

# **Yazılım**

# **Profesyoneli 2**

## **Software Professional 2**

Yazan: Fulya SATAR – Engin ÖREN

Editörler: Tamer ŞAHİNER – Tuncer KARAARSLAN

Yayına Hazırlayan: Selçuk TÜZEL

Grafik Uygulama: Zeynep ÇÖMLEKÇİ

Kapak Tasarımı: Selim Şahin

Baskı: Lebib Yalkın

Şef Editör: Mehmet ÇÖMLEKÇİ

1. Baskı: 2005

Copyright © 2005, Bilge Adam

Kitabın yayın hakları Bilge Adam Bilgi Teknolojileri Akademisi'ne aittir.  
Firmadan yazılı izin almadan kısmen veya tamamen alıntı yapılamaz, hiçbir  
şekilde kopya edilemez, çoğaltılamaz ve yayımlanamaz.



## Önsöz

.NET yazılım teknolojisi ile yeni tanışan ya da daha önceden .NET yazılım geliştirme araçlarıyla kısa bir süre çalışma imkanı bulmuş okurlar için temel bir başvuru niteliğindeki bu yayın ile windows ve web tabanlı uygulamalar geliştirebileceksiniz. Veri tabanı programlama ve web programlama konularında sizlerin çok iyi seviyelere gelmesine destek verecektir.

Eğitimcilerinizi yakından takip etmenizi sağlayacak bu yayın ile bol bol örnek kod inceleme fırsatına sahip olacaksınız.

Yazılım profesyonelleri için hazırlanan bu yayında emeği geçen tüm yazılım ekibindeki arkadaşlarıma teşekkür ederim. Yazarlık katkılarından dolayı Fulya SATAR'a, bu yayının hazırlanmasında büyük emek ve fedakarlıklarından dolayı özellikle Tamer ŞAHİNER'e ve yayındaki büyük emeklerinden dolayı Engin ÖREN'e teşekkür ederim. Ayrıca yayının içeriğinin oluşmasında fikirlerini sürekli bizimle paylaşan yazılım ekibindeki eğitimcilerimize teşekkürler.

Tuncer KARAARSLAN

# İçindekiler

## **Modül 1: Geliştirme Ortamını Tanımak .....3**

### **Konu 1: Visual Basic .NET ile Proje Oluşturmak ..... 4**

|                                     |    |
|-------------------------------------|----|
| Proje Şablonu Seçmek .....          | 5  |
| Proje Dosyalarına Genel Bakış.....  | 7  |
| Assembly Nedir?.....                | 9  |
| Projeye Referans Ekleme .....       | 10 |
| İsim Alanı (Namespace) Nedir? ..... | 12 |
| Yeni İsim Alanı Ekleme .....        | 13 |
| Projeye İsim Alanı Dahil Etmek..... | 15 |
| Proje Özelliklerini Ayarlamak.....  | 17 |

### **Konu 2 : Proje Bileşenlerini Tanımak ..... 19**

|                                  |    |
|----------------------------------|----|
| Solution Explorer Kullanmak..... | 20 |
| Object Browser Kullanmak.....    | 21 |
| Server Explorer Kullanmak.....   | 22 |
| Dinamik Yardım Almak .....       | 23 |
| Görev Listesini Kullanmak.....   | 24 |

### **Konu 3: Uygulamalarda Hata Ayıklama ..... 25**

|                                 |    |
|---------------------------------|----|
| BreakPoint .....                | 26 |
| Debug Panelleri .....           | 27 |
| Command Panelini Kullanmak..... | 28 |

### **Konu 4: Uygulamanın Derlenmesi..... 29**

|                                  |    |
|----------------------------------|----|
| Derleme Seçeneklerine Bakış..... | 30 |
|----------------------------------|----|

### **Modül Özeti ..... 31**

### **LAB 1: Geliştirme Ortamını Tanımak..... 32**

|   |    |
|---|----|
| Uygulama 1: Windows Uygulaması Oluşturmak ..... | 32 |
| Çağrı Merkezi Uygulamasını Oluşturmak .....     | 32 |
| Uygulama 2: Object Browser Kullanmak .....      | 33 |
| System.Data Kütüphanesini Açmak .....           | 33 |
| Uygulama 3: Debug Aracını Kullanma.....         | 33 |
| Kodların Yazılması .....                        | 33 |
| Hata Ayıklama.....                              | 34 |

## **Modül 2: Veri Merkezli Uygulamalar ve ADO.NET'e Giriş. 37**

### **Konu 1: Veri Merkezli Uygulamalar ..... 38**

|   |    |
|---|----|
| Veri Depolama.....                              | 39 |
| Bağlantılı (Connected) Veri Ortamları .....     | 40 |
| Bağlantısız (Disconnected) Veri Ortamları ..... | 41 |
| Veri Erişim Yöntemleri .....                    | 43 |

### **Konu 2: ADO.NET'e Giriş..... 46**

|                                  |    |
|----------------------------------|----|
| ADO.NET Nedir? .....             | 47 |
| ADO.NET Nesne Modeli .....       | 48 |
| ADO.NET Veri Sağlayıcıları ..... | 49 |

### **Modül Özeti ..... 53**

### **Lab 2: Veri Merkezli Uygulamalar ve ADO.NET'e Giriş ..... 54**

|   |    |
|---|----|
| Uygulama 1: Yeni bağlantı oluşturmak.....                   | 54 |
| Çağrı Merkezi Uygulaması İçin Yeni Bağlantı Oluşturmak..... | 54 |

## **Modül 3: Veri Kaynaklarına Bağlanmak..... 59**

### **Konu 1: Veri Sağlayıcı Seçmek..... 60**

|                                |    |
|--------------------------------|----|
| Veri Sağlayıcı Nedir? .....    | 61 |
| Veri Sağlayıcı Sınıfları ..... | 62 |

### **Konu 2: Bağlantı Oluşturmak..... 66**

|  |    |
|--|----|
| Bağlantı Cümlesi (Connection String) Oluşturmak .....  | 67 |
| Bağlantı Cümlesini (Connection String) Kullanmak ..... | 69 |
| Bağlantı Cümlesi(Connection String) Örnekleri .....    | 71 |
| Ms Access ile OLEDB Bağlantı Cümleleri .....           | 72 |
| SQL Server ile ODBC Bağlantı Cümleleri.....            | 73 |
| SQL Server ile OLEDB Bağlantı Cümleleri.....           | 74 |
| SQL Server ile SQL Server Bağlantı Cümleleri .....     | 75 |

### **Konu 3: Bağlantı Yönetimi ..... 76**

|   |    |
|---|----|
| Bağlantıyı Açmak ve Kapatmak.....       | 77 |
| Bağlantı Durumlarını Kontrol Etmek..... | 80 |

### **Modül Özeti ..... 82**

### **Lab 1: Bağlantı Oluşturmak ..... 83**

|                             |    |
|-----------------------------|----|
| Kontrollerin Eklenmesi..... | 83 |
| Kodların Yazılması .....    | 84 |

**Modül 4: Bağlantılı (Connected) Veritabanı İşlemleri .....91****Konu 1: Bağlantılı Veri Ortamlarıyla Çalışmak..... 92**

Bağlantılı Uygulamalar İçin Veritabanı Mimarisi..... 93

**Konu 2: Command ile Çalışmak..... 95**

Command Nedir?..... 96

Command Oluşturmak ..... 99

Parametre Kullanmak ..... 101

**Konu 3: Command ile Geriye Değer Döndürmek ..... 105****Konu 4: Command ile Geriye Kayıt Döndürmek ..... 107**

DataReader Özellik ve Metotları ..... 107

**Konu 5: Command ile Kayıt Döndürmeyen Sorgular Çalıştırmak ..... 113****Modül Özeti ..... 117****Lab 1: Veritabanı İşlemleri..... 118**

Veritabanının Oluşturulması ..... 118

Kontrollerin Eklenmesi ..... 119

Kodların Yazılması ..... 119

ExecuteNonQuery Metodu ..... 120

ExecuteReader ve DataReader ..... 121

Form Kontrolleri İşlemleri..... 122

Yordamların Formda Kullanılması..... 123

**Modül 5: Bağlantısız (Disconnected) Veritabanı İşlemleri129****Konu 1: Disconnected Uygulamalar İçin Veritabanı Mimarisi ..... 130****Konu 2: DataSet ve DataTable Oluşturmak ..... 132**

DataSet Nesne Modeli..... 133

**Konu 3 : DataAdapter ile Kayıtları Dataset'e Doldurmak..... 137****Konu4: DataSet Nesnesini Kontrollere Bağlamak..... 139**

DataSet İçindeki Veriyi Windows Kontrollerine Bağlamak ..... 140

DataSet İçindeki Veriyi DataGrid'e Bağlamak ..... 142

**Konu : 5 DataTable Üzerindeki Veriyi Düzenlemek..... 143**

Windows Form ile Kayıt Üzerinde Hareket Sağlamak ..... 146

|   |                |
|---|----------------|
| <b>Lab 1: Bağlantısız Veritabanı İşlemleri .....</b>            | <b>148</b>     |
| Veritabanının Oluşturulması .....                               | 148            |
| Kontrollerin Eklenmesi .....                                    | 149            |
| Bağlantı Cümlesinin Oluşturulması .....                         | 150            |
| Bağlantının Oluşturulması.....                                  | 150            |
| DataAdapter Nesnesinin Oluşturulması.....                       | 151            |
| DataSet Nesnesinin Oluşturulması.....                           | 151            |
| DataSet İçindeki Verinin DataGrid Kontrolüne Bağlanması.....    | 151            |
| DataSet İçindeki Verinin TextBox Kontrollerine Bağlanması ..... | 152            |
| Kodların Yazılması .....  | 152            |
| <br><b>Konu 6: Veri Arama ve Sıralama .....</b>                 | <br><b>154</b> |
| DataView Özellik ve Metotları.....                              | 157            |
| <br><b>Modül Özeti .....</b>                                    | <br><b>159</b> |
| <br><b>Lab 2: Çoklu Tablolarla Çalışmak.....</b>                | <br><b>160</b> |
| Veritabanının Projeye Eklenmesi .....                           | 161            |
| Kontrollerin Eklenmesi .....                                    | 161            |
| Bağlantı Cümlesinin Oluşturulması .....                         | 161            |
| Bağlantının Oluşturulması.....                                  | 162            |
| DataAdapter Nesnesinin Oluşturulması.....                       | 162            |
| DataSet Nesnesinin Oluşturulması.....                           | 164            |
| DataView Nesnesinin Oluşturulması.....                          | 164            |
| DataSet İçindeki Verinin ComboBox Kontrolüne Bağlanması.....    | 165            |
| Kodların Yazılması .....  | 165            |
| <br><b>Modül 6: ASP.NET'e Giriş .....</b>                       | <br><b>169</b> |
| <br><b>Konu 1: ASP.NET Nedir? .....</b>                         | <br><b>170</b> |
| <br><b>Konu 2: ASP Tarihçesi.....</b>                           | <br><b>171</b> |
| <br><b>Konu 3: ASP.NET Uygulama Mimarisi.....</b>               | <br><b>172</b> |
| İstemci Katmanı (Presentation Tier) .....                       | 172            |
| İş katmanı (Business Logic Tier).....                           | 172            |
| Veri Katmanı (Data Tier) .....                                  | 172            |
| <br><b>Konu 4: ASP.NET Çalışma Modeli.....</b>                  | <br><b>173</b> |
| Tür Yönetimi (Type Management) .....                            | 174            |
| JIT Derleme (JIT Compilation) .....                             | 175            |
| Hafıza Yönetimi (Memory Management) .....                       | 176            |
| Exception Yöneticisi (Exception Manager) .....                  | 177            |

|   |            |
|---|------------|
| <b>Konu 5: ASP.NET'in .NET Çatısındaki Yeri .....</b>           | <b>180</b> |
| <b>Konu 6: .NET Framework'ün ASP.NET'teki Avantajları .....</b> | <b>181</b> |
| <b>Konu 7: ASP.NET ile Uygulama Geliştirmek.....</b>            | <b>182</b> |
| IIS Nedir? .....  | 183        |
| IIS Kurulumu ve Yönetimi .....                                  | 184        |
| IIS Kurulumu .....  | 185        |
| IIS Yönetimi.....   | 187        |
| .NET Framework Kurulumu .....                                   | 190        |
| <b>Modül Özeti .....</b>  | <b>191</b> |
| <b>Lab 1: Web Tabanlı Uygulamaların Yayınlanması .....</b>      | <b>192</b> |
| IIS (Internet Information Services) Kurulması .....             | 192        |
| Uygulama Yayınlamak .....                                       | 192        |
| <b>Modül 7: ASP.NET Web Form ve Kontrolleri ile Çalışmak</b>    | <b>197</b> |
| <b>Konu 1: Web Form Bileşenleri .....</b>                       | <b>198</b> |
| Page Özelliği.....  | 200        |
| Body Özelliği .....   | 202        |
| Form Özelliği .....   | 203        |
| <b>Konu 2: Server (Sunucu) Kontroller .....</b>                 | <b>204</b> |
| <b>Konu 3: Kontrollerin Sınıflandırılması .....</b>             | <b>205</b> |
| Standart Kontroller .....                                       | 205        |
| Doğrulama Kontrolleri .....                                     | 206        |
| Zengin Kontroller .....   | 206        |
| İlişkisel Liste Tabanlı Kontroller .....                        | 206        |
| <b>Konu 4: Standart Kontroller .....</b>                        | <b>207</b> |
| Label .....   | 207        |
| TextBox .....   | 207        |
| Button .....  | 207        |
| CheckBox .....  | 208        |
| RadioButton .....   | 209        |
| HyperLink .....   | 210        |
| Image .....   | 210        |
| ImageButton .....   | 210        |
| LinkButton .....  | 211        |
| DropDownList .....  | 212        |
| ListBox .....   | 213        |



|   |            |
|---|------------|
| Panel .....   | 214        |
| Table .....   | 214        |
| <b>Konu 5: Doğrulama(Validation) Kontrolleri .....</b>              | <b>216</b> |
| RequiredFieldValidator .....  | 217        |
| CompareValidator .....  | 217        |
| RangeValidator .....  | 218        |
| RegularExpressionValidator .....                                    | 219        |
| CustomValidator .....   | 220        |
| ValidationSummary .....   | 222        |
| <b>Konu 6: Zengin Kontroller .....</b>                              | <b>223</b> |
| AdRotator .....   | 223        |
| Calendar .....  | 224        |
| <b>Konu 7: AutoPostBack Özelliği .....</b>                          | <b>226</b> |
| <b>Konu 8: ViewState .....</b>                                      | <b>228</b> |
| <b>Modül Özeti .....</b>  | <b>229</b> |
| <b>Lab 1: E-Ticaret Uygulaması Geliştirmek .....</b>                | <b>230</b> |
| Veritabanının Projeye Eklenmesi .....                               | 230        |
| Web Formların Eklenmesi .....                                       | 231        |
| UyeKayit Formunun Eklenmesi .....                                   | 231        |
| UyeGiris Formunun Eklenmesi .....                                   | 236        |
| Giris Formunun Eklenmesi .....                                      | 238        |
| Kayit Formunun Eklenmesi .....                                      | 240        |
| Satis Formunun Eklenmesi .....                                      | 242        |
| <b>Modül 8: ASP.NET ile Kod Geliştirmek .....</b>                   | <b>247</b> |
| <b>Konu 1: Kod Yazmak .....</b>                                     | <b>248</b> |
| Inline Kod Yazmak .....   | 249        |
| Code-Behind Kod Yazmak .....  | 250        |
| <b>Konu 2: Client Side (İstemci Tarafı) Olay Prosedürleri .....</b> | <b>252</b> |
| <b>Konu 3: Server Side (Sunucu Tarafı) Olay Prosedürleri .....</b>  | <b>253</b> |
| Olay Prosedürleri Oluşturmak .....                                  | 254        |
| Olay Prosedürlerinde Kontrollerle Etkileşim .....                   | 255        |
| <b>Konu 4: Sayfa Yaşam Döngüsü .....</b>                            | <b>256</b> |
| Response.Redirect .....   | 257        |
| PostBack İşlemleri .....  | 258        |

|  |            |
|--|------------|
| Page.IsPostBack.....                               | 259        |
| <b>Modül Özeti .....</b>                           | <b>260</b> |
| <b>Lab 1: ASP.Net ile Kod Geliştirmek.....</b>     | <b>261</b> |
| Web Uygulaması Oluşturmak .....                    | 261        |
| Web Form Eklenmesi .....                           | 261        |
| Kodların Yazılması .....                           | 262        |
| <b>Modül 9: Web Programlamaya Giriş.....</b>       | <b>267</b> |
| <b>Konu 1 : Web Programlamaya Giriş .....</b>      | <b>268</b> |
| <b>Konu 2: HTML.....</b>                           | <b>269</b> |
| HTML Yapısı .....                                  | 269        |
| Tag .....  | 270        |
| Attribute .....                                    | 271        |
| Value.....   | 272        |
| HTML Belgesi Nasıl Oluşturulur?.....               | 273        |
| En Sık Kullanılan Etiketler.....                   | 274        |
| Başlıklar .....                                    | 274        |
| Paragraf ve Satır Sonu .....                       | 274        |
| Sayfalara Bağlantı Vermek .....                    | 275        |
| Listeler .....                                     | 275        |
| Resim Görüntüleme .....                            | 276        |
| Tablolar.....                                      | 276        |
| <b>Konu 3: Script Nedir? .....</b>                 | <b>278</b> |
| JavaScript .....                                   | 279        |
| Değişkenler .....                                  | 281        |
| Operatörler .....                                  | 282        |
| Klavyeden Bilgi Almak ve Ekrana Çıktı Vermek ..... | 284        |
| Koşul ve Döngü Yapıları.....                       | 284        |
| Fonksiyonlar .....                                 | 286        |
| JavaScript Nesneleri .....                         | 286        |
| Olaylar.....                                       | 288        |
| VbScript .....                                     | 291        |
| <b>Konu 4: CSS.....</b>                            | <b>294</b> |
| İç (Inline).....                                   | 294        |
| Gömülü (Embedded).....                             | 294        |
| Bağlantılı (Linked).....                           | 295        |
| Style Sheet'lerin Söz Dizimi .....                 | 296        |
| Seçiciler .....                                    | 298        |

|  |            |
|--|------------|
| Linkler ve CSS .....   | 298        |
| Sınıf ve Gruplama.....   | 299        |
| <b>Modül Özeti .....</b>   | <b>300</b> |
| <b>Lab 1: Web Programlamaya Giriş .....</b>                            | <b>301</b> |
| Web Uygulaması Oluşturmak .....  | 301        |
| Sanal Klavye Oluşturmak .....  | 301        |
| Kodların Yazılması .....   | 304        |
| Popup Pencere Oluşturmak .....   | 306        |
| Kodların Yazılması .....   | 307        |
| <b>Modül 10: Kullanıcı Kontrolleri Oluşturmak .....</b>                | <b>311</b> |
| <b>Konu 1: Kullanıcı Kontrolleri .....</b>                             | <b>312</b> |
| Kullanıcı Kontrolünün Avantajları .....                                | 312        |
| Kullanıcı Kontrolünü Projeye Ekleme .....                              | 313        |
| <b>Modül Özeti .....</b>   | <b>317</b> |
| <b>Lab 1: E-Ticaret Uygulaması Geliştirmek .....</b>                   | <b>318</b> |
| Kullanıcı Kontrollerin Eklenmesi .....                                 | 318        |
| Ust Kontrolünün Eklenmesi .....  | 318        |
| Alt Kontrolünün Eklenmesi .....  | 320        |
| Yan Kontrolünün Eklenmesi .....  | 321        |
| DataSet Nesnesinin Oluşturulması .....                                 | 322        |
| Bağlantı Oluşturulması .....   | 322        |
| Kategori Kontrolünün Eklenmesi .....                                   | 322        |
| <b>Modül 11: ADO.NET ile Veriye Erişim .....</b>                       | <b>327</b> |
| <b>Konu 1: Veri Bağlantılı Kontroller .....</b>                        | <b>328</b> |
| CheckBoxList ve RadioButtonList Kullanımı .....                        | 329        |
| Repeater, DataList ve DataGrid Kullanımı .....                         | 331        |
| Repeater .....   | 332        |
| DataList .....   | 335        |
| DataGrid .....   | 338        |
| DataGrid Kontrolünde Kolon Oluşturmak.....                             | 340        |
| DataGrid Kontrolünde Sıralama ve Sayfalama.....                        | 349        |
| Placeholder Kullanımı .....  | 351        |
| <b>Konu 2: Connected ve Disconnected Uygulamalar Geliştirmek .....</b> | <b>353</b> |
| Namespace .....  | 355        |

|  |            |
|--|------------|
| <b>Modül Özeti .....</b>   | <b>356</b> |
| <b>Lab 1: E-Ticaret Uygulaması Geliştirmek .....</b>                     | <b>357</b> |
| Connect Veritabanı İşlemleri .....                                       | 357        |
| UyeKayıt Formu ile Veritabanı İşlemlerinin Yapılması .....               | 357        |
| UyeGiris Formu ile Veritabanı İşlemlerinin Yapılması .....               | 358        |
| KitapDetay Formunun Eklenmesi ve Veritabanı İşlemlerinin Yapılması ..... | 360        |
| Disconnect Veritabanı İşlemleri .....                                    | 365        |
| Default Formunun Eklenmesi ve Veritabanı İşlemlerinin Yapılması .....    | 365        |
| DataSet İçine DataTable Eklenmesi .....                                  | 367        |
| Kitap Formunun Eklenmesi ve Veritabanı İşlemlerinin Yapılması .....      | 368        |
| DataSet İçine DataTable Eklenmesi .....                                  | 371        |
| <b>Modül 12: ASP.NET ile Durum Yönetimi .....</b>                        | <b>375</b> |
| Durum Yönetimi .....   | 376        |
| <b>Konu 1: Session .....</b>   | <b>378</b> |
| Session Değişkenine İlk Değer Vermek .....                               | 380        |
| <b>Konu 2: Cookie .....</b>  | <b>381</b> |
| Cookie Türleri .....   | 382        |
| <b>Konu 3: Application .....</b>   | <b>385</b> |
| Application Değişkenine İlk Değer Vermek .....                           | 386        |
| <b>Konu 4: Global.asax .....</b>   | <b>388</b> |
| <b>Modül Özeti .....</b>   | <b>390</b> |
| <b>Lab 1: E-Ticaret Uygulaması Geliştirmek .....</b>                     | <b>391</b> |
| Session Kullanmak .....  | 391        |
| UyeGiris Formu İçinde Session Kullanmak .....                            | 391        |
| KitapDetay Formu İçinde Session Kullanmak .....                          | 393        |
| Ust Kullanıcı Kontrolü İçinde Session Kullanmak .....                    | 394        |

# Modül 1: Geliştirme Ortamını Tanımak

## Geliştirme Ortamını Tanımak

- Visual Basic .NET ile Proje Oluşturmak
- Proje Bileşenlerini Tanımak
- Uygulamalarda Hata Ayıklama
- Uygulamanın Derlenmesi

Bu modülde, Visual Studio .NET ortamı ile tanışacak ve bu ortam içinde kullanılan temel proje bileşenleri hakkında genel bilgiler edineceksiniz. 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 içine görevler ekleyebilecek
- Çalışma zamanı hatalarını yakalayabilecek,
- Uygulamaları derleyebileceksiniz.

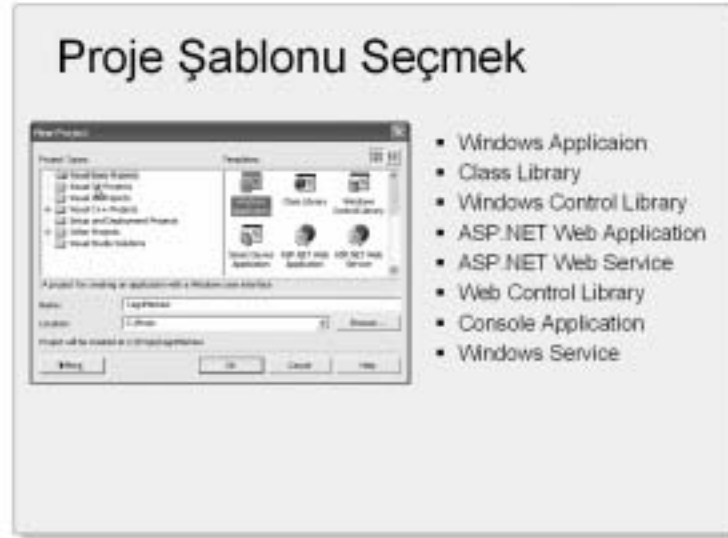
# Konu 1: Visual Basic .NET ile Proje Oluşturmak

## Visual Basic.NET ile Proje Oluşturmak

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

Visual Studio ortamı, Visual Basic .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 gerçekleştirilir.

## Proje Şablonu Seçmek



Visual Basic .NET ile Windows tabanlı ve Web tabanlı gibi çeşitli projeler geliştirilebilir. Bu projeler farklı platformlarda çalışacağı veya farklı amaçlara yönelik oluşturulacağı için, başlangıç bileşenleri farklılık gösterir. Örneğin, Windows tabanlı projeler için Windows formlarının kullanılması ve bazı referansların eklenmesi gerekir. 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 dahil edilir.

Visual Studio ile yeni bir proje birkaç adımda oluşturulabilir.

1. File menüsünden New alt menüsünü işaretleyin ve Project komutunu seçin.
2. New Project penceresinde Visual Basic Projects tipini ve çalışmak istediğiniz şablonunu seçin.
3. Name özelliğinde projeye vereceğiniz ismi yazın.
4. Location özelliği projenin dosyalarının bulunacağı yeri belirler. Browse düğmesini tıklayarak Windows dizinine ulaşın ve projenin yerini seçin.
5. More düğmesi tıklandığında, solution dosyası için yeni bir isim kullanılmasını 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 düğmesi tıklandığında 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 (.vbproj, .vbproj.user)
- Yerel Proje Dosyaları (.vb)
- Web Projesi Dosyaları (.aspx, .asmx, .asax)

Visual Basic .NET ile oluşturulan 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 Basic .NET projeleri bir solution dosyası (.sln) altında oluşturulur. Solution dosyası farklı projeleri bir arada tutar ve birden fazla projeyi içinde 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 (.vbproj, .vbproj.user).** Bir projenin içinde bulunan bileşenlerin, eklenen referansların tutulduğu proje dosyasıdır. Visual Basic projeleri .vbproj uzantılı dosya ile oluşturulur. Bu dosya aynı zamanda, bir solution içinde farklı dilde ve tipteki projeleri ayırt etmek için kullanılır. Projeye özgü ayarlar ise .vbproj.user dosyasında tutulur.
- **Yerel Proje Dosyaları (.vb).** Form, class, module gibi bileşenlerin tutulduğu dosyalardır. .vb uzantılı bir dosya içinde birden çok class ve module tutulabilir. Ancak projedeki her form için ayrı bir .vb 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 oluşturulduktan sonra yeni nesnelerin eklenmesi Project menüsü ile ya da Solution Explorer paneli kullanılarak gerçekleştirilir. Project menüsünden yeni bir form, module, class, component ya da 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 içinde dosyaya ait başlık, açıklama ve telif hakkı gibi kritik bilgiler tutulur.

Visual Studio .NET içinde 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 Ekleme

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

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

## İsim Alanı (Namespace) Nedir?

### İsim Alanı Nedir ?

Sınıflar, arayüzler ve modüller isim alanları kullanılarak gruplandırılır.

#### **System.Data.dll içinde;**

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

.NET içindeki 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 içinde sınıflar, arayüzler ve modüller bulunur.

.NET içinde veritabanı uygulamaları geliştirmek için **System.Data.dll** kütüphanesine ihtiyaç duyulur. Bu kütüphane Visual Studio .NET içindeki tüm proje şablonlarında otomatik olarak yer alır. **System.Data.dll** kütüphanesi içinde şu isim alanları bulunur:

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

## Yeni İsim Alanı Ekleme

### Yeni İsim Alanı Ekleme

Yeni isim alanı oluşturmak için;  
*Namespace Isimalani\_ismi*

...  
*End Namespace*  
söz dizimi kullanılır.

```
Namespace Egitim
Class Grup
...
End Class
End Namespace
```

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

```
Namespace Isimalani_ismi
...
End Namespace
```

Örnekte **NSBilgeAdam** isminde bir isim alanı tanımlanmıştır. Bu isim alanı içine **Egitim** isminde bir sınıf eklenmiştir.

```
Namespace NSBilgeAdam

' BilgeAdam isim alanında kullanılacak
' Sınıf, Modul ve Arayüzler tanımlanır

Class Egitim
' ...
End Class

Class Ogrenci
' ...
End Class

' vs...
End Namespace
```

**NSBilgeAdam** isim alanı içindeki **Ogrenci** sınıfını kullanmak için, sınıf ismi, isim alanı ile birlikte belirtilmelidir.

```
Dim yeniOgrenci As New bilgeadam.NSBilgeadam.Ogrenci()
```



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

Herhangi bir isim alanı içinde birden fazla isim alanı tanımlanabilir. Örnekte **NSBilgeAdam** isim alanı içinde **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

        End Class
    End Namespace

    Namespace Egitim

        Class Grup

        End Class
    End Namespace

    Namespace Ogrenci
        Class Bilgi

        End Class
    End Namespace
    'vs...
End Namespace
```



## Projeye İsim Alanı Dahil Etmek



Bir isim alanı içinde 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.

```
Dim kisi1 As New bilgeadam.NSBilgeadam.Idari.Personel
Dim OgrenciBilgi As New bilgeadam.NSBilgeadam.Ogrenci.Bilgi
```

Her sınıf için kütüphane yolunun tekrarını ortadan kaldırmak için, **Imports** anahtar sözcüğü kullanılır. **Imports** sözcüğü ile eklenen isim alanlarının nesnelere, proje içinden doğrudan erişilebilir.

Örnekte **NSBilgeAdam** isim alanının projeye dahil edilmesi gösterilmektedir:

```
Imports bilgeadam.NSBilgeadam
```

**NSBilgeAdam** isim alanında bulunan bir sınıfı kullanmak için sadece ismini yazmak yeterli olur:

```
Dim ogrenciBilgi As New Ogrenci.Bilgi
```

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

```
Imports ogr = bilgeadam.NSBilgeadam.Ogrenci
```

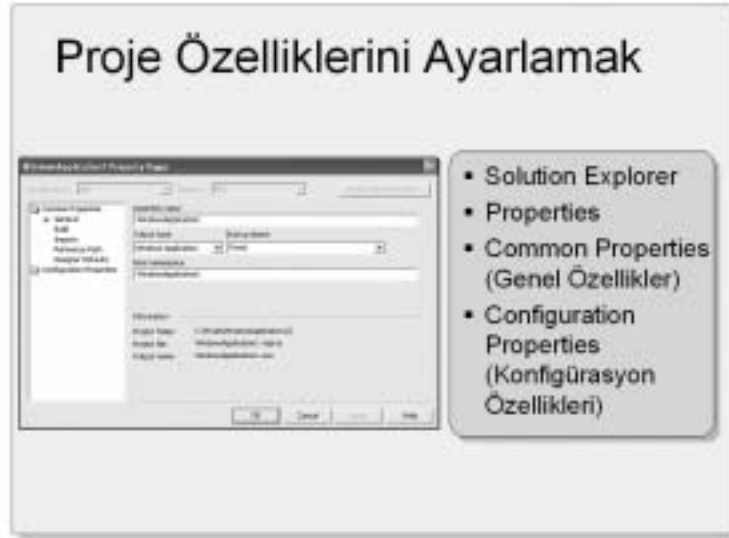
```
Public Class Form1
```

```
Inherits System.Windows.Forms.Form
' ...

