## İÇİNDEKİLER

# ASP'YE GİRİŞ

ASP, Server-Side Tabanlı bir dildir. Yani Sunucu tarafında yorumlanır. Ve bu sayede kimse kaynak kodlarınıza ulaşamaz...

Süreç şu şekilde işler; Siz URL hanesine adresi girdiğinizde, bana falanca dosyayı bul, çağır ve yorumla diye bir istemde bulunursunuz. (Bu durumda siz Client-Side yani istemci oluyorsunuz.) Server'da (Sunucu) kendisinden icra etmesini istediğiniz dosyayı arar ve eğer bulursa, bu dosyayı hemen "asp.dll" adlı bir programa iletir. asp.dll'de aldığı bu dosyayı hemen yorumlamaz. Önce serverda bulunan, belki sizin de gözünüze çarpmıştır, "Global.asa" adlı dosyanın çalışıp çalışmadığına bakar.

Global.asa'da diğer asp dosyaları gibi aslında düz bir text dosyasıdır ama farkı şudur. ASP dosyalarının çalışma kurallarını belirler.

Asp.dll önce gelen dosyada hangi script dilinin kullanıldığına bakar. Ve buna göre kendini hazırlar. Nereden mi anlar. Belki dikkatinizi çekmiştir. ASP veya Java dosyalarında. ScriptLanguage diye bir satır. İşte buradan anlar. Asp.dll sonra bu derlediği bilgileri, tamamen asp kodlarından ayrılmış, temiz bir halde browser'a gönderir. Bizde böylece sadece HTML kodlarını görürüz. ASP yazmak için iki dil kullanabiliriz.

Visual Basic ve Java Script... Fakat şu anda dünyada en geçerli olanı Visual Basic'tir. İşin bir ilginç yanı da şudur. Normalde Netscape ASP Scriptini yorumlayamaz. Ama bizim kodlarımızda böyle bir korkumuz yoktur. Çünkü niye. Hatırlayın bakalım niye? Umarım çoğunuz püf noktasını anlamıştır. Çünkü "asp.dll" Browser'a VB kodlarından tamamen arındırılmış, yorumlanmış, tertemiz bir HTML sayfası gönderir. Ve böylece bizim de acaba netscape'i olanda çalışacak mı diye bir korkumuz kalmaz.

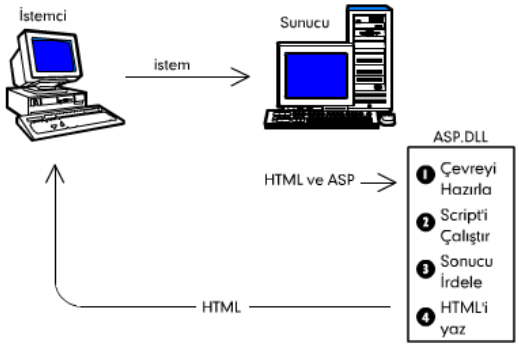
## *STATİK SAYFA NEDİR?*

Statik sayfa onu oluşturan kişinin ( webmaster ) bu sayfanın içeriğini bitirdiği tamamladığı ve kullanıcıların bu sayfaya her uğradıklarında aynı içeriği gördüğü sayfalardır. İçeriklerinin değişmesi için yeniden tasarlanmaları gereklidir. Bu zaman ve iş gücü kaybına yol açan bir etkendir.

## *DİNAMİK SAYFA NEDİR?*

İçeriği tamamı ile önceden belirlenmiş bazı kriterlere bağlı olarak değişen sayfaları dinamik sayfalar olarak adlandırabiliriz. Bu değişiklik sayfanın aldığı bazı girdilere göre olabilir.

## *ASP SAYFALARININ ÇALIŞMA İLKESİ*




Şimdi, ilk ASP'mizi çalıştırdığımıza göre, biraz teknikten söz edebiliriz. HTML'in ziyaretçinin bilgisayarında çalıştığını biliyorsunuz; istemci Browser, sizin URL hanesine adını yazdığınız HTML dosyasını yine adresteki sunucu Web Server'dan ister. Web Server da bu dosyayı bulur ve içinde kayıtlı resim ve diğer unsurlarla birlikte istek sahibine gönderir. Fakat kimi zaman Server'a bize sadece bir dosyayı göndermesini değil, fakat bu dosyanın içinde kayıtlı komutlar varsa onları icra etmesini de bildirebiliriz. Bunu yapmanın bir yolu CGI programlarıdır. Bir diğer yolu ise ASP'dir. Web Server, kendisinden bir ASP belgesi talep edildiğinde, kendi kendine "Dur bakalım! ASP istendiği zaman hemen alıp göndermek yok, önce bunu ASP.DLL programına gönderelim.. Ondan sonra işimize devam ederiz!" der.

ASP.DLL, kendisine bir .asp dosyasının geldiğini görünce, hemen ASP'lerin Anayasası olan **global.asa**'nin çalışıp çalışmadığına bakar. **global.asa**, tıpkı diğer ASP dosyaları gibi bir düz yazı dosyasıdır ve ASP programlarının çalışma koşullarını düzenleyen kuralları içerir. (Bu dosyayı sırası gelince ele alacağız.) Yukarıdaki örnekte gördüğümüz gibi ASP dosyası hem HTML kodları içerir, hem de içinde bir Script diliyle yazılmış kodlar vardır. ASP'ye "program" özelliği kazandıran bu Script dili ile yazılmış kodlardır. ASP.DLL, önce gelen .asp dosyasında hangi Script dilinin kullanıldığına bakar ve bunun için gerekli ortamı oluşturur; yani bu Script dilini yorumlayacak programı çalıştırır; bu program Script'i yorumlar ve icra edilecek komutları icra eder; ASP.DLL, icra edilen komutlar, işletim sisteminin yardımını istiyorsa (örneğin bir veritabanından veri çekmek gibi, veya dosya sistemine bir dosya açtırmak, yazdırmak, sildirmek gibi) bu yardımın edinilmesini de sağlar. Bütün bu işlerin sonunda sizin yazdığınız HTML kodlarına ek yapmak (örneğin bir tablonun içini, çekilen verilerle doldurmak veya dosya sisteminden edinilen bir dosyanın içeriğini sayfaya aktarmak gibi) gerekiyorsa bu ekleri ASP.DLL yapar.

ASP.DLL, kendisine sevk edilen dosyayı satır satır okur ve önce dosyadaki ASP kodlarının gerektirdiği HTML değerlerini bulur; yani önce ASP icra edilir, gereği yerine getirilir. Sonra HTML bölümleri ile birleştirilip sonuçta sunucuya saf ve temiz bir HTML sayfası gönderilir. Bu sayfanın içinde bir satır bile ASP kodu bulunmaz. Eğer sayfanıza ziyaretçinin Browser'ında çalışması amacıyla Javascript veya VBScript kodları koydu iseniz, elbette bu kodlar HTML'in parçası olarak ziyaretçiye gidecektir. Fakat giden sayfada artık ASP'ye ilişkin hiç bir şey kalmamış olacaktır.

## ASP'NİN DİLİ

ASP, bir teknolojidir. Kendi başına bir yazım kuralı yoktur. ASP tekniğini kullanabilmek için, ASP sayfasının talep edilmesi halinde ziyaretçiye gönderilmeden önce ASP.DLL'ye teslim edilmesi bu teknolojinin kullanılabilmesi için hemen hemen tek şarttır. Bunu, dosya uzantısını .asp yaparak sağlarız.

ASP.DLL ise, dünyada mevcut bütün Script dilleri ile verilecek komutları kabul edebilir. Sadece ASP.DLL'e sayfadaki kodların hangi dilde olduğunu söylemeniz gerekir. Bunu, ASP sayfasının birinci satırında yaparız. Örneğin ASP'ye VBScript dilini kullanmasını belirtmek için bu satırı şöyle yazarız:

**<% @Language=VBScript %>**

ASP sayfalarında genellikle VBScript, JavaScript ve JScript kullanılır. Ancak örneğin Perl dilinden türetilen PerlScript, PHP'den türetilen PHPScript de giderek ilgi çeken ASP dilleri arasına giriyor.

Bir ASP sayfası içinde farklı Script dilleri kullanılabilir.

Biz bu kitapçıkta örneklerimizi VBScript diliyle yazacağız.

## YAZILIM KURALLARI

VBScript komutları, anahtar kelimeleri ve değişken adlarının büyük harf-küçük harf olması önemli değildir. Yani yukarıdaki ifadelerden birini şu biçimlerden birinde yazabilirdik; kodumuz yine de çalışırdı:

**For fontBoyut = 1 To 7**

**FOR FONTBOYUT = 1 TO 7**

**for fontboyut = 1 to 7**

Fakat... Bu demek değildir ki, VBScript ile kodlamanın kendi gelenekleri yok! VBScript'çiler genellikle komutların birinci harfini büyük yaparlar: For gibi. Değişken adlarında ise kendinize anlamlı gelen bir biçim tutturabilir ve öyle devam edebilirsiniz.

Eğer bir kod satırı çok uzun geliyor ve daha sonra anlaşılması imkansız derecede uzuyorsa, bu satırı alt çizgi (_) ile aşağı satırda devam ettirebilirsiniz. Örnek:

```
<%
If degisken1 > 1 And _
   değisken1 < 10 Then
%>
```

## DEĞİŞKENLER

İyi programcılık değişkenlerin önceden beyan edilmesini gerektirir. Bunu DIM komutuyla yaparız. DIM, Dimension (boyutlandır) kelimesinden kısaltılmıştır. Pek akıllıca görünmese de bu komut, bilgisayarın değişken yeri olarak bir bellek alanının boyutunu belirtmesini sağlar.

Değişkenlere verilecek ismimlerin anlaşılır olması programın okunulurluğunu kolaylaştırır. Ayrıca değişken tanımlamaları ile ilgili bazı kurallar mevcuttur. Bu kurallar aşağıda verilmiştir:

**1-** Değişken ismi bir harf ile başlamalıdır. Ad1, Ad2 şeklinde değişken tanımı yapılabilir fakat 1Ad veya 2Ad kullanımı yanlıştır.

**2-**Değişken isminde boşluk bulunamaz. Bunun yerine alt çizgi karakteri kullanılabilir. Adi_Soyadi doğru bir kullanım fakat Adi Soyadi gibi arasında boşluk içeren değişken tanımı yapılamaz.

**3-**Değişkene verilecek isim Vbscript komutlarını içeremez Dim bir değişken ismi olamaz.

**4-**Değişken ismi 255 karakterden fazlada olamaz.

```
<%
DIM Gun, Ay, Ogrenci, Not
Gun = "Pazartesi"
Ay = "Ocak"
Ogrenci = "Necip"
Not = 5
%>
```

Bir ASP sayfasının birinci satırı olarak

**<% OPTION EXPLICIT %>**

yazarsanız VBScript DIM komutuyla belirlenmemiş değişken kullanmanıza izin vermez; kullanırsanız hata verir ve durur. Bu ifadenin işinize çok yarayacağı bir diğer nokta, kodda değişken adını yazarken hata yapmanızı önlemektir. VBScript sizin hata yaptığınızı bilemeyeceği için yanlış kelimeyi yeni bir değişken sayacaktır. Değer atamadığınız bu yeni değişkeni kullandığınız yerlerde ya programınız hata verir, ya da kendisine doğru görünen işlemler yapar, fakat beklediğiniz sonucu elde edemezsiniz. **OPTION EXPLICIT**, yanlış yazdığınız değişkeni yeni değişken sayarak önceden tanımlanmamış değişken kullandığınızı düşünerek, duracaktır.

### Dizi Değişkenleri

VBScript'in kullanılmaya hazır bir çok fonksiyonu vardır; bunlardan biri olan Array ile, kolayca dizi değişken oluşturabiliriz. Diyelim ki, Gunler(7) dizi-değişkenini gün adları ile doldurarak oluşturmak istiyoruz:

```
<%
Dim Gunler = Array ("Pazartesi" , "Salı" , "Çarşamba" ,
"Perşembe" , "Cuma" , "Cumartesi", "Pazar")
%>
```

ile hem dizi-değişkeni oluşturabiliriz; hem de değerlerini atayabiliriz. Bu suretle oluşturduğumuz dizi değişkenin üyelerine daha sonra sıra numaraları ile atıfta bulunabilirsiniz. Örneğin:

```
<%=Gunler(6)%>
```

bize Pazar'ı verir. Neden? Çünkü hayatlarının büyük bölümünü penceresiz ortamlarda geçiren kişiler olan dil tasarımcıları, sayı saymaya biz normal insanlar gibi 1'den değil 0'dan başlarlar; dolayısıyla Gunler dizi-değişkeni Gunler(0)'dan başlar!

**REDIM :**

Bazen elinizde eleman sayısı değişen bir data girdisi olabilir bu gibi durumlarda kullanılacak dizinin boyutuda değişken olabilir. Programınızın herhangi bir yerinde kullanılan bir diziyi yeniden boyutlandırma ihtiyacı duyulursa Redim keyword ü kullanılabilir böylece programımızın çalışması esnasında dizimizi yeniden boyutlandırabiliriz. Ancak unutulmaması gereken bir nokta bir dizi yeniden boyutlandırıldığında içerdiği eski veriler dizi içerisinden atılıır. Bir dizi içerisinde bulunan eski veriler kullanılmak isteniyorsa prereserve keywordü kullanılabilir.

**Çok Boyutlu Diziler**

Bazı uygulamalarda matris yapısında dizi tanımlamalarına ihtiyaç duyulabilir. Böyle durumlarda çok boyutlu dizi tanımları kullanılabilir. Çok boyutlu dizi tanımı kullanılacak dizinin boyutları verilmek süreti ile gerçekleştirilebilir.

**Dim myarray(3,3)**

şeklinde bir tanımlama 3 X 3 boyutlarında bir dizi oluşturur bu diziye atama yapılmak istenildiğinde ise:

```
myarary(0,0)="test"
myarary(0,1)="deneme"
myarary(0,2)="test2"
myarary(1,0)="deneme2"
myarary(1,1)="111-23-4"
myarary(1,2)="Mustafa"
```

şeklinde atama yapılabilir

## Sabit Değerler

VBScript'te bir kere aldığı değeri sürekli saklayan unsurlar vardır. Sabit değer, bütün ASP işlemleri boyunca (hatta isterseniz, bütün site, yani Uygulama boyunca) değişmeden kalır. Bu değerleri **Const** (**constant**, sabit kelimesinden türetilme) komutuyla belirtiriz:

```
Const DolarDeger = 560780
Const SirketinAdi = "Web Tasarım ve Site Onarım A.Ş."
Const Slogan = "Siteler ve Kırık Kalpler Onarılır"
```

## Veri Tipleri

### 1- Byte

1 Baytlık işaretsiz tamsayı tipidir. 0 ile 255 arasında değer alabilir.

### 2-Integer

2 Baytlık işaretli tamsayı tipidir. -32.768 ile 32.767 arasında değer alabilir.

### 3-Long

4 Baytlık işaretli tamsayı tipidir. -2.147.483.648 ile 2.147.483.647 arasında değer alabilir.

### 4-Long

4 Baytlık ondalık sayı tipidir. +- 3.402923 X 10 38 ile +- 1.401298 X 10-45 arasında değer alabilir.

### 5-Double

8 Byte lık ondalık sayı tipidir. +- 1.79769313486232 X 10308 ile +- 4.94065645841247 X 10-324 arasında değer alabilir.

### 6-String

Karakter sınırı verilmezse 2 milyar karaktere kadar atama yapılabilen sayısal olmayan veri tipidir. Bu tip karakter sayısı +10 byte yer kaplar. String bir veri tipi oluşturmak için değişkene atanacak değer " " işaretleri içerisinde yer almalıdır.

```
Adi="Mehmet"
```

### 7-Date

8 byte yer kaplayan bu değişkene 1/1/100 ile 31/12/9999 arasındaki tarih ve 0:00:00 ile 23:59:59 arasındaki saat atmaları yapılabilir. Bu tipteki değişkenlere atama string veri tipinde olduğu gibi veya # # karakterleri arasında yapılabilir.

```
Dogum_tarihi=#19/08/1978#
```

### 8-Boolean

2 byte lık bir veri tipi olmasına rağmen sadece True veya False değerleri alabilir. Yani daha çok iki durumlu değişkenlerde kullanılır. Bu tipten tanımlanan değişkenlere direkt True veya False atanabileceği gibi sayısal değerlerde atanabilir.. Atanan sayı 0 ise False, değilse True kabul edilir.

```
Evli=True
Evli=1
```

## Veri Tipi Değiştirme İşlemleri

Bazen uygulama esnasında bir veri tipini başka bir veri tipine değiştirme ihtiyacı duyulur. Bunun için kullanılabilecek veri dönüşüm fonksiyonları aşağıda verilmiştir.

