PHP'ye Giriş

PHP yorumlayıcısı, bu "programı" çalıştırabilmek için dosyanın içinde PHP komutlarını arar. PHP komutları birinci bölümde gördüğümüz gibi iki şekilde yazılabilir:

1. <?PHP .... ?>
2. <? .... ?>

Bunlara PHP komut ayracı denir; birinci türü uzun veya standart ayraç sayılır; ikincisine ike "kısa ayraç" denir.

PHP kodlarımız, oluşturulmasını istediğimiz sayfanın HTML kodları ile tabir yerinde ise içiçe yazılır.

```
<?PHP
        print "Merhaba genclik ☺☺!";
?>
```

Sayfalara yorum eklemek için;

```
<HTML>
<!- - Bu satır HTML'in yorum satırı
Buraya istediğimiz kadar yorum yazabiliriz..
Browser yani siteyi çalıştıran tarayıcı bu satırları dikkate almaz - - >
<HEAD>
<TITLE>PHP ile Merhaba</TITLE>
</HEAD>
<BODY>
<CENTER>
<B>
<H1>
<?PHP
/*
Bu satır da PHP'nin çok-satırlı yorum bölümü..
Buraya istediğimiz kadar yorum yazabiliriz.
*/
        print "Merhaba Genclik ☺☺!";
// Bu ise PHP'nin tek satırlı yorum bölümü
# Bu satırları da PHP yorumcusu dikkate almayacaktır.
?>
</H1>
</B>
</CENTER>
</BODY>
</HTML>
```

Değişkenler

PHP'de de, bir çok başka bilgisayar programlama dilinde olduğu gibi değişkenlerin içine bir değer konmadan önce tanımlanması mümkündür; fakat gerekli değildir. Değişkenleri adının önüne $ işareti koyarak tanımlarız:

```
$adi;
$soyadi;
$15;
$sevdigiRenk;
```

PHP'de genellikle değişkenleri değerini atayarak belirleriz:

```
$adi = "Fatih";
$soyadi = "Maraşlı";
$15 = 123;
$sevdigiRenk = "alabulabirrenk";
```

Değişkenler, kullanıldıkları işleme, taşıdıkları değeri verirler:

```
print $adi;
```

PHP'de özel bir değişkene değişken adı olarak kullanılacak değerleri de atayabiliriz:

```
$adi = "Fatih";
$degisken = "adi";
print $$degisken;
```

Burada Browser penceresine yine "Fatih" kelimesi yazılacaktır; çünkü PHP $degisken adlı değişkenin "adi" adlı değişkeni tuttuğunu bilecek ve iki Dolar işaretini görünce, $degisken'in değerini değil, onun tuttuğu değişkenin değerini yazacaktır.

Veri Türleri

PHP açısından dünyada altı tür değer vardır:

Tamsayı	**(Integer): 5,124, 9834 gibi**
Çift	**(Double): 3,567 gibi**
Alfanümerik	**(String): "Fatih" gibi**
Mantıksal	**(Boolean): doğru (true)/yanlış (false) gibi**
Nesne	**(Object)**
Dizi	**(Array)**

Tür Değiştirme

Bir değişkenin değerinin türü hakkında kuşkunuz varsa, en emin yol bunu PHP'nin kendisine sormaktır. Bu sorgulamayı **gettype()** fonksiyonu ile yaparız.

Şimdi, bir PHP programı yazalım, bir takım değişkenlere değerler atayalım ve bunların türlerini PHP'ye soralım.

```
$sayi = 5;
print("Birinci değişkenin adı: \$sayi<br>");
print("Değeri : ");
print "$sayi<br>";
print("Türü : ");
print gettype( $sayi ) ; //tamsayi/integer
print "<br>";
print "<br>";

$alfanumerik = "Fatih";
print "İkinci değişkenin adı: \$alfanumerik<br>";
print "Değeri : ";
print "$alfanumerik<br>";
print("Türü : ");
print gettype( $alfanumerik ) ; //alfanümerik/string
print "<br>";
print "<br>";

$ondalik = 5.1234;
print "Üçüncü değişkenin adı: \$ondalik<br>";
print "Değeri : ";
print "$ondalik<br>";
print("Türü : ");
print gettype( $ondalik ) ; //çift,ondalık/double
print "<br>";
print "<br>";

$mantiksal = true;
print "Üçüncü değişkenin adı: \$mantiksal<br>";
print "Değeri : ";
print "$mantiksal<br>";
print("Türü : ");
print gettype( $mantiksal ) ; //mantıksal/boolean
print "<br>";
print "<br>";
```

Burada mantıksal (boolean) değer olarak doğru anlamına true değeri atadığımız halde, PHP'nin bu değişkenin değeri olarak 1'i gösterdiğine dikkat edin. PHP'de bir fonksiyon, elde ettiği değer doğru ise sonuç olarak 1 değerini verir. (Yanlış/false değerinin neye çevrildiğini bulabilir misiniz?)

Escape

şu satırdaki, ters-bölü işareti dikkatinizden kaçmamış olmalı:

```
print "İkinci değişkenin adı: \$alfanumerik<br>";
```

PHP için özel anlamı olan işaretlerin anlamlandırılmasını önlemek ve bu işaretleri düz metin saymasını sağlamak için bu işaretlerin önüne ters-bölü işareti koyarız.
şöyledir:

\'	Tek tırnak
\"	Çift tırnak
\\	Ters-bölü
\$	Dolar işareti
\n	Yeni Satır (New Line)
\r	Satır Başı (Return)
\t	Sekme (Tab) karakteri

Kimi zaman bir değişkene atadığımız değerin türünü değiştirmek gerekir. Bunu **settype()** fonksiyonu ile yaparız.

```
$a = 5.67890;
        print("Değişkenin değeri : ");
        print "$a<br>";
        print("Türü : ");
        print gettype( $a ) ; //çift,ondalık/double
        print "<br>";
        print "<br>";

        print "İlk değiştirme işlemi: Alfanümerik/String:<br>";
        settype( $a, string ); //alfanümerik/string (integer,double,booleon)
        print "Değeri : ";
        print "$a<br>";
        print("Türü : ");
        print gettype( $a ) ; //alfanümerik/string
        print "<br>";
        print "<br>";
```

Dört yararlı fonksiyon

isset() ve **unset()**
isset() fonksiyonu, PHP'nin bir değişkenin içinde değer bulunup bulunmadığını sınamasını sağlar.
unset() ise varolan bir değişkeni yok eder.

```
if (isset($a))
{
print( $a );
}
else
{
```

```
unset($a);
}
```

Bu kod parçası, $a isimli dğişkenin içi boş değişse, içeriğini görüntüleyecek, içi boş ise varlığına son verecektir.

empty()

isset() fonksiyonun tersi işleve sahiptir; bir değişkene değer atanmamışsa, veya değeri sıfır veya boş alfanümerik (null string) ise, doğru (True) değeri verir.

```
$bir_degisken = 123;
echo empty($bir_degisken);
$bir_degisken = "";
echo empty($bir_degisken);
```

is_string(),is_integer(),is_double(),

Sınadıkları değişkenin aradıkları türden değer içermesi halinde doğru 'True) sonuç verirler.

```
<?php
        $a = 6.567;
if (is_double($a)) {
        print ("A Double'dır<br>");
        }
        $b = "Fatih";
if (is_string($b)) {
        print ("B String'dir<br>");
        }
        $c = 6;
if (is_int($c)) {
        print ("C Integer'dır<br>");
        }
?>
```

Bu kod, Browser penceresine "A double'dır, B String'dir, C Integer'dır" yazdıracaktır. PHP'de bu fonksiyonlara benzeyen fakat başka tür değer arayan şu fonksiyonlar da vardır: **is_array(), is_object.**

İşlemciler (Operatörler)

Aritmetik işlemciler:

+	Toplama	6+5	=	11
-	Çıkartma	6-5	=	1
/	Bölme	6/5	=	1.2
*	Çarpma	6*5	=	30
%	Kalan (Modulus)	6%5	=	1

$sayi=4.5;
echo floor($sayi); =>> 4 (Sayıyı aşağıya yuvarlar)
echo ceil($sayi); =>> 5 (Sayıyı yukarıya yuvarlar)
echo round($sayi); =>> 5 (Sayıyı yuvarlar)

echo max(13,12,5,7); =>> 13
echo min(13,12,5,7); =>> 5

Rastgele Sayı

Srand((double) microtime()*1000000);
echo rand(20,30) =>> (20-30 arası sayı döndürür)

PHP'nin atama işlemcisinin eşittir (=) işareti olduğunu hatırlıyorsunuz birleşik-atama (combined-assignment) işlemcileri, bu işarete diğer aritmetik işlemciler eklenerek oluşturulur.

İşlemci	Örnek	Anlamı
+=	$a += 5	$a = $a + 5
-=	$a -= 5	$a = $a - 5
/=	$a /=5	$a = $a / 5
*=	$a *= 5	$a = $a * 5
%=	$a %= 5	$a = $a % 5
.=	$a .= "metin"	$a = $a" metin"

Bir Arttırmak veya Azaltmak için

Değerleri sadece 1 arttırmak veya azaltmak için PHP, bir kolaylık sağlar:

$a++ veya ++$a : $a'nın değerini 1 arttırır;
$a-- veya --$a : $a'nın değerini 1 eksiltir.

PHP'nin karşılaştırma yapması için kullandığımız işlemciler ise işlem işaretinin sağı ve solundaki değerleri veya değişkenlerin değerlerini işaretin belirttiği karşılaştırmayı yaptıktan sonra ortaya ya doğru (true) ya da yanlış (false) sonucunu çıkartırlar.

İşlemci	Örnek	Örnek	$a=6 ise:
==	eşitse	$a == 5	Yanlış/False
!=	eşit değilse	$a != 5	Doğru/True
===	aynı ise	$a === 5	Yanlış/False
>	büyükse	$a > 5	Doğuru/True
<	küçükse	$a < 5	Yanlış/False
<=	küçükse veya eşitse	$a <= 5	Yanlış/False
>=	büyükse	$a >= 5	Doğru/True

veya eşitse

PHP'de bu karşılaştırmayı iki grubun arasına koyduğumuz işaretlerle yaparız İşaretin sağ ve sol tarafının doğruluğu veya yanlışığı işarete göre nihai sonucun doğru veya yanlış olmasını sağlar. Bu karşılaştırmaları yaparken şu işlemcileri kullanırız:

İşlemci	Adı	Anlamı	Örnek
||	veya	sol veya sağ doğru	doğru || yanlış = doğru
or	veya	sol veya sağ doğru	doğru || yanlış = doğru
&&	ve	sol ve sağ doğru	doğru || yanlış = yanlış
and	ve	sol ve sağ doğru	doğru || yanlış = yanlış
Xor	Şartlı-veya	Sadece sol veya sağ doğru	doğru || yanlış = doğru
!	Değil	sol veya sağ yanlış	doğru || yanlış = doğru

```
<?php
        $vize = 45;
        $final = 65;
        if ($vize >= 50 && $final >= 50) {
        print ("Öğrenci geçti!");
        }
        else {
        print ("Öğrenci kaldı!");
        }
?>
```

Sabit Değerler

define ("SABIT_DEGER", değer);

Burada SABIT_DEGER yerine, tanımlamak istediğimiz sabit değere vereceğimiz isim, değer yerine de sabit değeri yazarız. Örnek:

```
<?php
        $Dolar_miktar = 125;
        define ( "DOLAR_KURU", 625675);
        $TL_Tutar = $Dolar_miktar * DOLAR_KURU;
        print ($TL_Tutar);
?>
```

Tanımlanmış olan bir sabiti yeniden oluşturamayız; ama buna teşebbüs ettiğimizde PHP hata vermez. Bir sabit değerin oluşturulmuş olup olmadığını **defined()** fonksiyonu ile anlayabiliriz:

```
<?php
        $Dolar_miktar = 125;
        if (defined( "DOLAR_KURU" )) {
        echo ("Sabit değer daha önce tanımlanmıştı.<br>");
        }
```

```
?>
```

Dizi-Değişkenler

Dizi değişkenden ayrıntılı söz edebilmek için önce tipik bir dizi-değişkende neler olduğuna bakalım. Sözgelimi, verdiğiniz "PHP ile Programlama" kursundaki öğrencilerinizin listesi şöyle olabilir:

Dizi Değişken Oluşturalım

Şimdi, PHP bize öyle bir araç vermeli ki, biz bir kerede bu listenin tümünü, her bir ögesine sanki bir değişkenin değeri imiş gibi tek-tek, veya bir kaçına birden ulaşabilmeli ve arzu ettiğimiz zaman notları doldurabilmeliyiz. Öğrenciler de yapacağımız Web sitesine girerek, kendi notlarını görebilmeli ve notlarını inceleyebilmeli. PHP'nin bu amaçla sağladığı araç, çok-boyutlu dizi-değişken oluşturma aracıdır. Ve bu araçla yukarıdaki listeyi aynen şöyle yapabiliriz.

```
<?php
$ogrenciler = array (
array ( adi => "AliSelim", soyadi => "Efe", sinav1 =>"", sinav2 =>"", not =>""),
array ( adi => "Faruk", soyadi => "Taç", sinav1 =>"", sinav2 =>"", not =>""),
array ( adi => "Ali", soyadi => "Civelek", sinav1 =>"", sinav2 =>"", not =>""),
array ( adi => "Şükran", soyadi => "Tabak", sinav1 =>"", sinav2 =>"", not =>""),
);
// Buraya başka kodlar girecek
print $ogrenciler[0][adi];
?>
```

Programdaki "print()" komutunu sadece dizi değişkeni doğru yazıp yazmadığımızı sınamak amacıyla yazdık; bu programı Browser'da açtığınızda yazdığınız ilk ismi Browser penceresinde görüyorsanız, dizi-değişkeni doğru şekilde oluşturdunuz demektir. Burada, array() komutnu yazarken, süslü parantez değil, normal parantez kullandığımıza ve herbir elemanın değerlerinin sonunda vrigül olduğuna dikkat edir. Bir diğer önemli nokta: endeks adları bir kelimeden fazla ise bunları tırnak içine alarak belirtmektir. Örneğin:

```
array ( adi => "AliSelim", soyadi => "Efe", "Sinav 1 Notları" =>"", "Sinav 2 Notları" =>"",
"Toplam Not Ortalamasi" =>""),
```

Burada, daha öncekilere benzer bir şekilde adlandırılmış $ogrenciler değişkenin içeriğini array() komutu ile doldurduğumuzu görüyoruz. Array() ile böyle çok boyutlu ve içerdiği değerlerin her birinin bir "endeks adı" olan dizi-değişkene İlişkili Dizi (Associative array) de denir. Perl bilenler ise bu tür değişkenlere "Hash" dendiğini hatırlayacaklardır. İlişkili Dizi'lerin birinci satırı 0, ikinci satırı 1, üçüncü satırı 2.. diye numaralandırılır. Bu dizinin o satırındaki kaydın sıra endeksidir. Ayrıca burada "adi," "soyadi," "sinav1" .. dizi değişkenin içindeki değerlerin endeks adıdır. Yani bu değerlere atıfta bulunurken, referans yaparken veya bu değerleriekullanmak amacıyla erişirken sıra endeksi ve endeks adıyla hitabederiz. Yukarıdaki sınama amaçlı print() komutuna bakarsanız, birinci öğrencinin ismini "[0][adi]" olarak çağırıyor.

Çok elemanlı ilişkili dizi oluşturmanın bir diğer yolu, yeri geldiğinde böyle bir dizi için yeni bir üye ilgili bilgileri eleman endeksi ve değerler için endeks adı belirterek şöyle bir kod yazmaktan ibarettir.

```
<?php
        $ogrenciler[0][adi] = "AliSelim";
        $ogrenciler[0][soyadi] = "Efe";
        $ogrenciler[0][sinav1] = "";
        $ogrenciler[0][sinav2] = "";
        $ogrenciler[0][not] = "";
// Buraya Buraya başka kodlar girecek
        print $ogrenciler[0][adi];

?>
```

Bir dizi değişkende kaç boyut olacaksa, o kadar içiçe array() ögesi oluşturabiliriz. Buna göre tek boyutlu bir dizi değişken sadece bir array() komutu ile ve sadece değerler verilerek oluşturulabilir. Diyelim ki yukarıdaki öğrenci listemiz sadece öğrencilerin isimlerinden oluşacak. Bu durumda $ogrenciler değişkenine ilişkin satırı şöyle yazabilirdik:

```
$ogrenciler = array ("AliSelim", "Faruk", "Ali", "Şükran");
```

PHP, böyle tek boyutlu bir dizinin örneğin birinci elemanını, "$ogrenciler[0]" adıyla bilir. Böyle bir tek-boyutlu diziyi oluşturmak için PHP bize başka bir kolaylık da sağlar: array() komutunu kullanmadan, doğruca dizinin ögelerine değer vermemiz mümkündür. Yukarıdaki programın sadece PHP bölümünü şöyle değiştirerek kaydedin:

```
<?php
        $ogrenciler[] = "AliSelim";
        $ogrenciler[] = "Faruk";
        $ogrenciler[] = "Ali";
        $ogrenciler[] = "Şükran";
// Buraya başka kodlar girecek
    print $ogrenciler[0];
?>
```

Böyle sırayla dizi değişken oluşturur veya oluşturulmuş bir dizi değişkene ek yaparken, değişkenin sıra numarasını yazmazsak, PHP bunları kendisi sıralar. Yukarıdaki kodun da Browser penceresine "AliSelim" yazdırması gerekir. Mevcut tek-boyutlu bir dizi değişkene ek yaptığımızda, be yeni değerin dizinin en altına eklenmesini istiyorsak, sıra numarası yazmamıza gerek yoktur. Mevcut değerlerden birini değiştirmek istiyorsak, o değerin sıra numarasını yazmamız gerekir. Bunu denemek için yukarıdaki kodu şöyle değiştirilim

```
<?php
        $ogrenciler[] = "AliSelim";
        $ogrenciler[] = "Faruk";
        $ogrenciler[] = "Ali";
        $ogrenciler[] = "Şükran";
// Buraya başka kodlar girecek
        $ogrenciler[0] = "Ferhat";
        $ogrenciler[15] = "AliSelim";

    print ("Dizideki 1'nci isim: $ogrenciler[0] <br>");
```

```
        print ("Dizideki 2'nci isim: $ogrenciler[1] <br>");
        print ("Dizideki 3'üncü isim: $ogrenciler[2] <br>");
        print ("Dizideki 4'üncü isim: $ogrenciler[3] <br>");
        print ("Dizideki 5'inci isim: $ogrenciler[4] <br>");
        print ("Dizideki 6'ncı isim: $ogrenciler[5] <br>");
        print (".....<br>");
        print ("Dizideki 15'nci isim: $ogrenciler[15] <br>");

?>
```

Bu programın Browser penceresine göndereceği sırada, birinci öğrenci ($ogrenci[0]) olarak bu kez AliSelim değil Ferhat yazdığını göreceğiz.

Bunun sebebi, diziyi oluşturan ilk grup deyimden sonra,

```
$ogrenciler[0] = "Ferhat";
```

satırı ile birinci elemanın değerini değiştirmiş olduk. 15'nci elemana atama yapmakla, PHP'nin $ogrenciler dizisinde 6, 7, 8, 9,.. 14'e kadar boş elemanlar oluşturmasına sebep olduk.
Tek boyutlu dizileri de İlişkili Dizi olarak oluşturabilir yani değerlere endeks adı verebiliriz.

```
<?php $ogrenci[adi] = "AliSelim";
        $ogrenci[soyadi] = "Efe";
        $ogrenci[sinav1] = "";
        $ogrenci[sinav2] = "";
        $ogrenci[not] = "";
// Buraya başka kodlar girecek
        print $ogrenci[adi];
?>
```

PHP, $ogrenci adlı değişkenin beş ayrı değeri olduğunu ve bunların "adi," "soyadi," "sinav1"... olduğunu biliyor. Şimdi artık istediğimiz noktada bu değişkenin istediğimiz değerine, o değerin endeks adını yazarak, çağrıda bulunabiliriz; bu değeri yeniden verebiliriz.

Dizi değişkenleri kullanalım

Yukarıdaki paragrafta "..değişkenin istediğimiz değerine, o değerin endeks adını yazarak, çağrıda bulunabiliriz.." dediğimizi görmüş olmalısınız. Dizi veya tekil, değişkenleri oluşturmamızın sebebi, tuttukları değerleri programımızın gereği olan şekilde ve yerde kullanmaktır. Sadece bir değer tutan değişkenleri örneğin print() komutu ile sık sık kullandık. Yukarıda dizi değişken örneklerinde de bazı değişkenleri ve değerlerini çağırdık. Ancak dizi değişkenlerin değerlerinden yararlanabilmek için başka araçlar da vardır.

Herşeyden önce dizi değişkenlerin büyüklüğü, boyutu bizim için önem taşıyabilir. Özellikle bir veritabanı dosyasını okutarak oluşturacağımız dizi değişkenin kaç elemanı ve her bir elemanın kaç ögesi bulunduğunu bilmemiz gerekebilir.

Bir dizi değişkenin kaç elemanı bulunduğu, o değişkenin count() özelliği sorgulanarak öğrenilir. count(), dizideki eleman sayısını verir. Şimdi bunu bir örnekle görelim.

```
<?php
        $ogrenciler[] = "AliSelim";
        $ogrenciler[] = "Faruk";
        $ogrenciler[] = "Ali";
        $ogrenciler[] = "Şükran";
// Buraya başka kodlar girecek
print ("\$ogrenciler adlı dizide ". count($ogrenciler) ." adet eleman var.");
?>
```

Bu program Browser penceresine dizimizde 4 eleman bulunduğunu bildirecektir. Şimdi işleri biraz karmaşık hale getirelim! Yukarıdaki kodun, print() satırının yerine şu satırları ekleyerek, kaydedelim.

```
print ("\$ogrenciler adlı dizide ". count($ogrenciler) ." adet eleman var.");
print ("<br><br>");
for ($sayac=1 ; $sayac <= count($ogrenciler) ; $sayac++ )
{
print ("\$ogrenciler dizisinin ". $sayac ."'ncı elemanı: " . $ogrenciler[$sayac] ."<br>");
}
```

Bu programı çalıştırmadan önce, eklediğimiz satırları irdeleyelim. İlk print() komutunun Browser penceresine "yazdıracağı" metinde geçen ters bölü işaretini hatırlıyor olmalısınız. Bu, tek veya çift tırnak içine de almış bile olsak, PHP'nin, bir değişken adını gördüğü zaman onun yerine o değişkenin tuttuğu değeri yazması sebebiyle, $ işareti gibi PHP için özel anlamı olan işaretlerin anlamlandırılmasını önlemek için yaptığımız ve adına o karakteri kurtarma veya ESCaping dediğimiz işlemdir. Bu işlemle, PHP'nin anlamlı işaret değil de metin saymasını istediğimiz karakterlerin önüne ters bölü işareti koyarız: \" gibi. Buradaki örnekte, bu sayede PHP "$ogrenciler" kelimesini değişken adı olarak değil, düz metin olarak görüyor. Ki, aynı komutta aynı kelimeyi tekrar kullandığımızda bu kez değişken adı olarak kullanıyoruz ve bu değişkenin count() ögesinin değerini öğreniyoruz. $ogrenci değişkenin "AliSelim," "Faruk," "Ali" ve "Şükran" değerleri bulunduğuna göre, bu değişkenin count()'u 4 olacaktır. ("Ozbay" = 0, .. "Şükran" = 3 olmak üzere..) Bu print() komutu, Browser penceresine tahmin ettiğiniz gibi "$ogrenciler adlı dizide 4 adet eleman var." yazdıracaktır. İkinci print() satırı ise ekrana ardarda iki yeni satır işareti gönderecektir.

