ASP Güncesi
Mehmet Günce Akkoyun
25.07.1997 Yeniden Doğuş

Diyagramlar

Resim ve Ekran Görüntüleri

Tablolar

Örnekler

1. ASP'yi Anlama

1.1.İnternet protokollerine bakış ve HTTP

İlerideki bölümler, sunucu taraflı programlama dillerinden ASP'yi (Active Server Pages) ayrıntılı olarak anlatmak üzere hazırlanmıştır. Fakat burada şu soru olabilir : Bir sunucu taraflı programlama dili tam olarak nedir ?

Sunucu taraflı programlamanın ne demek olduğunu anlamak için isterseniz ilk önce client taraflı (istemci taraflı) programlama dillerinden HTML dilinin çalışma prensibine bakalım.

HTTP (HyperText Transfer Protokol), web üstünde bilgi dağıtımı için kullanılan temel protokoldür. HTTP, dosyaların kolaylıkla transfer edilebilmesi için oldukça etkin ve hızlıdır. HTTP diğer web teknolojileri ile birlikte gelişmektedir. HTTP'nin temel özellikleri HTTP 0.9 da bulunmaktadır. HTTP'nin 0.9 sürümünde birçok eksikler bulunuyordu. Bunlardan başlıcaları içerik tiplemesine (content typing) izin vermemesi ile istek ve cevapların sağlanmasında meta-bilgilerin kullanılması için koşullar olmamasıdır.

HTTP/0.9 un eksikliklerini gidermek için HTTP'nin şu anda kullandığımız sürümü olan HTTP/1.0 geliştirilmiştir. Bu sayede Content-Type (içerik tipi) alanı bulunan başlıklara ve diğer tipte meta bilgilerine izin verilmiştir. Aktarılan verinin tipi Content-Type alanında tanımlanır. Ayrıca dil, verinin kodlanma tipi ve durum bilgisi gibi veri hakkında başka bilgiler de sağlayabilirsiniz.

Çoğu web kullanıcısı ve yayımcısının HTTP'de istediği özelliklerden biri de güvenliktir. Web yayımcıları ve kullanıcıları güvenli olarak işlemleri (transaction) iletebilmeyi istemektedir. Elektronik ticaretin yaygın olarak kullanımını sağlamak için güvenlikle ilgili anahtar nokta, işlemlerin güvenliğini sağlayabilme ve kodlayabilmektir. Şu anda HTTP'nin güvenlik özellikli sürümleri için çeşitli taslaklar bulunmaktadır. Bu özelliklerden birisi kabul gördüğünde HTTP kullanan güvenlikli işlemler bir hayal olmaktan çıkacaktır.

1.1.1. Bağlantısız protokoller ve bağlantılı protokoller.

HTTP etkin bir protokoldür çünkü hızlı, rahat ve ayrıca yeteneklidir. Bu hız, yeteneklilik ve kuvvetliliği sağlayabilmek için HTTP, bağlantısız (connectionless) ve durumu olmayan (stateless) bir protokol olarak tanımlanmıştır.

HTTP bağlantısız bir protokoldür. Bağlantısız protokoller, bağlantı temelli protokollerden isteklere verilen cevap karşılıklar noktasında ayrılırlar. Bağlantısız bir protokolde istemci sunucuya bağlanır, bir istekte bulunur, cevap alır ve ardından diğer isteklere hizmet vermek için bağlantıyı keser.

Bağlantı temelli bir protokol örneği FTP'dir. Bir FTP sunucusuna bağlanıldığında dosya aktarımı bittikten sonra bile bağlantı devam eder. Bu bağlantının korunabilmesi için sistem kaynakları gerekir. Yani çok sayıda açık bağlantı tutmak sunucunun kolaylıkla çökmesine sebep olabilir. Sonuç olarak FTP sunucularının çoğu belirli bir anda en fazla 250 açık bağlantıya izin verecek şekilde ayarlanmaktadır. Bunun anlamı FTP sunucusuna aynı anda en fazla 250 kullanıcı bağlanarak işlem yapabileceğidir (bu özellik çeşitli FTP sunucularında değişiklik göstermektedir). Bu sorunla beraber doğru sonlandırılmayan bağlantılarda problemler yaratabilmektedir. Bu tip protokollerde işlemler kontrolden çıkarak sunucunun çökmesi gibi istenmeyen sonuçlar görülmektedir.

Tersine HTTP bağlantısız bir protokoldür. İstemciler sunucuya bağlanır bağlandıklarında istekte bulunur, cevabını alır ve ardında bağlantıyı keser. Bağlantı devamlı korunmadığı için işlemler tamamlanırken hiçbir sistem kaynağı meşgul edilmez. Sonuç olarak HTTP sunucuları sadece aktif bağlantılarla sınırlıdır ve az bir sistem yükü ile binlerce işleme olanak tanımaktadır. Bağlantısız protokollerin dezavantajı ise aynı istemci bir daha bilgi isteğinide bulunduğunda bağlantının yeniden kurulması gereğidir. İstemci browserlar bu yükü en aza indirebilmek için alınan verileri saklayarak yine aynı veri istendiğinde kullanan sistemler geliştirmişlerdir.

1.1.2. Durumu olmayan protokoller ve durumu olan protokoller.

HTTP durumu olmayan (stateless) bir protokoldür. Durumu olamayan protokoller durumlu protokollerden isteklerle ilgili bilgilerin tutuluş şeklinde farklılık gösterirler. Durumu olmayan protokollerde işlem işlendikten sonra işlem hakkında bilgi saklanmaz. Durumlu protokollerde ise işlem işlendikten sonra durum bilgisi saklanır.

Durumlu protokol kullanan sunucular işlemlerle ilgili bağlantının durumu, çalışmakta olan işler, bu işlerin durumu vs. gibi bilgileri tutar. Genel olarak bu durum bilgisi bellekte kalır ve sistem kaynaklarını kullanır. İstemci durumlu bir protokol kullanan sunucu ile bağlantısını kestiğinde bu durum bilgisi silinmeli ve oturum sonlanmalıdır.

Durumu olamaya protokoller küçük hacimlidir. Bu protokolleri kullanan sunucular tamamlanmış işlemler ve işler hakkında hiçbir bilgi tutmazlar. Bir istemci durumu olmayan protokol kullanan sunucu ile olan bağlantısını kestiğinde hiçbir verinin silinmesine ya da oturumun sonlandırılmasına gerek kalmaz. Durum bilgisinin kaydının tutulmaması sayesinde sunucu üzerine daha az yük biner ve sunucular işlemleri hızlı bir şekilde işleyebilirler. Web yayımcıları için bu protokolün dezavantajı ise eğer web dokümanları için durum bilgisinin tutulmasını istiyorsanız bunun doküman başlığında meta-bilgi şeklinde belirtilmesinin gerektiğidir.

1.2.HTML nedir ?

Tim Berners-Lee webi tasarlarken herkesin webde yayımcılık yapmasına olanak sağlayacak kadar genel ve kullanımı kolay bir arabirimi olacak şekilde tasarladı. Cern'deki çalışanlarla beraber HTML dilini geliştirdiler. HTML dili SGML'nin (Standart Genelleştirilmiş Anlamlandırma Dili) bir alt kümesi olarak tasarlanmıştır. HTML için SGML'nin temel alınması web için geliştirilen dilin platformlar arası bir çözüm olarak kendini kanıtlamış sağlam bir standartta kök salmasına sebep olmuştur.

HTML dili oluşturulurken SGML'den sadece gerekli olan alanlar alınmıştır. Bu şekilde HTML'in karışıklığı ve dokümanların network üzerinden transferi için harcanan kaynakların miktarı bir hayli azalmıştır. HTML için SGML'nin baz alınmasının bir diğer avantajı ise doküman tanım tiplerinin (DTD) HTML standartlarını genişletmek için kolaylık sağlamasıdır. Sonuçta HTML'I geliştirenlerin amacı zaman içinde geliştirilebilen basit bir dil yazmaktı.

HTML dilinin işletilme prensibi bir hayli basittir; Kullanıcı gözatıcısından (web browser) bir url girdiği zaman (URL hakkında daha fazla bilgi için

EK-1 : URL'ler Hakkında Ayrıntılı Bilgi) e bakınız) istek bu url'ye sahip sunucuya gider . Örneğin "*http://www.akkoyun.net/test.htm"* şeklindeki bir giriş olduğu zaman gözatıcı, serverdan "test.htm" dosyasını ister. Server ise bu dosyayı kendi iç dosya yapısında arayarak bulur ve dosyayı aynen gözatıcıya yollar.

Eğer istek sonrasında dosya bulunamaz ise server standart hata kodlarından oluşan kod numaralarından sayfa bulunamadı ("page can not found") hatası olan "404" kodlu hatayı gözatıcıya HTTP başlık bilgisi olarak geri yollar. Bu hata kodunu da gözatıcı kendine göre yorumlayarak gözatıcıda gösterir.

Dosya sistemi tarafından bulunan doküman gözatıcıya yollanır. Gözatıcıya gelen HTML belgesi gözatıcı tarafından yorumlanarak işletilir ve ekranda gösterilir.

Bu konuda üzerine basarak belirtmekte fayda gördüğüm nokta HTML'in sunucu tarafında işletilmeyip tamamen istemci tarafında (client) işletiliyor olmasıdır. Böylece sunucu üzerinde hiçbir işlem yapılmadığı için sunucuya yük binmeyecektir.

1.3.ASP'nin Doğuşu

HTML in gelişmesi ile birlikte kullanıcılara web sayfalarına bilgi girebilmelerine olanak tanında (<input> elementi yardımı ile). Bu şekilde bir çok uygulama geliştirildi çünkü artık kullanıcı da sunucuya bilgi gönderebiliyordu. Fakat çoğu uygulamada bu kullanıcıdan gelen bilgilerin anında işlenmesi ve yeniden bir text bazlı HTML dokümanı hanine getirilmesi gerekiyordu. Bu ise hiç hızlı bir yöntem değildi.

Bu zorluğu aşmak isteyen geliştiriciler kolları sıvayarak CGI (Common Gateway Interface) arabirimini geliştirmeye başladılar. Bu arabirimi standart haline getirdiler ve tamamen "C" dili üzerine kurdular. "Cgi-bin" dizini de bu şekilde doğmuştur ("bin" terimi derlenmiş "C" kodu olmasından dolayı "binary code" anlamında eklenmiştir). İlk uygulamalar derlenmiş ufak programcıklar halinde olmuştu. Fakat bu haliyle bile kullanışlı değildi çünkü dosya içinde yapılacak en ufak değişiklikte bile yeniden derlenmesi gerekmekteydi. Buda CGI'ın kullanımını olumsu yönde etkiliyordu.

Bu kısıtlamaları kendine sınır olarak görmeyen geliştiriciler yeni bir script dili geliştirdiler. Bu dil "Practical Extraction and Reporting Language" yani PERL adını aldı. Bu dil sunucu ile iletişim halinde olan ilk dildi yani "C veya C++" dilleri ile yazılan scriptin her seferinde derlenmesi derdi ortandan kaldırılmış oldu.

Perl hala popüler bir dil olarak çoğu uygulamada özellikle de Unix ve Linux tabanlı sistemlerde kullanılmaktadır.

1.3.1. Server-Tabanlı Script Teknolojileri

Şimdiye kadar anlattığım CGI dilleri web sunucusu üzerine bir yama yapmadan yada ekstra bir program yüklemeden çalışmamaktadır. Bu programlar kullanıcıdan gelen isteği algılar ve isteğe göre dosyayı okur daha sonra onu sunucu içinde işler ve bir çıkış dosyası oluşturarak kullanıcıya sunarlar.

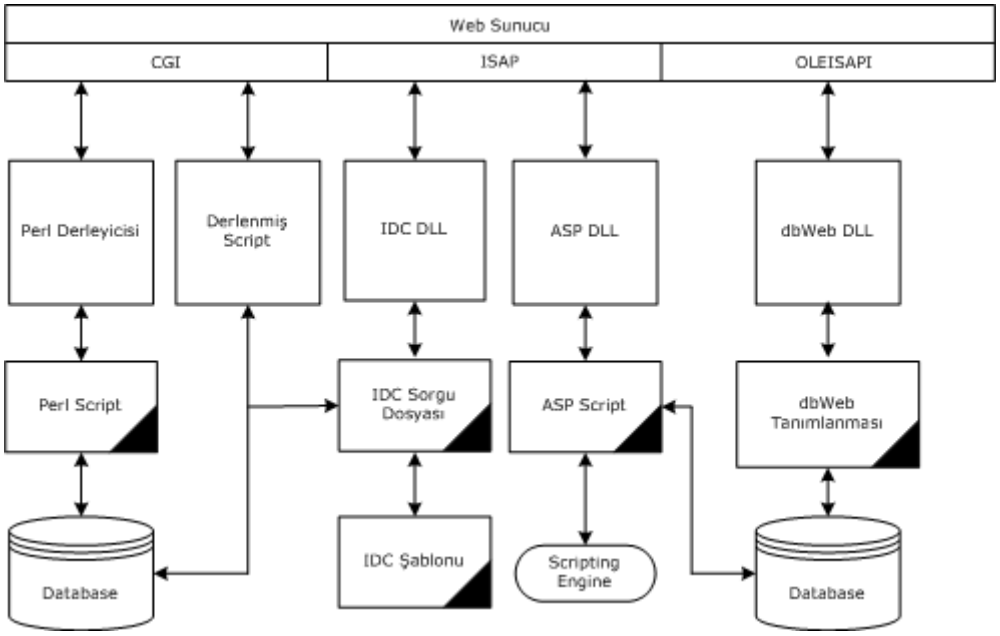
Perl ilk popüler sunucu-tabanlı uygulama geliştirme dili olarak literatüre geçmiştir. Fakat bu dil çok fazla gelişim geçirmiştir. Özellikle de Unix ve Linux tabanlı sunucularda yeni nesil programlama dili olan PHP (Personal Home Page) ye yerini bırakmıştır.

Microsoft firması web sunucu sektöründeki en önemli atılımını "Windows NT 3.51" ve bu işletim sistemine entegre halde olan "Internet Information Server 1.0" (IIS) sayesinde yapmıştır. Bu yazılım geçmişe dönük olarak CGI desteklemekle (her şekilde C ve C++ dili ile geliştirilmiş uygulamaları) birlikte yeni bir arabirim içeriyordu.

Bu arabirime "Internet Server Aplication Programing Interface" yani ISAPI adı verilmiştir. Bu arabirim sayesinde web sunucu perl dilinin tüm esnekliği standart hale getirilmiş oldu. Bu atılımla birlikte tüm yazılım geliştiriciler ISAPI ile uyumlu olan yazılımlar geliştirmeye başladılar.

Microsoft ISAPI ile beraber yeni teknoloji olan ASP'yi duyurdu. ASP teknolojisi IIS ile ISAPI sayesinde bağlanmış oldu. ASP den önce en çok "Internet Database Connector" (IDC) kullanılmaktaydı.

Aşağıdaki diyagramda (Diyagram 1 : Microsoft Sunucu Yapısı.) Microsoft server yapısı ayrıntılı olarak anlatılmıştır.



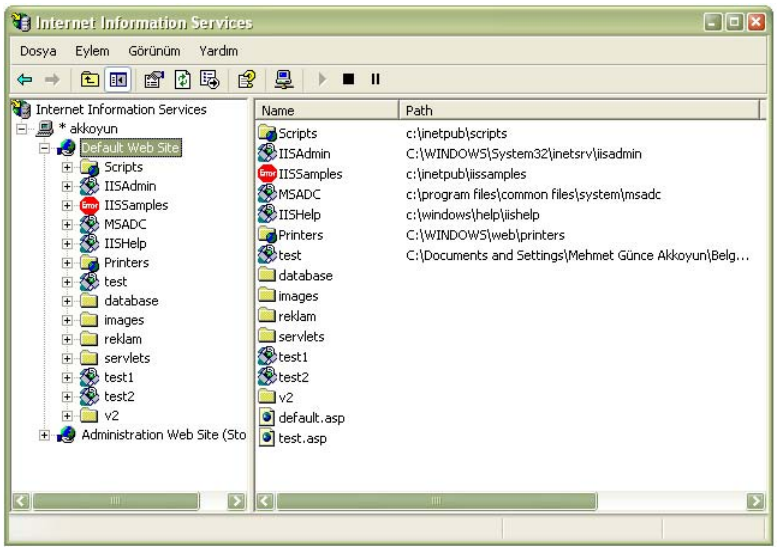
Diyagram 1 : Microsoft Sunucu Yapısı.

1.3.2. ASP ile IIS ilişkisi

ASP sadece kendisi için yazılmış olan DLL i kullanır (asp.dll). Bu dosya standart olarak web sunucu da yer almaktadır (sadece IIS 1.0 sonrası) (Winnt\System32\ inetsrv dizininde yer almaktadır). Bu DLL sadece ASP dosyalarını (genellikle .ASP uzantılıdırlar) okuyup içerisindeki script komutlarını işlemek ve sonuçlarını HTML ve metin içeriği ile birlikte Web gözatıcısına yollamak görevini üstlenir.

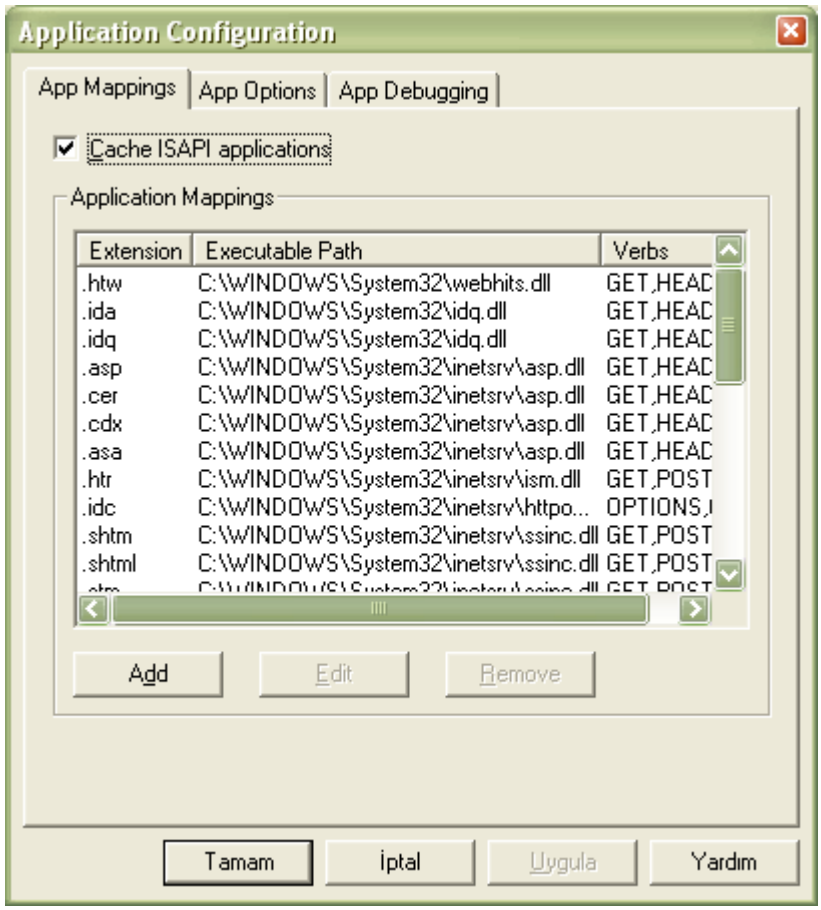
1.3.3. IIS Uygulama Yapıları

IIS içerisindeki işlemleri daha iyi anlayabilmek için uygulama yapılarının Windows içinde nasıl çalıştığını anlamalıyız. Web sunucudaki (IIS) her web sitesinin sunucu üzerinde yer alan bir kök dizini vardır. Varsayılan (Default) web sitesi otomatik olarak "c:\inetpub\wwwroot" dizinini kendine kök dizini atar (değiştirilebilir). Her yeni açılacak web sitesi için bir kök dizini belirlenmesi zorunludur. Sunucu üzerindeki web sitelerini görmek için IIS yönetim arabirimi olan "Internet Service Manager" programı kullanılır.



Resim 1 : Microsoft Internet Service Manager.

IIS üzerinden default web sitesine sağ tıklayıp özellikler seçeneğini seçtiğimiz zaman, açılacak olan "default web site özellikleri" ekranından "home directory" tabına tıklanır. Gelecek olan sekmeden "configration" butonuna basarak uzantılar ile ilişkilendirilen arabirimler görülebilir (ISAPI yapısı "Diyagram 1 : Microsoft Sunucu Yapısı." de görüldüğü gibidir).



Resim 2 : ISAPI Yapısı.

Resimden de (Resim 2 : ISAPI Yapısı.) görülebileceği gibi asp uzantılı dosyalar asp.dll dosyası ile derlenmektedir. HTM ve HTML uzantılı html sayfaları ve XML uzantılı xml sayfaları direkt olarak diskten okunup (web sunucu tarafından) istemciye gönderilmektedir ama asp uzantılı dosyalar ISAPI yardımı ile asp.dll tarafından okunup derlenip sonuç çıktıları istemciye göderilmektedir.

Not : İstemci sunucudan url yardımı ile bir dosya isteğinde bulunduğu zaman istek paketi şu şekilde olacaktır:

```
11/11/02 02:49:16 Sent  GET /deneme/ornek.asp HTTP/1.1
Accept: aplication/msword, application/vnd.ms-excel, application/vnd.ms-powerpoint, image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-comet, */*
Accept-Language: en-us
Encoding: gzip, deflate
Referer: http://www.akkoyun.net/kitap.asp
Cookie: VisitCount=2&LastDate=6%2F4%2F99
User-Agent: Mozilla/4.0 (compatible; MSIR 6.0; Windows XP)
Host: 212.98.198.111
Connection: Keep-Alive
```

Ve bu isteği takiben sunucu istemciye dosyayı gönderirken şu başlık kısmı ile beraber gönderecektir :

```
11/11/02 02:49:16 Recived HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Connection: Keep-Alive
Date: Thu, 11 Nov 2002 10:22:21 GMT
Content-Type: text/html
Accept-Ranges: bytes
Content-Lenght: 2946
Last-Modified: Thu, 11 Nov 2002 10:22:21 GMT
Cookie: VisitCount=2&LastDate=6%2F4%2F99
<HTML>
--Sayfanın geri kalanı
```

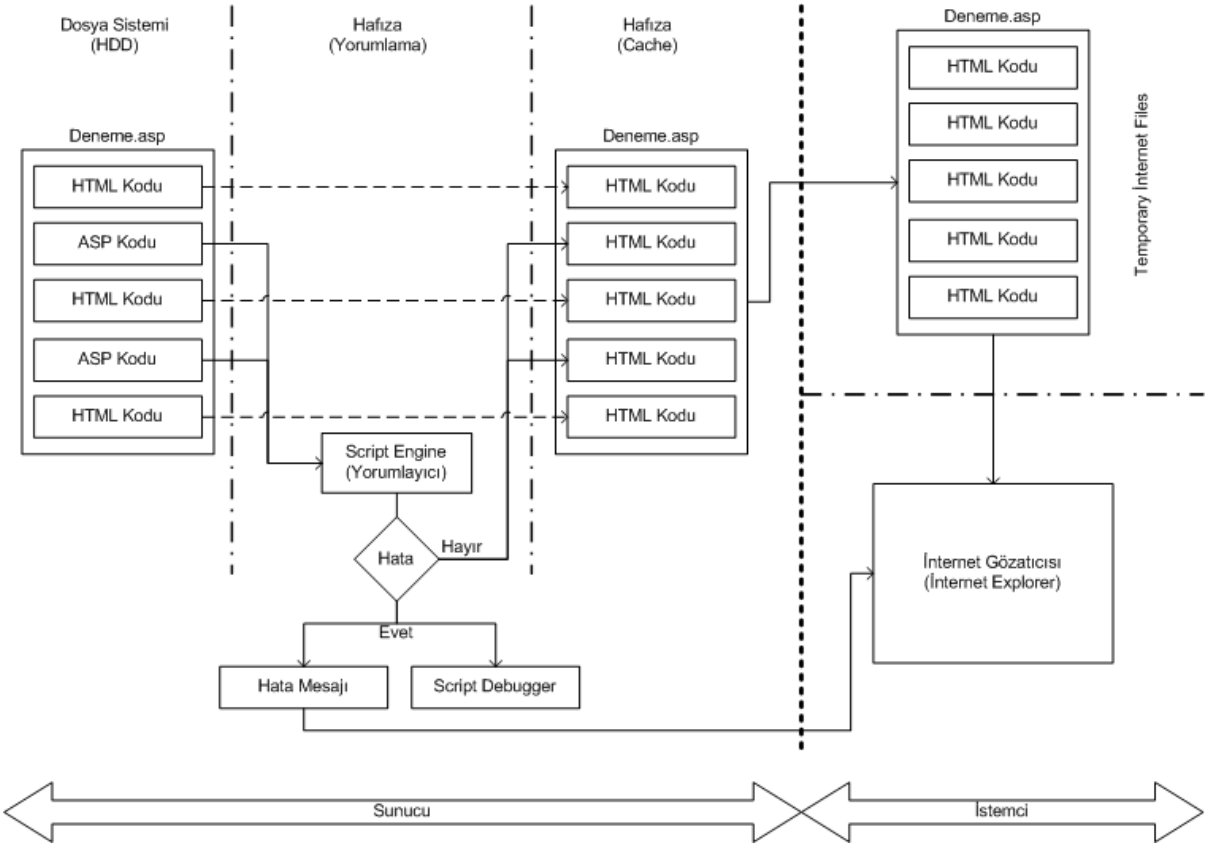
Bu bilgileri dikkatli olarak incelemenizi istiyorum çünkü ilerideki konularımızda bu sunucu ve istemci değişkenlerini okutmak ve değiştirmek üzerine konular göreceğiz.

1.3.4. ASP dosyalarının işletilmesi

Asp uzantılı dosyaların "asp.dll" yardımı ile derlendiğini bilmekteyiz peki ya bu derlenme nasıl olmaktadır.

Birinci basamak olarak asp dosyası içerisinde server taraflı kod olup olmadığı denetlenir. Eğer dosya içerisinde sunucu taraflı işletilecek bir kod bulunmaz ise IIS tarafından direkt istemciye gönderilir. Bu "Windows 2000" de yeni bir özellik olarak eklenmiştir. Bu sayede ".asp" uzantılı dosyaları kullanmamıza olanak tanır (içerisinde sunucu taraflı çalışacak kod bulunmayan dosyalara bile ".asp" uzantısı verilebilmektedir).

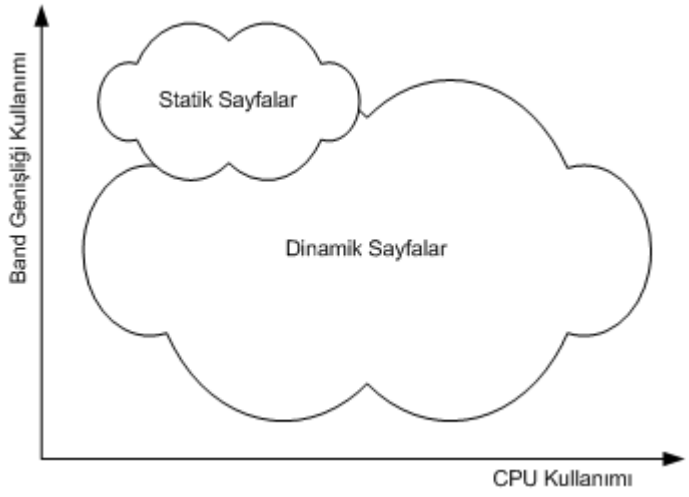
Eğer DLL, içerisinde server taraflı çalışacak bir kod olan dosya ile karşılaşırsa; satır satır bu dosyayı işleterek script blokları içerisindeki komutları işletir ve çıktısını yine aynı yere gelecek şekilde bir çıkış dosyasına (hafızada bulunan bir tampon bölgeye) kaydeder. İçerisinde script komutu olmayan kısımlarda aynen değiştirilmeden tampona yazılır. Tüm bu işlemler tamamlandıktan sonra eğer herhangi bir hata oluşmamış ise bu dosya (tamponda tutulan) istemci bilgisayarına gönderilerek işlem tamamlanır. Bu gönderme işlemi sonucunda tampon bölge temizlenir.



Diyagram 2 : ASP Dosyalarının İşletilmesi ve İstemciye Gönderilmesi.

Bilgi : IIS in ilk versiyonlarında ASP komutlarımızı işletecek olan ASP.DLL , asp dosyalarını satır satır okuyup satır satır işleyip yine satır satır çıktı alırdı. Buda performansta ciddi bir düşme meydana getirirdi.

Dinamik sunucu sayfalarının oluşturulması sırasında kullanılacak olan sistem kaynakları aşağıdaki grafikte resmedilmiştir.



Resim 3 : Sistem Kaynakları Kullanımı.

Asp script motoru (asp.dll) dosya içerisinde sunucu taraflı çalışacak kod olup olmadığını iki şekilde anlar.

Bilgi : Bazılarınızın bildiği gibi ASP'de FSO (File System Object) kullanarak yazılmış ve sunucudaki diğer dosyalara erişebilen bir script piyasada dolaşmaktadır. Çoğu programcının düşüncesinin aksine bu bir açık

(güvenlik deliği) değildir. Bu script sunucu yöneticilerinin yaptığı yanlış veya yetersiz yetkilendirme sonucu kendi yetkisi dışındaki dosyalara da erişebilmektedir. Bu yanlış yetkilendirmeyi isterseniz adım adım düzeltelim.

Öncelikle web sunucunuzun "root" klasörüne (tüm web sitesi dosyalarının bulunduğu sanal olarak en üst dizin anlamında), bir klasör oluşturunuz. Örneğin bu klasörün ismi "guvenli" olsun. Daha sonra "Computer Management" dan "Local Users and Groups" a tıklayınız. Gelen menüden "Users" seçeneğini seçtiğiniz zaman sağdaki pencerede bilgisayarda kayıtlı tüm kullanıcıların listesi gelecektir. Listenin boş bir yerine sağ tıklayarak "new user" seçeneği ile yeni bir kullanıcı oluşturalım. Daha sonra tekrar "Computer Management" penceresinden "Services and Application" kısmına buradan da "IIS" sekmesine tıklayalım. Buradan "Default Web Site" sekmesini işaretleyelim. Sağ tarafta web sayfanız içerisindeki tüm klasörler gözükecektir (tabi ki bizim oluşturduğumuz "guvenli" klasörü de gözükecek) buradan oluşturduğumuz "guvenli" klasörüne sağ tıklayarak "Properties" seçeneğini seçiyoruz. Açılan penceredeki "Directory > Application Settings" bölümündeki "create" düğmesine tıklayınız. Bu seçili olan klasör için ayrı bir uygulama yaratacaktır. Artık bu klasöre yetkili bir kullanıcı seçebiliriz ayrıca bu kullanıcının "IUSER" (İnternet user) olması da gerekmiyor. "Directory Security>Anonymous access and auth.. control" bölümündeki "Edit" düğmesine tıklayın. Buradan açılacak pencereden "Anonymous access" bölümündeki "Edit" düğmesine tıklayın. "Browse" düğmesine tıklayarak önceden yarattığınız user'ı seçiniz. "Allow IIS to control password" seçeneğini aktif hale getirip her şeye OK deyip çıkınız. Şimdi "c:\inetpub\....." yada her neyse. Yarattığınız "guvenli" klasörünü bulunuz. Klasöre sağ tıklayarak "properties" seçeneğini seçiniz. "Security" yi seçip buradaki tüm userları siliniz. Sadece Administrator ve yarattığınız user'ı ekleyin. Eğer bu klasörde dosya yaratma izni vereceksenız kendi user'ınıza "modify,write veya full control" verebilirsiniz. Ok deyip buradan da çıkınız.

Evet artık "http://127.0.0.1/guvenli" klasörünün içinden FSO ile değil c:\ ye bir üst klasöre bile çıkamazsınız. Bir üst klasördeki "txt" dosyalarını bile okuyamazsınız.

1.3.4.1. <% ve %> kullanarak

En çok kullanılan yöntem olup script bloğu başlangıcına "<%" ve script bloğu bitişine "%>" yazılarak arasında kalan kısma sunucu taraflı kod yazılır (Örnek 1 : <% %> Kullanmak.).

```
<HTML>
  <Body>
  Bu bir HTML metinidir
    <%
     Rem burasi script bloğudur
    %>
  </Body>
</HTML>
```

Örnek 1 : <% %> Kullanmak.

1.3.4.2. <script> elementini kullanarak

Nadir olarak kullanılan bu yöntem ile sunucu taraflı kodun yer aldığı script bloğunu <script> elementi ile açıp </script> elementi ile bitirilmesi baz alınmıştır. Bu şekilde ki kullanımda element içerisine yazılacak olan "Runat" özelliği sayesinde istemci veya sunucu taraflı çalışma özelliği eklenmiştir (Örnek 2 : <script> Elementi Kullanmak.).

```
<HTML>
  <Body>
  Bu bir HTML metinidir
    <script runat="server">
     Rem burasi script bloğudur
    </script>
  </Body>
</HTML>
```

Örnek 2 : <script> Elementi Kullanmak.

Bunun yanında "Script" elementi kullanılarak sunucu üzerinde yer alan bir dosya script bloğu içerisine dahil edilebilir. Bu şekilde tüm sayfalarda kullanılan ortak kodlar bir defaya mahsus olmak üzere yazılır ve bu kodlar gereken yerlere dahil ettirilir. Bunu şu şekilde yapabilmekteyiz (Örnek 3 : Harici Kod Kullanımı.).

```
<HTML>
  <Body>
    <script runat="server" src="/script.inc"></script>
  </Body>
</HTML>
```

Örnek 3 : Harici Kod Kullanımı.

Bilgi : Server Side Includes (SSI[i]) kullanarak harici dosyaları da asp dosyamız içerisine dahil edebiliriz. Böylece script blokları içerisinde yer alan kodları harici dosyalarda saklayabilmekteyiz. Bu konuyu ilerideki konularımızda ayrıntılı olarak ele alacağız.

Yukarıda anlatılan yöntemle yapılan bir uygulamada içeriği dahil edilen "script.inc" dosyasının içerisinde mutlaka geçerli bir script olmalıdır. Bu script metin (text) yada HTML olmamalıdır. Script elementinin içerisine (açılış ve kapanış tagları arasına) kesinlikle başka bir kod gelmemesi gerekmektedir.

1.3.5. Özel ASP Komutları

1.3.5.1. Script Dilini Belirlemek

IIS standart olarak iki script motoru (scripting engine) ile beraber gelir. Bunlar "VBScript" ve "Jscript" dir. Bu motorlar birlikte bulunurlar. Bunlar dışında TCL ve PerlScript gibi diğer script motorları da mevcuttur fakat bunlar IIS ile beraber gelmez sadece sonradan eklenir.

ASP ye biz hangi script motorunu kullanması gerektiğini söyleyebiliriz. Bu genelde standart olarak IIS de ayarlanmış haldedir (bu işlem için ilk kurulumda tanımlanmış bazı varsayılan değerler mevcuttur). Bu tanımı yapmanın en kolay yolu ASP sayfamızın ilk satırında özel içerik tanımı yapmaktır (Örnek 4 : Language Kullanımı.).

```
<%@Language = "dil"%>
```

Örnek 4 : Language Kullanımı.

Bu şekilde sayfanın VBScript mi yoksa Jscript mi kullanacağı tanımlanmış olur. VBScript için

```
<%@Language = "VBScript"%>
```

Örnek 5 : Varsayılan Dili VbScript Yapmak.

Yazabiliriz (Örnek 5 : Varsayılan Dili VbScript Yapmak.). Aynı Mantıkta Jscript için

```
<%@Language = "JScript"%>
```

Örnek 6 : Varsayılan Dili JScript Yapmak.

Şeklinde bir kod uygun olacaktır (Örnek 6 : Varsayılan Dili JScript Yapmak.).

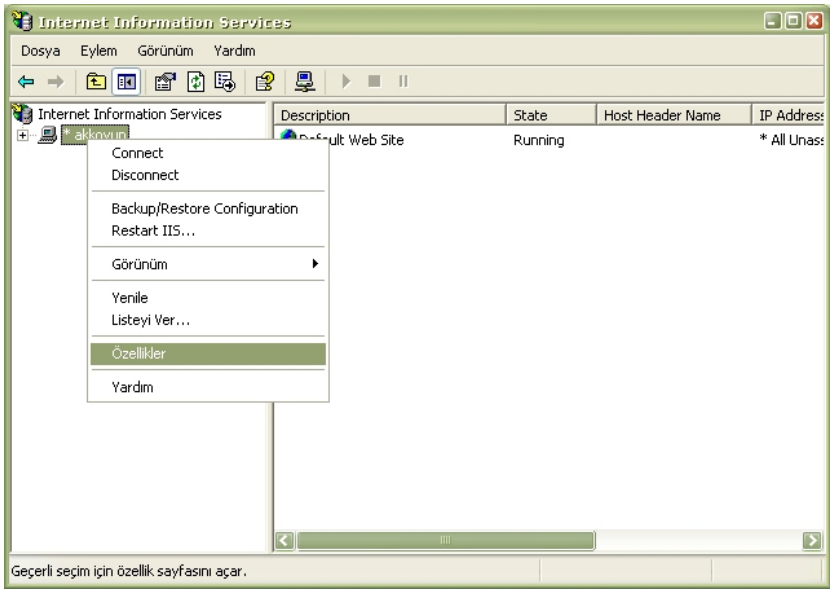
Bu tanımlama yapılsın yada yapılmasın script bloğumuzu <script> elementi ile tanımlamışsak istediğimiz dili orada da tanımlayabiliriz (Örnek 7 : <script> Elementi İçerisinde Dil Belirlemek.).

```
<HTML>
  <Body>
  Bu bir HTML metinidir

  <script Runat = ”server” Language = “VBScript”>
   Rem burasi script bloğudur ve dil olarak VBScript Kullanılmıştır
  </script>

  <script Runat = ”server” Language = “JScript”>
   Rem burasi script bloğudur ve dil olarak JScript Kullanılmıştır
  </script>

  </Body>
</HTML>
```

Örnek 7 : <script> Elementi İçerisinde Dil Belirlemek.

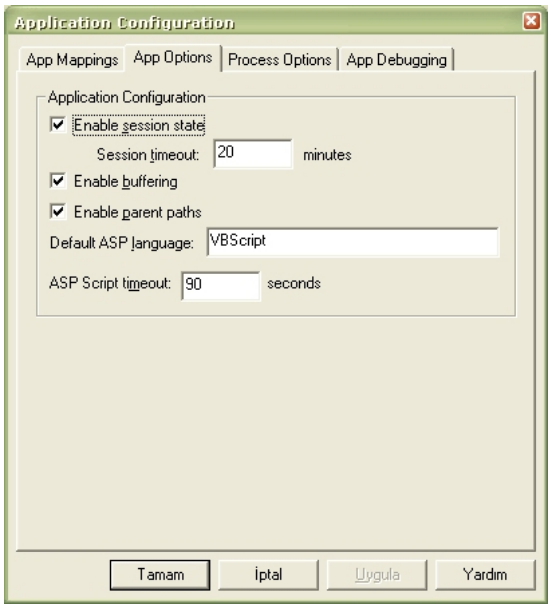
1.3.5.1.1.Varsayılan Script Dili

Script elementi içerisinde dil belirtilmemesi durumunda veya <% %> tagları arasında aksi bir ayar yoksa sunucunun varsayılan script dili geçerli olur. Bu varsayılan dil değerini IIS ayarları yardımıyla değiştirebiliriz bunun için, “Internet Service Manager” içerisinden web sitesi adı üzerine sağ tıklayınız, gelen menüden “Özellikler” (Properties) seçeneğine giriniz (Resim 4 : Sunucu Özellikleri.).



Resim 4 : Sunucu Özellikleri.

Özellikler öğesi seçildiği zaman gelecek olan pencerede yer alan “WWW Servis” seçeneğinin düzenle butonuna tıklayarak sistemin ana www servisi ayarlarına ulaşılır. “Home Directory” sekmesine tıklayınız. Gelen pencerede yer alan “Configration” butonuna tıklayınız. Gelen penceredeki “App Options” sekmesine yer alan “Default ASP Language” seçeneğini isteğimize göre düzenleyebiliriz. Bu alana ya “VbScript” yada “Jscript” değeri verilebilir. Aşağıda bu ayar ekranına ait ekran görüntüsü gösterilmektedir (Resim 5 : Varsayılan ASP Ayarları.).



Resim 5 : Varsayılan ASP Ayarları.

1.3.5.2. CodePage

Okunabilir ve yazılabilir bir değişken olan codepage sayfa başında özel biçimi ile kullanıldığı zaman içinde olduğu sayfaya ait gösterim ayarları verilmesini sağlar. Örneğin Amerika için "1252" olan bu değer Türkiye için "1254" olacaktır. Bu komutun sayfaya yazılması halinde "scriptengine" e bu çalışan sayfanın gösterim ayarlarının Amerika'ya göre ayarlanacağını belirtir (Örnek 8 : CodePage Kullanımı.).

```
<%@CODEPAGE = "1254"%>
```

Örnek 8 : CodePage Kullanımı.

Bu komuttan sonra sayfamızda Türkçe karakterler sorunsuz olarak gösterilebilecektir. Bu kod aynı zamanda yukarıda anlatılan script dili belirleme ile beraber kullanılabilmektedir (Örnek 9 : CodePage ve Language Beraber Kullanımı.).

```
<%@CODEPAGE = "1254" LANGUAGE="VbScript"%>
```

Örnek 9 : CodePage ve Language Beraber Kullanımı.

1.3.5.3. LCID

Okunabilir ve yazılabilir bir değişken olan LCID (Local Identifier : Bölge Tanımlayıcı) sayfa başında özel biçimi ile kullanıldığı zaman içinde olduğu sayfaya ait özel bölgesel ayarların verilmesini sağlar. Örneğin LCID değeri "2057" şeklinde ayarlandığı zaman para birimi "£" olarak ayarlanacaktır (Örnek 10 : LCID Kullanımı.). Türkiye için bu değer "1055" dir. Bu şekilde Türkiye'ye ait yerel saat ayarlanmış olacaktır.

```
<%@LCID = "2057"%>
```

Örnek 10 : LCID Kullanımı.

Bu komuttan sonra sayfamızda Türkçe bölgesel ayarlar yapılmış olacak ve tamamen Türkiye ayarları kullanılacaktır. Bu kod aynı zamanda yukarıda anlatılan script dili ve kod sayfası belirleme ile beraber kullanılabilmektedir. Aşağıda bu kullanımın örneği görülmektedir (Örnek 11 : Özel Komutların Beraber Kullanımı.).

```
<%@CODEPAGE="1254" LANGUAGE="VbScript" LCID="2057"%>
```

Örnek 11 : Özel Komutların Beraber Kullanımı.

Aşağıda dünya çapında kullanılmakta olan bölgesel ayarları gösteren bir tablo bulunmaktadır. Bu tablodaki LCID değerini kullanarak o bölgeye ait ayarları aktif hale getirebilirsiniz.

Bölgesel Açiklama	Kısaltma	Hex Değeri	Onluk Değeri (LCID)
Afrikaans	af	0x0436	1078
Albanian	sq	0x041C	1052
Arabic - U.A.E.	ar-ae	0x3801	14337
Arabic - Bahrain	ar-bh	0x3C01	15361
Arabic - Algeria	ar-dz	0x1401	5121
Arabic - Egypt	ar-eg	0x0C01	3073
Arabic - Iraq	ar-iq	0x0801	2049
Arabic - Jordan	ar-jo	0x2C01	11265
Arabic - Kuwait	ar-kw	0x3401	13313
Arabic - Lebanon	ar-lb	0x3001	12289
Arabic - Libya	ar-ly	0x1001	4097
Arabic - Morocco	Ar-ma	0x1801	6145
Arabic - Oman	Ar-om	0x2001	8193
Arabic - Qatar	ar-qa	0x4001	16385
Arabic - Saudia Arabia	ar-sa	0x0401	1025
Arabic - Syria	ar-sy	0x2801	10241
Arabic - Tunisia	ar-tn	0x1C01	7169
Arabic - Yemen	ar-ye	0x2401	9217
Basque	eu	0x042D	1069
Belarusian	be	0x0423	1059
Bulgarian	bg	0x0402	1026
Catalan	ca	0x0403	1027
Chinese	zh	0x0004	4
Chinese - PRC	zh-cn	0x0804	2052
Chinese - Hong Kong	zh-hk	0x0C04	3076
Chinese - Singapore	zh-sg	0x1004	4100
Chinese - Taiwan	zh-tw	0x0404	1028
Croatian	hr	0x041A	1050
Czech	cs	0x0405	1029
Danish	da	0x0406	1030
Dutch	nl	0x0413	1043
Dutch - Belgium	nl-be	0x0813	2067
English	en	0x0009	9
English - Australia	en-au	0x0C09	3081
English - Belize	en-bz	0x2809	10249
English - Canada	en-ca	0x1009	4105
English - Ireland	en-ie	0x1809	6153
English - Jamaica	en-jm	0x2009	8201
English - New Zealand	en-nz	0x1409	5129
English - South Africa	en-za	0x1C09	7177

English - Trinidad	en-tt	0x2C09	11273
English - United Kingdom	en-gb	0x0809	2057
English - United States	en-us	0x0409	1033
Estonian	et	0x0425	1061
Farsi	fa	0x0429	1065
Finnish	fi	0x040B	1035
Faeroese	fo	0x0438	1080
French - Standard	fr	0x040C	1036
French - Belgium	fr-be	0x080C	2060
French - Canada	fr-ca	0x0C0C	3084
French - Luxembourg	fr-lu	0x140C	5132
French - Switzerland	fr-ch	0x100C	4108
Gaelic - Scotland	gd	0x043C	1084
German - Standard	de	0x0407	1031
German - Austrian	de-at	0x0C07	3079
German - Lichtenstein	de-li	0x1407	5127
German - Luxembourg	de-lu	0x1007	4103
German - Switzerland	de-ch	0x0807	2055
Greek	el	0x0408	1032
Hebrew	he	0x040D	1037
Hindi	hi	0x0439	1081
Hungarian	hu	0x040E	1038
Icelandic	is	0x040F	1039
Indonesian	in	0x0421	1057
Italian - Standard	it	0x0410	1040
Italian - Switzerland	it-ch	0x0810	2064
Japanese	ja	0x0411	1041
Korean	ko	0x0412	1042
Latvian	lv	0x0426	1062
Lithuanian	lt	0x0427	1063
Macedonian	mk	0x042F	1071
Malay - Malaysia	ms	0x043E	1086
Maltese	mt	0x043A	1082
Norwegian - Bokmål	no	0x0414	1044
Polish	pl	0x0415	1045
Portuguese - Standard	pt	0x0816	2070
Portuguese - Brazil	pt-br	0x0416	1046
Raeto-Romance	rm	0x0417	1047
Romanian	ro	0x0418	1048
Romanian - Moldova	ro-mo	0x0818	2072
Russian	ru	0x0419	1049
Russian - Moldova	ru-mo	0x0819	2073
Serbian - Cyrillic	sr	0x0C1A	3098
Setsuana	tn	0x0432	1074
Slovenian	sl	0x0424	1060
Slovak	sk	0x041B	1051
Sorbian	sb	0x042E	1070
Spanish - Standard	es	0x040A	1034
Spanish - Argentina	es-ar	0x2C0A	11274
Spanish - Bolivia	es-bo	0x400A	16394
Spanish - Chile	es-cl	0x340A	13322
Spanish - Columbia	es-co	0x240A	9226
Spanish - Costa Rica	es-cr	0x140A	5130
Spanish-Dominican Republic	es-do	0x1C0A	7178

Spanish - Ecuador	es-ec	0x300A	12298
Spanish - Guatemala	es-gt	0x100A	4106
Spanish - Honduras	es-hn	0x480A	18442
Spanish - Mexico	es-mx	0x080A	2058
Spanish - Nicaragua	es-ni	0x4C0A	19466
Spanish - Panama	es-pa	0x180A	6154
Spanish - Peru	es-pe	0x280A	10250
Spanish - Puerto Rico	es-pr	0x500A	20490
Spanish - Paraguay	es-py	0x3C0A	15370
Spanish - El Salvador	es-sv	0x440A	17418
Spanish - Uruguay	es-uy	0x380A	14346
Spanish - Venezuela	es-ve	0x200A	8202
Sutu	sx	0x0430	1072
Swedish	sv	0x041D	1053
Swedish - Finland	sv-fi	0x081D	2077
Thai	th	0x041E	1054
Turkish	tr	0x041F	1055
Tsonga	ts	0x0431	1073
Ukranian	uk	0x0422	1058
Urdu - Pakistan	ur	0x0420	1056
Vietnamese	vi	0x042A	1066
Xhosa	xh	0x0434	1076
Yiddish	ji	0x043D	1085
Zulu	zu	0x0435	1077

Tablo 1 : LCID Değerleri.