**Ccur (ifade) :**Parantez içinde verilen ifadeyi Currency veri tipine dönüştürür**.**

**Cdbl (ifade) :**Parantez içinde verilen ifadeyi Double veri tipine dönüştürür.

**Cint (ifade) :**Parantez içinde verilen ifadeyi Integer veri tipine dönüştürür.

**CIng (ifade) :**Parantez içinde verilen ifadeyi Long veri tipine dönüştürür.

**CVar (ifade) :**Parantez içinde verilen ifadeyi Variant veri tipine dönüştürür.

**CBool (ifade) :**Parantez içinde verilen ifadeyi Booolean veri tipine dönüştürür.

**CByte (ifade) :**Parantez içinde verilen ifadeyi Byte veri tipine dönüştürür.

**CDate (ifade) :**Parantez içinde verilen ifadeyi Date veri tipine dönüştürür.

### Cınt

Herhangi tipteki geçerli bir ifadeyi integer (tam sayı) alt tipine dönüştürür.

Söz dizimi: CInt(ifade)

```
mystr = "12"
myint = CInt(mystr)/3 'myint değişkeninin değeri : 4
```

### CStr

Herhangi tipteki geçerli bir ifadeyi string (metin) alt tipine dönüştürür.

Söz dizimi: CStr(ifade)

```
myint = 41
mystr = CStr(myint) + " kere maşallah" 'mystr değeri : "41 kere maşallah"
```

### Cdate

Herhangi tipteki geçerli bir ifadeyi date (tarih) alt tipine dönüştürür.

Söz dizimi: CDate(ifade)

```
mystr = "Ekim 12, 1998" 'sunucu sistemin saat/tarihi İngilizce ise değişir.
mydate = CStr(mystr) 'mydate değeri : 12.10.1998
```

### VarType

```
<%
degisken_1 = 12
degisken_2 = "Ben sendeyim sen bendesin…"
degisken_3 = #08-12-2000#
response.write vartype(degisken_1) 'sayfaya 2 yazar
response.write vartype(degisken_2) 'sayfaya 8 yazar
response.write vartype(degisken_3) 'sayfaya 7 yazar
%>
```

### TypeName

```
Pi=3
Tarih=#10/10/1998#
Test="Selam Millet"
Response.write TypeName (pi) 'Sayfaya byte yazar
Response.write TypeName (tarih) 'Sayfaya date yazar
Response.write TypeName (text) 'Sayfaya text yazar
```

## Test Fonksiyonları

VBScript'te kullandığımız bazı değişkenlerin o andaki durumu, programımızın akışını kontrolde kullanacağımız bilgiyi sağlayabilir. Sözgelimi bir değişkenin değeri boş ise, ziyaretçimizin formu tam olarak doldurmadığını düşünebiliriz. VBScript, bize değişkenlerin durumunu sınamamız için bazı özel fonksiyonlar sağlar. Bu özel fonksiyonlardan dönen değer **True** (doğru) veya **False** (yanlış) olur; doğru sonucun değeri 1, yanlış sonucun değeri ise 0'dır:

| | |
|---|---|
| **IsArray** | Bir değişkenin dizi-değişken (Array) olup olmadığını sınar. |
| **IsDate** | Bir değişkenin değerinin tarihe (Date) çevrilip çevrilemeyeceğini sınar. |
| **IsEmpty** | Bir değişkenin tanımlanıp değer atanmış olup olmadığını sınar. |
| **IsNull** | Bir değişkenin geçerli bir değer tutup tutmadığını sınar. |
| **IsNumeric** | Bir değişkenin sayı olarak işleme tabi tutup tutulamayacağını sınar |
| **IsObject** | Bir ifadenin geçerli bir ActiveX veya OLE nesnesine referansta bulunup bulunmadığını sınar. |

**TypeName** Bir değişkenin türünü belirtir.

**VarType** Bir değişkenin türünü belirten sayıyı verir.

## OPERATÖRLER:

Bir programlama dilinde veya scriptini kullanarak artimetik ve lojik işlemleri yapmak için gerekli operatörler bulunur. Vbscripte de bazı işlemler operatörlerle yapılırken bazıları ise fonksiyonlarla yapılır.

| Operatör | Anlamı | Kullanılışı | Sonuç |
|---|---|---|---|
| = | Atama | X = 5 | |
| + | Toplama Print | 20+5 | 25 |
| - | Çıkarma Print | 10-5 | 5 |
| * | Çarpma Print | 3*5 | 15 |
| / | Bölme Print | 9/2 | 4.5 |
| \ | Tam Bölme Print | 9\2 | 4 |
| & | String Toplama Print | "Bayram" & "paşa" | Bayrampaşa |
| ^ | Üs | 4^2 | 16 |
| Mod | Bölmede Kalan | 5 Mod 2 | 1 |
| And | Ve | 5 And 17 | 21 |
| Or | Veya | 7 Or 17 | 23 |
| Not | Değil | Not &H1 | &HFFFE |

### Tesadüfî Sayı

Bilgisayarın matematik işlemlerde, özellikle istatistik hesaplamalarla kullanılması tesadüfî (rastlantısal) sayı üretmeyi gerekli kılmıştır. Fakat daha sonra bilgisayar oyunları bu işlemi adeta zorunla hale getirdi. Rastlantısal sayı, bir dizide tekrar etmesi belirli bir düzene tabi olmayan sayı demektir. Bilgisayar yokken, tesadüfî sayı tabloları matematikçiler tarafından uzun uğraşlarla üretilirdi.

VBScript, bu amaçla Visual Basic'in **Randomize** ve **Rnd** komutlarını kullanır. **Randomize**, tesadüfî sayı üretme sürecini başlatır; **Rnd** da bu sayıyı size verir. Kodunuzda bir yerde **Rnd** kullanacaksınız, ondan önce bir yerlerde mutlaka **Randomize** komutunun yer alması gerekir. Bunun bir uygulaması şu olabilir:

```
<% OPTION EXPLICIT %>
<HTML>
<%
Dim TesadufiSayi
Randomize
TesadufiSayi = Rnd
%>
<%=TesadufiSayi%>
</HTML>
```

Tam sayı elde etmek için, Int ve Round komutlarını kullanın.

**Tamsayi = Int(KesirliSayi)**

## INCLUDE FILE

Bu seçeneğimiz asıl olarak, herhangi bir ASP dosyamızı başka bir ASP dosyamıza dahil etmeye yarar.

```
<!--#INCLUDE FILE="../../Menuler/SAGTARAF.asp"-->
<!--#INCLUDE FILE="../../Menuler/SOLTARAF.asp"-->
```

## STRING İŞLEMLERİ

### UCASE ve LCASE

Ucase bir string (metin) içerisinde yer alan tüm karakterleri büyük karakterlere çevirmek için kullanılan bir fonksiyondur. Bu işlemin tam tersini yani bir string (metin) içinde yer alan karakterlerin tamamını küçük harflere çevirmeye yarayan fonksiyon ise Lcase fonksiyonudur.

**yazi="Selam Millet"**

**yazibuyuk=Ucase(yazi)** *'yazibuyuk değişkeninin değeri SELAM MILLET*

**yazikucuk=Lcase(yazi)** *'yazikucuk değişkenini değeri selam millet*

```
<%
yazi="Selam Millet"
response.write Ucase(yazi)
response.write Lcase(yazi)
%>
```

### LEN

Bir stringin içeirsinde bulunan karakter sayısının bulunmasına ihtiyaç duyulduğunda Len fonksiyonu kullanılır. Kullanım şekli: Len (string) şeklindedir.

```
<%
text="bunun içinde kaç karakter var"
sonuc=Len(text)
response.write sonuc
%>
```

Yukarıda ki örneği açıklayacak olursak text değişkenine içeriği "bunun içinde kaç karakter var" şeklinde bir değer atanıyor daha sonra len fonksiyonu kullanılarak bulunan değer yani stringin içerisnde yer alan karakter sayısı ki buna boşluklar (space ) de dahil sonuc adlı bir başka değişkene atanıyor en son satırda ise sonuc değişkenin içeriği ekrana yazılıyor.

**LEFT ve RIGHT**

Left ve Right fonksiyonları bir string içerisinden sağ veya soldan istenilen karakterin kopyalanmasını sağlamak amacı ile kullanılır. Bu string içerisinden seçilmiş bir parçanın kopyalanması gibidir. Aşağıda Left ve Right kullanımlarına ilişkin örnekler verilmiştir.

```
<%
test="Buradanekadarçokinsanvarmış"
soldan=Left(test,6)
sagdan=Right(test,6)
response.wrirte soldan
response.wrirte sagdan
%>
```

ilk satırda test adlı değişkene test="Buradanekadarçokinsanvarmış" satırı ile bir atama gerçekleştiriliyor, ikinci satırda kullanılan left(test,6) komutu ile test stringi içerisinde soldan başlanarak 6 adet karakter kopyalanıyor ve kopyalanan karakter soldan adlı değişkene atanıyor. Bu işlemin sonucunda soldan değişkeninin değeri soldan="burada" oluyor çünkü tüm yazı içerisinde soldan itibaren 6 karakter kopyalayacağımız left(test,6) ile belirtmiş oluyoruz. Aynı işelm right fonksiyonu içinde geçerki burada test verisi içinden gene 6 karakter kopyalıyoruz fakat bu işlem string içerisinde sağdan sola doğru yapılan bir haraket ile gerçekleşiyor. Bu işelmin sonucundada elde edilen değer sagdan değişkenine atanıyor. Sonuç itibarı ile sagdan değişkeninin içeriğide sagdan="varmış" oluyor. Son 2 satırda ise elde edilen bu değerler ekrana yazılıyor. (response.write kullanılarak)

**MID**

Bir string içerisinde belirli bir aralığı kopyalamak istediğimizde kullanabileceğimiz fonksiyon Mid fonksiyonudur. Genel kullanımı:

Mid(String , nereden_baslanacak , kaç_karakter_kopyalanacak) şeklindedir.

```
test="Buradanekadarçokinsanvarmış"
sonuc=Mid(test,7,10)
response.write sonuc
```

test içersinde yer alan test="Buradanekadarçokinsanvarmış" text içerisinde yedinci karakterden başlamak üzere toplam 10 karakter kopyalanarak sonuc adlı değişkene atanıyor sonuc değişkenin değeri ise "nekadarçok" şeklinde oluyor ve response.write kullanılarak sonuc ekrana (html data stream içerisine ekleniyor) yazdırılıyor.

## INSTR

Instr fonksiyonu kullanılarak bir string dizisi içerisinde yer alan bir karakterin nerede yer aldığını (kaçıncı karakter olduğunu) bulmak mümkündür.

```
strtext="HowLong"
karakter_nerede=Instr(strtext,"Long")
```

strtext değişkenine "HowLong" değeri atanmış ve instr fonksiyonu kullanılarak bu text içerisinde yer aldı düşünülen "Long" un nerede başladığı bilgisi karakter_nerede adlı değişkene atanmıştır. Burada karakter_nerede değişkeninin değer 4 olacaktır çünkü "Long" strtext içerisinde 4. cü karakterden itibaren başlamaktadır. Eğer aranılan text veya karakter bulunamaz ise Instr fonksiyoınu 0 değerini geri döndürür.

## LTRIM, TRIM ve TRIM

Bu üç fonksiyon verilen bir textin sağında, solunda veya hem sağında hem de solunda kalan boşlukların atılmasını sağlamak için kullanılan fonksiyonlardır. Her üç fonksiyonda sadece tek bir parametre alırlar ki buda boşlukların kaldırılmak istenildiği text in tam kendisidir.

Ltrim textin solunda bulunan tüm boşlukları, Rtrim textin sağında bulunan boşlukları Trim ise textin hem sağında hemde solunda bulunan boşlukları çıkarmak için kullanılan bir fonksiyondur.

## REPLACE

Uzun bir metin içerisindeki belirlediğiniz ifadeyi başka ifadeyle değiştirmenizi sağlar.

REPLACE(string, değişecek_olan, yerine_konulacak, başlangıç_indexi, değişiklik_sayısı, karşılaştırmakodu)

**String**: İçinde değişiklik yapacağımız metindir.

**Değişecek_olan**: Metin içinde değiştirmek istediğimiz alt-dizgi (sub-string) dir.

**Yerine_konulacak**: Belirtilen alt-dizginin yerine konulacak ifadedir.

**Başlangıç_indexi**: Değiştirme işleminin, stringin kaçıncı karakterinden başlayacağıdır.

Varsayılan: 1

**Değişiklik_sayısı**: Çok açık. -1 yazılırsa bulunan tüm alt-dizgiler değiştirilir,

**karşılaştırmakodu**: VBTextCompare için 1, VBBinaryCompare için 0 yazılabilir.

Varsayılan 0.

Replace fonksiyonu şu şekilde de işlev yapar. REPLACE(string, değişecek, yerinekonulacak). Ancak fonksiyonun daha verimli kullanılabilmesi için tüm parametlerinin belirtilmesi gerekir.

```
Dim mystr
mystr = "Ağır ağır çıkacaksın bu merdivenlerden"
Response.write replace(mystr, "Ağır", "Hızlı", 1, -1, 0)
' sayfaya "Hızlı ağır çıkacaksın bu merdivenlerden" yazar.
Response.write replace(mystr, "Ağır", "Hızlı", 1, -1, 1)
' sayfaya "Hızlı Hızlı çıkacaksın bu merdivenlerden" yazar.
Response.write replace(mystr, "a", "A", 1, -1, 1)
' sayfaya "Ağır Ağır çıkAcAksın bu merdivenlerden" yazar.
Response.write replace(mystr, "a", "A", 1, 2, 1)
' sayfaya "Ağır Ağır çıkacaksın bu merdivenlerden" yazar.
```

Replace fonksiyonu bir çok yerde yardımımıza yetişir. Örneğin bir formdan alınan içerikte yer alması muhtelemel istenmeyen ifadelerin ayıklanması sağlanabilir. HTML taglerinin kullanılması önlenebilir. Ve daha bir çok yerde REPLACE size büyük fayda sağlayacaktır.

## STRING Fonksiyonu

VBScript ten yine evlere şenlik bir fonksiyon. Belirtilen sayıda karakterin tekrarını içeren string oluşturuyor.

```
Response.write STRING(5, "*") ' sayfaya "*****" yazar
```

## SPLIT Fonksiyonu

Bir metinden belirlenmiş ayraçlar (delimiter) vasıtasıyla belirtilen sayıda alt-dizgi içeren 0 tabanlı, tek boyutlu dizi üretir.