Dim OgrBilgi As New ogr.Bilgi

End Class
```

## Proje Özelliklerini Ayarlamak



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

Proje özelliklerini değiştirmek için aşağıdaki adımları takip edin:

1. Proje ismini sağ tıklayın.
2. Açılan menüden Properties komutunu verin.
3. Açılan Property Page penceresi üzerinde Common Properties (Genel Özellikler) ve Configuration Properties (Konfigürasyon Özellikleri) sekmelerinden 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 düğmesini tıklayın.

En çok kullanılan proje özellikleri şunlardır:

## Proje Özelliklerini Ayarlamak

### Genel Özellikler

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

### Konfigürasyon Özellikleri

|                |   |                          |
|----------------|---|--------------------------|
| ▪ Debug, Build | - | Hata Ayıklama ve Derleme |
|----------------|---|--------------------------|

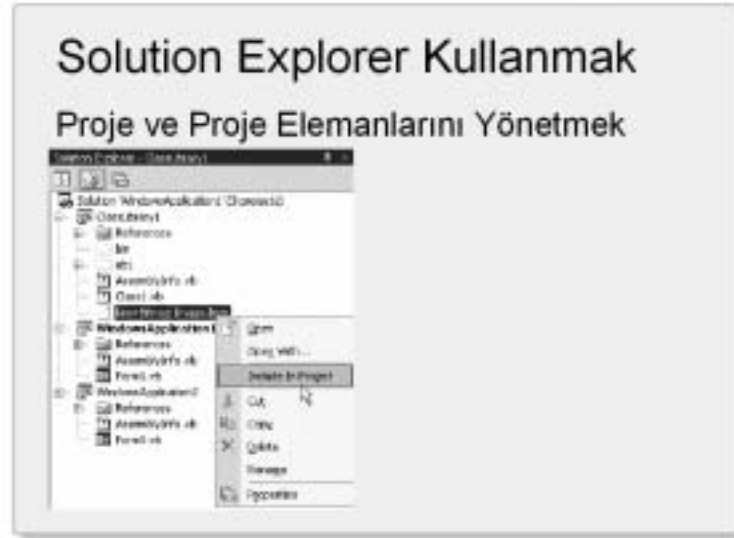
- **Assembly Name:** Derlenen uygulamanın .exe veya .dll uzantılı çıktı dosyasının adını belirler.
- **Root Namespace:** Kök isim alanını belirler. Varsayılan durumda projenin ismi gelir.
- **Project Output Type:** Derlenen uygulamanın hangi tipte assembly oluşturacağını belirler. Bu tipler Windows,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

## Konu 2 : Proje Bileşenlerini Tanımak

### Proje Bileşenlerini Tanımak

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

## Solution Explorer Kullanmak



Solution Explorer paneli, bir solution içindeki tüm dosyaları görüntüler. Solution içinde birden fazla proje bulunabildiği için, bu projeler sıralı bir şekilde listelenir. Koyu renkle gösterilen proje, solution içinde ilk çalıştırılacak projedir.

Bu panel ile solution içine proje ekleme ve silme, projelere yeni nesne ekleme ve silme işlemleri gerçekleştirilir.

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

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

## Object Browser Kullanmak



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

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

Object Browser penceresinin sol üst köşesinde Browse alanı Selected Components seçeneği ile birlikte varsayılan olarak görünür. Bu seçenek ile projeye dahil edilen referanslar ve bu referanslarla ilişkili isim alanları hiyerarşik bir şekilde listelenir.

Objects paneli içinden seçilen 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 içinde 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 seçeneği kullanılır.

Yeni bir veri sağlayıcı oluşturmak için belirtilen adımları takip edin.

1. Server Explorer üzerinden Data Connections seçeneğini işaretleyin.
2. Data Connections seçeneğini sağ 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 sürükleyip bırakarak form üzerine sürüklenebilir.







## Konu 3: Uygulamalarda Hata Ayıklama

### Uygulamalarda Hata Ayıklama

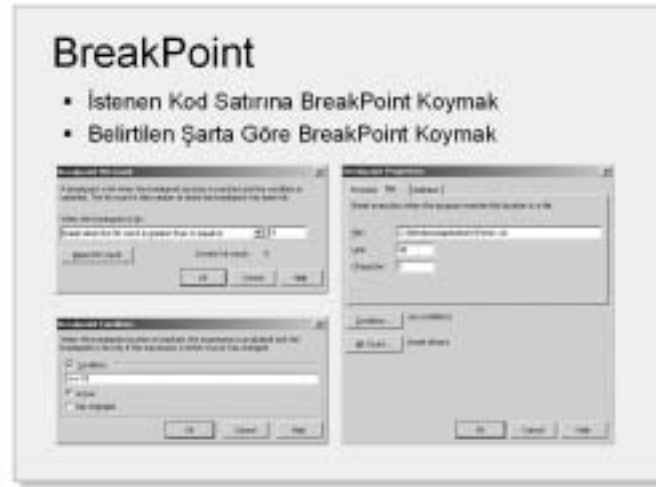
- BreakPoint
- Debug Panelleri
- Command Panelini Kullanmak

Uygulamaların geliştirilmesi sırasında birçok hata ile karşılaşılır. Bu hataların çoğu çalışma zamanında ortaya çıktığı için, kodun yazılması sırasında hatanın kaynağının anlaşılması zordur. Hata üreten kod satırlarını ve 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 panel tıklanır ya da **F9** tuşuna basılır.

Belirtilen bir şart gerçekleştikten sonra çalışmanın durması isteniyorsa, BreakPoint sağ tıklanıp BreakPoint Properties komutu verilmelidir. Çıkan pencerede Condition düğmesi tıklanarak BreakPoint Condition penceresi açılır. Bu pencerede bir değişkenin istenen bir değeri aldıktan sonra çalışmanın durması belirtilir (Resim 1.1).



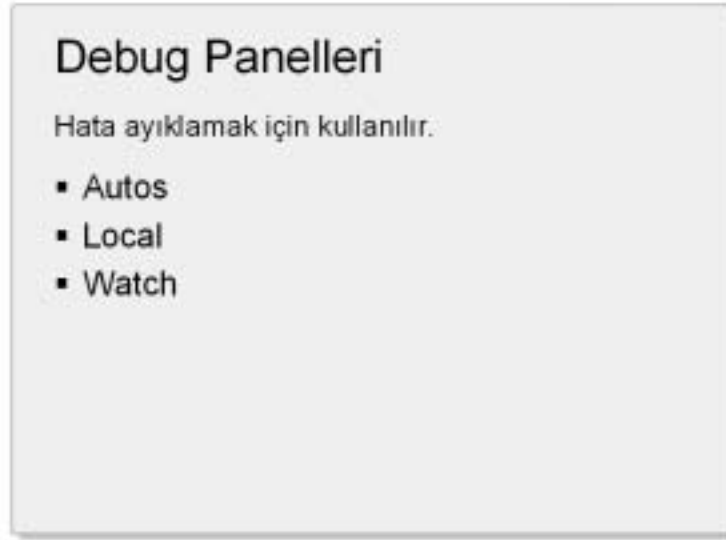
**RESİM 1.1.**

Çalışmanın, şartın belli bir sayı kadar sağlandığı zaman durdurulması için, BreakPoint Properties penceresinde Hit Count düğmesi tıklanır. BreakPoint Hit Count penceresinde, şartın gerçekleşme sayısı girilir. Örnekte, BreakPoint'e beş defa veya daha fazla ulaşıldığı zaman durulması belirtilir (Resim 1.5).



**RESİM 1.2.**

## Debug Panelleri



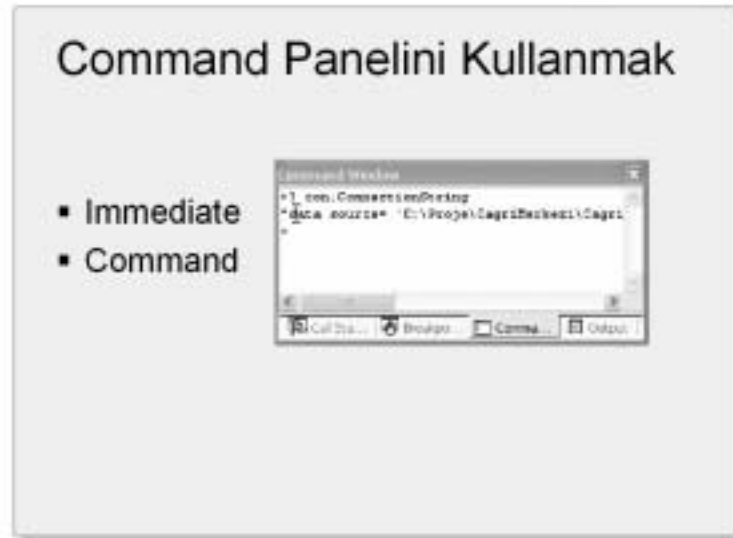
Çalışma durdurulduktan sonra, değişkenlerin o andaki durumları Debug panelleri ile gözlemlenir. Bu paneller ancak ata ayıklama sırasında kullanılabilir. Debug panelleri, Debug menüsü altında Windows menüsünden seçilebilir.

- **Autos:** Çalışmakta olan satırda, 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 gerekir. 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 çıkılarak yordam veya fonksiyonun çağırıldığı 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 metotlar veya kullanıcı tanımlı metotlar çalıştırılabilir. Immediate moduna geçmek için `immed` komutu kullanılır. Immediate modunda bir değişkenin değeri `?` ile öğrenilebilir 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.

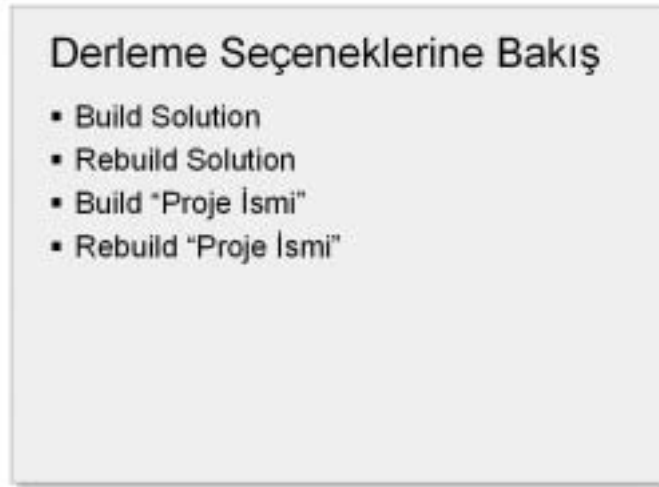
```
>cmd
>Debug.StepOver
>Help.About
```

## Konu 4: Uygulamanın Derlenmesi

### Uygulamanın Derlenmesi

- Derleme Seçeneklerine Bakış

## Derleme Seçeneklerine Bakış



Visual Basic .NET ile geliştirilen uygulamalar çalıştırılmadan önce derleme işleminden geçer. Derleme işlemi ile kodun Visual Basic söz dizimine uygun yazılıp yazılmadığı kontrol edilir ve kod çalıştırılmak üzere makine diline çevrilir.

Uygulamaların 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 hatalar çift tıklanarak, hatanın yapıldığı satıra ulaşılır.



**RESİM 1.3.**



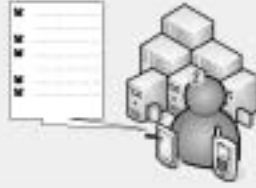
Visual Basic .NET ile uygulama geliştirirken yapılan söz dizimi hataları hemen Task List paneline yansır. Buna Background Compiling denir.



## Modül Özeti

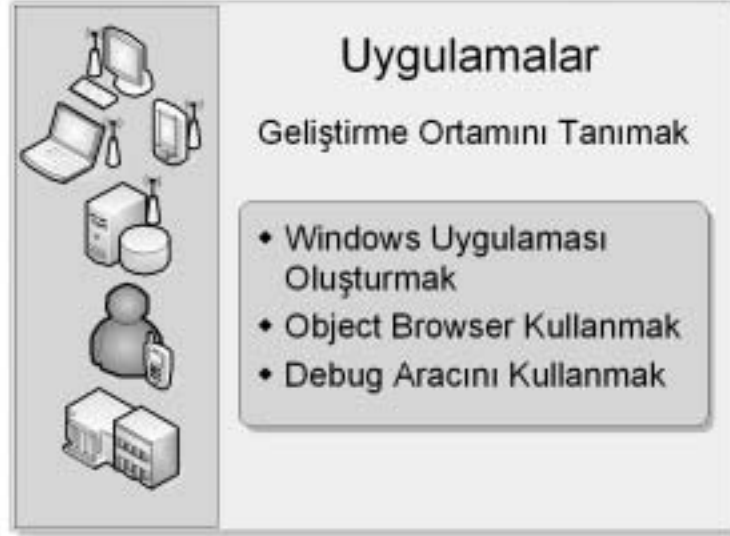
**Modül Özeti**

- 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?

## LAB 1: Geliştirme Ortamını Tanımak



### Uygulama 1: Windows Uygulaması Oluşturmak

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

#### Çağrı Merkezi Uygulamasını Oluşturmak

1. Visual Studio .NET'i kullanarak Çağrı Merkezi isminde uygulama oluşturmak.
  - File menüsündeki New alt menüsünü işaretleyin ve Project komutunu tıklayın.
  - New Project iletişim kutusunda Windows Application şablonunu seçin.
  - Name metin kutusuna **ÇağrıMerkezi** yazın.
  - Location metin kutusuna **C:\Proje** yazın ve OK düğmesini tıklayın.
2. Uygulamanın Assembly Name özelliğini **Çağrı** olarak değiştirmek.
  - Proje ismini sağ tıklayın.
  - Açılan menüden Properties komutunu verin.
  - Açılan penceredeki Common Properties (Genel Özellikler) klasörünü seçin.
  - Common Properties klasörü altındaki General sekmesinde Assembly Name metin kutusuna **Çağrı** yazın ve OK düğmesini tıklayın.

## Uygulama 2: Object Browser Kullanmak

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

### System.Data Kütüphanesini Açmak

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

## Uygulama 3: Debug Aracını Kullanma

Bu uygulamada çağrı merkezi veritabanına bağlantı açan kodlar Debug kullanarak incelenir.

### Kodların Yazılması

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

```
Private Function ConnectionOlustur(ByVal connectionString As String) As OleDb.OleDbConnection
    Return New OleDb.OleDbConnection(connectionString)
End Function
```

2. Formu çift tıklayarak **Load** olayına gelin. **Load** olayında, veritabanına bağlantı açan kodları yazın. Bu veritabanı **C:\Proje\CagriMerkezi** klasörü altında bulunacaktır.

```
Dim con As OleDb.OleDbConnection

' Connection oluşturan fonksiyon çağrılır
con = ConnectionOlustur( _
    "data source= 'C:\Proje\CagriMerkezi\CagriMerkezi.mdb';" &
    Provider = Microsoft.Jet.OleDB.4.0")

con.Open()

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

con.Close()
```

## Hata Ayıklama

1. Formun **Load** olayına bir BreakPoint yerleştirin.
2. Projeyi **F5** tuşuna basarak çalıştırın. Başlangıç formu yüklendiği zaman Load olayı çalışacağı için, çalışma belirtilen noktada durur.
3. Debug menüsünden Step Into komutunu seçerek ya da **F11** tuşuna basarak kod içinde ilerleyin. **Connection01** isimli metodun içine girildiği görülür.
4. **Connection01** metodundan çıkıldıktan sonra, Locals panelini açın ve **con** isimli değişkeni inceleyin.
5. Command panelini açın ve **immed** komutunu yazarak, Immediate moduna geçin.
6. Command paneline **msgbox(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 satıra BreakPoint koyun ve sağ tıklayarak BreakPoint Properties komutunu verin.
10. Condition düğmesini tıklayın ve metin kutusuna **con.State = 1** yazın. **con** nesnesinin **State** özelliği 1 (bağlantı açık) olduğu zaman çalışmanın durması ayarlanır.
11. OK düğmesini tıklayın ve projeyi çalıştırın. Çalışmanın belirtilen noktada durmadığı gözükür. Bunun nedeni, BreakPoint içinde verilen şartın sağlanmamasıdır.
12. Formu kapatarak uygulamayı sonlandırın ve Visual Studio ortamından çıkın.

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



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;

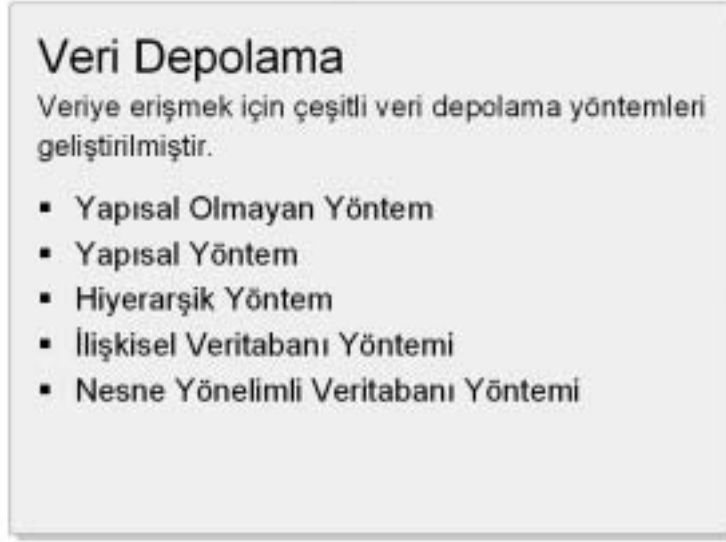
- Veri depolama yöntemlerini öğrenecek,
- Bağlantılı ve bağlantısız veri ortamlarını öğrenecek,
- Veri erişim yöntemlerini öğrenecek,
- ADO.NET nesne modelini öğrenecek,
- ADO.NET nesne modelinde veri sağlayıcılarını seçebileceksiniz.

## Konu 1: Veri Merkezli Uygulamalar

### Veri Merkezli Uygulamalar

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

## Veri Depolama



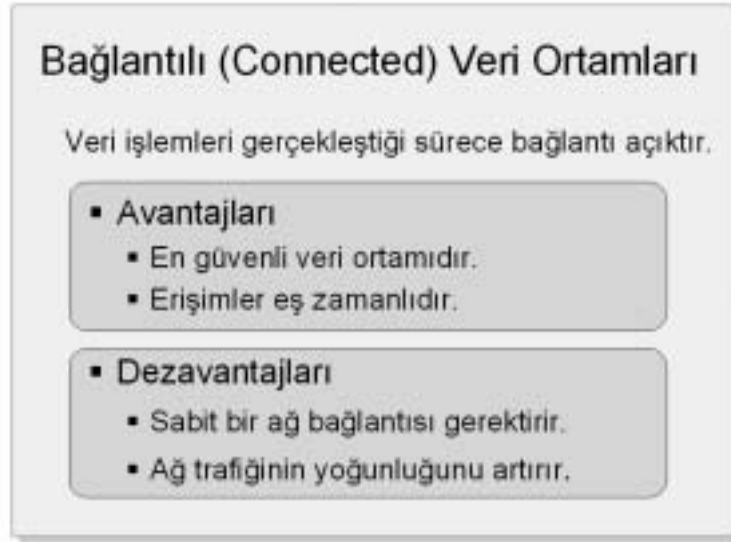
Günümüzde verileri saklamak için çeşitli teknikler kullanılır. Örneğin bir emlakçı emlak alım-satım bilgilerini dosya kağıtları üzerinde depolayabilir. Bu yöntem veri arama ve listeleme işlemlerinin karmaşık hale gelmesine ve arama süresinin uzamasına sebep olur. Hatta daha büyük organizasyonlarda işlemlerin yavaşlamasına ve durmasına sebep olabilir.

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

- **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 gruplara ayrılarak saklanır, fakat bu gruplar arasında bir alt-üst ayrımı yapılmaz. Örneğin virgülle ayrılmış dosyalar (csv), Excel belgeleri.
- **Hiyerarşik:** Hiyerarşik depolama yöntemini ağaç yapısına benzetebiliriz. Bu yöntemde veriler çeşitli kategorilere bölünerek depolanır. Her bir kategorinin içinde alt kategorilerde olabilir. Örneğin XML (eXtensible Markup Language) dosyaları.
- **İlişkisel Veritabanı:** İlişkisel veritabanlarında veriler tablolar üzerinde depolanır. Tablo içindeki her satır kaydı, her 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 içinde saklanır. Örneğin Versant, AOL.

ADO.NET bu depolama tekniklerinin tümünü destekler.

## Bağlantılı (Connected) Veri Ortamları



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.

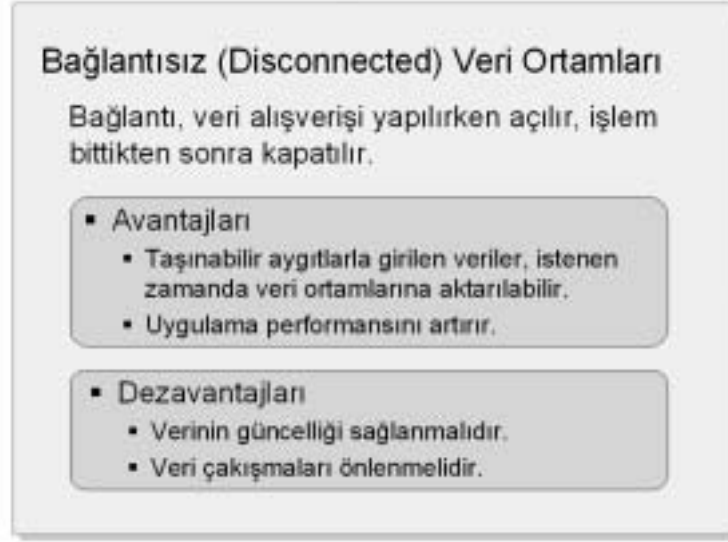
### Dezavantajları

- Uygulama ile veri kaynağı arasında gerçekleşen bağlantıyı koruyabilmek için sabit bir ağ bağlantısının olması gerekir.
- 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 veri ortamı, uygulamanın veri kaynağına sürekli bağlı kalmadığı veri ortamıdır. Uygulama ile veri kaynağı arasındaki bağlantı, veri alışverişi yapılırken 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

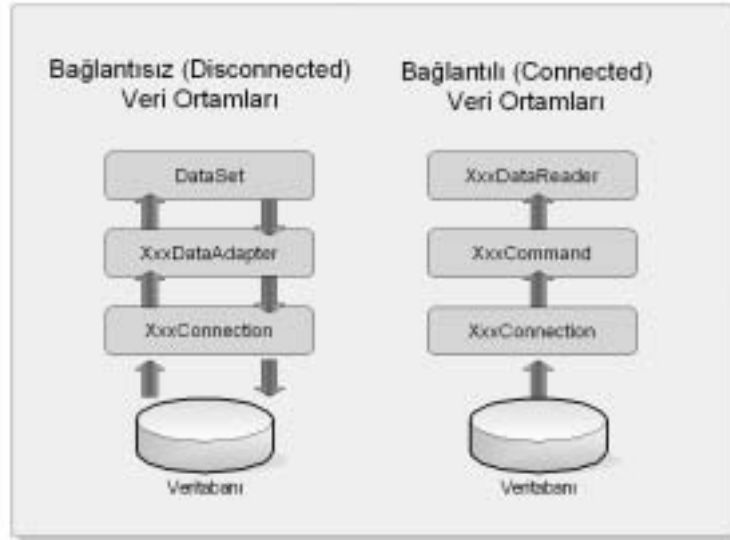
- Laptop, Notebook ve Pocket PC gibi araçlarla girilen veriler, istenilen zamanda veri ortamlarına aktarılabilir.
- Çevrimdışı ortamlar sayesinde, verilerin depolandığı uygulama üzerindeki yük hafifletilir. Bu durum performans artışını sağlar.

### Dezavantajları

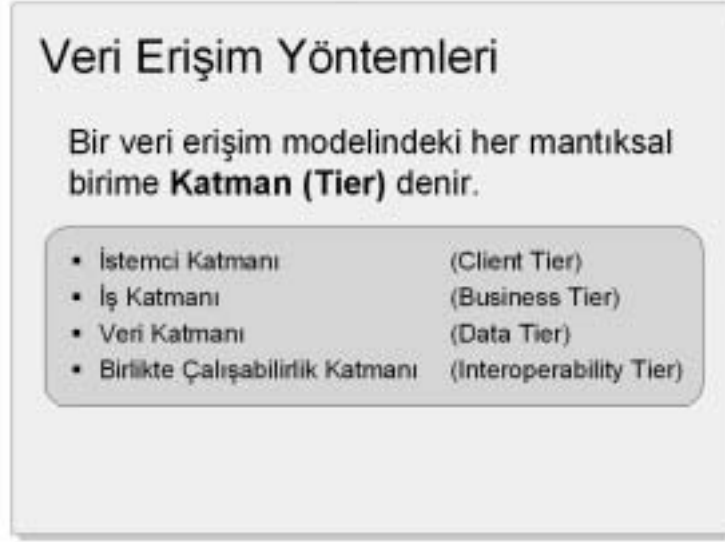
- Bağlantısız veri ortamlarında, verilerin güncel kalmasına dikkat edilmelidir. Bu ortamlarda veri güncelleme işlemleri farklı zamanlarda gerçekleştirilebilir. Veri üzerinde yapılan bu değişimlerin diğer kullanıcılara gösterilebilmesi için çeşitli çözümler geliştirilmelidir.

- Bağılantısız veri ortamları içinde farklı kullanıcılar eşzamanlı güncelleme işlemleri gerçekleştirebilir. Bu durumda oluşacak veri çakışmalarının engellenmesi gerekir.

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



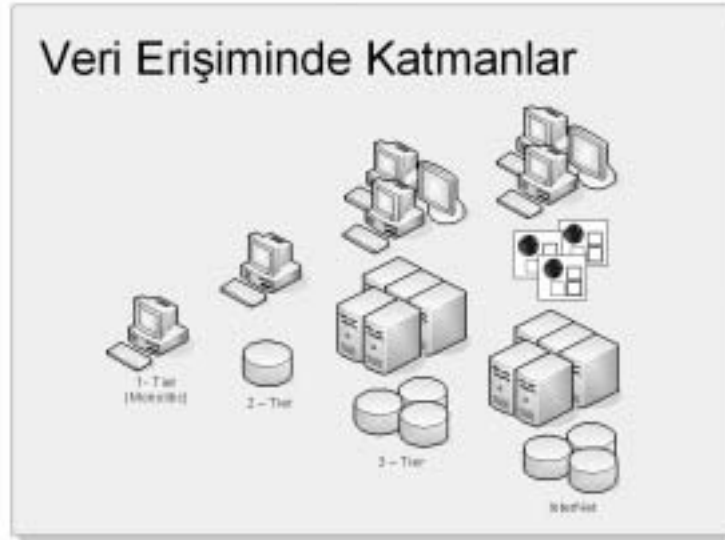
İlk bilgisayardan bugüne veriye erişmek için pek çok yöntem geliştirilmiştir. Bu yöntemlerin bazılarında amaç yerleşim, bazılarında ise paylaşım olmuştur. Amacın veriyi saklamak olduğu durumlarda paylaşım konusunda çözüm aranmış, amacın veriyi birçok kullanıcı arasında paylaşmak olduğu durumda ise ana 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 desteğine 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 her mantıksal 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 arayüzünü içerir.
- **İş Katmanı (Business Tier):** Bu katman, uygulamanın veri kaynağı ile etkileşen bölümüdür.
- **Veri Katmanı (Data Tier):** Veriyi içeren katmandır.
- **Birlikte Çalışabilirlik Katmanı (Interoperability Tier):** Platform ve dil-den bağımsız, her tür veriye etkileşim sağlayan katmandır. Bu katmana herhangi bir işletim sistemi üzerinde bulundurulabilen XML Web Servislerini örnek verebiliriz.



Uygulamalar katmanlara bölünerek ölçeklenebilirlikleri artırılır.



**Katman sayısı arttıkça, veri erişim modelinin ölçeklendirebilirliği ve karmaşıklığı da artar.**

## Konu 2: ADO.NET'e Giriş

### 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 formatını kullanır.
  - N-tier uygulamaları kullanarak uygulamaların farklı katmanlara dağıtımını sağlar.
  - Web uygulamaları için sistem kaynakları korunur.

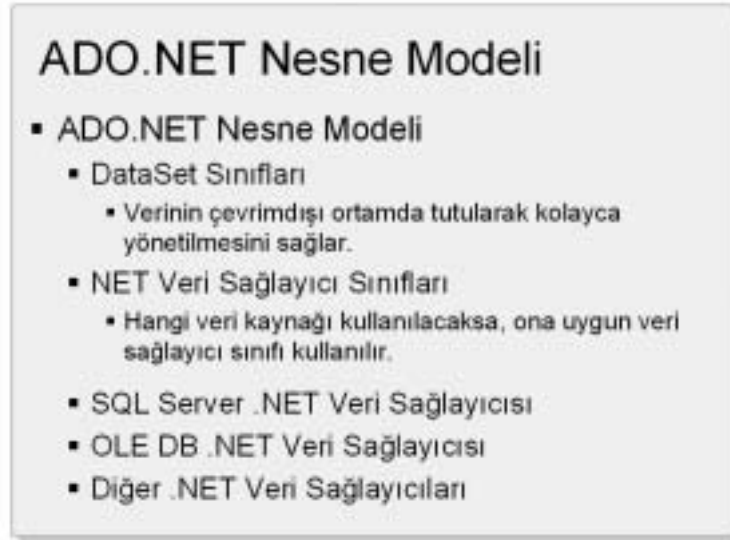
ADO (ActiveX Data Objects), farklı veri kaynaklarına hızlı ve güvenli erişim için Microsoft tarafından geliştirilen nesne modelidir. ADO.NET ise ADO teknolojisinin en yeni versiyonudur. ADO ile aynı programlama modelini kullanmamakla birlikte, ADO modelinden gelen pek çok çözüm yolunu da beraberinde 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 yönelim 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 artar. N-katmanlı (N-tier) uygulama yapısı kullanılarak, uygulamaların katmanlar üzerinden dağıtılması 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 iki ana bölümden oluşur.

- **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



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



## ADO.NET Veri Sağlayıcıları



.NET veri sağlayıcıları, ADO.NET mimarisinin veritabanı ile uygulama (Windows, Web) veya XML Web Servisi 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 veritabanlarına 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.

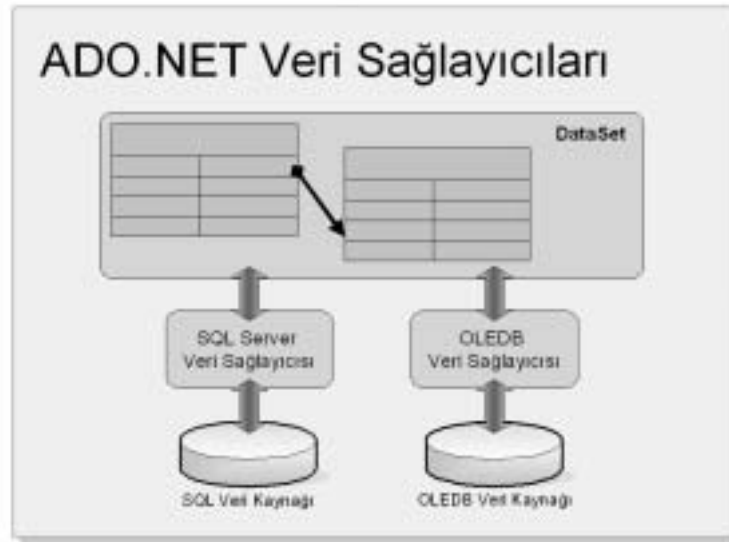
- **ORACLE .NET:** Oracle veritabanlarına bağlantı için tasarlanmış veri sağlayıcısıdır. Oracle bağlantı nesneleri **System.Data.OracleClient** isim alanında bulunur.



**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 sağlayıcıların 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.



Her veri sağlayıcısı içinde birçok bağlantı nesnesi bulunur:

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

## ADO.NET Veri Sağlayıcıları

Her veri sağlayıcısı aşağıdaki nesneleri içerir.

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

- **XxxConnection:** Veri kaynağına bağlantı için kullanılan sınıftır.
- **XxxCommand:** Veri kaynağı üzerinde sorgu çalıştırmak için kullanılır. Veri kaynağından dönen kayıtlar **XxxDataReader** veya **DataSet** kullanılarak veri bağlantılı kontrollere aktarılır.
- **XxxDataReader:** Çevrimiçi bağlantılarda sadece veri okumak için kullanılan sınıftır.
- **XxxDataAdapter:** Çevrimdışı bağlantılarda kullanılan veri işleme nesnesidir.

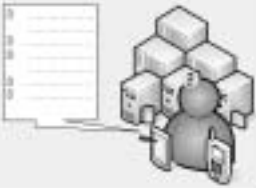


Xxx yerine seçilen veri sağlayıcısına göre SQL, OLEDB, Oracle ve ODBC eklerinden biri kullanılır.

## 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?



- Look in açılan kutusundan **C:\Proje\ CagriMerkezi** klasörünü seçin.
  - Açılan Open Project penceresinden **CagriMerkezi.sln** dosyasını seçerek Open düğmesini tıklayın.
2. Uygulamaya **CagriMerkezi** veritabanını eklemek.
- Proje ismi sağ tıklayın.
  - Açılan menüden Add alt menüsündeki Add Existing Item komutunu verin.
  - Açılan pencere üzerindeki Look in açılan kutusundan **CD Sürücüsü\Veritabanı** klasörünü seçin.
  - Açılan Add Existing Item penceresinden **CagriMerkezi.mdb** veritabanını seçerek Open düğmesini tıklayın.
3. **CagriMerkezi** uygulaması için yeni bağlantı oluşturmak.
- Server Explorer penceresi üzerinde sağ tıklayın. Açılan menüden Add Connection komutunu verin.
  - Açılan Data Link Properties penceresinin Provider sekmesini tıklayın.
  - Provider sekmesinden Microsoft.Jet.OLEDB.4.0 Provider seçeneğini işaretleyerek Next düğmesini tıklayın.



**RESİM 2.2.**

- Açılan Connection sekmesinin görüntüsünü resimdeki gibi düzenleyerek OK düğmesini tıklayın.





## 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 kaynağına bağlanmak için, kaynağın tipine ve yapısına göre farklı nesneler ve farklı veri sağlayıcıları kullanılır.

Bu modülün sonunda;

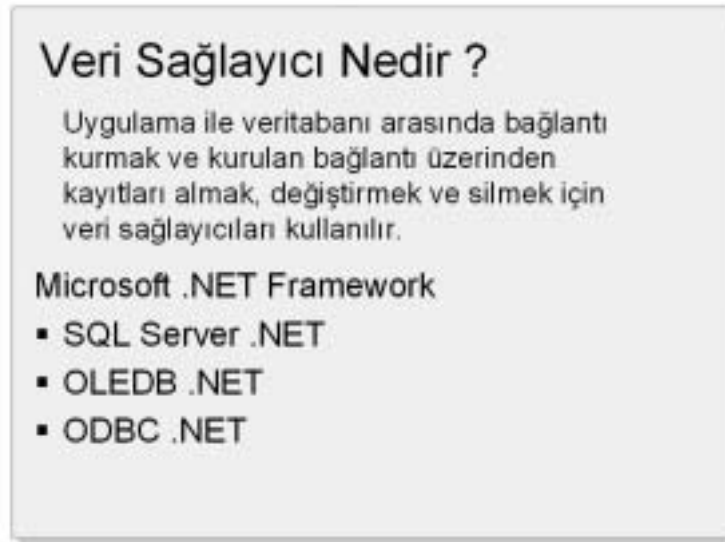
- Farklı veritabanlarına göre veri sağlayıcıları seçebilecek,
- Bağlantı cümlesi oluşturabilecek,
- Farklı veritabanları için **Connection** nesnelerini yönetebileceksiniz.

## Konu 1: Veri Sağlayıcı Seçmek

### Veri Sağlayıcı Seçmek

- Veri Sağlayıcı Nedir?
- Veri Sağlayıcı Sınıfları

## Veri Sağlayıcı Nedir?



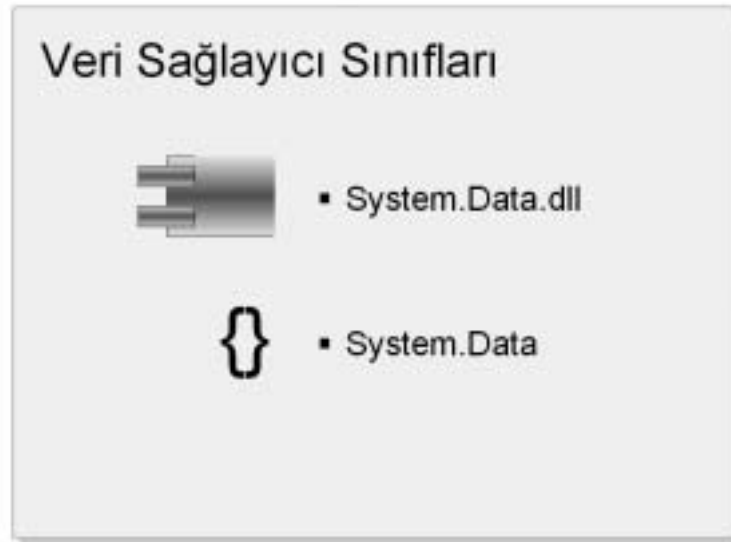
ADO.NET mimarisi, uygulama ile veritabanı arasında bağlantı kurmak ve kurulan bağlantı üzerinden kayıtları almak, değiştirmek ve silmek için veri sağlayıcılarını kullanır. Farklı veritabanları için farklı veri sağlayıcıları kullanılır.

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



.NET Framework içindeki veri sağlayıcıları, **System.Data.dll** içindeki **System.Data** isim alanında yer alır. Tablo 3.1’de hangi sağlayıcı isim alanı ile hangi veritabanına bağlanılabileceği gösterilmektedir.

| Veri Sağlayıcı Sınıfları   |                           |
|--|---------------------------|
| Uygulamalarımız için hangi veri sağlayıcısı en iyi performansı verir ? |                           |
| Veri Tabanı  | Veri Sağlayıcı İsim Alanı |
| Sql Server 7.0 ve sonraki sürümler                                     | System.Data.SqlClient     |
| Sql Server 6.5 ve önceki sürümler                                      | System.Data.OleDb         |
| Microsoft Access veritabanı  | System.Data.OleDb         |
| Oracle Server  | System.Data.OracleClient  |
| Diğer veritabanları  | System.Data.OleDb         |
| Class  | Veri Kaynağı              |
| System.Data.SqlClient.SqlConnection                                    | SQL Server                |
| System.Data.OleDb.OleDbConnection                                      | OleDb veri sağlayıcı      |
| System.Data.Odbc.OdbcConnection  | ODBC veri sağlayıcı       |
| System.Data.OracleClient.OracleConnection                              | Oracle                    |

**TABLO 3.1: Veritabanları ve Veri Sağlayıcı İsim Alanları**

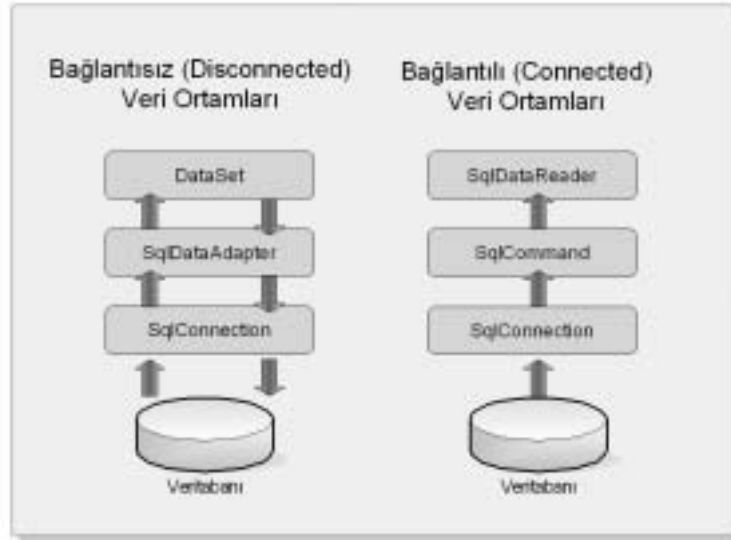
| Veritabanı                                  | Veri Sağlayıcısı İsim Alanı     |
|---|---------------------------------|
| Sql Server 7.0 ve sonraki sürümler          | <b>System.Data.SqlClient</b>    |
| Sql Server 6.5 ve önceki sürümler           | <b>System.Data.OleDb</b>        |
| Microsoft Access veritabanı                 | <b>System.Data.OleDb</b>        |
| Oracle Server                               | <b>System.Data.OracleClient</b> |
| Diğer veritabanları(Oracle, Sybase,DB2/400) | <b>System.Data.OleDb</b>        |

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 3.2'de belirtilen sınıfları kullanır. 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 bağlanılırsa **SQL** ön ekini alır.

**Tablo 3.2: Veri Sağlayıcı Sınıf İsimleri**

| Sınıf                 | Açıklama   |
|-----------------------|--|
| <b>XXXConnection</b>  | Bağlantı açmak ve kapatmak için kullanılan sınıftır.   |
| <b>XXXCommand</b>     | Veritabanı üzerinde Stored Procedure (Saklı Yordamlar) veya SQL cümleleri çalıştırmak için kullanılan sınıftır.  |
| <b>XXXDataReader</b>  | Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılan sınıftır.  |
| <b>XXXDataAdapter</b> | Veritabanından çekilen verileri <b>DataSet</b> içine veya <b>DataSet</b> 'e çevrimdışı eklenmiş verileri veritabanına aktarmak için kullanılan sınıftır. |



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

- **SqlConnection**; MS SQL Server üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **SqlCommand**; MS SQL Server üzerinde Stored Procedure (Saklı Yordamlar) veya SQL cümleleri çalıştırmak için kullanılan sınıftır.
- **SqlDataReader**; MS SQL Server üzerinde **SqlCommand** ile çalıştırılan **SELECT** sorgularının sonuçlarını geri döndürmek için kullanılan sınıftır.

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

- **SqlConnection**; MS SQL Server üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **SqlDataAdapter**; MS SQL Server'dan çekilen verileri **DataSet** içine ve **DataSet**'e çevrimdışı eklenmiş verileri MS SQL Server'a aktarmak için kullanılan sınıftır.
- **DataSet**; **SqlDataAdapter** nesnesinden gelen kayıtları çevrimdışı depolamak ve yönetmek için kullanılan sınıftır. **DataSet** tüm veri sağlayıcı sınıflar için ortaktır.



**DataSet**, **System.Data** isim alanı içinde yer alır.

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

- **OleDbConnection**; Access veya diğer veritabanları üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.

- **OleDbCommand;** Access veya diğer veritabanları üzerinde Stored Procedure (Saklı Yordamlar) veya SQL cümleleri çalıştırmak için kullanılan sınıftır.
- **OleDbDataReader;** Access veya diğer veritabanları üzerinde **OleDbCommand** ile çalıştırılan SELECT sorguların sonuçlarını geri döndürmek için kullanılan sınıftır.

**System.Data.OleDb** isim alanı içinden çevrimdışı bağlantılar geliştirmek için **OleDbConnection** ve **OleDbDataAdapter** sınıfları kullanılır.

- **OleDbConnection;** Access veya diğer veritabanları üzerinde bağlantı açmak ve kapatmak için kullanılan sınıftır.
- **OleDbDataAdapter;** Access veya diğer veritabanlarından çekilen verileri **DataSet** içine ve **DataSet**'e çevrimdışı eklenmiş verileri ilgili veritabanına aktarmak için kullanılan sınıftır.

## Konu 2: Bağlantı Oluşturmak

### Bağlantı Oluşturmak

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



## Bağlantı Cümlesi (Connection String) Oluşturmak

| Bağlantı Cümlesi Oluşturmak   |   |
|---|---|
| Bağlantı cümlesi, bir veri kaynağına bağlanmak için gerekli olan en temel öğedir. |   |
| Parameter   | Tanım   |
| Provider  | Bağlantı sağlayıcısının ismi tutar.                                 |
| ConnectionTimeout veya Connect Timeout  | Bağlantı için beklenmesi gereken maksimum saniye sayısıdır.         |
| Initial Catalog   | Veritabanı adı  |
| Data Source   | Veritabanı için dosya adı   |
| Password (pwd)  | Hangi bağlantı girilir  |
| User Id (uid)   | Hangi kullanıcı ismi  |
| Integrated Security veya Trusted Connection                                       | Bağlantının güvenli olup olmadığına ilişkin parametredir.           |
| Persist Security Info   | Varsayılan durumda güvenliğin için önceki bilgileri geri döndürmez. |
| WorkstationID (wid)   | Workstation veya client adı tutar.                                  |
| Packet Size   | Veri transferinde kullanılan paketlerin boyutunu belirtir.          |
| Mode  | Read-only ya da Write modunu belirtir.                              |

Bağlantı cümlesi, veri kaynağına bağlanmak için gerekli bilgileri tutar. Bu cümle, veri kaynağına bağlantı kurmak için gerekli bağlantı parametrelerinin birleşiminden oluşur. Bu parametrelerin listesi Tablo 3.3'te gösterilmiştir.

**Tablo 3.3: Bağlantı Cümlesinin Parametreleri**

| Parametre   | Tanımı  |
|---|---|
| <b>Provider</b>   | Sadece <b>OleDbConnection</b> nesnelerinde kullanılır. Bağlantı sağlayıcısının ismini tutar. Sağlayıcı isimleri Tablo 3.4'te belirtilmiştir.  |
| <b>ConnectionTimeout</b> veya <b>Connect Timeout</b>      | Veritabanı bağlantı için beklenmesi gereken maksimum saniye sayısıdır. Varsayılan değer 15 saniyedir.   |
| <b>Initial Catalog</b>                                    | Veritabanı adı  |
| <b>Data Source</b>  | SQL Server adı, veya MS Access veritabanı için dosya adı  |
| <b>Password (pwd)</b>                                     | SQL Server login (giriş) parolası   |
| <b>User Id (uid)</b>                                      | SQL Server login (giriş) adı  |
| <b>Integrated Security</b> veya <b>Trusted Connection</b> | SQL sunucusuna Windows hesabı ile bağlantı yapılacağını belirtir. <b>True</b> , <b>False</b> veya <b>SSPI</b> girilebilir. <b>SSPI</b> , <b>True</b> ile eş anlamlıdır ve bu durumda Windows hesabı kullanılır. |
| <b>Persist Security Info</b>                              | Varsayılan değeri <b>False</b> olur. Bu durumda güvenlik için hassas bilgileri geri döndürmez. <b>True</b> olduğunda ise güvenlik risk taşımaya başlar.   |

**Tablo 3.3: Bağlantı Cümlesinin Parametreleri**

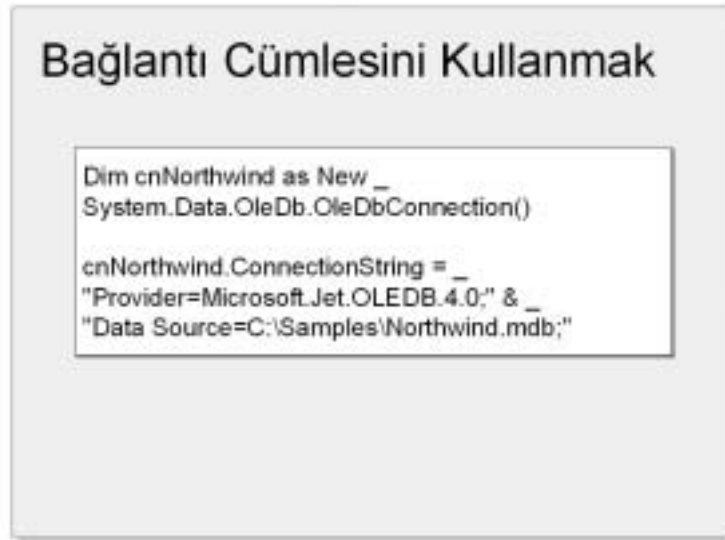
| Parametre                  | Tanımı  |
|----------------------------|---|
| <b>WorkstationID (wid)</b> | Workstation veya client (istemci) adını belirtir.   |
| <b>Packet Size</b>         | Client (istemci) - server (sunucu) arası veri transferinde kullanılan paketlerin boyutunu belirtir.                       |
| <b>Mode</b>                | Veritabanını Read-only (Sadece okunur) ya da Write (Yazılabilir) modunu belirtir. SQL Server bağlantılarında kullanılmaz. |

**Provider** parametresinin Access, SQL Server ve Oracle veritabanlarına göre alacağı değerler Tablo 3.4'te gösterilmiştir

**Tablo 3.4: Bağlantı cümlesinin parametreleri**

| Tür                            | Açıklama                                  |
|--------------------------------|---|
| <b>SQLLEDB</b>                 | SQL Server için Microsoft OLE DB Provider |
| <b>MSDAORA</b>                 | ORACLE için Microsoft OLE DB Provider     |
| <b>Microsoft.Jet.OLEDB.4.0</b> | Microsoft Jet için OLE DB Provider        |

## Bağlantı Cümlesini (Connection String) Kullanmak



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.

