DERS NOTLARI

Dr. Hakkı ÖCAL

Ders Sorumlusu: Tolga Güyer

22.02.2008

PHP

Önsöz

Rasmus Lerdorf, tarihin ilk dönemlerinde yani Internet'in henüz yayıldığı 1990'ların ortalarına doğru iş arıyordu; hayat öyküsünü bir kişisel sayfada yayınlamak ve başvuracağı yerlere bu sayfanın adresini verebileceğini düşündü. Fakat o dönemde özellikle üniversitelerin Unix ağlarında kurulan Web Sunucularda kişisel sayfa yapmak kolay değildi. Rasmus, kendisi için hazırladığı yazılımın Web'e aşina olmayanlar starafından da kolayca kullanabileceğini düşündü. Bu yazılımın büyükçe bir bölümü Perl dilinden alınmıştı. Adına Personal Home Page (Kişisel Ana Sayfa) dediği bu programın çok tutulması üzerine Rasmus Lerdorf, Internet Sitesi kuran herkesin birinci gün değilse bile ikinci gün farkına vardığı ihtiyacı, yani bir form yoluyla ziyaretçiden gelen bilgileri işlemeyi sağlayan ekleri yazdı ve programın adı PHP/FI (Form Interpreter/Form Yorumlayıcı) oldu. Kimileri programın bu sürümüne PHP2 adını taktı; ve bu ad, programın çok değişmesine ve gelişmesine rağmen uzun süre değişmeden kaldı.

Rasmus Lerdorf, 1995'in ortalarında, Zeev Suraski, Stig Bakken, Shane Caraveo ve Jim Winstead ile bir grup kurdu ve PHP'yi Perl'den ödünç alma rutinlerle iş yapan bir paket olmaktan çıkartıp, Nesne-Yönelimli (Object-Oriented) bir programlama dili haline getirdi. Bu grup, bugün PHP4 adı verilen bu dili geliştirmekte ve yeni işlevsellikler kazanması için sürekli çalışmaktadır. PHP, Linux gibi, Açık Sistem kurallarına tabidir; isteyen programda istediği değişikliği yapabilir; ancak bu değişiklikten para kazanamaz ve yaptığı değişiklikleri isteyen herkese açıklamak zorundadır.

Linux'un önümüzdeki yıllarda en çok kullanılan işletim sistemi olacağı yolundaki bilgiler kuşkusuz en çok Linus Torvalds'ı şaşırttı. Linux'un bu başarısındaki en önemli anahtar nokta olan açık kaynak geliştiriliyor ve dünyanın heryerinden yazılım geliştiricilerin desteğini alıyor olması. Aynı başarıyı Web yazılım geliştirme dili PHP de gösteriyor. PHP bundan iki yıl öncesine kadar adı duyulmamış basit bir dildi. Ancak şimdi gerek performans

gerek destek açısından diğer Web yazılım geliştirme dilleri ile kolayca yarışıyor; hatta bunları fazlasıyla geçiyor. Hemen her türlü işletim sistemiyle çalışıyor olması da cabası. PHP, her türlü veritabanıyla sorunsuz çalışıyor olmasıyla ve Zend derleyicisiyle daha da yaygınlık ve güç kazanacak. PHP ve açık kaynak olarak geliştirilmeye başlanan MySQL'in birlikte kullanıldıklarında yakaladıkları etkinlik binlerce Dolar verilerek alınan veritabanı ve uygulama dilleriyle yarışmanın ötesinde farklara sahip.

Türkiye'de PHP oldukça yeni kullanılmaya başlandı. Dünyadaki PHP kullanıcılarının tercih ettikleri PHP editörü PHPed'i bir Türk'ün geliştirmiş olması, hepimize haklı bir gurur veriyor. Ayrıca http://www.php.org.tr adresindeki oluşum çok kısa sürede Türkiye'de PHP'nin layık olduğu yeri alacağını gösteriyor.

Bu kitapçığı PHP'nin 4'ncü sürümüne göre yazdık. Burada verdiğimiz örnekleri uyguladığınızda beklenen sonucu alamazsanız, önce sizin sisteminizde kurulu PHP'nin sürümünü kontrol etmelisiniz. Bu kitapçıktaki örnek kodları PC Life dergisinin Internet sitesinde, http://www.pclife.com.tr/...../php_kodlar.zip adresinde bulabilirsiniz. Ayrıca bu kitapçıktaki kodların tümünün gerçek bir Web sunucuda nasıl işlediğini görmek için http://www.mycgiserver.com/~ocal/ adresine bakabilirsiniz.

Bununla birlikte bir programlama dilinin incelikleri, ancak kodlarınızı kendiniz yazarsanız öğrenilebilir. Örnek kodlardan, ancak kendi yazdığınız kodlar beklediğiniz sonucu vermediği taktirde, karşılaştırma amacıyla yararlanmalısınız. Sizi biraz da buna zorlamak için yer yer konuları bu kodların yazıldığı varsayımı ile ele alacağım.