**SPLIT(ifade, ayraç, sayi, karşılaştırmakodu).**

Sayi yerine -1 yazılırsa tüm belirlenen tüm alt-dizgiler işleme tabi tutulur.

```
Dim mystr, dizi(3)
mystr = "EvcilASP|Türkiye nin|ASP Bostanı"
dizi = SPLIT(mystr, "|", -1, 1)
Reponse.Write dizi(0) ' sayfaya "EvcilASP" yazar
Reponse.Write dizi(1) ' sayfaya "Türkiye nin" yazar
Reponse.Write dizi(2) ' sayfaya "ASP Bostanı" yazar
```

## JOIN Fonksiyonu

Split fonksiyonuna göre zır işlemi yapar. Bir dizinin elemanlarını belirtilen ayraç ile birleştirip bir string üretir.

**JOIN(ifade, ayraç)**

```
Dim dizi(3), str, str2
dizi(0) = "Ben"
dizi(1) = "Sekizinci"
dizi(2) = "Henri"
str = JOIN(mystr, "|")
str2 = JOIN(mystr, " ")
Reponse.Write str ' sayfaya "Ben|Sekizinci|Henri" yazar
Reponse.Write str2 ' sayfaya "Ben Sekizinci Henri" yazar
```

## MANTIKSAL İFADELER

### If - Then

VBScript'in vereceğiniz bir durumun bulunup bulunmadığını sınamasını sağlar.

**If şart Then**

**[şart doğru ise yapılacak işler]**

**Else**

**[şart doğru değilse yapılacak işler]**

**End If**

If...then yapısının en son ve geniş kullanımı ise Elseif yapısı ile birden fazla şart içeren durumlardır.

```
<%
if method= "Faks" then
Response.Write "Lütfen Faks numaranızı giriniz "
Elseif method= "Email" then
Response.Write "Lütfen E-mail adresinizi giriniz"
Elseif method= "Telefon" then
Response.Write "Lütfen telefon numaranızı giriniz"
Else
Response.Write "Herhangi bir bilgi iletilmeyecektir"
End if
%>
```

### Select Case

Olasılık sayısı artan daha karmaşık bir yapıda if...then yapısı karmaşık bir çözüm olacaktır. Böyle durumlarda Select...Case yapısı daha uygun bir çözüm olarak kullanılır .

```
<%
Select Case Secim
Case "Faks"
Response.Write "Lütfen Faks numaranızı giriniz"
Case "Telefon"
Response.Write "Lütfen telefon numaranızı giriniz"
Case "E-mail"
Response.Write "Lütfen e-mail adresinizi giriniz"
End Select
%>
```

## DÖNGÜLER

### For..Next Döngüsü

Programın bir işi belirli kere yapmasını istiyorsak, ona yapacağı işi bir sayaç değişkeniyle birlikte, **For** döngüsüyle bildiririz:

**For sayaç = başlangıç To son Step adım**

**...yapılacak işler...**

**Next**

### While...Wend

Ne var ki, program mantığı bazen bize for…next gibi açık ve seçik bir sayaç kurma imkanı vermez. Sayaç olarak kullanacağımız değer, programın başka bir bölümü tarafından üretiliyor olabilir. Veya bu değer ziyaretçi tarafından belirlenmiş olabilir. Özetle yapılmasını arzu ettiğimiz işin ancak sayaç bir değerden azsa, çoksa veya eşitse yapılmasını, bu durum değişirse durmasını isteyebiliriz. Bunu **While** (..iken) komutuyla yapabiliriz. **While** döngüsünü kullandığımız zaman sayacı bizim arttırmamız gerekir. Sözgelimi, yukarıdaki programın 7 günün tümünü ekrana yazmasını değil de, mesela gün sayısı 5'den küçük ise yazmasını istiyor olabiliriz.

```
While sayac <= 5
        Response.Write Gunler(sayac)
        Response.Write "<BR>"
sayac = sayac + 1
wend
```

### Do..Loop

**Do** (Yap) komutu ile kuracağımız döngüler iki ayrı türde olabilir: bu döngü ile bir dizi komutu, bir koşul doğru iken veya doğru oluncaya kadar yaptırabiliriz. Bu yöntemlerden her biri iki ayrı şekilde yazılabilir. Bir koşul doğru iken bazı işlerin yapılmasını istiyorsak, **Do While** yöntemini kullanırız:

**Do While koşul**

**koşul doğru iken yapılacak işler**

**Loop**

Bu ifade ile VBScript koşul doğru olduğu sürece istediğimiz işi yapacaktır. Buradaki **Loop** kelimesi, döngünün başa dönmesini sağlar. Bu yöntemden şu şekilde de yararlanabiliriz:

**Do**

**koşul doğru iken yapılacak işler**

**Loop While koşul**

Burada, **Loop** komutu şartın hâlâ doğru olup olmadığını sınar ve doğru ise verilen işleri yapar; artık değilse bir sonraki satıra geçer. Döngünün bir şart gerçekleşinceye kadar bir işi yapmasını ise **Do Until** yöntemiyle sağlarız. Bu durumda döngü şöyle yazılır:

**Do Until koşul**

**koşul gerçekleşinceye kadar yapılacak işler**

**Loop**

Bu ifade ile VBScript koşul doğru oluncaya kadar istediğimiz işi yapacaktır. Buradaki Loop kelimesi, döngünün başa dönmesini sağlar. Bu yöntemden şu şekilde de yararlanabiliriz:

**Do**

**koşul gerçekleşinceye kadar yapılacak işler**

**Loop Until koşul**

## For Each... Next

For...Next kullanımının özel bir türüdür. Bir dizi veya koleksiyon içerisinde kullanılır. For döngüsü dizi veya koleksiyon içerisinde kalan eleman sayısı kadar gerçekleştirilir.

```
<%
Dim eleman
Dim sehirler(2)
sehirler(0)="Ankara"
sehirler(1)="İstanbul"
sehirler(2)="Ordu"
For Each eleman In sehirler
Response.Write eleman & "<BR>"
Next
%>
```

Döngü şehirler dizisi içerisinde yer alan her eleman (Şehir için, her dizi elemanı) için tekrarlanır. Eleman (şehir) sayısı 3 olduğuna göre döngü 3 defa tekrarlanacak ve ekrana dizinin içerisinde yer alan elemanlar yazılacaktır

### Bir Döngüyü Durdurmak

Bir döngüden belirlediğiniz koşul gerçekleşsin-gerçekleşmesin çıkmanız gerekebilir. Bunu bir başka değişkendeki değişiklik zorunlu kılabilir. Bir döngüden çıkmak için **Exit** (çık) ifadesini kullanabilirsiniz. Bu ifade, döngünün yaptığı işler arasında, genellikle bir **If** deyimi ile birlikte yer alır.

**For sayac = 1 to 10**

**[..bir takım işler yap..]**

**If Degisken1 > Degisken2 Then Exit For**

**[..bir takım işlere devam et..]**

**Next**

## SÜREÇLER (PROSEDÜRLER)

VBScript'te programın akış kontrolünde kullanacağınız bir diğer grup araç ise örneğin Javascript veya Perl'de fonksiyon dediğimiz gruplandırılmış ve isimlendirilmiş işlem kümeleridir. Bu kümeler programın bir yerinde topluca dururlar ve programın başka bir yerinden isimleriyle çağrılırlar veya bu kümelere isimleriyle referans yapılır.

VBScript'te bu kümelenmiş kod gruplarına Prosedür (Süreç) denir. iki türlüdür: fonksiyon (**Function**) ve **Subroutine.** Bu iki süreç arasındaki başlıca fark, fonksiyondan kendisini çağıran komuta daima bir değer döner; **Sub**'dan dönmeyebilir. **Sub**, yapacağı işi yapar ve programın kontrolünü kendine atıf yapılan noktaya devreder. VBScript'de bir programa farklı yerlerde sık sık aynı işi yaptırıyorsak, bunu bir **Sub** ile yaptırırız; fakat programımıza bir değer gerekiyorsa, bu değeri bir fonksiyona hesaplattırırız. Her ikisi de kendilerine atıfta bulunan veya kendilerini göreve çağıran satırdan (komuttan, deyimden) verilebilecek değerleri kabul edebilirler.

Bir fonksiyonun adı, tıpkı bir değişken adı gibi, fonksiyonun ürettiği değeri tutar; ve bu değer kendisini çağıran komuta verilir. Diyelim ki, programımızın çeşitli noktalarında yazı-tura atıp, elde edilecek sonuca göre bir iş yapmak zorundayız. Bu ihtiyacın doğduğu yerde, yazı-tura komutlarını yazabiliriz. Ancak bu ortaya çok uzun bir programın çıkmasına sebep olur. Oysa yazı-tura işlemlerini bir fonksiyonda toplar ve ihtiyaç halinde sadece bu fonksiyonu çağırırsak ve fonksiyon bize o anda yazı mı geldiğini, yoksa tura mı geldiğini bildirirse, işimiz çok kolaylaşmış olur.

Böyle bir fonksiyon, yukarıdaki örnekten hareketle, şöyle olabilir:

```
<%
Function YaziTura
Dim ParaAt
Randomize
ParaAt = Int(Rnd * 2) + 1
If ParaAt = 1 Then
YaziTura = "Yazı"
Else
YaziTura = "Tura"
End If
End Function
%>
```

Bu fonksiyonu, ASP programının herhangi bir yerinden, şöyle bir yöntemle çağırabilir; ve vereceği sonucu programın akışına uygun şekilde kullanabilirsiniz:

```
<%
NeGeldi = YaziTura
Response.Write NeGeldi
%>
```

Fonksiyonun sonunda **End Function** ifadesinin bulunduğuna ve fonksiyonun elde ettiği sonucu kendi adına atadığımıza dikkat edin. **DIM** ifadesiyle böyle bir değişken tanımlamadığımız halde VBScript, fonksiyonu çağırdığınız anda bunu kendiliğinden yapacaktır.

Aynı işlemi Subroutine (**Sub**) olarak yazabiliriz. Fakat bu kez Sub, elde edeceği değeri kendisi kullanacak ve bittiği anda kontrol programa geri dönecektir:

Argümanli;

```
<%
Sub prosedürün_ismi (arguman1, arguman2, .. , argumanN)
' kodlar, kodlar, kodlar...
End Sub
%>
```

Argümansız;

```
<%
Sub yildirim_gursesin_anisina()
Response.write "leylaklar dökülür, güller ağlasın"
End Sub
%>
```

Fonksiyon adlarının sonuna, bizden beklediği değer varsa onları belirleyen değişken adlarını parantez içinde yazarız. Fonksiyon bizden bir değer beklemiyorsa açılan kapanan (boş) parantezlere ihtiyaç yoktur. ancak bir çok VBScript programcısı bunu adet edinmiştir. **Sub**'ların çağrılması, fonksiyondan farklıdır. **Sub**'ın icra edilmesini istediğiniz noktaya sadece adını yazarız. **Sub**'lar işleyebilmek için bizden değer bekliyorsa, bu değerleri **Sub** adının yanına, parantez içine almadan ve virgülle ayırarak, yazarız. Örneğin, Hesapla isimli ve bizden iki değer bekleyen bir **Sub** şöyle çağrılır:

Argümanlı sub işletimi

```
<%
' tercih 1
Call prosedur_ismi ( arguman1, arguman2, .. , argumanN )
' tercih 2
prosedur_ismi arguman1, arguman2, .. , argumanN
%>
```

Argümansız sub işletimi

```
<%
' başka işler...
Call yildirim_gursesin_anisina()
' veya
yildirim_gursesin_anisina
%>
```

Bu **Sub** işini bitirdiği anda programın akışı, **Sub**'a atıf yaptığımız noktada devam eder.

## EXIT SUB

Son olarak **<% exit sub %>** deyimini anlatarak prosedür konusunu bitirelim. Bu deyimi bir sub bloğu içinde kullanırsanız program akışı, bu deyimi görür görmez o sub bloğundan çıkar ve bu sub hangi satırdan çağrılmışsa o satırdan sonraki kodları icra etmeye koyulur. Bu deyim, bir şartlı sınama bloğu içinde kullanılır. Belli bir koşul gerçekleştiğinde veya gerçekleşmediğinde sub içerisinde, diğer deyimlerin işlenmeden çıkılmasını sağlar.

```
<%
pay = 5
payda = 2
Call bol ( pay, payda )
pay = 16
payda = 0
Call bol ( pay, payda )
Sub bol ( x , y )
If y=0 Then
Response.Write x & " \ " & y & ": müjde! Sıfıra Bölme Hatası..."
exit Sub
End If
bolum = x \ y ' tam bölme
kalan = x mod y
Response.Write x & " \ " & y & " bölümü : "
Response.Write "bölüm: " & bolum & " : kalan: " & kalan & "<br>"
End Sub
%>
```

## TARİH VE SAAT

Belki de Web'in zamana çok bağlı oluşu dolayısıyla, Visual Basic'in hemen hemen bütün zaman-tarih fonksiyonları VBScript'te de kullanılır.

**Date:** Bugün tarihini verir. (25.03.2000 gibi)

**Time:** O andaki saati verir. (22:24:40 gibi)

**Now:** O andaki tarih ve saati birlikte verir. (25.03.2000 22:24:40 gibi)

VBScript'in buna ek olarak **Weekday** (haftanın günü), **WeekdayName** (günün adı) ve **Monthname** (ayın adı) fonksiyonları da vardır. Bu fonksiyonlar değerlerini **Date** fonksiyonuna göre alırlar.

```
<%= WeekdayName(Weekday(Date))%>
```

komutu bize bugün Cumartesi ise "Cumartesi" değerini verir.

```
<%= MonthName(Month(Date))%>
```

komutu bize bu ay Mart ise "Mart" değerini verir. VBScript'in bunlara ek olarak **Day** (gün), **Month** (ay) ve **Year** (yıl) fonksiyonları da değerlerini **Date** fonksiyonundan alarak, size bir rakam verirler. Eğer tarih 25 Mart 2000 ise:

```
<%= Day(Date)%> '... 25
<%= Month(Date)%> '... 3
<%= Year(Date)%> '... 2000
```

değerini verir. VBScript, bu değerleri doğruca işletim sisteminden alır. Dolayısıyla işletim sisteminin bölgesel ayarları Türkiye için yapılmışsa, gün adları Türkçe olarak dönecektir. Ayrıca, tarih ve saat biçimleri de bölgesel ayarlara bağlı olarak, ay önde, gün arkada veya tersi, saat de 12 saat veya 24 saat esasına göre döner. ASP

programlarınızı kişisel Web Server'da denerken kendi bilgisayarınızın tarih ve saatini; gerçek Internet'te çalıştırırken Server'ın tarih ve saatini alırsınız. Sayfalarınızda ay ve gün adlarını Türkçe görüntülemek için, önce Server'ın bölgesel ayarlarını sınamanız ve eğer isimler Türkçe gelmiyorsa, bunları çeviren Sub'lar veya fonksiyonlar yazmanız gerekebilir.

## SERVER NESNESİ

Web Server, ASP için bir nesnedir, ASP'nin bir çok işini bu nesnenin özellikleri ve metodları halleder. Server nesnesinin bir özelliği (**ScriptTimeout**) ve dört metodu (**CreateObject, HTMLEncode, URLEncode, MapPath**) vardır. Web Server çalıştığı bilgisayarın sizin siteniz adına yönetiminden sorumludur; dolayısıyla bu kadar az özellik ve metodu var diye bu nesneden çok yararlanmayacağımızı sanmayın. ActiveX ve COM bileşenlerini çalıştırmak Server'ın görevidir.

