ASP/KITAP 1

Önsöz

"Merhaba: Ben bilgisayar hastası—Kroninukus computerium—üniversiteyi yeni bitirmiş bir gencim. Byte dergisiyle verilen "Internet Tasarım Rehberi" kitapçığını okuyarak HTML'i öğrenmeye çalışıyorum ve bu konuda çok istekliyim. Bundan iki ay kadar önce bir shareware program kullanarak bir site hazırladım. Fakat bu programı kullanmak için hiç bir HTML etiketini öğrenmek gerekmiyordu. Oysa ben kendi çabalarımla bu işi halletmeyi ve hatta daha da ileri götürmeyi düşünüyorum. Sizden ricam bana nereye gitmem gerektiğini bildirmeniz, ve bunun için gerekli bilgi ve belgeleri sağlamanız, bizlere yardım etmenizdir. Gayretleriniz için çok teşekkür ediyor, çalışmalarınızda başarılar diliyorum."

Merhaba: çok değil, bir buçuk yıl önce yazdığınız bu mektuba, o günden beri sürekli karşılık verdim. Önce Byte, ardından PC World ve şimdi de PC Life dergisinin eki kitapçıklarla, bu arzunuzu yerine getirdik. Size bir harita sunduk; bu haritaya göre ilerlediniz. HTML'den sonra site yeri edinme yollarıyla işe başladık. Ardından HTML'in durağan kalıbını yırtmak için JavaScript öğrendik. Sonra, sitemizi ziyaret edenlerle etkileşmeli sayfalar yapmak, onlardan Form yoluyla bilgiler almak, konuk defterleri yapmak için CGI imkanlarını kullanmak amacıyla Perl'e sıra geldi. Şimdi yolun sonundayız. ASP tekniği ile sitelerimizi veritabanına dayalı hale getireceğiz; Server nesnelerini kullanarak gerçek dinamizme kavuşturacağız. Bunu yaparken Visual Basic Scripting Edition (veya kısa adıyla VBScript) ile tanışacağız.

Gerçi bu ve gelecek ay sunacağımız ASP Uygulamaları kitapçığıyla, birlikte yürümek üzere çizdiğimiz eğitim döneminin sonuna gelmiş oluyoruz; ama sizin önünüzde uzun bir yol var. Bu yol, Internet denen yeni iletişim aracını kullanarak, toplumumuza yeni bir soluk kazandırma yoludur. Bu yolda yalnız ilerleyeceksiniz. Bilgisayarınızın ekranının aydınlattığı

yolda, klavye ile baş başasınız. Kalbiniz ve onu dolduran Bilişim Teknolojisi aşkı size rehber olacak. Daha öğreneceğiniz çok şey var hayatta. Hergün yeni teknolojilerin çıktığı bir alan seçtiğinizin farkındasınız. Sakın kafanız karışmasın: Siz ustasınız; teknoloji, adı üstünde, araçtan ibaret. Hergün yenisi ile karşılaştığınız teknolojiler sizi korkutmamalı. Sırasıyla hepsini başaracaksınız. Öğrendikçe, çok değil bir ay önce sizi korkutan bir dilin, bir tekniğin ne kadar basit olduğunu göreceksiniz. Asla "Ben bunların hepsinin altından nasıl kalkarım?" demeyeceksiniz. Sırasıyla herşeyi öğrenebilirsiniz. Daima şunu tekrar edin: "Öğrenen nasıl öğrendi ise, ben de öğrenirim!" Bu yolda aklınız ve kalbiniz size rehber olacaktır. Yeter ki onlara kulak verin.

ASP veya Active Server Pages (Etkin Sunucu Sayfaları) tekniği, sayfalarınızı canlandıracak bir tekniktir. Bu teknik, bir kaç sayfa sonra göreceksiniz ki, sil baştan bir bilgisayar programlama dili öğrenmeye gerek olmadan uygulanabilir. Bu kitapçığı yazarken sizinle yolun başından beri beraber olduğumuzu, yani HTML bildiğinizi varsayıyorum. Ayrıca Web'in nasıl çalıştığını, Server (Sunucu) ve Client (İstemci) ilişkisinin nasıl yürüdüğünü de biliyor olmalısınız.

ASP teknolojisinden yararlanmak istediğinizde Web sitesine evsahipliği yapan bilgisayarda çalışmakta olan Web Server'ın ASP teknolojisini tanıması ve sitenize bu hizmeti vermesi gerekir. ASP, bir zamanlar sadece Microsoft'un NT ve daha sonra Windows 2000 işletim sistemine dayanan MS-IIS (Internet Information Server) programında işleyebilirdi. Fakat şimdi artık NT-tabanlı diğer Web Server programları gibi Unix-tabanlı Web Server programları da ASP anlar ve işletir hale gelmiş bulunuyor. Fakat ASP sayfalarınızı gerçek Internet ortamında ziyaretçilerinizin hizmetine sunmadan önce, kendi bilgisayarınızda sınamanız gerekir. Bunun için bilgisayarınızın işletim sistemi Windows 95/98 ise Kişisel Web Server (PWS), NT 4.0 ise IIS kurulmuş olmalıdır. Sisteminiz Windows 2000 ise, IIS 5.0 kendiliğinden kurulmuş demektir.

Şimdi biraz ASP'den söz edelim. Bu kısaltmayı, "a..se..pe" veya "ey-es-pi" diye harf-harf okuyanlar olduğu gibi, "eyspi" diye bir kelime halinde okuyanlar da vardır; ve bana sorarsanız, hepsi de doğrudur. Internet programcıları, bütün görevi bir sabit diskteki HTML dosyalarını alıp, ziyaretçinin Browser'ına göndermekten ibaret olan Web Server programını yeniden tasarlamaya başladıklarında, Server'ın sadece durağan sayfaları göndermesi yerine, ziyaretçiden de veri kabul etmesinin uygun olacağını düşündüler. Bu amaçla Internet istemcisi ile sunucusunun buluştuğu noktada, yani Common Gateway Interface (Ortak Geçit Arayüzü) katmanında Web Server programının, istemci programdan (browser) kendisine bilgi ve komutlar gönderilmesini sağladılar. Örneğin bir Form'daki bilgilerin alınıp, bir dosyaya kaydedilmesi için vereceğimiz komut, sitemizin bulunduğu bilgisayarın işletim sisteminde CGI katmanı tarafından icra ettirilir. Bu manada CGI, Web Server'ın Internet ziyaretçisinin Browser'ın gelen bilgi ve komutları işlediği veya kendi işletim sistemine aktardığı noktadır. Bu gelişmenin sonucu olarak ortaya CGI programı dediğimiz şeyler çıktı. Perl, C/C++, Delphi, Visual Basic ile yazılan bu programlar Web Server tarafından çalıştırılır, ve vereceği komutlar işletim sistemine iletilir.

CGI programları ile çok şey yapılabilir; fakat Web Server'a aynı anda birden fazla kişi erişir ve aynı CGI programını çalıştırırsa, (yani aynı anda aynı formun "Gönder" düğmesini tıklarlarsa), CGI programının birden fazla "kopyası" çalışmaya başlar. Aynı anda aynı forma ulaşan kişi sayısı 4-5 ise bu belki sorun oluşturmaz; ama bu sayı arttıkça Web Server da adeta yerlerde sürünmeye başlar! Özetle CGI programları Web Server'ı yavaşlatır.

Microsoft programcıları biraz geriden gelmekle birlikte Server işine el attıklarında, istemci ile sunucunun etkileşmesini, bütün sistemi yavaşlatan "haricî" programlar yerine, işletim sisteminin bir işlevi haline getirebileceklerini düşündüler. Bunun yolu ise işletim sisteminin Uygulama Programı Arayüzü (API, Application Programming Interface) denen

unsurlarını kullanmaktı. Nitekim Microsoft, oturup bir dizi Internet Server API (veya kısaca ISAPI) tasarladı. ISAPI, tıpkı CGI gibi, Web Server programının bulunduğu bilgisayardaki diğer programlarla alışverişini sağlar. Ne var ki ISAPI'den yararlanan programlar üreterek bunları Web Server'ın emrine vermek oldukça pahalı bir yol. Başka bir deyişle, bir formda Gönder düğmesinin çalışabilmesi için sözgelimi Excel ayarında bir program yazdırtmak pek akıl kârı olmasa gerek.

Bu noktada çeşitli firmalar ISAPI benzeri "yorumlayıcılar" geliştirerek, düz yazı bir Script yazmakla ve bu Script'in içindeki komutları Server programına icra ettirmekle, Web tasarımcısının hayatını çok kolaylaştırabileceklerini gördüler. Microsoft'un bu noktadaki çözümü ASP oldu. Yani bir bakıma "Server'a Aktif Sayfalar sunma" teknolojisi! NetObjects firması kendi Server Extension'larını geliştirdi. Linux-Unix dünyasında bir başka grup PHP dilinden yararlanarak PHP Sayfaları sistemini geliştirdi. Bunlardan sadece MS'un ASP teknolojisi birden fazla dil ile Script kabul edebilen teknoloji olarak, diğerlerinden ayrıldı. Bu noktaya birazdan geleceğiz; ama önce "ASP nasıl çalışır?" sorusunun üzerinde duralım.

Ama önce biraz başa dönelim: Siz Web ziyaretçisi olarak Browser'ınızın URL hanesine bir HTML sayfasının Internet yolunu ve adını yazıp (örneğin, http://www.pclife.com.tr/index.htm) Git düğmesini tıkladığınızda veya klavyede Enter/Return tuşuna bastığınız zaman ne olur? Web Server, Internet'in bulutları arasından geçip kendisine gelen bu istem üzerine, index.htm'i bulur ve yine aynı bulutların arasından sizin blgisayarınıza kadar gönderir. ASP teknoloji ile üreteceğiniz sayfanın adının uzatması ise .asp şeklinde olur. Siz bu kez URL hanesine bu dosyanın adını (örneğin http://www.pclife.com.tr/index.asp) yazarsınız. Bu durumda Web Server, bu sayfayı alıp, doğruca size göndermez: önce içindeki kodları icra eder. Sonra bu kodları ayıklar ve geriye saf ve temiz HTML kalır. Ve bu sayfa, bizim Browser'ımıza gönderilir.

Yukarıdaki son üç cümleyi lütfen yeniden okur musunuz? Nelere dikkat ettiniz:

1. ASP sayfasının içinde kod vardır.

2. ASP sayfasının kodları Web Server tarafından icra edilir.

3. ASP sayfası, Browser'a salt HTML olarak gelir.

Şimdi burada ASP ile Javascript'in farkını görebiliyor musunuz? Javascript, HTML'in içine konuluyor; ama Server'da değil, Browser'da çalışıyor. Peki CGI/Perl programı ile ASP'nin farkını görebiliyor musunuz? HTML sayfasını Perl ile yazdığınız program üretiyor (print "<HTML>\n"; kodunu hatırlıyor musunuz?); oysa ASP'de ziyaretçiye gidecek HTML'i Server ve tasarımcı birlikte üretiyorsunuz. Bu bakımdan, Javascript "istemci-tarafında" çalışırken, ASP, CGI programları gibi, "sunucu-tarafında" çalışır. Bunun sonucu ise ASP ile yazacağınız sayfaların, ziyaretçinin bilgisayarında kurulu Browser programı ne olursa olsun (Netscape Navigator, Opera, Amaya, Internet Explorer), mutlaka görüntülenecek olmasıdır. ASP sayfalarınıza koyacağınız VBScript kodları Server'da icra edileceği için, ziyaretçinin Browser'ının örneğin VBScript'i tanımayan Netscape olması, sizi hiç etkilemeyecek. Bununla birlikte ASP yoluyla üreteceğiniz sayfalarda yer alacak dinamik HTML kodlarının, her Browser'da mutlaka arzu ettiğiniz gibi yorumlanmayacağını unutmamalısınız.

Peki, ASP'nin içinde HTML ile kod yan yanadır, dedik. Nedir bu kodlar? Hangi dille yazılır? ASP yoluyla Web Server'a vereceğiniz komutları içeren bu kodlar, tabir yerinde ise, "arzu ettiğiniz bir Script dili ile" yazılabilir. Bu VBScript olabilir; Javascript veya Jscript olabilir; hatta, MS'un yeni teknolojisi olan Windows Scripting Host ile uyumlu olmak koşuluyla, kendi Script dilinizi icad edip, bunu Web Server'a öğretecek plug-in (ek program) geliştirebilirsiniz. İlk dönemlerinde ASP kodlarını VBScript ile yazmak adeta şarttı. Giderek daha çok kaynakta ASP'nin Javascript ile yazıldığını görüyorum. Fakat BT gibi sürekli gelişen ve yenilenen bir alanda, Web tasarımcısın yeni bir dille tanışmasına imkan sağlamak için bu kitapçıkta kodlarımızı VBScript ile yazacağız; dolayısıyla kitapçığın büyükçe bir bölümü VBScript'e ayrılacak.

Şimdi kolları sıvayıp; işe başlıyoruz. Bir dizinin sonu olan bu kitapçığı, Kroninukus computerium'a adıyorum. Sadece BT alanında değil, tüm yaşamında yolun açık ve aydınlık olsun. Bu eğitime birlikte başladığımızda "bilgisayar hastası" idin; eğer şimdi "usta bir bilgisayar hastası" olmanı sağlayabildimse, ben görevimi yerine getirmiş sayılırım. Bundan sonra sıra sende. Şimdi yardım edecek olan sensin. Bu yardımı kimseden esirgeme. Kim olduğunu ve işlevini daima hatırla. Kalbinde duyduğun çağrıya her zaman kulak ver.

Örnek Kodlar

İki kitapçıktan oluşan ASP rehberinin bütün örnek kodları ve veritabanı dosyaları, http://www.pclife.com.tr/..... adresinden edinilebilir. Ancak ASP öğrenirken örnek kodları kendiniz yazmalısınız. Kodlamanın incelikleri ancak kodları siz yazarsanız öğrenilebilir. Bu örneklerden sadece kendi yazdığınız kodlar hata verdiği ve hatayı bulmaktan güçlük çektiğiniz zaman, karşılaştırma amacıyla yararlanmalısınız.

Teşekkürler

Bu kitapçıktaki kodların değişik ortamlarda sınanması için için bana yardımcı olan dilini ve yazımını denetleyen dostlarım Mustafa Durcan, Osman Nuri Hömek ve Armağan Ergon'a teşekkür ederim. Varolan hatalar elbette bana aittir. Sevgili arkadaşım Mustafa Durcan'ın bütün dizinin gelişimine katkısını daima şükranla hatırlayacağım.

ASP'ye Giriş

Bir HTML sayfayı sınamak için sadece Browser programına ihtiyacınız var; bir HTML belgesini iki kere tıkladığınızda varsayılan Browser açılacak ve bu sayfayı yorumlayarak, görüntüleyecektir. Fakat uzatması .asp olan bir dosyayı iki kere tıkladığınızda ya Windows size bu dosyayı hangi programla açmak istediğinizi soracak; ya da MS Visual Studio veya Adobe Photoshop çalışacaktır. Bilgisayarınızda MS Visual Studio programı kurulu ise, onun açılması normal: çünkü MS'un ASP dahil hemen hemen bütün Web tasarım araçları Visual Studio'nun üyesi. Photoshop ise renk paletlerini belirlemek üzere kullandığı dosyalara .asp uzatmasını veriyor. Bilgisayarınızda MS Visual Studio kurulu değilse, fakat Photoshop varsa, ASP dosyalarınızın simgesi Photoshop'a ait olacaktır.

MS Visual Studio'dan yararlanmıyorsanız, ASP dosyalarınızı, (herhangi bir klasörün Görünüm menüsünden Klasör Seçenekleri maddesini seçerek ve üçüncü sekme olan Dosya Türlerini tıkladıktan sonra ASP satırını işaretleyerek ve Düzenle düğmesine basarak) Not Defteri ile ilişkilendirmek en doğru yoldur. Bu, bu kitapçıktaki örnek ASP dosyalarını oluşturmak, düzenlemek, değiştirmek için kullanacağımız başlıca programın da herhangi bir düzyazı programı olduğu anlamına gelir. ASP üretim işini herhangi bir ortamda yapabilirsiniz. Bunun için bilgisayarınızda mutlaka Windows kurulu olması gerekmez. Fakat ASP dosyalarının çalışıp çalışmadığını sınamak için sayfalarınızı gerçek Web Server'a göndermeden önce kendi bilgisayarınızda çalıştırmanız gerekir. Bunu Kişisel Web Server (PWS) veya IIS ile yapacağız.

Kişisel Web Server Kuralım

Bilgisayarınız Windows 95, 98, NT4 WorkStation veya NT4 Server ile çalışıyorsa, sisteminize bir Web Server programını siz kurmak zorundasınız. Windows 2000 Professional veya Windows 2000 Server ise Kişisel Web Server programını kendiliğinden kurar. Windows 98'e bir kişisel Web Server kurmaya geçmeden önce bilgisayarımıza bir kimlik vermemiz gerekir: Bilgisayarım/Denetim Masası/Ağ'ı tıklayarak açacağınız diyalog kutusunda ikinci sekme olan Tanımlama'yı açın ve "Bilgisayar adı" kutusuna istediğiniz adı yazın. Bilgisayarın ağ ortamında olması gerekmez.

Windows 98'e Kişisel Web Server kurmak için iki yol izleyebilirsiniz. Windows 98 CD-ROM'unda Add-ons klasöründeki PWS dizininde Kur.exe'yi tıklayın veya Windows NT Option Pack CD-ROM'unda Default.htm'i açın. Bilgisayarınızın Windows 98 ile çalıştığını algılayacak olan program size Personal (kişisel) Web Server (PWS) kurmayı önerecektir. Kişisel Web Server'ı kurarken her iki durumda da ikinci diyalog kutusunda Minimum/En az veya Typical/Tipik seçeneğini değil, Custom/Özel'i seçin ve açılacak yeni diyalog kutusunda Microsoft Data Access Components (MS Veri Erişim Bileşenleri) satırına işaret koyarak, Alt Bileşenleri Göster düğmesini tıklayın. Açılacak seçme kutusunda ise ADO Documentation satırına işaret koyun. Bu belgelerden daha sonra veri-yönlendirmeli Web Uygulaması yaparken yararlanacağız.

<asp0001.tif>

Kişisel Web Server kurulduktan sonra bilgisayarı yeniden başlatmak gerekir.

//////////////////////KUTU//////////////////

PWS Kurulurken Hata Verirse

Windows 98'e PWS kurarken, programın Microsoft Transaction Server bölümüne ilişkin sistem kayıtları yapılırken, iki hata mesajı ile karşılaşabilirsiniz (0x80004005 ve 0xfee662). Bu, orijinal Windows 98 CD-ROM'undaki PWS Kur programının, Windows Registry dosyasının büyük olması halinde hata vermesinden kaynaklanıyor. Böyle bir durumla karşılaşırsanız, Bilgisayarım/Denetim Masası/Program Ekle Kaldır aracılığıyla, Personel Web Server'ı kaldırın. Bilgisayar kapanıp açıldıktan sonra, Windows 98 CD-ROM'unda Add-ons/PWS dizinindeki bütün dosyaları, sabit diskinizde Temp dizinine kopyalayın. Sonra http://support.microsoft.com/support/kb/articles/q246/0/81.asp adresinde "Download Mstsetup.dll" satırını tıklayın. Mssetup.exe adlı bir dosya bilgisayarınıza indirilince; bu dosyayı iki kere tıklayın ve dosyanın genişletileceği yer olarak C:\Temp'i gösterin; program Mstsetup.dll dosyasının değiştirilmesini isteyip istemediğinizi sorduğu zaman "Tamam"ı tıklayın. Şimdi, C:\Temp'deki Kur.exe (Windows CD-ROM'unuz İngilizce ise Setup.exe) programını iki kere tıklayın. PWS şimdi hatasız kurulacaktır.

//////////////////KUTU BİTTİ////////////////////////////

Windows NT4.0 Workstation veya Server'a IIS4.0 kurmak için Option Pack CD-ROM'undaki default.htm'i çalıştırmanız ve açılacak Browser penceresinde IIS'i kurma seçeneğini tıklamanız yeter. Burada da ADO Documentation'ı sabit diskinize aktarabilmek için gerekli seçenekği işaretleyin.

Windows 98'e Kişisel Web Server kurulduğunda Masaüstü'nde Yayımla (Publish) adlı bir simge belirecektir. NT sistemlerinde ise Başlat menüsünde Programlar bölümüne IIS Manager satırı eklenir. Bu yollardan biriyle PWS veya IIS'i çalıştırın.

Kişisel Web Server'da Personel Web Server Manager (Yönetici) kutusu açıldığında soldaki araç çubuğunda Yönetici'nin çeşitli bölümlerine gitmeniz için gereken gezinme simgelerini göreceksiniz. Şimdi, açılan ana pencerede iki unsura dikkat edin:

<asp002.tif>

1. Kişisel Web Server'ınızın adı. Bilgisayarınızın adı buraya Server adı olarak yazılmış olmalı. Biraz sonra, Internet'e koymadan önce sınayacağımız ASP sayfalarını çağırırken, Browser'ın adres kutusuna burada gördüğümüz adı yazacağız.

2. Kişisel Web Server'ın bilgisayarımızda sabit diskteki gerçek adresi. Bu, sizin Kişisel Web Server'ınızın kök (root) dizinidir. Bu genellikle C:\inetpub\wwwroot klasörüdür. Kişisel Web sitesi yaparsanız, sitenin gerektirdiği bütün dizinleriniz ve dosyalarınız burada gördüğünüz dizinin içinde olmalıdır. Yapacağımız ASP dosyalarını işte bu dizinin içine koyacağız.

Bunları bir kenara not ettikten sonra, soldaki araç çubuğunda Gelişmiş simgesini tıklayın; ortadaki pencerede sanal dizinlerinizi görüyorsunuz. Bunlardan Home'u seçin ve sağdaki "Özellikleri düzenle" düğmesini tıklayın.

<asp003.tif>

Ana dizinin okuma, yürütme ve makro erişim haklarının işaretli olmasına dikkat edin. İlerde kendinize Kişisel Web Server'ınızın kök dizininde yeni bir dizin oluşturursanız (örneğin "resimler" gibi) ve içine sitenizle ilgili dosyalar koyarsanız, Gelişmiş penceresinde Ekle düğmesini tıklayarak bu gerçek dizini de sitenin sanal dizinlerinden biri haline getirmeniz gerekir. Gerçek dizinin adı XYZ bile olsa, sanal dizin haline getirirken istediğiniz sanal adı verebilirsiniz. Ama unutmayın, Browser'ın adres hanesine gerçek dizin adını değil sanal dizin adını yazmanız gerekir.

Bu işlemleri IIS'te değişik araçlar ve diyalog kutularıyla, fakat temel ilkeler itibariyle aynı şekilde yapabilirsiniz. NT4 sistemlerine IIS'i kurmadan önce, Service Pack 3'ü

uygulayın; Internet Explorer 5'i kurun. Elinizde varsa Service Pack 4, 5 veya 6'yı en son uygulayın.

Bu noktada, ASP sayfalarınızı sınayacağınız bilgisayarda Microsoft Internet Explorer programının kurulu bulunmasının şart olmadığını hatırlatalım. ASP sayfalarınızı Netscape ile de sınayabilirsiniz.

Bir Örnek Yapalım

Şimdi ilk ASP sayfamızı yapalım ve bununla kişisel Web Server programının çalışıp çalışmadığını sınayalım. Buraya kadar anlamış olduğunuz gibi ASP sayfası da HTML gibi düz yazı dosyasıdır; dolayısıyla beğendiğiniz bir düz yazı programı ile ASP yazabilirsiniz. Şimdi açın en beğendiğiniz düz yazı programını ve başlayın yazmaya. Eğer kelime-işlemci kolayınıza gidiyorsa, dosyayı ASCII veya ANSI biçiminde kaydetmeyi unutmayın (10'ncu satırın sonunda nokta olduğuna dikkat edin):

```
<HTML>
<HEAD>
<TITLE>ASP ILE ILK SAYFA</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>
<H1><CENTER>Merhaba Dünya!</H1>
<H2>Bugün:
<% Response.Write(Date) %>.
</CENTER
</H2>
</BODY>
</HTML>
```

Bu dosyayı merhaba.asp adıyla kaydedin ve kendi bilgisayarınızda kişisel Web Server'da veya ISS'te sınayacaksanız, bu programları çalıştırın. Browser'ınızın URL hanesine, kişisel Server'ınızın adıyla birlikte dosyanın adını yazın; ve ilk ASP programınız dünyaya Merhaba desin.

<asp0004.tif>

////////KUTU//////////////////

Internet'te ASP

ASP teknolojisi ile oluşturduğunuz sayfaları, yani içinde ASP teknolojisinin gerektirdiği kodlar bulunan HTML dosyalarını, .asp uzatmasıyla kaydederiz. Bu dosyalar talep edildiğinde, Web Server programı, eğer ASP teknolojisini tanıyorsa, talep edilenin ASP sayfası olduğunu dosya adının uzatmasından anlar; ve bu sayfayı ziyaretçiye yollamadan önce kendisi işleme tabi tutar. ASP sayfamızdaki kodlar büyük bir ihtimalle bir takım dinamik işler yapacak, örneğin hesaplamalar, araştırmalar, veri tabanından veri çekme, dosya sisteminden dosya isteme gibi görevlerini yerine getirecek, ve ortaya çıkacak olan HTML dosyasını ziyaretçiye gönderecektir. Diyelim ki bu kitapçıktaki örnekleri yaptınız ve ücretsiz Site yeri edinmiş olduğunuz Hosting firmasının bilgisayarındaki sitenize yerleştirdiniz. Sonra Browser'ınızın URL hanesine bu ASP dosyasının adını yazdınız ve sayfayı talep ettiniz. Karşınıza, ya ASP sayfasının içindeki kodları görüntüleyen veya sadece sayfanın HTML unsurlarına yer veren bir sayfa geldi! Bu Server'ınızın ASP anlamadığını gösterir. ASP sayfalarınızı ASP-uyumlu Web sunucularında çalıştırabilirsiniz. Sitenizde ASP sayfaları bulunacaksa ve ücretli evsahibi firmalardan site alanı edineceksenız, sunucunun ASP-uyumlu olmasına dikkat edin.

//////////////KUTU BITTI////////

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.

<asp0005.tif>

ASP.DLL, kendisine bir .asp dosyasının geldiğini görünce, hemen ASP'lerin Anayasası olan global.asp'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.

Biliyorsunuz, mevcut sürümleri itibariyle Browser programları içinde Netscape VBScript dilini anlamaz. ASP sayfalarımızda istediğimiz Script dilini kullanabiliriz; VBScript de kullanabiliriz. Netscape'in VBScript anlamamasıyla, ASP sayfalarımızda VBScript kullanmamızın bir ilgisi yoktur; çünkü ASP sayfasının ortaya çıkartacağı HTML kodunda ASP dolasıyla VBScript bulunmayacaktır; dolayısıyla ASP sayfalarınız, Netscape tarafından da anlaşılıp, görüntülenebilecektir. Tabiî ortaya çıkacak HTML sayfasının Netscape tarafından arzu ettiğiniz gibi görüntülenebilmesi için Netscape'in anlayabildiği dinamik HTML unsurlarına yer verme zorunluğunuz hâlâ devam ediyor.

ASP tekniğinin nasıl çalıştığını anladığımıza göre şimdi biraz terminoloji serpiştirelim ki, Webmaster meslektaşlarınızla biraraya geldiğinizde, gerçekten ASP ile mesela Javascript arasındaki farkı bildiğiniz belli olsun. ASP bir Server-Side Script teknolojisidir. Internet'te istemci ile sunucu arasındaki çizginin sunucu tarafına Server-Side (Sunucu Tarafı), istemci tarafına da Client-Side (evet, doğru tahmin ettiniz: İstemci Tarafı) denir. Server-Side'da çalışan Script programları da "Server-Side Script" (Sunucu Tarafı Script'i) adını alır. Dolayısıyla şöyle bir sayfa yaparsanız (merhaba.htm), bu sayfada Client-Side Script teknolojisi kullanmış olursunuz:

```
<HTML>
<HEAD>
```

```
<TITLE>JavaScript ile Tarih</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>
<H1>Merhaba Dünya</H1>
<H2>Bugün:</H2>
<H3>
<SCRIPT LANGUAGE=JAVASCRIPT>
<!--
tarih = new Date();
document.write(tarih);
//-->
</SCRIPT>
.</H3>
</BODY>
</HTML>
```

HTML sayfanıza <SCRIPT>..</SCRIPT> etiketleri arasına yerleştireceğiniz bu kodun çalışması için Server'ın hiçbir şey yapması gerekmez; kodu Browser çalıştıracak ve günün tarihini size (Türkçeleştirmediğiniz için İngilizce olarak) bildirecektir. Şimdi, bir de Server tarafında çalışan Script içeren sayfaya örnek verelim (merhaba2.asp). Bu kodu yazarken kapanan Script etiketinden sonra nokta olduğuna dikkat edin:

```
<HTML>
<HEAD>
<TITLE>VBScript ile Tarih</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>
<H1>Merhaba Dünya</H1>
```

```
<H2>Bugün:</H2>
<H3>
<SCRIPT LANGUAGE=VBScript RUNAT=SERVER>
Response.write(Date)
</SCRIPT>.
</H3>
</BODY>
</HTML>
```

Bu sayfada kullandığınız Script'i VBScript ile yazdığınızı farkettiniz. Yani? Bu HTML sayfası, Netscape'de görüntülenemez! Hayıır. Görüntülenir, çünkü Script'i Netscape değil, Server çalıştıracak. Bunu <SCRIPT> etiketinin içindeki "RUNAT" özelliği ile belirtiyoruz. ("Run," çalıştır; "at" ise "içinde, üzerinde" anlamına gelir. "RUNAT" "...de çalıştır" gibi bir anlama sahiptir.) Burada RUNAT'in karşısına yazdığımız SERVER ifadesi ile, Script'in Browser'a gönderilmeden önce Server'da çalıştırılmasını sağlıyoruz; Server'a "Bu Script'i Server'da Çalıştır!" demiş oluyoruz.

<asp0006.tif>

Netscape bu sayfayı görüntüleyecektir; ama galiba bir terslik var. merhaba.asp ile merhaba2.asp'nin ekran görüntülerini ve kodlarını karşılaştırırsanız, birinde noktanın olması gereken yerde olduğu halde diğerinde noktanın yanlış yerde olduğunu görüyoruz. Oysa iki kodu da karşılaştırırsanız, ikisinde de nokta işareti, kodun bittiği yerde yer alıyor. Yukarıda ASP'nin icrasında HTML sayfa Server'a verilmeden önce ASP.DLL tarafından içindeki ASP kodlarının icra edildiğini söylemiştik. Nitekim, burada gördüğünüz gibi, ASP.DLL, HTML sayfayı içinde Server'ın çalıştırması gereken bölümle birlikte Server'a gönderdi; Server kendisine gelen dosyada kendi icra etmesi gereken (RUNAT=SERVER özelliği verilmiş olan Script'i) bölümü icra etti ve sonucu Browser'a aktardı. Yani, Server'ın çalıştırdığı Script'in sonucu sayfaya ASP'den sonra eklendi. Dolayısıyla, nokta, yanlış yerde kaldı.

Bundan çıkartacağımız sonuç şu olacak: ASP sayfalarımıza, Browser, Server ve ASP.DLL tarafından çalıştırılacak kodları yerleştirirken, sırasına dikkat edeceğiz ve hangi sırada icra edilmelerini istiyorsak, o sırada koyacağız.

Server ile ASP.DLL'in ilişkisi sadece Script dilini çalıştırmaktan ibaret değildir. ASP, istemciden gelen HTTP İstemi (Request) ve HTTP'ye giden Karşılık (Response) unsurlarından tutun, ActiveX Data Objects (ADO, AcvtiveX Veri Nesneleri) aracılığıyla, işletim sisteminin sunacağı veritabanına erişim imkanını ve işletim sisteminin sunduğu dosya yönetimine kadar bir çok imkanı kullanır. Bu "imkanlar" ASP'nin eline geçtiği anda "nesne" (Object) sayılırlar Şimdi bu nesnelere biraz yakından bakabiliriz.

//////////////////////////KUTU//////////////////

ODBC İşliyor Mu?

Kolları sıvayıp, ASP'ye kodu yazmaya başlamadan önce yapmamız gereken bir iş daha var: ASP sayfaları geliştirme ortamımızda, ODBC (Open Database Connectivity, Açık Veritabanı Bağlantısı) olması gerekir.

Windows 98, 95 (OSR2) veya NT4.0 işletim sisteminizde Denetim Masası'nda ODBC, ODBC32 veya "ODBC Veri Kaynakları (32 Bit)" adlı bir simge görüyor olmalısınız. Bunu açın, Sistem DSN sekmesini ve açılan pencerede göreceğiniz Ekle düğmesini tıklayın. Buradaki Access, dBase, Excel, FoxPro, Paradox sürücüleri 4.00.3711.08 veya daha büyük mü? Değilse, Microsoft'un sitesinden (http://www.microsoft.com/data/download.htm) Microsoft Data Access Components (sürüm 2.1.1.3711.11 GA, 6.2 MB) güncelleme dosyasını indirin ve sisteminizi güncelleştirin. Windows 2000 kurulu sistemlerde bunu yapmaya gerek yok. Böylece sisteminiz, ilerde yazacağımız veri-yönlendirmeli Web uygulamaları için hazır hale gelmiş olacaktır.

///////////////KUTU BİTTİ///////////////

ASP’nin Unsurları

ASP tasarımcısı olarak, biz gerçekte ASP’nin Nesneleri ile birşeyler yaparız; başka bir deyişle ASP kodlarımız bu nesnelere yöneliktir, onları kullanma ve onlardan bir sonuç alma veya onlara bir sonuç aktarma amacına yöneliktir. ASP’nin Nesneleri altı grupta toplanır:

Application/Uygulama: Bir ASP sitesi, gerçekte bir Uygulama Programı olarak görülür. Bu, HTML/CGI geleneğine aşina tasarımcı için yeni bir kavram. ASP’yi icad edenler; bir ziyaretçi bir ASP sayfasından girerek, bir sitede surfing’e başladığında, onu bir programı işleten bilgisayar kullanıcısı olarak görüyorlar. Böylece, sitemiz, her ziyaretçinin karşısına çıktığında “bir program çalışmış” gibi sayılıyor. Bu yaklaşımın Web tasarımcısı olarak bize kazandırdığı imkanları ele alacağız.

Session/Oturum: Bir ziyaretçi sitemize geldiğinde, hangi sayfamızı talep ederse etsin, bu bağlantı ASP açısından bir oturum sayılır. Her oturumun belirli bir süre devam eden özellikleri, değişkenleri ve değerleri vardır. Site tasarımında oturum özelliklerinden geniş ölçüde yararlanacağız.

Request/Talep: Browser’dan Server’a ulaşan bütün bilgiler, Request (Talep) nesnesinin ögeleridir. Bu nesneyi kullanarak, istemciden gelen her türlü HTTP bilgisini kullanırız.

Response/Karşılık: Server’dan ziyaretçinin bilgisayarına gönderdiğimiz bütün bilgiler, çerezler (cookie) ve başlıklar (Header) Response (Karşılık) nesnesinin ögeleridir. Bu nesneyi kullanarak ziyaretçiye göndermek istediklerimizi göndeririz.

Server/Sunucu: ASP, Web Server programını bir nesne olarak ele alır ve onun bize sağladığı araçları ve imkanları kullanmamızı sağlar.