```
Dim cnNorthwind as New _  
System.Data.SqlClient.SqlConnection()  
  
cnNorthwind.ConnectionString = _  
"User ID=sa;" & _  
"Password=2389;" & _  
"Initial Catalog=Northwind;" & _  
"Data Source=London;" & _  
"Connection Timeout=60;"
```

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

```
Dim cnNorthwind as New _  
System.Data.OleDb.OleDbConnection()
```

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

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

```
Dim cnNorthwind as New _  
System.Data.OleDb.OleDbConnection()  
  
cnNorthwind.ConnectionString = _  
"Provider=SQLLEDB;" & _  
"Data Source = ProdServ; Initial Catalog = Pubs;" & _  
"Integrated Security=SSPI;"
```



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

## Bağlantı Cümlesi(Connection String) 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.mdb"                                 |
| 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 3.5'te OLEDB ile Access'e bağlanmak için gerekli, örnek bağlantı cümleleri gösterilmektedir.

**Tablo 3.5: 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.mdb"                                 |
| 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" |

## 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 3.6'da ODBC ile SQL Server'a bağlanmak için gerekli örnek bağlantı cümleleri gösterilmektedir.

**Tablo 3.6: SQL Server ile ODBC Bağlantı Cümleleri**

|  |   |
|--|---|
| 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;" |

## 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 3.7'de OLEDB ile SQL Server'a bağlanmak için gerekli örnek bağlantı cümleleri gösterilmektedir.

**Tablo 3.7: SQL Server ile OLEDB Bağlantı Cümleleri**

|  |   |
|--|---|
| 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;"            |



## 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 3.8'de SQLClient ile SQL Server'a bağlanmak için gerekli örnek bağlantı cümleleri gösterilmektedir.

**Tablo 3.8: SQL Server ile SQL Server Bağlantı Cümleleri**

|  |  |
|--|--|
| 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;"                                      |



**Bağlantı cümle parametrelerinin benzer eşdeğerleri vardır. Bu eşdeğerler hem OLEDB hem de SQLClient veri sağlayıcılarda kullanılabilir. Tablo 3.9'da bu eşdeğerler gösterilmektedir.**

**Tablo 2.7: OleDb ve Sql Server Parametre Eşdeğerleri**

| OLEDB ve SqlServer Parametreleri | Eşdeğerleri |
|----------------------------------|-------------|
| Data Source                      | Server      |
| User ID                          | UID         |
| Password                         | PWD         |
| Initial Catalog                  | Database    |

## Konu 3: Bağlantı Yönetimi

### Bağlantı Yönetimi

- Bağlantıyı Açmak ve Kapatmak
- Bağlantı Durumlarını Kontrol Etmek

## Bağlantıyı Açmak ve Kapatmak

### Bağlantıyı Açmak ve Kapatmak

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

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

Bağlantı cümlesini oluşturduktan sonra, bağlantıyı açmak ve kapatmak 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 içinde kurmaya çalışır. Eğer belirtilen süre içinde bağlantı gerçekleşmiyorsa, 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 üretir. 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.



**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 gerçekleştirilir.
'Bağlantıyı kapatmak
cnNorthwind.Close()

```

**Open** metodu ile veri kaynağı açılırken, çeşitli çalışma zamanı hatalarından dolayı bağlantı açılmayabilir ve uygulama hata üretebilir. 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 oluşabilecek 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 içine, yakalanan hataya göre yapılacak işlemler yazılmalıdır. Örneğin bağlantının açılmadığı bir durumda veritabanı işlemleri gerçekleştirilmeye çalışıldığı zaman kullanıcıya bağlantının açılmasını bildiren mesaj kutusu çıkarılabilir.

**Finally** bloğunda, **Try** ve **Catch** bloklarından herhangi biri işlendikten 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.

```
Dim cnNorthwind As System.Data.SqlClient.SqlConnection
```

```
Try
```

```

cnNorthwind = New System.Data.SqlClient.SqlConnection
cnNorthwind.ConnectionString = _
    "Provider=Microsoft.Jet.OLEDB.4.0;" & _
    "Data Source=C:\Samples\Northwind.mdb;"

```

```

cnNorthwind.Open()
' Veritabanı işlemleri gerçekleştirilir.

```

```
Catch XcpInvOp As InvalidOperationException
    ' İlk önce bu tipte hata yakalanır.
    MessageBox.Show("Önce veritabanı bağlantısını kapatın")
Catch Xcp As Exception
    ' Diğer hatadan farklı bir tipte hata burda yakalanır.
    MessageBox.Show(Xcp.ToString())
Finally
    cnNorthwind.Close()
    cnNorthwind.Dispose()
End Try
```

## Bağlantı Durumlarını Kontrol Etmek

### Bağlantı Durumlarını Kontrol Etmek

Bağlantı nesnesinin State özelliği, 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 3.10'da belirtilmiştir.

**Tablo 3.10: Connection Nesnesinin State Özelliğinin Değerleri**

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

```

Private Sub ConnectionAc(ByVal con As OleDb.OleDbConnection)
    ' Connection, sadece kapalı ise açılacak
    If con.State = ConnectionState.Closed Then
        con.Open()
    End If
End Sub

```

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 3.11’de bu parametrenin **CurrentState** ve **OriginalState** özellikleri görülmektedir.

**Tablo 3.11: StateChangeEventArgs Parametresinin Özellikleri**

| Özellik              | Açıklama   |
|----------------------|--|
| <b>CurrentState</b>  | Bağlantının yeni durumu hakkında bilgi verir.              |
| <b>OriginalState</b> | Bağlantının değişmeden önceki durumu hakkında bilgi verir. |

```
Public Sub ConnectionOlustur()
    Dim conn As New System.Data.OleDb.OleDbConnection

    ' Bağlantı ayarları yapılır.
    ' ...

    ' Bağlantının StateChange olayı gerçekleştiği zaman
    ' DurumRapor yordamının çağırılması ayarlanır
    AddHandler conn.StateChange, AddressOf DurumRapor
End Sub
```

```
Public Sub DurumRapor(ByVal sender As Object, _
    ByVal e As StateChangeEventArgs)


    MessageBox.Show("Bağlantı " & _
        e.OriginalState.ToString & " durumundan " & _
        e.CurrentState.ToString & " durumu olarak değişti.")

End Sub
```

## 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 veritabanına bağlanmak için hangi veri sağlayıcının kullanılması gerekir?
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



Bu labda, kullanıcıyı 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        |

| Kontrol – Kontrol İsmi | Özellik | Değer                     |
|------------------------|---------|---------------------------|
| Button – Button1       | Text    | Connection String Oluştur |
| Button – Button2       | Text    | Connection Oluştur        |
| Button – Button3       | Text    | Connection Aç             |
| Button – Button4       | Text    | Connection Kapat          |



RESİM 3.1.

## Kodların Yazılması

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

1. Bağlantı cümlesi için gerekli bilgileri tutan değişkenleri tanımlayın.

```
Private ConnectionString As String
Private Provider As String
Private Database As String
Private Server As String
Private KullaniciAdi As String
Private Parola As String
Private TimeOut As String

' Sql veritabanına bağlanmak için
Private sqlCon As SqlClient.SqlConnection
' Access veritabanına bağlanmak için
Private oleDbCon As OleDb.OleDbConnection
```

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

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    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
    End Sub

```

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

```

Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ComboBox1.SelectedIndexChanged
    ' Farklı Veritabanı seçildiği zaman tetiklenir.
    Select Case ComboBox1.SelectedIndex
        Case 0
            Provider = "Microsoft.Jet.OleDB.4.0"
            EnableTextBoxes(False)
        Case 1
            Provider = ""
            EnableTextBoxes(True)
        Case 2
            Provider = "SQLOLEDB"
            EnableTextBoxes(True)
    End Select
End Sub

Private Sub EnableTextBoxes(ByVal SQLVeriTabaniSecili As
Boolean)
    txtKullaniciAdi.Enabled = SQLVeriTabaniSecili
    txtParola.Enabled = SQLVeriTabaniSecili
    txtTimeOut.Enabled = SQLVeriTabaniSecili
    txtSunucu.Enabled = SQLVeriTabaniSecili

    ' Access veritabanı seçili ise
    If Not SQLVeriTabaniSecili Then
        txtKullaniciAdi.Text = ""
        txtParola.Text = ""
        txtTimeOut.Text = ""
        txtSunucu.Text = ""
    End If
End Sub

```

- 4. Connection String oluşturma düğmesi tıklandığı zaman, girilen değerleri alın ve bağlantı cümlesi oluşturun.**

```

Private Sub Button1_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button1.Click
    ConnectionString = ""

    ' Access Veritabani seçili ise
    If ComboBox1.SelectedIndex = 0 Then
        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 <> "" Then
            ConnectionString = "Provider=" & Provider & ";"
        End If

        ConnectionString &= "Data Source=" & Server
        ConnectionString &= ";Initial Catalog=" & Database
        ConnectionString &= ";User ID=" & KullaniciAdi
        ConnectionString &= ";Password=" & Parola
        ConnectionString &= ";Connection Timeout=" & TimeOut
    End If

    lblConnectionString.Text = ConnectionString
End Sub

```

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 Sub Button2_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button2.Click
    ' Access veritabani seçili ise
    If ComboBox1.SelectedIndex = 0 Then
        OleDbCon = New _
            OleDb.OleDbConnection(ConnectionString)
        AddHandler OleDbCon.StateChange, _
            AddressOf DurumDegisti
    Else
        sqlCon = New _
            SqlClient.SqlConnection(ConnectionString)

```

```

        AddHandler sqlCon.StateChange, _
            AddressOf DurumDegisti
    End If
End Sub

' Bağlantı durumu değiştiği zaman, DurumDegisti
' yordamındaki kodlar çalışır.
Private Sub DurumDegisti(ByVal sender As Object, ByVal e As
System.Data.StateChangeEventArgs)
    ' Bağlantının durumu kullanıcıya bildirilir.
    lblConnectionString.Text = e.CurrentState.ToString
End Sub

```

6. Bağlantı nesnesi oluşturulduktan sonra, Connection Aç düğmesi tıklandığında bağlantıyı **Open** metodu ile açın. Ancak kullanıcının bazı değerleri yanlış girmesi durumu göz önüne alınarak **Try Catch** blokları kullanılmalıdır.

```

Private Sub Button3_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button3.Click
    If Not oleDbCon Is Nothing Then
        ' OleDb baglantisi olusturulduysa
        Try
            oleDbCon.Open()
        Catch ex As InvalidOperationException
            MsgBox(ex.Message)
        Catch ex As OleDb.OleDbException
            MsgBox(ex.Message)
        End Try
    Else
        ' Sql baglantisi olusturulduysa
        Try
            sqlCon.Open()
        Catch ex As InvalidOperationException
            MsgBox(ex.Message)
        Catch ex As SqlClient.SqlException
            MsgBox(ex.Message)
        End Try
    End If
End Sub

```

7. Connection Kapat düğmesi tıklandığında oluşturulan bağlantıyı bularak kapatın.

```

Private Sub Button4_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Button4.Click
    If Not oleDbCon Is Nothing Then

```

```
        ' OleDb baglantisi olusturulduysa  
        oleDbCon.Close()  
    Else  
        ' Sql baglantisi olusturulduysa  
        sqlCon.Close()  
    End If  
End Sub
```

## 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 kümesi döndürebilecek,
- Command ile INSERT, UPDATE ve DELETE sorgularını çalıştırabileceksiniz

## **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 çalıştırmak 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 içinde kullanılan sınıflar Tablo 4.1’de belirtilmiştir.

**Tablo 4.1. Bağlantılı Veri Ortamı Sınıfları**

| Sınıf         | Açıklama  |
|---------------|---|
| XXXConnection | Bağlantı açmak ve kapatmak için kullanılan nesnedir.  |
| XXXCommand    | Veritabanı üzerinde Stored Procedure (Saklı Yordam) veya SQL cümleleri çalıştırmak için kullanılan nesnedir.  |
| XXXDataReader | Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılan nesnedir. Kayıtlar <b>XXXCommand</b> nesnesinin <b>ExecuteReader</b> metodu ile <b>XXXDataReader</b> içine aktarılır. |

## Konu 2: Command ile Çalışmak

### Command ile Çalışmak

- Command Nedir?
- Command Oluşturmak
- Parametre Kullanmak

## Command Nedir?

### Commmand 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               |
|---|----------------------------------|
| <b>System.Data.SqlClient.SqlCommand</b> | SQL Server .NET Veri Sağlayıcısı |
| <b>System.Data.OleDb.OleDbCommand</b>   | OLE DB .NET Veri Sağlayıcısı     |
| <b>System.Data.OleDb.OdbcCommand</b>    | ODBC .NET Veri Sağlayıcısı       |

Veritabanı üzerinde Stored Procedure ve sorgu çalıştırmak için **Command** nesnelerinin belirli özelliklerini kullanmak gerekir. **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 içindeki ismidir. Bu isim Command nesnesine başvurmak için kullanılır.
- **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 içinde 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 güncellemesi yapılacağına 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 çalıştırmak için Tablo 4.3'teki metotlardan uygun olan seçilir.

**Tablo 4.3: Command Metotları**

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

## Command Oluşturmak

### Command Oluşturmak

- Command, kod içerisinde veya ToolBox üzerinden oluşturulabilir.
- Adımlar;
  - Veri kaynağına bağlantı kurmak için Connection tanımlanır.
  - ToolBox içinden Data paneli seçilir.
  - Data panelindeki OleDbCommand veya SqlCommand nesnesi form üzerine sürüklenir.

**Command**, kod içinden 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.  
Dim cmd As New OleDbCommand  
'Command Sınıfının CommandText özelliğine üniversiteler  
'tablosunun tüm kayıtlarını getirecek sorgu cümlesi yazılır.  
cmd.CommandText = "select * from üniversiteler"  
'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.

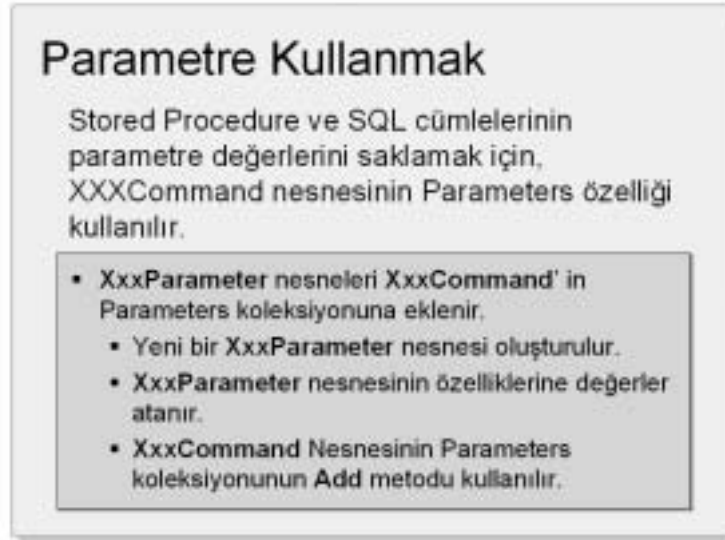
1. Veri kaynağına bağlantı kurmak için **Connection** tanımlanır.
2. ToolBox içinden Data paneli seçilir.
3. Data panelindeki **OleDbCommand** veya **SqlCommand** nesnesi form üzerine sürüklenir.
4. Eklenen **Command** nesnesinin özellikleri Tablo 4.4'e göre ayarlanır.

**Tablo 4.4: Command Özellikleri**

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



## Parametre Kullanmak



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 çalıştırdıktan 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'e göre ayarlanır.

**Tablo 4.5: XxxParameter Özellikleri**

| Özellik                             | Açıklama  |
|-------------------------------------|---|
| <b>ParameterName</b>                | Parametrenin ismi, @Ad gibi   |
| <b>DbType ,SqlDbType, OleDbType</b> | Parametrenin veri türü. Kullanılan veritabanına göre <b>SqlDbType</b> veya <b>OleDbType</b> enumeratörlerinden seçilir.   |
| <b>Size</b>                         | Parametredeki verinin byte olarak maksimum boyutu.  |
| <b>Direction</b>                    | Parametrenin türü.<br><b>ParameterDirection</b> değerlerinden biri ile belirtilir. Bu özelliğin alabileceği değerler:<br><b>ParameterDirection.Input</b> (varsayılan değer)<br><b>ParameterDirection.InputOutput</b><br><b>ParameterDirection.Output</b><br><b>ParameterDirection.ReturnValue</b> |

3. **XxxParameter** nesnelerini **Command** nesnesine eklemek için, **Command** nesnesinin **Parameters** koleksiyonunu içindeki **Add** metodu kullanılır. Eğer bu komut sonuç döndürecek bir Stored Procedure çağırıyorsa, herhangi bir parametre eklemekten önce **ParameterDirection.ReturnValue** parametresini eklenmelidir. Parametrelerin eklenme sırası önemli değildir.

| Parametre Kullanmak                  |   |
|--------------------------------------|---|
| XXXParameter nesnesinin özellikleri; |   |
| Property                             | Açıklama  |
| ParameterName                        | Parametrenin ismi. @Ad gibi   |
| DbType, SqlDbType, OleDbType         | Parametrenin veri türü. Kullanılan veri tabanına göre SqlDbType veya OleDbType enumeratörlerinden seçilir.  |
| Size                                 | Parametredeki verinin byte olarak maksimum boyutu.  |
| Direction                            | Parametrenin türü. ParameterDirection enumeratörlerinden biri ile belirtilir. Bu özelliğe atılabileceği değerler:<br>ParameterDirection.Input (varsayılan değer)<br>ParameterDirection.InputOutput<br>ParameterDirection.Output<br>ParameterDirection.ReturnValue |

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

**Tablo 4.6 Direction Özelliğinin Enumeratörleri**

| Enumeratör         | Özellik  |
|--------------------|--|
| <b>Input</b>       | Girdi parametresidir. Varsayılan değerdir.             |
| <b>Output</b>      | Çıktı parametresidir.                                  |
| <b>InputOutput</b> | Girdi ve çıktı parametresi olarak kullanılır.          |
| <b>ReturnValue</b> | Bir fonksiyon sonucunu geri döndürmek için kullanılır. |

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

```
Dim paramUser As New OleDbParameter("@userName", _
    OleDbType.Char, 50)
paramUser.Direction = ParameterDirection.Input

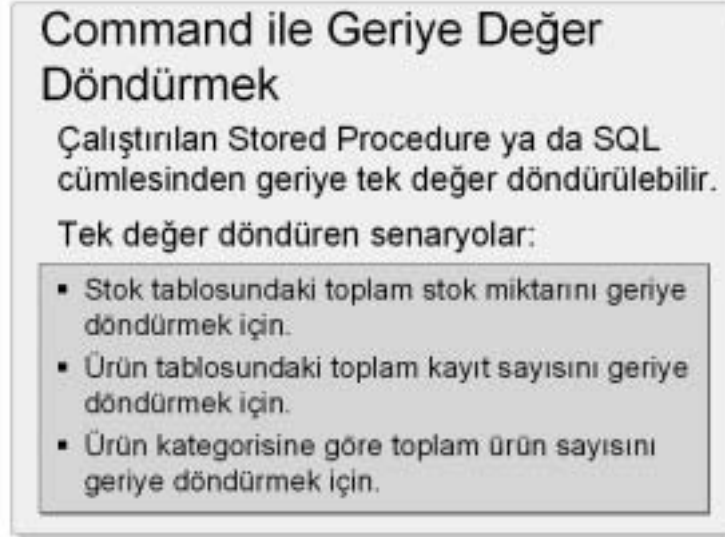
Dim paramPass As New OleDbParameter("@userPass", _
    OleDbType.Char, 20)
paramPass.Direction = ParameterDirection.Input

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?” anlamındaki ileti kutusuna olumlu yanıt verilir.
4. Visual Studio .NET ortamı **Command** nesnesi için parametre oluşturacak kodları otomatik olarak ekler.

## Konu 3: Command ile Geriye Değer Döndürmek



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 **Universiteler** tablosundaki toplam kayıt sayısı geri döndürülür.

```
Dim conn As New OleDbConnection("provider = _  
Microsoft.JET.OLEDB.4.0; Data source=..\universiteler.mdb")  
Dim cmd As New 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 **Urun** tablosundaki stok miktarı **SqlCommand** nesnesi ile geriye döndürülür.

```
Dim sql As String = "SELECT StokMiktari FROM Urun " & _  
    "WHERE UrunID = @UrunID"  
  
Dim cmUrun As New SqlCommand(sql, cnAlisveris)  
  
Dim prmID As SqlParameter = cmUrun.Parameters.Add( _  
    New SqlParameter("@UrunID", SqlDbType.Int, 4))  
  
cmUrun.Parameters("@UrunID").Value = 42  
  
cnAlisveris.Open()  
  
Dim adet As Integer = _  
    CType(cmUrun.ExecuteScalar(), Integer)  
  
cnAlisveris.Close()  
  
MessageBox.Show("Quantity in stock: " & adet.ToString())
```

## Konu 4: Command ile Geriye Kayıt Döndürmek

### Command ile Geriye Kayıt Döndürmek

Çalıştırılan Stored Procedure ya da SQL cümlesi, geriye birden çok değer veya kayıt kümesi döndürülebilir.

Kayıt kümesi döndüren senaryolar:

- Müsteri 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.(**Musteri** tablosundan **MusteriID**'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 metot 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 içinde değer olup olmadığını kontrol eder.

- Kolonların şema bilgilerini okur. (**ColumnName**, **ColumnOrdinal**, **ColumnSize**, **NumericPrecision**, **NumericScale**, **Datatype**, **ProviderType**, **IsLong**, **AllowDBNull**)



**DataReader**, verilere tek yönlü (forward-only) ve okunabilir (read-only) eriştiği için oldukça hızlıdır.



| DataReader Nesnesinin Metotları |  |
|---------------------------------|--|
| Metot                           | Açıklama   |
| Close()                         | DataReader kapatılır ve hafızada yer tutmaz.   |
| Get(Veritörü)                   | Belirli bir kolondaki değeri istenen veri türünde geri döndürür.   |
| GetString()                     | Belirli bir kolondaki değeri string olarak geri döndürür.  |
| GetValue()                      | Belirli bir kolondaki verinin doğal değerini alır.   |
| GetValues()                     | Bulunan satırdaki tüm attribute kolonları alır.  |
| NextResult                      | Komut metrinde birden fazla SELECT ifadesi varsa, sonuçlar bu metoda iki ayrı küme gibi alınabilir.        |
| Read()                          | DataReader'da okunacak kayıt olduğu sürece okuma yapar. Kayıt varsa True, yoksa False değerine geri döner. |

DataReader nesnesinin metotları Tablo 4.7'de gösterilmiştir.

**Tablo 4.7: DataReader Metotları**

| Metot                  | Açıklama   |
|------------------------|--|
| <b>Close</b>           | <b>DataReader</b> nesnesini kapatılır ve hafızadan kaldırır.                         |
| <b>GetBoolean</b>      | Belirli bir kolonun değerini <b>boolean</b> olarak geri döndürür.                    |
| <b>GetByte</b>         | Belirli bir kolonun değerini <b>byte</b> olarak geri döndürür.                       |
| <b>GetBytes</b>        | Belirli bir kolonun değerini <b>byte</b> dizisi olarak geri döndürür.                |
| <b>GetChar</b>         | Belirli bir kolonun değerini <b>char</b> olarak geri döndürür.                       |
| <b>GetChars</b>        | Belirli bir kolonun değerini karakter dizisi olarak geri döndürür.                   |
| <b>GetDataTypeName</b> | Belirli bir kolonun veri türünü verir.   |
| <b>GetDateTime</b>     | Belirli bir kolonun değerini <b>DateTime</b> olarak geri döndürür.                   |
| <b>GetDecimal</b>      | Belirli bir kolonun değerini <b>Decimal</b> olarak geri döndürür.                    |
| <b>GetDouble</b>       | Belirli bir kolonun değerini <b>Double</b> olarak geri döndürür.                     |
| <b>GetFieldType</b>    | Belirli bir kolonun veri türünü geri döndürür.                                       |
| <b>GetFloat</b>        | Belirli bir kolonun değerini <b>Float</b> olarak geri döndürür.                      |
| <b>GetGuid</b>         | Belirli bir kolonun değerini Globally-unique identifier (GUID) olarak geri döndürür. |
| <b>GetInt16</b>        | Belirli bir kolonun değerini 16-bit tamsayı ( <b>Short</b> ) olarak geri döndürür.   |
| <b>GetInt32</b>        | Belirli bir kolonun değerini 32-bit tamsayı( <b>Integer</b> ) olarak geri döndürür.  |
| <b>GetInt64</b>        | Belirli bir kolonun değerini 64-bit tamsayı( <b>Long</b> ) olarak geri döndürür.     |
| <b>GetName</b>         | Belirli bir kolonun ismini geri döndürür.  |

**Tablo 4.7: DataReader Metotları**

| Metot                 | Açıklama  |
|-----------------------|---|
| <b>GetOrdinal</b>     | Belirli bir kolonun sıra numarasını geri döndürür.  |
| <b>GetSchemaTable</b> | <b>DataReader</b> nesnesinin şema bilgilerini gösterir. Tablo hakkındaki detay bilgilerini gösterir.                                      |
| <b>GetString</b>      | Belirli bir kolonun değerini <b>string</b> olarak geri döndürür.  |
| <b>GetTimeSpan</b>    | Belirli bir kolonun değerini <b>TimeSpan</b> nesnesi olarak geri döndürür.  |
| <b>GetValue</b>       | Belirli bir kolonun değerini geri döndürür.   |
| <b>GetValues</b>      | Belirli bir kaydın tüm kolon değerlerini geri döndürür.   |
| <b>NextResult</b>     | Komut metninde birden fazla <b>SELECT</b> ifade varsa, sonuçlar bu metot kullanılarak farklı veri kümeleri gibi alınabilir.               |
| <b>Read</b>           | <b>DataReader</b> nesnesinde okunacak kayıt olduğu sürece okuma yapar. Kayıt varsa <b>True</b> , yoksa <b>False</b> değeri geri döndürür. |

**DataReader** bağlantılı 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   |   |
|---|---|
| <ul style="list-style-type: none"> <li>▪ Kaydın belirli bir kolonunu veya tüm kolonlarını okur.</li> <li>▪ Kolonların içinde değer olup olmadığını kontrol eder.</li> <li>▪ Kolonların şema bilgilerini okur. (ColumnName, ColumnOrdinal, ColumnSize, NumericPrecision, NumericScale, Datatype, ProviderType, IsLong, AllowDBNull)</li> </ul> |   |
| 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.

**Tablo 4.8 DataReader Özellikleri**

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

## 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.

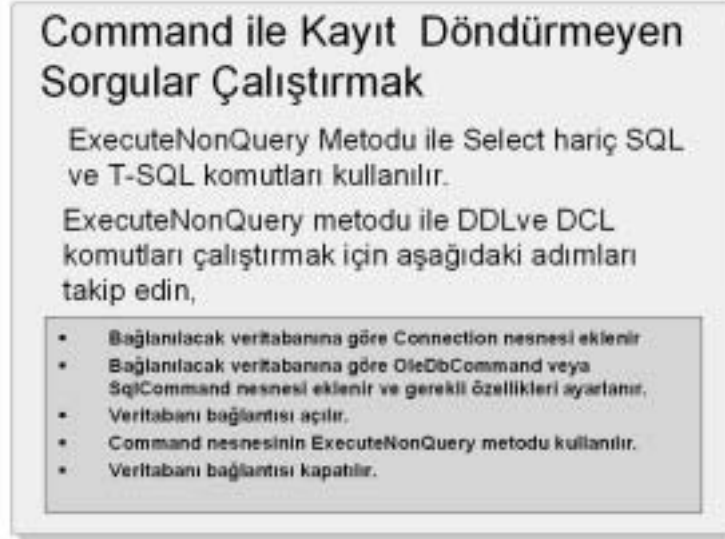
**DataReader** kullanarak kayıt çekmek için aşağıdaki adımları takip edin.

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 **Urun** tablosundaki tüm ürünler, **OleDbDataReader** ile form üzerindeki **ListBox** kontrolüne eklenir.

```
Dim cmUrun As New OleDbCommand( _
    "SELECT UrunAdi, StokMiktari " & _
    "FROM Urun", cnAlisveris)
cnAlisveris.Open()
Dim rdrUrun As OleDbDataReader
rdrUrun = cmUrun.ExecuteReader( _
    CommandBehavior.CloseConnection)
Do While rdrUrun.Read()
    ListBox1.Items.Add(rdrUrun.GetString(0) & _
        vbTab & rdrUrun.GetInt16(1))
Loop
rdrUrun.Close()
```

## Konu 5: Command ile Kayıt Döndürmeyen Sorgular Çalıştırmak



**Command** ile veritabanı yapısında değişiklik yapılabilir (tablo, View ve Stored Procedure oluşturmak, değiştirmek ve silmek), güvenlik seçenekleri ayarlanabilir (tablo ve View izinleri) ve veritabanı içindeki 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" (Yapılandırılmış Sorgulama Dili) veritabanı sorgu dilidir. SQL ile veritabanına kayıt ekleme, silme, varolan 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:

- **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'da gösterilmiştir.

**Tablo 4.9 DML Komutları**

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

- **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.

**Tablo 4.10 DDL Komutları**

| DDL Komutu | Açıklama   |
|------------|--|
| CREATE     | Yeni bir veritabanı nesnesi yaratmak. Örnek <b>CREATE TABLE, CREATE TRIGGER</b>    |
| ALTER      | Veritabanı nesnelerinde değişiklik yapmak. Örnek <b>ALTER TABLE, ALTER TRIGGER</b> |
| DROP       | Veritabanı nesnelerini silmek. Örnek <b>DELETE TABLE, DELETE TRIGGER</b>           |

- **SQL Veri Kontrol Dili (Data Control Language-DCL):** SQL Veri Kontrol Dili; bir veritabanı kullanıcısı veya rolü ile ilgili izinlerin düzenlenmesini sağlar. Tablo 4.11 DCL komutlarını ve fonksiyonlarını göstermektedir.

**Tablo 4.11 DCL Komutları**

| 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.               |

**ExecuteNonQuery** metodu ile DDL ve DCL komutları çalıştırmak 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, **Urun** isminde tablo oluşturulmaktadır.

```
Dim cmUrunTabloOlustur As New SqlCommand( _
    "CREATE TABLE Urun (UrunID int, UrunAdi " & _
    "varchar(50), " & _
    "StokMiktari int) ", connAlisveris)

cmUrunTabloOlustur.CommandType = _
CommandType.Text
Try
    connAlisveris.Open()
    Dim kayitSayisi As Integer = _
    cmUrunTabloOlustur.ExecuteNonQuery()
    connAlisveris.Close()
    MessageBox.Show(kayitSayisi & _
        "kayıt eklendi.")
Catch ex As Exception
    MessageBox.Show(ex.Message)
End Try
```

**ExecuteNonQuery** metodu ile **INSERT**, **UPDATE** ve **DELETE** sorguları çalıştırılabilir.

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

```
Dim cmd As New 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()
    Dim kayitSayisi As Integer = _
    cmd.ExecuteNonQuery()
    connAlisveris.Close()
    MessageBox.Show(kayitSayisi & _
        " kayıt eklendi.")
Catch ex As Exception
    MessageBox.Show(ex.Message)
End Try
```

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

```
Dim cmd As New SqlCommand("Update Urun " & _  
"Set StokMiktari = @StokMiktari " & _  
"Where UrunID = @UrunID", connAlisveris)  
  
cmd.Parameters.Add("@UrunID", 1)  
cmd.Parameters.Add("@StokMiktari", 30)  
  
Try  
    connAlisveris.Open()  
    Dim kayitSayisi As Integer = _  
    cmd.ExecuteNonQuery()  
    connAlisveris.Close()  
    MessageBox.Show(kayitSayisi & _  
    " kayıt değiştirildi.")  
Catch ex As Exception  
    MessageBox.Show(ex.Message)  
End Try
```

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


```
Dim cmd As New SqlCommand("Delete Urun " & _  
"Where UrunID = @UrunID", connAlisveris)  
  
cmd.Parameters.Add("@UrunID", 1)  
  
Try  
    connAlisveris.Open()  
    Dim kayitSayisi As Integer = _  
    cmd.ExecuteNonQuery()  
    connAlisveris.Close()  
    MessageBox.Show(kayitSayisi & _  
    " kayıt silindi.")  
Catch ex As Exception  
    MessageBox.Show(ex.Message)  
End Try
```



## Modül Özeti

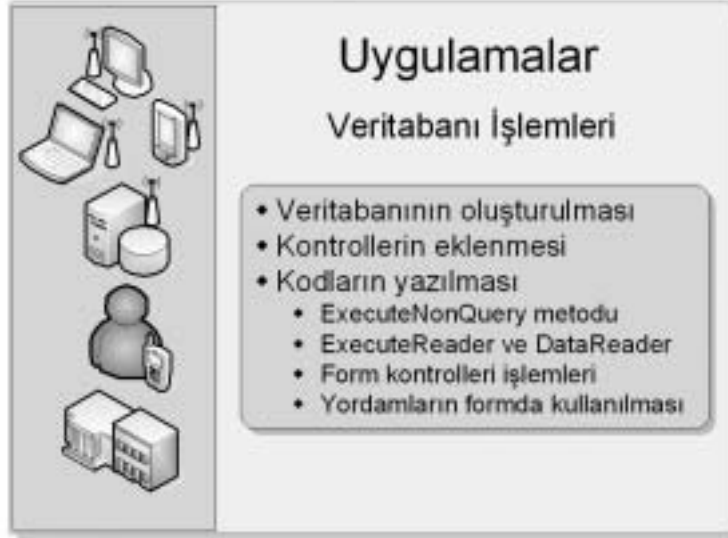
### Modül Özeti

- Bağlantılı veri ortamı hangi .NET nesneleri ile gerçekleştirilir?
- Command nesnesinin kaç farklı çalıştırılma 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 çalıştırılma 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



Bu uygulamada, veritabanındaki **Persone1** 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 **Persone1** tablosu için bir veritabanı oluşturulması gerekir.

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

Persone1 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          | İptal        |
| Button – btnKaydet     | Text          | Kaydet       |
| Button – btnSil        | Text          | Sil          |



RESİM 4.1.

## Kodların Yazılması

Persone1 tablosu üzerinde işlem yapılması için veritabanına bağlantı açılması gerekir. Bu bağlantı için gereken Connection String ifadesinin merkezi bir yerden alınması, değişiklik durumunda kolaylık sağlar.

1. Projeye bir modül ekleyin ve bağlantı dizisini tanımlayın.

```
Public connStr As String =
"Provider=Microsoft.Jet.OLEDB.4.0; Data Source=
C:\Samples\kisi.mdb"
```



Bu aşamadan sonra kodlar, formun kod tarafına yazılacaktır.