### ScriptTimeout Özelliği:

Diyelim ki bir ASP Script'i ya bizim, ya ziyaretçinin, ya da Server'ın bir hatası yüzünden sonsuz döngüye girdi! Döngünün durması için gerekli şart asla yerine gelmiyor ve Script bir türlü yapacağı işi yapıp, sonlandırmıyor. Bu durumlarda ziyaretçinin ve tabiî Server'ın sonsuza kadar beklemesi mümkün değil! Programın bir şekilde durdurulması gerekir. Bunu hemen hemen bütün Web server programlarının **Script Timeout** (Script süre sınırı) diyalog kutusuna bir değer girilerek yapılır. Öreğin MS-Internet Information Server için varsayılan **Script Timeout** süresi 90 saniyedir. Yani ISS, herhangi bir Script'in çalışıp-durmasını 90 saniye bekler; bu sürenin sonunda Script'in çalışması tamamlanmazsa ziyaretçiye arzu ettiği sayfanın veya unsurun bulunamadığını bildirir. Bu süreyi (Server'ın varsayılan değerinin altında) kısaltmak değilse bile uzatmak elimizdedir. Bunu **ScriptTimeout** özelliğini kullanarak yaparız. ASP sayfasının herhangi bir yerine örneğin şu kodu koymak yeter:

```
<% Server.ScriptTimeout = 100 %>
```

Bu örneğe göre Server'ın varsayılan **Script Timeout** süresi 90 saniye ise 100 saniyeye çıkmış olur.

Script'iniz çok karmaşık veya başka bir Server'daki veritabanından veri çekiyor, olabilir. Gerçi bu anlamda 90 saniye bilgisayar programcıları için bir asır anlamına gelir, ama yine de durdurulmasaydı işini başarıyla tamamlayacak bir Script, bu sürenin kısalığı yüzünden Server tarafından durdurulabilir. ASP sayfalarınız çok karmaşıksa ve sürekli **Timeout** hatası veriyorsa, hata aramadan önce bu süreyi uzatabilirsiniz.

### CreateObject Metodu:

İlk ASP kodunu yazdığımız andan beri bu metodu kullandığımızı biliyorsunuz. **CreateObject** (nesne oluştur) olmasa idi, asp'de olmazdı. Fakat sürekli kullandığımız **CreateObject** ile bu **CreateObject**'i birbirine karıştırmayın. Yukarıda kullandıklarımız **Scripting** nesnesinin bir metodu idi; bu Server nesnesine aittir. Diyelim ki sayfanızda reklam amaçlı **banner** grafiklerini belirli zaman aralığı ile veya ziyaretçiye gönderdiğiniz **Cookie** (çerez) bilgilerine göre değiştirmek istiyorsunuz. Bunun için

diyelim ki MS-Web Server Programının **AdRotator** bileşininden yararlanacaksınız; şöyle bir kod işinizi görebilir:

```
<% Set Reklam = Server.CreateObject ("MSWS.AdRotator")%>
<%= Reklam.GetAdvertisement("/reklamlar/buyukbanka.txt")%>
```

Burada **GetAdvertisement**, Server'ın **AdRotator** bileşininin bir metodudur. Server'ın **CreateObject** metodundan, veritabanına ulaşırken de yararlanacağız.

**MapPath (Yolu belirle) Metodu:**

Web Server açısından "kök dizin" (**root directory**) Server'ın bulunduğu bilgisayarın sabit diskinde, herhangi bir klasör olabilir. Örneğin IIS için bu varsayılan değer olarak "C:\inetbup\wwwroot" klasörüdür. Özellikle ASP ile "program niteliğinde siteler" yapmaya başladığımızda, sitenin ilgili bütün dosyalarının bulunduğu bir dizin için yol belirlemek isteyebiliriz. Bunu Server nesnesinin **MapPath** (Yolu belirle) metodu ile yapabiliriz:

```
WebDizini = Server.MapPath("/benim_site")
```

Bu komutla WebDizini değişkenin değeri muhtemelen şöyle olacaktır: "C:\inetbup\wwwroot\benim_site\" Fakat bu metodun sadece böyle duragan biçimde kullanılması gerekmez; bazen sayfalarımızda ziyaretçi ile etkileşmenin sonucu olarak varsayılan Web dizinimizi değiştirmek isteyebiliriz. Sözgelimi biri Türkçe, diğeri İngilizce iki sitemiz varsa, ve ana sayfamızda ziyaretçi Türkçe'yi seçtiyse, o noktadan itibaren Web uygulamamız için Web kök-dizini, "/turkish/" olacak ve mesela resimlerimiz için verdiğimiz "/resimler/" dizini kök dizinde değil, "/turkish/resimler/" klasöründe aranacaktır. Web yolunu dinamik olarak, yani ziyaretçinin tercihine bağlı şekilde değiştirebilmek için, önce ziyaretçiden gelecek bilgileri nasıl kullanacağımıza, yani **Request** (talep) nesnesine değinmemiz gerekir.

**HTMLEncode, URLEncode:**

İçinde HTML açısından kod parçası veya özel işaret sayılan karakterler bulunan metinleri sayfamıza içerik olarak göndereceğimiz zaman Server'ın işaretleri aynen metin gibi göndermesini sağlamak için, örneğin:

```
Server.HTMLEncode("Değisken1 < Değisken2")
```

yazarsak, ASP bu metni HTML kodu olarak yorumlamaz, metin olarak algılar.

Internet'te bazen özellikle sayfa adresleri belirtilirken bazı değerlerin "URL Kodu" dediğimiz şekilde kodlanmış olarak gönderilmesi gerekir. Bu kodlama türünde boşlukların yerine + işareti konmuş olması şarttır. Bu tür bilgiler göndereceğimiz zaman:

```
Server.URLEncode("kelime 1 kelime2 kelime3")
```

şeklindeki bir kod Bunu hemen şu şekle sokacaktır: kelime1+kelime2+kelime3

**Adrotator (Değişen Reklam Banner'ları):**

Bu Componentimiz sayesinde, sayfamızda her girişimizde veya sayfayı her Refresh edişimizde değişen bannerlar koyabileceğiz.

Ayrıca her banner'a ayrı ayrı link verebilecek, ve hatta hangi banner'ın kaç defa gösterileceğini ayarlayabileceğiz... Hemen nasıl yapıldığını inceleyelim...

```
<%@ Language=VBScript Codepage="1254"%>
<%
Set Reklam=Server.CreateObject("MSWC.AdRotator")
banner=Reklam.GetAdvertisement ("Banner.txt")
Response.Write Banner
%>
```

Banner.txt isimli txt dosyamız da aşağıdaki gibi olacaktır...

```
Redirect http:Rating.asp
width 423
height 53
border 0
*
Banner1.gif
http://www.massCars.com
MassCars
5
Banner2.gif
http://www.YemekTarifim.Com
Türkiye'nin En Büyük Sanal Mutfağı
6
Banner3.jpg
http://www.EgitimCenter.Com
Türkiye'nin En Büyük Sanal Dersanesi
1
```

Şimdi bu textimizde kullandığımız terimleri tanımlayalım.

WIDTH: Bu alana Bannerlarınızın Genişliğini gireceksiniz.

HEIGHT: Bu alana da Bannerlarınızın Yüksekliğini gireceksiniz.

BORDER: Bu alana o girerseniz Bannerlarınız çerçeve kullanmaz. Gireceğiniz 1 ve daha yüksek değerler, Banner'ınıza çerçeve vermenizi sağlar.

İlk Satırda imajın adını ve yolunu yazıyorsunuz.

İkinci Satırda, Linkini

Üçüncü satırda ise imajın üzerine gelince çıkacak ALT yazısını.

Dördüncü satırdaki rakamlarımızın toplamı 10'u verir. Bu rakamlar seçili banner'ımızın 10 defada kaç kere gösterileceğini belirler...

## Contentrotator

Bu başlık altında ASP'nin içerik çevirici özelliğini göreceğiz. Bir sayfaya girdiniz ve sayfada bilgisayar reklamları var. Düşünsenize her sayfaya girişinizde Bilgisayar dünyasıyla ilgili başka bir haber buluyorsunuz. Ne kadar ilginç değil mi?

```
<%
Set Tip = Server.CreateObject("MSWC.ContentRotator")
Response.Write Tip.ChooseContent("Content.txt")
%>
```

Evet yazacağımız kod bu kadar. Önemli olan Content.txt adlı dosyamızda ne yazacağımız ve ne yapacağımız. Şimdi isterseniz Content.txt adlı dosyamızı inceleyelim...

```
%% #2// Fikra1
> HOCANIN HANIMI
> Nasrettin Hoca'ya dert yanıyorlar:  Yahu Hoca senin hanım çok
>geziyor. Hoca:Olur mu canım? O kadar gezse arada bir bizim eve de uğrar!

%% #3// Fikra2
> İPE UN SERİLİR Mİ?
> Komşusu ,Hoca'dan çamaşır ipi ister.Hoca eve girer çıkar. Komşusuna :
> "Bizimkiler ipe un sermişler" der. Komşusu: Hocam ipe un serilir
> mi? deyince Hoca: "İpi vermek istemeyince serilir..." der.

%% #5// Fikra3
> DÜNYANIN ORTASI
> Çevreden bir grup insan, Hoca'yı çevirmişler. Hocam size bir sorumuz
> var demişler ve anlatmışlar: Hocam dünyanın ortası neresi?¿ Hoca, beş
> on adım ilerlemiş, bastonunu yere saplamış. Dünyanın ortası burasıdır
> demiş. Şaşkın şaşkın bakan cahiller,Nasıl olur Hocam ??derler.
> Hocam'da: İnanmazsanız ölçün... der.
```

Gördüğünüz gibi text dosyamız oldukça basit bir yapıya sahip. Hemen incelemeye alalım... Sanırım size tek farklı görünen karakterler %% #2// Fikra1 karakterleridir sanırım. Hemen bunların ne olduğunu açıklayalım.

%% işaretleri her bir bölümü birbirinden ayırmak için kullanılıyor.

Hemen arkasından gelen #2// işareti ise sayfanın toplamda kaç kere gösterileceğini belirliyor. Örnekte sayfanın yüklenme oranı sayıların toplamıyla bulunur. Örneğimizde yüklenme toplamı 10'ur. Mesela bu Fıkra #2// olduğu için sayfa 10 kere yüklenirse veya Refresh edilirse 2 kere gösterilecektir...

## Page.Counter

Bir sayfanın kaç defa ziyaret edildiğini göstermek için kullanılır.

```
<%
Set MyPageCounter = Server.CreateObject("MSWC.PageCounter")
MyPageCounter.PageHit
%>
Bu Web sayfası <%= MyPageCounter.Hits %> kez görüntülenmiştir.
```

## REQUEST NESNELERİ

### Request Objesi:

Request objesi bilgi toplamak amaçlı kullanılan bir objedir. Detaylı olarak anlatılacak olmasına rağmen bu obje içersinde (koleksiyon) yer alan metotlara kısaca bakalım.

**1-Querystring**: Eğer gönderilen bilgi url içersinde bulunan dosya ismi ile birlikte taşınıyorsa başka bir ifadeyle bilgi formu metodundan "GET" ile gönderiliyor ise Querystring metodu bu bilginin elde edilmesi için kullanılır.

**2-Form**: Eğer bilgi Form içerisinde "POST" metodu ile gönderiliyor ise bilginin elde edilmesi için kullanılan bir metodudur.

**3-Servervariables:** Web server request ile ilgili bilgileri (Http ServerVariables) tutar. Bu bilgilere bu koleksiyon içerisinde ulaşmayı sağlayan bir metoddur.

**4-Cookies:** Eğer client browser serverdan gelen cookileri (çerez: text dosyaları) kabul ediyor ise bu bilgi web servera cookie koleksiyonu içerisinde ulaşır.

**5-ClientCertificate**: Client Certificate dijital bir sertifikadır ve client ve web server arasında ulaşılan server ve client'ın birbirini tanımlaması için kullanılır.

### QueryString Koleksiyonu

Querystring server a iletilen bilgi kümeciği metotlarından biridir. Bu iletişim browser ın adres hanesinde yer alan dosya adına "?" ve "&" işaretleri vasıtası ile bilgilerin eklenmesi suretiyle gerçekleşir. Genel kullanımı

**filename.asp?kullanici=Mehmet**

şeklindedir.

Eğer daha fazla bilgi bir anda gönderilmek isteniyor ise bu "&" işareti kullanılarak;

**filename.asp?kullanici=Mehmet&email=mehmet@erciyes.edu.tr**

şeklinde gerçekleştirilir. "&" işareti bilgilerin birbirinden ayrılmasını sağlar. Kullanıcı adı ve soyadını alan ve bunları Querystring metodu ile ilgili forma gönderecek bir form dizayn edelim. İlgili Html formu aşağıdaki gibi olmalıdır;

```
<form name=login Action=Querystring.asp metod="GET">
Lütfen Adınızı Giriniz: < input type="text" name"adi" > <BR>
Lütfen SoyadınızıGiriniz:< input type="text" name="soyadi" > <BR>
<input type="submit" value="login" >
</Form >
```

Yukarıdaki kod adı soyadı alanları bulunan bir form elde etmemizi sağlar.

Bu Html kodu içerisinde yer alan input submit butonu <form action="Querystring.asp" metod= "GET"> bu kutucuklara girilen bilgileri action kısmında belirtilen Querystring.asp'ye "GET"metodu kullanılarak gönderilir. (Yani bilgiler browser'ın adres hanesi kullanılarak iletilir) Bu bilgilere ulaşmak içinse Request' in Querystring metodu aşağıdaki şekilde kullanılır.

Request. Querystring ("istek gönderen formdaki Html elemanının adı") bu genel kullanıma göre eğer biz "..." içerisine o formda yer alan bir kontrolün adını koyar, istek o input alanına girilmiş veriye ulaşırız, yani Request. Querystring ("adi") şeklinde bir kullanım "adi" adlı text box'a girilen veriye;

Request. Querystring ("soyadi")şeklinde bir kullanım ise "soyadi" adı verilen textbox' a girilen veriye ulaşmamızı sağlar. Eğer Response nesnesinin Write metodunu kullanırsak login formuna girilen verileri;

```
<%
Response.write Request. Querystring ("adi") & "<BR>"
Response.Write Request. Querystring ("soyadi")
%>
```

şeklinde ekrana yazabiliriz. Dikkat edecek olursanız "&" işareti "<BR>" Html etiketini formdan gelen adi verisine eklemek için kullanılmıştır. Bu da bize 1 satır aşağıya geçmemizi sağlar (satır beslemesi) sağlar. Koleksiyon birden fazla değer ve değişkeni içerisinde bulunduran bir grup olarak tanımlanabilir. Eğer Querystring metodu kullanılarak gönderilen değişken (kontrol sayısı) birden fazla ise bu bir koleksiyon oluşturuyor anlamı taşır. Eğer daha önceki notlarımızı hatırlayacak olursak koleksiyon oluşturan bilgi grupları veya dizin elemanları için kullandığımız özel bir döngümüz vardı. (FOR EACH....NEXT) bu döngüyü Querystring için uygulanacak olursak ;

```
<%
For Each eleman in Request. Querystring
Response.Write eleman & "....." & Request. Querystring(eleman)
Next
%>
```

şeklinde bir döngü ile bize Querystring ie gönderilen tüm elemanları ve bu elemanların değerlerini ekrana basabiliriz. Yukarıdaki döngü Request. Querystring ile oluşturulan her eleman için döngüyü tekrarlar.