Şimdi karışık noktaya geliyoruz! Burada bir for döngüsü başlıyor. Önce döngünün kaç kez tekrar edeceğini belirleyecek olan değişkeni tanımlıyoruz: $sayac. Sonra bu sayacın kaça kadar çıkacağını belirliyoruz. Bu sayıyı, bize yine count() veriyor. Ve tabiî for döngüsünün devam edebilmesi için gerekli son unsur olan, sayacın arttırılmasını sağlayan deyim var. Programımız bu döngünün içinde, yani dört kez, her seferinde dizinin bir elemanın adını Browser penceresine gönderiyor. Şimdi, hatırlayacaksınız, dizi değişkenlerin elemanlarının bir sıra sayısı vardı. Örneğin "Şükran" değeri, dizinin 3 numaralı, yani dördüncü elemanı; ve bu elemanın değerini ekrana göndermek için şu komutu vermemiz yeterli:

```
print ($ogrenciler[4]);
```

Programda ise buradaki endeks sayısını, $sayac değişkeninin o andaki değerinden alıyoruz. Döngünün her seferinde bu değer bir artacağı için bize $ogrenciler değişkeninin o anda hangi elemanının değeri çağırmak istiyorsak, o elemanın endeksini vermiş olacaktır. Ve sonuç olarak programımız, dizideki bütün değerleri Browser'a gönderecektir.
Kimi zaman buradaki örnekte olduğu gibi, dizinin bütün elemanlarını bir for döngüsüyle değil, foreach döngüsüyle bulmak daha kolay olabilir. Kısaca belirtmek gerekirse, foreach döngüsü, bir dizi değişkenin bütün elemanları için, arzu ettiğiniz işi yapar. foreach döngüsünü yazarken komutun kaç

kere icra edileceğini bir sayaçla tutmak gerekmez; çünkü döngü, ona adını verdiğiniz değişkenin içindeki bütün değerler bitinceye kadar devam edecektir.

```
foreach ($ogrenciler as $ogrenci)
        {
        print ("$ogrenci<br>");
        }
```

foreach döngüsü, bir dizi değişkenin adını içinden değer çekilecek kaynak olarak ister; bunu "as" (olarak) kelimesi izler; sonra diziden alınacak her bir değeri geçici olarak tutacak değişkenin adı verilir. Buradaki print() komutumuz, bu geçici değişkenin tuttuğu değeri Browser'a gönderecektir. Bu değer ise döngünün her adımında dizi değişkendeki bir değer yani öğrencilerin listesi olacaktır.

Dizi elemanlarının farklı özelliklerine ilişkin değerlere endeks adı verdiğimiz ilişkili dizilerde ise eleman değerlerini çağırmak foreach döngüsünün biraz farklı yazılmasını gerektirir. Perl'e aşina alanların bu dizi türüne "hash" dendiğini hatırlayacaklardır. PHP'de de Perl'ün hash türü değişkenlerinde olduğu gibi, endeks adlarına "anahtar" (key), bu endeksin belirlediği değere ise (evet, doğru tahmin ettiniz!) değer (value) denir. İlişkili dizilerden değer almak üzere foreach döngüsü yazılırken, değerin anahtarını ve değerin kendisini iki geçici değişkene yazmamız gerekir

```
foreach ($ogrenciler as $anahtar=>$deger)
        {
        print ("$anahtar = $deger<br>");
        }
```

Bu kodu çalıştırmadan önce foreach döngüsü üzerinde kısaca duralım: döngü, $ogrenciler dizisini okumaya başladığında içinde, benzetme yerinde ise, iki sütun, ve bir çok satırlar bulacaktır. Bu sütunlardan birincisi, ikinci sütundaki verinin adıdır; foreach, birinci sütundaki veriyi alarak $anahtar adlı geçici değişkenin değeri olarak atayacak; sonra ikinci sütuna geçecek ve bunu alarak $deger adlı geçici değişkenin değeri yapacaktır. Döngü, daha sonra print() komutunu icra edecektir. print() ise ve geçici $anahtar değişkeninin değerini, ardından eşittir işaretini ve son olarak da geçici $deger değişkeninin değerini Browser'a gönderecektir. print() komutunun icrası bitince, foreach, kendisine verdiğimiz $ogrenciler değişkeninde anahtar-değer çiftini ele almadığı satır kalıp kalmadığına bakacak, ve elemanların tümü bitinceye kadar bu işlemi tekrar edecektir. Tabiî, sonuç anahtar ve değerlerin altalta sıralanması olacaktır.

Bir de bu bölümün en başında ele aldığımız çok elemanlı ilişkili diziler vardı. Onların içindeki değerleri acaba nasıl alabilir ve kullanabiliriz? Tabiî yine bir döngü ile. Fakat bu kez, döngü-içinde-döngü kullanmak zorundayız. Böyle bir diziyi gözümüzde canlandırırsak, belki neden iki döngüye ihtiyaç olduğununu daha iyi görebiliriz. Gözümüzün önüne bir tablo getirelim: dizinin her bir elemanı (bizim öğrenimizde öğrenciler9 bir satırda yer almış olsun; sütunlar olarak da bu elemana ait değerler yer alıyor. Sütun başlığı ise, bu değerin endeksi olan anahtar

```
foreach ( $ogrenciler as $ogrenci )
        {
        foreach ( $ogrenci as $anahtar => $deger )
                {
                print ("$anahtar = $deger <br> ");
                }
        print ("<br>");
        }
```

Kısaca irdelersek, bu kodda foreach döngüsünün önce çok-boyutlu değişkenimizin bir satırını içindeki

bütün anahtar+değer çeftleri ile ele alıp, tümünü $ogrenci adlı değişkene geçici olarak yerleştirdiğini görüyoruz. Bu foreach döngüsünün ilk işi yeni bir foreach döngüsü başlatmak oluyor. Yeni foreach ise sazı eline alır almaz, önce, kendisi çok ögeli bir değişken olan (çünkü içinde bir öğrenciye ait, tüm değişkenler ve onların endeks adları var) $ogrenci değişkeninin içindeki anahtar ve değer çiftlerini tek-tek, $anahtar ve $deger değişkenlerine yerleştiriyor; sonra print() komutu ile, aralarına eşittir işareti koyarak bu değişkenlerin değerlerini Browser penceresine gönderiyor. Bu döngü biter bitmez, ilk foreach yaptıracağı işlere kaldığı yerden devam ediyor; ve ekrana bir yeni satır komutu gönderierek, başa dönüyor; bu kez çok boyutlu dizi değişkenin yeni bir elemana geçiyor. Taa ki, dizinin bütün elemanları ve elemanların bütün ögeleri bitinceye kadar.

Bu noktada bir uyarı: Gerçek programda bir dizinin elemanlarına ilk ulaştığımızda, elemanın içinde değer bulunup bulunmadığını anlamak yerinde olur. Bunu is_array() fonksiyonu ile yapabiliriz. Bu fonksiyon, dizinin içinde değer varsa, True/Doğru, yoksa False/Yanlış karşılığını verecektir. Buradaki örnekte, ilk foreach satırından hemen sonra:

is_array($ogrenci)

satırını koyarak, dizinin o anda okunan elemanın içinde değer bulunup bulunmadığını anlayabiliriz.

Dizi Değişkenlerin Düzenlenmesi

Dizi değişkenlerin daha verimli şekilde kullanılması için PHP bize bir takım araçlar sağlar. Bunlarla dizi değişkenleri birleştirebiliriz; içinden kesit alabiliriz, sıralayabiliriz veya bazı elemanlarını silebiliriz. Şimdi kısaca bu işlemleri ele alalım:

Dizileri birleştirme: array_merge()

İki veya daha fazla dizinin bütün elemanlarını birleştirerek, ortaya yeni bir dizi çıkartır. Örnek:

```
$birinci_dizi = array ( "AliSelim" , "Faruk" , "Ali" , "Şükran" );
$ikinci_dizi = array ( "Efe" , "Taç" , "Civelek" , "Tabak" );
$yeni_dizi = array_merge ( $birinci_dizi, $ikinci_dizi );
```

Bu kod ile oluşturulan $yeni_dizi isimli dizi değişkenin hangi elemanlara sahip olduğunu, şöyle bir kodla görebilirsiniz:

```
foreach ( $yeni_dizi as $yeni_eleman )
        {
        print (" $yeni_eleman <br>");
        }
```

İkinci dizinin bütün elemanları, birinci dizinin elemanlarının arkasına eklenmiş olmalı. array_merge() işlemi, çok-boyutlu ilişkili dizilere de uygulanabilir; PHP iki dizideki uyumlu-uyumsuz, yani birinde olan diğerinde olmayan bütün anahtar+değer çiftlerini yeni dizide de oluştur. (array_merge() işleminden sonra birleştirilen dizilerin değişmeden kaldığına dikkat edin.

Dizilere değişken ekleme: array_push()

Bir diziye yeni değişkenler eklemek için, array_push() fonksiyonuna mevcut dizinin adını ve yeni değerleri yazarız. Örnek:

```
$birinci_dizi = array ( "AliSelim" , "Faruk" , "Ali" , "Şükran" );
$yeni = array_push ($birinci_dizi, "Efe" , "Taç" , "Civelek" , "Tabak" );
```

Burada $yeni adlı değişken sadece $birinci_dizi adlı dizinin yeni eleman sayısını tutar. array_push(), kendisine adını verdiğimiz dizinin içeriğini değiştirir. Yukarıdaki örnekte içine yeni değerler yazılan dizinin elemanlarını görüntülemek için şöyle bir kod yazabiliriz:

```
print ("\$birinci_dizi adlı dizide $yeni_dizi adet değişken var<br>");
foreach ( $birinci_dizi as $ogrenci )
        {
        print ("$ogrenci <br> ");
        }
```

Dizinin ilk elemanını silme: array_shift()

Bir dizi-değişkenin ilk elemanını tümüyle silmek için array_shift() fonksiyonunu kullanırız. Bu fonksiyona sadece birinci elemanı silinecek dizinin adını vermek yeter.
Örnek:

```
$birinci_dizi = array ( "AliSelim" , "Faruk" , "Ali" , "Şükran" );
$silinen = array_shift ($birinci_dizi);
```

array_shift(), adını verdiğiniz dizinin içeriğini değiştirir; buradaki örnekte, $silinen adlı değişken dizinin silinen birinci elemanın değerini tutar.

Diziden kesit alma: array_slice()

Bir dizi-değişkenin bütün elemanları yerine bir kesitini kullanmak istiyorsak, bunu array_slice() fonksiyonu ile yapabiliriz. Bu fonksiyona kesit alınacak dizinin adı, kesitin başladığı yer ve kaç adet değişken alınacağı argüman olarak verilir.

Örnek

```
$birinci_dizi = array ( "AliSelim" , "Faruk" , "Ali" , "Şükran", "Efe" , "Taç" , "Civelek" ,
"Tabak");
$kesit = array_slice ($birinci_dizi , 3, 4);
```

Burada, PHP'ye $kesit adlı yeni dizi değişkene, $birinci_dizi adlı dizinin 3'ncü değerinden itibaren (3 dahil) dört değeri yerleştirmesini bildiriyoruz. array_slice(), adını verdiğimiz değişkenin içeriğine dokunmaz; yeni dizi değişken oluşturulur.

Dizileri sıralama: sort() ve rsort()

Bir dizinin içindeki değerleri alfabetik veya küçükten büyüğe doğru sıralamak için sort() fonksiyonunu kullanırız.

Örnek:

```
$birinci_dizi = array ( "AliSelim" , "Faruk" , "Ali" , "Şükran", "Efe" , "Taç" , "Civelek" ,
"Tabak");
```

sort ($birinci_dizi);

PHP, dizideki bütün değerleri A'dan Z'ye sıraya sokacaktır. sort() fonksiyonu dizinin içeğini değiştirir. Bir diziyi Z'den A'ya veya büyükten küçüğe doğru sıralamak için de rsort() fonksiyonunu kullanabilirsiniz. (PHP4.0 Türkçe karakterleri tanımıyor.) Bir noktada dikkatli olmak gerekir: bu fonksiyonu ilişkili (değerlerin anahtarı olarak endeks adı bulunan) dizide kullanırsanız, PHP, anahtar değerlerini (endeks adlarını) atar, yerine 0'dan itibaren rakam koyar. Bunu önlemek için, ilişkili dizileri asort() veya ksort() fonksiyonu ile sıralamak gerekir.

İlişkili dizileri sıralama: asort() ve ksort()

İlişkili dizilerin diğer dizi değişkenlere göre farkı, değerlerinin bir de adı bulunmasıdır. Değerlerin adlarına anahtar denir. Bir ilişkili diziyi değerlerine göre sıralamak için asort() fonksiyonu kullanılır.

Örnek:

$birinci_dizi = array (ogr_01=>"AliSelim", ogr_02=>"Faruk" , ogr_013>"Ali" , ogr_04=>"Şükran");
asort ($birinci_dizi);

PHP, bu diziyi değerler itibariyle alfabetik sıraya sokacaktır. Eğer sıranın değere göre değil de değerlerin anahtarına (burada ogr_01, ogr_02 olan kelimeler) göre yapılmasını istiyorsak, ksort() fonksiyonunu kullanırız.

Örnek:

$birinci_dizi = array (ogr_01=>"AliSelim", ogr_02=>"Faruk" , ogr_013>"Ali" , ogr_04=>"Şükran");
ksort ($birinci_dizi);

PHP, şimdi bu diziyi anahtarlara göre alfabetik sıraya sokacaktır.

Metin Düzenleme ve Düzenli İfadeler

substr($degisken,8);
substr ($degisken, 8, 20);
substr($degisken, -9);
trim ($degisken);
strlen($degisken);
strip_tags($metin) =>> (Metin içersindeki html ve php kodlarını atar)
strtolower($metin) =>> (Küçük harfe çevirir)
strtoupper($metin) =>> (Büyük harfe çevirir)
ucwords($metin) =>> (Sadece Baş Harfleri büyük yapar)
ucfirst($metin) =>>(Sadece cümlenin baş harflerini büyük yapar)
substr($metin,3,5) =>>(3. karekterden itibaren 5 karekter alır)
strpos($metin,"@") =>>(İstediğim karekterin yerini söyler)
strstr($metin,"@") =>>(İstediğim karekterden sonraki karekterleri alır)
substr_count($metin,"@") =>> (İstediğim karekterden kaçtane olduğunu yazar)

$bolumler=explode(",",$metin) =>> ((,) ler arasındaki ifadeleri dizi değişkenine aktarır)
implode(",",$bolumlar") =>> (Dizideki değerleri (,) işareti ile bağlar)

printf() ve sprintf()

Bu fonksiyonları bir değişkeni biçimlendirmekte kullanırız. Birincisinin elde ettiği sonuç ziyaretçinin Browser penceresine gönderilir; ikincisin elde ettiği sonuç ise değer olarak döner. Önce bu fonksiyonlarla kullanabileceğimiz biçim parametrelerini sıralayalım:

%	Yüzde işareti. Yanında biçim parametresi gerekmez.
b	Değişken tamsayı olarak işlem görür ve ikili sayı olarak döner.
c	Değişken tamsayı olarak işlem görür ve ASCII değerinin karşılığı olan karakter olarak döner.
d	Değişken tamsayı olarak işlem görür ve ondalık sayı olarak döner.
f	Değişken kesirli sayı olarak işlem görür ve kesirli sayı olarak döner.
o	Değişken tamsayı olarak işlem görür ve sekiz-tabanlı (octal) sayı olarak döner.
s	Değişken alfanümerik olarak işlem görür ve alfanümerik olarak döner.
x	Değişken tamsayı olarak işlem görür ve 16 tabanlı (hexadecimal) sayı olarak döner. (Harfler, küçük harf olur).
X	Değişken tamsayı olarak işlem görür ve 16 tabanlı (hexadecimal) sayı olarak döner. (Harfler, büyük harf olur).

Her iki fonksiyonun da kullanılış biçimi aynıdır:

printf("biçim" , $degisken1, $degisken2, ... "metin");

Burada "biçim" yerine yukarıdaki biçim parametlerini yazarız. Biçim parametrelerinin önüne yüzde işareti konur; en fazla beş belirleyici özellik alabilir. Yukarıdaki tür belirten biçimlendirme parametlerine ek olarak diğer özellikler şöyle sıralanır:
Doldurma karakteri: tek tırnak ve onu izleyen bir karakterden oluşur.
Hizalama: Eksi işaretinin varalığı yazının sola, yokluğu ise sağa hizalanma anlamına gelir.
En az-en çok uzunluk: Sayı-nokta-sayı (örneğin 40.40 gibi) yazılır; birinci sayı azamî, ikinci sayı asgarî uzunluğu belirtir.

Bu üç özelliğe bir örnek verelim.Bir değişkenin değerinin sonuna yanyana yeteri kadar nokta konarak uzunluğunun 40 karaktere çıkartılmasını şu deyimle sağlarız:

```
$degisken = " İyilik üzerine " ;
printf( "%'.-40.40s" , $degisken);
```

Burada "%'.-40.40s" şeklindeki biçim komutu, Browser penceresinde şu görüntüyü oluşturur:
`"İyilik üzerine........................."`
Burada "İyilik üzerine" değeri 14 karakter olduğu için, sonuna 26 adet nokta eklenmiş ve bütün değer sola hizalanmış olacaktır. Şu komut ise iki değişkenin değerini ve vereceğimiz bir metni aynı satıra yazdıracaktır:

```
$degisken1 = " İyilik üzerine " ;
$degisken2 = " İyilik üzerine " ;
$metin = "<br>\n" ;
printf( "%'.-40.40s%'.2d%s" , $degisken1, $degisken2, $metin);
```

Bu komut Browser penceresinde şu görüntüyü oluşturur:

```
"İyilik üzerine.........................86"
```

Burada eklediğimiz ikinci "%'.2d" şeklineki biçim komutu ile, ikinci değişkenin değeri, en az sıfır en çok iki adet nokta ile doldurulmak ve sağa hizalanarak ondalık sayı olarak görüntülenmek üzere biçimlendiriliyor. Üçüncü biçim komutu olan "%s" ise üçüncü değişkenin sadece alfanümerik olarak muamele görmesini sağlıyor. Biçim komutlarının arasında boşluk bulunmaması, ait oldukları değişken değerlerinin de aralarına boşluk konmamasına sebep oluyor. Üçüncü değişkenin etkisini, kağıt üzerinde göremiyoruz; ancak bu Browser penceresinde bundan sonra gelecek unsurların bir satır aşağı kaymasını sağlayacaktır.

Dördüncü biçim özelliği, ondalık sayıların virgülden (veya nokdtadan) sonra ondalık bölümünün kaç hane olacağını belirler. Bunu da bir örnekle görelim:

```
$degisken = " 124 " ;
printf( "Değeri (ABD) $%.2f" , $degisken);
```

Bu biçimlendirme komutu da Browser penceresine şu yazıyı yazdırır:

```
Değeri (ABD) $124.00
```

str_replace ()

```
str_replace("<script","<yasak_script",$satir[mesaj]);
```

number_format()

```
$degisken = 1234567890.1234567890 ;
echo (number_format($degisken, 4 chr(44) , ".") ); //chr(44)=virgü
```

Bu deyimle 1234567890.1234567890 şeklindeki değer, Browser penceresine "1.234.567.890,1235" şeklinde yazdırılacaktır.

Düzenli İfadeler

^.+@.+\\..+$

işaretlerinin, Düzenli İfade işlemlerine ait olduğunu belirtilim. Bu işaretler ve onların arasına koyduğumuz karakter örnekleri ile, PHP'nin aradığımız bir metnin karakterlerinin hangi diziliş, sıralanış konumunda olduğuna bakarak, bize o metni bulmasını sağlarız; ya bu metni kullanırız, sileriz veya değiştiririz. Dolayısıyla, Düzenli İfade demek, bir diziliş, sıralanış biçimi demektir.

Eşleştirme deyimleri ve işaretler

PHP'nin karakter ve sıralanış eşlemede kullanılan düzenli ifade komutlarını kısaca ele alalım; sonra bunları kullanmamıza imkan veren fonksiyonları görelim.

^hakk

"hakk" ile başlayan bütün kelimeleri bulur.

edilemez$

Bu deyim ise PHP'ye "edilemez" ile biten bütün kelimeleri bulur

^hakkı$

PHP, başında ^ işareti, sonunda $ işareti bulunan kararter sıralanışını, aynen arar; yani bu deyim, birinci örnekteki üç cümleyi de bulamaz.

Hakk

Bu deyim ise her üç cümleyi de buldurur; çünkü üçünde de bu dört karakter bu sıralanışla mevcuttur. PHP'nin Düzenli İfadeleri, bütün rakam ve harfleri eşleştirebilir. Fakat sorun, özel karakterlerde çıkar. Sözgelimi, sekme işareti, satır sonlarında yeni-satır/satırbaşı işareti, gibi özel karakterleri, ancak önlerine Escape işareti olan ters bölü işaretini koyarak buluruz.

Düzenli İfadelerde Özel Karakterler

[\b]	Geri (Backspace) karakterini bulur.
\b	Belirtilen karakterle sınırlanan kelimeyi bulur: k\b, "hak mücadelesi" ifadesindeki birinci k'yı bulur; çünkü bu harf, bir kelime sınırlayıcıdır.
\B	Belirtilen karakterle sınırlanmayan kelime yoksa, başlayanı bulur: k\Bi, "üç kişi" ifadesindeki 'ki'yi bulur.
\cX	X yerine yazacağımız kontrol karakterini bulur. Örneğin, \cA, Ctrl+A'yı, \cZ ise Ctrl+Z'yi bulur.
\d	0'dan 9'ya kadar bir rakamı bulur: IE\d, her ikisi de herhangi bir rakamla biten "IE5" ve "IE4" değerlerini ikisini de bulur,
\D	Herhangi bir ondalık işaretini bulur.
\f	Form-feed (kağıt çıkart) karakterini bulur.
\n	Newline (yeni satır) karakterini bulur.
\r	Return (satırbaşı) karakterini bulur.
\s	Boşluk (space) bulur.
\S	Yatay ve düşey sekme, kağıt-çıkart, yeni satır, satırbaşı ve boşluk dışındaki herhangi bir karakteri bulur.
\t	Yatay sekme (Tab) karakterini bulur.
\v	Düşey sekme karakterini bulur.
\w	Herhangi bir harf, rakam veya alt-çizgiyi bulur.
\W	Harf, rakam ve alt-çizgi dışındaki karakteri bulur.
\xHex	Verilen 16 tabanlı (Hexadecimal) sayıya uygun Escape karakterini bulur. Örneğin, \n25, % işaretini bulur.

Bu arada noktalama işaretlerini arattırırken, önlerine ters bölü işareti koymak gerekir. Ters bömü işaretini de yine önüne ters bölü işareti koyarak (\\) arttırabilirsiniz.

