ın yeniden derlenme dezavantajını ortadan kaldırmıştır. PERL halen aktif olarak kullanılmaktadır.

Microsoft NT teknolojisini ile birlikte Internet Information Server (IIS) sunucusunu geliştirmiştir. IIS, Windows NT 3.51 ile gelen web sunucusudur. IIS sunucusunun "Windows NT 3.51" ile gelen ilk sürümü CGI arabirimini desteklemektedir. Microsoft IIS sunucusunu geliştirdikten sonra, "Internet Server Application Programming Interface" (ISAPI) arabirimini geliştirmiştir. Microsoft ilk defa ISAPI ile birlikte ASP teknolojisini duyurmuştur. ASP teknolojisi, IIS ve ISAPI alt yapısını birleştirir. ASP, statik HTML sayfaları içerisinde, dinamik veri alışverişi için Microsoft tarafından geliştirilmiştir

Konu 3: ASP.NET Uygulama Mimarisi

ASP.NET Uygulama Mimarisi

- ◆ İstemci Katmanı (Presentation Tier)
- ◆ İş katmanı (Business Logic Tier)
- ◆ Veri Katmanı (Data Tier)



ASP.NET, multi-tier (çok katmanlı) veri erişim modelini kullanır. Bu veri erişim modeli İstemci, İş ve Veri katmanlarından oluşur.

İstemci Katmanı (Presentation Tier):

Bu katman, kullanıcı ile diğer katmanların iletişimini sağlar. Bu katmanda, kullanıcı arayüzü oluşturan bileşenler bulunur. Html ve Sunucu kontroller, Web Formlar ve kullanıcı tanımlı kontroller (User Controls) bu katman içerisinde yer alır.

İş katmanı (Business Logic Tier) :

Bu katman uygulama ile veritabanı arasında iletişimini sağlar. Bu katmanda iş servisleri ve kurallarını içeren bileşenler bulunur. XML Web servisleri, COM ve COM+ nesneleri bu katman içerisinde yer alır

Veri Katmanı (Data Tier) :

Veri katmanıdır. Bu katmanda veriyi saklamak için gerekli araçlar bulunur. İlişkisel veritabanları, e-mail alanları, mesaj kuyrukları ve dizin servisleri bu katman içerisinde yer alır. Web uygulamalarda, ASP.NET ile veri kaynağuna erişim için ADO.NET kullanılır.

Konu 4: Asp.Net Çalışma Modeli

Asp.Net Çalışma Modeli

- ◆ Tür Yönetimi (Type Management)
- ◆ JIT Derleme (JIT Compilation)
- ◆ Hafıza Yönetimi (Memory Management)
- ◆ Exception Yöneticisi (Exception Manager)

ASP.NET, ASP ve diğer web platformlarına göre daha yüksek performans ile çalışır. ASP.NET bu performans artışını Visual Studio .NET ile gelen, .NET Framework ve CLR (Common Language Runtime) ile sağlar.

ASP.NET platformunu en verimli şekilde kullanmayı sağlayan CLR bileşenleri aşağıdaki gibidir.

- Type Management
- Memory Management
- JIT Compilation
- Exception Manager

Tür Yönetimi (Type Management)

Tür Yönetimi

- Güvenli olmayan bilgilere ve başlatılmamış değişkenlere izin vermez.
- ASP.NET'i ASP'den tamamen ayıran bir özelliktir.

Güvenli olmayan bilgilere ve başlatılmamış değişkenlere izin vermez Bu yönetim, ASP.NET'i ASP'den tamamen ayıran bir özelliktir.

JIT Derleme (JIT Compilation)

JIT Derleme

- ASP.NET Web sayfaları
- MSIL (Microsoft Intermediate Language)
- MSIL kodu çalışma zamanında, JIT (Just In Time Compiler) ile “native code” adı verilen dile çevrilir.

ASP.NET Web sayfaları, kullanılan dilin editöründe derlenerek, MSIL (Microsoft Intermediate Language) diline çevrilir. MSIL kodu çalışma zamanında, JIT ile “native code” adı verilen dile çevrilir.

Hafıza Yönetimi (Memory Management)

Hafıza Yönetimi

- CLR ile otomatik hafıza yönetimi
- Nesneler için, hafızada yer ayrılması
 - “New” anahtar sözcüğü
- Nesneler referanslarını kaybettikten sonra
 - “Garbage Collection”

CLR ile hafıza yönetimi otomatik olarak işlenir. **New** anahtar sözcüğü ile oluşturulan nesneler için, CLR hafızada yer ayırır. Nesneler referanslarını kaybettikten sonra “Garbage Collection” mekanizması ile bellekten silinir.

Exception Yöneticisi (Exception Manager)

Exception Yöneticisi

- Yapısal hata yakalama
 - Try...Catch...Finally
- Uygulama konfigürasyonu
 - Web.config
- Sunucu konfigürasyonu
 - Machine.config

CLR, ASP.NET uygulamaları için yapısal hata yakalama altyapısı sunar. ASP.NET uygulamalarında **Try...Catch...Finally** blokları kullanılarak ,hata yakalama altyapısı kolayca devreye sokulur.

ASP.NET uygulamaların konfigürasyon ayarları, XML dosyalar içerisinde saklanır. Bu dosyalar kolayca okunur ve yazılabilir. Her web uygulamanın kendisine ait bir konfigürasyon dosyası vardır. ASP.NET uygulamaların konfigürasyon dosyası **web.config** dir.

Sunucuya ait konfigürasyon ayarları ise machine.config içerisinde saklanır. Her web sunucusunda tek **machine.config** dosyası bulunur.

Visual Studio .NET, web uygulamalarının performansını artırmak ve güvenliğini sağlamak için pek çok servis sunar.

ASP.NET Uygulama Bileşenleri

- Web Formlar
 - Web uygulama için kullanıcı arayüzü sağlar.
- Code-behind sayfalar
 - Web Formların sunucu tarafında çalışan kodlarını içerir.
- Konfigürasyon dosyaları
 - Web uygulama ve sunucu ayarlarının tutulduğu XML dosyalarıdır.
- Global.asax dosyaları
 - Web uygulamasının genel olaylarını içerir.

Bir ASP.NET uygulamasını oluşturan bileşenler aşağıdaki gibidir.

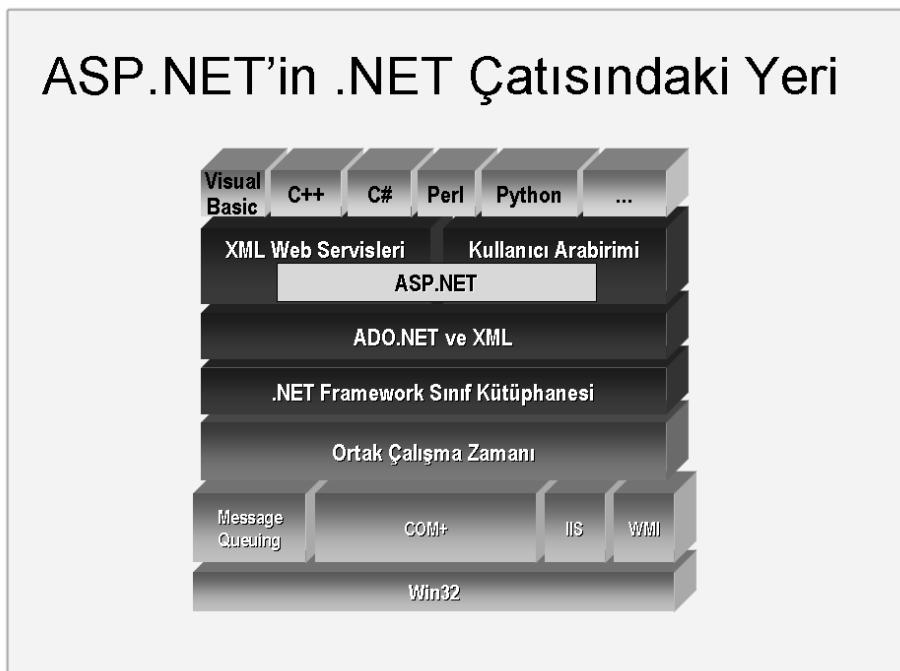
- Web Formlar: Web uygulama için kullanıcı arayüzü sağlar.
- Code-behind sayfalar: Web Formların sunucu tarafında çalışan kodlarını içerir.
- Konfigürasyon dosyaları: Web uygulama ve sunucu ayarlarının tutulduğu XML dosyalarıdır.
- Global.asax dosyaları: Web uygulamanın genel olaylarını içerir. Örneğin Web uygulamanın başlatılması veya durdurulması. Global.asax dosyası ASP deki global.asa dosyasının gelişmiş versiyonudur.

ASP.NET Uygulama Bileşenleri

- XML Web Servis bağlantıları
 - Web uygulamasının, XML web servisi üzerinden veri alışveriшини sağlar.
- Veritabanı bağlantıları
 - Web uygulaması ile veri kaynağı arasında veri alışveriшинi sağlar.
- Caching (Ön Belleğe Alma)
 - Uygulamanın ilk çalıştığı anda ön belleğe atılmasını sağlar.

- XML Web Servis bağlantıları: Web uygulamanın, XML web servisi üzerinden veri alışveriшинi sağlar.
- Veritabanı bağlantıları, Web uygulama ile veri kaynağı arasında veri alışveriшинi sağlar.
- Caching (Ön Belleğe Alma): Uygulamanın ilk çalıştığı anda ön belleğe atılmasını sağlar. Bu durum uygulamanın bellekten çalışmasını sağlayarak, performansı artırır.

Konu 5: ASP.NET'in .NET Çatısındaki Yeri



Microsoft .NET platformu, geniş çaplı web uygulamaları geliştirebilmek için, gerekli her türlü araç ve teknolojiye sahiptir. Dilden bağımsız çalışabilme, eski teknolojiden yeni teknolojilere kolayca geçiş imkanı sağlar. Tamamen nesne yönelimli programlamayı destekleyen bir platform olan Visual Studio .NET, web uygulamalarında da nesne yönelimli programlama modelini destekler.



Şekil 5.1'de belirtildiği gibi en üst katman, kullanıcı ve program arayüzlerini gösterir. Bu arayüzler Windows Form, Web Form, Web Service ve uygulama servislerinden oluşabilir. Orta katmanda .NET Framework sınıfları, alt katmanda ise CLR bulunur.

Konu 6: .Net Framework'un Asp.Net'teki Avantajları

.Net Framework'un Asp.Net' teki Avantajları

- Çoklu dil desteği.
- Gelişmiş güvenlik sınıfları.
 - System.Web.Security
- HTML ve Kaynak kod birlikte çalıştırılır.
- Kodlar satır satır derlenmez. Web formlar derlenir. Performans artışı sağlanır.
- Güçlü hata yakalama araçları.
- Web servisleri desteği.
- ADO.NET kullanımını kolaylaştıran web nesneleri içerir.

.Net Framework, ASP.NET ile uygulama geliştirmek için birçok avantaj sağlar. Bu avantajlar aşağıda listelenmiştir.

- Visual Studio .NET ortamının en büyük avantajı, birden fazla dili destekliyor olmasıdır. ASP.NET ile geliştirilen uygulamalarda, farklı .NET dilleri bir arada kullanılabilir. Örneğin VB.NET ile geliştirilen bir uygulama içerisinde C# ile yazılan kod blokları eklenebilir.
- Visual Studio .NET, web uygulamaların güvenliğini sağlayan çeşitli sınıflar içerir. Bu sınıflar System.Web.Security isim alanı içerisinde bulunur.
- ASP .NET sayfaları içerisinde, HTML ve Kaynak kod birlikte çalıştırılır. Bu durum tasarım ve programlama kolaylığı sağlar.
- ASP.NET içerisinde kodlar satır satır derlenmez. Bunun yerine Web formlar derlenir. Bu durum performansın artmasını sağlar.
- Güçlü hata yakalama araçları sunar.
- Web servisleri ile birlikte çalışabilir.
- ASP.NET, ADO.NET kullanımını kolaylaştıran web nesneleri içerir.

Konu 7: ASP.NET ile Uygulama Geliştirmek

ASP.NET ile Uygulama Geliştirmek

- ◆ IIS Nedir?
- ◆ IIS Kurulumu ve Yönetimi
- ◆ .Net Framework Kurulumu

ASP.NET ile geliştirilen uygulamaların; Internet, Extranet veya Intranet üzerinde çalışabilmesi için Web Sunucularına ihtiyaç duyulur. IIS (Internet Information Services) Windows işletim sistemleri için geliştirilmiş web sunucusudur.

IIS Nedir?

ASP.NET ile Uygulama Geliştirmek

- ASP.NET ile geliştirilen uygulamaların; Internet, Extranet veya Intranet üzerinde çalışabilmesi için Web Sunucularına ihtiyaç duyulur.

▪ IIS Nedir?

- IIS(Internet Information Services), Windows sistemler için web tabanlı uygulama geliştirme ve yayınılama amacıyla kullanılan web sunucusudur.

IIS(Internet Information Services), Windows sistemler için web tabanlı uygulama geliştirme ve yayınılama amacıyla kullanılan web sunucusudur.

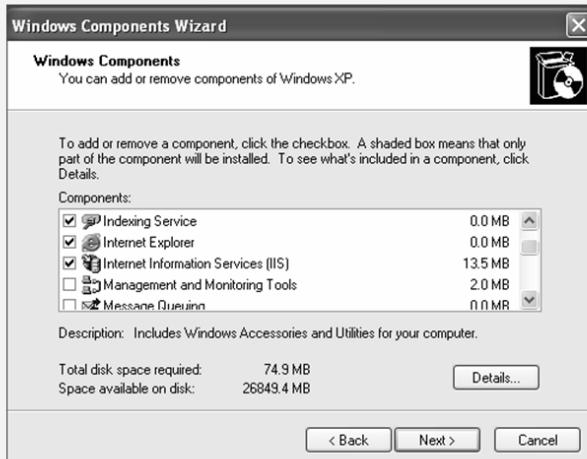
IIS Kurulumu ve Yönetimi

IIS Kurulumu

- ✓ Denetim Masası (Control Panel) penceresinde “Program Ekle/Kaldır” (Add or Remove Programs) simgesini seçin.
- ✓ Açılan pencerenin sol panelinden “Windows Bileşeni Ekle/Kaldır” (Add/Remove Windows Components) bileşenini seçin.
- ✓ “Windows Bileşeni Ekle/Kaldır” penceresinden “Internet Information Services” (IIS) seçerek yükleme işlemini başlatın.

IIS Kurulumu

IIS Kurulumu



Web uygulamaları geliştirmek için IIS 5.0 veya daha üst versiyonu kurulmalıdır. IIS, Windows 2000 Server işletim sistemi ile varsayılan bileşen olarak gelir.

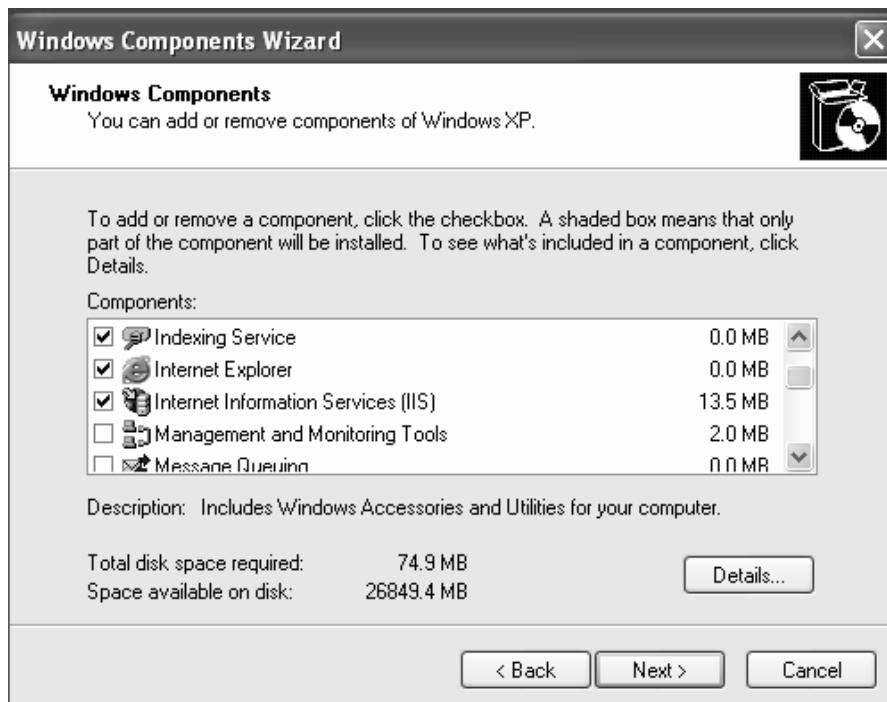
Windows 2000 Professional, Windows XP Professional ve sonraki sistemlerde ise, bu aracın kullanıcı tarafından kurulması gereklidir.

IIS kurulumu için aşağıdaki adımlar takip edilir.

- 1- Denetim Masası (Control Panel) penceresinde “Program Ekle/Kaldır” (Add or Remove Programs) simgesini seçin.
- 2- Açılan pencerenin sol panelinden “Windows Bileşeni Ekle/Kaldır” (Add/Remove Windows Components) bileşenini seçin.
- 3- “Windows Bileşeni Ekle/Kaldır” penceresinden “Internet Information Services” (IIS) seçenek yükleme işlemini başlatın.

DİKKAT: Windows NT 4.0 ve Windows XP Home Edition işletim sistemleri ile ASP.NET uygulaması geliştirilemez.

.NET Framework kurulmadan önce IIS sunucusunun kurulmuş olmasına dikkat edilmelidir. Aksi halde ASP.NET dosyaları, ilgili kütüphane dosyaları ile düzgün bir şekilde kullanılamaz. Eğer IIS kurulmadan .NET Framework kurulmaya çalışılırsa, uyarı mesajı ile karşılaşılır. Bu uyarı mesajı önemsenmeden kurulumu devam edilebilir. Framework kurulumu tamamlandıktan sonra IIS kurulmalıdır. Ancak IIS yüklenikten sonra, sistemin ASP.NET sayfaları ile uyum içinde çalışabilmesi için Visual Studio .NET komut satırında “aspnet_regiis.exe -I” komutu çalıştırılmalıdır.



Şekil 6.2: IIS Kurulumu

IIS Yönetimi

- ### IIS Yönetimi
- ✓ “Bilgisayarım”(My Computer) ikonuna sağ tıklanır.
 - ✓ Kısayol menüsünden Yönet (Manage) komutu seçilir.
 - ✓ Computer Manager penceresinin Services and Applications menüsünden Internet Information Services (IIS) seçilir.
 - ✓ Denetim Masası (Control Panel) içerisinde Administrative Tools simgesi seçilir.
 - ✓ Internet Information Services Manager seçilir.

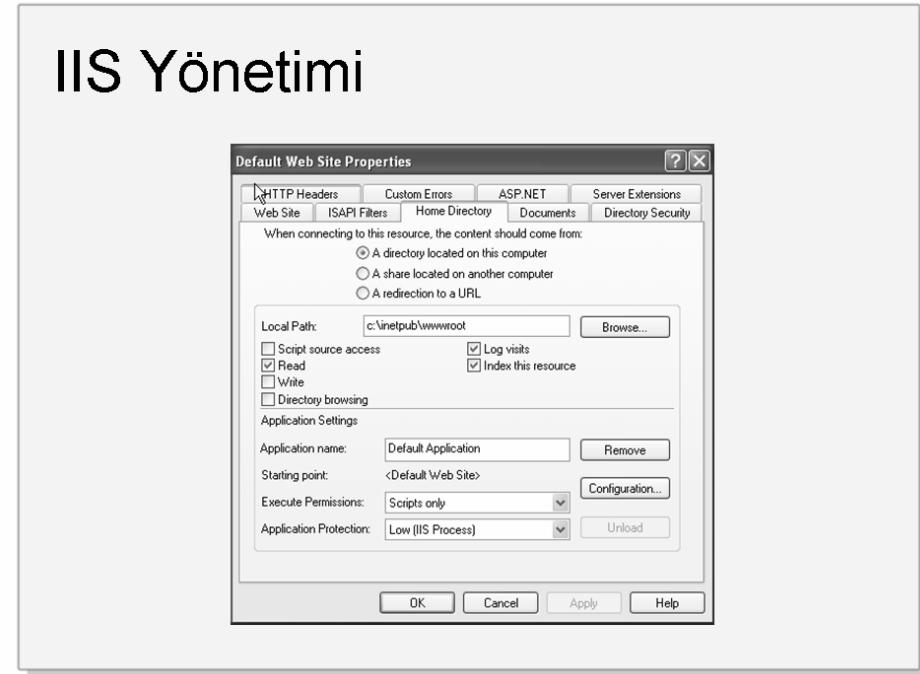
IIS yönetimi, **Internet Information Services (IIS) Manager** ile gerçekleştirilir.

IIS Manager açmak için aşağıdaki adımlar takip edilir.

- “Bilgisayarım”(My Computer) ikonuna sağ tıklanır. Açılan kısayol menüsünden Yönet (Manage) komutu seçilir. Açılan Computer Manager penceresinin **Services and Applications** menüsünden **Internet Information Services (IIS)** seçilir.
- Denetim Masası (Control Panel) içerisinde **Administrative Tools** simgesi seçilir. Açılan pencereden **Internet Information Services Manager** seçilir.

IIS içerisinde aşağıdaki altklassörler bulunur.

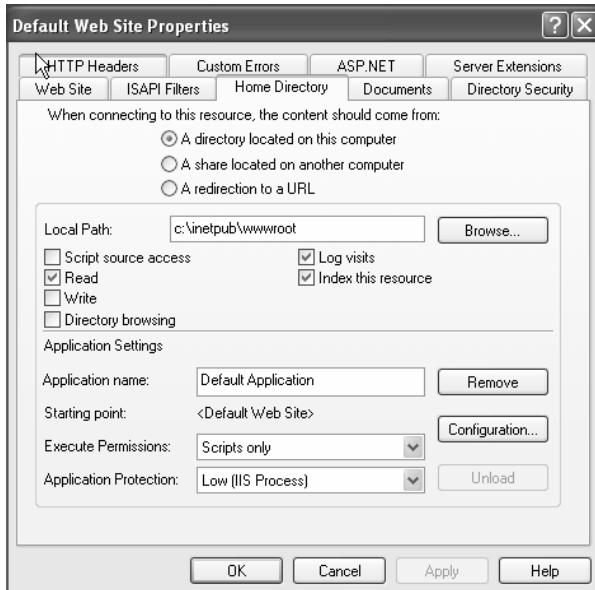
- Application Pools
- Web Sites
- Web Service Extensions



web Sites çalışan web uygulamalarını liseleler. **web Sites** klasörü altındaki **Default web site** sekmesi üzerinden web sunucu seçenekleri ayarlanabilir. Web sunucu özelliklerini değiştirmek için aşağıdaki adımlar takip edilir.

1. **Internet Information Services** üzerinden **web Sites** seçilir.
2. **web sites** üzerinde farenin sağ tuşu tıklanır. Açılan menüden **Properties** menüsü seçilir.

Home Directory kategorisinde **Local Path** alanında “c:\inetpub\wwwroot” ifadesi, sistemde IIS sunucusunun çalıştıracağı uygulamaların yer bilgisini tutar.



Resim 6.3: IIS Yönetimi

ASP.NET web uygulamaları wwwroot klasörü altında tutulur. Bu klasör altında tutulan klasörlerin diğerlerinden farkı **virtual directory** (sanal klasör) olmalarıdır. .NET ile açılan her yeni web uygulaması için, wwwroot altında yeni bir **virtual directory** oluşturulur.

Visual Studio .NET kullanmadan yeni bir **virtual directory** oluşturmak için **Default Web Site** Üzerinde sağ tıklanır. Çıkan menüden **New** alt menüsüne işaret edilir ve **Virtual Directory** seçilir. **Virtual Directory Creation Wizard** ile yeni bir **virtual directory** oluşturulur.

.Net Framework Kurulumu

- ### .Net Framework Kurulumu
- ✓ Framework kurulum dosyası çalıştırılır.
 - ✓ Açılan penceredeki "Would you like to Install Microsoft .NET Framework Package?" sorusuna Yes cevabı verilir.
 - ✓ Next butonları tıklanarak kurulum tamamlanır.

ASP.NET ile uygulama geliştirmek için .NET Framework'ün kurulu olması gereklidir. Framework'ün,. SDK olarak isimlendirilen 130MB'lık full versiyonu ve yalnızca temel bileşenleri kapsayan 20MB'lık iki farklı kurulum dosyası bulunmaktadır.

Framework versiyon ve yamaları (Service Pack) <http://msdn.microsoft.com/netframework/downloads/updates/default.aspx>" adresinden ücretsiz olarak indirilebilir.

Framework'ü kurmak için aşağıdaki adımlar takip edilir.

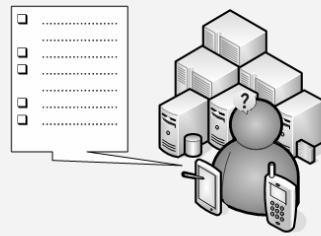
1. Framework kurulum dosyası çalıştırılır.
2. Açılan penceredeki "Would you like to Install Microsoft .NET Framework Package?" sorusuna Yes cevabı verilir.
3. Next butonları tıklanarak kurulum tamamlanır.

İPUCU: .Net Framework kurabilmek için işletim sisteminin Windows NT tabanlı olması gereklidir. Windows 2000 işletim sisteminde minimum SP2 yapılandırması gereklidir.

Modül Özeti

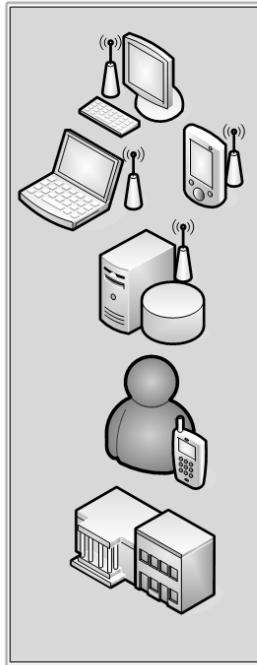
Modül Özeti

- ASP nedir?
- ASP çalışma modelini açıklayın.
- ASP .Net uygulamasını oluşturan bileşenler nelerdir?
- IIS nedir?



11. ASP nedir?
12. ASP çalışma modelini açıklayın.
13. ASP .Net uygulamasını oluşturan bileşenler nelerdir?
14. IIS nedir?

Lab 1: Web Tabanlı Uygulamaların Yayınlanması



Uygulamalar

Web Tabanlı Uygulamaların Yayınlanması

- ◆ IIS Kurulumu
- ◆ Uygulama yayınılmak

Bu uygulamada IIS(Internet Information Services) kurulumu öğreneceksiniz. Aynu zamanda IIS ile web tabanlı uygulamaların yayınımasını öğreneceksiniz.

IIS(Internet Information Services) Kurulması

1. Denetim Masası (Control Panel) penceresinde “Program Ekle/Kaldır” (Add or Remove Programs) simgesini seçin.
2. Açılan pencerenin sol panelinden “Windows Bileşeni Ekle/Kaldır” (Add/Remove Windows Components) bileşenini seçin.
3. “Windows Bileşeni Ekle/Kaldır” penceresinden “Internet Information Services” (IIS) seçin.
4. Next butonunu tıklayarak kurulumu başlatın.

Uygulama Yayınılamak

Default.htm ismindeki HTML sayfayı IIS üzerinden yayinallyn.

3. “C:\Inetpub\wwwroot” klasörüne gidin.
4. wwwroot penceresi içerisinde **Dosya** menüsünü tıklayın.
5. **Dosya** menüsü içerisinde **Yeni** alt menüsünü tıklayın.

6. Yeni alt menüsü içerisinde **Metin Belgesi** komutunu vererek Yeni **Metin Belgesi** oluşturun.
7. Oluşturduğunuz metin belgesi içerisinde aşağıdaki HTML(Hyper Text Markup Language) kodları ekleyin ve dosyayı kaydedin.

```
<html>
  <head>
    <title>HTMLPage1</title>
  </head>
  <body>
    <p>Hoş Geldiniz.</p>

  </body>
</html>
```

8. Metin belgesinin ismini Default.htm olarak değiştirin.
9. Internet Explorer açın ve aşağıdaki adreslerden herhangi birini adres çubuğuuna yazın.
 - a. <http://localhost>
 - b. <http://127.0.0.1>
 - c. <http://MakinaAdı>
 - d. <http://IpNumarası>

localhost: Lokal makina adı.

127.0.0.1 : Lokal IP numarası.

Makina Adı: Ağ içerisindeki bilgisayar adı.

Ip Numarası: Ağ içerisindeki Ip Numarası.

İpucu : Web uygulamanın yayını, wwwroot içerisindeki herhangi bir alt klasörden yapılabilir. Örneğin <http://localhost/WebUygulama>. “ASP.NET Web Application” uygulamaların yayını bu yöntem ile yapılır.

ASP.NET Web Form ve Kontrolleri ile Çalışmak



Modül 7: ASP.NET Web Form ve Kontrolleri ile Çalışmak

ASP.NET Web Form ve Kontrolleri ile Çalışmak

- ◆ Web Form Bileşenleri
- ◆ Server (Sunucu) Kontroller
- ◆ Kontrollerin Sınıflandırılması
- ◆ Standart Kontroller
- ◆ Doğrulama (Validation) Kontrolleri
- ◆ Zengin Kontroller
- ◆ AutoPostBack Özelliği
- ◆ ViewState

ASP.NET ile uygulama geliştirirken kullanılan temel bileşenler Web Formlar ve Web kontrolleridir. Web Form, IIS tarafından çalıştırılan HTML kod ve

kontrollerin birleşiminden oluşur. Bu formlara eklenen kontroller, sunucu veya istemci taraflı çalışabilirler.

Bu modül tamamlandıktan sonra:

- Web Form yapısını ve bileşenlerini öğrenecek,
- Sunucu ve istemci taraflı kontrollerin farklarını öğrenecek,
- Web kontrollerini tanıyacak,
- **ViewState** ve **PostBack** kavramlarını öğreneceksiniz.

Konu 1: Web Form Bileşenleri

Web Form Bileşenleri

- ◆ Page Özelliği
 - ◆ Language
 - ◆ CodeBehind
 - ◆ SmartNavigation
 - ◆ ViewState
- ◆ Body Özelliği
 - ◆ PageLayout
- ◆ Form Özelliği
 - ◆ Method
 - ◆ Id
 - ◆ Runat

Web Form, ASP.NET uygulamalarının yapı taşıdır. Visual Studio .NET ortamı aracılığı ile eklenen kontrollerin ve Visual Basic .NET kodlarının birleşimi Web Formu oluşturur.

Web formlar, .aspx uzantılı arayüz dosyası ve. aspx.cs uzantılı kod dosyalarından oluşur. Örneğin default.aspx isimli ASP.NET sayfasının, sunucu taraflı Visual Basic .NET kodları default.aspx.cs isimli dosyada tutulur.

Kullanıcı arayüz sayfası ve kod sayfasının ayrı tutulmasının yararı, web programcısına ve web tasarımcısına ayrı kaynaklar sunarak bağımsız çalışma ortamı sağlamaktır.

Web Formları Visual Studio ile iki farklı şekilde tasarılanabilir. Design sekmesi, web kontrollerinin görsel olarak düzenlenmesini sağlar. HTML sekmesi ise, kontrollerin HTML kodları ile eklenmesini sağlar.

Görsel kısmında Web Form kontrolleri ve bu kontrollere ait HTML kodları, kod sayfasında da bu kontrollerin davranışlarını belirleyen Visual Basic .NET kodları bulunur.

Web Form Bileşenleri

- **Web Formu**
 - Visual Studio .NET ortamı aracılığı ile eklenen kontrollerin ve C# .NET kodlarının birleşimi
- **Default.aspx**
 - aspx uzantılı arayüz dosyası
- **Default.aspx.cs**
 - aspx.cs uzantılı kod dosyası

Web Formların genel özellikleri aşağıdaki gibidir.

- @Page Özelliği
- Body Özelliği
- Form Özelliği

Page Özelliği

- # Page Özelliği
- Language
 - Sayfa içinde kullanılacak dil seçeneği.
 - CodeBehind
 - Web formların, kod dosyaları.
 - SmartNavigation
 - Kaydırma çubuklarının sayfa içerisindeki yerinin korunması.
 - ViewState
 - Nesne içerisinde girilen bilginin, sunucu tarafında değişkende tutulması.

Tüm sayfa içinde tanımlanacak fonksiyonların değerlerini içerir. <@Page> etiketi ile gösterilir ve her .aspx uzantılı dosyada bulunması gereklidir.

<@Page> etiketinde, sayfanın yapısı ile ilgili özellikler bulunur.

Language

Sayfa içinde kullanılacak dil seçeneğini bildirilir. ASP.NET uygulamalarında genellikle VB ve C# dilleri tercih edilir.

```
<"@Page Language="c#" ...>  
<"@Page Language="vb" ...>
```

CodeBehind

Web formlarının, Visual Basic .NET veya C# uzantılı kod dosyasını belirtir.

```
<@Page CodeBehind="WebForm1.aspx.vb" ...>  
<@Page CodeBehind="WebForm1.aspx.cs" ...>
```

SmartNavigation

SmartNavigation özelliğine True değeri ayarlanırsa, sayfa yeniden yüklentiği zaman, kaydırma çubuklarının sayfa içerisindeki yeri korunur. Böylece sayfa ilk konumunda kalır. Bu özellik Internet Explorer 5.5 ve üstü tarayıcılar tarafından desteklenir.

```
<%@Page Language="c#" CodeBehind="WebForm1.aspx.cs" _  
SmartNavigation="True" %>
```

ViewState

ASP.NET teknolojisi ile gelen yeniliklerden biridir. **EnableViewState** özelliği ile objenin içerisinde girilen bilgi ne olursa olsun, sunucu bunu bir değişkende tutup tekrar kullanıcıya geri döndürür. Bu durum sunucuya gönderilen veriler üzerinde hata oluşması durumunda, bilgilerin kaybolmamasını sağlar. Bu özelliğin tüm kontrolleri içermesi için, **Page** yönerge satırında tanımlanması gereklidir. Bu özellik True veya False değeri alabilir.

```
<%@Page EnableViewState="True" ...%>
```

Ayrıca kontrol düzeyinde **EnableViewState** özelliği kullanılabilir. Bu durumda, **Page** yönerge satırında belirtilen değer geçersiz olur.

```
<asp:Button ... EnableViewState="false" ...>
```

Body Özelliği

Body Özelliği

- **PageLayout**
 - Web form içinde kullanılan nesnelerin, görüntülenme biçimleri
 - **FlowLayout**
 - Kontroller eklenme sırasına göre sıralanır, yerleri sürükleme ile değiştirilemez.
 - **GridLayout**
 - Kontroller form üzerindeki herhangi bir yere eklenebilir, yerleri sürükleme ile değiştirilebilir.

Web sayfasının ana bölümündür.

<body> etiketi ile web formun gövdesi oluşturulur. Kullanılan her kontrol **<body> ... </body>** etiketleri arasında bulunmalıdır.

body etiketi içerisinde **PageLayout(ms_positioning)** özelliği tanımlanabilir.

PageLayout(ms_positioning)

Web form içinde kullanılan nesnelerin, görüntülenme biçimini ayarlar. Bu özellik iki değer alabilir:

- **FlowLayout:** Sayfaya eklenen kontroller eklenme sırasına göre sıralanır. Kontrollerin yerleri sürükleme ile değiştirilemez. Nesneler için **style** tanımlamaz.

```
<body ms_positioning="FlowLayout">
...
</body>
```

- **GridLayout:** Kontroller form üzerindeki herhangi bir yere eklenebilir. Kontrollerin yerleri sürükleme ile değiştirilebilir. Bu görünümde nesneler için **style** tanımlanır. Bu görünüm Windows Uygulamalardaki Form görünümüne benzemektedir.

```
<body ms_positioning="GridLayout">
...
</body>
```

Form Özelliği

Form Özelliği

■ Method

- Web kontrol özelliklerinin, sunucuya gönderilme şeklini belirler.
- Post : İsim ve değer bilgilerini, HTML bilgisinin üst bilgisine yazarak gönderir.
- Get : İsim ve değer bilgilerini, sayfa adının sonuna ekleyerek gönderir.
- Id : Formun isim bilgisi
- Runat : Kontrollerin sunucu ile haberleşerek çalışabilmesi

Web kontrolleri gruplandırmak için kullanılır. Her web form için tek Form etiketi tanımlanır. Tüm kontroller **<form> ... </form>** etiketleri arasında eklenir.