Request. Querystring ("kontrol").count özelliği gönderilen elemanları saymak için kullanılır. Eğer bu sayı"o" 'a eşit ise herhangi bir bilgi gönderilmemiş demektir.

Şimdi değişik konularda başlıkların yer aldığı ve bu başlıklar arasından kullanıcının yaptığı seçimlere bağlı olarak bu konularda kendisine bilgi göndereceği mesajını ekrana basan bir ASP uygulaması oluşturalım. Kullanıcının seçim yapabileceği selectbox'ın yer aldığı Html formunu dizayn edelim:

```
< HTML >
< HEAD >
< TITLE > kitap başlıkları < /TITLE >
< /HEAD >
< BODY >
Aşağıdaki konu başlıklarından ilgilendiklerinizi seçiniz.
< FORM ACTION="responseQuerystring.asp" METHOD="GET" >
< Select size=3 name="Konular" MULTIPLE >
< OPTION > Bilgisayar < /OPTION >
< OPTION > Hikaye < /OPTION >
< OPTION > Şiir < /OPTION >
< OPTION > Roman < /OPTION >
< /select >
< input type= "submit" value= "Gönder >
< /FORM >
< /BODY >
< /HTML >
```

Şimdi bu formdan gönderilen verileri işleyen ASP kodunu oluşturalım:

```
Kitap istek formu : < BR >
<%
if Request. Querystring("Konular").count= 0 then Response.write "Herhangi bir konu
seçmediniz."
ELSE
Response.write size seçmiş olduğunuz "&"< BR >
Response.Write Request. Querystring ("konular")
Response.write "hakkında broşür yollayacağız. Teşekkürler"
END IF
%>
```

İlk form basit bir html formudur. Selectbox butonun oluşturulması için kullanılmıştır. <select....multiple> yer alan multiple seçeneği CTRL tuşuna basılarak birden fazla seçim yapılmasına olanak tanır. ASP içersinde yer alan Request.Querystring("konular").count=0 şartı eğer hiçbir konu başlığı seçilmemiş ise devreye girer. Eğer kullanıcı formda seçim yapmış ise ELSE şartı devreye girer ve seçilen konu başlıkları ekrana yazılır.

### Form Koleksiyonu

Eğer form içerisinde gönderilen bilgiler POST metodu kullanılarak gönderilmiş ise bu bilgiler FORM koleksiyonu içerisinde yer alır. Bu bilgiler request .FORM kullanarak kontrol edilebilirler. Genel kullanımı:

```
Request. Form ("kontrol_adi")
```

şeklindedir.

Form nesnesi içerisinde textbox dışında aşağıda ki elemanlarda bulunabilir. Bunlar:

- Textbox
- Checkbox
- Option Buttons
- Listbox ve türevleri
- Hidden Fields
- Text Areas

Formun POST metodu ile gönderilen bilgiler HTML veri akışı içerisinde kullanıcıya gönderilir, bilginin iletilmesi için browser ın adres hanesi kullanılmaz. Bir önceki örneği form koleksiyonu ile oluşturmak isteseydik ;

```
Request.Form ("Konular")
```

şeklinde bir değişiklik yapmamız yeterli olacaktı.

### ClientCertificate Koleksiyonu

Web browser, web server ile Secure (SSL) bağlantısı üzerinden bağlantı kurmak istediğinde bu işlem dijital sertifikalar üzerinden sağlar. Bu dijital sertifika bağlantı yapılan web server ve organizasyon hakkında gerekli bilgileri taşır bu CA Certificate Authority olarak bilinir. Bu işlem bilgilerin güvenli bir bağlantı üzerinden aktarılmasını sağlar bilgi aktarımı SSL, Secure Sockets Layer protokolü üzerinden gerçekleştirilir. SSL, HTTP protokolünün daha güvenli hale getirilmiş bir varyasyonudur. En son versiyonu SSL 3.0/PCT 1 dir. (PCT: Private Communication Technology) Clientcertificate de bir koleksiyon meydana getirir ve bu koleksiyona:

```
<%
For Each Key in Request. ClientCertificate
Response.Write ( Key & " : " & Request.ClientCertificate (key) & "<BR>")
Next
%>
```

şeklinde ulaşabilirsiniz.

Bu koleksiyonda yer elemana ise ; Request. ClientCertificate (Key) şeklinde ulaşabilirsiniz. İlerleyen bölümlerde ClientCertificate ile ilgili daha ayrıntılı bilgilere ve örneklere yer verilecektir.

### Request Nesnesinin Özellikleri ve Metodları

Request nesnesi koleksiyonun dışında da bazı özel özellik ve metotlara sahiptir. Bu özelliklere ve metotlara göz atalım .

## TotalBytes Özelliği

Bu özellik browser tarafından gönderilen bilginin toplam olarak kaç byte olduğunu bulmak için kullanılır. Aşağıda ki kod gönderilen bilginin kaç byte olduğunu ekrana yazar

```
<%
Response.Write "Göndermiş olduğunuz Bilgi:"
Response.Write Request.TotalBytes & "dır"
%>
```

## BinaryRead Metodu

BinaryRead metodu POST ile server a gönderilen bilgilerin alınması için kullanılır. Form ve querystring metodlarında farklı olarak text dışında veri gönderildiğinde bu metod bu verilerin okunması için kullanılır. Daha önce anlatılan TotalBytes özelliği bu metoda parametre olarak sunulur.

```
<%
Dim Dosya_boyutu, Oku
Dosya_boyutu=Request.Totalbytes
Oku=Request.BinaryRead(Dosya_boyutu)
For i= 1 to Dosya_boyutu
Response.Write MidB (oku, i, 1)
Next
%>
```

MidB, Mid fonksiyonuna benzeyen fakat düzensiz oluşturulmuş bilgileri okumak için kullanılan bir text fonksiyonudur. (Aslında N boyutlu dizi haline getirilmiş veriler demek daha doğru olur.) Bu fonksiyon döngü içerisinde her defasında bir karakter okunur ve bu karakter ekrana yazılır.

## ServerVariables (Server Değişkenleri)

Request nesnesinin bir diğer kolleksiyonu, bizim kendi Web Server'ımızın o anda çalışmakta olan ASP sayfası için oluşturduğu ortamın değişkenleridir. Bunların arasındaziyaretçinin Browser'ına ilişkin bilgiler de vardır. Önce şu kısa ASP sayfasını çalıştırarak kendi Server'ımızın şu andaki değişkenlerini görelim; sonra bunları ayrıntılı ele alalım;

```
<HTML>
<HEAD>
<TITLE>HTTP ServerDegişkenleri Kolleksiyonu</TITLE>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
<BODY BGCOLOR=white>
<CENTER>
<H2>HTTP Server Değişkenleri Kolleksiyonu</H2>
</CENTER>
<TABLE BORDER=1>
<TR><TD><B>Değişkenin adı</B></TD> <TD><B>Değeri</B></TD></TR>
<% For Each key in Request.ServerVariables %>
<TR>
<TD><% = key %></TD>
```

```
<TD>
<%If Request.ServerVariables(key) = "" Then
Response.Write " "
Else
Response.Write Request.ServerVariables(key)
End If
Response.Write "</TD>"%>
</TR>
<% Next %>
</TABLE>
<p>
Sizin Host'unuzun adı:<B><%=Request.ServerVariables("HTTP_HOST")%></B>
</BODY>
</HTML>
```

Bu sayfayı çalıştırdığımız zaman görüntülenecek tabloda, Bir HTTP Server'ın sayfasını çalıştırdığı anda oluşturduğu ortama şekil veren bütün değişkenleri göreceksiniz. Bu değişkenleri programlama yoluyla değiştiremeyiz; sadece okuyup, yararlanabiliriz. **HTTP Server Değişkenleri Kolleksiyonunun** elemanları şöyle sıralanır:

| **Değişkenin adı** | **Değeri** |
|---|---|
| **ALL_http** | HTTP **Header** içinde yer alan bütün değişkenler ve değerleri. **Header** adlarının önünde "HTTP_" öneki vardır. |
| **ALL_RAW http** | Header içinde yer alan bütün değişkenler ve değerleri. Header adları ve değerleri istemci Browser'ın verdiği şekilde gösterilir. |
| **APPL_MD_PATH** | Web Server'ın ISAPI.DLL dosyası için varsaydığı kök dizin |
| **APPL_PHYSICAL_PATH** | Web Server'ın varsaydığı kök dizinin gerçek yolu |
| **AUTH_PASSWORD** | Kullanıcı Web Server'a kullanıcı adı/parola yöntemiyle bağlanabiliyorsa, kullanılan parola |
| **AUTH_TYPE** | Kullanıcı Web Server'a kullanıcı adı/parola yöntemiyle bağlanabiliyorsa, kullanılan yetkilendirme yöntemi |
| **AUTH_USER** | Kullanıcı Web Server'a kullanıcı adı/parola yöntemiyle bağlanabiliyorsa, kullanıcı adı |
| **CERT_COOKIE** | Kullanıcı siteye bağlanmak için yetkilendirme sertifikası kullanıyorsa kendisine verilen kimlik (ID) |
| **CERT_FLAGS** | Sertifikanın varlığını gösteren bit değeri |
| **CERT_ISSUER** | Sertifika varsa veren kurum |
| **CERT_KEYSIZE** | Secure Socket Layers kullanılıyorsa, bağlantı için anahtar değeri |
| **CERT_SECRETKEYSIZE** | Özel anahtar değeri |

| | |
|---|---|
| **CERT_SERIALNUMBER** | Sertifika seri no. |
| **CERT_SERVER_ISSUER** | Sertifikayı veren merci |
| **CERT_SERVER_SUBJECT** | Server Sertifikasının "konu" alanı değeri |
| **CERT_SUBJECT** | İstemci Sertifikasının konu alanı değeri |
| **CONTENT_LENGTH** | İstemcinin gönderdiği bilgi yumağının boyutu |
| **CONTENT_TYPE** | Ziyaretçiden gelen bilgilerin GET veya POST metoduna göre edindiği tür |
| **GATEWAY_INTERFACE** | Web Server'ın ziyaretçi ile etkileşim arayüzünün adı ve sürümü. Genellikle: CGI/1.1 |
| **HTTPS** | Ziyaretçi ile bağlantı güvenli ise ON, değilse OFF |
| **HTTPS_KEYSIZE** | **Secure Sockets Layer** için bağlantı anahtar sayısı |
| **HTTPS_SECRETKEYSIZE** | Özel Server sertifikasının gizli anahtar sayısı |
| **HTTPS_SERVER_ISSUER** | Özel Server sertifikasının veren merci |
| **HTTPS_SERVER_SUBJECT** | Özel Server sertifikasının konusu |
| **INSTANCE_ID** | Web Server'ın aynı anda kaç kere çalışmakta olduğu |
| **INSTANCE_META_PATH** | Şu anda çalışmakta olan Web Server'ın Meta yolu |
| **LOCAL_ADDR** | İstemcinin IP numarası |
| **LOGON_USER** | İstemci Windows NT sisteminde ise oturum açma adı |
| **PATH_INFO** | Çalışmakta olan ASP'nin göreli yolu ve adı |
| **PATH_TRANSLATED** | Çalışmakta olan ASP'nin gerçek yolu ve adı |
| **QUERY_STRING** | İstemcinin gönderdiği bilgi kümesi |
| **REMOTE_ADDR** | İstemcinin Host'unun (ISS'inin) IP'si |
| **REMOTE_HOST** | İstemcinin Host'unun (ISS'inin) adı |
| **REMOTE_USER** | İstemcinin gerçek adı |
| **REQUEST_METHOD** | İstemciden bilgi isteme yöntemi (GET veya POST) |
| **SCRIPT_NAME** | Çalışmakta olan ASP'nin adı |
| **SERVER_NAME** | Sunucu'nun adı |
| **SERVER_PORT** | Sunucuya bağlantının geldiği TCP kapı numarası |
| **SERVER_PORT_SECURE** | TCP kapısı güvenli ise 1, değilse 0 |
| **SERVER_PROTOCOL** | Server'ın çalıştırdığı HTTP'nin sürümü |
| **SERVER_SOFTWARE** | Server programının adı ve sürümü |
| **URL** | Şu anda geçerli URL |

## RESPONSE NESNELERİ

Bu başlık altında her iki objeyi daha detaylı olarak inceleyeceğiz. Bu objeler browser ve web server arasındaki iletişiminin sağlanması için kullanılırlar. Şimdi bu detaylara ayrıntılı olarak bakalım.

### Write Metodu:

Response objesinin en sık kullanılan metodudur. Write metodu ASP sayfamızda bilgileri görüntülemek için kullanılır. Genel kullanımı;

**Response.Write [değer]**

şeklindedir.

```
<%
Yazılacak_text="Bugün günlerden ne?"
Response.Write yazılacak_text
%>
```

Şeklinde bir kullanım browser ekranına "Bugün günlerden ne?" diye bir mesaj yazar. Asp script bildiğiniz gibi <%.....%> script sınırlayıcıları arasına yazılır. Bu kullanımın iki farklı şekli vardır. Eğer html tagları içerisinde herhangi bir değişkenin taşıdığı bir değeri görüntülemek istiyorsak <%=değer%> daha uygun bir kullanım olacaktır. Fakat bu kullanımın dezavantajı sadece 1 satırlık bir kodlamaya izin vermesidir. Yani;

```
<%=değer
Response.Write "Bu kullanım hata verir."
%>
```

Şeklinde bir kullanım yanlıştır. Dolayısıyla birden fazla satır ASP kodlaması gereken durumlarda ASP kodunu <%.....%> satırları arasına yazmak daha uygun olacaktır.

```
<%
text1= "Merhaba"
text2= "Yeni ASP programcıları"
Response.write text1 & text2
%>
```

Response.write text1 & text2 kullanımı iki farklı değişken içerisinde tutulan stringleri birleştirerek tek satırda yazmak için kullanılmıştır. "&" işareti iki değişkeni birleştirmek için kullanılır. Response.write bir işlemin sonucu ekrana yazmak içinde kullanılabilir. Aşağıdaki örneği inceleyelim;

```
<%
sayı1= 24
sayı2= 8
Response.write sayı1/sayı2
%>
```

Şeklinde bir kullanım ekrana 3 yazılmasına neden olur. Çünkü sayı1 /sayı2 yani 24/8=3 dür, ve response.write 3 şeklinde bir kullanıma eşdeğerdir. Bir formdan bize iletilen bilgileri Request objesini kullanarak alabilir ve gene bu bilgiyi Response objesinin write metodunu kullanarak ekrana yazabiliriz.

```
<%
Response.Write (Request.QveryString("text1"))
%>
```

şeklinde bir kullanım bize istek gönderen formdaki text1 adlı değişkeninin değerini ekrana yazar.

**Buffer:**

Buffer özelliği response ile oluşturulan Html Data akışının html dosya oluşturulması tamamlandıktan sonra veya her satır iletildiğinde gösterilmesi ile ilgili bir özelliktir. Default değeri "on" dur bu değere false u set edecek olursak ;

```
Response.Buffer=False
```

şeklinde bu işlem yapılabilir.

**Clear:**

**Response.Clear** metodu buffer (tampon) edilmiş tüm html bilgisinin silinmesi için kullanılır. Kullanımı: Response.clear şeklindedir. Fakat unutulmaması gereken nokta eğer Response.Buffer= false olarak set edilmiş ise Run-time (çalışma zamanı ) hataya neden olur.