2. ASP'nin Temelleri

2.1.Değişkenler

Bütün programlama dillerinde olduğu gibi vbscriptte de değişkenler terimi vardır. Bu terim sayesinde program içerisinde işleyeceğimiz sayısal veya metinsel değerleri kullanmaktansa bu değerlerin yerini tutan bir değişkeni kullanırız. Bu sayede program içerisinde daha esnek bir yapı kurabiliriz. Değişkenleri temsil eden adlar vardır, örneğin "isim" değişkeni (tarih, toplam… gibi) isim değerleri alır. Anlaşıldığı üzere değişkenin adı değişmiyor fakat değişken içerisinde tutulan değer değişebiliyor. (bu değişiklik programcının kontrolünde meydana geliyor). Verilen değer program akışına göre farklılaşabildiği için bu terime değişken diyoruz.

Piyasada kullanılan bütün programlama dillerinde değişkenler kullanılmadan önce tanımlanmalı ve boyutlandırılmalıdır. Fakat vbscript de bu dillerin aksine değişkenlerin tanımlanması ve boyutlandırılması zorunlu değildir. Vbscript tanımlanmamış değişkenleri hiç bir hata vermeden kabul eder. Fakat karmaşık programlarda sayfalar arası değişken problemleri yaşanabilir (daha önceden kullandığımız bir değişkene bir değer atamadan tekrar başka bir yerde kullanırsak eski değeri kalacağı için programda hatalar oluşacaktır). Bunu değişkenlerimizi kullanılmadan önce tanımlayarak engelleyebiliriz. (profesyonel bir programcı değişkenlerini kullanmadan önce mutlaka tanımlar böylece programın ilerleyen safhalarında değişken hatası yapmamış olacaktır) Bu alışkanlığı zorunluluk haline getirmek için vbscipt de "Option Explicit" komutu kullanılır. Komutu kullanmamız durumunda ASP yordamcısı her kullandığımız değişkeni kullanmadan önce tanımlanmasını zorunlu kılar, bu sayede sayfalar arası değişken karmaşası oluşmaz. Ve değişken adlarının yanlış yazılması bir nevi engellenmiş olur. Bu komutu mutlaka ASP programlarınızın ilk satırında kullanmalısınız, diğer hallerde program hata verecektir.

Asp de değişken tanıtma işlemi için "Dimension" (boyut) kelimesinin kısaltılmışı olan "DIM" komutu kullanılır. Bu komut yardımı ile, kullanılan değişkenler kod başlangıcında belirtilerek tanımlanır (iyi bir programlama için kod başında değişken tanımlaması yapmak daha iyidir. Ama sayfa içerisinde herhangi bir yerde değişken tanımlaması yapmak, bir hata meydana getirmeyecektir.).

```
<%
  Option Explicit
  Dim Ad
%>
```

Örnek 12 : Değişken Tanımlama.

Örnekte görüldüğü gibi "DIM" komutu ile "Ad" değişkenini tanımlamış olduk bu tanımlamayı yapmadan önce "option explicit" komutunu kullandık. Bu komutu kullanma amacımız Ad değişkenini programın ilerleyen aşamalarında bir daha kullandırtmamak (değişken adı olarak) ve karışıklık olmasını engellemektir (değişken değeri değiştirilebilir fakat bu isimde başka bir değişken kullanılamaz). Bu tanımlama işlemini:

```
<%
  Ad = "Günce Akkoyun"
%>
```

Örnek 13 : Değişkene Değer Atama.

Şeklinde kullansak dahi program hatasız çalışacaktır. Fakat kullanım pratikliği ve kodlama sağlamlığı bakımından "Option Explicit" kullanımı daha iyi olacaktır. "Option Explicit" komutu ile,

```
<%
  Option Explicit
  Ad = "Günce Akkoyun"
%>
```

Örnek 14 : Hatalı Değer Atama.

Şeklinde bir kullanım mutlaka hata verecektir.

```
Microsoft VBScript runtime (0x800A01F4)
Variable is undefined: 'Ad'
/asp/test.asp, line 3
```

Çıktı 1 : Tanımlanmamış Değişkene Değer Atama.

Bu hata bize değişkenimizin tanımlanmadan kullanıldığını belirtecektir. Doğru kullanımında mutlaka değişken tanımlanmalıdır.

```
<%
  Option Explicit
  Dim Ad
  Ad = "Günce Akkoyun"
%>
```

Örnek 15 : Geçerli Değişken Tanımlama ve Değer Atama.

Şeklinde olacaktır. Bu şekilde yapılan değişken tanımlamalarında programın ilerleyen aşamalarında tekrar "Ad" isimli bir değişken kullanamayacağımız için bir karışıklık olmayacaktır. Bu size şu anda anlamsız gelebilir ama büyük kodlar (10000-20000 satır) olduğu zaman bu size anlamlı gelecektir.

Temelde kullanılandığımız değişken türleri sayısal, alfa-sayısal ve mantıksal olmak üzere üçe ayrılmaktadır. Ancak VbScript, her programlama dilinde karşımıza çıkan integer, real, string, boolean vs.. gibi temel veri tipi tanımlarından yoksun bir dildir. Peki VbScript bir değişkenin hangi türde olduğunu nasıl anlar? Bunu o değişkene değer atandığı zaman anlar. Aşağıdaki atamaları inceleyelim,

```
<%
  Option Explicit
  Dim x, y, z, t, u

    x = 15
    y = "15"
    z = "Zeytinyağlı yiyemem aman"
    t = True
    u = ASPgüncesi
%>
```

Örnek 16 : Değişkene Değişik Veri Tipleri Atama.

Yukarıdaki örneğimizde "x" değişkenine sayısal bir değer atadığımız için o andan itibaren "x" değişkeni sayısal bir veri tipinde olacaktır (bu tip belirleme VbScript tarafından otomatik olarak yapılmaktadır). Bu nedenle tüm sayısal işlemlerde kullanılabilir. Durum "y" değişkeni için ise biraz farklıdır. "y" değişkeni tırnak içerisinde olduğundan dolayı alfa-sayısal bir veri tipinde olacaktır. Bu nedenle sayısal işlemlere girmeyecek ve sadece metinsel işlemlerde kullanılacaktır. "z" değişkeninde ise sayısal içeriğe sahip olmayan, metinsel bir değere sahip olan bir değişkeni temsil etmektedir. "t" değişkeninde ise sadece "doğru" (true - 1) veya "yanlış" (false - 0) değerini alabilen bir boolean veri tipi gösterilmektedir. "u" değişkeni de ne oluyor diye düşünüyor olmalısınız.

Düşünmekte haklısınız çünkü metin değişkenleri sadece tırnak işareti içerisinde tanımlanabilir diğer hallerde hata verecektir.

2.1.1. Seri Değişkenler

Vbscript programları içerisinde bir değişkene birden çok değer verilebilmektedir. Bu olaya "Seri Değişken" denilmektedir. Örnek vermek gerekirse 7 kişilik bir sınıf içersindeki öğrencilerin isimlerini tek bir değişkende tutabiliriz.

```
<%
  Option Explicit
  Dim Ad(7)

    Ad(1) = "Günce"
    Ad(2) = "Refiye"
    Ad(3) = "Haldun"
    Ad(4) = "Meral"
    Ad(5) = "Alp"
    Ad(6) = "Seda"
    Ad(7) = "Gökçe"
%>
```

Örnek 17 : Diziye Değer Atama.

Şeklinde bir tanımlama yaptığımız zaman "Ad" değişkenine 7 farklı değer vermiş olduk. Kaç farklı değer alabileceğini "DIM" komutu ile değişken tanımlarken parantez içerisinde boyutlandırmış olduk. Bu örnekte neden her isim için farklı değişken kullanmadık ta bir seri kullandık diyecek olursanız. Yani

```
<%
  Option Explicit
  Dim Ad1, Ad2, Ad3, Ad4, Ad5, Ad6, Ad7

    Ad1 = "Günce"
    Ad2 = "Refiye"
    Ad3 = "Haldun"
    Ad4 = "Meral"
    Ad5 = "Alp"
    Ad6 = "Seda"
    Ad7 = "Gökçe"
%>
```

Örnek 18 : Kullanışsız Değişken Dizisi.

Hemen cevabını verelim; dizi değişkenlerde bir değişkeni numarasıyla kullanabilirsiniz ve numara yerine başka bir değişken kullanabilirsiniz.

Bilgi : String özelliği taşıyan değişkenler (yani tırnak içerisinde tanımlanmış metinsel değere sahip değişkenler) hiçbir şekilde dönüştürme işlemi yapılmadan matematiksel işlem içerisinde kullanılamaz.

```
<%
  ...
  numara = 6
  response.write
  Ad(numara)
  ...
%>
```

Örnek 19 : Dizi İçindeki Bir Öğenin Kullanımı.

Şeklinde bir kullanım ile 6 numaralı "Ad" değişkenini ekrana yazdırmak gibi bir işlem gerçekleşecektir. (bu programcıkta ilk olarak "numara" değişkenine 6 değerini atadık ve bu "numara" değişkenini "Ad(numara)" şeklinde seri içerisinde kullandık.)

Bir dizi değişkeni boyutu isteklerimizin altındaysa yani elimizdeki toplam dizi öğesi adedi, dizinin tanım aralığı dışında kalmış ise, dizimizin tanım aralığını yani boyutunu genişletebiliriz. Bunun için "redim" komutu kullanılır (yeniden boyutlandır). Fakat bu komutun şöyle bir özelliği vardır: yeniden tanımlanan bir dizi yeniden tanımlama öncesi kendisine verilen hiçbir değeri yeniden boyutlandırma sonrası vermez. Bir diğer değişke içeriği sıfırlanır.

```
<%
  Option Explicit
  Dim Ad(5)

    Ad(0) = "Günce"
    Ad(1) = "Refiye"
    Ad(2) = "Haldun"
    Ad(3) = "Meral"
    Ad(4) = "Alp"

  ReDim Ad(7)

    Ad(5) = "Seda"
    Ad(6) = "Gökçe"
%>
```

Örnek 20 : Redim Komutu ile Diziyi Yeniden Boyutlandırma.

Verilen örnekte yeniden tanımlama sonrası "Ad(0) – Ad(4)" değişkenleri sıfırlanacak yani null içeriğe sahip olacaktır. Yeniden tanımlama sırasında eski dizi değerlerinin korunmasını istersek "Redim" komutunu "Preserve" komutuyla beraber kullanmalıyız. Bu şekilde eskiye dönük verinin korunacağını yeniden tanımlama sırasında belirtmiş oluruz.

```
<%
  Option Explicit
  Dim Ad(5)

    Ad(0) = "Günce"
    Ad(1) = "Refiye"
    Ad(2) = "Haldun"
    Ad(3) = "Meral"
    Ad(4) = "Alp"

  ReDim Preserve Ad(7)

    Ad(5) = "Seda"
    Ad(6) = "Gökçe"
%>
```

Örnek 21 : Redim Preserve Komutu ile Diziyi Yeniden Tanımlama.

Bu şekilde yeniden tanımlanan dizi eski değerlerinden hiçbir veriyi kaybetmez. Böylece eski verilerde kullanılabilmektedir.

Dizi değişkenleri yukarıdaki örneklerdeki gibi tek boyutlu olabilmesinin yanında birden çok boyuta da sahip olabilirler. Örnek vermek gerekirse matematikte kullanılan matriks sistemini düşünün bu size iki boyutlu dizi hakkında fikir verecektir. Matriks

üzerinde birinci satır birinci sütunda ki değer x(1,1) şeklinde ifade edilmektedir. Örneğin aşağıdaki 4x4 matriksi ASP içerisinde tanımlamamız için,

$$\begin{vmatrix} 25 & 7 \\ 19 & 97 \end{vmatrix}$$

```
<%
  Option Explicit
  Dim x(7,5)
    x(1,1) = 25
    x(1,2) = 7
    x(2,1) = 19
    x(2,2) = 97
%>
```

Örnek 22 : Çok Boyutlu Dizi Kullanımı.

Şeklinde bir değişken ataması yapabiliriz. Bu atamada iki boyutlu bir matriksi VbScript'imize tanımlamış olduk aynı yöntemle üç ve dört boyutlu seriler tanımlayabilmekteyiz.

2.2.Sabitler

Program içerisinde her zaman değişken değerleri değişebilir değil bazen de sabit olmaları istenebilir. Bu tip değişkenlere (değişken demek biraz yanlış ama..) bir kere değer atandıktan sonra bir daha değiştirilemez (sabit değer bütün program boyunca (hatta isterseniz bütün site boyunca) değişmeden kalır). Bu işlemi vbscript içerisinde "Constant" (Sabit) kelimesinin kısaltılmışı olan "Const" komutu yardımı ile yapılır. Const terimi ile değer verilmiş bir değişkenin tanımlanmasına gerek yoktur.

Sabit olduğu "Const" komutu ile belirtilmiş bir değişkene..

```
<%
  Option Explicit
  Const Url = "www.akkoyun.net"
%>
```

Örnek 23 : Sabit Kullanımı.

Şeklinde kullanılan bir sabit değişkene başka bir veri atanmaya çalışıldığı zaman hata verecektir.

```
<%
  Option Explicit
  Const Url = "akkoyun.net"
  ....
  Url = "pcworld.com.tr"
%>
```

Örnek 24 : Bir Sabite Değer Atamak.

Yani "Url" sabitine kendi değeri dışında bir değer verilmek istenmiştir fakat hata mesajı verecektir. (Illegal assignment: 'Url').

Vbscript programlama dilinde kullanılan değişkenlere genel olarak variant denir. Variant genel olarak bir grubu temsil eder ve alt grupları mevcuttur. Bunlar değişkenin cinsine göre otomatik olarak tanınır. (diğer programlama dillerinde bu özellik yoktur ve değişken tanımlanırken veri tipi yani variant grubu tanımlanır fakat asp de bu olay yordamcı tarafından otomatik gerçekleşir).

VbScript içerisinde kullanıcıya kolaylık sağlamak için bazı hazır tanımlanmış sabitler mevcuttur. Bu sabitler kodlama sırasında istenildiği şekilde kullanılabilmektedir. Bu sabitler ve değerleri aşağıda verilmiştir.

2.2.1. Renk Sabitleri

Sabit	Değer	Açıklama
vbBlack	&h00	Siyah
vbRed	&hFF	Kırmızı
vbGreen	&hFF00	Yeşil
vbYellow	&hFFFF	Sarı
vbBlue	&hFF0000	Mavi
vbMagenta	&hFF00FF	Gül Kurusu
vbCyan	&hFFFF00	Turkuaz
vbWhite	&hFFFFFF	Beyaz

Tablo 2 : Renk Sabitleri.

2.2.2. Tarih ve Zaman Sabitleri

Sabit	Değer	Açıklama
vbSunday	1	Pazar
vbMonday	2	Pazartesi
vbTuesday	3	Salı
vbWednesday	4	Çarşamba
vbThursday	5	Perşembe
vbFriday	6	Cuma
vbSaturday	7	Cumartesi
vbUseSystem	0	Sunucunun bölgesel ayarlarında yer alan tarih veya zaman biçimini kullan
vbUseSystemDayOfWeek	0	
vbFirstJan1	1	1 Ocak gününün içerisinde yer aldığı haftayı kullan
vbFirstFourDays	2	En az 4 günü yeni yılda olan ilk haftayı kullan
vbFirstFullWeek	3	7 günüde yeni yıl içersinde yer alan ilk haftayı kullan

Tablo 3 : Tarih ve Zaman Sabitleri.

2.2.3. Tarih Biçim Sabitleri

Sabit	Değer	Açıklama
vbGeneralDate	0	Sistem ayarlarındaki gösterim biçimi
vbLongDate	1	Uzun tarih gösterimi (June 26, 1943)
vbShortDate	2	Kısa tarih gösterimi (6/26/43)
vbLongTime	3	Uzun zaman gösterimi (3:48:01 Pm)
vbShortTime	4	Kısa zaman gösterimi (15:48)

Tablo 4 : Tarih Biçim Sabitleri.

2.2.4. Çeşitli Sabitler

Sabit	Değer	Açıklama
vbObjectError	-2147221504	Kullanıcı tarafından tanımlanan hata tipi numaraları

Tablo 5 : Çeşitli Sabitler.

2.2.5. Mesaj Kutusu Sabitleri

Sabit	Değer	Açıklama
vbOkOnly	0	Sadece OK butonu göster
vbOkCancel	1	Ok ve İptal butonu göster
vbAbortRetryIgnore	2	Abort, Retry ve Ignore butonu göster
vbYesNoCancel	3	Yes, No ve Cancel butonu göster
vbYesNo	4	Yes ve No butonu göster
vbRetryCancel	5	Retry ve Cancel butonu göster
vbCritical	16	Kritik mesaj iconu göster
vbQuestion	32	Uyarı soru mesajı iconu göster
vbExclamation	48	Uyarı mesajı iconu göster
vbInformation	64	Bilgi mesajı iconu göster
vbDefaultButton1	0	Ilk buton varsayılan
vbDefaultButton2	256	Ikinci buton varsayılan
vbDefaultButton3	512	Üçüncü buton varsayılan
vbDefaultButton4	768	Dördüncü buton varsayılan

Tablo 6 : Mesaj Kutusu Sabitleri.

2.2.6. Mesaj Kutusu Cevap Sabitleri

Sabit	Değer	Açıklama
vbOk	1	Ok butonu tıklandı
vbCancel	2	Cancel butonu tıklandı
vbAbort	3	Abort butonu tıklandı
vbRetry	4	Retry butonu tıklandı
vbIgnore	5	Ignore butonu tıklandı
vbYes	6	Yes butonu tıklandı
vbNo	7	No butonu tıklandı

Tablo 7 : Mesaj Kutusu Cevap Sabitleri.

2.2.7. String Cevap Sabitleri

Sabit	Değer	Açıklama
vbCr	Chr(13)	-
vbCrLf	Chr(13) & Chr(10)	-
vbFormFeed	Chr(12)	Form Besleme
vbLf	Chr(10)	Satır Besleme
vbNewLine	Chr(13) & Chr(10) veya Chr(10)	Yeni Satır
vbNullChar	Chr(0)	0 değerine sahip karakter
vbNullString	-	“”
vbTab	Chr(9)	Horizantal Tab
vbVerticalTab	Chr(11)	Vertical Tab

Tablo 8 : String Cevap Sabitleri.

2.2.8. Durum Sabitleri

Sabit	Değer	Açıklama
vbTrue	-1	Doğru
vbFalse	0	Yanlış

Tablo 9 : Durum Sabitleri.

2.2.9. Karşılaştırma Sabitleri

Sabit	Değer	Açıklama
VbBinaryCompare	0	Binary Karşılaştırma
VBTextCompare	1	Metin Karşılaştırma
VBDataBaseCompare	2	Veri Tabanı Karşılaştırması

Tablo 10 : Karşılaştırma Sabitleri.

2.2.10. Değişken Tipi Sabitleri

Sabit	Değer	Açıklama
vbEmpty	0	Tanımsız data
vbNull	1	Geçerli data içermeyen
vbInteger	2	Integer
vbLong	3	Long
vbSingle	4	Single
vbCurrency	6	Currency
vbDate	7	Date
vbString	8	String
vbObject	9	Object
vbError	10	Error
vbBoolean	11	Boolean
vbVariant	12	Variant
vbDataObject	13	Data object
vbDecimal	14	Decimal
vbByte	15	Byte
vbArray	16	Array

Tablo 11 : Değişken Tipi Sabitleri.

2.2.11. Sürücü Tipi Sabitleri

Sabit	Değer	Açıklama
Unknown	0	Tanımlanmamış sürücü
Removable	1	Tüm taşınabilir medya aygıtları. Içerisinde disket sürücüleri de vardır
Fixed	2	Sabit diskler
Remote	3	Network sürücüleri
CDROM	4	CdRom sürücüsü
RamDisk	5	RAM üzerinde yer alan sanal disk alanı

Tablo 12 : Sürücü Tipi Sabitleri.

2.2.12. Dosya Özelliği Sabitleri

Sabit	Değer	Açıklama
Normal	0	Normal dosya

ReadOnly	1	Salt okunur
Hidden	2	Gizli
System	4	Sistem dosyası
Directory	16	Dizin
Archive	32	Arşiv
Alias	1024	Link veya kısayol
Compressed	2048	Sıkıştırılmış dosya

Tablo 13 : Dosya Özelliği Sabitleri.

2.2.13. Dosya Girdi Çıktı Sabitleri

Sabit	Değer	Açıklama
ForReading	0	Dosyayı sadece okuma için aç. Bu tip açılan dosyaya yazılamaz.
ForWriting	1	Dosyayı yazma için aç. Eğer bu dosya daha önceden varsa üzerine yazar.
ForAppending	8	Dosyayı yazmak için açar ve sonuna yazar

Tablo 14 : Dosya Girdi Çıktı Sabitleri.

2.2.14. Özel Dizin Sabitleri

Sabit	Değer	Açıklama
WindowsFolder	0	Windowsun kurulu olduğu dizin
SystemFolder	1	Fontların ve sürücülerin bulunduğu sistem dizini
TemporaryFolder	2	Temprorary dizini

Tablo 15 : Özel Dizin Sabitleri.

2.3.Operatörler

Değişkenlerimiz arasında işlemler yaptırabiliriz. Bu işlemler için operatör dediğimiz işaretleri kullanırız. Bu operatörler bildiğiniz matematiksel operatörlerdir.

Aritmetik		**Karşılaştırma**		**Lojik**	
Açıklama	**Sembol**	**Açıklama**	**Sembol**	**Açıklama**	**Sembol**
Üst Alma	^	Eşitlik	=	Lojik Zıtlık	Not
Çıkarma	-	Eşitsizlik	<>	Lojik ve	And
Çarpma	*	Küçüktür	<	Lojik veya	Or
Bölme	/	Büyüktür	>	Lojik özel veya	Xor
Integer Bölme	\	Küçük Eşittir	<=	Lojik eşdeğer	Eqv
Modüler	Mod	Büyük Eşittir	>=	Lojik içerme	Imp
Toplama	+	Obje Eşdeğeri	Is		
Metinsel Birleştirme	&				

Tablo 16 : Operatörler.

Kullanımları ise şu şekildedir.

2.3.1. Aritmetik Operatörler

2.3.1.1. Üst Alma

İstenilen sayısal değerin (integer) istenilen sayısal kuvvetini almaya yarayan operatördür. Üstü alınan sayı veya üst değeri "Null" ise sonuçta "Null" olacaktır. Üst alma operatörü SHIFT+3 tuşlarına basarak elde edilebilir.

$$x^y$$

```
<%
  Option Explicit
  Dim x, y, Sonuc
  x = 25
  y = 7
  Sonuc = x ^ y
  respose.write Sonuc
%>
```

Örnek 25 : Üst Alma.

Sonuç "6103515625" olacaktır.

2.3.1.2. Matematiksel İşlemler (+,-,*,\,/)

Matematikte kullanıldığı gibi istenilen şekilde işlemlerde kullanılabilir fakat unutulmaması gereken bir kaç nokta vardır. Bunlardan birincisi işlem yapılacak değişkenin veya sabitin numerik sayı değerine sahip olduğundan yani değişken alt tipinin uygun olduğundan emin olunuz. Eğer emin olamıyorsanız dönüştürme işlemi ile bu veri tipine çeviriniz. İkinci dikkat etmeniz gereken nokta ise işlemlerin sırasıdır. Bu sıra çarpma, bölme, toplama ve çıkartma sırasına göre yapılamaktadır.

$$\left[\frac{(a+b)}{(b*c)}\right]*d$$

```
<%
  Option Explicit
  Dim a, b, c, d,Sonuc
  a = 1
  b = 31
  c = 11
  d = 4
  Sonuc = [(a+b)/( b*c)]*d
  respose.write Sonuc
%>
```

Örnek 26 : Matematiksel Eşitlikler.

Sonuç olarak "0,3753" dönecektir.

2.3.1.3. Modüler Aritmetik

İstenilen bir sayının modüler aritmetiğe uygun olarak hangi sistemde hangi değeri alacağını veriri. Matematikte kullanılan "Mod" fonksiyonu ile aynıdır hiçbir farkı yoktur.

$$a \quad Mod \quad b$$

$$a_b$$

```
<%
  Option Explicit
  Dim a, b, Sonuc
  a = 25
  b = 7
  Sonuc = a Mod d
  respose.write Sonuc
%>
```

Örnek 27 : Modüler Aritmetik.

Yukarıdaki örnekte 7'lik tabanda 25 değerini verecektir. Yani "4" değeri dönecektir.

2.3.1.4. Metin Birleştirme

Metin birleştirme işlemi temel olarak iki veya daha çok karakter tabanlı değişkenin tek bir değişken gibi birleştirilmesine dayanır. Bu işlemi yaparken operatör seçimimize dikkat etmemiz gerekecektir.

```
<%
  Option Explicit
  Dim Ad,Soyad,isim

    Ad = "Günce "
    Soyad = "Akkoyun"
    isim = Ad & Soyad

  respose.write isim
%>
```

Örnek 28 : Metin Birleştirme.

Sonuç "GünceAkkoyun" şeklinde olacaktır. Unutmamak gerekir ki her operatör her veri tipinde kullanılamaz. Sayısal veya metinsel özellikteki verilere göre kullanım alanları değişir. Son olarak operatörlerin kullanımı sırasında veri tipine uygun operatör kullanmak gereklidir. Örneğin iki karakter değişkenini toplarken (birleştirirken) "+" yerine metinsel işlemlerde birleştirme anlamına gelen "&" sembolü kullanılmalıdır.

Not : İki veya daha fazla metin birleştirilirken, örneğin bir "Ad" ve "Soyad", bu ikisi arasında boşluk bırakılmadan birleştirilecektir. Oysaki çoğu durumlarda arada boşluk bırakılması istenebilir. Böyle durumlarda manuel olarak araya boşluk bırakılmalıdır.

2.3.2. Karşılaştırma Operatörleri

Programlarımız içerisinde kimi zaman iki farklı değişkeni birbirine göre kıyaslamamız (karşılaştırmamız) gerekebilmektedir. Bu gibi durumlarda karşılaştırma operatörleri kullanılmaktadır. Karşılaştırma operatörleri sayesinde iki değişken birbirine eşit mi?, eşit değil mi?, büyük mü?, küçük mü? veya bunların kombinasyonları şeklinde sınarız.

2.3.3. Lojik Operatörler

Lojik operatörler karşılaştırma gibi kullanılabildikleri gibi birden çok operatörü birleştirmek içinde kullanılabilir (and veya or ile). Bu kullanımın dışında pek fazla kullanımı yoktur daha çok mantıksal sorgularda birkaç şartı birleştirmek için kullanılır. Özellikle "and", "or" veya "not" operatörleri çok fazla kullanılmaktadır.

2.4.Hazır Fonksiyonlar

Vbscript içerisinde değişkenlerimizi kontrol altında tutabilmemiz için bir dizi hazır fonksiyon bulunmaktadır. Bu fonksiyonları gruplar halinde inceleyeceğiz. Tüm bu fonksiyonları beş ana grupta toplayabiliriz.

1. Metin işlemleri
2. Sayısal işlemler
3. Tarih işlemleri
4. Test işlemleri
5. Dönüştürme işlemleri
6. Biçimlendirme işlemleri

2.4.1. Metin işlemleri

Metin işlemleri isminden anlaşılacağı metinler üzerinde işlemler yapmamıza olanak tanıyan fonksiyonlardır. Bu fonksiyonlar diğer programlama dillerinde de aynen vardır. Kullanımları sadece metinsel tabanlı değişkenler (string,char,byte..) üzerinde mümkündür.

2.4.1.1. Asc

Açıklama

İçerisine gönderilen metinin (veya karakterin) ilk harfine ait ANSI karakter kodunu gönderir. ANSI karşılık tablosu bu kitap içerisinde yer alan tabloda sunulmuştur.

Yazım

```
Deger = Asc(String)
```

Fonksiyon içerisine gönderilen değer mutlaka bir karakter tabanlı değer içermek zorundadır. Fonksiyona gönderilen string eğer boş olursa çalışma zamanı hatası oluşacaktır.

Örnek

```
<%
  Option Explicit
  Dim Deger

    Deger = ASC("A")          ' 65 değeri döner.
    Deger = ASC("a")          ' 97 değeri döner.
    Deger = ASC("Armut")      ' 65 değeri döner.
%>
```

Örnek 29 : ASC Fonksiyonu.

2.4.1.2. Chr

Açıklama

Asc fonksiyonunun tam tersini yapar ve verilen numerik değerleri ANSI karşılığı olan karaktere çevirir.

Yazım

```
Deger = Chr(numeric)
```

Fonksiyon içerisine gönderilen değer mutlaka bir sayısal değer içermek zorundadır. Fonksiyona gönderilen numerik değer 0-31 aralığında olduğu zaman bir karakter görüntülenmez, bunun sebebi ise bu aralıktaki karakterlerin özel karakterler olmasıdır. Örneğin Chr(10) satır atlatma kodudur ve görüntülenmesi mümkün değildir (kullanıldığı satırın bir altındaki satıra geçiş sağlar).

Örnekler

```
<%
  Option Explicit
  Dim Karakter

    Karakter = Chr(65)          ' A değeri döner.
    Karakter = Chr(97)          ' a değeri döner.
    Karakter = Chr(62)          ' > değeri döner.
    Karakter = Chr(37)          ' % değeri döner.
%>
```

Örnek 30 : Chr Fonksiyonu.

2.4.1.3. Lcase

Açıklama

Lcase fonksiyonu içerisine gönderilen karakterleri veya karakter setlerini tamamen küçük harfe dönüştürülmüş olarak geri verir.

Yazım

```
Deger = Lcase(string)
```

Fonksiyon içerisine gönderilen değer mutlaka bir metinsel değer içermek zorundadır. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır. Dikkat edilecek nokta şudur fonksiyona gönderilen metindeki sadece büyük harfler küçük harfe dönüştürülecektir. Metin içerisindeki küçük harfler veya metin tabanlı olmayan harfler bir dönüşüme uğramayacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, KucukMetin

    Degisken = "VBScript"
    KucukMetin = LCase(Degisken)        '   "vbscript" değeri döner.
%>
```

Örnek 31 : LCase Fonksiyonu.

2.4.1.4. Ucase

Açıklama

Ucase fonksiyonu çalışma olarak Lcase fonksiyonunun tamamen zıttıdır. Ucase fonksiyonu içerisine gönderilen karakterleri veya karakter setlerini tamamen büyük harfe dönüştürülmüş olarak geri verir.

Yazım

```
Deger = Ucase(string)
```

Fonksiyon içerisine gönderilen değer mutlaka bir metinsel değer içermek zorundadır. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır. Dikkat edilecek nokta şudur fonksiyona gönderilen metindeki sadece küçük harfler büyük harfe dönüştürülecektir. Metin içerisindeki büyük harfler veya metin tabanlı olmayan harfler bir dönüşüme uğramayacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, BuyukMetin

    Degisken = "VBScript"
    BuyukMetin = UCase(Degisken)        '   "VBSCRIPT" değeri döner.
%>
```

Örnek 32 : UCase Fonksiyonu.

2.4.1.5. Left

Açıklama

Left fonksiyonu verilen bir metinsel değişkenin solundan belirtilen kadar karakteri alması için kullanılır. Soldaki birinci harf her zaman 1. harf olacaktır.

Yazım

```
Deger = Left(string, sayı)
```

Fonksiyon içerisine gönderilen stringin solundan itibaren yine fonksiyona gönderilen sayı kadar karakterini alacaktır. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, SolMetin

    Degisken = "VBScript"
    SolMetin = Left(Degisken, 3)        '   "VBS" değeri döner.
%>
```

Örnek 33 : Left Fonksiyonu.

2.4.1.6. Right

Açıklama

Right fonksiyonu verilen bir metinsel değişkenin sağından belirtilen kadar karakteri alması için kullanılır. Sağdaki ilk harf her zaman 1. harf olacaktır.

Yazım

```
Deger = Right(string, sayı)
```

Fonksiyon içerisine gönderilen stringin solundan itibaren yine fonksiyona gönderilen sayı kadar karakterini alacaktır. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, SagMetin

    Degisken = "VBScript"
    SagMetin = Right(Degisken, 6)        '   "Script" değeri döner.
%>
```

Örnek 34 : Right Fonksiyonu.

2.4.1.7. Mid

Açıklama

Bir string içerisinde başlangıç noktasını ve karakter olarak boyutunu verdiğimiz alanda yer alan stringi verecektir. Soldaki ilk harf 1. harf olarak kabul edilmektedir.

Yazım

```
Deger = Mid(string, başlangıç[, uzunluk])
```

Fonksiyon içerisine gönderilen stringin solundan itibaren yine fonksiyona gönderilen başlangıç değeri kadara içeriden, fonksiyona gönderilen uzunluk değeri kadar bir kısmı keser alır. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır. Uzunluk opsiyonu zorunlu değildir gönderilmediği zaman başlangıçtan son harfe kadar olan kısmı alır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Metin

    Degisken = "Erhan Arı"
    Metin = Mid(Degisken, 7, 3)        '   "Arı" değeri döner.
%>
```

Örnek 35 : Mid Fonksiyonu.

2.4.1.8. Len

Açıklama

Fonksiyon içerisine gönderilen metinsel değişkenin karaktersel olarak uzunluğunu bize verir.

Yazım

```
Deger = Len(string)
```

Fonksiyon içerisine gönderilen stringin solundan itibaren boyu sayılacak ve fonksiyon çıktısı olarak verilecektir. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Uzunluk

    Degisken = "Arif Haldun Özer"
    Uzunluk = Len(Degisken)        '   16 değeri döner.
%>
```

Örnek 36 : Len Fonksiyonu.

2.4.1.9. Ltrim

Açıklama

Fonksiyon içerisine gönderilen metinsel değişkenin solunda yer alan boşlukları atar ve değişkenin solunda yer alan boşluklar hariç diğer kısımları alır.

Yazım

```
Deger = LTrim(string)
```

Fonksiyon içerisine gönderilen stringin solundan itibaren kontrol ederek solda yer alan boşlukları yok eder. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = LTrim("   Alp İlhan ")        '   "Alp İlhan " değeri döner.
%>
```

Örnek 37 : LTrim Fonksiyonu.

2.4.1.10. Rtrim

Açıklama

Fonksiyon içerisine gönderilen metinsel değişkenin sağında yer alan boşlukları atar ve değişkenin sağında yer alan boşluklar hariç diğer kısımları alır.

Yazım

```
Deger = RTrim(string)
```

Fonksiyon içerisine gönderilen stringin sağından itibaren kontrol ederek sağda yer alan boşlukları yok eder. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = RTrim("   Alp İlhan ")        '   "   Alp İlhan" değeri döner.
%>
```

Örnek 38 : RTrim Fonksiyonu.

2.4.1.11. Trim

Açıklama

Fonksiyon içerisine gönderilen metinsel değişkenin sağında ve solunda yer alan boşlukları atar ve değişkenin sağ ve sol boşlukları hariç diğer kısımları alır.

Yazım

```
Deger = Trim(string)
```

Fonksiyon içerisine gönderilen stringin sağında ve solunda yer alan boşlukları yok eder. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

   Degisken = Trim("   Alp İlhan ")        '   "Alp İlhan" değeri döner.
%>
```

Örnek 39 : Trim Fonksiyonu.

2.4.1.12. Space

Açıklama

Fonksiyon içerisine gönderilen sayı değeri kadar boşluktan oluşan bir metinsel değişken oluşturur.

Yazım

```
Deger = Space(Sayı)
```

Fonksiyon içerisine gönderilen sayı değeri kadar boşluk olur. Fonksiyona gönderilen değer "Null" olursa geriye dönen değerde "Null" olacaktır.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

   Degisken = Space(10)                          ' 10 adet boşluk değeri döner.
   Degisken = "Merhaba" & Space(1) & "Dünya"   ' "Merhaba Dünya" değeri döner.
%>
```

Örnek 40 : Space Fonksiyonu.

2.4.1.13. String

Açıklama

Fonksiyon içerisine gönderilen sayı değeri kadar yine fonksiyon içerisine gönderilen karakteri yan yana koyar. Mantık olarak "space" fonksiyonuna çok benzer.

Yazım

```
Deger = String(Sayı, Karakter)
```

Fonksiyon içerisine gönderilen sayı değeri kadar karakteri yan yana koyar fakat fonksiyon içerisine gönderilen karakterin sadece ilk harfini alır yani karakter yerine bir metin girilirse sadece metnin ilk harfini alır. Bunun dışında karakter yerine karakterin ANSI kodunu da yazabilirsiniz.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = String(5, "*")        ' "*****" değeri döner.
    Degisken = String(5, 42)         ' "*****" değeri döner.
    Degisken = String(10, "ABC")     ' "AAAAAAAAAA" değeri döner.
%>
```

2.4.1.14. Replace

Açıklama

Verilen string içerisinde istenilen bölümleri bulup başka bir string ile değiştirmeye yarar.

Yazım

```
Deger = Replace(Metin, bulunacakmetin, değiştirilecekmetin)
```

Fonksiyon içerisine gönderilen metin içerisinde bulunacakmetini arayarak yerine değiştirilecekmetin i değiştirir.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Replace("XXpXXPXXp", "p", "Y")        ' "XXYXXPXXY" değeri döner.
%>
```

Örnek 41 : Replace Fonksiyonu.

2.4.1.15. InStr

Açıklama

Uzun bir değişken içerisinde (string) daha kısa bir değişkenin (string) bulunup bulunmadığını arar; bulursa kısa değişkenin uzun değişken içerisinde kaçıncı karakterden itibaren başladığını belirtir.

Yazım

```
Deger = InStr(Metin, aranacakmetin)
```

Fonksiyon içerisine gönderilen metin içerisinde aranacakmetin'i bulup soldan itibaren kaçıncı karakterden başladığını verir.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Bul, Sonuc

    Degisken = "XXpXXpXXPXXP"
    Bul = "P"
    Sonuc = Instr(Degisken, Bul)        ' 9 değeri döner.
%>
```

Örnek 42 : Instr Fonksiyonu.

2.4.1.16. InStrRev

Açıklama

Uzun bir değişken içerisinde (string) daha kısa bir değişkenin (string) bulunup bulunmadığını metinin sonundan itibaren arar; bulursa kısa değişkenin uzun değişken içerisinde sondan kaçıncı karakterden itibaren başladığını belirtir.

Yazım

```
Deger = InStrRev(Metin, aranacakmetin)
```

Fonksiyon içerisine gönderilen metin içerisinde aranacakmetin'i bulup sağdan itibaren kaçıncı karakterden başladığını verir.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Bul, Sonuc

    Degisken = "XXpXXpXXPXXP"
    Bul = "P"
    Sonuc = Instrrev(Degisken, Bul)        ' 1 değeri döner.
%>
```

Örnek 43 : Instrrev fonksiyonu.

2.4.1.17. StrReverse

Açıklama

Fonksiyon içerisine gönderilen metinin tersten okunuşu olarak bize verir.

Yazım

```
Deger = StrReverse(Metin)
```

Fonksiyon içerisine gönderilen metinin tersten okunuşunu verir.

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Sonuc

    Degisken = "VbScript"
    Sonuc = StrReverse(Degisken)        ' "tpircSbV" değeri döner.
```

```
%>
```

Örnek 44 : StrReverse Fonksiyonu.

2.4.2. Sayısal işlemleri

Kullanılan tüm sayısal fonksiyonların kullanım biçimi aynıdır. Fonksiyon sonucu bir değişkene atılır ve işlemlerimize devam edilir. Matematikte kullanılan tüm fonksiyonlar aynen geçerlidir bunların bazıları Cos, Sin, Tan, Exp, vb.. şeklindedir. Kısa kısa bu fonksiyonlara değinelim.

2.4.2.1. Abs

Açıklama

Fonksiyon içerisine gönderilen sayının tam değerini verir. Negatif sayıları tam değer fonksiyonu ile kullandığımız zaman pozitif değer dönecektir.

Yazım

```
Deger = Abs(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

   Degisken = Abs(50.3)          ' 50.3 değeri döner.
   Degisken = Abs(-50.3)         ' 50.3 değeri döner.
%>
```

Örnek 45 : Abs Fonksiyonu.

2.4.2.2. Atn

Açıklama

Fonksiyon içerisine gönderilen sayının arctanjantını verir. Fakat bu işlemi radyan üzerinde yapar.

Yazım

```
Deger = Atn(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Pi
   Pi = Atn(1) * 4            ' pi değeri (3.14159276) döner.
%>
```

Örnek 46 : Atn Fonksiyonu.

2.4.2.3. Log

Açıklama

Fonksiyon içerisine gönderilen sayının doğal logaritmasını verir.

Yazım

```
Deger = Log(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Log(10)          ' e tabanında logaritma 10 u hesaplar.
%>
```

Örnek 47 : Log Fonksiyonu.

2.4.2.4. Exp

Açıklama

Doğal logaritma tabanı olan "e" değerinin üstel fonksiyonu yapar. Yani logaritma işleminin tam tersi söz konusudur.

Yazım

```
Deger = Exp(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Exp(10)          ' e üstü 10 u hesaplar.
%>
```

Örnek 48 : Exp Fonksiyonu.

2.4.2.5. Cos, Sin, Tan

Açıklama

Sayının radyan degeri olarak trigonometrik fonksiyonunu hesaplar. Tüm trigonometrik fonksiyonların kullanımları aynıdır. Eğer derece olarak çalışılacaksa mutlaka dönüşüm işlemi yapılmalıdır.

Yazım

```
Deger = Cos(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Cos(10)          ' radyan olarak 10 un kosinisinü verir.
%>
```

Örnek 49 : Cos Fonksiyonu.

2.4.2.6. Sqr

Açıklama

Sayının karekökünü verir.

Yazım

```
Deger = Sqr(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Sqr(25)            ' 5 değerini verir.
%>
```

Örnek 50 : Sqr Fonksiyonu.

2.4.2.7. Rnd

Açıklama

Rastgele bir sayı geri döndürür.

Yazım

```
Deger = Rnd(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Rnd(25)            ' rastgele bir değer verir.
%>
```

Örnek 51 : Rnd Fonksiyonu.

2.4.2.8. Round

Açıklama

Fonksiyona gönderilen sayıyı istenilen basamak kadar yuvarlar.

Yazım

```
Deger = Round(Sayı, basamak)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Round(3.14159276, 2)            ' 3.14 değerini verir.
%>
```

Örnek 52 : Round Fonksiyonu.

2.4.2.9. Int, Fix

Açıklama

Fonksiyona gönderilen sayının tam sayısını verir.

Yazım

```
Deger = Int(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Int(99.8)          ' 99 değerini verir.
    Degisken = Fix(99.8)          ' 99 değerini verir.
    Degisken = Int(-99.8)         ' -100 değerini verir.
    Degisken = Fix(-99.8)         ' -99 değerini verir.
    Degisken = Int(-99.2)         ' -100 değerini verir.
    Degisken = Fix(-99.2)         ' -99 değerini verir.
%>
```

Örnek 53 : Int ve Fix Fonksiyonu.

2.4.2.10. Sgn

Açıklama

Fonksiyona gönderilen sayının işaret fonksiyonu değerini verir.

Yazım

```
Deger = Sgn(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Sgn(12)            ' 1 değerini verir.
    Degisken = Sgn(-2.4)          ' -1 değerini verir.
    Degisken = Sgn(0)             ' 0 değerini verir.
%>
```

Örnek 54 : Sgn Fonksiyonu.

2.4.3. Tarih işlemleri

Visual basic programlama dilinin hemen hemen tüm zaman fonksiyonları vbscript'te de aynen kullanılır. Bunlara ek olarak ekstra birkaç fonksiyon eklenmiştir. Tüm tarih fonksiyonlarını sırasıyla inceleyelim;

2.4.3.1. Date

Açıklama

Bu fonksiyon sunucuda ayarlı olan tarih formatına uygun olarak o güne ait tarihi verir.

Yazım

```
Deger = Date
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Date()              ' o günkü tarih değerini verir.
%>
```

Örnek 55 : Date Fonksiyonu.

2.4.3.2. Time

Açıklama

Bu fonksiyon sunucuda ayarlı olan zaman formatına uygun olarak o ana ait zamanı verir.

Yazım

```
Deger = Time
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Time()              ' o an ki zaman değerini verir.
%>
```

Örnek 56 : Time Fonksiyonu.

2.4.3.3. Now

Açıklama

Bu fonksiyon sunucuda ayarlı olan tarih ve zaman formatına uygun olarak o an ki tarih ve zamanı birlikte verir.

Yazım

```
Deger = Now
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Now()               ' o an ki tarih ve zaman değerini verir.
%>
```

Örnek 57 : Now Fonksiyonu.

2.4.3.4. Day

Açıklama

Bu fonksiyon içerisine gönderilen tarihe ait gün değerini sayısal olarak verir. Bu tarih başka fonksiyonların çıktıları olabildiği gibi metinsel tabanlı değişkende olabilmektedir.

Yazım

```
Deger = Day(Tarih)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Day("25.07.1997")          ' "25" değerini verir.
%>
```

Örnek 58 : Day Fonksiyonu.

2.4.3.5. Month

Açıklama

Bu fonksiyon içerisine gönderilen tarihe ait ay değerini sayısal olarak verir. Bu tarih başka fonksiyonların çıktıları olabildiği gibi metinsel tabanlı değişkende olabilmektedir.

Yazım

```
Deger = Month(Tarih)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Month("25.07.1997")          ' "7" değerini verir.
%>
```

Örnek 59 : Month Fonksiyonu.

2.4.3.6. Year

Açıklama

Bu fonksiyon içerisine gönderilen tarihe ait yıl değerini sayısal olarak verir. Bu tarih başka fonksiyonların çıktıları olabildiği gibi metinsel tabanlı değişkende olabilmektedir.

Yazım

```
Deger = Year(Tarih)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Year("25.07.1997")          ' "1997" değerini verir.
%>
```

Örnek 60 : Year Fonksiyonu.

2.4.3.7. WeekDay

Açıklama

Bu fonksiyon içerisine gönderilen tarihin haftanın kaçıncı günü olduğunun değerini sayısal olarak verir. Bu tarih başka fonksiyonların çıktıları olabildiği gibi tarih tabanlı değişkende olabilmektedir. Geriye donen değer sabitler bölümünde anlatılan değerler olacaktır.

Yazım

```
Deger = WeekDay(Tarih)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = WeekDay("25.07.1997")          ' "3" değerini verir.
%>
```

Örnek 61 : Weekday Fonksiyonu.

2.4.3.8. WeekDayName

Açıklama

Bu fonksiyon içerisine gönderilen tarihin haftanın kaçıncı günü olduğunun değerini gün ismi olarak verir. Bu tarih başka fonksiyonların çıktıları olabildiği gibi tarih tabanlı değişkende olabilmektedir. Geriye donen değer metin tabanlı olarak gün ismi olacaktır.

Yazım

```
Deger = WeekDayName(Tarih)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = WeekDayName("25.07.1997")          ' "Salı" değerini verir.
%>
```

Örnek 62 : Weekdayname Fonksiyonu.

2.4.3.9. MonthName

Açıklama

Bu fonksiyon içerisine gönderilen sayısal değere karşılık gelen ay adını verir. Geriye donen değer metin tabanlı olarak ay ismi olacaktır. Fakat bu isim sunucunun bölgesel ayarlarında ayarlandığı şekilde olacaktır.

Yazım

```
Deger = MonthName(Sayı)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

   Degisken = MonthName(11)          ' "Kasım" değerini verir.
%>
```

Örnek 63 : MonthName Fonksiyonu.

2.4.3.10. Hour

Açıklama

Bu fonksiyon içerisine gönderilen zamana ait saat bilgisini sayısal olarak verir. Geriye donen değer sayısal olarak saati verecektir. Bu alınan saat bilgisi bölgesel ayarlara bağlı değildir.

Yazım

```
Deger = Hour(Zaman)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

   Degisken = Hour("15:32:33")          ' "15" değerini verir.
%>
```

Örnek 64 : Hour Fonksiyonu.

2.4.3.11. Minute

Açıklama

Bu fonksiyon içerisine gönderilen zamana ait dakika bilgisini sayısal olarak verir. Geriye donen değer sayısal olarak dakikayı verecektir. Bu alınan dakika bilgisi bölgesel ayarlara bağlı değildir.

Yazım

```
Deger = Minute(Zaman)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken
```

```
    Degisken = Minute("15:32:33")          ' "32" değerini verir.
%>
```

Örnek 65 : Minute Fonksiyonu.

2.4.3.12. Second

Açıklama

Bu fonksiyon içerisine gönderilen zamana ait saniye bilgisini sayısal olarak verir. Geriye donen değer sayısal olarak saniyeyi verecektir. Bu alınan saniye bilgisi bölgesel ayarlara bağlı değildir.

Yazım

```
Deger = Second(Zaman)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = Second("15:32:33")          ' "33" değerini verir.
%>
```

Örnek 66 : Second Fonksiyonu.

2.4.3.13. DateAdd

Açıklama

Bu fonksiyon içerisine gönderilen zamana, belirtilen miktarda belirtilen birimi ekleyecektir. Geriye dönen değer yine tarihsel bir değişken olacaktır. Kullanımda adı geçen birimler aşağıdaki şekillerde olabilir.