## ExecuteNonQuery Metodu

- Veritabanına yeni bir personel kaydı eklemek için gereken kodları bir yordam altında yazın.

```
Public Sub Kaydet()

    Dim conn As New OleDbConnection
    conn.ConnectionString = Module1.connStr

    Dim comm As New 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 ex As Exception
        MessageBox.Show(ex.Message)
    Finally
        conn.Close()
    End Try
End Sub
```

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

```
Public Sub Sil(ByVal ID As Integer)
    Dim conn As New OleDbConnection
    conn.ConnectionString = Module1.connStr
```

```
Dim comm As New 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 ex As Exception
    MessageBox.Show(ex.Message)
Finally
    conn.Close()
End Try
End Sub
```

## ExecuteReader ve DataReader

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

```
Public Sub IDdoldur()
    cbNo.Items.Clear()

    Dim conn As New OleDbConnection
    conn.ConnectionString = Module1.connStr
    Dim comm As New OleDbCommand
    comm.Connection = conn
    comm.CommandType = CommandType.Text
    comm.CommandText = "Select * from Personel"
    Dim dr As OleDbDataReader

    Try
        conn.Open()
        dr = comm.ExecuteReader()
        Do While dr.Read = True
            cbNo.Items.Add(dr.Item("Numara"))
        Loop
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    Finally
        dr.Close()
        conn.Close()
    End Try
End Sub
```

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

```
Public Sub IDyeGoreFormDoldur(ByVal ID As Integer)
    Dim conn As New OleDbConnection
    conn.ConnectionString = Module1.connStr
    Dim comm As New OleDbCommand
    comm.Connection = conn
    comm.CommandType = CommandType.Text
    comm.CommandText = _
        "Select * from Personel Where Numara=@No"
    comm.Parameters.Add("@No", ID)
    Dim dr As OleDbDataReader

    Try
        conn.Open()
        dr = comm.ExecuteReader
        If dr.Read = True Then
            txtAd.Text = dr.Item("Ad").ToString
            txtSoyad.Text = dr.Item("Soyad").ToString
            txtAdres.Text = dr.Item("Adres").ToString
            txtSehir.Text = dr.Item("Sehir").ToString
            txtDTarihi.Text = _
                dr.Item("DogumTarihi")
        End If
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    Finally
        dr.Close()
        conn.Close()
    End Try
End Sub
```

## Form Kontrolleri İşlemleri

6. Formdaki **TextBox** kontrollerinin değerlerini sıfırlayan kodları yazın.

```
Public Sub Temizle()
    Dim kontrol As New Control
    For Each kontrol In Me.Controls
        If TypeOf kontrol Is TextBox Then
            kontrol.Text = ""
        End If
    Next
    txtAd.Focus()
End Sub
```

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

```
Public Function Kontrol() As Boolean
    If txtAd.Text = "" Then
        MessageBox.Show("Adı Giriniz")
        txtAd.Focus()
        Return False
    ElseIf txtSoyad.Text = "" Then
        MessageBox.Show("Soyadı Giriniz")
        txtSoyad.Focus()
        Return False
    ElseIf txtDTarihi.Text = "" Then
        MessageBox.Show("Doğum Tarihini Giriniz")
        txtDTarihi.Focus()
        Return False
    ElseIf IsDate(txtDTarihi.Text) = False Then
        MessageBox.Show("Hatalı Tarih")
        txtDTarihi.Focus()
        txtDTarihi.SelectAll()
        Return False
    ElseIf txtAdres.Text = "" Then
        MessageBox.Show("Adresi Giriniz")
        txtAdres.Focus()
        Return False
    ElseIf txtSehir.Text = "" Then
        MessageBox.Show("Şehiri Giriniz")
        txtSehir.Focus()
        Return False
    Else
        Return True
    End If
End Function
```

## Yordamların Formda Kullanılması

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

```
Private Sub btnKaydet_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnKaydet.Click
    If Me.Kontrol = True Then
        Me.Kaydet()
        btnYeni.Enabled = True
        btnKaydet.Enabled = False
    End If
End Sub
```

```

        btnIptal.Enabled = False
        Me.IDDoldur()
        cbNo.SelectedIndex = cbNo.Items.Count - 1
    End If

```

```
End Sub
```

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

```

Private Sub btnYeni_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnYeni.Click
    Me.Temizle()
    btnYeni.Enabled = False
    btnKaydet.Enabled = True
    btnIptal.Enabled = True
    cbNo.SelectedIndex = -1
End Sub

```

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

```

Private Sub btnIptal_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnIptal.Click
    Me.Temizle()
    btnYeni.Enabled = True
    btnKaydet.Enabled = False
    btnIptal.Enabled = False
    cbNo.SelectedIndex = 0
End Sub

```

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

```

Private Sub btnSil_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnSil.Click
    If MessageBox.Show(cbNo.SelectedItem & _
        " nolu kaydı silmek istiyor musunuz?", _
        Me.Text, MessageBoxButtons.YesNo, _
        MessageBoxIcon.Question, _
        MessageBoxDefaultButton.Button2) = _
        DialogResult.Yes Then
        Me.Sil(cbNo.SelectedItem)
        Me.IDDoldur()
        cbNo.SelectedIndex = cbNo.Items.Count - 1
    End If
End Sub

```



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

```
Private Sub frmPersonel_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    Me.IDDoldur()
    cbNo.SelectedIndex = 0
End Sub
```

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

```
Private Sub cbNo_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cbNo.SelectedIndexChanged
    Me.IDyeGoreFormDoldur(cbNo.SelectedItem)
End Sub
```



# 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 Aramak ve Sıralamak

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.

Bağılantısız veri ortamı, uygulamanın veri kaynağına 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'te yapılan değişikliklerin veritabanına kaydedilmesini sağlar. |
| XXXConnection                     | Bağlantı açmak ve kapatmak için kullanılan nesnedir.   |
| XXXCommand                        | Veritabanı üzerinde Stored Procedure (Saklı Yordam) veya SQL cümleleri çalıştırmak için kullanılan nesnedir.   |
| XXXDataReader                     | Veritabanından sadece okunur ve ileri hareketli kayıtlar çekmek için kullanılır.   |

Bağlantısız veri ortamları içinde kullanılan sınıflar Tablo 5.1'de belirtilmiştir.

**Tablo 5.1. Bağlantısız Veri Ortamı Sınıfları**

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

## Konu 2: DataSet ve DataTable Oluşturmak

### DataSet ve DataTable Oluşturmak

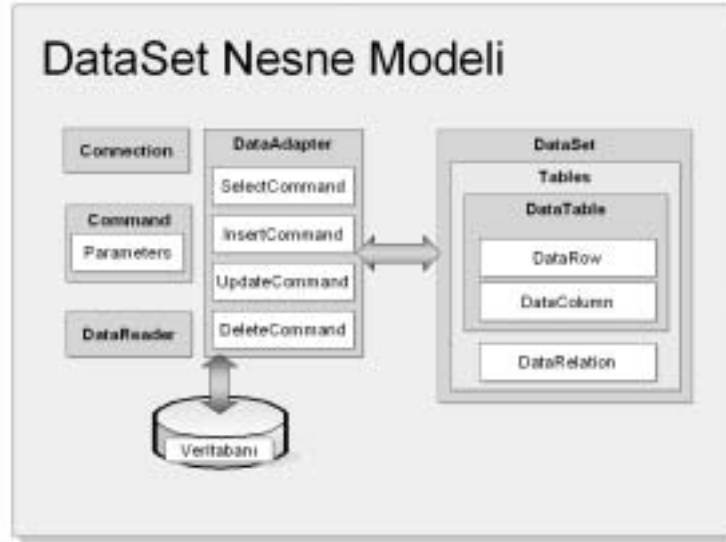
- Veri sağlayıcı türünden bağımsız çalışır.
  - DataSet tüm veri sağlayıcıları ile kullanılabilir.
- Sürekli çevrimdışıdır.
  - DataAdapter nesnesi ile veriler DataSet içerisine aktarılır ve bağlantı kapatılır.
- Değişikliklerin kaydını tutar.
  - DataSet içerisinde yapılan tüm değişiklikler, DataAdapter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

Veri kaynağından **DataAdapter** ile çekilen verilerin çekirdek belleğe atılan kopyası **DataSet** içinde 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 veri sağlayıcıları ile kullanılabilir. Tamamen türden bağımsız çalışır.
- **Sürekli çevrimdışıdır:** **DataAdapter** nesnesi ile veriler **DataSet** içine aktarılır ve bağlantı kapatılır. Bağlantı kesildikten sonra yapılan tüm değişiklikler **DataSet** içine kaydedilir. Bu durum uygulamanın çevrimdışı çalışmasını sağlar.
- **Değişikliklerin kaydını tutar:** **DataSet** içinde yapılan tüm değişiklikler, **DataAdapter** nesnesi ile veri kaynağına aktarılır.
- **Birden fazla tablo bulundurabilir:** İlişkisel veritabanlarında olduğu gibi, birden fazla tablo ve ilişkileri hafızada tutmanın tek yolu **DataSet** kullanmaktır.

## 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** ve **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 güncellemeleri işlemleri yapmak için kullanılır.

- **DataRow:** **DataTable** nesneleri için veri satırlarını temsil eder.
- **DataRelationship:** Tablolar arasındaki ilişkileri temsil eder.
- **DataView:** **DataTable** nesneleri üzerinde filtreleme, veri güncellemeleri 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.

```
Dim ds As New DataSet("YeniDataSet")
```

**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 (şema bilgileri) kopyalanarak yeni bir **DataSet** oluşturmaktır. İkinci yöntem **Clone** metodu ile diğer bir **DataSet** nesnesinin şema bilgilerini kopyalanarak, yeni bir **DataSet** oluşturmaktır. Bu yöntem şablon kopyalamak için kullanılır.

Örnekte **ds** ismindeki **DataSet** nesnesinin tüm tablo, ilişki ve verileri **dsCopy** ismindeki **DataSet** nesnesinin içine aktarılmıştır.

```
Dim dsCopy As DataSet  
dsCopy = ds.Copy()
```

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

```
Dim dsClone As DataSet  
dsClone = ds.Clone()
```

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

```
Dim conn As New OleDbConnection("provider = " & _
    "microsoft.jet.oledb.4.0; data source=../stok.mdb")

Dim da As New OleDbDataAdapter("select * from " & _
    "kitaplar", conn)
Dim ds As New DataSet("set")

da.Fill(ds, "kitaplar")
DataGrid1.DataSource = ds.Tables("kitaplar")
```

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

Örnekte **dtKitaplar** isminde yeni bir **DataTable** oluşturulmaktadır.

```
Dim dtKitaplar As New DataTable("kitaplar")
```

Oluşturulan tabloyu **DataSet** içine eklemek için **DataSet** nesnesinin **Tables** koleksiyonu kullanılır.

```
Ds.Tables.Add(dtKitaplar)
```

**DataTable** nesnesinin içine kolon eklenebilir. Örnekte **dtKitaplar** ismindeki **DataTable** nesnesinin içine, **yeniId** isminde yeni bir kolon eklenmektedir. Yeni kolon eklemek için, **DataTable** nesnesinin **Columns** koleksiyonu kullanılır.

```
Dim colKitapId As DataColumn =
dtKitaplar.Columns.Add("yeniId", GetType(System.Int32))
```

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.

```
Dim colUcret As New _
DataColumn("Ucret", GetType(System.Decimal))
Dim colKdv As New _
DataColumn("KDV", GetType(System.Decimal))
colKdv.Expression = "Ucret * 0.17"
Dim colTutar As New _
DataColumn("Tutar", GetType(System.Decimal))
colTutar.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** isimindeki **DataSet** nesnesine aktarılır. **DataSet** içindeki veriler **DataGrid** ile ekranda gösterilir.

```
Dim conn As New OleDbConnection ("provider = " & _  
    "microsoft.jet.oledb.4.0; data source=C:\Stok.mdb")  
Dim da As New OleDbDataAdapter("select * from kitaplar", _  
    conn)
```

```
Dim ds As 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.

```
Dim da As New OleDbDataAdapter("select * from " & _  
    "kitaplar", conn)
```

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

```
Dim conn As New OleDbConnection (" provider = " & _  
"microsoft.jet.oledb.4.0; data source=C:\Stok.mdb")  
  
Dim cmd As New OleDbCommand("select * from kitaplar")  
cmd.CommandType = CommandType.Text  
cmd.Connection = conn  
  
Dim da As New OleDbDataAdapter(cmd)  
Dim ds As 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** içine 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 **Siparisler** tablosundaki tüm değişiklikler veri kaynağına aktarılmaktadır.

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

## Konu4: DataSet Nesnesini Kontrollere Bağlamak

DataSet nesnesini Kontrollere  
bağlamak

- DataSet İçindeki Veriyi Windows Kontrollerine Bağlamak
- DataSet İçindeki Veriyi DataGrid'e Bağlamak

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

## DataSet İçindeki Veriyi Windows Kontrollerine Bağlamak

### DataSet İçindeki Veriyi Windows Kontrollerine Bağlamak

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

**DataSet** nesnesin içerdiği veri, Windows Form içindeki herhangi bir kontrolün herhangi bir özelliğine bağlanabilir. Örneğin **DataSet** içindeki bir tablonun ilk satır ve sütunundaki veri, **TextBox** kontrolünün **Text** özelliğine bağlanabilir.

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

Basit veri bağlama; **DataSet** içindeki bir veri elemanını (**DataTable** kolonunu) Windows kontrollere 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** içindeki birden fazla veri elemanını Windows kontrollerine bağlama işlemidir. **DataGrid**, **ListBox**, **ErrorProvider** gibi kontroller bu gruba girer.

Örnekte, **Dataset** içindeki **kitap\_baslik** kolonunun 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** içindeki 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.

```
Dim count as Integer
Count = ds.Tables("kitaplar").Columns.Count - 1

For i As Integer = 0 To count
    ListView1.Items.Add(ds.Tables("kitaplar").Rows(i). _
        Item("kitap_baslik"))
Next

For i As Integer = 0 To count
    CheckedListBox1.Items.Add(ds.Tables("kitaplar"). _
        Rows(i).Item("kitap_baslik"))
Next
```

## DataSet İçindeki Veriyi DataGrid'e Bağlamak

### DataSet İçindeki Veriyi DataGrid'e Bağlamak

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

**DataGrid**, veriyi satırlar ve sütunlar halinde görüntüler. **DataGrid** ile ilişkisiz **DataSet** tabloları kolay bir şekilde görüntülenebilir. Bu görüntü Excel tablolarına benzer.

**DataGrid** ilişkili **DataSet** tabloları da gösterebilir. Bu durumda istenilen tabloya **DataGrid** üzerindeki gezinti köprülerinden erişilebilir.

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

Grafiksel yöntem ile bağlantı sağlamak için aşağıdaki adımları takip edin.

1. Araç kutusu üzerindeki **DataGrid** kontrolünü form üzerine sürükleyin.
2. **DataGrid** kontrolünün **DataSource** özelliğini, önceden oluşturulmuş **DataSet** nesnesine bağlayın.
3. **DataGrid** kontrolünün **DataMember** özelliğini, **DataSet** tablolarının herhangi biri ile bağlayın.

Programlama yöntemi ile bağlantı sağlamak için **DataGrid** nesnesinin **DataSource** özelliği kullanılır. Örnekte, **DataSet** içindeki **Kitaplar** tablosu **DataGrid** kontrolüne 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** içinde yeni bir **DataTable** oluşturduktan 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** isiminde yeni bir kolon eklenmiştir.

```
Dim drNew As DataRow = dtKitaplar.NewRow()
```

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

```
drNew(0) = "975-8725-14-9"  
drNew(1) = "Tamer Şahiner"
```

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** isimindeki **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** içindeki kayıtlar değiştirilebilir. **DataRow** nesnesi ile satır düzenleme işlemleri için aşağıdaki metotlar 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** içindeki dördüncü kayıt için güncelleme işlemi yapılmıştır.

```
Dim drNew As DataRow = dtKitaplar.Rows(3)
drNew.BeginEdit()
drNew("kitap_baslik") = "yeni hayat"
drNew("kitap_yazar") = "can dündar"
drNew.EndEdit()
```

**DataRow** ile **DataTable** içindeki belirli bir satır silinebilir. Örnekte, **DataTable** içindeki dördüncü kayıt silinmiştir.

```
Dim drSil As DataRow = 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 gerekir. 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 satıra gidebilmek için, **CurrencyManager** nesnesinin **Position** özelliği kullanılır.

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

```
Private cmKitaplar As CurrencyManager
Private Sub Form1_Load() Handles Form1.Load
    txtKitapAdi.DataBindings.Add("Text", dtKitaplar, _
        "kitap_baslik")
    cmKitaplar = CType(Me.BindingContext(dtKitaplar), _
        CurrencyManager)
    cmKitaplar.Position = 0
End Sub

Private Sub btnMoveNext()
    If cmKitaplar.Position <> cmKitaplar.Count - 1 Then
```

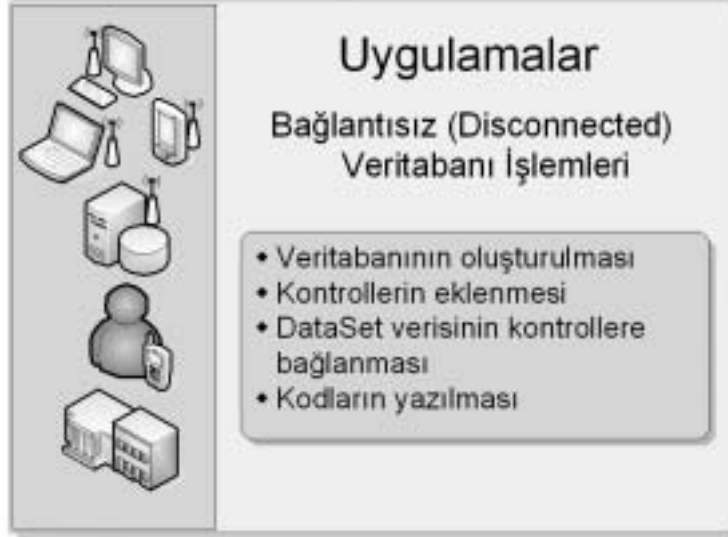
```
        cmKitaplar.Position += 1
    End If
End Sub

Private Sub btnMoveFirst()
    cmKitaplar.Position = 0
End Sub

Private Sub btnMovePrevious()
    If cmKitaplar.Position <> 0 Then
        cmKitaplar.Position -= 1
    End If
End Sub

Private Sub btnMoveLast()
    cmKitaplar.Position = cmKitaplar.Count - 1
End Sub
```

## Lab 1: Bağılantısız Veritabanı İşlemleri



Bu uygulamada, veritabanındaki **Persone1** 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** içinde gerçekleştirebilecek,
- **DataSet** içindeki değişiklikleri veritabanına kaydedebileceksiniz.

### Veritabanının Oluşturulması

Bu uygulamada kullanılacak **Persone1** tablosu için bir veritabanı oluşturulması gerekir.

1. Microsoft Access ile **kisi** isminde bir veritabanı oluşturun.
2. Veritabanına **Persone1** isminde bir tablo ekleyin ve tabloda belirtilen kolonları ekleyin.

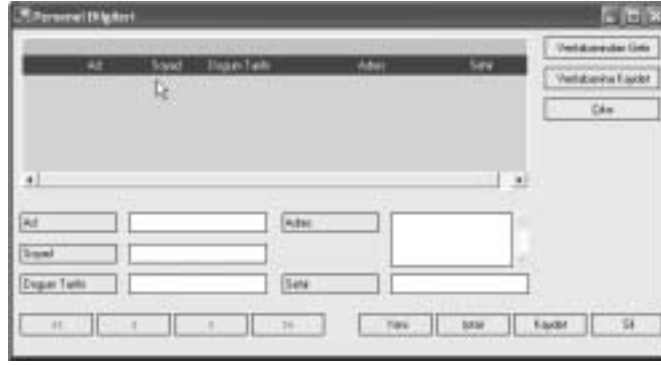
| Alan Adı    | Veri Türü  |
|-------------|------------|
| Numara      | AutoNumber |
| Ad          | Text       |
| Soyad       | Text       |
| DogumTarihi | Date/Time  |
| Adres       | Text       |
| Sehir       | Text       |

## Kontrollerin Eklenmesi

Baglantısız\_Personel isminde yeni bir Windows projesi açın.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

| Kontrol – Kontrol İsmi   | Özellik     | Değer                |
|--------------------------|-------------|----------------------|
| DataGrid – 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                |



RESİM 5.1.

## Bağlantı Cümlesinin Oluşturulması

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

**Bağlantısız\_Personel** uygulaması için yeni bağlantı oluşturmak.

1. Server Explorer penceresi üzerinde sağ 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 seçeneğini işaretleyin ve Next düğmesini tıklayın.



RESİM 5.2.

4. Açılan Connection sekmesinin görüntüsünü resimdeki gibi düzenleyerek OK düğmesini tıklayın.

## Bağlantının Oluşturulması

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



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

## DataAdapter Nesnesinin Oluşturulması

**Personel** tablosunu **DataSet** içine aktarmak için **OleDbDataAdapter** oluşturun.

1. Araç kutusu üzerindeki **OleDbDataAdapter** kontrolünü form üzerine sürükleyin.
2. Karşınıza çıkan Data Adapter Configuration Wizard penceresi üzerinde Next düğmesini tıklayarak bir sonraki adıma geçin.
3. Choose Your Data Connection penceresi üzerinde, oluşturduğunuz bağlantı cümlesini seçin ve Next düğmesini tıklayarak, bir sonraki adıma geçin.
4. Choose a Query Type penceresinde Use Sql statements seçeneğini işaretleyin ve Next düğmesini 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 düğmesini tıklayarak bir sonraki adıma geçin.
6. Finish düğmesini tıklayarak Data Adapter Configuration Wizard sihirbazını sonlandırın.

## DataSet Nesnesinin Oluşturulması

Personel kayıtları ile çevrimdışı çalışmak için **DataSet** oluşturun.

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

## DataSet İçindeki Verinin DataGrid Kontrolüne Bağlanması

**Personel** tablosunu **DataGrid** kontrolüne bağlayın.

1. **dgPersonel** isimli **DataGrid** nesnesinin **DataSource** özelliğine **ds1** isimli **DataSet** nesnesini seçin.
2. **dgPersonel** isimli **DataGrid** nesnesinin **DataMember** özelliğine **Personel** tablosunu seçin.

## DataSet İçindeki Verinin TextBox Kontrollerine Bağlanması

Personel tablosu içindeki Ad, Soyad, DTarihi, Adres ve Sehir kolonlarını sırayla txtAd, txtSoyad, txtDogumTarihi, txtAdres ve txtSehir metin kutularına bağlayın.

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

## Kodların Yazılması

1. btnDDoldur kontrolünün Click olayına kayıtları DataGrid kontrolüne dolduran kodları yazın.

Try

```
conn.Open()  
da.Fill(Ds1, "Personel")
```

Catch ex As Exception

```
MessageBox.Show(ex.Message)
```

Finally

```
conn.Close()
```

End Try

2. btnDKaydet kontrolünün Click olayına DataSet kontrolündeki tüm değişiklikleri veritabanına kaydeden kodu yazın.

Try

```
conn.Open()  
da.Update(Ds1, "Personel")
```

Catch ex As Exception

```
MessageBox.Show(ex.Message)
```

Finally

```
conn.Close()
```

End Try

3. btnYeni kontrolünün Click olayına DataSet tablosu için yeni satır oluşturan kodu yazın.

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

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

4. btnIptal kontrolünün Click olayına DataSet tablosu içinde eklenen yeni satırı iptal eden kodu yazın.

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

5. **btnKaydet** kontrolünün **Click** olayına **DataSet** tablosuna yeni kayıt ekleyen kodu yazın.

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

6. **btnSil** kontrolünün **Click** olayına **DataSet** tablosundan aktif kaydı silen kodu yazın.

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

7. **btnIlk** kontrolünün **Click** olayına ilk kayda giden kodu yazın.

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

8. **btnOnceki** kontrolünün **Click** olayına önceki kayda giden kodu yazın.

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

9. **btnSonraki** kontrolünün **Click** olayına sonraki kayda giden kodu yazın.

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

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

**DataSet** içinde veri arama ve sıralama işlemleri yapmak için, **DataTable** ve **DataRow** 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 yapılır. Bu aramanın sonucunda yazar adı geriye döndürülür.

```
Dim conn As New OleDbConnection("provider = " & _  
    "microsoft.jet.oledb.4.0;data source=c:\Proje\stok.mdb")  
Dim da As New OleDbDataAdapter("select * from" & _  
    "kitaplar", conn)  
  
Dim tbl As New DataTable  
da.Fill(tbl)  
tbl.PrimaryKey = _  
    New DataColumn(){tbl.Columns("kitap_ISBN")}  
Dim row As DataRow = tbl.Rows.Find("975-12-53-3")  
If row Is Nothing Then  
    MessageBox.Show("Kayıt Bulunamadı!")  
Else  
    MessageBox.Show(row("kitap_yazar"))  
End If
```

Örnekte **Tbl1** isimindeki **DataTable** nesnesine, **DataAdapter** nesnesinden gelen kayıtlar doldurulur. **Tbl1** tablosunun **Kitap\_ISBN** kolonu birincil anahtar olarak atanır. **Row** isminde yeni bir **DataRow** tanımlanır. **DataTable** nesnesinin **Find** metoduna kitabın barkod numarası girilir ve sonuç **Row** değişkenine aktarılır. Son olarak **Row** değişkeni içindeki 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 orijinal **DataSet** içindeki satırları işaret eden işaretçiler kümesi olarak da algılanabilir. Veri kopyalama yapmaz, ancak değişimleri görüntüler.

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

```
Dim selRows As DataRow() = tbl.Select("kitap_fiyat <> 8", _
    "kitap_baslik ASC", DataRowVersion.CurrentRows)
Dim row As DataRow
```

```
For Each row In selRows
    MessageBox.Show("kitap: " & _
        row("kitap_baslik", DataRowVersion.Original))
Next
```

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

```
public DataRow() Select()

public DataRow() Select(filterExpression as string)

public DataRow() Select(filterExpression as string, _
    sort as string)

public DataRow() Select(filterExpression as string, _
    sort as string, _
    recordStates as DataRowVersion)
```

**Select** metodunun **filterExpression**, **Sort** ve **recordStates** isminde 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 Metotları



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 adımları takip edin:

1. Araç kutusu üzerindeki **DataView** kontrolünü form üzerine sürükleyin.
2. **DataView** kontrolünün **Table** özelliği ile kullanılacak **DataTable**'i seçin.
3. **DataView** kontrolünün **Sort** özelliğine, sıralanacak kolon bilgilerini girin.
4. **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.

```
Dim dvProducts As New DataView(ds.Tables("kitaplar"))
dvProducts.Sort = "kitap_yazar"
dvProducts.RowFilter = "kitap_fiyat > 8"
DataGrid1.DataSource = dvProducts
```

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

```
Dim dvProducts As New DataView(ds.Tables("kitaplar"))
dvProducts.Sort = "kitap_yazar"
dvProducts.RowFilter = "kitap_fiyat > 8"
DataGrid1.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

- Bağılantısız 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?



1. Bağılantısız veri ortamı hangi .NET nesneleri ile gerçekleştirilir?
2. DataSet nedir? DataSet hangi nesnelerden oluşur?
3. DataTable nedir? Hangi durumlarda kullanılır?
4. DataColumn nedir? Hangi durumlarda kullanılır?
5. DataView nedir? Hangi durumlarda kullanılır?

## Lab 2: Çoklu Tablolarla Çalışmak



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

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

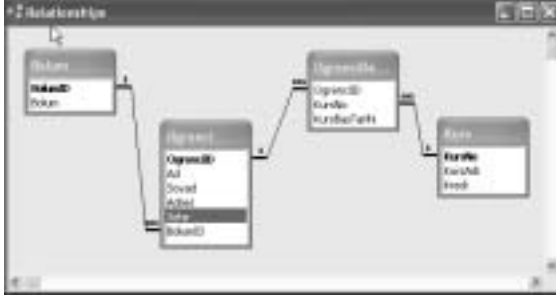
Bu lab tamamlandıktan sonra;

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

## Veritabanının Projeye Eklenmesi

Bu uygulamada kullanılacak **Course** veritabanı oluşturun.

1. Microsoft Access ile **Dershane** isminde bir veritabanı oluşturun.
2. Veritabanının tablolarını aşağıdaki diyagrama göre oluşturun.



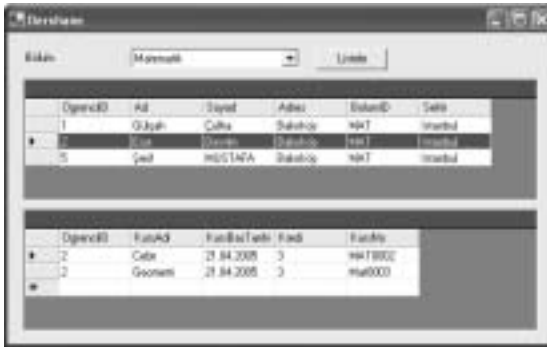
RESİM 5.3.

## 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      |
| Label – Label1         | Text          | Bölüm        |



RESİM 5.4.

## Bağlantı Cümlesinin Oluşturulması

**Dershane** veritabanı üzerinde işlem yapılması için bağlantı kurulması gerekir.

Bu bağlantı için gerekli Connection String ifadesini Server Explorer'ı kullanarak oluşturun.

Dershane uygulaması için yeni bağlantı oluşturmak:

1. Server Explorer penceresi üzerinde sağ 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 seçeneğini işaretleyin ve Next düğmesini tıklayın.



**RESİM 5.5.**

4. Açılan Connection sekmesinin görüntüsünü resimdeki gibi düzenleyerek OK düğmesini tıklayın.

## Bağlantının Oluşturulması

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

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

## DataAdapter Nesnesinin Oluşturulması

Bolum tablosunu **DataSet** içine aktarmak için **OleDbDataAdapter** oluşturun.

1. Araç kutusu üzerindeki **OleDbDataAdapter** kontrolünü form üzerine sürükleyin.
2. Karşınıza çıkan Data Adapter Configuration Wizard penceresi üzerinde Next düğmesini tıklayarak bir sonraki adıma geçin.
3. Choose Your Data Connection penceresi üzerinde, oluşturduğunuz bağlantı cümlesini seçin ve Next düğmesini tıklayarak bir sonraki adıma geçin.

4. Choose a Query Type penceresinden Use Sql statements seçeneğini işaretleyin ve Next düğmesini tıklayarak bir sonraki adıma geçin.
5. Generate the Sql statements penceresindeki metin kutusuna **SELECT Bolum, BolumID FROM Bolum** yazın ve Next düğmesini tıklayarak bir sonraki adıma geçin.
6. Finish düğmesini tıklayarak Data Adapter Configuration Wizard sihirbazını sonlandırın.
7. Eklediğiniz **OleDbDataAdapter** kontrolünün ismini **daBolum** olarak değiştirin.

**Ogrenci** tablosunu **DataSet** içine aktarmak için **OleDbDataAdapter** oluşturun.

1. Araç kutusu üzerindeki **OleDbDataAdapter** kontrolünü form üzerine sürükleyin.
2. Karşınıza çıkan Data Adapter Configuration Wizard penceresi üzerinde Next düğmesini tıklayarak bir sonraki adıma geçin.
3. Choose Your Data Connection penceresi üzerinde, oluşturduğunuz bağlantı cümlesini seçin ve Next düğmesini tıklayarak bir sonraki adıma geçin.
4. Choose a Query Type penceresinden Use Sql statements seçeneğini işaretleyin ve Next düğmesini 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 düğmesini tıklayarak bir sonraki adıma geçin.
6. Finish düğmesini tıklayarak Data Adapter Configuration Wizard sihirbazını sonlandırın.
7. Eklediğiniz **OleDbDataAdapter** kontrolünün ismini **daOgrenci** olarak değiştirin.

**Kurs** ve **OgrenciDersKayit** tablolarını **DataSet** içine aktarmak için **OleDbDataAdapter** oluşturun.

1. Araç kutusu üzerindeki **OleDbDataAdapter** kontrolünü form üzerine sürükleyin.
2. Karşınıza çıkan Data Adapter Configuration Wizard penceresi üzerinde Next düğmesini tıklayarak bir sonraki adıma geçin.
3. Choose Your Data Connection penceresi üzerinde, oluşturduğunuz bağlantı cümlesini seçin ve Next düğmesini tıklayarak bir sonraki adıma geçin.
4. Choose a Query Type penceresinden Use Sql statements seçeneğini işaretleyin ve Next düğmesini 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 düğmesini tıklayarak bir sonraki adıma geçin.

6. Finish düğmesini tıklayarak Data Adapter Configuration Wizard sihirbazını sonlandırın.
7. Eklediğiniz **OleDbDataAdapter** kontrolünün ismini **daKurs** olarak değiştirin.

## DataSet Nesnesinin Oluşturulması

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

1. **daBolum** üzerinde sağ tıklayın.
2. Açılan kısayol menüsünden Generate DataSet menüsünü tıklayın.
3. Choose a Dataset menüsünden New seçeneğini işaretleyin ve metin kutusuna **ds** yazın.
4. OK düğmesini tıklayın.
5. **daOgrenci** üzerinde sağ tıklayın.
6. Açılan kısayol menüsünden Generate DataSet menüsünü tıklayın.
7. Choose a Dataset menüsünden Existing seçeneğini işaretleyin ve açılan kutudan **Dershane.ds** seçeneğini işaretleyin.
8. OK düğmesini tıklayın.
9. **daKurs** üzerinde sağ tıklayın.
10. Açılan kısayol menüsünden Generate DataSet menüsünü tıklayın.
11. Choose a Dataset menüsünden Existing seçeneğini işaretleyin ve açılan kutudan **Dershane.ds** seçeneğini işaretleyin.
12. OK düğmesini tıklayın.

## DataView Nesnesinin Oluşturulması

**DataTable** nesneleri üzerinde filtreleme işlemleri yapmak için **DataView** oluşturun.

**Ogrenci** tablosunu filtrelemek için aşağıdaki adımları takip edin.

1. Araç kutusu üzerindeki **DataView** kontrolünü 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 adımları takip edin.

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

## DataSet İçindeki Verinin ComboBox Kontrolüne Bağlanması

**Bolum** tablosunu **cbBolum** isimli açılan kutuya bağlayın.

1. **cbBolum** isimli açılan kutunun **DataSource** özelliğine **Bolum** tablosunu seçin.
2. **cbBolum** isimli açılan kutunun **DisplayMember** özelliğine **Bolum** kolonunu seçin.
3. **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 ex As Exception

```
MessageBox.Show(ex.Message)
```

Finally

```
Conn.Close()
```

End Try

2. **btnListele** kontrolünün **Click** olayına **dvOgrenci** kontrolünü **DataGrid** kontrolüne bağlayan kodu yazın.

```
dvOgrenci.RowFilter = "BolumID='" & _  
    cbBolum.SelectedValue & "'" & _  
dgOgrenci.DataSource = dvOgrenci
```

3. **dgOgrenci** kontrolünün **CurrentCellChanged** olayına **dvKurs** kontrolünü **DataGrid** kontrolüne bağlayan kodu yazın.