Karakter Grupları

PHP'nin Düzenli İfadeleri'nde kolaylık sağlayan ve mesela ziyaretçinin bir Form'da bir INPUT etiketine verdiği yanıtların içinde olmaması veya olmaması gereken karakterleri bulmamıza imkan veren karakter grupları oluşturma yöntemini de kullanabiliriz. Sözgelimi bütün sesli hafleri aratmak için şöyle bir karakter grubu oluşturabiliriz:

[OoUuÖöAaOoEeıIiİ]

Karakter gruplarını köşeli parantez içinde yazarız. Bu deyimle, PHP, içinde herhangi bir sesli harf

bulunan bütün değerleri eşleştirecektir. Bu yöntemden yararlanarak, şu grupları kullanabiliriz:

[a-z]	Herhangi bir küçük harfi bulur.
[A-Z]	Herhangi bir büyük harfi bulur.
[a-zA-Z]	Herhangi bir büyük veya büyük harfi bulur.
[0-9]	Herhangi bir rakamı bulur.
[0-9\.\-]	Herhangi bir rakamı, noktayı veya kesme çizgisini bulur.
[\f\r\t\n]	Herhangi bir Form-feed (kağıt çıkart), Newline (yeni satır), Return (satırbaşı) karakterini veya boşluğu (space) bulur.

Sözgelimi, bir alfanümerik değer kümesinde b3, u2, n9 gibi birincisi küçük harf, ikincisi rakam olan iki karakterlik dizileri bulmak istiyorsak, arama grubunu şöyle kurarız:

^[a-z][0-9]$

Bu deyim PHP'ye, a'da z'ye küçük harfle başlayan, (^işareti aranan unsurun değerin başında olması gerektiğini söylüyor) ve sonunda 0'dan 9'a bir rakam bulunan kelimeleri bulmasını söyleyecektir. PHP, bu kelimenin sadece iki harfli olmasına dikkat edecektir; çünkü grubumuzun bir başı ve bir de sonu belirlendiğine göre, üç karakterli değerlerin bulunması imkanı yoktur.

^ işareti köşeli parantez içinde grup deyimi oluştururken kullanılırsa, bu olumsuzluk anlamı taşır. Sözgelimi, iki rakamlı ancak birinci karakteri rakam olmayan fakat ikinci karakteri rakam olan değerlerin bulunması için şu deyim gerekir:

^[^0-9][0-9]$

Burada en baştaki ^işareti "başında" demektir; ancak hemen arkasından gelen grupta "rakam olmayan" demiş oluyoruz; ikinci grup ve sonundaki $ işareti ile "rakamla biten" anlamına geliyor. Deyimde sadece baş ve sonu gösteren iki eşleştirme unsuru bulunduğuna göre bu deyim, "başında rakam olmayan, sonunda rakam olan iki karakterli değerleri" bulmaya yarayacaktır. Bu deyim söz gelimi 13'ü bulmayacak, fakat u2'yi bulacaktır. Bu yöntemle şu grupları yapabiliriz:

[^a-z]	Küçük harf olmayan herhangi bir harfi bulur.
[^A-Z]	Büyük harf olmayan herhangi bir harfi bulur.
[^\\\/\^]	\ , / veya ^ dışında herhangi bir karakteri bulur.
[^\"\']	Çift ve tek tırnak dışında herhangi bir karakteri bulur.

Grup oluşturmada kullandığımız özel karakterler de vardır. Örneğin nokta işareti (.), yeni satır başlangıcı olmayan herhangi bir karakter anlamına gelir. Dolayısıyla,

^.0$

deyimi yeni satırla başlamayan ve sıfır ile biten herhangi iki karakterli değeri bulacaktır.

Karakter eşleştirmede tekrar sayısı da bir özellik olarak kullanılabilir. Tekrar sayısı belirtmek için süslü parantez ({}) kullanırız.

Örnekler:

^a{4}$	İçinde sadece dört adet küçük a harfi bulunan kelimeleri seç: aaaa.
^a{2,4}$	İçinde sadece iki üç veya dört adet küçük a harfi bulunan kelimeleri seç: aa, aaa, aaaa gibi

^a{2, }	İki veya daha fazla küçük a harfi bulunan kelimeleri seç: haar, haaar, haaaar gibi. Bu deyim "har" kelimesini seçmez.
\t{2}	Ardarda iki sekme işaretini bul
.{2}	Herhangi çift karakteri bul: aa, &&, == gibi
^\-{0,1}[0-9]{1,}$	Negatif veya pozitif herhangi bir tam sayıyı bul
^[0-9]{1,}$	Pozitif herhangi bir tam sayıyı bul

Bu tür deyim oluşturma işlemleri giderek karmaşıklaşabilir. Örneğin:

^\-{0,1}[0-9]{0, }\.{0,1}[0-9]{0, }$

Bu karmaşık deyim aslında sadece "Negatif veya pozitif bir ondalık (double) değeri bul," anlamına geliyor. Kısaca irdelersek, aranan degerin sıfır veya bir kere tekrarlanan bir kesme çizgisiyle başlayabileceğini ("Sıfır veya bir kere" demek, olsa da olur, olmasa da anlamına geliyor!) bunu sıfır veya daha fazla kere tekrarlanan bir rakamın izleyebileceğini, onu da sıfır veya bir kere tekrarlanan bir nokta işareti ile sonunda sıfır veya daha fazla kere tekrarlanan herhangi bir rakamın izleyebileceğini söylemiş oluyoruz.

PHP bu tür karmaşık ifadelerin hatasız yazılmasını sağlayan kısayollara sahiptir. Bunları sıralayalım:

?	{0,1} anlamına gelir. Kendisinden önce yer alan unsurun en az sıfır en çok bir kere tekrar edilmesi gerektiğini (olmayabileceğini ama olursa en fazla bir kere olabileceğini) belirtir.
*****	{0, } anlamına gelir. Kendisinden önce yer alan unsurun sıfır veya daha fazla kere tekrar edilmesi gerektiğini (tümüyle opsiyonel olduğunu) belirtir.
+	{1, } anlamına gelir. Kendisinden önce yer alan unsurun en az bir veya daha çok kere tekrar edilmesi gerektiğini (bulunmasının zorunlu olduğunu) belirtir.

Bu kısa-yolları kullanarak, yukarıdaki karmaşık ifadeleri basitleştirelim:

^[a-zA-Z0-9_]+Ş	En az bir harf veya rakam veya altçizgi içeren herhangi bir kelime
^[0-9]+Ş	Herhangi bir pozitif tamsayı
^\-?[0-9]+Ş	Herhangi bir tamsayı
^\-?[0-9]*\.[0-9*$]+Ş	Herhangi bir kesinli (double) sayı

Bir Düzenli İfade'nin yazılışında birden fazla arama-sıralanış deyimine yer verebiliriz. Bunu yapmamızı sağlayan | işaretidir. Örneğin,

\.com|\.co\.uk

ifadesi ile, ya ".com" ya da ".co.uk" değerlerinin bulunmasını sağlayabiliriz. Burada | işareti "veya" kelimesi gibi düşünebilirsiniz.

Düzenli ifadeler yoluyla INPUT etiketinden gelen değerleri incelerken hata yapmak kolaydır. Bunun için kendi ifadelerinizi mutlaka sçeşitli olasılıklara karşı sınamalısınız. Bu bölümün başında örnek olarak verdiğimiz Düzenli İfade'yi hatırlıyor musunuz?

^.+@.+\\..+$

Örneğin bu ifade, ziyaretçinin elektronik posta adresini yazması gereken bir INPUT etiketinin sağladığı değerin gerçekten elektronik adres biçimi taşıyıp taşımadığını sınar. Baştaki ^ ve nokta

işaretleri ile artı işareti değerin önünde boşluk olmamasını sağlıyor; @ işareti ise değerin içinde @ bulunması gerektiğine işaret ediyor. Tekrar eden nokta ve artı işaretleri "ne kadar olursa olsun ve ne olursa olsun" anlamına geliyor. Bunu izleyen nokta karakterini gösteren (\.) işaret buralarda bir de gerçekten nokta olması gerektiğini ve bunu izleyen nokta ve artı tekrar "ne olursa olsun, ne kadar olursa olsun" anlamını taşıyor. Başka bir deyişle, aradığımız değerin "herhangi bir şey" @ "herhangi bir şey daha" . "birşeyler daha" şeklinde olduğunu belirtmiş oluyoruz. Ne var ki deyimiçinde iki nokta veya iki @ işareti olan veya @ işareti ile nokta arasında bir şey bulunmayan veya @ veya noktadan öncesi ya da sonrası boş olan bütün değerleri safdışı etmeye yetmeyecektir.

Düzenli İfade Fonksiyonları

Yukarıda öğrendiğimiz Düzenli İfade yazma tekniklerini, PHP'nin bize sağladığı beş fonksiyonda parametre olarak kullanırız. PHP'nin ayrıca Perl-tarzı düzenli ifade fonksiyonları da vardır. Bu fonksiyonlardan, ya bize bir boolean (doğru/yanlış) değer döner; ya da fonksiyon istediğimiz işi yaparak vardığı sonuçları verdiğimiz değişkene yazar. Biz, daha sonra bu değere bakarak veya değişkenin değerlerini kullanarak, PHP programımızın akışını kontrol edebiliriz. Burada ele alacağımız fonksiyonlara ilişkin örneklerde, daha önceki bölümlerde oluşturduğumuz konuk defteri programı ile Web ziyaretçilerimizin sunucuya göndereceği bilgileri doğrulamaya ve muhtemel zararlı kodlardan ayıklamaya çalışacağız.

ereg() ve eregi()

PHP'nin temel Düzenli İfade Fonksiyonu, ereg(), arattığımız karakter sıralanışı bulunduğu taktirde doğru, bulamadığı taktirde yanlış karşılığı bir değer verir. Fonksiyonu şöyle yazarız:

```
$bir_degisken = ereg("eşleştirilecek_sıra" , $kaynak , $yeni_değişken);
```

Fonksiyonun aradığımız eşleştirmeyi yapması halinde, buradaki $bir_degisken'in değeri true/doğru, yapamaması halinde false/yanlış olacaktır. Eşleştirme sırasının nasıl oluşturulduğunu yukarıda gördük; bu ifadelerden işimize uygun olanı buraya tırnak içinde yazarız. $kaynak, eşleştirilecek sıralamanın içinde aranacağı değeri tutan değişkendir. Fonksiyonun bir diğer becerisi, eğer eşleştirilecek sıralamayı gruplar halinde verirsek, kaynakta yapacağı eşleştirme olursa, buna uygun değerleri bir dizi değişkene yazabilmesidir; istersek bir parametre olarak bu yeni değişkenin almasını istediğimiz adı veririz; böylece eşleştirme sonucu bulunan değerler kaydedilmiş olur.

eregi(), aynen ereg() fonksiyonu gibi çalışır; sadece eşleştireceği değerlerde büyük-harf/küçük-harf farkı gözetmez.

Daha önceki bölümde oluşturduğumuz ve kd_01.php adıyla kaydettiğimiz konuk defteri programının akış planını, ziyaretçinin Form'a yazdığı ve sunucuda $HTTP_POST_VARS dizi-değişkeninde tutulan değişkenlerinden elektronik posta adresi ilge ilgili olanı gerçekten içinde en az bir @ işareti ile en az bir adet nokta içip içermediğine bakarak sınayabiliriz. Böyle bir sınama için gerekli kod şöyle olabilir:

```
if (eregi("^.+@.+\\..+$", $adres, $email)) {
        }
        else {
```

```
$hata = "Elektronik posta adresinizde bir hata var!<br>";
                echo $hata;
                include("kd_hata_halinde.htm");
                exit;
}
```

Program, bu örnekte $adres değişkeninde kayıtlı değerin içinde aradığı sıralamayı bulursa, eşleşen değeri $email adlı yeni bir değişkene yazacak ve if sınavının sonucu doğru olacaktır. Bu sıralamaya uygun bir değer bulunamazsa, if sınavı else deyimine atlayacak ve bir hata mesajı üretilerek, bu program durdurulacaktır.

ereg_replace() ve eregi_replace()

Gördüğümüz gibi, ereg() arattığımız karakter sıralanışı bulunduğu taktirde doğru, bulamadığı taktirde yanlış karşılığı verdikten sonraduruyor! Oysa kimi zaman arattığımız ve bulunan değerin başka bir değierle değiştirilmesi gerekebilir. Bunun için **ereg_replace()** ve eregi_replace() fonksiyonlarını kullanırıız:

```
ereg_replace("eşleştirilecek_sıra" , yeni_metin , $kaynak);
```

Fonksiyonun aradığımız eşleştirmeyi bulursa, bu değerin yerine verdiğimiz yeni metni koyacaktır; yeni metni bir değişkenin değeri olarak da verebiliriz. Uygulama örneği için yine konuk defteri örneğine dönelim. Ziyaretçilerimiz kimi zaman yanlışlıkla, kimi zaman pek de iyi niyet sonucu olmadan, kendilerinden beklediğimiz isim, adres ve mesaj yerine sunucu veya başka ziyaretçilerin Browser programları tarafından kod gibi algılanacak metinler yazabilirler. Burada sadece bu tür zararlı metinlerin genellikle programlarda bulunması gereken karakterler içerdiğini söylemekle yetinelim. Bu tür karakterlerin başında < ve > işaretleri bulunur! Dolayısıyla, biz de ziyaretçimizden gelecek verilerin yazıldığı değişkenlerin değerlerinde bu işaretleri aratabilir ve bunları içi boş bir alfanümerik değer ile değiştirebilir; yani silebilir. Zararlı olabilecek kodların arasında daha bir çok karakter bulunabilir; ancak Script diliyle yazılması gereken bu kodlardan < ve > işaretlerini kaldırılması kodları işlemez hale getireceği için, şu aşağıdaki örnek yeterli olabilir:

```
$adi = ereg_replace("<","",$adi);
$adi = ereg_replace(">","",$adi);
$adres = ereg_replace("<","",$adres);
$adres = ereg_replace(">","",$adres);
$mesaj = ereg_replace("<","",$mesaj);
$mesaj = ereg_replace(">","",$mesaj);
```

Burada ereg_replace() fonksiyonu, ziyaretçiden gelecek üç değişkenin değerlerinde < ve > işaretlerini aramakta onların yerine içi boş bir metin ("") yazmaktardır.

split()

Düzenli İfade ile çalışan bu fonksiyon, vereceğimiz eşleştirme sıralamasını sınırlayıcı olarak kullanarak, belirteceğimiz değerde bulduğu değer parçalarını ayırır ve bunları ayrı ayrı bir dizi değişkenin elemanları olarak kaydeder. Bu fonksiyonu şöyle yazarız:

```
$yeni_dizi_değişken = split("eşleştirilecek_sıra" , $kaynak, sınır_sayısı);
```

Fonksiyon, aradığı sıralamayı bulamazsa, false/yanlış sonucunu verir. Burada sınır sayısı olarak

vereceğimiz rakam, oluşturulacak yeni dizi değişkene en fazla kaç eleman yazılmasını istediğimizi gösterir. Bu sayıyı vermezsek, PHP yeni dizi değişkenin gerektiği kadar elemana sahip olmasını sağlar. Bir örnek vererek, bu fonksiyonu nasıl kullanibileceğimizi görelim:

```
$metin = "İnsan sözüyle kendini gösterir, davranışlarıyla ruh halini aksettirir.";
$aranan = " ";
$yeni_dizi_değişken = split($aranan, $metin);
foreach ($yeni_dizi_değişken as $eleman) {
print "$eleman <br>";
}
```

Bu programda PHP, $metin değişkeninin içerdiği değerde $aranan değişkeninin içerdiği değeri, yani boşluğu, eşleştirilecek unsur olarak kullanacak ve $metin değişkeninin değerini boşluklarından parçalara ayıracaktır. Ayrılıcak her yeni parça, $yeni_dizi_değişken adlı değişkenin elemanları olarak atanacaktır. Programın geri kalan kısmı ise, bu yeni dizinin elemanlarını görüntülemekterdir.

sql_regcase()

İçinde büyük harf-küçük harf ayrımı olan bir değeri büyük harf-küçük harf ayrımı olmayan Düzenli İfadeler haline çevirir. Bu fonksiyon bizden Düzenli İfade almaz, tersine Düzenli İfade oluşturur. Örnek:

```
<?php
$metin = "Sözler";
echo(sql_regcase($metin);
?>
```

Bu program, Browser penceresine şu metni yazdırır:
[Ss][Öö][Zz][Ll][Ee][Rr]

Tarih ve saat Verisi

PHP, o andaki zaman bilgisini, saat, dakika, saniye ve salise olarak; tarih bilgisini yıl, ay, gün (sayı veya isim olarak), programımızın herhangi bir yerinde bize bildirebilir. Bu bilgiyi Web sunucusunda istediğimiz anda, muhtemelen sunucunun bulunduğu bilgisayarın sistem saatinden alacak olan PHP, sunucu programında farklı bölgesel ayarlar için gerekli düzenleme yapılmışsa, bu imkandan yararlanarak bize sunucunun değil, arzu ettiğimiz bölgenin saat ve tarihini bildirebilir.

Özellikle Türkiye'de olmayan bir sunucuda bu imkanın bulunup bulunmadığını, ancak sınayarak veya sistem yöneticisine sorarak öğrenebiliriz. Böylebir sıNama için şu kodları programınızın başına koyun:

```
<?php
setlocale ("LC_TIME", "TR");
print (strftime ("Türkçe bugün günlerden: %A "));
?>
```

Browser penceresinde "Türkçe bugün günlerden Sunday" yazısını okursanız, sunucuda Türkçe için bölgesel ayar desteği yok demektir!

PHP'nin zaman ve tarih belirlemekte kullanabileceğiniz başlıca fonksiyonu **getdate()** ise şöyle kullanılır.

getdate() Tarih ve saat bilgisini alır ve ve vereceğiniz bir isimdeki dizi-değişkende kaydeder. Örnek:

```
$saat_tarih = getdate()
```

Bu durumda, $saat_tarih dizi değişkeninde sırasıyla şu bilgiler yer alır:

32	saniye
57	dakika
6	saat
30	ayın gün sayısı (1-31)
0	haftanın gün sayısı (1-7)
7	ayın sayısı (1-12)
2000	yıl
211	yılın kaçıncı günü
Sunday	günün adı
July	ayın adı
964929452	Unix sistemlerinde Epoch biçiminde zaman bilgisi

PHP'nin tarih ve saat bilgisini biçimlendirmede yararlandığımız date() fonksiyonunu daha sonra metin biçimlerdirme bölümünde ele alacağız

Tarih ve Saat Düzenleme

PHP'nin tarih ve zaman bilgisini kullanmamızı sağlayan getdate() fonksiyonunun yanı sıra, elde ettiğimiz bilgileri çok daha esnek biçimlendirmemizi sağlayan date() fonksiyonu da sık kullanılır. Bu fonksiyon çağrıldığı yerde bize Unix sisteminin Epoch zaman-tarih damgasını verir. Bu verinin biçimlendirilmesi için çeşitli parametreler vardır. Bu parametreler, gün adlarının dili ve tarihlerin yazılışı bakımından Web sunucusunun bulunduğu bilgisayardaki bölgesel ayarlara göre sonuç verir.

date() $tarih date(biçimlendirme_parametreleri);

Burada biçimlendirme_parametleri yerine şunları yazabilirsiniz:

a 12 saat esasına dayanan Anglo-Sakson sistemlerinde öğleden önce ("am") veya öğleden sonra ("pm") işaretinin verilmesini sağlar.
A Aynı işaretlerin büyük harfle yazılmasını sağlar.
d İki haneli gün sayısı, tek haneli günlerin önüne sıfıır konur: "01" - "31"
D Üç haneli gün adı kısatması: "Cum"
F Uzun ay adı: "Ocak"
h 12 saatlik sistemde saat: "01" - "12"
H 24 saatlik sistemde saat: "00" - "23"
g 12 saatlik sistemde tek haneli saatlerin önüne sıfır konmadan saat: "1" - "12"
G 24 saatlik sistemde tek haneli saatlerin önüne sıfır konmadan saat: "0" - "23"
i Dakika: "00" - "59"
j Tek haneli sayıların önüne sıfır konmadan gün sayısı "1" - "31"

l (küçük L harfi) Uzun gün adı: "Cuma"
L Artık yıl olup olmadığına ilişkin Boolean (doğru/yanlış) değişken. Artık yıl ise 1, değilse 0.
m Tek hanelilerin önüne sıfır konarak ay sayısı: "01" - "12"
n Tek hanelilerin önüne sıfır konmadan ay sayısı: "1" - "12"
M Kısaltılmış ay adı: "Şub"
s Saniye: "00" - "59"
S İngilizce ('ncı anlamına) 2 karakter ek: "th", "nd"
t Belirtilen ayın gün sayısı; "28" - "31"
w Haftanın gün sayısı: "0" (Pazar veya Pazartesi) - "6" (Cumartesi veya Pazar)
Y Dört haneli yıl: "2000"
y İki haneli yıl "00"
z Yılın gün sayısı: "0" - "365"

Örnek:

print (date ("l dS of F Y h:i:s A"));

Browser penceresine İngilizce bölgesel ayarlar yapılmış bir bilgisayara kurulu Web sunucusunda: "Sunday 30th of July 2000 07:51:08 AM" yazdırır.

print (date ("l, d F Y g:i:s"));

Browser penceresine Türkçe bölgesel ayarlar yapılmış bir bilgisayara kurulu Web sunucusunda: "Pazar, 30 Temmuz 2000 07:51:08" yazdırır.
date() ve mktime() fonksiyonlarını birlikte kullanarak geçmiş veya gelecek tarihleri bulma imkanı de vardır.