```
yyyy        Yıl
q           Çeyrek yıl
m           Ay
d           Gün
w           Hafta
h           Saat
n           Dakika
s           Saniye
```

Yazım

```
Deger = DateAdd(Birim, Miktar, Tarih)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = DateAdd("m", 1, "25.07.2002")      ' "25.08.2002" değerini verir.
    Degisken = DateAdd("yyyy", 3, "25.07.2002")   ' "25.07.2005" değerini verir.
%>
```

Örnek 67 : DateAdd Fonksiyonu.

2.4.3.14. DateDiff

Açıklama

Bu fonksiyon içerisine gönderilen iki farklı zaman arasında belirtilen birimle ne kadar fark olduğunu verecektir. Geriye dönen değer sayısal bir değişken olacaktır. Kullanımda adı geçen birimler aşağıdaki şekillerde olabilir.

```
yyyy        Yıl
q           Çeyrek yıl
m           Ay
d           Gün
w           Hafta
h           Saat
n           Dakika
s           Saniye
```

Yazım

```
Deger = DateDiff(Birim, Tarih1, Tarih2)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

   Degisken = DateDiff("m", "25.07.2001", "25.07.2002")   ' "12" değerini verir.
%>
```

Örnek 68 : DateDiff Fonksiyonu.

2.4.3.15. DateSerial

Açıklama

Bu fonksiyon içerisine gönderilen tarih değerlerini birleştirerek tarih formatında bir tarih değişkeni oluşturur. Geriye dönen değer tarihsel bir değişken olacaktır.

Yazım

```
Deger = DateSerial(Yıl, Ay, Gün)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

   Degisken = DateSerial("1997", "07", "25")   ' "25.07.1997" değerini verir.
%>
```

Örnek 69 : DateSerial Fonksiyonu.

2.4.3.16. DateValue

Açıklama

Bu fonksiyon içerisine gönderilen metinsel tabanlı tarih değerini tarih formatında bir tarih değişkenine çevirir. Geriye dönen değer tarihsel bir değişken olacaktır.

Yazım

```
Deger = DateValue(Metin)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = DateValue("4 November 1979")    ' "04.11.1979" değerini verir.
%>
```

Örnek 70 : DateValue Fonksiyonu.

2.4.3.17. TimeSerial

Açıklama

Bu fonksiyon içerisine gönderilen zaman değerlerini birleştirerek zaman formatında bir zaman değişkeni oluşturur. Geriye dönen değer zaman değişkeni olacaktır.

Yazım

```
Deger = TimeSerial(Saat, Dakika, Saniye)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = TimeSerial("12", "07", "25")    ' "12:07:25" değerini verir.
%>
```

Örnek 71 : TimeSerial Fonksiyonu.

2.4.3.18. TimeValue

Açıklama

Bu fonksiyon içerisine gönderilen metinsel tabanlı zaman değerini zaman formatında bir zaman değişkenine çevirir. Geriye dönen değer zaman değişkeni olacaktır.

Yazım

```
Deger = TimeValue(Metin)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = TimeValue("4:35:32 PM")    ' "04.35.32 PM" değerini verir.
%>
```

Örnek 72 : TimeValue Fonksiyonu.

2.4.4. Test işlemleri

Bazen de değişkenlerimizin hangi tip de olduğunu sınamamız gerekir bu eksiği kapatmak için vbscript içerisinde test fonksiyonları geliştirilmiştir. Bu fonksiyonlardan dönen değer 1 veya 0 yani doğru yada yanlıştır. Bu fonksiyonlar şunlardır.

2.4.4.1. IsArray

Açıklama

Bu fonksiyon içerisine gönderilen değişkenin dizi değişkeni olup olmadığını sınamak için kullanılır.

Yazım

```
Deger = IsArray(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Degisken1

   Degisken = IsArray(Degisken1)   ' Degisken1 dizi ise "true" değerini verir.
%>
```

Örnek 73 : IsArray Fonksiyonu.

2.4.4.2. IsDate

Açıklama

Bu fonksiyon içerisine gönderilen değişkenin tarih değişkeni olup olmadığını sınamak için kullanılır.

Yazım

```
Deger = IsDate(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Degisken1

   Degisken = IsDate(Degisken1)   ' Degisken1 tarih ise "true" değerini verir.
%>
```

Örnek 74 : IsDate Fonksiyonu.

2.4.4.3. IsEmpty

Açıklama

Bu fonksiyon içerisine gönderilen değişkenin bir değere sahip olup olmadığını sınamak için kullanılır.

Yazım

```
Deger = IsEmpty(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Degisken1

   Degisken = IsEmpty(Degisken1)   ' Degisken1 boş ise "true" değerini verir.
%>
```

Örnek 75 : IsEmpty Fonksiyonu.

2.4.4.4. IsNull

Açıklama

Bu fonksiyon içerisine gönderilen değişkenin tanımlanıp tanımlanmadığını sınamak için kullanılır.

Yazım

```
Deger = IsNull(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Degisken1

   Degisken = IsNull(Degisken1)   ' Degisken1 null ise "true" değerini verir.
%>
```

Örnek 76 : IsNull Fonksiyonu.

2.4.4.5. IsNumeric

Açıklama

Bu fonksiyon içerisine gönderilen değişkenin sayısal bir değişken olup olmadığını sınamak için kullanılır.

Yazım

```
Deger = IsNumeric(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Degisken1

   Degisken = IsNumeric(Degisken1)   ' Degisken1 sayısal ise "true" değerini verir.
%>
```

Örnek 77 : IsNumeric Fonksiyonu.

2.4.4.6. ScriptEngine

Açıklama

Bu fonksiyon sunucu üzerinde kullanılan "Script Engine" de kullanılan script dilinin ne olduğunu verir. Sonuç olarak "VbScript" veya "JsScript" değerini verir.

Yazım

```
Deger = ScriptEngine
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Degisken1

   Degisken = ScriptEngine   ' Sunucunun Kullandığı Dili verir.
%>
```

Örnek 78 : ScriptEngine Fonksiyonu.

2.4.4.7. VarType

Açıklama

Bu fonksiyon içerisine gönderilen değişkenin tipini verir. Sonuç olarak sayısal bir değer döner bu değer sabitler konusunda anlatılan sabit değerlerinden birisi olacaktır.

Yazım

```
Deger = VarType(Degisken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken, Degisken1

   Degisken = VarType(Degisken1)   ' degisken1 in ne tip olduğunu verir

  Select Case VarType(Degisken1)
    Case 0 : Response.Write "Empty"
    Case 1 : Response.Write "Null"
    Case 2 : Response.Write "Integer"
    Case 3 : Response.Write "Long Integer"
    Case 4 : Response.Write "Single-Precision Number"
    Case 5 : Response.Write "Double-Precision Number"
    Case 6 : Response.Write "Currency"
    Case 7 : Response.Write "Date"
    Case 8 : Response.Write "String"
    Case 9 : Response.Write "Object"
    Case 10 : Response.Write "Error"
    Case 11 : Response.Write "Boolean"
    Case 12 : Response.Write "Variant"
    Case 13 : Response.Write "Data Object"
    Case 17 : Response.Write "Byte"
    Case 8192 : Response.Write "Variant Array"
   End Select
%>
```

Örnek 79 : VarType Fonksiyonu.

2.4.5. Dönüştürme işlemleri

Bazen de değişkenlerin birbirleri arasında dönüştürülmesi gerekebilir. Bu dönüştürme fonksiyonları şunlardır.

2.4.5.1. Array

Açıklama

Bu fonksiyon içerisine gönderilen değişkenleri statik tek boyutlu bir seri değişken tipine dönüştürür. İçerisine gönderilecek değer sayısına göre boyut büyüyebilir. Unutulmaması gereken en önemli nokta seri değişkenlerin "0" dan başladığıdır. Yani ilk seri değişkeni 0 olacaktır.

Yazım

```
Deger = Array(DeğişkenListesi)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken, A

    Degisken = Array(10, 20, 30, 40)    ' Fonksiyon yardımı ile diziye dönüştürdük.
    A = Degisken(2)                     ' A değeri olarak "30" değeri döner.
%>
```

Örnek 80 : Array Fonksiyonu.

2.4.5.2. CBool

Açıklama

Bu fonksiyon içerisine gönderilen değişkenin değerine göre boolean veri tipinde bir değer döndürür. Bunu yaparken de değişken eğer "0" sa dönüştürülen değer "False" olacaktır. Bunun yanında eğer değişken "0" dan farklı ise o zaman dönüştürülen değer "True" olacaktır.

Yazım

```
Deger = CBool(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = CBool(0)       ' "False" değeri dönecektir.
    Degisken = CBool(434)     ' "True" değeri dönecektir.
%>
```

Örnek 81 : CBool Fonksiyonu.

2.4.5.3. CByte

Açıklama

Fonksiyona gönderilen değişkeni Byte tipi bir değişkene dönüştürür. Byte tipinden kasıt string değildir unutmamak gerekir.

Yazım

```
Deger = CByte(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = CByte(125.567)       ' "126" değeri dönecektir.
%>
```

Örnek 82 : CByte Fonksiyonu.

2.4.5.4. CDate

Açıklama

Fonksiyona gönderilen değişkeni tarih tipi bir değişkene dönüştürür.

Yazım

```
Deger = CDate(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = CDate("4.11.1979")       ' "04.11.1979" değeri dönecektir.
%>
```

Örnek 83 : CDate Fonksiyonu.

2.4.5.5. CInt

Açıklama

Fonksiyona gönderilen değişkeni sayısal veri tipi bir değişkene dönüştürür.

Yazım

```
Deger = CInt(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = CInt("2507")       ' "2507" değeri dönecektir.
%>
```

Örnek 84 : CInt Fonksiyonu.

2.4.5.6. CStr

Açıklama

Fonksiyona gönderilen değişkeni metinsel veri tipi bir değişkene dönüştürür.

Yazım

```
Deger = CStr(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = CStr(2507)       ' "2507" değeri dönecektir.
%>
```

Örnek 85 : CStr Fonksiyonu.

2.4.5.7. TypeName

Açıklama

Fonksiyona gönderilen değişkenin tipini verir. Bu tipler sabitler kısmında tanımlanmış değerlerden oluşmaktadır.

Yazım

```
Deger = TypeName(Değişken)
```

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = TypeName("2507")       ' "String" değeri dönecektir.
    Degisken = TypeName(2507)         ' "Integer" değeri dönecektir.
    Degisken = TypeName(37.5)         ' "Double" değeri dönecektir.
%>
```

Örnek 86 : TypeName Fonksiyonu.

2.4.6. Biçimlendirme işlemleri

Bazen de değişkenleri biçimlendirmek gerekebilir. Bu biçimlendirme işlemleri için aşağıda verilen bir dizi fonksiyon kullanılır.

2.4.6.1. FormatNumber

Açıklama

Fonksiyona gönderilen numerik değişkeni biçimlendirmek için kullanılır. Bu biçimlendirme işlemi sunucunun bölgesel ayarlarına göre yapılabildiği gibi verilen opsiyonlara göre de yapılabilmektedir.

Yazım

```
Deger = FormatNumber(Değişken, [Ondalık Kısım], [Öncü Değer], [Mutlak Değer], [Binlik Grup Ayırma])
```

Yazım içerisinde kullanılan "[" ve "]" işaretleri içerisinde olan kısımlar zorunlu olmayan opsiyonlardır. Bu opsiyonlar belirtilmediği zaman sistem otomatik olarak sunucunun geçerli değerlerini alacaktır. Bu opsiyonların açıklamaları şu şekildedir.

Ondalık Kısım : Bu seçeneğe verilen numerik değer kadar ondalık kısım basamağı oluşturacaktır. Yani virgülden sonra kaç basamağın gösterileceği ayarlanabilmektedir. Unutulmaması gereken en önemli konu virgülden sonraki ondalık değerin burada atanan basamak sayısına yuvarlanmasıdır. Örneğin "1,2506943543" değerinin "4" haneli bir ondalık kısmına yuvarlandığı zaman "1,2507" şeklinde bir biçim alacaktır.

Öncü Değer : Bu seçeneği aktif hale getirdiğimiz zaman sıfırdan küçük ondalık değerler için sayının öncü değerini aktif hale getirecektir. Örneğin ",44" değerini bu seçeneği aktif hale getirdiğimiz zaman "0,44" şeklinde görüntülenecektir. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Mutlak Değer : Bu seçeneği aktif hale getirdiğimiz zaman fonksiyona gönderilen negatif değerin pozitif değerini verecektir. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Binlik Grup Ayırma : Bu seçenek aktif hale getirildiği zaman sayıyı sağdan başlamak üzere binlik gruplara ayıracaktır. Örneğin "1000000" değerini binlik grupladığımız zaman "1.000.000" şeklinde gruplayacaktır. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = FormatNumber(2507)              ' "2.507,00" değeri dönecektir.
    Degisken = FormatNumber(2507,4)            ' "2.507,0000" değeri dönecektir.
    Degisken = FormatNumber(.2507,4,-1)        ' "0.2507" değeri dönecektir.
    Degisken = FormatNumber(-2507,2,0,-1)      ' "2.507,00" değeri dönecektir.
    Degisken = FormatNumber(15.678,2,-1,-1)    ' "15.68" değeri dönecektir.
%>
```

Örnek 87 : FormatNumber Fonksiyonu.

2.4.6.2. FormatDateTime

Açıklama

Fonksiyona gönderilen tarihsel değişkeni biçimlendirmek için kullanılır. Bu biçimlendirme işlemi sunucunun bölgesel ayarlarına göre yapılabildiği gibi verilen opsiyonlara göre de yapılabilmektedir.

Yazım

```
Deger = FormatDateTime(Değişken, [Tarih, Zaman Formatı])
```

Yazım içerisinde kullanılan "[" ve "]" işaretleri içerisinde olan kısımlar zorunlu olmayan opsiyonlardır. Bu opsiyonlar belirtilmediği zaman sistem otomatik olarak sunucunun geçerli değerlerini alacaktır. Bu opsiyonların açıklamaları şu şekildedir.

Tarih Zaman Formatı : Bu seçeneğe verilen sabit (sabitler kısmında verilmiştir) tarih veya zamanın nasıl biçimlenmesi gerektiğini belirtir.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = FormatDateTime("25/7/97")        ' "25/07/1997" değeri dönecektir.
    Degisken = FormatDateTime("15:34")          ' "3:34:00 PM" değeri dönecektir.
    Degisken = FormatDateTime("3:34:00 PM", 4) ' "3:34" değeri dönecektir.
%>
```

Örnek 88 : FormatDateTime Fonksiyonu.

2.4.6.3. FormatCurrency

Açıklama

Fonksiyona gönderilen para değeri içeren değişkeni biçimlendirmek için kullanılır. Bu biçimlendirme işlemi sunucunun bölgesel ayarlarına göre yapılabildiği gibi verilen opsiyonlara göre de yapılabilmektedir.

Yazım

```
Deger = FormatCurrency(Değişken, [Ondalık Kısım], [Öncü Değer], [Mutlak Değer], [Binlik Grup Ayırma])
```

Yazım içerisinde kullanılan "[" ve "]" işaretleri içerisinde olan kısımlar zorunlu olmayan opsiyonlardır. Bu opsiyonlar belirtilmediği zaman sistem otomatik olarak sunucunun geçerli değerlerini alacaktır. Bu opsiyonların açıklamaları şu şekildedir.

Ondalık Kısım : Bu seçeneğe verilen numerik değer kadar ondalık kısım basamağı oluşturacaktır. Yani virgülden sonra kaç basamağın gösterileceği ayarlanabilmektedir. Unutulmaması gereken en önemli konu virgülden sonraki ondalık değerin burada atanan basamak sayısına yuvarlanmasıdır. Örneğin "1,2506943543" değerinin "4" haneli bir ondalık kısmına yuvarlandığı zaman "1,2507 TL" şeklinde bir biçim alacaktır.

Öncü Değer : Bu seçeneği aktif hale getirdiğimiz zaman sıfırdan küçük ondalık değerler için sayının öncü değerini aktif hale getirecektir. Örneğin ",44" değerini bu seçeneği aktif hale getirdiğimiz zaman "0,44 TL" şeklinde görüntülenecektir. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Mutlak Değer : Bu seçeneği aktif hale getirdiğimiz zaman fonksiyona gönderilen negatif değerin pozitif değerini verecektir. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Binlik Grup Ayırma : Bu seçenek aktif hale getirildiği zaman sayıyı sağdan başlamak üzere binlik gruplara ayıracaktır. Örneğin "1000000" değerini binlik grupladığımız zaman "1.000.000 TL" şeklinde gruplayacaktır. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Örnekler

```
<%
  Option Explicit
  Dim Degisken

    Degisken = FormatCurrency(2507)       ' "2.507,00 TL" değeri dönecektir.
    Degisken = FormatCurrency(2507,4)     ' "2.507,0000 TL" değeri dönecektir.
    Degisken = FormatCurrency(2507,4,-1)  ' "0,2507 TL" değeri dönecektir.
%>
```

Örnek 89 : FormatCurrency Fonksiyonu.

2.4.6.4. FormatPercent

Açıklama

Fonksiyona gönderilen para değeri içeren değişkeni biçimlendirmek için kullanılır. Bu biçimlendirme işlemi sunucunun bölgesel ayarlarına göre yapılabildiği gibi verilen opsiyonlara göre de yapılabilmektedir.

Yazım

```
Deger = FormatCurrency(Değişken, [Ondalık Kısım], [Öncü Değer], [Mutlak Değer], [Binlik Grup Ayırma])
```

Yazım içerisinde kullanılan "[" ve "]" işaretleri içerisinde olan kısımlar zorunlu olmayan opsiyonlardır. Bu opsiyonlar belirtilmediği zaman sistem otomatik olarak sunucunun geçerli değerlerini alacaktır. Bu opsiyonların açıklamaları şu şekildedir.

Ondalık Kısım : Bu seçeneğe verilen numerik değer kadar ondalık kısım basamağı oluşturacaktır. Yani virgülden sonra kaç basamağın gösterileceği ayarlanabilmektedir. Unutulmaması gereken en önemli konu virgülden sonraki ondalık değerin burada atanan basamak sayısına yuvarlanmasıdır. Örneğin "1,2506943543" değerinin "4" haneli bir ondalık kısmına yuvarlandığı zaman "%125,07" şeklinde bir biçim alacaktır.

Öncü Değer : Bu seçeneği aktif hale getirdiğimiz zaman sıfırdan küçük ondalık değerler için sayının öncü değerini aktif hale getirecektir. Örneğin ",44" değerini bu seçeneği aktif hale getirdiğimiz zaman "% 0,44" şeklinde görüntülenecektir. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Mutlak Değer : Bu seçeneği aktif hale getirdiğimiz zaman fonksiyona gönderilen negatif değerin pozitif değerini verecektir. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Binlik Grup Ayırma : Bu seçenek aktif hale getirildiği zaman sayıyı sağdan başlamak üzere binlik gruplara ayıracaktır. Örneğin "1000" değerini binlik grupladığımız zaman "%1.000" şeklinde gruplayacaktır. Bu seçeneği aktif hale getirmek için sabitler kısmında anlatılan "durum sabitleri" kullanılmalıdır, bu sabitlerin yerine, pasifleştirmek için "0", aktifleştirmek için "-1" değeri girilebilir.

Örnekler

```
<%
  Option Explicit
  Dim Degisken
```

```
    Degisken = FormatPercent(.25)           ' "%25" değeri dönecektir.
    Degisken = FormatPercent(.2507)         ' "%25" değeri dönecektir.
    Degisken = FormatPercent(.2507,4,-1)  ' "%0,2507" değeri dönecektir.
%>
```

Örnek 90 : FormatPercent Fonksiyonu.

2.5.Mantık kontrolleri :

İster ASP diliyle, ister başka programlama diliyle yazılmış olsun, bilgisayar programlarının varlık sebebi, çeşitli durumları değerlendirmek ve kararlar verip bu doğrultuda işlemine devam etmektir. Bunu programlarda mantık kontrol komutları ile yaparız. Program bu öğeler sayesinde karşılaştırmalar yapar, belirli koşulların yerine gelip gelmediğine bakar veya belirli bir durumun oluşuna veya sona erişine bağlı olarak başka bir işlemi başlatır veya bitirir. Kimi zaman da, programa yapmakta olduğu işi durdurarak, başka bir iş yapmasını bildirebiliriz. Bunlara süreçler veya prosedürler denir.

ASP programlarında kara mekanizmasının en temel kontrol öğesi "eğer...se...yap!" şeklindedir.

2.5.1. If – Then – Else

Vereceğiniz bir durumun oluşup oluşmadığını sınar.

```
<%
  if şart then
    [şart doğru ise yapılacak işlemler]
  else
    [şart yanlış ise yapılacak işlemler]
  end if
%>
```

Örnek 91 : İf Blok Diyagramı.

Bunu bir örnekle ifade edelim; Programımız eğer saat 12'den önce çalıştırılırsa sayfada "Günaydın"; eğer saat 12'den sonra çalıştırılırsa sayfa da "Tünaydın" yazısının yazdırılacağı bir program yazalım.

```
<%
  if hour(now) < 12 then
    response.write "Günaydın!"
  else
    response.write "Tünaydın!"
  end if

    response.write "Sayfamıza hoş geldiniz"
%>
```

Örnek 92 : İf Yapısı.

Program çalıştırıldığı zaman, çalıştırdığınız saate göre programdaki selam tarzının değiştiğini göreceksiniz. Programın nasıl çalıştığına gelirsek. Önce ki sayılarda asp içerisinde kullanmaya hazır mevcut fonksiyonların varlığından söz etmiştik bunlardan biriside o an ki saati ve tarihi bildiren now() fonksiyonudur. (hour() fonksiyonu ise içindeki tarih ve saatin yalnızca saat basamağını gösteren bir fonksiyondur) fonksiyondan dönen değer eğer 12'den küçükse, programımız response (karşılık) nesnesinin write

(yazdır) methodunu kullanarak (nesne ve method konularına fazla takılmayın, ileri sayılarda anlatılacaktır) ziyaretçinin browser penceresine "Günaydın" yazdırır.

Eğer bu ilk karşılaştırmanın sonucu doğru değil ise yani saat 12 den büyük ise, doğru ise yapılacaklar alanındaki komutlar pas geçilerek yanlış ise yapılacaklar alanındaki komutlar işleme sokulur. Bu işlem sonucunda "Tünaydın" kelimesi çıktı olacaktır.

Fakat burada bir eksik var: programı saat 18'den sonra çalıştıranlara "iyi akşamlar" şeklinde bir karşılama yazmamız daha doğru olmaz mı? If döngüsü kendi içerisinde sınırsız "elseif" imkanı vererek bize bunu sağlar. Her "elseif" yeni bir "if" gibi işlem yapar.

```
<%
  if şart1 then
    [şart1 doğru ise yapılacak işlemler]
  elseif şart2 then
    [şart2 doğru ise yapılacak işlemler]
  .
  .
  else
    [şartlar yanlış ise yapılacak işlemler]
  end if
%>
```

Örnek 93 : Elseif Fonksiyonu.

Yeni öğrendiğimiz bu komutu kullanarak karşılama sistemimizi biraz genişletelim.

```
<%
  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 "Sayfamıza hoş geldiniz"
%>
```

Örnek 94 : Elseif Yapısı.

2.5.2. 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
    Durum1 : yapılacak işler
    Durum2 : yapılacak işler
    Durum3 : yapılacak işler
  Seçmeyi bitir
%>
```

Örnek 95 : Select Case Fonksiyonu.

Vbscript, verdiğiniz durum listesine veya içinde çeşitli değerler bulunan değişkene bakarak, bu değişkeni bir "durum" sayacak ve verdiğimiz durumlardan hangisini tutuyorsa, ona ait komut dizisini çalıştıracaktır. Yukarıdaki örneğimizi bu kez bu yapıyı kullanarak yazalım.

```
<%
  select case Hour(Now)
  case 5,6,7,8,9,10,11
    response.write "Günaydın!"
  case 12,13,14,15,16,17,18
    response.write "Tünaydın"
  case 19,20,21,22
    response.write "İyi akşamlar"
  case else
    response.write "İyi geceler"
  end select

  response.write "Sayfamıza hoş geldiniz"
%>
```

Örnek 96 : Select Case Yapısı.

"Select case" komutuna, içindeki değerleri "durum" sayacağı dizi veya değişken olarak vbscript'in kullanmaya hazır fonksiyonlarından Hour(Now)'ı veriyoruz. Bu fonksiyonlardan, 0 ile 24 arasında bir değer dönecektir. Bu değer "select case" için durum demektir. Select case bu değer ile altta sıralanan case'leri karşılaştıracak ve elindeki değer hangi case'i tutuyorsa ona ait komutlar çalıştırılacaktır. Sonuncu case'e lütfen dikkat edin: burada case olarak else (başka) veriliyor. Bu bizi 22 den 5 e kadar olan saatleri sıralamaktan kurtarır. 22 – 5 arasında kullanıcılarımıza "iyi geceler" diyebiliriz.

2.6.Döngüler

Mantık kontrolleri bir programın akışını kontrol için kullanacağımız birinci önemli unsur ise, döngü de ikinci en önemli unsur sayılır. Hatta programcının tembellik katsayısına göre belki de birinci en önemli unsur bile sayılabilir! Çünkü döngü (loop) programa, bir işi bitesiye kadar yaptırma imkanı verir. Tabii bu iş sonsuza kadar sürecek olursa, buna "endles loop" (sonsuz döngü) denir. Vbscript'te kullanabileceğiniz döngü yöntemleri şunlardır.

2.6.1. For-Next döngüsü

Programın bir işi belirli kere yapmasını istiyorsak, ona yapacağı işi bir sayaç değişkeni ile birlikte, "for" döngüsüyle bildiririz.

```
<%
  for sayac = baslangic to son step adim
    yapılacak işler
  Next
%>
```

Örnek 97 : For - Next Yapısı.

Burada, "sayaç" yerine istediğimiz bir değişken adını, "başlangıç" yerine sayacın başlatılmasını istediğimiz sayıyı, "son" yerine sayacın durmasını istediğimiz sayıyı, ve "adım" yerine, sayacın kaçar kaçar artmasını istediğimizi yazabilirsiniz. En sondaki "next" deyimi ise döngünün bir sonraki adıma geçmesini sağlar. Bu adımda sayaç, "step" kelimesi varsa karşısındaki değer kadar artırılır ve yapılacak işler yeniden yapılır.

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

  for sayac = 0 to 6
    response.write gunler(sayac)
  next
%>
```

Örnek 98 : For – Next Örneği.

Bu asp kodunda, gunler adıyla bir dizi-değişken oluşturuyoruz ve bu değişkenin 7 hanesine, günlerin adlarını atıyoruz. Sonra sayac adlı sayacı 0 dan 6'ya kadar arttırıyoruz (sayacın birer birer artmasını istersek step komutunu yazmayız). Şimdi kendimizi bir an için vbscript yerine koyalım ve birinci adımda yaptığımız işi düşünelim: "hımm.. programcı bey, benim sayac'ı önce 0 yapmamı istiyor; peki sayac 0 olsun. Sonra günler dizi-değişkeninden sayac değeri ile aynı sayıyı taşıyan değişkeni alıp bunu ekrana yazdırmamı istiyor. Peki, gunler(0) ne imiş, bakalım. gunler(0) Pazartesi imiş. O halde ziyaretçinin ekranına bir Pazartesi kelimesi yazalım. Şimdi sırada next var. Yani sonraki adıma devam edeceğiz, step değeri olmadığına göre sayacı bir artırayım. Sayaç böylece 1 oldu"

Ve böylece vbscript, sayacın son değeri olan 6'ya ulaşıncaya kadar günler dizi değişkeninden sayacın değerine göre değer seçerek ve bunu ekrana yazdırarak, işini yapacaktır. Bu bakımdan vbscript güvenilir ve çalışkan bir arkadaştır.

Unutulmaması gereken bir diğer noktada diğer programlama dillerinde olduğu gibi "next" komutunun "next a" gibi yani bir değişkenle beraber kullanılmamasıdır. Vbscript de bu tip bir kullanım hata verecektir.

2.6.2. While-wend döngüsü

Bazen program içerisinde kullanacağımız döngü sabit olmayabilir, yani programın üst kesimlerinden veya kullanıcıdan alınan bir değer kadad döngü kurulması istenebilir. Özetle yapılmasını gereken işin ancak sayaç bir değerden azsa, çoksa ve eşitse yapılmasını, bu durum değişirse durmasını isteyebiliriz. Bunu while (..iken) komutuyla yapabiliriz. While döngüsü kullandığımız zaman sayacı bizim artırmamız gerekir.

Sözgelimi, yukarıdaki programın 7 günün tümünü ekrana yazdırması değil de, mesela gün sayısı 5'den küçük ise yazdırmasını istiyor olabiliriz. Bu durumda kodumuzda for-next arasındaki bölümde şu değişikliği yapmalıyız.

```
while sayac <= 5
   response.write gunler(sayac)
   sayac = sayac + 1
Wend
```

Örnek 99 : While-Wend Yapısı.

Burada while döngüsünün wend kelimesi ile sonlandığına dikkat edin. While satırındaki sayacı değiştirdik, programın sayaç 5'den küçük veya 5'e eşit ilen çalışmasını sağladık. For'dan farklı bir diğer ifade ise sayacı artıran "sayac = sayac + 1" ifadesidir. Bu ifade ilk bakışta garip görünebilir. Fakat bilgisayar açısından bu "sayacın o anki değerini al, 1 ile topla ve bulduğun yeni değeri sayacın mevcut değerinin üzerine yaz" demektir.

2.6.3. Do-Loop döngüsü

Bazende bir döngü içerisinde bir durumun oluşması halinde döngünün sonlandırılması istenebilir. Böyle durumlarda do-loop döngüsü kullanılabilir. Kullanılması diğer döngülerle hemen hemen aynıdır. Türkçe anlatımı "şart sağlanana kadar döngüyü devam ettir" şeklinde olacaktır.

```
Do while Şart
  ....
  ....
  ....
[Exit do]
  ...
  ...
  ...
Loop
```

Örnek 100 : Do - Loop Fonksiyonu.

Döngü sırasında başka bir şartın sağlanması durumunda ise exit do ifadesi ile do döngüsü sonlandırılabilmektedir.

Do-loop ile while-wend arasında bir fark görememiş olabilirsiniz ama ikisi arasında çok ufak bi fark vardır. Do-loop da döngüye girildikten sonra şart kontrol edilir, while-wend de ise döngüye girmeden önce şart kontrol edilir.

2.7.Diğer faydalı komutlar

2.7.1. With – End With

Yukarıda belirtilen ve açıklanan komutların yanında bir takım kullanışlı komut da yer almaktadır. Bunlardan ilki "with" komutudur. Bu komut yardımı ile bir objenin birden fazla komut için kullanılması istenirse işimize yarayacaktır. Bu komut sayeside "with" komutuna atayacağız obje, with bloğu içerisinde adı geçmeden de kullanılabilmektedir. Örneğin:

```
With Obje
  .metod = xx
End With
```

Örnek 101 : With Kullanımı.

Örnekte de görüleceği gibi with bloğu içerisinde obje adı vermeden sadece "." Yazılır ve objeye ait özellik veya metod kullanılabilir. Böylece birçok kere aynı obje adını yazmaya gerek kalmaz.

2.8.Script Performansı

Genelde web sunucuları dağıtılmış işlemci yapısında çalışır. Örnek vermek gerekirse 20 işlemin aynı anda çalıştığı bir sunucu düşünelim bu işlemlerin hepsi sırayla bir miktar işletilir. Bu sıra işlemcinin saat hızı ile doğru orantılıdır (1.4 GHz işlemci saniyenin 1/ 1468006,4 si bir sürede işletmektedir.). bu sıra ile işletme sırasında işlemlerden birisi takıntıya uğrarsa diğerleri de dolaylı olarak bu gecikmeden etkilenir. Bu sebeple ASP içerisinde işlemleri aza indirmek bu performansı yükseltecektir. ASP programcıları arasında yaygın olarak şu hatalar (hata demek belki yanlış olacaktır, sadece bu tip kullanımlar performans düşüklüğüne yol açacaktır ki bu bazı uygulamalarda yanlış işe aynı anlamdadır.) yapılmaktadır.

Aynı sayfa içerisinde birden fazla script dili kullanmak: bu şekilde bir kullanımca aynı sayfa içerisinde birden fazla script engine çalıştırılacağı için verimde ciddi düşüş meydana gelecektir.

Gereksiz yere değişken ataması yapmak: bu programlama aksaklığı yüzünden hafızada gerektiğinden fazla yer atanacak ve uygulama yavaşlayacaktır. Mümkün olduğu kadar az değişkenden kastım özellikle yazdırılacak bir değerin ilk olarak bir değişkene aktarılması daha sonra bu değişkenin ekrana yazdırılmasıdır. Bunun yerine direkt olarak değeri yazdırmak daha performanslıdır.

Çok fazla işlem gerektiren fonksiyonlar için component tasarlanması : sunucu içerisinde EXE dosyalarının derlenmesi ASP dosyalarının derlenmesinden çok daha hızlıdır. Bu sebeple işlemleri componentlere (exe veya dll dosyasına) yaptırmak işlemciyi rahatlatacaktır.

2.9.Nesne Kavramı

Günlük hayat da olduğu gibi proglamlamada da işlemlerimizi değişik nesnelere yaptırırız. Örneğin birşey kesmek istediğimizde bıçak kullanırız, su içmek istediğimizde bardak kullanırız vs. Kullandığımız her nesnenin ise belirli özellikleri vardır. Örneğin kesme işlevini gerçekleştiren bıçak keskindir. Yada su içeceğimiz bardak suyun dökülmesini engelliyecek şekilde kapalı bir cisimdir.

ASP tekniğinde işlemlerimizi gerçek hayat da olduğu gibi nesnelere yaptırtırız. Nesne yönelimli programlama kavramı (Object Oriented Programming OOP) bu şekilde ortaya çıkmıştır. Diyelim ki öğretmensiniz ve ASP programınızda her öğrencinin notunu veritabanına işleyen, veritabanından notları alarak geçeni kalanı belirleyen veya öğrencilerle ilgili yapılması gereken birçok işi yapan programcıklarımız mevcut; bu programcıkların kullandığı birçok değişkenimiz var: demek ki sizin ASP programınızda "öğrenci" diye bir nesneniz var. Ve siz bu nesneye yönelimli program yazmışsınız.

Her program nesnesi iki unsura sahiptir.

Özellik (property) : bir nesnenin özellikleri, onun değişkenleridir. "Öğrenci" nesnesinin "Öğrenci Adı", "Notlar", "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 kullandığımız script dilinin getirdiği nesneler vardır. ASP programınızı javascript ile yazarsanız başka, vbscript ile yazarsanız başka nesnelere sahip olursunuz; ancak her ikisinde de ortak olan "scripting" nesneleri vardır. Bunları birazdan ayrıntılı olarak ele alacağız. Sonra web serverin size hazır sunduğu nesneler vardır. Bunları daha sonraki sayılarda göstereceğiz. Ve tabii, browser'ın bir HTML sayfasının bölümlerini nesne sayarak oluşturduğu nesneler vardır. Bunları da diğer nesneleri ele alırken sırası geldikçe değineceğiz. (Tabii bir de ASP programınızı javascript ile yazarsanız vbscript'ten farklı olarak kendisi nesne-yönelimli bir dil olan javascript'in oluşturmanıza imkan veren nesneleri vardır. Fakat bu nesneler bizim konumuz dışında kaldığı için incelenmeyecektir.)

Nesneler nasıl olmuş olursa olsunlar, daima size bir değer verirler.

Nesne.Özellik = Değer

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

İf Nesne.Özellik > Değer Then

Nesnelerin özelliklerinin değerini değişkenlere atayabiliriz ancak bunu yaparken nesnenin bir metoduna (fonksiyonu) göndermede bulunmamız ve gerekiyorsa bu fonksiyona kullanması için veri göndermeliyiz. (bir fonksiyona kullanması için gönderilen değere argüman/argument denir);

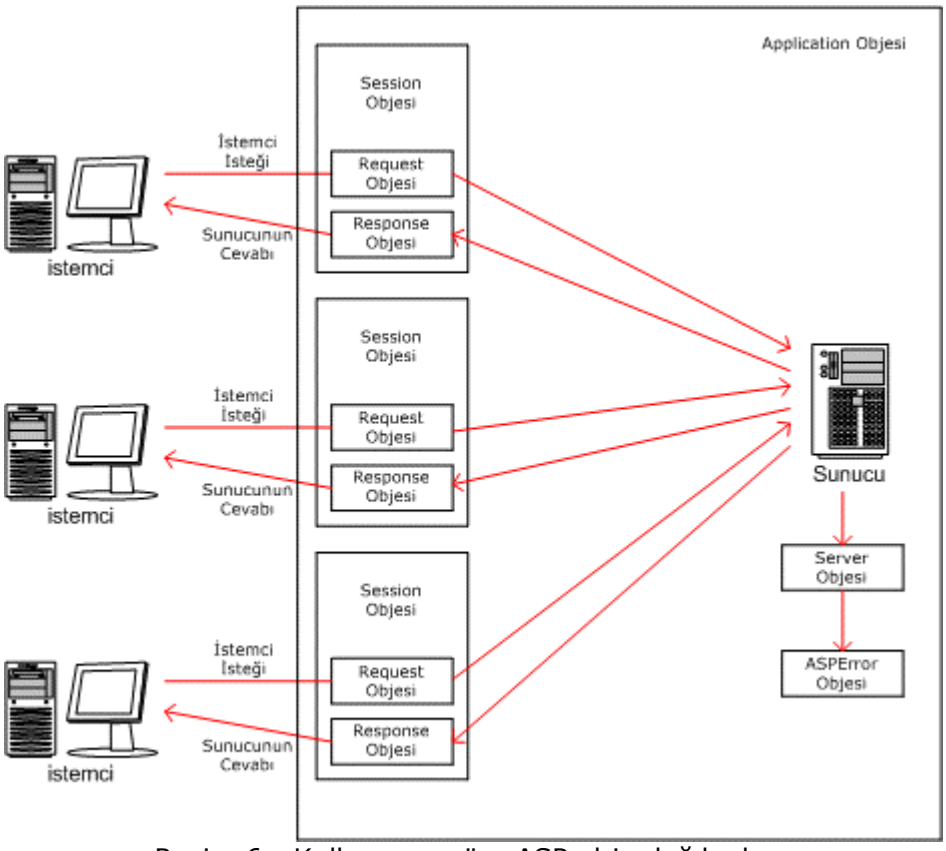
Değişken = Nesne.Metod (argüman1,argüman2..)

Tarayıcı nesne modelinin tam aksine ASP nesnelerinde bir hiyerarşi söz konusu değildir. Nesne modeli, bize bir sunucu nesnesini metodları ve özellikleri ile birlikte, scriptler arasında kullanabilmemizi, sağlar. Bir ASP uygulaması geliştirebilmek için server (sunucu), application (uygulama), session (oturum), request (istek) ve (response) yanıt nesneleri kullanılır.

2.9.1. Uygulama ve Oturumlar

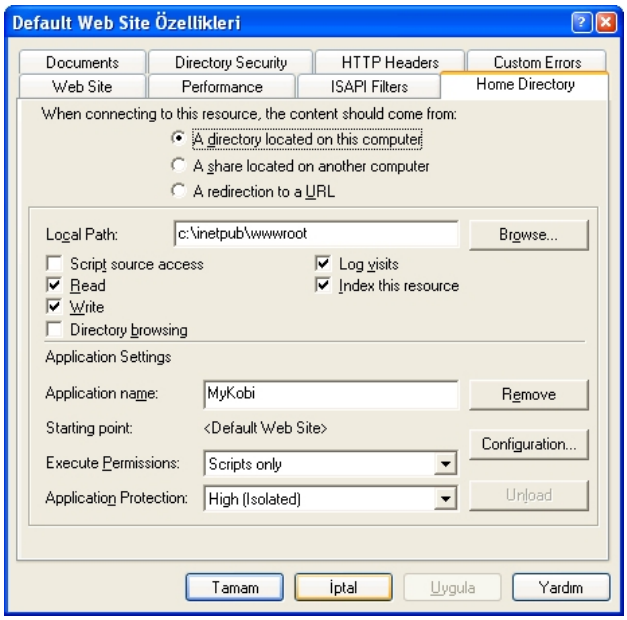
Aplication (uygulama) nesnesi bir ASP uygulamasının tamamını simgelendirir. Session (oturum) nesnesi ise, her bir kullanıcının uygulamadan istediği sayfaları simgelendirmektedir. Bu durum "**Error! Reference source not found.**)" de açıklanmıştır. Her istemci için ayrı ayrı oluşturulan "session" nesnesi yine her kullanıcı için oluşan request ve response nesneleri ile iletişim sağlanır. Bütün bunları kapsayan "application" nesnesi ise tüm kullanıcı oturumlarını kapsar şekilde oluşacaktır. Uygulama objesi siteden herhangi bir sayfa istendiği an otomatik olarak oluşacaktır fakat oturum objesi sadece isteyen kullanıcı için oluşacaktır.

Bir oturum, bir request nesnesinin HTTP iletişimini başlatır, bir web gezgininden, bir sunucuya gönderilir ve sunucudan response nesnesi ile geri gönderilen sonuç bilgisi ile sonlanır.



Resim 6 : Kullanıcıya göre ASP obje dağılımları.

Hafızada oluşturulan kullanıcıya veya aplikasyona ilişkin objeler içinde herhangi bir terslik olması halinde bu diğer çalıştırılan objeleri de etkileyecektir. Bunu önlemek için Microsoft firması "IIS" üzerine bir opsiyon koyarak her sitenin kendisine ait bir bellek alanında çalışmasını olağan kılmıştır. Bu seçenek normalde kapalı haldedir yani sunucudaki tüm objeler tek bir bellek alanında çalışacaktır. Bu özelliği aktif hale getirmek için "IIS" üzerinden, genel bellek alanından ayırmak istediğimiz sitenin özelliklerine girilir. Açılan özellik alanında en üstteki sekmeleri kullanarak "Home Directory" kısmına gelinir (**Error! Reference source not found.**).



Resim 7 : IIS Site Özellikleri.

Bu alanda yer alan "application protection" yani "uygulama koruma" alanında üç maddeden oluşan seçenek kısmı görülmektedir bu seçeneklerden bizim için hangisi uygunsa onu seçeriz ve okleyip işlemi sonlandırabiliriz. Bu seçeneklerin neler olduğunu isterseniz teker teker inceleyelim.

Low (IIS Process) : Bu seçenek aktif olduğu zaman tüm asp uygulamaları ve nesneler tek bir hafıza alanında ortak çalışıyor olacaktır. Bu hafıza alanı web sunucunun kendi işlemleri ile aynı yerde tutulacaktır ve "Windows Görev Yöneticisi" nde "Inetinfo.exe" dosyası olarak gözükecektir. Ve bu uygulamanın sonlandırılması halinde sunucuda yer alan ve ortak çalışan uygulamalara ait bilgiler (çalışma ve oturum bilgileri) silinecektir. Bu seçenek aktif olduğu zaman sunucu bir risk altında olacaktır çünkü tüm uygulamaların tek bir alanda çalışması ki özellikle bu alanın web sunucu çalışma alanı olduğu düşünülürse, uygulamaların birinde hata olsa bile diğerlerini etkileyeceği için; ayrıca herhangi bir hatanın web sunucuyu şişirip çalışmaz hale getireceği için uygunsuz olacaktır. Mümkün oldukça seçili olmamasına dikkat ediniz.

Medium (Pooled) : Bu seçeneğin aktif olması halinde tüm asp uygulamaları ve nesneleri tek bir hafıza alanında çalışacaktır fakat bu alan web sunucunun işlemlerinin yapıldığı alandan farklı olacaktır. Bu sebepten dolayı sitede herhangi bir aksaklı olsa bile bu sadece siteleri etkileyecektir. Sunucunun şişmesini ve çalışmaz hale gelmesini engelleyecektir. Bu web sunucudan bağımsız yerde işletilen uygulamalar "Windows Görev Yöneticisi" nde "DLLHost.exe" dosyası olarak gözlenecektir.

High (Isolated) : Bu seçenek aktif olduğu zaman her sitenin uygulamaları ve objeleri kendisine ait bir hafıza alanında çalıştırılmasına olanak verecektir. Böylece sitelerde oluşacak aksaklıklardan sadece kendisi sorumlu olacaktır ne web sunucuyu nede diğer siteleri aksatabilecektir. Bu hafıza alanı bir önceki seçenekte olduğu gibi "Windows Görev Yöneticisi" nde "DLLHost.exe" olarak gözlemlenecektir fakat tek fark her site için ayrı ayrı bir "DLLHost.exe" olmasıdır. Seçenekler arasında en sağlıklı olanı budur fakat sunucunun sistem kaynaklarının yeterli olması gerekmektedir.

2.10. Request Nesnesi

Request nesnesi bir bilgi alma nesnesidir. Bu nesne yardımı ile yazdığımız scriptler dışarıdan veya bir önceki sayfadan bilgi almakta kullanılır. Bünyesinde beş adet ana fonksiyon bulundurmaktadır. Bunlar;

1. Cookie
2. ServerVariables
3. Form
4. QueryString
5. TotalBytes

2.10.1. Veri Transferi

İstemci tarafında, request nesnesine bilgi depolanır ve sunucuya HTTP belge isteğinin bir parçası olarak gönderilir. Sunucu bu bilgiyi dekod eder ve ASP boyunca kullanılabilir hale getirir. Sunucuya bilgi göndermenin iki yöntemi vardır; birincisi sayfadaki <form> kısmını kullanmak ve ikinci olarak da URL'nin sonuna bir sorgu stringi gibi eklenerek doğrudan sağlanabilir.

Bu veriyi bir ASP scriptine göndermek için HTML içinde form oluşturmak ile mümkün olmaktadır. Eğer formun metod özelliği "GET" şeklinde ise verimiz URL sonuna ekli olarak gönderilmektedir. Ama formun metod özelliği "POST" şeklinde ise verimiz HTTP belgesinin bir parçası olarak gizli şekilde gönderilmektedir.

Gerek form içerisinde gerekse URL sonunda string olarak gönderilen verileri ASP içerisinde kullanmak için veriyi (server tarafından dekod edilmiş bilgiyi) bir değişkene aktarmak gerekmektedir.

Bunu ASP içinde 2 farklı şekilde yapmaktayız. Bu farklılık verinin gönderiliş tarzından ileri gelmektedir. Eğer verimiz URL sonunda bir sorgu stringi şeklinde gönderilmiş ise veriyi:

```
<%
  isim = request.querystring("veri")
%>
```

Örnek 102 : QueryString ile Veri Alımı.

Şeklinde alabilmekte ve asp içerisinde kullanabilmekteyiz. Bu kod içinde "isim" değişkenine URL sonuna ek veri olarak gönderilen "veri" datasını verdik. Nesne metodunun isminden de anlaşılacağı gibi "querystring" yani sorgu cümleciği şeklinde bir veri transferi yapılmıştır. Bu data url sonunda şu şekilde gözükmektedir.

Test.asp?veri="M. Günce"

Yukarıda ki örnekdende görüleceği gibi "M. Günce" verimizi test.asp sayfasına URL sonuna eklenmiş olarak gönderdik. Ve bunu yaparken ara değişken olarak "veri"'yi kullandık. (unutmadan belirtmekte fayda gördüğüm bir nokta da URL sonuna eklenmiş olarak gönderilen verileri form kullanmadan da göndermek mümkün olmaktadır. Bu şekilde form olmayan scriptlerimizde de sayfalar arası veri transferi yapmamız olağan hale gelmiştir.)

Bu veri transferini yapmanın bir diğer yolu da gönderen form özelliğini "POST" yaparak mümkün olmaktadır. Bu özellik sayesinde verilerimiz URL satırında gözükmeden alıcı tarafından kullanılabilmektedir.

Eğer verimiz form içerisinde "POST" şeklinde gönderilmiş ise bunu ASP içerisinde bir değişkene şu şekilde verebiliriz.

```
<%
  isim = request.form("veri")
%>
```

Örnek 103 : Form ile Veri Alımı.