Form etiketi içinde tanımlanabilecek birçok özellik vardır.

Method

Web kontrol özelliklerinin, sunucuya gönderilme şeklini belirler. İki değer alabilir:

- **Post:** İsim ve değer bilgilerini, HTML bilgisinin üst bilgisine yazarak gönderir.

```
<form method="Post" ...>
```

- **Get** İsim ve değer bilgilerini, sayfa adının sonuna ekleyerek gönderir.

```
<form method="Get" ...>
```

Id

Formun isim bilgisini verir. **CodeBehind** sayfası içerisinde, forma işlem yaptırmak için kullanılır.

```
<form id="deneme" ...>
```

Runat

Web formlarda kullanılan kontrollerin sunucu ile haberleşerek çalışabilmesi için runat="server" bildirimi kullanılır. Bu özellik sadece **server** değerini alabilir.

```
<form runat="server" ...>
```

Konu 2: Server(Sunucu) Kontroller

Server (Sunucu) Kontroller

- ◆ **System.Web.UI.HtmlControls**
 - ◆ HTML Server Kontrolleri
- ◆ **System.Web.UI.Control**
 - ◆ Web Server Kontrolleri (ASP.NET Kontrolleri)

Web sunucu üzerinde çalışan kontrollerdir. İki tür server .kontrolu vardır. Bunlar:

- HTML Server Kontrolleri
- Web Server Kontrolleri (ASP.NET Kontrolleri)

ASP.NET server kontrolleri **System.web.UI.Control** sınıfından türetilir. Her ASP.NET server kontrolü <asp:Kontrollsmi> etiketi ile bildirilir. HTML kontrolleri ise **System.web.UI.HtmlControls** isim alanında bulunur.

Button, **TextBox**, **DropDownList** gibi server kontrollerinin çalışma modeli, istemci tarafı HTML kontrollerinin çalışma modelinden oldukça farklıdır. ASP.NET server kontrolleri, tamamen sunucu üzerinde çalışır ve geri plandaki tüm işleyişleri ara yüze gizlenerek gerçekleştirilir.

Bir kontrolün sunucu tarafında çalıştığı **runat="server"** özelliği ile belirlenir.

```
<asp:Button id="Buton1" runat="server" Text="Tıklayınız" />
```

Örnekte istemci tarafında çalışan HTML **Button** kontrolü gösterilmektedir.

```
<INPUT type="button" value="Bu Bir Html Button" >
```

Bu kontrolün sunucu tarafında çalışması için, kontrole **runat** özelliği eklenmelidir. Böylece kontrol HTML server kontrolü haline getirilir.

```
<INPUT type="button" id="button1" runat="server" _  
value="Bu Bir Html Button" >
```

Konu 3: Kontrollerin Sınıflandırılması

Kontrollerin Sınıflandırılması

- Standart Kontroller
 - Button, CheckBox, image, ImageButton, LinkButton, ListBox, TextBox, Table...
- Doğrulama Kontrolleri
 - RequiredFieldValidator, RangeValidator, CompareValidator, RegularExpressionValidator, CustomValidator, ValidationSummary...
- Zengin Kontroller
 - AdRotator ve Calendar
- İlişkisel Liste Tabanlı Kontroller
 - DataList, DataGrid, Repeater...

ASP.NET Web kontrolleri dört grupta listelenir. Bunlar;

- 1- Standart Kontroller (**ListBox**, **Button**, **CheckBox**, **Table** vs.)
- 2- Doğrulama Kontrolleri (**RequiredFieldValidator**, **RangeValidator**, **CompareValidator**, **RegularExpressionValidator**, **CustomValidator**, **ValidationSummary**)
- 3- Zengin Kontroller (**Calendar**, **Adrotator**)
- 4- İlişkisel Liste Tabanlı Kontroller (**DataGrid**, **DataList**, **Repeater**)

Standart Kontroller

Bu kontroller, HTML kontrollere alternatif olarak tasarlanmıştır. Eski tip HTML kontrolleri ile yeni ASP.NET kontrolleri arasındaki en belirgin fark, her Web kontrolünden önce **asp:** ön ekinin kullanıldığı olmasıdır.

```
<asp:TextBox runat="server" id="giris" Text="Hoş Geldiniz">  
</asp:TextBox>
```

Bu kontrollerin avantajları aşağıdaki gibidir.

- Benzer kontrollere düzenli biçimde isimler verilir.
- Tüm kontroller aynı genel özelliklere sahiptir.
- Tarayıcı için özel kodlar kendiliğinden üretilir.

Bu grupta bulunan kontrollerin tümü **id**, **text**, **backcolor**, **runat** özelliklerine sahiptir. Ancak **CheckBox** kontrolünün **Checked** ve **ListBox** kontrolünün **SelectedItem** özellikleri vardır.

Tablo 8.1'de Html ve Standart sunucu kontroller gösterilmektedir.

Web kontrol	Html Kontrol
<asp:Button>	<input type=submit>
<asp:CheckBox>	<input type=checkbox>
<asp:HyperLink>	
<asp:image>	
<asp:ImageButton>	<input type=image>
<asp:LinkButton>	Yok
<asp:Label>	
<asp:ListBox>	<select size="5"> </select>
<asp:Panel>	<div> </div>
<asp:TextBox>	<input type=text>
<asp:RadioButton>	<input type=radio button>
<asp:DropDownList>	<select> </select>
<asp:Table>	<table> </table>

Tablo 7.1 Standart Kontroller

Doğrulama Kontrolleri

Kullanıcının girdiği değerleri kontrol etmek için kullanılır. Kontrolün yapılacak olduğu alana ve veriye göre, farklı doğrulama kontrolleri kullanılır. ASP.NET, belirli bir aralıkta veri girişi sağlayan, karşılaştırma yapan ve belirli değerlerin boş geçilmemesini sağlayan çeşitli doğrulama kontrolleri sunar. **RequiredFieldValidator**, **RangeValidator**, **CompareValidator**, **RegularExpressionValidator**, **CustomValidator**, **ValidationSummary** kontrolleri, bu grupta yer alır.

Zengin Kontroller

AdRotator ve **Calendar** zengin kontroller grubunda yer alır. **AdRotator**, Web sayfaları üzerinde reklam yayını yapmak için kullanılır. **Calendar** ise Web sayfaları üzerinde Takvim göstermek için kullanılır.

İlişkisel Liste Tabanlı Kontroller

Bu kontroller, veritabanından çekilen kayıtların gösterilmesini sağlar. **DataList**, **DataGridView** ve **Repeater** kontrolleri, bu grupta yer alırlar.

Konu 4: Standart Kontroller

Standart Kontroller

- **Label**
 - Kullanıcıya bilgi vermek için kullanılır.
- **TextBox**
 - Kullanıcının bilgi girişini sağlar.
- **Button**
 - Form üzerindeki olayları sunucuya yollamak için kullanılır.
- **CheckBox**
 - Kullanıcıya seçenekler arasından çoklu seçim yapma imkânı sunar.

Label

Label, kullanıcıya bilgi vermek için kullanılır.

```
<asp:Label runat="server" Text="Label Control" Font-  
Italic="true" />
```

Bu kontrol, Internet Explorer tarayıcısında şu şekilde gösterilir.

```
<span style="font-style:italic;">Label Control</span>
```

TextBox

TextBox, kullanıcının bilgi girişini sağlar. En sık kullanılan giriş .kontroludür.

```
<asp:TextBox id="userName" type="text" runat="server">
```

Bu kontrol, Internet Explorer tarayıcısında şu şekilde gösterilir.

```
<input name="userName" id="userName" type="text" />
```

Web kontroller, **WebControl1** sınıfından türemişlerdir. Bu yüzden Web kontroller **BackColor**, **BorderColor**, **Enabled**, **Font**, **Height**, **Width** özelliklerine sahiptir.

Button

Button, form üzerindeki olayları sunucuya yollamak için kullanılır. En sık kullanılan onay kontroludur.

Örnekte Dugme1 isimli button tıklandığında, Label kontrolune mesaj yazılır.

```
<asp:Button id="Dugme1" runat="server" Text="Tıklayınız"
    OnClick="Dugme1_Click" runat="server"/>
<span id="Message" runat="server" />
private void Dugme1(object sender, System.EventArgs e)
{
    Message.InnerHtml="Beni Tıkladın." ;
}
```

CheckBox

CheckBox, kullanıcıya seçenekler arasından seçim yapma imkânı sunar. Onay kutusu işaretlenmiş ise **True**, işaretlenmemiş ise **False** değerini alır. Onay kutusunun durumu **CheckedChanged** metodu ile takip edilebilir.

Örnekte **CheckBox** kontrolünün onay kutusu tıklandığı anda “Seçili”, seçim işlemi geri alındığı anda “Seçili değil” mesajı yazılır. Seçim yapıldığı anda mesajın yazdırılmasını sağlayan **AutoPostBack** özelliğinin, **True** değeridir.

```
void Check_Clicked(Object Sender, EventArgs e)
{
    If (checkbox1.Checked )
    {
        Message.InnerHtml="Seçili" ;
    }
    else
    {
        Message.InnerHtml="Seçili Değil" ;
    }
}
<asp:CheckBox id="checkbox1" runat="server"
    AutoPostBack="True"
    Text="Üye Olmak İster Misiniz?"
    TextAlign="Right"
    OnCheckedChanged="Check_Clicked"/>
<br>
<span id="Message" runat="server" />
```

Standart Kontroller

- RadioButton
 - Kullanıcıya seçenekler arasından tek bir seçim yapma imkânı sunar.
- Hyperlink
 - Sayfalar arası dolaşımı sağlar.
- Image
 - Sayfa içinde resim görüntülemek için kullanılır
- ImageButton
 - Resimli button kontroludur.
- LinkButton
 - HyperLink görünümlü Button kontrolüdür.

RadioButton

RadioButton, **CheckBox** kontrolüne benzerlik gösterir. Ancak **RadioButton** kontrolünün **GroupName** özelliği ile, birden fazla **RadioButton** arasında grup oluşturulur. Aynı grup içerisinde sadece bir **RadioButton** seçilebilir. Birden fazla seçeneğin işaretlenmesine izin verilmez. Onay kutusunun durumu **Checked** metodu ile takip edilebilir.

Örnekte, RadioButton kontrolleri arasında müzik isminde bir grup oluşturulmuştur. Bu grup içerisindeki Pop, Jazz ve Classic RadioButton kontrollerinden sadece bir tanesi seçilebilir. BtnOnay isimli button tıklandığında, seçilen RadioButton kontrolünün değeri Message isimli Label kontrolune yazılır.

```
void BtnOnay_Clicked(Object Sender, EventArgs e)
{
    if (Radio1.Checked)
    {
        Message.InnerHtml = "Seçiminiz" + Radio1.Text;
    }
    else if (Radio2.Checked)
    {
        Message.InnerHtml = " Seçiminiz " + Radio2.Text;
    }
    else if (Radio3.Checked)
    {
        Message.InnerHtml = " Seçiminiz " + Radio3.Text;
    }
}
```

<h4>Beğendiğiniz müzik türünü seçiniz:</h4>

```

<asp:RadioButton id=Radio1 Text="Pop" Checked="True"
    GroupName="muzik" runat="server"/>
<br>
<asp:RadioButton id=Radio2 Text="Jazz"
    GroupName="muzik" runat="server"/>
<br>
<asp:RadioButton id=Radio3 Text="Classic"
    GroupName="muzik" runat="server"/>
<br>
<asp:button text="Seçiniz" id="BtnOnay"
    OnClick="BtnOnay_Clicked" runat="server"/>
<br><br>
<span id="Message" runat="server" />

```

HyperLink

Hyperlink, sayfalar arası dolaşımı sağlar. **Hyperlink** kontrolünün görünümü metin veya resim olabilir. **ImageUrl** özelliği ile görüntülenecek resim dosyası belirlenir. **NavigateUrl** özelliği ile gidilecek sayfa belirlenir.

Örnekte Hyperlink kullanımı gösterilmektedir.

```

<asp:HyperLink id="hyperlink1" runat="server"
    ImageUrl="image1.gif"
    NavigateUrl="http://www.bilgeadam.com"
    Text="Bilge Adam BTA"
    Target="_blank"/>

```

Target özelliği, açılacak sayfanın aynı sayfa üzerinde veya yeni bir sayfada gösterilmesini sağlar. Tablo 8.2'de Target özelliğinin değerleri gösterilmektedir.

Target Özelliği	Açıklama
_blank	Yeni sayfa
_self	Aynı sayfa içinde
_search	Arama sayfası görünümünde

Tablo 7.2: Target Özelliğinin Değerleri

Image

Image, sayfa içinde resim görüntülemek için kullanılır. **ImageUrl** özelliği ile görüntülenecek resim dosyası belirlenir. **ImageAlign** özelliği resmin hizalanması için kullanılır. **AlternateText** resime alternatif metin göstermek için kullanılır.

Örnekte Image kullanımı gösterilmektedir.

```

<asp:Image id="Image1" runat="server"
    AlternateText="Logomuz"/>

```

```
ImageAlign="left"  
ImageUrl="logo.gif"/>
```

ImageButton

ImageButton resimli button kontroludur.

Örnekte ImageButton kullanımı gösterilmektedir.

```
void ImageButton_Click(object Source, ImageClickEventArgs e)  
{  
    Message.InnerHtml="Resimli Düğme kontrolünü Tıkladınız" +  
        " Koordinatlar: (" & e.X.ToString() + ", " +  
            e.Y.ToString() & ")" ;  
}  
<asp:ImageButton id="imagebutton1" runat="server"  
    AlternateText="Resimli Düğme Kontrolü"  
    ImageAlign="right"  
    ImageUrl="image1.gif"  
    OnClick="ImageButton_Click"/>  
<br><br>  
<span id="Message" runat="server"/>
```

ImageButton kontrolünün **ImageClickEventArgs** argümanı nesnesi kullanarak, kontrolün bulunduğu yerin koordinat değerleri alınabilir.

LinkButton

LinkButton, **HyperLink** görünümlü **Button** kontrolüdür. **LinkButton** kontrolünün **HyperLink** kontrolünden farkı ise olaylarının olmasıdır.

Örnekte LinkButton kullanımı gösterilmektedir.

```
void LinkButton1_Click(Object sender, EventArgs e)  
{  
    Label1.Text="Link Button'a tıkladınız" ;  
  
<asp:LinkButton Text="Mesajı Görmek İçin Tıklayınız."  
    Font-Name="Verdana" Font-Size="14pt"  
    onclick="LinkButton1_Click" runat="server"/>  
  
<br>  
<asp:Label id=Label1 runat=server />
```

Standart Kontroller

- DropDownList
 - Açılan kutuda veri görüntülemek için kullanılır.
- ListBox
 - DropDownList kontrolüne benzer. Elemanlar liste halinde gösterilir.
- Panel
 - Diğer kontrolleri gruplandırmak için kullanılır.
- Table
 - Satırlarına ve sütunlarına programlama yoluyla müdahale edilebilen tablo kontrolüdür.

DropDownList

DropDownList, açılan kutuda veri görüntülemek için kullanılır. DropDownList öğeleri **Items** koleksiyonunda tutulur. **Items** koleksiyonunun **Count** özelliği ile toplam öğe sayısı bulunur. DropDownList kontrolüne tasarım veya çalışma zamanında öğe eklenebilir.

Örnekte DropDownList kontrolüne tasarım zamanında öğe eklenmektedir.

```
void Button_Click(object sender, EventArgs e)
{
    Label1.Text = "Konuştuğunuz Dil " +
        dropdownlist1.SelectedItem.Text + "." ;
}

<asp:DropDownList id="dropdownlist1" runat="server">
    <asp:ListItem>Türkçe</asp:ListItem>
    <asp:ListItem>İngilizce</asp:ListItem>
    <asp:ListItem>Almanca</asp:ListItem>
    <asp:ListItem>İtalyanca</asp:ListItem>
</asp:DropDownList>
<asp:Button id="Button1" Text="Submit"
    onclick="Button_Click" runat="server"/>
<asp:label id="Label1" runat="server"/>
```

ListItem etiketi içindeki değerler, DropDownList öğelerini temsil eder.

Örnekte DropDownList kontrolüne çalışma zamanında öğe eklenmektedir.

```
<asp:DropDownList id="DropDownList1" style="Z-INDEX: 101;
```

```

LEFT: 128px; POSITION: absolute; TOP: 160px"
runat="server" width="152px"/>

void Page_Load(System.Object sender, System.EventArgs e)
{
    for (int i=0;i<=5;i++)
    {
        DropDownList1.Items.Add(i.ToString());
    }
}
Çalışma zamanında eleman eklemek için, Items koleksiyonunun Add metodu kullanılır.

```

ListBox

ListBox, **DropDownList** kontrolüne benzer. Elemanlar liste halinde gösterilir ve **SelectionMode** özelliğine **Multiple** değeri atanarak, çoklu seçim yapma imkânı sağlanır.

Örnekte ListBox kontrolünün çoklu seçim özelliği kullanılmaktadır. ListBox kontrolu içerisinde seçilen tüm elemanlar Label kontrolune yazdırılır.

```

public void SubmitBtn_Click(object sender, EventArgs e)
{
    ListItem item;
    Message.Text = "";
    foreach(ListItem item in ListBox1.Items)
    {
        if(item.Selected == true)
        {
            Message.Text += item.Text + " ";
        }
    }
}
<asp:ListBox id=ListBox1 Rows=4
SelectionMode="Multiple" width="100px" runat="server">
    <asp:ListItem>Türkçe</asp:ListItem>
    <asp:ListItem>İngilizce</asp:ListItem>
    <asp:ListItem>Almanca</asp:ListItem>
    <asp:ListItem>İtalyanca</asp:ListItem>
</asp:ListBox>
<br>
<asp:button Text="Submit" onclick="SubmitBtn_Click"
    runat="server" />
<br>
<asp:Label id="Message" runat="server"/>

```

Panel

Panel, diğer kontrolleri gruplandırmak için kullanılır.

Örnekte panel kullanımı gösterilmektedir.

```

public void Button1_Click(object sender, EventArgs e)
{
}

```

```

// Label kontrolü oluşturalım
Label label;
label = new Label();
label.Text = "Etiket";
label.ID = "Label1";
Panel1.Controls.Add(label);
Panel1.Visible = true;
}
<asp:Panel id="Panel1" runat="server"
    BackColor="blue"
    Height="150px"
    Width="200px"
    Visible=false>
    Panel1
    <p>
</asp:Panel>

<asp:Button id="Button1" onclick="Button1_Click"
    Text="Panel'i Göster" runat="server"/>

```

Panel1 isimli panel kontrolü başlangıçta gösterilmemektedir. Button1 düğmesi tıklanınca, panel kontrolünün içerisine label kontrolu eklenir ve görünür hale getirilir. **BackImageUrl** özelliği ile panele arka plan resmi verilir.

Table

Table, satırlarına ve sütunlarına programlama yoluyla müdahale edilebilen tablo kontroldür. **Table** kontrolü içinde **TableRow** ve **TableCell** nesneleri kullanılır. **TableCell**, tabloda bir hücreyi temsil eder. **TableRow** ise tabloda bir satırı temsil eder.

Örnekte Tabel kullanımı gösterilmektedir.

```

private void Page_Load(object sender, System.EventArgs e)
{
    //Satır ve Sütun Oluşumu
    int nrows = 3;
    int ncells = 2;
    int i;
    int j;
    for(j=0;j<=nrows - 1;j++)
    {
        TableRow r;
        r = new TableRow();
        for(i=1;i<=ncells;i++)
        {
            TableCell c;
            c = new TableCell();
            c.Controls.Add(new LiteralControl("Satır " +
j.ToString() + ", hücre " + i.ToString()));
            r.Cells.Add(c);
        }
        Table1.Rows.Add(r);
    }
}

```

```
<asp:Table id="Table1" GridLines="Both"
    HorizontalAlign="Center" Font-Name="Verdana"
    Font-Size="8pt" CellPadding=15 CellSpacing=0
    Runat="server"/>
```

Konu 5: Doğrulama(Validation) Kontroller

Doğrulama Kontroller

- **RequiredFieldValidator**
 - Veri girilmesi zorunlu alanlarda kullanılır.
- **CompareValidator**
 - Girilen değeri, sabit değerle veya başka bir kontrole girilen değerle karşılaştırır.
- **RangeValidator**
 - Kontrol içerisinde girilen değerin, İki sabit değer arasında olmasını sağlar.

Web forma girilecek verinin doğruluğunu kontrol etmek için sıkılıkla JavaScript fonksiyonları veya uzun ASP kodları kullanılırdı. Bu durum uygulama geliştirme sürecinin artmasına neden olurdu.

ASP.NET ile birlikte verinin doğruluğunu kontrol etmek için doğrulama kontrolleri geliştirildi. ASP.NET, belirli bir aralıkta veri girişi sağlayan, karşılaştırma yapan ve belirli değerlerin boş geçilmemesini sağlayan çeşitli doğrulama kontrolleri sunar. Tablo 7.1 de doğrulama kontrolleri listelenmiştir.

Validation Kontroller	Görevi
RequiredFieldValidator	Bir kontrol içerisinde değer girilip girilmediğini kontrol eder. Veri girilmesi zorunlu alanlarda kullanılır.
CompareValidator	Kontrol içerisinde girilen değeri, sabit değerle veya başka bir kontrole girilen değerle karşılaştırır.
RangeValidator	Kontrol içerisinde girilen değerin, İki sabit değer arasında olmasını sağlar.

RegularExpressionValidator	Bir kontrol içeresine girilen değerin, istenilen formatta girilmesini sağlar.
CustomValidator	Özel doğrulama kontrolu yazmayı sağlar.
ValidationSummary	Sayfada kullanılan tüm validation kontrollerin, doğrulama hatalarını özet olarak görüntüler.

Tablo 7.3: Doğrulama Kontrolleri

Doğrulama kontrollerinin ortak özellikleri aşağıdaki gibidir.

- **ControlToValidate:** Hangi kontrolün doğrulanacağını belirtir.
- **ErrorMessage:** Geçerli giriş yapılmamışsa görüntülenecek hata mesajını verir.
- **Text:** **ErrorMessage** ve **Text** özelliği birlikte kullanılabilir. Bu durumda **Text** özelliğindeki mesaj görüntülenir. Doğrulama kontrollerin **ErrorMessage** özelliğine girilen tüm mesajlar **ValidationSummary** içerisinde listelenir.
- **Display:** Validation kontrolünün nasıl görüntüleneceği bilgisini tutar. **Static**, **Dynamic** ve **None** değerleri alır.

RequiredFieldValidator

RequiredFieldValidator, belirtilen kontrolün boş geçilmemesini sağlar. Doğrulama yapılacak web kontrolünün ismi **ControlToValidate** özelliğine girilir. Geçerli giriş yapılmadığında ortaya çıkacak hata mesajı **ErrorMessage** özelliği ile belirtilir.

Örnekte **RequiredFieldValidator** kullanımı gösterilmektedir..

```
<asp:RequiredFieldValidator id="RequiredFieldValidator1"
    style="Z-INDEX: 103; LEFT: 224px; POSITION: absolute;
    TOP: 48px" runat="server"
    ErrorMessage="Adınızı Girmelisiniz!!!"
    ControlToValidate="txtad">
</asp:RequiredFieldValidator>
```

CompareValidator

Kontrol içeresine girilen değeri, sabit değerle veya başka bir kontrol ile karşılaştırmak için kullanılır. Doğrulama yapılacak web kontrolünün ismi **ControlToValidate** özelliğine girilir. Karşılaştırma yapılacak sabit değer **ValueToCompare** özelliğine girilir. **Type** özelliğine girilen değerin veri türü, **Operator** özelliğine ise mantıksal operatör girilir.

Örnekte txtYas kontrolune yirmi veya yirmiden büyük tam sayı girişi sağlayan doğrulama işlemi yapılmaktadır.

```
<asp:CompareValidator id="Comparevalidator1"
    style="Z-INDEX: 109; LEFT: 232px; POSITION: absolute;
    TOP: 88px" runat="server"
    ErrorMessage="Yaşınız 20 ye eşit veya büyük olmalıdır."
    ValueToCompare="20"
    ControlToValidate="txtYas"
    Type="Integer"
    Operator="GreaterThanOrEqualTo"
    width="128px">
</asp:CompareValidator>
```

Doğrulama yapılacak web kontrolü, başka bir web kontrolü ile karşılaştırılacağsa **ControlToCompare** özelliği kullanılır.

Örnekte txtYas kontrolünün değeri txtKontrol değerinden büyük olmalıdır.

```
<asp:CompareValidator id="Comparevalidator1"
    style="Z-INDEX: 109; LEFT: 240px; POSITION: absolute;
    TOP: 128px" runat="server"
    ErrorMessage="yaşınız kontrol alanında yazılan
    değerden büyük olmalıdır."
    ControlToValidate="txtYas"
    Type="Integer"
    Operator="GreaterThan"
    width="144px"
    ControlToCompare="txtKontrol">
</asp:CompareValidator>
```

RangeValidator

Kontrol içerisine girilen değerin, belirli bir değer aralığında olmasını sağlar. Doğrulama kontrollerinin ortak özelliklerine ek olarak **MinimumValue**, **MaximumValue** ve **Type** özellikleri vardır.

Örnekte txtYas kontrolune girilen değerin, otuzbeş ile ellı arasında olmasını sağlayan doğrulama işlemi yapılmaktadır. Bu özel karakterler Tablo 7.4

```
<asp:RangeValidator id="RangeValidator1"
    style="Z-INDEX: 109; LEFT: 240px; POSITION:
    absolute; TOP: 128px" runat="server"
    ErrorMessage="Yaşınız 35 ile 50 arasında olmalıdır."
    ControlToValidate="txtYas"
    Type="Integer"
    MaximumValue="50"
    MinimumValue="35">
</asp:RangeValidator>
```

Doğrulama Kontroller

- **RegularExpressionValidator**
 - Bir kontrol içerişine girilen değerin, istenilen formatta girilmesini sağlar.
- **CustomValidator**
 - Özel doğrulama kontrolu yazmayı sağlar.
- **ValidationSummary**
 - Sayfada kullanılan tüm validation kontrollerinin, doğrulama hatalarını özet olarak görüntüler.

RegularExpressionValidator

Kontrol içerişine girilen değerin, istenilen formatta girilmesini sağlar.

ValidationExpression özelliğine girilen özel karakterler ile veri giriş formatı sağlanır. Bu özel karakterler Tablo 7.4 de gösterilmektedir.

Örnekte RegularExpressionValidator kontrolu ile txtMail metin kutusu için geçerli e-mail girişini sağlanmaktadır.

```
<asp:regularexpressionvalidator
    id="RegularExpressionValidator1" style="Z-INDEX: 111;
    LEFT: 256px; POSITION: absolute; TOP: 168px"
    runat="server"
    ErrorMessage="Mail giriş hatası"
    ControlToValidate="txtMail"
    ValidationExpression=
        "\w+([-.\w+]*)@\w+([-.\w+]*)\.\w+([-.\w+]*)">
</asp:regularexpressionvalidator>
```

Karakter	Tanımı
a	Bir harf kullanımını zorunlu kılar.
1	1 sayısı kullanılmak zorunda.
?	0 veya 1 öğe olmak zorunda
*	0'dan n'e kadar bir değer
+	1'den n'e kadar bir değer

[0-n]	0'dan n'e kadar sayı değer dizisi
{n}	N ile belirtilen değer uzunlığında olmalı
	Farklı geçerli dizinler.
\	Bir komut karakterini devam ettiren karakter
\w	Bir karakter olmak zorunda.
\d	Bir rakam olmak zorunda.
\.	Bir nokta olmak zorunda.

Tablo 7.4: Kontrol Karakterleri

Örnekte **ValidationExpression** özelliğine girilen özel karakterler ile e-mail formatı oluşturulmaktadır.

```
ValidationExpression= "

```

\w+ : En az bir karakter içeren metin anlamına gelir.

([-.+]) : -, +, . karakterlerinden herhangi biri anlamına gelir.

***** : 0'dan n'e kadar bir değer girilmesi gerekiği anlamına gelir.

@ : @ işaretinin kullanılması gerektiğini belirtir.

Örnekte **ValidationExpression** özelliğine girilen özel karakterler ile e-mail formatı oluşturulmaktadır.

```
ValidationExpression ="\w+@\w+\.\w+"
```

CustomValidator

Aynı anda birden fazla nesnenin değerini kontrol etmek veya kullanıcı tanımlı kontrol yazmak için CustomValidator kontrolü kullanılır.

Özellik	Açıklama
ClientValidationFunction	İstemci fonksiyon ismini belirtir.
ControlToValidate	Doğrulama yapılacak kontrolü belirler.
Display	Text özelliğine girilen hata mesajının nasıl görüntüleneceği belirtir.
EnableClientScript	İstemci skriptleri aktif hale getirir. Varsayılan değer True dur.
Enabled	Sunucu ve istemci taraflı skriptleri aktif hale getirir. Varsayılan değer True dur.
ErrorMessage	Kontrol hata mesajını gösterir.
IsValid	Kontrol işlemi başarı ile sonuçlanmışsa True , değilse False değerini döndürür.
Text	Kontrol hata mesajını gösterir. ErrorMessage ve Text özelliği birlikte kullanılabilir. Bu durumda Text özelliğindeki mesaj görüntülenir.

Tablo 7.3: CustomValidator Kontrolünün Özellikleri

ServerValidate olayı ve **OnServerValidate** metodu ile sunucu taraflı kontroller tetiklenir.

Örnekte sunucu taraflı metod oluşturulmaktadır.

```
public void CustomValidator_SunucuKontrol(object s,
ServerValidateEventArgs e)
{
}
```

ServerValidateEventArgs parametresinin iki özelliği vardır:

IsValid: Bu özellik **True** ise kontrol içerisinde girilen değerin doğruluğu sağlanmıştır.

Value: Doğrulama kontrolünün değerini verir.

CustomValidator, kredi kart numaralarının doğruluğu kontrol etmek için kullanılabilir.

Örnekte metin kutusuna girilen değerin çift olması kontrol edilmektedir. **CustomValidator** kontrolünün HTML kodları aşağıdaki gibidir.

```
<form id="Form1" method="post" runat="server">

<asp:Button id="Button1" style="Z-INDEX: 101; LEFT:
200px; POSITION: absolute; TOP: 96px" runat="server"
Text="gonder" onclick="gonder_OnClick">
</asp:Button>

<asp:CustomValidator id="CustomValidator1"
```

```

        style="Z-INDEX: 102; LEFT: 312px; POSITION: absolute;
        TOP: 64px" runat="server"
        ErrorMessage="çift rakam giriniz."
        ControlToValidate="txtcustom" Display="Static"
        OnServerValidate="ServerKontrol">
    </asp:CustomValidator>

    <asp:TextBox id="txtmessage" style="Z-INDEX: 103;
        LEFT: 200px; POSITION: absolute; TOP: 152px"
        runat="server">
    </asp:TextBox>

    <asp:TextBox id="txtcustom" style="Z-INDEX: 104;
        LEFT: 144px; POSITION: absolute; TOP: 64px"
        runat="server">
    </asp:TextBox>
</form>
```

CustomValidator kontrolünün C# kodları aşağıdaki gibidir.

```

public void ServerKontrol(object source,
ServervalidateEventArgs args)
{
    try
    {

        int num = int.Parse(args.Value);
        args.IsValid = ((num%2) == 0);

    }
    catch(Exception ex)
    {
        args.IsValid = false;
    }
}

public void gonder_OnClick(object sender, EventArgs e)
{
    if(Page.IsValid)
    {
        txtmessage.Text = "Sayfada Hata Yok.";
    }
    else
    {
        txtmessage.Text = "Sayfa Hatalı!";
    }
}
```

ValidationSummary

ValidationSummary, doğrulama kontrollerin **ErrorMessage** özelliğine girilen tüm mesajları listeler.

Örnekte **ValidationSummary** kullanımı gösterilmektedir.

```

<asp:ValidationSummary id="ValidationSummary1"
    style="Z-INDEX: 114; LEFT: 72px; POSITION: absolute;
    TOP: 336px" runat="server"
    HeaderText="Hatalar">
</asp:ValidationSummary>
```

DisplayMode özelliği ile ValidationSummary kontrolünün görüntüsü değiştirilebilir. **DisplayMode**, **BulletList**, **List** ve **SingleParagraph** değerlerini alır. **ShowMessageBox**, hata listesinin mesaj kutusu içinde görüntülenmesini sağlar.

Konu 8: Zengin Kontroller

Zengin Kontroller

▪ AdRotator

- Web sayfaları üzerinde reklam resimleri görüntülemek için kullanılır.
- Ayarları XML dosya içerisinde kaydedilir.

▪ Calendar

- Web sayıları üzerinde Takvim görüntülemek için kullanılır.

AdRotator

AdRotator, Web sayfaları üzerinde reklam yayını yapmak için kullanılır. Reklam için kullanılan banner nesneleri XML dosya içerisinde kaydedilir.

Örnekte AdRotator kullanımı için gerekli XML dosya (Ads.Xml) gösterilmektedir.

```
<Advertisements>
  <Ad>
    <ImageUrl>image1.gif</ImageUrl>
    <NavigateUrl>http://www.bilgeadam.com</NavigateUrl>
    <AlternateText>BilgeAdam BTA</AlternateText>
    <Impressions>80</Impressions>
    <Keyword>Yazılım</Keyword>
  </Ad>
  <Ad>
    <ImageUrl>image2.gif</ImageUrl>
    <NavigateUrl>http://www.microsoft.com</NavigateUrl>
    <AlternateText>Microsoft Site</AlternateText>
    <Impressions>80</Impressions>
```

```
<Keyword>microsoft</Keyword>
</Ad>
</Advertisements>
```

Ad: Her bir banner nesnesini temsil eder.

ImageUrl: Banner içerisinde görüntülenecek resim dosyasını belirtir.

NavigateUrl: Gidilecek sayfanın adres bilgisini belirtir.

AlternateText: Resime alternatif metin göstermek için kullanılır.

Impression: -Banner resimlerinin uygulama süresini belirtir.

Keyword: .Banner nesneleri arasında filtreleme oluşturacak kategori adını belirler.

AdRotator kontrolünün **AdvertisementFile** özelliği, reklâm bilgilerinin bulunduğu XML dosyayı içerir. **KeywordFilter** özelliği ise reklâmlarınfiltrelenerek görüntülenmesini sağlar.

Calendar

Calendar, web sayıları üzerinde Takvim göstermek için kullanılır..

Örnekte Calender kullanımı gösterilmektedir.

```
<asp:Calendar id="takvim" runat="server"/>
```

Calendar kontrolüne ait özellikler tablo 7.4'de listelenmiştir.

Özellik	Açıklama
CellPadding	Hücreler ve kenarlıklar arasındaki boşluk değerini tutar.
CellSpacing	Hücreler arasındaki boşluk değerini tutar.
DayNameFormat	Gün isimlerinin görüntülenme biçimini tutar. FirstLetter , FirstTwoLetters , Full ve Short değerleri vardır. Varsayılan Short değeridir.
FirstDayOfWeek	Haftanın ilk gününü belirtir.
NextPrevFormat	Next ve Previous linklerinin biçimini ayarlar. CustomText , FullMonth , ShortMonth değerleri alır. Varsayılan CustomText değeridir.
SelectedDate	Seçilen gün bilgisini tutar. Varsayılan değer günün tarihidir.
SelectionMode	Calendar nesnesinin seçim modunu belirler. Day , DayWeek , DayWeekMonth ve None değerleri

	alır. Varsayılan seçenek Day değeridir.
ShowDayHeader	Gün isimlerini kolonların üzerinde görüntüler.
ShowGridLines	Günleri hücreler içinde görüntüler.
ShowNextPrevMonth	Önceki ve sonraki ay linklerinin görüntülenmesini sağlar.
ShowTitle	Takvim başlığını görüntüler.
TitleFormat	Başlık yazısının biçimini belirtir.