Örnek:
```
$yarin = mktime (0,0,0,date("m") ,date("d")+1,date("Y"));
$gecen_ay = mktime (0,0,0,date("m")-1,date("d"), date("Y"));
$gelecek_lyl = mktime (0,0,0,date("m"), date("d"), date("Y")+1);
```

PHP'de Program Denetimi

if Deyimi

```
if ( koşullar ) {
koşullar doğru ise yapılacak işlere ilişkin komutlar
}
elseif (diğer koşullar) {
diğer koşullar doğru ise yapılacak işlere ilişkin komutlar
}
else {
diğer her durumda yapılacak işlere ilişkin komutlar
}
```

```
<?php
        if ( $parola == "" ) {
        echo ("Sitemize girmek için parola yazmanız gerekir.<br>");
        echo ("Lütfen parolayı yazın! <br>");
```

```
        }
?>
```

switch deyimi

```
switch ( değişken ) {
case KOŞUL-1 ;
        Koşul-1 doğru ise yapılacak işlere ilişkin komutlar
break;
case KOŞUL-2 ;
        Koşul-2 doğru ise yapılacak işlere ilişkin komutlar
break;
case KOŞUL-3 ;
        Koşul-3 doğru ise yapılacak işlere ilişkin komutlar
break;
case KOŞUL-4 ;
        Koşul-4 doğru ise yapılacak işlere ilişkin komutlar
break;
.....
.....
default:
        diğer her durumda yapılacak işlere ilişkin komutlar
}
```

switch için kısa yol

```
<?php
$uyari = ($parola == "" ) ? "Parola yazmanız gerekir" : "Teşekkür ederiz" ;
echo ($uyari);
?>
```

Bu kod parçacığı, ziyaretçinin parola girip girmediğini $parola değişkeninin içinin boş olup olmadığına bakarak anlayacak ve $parola değişkenin içi boş ise (yani soru işaretinin sorguladığı durumun doğru olması halinde) iki nokta üstüskte işaretinden önceki metni $uyarı değişkenin içeriği haline getirecek; $parola değişkeninin içi dolu ise (yani koşul yerine gelmiyorsa, durum yanlış ise) iki nokta üstüste işaretinden sonraki metni $uyarı değişkeninin içeriği yapacaktır. Bir sonraki echo() komutu ise içeriği bu sınav sonucuna göre belirlennen $uyarı değişkeninin değerini Browser penceresinde görüntüleyecektir.

while döngüsü

```
while (koşul) {
        Koşul doğru ise yapılacak işlere ilişkin komutlar
        }
```

```
<?php
	$sayac = 1;
	while ( $sayac <= 7 ) {
	print ("<font size= $sayac >");
	print ("<b><p>İyileri iyilikleri ile alkışlayınız!</b></p>");
	print ("</font>");
	$sayac ++;
	}
?>
```

do..while

```
	do {
	Koşul doğru ise yapılacak işlere ilişkin komutlar
	}
```

while (koşul);

```
	$sayac = 1;
	do {
	print ("<font size= $sayac >");
	print ("<b><p>İyileri iyilikleri ile alkışlayınız!</b></p>");
	print ("</font>");
	$sayac ++;
	}
	while ( $sayac <= 7 ) ;
```

for döngüsü

```
	for ( $yeni_degisken atama ; koşul ; artış basaması ) {
	Koşul doğru ise yapılacak işlere ilişkin komutlar
	}
```

```
<?php
	for ($sayac = 1; $sayac <= 7 ; $sayac++ ) {
	print ("<font size= $sayac >");
	print ("<b><p>İyileri iyilikleri ile alkışlayınız!</b></p>");
	print ("</font>");
	}
?>
```

Döngüyü sona erdirmek için: break

```
<?php
        $tekrar = 10 :
        for ($sayac = 1; $sayac <= $tekrar ; $sayac++ ) {
        if ( $tekrar <= 0 )
        break;
        print ("<font size= $sayac >");
        print ("<b><p>İyileri iyilikleri ile alkışlayınız!</b></p>");
        print ("</font>");
        }
?>
```

Döngüyü sürdürmek için: continue

```
<?php
        $sayac = -5 :
        for (; $sayac <= 7 ; $sayac++ ) {
        if ( $sayac <= 0 )
        continue;
        print ("<font size= $sayac >");
        print ("<b><p>İyileri iyilikleri ile alkışlayınız!</b></p>");
        print ("</font>");
        }
?>
```

Bu durumda PHP, conktinue komutunu gördüğü anda for döngüsünün geri kalan kısmını icra etmek yerine başa dönecek ve döngüyü yeniden icra etmeye başlayacaktır. Döngünün ilk satırınnda kod parçasını, bir sayfa kodunun içine yerleştirir ve önce bu şekliyle, danha sonra $sayac= 1 yazarak sınarsanız, continue komutunun programı sayaç değişkeninin değeri 1 oluncaya kadar durdurduğunu ama bu sırada for döngüsünün devam ettiğini göreceksiniz. ($sayac değişkeninin değerini, Form yoluyla ziyaretçiden nasıl alacağımızı daha sonra göreceğiz.)

Fonksiyonlar

Fonksiyon Tanımlama ve Çağırma

PHP'de fonksiyonlar function komutu ile oluşturulur. Tanımladığımız fonksiyon, kendisini göreve çağıracak komuittan, yapacağı işlemde kullanmak üzere değer alacaksa, bu değerlere vereceğimiz değişken isimleri fonksiyon adının yanında parantez içinde gösterilir. Fonksiyon birden fazla değer bekleyecekse, bunların değişken adlarının arasına virgül koyarız. Fonksiyona ulaştırılan değerlere argüman denir. Fonksiyon, kendisine bir değer ulaştırılmasını beklemese bile içi boş parantez koymamız gerekir. Buna göre PHP'de fonksiyon şöyle yazılır:

function fonksiyonun_adı (argüman1, argüman2, ... argümanN) {

```
        fonksiyonun yapacağı işe ilişkin komutlar
        }
```

Fonksiyon adları, değişken adları ile aynı kurallara tabidir. Fonksiyon içinden fonksiyon çağrılabilir; fonksiyonların içinde PHP'nin kendi fonksiyonları kullanılabilir. Şimdi, ilk örnek fonksiyonumuzu yazalım.

Yukarıdaki örneklerde, PHP'ye, bir Web sayfası oluştururken HTML ögesi olacak bazı satırların başına <H1>, satır sonlarına <BR> gibi etiketler koydurduğumuza dikkat etmiş olmalısınız. Bir programda bu işi sık sık yapacaksak, yani metinlerimiz <BR> ile bitecek, bazı başlıklarımız ise sayfamıza HTML'in başlık etiketleri ile gönderilecekse, bu etiketleri defalarca yazmamız sadece zorluk değil, fakat aynı zamanda yazım hatasını davet eden bir davranır olur. PHP'de kendisine vereceğimiz metinleri başlık veya gövde yazısı olarak biçimlendirecek fonksiyonlar tanımlayarak, programcılıkta daima tehlikeli olan insan unsurunu asgarîye indirebiliriz. Şu programı yazdir01.php adıyla kaydederek, Browser'da açın:

```
<?php
function yazdirBR ($metin) {
        print ("$metin<br>\n");
        }
function yazdirH1 ($metin) {
        print ("<h1>$metin</h1>\n");
        }
function yazdirH2 ($metin) {
        print ("<h2>$metin</h2>\n");
        }
function yazdirH3 ($metin) {
        print ("<h3>$metin</h3>\n");
        }
function yazdirH4 ($metin) {
        print ("<h4>$metin</h4>\n");
        }
function yazdirP ($metin) {
        print ("<p>$metin</p>\n");
        }
// Başka kodlar buraya girebilir
yazdirH1("Bu H1 Başlık");
yazdirH2("Bu H2 Başlık");
yazdirH3("Bu H3 Başlık");
yazdirH4("Bu H4 Başlık");
yazdirBR("Bu kendisinden sonra BR olan birinci metin.");
yazdirBR("Bu kendisinden sonra BR olan ikinci metin.");
yazdirP("Bu uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun
uzun uzun uzun uzun uzun uzun uzun uzun uzun bir paragraf metni.");
yazdirP("Bu uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun uzun
uzun uzun uzun uzun uzun uzun uzun uzun uzun bir diğer paragraf metni.");
?>
</BODY>
</HTML>
```

Bu programda, yazdirBR(), yazdirH1(), yazdirH3(), yazdirH4(), ve yazdirP() adlarıyla altı fonksiyon tanımladığımızı görüyorsunuz. Bu fonksiyonların hepsi kendilerini göreve çağıran satırdan, kendilerine bir değer verilmesini istiyorlar ve bu değeri $metin adlı değişkende tutuyorlar. Fonksiyonlarımız tümü de PHP'nin print() fonksiyonundan yararlanıyor; ancak bu fonksiyonun nasıl kullanılacağını, nasıl işletileceğini de belirliyorlar. Buna göre bazı fonksiyonlarımız $metin adlı değişkenin değerini önüne ve arkasına bir HTML etiketi koyarak Browser'a gönderiyor; birisi ise sadece $metin değişkenin tuttuğu değerin sonuna bir HTML etiketi koyduruyor. Fonksiyonlarımı buradaki örnekte olduğu gibi hemen oluşturulduktan sonra göreve çağırmayız. Geleneksel olarak, bir programda kullanılacak fonksiyonlar, programın baş tarafında toplanır ve daha sonra nerede gerekirse orada, çağrılırlar. Fonksiyonları adlarını ve kendilerine verilmesi gereken bir değer varsa o değeri parantez içinde yazarak çağırırız. Burada olduğu gibi, fonksiyonu göreve çağırırken parantez içinde değerin kendisini yazabileceğimiz gibi, bu değeri tutan bir değişkenin adını da yazabiliriz.

```
<?php
function topla ($sayi1, $sayi2) {
        $sonuc = $sayi1 + $sayi2;
        return $sonuc;
        }
function cikart ($sayi1, $sayi2) {
        $sonuc = $sayi1 - $sayi2;
        return $sonuc;
        }
function carp ($sayi1, $sayi2) {
        $sonuc = $sayi1 * $sayi2;
        return $sonuc;
        }
function bol ($sayi1, $sayi2) {
        $sonuc = $sayi1 / $sayi2;
        return $sonuc;
        }
// Başka kodlar buraya girebilir
$sayi1 = 12;
$sayi2 = 5;
        print topla($sayi1, $sayi2);
        print ("<br>");
        print cikart($sayi1, $sayi2);
        print ("<br>");
        print carp($sayi1, $sayi2);
        print ("<br>");
        print bol($sayi1, $sayi2);
        print ("<br>");
?>
```

Bu programda, dört aritmetik işlemi yapan dört ayrı fonksiyon tanımlıyoruz. Fonksiyonlarımız kendilerini göreve çağıran komuttan, kendilerine iki değer vermesini bekliyorlar ve bu değerleri $sayi1 ve $sayi2 adlı değişkenlere yazıyorlar. Sonra herbiri, kendisinden beklenen aritmetik işlemi yaparak, sonucunu $sonuc adlı değişkene yazıyor. Burada dikkat edeceğimiz nokta, return komutudur. Bu komut, f onksiyonun elde ettiği değeri, değeri tutan değişkenin adıyla, fonksiyonu çağırmış olan satıra gönderir. return komutuyla, kendisini göreve çağıran satıra değil fakat mesela başka bir fonksiyona da değer gönderebiliriz:

return (baska_fonksiynon ($degisken)) ;

Fonksiyonun return satırında böyle bir başka fonksiyonun adı yazmıyorsa, bulunan değer göreve çağıran satıra gönderilir.
Bu program, içine bizim yazdığımız iki değeri hesaplayarak, Browser'a gönderecektir. Ancak bu değerler ziyaretçinin dolduracağı bir formdan alınabileceği gibi, program tarafından da hesaplanabilir

Fonksiyona varsayılan değer verebiliriz

Buradaki örneklerde tanımladığımız fonksiyonlara bekledikleri değerleri, onları göreve çağırdığımız noktada biz veriyoruz. Ancak öyle durumlar olabilir ki, fonksiyonun beklediği değerlerden biri veya bir kaçı, göreve çağıran satır tarafından verilmeyebilir; fonksiyon varsayılan bir değerle çalıştırılabilir.

```
<?php
function yazdir ($metin, $boyut=3) {
        print ("<font size=\"$boyut\">$metin</font><br>");
        }
// Başka kodlar buraya girebilir
yazdir("Bu Başlık", 5 );
yazdir("Bu küçük boyutta bir metin", 2);
yazdir("Bu varsayılan boyutta bir metin");
yazdir("Bu çok büyük Başlık", 8);
yazdir("Bu uzun bir paragraf metni. Boyutu varsayılan ölçüde. Bu uzun bir paragraf metni.
Boyutu varsayılan ölçüde. Bu uzun bir paragraf metni. Boyutu varsayılan ölçüde.");
?>
```

Bu örnekte, daha öncekilerden farklı olarak fonksiyonun, beklediği iki argümandan birisini, fonksiyonu tanımlarken verdiğimize dikkat edin: " function yazdir ($metin, $boyut=3)" ifadesi, PHP'ye, "Bu fonksiyona ikinci argüman eksik gönderilirse, telaşa kapılma, onun yerine 3 rakamını kullan!" anlamına gelir. Nitekim, programın daha ilerdeki bölümlerinde bu fonksiyon görevlendirilirken birinci argümanın değeri olan metin verildiği halde, iki ayrı yerde ikinci argümanın değeri verilmiyor. Bu iki durumda., PHP, yazdir() fonksiyonunda varsayılan değer olan 3'ü kullanıyor.

Değişkenlerin kapsamı: global ve static

dört fonksiyonda da aynı değişken adlarını kullanıyoruz ve sonucu aynı isimle print() fonksiyonuna gönderiyoruz. PHP nasıl oluyor da, aynı isimli değişkenleri buradaki gibi ayrı ayrı değerlerle ele alabiliyor? Bu sorunun cevabını verebilmek için değişkenlerin kapsam alanına bakmamız ve ve bu arada global deyimi ile tanışmamız gerekir.

Bir fonksiyonun değişkenleri, sadece o fonksiyonun ömrü süresince vardır; hiç bir fonksiyon diğer bir fonksiyonun değişkenlerinin veya kendisine verilmemiş bir başka değişkenin değerini bilemez; kullanamaz.

```
<?php
$metin = "Başkalarına yararlı olmanın sınırı yoktur!";
function yazdir () {
        print ("<h1>İşte metin: $metin </h1>");
        }
// Başka kodlar buraya girebilir
```

```
        yazdir();
?>
```

Normal görünüşlü bir fonksiyon ve kendisinden önce tanımlanmış olan $metin adlı değişkenin tuttuğu değeri, kendisi göreve çağrıldığı anda Browser penceresinde görüntülemeye hazır görünüyor! Fonksiyonun dışarıdan argüman istemediğini, kullanacağı print() komutunun konusu olarak $metin değişkeninin değerini kullanacağını da anlıyabiliyoruz. Fakat bu programı çalıştırdığımızda karşımıza çıkan manzara çok farklı oluyor.

Fonksiyonun çalıştığını, yazdırması beklenen şeylerin ilk bölümünü yazdırmış olmasından anlayabiliriz. Fakat, $metin değişkeninin değeri nerede? Şimdi bu sorunun cevabını biliyoruz: $metin değişkeni fonksiyonun kapsama alanı (scope) dışında! Bir fonksiyon kendisine verilmeyen veya kendi içinde tanımlanmayan bir değişkeni kullanamaz; bilemez, değerinden haberi olmaz. Nitekim bu görüntüyü aldığınız sırada kaynak kodlarına bakarsanız, "<h1>..</h1>" etiketlerinin arasında sadece "İşte metin:" kelimelerinin bulunduğunu göreceksiniz.

Bir fonksiyonun dışında tanımladığımız değişkenlerimizi fonksiyona tanıtabilmek için global deyimini kullanırız; böylece değişken bütün program boyuinca küresel nitelik kazanmış olur:

```
function yazdir () {
        global $metin;
        print ("<h1>İşte metin: $metin </h1>");
        }
```

Bu noktada dikkat etmemiz gereken şey, global deyimi ile kendisine kullanılmak üzere verilen değişken, artık bir bakıma fonksiyonun malı olur ve fonksiyon tarafından değeri değiştirilebilir. Bütün program boyunca kullanmaya niyetli olduğunuz global değişkenlerin, kullanımına verildiği fonksiyon tarafından değiştirilip-değiştirilmediğine dikkat edin.
Bir değişkenin bütün programda gerekli olmadığı ve sadece bir fonksiyon içinde kullanılacağı durumlarda bu değişkeni fonksiyonun içinde tanımlamak daha doğru olur. Fakat daha önce belirttiğimiz gibi fonksiyonların içinde tanımlanan değişkenler fonksiyon çalışırken var olur; fonksiyon sona erdiğinde de ölür. Başka bir deyişle, bir fonksiyonun içinde oluşturduğumuz bir değişkenin fonksiyon sona erdiğinde sıfırlanmaması için bir çare olması gerekir. Bu çarenin adı, static deyimidir. Şöyle bir örnek düşünelim: Sitemizdeki bir hesaplama fonksiyonu ziyaretçilerimizin yeni alışverişlerinin toplamını eski toplama ekleyerek, yeni ana toplamı buluyor ve ziyaretçi alışveriş sepetine yeni bir mal ekledikçe, ve o andaki ana toplamın ne olduğunu öğrenmek istedikçe, fonksiyonumuzun eski ana toplamı hatırlaması gerekiyor. Şu andaki PHP bilgimizle böyle bir program yazmaya kalkmayalım;

```
<?php
function saydir () {
        static $sayi = 0;
        $sayi++;
        print ("<h3>Fonksiyonun tuttuğu sayı: $sayi </h3>");
        }
// Başka kodlar buraya girebilir
        print ("<h2>Fonksiyonun birinci kez çağrılması:</h2>");
        saydir();
        print ("<h2>Fonksiyonun ikinci kez çağrılması:</h2>");
        saydir();
```

```
        print ("<h2>Fonksiyonun üçüncü kez çağrılması:</h2>");
        saydir();
        print ("<h2>Fonksiyonun dördüncü kez çağrılması:</h2>");
        saydir();
?>
```

Bu programı, static deyimi olmadan çalıştıracak olursak, saydir() fonksiyonu her çağrıldığında $sayı değişkeninin baştan tanımlandığını ve bir türlü 1'den ileri gidemediğini göreceğiz. static ise fonksiyon bittikten sonra $sayi değişkeninin değerinin değişmeden kalmasını sağlayacaktır.

Şimdi, bir PHP programının gerektirdiği hemen herşeyi görmüş olduk; sadece iki özel araç, dizi değişkenler ve nesneler (Objects) kaldı. Bunları da gelecek bölümde ele aldıktan sonra, PHP'yi gerçek ortamda, Web'de kullanmaya başlayabiliriz.

Browser bilgilerini çekme

```
<?php
foreach ($GLOBALS as $anahtar=>$deger ) {
        print ($anahtar . " = " . $deger . "<br>");
        }
?>
```

Şimdi Browser penceresinde gördüğünüz bilgileri irdeleyelim; çünkü biraz sonra ziyaretçiden Form ile gelen verileri yakalarken ve işlerken bu bilgilerden yararlanacağız. (Bu programı kişisel bilgisayarınızda, kişisel Web sunucuda çalıştırıyorsanız, aynı programın gerçek bir Unix-tabanlı Web sunucuda (Apache) nasıl sonuç verdiğini

Bu programla PHP'nin daima varolan $GLOBALS dizisinin üyelerini görüntülüyoruz. $GLOBALS bir ilişkili-dizi-değişken, yani değişken değerlerinin endeks adı (anahtarı) bulunan bir dizi olduğu için, içerdiği değerlere adları ile ulaşabiliriz. Bu programda, $GLOBALS'ın anahtarlarını $anahtar, değerlerini ise $değer değişkenine yazdırıyoruz ve bir foreach döngüsü ile Brnowser penceresine gönderiyoruz. Programı çalıştırdığımız sisteme ve Web sunucu programına bağlı olmak üzere, ekranımızda bir çok değişken görebiliriz. Bunlar arasında bütün HTTP Server programları için ortak ve Web programcısı için önemli değişkenler şunlardır:

HTTP_ENV_VARS HTTP Sunucu programın çalışmakta olan PHP dosyası için oluşturduğu çevre değişkenlerinin yazılı olduğu dizi değişken. Bu değişkenin içinde şu unsurlar bulunur:
HOSTNAME: Sunucunun IP adresi
SHELL: Unix sisteminde kullanılan Shell programı
HOSTTYPE: Sunucunun adı ve sürünü
OSTYPE: Sunucu'nun işletim sistemi
HOME: Çalışan programın kök dizini
PATH: Çalışan programın Sunucu'daki yolu

HTTP_SERVER_VARS Sunucu programın çalışmakta olan PHP dosyasına sunduğu bazı bilgilerin bulunduğu dizi değişken. Bu değişkenin içinde şu unsurlar bulunur:
PHP_SELF: Çalışan PHP programının bulunduğu dizin ve adı
PATH_TRANSLATED: Çalışan PHP programının fiziksel yolu

HTTP_GET_VARS Bir Form'dan GET metoduyla alınan bilgilerin anahtar=değer çiftleri

olarak kaydedildiği dizi değişken
HTTP_POST_VARS Bir Form'dan POST metoduyla alınan bilgilerin anahtar=değer çiftleri olarak kaydedildiği dizi değişken
HTTP_USER_AGENT Ziyaretçinin bilgisayarında kurulu Internet Browser programı
QUERY_STRING Form ile bilgi alırken GET metodunu kullandığımız takdirde, Browser'ın göndereceği bilgilerin tutulduğu değişken
REMOTE_ADDR Ziyaretçinin bilgisayarına ISS tarafından atanmış IP adresi
REQUEST_METHOD Form ile gelen bilgilerin gönderildiği metod: GET veya POST
REQUEST_URI O anda çalışmakta olan PHP dosyasının adı ve varsa bu ada eklenmiş Query_String
SCRIPT_FILENAME O anda çalışmakta olan PHP programının dosya adı
SCRIPT_URI O anda çalışmakta olan PHP programının tam URL adresi
SERVER_ADDR Sunucunun IP adresi
SERVER_PROTOCOL Sunucunun HTTP protokolünün sürümü

Form'dan GET Metoduyla Gelen Bilgiler

Ziyaretçilerimizin ne tür Browser kullandıklarını HTTP_USER_AGENT değişkeninin değerini alarak ve bu değerin içinde belirli anahtar kelimeleri aratarak bulabiliriz. Form ile gelen bilgiler, GET metodu ile alınıyorsa, hem QUERY_STRING, hem de HTTP_GET_VARS dizisine kaydolur. POST metoduyla aldığımız bilgileri HTTP_POST_VARS değişkenin değerleri arasında buluruz. Bunları öğrendiğimize göre, şimdi gerçekten bir HTML Form'u yapabil ve bununla ziyaretçimizden bilgi alabiliriz.
Basit bir HTML Form'u tasarlayalım.

```
<HTML>
<HEAD>
<TITLE>PHP'de Formlar</TITLE>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
<BODY>
<FORM ACTION="formlar02_isle.php" METHOD="GET">
Adınız, Soyadınız: <INPUT TYPE="TEXT" NAME="adi">
<br>
Elektronik Adresiniz: <INPUT TYPE=TEXT" NAME="adres">
<br>
<INPUT TYPE="SUBMIT" VALUE="Gönder Gitsin!"> <INPUT TYPE="RESET"
VALUE="Vazgeç, Gönderme!">
</FORM>
</BODY>
</HTML>
```

Bu Form'la Web tasarım yarışmasına katılmayacağımıza göre, şimdilik sadece Web Sunucuya bilgi göndermekte kullanabiliriz. Form'un ACTION parametresine dikkat ederseniz, formlar02_isle.php adlı bir dosyanın adını göreceksiniz. Bu, ziyaretçinin Gönder düğmesini tıklamasıyla birlikte Form'un içerdiği bilgilerin METHOD parametresinde pazılı olan GET yöntemiyle Sunucu'da gönderileceği programın adıdır. Bu sayfa, Browser'da şöyle bir görüntü verecektir:

Şimdi bir an için ne olacağını düşünmeden, formu doldurun ve Gönder düğmesini tıklayın; ve Browser'ınızdaki hata mesajına aldırmadan, URL adres kutusunda ne yazdığını okuyun:

http://server/formlar02_isle.php?adi=Faruk+Ta%E7&adres=Faruktac@mynet.com

Bu, HTTP protokolüne göre GET yoluyla bilgi göndermekte kullanılan yöntemin tam bir örneğidir: Browser, GET yoluyla bilgi göndereceği zaman, Form'daki bütün bilgileri URL-Encoding denen sistemle kodlar; Form'un alan adlarına o alanlara ziyaretçinin yazdığı bilgileri bir eşittir işaretiyle ekler; bu tür alan=girdi çiftlerinin arasına & (ve işareti) koyar ve gönderir. Web sunucu, bu bilgileri alınca, önce kendi oluşturduğu bazı değişkenlere (hem QUERY_STRING, hem de HTTP_GET_VARS dizisine) yazar ve sonra URL hanesinde adı yazılı olan programa (sayfaya) verir. Şimdi bizim bu bilgilerin gönderildiği PHP programını kendisine verilecek bu bilgileri işlemeye hazır şekilde yazmamız gerekir.

```
<?php
print ("Sayın <b>$adi</b>\n\n");
print ("<p>Elektronik adresiniz: <b>$adres </b></p>\n\n");
?>
```

Şimdi, Browser'ınızda formlar02.htm sayfasını yeniden açın, Form'u doldurun ve gönderin. Açılacak sayfa, Form'un göndereceğii bilgileri alacak ve kendi görünteleyecektir.