Yukarıdaki örneğin aynısını bu defa form içerisinde gizli olarak gönderdik. Tek farkımız URL satırında veri gözükmemesidir (bu bilgi sunucuya gönderilen URL isteğinin içerisinde gömülü olarak, başlık olarak gönderilmiştir) yani;

Test.asp

Şeklinde bir URL görmemizdir. Bu özellik sayesinde hem URL satırını gereksiz doldurmamış hem de verilerimize gizlilik kazandırmış olmaktayız. (Not: bu veri transferi tipini sadece form kullanarak veri gönderdiğimiz zaman kullanabilmekteyiz.)

Bir request nesnesini hiçbir metod kullanmadan çağırmamız halinde; yani

Isim = request("veri")

Şeklinde bir kullanım olduğu zaman ASP bu nesnenin metodlarını bir sıra içerisinde dener. Bu sıra

-QueryString
-Form
-Cookie
-ServerVariables

şeklindedir. Bu şekilde kullanmanız size kodlamada zaman ve hız kazandırabilir. Fakat sistem kaynaklarından kullanarak birkaç denemede veriyi alacağı için performansta bir düşme yapabilecektir.

Bir formdan veri isteğinde bulunurken dikkat etmeniz gereken bir diğer nokta da veriyi alacağınız HTML elemanının tipidir. Şimdiye kadar anlatılan veri alımı sadece normal textbox için geçerli idi fakat diğer öğeler için gelen veriler farklı olacaktır.

Bir formdan gönderilen verinin hepsini teker teker aldırmaktansa bir döngü içerisinde bu işlemi kolayca yapabiliriz. Bu döngü aşağıdaki gibi olacaktır;

```
<%
For Each objitem in Request.Form
  İf request.form(objitem).count > 0 then
    For intLoop = 1 to Request.Form(objitem).count
    Objitem = request.form(objitem)
  End if
Next
%>
```

Örnek 104 : Formdaki Tüm Veriyi Alma.

Yukarıdaki örnekte bir önceki formdan gönderilen değiskenlerin değerleri kendi isminde bir değişkene atanmiş oldu. Bu sekilde bir kullanımla onlarca satır kod yazmaktansa tüm form elemanlarını basitce alabilmektesiniz.

2.10.2. Server Variables

Request nesnesinin sağladığı bir diğer özellik de "sunucudeğişkenleri" dir. Bir istemci tarafından serverda bir ASP programcığının çalıştırılmasıyla server kullanıcı için bir takım değişkenler oluşturur. Bu değişkenler ASP programcığını çalıştıran her kullanıcı (client) için farklılık taşıyabilir. Bu oluşturulan değişkenleri server variables özelliğini kullanarak programımız içerisinde kullanılabileceğimiz bir değişkene atayabiliriz. Server variables özelliği ile serverdan hangi değişkeni alıcağımızı ise "istek metodu" yardımı ile yaparız. Aşağıda server variables özelliğinin kullanımı gösterilmektedir.

```
<%
  değişken = Request. servervariables("istekmetodu")
%>
```

Örnek 105 : ServerVariables Kullanımı.

Bu örnekte server tarafında olan bir değişkeni programcığımız içerisinde kullanmak amacıyla bir değişkene atadık. Bunu da yapmak için request (istek) nesnemizin server variables özelliğini kullandık.

İstek metodu serverdan alacağımız değişkenin tanımıdır. Bu metodları ise aşağıdaki gibi sıralayabiliriz.

AUTH_TYPE	Kullanıcı kimlik doğrulama tipi.
CONTENT_LENGTH	İstemci tarafından gönderilen veri boyutu (byte biriminde)
DOCUMENT	O an kullanılmakta olan doküman dosyasının adı.
DOCUMENT_URL	O an kullanılmakta olan doküman dosyasının sanal yeri.
DATE_GMT	Server'ın o anki tarihi (GMT formatında)
DATE_LOCAL	İstemcinin o anki tarihi (GMT formatında)
GATEWAY_INTERFACE	Ağ geçidini CGI tanımları
LAST_MODIFIED	O an kullanılmakta olan doküman dosyasının son değiştirilme tarihi
LOGON_USER	Server'da giriş yapmış NT kullanıcısı hakkında detay
PATH_INFO	İstemci tarafından sağlanan ek yol bilgisi
PATH_TRANSLATED	Fiziksel dizin adına çevrilmiş sanal yol
QUERY_STRING	URL dizisi sonundaki soru işareti ile (?) ayrılmış bölümden sonraki bilgi.
REMOTE_ADDR	İstemcinin IP adresi
REMOTE_HOST	İstemcinin Host adı
REMOTE_IDENT	Eğer RFC931 doğrulamasını destekliyor ise istemcinin Host adı
REMOTE_USER	Sunucu tarafından yetkilendirilen istemcinin kullanıcı adı
REQUEST_METHOD	İstemcinin yolladığı istek metodu tipi.
SCRIPT_MAP	Scripting için kullanılan harita.
SCRIPT_NAME	Çalıştırılacak uygulama yada script adı.
SERVER_NAME	Sunucunun IP adresi veya Host adı.
SERVER_PORT	İstemciden isteği alan port adresi http için 80 değerini alır
SERVER_PORT_SECURE	İsteğin şifreli portda olması durumunda "true" (1) değerini alır.
SERVER_PROTOCOL	Protokol adı ve versiyonu, genellikle HTTP/1.1 sunucu ve istemci tarafındaki yazılıma bağlıdır.
SERVER_SOFTWARE_URL	Web sunucusunun yazılımının adı ve versiyonu ve o anki sayfasının URL adresi

Bu istek metotlarından bir kaç tanesini örnekle açıklamak gerekirse.

```
<%
  IP = Request. servervariables("REMOTE_ADDR")
%>
```

Örnek 106 : Kullanıcı IP Adresinin Öğrenilmesi.

Bu programcık çalıştıktan sonra IP değişkenine kullanıcının IP si olan "195.174.129.113" şeklinde bir değer alır.

```
<%
  SN = Request. servervariables("SCRIPT_NAME")
%>
```

Örnek 107 : Script Adının Öğrenilmesi.

Bu programcığın çıktısı olarak SN değişkenimize o an çalışmakta olan scriptin adı dönecektir. Örneğimizde bu "script_name_test.asp" şeklindedir.

```
<%
  LM = Request. servervariables("LAST_MODIFIED")
%>
```

Örnek 108 : Son Düzenlenme Tarihinin Öğrenilmesi.

Bu scriptimizde ise LM değişkenine o an çalışmakta olan programcığın en son düzenlenme tarihi atanacaktır. Örneğimizde bu "03/09/2001" şeklinde olacaktır.

```
<%
  RM = Request. servervariables("REQUEST_METOD")
%>
```

Örnek 109 : İstek Metodunun Öğrenilmesi.

Request metod özelliğini kullanan bu örnekte ise RM değişkenimize örneğe gönderilen verinin gönderiliş biçimi atanacaktır. İki farklı değer karşımıza çıkabilir bunlar GET yada POST olacaktır (eski konularımızda form ile veri gönderme konusunda bilgi verilmişti).

2.10.3. Cookies:

Cookieler, Netscape tarafından, web sunucularıyla istemcilerin etkileşiminde ortaya çıkan bir açığı düzeltmek için geliştirildi. Cookieler olmadan, web sunucuları ve tarayıcılar arasındaki etkileşim bir monologdan farklı olmazdı. Yani server clientdan gelen bilgileri saklayabilirken, istemciler bu işlemi yapamayacaklardı.

Bu işlem için Cookieler kullanılmaya başladı. Böylece asp programları da istemci tarafında veri kaydı yapabilecek ve bunu istediği zaman kullanabilecektir.

Ön bir bilgi verecek olursak şöyle söyleyebiliriz: Cookieler response nesnesi ile kaydederken request nesnesi ile okutulabilen metin dosyaları şeklinde söyleyebiliriz.

İki tür Cookie mevcuttur: oturum Cookieleri ve kalıcı Cookieler. Oturum Cookieleri hafızada saklanırlar, diğer taraftan kalıcı Cookieler aylarca hatta yıllarca kalabilirler. Kalıcı Cookieler kullanıcının bilgisayarında metin dosyası olarak saklanır.

Request nesnesi ile response nesnesinin Cookie özelliği ile kaydedilmiş Cookieleri çağırabiliriz (not: response nesnesi ileri ki bölümlerde ayrıntılı açıklanacaktır). Response nesnesi ile Cookie kaydı şu şekilde yapılır:

```
<%
  Response.Cookies("degisken") = deger
%>
```

Örnek 110 : Çerezlere Değer Atamak.

Bu şekilde "degisken" isimli cookiemize bir değer atamış olduk.

Kaydedilmiş bir cookieyi asp programcığı ile alıp bir değişkene atamak için ise,

```
<%
  deger = Request.Cookies("degisken")
%>
```

Örnek 111 : Çerezden Değer Okumak.

Görüldüğü üzere request nesnesini kullanarak değer isimli cookiemizi bir değişkene atadık. Ve bu şekilde asp scriptimiz içerisinde istediğimiz gibi kullanabiliriz.

Cookieler alt anahtarlara bölünebilir. Yani bir diğer değişle alt kategorilere ayrılabilir. Örnek vermek gerekir ise:

```
<%
  degisken = Request.Cookies("deger")("altdeger")
%>
```

Örnek 112 : Çereze Alt değer vermek.

Yukarıdaki örneğimizde değer kategorisine ait alt degeri değişkenimize aktardık. Bu şekilde kullanılan Cookieler için alt anahtarlarını olduğunu belirtmek gerekmektedir. Bunun için ise :

```
<%
  degisken = Request.Cookies("deger").HasKey
%>
```

Örnek 113 : Çerezlerde Alt değer Kullanılacağını Belirtmek.

Şeklinde bir tanımlama gerekmektedir. Bu şekilde kullandığımız cookienin alt anahtarlarını kullanacağımızı belirtmiş oluyoruz.

Aynı şekilde Cookielere veri kaydı yaparken (response nesnesi ile) alt anahtarlar kullanılabilir.

2.10.4. TotalBytes:

Request nesnesinin bize sunduğu kolaylıklardan bir tanesi de TotalBytes özelliğidir. Bu özellik sayesinde request nesnesi ile istemciden alınan bilgilerin (querystring, form ve Cookie yardımı ile) toplamda ne kadar veri içerdiğini gösterir.

Örneğin istemci tarafından doldurulan bir form içerisinde 100 karakter var ise ve bu server da ki bir asp tarafından request nesnesi ile alınıyorsa TotalBytes özelliğimiz de 100 değerini alır. Yani istemciden sunucuya gönderilen verinin tamamının byte biriminden değeridir.

Kullanımı oldukça basit olan bu nesne çok fazla kullanım alanına sahip olmamakla birlikte bazı durumlarda oldukça işe yarar.

```
<%
  sayac = Request.TotalBytes
%>
```

Örnek 114 : TotalBytes Özelliği.

Yukarıdaki örneğimizde istemciden sunucuya gönderilen verinin miktarını "sayac" isimli değişkenimize atadık.

2.11. Response Nesnesi

Bu nesnemiz bünyesinde on adet fonksiyon bulundurmaktadır. Bunlar

1. Cookie
2. Buffer
3. Expires
4. ExpiresAbsolute
5. isclientconnected
6. Add Header
7. Clear
8. End
9. Flush
10. Redirect
11. Write

Bu fonksiyonlardan bir tanesini request nesnesi içerisinde anlatmış fakat tam anlaşılması için response nesnesi ile beraber kullanılmalı ve anlatılmalı demiştik.

2.11.1. Cookie (çerezler):

Kalıcı Cookielere asp içerisinde Cookie komutu ile erişim sağlayıp üzerinde işlem yaptırtabiliriz. Bunu iki farklı şekilde yaparız.

Bunlardan ilki bir Cookie kaydetmek için kullandığımız yöntemdir. Bu yöntem response nesnesinin içerisinde kullanılır. Bu şekilde istenilen değişkenler bilgisayar üzerinde kalıcı olarak saklanır ve programlarımız tarafından direkt çağrılarak kullanılır.

Aşağıda bir Cookie değişkeninde nasıl veri saklanacağına ilişkin kod yer almaktadır. Dikkate edilecek olursa değişken tipi ne olursa olsun değişken adı eşitliğin sol tarafında yazılmıştır. Değişkene verilecek değer ise eşitliğin sağındadır.

```
<%
  Response.Cookie("ad") = "gunce"
%>
```

Örnek 115 : Çereze Değer Atamak.

Yukarıdaki örneğimizde response (yanıt) nesnesini kullanarak "ad" isimli çerezimize (Cookie) gunce değerini atadık. Bu şekilde kaydettiğimiz Cookie ziyaretçi bilgisayarında bir sonraki işlemimize kadar saklanacaktır. (genel kullanım)

Bir Cookie, Cookie adı ve anahtar (key) ile bu anahtara karşılık bir değerden oluşur. (anahtar kısmı zorunlu kullanım değildir kullanıl masada olur ama işlevsellik açısından kullanmak faydalıdır). Örneğin:

```
<%
  Response.Cookies("cerez")("ad") = "Haldun Özer"
%>
```

Örnek 116 : Çereze Alt değer Atamak.

Yukarıdaki örnekte ziyaretçinin bilgisayarında çerez isminde bir Cookie oluşturup bu Cookie içerisinde ad değişkenine Haldun özer değerini atamış olduk. Bu şekilde

verilerimizi daha derli toplu bir şekilde saklayarak hem kullanım hem de güvenlik açısından kolaylık sağlamış oluruz.

Bir çereze istediğiniz içeriği koyabilirsiniz. Bununla birlikte, çerezlerin belirli sınırlamalarının da farkında olmanız gerekir. Orijinal Cookie spesifikasyonuna göre (http://home.netscape.com/newsref/std/cookie_spec.html), bir tek bilgisayar bütün web sitelerinden toplam en fazla 300 çerez bulundurabilir. Ayrıca, tek bir web sitesi ziyaretçinin bilgisayarına 20'den fazla çerez ekleyemez. Son olarak belirli bir Cookie 4 KB'tan fazla veri barındıramaz. Bu sınırlama, çerezin isminin boyu ve çerez içinde bulunan bilginin boyutuna göre ayarlanır.

Cookie fonksiyonunun veri alışverişi dışında iki adet özelliği vardır bunlar yardımı ile sistemlerimizi daha interaktif geliştirmemiz mümkün olmaktadır. Bunlardan ilki Expire (zaman aşımı süresi) dir.

2.11.1.1. CookieExpire

Cookielerimizi ziyaretçinin bilgisayarına kaydettik, kaydettik de bu çerezler ne zamana kadar o bilgisayarda duracak onu kaydetmedik? İşte bu soruya yanıt bulmaya çalışan Microsoft ustaları Cookie fonksiyonuna bir özellik ekleyerek bu engeli aşmışlar.

Bu özellik sayesinde cookie'lerin ömrünü belirleyebilirsiniz. Bu tarih geçtikden sonra çerezlerimiz artık kullanılmaz olacaktır. Örneğin :

```
<%
  Response.Cookies("cerez") = "Haldun Özer"
  Response.Cookie("cerez").CookieExpires = "November 14, 2001"
%>
```

Örnek 117 : Çerezlere Geçerlilik Süresi Atamak.

Bu örnekte kullandığımız "cerez" isimli Cookie "14 Kasım 2001" tarihinden sonra kullanılamaz hale gelecektir.

2.11.1.2. HasKeys

Cookie'lerin alt anahtarlar içererek birçok değişkeni bir tek cookiede tutabileceğini yukarıda örnekleyerek anlatmıştık. Bu özelliğin kullanılabilmesi için belirtilmesi gerekir bunun içinde haskeys komutu kullanılır. Örneğin :

```
<%
  Response.Cookies("cerez").("a") = "Refiye"
  Response.Cookies("cerez").("b") = "Meral"
  Response.Cookies("cerez").("c") = "Özge"
  Response.Cookie("cerez").CookieExpires = "November 14, 2001"
  Response.Cookies("cerez").HasKeys
%>
```

Örnek 118 : HasKeys Kullanımı.

Bu kullanım tarzı ile cerez isimli cookimize 3 adet alt anahtar atayıp bu anahtarlara bir değer verdik. Daha sonra bu cookiemizin ömrünü ayarladık ve alt anahtarları (dictinary) kullandığını belirttik. Böylece alt anahtar kullanan 14 kasım 2001 son kullanım tarihli bir çerez yaratmış olduk.

Cookilerin ziyaretçinin bilgisayarına nasıl kaydedeceğimizi öğrenmiş olduk. Peki bu çerezleri bilgisayardan nasıl geri çağıracağımızı biliyor muyuz? İsterseniz bir de bu konuya göz atalım.

Not: Bir siteyi ziyaretimizde bilgisayarımıza kaydedilen bir çerezi sadece o site geri çağırıp kullanabilir. O site dışındaki siteler yetkisi olmayan cookilere erişemez.

2.11.1.3. Çerezleri Okumak:

Yukarıda anlatılan şekilde kaydettiğimiz çerezlerimizi programlarımız içerisinde geri çağırmak için request (istek) nesnesinin Cookie fonksiyonunu kullanabiliriz. Örneğin :

```
<%
  ad = Request.Cookies("cerez")
  Response.Write ad
%>
```

Örnek 119 : Çerezden Veri Almak.

Yukarıdaki örneklerde kaydettiğimiz "cerez" isimli cookieyi request nesnesini kullanarak geri çağırdık ve ad değişkenine atadık. (dikkat edilecek olursa değer atanacak değişken eşitliğin solunda, bu değişkene atanan değer ise eşitliğin sağında yer almaktadır).

Bu şekilde basit çerezleri okuyabileceğimiz gibi alt anahtarları olan bir çerezide aşağıdaki şekilde okuyabiliriz.

```
<%
  a=Request.Cookies("cerez").("a")
  b=Request.Cookies("cerez").("b")
  c=Request.Cookies("cerez").("c")

Response.Write a
Response.Write b
Response.Write c
%>
```

Örnek 120 : Çerezden Alt veri Almak.

Örneğimizde 3 adet alt anahtara sahip bir cookienin nasıl okunduğu görülmektedir. Unutmadan belirtmekte fayda gördüğüm bir konuda şu; kayıt yaparken kullandığınız çerez adını çerezi okuturken de aynen kullanmanızdır. Aksi hallerde hata ile karşılaşırsınız.

2.11.2. Buffer

Response nesnesinin bize tanıdığı bir diğer kolaylık ta Buffer (tampon) özelliğidir. Standart bir asp programcığı server da derlendiği an ziyaretçinin browserine yollanır.

Bu işlemin kontrolunu buffer komutu ile yapmaktayız. Varsayılan değeri false (yanlış) olan bu özellik eğer aktif hale geçirilir ise serverda bulunan script server tarafındaki derlenme işlemi bitmeden istekde bulunan istemciye gönderilmez.

Bu komutun çalışabilmesi için HTML başlangıç tagından (<HTML>) önce kullanılması gerekmektedir. Özellikle çok işlem gerektiren scriptlerde baş tarafında bu kodun kullanmakta yarar olabilir.

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

Örnek 121 : Tamponlamayı Ayarlamak.

Bilgisayar programlama literatüründe bir verinin işletilmesi bitmeden çıktı verilmemesi işlemine tamponlama yani buffer denilmektedir.

Tamponlamanın çalışma biçimini daha iyi anlamak için Diyagram 2 : ASP Dosyalarının İşletilmesi ve **İstemciye Gönderilmesi.**) yi inceleyebilirsiniz. Diyagramda adı geçen "Hafıza (Cache)" burada adı geçen "Buffer" ile aynı anlamdadır.

2.11.3. Flush (Hemen Gönder):

Buffer komutunun yaptığı işlemde kod tamamen derlenmeden browsera gönderilmesini engeller demiştik. Flush ise bunun tam aksine o ana kadar işlenmiş (derlenmiş) kodu browser tarafına yollar. Kullanımı ise aşağıdaki gibidir. Programımızın istenilen herhangi bir noktasında o ana kadar işletilmiş kodu ziyaretcinin browserine gönderebilmekteyiz.

```
<%
  ...
  ...
  Response.Flush
  ...
  ...
%>
```

Örnek 122 : Cache'i İstemciye Göndermek.

Bu şekilde kullanıldığı zaman response.flush komutunun üzerinde kalan kısım tamamen işletilene kadar beklenir ve bu işletim bittiği anda çıktılar ziyaretçiye yollanır, ve ardından alt tarafta kalan kısım işletilmeye devam eder.

2.11.4. Clear (Temizle):

Buffer methodu ile script'in sonuna kadar derlenilmesini beklerken bir alanda tutulan HTML, clear metodu ile tamamen temizlenebilir. Flush metodunda o ana kadar işletilmiş olan kod ziyaretçiye yollanırken clear metodunda ise o tamponda o ana kadar yapılmış tüm işlemler silinir.

Böylesine tehlikeli bir metodun kullanım alanlarını sorgulayacak olursak, -ki aklınızda neden böyle bir komut yapılma gereksinimi duyulmuş gibi bir soru kalabilir- birçok yerde sözgelimi elektronik alışveriş sitemizde alışverişten vazgeçildiğinin belirtilmesi üzerine, tampon bölgede tutulmakta olan alınan mallar listesini içeren HTML bu yöntemle temizlenebilir.

```
<%
  ...
  ...
  Response.Clear
  ...
  ...
%>
```

Örnek 123 : Tamponu İstemciye Göndermeden Silmek.

Bu şekildeki bir kullanımda clear metodunun üzerinde kalan ve işletilen kodların HTML çıktıları bu satıra gelindiği an temizlenir. Ve tamamen silinir. O satırdan sonraki komutlar ise yine normal olarak ziyaretçiye gönderilir. Bu sayede programcığımızın çıktısını istenilen anda temizlememiz mümkün olmaktadır.

2.11.5. Expires (Süresi Dolan):

Kullanıcı aksi bir ayar yapmadıkça ziyaret edilen web siteleri "Geçici İnternet Dosyaları" dizinine (cache) kaydedilir ve bir daha ki kullanımda bu dizinden çağırılarak kullanılır. Bu hız bakımdan artısı olan bir özelliktir. Ama haberler gibi süreli bilgilerin yer aldığı bir sitede bu tip bir uygulama sitenin prestijini sarsacaktır. Bu sistemi asp ile kontrol edebiliriz ve dosyaların cache de en fazla ne kadar saklanacağını belirleyebiliriz. Bunu ise asp içerisinde response nesnesinin bir metodu olan expires metoduyla yapabilmekteyiz.

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

Örnek 124 : Expires Methodunu Kullanmak.

Bu şekilde kullanıldığı zaman o sayfanın ziyaretçinin cahce kısmında en fazla 60 dakika saklanabileceğini göstermiş olur. Yani bu kodun yer aldığı sayfaya girdikten sonra 60 dakika geçmeden bu sayfaya tekrar girilirse yine cacheden bilgi alınarak ekrana getirilir.

Not : bu işlemi atlayarak o anki ziyaret edilmekte olan siteyi serverdan yeniden almak için (cacheden almadan) İnternet Explorer'da CTRL+F5 yapılabilir.

2.11.6. ExpiresAbsolute

Yukarıda ayrıntılı olarak anlatılan expires methoduna çok benzeyen bir method olan expiresabsolute özelliği, expires özelliği gibi süre sınırı olarak dakika vermektense belirli bir tarih saat verilir. Bu şekilda o tarih ve/veya saat geldiği cache de saklanan dosyaları Expire edebiliriz (yani süresi dolmuştur diyip geçersiz yapabiliriz).

```
<%
  Option Explicit
  Response.ExpiresAbsolute = #25.07.2003#
%>
```

Örnek 125 : ExpiresAbsolute Methodunu Kullanmak.

Yukarıdaki örnekte belirtilen süre geldiği anda sayfa geçersiz olacaktır. Yani expires methodundaki gibi belirli bir sürede değil belirli bir tarih de sayfa Expire olacaktır.

2.11.7. isclientconnected

Yalnızca okunabilir bir boolean değer olan bu özellik kullanıcının o anda sunucuya bağlı olup halen dosya çekmekte olup olmadığını analiz eder.

```
<%
  if response.isclientconnected then
    Response.flusfh
  Else
    Response.end
  End if
%>
```

Örnek 126 : ExpiresAbsolute Methodunu Kullanmak.

Örnekte eğer kullanıcı sunucuya bağlı ve dosya çekmeye devam ediyor ise işletilen kodun direkt kullanıcıya yollanması aksi halde yani kullancı bağlantısını kesmiş ise işletimin bitirilmesi istenmiştir. Bu şekilde sistem kaynakları kullanımı azalacaktır.

2.11.8. End (Sonlandır)

Response nesnesinim bir diğer metodu olan end metodu o ana kadar işletilmiş olan tüm bilgileri ziyaretciye yollayarak kodun işletiminin durdurulmasını sağlar. Bu işlem sırasında buffer metoduyla da hafızada tamponlanan veri ziyaretciye gönderilir.

```
<%
  ...
  Response.End
%>
```

Örnek 127 : Scripti Sonlandırmak.

2.11.9. AddHeader (başlıklar):

Response nesnesi bize istemci browserindeki sayfanın HTTP parçasının başlık olarak yollanarak girildiğini, sayfanın yapısının nasıl değiştirilebileceğini, kullandıkları her başlığın birer tekil (unique) sayı olduğunu belirtir. Örneğin yönlendirme (redirection) metodu tarayıcıya 303 başlıklı değerini yollar. Başlığın değeri statuscode olarak alınır ve çoğunlukla response nesnesinin statüsü olur.

Bilgi : http://www.w3.org/pub/www/protocols/rfc2068/rfc2068.txt adresinden durum kodlarının tüm listesine ulaşabilirsiniz.

Mutlaka şimdiye kadar benzer bir hata mesajı görmüşsünüzdür. 404 (document no found) sayfa bulunamadı gibi. Tüm kodlar 3 dijit içerir normal doküman 200 OK değerini verir. İlk dijit sorunun kaynaklandığı yeri belirtir.

İlk dijit 1 olduğu zaman bunun anlamı "bilgi" demektir. İlk dijit 2 olduğu zaman bunun anlamı "başarılı" demektir. İlk dijit 3 olduğu zaman bunun anlamı "yönlendirme" demektir. İlk dijit 4 olduğu zaman bunun anlamı "istemci hatası" demektir. İlk dijit 5 olduğu zaman ise bunun sunucudan kaynaklanan bir hata olduğunu anlayabiliriz.

Kendi başlığımızı response nesnesinin addheader metodu ile oluşturabiliriz. Bunun için iki bağlı parametre gerekmektedir. İlki isim ve ikincisi içerikdir. Bunun için aşağıdaki gibi bir yazım geçerli olucaktır.

```
<%
  Response.Addheader "deneme","123"
%>
```

Örnek 128 : İstemciye Gönderilen Belgeye Header Eklemek.

Burada "deneme" isimli yeni başlık için "123" değerini tanımlamış oluyoruz.

Headerlarla ilgili ufak bir bilgiyi 1.3.3 bölümünde bulabilirsiniz.

2.11.10. Write (yazdır):

Response nesnesinin en çok kullanılan metodlarından birisi de write metodudur. Bu metod yardımı ile HTML yardımı ile ekrana çıktı alabilmekteyiz. Bu metodun kullanımını bundan önceki sayılarımızda yer yer anlatmıştık. Genel olarak kullanımı aşağıdaki gibidir.

```
<%
  Response.Write "PC World Türkiye"
%>
```

Örnek 129 : Çıktı Almak.

Burada kullandığımız yere bağımlı olarak bir "PC World" çıktısı alınacaktır.

Bir diğer kullanım şekli ise başlangıç ve bitiş tagları arasına eşitdir kullanarak yazdırmaktır. Aşağıdaki örnekte görülmektedir.

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

Örnek 130 : O Anki Tarihi Yazdırmak.

Bu örnekte response.write ile aynı anlama gelmekte olan yazdırma biçimi gösterilmiştir. Ve ekrana o an ki tarih çıktısı alınır.

Response.write metodunda dikkat edilmesi gereken bir özellik özel karakterlerin yazımıdır. Bu özel karakterlerden en önemlisi " işaretidir. Bu işaretin kullanımı aşağıdaki gibi olmalıdır.

```
<%
Response.Write "Şampiyon ""Beşiktaş"" "
%>
```

Örnek 131 : Tırnak İşaretinin Yazdırılması.

Görüleceği gibi tırnak kullanılmasını istediğimiz yerlere çift tırnak işareti yerleştirdik.

Bu özel karakter yazımının bir diğer yolu da CHR() fonksiyonu yardımı ile yazdırmaktır.

2.11.11. Redirect (Yönlendir):

Belki de çok sık kullanmayacağınız bir komut olmasına rağmen oldukça işlevseldir. Bu komut yardımı ile ziyaretçiyi istediğiniz sayfalara otomatik olarak yönlendirebilirsiniz. Kullanımı aşağıdaki şekildedir.

```
<%
  Response.Redirect "http://www.akkoyun.net"
%>
```

Örnek 132 : Sayfanın Yönlendirilmesi.

Bu şekildeki bir kullanımda site otomatik olarak tırnak içerisinde parametre olarak verilmiş siteye veya sayfaya yönlenecektir. Fakat unutulmaması gereken bir nokta vardır, belirtilen siteye yönlendirildiği zaman yönlendirme öncesi tanımlanan değişkenler ve oturumlar hafızadan tamamen silinir ve yeni sayfaya öyle yönlendirilir. Ve bu şekilde birbirinden bağımsız çalışan siteye yönlenir.

Değişkenlerinizin ve oturumlarınızın yönlendireceğiniz sayfada da geçerli olmasını istiyor iseniz o zaman server objesinin transfer metodunu incelemenizi öneririz.

2.12. Uygulama (Application) ve Oturum (Session) Nesnesi:

ASP'nin varoluş sebeplerinden biriside CGI dillerinin yetersiz kaldığı uygulama ve oturumların başından sona izlenmesidir dersek hiçte abartmamış oluruz. ASP'nin bakış açısında sunucudaki bir siteye bağlanıldığı zaman otomatik olarak bir uygulama (application) başlatılmış demektir. Bunun yanında bağlanan her kullanıcı da oturum açmış olacaktır. Bir grup asp ve html sayfasından oluşan bir siteye toplam olarak uygulama, bu siteye bağlanan her kullanıcıya da oturum denmesinin sebebini şu şekilde açıklayabiliriz.

Uygulama yani application nesnesi sitenin tümüyle ilgili bilgileri yani değişken veya nesne yapılarının tamamı uygulama nesnesi dahilinde tutulmaktadır. Oturum nesnesi ise siteye bağlantı kuran ziyaretçiye özel bilgilerin tutulduğu kısım olmaktadır. Daha önceki konularımızda anlatılan **Error! Reference source not found.**) de ayrıntılı olarak gösterilmektedir.

Kitabın başlarındaki değişkenler konusu iyi anlaşıldıysa her değişkenin sadece kullanıldığı sayfada geçerli olduğunu kavramış olmanız gerekmektedir. Fakat bazı uygulamalarımızda değişkenlerimizin tüm kullanıcılar için aynı olması veya bir kullanıcının tüm oturumu boyunca sabit kalması istenebilir. ASP'nin genel olarak en iyi özelliklerinden biriside budur. Örneğin sadece bağlı olan kullanıcının tüm oturumu boyunca bir değerin sabit kalmasını istediğimiz zaman

```
<%
  Session("pi") = 3.14159276
%>
```

Örnek 133 : Oturuma Özel Veri Atama.

Bu şekildeki bir kullanımda o kullanıcı oturumunda tüm sayfalarda bu "pi" değeri geçerli olacaktır. Ek bir bilgi vermeyi gerekli görüyorum uygulama ve oturum nesnelerinde bir değişkene değer atanacaksa yani uygulama veya oturum nesnesinden değer okutulacaksa "session" yada "application" komutu eşitliğin sağına yazılacaktır. Bunun tam tersi durumlarda yani uygulama yada oturum nesnelerine değer atamak istiyorsanız bu komutlar eşitliğin sol tarafına yazılacaktır. örneğin

```
<%
  Session("pi") = 3.14159276 'değer atama
  Pi = Session("pi") 'değer alma
%>
```

Örnek 134 : Oturum Verisi Atama ve Alma.

Bağlı olan kullanıcıya özel değil de tüm uygulama içerisinde geçerli olan bir değişken atamak istiyorsanız uygulama nesnesini aşağıdaki gibi kullanmanız gerekecektir.

```
<%
  Application("pi") = 3.14159276 'değer atama
  Pi = Application("pi") 'değer alma
%>
```

Örnek 135 : Uygulama Verisi Atama ve Alma.

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 dosyasına bir sayaç yerleştirilerek yapılır.

2.13. Server Nesnesi:

Request nesnesi ile ASP programcıklarımıza veri girişi (gönderimi) yaptık. Response nesnesi ile programdan veri çıktısı aldık. Peki bu verilerimizi işlerken server hep sabit mi çalışacak. Tabi ki hayır. ASP'nin dinamik yapısı nedeniyle server üzerinde yapılan işlemler belirli parametrelerle tanımlanabilir. İşte bu tanımlamaları yapabilmek için server nesnesi kullanılır.

Unutulmaması gereken en önemli nokta server nesnesi ile request nesnesinin bir özelliği olan ServerVariables özelliğinin karıştırılmamasıdır. Servervariables özelliği sadece server ile ilgili değişkenlerin alınabileceği bir özelliktir (sadece okunabilir / readonly). Fakat server nesnesi serverın ASP programcıklarını yorumlarken kullanacağı parametreleri ayarlayabilen bir nesnedir.

Server nesnesinin toplam 5 adet özelliği vardır. Bunlar;

1. ScriptTimeOut
2. CreateObject
3. MapPath
4. HTMLEncode
5. URLEncode
6. Execute
7. Transfer

2.13.1. ScriptTimeOut

Türkçe'si "zaman aşım süresi" olan bu komut yardımıyla ASP'nin dinamik olarak oluşturduğu HTML sayfalarının maksimum oluşturulma süresi ayarlanabilir.

Bir ASP scriptinin dinamik olarak bir HTML sayfasını oluşturma hızı tamamen ASP içerisindeki işlem yoğunluğuna bağlıdır. Örneğin içerisinde birçok birbiri içine geçmiş döngü olan bir script tamamen saf bir scriptten çok daha yavaş işlenerek istemciye gönderilecektir.

Bu özellik hem okunabilir hem de yazılabilir bir özelliktir. Bu yazılabilme ve okunabilme deyimleri ileride sık sık karşımıza çıkacaktır. Bunu şu şekilde açıklayabiliriz: okunabilme özelliği bir nesnenin bir metodunu kullanırken o metoddan değer alabileceğimiz anlamına gelir. Bir örnekle açıklarsak;

```
<%
  A = Nesne(1).Metod(1)
%>
```

Örnek 136 : Metoddan Değer Almak.

Bu örnekte "A" değişkenimize "Nesne(1)" nesnesinin "Metod(1)" metodundan geri dönen değeri atadık.

Yazılabilme özelliği ise bir nesnenin bir metoduna değer verebilme özelliğidir. Örnekle açıklamak gerekirse;

```
<%
  Nesne(2).Metod(2) = "B"
%>
```

Örnek 137 : Metoda Değer Vermek.

Bu kullanım tarzını derslerimize ilk başladığımız sıralarda anlatmıştık. Eşitliğin sol tarafı her zaman kaynağı vermektedir. Sağ taraf ise her zaman hedefi verir. Örneğimizde "Nesne(2)" nesnesinin "Metod(2)" metoduna "B" değerini vermiş bulunuyoruz. Bu iki özelliği aynı anda da kullanabilmekteyiz. Yani bir nesne hem yazılabilir hem de okunabilir olmaktadır.

Bu kadar teorik bilgi verdikten sonra server nesnesinin ScriptTimeOut özelliğine geçebiliriz.

ScriptTimeOut özelliğini okuma amaçlı kullanırken; bu özelliğin o andaki değerinin kaç olduğunu öğrenebiliriz. Burada adı geçen değerin birimi saniyedir yani timeout olayı saniyelerle ölçülür. Okuma özelliğine örnek verecek olursak;

```
<%
  STO = Server.ScriptTimeOut
  Response.Write STO
%>
```

Örnek 138 : Script Derlenme Süresinin Belirlenmesi.

Bu kullanım şekli ile ekrana o anda ayarlı olan timeout değeri dönecektir. Örneğimizde bu değer "90" olarak bulunmuştur. Bu değer saniye cinsinden maksimum sayfa oluşturma süresini verecektir.

Bilgi : Timeout değeri herhangi bir ayar yapılmamış ise geçerli değer olarak 90 alacaktır.

ScriptTimeOut özelliğini ayarlamak ve yeni bir değer vermek istersek bunu nesnenin yazılma özelliği yardımı ile yapacağız. Örneğin;

```
<%
  Server.ScriptTimeOut = 60
%>
```

Örnek 139 : ScriptTimeOut Kullanımı.

Bu şekildeki bir kullanım ile scriptimizin timeout süresini 60 saniye olarak ayarlamış bulunuyoruz. Yani bir diğer değişle scriptimiz 60 saniye içerisinde bir HTML kodu üretemez ise server otomatik olarak işlemi sonlandırarak hata verecektir.

2.13.2. CreateObject

Sunucuya yüklenen her component için bir PROGID (GUID) belirleyici vardır. Bu numara komponenti temsil eder. Bu numara yardımı ile component'i ASP içerisinde kullanılabilen bir nesne olarak tanımlayabiliriz. İşte bu işlem için Server nesnesinin CreateObject özelliğinden faydalanabiliriz.

Bizim burada anlattığımız nesnelerin dışında piyasada kullanılan ve bir çok değişik işlemi gerçekleştiren component mevcuttur. Bu componentleri ASP içerisinde direkt

olarak kullanabilmemiz için ilk olarak tanımlamamız gerekir. Server CreateObject bu işlemi yapan komuttur.

Kullanımı oldukça basit olan bu özellik yardımı ile kendi nesnelerimizi tanımlayabilir ve kullanabiliriz. Örneğin;

```
<%
  Set Nesne = Server.CreateObject("Akkoyun.Web")
%>
```

Örnek 140 : CreateObject Yapısı.

Bu şekilde "Akkoyun.Web" Nesnesini "Nesne" ismi ile tanımlamış bulunuyoruz bu tanımlama işleminden sonra nesne özelliklerini yine her zamanki gibi kullanabilirsiniz. Örneğin;

```
<%
  Nesne.Metod = "ref MTSCS"
%>
```

Örnek 141 : Metoda Değer Atama.

Yukarıda CreateObject ile tanımladığımız "Nesne" isimli nesnemizin bir metodunu kullandık.

2.13.3. MapPath:

Server nesnesinin MapPath metodunu kullanarak, dosyaların konum bilgisini scriptlerimizde kullanabiliriz. Amacı, mantıksal (sanal) yol bilgisini bir istemci tarayıcısı için kullanılabilir hale, yani sunucudaki fiziksel yola uygun hale getirmektir. Aşağıdaki örnekte olduğu gibi, root dizininden hangi sanal dizine dallanacağını gösterilebilir.

```
<%
  'Sanal Dizin /Akkoyun Olsun
  Dim Fiziksel
  Fiziksel = Server.MapPath("/Akkoyun")
%>
```

Örnek 142 : MapPath Metodu.

"Fiziksel" değişkeni içerisindeki gerçek yol bilgisi, "c:\inetpub\wwwroot\akkoyun" olacaktır.

MapPath metodu sunucu üzerinde o dizinin var olup olmadığını anlayamaz.

2.13.4. HTMLEncode:

Normal şartlar altında bir web sayfasında "<Table>" yazdırmak bir hayli zordur çünkü server bu komutu bir HTML tagı olarak yorumlar ve derlemede işletir. Halbuki biz bu tagı sunucuya göstermek istiyoruz.

İşte bu noktada HTMLEncode metodu devreye girer. Bu metod yardımı ile kullanıcının görmesini istediğimiz tag, komut yada özel sembolleri kullanıcıya direkt olarak gösterebiliriz. Kullanımı yine her zamanki gibi oldukça kolaydır.

```
<%
  Response.Write(Server.HTMLEncode("<%=now%>"))
%>
```

Örnek 143 : HTMLEncode Metodu.

Yukarıdaki örnekte görüldüğü gibi kullanıcı ekranında "<%=now%>" şeklinde bir çıktı almış olacağız.

2.13.5. URLEncode:

URLEncode metodu HTMLEncode metoduyla benzerdir. Fakat, bir bilgiyi alır ve URLEncode yapılmış hale dönüştürür. Tüm boşluklar + işareti ile değiştirilir ve diğer işaretler ise % işareti ile başlayan onaltılık düzendeki ANSI eşdeğerleri ile değiştirilirler. URLEncode metodunun amacı, kodumuzda hiperlink bağlantıları oluşturacaksak bunların doğru formatta olduğunu garantilemektir. Aşağıdaki örnekte bu özelliğin nasıl kullanıldığını görebilirsiniz.

```
<A HREF="Test.asp?ID=
<%
  Server.URLEncode("33%")
%>
33% </A>
```

Örnek 144 : URLEncode Metodu.

Tablo 17 : ANSI Kod Tablosu.da ANSI standartlarında decimal değerlere karşılık gelen karakterler bulunmaktadır. Bu tablonun kullanılış biçimi şu şekildedir. Kullanacağınız karakterin bulunduğu sütundaki değer ile bulunduğu satırdaki değeri toplayarak bunu "%" işareti ile belirtmektir.

Örneğin;
® sembolünü kullanmak istiyorsanız bulunduğu sütundaki değere (160) bulunduğu satırdaki değeri (14) ekleyerek ulaşabilirsiniz (174). Bu kodu ise "0174%" şeklinde urlencoded yapabilirsiniz.

ANSI kodlarının toplam 4 basamaktan oluşması gerektiğini unutmayın ve değer eğer 4 basamaktan küçük ise sol tarafına 4 basamak oluncaya kadar sıfır koyunuz.

Bu kodları ALT+ şeklinde de kullanabilirsiniz. ALT tuşuna basılı tutarken "0033" rakamlarını kodlamanız durumunda ANSI karşılığı olan "!" sembolünü elde edersiniz.

DEC	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
0			space	0	@	P	`	p	€	□	nbsp	°	À	Ð	à	ð
1			!	1	A	Q	a	q	□	‘	¡	±	Á	Ñ	á	ñ
2			"	2	B	R	b	r	‚	’	¢	²	Â	Ò	â	ò
3			#	3	C	S	c	s	ƒ	“	£	³	Ã	Ó	ã	ó
4			$	4	D	T	d	t	„	”	¤	´	Ä	Ô	ä	ô
5			%	5	E	U	e	u	…	•	¥	µ	Å	Õ	å	õ
6			&	6	F	V	f	v	†	–	¦	¶	Æ	Ö	æ	ö
7			'	7	G	W	g	w	‡	—	§	·	Ç	×	ç	÷
8			(	8	H	X	h	x	ˆ	˜	¨	¸	È	Ø	è	ø
9	TAB		)	9	I	Y	i	y	‰	™	©	¹	É	Ù	é	ù
10	LF		*	:	J	Z	j	z	Š	š	ª	º	Ê	Ú	ê	ú
11			+	;	K	[	k	{	‹	›	«	»	Ë	Û	ë	û
12			,	<	L	\	l	|	Œ	œ	¬	¼	Ì	Ü	ì	ü
13	CR		-	=	M	]	m	}	□	□		½	Í	Ý	í	ý
14			.	>	N	^	n	~	Ž	ž	®	¾	Î	Þ	î	þ
15			/	?	O	_	o	□	□	Ÿ	¯	¿	Ï	ß	ï	ÿ

Tablo 17 : ANSI Kod Tablosu.

2.13.6. Execute

ASP scriptlerimiz veya HTML sayfalarımız içerisinde harici bir kod dosyasını dahil etmek gerekebilir. Bu durumlarda Include özelliğinin yetersiz kaldığı durumlarda server objesinin Execute metodu kullanılır. Bu metod sayesinde harici bir kod ya direkt olarak dosyadan çağırılarak çalıştırılır yada bir string değişkeni içerisinde saklanan script işletilebilir.

```
<%
  ' harici dosyanın direkt işletilmesi
  Dosya = "harici.txt"
  Server.Execute(Dosya)

  ' string değişkenin işletilmesi
  Komut = "Response.Write('deneme')"
  Server.Execute(Komut)
%>
```

Örnek 145 : URLEncode Metodu.

Bu şekilde bir kullanım ile hem daha dinamik hem de daha esnek siteler geliştirebilirsiniz. Dinamik içerik kullanımı ilerideki konularda ayrıntılı anlatılacaktır.

2.13.7. Transfer

Response nesnesinin redirect metodu ile aynı işi yapan transfer metodu, redirect'in aksine yönlendirdiği sayfaya mevcut olan değişkenleri de taşır.

```
<%
  Server.Transfer("http://www.akkoyun.net")
%>
```

Örnek 146 : Sayfanın Yönlendirilmesi.

Yeni siteye veya asp dosyasına yönlendirme yapılmış oldu ve yönlendirme yapılan uygulamadaki tüm değişkenler ve oturumlar da yeni siteye aktarıldı.

2.14. Global.asa

Global.asa dosyası ASP scriptleri içerir fakat normal bir ASP sayfasının aksine global.asa dosyası içerik görüntülemek için kullanılmaz. Bunun yerine global.asa dosyası global uygulama olaylarını ele almak için kullanılır. Bir diğer değişle tüm scriptlerimiz içerisinde kullanacağımız fonksiyonları yani global fonksiyonları bu dosya içine yerleştiririz. Çünkü server ASP dosyalarından önce bu dosyayı işleme sokacaktır.

Global.asa dosyasını kullanmadan önce bir ASP uygulaması oluşturmanız gerekir. Bu işlemi sadece PWS kullanan kullanıcılarımız yapacaktır. IIS kullanan kullanıcılar bu işlemi atlayabilirler.

Bu dosyanın içeriği standart bir ASP dosyasına benzemekle birlikte adındaki uzantının .asp değil de .asa olmasının sebebi, dosyanın Active server application dosyası olmasıdır.

Bir asp uygulaması oluşturduktan sonra global.asa dosyasını uygulamanızın ana dizinine (root) ekleyebilirsiniz. Global.asa dosyası içerisinde yine vbscript kullanacağız fakat normal vbscript komutlarına ek olarak 4 yeni komut mevcuttur.

Session_OnStart:
Bu olay bir ziyaretçi web sitesine girdiğinde tetiklenir. Ziyaretçi ilk sayfa talep anında bu işlem gerçekleşir.

Session_OnEnd :
Bu olay bir oturum sona erdiğinde tetiklenir. Bir kullanıcı oturumu zaman aşımına uğradığında yada session nesnesinin abandon metodu ile sonlandırıldığında bu olay gerçekleşir.

Application_OnStart:
Bu olay web sunucunuz başlatıldıktan sonra web sitenizden ilk sayfa talep edildiği zaman tetiklenir. Her zaman session_OnStart olayından önce gerçekleşir.

Application_OnEnd:
Bu olay sunucu kapatıldığı zaman tetiklenir ve her zaman session_OnEnd olayından sonra gerçekleşir.

Application ve session nesnelerinin kendi başlarına en çok kullanıldığı yer, siteye gelen ziyaretçilerin sayısını tutmasını sağlamaktır. Bu genellikle global.asa dosyasına bir sayaç yerleştirerek yapılır.

Global.asa dosyası içerisinde "<%" veya "%>" tagları kullanılmaz. Bunun yerine, scriptin başlangıç ve bitişi HTML "<script>" tagı ile işaretleniyor. "<script>" tagının runat özelliğine server değeri verilerek, bunun istemci tarafında değil, sunucu tarafında çalışan bir script olduğu belirtiliyor.

2.15. Dış Kaynaklı Kod Kullanımı

Bir ASP sayfasının içine bir dosya dahil etmek için temel HTML taglarından olan "#INCLUDE" tagını kullanabilirsiniz. Bu tag hem ASP sayfalarında hem de standart HTML sayfalarında kullanılabilmektedir.

Dosya dahil etme işlemi iki durumda faydalıdır. İlk olarak aynı içeriği web sitenizdeki birçok sayfaya eklemeniz gerektiğinde işinize yarayacaktır.

Diyelim ki, web sitenizdeki her sayfanın tepesine koymak istediğiniz standart bir logonuz var. Logoyu bir başlık dosyasına yerleştirip, daha sonra bu başlık dosyasını her ASP sayfasının içine dahil edebilirsiniz.