Tablo 7.4: Calendar Kontrolünün Özellikleri

Örnekte Calender kullanımı gösterilmektedir.

```
<asp:Calendar id="Calendar1" style="Z-INDEX: 105; LEFT: 160px; POSITION: absolute; TOP: 248px" runat="server" CellSpacing="2" Width="240px" Height="152px" BorderStyle="Groove" DayNameFormat="Full" NextPrevFormat="FullMonth">
    <DayStyle Font-Bold="True" HorizontalAlign="Center" BorderStyle="None" BorderColor="Transparent" VerticalAlign="Top" BackColor="LightCyan">
    </DayStyle>
    <DayHeaderStyle Font-Underline="True" Font-Italic="True" HorizontalAlign="Right" BorderWidth="3px" ForeColor="DarkBlue" BorderStyle="Groove" BorderColor="#C0FFFF" VerticalAlign="Top" BackColor="#FFCC00">
    </DayHeaderStyle>
    <WeekendDayStyle BackColor="Salmon"></WeekendDaystyle>
</asp:Calendar>
```

Konu 9: AutoPostBack Özelliği

AutoPostBack Özelliği

Bir sunucu kontrolünün web sunucuya otomatik olarak bilgi göndermesini sağlar.

- Bu özelliği destekleyen kontroller;
 - DropDownList,
 - ListBox,
 - CheckBox,
 - CheckBoxList,
 - RadioButton,
 - RadioButtonList,
 - TextBox
 - Button

AutoPostBack özelliği, herhangi bir sunucu kontrolünün web sunucuya otomatik olarak bilgi göndermesini sağlar. Web formu dolduruktan sonra sunucuya göndermek için genellikle button kontrolü kullanılır.

AutoPostBack özelliği, **DropDownList**, **ListBox**, **CheckBox**, **CheckBoxList**, **RadioButton**, **RadioButtonList**, **TextBox** ve **Button** kontrolünde bulunur.

Bu özelliğin **True** olması, seçim yapıldığında veya **TextBox** kontrolüne yeni bir değer girildiğinde sayfanın yeniden yüklenmesi anlamına gelir.

Örnekte AutoPostBack kullanımı gösterilmektedir.

```
<%@ Page Language="C#" Debug="true" %>
<html>
<head></head>
<body>
    <form runat="server">
        Bilgiiniz Yabancı Dili Seçiniz:<br/><br/>
        <asp:listbox id="lstDiller" runat="server" rows="3"
            AutoPostBack="true"
            onSelectedIndexChanged="secimGoster"/>
        <br><br>
        <asp:Label id=lblMesaj runat="server" /> <br/><br/>
    </form>
</body>
</html>
<script language=c# runat="server">
```

```

private void Page_Load(object sender, System.EventArgs e)
{
    if(!Page.IsPostBack)
    {
        lstFlowers.Items.Add(new ListItem("İngilizce"));
        lstFlowers.Items.Add(new ListItem("Almanca"));
        lstFlowers.Items.Add(new ListItem("Fransızca"));
        lstFlowers.SelectedIndex = 0;
    }
}
public void showSelection(object source, EventArgs e)
{
    lblMesaj.Text = "Seçtiğiniz Dil " +
    lstDiller.SelectedItem.Text;
}

</script>

```

AutoPostBack özelliğinin gereksiz yere kullanılması performansı olumsuz yönde etkiler.

Konu 10: ViewState

ViewState

- ViewState

- Kullanıcı ve sunucu arasında taşınan verilerin gizli bir alanda şifrelenerek saklanması sağlar.

```
<input type="hidden" name="__VIEWSTATE"
value="dDwtMTY5NzI1NjkxNzAdJkhGyy1Rw1gGcc+ia" />
```

ViewState, kullanıcı ve sunucu arasında taşınan verilerin gizli bir alanda şifrelenerek saklanması sağlar. Forma ait tüm kontrollerin değerleri şifrelenir ve **VIEWSTATE** değişkende saklanır. Form sunucuya gönderildikten sonra bir hata oluşması halinde kullanıcının tekrar aynı verilerin girilmesi istenmez. Çünkü kullanıcının girmiş olduğu değerler bu gizli değişkende saklanır ve sayfa açılınca tekrar kullanıcıya sunulur.

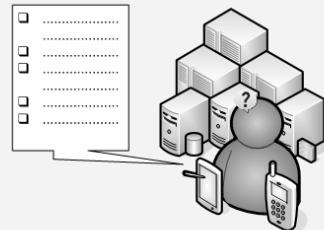
Örnekte ViewState kullanımı gösterilmektedir.

```
<input type="hidden" name="__VIEWSTATE"  
value="dDwtMTY5NzI1NjkxNzs7Po5lYTU9gAdJkhGyy1Rw1gGcc+ia" />
```

Modül Özeti

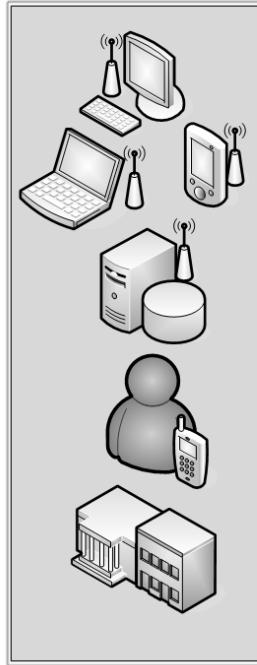
Modül Özeti

- Web form bileşenlerini açıklayın?
- Stardart kontroller nelerdir ? Açıklayın.
- Doğrulama kontrolleri nelerdir ? Açıklayın.
- Zengin kontroller nelerdir ? Açıklayın.
- AutoPostBack nedir ?
- ViewState özelliğini açıklayın?



15. Web form bileşenlerini açıklayın?
16. Stardart kontroller nelerdir ? Açıklayın.
17. Doğrulama kontrolleri nelerdir ? Açıklayın.
18. Zengin kontroller nelerdir ? Açıklayın.
19. AutoPostBack nedir ?
20. ViewState özelliğini açıklayın?

Lab 1: E-Ticaret Uygulaması Geliştirmek



Uygulamalar

ASP.NET Giriş

- ◆ E-Ticaret Uygulaması
Geliştirmek

Bu uygulamada, e-ticaret uygulamasının web formları tasarılanacaktır. E-ticaret uygulaması içerisinde müşterilerinin kendi kayıtlarını yapabilmesi için UyeKayıt formu tasarlanacaktır. Kayıt olan müşterilerin ürün satın alabilmesi için sisteme giriş yapmaları gerekmektedir. Kayıtlı müşterilerin sisteme girişi UyeGiris formu ile sağlanacaktır. Her iki form içerisinde standart ve doğrulama kontrolleri kullanılacaktır. Ayrıca kullanıcıyı yönlendirmek için Giriş, Kayıt ve Satış isminde 3 ayrı web form tasarılanacaktır.

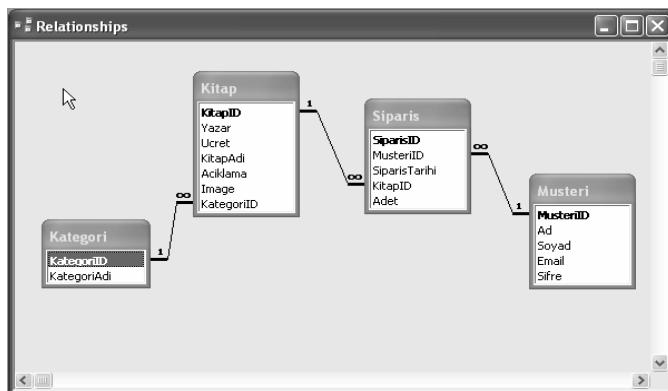
Bu lab tamamlandıktan sonra:

- Access veritabanına bağlantı oluşturabilecek,
- Web Formları tanıyacak,
- Standart Kontrolleri kullanabilecek,
- Doğrulama kontrollerini kullanabileceksiniz.

Veritabanının projeye eklenmesi

Bu uygulamada kullanılacak Course veritabanı oluşturun.

10. Microsoft Access ile “KitapDb” isminde bir veritabanı oluşturun.
11. Veritabanın tablolarını aşağıdaki diyagrama göre oluşturun.



12. Veritabanı içerisinde EnCokSatanlar isminde sorgu oluşturun.

13. Sorgunun içerisinde aşağıdaki Select cümlesini ekleyin.

```

SELECT Kitap.KitapAdi, Kitap.Ucret, Kitap.KitapID
FROM Kitap INNER JOIN Siparis ON
Kitap.KitapID=Siparis.KitapID
GROUP BY Kitap.KitapAdi, Kitap.Ucret, Kitap.KitapID
ORDER BY Count(Siparis.Adet) DESC;
    
```

Web Formların eklenmesi

AspEticaret isminde yeni bir ASP.NET Web Application projesi açın.

UyeKayit formunun eklenmesi

ASPEticaret projesine UyeKayit isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
TextBox – txtAd		
TextBox – txtSoyad		
TextBox – txtEmail		
TextBox – txtSifre	TextMode	Password
TextBox – txtSifreDogrula	TextMode	Password
RequiredFieldValidator – rfvAd	ControlToValidate	txtAd
	ErrorMessage	Adı boş geçemezsiniz
	Text	*
RequiredFieldValidator – rfvSoyad	ControlToValidate	txtSoyad
	ErrorMessage	Soyadı boş geçemezsiniz
	Text	*

RequiredFieldValidator – rfvEmail	ControlToValidate	txtEmail
	ErrorMessage	E-maili boş geçemezsiniz
	Text	*
RequiredFieldValidator – rfvSifre	ControlToValidate	txtSifre
	ErrorMessage	Şifreyi giriniz.
	Text	*
RequiredFieldValidator – rfvSifre2	ControlToValidate	txtSifreDogrula
	ErrorMessage	Doğrulama şifresini giriniz
	Text	*
RegularExpressionValidator – revEmail	ControlToValidate	txtEmail
	ErrorMessage	Hatalı Email
	Text	*
	Validation Expression	\w+([-.\w+]*)*\@\w+([-.\w+]*)*\.\w+([-.\w+]*)*
CompareValidator – cvSifreDogrula	ControlToCompare	txtSifre
	ControlToValidate	txtSifre2
	ErrorMessage	Şifreler uyumsuz
	Text	*
ValidationSummary – vsHata		
Button - btnKaydet	Text	Kaydet

Yazılım Uzmanı Kitapevi

AnaSayfa | Üye Giriş | Üye Kayıt

- [Klasik](#)
- [Bilgisayar](#)
- [Edebiyat](#)
- [Roman](#)
- [Şair](#)
- [Fıkra](#)
- [Macera](#)
- [Bilim-Kurgu](#)
- [Tarih](#)
- [Sağlık](#)
- [Tarih](#)

Üyelik Bilgileri

Ad*	<input type="text"/>
Soyad*	<input type="text"/>
E-Mail*	<input type="text"/>
Şifre*	<input type="password"/>
Şifre Doğrula*	<input type="password"/>
<input type="button" value="Kaydet"/>	

[AnaSayfam Yap](#) | [Sık Kullanılanlara Ekle](#) | [İletişim](#)

UyeKayit Web formun Html kodları aşağıdaki gibi olacaktır.

```
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx"
%>
<%@ Page Language="C#" AutoEventWireup="false"
Codebehind="UyeKayit.aspx.cs"
Inherits="AspEticaret.UyeKayit" %>
<%@ Register TagPrefix="uc1" TagName="kategori"
Src="kategori.ascx" %>
<%@ Register TagPrefix="uc1" TagName="ust" Src="ust.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
    <HEAD>
        <title>UyeKayit</title>

    <meta content="Microsoft Visual Studio .NET 7.1"
name="GENERATOR">
        <meta content="C#" name="CODE_LANGUAGE">
        <meta content="JavaScript"
name="vs_defaultClientScript">
        <meta
content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
    </HEAD>
    <body bgColor="#f0fff0">
        <form id="Form1" method="post" runat="server">
            <TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" align="center" border="0">
                <TR>
                    <TD bgColor="#99ccff" colspan="3"><uc1:ust
id="ust1" runat="server"></uc1:ust></TD>
                </TR>
                <TR>
                    <TD vAlign="top" width="150"><uc1:kategori
id="Kategori1" runat="server"></uc1:kategori></TD>
                    <TD width="400" height="100%">
                        <P>
                            <TABLE id="Table2" cellSpacing="0"
cellPadding="0" width="300" align="center" border="0">
                                <TR>
                                    <TD style="HEIGHT: 48px"
colspan="2">
                                        <P align="center"><FONT
face="Tahoma" size="2"><STRONG>Üyelik
Bilgileri</STRONG></FONT></P>
                                    </TD>
                                </TR>
                                <TR>
                                    <TD style="WIDTH: 100px"><FONT
face="Tahoma" size="2">Ad*</FONT></TD>
                                    <TD><asp:textbox id="txtAd"
runat="server"
Width="176px"></asp:textbox><asp:requiredfieldvalidator
id="rfvAd" runat="server" ControlToValidate="txtAd"
ErrorMessage="Adı boş
geçemezsiniz">*</asp:requiredfieldvalidator><FONT
face="Tahoma" size="2"></FONT></TD>
                                </TR>
                                <TR>
                                    <TD style="WIDTH: 100px"><FONT
face="Tahoma" size="2">Soyad*</FONT></TD>
                                    <TD><asp:textbox id="txtSoyad"
runat="server">
```

```

width="176px"></asp:textbox><asp:requiredfieldvalidator
id="rfvSoyad" runat="server" ControlToValidate="txtSoyad"
ErrorMessage="Soyadı boş
geçemezsiniz.">*</asp:requiredfieldvalidator><FONT
face="Tahoma" size="2"></FONT></TD>
</TR>
<TR>
    <TD style="WIDTH: 100px"><FONT
face="Tahoma" size="2">E-Mail*</FONT></TD>
        <TD><asp:textbox id="txtEmail"
runat="server"
width="176px"></asp:textbox><asp:requiredfieldvalidator
id="rfvEmail" runat="server" ControlToValidate="txtEmail"
ErrorMessage="E-maili boş geçemezsiniz.

Display="Dynamic">*</asp:requiredfieldvalidator><asp:regu
larexpressionvalidator id="revEmail" runat="server"
ControlToValidate="txtEmail" ErrorMessage="Hatalı Email"
Display="Dynamic"
validationExpression="\w+([-.]\w+)*@\w+([-.]\w+)*\.\w+([-.]
]\w+)*">*</asp:regularexpressionvalidator><FONT
face="Tahoma" size="2"></FONT></TD>
</TR>
<TR>
    <TD style="WIDTH: 100px"><FONT
face="Tahoma" size="2">Şifre*</FONT></TD>
        <TD><asp:textbox id="txtSifre"
runat="server" width="176px"
TextMode="Password"></asp:textbox><asp:requiredfieldvali
dator id="rfvSifre" runat="server" ControlToValidate="txtSifre"
ErrorMessage="Şifreyi
giriniz.">*</asp:requiredfieldvalidator><FONT face="Tahoma"
size="2"></FONT></TD>
</TR>
<TR>
    <TD style="WIDTH: 100px"><FONT
face="Tahoma" size="2">Şifre Doğrula*</FONT></TD>
        <TD><asp:textbox
id="txtSifreDogrula" runat="server" width="176px"
TextMode="Password"></asp:textbox><asp:requiredfieldvali
dator id="rfvSifre2" runat="server"
ControlToValidate="txtSifreDogrula" ErrorMessage="Doğrulama
şifresini giriniz.

Display="Dynamic">*</asp:requiredfieldvalidator><asp:comp
arevalidator id="cvSifreDogrula" runat="server"
ControlToValidate="txtSifreDogrula" ErrorMessage="Şifreler
uyumsuz.">
            <TD style="HEIGHT: 47px"
colspan="2">
                <P align="center"><asp:button
id="btnKaydet" runat="server"
Text="Kaydet"></asp:button></P>
            </TD>
        </TR>
        <TR>
            <TD style="WIDTH: 100px"
colspan="2"><asp:validationsummary id="vsHata"
runat="server" width="286px"></asp:validationsummary></TD>
            </TR>
        <TR>

```

```

        <TD style="WIDTH: 100px; HEIGHT:
15px" colspan="2"></TD>
    </TR>
    <TR>
        <TD style="WIDTH: 100px"></TD>
        <TD></TD>
    </TR>
    </TABLE>
</P>
</TD>
<TD vAlign="top" width="150"
bgColor="#0099ff"><uc1:yan id="Yan1"
runat="server"></uc1:yan></TD>
</TR>
<TR>
    <TD bgColor="#99ccff" colspan="3">
        <uc1:Alt id="Alt1"
runat="server"></uc1:Alt></TD>
    </TR>
</TABLE>
</form>
</body>
</HTML>

```

UyeGiris formunun eklenmesi

ASPEticaret projesine UyeGiris isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
TextBox – txtEmail		
TextBox – txtSifre	TextMode	Password
	Text	*
Button – btnGiris	Text	Giriş
Label – lblMesaj	Text	

UyeGiriş Web formun Html kodları aşağıdaki gibi olacaktır.

```
<%@ Register TagPrefix="uc1" TagName="kategori"
Src="kategori.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Ust" Src="Ust.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="C#" AutoEventWireup="false"
Codebehind="UyeGiris.aspx.cs"
Inherits="AspEticaret.UyeGiris" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
    <HEAD>
        <title>UyeGiris</title>
        <meta name="GENERATOR" content="Microsoft Visual
Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" content="C#">
        <meta name="vs_defaultClientScript"
content="JavaScript">
        <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body bgColor="#f0ffff">
        <form id="Form1" method="post" runat="server">
            <TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" align="center" border="0">
                <TR>
                    <TD bgColor="#99ccff" colspan="3">
                        <uc1:Ust id="Ust1"
runat="server"></uc1:Ust></TD>
                    </TR>
                    <TR>
                        <TD width="150" vAlign="top">
                            <uc1:kategori id="Kategori1"
runat="server"></uc1:kategori></TD>
                        <TD width="400" vAlign="top">
                            <P>
                                <TABLE id="Table2" cellSpacing="0"
cellPadding="0" width="200" align="center" border="0">
                                    <TR>
                                        <TD style="HEIGHT: 63px"
colspan="2">
                                            <P align="center"><STRONG><FONT
face="Verdana" size="2"><BR>
                                                Üye
                                            <STRONG><FON
Giriş</FON
Giriş</STRONG></P>
                                            </TD>
                                        </TR>
                                        <TR>
                                            <TD style="WIDTH: 66px">E-
                                                Mail </TD>
                                            <TD>
                                                <asp:TextBox id="txtEmail"
runat="server" width="128px"></asp:TextBox></TD>
                                            </TR>
                                            <TR>
                                                <TD style="WIDTH: 66px; HEIGHT:
11px">Sifre</TD>
                                                <TD style="HEIGHT: 11px">
                                                    <asp:TextBox id="txtSifre"
runat="server" width="127px"
TextMode="Password"></asp:TextBox></TD>

```

```

        </TR>
        <TR>
            <TD style="HEIGHT: 54px"
colspan="2">
                <P align="center">
                    <asp:Button id="btnGiris"
runat="server" Text="Giriş"></asp:Button></P>
                </TD>
            </TR>
            <TR>
                <TD colspan="2">
                    <asp:Label id="lblMesaj"
runat="server"></asp:Label></TD>
                </TR>
            </TABLE>
        </P>
    </TD>
    <TD width="150" bgColor="#0099ff" vAlign="top">
        <uc1:yan id="Yan1"
runat="server"></uc1:yan></TD>
    </TR>
    <TR>
        <TD bgColor="#99ccff" colspan="3">
            <uc1:Alt id="Alt1"
runat="server"></uc1:Alt></TD>
        </TR>
    </TABLE>
</form>
</body>
</HTML>

```

Giris formunun eklenmesi

ASPEticaret projesine Giriş isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
HyperLink – lnk1	NavigateUrl	UyeGiris.aspx
	Text	Tıklayınız
HyperLink – lnk2	NavigateUrl	UyeKayit.aspx
	Text	Tıklayınız

The screenshot shows a website layout. At the top right are links for 'AnaSayfa', 'Üye Giriş', and 'Üye Kayıt'. On the left, a sidebar lists book categories: Klasik, Bilgisayar, Edebiyat, Roman, Sür, Fıkra, Macera, Bilim-Kurgu, Tarih, Saçılık, and Tarih. To the right of the sidebar, there is a message: 'Sayfaya erişmek için üye girişi yapmalısınız.' followed by 'Giriş yapmak için [tıklayınız](#)'. Below this message is another link: 'Üye olmak için [tıklayınız](#)'. In the bottom right corner of the main content area, there is a logo for 'BilgeAdam' with the text 'YAZILIM UZMANI' and 'BilgeAdamsd.net club'.

Giriş Web formun Html kodları aşağıdaki gibi olacaktır.

```

<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Ust" Src="Ust.ascx" %>
<%@ Register TagPrefix="uc1" TagName="kategori"
Src="kategori.ascx" %>
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="C#" AutoEventWireup="false"
Codebehind="Giris.aspx.cs" Inherits="AspEticaret.Giris" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
    <HEAD>
        <title>Kayıt</title>
        <meta name="GENERATOR" content="Microsoft Visual
Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" content="C#">
        <meta name="vs_defaultClientScript"
content="JavaScript">
        <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body bgColor="#f0ffff0">
        <form id="Form1" method="post" runat="server">
            <TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" align="center" border="0">
                <TR>
                    <TD bgColor="#99ccff" colspan="3">
                        <uc1:Ust id="Ust1"
runat="server"></uc1:Ust></TD>
                    </TR>
                    <TR>
                        <TD width="150" vAlign="top">
                            <uc1:kategori id="Kategori1"
runat="server"></uc1:kategori></TD>
                        <TD width="400" vAlign="top">
                            <P><BR>
                                Sayfaya erişmek için üye girişi
yapmalısınız.<BR>
                                Giriş yapmak için
                                <asp:HyperLink id="lnk1" runat="server"
NavigateUrl="UyeGiris.aspx">tıklayınız</asp:HyperLink></P>
                            <P>Üye olmak için

```

```

<asp:HyperLink id="lnk2" runat="server"
NavigateUrl="UyeKayit.aspx">tıklayınız</asp:HyperLink></P>
</TD>
<TD width="150" bgColor="#0099ff" vAlign="top">
<uc1:yan id="Yan1"
runat="server"></uc1:yan></TD>
</TR>
<TR>
<TD bgColor="#99ccff" colspan="3">
<uc1:Alt id="Alt1"
runat="server"></uc1:Alt></TD>
</TR>
</TABLE>
</form>
</body>
</HTML>

```

Kayıt formunun eklenmesi

ASPEticaret projesine Kayıt isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
HyperLink – lnk1	NavigateUrl	UyeGiris.aspx
	Text	tıklayınız



Kayıt web formun Html kodları aşağıdaki gibi olacaktır.

```

<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="c#" AutoEventWireup="false"
Codebehind="Kayit.aspx.cs" Inherits="AspETicaret.Kayit" %>
<%@ Register TagPrefix="uc1" TagName="kategori"
Src="kategori.ascx" %>
<%@ Register TagPrefix="uc1" TagName="ust" Src="ust.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>

```

```

<HEAD>
    <title>Kayıt</title>
    <meta name="GENERATOR" content="Microsoft Visual
Studio .NET 7.1">
    <meta name="CODE_LANGUAGE" content="C#">
    <meta name="vs_defaultClientScript"
content="JavaScript">
    <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
</HEAD>
<body bgColor="#f0ffff">
    <form id="Form1" method="post" runat="server">
        <TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" align="center" border="0">
            <TR>
                <TD bgColor="#99ccff" colspan="3">
                    <uc1:Ust id="Ust1"
runat="server"></uc1:Ust></TD>
                </TR>
                <TR>
                    <TD width="150" vAlign="top">
                        <uc1:kategori id="Kategori1"
runat="server"></uc1:kategori></TD>
                    <TD width="400" vAlign="top">
                        <P><BR>
                            Kayıt işlemi başarıyla tamalandı.<BR>
                            Giriş yapmak için
                            <asp:HyperLink id="lnk1" runat="server"
NavigateUrl="UyeGiris.aspx">tıklayınız</asp:HyperLink></P>
                    </TD>
                    <TD width="150" bgColor="#0099ff" vAlign="top">
                        <uc1:yan id="Yan1"
runat="server"></uc1:yan></TD>
                    </TR>
                    <TR>
                        <TD bgColor="#99ccff" colspan="3">
                            <uc1:Alt id="Alt1"
runat="server"></uc1:Alt></TD>
                        </TR>
                    </TABLE>
                </form>
            </body>
        </HTML>
    
```

Satis formunun eklenmesi

ASPEticaret projesine Satis isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Deger
HyperLink – lnk1	NavigateUrl	Default.aspx
	Text	tıklayınız



Satış web formun Html kodları aşağıdaki gibi olacaktır.

```

<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Ust" Src="Ust.ascx" %>
<%@ Register TagPrefix="uc1" TagName="kategori"
Src="kategori.ascx" %>
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="c#" AutoEventWireup="false"
Codebehind="Satis.aspx.cs" Inherits="AspEticaret.Satis" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
    <HEAD>
        <title>Kayıt</title>
        <meta name="GENERATOR" content="Microsoft Visual
Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" content="C#">
        <meta name="vs_defaultClientScript"
content="JavaScript">
        <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body bgColor="#f0ffff">
        <form id="Form1" method="post" runat="server">
            <TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" align="center" border="0">
                <TR>
                    <TD bgColor="#99ccff" colspan="3">
                        <uc1:Ust id="Ust1"
runat="server"></uc1:Ust></TD>
                    </TR>
                    <TR>
                        <TD width="150" vAlign="top">
                            <uc1:kategori id="Kategori1"
runat="server"></uc1:kategori></TD>
                        <TD width="400" vAlign="top">
                            <P><BR>
                                Satış  İşlemi başarıyla
tamalandı.<BR>
                                Devam etmek  İçin
                                <asp:HyperLink id="link1" runat="server"
NavigateUrl="Default.aspx">tıklayınız</asp:HyperLink></P>
                        </TD>
                        <TD width="150" bgColor="#0099ff" vAlign="top">

```

```
<uc1:yan id="Yan1"
runat="server"></uc1:yan></TD>
</TR>
<TR>
<TD bgColor="#99ccff" colspan="3">
<uc1:Alt id="Alt1"
runat="server"></uc1:Alt></TD>
</TR>
</TABLE>
</form>
</body>
</HTML>
```

ASP.NET İle Kod Geliştirmek



Modül 8: ASP.NET ile Kod Geliştirmek

ASP.NET İle Kod Geliştirmek

- ◆ Kod Yazmak
- ◆ İstemci Taraflı Olay Prosedürleri
- ◆ Sunucu Taraflı Olay Prosedürleri
- ◆ Sayfa Yaşam Döгüsü

Bu modülde Visual Studio .NET ortamı içerisinde ASP.NET uygulamalarının kullanım yollarını öğreneceksiniz.

Bu modül tamamlandıktan sonra:

- Inline ve Code Behind kod yazmayı öğrenecek,

- Client Side – Server Side olay prosedürlerini öğrenecek,
- Page Event yaşam döngüsünü tanıယaksınız.

Konu 1: Kod Yazmak

Kod Yazmak

- Web Form' a kod ekleme yöntemleri:
 - Mixed Code
 - Web içeriği ile kod aynı sayfada.
 - Inline Code
 - Kod, aynı HTML dosyasında Script etiketi içerisinde.
 - Code-behind
 - Kod ve HTML içeriği farklı dosylarda.

Web Form içerisinde kullanılan kontrollere ait HTML kodları ve bu kontrollere ait Visual Basic .NET kodları bulunur. Web Formların en önemli özelliği ise, tasarım ve kod ara yüzlerinin ayrı tutulmasıdır.

Web Forma kod eklemek için üç yol izlenir:

- **Mixed Code:** Bu metotta web içeriği ile kod aynı sayfa içerisinde yazılır. Bu metot pek tercih edilmez çünkü okunması ve düzenlenmesi zordur.
- **Inline Code:** Web içeriği ile kod aynı sayfa içerisinde yer alır. ASP.NET kodu **Script** etiketi içerisine yazılır.
- **Code-behind:** HTML içeriği ve Visual Basic .NET kodu tamamen ayrı dosyalarda tutulur. Kod dosyasına code-behind sayfası denir. Bu metot Visual Studio .NET ortamının varsayılan çalışma şeklidir.

Inline Kod Yazmak

- ### Inline Code Yazmak
- Server taraflı kodlar Script etiketi içersine yazılmalıdır ve runat="server" özelliği belirtilmelidir.

```
<HTML>
<asp:Button id="btnTamam" runat="server"/>
...
</HTML>

<SCRIPT Language="c#" runat="server">
    private void btnTamam_Click(object sender, System.EventArgs e)
    {
        ...
    }
</SCRIPT>
```

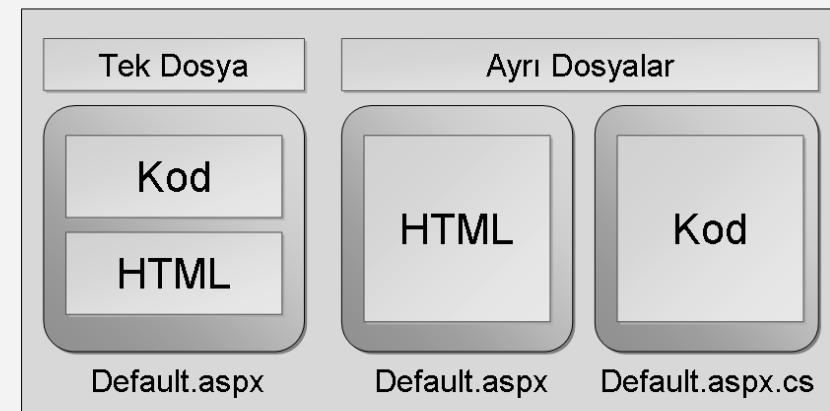
Aynı .aspx dosyası içinde HTML kodu ve Visual Basic .NET kodu ayrı böümlere yazılır. Bölümllerin ayrı tutulması okunabilirliği arttırr. Server taraflı kodlar **Script** etiketi içersine yazılmalıdır ve **runat="server"** özelliği belirtilmelidir.

Kod 8.1: Inline Kod Yazmak

```
<%@ Page Language="C#" %>
<html>
<head>
<title>My First Web Form</title>
<script runat="server">
    private void Page_Load (Sender As Object , e As EventArgs )
    {
        Message.Text = "Inline Kod Yazdım";
    }
</script>
</head>
<body>
    <form runat="server">
        <asp:Label id="Message" runat="server" />
    </form>
</body>
</html>
```

Code-Behind Kod Yazmak

- Code-behind Kod Yazmak**
- Programlama ve tasarım sayfaları ayrı tutularak çalışma mantığına göre ayrılmıştır.



Visual Studio .NET ortamının kullandığı varsayılan model code-behind tasarım modelidir. Programlama ve tasarım sayfaları ayrı tutularak çalışma mantığına göre ayrılmış olur. Code-behind sayfaları, .aspx uzantılı sayfanın sonuna .cs eklenerek isimlendirilir. **webform1.aspx** sayfasının code-behind sayfası, **WebForm1.aspx.cs** şeklindedir.

Form üzerinde bir kontrole çift tıklandığında code-behind sayfası açılır ve o kontrole ait olay için metod tanımlanır.

Code-behind sayfasının .aspx sayfasıyla birlikte çalışabilmesi için, .aspx sayfasının **Page** bildiriminde, **CodeBehind** ve **Src** özelliklerine ilgili değerlerin girilmesi gereklidir.

Kod 8.2: WebForm.aspx - Code-behind kod yazmak

```
<%@ Page Language="c#" AutoEventWireup="false"
  Codebehind="WebForm1.aspx.cs"
  Inherits="ilkAspNet.WebForm1"%>
<HTML>
  <HEAD>
    <title>BilgeAdam</title>
  </HEAD>
  <body MS_POSITIONING="GridLayout">
    <form id="Form1" method="post" runat="server">
      <asp:TextBox id="txtAd" runat="server"/>
      <asp:Button id="btnGonder" runat="server"
        Text="Gönder"/></asp:Button>
      <asp:Label id="lblMesaj" runat="server"></asp:Label>
```

```
</form>
</body>
</HTML>
```

Kod 8.3: WebForm.aspx.cs – Code-Behind Sayfası

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
public class WebForm1 : System.Web.UI.Page
{
    protected System.web.UI.WebControls.TextBox txtAd;
    protected System.web.UI.WebControls.Label lblMesaj;
    protected System.web.UI.WebControls.Button btnGonder;

    private void Page_Load(object sender, System.EventArgs e)
    {
        Response.Write("Selam");
    }
    private void Button1_Click(object sender, System.EventArgs e)
    {
        lblMesaj.Text = "Hoşgeldin " +
txtAd.Text.ToString();
    }
}
```

Codebehind

Code-behind sayfasının ismini temsil eder. Visual Studio .NET platformunun dosyayı birleştirebilmesi için gereklidir.

Src

Code-behind sayfasının ön derleme işleminden geçmediği durumlarda bu özellik kullanılır. Code-behind sayfasının ismini temsil eder.

Konu 2: Client Side(İstemci Taraflı) Olay Prosedürleri

- ### İstemci Taraflı Olay Prosedürleri
- İstemci taraflı olay prosedürleri, web forma istekte bulunan kullanıcının, bilgisayarı üzerinde işlenen olaylardır.
 - İstemci taraflı olay prosedürleri hiçbir zaman sunucu kaynağını kullanmazlar.

Client-side olay prosedürleri, web forma istekte bulunan kullanıcı bilgisayarı üzerinde işlenen olaylardır. Kullanıcı tarafından oluşan bu olaylar, sunucuya hiçbir bilgi göndermezler. Çünkü istemci tarafındaki Internet tarayıcısı kodu alır ve işler.

Client-side olay prosedürleri yalnızca HTML kontrolleri tarafından kullanılır. Client-side olay prosedürleri hiçbir zaman sunucu kaynağı kullanmaz. Örneğin SQL Server veritabanına erişmek için client-side kod kullanılamaz.

Client-side olay prosedürlerini, istemci tarafında kısa zamanda oluşması istenilen olaylar için kullanılır. Örneğin, metin kutusuna girilen değerin doğruluğunu kontrol etmek için client-side kod kullanılabilir.

Client-side kodlar **<Script>** blokları içerisinde tanımlanır. Örnekte JavaScript ile istemci taraflı kod tanımlanmaktadır.

```
<SCRIPT language="JavaScript">  
</SCRIPT>
```

Konu 3: Server Side(Sunucu Taraflı) Olay Prosedürleri

Sunucu Taraflı Olay Prosedürleri

- Sunucu üzerinde çalışan olay prosedürleridir.
- Web ve HTML server kontrolleri tarafından oluşturulan olayların, işlenmesinde kullanılır.
- Sunucu kaynaklarını kullanırlar.

Server-side olay prosedürleri, sunucu üzerinde çalışan olaylardır. Server-side olay prosedürleri client-side olay prosedürlerinden oldukça güçlüdür.

Server-side olay prosedürleri, web sunucusu üzerinde bulunan derlenmiş kodlardan oluşur. Web ve HTML server kontrolleri tarafından oluşturulan olayların, işlenmesinde kullanılır. Client-side olay prosedürleri sunucu kaynaklarını kullanamazken, server-side olay prosedürleri sunucu kaynaklarını kullanır.

Server-side olay prosedürleri için aşağıdaki tanımlama yapılır.