Fakat burada gördüğünüz gibi Sunucu'nun ziyaretçiden gelen bilgileri depoladığı dizileri kullanmadık. Bunu yaparken, GET ile gelen bilgiler kendisinde verildiğinde PHP programının alan adlarını değişken adı, bunların karşısında yazılı olan verileri de bu değişkenin değeri saymasından yararlandık. Fakat isteseydik, bu değişkenleri, Sunucu'nun oluşturduğu dizilerden de alabilirdik. Şimdi hem Form'umuzu geliştirelim; hem de bu kez okuma işini Sunucu dizisinden yapalım.

```
<FORM ACTION="formlar03_isle.php" METHOD="GET">
Adınız, Soyadınız: <INPUT TYPE="TEXT" NAME="adi">
<BR>
Elektronik Adresiniz: <INPUT TYPE=TEXT" NAME="adres">
<BR>
Hangi notunuzu öğrenmek istiyorsunuz?
<BR>
<SELECT NAME="hangi_not">
<OPTION>--Lütfen seçiniz--
<OPTION>Sınav 1
<OPTION>Sınav 2
<OPTION>Ortalama
</SELECT>
<BR>
<INPUT TYPE="SUBMIT" VALUE="Gönder Gitsin!"> <INPUT TYPE="RESET"
VALUE="Vazgeç, Gönderme!">
</FORM>
```

Yeni Form'da yeni bir HTML unsuruna yer verdiğimizi ve SELECT..OPTION etiketi ile ziyaretçiye bir seçim imkanı verdiğimizi görüyorsunuz. Şimdi, bu Form'un göndereceği bilgileri işleyecek PHP

programını yazalım. Aşağıdaki kodları formlar03_isle.php adıyla kaydedelim:

```
<?php
        foreach ($HTTP_GET_VARS as $anahtar=>$deger ) {
                print ("<b>$anahtar = $deger <br>\n");
        }
?>
```

Demiştik ki, Web sunucu, bir istemci Browser'dan kendisine GET yöntemiyle yollanan Form bilgilerini $HTTP_GET_VARS adlı dizi değişkende tutar. Yine daha önce görmüştük ki dizi değişkenlerin içinde ya sayı ya da isim olarak bir anahtar ve bu anahtarın temsil ettiği bir değer vardır. Burada, $HTTP_GET_VARS değişkeninin anahtar ve değerlerini $anahtar ve $deger değişkenlerine => operatörünün yardımıyla, bir foreach döngüsü içinde atıyoruz. Döngü kullanmamızın sebebi, dizi değişkenin içinde birden fazla anahtar=değer çifti bulunması ihtimali bulunması; döngü olarak da foreach kullanmamızın sebebi dizide kaç adet anahtar=değer çifti bulunduğunu bilmememizdir.

Form sayfasının gönderdiği bilgilerin nasıl derlenip toplanıp URL-koduyla Sunucuya gönderildiğini, Form'un Gönder düğmesini tıkladığımızda Browser'ın URL adres hanesinde ilen bilgilerin görülecektir. Buradaki örnekte bu bilgi (HTTP bölümünü ve URL kodlarını kaldırarak):

```
adi=Şükran+Tabak&adres=stabak@somenet.com&hangi_not=Sınav+1
```

şeklindedir. Bu bilgi, sunucu tarafından $HTTP_GET_VARS dizi değişkeninin içine yazıldığına göre, daha önce gördüğümüz gibi dizi değişkenlerin anahtarlarını ve bu anahtarların temsil ettiği değerleri bir döngü içinde $anahtar ve $deger değişkenlerine atarsak, daha sonra bu değişkenlerin değerlerini Browser penceresine göndermemiz mümkün olur.

Şimdi biraz dizi-değişken içine dizi-değişken koyalım! Yani ziyaretçinin göndereceği bilgiler, buradaki gibi SELECT..OPTION etiketinde yapacağı sadece bir unsur seçimi olmasın da çoklu-seçim olsun. HTML bilgilerinizi yoklarsanız, bunu SELECT etiketini MULTIPLE parametresi ile yapabildiğimizi hatırlayacaksınız.

```
<FORM ACTION="formlar03a_isle.php" METHOD="GET">
Adınız, Soyadınız: <INPUT TYPE="TEXT" NAME="adi">
<BR>
Elektronik Adresiniz: <INPUT TYPE=TEXT" NAME="adres">
<BR>
Hangi notunuzu öğrenmek istiyorsunuz?
<BR>
<SELECT NAME="hangi_not[]" MULTIPLE>
<OPTION>Sınav 1
<OPTION>Sınav 2
<OPTION>Ortalama
</SELECT>
<BR>
<INPUT TYPE="SUBMIT" VALUE="Gönder Gitsin!"> <INPUT TYPE="RESET"
VALUE="Vazgeç, Gönderme!">
</FORM>
```

Burada, HTML'in SELECT.. OPTION etiketlerini kullanarak, ziyaretçimizden hangi sınav notunu

öğrenmek istediğini bize bildirmesini istiyoruz. Dikkat ettiğiniz gibi, bu kez Form, elde edeceği verileri formlar03a_isle.php programına yollamak istiyor. Form'daki <SELECT NAME="hangi_not[]" MULTIPLE> satırına da dikkat ettiniz mi? Bu satırın özelliği, daha önceki SELECT..OPTION etiketinden farklı olarak ziyaretçinin çoklu seçme yapmasına imkan veriyor; ve elde edilecek değeri "hangi_not[]" alanının değeri olarak bildiriyor. HTTP iletişim ilkelerine göre çoklu-seçim halinde seçilen OPTION değerleri Sunucu'ya aynı alan adının karşısına yazılarak gönderilir. Formumuzun göndereceği bilgi yumağını satırlar haline getirirsek (HTTP bölümünü atar ve URL kodlarını çözersek) bunu görebiliriz:

```
adi=Şükran Tabak
adres=stabak@somenet.com
hangi_not[]=Sınav 1
hangi_not[]=Sınav 2
hangi_not[]=Ortalama
```

Kendisine böyle bir bilgi yumağı gelen Server, bunun tümünü $HTTP_GET_VARS dizi değişkeninin içine yazacaktır. Başka bir deyişle, bu dizi değişken çıok-boyutlu çok-elemanlı ilişkili-dizi olduğu için, içinde rahatça aynı isimde değişkenlere farklı endeks sayısı verecektir. Fakat sorun PHP'nin, bu dizinin içinden değişkenleri almasında ortaya çıkacak ve endeks ismi aynı olan değişkenler sorun olacaktır. Bunu değişkenin endeks adı olarak kullanılacak kelimenin yanına köşeli parantez koyarak çözüyoruz. PHP bu adı görünce, bunun çok-elemanlı bir dizi değişken olacağını anlayacaktır.

Eğer bu formu, formlar03_isle.php programına gönderseniz (bunu nasıl yapabilirsiniz?), "hangi_not" değişkeninin değeri olarak Browser penceresinde "array" kelimesinin belirdiğini görebilirsiniz. Çünkü PHP açısından bu değişken bir dizidir ve içinde anahtar=değer çiftleri vardır. Daha önce anahtar=değer çiftlerini geçici değişkenlere atayıp bir döngü ile yazdırmıştık. Şimdi, PHP kodumuzu bu duruma uygun hale getirelim. Biraz önce yazdığımız Form işleme programının sadece PHP bölümünü şöyle değiştirerek, formlar03a_isle.php adıyla kaydedelim:

```
<?php
        foreach ($HTTP_GET_VARS as $anahtar=>$deger ) {
                if ( gettype ($deger ) == "array" ) {
                        print ("$anahtar == <br>\n");
                        foreach ( $deger as $yeni_degerler )
                                print (".. $yeni_degerler<br>");
                                }
                else {
                print ("<b>$anahtar = $deger <br>\n");
                }
        }
?>
```

PHP'nin gettype() fonksiyonunu daha önce görmüş ve bir değişkenin türünü anlamaya yaradığını öğrenmiştik. Burada $HTTP_GET_VARS değişkeninden aldığımız değerlerden herhangi birinin gerçekten bir değişken değeri mi, yoksa bir dizi (array) mi olduğunu gettype() ile anlayabiliriz. Eğer değer olarak karşımıza "array" kelimesi çıkarsa, bunu kendi içinde anahtar ve değer olarak bölebilir ve herbirini ayrı ayrı görüntüleyebiliriz. Eğer $HTTP_GET_VARS değişkeninden aldığımız değer, dizi değil de gerçekten bir değişken ise (else) doğruca bu değeri ve anahtarını yazdıracaktır. Sonuç ise dizi-değişken içindeki dizi-değişkenin değerlerinin tek tek görüntülenmesi olacaktır.

Form'dan POST Metoduyla Gelen Bilgiler

HTML Form etiketinin METHOD parametresinin değeri GET olabildiği gibi POST da olabilir; ve HTTP sunucusu bu yöntemle gelen bilgileri $HTTP_POST_VARS dizi-değişkeninde tutar. Yukarıdaki çok-seçmeli Form'un FORM etiketini şöyle değiştirerek, formlar03b.htm adıyla kaydedelim:

```
<FORM ACTION="formlar03a_isle.php" METHOD="POST">
```

Aynı şekilde son Form işleme programımızda da sadece şu değişikliği yapalım:

```
foreach ($HTTP_POST_VARS as $anahtar=>$deger ) {
```

Bu dosyayı da formlar03b_isle.php adıyla kaydedelim. HTML sayfasını açarak formu doldurur ve gönderirseniz, sonucun metod olarak GET kullanan Form'dan hiç farklı olmadığını göreceksiniz. Çünkü PHP programı bu Form'un gönderdiği bilgilerin $HTTP_POST_VARS değişkenine yazıldığını biliyordu. $HTTP_POST_VARS da PHP açısından içinde anahtar=değer çiftleri olan bir dizi-değişkendir; bu değişkenin değerlerine de tıpkı daha önce olduğu gibi erişiriz.

HTTP açısından GET ile POST'un tek farkı gelen değerlerin nerede nasıl tutuldueğundan ibaret değildir. GET yönteminde, bir Browser'ın sunucuya gönderebileceği verinin uzunluğu, Sunucunun ayarlarına bağlı olmak üzere, sınırlıdır. Oysa POST ile alacağımız veri miktarı, sadece sunucunun bulunduğu bilgisayarın sabit disk alanıyla sınırlıdır. (Tabiî bu günümüzde sınırsızdır, anlamına geliyor!) Bir başka fark, Browser'ın GET yoluyla gönderdiği verilerin (ve bu arada ziyaretçinin parola olarak yazdıklarında ekrana yıldız olarak çıkan metinler dahil) tümü, sunucuya, URL-kodlanmış metin olarak, Browser'ın URL adres hanesine de yazılmasıdır. Bir çok kullanıcı için bu bir güvensizlik belirtisi sayılır. Bu iki unsur Formlarımızda metod olarak GET yerine POST kullanmanın daha yerinde olduğunu gösterir.
Tedbirli Web programcılığı, özellikle birden fazla tasarımcı ve programcının birlikte çalıştığı ve Formlarda hangi yöntemin tercih edildiğini bilmenin kolay olmadığı projelerde, Form bilgisi işleyen PHP programlarımızda Form'da hangi metod kullanılmış olursa olsun, işlyeyici programın iki duruma da elverişli olmasını sağlamaktır. Sözgelimi son yazdığımız Form işleme programımızı şöyle değiştirirsek, ve Form içeren HTML sayfasını bu programı veri gönderecek şekilde değiştirirsek (nasıl?), her iki metodla gönderilen verileri işleme yeteneğine sahip bir program elde etmiş oluruz.

```
<?php
$form_bilgisi = ( isset($HTTP_POST_VARS ) )
        ? $HTTP_POST_VARS : $HTTP_GET_VARS;
                foreach ($form_bilgisi as $anahtar=>$deger ) {
                if ( gettype ($deger ) == "array" ) {
                        print ("$anahtar == <br>\n");
                        foreach ( $deger as $yeni_degerler )
                                print (".. $yeni_degerler<br>");
                                }
                else {
                print ("<b>$anahtar = $deger <br>\n");
                }
        }
?>
```

Form ile işlemciyi Birleştirelim

Şu ana kadar yaptığımız bütün Form örneklerinde, Form'un bulunduğu HTML sayfası ile bu Form'un göndereceği verileri işleyen PHP programı iki ayrı belge halinde idi. Bu, buradaki örneklerde olduğu gibi, ziyaretçinin verdiği bilgileri sadece Brıowser penceresine yazdıran bir eğitim çalışması için belki uygun; ama gerçek Web sitelerimizde ziyaretçilerimizin vereceği bilgileri çoğu zaman sadece onların Browser pencerelerinde göstermekle kalmayız, fakat bu bilgileri ya elektronik posta yoluyla kendimize yollarız, ya da sunucuda bir düzyazı veya veritabası dosyasına işleriz. Bu ve diğer amaçlarla yapacağımız Form içeren HTML sayfaları, aslında PHP programımızın bir içinde yer alabilir; ya da başka bir deyişle, Form'umuz ziyaretçinin vereceği bilgileri kendi bulunduğu PHP programına gönderebilir!

Bu karmaşık ifadeyi bir örnekle açalım. Yukarıda yaptığımız son HTML sayfası ile ve PHP programını şöyle birleştirelim;

```
if ( isset ( $HTTP_POST_VARS )) {
print ("<HTML>\n");
print ("<HEAD>\n");
print ("<TITLE>PHP'de Formlar</TITLE>\n");
print ("<meta http-equiv=\"content-type\" content=\"text/html; charset=ISO-8859-9\">\n");
print ("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=windows-
1254\">\n");
print ("</HEAD>\n");
print ("<BODY>\n");
        foreach ($HTTP_POST_VARS as $anahtar=>$deger ) {
                if ( gettype ($deger ) == "array" ) {
                        print ("$anahtar == <br>\n");
                        foreach ( $deger as $yeni_degerler )
                                print (".. $yeni_degerler<br>");
                                }
                else {
                print ("<b>$anahtar = $deger <br>\n");
                }
        }
print ("</BODY>\n");
print ("</HTML>\n");
}
else {
print ("<HTML>\n");
print ("<HEAD>\n");
print ("<TITLE>PHP'de Formlar</TITLE>\n");
print ("<meta http-equiv=\"content-type\" content=\"text/html; charset=ISO-8859-9\">\n");
print ("<meta http-equiv=\"Content-Type\" content=\"text/html; charset=windows-
1254\">\n");
print ("</HEAD>\n");
print ("<BODY>\n");
print ("<FORM ACTION=\"$PHP_SELF\" METHOD=\"POST\">\n");
print (" Adınız, Soyadınız: <INPUT TYPE=\"TEXT\" NAME=\"adi\">\n");
print ("<BR>\n");
print (" Elektronik Adresiniz: <INPUT TYPE=\"TEXT\" NAME=\"adres\">\n");
print ("<BR>\n");
print (" Hangi notunuzu öğrenmek istiyorsunuz? \n");
print ("<BR>\n");
```

```
print ("<SELECT NAME=\"hangi_not[]\" MULTIPLE>\n");
print ("<OPTION>Sınav 1 \n");
print ("<OPTION>Sınav 2 \n");
print ("<OPTION>Ortalama \n");
print ("</SELECT>\n");
print ("<BR>\n");
print ("<INPUT TYPE=\"SUBMIT\" VALUE=\"Gönder Gitsin!\">\n");
print ("<INPUT TYPE=\"RESET\" VALUE=\"Vazgeç, Gönderme!\">\n");
print ("</FORM>\n");
print ("</BODY>\n");
print ("</HTML>\n");
}
```

Bu dosyanın tümüyle PHP programı olduğuna dikkat ettiniz, tabiî? Program açıldığında sunucunun $HTTP_POST_VARS dizi-değişkeninin bir değer içerip içermediğini bir if deyiminin içinden bir değişkenin içeriği olup olmadığını anlamamıza yarayan isset () fonksiyonu ile yapıyoruz. Bu şart doğru ise, yani $HTTP_POST_VARS dizi-değişkeni bir değer içeriyorsa, program, foreach döngüsünün içinde bu değişkenin içindekileri almaya ve Broüwser penceresinde görüntülemeye başlıyor. Bu şart doğru değilse, yani $HTTP_POST_VARS dizi-değişkeni henüz bir değer içermiyorsa, if deyiminin birinci bölümünü içindeki hiç bir kod icra edilmiyor ve prgram else deyimine sıçrıyor. Programın else bölümü ise daha önceki HTML kodlarımızı içeren bir dizi print() fonksiyonu yerine getiriyor; yani Browser'a içinde Form bulunan HTML sayfasını yazdırıyor. Burada FORM etiketine dikkat edelim:

```
print ("<FORM ACTION=\"$PHP_SELF\"METHOD=\"POST\">");
```

Form'un ACTION parametresinde bir PHP programının adı yerine "$PHP_SELF" değişken adını görüyoruz. Bu, bu bölümün başında ele aldığımız gibi, sunucunun bu PHP programına sağladığı çevre değişkenlerinden biridir ve o anda çalışmakta olan PHP programının dosya adını içerir. (Bizim örneğimizde bu değişkenin değeri nedir?)

PHP ile HTTP Başlıklarının Kullanımı

Bu kitapçık HTTP başlık fonksiyonlarını anlatmak amacında olmayıp bu geniş konu hakkında sadece pratikte bilgi aktarmakla yetinecektir.
Bir web tarayıcı bir web sunucuya bir döküman istediği gönderdiğinde web sunucu istenilen dökümanın yanısıra header(başlık) adı verilen bazı açıklayıcı ve yönlendirici bilgiler de gönderir. Konuyu örneklerle açıklamaya çalışalım.

```
header("Pragma: no-cache");
header("Cache-Control: no-cache, must revalidate");
```

PHP, dinamik içerikli web siteleri oluşturmak amaçlı kullanıldığı için kimi zaman bazı sayfaların web tarayıcının cache denilen ara belleğinde saklanmaması daha doğru olmaktadır. Bu, özellikle yüksek güvenlik gerektiren ve çok sık değişen siteler için gereklidir. Yukarıdaki kod parçası kullanıldığı PHP betiğinin çıktısının cache denilen ara bellekte saklanmasını engellemek amacı ile kullanılır.

```
header("Location: http://www.php.org.tr");
```

Bu kod parçası ise dökümanın herhangi bir yerinde, kullanıcıdan hiç bir girdi almaya ihitiyaç duymadan aktif adresi değiştirmeye yarar. Çok sık kullanılır ve özellikle bir login ekranından sonra

girilen bilgiye göre karar vererek aktif adresi değiştirmek gibi uygulamalar için faydalıdır.

Örnek vermek gerekirse... login.html adlı bir HTML dökümanımız olduğunu varsayalım. Bu HTML dökümandan alınan girdi "karar.php" adlı PHP betiğinde işlenecek ve girilen
bilgiye göre adresi değiştirecektir.

```
<html>
<head>
<title> Login Form </title>
</head>
<body>
<form action="karar.php" method="post">
<input type="text" name="kullanici_adi"><p>
<input type="password" name="sifre"><p>
<input type="submit" name="submit" value="Tamam">
</form>
</body>
</html>
```

Bu sayfadan alınan girdiler kullanici_adi ve sifre seklindedir. Bu bilgileri karar.php dosyasında inceleyerek karar veriyoruz.

```
// başındaki ve sonundaki boşlukları yok ediyoruz.
$kullanici_adi = trim($kullanici_adi);
$sifre = trim(sifre);
if ($kullanici_adi == "umut") {
if ($sifre = "bu_bir_sifredir") {
header ("Location : giris_basarili.php");
}
}
header ("Location : giris_basarisiz.php");
```

Yukarıdaki kod parçasında kullanıcı adı ve şifrenin doğru olup olmadığı kontrol edildikten sonra eğer her iki bilgi de doğru ise giris_basarili.php adlı döküman çağrılıyor. Eğer bu bilgilerden herhangi birisi doğru değilse giris_basarili.php betiği çağrılmayacak ve dolayısı ile aktif adres, koşulsuz çağrılan giris_basarisiz.php dosyası olacaktır.

Yukarıdaki örnekte dikkat edilmesi gereken bir nokta vardır. header ("Location....."); kullanımı sadece kendisinden önce ekrana bir çıktı yapılmadığı durumlarda çalışacaktır.

Örneğin :

```
echo "Öylesine bir çıktı";
header(Location : "deneme.html");
```

Kod parçacığı beklendiği gibi deneme.html dosyasını çağırmayacaktır. Bu kısıtlama PHP' nin zayıflığından değil HTTP protokolünün tanımlarından kaynaklanmaktadır. Diğer bazı çok kullanılan HTTP başlıkları şu şekildedir:

Content-Encoding
Content-Language
Content-Type

Expires
Referrer
Last-Modified
User-Agent
Accept-Encoding
Accept-Language
.....

Bu başlıklar hakkında ayrıntılı bilgiye ftp://ftp.isi.edu/in-notes/rfc2626.txt adresinden ulaşabilirsiniz.

PHP ile Posta Gönderme

Elektronik posta, kuşkusuz Internet üzerindeki bir numaralı iletişim aracıdır. PHP ile posta gönderme işlemi, tek bir fonksiyon kullanımı ile halledilebilecek kadar kolaylaştırılmıştır. Yalnız uyaralım, aşağıda gördüğünüz örnek kodların UNIX / Windows 2000 ve Windows 95 / 98 sistemlerde çalışma şekilleri farklıdır. UNIX türevlerinde ve Windows 2000'nde, e-posta işlemlerinin yapılması için gerekli olan POP3 ve SMTP posta sunucuları önyüklüdürler. Windows 95 ve 98 sistemlerde ise, bu sunucular olmadığı için, bu sunucuları yüklemeden örnek kodları çalıştırmanız mümkün değildir

Mail Fonksiyonu

Mail Fonksiyonu'nun kullanım şekli aşağıdaki gibidir:

Mail (kime, konu, mesaj, [ek_başlıklar]);

Bu fonksiyon ile e-posta otomatik olarak "kime" kısmındaki kişiye veya kişilere gönderilir. Her bir virgül (,) ayrı bir kısmı göstermektedir. Örneğin:

```
mail("deneme@php.org.tr", "Deneme", "Merhaba\nBu bir denemedir\nHoşçakalın.");
```

Ek başlıkları da yazarsak örneğimiz şöyle olacaktır:

```
Mail ("deneme@php.org.tr", "Deneme", "Merhaba\nBu bir denemedir\nHoşçakalın.", "From: webmaster@php.org.tr\nReply-To: webmaster@php.org.tr");
```

Ek Başlıkların tümünün daima çift tırnak içinde olduğuna ve birbirlerinden "\n" ile ayrıldıklarına dikkat edin. Bu yazım şekli mesajı oluştururken de geçerlidir. Mesajı oluştururken bir alt satıra geçmek istediğinizde her zaman "\n" karakterini kullanmalısınız.