```
dvKurs.RowFilter = "OgrenciID='" & _  
    dgOgrenci.Item(dgOgrenci.CurrentRowIndex, 0) & "'" & _  
dgKurs.DataSource = dvKurs
```





## Modül 6: ASP.NET'e Giriş

### ASP.NET'E 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 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 benzer. Web Server tarafında 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 tarafı kodları çalıştırdığı için, kullanıcı tarafındaki 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)

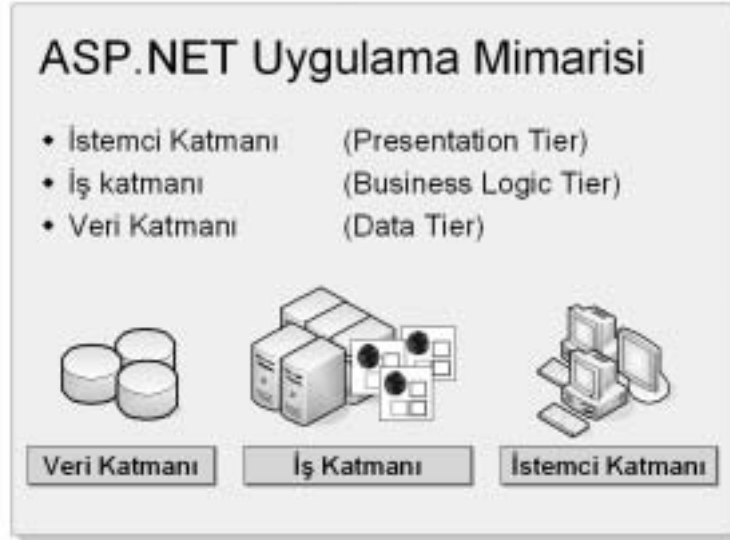
HTML (HyperText Markup Language) Web sayfası hazırlamak için kullanılan temel Web programlama dilidir.

HTML, kullanıcı ile sunucu arasında dinamik veri alışverişi sağlamaz. HTML'in bu açığını kapatmak için ilk olarak CGI arabirimi (Common Gateway Interface) geliştirilmiştir. CGI arabirimi 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, sunucu ile haberleşen ilk dil olan PERL(Practical Extraction and Reporting Language) geliştirilmiştir. Bu dil C ve C++ ile yazılan script'ler 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 destekler. 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 içinde, dinamik veri alışverişi için Microsoft tarafından geliştirilmiştir

## Konu 3: ASP.NET Uygulama Mimarisi



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 arayüzünü oluşturan bileşenler bulunur. HTML ve Sunucu kontroller, Web Formlar ve kullanıcı tanımlı kontroller (User Controls) bu katman içinde yer alır.

### İş katmanı (Business Logic Tier)

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

### Veri Katmanı (Data Tier)

Veri katmanıdır. Bu katmanda veriyi saklamak için gerekli araçlar bulunur. İlişkisel veritabanları, e-mail alanları, mesaj kuyukları ve izin servisleri bu katman içinde yer alır. Web uygulamalarda, ASP.NET ile veri kaynağına 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)



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 dosyaları içinde saklanır. Bu dosyalar kolayca okunur ve yazılabilir. Her Web uygulamasının kendisine ait bir konfigürasyon dosyası vardır. ASP.NET uygulamalarının konfigürasyon dosyaları **web.config** dir.

Sunucuya ait konfigürasyon ayarları ise **machine.config** içinde 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:

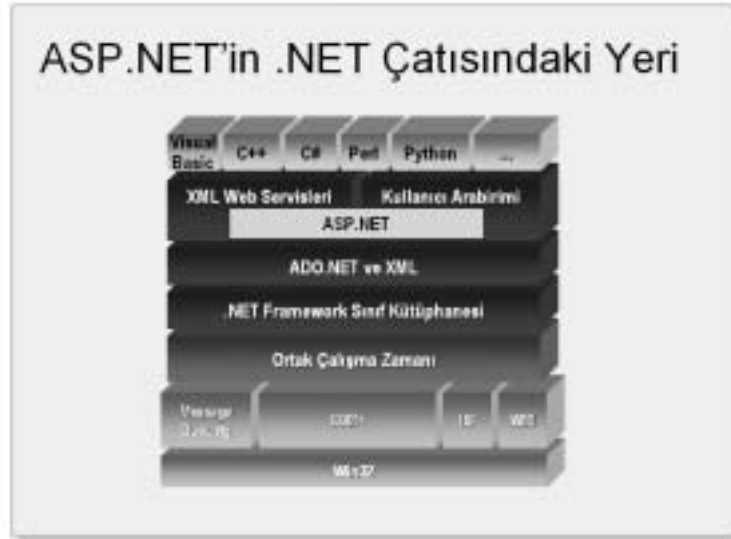
- **Web Formlar:** Web uygulaması için kullanıcı arayüzü sağlar.
- **Code-behind sayfalar:** Web Formların sunucu tarafında çalışan kodlarını içerir.
- **Konfigürasyon dosyaları:** Web uygulama ve sunucu ayarlarının tutulduğu XML dosyalarıdır.
- **global.asax dosyaları:** Web uygulamasının genel olaylarını içerir. Örneğin Web uygulamasının 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 alışverişini sağlar.
- Veritabanı bağlantıları
  - Web uygulaması ile veri kaynağı arasında veri alışverişini sağlar.
- Caching (Ön Belleğe Alma)
  - Uygulamanın ilk çalıştığı anda ön belleğe atılmasını sağlar.

- **XML Web Servis bağlantıları:** Web uygulamasının, XML Web servisi üzerinden veri alışverişini sağlar.
- **Veritabanı bağlantıları:** Web uygulaması ile veri kaynağı arasında veri alışverişini 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



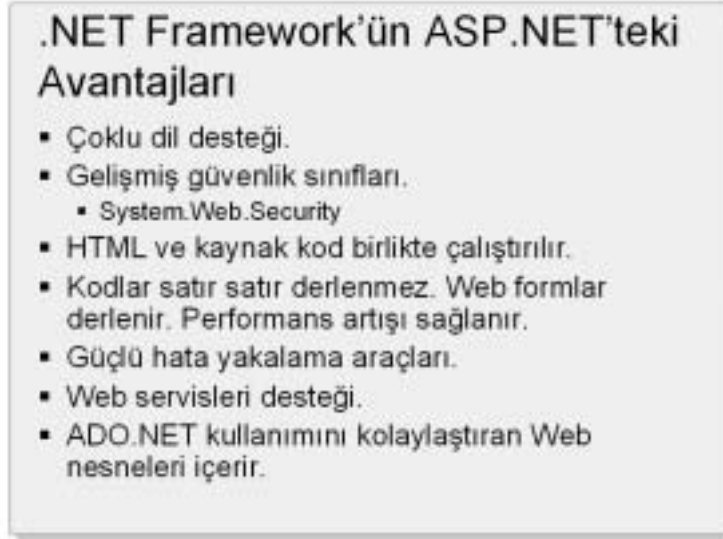
Microsoft .NET platformu, geniş çaplı Web uygulamaları geliştirebilmek için gerekli her türlü araç ve teknolojiye sahiptir. Dilden bağımsız çalışabilme, eski teknolojiden yeni teknolojilere kolayca geçiş imkanı sağlar. Tamamen nesne yönelimli programlamayı destekleyen bir platform olan Visual Studio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



**RESİM 6.1.**

Resim 6.1'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'ün ASP.NET'teki Avantajları



.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 içine C# ile yazılan kod blokları eklenebilir.
- Visual Studio .NET, Web uygulamaların güvenliğini sağlayan çeşitli sınıflar içerir. Bu sınıflar **System.Web.Security** isim alanı içinde bulunur.
- ASP .NET sayfaları içinde, HTML ve kaynak kod birlikte çalıştırılır. Bu durum tasarım ve programlama kolaylığı sağlar.
- ASP.NET içinde kodlar satır satır derlenmez. Bunun yerine Web formlar derlenir. Bu durum performansın artışını sağlar.
- Güçlü hata yakalama araçları sunar.
- Web servisleri ile birlikte çalışabilir.
- ASP.NET, ADO.NET kullanımını kolaylaştıran Web nesneleri içerir.

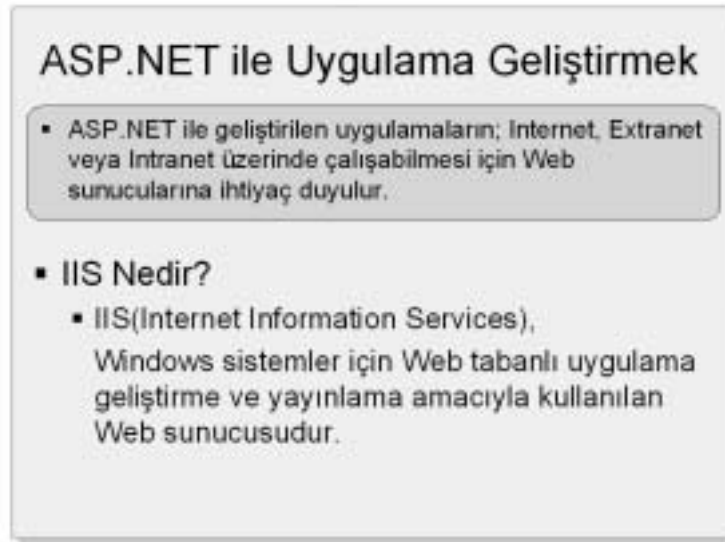
## Konu 7: ASP.NET ile Uygulama Geliştirmek

### 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?



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

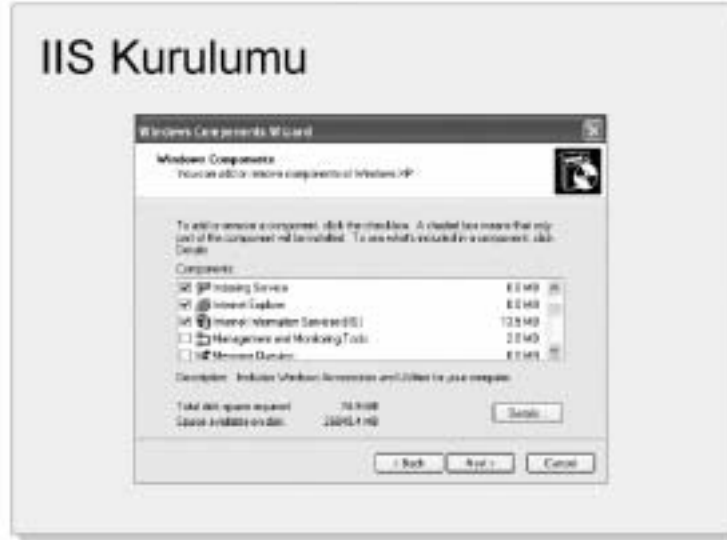
## IIS Kurulumu ve Yönetimi

### 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



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 gerekir.

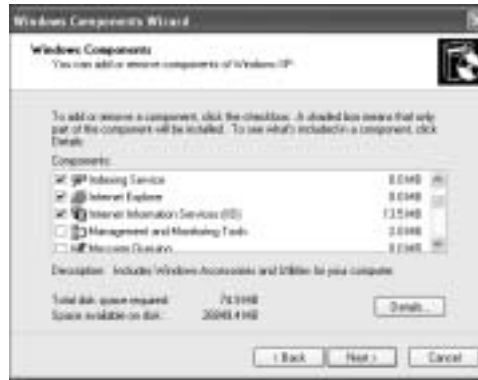
IIS kurulumu için aşağıdaki adımlar takip edilir:

1. Denetim Masası (Control Panel) penceresinde Program Ekle/Kaldır (Add or Remove Programs) simgesini çift tıklayın.
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) seçeneğini işaretleyerek yükleme işlemini başlatın.



**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 yüklendikten sonra, sistemin ASP.NET sayfaları ile uyum içinde çalışabilmesi için Visual Studio .NET komut satırında `aspnet_regiis.exe -I` komutu çalıştırılmalıdır.



**RESİM 6.2:** IIS kurulumu.

## IIS Yönetimi



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

IIS Manager'ı açmak için aşağıdaki adımlar takip edilir:

1. Bilgisayarım (My Computer) simgesi sağ tıklanır. Açılan kısayol menüsünden Yönet (Manage) komutu seçilir. Açılan Computer Manager penceresinin Services and Applications menüsünden Internet Information Services (IIS) seçilir.
2. Denetim Masası (Control Panel) içinden Administrative Tools simgesi seçilir. Açılan pencereden Internet Information Services Manager seçilir.

IIS içinde aşağıdaki alt klasörler bulunur:

- **Application Pools**
- **Web Sites**
- **Web Service Extensions**



**Web Sites** çalışan web uygulamalarını listeler. **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** sağ tıklanır. Açılan menüden Properties komutu seçilir.

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



**RESİM 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 sağ tıklanır. Çıkan menüden New alt menüsü işaretlenir 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 düğmeleri tıklanarak kurulum tamamlanır.

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

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

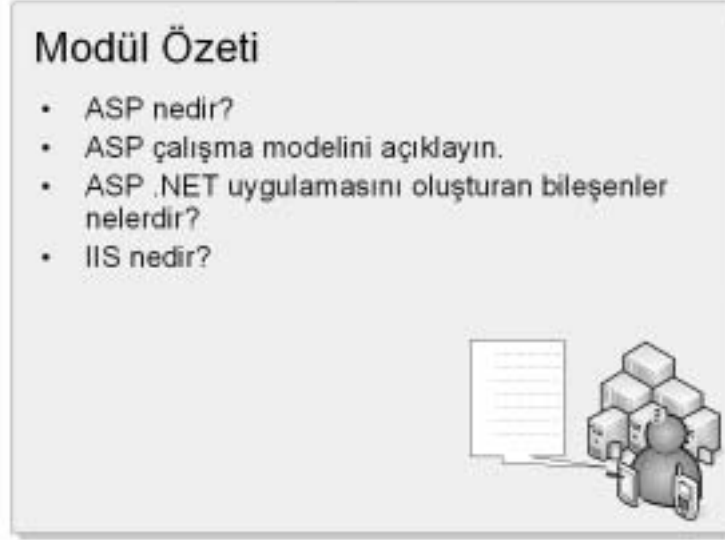
Framework'ü kurmak için aşağıdaki adımlar takip edilir:

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



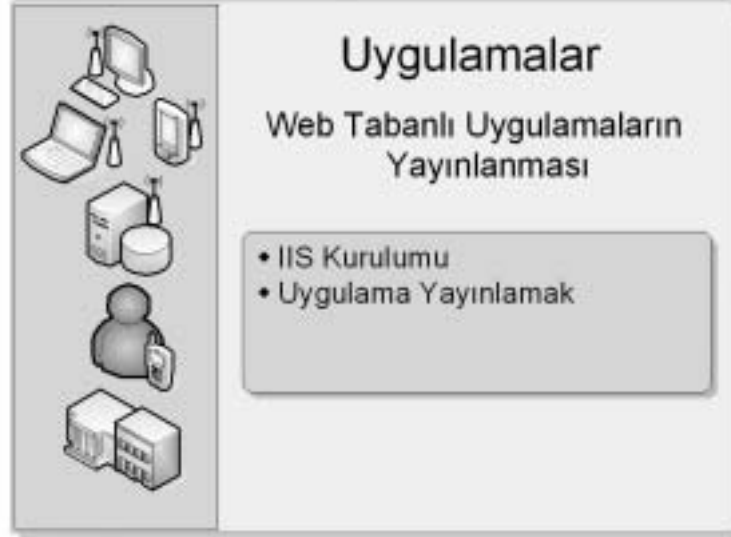
.NET Framework kurabilmek için işletim sisteminin Windows NT tabanlı olması gerekir. Windows 2000 işletim sisteminde minimum SP2 yapılandırması gereklidir.

## Modül Özeti



1. ASP nedir?
2. ASP çalışma modelini açıklayın.
3. ASP .Net uygulamasını oluşturan bileşenler nelerdir?
4. IIS nedir?

## Lab 1: Web Tabanlı Uygulamaların Yayınlanması



Bu uygulamada IIS (Internet Information Services) kurulumu öğreneceksiniz. Aynı zamanda IIS ile Web tabanlı uygulamaların yayınlanmasını öğreneceksiniz.

### IIS (Internet Information Services) Kurulması

1. Denetim Masası (Control Panel) penceresinde Program Ekle/Kaldır (Add or Remove Programs) simgesini çift tıklayın.
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) seçeneğini işaretleyin.
4. Next düğmesini tıklayarak kurulumu başlatın.

### Uygulama Yayınlamak

Default.htm isimindeki HTML sayfaı IIS üzerinden yayınlayın.

1. C:\Inetpub\wwwroot klasörüne gidin.
2. wwwroot penceresi içinden Dosya menüsünü tıklayın.
3. Dosya menüsü içinden Yeni alt menüsünü tıklayın.
4. Yeni alt menüsü içinden Metin Belgesi komutunu vererek Yeni Metin Belgesi oluşturun.



5. Oluşturduğunuz metin belgesi içine aşağıdaki HTML (Hyper Text Markup Language) kodlarını ekleyin ve dosyayı kaydedin.

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

  </body>
</html>
```

6. Metin belgesinin ismini Default.htm olarak değiştirin.
7. Internet Explorer açın ve aşağıdaki adreslerden herhangi birini adres çubuğuna yazın.
- a. `http://localhost`
  - b. `http://127.0.0.1`
  - c. `http://MakinaAdı`
  - d. `http://IpNumarası`

**localhost:** Lokal makine adı.

**127.0.0.1 :** Lokal IP numarası.

**MakinaAdı:** Ağ içindeki bilgisayar adı.

**IpNumarası:** Ağ içindeki Ip Numarası.



Web uygulamanın yayını, `wwwroot` içindeki herhangi bir alt klasörden yapılabilir. Örneğin `http://localhost/WebUygulama`. ASP.NET Web Application uygulamaların yayını bu yöntem ile yapılır.



# 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 tarafı çalışabilirler.

Bu modül tamamlandıktan sonra;

- Web Form yapısını ve bileşenlerini öğrenecek,
- Sunucu ve istemci tarafı kontrollerin farklarını öğrenecek,
- Web kontrollerini tanıyacak,
- ViewState vePostBack kavramlarını öğreneceksiniz.

## Konu 1: Web Form Bileşenleri



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

Web formlar, `.aspx` uzantılı arayüz dosyası ve `.aspx.vb` uzantılı kod dosyalarından oluşur. Örneğin `default.aspx` isimli ASP.NET sayfasının sunucu tarafı Visual Basic .NET kodları `default.aspx.vb` 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 tasarlanabilir. Design sekmesi, Web kontrollerinin görsel olarak düzenlenmesini sağlar. HTML sekmesi ise, kontrollerin HTML kodlar ile eklenmesini sağlar.

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

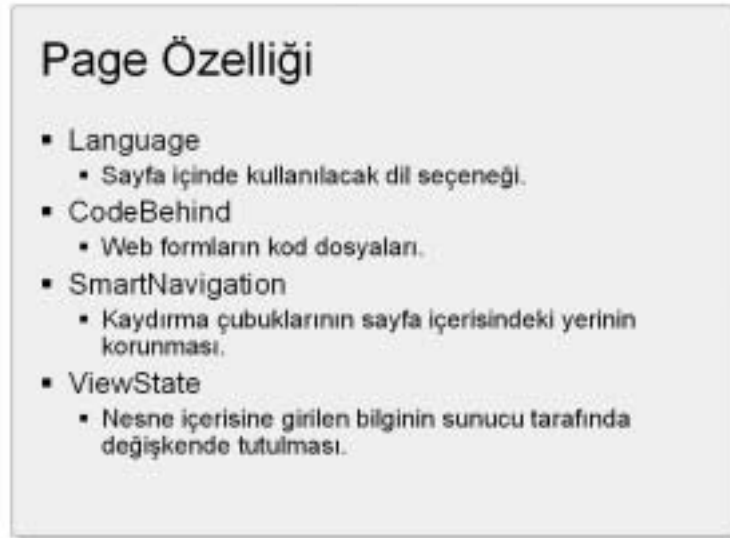
## Web Form Bileşenleri

- Web Formu
  - Visual Studio .NET ortamı aracılığı ile eklenen kontrollerin ve Visual Basic .NET kodlarının birleşimi
- Default.aspx
  - aspx uzantılı arayüz dosyası
- Default.aspx.vb
  - aspx.vb uzantılı kod dosyası

Web Formların genel özellikleri aşağıdaki gibidir.

- @Page Özelliği
- Body Özelliği
- Form Özelliği

## Page Özelliği



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

`<@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="vb" ...">
```

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

### CodeBehind

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

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

### SmartNavigation

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

```
<@Page Language="vb" CodeBehind="WebForm1.aspx.vb" _  
SmartNavigation="True" >
```

## ViewState

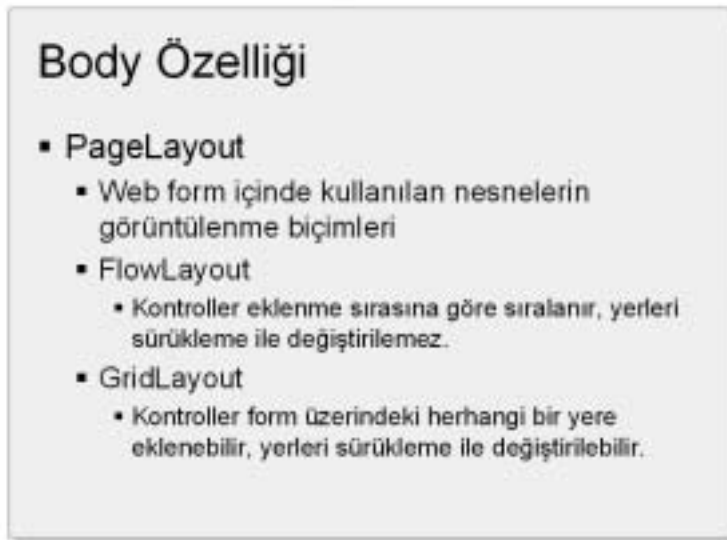
ASP.NET teknolojisi ile gelen yeniliklerden biridir. **EnableViewState** özelliği ile objenin içine girilen bilgi ne olursa olsun, sunucu bunu bir değişkende tutup tekrar kullanıcıya geri döndürür. Bu durum sunucuya gönderilen veriler üzerinde hata oluşması durumunda, bilgilerin kaybolmamasını sağlar.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



Web sayfasının ana bölümüdür.

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

`body` etiketi içinde **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 uygulamalarındaki Form görünümüne benzer.

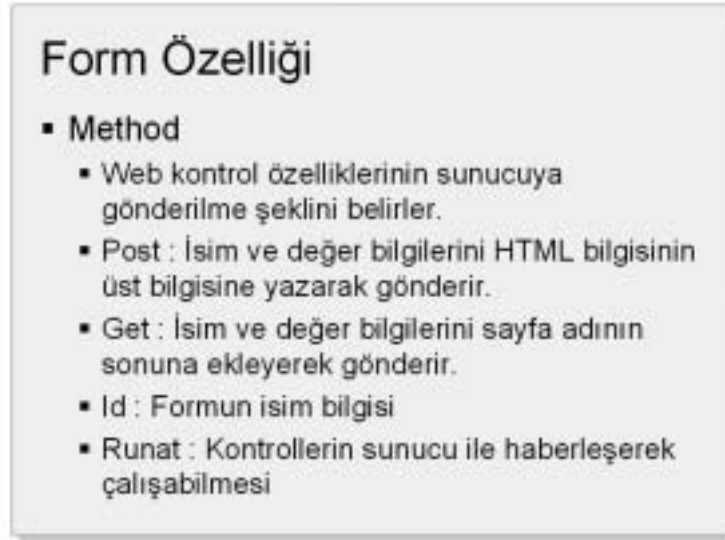
```
<body ms_positioning="GridLayout">
```

```
...
```

```
</body>
```



## Form Özelliği



Web kontrolleri gruplandırmak için kullanılır. Her web form için tek Form etiketi tanımlanır. Tüm kontroller `<form> ... </form>` etiketleri arasına 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 içinde, 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



Web sunucu üzerinde çalışan kontrollerdir. İki tür server kontrolü vardır:

- 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:Kontrolİsmi>` etiketi ile bildirilir. HTML kontrolleri ise **System.Web.UI.HtmlControls** isim alanında bulunur.

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



ASP.NET Web kontrolleri dört grupta listelenir:

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 kullanılıyor 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 7.1'de Html ve Standart sunucu kontroller gösterilmektedir.

**Tablo 7.1: Standart Kontroller**

| Web kontrol        | Html Kontrol                |
|--------------------|-----------------------------|
| <asp:Button>       | <input type=submit>         |
| <asp:CheckBox>     | <input type=checkbox>       |
| <asp:HyperLink>    | <a href="..."> </a>         |
| <asp:image>        |              |
| <asp:imageButton>  | <input type=image>          |
| <asp:LinkButton>   | Yok                         |
| <asp:Label>        | <span> </span>              |
| <asp:ListBox>      | <select size="5"> </select> |
| <asp:Panel>        | <div> </div>                |
| <asp:TextBox>      | <input type=text>           |
| <asp:RadioButton>  | <input type=radio>          |
| <asp:DropDownList> | <select> </select>          |
| <asp:Table>        | <table> </table>            |

## Doğrulama Kontrolleri

Kullanıcının girdiği değerleri kontrol etmek için kullanılır. Kontrolün yapılacağı 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.

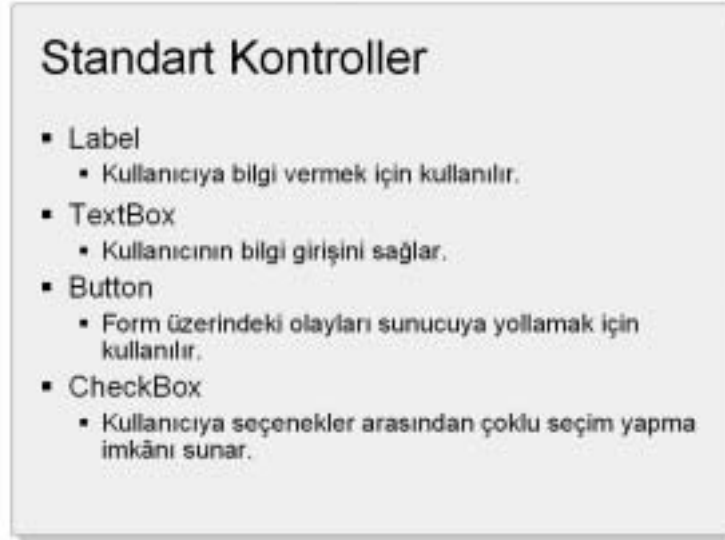
## Zengin Kontroller

**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**, **DataGrid** ve **Repeater** kontrolleri bu grupta yer alır.

## Konu 4: Standart Kontroller



### 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 kontrolüdür.

```
<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, **WebControl** 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 kontrolüdür.

Örnekte, **Dugme1** isimli düğme tıklandığında, **Label1** kontrolüne mesaj yazılır.

```
<asp:Button id="Dugme1" runat="server" Text="Tıklayınız"
    OnClick="Dugme1_Click" runat="server" />
<span id="Message" runat="server" />
```

```
Sub Dugme1_Click(Sender As Object, e As EventArgs)
    Message.InnerHtml="Beni Tıkladın."
End Sub
```

## CheckBox

**CheckBox**, kullanıcıya seçenekler arasından seçim yapma imkanı 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.

```
Sub Check_Clicked(Sender As Object, e As EventArgs)
    If checkbox1.Checked Then
        Message.InnerHtml="Seçili"
    Else
        Message.InnerHtml="Seçili Değil"
    End If
End Sub
```

```
<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 arasında bir tek seçim yapma imkânı sunar.
- **Hyperlink**
  - Sayfalar arası dolaşımı sağlar.
- **Image**
  - Sayfa içinde resim görüntülemek için kullanılır.
- **ImageButton**
  - Resimli button kontrolüdür.
- **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 içinden sadece bir **RadioButton** seçilebilir. Birden fazla seçeneğin işaretlenmesine izin verilmez. Onay kutusunun durumu **Checked** metodu ile takip edilebilir.

Örnekte, **RadioButton** kontrolleri arasında **muzik** isminde bir grup oluşturulmuştur. Bu grup içindeki **Pop**, **Jazz** ve **Classic** **RadioButton** kontrollerinden sadece bir tanesi seçilebilir. **BtnOnay** isimli button tıklandığında, seçilen **RadioButton** kontrolünün değeri **Message** isimli **Label** kontrolüne yazılır.

```
Sub BtnOnay_Clicked(Sender As Object, e As EventArgs)
    If Radio1.Checked Then
        Message.InnerHtml = "Seçiminiz" + Radio1.Text
    ElseIf Radio2.Checked
        Message.InnerHtml = " Seçiminiz " + Radio2.Text
    ElseIf Radio3.Checked
        Message.InnerHtml = " Seçiminiz " + Radio3.Text
    End If
End Sub
```

```
<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 7.2'de **Target** özelliğinin değerleri gösterilmektedir.

**Tablo 7.2: Target Özelliğinin Değerleri**

| Target Özelliği | Açıklama                  |
|-----------------|---------------------------|
| <b>_blank</b>   | Yeni sayfa                |
| <b>_self</b>    | Aynı sayfa içinde         |
| <b>_search</b>  | Arama sayfası görünümünde |

## Image

**Image**, sayfa içinde resim görüntülemek için kullanılır. **ImageUrl** özelliği ile görüntülenecek resim dosyası belirlenir. **ImageAlign** özelliği resmin hizalanması için kullanılır. **AlternateText** resme 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 düğme kontrolüdür.



Örnekte, **ImageButton** kullanımı gösterilmektedir.

```
Sub ImageButton_Click(Source As Object, e As _
    ImageClickEventArgs)
    Message.InnerHtml="Resimli Düğme kontrolünü Tıkladınız" _
    & "Koordinatlar: (" & e.X.ToString() & ", " & _
    e.Y.ToString() & ")"
End Sub

<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** argüman 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.

```
Sub LinkButton1_Click(sender As Object, e As EventArgs)
    Label1.Text="Link Button'a tıkladınız"
End Sub

<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 kontroldür.

### 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.

```
Sub Button_Click(sender As Object, e As EventArgs)
    Label1.Text = "Konuştuğunuz Dil " &
        dropdownlist1.SelectedItem.Text & "."
End Sub
```

```
<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" />
```

```
Sub Page_Load(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles MyBase.Load  
  
    Dim i As Integer  
  
    For i = 0 To 5  
        DropDownList1.Items.Add(i.ToString)  
    Next  
End Sub
```

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 imkanı sağlanır.

Örnekte, **ListBox** kontrolünün çoklu seçim özelliği kullanılmaktadır. **ListBox** kontrolü içinde seçilen tüm elemanlar **Label** kontrolüne yazdırılır.

```
Sub SubmitBtn_Click(ByVal sender As Object, _  
    ByVal e As EventArgs)  
    Dim item As ListItem  
    Message.Text = ""  
    For Each item In ListBox1.Items  
        If item.Selected = True Then  
            Message.Text += item.Text + " "  
        End If  
    Next  
End Sub
```

```
<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.

```
Sub Button1_Click(sender As Object, e As EventArgs)
```

```
    ' Label kontrolü oluşturalım
```

```
    Dim label As Label
```

```
    label = new Label()
```

```
    label.Text = "Etiket"
```

```
    label.ID = "Label1"
```

```
    Panel1.Controls.Add(label)
```

```
    Panel1.Visible = true
```

```
End Sub
```

```
<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 içine **Label** kontrolü eklenir ve görünür hale getirilir. **BackColor** 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, **Table** kullanımı gösterilmektedir.

```
Sub Page_Load(Sender As Object, e As EventArgs)
```

```
    ' Satır ve Sütun Oluşumu
```

```
    Dim nrow As Integer = 3
```

```
    Dim ncell As Integer = 2
```

```
    Dim i As Integer
```

```
Dim j As Integer
For j = 0 To nRows - 1
    Dim r As TableRow
    r = new TableRow()
    For i = 1 To nCells
        Dim c As TableCell
        c = new TableCell()
        c.Controls.Add(new LiteralControl("Satır " & _
            j.ToString() & ", hücre " & i.ToString()))
        r.Cells.Add(c)
    Next i
    Table1.Rows.Add(r)
Next j
End Sub
```

```
<asp:Table id="Table1" GridLines="Both"
HorizontalAlign="Center" Font-Name="Verdana"
    Font-Size="8pt" CellPadding=15 CellSpacing=0
Runat="server"/>
```

## Konu 5: Doğrulama(Validation) Kontrolleri

### Doğrulama Kontroller

- **RequiredFieldValidator**
  - Veri girilmesi zorunlu alanlarda kullanılır.
- **CompareValidator**
  - Girilen değeri, sabit değerle veya başka bir kontrole girilen değerle karşılaştırır.
- **RangeValidator**
  - Kontrol içine girilen değer iki sabit değer arasında olmasını sağlar.

Web forma girilecek verinin doğruluğunu kontrol etmek için sıklıkla 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 7.3'te doğrulama kontrolleri listelenmiştir.

**Tablo 7.3: Doğrulama Kontrolleri**

| Validation Kontrolleri            | Görevi   |
|-----------------------------------|--|
| <b>RequiredFieldValidator</b>     | Bir kontrol içine değer girilip girilmediğini kontrol eder. Veri girilmesi zorunlu alanlarda kullanılır. |
| <b>CompareValidator</b>           | Kontrol içine girilen değeri, sabit değerle veya başka bir kontrole girilen değerle karşılaştırır.       |
| <b>RangeValidator</b>             | Kontrol içine girilen değer, İki sabit değer arasında olmasını sağlar.                                   |
| <b>RegularExpressionValidator</b> | Bir kontrol içine girilen değer istenilen formatta girilmesini sağlar.                                   |
| <b>CustomValidator</b>            | Özel doğrulama kontrolü yazmayı sağlar.  |
| <b>ValidationSummary</b>          | Sayfada kullanılan tüm Validation kontrollerin, doğrulama hatalarını özet olarak görüntüler.             |

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** içinde listelenir.
- **Display:** Validation kontrolün nasıl görüntüleneceği bilgisini tutar. **Static**, **Dynamic** ve **None** değerlerini 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 içine 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** kontrolüne yirmi veya yirmiden büyük tamsayı 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="GreaterThanOrEqual">
```

```
Width="128px">
</asp:CompareValidator>
```

Doğrulama yapılacak Web kontrolü, başka bir Web kontrolü ile karşılaştırılacaksa **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 içine girilen değerin belirli bir değer aralığında olmasını sağlar. Doğrulama kontrollerinin ortak özelliklerine ek olarak **MinimumValue**, **MaximumValue** ve **Type** özellikleri vardır.

Örnekte **txtYas** kontrolüne girilen değerin, otuzbeş ile elli arasında olmasını sağlayan doğrulama işlemi yapılmaktadır. Bu özel karakterler Tablo 7.4'te gösterilmiştir.

```
<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 içerisine girilen değerin istenen formatta girilmesini sağlar.
- **CustomValidator**
  - Özel doğrulama kontrolü yazmayı sağlar.
- **ValidationSummary**
  - Sayfada kullanılan tüm Validation kontrollerinin, doğrulama hatalarını özet olarak görüntüler.

### RegularExpressionValidator

Kontrol içine girilen değerin istenen formatta girilmesini sağlar. **ValidationExpression** özelliğine girilen özel karakterler ile veri giriş formatı sağlanır. Bu özel karakterler Tablo 7.4'te gösterilmektedir.

Örnekte **RegularExpressionValidator** kontrolü ile **txtMail** metin kutusu için geçerli e-mail girişi 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>
```

**Tablo 7.4: Kontrol Karakterleri**

| Karakter     | Tanımı                                     |
|--------------|--|
| <b>a</b>     | Bir harf kullanımlarını zorunlu kılar.     |
| <b>1</b>     | 1 sayısı kullanılmak zorunda.              |
| <b>?</b>     | 0 veya 1 öge olmak zorunda.                |
| <b>*</b>     | 0'dan n'e kadar bir değer.                 |
| <b>+</b>     | 1'den n'e kadar bir değer.                 |
| <b>[0-n]</b> | 0'dan n'e kadar sayı değer dizisi.         |
| <b>{n}</b>   | N ile belirtilen değer uzunluğunda olmalı. |
| <b> </b>     | Farklı geçerli dizinler.                   |

**Tablo 7.4: Kontrol Karakterleri**

| Karakter | Tanımı  |
|----------|---|
| \        | Bir komut karakterini devam ettiren karakter. |
| \w       | Bir karakter olmak zorunda.                   |
| \d       | Bir rakam olmak zorunda.                      |
| \.       | Bir nokta olmak zorunda.                      |

Örnekte **ValidationExpression** özelliğine girilen özel karakterler ile e-mail formatı oluşturulmaktadır.