**End:**

End metodu web server işlemi durdurmasını ve Response.End noktasına kadar oluşturulmuş Html datanın gönderilmesini sağlar. Kullanımı :

**Response.End** şeklindedir

**Expires:**

Expires özelliği dakika cinsinden sayfanın browser tarafından hafızada ne kadar tutulacağı ile ilgili bilgiyi set etmek için kuyllanılır. Eğer kullanıcı aynı sayfaya Response.Expires ile belirlenen zamandan daha önce geri dönerse sayfanın hafızadaki versiyonu kullanıcıya gösterilir. Kullanımı :

```
<% Response.Expires=dakika %>
```

şeklindedir. Eğer bu değere "0" set edilirse (**Response.Expires=0**) sayfaya her geri dönüldüğünde sayfa yeniden yüklenir.

**ExpiresAbsolute:**

ExpiresAbsolute tıpkı Expire özelliğibne benzer olarak fakat belirtilen sürenin dakika değil tarih veya saat olacak şekilde set edilmesi suretiyle sayfanın geçerlilik süresinin belirlenmesi için kullanılır. Kullanımı :

**<% Response.ExpiresAbsolute=#tarih# %>**

şeklindedir.

**Redirection:**

Kullanıcıyı bir Asp sayfasından diğerine yönlendirmek için kullanılan bir metoddur. Bu metodun yerini alan 2 yeni metod olan server.transfer ve server.Execute metodlarına daha sonra ayrıntılı olarak göz atacağız. Redirection iel ilgili olarak bilmemiz gereken bazı önemli noktalar vardır. Eğer http header client a gönderilmil ise Response.redirection hataya neden oluyur . bu hatanın önüne geçmek için response. Buffer özelliğini true ya set etmek sureiyle ve herhangi bir noktada response.clear ı kullanarak response.Redirection in hata vermesini engelleyebiliriz. Response.Redirectionun genek kullanımı:

**Response.Redirection("yönlendirilecek_sayfa")** şeklindir.

**Server. Execute ve Server.Transfer**

Server. Execute ve Server.Transfer bir ASP sayfası içersinden başka bir sayfanın çalıştırılması veya o sayfanın içeriğinin aynı sayfa içerisinde gösterilmesi için kullanılır. Aralarında ki tek fark ise Server. Execute kullanıldığında hedef sayfa çalıştırıldıktan sonra orijinal sayfa nın işletilmesine devam edilmesine rağmen Server.Transfer kullanıldığında orijinal sayfaya geri dönülmez.

## ÇEREZLER (COOKIES)

Daha önce girmiş olduğunuz bir sitede sizden abone olmanız istenebilir. Ve genellikle abonelik gerektiren bu tür sitelerde “Beni hatırla” seçeneği mevcuttur. Şayet bu seçeneği işaretlediğinizde o sitenin sizi adınızla karşılaması sizi şaşırtmış olabilir. Web sitesinin sizi hatırlamasını sağlayan nesneler çerez adı verilen ve sizin bilgisayarınızda saklanan küçük dosyalardır.

ASP ile istemci bilgisayarlara çerez yollamak çok basit bir iştir. ASP nin yerleşik elemanlarından RESPONSE un bu iş için özel bir metodu var: COOKIES . İstemciye çerez yollamak için bu metodla birlikte anahtar değerler (bir veya iki) gönderiyoruz. (bir ayrıntı: ASP de iki anahtar sadece cookies kolleksiyonunda kullanılıyor).

```
Response.cookies("kabuk")= "Evet, kabuk"
Response.cookies("kabuklarim")("adi") = "Mucit"
Response.cookies("kabuklarim")("sevdigi_icecek") = "Kızılcahamam Maden Suyu ve Sodası"
Response.cookies("kabuklarim")("medeni_durumu") = "Bekar"
```

```
Response.Cookies("kabuklarim").Expires = Now() + 90
```

Çerezi tarayıcıya yukarıdaki gibi yolladık. Peki bu çerezleri görmek istediğimiz zaman ne yapmamız gerekiyor. İşte o zaman ASP nin diğer bir nesnesi REQUEST bu iş için kullanılır. Bu metodu kullanmak, response ile çerez yollamaya çok benzer. Ancak bir fark vardır. Response de ilgili anahtara atama yapılır, burada ise request zaten bize bir değer getirir, biz de bu değeri işimize geldiği gibi kullanırız. Aşağıda REQUEST ile çerezden bilgi almayı görüyorsunuz.

```
Adim = Request.cookies("kabuklarim")("adi")
ne_icerim = Request.cookies("kabuklarim")("sevdigi_icecek")
medeni_halim = Request.cookies("kabuklarim")("medeni_durumu")
Response.Write Adim & " / " & ne_icerim & " / " & medeni_halim
' sayfadaki çıktı: Mucit / Kızılcahamam Maden Suyu ve Sodası / Bekar!
```

Bir abonenin siteyi kaç kez ziyaret ettiğini göstermek için aşağıdaki kodları kullanabiliriz.

```
<html>
<head>
<title> KİŞİSEL ZİYARET SAYACI </title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<meta name="Generator" content="EditPlus">
</head>
<body bgcolor="#FFFFFF">
<%
ziyaret = Request.cookies("sitem")("ziyaret")
If ziyaret = "" Or Not Isnumeric(ziyaret) Then
Response.cookies("sitem")("ziyaret") = "0"
Response.Cookies("kabuklarim").Expires = Now() + 365
Response.Write "Bu siteyi ilk defa ziyaret ediyorsunuz..."
Else
ziyaret = ziyaret + 1
Response.cookies("sitem")("ziyaret") = ziyaret
Response.Cookies("kabuklarim").Expires = Now() + 365
Response.Write "Siteye " & ziyaret & ". gelişiniz."
End If
%>
</body>
</html>
```

## APPLICATION

Belirtilen uygulamadaki tanımlı bütün kullanıcıların bilgiyi paylaşmaları için Application nesnesini kullanabiliriz. Bir ASP-uzantılı uygulama, sanal dizin ve onların altdizinlerindeki tüm .asp dosyaları gibi tanımlanır.

Application nesnesi birden fazla kullanıcı tarafından paylaşılabildiği için, kullanıcıların bir özelliği aynı anda değiştirme girişimlerini önlemek için Lock ve Unlock yöntemleri kullanılır. Bir örnek yaparak çalışmasını inceleyelim. Bu örneğimizle kullanışlı sayfa sayaçları yapabilirsiniz.

```
<%
Response.Expires = 0
'Expires sayesinde sayfanın içeriğinin bilgilerinin cache bellekte saklanmasını engelleriz.
%>
<%
If (Application("Sayac") = "") Then
'Bu ziyaretçi için Application başlatılmadıysa başlatılsın.
Application("Sayac") = 0
End If
Application.Lock
'Application'ı Kilitliyoruz. Ve kullanıcı giriş yaptıktan sonra 1 değer
'arttırıyoruz.
Application("Sayac") = Application("Sayac") + 1
Application.UnLock
'Ve tekrar serbest bırakıyoruz. Diğer kullanıcılarla bu Application kullanılabilsin diye...
%>
Bu Sayfaya <%= Application("Sayac") %> Kere Giriş Yapıldı.
<P><A HREF="sayac.asp">Yenilemek İçin Tıklayın.</A>
```

Bu kodu yazıp çalıştırdığımız zaman, sayfayı her yenilediğimizde sayaç bir değeri kadar arttırılacaktır. Eğer Sayaç+1 yerine 5 yazarsanız sayacınız 5'er 5'er artacaktır... İsterseniz başka bir örnekle bu nesnemize ait bilgilerimizi pekiştirelim...

Diyelim ki bir chat scripti yazıyorsunuz ya da bir Active Users Scripti yazıyorsunuz, o zaman da bu nesnemizden faydalanmamız gerekecektir. Şimdi hemen diğer bir örneğe geçelim. İlk Olarak bir Chat'e Giriş sayfası yapalım. Tabii bu sanal bir chat. Yani sadece sayıyı kontrol etmeyi yazacağız... NotePad'i açalım ve aşağıdaki kodları chat.asp diye kaydedelim...

```
<%@ Language=VBScript Codepage="1254"%>
Chat Odamıza Gitmek İçin <A HREF="chatekatil.asp">Tıklayın</A>
<%
Response.Write "Şu anda Chat Odamızda "&application("chat")&" kullanıcı bulunmaktadır..."
%>
```

Aşağıda göreceğiniz scripti de ChateKatil.asp diye kaydedelim. Ve bir önceki sayfada yazdığımız scriptten bu sayfaya link verdiğimizi hatırlayarak, o sayfada <a href'ine hangi link ismini verdiyseniz o ismi de verebilirsiniz...

```
<%@ Language=VBScript Codepage="1254"%>
Chat Odamızdan ayrılmak İçin <A HREF="ayril.asp">Tıklayın</A>
<%
Application.Lock
'Hatırladığınız gibi, önce Application'ımızı kilitliyoruz ve sayacı bir
'arttırdıktan sonra kilidini kaldırıyoruz...
Application("chat")=Application("chat")+1
'Sayacı bir arttırıyoruz.
Application.Unlock
'Ve tekrar Application'ımızı serbest bırakıyoruz ki, diğer kullanıcılar
'giriş yaptığı zaman kilitli kalmasın....
Response.Write "Şu anda Chat Odamızda "&application("chat")&" Kişi Bulunmaktadır..."
'Daha sonra Application'ımızın tuttuğu sayısal değeri, Response
'metoduyla sayfaya basıyoruz. Ve böylece kaç kişinin
'Application'ımızı açtığını görebiliyoruz. Bir ilerki sayfada ise
'Ziyaretçimizin Application'ımızı terkettiği zaman diğer kullanıcıların
'sayacı doğru görüntüleyebilmesi için 1 değer azaltmayı göreceğiz...
%>
```

Aşağıda göreceğiniz scripti de Ayril.asp diye kaydedelim.

```
<%@ Language=VBScript Codepage="1254"%>
Chat Odamıza Tekrar Bağlanmak İçin <A HREF="Chat.asp">Tıklayın</A>
<%
Application.Lock
'Hatırladığınız gibi, önce Application'ımızı kilitliyoruz ve sayacı bir
'arttırdıktan sonra kilidini kaldırıyoruz...
Application("chat")=Application("chat") -1
'Sayacı bir azaltıyoruz.
If Application ("chat")<0 then
Application("chat")=0
'Eğer Application'ımızın değeri 0'dan küçükse Application'ımızı 'durduruyoruz.
End If
Application.Unlock
'Değilse kilidi kaldırıyoruz.
Response.Write "Şu anda Chat Odamızda "&application("chat")&" Kişi Kalmıştır..."
'Ve kalan sayıyı sayfaya Response ediyoruz...
%>
```

## SESSION

HTML ve Javascript ile biraz uğraştıysanız, bilirsiniz ki bir sayfadan ötekine değişken değeri aktarmak, imkansıza yakın derecede zordur. Değişkenlerin ömrü, fonksiyonla sınırlıdır. Bir ASP sayfasında herhangi bir değişkeni fonksiyon dışında tanımlamakla ve değer atamakla onu bütün fonksiyonlar için geçerli hale getirebiliriz. Fakat kimi zaman isteriz ki, bir fonksiyonun değeri bütün sayfalarda aynı olsun; ziyaretçinin sayfa değiştirmesi ile değişkenin değeri değişmesin. Bunu ASP'de yapmak çok kolaydır. ASP'de bu zorluğu yenebilmek için değişkenlerimizi **Session** nesnesi için oluşturabiliriz; ve bu değer ziyaretçinin oturumu boyunca devam eder; bütün ASP sayfalarındaki bütün Fonksiyonlar tarafından bilinebilir.

**Session ("Tupras") = 44500**

Session Türkçe oturum demektir. ASP Sunucusu Server'a bağlanan, yani bir talepte (Request'de( bulunan her bir ziyaretçiye ayrı bir oturum ayrı bir session açar. Tabii Cookie denetimi yoksa... Ve her Session'a bir ID verir...

Session'ın en büyük özelliği dinamik olmasıdır. Yani ziyaretçinin sitede bulunduğu her saniye Session nesnesi tarafından takibe alınır... Bu sayede eğer gerçekten profesyonel planlamalar yaparken Session nesnesinin metotlarından bir çok şekilde faydalanacağız...

Session nesnemizin en büyük özelliği tanımladığımız değişkenleri diğer geçiş sayfalarında da kullanabilmemiz için tutmasıdır... Hemen bir örnek yaparak Session nesnemize giriş yapalım... Önce Bir Form Hazırlayalım ve bu formu Session.asp diye kaydedelim.

```
<FORM NAME="formadi" ACTION="SessionDevam.asp" METHOD="POST" >
Lütfen Adınızı Giriniz:
<INPUT TYPE="text" NAME="Isim">
<BR>
Soyadınız:
<INPUT TYPE="text" NAME="Soyad">
<BR>
<INPUT TYPE="submit" NAME="cmdGonder" VALUE="Gönder">
</FORM>
```

Aşağıdaki Kodları da SessionDevam.asp adıyla kaydedelim.

```
<%@ Language=VBScript Codepage="1254"%>
<%
Session.Timeout=15
'Oturum süresi, eğer sayfaya hiç bir müdahalede bulunulmazsa 15 'dakikadır.
Session ("Isim")=Request.Form("Isim")
Session ("Soyad")=Request.Form("Soyad")
'Bir önceki sayfadan
'taşıdığımız değeri Request metoduyla Oturumda tutuyoruz.
Dim Icerik
Dim IcerikSonu
%>
<%
IcerikSonu=Session.Contents.Count
For Icerik=1 To IcerikSonu
'Session nesnemizde tuttuğumuz değerleri
'Session.Contents yardımıyla saydırıyoruz.
Response.Write (Session.Contents(Icerik) &"<br>")
'Sonucu sayfaya Response ederek yazdırıyoruz. Ve bir <br> yani
'Break Line koyarak (Alt satıra Geçmemizi sağlar.) Next yordamıyla
'sıradaki Kayıtları ekrana yazdırmasını sağlıyoruz...
Next
%>
```

Bir ASP sayfasını oturum açmadan yapılandırmak için alttaki kodu ekleyebilirsiniz.. Değerini True yaparak başka bir sayfada oturumu başlatabilirsiniz...

```
<%@ EnableSessionState=False %>
```

## Session Timeout:

Kullanıcı bir uygulamadaki bir sayfayı belirli bir süre istemez ya da yenilemezse, oturum otomatik olarak sona erer. Bu süre için varsayılan değer 20 dakikadır. Bir uygulama için varsayılan değeri, Internet Information Service çalışma ekindeki Uygulama Seçenekleri özellik sayfasında değiştirebilirsiniz.

```
<% Session.Timeout = 16 %>
```

**Session Abandon:**

Abandon yöntemi, bir Session nesnesinde saklı olan tüm nesneleri yok eder ve kaynaklarını bırakır.

Abandon yöntemini çağırmak istemiyorsanız, sunucu bu nesneleri oturumun süresi bittiğinde yok eder.

Kullanımı:

**Session.Abandon**

Abandon yöntemi çağrıldığında, geçerli Session nesnesi silinme için havuza alınır, ancak geçerli sayfadaki komut dosyası komutlarının tümü işlenmeden tam olarak silinmez.

Böylece, sonraki Web sayfaları dışında, yalnızca Abandon yönteminin çağrıldığı sayfadaki Session nesnesinde saklı olan değişkenlere erişebilirsiniz.

Örneğin, aşağıdaki komut dosyasında üçüncü satır Oğuz değerini yazar. Bu, sunucu komut dosyasının işlenmesini bitirene kadar Session nesnesinin yok edilmemesinden dolayıdır.

```
<%
Session.Abandon
Session("Isim") = "Oğuz"
Reponse.Write(Session("Isim"))
%>
```

Sonraki Web sayfalarında Isim değişkenine erişirseniz, değerinin boş olduğunu göreceksiniz.