Posta yollarken en sık kullanılan başlıkları listeleyelim:
From (Kimden)
Reply-to (Cevabın yollanacağı adres)
Cc (Carbon Copy - Karbon kopya)
Bc (Blind Copy - Kör Kopya - gönderdiğiniz e-posta adreslerini gizler)

Tüm bunları arka arkaya kullanabilirsiniz.

E-posta için temel olarak iki şey gereklidir: Bunlardan ilki, bir formdan gerekli *Anahtar*= Değer

(Key= Value) çiftlerini almak ve php dosyasına iletmektir. Bildiğiniz gibi Internet üzerinde tüm girdiler bu şekilde gönderilmektedir. Bunu bildiğinizi varsayarak aşağıdaki örnekleri inceleyelim.

Önce e-posta gönderilecek formumuzu hazırlayalım ve dosyamızı eposta.php olarak saklayalım:

```
<form name="eposta" action="form.php" method="post">
<table border="0" cellspacing="2" cellpadding="2" align="left">
<tr>
<td colspan="2">
<div align="center">
<p align="left">Sitemizi inşa etmede ve yenilemede görüşleriniz
çok önemlidir. Lütfen doldurmaktan çekinmeyin.</p>
</div>
</td>
</tr>
<tr>
<td width="98">
<div align="right">Ad Soyad: </div>
</td>
<td width="450">
<input type="text" name="adsoyad"><!-- 1. Değişken "adsoyad" -->
</td>
</tr>
<tr>
<td width="98">
<div align="right">E-Posta: </div>
</td>
<td width="450">
<input type="text" name="eposta"><!-- 2. Değişken "eposta" -->
</td>
</tr>
<tr align="left" valign="top">
<td width="98">
<div align="right">Yorumlarınız:</div>
</td>
<td width="450">
<!-- 3. Degisken "yorum" -->
<textarea name="yorum" cols="55" rows="10"></textarea>
<!-- 4. Degisken "kime" -->
<br><input type="hidden" name="kime" value="deneme@php.org.tr">
<!-- 5. Degisken "konu" -->
<input type="hidden" name="konu" value="Web Sitemden">
                <!-- 6. Degisken "gonder" ancak bunlar sabittir. php uzantili dosyamizda
degisken atamamiza gerek yok. -->
<input type="submit" name="gonder" value="Gönder">
<input type="reset" name="Submit" value="Sil">
</p>
<p style="font-size: 10px;">Lütfen 20 sn. bekleyin. Tekrar Gönder butonuna basmanıza
gerek yoktur.</p>
</td>
</tr>
```

```
</table>
</form>
```

Her zaman için öncelikle formunuzdaki Anahtar = Değer çiftlerini aklınızda tutarsanız, PHP'de değişkenleri atamanız çok kolaylaşacaktır.

<input type="hidden" name="kime" value="deneme@php.org.tr">
Dikkat ederseniz "kime" anahtarına "deneme@php.org.tr" değerini atadım.

<input type="hidden" name="konu" value="Web Sitemden">
"konu" anahtarına da "Web Sitemden" değerini atadım.

Değerleri bu şekilde atamamızın amacı, ileride sadece bu iki alanı değiştirerek farklı formlar yaratabilmenizi sağlamak. Böylece hem daha anlaşılır bir sisteminiz olacak, hem de yeni bir form yaratmanız daha az zamanınızı alacak.

Yukardaki formda yer alan değişkenleri sıralayalım:

1.adsoyad (ziyaretçi dolduracak)
2.eposta (ziyaretçi dolduracak)
3.yorum (ziyaretçi dolduracak)
4.kime (deneme@php.org.tr)
5.konu (Web sitemden)

Şimdi bu verileri kullanarak elektronik postamızı hazırlayıp yollayacak PHP kodumuzu yazalım:

```
<?php
        $mesaj = "Ad Soyad: " . $adsoyad . "\n";
        $mesaj .= "E-Posta: " . $eposta . "\n";
        $mesaj .= "Yorum: " . $yorum . "\n";
        $extra_baslik = "From: $kime\n";
        $extra_baslik .= "Reply-To: $eposta\n";
        $extra_baslik .= "Bcc:arsiv@php.org.tr\n";
        $extra_baslik .= "Content-Type:text/plain; charset=\"iso-8859-9\"\n";
        $extra_baslik .= "Content-Transfer-Encoding: 8bit\n";
        mail($kime, $konu, $mesaj, $extra_baslik);
?>
```

Önce size yabancı gelebilecek birkaç kullanım şeklini açıklayalım.

Eğer Perl veya C tabanlı bir kullanıcı iseniz, ".=" kullanımına zaten aşinasınız demektir. Basic benzeri dillerden gelen kullanıcılar için de onlara tanıdık gelecek kullanım örneğini verelim:

$mesaj = $mesaj . "E-Posta: " . $eposta . "\n";

PHP'de, Basic benzeri dillerden farklı olarak, değişkenleri birbirlerine eklemek için nokta kullanılır. PHP ile her iki kullanım şekli de doğrudur. Biz bütün örneklerimizde ".=" biçimini kullanacağız.

HTML sayfalarını yayınlarken geçerli olan bütün kurallar, e-posta hazırlarken de geçerlidir: İçeriğin hangi karakter seti ile okunması gerektiğini mutlaka belirtmelisiniz, aksi takdirde e-postanız farklı e-posta istemcilerinde farklı sonuçlar verecektir! Bu sorunu çözmek için, $extra_baslik değişkenine iki yeni satır ekledik: Content-Type ve Content-Transfer-Encoding.

Dikkat ettiyseniz, postanın BCC bölümüne de bir e-posta adresi yazdık. Böylece yollanan bütün formların bir kopyasının da "arsiv@php.org.tr" adresine gitmesini sağlamış olduk.

Son olarak unutmamamız gereken bir şey daha var. Formu gönderdikten sonra, ziyaretçimize e-postanın gönderildiğine ilişkin bir mesaj vermeliyiz. Bunun için yukarıdaki kodumuza dokunmadan onu HTML kodlarıyla sarıp, gonder.php ismiyle kaydedeceğiz. Gonder.php dosyasının son hali aşağıdaki gibi olacaktır:

```
<?php
$mesaj = "Ad Soyad: " . $adsoyad . "\n";
$mesaj .= "E-Posta: " . $eposta . "\n";
$mesaj .= "Yorum: " . $yorum . "\n";
$extra_baslik = "From: $kime\n";
$extra_baslik .= "Reply-To: $eposta\n";
$extra_baslik .= "Bcc:arsiv@php.org.tr\n";
$extra_baslik .= "Content-Type:text/plain; charset=\"iso-8859-9\"\n";
$extra_baslik .= "Content-Transfer-Encoding: 8bit\n";
mail($kime, $konu, $mesaj,$extra_baslik);
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-9">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
<title>Sayın <?php echo($adsoyad) ?>, formunuz başarıyla alındı.</title>
</head>
<body>
<b>Sayın <font color="Purple"><?php echo($adsoyad); ?></font>,
formunuz <b><font color="Purple">
<?php echo($kime); ?></font></b> adresine gönderilmiştir.
Teşekkür ederiz.</b>
</body>
</html>
```

Başta da belirttiğimiz gibi, bu kodları denemek için iki şansınız var, Linux veya Windows 2000 kullanıcısı iseniz, daha şanslısınız, sisteminizde zaten yüklü bir SMTP sunucu olma ihtimali çok yüksek. Eğer Windows 95 / 98 kullanıcısı iseniz, www.php.org.tr adresinden, Windows 95 / 98 altında kullanabileceğiniz SMTP sunucusunun adresini ve kurulum bilgilerini gerekli bütün detaylarıyla birlikte bulabilirsiniz.

İkinci bir şansınız daha olduğunu söyledik, o da Internet'e bağlı olmak koşuluyla kullanabileceğiniz kendi ISP'nizin SMTP sunucusu, ya da Yahoo! gibi ücretsiz posta hizmeti veren ve SMTP adresi bilinen bir site. Her iki olanak için de yapmanız gereken değişikliklere bakalım. Windows altında çalışıyorsanız, C:\Windows\php.ini dosyasını açın ve [mail function] başlığı altındaki ayarlarınızı şu şekilde değiştirin:

Kullandığınız bilgisayarda kurulu bir SMTP sunucusu varsa:

SMTP = localhost ;for win32 only (Kendi serverınızın smtp ayarını yazın)
sendmail_from=deneme@php.org.tr ;for win32 only (Email adresinizi yazın)

Kullandığınız bilgisayarda kurulu bir SMTP sunucusu yoksa:
SMTP = smtp.mail.yahoo.com (Yahoo! dan bir e-posta adresiniz olduğunu varsayıyoruz)
sendmail_from=deneme@php.org.tr

Yahoo!'nun SMTP'sini kullanmak istiyorsanız, Yahoo! sitesindeki hesabınıza girin ve Options (Seçenekler) kısmından POP3 ile e-posta alacağınızı belirtin. Yukardaki satırları php.ini dosyanıza ekledikten sonra, Yahoo! nun SMTP sunucusunu kullanarak e-posta yollayabilirsiniz.

Formu istediğiniz gibi çoğaltabilirsiniz. Gerisi sizin yaratıcılığınıza kalmıştır.

COOKİE YOLLAMA

setcookie(“tanitici”,”1”,time()+600); => (tanitici adında değeri 1 olan cookie atar)

var olan cookieyi tespit etme

```
$cook=$HTTP_COOKIE_VARS[“tanitici”];
if ($cook!=””):
echo “siteye ilk giriş”;
end if:
```

var olan cookieyi silmek

Aynı isimde süresi verilemeyen cookie atılarak var olan cookie silinir.

```
setcookie(“tanitici”,”1”,”time());
```

OTURUM (SESSION)

```
session_name(“serdar”);
$degisken=”serdar”;
session_register(“degisken”);
```

Bir değişkenin register edildiğini sınamak

```
if(session_is_registered(“degisken”)):
echo “register edilmiş”;
endif;
```

Register edilmiş değişkeni öldürmek

```
Session_unregister(“degisken”);
```

Açılmış Oturumu Kapatmak

```
session_unset();
session_destroy();
```

Dosya "çıkartma"

Internet'ten hep dosya "indiririz!" Bir sunucuya, Web ziyaretçisi olarak gönderebildiğimiz tek şey ise, Formlara yazdığımız yazılardır! Oysa HTML'in INPUT INPUT etiketinin çok az kullanılan TYPE="file" parametresi ziyaretçiye Web sunucusuna dosya gönderme (upload) imkanı sağlar. HTTP protokolü buna imkan vermekle birlikte Browser'lar bu imkanı kullanmaya ileri sürümlerinde kavuştular. PHP4, ziyaretçilerimizin sitemize dosya göndermeleri halinde, bu dosyaların yönetimine ayrıca kolaylık sağlayan değişkenlere sahiptir.

```
<?php
$dosya_dizin = "/inetpub/wwwroot/";
$dosya_url = "http://server/";
if ( isset ( $dosya_gonder )) {
        print ("<b>Yol:</b> $dosya_gonder<br>\n");
        print ("<b>Adı:</b> $dosya_gonder_name<br>\n");
        print ("<b>Boyut:</b> $dosya_gonder_size<br>\n");
        print ("<b>Tür:</b> $dosya_gonder_type<br>\n");
        copy ( $dosya_gonder, "$dosya_dizin/$dosya_gonder_name" )or die ("Dosya
kopyalanamıyor!");
                if ( $dosya_gonder_type == "image/gif" ||$dosya_gonder_type ==
"image/pjpeg" ) {
                print ("<img src=\"$dosya_url/$dosya_gonder_name\"><p>\n\n");
                }
}
?>
<FORM ENCTYPE="multipart/form-data" ACTION="<?php print $PHP_SELF?>"
METHOD="POST">
<INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" VALUE="951200">
<INPUT TYPE="file" NAME="dosya_gonder"><BR>
<INPUT TYPE="SUBMIT" VALUE="Dosya Yolla!">
</FORM>
```

Bu programda <INPUT TYPE="file" NAME="dosya_gonder"> etiketinde kullandığımız NAME parametresine verdiğimiz değer, ziyaretçimizin göndereceği dosyanın sunucu tarafından kaydedileceği geçici dizinin tam yolunun yazılacağı değişkenin adı olacakdır. PHP, bu dosya ile ilgili her türlü bilgiyi bu adla kaydedektir. PHP, ziyaretçiden bir dosya başarıyla aktarıldığı anda otomatik olarak bu isimden yararlanarak şu değişkenleri oluşturur:

$dosya_gonder	Geçici kayıt dizini yolu (UNIX'te /tmp/phpXXX, Windows'da Windows/TEMP0phpXXX. Burada XXX yerine ziyaretçilerin gönderdiği dosyaların sıra numarasını göreceksiniz.)
$dosya_gonder_name	Ziyaretçinin gönderdiği dosyanın adı.
$dosya_gonder_size	Ziyaretçinin gönderdiği dosyanın boyutu.
$dosya_gonder_type	Ziyaretçinin gönderdiği dosyanın türü

PHP ayrıca bu bilgileri $HTTP_POST_FILES dizi-değişkeninde de tutar.

Yukardaki programda şu iki değişken çok önemlidir:

```
$dosya_dizin = "/inetpub/wwwroot/";
$dosya_url = "http://server/";
```

$dosya_dizin adıyla oluşturduğumuz değişkene vereceğimiz değer, ziyaretçinin göndereceği dosyanın kopyalanacağı klasörün adı olarak kullanlacaktır. Sözgelimi Windows ortamında buraya kişisel Web sunucunun varsayılan klasörünün adını yazabilirsiniz. Ziyaretçinin göndereceği dosya bir GIF biçiminde grafik dosyası ise bunu Browser'da görünteleyeceğimiz için, bu dizinin Web'e açık olması, başka bir deyişle bizim Web sunucumuzun erişebileceği bir dizin olması gerekir. Nitekim, $dosya_url değişkenine değer olarak bu klasörün URL adresini veriyoruz. Bu iki değişkeni gerçek Web sunucu için yazacağımız zaman, bizim sunucumuzun varnaydığı fiziksel klasör adını ve yolunu bulmamız gerekir. Bunu daha önce yazdığımız php.php veya formlar01.php programlarını sitemizde çalıştırarak bulabiliriz. (Nasıl?) Sözgelimi, http://www.mycgiserver.com/~ocal/ adresindeki sitenin fiziksel adresi ile bu adresin URL'ini dikkate alarak bu iki değişkeni yazmış olsaydık, şunu yazacaktık:

```
$dosya_dizin = "/wwwroot/mycgiserver.com/members/uNhM13/";
$dosya_url = "http://www.mycgiserver.com/~ocal/";
```

Bu uygulamayı kendi sunucunuzda yapmak isterseniz, mutlaka bu iki değişkeni doğru yazmanız gerekir. http://www.mycgiserver.com/~ocal/ dosya_gonder_server.php programı ile bir dosya gönderme (upload) işleminin sonucu şöyle:

Harici Dosya (include)

PHP programlarımızda bilmemiz gereken ilk dosya işlemi, bir PHP programına, kendi dışındaki dosyayı, tabir yerinde ise, okutmak ve içindekileri aynen alıp, görüntülemesini sağlamaktır. Bunu include komutu ile yaparız. Bu komut, kendisine adı verilen düzyazı dosyasının içeriğini aynen bu komutun bulunduğu noktaya "yazar." Bu yolla bir PHP programına sık kullandığınız bir metni veya program parçasını dahil edebilirsiniz. Bir güvenlik önlemi olarak bu tür dosyaların uzantılarını, sunucu ve Browser'ların tanıdığı MIME türlerine ait uzantılardan farklı yaparsanız, ziyaretçiler şans eseri de olsa bu dosyaları doğruca edinme imkanı bulamazlar. Bir örnek yapalım. Önce şu metni, harici_dosya01.x adıyla ve düzyazı biçiminde kaydedin (Windows ortamında Notepad'i kullanıyorsanız, dosya adı uzatması olarak .x harfinden sonra .txt harflerini eklediğine dikkat edin!):

"Ben harici bir dosyanın içindeki yazıyım.. Beni bir PHP programı alıp buraya getirdi! Kendisine teşekkür ederim"

Sonra, şu programı dosya_ekle01.php adıyla kaydedin, ve Browser'da açın:

```
<?php
```

```
include ("harici_dosya01.x");
print ("\n<p> Ben zaten bu programının içinde olan bir yazıyım. Baştan beri burada olduğuma çok memnununum.. Harici dosyaya hoşgeldin diyorum!</p>");
?>
```

include mu, require mı?

PHP4.0 ile, include komutu gibi işleyen ancak ondan farklı olarak kendisini çağıran programa değer veremeyen require komutu da kullanılabilir hale geldi. İçinde bir hesaplama bulunmayan veya kendisini çağıran dosyaya bir return komutu ile bir değer dönmesini sağlaması beklenmeyen dosyaları require komutu ile de ana programımıza dahil edebiliriz.

include ile alacağımız dosyaların adını PHP programının oluşturmasını sağlayarak kimi zaman pogramlarımıza dinamizm sağlamamız mümkündür. Sözgelimi bir menü maddesinin tıklanmasıyla harekete geçen bir Javascript fonksiyonunun, ziyaretçinin gideceği sayfaya göndereceği değeri include komutuna dosya adı oluşturmakta kullanabiliriz. Bazen include komutuna dosya adı oluşturmakta program içindeki döngülerden yararlanınız. Örnek:

```
for ( $i = 1 ; $i <= 3 ; ++$i) {
        include ("dosya0" . $i . ".x");
        }
```

Haricî dosyalarımızın adlarının dosya01.x, dosya02.x ve dosya03.x olmalı halinde, bu döngü sırasıyla her üç dosyayı da çağıracak ve altalta ana programa dahil edecektir.

Dosyalar hakkında bilgi

PHP'de yukarıda ele aldığımız include ve require komutları ve biraz sonra değineceğimiz dosya okutma ve yazdırma işlemleri dolayısıyla bir dosyanın varlığı veya yokluğu, ya da bir dosyaya ait sandığımız ismin bid klasöre ait olması, programımızın sağlıklı işleyebilmesi açısından büyük önem taşır. PHP bu amaçla bize bir kaç kullanıma hazır fonksiyon sağlıyor. Burada kısaca bu fonksiyonlara ve nasıl kullanıldıklarına değinelim:

Dosya var mı? file_exits()

Bir dosyanın var olup olmadığını denetleyen bu fonksiyon, dosya varsa true/doğru, yoksa false/yanlış sonucunu verir. Örnek:

```
if ( file_exits ( "bir_dosya.txt" ) )
print ("Dosya var!");
```

Dosya yoksa, program "Dosya var!" yazmadan yoluna devam edecektir.

Dosya mı, dizin mi? is_file() ve is_dir()

Kimi zaman klasörler de tıpkı dosyalar gibi adlandırılabilir. Bir dizinde gördüğümüz ismin gerçekten bir dosyaya ait olup olmadığını bu fonksiyonla sınarız. Sınama doğru, yani isim bir dosyaya ait ise fonksiyon true/doğru, değilse false/yanlış sonuç verir. Örnek:

```
if ( is_file ( "bir_dosya.txt" ) )
print ("Bu bir dosyadır!");
```

İsim bir dosyaya ait değilse program "Bu bir dosyadır!" yazmadan yoluna devam edecektir. Sınamayı ismin bir klasöre ait ait olup olmadığına bakrarak da yaparız. Bu durumda is_dir() fonksiyonunu kullanırız. isim bir dizine aitse fonksiyon true/doğru, değilse false/yanlış sonuç verir. Örnek:

```
if ( is_dir ( "/bir_isim" ) )
print ("Bu bir dizindir!");
```

İsim bir dizine ait değilse program "Bu bir dizindir!" yazmadan yoluna devam edecektir.

Dosya okunabilir mi? is_readable()

Programda kullanmaya karar vermeden önce bir dosyanın erişilebilir ve PHP tarafından okunabilir olup olmadığını sınayan bu fonksiyon, dosya okunabilir ise true/doğru, değilse false/yanlış sonuç verir. Örnek:

```
if ( is_readable ( "bir_dosya.txt" ) )
print ("Bu dosya okunabilir!");
```

Dosya okunabilir değilse program "Bu dosya okunabilir!" yazmadan yoluna devam edecektir. (Unix ortamında varlığını görebildiğimiz her dosyanın okuma izni bulunmayabilir.)

Dosya yazılabilir mi? is_writable()

Programda kullanmaya karar vermeden önce bir dosyanın yazılabilir olup olmadığını sınayan bu fonksiyon, dosya yazılabilir ise true/doğru, değilse false/yanlış sonuç verir. Örnek:

```
if ( is_writable ( "bir_dosya.txt" ) )
print ("Bu dosyaya yazılabilir!");
```

Dosya yazılabilir değilse program "Bu dosyaya yazılabilir!" yazmadan yoluna devam edecektir. (Unix ortamında varlığını görebildiğimiz hatta okuyabildiğimiz her dosyanın yazma izni bulunmayabilir.)

Dosya çalıştırılabilir mi? is_executable()

PHP programında kimi zaman sunucunun kullanmamıza izin verdiği haricî programları çalıştırız; PHP programımızın düzgün işlemesi bu harici programa bağlı olabilir. Böyle bir programı çalıştırmaya karar vermeden önce bir dosyanın çalıştırılabilir olup olmadığını sınayan bu fonksiyon, dosya çalıştırılabilir ise true/doğru, değilse false/yanlış sonuç verir. Örnek:

```
if ( is_executable ( "bir_dosya" ) )
print ("Bu dosya çalıştırılabilir!");
```

Dosya çalıştırılabilir bir program değilse PHP programı "Bu dosya çalıştırılabilir!" yazmadan yoluna devam edecektir. (Unix ortamında varlığını görebildiğimiz her pprogram dosyasının çalıştırma izni bulunmayabilir.)

Dosya boyutu: filesize()

Adını verdiğimiz dosyanın boyutunu byte olarak bildirir: Örnek:

```
print ("Dosyanın boyutu:");
print filesize( "bir_dosya.txt" );
```

Dosyaya son erişim tarihi: fileadate(), filemtime() ve filectime()

Adını verdiğimiz dosyaya son erişim tarihini bildirir. ne var ki bu bilgi Unix'in "epoch" biçimindedir: Örnek:

```
print ("Dosyanın son erişim tarihi:");
$dosya_tarihi = fileadate( "bir_dosya.txt" );
print ( " $dosya_tarihi " );
```

Eğer bu dosyanın son erişim tarihi 28 Temmuz 2000, Cuma 24:00:00 ise, PHP, Browser penceresine 964731600 yazacaktır. Bu komutun ürettiği bilgiyi, date() fonksiyonu ile anlaşılabilir hale getirebiliriz:

```
print ("Dosyanın son erişim tarihi:");
$dosya_tarihi = fileadate( "bir_dosya.txt" );
print date("D d M Y G:i:s H", $dosya_tarihi );
```

Bu kez PHP Browser penceresine 28 Jul 2000 242:00:00 00 yazdıracaktır. Tarih verilerinin date() fonksiyonu ile biçimlendirilmesini daha sonra ayrıntılı ele alacağız.
filemtime(), bir dosyanın son değiştirildiği tarihi; filectime(), ise oluşturulduğu tarihi, yine Unix Epoch biçiminde bildirir; bu verinin anlaşılır biçimde görüntülenmesi için PHP'nin date() fonksiyonu kullanılır.