ObjectContext/Nesne Bağlamı: Microsoft'un Transaction Server (MTS) programının sunduğu hizmetlere erişmemizi sağlar. MTS, ASP sayfaları içinden, uygulama programlarından yararlanmamızı sağlar. ASP uzmanlığınızı ileri düzeylere ulaştırdığınız zaman MTS ve ObjectContext nesnesinden yararlanabilirsiniz.

ASP'nin Dili

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.

VBScript'e Giriş

Bu kitapçıktaki örneklerimizi VBScript diliyle yazacağımıza göre, önce hızlı bir VBScript kursu görsek iyi olur. Visual Basic dilini biliyorsanız, VBScript biliyorsunuz sayılır. VBScript, güçlü bir dildir; ancak Netscape firmasının hiç bir zaman Browser'ında istemci tarafında çalıştırılabilecek diller arasında kabul etmemesi sebebiyle VBScript, Web'in istemci tarafında kendisinden bekleneni yapamadı. MS'un Browser'ı Internet Explorer ise VBScript ile yazacağınız İstemci-Tarafı kodları okuyabilir ve icra edebilir.

Ne var ki ASP kodlarımız hiç bir zaman ziyaretçinin Browser'ının yüzünü göremeyeceği ve sadece Server'da çalışacağı için Server'da VBScript desteği bulunduğu sürece, ASP sayfalarınızı VBScript ile yazabilirsiniz. Bir Server'da ASP desteği varsa, VBScript desteği de var demektir.

VBScript'in hemen hemen bütün komutlarını ve yöntemlerini ASP'de kullanabilirsiniz. Ancak bunun bir kaç kısıtlaması vardır. VB veya VBScript'e ASP dışında aşina iseniz, mesaj kutusu (MsgBox) ve girdi kutusu (InputBox) aracılığı ile programlarınıza kullanıcının bilgi girmesini sağlayabileceğinizi biliyorsunuz demektir. Bu iki komutu ASP içindeki VBScript kodunda kullanamayız. Ayrıca ASP teknolojisi zaten VBScript'in bütün komutlarını ve deyimlerini kullanmanızı da gerekli kılmayacaktır. Göreceksiniz ki, mükemmel ASP sayfaları oluşturmak için bile bir avuç VBScript komutu kullanacağız.

ASP sayfalarımızdaki HTML kodları ile VBScript (veya diğer Script dillerinin) kodlarını birbirinden ayırmamız gerekir. Bu ASP.DLL'ye, HTML'in nerede bittiğini, Script diliyle yazılmış kodun nerede başladığını gösterebilmemiz için gerekli. Bunu sağlamak için Script diliyle yazılmış herşeyi "<%" ve "%>" işaretleri arasına alırız. ASP.DLL bu işaretleri

görünce, içindekileri "yazmak" yerine "yapar." Bir ASP sayfanızda HTML'in klasik "<" ve ">" işaretleri arasındaki unsurlar, ASP.DLL tarafından ziyaretçiye gönderilecek olan sayfaya aynen aktarılır; ancak "<%" ve "%>" arasındaki herşey, başta belirttiğiniz LANGUAGE etiketinde yazılı Script dilinin yorumlayıcısına verilir; yorumlatılarak, gereği yerine getirilir.

"<%" ve "%>" işaretlerine "sınırlayıcı" denir. Sınırlayıcının içinde bir veya daha çok satır kod bulunabilir. Sınırlayıcılar ve içindeki Script, HTML etiketlerinin içinde veya dışında yer alabilir. Sınırlayıcının içindeki kodlarımızı açıklamak için koyacağımız yorum satırlarının başına tek tırnak işareti (') koyarız. İşte bu kuralları uyguladığımız bir ASP sayfası örneği:

```
<% @LANGUAGE=VBscript %>
<html>
<head>
<title>Hoşgeldiniz!</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>
<center>

<%
' Yazı tipi boyutunu tutacağımız bir değişken tanımlayalım
Dim fontBoyut
%>

<%
' yazı tipi boyutunu 1'den 7'ye kadar değiştirelim
For fontBoyut = 1 To 7
%>
<font size = <%=fontBoyut%>>
Hoşgeldiniz!<br>
<% Next %>
```

```
</center>
<h3>Bugün <% =WeekdayName(Weekday(Date)) %>, <% = Date %>.
Şu anda Server'da saat: <% = Time %>.<p>
</h3>
</body>
</html>
```

Burada sınırlayıcı arasında tek veya çok satırlı VBScript kodları ile başında tek tırnak olan icra edilmeyen, yorum satırlarını görüyorsunuz. HTML etiketinin içine gömülmüş VBScript kodu ise HTML'in <FONT> etiketinde yer alıyor: <font size = <%=fontBoyut%>>. Burada karşılaştığımız "<%=" ifadesi, ASP'ye, "Bu değişkenin değerini bul ve tam buraya yaz!" dememizi sağlıyor. Bu ifade daha sonra yakından ele alacağımız Response.Write metodunun kısaltılmış halidir. HTML etiketinin içine yazdığımız VBScript bölümünün kendi sınırlayıcı işaretlerinin yine de kullanıldığına dikkat edin.

<asp0007.tif>

Bir iki yazım kuralı

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
%>
```

Değişkenler

Programcılıkta işlemlerimizi değişkenlerle yaparız. değişkeni bir kap gibi düşünebilirsiniz. Sözgelimi "Gün," değişkenin adı ise bu değişkenin değeri Pazar, Pazartesi, Salı, vd., olabilir. Her değişken, türüne göre, ya bir ya da daha fazla değer tutar. Adından da anlaşılacağı gibi değişkenin değeri değişir! Bu değişikliği biz yapabiliriz; programın kendisi yapabilir.

VBScript'te, bir çok başka bilgisayar programlama dilinden farklı olarak değişkenlerin tanımlanması veya "beyan edilmesi," "boyutlandırılması" gerekmez. Belirtilmemiş, önceden tanımlanmamış bir değişkene değer atamaya kalkarsanız, VBScript bunu mükemmel şekilde kabul eder. Fakat bu kötü bir programcılıktır. 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. Örnekler:

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

Şimdi burada Gun, Ay, Ogrenci, Not adıyla dört değişken oluşturduğumuzu ve bunlara sırasıyla "Pazartesi," "Ocak," "Necip" ve "5" değerlerini atadığımızı görüyorsunuz. Bu noktada VBScript yorumlayıcısına sorsanız, "Peki, söyle bakalım, Gun değişkeninin

değeri nedir?" diye, yorumlayıcı size "Bunu bilmeyecek ne var? Elbette Pazartesi'dir!" derdi. Belki bu noktada siz de kendi kendinize "Pazartesi, Ocak ve Necip kelimeleri tırnak içinde iken, acaba 5 rakamı neden tırnak içinde değil?" diye soruyorsunuz. Güzel soru. Ama cevabı için biraz bekleyeceksiniz.

Değişken isimleri, mutlaka harfle başlamalıdır; içinde noktalama işaretleri bulunamaz ve uzunluğu 255 karakteri geçemez.

////////////////KUTU////////////

Kötü programcılığı önlemek için!

Değişkenleri tanımlamadan kullanmak "kötü programcılıktır" demek, ilerde kendi başınıza elinizle program hatası getirirsiniz demektir. Daha sonra örneklerini göreceğiz; ASP, sitemize giren bir ziyaretçiye bir oturum (Session) açar ve bazı değişkenleri oturum boyunca aklında tutar. Bir sayfada kullandığınız bir değişkeni, daha sonraki sayfada kullanabilmek için, bu değişkenin değerinin yeni sayfada değişmemesi gerekir. ASP programı yazarken, bazen gelişi-güzel değişkenlere değer atayabilirsiniz. Neden? O anda aklınıza değişken adı olacak kelime gelmediği için! Bu değişken adını daha önce kullanmışsanız ve içinde biraz sonra kullanacağınız bir değer varsa, değer değiştirilmiş olacaktır. VBScript, savurgan ve dağınık programcılığı önlemek için OPTION EXPLICIT imkanını verir. 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.

//////////////////////KUTU BİTTİ//////////////

Peki, şöyle bir değişken tanımlama acaba ortaya nasıl bir değişken çıkartır:

```
<%
DIM Gunler(31), Aylar(12), Ogrenciler(210), Notlar(10)
%>
```

Bu yöntemle oluşturduğumuz kodlarla elde edilecek "kaplar," birden fazla değer tutabilir. Yani:

```
<%
DIM Gunler(7), Aylar(12), Ogrenciler(21), Notlar(10)
Gunler(1) = "Pazartesi"
Aylar(3) = "Mart"
Ogrenciler(12) = "Necip"
Notlar(5) = 5
%>
```

Bu durumda içinde 7 ayrı değişken tutabilecek olan Günler değişkeninin 1 numaralı olanının değeri "Pazartesi," 12 değer tutabilecek olan Aylar değişkeninin 3 numaralı olanının değeri "Mart," 21 ayrı değer tutabilecek olan Ogrenciler değişkeninin 12 numaralı olanının değeri "Necip" ve nihayet 10 ayrı değer tutabilecek olan Notlar değişkeninin 5 numaralı olanının değeri ise 5 olacaktır. Böyle, birden fazla değer tutabilen değişkenlere Dizi Değişken veya Array denir.

//////KUTU////

Array Fonksiyonu

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!

////////////////KUTU BİTTİ///////////////

Şimdi, aklınıza şu soru gelebilir: Dizi değişken oluşturacağımıza, neden “ogrenci1,” “ogrenci2,” “ogrenci3” diye değişkenler oluşturmuyoruz ve bunlara değerler atamıyoruz? Kötü soru! Dolayısıyla cevabını hemen verelim. Dizi değişkenlerde bir değişkene numarasıyla atıfta bulunabilirsiniz. Ve numara yerine bir başka değişkeni kullanabilirsiniz. Örneğin, OgrenciNo değişkeninizin değeri 12 ise

```
<%
....
If Ogrenciler(OgrenciNo) = “Necip”
...
%>
```

şeklinde bir döngü ile aradığınız öğrencinin Necip olup olmadığını sınayabiliriz. (Döngülere birazdan geliyoruz!)

Başka programlama dillerine özellikle Visual Basic’e aşina olanların aklına şöyle bir soru gelebilir: VBScript’te değişkenin metin mi, sayı mı olduğunu ayırt etmiyor muyuz? Başka programlama dillerinde bir değişkenin değeri harf ve rakamlardan oluşuyorsa, yani

matematik işlem yapmaya elverişli değilse bunlara String (Alfanümerik, karakter değerler) denir. Programlama dillerinde bir de matematik işlem yapmaya elverişli değişken türü vardır: Sayı (Number). VBScript, bir değişkene alfanümerik (karakter, metin) olarak atadığınız değeri çift tırnak içine almanızı ister. Sözgelimi Ogrenci(12) değişkeni için Necip değerini atamak istiyorsunuz: Necip kelimesini çift tırnak içine almak zorundasınız. Sayı olarak kullanacağınız değerleri ise tırnak içine almazsınız. (Sayı olarak kullanacağınız değerlerin sadece rakam olabileceğini söylemek gerekmez!) Fakat, işleri karıştıran nokta, VBScript açısından değişken türü diye bir şey olmamasıdır. Visual Basic, küçültülüp Script dili haline getirilirken, değişken türü özelliği alınmayınca, ortaya işleri karıştıran (veya kolaylaştıran) böyle bir durum çıkmış bulunuyor. Karışıklık, çift tırnak içinde verdiğiniz bir değeri matematik işlemde kullanmaya kalktığınızda karşınıza çıkabilir. Rakam olmayan bir karakter-dizisini bir değişkene tırnaksız olarak atadığınızda VBScript "tanımsız değişken" vermeye kalktığınızı söyleyerek, duracaktır.

VBScript'in bu eksikliğini gidermek için bazı ASP programcıları değişken adlarının önüne karakter-dizileri için "str" harflerini koyarlar: strAy, strOgrenciler, gibi.

İsterseniz, VBScript'in karakter-dizisi ve sayı değişkenleri nasıl ayırt ettiğini veya etmediğini bir kaç örnekle görelim. Şu kodu yazarak, çalıştırın:

```
<% Option Explicit %>
<HTML>
<%
Dim Degisken(2), Toplam
Degisken(1) = "Necip"
Degisken(2) = "Dayanır"
Toplam = degisken(1) + Degisken(2)
%>
<% =Toplam %>
</HTML>
```

Sonuç: "NecipDayanır" değil mi? (Arada boşluk olmamasına aldırmayın şimdilik!) Peki; şimdi Degisken(1) değeri olarak tırnak içine alarak 5, Degisken(2) değeri olarak yine tırnak içinde 10 yazın. Programı tekrar çalıştırın. Sonuç? 510!? Peki; tırnakları kaldırın 5 ve 10'un çevresinden. Sonuç? 15. Oh, neyse! Bir an için VBScript matematik bilmiyor sandık! Bu durum VBScript'in matematik bilgisinin kıtlığından değil, tırnak içindeki değerlerin, VBScript ayrım yapmıyor bile olsa, karakter-dizisi (String) iken toplama işleminde ard arda eklenmesinden (concatenate) kaynaklanıyor. Tırnakları kaldırdığınızda, VBScript bu değerlerin sayı olduğunu anladı ve doğru işlemi, yani toplama işlemini yaptı. Şimdi VBScript'in değişkenleri ele alış tarzını daha iyi kavramak için Degisken(1) değeri olarak tırnak işareti koymadan kendi adınızı, Degisken(2) değeri olarak da soyadınızı yazın ve programı çalıştırın. Ne sonuç aldınız?

VBScript'in kullandığı tek tür değişkene variant denir. Variant, karakter-dizini (String) de olabilir, sayı (Number) da. Fakat bir variant'ın içindeki değer, veri olarak nitelendiğinde şu türlerden birine girer:

Boolean	Değişkenin değeri ya True (doğru) ya da False (yanlış) olur; True'nun değeri –1, False'un değeri ise 0'dır.
Byte	0 ile 255 arasında bir sayısal değer olabilir.
Double	Yüzen noktalı değer denilen veri türüdür. Pozitif sayılar için $4.9E^{-324}$ ile $1.8E^{308}$ arasında, negatif sayılarda $-4.9E^{-324}$ ile $-1.8E^{308}$ arasında bir değer alabilir.
Date/Time	Önceden belirlenmiş biçimlerde tarih ve zaman bilgisi içerir.
Empty	Tanımlanmış ancak henüz değer atanmamış (boş) değişken türüdür.
Error	Programın hata mesajlarını ve hata değerlerini tutar.
Integer	Noktalık bölüm içermeyen tamsayı verilerdir; değeri -32.768 ile +32.767 arasında olabilir.

Long Noktalık bölüm içermeyen tamsayı verilerdir; değeri -2.147.483.648 ile 2.147.483.648 arasında olabilir.

Null İçinde veri bulunmamak üzere tanımlanmış değişkenlerdir.

Object Windows OLE Nesnelerini tutmak üzere tanımlanmış değişkenlerdir.

Single Yüzen noktalı değer denilen veri türüdür. Pozitif sayılar için $1.4E^{-45}$ ile $3.4E^{38}$ arasında, negatif sayılarda $-1.4E^{-45}$ ile $-3.4E^{38}$ arasında bir değer alabilir.

String Alfanumerik veri tutan değişkenlerin değeridir

Sabit Değerler

VBScript'te bir kere verdiğiniz değeri hiç değişmeyen unsurlar (değişken diyemezdik, değil mi?) vardır. Sabit değer, bütün ASP sayfası 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"
```

VBScript'te İşlemciler (Operatörler)

Bilgisayar Operatörü ile program Operatörü arasında fazla bir fark yoktur. Her ikisi de verdiğiniz değerleri ya karşılaştırır bir sonuç bulurlar; ya da bu değerlerle aritmetik işler yapar ve bir sonuç ortaya çıkartırlar. Bilgisayar Operatörü daha sonra bu iş için sizden para ister; program operatörü istemez! VBScript'in operatörleri ve yaptıkları işler şöyle sıralanır:

Operatör	İşlev	Sınıfı
+	Toplama	Aritmetik
-	Çıkartma	
*	Çarpma	
/	Bölme	

^	Üssünü alma	
\	Tamsayı bölme	
Mod	Modüler aritmetik	
=	Bir değişkenin diğerine eşit olduğunu sınar	Karşılaştırma
<>	Bir değişkenin diğerine eşit olmadığını sınar	
>and<	Bir değişkenin diğerinden büyük veya küçük olduğunu sınar (and kelimesi var)	
>= and <=	Bir değişkenin diğerinden büyük veya eşit, veya küçük veya eşit olduğunu sınar (and kelimesi var)	
Is	Bir ifadedeki iki referansın aynı Nesne'ye yapılıp yapılmadığını sınar	
And	Bir veya daha fazla değişkeni test olarak karşılaştırır	Mantıksal
Or	Bir işlemin devamı için hangi koşulun oluşması gerektiğini sınar	
Not	Bir ifadeyi negatif hale getirir	
XoR	Sadece bir koşulun doğru olup olmadığını sınar	
Eqv	İki değişkenin eşitliğini sınar	
Imp	İki ifadede mantıksal implikasyon işlemi yapar.	

VBScript ile yazacağımız ASP sayfalarında bu işlemcileri beklediğimiz sonucu verebilmesi için kullanım sıraları önemlidir. Bunu bir örnekle açıklayalım. 100'den 6'yı çıkarmak ve sonucu 2'ye bölmek istiyorsunuz; sonuç 47 olacaktır. Yani: 100-6/2. Değil mi? Bu işlemin VBScript'teki sonucu 97 olacaktır. Çünkü, VBScript önce 6'yı 2'ye bölecek ve

elde ettiği sonucu 100'den çıkartacaktır. VBScript'te, aritmetik işlemlerin yapılma sırası şöyledir:

Operatör	İşlev	Öncelik
+	Toplama	3
-	Çıkartma	3
*	Çarpma	2
/	Bölme	2
^	Üssünü alma	1

VBScript ile hesap işlemi yaparken, aritmetik işlem sırasını karıştırarak hatalı sonuç almamak için sık sık parantez kullanmak yerinde olur. Yukarıdaki örnek şöyle yazılabilirdi: (100-6)/2. Tabii amacınız 100'ü 6/2'ye bölmek ise bu işlemi şöyle yazmalısınız: 100-(6/2).

VBScript'de Program Kontrolü

İster Script diliyle, ister gerçek programlama diliyle yazılsın, bir bilgisayar programının varlık sebebi, çeşitli durumları değerlendirerek, belirli durumlarda belirli kararlar verebilmesidir. Bunu programın kontrol öğelerini kullanarak yaparız. Programlar, bu öğeler sayesinde karşılaştırma yaparlar; belirli durumların oluşup oluşmadığını sınarlar; veya belirli bir durumun oluşmasına veya sona ermesine bağlı olarak bir iş yaparlar veya yapmazlar. Bunu sınamalarla (koşullu ifadelerle) veya döngülerle sağlarız. Kimi zaman da, programa (programın mantığı çerçevesinde) istediğimiz anda yapmakta olduğu işi durdurarak, başka bir işi yapmasını bildirebiliriz. Bunlara da Süreçler (veya Prosedürler) denir. (Adındaki Basic'e bakarak, VBScript'i basit bir dil sanmıyordunuz, umarım!)

Mantıksal Sınamalar

VBScript'te programın karar vermesini sağlayan ilk kontrol ögesini "eğer ... ise... yap!" şeklinde özetleyebiliriz. VBScript bu işlemi iki ayrı ifadeyle yaparız:

If.. Else

VBScript'in vereceğiniz bir durumun bulunup bulunmadığını sınamasını sağlar. Genel yazım kuralı şöyledir:

```
If şart Then
            [şart doğru ise yapılacak işler]
Else
            [şart doğru değilse yapılacak işler]
End If
```

Bunu bir örnekle ifade edelim: Eğer saat 12'den önce ise sayfaya "Günaydın" yazdır; saat 12'den sonra ise "Tünaydın" yazdır.

Fakat burada bir eksiklik var: Saat 18'den sonra ise sayfaya "İyi akşamlar!" yazdırmamız daha doğru olmaz mı? If döngüsü kendi içinde sınırsız Elseif (ikinci şartlı döngü) imkanı vererek bize bunu sağlar. Her Elseif'i yeni bir If gibi düşünebilirsiniz. İşte şu senaryoyu gerçekleştirecek kod örneği (hosgeldiniz01.asp):

```
<HTML>
<HEAD>
<TITLE>ASP ILE SAATE GORE SELAM</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>
<H2>
<CENTER>
<%
If Hour(Now) <12 Then
            Response.Write "Günaydın! "
```

```
ElseIf Hour(Now) >= 18 Then
            Response.Write "İyi akşamlar! "
Else
            Response.Write "Tünaydın! "
End If
Response.Write "<BR>"
Response.Write "Site Onarım Sitesine Hoşgeldiniz"
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

Bu programı çalıştırdığınız zaman, çalıştırdığınız saate göre sayfadaki selamın değiştiğini göreceksiniz. (Bu örnekte, sadece şartlı döngü işlemi yapmıyoruz; fakat aynı zamanda Hour(Now) fonksiyonu ile tanışıyoruz ve sayfadaki bütün unsurları ASP yöntemiyle yazdırıyoruz! Fakat şimdilik bu iki unsurun üzerinde de durmayın.) Programımız nasıl çalışıyor? Yukarıda başka vesile ile VBScript'in kullanılmaya hazır fonksiyonları vardır, demiştik. Bunlardan biri de o andaki saati ve tarihi bildiren Now() fonksiyonudur. Bu fonksiyondan dönen değerle bu andaki saati öğreniyoruz; ve bunu önce 12 ile karşılaştırıyoruz. Fonksiyondan dönen değer, eğer 12'den küçükse, programımız Response (Karşılık) Nesnesi'nin .Write Metodu'nu kullanarak (Nesneler ve Metodlar meselesi üzerinde de durmayın!) ziyaretçinin Browser penceresine "Günaydın" yazdırıyor.

//////////KUTU//////////

Dönen Değer

Fonksiyonlar, kendilerini göreve çağıran VBScript komutlarına ve işlemlerine bir değer sunarak karşılık verirler. Buna fonksiyondan dönen değer denir. Yani diyelim ki Now() fonksiyonunu göreve çağırdınız. Bu fonksiyon derhal işletim sisteminden saati ve tarihi öğrenerek kendisini göreve çağıran işleme bildirir. Daha sonra VBScript'in kullanılmaya

hazır diğer bir çok fonksiyonunu kullanacağız ve kendi fonksiyonlarımızı yazacağız. Ne zaman bir fonksiyona bir noktada atıf yaparsak, o noktaya fonksiyon tarafından bir değer getirileceğini bimemiz gerekir.

////////////////KUTU BİTTİ////////

Eğer bu ilk sınamanın sonucu doğru değilse, VBScript If satırından sonraki birinci deyimi atlayacak ve ikinci deyimi icra edecektir. Yani eğer saat 12'den küçük değilse, ElseIf satırı icra edilecektir. ElseIf de tıpkı If gibi işlediği için bu kez yeni bir sınav yapılacak ve saatin 18'e eşit veya büyük olup olmadığı sınanacaktır. Eğer saat 18'e eşit veya büyükse, (sınav sonucu doğru ise) ilk satır icra edilecek ve ziyaretçinin Browser penceresine "İyi akşamlar!" yazdırılacaktır. Eğer bu sınavın sonucu da doğru değilse, ElseIf'in ikinci satırı icra edilecektir. Bu satırda ise Else bulunuyor. Else, If ve ElseIf gibi sınav yapmaz; ne olursa olsun, kendisinden sonra gelen deyimi yerine getirir. Yani saat 12'den küçük değilse, 18'den küçük veya 18'e eşit değilse, yani 12 ile 17 arasında ise, ekrana "Tünaydın" yazılacaktır.

Select Case

VBScript'in bir diğer duruma bakarak karar verme ifadesi, Select Case (Durum Seç) yapısıdır. Bu kontrol öğesinin nasıl çalıştığını şöyle özetleyebiliriz:

Durum Seç (Durumların listesi veya durumları belirten bir değişken)

Durum 1 : Yapılacak işler

Durum 2: Yapılacak işler

Durum 3: Yapılacak işler

Durum n: Yapılacak işler

Seçmeyi Bitir

VBScript, verdiğiniz durum listesine veya içinde çeşitli değerler bulunan değişkene bakarak, bu değişkenin her bir değerini bir "durum" sayacak ve verdiğiniz durumlardan hangisini tutuyorsa, ona ait komut dizisini icra edecektir. Yukarıdaki sayfamızı bu kez bu yapıyı kullanarak yazalım (hosgeldiniz02.asp):

```
<HTML>
<HEAD>
<TITLE>ASP ILE SAATE GÖRE SELAM</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>
<H2>
<CENTER>
<%
Select Case Hour(Now)
            Case 0,1,2,3,4,5,6,7,8,9,10,11
            Response.Write "Günaydın!"
            Case 12,13,14,15,16,17
            Response.Write "Tünaydın"
            Case Else
            Response.Write "İyi Akşamlar!"
End Select
Response.Write "<BR>"
Response.Write "Site Onarım Sitesine Hoşgeldiniz"
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

Select Case komutuna, içindeki değerleri "durum" sayacağı dizi veya değişken olarak VBScript'in kullanılmaya hazır fonksiyonlarından Hour(Now)'ı veriyoruz. Bu fonksiyondan, 0 ile 24 arasında bir değer dönecektir. Bu değer Select Case için bir durum

demektir. Select Case, bu değer ile altta sıralanan Case'leri karşılaştıracak ve elindeki değer hangi Case'i tutuyorsa ona ait komutları icra edecektir. Sonuncu Case'e lütfen dikkat edin: Burada Case olarak Else (başka) veriliyor. Bu bizi, 17'den 23'e kadar olan saatleri sıralamaktan kurtarır. 0'dan 11'e kadar olan saatlerle 12'den 17'ye kadar olan saatleri sıraladığımıza göre başka hangi saat olursa olsun, ziyaretçimize "İyi akşamlar!" dileyebiliriz. Eğer 24'den sonra ve 04'den önce ziyaretçinize "İyi geceler!" dilemek isterseniz, bu programı nasıl değiştirirdiniz?

Döngüler

Sınama bir programın akışını kontrol için kullanacağımız birinci en önemli unsur ise, döngü de ikinci en önemli unsur sayılır. Hatta programcının tembellik katsayısına göre, belki de birinci en önemli unsuru bile sayılabilir! Çünkü Döngü (Loop) programa, bir işi biteviye yaptırmaya yarar. Tabiî bu iş sonsuza kadar sürecek olursa, buna Endless Loop (Sonsuz Döngü) denir; en iyi program ve Windows dondurma yöntemidir! VBScript'te kullanabileceğimiz döngü yöntemleri şunlardır:

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
```

Burada, "sayaç" yerine istediğiniz bir değişken adını, "başlangıç" yerine sayacın başlamasını istediğiniz sayıyı, "son" yerine sayacın durmasını istediğiniz sayıyı, ve "adım" yerine, sayacın kaçar-kaçar artmasını istediğinizi yazabilirsiniz. En sondaki Next deyimi ise döngünün bir sonraki adıma geçmesini sağlar. Bu adımda sayaç, Step kelimesi varsa,

karşısındaki değer kadar arttırılır ve yapılacak işler yeniden yapılır. Bir örnek yapalım (gunler.asp):

```
<HTML>
<HEAD>
<TITLE>ASP ILE GÜNLERI SAYMA</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>
<H2>
<CENTER>
<%
Dim Gunler
Gunler = Array("Pazartesi", "Salı", "Çarşamba", "Perşembe", "Cuma", "Cumartesi",
"Pazar")
For sayac = 0 to 6
            Response.Write Gunler(sayac)
            Response.Write "<BR>"
Next
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

Bu ASP kodunda, Gunler adıyla bir dizi-değişken oluşturuyoruz ve bu değişkenin yedi hanesine, günlerin adlarını atıyoruz. Sonra, sayac adlı sayacı 0'dan 6'ya kadar arttırıyoruz (Bir sayaç birer birer artsın istersek, Step bölümüne adım sayısı yazmayız). Şimdi kendimizi bir an için VBScript'in yerine koyalım ve birinci adımda yapacağımız işi düşünelim: "Hmm.. Programcı bey, benim sayac'ı önce 0 yapmamı istiyor; peki sayac 0 olsun. Sonra, Gunler dizi-değişkeninden sayaç değeri ile aynı sayıyı taşıyan değişkeni alıp, bunu ziyaretçinin Browser'ına yazmamı istiyor. Peki, Gunler(0) ne imiş, bakalım. Hmm

Gunler(0) Pazartesi imiş. o halde ziyaretçinin ekranına bir Pazartesi kelimesi yazalım. Sonra bir de
 kodu yazmamı istiyor. Onu da yazalım.. Şimdi, sırada Next var. Yani bir sonraki adıma devam edeceğiz. Step değeri olmadığına göre sayacı bir arttırayım. Sayaç böylece 1 oldu. ...”

Ve böylece VBScript, sayacın son değeri olan 6’ya ulaşıncaya kadar, biteviye Gunler dizi-değişkeninden sayacın değerine göre değer seçerek ve bunu ekrana yazdırarak, işini yapacaktır. Bu bakımdan VBScript, güvenilir ve çalışkan bir arkadaştır!

While...Wend

Ne var ki, program mantığı bazen bize böyle 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. Bu durumda kodumuzda For.. Next arasında kalan bölümde şu değişikliği yapabiliriz:

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

Burada While döngüsünün Wend kelimesiyle sonlandırıldığına dikkat edin. While satırındaki sayacı değiştirdik, programın sayaç 5’den küçük veya 5’e eşit iken işlemesini sağladık. For’dan farklı bir diğer ifade ise sayacı arttıran “sayac = sayac + 1” ifadesidir. Bu ifade, ilk bakışta garip görünebilir. Fakat bilgisayar açısından bu “sayac’ın o andaki değerini

al, 1 ile topla ve bulduğun yeni değeri sayacın mevcut değerinin yerine yaz!" demektir. VBScript sayacı bir arttırdıktan sonra önce While satırındaki şartın gerçekleşip gerçekleşmediğine bakar; gerçekleşmiş ise Wend'i izleyen ilk satıra gider; gerçekleşmemişse While döngüsünün içindeki işi yapmaya devam eder. Kendinizi VBScript'in yerine koyup, birinci ve ikinci adımda nasıl düşündüğünüzü ve ne iş yaptığınızı söyleyebilir misiniz?

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 biteviye 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

Burada, Loop komutu şartın henüz gerçekleşip gerçekleşmediğini sınar ve henüz gerçekleşmemişse verilen işleri yapar; gerçekleşmişse bir sonraki satıra geçer.

Visual Basic metinlerinde bu döngüye verilen klasik örnek, bilgisayara yazı-tura attırmaktır! Biz de ASP sayfamıza yazı-tura attırabiliriz. Bunun için şu kodu yazın ve yazi-tura.asp adıyla kaydedin:

```
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP ILE YAZI-TURA</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>
<H2>
<CENTER>
<%
Dim ParaAt, Yazi, Tura, Atis
Randomize
Yazi = 0
Tura = 0
```

```
Atis = 0
Do While Tura < 3
            atis = Atis + 1
            ParaAt = Int(Rnd * 2) + 1
            If ParaAt = 1 Then
%>
Yazı!<P>
<%
            Yazi = Yazi + 1
            Else
%>
Tura!<P>
<%
            Tura = Tura + 1
            End If
Loop
%>
3 Tura getirebilmek için parayı <%=Atis%> kere atmak gerekti!
</HTML>
```

Bu kodda şu ana kadar karşımıza çıkmamış olan Randomize (Tesadüfî sayı bulma) Fonksiyonu'nun kullanıldığını görüyoruz.

//////////////////KUTU//////////////////

Tesadüfî Sayı

Bilgisayarın matematik işlemlerde, özellikle istatistik hesaplamalarla kullanılması tesadüfî (rastlantısal) sayı üretmeyi gerekli kılmıştı. 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 almıştır. 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>
```

Bu dosyayı tesaduf.asp adıyla kaydedip çalıştırın; Browser'ın Yenile düğmesini her tıkladığınızda ekranda yeni bir sayı göreceksiniz. Bu sayıların rastlantısal olması, bir kere daha gelmeleri için hiç bir kural (örneğin her rakamın 123 kerede bir gelmesi veya 1 milyon 245 bin kerede bir gelmesi gibi) bulunmamasıdır. İşin tekniğini bilmek isterseniz, VBScript, her Rnd komutu icra edildiğinde bilgisayarın saatini öğrenir; içinden seçeceği bir rakamı son derece karmaşık bir formülden geçirerek size bir rakam verir. bu rakam daima 0 ile 1 arasında olur. "(Rnd*6)+1" formülü bize 1 ile 6 arasında (yani bir zarın değerlerinden her hangi biri), "(Rnd*13)+1" formülü ise 1 ile 13 arasında (yani bir iskambil destesindeki kağıt değerlerinden herhangi biri) bir değer verir. Fakat dikkat: bu değerler tam sayı değildir!

//////////////////KUTU BİTTİ//////////////

Programımızın bütün işlemi Do döngüsü bölümünde yapılıyor ve bilgisayarın bir tesadüfî sayı üretmesi esasına dayanıyor. Bunu Randomize ve Rnd fonksiyonları ile yapıyoruz. Rnd'un verdiği tesadüfî rakamı, iki ile çarpıyor ve çıkan sayıyı 1 ile topluyoruz; böylece ortaya 1'den büyük 3'den küçük bir kesirli rakam çıkmış oluyor (Neden?). Bu rakamı Int() fonksiyonundan geçirerek, kesirinden kurtarıyoruz.

//////////////////////KUTU//////////

Tam Sayı Elde Etmek için: Int ve Round

Rnd fonksiyonu ile ilgili yukarıdaki örneği yaptıysanız, dönen sayının 0 ile 1 arasında, yani daima kesirli olduğunu görmüş olmalısınız. Bazen bizim sayfalarımızdaki hesaplamalar veya veritabanından alınan değerler de kesirli olabilir. Örneğin öğrencilerin not ortalamalarını hesaplattırırken VBScript size sonu gelmez kesirler verecektir. Oysa çoğu zaman bu rakamların ya yukarı "yuvarlanması", ya da sadece tam sayı bölümü gerekir.

VBScript'te Int() fonksiyonu, bize bir sayının tam sayı bölümünü verir. Diyelim ki elimizdeki KesirliSayi değişkeninin değeri 123,234567 olsun.

```
Tamsayi = Int(KesirliSayi)
```

işleminden sonra Tamsayi değişkenin değeri 123 olur.

Fakat kimi zaman bir sayının kesirli bölümünü böyle bıçakla kesip atmak işimize gelmeyebilir. Round() fonksiyonu, kesirli bir sayıyı yukarı veya aşağı "yuvarlayarak" tam sayı haline getirir. Bu kez ki elimizdeki KesirliSayi değişkeninin değeri 5,6 olsun.