Şirket logosunu bir başlık dosyasıyla görüntülemek web siteniz için tutarlı bir görünüm oluşturmayı kolaylaştırır, ayrıca ileride şirket logosu değişirse sayfalarınızda değişiklik yapmayı da daha kolay hale getirir. Bu durumda web sitenizdeki bütün sayfalarda değişiklik yapmak yerine, sadece başlık dosyasını değiştirmeniz yeterli olacaktır.

Dosya dahil etme işlemi ayrıca, standart bir fonksiyonlar ve prosedürler kümesini birden fazla ASP sayfasında kullanmanız gerektiğinde de çok faydalı olacaktır. Fonksiyon ve prosedürlerinizi bir kütüphane olarak tek bir dosyada toplayıp, sonra da bu dosyayı diğer ASP sayfalarına dahil edebilirsiniz. Eğer birden fazla ASP sayfasında kullanmanız gereken yeni bir fonksiyona ihtiyacınız olursa yeni fonksiyonu dahil edilen dosyaya eklemeniz yeterli olacaktır.

2.15.1. #INCLUDE:

Bir ASP sayfasına bir dosya dahil etmek için #INCLUDE tagı kullanılır. Dahil edeceğiniz dosya web sunucunuzun erişebileceğiniz herhangi bir klasöründe bulunabilir. Tagın iki kullanım şekli bulunmaktadır. ASP sayfasına kendisi ile aynı klasörde bulunan bir dosya dahil edileceği zaman, aşağıdaki gibi kullanılır.

```
<!-- #INCLUDE FILE="baslik.asp" →
```

Örnek 147 : Include Kullanımı.

Yukarıda bulunan ASP sayfası kodun bulunduğu satırda baslik.asp isimli dosyayı dahil ediyor. Burada #INCLUDE tagının <% ve %> sembolleri arasında kullanılmadığına dikkat etmek gerekiyor.

Dahil edeceğiniz dosya ASP scriptleri içeren bir dosya ise, bu dosyaya dosya uzantısı olarak ".asp" vererek onları ASP dosyası haline getirmeniz iyi olur. Böylece, web sitenize giren kullanıcıların dosyanın içeriğini görmesini engellemiş olursunuz. Eğer dahil edeceğiniz dosyayı ".htm" yada ".inc" gibi bir uzantı verirseniz, herhangi bir dosyayı bir web tarayıcısında açarak dosya içeriğini (ASP kodlarını) görüntüleyebilir.

2.15.2. Dinamik Include:

Dosya dahil ederken bir değişkenin değerine bağlı olarak farklı dosyaları dahil etmek isteyebilirsiniz. Mesela aşağıdaki kodda ASP sayfası #INCLUDE tagını kullanarak ID isimli bir değişkenin değerine bağlı olarak bir dosya görüntüleyecektir.

```
<%
  ID = "5.txt"
%>

<!-- #INCLUDE FILE="<%=ID%>" →
```

Örnek 148 : Dinamik Dosya İçermek.

Maalesef yukarıdaki ASP sayfası istendiği şekilde çalışmayacaktır. Problem şudur, #INCLUDE tagı da dahil olmak üzere bütün taglar, sayfadaki scriptler işlenmeden önce işlenir. Bu da demektir ki yukarıdaki örnekte bulunan #INCLUDE tagı, ismi <%=ID%> olan bir dosyayı dahil etmeye çalışacaktır. Tabi ki buda bizim istediğimiz şey değildir.

Eğer bir değişkenin değerine bağlı olarak farklı dosyaları dinamik şekilde dahil etmek istiyorsanız, ya bir ya bir vbscript şartlı ifadesi (if), yada bir vbscript select...case ifadesi kullanmanız gerekir. Aşağıdaki örnekte ID değişkeninin değerine bağlı olarak farklı dosyaları doğru şekilde içerdiği görülmektedir.

```
<%
  ID = 2
  Select Case ID
  Case 1
%>
  <!-- #INCLUDE FILE="1.asp" -->
<%
  Case 2
%>
  <!-- #INCLUDE FILE="2.asp" -->
<%
  End Select
%>
```

Örnek 149 : Dinamik Veri İçeriği Ekleme.

Burada şu husus önemli: yukarıdaki scriptteki dosyalar şartlı olarak dahil edilmiyor. #INCLUDE tagı bütün scriptlerin çalıştırılmasından önce yorumlandığı için, "Select...Case" ifadesi yorumlanmadan önce bütün sayfalar dahil edilerek tek bir büyük dosya halinde birleştiriliyor. Sonra bu büyük dosyanın sadece şartlara uyan kısmı görüntüleniyor.

Bu tip bir dinamik içerik eklemenin yanında birde server objesinin bir metodu olan Execute komutu vardır (2.13.6 Execute). Bu metod yardımı ile harici dosyalarda saklanmış scriptleri direkt olarak okutup çalıştırtabiliriz bu şekilde dinamik içerik sağlanmış olacaktır. Bu yöntem çoklu seçme yapılmayacak durumlarda çok kullanışlı ve performanslı olacaktır.

```
<%
  ID = "5.txt"
  Server.Execute(ID)
%>
```

Örnek 150 : Dinamik Dosya Çalıştırmak

Çok fazla dosya dahil edilerek büyük bir ASP dosyası oluşturulması problemlere yol açabilir. Problemler bir kullanıcı tarafından ilk defa bu sayfayı talep ettiğinde çıkacaktır. Web sunucunuzun sayfayı oluşturması uzun bir zaman alabilir. Sayfa değiştirilmediği yada web sunucunuz kapatılmadığı müddetçe, sonraki talepler çok daha hızlı bir şekilde karşılanabilir. Çünkü web sunucusu sayfayı ön hafızaya alacaktır.

2.16. Aktif Sunucu Bileşenleri

Birçok yönden bu bileşenler Active-x kontrollerine benzer. Fakat, aktif sunucu bileşenleri tarayıcıya bir nesne göndermektense nesneyi sunucu üzerinde çalıştırmak için tasarlanmıştır. Bunun da, bu yazımızda göreceğimiz gibi birçok avantajı vardır.

Sorulması gereken bir soru da şudur: Bu bileşenler nereden geliyor? Bunların bir kısmı ASP kurulumu sırasında sağlanır.

Diğerleri de ya satın alınır ya da bedava bulunabilir. Bu yazıda Microsoft tarafından sağlanan birçok nesnenin aslında ne kadar çok faydalı olduğunu göstermeye çalışacağım. Bu bileşenleri kullanmaya alışırsanız başkalarının sağladığı bileşenleri de kendi sayfanızda kullanırken zorlanmazsınız.

Bununla birlikte standart bileşenler vardır. Biz bu bileşeni bu yazımızda ele almayacağız. Dinamik web sitesi teknolojileri kullanmanın esaslarından bir tanesi de, farklı tipteki veri tabanı yönetim sistemlerindeki bilgiyi doğrudan yayımlamaktır. Burada ihtiyaç duyulan veriyi toplamak ve bir veri tabanında saklamaktır. Bu hedefe, ASP kullanarak ulaşmak için özel fakat genel amaçlı bir bileşen olan Active-X veri nesneleri olarak bilinen bileşenin avantajlarını kullanacağız.

2.16.1. Sunucu Bileşenleri:

VBScript ve JsScript için kullanılabilir çeşitli script geliştirme nesnelerini görmüştük, şimdi bunların diğer bileşenlerle nasıl birlikte kullanıldıklarını göreceğiz. Buradaki önemli bir nokta da, sunucu bileşenleri ve hazır script geliştirme nesnelerinden bahsederken de bunları birbirine karıştırmamaktır. TextStream ve Dictionary gibi script geliştirme nesneleri script geliştirme ortamının birer parçasıdır. Bunlar ASP sisteminin bir parçası olan bir DLL tarafından oluşturulmuştur.

Sunucu bileşenleri, script geliştirme dilleri DLL'lerinden farklıdır. Bunlar kendilerine ait DLL'lerin içinde oluşturulurlar. Mesela bu yazıda inceleyeceğimiz Content Linking bileşeni NextLink.DLL dosyasının içerisinde oluşturulur. Bu, sunucu üzerine kurulup kaydedildikten (register) sonra ASP'nin kurulumunda desteklediği tüm script geliştirme dilleri ile beraber kullanılabilir.

Göreceğiniz gibi bazı sunucu bileşenleri daha önce gördüğümüz built-in script geliştirme nesnelerinin gücünü kullanır. Bu bileşenlerin her birisine bakmadan önce genel olarak kullanacağımız sunucu bileşenlerine bakalım.

2.16.2. Genel Bakış:

Geçtiğimiz konularda, ASP paketi tarafından sağlanan bileşenlerin kullanımının yönetimini gördük. Sunucu nesnesinin metotlarından birisini kullanarak browser capabilities bileşeninin bir kopyasını oluşturduk:

```
Set NT = Server.CreateObject("MSWC.BrowserType")
```

Örnek 151 : BrowserCapabilities Objesinin Kullanımı

Bu nesne NT değişkenindeki nesne için bir referans oluşturur ve biz daha sonraki scriptlerimiz içinde nesne ile çalışabiliriz. Başka bir değişle bu nesnenin özelliklerinin bize uygun olanlarını kullanır ve metotlarının gerektiği kadarını koddan çağırırız.

Standart bileşenlerin çoğu, web sitenizin veya şirket İntranet'inizin başarısı için belirli türdeki görevleri hedef edinmiştir. ASP kullanarak başarmak isteyeceğiniz şey kesinlikle yayımlamak istediğiniz bilginin türüne ve sitenizin genel amaçlarına bağlıdır. Hazır bir bileşeni kullanmak başlangıç için size bir yol gösterir.

2.16.3. Bileşenleri Kullanmak:

VBScript veya JsScript ile sunucu nesnesinin CreateObject metodunu kullanabiliriz. Bunu daha önce bir bileşenin kopyasını oluştururken anlatmıştık. Yada nesne referansını doğrudan bir değişkene atayarak daha sonra kodumuzda kullanabiliriz.

2.16.4. <Object> Kullanmak:

Başka bir yol olarak ASP'deki bir nesnenin kopyasını oluşturmak için <OBJECT> etiketini kullanabiliriz. Aynı yolla tarayıcı üzerindeki bir Web sayfasındaki bir nesnenin kopyasını da oluşturabiliriz. ASP , HTML <OBJECT> etiketi için özel bir uygulama sağlar ve biz bunu sayfamıza bir nesne yerleştirmek için kullanabiliriz. Normal bir .asp dosyası içinde bir bileşenin kopyasını veya script geliştirme nesnesini tanımlamak için şöyle bir yazılım kullanırız:

```
<Object Runat=Server ID=NesneReferansı ProgID=”Nesne Tanımlayıcı”></Object>
```

Örnek 152 : <Object> Kullanmak

Nesne Referansı, Nesne Kopyası gibi bir isimdir. Bu isimler kodumuz içinde kullanacağımız nesneleri gösterir. Nesne Referansı Windows Registry içindeki bir nesnenin veya bileşenin ismidir. Mesela MSWC. Adrotator gibi. Böylelikle Ad Rotator nesnesinin bildirimini kodumuz içerisinde aşağıdaki gibi yapabiliriz:

```
<Object Runat=Server ID=ObjAdRot ProgID=”MSWC.Adrotator”></Object>
```

Örnek 153 : <Object> Kullanımı

Sunucu olarak ayarlanması gereken RUNAT özelliğine dikkat edin. PROGID,[Vendor.] Component [.Version]. formunda bir bileşeni veya nesneyi tanımlayan tek (unique) bir metin dizisidir.

Ya da PROGID yerine nesnenin CLASSID' sini kullanabiliriz:

```
<Object Runat=Server ID=ObjAdRot ClassID=”Cisid: OACE4881-8305-11CF-
9427-444553540000”></Object>
```

Örnek 154 : ClassID Kullanımı

2.16.5. Çalışma Alanı:

Normal bir .asp dosyası içerisinde CreateObject veya <OBJECT> etiketinin kullanılması genel olarak bir performans farkına sebep olmaz.Gerçek fark nesnenin yaratıldığı veya kopyalandığı noktadır. CreateObject metodu nesnenin kopyasını, çağırıldığı zaman oluşturur. <OBJECT> etiketi ile tanımlanan bir nesne ise kendisine ilk referans verilen noktaya kadar gerçek olarak oluşturulmaz. Bu da bize, nesnenin global.asa dosyasında tanımlanıp .asp dosyasında gerçek olarak tanımlanmadığı yerlerde, kaynak kullanımını azaltmamıza veya sunucu performansını arttırmamıza olanak verir.

Geçmiş sayılarda global.asa ve Session ve Application nesneleri ile karşılaşmıştık. Biz sadece bu dosya içerisinde nesnelerin kopyalarının oluşturmanın önemini anlamalıyız ve böylece bunlar Session veya Application' da tanımlanan bir çalışma alanına sahip olurlar.

Bunu <OBJECT> etiketi içinde SCOPE özelliğini koyarak sağlarız:

```
<Object Runat=Server Scope=Session ID=ObjAdRot ProgID=”MSWC.Adrotator”>
</Object>
```

Örnek 155 : Scope Özelliğinin Kullanımı

SCOPE için geçerli değerler Session.Application veya Page'dir. Eğer <OBJECT> etiketini normal bir .asp dosyası içinde kullanırsak ya Page'i kullanmalıyız yada SCOPE özelliğini tamamen ihmal etmeliyiz. Eğer <OBJECT> etiketini global.asa içerisinde kullanırsak bir nesne oluştururuz, bu nesne ya mevcut oturumun tamamı için kullanılabilir olur yada global olarak uygulamanın tamamen dışında olur.

2.16.6. Bileşen Metodları:

ASP içerisindeki bileşenleri kullanmamızın tüm sebebi sağladıkları ekstra işlevsellikten faydalanmaktır. Bu işlevselliğe sadece script geliştirme dillerini kullanarak erişmek ya mümkün değil yada ulaşılsa bile etkisiz veya zordur. Bir nesnenin kopyası kullanılabilir olduğu zaman onun için bir referansımız olur ve ihtiyaç duyduğumuz sonuçlara ulaşmak için onun metotlarını çağırabilir, özelliklerini kullanabiliriz. Her bileşenin kendine özgü metotları ve özellikleri vardır. Bu yüzden, bunların ne olduğunu ve her birinin sağladığı argümanların ve doğru yazılımlarını bilmeliyiz. Mesela, Content Linking bileşeni GetNextURL adındaki bir metoda sahipken, Advertisement Rotator bileşeni de GetAdvertisement metoduna sahiptir.

Birçok bileşen metodu bir değer döndürür. Birçok durumda bu bizim istediğimiz bir şeydir. Mesela Content Linking bileşeninin GetNextURL metodunun döndürdüğü URL gibi. Diğerleri ise, sadece dışardan değer alırlar. Mesela User Properties bileşeninin Append metodu gibi. Ayrıca bu metodlar işlemin başarılı olup olmadığından emin olmamız için True veya False değeri döndürürler.

3. ADO nedir? Ne değildir?

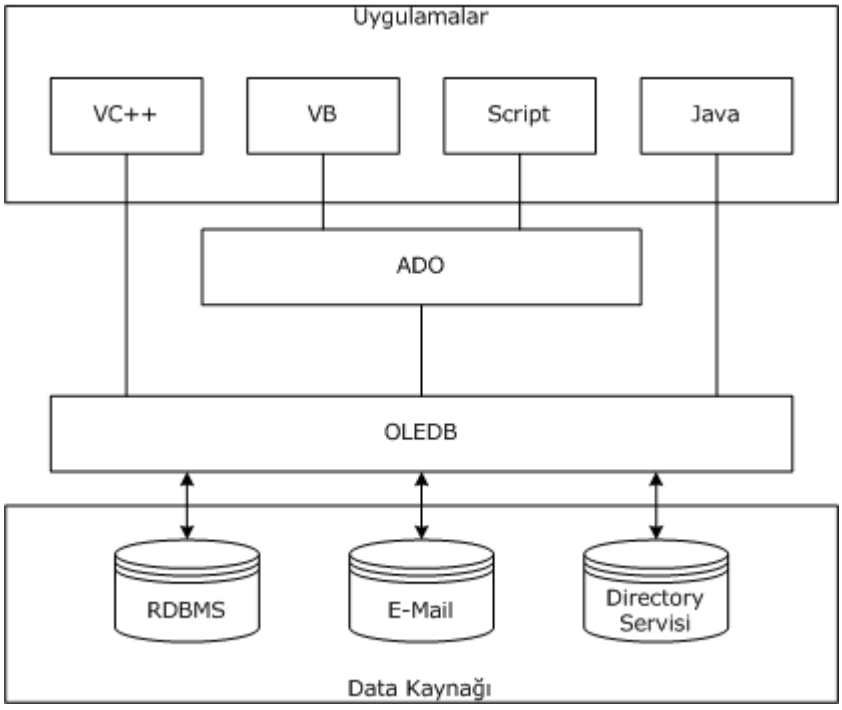
3.1.Neden ADO ?

"ADO" basit ve zengin bir fikir olarak ortaya atılmıştır. Bunu şu şekilde açıklayabiliriz: "Veriye erişebilmek için sadece bir yolunuz vardır" . Bu teknoloji yeni bir teknoloji değildir, uzun zamandır kullanılmakta olan ve gelecekte de kullanılacak olan bir teknolojidir. Ama gelecekte bir çok yeni teknolojinin (DAO ve ODBC gibi) geliştirilmekte olan bir çok uygulama için biçilmiş kaftan olacağı da bir gerçektir.

Daha önceleri veritabanı programcılığı ile uğraşanlar ODBC ve RDO'yu yakından tanımaktadırlar. Open Database Connectivity (ODBC) bir Aplication Programing Interface (API) olup "Access" ve "SQL Server" gibi veri kaynaklarına erişebilmektedir. Bir API olmasından dolayı birçok uygulama geliştirici ODBC'yi (özellikler Visual Basic alanında) komplike bulmaktadır. Remote Data Object (RDO) ise ODBC'nin en üst katmanında yer alan bir ActiveX'dir. ODBC ile tamamen tümleşik çalışmaktadır fakat çok daha kolay kullanışlıdır. Genel anlamda OLE DB ile ODBC'yi ve ADO ile RDO'yu eşleştirebiliriz.

3.2.OLE DB ve ADO Yapıları

Şimdiye kadar OLE DB ve ADO ile ilgili yüzeysel bilgi verdik fakat birbirleri ile olan ilişkilerini ancak bir diyagram üzerinde anlayabiliriz. Bu diyagramda uygulamalar ile veri kaynakları arasında nasıl bir ilişki olabileceği hakkında bilgi verilmiştir.



Diyagram 3 : ADO Köprü Yapısı.

Diyagramdan da görüleceği gibi en üst seviyede uygulamalarımız yer almaktadır (bir web veya normal bir uygulama olması hiç fark etmez). Bu katmanın hemen altında ADO ve/veya OLE DB veri kaynağından alınan verileri uygulamaya iletmek için yer alırlar. Fakat OLEDB tüm programlama dilleri ile beraber çalışabilecek şekilde değildir bundan dolayı ADO, OLEDB üzerinde bir geçiş katmanı görevi yapmaktadır. ADO OLEDB ile OLEDB'nin desteklemediği diller arasında bir arabirim görevi yapmaktadır. ADO OLEDB ye göre çok daha kolay bir programlama arabirimine sahiptir bu sebepten dolayı direkt OLE

DB erişimi (kullanımı) olabilen programlama dilleri (C++ ve Java gibi) veri kaynağına erişimlerini daha kolay hale getirebilmek için ADO kullanabilmektedirler.

Diyagramda sadece Microsoft programlama dilleri gösterilmiştir fakat ADO'nun bir COM bileşeni (component) olduğu düşünülürse ADO diğer COM destekli programlama dillerinde de (Delphi veya Active Scripting Interface destekleyen Script dilleri) kullanılabilmektedir. Şüphesiz ki VBScript ve Jscript içeren ASP sayfalarımızda da ADO bileşenini kullanabiliriz.

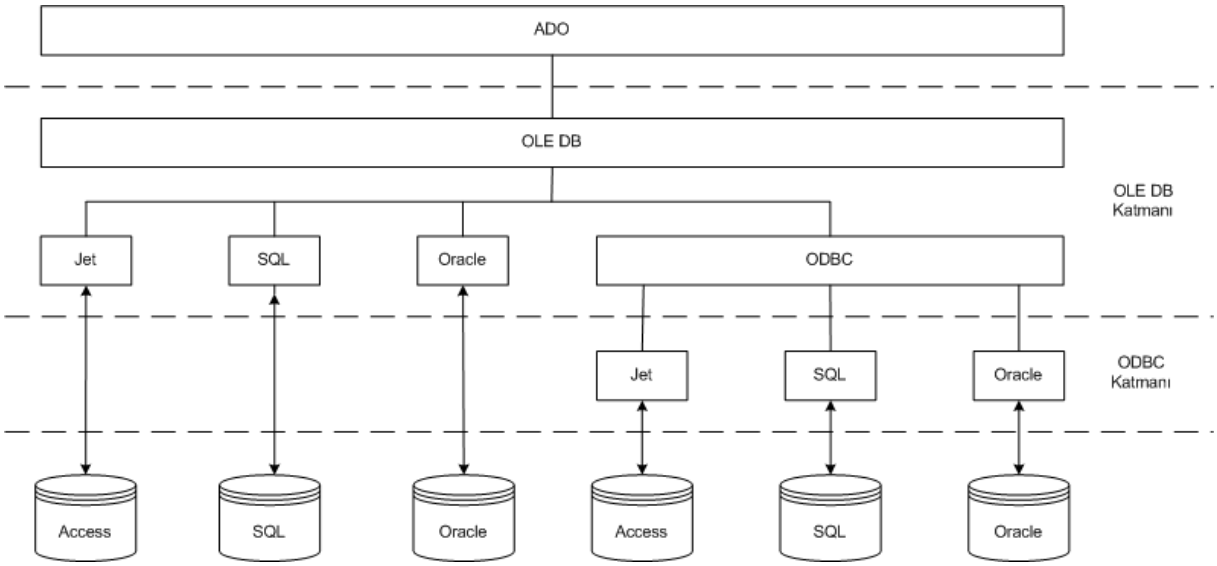
Data erişimi için OLEDB ve ADO kullanılabileceğini öğrendik. Peki ya neden? Eski metodları neden kullanmıyoruz? Bunun iki büyük sebebi var.

Birinci sebep OLEDB ve ADO'nun bir "Veri Kaynağına" erişmek için tasarlanmış olmasıdır. Dikkatinizi çekerim "Veri Tabanı" demedim "Veri Kaynağı" dedim. Veri tabanları en çok kullanılan veri kaynağı da olsa birçok uygulama (mesajlaşma sistemleri, Microsoft Exchange Server, Dizin Hizmetleri ve tabi ki Web Sunucuları) veritabanı dışında bir yapı kullanmaktadırlar.

İkinci sebep ise İnternet uygulamalarının hızla yaygınlaşmasıdır. Eski data erişim metodları webden data erişimi için geliştirilmemiştir.

3.3.Destekleyiciler ve Sürücüleri

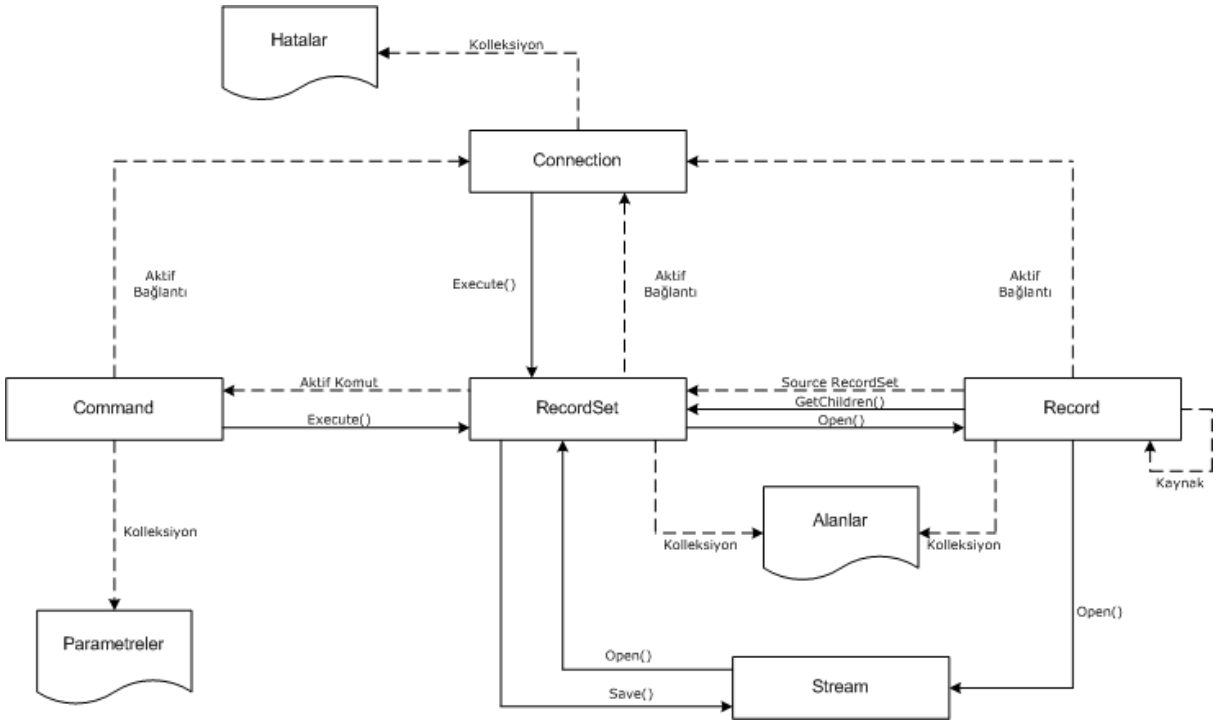
Unutulmaması gereken bir nokta da ODBC için OLEDB destekçilerinin olduğudur. Bu OLEDB'nin ODBC data kaynaklarına erişmesinde bir aracıdır. ODBC için geliştirilen sürücüler (Destekleyiciler tarafından) mevcuttur bu sürücüler sayesinde ODBC esnek bir yapıya sahip olur. Diyagram 4 : ADO Bağlantı **Yapısı.**) de açıkça görüleceği gibi veri erişiminde çeşitli katmanlar vardır. Destekleyiciler OLE DB katmanında yer alırken sürücüler ise ODBC katmanında yer almaktadır. Eğer bir ODBC data kaynağı kullanmak istersek ODBC için bir OLEDB destekleyicisi kullanmamız gerekir. Şayet ODBC data kaynağı kullanmak istemezseniz o zaman bir OLEDB destekçisi kullanmanız gerekir.



Diyagram 4 : ADO Bağlantı Yapısı.

3.4.ADO 2.5 Obje Modeli

ADO 2.5 obje yapısı bir önceki versiyonuna göre çok daha basit bir yapıdadır. Bir önceki versiyonuna ek olarak iki yeni obje eklenmiştir. Diyagram 5 : ADO Obje **Yapısı.** de bu yapıya ait diyagram verilmiştir.



Diyagram 5 : ADO Obje Yapısı.

Eğer daha önceden ADO kullanmış iseniz (eski versiyonlarını) Stream ve Record objelerinin yeni eklendiğini fark edebilirsiniz.

3.4.1. Connection Objesi

"Connection" objesi data kaynağına bağlanmak için kullanılmaktadır. Bu obje sayesinde hangi OLEDB destekçisinin kullanmak istediğimizi belirtebiliriz. Connection objesi data kaynağına bağlanmak için tek yol değildir. Command, Recordset ve Record objelerini Connection objesi olmadan da kullanabilirsiniz. Connection objesini daha çok bağlantıya ait özellikleri belirtmek için kullanırız. Eğer destekleyici üzerinden bir dizi komut çalıştırmak isterseniz Connection objesini kullanmanızı öneririz.

3.4.2. Command Objesi

Command objesi data kaynağı üzerinden komutlar çalıştırmak için tasarlanmıştır. Fakat bir dakika "Connection objesi de bunu yapabilir" diyenlerinizi duyuyorum. Evet Connection objesi ile de yapılabilir ama bir farkla Connection objesinden çalıştırılacak bir komut arka alanda yine Command objesini çalıştırır. Bir komut çalıştırdığımız zaman geriye bir dizi kayıt alırız. Command objesi geriye kayıt döndürmeyen komutların kullanımı içinde uygundur (yeni kayıt eklemek veya kayıt güncellemek için kullanılan SQL sorguları buna örnektir).

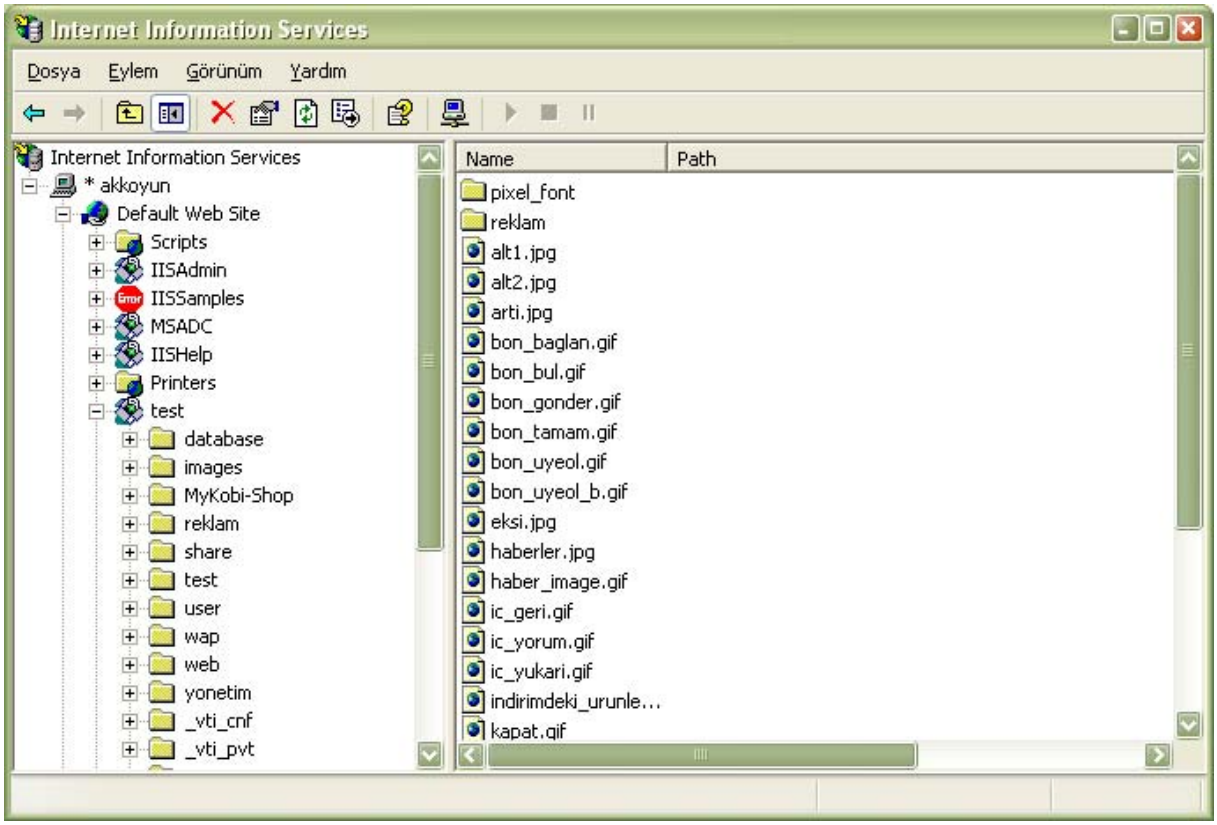
3.4.3. Recordset Objesi

Recordset objesi ADO içerisinde en çok kullanılan objedir. Bu obje data kaynağından aldığı veriyi bir dizi şeklinde bize sunar. Bu obje sayesinde ADO bize veriler üzerinde değişiklik yapmamıza, kayıtları taşımamıza ve kayıtları filtrelememize izin verir. Recordset objesi "Fields" koleksiyonunu içerir. Bu koleksiyon sayesinde veri kaynağındaki tüm alanlara (kolonlara) erişebiliriz.

3.4.4. Record Objesi

ADO'nun daha önceki versiyonlarında veriyi işlemek ve kayıt setleri oluşturmak şimdiki gibi kolaydı, ama sadece veri tabanlarında yani belirli bir kolon yapısına ve data yapısına sahip olanlar için. Bunlar dışında kalan dosya ve posta sistemlerinin veri kaynakları için kullanılamıyordu. Bu tip veriler için 2.5 versiyonunda "Record" objesi geliştirildi.

Bu tip yarı-biçimlenmiş veri kaynakları bir ağaç yapısı şeklinde olurlar, yani ana düğümler ve alt düğümler içerirler. Resim 8 : IIS Ağaç **Yapısı.** de görülen bir web sunucusunun ağaç yapısıdır. Görüleceği üzere "test" ana düğümü altında "Database" ve "images" gibi alt düğümler mevcuttur.



Resim 8 : IIS Ağaç Yapısı.

Bu düğüm noktaları da kendi içerisinde bir çok dosya barındırır. Bir asp dosyası, bir metin dosyası ve bir Word dosyası bunları kafanızda bu ağaç yapısında olduğu gibi hayal edebilirsiniz. Dosya adının yanı sıra tür, son erişim tarihi ve boyut gibi bilgileri de bu yapıya ekleyebiliriz. Peki ya erişimler yani kullanıcıların nerelere erişeceklerini ve nelere erişemeyecekleri.

Tüm bu yapının kompleks haline Record objesi adı verilir

3.4.5. Stream Objesi

Stream objesi Record objesinde açılan düğümleri okumak için kullanılır (web sunucu veya bir elektronik posta ya erişim yapamaz ama içeriğini okuyabilir). Bundan dolayıdır ki Record ve Recordset objeleri ile bütünleşik çalışır.

Stream objesinin bir diğer önemli kullanım alanı da XML dosyalarına erişimdir (XML verisi yukarıda anlatılan yarı-yapılandırılmış data yapısındadır).

Stream objesi binary verileri işlemek içinde kullanılabilir. Buna örnek olarak resim işleme veya büyük boyutlu metin veri tabanları gösterilebilir.

3.4.6. Parametreler Koleksiyonu

Parametre koleksiyonu sadece Command objesi ile birlikte kullanılır. Ve komutların parametrelerini belirlemeye yarar. En çok kullanım alanı SQL Server'da bulunan "Stored Procedure" dır. Bu özellik sayesinde SQL Server'da önceden ayarlanmış olan SQL komutları işletilmiş olur. Bu çok kullanışlı bir özelliktir ve zamandan kazandırır.

Bir diğer özelliği ise komut çalıştırılmasından sonra geriye dönen değer eğer tek bir veriden oluşuyorsa yani bir set şeklinde değilse o zaman parametre özelliği olarak geri döndürülür.

3.5.ADO Sabitleri

ADO kullanımıyla beraber bir çok sabit devreye girer (cursor tipi ve kilit tipi gibi). Bu sabitler ADO içerisinde sayılarla ifade edilir ama daha kullanışlı hale getirmek için bu sayılara isimler atanmıştır. Visual Basic ve Visual C++ da bu sabitler ADO tip kütüphanesini (Type library) tanımladığınız anda otomatik olarak tanımlanacaktır. Fakat ASP için bu geçerli değildir. Bu sabitleri tanımlayabilmemiz için iki yöntem vardır.

Sabitleri tanımlamanın ilk yöntemi onları ASP dosyalarınızın içerisine dahil etmeniz (Include) olacaktır.

```
<!--#INCLUDE FILE = "adovbs.inc"-->
```

Örnek 156 : ADO Sabitleri Tanımlamak

Bu satırı kullanarak ADO sabitlerini içeren "adovbs.inc" dosyasını ASP sayfamıza dahil ediyoruz. Böylece tüm sabitler tanımlanmış oluyor. Fakat burada dikkat edilmesi gereken önemli bir nokta bu dosyanın sitenin ana dizini içerisinde bulunmasıdır. Bu dosyanın varsayılan olarak bulunduğu yer "c:\program files\common files\system\ado" dizinidir. Aşağıda örnekte de gösterildiği gibi bu dosya sitenin ana dizinine yerleştirmeyip sistem dizini üzerinden de tanımlama yapılabilmektedir.

```
<!--#INCLUDE FILE = "c:\program files\common files\system\ado adovbs.inc"-->
```

Örnek 157 : ADO Sabitlerinin Fiziksel Yolunu Belirtmek

Bu dosya sadece VBScript için ADO sabitleri içerir eğer Jscript için ADO sabitleri kullanacaksanız "adojavas.inc" dosyasını kullanmalısınız. Bu dosyayı kullanmanın tek dezavantajı ise ASP dosyalarınız tüm sabitleri tanımladığı için büyütmesidir, bunların içerisinde sizlerin kullanmayacağı onlarca sabit vardır. Bu dezavantajı yok etmek için

kendi sabit dosyanızı hazırlayabilirsiniz fakat bu işlemi yapabilmeniz için ADO komutlarını çok iyi bilmelisiniz.

Birinci yöntemden daha iyi performans veren ve çoğu programcı tarafından kullanılmayan ikinci yöntem ise bu sabitlerin bulunduğu tip kütüphanesini (type library) ASP dosyamıza referans olarak vermektir. Böylece tüm sabitlerimize onları dosyamıza dahil etmeden (ve dolayısı ile performansımızı düşürmeden) dahil etmektir. Bunu şu şekilde yapabilirsiniz.

```
<!--METADATA TYPE = "typelib" FILE="c:\program files\common files\system\ado\msado15.dll" -->
```

Örnek 158 : ADO Sabitlerini Kütüphaneden Almak

Komutta adı geçen "msado15.dll" dosyasının güncelliği hakkında endişelenmenize gerek yoktur çünkü o dosya her zaman güncel sabitleri içerir. ASP dosyalarınızın hangisinde bu sabitler gerekli ise o sayfaya bu METADATA komutunu ekleyerek o sayfada sabitleri kullanabilirsiniz. Yada bu komutu "global.asa" dosyanıza yerleştirmeniz durumunda site içindeki tüm ASP sayfalarınızda otomatik olarak tüm ADO sabitlerini kullanabilirsiniz.

3.6.Veri Kaynağına Bağlanmak

Eğer bir veri kaynağına (data stores) erişmek istiyorsak ilk olarak o veri kaynağına bir bağlantı oluşturmalıyız. Bu işi ADO bizim yerimize yaparak veri kaynağına bağlantı kurmamıza olanak tanır. Geçen ayki sayımızda ADO'nun birçok destekleyici tarafından yazılan sürücüler yardımı ile çeşitli veritabanı tiplerine bağlandığını söylemiştik.

ADO ile veri kaynağına bağlantı kurmamız için bir çok yol vardır.

3.6.1. Bağlantı metni (Connection String)

Bağlantı metnini kullanırken unutulmaması gereken en önemli etken destekleyiciyi belirtmektir. Destekleyiciyi belirtirken "provider=" değimi kullanılır bu değimin kullanılmadığı bağlantı metni otomatik olarak ODBC bağlantı olarak tanımlanmış olur. Aşağıda çeşitli destekleyici tiplerine göre bağlantı metinleri yazılmıştır.

3.6.2. Microsoft Access:

DSN'siz ODBC bağlantısı yapmak için.

```
Driver = {Microsoft Access Driver (*.mdb)}; DBQ=c:\veritabani.mdb
```

Örnek 159 : DSN'siz ODBC Bağlantısı

Buna alternatif olarak OLE DB bağlantısı yapmak için.

```
Provider=Microsoft.Jet.OLEDB.4.0; Data Source = c:\veritabani.mdb
```

Örnek 160 : DSN'siz OLE DB Bağlantısı

Buradaki bağlantı örnekleri fiziksel bir dizin içerisindeki veritabanına bağlantı için sözkonusudur. Bu dizin istenildiği gibi düzenlenebilir. Yine aynı şekilde fiziksel dizin yerine

sanal bir dizinde kullanılabilir ama bu sanal dizinin fiziksel dizin eşdeğerini veren "server.mapath" komutu ile kullanılabilir.

3.6.3. Microsoft SQL Server:

ODBC destekleyicisi kullanarak MSSQL Server'a bağlanmak için.

```
Driver={SQL Server}; Server=Server_adi; Database=veritabani_adi;
UID=kullanici_adi; PWD=kullanici_sifre
```

Örnek 161 : SQL Server Bağlantısı

Örnek olarak:

```
Driver={SQL Server}; Server=Akkoyun; Database=Muzikler;
UID=akkoyun_db_user; PWD=akkoyun_db_password
```

Örnek 162 : SQL Server Bağlantı Örneği

Buna alternatif olarak aynı bağlantıyı OLE DB ile yapmak istersek:

```
Provider=SQLOLEDB;
Data Source=Server_adi;
Initial Catalog=veritabani_adi;
User ID=kullanici_adi; Password=kullanici_sifre
```

Örnek 163 : SQL Server OLE DB Bağlantısı

Bu bağlantı tipine örnek olarak:

```
Provider=SQLOLEDB;
Data Source=Akkoyun;
Initial Catalog=Muzikler;
User ID=akkoyun_db_user; Password=akkoyun_db_password
```

Örnek 164 : SQL Server OLE DB Bağlantı Örneği

Şeklinde olacaktır.

3.6.4. Microsoft Indexing Service:

Indexing servisine sadece OLEDB bağlantısı ile bağlanılabilir bağlantı metni aşağıdaki gibidir.

```
Provider=MSIDX; Data Source=Veri_kaynağı
```

Örnek 165 : Index Bağlantı Metni

Bu bağlantı tipine örnek olarak "web" kataloguna bağlanmak için:

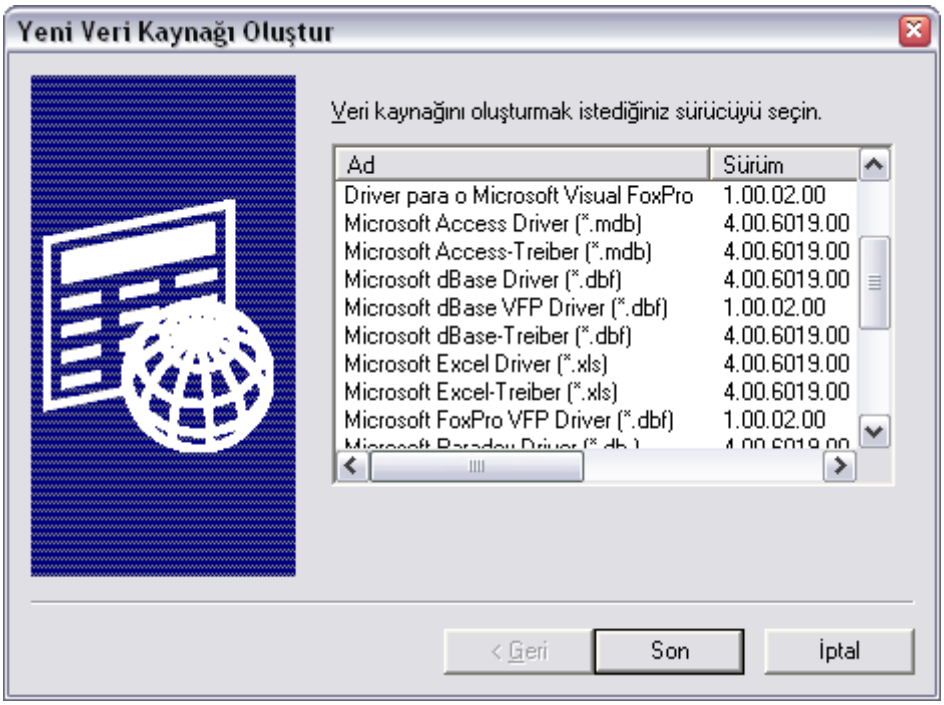
```
Provider=MSIDX; Data Source=Web
```

Örnek 166 : Index Bağlantı Örneği

Şeklinde bir bağlantı metni kullanılabilir. Bu şekilde "web" isimli bir index kataloguna bağlanmış oluruz.

3.7.ODBC Sürücüleri:

OLE DB kullanılan örneklerde yer alan "driver" özelliği veri kaynağına bağlantı tipini göstermektedir. Bu veri bağlantı sürücülerini (ODBC sürücüleri) yeni bir DSN bağlantı yaparken liste şeklinde görülmektedir.



Resim 9 : Yeni Veri Kaynağı Oluşturma Ekranı.

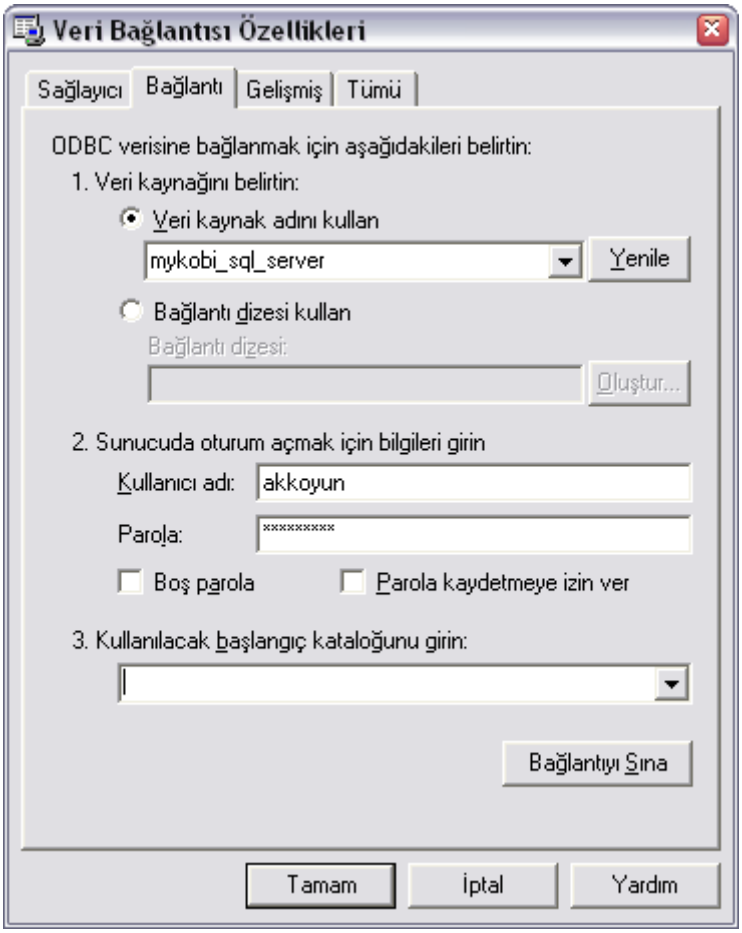
Bağlanmak istediğiniz veri kaynağının tipine bağlı olarak bir ODBC sürücüsü seçerek OLEDB bağlantısı kurabilirsiniz.

3.8.Data Link Files

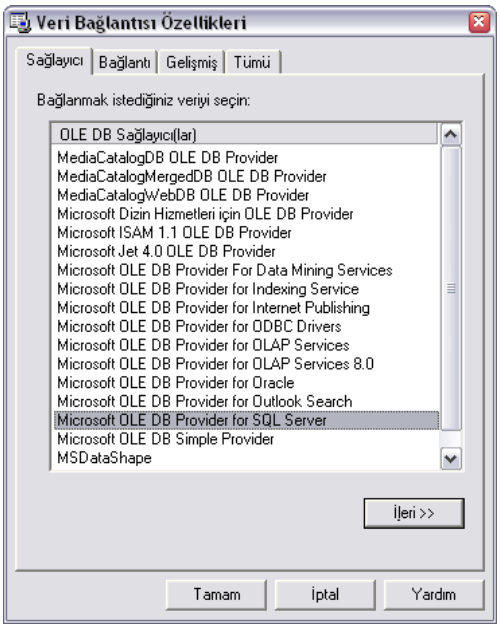
ADO'nun önceki versiyonlarında Explorer içerisinde sağ-tuş tıklayarak yeni data bağlantı dosyası oluşturulabiliyordu. Yeni data bağlantı dosyası oluştur komutu verildiği zaman otomatik olarak "Veri İlişkilendirme Sihirbazı" (Data Link Properties) penceresi açılmaktaydı. Bu yazı yazıldığı zamanlarda Microsoft işletim sistemlerinden sağ tuş ile veri bağlantı dosyası oluşturma fonksiyonunu iptal edilmişti. Yeni bir veri bağlantı dosyası oluşturmak için uzantısı ".udl" olan bir metin dosyası yaratabilirsiniz (not defteri yardımı ile yeni bir dosya yaratıp uzantısını udl şeklinde değiştiriniz). Veri bağlantı dosyasını oluşturduktan sonra dosyaya çift tıklayarak bağlantı özelliklerini görebilirsiniz.