Dosyalar oluşturma ve silme

PHP ile yapabileceğimiz önemli dosya işlemlerinin başında olmayan bir dosyayı oluşturmak ve olan bir dosyayı silmek gelir. PHP'nin dosya oluşturma komutu touch() fonksiyonudur. Bu fonksiyona

oluşturulmasıın istediğimiz dosyanın adını vermemiz gerekir. Örnek:

```
<?php
$dosya_dizin = "/inetpub/wwwroot/";
                touch ("$dosya_dizin/yeni_belge.txt");
print ("yeni_belge adlı bir dosya oluşturuldu!");
?>
```

Bu programı kişisel Web sunucuda denerken yeni dosyanın oluşturulacağı dizin olarak "/" işaretiyle sadece kökdizini belirtirseniz, dosya C: diskinde kökdizinde oluşturulur. Bu programı gerçek sunucuda çalıştırabilmek için yazma/okuma izni bulunan ve Web sunucunun erişebileceği bir dizinin adını vermeniz gerekir. Örneğin:

```
<?php
$dosya_dizin = "/wwwroot/mycgiserver.com/members/uNhM13Qnm/";
touch ("$dosya_dizin/yeni_belge.txt");
print ("yeni_belge adlı bir dosya oluşturuldu!");
?>
```

Bu komutla oluşturacağınız dosya içi boş bir metin dosyası olacaktır. Eğer belirttiğiniz dizinde bu adı taşıyan bir dosya varsa, PHP dosyanın içeriğine dokunmayacak, fakat dosyanın erişim ve değişim tarihlerini değiştirecektir.

PHP ile mevcut bir dosyayı silmek için **unlink()** fonksiyonunu kullanırız. Bu fonksiyon da silinecek dosyanın adı ile birlikte yolunu ister. Örnek:

```
<?php
$dosya_dizin = "/wwwroot/mycgiserver.com/members/uNhM13Qnm/";
unlink ("$dosya_dizin/yeni_belge.txt");
print ("yeni_belge adlı dosya silindi!");
?>
```

Bu komut Windows sistemlerinde işlemeyebilir.

Dosya açma

PHP'de bir dosyanın içeriğini alarak sayfalarımızda kullanma veya bir dosyanın içeriğini değiştirmek gibi işlemler için önce dosyanın açılmış olması gerekir. Bunu gerçekleştiren fopen() fonksiyonudur. Bu fonksiyonla bir dosyayı okumak ('r'), yazdırmak ('w') veya ek yapmak ('a') için açabiliriz. Bu fonksiyon dosyanın başarıyla açılması halinde bir tamsayı verecektir. PHP programlarımızda, açılan dosyanın mutlata ona işaret eden bir değişkene (file pointer) bağlı olması gerekir; daha sonra bu dosya ile ilgili bütün işlemleri bu işaret değişkeni ile yaparız. Örnek:

```
$dosya = fopen( "bir_dosya.txt" , 'r' );
```

PHP, bu dosyayı sadece okumak amacıyla açacak ve fonksiyondan dönen değeri $dosya değişkenine atayacaktır. Olmayan bir dosyayı açmak istediğimiz zaman PHP hata mesajı verir. Bir dosyayı yazmak amacıyla açacağımız zaman, bu kodu şöyle yazarız:

```
$dosya = fopen( "bir_dosya.txt" , 'w' );
```

Olmayan bir dosyayı yazmak amacıyla açmak istediğimizde PHP önce bu dosyayı oluşturur. Bir dosyaya ek yapmak istediğimiz zaman ise kodumuz şu şekilde yazılır:

```
$dosya = fopen( "bir_dosya.txt" , 'a' );
```

Olmayan bir dosyayı ek yapmak amacıyla açmak istediğimizde PHP hata mesajı verir.

Bir Fonksiyonu durdurmak için: Öl!

Bir PHP fonksiyonunun kendisinden beklenen işi yapamaması halinde oracıkta durdurulmasını die() komutu sağlar. "Öl!" anlamına gelen bu fonksiyona ekranda görüntülenmesini istediğimiz bir metni parametre olarak verebiliriz. Bu bölümdeki örnekler bu komutla birlikte şöyle yazılabilir:

```
$dosya = fopen( "bir_dosya.txt" , 'a' ) or die ("Dosya açılamıyor!") ;
```

Bu komutu kişisel Web sunucusunda denerken, dosyanın açılacağı dizinin yolunu belirtmemiz gerekir. Örneğin:

```
<?php
$dosya_dizin = "/inetpub/wwwroot/";
if ($dosya = (fopen ("$dosya_dizin/bir_dosya.txt" , 'r') ) ) {
        print ("Dosya açıldı!");
        }
        else {
        print ("Dosya açılamadı!");
        }
?>
```

PHP, bu dosyayı açtığında Browser penceresinde dosyanın açıldığını belirten mesajı görüntülecektir. Bu işlemi Internet'teki bir Web sunucuda uygulayabilmek için yazma/okuma izni bulunan ve Web sunucunun erişebileceği bir dizinin adını vermeniz gerekir. Örneğin:

```
<?php
$dosya_dizin = "/wwwroot/mycgiserver.com/members/uNhM13Qnm/";
if ($dosya = (fopen ("$dosya_dizin/bir_dosya.txt" , 'r') ) ) {
        print ("Dosya açıldı!");
        }
        else {
        print ("Dosya açılamadı!");
        }
?>
```

Açtığımız bir dosya ile yaptığımız bütün işlemler bittikten sonra, dosyanın kapatılması gerekir. Dosya kapatma işlemini fclose() fonksiyonu yapar. Bu fonksiyona parametre olarak dosya adını değil, dosyanın işaretçisi olan değişkenin adını veririz. Örnek:

```
fclose ( $dosya );
```

Dosya okuma: fgets(), fread() ve fgetc()

Açtığımız bir dosyanın içindeki yazıları çoğu zaman programımıza satır satır okuturuz. PHP'de bir dosyanın içeriğini satır satır okutmamızı fgets() fonksiyonu sağlar. Bu fonksiyona daha önce açılmış olan dosyanın işaret değişkenin adını ve okunmasını istediğimiz asgari byte ölçüsünü parametre olarak veririz. fgets() fonksiyonu verdiğimiz uzunluk ölçüsüne ulaşmadan önce dosyada bir yeni satır işareti görürse, veya dosyanın sonuna ulaşırsa, okumaya son verir. Bu fonksiyonu çoğu zaman bir döngünün içinde kullanırız. Ancak döngünün hata vermemesi için, PHP'nin dosyanın sonuna ulaştığında döngüyü durdurmamız gerekir. fgets() fonksiyonunun okuyacağı satırı bir fonksiyona değer olarak verebilir ve daha sonra bu değeri programlarımızda kullanabiliriz. Örnek:

```
<?php
$dosya_dizin = "/inetpub/wwwroot/";
if ($dosya = (fopen ("$dosya_dizin/bir_dosya.txt" , 'r') ) ) {
        print ("Dosya açıldı!<br>");
        }
        else {
        print ("Dosya açılamadı!");
        }
while ( ! feof ($dosya ) ) {
        $satir = fgets ( $dosya, 1024 ) ;
        print ("$satir<br>");
        }
fclose ($dosya);
?>
```

Bu program kişisel Web sunucuda önce varolan bir dosyayı açıyor; ve bu dosyanın işaretçisi olarak $dosya değişkenini kullanıyor. Daha sonra bir while() döngüsü içinde bu değişkeni ve 1 KB (1024 byte) ölçüsünü parametre olarak vererek fgets() fonksiyonu ile dosyadan bir satır okutuyoruz. fgets() fonksiyonundan dönen bir satırlık metni, burada $satir değişkenine atıyoruz; ve daha sonra print() fonksiyonu ile bu satırı Browser penceresine gönderiyoruz. Bu işlemi Internet'teki bir Web sunucuda uygulayabilmek için yazma/okuma izni bulunan ve Web sunucunun erişebileceği bir dizinin adını vermeniz gerekir. Bu işlemin içinde yapıldığı while() döngüsünün devam şartı olarak kullandığımız ifadede yer alan feof() fonksiyonu bir dosyanın dosya-sonu (end-of-file) durumuna ulaşıp ulaşmadığını sınar. PHP, her dosyanın sonunda yer alan eof (Ctrl-z, ^z) işaretine eriştiği anda feof() fonksiyonu doğru/true değerini verir. while() döngüsü, feof() doğru değilken devam etmek üzere kurulmuş olduğu için, dosya sonuna ulaştığımızda döngü duracaktır. Okuma işlemini durabilen bir döngüye bağlamazsak, fgets() PHP'nin bir programın sona ermesi için verilmiş varsayılan süresi doluncaya kadar dosyayı okumak isteyecektir.

Kimi zaman dosyalarımızın içeriğini satır-satır okutmak yerine, kendi tayin edeceğimiz uzunlakta parçalar halinde okutmak isteriz. Bunu, fread() fonksiyonu ile sağlarız. Örnek:

```
<?php
```

```
$dosya_dizin = "/inetpub/wwwroot/";
if ($dosya = (fopen ("$dosya_dizin/bir_dosya.txt" , 'r') ) ) {
        print ("Dosya açıldı!<br>");
        }
        else {
        print ("Dosya açılamadı!");
        }
while ( ! feof ($dosya) ) {
        $paragraf = fread ( $dosya, 1024 ) ;
        print ("$paragraf<br>");
        }
fclose ($dosya);
?>
```

fread() fonksiyonu da daha önce açılmış olan dosyanın işaret değişkenin adını ve okunmasını istediğimiz asgari byte ölçüsünü parametre olarak alır. fread() fonksiyonu verdiğimiz uzunluk ölçüsüne ulaşmadan önce dosyada bir yeni satır işareti görürse, veya dosyanın sonuna ulaşırsa, okumaya son verir. Bu fonksiyondan yararlanırken, verdiğiniz uzunluk ölçüsünün, almak istediğiniz metin parçasına uygun olduğunu sınamalısınız. fread(), bu ölçüye ulaştığında okumayı keser. Buradaki örneği 1024 byte ölçüsünü değiştirerek ve mesela 1, 2, 3 yaparak deneyebilir ve böylece vereceğiniz ölçünün okunan metnin uzunluğunu nasıl tayin ettiğini görebilirsiniz.

fseek() ile ölçü belirleme

PHP'nin dosya işleminde kullabileceğiniz bir diğer fonksiyonu fseek() adını taşır. Daha önce açılmış olan dosyanın işaret değişkenin adını ve ve programın bu metin içinde zıplamasını istediğiniz noktanın dosyanın başından itibaren byte değerini parametre olarak alan bu fonksiyon ile, bir dosyanın içinnde istediğimiz yere gitme imkanı vardır. Örnek:

```
<?php
$dosya_adi = "/inetpub/wwwroot/bir_dosya.txt";
if ($dosya = (fopen ($dosya_adi , 'r') ) ) {
        print ("Dosya açıldı!<br>");
        }
        else {
        print ("Dosya açılamadı!");
        }
$dosya_boyut = filesize($dosya_adi);
$olcu = (int) ($dosya_boyut / 2 );
while ( ! feof ($dosya) ) {
        $paragraf = fread ( $dosya, $olcu) ;
        print ("$paragraf<br>");
        }
fclose ($dosya);
?>
```

Burada, okutulacak dosyanın boyutunun yarısını atadığımız $olcu değişkenini, okutulacak metnin ölçüsü olarak kullanıyoruz. Bu durumda PHP, dosyayı iki paragraf halinde görüntüleyecektir.

Dosyalarımızın içeriğini satırlar veya belirli ölçüde parçalar halinde değil de, tek-tek karakter olarak okutmak için fgetc() fonksiyonundan yararlanırız. Bu fonksiyon, daima 1 byte ölçüsünde (bir karakter) metin okuyabileceği için, parametre olarak sadece daha önce açılmış olan dosyanın işaret değişkenin adını alır. Örnek:

```
<?php
$dosya_adi = "/inetpub/wwwroot/bir_dosya.txt";
if ($dosya = (fopen ($dosya_adi , 'r') ) ) {
        print ("Dosya açıldı!<br>");
        }
        else {
        print ("Dosya açılamadı!");
        }
while ( ! feof ($dosya) ) {
        $karakter = fgetc ( $dosya ) ;
        print ("$karakter<br>");
        }
fclose ($dosya);
?>
```

Burada fgetc() fonksiyonundan dönen değeri (yani dosyadan okunan bir karakteri), $karakter değişkenine atıyoruz ve daha sonra print() fonksiyonu ba karakteri ve HTML'in satır bölme kodu olan <br> işaretini Browser penceresine gönderiyor. Programı bu şekliyle sınarsanız, dosyadaki metnin tek karakter olarak Browser penceresinin soluna dizildiğini göreceksiniz. Programı <br> kodunu silerek çalıştırırsanız, bu kez dosyadaki metinde var olan satır sonu işaretlerinin de kaldırıldığını ve metnin bir paragraf oülarak görüntülendiğıini görebilirsiniz.

Dosyaya yazma ve ek yapma: fwrite() ve fputs()

Bir dosyaya yazma veya ek yapma, PHP açısından aynı işlemdir; sadece dosyaların açılışında fark vardır. Hatırlayacaksınız, bir dosyayı yazmak amacıyla açmak için:

```
$dosya = fopen( "bir_dosya.txt" , 'w' ) or die ("Dosya açılamıyor!") ;
```

ek amacıyla açmak için ise

```
$dosya = fopen( "bir_dosya.txt" , 'a' ) or die ("Dosya açılamıyor!") ;
```

kodunu yazmamız gerekir. Dana sonra yapılacak yazma ve ekleme işlemlerinin farkı, 'w' parametresi ile açılan dosyaya yazma işlemi en başından başlar ve devam eder; 'a' parametresi ile açaılan dosyaya yazma işlemi ise en sondan başlar ve devam eder.

PHP'nin bir dosyaya metin yazdırma fonksiyonları olan fwrite() ve fputs() aynı biçimde yazılır ve aynı işlevi yerine getirirler; aralarında kesinlikle fark yoktur. Örnek:

```
<?php
$dosya_adi = "/inetpub/wwwroot/bir_dosya.txt";
$dosya = fopen ($dosya_adi , 'w') or die ("Dosya açılamadı!");
$metin = "Bu satır dosyaya yazılacak: Merhaba Dünya!\n";
fwrite ( $dosya , $metin ) ;
fputs ( $dosya , "Bu satır ise sonradan eklenecek\n" ) ;
fclose ($dosya);
?>
```

Bu programı çalıştırdığınızda, bir_dosya.txt adlı dosyada mevcut bütün içerik silenecek ve yerini $metin değişkeninin içerdiği "Bu satır dosyaya yazılacak: Merhaba Dünya!" yazısı ile "Bu satır ise sonradan eklenecek" cümlesi alacaktır. Her iki metnin sonunda da yeni satır işareti bulunduğuna dikkat edin. Bu programda dosya açma kıomutundaki 'w' parametresini siler, yerine 'a' yazarsanız, bu metinlerin dosyanın içeriğine eklendiğini görebilirsiniz.

Kullanımdaki dosyayı kilitleyin!

Web sunucusundaki dosyalarımızla sadece bir kişi işlem yapıyor olsa idi, bir sorun olmazdı; ne var ki, bir Web sitesine aynı anda birden fazla kişi eriyebilir ve dosyalarla işlem yapan programları çalıştırıyor olabilir. Bu, PHP'nin dosya işlemlerine engel olabilir. Bu sebeple, işlem için açacağımız bir dosyayı, önce kilitlemek yerinde bir önlem sayılır. Bunu, flock() fonsiyonu ile yaparız; bu fonksiyona kilitlemek istediğimiz dosyanın işaret değişkeninin adını ve kilit türünü belirten endeks sayısını parametre olarak yazarız. Örnek:

```
<?php
$dosya_adi = "/inetpub/wwwroot/bir_dosya.txt";
$dosya = fopen ($dosya_adi , 'w') or die ("Dosya açılamadı!");
flock ( $dosya , 2); // dosyayı kilitle
$metin = "Bu satır dosyaya yazılacak: Merhaba Dünya!\n";
fwrite ( $dosya , $metin ) ;
fputs ( $dosya , "Bu satır ise sonradan eklenecek\n" ) ;
flock ( $dosya , 3); //dosyayı kilidini aç
fclose ($dosya);
?>
```

Bu fonksiyon ile kullanabileceğimiz endeks parametreleri şunlardır:

1	Paylaşım	Diğer proseslerin dosyayı paylaşmalarına imkan verir
2	Tam	Diğer proseslerin dosyaile işlem yapmasına engel olur
3	Serbest	Dosyanın 1 veya 2 olan kilidini kaldırır

Bir dosya, herhangi bir PHP programı tarafından kilitlendiği anda, aynı dsyayı daha sonra kilitlemeye kalkan diğer programlar kendilerinden önce konulmuş kilide saygı gösterirler.

Dizinlerle İşlemler

PHP ile sunucuda, Web dizini olarak kullandığımız alanda yeni dizinler oluşturabiliriz, silebiliriz, ve bunlar hakkında bilgi edinebiliriz. Bu işlemleri Web sunucuda yapabilmek için Web dizininde okuma ve yazma izni bulunması gerekir.

Dizin içeriğini listeleme: opendir() ve readdir()

Belirttiğiniz bir dizindeki tüm dosya ve alt-dizin adlarını bir liste, hatta ilişkilendirilmiş hypertext (köprü, link) listesi olarak sunabilmek için önce dizini içeriğini okuyabilmek için opendir() fonksiyonu ile açmak, sonra da içindeki bilgileri readdir() fonksiyonu ile edinmek gerekir. readdir() fonksiyonu dizinin içindeki alt-dizin ve dosyaların adlarını sırayla, tek-tek okur. Bu fonksiyondan gelen bilgileri bir değişkene atayarak ve bir döngü içinde yazdırarak, dizin listesi çıkartabiliriz. Örnek

```
<?php
$dizin_adi = "./";
$dizin = opendir ($dizin_adi);
while ( gettype ( $bilgi = readdir( $dizin ) ) != boolean ) {
        if ( is_dir( "$dizin_adi/$bilgi" ) )
        print " [Dizin] " ;
        print ("<A href=\"$dizin_adi/$bilgi\">$bilgi</A><br>\n");
}
closedir ($dizin);
?>
```

Kişisel Web sunucuda sınama amacıyla çalıştırabilmek için dizin adı olarak bu dosyanın içinde bulunduğu dizini verebiliriz. opendir() fonksiyonu da okunmak amacıyla açacağı dizine işaret eden bilgiyi bir değişkene yazmak zorundadır; bu işaret değişkenine burada $dizin adını veriyoruz. $dizin değişkeninin işaret ettiği dizini okuyan readdir() fonksiyonundan dönen bilgileri ise $bilgi değişkenine yazıyoruz. readdir() fonksiyonu dizin bilgisini okumanının sonuna vardığında, bir dosya ve dizin adı yerine doğru veya yanlış şeklinde bir mantıksal (boolean) değer verecektir; buradaki while döngüsü $bilgi değişkeninin türünü gettype() fonksiyonu ile sürekli sınayarak, henüz dizin ve dosya adı edinildiği sırada bu bilgileri print() fonksiyonuna verecektir. Bu bilginin bir dizine ait olup olmadığını is_dir() fonksiyonu ile sınayan if döngüsü ise bilgi bir dizine aitse, bu bilgnin baştarafına "[Dizin]" kelimesini yazdıracaktır.

Dizin oluşturma: mkdir()

PHP programlarımız gerektiğinde sunucunun yazma yetkisi verdiği Web'e açık kök ve alt-dizinlerde yeni dizinler oluşturabilir. Bunu, mkdir() fonksiyonu ile yaparız. Bu fonksiyona oluşturulacak dizinin adı ve 8 tabanlı (octal) sayı cinsinden ve önüne bir sıfır konarak dizinin okuma/yazma/çalıştırma izinlerini belirten parametre verilir. Örnek:

```
mkdir ("deneme", 0777);
```

Burada yer alan 0777, bu dizinin herkes için okuma ve yazma izni olduğunu gösterir. Bu parametre sadece Unix işletim sisteminde çalışan sunucular için geçerlidir.

Dizin silme: rmdir()

PHP programlarımızda gerektiğinde sunucunun yazma yetkisi verdiği Web'e açık kök ve alt-dizinlerde mevcut dizinleri silebiliriz. Bunu, rmdir() fonksiyonu ile yaparız. Bu fonksiyona oluşturulacak dizinin adı parametre verilir. Örnek:

```
rmdir ("deneme");
```

MySQL Veritabanı

mySQL veri türleri

MySQL'de bir çok veri türü oluşturulabilir. Ancak Web programları açısından önemli olan bir kaçı ve özellikleri şöyle saralanabilir:

INT	Tamsayı: -2147483648'den 2147483647 kadar değişen diziye "signed" (işaretli), 0'dan 4294967295'e kadar değişenine "unsigned" (işaretsiz) denir.
VARCHAR(n)	n sayısını geçmemek şartıyla değişen boyutta karakter olabilir.
CHAR(n)	Kesinlikle n sayısı kadar karakter olabilir.
TEXT	En fazla 65535(2^16-1) karakter alabilen metin alanı.
MEDIUMTEXT	En fazla 16777215(2^24-1) karakter alabilen metin alanı.
DATE	1000-01-01'den 9999-12-31'e kadar değişebilen tarih alanı.
TIMESTAMP	1 Ocak 1970'den 18 Ocak 2038'e kadar olan ve Yıl+Ay+Gün+Saat+Dakika+Saniye biçimindeki zaman bilgisi.