```
<SCRIPT language="c#" runat="server">
```

Client-side olay prosedürleri ile fare ve klavye olaylarına kod yazılabilir. Server-side olay prosedürleri **Click** ve **Change** gibi olaylar için kullanılır. Fare ve klavye olayları gibi çok sık gerçekleşebilecek olayları desteklenmez.

Olay Prosedürleri Oluşturmak

Olay Prosedürleri Oluşturmak

- **WithEvents**
 - Server olay prosedürlerinin çalışmasını sağlamak için kontrollerin **WithEvents** anahtar sözcüğü ile tanımlanması gereklidir.
 - **Olay prosedürlerine iki parametre girilmelidir,**
 - **Sender** : Olayı tetikleyen kontrol nesnesidir.
 - **EventArgs** : Olaya özgü parametreleri içeren nesnedir.
 - **Olay Prosedürlerinde Kontrollerle Etkileşim**

Visual Studio .NET içerisinde server-side olay prosedürleri iki adım ile oluşturulur. Birinci adım olay üretecek kontrolu web form üzerine eklemektir. İkinci adım ise code-behind sayfasına olay prosedürü eklemektir.

Olay prosedürleri kontrolün **ID** özelliğine girilen değerden faydalananarak oluşturulurlar. **Handles** anahtar sözcüğü kontrolün hangi olay ile ilişkilendirileceğini belirler. **Handles** anahtar sözcüğü ile tek bir olay için birden fazla olay prosedürü oluşturulabilir.

Örnekte Web form içerisinde btn1 isminde bir button yerleştirilmiştir. Eklenen btn1 kontrolünün, üretilen HTML kodu ve code-behind sayfasına eklenen olay prosedürü kod 8.4 belirtilmiştir.

Kod 8.4: Button Kontrolüne Click Olayı ile İlişkilendirmek

```
<asp:Button id="btn1" runat="server"/>  
'...  
  
protected System.Web.UI.WebControls.Button btn1  
private void btn1_Click(object sender, System.EventArgs e)  
{  
//...  
}
```

Server olay prosedürlerinin çalışmasını sağlamak için kontrollerin **WithEvents** anahtar sözcüğü ile tanımlanması gereklidir.

Olay prosedürlerine iki parametre girilmelidir.

- **Sender**: Olayı tetikleyen kontrol nesnesidir.
- **EventArgs**: Olaya özgü parametreleri içeren nesnedir.

Olay Prosedürlerinde Kontrollerle Etkileşim

Web uygulamalarında, genellikle kontrollerden veri alma ve kontrollere veri gönderme işlemlerine ihtiyaç duyulur. Server-side olay prosedürleri bu tür zor işlemlerin kolayca yapılmasını sağlar.

Örnekte **txtAd** isimli metin kutusuna girilen değer **lblMesaj** isimli etikete yazdırılmaktadır.

Kod 8.5: Olay prosedürleriyle Çalışmak

```
<asp:TextBox id="txtAd" runat="server" />
<asp:Button id="btn1" runat="server"/>
<asp:Label id="lblMesaj" runat="server" />

private void Page_Load(System.Object sender,
System.EventArgs e)
{
    Button1.Click +=new EventHandler(btn1_Click);
}

private void btn1_Click (ByVal sender As System.Object, _
ByVal e As System.EventArgs)
{
    string mesaj = "Merhaba " + txtAd.Text.ToString();
    lblMesaj.Text = Mesaj;
}
```

Konu 4: Sayfa Yaşam Döngüsü

Sayfa Yaşam Döngüsü

- **Page_Init**
 - Sayfa başlatılır ve kontroller oluşturulur.
- **Page_Load**
 - Sayfa yükleniği zaman tetiklenir.
- **Control Events**
 - Kontrollere ait olaylar.
- **Page_Unload**
 - Sayfa kapandığı zaman tetiklenir.

Bir ASP.NET sayfası belirli olaylar sırası ile çalışır. Bu sıraya “Sayfa Yaşam Döngüsü” denir

Bu döngü olayları aşağıdaki sırada gerçekleşir.

- 1- **Page_Init (Page Initialize – Sayfanın oluşmaya başlaması):** Bu aşamada sayfa başlatılır ve sayfadaki kontroller oluşturulur.
- 2- **Page_Load (Sayfanın yüklenmesi):** Bu olay, sayfa yükleniği zaman tetiklenir.
- 3- **Control Events (Kullanıcı kontrol olayları):** **Click** ve **Change** olaylarıdır. Bu olaylar kontrollerin tıklanması veya kontrol değerlerinin değişimi ile tetiklenir. **TxtAd_Changed** ve **Btn1_Click** gibi.
- 4- **Page_Unload (Sayfanın Kapanması):** Bu olay, sayfa kapandığı zaman tetiklenir.

Page Event yaşam döngüsü sonlandığında sayfaya ait bilgiler hafızadan silinir.

Kontrol olaylarının çoğu, sayfa sunucuya geri gönderilene kadar gerçekleşmez. Örneğin **Click** olayı ile form sunucuya gönderilmeden **Change** olayları gerçekleşmez.

Response.Redirect

Response.Redirect

- Kullanıcıyı bir sayfadan başka bir sayfaya yönlendirmek için kullanılır.
- Parametre olarak gidilecek sayfanın adresini belirten String türünden bir değişken alır.

Kullanıcıyı bir sayfadan başka bir sayfaya yönlendirmek için kullanılır. **HyperLink** kontrolü gibi sayfalar arasında dolaşmayı sağlar. Parametre olarak gidilecek sayfanın adresini belirtilir.

Örnekte **Response.Redirect** metodu ile yönlendirme işlemi gerçekleştirilmektedir.

Kod 8.6: Response.Redirect

```
private void ListBox1_SelectedIndexChanged(object sender,  
System.EventArgs e)  
{  
    Response.Redirect(ListBox1.SelectedValue);  
  
<asp:ListBox id="ListBox1" runat="server"  
    AutoPostBack="True">  
    <asp:ListItem  
        Value="http://www.yahoo.com">yahoo  
    </asp:ListItem>  
    <asp:ListItem  
        Value="http://www.google.com">google  
    </asp:ListItem>  
</asp:ListBox>
```

Postback İşlemleri

- ### Postback İşlemleri
- Sunucuya veri gönderip, veri alma işlemine “postback” denir.
 - Page.IsPostBack
 - Page.IsPostBack özelliği ile, sayfanın sadece ilk defa çalıştırılmasında yapılması istenen işlemler yazılır .

Sunucuya veri gönderme işlemine “postback” denir. Örneğin **Button** kontrolü tıklanınca otomatik olarak sunucuya veri yollanır.

Varsayılan olarak veri yollayan tek kontrol **Button** kontrolüdür. Diğer kontroller için **AutoPostBack** özelliğinin **True** yapılması gereklidir.

Örnekte **DropDownList** kontrolünün **AutoPostBack** özelliğine **True** değeri girilmiştir. Bu durum liste kutusundan seçilen değerin sunucuya gönderilmesini sağlar.

Kod 8.7 DropDownList Kontrolunun Sunucuya Gönderilmesi

```
<asp:DropDownList id="DropDownList1"
    runat="server" AutoPostBack="True">
    <asp:ListItem>Türkçe </asp:ListItem>
    <asp:ListItem>İngilizce</asp:ListItem>
</asp:DropDownList>

<asp:TextBox id="mesaj" runat="server"/>

private void DropDownList1_SelectedIndexChanged(object
sender, System.EventArgs e)
{
    mesaj.Text = DropDownList1.SelectedItem.Value;
}
```

Page.IsPostBack

Page Load olayı, sayfa yükleniği zaman gerçekleşir. Sayfaya yapılan her istekte **Page Load** olayı içindeki kodlar çalışır. Aynı sayfanın her seferinde yeniden çalıştırılması, istenilen bir durum değildir.

Page.IsPostBack özelliği ile, sayfanın **Load** olayı içindeki kodlar sadece bir kez çalıştırılır. Böylece sayfa yeniden çağrıldığında bu işlemler gerçekleşmez.

Örnekte **Page.IsPostBack** özelliğinin kullanımı gösterilmektedir.

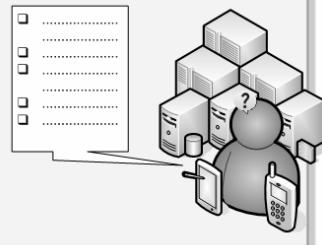
Kod 8.8: Page.IsPostBack

```
private void Page_Load(System.Object sender,
System.EventArgs e)
{
    //Put user code to initialize the page here
    if(!Page.IsPostBack)
    {
        //sadece sayfanın ilk yüklenliğinde çalışan istenilen
        alan
    }
    //sayfa her yüklenliğinde çalışan alan
}
```

Modül Özeti

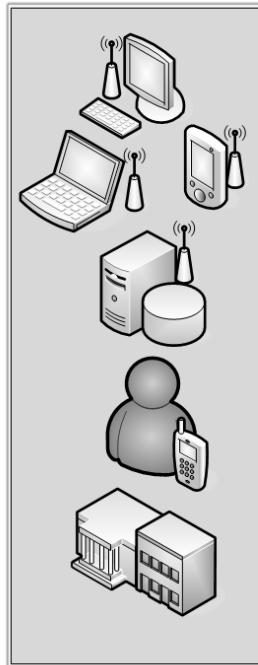
Modül Özeti

- Web form içerisinde kod yazmak için hangi yöntemler kullanılır?
- Code-Behind kod yazma tekniğini açıklayın?
- Sayfa yaşam döngüsünü açıklayın?
- Response.Redirect niçin kullanılır?
- Page.IsPostBack niçin kullanılır?



21. Web form içerisinde kod yazmak için hangi yöntemler kullanılır?
22. Code-Behind kod yazma tekniğini açıklayın?
23. Sayfa yaşam döngüsünü açıklayın?
24. Response.Redirect niçin kullanılır?
25. Page.IsPostBack niçin kullanılır?

Lab 1: ASP.Net ile Kod Geliştirmek



Uygulamalar ASP.NET İle Kod Geliştirmek

- ◆ ASP.Net ile Kod Geliştirmek

Bu uygulamada, Code Behind ve Inline kod yöntemlerinin kullanımını öğreneceksiniz. Ayrıca istemci taraflı skriptlerin yazılımını öğreneceksiniz.

Bu lab tamamlandıktan sonra:

- Code Behind kod yazma yöntemini öğrenecek,
- Inline kod yazma yöntemini öğrenecek,
- İstemci taraflı skriptlerin kullanımını öğreneceksiniz.

Web uygulaması oluşturmak

Bu uygulamada kullanılacak **ASP .Net Web Application** projesini oluşturun.

14. **File** menüsü altında **New** alt menüsünü işaret edin ve **Project** komutunu tıklayın.
15. **New Project** iletisi kutusundan “ASP.Net Web Application” şablonunu seçin.
16. **Location** metin kutusuna “<http://localhost/AspCode>” yazın.

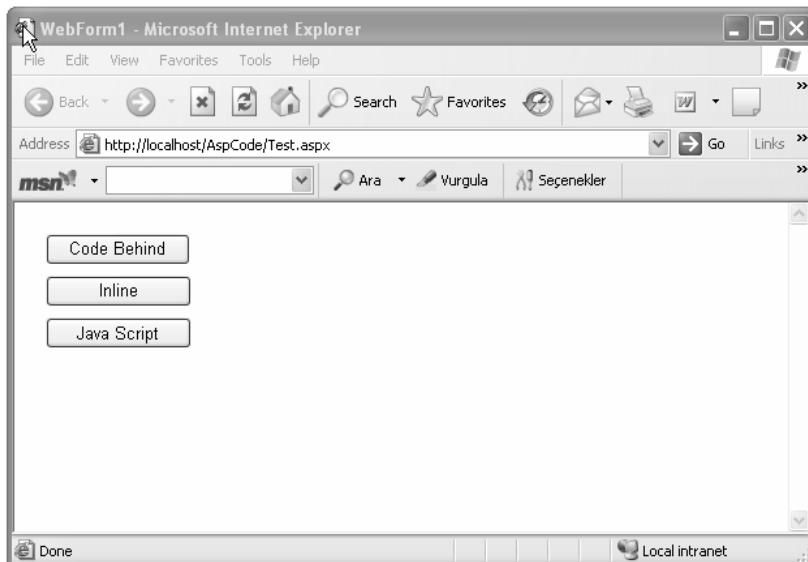
17. **Enter** butonu tıklayın.

Web Form eklenmesi

AspCode projesine Test isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
Button – btnCodeBehind	Text	Code Behind
Button – btnInline	Text	Inline
<INPUT type="button">	Value	Java script



Kodların yazılması

- 1- btnCodeBehind kontrolünün Click olayına "Code Behind" yazan kodu yazın. Bu kod code behind yöntemi ile btnCodeBehind kontrolünün **click** olayını çalıştırır.

```
Response.Write("Code Behind")
```

- 2- Test web formunun HTML bölümüne aşağıdaki kodları yazın. Bu kod, inline kod yöntemi ile btnInline butonunun **click** olayını çalıştırır. Bu kodu **<Body> .. </Body>** etiketleri arasına ekleyin.

```
<script language="c#" runat="server">
    private void btn_Click(object sender, System.EventArgs e)
{
    Response.Write("Inline");
}
```

```
}
```

- 3- Test web formunun HTML bölümüne aşağıdaki kodları yazın. Bu kod, Java Script ile istemci taraflı kod tanımlamaktadır. . Bu kodu `<Head>..</Head>` etiketleri arasına ekleyin.

```
<script language="javascript">
    function mesaj()
    {
        alert('Hello World')
    }

</script>
```

Fonksiyonu çağrılmak için `<Body>..</Body>` etiketleri arasına aşağıdaki kodu ekleyin.

```
<INPUT style="Z-INDEX: 103; LEFT: 24px; WIDTH: 112px;
POSITION: absolute; TOP: 88px; HEIGHT: 24px"
type="button" value="Java Script" onclick="mesaj()">
```

Web Programlamaya Giriş



Modül 9: Web Programlamaya Giriş

Web Programlamaya Giriş

- ◆ Web Programlamaya Giriş
- ◆ HTML Nedir ?
- ◆ Script Nedir ?
 - JavaScript
 - VBScript
- ◆ CSS Nedir ?

Web uygulamalarını zenginleştiren birçok programlama dili bulunur. Web sayfalarını geliştirmek için HTML dilinin kullanılması gereklidir. Sadece HTML kullanımı statik (sabit) sayfalar geliştirmek için yeterlidir. Ancak içeriği kolayca şekillendirmeyi sağlayan CSS dili, istemci tarafından çalışan kodların yazılması için Javascript – VBScript dilleri de Web uygulamalarını zenginleştirir.

Bu modül tamamlandıktan sonra:

- HTML nesnelerini öğrenecek,
- Javascript ve VBScript dilleri ile istemci taraflı kod yazabilecek,
- CSS ile sayfalara özel stiller kazandıracaksınız.

Konu 1 : Web Programlamaya Giriş

İstemci ve sunucu tarafında çalışan çeşitli teknolojiler mevcuttur. Bir Web sayfası tasarlamak için kullanılan ortak dil HTML dilidir. Ancak HTML sayfaları içinde, sunucu tarafında ve kullanıcı tarafında çalışabilen ayrı web programlama dilleri kullanılabilir. Örneğin JavaScript, kullanıcı tarafında kodlama imkanı sunarken, CGI, ASP, PHP, Perl, ASP.NET gibi web programlama dilleri, sunucu tarafında kod yazma imkanı sunar.

Konu 2: HTML

Web Programlamaya Giriş

- ◆ Web Programlamaya Giriş
- ◆ HTML Nedir ?
- ◆ Script Nedir ?
 - JavaScript
 - VBScript
- ◆ CSS Nedir ?

HTML (Hyper Text Markup Language), bir işaretleme dilidir. İstemci tarafında çalışan Internet tarayıcısı tarafından okunur, yorumlanır ve alınan işaretler neticesinde ekrana ilgili görüntü yansıtılır.

HTML, HTTP (HyperText Transfer Protocol) ile bir arada çalışır. Bu işaretleme dili ile oluşturulmuş dökümanlar yalnızca istemci tarafında çalışır.

HTML dökümanlarının tümü ASCII karakterlerinden oluşur ve herhangi bir metin editöründe yazarak oluşturulabilir.

HTML Yapısı

HTML dilini oluşturan bileşenler

- **Tag** (etiket)
- **Attribute** (nitelik)
- **value** (değer) olarak tanımlanır.

HTML kodlarının temeli etiketlerdir. Bir etiket, HTML'e yapılması istenen bir olayın bildirimini temsil eder. Etiketlerin içerisinde **attribute** adı verilen değerler girilir. Etiketlere ait değişik özellikler, **attribute** nesnelerinde saklanır. **Attribute** değerleri, olaylara ait ayrıntıları tutar. **value** ise, bir **attribute** değerinin davranışacağı tarzı belirler.

Tag

Tag (Etiket)

- Etiketler, <> işaretleri arasında yazılır.
Semboller ve etiket arasında boşluk yoktur.
- Açma ifadesi <Html>
- Kapatma ifadesi </Html>
- HTML etiketlerinde büyük harf, küçük harf duyarlılığı yoktur

Etiketler (Tag), <> işaretleri arasında yazılır. Semboller ve etiket arasında boşluk yoktur.

<Html>

Bir etikete ait açma ve kapatma ifadeleri vardır. Açma ifadesi <Html>, kapatma ifadesi ise </Html> şeklinde bildirilir. Yani açılan bir etiket, aynı isim başına / işaretini getirilerek kapatılır. Örneğin <P> etiketi, bir paragraf açmaya yarar. </P>

Kapatma ifadesi kullanıldığında ise bu kapatma ifadesinin kullanıldığı yere kadar olan bölüm paragrafin , içine dahil edilir. Ancak bazı istisnalarda vardır. **** ve **
** etiketi gibi bazı etiketler için bir kapatma ifadesi yoktur.

HTML etiketlerinde büyük harf, küçük harf duyarlılığı yoktur.

Attribute

Attribute (Nitelik)

- Attribute, bir etikete ait özellikleri belirler.
- Arka plan rengi
 - Bgcolor
- Yazı tipi
 - Text

Öznitelikler (Attribute), bir etikete ait özellikleri belirler. **<html>** etiketi ve **</html>** kapatma etiketi, HTML belgesinin başladığı ve bittiği yeri belirtir ve **Attribute** değerine ihtiyacı yoktur. **Attribute** olmadan çalışan etiketlerin yanı sıra, **attribute** ile birlikte çalışan etiketler de vardır. **<body>** etiketi, **</body>** kapatma ifadesi ile görüntülenecek alanın başlangıcını ve bitişini bildirir. Bu etiket, hiçbir **attribute** tanımlanmadan çalışabilir. Görüntülenecek alana bir arka plan rengi tanımlamak istediğinizde **bgcolor** veya görüntülenecek yazı tipini tanımlamak istediğinizde **text** özniteliklerine değerler atanmalıdır.

Aşağıdaki örnekte siyah zemin üzerine beyaz yazı yazılmaktadır.

```
<Body bgcolor="#000000" text="#ffffff">
```

Value

Value (Değer)

- Etiketin değer bilgileri value ile bildirilir.
- Align Etiketi için,
 - Left,
 - Right,
 - Center,
 - Justify

Bir olayın ne şekilde gerçekleşeceği **value** ile bildirilir. Örneğin bir paragraflik metinin hizalanması, **<DIV>** etiketi kullanılarak yapılır. **DIV** etiketinin **align attribute** değeri, **left**, **right**, **center** ve **justify** değerlerinden birini alır.

HTML Belgesi Nasıl Oluşturulur?

HTML Belgesi Nasıl Oluşturulur?

- <html>... </html>
 - Internet tarayıcısına HTML belgesinin başladığı ve bittiği alanı bildirir.
- <head>... </head>
 - HTML belgesi hakkında tarayıcıya bilgi verilir.
<title> ve <meta> etiketleri burada yer alır.
- <title>... </title>
 - Belgenin başlık yazısını tutar.
- <body>... </body>
 - Belgenin gövde bölümüdür. Sayfada görüntülenmesi istenen tüm değerler burada bildirilir.

HTML belgeleri **tag**, **attribute** ve **value** bileşenleri kullanılarak oluşturulur. Bir HTML belgesi en basit şekilde kod 9.1'de gösterilmiştir.

Kod 9.1: Temel HTML Belgesi

```
<html>
  <head>
    <title> Temel HTML Belgesi </title>
  </head>
  <body>
    Sayfada kullanılacak herşey burada bildirilir.
  </body>
</html>

<html>...</html>, Tarayıcıya HTML belgenin başıldığı ve bittiği alanı bildirir.
<head>...</head>, Tarayıcıya HTML belge hakkında bilgi verir. <title> ve
<meta> etiketleri <head>...</head> etiketleri arasında yer alır.
<title>...</title>, belgenin başlık yazısını tutar.
<body>...</body>, belgenin gövde bölümüdür. Sayfada görüntülenmesi istenen
tüm değerler burada bildirilir.
```

En Sık Kullanılan Etiketler

En Sık Kullanılan Etiketler

- Başlıklar
- Paragraf ve Satır Sonu
- Sayfa Ortalama
- Sayfalara Bağlantı Vermek
- Listeler
- Resim Görüntüleme
- Tablolar
- Formlar

Başlıklar

Bir sayfadaki yazının başlıklarını standart şekilde tutmak için HTML başlıkları kullanılır. Başlıklar 1 ve 6 arası değerler alır.

Kod 9.2: HTML Başlık Etiketleri

```
<html>
<head>
    <title>Başlık Etiketleri</title>
</head>

<body>
    <h1>html başlıklar</h1>
    <h2>Bu başlık H1 tag'i ile oluştu</h1>
    <h3>Bu başlık H2 tag'i ile oluştu</h2>
    <h4>Bu başlık H3 tag'i ile oluştu</h3>
    <h5>Bu başlık H4 tag'i ile oluştu</h4>
    <h6>Bu başlık H5 tag'i ile oluştu</h5>
    <h6>Bu başlık H6 tag'i ile oluştu</h6>
</body>
</html>
```

Paragraf ve Satır Sonu

`<p>... </p>` etiketleri arasında paragraflar tanımlanır. Paragrafta, satır sonunu bildirmek için `
` etiketi kullanılır.

Sayfalara Bağlantı Vermek

Birden fazla sayfayı birbirine bağlamak için sayfa bağlantıları kullanılır.

Bir sayfada başka bir sayfaya bağlantı verebilmek için

- **<a>** etiketi yazılır,
- **<a>** ifadesinden sonra **href attribute** değerine bağlantı verilecek sayfanın adresi girilir,
- **Target attribute** değerine bağlantı sayfasının nasıl bir sayfada görüntüleneceği bilgisi girilir. (Varsayılan olarak **_self** değeri alınır. Yani kendi sayfasında açılır.)
- **>** karakteri ile **<a>** etiketi sonlanır. Bağlantının açılması için tıklanması gereken metin girilir.
- Ve metin bitimine **** etiketi yerleştirilir.

Kod 9.3: Sayfa içinde Bağlantı Vermek

```
<a href="http://www.bilgeadam.com" target="_blank">BilgeAdam  
BTA</a>
```

Listeler

Belge içinde metine liste görünümü vermek için listeleme etiketleri kullanılır. Sırasız ve sıralı listeler oluşturulabilir. Sırasız listeler, **...** etiketleri arasında oluşturulur. Her bir liste nesnesi için **** etiketi kullanılır.

Sıralı listeler, **...** etiketleri arasında oluşturulur. Her bir liste nesnesi için **** etiketi kullanılır.

Kod 9.4: Liste Oluşturmak

```
<h4> Sıralı Liste </h4>  
<ol>  
    <li>nesne 1  
    <li>nesne 2  
</ol>  
  
<h4>Sırasız Liste</h4>  
<ul>  
    <li>nesne 1  
    <li>nesne 2  
</ul>
```

Sıralı ve sırasız listelerin dışında programcı tarafından tanımlı listeler oluşturulabilir. **<dl>...</dl>** etiketleri arasında listelenen metinler girilir. Bu etiketler arasına, başlığı tutan **<dt>** etiketi ve başlık altında görüntülenecek metini tutan **<dd>** etiketi yerleştirilir.

Kod 9.5: Tanımlı Liste Oluşturmak

```
<dl>
  <dt> Başlık 1:
    <dd> Başlık 1'e ait açıklama bu paragrafta girilir. Başlık
    1'e ait açıklama bu paragrafta girilir.
    <dt> Başlık 2:
      <dd> Başlık 2'ye ait açıklama bu paragrafta girilir.
      Başlık 2'ye ait açıklama bu paragrafta girilir.
</dl>
```

NOT: Listeleme ifadeleri iç içe kullanılabilir.

Resim Görüntüleme

HTML sayfalarında resim görüntülemek için **** etiketi kullanılır. **src attribute** değeri, görüntülenecek resmin adresini tutar.

Kod 9.6'da **** kullanımı gösterilmektedir.

Kod 9.6: Resim Görüntülemek

```
<a href="http://www.bilgeadam.com">
  
</a>
```

img etiketinin sonlandırma ifadesi yoktur.

Tablolar

Satır ve sütünlardan oluşan yapılara tablo denir. Bir Web sayfası içerisinde, görünümü belirli sınırlarda tutmak için tablolardan yararlanılır.

<table>...</table> etiketleri ile tablonun başlangıç ve bitiş alanı bildirilir. Bu etiketler arasındaki, **<tr>** satırları, **<td>** ise sütunları temsil eder. Tablonun **align**, **border**, **width**, **height** **bgcolor attribute** değerleri ile tabloya çeşitli nitelikler verilebilir. **Align** attribute nesnesi **center**, **left** veya **right** hizalama değerini alır. **Border** tablo kenarlıklarının kalınlık değerini tutar.

Kod 9.7: 4x3 Boyutlarında Tablo oluşturmak

```
<table width="140" border="2" bgcolor="#6633CC"
align="left">
  <tr>
    <td>1. satır 1. sütun</td>
    <td>1. satır 2. sütun </td>
    <td>1. satır 3. sütun </td>
  </tr>
  <tr>
    <td>2. satır 1. sütun </td>
    <td>2. satır 2. sütun </td>
    <td>2. satır 3. sütun </td>
  </tr>
  <tr>
```

```
<td>3. satır 1. sütun </td>
<td>3. satır 2. sütun </td>
<td>3. satır 3. sütun </td>
</tr>
<tr>
<td>4. satır 1. sütun </td>
<td>4. satır 2. sütun </td>
<td>4. satır 3. sütun </td>
</tr>
</table>
```

Konu 3: Script Nedir?

Script Nedir?

- **JavaScript**
 - Netscape firması tarafından geliştirilmiştir.
 - C dili esas alınarak tasarlanmıştır.
 - JavaScript, istemci tarafında çalışır.
- **VbScript**
 - Microsoft firması tarafından geliştirilmiştir.
 - Microsoft firması tarafından desteklenmez.
 - Koşul ve döngü yapılarının kullanımı Visual Basic .NET dilindeki gibidir.

HTML dosyası içine gömülü kodlara script denir. Yorumlanması için internet tarayıcısına ihtiyaç vardır. HTML dilinin karşılayamadığı bazı ihtiyaçlara çözüm üretmek için scriptler kullanılır.

JavaScript

JavaScript

- JavaScript
 - Değişkenler
 - Operatörler
 - Klavyeden Bilgi Alma ve Ekrana Çıktı Verme
 - Koşul ve Döngü Yapıları
 - Fonksiyonlar
 - JavaScript Nesneleri
 - Olaylar

JavaScript dili, Netscape firması tarafından oluşturulmuştur. Yazım biçimini olarak C dili esas alınarak tasarlanmıştır. Amaç olarak HTML'in yetmediği yerlere script'ler ile destek vermesi düşünülmüştür. Web programcılığına dinamik bir yapı kazandıran JavaScript, istemci tarafında çalışır. Kullanımı giderek yaygınlaşan JavaScript, daha sonra Microsoft firmasının Internet Explorer web tarayıcısında da kullanılabilir hale geldi. Günümüzde tüm tarayıcıların desteklediği bir script dilidir.

JavaScript kodları yazmak için Notepad gibi bir metin editörü yeterlidir. Kodlar **<script>...</script>** etiketleri arasında yazılmalıdır.

Bu kod alanı içinde yorum satırları için // ve /* ... */ ifadeleri kullanılabilir. Sadece bir satır yorum satırı yapılacaksa // ifadesi kullanılır.

// bu satır yorum satırıdır.

Birden fazla satır yorum satırı yapılacaksa, satırların başladığı yere /*, bittiği yere */ ifadeleri yerleştirilir.

```
/* yorumu alınan 1. satır
2. satır
...
*/
```

JavaScript kodları HTML sayfaları içine **<head>** etiketlerine gömülü olarak veya .js uzantılı dosyalara referans gönderilerek HTML içinden çağrılabilir.

JavaScript dilinde nesneler, nesnelere uygulanan olaylar ve olaylara ilişkin görevler vardır. Bir nesneyi tıklamak, üzerine gelmek, üzerinde dolaşmak gibi işlemler, sayfa ile kullanıcının etkileşimli olarak çalışmasını sağlar.

JavaScript, aynı bir programlama dilinde olduğu gibi değişkenlere, klavyeden bilgi alma, ekrana çıktı verme işlemlerine, koşul ve döngü yapılarına, fonksiyon, nesne ve olay kavramlarına sahiptir.

Kod 9.8: Örnek JavaScript

```
<html>
  <head><title>onClick</title>
  <script language="javascript">
    function merhaba()
    {
      alert ("beni tikladiniz");
    }
  </script>
  </head>
  <body>
    <input type="button" name="tikla" value="tikla"
           onClick=merhaba()>
  </body>
</html>
```

script etiketinin **Language** attribute değeri ile kullanılacak script dili belirtilir. Javascript kullanılması için burada **language="javascript"** bildirimi yapılır.

Kod 9.9: Örnek JavaScript

```
<html>
  <head>
    <title>JavaScript Örneği</title>
  </head>
  <body>
    <br> Bu yazı html ile yazıldı.
    <br>
    <script language="JavaScript">
      document.write("İşte bu ise JavaScript ile yazıldı!")
    </script>
    <br>
    Bu yazı yine HTML ile yazıldı.
  </body>
</html>
```

Buradaki **script** ifadesi **head** etiketleri arasında bir fonksiyon olarak değil, **body** etiketleri arasında satır halinde kullanılmıştır.

Değişkenler

var anahtar sözcüğü ile yeni bir değişken oluşturulur. Tür bilgisi saklanmaz. Sayısal değerler verildiğinde işlem yapma yeteneğine sahip olurlar. Çift tırnak içerisinde değer verildiğinde ise metin ifadesi olarak anlaşılır.

Dikkat edilmesi gereken nokta değişkenlerin küçük – büyük harf duyarlı olmasıdır. Bazı tarayıcılar için değişken isimlerinde bu duyarlılık göz önünde bulundurulmazken, çoğu tarayıcıda küçük – büyük harf duyarlılığına dikkat edilir. Bu nedenle her değişken adı bu durum göz önünde bulundurularak verilmelidir.

```
var deger1;
var deger2=20;
var deger3=30;
var ay="Mayıs";
var yıl="2005";

var degerToplam=deger2+deger3;
var tarih=ay+yıl;
```

Satırın sonunda sonlandırma karakteri olarak ; kullanılır. **degerToplam** isimli değişkende 20 ve 30 değerleri toplanarak elde edilen 50 değeri tutulurken, tarih isimli değişkende, ay ve yıl değişkenlerinden gelen metin ifadeleri birleştirilir ve “Mayıs2005” değeri oluşturulur.

Koşul Operatörü

[koşul ifadesi] ? koşul_dogrú_ise : koşul_yanlış_ise

Değişken tanımlarken aritmetik, karşılaştırma ve mantıksal operatörler kullanılabilir. Bunlara ek olarak C dilinden gelen koşul operatörleri kullanılabilir. Bir **if** deyiminin tek satırda yazılmış haline benzeyen bu operatörün kullanımı kod 3.1.1'de gösterilmiştir.

Genel kullanım biçimini ise

[koşul ifadesi] ? koşul_dogrú_ise : koşul_yanlış_ise

şeklindedir.

Kod 9.10: Koşul Operatörünün Kullanımı

```
var a=5;
var b=7;
var c=14;
var d=23;
var e;

e = (a + b < c) ? d : a+b ;
```

Bu kodda, **(a + b < c)** ifadesi ile elde edilen sonuca göre, **e** değerine **d** veya **a + b** değerleri atanır.

a + b işleminin sonucu olan 12 değeri, **c** değerinden küçük olduğu için ifade doğru olarak sonuçlanır. Bu durumda **d** değeri, **e** değişkenine atanır ve **e** değişkeni 23 değerini taşır.

Bu koşulu **if** deyimi ile yazılabilir.

```
if (a + b < c)
    e = d;
else
```

```
e = a+ b;
```

Operatörler

JavaScript operatörleri, Visual Basic .NET dilinde kullanılan operatörlerden biraz farklıdır. Örneğin mod almak için **Mod** anahtar sözcüğü yerine **%** mod alma operatörü kullanılır.

Atama Operatörü (=)

Değişkenlere değer atamak için = karakteri kullanılır.

Aritmetik Operatörler

Değişkenler üzerinde aritmetik işlemler yapmak için tanımlanmış operatörlerdir.

Operatör	Açıklama
+	Sayısal değişkenleri toplar. String değişkenlerini birbirine ekler.
-	Sayısal değişkenlerde çıkarma işlemi yapar.
*	Sayısal değişkenlerde çarpma işlemi yapar.
/	Sayısal değişkenlerde bölme işlemi yapar.
%	Sayısal değişkenlerde mod alma işlemini yapar.
++	Sayısal değişkenlerde artma işlemini yapar.
--	Sayısal değişkenlerde azalma işlemini yapar.

Tablo 9.1: Aritmetik Operatörler

Visual Basic .NET aritmetik operatörlerinden farklı olan **++** ve **--** operatörleri, C dili operatörlerindendir. Değişkeni bir arttırma veya bir azaltma yeteneğine sahiptir. **Prefix** (değişken isminin önünde) ve **suffix** (değişken isminin arkasında) olmak üzere iki kullanım şekli vardır.

Değişkenin prefix kullanımı kod 9.11 de gösterilmektedir.

Kod 9.11: Prefix ++ operatörü

```
var x = 5;
// x değişkeni bir arttırılır ve ekrana 6 değeri yazılır
document.write(++x);
```

Değişkenin subfix kullanımında ise önce değer alınır, akış bir sonraki satıra geçtikten sonra değişkenin değeri bir arttırılır.

Kod 9.12: Subfix ++ operatörü

```
var x = 5;
/* x değişkeni önce yazılır, sonra bir arttırılır.
   Yani ekrana 5 yazılır. */
document.write(x++);
// Ekrana 6 değeri yazılır.
document.write(x);
```

Karşılaştırma Operatörleri

JavaScript kodları içerisinde de karşılaştırma işlemleri yapılabilir. Ancak bu operatörler Visual Basic .NET karşılaştırma operatörlerinden biraz farklıdır.

Operatör	Açıklama
==	Eşit midir? operatörü. İki değer de birbirine eşit ise true sonucu verir.
!=	Eşit değil midir? operatörü. İki değer birbirine eşit değilse true sonucunu verir.
<	Küçük operatörü. Sol taraf değeri, sağ taraf değerinden küçükse true sonucunu verir.
>	Büyütür operatörü. Sol taraf değeri, sağ taraf değerinden büyük ise true sonucunu verir.
<=	Küçük eşittir operatörü.
>=	Büyük eşittir operatörü.

Tablo 9.2: Karşılaştırma Operatörleri

İki değerin eşitliğinin karşılaştırılması için **==** operatörü kullanılır.

```
if (a == b) {
    document.write("a ile b değişkeni eşit")
}
```

İki değerin eşitsizliğinin karşılaştırılması için **!=** operatörü kullanılır.