```
ValidationExpression=
"\w+([-+.]\w+)*@\w+([-+.\w+)*\.\w+([-+.\w+)*">
```

\w+ : En az bir karakter içeren metin anlamına gelir.

([-+.]) : -, +, . karakterlerinden herhangi biri anlamına gelir.

\* : 0'dan n'e kadar bir değer girilmesi gerektiği 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.

**Tablo 7.5: CustomValidator Kontrolünün Özellikleri**

| Özellik                         | Açıklama  |
|---------------------------------|---|
| <b>ClientValidationFunction</b> | İstemci fonksiyon ismini belirtir.  |
| <b>ControlToValidate</b>        | Doğrulama yapılacak kontrolü belirler.  |
| <b>Display</b>                  | <b>Text</b> özelliğine girilen hata mesajının nasıl görüntüleneceği belirtir.   |
| <b>EnableClientScript</b>       | İstemci script'leri aktif hale getirir. Varsayılan değer <b>True</b> 'dur.  |
| <b>Enabled</b>                  | Sunucu ve istemci taraflı script'leri aktif hale getirir. Varsayılan değer <b>True</b> 'dur.  |
| <b>ErrorMessage</b>             | Kontrol hata mesajını gösterir.   |
| <b>IsValid</b>                  | Kontrol işlemi başarı ile sonuçlanmışsa <b>True</b> , değilse <b>False</b> değerini döndürür.   |
| <b>Text</b>                     | Kontrol hata mesajını gösterir. <b>ErrorMessage</b> ve <b>Text</b> özelliği birlikte kullanılabilir. Bu durumda <b>Text</b> özelliğindeki mesaj görüntülenir. |

**ServerValidate** olayı ve **OnServerValidate** metodu ile sunucu taraflı kontroller tetiklenir.

Örnekte, sunucu taraflı metot oluşturulmaktadır:

```
Sub CustomValidator_SunucuKontrol _  
    (s as Object, e as ServerValidateEventArgs)  
    ' ...  
End Sub
```

**ServerValidateEventArgs** parametresinin iki özelliği vardır:

- **IsValid:** Bu özellik **True** ise kontrol içine 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 VB.NET kodları aşağıdaki gibidir:

```
Sub ServerKontrol(ByVal source As Object, _
    ByVal args As ServerValidateEventArgs)
    Dim num As Integer = Integer.Parse(args.Value)

    If args.Value = ((num Mod 2) = 0) Then
        args.IsValid = True
    Else
        args.IsValid = False
    End If
End Sub

Sub gonder_OnClick(ByVal sender As Object, _
    ByVal e As EventArgs)
    If Page.IsValid Then
        txtmessage.Text = "Sayfada Hata Yok."
    Else
        txtmessage.Text = "Sayfa Hatalı!"
    End If
End Sub
```

## 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 6: Zengin Kontroller

### Zengin Kontroller

- **AdRotator**
  - Web sayfaları üzerinde reklam resimleri görüntülemek için kullanılır.
  - Ayarları XML dosya içerisine kaydedilir.
- **Calendar**
  - Web sayfaları üzerinde takvim görüntülemek için kullanılır.

### AdRotator

**AdRotator**, Web sayfaları üzerinde reklam yayını yapmak için kullanılır. Reklam için kullanılan banner nesneleri XML dosya içine 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 içinde görüntülenecek resim dosyasını belirtir. **NavigateUrl:** Gidilecek sayfanın adres bilgisini belirtir.
- **AlternateText:** Resme alternatif metin göstermek için kullanılır.
- **Impression:** Banner resimlerinin uygulama süresini belirtir.
- **Keyword:** Banner nesneleri arasında filtreleme oluşturacak kategori adını belirler.

**dRotator** kontrolünün **AdvertisementFile** özelliği, reklam bilgilerinin bulunduğu XML dosyayı içerir. **KeywordFilter** özelliği ise reklamların filtrelenerek görüntülenmesini sağlar.

## Calendar

**Calendar**, Web sayıları üzerinde takvim göstermek için kullanılır.

Örnekte, **Calendar** kullanımı gösterilmektedir:

```
<asp:Calendar id="takvim" runat="server"/>
```

**Calendar** kontrolüne ait özellikler Tablo 7.6'da listelenmiştir.

**Tablo 7.6: Calendar Kontrolünün Özellikleri**

| Özellik                  | Açıklama  |
|--------------------------|---|
| <b>CellPadding</b>       | Hücreler ve kenarlıkları arasındaki boşluk değerini tutar.  |
| <b>CellSpacing</b>       | Hücreler arasındaki boşluk değerini tutar.  |
| <b>DayNameFormat</b>     | Gün isimlerinin görüntülenme biçimini tutar. <b>FirstLetter</b> , <b>FirstTwoLetters</b> , <b>Full</b> ve <b>Short</b> değerleri vardır. Varsayılan <b>Short</b> değeridir. |
| <b>FirstDayOfWeek</b>    | Haftanın ilk gününü belirtir.   |
| <b>NextPrevFormat</b>    | Next ve Previous linklerinin biçimini ayarlar. <b>CustomText</b> , <b>FullMonth</b> , <b>ShortMonth</b> değerleri alır. Varsayılan <b>CustomText</b> değeridir.             |
| <b>SelectedDate</b>      | Seçilen gün bilgisini tutar. Varsayılan değer günün tarihidir.  |
| <b>SelectionMode</b>     | <b>Calendar</b> nesnesinin seçim modunu belirler. <b>Day</b> , <b>DayWeek</b> , <b>DayWeekMonth</b> ve <b>None</b> değerleri alır. Varsayılan seçenek <b>Day</b> değeridir. |
| <b>ShowDayHeader</b>     | Gün isimlerini kolonların üzerinde görüntüler.  |
| <b>ShowGridLines</b>     | Günleri hücreler içinde görüntüler.   |
| <b>ShowNextPrevMonth</b> | Önceki ve sonraki ay linklerinin görüntülenmesini sağlar.   |
| <b>ShowTitle</b>         | Takvim başlığını görüntüler.  |
| <b>TitleFormat</b>       | Başlık yazısının biçimini belirtir.   |

Örnekte **Calendar** 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="#FFC0FF">
</DayHeaderStyle>
<WeekendDayStyle BackColor="Salmon"></WeekendDayStyle>
</asp:Calendar>
```

## Konu 7: 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 doldurduktan 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="VB" Debug="true" %>
<html>
<head></head>
<body>
  <form runat="server">
    Bilgiğiniz 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=VB runat="server">
```



```
Sub Page_Load(source As Object, e As EventArgs)
    If Not Page.IsPostBack Then
        lstFlowers.Items.Add(New ListItem("İngilizce"))
        lstFlowers.Items.Add(New ListItem("Almanca"))
        lstFlowers.Items.Add(New ListItem("Fransızca"))
        lstFlowers.SelectedIndex=0
    End If
End Sub
Sub showSelection(source As Object, e As EventArgs)
    lblMesaj.Text="Seçtiğiniz Dil " + _
    lstDiller.SelectedItem.Text
End Sub
</script>
```

**AutoPostBack** özelliğinin gereksiz yere kullanılması performansı olumsuz yönde etkiler.

## Konu 8: ViewState

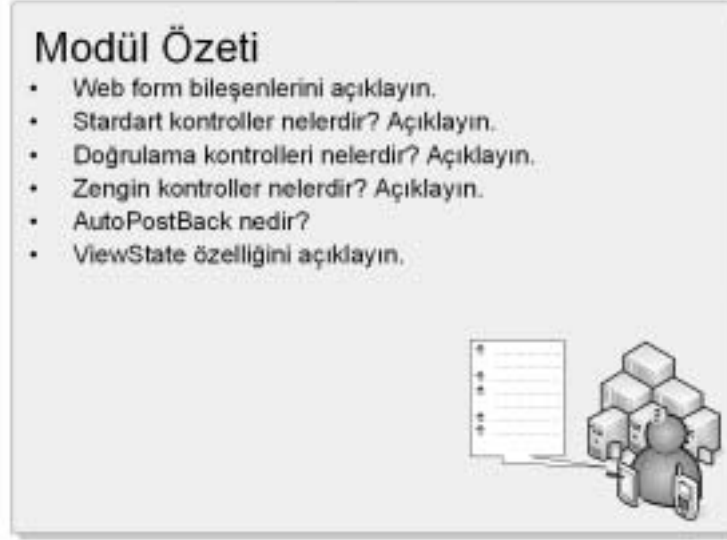


**ViewState**, kullanıcı ve sunucu arasında taşınan verilerin gizli bir alanda şifrelenerek saklanmasını sağlar. Forma ait tüm kontrollerin değerleri şifrelenir ve **VIEWSTATE** değişkeninde saklanır. Form sunucuya gönderildikten sonra bir hata oluşması halinde kullanıcıdan tekrar aynı verilerin girilmesi istenmez. Çünkü kullanıcının girmiş olduğu değerler bu gizli değişkeninde saklanır ve sayfa açılınca tekrar kullanıcıya sunulur.

Örnekte **ViewState** kullanımı gösterilmektedir:

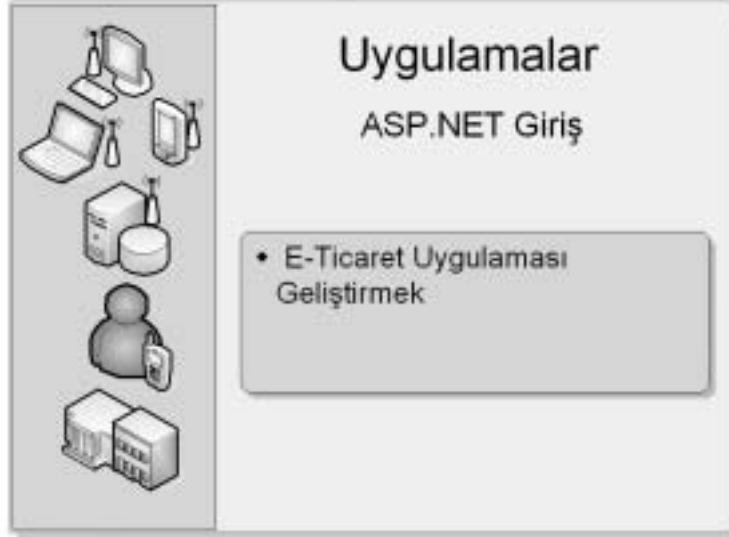
```
<input type="hidden" name="__VIEWSTATE" value="dDwtMTY5NzI1NjlxNz7Po5lYTU9gAdJkhGyy1Rw1gGcc+ia" />
```

## Modül Özeti



1. Web form bileşenlerini açıklayın.
2. Standart kontroller nelerdir? Açıklayın.
3. Doğrulama kontrolleri nelerdir? Açıklayın.
4. Zengin kontroller nelerdir? Açıklayın.
5. AutoPostBack nedir?
6. ViewState özelliğini açıklayın.

## Lab 1: E-Ticaret Uygulaması Geliştirmek



Bu uygulamada, e-ticaret uygulamasının Web formları tasarlanacaktır. E-ticaret uygulaması içinde müşterilerinin kendi kayıtlarını yapabilmesi için **UyeKayıt** formu tasarlanacaktır. Kayıt olan müşterilerin ürün satın alabilmesi için sisteme giriş yapmaları gerekir. Kayıtlı müşterilerin sisteme girişi **UyeGiris** formu ile sağlanır. Her iki form içinde standart ve doğrulama kontrolleri kullanılacaktır. Ayrıca kullanıcıyı yönlendirmek için **Giris**, **Kayıt** ve **Satis** isiminde 3 ayrı Web form tasarlanacaktır.

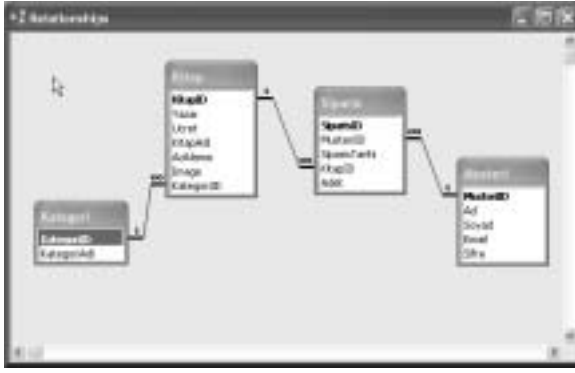
Bu lab tamamlandıktan sonra;

- Access veritabanına bağlantı oluşturabilecek,
- Web Formları tanıyacak,
- Standart Kontrolleri kullanabilecek,
- Doğrulama kontrollerini kullanabileceksiniz.

### Veritabanının Projeye Eklenmesi

Bu uygulamada kullanılacak **Course** veritabanı oluşturun.

1. Microsoft Access ile **KitapDb** isiminde bir veritabanı oluşturun.
2. Veritabanının tablolarını aşağıdaki diyagrama göre oluşturun.



RESİM 7.1.

3. Veritabanı içinde **EnCokSatanlar** isminde sorgu oluşturun.
4. Sorgunun içine 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.

### UyeKayıt Formunun Eklenmesi

ASPEticaret projesine **UyeKayıt** 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             |

| Kontrol – Kontrol İsmi                | Özellik               | Değer  |
|---------------------------------------|-----------------------|--|
| RequiredFieldValidator – rfvEmail     | ErrorMessage          | Soyadı boş geçemezsiniz                          |
|                                       | Text                  | *  |
|                                       | ControlToValidate     | txtEmail   |
| RequiredFieldValidator – rfvSifre     | ErrorMessage          | E-maili boş geçemezsiniz                         |
|                                       | Text                  | *  |
|                                       | ControlToValidate     | txtSifre   |
| RequiredFieldValidator – rfvSifre2    | ErrorMessage          | Şifreyi giriniz.                                 |
|                                       | Text                  | *  |
|                                       | ControlToValidate     | txtSifreDogrula                                  |
| RegularExpressionValidator – revEmail | ErrorMessage          | Doğrulama şifresini giriniz                      |
|                                       | Text                  | *  |
|                                       | ControlToValidate     | txtEmail   |
| CompareValidator– cvSifreDogrula      | ErrorMessage          | Hatalı Email                                     |
|                                       | Text                  | *  |
|                                       | Validation Expression | \w+([-\+.]\w+)*@\w+([-\+.]\w+)*\.\w+([-\+.]\w+)* |
| ValidationSummary – vsHata            | ControlToCompare      | txtSifre   |
|                                       | ControlToValidate     | txtSifre2  |
|                                       | ErrorMessage          | Şifreler uyumsuz                                 |
| Button – btnKaydet                    | Text                  | *  |
|                                       | Text                  | Kaydet   |

**RESİM 7.2.**

**UyeKayıt** Web formunun HTML kodları aşağıdaki gibi olacaktır:

```
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="UyeKayit.aspx.vb"
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="Visual Basic .NET 7.1"
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>
      </TABLE>
    </form>
  </body>
</HTML>
```

```

        <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:requiredfieldvalidato
        r 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:requiredfieldvalidato
        r 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."
        Display="Dynamic"
        ControlToCompare="txtSifre">*</asp:comparevalidator></TD>
    </TR>
    <TR>
        <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>

```



```

<%@ 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="vb" AutoEventWireup="false"
Codebehind="UyeGiris.aspx.vb"
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="Visual Basic .NET
7.1">
    <meta name="vs_defaultClientScript"
content="JavaScript">
    <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
  </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 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
                        Giriş</FONT></STRONG></P>
                    </TD>
                  </TR>

```

```

        <TR>
            <TD style="WIDTH: 66px">E-
Mail&nbsp;</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 **Giris** 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    |



**RESİM 7.4.**

Giris Web formunun 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="vb" AutoEventWireup="false"
CodeBehind="Giris.aspx.vb" Inherits="AspEticaret.Giris" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
  <HEAD>
    <title>Kayit</title>
    <meta name="GENERATOR" content="Microsoft Visual
Studio .NET 7.1">
    <meta name="CODE_LANGUAGE" content="Visual Basic .NET
7.1">
    <meta name="vs_defaultClientScript"
content="JavaScript">
    <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
  </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 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    |

**RESİM 7.5.**

**Kayıt** Web formunun HTML kodları aşağıdaki gibi olacaktır:

```
<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="Kayit.aspx.vb" 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>Kayit</title>
    <meta name="GENERATOR" content="Microsoft Visual
Studio .NET 7.1">
    <meta name="CODE_LANGUAGE" content="Visual Basic .NET
7.1">
    <meta name="vs_defaultClientScript"
content="JavaScript">
    <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
  </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 width="150" vAlign="top">
```

```

        <uc1:kategori id="Kategori1"
runat="server"></uc1:kategori></TD>
        <TD width="400" vAlign="top">
            <P><BR>
                Kayıt işlemi başarıyla tamamlandı.<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     | Değer        |
|------------------------|-------------|--------------|
| HyperLink – lnk1       | NavigateUrl | Default.aspx |
|                        | Text        | tıklayınız   |



RESİM 7.6.

Satis Web formunun HTML kodları aşağıdaki gibi olacaktır:

*BilgeAdam*



```

<%@ 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="vb" AutoEventWireup="false"
Codebehind="Satis.aspx.vb" Inherits="AspEticaret.Satis" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
  <HEAD>
    <title>Kayit</title>
    <meta name="GENERATOR" content="Microsoft Visual
Studio .NET 7.1">
    <meta name="CODE_LANGUAGE" content="Visual Basic .NET
7.1">
    <meta name="vs_defaultClientScript"
content="JavaScript">
    <meta name="vs_targetSchema"
content="http://schemas.microsoft.com/intellisense/ie5">
  </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 width="150" vAlign="top">
              <uc1:kategori id="Kategori1"
runat="server"></uc1:kategori></TD>
            <TD width="400" vAlign="top">
              <P><BR>
                Satış&nbsp;işlemi başarıyla
tamalandı.<BR>
                Devam etmek&nbsp;için
                <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>
```

## Modül 8: ASP.NET ile Kod Geliştirmek

### ASP.NET İle Kod Geliştirmek

- Kod Yazmak
- İstemci Taraflı Olay Prosedürleri
- Sunucu Taraflı Olay Prosedürleri
- Sayfa Yaşam Döngüsü

Bu modülde Visual Studio .NET ortamı içinde 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 tanıyacaksınız.

## Konu 1: Kod Yazmak

### Kod Yazmak

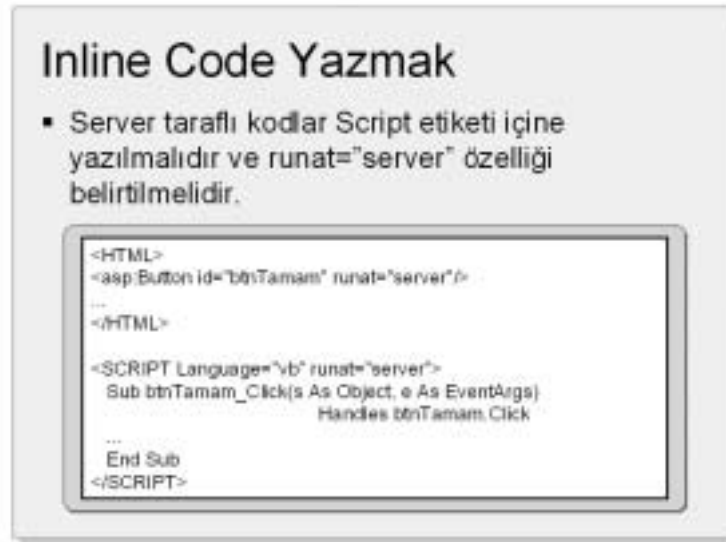
- Web Form'a kod ekleme yöntemleri:
  - Mixed Code
    - Web içeriği ile kod aynı sayfada.
  - Inline Code
    - Kod, aynı HTML dosyasında Script etiketi içerisinde.
  - Code-behind
    - Kod ve HTML içeriği farklı dosyalarda.

Web Form içinde 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 içinde yazılır. Bu metot pek tercih edilmez, çünkü okunması ve düzenlenmesi zordur.
- **Inline Code:** Web içeriği ile kod aynı sayfa içinde yer alır. ASP.NET kodu `Script` etiketi içine 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



Aynı .aspx dosyası içinde HTML kodu ve Visual Basic .NET kodu ayrı bölümlere yazılır. Bölümlerin ayrı tutulması okunabilirliği artırır. Server taraflı kodlar **Script** etiketi içersine yazılmalıdır ve **runat="server"** özelliği belirtilmelidir.

### Kod 8.1: Inline Kod Yazmak

```
<%@ Page Language="VB" %>
<html>
<head>
<title>My First Web Form</title>
<script runat="server">
  Sub Page_Load(Sender As Object , e As EventArgs )
    Message.Text = "Inline Kod Yazdık"
  End Sub
</script>
</head>
<body>
  <form runat="server">
    <asp:Label id="Message" runat="server" />
  </form>
</body>
</html>
```

## Code-Behind Kod Yazmak



Visual Studio .NET ortamının kullandığı varsayılan model code-behind tasarım modelidir. Programlama ve tasarım sayfaları ayrı tutularak çalışma mantığına göre ayırım yapılmış olur. Code-behind sayfaları, .aspx uzantılı sayfanın sonuna .vb eklenerek isimlendirilir. WebForm1.aspx sayfasının code-behind sayfası, WebForm1.aspx.vb şeklindedir.

Form üzerinde bir kontrol çift tıkladığında 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 gerekir.

### Kod 8.2: WebForm.aspx - Code-behind kod yazmak

```
<%@ Page Language="vb" AutoEventWireup="false"
    Codebehind="WebForm1.aspx.vb"
    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>
```

```
</form>
</body>
</HTML>
```

---

**Kod 8.3: WebForm.aspx.vb – Code-Behind Sayfası**

---

```
Imports System
Imports System.Web.UI
Imports System.Web.UI.WebControls
Imports System.Web.UI.HtmlControls

Public Class WebForm1
    Inherits System.Web.UI.Page

    Protected WithEvents txtAd As
        System.Web.UI.WebControls.TextBox
    Protected WithEvents btnGonder As
        System.Web.UI.WebControls.Button
    Protected WithEvents lblMesaj As
        System.Web.UI.WebControls.Label

    Private Sub Page_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
        response.write("selam")
    End Sub

    Private Sub btnGonder_Click(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles btnGonder.Click
        lblMesaj.Text = "Hoşgeldin " & txtAd.Text
    End Sub
End Class
```

**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, bilgisayarında işlenen olaylardır.
- İstemci taraflı olay prosedürleri hiçbir zaman sunucu kaynağını kullanmazlar.

Client-side olay prosedürleri, Web forma istekte bulunan kullanıcı bilgisayarında işlenen olaylardır. Kullanıcı tarafında 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 değer doğruluğunu kontrol etmek için client-side kod kullanılabilir.

Client-side kodlar `<Script>` blokları içinde 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 kullanmazken, server-side olay prosedürleri sunucu kaynaklarını kullanır.

Server-side olay prosedürleri için aşağıdaki tanımlama yapılır.

```
<SCRIPT language="vb" 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 gerekir.
- Olay prosedürlerine iki parametre girilmelidir:
  - **Sender**: Olayı tetikleyen kontrol nesnesidir.
  - **EventArgs**: Olaya özgü parametreleri içeren nesnedir.
- Olay Prosedürlerinde Kontrollerle Etkileşim

Visual Studio .NET içinde server-side olay prosedürleri iki adım ile oluşturulur. Birinci adım, olay üretecek kontrolü Web form üzerine eklemektir. İkinci adım ise, code-behind sayfasına olay prosedürünü eklemektir.

Olay prosedürleri kontrolün **ID** özelliğine girilen değerden faydalanarak oluşturulur. **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 içine **btn1** isminde bir **Button** yerleştirilmiştir. Eklenen **btn1** kontrolünün üretilen HTML kodu ve code-behind sayfasına eklenen olay prosedürü Kod 8.4'te belirtilmiştir.

### Kod 8.4: Button Kontrolüne Click Olayı ile İlişkilendirmek

```
<asp:Button id="btn1" runat="server"/>
```

```
' ...
```

```
Protected WithEvents btn1 as _  
System.Web.UI.WebControls.Button
```

```
Private Sub btn1_Click (ByVal sender As System.Object, _  
ByVal e As System.EventArgs) Handles btn1.Click
```

```
' ...  
End Sub
```

Server olay prosedürlerinin çalışmasını sağlamak için kontrollerin **WithEvents** anahtar sözcüğü ile tanımlanması gerekir.

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 Sub btn1_Click (ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btn1.Click
    Dim Mesaj as String = "Merhaba " & txtAd.Text
    lblMesaj.Text = Mesaj
End Sub
```

## Konu 4: Sayfa Yaşam Döngüsü

### Sayfa Yaşam Döngüsü

- **Page\_Init**
  - Sayfa başlatılır ve kontroller oluşturulur.
- **Page\_Load**
  - Sayfa yüklendiği 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 yüklendiği 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 Sub ListBox1_SelectedIndexChanged(ByVal sender _  
    As System.Object, ByVal e As System.EventArgs) _  
    Handles ListBox1.SelectedIndexChanged  
    Response.Redirect(ListBox1.SelectedValue)  
End Sub
```

```
<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 durumda veri yollayan tek kontrol **Button** kontrolüdür. Diğer kontroller için **AutoPostBack** özelliğinin **True** yapılması gerekir.

Örnekte **DropDownList** kontrolünün **AutoPostBack** özelliğine **True** değeri girilmiştir. Bu durum liste kutusundan seçilen değerın sunucuya gönderilmesini sağlar.

### Kod 8.7 DropDownList kontrolünün 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 Sub DropDownList1_SelectedIndexChanged _
    (ByVal sender As Object, ByVal e As System.EventArgs) _
    Handles DropDownList1.SelectedIndexChanged

    mesaj.Text = DropDownList1.SelectedItem.Value
End Sub
```

## Page.IsPostBack

**Page Load** olayı, sayfa yüklendiği zaman gerçekleşir. Sayfaya yapılan her istekte **Page Load** olayı içindeki kodlar çalışır. Aynı sayfanın her seferinde yenisinden 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.

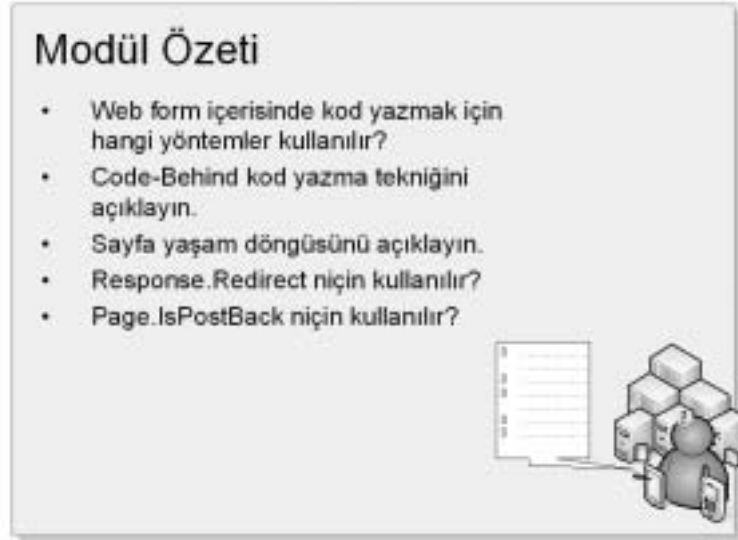
Örnekte **Page.IsPostBack** özelliğinin kullanımı gösterilmektedir:

### Kod 8.8: Page.IsPostBack

---

```
Private Sub Page_Load(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles MyBase.Load  
    'Put user code to initialize the page here  
    If Not Page.IsPostBack Then  
        'sadece sayfanın ilk yüklendiğinde çalışan istenilen alan  
    End If  
    'sayfa her yüklendiğinde çalışılan alan  
End Sub
```

## Modül Özeti



1. Web form içinde kod yazmak için hangi yöntemler kullanılır?
2. Code-Behind kod yazma tekniğini açıklayın.
3. Sayfa yaşam döngüsünü açıklayın.
4. Response.Redirect niçin kullanılır?
5. Page.IsPostBack niçin kullanılır?



## Lab 1: ASP.Net ile Kod Geliřtirmek



Bu uygulamada, Code Behind ve Inline kod yöntemlerinin kullanımını öğreneceksiniz. Ayrıca istemci taraflı script'lerin yazılımını öğreneceksiniz.

Bu lab tamamlandıktan sonra;

- Code Behind kod yazma yöntemini öğrenecek,
- Inline kod yazma yöntemini öğrenecek,
- İstemci taraflı script'lerin kullanımını öğreneceksiniz.

### Web Uygulaması Oluřturmak

Bu uygulamada kullanılacak ASP.Net Web Application projesini oluřturun.

1. File menüsündeki New alt menüsünü iřaretleyin ve Project komutunu tıklayın.
2. New Project ileti kutusundan ASP.Net Web Application řablonunu seřin.
3. Location metin kutusuna **http://localhost/AspCode** yazın.
4. Enter tuřuna basın.

### 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 |



RESİM 8.1.

## 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** düğmesinin **Click** olayını çalıştırır. Bu kodu <Body>..</Body> etiketleri arasına ekleyin.

```
<script language="vb" runat="server">
    Private Sub btn(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles btnInline.Click
        Response.Write("Inline")
    End Sub
</script>
```

3. **Test** Web formunun HTML bölümüne aşağıdaki kodları yazın. Bu kod, Java Script ile istemci taraflı kod tanımlar. Bu kodu <Head>..</Head> etiketleri arasına ekleyin.

```
<script language="javascript">
    function mesaj()
    {
        alert('Hello World')
    }
</script>
```

```
}
```

```
</script>
```

Fonksiyonu çağırmak 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()">
```



# 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 gerekir. Sadece HTML kullanımı statik (sabit) sayfalar geliştirmek için yeterlidir. Ancak içeriği kolayca şekillendirmeyi sağlayan CSS dili, istemci taraflı çalışan kodların yazılması için JavaScript – VBScript dilleri de Web uygulamalarını zenginleştirir.

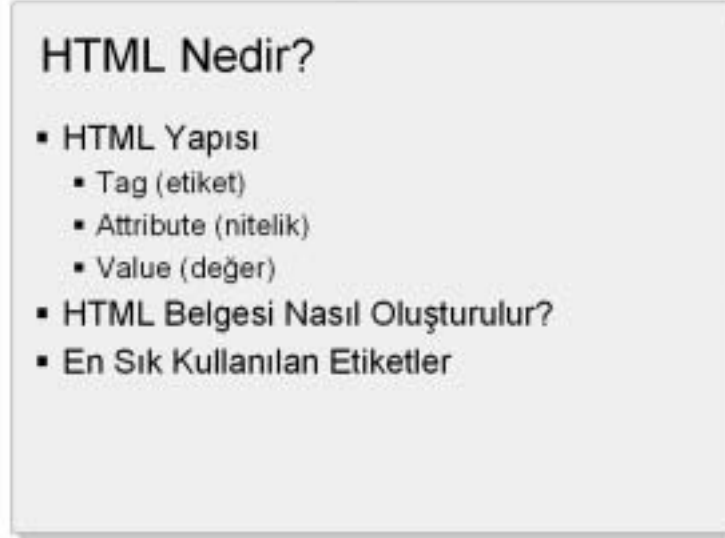
Bu modül tamamlandıktan sonra;

- 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



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 dokümanlar yalnızca istemci tarafında çalışır.

HTML dokümanlarının tümü ASCII karakterlerinden oluşur ve herhangi bir metin editöründe yazarak oluşturulabilir.

### HTML Yapısı

HTML dilini oluşturan bileşenler:

- Tag (etiket)
- Attribute (nitelik)
- Value (değer)

HTML kodlarının temeli etiketlerdir. Bir etiket, HTML'e yapılması istenen bir olayın bildirimini temsil eder. Etiketlerin içine attribute adı verilen değerler girilir. Etiketlere ait değişik özellikler, attribute nesnelerinde saklanır. Attribute değerleri, olaylara ait ayrıntıları tutar. Value ise, bir attribute değerinin davranacağı 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.

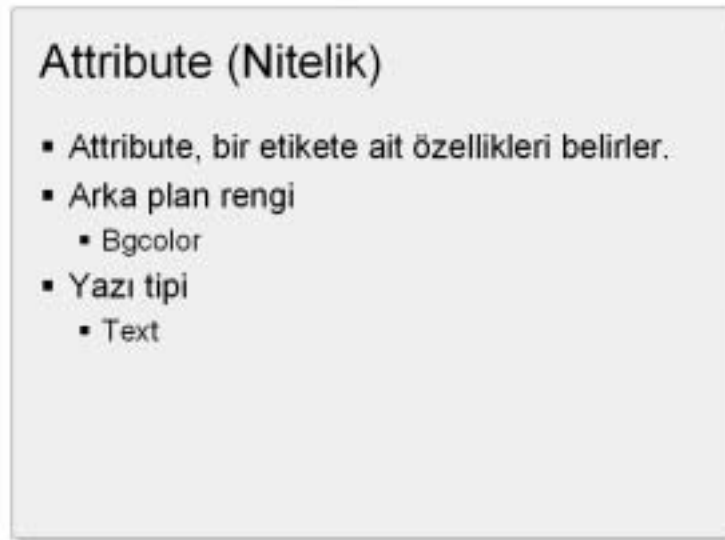
<Html>

Bir etikete ait açma ve kapatma ifadeleri vardır. Açma ifadesi <Html>,kapatma ifadesi ise </Html> şeklinde bildirilir. Yani açılan bir etiket, aynı isim başına / işareti getirilerek kapatılır. Örneğin <P> etiketi,bir paragraf açmaya yarar. </P> kapatma ifadesi kullanıldığında ise bu kapatma ifadesinin kullanıldığı yere kadar olan bölüm, paragrafın içine dahil edilir. Ancak bazı istisnalar da vardır. <Img> ve <Br> etiketi gibi bazı etiketler için bir kapatma ifadesi yoktur.

HTML etiketlerinde büyük harf-küçük harf duyarlılığı yoktur.



## Attribute



Nitelikler (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` niteliklerine değerler atanmalıdır.

Aşağıdaki örnekte siyah zemin üzerine beyaz yazı yazılmaktadır:

```
<Body bgcolor="#000000" text="#ffffff">
```

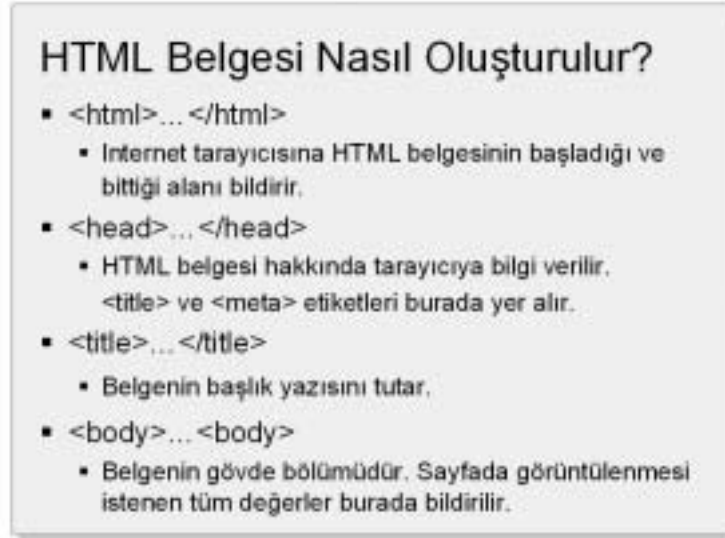
## Value

### 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 paragraflık metnin hizalanması, `<DIV>` etiketi kullanılarak yapılır. `DIV` etiketinin `align` attribute değeri, `left`, `right`, `center` ve `justify` değerlerinden birini alır.

## HTML Belgesi Nasıl Oluşturulur?



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 başladığı 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 Göüntülemek
- 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 başlıkları</h1>
  <h1>Bu başlık H1 tag'i ile oluştu</h1>
  <h2>Bu başlık H2 tag'i ile oluştu</h2>
  <h3>Bu başlık H3 tag'i ile oluştu</h3>
  <h4>Bu başlık H4 tag'i ile oluştu</h4>
  <h5>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 <br> 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.
- Metin bitimine `</a>` 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, `<ul>...</ul>` etiketleri arasında oluşturulur. Her bir liste nesnesi için `<li>` etiketi kullanılır.

Sıralı listeler, `<ol>...</ol>` etiketleri arasında oluşturulur. Her bir liste nesnesi için `<li>` 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 listelenecek metinler girilir. Bu etiketler arasına, başlığı tutan `<dt>` etiketi ve başlık altında görüntülenecek metni 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>
```



Listeleme ifadeleri iç içe kullanılabilir.

## Resim Görüntüleme

HTML sayfalarında resim görüntülemek için `<img>` etiketi kullanılır. `src` attribute değeri, görüntülenecek resmin adresini tutar.

Kod 9.6'da `<img>` kullanımı gösterilmektedir.

### Kod 9.6: Resim Görüntülemek

```
<a href="http://www.bilgeadam.com">
  
</a>
```

`img` etiketinin sonlandırma ifadesi yoktur.

## Tablolar

Satır ve sütunlardan oluşan yapılara tablo denir. Bir Web sayfası içinde, 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?

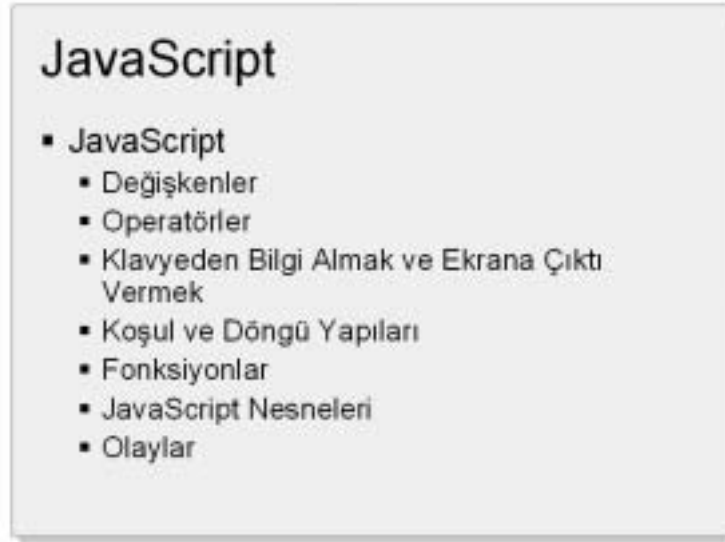
### Script Nedir?

- JavaScript
  - Netscape firması tarafından geliştirilmiştir.
  - C dili esas alınarak tasarlanmıştır.
  - JavaScript, istemci tarafında çalışır.
- VbScript
  - Microsoft firması tarafından geliştirilmiştir.
  - Diğer tarayıcılar tarafından desteklenmez.
  - Koşul ve döngü yapılarının kullanımı Visual Basic .NET dilindeki gibidir.

HTML dosyası içine gömülmüş kodlara script denir. Yorumlanması için Internet tarayıcısına ihtiyaç vardır. HTML dilinin karşılayamadığı bazı ihtiyaçlara çözüm üretmek için script'ler kullanılır.



## JavaScript



JavaScript dili, Netscape firması tarafından oluşturulmuştur. Yazım biçimi 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 gelmiştir. 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 tikladınız");
    }
  </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 içinde 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_doğru_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 9.10'da gösterilmiştir.

Genel kullanım biçimi ise şöyledir:

```
[koşul ifadesi] ? koşul_doğru_ise : koşul_yanlış_ise
```

### 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.

**Tablo 9.1: Aritmetik Operatörler**

| Operatör | Açıklama   |
|----------|--|
| +        | Sayısal değişkenleri toplar. <b>String</b> değişkenlerini birbirine ekler. |
| -        | Sayısal değişkenlerde çıkarma işlemi yapar.                                |
| *        | Sayısal değişkenlerde çarpma işlemi yapar.                                 |
| /        | Sayısal değişkenlerde bölme işlemi yapar.                                  |
| %        | Sayısal değişkenlerde mod alma işlemi yapar.                               |
| ++       | Sayısal değişkenlerde artma işlemi yapar.                                  |
| --       | Sayısal değişkenlerde azalma işlemi yapar.                                 |

Visual Basic .NET aritmetik operatörlerinden farklı olan `++` ve `--` operatörleri, C dili operatörlerindendir. Değişkeni bir artırma 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 artırılır ve ekrana 6 değeri yazılır
```

```
document.write(++x);
```

Değişkenin subfix kullanımında ise önce değer alınır, akış bir sonraki satıra geçtikten sonra değişkenin değeri bir artırılır.

#### Kod 9.12: Subfix ++ operatörü

```
var x = 5;
/* x değişkeni önce yazılır, sonra bir artırılır.
   Yani ekrana 5 yazılır. */
document.write( x++);
// Ekrana 6 değeri yazılır.
document.write(x);
```

#### Karşılaştırma Operatörleri

JavaScript kodları içinde de karşılaştırma işlemleri yapılabilir. Ancak bu operatörler Visual Basic .NET karşılaştırma operatörlerinden biraz farklıdır.

**Tablo 9.2: Karşılaştırma Operatörleri**

| Operatör | Açıklama  |
|----------|---|
| ==       | Eşit midir? operatörü. İki değer de birbirine eşit ise <b>true</b> sonucu verir.              |
| !=       | Eşit değil midir? operatörü. İki değer birbirine eşit değilse <b>true</b> sonucunu verir.     |
| <        | Küçük operatörü. Sol taraf değeri, sağ taraf değerinden küçükse <b>true</b> sonucunu verir.   |
| >        | Büyük operatörü. Sol taraf değeri, sağ taraf değerinden büyük ise <b>true</b> sonucunu verir. |
| <=       | Küçük eşittir operatörü.  |
| >=       | Büyük eşittir operatörü.  |

İki değer in eşitliğinin karşılaştırılması için == operatörü kullanılır.

```
if (a == b) {
    document.write("a ile b değişkeni eşit")
}
```

İki değer in 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.

**Tablo 9.3: Mantıksal Operatörler**

| Operatör          | Açıklama  |
|-------------------|---|
| <b>&amp;&amp;</b> | <b>And</b> (ve) operatörü. İki tarafta belirtilen ifadeler <b>true</b> ise, sonuç olarak <b>true</b> değerini döndürür.         |
| <b>  </b>         | <b>Or</b> (veya) operatörü. İki tarafta verilen ifadelerden en az birinin doğru olması durumunda <b>true</b> değerini döndürür. |
| <b>!</b>          | <b>Not</b> operatörü: Koşulun yanlış olması durumunda <b>true</b> değerini verir.   |

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 Almak ve Ekrana Çıktı Vermek

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 yazdırmak için **write** komutu kullanılır.