```
Tamsayi = Int(KesirliSayi)
```

işleminden sonra Tamsayi değişkenin değeri 6 olur. Kesirli sayı 56,2 ise, Round() fonksiyonu bize 56 değerini verir.

//////////////////////////KUTU BİTTİ///////////////

Programımız, elde ettiği ve Paraat değişkenine kaydettiği bu sayı 1 ise, Yazı gelmiş sayıyor; ve Browser Penceresine "Yazı!" yazıyor. Bu arada yapılan atış sayısını kaydettiğimiz Atis ve gelen tura sayısını tuttuğumuz Tura değişkenlerinin değeri bir arttırılıyor. ParaAt değişkeninin değeri başka bir şeyse (ne olabilir?), programımız bu kez tura geldiğine hükmediyor ve Browser penceresine "Tura!" yazıyor. Do döngüsü, Tura gelen atışların sayısı 3 oluncaya kadar devam ediyor. Çünkü Do döngüsünü While Tura < 3 (Tura

3’den az iken) deyimi ile çalıştırıyoruz. Ve program sonunda 3 tura gelinceye kadar kaç atış yaptığını yazıyor.

<asp0008.tif>

Bu ASP sayfası görüntülenirken Browser’ın Yenile düğmesini tıklarsanız, her seferinde Tura getirmek için farklı sayıda atış yapmak gerektiğini; aynı sayıda atış yapılsa bile turalarla yazıların yerinin değiştiğini göreceksiniz.

Dizi değişkenler için döngü: For Each..Next

For..Next gibi çalışan bu özel döngü, sayaç değeri kullanmaz, fakat bir dizi değişkenin bütün değerleri için bir kere icra edilir. Dizi-değişkenler, VBScript ile yapacağımız işlemlerde önemli bir yer tutar. Örneğin bir sınıftaki öğrencilerin veya müşterilerimizin listesi bir dizi değişkenin elemanları olabilirler. Yapmak istediğimiz işlem, dizi-değişkenin bütün elemanları için tekrar edilecekse, For Each..Next döngüsü daha elverişli olabilir. Bir dizi-değişkenin eleman sayısı ilerde değişirse ve siz döngüyü For..Next ile kurmuşsanız döngünün sayacı için verdiğiniz için alt ve üst sınırı değiştirmek zorunda kalırsınız. Oysa For Each, kaç kere tekrar edeceğine ilişkin değeri her zaman dizi-değişkenin elemanların sayısından alır. Örneğin, bütün öğrencilerin listesini tutan Ögrenciler dizi-değişkeninin bütün elemanlarının değerini ekrana yazdıralım:

```
For Each Ogrenci In Ogrenciler
Response.Write Ogrenci
Next
```

Burada “Ogrenci” Ogrenciler dizi-değişkeninde döngünün her adımında okunan bir elemanın değerini tutar. For Each döngüsü tarafından “okunmakta olan” dizi-değişkenin her bir değeri sırayla bu değişkene yazılacaktır. Bunu bir tür endeks değişken olarak düşünebilirsiniz.

Döngüyü durdurmak isterseniz

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. Örneğin:

```
For sayac = 1 to 10
            [..bir takım işler yap..]
            If Degisken1 > Degisken 2 Then Exit For
            [..bir takım işlere devam et..]
Next
```

Bu durumda For..Next döngüsü, Degisken1'in değerinin Degisken2'den yüksek olduğunu belirlerse, derhal döngüyü durdurarak, Next'ten sonraki satıra gidecektir.

Do döngüsünden ise Exit Do ile çıkababiliriz. Bu ifadenin kullanımı da Exit For gibi olur.

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 türlü olur: fonksiyon (Function) ve Subroutine (sab-rutin okunur; kısaca Sub diye yazılır ve sab diye okunur). 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.

Biraz karışık oldu; ama bunu ilerde gerçek ASP uygulamaları yazarken daha iyi anlayacağız. Şimdilik bir iki noktayı öylece öğrenmeye bakalım. 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:

```
<%
Sub YaziTura()
Dim ParaAt
Randomize
ParaAt = Int(Rnd * 2) + 1
If ParaAt = 1 Then
Response.Write "Yazı"
Else
Response.Write "Tura"
End If
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:

```
Hesapla 10, 20
```

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

Sık Kullanacağımız Hazır Fonksiyonlar

VBScript'te kullanabileceğimiz bir iki hazır-fonksiyona yukarıda değindik. Tesadüfî sayı üreten Rnd() fonksiyonu bunlardan biriydi; ayrıca Int() fonksiyonu ile kesirli bir sayının tam bölümünü alabildiğimizi gördük. VBScript'in kullanılmaya hazır daha bir çok fonksiyonu vardır; ancak ASP uygulamalarında sık kullanacağımız ve özellikle metin düzenlemeye ait olan bir kaçını burada sıralayalım.

Tarih 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. Örneğin,

```
<%= 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.

Karakter-dizisi Düzenleme

Karakter-dizisi veya String, VBScript için herşey olabilir. "Sana Sevdanın Yolları Bana Kurşunlar" bir String'dir. "Bugün 7 Ağustos 2000" bir String'dir. "Doğum Günün Kutlu Olsun!" bir String'dir. Web sitemizi ziyaret eden kişinin formlarımıza yazacağı ve Gönder tuşunu tıklayarak Server'a göndereceği bilgiler String'dir. Fakat bunların hepsi olduğu şekliyle işimize yaramaz. Bunları yeniden düzenlemek, içinden seçmeler yapmak veya biçimlerini değiştirmek gerekebilir. VBScript bu amaçla kullanılmaya hazır bir dizi fonksiyon verir:

InStr — Uzun bir String'in içinde vereceğiniz daha kısa bir String'in bulunup bulunmadığını arar; bulursa bu kısa String'in başlama noktasının değerini verir. Diyelim ki, "Sana Sevdanın Yolları Bana Kurşunlar" String'ini Kayahan değişkenine, "Sevda" kelimesini de Ara değişkenine atadınız. InStr fonksiyonu ile Ara'nın değerinin yerini Kayahan'ın değerinin içinde bulabilirsiniz:

```
Yer = InStr(Kayahan, Ara)
```

Yer'in değeri 6 olacaktır; çünkü "Sevda" kelimesi, uzun String'de 6'ncı karakterden başlamaktatır.

Len Bir String'in uzunluğunu belirler. Yukarıdaki örnekte yer alan Kayahan değişkenin uzunluğunu şöyle belirleyebiliriz:

```
Uzunluk = Len(Kayahan)
```

Uzunluk değişkeninin değeri 36 olacaktır.

UCase Vereceğiniz bir String'in tüm karakterlerini büyük harfe çevirir.

```
YeniString = UCase(Kayahan)
```

Yeni String'in değeri: "SANA SEVDANIN YOLLARI BANA KURŞUNLAR" olacaktır.

LCase Vereceğiniz bir String'in tüm karakterlerini küçük harfe çevirir.

```
YeniString = LCase(Kayahan)
```

Yeni String'in değeri: "sana sevdanın yolları bana kurşunlar" olacaktır.

LTrim, RTrim, Trim Verdiğiniz String'in (sırasıyla) solunda yani baş tarafında; sağında yani sonunda ve hem başında ve hem de sonundaki boşlukları temizler.

Space İçinde, vereceğiniz sayı kadar boşluk olan boş bir String oluşturur. Örneğin

```
Bosluk = Space(20)
```

Bosluk değişkenin değeri " " (20 boşluk) olacaktır.

String İstediğiniz sayıda ve istediğiniz bir karakterle bir String oluşturur.

```
YeniString = String(3, "*")
```

YeniString değişkeninin değeri "***" olacaktır.

Left, Right Bir String'in içinde soldan (baştan) veya sağdan (sondan) verdiğiniz sayıya kadar olan karakterleri verir. Örneğin, yine yukarıdaki Kayahan değişkenini kullanırsak:

```
Solda = Left(Kayahan, 4)
```

Solda değişkeninin değeri "Sana" olacaktır; çünkü Kayahan değişkeninin soldan itibaren dört harfi "Sana" kelimesine denk geliyor.

Mid Bir String'in içinde başlangıç noktasını ve karakter olarak boyunu verdiğiniz alanda yer alan String'i verir.

```
Ortada = Mid(Kayahan, 5, 8)
```

Ortada değişkeninin değeri "Sevdanı" olacaktır; çünkü Kayahan değişkeninin soldan 5'nci değişkeninden itibaren 8 karakterlik alanda "Sevdanı" karakterleri yer alıyor.

Dizi-Değişken (Array) Fonksiyonu

VBScript'in dizi-değişken oluşturmada Array() fonksiyonu ile sağladığı kolaylıklara kısaca değindik. Fakat Array ile daha bir çok iş yapabiliriz; ve dizi değişken oluşturmakta VBScript'in diğer bazı kolaylıklarından yararlanabiliriz. Dizi-değişkenler, özellikle Web ziyaretçilerimizden gelecek bilgilerin kaydedilmesinde; veritabanından çekeceğimiz verilerin kullanılır hale getirilmesinde yararlı bir araçtır. Dolayısıyla ASP sayfalarınızda sık sık çok-boyutlu dizi değişkenlerden yararlanacaksınız. Bunun için gerekli araçları kısaca ve topluca ele almamız yerinde olur.

Bir dizi değişken oluştururken, değişkenin eleman sayısını belirtmezsek, VBScript, kendi kendine "Anlaşılan bu diziyi dinamik yapmamı istiyorlar!" der; ve daha sonra elemanlarının değerleri sonradan belirtilebilecek ve eleman sayısı sonradan arttırılabilecek bir dinamik dizi-değişken oluşturur. ("VBScript iyi huyludur," demiş miydim?) Örnek:

```
Dim Ogrenciler()
```

Bu komutla, Ogrenciler dizi-değişkeni oluşturulur; ancak eleman sayısı belirtilmediği için dizi dinamiktir; yani daha sonra bu dizinin eleman sayını belirleyebilirsiniz. Bunu:

```
ReDim Ogrenciler(15)
```

gibi bir komutla yapabiliriz. Şimdi aklınıza şu soru gelebilir: Peki neden Ogrenciler dizisini baştan eleman sayısını belirterek tanımlamıyoruz? Güzel soru! Cevabı şu olabilir mi?

Dizi-değişkenimizin eleman sayısını henüz bilmiyoruz; ve programın akışı içinde bu sayı, başka bir fonksiyonun, Sub'ın veya kullanıcı girdisinin sonucu olarak belirlenebilir. Fakat hemen belirtmek gereken bir nokta var: ReDim komutu, mevcut bir dizi-değişkenin içindeki herşeyi siler! Mevcut dizinin elemanlarını ve onların değerlerini korumak istiyorsak:

```
ReDim Preserve Ogrenciler(20)
```

yazmamız gerekir. Buradaki Preserve (koru) komutu, VBScript'e mevcut dizi içindeki elemanları korumasını, ve eleman sayısını 20'ye çıkartmasını bildirir. Buna neden gerek olabilir? Ziyaretçinin tercihleri değişebilir; örneğin bir elektronik alışveriş sitesinde ziyaretçiniz yeni şeyler alabilir; daha önceki alışverişlerine ilişkin verileri tuttuğunuz dizi-değişkenin eleman sayısını, daha önceki bilgileri silmeden arttırmanız gerekir.

VBScript'in dizi-değişkenlerini tümü aynı adı taşıyan bir liste olarak düşünebilirsiniz; sadece değişken adının yanında dizinin kaçıncı elemanı olduğunu belirten sayı bulunur:

Ogrenciler(1): Necip

Ogrenciler(2): Serap

Ogrenciler(3): Neslihan

Fakat VBScript çok boyutlu dizi değişken de oluşturabilir. İki boyutlu dizi-değişkeni tablo gibi düşünün; dizinin elemanları aynı adı taşıyan değişkenler fakat bu kez sadece tek sayı değil sıra ve sütun numaraları ile belirleniyorlar:

Ogrenciler(1,1): Necip

Ogrenciler(1,2): Serap

Ogrenciler(1,3): Neslihan

Ogrenciler(2,1): Selim

Ogrenciler(2,2): Murat

Ogrenciler(2,3): Merve

Ogrenciler(3,1): Elif

Ogrenciler(3,2); Hande

Ogrenciler(3,3): Leyla

Şimdi, burada üç sıralı, üç sütunlu bir tablo getirebilirsiniz gözünüzün önüne. Bu dizi-değişkeni şu komutla oluşturabiliriz:

```
Dim Ogrenciler(3,3)
```

Böyle bir değişkende sözgelimi birinci sıra (numarası 1,x olanlar) çalışkanları, ikinci sıradakiler (2,x'ler) daha az çalışkanları vs., belirtebilir. VBScript, üç, dört ve hatta beş boyutlu dizi-değişken oluşturur. Ama bunu nerede kullanacağınızı siz kararlaştırabilirsiniz.

Bir dizi-değişkenin herhangi bir elemanın değerini, programın herhangi bir aşamasında değiştirebilirsiniz:

```
Ogrenciler(3,2) = "Caner"
```

komutu, Hande'nin adını siler ve yerine Caner'in adını yazar.

Dizi-değişkenlerimizin eleman sayısını bilmek isteyebiliriz. Kimi zaman dizi-değişkenlerimizin eleman sayısı biz belirlemeyiz; bu bilgi bir formdan gelebilir; bir veritabanından alınabilir; fakat mesela bir döngü için bu değişkenin kaç elemanı olduğunu bilmek gerekir. Örneğin elimizde 35 elemanı olan Ogrenciler dizi-değişkeni varsa, bu sayıyı

```
ElemanSayisi = UBound(Ogrenciler)
```

komutu ile ElemanSayisi değişkenine yazdırırız. ElemanSayisi'nin değeri bu durumda 35 olacaktır.

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.

***

Şimdi ASP için gerekli VBScript bilgisi bu kadar değil tabiî. Ama bu, işe başlamamız için yeter. İlerde yeni VBScript komutları, veya fonksiyonları ile karşılaşırsak (ki size garanti ederim, karşılaşacağınızı!) onları orada kısaca ele alırız. Ama yukarıdaki VBScript bilgisi, her zaman lazım!

ASP'nin Nesneleri

Masa, nesnedir. Notebook bilgisayarınız da nesnedir! Arkadaşınız da bir nesnedir. Tabiî bunu onun yüzüne karşı söylemeseniz, iyi olur. Her nesnenin özellikleri vardır. Masa serttir. Su, içine konulduğu nesnenin biçimi alır.

ASP tekniğinde de amacımız nesnelerin özelliklerini kullanarak, ya bu özellikleri belirlemek, ya da değiştirmektir. Nesne Yönelimli Programlama (Object Oriented Programming, OOP) kavramını daha önce çok duymuş olmalısınız. Bir önceki bölümde fonksiyonlarla, Sub'larla, değişkenlerle tanıştık. Diyelim ki öğretmensiniz (ah, nerede o günler!) ve ASP programınızda her öğrencinin notunu veritabanına işleyen, veritabanından notları alarak geçeni-kalanı belirleyen veya öğrencilerle ilgili daha yapılması gereken bir çok işi yapan fonksiyonlarınız ve Sub'larınız var; bunların kullandığı bir çok değişkeniniz var: demek ki sizin ASP programınızda "öğrenci" diye bir nesneniz var. Ve siz bu nesneye yönelimli program yapmışsınız!

Her "program nesnesi" iki unsura sahiptir:

Özellik (Property, Attribute): Bir nesnenin özellikleri, onun değişkenleridir. "Öğrenci" nesnesinin "Öğrencinin Adı," "Notları," "Adresi" gibi değişkenleri, yani özellikleri vardır.

Metod (Method): Bir nesnenin işlemesi, çalışması için, kısaca kendisinden bekleneni yerine getirebilmesi için çalışma yöntemlerine ihtiyacı vardır. Dolayısıyla bir ASP nesnesinin fonksiyonları, onun metodlarıdır.

Fakat ASP'de nesneler sadece sizin öbekler halinde toplayacağınız fonksiyonlar ve değişkenlerden ibaret değildir. Bir kere, ASP programında kullandığınız Script dilinin getirdiği nesneler vardır. ASP sayfasını Javascript ile yazarsanız başka, VBScript ile yazarsanız başka dil nesnelerine sahip olursunuz; ancak her ikisinde de ortak olan

"Scripting" nesneleri vardır. Bunlara birazdan ayrıntılı ele alacağız. Sonra Web Server'ın size hazır sunduğu nesneler vardır. Bunları daha sonraki bölümde göreceğiz. Ve tabiî, Browser'ın bir HTML sayfasının bölümlerini nesne sayarak oluşturduğu nesneler vardır. Bunlara da diğer nesneleri ele alırken sırası geldikçe değineceğiz. (Tabiî bir de ASP programınızı Javascript ile yazarsanız, VBScript'ten farklı olarak kendisi nesne-yönelimli bir dil olan Javascript'in oluşturmanıza imkan vereceği nesneler vardır. Fakat bu nesneler, bu kitapçığın kapsamı dışında kalıyor.)

Nesneler nasıl oluşmuş olursa olsunlar, daima size bir değer verirler:

```
Nesne.Özellik = Değer
```

Bir nesnenin bir özelliğinin değeri, bizim için bir değişken değeri gibi önem taşır:

```
If Nesne.Özellik > Değer Then ...
```

Nesnelerin özelliklerinin değerlerini değişkenlere atayabiliriz; ancak bunu yaparken Nesne'nin bir metoduna (fonksiyonu) göndermede bulunmamız ve gerekiyorsa bu fonksiyona kullanması için veri göndermeliyiz (bir fonksiyona kullanması için gönderilen değere argüman/argument denir):

```
Degisken = Nesne.Metod(argüman1, argüman2...)
```

Daha sonra bu değişkeni istediğimiz yerde kullanırız. Bu kadar teoriden sonra bir örnek nesne oluşturursak, belki işin içinden daha kolay çıkabiliriz.

Nesneler, diğer yararlarının yanı sıra, birbiri ile ilgili Sub'larımızı, fonksiyonlarımızı ve değişkenlerimizi birarada tutmamızı sağlar. Şimdi VBScript ile bir nesne oluşturabiliriz:

```
<%
Class Ogrenci
            Public Adi, Soyadi, No
            Function AdiSoyadi
            AdiSoyadi = Adi & " " & Soyadi
            End Function
```

```
End Class
%>
```

Bir dakika! Nesne (object) adını verdik fakat Class (sınıf) oluşturduk! Bu grubun Object..End Object olması gerekmiyor mu? Belki haklısınız; ama bir "sınıf" ancak VBScript tarafından kullanılmaya başlanırsa Nesne olur. Dolayısıyla biz "sınıf" yazarız; VBScript onu Nesne haline getirir. Peki, nesneyi oluşturduk; şimdi bunu nasıl kullanacağız. Çok kolay; önce bu nesneden yeni bir olgu (instance) oluşturacağız; sonra.. Ama önce bunu nasıl yapacağımızı yazalım:

```
<%
Dim Ogr1
Set Ogr1 = New Ogrenci
Ogr1.Adi = "Necip"
Ogr1.Soyadi = "Fazıl"
Ogr1.No = "181"

Response.Write Ogr1.AdiSoyadi
%>
```

Her nesne, New (yeni) komutu ile yeni bir değişkene bütün özelliklerini verir. Burada Ogr1 değişkeni, yukarıda oluşturduğumuz Ogrenci nesnesinin bütün özelliklerini kazanmış oluyor. Ogrenci nesnesinin ".Adi", ".Soyadi" ve ".No" özellikleri olması gerekir; nitekim Ogr1'e bu özelliklerini burada veriyoruz. Ogrenci nesnesinin bir de metodu (fonksiyonu) var; Ogr1 bunu da kazanabilir mi? Tabiî; hem kazanır, hem de bunu ziyaretçinin Browser penceresine yazdırabilir.

Nesne kavramını kavramak zordur. Fakat yukarıdaki örnekte görüldüğü gibi, ASP sayfalarınızda nesne oluşturarak çalışmaya alıştığınız zaman bunun sağladığı kolaylıktan kolayca vazgeçmeyeceksiniz.

////////////////KUTU//////////////////////

Hata (Err) Nesnesi

Hangi dille olursa olsun program yazarken hata yapmak kaçınılmaz bir kuraldır. Dolayısıyla kullandığınız programlama dili hatalarınızı kolayca yakalamanıza imkan vermelidir.

ASP programlarınızda yazım yanlışlığı, olmayan değişkene gönderme gibi Script hatası olmaması gerekir. Bu tür hatalar, program Web'e gönderilmeden mutlaka ayıklanmalıdır. Fakat programcı olarak öngöremeyeceğiniz, ve çoğu Web ziyaretçisinden veya ziyaretçinin bilgisayarından kaynaklanan hata durumları olabilir. VBScript, şu standart komutla beklenmedik hata durumlarında programın yoluna devam etmesini sağlayabilir:

```
<% On Error Resume Next %>
```

Bu komutla VBScript'e, hata halinde bir sonraki satırdan yoluna devam edecektir. Fakat oluşan hata, programın daha sonra vermesi beklenen sonucu vermesini önleyebilir. VBScript, Err (Hata) Nesnesi'nin bir çok özelliğinden özellikle hata sayısı (Number), tanımı (Description) ve kaynak (Source) özellikleri ile size hatanın ne olduğunu ve nereden kaynaklandığını söyleyebilir. Bu özellikleri kullanarak, programlarınızda, en azından geliştirme aşamasında, örneğin,

```
If Err:Number = xx Then
```

şeklinde bir ifade ile hatanın türüne göre programın kazasız yürümesini sağlayabilirsiniz. Burada xx yerine 108 ayrı hata numarası yapabilirsiniz. Hata numaraları, Microsoft'un VBScript sitesinden edinilebilir.

////////////////////////////////KUTU BİTTİ///////////////

Nesneler hakkında aşağı yukarı bir fikir edindiğimize göre, şimdi kullanılmaya hazır Nesne'lerden başlayarak, VBScript kullanırken, ASP sayfalarımızda yararlanabileceğimiz Nesne'leri ele alabiliriz.

Dosya Sistemi Nesnesi

Dosya Sistemi Nesnesi (FileSystemObject), ASP programının, Web Sunucusunun sabit disk sisteminde, sürücüleri, klasörleri ve dosyaları yönetmekte kullanacağımız temel araçtır. Burada, ne denli güçlü bir araçtan söz ettiğimizi hemen görebilmek için şu kodu yazalım ve dosya_yaz.asp adıyla kaydedelim:

```
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP ILE DOSYA YAZMA</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>
<%
Dim YaziFSO, yaz
Set YaziFSO = CreateObject("Scripting.FileSystemObject")
Set yaz = YaziFSO.CreateTextFile("c:\yazi_deneme.txt",True)
yaz.WriteLine("Bu bir denemedir.")
yaz.Close
%>
<H2><CENTER>Bu Web sayfası sabit diske yazı yazdırır!!
<BR>Şimdi C: sürücüsünde yazi_deneme.txt adlı bir dosya olması gerekir!
<BR>Lütfen bakar mısınız?</H2></CENTER>
</BODY>
</HTML>
```

Kodumuzun Dim satırında iki değişken belirlediğimizi görüyorsunuz. Fakat bu iki değişkeni sistem nesnesi olan Scripting'in yeni bir olgusu olarak kullanacağımız için daha önce standart değişkenlere değer atadığımız gibi değil, fakat Set komutundan yararlanıyoruz, ve YaziFSO değişkeninde bir "Scripting.FileSystemObject" nesnesi

oluşturulmasını sağlıyoruz. (ASP uzmanları arasında gelenek, nesne değeri tutan değişkenlere, ilgili nesnenin baş harflerini eklemektir. Böylece bir değişkenin adına bakarak, işlevini anlamak mümkün olur.)

"yaz" değişkeni YaziFSO'da yeni bir olgusunu oluşturduğumuz FileSystemObject'in CreateTextFile (Düzyazı dosyası oluştur) metodunu kullanıyoruz; bu metod oluşturulacak dosyanın adını ve eğer bu dosya varsa üzerine yazılmasına izin veren True (doğru) veya buna izin vermeyen False (yanlış) kelimesini argüman olarak alır. "yaz" değişkeni şimdi kendisi bir metod kullanabilecek şekilde, FileSystemObject'in bir örneğidir; nitekim WriteLine metodu ile biraz önce oluşturulan dosyaya, argüman olarak verdiğimiz metni yazdırmaktadır. Bu kodu çalıştırdıktan sonra, sabit diskinize bakarsanız, düzyazı dosyasını göreceksiniz:

<asp0009.tif>

Bugüne kadar Web tekniği olarak Browser'da bir sayfayı görüntüleyebileceğimizi sanıyor idiysek, sistem nesneleri kullanarak çok daha farklı şeyler yapabileceğimizi görmüş olduk. Şimdi FileSystemObject'i daha yakından tanıyabiliriz.

Bu nesne bize sabit diske erişme ve onun kaynaklarını kullanma imkanı verir. Bütün nesneler gibi kullanılabilmesi için önce bir değişkenin bünyesinde oluşturulması gerekir:

```
<%
Dim DosyaSistemi
Set DosyaSistemi = CreateObject("Scripting.FileSystemObject")
%>
```

Dosya Sistemi Nesnesi'nin 20'den fazla metodu vardır; fakat bunlardan önemlileri şöyle sıralanabilir:

CopyFile (dosya kopyala), MoveFile (Dosya taşı), CopyFolder (klasör kopyala), MoveFolder (klasör taşı), Create Folder (klasör oluştur), DeleteFile (dosya sil), DeleteFolder (klasör sil).

Şimdi bunlardan birinin nasıl kullanılabileceğine bir örnek verelim:

```
<%
Dim DosyaSistemi
Set DosyaSistemi = CreateObject("Scripting.FileSystemObject")
DosyaSistemi.DeleteFile "c:\belgelerim\test.*"
%>
```

Bu program ile "Belgelerim" klasöründeki "test" isimli bütün dosyaları silmiş olursunuz. (Bu programı çalıştırmadan önce bir kaç kez düşünmeniz, sanırım iyi olur; çünkü ASP yoluyla sildiğiniz dosyalar, Geri Dönüşüm Kutusu'na gitmez!) Daha az zararlı bir diğer örnek ise şöyle olabilir:

```
<%
Dim DosyaSistemi
Set DosyaSistemi = CreateObject("Scripting.FileSystemObject")
DosyaSistemi.CopyFile "c:\belgelerim\*.*", "c:\yedekler\"
%>
```

Bu program "Belgelerim" dizinindeki bütün dosyaları "Yedekler" dizinine kopyalar.

FileSystemObject'in sadece bir özelliği (Property) vardır: Drives (sürücüler). Fakat bu özellik, bir değil bir çok elemandan oluşan bir dizi-değişken gibi Kolleksiyon (Collection) sayılır. Nede? Çünkü bir Web Server'da birden çok sürücü bulunur. Her sürücü, bu kolleksiyonun üyesidir (FileSystem.Drives) ve her birinin sürücü harfi (.DriveLetter), disk adı (.VolumeName), byte olarak boş alanı (.FreeSpace) özellikleri vardır. suruculer.asp adıyla kaydedeceğiniz şu program, denediğiniz sistemin disk-disket-CD-ROM durumunu size listeleyecektir.

```
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP ILE SÜRÜCÜ KOLLEKSİYONU</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>
<%
Dim DosyaSistemi, Surucu, Suruculer
Set DosyaSistemi = CreateObject("Scripting.FileSystemObject")
Set Suruculer = DosyaSistemi.Drives
For Each Surucu In Suruculer
%>
<b>Sürücü:</b> <%=Surucu.DriveLetter%><br>
<% If Surucu.IsReady = True Then%>
<b>Disk Adı:</b> <%=Surucu.VolumeName%><br>
<b>Boş alan:</b> <%=Surucu.FreeSpace%><br>
<% Else %>
<i>Sürücü hazır değil!</i><br>
<% End If
Next %>
</BODY>
</HTML>
```

Burada özelliklerini ve metodlarını DosyaSistemi adlı değişkene atadığımız Dosya suistemi Nesnesi'nin sürücüler kolleksiyonunun dizi-değişken gibi olduğunu söylemiştik. For..Next akış kontrolü ile bu kolleksiyonun bütün üyelerinin sırayla sürücü harfi, ve hazırsa disk adı ve boş alanı bilgilerini alıyoruz. Drives kolleksiyonunun diğer özellikleri arasında toplam yüzey genişliği (TotalSize), sürücü türü (DriveType; 0=bilinmiyor; 1=çıkartılabilir; 2=sabit; 3=ağ; 4=CD-ROM; 5= RAM-Drive), ve dosya sistemi (FileSystem; FAT, NTFS, CDFS), kök dizin (RootFolder) vardır. Bu program bir PWS'da çalıştığında, şu sonucu alıyoruz:

<asp0010.tif>

VBScript açısından, her sürücüde klasörler (Folders) ve onların içinde alt-klasör (Subfolders) ve dosya (Files) kolleksiyonları bulunur. (Her klasörün içinde içinde bir alt-

klasör nesnesi bulunduğu için ASP ile sonsuza kadar bütün klasörlere gönderme yapabilirsiniz. Klasör nesnesinin bazı özellikleri şunlardır:

Adı (Name), oluşturulma (DateCreated), erişim (DateLastAccessed), değiştirme (DateLastModified) tarihleri, içindeki dosyalar ve alt-klasörlerdeki dosyalarla birlikte boyutu (Size), bulunduğu sürücü (Drive), içinde bulunduğu klasör (ParentFolder), alt-klasörler (SubFolders), kök dizin olup olmadığı (IsRoot).

Klasör nesnesinin kopyala (Copy), sil (Delete) ve Taşı (Move) metodları vardır.

Dosya (File) nesnesinin de ad, oluşturma, erişim, değiştirme, boyut, sürücü ve içinde bulunduğu sürücü özellikleri, ve kopyala, sil, taşı metodları vardır.

PWS'ınızın bulunduğu sistemde, söz gelisi C: sürücüsünün kök dizinindeki bütün dosyaların listesini size veren bir kod şöyle yazılabilir (dosyalar.asp):

```
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP ILE KLASOR - DOSYA KOLLEKSİYONU</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>
<%
Dim DosyaSistemi, Surucu, Dosya, KokDizin, KokDosyalar, DosyaNesnesi
Dim SurucuHarfi
SurucuHarfi = "C:"
Set DosyaSistemi = CreateObject("Scripting.FileSystemObject")
Set Surucu = DosyaSistemi.GetDrive(SurucuHarfi)
Set KokDizin = Surucu.RootFolder
Set KokDosyalar = KokDizin.Files
For Each DosyaNesnesi In KokDosyalar
%>
<%=DosyaNesnesi.Name%><br>
```

```
<% Next %>
</BODY>
</HTML>
```

Burada SurucuHarfi değişkenin değerini değiştirerek, arzu ettiğiniz disk/disket veya CD-ROMa ulaşabilirsiniz. GetDrive metodu dikkatinizi çekmiş olmalı; bu metodla, VBScript, fiilen disk/disket sistemine erişir.

Daha sonra yapacağımız ASP sayfalarında dosya sistemi nesnesinden ve disk sürücüleri kolleksiyonundan yararlanacağız.

Metin (TextStream) Nesnesi

Dosya sistemi nesnesi bize disk sistemine, klasörlere ve dosyalara erişme imkanı verir ama yeni dosyaları oluşturmak veya mevcutlara ek yapmak için yeterli özellik ve metoddan yoksundur. Bunu TextSream nesnesi sağlar.

Bilgisayar işletim sistemlerinin anası, Unix'e aşina iseniz, klavyeden sabit diske kadar bir bilgisayara girdi akımı bulunduğunu, bunlardan birinin de metinler olduğunu hatırlayacaksınız (Aşina değilseniz, şimdi oldunuz!). Bir işletim sistemi, metin dosyalarını okurken, yazarken bir metin akışı olur; TextStream nesnesinin adı da bunu anlatıyor: Metin Akımı. Web Server ve dolayısıyla ASP açısından sabit diske bir metin yazarken, veya sabit diskten bir metin okurken, bir metin akışı nesnesi oluşur. Bu nesnenin özellikleri ve metodlarını kullanarak, örneğin ziyaretçilerin sitemize bırakacakları form bilgilerini Web Server'ın sabit diskine yazdırabiliriz. Veya mevcut metinleri okuyabilir ve bunların içeriğini ziyaretçimize göndereceğimiz HTML sayfanın etiketlerinin içeriği olarak kullanabiliriz. Metin dosyası okumak ve yazmak disk sistemini ilgilendiren bir eylem olduğu için yine Scripting nesnelerinden FileSystemObject nesnesinden yararlanacağız; fakat bu kez değişik metodlar kullanacağız.

Metin Dosyası Oluşturma (CreateTextFile)

Aslında biraz önce, ASP nesnelerinin neler yapabileceğine örnek olarak yazıp dosya_yaz.asp adıyla kaydettiğimiz program, bir metin dosyasını yazdırma işlemiydi. Oradaki kodlara yeniden göz atarsanız, CreateTextFile (metin dosyası oluştur) metodunu yardıma çağırıyor ve bu metoda argüman olarak yeni metin dosyasının yolunu ve adını veriyorduk. Bu metod TextStream nesnesinindir; ve otomatik olarak bu nesnenin diğer metodlarını kullanmamızı sağlar. kullandığımız metodlar ise WriteLine (satır yaz: bir String'i sonuna yeni satır karakteri koyarak dosyaya yazar) ve Close (kapat: açılan metin dosyasını kapatır).

TextStream'in burada kullandığımız ikisinin dışında iki metodu daha vardır:

Write (yaz): Bir String'i dosyaya yazdırır; satır sonuna yeni batır karakteri (Return kodu) koymaz.

WriteBlankLines (boş satır yaz): Bir metin dosyasına argüman olarak vereceğiniz sayıda boş satır yazdırır.

Varolan Metin Dosyasına Ek Yapma (OpenTextFile)

Metin yazdırma işlerinde sık sık uygulayacağımız bir senaryo, mevcut bir metin dosyasına ek yapmak olacaktır. Örneğin bütün ziyaretçilerimizin sitemizdeki konuk defterine yazdıklarını, bir metin dosyasında toplamak isteyebiliriz.

Bunu OpenTextFile metodu ile yapacağız. Bu metod, tahmin edeceğiniz gibi, açılacak dosyanın yolunu ve adını isteyecektir. Örneğin, dosya_yaz.asp'nin ilgili satırı şöyle olacak:

```
Set yaz = YaziFSO.OpenTextFile("c:\yazi_deneme.txt",8,0)
```

Burada dosya yolunu ve adını veren birinci argümana ek olarak iki yeni argüman görüyorsunuz: "8,0" şeklinde. Bunlardan birinicisi girdi/çıktı durumu (I/O Mode), ikincisi ise biçim (Format) ile ilgilidir. I/O Mode parametreleri şunlardır:

1: okumak için aç

8: eklemek için aç

Açılacak dosyanın biçimini belirttiğimiz son argüman ise şu değerlerden birini alabilir:

0: ASCII dosyası olarak aç

-1: Unicode dosyası olarak aç (Örneğin içinde Türkçe karakterler varsa)

-2: Sistemin varsayılan dosya türü olarak aç

Buna göre, bir dosyayı salt okumak için açmak amacıyla "1,0" argümanlarını kullanmamız gerekir. Diyelim ki bir dosyayı açtık. İçindekileri nasıl okuyabiliriz? Bir döngüyle. İşte örneği:

```
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP ILE DOSYADAN METİN OKUMA</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>
<%
Dim DosyaSistemi, MetinDosyasi, Satir
Set DosyaSistemi = CreateObject("Scripting.FileSystemObject")
Set MetinDosyasi = DosyaSistemi.OpenTextFile("c:\yazi_deneme.txt",1, 0)
Do
Satir = MetinDosyasi.ReadLine
%>
<%=Satir%>
```

```
<%
Loop Until MetinDosyasi.AtEndOfStream
MetinDosyasi.Close
%>
</BODY>
</HTML>
```

Burada kullandığımız metod dikkatinizi çekmiş olmalı: ReadLine. Bu, açılan metin dosyasından bir satır okutmamızı sağlar. İkinci ve son satırları okutmamızı ise Do..Loop kontrolü sağlıyor. Bu döngü ne kadar sürüyor? MetinDosyası nesnesi, AtEndOfStream (akımın sonunda) oluncaya kadar. Bu, dosyanın sonuna geldiğimiz anda ortaya çıkan bir durum; bu durumla karşılaştığı anda Do..Loop, yaptığı işi durdurup, bir sonraki komuta geçecektir. Burada kullandığımız ReadLine metoduna ek olarak yararlanabileceğimiz diğer metodlar ise şunlardır:

Read (oku): Bir sayı örgümanı ile çalışır ve verdiğiniz sayı kadar karakter okur.

ReadLine (satır oku): Bir satır okur ve String olarak verir.

ReadAll (tümünü oku): Bütün satırları okur ve tek String olarak verir.

Skip (atla): Bir sayı argümanı ile çalışır ve verdiğiniz sayı kadar karakteri atlar.

SkipLine (satır atla): Bir sonraki satıra atlar.

Bu metodlarla sağladığımız okuma işinin kontrolü amacıyla şu özellikleri de kullanabiliriz:

AtEndOfStream (akımın sonunda): Okutulan dosyanın sonuna gelinmesi halinde True (doğru) olur.

AtEndOfLine (satırın sonunda): Okutulan satırın sonuna gelinmesi halinde True (doğru) olur.

Sunucu (Server) Nesneleri

Buraya kadar ele aldığımız nesneler bir anlamda bizim sadece tek tek sayfalarda yararlanacağımız araçları sağlıyor. Oysa ASP'yi diğer CGI teknolojilerinden ayıran başlıca özelliklerden biri tek tek Web sayfalarını sanki bir bilgisayar programının çeşitli pencereleri, diyalog kutuları, mesaj kutuları, girdi kutuları gibi, birarada bir "uygulama programı" olarak bağlayabilmesidir. Başka bir deyişle bize, ziyaretçinin sitemize bağlandığı ve ana sayfamızı açtığı andan itibaren sitemizin bir program bütünlüğünde çalışmasını sağlayacak araçlar gerekir. Bunu ancak Web Server'ın yardımıyla yapabiliriz.

ASP sayfalarımızda kullanacağımız ikinci grup nesne, Sunucu Nesneleri'dir. Bu grupta tabiî önce Sunucu'nun kendisi yer alır; sonra ziyaretçi ile kurduğumuz ilişki gelir. Ziyaretçi ile ilişkimizi iki yönlü trafiğe benzetebilirsiniz: ondan bize gelen talepler, bizim ona karşılıklarımız.

Ziyaretçiden bize gelen trafiğe neden "Talep" diyoruz? Ziyaretçi, Browser'ının URL hanesine yazdığı her adresle, veya formlardaki bir düğmeyi veya sayfalarımızdaki herhangi bir köprüyü tıklamakla, Server'a "Bana şunu göndersene!" demiş olur. Bu taleptir. Ziyaretçi taleplerinin tümü Talep Nesnesi (Request Object) olarak bir arada ele alınabilir. Server'ın bu taleplere verdiği karşılıklar, yani ziyaretçinin Browser'ına gönderdiği sayfalar, resimler, sesler, videolar ise karşılıktır ve ASP açısından Karşılık Nesnesi'ni (Response Object) oluşturur.

Bu bölümde bu nesneleri daha yakından tanıyacağız.

Server Nesnesi

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 üzünden 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.

Böyle bir şeyi neden yapmak isteyebiliriz? Script'iniz çok karmaşık veya başka bir Server'daki veritabanından veri çekiyor, olabilir. Gerçi bu anlamda 90 saniye bilgisayar milleti 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 görüyorsunuz. CreateObject (nesne oluştur) olmasa idi, dört mevsim birbirini izleyebilir, Dünya Güneş'in etrafında dönebilir miydi? Hiç sanmam. Fakat lütfen o 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 kelimeme2")
```

şeklindeki bir kod Bunu hemen şu şekle sokacaktır:

kelime1+kelime2+kelime3

Talep (Request) Nesnesi

Web Server çok akıllı bir programdır; bir Web ziyaretçisi herhangi bir talepte bulunduğu, yani bir sayfanın gönderilmesini istediği anda, bu talebi, bir nesne halinde ele alır; kolleksiyonlar oluşturur. Bu kolleksiyonlar, HTTP protokolü ile iletişimin sonucu olarak ziyaretçinin Browser'ından ve Internet'e giriş noktası olan ISS'in bilgisayarından başlayan ve Web Server'dan derlenen bir dizi bilgidir. Bir anlamda, Request nesnesi, Web programımızın Girdi (Input) bölümünü oluşturur.