Bunun nedeni, Isim değişkeninin, yukarıdaki işlemi bitirdiğinde önceki Session.Abandon nesnesiyle yok edilmesidir.

Server, bir oturumu kapattıktan sonra başka bir Web sayfasını açtığınızda, yeni bir Session nesnesi oluşturur. Değişkenleri ve nesneleri, bu yeni Session nesnesinde saklayabilirsiniz.

## FORM ELEMANLARINDAN DEĞER ALMA

### ActiveX Veri Erişim (ADO) Nesneleri

ASP'nin diğer CGI tekniklerine göre kolay olmasının sebebi belki de sadece veri erişimini adeta çocuk oyuncağı haline getirmesidir. ADO, gerçekte bir ASP nesnesi olmaktan çok **Server Component**'i (sunucu bileşeni) sayılır. Bu bileşene biz ASP içinden bir ActiveX nesnesi ile ulaşırız.

Veritabanı, günümüzde giderek Web Programlarının temelini oluşturuyor. Sayfaların unsurları veritabanı dosyasından alınıyor; ziyaretçilerin verdikleri bilgiler

veritabanına yazılıyor. Bu gelişimin başlıca sebebi, veritabanının site güncelleştirme işlerini kolaylaştırmasıdır. Söz gelimi bir sayfadaki seçenekleriniz, bir veritabanından alınıyorsa, bu seçenekleri alan VBScript kodu hiç değişmeden kalacak ve siz sadece veritabanı dosyanızda ilgili verinin alındığı alana yeni değerler girerek, sayfanızı sürekli güncel tutmuş olacaksınız. Bir diğer sebep ise veritabanı dosyalarının idaresinin kolay olmasıdır. Sözgelimi ziyaretçilerinizden aldığınız bilgileri daha sonra muhasebe kayıtlarınıza veya adres defterinize, müşteri kütüğüne ya da başka suretle kayda geçirmek istiyorsunuz. Ziyaretçilerimizin form yoluyla bize ilettiği bilgileri düzyazı dosyasına işlemenin yollarını Dosya sistemi Nesnesi'ni (**FileSystem**) görürken, ele aldık. Bunu yapabiliriz kolayca. Ama daha sonra düz yazı dosyasının idaresi, veritabanının idaresi kadar kolay olamaz. ASP sayfalarınız Access, Excel, Paradox, FileMakerPro, SQL Server ve Oracle veritabanlarına ve spreadsheet dosyalarına erişebilir; bu dosyalardan veri okur ve bu dosyalara veri yazabilir. Özetle, ASP programlarımızla, SQL-uyumlu veya Windows ve diğer sistemler için yazılmış ODBC (**Open Database Connectivity**/Açık Veritabanı Bağlantısı) ile uyumlu her türlü dosyaya, ADO nesnesi aracılığıyla ulaşabiliriz.

**ODBC ve OLE-DB**

Bu kitapçığın baştarafında, ASP dosyalarınızı geliştirmeye başlamadan önce bilgisayarınızda ODBC (**Open Database Connectivity**/Açık Veritabanı Bağlantısı) sürücülerinin kurulu olması gerektiğini belirtmiştik. ODBC, ADO'nun kullandığı tek sistem değildir; ve Microsoft firması, ODBC'nin yerine hızla OLE-DB adını verdiği yeni bir teknolojinin alması için yoğun çaba içinde. OLE-DB, ODBC'nin Web'de sağladığı başarının üzerine bina edilen yeni bir teknoloji. ODBC, ilişkilendirilmiş (relational) veritabanlarına erişmek üzere tasarlandığı halde OLE-DB her türlü veritabanına erişebilir. OLE-DB, ASP programlarımıza yeni nesneler kazandırabilir; kullanılmaya hazır elektronik ticaret bileşenlerini kullanmaya imkan verir. Bu konuda geniş bilgiyi, Microsoft'tan edirebilirsiniz. ASP sayfalarımızda kullanacağımız ADO nesneleri ilerde de ODBC sürücülerine erişme imkanını koruyacağı için, şimdilik sadece ODBC tekniği ile çalışmakta ve bu tekniği öğrenmekte sakınca yok. OLE-DB, ODBC'nin yerini almayacak; fakat içinde ODBC'yi de bulunduracak. Bu da şu anda oluşturacağımız ASP uygulamalarının ilerde OLE-DB tekniği ile çalışan sunucularda işleyeceği anlamına geliyor.

Şimdi ADO ile aşağıda yapacağımız küçük örnekler için bilgisayarınızda kurulu bir veritabanı programı (MS Access gibi) varsa onu kullanarak bir veritabanı dosyasında **uyeler** adıyla şu tabloyu oluşturabilirsiniz:

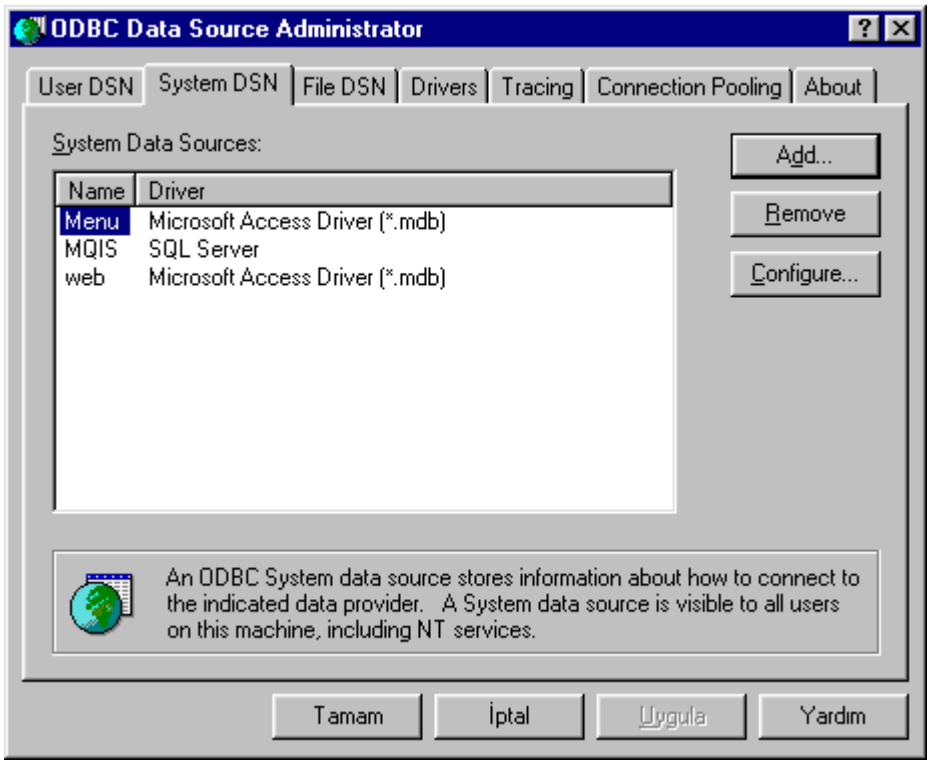
| **Alan Adı:** | **Veri türü** |
|---|---|
| uyeNo | AutoNumber (Birincil Anahtar/Primary Key) |
| uyeAdi | metin |
| uyeSoyadi | metin |

| | |
|---|---|
| email | metin |
| mesaj | memo |

Daha sonra da renkler adıyla şu tabloyu yapın:

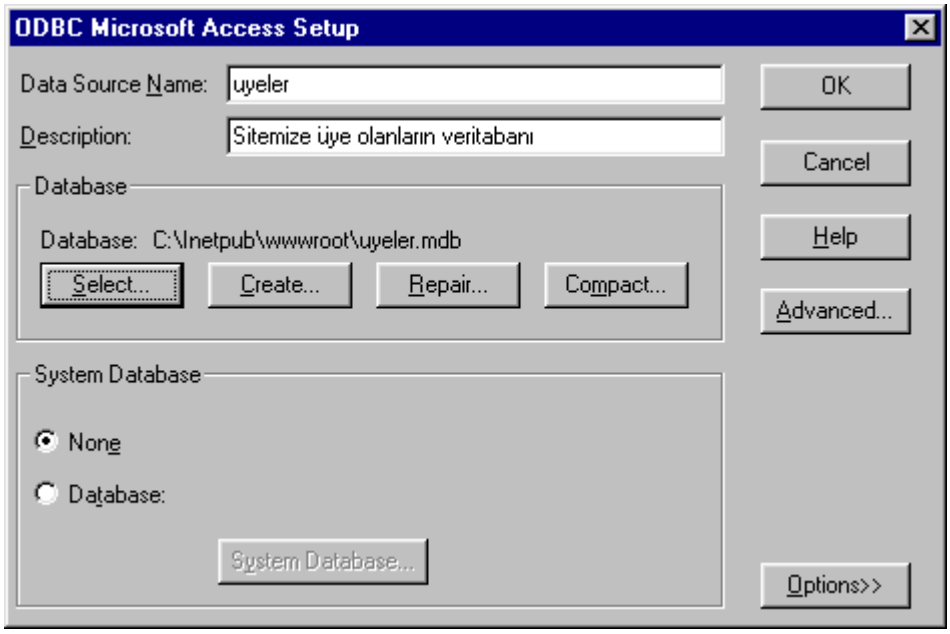
| **Alan Adı:** | **Veri türü** |
|---|---|
| renkID | AutoNumber (Birincil Anahtar/Primary Key) |
| renk | metin |

Bu tablolardan birincisine bir kaç isim ve diğer bilgileri; ikincisine ise dört-beş renk adı girin. Bu dosyayı, kişisel Web Server'ınızın kök dizinine kopyalayın. Sonra, Denetim Masası'nı açın ve adı **ODBC**, **ODBC 32 Bit**, ya da **ODBC Data Source** olan simgeyi çalıştırın; ikinci sekme olan System DSN'i tıklayın.




Açılacak kutuda **Add/Ekle** düğmesini tıklayarak, yeni veri kaynağı oluşturmak için ilk adım olan veriyi okumakta kullanacağımız sürücüyü seçebileceğimiz kutunun açılmasını sağlayın. Burada, yukarıda oluşturduğunuz veri dosyasına uygun sürücüyü seçin. Örnek **uyeler.mdb**'yi kullanıyorsanız, birinci seçenek olan **Microsoft Access Driver**'ı seçmeniz gerekir. Son düğmesini tıklayın ve **Access** dosyasının kurulumunu yapmaya başlayalım. Buradaki **Data Source Name** (DSN, Veri Kaynak Adı), biraz sonra **ADO** nesnesiyle ilgili metodları ve deyimleri yazarken kullanacağımız veri adıdır; buraya **"uyeler"** yazın; çünkü örneklerde bu veriye **"uyeler"** adıyla gönderme yapacağız. İsterseniz, **Description/Açıklama** bölümüne veritabanının niteliğini belirten bir kaç kelime yazabilirsiniz. Sonra, **Select/Seç**

düğmesini tıklayarak ve açılıcak diyalog kutusu yardımıyla veritabanı dosyasını kopyaladığınız yerde bulun; **OK/Tamam**'ı tıklayarak, veritabanı seçme işlemini tamamlayın.




**DSN** oluşturma kutularını sırasıyla **OK/Tamam** düğmelerini tıklayarak kapatın; "**uyeler**" verisi, şu andan itibaren bütün Web uygulamalarımızın hizmetine girmiş demektir. Internet sitenize koyacağınız ve veritabanına erişmesi gereken sayfalarınız için bu işlemi gerçek Internet ortamında da yapmak zorundasınız. Veritabanı dosyanızı Internet sitenizde kök dizinine veya bir diğer dizine kopyaladıktan sonra sistem yöneticisine ya elektronik mektupla, ya da evsahibi firmanın yönetim ve teknik destek yardımı sağlayan sayfasında veritabanınızın dosya adını, yolunu, ve DSN olarak kullanmak istedeğiniz ismi bildirerek, bizim burada yaptığımız işi Server yöneticisinin yapmasını sağlamamız gerekir. ADO'nun bize sağladığı imkanlardan yararlanabilmek için onun nesnelerini kullanılırız. Bu bölümde ADO'nun nesneleri ve metodlarını ele alacağız.

## Connection (Veritabanına bağlantı)

ADO'dan yararlanabilmek için kullanacağımız ilk nesne **Connection**'dır. Bu nesne ile veritabanı ile bağlantı sağlarız, (yol açarız):

```
<%
Dim Veriyolu
Set Veriyolu = Server.CreateObject("ADODB.Connection")
Veriyolu.Open "Veri_adi"
%>
```

## RECORDSET (Kayıt dizisi)

### Recordset.Open

Veritabanına dayanan Web uygulamalarımızda sorun buradaki gibi sadece veriyi okumakla bitmeyebilir; veriyi güncelleştirmek veya silmek isteyebiliriz. Bunun için doğruca ADO'nun **.Recordset** metodundan yararlanmamız gerekir. **.Recordset** metodu ne yapar? Tıpkı ekranınızdaki bir yazının içinde duran imleç (cursor) gibi hayalî bir imleci götürür verilerinizin en başına koyar. Bu hayali imleci veritabanı üzerinde dolaştırmak ve gittiği yerdeki değeri okutmak bizim işimizdir.

Bir veriye bağlantıyı kurduktan sonra kayit dizimizi **.Recordset** metodu ile sağlayacaksak, yukarıdaki örnek kodumuzu şöyle yazmak gerekir:

```
<!- - #include file="adovbs.inc" - - >
<%
Dim Veriyolu, Kayitdizisi, Sorgu
Set Veriyolu = Server.CreateObject("ADODB.Connection")
Veriyolu.Open "Veri_adi"
Set Kayitdizisi = Server.CreateObject("ADODB.Recordset")
Sorgu = "SELECT * FROM Veri_adi"
Kayitdizisi.Open Sorgu, Veriyolu, aOpenStatic
%>
```

Bu kod ile, **.Recordset** metodu son **.Open** komutu ile bizim için veri bağlantısını sağlar; verdiğimiz SQL Sorgusu icra edilir ve kayıt diziniz Kayitdizisi'ne kaydedilmeye hazır hale gelir. Şimdi imlecinizi ilerleterek, veriyi fiilen okutmanız gerekir; ki bunu yapmak için yukarıda kolayca **.Execute** metodu ile oluşturduğumuz kayıt dizisinde kullandığımız basit **.MoveNext**'ten daha çok imkana sahibiz: **MoveFirst:** Kayıt dizisinin (Recordset'in) birinci satına gider.

**MoveLast:** Kayıt dizisinin (Recordset'in) son satına gider.

**MoveNext:** Kayıt dizisinin (Recordset'in) bir sonraki satına gider.

**MovePrevious:** Kayıt dizisinin (Recordset'in) bir önceki satına gider.

**Move:** Kayıt dizisinin (Recordset'in) içinde vereceğiniz sayıya göre ilerler. Bunun için iki sayı vermeniz gerekir: başlangıç noktası ve ilerlenecek kayıt sayısı.

RecordSet nesnemize. RecordSet.Open dediğimiz zaman bu nesnemiz ne yapar?

RecordSet, ekranımızda bir yazının içinde yanıp sönen bir imleç gibi çalışır. Yani bir nevi Hayalet Cursor. Recordset'in amacı bu imleci verilerinizin en başına koymaktır.

Artık bundan sonrası bize kalan bir iştir. Bu imleci, kayıtlar arasında çalıştırmak bizim işimizdir.