Ekran görüntülerinden de görüleceği gibi (Resim 10 : Veri Bağlantısı Özellikleri. Ve Resim 11 : Veri Bağlantısı Özellikleri.) "mykobi_sql_server" adı altında oluşturulan bir MSSQL Server bağlantısıdır. Örnekte de görüleceği gibi kullanıcı adı "akkoyun" dur. Destekleyici tipini değiştirmek için data bağlantı özellikleri penceresinden sağlayıcı tabını seçiniz bu tab içerisinde yer alan destekleyici listesinden istediğiniz bağlantı tipini belirleyebilirsiniz.

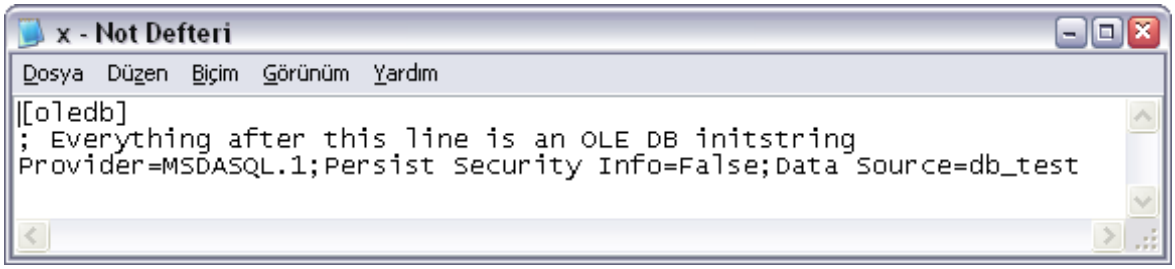
Sağlıklı bir bağlantı oluşturabilmek için data bağlantı dosyasına ait tüm özellikleri doğru olarak girilmelidir. Bunun dışında bu data bağlantı dosyasına herhangi bir metin editörü ile düzenlenebilmektedir.



Resim 10 : Veri Bağlantısı Özellikleri.



Resim 11 : Veri Bağlantısı Özellikleri.



Resim 12 : Bağlantı Dosyası.

Örnekte de gözlemlendiği gibi data bağlantı dosyasında sadece bağlantı metni yer almaktadır. Bu oluşturulan veri bağlantı dosyasına ASP sayfalarımızdan bağlanmak için:

```
Conn.Open "File Name = c:\data.udl"
```

Örnek 167 : Bağlantı Dosyası Yardımı ile Veritabanı Bağlantısı

Şeklinde bir komut kullanacağız.

3.9.ODBC Data Kaynakları

ODBC data kaynakları (genellikle "Data Source Name" olarak geçer ve DSN olarak kısaltılır) denetim masasında bulunan "veri kaynakları (ODBC)" penceresi yardımıyla ayarlanmaktadır. ASP sayfalarınızdan bir DSN e bağlanmak isterseniz o oluşturduğunuz DSN'nin "System DSN" olarak yapılandırılmış olmasına dikkat ediniz. Bu yapılandırma için "ODBC Veri Kaynağı yöneticisi"nden "Sistem DSN" sekmesini seçerek DSN i orada oluşturunuz. Açılacak olan sekmeden "Ekle" butonunu seçerek yeni bir DSN oluşturabilirsiniz.

ODBC veri kaynağı oluşturmak için hangi veritabanı tipine bağlanacaksanız o veri kaynağına ait bağlantı sürücüsünü seçerek ayrıntılı bilgileri doldurabilir ve veri kaynağınızı kendinize has ayarlarla oluşturabilirsiniz.

DSN'i ayarladıktan sonra ASP içerisinden bu kaynağa bağlanmak gerçekten çok kolaydır. Aşağıda bu bağlantı tipine ait bir örnek verilmiştir.

```
Conn.Open "DSN = veritabani"
```

Örnek 168 : DSN Veritabanı Bağlantısı

3.10. Bağlantı Dosyasını İçermek

Bağlantı dosyası içerme metodu temel olarak bağlantı satırının kendi başına bir dosyaya kaydedilmesine ve diğer sayfalarda (bağlantı yapılması istenilen sayfalarda) bu dosyanın "include" tagı ile o ASP ye dahil edilmesidir. Örneğin tüm bağlantı ayarlarını tek bir sayfada (mesela baglanti.asp) toplayarak diğer sayfalarda bu dosyayı içerebilirsiniz. Farzı mahal "baglanti.asp" dosyamız aşağıdaki gibi olsun.

```
<%
Conn="Provider=SQLOLEDB;Data Source=Akkoyun;
Initial Catalog=Makaleler;User ID=gunce;Password=12345"
%>
```

Örnek 169 : Bağlantı Dosyası

Bu dosya içerisinde sadece bağlantımıza ait opsiyonları belirterek sistemimizi daha fonksiyonel hale getirmiş olduk şimdi de bağlantı yapmak istediğimiz dosyalarda bu dosyayı nasıl içereceğimizi gösteren bir örnek verelim.

```
<!-- #INCLUDE FILE = "baglanti.asp" -->
```

Örnek 170 : Bağlantı Dosyasını İçermek

Bu şekilde her veri tabanı bağlantımızda teker teker bağlantı kurmak zorunda değiliz çünkü "baglanti.asp" bunu bizim yerimize yapmaktadır. Bu yöntemin en önemli faydası da bağlantıların tek bir yerden kontrol edilmesine olanak verdiği için herhangi bir ayar değişiminde uzun uzun tüm sayfaların düzenlenmesi yerine sadece "baglanti.asp" sayfasının düzenlenmesi ile diğer sayfaların otomatik güncellenmesidir. Yani merkezi ayar yapılmasına olanak tanımasıdır.

3.11. Global.ASA'da tanımlama

Bağlantı metnimizi merkezileştirmenin bir diğer yolu da "Application" nesnesi içerisinde saklanarak tüm sezon boyunca kullanılmasıdır. Bunun içinde bu tanımlamaları sezonun başlangıcında her zaman otomatik olarak çalıştırılan "Global.ASA" dosyası içerisinde yapmaktır. Bilindiği üzere bir web sayfasına gelen URL isteği sunucu tarafından çözümlenir ve otomatik olarak o istekte bulunan kullanıcıya bir sezon açar. Bu sezon açılımında ilk olarak yukarıda adı geçen "Global.ASA" dosyası işletilir. Bu sebepten dosya içerisinde tanımlanan her değişken tüm kullanıcılar için geçerli olmuş olacaktır. Örneğin aşağıdaki kodu Global.ASA içerisine yerleştirdiğiniz zaman tüm kullanıcılar için geçerli bir bağlantı metni tanımlaması yapmış olacaksınız.

```
Sub Application_onstart()

Conn = Provider=SQLOLEDB;
```

```
Data Source=Akkoyun;
Initial Catalog=Makaleler;
User ID=gunce; Password=12345

Set Application("baglanti") = Conn

End Sub
```

Örnek 171 : Global Olarak Bağlantı Değişkeni Tanımlamak

Bu şekilde bir global.asa dosyası bulunan sistemde her asp sayfası aşağıda şekilde bu tanımlamayı kullanabilir.

```
<%

Set Con_Mak = Server.CreateObject("ADODB.Connection")
Con_Mak.Open Application("baglanti")

%>
```

Örnek 172 : Global Bağlantı Değişkenini Kullanmak

Kişisel görüşüme ve deneyimlerime göre bu yöntemin bağlantı dosyası içerme yöntemiyle kıyasladığım zaman dosya içerme metodunun daha sağlıklı olduğunu söyleyebilirim çünkü global.asa ya yazılan her tanımlama göz atıcı (browser) kapatıldığı zaman kapatılacak ve her yeni sayfa içinde bunun tersi olarak bir uygulama başlatılacaktır bu da sunucuya işlem yapma gereksinimi gösterecek ve sisteminizi yavaşlatacaktır. Yazdığım bir çok ASP scriptinde bu yöntemin sistemi daha fazla yorduğunu gözlemledim. Tabi sizde kendi durumunuza göre bağlantı tipi seçebilirsiniz (Mission Critical).

3.12. Bağlantı yazımı

Teoride ismi bağlantı nesnesi olarak gözükmesine rağmen "Connection" (bağlantı) objesi sadece "open" methodu ile bir bağlantı açabilir. Aşağıda bu methoda ait yazım kuralı verilmiştir.

Connection.Open [Bağlantı_metni], [Kullanici_adi],[Parola],[Opsiyonlar]

Yukarıda belirtilen terimler hakkında biraz bilgi vermek istiyorum çünkü ADO içerisinde yer alan en önemli metodlardan birisidir Open.

[Bağlantı_metni] : bu yukarıda adı geçen ve hangi veritabanına nasıl bağlantı yapacağımızı belirten bir metindir. Bu metin standart bir bağlantı metni olabilir, bir DSN olabilir veya bir Data Link File olabilir.

[Kullanıcı_adi] : bağlantı sırasında kullanmak istediğiniz kullanıcıya ait tanımlama adıdır. Eğer DSN bağlantı kullanıyorsanız DSN de kullandığınız kullanıcı adını kullanmalısınız. Dip not olarak belirtelim kullanici_adi bağlantı sırasında zorunlu bir opsiyon değildir ama kullanıcı adı ve parola gerektiren bağlantılarda kullanmanız zorunludur.

[Parola] : kullanıcı adında tanımlı olan kullanıcıya ait paroladır.

[Opsiyonlar] : bu genelde bağlantı eşzamanlı olduğu zaman kullanılır. Bu opsiyonları ileriki sayılarda ayrıntılı inceleyeceğiz.

Not : eşzamanlı bağlantı ASP içerisinde kullanılamaz, ta ki scripting dilleri ADO dan olayları alana kadar.

3.13. Bağlantı Örnekleri

Yukarıda ve önceki saylarımızda bahsettiğimiz bağlantı metni ve "Connection.Open" methoduna ait örnekler verelim ve konumuzu biraz daha olsun pekiştirelim.

Bir bağlantı açmak için "Connection" objesinin "Open" methodunu kullanmanız gerekmektedir. Örneğin.

```
<%
Set Con_Mak = Server.CreateObject("ADODB.Connection")

Con_Mak.Open Baglanti_Metni
..
..
Con_Mak.Close
%>
```

Örnek 173 : Open.Connection ile Veritabanı Bağlantısı

Alternatif olarak "ConnectionString" özelliğini de kullanabilirinsiniz. Örneğin.

```
<%

Set Con_Mak = Server.CreateObject("ADODB.Connection")

Con_Mak.connectionstring Baglanti_Metni

Con_Mak.Open
..
..
Con_Mak.Close

%>
```

Örnek 174 : ConnectionString Kullanımı

İki örnek arasında fark yoktur sadece Connection string özelliği kullanılmış oldu. Böylece karışık bağlantılarda karmaşa ortadan kaldırılmış olur.

4. EK-1 : URL'ler Hakkında Ayrıntılı Bilgi

URL'ler İnternet protokollerini kullanarak erişilebilen kaynakları tanımlamaya yarar. URL'leri daha iyi anlamak için URL şemalarını ve formatlarını, URL'lerin nasıl tanımlandığını ve URL'lerde escape kodlarının nasıl kullanılacağını öğrenmelisiniz.

4.1.URL şemaları ve formatları

URL'lerin bu kadar yetenekli olmasını sağlayan mekanizma standart isimlendirme şemalarıdır. URL şeması, protokol isminin ardından gelen bilginin formatını belirtmek için istemcinin kullanacağı protokolün ismini kullanır. Genellikle protokol isminden sonra iki nokta üst üste ve iki tane bölü işareti kullanılır. İki nokta üst üste karakteri ayırıcı olarak kullanılır. İki adet bölü işareti ise protokolün yaygın İnternet şema formatı tarafından tanımlanan formatı kullandığını gösterir.

Yaygın İnternet şema formatı, İnternet protokolü tabanlı protokollerin doğrudan kullanılmasına imkan tanıyan URL şemaları için ortak söz dizimidir. IP tabanlı protokoller, IP adres ismi verilen sayısal tanımlayıcı (identifier) yada IP adresine dönüştürülebilecek bir isim ile internet üzerinde bir evsahibi (host) bilgisayarı belirlerler. Çift bölüm işaretini takip eden kısım ise URL'de işaret edilen protokole bağlıdır. Genel olarak URL formatı aşağıdaki şekildedir.

```
Protokol://evsahibi_adi:port/kaynağin_yolu
Protokol://kullanici_adı:şifre@evsahibi_adi:port/kaynağin_yolu
```

Not: Normalde DOS / Windows tabanlı uygulamalarda sistem içinde dizin veya dosya yerleri tanımlarken ters bölü işaretini kullanıyorduk. Fakat internetin UNIX sistemine uyduğuna (UNIX içinde yol belirtimi normal bölüm işaretleri kullanılarak yapılmaktadır) kendimizi alıştırmalıyız. Yani URL içerisinde her zaman normal bölü işareti kullanacağız.

4.2.URL'lerde host ile ilgili bilgiyi tanımlama

URL'lerin ilk bölümünden sonra (protokol ve ayrım işaretlerinden sonra) domain (alanadı) bilgisi gelmektedir. Alan adları dosyalarımızın bulunduğu sunucuyu işaret etmek için kullanılır. Alan adlarını gerçek hayatda kullandığımız adreslere benzetebiliriz. Örneğin "Köylüler Sok. Başak Apt." şeklinde yazılan bir ev adresinden, o evin Köylüler sok. İçerisindeki Başak apartmanında olduğunu anlıyoruz. Domain adresi de aynıdır yani "www.akkoyun.net" şeklinde girilmiş bir domain adresine karşılık gelen IP adresinin bulunduğu sunucularda (DNS karşılaştırması ile 212.98.198.111 nolu IP ye eşlenmiştir) bulunmaktadır demektir (ayrıntılı bilgi için bölüm 1 e bakınız).

Nasıl her yazdığımız adresin sonuna şehirde ve ilçe bilgisi giriyorsak domain adlarında da bu geçerlidir. Yani domain adresimizden sonra gelen uzantı bu ismin hangi sektöre ve hangi ülkeye ait olduğunu belirtir.

```
Com       Commercial (Ticari kuruluşlar)
Net       Network (Bilgisayar ağı merkezleri)
Org       Organization (Organizasyon siteleri)
Gov       Government (Devlet kurumları)
Mil       Military (Ordu kuruluşları)
Edu       Education (Eğitim kuruluşları)
```

Bu uzantılardan sonra gelen (nokta ile ayrılarak) kısımda ise ülke bilgisi yer almaktadır. Örneğin www.microsoft.com.tr adresinde firma isminin Microsoft olduğunu ve ticari bir kuruluş olduğunu anlıyoruz. En sonda kullanılan "tr" ibaresinden de bu kuruluşun Türkiye'de faliyet gösterdiğini belirtir. Bu isimler DIN standartlarında

belirtilmiştir. Biz bu ülke kodlarına TLD (Top Level Domain) diyoruz. Firma ismiyle beraber kullanılan domainlere ise SLD (Second Level Domain) diyoruz.

4.2.1. Top Level Domain Ulke Kodları

		Cuba	CU
Afghanistan	AF	Cyprus	CY
Albania	AL	Czech Republic	CZ
Algeria	DZ		
American Samoa	AS	Denmark	DK
Andorra	AD	Djibouti	DJ
Angola	AO	Dominica	DM
Anguilla	AI	Dominican Republic	DO
Antarctica	AQ		
Antigua and Barbuda	AG	East Timor	TP
Argentina	AR	Ecuador	EC
Armenia	AM	Egypt	EG
Aruba	AW	El Salvador	SV
Australia	AU	Equatorial Guinea	GQ
Austria	AT	Eritreta	ER
Azerbaijan	AZ	Estonia	EE
		Ethiopia	ET
Bahamas	BS		
Bahrain	BH	Falkland İslands	FK
Bangladesh	BD	Faroe İslands	FO
Barbados	BB	Fiji	FJ
Belarus	BY	Finland	FI
Belgium	BE	France	FR
Belize	BZ	French Guinea	GF
Benin	BJ	French Polinesia	PF
Bermuda	BM	French Southern Terrriotories	TF
Buhutan	BT		
Bolivia	BO	Gabon	GA
Bosnia and Herzegovina	BA	Gambia	GM
Botswana	BW	Georgia	GE
Bouvet İsland	BV	Germany	DE
Brazil	BR	Ghana	GE
British İndia Ocean Territory	IO	Gibraltar	GI
Brunei Darussalam	BN	Greece	GR
Bulgaria	BG	Green Land	GL
Burkina Faso	BF	Grenada	GR
Burundi	BI	Guam	GU
		Guatemala	GT
Cambodia	KH	Guinea	GN
Cameroon	CM	Guinea Bissau	GW
Canada	CA	Guyana	GY
Cape Verde	CV		
Cayman İslands	KY	Haiti	HT
Central African Republic	CF	Heard İsland and MCDonalds	HM
Chad	TD	Holy Sea	VA
Chile	CL	Honduras	HN
China	CN	Hong Kong	HK
Christmas İslands	CX	Hungary	HU
Cocos İslands	CC		
Colombia	CO	İceland	IS
Comoros	KM	İndia	IN
Congo	CG	İndonesia	ID
Congo (democratic republic)	CD	İran	IR
Cook İslands	CK	Iraq	IQ
Costa Rica	CR	İreland	IE
Cöte D'ivoire	CI	İsrael	IL
Croatia	HR	İtalia	IT
		Norfolk İsland	NF
Jamaica	JM	Northern Mariana İslands	MP
Japon	JP	Norway	NO
Jordan	JO		
		Oman	OM
Kazakstan	KZ		
Kenya	KE	Pakistan	PK
Kribati	KI	Palau	PW

Korea	KP	Palestinian	PS
Korea Republic	KR	Panama	PA
Kuwait	KW	Papau New Guine	PG
Kyrgyzstan	KG	Paraguay	PY
		Peru	PE
Lao Democratic Republic	LA	Phillippines	PH
Latvia	LV	Pitcairn	PN
Lebanon	LB	Poland	PL
Lesotho	LS	Portugual	PT
Liberia	LR	Puerto Rico	PR
Libyan	LY		
Liechstain	LI	Qatar	QA
Lithuania	LT		
Luxembourg	LU	Reunion	RE
		Romania	RO
Macau	MO	Russian Federation	RU
Makedonia	MK	Rwanda	RW
Madagascar	MG		
Malawi	MW	Saint Helena	SH
Malaysia	MY	Saint Kitts	KN
Maldives	MV	Saint Lucia	LC
Mali	ML	Saint Pierre	PM
Malta	MT	Samoa	WS
Marshall İslands	MH	San Marino	SM
Martinique	MQ	Sao Tome	ST
Mauritania	MR	Saudi Arabia	SA
Mauritus	MU	Senegal	SN
Mayotte	YT	Seychelens	SC
Mexico	MX	Sierra Leone	SL
Micronesia	FM	Singapore	SG
Moldova	MD	Slovakia	SK
Monaco	MC	Slovenia	SI
Mangolia	MN	Solomons İslands	SB
Montserrat	MS	Somalia	SO
Morrocco	MA	South Africa	ZA
Mozambique	MZ	South Georgia	GS
Myanmar	MM	Spain	ES
		Sri Lanka	LK
Namibia	NA	Sudan	SD
Nauru	NR	Suriname	SR
Nepal	NP	Svalbard	SJ
Netherlands	NL	Swaizland	SZ
Netherlands Antilles	AN	Sweeden	SE
New Caledonia	NC	Switzerland	CH
New Zealand	NZ	Syrian Arab	SY
Nicaragua	NI		
Nıger	NE		
Nigeria	NG		
Niue	NU		
Taiwan	TW	United States	US
Tajicistan	TJ	United States Minor Outlying	UM
Tanzai	TZ	Uruguay	UY
Thailand	TH	Uzbekistan	UZ
Togo	TG		
Tokelau	TK	Vanuatu	VU
Tonga	TO	Venezuella	VE
Trindat and Tobago	TT	Viet Nam	VN
Tunisia	TN	Virgin İslands Eng.	VG
Turkey	TR	Virgin İslands US.	VI
Turkmenistan	TM		
Turks and Coicos İslands	TC	Wallis and Futuna	WF
Tuvalu	TV	Western Sahara	EH
Uganda	UG	Yemen	YE
Ukraina	UA	Yugostlavia	YU
United Arab Emirates	AE		
United Kingdom	GB	Zambia	ZM
		Zimbabwe	ZW

4.3.URL'lerde port bilgisini tanımlama

Port numaraları bir binanın kapılarına benzer. Önceki örneklerde URL'yi adrese benzetmiştik port ise bu adres içinde geçen kapı numaraları gibi düşünülmelidir. URL ile evsahibi bilgisayara gelen veri yapılacağı işleme göre bir porta gider. Genellikle önceden tanımlanmış portlar kullanılır. Eğer port önceden tanımlanmamış ise URL satırındaki port adresine giriş yapılır. Örneğin göz atıcınızdan bir web sitesi adresi yazdığınız zaman sunucu bu adresin portuna otomatik olarak 80 değerini verir çünkü HTML dokümanları için default (varsayılan) giriş portu 80 numaralı portdur. Aşağıda önceden tanımlanmış protlar ve protokoller verilmiştir.

```
FTP         Port 21         File Transfer Protocol
Gopher      Port 70         Gopher Search Protocol
HTTP        Port 80         HTTP Protocol
NNTP        Port 119        Network News Terminal Protocol
Prospero    Port 1525       Prospero Protocol
telnet      Port 23         Telnet Protocol
WAIS        Port 210        Wais Protocol
```

4.4.URL'lerde kullanıcı adı ve şifre tanımlama

Kullanılan bazı protokollerde gizlilik ve güvenlik için kişiye özel oturum açmak gerekebilir. Böyle durumlarda kullanıcı protokolü kullanan servislere login olacak ve ancak bu şekilde kullanabilecektir. Aynı şekilde HTTP protokolünde de bazı durumlarda kişiye özel giriş istenmektedir. Bu durumlarda kişinin şifre ve kullanıcı adını bir şekilde servera yollaması gerekir. Kullanıcı adı ve şifre belirlenmeden bir protokolle bağlantı kurmak için şu değerler önceden belirlenmiş olarak kullanılır : Kullanıcı adı olarak anonymous ve şifre olarak da kullanıcının e-posta adresi.

Telnet'te önceden belirlenmiş değerler yoktur. Şifre ve kullanıcı adını belirtmediğiniz taktirde bu bilgi sizden istenecektir. Bunu önlemek için kullanıcıların URL'lerde kullanıcı adı isimlerini ve şifrelerini belirleyerek. Otomatik olarak bağlanmalarını sağlayabilirsiniz.

URL metinleri içerisinde şifre ve kullanıcı adı belirtmek zorunlu bir durum değildir. Belirtilmediği taktirde sunucu bu işlemin şifre gerektirmeyen bir işlem olduğunu algılayıp ona göre işlem yapmasını sağlayacaktır. Ama bu işlem login (giriş) gerektiren bir işlemse şifre girilmemesi durumunda sunucu bunu sizden isteyecektir.

4.5.URL'lerde yol bilgisini tanımlama

URL'lerin son kısmında yer alan ve istenilen dokümanın virtual (sana) yolunu (path) veren kısmına virtual path denir. Virtual path Host için tanımlanan ana dizini (root) baz alınarak oluşturulmuştur. Sanal yol ile fiziksel yol birbirinden farklı terimlerdir. Fiziksel yolu dosyanın disk üzerindeki tam adresini verirken, sanal yol ise root dan sonraki dizinleri kapsar.

4.6.Tanımlanmış protokol şemaları

Protokol şemaları çoğu daha önceden bahsedilen iki gelen URL formuna uyar. CISS standartlarına uygun hazırlanmış protokol şemaları çift eğik çizgi kullanır. HTTP, NNTP, WAIS ve File CISS-uyumlu protokollerdir. CISS standartlarına uygun olmayan protokoller çift çizgi kullanmaz. Mail-to ve News uyumlu olmayan protokollerdir. Aşağıdaki tabloda her protokol için kullanılan URL şeması görülmektedir.

Protokol	Tanım	URL formatı
FTP	File Transfer Protocol	ftp://kullaniciadi:sifre@evsahibi:port/yol
Gopher	Gopher Protocol	Gopher://evsahibi:port/yol
HTTP	Hypertext Transfer Protocol	http://evsahibi:port/yol
MailTo	Electronik mail address	

Prospero	Prospero Directory Service	prospero://evsahibi:port/hso_adi;alan=deger
News	Usenet news	News:/mesaj_numarasi
NNTP	Network News Transfer Pro.	nntp://evsahibi/habergrubuadi
telnet	Uzaktan bağlanılan oturumlar	telnet:/kullaniciadi:sifre@evsahibi:port
WAIS	Wide Area Information Serv.	wais://evsahibi:port/veritabani
File	Yerel dosya	File://evsahibi/yol

Not : URL belirtimleri prospero protokolünü de tanımladığı halde genellikle prospero gözatıcılar tarafından doğrudan kullanılmaz. prospero, FTP arşivlerine erişmek için Archie'nin kullandığı bir protokoldür. prospero dizin hizmeti ile ilişkilendirilen biçimleyiciler hso_adı, alan ve değerdir. Hso_adı, prospero protokolü kullanarak alınacak evsahibine özel nesneyi tanımlar. Alan ve değer ise nesne hakkında daha fazla bilgi vermek için kullanılır.

4.7.URL'ler nasıl tanımlanır

URL'ler ASCII karakter kümesinde tanımlanan karakterlerden oluşur. URL belirtimi büyük ve küçük harflerin kullanılmasına imkan tanır. Bununla birlikte URL'lerdeki büyük harfler, küçük harf gibi yorumlanır. Örneğin aşağıdaki adreslerin hepsi aynı şeyi ifade eder.

```
http://www.akkoyun.net
HTTP://www.Akkoyun.Net
HTTP://WWW.AKKOYUN.NET
```

Ipucu : büyük harflerle yazılan URL'ler küçük harfle yazılmış gibi yorumlandığı için çoğu yayımcı URL'lerinde küçük harfler kullanılır. Web dokümanı ve nesne isimlerinin çoğu da küçük harfle yazılır.

URL'ler ASCII karakter kümesinde tanımlanmış metinler olduğu halde. URL'lerde tüm ASCII karakter kümesini kullanamazsınız. A-Z arasındaki harfler ile 0-9 arasındaki rakamları aşağıda belirtilmiş olan özel karakterleri kullanabilirsiniz.

```
Yıldız (Asteriks) (*)
Dolar işareti ($)
Ünlem işareti (!)
Kısa çizgi (-)
Parantez (sağ ve sol)
Nokta (.)
Artı işareti (+)
Tek tırnak (‘)
Alt çizgi (_)
```

Bu karakterlerle sınırlısınız çünkü tabloda görüldüğü gibi URL'lerde diğer karakterlerin başka anlamaları vardır.

Karakter	Anlamı
:	İki nokta üst üste ayraçtır. Protokolü URL şemasının kalanından ayırır. Ev sahibi adını port numarasından ayırır. Kullanıcı adını şifreden ayırır.
//	Çift bölü işareti protokolün CISS tarafından tanımlanan şemayı kullandığını gösterir.
/	Eğik çizgi ayraçtır ve yol bilgisini ev sahibi ve portdan ayırmak için kullanılır. Eğik çizgi, URL'de belirtilen kaynağın bulunduğu dizinin yolunu göstermek için de kullanılır.
~	Tilda işareti genellikle yol bilgisinin başında bulunur. Kaynağı belirtilen kullanıcının halka açık html dizininda bulunduğunu gösterir.
%	Escape kodunu belirler. Normalde başka anlamlar taşıyan ya da başka şekillerde kullanılmasına izin verilmeyen özel karakterleri göstermek için kullanılır.
@	Bu işaret URL'de bulunan kullanıcı adı ve/veya şifre bilgisini ev sahibi adından ayırmak için kullanılır.
?	URL yolunda kullanılan soru işareti URL sorgulama katarının başlangıcını belirlemek için kullanılır. Sorgulama katarı CGI veya ASP programcıklarına gönderilen veri anlamına gelir. Soru işaretini takip eden tüm bilgi kullanıcının sunduğu verilerdir ve bu bilgi yol bilgisinin

devamı şeklinde yorumlanır.

+	Artı işareti sorgulama katarında sözcükler arasındaki boşluklar yerine kullanılır. Gözatıcı kullanıcının sorgulamada kullandığı sözcükleri ayırmak için boşluklar yerine artı işareti yerleştirir.
=	Sorgulama katarındaki değişkenlere veri atamak için kullanılır. Soru işaretinin solunda kalan kısım değişkeni sağında kalan kısım ise değişkene atanan veriyi temsil eder.
&	Sorgulama katarında kullanılan ampersand işareti, sorgulama katarındaki değişkenleri ayırır. Örneğin bir sorgu katarı içerisinde 3 değişken ve değer yollamak istiyorsak her değişken arasına bu işareti kullanmamız gerekir.
^	Sonraki kullanımlar için rezerve edilmiştir
{ }	Sonraki kullanımlar için rezerve edilmiştir
[ ]	Sonraki kullanımlar için rezerve edilmiştir

4.7.1. URL'lerde escape kodlarının kullanımı

URL'leri daha esnek ve kullanışlı hale getirmek için URL metinleri içerisinde escape karakterlerinin kullanılmasına imkan tanınmıştır. Escape kodları, başka işlemler için ayrılmış veya başka şekillerde kullanımına izin verilmeyen özel karakterleri URL'lerde kullanabilmek için yapılmışlardır.

Mehmet Günce Akkoyun

Bu şekilde girilen bir metni URL içinde escape kodları kullanmadan kullanmak sakıncalar getirmiştir. Boşluklar için escape kodları kullanılarak ismini tekrar yazmak gerekirdi. Yüzde işaretinden sonra gelen escape kodları sayesinde bu engel aşılmış olur. Boşluk için yüzde işareti ve ardından 20 sayısı gibi.

Mehmet%20Günce%20Akkoyun

ISO latin 1 karakter kümesi kullanarak atlamanız gereken karakterlerin değerini belirleyebilirsiniz. Bunun için ondalık karakter kümesindeki karaktere eş gelen sayıyı onaltılık değere çevirip kullanabilirsiniz. Aşağıda sıkça kullanılan escape karakterli bulunmaktadır.

Sayısal Değer	Karakter Karşılığı	Escape kodu
09	Sekme	%09
32	Boşluk	%20
35	Sayı işareti (#)	%23
37	Yüzde işareti (%)	%25
38	Amperdans (&)	%26
39	Kesme işareti (‘)	%27
63	Soru işareti (?)	%3f
64	Et işareti (@)	%40
95	Inceltme işareti (^)	%5f

İpucu : escape kodları kullandığımız zaman dikkate etmemiz gereken noktalardan bir tanesi sayısal değerlerin değil onaltılık sistemdeki escape kodlarının kullanılmasıdır. Bu sebeple 09 değerini 9 şeklinde kullanamayız.

5. Ek-2 : IIS Özellikleri ve Ayarları

5.1.Neden IIS?

IIS' ten bahsetmeden önce kısaca PWS (Personel Web Server) ile ilgili birkaç noktaya değinme gereği hissediyorum :

1. PWS, hepimizin bildiği gibi Windows 98 üzerinde çalışmaktadır. Windows ME ile beraber PWS gelmemektedir. Birkaç makine üzerinde yapmış olduğum denemelerde Windows 98 CD'si içerisinde yer alan PWS' yi Windows ME üzerinde kurduğumda hiçbir problemle karşılaşmadım (Uzun süreli kullanmadığımı belirtmek isterim). Ancak düzgün çalıştıramayan ve problem yaşayanların varlığını biliyorum (Windows ME ve PWS ile ilgili olarak "http://www.aspnedir.com/makaleler_icerik.asp?id=169" adresinde yer alan makaleye bakabilirsiniz).

2. Belki de çoğumuz ilk ASP kodlarını PWS üzerinde yazmıştır. Ancak Windows 98 ve PWS bize gerçek server ortamını sağlamadıkları için çoğu zaman yazdığımız kodları servera gönderdiğimiz zaman çalışmadığını, hata verdiğini görmüşüzdür.

3. Windows 98 bir server işletim sistemi olmadığı için, kullandığımız bileşenler (component) ile ilgili hatalar ve problemler oluşabilmektedir.

4. Eğer ASP ile sadece hobi olarak uğraşmıyorsanız, çalıştığınız ortamlarda Windows 2000 Server ile meşgul olacaksınız demektir. Bu nedenle Windows 2000 Server üzerinde çalışmak yararlı olacaktır.

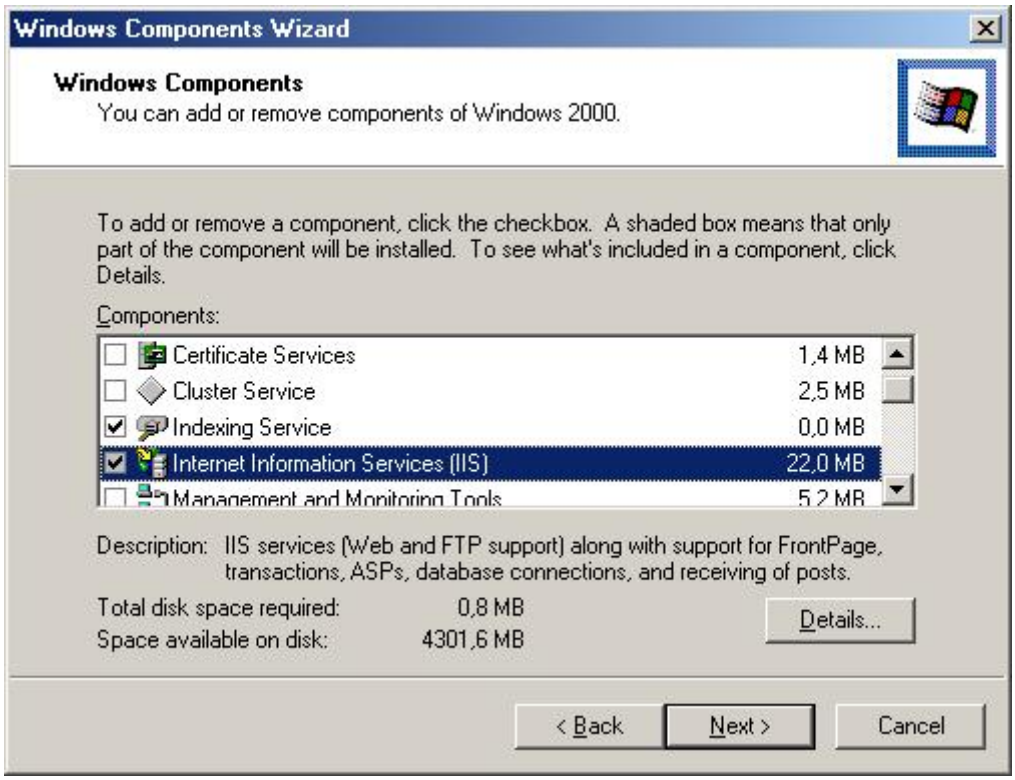
5. Server üzerinde çalışacak olan ASP kodlarının tüm geliştirme aşamasında benzer bir ortamda çalışmak size oluşabilecek hataları daha kolay görme, anında çözümler üretebilme gibi avantajlar sağlayacaktır. Yazdığınız kodun IIS üzerinde nasıl bir tepki vereceğini programlamayı tam olarak bitirmeden görme şansına sahip olabileceksiniz.

Yukarıda yazdığım sebeplerden dolayı tüm arkadaşlara kodlarını, eğer imkanları müsaitse Windows 2000 Server üzerinde veya en azından "Windows 2000 Professional Edition" üzerine IIS kurarak (kurulum sırasında default olarak yüklenmez) yazmalarını tavsiye ediyorum.

5.2.Kurulum

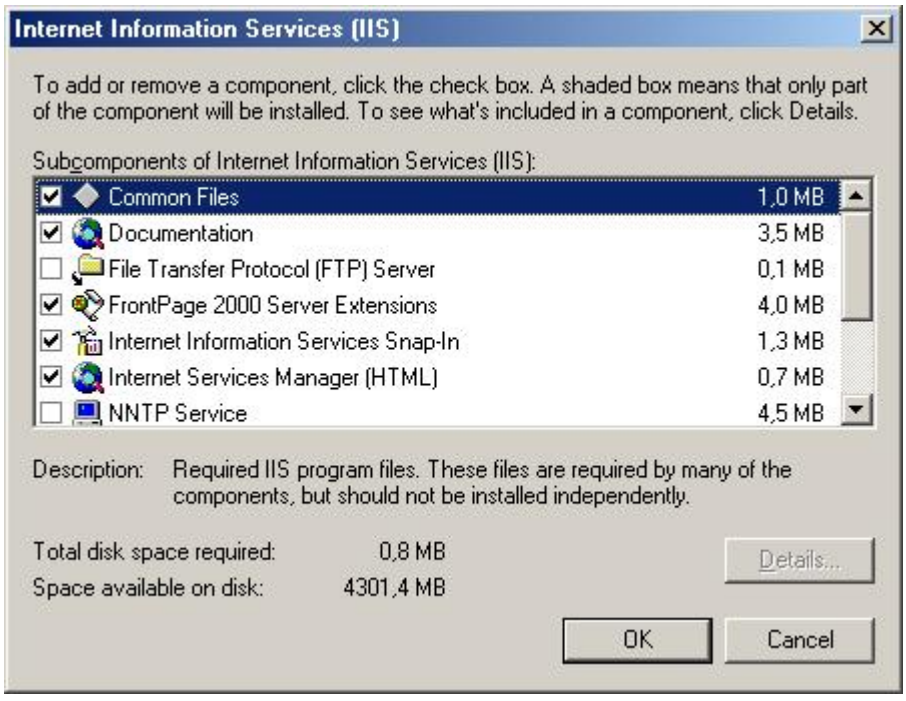
Windows 2000 Server üzerinde yüklenmemiş olma ihtimali ve Professional üzerine kurulması gerektiği için nasıl olduğunu bilmek amacıyla kısaca IIS' in kurulumundan bahsetmek istiyorum :

Start/Settings/Control Panel'e (Başlat/Ayarlar/Denetim Masası) tıklayarak Control Panel'i (Denetim Masası) açıyoruz. Control Panel'de Add Remove Programs (Program Ekle/Kaldır) linkine çift tıklayarak açılan pencerede "Add/Remove Windows Components"i (Windows Bileşenleri Ekle/Kaldır) tıklıyoruz. Karşınıza gelecek olan yeni pencerede Windows bileşenlerini görebilirsiniz :



Resim 13 : Windows Components Wizard

IIS linki üzerine çift tıklayarak veya "Details" linkine basarak IIS içerisinde yer alan bileşenleri (server ve servisler) görebilir, ihtiyacınız olanları işaretleyerek kurulmalarını sağlayabilirsiniz. Sıra ile bunlardan bahsedelim :



Resim 14 : IIS Alt Bileşenleri

Common Files : IIS'in çalışabilmesi için gerekli dosyalar. Bu dosyaların mutlaka kurulması gerekmektedir.

Documentation : IIS, Web ve FTP Serverlar üzerinden publish (yayın) ile ilgili yardımları ve örnekleri içeren dokümanlar.

File Transfer Protocol (FTP) Server : Dosya upload ve download işlemleri için gerekli olan servis.

Frontpage 2000 Server Extensions : Frontpage ve Visual Interdev aracılığı ile web siteleriniz üzerinde işlem yapabilmenizi sağlayacak olan araç.

Internet Information Services Snap-In : IIS için gerekli olan yönetim arabirimi.

Internet Services Manager (HTML) : Browser aracılığı ile IIS'i ve web sitelerinizi yönetmenizi sağlar.

NNTP Service : Açılımı Network News Transfer Protocol olan NNTP servisi vasıtasıyla server üzerinden haber grupları yayınlayabilirsiniz.

SMTP Service : Açılımı Simple Mail Transfer Protocol olan SMTP servisi aracılığıyla server üzerinden mail gönderebilirsiniz. (Bu servis ile sadece mail gönderebilirsiniz, mail alabilmek için bir mail Server'a ihtiyacınız vardır.)

Visual Interdev RAD Remote Deployment Support : Bu bileşen yardımıyla Server'ınız üzerindeki dosyalara Visual Interdev ile uzaktan bağlanılarak direkt üzerinde çalışılabilmesi için gerekli desteği sağlayabilirsiniz.

World Wide Web Server : En önemli bileşen. Server'ınız üzerinden web sitelerinin tüm dünyaya yayınlanabilmesi için gerekli servis.

İhtiyacınıza göre istediğiniz bileşenleri seçtikten sonra sırasıyla OK ve Next tuşlarına basıyoruz. Böylece IIS kurulumunu tamamlamış oluyoruz (Bu işlem için işletim sisteminin CD'sine ihtiyacınız olacaktır).

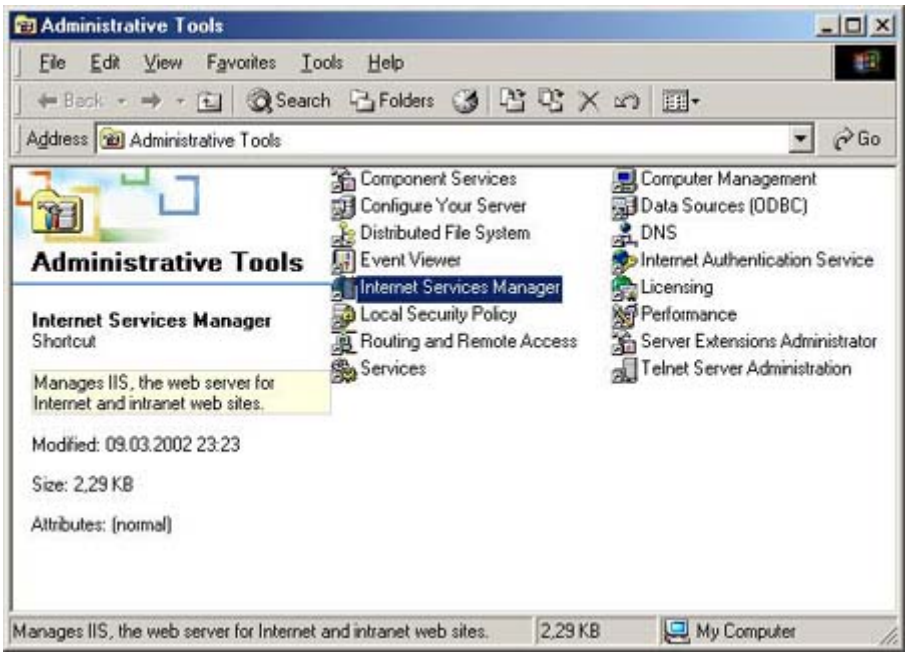
NOT : Windows 2000 Professional yukarıda saydığımız bileşenlerin tümünü içermez. FTP Server, Frontpage 2000 Server Extensions ve SMTP Service bileşenlerini içermektedir.

5.3.IIS'e Erişim

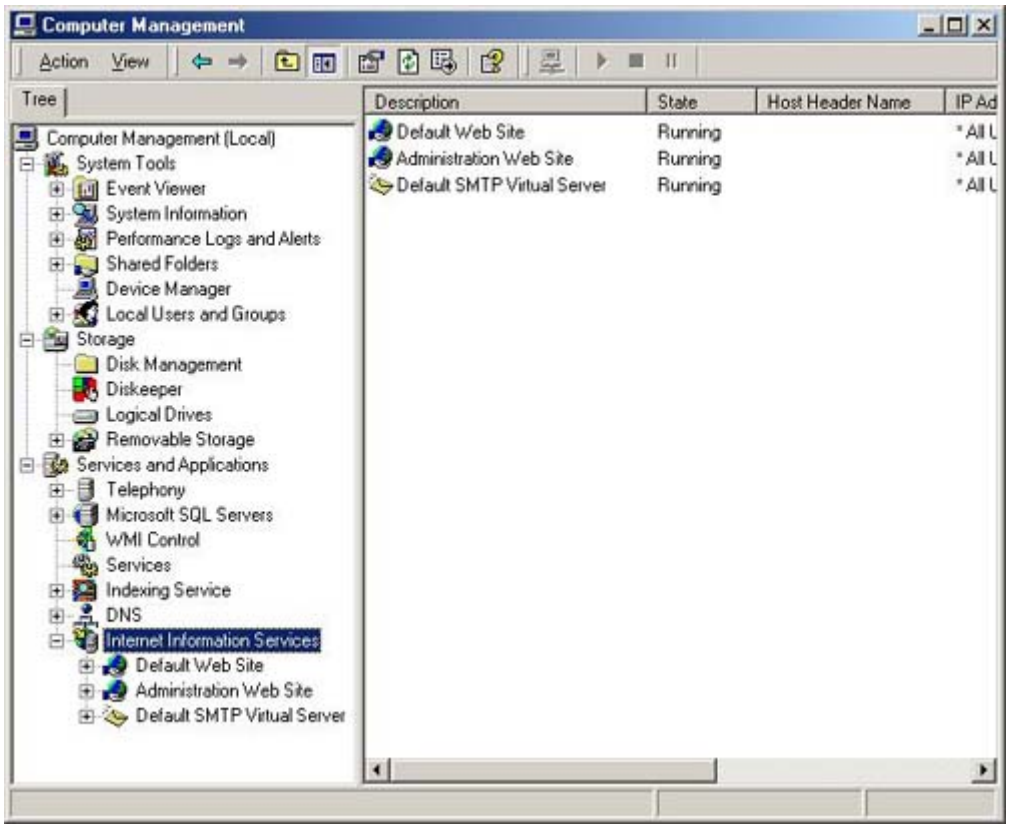
Web-FTP sitesi tanımlamak ve ayarlarını yapmak için IIS'e birkaç farklı yoldan ulaşabilirsiniz :

1. Start/Settings/Control Panel/Administrative Tools/Internet Services Manager yolunu izleyerek,

2. Start/Programs/Administrative Tools/Internet Services Manager yolunu kullanarak,



3. "My Computer"e sağ tıklayıp, "Manage" seçeneğine basarak açılacak olan "Computer Management" penceresinde "Services and Applications" bölümünden IIS'e ulaşabilirsiniz.



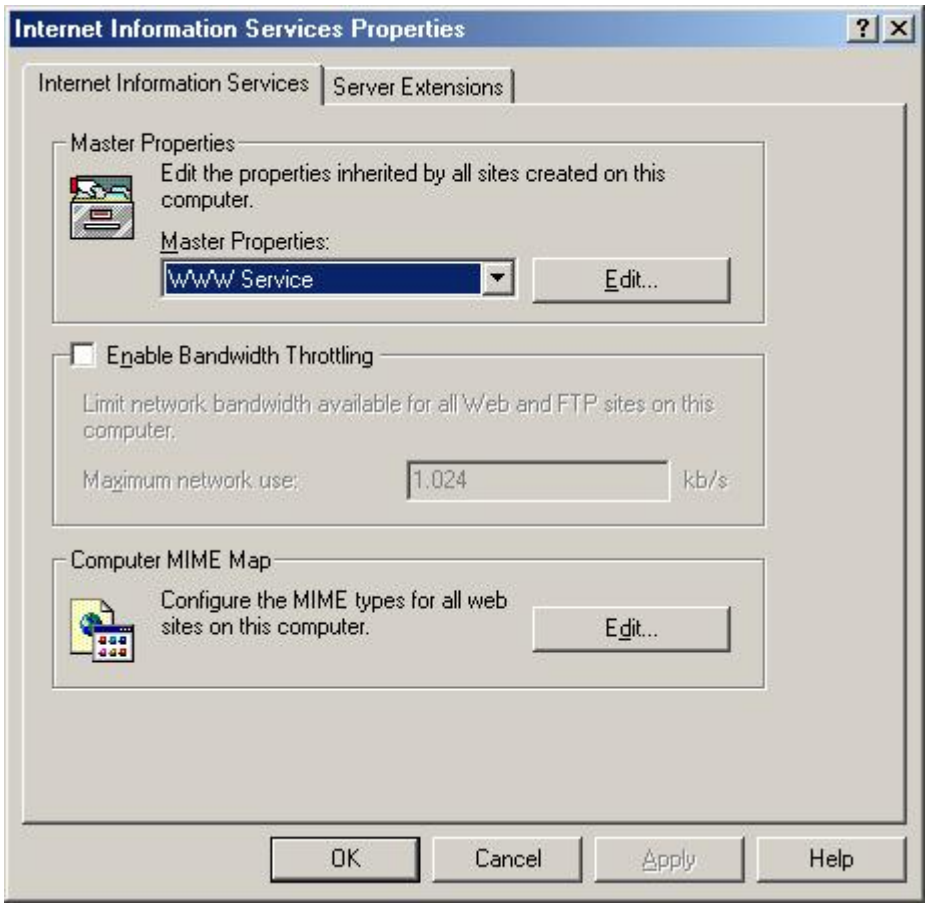
Resim 15 : Computer Management Console

Genelde üçüncü yolu kullanmayı tercih ediyorum, çünkü "Computer Management" penceresi vasıtasıyla IIS ile uğraşırken erişmem gerebilecek hemen tüm araçlara ulaşabiliyorum (Event Viewer, Local Users and Groups, Services, DNS servisi gibi).