Request nesnesi kendi içinde dört ana nesne barındırır:

QueryString ve Form

Web ziyaretçisinin bilgisayarından kalkıp Server'a gelen herşey, QueryString kolleksiyonunu oluşturur. Bu ziyaretçinin Browser'ın URL adresi hanesine yazdığı bir basit HTML sayfası yolu ve adı olabilir; bir Form'un Gönder düğmesini tıkladığında gelen bilgiler olabilir. Bu bilgilerin şu özelliklerini kullanabiliriz:

Content_Length: Bir Form'dan gelen bilgilerin tümümün byte olarak boyutudur.

Remote_Host: Ziyaretçinin IP adresini verir; ancak Internet'e çevirmeli ağ ile bağlanan ziyaretçilerimiz her seferinde farklı bir IP bildirebilirler. Bu yüzden bu bilgiyi ziyaretçinin kimliği sayamayız.

Request_Method: Form'da kullandığımız GET veya POST metodunu bildirir. İki yöntemle gelen bilgi farklıdır. Form'un oluşturduğu bilgileri GET yöntemi ile alırsak bu, çevre değişkenlerinden QUERY_STRING değişkeninin içine yazılır. Başka bir ifade ile Form'daki bütün değişkenlerin adları ve bu değişkenin içerdiği değer yumak yapılır (bu yumağın niteliğine ve nasıl çözeceğimize geleceğiz!) ve Server'da QUERY_STRING değişkeninin değeri olarak yazılır. Form'un bilgilerini POST yoluyla alıyorsak bunlar Request nesnesinin Form kolleksiyonunun içinde Form'un değişken adları ve ziyaretçinin bu değişkenler için sağladığı değerler olarak ayrı ayrı yazılır. GET ile sınırlı, POST ile sınırsız bilgi alabiliriz.

Script_Name: O anda çalıştırılmakta olan ASP sayfasının adını verir.

ServerVariables (Server Değişkenleri)

Request nesnesinin bir diğer kolleksiyonu, bizim kendi Web Server'ımızın o anda çalışmakta olan ASP sayfası için oluşturduğu ortamın değişkenleridir. Bunların arasında

ziyaretçinin Browser'ına ilişkin bilgiler de vardır. Önvc şU kısa ASP sayfasını çalıştırarak kendi Server'ımızın şu andaki değişkenlerini görelim; sonra bunları ayrıntılı ele alalım (SerDeg.asp):

```
<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 SerDeg.asp 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

Cookie (Çerez)

Daha önce ne kadar çok olursa olsun tek-tek Web sayfalarından oluşan siteler yaptıysanız bile, Cookie (Çerez) ile ilginiz olmamış olabilir. Ya da sadece Internet ziyaretçisi olarak başkalarının size gönderdiiği Cookie'lere sinirlenmekle yetinmiş olabilirsiniz. Fakat şimdi ASP ile Web Programı oluşturmaya hazırlanan kişi olarak Cookie'lerle daha yakından ilgilenmeniz gerekiyor. Çünkü artık siz de başkalarına çerez göndereceksiniz!

Tabiî bunun için önce "Cookie neden gereklidir?" sorusunu yanıtlamak gerekir. HTTP ile yapılan iletişim, belirgin olmayan durum bağlantısına dayanır: yani ne istemci sunucunun, ne de sunucu istemcinin o anda hatta (on-line) olduğunu bilmek zorunda değildir; birbirlerinden istedikleri ve gönderdikleri şeyleri karşı tarafından almaya hazır olduğunu bilmeden gönderirler. Oysa sözgelimi elektronik alışveriş gibi ziyaretçinin bir yerlere birşeyler kaydettiği, geçici değişkenler oluşturduğu durumlarda sitemizde kimin ne yaptığıını bilmek zorundayız. Ziyeretçinin bir sayfada yaptığı tercihler diğer sayfalarda ona sunacağımız içeriği etkileyebilir, belirleyebilir. Oysa aynı ziyaretçi bir sayfadan diğerine

geçerken Server ile ilişkisini kaybedebilir. Bunun için ziyaretçinin Internet'ten kopması gerekmez; sadece TCP/IP protokolü gereği bizimle bağlantısı kesilebilir. Bunu ziyaretçi farketmeyebilir; ama Server etmek zorundadır. Heryeni ilişkiye yeni bir "application" (uygulama programı) başlatamayız; ziyaretçinin bir önceki sayfada yaptığı tercihlerin devam etmesi gerekir. Bu devamlılığı ziyaretçiyi işaretleyerek yaparız; bu işareti de Cookie sağlar.

ASP tekniğiyle tasarladığımız sitede, ziyaretçilerimize Cookie göndermek zorunluktur diyebiliriz. ASP açısından Cookie'ler bir nesne oluştururlar. Aslında, Cookie'ler ASP için iki ayrı grup Cookie nesnesi oluştururlar: verilenler, ve hakkında bilgi alınan varolan Cookie'ler. Birinci grup Request (talep) nesneleri, ikinci grup ise Response (karşılık) neslereni içinde ele alınabilir. Cookie konusunun devamını Response nesnelerine değindiğimiz yere bırakalım.

Sertifika Nesnesi

Sertifika, HTTP bağlantısında "Ben filancayım!" diyen istemcinin, gerçekten filanca olup olmadığını gösterir. Bir yazılımdan ibaret olan sertifikaları yetkili bir kurum veya şirket verir; bir seri numarası olur. Şifreleme teknikleri gelişmiş olduğu için taklit edilmesi zordur. Sertifika uygulaması için Web Server'ın Secure Socket Layers denen güvenli HTTP protokolünü kullanması gerekir. Bu durumda Server'ın URL'i, http:// diye değil https:// diye yazılır.

ASP açısından sertifika ile ilgili her türlü bilgi ClientCertificate kolleksiyonunda durur. Sözgelimi, bir ziyaretçinin gerçekten sitenize girmeye yetkili olup olmadığını anlamak için:

```
<%SertifikaNo = Request.ClientCertificate(SerialNumber) %>
```

gibi bir kodla istemcinin Sertifika seri numarasını SertifikaNo değişkenine atayabilir ve daha sonra bu değişkenin değerini elinizdeki bir liste ile karşılaştırabilirsiniz.

Karşılık (Response) Nesnesi

Ve geldik Web Server'in Çıktı (Output) sağladığı istemciye giden karşılıkların oluşturduğu nesneye. Server'dan Browser'a giden herşey karşılıktır. Bu bir ASP veya HTML sayfası olabilir, sayfanın içindeki GIF, JPG veya PNG grafiği, bir Flash, video veya ses dosyası olabilir. Böylesine zengin içeriği olmakla birlikte Response nesnesinin sadece bir kolleksiyonu vardır: Cookie'ler. Buna karşılık Response nesnesinin kullanabileceğimiz çok sayıda özelliği ve metodu bulunur. Önce Cookie kolleksiyonunu ele alalım; sonra önemli Response nesnesi özellikleri ve metodları üzerinde duralım.

Cookie'ler

Request nesnesinin Cookie kolleksiyonuna değinirken, ziyaretçilerimizi Browser'larına Cookie göndererek tabir yerinde ise işaretleriz, demiştik. Bu işaretleme, söz gelimi, ziyaretçinin bir Form'a yazdığı adı ve soyadı ile elektronik posta adresini Cookie olarak onun bilgisayarına kaydetmek şeklinde olur. Bir kişi sayfamızı talep ettiği anda, ASP programımız bu kişinin bilgisayarında daha önce koyduğumuz Cookie'yi arar ve bulur; Cookie'den bu kişinin adını öğrenir ve mesela sayfamız "Sayın Filanca, sitemize hoş geldiniz!" başlığını görüntüler.

Cookie'ler sadece böyle fiyaka amaçlı olarak kullanılmaz; ziyaretçinin daha önce sitemizde ziyaret ettiği sayfaları veya elektronik ticaret sitemizden satın aldığı kitap türlerini Cookie'ye kaydedebiliriz. Ziyaretçinin ikinci ziyaretinde ona önce bu sayfaların veya kitapların köprülerini sunabiliriz.

Bir Cookie'nin adı ve anahtarları (key) ile bu anahtarlara karşılık değerler olur. Örneğin:

```
<% Response.Cookie("Bizim_Cerez")("Adi_Soyadi")= "Necip Fazıl" %>
```

Bu, ziyaretçinin Browser'ına (yani sabit diskine) "Bizim_Cerez" isimli bir Cookie gönderir; bu Cookie'nin "Adi_Soyadi" adlı bir anahtarı vardır; bu anahtarın değeri ise "Necip Fazıl" olur.

Cookie kolleksiyonunun iki özelliği bulunur:

Expires (zaman aşamı süresi): Bir cookie'nin artık geçersiz olduğu tarihi gösterir.:

```
<% Response.Cookie("Bizim_Cerez").CookieExpires = "August 7, 2000" %>
```

Bu Cookie, 7 Ağustos 2000 tarihinden sonra kullanılmaz olacak demektir.

HasKeys: Cookie'nin String mi içerdiği yoksa anahtarları ve değerleri mi bulunduğunu belirtir, "HasKeys = False" birinci, ""HasKeys = True" ise ikinci durumu belirtir. Bir Cookie'de birden fazla anahtar ve değer bulunuyorsa, tümünü bir ASP kodu içinde yazmak, Cookie'nin gönderilmesini kolaylaştırır

Metodlar

Response nesnesinin bir çok metodu vardır; bunlardan .Write'ı yukarıdaki örneklerde sık sık kulandık:

```
<%
DIM Adi_Soyadi
Adi_Soyadi = "Necip Fazıl Dayanır"
Response.Write("Merhaba, benim adım, " & Adi_Soyadi)
%>
```

örneği, ziyaretçinini Browser penceresine: "Merhaba, benim adım Necip Fazıl Dayanır" yazdırır. Fakat VBScript, size bir kolaylık sağlar; buna bazı ASP tasarımcıları "eşittir metodu" adını verir:

```
<%
DIM Adi_Soyadi
Adi_Soyadi = "Necip Fazıl Dayanır"
%>
<%= "Merhaba, benim adım, " & Adi_Soyadi %>
```

Özellikler

Response nesnesinin bir çok özelliğini kullanarak ziyaretçimize göndereceğimiz sayfaları ve diğer unsurları yönetiriz:

Buffer (Tampon) : True (doğru) olarak ayarlandığında ziyaretçiye gönderilecek sayfanın bütün unsurları bir tampon bölgede toplanır, Script'in çalışması bitinceye kadar beklenir ve HTML sayfa toptan gönderilir. Kimi zaman ASP kodumuz sonuna kadar çalıştığında ziyaretçiyi başka bir sayfaya ve siteye yönlendirebilir. Bu gibi sebeplerle, özellikle çok işlem gerektiren ASP sayfalarının baştarafına bunu sağlayan kodu koymakta yarar olabilir:

```
<%
Option Explicit
Response.Buffer = TRUE
%>
```

Flush (hemen gönder): Buffer metodu sayfanın tümünün Script'in icrası bitmeden gönderilmesini önlerken, Flush bunun tam tersini yapar. ASP, Response nesnesinin Flush metodu kullandığımızı gördüğü anda, o ana kadar icra edilmiş kodun sonucu olan HTML'i Browser'a gönderir:

```
<%
Option Explicit
Response.Flush
%>
```

Clear (Boşalt): Buffer metodu ile Script'in sonunu beklerken geçici bir alanda tutulmakta olan HTML, Clear metodu ile temizlenir, yok edilir. Flush metodunda tampondaki HTML Browser'a gönderilir; ancak Clear metodu tampon bölgedeki herşeyi yok eder. Böyle "tehlikeli" bir metod neden vardır, ve nerede kullanılabilir? Bir çok yerde: ziyaretçinin sözgelimi elektronik alışveriş sitemizde alışverişten vazgeçtiğini belirtmesi

üzerine tampon bölgede tutmakta olduğumuz ve alınan mallar listesini içeren HTML'i bu yöntemle temizleriz:

```
<%
Option Explicit
Response.Clear
%>
```

Expires (Süresi dolar): Kullanıcı tersine bir ayar yapmadıysa, Browser genellikle görüntülediği sayfaları Geçici Internet Dosyaları dizinine (cache) kaydeder ve tekrar aynı sayfayı görüntülemek istediğinizde sayfayı Internet'ten edinmek yerine kendi sabit diskinden alır. Oysa özellikle haber gibi süreli bilgilerin sunulduğu Web sitelerinde bu sitenin itibarını sarsar. ASP tekniğiyle bunu önleyebiliriz. ASP sayfamızda bu sayfanın gözgelimi 60 dakikadan fazla cach dizinde tutulmamasını sağlayacak Expires metodunu kullanabiliriz:

```
<%
Option Explicit
Response.Expires = 60
%>
```

Burada yazdığımız rakamı değiştirerek, sayfanın cach'de tutulacağı dakikayı değiştirebiliriz. "Expires = 0" sayfanın hiç saklanmamasını sağlar.

End (Son): Response nesnesinin o anda icra edilmekte olan Script'i durdurarak, o ana kadar ne elde edilmişse hepsini Browser'a göndermesini sağlayan metodu olan End, aynı zamanda Buffer metoduyla tutulan HTML'in de gönderilmesine yol açar. Bu metoddan sonraki HTML veya ASP kodları icra edilmez:

```
<%
Option Explicit
Response.End
%>
```

Response nesnesinin ASP sayfasının çıktı kontrolünü sağlayan bu metodlarını aşağıdaki eğlenceli Script'le sınayabilirsiniz (ASP ile eğlence de bundan daha fazla olamaz!). Burada Browser'a gitmesi ümidiyle üç cümle var. Programı çalıştırın ve bakalım hangisi ekranda kalacak? Bu kadar eğlendiğinize göre şu soruyu da yanıtlayabilirsiniz: Neden?

```
<% @LANGUAGE = VBScript %>
<%
Option Explicit
Response.Buffer = True
Response.Expires = 60
%>
<HTML>
<BODY>
Bu 1 Numaralı mesajı mı Browser'a gidecek?
<%
Response.Clear
Response.Expires = 0
%>
<HTML>
<BODY>
Bu 2 numaralı mesaj mı Browser'a gidecek)
<%
Response.Flush
%>
</BODY>
</HTML>
<%
Response.End
%>
Bu 3 numaralı mesaj mı Browser'a gidecek?
</BODY>
</HTML>
```

Uygulama (Application) ve Oturum (Session) Nesnesi

ASP'nin varlık sebebi, standart CGI'ın yetersiz kaldığı noktalardan biri olan Web Server'ın her bir Web ziyaretçiyi oturumunun başından sonuna izleyebilmesi içindir, dersek durumu abartmış olmayız. ASP açısından, bir site "uygulama programı" (Application) sayılır. Her ziyaretçi de bir "oturum" (Session) sayılır. Bir takım ASP ve HTML sayfalarından oluşan bildiğimiz Site'ye application, her hangi bir ziyarete de session denmesinin sebebi nedir? Bunu her iki nesnenin işlevleri ile açıklayabiliriz.

Application nesnesi, sitenin tümüyle ilgili bilgileri (değişkenleri, nesneleri ve metodları) tutar; Session nesnesi ziyaretçinin sitemize girmesinden itibaren izini sürer. Diyelim ki bir borsa sitesi yaptınız; ziyaretçileriniz gelerek, satışa sunulan hisse senetlerinin değerlendirmelerini okuyacak ve size "Şu, şu hisse senetleri al!" diye talimat bırakacak. Bütün ziyaretçilerinizin erişeceği sadece bir veritabanınız var; buna karşılık her bir ziyaretçinin yapacağı farklı tercihler, vereceği farklı kararlar olacaktır. Application nesnesi, sitenizle (artık site yerine Web Uygulama Programı desek de ağzımız alışmaya başlasa!) veritabanına erişmekten tutun, alışverişlerie kadar sitede yapılacak bütün işlerin bütün kurallarını bilecek ve uygulayacak; Session nesnesi ise sözgelimi benim alışverişlerimi, tercihlerimi bilecektir.

HTML ve Javascript ile biraz oynadıysanız, 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. Örneğin:

```
Session ("Tupras") = 44500
```

bütün Session için geçerli bir Tupras değişkeni oluşturur ve ona "44500" değerini atar. Kimi zaman, değişkenin çok daha geniş kapsamlı olmasını, yani ömrünün Session ile değil bütün Application boyunca belirli olmasını isteyebiliriz. O zaman bu değişkeni Application düzeyinde tanımlayabiliriz:

```
Application ("Tupras") = 44500
```

Bu durumda Tupras değişkeni bütün ziyaretçiler için aynı değere sahip olacakatır.

Session nesnesinin oluşabilmesi için, ziyaretçiye mutlaka bir Cookie göndererek, sitemizde (hani "Uygulama Programı" diyecektik?) bir işaret vermemiz gerekir. Daha önce, HTTP ile kurduğumuz bağlantı, belirsiz durum bağlantısıdır demiştik. Bu, Server'ın bir ziyaretçiye arzu ettiği sayfayı gönderdikten sonra, onu alıp almadığını, o sayfada ne tercihler yaptığını bilmemesi demektir. Oysa, ziyaretçiye sitemize bağlandığı anda bir Session kimliği verirsek ve her yeni sayfa talebinde bu kimliği kontrol edersek, kimin hangi oturumunu sürdürdüğünü biliriz. ASP–uyumlu bir Web Server, ziyaretçi yeni bir tercih yapmadığı taktirde her Session nesnesini 20 dakika açık tutar; sonra siler. Bu süreyi Session nesnesinin Timeout özelliği yoluyla değiştirebilirsiniz. Session belirleyen Cookie ASP-uyumlu Web Server tarafından otomatik olarak gönderilir ve takip edilir; tasarımcı olarak bizim bu konuda bir şey yapmamız gerekmez.

Bir Web programınıza aynı anda kaç kişi ulaşırsa (yani sayfalarınızı kaç kişi talep ederse), o kadar Session nesnesi oluşur; fakat siteniz bir adet olduğuna göre bir adet Application nesnesi vardır. Bu nesnenin bütün Session'lar için sitemizin ihtiyaçlarına uygun ve aynı uygulama kurallarına sahip olmasını sağlayan bir dosya vardır: Global.asa. Bu dosya PWS veya IIS kurulurken oluşturulur. ASP ile Web programlarınızı, örneğin MS Visual

Studio ile oluşturuyorsanız, program sizin için seçtiğiniz dizinde bir Global.asa dosyası oluşturacaktır. Bu dosyada, çoğu zaman, sitemize ilk ziyaretçinin gelmesiyle oluşan Application_OnStart ve son ziyaretçinin çıkmasıyla oluşan Application_OnEnd ile herhangi bir ziyaretçinin bir sayfaya erişmesiyle oluşan Session_OnStart ve ziyaretçinin sitemizden çıkması ile oluşan Session_OnEnd olayları halinde ne yapılacağı yazılıdır. Bu dosyanın içeriği standart bir ASP dosyasına benzemekle birlikte adındaki uzatmanın .asp değil de .asa olmasının sebebi, dosyanın Active Server Application dosyası olmasıdır. ASP-uyumlu bir Web Server programı sitemize ulaşan ilk ziyaretçiyi gördüğü anda Global.asa dosyasını çalıştırır.

Application ve Session nesnelerin kendi başlarına en çok kullanıldığı yer, sitemize gelen ziyaretçilerin sayısını (sitemizin aldığı Hit sayısını) tutmasını sağlamaktır. Bu genellikle Global.asa pogramına bir sayaç yerleştirilerek yapılır.

ActiveX Veri Erişim (ADO) Nesneleri

ASP'nin diğer CGI tekniklerine göre kolay olmasının (peki, kabul, "kolay görünmesi" diyelim!) 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, FilePro, 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.

//////////////////KUTU//////////////////

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.

//////////////////////KUTU BİTTİ//////////////////

Şimdi ADO ile aşağıda yapacağımız küçük örnekler için bilgisayarınızda kurulu bir veritabanı programı 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. Bilgisayarınızda veritabanı oluşturma programı yoksa bu kitapçığın kodları arasında bulunan uyeler.mdb adlı MS-Access dosyasını kullanabilirsiniz. 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.

<odbc0001.tif>

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.

<odbc0002.tif>

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"
%>
```

Burada, Server'ın CreateObject metodu ile ADOBD.Connection nesnesini oluşturuyoruz. Oluşturduğumuz bağlantıya istediğimiz değişken adını verebiliriz. Bu örnekte veriye kurduğumuz bu bağlantı Veriyolu adıyla biliyor. Bu yolla sağlayacağımız veriler, ASP programı boyunca bir isimle bilinmelidir. Veriyolunun açacağı veri kümesinin ismini buradaki "Veri_adi" kelimelerinin yerine yazarız. Bu isim, bağlantının .Open metodu ile açacağı verinin adıdır. Bu, kullanacağımız veritabanı dosyasının adı değildir. Bu isim ile söz konusu veritabanı dosyasını işletim sisteminin ODBC aracına tanıtırken kullandığınız isim aynı olmalıdır. Bir veritabanı dosyasını ODBC aracını kullanarak sisteme tanıtma (DSN-Data Source Name) ayarının nasıl yapıldığını daha önce ele aldık. Bu üç satırla, ASP programı, Server'dan ADO aracılığıyla, sistemin "Veri_adi" kelimelerinin yerine yazacağınız isimli veriye yol açacaktır. Örneğin yukarıdaki kutuda oluşturduğumuz ODBC veri kaynağını kullanacağımız zaman, buraya "uyeler" kelimesini yazacağız,

Recordset (Kayıt dizisi)

Veritabanına bağlantıyı oluşturduktan sonra, buradaki tabir yerindeyse ham verileri, kullanılır kayıtlar haline getirmemiz gerekir. Bunu ise ADO'nun Recordset nesnesi sağlar. Kurduğumuz veriyolundan programımıza bilgi gelmesi için .Execute (icra et) metodunu kullanırız; ancak bu komuta icra edeceği bir komut vermemiz gerekir.

Baştan beri ADO ile kullanabileceğimiz veritabanının SQL (sequyel okunur; Structured Query Language/Yapısal Sorgu Dili) uyumlu olması gerektiğini söylüyoruz. Bu dil, verilerin sabit diske yazılması ve okunmasını düzenleyen bir çok veritabanı dilinden sadece biri, fakat en yaygınıdır. Bir veritabanından veri okumak, veri değiştirmek veya eklemek için komutlarımızı bu dille vermek zorundayız.

ASP amacıyla SQL komutlarından çok az kısmını kullanırız; bu bakımdan ASP Tasarımcısı olmak için sınırlı da olsa SQL öğrenmek gerekir.

//////////////////KUTU//////////////////////

Hızlı SQL Kursu: Select

ASP amaçlı olarak kullanacağımız komut gerçekte sadece SELECT'tir. Fakat hatırlamamız gereken veritabanı ilkeleri var. Bir veritabanı kabaca alanlar (sütunlar) ve bunların içinde yazılı değerler (satırlar) halinde olur; her satır bir elemanın değerleridir; ve Kayıt adını alır.

SELECT

Bir veritabanından veri seçmeye yarar. SQL Sorgusu da denir. Dört bölümü vardır. Tipik bir SELECT komutu şöyle yazılır:

```
SELECT alan1, alan2.. FROM tablo WHERE koşul = değer
ORDER BY alan1
```

Seçilecek alanların adı SELECT komutunun ilk bölümünü oluşturur. Bir veritabanında birden fazla tablo bulunabilir; seçimin hangi tabloda yapılacağı FROM bölümünde gösterilir. Kimi zaman bir tablodaki alanda bulunan bütün kayıtları seçmek isteyebiliriz; fakat çoğu zaman seçimin sınırlarını daraltmak için sözgelimi bir alandaki değerlerin vereceğimiz bir koşula uymasını isteyebiliriz. Bu durumda "koşul = değer" testini WHERE bölümünde yaparız. Seçilen

değerlerin hangi alandaki kayıtlara göre sıralanmasını istiyorsak, ORDER BY bölümünde bunu belirtelibiliriz. Örnek:

```
SELECT Adi, Soyadi, TelNo FROM Telefonlar WHERE Alankodu = 0535 ORDER BY Adi
```

Bu komutla veritabanının Telefonlar isimli tablosundan Adi, soyadi ve TelNo adlı sütunlarındaki kayıtlardan Alankodu sütunundaki değeri "0535" olanları seçmiş oluruz. Bir tablodaki bütün alanların bütün değerlerini seçmek için SELECT komutunu şöyle yazarız:

```
SELECT * FROM Veri_adi
```

Buradaki "Veri_adi" kelimelerinin yerine DSN'e verdiğiniz adı (orneğin yukarıdaki örnekte olduğu gibi, "uyeler" kelimesini) yazacaksınız.

SQL'in INSERT, UPDATE ve DELETE komutlarının nasıl kullanıldığını öğrenirseniz, ADO nesnesinin bunlara denk gelen ve aşağıda nasıl kullanıldıklarını göreceğimiz yeni kayıt ekleme, kayıtları güncelleme ve silme metodları yerine kendi SQL komutlarınızı yazabilirsiniz.

//////////////////////KUTU BİTTİ/////////////////

Sağladığımız veri bağlantısını kullanarak, yararlanabileceğimiz bir veri grubu oluşturmak için, yukarıda .Connection metodunu kullanırken yazdığımız kodu şöyle geliştirmemiz gerekir:

```
<%
Dim Veriyolu, Kayitdizisi
Set Veriyolu = Server.CreateObject("ADODB.Connection")
Veriyolu.Open "Veri_adi"
Set Kayitdizisi = Veriyolu.Execute("SELECT * FROM Veri_adi")
%>
```

Buradaki .Execute metodu, DSN'ini verdiğiniz kaynaktaki veritabanından verileri fiilen alıp getirmeye ve bir Recordset (Kayıt dizisi) oluşturmaya yarar. Kayıtları tek tek okuması için Kayitdizisi'ne bir sonraki kayda gitmesini bildirmemiz gerekir.
Bunu .MoveNext (bir sonrakine git) metodu ile yaparız. Okunan her kayıt Kayitdizi adlı değişkenin içindedir. Bu nesnenin elemanlarını herhangi bir döngü yöntemiyle ziyaretçinin Browser penceresinde görüntüleyebiliriz; bir HTML etikenin içeriği yapabiliriz; veya başka bir şekilde kullanabiliriz. Aynı döngü, .MoveNext ile bir sonraki kaydın okunmasını da sağlayacaktır. Bunun bir örneğini daha sonra göreceğiz.

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.

.Recordset metodu, ile bir veritabanını okuyacak imleci üç şekilde ayarlayabilirsiniz:

Static	(Duragan) SELECT komutu icra edilir ve okunan kayıt arzu ettiğiniz değişkene yazılır. (ADO Sabit Değerleri dosyasınıdan yararlanıyorsak, adOpenStatic)
Forward only	(Sadece ilerle) İmleç veritabanı içinde sadece ileri doğru gider ve her seferinde bir kayıt okunur. (Varsayılan imleç türü budur.) (ADO Sabit Değerleri dosyasınıdan yararlanıyorsak, adOpenForwardonly)

Dynamic (Dinamik) Veritabanına ulaşan ve değişiklik yapan başka bir kullanıcı varsa, bu değişiklik size anında yansıtılır. (ADO Sabit Değerleri dosyasınıdan yararlanıyorsak, adOpenDynamic)

Bu yöntemlerden birini seçmekle veriyi belirli bir okuma tarzında açmış olursunuz. Bu yöntemlerden hangisini seçtiğinizi .Recordset metodunu kullanacak olan .Open komutunun argümanı olarak açıkça belirtmeniz gerekir. ADO, bunun için sizden sayılar halinde argümanlar ister.

//////////////////KUTU//////////////////

ADO Sabit Değerleri

ADO+ODBC yoluyla kuracağımız veri bağlantıları, çoğu zaman adeta şifreli ifadeler içerebilir ve bir çok komutun argümanı öğrenmesi zor sayılar halinde verilir. Microsoft ve kullanılmaya hazır ASP Uygulamaları üreten firmalar, bu karmaşık ifadeleri düz metinler olarak ifade etmeye yarayan haricî dosyalar (include files) hazırlar ve sunarlar. Bunlar arasında en yaygın olanı (bu kitapçığın kodları arasında bulunan) Microsoft'un ADOVBS (adovbs.inc) dosyasıdır. (Aynı dosyanın JavaScript sürümü ise adojavs.inc ardını taşır). Bu dosyadan yararlanabilmek için, sitenize kopyalamanız ve daha sonra sayfalarınıza şu kodu eklemeniz gerekir:

```
<!- - #include file="adovbs.inc" - - >
```

Bu dosya, Server tarafından icra edilir ve ADO nesnesinin sayı halindeki bütün argümanlarını anlaşılabilir İngilizce kelimelere çevirir. Bu dosyanın içeriğinden nasıl yararlanacağımızı ele alacağız.

//////////////////KUTU BİTTİ///////////////

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.Update

Veritabanından aldığımız değerleri, kimi zaman ziyaretçinin vereceği değerlerle veya ziyaretçinin bir takım tercihleri sonucu güncelleştirmemiz gerekir. Bu Recordset nesnesinin .Update metodu ile kolayca yapılır. Yalnız burada hassas bir nokta var: diyelim ki aynı anda iki veya daha fazla kullanıcı Web programımızın veritabanına ulaşır ve aynı anda değişiklik yaparlarsa ne olur?

Biraz önce Recordset'in .Open metodunun imleçlerinden söz ederken, okumanın yönünü veya imlecin hareket tarzını belirleyen argümanları sıralamıştık. Bu argüman dizisine bir yenisini ekleyerek, veritabanına erişimin niteliğini ve güncelleştirmenin nasıl yapılacağı ve yansıtılacağını da belirleyebiliriz. Bu işlemin temel ilkesi veritabanı kayıtlarının kilitlenmesi esasıdır. Bu kilitlemenin türünü belirleyerek, güncelleştirmenin de nasıl yapılacağını belirlemiş oluruz. Burada kullanacağımız argümanlar da ADO'nin şifreli sayıları olması gerekirken, adovbs.inc dosyası sayesinde İngilizce (ve dolayısıyla anlaşılabilir) kelimeler olur. advbs.inc dosyasını devreye soktuysanız, şu iki tür kiliti kullanabiliriz:

adLockReadOnly	Kayıtların güncelleştirilmesini önler; ziyaretçimiz veritabanına kayıt yapmayacaksa, bu kilit türünü kullanmamız gerekir.
adLockOptimistic	Veritabanına ek yapacaksak, mevcut kayıtmları düzelteceksek ve bazılarını sileceksek, bu kilit türünü kullanmamız gerekir.