```
if (a != b) {
    document.write("a ile b değişkeni eşit değildir")
}
```

Mantıksal Operatörler

Mantıksal operatörler ise Visual Basic .NET mantıksal operatörlerinden tamamen farklıdır.

Operatör	Açıklama
&&	And (ve) operatörü. İki tarafta belirtilen ifadeler true ise, sonuç olarak true değerini döndürür.
 	Or (veya) operatörü. İki tarafta verilen ifadelerden en az birinin doğru olması durumunda true değerini döndürür.
!	Not operatörü: Koşulun yanlış olması durumunda true değerini verir.

Tablo 9.3: Mantıksal Operatörler

Visual Basic .NET programlamada **And** operatörünün karşılığı **&&** operatörüdür. **Or** operatörünün karşılığı ise **||** operatörüdür. Bir değerin değili anlamına gelen **Not** operatörünü karşılığı ise **!** operatörüdür.

Klavyeden Bilgi Alma ve Ekrana Çıktı Verme

JavaScript dilinde kullanıcıdan bilgi almak için formların dışında **prompt** komutu kullanılır. **prompt** komutu ile kullanıcıdan bilgi alırken ayrı bir pencere açılır.

```
prompt("soru", "cevap için rehber ifade");
```

Kod 9.13: Prompt ile kullanıcıdan değer almak

```
var sehir;
sehir=prompt("Yaşadığınız şehrin trafik kodunu giriniz",
"İstanbul için 34, Ankara için 6 gibi");
```

JavaScript dilinde HTML sayfasına yazı yazdırma için **write** komutu kullanılır.

```
document.write("Yazılacak istenen değişkene ilişkin
açıklama", degisken);
```

Gördüğü gibi **write** komutu **document** fonksiyonuyla birlikte kullanılır.

Koşul ve Döngü Yapıları

Programmanın akışını yönlendiren koşul yapıları ve döngülerdir. Döngüler birden fazla gerçekleştirilecek işlemlerin blok halinde yazılmasını sağlar.

if koşul ifadesinin genel yapısı

```
if ( koşul )
    // koşul doğru ise çalışacak ifade
    // koşul yanlış ise akışın devam edeceği alan
```

Koşulun doğru olması halinde yapılacak işlemler bir satırdan fazla yer tutuyorsa, bu satırlar **{}** parantezleri ile gruplanır. Visual Basic .NET dilindeki gibi **End if** ifadesi kullanılmaz.

```
if (koşul) {
    //koşul doğru ise
}
else {
    //koşul yanlış ise
}
```

Tekrarlanan belirli bir işlemi yaptmak için kullanılan döngülerin JavaScript dilindeki kullanımı tamamen C dilinin yapısına göre tasarlanmıştır.

```
for döngüsünün genel kullanım biçimi;
for(başlangıç_değeri; döngü_ifadesi; değişcek_değişken_adı)
{
    //yapılacak işlemler
}
şeklindedir.
```

Kod 9.14: For Döngüsünün Kullanımı

```
var a;
var b = 10;
for (a = 1; a <= b; a++) {
    document.write( a , ". sayı", "<br>");
}

while döngüsünün yapısı
while ( döngü_koşul_ifadesi )
{
    //şart doğruysa yapılacak işlemler
}
//şart doğru değilse yapılacak işlemler
```

Visual Basic .NET dilindeki **Select Case** döngüsüne karşılık olarak JavaScript dilinde switch-case ifadesi vardır. Genel kullanımı:

```
switch (parametre)
{
    case "fade1":
        // fade1 koşulu doğru ise yapılması istenenler
        break; //break ile diğer koşulların da çalışması
                //engellenir ve döngüden çıkarılır.
    case "fade2":
        //fade2 koşulu doğru ise yapılması istenenler
        break;
}
```

Fonksiyonlar

JavaScript dilinde, kodların yeniden kullanılabilmesi için kullanılır. Genel kullanımı:

```
function fonksiyon_ismi(parametre1, parametre2)
{
    //yapılacak işlemler
}
```

Fonksiyon içinde hesaplanan değer, **return** ifadesi ile geri döndürülür.

Kod 9.15: JavaScript ile Toplama

```
function topla(deger1, deger2)
{
    var sonuc= deger1+deger2;
    return sonuc;
}
```

Topla fonksiyonuna gönderilen **deger1** ve **deger2** değişkenleri toplanarak, fonksiyon içinde oluşturulan **sonuc** değişkenine atanır. **return sonuc;** ifadesi ile topla fonksiyonunda elde edilen sonuç geri döndürülür.

JavaScript Nesneleri

JavaScript içinde bazı işlemler, bazı nesnelerin fonksiyonları çağrılarak yapılır. Örneğin **document.write** komutu, aslında **document** nesnesinin **write** metodunu çağırır.

Window Nesnesi

Genel pencere özelliklerini tutan nesnedir. Pencere açma ve kapama işlemleri için bu nesne kullanılır.

Genel kullanımı:

```
window.open(" url ","pencere_ismi","pencere_ozellikleri");
window.close();
```

Open komutu ile yeni bir pencere açılırken, **close** komutu ile pencere kapatılır. Yeni bir pencere açmak için **open** komutuna ilk parametrenin girilmesi zorunludur. Pencere_ismi, birden fazla pencere ile işlem yapıldığı durumlarda kullanılabilir. Pencereye ait özellikler tablo 9.4'te belirtilmiştir.

Özellik	Açıklama
Menubar	Menu çubuğunun görüntülenmesini sağlar.
Toolbar	Araç çubuğunun görüntülenmesini sağlar.
Location	Adres çubuğunun görüntülenmesini sağlar.
Status	Durum çubuğunun görüntülenmesini sağlar.

Scrollbars	Kaydırma çubuklarının görüntülenmesini sağlar.
Resizable	Penceresinin boyutlandırılmasını sağlar.
width	Açılan pencerenin pixel genişliğini belirtir.
Height	Açılan pencerenin pixel yüksekliğini belirtir.
Left	Ekranın sol noktasına ile pencere arasındaki uzaklığını verir.
Top	Ekranın üst noktasına ile pencere arasındaki uzaklığını verir.

Tablo 9.4: Pencere Özellikleri

Kod 9.16: Yeni bir pencere açma

```
window.open("http://www.bilgeadam.com", "bilgeadam" ,
"menubar=no, toolbar=no, scrollbars=yes, location=yes,
width=300, height=300");
```

Internet tarayıcısı ile daha önce ziyaret edilmiş sayfalara tekrar ulaşabilmek için **window.history.go(-1)** komutu kullanılabilir. -1 ifadesi ile bir önceki sayfaya gidilir. Sayı artırılarak daha önceki sayfalara da gidilebilir.

Internet tarayıcısının en alt kısmında bulunan **status** penceresine erişmek için **window.status** komutu kullanılır.

```
window.status = "JavaScript öğreniyoruz!";
```

Navigator (tarayıcı) Nesnesi

JavaScript, tarayıcıları da bir nesne olarak değerlendirir. Kullanıcının tarayıcısına ilişkin bilgileri almak için tablo 3.1.5'te belirtilen değişkenler kullanılabilir.

Değişken İsmi	Açıklama
appname	Tarayıcı adı
appversion	Tarayıcı versiyonu
appCodeName	Tarayıcının kod adı
userAgent	Tarayıcının sunucuya kendini tanıtırken verdiği isim.

Tablo 9.5: Navigator (Tarayıcı) Nesnesinin Değişkenleri

Kod 9.17: Tarayıcı nesnesi ile bilgi almak

```
<html>
```

```

<head>
<title>Browser'ımızı tanıyalım</title>
<META content="text/html; CHARSET=iso-8859-9 http-equiv=Content-Type>
<script language="Javascript">
function Tarayici()
{
    var browseradi=" ";
    browseradi += "Browser:" + navigator.appName + "\r" ;
    browseradi += "Surumu:" + navigator.appVersion + "\r" ;
    browseradi += "Kodadi:" + navigator.appCodeName + "\r" ;
    browseradi += "Useragent:" + navigator.userAgent + "\r" ;
    alert(browseradi);
}
</script>
</head>
<body onLoad="Tarayici()"></body>
</html>

```

Olaylar

Bir Web sayfası üzerinde kullanıcının her türlü hareketi kontrol edilebilir. Bir kontrolün üzerine gelmesi, dolaşması ve üzerinden ayrılması gibi hareklere olay denir. Bu olaylar ise **onClick**, **onMouseOver**, **onMouseOut**, **onSubmit**, **onReset**, **onChange**, **onLoad**, **onUnLoad**, **onError**, **onAbort**, **onFocus**, **onBlur** olarak belirtilebilir.

onClick

Internet sitelerinin çoğunda en sık kullanılan JavaScript olayıdır. Sayfa üzerinde bir nesnenin fare ile tıklanıp bırakılması sonucunda gerçekleşen olaydır. **Link**, **button** ve resim nesneleri tıklanarak **onClick** olayı tetiklenebilir. Nesnelerin etiketlerinde ise **onClick** olaylarını tetikleyen fonksiyonların ismi bildirilmelidir.

Kod 9.18: onClick Olayı

```

function tıkla()
{
    alert("Tıklama işlemi gerçekleşti...");
}
<input type="button" name="tıkla" value="tıkla"
onClick=tıkla()>

```

Butona tıklanıp bırakıldığından, **onClick** olayı tetiklenir ve bu olayla ilişkilendirilen **tıkla** fonksiyonu devreye girer. **alert** komutu ile ekrana bir mesaj kutusu çıkar.

Ayrıca **onDoubleClick**, çift tıklanma olayını tetikler.

onMouseOver, onMouseOut

Fare nesnenin üzerindeyken **onMouseOver**, fare nesne üzerinden ayrılnca **onMouseOut** olayları devreye girer.

Kod 9.19: onMouseOver ve onMouseOut Olayı

```
function nesneUzerinde()
{
    window.status="Şu anda nesne üzerindesiniz.";
}
function nesneDisinda()
{
    window.status="nesnenin dışına çıktıınız." ;
}

<a href="http://www.google.com"
    onMouseOver = nesneUzerinde()
    onMouseOut = nesneDisinda()> Google
</a>
```

onSubmit

Web safyalarında ziyaretçinin forma bilgi girip sunucuya göndermesi durumlarında **onSubmit** devreye girer. Gönderilecek forma girilen verilerin uygunluğunun kontrolü bu olayın tetiklediği fonksiyonlara yaptırılabilir.

Kod 9.20: onSubmit Olayı

```
function dogrula()
{
    confirm ('Formu doldurduysanız ok'i tıklayınız');

<form action="mail.pl" method="post" onSubmit="dogrula()">
```

confirm komutu, kullanıcıya **ok** ve **Cancel** butonlarından oluşan bir diyalog penceresi açar.

onReset

Form içinde kullanılan tüm metin alanlarının temizlenmesini sağlar. Doldurulan formda yanlışlık olduğunda bu olay tetiklenir. Kullanıcıya onay penceresi çıkartmak için de kullanılabilir.

Kod 9.21: onReset Olayı

```
function sil()
{
    return confirm('silmek istediginize emin misiniz?');

<form onReset="return sil()">
    <input type="text" name="mail">
    <input type="reset" value="sil">
</form>
```

onChange

Bilgi girişi yapılan alanlarda, değişikliğin gerçekleştiği bilgisi **onChange** olayı ile tetiklenir.

Kod 9.22: onChange Olayı

```
function degisti()
{
    alert('Seçimi değiştirdiniz');

}

<form method="post">
    <p>
        <select name="degistir" size="1" onChange="degisti()">
            <option>İstanbul
            <option>Ankara
            <option>Antalya
        </select>
    </form>
```

Sunulan seçeneklerden herhangi biri seçildiğinde uyarı penceresi çıkacaktır.

onLoad, onUnLoad

onLoad olayı sayfaya giriş yapıldığında gerçekleşir. **onUnLoad** olayı sayfadan çıkışlığında gerçekleşir.

Kod 9.23: onLoad ve onUnLoad Olayı

```
function giris()
{
    alert("Sayfaya Giriş Yaptınız!");
}
function cikis()
{
    alert("Sayfadan çıktınız..");
}

<body onLoad="giris()" onUnload="cikis()">
</body>
```

onError, onAbort

Ziyaret edilen sayfadaki nesneler, çeşitli nedenlerden dolayı tam olarak yüklenememiş olabilir. Genellikle resim nesnelerinin yüklenmesinde problem çıkabilir. Bu tür durumları ziyaretçiye bildirmek için **onError** veya **onAbort** olayları kullanılır.

Kod 9.24: onError ve onAbort Olayı

```

```

onFocus, onBlur

onFocus olayı kullanıcı kontrollerine giriş yapılırken gerçekleşir. **OnBlur** olayı ise ve kullanıcı kontrollerinden çıkış yapılırken gerçekleşir.

Kod 9.25: onFocus ve onBlur Olayı

```
function dogru()
{
```

```

        document.form1.mesaj.value="Lütfen hata yapmayın!";
    }
    function sor()
    {
        document.form1.mesaj.value="isminiz alındı";
    }

<form name="form1" method="post">
    <p><h3>Lütfen isminizi yazınız!</h3></p>
    <input type="text" size="20" name="isim"
           onfocus="dogru()" onblur="sor()">
    <p>
        <input type="text" name="mesaj"></p>
</form>

```

VbScript

VbScript , Microsoft tarafından geliştirilmiştir. Ancak, VbScript tüm tarayıcılar tarafından desteklenmez. Bu nedenle tüm tarayıcılarda çalışacak script yazılmak isteniyorsa, JavaScript kullanılmalıdır.

Vbscript dilinin kullanılabilmesi için **script** etiketi içerisindeki language özniteligiine **vbscript** değeri atanır.

```
<script language="vbscript">...</script>
```

Kod 9.26: İlk VbScript Örneği

```

<html>
    <body>
        <script language="vbscript">
            document.write("İlk VBScript!")
        </script>
    </body>
</html>

```

Değişken tanımlamaları **dim** anahtar sözcüğü ile yapılır.

Kod 9.27: Değişken tanımlaması

```

<html>
    <body>
        <script language="vbscript">
            dim isim
            isim="Bilge Adam"
            document.write(isim)
        </script>
    </body>
</html>

```

Metot tanımlaması Visual Basic.NET diline benzer. Geriye sonuç döndürmeyen metotlar **Sub**, geriye sonuç döndüren metotlar ise **function** anahtar sözcüğü ile tanımlanır. Bu tanımlanmış metotlar “**call MetotAdı()**“ anahtar sözcüğü ile çağrılabılırler.

Kod 9.28: Sub Tanımlaması

```
<html>
  <head>
    <script language="vbscript">
sub mySub()
  msgbox("sub procedure")
end sub
</script>
</head>

<body>
<script language="vbscript">
  call mySub()
</script>
<p> sub procedure geri dönüş değeri taşımaz.</p>
</body>
</html>
```

Kod 9.29: Function Tanımlaması

```
<html>
  <head>
    <script language="vbscript">
      function sehir()
        sehir = "Trabzon"
      end function
    </script>
  </head>

  <body>
    <script language="vbscript">
      document.write("En sevdiğim şehir: " & sehir())
    </script>
    <p> function procedure geri dönüş değerine sahiptir.</p>
  </body>
</html>
```

Koşul ve döngü yapılarının kullanımı Visual Basic .NET dilindeki gibidir.

Konu 4: CSS

CSS

- HTML sayfalarına özel stiller kazandırmak için kullanılır.
- **Inline (iç)**
 - Herhangi bir HTML etiketi içinde.
- **Embedded (gömülü)**
 - <Style></Style> etiketleri arasında.
- **Linked (bağlantılı)**
 - .css uzantılı ayrı bir dosyada

CSS (Cascading Style Sheets), HTML sayfaları içerisinde özel stiller tanımlamak için kullanılır. Sayfanın yazı türü, arka plan rengi, link renkleri, nesnelerin sayfa üzerindeki yerleşimi birer stil öğeleridir. Bir sitedeki tüm sayfalara aynı stili vermek için CSS dosyaları hazırlanır. Bu CSS dosyaları HTML sayfalarına dahil edilir.

Stiller HTML sayfalarına üç yöntem ile dahil edilebilir:

- **Inline (iç)**
- **Embedded (gömülü)**
- **Linked (bağlantılı)**

İç (Inline)

Herhangi bir HTML etiketi içinde stil kullanılabilir. Örneğin paragraf etiketine **style="x"** attribute değeri eklenebilir. Ve o paragrafa özel tasarım özelliklerini bildirilebilir.

```
<p style="font: 12pt arial"> Bu paragraftaki tüm yazılar  
arial 12 stilideler.</p>
```

Gömülü (Embedded)

Gömülü stiller, HTML sayfasının **Head** etiketi içerisinde tanımlanır. Bu stilleri tanımlamak için **<Style>..</Style>** etiketleri kullanılmalıdır.

Kod 9.30: Gömülü stillerin Tanımlanması

```
<html>
  <head>
    <style>
      body
      {
        background:#ff0000;
        color:#ffffff;
      }
    </style>
  </head>
  <body>
    Kırmızı zemin üzerine beyaz renkle yazı
  </body>
</html>
```

Bağlantılı (linked)

Stil verileri .css uzantılı dosyalarda tanımlanır. En güçlü tasarım özelliği bağlantılı stil dosyalarındadır. Tasarım özellikleri bu dosyada tanımlanır ve her sayfa içinden bu stil dosyasına bağlantı verilerek etiketlere gerekli stiller uygulanır.

HTML sayfalarda stil dosyası ile bağlantı kurmak için aşağıdaki tanımlama yapılır. Bu tanımlama head etiketleri içerisinde yapılır.

```
<link rel="stylesheet" href="stil.css" type="text/css">
```

Style Sheet'lerin Söz Dizimi

Slide12

Stil nesnelerinin söz dizimi, HTML söz dizimine benzer yapıdadır. Stil nesnelerinin belirli kısımları vardır:

- **Seçici (Selector):** Atanılan özellikler ve değerleri alır. **H1** ve **P** gibi HTML etiketlerine benzerler.
- **Özellik (Property):** Bir seçiciyi tanımlar. **P** paragraf etiketine verilen özellikler o seçiciyi tanımlar. Kenar boşlukları, font ve arka plan değerleri birer özellik ögesidir.
- **Değer (Value):** Özellikleri tanımlayan öğelerdir.

Özellikler ve değerler birleşerek bir tanım oluşturur. Seçici ve tanım ise bir kural oluşturur. Sayfa düzeni ,kenar boşlukları, girinti ve hizalama değerleri kontrol edilerek hazırlanmış bir site profesyonel bir görünümü sahip olabilir.

En çok kullanılan yazı özellik ve değerleri tablo 4.1'de listelenmiştir.

Özellik ve Değer	Açıklama
Margin-left	Sol kenar boşluğunu belirlemek için kullanılır. Punto, inc, cm ve piksel cinsinden değer verilir. {margin-left: 10px;}
Margin-right	Sağ kenar boşluğunu belirlemek için kullanılır.
Margin-top	Üst kenar boşluklarını belirlemek için kullanılır.
text-indent	Bir yazı için girinti bilgisini belirler.
text-align	Yazının hizalanmasını sağlayan değeri tutar. Left, center, right
text-decoration	underline, overline, line-through, none değerleriyle yazıya şekil verir.
text-transform	Yazının büyük veya küçük harflerle görüntülenmesini sağlar. Uppercase, lowercase

Tablo 9.6: En çok kullanılan yazı özellik ve değerleri

```
Body {
margin-left: 10px;
margin-right: 10px;
margin-top: 20px;
margin-bottom: 15px;
}
```

Font özellik ve değerleri tablo 9.7'de belirtilmiştir.

Özellik ve Değer	Açıklama
font-size	Yazı büyülüüğünü belirler.
color	Yazının rengini tutar.
font-family	Yazının tipini belirler.
font-style	Yazının italikliğini belirler. italic, normal .
font-weight	Yazı kalınlığını belirler. bold, normal

Tablo 9.7: Font özellik ve değerleri

```
p {
font-size: 20;
color: blue;
font-weight: bold;
font-style: italic;
font-family: Times New Roman;
}
```

Liste özellikleri ve değerleri tablo 4.3'te belirtilmiştir.

Özellik ve Değer	Açıklama
list-style-type	Liste elemanlarının başına gelecek karakteri belirler. disc, circle, square, decimal
lower(upper)-roman	Liste elemanlarının başına küçük veya büyük Roma rakamları koyar.
lower(upper)-	Liste elemanlarının başına küçük(büyük) harfler

alpha	koyar.
none	Liste elemanları için bir simbol almaz.
list-style-image	Liste imleri yerine resim kullanır.
list-style-position	indise : Listenin ikinci satırını en soldan başlatır. outside : İkinci satırı bir öncekinin dikey hizasından başlatır.

Tablo 9.8: Liste özelliklerini ve değerleri

Background özellikleri ve değerleri tablo 9.9'de belirtilmiştir.

Özellik ve Değer	Açıklama
background-color	Arka plan renk değerini tutar.
background-image	Arka plan resminin yol bilgisini tutar.
background-repeat	Resmin x ve y koordinatları boyunca tekrarlanması bilgisini tutar. repeat : tüm yönlerde repeat-x : x ekseni boyunca repeat-y : y ekseni boyunca no-repeat : tekrar edilmez
background-position	left : Resmi pencerenin sol kenarına yaklaştırır. right : Sağ kenara yaklaştırır. center : Resmi ortalar.

Tablo 9.9: Background özellik ve değerleri

```
p {
background-color:blue;
background-image: url(back.gif);
background-position:left;
background-repeat:repeat-x;
}
```

Seçiciler

Seçiciler, oluşturulan **<H1>**, **<P>** gibi etiketlerin mevcut özelliklerini aynı tutarak onlara yeni özellikler ekleme olanağı verir. Ayrıca istenilen bir kelimeye stil özelliği atayıp istenilen zamanda çağrılmmasını sağlar.

```
h1 {
background:green;
color:white;
font-weight:bold;
font-family:arial;
}
h1.kirmizi{color:red}
```

Linkler ve CSS

Sayfalarda ziyaret edilen linklerin mavi alt çizgilerini ortadan kaldırmak veya başka stiller vermek için CSS dilinden yararlanılabilir. **<A>** etiketinin stilini belirlemekte kullanılan dört ifade vardır:

- **active**: Tıklanan linkin stilini belirler.
- **link**: Link yazı stilini belirler.
- **visited**: Ziyaret edilmiş linkin stilini belirler.
- **hover**: Fare linkin üzerindeyken nasıl bir stil alacağını belirler.

```
a:link{text-decoration:none; color:teal}
a:active{text-decoration:none; color:red}
a:visited{text-decoration:none; font-family:Times New Roman;
color:green}
a:hover{background-color:teal; color:white; font-
family:arial}
```

Sınıf ve Gruplama

Sınıf (**class**), stil kurallarının küçük parçalara ayırmasını sağlar. Sayfadaki herhangi bir yazının diğer yazılarından farklı görünmesi istediği durumda, istenilen sayıda özel HTML etiketi oluşturulabilir. Örneğin sayfada iki farklı türde **H1** etiketi kullanılmak istensin.

```
H1.serif {
font: 14pt Century Schoolbook;
}
H1.sans{
font: 20pt Arial;
}
```

Bu stilleri kullanmak için kod içine serif veya sans sınıf ismi vermek yeterlidir.

Gruplama, stil özellikleri ve değerleri yoğunlaştırıldığında oluşur. Örneğin:

```
P.1 {
font: verdana;
font-size: 12pt;
line-height: 18pt;
}
```

1 sınıfındaki tüm paragraflar 12 punto **verdana** fontıyla ve 18 punto satır yüksekliğiyle görüntülenir. Sınıf yerine gruplama yapılabilir.

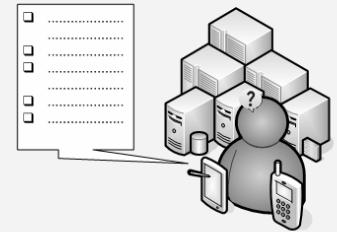
```
P.1{font:12pt/18pt verdana}
```

Her iki gösterimde de aynı görüntü elde edilir. Ancak gruplamada değerler girerken **font-size**, **font-height** ve font sıralamasına uyulması gerektiğine dikkat edilmelidir.

Modül Özeti

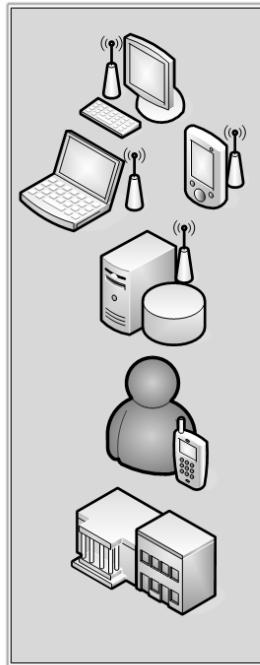
Modül Özeti

- HTML nedir? En sık kullanılan HTML etiketlerini açıklayın.
- Script Nedir?
- Java Script nedir? Java Script niçin kullanılır?
- CSS nedir?
- CSS, Html sayfalara nasıl dahil edilebilir?



26. HTML nedir? En sık kullanılan HTML etiketlerini açıklayın.
27. Script Nedir?
28. Java Script nedir? Java Script niçin kullanılır?
29. CSS nedir?
30. CSS, Html sayfalara nasıl dahil edilebilir?

Lab 1: Web Programlamaya Giriş



Uygulamalar Web Programlamaya Giriş

- ◆ Sanal Klavye Oluşturmak
- ◆ Popup Pencere Oluşturmak

Bu uygulamada, Java Script ile sanal klavye yapmasını öğreneceksiniz. Ayrıca Java Script ile popup pencere oluşturmayı öğreneceksiniz.

Bu lab tamamlandıktan sonra:

- Java Script ile Sanal Klavye oluşturabilecek,
- Java Script ile popup pencere oluşturabileceksiniz.

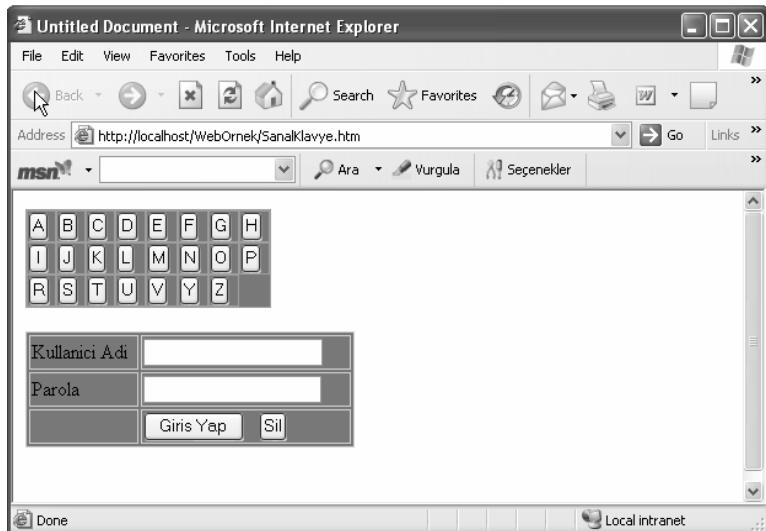
Web uygulaması oluşturmak

Bu uygulamada kullanılacak **ASP .Net Web Application** projesini oluşturun.

18. **File** menüsü altından **New** alt menüsünü işaret edin ve **Project** komutunu tıklayın.
19. **New Project** iletisi kutusundan “ASP.Net Web Application” şablonunu seçin.
20. **Location** metin kutusuna “<http://localhost/> WebOrnek” yazın.
21. **Enter** butonu tıklayın.

Sanal Klavye oluşturmak

WebOrnek projesine SanalKlavye isminde yeni bir Html sayfa ekleyin.



Aşağıdaki kodları <Body>..</Body> etiketlerinin arasına yazarak Html sayfayı tasarlilyn.

```

<div id="klavyem">
    <table width="200" border="1" cellpadding="0"
cellspacing="0" bordercolor="#00ff00" bgcolor="#0099cc"
ID="Table1">
        <tr>
            <td><input type="button" id="ba"
onClick="HarfA()" value="A" NAME="ba"></td>
            <td><input type="button" id="bb"
onClick="HarfB()" value="B" NAME="bb"></td>
            <td><input type="button" id="bc"
onClick="HarfC()" value="C" NAME="bc"></td>
            <td><input type="button" id="bd"
onClick="HarfD()" value="D" NAME="bd"></td>
            <td><input type="button" id="be"
onClick="HarfE()" value="E" NAME="be"></td>
            <td><input type="button" id="bf"
onClick="HarfF()" value="F" NAME="bf"></td>
            <td><input type="button" id="bg"
onClick="HarfG()" value="G" NAME="bg"></td>
            <td><input name="button2" type="button" id="bh"
onClick="HarfH()" value="H"></td>
        </tr>
        <tr>
            <td><input type="button" id="bi"
onClick="HarfI()" value="I" NAME="bi"></td>
            <td><input type="button" id="bj"
onClick="HarfJ()" value="J" NAME="bj"></td>
            <td><input type="button" id="bk"
onClick="HarfK()" value="K" NAME="bk"></td>
            <td><input type="button" id="bl"
onClick="HarfL()" value="L" NAME="bl"></td>
            <td><input type="button" id="bm"
onClick="HarfM()" value="M" NAME="bm"></td>
            <td><input type="button" id="bn"
onClick="HarfN()" value="N" NAME="bn"></td>
            <td><input type="button" id="bo"
onClick="HarfO()" value="O" NAME="bo"></td>
            <td><input type="button" id="bp"
onClick="HarfP()" value="P" NAME="bp"></td>

```

```

</tr>
<tr>
    <td><input type="button" id="br"
    onclick="HarfR()" value="R" NAME="br"></td>
    <td><input type="button" id="bs"
    onclick="HarfS()" value="S" NAME="bs"></td>
    <td><input type="button" id="bt"
    onclick="HarfT()" value="T" NAME="bt"></td>
    <td><input type="button" id="bu"
    onclick="HarfU()" value="U" NAME="bu"></td>
    <td><input type="button" id="bv"
    onclick="HarfV()" value="V" NAME="bv"></td>
    <td><input type="button" id="by"
    onclick="HarfY()" value="Y" NAME="by"></td>
    <td><input type="button" id="bz"
    onclick="HarfZ()" value="Z" NAME="bz"></td>
</tr>
</table>
</div>
<br>
<form name="form1" method="post" ID="Form1">
    <table width="268" border="1" cellpadding="1"
    cellspacing="2" bordercolor="#99ff66" bgcolor="#0099cc"
    ID="Table2">
        <tr>
            <td width="85">Kullanici Adi</td>
            <td width="167"><input name="text" type="text"
id="user"></td>
        </tr>
        <tr>
            <td>Parola</td>
            <td><input type="password" id="password"
readonly NAME="password"></td>
        </tr>
        <tr>
            <td>&nbsp;</td>
            <td><input name="button2" type="button"
id="giris" value="Giris Yap" onclick="hosgeldin()">
&nbsp; <input name="button2" type="reset"
id="temizle" value="Sil"></td>
        </tr>
    </table>
</form>

```

Kodların yazılması

Sanal Klavye Html sayfasının HTML bölümüne aşağıdaki kodları yazın. Bu kod, Java Script ile istemci taraflı fonksiyonlar eklemektedir. Bu kodu `<Head>..</Head>` etiketleri arasına ekleyin.

```

<script language="javascript">

function HarfA()
{
    form1.password.value += "A";
}

function HarfB()
{
    form1.password.value += "B";
}

```

```
function HarfC()
{
    form1.password.value += "C";
}

function HarfD()
{
    form1.password.value += "D";
}
function HarfE()
{
    form1.password.value += "E";
}

function HarfF()
{
    form1.password.value += "F";
}
function HarfG()
{
    form1.password.value += "G";
}

function HarfH()
{
    form1.password.value += "H";
}
function HarfI()
{
    form1.password.value += "I";
}

function HarfJ()
{
    form1.password.value += "J";
}
function HarfK()
{
    form1.password.value += "K";
}

function HarfL()
{
    form1.password.value += "L";
}
function HarfM()
{
    form1.password.value += "M";
}

function HarfN()
{
    form1.password.value += "N";
}
function HarfO()
{
    form1.password.value += "O";
}

function HarfP()
{
```

```
        form1.password.value += "P";
    }
    function HarfR()
    {
        form1.password.value += "R";
    }
    function Harfs()
    {
        form1.password.value += "S";
    }
    function HarfT()
    {
        form1.password.value += "T";
    }

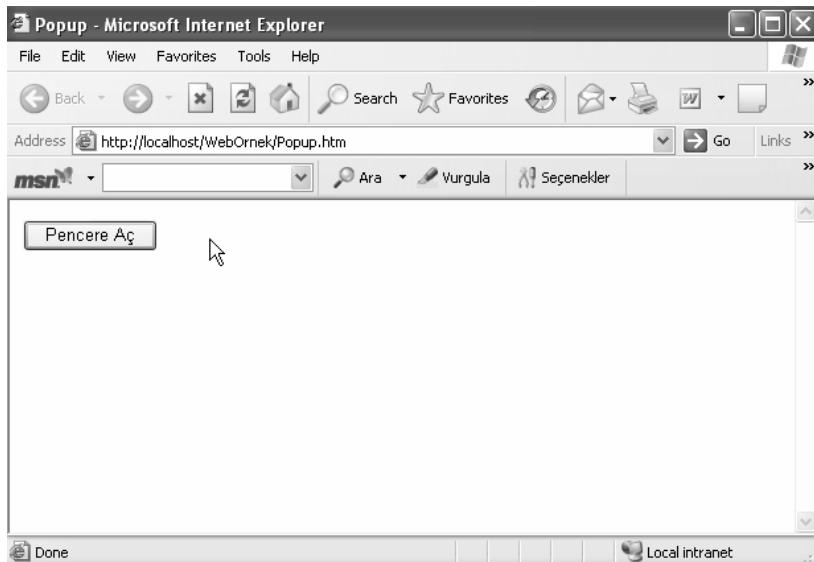
    function Harfu()
    {
        form1.password.value += "U";
    }
    function HarfV()
    {
        form1.password.value += "V";
    }

    function HarfY()
    {
        form1.password.value += "Y";
    }
    function HarfZ()
    {
        form1.password.value += "Z";
    }

    function hosgeldin()
    {
        alert("hosgeldiniz sayin: " + form1.user.value)
    }
</script>
```

Popup pencere oluşturmak

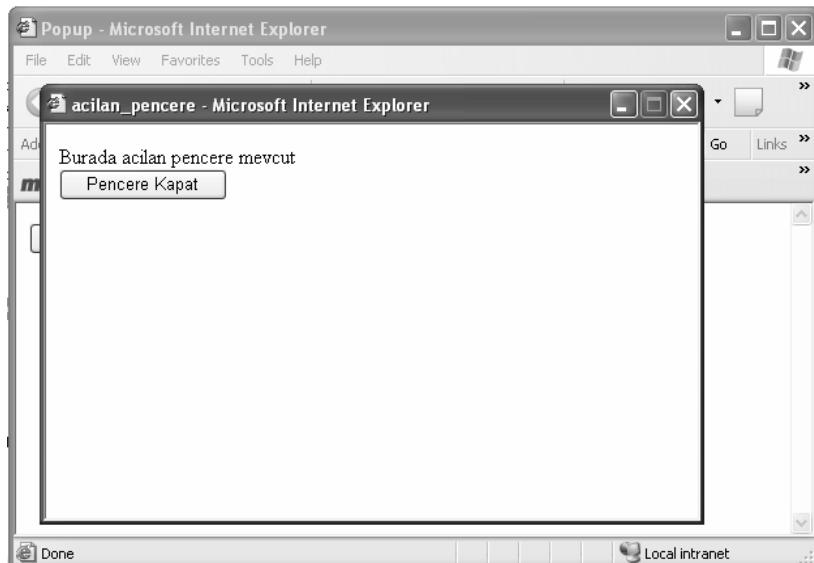
WebOrnek projesine Popup isminde yeni bir Html sayfa ekleyin.



Aşağıdaki kodları <Body>..</Body> etiketlerinin arasına yazarak Html sayfayı tasarlayın.