```
document.write("Yazılmak istenen değişkene ilişkin
açıklama", degisken);
```

Görüldüğü gibi **write** komutu **document** fonksiyonuyla birlikte kullanılır.

## Koşul ve Döngü Yapıları

Programlamanın 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 yaptırmak için kullanılan döngülerin JavaScript dilindeki kullanımı tamamen C dilinin yapısına göre tasarlanmıştır.

**for** döngüsünün genel kullanım biçimi aşağıdaki gibidir:

```
for(başlangıç_değeri; döngü_ifadesi; değişecek_değişken_adı)  
{  
    //yapılacak işlemler  
}
```

---

**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 "ifade1":  
        // ifade1 koşulu doğru ise yapılması istenenler  
        break; //break ile diğer koşulların da çalışması  
            //engellenir ve döngüden çıkılır.  
    case "ifade2:"  
        //ifade2 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** metodu 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.

**Tablo 9.4: Pencere Özellikleri**

| Özellik        | Açıklama                                |
|----------------|---|
| <b>MenuBar</b> | Menü çubuğunun görüntülenmesini sağlar. |
| <b>ToolBar</b> | Araç çubuğunun görüntülenmesini sağlar. |



**Tablo 9.4: Pencere Özellikleri**

| Özellik           | Açıklama   |
|-------------------|--|
| <b>Location</b>   | Adres çubuğunun görüntülenmesini sağlar.                   |
| <b>Status</b>     | Durum çubuğunun görüntülenmesini sağlar.                   |
| <b>Scrollbars</b> | Kaydırma çubuklarının görüntülenmesini sağlar.             |
| <b>Resizable</b>  | Penceresinin boyutlandırılmasını sağlar.                   |
| <b>Width</b>      | Açılan pencerenin pixel genişliğini belirtir.              |
| <b>Height</b>     | Açılan pencerenin pixel yüksekliğini belirtir.             |
| <b>Left</b>       | Ekranın sol noktası ile pencere arasındaki uzaklığı verir. |
| <b>Top</b>        | Ekranın üst noktası ile pencere arasındaki uzaklığı verir. |

**Kod 9.16: Yeni bir pencere açmak**

```
window.open("http://www.bilgeadam.com", "bilgeadam" ,
"menubar=no, toolbar=no, scrollbars=yes, location=yes,
width=300, height=300");
```

İnternet 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.

İnternet 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 9.5'te belirtilen değişkenler kullanılabilir.

**Tablo 9.5: Navigator (Tarayıcı) Nesnesinin Değişkenleri**

| Değişken İsmi      | Açıklama  |
|--------------------|---|
| <b>Appname</b>     | Tarayıcı adı  |
| <b>Appversion</b>  | Tarayıcı versiyonu                                    |
| <b>AppCodeName</b> | Tarayıcının kod adı                                   |
| <b>UserAgent</b>   | Tarayıcının sunucuya kendini tanıtırken verdiği isim. |

**Kod 9.17: Tarayıcı nesnesi ile bilgi almak**

```
<html>
<head>
<title>Browser'ımızı tanıyalım</title>
<METAcontent=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 hareketlere olay denir. Bu olaylar ise **onClick**, **onmouseover**, **onmouseout**, **onsubmit**, **onreset**, **onchange**, **onload**, **onunload**, **onerror**, **onabort**, **onfocus**, **onblur** olarak belirtilebilir.

### onClick

İnternet 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 tikla()
{
    alert("Tıklama işlemi gerçekleşti...");
}
<input type="button" name="tikla" value="tikla"
onClick=tikla()>
```

Düğme tıklanıp bırakıldığında, **onClick** olayı tetiklenir ve bu olayla ilişkilendirilen **tikla** fonksiyonu devreye girer. **alert** komutu ile ekrana bir mesaj kutusu çıkar.

Ayrıca **ondblclick**, çift tıklama olayını tetikler.

**onMouseOver, onMouseOut**

Fare nesnenin üzerindeyken **onMouseOver**, fare nesne üzerinden ayrılınca **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 çıktınız." ;
}

<a href="http://www.google.com"
    onMouseOver = nesneUzerinde()
    onMouseOut = nesneDisinda(> Google
</a>
```

**onSubmit**

Web sayfalarında 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** düğmelerinden 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>Istanbul
      <option>Ankara
      <option>Antalya
    </select>
  </p>
</form>

```

Sunulan seçeneklerden herhangi biri işaretlendiğinde uyarı penceresi çıkar.

### onLoad, onUnload

**onLoad** olayı sayfaya giriş yapıldığında gerçekleşir. **onUnload** olayı sayfadan çıkıldığında 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 içindeki **language** niteliğine **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 çağrılabilir.

---

#### 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 sehir: " & 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 içinde 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 özellikleri bildirilebilir.

```
<p style="font: 12pt arial"> Bu paragraftaki tüm yazılar  
arial 12 stilidedir.</p>
```

### Gömülü (Embedded)

Gömülü stiller, HTML sayfasının `Head` etiketi içinde 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>
```

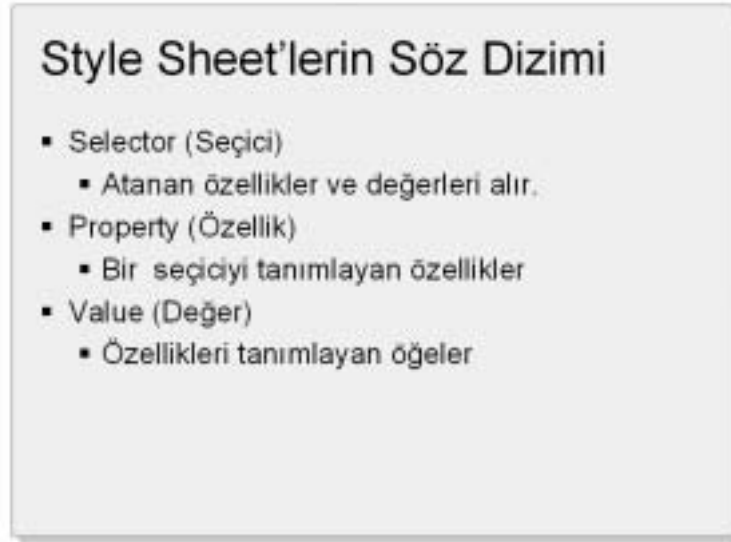
**Bağlantılı (Linked)**

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 içinde yapılır.

```
<link rel=stylesheet href="stil.css" type="text/css">
```

## Style Sheet'lerin Söz Dizimi



Stil nesnelerinin söz dizimi, HTML söz dizimine benzer yapıdadır. Stil nesnelerinin belirli kısımları vardır:

- **Seçici (Selector):** Atanan özellikler ve değerleri alır. **H1** ve **P** gibi HTML etiketlerine benzer.
- **Özellik (Property):** Bir seçiciyi tanımlar. **P** paragraf etiketine verilen özellikler o seçiciyi tanımlar. Kenar boşlukları, font ve arka plan değerleri birer özellik öğesidir.
- **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 görünüme sahip olabilir.

En çok kullanılan yazı özellik ve değerleri Tablo 9.6'da listelenmiştir.

**Tablo 9.6: En Çok Kullanılan Yazı Özellik ve Değerleri**

| Özellik ve Değer    | Açıklama   |
|---------------------|--|
| <b>Margin-left</b>  | Sol kenar boşluğunu belirlemek için kullanılır. Punto, inç, cm ve piksel cinsinden değer verilir.<br><b>{margin-left: 10px;}</b> |
| <b>Margin-right</b> | Sağ kenar boşluğunu belirlemek için kullanılır.  |
| <b>Margin-top</b>   | Üst kenar boşluklarını belirlemek için kullanılır.   |
| <b>text-indent</b>  | Bir yazı için girinti bilgisini belirler.  |
| <b>text-align</b>   | Yazının hizalanmasını sağlayan değeri tutar. <b>left</b> , <b>center</b> , <b>right</b>  |

**Tablo 9.6: En Çok Kullanılan Yazı Özellik ve Değerleri**

| Özellik ve Değer       | Açıklama  |
|------------------------|---|
| <b>text-decoration</b> | <b>underline, overline, line-through, none</b> değerleriyle yazıya şekil verir.         |
| <b>text-transform</b>  | Yazının büyük veya küçük harflerle görüntülenmesini sağlar. <b>uppercase, lowercase</b> |

```

Body {
margin-left: 10px;
margin-right: 10px;
margin-top: 20px;
margin-bottom: 15px;
}

```

Font özellik ve değerleri Tablo 9.7’de belirtilmiştir.

**Tablo 9.7: Font Özellik ve Değerleri**

| Özellik ve Değer   | Açıklama  |
|--------------------|---|
| <b>font-size</b>   | Yazı büyüklüğünü belirler.                            |
| <b>color</b>       | Yazının rengini tutar.                                |
| <b>font-family</b> | Yazının tipini belirler.                              |
| <b>font-style</b>  | Yazının italikliğini belirler. <b>italic, normal.</b> |
| <b>font-weight</b> | Yazı kalınlığını belirler. <b>bold, normal</b>        |

```

p {
font-size: 20;
color: blue;
font-weight: bold;
font-style: italic;
font-family: Times New Roman;
}

```

Liste özellikleri ve değerleri Tablo 9.8’de belirtilmiştir.

**Tablo 9.8: Liste Özellikleri ve Değerleri**

| Özellik ve Değer             | Açıklama  |
|------------------------------|---|
| <b>list-style-type</b>       | Liste elemanlarının başına gelecek karakteri belirler. <b>disc, circle, square, decimal</b> |
| <b>lower (upper) - roman</b> | Liste elemanlarının başına küçük veya büyük Roma rakamları koyar.                           |
| <b>lower (upper) - alpha</b> | Liste elemanlarının başına küçük(büyük) harfler koyar.                                      |
| <b>none</b>                  | Liste elemanları için bir sembol almaz.   |

**Tablo 9.8: Liste Özellikleri ve Değerleri**

| Özellik ve Değer           | Açıklama  |
|----------------------------|---|
| <b>list-style-image</b>    | Liste imleri yerine resim kullanır.   |
| <b>list-style-position</b> | <b>inside:</b> Listenin ikinci satırını en soldan başlatır.<br><b>outside:</b> İkinci satırı bir öncekinin dikey hizasından başlatır. |

Background özellikleri ve değerleri Tablo 9.9'da belirtilmiştir.

**Tablo 9.9: Background Özellik ve Değerleri**

| Özellik ve Değer           | Açıklama  |
|----------------------------|---|
| <b>background-color</b>    | Arka plan renk değerini tutar.  |
| <b>background-image</b>    | Arka plan resminin yol bilgisini tutar.   |
| <b>background-repeat</b>   | Resmin x ve y koordinatları boyunca tekrarlanması bilgisini tutar.<br><b>repeat:</b> tüm yönlerde<br><b>repeat-x:</b> x ekseni boyunca<br><b>repeat-y:</b> y ekseni boyunca<br><b>no-repeat:</b> tekrar edilmez |
| <b>background-position</b> | <b>left:</b> Resmi pencerenin sol kenarına yaklaştırır.<br><b>right:</b> Sağ kenara yaklaştırır.<br><b>center:</b> Resmi ortalar.   |

```
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 çağrılmasını 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 belirlemede 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 ayrılmasını sağlar. Sayfadaki herhangi bir yazının diğer yazılardan farklı görünmesi istendiği 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:

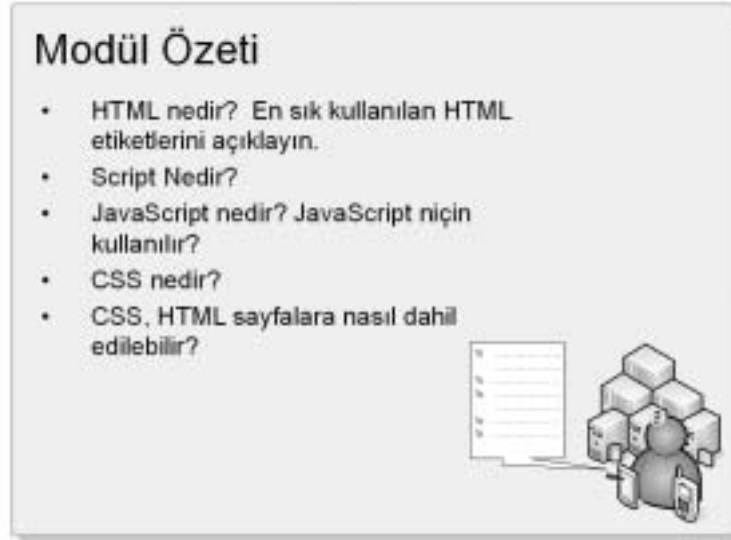
```
P.1 {
font: verdana;
font-size: 12pt;
line-height: 18pt;
}
```

1 sınıfındaki tüm paragraflar 12 punto **Verdana** fontuyla 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



1. HTML nedir? En sık kullanılan HTML etiketlerini açıklayın.
2. Script nedir?
3. JavaScript nedir? JavaScript niçin kullanılır?
4. CSS nedir?
5. CSS, HTML sayfalara nasıl dahil edilebilir?

## Lab 1: Web Programlamaya Giriş



Bu uygulamada, JavaScript .ile sanal klavye yapmasını öğreneceksiniz. Ayrıca JavaScript 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.

1. File menüsündeki New alt menüsünü işaretleyin ve Project komutunu tıklayın.
2. New Project ileti kutusundan ASP.Net Web Application şablonunu seçin.
3. Location metin kutusuna `http://localhost/ WebOrnek` yazın.
4. Enter tuşuna basın.

### Sanal Klavye Oluşturmak

WebOrnek projesine Sana1Klavye isiminde yeni bir HTML sayfa ekleyin.

**RESİM 9.1.**

Aşağıdaki kodları `<Body>..</Body>` etiketlerinin arasına yazarak HTML sayfaı tasarlayın:

```
<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

**Sana1Klavye** 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

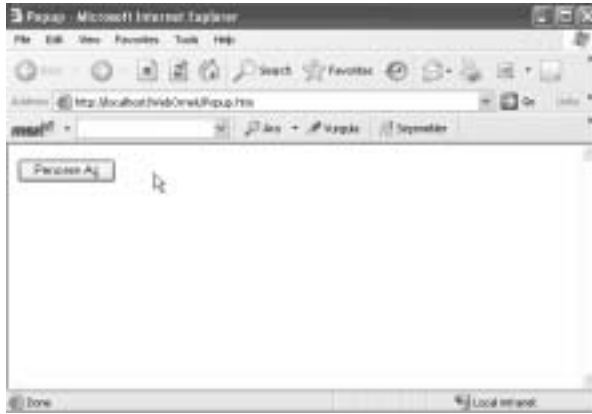
**WebOrnek** projesine **Popup** isminde yeni bir HTML sayfa ekleyin.

Aşağıdaki kodları **<Body>..</Body>** etiketlerinin arasına yazarak HTML sayfayı tasarlayın.

```
<INPUT id="Button1" type="button" onclick="pencereac()"
value="Pencere Aç" name="Button1">
```

**WebOrnek** projesine **acilan\_pencere** isminde yeni bir HTML sayfa ekleyin.

Aşağıdaki kodları **<Body>..</Body>** etiketlerinin arasına yazarak HTML sayfayı tasarlayın.

**RESİM 9.2.****RESİM 9.3.**

Burada açılan 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, JavaScript ile istemci tarafı fonksiyon eklemektedir. Bu kodu **<Head>...</Head>** etiketleri arasına ekleyin.

```
<script language="javascript">
    function pencereac()
    {
        window.open('acilan_pencere.htm','acilan','width=500,height=300')
    }
</script>
```



# Modül 10: Kullanıcı Kontrolleri Oluşturmak



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 isim alanı içinde tanımlanırlar.
- Aynı sayfa içinde birden fazla kez 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 içinde 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 isim alanı içinde tanımlanır. Bu durum kullanıldıkları Web formları ile oluşabilecek isim çakışmasını ortadan kaldırır.
- Kullanıcı kontrolleri, aynı sayfa içinde birden fazla kez 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 Ekleme

### Kullanıcı Kontrollerini Projeye Ekleme

- ✓ Solution Explorer penceresi açılır.
- ✓ Proje adı 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.vb uzantılıdır.

Projeye yeni bir kullanıcı kontrolü eklemek için aşağıdaki adımları takip edin:

1. Solution Explorer penceresini açın.
2. Proje adını sağ tıklayın.
3. Açılan penceredeki Add menüsünden Add Web User Control komutunu seçin.
4. Name metin kutusuna kullanıcı kontrolüne verilecek ismi girin.

## Kullanıcı Kontrollerini Projeye Ekleme

- ✓ 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 tasarlanır. Kullanıcı kontrol dosyaları `.ascx` ve bu kontrollere ait code-behind sayfası ise `.ascx.vb` 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 sürüklenip 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 içine eklemek için **@Register** ifadesi kullanılır. Bu ifade kullanıcı kontrollerinin Web forma bağlanmasını 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 kontrolü, aşağıda kod ile Web form içinde görüntülenir. Kullanıcı kontrolleri sunucu üzerinde çalıştığı için, **runat="server"** parametresi ile tanımlanmalıdır.

```
<deneme:Login id="Login1" runat="server" />
```

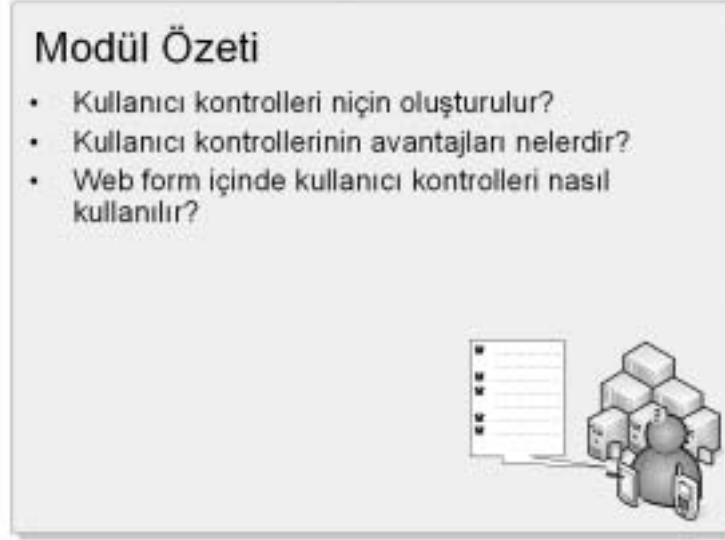
## Kullanıcı Kontrolü Tasarımı

- <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="vb" AutoEventWireup="false"
Codebehind="Header.ascx.vb"
Inherits="KullaniciKontrolleri.UserControls.Header" %>
<table width="100%" height="100%" bgcolor="#ffeeaa">
  <tr valign="middle">
    <td align="center">
      <asp:Label id="lblHeader" runat="server"
        Font-Bold="True" ForeColor="White"
        Font-Size="X-Large">Hoşgeldiniz
      </asp:Label>
    </td>
  </tr>
</table>
```

## Modül Özeti



1. Kullanıcı kontrolleri niçin oluşturulur?
2. Kullanıcı kontrollerinin avantajları nelerdir?
3. Web form içinde kullanıcı kontrolleri nasıl kullanılır?

## Lab 1: E-Ticaret Uygulaması Geliştirmek



Bu uygulamada, e-ticaret uygulamasının kullanıcı kontrolleri tasarlanacaktır. Bu uygulamada, bütün sayfalarda kullanılacak üst ve alt menü oluşturulacaktır. Ayrıca kategori isimli kullanıcı kontrolü ile tüm kategoriler listelenecektir. Kategori değerleri veritabanı içinden alınır.

Bu lab tamamlandıktan sonra;

- Kullanıcı kontrolü oluşturabileceksiniz.

### Kullanıcı Kontrollerin Eklenmesi

AspEticaret isimli projeyi açın.

### Üst Kontrolünün Eklenmesi

ASPEticaret projesine Üst isminde yeni bir kullanıcı kontrolü ekleyin.

Kullanıcı kontrolü içine 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 |

| Kontrol – Kontrol İsmi | Özellik | Değer     |
|------------------------|---------|-----------|
|                        | Text    | Üye Kayıt |
| LinkButton – btnCikis  | Text    | Çıkış     |
|                        | Visible | False     |
| Label – lblAd          |         |           |



### RESİM 10.1.

Ust isimli kullanıcı kontrolünün HTML kodları aşağıdaki gibi olacaktır:

```
<%@ Control Language="vb" AutoEventWireup="false"
CodeBehind="Ust.ascx.vb" 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">Yazilim Uzmani&nbsp;Kitapevi</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>
```

Ust isimli kullanıcı kontrolünün code-behind kodları aşağıdaki gibi olacaktır:

```
Imports System.Data.OleDb

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    If Session("user") <> "" Then
        lblAd.Text = "Bay / Bayan : " & Session("ad") & " " & Session("soyad")
        btnCikis.Visible = True
    Else
        btnCikis.Visible = False
    End If
End Sub

Private Sub btnCikis_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCikis.Click
    Session.Abandon()
    btnCikis.Visible = False
    Response.Redirect("Default.aspx")
End Sub
```

## Alt Kontrolünün Eklenmesi

ASPEticaret projesine Alt isminde yeni bir kullanıcı kontrolü ekleyin.

Kullanıcı kontrolü içine tablodaki kontrolleri ekleyin.

| Kontrol – Kontrol İsmi | Özellik     | Değer                              |
|------------------------|-------------|------------------------------------|
| HyperLink – Link1      | NavigateUrl |                                    |
|                        | Text        | AnaSayfamYap                       |
| HyperLink – Link2      | NavigateUrl |                                    |
|                        | Text        | SikKullanilanlaraEkle              |
| HyperLink – Link3      | NavigateUrl | mailto:tamer.sahiner@bilgeadam.com |
|                        | Text        | Iletisim                           |

AnaSayfamYap | SikKullanilanlaraEkle | Iletisim

### RESİM 10.2.

Alt isimli kullanıcı kontrolünün HTML kodları aşağıdaki gibi olacaktır:



```

<%@ Control Language="vb" AutoEventWireup="false"
CodeBehind="Alt.ascx.vb" 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;|
      <asp:hyperlink id="Link2"
runat="server">SikKullanilanlaraEkle</asp:hyperlink>&nbsp;|
      <asp:hyperlink id="Link3" runat="server"
NavigateUrl="mailto:tamer.sahiner@bilgeadam.com">Iletisim</a
sp:hyperlink></P>
    </TD>
  </TR>
</TABLE>

```

## Yan Kontrolünün Eklenmesi

ASPEticaret projesine Yan isiminde yeni bir kullanıcı kontrolü ekleyin.



**RESİM 10.3.**

Yan isimli kullanıcı kontrolünün HTML kodları aşağıdaki gibi olacaktır:

```

<%@ Control Language="vb" AutoEventWireup="false"
CodeBehind="yan.ascx.vb" 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/YazilimUzmani.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 penceresini açın.
2. Proje adını sağ tıklayın.
3. Açılan penceredeki Add menüsünden Add New Item komutunu seçin.
4. Templates seçeneği içinden DataSet ögesini seçin.
5. Name metin kutusuna **dsBook** ismini girin.

## Bağlantı Oluşturulması

**KitapDB** veritabanı üzerinde işlem yapılması için bağlantı kurulması gerekir. Bu bağlantıyı Server Explorer'ı kullanarak oluşturun. Bu bağlantı ile veritabanı içindeki **Kategori** tablosu, **dsBook** isimli **DataSet** içine eklenecektir.

**KitapDb** uygulaması için yeni bağlantı oluşturmak:

1. Server Explorer penceresi üzerinde sağ 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 seçeneğini işaretleyin ve Next düğmesini tıklayın.
4. Açılan Connection sekmesinin görüntüsünü resimdeki gibi düzenleyerek OK düğmesini tıklayın.
5. Server Explorer penceresinden DataConnections seçeneğini işaretleyin.
6. Eklediğiniz bağlantı içinden Tables seçeneğini işaretleyin.
7. Tables içindeki **Kategori** tablosunu **dsBook** nesnesinin içine sürükleyin.

## Kategori Kontrolünün Eklenmesi

**ASPEticaret** projesine **Kategori** isminde yeni bir kullanıcı kontrolü ekleyin.

Kullanıcı kontrolü içine tablodaki kontrolleri ekleyin.



RESİM 10.4.

| Kontrol – Kontrol İsmi | Özellik     | Değer        |
|------------------------|-------------|--------------|
| Repeater – rptKategori | NavigateUrl | Default.aspx |



RESİM 10.5.

**Kategori** isimli kullanıcı kontrolünün HTML kodları aşağıdaki gibi olacaktır:

```
<%@ Control Language="vb" AutoEventWireup="false"
Codebehind="kategori.ascx.vb"
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:

```
Imports System.Data.OleDb  
  
Private Sub Page_Load(ByVal sender As System.Object, ByVal e  
As System.EventArgs) Handles MyBase.Load  
    Dim connStr As String =  
    "Provider=Microsoft.Jet.OleDB.4.0;Data Source=" &  
    Server.MapPath("KitapDb.mdb")  
    Dim conn As New OleDbConnection  
    conn.ConnectionString = connStr  
  
    Dim comm As New OleDbCommand  
    comm.CommandType = CommandType.Text  
    comm.CommandText = "Select * from Kategori"  
    comm.Connection = conn  
  
    Dim da As New OleDbDataAdapter  
    da.SelectCommand = comm  
  
    Dim ds As New dsBook  
    Try  
        conn.Open()  
        da.Fill(ds, "Kategori")  
        rptKategori.DataSource = ds.Kategori  
        rptKategori.DataBind()  
    Catch ex As Exception  
        Response.Write(ex.Message)  
    Finally  
        If conn.State = ConnectionState.Open Then  
            conn.Close()  
        End If  
    End Try  
End Sub
```

# Modül 11: ADO.NET ile Veriye Erişim

## ADO.NET ile Veriye Erişim

- Veri Bağlantılı Kontroller
- Connected ve Disconnected Uygulamalar Geliştirmek

Web uygulamaları ile veriye erişim, Windows uygulamalarına oldukça benzer. Ancak verileri listelemek için kullanılan kontrollerin çalışma yapısı farklılık gösterir. Veriye ulaşım ADO.NET nesneleri ile gerçekleşir.

Bu modül tamamlandıktan sonra;

- **Repeater**, **DataList** ve **DataGrid** gibi listeleme kontrollerini öğrenecek,
- Web uygulamalarında Connected ve Disconnected çalışma yapısını öğreneceksiniz

## Konu 1: Veri Bağlantılı Kontroller

### Veri Bağlantılı Kontrolleri

- CheckBoxList ve RadioButtonList Nesnelerini Kullanmak
- Repeater, DataList ve DataGrid Nesnelerini Kullanmak
- Placeholder Nesnesini Kullanmak

ASP.Net ile veritabanı içindeki veriyi görüntülemek ve düzenlemek için veri bağlantılı kontroller kullanılır. **ListBox** ve **DropDownList** kontrolleri dışında **CheckBoxList** ve **RadioButtonList** kontrolleri veri bağlantılı olarak çalışabilir. **Repeater**, **DataList** ve **DataGrid** kontrolleri veri listelemek için kullanılır.

## CheckBoxList ve 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 tanesinin 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** içindeki 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 içine yazdırılmaktadır.

```
Dim i As Integer
```

```
For i=0 To checkboxlist1.Items.Count - 1  
    If checkboxlist1.Items(i).Selected Then  
        lblMsg.Text &= checkboxlist1.Items(i).Text & "<br>"  
    End If  
Next
```

**CheckBoxList** kontrolü ile birden fazla seçim yapılabilir. Fakat **RadioButtonList** kontrolü ile sadece bir öge seçilebilir.

Örnekte **CheckBoxList** veya **RadioButtonList** kontrolü ile veritabanı bağlantısı gösterilmektedir.

```
Private Sub Page_Load(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles MyBase.Load  
  
    da.Fill(DataSet1, "Kitaplar")  
    CheckBoxList1.DataSource = DataSet1  
    CheckBoxList1.DataMember = "Kitaplar"  
    CheckBoxList1.DataTextField = "Kitap_baslik"  
    CheckBoxList1.DataValueField = "kitap_ISBN"  
  
    CheckBoxList1.DataBind()  
End Sub
```



## 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 içinde 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 | Ardeşık olarak gelen satırların birbirinden farklı olmasını sağlar.                     |
| HeaderTemplate          | Repeater kontrolünün başlığıdır. İstenen stil verilebilir.                              |
| FooterTemplate          | Repeater kontrolünün en altındaki alanıdır. Alt başlık olarak istenen stil verilebilir. |
| SeperatorTemplate       | Veritabanından gelen her bir satırın 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>
        <td>
          <asp:Label ID=Lbl2 Runat=server
```

```

        text='<%# Databinder.eval(container.dataitem,
        "contactname")%> ' >
    </asp:Label>
</td>
</tr>
</table>
</ItemTemplate>
<FooterTemplate>
    <b> BilgeAdam BTA</b>
</FooterTemplate>
</asp:Repeater>

```

**Repeater** kontrolünün **ItemTemplate** şablonunda, her kayıt için yapılacak gösterim şekli belirlenir. HTML etiketleri kullanarak çıktıya şekil verilebilir. **HeaderTemplate** şablonu **repeater** kontrolünün başlığının, **FooterTemplate** alt başlığın biçimini belirler.

**HeaderTemplate** içinde açılan bir **<table>** etiketi, **FooterTemplate** içinde **</table>** ifadesiyle kapatılabilir. **SeperatorTemplate** şablonu, kayıtlar arasında ayraç stili belirler.

Tablo 11.1'de **Repeater** kontrolünün şablonları açıklanmıştır.

**Tablo 11.1: Repeater Kontrolünün Şablonları**

| Şablon Adı                     | Açıklama  |
|--------------------------------|---|
| <b>ItemTemplate</b>            | Veritabanından gelecek satırların görüntüleneceği stilin belirlendiği alan.                   |
| <b>AlternatingItemTemplate</b> | Ardışık olarak gelen satırların birbirinden farklı olmasını sağlar.                           |
| <b>HeaderTemplate</b>          | <b>Repeater</b> kontrolünün başlığıdır. İstenen stil verilebilir.                             |
| <b>FooterTemplate</b>          | <b>Repeater</b> kontrolünün en altındaki alandır. Alt başlık olarak istenen stil verilebilir. |
| <b>SeperatorTemplate</b>       | Veritabanından gelen her bir satırı diğerinden ayıran şablondur.                              |

**<table>** **</table>** etiketleri ile kayıtların bir tablonun satırları biçiminde görüntülenmesini sağlar. **<tr>** **</tr>** etiketi arasında iki **<td>** **</td>** etiketi kullanır. Bu şekilde, bir satır bilgiyi iki kolona ayrılmış biçimde görüntülenmesini ister. 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 aşağıdaki ifade yazılarak veri bağlama işlemi gerçekleştirilir.

```

text='<%#Databinder.eval(container.dataitem,
    "companyname")%> '

```

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 şablona uygun biçimde görüntülenmesini sağlar.

## DataList



**DataList** kontrolü, **Repeater** kontrolünün daha gelişmiş halidir. Veri görüntülemek dışında, verilerin seçilip ve üzerinde güncelleme işlemleri yapmaya olanak sağlar.

**DataList** eklemek için aşağıdaki adımları takip edin. Örnekte **DataList** kullanımı gösterilmektedir.

1. Araç kutusundan **DataList** kontrolünü seçerek formun üzerine sürükleyip bırakın.
2. Kontrolü sağ tıklayın ve açılan menüden Edit Template komutunu 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** değerlerini verin. Footer alanına yine bir **Label** ekleyin ve **Text** özelliğine **Bilge Adam BTA** yazın.
4. Kontrol üzerinde tekrar sağ tıklayın ve Item Templates komutunu seçin. **ItemTemplate** ve **AlternatingItemTemplate** şablonuna ek olarak, **SelectedItemTemplate** ve **EditItemTemplate** şablonları bulunur. **ItemTemplate** alanında, görüntülemek istenilen alanları temsil edecek kontroller 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**

```

Dim connStr As String = "Provider=Microsoft.Jet.OLEDB.4.0;_
Data Source=" & Server.MapPath("./Stok.mdb")
Dim conn As New OleDbConnection(connStr)
Dim da As New OleDbDataAdapter("select * from kitaplar",_
conn)
Dim ds As New DataSet

Private Sub Page_Load(ByVal sender As Object, _
ByVal e As EventArgs) Handles MyBase.Load
    da.Fill(ds, "kitaplar")
    DataList1.DataSource = ds
    DataList1.DataBind()
End Sub

```

**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="Label2" runat="server" Width="65px"
            Font-Bold="True">Yazar</asp:Label>
        </HeaderTemplate>
        <FooterTemplate>
            <asp:Label id="Label3" runat="server" Font-
            Bold="True">Bilge adam Bta</asp:Label>
        </FooterTemplate>
        <ItemTemplate>
            <asp:Label id=Label5 runat="server" Width="300px"
            text='<%# databinder.eval(container.dataitem,
            "kitap_baslik")%>'> </asp:Label>
            <asp:Label id=Label4 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 görölüm **Table** biçimindedir.

**RepeatColumns** özelliđi ise verilerin kaç sütun halinde görüntüleneceđini belirler. **RepeatDirection** özelliđi ise tekrarlanan kayıtların alt alta veya yan yana sıralanarak görüntülenmesini sağlar.

**GridLines** özelliđi ise dikey ve yatay çizgilerle kayıt görüntülerini birbirinden ayırır.

**DataList** içinde görüntü formunu düzenlemek için properties penceresindeki görünüme ilişkin pek çok özellik sunulmuştur.

**SelectedItemTemplate** şablonu, listeden seçilen nesneye ait ayrıntıların görüntülenmesini sağlar.

**EditItemTemplate** şablonu, kullanıcının seçtiđi kayıt üzerinde düzenleme yapmasını sağlayan alana ait kodların girildiđi bölümdür.

**DataList** kontrolünün şablonları içine kullanılan kontrollere, formun üzerinden direkt erişilemez. Örneđin **DataList** içindeki bir **Button** kontrolünün **Click** olayına kod yazılamaz. **DataList** içinde kullanılan **Button** kontrolüne kod yazmak için, **DataList** kontrolünün **CommandName** özelliđi 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 yapmak
- Seçilen kayıdı silmek
- Seçilen kayıtları sayfalamak
- Seçilen kayıtları sıralamak

**DataGrid** kontrolü, **DataList** kontrolünden daha gelişmiş özelliklere sahiptir. Verileri sayfalama ve sıralama yeteneği sayesinde görüntüleme işlemleri özelleşmiştir. **DataGrid**, veritabanından alınan bir tablonun, tablo biçimi 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

```
Dim connStr As String = "Provider= Microsoft.Jet.OLEDB.4.0;_
    Data Source=" & Server.MapPath("./Stok.mdb")
Dim conn As New OleDbConnection(connStr)
Dim da As New OleDbDataAdapter _
    ("select * from kitaplar", conn)
Dim ds As New DataSet

Private Sub Page_Load(ByVal sender As Object, _
    ByVal e As EventArgs) Handles MyBase.Load

    da.Fill(ds, "kitaplar")
    DataGrid1.DataSource = ds
    DataGrid1.DataMember = "kitaplar"
    DataGrid1.DataBind()
End Sub
```



## 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.

1. Server Explorer panelinden yeni bir Access veritabanı bağlantısı oluşturun.
2. Bu veritabanından, kullanmak istediğiniz tabloyu sürükleyerek form üzerine bırakın. Formun alt penceresinde iki yeni nesne oluşacaktır. (**OleDbConnection1** ve **OleDbDataAdapter1**)
3. **OleDbDataAdapter1** nesnesini seçin ve Properties panelinden Generate DataSet komutunu verin.
4. Açılan pencerede Next düğmesi ile ilerleyin.
5. Araç kutusundan **DataGrid** kontrolünü sürükleyip forma bırakın.
6. Properties penceresinde DataSource alanına oluşturulan **DataSet** kontrolünün ismini, DataMember alanına, **DataSet** içine alınan tablolar-dan birini girin.
7. **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 durumda **DataGrid** verileri Grid görünümünde sunar. **GridLines** özelliğine **Both**, **Horizontal**, **Vertical** ve **None** değerlerinden biri atanabilir. **BackImageUrl** özelliği sayesinde **DataGrid** kontrolünde bir arka plan resmi görüntülenebilir.

## DataGrid Kontrolünde Kolon Oluşturmak

### DataGrid Kontrolünde Kolon Oluşturma

- **BoundColumn**
  - Kayıtları görüntüler.
- **HyperLinkColumn**
  - Kayıtları linkler şeklinde görüntüler.
- **TemplateColumn**
  - Kayıtları bir şablona uyarak görüntüler.

DataGrid kontrolü içinde çeşitli kolon türleri bulunur.

- **BoundColumn**
- **HyperLinkColumn**
- **TemplateColumn**
- **ButtonColumn**
- **EditCommandColumn**

**AutoGenerateColumns** özelliği, varsayılan durumda **True** değerini alır ve tablodan gelen kolonları değiştirmeden görüntüler.

### BoundColumn

**BoundColumn**, **DataGrid** kontrolünün varsayılan kolonudur. Kayıtları görüntüler. Veri kaynağından alınan tablodan sadece belirli kolonların görüntülenmesi istenirse, **BoundColumn** kontrolleri kullanılabilir. Kod 11.5'te veri kaynağından alınan tablonun istenilen kolonları görüntülenir.