Bu imleci 4 şekilde ayarlayabiliriz;

**Forward Only:** Bu imlecimizle, veritabanındaki kayıtlar arasında sadece ileri doğru ilerleyebiliriz. Geri yani yukarı gitme imkanımız yoktur. Ayrıca yeni kayıt ekleyemeyiz. Ve eğer biz açtıktan sonra, o anda başkaları tarafından değiştirilen veriler varsa, veritabanını yeniden kapatıp açana kadar göremeyiz. Eğer Adovbs.inc'le birlikte kullanıcaksak kod içindeki kullanımı adOpenForwardOnly'dir.

**Static:** Durağan anlamına gelir. ForwardOnly'e ek olarak, yukarı doğru ilerleyebilir ve yeni kayıtlar ekleyebiliriz... Adovbs.inc'le birlikte kullanıldığında kod kullanımı, adOpenStatic'tir... Eğer biz açtıktan sonra, o anda başkaları tarafından değiştirilen veriler varsa, veritabanını yeniden kapatıp açana kadar göremeyiz.

**Dynamic:** Adından da anlaşılabileceği gibi, bu cursor tipi, tam özelliklere sahip cursor tipidir. Yukarı ve aşağıya ilerleyebilir, yeni kayıt ekleyebilir ve değiştirebiliriz. Eğer biz açtıktan sonra, o anda başkaları tarafından değiştirilen veriler varsa, veritabanını yeniden kapatıp açmadan anında görebiliriz... Kod sayfasında, Adovbs.inc'le birlikte kullanıcaksak, kod içinde kullanımı adOpenDynamic'tir...

**Keyset:** Bu metodumuzla, başkaları tarafından değiştirilen kayıtları görebiliriz... Cursor'umuz yukarı ve aşağı ilerleyebilir. Adovbs.inc'le birlikte kod içinde kullanımı, adOpenKeyset'tir...

Kod içinde kullandığımız, adOpenStatic'in hemen yanındaki ise RecordSet nesnemizin kilit özelliğidir. Farklı çeşitlerde kilitler vardır. Bunlardan birkaç tanesi şöyledir.

**AdLockReadOnly:** Kayıtlı verilerimizin değiştirilmesini, üzerine yazılmasını engellememizi sağlar. Yeni kayıt girişini engeller. Sadece Listelemeye izin verir...

**AdLockOptimistic:** Bu metodumuz da kayıtlarımızın güncelleştirilmesini, yeni kayıtlar eklememizi, ve istersek silmemizi sağlar.

**AdLockPessimistic**: Bu metodumuz da eğer bir veritabanı üzerinde işlem yapıyorsak, işlemlerimiz bitene kadar, diğer işlem yapabilecek kişileri engellemek için RecordSet'e kilit koyar.

### DSN'siz Veri Bağlantısı

```
Veriyolu.Open "Veri=" & Server.MapPath("..../veriler/uyeler.mdb") & "; Driver = {Microsoft Access Driver (*.mdb);"
```

Burada, DNS'siz bağlantı için veritabanı dosyasının Server'daki göreli yerini, adını ve hangi sürücünün kullanılacağını belirtiyoruz. Aynı bağlantıyı, doğruca Jet sürücüsü için de yazabilirdik:

```
Veriyolu.Open "Veri=" & Server.MapPath("..../veriler/uyeler.mdb") & "; Provider=Microsoft.Jet.OLEDB4.0;"
```

Tabiî buradaki sorun kullandığınız veritabanı dosya türüne uygun Microsoft Jet sürücüsü seçebilmektir. Bu konuda geniş bilgi Microsoft'un Internet sitesinde bulunabilir.

## SQL

SQL parametreleri çeşitli temel komutlardan oluşur. Bunlar SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING, UPDATE, DELETE, INSERT dir. Gördüğünüz gibi ne kadar da fakir bir dil. Biz bunlardan işimize yarayacak olanları inceleyeceğiz.

### Popüler Sql Parametreleri

### Tablodan Tüm Kayıtları Tüm Alanlarıyla Seçmek

Yani tablonun tamamını seçmek için kullanacağımız SQL parametresi şudur:

```
"SELECT * FROM <tablo_adi>"
```

### Tablodan Kayıtları İstediğimiz Alanları Seçmek

```
"SELECT <alan_adi_1>, <alan_adi_2>, .. , <alan_adi_n> FROM <tablo_adi>"
```

### Tablodan Belirli Kayıtları Seçmek (Süzgeçleme)

Örneğin tablomuzda kent diye bir alanımız var. Ve mevcut kayıtlar arasından kenti "ankara" olanları seçmek istiyoruz. SQL parametresi:

```
"SELECT * FROM tablomuz WHERE kent = 'ankara' "
```

Birden fazla koşul da süzgeç de koyabiliriz. Tablomuzda yas diye bir alan olsun. Kenti ankara olan ve yaşı 30 dan büyük olanları seçmek için SQL parametresi:

```
"SELECT * FROM tablomuz WHERE kent='ankara' AND yas < 37"
```

Gördüğünüz gibi WHERE ifadesi bize kayıtları süzgeçleme imkanı veriyor. Burada = , > , < , <> gibi mukayese operatörleri kullanılıyor.

### Tablodan Kayıtları Sıralı Halde Seçmek

Örneğin tablomuz da kenti 'ankara' olanları yaş sırasında seçmek istiyoruz. Bunun için ORDER BY alan_adi yazımını kullanırız. Şunun gibi :

```
"SELECT * FROM tablomuz WHERE kent = 'ankara' ORDER BY yas"
```

Eğer azalan sırada seçeceksek ORDER BY alan_adi DESC yazımını kullanacağız.

### Tablodan Kayıt Silmek

Örneğin tablomuzdan yaşı 18 den küçük olanları silmek için SQL parametresi:

```
"DELETE FROM tablomuz WHERE yas < 18"
```

### Tabloya Kayıt Eklemek

```
SQL="INSERT INTO tblVeri (Adi,Soyadi,Telefon,Dogum_Tarihi,Mail) values
('"&txtAdi&"','"&txtSoyadi&"','"&txtTelefon&"','"&txtDogum_Tarihi&"','"&txtMail&"')"
```

### LIKE Kullanarak Kayıt Seçimi Yapmak

LIKE kullanarak tam karşılaştırma yapamadığımız alanlarla kayıt seçme imkanı elde ederiz. WHERE ifadesinde alan adından sonra kullanılan karşılaştırma operatörü yerine LIKE yazılır. Bu ifadeyle veritabanımızda arama motoru gibi bir şey bile yapabileceğiz ileride. Şimdi örnek kullanımlar görelim.

Tablomuzdan adı "A" ile başlayan müşterileri seçeceğiz.

```
"SELECT * FROM tablomuz WHERE musteri_adi LIKE 'A%' "
```

Tablomuzdan yazı alanında "asp" içeren kayıtları seçeceğiz. (bir arama motoru misali)

```
"SELECT * FROM tablomuz WHERE yazi LIKE '%asp%' "
```

Evet bu temel komutlar ile yolumuza devam edeceğiz. Artık bu laflar ile veritabanımızdan istediğimiz rafinelikte kayıt seçebiliriz, silebiliriz, arayabiliriz.

## VERİTABANI İŞLEMLERİ

Veritabanı muhabbetimiz tam gaz devam ediyor. Bu yazıda bir çok veritabanı operasyonunu birlikte göreceğiz. Kayıt ekleme, silme, güncelleme gibi. Kullanacağımız veritabanı dosyası burada (8,15 kb). Bu dosya C:\inetpub\wwwroot\db konumunda olsun. Sıra geldi script dosyalarımıza.

### KAYITLAR.ASP

Öncelikle kayıtları listeleyeceğimiz bir arabirim inşa ettik. kayitlar.asp ye göz atalım. Bu ve diğer ASP dosyaları C:\inetpub\wwwroot\ konumunda olsun.

Kod1. kayitlar.asp

```
<%
Veri_yolu = Server.MapPath("db/kisiler.mdb")
Bcumle = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ=" & Veri_yolu
Set bag = Server.Createobject("ADODB.Connection")
bag.Open (Bcumle)
Set kset = bag.execute("SELECT * FROM kisiler")
%>
<% i=1 %>
```

```
<p><a href="kayit_yeni.asp">Yeni Kayıt</a></p>
<table border=1>
<tr>
<th>#</th>
<th>Ad</th>
<th colspan=2>Eylemler</th>
</tr>
<% Do While Not kset.eof %>
<tr><td><%= i %></td>
<td><%= kset("ad") %></td>
<td><a href="kayit_duzenle.asp?id=<%= kset("id") %>">düzenle</a></td>
<td><a href="kayit_sil.asp?id=<%= kset("id") %>">sil</a></td></tr>
<% kset.movenext %>
<% i = i + 1 %>
<% Loop %>
</table>
<%
kset.Close
Set kset = Nothing
bag.Close
Set bag = Nothing
%>
```

Bu script, basit bir şekilde /db alt dizinindeki kisiler.mdb ile bağlantı kuruyor. kisiler tablosundan aldığı tüm kayıtların "ad" alanındaki değerlerini yazdırıyor. Ve her bir kayıt için düzenleme ve silme linkleri oluşturuyor. Linkler oluşturulurken sorgu stringi içinde eylemi gerçekleştirecek script dosyasına "id" anahtarıyla kaydın "id" alanındaki değeri gönderiliyor. Tabi bunlar kayıtseti sonunda duracak bir döngü içerisinde yapılıyor. Verilerin tablo hücrelerine döngüyle döküldüğüne dikkat ediniz.

## KAYIT_DUZENLE.ASP ve KAYIT_GUNCELLE.ASP

Kayıtların listelendiği kayitlar.asp de her kaydın yanında bir düzenle bağlantısı var. Bu bağlantı ile duzenle.asp dosyasına işaret ediliyor. Bir de sorgu cümlesi konuluyor yanına: "duzenle.asp?id=XX" Buradaki XX yerine geçerli kaydın id alanındaki değeri yazılıyor. Şimdi kayıt_duzenle.asp ye bakalım.

Kod2. kayit_duzenle.asp

```
<%
id = Request.Querystring("id")
If Not Isnumeric(id) Or Len(id)=0 Then
mesaj "Yanlış Sorgu Cümlesi"
End If
'--------------------------------------------------------------------------------
Veri_yolu = Server.MapPath("db/kisiler.mdb")
Bcumle = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ=" & Veri_yolu
Set bag = Server.Createobject("ADODB.Connection")
bag.Open (Bcumle)
Set kset = bag.execute("SELECT * FROM kisiler where id =" & id)
If kset.eof Then
mesaj "Olmayan Kayıt İstendi"
End If
%>
<a href="kayitlar.asp">Kayıtlar</a>
<form method=post action="kayit_guncelle.asp">
<table border=1>
```

```
<tr>
<td>Ad</td>
<td><input type="text" name="ad" value="<%= kset("ad") %>"></td>
</tr>
<tr>
<td>Teleon</td>
<td><input type="text" name="telefon" value="<%= kset("telefon") %>"></td>
</tr>
<tr>
<td>Email</td>
<td><input type="text" name="email" value="<%= kset("email") %>"></td>
</tr>
<tr>
<td>ICQ</td>
<td><input type="text" name="icq" value="<%= kset("icq") %>"></td>
</tr>
<tr>
<td>Adres</td>
<td><input type="text" name="adres" value="<%= kset("adres") %>"></td>
</tr>
<tr>
<td>Doğum Günü</td>
<td><input type="text" name="dgunu" value="<%= kset("dogum_gunu") %>"></td>
</tr>
<tr>
<td></td>
<td><input type="submit" value="güncelle"></td></tr>
</table>
<input type="hidden" name="id" value="<%= kset("id") %>">
</form>
<%
kset.Close
Set kset = Nothing
bag.Close
Set bag = Nothing
%>
<%'-------------------------%>
<% Sub mesaj(msg) %>
<p><%= msg %></p>
<% response.end %>
<% End Sub %>
<%'-------------------------%>
```

Aslında bu scriptin de öncekinden pek farkı yok. sadece sorgu cümlesinden "id" anahtarındaki değeri alıyor. Ve bu id değerini kullanarak tablodan tek kayıt seçiyor. ("SELECT * FROM kisiler where id =" & id). Seçtiği kayda ait değerleri ise form input alanlarına döküyor. Formun action özelliğine ise kayit_guncelle.asp yazılmış. Güncelleme işini bu dosyaya havale ediyoruz.

Kod3. kayit_guncelle.asp

```
<%
ad = Request.Form("ad")
id = Request.Form("id")
telefon = Request.Form("telefon")
dgunu = Request.Form("dgunu")
icq = Request.Form("icq")
email = Request.Form("email")
adres= Request.Form("adres")
```

```
'-------------------------------------------------------------------------------
Veri_yolu = Server.MapPath("db/kisiler.mdb")
Bcumle = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ=" & Veri_yolu
Set bag = Server.Createobject("ADODB.Connection")
bag.Open (Bcumle)
Set kset = Server.Createobject("ADODB.Recordset")
sql = "SELECT * FROM kisiler WHERE id = " & id
kset.open sql, bag, 1, 3
kset("ad") = ad
kset("telefon") = telefon
kset("email") = email
kset("adres") = adres
kset("icq") = icq
kset("dogum_gunu") = dgunu
kset.update
kset.Close
Set kset = Nothing
bag.Close
Set bag = Nothing
Response.Write "<p>Kayıt Yapıldı.. Başka bir arzunuz? "
Response.Write "<p><a href=""kayitlar.asp"">Kayıtlar</a>"
%>
```

## KAYIT_YENI.ASP

Yeni Kayıt linkinde işaret edilen kayit_yeni.asp veritabanına bağlanıp ilgili tablodan oluşturduğu kayıt seti içine yeni bir kayıt ekliyor. Kayıtsetini güncelliyor, "id" alanındaki değeri bir değişkende saklıyor. nesneleri kapatıyor. Ve kayit_duzenle.asp dosyasına "id" anahtarıyla yeni eklenen kaydın "id" alanındaki değeri gönderiyor. Yani bu sayfa sadece işlem yapıyor. Kullanıcıya gözükmeden işlemi tamamlayıp düzenleme sayfasına yöneliyor.

Kod4. kayit_yeni.asp

```
<%
response.buffer=true
Veri_yolu = Server.MapPath("db/kisiler.mdb")
Bcumle = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ=" & Veri_yolu
Set bag = Server.Createobject("ADODB.Connection")
bag.Open (Bcumle)
Set kset = Server.Createobject("ADODB.Recordset")
sql = "SELECT * FROM kisiler"
kset.open sql, bag, 1, 3
kset.addnew
kset.update
id = kset("id")
kset.Close
Set kset = Nothing
bag.Close
Set bag = Nothing
Response.Redirect "kayit_duzenle.asp?id=" & id
%>
```

**KAYIT_SIL.ASP**

Sorgu cümlesindeki "id" anahtarından alınan id değeri kullanılarak şu SQL ifadesi oluşturuluyor.

**DELETE FROM kisiler WHERE id = " & id** .

bu ifade bağlantı tarafından yürütülüyor (execute). Sonuçta id si alınan kayıt tablodan silinmiş oluyor. Kullanıcı da geldiği sayafaya yönleniyor.

**Kod5. kayit_sil.asp**

```
<%
response.buffer = True
id = Request.Querystring("id")
'-------------------------------------------------------------------------------
Veri_yolu = Server.MapPath("db/kisiler.mdb")
Bcumle = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ=" & Veri_yolu
Set bag = Server.Createobject("ADODB.Connection")
bag.Open (Bcumle)
sql = "DELETE FROM kisiler WHERE id = " & id
Set kset = bag.execute(sql)
Set kset = Nothing
bag.Close
Set bag = Nothing
Response.Redirect (Request.ServerVariables("HTTP_REFERER"))
%>
```