```
<INPUT id="Button1" type="button" onclick="pencereac()" value="Pencere Açı" name="Button1">
```

WebOrnek projesine acilan_pencere isminde yeni bir Html sayfa ekleyin.



Aşağıdaki kodları <Body>..</Body> etiketlerinin arasına yazarak Html sayfayı tasarlayın.

```
Burada acilan pencere mevcut
<br>
<INPUT id="Button2" onclick="window.close()" type="button" value="Pencere Kapat" name="Button2">
```

Kodların yazılması

Popup, Html sayfasının HTML bölümüne aşağıdaki kodları yazın. Bu kod, Java Script ile istemci taraflı fonksiyon eklemektedir. Bu kodu `<Head>..</Head>` etiketleri arasına ekleyin.

```
<script language="javascript">
    function pencereac()
    {
        window.open('acilan_pencere.htm','acilan','width=500,heig
ht=300')
    }
</script>
```

Kullanıcı Kontrolleri Oluşturmak



Modül 10: Kullanıcı Kontrolleri Oluşturmak

Kullanıcı Kontrolleri Oluşturmak

- ◆ Kullanıcı Kontrolleri
 - ◆ Kullanıcı Kontrolünün Avantajları
 - ◆ Kullanıcı Kontrolünü Projeye Eklemek
 - ◆ Kullanıcı Kontrolü Tasarımı

Web uygulamaları geliştirirken, her sayfada görüntülenecek sabit paneller gerekebilir. Bu panelleri her sayfa için tekrar oluşturmak zaman ve performans

kaybına yol açar. Bu paneller, User Controls (kullanıcı kontrolleri) biçiminde oluşturulup, proje içinde birçok yerde kullanılabilir.

Bu modül tamamlandıktan sonra :

- Kullanıcı kontrollerin yapısını öğrenecek,
- Kullanıcı kontrolü oluşturabilecek,
- Kullanıcı kontrollerini proje içinde kullanabileceksiniz.

Konu 1: Kullanıcı Kontrolleri

Kullanıcı Kontrollerinin Avantajları

- Ayrı bir namespace içinde tanımlanırlar.
- Aynı sayfa içinde birden fazla kullanılabilirler.
- Hiçbir özellik veya metot için isim çakışması söz konusu değildir.
- Kullanıcı kontrolleri web formlardan ayrı dillerde yazılabilir.

Sık kullanılan kontroller bir araya getirilerek yeni bir kontrol oluşturulur. Bu kontroller uygulama içerisinde her sayfada kullanılabilir. Örneğin sayfalar arası dolaşımı sağlayan menü paneli, kullanıcı kontrolü haline getirilebilir.

Web kontrollerinde olduğu gibi kullanıcı kontrolleri de sunucu tarafında çalışır.

Kullanıcı kontrolleri **System.Web.UI.UserControl** sınıfından türetilmiştir.

Kullanıcı Kontrolünün Avantajları

- Kullanıcı kontrolleri, ayrı bir namespace içinde tanımlanır. Bu durum kullanıldıkları Web formları ile oluşabilecek isim çakışmasını ortadan kaldırır.
- Kullanıcı kontrolleri, aynı sayfa içinde birden fazla kullanılabilir. Hiçbir özellik veya metot için isim çakışması söz konusu değildir.

- Kullanıcı kontrolleri ayrı dillerde yazılabilir.

Kullanıcı kontrolleri, uygulama içindeki tüm sayfalara eklenebilir. Ancak diğer uygulamalardaki kullanıcı kontrolleri sayfalara direkt eklenemez. Diğer uygulamalardaki kullanıcı kontrolleri, kullanmadan önce uygulamaya eklenmelidir.

Kullanıcı Kontrolünü Projeye EklemeK

Kullanıcı Kontrollerini Projeye EklemeK

- ✓ Solution Explorer penceresi açılır.
- ✓ Proje adına sağ tıklanır.
- ✓ Add menüsünden Add Web User Control komutu seçilir. Kullanıcı kontrol dosyaları .ascx uzantılıdır.
- ✓ kontrollere ait Visual Basic .NET kodlarının bulunduğu code-behind sayfası .ascx.cs uzantılıdır.

Projeye yeni bir kullanıcı kontrolü eklemek için aşağıdaki adımları takip edin.

1. **Solution Explorer** penceresi açın.
2. Proje adı üzerinde farenin sağ butonunu tıklayın.
3. Açılan penceredeki **Add** menüsünden **Add web User Control** komutunu seçin.
4. **Name** metin kutusuna kullanıcı kontrolune verilecek ismi girin.

Kullanıcı Kontrollerini Projeye Eklemek

- ✓ Web forma eklenen kullanıcı kontrolü, `@Register` ifadesi ile forma bağlanır.
- ✓ `<%@ Register TagPrefix="deneme" TagName="Login" src="login.ascx" %>`

Kullanıcı kontrol formları, normal Web formlar gibi tasarılanır. Kullanıcı kontrol dosyaları .ascx ve bu kontrollere ait code-behind sayfası ise ascx.cs uzantılıdır.

Kullanıcı kontrollerinde HTML ve Visual Basic .NET kodu birlikte kullanılabilir. Ancak kullanıcı kontrolleri web formları tarafından kullanıldığı için `<head>`, `<body>`, `<form>` gibi HTML elementleri bulundurmaz.

Web form direktifi olan `@Page` yerine kullanıcı kontrollerinde `@Control` ifadesi kullanılır. Bu direktif `@Page` direktifinin `AspCompat` ve `Trace` dışındaki tüm `attribute` değerlerine sahiptir.

Kullanıcı Kontrolü Tasarımı

- Solution Explorer panelinden kontrol sürükleenip bırakılır.
- @page direktifinin altına,
`<%@ Register TagPrefix="uc1"
TagName="login" Src="login.ascx" %>` ifadesi eklenir.
- Web form içinde kullanıcı kontrolünü kullanmak için,
`<uc1:login id="Login1"
runat="server"></uc1:login>` kodu yazılır.

Kullanıcı kontrollerini Web form içerisine eklemek için @Register ifadesi kullanılır. Bu ifade kullanıcı kontrollerinin web forma bağlanması sağlar.

```
<%@ Register TagPrefix="deneme" TagName="Login"  
src="Login.ascx" %>
```

TagPrefix attribute değeri kullanıcı kontrolü için bir namespace oluşturur. Böylece her kontrol ayrı bir namespace içinde tanımlanır. **TagName**, kullanıcı kontrolünün ismidir. **Src** ise kullanıcı kontrolünün bulunduğu yolu belirtir.

@Register ifadesi ile forma bağlanan kullanıcı kontrolu, aşağıda kod ile web form içerisinde görüntülenir. Kullanıcı kontrolleri sunucu üzerinde çalıştığı için, **runat="server"** parametresi ile tanımlanmalıdır.

```
<deneme:Login id="Login1" runat="server"/>
```

Kullanıcı Kontrolü Tasarımı

- <uc1> etiketi, Register ifadesinin TagPrefix attribute değerinde belirtilen isimle aynıdır.
- uc1:login , TagName attribute değerinde belirtilen isimle aynıdır.

Örnekte tüm sayfalarda kullanılacak sayfa başlığı, kullanıcı kontrolü olarak tasarlanmıştır.

```
<%@ Control Language="cs" AutoEventwireup="false"
Codebehind="Header.ascx.cs"
Inherits="KullaniciKontrolleri.UserControls.Header" %>

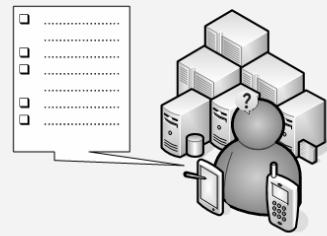

|                                                                                                                          |
|--------------------------------------------------------------------------------------------------------------------------|
| <asp:label font-bold="True" font-size="X-Large" forecolor="white" id="lblHeader" runat="server">Hoşgeldiniz </asp:label> |
|--------------------------------------------------------------------------------------------------------------------------|


```

Modül Özeti

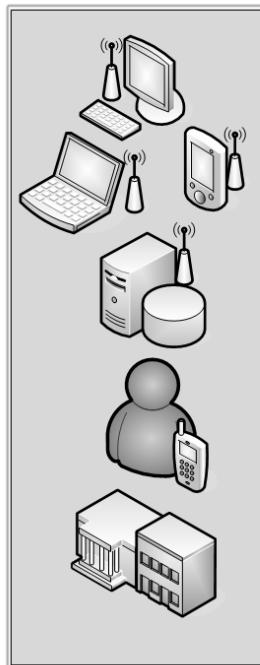
Modül Özeti

- Kullanıcı kontrolleri niçin oluşturulur?
- Kullanıcı kontrollerinin avantajları nelerdir?.
- Web form içerisinde kullanıcı kontrolleri nasıl kullanılır?



31. Kullanıcı kontrolleri niçin oluşturulur?
32. Kullanıcı kontrollerinin avantajları nelerdir?.
33. Web form içerisinde kullanıcı kontrolleri nasıl kullanılır?

Lab 1: E-Ticaret Uygulaması Geliştirmek



Uygulamalar Kullanıcı Kontrolleri Oluşturmak

- ◆ E-Ticaret Uygulaması
Geliştirmek

Bu uygulamada, e-ticaret uygulamasının kullanıcı kontrolleri tasarılanacaktır. Bu uygulamada, bütün sayfalarda kullanılacak üst ve alt menü oluşturulacaktır. Ayrıca kategori isimli kullanıcı kontrolu ile tüm kategoriler listelenecektir. Kategori değerleri veritabanı içerisinde alınır.

Bu lab tamamlandıktan sonra:

- Kullanıcı kontrolu oluşturabileceksiniz.

Kullanıcı kontrollerin eklenmesi

AspEticaret isimli projeyi açın.

Üst kontrolünün eklenmesi

ASPEticaret projesine Üst isminde yeni bir kullanıcı kontrolü ekleyin.

Kullanıcı kontrolu içerisinde tablodaki kontrolleri ekleyin.

Kontrol – Kontrol İsmi	Özellik	Değer
HyperLink – btnAnaSayfa	NavigateUrl	Default.aspx
	Text	AnaSayfa
HyperLink – btnUyeGiris	NavigateUrl	UyeGiris.aspx
	Text	tıklayınız

HyperLink – btnUyeKayit	NavigateUrl	UyeKayit.aspx
	Text	Üye Kayıt
LinkButton – btnCikis	Text	Çıkış
	Visible	False
Label – lblAd		



Üst isimli kullanıcı kontrolünün Html kodları aşağıdaki gibi olacaktır.

```
<%@ Control Language="cs" AutoEventWireup="false"
Codebehind="Ust.ascx.cs" Inherits="AspEticaret.Ust"
TargetSchema="http://schemas.microsoft.com/intellisense/ie5"%>
<TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" border="0">
<TR>
    <TD style="HEIGHT: 17px">
        <P align="center"><FONT face="Lucida Handwriting"
size="6">Yazılım Uzmanı&ampnbspKitapevi</FONT></P>
    </TD>
</TR>
<TR>
    <TD>&nbsp;
        <asp:Label id="lblAd"
runat="server"></asp:Label>&nbsp;</TD>
    </TR>
    <TR>
        <TD>
            <P align="center">
                <asp:HyperLink id="btnAnaSayfa" runat="server"
NavigateUrl="Default.aspx">AnaSayfa</asp:HyperLink>&nbsp;|
                <asp:HyperLink id="btnUyeGiris" runat="server"
NavigateUrl="UyeGiris.aspx">Üye Giriş</asp:HyperLink>&nbsp;|
                <asp:HyperLink id="btnUyeKayit" runat="server"
NavigateUrl="UyeKayit.aspx">Üye
Kayıt</asp:HyperLink>&nbsp;&nbsp;&nbsp;
                &nbsp;
                <asp:LinkButton id="btncikis" runat="server"
visible="False">Çıkış</asp:LinkButton></P>
        </TD>
    </TR>
</TABLE>
```

Üst isimli kullanıcı kontrolünün code behind kodları aşağıdaki gibi olacaktır.

```
Using System.Data.OleDb

private void Page_Load(System.Object sender,
System.EventArgs e)
{
    If (Session("user") != "")
    {
```

```

        lblAd.Text = "Bay / Bayan : " + Session("ad") +
" " + Session("soyad");
        btnCikis.Visible = true;
    else
    {
        btnCikis.Visible = false;
    }
}

private void btnCikis_Click(System.Object sender,
System.EventArgs e)
{
    Session.Abandon();
    btnCikis.Visible = false;
    Response.Redirect("Default.aspx");
}

```

Alt kontrolünün eklenmesi

ASPEticaret projesine Alt isminde yeni bir kullanıcı kontrolü ekleyin.

Kullanıcı kontrolu içerisinde tablodaki kontrolleri ekleyin.

Kontrol – Kontrol İsmi	Özellik	Değer
HyperLink – Link1	NavigateUrl	
	Text	AnaSayfamYap
HyperLink – Link2	NavigateUrl	
	Text	SıkKullanicilaraEkle
HyperLink – Link3	NavigateUrl	mailto:tamer.sahiner@bilgeadam.com
	Text	İletisim



Alt isimli kullanıcı kontrolünün Html kodları aşağıdaki gibi olacaktır.

```

<%@ Control Language="cs" AutoEventWireup="false"
Codebehind="Alt.ascx.cs" Inherits="AspEticaret.Alt"
TargetSchema="http://schemas.microsoft.com/intellisense/ie5"
%>
<TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" border="0">
<TR>
<TD>
<P align="center"><asp:hyperlink id="Link1"
runat="server">AnaSayfamYap</asp:hyperlink>&nbsp;|<br/>
<asp:hyperlink id="Link2"
runat="server">SıkKullanicilaraEkle</asp:hyperlink>&nbsp;|<br/>
<asp:hyperlink id="Link3" runat="server"
NavigateUrl="mailto:tamer.sahiner@bilgeadam.com">İletisim</a
sp:hyperlink></P>

```

```

</TD>
</TR>
</TABLE>

```

Yan kontrolunun eklenmesi

ASPEticaret projesine Yan isminde yeni bir kullanıcı kontrolü ekleyin.



Yan isimli kullanıcı kontrolünün Html kodları aşağıdaki gibi olacaktır.

```

<%@ Control Language="cs" AutoEventWireup="false"
Codebehind="yan.ascx.cs" Inherits="AspEticaret.yan"
TargetSchema="http://schemas.microsoft.com/intellisense/ie5"
%>
<TABLE id="Table1" cellspacing="0" cellpadding="0"
width="150" border="0">
<TR>
<TD>
<P align="center"><A
href="http://www.yazilimuzmani.com"><IMG
src="resimler/YazilimUzmanı.gif" border="0"></A></P>
</TD>
</TR>
<TR>
<TD style="HEIGHT: 35px">
<P align="left"><IMG
src="resimler/bilgeadam%20logo.jpg" border="0"><A
href="http://www.yazilimuzmani.com"></A></P>
</TD>
</TR>
<TR>
<TD>
<P align="left"><A
href="http://sdnet.bilgeadam.com"><IMG
src="resimler/sdNetLogo.gif" border="0"></A></P>
</TD>
</TR>
</TABLE>

```

DataSet nesnesinin Oluşturulması

dsBook isminde yeni bir DataSet oluşturun.

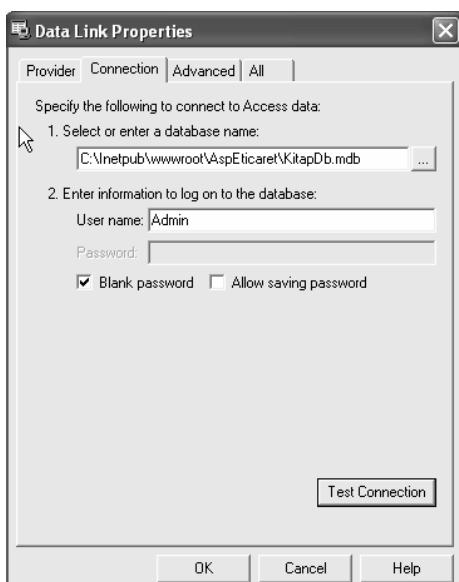
- 1- **Solution Explorer** penceresi açın.
- 2- Proje adı üzerinde farenin sağ butonunu tıklayın.
- 3- Açılan penceredeki **Add** menüsünden **Add New Item** komutunu seçin.
- 4- **Templates** seçenekleri içerisinde DataSet öğesini seçin
- 5- **Name** metin kutusuna dsBook ismini girin.

Bağlantı oluşturulması

KitapDB veritabanı üzerinde işlem yapılması için bağlantı kurulması gereklidir. Bu bağlantıyı Server Explorer’ı kullanarak oluşturun. Bu bağlantı ile veritabanı içerisindeki Kategori tablosu, dsBook isimli dataset’ine eklenecaktır.

KitapDb uygulaması için yeni bağlantı oluşturmak.

5. **Server Explorer** penceresi üzerinde farenin sağ butonunu tıklayın. Açılan menüden **Add Connection** komutunu tıklayın.
6. Açılan **Data Link Properties** penceresinin **Provider** sekmesini tıklayın.
7. Provider sekmesinden **Microsoft.Jet.OLEDB.4.0 Provider**’ı seçerek **Next** butonunu tıklayın.



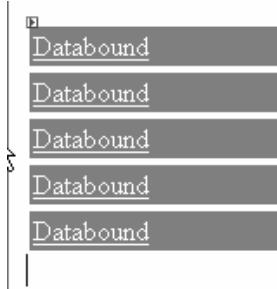
8. Açılan **Connection** sekmesinin görüntüsünü resimdeki gibi düzenleyerek **OK** butonunu tıklayın.
9. **Server Explorer** penceresinden **DataConnections** seçenekini seçin
10. Eklediğiniz bağlantı içerisinde **Tables** seçenekini seçin.
11. **Tables** içerisindeki Kategori tablosunu dsBook nesnesinin içerisine sürükleyin.

Kategori kontrolunun eklenmesi

ASPEticaret projesine Kategori isiminde yeni bir kullanıcı kontrolü ekleyin.

Kullanıcı kontrolu içerisinde tablodaki kontrolleri ekleyin.

Kontrol – Kontrol İsmi	Özellik	Değer
Repeater – rptKategori	NavigateUrl	Default.aspx



Kategori isimli kullanıcı kontrolünün Html kodları aşağıdaki gibi olacaktır.

```
<%@ Control Language="cs" AutoEventWireup="false"
Codebehind="kategori.ascx.cs"
Inherits="AspETicaret.kategori"
TargetSchema="http://schemas.microsoft.com/intellisense/ie5"
%>
<asp:Repeater id="rptKategori" runat="server">
    <ItemTemplate>
        <table width="150" cellpadding="2" cellspacing="2">
            <tr bgcolor="#0099ff">
                <td>
                    <a style="COLOR: white">
                        href='Kitap.aspx?KategoriID=<#
                            databinder.eval(Container.dataitem,"KategoriID") %>'>
                            <%# Databinder.eval(container.dataitem
                            , "KategoriAdi") %>
                    </a>
                </td>
            </tr>
        </table>
    </ItemTemplate>
</asp:Repeater>
```

Kategori isimli kullanıcı kontrolünün code behind kodları aşağıdaki gibi olacaktır.

```
using System.Data.OleDb;

private void Page_Load(System.Object sender,
System.EventArgs e)
{
    String connStr = "Provider=Microsoft.Jet.OleDB.4.0; " +
        "Data Source=" + Server.MapPath("KitapDb.mdb");
    OleDbConnection conn = new OleDbConnection();
```

```
conn.ConnectionString = connStr;

oleDbType comm = new OleDbCommand();
comm.CommandType = CommandType.Text;
comm.CommandText = "Select * from Kategori";
comm.Connection = conn;

OleDbDataAdapter da = new OleDbDataAdapter();
da.SelectCommand = comm;

DataSet ds = new DataSet();
try
{
    conn.Open();
    da.Fill(ds, "Kategori");
    rptKategori.DataSource = ds.Kategori;
    rptKategori.DataBind();
}
catch (Exception ex)
{
    Response.Write(ex.Message);
}
finally
{
    if (conn.State = ConnectionState.Open)
    {
        conn.Close();
    }
}
```


ADO.NET ile Veriye Erişim



Modul 11: ADO.NET ile Veriye Erişim

ADO.NET ile Veriye Erişim

- ◆ Veri Bağlantılı Kontroller
- ◆ Connected ve Disconnected Uygulamalar Geliştirme

Web uygulamaları ile veriye erişim, Windows uygulamalarına oldukça benzemektedir. Ancak verileri listelemek için kullanılan kontrollerin çalışma yapısı farklılık gösterir. Veriye ulaşım ADO.NET nesneleri ile gerçekleşir.

Bu modül tamamlandıktan sonra:

- **Repeater**, **DownList** ve **GridView** gibi listeleme kontrollerini öğrenecek,
- Web uygulamlarında Connected ve Disconnected çalışma yapısını öğrenecek,

Konu 1: Veri Bağlı Kontroller

Veri Bağlı Kontrolleri

- CheckBoxList, RadioButtonList Nesnelerini Kullanmak
- Repeater, DataList ve GridView Nesnelerini Kullanmak
- PlaceHolder Nesnesini Kullanmak

ASP.Net ile veritabanı içindeki veriyi görüntülemek ve düzenlemek için veri bağlantı kontrolleri kullanılır. **ListBox**, **DropDownList** kontrolleri dışında **CheckBoxList**, **RadioButtonList** kontrolleri veri bağlantılı olarak çalışabilir. **Repeater**, **DownList** ve **GridView** kontrolleri veri listelemek için kullanılır.

CheckBoxList, RadioButtonList Kullanımı

CheckBoxList, RadioButtonList Nesnelerini Kullanmak

- **DataSource**
 - DataSet nesnesine bağlanır.
- **DataMember**
 - DataSet içinden alınmak istenen verilerin ait olduğu tablo ismi.
- **DataValueField**
 - Value özelliğinde tutulması istenen alan.
- **DataTextField**
 - Text özelliğinde görüntülenmek istenen alan.

CheckBox ve **RadioButton** kontrollerinden farklı olarak, birden fazla seçenek arasında seçim yapılmasını sağlayan **CheckBoxList** ve **RadioButtonList** kontrolleri kullanılabilir. Örneğin bir sayfada dört tane isteğe bağlı seçenek varsa dört ayrı **CheckBox** kullanmak yerine, bir **CheckBoxList** kontrolü kullanılır. Aynı şekilde beş seçenekten sadece bir tanesi seçilmesi gerekiyorsa, beş ayrı **RadioButton** oluşturmak yerine, bir **RadioButtonList** kontrolü kullanılır.

CheckBoxList kontrolünün **DataSource**, **DataMember**, **DataTextField** ve **DataValueField** özellikleri ile veritabanı işlemleri gerçekleştirilir. **DataSource**, bağlantısız çalışan DataSet nesnesine bağlanır. **DataMember**, bu **DataSet** içerisindeki tablo ismini temsil eder. **DataValueField**, **value** özelliğinde tutulması istenen kolonu, **DataTextField** ise **text** özelliğinde görüntülenmek istenen kolonu temsil eder.

Örnekte **CheckBoxList** kontrolü ile seçilen tüm öğeler, **lblMsg** isimli etiketin içerisinde yazdırılmaktadır.

```
int i ;  
for (int i=0;i< checkboxlist1.Items.Count ;i++)  
{  
    if (checkboxlist1.Items(i).Selected)  
    {  
        lblMsg.Text &= checkboxlist1.Items(i).Text & "<br>"
```

```

        }
    }

CheckBoxList kontrolü ile birden fazla seçim yapılabılır. Fakat RadioButtonList kontrolü ile sadece bir öğe seçilebilir.

```

Örnekte **CheckBoxList** veya **RadioButtonList** kontrolü ile veritabanı bağlantısı gösterilmektedir.

```

private void Page_Load(System.Object sender,
                      System.EventArgs e)
{
    da.Fill(dataSet1, "Kitaplar");
    CheckBoxList1.DataSource = DataSet1;
    CheckBoxList1.DataMember = "Kitaplar";
    CheckBoxList1.DataTextField = "Kitap_baslik";
    CheckBoxList1.DataValueField = "kitap_ISBN";

    CheckBoxList1.DataBind();
}

```

Repeater, DataList ve DataGrid Kullanımı

Repeater, DataList ve DataGrid Kullanımı

- Ortak Şablonlar;
 - HeaderTemplate
 - ItemTemplate
 - FooterTemplate
- DataList ve DataGrid, veriler üzerinde güncelleme yapma imkanı sunar.
- Repeater sadece veri görüntüleme sağlar.

Repeater, **DataList** ve **DataGrid**, veri listelenmesi için tasarlanmış özel kontrollerdir.

Bu üç kontrol şablonlardan oluşur. Ortak şablonları ise **HeaderTemplate**, **ItemTemplate** ve **FooterTemplate** şablonlarıdır. Şablonlar içerisinde verinin görüntülenmesine yönelik tanımlamalar yapılır.

DataList ve **DataGrid**, veriler üzerinde güncelleme yapma imkanı sunarken, **Repeater** sadece veri görüntülemeyi sağlar. Ancak **DataGrid**, **DataList** kontrolünden farklı olarak veri sayfalama ve sıralama özellikleri sunar.

Repeater

Repeater

Şablon Adı	Açıklama
ItemTemplate	Veritabanından gelecek satırların görüntüleneceği stilin belirlendiği alan.
AlternatingItemTemplate	Ardışık olarak gelen satırların birbirinden farklı olamsını sağlar.
HeaderTemplate	Repeater kontrolünün başlığıdır. İstenilen stil verilebilir.
FooterTemplate	Repeater kontrolünün en altındaki alandır. Alt başlık olarak istenilen stil verilebilir.
SeparatorTemplate	Veritabanından gelen her bir satırı diğerinden ayıran şablondur.

Repeater; veriyi veritabanından alarak istenilen biçimde görüntülenmesini sağlayan oldukça güçlü bir kontroldür. Her kaydın görüntülenme şekli, HTML etiketleri ile oluşturulan bir şablon ile belirlenir. Haber yayını yapan sitelerinin çoğunda bu kontrol kullanılır.

Kod 11.1 de Repeater kullanımı gösterilmektedir.

Kod 11.1:Repeater Kullanımı

```
<asp:Repeater id="Repeater1" runat="server">
    <HeaderTemplate>
        <asp:Label id="lblh" Runat="server" text="Company _ 
            Name" Font-Bold="True" Width="260"></asp:Label>
        <asp:Label id="lblh2" Runat="server" text="Contact _ 
            Name" Font-Bold="True"></asp:Label>
    </HeaderTemplate>
    <ItemTemplate>
        <table>
            <tr>
                <td width="260">
                    <asp:Label ID=lbl1 Runat=server
                        text='<%#Databinder.eval(container.dataitem,
                            "companyname")%> ' >
                    </asp:Label>
                </td>
            </tr>
        </table>
    </ItemTemplate>
</asp:Repeater>
```

```

<td>
    <asp:Label ID=Lb12 Runat=server
        text='<%# Databinder.eval(container.dataitem,
        "contactname")%> ' >
    </asp:Label>
</td>
</tr>
</table>
</ItemTemplate>
<FooterTemplate>
    <b>BilgeAdam BT&lt;/b>
</FooterTemplate>
</asp:Repeater>

```

Repeater kontrolünün **ItemTemplate** şablonda, her kayıt için yapılacak gösterim şekli belirlenir. HTML etiketleri kullanarak çıktıya şekil verilebilir. **HeaderTemplate** şablonu **repeater** kontrolünün başlığının, **FooterTemplate** alt başlığının biçimini belirler.

HeaderTemplate içinde açılan bir **<table>** etiketi, **FooterTemplate** içinde **</table>** ifadesiyle kapatılabilir. **SeperatorTemplate** şablonu, kayıtlar arasında ayraç stilini belirler.

Tablo 11.1'de **Repeater** kontrolünün şablonları açıklanmıştır.

Şablon Adı	Açıklama
ItemTemplate	Veritabanından gelecek satırların görüntüleneceği stilin belirlendiği alan.
AlternatingItemTemplate	Ardışılı olarak gelen satırların birbirinden farklı olmasını sağlar.
HeaderTemplate	Repeater kontrolünün başlığıdır. İstenilen stil verilebilir.
FooterTemplate	Repeater kontrolünün en altındaki alandır. Alt başlık olarak istenilen stil verilebilir.
SeperatorTemplate	Veritabanından gelen her bir satırı diğerinden ayıran şablondur.

Tablo 11.1: Repeater kontrolünün şablonları

<table> </table> etiketleri ile kayıtların bir tablonun satırları biçiminde görüntülenmesini sağları. **<tr> </tr>** etiketi arasında iki **<td> </td>** etiketi kullanır. Bu şekilde, bir satır bilgiyi iki kolona ayrılmış biçimde görüntülenmesini istenir. İlk td etiketinde bir **<asp:Label>** etiketi kullanarak bu birinci sütunda verinin bir **Label** kontrolü içinde görüntülenmesi isteğini bildirilir. Ve Label etiketinin **Text** özelliği içine,

```
text='<%#Databinder.eval(container.dataitem,  
"companynname")%> '
```

ifadesi yazılarak veri bağlama işlemi gerçekleştirilir. Burada **Databinder.eval**, **repeater** kontrolüne bağlanan veri kümesi içinden **CompanyName** adı verilen kolonu bulur ve o kolondaki verileri sırasıyla repeater içine alır ve görüntüler.

İkinci **td** etiketinde, **Label** kontrolünün **Text** özelliğine **ContactName** kolonunu bağlar.

```
text='<%#Databinder.eval(container.dataitem,  
"contactname")%> '
```

Code-behind sayfasında ise, **Repeater** kontrolünün **DataSource** özelliğine, veri kaynağını temsil eden **DataSet** nesnesinin ismi bildirilir. Ve **DataBind** metodu ile bağlantının işlenmesi sağlanır.

```
Repeater1.DataSource = DataSet2;  
Repeater1.DataBind();
```

Sonuç olarak **Repeater** kontrolü, HTML kullanımını yoğun olarak gerektirir ve karşılığında, verilerin istenilen şablonla uygun biçimde görüntülenmesini sağlar.

DataList

DataList

- **SelectedItemTemplate**
 - Nesneye ait ayrıntıları görüntüler.
- **EditItemTemplate**
 - Kayıt üzerinde düzenleme yapmasını sağlar.
- **Özellikleri;**
 - RepeatLayout
 - RepeatColums
 - RepeatDirection
 - GridLines

DataList kontrolü, **Repeater** kontrolünün daha gelişmiş halidir. Veri görüntülemek dışında, verilerin seçili ve üzerinde güncelleme işlemleri yapmaya olanak sağlar.

Datalist eklemek için aşağıdaki adımları takip edin. Örnekte **DataList** kullanımı gösterilmektedir.

1. Araç kutusundan **DataList** kontrolü seçerek formumuzun üzerine sürükleyp bırakın.
2. Kontrol üzerine sağ tıklayın ve açılan menüden “Edit Template” alanını seçin.
3. Açılan yeni pencerede **Header and Footer Templates**, **Item Template** ve **Separator Templates** alanları çıkacaktır. **Header and Footer Templates** alanını seçerek, başlık ve alt başlık alanlarına istenilen form girilebilir. Header alanına iki label ekleyin ve **Text** özelliğine “Kitap Adı” ve “Yazar” değerini verin. **Footer** alanına yine bir **Label** ekleyin ve **Text** özelliğine “Bilge Adam BTA” yazın.
4. Kontrol üzerinde tekrar sağ tıklayın ve “Item Templates” alanını seçin. **ItemTemplate** ve **AlternatingItemTemplate** şablonuna ek olarak, **SelectedItemTemplate** ve **EditItemTemplate** şablonları bulunur. ItemTemplate alanında, görüntülemek istenilen alanları temsil edecek kontrolleri oluşturulur. Bu kontrolde repeater kontrolünden farklı olarak, tasarım ekranında araç kutusundan istenen kontrol **ItemTemplate** şablonuna eklenebilir.
5. İki tane **Label** kontrolünü **ItemTemplate** alanına ekleyin ve HTML koduna geçerek veri bağlama işlemlerini gerçekleştirin.

Kod 11.2: **DataList** kullanımında code-behind sayfası

```
string connStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
    "Data Source=" & Server.MapPath("./Stok.mdb");
OleDbConnection conn = new OleDbConnection(connStr);
OleDbDataAdapter da = new OleDbDataAdapter("select * from
    kitaplar", conn);
DataSet ds = New DataSet();

private void Page_Load(Object sender, EventArgs e)
{
    da.Fill(ds, "kitaplar");
    DataList1.DataSource = ds;
    DataList1.DataBind();
}
```

Kod 11.3: **DataList aspx** sayfası ve veri bağlama

```
<form id="Form1" method="post" runat="server">
    <asp:DataList id="DataList1" style="Z-INDEX: 101; LEFT:
        88px; POSITION: absolute; TOP: 168px" runat="server">
        <HeaderTemplate>
            <asp:Label id="Label1" runat="server" width="300px"
                Font-Bold="True">Kitap Adı</asp:Label>
```

```
<asp:Label id="Label12" runat="server" width="65px"
    Font-Bold="True">Yazar</asp:Label>
</HeaderTemplate>
<FooterTemplate>
    <asp:Label id="Label13" runat="server" Font-
        Bold="True">Bilge adam Bta</asp:Label>
</FooterTemplate>
<ItemTemplate>
    <asp:Label id=Label15 runat="server" width="300px"
        text='<%# databinder.eval(container.dataitem,
            "kitap_baslik")%>'> </asp:Label>
    <asp:Label id=Label14 runat="server" text='<%#
databinder.eval(container.dataitem, "kitap_yazar")%>'>
        </asp:Label>
    </ItemTemplate>
</asp:DataList>
</form>
```

DataList kontrolünün bir diğer farkı, çıktı görünümünün tablo içinde veya düz bir biçimde verilmesidir. **RepeatLayout** özelliğinin **Table** ve **Flow** değerlerini kullanarak tablo görünümü ve düz görünüm verilir. Varsayılan görülüm **Table** biçimindedir.

RepeatColumns özelliği ise verilerin kaç sütun halinde görüntüleneceğini belirler. **RepeatDirection** özelliği ise tekrarlanan kayıtların alt alta veya yan yana sıralanarak görüntülenmesini sağlar.

GridLines özelliği ise dikey ve yatay çizgilerle kayıt görüntülerini birbirinden ayırrır.

DataList içerisinde görüntü formunu düzenlemek için properties penceresindeki görüntüme ilişkin pek çok özellik sunulmuştur.

SelectedItemTemplate şablonu, listeden seçilen nesneye ait ayrıntıların görüntülenmesini sağlar.

EditItemTemplate şablonu, kullanıcının seçtiği kayıt üzerinde düzenleme yapmasını sağlayan alana ait kodların girildiği bölümdür..

DataList kontrolünün şablonları içerisinde kullanılan kontrollere, formun üzerinden direk erişilemez. Örneğin **DataList** içindeki bir **Button** kontrolünün **Click** olayına kod yazılamaz. **DataList** içinde kullanılan **Button** kontrolüne kod yazmak için, **DataList** kontrolünün **CommandName** özelliği kullanılır. Bu özellik, **Button** kontrolünü Command nesnesi ile alır ve forma yollar. Ve **DataList** kontrolünün **ItemCommand** olayında, gelecek komutun adına göre kod yazılır.

DataGrid

DataGrid

- Seçilen kayıt üzerinde değişiklik yapma.
- Seçilen kaydı silme.
- Seçilen kayıtları sayfalama.
- Seçilen kayıtları sıralama.

DataGrid kontrolü, **DataList** kontrolünden daha gelişmiş özelliklere sahiptir. Verileri sayfalama ve sıralama yeteneği sayesinde görüntüleme işlemleri özelleşmiştir. **DataGrid**, veritabanından alınan bir tabloyu, tablo biçimini ile ekrana yansıtılmasını sağlar. Seçilen kayıt üzerinde değişiklik yapma ve kayıt silme olanaklarını sağlar. Sayfalama, sıralama, seçme, düzenleme ve silme işlemlerini destekler. **DataGrid** kontrolüne veri bağlamak için şablon kullanmaya gerek yoktur.

Kod 11.4: DataGrid kontrolünün en basit kullanımı

```
string connStr = "Provider= Microsoft.Jet.OLEDB.4.0;"+  
    "Data Source=" + Server.MapPath("./Stok.mdb");  
    OleDbConnection conn = new OleDbConnection(connStr);  
    OleDbDataAdapter da = new OleDbDataAdapter("select * from  
    kitaplar", conn);  
    DataSet ds = new DataSet;  
  
private void Page_Load(Object sender, EventArgs e)  
{  
    da.Fill(ds, "kitaplar");  
    DataGrid1.DataSource = ds;  
    DataGrid1.DataMember = "kitaplar";  
    DataGrid1.DataBind();  
}
```

HTML

```
<asp:DataGrid id="DataGrid1" style="Z-INDEX: 101;  
LEFT: 28px; POSITION: absolute; TOP: 96px"  
runat="server" width="432px" Height="203px">
```

```
</asp:DataGrid>
```

Sadece tasarım ekranı kullanılarak **DataGrid** oluşturulabilir.

- **Server Explorer** panelinden yeni bir Access veritabanı bağlantısı oluşturun.
- Bu veritabanından, kullanmak istediğiniz tabloyu sürükleyerek form üzerine bırakın. Formun alt penceresinde iki yeni nesne oluşacaktır. (**OleDbConnection1** ve **OleDbDataAdapter1**)
- **OleDbDataAdapter1** nesnesi seçin ve **Properties** panelinden "Generate DataSet" komutunu verin.
- Açılan pencerede **Next** düğmeleri ile ilerleyin.
- Araç kutusundan **DataGrid** kontrolünü sürükleyip forma bırakın.
- **Properties** penceresinde **DataSource** alanına oluşturulan **DataSet** kontrolünün ismi, **DataMember** alanına, **DataSet** içine alınan tablolardan birini girin.
- **DataSet** kontrolünü veri ile dolduran ve bağlama işlemlerini gerçekleştiren kodları yazın.
`OleDbDataAdapter1.Fill(DataSet1, "Tablo_ismi");
DataGrid1.DataBind();`

DataGrid için hazırlanmış çeşitli şablonlar vardır. Hazır şablonları seçmek için **DataGrid** kontrolü üzerinde sağ tıklanır ve **AutoFormat** seçilir.

Varsayılan olarak **DataGrid** verileri **Grid** görünümünde sunar. **GridLines** özelliğine **Both**, **Horizontal**, **Vertical** ve **None** değerlerinden biri atanabilir. **BackImageUrl** özelliği sayesinde **DataGrid** kontrollünde bir arka plan resmi görüntülenebilir.

DataGrid Kontrolünde Kolon Oluşturma

Slide9

DataGrid kontrolü içerisinde çeşitli kolon türleri bulunur.

- **BoundColumn**
- **HyperLinkColumn**
- **TemplateColumn**
- **ButtonColumn**
- **EditCommandColumn**

AutoGenerateColumns özelliği, varsayılan olarak **True** değerini alır ve tablodan gelen kolonları değiştirmeden görüntüler.

BoundColumn

BoundColumn, **DataGrid** kontrollünün varsayılan kolonudur. Kayıtları görüntüler. Veri kaynağından alınan tablodan sadece belirli kolonların

görüntülenmesi istenirse, **BoundColumn** kontrolleri kullanılabilir. Kod 11.5'de veri kaynağından alınan tablonun istenilen kolonları görüntülenir.

Kod 11.5: DataGrid içerisinde BoundColumn kullanımı

BoundColumn ile sadece görüntülenmesi istenen kolonları **DataGrid** kontrolüne eklenir.