**BoundColumn** ile sadece görüntülenmesi istenen kolonları **DataGrid** kontrolüne eklenir.

#### Kod 11.5: DataGrid içinde BoundColumn kullanımı

```
Private Sub Page_Load(ByVal sender As Object, _
    ByVal e As EventArgs) Handles MyBase.Load

    Dim connStr As String =
        "Provider=Microsoft.Jet.OLEDB.4.0; _
        Data Source=" & Server.MapPath("./Stok.mdb")
```

```

Dim conn As New OleDbConnection(connStr)
Dim cmdSelect As OleDbCommand("Select * From kitaplar", _
    conn)
conn.Open()
DataGrid1.DataSource = cmdSelect.ExecuteReader()
DataGrid1.DataBind()
conn.Close()
End Sub

```

## 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 durumda **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 içinde tanımlanmıştır. **DataField** özelliğinde ise kolon adı belirtilmiştir



**DataGrid** içinde görüntülenecek kolonları seçen **SQL** komutu tanımlanırken **select \* from ...** ifadesinden kaçınılmalıdır. Bu komut yerine sadece ihtiyaç duyulan kolonlar **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 ve Footer alanlarına görüntülenmesi istenilen yazıları, ve başlıkta

görüntülenecek resmi belirler. **HeaderText** alanına yazılan yazının görüntülenebilmesi için **DataGrid** kontrolünün **ShowFooter** özelliği **True** yapılmalıdır. Bu özellik varsayılan durumda **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="<img src=myImage.Gif>Başlık"
```

### 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 11.6: DataGrid içinde HyperLinkColumn kullanımı

```
Private Sub Page_Load(ByVal sender As Object, _
    ByVal e As EventArgs) Handles MyBase.Load

    Dim connStr As String =
    "Provider=Microsoft.Jet.OLEDB.4.0; _
        Data Source=" & Server.MapPath("./Stok.mdb")
    Dim conn As New OleDbConnection(connStr)
    Dim cmdSelect As OleDbCommand("Select * From _
        musteri", conn)
    conn.Open()

    DataGrid1.DataSource = cmdSelect.ExecuteReader()
    DataGrid1.DataBind()
    conn.Close()
End Sub
```

### 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 Yollamak

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 başlıkları (**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 11.7: DataGrid içinde HyperLinkColumn kullanımı

```

Private Sub Page_Load(ByVal sender As Object, _
    ByVal e As EventArgs) Handles MyBase.Load

```

```

    Dim connStr As String =
    "Provider=Microsoft.Jet.OLEDB.4.0; _
      Data Source=" & Server.MapPath("./Stok.mdb")
    Dim conn As New OleDbConnection(connStr)
    Dim cmdSelect As OleDbCommand("Select * From linkler", _
      conn)
    conn.Open()

    DataGrid1.DataSource = cmdSelect.ExecuteReader()
    DataGrid1.DataBind()
    conn.Close()
End Sub

```

### 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 ayrılır.

Kod 11.8'de **TemplateColumn** kullanımı gösterilmektedir. **TemplateColumn** alanında, kitaba ait yazar ve açıklama bilgileri görüntülenir. Ancak bu iki kolon bilgileri HTML kodlarıyla alınır.

### Kod 11.8: DataGrid içinde TemplateColumn kullanımı

```

Private Sub Page_Load(ByVal sender As Object, _
  ByVal e As EventArgs) Handles MyBase.Load

    Dim connStr As String =
    "Provider=Microsoft.Jet.OLEDB.4.0; _
      Data Source=" & Server.MapPath("./Stok.mdb")
    Dim conn As New OleDbConnection(connStr)
    Dim cmdSelect As OleDbCommand("Select * From kitaplar", _

```

```
        conn)
    conn.Open()

    DataGrid1.DataSource = cmdSelect.ExecuteReader()
    DataGrid1.DataBind()
    conn.Close()
End Sub
```

## 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 metod kolon üzerinde, düğme şeklinde görüntülenir. Örneğin “Sepete Ekle” gibi bir iş için **Button** kullanılır ve **ButtonColumn** içinde tanımlanır.

**ButtonColumn** alanı kullanarak **Select** ismindeki düğme tıklandığı zaman kontrolün arka plan rengi ve yazı kalınlığı değiştirilir. **UnSelect** seçildiğinde kontrol eski haline getirilir.

### Kod 11.9: DataGrid içinde ButtonColumn kullanımı

```
Private Sub Page_Load(ByVal sender As Object, _
    ByVal e As EventArgs) Handles MyBase.Load

    Dim connStr As String =
    "Provider=Microsoft.Jet.OLEDB.4.0; _
        Data Source=" & Server.MapPath("./Stok.mdb")
    Dim conn As New OleDbConnection(connStr)
    Dim cmdSelect As OleDbCommand("Select * From kitaplar", _
        conn)
    conn.Open()

    DataGrid1.DataSource = cmdSelect.ExecuteReader()
    DataGrid1.DataBind()
    conn.Close()
End Sub

Sub DataGrid1_ItemCommand( s As Object, _
    e As DataGridCommandEventArgs )
    If e.CommandName="select" Then
```



```

        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
    End If
End Sub

```

## 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** düğmesi tıklandığında, **DataGrid** kontrolünün **OnItemCommand** özelliğinde belirtilen **DataGrid1\_ItemCommand** isimli metot devreye girer. **ItemCommand** olayını tetikleyen **DataGrid1\_ItemCommand** isimli metot, ilgili işlemleri gerçekleştirir.

**Unselect** düğmesi tıklandığında ise, yine **OnItemCommand** özelliğinde tutulan **DataGrid1\_ItemCommand** metodu devreye girer. Ancak tıklanan düğmenin ismine göre yapılacak işlem belirlenir.

```

If e.CommandName="select" Then
    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
End If

```

e. `CommandName`, hangi düğmenin tıklandığını belirtir. Tıklanan düğmeye göre hangi metodun uygulanacağını belirler.

**ButtonColumn** özellikleri:

- **ButtonType**: `LinkButton` veya `PushButton`
- **CommandName**
- **DataTextField**
- **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 gerektirir.

`EditCommandColumn` kolonunun görüntülediği kayıt, düzenleme için kayıt seçen `DataGrid` nesnesinin `EditRowIndex` özelliğine göre değişir.

Düzenleme işleminin seçili olmadığı durumda bu kolonda `Edit` düğmesi gözükür. `Edit` seçildiği anda ise `Update` ve `Cancel` düğmeleri gözükür.

**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 olayı 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 durumda **False** olan **AllowSorting** özelliği **True** yapılır ve **SortCommand** olayını tetikleyecek bir metod yazılır.

#### Kod 11.10: DataGrid içinde Sıralama

```
Private Sub Page_Load(ByVal sender As Object, _  
    ByVal e As EventArgs) Handles MyBase.Load  
    If Not IsPostBack Then  
        BindDataGrid( "kitap_baslik" )  
    End If  
End Sub  
  
Sub BindDataGrid( strSortField As String )  
    Dim connStr As String =  
    "Provider=Microsoft.Jet.OLEDB.4.0; _  
    Data Source=" & Server.MapPath("./Stok.mdb")  
    Dim conn As New OleDbConnection(connStr)  
    Dim cmdSelect As OleDbCommand("Select * From Kitaplar _  
    Order By " & strSortField, conn )  
    conPubs.Open("Select * From kitaplar", conn)  
    conn.Open()
```

```

        DataGrid1.DataSource = cmdSelect.ExecuteReader()
        DataGrid1.DataBind()
        conn.Close()
    End Sub

```

```

Sub DataGrid1_SortCommand( s As Object, e As
DataGridSortCommandEventArgs )
    BindDataGrid( e.SortExpression )
End Sub

```

### 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 gerekir.

#### Kod 11.11: DataGrid içinde Sayfalama

---

```

Private Sub Page_Load(ByVal sender As Object, _
    ByVal e As EventArgs) Handles MyBase.Load
    If Not isPostBack Then
        BindDataGrid
    End If
End Sub

Sub BindDataGrid
    Dim connStr As String =
        "Provider=Microsoft.Jet.OLEDB.4.0; _
        Data Source=" & Server.MapPath("./Stok.mdb")
    Dim conn As New OleDbConnection(connStr)
    Dim da As New OleDbDataAdapter("Select * From Kitaplar _
        Order By Kitap_baslik", conn )

    Dim ds As New DataSet

    da.Fill( ds )

    DataGrid1.DataSource = ds

```

```
DataGrid1.DataBind()  
End Sub  
  
Sub DataGrid1_PageIndexChanged( s As Object, _  
    e As DataGridPageChangedEventArgs )  
    DataGrid1.CurrentPageIndex = e.NewPageIndex  
    BindDataGrid  
End Sub
```

## HTML

```
<asp:DataGrid ID="DataGrid1"  
    AllowPaging="True" PageSize="5"  
    OnPageIndexChanged="DataGrid1_PageIndexChanged"  
    CellPadding="3" Runat="Server" />
```

**DataGrid** kontrolünde sayfalama yapıldığında kayıtlar sayfalara ayrılır ve diğer sayfalara linkler verilir. **PageSize** özelliği, bir sayfada kaç kayıt görüntüleneceği bilgisini tutar.

Sayfamaya ait stiller, tasarım penceresinde **DataGrid** kontrolü 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 satırdaki tüm veriler **TextBox** kontrollü biçiminde görünür. Üzerinde düzenleme yapılması istenmeyen kolonda, **BoundColumn** alanının **ReadOnly** özelliğine **True** değeri verilmelidir.

**Update** düğmesi tıklandığında 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**

---

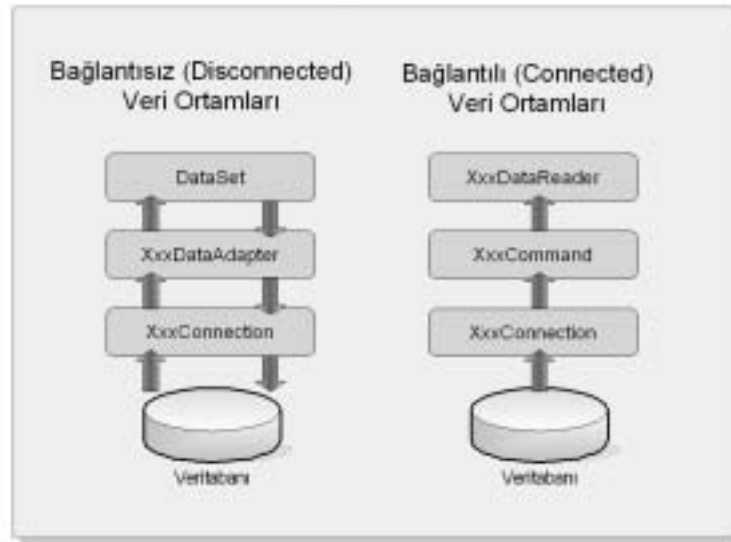
```
Private Sub Page_Load(ByVal sender As Object, _  
    ByVal e As EventArgs) Handles MyBase.Load  
    Dim i As Integer  
    Dim btnNewButton As Button  
  
    For i = 1 To 10  
        Placeholder1.Controls.Add( _  
            New LiteralControl("<p>Alan " & i & ": "))  
        Placeholder1.Controls.Add(New TextBox)  
    Next  
  
    btnNew = New Button  
    btnNew.Text = "Tıklayın!"  
    Placeholder1.Controls.Add(btnNew)  
End Sub
```

## Konu 2: Connected ve Disconnected Uygulamalar Geliřtirmek

### Connected ve Disconnected Uygulamalar Geliřtirme

- Namespace
- Connected Uygulama
- Disconnected Uygulama

ADO.NET ile veriye eriřmek iin Connected ve Disconnected veri eriřim yn-temi kullanılır. Bu yn-temler ile ASP.NET sayfalarında veri alıřveriři 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 konusuysa Disconnected bağlantı modeli uygulanmalıdır.



## Namespace

### Namespace

- **System.Data** isim alanı import edilir.
  - Imports System.Data
- **System.Data.OleDb** isim alanı 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** isim alanı import edilmelidir. Ayrıca Access veritabanına bağlantı için **System.Data.OleDb** isim alanı import edilmelidir.

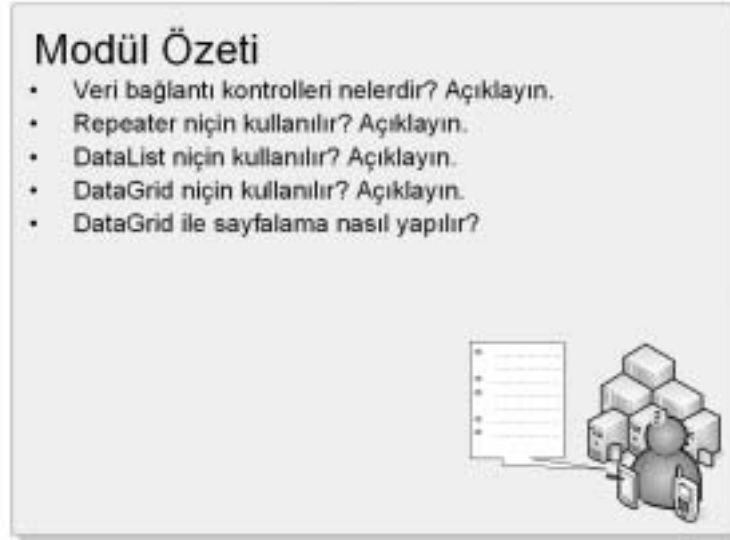
Code-behind sayfasında import işlemi, Windows uygulamalarında kullanılan biçimdedir.

```
Imports System.Data
Imports System.Data.OleDb
```

Inline kod yazımında <%@ %> ifadeleri arasında isim alanları import edilir.

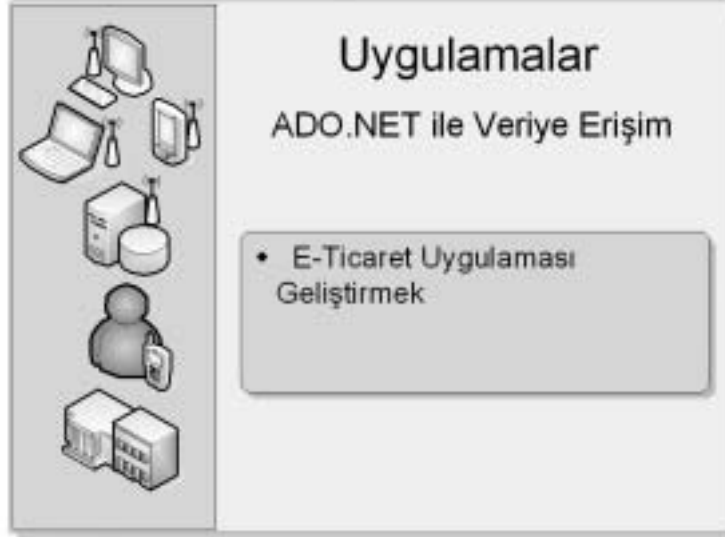
```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>
```

## Modül Özeti



1. Veri bağlantı kontrolleri nelerdir? Açıklayın.
2. Repeater niçin kullanılır? Açıklayın.
3. DataList niçin kullanılır? Açıklayın.
4. DataGrid niçin kullanılır? Açıklayın.
5. DataGrid ile sayfalama nasıl yapılır?

## Lab 1: E-Ticaret Uygulaması Geliřtirmek



Bu uygulamada, e-ticaret uygulaması ile Connected ve Disconnected veritabanı iřlemleri gerekleřtirilecektir. Bu uygulamada üye kayıt ve üye giriř iřlemlerini gerekleřtirebileceksiniz. Ayrıca kategoriye göre tüm kitapları listeyecek ve kitap satın alma iřlemini gerekleřtirebileceksiniz.

Bu lab tamamlandıktan sonra;

- Connect ve Disconnect veritabanı iřlemlerini öğreneceksiniz.
- DataSet içindeki veriyi Repeater, DataGrid ve DataList kontrollerine bağlayabileceksiniz.

### Connect Veritabanı İřlemleri

AspEticaret isimli projeyi açın.

### UyeKayıt Formu ile Veritabanı İřlemlerinin Yapılması

UyeKayıt Web formunu açın.

UyeKayıt Web formunun Code Behind kodları ařağıdaki gibi olacaktır:

```
Imports System.Data.OleDb
```

```

Private Sub btnKaydet_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnKaydet.Click
    Dim connStr As String =
    "Provider=Microsoft.Jet.OleDB.4.0;Data Source=" &
    Server.MapPath("KitapDb.mdb")

    Dim conn As New OleDbConnection
    conn.ConnectionString = connStr

    Dim comm As 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)
    Dim sonuc As Integer
    Try
        conn.Open()
        sonuc = comm.ExecuteNonQuery()
    Catch ex As Exception
        Response.Write(ex.Message)
    Finally
        conn.Close()
    End Try
    If sonuc = 1 Then
        Response.Redirect("Kayit.aspx")
    End If
End Sub

```

## UyeGiris Formu ile Veritabanı İşlemlerinin Yapılması

UyeGiris Web formunu açın.

UyeGiris Web formunun Code Behind kodları aşağıdaki gibi olacaktır:

```
Imports System.Data.OleDb
```

```

Private Sub btnGiris_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnGiris.Click
    ' Session("user") = "tamer"
    Dim connStr As String = _

```

```
"Provider=Microsoft.Jet.OleDB.4.0;Data Source=" _  
& Server.MapPath("KitapDb.mdb")  
  
Dim conn As New OleDbConnection  
conn.ConnectionString = connStr  
  
Dim comm As 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)  
Dim sonuc As Boolean  
Dim dr As OleDbDataReader  
Try  
    conn.Open()  
    dr = comm.ExecuteReader  
  
    If dr.HasRows = True Then  
        sonuc = True  
        If dr.Read = True Then  
            Session("user") = dr.Item("Email")  
            Session("ad") = dr.Item("Ad")  
            Session("soyad") = dr.Item("Soyad")  
            Session("musteriId") =  
dr.Item("MusteriID")  
        End If  
    Else  
        sonuc = False  
    End If  
  
    dr.Close()  
  
Catch ex As Exception  
    Response.Write(ex.Message)  
Finally  
    If conn.State = ConnectionState.Open Then  
        conn.Close()  
    End If  
End Try  
If sonuc = True Then  
    Response.Redirect("Default.aspx")  
Else
```

```

        lblMesaj.Text = "Hatalı kullanıcı adı veya
sifre"
    End If
End Sub

```

## KitapDetay Formunun Eklenmesi ve Veritabanı İşlemlerinin Yapılması

ASPEticaret projesine KitapDetay isiminde yeni bir Web form ekleyin.

Form üzerine, tablodaki kontrolleri ekleyin belirtilen özelliklerini ayarlayın.

| Kontrol – Kontrol İsmi | Özellik | Değer    |
|------------------------|---------|----------|
| Label – lblKitapAdi    |         |          |
| Label – lblYazarAdi    |         |          |
| Label – lblFiyat       |         |          |
| Label – lblAciklama    |         |          |
| Label – lblMesaj       |         |          |
| Image – imgResim       |         |          |
| TextBox – txtAdet      |         |          |
| Button – btnSatinAl    | Text    | Satın Al |



**RESİM 11.1.**

KitapDetay Web formunun HTML kodları aşağıdaki gibi olacaktır:

```

<%@ Register TagPrefix="uc1" TagName="yan" Src="yan.ascx" %>
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="KitapDetay.aspx.vb"
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="Visual Basic .NET 7.1"
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 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:
                <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>
    </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:



```
Imports System.Data.OleDb
```

```
Dim kID As String
```

```
Private Sub Page_Load(ByVal sender As System.Object,  
ByVal e As System.EventArgs) Handles MyBase.Load
```

```
    If Session("user") = "" Then
```

```
        Response.Redirect("Giris.aspx")
```

```
    End If
```

```
    kID = Request.Params("kID")
```

```
    'Response.Write(kID)
```

```
    Dim connStr As String = _
```

```
    "Provider=Microsoft.Jet.OleDB.4.0;Data Source=" _  
& Server.MapPath("KitapDb.mdb")
```

```
    Dim conn As New OleDbConnection
```

```
    conn.ConnectionString = connStr
```

```
    Dim comm As New OleDbCommand
```

```
    comm.CommandType = CommandType.Text
```

```
    comm.CommandText = _
```

```
    "Select * from Kitap Where KitapID =@kitapID"
```

```
    comm.Connection = conn
```

```
    comm.Parameters.Add("@kitapID", _  
Convert.ToInt32(kID))
```

```
    Dim dr As OleDbDataReader
```

```
    Try
```

```
        conn.Open()
```

```
        dr = comm.ExecuteReader
```

```
        If dr.Read = True Then
```

```
            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")
```

```
        End If
```

```
        dr.Close()
```

```
    Catch ex As Exception
```

```
        Response.Write(ex.Message)
```

```
    Finally
```

```
        If conn.State = ConnectionState.Open Then
```

```

        conn.Close()
    End If
End Try
End Sub

Private Sub btnSatinAl_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnSatinAl.Click
    If txtAdet.Text = "" Then
        lblMesaj.Text = "Adet Giriniz"
        Exit Sub
    End If

    Dim connStr As String =
"Provider=Microsoft.Jet.OleDB.4.0;Data Source=" &
Server.MapPath("KitapDb.mdb")

    Dim conn As New OleDbConnection
    conn.ConnectionString = connStr

    Dim comm As 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)
    Dim sonuc As Integer
    Try
        conn.Open()
        sonuc = comm.ExecuteNonQuery()
    Catch ex As Exception
        Response.Write(ex.Message)
    Finally
        conn.Close()
    End Try
    If sonuc = 1 Then
        Response.Redirect("Satis.aspx")
    End If

End Sub

```



```

        <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>
                                </TR>
                                </TABLE>
                                </P>
                                </TD>
                                <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 İçine DataTable Eklenmesi

1. Server Explorer penceresinden DataConnections seçeneğini işaretleyin.
2. Veritabanı tablo ve sorgularına erişmek için, oluşturduğunuz bağlantı içinden Views seçeneğini işaretleyin.
3. Views içindeki **EnCokSatanlar** sorgusunu **dsBook** nesnesinin içine sürükleyin.

**Default** Web formunun Code Behind kodları aşağıdaki gibi olacaktır:

```

Imports System.Data.OleDb

Private Sub Page_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
    If Page.IsPostBack = False Then
        Dim connStr As String =
        "Provider=Microsoft.Jet.OleDB.4.0;Data Source=" &
        Server.MapPath("KitapDb.mdb")

```

```

Dim conn As New OleDbConnection
conn.ConnectionString = connStr

Dim comm As 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

Dim da As New OleDbDataAdapter
da.SelectCommand = comm

Dim ds As New dsBook
Try
    conn.Open()
    da.Fill(ds, "EnCokSatanlar")
    dgEncokSatanlar.DataSource =
ds.Tables("EnCokSatanlar")
    dgEncokSatanlar.DataBind()
Catch ex As Exception
    Response.Write(ex.Message)
Finally
    If conn.State = ConnectionState.Open Then
        conn.Close()
    End If
End Try
End If
End Sub

```

## Kitap Formunun Eklenmesi ve Veritabanı İşlemlerinin 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 formunun HTML kodları aşağıdaki gibi olacaktır:

```

<%@ Register TagPrefix="uc1" TagName="Alt" Src="Alt.ascx" %>
<%@ Register TagPrefix="uc1" TagName="Ust" Src="Ust.ascx" %>

```



```

        <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 İçine DataTable Eklenmesi

1. Server Explorer penceresinden DataConnections seçeneğini işaretleyin.
2. Veritabanı tablo ve sorgularına erişmek için, oluşturduğumuz bağlantı içinden Tables seçeneğini işaretleyin.
3. Tables içindeki **Kitap** tablosunu **dsBook** nesnesinin içine sürükleyin.

**Kitap** Web formunun Code Behind kodları aşağıdaki gibi olacaktır:

```
Imports System.Data.OleDb
```

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
```

```

    Dim kategoriID As String =
Request.Params("KategoriID")
    Session("KategoriID") = kategoriID
    'Response.Write(kategoriID)
    Dim connStr As String =
"Provider=Microsoft.Jet.OleDB.4.0;Data Source=" &
Server.MapPath("KitapDb.mdb")
    Dim conn As New OleDbConnection
    conn.ConnectionString = connStr

    Dim comm As New OleDbCommand
    comm.CommandType = CommandType.Text
    comm.CommandText = "Select * from Kitap Where
KategoriID =@kID"
    comm.Connection = conn

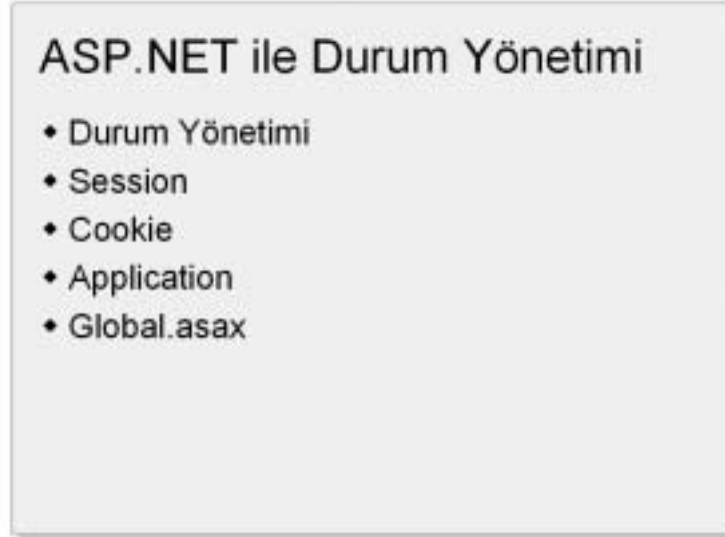
    comm.Parameters.Add("@kID", _
Convert.ToInt32(kategoriID))

    Dim da As New OleDbDataAdapter
    da.SelectCommand = comm

    Dim ds As New dsBook
    Try
        conn.Open()
        da.Fill(ds, "Kitap")
        dlKitap.DataSource = ds.Kitap
        dlKitap.DataBind()
    Catch ex As Exception
        Response.Write(ex.Message)
    Finally
```

```
        If conn.State = ConnectionState.Open Then  
            conn.Close()  
        End If  
    End Try  
End Sub
```

## Modül 12: ASP.NET ile Durum Yönetimi



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 kullanıcılardan 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 bilgisayar 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 bilgisayar 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 çıkana 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 çıkana kadar bu `SessionID` değişkeni sunucuda tutulur.

### Kod 12.1: SessionID

```
Private Sub Page_Load(ByVal sender As Object, _  
    ByVal e As EventArgs) Handles MyBase.Load  
    Response.Write(Session.SessionID)  
End Sub
```

### Kod 12.2: Session Nesnesini kullanmak

#### Login.aspx sayfası

```
Private Sub BtnGiris_Click(ByVal sender As Object, _  
    ByVal e As EventArgs) Handles BtnGiris_Click  
    Session("ad") = TxtAd.Text  
    Response.Redirect("Sayfam.aspx")  
End Sub
```

#### Sayfam.aspx sayfası

```
Private Sub Page_Load(ByVal sender As Object, _  
    ByVal e As EventArgs) Handles MyBase.Load  
    lblAd.Text = Session("ad") & "'Hoş Geldiniz"  
End Sub
```

ASP.NET uygulamalarının, **Session** değişkenlerine ait çeşitli özellikleri, **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 saklanır.
- **Cookieless:** Varsayılan durumda **False** değerini alır. **Session** değişkenlerinin kullanıcı bilgisayarında cookie içinde tutulmasını belirler. **True** değeri verildiğinde ise **SessionID** değeri URL'ye eklenerek kullanıcıya geri yollar.
- **Timeout:** **Session** değişkenlerinin yaşam süresini belirler. Varsayılan durumda 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** olayının kullanımı gösterilmektedir.

### Kod 12.3: Session\_Start

```
Sub Session_Start(ByVal Sender As Object, _
  ByVal e As EventArgs)
  Session("ArkaPlan") = "blue"
  Session("Yazi") = "gray"
End Sub
```



## Konu 2: Cookie

### Cookie

- Kullanıcı taraflı durum yönetimi için **cookie** değişkenleri kullanılır.
- ASP.NET **Cookie** değişkenlerini yönetirken **HttpCookie** sınıfını kullanır.
- **Cookie** değişkenleri için yazma ve okuma işlemleri yapılırken **Response** ve **Request** nesneleri kullanılır.

Kullanıcı taraflı durum yönetimi için cookie değişkenleri kullanılır. İnternet 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 yapılırken **Response** ve **Request** nesneleri kullanılır.

Örnek: Kullanıcı adı girilip “Cookie yap” düğmesi tıklanınca kullanıcı tarafında bir cookie oluşturulur. “Cookie oku” düğmesi tıklandığında ise oluşturulan cookie nesnesinden veri alınır.

|  |  |
|--|--|
| kullanıcı adı <input type="text" value="bilge"/> | kullanıcı adı <input type="text" value="bilge"/> |
| <input type="button" value="cookie yap"/>        | <input type="button" value="cookie oku"/>        |

**RESİM 12.1:** Cookie kullanımı.

**Kod 12.4: Cookie oluşturma ve okuma**

```
Private Sub btnYap_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnYap.Click
    ' Cookie oluşturmak için verilen direktif.
    Dim mycookie As New HttpCookie("sitem")
    ' Formdan Gelen Bilgileri Anahtarlara Yazar.
    mycookie("ad") = txtad.Text
    ' Cookie'nin Bitiş Süresi.
    mycookie.Expires = DateTime.Now.AddDays(30)
    ' Cookie'yi Gönder.
    Response.Cookies.Add(mycookie)
End Sub

Private Sub btnOku_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles btnOku.Click
    ' Cookie'yi oluşturur.
    Dim mycookie As HttpCookie
    ' Cookie'yi kullanıcı tarafından alır.
    myCookie = Request.Cookies("sitem")
    ' Cookie'den gelen değerlerle formu doldurur.
    txtad2.Text = mycookie("ad")
End Sub
```

**Cookie Türleri**

İki tür cookie vardır:

- **Temporary (Geçici).** 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 atılır.

## Cookie

- **Temporary (Geçici)**

- Tarayıcının hafızasında tutulur.
- Tarayıcı kapatıldığında tüm temporary cookie nesneleri kaybolur.

- **Persistent (Kalıcı):** Persistent cookie nesneleri, temporary cookie nesnelerinden farklı olarak hafızadan silineceği zamanı tutan bir değişkene sahiptirler. Tarayıcı, kalıcı bir cookie isteğinde bulunan bir sayfa açıldığında, cookie sabit diske yazılır. Bu tür cookie nesneleri kullanıcı bilgisayarında istenilen sürede tutulabilir.

## 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 garantisi yoktur. Kullanıcı sabit diskten bu dosyaları silmiş olabilir.

Cookie nesnelerinin diskte tutulacağının garantisi yoktur. Kullanıcı sabit diskten bu dosyaları silmiş olabilir.

## Konu 3: Application

### Application

- Session nesnesine benzer.
- Web uygulamasına giriş yapan ilk kullanıcıdan son kullanıcıya kadar devam eder.
- Tüm kullanıcılara 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 kullanıcıdan son kullanıcıya kadar devam eder. Tüm kullanıcılara 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. Değişken ile işiniz bittikten 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("degisken\_ismi") ifadesi kullanılır.

Session kullanıcıya özgü değişkenleri tutarken Application uygulamanın kendisine ait değişkenleri tutar.

ASP.NET Application değişkeni kullanırken HttpApplicationState sınıfını kullanır.

### Kod 12.5: Application Değişkeni

```
Sub Session_Start (ByVal sender as Object, _
    ByVal e as EventArgs)
    If (Application("ziyaret") = Nothing) Then
        Application("ziyaret") = 0
    End If
    Application.Lock()
    Application("Ziyaret") = Application("Ziyaret") + 1
    Application.Unlock()
    TextBox1.Text = "Ziyaret Sayısı: " & _
        Application("ziyaret").ToString()
End Sub
```

Bu örnekte her bir yeni session açıldığında, yani siteye her istek yapıldığında ziyaretçi sayısı birer artırılır.

Application değişkeni doldurulduktan sonra uygulama içinden çağırmak için Application("degisken\_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 12.6'da **Application\_Start** olayının kullanımına örnek verilmiştir.

---

**Kod 12.6: Application\_Start**

```
Sub Application_Start(ByVal sender As Object, _  
    ByVal e As EventArgs)  
    Application("ziyaret") = 0  
End Sub
```

## 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
  - İsteğe bulunan sayfa istemciye yollandığında
  - Koşullu application olayları gerçekleştiğinde

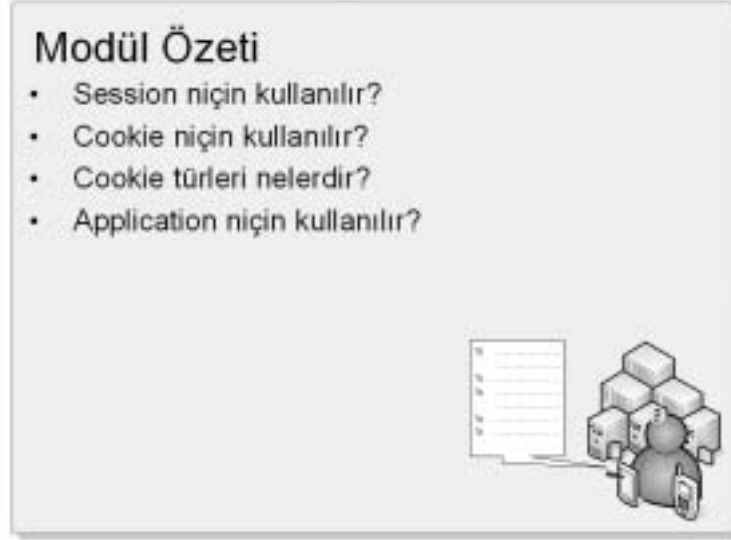
- Sayfaya bir istekte bulunulduğunda
- İsteğe bulunan sayfa istemciye yollandığında
- Koşullu Application olayları gerçekleştiğinde

Koşullu Application olayları ise Tablo 12.1’de listelenmiştir.

**Tablo 12.1: Koşullu Application Olayları**

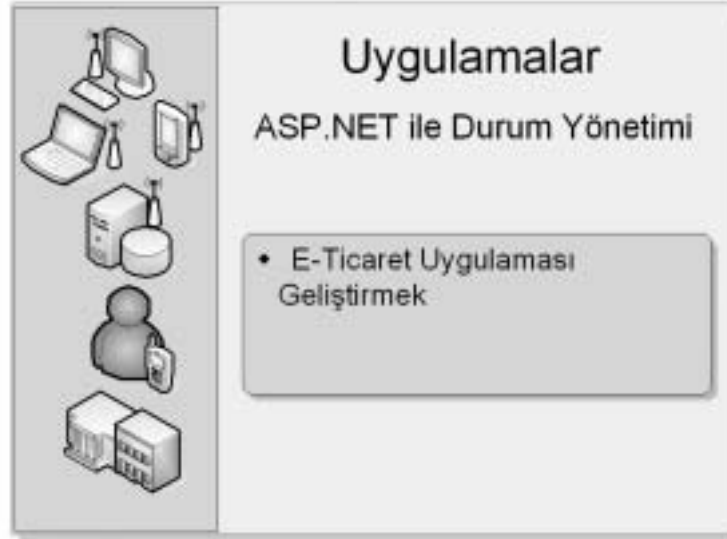
| Olay ismi                | Açıklama  |
|--------------------------|---|
| <b>Application_Start</b> | Uygulama ilk çalışmaya başladığında çalışır.                  |
| <b>Application_End</b>   | Uygulama sona erdiğinde çalışır.                              |
| <b>Session_Start</b>     | Yeni bir session oluştuğunda çalışır.                         |
| <b>Session_End</b>       | Session kapandığında çalışır.                                 |
| <b>Application_Error</b> | Uygulamanın çalışması sırasında bir hata oluştuğunda çalışır. |

## Modül Özeti



1. Session niin kullanılır?
2. Cookie niin kullanılır?
3. Cookie trleri nelerdir?
4. Application niin kullanılır?

## Lab 1: E-Ticaret Uygulaması Geliştirmek



Bu uygulamada **Session** nesnesi ile **KitapDetay** sayfasına erişim engellenecektir. **KitapDetay** sayfasına sadece sisteme giriş yapan kullanıcılar erişebilecektir.

Bu lab tamamlandıktan sonra:

- Session kullanımını öğreneceksiniz.

### Session Kullanmak

ASPEticaret isimli projeyi açın.

#### UyeGiris Formu İçinde Session Kullanmak

UyeGiris Web formunu açın.

UyeGiris Web formunun Code Behind kodları aşağıdaki gibi olacaktır. UyeGiris kod dosyası içindeki işaretli satırlar, veritabanı içinden çekilen kayıtların **Session** değişkenlere aktarılmasını sağlar.

```
Imports System.Data.OleDb
```

```
Private Sub btnGiris_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnGiris.Click
```

```
    ' Session("user") = "tamer"
    Dim connStr As String = _
```

```

        "Provider=Microsoft.Jet.OleDB.4.0;Data Source=" _
        & Server.MapPath("KitapDb.mdb")

Dim conn As New OleDbConnection
conn.ConnectionString = connStr

Dim comm As 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)
Dim sonuc As Boolean
Dim dr As OleDbDataReader
Try
    conn.Open()
    dr = comm.ExecuteReader

    If dr.HasRows = True Then
        sonuc = True
        If dr.Read = True Then
            Session("user") = dr.Item("Email")
            Session("ad") = dr.Item("Ad")
            Session("soyad") = dr.Item("Soyad")
            Session("musteriId") =
dr.Item("MusteriID")
        End If
    Else
        sonuc = False
    End If

    dr.Close()

Catch ex As Exception
    Response.Write(ex.Message)
Finally
    If conn.State = ConnectionState.Open Then
        conn.Close()
    End If
End Try
If sonuc = True Then
    Response.Redirect("Default.aspx")
Else

```

```

        lblMesaj.Text = "Hatalı kullanıcı adı veya
sifre"
    End If
End Sub

```

## KitapDetay Formu İçinde Session Kullanmak

**KitapDetay** Web formunu açın.

**KitapDetay** Web formunun Code Behind kodları aşağıdaki gibi olacaktır. **KitapDetay** kod dosyası içindeki işaretli satırlar, kullanıcının sisteme girişini kontrol eder. Eğer kullanıcı sisteme giriş yapmadıysa, **user** değişkeni içine değer aktarılmaz. Bu durum kullanıcının **Giris.aspx** sayfasına yönlendirilmesine sebep olur.

```
Imports System.Data.OleDb
```

```
Dim kID As String
```

```
Private Sub Page_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
```

```

    If Session("user") = "" Then
        Response.Redirect("Giris.aspx")
    End If

```

```

    kID = Request.Params("kID")
    'Response.Write(kID)
    Dim connStr As String = _
    "Provider=Microsoft.Jet.OleDB.4.0;Data Source=" _
    & Server.MapPath("KitapDb.mdb")

```

```

Dim conn As New OleDbConnection
conn.ConnectionString = connStr

```

```

Dim comm As New OleDbCommand
comm.CommandType = CommandType.Text
comm.CommandText = _
    "Select * from Kitap Where KitapID =@kitapID"
comm.Connection = conn

```

```

comm.Parameters.Add("@kitapID", _
Convert.ToInt32(kID))
Dim dr As OleDbDataReader
Try
    conn.Open()
    dr = comm.ExecuteReader
    If dr.Read = True Then

```

```

        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")
    End If
    dr.Close()

Catch ex As Exception
    Response.Write(ex.Message)
Finally
    If conn.State = ConnectionState.Open Then
        conn.Close()
    End If
End Try
End Sub

```

## Ust Kullanıcı Kontrolü İçinde Session Kullanmak

Ust kullanıcı kontrolünü açın.

Ust kullanıcı kontrolünün Code Behind kodları aşağıdaki gibi olacaktır. **Ust** kod dosyası içindeki işaretli satırlar, **Session** değişkeninin değerini **lblAd** isimli etikete yazar.

```

Private Sub Page_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
    If Session("user") <> "" Then
        lblAd.Text = "Bay / Bayan : " & Session("ad") &
" " & Session("soyad")
        btnCikis.Visible = True
    Else
        btnCikis.Visible = False
    End If
End Sub

```

**btnCikis** düğmesindeki işaretli kod satırları, tüm **Session** değişkenlerinin değerini sıfırlar.

```

Private Sub btnCikis_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
btnCikis.Click
    Session.Abandon()
    btnCikis.Visible = False
    Response.Redirect("Default.aspx")
End Sub

```

## Okuyucu Notları

## Okuyucu Notları