Yukarıdaki kod örneğimizin sadece son satırını, bu metodu kullanmak amacıyla, şöyle yazabiliriz:

```
Kayitdizisi.Open Sorgu, Veriyolu, aOpenStatic, adLockOptimistic
```

Tabiî bir veritabanını güncelleştirmek için imleci veritabanında doğru kaydın üzerine götürmek ve bu arada Recordset'in bize sağladığı mevcut verilerin yerine yeni değerleri atamış olmak gerekir. Bunu sağladıktan sonra bütün yapacağımız şey .Update metodunu kullanmak ibarettir:

```
Kayitdizisi("Adi") = "Necip Fazıl"
Kayitdizisi("Soyadı") "Dayanır"
Kayitdizisi.Update
```

Bu komut, imleç o sırada hangi kaydın üzerinde ise o kaydın "Adi" ve "Soyadi" alanlarındaki veriyi "Necip Fazıl" ve "Dayanır" haline getirir. Bu metodu kullanırken bir

kaydın bütün alanlarını güncelleştirmemiz veya güncelleştirilmeyen alanları eski değerleri ile tekrar etmemiz gerekmez.

Recordset.Delete

Bir veritabanındaki kaydı silmek de ADO ile oldukça kolaydır. İmleci, silinecek kaydın üzerine götürdükten sonra, Recordset'in , .Delete metodunu çağırarak o andaki kayıt silinir. Bu metod, bir kaydı bütün alanlarındaki değerlerle birlikte (yani veritabanının bir satırını tümüyle) siler:

```
Kayitdizisi.Update
```

Recordset.AddNew

Bir veritabanına yeni kayıt eklemek istediğimizde, Recordset'in .AddNew (yeni ekle) metodundan yararlanırız. Bu metodun özelliği bizim imleci veritabanı içinde bir yere götürme zorunluğumuz olmamasıdır. Bu metod kendiliğinden imleci dosyanın en son satırının altına götürür. .AddNew metodu bir veritabanı dosyasına kayıt eklerken, veritabanında mevcut bütün alanlar için değer vermenizi isteyecektir. Örneğin

```
<%
Kayitdizisi.AddNew
Kayitdizisi("Adi") = "Necip Fazıl"
Kayitdizisi("Soyadı") "Dayanır"
Kayitdizisi("TelNo") = "0342-3390000"
Kayitdizisi.Update
%>
```

Veritabanına yeni kaydı, .Update metodunun yaptığına dikkat edin.

DSN'siz Veri Bağlantısı

Bu kadar DSN oluşturmayı öğrendikten sonra, "Aslında DSN olmadan da veritabanlarınıza ulaşabilirsiniz!" derlerse, herhalde çok sevinmeyiz. Ama işin doğrusu DSN yoludur. Fakat yine de DSN oluşturmadan veritabanına ulaşabileceğimizi bilmemiz gerekir.

DSN, genellikle Web Server'ları yavaşlatır; Web Server, DSN'ini belirttiğiniz veriye ulaşmak için önce ODBC'nin yardımını ister; ODBC, bir takım sürücüleri devreye sokar ve sonunda veriye ulaşırız. Bir DSN-verisine 20-30'dan fazla kullanıcı aynı anda eriştiği zaman bu yavaşlama gözle görünür hale gelebilir. Bir süre öncesine kadar Microsoft firması, veriye dayanan Web sitelerinin veri-bağını DSN yoluyla kurmasını tavsiye ederken, şimdi MS yayınlarında sık sık DSN'siz veri bağlantısının da etkin şekilde kullanılacağı belirtiliyor.

Yukarıda verdiğimiz DSN örneği şöyle idi:

```
<%
Dim Veriyolu, Kayitdizisi
Set Veriyolu = Server.CreateObject("ADODB.Connection")
Veriyolu.Open "Veri_adi"
Set Kayitdizisi = Veriyolu.Execute("SELECT * FROM Veri_adi")
%>
```

Böyle bir DSN bağlantısını kullanabilmemiz için, kendi kişisel Web Server ortamımızda Denetim Masası'ndaki ODBC aracını kullanarak bir DSN oluşturmamız; Internet ortamında ise bu adı vereceğimiz veritabanı dosyasına DSN oluşturulması için Web Server yöneticisinin yardımını istememiz gerekiyor. Oysa aynı işlemi DSN'siz veri bağlantısı kurarak da yapabiliriz. Bunun için, DSN'e yukarıdaki gibi doğrudan göndermede bulunmak yerine; ya ODBC sürücüsüne ya da ODBC'nin kullandığı Microsoft Jet OLEDB sürücüsüne doğrudan atıfta bulunuruz. Örnek:

```
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.

Veri ile HTML Etiketlerini Doldurma

ADO nesnesini tanıdık; metodlarını gördük. ADO'nun veritabanı ile DSN ile ve DSN'siz nasıl bağlantı kuracağını ele aldık. Şimdi çok kısa olarak elde ettiğimiz verilerle, HTML etiketlerinin içini nasıl dolduracağımızı görelim. Burada "içini doldurmak" (veya bir çok İngilizce kaynakta göreceğiniz üzere populate etmek) bir etiketin değer (value) bölümünü yazmak anlamına geliyor). Burada küçük bir kaç örnekle, veri kaynağını kullanarak HTML etiketlerinin içini doldurma alıştırması yapalım. Buradaki bzı örneklerde ODBC konusunu ele alırken oluşturduğumuz uyeler.mdb'yi DSN olarak "uyeler" verisi şeklinde kullanacağız; fakat siz istediğiniz veriyi kullanabilirsiniz.

Seçme Kutuları: SELECT

SELECT, ziyaretçilerimize önceden belirlenmiş bir çok unsurdan birini veya daha fazlasını seçmelerine imkan veren bir etikettir. Ziyaretçi, seçimini SELECT'in OPTION'ları arasından yapar. Seçenekler (OPTION), sahip oldukları değeri Server'a gönderirler. Genel yazım kuralı şöyledir:

```
<FORM ACTION="..." METHOD=POST|GET>
<SELECT NAME="metin">
<OPTION VALUE="değer1">Tercih 1
<OPTION VALUE="değer2">Tercih 2
<OPTION VALUE="değer3">Tercih 3
</SELECT>
```

Bu Form'un gönder (Submit) düğmesi ile sağlanan hareket (ACTION), seçilen değeri veya değerleri, Form'u işleyecek ASP programına gönderir.

Ziyaretçimize sunacağımız seçenekler, iki-üç adet ise, bunu HTML dosyasını yazarken, OPTION'lar halinde kodlamak kolay olabilir. Ancak seçenek sayısı artıyorsa, veya

seçeneklerimiz sık sık değişiyorsa, bunları bir veri tabanında toplamak ve OPTION değerlerini veritabanının bir alanından alarak ziyaretçiye sunmak çok daha kolay olur. Böylece ASP sayfası değişmeden kalır; biz sadece veritabanını güncelleştiririz. Çoğu zaman bu güncelleştirme ziyaretçilerin yapacakları eklerin veritabanına yazılmasıyla sağlandığı için, ortaya gerçekten dinamik bir Web Uygulaması çıkmış olur.

Diyelim ki, bizim grubumuzun üyelerini gösteren yukarıda oluşturduğumuz uyeler.mdb (DSN'i uyeler olan veritabanı) dosyasının adı-soyadı alanlarını birleştirerek, sayfamızdaki bir SELECT etiketinin OPTION'larına yazmak istiyoruz. Bunun için önce sayfamızda kullanacağımız değişkenleri tanımlayalım:

```
<%
Dim connVeriyolu, rsVeri, SQL
%>
```

Sonra, bu değişkenlerden veri ile ilgili olanlara .Connection ve .Recordset için gerekli ifadeleri yazalım. Veri ile çalışırken tasarımcının değişken adlarına bakarak hangisinin .Connection, hangisinin .Recordset değerlerini içerdiğini anlaması zorlaşabilir. Bu bakımdan değişken adlarının önüne .Connection için olanında conn, .Recordset için olanında rs harflerini kulllanmak yararlı olabilir. Veritabanından fiilen hangi verileri çekeceğimizi gösteren SQL deyimini de belirgin bir şekilde SQL değişkenine yazabiliriz:

```
<%
Set connVeriyolu = Server.CreateObject("ADODB.Connection")
SQL ="SELECT uyeAdi, uyeSoyadi FROM uyeler"
%>
```

Şimdi bu değerlere dayanan ve adına uyeler diyeceğimiz veri kümesini oluşturalım:

```
<%
connVeriyolu.open "uyeler"
Set rsVeri=connVeriyolu.execute(SQL)
%>
```

Artık elimizde içinde bütün üyelerin adı ve soyadını tutan bir dizi-değişken var. Şimdi biliyoruz ki veritabanından veri satır-satır okunur. Birinci satırın okunması sırasında bu değişkenin değerlerini yazacak olursak:

rsVeri (0) = üye 1'in adı

rsVeri(1) = üye 1'in soyadı

olacaktır. Veritabanından ikinci satırın okunmasında ikinci üyenin adı ve soyadı, üçüncü satırın okunmasında üçüncü üyenin adı ve soyadı bu değişkenlerin değeri olacaktır. Demek ki, bu değerleri bir SELECT etiketinin OPTION değeri olarak kullancaksak, bu işlemi ikinci satır okunmadan yaptırmamız gerekir. O halde:

```
<SELECT NAME="AdSoyad">
<% Do While Not uyeler.eof %>
<OPTION VALUE = "<%= rsVeri(0) & " " & rsVeri(1)%>"><%= rsVeri(0) & " " &
rsVeri(1)%>
</Option>
<%rsVeri.movenext
loop%>
</select>
<% rsVeri.close %>
</SELECT>
```

Do döngüsünün içinde iken veritabanından alınan değer, herhangi bir değişkenin değeri gibi kullanılabilir. Burada verilerin uyeler dizisinin dosya sonuna (eof, End Of File) okunduğuna dikkat edin. Şimdi yukarıdaki kodları bir Form içinde birleştirelim:

```
<%@ LANGUAGE="VBSCRIPT" %>
<% Option Explicit %>
<HTML>
<HEAD>
<TITLE>ASP SELECT DOLDURMA</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>
```

```
<%
' Değişkenleri tanımlayalım
Dim connVeriyolu, rsVeri, SQL
Set connVeriyolu = Server.CreateObject("ADODB.Connection")
SQL ="SELECT uyeAdi, uyeSoyadi FROM uyeler"
connVeriyolu.open "uyeler"
Set rsVeri=connVeriyolu.execute(SQL)
%>
<BODY>
Bu listeden bir üyenin adını seçiniz:
<SELECT NAME="AdSoyad">
<% Do While Not rsVeri.eof %>
<OPTION VALUE = "<%= rsVeri(0) & " " & rsVeri(1)%>"><%= rsVeri(0) & " " &
rsVeri(1)%>
</Option>
<%rsVeri.movenext
loop%>
</select>
<% rsVeri.close %>
</SELECT>
</BODY>
</HTML>
```

Bu sayfayı option.asp adıyla kaydederek sınayabiliriz. Alacağımız sonuç şuna benzemelidir:

<odbc0003.tif>

Burada yapılan seçim sonucu elde edilen değer Server'a gönderilebilir; ve söz gelimi ziyaretçinin seçtiği kişiye ait bilgiler kendisine ulaştırılabilir.

İşaretleme Alanları: INPUT-RADIO

INPUT etiketi türleri ziyaretçilerimizin önceden belirlenmiş bir çok unsurdan birini veya daha fazlasını seçmelerine veya kendilerinin girdi yapmalarına imkan veren bir etikettir. INPUT türlerinden Radio ve Checkbox (işaretleme kutusu) veritabanından çekilen

değerlerle doldurularak ziyaretçiye sunulabilir. Ziyaretçi, seçimini radyo düğmelerinden veya işaret kutularından birini işaretleyerek yapar.

INPUT etiketinin radyo düğmesi türünün genel yazım kuralı şöyledir:

```
<FORM ACTION=”...” METHOD=POST|GET>
<INPUT TYPE=”Radio” NAME=metin1 VALUE=deger1>
<INPUT TYPE=”Radio” NAME=metin1 VALUE=deger2>
<INPUT TYPE=”Radio” NAME=metin1 VALUE=deger3>
</SELECT>
```

Bu Form’un gönder (Submit) düğmesi ile sağlanan hareket (ACTION), seçilen değeri Form’u işleyecek ASP programına gönderir. Bir Form’daki bir grup oluşturan bütün radyo düğmeleri aynı adı alırlar, ki böylece ASP programına bir değişken için değer gönderilmiş olur.

Diyelim ki ziyaretçimizden beğendiği rengi seçmesini isteyen bir grup radyo düğmesi sunan bir Form yapacağız. Şu kodu radyo.asp adıyla kaydedelim:

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>ASP OPTION-RADIO DOLDURMA</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>
<%
' Değişkenleri tanımlayalım
Dim connVeriyolu, rsVeri, SQL
Set connVeriyolu = Server.CreateObject("ADODB.Connection")
SQL ="SELECT renk FROM renkler"
connVeriyolu.open "uyeler"
Set rsVeri=connVeriyolu.execute(SQL)
%>
<BODY>
<FORM><DIV ALIGN="center"><center><TABLE BORDER="0">
```

```
<TR>
<TD colspan="2" align="center"><h3>Renk</h3></TD>
</TR>
<% Do While Not rsVeri.eof %>
<TR><TD><INPUT TYPE="radio" VALUE="<%=rsVeri(0)%>" NAME="Radyo"></TD>
<TD><%=rsVeri(0)%></TD></TR>
<%rsVeri.movenext
loop%>
</TABLE></CENTER></DIV></FORM>
</BODY>
</HTML>
```

Burada "Radyo" isimli radyo düğmesine verdiğimiz değerleri, ODBC'nin "uyeler" adıyla tanıdığı veritabanından alıyoruz; bir önceki örnekten farklı olarak bu kez aynı veritabanındaki farklı tablodan, renkler tablosundan ve sadece bir alanın, renk alanının değerlerini çekiyoruz. Bu örnek programda da ziyaretçinin seçtiği değerleri Server'a gönderecek bir Gönder düğmesi yok. Ama içeriğini veritabanından aldığımız değerlerle doldurduktan sonra, ziyaretçinin radyo düğmeleriyle yapacağı tercihler, tıpkı klasik HTML'deki gibi kullanılabilir; Server'a değişken olarak gönderilebilir; veya ziyaretçinin bilgisayarında (client-side) herhangi bir Script tarafından kullanılabilir.

İşaretleme Alanları: INPUT-CHECHBOX

INPUT etiketinin ziyaretçiye işaretleyerek tercih imkanı veren diğer aracı Checkbox (işaretleme kutusu) türüdür. Tıpkı radyo düğmesinde olduğu gibi veritabanından çekilen değerlerle doldurularak ziyaretçiye sunulabilir. Ziyaretçi, seçimini işaret kutularından birini işaretleyerek yapar.

INPUT etiketinin Checkbox türünün genel yazım kuralı şöyledir:

```
<FORM ACTION="..." METHOD=POST|GET>
<INPUT TYPE="checkbox" NAME=metin1 VALUE=deger1>
<INPUT TYPE="checkbox" NAME=metin1 VALUE=deger2>
<INPUT TYPE="checkbox" NAME=metin1 VALUE=deger3>
```

```
</SELECT>
```

Bu Form'un gönder (Submit) düğmesi ile sağlanan hareket (ACTION) seçilen değeri Forma gönderecektir. Radyo düğmesi ile Checkbox'ın arasındaki fark, ziyaretçinin aynı ismi taşıyan radyo düğmelerinden birini işaretleyebilirken; istediği kadar Checkbox'a işaret koyabilmesidir. Birden fazla Checkbox işaretlendiği taktirde Server'a "metin1=deger1, deger2.." şeklinde bilgi gönderirler. (ASP programlama açısından, bu değişken Request.Form nesnesinde Checkbox'ın adını taşıyan kolleksiyonun içinde dizi-değişken olarak yazılır.)

Yukarıdaki radyo düğmesi örneğimizin sadece Do döngüsüne ait kısmını değiştirerek,Checkbox'a uyarlayalım, ve isaret.asp adıyla kaydedelim:

```
<% Do While Not rsVeri.eof %>
<TR><TD><INPUT TYPE="checkbox" VALUE="<%=rsVeri(0)%>" NAME="Isaret"></TD>
<TD><%=rsVeri(0)%></TD></TR>
<%rsVeri.movenext
loop%>
```

Ziyaretçi bu form ile birden fazla kutu işaretleyerek Gönder düğmesine basarsa, Server'a gelecek bilgi örneğin, "Isaret=Kırmızı, Mavi" şeklinde olacaktır.

ASP Kaynakları

Bu kitapçıkta öğrendiğimiz ASP ve VBScript tekniklerinin uygulamasına ilişkin örnekleri, ikinci kitapçıkta bulacaksınız. İkinci kitapçığa geçmeden önce, buradaki küçük uygulamaların yapılması ve örneklerin çalıştırılması yerinde olur.

Bu sırada ASP teknikleri ve VBScript komutları hakkında geniş bilgiyi Internet'te şu kaynaklarda bulabilirsiniz:

asp.superexpert.com/

aspwire.com/

www.15seconds.com/

www.asp101.com/home/home.asp

www.aspalliance.com/

www.aspdeveloper.net/

www.aspexpress.com/aspexpress.asp

www.aspforums.com/

www.aspguild.org/default.asp

www.asp-help.com/

www.asphole.com/asphole/default.asp

www.aspin.com/index/default.asp

www.asplists.com/asplists/

www.aspmessageboard.com/

www.asppipeline.com/

www.aspsite.com/

www.codeave.com/

www.devasp.com/

www.haneng.com/

www.mavweb.net/

www.programmersresource.com/

www.shopit.co.uk/

www.takempis.com/

www.tcp-ip.com/

ASP/KITAP 2

Önsöz

Bu kitapçık önce Byte, ardından PC World ve şimdi de PC Life dergisinin eki bid dizinin son bölümü olan ASP tekniklerinin temel ilkelerini ve VBScript kullarını anlattığımız kitapçığın devamı niteliğindedir. Bu kitapçıktan yararlanabilmek için PC Life dergisi Eğitim dizisinin Haziran 2000 sayısıyla verilen birinci bölümünü edinmeniz gerekir.

ASP veya Active Server Pages (Etkin Sunucu Sayfaları) tekniği, sayfalarınızı canlandıracak bir tekniktir. Bu teknik, sil baştan bir bilgisayar programlama dili öğrenmeye gerek olmadan uygulanabilir. Bununla birlikte ASP tekniğini kullanabilmek için HTML ve Web'in nasıl çalıştığını, Server (Sunucu) ve Client (İstemci) ilişkisinin nasıl yürüdüğünü de bilmek gerekir.

ASP teknolojisinden yararlanabilmek için Web sitesine evsahipliği yapan bilgisayarda çalışmakta olan Web Server'ın ASP teknolojisini tanıması ve sitenize bu hizmeti vermesi şarttır. ASP, bir zamanlar sadece Microsoft'un NT ve daha sonra Windows 2000 işletim sistemine dayanan MS-IIS (Internet Information Server) programında işleyebilirdi. Fakat şimdi artık NT-tabanlı diğer Web Server programları gibi Unix-tabanlı Web Server programları da ASP anlar ve işletir hale gelmiş bulunuyor. Fakat ASP sayfalarınızı gerçek Internet ortamında ziyaretçilerinizin hizmetine sunmadan önce, kendi bilgisayarınızda sınamanız gerekir. Bunun için bilgisayarınızın işletim sistemi Windows 95/98 ise Kişisel Web Server (PWS), NT 4.0 ise IIS kurulmuş olmalıdır. Sisteminiz Windows 2000 ise, IIS 5.0 kendiliğinden kurulmuş demektir.

Örnek Kodlar

İki kitapçıktan oluşan ASP rehberinin bütün örnek kodları ve veritabanı dosyaları, http://www.pclife.com.tr/..... adresinden edinilebilir. Ancak ASP öğrenirken örnek kodları kendiniz yazmalısınız. Kodlamanın incelikleri ancak kodları siz yazarsanız öğrenilebilir. Bu örneklerden sadece kendi yazdığınız kodlar hata verdiği ve hatayı bulmaktan güçlük çektiğiniz zaman, karşılaştırma amacıyla yararlanmalısınız.

Teşekkürler

Bu kitapçıktaki kodların değişik ortamlarda sınanması için için bana yardımcı olan dilini ve yazımını denetleyen dostlarım Mustafa Durcan, Osman Nuri Hömek ve Armağan Ergon'a teşekkür ederim. Varolan hatalar elbette bana aittir. Çok sevgili arkadaşım Mustafa Durcan'ın bütün dizinin gelişimine katkısını daima şükranla hatırlayacağım.

Alıştırma Uygulamaları

Bu bölümde, birinci kitapçıkta öğrendiklerimizi küçük ASP uygulamalarında kullanacağız; ve böylece son iki bölümde yer alan nisbeten daha büyük ASP Programları için hazırlık yapmış olacağız. Bu bölümdeki örnekleri mutlaka kendiniz yazmalısınız ve açıklanmamış kodlardan anlamadıklarınız olursa mutlaka daha önceki bölümlere dönerek gözatmalısınız. Bu bölümdeki kodları dikkatle incelerseniz, çoğunda daha önce ele aldığımız temel VBScript komutlarının veya ASP nesnelerine ait metodlarının kimi zaman değişik tarzda kullanımına yer verdiğimizi göreceksiniz. Temel kullanım biçimlerini öğrendikten sonra, mutlaka Microsoft'un MS-Developers Network sitesinde (http://www.microsoft.com/msdn) VBScript ve ASP konularındaki belgeleri ve eğitim metinlerini incelemelisiniz.

Doğum Günü Hesabı

VBScript'in dil olarak bize sağladığı bazı kolaylıkları ASP teknolojisi ile birleştirebiliriz. ASP sayfalarımızda istediğimiz gibi HTML kodları da kullanırız. Bu örnekte, bu iki unsuru da birarada göreceğiz. Aşağıdaki kodu dogumgunu01.asp adıyla kaydedin:

```
<% @LANGUAGE=VBscript %>
<%
Option Explicit
Response.Expires = 0
Dim serverSaat, kalanSaat
serverSaat = Time
' aşağıdaki satırda işaretler arasındaki yere kendi doğum gününüzü yazın
kalanSaat = DateDiff("h",Now,#7/8/2000#)
%>
<HTML>
```

```
<HEAD>
<TITLE>Dogum Günü Hesabı!</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>
<H3>
<CENTER>
<p>Selam:</p>
<p>Şu anda saat: <%=ServerSaat%></p>
<%
If kalanSaat > 0 Then
    Response.Write "Doğum gününüze " & kalanSaat & " saat var." & VbCrLf
ElseIf kalanSaat < 0 Then
    Response.Write "Doğum gününüz " & kalanSaat & " geçmiş buluyor." & VbCrLf
Else
    Response.Write "<b>Doğum gününüz kutlu olsun!</b>" & VbCrLf
End If
%>
</CENTER>
</H3>
</BODY>
</HTML>
```

Bu kodda bulunan ASP nesnelerini ve metodlarını kolayca görüyor olmalısınız. Tanımlamadığımız bir değişken kullanmayı veya değişken adlarını yanlış yazmayı önleyen Option Explicit ifadesi dikkatinizi çekiyor olmalı. Sayfamızda bir hesap yapılacağı için Browser tarafından geçici Internet dosyaları arasında (cache) saklanmasını istemediğimizi, Response.Expires'ın değerini 0 yaparak belirtiyoruz. iki değişken tanımlıyoruz (serverSaat ve kalanSaat) ve bunlardan birincisine Server'ın o andaki saatini Time fonksiyonu ile atıyoruz. İkinci değişken ise hesaplama sonucu belirlenecek. Bu arada ASP programının

hangi tarihe kaç saat kaldığını hesaplamasını istiyorsak, kalanSaat'i belirleyecek formüle o tarihi yazıyoruz. Bu formüle dikkatle bakalım:

```
kalanSaat = DateDiff("h",Now,#7/8/2000#)
```

DateDiff ve DateAdd, VBScript'in vereceğiniz iki tarih arasında, yine vereceğiniz biçime göre farkı bulmasını veya toplamasını sağlar. Buradaki "h" hesap sonucunun saat olarak verilmesini istediğimiz gösterir. "d" ise sonucu gün olarak istediğimiz anlamına gelir. Now, Server'ın o andaki saatidir; "#" işaretleri arasında ise hesabın yapılacağı tarih, veya buradaki örnekte doğum gününüz yer alıyor. kalanSaat değişkeninin değeri bu hesaplama sonucu olarak doğum gününüze kalan saat olacaktır.

Kodumuzun geri kalan bölümünde ise elde ettiğimiz sonucu Browser'a yollayan bir If seçimi var: doğum gününüze kalan saat 0'dan büyükse veya küçükse bunu farklı cümlelerle belirtiyoruz. Kalan süre 0 ise, bugün doğumgününüz demektir! (ASP ile bu kadar uğraşırsa, insanın doğum gününü filan unutması normaldir. üzülmeyin!)

Daha önce VBScript'in metin ve değişkenleri birbirine "&" işareti ile eklediğini görmüştük. VBScript ile yazdırdığımız satırların sonuna kimi zaman
 kodu koyarak, yeni satırın bir satır aşağıdan başlamasını sağladığımızı hatırlıyor olmalısınız. Burada & VbCrLf şeklinde gördüğümüz ifade de bunu sağlar. Bu, Visual Basic'in "alt satıra geç, başa git!" (Carriage Return, Line Feed) komutudur.

<asp0011.tif>

Dedik ki, DateDiff ve DateAdd fonksiyonlarının, sonucu gün olarak vermesini istiyorsanız, "d" argümanını kullanmanız gerekir. Şimdi burada öğrendikleriniz bu programı doğumgününüze kalan saati değil de günü hesaplayacak şekilde değiştirebilir misiniz? (Ortaya çıkartacağınız kodu bu kitapçığın kodları arasındaki dogumgunu02.asp dosyası ile karaşılaştırabilirsiniz.)

<asp0012.tif>

//////////////////KUTU//////////////////////

Çift Tırnak Gerekince!

Kimi zaman ASP yoluyla ziyaretçininr Browser'ına HTML kodu göndermek gerektiğinde HTML'in gerektirdiği çift tırmak işaretini yazdırma sorunuyla karşılaşırız. VBScript'in gereği olan çift-tırnak işaretinin içine koyacağımız HTML'in gereği olan çift tırnak, ASP hatasına yol açar, C'den türetilen dillerde ters bölü işaretiyle yaptığımız türden, kontrol karakterlerinin metin olarak anlaşılmaması işini (ESCape-ing'i) VBScript'te türlü şekillerde yapabiliriz. Burada, yerine göre, üç yöntemden yararlanabiliriz işe yarar:

1. Tırnak işareti gereken yerde iki kere tırnak yazmak;
2. İçerdeki tırnak yerine tek-tırnak kullanmak
3. İçerdeki tırnak yerine & Chr(34) & kullanmak. (Bu yöntemi kullandığımızda "dışardaki" tırnak diye bir şey kalmadığına dikkat etmek gerekir.)

Örnekler:

```
objMetin.WriteLine("<%@ LANGUAGE=""VBScript""%\>")
objMetin.WriteLine("<%@ LANGUAGE='VBScript'%\>")
objTsOut.WriteLine("<%@ LANGUAGE=" & Chr(34) & "VBScript" & Chr(34) & "%\>")
```

//////////////////KUTU BİTTİ///////////////

HTML Dışında İçerik

HTTP iletişiminde istemci Browser ile Server arasında her talep ve sunuşla birlikte bir dizi HTTP Header (başlık) bilgisi gönderildiğini biliyoruz. Bu bilgiler arasında neler bulunduğunu dabu kitapçığın kodları arasında bulunan SerDeg.asp dosyasını çalıştırarak inceleyebilirsiniz. Browser penceresinde beliren bilgilere dikkat ederseniz, Browser, Server'a HTTP yoluyla kabul edebileceği bilgi türlerini de sıralıyor:

```
HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/msword, application/vnd.ms-powerpoint, application/vnd.ms-excel,
*/*
```

Burada düzyazı (text/plain) türünün bulunmadığını görüyorsunuz. Fakat ASP yoluyla, sayfanıza düz yazı gönderebilirsiniz. Aşağıdaki kodu duzyazi.asp adıyla kaydedin, ve çalıştırın:

```
<% @LANGUAGE=VBscript %>
<%
Response.ContentType = "text/plain"
Response.Expires = 0
%>
Selamlar
Bu dosyada HTML etiketi kullanmıyoruz.
Metin tamamen bir düz yazıdan ibaret. İçinde <BR> kodu
bulunmadığı halde
satırlar Enter'a bastığımız yerde alt satıra geçecektir.
```

Bu kodu yazarken buradaki örnekte gördüğünüz satır sonlarında Enter'a basmalısınız. Browser'ınızdaki sonuç şuna benzer ıolmalı:

<asp0013.tif>

Genellikle HTML sayfalarımızı tasarlarken içerik türünü belirten Content-Type özelliğini yazmayız; çünkü Browser, içerik türü belirtilmemişse gelen belgenin mutlaka "text/html" olduğunu varsayar. Fakat burada olduğu gibi, içinde bir tek HTML etiketi (hatta <HTML> etiketi bile) olmayan bir metni Browsar'a gönderebilir ve istediğimiz şekilde görüntületebiliriz. Bunu Response.ContentType = "text/plain" ifadesi sağlamaktadır.

Response nesnesi, varsayılan türler dışında sadece düz yazı türü göndermekle kalmaz, Browser'ın kabul edebileceği ve HTTP'nin aktarılmasına elverişli olduğu her türlü dosyayı gönderebilir. Bu dosya türlerinin hangileri olabileceğini çevre değişkenlerinden HTTP_Accept'in içinde görebilirsiniz. Bunu sınamak için duzyazi.asp'nin Content-Type satırını şu şekilde değiştirin ve dosyayı msword.asp adıyla kaydedin:

```
Response.ContentType = "application/msword"
```

Bu dosyayı çalıştırdığınızda, Broewser türünüze ve ayarlarınıza ve sisteminizde MS-Word programının bulunup bulunmamasına bağlı olarak, ya doğruca ya da size sorarak, bilgisayarınız MS Word kelime işlemcisini açacaktır:

<asp0014.tif>

Bu imkandan ziyaretçiye resim veya diğer tür dosyaları göndermek için yayarlanabiliriz. Ancak bir resim dosyasını gönderirken, metin vermek yerine dosya yolu ve adını belirtmemiz gerekir.

Başka Sayfaya Yönlendirme

Diyelim ki ziyaretçilerimizin Web programımızda belirli işleri yapabilmeleri veya eski ifadesiyle sitemizde belirli bir sayfaya erişmek için kayıt yaptırmış olmaları gerekiyor; bu kaydı yaptırmamış olanları kayıt sayfasına yönlendirmek istiyoruz. Birazdan form yoluyla ziyaretçiden bilgi alma ve bunu işlemeyi göreceğiz, ama şimdilik sadece bu senaryonun ikinci kısmını ele alalım: ziyaretçimizi başka sayfaya yönlendirelim. Bunun için iki ayrı sayfa yapacağız. Önce şu kodu, gonder.asp adıyla kaydedin:

```
<% @LANGUAGE=VBscript %>
<%
Response.Redirect("yeni.asp")
Response.Expires = 0
%>
<HTML>
<HEAD>
<TITLE>ASP ILE YONLENDİRME</TITLE>
</HEAD>
<BODY>
Bu sayfada yazılı olanları okumanız mümkün olmayacak<br>
çünkü bu sayfa sizi başka sayfaya gönderecek..
</BODY>
</HTML>
```

Şimdi şu kodu da yeni.asp adıyla kaydedin:

```
<HTML>
<HEAD>
<TITLE>YENİ SAYFA</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>
<%= "Sizi buraya başka sayfa gönderdi.. <BR>"%>
</BODY>
</HTML>
```

gonder.asp'yi çağırın; karşınıza yeni.asp'nin içeriği gelecektir; çünkü gonder.asp kendisinden beklediğimizi yapacak ve sizi yeni.asp'ye gönderecektir. Bunu "Response.Redirect" metodu yapmaktadır. Bu metodun argümanı olarak kendi programımızın bir başka sayfasını verebileceğiz gibi, bir Internet sitesinin URL'ini de verebiliriz.

Ziyaretçiden Bilgi Alma

İstersek adına Web Programı diyelim, sitemize ziyaretçilerin bilgi ulaştırması ancak HTML'in Form etiketi yoluyla olur. Klasik HTML+CGI yöntemleri ile ASP tekniğinde Form etiketlerinin işlevleri arasında bir fark yoktur. Ancak aralarındaki benzerlik de hemen hemen bundan ibarettir. Klasik HTML+CGI yönteminde Form'larımızın Action özelliğine değer olarak çoğu zaman bir CGI programının yolunu ve adını yazarız; oysa ASP'de Form bu bilgileri kendisinin de içinde bulunduğu ASP sayfasına gönderebilir; ve örneğin formun eksik doldurulup doldurulmadığını sınayabilir.

HTML bilgilerinizi tazelerseniz; bir Form'dan Server'a iki metodla bilgi gelebilir: Get ve Post yöntemleriyle. Get, Browser'ın bilgileri yumak yapıp, aradaki boşlukları kaldırarak

ve ASCII olmayan karakterleri URL koduyla şifreleyerek Query_String içine yazdırmasını sağlar. Diğer yöntemi birazdan ele alacağız.

Küçük bir form oluşturalım ve bu formdan Get yoluyla gelecek bilgileri daha sonra nasıl kullanabileceğimizi görmek için, şimdilik sadece sayfamıza yazdıralım. Şu kodu miniform_get.asp adıyla kaydedin:

```
<% @LANGUAGE=VBscript %>
<%
Option Explicit
Response.Expires = 0
Dim strAdi, strSoyadi, hamBilgi, islenmisBilgi
If Request.ServerVariables("QUERY_STRING") <> "" Then
     strAdi = Trim(Request.QueryString("adi"))
     strSoyadi = Trim(Request.QueryString("soyadi"))
     hamBilgi = Trim(Request.QueryString("mesaj"))
     islenmisBilgi = Replace(hamBilgi, vbcrlf, "<BR>" & vbcrlf)
%>
<html>
<head>
<title>Mini Form</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>
Script'imize Form'dan bilgi ulaştı <BR><BR>
Gelen bilgiler:<BR>
Formu dolduran kişinin adı: <%= strAdi%> <BR><BR>
Formu dolduran kişinin Soyadı: <%= strSoyadi%> <BR><BR>
Ham Bilgiler: <%= hamBilgi%> <BR><BR>
İşlenmiş Bilgiler: <%= islenmisBilgi%> <BR><BR>
"Query_String" olarak gelen bilgi: <BR>
<%= Request.ServerVariables("QUERY_STRING")%>
</body>
```

```
</html>
<%
Else
%>
<html>
<body>
Bize bilgi verir misiniz?<BR>
<FORM ACTION= "<%= Request.ServerVariables("SCRIPT_NAME") %>" METHOD="GET">
Adınız: <INPUT TYPE="Text" NAME="adi"><BR>
Soyadınız: <INPUT TYPE="Text" NAME="soyadi"><BR>
Mesajınız: <TEXTAREA NAME="mesaj">Mesajınızı buraya yazın!</TEXTAREA><BR>
<INPUT TYPE="Submit" NAME="Gönder" VALUE="Gönder">
</FORM>
</body>
</html>
<% End If %>
```

Kodumuzun bir değişken tanımladığımız ve Form'dan gelen bilgileri işleyen bölümü, bir de henüz bu bilgi gelmeden önce Form'u oluşturan bölümü olduğuna dikkat ediyor musunuz? Bu iki bölümü bir If sınama deyimiyle oluşturduğumuzu da görüyorsunuz.