```
private void Page_Load(Object sender, EventArgs e)
{
    string connStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
                    "Data Source=" & Server.MapPath("./Stok.mdb")
    OleDbConnection conn = new OleDbConnection(connstr);
    OleDbCommand cmdSelect = new OleDbCommand("Select * From
                                                kitaplar", conn);

    conn.Open();
    DataGrid1.DataSource = cmdSelect.ExecuteReader();
    DataGrid1.DataBind();
    conn.Close();
}
```

Html

```
<asp:DataGrid ID="DataGrid1"
    AutoGenerateColumns="False"
    EnableViewState="False" Runat="Server">

    <Columns>
        <asp:BoundColumn DataField="Kitap_baslik"
            HeaderText="Kitap Adı" />
        <asp:BoundColumn DataField="Kitap_yazar"
            HeaderText="Yazar Adı"/>
    </Columns>
</asp:DataGrid>
```

AutoGenerateColumns özelliği varsayılan olarak **True** değerindedir ve tüm kolonların otomatik olarak görüntülenmesini sağlar. Örnekte bu özelliğe **False** değeri atanmıştır.

BoundColumn kolonunun birçok özelliği vardır.

- **DataField**
- **DataFormatString**
- **FooterText**
- **HeaderImageUrl**
- **HeaderText**

BoundColumn kolonu, **DataGrid** kontrolünün **Columns** etiketi içerisinde tanımlanmıştır. **DataField** özelliğinde ise kolon adı belirtilmiştir.

İPUCU: DataGrid içinde görüntülenecek kolonları seçen sql komut tanımlanırken select * from ... ifadesinden kaçınılmalıdır. Bu komut yerine sadece ihtiyaç duyulan kolonlar tek tek belirtilmelidir. Aksi halde, Web Üzerinde yayın anında performans kaybı ortaya çıkar. BoundColumn'un DataFormatString özelliği ise, kolondan alınan ifadenin belirli bir formatta görüntülenmesini sağlar. Örneğin bir para miktarı söz konusuysa bu özellik kullanılabilir.

```
<asp:BoundColumn DataField="Kitap_fiyat"
DataFormatString="{0:c}" />
```

BoundColumn'un **HeaderText**, **FooterText** ve **HeaderImageUrl** özellikleri, header, footer alanlarına görüntülenmesi istenilen yazıları, ve başlıkta görüntülenecek resimi belirler. **HeaderText** alanına yazılan yazının görüntülenebilmesi için **DataGrid** kontrolünün **ShowFooter** özelliği **True** yapılmalıdır. Bu özellik varsayılan olarak **False** değerindedir. Aynı anda **HeaderImageUrl** ve **HeaderText** özelliklerine değer girildiğinde, resim ve yazı beraber görüntülenemez.

Örnekte **Header** alanında hem resim hem de yazı gösterilir

```
HeaderText="

```

HyperLinkColumn

HyperLinkColumn, kayıtları linkler şeklinde görüntüleyen kolondur. Yani **DataGrid** kontrolünde görüntülenen kayıtlar üzerinden başka sayfalara ilgili linkler verilmek isteniyorsa, **HyperLinkColumn** kullanılmalıdır. **DataGrid** kayıtlarına ilişkin ayrıntılı bilgi verilmek isteniyorsa master/detail formları şeklinde görüntü vermek için yine bu kolon kullanılabilir.

Örnekte HyperLinkColumn kullanımı gösterilmektedir. DataGrid nesnesi üzerindeki Detaylar kolunu tıklanarak, detay bilgileri getirilebilir. Bu bilgiler Detaylar.aspx sayfası üzerinde gösterilir.

Kod 11.7: DataGrid içerisinde HyperLinkColumn kullanımı

```
private void Page_Load(Object sender,EventArgs e)
{
    string connStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
        "Data Source=" & Server.MapPath("./Stok.mdb");
    OleDbConnection conn =new OleDbConnection(connStr);
    OleDbCommand cmdSelect = OleDbCommand("Select * From" +
        "musteri", conn);
    conn.Open();

    DataGrid1.DataSource = cmdSelect.ExecuteReader();
    DataGrid1.DataBind();
    conn.Close();
}
```

Html

```
<asp:DataGrid ID="DataGrid1"
    AutoGenerateColumns="False" EnableViewState="False"
    CellPadding="10" Runat="Server">

    <Columns>
        <asp:BoundColumn
            HeaderText="Müşteri Adı"
            DataField="musteri_ad" />
        <asp:BoundColumn
            HeaderText="Müşteri Soyadı"
            DataField="musteri_soyad" />
        <asp:HyperLinkColumn
            HeaderText="Detaylar"
            DataNavigateUrlField="musteri_id"
            DataNavigateUrlFormatString="Detaylar.aspx?id={0}"
            Text="Detay Görüntüle" />
    </Columns>

</asp:DataGrid>
```

Sayfaya Parametre Yollama

Tablodan genel bilgi verilecek alan belirlenir ve tıklandığı zaman o kolona ait veri hakkında daha ayrıntılı bilgi görüntülemek için detay sayfasına link verilir.

HyperLinkColumn kolonunda görüntülenen linkler, **DataNavigateUrlField**, **DataNavigateUrlFormatString** ve **Text** özelliklerine girilen bilgiler ile yapılandırılır. **DataNavigateUrlField** linkin adresini, **DataTextField** link üzerinde görüntülenecek yazıyı tutar.

HyperLinkColumn kolonunun özellikleri:

- **DataNavigateUrlField**
- **DataNavigateUrlFormatString**
- **DataTextField**
- **DataTextFormatString**
- **FooterText**
- **HeaderImageUrl**
- **HeaderText**
- **NavigateUrl**
- **Target**
- **Text**

DataTextField ve **DataTextFormatString** özellikleri, her bir **hyperlink** için farklı etiketler görüntülenmesi için kullanılabilir.

Örnekte HyperLinkColumn kullanımı gösterilmektedir. Site linkleri(link_url) ve başlıklar(link_title) veritabanındaki linkler tablosundan çekilerek, DataGrid

Üzerinde gösterilir. Link_title kolonu site başlıklarının görüntülenmesini sağlar. Link_title kolonu tıklanarak link_url kolonundaki adres bilgisine yönlendirilir.

Kod 11.8: DataGrid içerisinde HyperLinkColumn kullanımı

```
private void Page_Load(Object sender, EventArgs e)
{
    string connStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
        "Data Source=" + Server.MapPath("./Stok.mdb");
    OleDbConnection conn = new OleDbConnection(connStr);
    OleDbCommand cmdSelect = OleDbCommand ("Select * From
        linkler", conn);
    conn.Open();

    DataGrid1.DataSource = cmdSelect.ExecuteReader();
    DataGrid1.DataBind();
    conn.Close();
}
```

Html

```
<asp:DataGrid ID="DataGridLink"
    AutoGenerateColumns="False" EnableViewState="False"
    ShowHeader="False" CellPadding="10" Runat="Server">

<Columns>
    <asp:HyperLinkColumn
        DataNavigateUrlField="link_url"
        DataTextField="link_title" />
</Columns>

</asp:DataGrid>
```

TemplateColumn

TemplateColumn, kayıtları bir şablona uyarak görüntüleyen kolondur. **DataGrid** hücreleri içinde görüntülenecek verileri çeşitli kontroller kullanarak ekrana yansıtmak için bu kolon kullanılır. Ancak **TemplateColumn**, kendi içinde **HeaderTemplate**, **FooterTemplate**, **ItemTemplate** ve **EditItemTemplate** olmak üzere alanlara ayrıılır.

Kod 11.9 da **TemplateColumn** kullanımı gösterilmektedir. **TemplateColumn** alanında, kitabı ait yazar ve açıklama bilgileri görüntülenir. Ancak bu iki kolon bilgileri HTML kodlarıyla alınır.

Kod 11.9: DataGrid içerisinde TemplateColumn kullanımı

```
private void Page_Load(Object sender, EventArgs e)
{
    string connStr= "Provider=Microsoft.Jet.OLEDB.4.0;" +
        "Data Source=" + Server.MapPath("./Stok.mdb");
```

```
oleDBConnection conn = new oleDBConnection(connstr);
oleDBCommand cmdSelect = oleDBCommand("Select * From
    kitaplar",conn);
conn.Open();

DataGrid1.DataSource = cmdSelect.ExecuteReader();
DataGrid1.DataBind();
conn.Close();
}
```

HTML

```
<asp:DataGrid ID="DataGrid1"
    AutoGenerateColumns="False" EnableViewState="False"
    ShowHeader="False" CellPadding="10" Runat="Server">

<Columns>
    <asp:BoundColumn
        DataField="kitap_baslik" />

    <asp:TemplateColumn>
        <itemTemplate>
            <table>
                <tr>
                    <td>Yazar:</td>
                    <td><%# DataBinder.Eval(Container.DataItem,
"kitap_yazar" )%></td>
                </tr>
                <tr>
                    <td>Açıklama:</td>
                    <td><%# DataBinder.Eval(Container.DataItem,
"kitap_aciklama" )%></td>
                </tr>
            </table>
        </itemTemplate>
    </asp:TemplateColumn>
</Columns>

</asp:DataGrid>
```

DataGrid Kontrolünde Kolon Oluşturma

- **ButtonColumn**
 - Button kontrollerini görüntüler.
- **EditCommandColumn**
 - Edit, Update, Cancel gibi düzenleme komutlarını görüntüler.

ButtonColumn

ButtonColumn, **Button** kontrollerinin görüntülenmesini sağlar. Uygulanacak metot kolon üzerinde, button şeklinde görüntülenir. Örneğin “Sepete Ekle” gibi bir iş için **Button** kullanılır ve **ButtonColumn** içerisinde tanımlanır.

ButtonColumn alanı kullanarak **Select** ismindeki butona tıkladığı zaman kontrolün arka plan rengi ve yazı kalınlığı değiştirilir. **UnSelect** seçildiğinde kontrol eski haline getirilir.

Kod 11.9: DataGrid içerisinde ButtonColumn kullanımı

```
private void Page_Load(Object sender, EventArgs e)
{
    string connStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
        "Data Source=" + Server.MapPath("./stok.mdb");
    OleDbConnection conn = new OleDbConnection(connStr);
    OleDbCommand cmdSelect = OleDbCommand("Select * From
        kitaplar", conn);

    conn.Open();
    DataGrid1.DataSource = cmdSelect.ExecuteReader();
    DataGrid1.DataBind();
    conn.Close();
}

void DataGrid1_ItemCommand(s, DataGridCommandEventEventArgs e)
{
    if (e.CommandName == "select")
    {
        e.Item.BackColor = System.Drawing.Color.LightGreen;
```

```

        e.Item.Font.Bold = true;
    }
    else
    {
        e.Item.BackColor = System.Drawing.Color.Blue
        e.Item.Font.Bold = False
    }
}

```

HTML

```

<asp:DataGrid ID="DataGrid1"
    OnItemCommand="DataGrid1_ItemCommand"
    AutoGenerateColumns="False"
    CellPadding="10" Runat="Server">
<Columns>
    <asp:BoundColumn
        HeaderText="Kitap Adı" DataField="kitap_baslik" />

    <asp:ButtonColumn
        CommandName="select"
        Text="Select!" />

    <asp:ButtonColumn
        CommandName="unselect"
        Text="UnSelect!" />
</Columns>
</asp:DataGrid>

```

Select butonuna basıldığında, **DataGrid** kontrolünün **OnItemCommand** özelliğinde belirtilen **DataGrid1_ItemCommand** isimli metot devreye girer. **ItemCommand** olayını tetikleyen **DataGrid1_ItemCommand** isimli metot, ilgili işlemleri gerçekleştirir.

UnSelect düğmesine tıklandığında ise, yine **OnItemCommand** özelliğinde tutulan **DataGrid1_ItemCommand** metodu devreye girer. Ancak tıklanan butonun isimlerine göre yapılacak işlem belirlenir.

```

if (e.CommandName=="select")
{
    e.Item.BackColor = System.Drawing.Color.LightGreen;
    e.Item.Font.Bold = true;
}
else
{
    e.Item.BackColor = System.Drawing.Color.White;
    e.Item.Font.Bold = False;
}

```

e.CommandName, hangi buttonun tıklandığını belirtir. Tıklanan butona göre hangi metodun uygulanacağını belirler.

ButtonColumn özellikleri:

- **ButtonType:** **LinkButton** veya **PushButton**

- **CommandName**
- **TextField**
- **DataTextFormatString**
- **FooterText**
- **HeaderImageUrl**
- **HeaderText**
- **Text**

EditCommandColumn

EditCommandColumn, **Edit**, **Update**, **Cancel** gibi düzenleme komutlarının görüntülenmesini sağlar. **EditCommandColumn** ile sadece bir satır düzenlenebilir. Düzenlemenin veritabanı geçmesi ayrı işlemler gerektir.

EditCommandColumn kolonunun görüntülediği kayıt, düzenleme için kayıt seçilen **DataGridView** nesnesinin **EditRowIndex** özelliğine göre değişecektir.

Düzenleme işleminin seçili olmadığı durumda bu kolonda **Edit** butonu gözükmür. **Edit** seçildiği anda ise **Update** ve **Cancel** butonları gözükmür.

EditCommandColumn özellikleri:

- **ButtonType**
- **CancelText**
- **EditText**
- **FooterText**
- **HeaderImageUrl**
- **HeaderText**
- **UpdateText**

DataGrid Kontrolünde Sıralama Ve Sayfalama

DataGrid Kontrolünde Sıralama ve Sayfalama

- Sıralama için;
 - AllowSorting özelliği True yapılır.
 - SortCommand olayı tetiklenir.
- Sayfalama
 - AllowPaging özelliği True yapılır.
 - PageIndexChanged olayını tetiklenir.

DataGrid kontrolünün kolonlarında sıralama yapmak için hazırlanmış özellikler vardır. İsteğe göre tüm kolonlarda veya sadece belirli kolonlarda sıralama yapılabilir.

DataGrid içindeki tüm kolonlara sıralama yapma izni vermek için, varsayılan olarak **False** olan **AllowSorting** özelliği **True** yapılır. Ve **SortCommand** olayını tetikleyecek bir metot yazılır.

Kod 11.10: DataGrid içerisinde Sıralama

```
private void Page_Load(Object sender,EventArgs e)
{
    If (! IsPostBack)
    {
        BindDataGrid( "kitap_baslik" );
    }
}

void BindDataGrid(string strSortField)
{
    String connStr = "Provider=Microsoft.Jet.OLEDB.4.0;" +
                    "Data Source=" + Server.MapPath("./stok.mdb");
    oleDbConnection conn = new oleDbConnection(connStr);
    oleDbCommand cmdSelect = oleDbCommand("Select * From
                                            Kitaplar Order By " & strSortField, conn );
    conPubs.Open("Select * From kitaplar", conn)
    conn.Open();
```

```

        DataGrid1.DataSource = cmdSelect.ExecuteReader();
        DataGrid1.DataBind();
        conn.Close();
    }

    void DataGrid1_SortCommand (Object s,
DataGridSortCommandEventArgs e)
{
    BindDataGrid( e.SortExpression );
}

```

Html

```
<asp:DataGrid ID="DataGrid1" AllowSorting="True"
    OnSortCommand="DataGrid1_SortCommand"
    CellPadding="10" Runat="Server" />
```

BoundColumn kolonunun **SortExpression** özelliğine ilgili kolon isimleri girilerek sıralama yapılacak kolonlar belirtilebilir.

Sayfalama

Sayfalarca uzunluktaki kayıtları bir seferde göstermek yerine sayfalara ayırmak daha kullanışlı olur. **DataGrid** kontrolünde sayfalama yapabilmek için kontrolün **AllowPaging** özelliği **True** yapılır. Varsayılan değer **False** değeridir. **PageIndexChanged** olayını tetikleyecek bir metodun yazılması gereklidir.

Kod 11.11: DataGrid içerisinde Sayfalama

```

private void Page_Load(Object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        BindDataGrid();
    }
}

void BindDataGrid()
{
    string connStr =
"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source='"+Server.MapPath("./Stok.mdb")+"'";

    OleDbConnection conn = new OleDbConnection(connStr);
    OleDbDataAdapter da = new OleDbDataAdapter("Select * 
From Kitaplar Order By Kitap_baslik", conn);
    DataSet ds = new DataSet();
    da.Fill(ds);
}

private void DataGrid1_PageIndexChanged(object source,
System.Web.UI.WebControls.DataGridPageChangedEventArgs e)
{
    DataGrid1.CurrentPageIndex = e.NewPageIndex;
}

```

html

```
<asp:DataGrid ID="DataGrid1"
```

```
AllowPaging="True" PageSize="5"
OnPageIndexChanged="DataGrid1_PageIndexChanged"
CellPadding="3" Runat="Server" />
```

DataGrid kontrolünde sayfalama yapıldığında kayıtlar sayfalara ayrıılır ve diğer sayfalara linkler verilir. **PageSize** özelliği, bir sayfada kaç kayıt görüntüleneceği bilgisini tutar.

Sayfalamaya ait stiller, tasarım penceresinde **DataGrid** kontrolüne sağ tıklanıp **Property Builder** ile seçilebilir.

DataGrid Kontrolü Üzerinde Kayıt Düzenleme İşlemleri

DataGrid kontrolünün **EditCommand**, **UpdateCommand** ve **CancelCommand** olayları kullanılarak **DataGrid** içinde görüntülenen veriler üzerinde istenilen değişiklikler yapılabilir. Aynı şekilde kayıt silme işlemi de gerçekleştirilir.

Düzenleme yapılacak kayıt seçildiğinde **EditCommand** olayı devreye girer. **EditItemIndex** özelliği ile düzenleme yapılacak kaydın indeksi alınır ve o satırda tüm veriler **TextBox** kontrollü biçiminde görünür. Üzerinde düzenleme yapılması istenmeyen kolona, **BoundColumn** alanının **ReadOnly** özelliğine **True** değeri verilmelidir.

Update düğmesine tıklanlığında ise **UpdateCommand** olayı devreye girer. İlgili kaydın **Primary Key** değeri alınır ve **Primary Key** ile güncelleme kodu çalıştırılır.

PlaceHolder Kullanımı

Programın çalışma zamanı sırasında, kullanıcıdan gelecek isteğe göre yeni kontroller eklenmek isteniyorsa **PlaceHolder** kontrolü kullanılır. **PlaceHolder** kontrolünün amacı, dinamik olarak eklenen bu kontrollerin bir arada tutulmasıdır. Dinamik olarak oluşturulan kontroller istenildiği gibi dizayn edilebilir.

```
<asp:PlaceHolder id="PlaceHolder1" runat="server">
</asp:PlaceHolder>
```

Çalışma zamanında forma yeni bir kontrol eklemek için **Controls.Add()** metodu kullanılır.

Kod 11.12: PlaceHolder Ekleme

```
private void Page_Load(Object sender, EventArgs e)
{
    int i;
    Button btnNewButton;
```

```
for (i = 1;i<=10;i++)
{
    PlaceHolder1.Controls.Add(
        New LiteralControl("<p>Alan " & i & ": ">"));
    PlaceHolder1.Controls.Add(New TextBox());
}

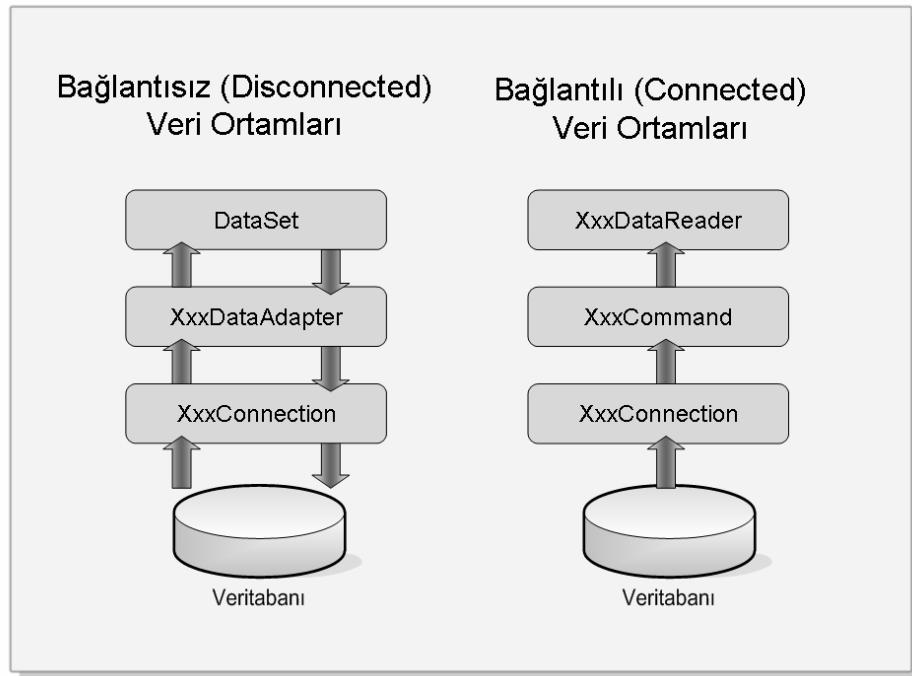
btnNewButton = New Button();
btnNewButton.Text = "Tıklayın!";
PlaceHolder1.Controls.Add(btnNewButton);
}
```

Konu 2: Connected ve Disconnected Uygulamalar Geliştirme

Connected ve Disconnected Uygulamalar Geliştirme

- ◆ Namespace
- ◆ Connected Uygulama
- ◆ Disconnected Uygulama

ADO.NET ile veriye erişmek için Connected ve Disconnected veri erişim yöntemi kullanılır. Bu yöntemler ile ASP.NET sayfalarında veri alışverişi yapılır.



ASP.NET uygulamaları web sunucuları üzerinde işlem yapacağı için performans çok önemlidir. Dolayısıyla, çalışma modelinin yerinde seçilmesi gereklidir. Örneğin veriler sadece görüntülenmek amacıyla alınacaksa Connected bağlantı modeli kurulmalı ve kaynaklar mümkün olan en az seviyede tüketilmelidir. Ancak veri üzerinde güncelleme işlemleri söz konusuya disconnected bağlantı modeli uygulanmalıdır.

Namespace

Namespace

- System.Data isimini import edilir.
 - Imports System.Data
- System.Data.OleDb isimini import edilir.
 - Imports System.Data.OleDb
- Inline kod yazımında
 - <%@ Import Namespace="System.Data" %>
 - <%@ Import Namespace="System.Data.OleDb" %>

ADO.NET sınıflarını, ASP.NET uygulaması içinde kullanabilmek için **System.Data** isimini **using** yapılmalıdır. Ayrıca Access veri tabanına bağlantı için **System.Data.OleDb** isimini import edilmelidir.

Code behind sayfasında **using** işlemi, Windows uygulamalarında kullanılan biçimdedir.

```
using System.Data;  
using System.Data.OleDb;
```

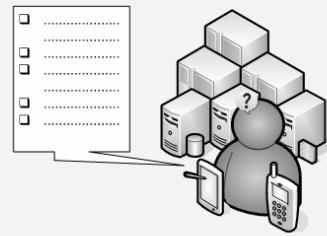
Inline kod yazımında **<%@ %>** ifadeleri arasında isim alanları **using** yapılır.

```
<%@ using Namespace="System.Data" %>  
<%@ using Namespace="System.Data.OleDb" %>
```

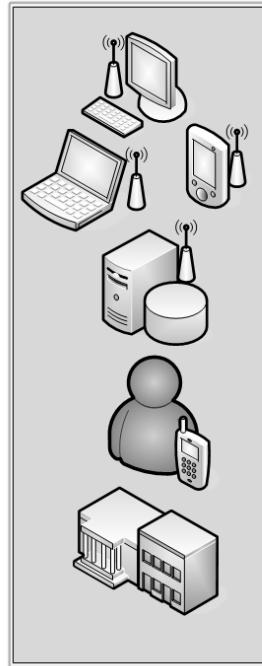
Modül Özeti

Modül Özeti

- Veri bağlantı kontrolleri nelerdir? Açıklayın
- Repeater niçin kullanılır? Açıklayın
- DataList niçin kullanılır? Açıklayın
- DataGridView niçin kullanılır? Açıklayın
- DataGridView ile sayfalama nasıl yapılır?



34. Veri bağlantı kontrolleri nelerdir? Açıklayın
35. Repeater niçin kullanılır? Açıklayın
36. DataList niçin kullanılır? Açıklayın
37. DataGridView niçin kullanılır? Açıklayın
38. DataGridView ile sayfalama nasıl yapılır?



Uygulamalar

ADO.NET ile Veriye Erişim

- ◆ E-Ticaret Uygulaması Geliştirmek

Bu uygulamada, e-ticaret uygulaması ile Connected ve Disconnected veritabanı işlemleri gerçekleştirilecektir. Bu uygulamada Üye kayıt ve Üye giriş işlemlerini gerçekleştirebilirsiniz. Ayrıca kategoriye göre tüm kitapları listeleyecek ve kitabı satın alma işlemini gerçekleştirebilirsiniz.

Bu lab tamamlandıktan sonra:

- Connect ve Disconnect veritabanı işlemlerini öğreneceksiniz.
- DataSet içerisindeki veriyi Repeater, DataGrid ve DataList kontrollerine bağlayabileceksiniz.

Connect Veritabanı İşlemleri

AspEticaret isimli projeyi açın.

UyeKayıt formu ile veritabanı işlemlerinin yapılması

UyeKayıt web formunu açın.

UyeKayıt web formunun Code Behind kodları aşağıdaki gibi olacaktır.

```
using System.Data.OleDb ;  
  
private void btnKaydet_Click(System.Object sender,  
System.EventArgs e)  
{  
    string connStr= "Provider=Microsoft.Jet.OleDb.4.0;"+  
    "Data Source=" & Server.MapPath("KitapDb.mdb");  
}
```

```

        OleDbConnection conn = new OleDbConnection();
        conn.ConnectionString = connStr

        OleDbCommand comm = new OleDbCommand();
        comm.Connection = conn;
        comm.CommandType = CommandType.Text;
        comm.CommandText = "INSERT INTO
                           Musteri(Ad,Soyad,Email,Sifre)
                           values(@ad,@soyad,@email,@sifre) " ;

        comm.Parameters.Add("@ad", txtAd.Text);
        comm.Parameters.Add("@soyad", txtSoyad.Text);
        comm.Parameters.Add("@email", txtEmail.Text);
        comm.Parameters.Add("@sifre", txtSifre.Text);
        int sonuc;
        try
        {
            conn.Open();
            sonuc = comm.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            Response.Write(ex.Message);
        }
        finally
        {
            conn.Close();
        }
        if (sonuc == 1)
        {
            Response.Redirect("KAYIT.aspx");
        }
    }
}

```

UyeGiris formu ile veritabanı işlemlerinin yapılması

UyeGiris web formunu açın.

UyeGiris web formunun Code Behind kodları aşağıdaki gibi olacaktır

```

using System.Data.OleDb;

private void btnGiris_Click(System.Object sender,
System.EventArgs e)
{
    Session["user"] = "tamer";
    string connStr ="Provider=Microsoft.Jet.OleDB.4.0;" +
                    "Data Source=" + Server.MapPath("KitapDb.mdb");

    OleDbConnection conn = new OleDbConnection();
    conn.ConnectionString = connStr;

    OleDbCommand comm =new OleDbCommand();
    comm.CommandType = CommandType.Text;
    comm.CommandText =
        "Select * from Musteri where Email=@email and" +
        "Sifre=@sifre";
    comm.Connection = conn;
}

```

```

comm.Parameters.Add("@email", txtEmail.Text);
comm.Parameters.Add("@sifre", txtSifre.Text);
bool sonuc;
oleDbTypeReader dr = oleDbTypeReader();
try
{
    conn.Open();
    dr = comm.ExecuteReader();

    if (dr.HasRows == true)
    {
        sonuc = true;
        if (dr.Read = true)
        {
            Session["user"] = dr.Item("Email");
            Session["ad"] = dr.Item("Ad");
            Session["soyad"] = dr.Item("Soyad");
            Session["musteriId"] = Item("MusteriID");
        }
    }
    else
    {
        sonuc = false;
    }

    dr.Close();
}
catch (Exception ex)
{
    Response.Write(ex.Message);
}
finally
{
    if (conn.State == ConnectionState.Open)
    {
        conn.Close();
    }
}
if (sonuc == true)
{
    Response.Redirect("Default.aspx");
}
Else
{
    lblMesaj.Text = "Hatalı kullanici adi veya sifre";
}
}

```

KitapDetay formunun eklenmesi ve veritabanı işlemlerinin yapılması

ASPEticaret projesine KitapDetay isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
Label1 – lblKitapAdi		

Label – lblYazarAdi		
Label – lblFiyat		
Label – lblAciklama		
Label – lblMesaj		
Image – imgResim		
TextBox – txtAdet		
Button – btnSatinAl	Text	Satin Al

Yazılım Uzmanı Kitapevi

Bay / Bayan : Tamer ŞAHİNER

AnaSayfa | Üye Giriş | Üye Kayıt | Çıkış

Klasik

Bilgisayar

Edebiyat

Roman

Şair

Fıkra

Macera

Bilim-Kurgu

Tarih

Sağlık

S.Tarih



ADO .NET
Tamer ŞAHİNER
100
ADO .NET

Adet: Satın Al



Bilge Adam
Bilge Adam's Club

AnaSayfanı Yap | Sık Kullanılanlara Ekle | İletişim

KitapDetay Web formun Html kodları aşağıdaki gibi olacaktır

```
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="c#" AutoEventWireup="false"
Codebehind="KitapDetay.aspx.cs"
Inherits="AspEticaret.KitapDetay" %>
<%@ Register TagPrefix="uc1" TagName="kategori"
Src="kategori.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Ust" Src="Ust.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
    <HEAD>
        <title>KitapDetay</title>
        <meta content="Microsoft Visual Studio .NET 7.1"
name="GENERATOR">
        <meta content="C#" name="CODE_LANGUAGE">
        <meta content="JavaScript"
name="vs_defaultClientScript">
        <meta
content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
    </HEAD>
    <body bgColor="#f0ffff">
        <form id="Form1" method="post" runat="server">
            <TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" align="center" border="0">
                <TR>
                    <TD bgColor="#99ccff" colspan="3"><uc1:ust
id="Ust1" runat="server"></uc1:ust></TD>
```

```

        </TR>
        <TR>
            <TD width="150" vAlign="top"><uc1:kategori
id="Kategori1" runat="server"></uc1:kategori></TD>
            <TD vAlign="top" width="400">
                <P><BR>
                </P>
                <P>
                    <TABLE id="Table2" borderColor="#000033"
cellspacing="0" cellPadding="0" width="300" align="center"
border="0">
                        <TR>
                            <TD width="100" rowspan="5">
                                <P align="center"><asp:image
id="imgResim" runat="server"></asp:image></P>
                            </TD>
                            <TD>
                                <P align="center"><asp:label
id="lblKitapAdi" runat="server"></asp:label></P>
                            </TD>
                        </TR>
                        <TR>
                            <TD>
                                <P align="center"><asp:label
id="lblYazarAdi" runat="server"></asp:label></P>
                            </TD>
                        </TR>
                        <TR>
                            <TD>
                                <P align="center"><asp:label
id="lblFiyat" runat="server"></asp:label></P>
                            </TD>
                        </TR>
                        <TR>
                            <TD>
                                <P align="center"><asp:label
id="lblAciklama" runat="server"></asp:label></P>
                            </TD>
                        </TR>
                        <TR>
                            <TD>
                                <P align="center">Adet:<br/>
                                    <asp:textbox id="txtAdet"
runat="server" width="68px"></asp:textbox>&nbsp;
                                    <asp:button id="btnSatinAl"
runat="server" Text="Satın Al"></asp:button></P>
                            </TD>
                        </TR>
                        <TR>
                            <TD colspan="2">
                                <P align="center">
                                    <asp:Label id="lblMesaj"
runat="server"></asp:Label></P>
                                </TD>
                            </TR>
                            <TR>
                                <TD>
                                    <P align="right">
                                        <asp:button id="btnGeri"
runat="server" Text="Geri" style="width: 100px; height: 30px;"></asp:button>
                                    </P>
                                </TD>
                            </TR>
                        </TABLE>
                    </P>
                </TD>
            <TD width="150" bgColor="#0099ff" vAlign="top">
                <uc1:yan id="Yan1"
runat="server"></uc1:yan></TD>
            </TR>
            <TR>

```

```

        <TD bgColor="#99ccff" colspan="3"><uc1:alt
id="Alt1" runat="server"></uc1:alt></TD>
    </TR>
</TABLE>
</form>
</body>
</HTML>

```

KitapDetay web formunun Code Behind kodları aşağıdaki gibi olacaktır.