5.4.IIS' de Genel Ayarlar

IIS'e ulaştıktan sonra hemen ayarları yapmaya başlayalım :

"Internet Information Services" linkine sağ tıklayıp, "Properties" seçeneğine basarak "Internet Information Services Properties" penceresini açıyoruz.



Resim 16 : IIS Genel Ayarları

Bu penceredeki seçenekler yardımıyla yapılacak olan ayarlar, o makine üzerinde yer alan tüm siteler için geçerli olacaktır.

5.4.1. Internet Information Services

Internet Information Services bölümünde Web ve FTP siteleri için genel ayarlar, bant genişliği sınırlaması ve dosya tipleri ile ilgili işlemler gerçekleştirilebilir.

5.4.1.1. Master Properties

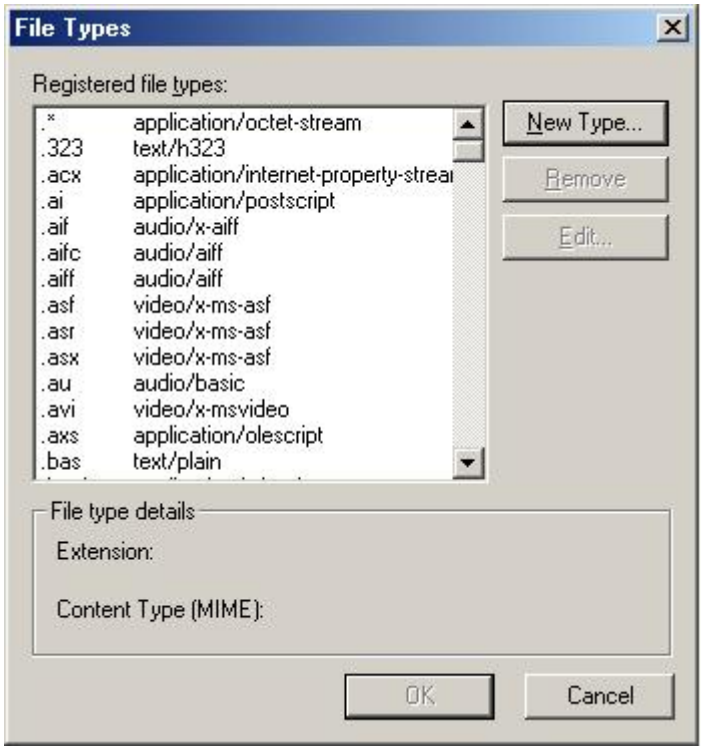
Bu bölümde WWW Service (Web Sitesi Yayınlama Servisi) ve FTP Service (FTP Sitesi Yayınlama Servisi) hizmetlerinin tüm sitelerinde geçerli olacak ayarlar yapılabilir. Her bir servis içi "Edit" tuşuna basılarak açılacak olan pencere yardımıyla ayarlar gerçekleştirilebilir. Bu ayarlardan daha sonra, Web sitesi ve FTP sitesi oluşturmayı anlatacağım bölümlerde bahsedeceğim.

5.4.1.2. Enable Bandwidth Throttling

Server üzerinde çalışacak olan tüm Web ve FTP siteleri için geçerli olacak bant genişliği sınırlaması gerçekleştirilebilir. Bu tür genel bir kısıtlama, bir veya birkaç adet sitenin yer aldığı kendinize ait bir server üzerinde uygulanmaz. Daha çok birçok sitenin barındırıldığı hosting firmaları için uygundur.

5.4.1.3. Computer MIME Map

Bu bölümde makine üzerindeki tüm Web siteleri için geçerli olacak ve Web siteleri tarafından kullanıcıya ulaşması istenen dosya tipleri belirlenebilir.

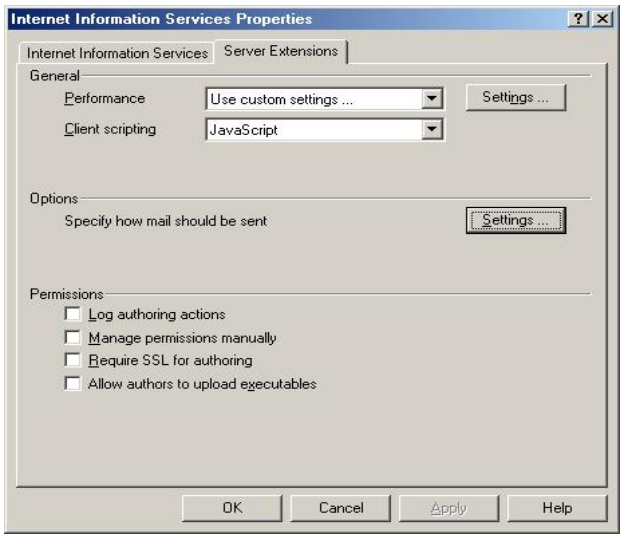


Resim 17 : MIME File Types

Web siteleri üzerinden ziyaretçiye iletilmesi istenmeyen dosya tipleri de yine bu bölümden kaldırılabilir.

5.4.2. Server Extensions :

Server Extensions bölümünde Frontpage Server Extensions ile hazırlanmış Web siteleri için ayarlar yapılabilir.



Resim 18 : Server Extensions Ayarları

5.4.2.1. General

Performance: Web sitesinde yer alacak sayfa sayısını belirtilerek veya özel ayarlar yapılarak en iyi performans alınması sağlanabilir.

Client scripting : Frontpage Server Extensions tarafından otomatik oluşturulacak istemci taraflı scriptler için kullanılacak olan dil Javascript veya VBScript olarak belirlenir.

5.4.2.2. Options

Specify how mail should be sent : Gerektiği durumda e-mail tabanlı web özelliklerini (e-mail form handler, ziyaretçiye mail göndermek) kullanırken gerekli olan mail ayarları "Settings" tuşuna basılarak yapılabilir.

5.4.2.3. Permissions

Log authoring actions : Bu kutucuk işaretlenerek Web sitesi üzerinde yapılan işlemlerin (dosya ekleme, silme gibi) kaydı (Log) tutulabilir. Bu loglar _vti_log klasörü altında yer almaktadır.

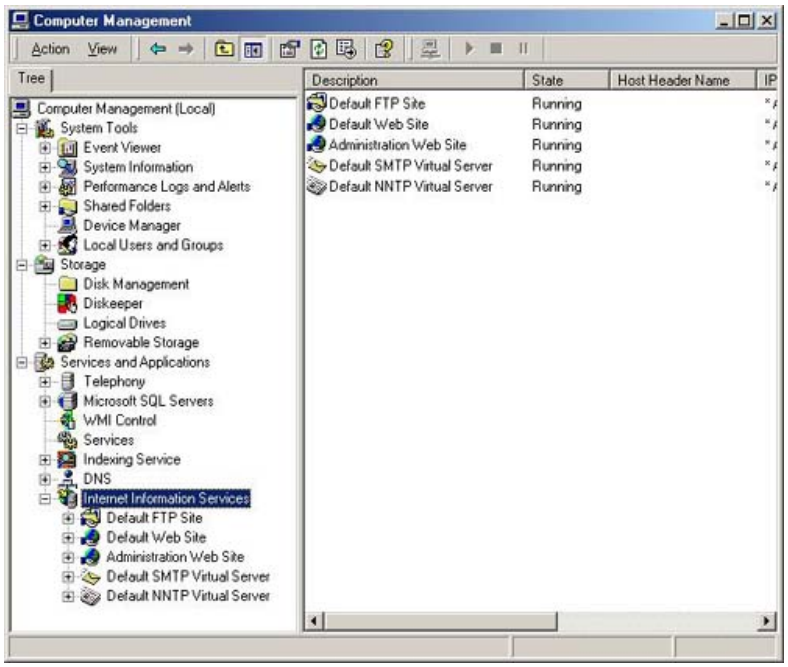
Manage Permissions Manually : Bu kutu işaretlenerek güvenlik ayarlarının Frontpage Server Extensions yerine elle yapılması sağlanabilir.

Require SSL for authoring : Web sitesine üzerinde dosya ekleme ve silme gibi işlemlerin gerçekleştirilmesi sırasında SSL ile güvenlik sağlanması için bu bölüm işaretlenmelidir.

Allow authors to upload executables : Web sitesine CGI veya ASP gibi scriptlerin veya diğer çalıştırılabilir dosyaların atılıp atılamayacağına bu bölümden karar verilebilir.

5.5.IIS' de Yer Alan Servisler

IIS'e bir önceki makalemizde bahsettiğimiz yollardan birisi ile ulaşıyoruz.



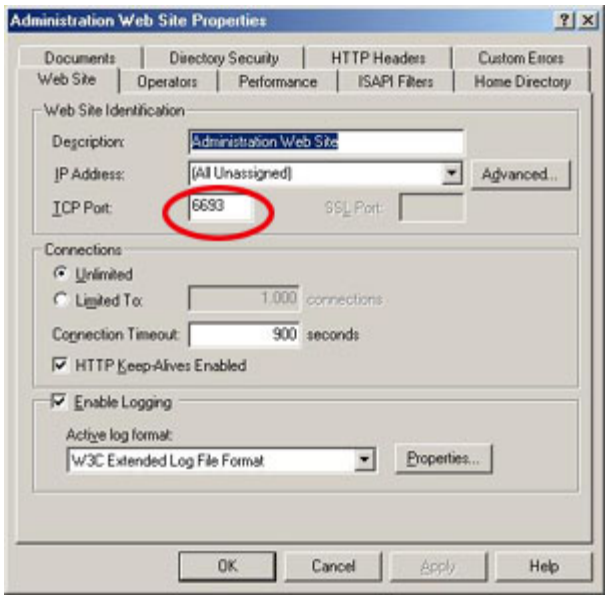
Resim 19 : Computer Management

Internet Information Services altında Default Web Site, Administration Web Site ve yaptığınız kuruluma göre Default FTP Site, Default SMTP Virtual Server, Default NNTP Virtual Server servislerini bulabilirsiniz.

Default Web Site, IIS'in kurulumu esnasında oluşturulur. Ana dizin olarak "inetpub/wwwroot" klasörü seçilidir. Eğer bu siteyi hiçbir ayarını değiştirmeden korur ve tüm web sitelerini "inetpub/wwwroot" klasörü altına yerleştirirseniz http://IP_Adresi/klasör_adı şeklinde ulaşabilirsiniz. Default Web Site, otomatik oluşturulmasına rağmen Microsoft tarafından kullanılması pek tavsiye edilmemektedir.

Administration Web Site'de yine IIS kurulumu esnasında oluşturulur. IIS ve oluşturacağınız tüm web ve FTP sitelerinin ayarlarını makinenin başında olmaksızın İnternet üzerinden yapabilmenizi sağlar. http://IP_Adresi:port_numarası şeklinde ulaşabilirsiniz.

Her server veya IIS kurulumunda Administration Web Site için Port değişmektedir. Port numarasını sağ tıklayarak açılan menüden "Properties" e basıldığında görüntülenen "Administration Web Site Properties" penceresinde yer alan "TCP Port" bölümünden öğrenebilir ve değiştirebilirsiniz.



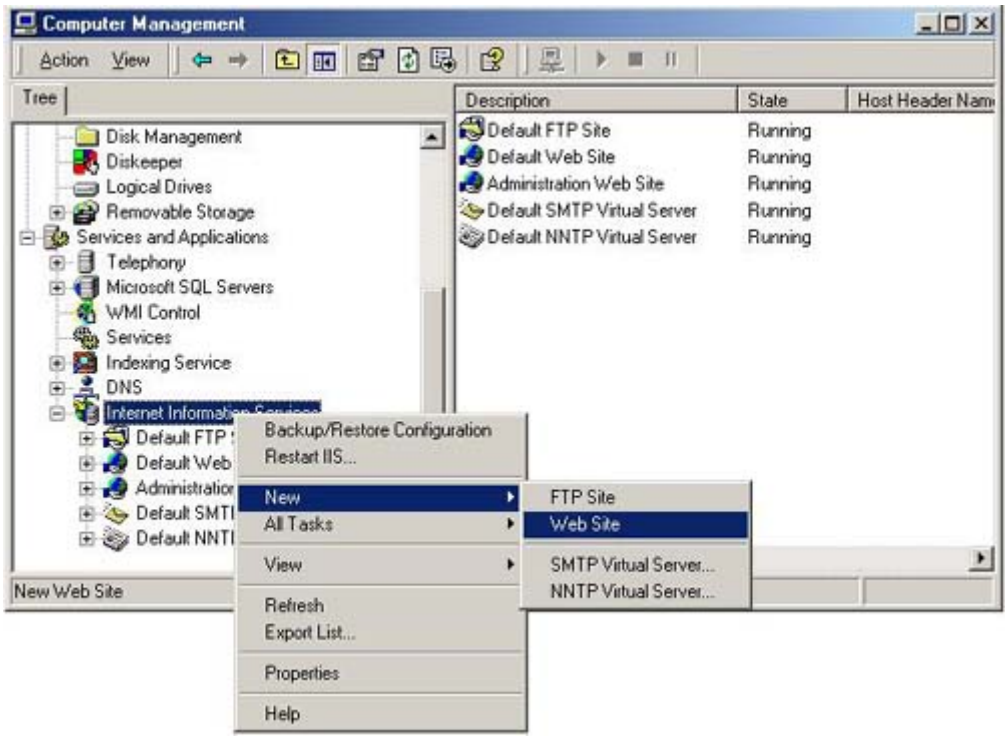
Resim 20 : Yönetim Arabirimi Port Numarası

Default SMTP Virtual Server ile serverdan herhangi bir mail server kurmaya gerek kalmadan mail gönderebilirsiniz. Bu servis ile SMTP aracılığıyla mail gönderebilirsiniz, mail alamazsınız.

Default NNTP Virtual Server , server üzerinden newsgroup hizmeti vermenizi sağlar.

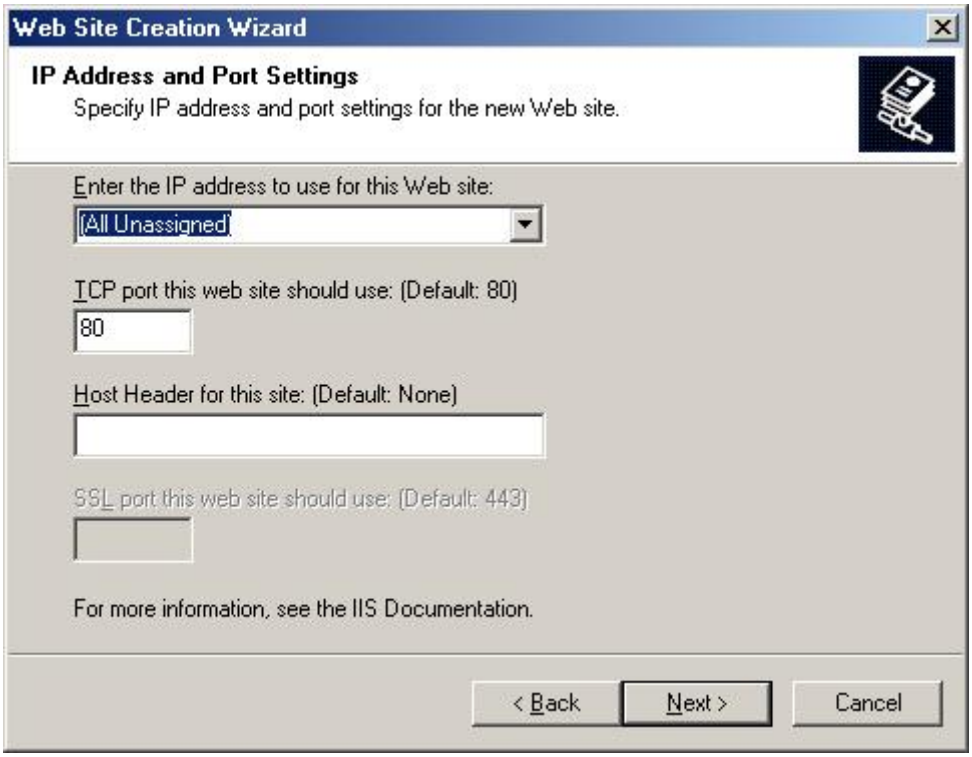
5.6.Web Sitesi Oluşturma

Web sitesi oluşturmak için, Internet Information Services'a sağ tıklayıp açılan menüden New >> Web Site 'a basıyoruz.



Resim 21 : Web Sitesi Oluşturmak

Açılacak olan "Welcome to the Web Site Creation Wizard" isimli sayfada "Next" tuşuna basıyoruz. "Web Site Description" sayfasında oluşturacağımız web sitesi için bir açıklama giriyoruz. Örnek : "Test Sitesi". Bu açıklama IIS' te görünecektir (Default Web Site gibi). Tekrar "Next" tuşuna basıyoruz. "IP Address and Port Settings" sayfasında web sitemiz için IP, Port ve header bilgilerini giriyoruz.



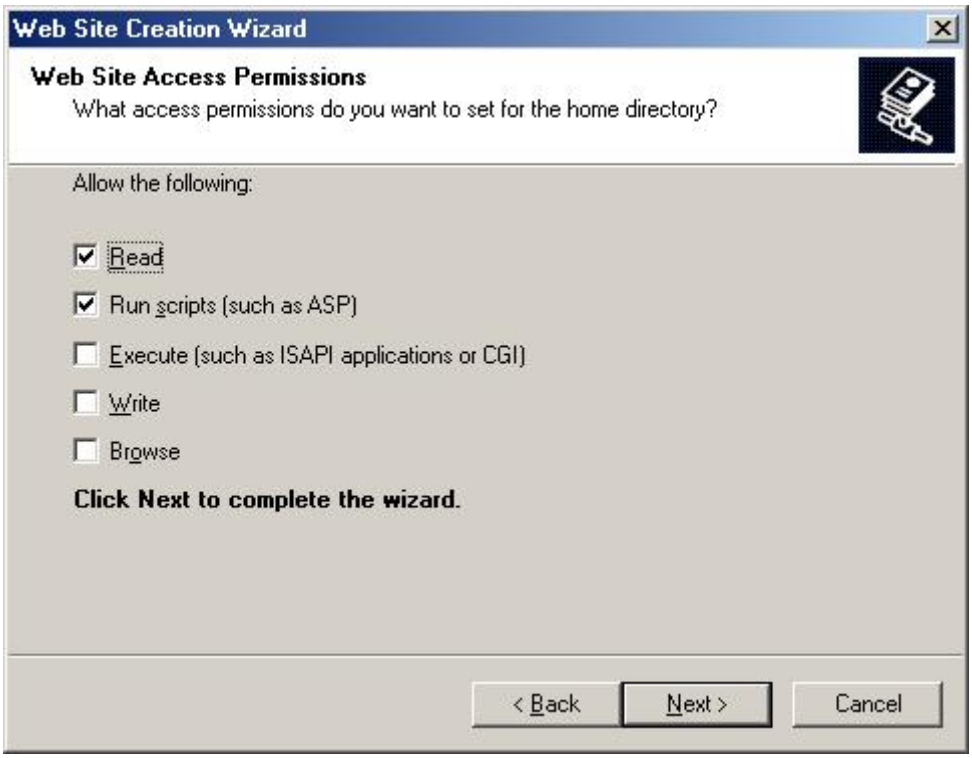
Resim 22 : Web Sitesi Oluşturma Sihirbazı

Enter the IP Address to use for this Web site : Oluşturduğumuz web sitesine ulaşılabilecek olan IP adresini burada tanımlıyoruz. Bu IP adresi, domainin DNS kayıtlarında yer alan IP adresi ile aynı olmalıdır.

TCP port this web site should use : Oluşturduğumuz siteye hangi porttan ulaşılacağını burada belirliyoruz. Varsayılan olarak bu değer 80 'dir. Normalde bu port değiştirilmez.

Host Header for this site : Oluşturduğumuz siteye ulaşılması için gerekli olan bilgiyi buraya yazıyoruz. Bu genelde www.siteadi.com şeklindedir. Eğer www'den farklı bir header kullanmak istiyorsanız öncelikle bunu DNS kayıtlarına geçirmeniz gerekir. Daha sonra da bu kısma yazabilirsiniz. Ör : test.siteadi.com

Tekrar "Next" tuşuna basıyoruz. "Web Site Home Directory" sayfasında sitemizin yer aldığı klasörün hard diskteki tam adresini ister elimizle giriyoruz ister "Browse" tuşu ile yerini buluyoruz. Adres kutusunun altında yer alan "Allow anonymous Access to this Web site" kutusu varsayılan olarak işaretlidir. Böylece İnternet üzerinden herkes web sitenize ulaşabilir. Tekrar "Next" tuşuna basıyoruz. "Web Site Access Permissions" sayfasında web sitesi üzerinde nelerin çalışabileceğini ve ziyaretçilerin web sitesi üzerinde yapabileceklerini belirliyoruz.



Resim 23 : Web Site Oluşturma Sihirbazı Yetki Basamağı

Read : Varsayılan olarak işaretlidir. Web sitesinde yeralan tüm sayfaların görüntülenmesi için gereklidir.

Run scripts (such as ASP) : Varsayılan olarak işaretlidir. Web sitesinde yer alan ASP sayfalarının çalışabilmesi için gereklidir.

Execute (such as ISAPI applications or CGI) : Oluşturduğumuz web sitesinde CGI ve ISAPI uygulamalarının çalışması için bu kutuyu işaretliyoruz.

Write : Oluşturduğumuz web sitesinde ziyaretçilere dosya yazma hakkı vermek için bu kutuyu işaretliyoruz. Dikkat : Normalde böyle bir izin verilmez!.

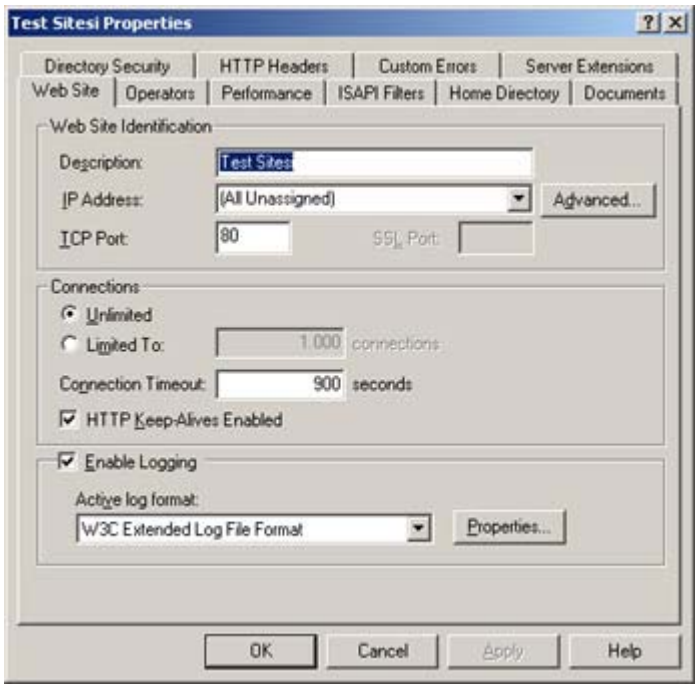
Browse : Oluşturduğumuz sitenin klasörlerinde ziyaretçilerin dolaşabilmesini sağlamak için bu kutuyu işaretliyoruz. Böylece ziyaretçiler sitede yer alan tüm sayfaları ve resim gibi nesneleri görebilirler. Dikkat : Normalde böyle bir izin verilmez!.

Tekrar "Next" tuşuna basarak sitenin oluşturulma aşamasındaki ayarları bitiriyoruz. Son sayfada "Finish" butonuna tıklayarak başarıyla web sitemizi oluşturmayı tamamlıyoruz.

5.7.Web Sitesinin Ayarları

Daha önce bahsettiğimiz yöntemlerden biri ile IIS'e ulaşıyoruz. IIS' de daha önce oluşturduğumuz "Test Sitesi" isimli web sitesinin üzerine gelip sağ tuşa tıklayarak Properties' e basıyoruz.

5.7.1. Web Site



Resim 24 : Web Site

Web Site Identification Description : Web sitesini oluştururken verdiğimiz ve IIS Manager'da görünen isim.

IP Address : Web sitesine ulaşılabilecek IP adresi burada tanımlanır. Makine üzerinde birden fazla ve IP adreslerine sahip site tutulacak ise bunlara ait IP'ler tanımlandıktan sonra bu bölümden ilgili IP adresi seçilir.

TCP Port : Web sitesine ulaşılabilecek port numarası. Varsayılan olarak 80'dir. Genelde bu port numarası değiştirilmez. Ancak özel durumlarda, özel amaçlar için değiştirilir. Port değiştirildiğinde siteye http://www.siteadi.com:port_numarasi şeklinde ulaşılabilir.

"Advanced" tuşuna basarak açılan "Advanced Multiple Web Site Configration" isimli pencerede "Multiple identities for this Web Site" bölümünde web sitesine birden fazla isim ile ulaşılması sağlanabilir. Aynı şekilde eğer IIS'e SSL sertifikası eklenmiş ise "Multiple SSL identities for this Web Site" bölümünden web sitesine SSL ile güvenli bir şekilde ulaşılması sağlanabilir.

Connections : Bu bölümde "Unlimited" seçili durumda ise web sitesine bağlanan kişi sayısında bir sınırlama olmaz. "Limited To:" seçili durumda ise sadece belirtilen sayıda ziyaretçi kabul edilecektir.

Connection Timeout: Saniye cinsinden IIS'in aktif olmayan kullanıcılar ile bağlantıyı kesme zamanı. Ziyaretçi tarafında oluşan bir hata sonucu düzgün bir şekilde kapatılamayan bağlantılar belirtilen süre sonunda otomatik olarak kapatılır.

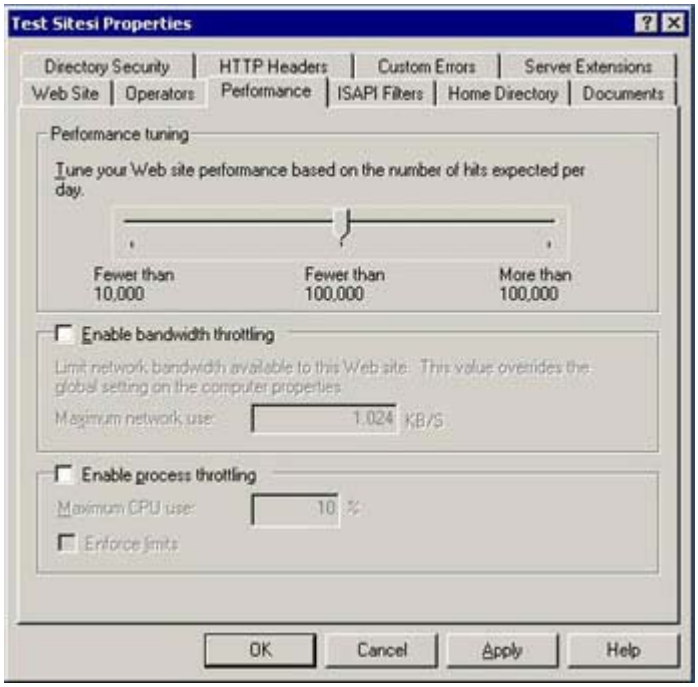
Enable Logging : Bu bölümü işaretleyerek web sitesine yapılan tüm ziyaretlerin kaydını tutabilirsiniz. "Active log format" bölümünden Microsoft IIS Log File Format, NCSA Log File Format, ODBC Logging ve W3C Extended Log File Format seçeneklerinden istediğinize uyanı seçerek kayıtları tutmaya başlayabilirsiniz. "Properties" tuşuna basarak seçmiş olduğunuz log tutma yöntemi ile ilgili ayarları yapabilirsiniz. Böylece hangi sayfaların ne kadar ziyaret edildiğini, web sitesinin en çok hangi saatlerde ziyaret edildiği vb. gibi bilgiler öğrenilebilir.

5.7.2. Operators

Bu bölümde IIS Manager yardımıyla web sitesine ulaşabilecek ve ayarlarına müdahale edebilecek kullanıcılar tanımlanabilir. Varsayılan olarak "Administrators" grubuna dahil olan kullanıcılar IIS Manager'dan web sitesinin ayarlarına müdahale edebilirler. IIS üzerinde tanımlı birden fazla siteden sadece bir tanesi için yetki verilmek istenilen kullanıcı bu bölümden eklenebilir.

5.7.3. Performance

Bu bölümde adından da anlaşılabileceği gibi web sitesinin ne kadar performans ile çalışabileceği belirlenebilir.



Resim 25 : Performance

Performance tuning : Günlük beklenen hit sayısına göre web sitesinin performansını ayarlamak için kullanılır. Beklenen hit sayısının biraz üzerinde ayarlanarak daha hızlı bir şekilde siteye ulaşılması sağlanabilir. Ancak günlük ziyaretçi sayısının çok üzerinde ayarlanması gereksiz bir şekilde Server'ın memory (hafıza)'sinin dolmasına ve serverın performansının düşmesine neden olmaktadır.

Enable Bandwidth Throttling : Web sitesinin kullanacağı bant genişliğini sınırlamak için kullanılır. Saniyede Kilo byte cinsinden ne kadar bant genişliği kullanabileceği belirlenir. Bu limit, eğer genel olarak tanımlanmış limitin üzerinde ise genel olan limit geçerli olacaktır.

Enable Process Throttling : Web sitesinin maksimum CPU kullanımına sınırlama getirmek için kullanılır. Birden çok web sitesinin tutulduğu Serverlarda düzgün bir çalışmanın sağlanması için gerekli olabilir. Böylece hatalı çalışan bir web sitesinin serverı iş görmez hale getirmesi önlenebilir. "Enforce limits" işaretlendiğinde web sitesinin kısa süreler için limiti aşması sağlanabilir.

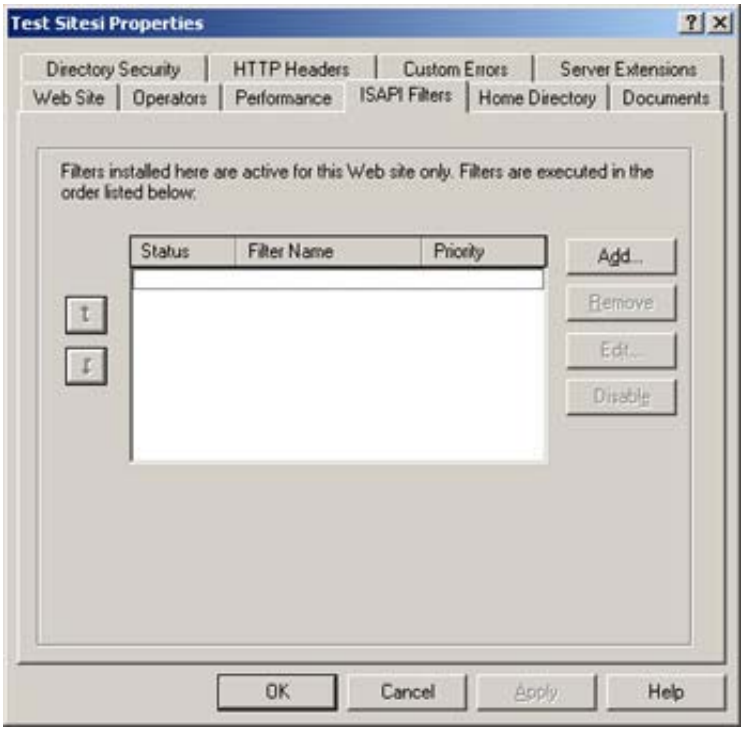
Limitlerin kontrol süresi 24 saattir. Her 24 saat sonrasında limit aşım kontrolleri sıfırlanır.

Örnek : Bir web sitesi %10 CPU kullanımı ile sınırlanmışsa ve 24 saat içerisinde, bu sürenin %10'u olan 2.4 saatlik sürede limitleri aşmışsa Event Log'a bir kayıt eklenir. Web sitesi limitini % 150 aşmış ise ve bu süre 3.6 saat toplamına erişmiş ise Event Log'a bir kayıt yazılır ve limiti aşmasını sağlayan işlemler engellenir. Web sitesi limiti % 200 aşmış ve bu süre toplam 4.8 saat veya üzeri olmuş ise Event Log'a bir kayıt yazılır ve Web sitesinin çalışması durdurulur.

5.7.4. ISAPI Filters

Açılımı Internet Server Application Programming Interface olan ISAPI Filtreleri, HTTP isteklerine cevap verebilen programlar olarak tanımlanabilir. ISAPI Filtreleri, DLL dosyalarıdır.

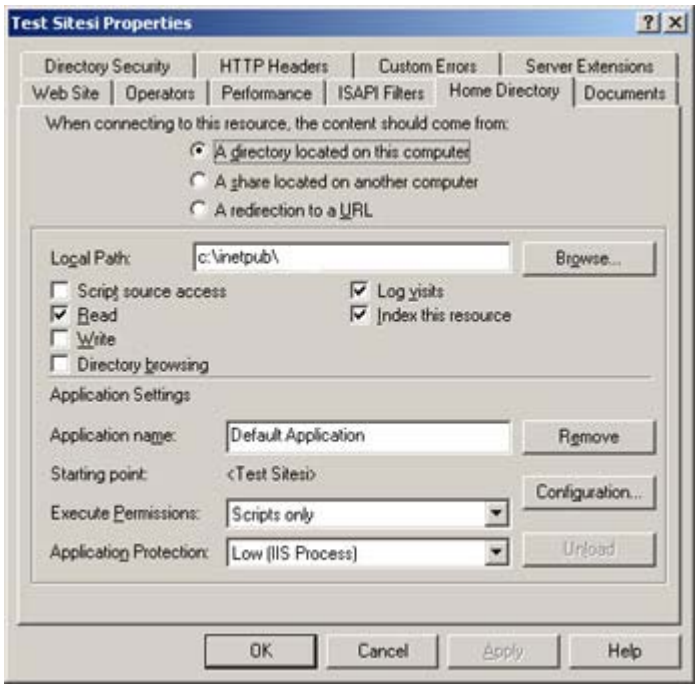
Add tuşunu kullanarak ISAPI Filtresi eklenebilir, Remove tuşu ile işaretlenen ISAPI Filtresi'ni silinebilir, Edit tuşu ile ISAPI Filtresi ile ilgili düzenleme yapılabilir. Sol tarafta yer alan ok tuşlarını kullanarak ISAPI Filtrelerinin çalışma öncelikleri değiştirilebilir.



Resim 26 : ISAPI Filters

5.7.5. Home Directory

Bu bölümde web sitesine ait dosyaların yer aldığı klasör bilgileri ve uygulama ayarları ile ilgili bölümler yer almaktadır.



Resim 27 : Home Directory

When connecting to this source, the connect should come from Bu kısımda web sitesinin dosyalarının nerede yer aldığı belirlenir. "A directory located on this Computer" varsayılan olarak seçilidir ve web sitesi tanımının yapıldığı makinede varolan dosyaların çalıştırılmasını sağlar. "A share located on another Computer" seçeneği ile aynı network üzerinde bulunan başka bir makinede yaralan dosyaların tanımlanan web sitesinde çalıştırılması sağlanır. "A redirect on to a URL" seçeneği yardımıyla da web sitesi başka bir web sitesine yönlendirilebilir.

Local Path : Bu seçenek "A directory located on this Computer" seçeneği işaretli olduğunda görüntülenir. Web sitesine ait olan dosyaların yer aldığı klasör seçilerek web sitesinin çalışması sağlanır.

Network Directory : Bu seçenek "A share located on another Computer" seçeneği işaretli olduğunda görüntülenir. "\\makina_adi\paylasim_adi" şeklinde diğer makinede yaralan dosyalara ulaşılabilir.

Redirect to : "A redirection to a URL" seçeneği işaretli olduğunda görüntülenir. "http://www.xxx.com" şeklinde başka bir siteye yönlendirme sağlanır.

Script source access : Bu seçenek işaretlendiğinde Read veya Write hakkı verilmiş olan kullanıcıların kaynak koduna ulaşması sağlanabilir. Normal şartlarda kullanılması tavsiye edilmez.

Read : Web sitesine bağlanacak kullanıcıların dosyaları okuması veya çalıştırabilmesi için verilmesi gerekir.

Write : Web sitesine bağlanan kullanıcıların dosyalar üzerinde değişiklik yapabilmesine, yeni dosya upload edebilmesine olanak sağlar. Normal şartlarda kullanılması tavsiye edilmez. Mutlaka write hakkı verilmesi gerekiyorsa ilgili klasörde bu hakkın verilmesi daha sağlıklıdır.

Directory browsing : Bu seçenek işaretlenerek web sitesine bağlanan kullanıcıların ilgili klasörde varolan dosyaların bir listesini görmesi sağlanır. Ancak bu klasörde default.asp, default.html, index.asp gibi klasör adresi verildiğinde çalışan dosyalar varsa klasör içeriği görüntülenemeyecektir. Gerekli olmadığı sürece bu seçeneğin aktif olması tavsiye edilmez.

Log visits : Bu seçenek varsayılan olarak işaretlidir ve ziyaretçilerin yaptıkları tüm işlemlerin kaydının tutulmasını sağlar.

Index this source : Bu seçenek yardımıyla Microsoft Indexing Service kullanılarak full-text search yapılabilmesi sağlanır.

"A redirection to URL" seçeneği işaretli olduğunda aşağıdaki seçenekler görüntülenecektir :

The exact URL entered above : Girilmiş olan tam adrese gidilmesini sağlar. Böylece web sitesinin yönlendirildiği sitede istenilen herhangi bir alt sayfadan başlanması sağlanabilir.

A directory below this one : Bu seçenek yardımıyla web sitesinin kendi içerisindeki herhangi bir alt klasöre yönlendirilmesi sağlanabilir. Ör : /yenisite yazıldığında www.test.com yazıldığında sayfa www.test.com/yenisite adresine yönlendirilecektir.

A permanent redirection for this source : Web sitesine bağlanan kullanıcıya yönlendirme ile ilgili bir mesaj gönderir.

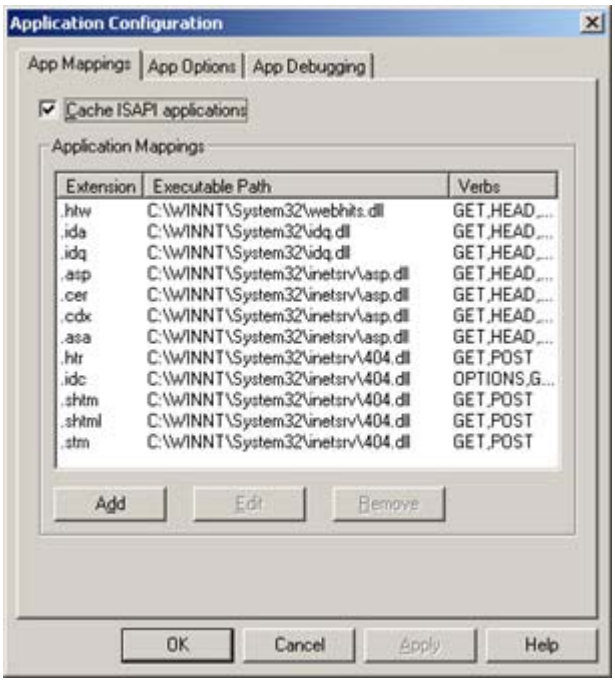
Application Settings

Application name : Web sitesinin bulunduğu uygulama bölümünün adı.

Execute Permissions : Bu bölümde çalıştırılabilecek dosya tipleri belirlenir. None işaretlendiğinde sadece statik HTML sayfaları ve resim dosyaları görüntülenebilir. Scripts only seçeneği işaretlendiğinde ASP gibi script dosyalarının çalıştırılması sağlanır. Scripts and Executables işaretli ise her türlü dosya çalıştırılabilir hale gelir.

Application Protection : Web sitelerinin yaptığı işlemlerin nasıl gerçekleştirileceği bu bölümde belirlenir. Low(IIS Process) işaretlendiğinde tüm işlemler aynı süreç içerisinde çalıştırılır. Medium(Pooled) seçeneği işaretlendiğinde işlemler application bazında ayrı olarak yürütülür. High(Isolated) seçeneği işaretlendiğinde ilgili web sitesinin işlemleri tamamen ayrı olarak yürütülür.

Configuration tuşuna basıldığında Application Configuration isimli pencere görüntülenir.



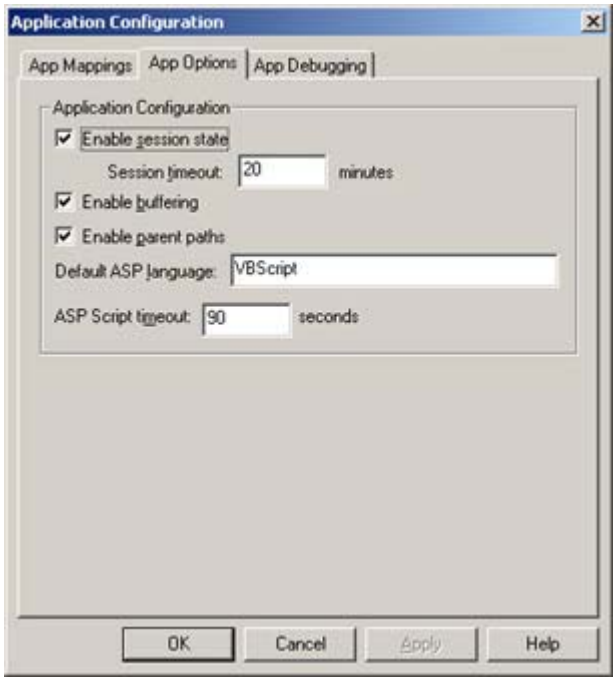
Resim 28 : App Mappings

App Mappings

Cache ISAPI applications : Bu seçenek işaretlenerek çalıştırılan ISAPI filtrelerinin cache'lenmesi ve daha sonraki kullanımlarda daha hızlı işlem yapılması sağlanır.

Applicaton Mappings : Bu listede hangi uzantıya sahip olan dosyanın hangi DLL ile ilişkili olduğu, hangi işlemlerin gerçekleştirilebileceği görüntülenmektedir. Yapılmak istenilenlere göre bu listede ekleme, çıkarma veya düzenleme işlemleri yapılabilir.

App Options :Bu bölümde ASP sayfalarının nasıl çalışacağı ile ilgili ayarlar yapılabilir.



Resim 29 : App Options

Enable session state : Varsayılan olarak işaretlidir. Web sitesinde session ile ilgili işlemlerin yapılabilmesini sağlar.

Session timeout : Dakika olarak session ile ilgili bilgilerin tutulacağı zamanı gösterir. Varsayılan olarak 20 dakikadır. Amaca uygun olarak değiştirilebilir. Bu süre script içerisinden Session.Timeout = xx olarak değiştirilebilir.

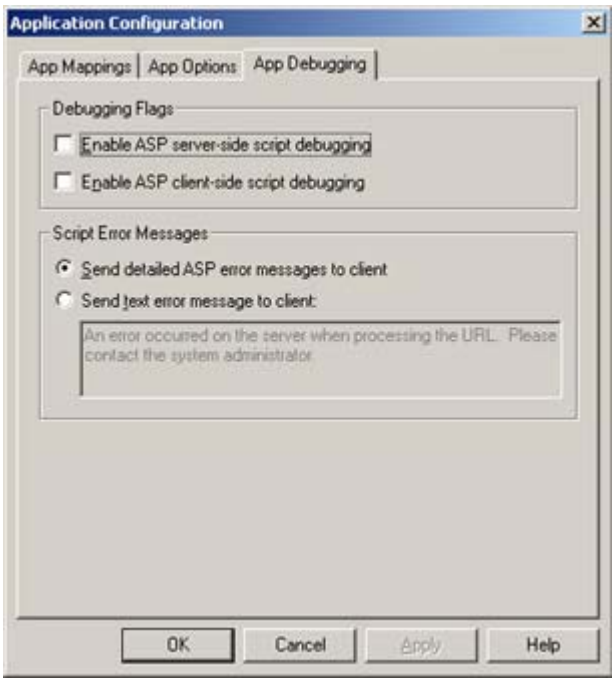
Enable buffering : Varsayılan olarak işaretlidir. Bu seçenek ASP sayfaların işletilmesiyle çıkan sonuçların tamponlanmasını sağlar. Seçenek işaretlendiğinde ASP sayfaları browser'e gönderilmeden önce tamponlanır. Böylece aynı istek tekrarlandığında daha hızlı yanıt verilebilir. Ancak zaman zaman sayfaların her istenildiğinde tekrar üretilmesi gerekebilir. Bu durumda bu seçenek deki işaret kaldırılır.

Enable parent paths : Bu seçenek işaretlenerek ASP dosyaları içerisinde göreceli yolların kullanılması sağlanabilir. Bulunulan klasörden üst klasördeki bir dosyaya ".." şeklinde ulaşılabilmesi sağlanır.

Default ASP Language : Web sitesinde ASP scriptlerinin yazımında kullanılacak olan script dilinin belirlenmesi için kullanılır. Varsayılan olarak VBScript tanımlıdır. Genel olarak dünya çapında kabul görmüştür. Farklı bir dil kullanılacaksa bu kısımdan değiştirilebilir. Eğer herhangi bir sayfada, sadece o sayfa için başka bir dil kullanılacaksa sayfanın ilk satırında yazılmalıdır.

ASP Script timeout : ASP kodlarının çalıştırılacağı zaman dilimini saniye cinsinden gösterir. Varsayılan olarak 90 saniyedir. Bu süre sonucunda ASP kodlarının çalışması sonlanmış olmalıdır. Eğer sonlanmamış ise kodların çalışması durdurulur, browser'da ilgili hata mesajı gösterilir ve Event Log'a kayıt eklenir. Sadece belirli bir sayfa için farklı bir timeout süresine ihtiyaç varsa bu sayfa içerisinde Script.Timeout = xx şeklinde belirtilebilir.

App Debugging : Bu bölümde ASP sayfalarının çalışması sırasında oluşan hataların ayıklanabilmesi ile ilgili işlemler gerçekleştirilebilir.



Resim 30 : App Debuging

Debugging Flags

Enable ASP server-side script debugging : Bu seçenek işaretlendiğinde ASP sayfalarının çalışması sırasında Microsoft Script Debugger'da devreye girecektir. Microsoft Script Debugger kullanılarak kodlarda oluşan hatalar incelenebilir. Ancak bu durumda ASP kodları single-threaded (her bir iş sırayla yapılır) olarak çalışır ve performans kaybı yaşanır. Bu sebeple bu işlem normal uygulamalarda tavsiye edilmez. Uygulama geliştirme sırasında kullanılabilir.

Enable ASP client-side script debugging : Bu seçenek ASP 3.0 versiyonunda hiçbir etki yapmamaktadır ve daha sonraki sürümler için eklenmiştir.

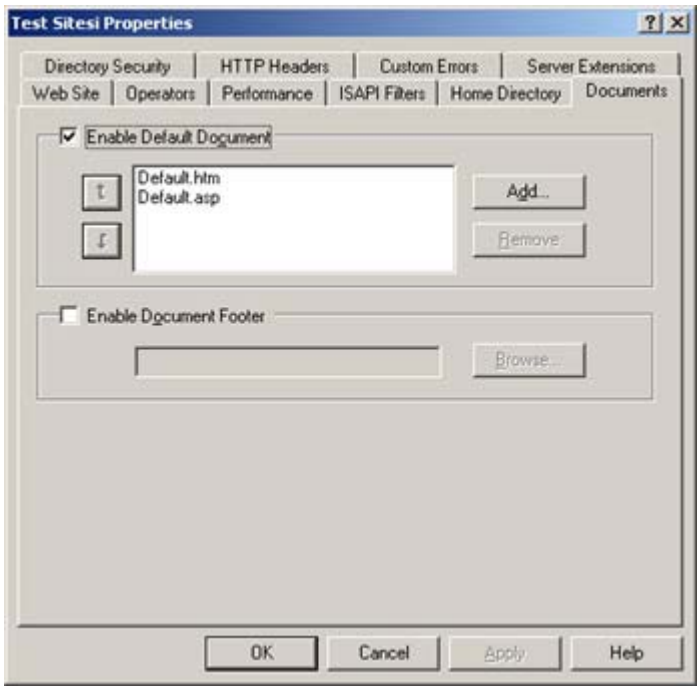
Script Error Messages : Bu bölümde ASP kodlarının çalışması sırasında bir hata oluştuğunda ziyaretçiye gösterilecek mesaj ile ilgili ayarlamalar yapılabilir.

Send detailed ASP error messages to client : Varsayılan olarak işaretlidir. Hata oluştuğunda hata ile ilgili tüm açıklamaların ziyaretçinin browser'ında görünmesini sağlar. Bu açıklamalar hatanın oluştuğu satır, hatanın oluştuğu ASP sayfası, kod ile ilgili açıklama vs. içerir.