ASP tekniği ve VBScript'in kolaylıkları birleşince, CGI'da olduğu gibi, Formdan gelen bilgileri özel fonksiyonlar yazarak ayıklamamız gerekmez. ASP'nin anladığı şekliyle, Query_String Form'dan GET metoduyla gelen bilgileri Form'un bilgileri atadığı değişkenlerin adlarıyla eşleştirerek tutar; ve bize sadece Form'daki değişkenlerin değerlerini, istiyorsak, başka değişkenlere atamak kalır. Burada Form'un "adi" olan değişkenini bir String değişkeni olan strAdi değişkenine, yine Form'un "soyadi" olan değişkenini başka bir bir String değişkeni olan strSoyadi değişkenine atıyoruz. Bunu yapmamızın sebeplerinden biri iki dizi değişkenden hangisinin ASP tarafında kullanıldığını bilmektir.

Form'un üçüncü değişkeni olan "mesaj" ise ASP tarafından işlenirken, iki şekilde ele alınacak: hamBilgi değişkeninin değeri olarak, ve bunu Replace() fonksiyonu ile işledikten sonra atayacağımız islenmisBilgi değişkenini değeri olarak.

Replace() fonksiyonu, bir değişkende bir değeri bulur ve arzu ettiğimiz bir başka değerle değştirir. Üç argüman alır: içinde arama ve değişiklik yapılacak değişken, aranacak değer ve yerine konacak değer. Burada iki veya daha fazla paragraf içeren bir mesajın içindeki satırbaşı-yeni paragraf karakterini, HTML'in anlayacağı
 ve satırbaşı karakteri ile değiştiriyoruz.

Şimdi miniform_get.asp'yi çalıştırın, doldurun (mesaj bölümünde iki paragraf oluşturmak için bir yerde Ctrl+Enter'a basmayı unutmayın! Yoksa işlenmiş bilgi ile ham bilginin farkını göremeyiz!) ve Gönder tuşunu tıklayın. Bu arada Browser'ınızın URL hanesine bakın. Bu Get metodunun bilgi gönderme şeklidir. Bilgiler URL-kodu olarak değiştirilir ve gideceği sayfanın adına, soru işareti ile eklenerek, gönderilir. (Tabii ziyaretçiniz yıldızlar halinde görüntülenen bir parola yazdıysa onu da burada açık şekilde görecektir!)

<asp0015.tif>

Bizim Form bilgilerimiz hangi sayfaya gönderiliyor? Yine kendisine. Bunu Form etiketinin Action özelliğinin karşısında yazılı olan VBScript kodu sağlıyor. Bu yolla gelen bilgi, şuna benzer olmalıdır:

```
Script'imize Form'dan bilgi ulaştı

Gelen bilgiler:

Formu dolduran kişinin adı: Kroninukus

Formu dolduran kişinin Soyadı: Computerium

Ham Bilgiler: Merhaba Ben üniversiteyi yeni bitirmiş bir gencim!

İşlenmiş Bilgiler: Merhaba
```

```
Ben üniversiteyi yeni bitirmiş bir gencim!

"Query_String" olarak gelen bilgi:

adi=Kroninukus+&soyadi=Computerium+&mesaj=Merhaba%0D%0ABen+%FCniversiteyi+yeni
+bitirmi%FE+bir+gencim%21+%0D%0A&G%F6nder=G%F6nder
```

Replace() fonksiyonunun, mesajın içindeki CRLF karakterini bulup yerine
 koyduğuna dikkat edin. Query_String'in yazdığınız ad ve soyad ile mesajı nasıl yumak (tek String) yaptığına bakın. Ve programımızın bu yumaktan bilgileri alıp, yeni değişkenlere atayarak Browser penceresinde nasıl görüntülediğini inceleyin.

Bu programın canalıcı noktası, Request nesnesinin ServerVariables kolleksiyonunda, Query_String koleksiyonunu kullanmasıdır. Çünkü Get, Form'un sağladığı bilgileri bu koleksiyona kaydetmiş bulunuyor.

Şimdi aynı işi Post metodu ile yapalım. Bunun için biraz önce kaydediğiniz kodda şu değişiklikleri yapın ve miniform_post.asp adıyla kaydedin:

```
<% @LANGUAGE=VBscript %>
<%
Option Explicit
Response.Expires = 0
Dim strAdi, strSoyadi, strBilgi
If Request.ServerVariables("CONTENT_LENGTH") <> 0 Then
            strAdi = Trim(Request.Form("adi"))
            strSoyadi = Trim(Request.Form("soyadi"))
            strBilgi = Trim(Request.Form("mesaj"))
            strBilgi = Replace(strBilgi, vbcrlf, "<BR>" & vbcrlf)
%>
<html>
<head>
<title>Mini Form (Post)</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>
Script'imize Form'dan bilgi ulaştı <BR><BR>
Gelen bilgiler:<BR>
Formu dolduran kişinin adı: <%= strAdi%> <BR><BR>
Formu dolduran kişinin Soyadı: <%= strSoyadi%> <BR><BR>
Mesaj: <%= strBilgi%> <BR><BR>
</body>
</html>
<%
Else
%>
<html>
<body>
Bize bilgi verir misiniz?<BR>
<FORM ACTION= "<%= Request.ServerVariables("SCRIPT_NAME") %>" METHOD="POST">
Adınız: <INPUT TYPE="Text" NAME="adi"><BR>
Soyadınız: <INPUT TYPE="Text" NAME="soyadi"><BR>
Mesajınız: <TEXTAREA NAME="mesaj">Mesajınızı buraya yazın!</TEXTAREA><BR>
<INPUT TYPE="Submit" NAME="Gönder" VALUE="Gönder">
</FORM>
</body>
</html>
<% End If %>
```

İki Miniform sayfası arasındaki değişikliği farkettiniz mi? Önce, Form etiketinin METHOD özelliğinin Post olduğuna dikkat edin. Bu yöntemle gelen bilgiler, Request nesnesinin Form kolleksiyonuna, işlenmiş (yani değişken=değer çiftleri halinde yazılmış) olacağı için, Form bilgilerini yeni değişkenlere atama ifadelerimizi de farklı yazıyoruz. Önce formdan gerçekten bilgi gelip gelmediğini Content_Length değişkeninin boyutunun sıfır olup olmadığına bakarak anlıyoruz. Bu değişkenin boyutu 0 ise form içi boş gönderilmiş demektir; bu durumda ziyaretçiyi form'u doldurması için uyaran bir Sub'a gönderebiliriz.

Form doldurulmuş ise, kodumuzun birinci bölümü yapması gereken değişkenlere atama işlemini yapıyor; ve sonuçları ziyaretçinin Browser penceresine yazıyor. Daha sonra yapacağımız örnek Konuk Defteri'nde bu bilgileri ziyaretçinin ekrarına değil, bir metin dosyasına yazacağımızı göreceksiniz.

Form etiketlerinden bilgi alma

Form içinde kullandığımız bir çok etiket, Server'a farklı biçimde bilgi gönderir. Yukarıdaki örnekte gördüğümüz Input ve Textarea etiketleri, kendi adlarını değişken adı yaparak ve değişkene ziyaretçinin yazdığı metni değer olarak atayarak, gönderir. Fakat HTML'in <SELECT MULTIPLE> etiketi biraz daha dikkat etmemizi gerektirir; çünkü ziyaretçi bu etiketle oluşturduğunuz seçeneklerden birden fazla seçebilir ve bunlar Server'a bir kolleksiyonun elemanları olarak gelir. Şöyle bir örnek düşünün:

```
<SELECT NAME="CokluSecme" SIZE="3" MULTIPLE>
<OPTION VALUE="Emrah" SELECTED> Emrah
<OPTION VALUE="Karacaoglan"> Karacaoğlan
<OPTION VALUE="Sulari"> Davud Sulari
<OPTION VALUE="Daimi"> Aşık Daimi
<OPTION VALUE="Pirsultan"> Pir Sultan Abdal
```

Browser, kullanıcı birden fazla seçim yaptıysa, yapılan seçim sayısını, yani kaç seçenek seçildiğini bize bildirir ve bu Request nesnesinin Form kolleksiyonunda "CokluSecme" elemanının ".Count" değişkenine yazılır. Bu bakımdan "CokluSecme" kendisi bir kolleksiyondur. Yani, bu değişkenin değerini kullanarak, "CokluSecme" kolleksiyonunun elemanlarını tek tek okutmak için bir For döngüsünden yararlanabiliriz:

```
For sayac = 1 to Request.Form("CokluSecme").Count
            Response.Write(sayac & ". seçim: " &
Request.Form("CokluSecme")(sayac) & "<BR>")
Next
```

Bu iki kodu, miniform_post.asp'ye katabilir misiniz? (Ortaya çıkartacağınız örneği bu kitapçığın örnekleri arasındaki miniform_multi.asp ile karşılaştırabilirsiniz.)

Parola İle Sayfa Koruma

"Internet'te gizlilik olmaz!" ilkesine rağmen, bazen öyle sayfalarımız olur ki, bunların içeriğini bütün Internet ziyaretçilerinin görmesini istemeyebiliriz. Örneğin derneğimizin veya grubumuzun telefon numaraları ve elektronik posta adreslerini gösteren sayfamıza sadece grup üyelerinin ulaşmasını arzu edebiliriz. Bir Web sitesinin herhangi bir sayfasını ziyaretçilerden gizlemek için sayfanın açılmasını belirli bir şartın yerine gelmiş olmasına, mesela ziyaretçinin bir Form aracılığıyla bize doğru parolayı göndermesine bağlayabiliriz.

Bu tür sayfa gizleme yollarını istemci-tarafında çalışan bir programla yapmak mümkündür; ancak sayfa gizlemenin mantığına aykırıdır. sözgelimi bir Form'da parola alanına girilecek bilginin gerçek parola ile karşılaştırılmasını bir Script fonksiyonu ile yapmaya kalkarsak, ziyaretçinin kaynağı görüntülemesi halinde ziyaretçinin girdisi ile parolayı karşılaştıran Script, ve tabiî bizim kıymetli parolamız, ziyaretçi tarafından öğrenilecektir. Oysa ASP kodları ile gizlediğimiz parolanın ziyaretçiye görünmesi imkansızdır; çünkü ASP kodları hiç bir zaman ziyaretçiye gönderilmeyeceği için parola sınayan program da ziyaretçiye gidemeyecektir.

Önce şu kodu, parola.asp adıyla kaydedelim:

```
<%@ LANGUAGE="VBSCRIPT" %>
<%
Response.ExpiresAbsolute = Now() - 1 'Sayfanın yedeklenmesini önleyelim
FormParola = ucase(trim(request.form("FormParola")))
If FormParola <> "PAROLA" Then
%>
<HTML>
```

```
<HEAD>
<TITLE>ASP ILE PAROLA</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="#ccffff" text="black" link="navy" vlink="purple">
<DIV align="center">
<FORM action="parola.asp" method="POST">
<h2>Ana sayfaya girmek için parolayı yazınız (Mesela, PAROLA) ?</h2>
  <input type="password" name="FormParola" size="10"><br><br>
  <input type="submit" value="Girebilir Miyim?">
</form>
</div>
</body>
</html>
<% Else %>
Şimdi ana sayfaya girmiş oldunuz..
<% End If %>
```

Şimdi kodumuzu ayrıntılı indereleyebiliriz. Önce, bu sayfanın ziyaretçinin Browser'ı tarafından ziyaretçinin bilgisayarında yedeklenmesini önlediğimize dikkat edin: Bunu Response nesnesinin .Expries metodunun farklı bir türü ile yapıyoruz. .ExpiresAbsolute sadece .Expires metodundan farklı olarak, bizden bir değer ister. Bu değerin zaman olarak verilmesi gerekir. Burada VBScript'in o andaki zamanı belirten Now() fonksiyonu ile yapıyoruz ve bu fonksiyonun verdiği değerden (yani o andaki saatten) 1 saat çıkartıyoruz. Böylece sayfanın geçerliği çoktan dolmuş oluyor. Browser, bu ifadeyi gördüğü anda sayfanın daha sonra başvurulmak üzere bir kopyasını saklamayacaktır. Bunu neden yapıyoruz? Diyelim ki parolayı doğru bilen bir ziyaretçi kullandığı bilgisayarın başından kalktı ve yerine başka bir kullanıcı geçti. Bu kişinin Browser'ın Yenile düğmesini tıklaması halinde bizim özene-bezene gizlediğimiz sayfa yeniden görüntülenecektir. Oysa Browser'a bu sayfanın artık geçerli olmadığını bildirmekle, yenilenme işleminin Cache bellekten

(ziyaretçinin bilgisayarında sabit diskte kaydedilen kopyadan) değil de, mutlaka Server'dan yapılmasını istiyoruz.

Sayfamız açılırken, (kodumuzun dört ve beşinci satırında) kendi içerdiği formda bilgi olup olmadığını sınıyor. Bunu formdaki formParola isimli INPUT alanının değerini atadığımız değişkenin içeriğini parola olarak seçtiğimiz kelime karşılaştırarak yapıyoruz. Eğer bu değişken (buradaki örnekte "PAROLA" kelimesine) eşit değilse, yani parola yanlış veya boşsa If sorgusu olumlu sonuç verecek ve program devam edecektir. If sorgusu yanlış sonuç vermiş olsaydı, yani ziyaretçi Form'a bilgi girmiş ve bu bilgi bizim karşılaştırmayı yaptığımız kelimeye eşit olmuş olsaydı, program ELSE ifadesine atlayacaktı. Programın devamı formu oluşturmaktadır; ELSE ifade ise formu atlamakta ve ziyaretçinin Browser penceresine "Şimdi ana sayfaya girmiş oldunuz" cümlesini yazdırmaktadır. Tabiî gerçek bir uygulamada bu cümlenin yerinde gerçek bir sayfanın ögeleri yer alacaktı. Bu yöntemi uygularken sayfanın VBScript'nin End If ifadesiyle bittiğine dikkat edin.

Dinamik İçindekiler Sayfası

Internet sitemizin sayfaları genellikle çok özenilmiş, cicili-bicili, albenisi olan, görsel odak noktalarının oluşmasına dikkat edilmiş tasarımlar olur. Fakat bazen, önemli olan sadece bir dizindeki bütün HTML ve ASP belgelerinin adlarını ve başlıklarını liste halinde ziyaretçiye sunmaktır. Böyle bir sayfada da tasarımın görsel ilkelerine dikkat edilebilir. Fakat burada önemli olan, sayfanın kendi kendisini inşa etmesi ve sayfanın bir dizinin içindeki tüm dosyaları zyaretçiye listeyen bölümünün otomatik olarak güncellenmesidir.

Bunu, bir HTML sayfası oluşturan ASP programı olarak tasarlayabiliriz. Aşağıdaki biraz uzunca kodu, menu.asp adıyla kaydedin veya bu kitapçığın örnek dosyaları arasında bulun:

```
<%@ LANGUAGE=VBSCRIPT %>
<%
```

```
Server.ScriptTimeOut = 300  'Server'a daha çok zaman tanımak için
strMenuSayfaURL = "/menu.htm" 'oluşturacağımız sayfanın yolu
strListKlasor = "/"      'içindekileri bulacağımız dizin
%>
<HTML>
<TITLE>ASP ILE MENU OLUSTURMA</TITLE>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
<META http-equiv="Copyright" content="Alex Homer">
<BODY>

<%
Response.Write "<P>Menü sayfası oluşturuluyor: " & strMenuSayfaURL & " ...</P>"

'menü sayfası olacak düz yazı dosyasını oluşturalım
Set objFSO = CreateObject("Scripting.FileSystemObject")
strDosyaAdi = Server.MapPath(strMenuSayfaURL)
Set objMenuSayfa = objFSO.CreateTextFile(strDosyaAdi, True) 'dosyanın üstüne yaz

'menü sayfasının başlık bölümünü yazalım
objMenuSayfa.WriteLine "<HTML><BODY><P><B>Dosyaların listesi</B></P>"

'belirtilen dizindeki dosyaların listesini içeren kolleksiyonu oluşturalım
Set objKlasor = objFSO.GetFolder(Server.MapPath(strListKlasor))
Set kolDosyalar = objKlasor.Files

'Her bir dosyanın başlığını okuyarak listemizi yapalım
  For Each objDosya in kolDosyalar

'dosya adının uzantısı ASP ve HTM olanları ayıralım
  strDosyaTuru = objFSO.GetExtensionName(objDosya.Name)
  If (strDosyaTuru = "asp") Or (Left(strDosyaTuru, 3) = "htm") Then

'dosyanın tümünü okuyup bir String'de tutalım
```

```
  Set objOku = objDosya.OpenAsTextStream(1) 'okumak için
  strIcerik = objOku.ReadAll
  objOku.Close

'içinden başlık bölümünü alalım
  strBaslik = ""
  intBaslangic = Instr(UCase(strIcerik), "<TITLE>") + 7
  intSon = Instr(UCase(strIcerik), "</TITLE>")
  If (intBaslangic > 0) And (intSon > intBaslangic) Then
  strBaslik = Trim(Mid(strIcerik, intBaslangic, intSon - intBaslangic))
  End If
  If Len(strBaslik) = 0 Then strBaslik = "Adsız sayfa '" & objDosya.Name & "'"

'Menü sayfası için metni oluşturalım
strBuDosyaURL = strListKlasor & objDosya.Name
strKopru="<A HREF=" & Chr(34) & strBuDosyaURL & Chr(34) & ">" & strBaslik & "</A><BR>"
objMenuSayfa.WriteLine(strKopru)
End If
Next

'Menü sayfasının son bölümünü yazalım
  objMenuSayfa.WriteLine "</BODY></HTML>"
  objMenuSayfa.Close
  Response.Write "<P>Menü sayfası oluşturuldu.</P>"
%>
<P><A HREF="<% = strMenuSayfaURL %>">Menü sayfasını aç</A></P>
</BODY>
</HTML>
```

Programın daha öncekilere göre uzun oluşu, Dosya Sistemi (FileObject) Nesnesini kullanarak oluşturacağı menu.htm dosyasının içeriğini sağlamak için verdiğiniz dizinde bulunan ve dosya adı uzatması .htm ve .asp olan bütün dosyaların içinde <TITLE>..</TITLE> etiketini aramasından kaynaklanıyor. Programın yazarı Alex Homer'in bütün değişken adlarının önünde, değişkenin türünü belirten ön-ekler kullandığına dikkat

edin. Böylece nesneleri (obj), sayısal (int) ve kolleksiyon (kol) değerlerden ayırmamız kolaylaşıyor.

<asp0016.tif>

Programı çalıştırdığımız zaman, Browser penceresinde menü sayfasının oluşturulduğuna ilişkin mesaj belirliyor; ve işlem tamamlandığında da oluşturulan menu.htm sayfasının köprüsü veriliyor. Bu köprüyü tıkladığımızda ise programın beşinci satırında strListKlasor = "/" ifadesiyle verdiğimiz dizinin içindeki bütün HTML ve ASP dosyalarının başlığını, bu dosyalara köprü verilmiş olarak görürüz.

<asp0017.tif>

İsterseniz, strListKlasor değişkeninin değeri olarak, sözgelimi "/html/" gibi, kendi Web Server'ınızdaki diğer herhangi bir dizinin adını da verebilirsiniz. Bunu yaparken dizin adının sonuna bölü işareti koymayı unutmamak gerekir. Bu program böyle bağımsız olarak çalıştırılabileceği gibi, bir çerçevenin (Frame) içinde çalıştırılabilir ve sonuçları, başka bir çerçevenin içinde görüntülenebilir. Bu programın "İçindekiler" sayfasını hazırladığı dizine ne kadar yeni dosya koyarsanız koyun veya mevcut dosyaları çıkartırsanız çıkartın, ne zaman bu programı çalıştırırsanız, İçindekiler listesi dinamik olarak mevcudu yansıtacaktır.

Gecikme Bildirme Sayfası

Yukarıdakı örneği birlikte uyguladıysak, ASP programına İçindekiler listesini çıkartması için verdiğimiz dizinin içerdiği dosya sayısına bağlı olarak, programın çalışması epey uzun süre alacaktır. 20 saniyenin üzerinde beklemenin bir yüzyıla yakın etki yaptığı günümüz Internet ziyaretçisi için bu süre çok uzun görünebilir. Ziyaretçimize, sözgelimi Browser programının donmadığını, veya Internet bağlantısının kesilmediğini belirtebilmek için, "Lütfen bekleyiniz!" mesajı vermek yerinde olabilir. Yukarıdaki örnekte bunu aynı

sayfa içinde yapma imkanımız vardı; çünkü programımız tabir yerinde ise ziyaretçiyi bu sayfadan alıp, başka bir sayfaya götürmüyordu.

Oysa Web uygulamamızın akış planı öyle gerektirebilir ki, ziyaretçimizin bir talebini karşılayabilmek için ona "Lütfen bekleyiniz!" mesajını ne hareket ettiği, ne de gittiği sayfada veremeyiz. Bunun için bir "ara" sayfa gerekebilir; bir tür "Lütfen bekleyiniz!.." sayfası.

Bu örnekte böyle bir sayfa yapacağız; ancak bu amacımızı tek sayfa karşılamayacağı için, ortaya iki sayfa çıkartacağız. Şimdi şu kodu bekle01.asp adıyla kaydedin:

```
<% Response.Buffer = True %>
<HTML>
<% mesaj = Server.URLEncode("Arzu ettiğiniz iş yapılıyor.. Lütfen
bekleyiniz") %>
<% Response.Redirect ("bekle02.asp?BEKLE_SURE=3&BEKLE_MESAJ=" & mesaj & _
"&GONDER_URL=index.htm") %>
<HEAD>
<TITLE>ASP ILE BEKLETME 01</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>
</BODY>
</HTML>
```

Burada ASP kodunun tümü işlemeden, ortaya çıkan HTML'in ziyaretçinin Browser'ına parça parça gönderilmesini önlemek amacıyla Response nesnesinin .Buffer metodunun True (doğru) olarak belirlendiğini görüyorsunuz. Programın kullanacağı mesajın, mesaj adlı değişkene atandığına dikkat edin. Bu sayfanın tek işlevi ise Response nesnesinin .Redirect metodu ile bekle02.asp sayfasını çağırmaktan ibaret. Başka bir deyişle bekle01.asp, gerçekte Browser'ın yüzünü bile göremeyecektir!

Biraz sonra kendisine üç değişken verilmesini bekleyen bekle02.asp programı yazacağız. Nitekim bekle01.asp'nin içinde üç değişken değeri belirleniyor; ve bunlar bekle02.asp'ye URL-GET metodu ile "yazılıyor." Burada GET metodu ile bir bir sayfaya veri gönderme tekniğini hatırlayalım. Bir Form'un Server'a veri gönderme metodu GET ise, Form'dan (veya sayfadan URL yoluyla) derlenen verilerin "değişken=değer" çiftleri halinde ve çiftlerin arasında & işareti konularak gönderilir. Bu yöntemi, HTTP yoluyla Browser nasıl kullanıyorsa, biz de istediğimiz anda kullanabiliriz. Burada;

```
bekle02.asp?BEKLE_SURE=3&BEKLE_MESAJ=" & mesaj & _ "&GONDER_URL=index.htm"
```

şeklindeki ifade de sayfa adından sonra konan soru işareti, Server'a bu bilgi kümesinin bu sayfaya iletilmesi komutunu vermenizi sağlar; bir bakıma Server'a "Şu bilgileri al; bekle02.asp sayfasına ver!" demiş oluyoruz. Bu bilgiler hatırlayacaksınız, Server'da Request nesnesinde tutulur; eşittir işaretinin önündeki kısım değişken, arkasındaki kısım ise değer sayılır.

Dedik ki: bekle02.asp kendinise üç değişken verilmesi beklemektedir. Bunu sağlayan nedir? Resquest nesnesinden alınıp, bu sayfanın içindeki değişkenlere atanan üç değer bulunması. bekle02.asp'yi de yazalım:

```
<%@ LANGUAGE="VBSCRIPT" %>
<%
BEKLE_SURE = Request("BEKLE_SURE")
GONDER_URL = Request("GONDER_URL")
BEKLE_MESAJ = Request("BEKLE_MESAJ")
%>
<html>
<head>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
<META http-equiv="Refresh" content="<% =BEKLE_SURE %>; URL=<% =GONDER_URL %>">
<title>ASP ile Bekletme</title>
```

```
</head>
<body color="#FFFFFF">
<font face="Arial">
<p align = "center"><strong><% =BEKLE_MESAJ%></strong> </p> </font>
</body>
</html>
```

Daha önce yazdığımız bekle01.asp, ziyaretçiyi bekle02.asp'ye yönlendirirken ona üç "değişken=değer" çiftti gönderiyor; bekle02.asp bunları Request nesnesinden alarak, kendi değişkenlerine atıyor. Bu üç değişkenini inceleyelim.

- Ziyaretçinin bekleme süresince Browser penceresinde göreceği mesaj, BEKLE_MESAJ değişkeninde yer alıyor;
- Bu mesajı görüntülendiği sayfanın ziyaretçinin Browser'ında kalacağı süreyi BEKLE_SURE değişkenin değeri belirliyor:
- Bu sayfanın yerini alacak olan hedef sayfa veya ziyaretçinin Browser'ına gönderilecek yeni sayfanın adresini, GONDER_URL değişkeni belirliyor.

Bu programları çalıştırdığınızda ziyaretçinin Browser'ında üç saniye süreyle "Arzu ettiğiniz iş yapılıyor.. Lütfen bekleyiniz!" yazısı görüntülenecek; daha sonra Browser'a index.htm sayfası gönderilecektir. Bu iki programı kendi ihtiyaçlarınıza uyarlamak isterseniz, sözgelimi bir veritabanının güncellenmesi sırasında veya benzeri bir muhtemel gecikme durumunda, ziyaretçiye durumu bildirmek için, köprüyü bekle01.asp'e verebilirsiniz; asıl hedef URL'i ise bekle01.asp'de verebilirsiniz.

Form Değerlerini Yakalayalım

Bir veritabanından çektiğimiz verilerle, Form etiketi içinde kullanabileceğimiz ve ziyaretçinin ya metin girmesine ya da seçim yapmasına imkan veren kutuları, seçenekleri nasıl doldurabileceğimize ilişkin bir örneği birinci kitapçıkta görmüştük. Burada şimdi bir

Form'dan gelen verileri nasıl değişkenlere atayacağımızı ve bunları nasıl kullanacağımızı ele alalım.

Bu amaçla iki sayfa hazırlayacağız: birincisi içinde bir çok tercihler bulunan formlar olan bir HTML dosyası, diğeri ise bu Formun değerlerini alarak, kullanılır hale getirecek bir ASP programı olacak.

Önce FormOrnek.htm adlı şu dosyayı kaydedelim:

```
<HTML>
<HEAD>
<TITLE>Form Ornegi</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>
<H4>Birinci Form</H4>
<FORM ACTION="FormOrnek.asp?HangiForm=1" METHOD="POST">
<INPUT TYPE="Text" NAME="MetinGir" VALUE="Mesajınızı buraya yazınız" SIZE="30"
MAXLENGTH="75"><BR>
<INPUT TYPE="Password" NAME="ParolaAlani" SIZE="30" VALUE="parola"><BR>
<TEXTAREA NAME="MetinAlani" COLS="30" ROWS="3" WRAP="VIRTUAL">
Buraya istediğiniz yazıyı yazabilirsiniz...</TEXTAREA><P>
<INPUT TYPE="Submit" VALUE="Gönder">  <INPUT TYPE="RESET"
VALUE="Sil">
</FORM>
<HR>
<H4>İkinci Form</H4>
<FORM ACTION="FormOrnek.asp?HangiForm=2" METHOD="POST">
<TABLE BORDER = 0><TR><TD><INPUT TYPE="Radio" NAME="Radyo" VALUE="Tercih1"
CHECKED>Radyo Düğmesi Tercih 1<BR>
<INPUT TYPE="Radio" NAME="Radyo" VALUE="Tercih2">Radyo Düğmesi Tercih 2<BR>
<INPUT TYPE="Radio" NAME="Radyo" VALUE="Tercih3">Radyo Düğmesi Tercih 3</TD>
<TD width=30> </TD>
<TD><INPUT TYPE="Checkbox" NAME="IsaretKutusu" VALUE="Isaret1">İşaret Kutusu
1<BR>
```

```
<INPUT TYPE="Checkbox" NAME="IsaretKutusu" VALUE="Isaret2">İşaret Kutusu 2<BR>
<INPUT TYPE="Checkbox" NAME="IsaretKutusu" VALUE="Isaret3">İşaret Kutusu 3</TD>
</TR></TABLE><BR>
<INPUT TYPE="Submit" VALUE="Gönder">  <INPUT TYPE="RESET"
VALUE="Sil"></FORM>
<HR>
<H4>Üçüncü Form</H4>
<FORM ACTION="FormOrnek.asp" METHOD="GET">
<INPUT TYPE="Hidden" NAME="HangiForm" VALUE="3">
<SELECT NAME="SecmeListesi" SIZE="1">
<OPTION VALUE="ListeTercih1">Liste Tercih 1
<OPTION VALUE="ListeTercih2">Liste Tercih 2
<OPTION VALUE="ListeTercih3">Liste Tercih 3
</SELECT>
<SELECT NAME="CokluSecme" SIZE="3" MULTIPLE>
<OPTION VALUE="ListeKutu1">Çoklu Seçme Kutusu 1
<OPTION VALUE="ListeKutu2">Çoklu Seçme Kutusu 2
<OPTION VALUE="ListeKutu3">Çoklu Seçme Kutusu 3
</SELECT><P>
<INPUT TYPE="Submit" VALUE="Gönder">  <INPUT TYPE="RESET"
VALUE="Sil">
</FORM>
</BODY>
</HTML>
```

Daha sonra belirlediğimiz değişkenlerin Form'un hangi öğelerinden geldiğinin rahat anlaşılması için burada Form unsurlarının değeri olarak unsurun adına yakın kelimeleri kullandığımıza dikkat edin. Sayfada üç ayrı Form bulunduğu, ilk iki formun POST, üçüncüsünün ise GET metoduyla Server'a bilgi gönderdiği de dikkatinizi çekmiş olmalı. POST metodunda Server'a giden bilgiler, talep edilen sayfanın adına URL kodlama yöntemiyle (ve dolayısıyla Browser'ınızın URL kutusuna yazılarak) gönderilmez; Fakat ilk iki Form'da POST metodu kullanmış olmamıza rağmen, formu işleyecek programa bilgilerin hangi formdan gittiği sanki GET metodu kullanılmış gibi bildiriliyor. Bu yöntemi,

Request.Form kolleksiyonuna yazılmasını istemediğimiz bilgiler için kullanabibiliriz. Üçüncü Form ise bilgilerini GET yoluyla gönderiyor ve kendisinin hangi form olduğuna ilişkin bilgiyi ise gizli bir değişkene yazıyor. Her üç durumda da HagiForm değişkeninin değeri olarak, ASP programına formun sıra numarası anlamına gelen sayı gönderiliyor. Birinci form 1, ikinci form 2, üçüncü form ise 3 değerini yolluyor. Bu HTML dosyasını Browser'da açarsanız, üç ayrı formu göreceksiniz.

<asp0018>

Her üç formun gönderdiği bilgiler aşağıdaki programa ulaşıyor. Bu programı FormOrnek.asp adıyla kaydedebilirsiniz:

```
<HTML>
<HEAD>
<TITLE>ORNEK FORM SONUCLARI</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>
<%
dim FormTercihi
FormTercihi = Request.QueryString("HangiForm")
Select Case FormTercihi
Case 1
%>
Metin alanı sonuçları<P>
<TABLE BORDER=1 CELLPADDING=5 width=75%>
<TR><TD>Input/Text Alanı</TD><TD><B><%=
Request.Form("MetinGir") %></B></TD></TR>
<TR><TD>Input/Passsword Alanı</TD><TD><B><%=
Request.Form("ParolaAlani") %></B></TD></TR>
<TR><TD>TextArea Alanı</TD><TD><B><%=
Request.Form("MetinAlani") %></B></TD></TR>
</TABLE>
<HR>
```

```
<%
Case 2
%>
Radyo Düğmesi ve İşaret Kutusu Sonuçları<P>
<TABLE BORDER=1 CELLPADDING=5 width=25%>
<TR><TD>Seçilen Radyo Düğmesi</TD><TD><B><%=
Request.Form("Radyo") %></B></TD></TR>
<TR><TD>İşaretlenen Kutular</TD><TD><B><%
dim strIsaretlenen
            for each strIsaretlenen in Request.Form("IsaretKutusu")
            Response.Write strIsaretlenen & "<BR>"
            next
 %></B></TD></TR>
</TABLE>
<HR>
<%
Case 3
%>
Seçme Alanları sonuçları<P>
<TABLE BORDER=1 CELLPADDING=5 width=35%>
<TR><TD>Seçilen Liste Ögesi</TD><TD><B><%=
Request("SecmeListesi") %></B></TD></TR>
<TR><TD>Seçilen Çoklu Liste Ögeleri</TD><TD><B><%
            for each strIsaretlenen in Request("CokluSecme")
            Response.Write strIsaretlenen & "<BR>"
            next
%></B></TD></TR>
</TABLE>
<HR>
<%
End Select
%>
```

Bu programda, üç ayrı formdan gelebilecek bilgiler, HangiForm değişkeninin değerine göre, Select Case döngüsü ile üç ayrı bölümde işleniyor. İlk iki formda, bilgiler

POST metodu ile geldiği için Request.Form nesnesinin içinde, değişken adıyla aranabilir. Üçüncü Formda ise bilgiler Request.QueryString nesnesinde bulunuyor. Burada bu bilgilerin daha önce görmediğimiz kısaltma yöntemiyle alınıp, sayfaya aktarıldığını görüyorsunuz; <%=Request("SecmeListesi")>, aslında;

```
Response.Write Request.QueryString("SecmeListesi")
```

ifadesinin kısaltılmışıdır.

FormOrnek.htm'in her üç formunda da bazı tercihler yaparak Gönder düğmesini tıkladığınızda, FormOrnek.asp'ye gönderilen URL bilgisine dikkat edin. İlk ikisinde sadece seçilen formun HangiForm değeri yer aldığı halde. üçüncüsünde formdaki bütün bilgiler burada URL-kodlanmış olarak yer alacaktır.

<asp0019>

<asp00208>

<asp0021>

Programın bilgi işleyen bölümünü kendiniz irdeleyebilir misiniz?

Konuk Defteri Uygulaması

Bu bölümde ilk tam ASP sitesini adım adım birlikte oluşturacağız; daha önce teorik veya kısaca uygulamalı gördüğümüz komutları, metodları ve ifadeleri toplu halde kullanma imkanı bulacağız.

Konuk Defteri, Internet sitemizi ziyaret edenlerle en kolay bilgi alışverişi yöntemidir; konuklarımız defterimize kendileri hakkında bazı bilgiler girerler. Burada yapacağımız örnekte, ziyaretçimize önce konuk defterimize bilgi girmek isteyip istemediğini soran bir sayfa sunacağız; burada konuk defteri bilgilerinin yer aldığı bir Form sayfasına gitmesini sağlayan köprü olacak. Form sayfamızdaki Gönder düğmesi tıklandığında bir ASP sayfası, Form'un yolladığı bilgileri ziyaretçimize gösterecek ve arzu ederse defterde yer alan diğer bilgileri okuma imkanı verecek. Ziyaretçimiz defterdeki diğer bilgileri okumak isterse, bir diğer ASP sayfası, bu bilgileri dosyadan okuyarak ziyaretçimize sunacak.