```

using System.Data.OleDb ;

string kID;
private void Page_Load(System.Object sender,
System.EventArgs e)
{
    if (Session["user"] == "")
    {
        Response.Redirect("Giris.aspx");
    }

    kID = Request.Params["kID"];
    'Response.Write(kID)
    string connStr= "Provider=Microsoft.Jet.OleDb.4.0;" +
                    "Data Source=" + Server.MapPath("KitapDb.mdb");

    OleDbConnection conn = new OleDbConnection();
    conn.ConnectionString = connStr;

    OleDbCommand comm =new OleDbCommand();
    comm.CommandType = CommandType.Text;
    comm.CommandText =
        "Select * from Kitap where KitapID =@kitapID";
    comm.Connection = conn;

    comm.Parameters.Add("@kitapID",Convert.ToInt32(kID));
    OleDbDataReader dr = OleDbDataReader();
    try
    {
        conn.Open();
        dr = comm.ExecuteReader();
        if (dr.Read = True)
        {
            lblKitapAdi.Text = dr.Item("KitapAdi");
            lblYazarAdi.Text = dr.Item("Yazar");
            lblFiyat.Text = dr.Item("Ucret");
            lblAciklama.Text = dr.Item("Aciklama");
            imgResim.ImageUrl = "resimler/" +
                dr.Item("Image");
        }
        dr.Close();
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
    finally
    {
        If (conn.State == ConnectionState.Open)
    }
}

```

```

                conn.Close();
            }
        }

private void btnSatinAl_Click(System.Object sender,
System.EventArgs e)
{
    if (txtAdet.Text == "")
    {
        lblMesaj.Text = "Adet Giriniz";
        return;
    }

    string connStr = "Provider=Microsoft.Jet.OleDb.4.0;" +
                    "Data Source=" + Server.MapPath("KitapDb.mdb");

    OleDbConnection conn = new OleDbConnection();
    conn.ConnectionString = connStr;

    OleDbCommand comm = new OleDbCommand();
    comm.Connection = conn;
    comm.CommandType = CommandType.Text;
    comm.CommandText = "INSERT INTO
Siparis(MusteriID,SiparisTarihi,KitapID,Adet)
values(@MusteriID,@SiparisTarihi,@KitapID,@Adet)";

    comm.Parameters.Add("@MusteriID",
Session["musteriId"]);
    comm.Parameters.Add("@SiparisTarihi",
DateTime.Now.ToShortDateString());
    comm.Parameters.Add("@KitapID", CInt(kID));
    comm.Parameters.Add("@Adet", txtAdet.Text);
    int sonuc
    try
    {
        conn.Open();
        sonuc = comm.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        Response.Write(ex.Message);
    }
    finally
    {
        conn.Close();
    }
    if (sonuc = 1)
    {
        Response.Redirect("Satis.aspx");
    }
}

```

Disconnect Veritabanı İşlemleri

AspEticaret isimli projeyi açın.

Default formunun eklenmesi ve veritabanı işlemlerinin yapılması

ASPEticaret projesine Default isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
DataGrid – dgEncokSatanlar		

The screenshot shows a website layout with a sidebar on the left containing categories like Klasik, Bilgisayar, Edebiyat, Roman, Sır, Fıkra, Macera, Bilim-Kurgu, Tarih, Sağlık, and S.Tarih. The main content area features a title 'Yazılım Uzmanı Kitap evi', navigation links (AnaSayfa, Üye Giriş, Üye Kayıt), and a DataGrid titled 'En Çok Satanlar' showing book sales data. The DataGrid has columns for 'Kitap Adı' (Book Name) and 'Fiyatı' (Price). The data includes books like ADO .NET (100), Hayat Tüpşüz dalmayacak kadar derin mi? (45), Çalkunu (50), SQL Server (50), and .NET (250). On the right side, there is a logo for 'Bilge Adam' and a link to 'BilgeAdamsd.net club'.

Kitap Adı	Fiyatı
ADO .NET	100
Hayat Tüpşüz dalmayacak kadar derin mi?	45
Çalkunu	50
SQL Server	50
.NET	250

Default Web formun Html kodları aşağıdaki gibi olacaktır.

```
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="cs" AutoEventWireup="false"
Codebehind="Default.aspx.cs" Inherits="AspETicaret._Default"
%>
<%@ Register TagPrefix="uc1" TagName="kategori"
Src="kategori.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Ust" Src="Ust.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
    <HEAD>
        <title>Default</title>
        <meta name="GENERATOR" content="Microsoft Visual
Studio .NET 7.1">
        <meta name="CODE_LANGUAGE" content="C#">
        <meta name="vs_defaultClientScript"
content="JavaScript">
        <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
    </HEAD>
    <body bgColor="honeydew">
        <form id="Form1" method="post" runat="server">
            <TABLE id="Table1" cellspacing="0" cellpadding="0"
width="700" align="center" border="0">
                <TR>
                    <TD bgColor="#99ccff" colspan="3">
```

```
<uc1:Ust id="Ust1"
runat="server"></uc1:Ust></TD>
</TR>
<TR>
    <TD width="150" vAlign="top">
        <uc1:kategori id="Kategori1"
runat="server"></uc1:kategori></TD>
        <TD width="400" vAlign="top">
            <P>
                <TABLE id="Table2" cellSpacing="0"
cellPadding="0" width="350" align="center" border="0">
                    <TR>
                        <TD style="HEIGHT: 55px">
                            <P align="center">En Çok
Satanlar</P>
                        </TD>
                    </TR>
                    <TR>
                        <TD>
                            <DIV align="center">
                                <asp:DataGrid
id="dgEncoksatanlar" runat="server"
AutoGenerateColumns="False" Width="253px" Borderwidth="1px"
BorderColor="#003333">
                                    <HeaderStyle Font-
Bold="True"></HeaderStyle>
                                    <Columns>
                                        <asp:HyperLinkColumn
DataNavigateUrlField="KitapID"
DataNavigateUrlFormatString="Kitapdetay.aspx?KID={0}">
                                            DataTextField="KitapAdi" HeaderText="Kitap Adi">
                                                <HeaderStyle
width="275px"></HeaderStyle>
                                            </asp:HyperLinkColumn>
                                        <asp:BoundColumn
DataField="Ucret" HeaderText="Fiyati">
                                            <HeaderStyle
width="75px"></HeaderStyle>
                                            </asp:BoundColumn>
                                        </Columns>
                                    </asp:DataGrid></DIV>
                            </TD>
                        </TR>
                        <TR>
                            <TD><TD>
                                <TABLE>
                                    <TR>
                                        <TD width="150" bgColor="#0099ff" vAlign="top">
                                            <uc1:yan id="Yan1"
runat="server"></uc1:yan></TD>
                                        </TR>
                                        <TR>
                                            <TD colspan="3" bgColor="#99ccff">
                                                <uc1:Alt id="Alt1"
runat="server"></uc1:Alt></TD>
                                            </TR>
                                        </TABLE>
                                    &nbsp;
                                </form>
                            </body>
                        </HTML>
```

DataSet içerişine DataTable Eklentimesi

12. **Server Explorer** penceresinden **DataConnections** seçeneğini seçin
13. Veritabanı tablo ve sorgularına erişmek için oluşturduğumuz, bağlantı içerisindeki **Views** seçeneğini seçin.
14. **Views** içerisindeki EnCokSatanlar sorgusunu dsBook nesnesinin içerişine sürükleinyin.

Default web formunun Code Behind kodları aşağıdaki gibi olacaktır.

```
using System.Data.OleDb;

private void Page_Load(System.Object sender,
System.EventArgs e)
{
    if(!Page.IsPostBack)
    {
        string connStr =
"Provider=Microsoft.Jet.OleDb.4.0;Data Source=" +
Server.MapPath("KitapDb.mdb");
        OleDbConnection conn = new OleDbConnection();
        conn.ConnectionString = connStr;

        OleDbCommand comm = new OleDbCommand();
        comm.CommandType = CommandType.Text;
        comm.CommandText = "SELECT Kitap.KitapAdi, "+
"Kitap.Ucret, Kitap.KitapID FROM Kitap INNER JOIN Siparis "+
"ON Kitap.KitapID = Siparis.KitapID GROUP BY "+
"Kitap.KitapAdi, Kitap.Ucret, Kitap.KitapID ORDER BY"+
"Count(Siparis.Adet) DESC";

        comm.Connection = conn;

        OleDbDataAdapter da = new OleDbDataAdapter();
        da.SelectCommand = comm;

        DataSet ds = new DataSet();
        try
        {
            conn.Open();
            da.Fill(ds, "EnCokSatanlar");
            dgEncokSatanlar.DataSource =
ds.Tables["EnCokSatanlar"];
            dgEncokSatanlar.DataBind();
        }
        catch(Exception ex)
        {
            Response.Write(ex.Message);
        }
        finally
        {
            if(conn.State == ConnectionState.Open)
            {
                conn.Close();
            }
        }
    }
}
```

}{}

Kitap formunun eklenmesi ve veritabanı işlemlerinin yapılması

ASPEticaret projesine Kitap isminde yeni bir web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

Kontrol – Kontrol İsmi	Özellik	Değer
DataList – dlKitap	RepeatColumns	2



Kitap Web formun Html kodları aşağıdaki gibi olacaktır.

```
<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Ust" Src="Ust.ascx" %>
<%@ Register TagPrefix="uc1" TagName="kategori"
Src="kategori.ascx" %>
<%@ Page Language="cs" AutoEventWireup="false"
Codebehind="Kitap.aspx.cs" Inherits="AspEticaret.Kitap" %>
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<!DOCTYPE HTML PUBLIC "-//w3c//DTD HTML 4.0
Transitional//EN">
<HTML>
    <HEAD>
        <title>Kitap</title>
        <meta content="Microsoft Visual Studio .NET 7.1"
name="GENERATOR">
        <meta content="C#" name="CODE_LANGUAGE">
        <meta content="JavaScript"
name="vs_defaultClientScript">
        <meta
content="http://schemas.microsoft.com/intellisense/ie5"
name="vs_targetSchema">
```

```
</HEAD>
<body bgColor="#f0ffff">
    <form id="Form1" method="post" runat="server">
        <TABLE id="Table1" cellSpacing="0" cellPadding="0"
width="700" align="center" border="0">
            <TR>
                <TD bgColor="#99ccff" colspan="3"><uc1:ust
id="Ust1" runat="server"></uc1:ust></TD>
            </TR>
            <TR>
                <TD width="150" vAlign="top"><uc1:kategori
id="Kategori1" runat="server"></uc1:kategori></TD>
                <TD vAlign="top" align="center" width="400">
                    <asp:datalist id="dlKitap" runat="server"
RepeatColumns="2">
                        <ItemTemplate>
                            <table width="180">
                                <tr align="center">
                                    <td>
                                        <a
href='KitapDetay.aspx?kID=<%#
databinder.eval(Container.dataitem,"KitapID") %>'><img
border=0 src='resimler/<%#
databinder.eval(Container.dataitem,"Image") %>'>
                                        </a>
                                    </td>
                                </tr>
                                <tr align="center">
                                    <td>
                                        <%#
databinder.eval(Container.dataitem,"KitapAdi") %>
                                    </td>
                                </tr>
                                <tr align="center">
                                    <td>
                                        <%#
databinder.eval(Container.dataitem,"Yazar") %>
                                    </td>
                                </tr>
                                <tr align="center">
                                    <td>
                                        <%#
databinder.eval(Container.dataitem,"Ucret") %>
                                    </td>
                                </tr>
                            </table>
                        </ItemTemplate>
                    </asp:datalist>
                </TD>
                <TD width="150" bgColor="#0099ff" vAlign="top">
                    <uc1:yan id="Yan1"
runat="server"></uc1:yan></TD>
            </TR>
            <TR>
                <TD bgColor="#99ccff" colspan="3"><uc1:alt
id="Alt1" runat="server"></uc1:alt></TD>
            </TR>
        </TABLE>
    </form>
</body>
</HTML>
```

DataSet içeresine DataTable Eklenmesi

1. **Server Explorer** penceresinden **DataConnections** seçeneğini seçin
2. Veritabanı tablo ve sorgularına erişmek için oluşturduğumuz, bağlantı içerisindeki **Tables** seçeneğini seçin.
3. **Tables** içerisindeki Kitap tablosunu dsBook nesnesinin içerisinde sürükleyin.

Kitap web formunun Code Behind kodları aşağıdaki gibi olacaktır.

```
using System.Data.OleDb

private void Page_Load(System.Object sender,
System.EventArgs e)
{
    string kategoriID = Request.Params["KategoriID"];
    Session["KategoriID"] = kategoriID;
    //Response.Write(kategoriID)
    string connstr =
"Provider=Microsoft.Jet.OleDb.4.0;Data Source=" +
Server.MapPath("KitapDb.mdb");
    OleDbConnection conn = new OleDbConnection();
    conn.ConnectionString = connStr;

    OleDbCommand comm = new OleDbCommand();
    comm.CommandType = CommandType.Text;
    comm.CommandText = "Select * from Kitap where" +
"KategoriID =@KID";
    comm.Connection = conn;

    comm.Parameters.Add("@KID",
Convert.ToInt32(kategoriID));

    OleDbDataAdapter da = new OleDbDataAdapter();
    da.SelectCommand = comm;

    DataSet ds = new DataSet();
    try
    {
        conn.Open();
        da.Fill(ds, "Kitap");
        dlKitap.DataSource = ds.Kitap;
        dlKitap.DataBind();
    }
    catch(Exception ex)
    {
        Response.Write(ex.Message);
    }
    finally
    {
        if(conn.State == ConnectionState.Open)
        {
            conn.Close();
        }
    }
}
```


ASP.NET ile Durum Yönetimi



Modül 12: ASP.NET ile Durum Yönetimi

ASP.NET ile Durum Yönetimi

- ◆ Durum Yönetimi
- ◆ Session
- ◆ Cookie
- ◆ Application
- ◆ Global.asax

Bu modülde ASP.NET Web uygulamalarında kullanılan durum yönetimi üzerinde durulacaktır. Durum yönetim alt yapısı kullanılarak uygulama seviyesinde veri paylaşımı gerçekleştirilebilir.

Bu modül tamamlandıktan sonra:

- ASP.NET Web uygulamalarında kullanılan durum yönetim alt yapısını tanımlayabilecek,
- **Application** ve **Session** ile web uygulamalarını yönetebilecek,
- **Cookies** ve **Cookieless Session** kavramlarını açıklayabileceksiniz.

Durum Yönetimi

Durum Yönetimi

- Web formları stateless çalışır.
- ASP.NET, sunucuda uygulamaya ait özel bilgileri tutan ve sayfalar arası veri aktarımı gerçekleştiren bir mekanizma sağlar.
- Veriler sunucuya gönderilip geri geldiğinde kullanıcının yeniden veri girişi yapmasına gerek kalmaz.

Web formları **stateless** çalışır. Yani kullanıcılarından gelen isteklerin nereden geldiği anlaşılmaz. Web sunucusuna yapılan her istekte web formlar yeniden oluşturulur.

ASP.NET, sunucuda uygulamaya ait özel bilgileri tutan ve sayfalar arası veri aktarımı gerçekleştiren bir altyapı sağlar.

Durum Yönetimi

- Sunucu taraflı durum yönetimi birden fazla yönetim seçeneğine sahiptir.
 - Application state
 - Session state
- Kullanıcı taraflı durum yönetimi ise genellikle cookie nesneleri ile sağlanır.

Sayfalar arası **state** yönetimi sayesinde sunucuda tutulan bilgiler yeniden kullanılabilir. Böylece veriler sunucuya gönderilip geri geldiğinde kullanıcının yeniden veri girişi yapmasına gerek kalmaz.

Örneğin bir Login sayfasına “Bilge” kullanıcı ismiyle giriş yapıldıktan sonra, diğer sayfalarda “Merhaba Bilge” mesajını verilebilir. Bu mesajı göstermek için “Bilge” kullanıcı adı **State** yönetimi ile bir değişkende tutulmalıdır.

Sunucu taraflı durum yönetimi birden fazla yönetim seçeneği sunar.

- Application state
- Session state

Kullanıcı taraflı durum yönetimi ise genellikle **cookie** nesneleri ile sağlanır.

Konu 1: Session

Session

- Kullanıcı bilgisayarı ve web sunucusu arasında kurulan bağlantıya session denir.
- Kullanıcıya özeldir.
- Sayfalar arası bilgi aktarmak için pratik bir yöntemdir.
- Veritabanına bağlantı kurularak alınan ve uygulama içinde sürekli kullanılan bilgiler session değişkeni içinde tutulur.

Kullanıcı bilgisayarı ve web sunucusu arasında kurulan bağlantıya **session** denir. Bir **session**, birden fazla web sayfasını kapsayabilir. Kullanıcının web uygulamasına girişi ile çıkışı arasında tutulan değişkenlerdir ve bu değişkenler kullanıcıya özeldir.

Session değişkenlerine, uygulama süresince erişilip gerekli bilgiler hızlı bir şekilde kullanılabilir. Sayfalar arası bilgi aktarmak için pratik bir yöntemdir.

Veritabanına bağlantı kurularak alınan ve uygulama içinde sürekli kullanılan bilgiler **session** değişkenleri içinde tutulur.

ASP.NET **session** değişkenlerini yönetirken **HttpSessionState** sınıfını kullanır.

Session

- ASP.NET, session değişkeni kullanılırken HttpSessionState sınıfını kullanır.
- Kullanıcı web sunucusuna bağlanıp bir ASP.NET sayfası görüntülemeyi talep ettiği zaman,
 - Sunucu, kullanıcıya bir SessionID atar.
 - Bu değeri kullanıcıya gönderir.
- Kullanıcı uygulamadan çıkışa kadar bu SessionID değişkeni sunucuda tutulur.

Kullanıcı web sunucusuna bağlanıp bir ASP.NET sayfası görüntülemeyi talep ettiği zaman, sunucu, kullanıcıya bir **SessionID** atar ve bu değeri kullanıcıya gönderir. Kullanıcı uygulamadan çıkışa kadar bu **SessionID** değişkeni sunucuda tutulur.

Kod 12.1: SessionID

```
private void Page_Load(Object sender, EventArgs e)
{
    Response.Write(Session.SessionID);
}
```

Kod 12.2: Session Nesnesini kullanmak

Login.aspx sayfası

```
private void BtnGiris_Click(Object sender, EventArgs e)
{
    Session["ad"] = TxtAd.Text;
    Response.Redirect("Sayfam.aspx");
}
```

Sayfam.aspx sayfası

```
private void Page_Load(Object sender, EventArgs e)
{
    lblAd.Text = Session["ad"] + "Hoş Geldiniz";
}
```

ASP.NET uygulamalarının, **Session** değişkenlerine ait çeşitli özellikler, web.config dosyası içinde tanımlanır.

```
<sessionState
    mode="InProc"
    stateConnectionString="tcpip=127.0.0.1:42424"
    sqlConnectionString="data
    source=127.0.0.1;Trusted_Connection=yes"
    cookieless="false"
    timeout="20"
/>
```

SessionState 'in varsayılan **attribute** değerleri Visual Studio.NET tarafından atanmıştır.

Mode: **session** değerlerinin nerede tutulacağını belirler. **InProc**, değerler IIS içinde saklanır. **StateServer** değerler sunucuda arka planda çalışan ASP.NET **state** servisinde saklanır. **SqlServer**, değerler SQL Server içindeki tablolarda saklar.

Cookieless: Varsayılan olarak **False** değerini alır. **Session** değişkenlerinin kullanıcı bilgisayarında **cookie** içinde tutulmasını belirler. **True** değeri verildiğinde ise **SessionID** değeri URL'ye eklenecek kullanıcıya geri yollanır.

Timeout: **Session** değişkenlerinin yaşam süresini belirler. Varsayılan olarak 20 dakikadır.

Bazı tarayıcıların **cookie** desteği olmadığı düşünüldüğünde, kullanıcıya ait bilgileri **session** değişkenlerinde tutmak daha geçerli olacaktır.

Session Değişkenine İlk Değer Vermek

Global.asax dosyasında, **Session** nesnesinin **Start** olay prosedürü içinde ilk değer verme işlemleri gerçekleştirilebilir.

Kod 12.3' de Session_Start olayınının kullanımı gösterilmektedir.

Kod 12.3: Session_Start

```
void Session_Start(Object Sender, EventArgs e)
{
    Session["ArkaPlan"] = "blue";
    Session["Yazi"] = "gray";
}
```

Konu 2: Cookie

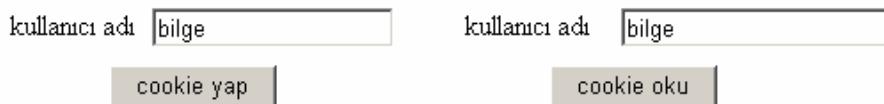
Kullanıcı taraflı durum yönetimi için **cookie** değişkenleri kullanılır. Internet sitelerinin çoğu istemci bilgisayarda **cookie** denilen küçük metin dosyaları

oluşturur. Microsoft XP, Windows 2000 sistemlerinde **cookie** nesneleri "C:\Documents And Settings\Kullanıcı Adı\Cookies" klasöründe saklanır. Bir siteye ilk defa giriş yapıldığında **cookie** oluşur. Daha sonra tekrar giriş yapıldığında **cookie** içindeki değerler okunur ve bu değerlere göre gerekli işlemler yapılır. Örneğin üyelik sistemi içeren web sitelerindeki "Beni Hatırla" seçeneği bu mantıkla çalışır.

ASP.NET **Cookie** değişkenlerini yönetirken **HttpCookie** sınıfını kullanır. **Cookie** değişkenleri için yazma ve okuma işlemleri yapılrken **Response** ve **Request** nesneleri kullanılır.

Örnek:

Kullanıcı adı girilip "Cookie yap" butonuna tıklanınca kullanıcı tarafından bir **cookie** oluşturuluyor. "Cookie oku" butonuna tıklandığında ise oluşturulan **cookie** nesnesinden veri alınıyor.



Şekil 2.1: Cookie kullanımı

Kod 12.4: Cookie oluşturup okumak

```
private void btnYap_Click(System.Object sender,
System.EventArgs e)
{
    // Cookie oluşturmak için verilen direktif.
    HttpCookie mycookie = new HttpCookie("sitem");
    // Formdan Gelen Bilgileri Anahtarlaraya Yazar.
    mycookie["ad"] = txtad.Text;
    // Cookie'nin Bitiş Süresi.
    mycookie.Expires = DateTime.Now.AddDays(30);
    // Cookie'yi Gönder.
    Response.Cookies.Add(mycookie);
}

private void btnoku_Click(System.Object sender,
System.EventArgs e)
{
    ' Cookie'yi oluşturur.
    HttpCookie mycookie;
    ' Cookie'yi kullanıcı tarafından alır.
    myCookie = Request.Cookies("sitem");
    ' Cookie'den gelen değerlerle formu doldurur.
    txtad2.Text = mycookie("ad");
```

{}

Cookie Türleri

İki tür **Cookie** vardır:

- Temporary (Geçici)

Cookie

- **Temporary (Geçici)**
 - Tarayıcının hafızasında tutulur.
 - Tarayıcı kapatıldığında tüm temporary cookie nesneleri kaybolur.

Temporary cookie nesneleri, **session cookie** veya **non-persistent cookie** olarak da isimlendirilir. Bu **cookie** 'ler sadece tarayıcının hafızasında tutulup, tarayıcı kapatıldığında tüm **temporary cookie** nesneleri hafızadan atılırlar.

- Persistent (Kalıcı)

Cookie

▪ Persistent (Kalıcı)

- Hafızadan silineceği zamanı tutan bir değişkene sahiptir.
- Tarayıcı, kalıcı bir cookie isteğinde bulunan bir sayfa açtığında, cookie sabit diske yazılır.
- Bu tür cookie nesneleri kullanıcı bilgisayarında istenilen sürede tutulabilir.
- **Cookie** nesnelerinin diskte tutulacağının garantisini yoktur. Kullanıcı sabit diskinden bu dosyaları silmiş olabilir.

Persistent **cookie** nesneleri, temporary **cookie** nesnelerinden farklı olarak hafızadan silineceği zamanı tutan bir değişkene sahiptirler. Tarayıcı, kalıcı bir **cookie** isteğinde bulunan bir sayfa açtığında, **cookie** sabit diske yazılır. Bu tür **cookie** nesneleri kullanıcı bilgisayarında istenilen sürede tutulabilir.

Cookie nesnelerinin diskte tutulacağının garantisini yoktur. Kullanıcı sabit diskinden bu dosyaları silmiş olabilir.

Konu 3: Application

Application

- Session nesnesine benzer.
- Web uygulamasına giriş yapan ilk kullanıcidan son kullanıcıya kadar devam eder.
- Tüm kullanıcılar ait olan bir değişkendir.
- Application değişkeni kullanılırken lock yapılarak başka kullanıcıların kullanması engellenir.
- Değişken kullanıldıktan sonra unlock yapılmalıdır.

Application nesnesinin tanımlanması **session** nesnesine benzer. Ancak kullanım alanı çok farklıdır. Web uygulamasına giriş yapan ilk kullanıcidan son kullanıcıya kadar devam eder. Tüm kullanıcılar ait olan bir değişkendir. Örneğin sitenin kaç kişi tarafından ziyaret edildiği, **Application** nesnesinde bir değişken tanımlanarak belirlenebilir. **Application** değişkenini kullanırken **Lock** yaparak başka kullanıcıların kullanması engellenir ve değişken ile işiniz sonra **unlock** yapılmalıdır.

Application

- Session kullanıcıya özgü değişkenleri tutarken Application uygulamanın kendisine ait değişkenleri tutar.
- ASP.NET Application değişkeni kullanırken HttpApplicationState sınıfını kullanır.
- Application değişkenine değer atandıktan sonra uygulama içinden çağırmak için, Application["değisen_ismi"] ifadesi kullanılır.

Session kullanıcıya özgü değişkenleri tutarken Application uygulamanın kendisine ait değişkenleri tutar.

ASP.NET Application değişkeni kullanırken HttpApplicationState sınıfını kullanır.

Kod 12.4: Application Değişkeni

```
void Session_Start (Object sender, EventArgs e)
{
    If (Application("ziyaret") == null)
    {
        Application("ziyaret") = 0;
    }
    Application.Lock();
    Application("ziyaret") = Application("ziyaret") + 1;
    Application.UnLock();
    TextBox1.Text = "Ziyaret Sayısı: " +
                    Application("ziyaret").ToString();
}
```

Bu örnekte her bir yeni session açıldığında, yani siteye her istek yapıldığında ziyaretçi sayısı birer artırılmaktadır.

Application değişkeni doldurulduktan sonra uygulama içinden çağırmak için Application("değisen_ismi") ifadesi kullanılır.

Application Değişkenine İlk Değer Vermek

Global.asax dosyasında, Application nesnesinin Start olay prosedürü içinde başlangıç değerleri verilir. Bu olay prosedürü uygulama çalışmaya

başladığında ve ilk istek geldiğinde çalışır. **Application** değişkeni Web uygulaması kaldırıldığında sonlanır.

Kod 12.5'de Application_Start olayının kullanımına örnek verilmiştir.

Kod 12.5: Application_Start

```
void Application_Start(Object sender,EventArgs e)
{
    Application("ziyaret") = 0;
}
```

Konu 4: Global.asax

Global.asax

- Her bir web uygulamasına ait bir global.asax dosyası vardır.
- Global.asax dosyası, web uygulamasına ait sanal dosya içinde saklanır.
- Uygulamaya ait application ve session değişkenlerine ilk değer vermek için kullanılan başlangıç ve bitiş olaylarını tutar.

Sadece sunucu üzerindeki uygulama üzerinde çalışabilen bir dosyadır. Global.asax, ASP.NET web uygulamasının çalıştığı sırada, çeşitli olayları ele alacak bir dosyadır.

Bu dosyanın birçok özelliği vardır.

- Her bir web uygulamasına ait bir global.asax dosyası vardır.
- Global.asax dosyası, web uygulamasına ait sanal dosya içinde saklanır.
- Uygulamaya ait **application** ve **session** değişkenlerine ilk değer vermek için kullanılan başlangıç ve bitiş olaylarını tutar.
- Bu dosyanın tanımlanması isteğe bağlıdır. Eğer bu dosya projede bulunmuyorsa, ASP.NET hiçbir **application** ve **session** olay prosedürü tanımlanmamış varsayar.

Global.asax dosyasında desteklenen olaylar üç kategoride toplanabilir:

Global.asax

- Bu dosyanın tanımlanması isteğe bağlıdır.
- Global.asax dosyasında desteklenen olaylar üç kategoride toplanabilir,
 - Sayfaya bir istekte bulunulduğunda.
 - İstekte bulunan sayfa istemciye yollandığında.
 - Koşullu application olayları gerçekleştiğinde.

- Sayfaya bir istekte bulunulduğunda
- İstekte bulunan sayfa istemciye yollandığında
- Koşullu **application** olayları gerçekleştiğinde

Koşullu **application** olayları ise tablo 12.1' de listelenmiştir.

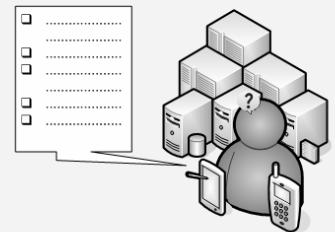
Olay ismi	Açıklama
Application_Start	Uygulamanın ilk çalışmaya başladığında çalışır.
Application_End	Uygulama sona erdiğinde çalışır.
Session_Start	Yeni bir session oluştuğunda çalışır.
Session_End	Session kapandığında çalışır.
Application_Error	Uygulamanın çalışması sırasında bir hata oluştuğunda çalışır.

Tablo 12.1: Koşullu Application olayları

Modül Özeti

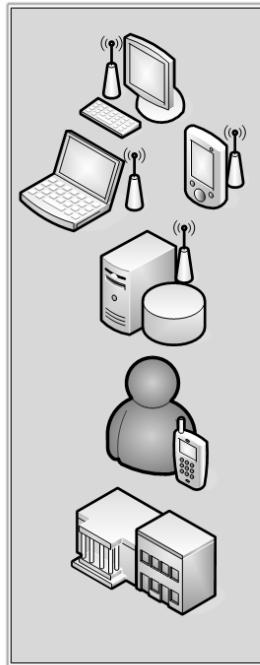
Modül Özeti

- Session için kullanılır?
- Cookie için kullanılır?
- Cookie türleri nelerdir?
- Application için kullanılır?



39. Session için kullanılır?
40. Cookie için kullanılır?
41. Cookie türleri nelerdir?
42. Application için kullanılır?

Lab 1: E-Ticaret Uygulaması Geliştirmek



Uygulamalar ASP.NET ile Durum Yönetimi

- ◆ E-Ticaret Uygulaması
Geliştirmek

Bu uygulamada session nesnesi ile KitapDetay sayfasına erişim engelenecektir. KitapDetay sayfasına sadece sisteme giriş yapan kullanıcılar erişebilecektir.

Bu lab tamamlandıktan sonra:

- Session kullanımını öğreneceksiniz.

Session kullanmak

AspEticaret isimli projeyi açın.

UyeGiris formu içerisinde Session Kullanmak

UyeGiris web formunu açın.

UyeGiris web formunun Code Behind kodları aşağıdaki gibi olacaktır. UyeGiris kod dosyası içerisindeki işaretli satırlar, veri tabanı içerisinde çekilen kayıtların session değişkenlere aktarılmasını sağlar.

```
using System.Data.OleDb

private void btnGiris_Click(System.Object sender,
System.EventArgs e)
{
    ' Session["user"] = "tamer";
```

```
string connStr =
    "Provider=Microsoft.Jet.OleDb.4.0;Data Source="
    + Server.MapPath("KitapDb.mdb");

OleDbConnection conn = new OleDbConnection();
conn.ConnectionString = connStr;

OleDbCommand comm = new OleDbCommand();
comm.CommandType = CommandType.Text;
comm.CommandText = "Select * from Musteri "
    + "where Email=@email and Sifre=@sifre";
comm.Connection = conn;
comm.Parameters.Add("@email", txtEmail.Text);
comm.Parameters.Add("@sifre", txtSifre.Text);
Boolean sonuc;
OleDbDataReader dr;
try
{
    conn.Open();
    dr = comm.ExecuteReader();

    if (dr.HasRows == True)
    {
        sonuc = True;
        if (dr.Read == True )
        {
            Session["user"] = dr.Item("Email");
            Session["ad"] = dr.Item("Ad");
            Session["soyad"] = dr.Item("Soyad");
            Session["musteriId"] = dr.Item("MusteriID");
        }
    }
    else
    {
        sonuc = False;
    }

    dr.Close();
}

catch (Exception ex)
{
    Response.Write(ex.Message);
}
finally
{
    If (conn.State == ConnectionState.Open)
    {
        conn.Close();
    }
}
if (sonuc == True )
{
    Response.Redirect("Default.aspx")
}
else
{
    lblMesaj.Text = "Hatalı kullanici adı veya şifre";
}
}
```

KitapDetay formu içerisinde Session Kullanmak

KitapDetay web formunu açın.

KitapDetay web formunun Code Behind kodları aşağıdaki gibi olacaktır. KitapDetay kod dosyası içerisindeki işaretli satırlar, kullanıcının sisteme girişini kontrol etmektedir. Eğer kullanıcı sisteme giriş yapmadıysa, **user** değişkeni içerisine değer aktarılmaz. Bu durum kullanıcının Giriş.aspx sayfasına yönlendirilmesine sebep olur.

```
using System.Data.OleDb

string kID
    private void Page_Load(System.Object sender,
System.EventArgs e)
    {
        if (Session["user"] == "" )
        {
            Response.Redirect("Giris.aspx");
        }

        kID = Request.Params("kID");
        'Response.Write(kID)
        string connStr =
            "Provider=Microsoft.Jet.OleDb.4.0;Data Source=" +
            "Server.MapPath("KitapDb.mdb")";
        OleDbConnection conn ;
        conn.ConnectionString = connStr;

        OleDbCommand comm = new OleDbCommand();
        comm.CommandType = CommandType.Text;
        comm.CommandText =
            "Select * from Kitap where KitapID =@kitapID" ;
        comm.Connection = conn ;

        comm.Parameters.Add("@kitapID",Convert.ToInt32(kID));
        OleDbDataReader dr = new OleDbDataReader();
        Try
        {
            conn.Open();
            dr = comm.ExecuteReader()
            If (dr.Read == true)
            {
                lblKitapAdi.Text = dr.Item("KitapAdi");
                lblYazarAdi.Text = dr.Item("Yazar");
                lblFiyat.Text = dr.Item("Ucret");
                lblAciklama.Text = dr.Item("Aciklama");
                imgResim.ImageUrl = "resimler/" +
                    dr.Item("Image");
            }
            dr.Close();
        }
        Catch (Exception ex)
        {
            Response.Write(ex.Message);
        }
        finally
```

```
{  
    If (conn.State = ConnectionState.Open)  
    {  
        conn.Close();  
    }  
}
```

Üst kullanıcı kontrolu içerisinde Session kullanmak

Üst kullanıcı kontrolunu açın.

Üst kullanıcı kontrolünün Code Behind kodları aşağıdaki gibi olacaktır. Üst kod dosyası içerisindeki işaretli satırlar, session değişkenin değerini lblAd isimli etikete yazmaktadır.

```
private void Page_Load(System.Object sender,  
System.EventArgs E)  
{  
    if (Session["user"] != "")  
    {  
        lblAd.Text = "Bay / Bayan : " + Session["ad"] + "  
" + Session["soyad"];  
        btnCikis.Visible = true ;  
    }  
    else  
    {  
        btnCikis.Visible = false;  
    }  
}
```

btnCikis butonundaki işaretli kod satırları, tüm session değişkenlerin değerini sıfırlar.

```
private void btncikis_Click(System.Object sender,  
System.EventArgs e)  
{  
    Session.Abandon() ;  
    btnCikis.Visible = false;  
    Response.Redirect("Default.aspx");  
}
```