Send text error message to client : ASP kodlarının çalışması sırasında bir hata oluştuğunda istenilen mesajın ziyaretçiye ulaştırılmasını sağlar. Böylece hata ile ilgili ayrıntılar ziyaretçiden gizlenmiş olur.

5.7.6. Documents

Bu bölümde Web sitesinin çalışması sırasında varsayılan olarak çalıştırılacak sayfalar belirlenebilir.



Resim 31 : Documents

Enable Default Document : Varsayılan olarak Default.asp ve Default.htm sayfaları yer almaktadır. İsteğe göre farklı dosyalar eklenebilir.

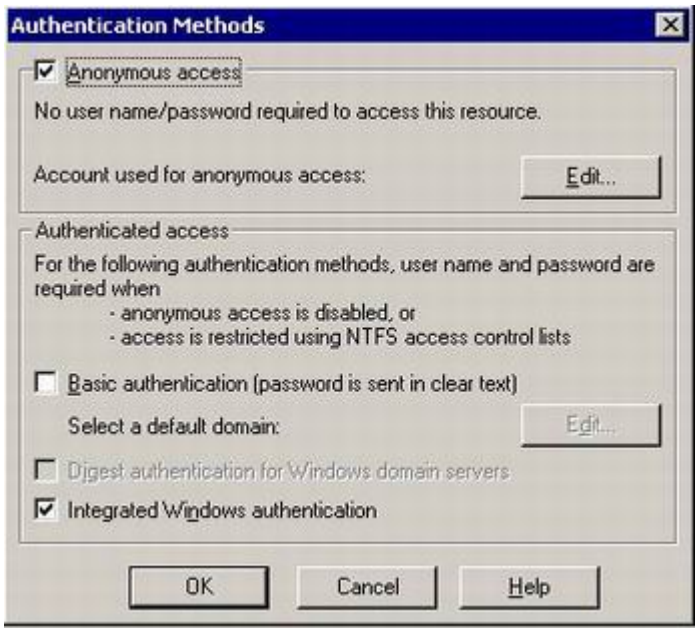
Enable Document Footer : Bu seçenek işaretlenerek Web sitesinde yar alan herhangi bir dosya çalıştırıldığında sayfanın altında HTML formatlı bir mesajın görüntülenmesi sağlanabilir.

5.7.7. Directory Security:

Bu bölümde web sitesi ve klasörlerine erişim ve güvenli erişim sertifikaları ile ilgili ayarlar belirlenebilir.

Resim 32 : Directory Security

Anonymous access and authentication Control : Bu kısımda erişimleri için yetkilendirme (authentication) ve anonim bağlantılar (anonymous access) özellikleri ayarlanabilir. Böylece özel bölümlere herkesin erişmesi engellenebilir, erişebilecek kişiler tanımlanabilir. Herkesin web sitesine erişebilmesi isteniyorsa bir ayar yapma gereği bulunmamaktadır. Ancak belirli bölümlere tanımlanmış kişilerin erişmesi isteniyorsa o zaman ayarların yapılması gerekmektedir. Ayarları yapmak için "Edit" tuşuna basıyoruz.



Resim 33 : Yetki Metodları

Bu pencerede de görüldüğü gibi web sitesine anonim ve yetkilendirilmiş olarak 2 farklı erişim vardır. Yetkilendirilmiş erişim genelde sadece belirli bölümlerde kullanılır.

Anonymous access : Varsayılan olarak bu seçenek işaretlidir. Bu şekilde web üzerinden sitesine gelen ziyaretçilerin sayfalarınızı görüntülemesi sağlanır. Ziyaretçiler, site ziyaretleri sırasında IUSR_makinadi isimli kullanıcı ile bağlanmış olarak görünürler. Bu kullanıcı IIS ile beraber sistemde oluşturulmaktadır.

Edit tuşuna basıldığında Anonymous User Account isimli pencere görüntülenir.

Resim 34 : Yetkili Kullanıcı Belirleme

Bu ekranda resimde de görüldüğü gibi anonim bağlantılar için IUSR_makinadi kullanıcısı seçili durumdadır. Allow IIS to Control password kutusu da varsayılan olarak işaretlidir. Bu ayarda bir değişiklik gerçekleştirilmediği sürece anonim bağlantılar problemsiz olarak gerçekleşecektir. Anonim bağlantıların gerçekleştirildiği kullanıcının adı değişmiş ise Username kısmına yazılarak veya Browse tuşu yardımıyla kullanıcı bulunarak değişiklik aktif hale getirilebilir. Aksi takdirde site düzgün çalışmayacaktır.

Authenticated access : Genellikle yetkiye sahip belirli kişilerin ulaşması istenilen web siteleri veya web sitesi bölümleri için kullanılır. Burada bahsedilen yetkilendirme NTFS dosya sistemi vasıtasıyla sağlanılan web sitesi, klasör veya dosya bazında yetkilendirmedir.

Yetkilendirme için kullanılan kullanıcılar Windows kullanıcılarıdır. Dolayısıyla yetkilendirilmiş erişim için öncelikle sistemde bu kullanıcıların tanımlanması ve ilgili klasörlerde gerekli haklarının verilmiş olması gerekmektedir.
Not : Kullanıcıların belirtilen alanlara web üzerinden yetkilendirilmiş erişimle ulaşabilmeleri için, ilgili makineye logon yetkilerinin olması ve kullanıcı hesabının aktif olması gerekmektedir.

Bu yetkilendirme türü aktif edildiğinde ziyaretçiler web sitesine erişmek istediklerinde kullanıcı adı ve şifre girilmesini isteyen bir pencere görüntülenir. İstenilen bilgiler doğru girildiğinde sayfa görüntülenir. Aksi takdirde hata mesajı içeren sayfa ekrana gelir.

Basic authentication : IIS tarafında yetkilendirme için en çok kullanılan yöntemdir. Girilen kullanıcı adı ve şifre bilgileri normal text halinde şifrelenmemiş olarak gönderilir. Bu yetkilendirme daha çok İnternet üzerinde kullanılır, İntranet üzerinde "Integrated Windows authentication" ile kullanmak daha mantıklıdır.

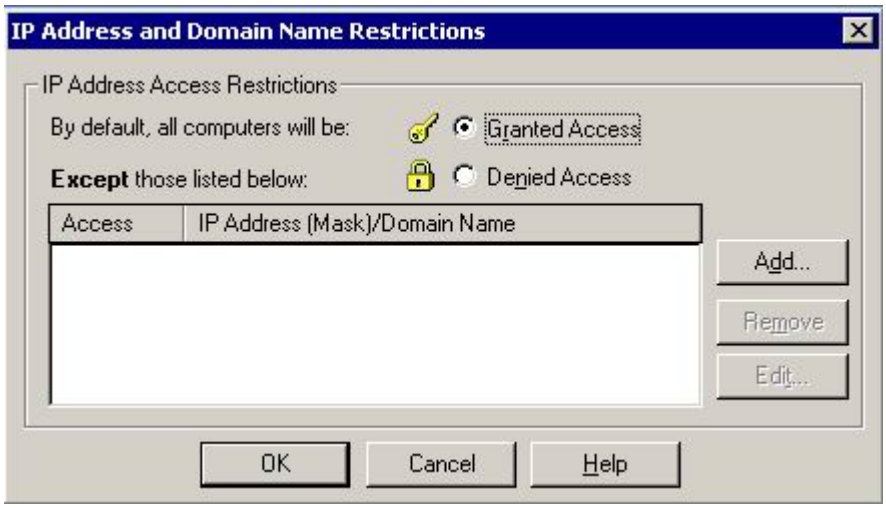
Digest authentication for Windows domain servers : Bu yöntem IIS 5.0 ile beraber gelmektedir. Girilen kullanıcı adı ve şifre bilgileri özel bir algoritma ile şifrelenmiş olarak gönderilir.

Integrated Windows authentication : Windows'un normal kullanıcı işlemleri sırasında kullandığı yetkilendirme metodunu kullanır. Genelde intranette ve domain yapısı içeren networklerde kullanılır. Internet üzerinden de kullanılabilir.

Bu yöntemin 2 kısıtlaması vardır :

1. Sadece Internet Explorer ile çalışır, diğer browser'ler ile çalışmaz. Internet Explorer versiyonu 2.0 ve üzeri olmalıdır.
2. Proxy sunucuları ve HTTP Proxy ile sağlanan bağlantılarla çalışmaz.

IP Address and domain name restrictions : Bu bölüm yardımıyla web sitesine erişmesi veya erişmemesi istenilen IP adresleri, bilgisayarlar veya kullanıcılar tanımlanabilir. Bu özellik sadece server sistemlerde çalışmaktadır.
Edit tuşuna basıldığında "IP Address and Domain Name Restrictions" isimli pencere görüntülenir.



Resim 35 : Giriş Yetkilendirme Ekranı

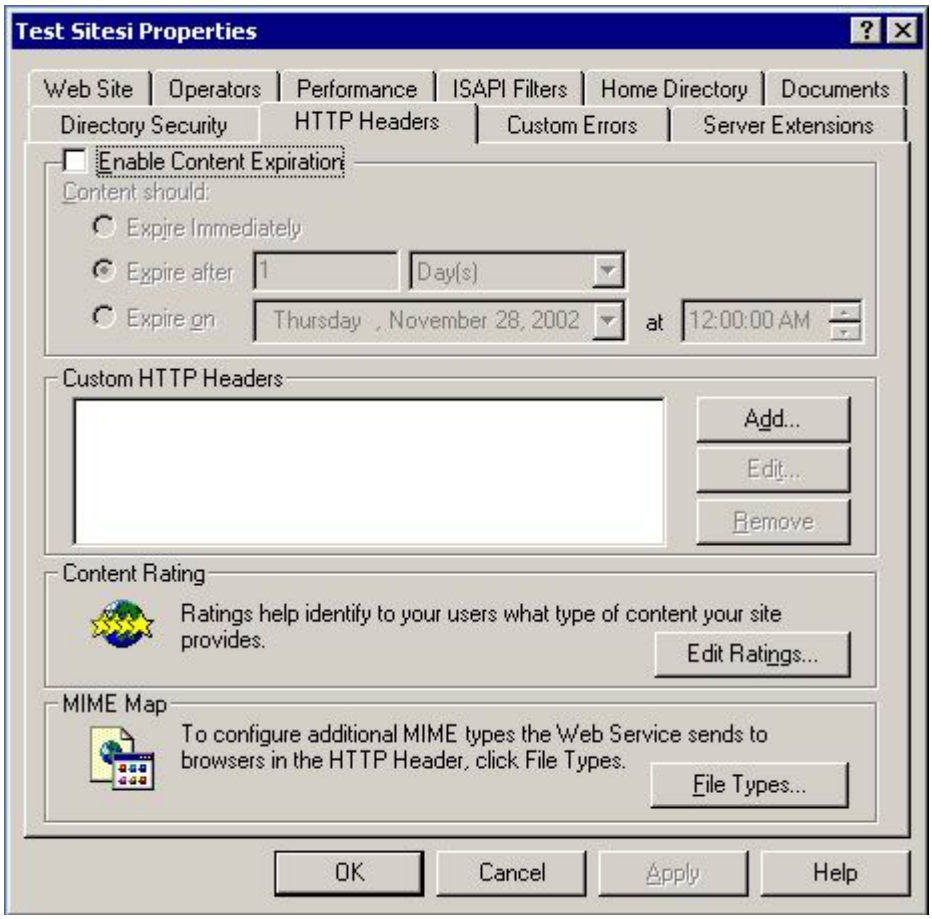
Pencerede görüleceği gibi varsayılan olarak web sitesi herkese açıktır. Granted Access işaretli iken Add tuşu yardımıyla IP adresi, IP adres grubu veya domain adı

eklenirse; eklenen bilgilere sahip kullanıcılar web sitesine erişemeyecektir. Eğer Denied Access işaretli duruma getirilir ise o zaman da sadece listeye eklenmiş olan kullanıcılar web sitesine erişebilecektir.

Secure communications : Bu bölüm, güvenli bağlantıların gerektiği durumlarda (ör: İnternet bankacılığı, şirkete ait gizli bilgilere erişilmesi vb.) kullanılır. SSL sertifikaları bu bölüm yardımıyla eklenerek güvenli bağlantı kurulması sağlanır.

5.7.8. HTTP Headers

Enable Content Expiration : Bu bölümde Enable Content Expiration bölümünü işaretlenerek web sitesindeki tüm sayfaların belirli bir süre sonra Expire (süresi dolması) olması sağlanabilir.



Resim 36 : HTTP Headers

Bu özellik daha çok sürekli değişen sayfalar için kullanılabilir.

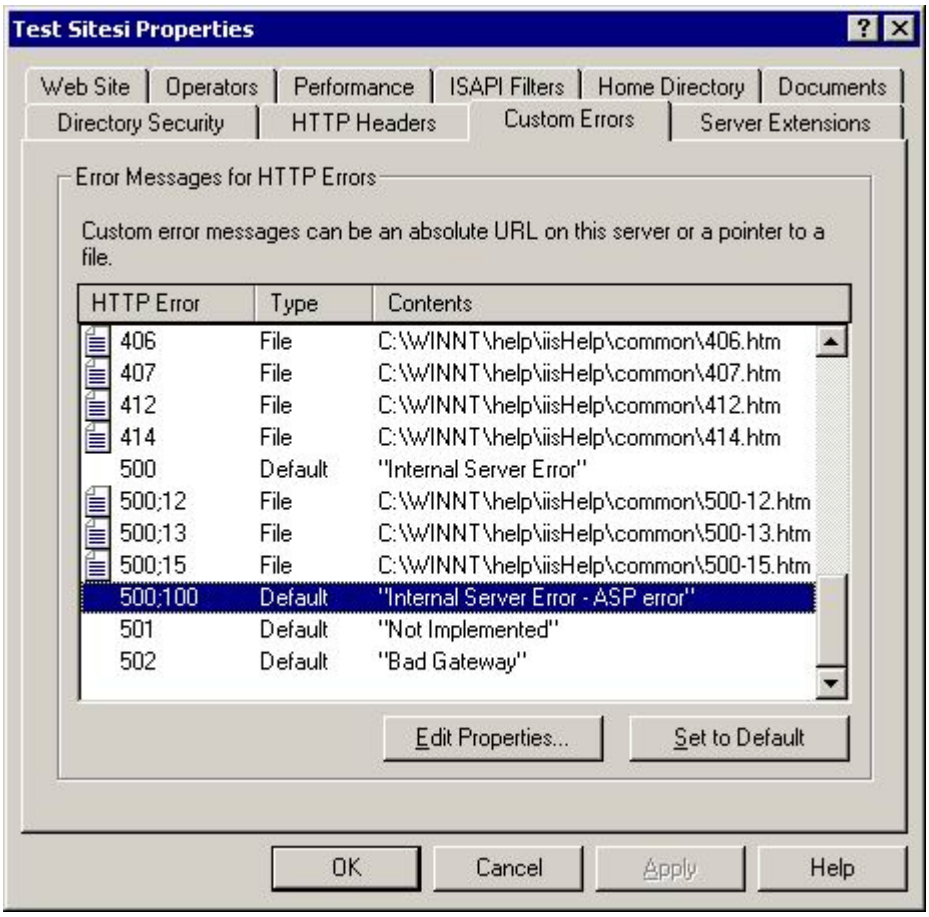
Custom HTTP Headers : Adından da anlaşılabileceği gibi web sitesine özel olarak bilgiler tanımlanabilir. Bu tanımlanan bilgiler Server'dan client'e gönderilir.

Content Rating : Web sitesinde barındırılan özel içerik (yetişkin vs.) ve seviyesi tanımlanarak bu bilgiler HTTP header bilgileriyle client'e gönderilir. Böylece kişilerin istenmeyen içeriğe erişmesi engellenebilir.

MIME Map : Web sitesinde kullanılmak istenilen farklı dosya tipleri bu bölümden eklenebilir.

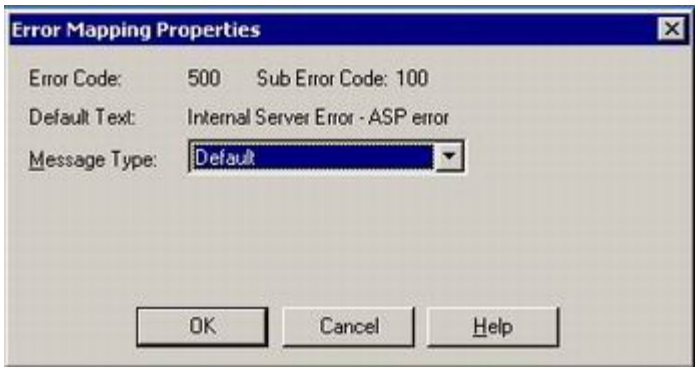
5.7.9. Custom Errors

Bu bölümde web sitesinde oluşan hatalar sonucunda kullanıcıya gösterilen sayfalar değiştirilebilir. Böylece kullanıcıya istenilen şekilde mesajlar verilebilir.



Resim 37 : Custom Errors

Değiştirilmek istenilen mesaja ait link seçili hale getirilerek Edit Properties'e basıldığında Error Mapping Properties isimli pencere görüntülenir.



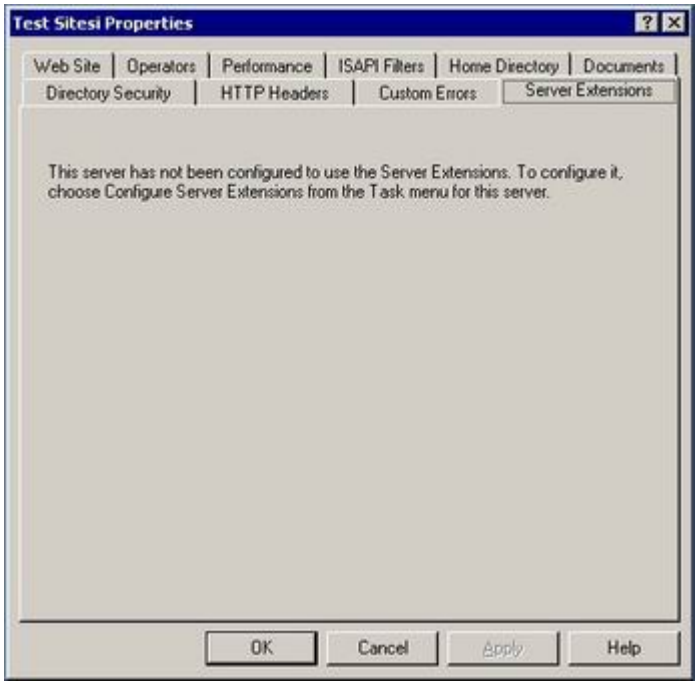
Resim 38 : Hata Özellikleri

Message Type kısmından isteğe uygun seçim yapılarak istenilen mesajlar görüntülenebilir.

Default'a bırakıldığında varsayılan hata mesajı görüntülenir. URL seçildiğinde web sitesinin bulunduğu makinede herhangi bir klasör altında yer alan dosyayı tanımlayabilmek için öncelikli olarak dosyanın bulunduğu klasör virtual directory olarak eklenmelidir. (Ör: C:\test klasöründe yer alan my500.asp isimli dosyayı kullanmak için test isimli bir virtual directory oluşturulur ve adres /test/my500.asp olarak verilir.)

File seçildiğinde makine üzerinden direkt olarak ilgili dosya bulunabilir veya adresi yazılabilir.

5.7.10. Server Extensions



Resim 39 : Server Extensions

Bu bölüm Frontpage Server Extensions (sunucu uzantıları) kullanılabilmesini ve ayarlarının yapılmasını sağlar. Frontpage Server Extensions yardımıyla web sitesindeki dosyalar direkt olarak sunucuya erişilerek, programcının kendi makinesindeymiş gibi düzenlenebilir. Dosyaların FTP veya diğer yöntemlerle sunucuya gönderilmesi gerekmez.

Bu özelliği kullanabilmek için Frontpage Server Extensions'ın sunucu üzerine kurulması gerekir.

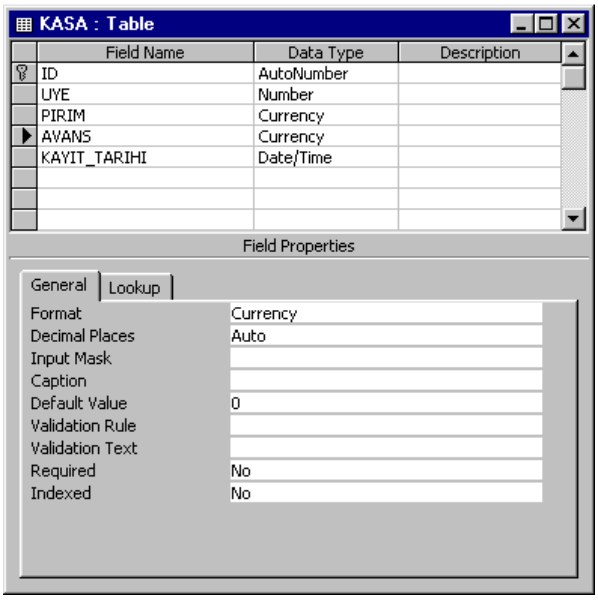
Bu özelliğin kullanılması direkt olarak dosyalara erişim olduğu için güvenlik açıkları meydana getirebilir. Çok gerekmedikçe kullanılması tavsiye edilmez.

6. Ek-3 : SQL Sorgulama Dili

Sunucu tabanlı bir programlama dili ile uğraştıkça ve bu uğraşının içinde veritabanı oldukça, SQL size hiç yabancı gelmeyecektir. Bu deyim, Dr. Edgar F. Codd ve IBM tarafından geliştirilen *Structured Query Language* (Yapısal Sorgu Dili) kelimelerinden geliyor. Biz bu dilin, sunucu tabanlı programlama dillerinde, veri tabanı içindeki kayıtlar arasında sorgulama yapma kısmını inceleyeceğiz. Web sayfalarının en önemli özelliklerinden birisi de, sayfanın açılma hızıdır. Eğer sayfanızda, bir veritabanından bilgi alış-verişi yapıyorsanız, hedef noktaya, yani veritabanı içindeki, sadece istenilen kayıtlara ulaşmak, sayfanızın açılma hızını arttıracaktır. O halde, istenilen kayıtlara ulaşmak için SQL sorgulama dilini iyi bilmek lazımdır.

Bu dili öğrenirken, oluşturacağımız bir veritabanı üzerinde işlem yapacağız. Bu veritabanını *Microsoft Access* programı ile oluşturacağız. Projemiz, bir derneğin üyeleri için bir veritabanı oluşturup, bu veritabanı içinde sorgulama yapmaktır. İki tablodan sorgulama yapmasını öğrenebilmek için, veritabanını iki kısımda oluşturuyoruz. Birinci kısım, üye bilgilerinin olduğu tablo (uye).

Resim 40 : Access Veritabanı .



Resim 41 : Access Veritabanı.

İkinci kısım ise, üyelerin avans ve prim durumlarını gösteren bir tablo olacaktır.

Bir veri tabanı içindeki kayıtlar arasında seçim yapmak için, SELECT deyimi kullanılır. Veri tabanı içinde birden fazla tablo olabilir. Buna göre hangi tablodan seçim yapacağınızı belirtmelisiniz. SQL ile veritabanına FROM deyimi ile ulaşılır. Projemizdeki *uye* adlı tabloya ulaşıp, bütün bilgileri okumak için şöyle bir ifade yazılmalıdır.

```
SQL = "SELECT * FROM uye"
```

Program bu ifadeyi görünce, *uye* adlı tablodaki tüm sütun başlıkları ve kayıtları tek tek okuyacaktır. Burada kullandığımız [*] ifadesi, bütün sütun başlıkları demektir. Eğer, bu tablo içindeki bir veya bir kaç sütun başlığı okunmak istenirse, bu sütun başlık isimlerini SELECT deyiminden sonra yazmak lazımdır. Mesela *uye* adlı tablodan, sadece *AD*, *SOYAD* ve *MESLEK* sütun başlıklarına ait kayıtları okumak istediğimizde, şöyle bir ifade yazılmalıdır.

```
SQL = "SELECT AD, SOYAD, MESLEK FROM uye"
```

Buna göre, *uye* adlı tablodaki başka bir sütun başlığına ulaşamazsınız. Program ancak belirtilen sütun başlıklarına ait kayıtları okuyarak listeler. Eğer bunların dışında bir sütuna ulaşmak isterseniz, ASP programı şöyle bir hata verir.

```
ADODB.Fields error '800a0cc1'
ADO could not find the object in the collection
corresponding to the name or ordinal
reference requested by the application.
/sql/sql_02.asp, line 26
```

Veri tabanı içindeki kayıtları, belirli bir düzende okumak ve listelemek için ORDER BY komutu kullanılır. Bu komuttan sonra, sıralamanın hangi sütun başlığına göre yapılacağını bildirmelisiniz. Aksi belirtilmedikçe, sıralamayı artan şekilde yapar. Yani, sayısal verileri, küçükten büyüğe, alfanümerik verileri de, A'dan Z'ye göre sıralayarak listeler. Mesela, *uye* adlı tablodan *AD*, *SOYAD* ve *MAAS* adlı sütün başlıkları altındaki kayıtları listelerken, *AD*'a göre sıralamak için şöyle bir ifade yazılmalıdır.

```
SQL = "SELECT AD, SOYAD, MAAS FROM uye ORDER BY AD"
```

Eğer, *MAAS* adlı sütun başlığına göre sıralarsanız, liste *MAAS* sütunundaki, rakamlara göre, küçükten büyüğe göre sıralanır. Bu durumu tersine çevirmek isterseniz, sütun başlığının yanına artma veya azalma durumunu belirtmelisiniz. Listeyi artan şekilde sıralamak için, ASC (*ascending*) komutu, azalan şekilde sıralamak için DESC (*descending*) komutu kullanılır. Mesela *AD*, *SOYAD* ve *MAAS* sütun başlıkları altındaki kayıtları listelerken, *MAAS* sütun başlığındaki verilere göre, büyükten küçüğe doğru sıralamak için, şöyle bir ifade yazılır.

```
SQL = "SELECT AD, SOYAD, MAAS FROM uye ORDER BY MAAS DESC"
```

Sıralamayı birden fazla sütun başlığına göre yaptırmak için, sırası ile sütun başlıkları yazılmalıdır. Yani, asıl sıralama yapılacak sütun başlığı ilk yazılır. Bundan sonra yazılacak sütun başlıkları, kendi içinde sıralanır. Mesela, *uye* adlı tablodaki *AD*, *MESLEK* ve *MAAS* sütun başlıkları altındaki kayıtları, ilk önce *MESLEK* sütun başlığındaki verileri azalan, sonra aynı meslek grupları içinde kalanları da, *MAAS* sütun başlığındaki verilere göre artan şekilde sıralamak için, şöyle bir ifade yazılmalıdır.

```
SQL = "SELECT AD, MESLEK, MAAS FORM uye"
SQL = SQL & " ORDER BY MESLEK DESC, MAAS ASC"
```

Her bir sıralamanın arasına virgül işareti koyarak, sıralama kategorisini arttırabilirsiniz. Yukarıdaki ifadede, iki satır kullandık. Ancak ikinci satırda yazdıklarımız, birinci satıra & işareti ile eklenmiş olduğundan, program bu SQL sorgulamasını tek bir satır olarak algılar. Bundan sonra, karışık bazı sorgulamaları, iki veya daha fazla satırda yapacağız.

Veri tabanı içindeki kayıtlardan aynı olanları görüntülemek istemiyorsanız, DISTINCT komutu kullanmalısınız. Program bu komutun arkasında gördüğü sütun başlığında, aynı veriye sahip olanlardan sadece birini okur. Bu komutla birlikte başka bir sütun başlığına göre sıralama komutu kullanılamaz. Sadece DISTINCT komutundan sonra yazılan sütun başlığı okunabilir. Mesela, *uye* adlı tablodaki meslek gruplarını öğrenmek için, şöyle bir ifade yazılmalıdır.

```
SQL = "SELECT DISTINCT MESLEK FROM uye ORDER BY MESLEK DESC"
```

Veri tabanı içindeki kayıtları listelerken belirli bir şarta göre okunmasını sağlayabilirsiniz. Bunun için, WHERE komutu kullanılır. Bu komuttan sonra yazılan şarta uyan kayıtlar listelenmiş olur. Mesela, *uye* adlı tablodan, maaşı 2.000.000 'un üzerinde olanları listelemek için, şöyle bir ifade yazılır.

```
SQL = "SELECT AD, SOYAD, MAAS FROM uye WHERE MAAS > 2000000"
```

Bu şartı belirlerken kullanabileceğiniz işlemciler şunlardır.

- Büyük.
- < Küçük.
- = Eşit.
- >= Büyük veya Eşit.
- <= Küçük veya Eşit.
- Eşit Değil.

Karşılaştırma bir string'e göre yapılacaksa, string tek tırnak içine alınmalıdır. Mesela, *uye* adlı tablodan mesleği memur olanları görüntülemek için, ifade aşağıdaki gibi yazılmalıdır.

```
SQL = "SELECT AD, SOYAD, MESLEK FROM uye WHERE MESLEK='memur'"
```

Sorgulamayı yaparken, tarihlere göre şart koymak için, tarih olarak yazılan değeri # işaretleri arasına almak ve veri tabanındaki tarih formatına uygun olarak yazmak gereklidir. Mesela, *uye* adlı tablodan, doğum tarihi 01.01.1980'den büyük olanları listelemek için, aşağıdaki ifade yazılmalıdır.

```
SQL = "SELECT AD, SOYAD, DOGUM_TARIHI FROM uye"
SQL = SQL&" WHERE DOGUM_TARIHI > #01/01/1980#"
```

Veri tabanına kayıtları işlerken, bazı kayıtların *True/False* (Doğru/Yanlış) olarak girilmesi gerekir. Bunu veri tabanında *Yes/No* olarak tanımlarız. Bu şekilde tanımlanmış olan, kayıt sütunu için, seçmeli kutucuklar çıkar. Bu kutucuklar seçili ise, kaydın değeri 1 (yani *True*), seçili değilse kaydın değeri, 0 (yani *False*) olur. Projemizde, üyelerin cinsiyet durumlarını, veri tabanına *Yes/No* olarak giriyoruz. Yani, bay için 1 değeri, bayan için 0 değeri verilmiş olur. Veri tabanından sadece bayanların listesini almak için, sorgulama aşağıdaki gibi yazılır.

```
SQL = "SELECT AD, SOYAD, CINSIYET FROM uye"
SQL = SQL&" WHERE CINSIYET = 0"
```

Çoğu sorgulamada bir koşul işimizi görmeyebilir. Birden fazla koşul belirtmek için AND, OR veya NOT komutları kullanılır. Bunun için birkaç örnek yapalım. Mesela, erkek öğrencileri listelemek için, aşağıdaki sorgulama yazılmalıdır.

```
SQL = "SELECT AD, SOYAD, CINSIYET, MESLEK FROM uye"
SQL = SQL&" WHERE CINSIYET = 1"
SQL = SQL&" AND MESLEK = 'öğrenci'"
```

Maaşı 2.500.000'nin altında olmayan ve doğum tarihi 01.01.1975'ten büyük olanları listelemek için;

```
SQL = "SELECT AD, SOYAD, DOGUM_TARIHI, MAAS FROM uye"
SQL = SQL&" WHERE DOGUM_TARIHI > #01/01/1975#"
SQL = SQL&" AND NOT MAAS < 2500000"
```

Serbest meslek sahibi veya muhasebeci olanları isim sırasına göre listelemek için;

```
SQL = "SELECT AD, SOYAD, MESLEK FROM uye"
SQL = SQL&" WHERE MESLEK = 'serbest'"
SQL = SQL&" OR MESLEK = 'muhasebe'"
SQL = SQL&" ORDER BY AD"
```

AND ve OR komutlarını istediğimiz kadar kullanabiliriz. Ancak bu komutları arka arkaya kullanmaktansa, IN komutu ile, birden fazla sorgulama aynı anda yapılabilir. Mesela, *uye* adlı tablodaki, mesleği öğrenci, işçi ve serbest olmayan bayanları soyadlarına göre listelemek için, aşağıdaki gibi sorgulama yapılabilir.

```
SQL = "SELECT AD, SOYAD, MESLEK, CINSIYET FROM uye"
SQL = SQL&" WHERE NOT MESLEK IN ("serbest','öğrenci','işçi')"
SQL = SQL&" AND CINSIYET = 0"
SQL = SQL&" ORDER BY SOYAD"
```

Yukarıdaki yazılımda, her bir meslek için ayrı ayrı OR komutu kullanabilirdik. Ancak IN komutu ile bir veri kümesi tanımlamış olduk ve bu veri kümesine göre arama yaptırdık. IN komutu NOT ile beraber kullanılabilir. Yani, yukarıdaki örneğin 2. satırını, şu şekilde de yazabiliriz.

```
WHERE MESLEK NOT IN ("serbest','öğrenci','işçi')
```

Sorgulamayı belirli bir aralıkta yaptırmak için, ya AND yada BETWEEN komutu kullanılır. Mesela, *uye* adlı tablodan, doğum tarihi 01.01.1970 ile 01.01.1980 arasında olan erkekleri, doğum tarihi sırasına göre listelemek için, aşağıdaki ifade yazılabilir.

```
SQL = "SELECT AD, SOYAD, CINSIYET, DOGUM_TARIHI FROM uye"
SQL = SQL&" WHERE DOGUM_TARIHI > #01/01/1970#"
SQL = SQL&" AND DOGUM_TARIHI < #01/01/1980#"
SQL = SQL&" AND CINSIYET = 1"
SQL = SQL&" ORDER BY DOGUM_TARIHI"
```

Aynı sorgulama, aşağıdaki şekilde daha kısa olarak yazılabilir.

```
SQL = "SELECT AD, SOYAD, CINSIYET, DOGUM_TARIHI FROM uye"
SQL = SQL&" WHERE DOGUM_TARIHI BETWEEN #01/01/1970#"
SQL = SQL&" AND #01/01/1980#"
SQL = SQL&" AND CINSIYET = 1"
SQL = SQL&" ORDER BY DOGUM_TARIHI"
```

Veri tabanı içinde SQL ile arama yaptırmak mümkündür. Yani, bir karakter dizisini, belirli bir sütun başlığı içinde, arama yaptırabilirsiniz. Bunun için, LIKE komutu kullanılır. Arama yapılacak sütun başlığını, WHERE komutundan sonra yazıp, sonra LIKE komutunu yazmak lazımdır. Bu komutla, iki şekilde arama yaptırılabilir. Arama yapılacak karakterleri, tek tırnak içinde ve % işaretleri arasında yazarsanız, içinde bu karakterlerin geçtiği tüm kayıtları verir. Mesela, *uye* adlı tablodan, soyadında "ak" karakterleri olanları listelemek için, aşağıdaki sorgulama ifadesi yazılır.

```
SQL = "SELECT AD, SOYAD, MESLEK FROM uye"
SQL = SQL&" WHERE SOYAD LIKE '%ak%'"
```

Bu aramayı birden fazla sütun başlığında yaptırabilmek için, OR komutu ile devam edebilirsiniz. Yukarıdaki örneğe ilave olarak, adının içinde "ay" karakterleri geçenleri de sorgulayalım.

```
SQL = "SELECT AD, SOYAD, MESLEK FROM uye"
SQL = SQL&" WHERE SOYAD LIKE '%ak%'"
SQL = SQL&" OR AD LIKE '%ay%'"
```

Yukarıdaki her iki örnekte de, aranacak karakterlerin başına ve sonuna % işareti koyduk. Böylece, bu karakterlerin, ilgili sütun başlığı altındaki kayıtların herhangi bir yerinde geçip geçmediğini kontrol ederiz. Eğer yüzde işareti koyulmaz ise, eşitlik durumunda olan kayıtları verir. Yani yukarıdaki örneğe göre, soyadı "ak" olanları listeler. Aşağıdaki iki ifade de aynıdır.

```
WHERE SOYAD LIKE 'ak'
WHERE SOYAD='ak'
```

LIKE komutu ile belirli sayıda karakteri ve bu karakterlerin içinde geçenleri de belirleyip de aramak mümkündür. Bunun için alt çizgi işareti kullanılır (_). Mesela, *uye*

adlı tablodan adı, "er" ile başlayan ve 5 karakter olanları listelemek için, aşağıdaki ifade yazılmalıdır.

```
SQL = "SELECT AD, SOYAD, MESLEK FROM uye"
SQL = SQL&" WHERE AD LIKE 'er_ _ _'"
```

Yukarıdaki yazılımda, alt çizgiler birleşik olmalıdır. Yukarıdaki gibi yazdığınız zaman, hata verir veya kayıt bulamaz.

SQL sorgulama dilinde matematiksel işlemler de yaptırmak mümkündür. Bir sütun başlığına ait kayıtların toplamını almak için SUM komutu kullanılır. Mesela, *uye* adlı tablodaki, tüm üyelere ait maaşların toplamını bulmak için, aşağıdaki ifade yazılır.

```
SQL = "SELECT SUM (MAAS) FROM uye"
```

ASP sayfasında bu toplamı gösterebilmek için, kayıt dizisinin arkasına "(0)" yazılmalıdır. Yukarıdaki örneğe göre, maaş toplamı şu şekilde alınır.

```
<%
Dim veriyolu, SQL, dizi
Set veriyolu=Server.CreateObject("ADODB.Connection")
veriyolu.Open ("Driver={Microsoft Access Driver
(*.mdb)};DBQ="&Server.MapPath("dernek.mdb"))

SQL = "SELECT SUM(MAAS) FROM uye"

Set dizi=veriyolu.Execute(SQL)

Response.Write "Maaş toplamı = "
Response.Write dizi(0)
%>
```

Burada *dizi(0)* ifadesi, tüm üyelerin maaş toplamını verir. Eğer, belirli bir kritere göre toplam almak isterseniz, WHERE komutu ile bu koşulu belirleyebilirsiniz. Mesela, *uye* adlı tablodaki, bayan öğrencilerin maaş toplamını, aşağıdaki ifade ile alabiliriz.

```
SQL = "SELECT SUM (MAAS) FROM uye"
SQL = SQL&" WHERE MESLEK = 'öğrenci'"
SQL = SQL&" AND CINSIYET = 0"
```

Aritmetiksel ortalama almak için, AVG komutu kullanılır. Bu komutla, kayıtlardaki rakamlar toplanır ve kayıt sayısına bölünür. Mesela, *uye* adlı tablodaki, memur ve yazarların maaş ortalamasını almak için, aşağıdaki ifade yazılmalıdır.

```
SQL = "SELECT AVG (MAAS) FROM uye"
SQL = SQL&" WHERE MESLEK IN ('memur','yazar')"
```

Bir tablo içindeki en yüksek veya en düşük değeri bulmak için, MAX ve MIN komutları kullanılır. Mesela, *uye* adlı tablodaki, mesleği, işçi ve avukat olmayanlar içinde, yaşı en genç olanı bulmak için, aşağıdaki ifade yazılır.

```
SQL = "SELECT MAX (DOGUM_TARIHI) FROM uye"
SQL = SQL&" WHERE MESLEK NOT IN ('işçi','avukat')"
```

Tablo içindeki kayıtların sayılması için, COUNT komutu kullanılır. Herhangi bir koşul belirtilmediği zaman, tüm kayıtları sayar. Mesela, *uye* tablosundaki 2.000.000'nun üzerinde maaş alan kaç tane erkek üye olduğunu öğrenmek için, aşağıdaki ifade yazılır.

```
SQL = "SELECT COUNT(*) FROM uye"
SQL = SQL&" WHERE MAAS > 2000000"
SQL = SQL&" AND CINSIYET = 1"
```

Veri tabanında bazı işlemleri yaptırırken, gruplandırarak yapmak gerekebilir. Mesela, her meslek grubuna ait işlemler için, ayrı ayrı işlem yapmaktansa, sorgulamayı gruplandırmak daha pratik olacaktır. Bunun için, GROUP BY komutu kullanılır. *uye* adlı tablodaki, her mesleğin maaş toplamlarını almak için, en pratik çözüm aşağıdaki gibidir.

```
SQL = "SELECT MESLEK, SUM (MAAS) FROM uye"
SQL = SQL&" GROUP BY MESLEK"
```

ASP programında, hem meslek gruplarını görmek, hem de maaş toplamlarını görmek için, yine kayıt dizisi değişkeninden faydalanırız. Yani bu sorgulamayı yaptıktan sonra, değerler aşağıdaki gibi alınır.

```
<table>
<tr>
<td>GRUPLAR</td>
<td>MAAŞ TOPLAMI</td>
</tr>

<%
Set dizi = veriyolu.Execute(SQL)
Do While Not dizi.eof
%>

<tr>
<td> <% Response.Write dizi(0) %> </td>
<td> <% Response.Write dizi(1) %> </td>
</tr>

<%
Loop
dizi.Close
%>

</table>
```

Buna göre, her grup, kayıt dizisinin bir elemanı olacaktır. Sorgulamayı gruplandırarak yaparken, koşul vermek için, HAVING komutunu kullanmalısınız. Mesela, *uye* tablosundaki, ortalama maaşın 3.000.000'nun üzerinde olan meslek gruplarına ait, en yüksek maaşları listelemek için, aşağıdaki ifade yazılmalıdır.

```
SQL = "SELECT MESLEK, MAX (MAAS) FROM uye"
SQL = SQL&" GROUP BY MESLEK"
SQL = SQL&" HAVING AVG (MAAS) > 3000000"
```

HAVING komutunu kullanırken, MIN, MAX, COUNT, AVG veya SUM komutlarından en az birinin olması gerekir. HAVING komutu WHERE ile karıştırılmamalıdır. WHERE ile, tablo içindeki kayıtları tek tek sorgularken, HAVING grup bazında işlem yapar.

Veri tabanında, birden fazla tablo içinde sorgulama yapmak için, iki tabloyu JOIN metodu birleştirmek gerekir. Bu metod ile, seçim yapılacak olan tablo adları, FROM komutundan sonra yazılır. Birleştirmenin hangi koşulda yapılacağını WHERE komutu ile bildirmelisiniz. Birleştirmeyi yaparken, ilk önce tablo adını, sonra sütun başlığı adını yazmak gerekir. Mesela, *kasa* tablosundaki prim ve avansların kimlere ait olduğunu göstermek için, aşağıdaki ifade yazılmalıdır.

```
SQL = "SELECT AD, SOYAD, PRIM, AVANS FROM uye, kasa"
SQL = SQL&" WHERE uye.UYE_NO = kasa.UYE"
SQL = SQL&" ORDER BY AD"
```

2. satırdaki, "WHERE uye.UYE_NO = kasa.UYE" ifadesi, birleştirmenin olduğu yerdir. Görüldüğü gibi, ilk önce tablo adı ve arkasına sütun başlığı yazılmış. Tablo adı ve sütun başlığı, nokta işareti ile birleştirilmiştir.

İki tabloyu birleştirerek, yukarıda öğrendiğimiz diğer özellikleri kullanmak mümkündür. Mesela, her meslek gruplarına ait, ödenmiş olan prim toplamlarını listelemek için, aşağıdaki sorgulama ifadesi yazılmalıdır.

```
SQL = "SELECT MESLEK, SUM (PRIM) FROM uye, kasa"
SQL = SQL&" WHERE uye.UYE_NO = kasa.UYE"
SQL = SQL&" GROUP BY MESLEK"
SQL = SQL&" ORDER BY MESLEK"
```

Yukarıdaki örnekte, birleştirme üye numaralarına göre yapılıyor. Yani, *kasa* tablosundaki üye numarası ile *uye* tablosundaki üye numarası aynı ise, o kayda ait meslek gurubunun prim toplamına, *kasa* tablosundaki prim miktarı eklenir. Bu listeyi ASP sayfasında yazdırma işlemini yukarıda öğrenmiştik. Yani, kayıt dizisinin elemanlarını tek tek çağırıyoruz.

İki tabloyu birleştirmeden sorgulama yapmak için, iç içe sorgulama yapmak gerekir. Mesela, 02.01.2001 tarihinden sonra, 2.000.000'nin üzerinde avans çekmiş üyelerin listesini görüntülemek için, iki adet sorgulama yapmak gerekir. Bunlardan ilki, *kasa* tablosunda, bu iki koşula uyan üye numaralarını almak olacaktır. Alınan bu üye numaralarına göre, *uye* tablosundaki üyeler listelenecektir. İfade aşağıdaki gibi yazılır.

```
SQL = "SELECT UYE_NO, AD, SOYAD, MESLEK FROM uye"
SQL = SQL&" WHERE UYE_NO IN"
SQL = SQL&" (SELECT UYE FROM kasa"
SQL = SQL&" WHERE AVANS > 2000000"
SQL = SQL&" AND KAYIT_TARIHI > #02/01/2001#)"
SQL = SQL&" ORDER BY UYE_NO"
```

Yukarıdaki örneğe dikkat ettiyseniz, ilk sorgulama parantez işareti içinde yapılıyor. Bu sorgu neticesinde çıkan üye numaraları, ikinci sorgulamanın IN listesini oluşturuyor. Bu listedeki üye numaralarına eşit olan numaralar, ikinci sorguda listeleniyor. İki tabloyu birleştirmenin bir diğer yolu da, UNION komutudur. Bu komutla, iki sorgulamanın sonuçlarını birleştirebilirsiniz. Ancak sorgu neticesinde, aynı sayıda sütun başlığı içermelidir. Mesela, maaşından fazla prim veya avans almış olanları listelemek için, aşağıdaki ifadeler yazılabilir.

```
SQL = "SELECT AD, SOYAD, MAAS, AVANS, PRIM FROM uye, kasa"
SQL = SQL&" WHERE uye.UYE_NO = kasa.UYE"
SQL = SQL&" AND PRIM > MAAS"
SQL = SQL&" UNION SELECT AD, SOYAD, MAAS, AVANS, PRIM FROM uye, kasa"
SQL = SQL&" WHERE uye.UYE_NO = kasa.UYE"
```

```
SQL = SQL&" AND AVANS > MAAS"
```

Yukarıdaki örnekte, iki adet sorgulama var. Bunlardan ilki primin maaştan büyük olup olmadığını kontrol ediyor. İkincisi ise, avansın maaştan büyük olup olmadığını kontrol ediyor. Bu iki sorgulamanın neticesinde, *AD, SOYAD, MAAS, AVANS* ve *PRIM* sonuçları çıkıyor. Her iki sorguyu birleştirmek için, 4. satırda UNION komutu kullanılıyor.

7. Ek-4 : Bu Kitapta Kullanılan Standartlar

Aşağıdaki tablo, bu kitapta kullanılan yazım standartlarını açıklamaktadır.

Standart Örneği	Açıklaması
Sub, If, Case Else, Print, True, BackColor, Click, Debug, Long	Metinde, dile özel komut sözcükleri ilk harf büyük olarak ve kalın yazılır.
File Menüsü, **Add Project iletişim kutusu**	Arabirim öğelerinin çoğu, ilk harf büyük olarak ve kalın yazılır.
Setup	Yazmanız istenen sözcükler kalın yazılır.
Değişken	Sözdizimi ve metinde, italik harfler, sizin sağlayacağınız bilgiler için yer tutucuları belirtebilir.
[ifadelistesi]	Sözdiziminde, köşeli parantezler içindeki öğeler isteğe bağlıdır.
{ while | Until }	Sözdiziminde, küme işaretleri ve dik bir çubuk, iki yada daha çok öğe arasında bir seçim yapmanızı belirtir.
Sub merhaba() Response.write ("Merhaba") End Sub	Bu yazı tipi kod için kullanılmaktadır.
ENTER	Büyük harfler, ENTER ve CTRL+R gibi, tuş adları ve tuş birleşimleri için kullanılmaktadır.
ALT+F1	Tuş adları arasındaki artı işareti (+), bir tuş birleşimini göstermektedir. Örneğin, ALT+F1, ALT tuşuna basarken F1 tuşuna basmanız anlamına gelir.
AŞAĞI OK	Tek yön tuşlarına tuşun üstündeki ok yönü (SOL, SAĞ, YUKARI ve AŞAĞI) ile başvurulur. "ok tuşları" ifadesi, bu tuşların tümü için kullanılmaktadır.
BACKSPACE, HOME	Diğer gezinme tuşları, kendi özel adları ile başvurulur.
C:\inetpub\wwwroot\test.asp	Yol ve dosya adları, sürücü harfi büyük diğer kısım küçük harf şeklinde yazılmaktadır
↵ yada _	Bir kod satırının sonunda kullanıldığında, alt çizgi (_) ya da ok (↵), kodun sonraki satırda devam ettiğini belirtir. Kendi kodunuzu yazarken bu karakteri yazmayınız.