Birinci sayfamız burada basit bir HTMLsayfası ancak siz bunu isterseniz kendi ana sayfanızla bütünleştirebilirsiniz. Diyelim ki aşağıdaki kodu, konuk01.htm adıyla kaydettik:

```
<HTML>
<HEAD>
<TITLE>KONUK DEFTERIM</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>
<H1><CENTER>Hoşgeldiniz!</CENTER></H1>
Konuk Defterimi imzalamak ister misiniz?
Çok teşekkürler
<p><A HREF="konuk02.htm">Deftere Git</A>
</BODY>
```

```
</HTML>
```

Burada konuk defterini izmalamak isteyenler için konuk defterini içeren dosyaya bir köprü var. Aşağıdaki form da konuk defterimiz olsun! Bunu da konuk02.htm adıyla kaydedelim:

```
<HTML>
<HEAD>
<TITLE>KONUK DEFTERIM</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>
<H1><CENTER>Konuk Defteri!</CENTER></H1>
Konuk defteri sayfama hoşgeldiniz. Buraya kaydedeceğiniz bilgilerle birbirimizi daha iyi tanıma imkanı bulabiliriz.
<p>Çok teşekkürler</p>
<FORM ACTION="konuk_isle.asp" METHOD="post">
<p>Adınız: <INPUT TYPE="Text" SIZE="20" NAME="Adi"></p>
<p>Soyadınız: <INPUT TYPE="Text" SIZE="20" NAME="Soyadi"></p>
<p>E-Posta Adresiniz: <INPUT TYPE="Text" SIZE="20" NAME="Email"></p>
<p>Düşünceleriniz: <br><TEXTAREA NAME="Mesaj" COLS="40" ROWS="4">Sitem hakkındaki düşünceleriniz</TEXTAREA></p>
<INPUT TYPE="Submit" NAME="Gonder" VALUE="Gönder!"> <INPUT TYPE="Reset" NAME="Sil" VALUE=" Sil! ">
</BODY>
</HTML>
```

Burada oluşturduğumuz HTML alanlarından dördü, daha sonra işlenmek üzere, bize dört değişken verecek: Adi, Soyadi, Email ve Mesaj.

<asp0015.tif>

Aşağıdaki kod ile bu verileri işleyeceğiz. Bu kodları da konuk_isle.asp adıyla kaydedelim:

```
<%
@LANGUAGE=VBscript
```

```
Option Explicit
%>
<HTML>
<HEAD>
<TITLE>KONUK DEFTERI KAYIT</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>
<h2>Verdiğiniz Bilgiler:</h2>
Adınız: <%=Request.Form("Adi")%><BR>
Soyadınız: <%=Request.Form("Soyadi")%><BR>
E-Posta Adresiniz: <%=Request.Form("Email")%><BR>
Düşünceleriniz: <%=Request.Form("Mesaj")%><BR>
<p>
<%
Dim DosyaSistemi, KonukDosyasi
Set DosyaSistemi = CreateObject("Scripting.FileSystemObject")
Set KonukDosyasi =
DosyaSistemi.OpenTextFile("c:\inetpub\wwwroot\konuklar.txt",8, True)
KonukDosyasi.WriteLine Request.Form("Adi")
KonukDosyasi.WriteLine Request.Form("Soyadi")
KonukDosyasi.WriteLine Request.Form("Email")
KonukDosyasi.WriteLine Request.Form("Mesaj")
KonukDosyasi.Close
%>
<H3>Konuk Defterime kaydedildi. Çok teşekkür ederim.</H3>
<A HREF="konuk_oku.asp">Konuk Defterini Oku!</A>   <A
HREF="index.htm">Ana Sayfaya Dön!</A>
</BODY>
</HTML>
```

Burada biraz duralım ve konuk_isle.asp'yi irdeleyelim. Form'u oluşturan sayfamızda dikkat ettiğiniz gibi, verileri POST metodu ile alıyoruz; dolayısıyla Form'dan gelen bilgiler, Request nesnesinin Form kolleksiyonunda yer alıyor; ve önce bu bilgileri ziyearetçimizin

Browser penceresine yazdırıyoruz. Ziyaretçimiz böylece ne bilgi verdiğini ber kere daha görmüş oluyor. Sonra, FileSystem nesnesini kullanarak, bu bilgileri Web Server'ın kök dizininde konuklar.txt adlı dosyaya eklettiriyoruz. Bu işlemi yapan WriteLine metodu, her bir değişkenin değerini yazdıktan sonra satır sonuna yeni satır karakteri girecektir. Yine Bu bilgilerin kaydedildiğini ziyaretçiye bildirdikten sonra, kendisine defterimizdeki diğer girdileri okuma veya ana sayfaya dönme seçeneğini veriyoruz. Ziyaretçi, defterdeki diğer bilgileri okuma seçeneğini seçecek olursa gideceği sayfanın kodları ise aşağıda. Bu kodları da konuk_oku.asp adıyla kaydedelim:

```
<%
@LANGUAGE=VBscript
Option Explicit
%>
<HTML>
<HEAD>
<TITLE>KONUK DEFTERI OKUMA</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>
<h2>Konuk Defterimde Yeralan Bilgiler:</h2>
Bugüne kadar konuk defterimi imzalayan bütün dostlarıma teşekkür ederim.
<p>
<%
Dim DosyaSistemi, KonukDosyasi, Adi, Soyadi, Email, Mesaj
Set DosyaSistemi = CreateObject("Scripting.FileSystemObject")
Set KonukDosyasi = DosyaSistemi.OpenTextFile("c:\inetpub\wwwroot\konuklar.txt",1)
Do While Not KonukDosyasi.AtEndOfStream

                Adi = KonukDosyasi.ReadLine
                Soyadi = KonukDosyasi.ReadLine
                Email = KonukDosyasi.ReadLine
```

```
                Mesaj = KonukDosyasi.ReadLine
                Response.Write Adi & "<BR>"
                Response.Write Soyadi & "<BR>"
                Response.Write Email & "<BR>"
                Response.Write Mesaj & "<P>"
Loop
KonukDosyasi.Close
%>
<A HREF="index.htm">Ana Sayfaya Dön!</A>
</BODY>
</HTML>
```

Şimdi de bu kodun üzerinde duralım: Yine FileSystem nesnesinin OpenTextFile metodu ile, mevcut konuk girdileri dosyasını açıyoruz; ve içeriğini Do döngüsü ile, dosya sonuna varıncaya kadar ReadLine metodu ile okutuyoruz. ReadLine, dosyayı satır –satır okuyacaktır; burada ”satır” ölçüsü, dosya başı ile satırsonu-yeni satır (CRLF veya enter/Return) karakteri, iki satırsonu arası, veya bir satırsonu ve dosya sonu işareti arasında kalan metindir. Dosyamızın her bir satırı bir değişkenin değeri olarak yazıldığına göre, okunan her satır sırasıyla, Adi, Soyadi, Email ve Mesaj değişkenlerine değer olarak kaydedilecek ve ziyaretçininin Brıowser penceresine yazılacaktır.

Bu ana kodun etrafında sayfalarını görsel olarak zenginleştirmek, tabiî sizin elinizde!

Veri Yönlendirmeli Web Uygulaması

Birinci kitapçıkta, ADO nesnesini tanıdık ve ne işe yaradığını gördük. Bu bölümde ADO nesnesinin nasıl kullanıldığını ele alacağız; bu amaçla veritabanına dayanan, HTML etiketlerinin içeriğini bir veritabanından alan ve ziyaretçinin girdilerini bir veritabanına yazan örnek uygulama oluşturacağız. Bunu yaparken ADO'yu tanıtırken ele almadığımız bazı SQL komutları ile tanışacağız.

Örnek uygulamamız, ünlü bir Web Tasarımcısının müstakbel müşterilerine randevu verdiği bir site olacak. Müstakbel müşterilerimiz, Web sitemize bağlanacaklar, nasıl bir Web sitesi oluşturmak istediklerini onlara sunacağımız bir listeden seçecekler; sitelerinin özelliklerini ve bizimle ilk görüşmeyi yapmak istedikleri zamanı belirtecekler; Web programımız, bütün bu bilgileri bir veritabanındaki veri tablolarına dayanarak müstakbel müşterimize seçenek olarak sunacak; ziyaretçinin kabul ettiği randevu zaman dilimini veritabanındaki tabloda başka müşterilere verilmemek üzere, kapatacak. Fakat bu işe başlamadan önce hızlı bir bir veritabanı oluşturma kursu görelim.

Veri için hazırlık

Kullanacağımız veritabanını burada MS Access ile oluşturacağız. Fakat siz istediğiniz herhangi bir programı, örneğin FileMaker Pro veya Paradox'u kullanabilirsiniz. Elinizin altında Oracle veya MS SQL Server varsa, onları kullanmanızdan daha tabiî bir şey olamaz.

İlişkilendirilmiş Veritabanı (Relational Database) deyimini daha öce duymuş olmalısınız. Web Programımızda, böyle bir veritabanı kullanacağız. İlişkilendirilmiş Veritabanı, bir dosya içinde tabloların en az bir sütunundaki kaydın diğer tablolardaki en az bir sütunla aynı olduğu ve bu iki sütun birbirine bağlanmış veritabanı demektir. (Bu örnekte kullanacağımız veritabanının Microsoft Access ile oluşturulmuş örneğini, bu kitapçığın kod örnekleri arasında bulabilir ve bilgisayarınızda kurulu bir veritabanı işleme programı ile inceleyebilirsiniz. Bu tabloları kendi programınızda oluşturmak isterseniz, kayıt alanları ve özelliklerini buradaki örneklere uygun olarak oluşturmalısınız.)

Veritabanı tablolarının ilişkilendirilme durumunu, oluşturacağımız veritabanının iki tablosu üzerinde gösterelim:

<musteri_tablo.jpg>

Bu tabloda, müşterinin kurmak istediği site türünü belirten bir endeks sütunu var: TurNO. (Daha sonra bu veritabanını yükleyeceğimiz Web Server'ın işletim sistemini bilmediğimiz için, Türkçe alfabeyi desteklemeyen bir Server'ın "Tablo veya Alan Bulunamadı" hatası vermesini önlemek için alan adlarında Türkçe karakter kullanmamaya özen göstermemiz gerekir.) Bu kayıt, Site Türleri tablosunda da var:

<tur_tablo.jpg>

Veritabanını oluştururken bu iki tabloyu TürNO sütunlarından ilişkilendirir ve ilerde veritabanına, "Bana, Müşteriler ve Site Türleri tablolarından TürNo sütunundaki veri aynı olan bütün kayıtları bul" şeklinde bir sorgulama emri (SQL JOIN komutu) verecek olursanız, Windows'un ODBC programı, veritabanını inceleyecek ve ortaya çıkartacağı verilerden biri şöyle olacaktır:

ADI	SOYADI	EMAIL	SİTENO	GRAFNO	TURNO	TURADI	TURRAYİÇ
Abdullah	Can	acan@sirket.com.tr	1	1	2	Intranet	4

Şimdi diyebilirsiniz ki, daha sonra iki tabloyu birleştirmek yerine, neden baştan tek tablo yapmadık da, ilişkilendirme ve bir bir yığın SQL komutu yazma (ve tabiî öğrenme) külfetine girdik? Bunun çeşitli sebepleri var: Bir kere veritabanı uzmanları, birden fazla tabloda aynen tekrar eden değerleri, ayrı bir tabloda toplamayı bilgisayarın bellek yönetimi açısından daha etkin bir çalışma tarzı sayarlar. İkincisi, fiyat belirlerken Intranet türü Web siteleri için uygulayacağımız katsayıyı ilerde değiştirmek zorunda kalırsak, (bu iş tutarsa, en kısa zamanda fiyatlara zam yapacağımız şüphesiz olduğuna göre!) ve elimizde içinde 1500 kayıt bulunan bir tablo varsa, herbirinde türRayiç sütundaki veriyi tek-tek güncelleştirmek çok zor olabilir. Oysa bu tür ilişkilendirilmiş bir veritabanında bir tabloda bir kaydı değiştirmekle, bir diğer tablodaki binlerce, hatta milyonlarca kaydı güncelleştirmiş oluruz. Siz kendi veritabanınızı oluştururken, hangi verilerin hangi tabloda toplanacağına ve hangilerinin ayrı tablolara konulacağına şu kriterle karar verebilirsiniz: Bir tablonun birincil endeksine bağımlı olmayan bütün bilgiler o tablonun dışına çıkmalıdır. (Bu kriter size bir ölçüde şifreli görünüyorsa, veritabanı oluşturma ve yönetme konusunda bir kitap edinmeniz yerinde olur!)

Veritabanı konularına aşina olmayanlar için bir iki noktaya daha açıklık getirmek gerekir. Tablolarımızın alan adlarını ve alanlardaki verilerin türlerini gösteren tabloya bakarsanız, bütün tablolarda bir sütunun "Birincil endeks" diye işaretlendiğini göreceksiniz. Bu alan, biri dışında bütün tablolarda otomatik sıra numarası şeklinde; Müşteri tablosunda ise müşterinin elektronik posta adresi şeklinde tayin edilmiş bulunuyor. Bu alanın varlık sebebi, kayıtlarımızda diğer bütün alanlardaki verileri aynı bile olsa her bir müşterinin ayrı

bir varlık olarak korunmasını sağlar. (Tabiî, burada, iki müşterinin aynı elektronik posta adresine sahip olmayacağını varsayıyoruz!)

<randevu_tablo.jpg>

<siteler_tablo.jpg>

<olcu_tablo.jpg>

<graf_tablo.jpg>

İnşaata Başlarken

Biraz daha teknik olarak ifade edersek, inşa ettiğimiz programın (Web sitesinin) fonksiyonu, müşteriden bazı temel bilgileri alıp, bu bilgileri müşteri tablosuna işlemek, müşteriye bizim randevu defterimize (veri tabanındaki Randevu tablosuna) bakarak, boş bir zaman önermek, müşterinin kabul ettiği zaman dilimini Randevu tablosuna işlemek, ve bu zaman dilimini başkasına vermemek üzere kapatmak olacak. Biz, Web sitesi tasarımcısı olarak, sürekli, veritabanı dosyasını açarak, yeni randevu alan olup olmadığını kontrol etmek zorundayız. Yine arada bir, Randevu tablosuna yeni müşteri kabul edebileceğimiz boş zamanlarımızı eklememiz gerekir. Bu işlemi, örneğin Access programını kullanarak yapabiliriz.

Bu amaçla kullanacağımız tabloların ilişkilerini şöyle gösterebiliriz:

<relation.jpg>

Bu şemada görülen ilişkiler şu anda bir anlam ifade etmiyorsa, biraz sonra tablo çok daha aydınlanacaktır. Şimdilik, ya bu tablolara ve ilişkilere sahip bir veritabanı oluşturun; ya da örnek veritabanı dosyasını Kişisel Web Server'ın dizini içine kopyalayın.

ODBC'e Veritabanımızı Bildirelim

Windows'un Denetim Masası'nı açtığınızda, ya ODBC Data Sources, ya da ODBC(32 Bit) adlı bir simge göreceksiniz.

<DATA00001.TIF>

İkinci sekme olan System DSN'i açın ve Add (Ekle) düğmesini tıklayın. Açılacak kutuda, veritabanı dosyanıza uygun sürücüyü seçin. Bu kitapçığın örnek kodları arasında bulacağınız web.mdb adlı dosyayı kullanıyorsanız, Microsoft Access Driver adlı sürücüyü seçin) ve Son düğmesini tıklayın.

<DATA00002.TIF>

Açılacak kutuda Data Source Name (Veri kaynağının adı) kutusuna Web (veya kendi veritabanınızın adını) yazın, isterseniz Description kutusuna veritabanını tanımlayan bir kaç kelime yazabilirsiniz. Database bölümünde Select (Seç) düğmesini tıklayarak veritabanı dosyanızı bulun.

<DATA00003.TIF>

Şimdi, bilgisayarınızın ODBC arayüzü, Kişisel Web Server'ınız (vereceğimiz ASP komutları dolayısıyla) talep ettiği anda Browser'ınıza Web adlı veritabanından arzu edilen verileri seçip verecek veya bu dosyadaki tablolarda (yine ASP komutları ile talep edeceğiniz) güncelleştirmeleri yapacaktır.

<DATA00004.TIF>

Sıra Web Programı'nda

Kuracağımız sitenin işleyiş tarzını veya stratejisini bir kere daha belirtelim: Sitemize ev sahipliği yapan Web Server'da bir veritabanı dosyamız var ve elimizdeki iş durumuna

göre randevu verebileceğimiz, serbest olduğumuz günleri bu veritabanındaki Randevu tablosuna işliyoruz. Ziyaretçilerimiz, ana sayfadan geçtikten sonra, bir Web sitesi yaptğrmek için bizimle görüşmek istiyorlarsa, bazı bilgiler veriyorlar ve bu bilgilere göre, Web programımız, müşteriye tanımladığı siteyi kaça yapacağımızı bildiriyor ve boş olduğumuz zamanların bir listesini sunuyor. Müşteri adayı, bu listeden kendisi için elverişli zamanı seçiyor. Web programımız müşterinin verdiği bilgileri veritabanına işliyor; ve müşterinin seçtiği zamanı randevu tablosunda kapalı hale getiriyor, ki aynı zaman aralığı başka bir müşteriye daha önerilmesin! (Aşağıda, sayfalarımızı adım-adım oluştururken, oluşturduğmuz bazı değişkenlerin ne işe yarayacağı o anda belli olmayabilir. Bunları anlamadan geçmemek için önerim, bu kitapçığın kodlarını edindiyseniz, veritabanını sisteminize tanıtarak, ilgili ASP programını index.htm'den başlayarak bir kaç kere çalıştırmanızdır.)

Önce index.htm sayfamızı yapalım. Bu basit bir "Hoşgeldiniz!" sayfası olacak. aşağıdaki kodu index.htm adıyla kaydedin:

```
<HTML>
<HEAD>
<TITLE>Web Sitesi Yapılır</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=DarkOrange text="white">
<br><br><br><br><br>
<center>
<table width="250">
<tr><td><font face="arial" size="6">Web Sitenize Sahip Olmak
İstiyorsunuz?<br></font>
<p><p><p><font face="arial" size="3">Ama nasıl? HTML, ASP, ADO, ODBC ve daha
bir çok alfabe çorbası öğrenmek zorundasınız
```

```
<p>Ama bunları öğrenmeye zamanınız yok. Aslında öğrenirsiniz öğrenmeye, fakat
zaman meselesi.
<p>Biz çook siteler yaptık, ve size de yardıma hazırız. Bizim işimiz Web sitesi
yapmak!
<p>Sizin de sitenizi yapalım.
<p><p><center><a href="sayfa1.asp">Size de yardımcı olalım</a></center></font>
</td></tr></table></center>
</BODY></HTML>
```

<veriuyg0001.tif>

Bu sayfamızda verdiğimiz mesajı beğenen müstakbel müşterilerimiz, sayfadaki köprüyü tıklayarak, nereye gidecekler? Şu aşağıdaki kodun yer aldığı sayfaya. Bunu da sayfa1.asp adıyla kaydedin:

```
<% @Language = VBscript %>
<HTML>
<HEAD>
<TITLE>Web Sitesi Yapilir</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=darkorange text="white" language=Turkish>
<br><br><br><br><br><center><table width="250"><tr><td>
<font face="arial" size="6">Arzu ettiğiniz siteyi sür'atle
oluşturabiliriz.<p>Önce
nasıl bir site istediğinizi ve grafik malzemenin durumunu belirtin?</font> </P>
</td></tr></table>
<p><p><table width="500">
<form action="sayfa2.asp" method="get">
<TBODY><tr><td width="500" align="middle">
<p><!--#include file="siteler.inc"-->
Kullanılacak grafik malzemenin durumu: <!--#include file="grafikler.inc"--><br>
<br><br><br><p><input type="submit" value="Gönder" align="left"></p>
</td></tr></form></TBODY></table></center>
</BODY>
```

```
</HTML>
```

"Bu sayfada ASP tekniği kullanmayı gerektiren hemen hemen hiç bir öge yok!" diyebilirsiniz; ama demeyin. İki haricî dosyayı #include yoluyla sayfaya eklediğimizi göruyor musunuz? Bu sayfanın bütün işlevi işte bu iki dosyada gizli!

siteler.inc ve grafikler.inc, veritabanından veri çeken ve bunu getirip yukarıdaki yerlere yazmakla görevli olacaklar. Önce siteler.inc'i yapalım:

```
<%
'Siteler tablosundan siteAdi sütunundaki bilgileri al.
SQLSITE = "SELECT siteAdi, siteNO FROM Siteler"
set connsite = server.createobject("ADODB.Connection")
connsite.open "web"
set siteler=connsite.execute(SQLSITE)
%>
<select name="siteNO">
<% 'Listedeki her kayit için su isi yap. %>
<% Do while Not siteler.eof %>
      <Option value = "<%= siteler(1) %>"><%= siteler(0) %></Option>
      <%siteler.MoveNext
Loop%>
</select>
<% siteler.close %>
```

Hatırlarsanız, #include yoluyla sayfaya eklenen dosyanın içeriği ne ise aynen bu komutun olduğu yere yazılmış gibi olur, demiştik. sayfa1.asp'ye bakın; ve yukarıdaki kodu tam siteler.inc'in olduğu yerde düşünün. Şimdi siteler.inc, tek bir iş yapıyor: web isimli DSN'in verdiği bilgiler arasından Siteler tablosundan siteAdi ve siteNo sütunlarındaki bütün bilgileri çekiyor ve bunu HTML'in SELECT etiketinin içini doldurmakta kullanıyor. OPTION'ın nasıl yazıldığını hatırlıyorsunuz değil mi?

Şimdi siteler.inc'in içine bakabiliriz. Önce "SELECT siteAdi, siteNO FROM Siteler" ifadesini ele alalım. Bu, yukarıda sözünü ettiğimiz SQL dili ile, Windows'un ODBC arayüzüne vereceğimiz VBScript komutudur. Peki, bu edindiğimiz veriler nerede duruyor? ASP'nin çalıştığı Web Server'ın bulunduğu işletim sisteminin RAM'inde (veya sanal belleğinde) oluşturulan geçici bir tabloda duruyor. Veritabanındaki Siteler tablosunda birinci sütunda siteNo, ikinci sütunda ise siteAdi alanları var. Burada önce siteAdi'ni okutuyoruz; yani dizi değişkenin siteler(0) adlı birinci ögesi site türü adını, siteler(1) adlı ikinci ögesi ise bu türün numarasını tutuyor. Nitekim, bir Do döngüsü ile bu değerleri SELECT'in ögeleri olarak kullandığımızda önce OPTION'ın VALUE özelliğini siteNo, metnini ise siteAdi ile dolduruyoruz. Ve tabiî açtığımız veri bağlantısını kapatıyoruz: "siteler.close".

Şimdi sayfa1.asp'nin ikinci haricî dosyası olan grafikler.inc'i yazalım:

```
<%
'Grafik durumuyla ilgili bilgileri Grafik tablosundan al.
SQLGRAF = "SELECT grafNO, grafDurum FROM Grafik"
Set conngraf = server.createobject("ADODB.Connection")
conngraf.open "web"
Set graf=conngraf.execute(SQLGRAF)
%>
<SELECT name="grafNO">
<% Do while Not graf.eof %>
<% 'Bu bilgilerle seçenek kutusu yap. %>
            <Option value = "<%= graf(0) %>"><%= graf(1) %></Option>
            <%graf.movenext
loop%>
<% graf.close %>
</select>
```

Bu dosya üzerinde uzun uzadıya durmaya gerek yok; yine aynı DSN'den, fakat bu kez veritabanıın Grafik tablosundan iki alandaki verileri çekiyoruz; Do döngüsü ile yeni bir SELECT'in içini dolduruyoruz. include dosyaları ile birlikte bu sayfa şu görüntüyü veriyor:

<veriuyg0002.tif>

sayfa1.asp'deki formun Gönder düğmesi tıklandığında bu bilgilerin GET metoduyla sayfa2.asp'ye gönderileceğini hatırlayacaksınız. O halde, bu HTML kodunu, sayfa2.htm adıyla kaydedin:

```
<%@ Language = VBscript %>
<%
Dim siteNO
Dim grafNO
Dim SQLSITEADI
Dim connsiteadi
Dim siteadi
siteNO=Request.Querystring("siteNO")
grafNO=Request.Querystring("grafNO")
SQLSITEADI="SELECT siteAdi FROM Siteler "
SQLSITEADI=SQLSITEADI & "WHERE siteNO= " & siteNO
set connsiteadi = server.createobject("ADODB.Connection")
connsiteadi.open "web"
set siteadi=connsiteadi.execute(SQLSITEADI)
%>
<HTML>
<HEAD>
<TITLE>Web Sitesi Üretim Merkezi</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=DarkOrange text="white"><br><br><br><br><br><center>
<table width="250"><tr><td>
```

```
<font face="arial" size="6">Arzu ettiğiniz <%= siteadi(0) %> sitesini
tasarlamaya hazırız.
<p>Bu sitenin kullanım amacı hakkında bilgi verir misiniz?</font>
<form action="icra.asp" method="get">
<input type="Hidden" name="siteNO" value="<%= siteNO %>">
<input type="Hidden" name="grafNO" value="<%= grafNO %>">
<% 'Bu bilgilerle Tür tablosundan bilgi seç. %>
<!--#include file="turler.inc"-->  
<input type="Submit" value="Gönder" align="LEFT">
</form></td></tr></table></center>
</BODY>
</HTML>
```

sayfa1.asp, ziyaretçimizin Form'da yaptığı iki seçimin değerini sayfa2.asp'ye gönderdiğinde, sayfa2.asp bunları otomatik olarak kullanamaz; bilgiler GET yoluyla geldiği için QueryString değişkeninin içine yazılacak bu bilgileri bizim kullanılır hale getirmemiz gerekir. Yukarıdaki kodun şu satırları bu işi yapıyor:

```
siteNO=Request.Querystring("siteNO")
grafNO=Request.Querystring("grafNO")
```

sayfa1.asp'nin gönderdiği siteNo ve grafNo değişkenlerinin değerleri (bu kez bu sayfanın aynı isimdeki) değişkenlerine değer olarak atanıyor. sayfa2.asp'de yapacağımız veri işlemlerinde bu değerleri ölçüt olarak kullanacağız:

```
SQLSITEADI="SELECT siteAdi FROM Siteler "
SQLSITEADI=SQLSITEADI & "WHERE siteNO= " & siteNO
```

Burada aslında tek sorgu deyimi oluşturduğumuz halde, derdimizi neden iki ayrı satırda anlatıyoruz? Daha önce yazdığımız sorgu deyimlerine bakın? Tümü çift tırnak içinde değil mi? Peki, burada siteNo değişkenini tırnak içine alırsak ne olur? Değişken, değişken olmaktan çıkar; metin olur; oysa biz burada siteNo değişkeninin vereceği değeri kullanmak istiyoruz. Bu yolla veritabanından çekeceğimiz değeri hemen aşağıda metnin içinde kullanacağız:

```
Arzu ettiğiniz <%= siteadi(0) %> sitesini tasarlamaya hazırız.
```

Sadece bir değer istediğimiz ve bu değiri ihtiyacımıza uygun seçtiğimiz için veri okutma işini döngüyle yapmıyoruz bu kez; okunan ilk değeri Response.Write metodunun kısaltmasıyla, sayfaya gönderiyoruz. Bu arada ziyaretçimizden ek bilgi almaya devam ediyoruz; yeni bir haricî dosya ile bu kez size tasarlatmak istedikleri sitenin türünü soruyoruz; bu bilgileri ise turler.inc sağlıyor:

```
<%
SQLTUR="SELECT turNO, turAdi FROM Tur"
set conntur = server.createobject("ADODB.Connection")
conntur.open "web"
set tur=conntur.execute(SQLTUR)
%>
<SELECT NAME="turNO">
<% Do While not tur.eof %>
            <Option value = "<%= tur(0) %>"><%= tur(1) %></Option>
            <%tur.movenext
Loop%>
<% tur.close %>
</select>
```

Bu kodun irdelenmesine gerek yok; Tur tablosundan turNo ve turAdi alanlarındaki değerleri alıyoruz ve OPTION etiketinin içini dolduruyoruz. Devam etmeden önce sayfa2.asp'de dikkatinizi çekmiş olması gereken şu iki satıra dönelim:

```
<input type="Hidden" name="siteNO" value="<%= siteNO %>">
<input type="Hidden" name="grafNO" value="<%= grafNO %>">
```

HTML'den hatırlayacaksınız, Form'un içinde Server'a "gizli" (HIDDEN) türü değişken ve değer gönderebiliriz. Buradaki "gizli" kelimesi sizi aldatmasın; Form bir HTML ögesidir ve ziyaretçi Browser'ının kaynağı görüntüleme aracı vasıtasıyla gizli-açık herşeyi görebilir. Buradaki gizlilik sadece bu değişkenlerin sayfada görüntülenmemesinden ibarettir. Bu iki sözüm-ona gizli değişkene, taa kodumuzun başında elde ettiğimiz siteNo değişkeni ile

biraz önce veritabanından çektiğimiz grafNo değişkenlerini atıyoruz. (Merak etmeyin! Hepsini daha sonra kullanacağız.)

<veriuyg0003.tif>

Ve bu kullanımı, Form'umuzun ACTION özelliğinde adı yazılı olan icra.asp yapacak. Şu uzunca kodu bu isimle kaydedin:

```
<%@ Language = VBscript %>
<%
Dim siteNO, grafNO, turNO, randNO
Dim SQLSITETUTAR, SQLGRAFTUTAR, SQLTURTUTAR
Dim bedel, turkatsayi, grafkatsayi, sitekatsayi
siteNO=Request.Querystring("siteNO")
grafNO=Request.Querystring("grafNO")
turNO=Request.Querystring("turNO")
randNO=Request.Querystring("randNO")

SQLSITETUTAR="SELECT siteAdi, olcRayic FROM Siteler, Olcu "
SQLSITETUTAR=SQLSITETUTAR & "WHERE Siteler.olcNO = Olcu.olcNO and siteNO=" &
siteNO

Set conn = server.createobject("ADODB.Connection")
conn.open "web"
Set sitetutar=conn.execute(SQLSITETUTAR)
siteadi=sitetutar(0)
sitekatsayi=sitetutar(1)

SQLGRAFTUTAR="SELECT grafRayic, grafDurum FROM Grafik "
SQLGRAFTUTAR=SQLGRAFTUTAR & "WHERE grafNO=" & grafNO

Set graftutar=conn.execute(SQLGRAFTUTAR)
grafkatsayi=graftutar(0)
grafdurum=graftutar(1)
```

```
SQLTURTUTAR="SELECT turRayic, turAdi FROM Tur "
SQLTURTUTAR=SQLTURTUTAR & "WHERE turNO=" & turNO

set turtutar=conn.execute(SQLTURTUTAR)
turkatsayi=turtutar(0)
turadi=turtutar(1)
conn.close
SET conn = Nothing
bedel = 100 * turkatsayi * grafkatsayi * sitekatsayi
%>

<HTML>
<HEAD>
<TITLE>Web Tasarim Merkezi</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=DarkOrange text="white"><br><br><br><br><br><center>
<table width="250"><tr><td>
<font face="arial" size="6">Çok güzel!<p>Grafik malzemesi <%= grafdurum %> olan
<%= siteadi %> ve <%= turadi %> amaçlı bir Web sitesi, için ücret US$<%= bedel %>
olacaktır.</font></td></tr></table><br><br><br><br>
<table width="500"><tr><td width="150" valign="top"><font face="arial"
size="5">İlk görüşme için iki saatlik randevu almanız gerekir.</font>
<p><font face="arial" size="3">(Açık olan randevu tarihi ve saatinden
beğendiğinizi tıklayınız)</font></td>
<td width="50"></td>
<td width="300" valign="top"><!--#include file="rand.inc"--></td></tr></table>
</center>
</BODY>
</HTML>
```

Adı icra.asp olduğuna göre, bu sayfa çok iş icra ediyor olsa gerek! Gerçekten de bu sayfa, daha önceki sayfalarda elde ettiğimiz bütün bilgileri kullanarak ziyaretçi sitesini kaç paraya yapacağımızı hesap edecektir. icra.asp, önce Querystring'den alacağı bilgileri, kendi

işine yarayacak değişkenlere yerleştirecek ve bunlarla üç katsayı hesaplayacak ve bu katsayıları kullanarak müşterinin sitesi için bir bedel çıkartacak ("bedel = 100 * turkatsayi * grafkatsayi * sitekatsayi"). Sonra "bedel" değişkeninin içindeki değeri müşteriye Dolar olarak bildirecek (Bedel formülünde 100 yerine mesela 600000 rakamını, veya bu işi yaptığınız andaki Dolar'ın TL cinsinden kurunu gösteren rakamı, kullanarak, fiyatı Dolar yerine TL olarak da bildirebilirsiniz). icra.asp daha sonra rand.inc'in yardımıyla veritabanından boş saatlerimizi seçerek müşteriye randevu alması için sunacaktır. rand.inc, veritabanımızın Randevu tablosundan serbest zaman dilimlerini okumak ve elde edeceği sonucu icra.asp programına vermektedir. Şu kodu rand.inc adıyla kaydedelim:

```
<%
SQLRAND="SELECT randNO, randZaman FROM Randevu WHERE randDurum = 'SERBEST'
ORDER BY randZaman"
Set connrand = server.createobject("ADODB.Connection")
connrand.open "web"
Set rand=connrand.execute(SQLRAND)
%>
<font face="arial" size="5">
<% do while not rand.eof
%>
<a href="rezerv.asp?siteNO=<%= siteNO %>&grafNO=<%= grafNO %>&turNO=<%=
turNO %>&randNO=<%= rand(0) %>&randRayic=<%= bedel %>"><%= rand(1) %></a><br>
<%rand.movenext
loop%>
</font>
<% connrand.close
SET connrand = Nothing
%>
```

Bu haricî dosyamız, öncekilerden farklı: bir OPTION etiketini değil; fakat bir Anchor etiketinin içini dolduruyor. Bu etikete dikkat edersek, HREF özelliğinin değeri olan

rezerv.asp'ye aslında bir çok bilginin gönderilmesine de yaradımcı oluyor, Herhangi bir ziyaretçinin seçimleri sonucu oluşacak bir örnek şu olabilirdi:

```
<a href="rezerv.asp?siteNO=1&grafNO=1&turNO=1&randNO=10&randRayic=500">
05.05.2000 14:00:00</a><br>
```

<veriuyg0004.tif>

Bu, rezervasyon yaptıracak olan sayfaya, siteNo, grafNo, randNo ve biraz önce hesapladığımız randRayic değişkenlerini gönderiyor. Bu sayfa ise, aşağıdaki kodlarla kaydedeceğimiz rezerv.asp:

```
<%@ Language = VBscript %>
<%
Dim siteNO, grafNO, turNO, randNO, randRayic
Dim SQLZAMAN, connzaman, zaman, randzaman
siteNO=Request.Querystring("siteNO")
grafNO=Request.Querystring("grafNO")
turNO=Request.Querystring("turNO")
randNO=Request.Querystring("randNO")
randRayic=Request.Querystring("randRayic")

SQLZAMAN="SELECT randZaman FROM randevu "
SQLZAMAN=SQLZAMAN & "WHERE randNO=" & randNO

Set connzaman = server.createobject("ADODB.Connection")
connzaman.open "Web"
Set zaman=connzaman.execute(SQLZAMAN)
randzaman=zaman(0)
connzaman.close
%>
<HTML>
<HEAD>
<TITLE>Randevu Defteri</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=DarkOrange text="white"><br><br><br><br><br><center>
<table width="250"><tr><td>
<font face="arial" size="6">Lütfen elverişli randevu zamanı olan<font
color="blue"><%= randzaman %></font> için rezervasyon yapmak üzere gerekli
bilgileri giriniz.</font>
<p><font face="arial" size="4">
<form action="guncelle.asp" method="get">
<input type="Hidden" name="siteNO" value="<%= siteNO %>">
<input type="Hidden" name="grafNO" value="<%= grafNO %>">
<input type="Hidden" name="turNO" value="<%= turNO %>">
<input type="Hidden" name="randNO" value="<%= randNO %>">
<input type="Hidden" name="randRayic" value="<%= randRayic %>">
<input type="Text" name="adi" size="20"><i>Adınız</i>
<p><input type="Text" name="soyadi" size="20"><i>Soyadınız </i>
<p><input type="Text" name="email" size="20"> <i>e-adresiniz</i>
<p><input type="Submit" value="Gönder">
</form>
</td></tr></table>
<p><p></center>
</BODY>
</HTML>
```

Gerçi bu sayfanın sadece ziyaretçiye seçtiği randevu tarih ve saatini bildirdiğini ve ziyaretçiden bize adını, soyadını ve elektronik mektup adresini vermesi istediğini göreceksiniz, ama gerçekte bu sayfa şu ana kadar oluşturduğumuz veri kümesini veritabanına yazmak üzere hazırlık yapmaktadır. Gizlenmiş (HIDDEN) değişkenlerin görevi bu.