MySQL'de bir tablo oluşturmak için gerekli CREATE TABLE komutu şöyle kullanılır:

```
CREATE TABLE uyeler (adi VARCHAR(30), soyadi VARCHAR(30), üye_no INT ) ;
```

Bu komutla, "uyeler" isimli üç sütunlu bir tablo oluşturulur: birinci ve ikinci sütunlarda en fazla 30, karakterlik değişen boyutta alfanümerik değerler yer alırken, üçüncü sütunda sadece tam sayı olan değerler bulunabilir. Bu komutla oluşturulan tabloya INSERT INTO komutuyla veri girebilirsiniz:

```
INSERT INTO uyeler (adi, soyadi, uye_no) VALUES ('Faruk','Taç','1234')
```

Bir tablonun oluşturulması ile içine veri yerleştirilmesi komutları ayrı ayrı zamanlarda, ayrı işlemler olarak yapılabileceği gibi, toplu bir metin halinde, otomatik olarak da yapılabilir.
MySQL veritabanından bilgi edinmek için SELECT komutunu kullanırız:

```
SELECT * FROM uyeler ;
```

Bu, MySQL'e, uyeler adlı tablodaki bütün değerlerin okunmasını bildirir. Buradaki "*" işareti, "bütün sütunlardaki bütün değerler" anlamına gelir. Diyelim ki yukardıda oluşturduğumuz tablonun sadece "adi" ve "soyardi" sütunlarındaki bilgileri almak isteseydik, bu komutu şöyle yazacaktık:

```
SELECT adi soyadi FROM uyeler ;
```

Bir veritabanındaki bilgilerin yenileriyle değiştirilmesini, yani veritabanı dosyasının güncelleştirilmesini UPDATE komutu sağlar. Bu komutu kullanarak veritabanımızdaki bazı kutucukların içindeki bilgileri değiştirebiliriz. Veritabanı dosyalarını güncelleştirme zorunluğu bulunması ise bize veritabanı tasarımının çok önemli olduğunu gösterir. Örneğin:

```
UPDATE uyeler SET adi = "Şükran" ;
```

Bu komut, veritabanındaki bütün satırlarda, birinci sütundaki değerleri "Şükran" olarak değiştirmekle sonuçlanırdı. Amacımız bu ise, sorun değil; ancak çoğu kez MySQL'e hangi satırda (veritabanı tekniğindeki terimle söylersek, hangi kayıtlarda) değişiklik yapılacağını daha ayrıntılı sölememiz gerekir. Veritabanı dosyamızı oluştururken, her kaydın diğer kayıtlarda olmayan (unique) bir sütun (bunu da veritabanı tekniğindeki terimle söylersek. alan) bulunmalıdır, ki MySQL'e yapılacak değişikliğin tam yerini söyleyelim. Örneğin

UPDATE uyeler SET adi = "Şükran" WHERE uye_no = 1234;

MySQL bu komutu alınca sadece üye numarası 1234 olan kişinin (yani uye_no alanındaki değer 1234 olan kaydın) "adi" alanındaki değeri silecek ve yerine verdiğimiz yeni değeri yazacaktır. Böyle birincil alanı bulunan, iyi düşünülmüş bir veritabanından seçim yapmak da kolay olur. Örneğin:

SELECT adi soyadi FROM uyeler WHERE uye_no >= 123;

deyimi ile tablomuzda bulunan kayıtlardan sadece üye numarası 123'den büyük olanları seçebiliriz.

Bir MySQL veritabanındaki kaydı silmek için DELETE komutunu kullanırız:

DELETE FROM uyeler WHERE uye_no = 1234;

Veritabanında sadece bir kayıtta üye numarası 1234 olacağı için bu komutla sadece bir satır silinecektir. Bu komutu, diyelim ki üyelik kaydını yenilememiş kişilerin tümünü silmek için de kullanabiliriz. Veritabanımızda üyelik kaydının yenilendme tarihini gösteren bir alan bulunduğunu varsayalım:

DELETE FROM uyeler WHERE yenileme_tarihi < 2000-01-31;

Bu komutla, üyeliğini yenileme tarihi 31 Ocak 2000'den eski olan bütün üyelerimizin kaydını veritabanından silmiş oluruz.

Bu komutların MySQL'in DOS komut işlemcisi ile komutsatırından yapılacağını belirtmemiz gerekir. Bunu yapabilmek için MySQL Server'ın Windows'da çalıştırılması gerekir. MySQL'in paylaşım sürümünü kullanmak için Windows sistemlerinde mysqld-shareware.exe programını çalıştırmanız gerekir. Bunun için DOS komut istemcisi penceresinde "C:/mysql/bin" dizinine giderek, şu komutu vermemiz yeter:

mysqld-shareware

MySQL sürücülerinizin kişisel Web sunucusunda başarıyla çalıştığına, kitapçığın baş tarafında belirttiğimiz küçük alıştırmayı yaparak emin olduktan sonra gerçek bir veritabanı dosyası yazabiliriz.

Yukarıda gördüğümüz komutları komut istemci satırından tek tek verebileceğimiz gibi, bir düzyazı dosyasında toplayıp, MySQL programına da otomatik olarak yaptırabiliriz. Bu dosyaya, içindeki verileri alıp veritabanına boca edeceğimiz için, Dump dosyası denir.

Aşağıdaki metni MySQL programının kurulu olduğu dizindeki /bin/ alt-dizinine (muhtemelen c:\mysql\bin) veri.dump adıyla kaydedin (Notpad kullanıyorsanız, dosya adına .txt eklendiğine dikkat edin!)

CREATE TABLE calisanlar (id INT NOT NULL AUTO_INCREMENT, PRIMARY KEY (id), adi VARCHAR(20), soyadi VARCHAR(20), adres VARCHAR(60), pozisyon VARCHAR(60));
INSERT INTO calisanlar VALUES (1 , 'Sahika' ,'Tabak' ,'PCLife Dergisi, Istanbul' , 'Yazar');
INSERT INTO calisanlar VALUES (2 , 'Faruk' , 'Tac' , 'Işık Kultur Merkezi, Bursa' , 'Yonetmen');

Bu metnin sadece üç satır olmasına, örneğin Notpad'de Düzen menüsünde Sözcük Kaydır maddesinin işaretsiz olmasına dikkat edin. Daha sonra yine DOS komut istemcisi penceresinde MySQL programının dizininde /bin alt-dizinine gidin ve şu komutu yazın:

```
mysqladmin -u root create veri
```

MySQL veri adlı veritabanının oluuştuğunu bildirecektir. Şimdi içi boş bir veri dosyamız oldu. Yazdığımız dump dosyasındaki bilgileri veritabanı dosyasına işletmek işini MySQL yapacakatır. Bunu, şu komutla yapabiliriz:

```
mysql –u root veri < veri.dump
```

İşlerin yolunda gidip gitmediğini c:\mysql\data dizininde veri adlı bir klasör oluşturulduğunu kontrol ederek anlayabiliriz. Bu klasörün içinde calisanlar.frm, calisanlar.isd ve calisanlar.ism adında dosyalar bulunması gerekir.

PHP-MySQL İlişkisi

Böylece, MySQL ile bir veritabanı dosyası oluşturma işlemi bitmiş oldu. bir PHP programı yazarak bu veritabanındaki kayıtları okutabiliriz. Bunu yapmadan önce yine hızlı şekilde PHP-MySQL ilişkisinini sağlayan fonksiyonlara gözatalım.

PHP programlarımızda veritabanından yararlanabilmek için programın önce Web sunucusu aracılığıyla veritabanı dosyası ile bağlantı kurması gerekir. Başka bir deyişle, PHP programının veri ile arasında bir yol açması gerekir. MySQL açısından ise bu bağlantı, veri sunucusunda yeni bir oturum açılması anlamına gelir. İki program arasındaki bu ilişkiyi PHP'nin **mysql_connect()** fonksiyonu yapar. Bu fonksiyonun alabileceği üç parametre vardır:

```
$veri_yolu = mysql_connect ("localhost" , "root" , "parola" );
```

Burada "localhost" yerine MySQL programının parçası olarak çalıştığı sunucunun adı yazılır. "root" bu MySQL sunucusunda açılacak oturumun kimin adına açılacağını belirter. "root" kelimesi, sunucunun yönetici olarak oturum açılacağı anlamına gelir: "parola" kelimesinin yerine de MySQL'i kurarken belirlediğimiz bir kullanıcı parolası varsa, onu yazarız. Bu komutta yer alan $veri_yolu değişkeni, açılacak veri yolunun, PHP ile MySQL veritabanı sunucusu arasındaki bağın tanıtıcı işareti olacaktır. Bu bağlantı kurulduktan sonra, açtığımız veri yolundan gelecek bilgiler ve veritabanına gidecek bilgiler bu değişken aracılığıyla gerçekleşecektir. Veri sunucusu ile veri yolu bağlantısı kurulursa, bu değişken değer tutar hale gelir; bağlantı kurulamazsa bu değişken boş kalır. mysql_connect() fonksiyonunun başarılı olup olmadığını bu değişkenin durumunu sınayarak anlayabiliriz. Örneğin:

```
$veri_yolu =mysql_connect("kara-murat", "root");
if ( ! $veri_yolu) die ("MySQL ile veri bağlantısı kurulamıyor!);
```

Burada veri sunucusunun bulunduğu Web sunucusunun adının "kara-murat" olduğuna, ve ooturumun "root" yetkileriyle açıldığına dikkat edin. İkinci satırdaki if deyimi, $veri_yolu değişkeninin değer içerip içermediğine bakıyor ve değişkende bir değer yoksa, bağlantı kurma girişini durdurarak, ziyaretçiye hata mesajı gönderiyor.

Bağlantı başarıyla kurulduktan sonra PHP programı, bu yoldan, veritabanı sunucusuna, hangi veritabanı dosyasından yararlanmak istediğini bildirmelidir. Buna veritabanı dosyası seçme işlemi denir ve mysql_select_db() fonksiyonu ile yapılır:

```
mysql_select_db( "veritabanın_adı" , $veri_yolu ) or die ("Veritabanı
açılamıyor!".mysql_error() );
```

Bu fonksiyonun başarıyla icra edilip edilmediği fonksiyondan dönen değerin true/doğru veya false/yanlış olmasından anlarız. Bu değer false ise bu deyimin die() bölümü icra edilecek ve Browser penceresine veritabanının açılamadığı mesajıyla birlikte MySQL'in hata mesajı da gönderilecektir. PHP'nin MySQL veritabanını seçememesi çoğu zaman kullanıcı yetkilerinin Internet ziyaretçilerini kapsayacak şekimde düzenlenmemiş olmasından kaynaklanır. Bu durum gerçek Web sunucusunda ortaya çıkarsa, Web sunucusu yönetimine başvurmak gerekir.

Şimdi bu anlattıklarımızı biraraya getiren kolay bir PHP programıyla biraz önce oluşturduğumuz "veri" adlı veritabanından bir birinci kişiye ait verileri "okutarak, HTML sayfamızda kullanalım. Aşağıdaki programı, veri_01.php adıyla kaydedelim:

```
<?php
$veri_yolu = mysql_connect("kara-murat", "root");
if ( ! $veri_yolu ) die ("MySQL ile veri bağlantısı kurulamıyor!");
mysql_select_db("veri" , $veri_yolu)
        or die ("Veritabanına ulaşılamıyor!" . mysql_error() );
$sonuc = mysql_query("SELECT * FROM calisanlar",$veri_yolu);
        printf("Adı: %s<br>\n", mysql_result($sonuc,0,"adi"));
        printf("Soyadı: %s<br>\n", mysql_result($sonuc,0,"soyadi"));
        printf("Adresi: %s<br>\n", mysql_result($sonuc,0,"adres"));
        printf("Görevi: %s<br>\n", mysql_result($sonuc,0,"pozisyon"));
?>
```

Burada, mysql_connect() fonksiyonu ile "kara-murat" isimli sunucuda root adına MySQL sunucu ile bağ kurduktan sonra mysql_select_db() fonksiyonu ile bu bağı kullanarak veri isimli veritabanından veri çekeceğimizi bildiriyoruz. Daha sonra mysql_query() fonksiyonu ile bu veritabanındaki "calisanlar" isimli tablodan "herşeyi" seçiyoruz ve seçilenleri $sonuc dizi-değişkeninde topluyoruz. $sonuc değişkenin değerlerini görüntülemek için PHP'nin özel bir fonksiyonu olan mysql_result() fonksiyonu kullanıyoruz. Burada metin biçimlendirmekte yararlandığımız printf() fonksiyonunu daha önce tanımıştık.

mysql_query() fonksiyonu, PHP'nin SQL dilini kullahnarak veritabanı işlemleri yapmasını sağlayan başlıca aracıdır. Yukarıda kısaca değindiğimiz bütün SQL komutlarıyla yazacağımız bütün "query" deyimlerini bu fonksiyon ile icra ettireceğiz. mysql_result() ise SQL değil, Data Manipulation Language (DML) denen başka bir veri-biçimlendirme dilinin inceliklerinden yararlanmamızı sağlar. Burada $sonuç değişkeninde veritabanı kayıt biçiminde tutulan verileri PHP'nin ve dolayısıyla HTML'in anlayacağı biçime çeviren bu fonksiyondur.

Şimdi bu programı biraz geliştirilim ve daha önce kendi kendine bilgi veren Form örneğimizi buraya uygulayalım; ancak bu kez, ziyaretçimizin vereceği bilgileri veritabanına ekleyelim; ve kendi adının veritabanına eklendiğini sayfadaki tabloyu güncelleyerek bildirelim. Önce şu programı, veri_02.php adıyla kaydedelim:

```
<?php
// Form doldurulduktan sonra program buradan başlıyor
if ( isset ( $HTTP_POST_VARS )) {
$veri_yolu = mysql_connect("server", "root");
if ( ! $veri_yolu ) die ("MySQL ile veri bağlantısı kurulamıyor!");
mysql_select_db("veri" , $veri_yolu) or die ("Veritabanına ulaşılamıyor!" . mysql_error() );
```

```
$ekle = mysql_query("INSERT INTO calisanlar ( adi , soyadi , adres , pozisyon ) VALUES
('$adi', '$soyadi', '$adres', '$pozisyon' )", $veri_yolu );
echo ("
        <HTML>
        <HEAD>
        <TITLE>PHP'de Veritabanı</TITLE>
        <meta http-equiv='content-type' content='text/html; charset=ISO-8859-9'>
        <meta http-equiv='Content-Type' content='text/html; charset=windows-1254'>
        ");
$sonuc = mysql_query("SELECT * FROM calisanlar", $veri_yolu);
echo ("
        <TABLE>
        <TR>
                <TD><B>Uzmanın Adı</B></TD>
                <TD><B>Çalıştığı Yer</B></TD>
                <TD><B>Görevi</B></TD>
                </TR>
        \n");
while ($satir = mysql_fetch_row($sonuc)) {
        printf("<TR><TD>%s %s</TD><TD>%s</TD></TD><TD>%s</TD></TR>\n",
$satir[1], $satir[2], $satir[3], $satir[4]);
        }
echo ("
        </TABLE>\n
        <p><B>Teşekkür ederiz.</B></P>
        <A HREF='index.php'>Ana sayfaya dönmek için tıklayınız</A>
        ");
}
// program ilk kez açılıyorsa buradan başlayacak
else {
echo ("
        <HTML>
        <HEAD>
        <TITLE>PHP'de Veritabanı</TITLE>
        <meta http-equiv='content-type' content='text/html; charset=ISO-8859-9'>
        <meta http-equiv='Content-Type' content='text/html; charset=windows-1254'>
        </HEAD>
        <BODY>
        <p><B>Mevcut Üyelerimiz</B></P>
        ");
$veri_yolu = mysql_connect("server", "root");
mysql_select_db("veri", $veri_yolu);
$sonuc = mysql_query("SELECT * FROM calisanlar", $veri_yolu);
echo ("<TABLE>
        <TR>
                <TD><B>Uzmanın Adı</B></TD>
                <TD><B>Çalıştığı Yer</B></TD>
                <TD><B>Görevi</B></TD>
        </TR>
        \n");
```

```
while ($satir = mysql_fetch_row($sonuc)) {
        printf("<TR><TD>%s %s</TD><TD>%s</TD></TD><TD>%s</TD></TR>\n",
$satir[1], $satir[2], $satir[3], $satir[4]);
        }
echo ("
        </TABLE>\n
        <p></p>
        <p><B>Siz de aramıza katılmak ister misiniz?</B></P>
        <FORM ACTION='$PHP_SELF' METHOD='POST'>
        <TABLE>
        <TR><TD>Adınız: </TD><TD><INPUT TYPE='TEXT' NAME='adi'></TD></TR>
        <TR><TD>Soyadınız: </TD><TD><INPUT TYPE='TEXT'
NAME='soyadi'></TD></TR>
        <TR><TD>İş Yeriniz: </TD><TD><INPUT TYPE='TEXT'
NAME='adres'></TD></TR>
        <TR><TDALIGN='left'>Göreviniz: </TD><TD><INPUT TYPE='TEXT'
NAME='pozisyon'></TD></TR>
        <TR><TD ALIGN='center'><INPUT TYPE='SUBMIT' VALUE='Defteri
imzala!'></TD><TD ALIGN='center'><INPUT TYPE='RESET' VALUE='Tümünü
sil!'></TD></TR>
        </TABLE>
        </FORM>
        </BODY>
        </HTML>
        ");
}
?>
```

Program, ilk kez çalıştığında, çalışmaya ikinci yarısındaki else() deyiminden itibaren icra ediliyor; ziyaretçilerimize mevcut üyelerimizin listesini veriyor ve üye olmak isteyip istemediğini soruyor. Arzu edenin üye olabilmesi için gerekli Form'u da sunuyoruz.

Programın her iki bölümünde de veri okuyan ve bunu görüntüleyen, yani programın canalıcı noktası, mysql_fetch_row() fonksiyonudur. PHP'nin DML araçlarından olan bu fonksiyonun marifeti, bir veritabanından elde edilen sonucu satır-satır okumasıdır. Nitekim, burada bu fonksiyondan dönen değeri $satir adını verdiğimiz dizi-değişkene yazıyoruz ve sonra printf() bu dizinin elemanlarını sırayla Browser penceresine gönderiyor. (Burada olduğu gibi $satır değişkenin içinde kaç kaç sütun olduğunu bildiğimiz durumlarda printf() fonksiyonunu döngüsüz kullanmak mümkündür. Ancak veritabanının sütun sayısını bilmiyorsak bunu sözgelimi count($satir) yoluyla öğrenip, bu bilgiyle bir for döngüsü kurmak yerinde olur.

Programın iki bölümü arasındaki tek fark, $HTTP_POST_VARS dizi-değişkeninin bir değer tutması halinde (yani ziyaretçi sayfayı açtığında karşısına çıkan Form'u doldurduğu ve gönderdiği zaman) çalışan birinci bölümünde, mysql_query() fonksiyonunun bu kez veritabanı dosyasına ziyaretçinin verdiği bilgileri işlemek üzere farklı bir SQL deyimi içermesidir. Nasıl yazıldığını daha önce ele aldığımız bu fonksiyon "calisanlar" tablosundaki dört alana elimizdeki dört değişkenin değerlerini SQL'in INSERT komutuyla ekliyor.

PHP'nin MySQL ile yapabileceğimiz veritabaynı yönetimi için 20'ye yakın fonksiyonu vardır; MySQL de bu fonksiyonlar yardımıyla çok sayıda iş yapabilir. Bu konuda ayrıntılı bilgiyi MySQL ve PHP'nin Internet sitelerin yanı sıra, http://hagen.let.rug.nl/~s0367672/pm_lin_e.htm adresinde bulabilirsiniz. Bu konuya son verirken, Form örneğindenden farklı olarak yukarıdaki veritabanı örneğinde

ziyaretçinin gireceği bilgilerle ilgili güvenlik önlemleri alınmadığına dikkatinizi çekerim.

Tırnak İşareti Sorunu

PHP4.0'ü tasarlayanlar, özellikle zararlı kodları gizlemekte kötüniyetli kişilerin yararlandığı tek ve çift tırnak işaretlerinin sisteme zarar vermesini önlemek amacıyla, ziyaretçi girdilerindeki tırnak işaretlerinin otomatik olarak Escape karakteri ile zararsız hale getirilmesini sağlamış bulunuyorlar. Buna göre bir ziyaretçi bir forma söz gelimi "PHP'nin yararları" yazacak olursa bu "PHP\'nin Yararları" haline dönecektir. Kimi zaman çirkin görünse de bu, bir sitenin güvenliği için önemli bir kazançtır.

Bu sistemin işleyebilymesi için sunucu yönetiminin PHP kurulumu sırasında php.ini dosyasında gereken düzenlemeyi yapmış olması gerekir. Bunun yapılmadığı durumlarda, programcı olarak siz, PHP'nin değişkenlerin değerlerinde gerektiğinde tersbölü işareti uygulamasını, bu değişkeni addslashes() fonksiyonu ile birlikte kullanarak çözümleyebilirsiniz. Örnek:

```
$yeni_degisken = addslashes($eski_degisken)
```

MySQL Yönetimi

Tüm SQL işlemleri için Php kullanılabileceği gibi veritabanı sunucusunda veritabanı, tablo oluşturmak, değiştirmek ya da diğer işlemler için farklı arayüzler kullanmak zaman kazandırıcı olabilir. Php kullanılarak geliştirilen PhpMyAdmin sayesinde web sunucu aracılığıyla tüm MySQL özelliklerini yönetebilmek mümkündür. Hem Unix hem de Windows ortamları için PhpMyAdmin tavsiye edilir. Windows'da da MySQL'i yönetmek için myAdmin adındaki yazılım da oldukça kullanışlıdır. MySQL'e ve Php'ye yeni başlayan kullanıcılar için veritabanını tanımak amacıyla ayrı bir yazılım kullanmaları tavsiye edilebilir.

Php'de kullanılabilecek MySQL Komutları :

mysql_affected_rows :	Bir önceki işlemde etkilenen satır sayısı
mysql_close :	Belirtilen MySQL bağlantısını kapatır
mysql_connect :	Sunucuya veritabanı bağlantısı açar
mysql_create_db :	MySQL'de veritabanı açar
mysql_data_seek :	Sonuç satırında belirtilen sıraya geçer
mysql_db_query :	MySQL'e sorgu gönderir
mysql_drop_db :	Sunucudan veritabanı siler
mysql_errno :	Bir önceki işlemdeki MySQL hata numarasını verir
mysql_error :	Bir önceki işlemdeki MySQL hata mesajını verir
mysql_fetch_array :	Sonuçları dizi değişkeni olarak alır
mysql_fetch_field :	Sonuç tablosundaki alan adını obje olarak alır
mysql_fetch_lengths :	Sonuç tablosundaki dizi değişkenin uzunluğunu alır
mysql_fetch_object :	Sonuç satırını obje olarak alır
mysql_fetch_row :	Sonuç tablosundan dizi değişkeni olur
mysql_field_name :	Sonuç tablsundaki sonucun tablodaki alan adını verir
mysql_field_seek :	Sonuç tablosunda sıra indisini belirtilen yere götürür
mysql_field_table :	Alan adı verilen sonucun tablo adını verir
mysql_field_type :	Sonuçtaki alanın hangi tip olduğunu belirtir
mysql_field_flags :	Sonuçtaki alanın hangi tür ekstra parametrelerle tanımlandığını belirtir
mysql_field_len :	Sonuçtaki alanın veritabanındaki uzunluğunu verir

mysql_free_result :	Sonuçlar için atanan hafızayı boşaltır
mysql_insert_id :	Bir önceki veri yerleştirmede oluşan otomatik veri değerini verir
mysql_list_fields :	Sonuçtaki tüm tablo alanlarını listeler
mysql_list_dbs :	Sunucudaki tüm veritabanlarını listeler
mysql_list_tables :	Veritabanındaki tüm tabloları listeler
mysql_num_fields :	Sonuçtaki alan sayısını verir
mysql_num_rows :	Sonuçtaki satır sayısını verir
mysql_pconnect :	Sunucuya kalıcı bir bağlantı tanımlar
mysql_query :	Veritabanına sorgu gönderir
mysql_result :	Sorgudan dönen sonuçları alır
mysql_select_db :	Sunucudan veritabanı seçer
mysql_tablename :	Verilen alanın ait olduğu tablo adını verir