<veriuyg0005.tif>

Derlenen bu değişkenlerin tümü, ziyaretçinin gireceği bilgiler dahil, guncelle.asp'ye gönderilecek. O halde o kodu yazalım:

```
<%@ Language = VBscript %>
<%

Dim siteNO, grafNO, TurNO, randNO, randRayic, adi, soyadi, email
Dim SQLINSERT, connupdate, SQLUPDATE, URL, simdi
siteNO=Request.Querystring("siteNO")
grafNO=Request.Querystring("grafNO")
turNO=Request.Querystring("turNO")
randNO=Request.Querystring("randNO")
randRayic=Request.Querystring("randRayic")
adi=Request.Querystring("adi")
soyadi=Request.Querystring("soyadi")
email=Request.Querystring("email")

SQLINSERT="INSERT INTO Musteriler (adi, soyadi, email, siteNO, grafNO, turNO) "
SQLINSERT=SQLINSERT & "VALUES ("
SQLINSERT=SQLINSERT & "'" & adi & "', "
SQLINSERT=SQLINSERT & "'" & soyadi & "', "
SQLINSERT=SQLINSERT & "'" & email & "', "
SQLINSERT=SQLINSERT & siteNO & ", "
SQLINSERT=SQLINSERT & grafNO & ", "
SQLINSERT=SQLINSERT & turNO & ") "

set connupdate = server.createobject("ADODB.Connection")
connupdate.open "web"
connupdate.execute(SQLINSERT)
simdi = FormatDateTime(now,vbLongDateTime)

SQLUPDATE="UPDATE Randevu SET "
SQLUPDATE=SQLUPDATE & "email = '" & email & "', "
SQLUPDATE=SQLUPDATE & "randNezaman = '" & simdi & "', "
```

```
SQLUPDATE=SQLUPDATE & "randDurum = 'DOLU', "
SQLUPDATE=SQLUPDATE & "randRayic = " & randRayic
SQLUPDATE=SQLUPDATE & " WHERE randNO =" & randNO

connupdate.execute(SQLUPDATE)
connupdate.close
SET connupdate = Nothing

URL="son.asp?adi=" & adi
Response.Redirect (URL)
%>
```

Programı çalıştırdıysanız, guncelle.asp'nin Browser'ın yüzünü bile görmediğini farketmiş olmalısınız; Yukarıdaki kodun, ziyaretçiye kendi yerine başka bir sayfayı gönderdiğini nereden anlıyoruz? En sondaki "Response.Redirect" komutundan. Bu komut ziyaretçiye, son.asp sayfasını yolluyor. Fakat bu sayfaya geçmeden önce yaptığımız güncelleme işleminin üzerinde duralım. Bu programda uzunca bir SQL INSERT deyimi hazırlıyoruz:

```
INSERT INTO Musteriler (adi, soyadi, email, siteNO, grafNO, turNO) VALUES (adi,
soyadi, email, siteNO, grafNO, turNO)
```

Daha önce SQL dilinin SELECT deyimini öğrenmiştik. INSERT de bir veritabanına veri ekleme işini yapan SQL deyimidir. Deyimin INTO bölümüne tablonun adını ve hangi alanların bulunduğunu; VALUES bölümüne ise bu alanlara atayacağımız değerleri yazarız. Bizim programımızda değerler, değişkenlerden alınacağı için dikkatli bir yazma işlemi gerekiyor. Bu yüzden değişken adları ile & (ve) işareti ve virgülü birbirinden kolayca ayırt etmek için uzunda bir yazma yöntemi kullanıyoruz. Hazırladığımız bu deyimle yeni müşteriyi müşteriler tablosuna ekliyoruz. Programımızda bir de SQL UPDATE deyimi var. Bu deyimle, mevcut Randevu tablomuzda müşterinin seçtiği zaman aralığına ait girdiyi, SERBEST'ten DOLU'ya çeviriyoruz ve bu zamanı kime ayırdığımızı, bu görüşmenin saat

ücreti olan rayici tabloya işliyoruz. Bu amaçla oluşturduğumuz SQL UPDATE deyimi şöyle gösterilebilir:

```
UPDATE Randevu SET email = "email", randNezaman = "simdi", randDurum = "DOLU",
randRayic = "randRayic" WHERE randNO = "randNO"
```

SQL'in UPDATE deyiminin de bölümleri vardır. SET bölümünde hangi alana ne değeri gireceğimizi belirtiriz. Normal bir SQL deyiminde alan adının kanşısına bu alana yazılacak değer girerken, burada olduğu gibi değerleri değişkenlerden de alabiliriz. Burada gösterilen kelimeler değişken adı ise tırnak dışında yazılmalıdır. Bunu sağlamak ve aralarına gereken virgülleri koyabilmek için, yine uzunca bir deyim yazıyoruz.

Şimdi, aşağıdaki kodu son.asp adıyla kaydedelim:

```
<%@ Language = VBscript %>
<%
adi=Request.Querystring("adi")
%>
<html>
<head>
<title>Randevunuz Kesinlesti</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=DarkOrange text="white">
<br><br><br><br><br><center>
<table width="250">
<tr><td><font face="arial" size="6"><%= adi %>, sizinle verimli bir işbirliği
yapacağımızdan eminiz. Çok teşekkürler. Görüşmek üzere.</font></td></tr>
</table>
</BODY>
</html>
```

Bu sayfa, ziyaretçimize bir önceki sayfada kaydettiğimiz adi değişkenin değeri alarak, yani müşteriye adıyla hitabediyor ve randevusunun kesinleştiğini bildiriyor; ve veri-yönlendirmeli Web uygulamamızın da sonu oluyor.

<veriuyg0006.tif>

İşte hepsi bu. Ziyaretçimizden veri aldık; veritabanından veri çektik; bunları sayfalarımızda kullandık ve veritabanına işledik. Fakat bu ASP programlarında, veritabanıyla ilgili unsurların dikkatten kaçmaması için, güvenlikle ilgili önlemler alınmadığına, ziyaretçinin girdiği değerlerin denetlenmediğine dikkat etmiş olmalısınız. Normal olarak Internet'e koyacağımız ASP programlarımızda güvenlikle ilgili bölümler olması gerekir. Bu ve diğer program-konulu unsurları son bölümde ele alacağız.

ASP'de Güvenlik

Internet'te iyiniyetli olunmaz. Acı, ama gerçek. Internet'e içine zararlı kod yerleştirilebilecek bir Form koyarsanız, emin olmalısınız ki, birisi bu zararlı kodu koyacaktır. Sadece kötüniyetli kişilerin size söz gelimi elektronik posta adresi veya mesaj görünümünde zararlı kod göndermesini önlemek için değil, fakat normal kullanıcıların yapabilecekleri sıradan hataları yakalamak ve düzeltmek için de önlem almanız gerekir.

Server güvenliği son derece önemli bir konu olmakla birlikte, bunu sağlamak genellikle tasarımcının sorumluluğunda değildir. Bununla birlikte sayfalarımızın ve veritabanımızın güvenliği ve Web uygulamalarımızın doğru işlemesi bizden sorulur.

Bir form ile sizin sayfalarınıza veya veritabanınıza ne gibi zararlar verilebilir? Bu zararların başında, en hafifinden sizin sayfanızı başka yere yönlendirmek gelebilir. Kötüniyetin ölçüsü arttıkça bu, Server programının açıklarından yararlanarak, Server'daki dosyaları değiştirmeye veya tahrip etmeye kadar gidebilir. Normal ziyaretçi hataları arasında ise sözgelimi elektronik posta adresini iki @ işaretiyle yazmaktan tutun, bu işareti koymamaya, metktubun gideceği bilgisayarın adresini eksik yazmaya kadar uzanan bir dizi yanlışlık bulunabilir. Ziyaretçiler formu eksik doldurabilirler. Bu gibi eksiklikler ve yanlışlıkları daha sonra düzeltmek zaman kaybına yol açar. Kimi zaman eksik doldurulmuş bir Form, bu bilgilerin ulaştırılacağı ASP programında hataya yol açabilir. Bu sebeple, özellikle Form yoluyla alınan bilgilerin denetimi ve doğrulatılması şarttır.

Bir Form'un içerdiği bilgilerin denetimi ve doğrulanması iki yerde yapılabilir: istemci-tarafında, sunucu-tarafında. İstemci tarafında, yani ziyaretçiye göndereceğimiz HTML sayfasının içinde yer alan kodlarla yapacağımız denetim-doğrulama, hızlı çalışır; ve ziyaretçi ile sunucu arasında iletişim kurulmadan önce yapılır; böylece ziyaretçinin

sözgelimi gönder düğmesini tıkladıktan sonra çok beklemesi gerekmez. Ne var ki bu yöntemde denetim-doğrulama kodu ziyaretçiye gönderilmiştir; kötü niyetli kişi neyi denetlediğinizi görecek ve isterse bu denetimi kaldıracak size zararlı kod parçacıkları gönderebilecektir. Denetimin sunucu tarafında yapılması, belki biraz daha fazla zaman harcamayı gerektirir, fakat güvenlik açısından daha etkili olabilir.

Bu bölümde ASP programlarınızın güvenliği ve etkinliği açısından almanız gereken bir çok önlemden ikisini, elektronik posta adresi doğırulama ile mesajların içinden zararlı kodları ayıklama yöntemlerini görelim.

Elektronik Adres Doğrulama

Ziyaretçimizin doğru adres verdiğini, bu adresin geçerli bir elektronik posta alıcısına ait olduğunu doğrulamamız mümkün değil, ama en azından adresin doğru yazılıp yazılmadığını denetleyebiliriz. Bunu yapmanın bir yolu elektronik adresin içinde @ karakteri ile en az bir adet nokta bulunduğunu garantilemek olabilir.

Bunu denetleyecek kodu içeren aşağıdaki örnek kodu email.asp adıyla kaydedin:

```
<% @Language = VBscript %>
<HTML>
<HEAD>
<TITLE>E-Mail Form'u</TITLE>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254"><%
If Not IsEmpty(Request.Form("Email")) Then
 strEmail = Request.Form("Email")
 If dogruMu(strEmail) Then
            Response.Write strEmail & " adresini aldık; teşekkür ederiz.<BR>"
'           .....[BURAYA DIĞER KODLAR GİRECEK].....................
 Else
```

```
  Response.Write strEmail & " adresi doğru görünmüyor.<BR>"
 End If
End If
%>
<FORM "Name="Email" Action="email.asp" Method="post">
Enter an email address:
<INPUT Name="Email" Type=Text>
<BR>
<!- - Buraya formun diğer unsurları girecek - - >
<INPUT Type=Submit Value="Gönder">
</FORM>
<SCRIPT RUNAT=SERVER LANGUAGE=VBScript>
Function dogruMu (byval adres)
 AtIsareti=0                    'sayaç olarak kullanacağımız
 Nokta=0          'değişkenleri sıfırlayalım
 dogruMu=false                  'Fonksiyonun değerini yanlış olarak belirleyelim
 KacKarakter=len(adres)         'adresin boyutunu bir değişkene atayalım
 For i=1 to KacKarakter         'döngüyü başlatalım
  karakter=mid(adres, i, 1)     'sayacın gösterdiği karakteri alalım
  if karakter="@" then          '@ işareti olup olmadığına bakalım
   AtIsareti=AtIsareti + 1      '@ işareti ise sayacı bir arttıralım
  End If
  if karakter="." Then          'nokta işaretini arayalım
   Nokta=Nokta + 1              'nokta ise nokta sayasını bir arttıralım
  End if
 Next              'bir sonraki karaktere geçelim
If AtIsareti=1 and Nokta >0 Then            'Bir @ ve en az bir nokta olduysa
dogruMu=true                    'Fonksiyonun değerini doğru yapalım
End If
End Function
</SCRIPT>
</HTML>
```

Bu programı, bu şekliyle sınama amacıyla çalıştırabilirsiniz. Fakat daha sonra programlarınızda kullanmanız gerekirse, bazı değişiklikler yapmanız gerekir. Bu değişiklikleri kodu inceledikten sonra ele alalım.

Sayfamızda tek elemanlı bir Form var ve sınama amacıyla buraya elektronik posta adresimizi yazabiliriz. Sayfa açıldığında çalışan VBScript'in kendi-içinde varolan isEmpty fonksiyonunu bir If döngüsü içinde çağırarak, kendi içindeki formdan kendisine bir değer gelip gelmediğine bakacak; değer olmadığını görünce formu sunacaktır. Forma herhangi bir şey yazıp, Gönder düğmesini tıkladığımız zaman ASP kodumuz, bu kez dogruMu adlı fonksiyona formdan gelen Email değişkeninin değerini vererek sonucu bekleyecektir.

<aspguv0001.tif>

dogruMu fonksiyonu güvenlik kaygısıyla Server'da çalışan sunucu-tarafı Script'tir; metni ve sonuçları kesinlikle kullanıcıya gönderilmeyecektir. Bu fonksiyon, kendisine aktarılan değişken değerinde, içiçe iki If döngüsü ile @ ve nokta işaretlerini arayarak sayacaktır. Bu sayımın sonucu iki değişkenin değerleri arzu ettiğimiz sayıda (AtIsareti bire eşit ve Nokta sıfırdan büyük) ise kendisini çağıran satıra True (doğru), değilse False (yanlış) değerini gönderecektir. Biliyoruz ki, bir fonksiyon doğru sonuç verirse, If döngüsü birinci komuttan, yanlış sonuç verirse Else bölümünden yoluna devam eder. Bu örnekte, dogruMu fonksiyonu doğru sonuç verirse Response.Write metoduyla Browser penceresine elektronik posta adresinin alındığına ilişkin teşekkür mesajı yazdırılacak; yanlış sonuç verirse, adresin yanlış olduğu bildirilecektir.

Gerçek Internet uygulamasında bu tür bir sınama yapacağınız zaman, yukardaki programın fonksiyonu içeren SERVER SCRIPT ("<SCRIPT RUNAT=SERVER....>" diye başlayan ve "</SCRIPT>" diye biten) bölümünü aynen sayfanızın herhangi bir yerine koyabilirsiniz. Daha sonra ziyaretçiden gelecek elektronik posta adresine göndermede

bulunduğunuz ilk yerde ve bu adresi tutan değişken ile herhangi bir işlem yapmadan, örneğin veri tabanına yazmadan veya programın içinde bir başka şekilde kullanmadan önce, sadece şuna benzer bir kod bölümü yazmanız yeter:

```
strEmail = Request.Form("Email")
 If dogruMu(strEmail) Then
.....[BURAYA KODLAR GİRECEK].....................
 Else
  Response.Write strEmail & " adresi doğru görünmüyor.<BR>"
 End If
```

Programınızın gereği olarak döngünün ELSE bölümünü değiştirebilirsiniz.

Zararlı Kod Temizleme

ASP-uyumlu Server dendiği zaman akla ilk gelen Microsoft Internet Information Server, site güvenliği açısından sürekli sınanan ve açığı bulunduğu taktirde MS'un yama programları ile bu açığı giderilen bir sunucu programı olarak bilinir. Bu, IIS'in ve diğer MS ürünü Web sunucu programlarının (MS Site Server ve MS Transaction Server ürünlerinin) "kurşun geçirmez" olduğu anlamına gelmez. Fakat sunucu program üreticisi ne kadar önlem alırsa alsın, sizin sitenize, kulllanıcıların yollayacağı zararlı kodları önlemeye çalışmayacaktır. Bu bir zorunluktur. Sizin için zararlı kod sayılan metin, bir başka Internet sitesi için normal kullanıcı girdisi olabilir. Server programını geliştirenler sizin ne tür girdiyi zararlı sayacağını bilemezler. Bunu siz bilmek zorundasınız.

Son yılların deneyimleri gösteriyor ki, başka yollarla Internet sitenizin kök dizinine ulaşmaya çalışan Hacker tiplerin dışında kalan zararlı kullanıcılar (Hacker özentisi kişiler!), genellikle bir veritabanına gittiğini anladıkları Form bilgilerinin içine Server Side Include (sunucu tarafında çalışacak haricî dosya) içeriği görevini yapacak kod parçacıkları katabilirler. Bu amaçla kullanılabilecek zararlı bir kod şöyle olabilir:

```
<script language="JavaScript"><!--
function reload() {
self.focus();
document.location.href = "http://www.geocities.com/rasimy";
}
setTimeout("reload();", 5000);
// --></script>
```

Bu kodun bütün zararı, içerdiği veritabanını ASP sayfasının herhangi bir etiketi içinde kullanıldığı taktirde ziyaretçiyi burada yazılı URL'e götürmesi ve buradaki sayfayı Browser'ınızda görüntülemek olacaktır. Acemi Hacker'ın ziyaretçinizi zorla götürdüğü yeni Internet sitesi, sadece içerik bakımından sakıncalı olmayabilir; ziyaretçiniz açısından gerçekten tehlikeli bir alan olabilir; göndereceği çerezlerle (cookie) ziyaretçinin bilgisayarında zararlı işler yapabilir. Ziyaretçi bu tuzağa sizin sitenizde bulunduğu sırada düştüğü için, sorumluluk, en azından manevi olarak, size ait demektir.

Zararlı kodlar genellikle bu örnekte olduğu gibi büyüktür-küçüktür işaretleri, düz ve ters bölü ile kesme işareti içerir. Örneğin bir konuk defterinde ziyaretçimizden ne gibi bilgi istersek isteyelim, yazacağı metinde "<" ve ">" işaretlerinin bulunmaması gerekir. Normal bir mesajda, ancak kod yazarken kullanılan bu işaretlerin yeri olamaz. Hatta daha hassas bir düşünceyle, normal ve kısa bir konuk defteri mesajında tek ve çift tırnak, noktalı virgül satır başı (CR) ve yeni satır (LF) işaretleri de bulunmamalıdır. Eğer normal bir ziyaretçi bu gibi işaretlerle dolu bir mesaj yazmışsa, kötü niyetli kişilere karşı alacağımız önlemlerle bu iyiniyetli mesaj da yazarının verdiği biçimi kaybedecektir. Bir kaç ziyaretçimizin konuk defterine fiyakalı yazılar yazmasını sağlamak için, güvenlikten vaz geçmemek gerektiğine göre, "Kurunun yanı sıra yaş da yanar!" demekten başka bir çare düşünmek kolay değil!)

Daha önceki bölümlerde yazdığımız kodları hatırlıyorsanız (veya kimse bakmazken o sayfaları çevirir bakarsanız!), ziyaretçilerimizin Form yoluyla gönderdikleri verileri iki

yolla alır ve bir değişkene yazarız: QueryString ve Request.Form. İşte size bir Internet'te gerçekten kullanılan bir konuk defterinin konuk girdilerini veritabanına işleyen bölümü:

```
<%
Ad = Request.Form("Isim")
Mektup = Request.Form("MailAdr")
Gorus = Request.Form("Mesaj")
Defter = "INSERT INTO mesaj (Isim, Email, Mesaj) VALUES ( "&" '" & Ad &"' , "&"
                '" & Mektup &"' , "&" '" & Gorus &"')"
Set Veriyolu = Server.CreateObject("ADODB.Connection")
Veriyolu.Open("DSN=misdefter;")
Set RS = Veriyolu.Execute(Defter)
%>
```

Tasarımcı, Form'dan gelen üç değişkenin (Isim, MailAdr ve Mesaj) değerini, yeni üç değişkene atamakta ve bunları oluşturduğu SQL INSERT deyimi yoluyla "misdefter" adlı veriye yazdırmaktadır. Tasarımcının dünyayı sadece iyi insanlardan oluşan bir cennet sandığı, kendisini tanımasanız bile, bu koddan anlaşılıyor. Dünyayı cennet yapmak elimizde olmakla birlikte, henüz bütün insanlar bu konuda görüşbirliği içinde olmadığına göre, ziyaretçiden gelen Form değişkenlerini atadığımız yeni değişkenlerin değerlerini elden geçirmeli ve içindeki zararlı kodları ayıklamalıyız, ki sonra misafir defteriniz kevgire dönmesin!

Bu işlemi, sözgelimi yukarıdaki kodun sadece birinci değişkeni (Ad) için yapalım. Buradaki Ad = Request.Form("Isim") satırını atacağız ve yerine şu 14 satırı koyacağız:

```
Ad = Trim(Request.Form("Isim"))'önündeki sonundaki boşluklari atalım
If Len(ad) Then
'**************** BURADA ISTENMEYEN KARAKTERLERI AYIKLIYORUZ *****************
 Ad = Replace(Ad, Chr(10) , "") 'satır başı
 Ad = Replace(Ad, Chr(34) , "") 'çift tırnak
 Ad = Replace(Ad, "'" , "")   'tek tırnak
 Ad = Replace(Ad, Chr(13) , " ") 'yeni satır
 Ad = Replace(Ad, "<" , " ")   'küçüktür
```

```
Ad = Replace(Ad, ">" , " ")    'büyüktür
Do While Instr(Ad, " ")     'iki aralık
 ad = Replace(Ad, " " , " ")
Loop
Ad = Trim(Ad)  'önündeki sonundaki boşlukları da attık mı tamam
End If
```

Bu kod, Ad değişkenini, VBScript'in kullanılmaya hazır Replace() fonksiyonundan geçirerek, içindeki istenmeyen karakterleri ya yok edecek ya da aralıkla değiştirecektir. Böylece eğer kötüniyetli bir kişi, Form bilgisi olarak bize kod olarak kullanılacak bir takım metinler yollamışsa, kodun ana içeriği yerinde kalmakla birlikte, kod olarak kullanılmasını sağlayacak işaretler yok olacağı için kod düz yazıya dönmüş olacaktır.

ADO Güvenliği ve Hata Mesajları

MS Internet Information Server (IIS) ve diğer Web Server programları, istemcinin siteye gönderebileceği talepleri hem kendi açılarından, hem de işletim sistemi açısından belirleme ve sınırlama imkanı sağlarlar. Windows NT işletim sistemi bakımından Internet ziyaretçisi herhangi bir ağ kullanıcısından farksızdır ve ulaşabileceği sayfalar (dosyalar) ve bu dosyalarla yapabileceği işler, "Internet Kullanıcı Hesabı" denen kullanıcının haklarına bağlıdır. IIS'i kuran Web Yöneticisi, bu hesaba istediği gibi haklar kazandırabilir veya sınırlamalar getirebilir. Burada iki noktayı birbirinden dikkatle ayırmak gerekir:

Web ziyaretçisinin bir Web Sitesi'nde kullanabileceği haklar HTTP kaynakları ile sınırlıdır. İşletim sistemi ile ilgili haklar bunun dışındadır. Fakat ikisinin çakıştığı nokta, özellikle veritabanına dayanan Web sitelerinde, veritabanı dosyasının güncelleştirilmesi sırasında ortaya çıkar. HTTP'nin ziyaretçiye sağlayabileceği bütün okuma-yazma hakları tanınsa bile, işletim sistemi bir veritabanı dosyasının yeniden yazılmasına izin vermeyebilir.

Bunu, önceki bölümlerde veritabanı dosyasını güncelleştirme örneklerini yeniden çalıştırarak sınayabilirsiniz. Kullandığınız veri tabanını sözgelimi Windows Gezgini'nde bulun ve sağ tıklayarak "Salt okunur" hale getirin ve ilgili ASP programını çalıştırın; ODBC hata mesajı verecektir.

Bu noktada Web Tasarımcısı olarak akılda tutacağımız ilke şudur: HTTP izinleri ne olursa olsun, eğer işletim sisteminin izinleri daha kısıtlayıcısı ile, işletim sisteminin dediği olur.

Tasarımcı olarak bunun bize etkisi, genellikle Web sitemize evsahipliği yapan Server işletmecisinin veritabanı dosyalarımıza yazma-okuma izni vermesini sağlamaktır. Web sitesi evsahibi (Hosting) firması, veritabanı dosyasına DSN ayarı yaparken bu izni verecektir. Ancak DSN kaydı yaptırıldığı halde veritabanınız ziyaretçilerinize ODBC hata mesajı veriyorsa evsahibi firmanın yönetimine, veritabanı dosyasının "haklarını" yeniden belirlemesini hatırlatmanız gerekebilir.

Veritabanı ile yönlendirilmiş Web sayfalarımızda en sık aldığımız hata mesajı:

```
[Microsoft] [ODBC Driver Manager] Data source name not found
```

olsa gerek. Bu mesaj, veritabanının ODBC'ye tanıtılması ile ilgili işlemde arıza olduğunu gösterir. Ya veritabanı dosyasının sürümü, ODBC sürücülerinin kapsamı dışındadır, ya da ODBC sürücüleri eksik kurulmuştur. Bunu, Windows'un Denetim Masası'nda ODBC Yönieticisi'ni çalıştırarak ve sürücüler sekmesini seçerek denetleyebilirsiniz.

Bir diğer sık alınan hata türü ise 80004005'dir. Bu hatanın bir kaç türü olabilir:

```
Microsoft OLE-DB Provider for ODBC Driver error '80004005' [Microsoft][ODBC
Microsoft Access 97 Driver] The Microsoft Jet database engine cannot open the
file '(unknown)'..
```

Bu hata mesajına bir kaç farklı durum sebep olabilir:

1. Internet kullanıcısına veritabanının işletim sistemi düzeyinde yazma-okuma yetkisi verilmemiş olabilir.

2. Veritabanının bulunduğu dizinin işletim sistemi açısından dosya oluşturma ve silme yetkileri yoktur.

3. Veritabanının bulunduğu dizin bir ağ sabit diskinde ise Internet kullanıcısının bu diske erişim hakkı yoktur.

4. DSN oluşturulurken veritabanı salt okunur veya Exclusive olarak işaretlenmiş olabilir.

5. O anda dosyaya Server tarafında InterDev gibi bir Web Tasarım Programı erişiyor olabilir.

6. Sözkonusu Access dosyası, bulunduğu ağda yerel kullanıcıların hizmetine açık olabilir.

Bu sorunların çözümü için ODBC yapılandırma işleminin dikkatle yeniden tekrarı ve özellikle sabit disk izinlerinin gözden geçirilmesi gerekir. Bu hata mesajını evsahibi firmanın bilgisayarından alıyorsanız; Web Server yönetimine yeniden başvurmanız şarttır.

```
Microsoft OLE-DB Provider for ODBC Driver error '80004005' [Microsoft][ODBC
Microsoft Access 97 Driver] Couldn't use the file '(unknown)'; the file already
in use
```

Bu hata mesajının tek sebebi veritabanı dosyasının birden fazla kullanıcı tarafından kullanılmasını önleyen kilit deyimlerinin kullanılmış olmasıdır. Çözümü, dosya ile ilgili .Recordset deyiminin kilidi önleyecek şekilde yazılmasından ibarettir.

```
Microsoft OLE-DB Provider for ODBC Driver error '80004005' [Microsoft][ODBC
Microsoft Access 97 Driver] Data source name not found and no default driver
specified
```

Veri tabanı ile veri-yolu bağlantısı kuracak Connection komutu, Global.asa dosyasından alınmak isteniyorsa ve IIS Global.asa dosyasını çalıştıramıyorsa bu hata mesajıyla karşılaşırsınız. Global.asa dosyasında şu kodun bulunup bulunmadığını kontrol edin:

```
<%="'auth_user' ise & request.servervariables("auth_user")%>
<p>
```

```
<%="'auth_type' is & request.servervariables("auth_type")%>
<p>
<%="connection string is & session("baglanti_deyimi"%>
<p>
```

Burada "baglanti_deyimi ifadesi yerinde sizin Connection deyiminiz yer almalıdır. Global.asa'da bu ifadelerin bulunmasına rağmen yine de çalışmıyorsa, Web Server programına, global.asa'nın içinde bulunduğu kök dizin için çalıştır (execute) izni verilmemiş olabilir. Bu kişisel Web Server'da veya yerel IIS'te oluyorsa, bütün yapacağınız şey kök dizin olan klasörü sağ tıklayarak izinlerini değiştirmektir. Bu hata evsahibi firmanın sitesinde oluyorsa, Web Sitesi yönetimi ile temasa geçmeniz gerekir.

```
Microsoft OLE-DB Provider for ODBC Driver error '80004005' [Microsoft][ODBC
Microsoft Access 97 Driver] Data source name not ??
```

Web Server'ın bulunduğu bilgisayarda MDAC (Microsoft Data Access Component) dosyaları ya bozulmuş ya da yanlış kurup kaldırma yüzünden bazı bileşenleri arasında sürüm farkı doğmuş demektir. MDAC'ın güncellenmesi çözüm sağlayabilir.

```
Microsoft OLE-DB Provider for ODBC Driver error '80004005' [Microsoft][ODBC
Microsoft Access ODBC driver Driver] General error Unable to open registry
key...
```

Registry Editor ile burada adı verilen kayıt anahtarı (Key) için verilen izinlerikontrol etmelisiniz. Bunun için Regedt32.exe programını kullanabilirsiniz.

```
Microsoft OLE-DB Provider for ODBC Driver error '80004005' [Microsoft][ODBC SQL
Server Driver] [dbnmpntw] ConnectionOpen (CreateFile()).
```

Aynı bilgisayarda bile olsa SQL Server izinleri yeterli değilse, ODBC sürücüleri veritabaına ulaşamazlar. Sorunu çözmek için SQL Server yönetimiyle görüşmek ve izinleri değiştirmek gerekir.

Bu ve diğer 80004005 hata mesajları için Microsoft'un sitesinde Q189408, Q174943, Q173742, ve Q175671 numaralı bilgi notlarına (Knowledgebase articles) başvurabilirsiniz.

ASP Hatası Arama

ASP teknolojisi, Internet'in kendisine göre nisbeten eski HTTP protokolü ile işbirliği yapmak zorundadır ve bazen ASP sayfalarımızda oluşan hatalar, bizden (yazdığımız VBScript veya diğer Script kodlarından) değil, bu iki teknolojinin Server'daki uyumsuzluğundan kaynaklanıyor olabilir.

ASP teknolojisi üç adımda çalışır:

1. Ziyaretçi bir ASP sayfası talep eder

2. Server talep edilen belgenin bir ASP programı olduğunu belirler ve bunu ziyaretçiye göndermeden önce ASP.DLL aracılığıyla çalıştırır.

3. Ortaya çıkacak olan HTML belgesi ziyaretçiye gönderilir.

Bu bakımdan ASP hatasını ararken sorulacak birinci soru hatanın nerede olduğu olmalıdır. Hata istemci tarafında ise ASP sayfamız muhtemelen ziyaretçinin Browser'ı ile uyumlu olmayan HTML kodu üretiyor olabilir. Ayrıca hatanın hangi aşamada olduğunu yakalamamız gerekir.

Sık alacağımız bir ASP hatası, Server'ın vereceği "VBS Script Error:" şeklinde başlayan mesajlar olacaktır. Bu, gerçekten de sayfamızdaki VBScript kodlarında hata olduğunu gösterir. Yapılacak tek şey, iyi bir programcı gibi, biraz açık havada gezdikten sonra kodu baştan sonra yeniden gözden geçirmek veya en iyisi başka bir programcının yardımını istemektir. Bir süre sonra insan kendi yazdığı kodlardaki hataları göremez hale gelir!

Nesne oluşturma hataları ise "Failed to create ..." diye başlar ve hemen hemen daima ASP sayfanızdaki Server.CreateObject.. diye başlayan deyimin icra edilemediğini gösterir. Bu sizin yanlış nesneye yanlış metod kullandırmaya kalkmanızdan olabileceği gibi,

Web Server'ın size bazı bileşen dosyalarına (.dll ve .exe) erişim hakkı vermiyor olmasından kaynaklanabilir. Ücretsiz ASP desteği veren sitelerde çoğu zaman bu tür komutlar içeren Script'lere izin verilmez.

İleri ASP Konuları

ASP'ye giriş niteliğindeki bu iki kitapçığın kapsamı dışında bırakılan bazı ASP konularını burada sıralayarak, bundan sonraki adımlarınıza yardımcı olabiliriz. Server izin verdiği ve gerekli yazılımı sağladığı taktirde, ASP sayfalarınızdan elektronik posta gönderebilirsiniz. Bunun bir Form'un Server'ın CreateObject metodu ile Server'ın SMTP protokülünü kullanan mesaj gönderme programına (IIS'te "CDONTS.Mail" nesnesine) ulaşması ve bu nesneye alıcı ve gönderenin adresleri ile konu ve mesaj bölümlerini iletmesi gerekir. Bunun için gerekli program (veya nesne) adını ve ASP sayfalarınızdan mektup gönderme izniniz olup olmadığını Web Server yönetiminden öğrenmeniz gerekir.

Bir diğer ileri ASP tekniği ise COM bileşenlerini kullanarak, ASP sayfalarınızı VBScript veya JavaScript yerine daha hızlı ve daha güvenli binary (program) dosyaları ile birleştirmek olabilir. C++, Visual Basic, Borland Delphi gibi bir dille yazılmış ve derlenmiş olan COM bileşenleri, kendilerine ASP sayfası tarafından gönderilen çağrı üzerine harekete geçerler ve büyük bir ihtimalle Server dışında yapmaları gereken işi yaparak sonucunu ASP sayfasına ulaştırılmak üzere Server'a bildirirler. Derlenmiş (program haline getirilmiş) oldukları için COM bileşenleri daha hızlı çalışırlar ve Server'ın kaynaklarını kullanmadıkları için de Web iletişimini yavaşlatmazlar. Bu tür bileşenlere çoğu zaman ADODB nesnesinin .Command metodu ile erişiriz. Microsoft ve diğer firmalar, ASP sayfalarımızla birlikte kullanılmak üzere COM programları üretmeye ve pazarlamaya başlamış bulunuyorlar. Bu tür bileşenleri gerçek Internet Server ortamında kullanabilmek için,

programın Server'ın bulunduğu bilgisayarda Registry kaydının yapılması gerekir. Bunu Web Server yönetimi ile görüşmeniz şarttır.

W/O: 226074

W/O: 145821

W: 167316