Bugüne kadar PHP programlarrına .php3 uzatmasını vermek adeta gelenek halini almıştı; ancak PHP'nin 4'ncü sürümüyle birlikte, konfigürasyon ve INF dosyaları, yapılandırma sırasında işletim sistemlerine ve Web sunucularına varsayılan değer olarak sadece .php uzatmasını tanımasını bildiriyorlar. PHP programlarınıza bundan başka uzatmalar vereceksenız, Linux'ta Apache konfigürasyon dosyasını, Windows'da ise Registry kayıtlarını değiştirmeniz gerekir.

İki bölüm olarak sunacağımız PHP kitapçığının birinci bölümünün yazılmasında yurdumuzda PHP kullanıcılarını eğitme ve destekleme hareketinin öncüleri Sayın Kayra Otaner, Serdar Soydemir ve Özgür Akan, değerli yardımlarını esirgemediler. İkinci kitapçıkta yer alacak uygulamaların tümünü bu grup geliştirdi ve Açık Kaynak hareketinin gerçek temsilcileri olduklarını göstererek, serbestçe kullanılmak üzere okuyucularımıza sundular. Kendilerine sonsuz teşekkürlerimizi sunarız. Bununla birlikte bu kitapçıkta varolan hataların tümü bana aittir.

PHP Desteği veren sitelerle ilgili not:

Bu kitapçıkta bir kaç sayfa ilerledikten sonra, yapmak isteyeceğiniz ilk işlerden birinin ücretli veya ücretsiz PHP desteği veren bir Wes sitesi evsahibi (Hosting) firması aramak olacağını tahmin ediyorum. Bu amaçla aramalarınıza http://www.pageresource.com/putweb/index.html adresinden başlamanızı salık veririm. Eğer doğruca böyle bir evsahibi firmaya ulaşmak isterseniz http://www.mycgiserver.com adresine gidebilirsiniz. Kitapçığın sonunda çeşitli PHP kaynaklarının bir listesini de bulabilirsiniz.

PHP'ye Giriş

C/C++, Pascal, hatta Visual Basic bilen kişiler için PHP öğrenmek hemen hemen bir öğleden sonrayı ayırmak suretiyle yapılabilecek bir iştir. Bu dillere aşina olmayanlar ise belki bir kaç öğleden sonralarını ayırmak zorunda kalabilirler! PHP, bir Script dilidir; yani kodları düz yazı dosyaları halinde kaydedilir ve kullanılacağı ortamda bir yorumlayıcı tarafından yorumlanır. Bu, PHP ile yazacağınız programları, derlemek yani ortaya bir EXE veya çalıştırılabilir başka bir dosya çıkartmaya gerek olmadığı anlamına gelir. Fakat PHP Script'lerinizi çalıştırabilmek için bu dili bilen bir programa ihtiyacınız vardır. Bu programın tek başına (komut istemci penceresinde veya terminal ekranında) çalışması mümkün olduğu gibi, Web Server tarafından da çalıştırılabilir olması gerekir. Başka bir deyişle Web Sunucu programının PHP anlar hale getirilmesini sağlamak zorundayız.

PHP yorumcusu, bugün Zend-çekirdeği adı verilen bir Scripting teknolojine dayalı olarak, Zeev Suraski ile Andi Gutmans tarafından sıfırdan, tamamen yeniden yazılmış bulunuyor. Zend, tıpkı Windows Script Host ve mesela Visual Basic for Applications (VBA) gibi, işletim sistemine kendisi ile uyumlu Script dilleriyle yazılmış programları çalıştırma imkanı kazandıran bir teknolojidir (Zend konusunda ek bilgiyi Internet'te www.zend.com adresinde bulabilirsiniz.) Bugün bu dilin resmî adı "PHP: Hypertext Preprocessor" (Hiper-metin ön-işlemcisi) olarak değiştirildi, ve çeşitli Web Server'ların PHP dili anlaması için gerekli yorumlayıcıları yeniden üretlidi. 20 Ocak 200'de İsrail'de PHP'nin ilk uluslararası konferansı yapıldı ve 4'ncü sürümünün betası ortaya kondu.

Peki ama PHP nedir?

PHP ile artık oturup her türlü işlevselliği sahip programlar yazabilirsiniz; yani PHP başlı başına uygulama alanları bulabilecek düzeye ulaşmıştır. Başka bir deyişle PHP ile oyun programından tutun veritabanı yönetimine kadar hemen her türlü programı

yazabilirsiniz. Bizi bu kitapçıkta ilgilendiren tarafıyla PHP, Web Sunucu'ya bir takım işler yaptırmak için program yazma dilidir. Tıpkı CGI/Perl veya ASP ya da ColdFusion, Java (Javascript değil!) gibi! PHP kodları, oluşturacağımız HTML sayfalarında HTML etiketlerinin arasında kendi özel ayracı içinde yazılır:

```
<html>
<?PHP
      echo ("Merhaba Dünya!");
?>
</html>
```

İster kişisel bilgisayarımızda, ister gerçek Internet ortamında olsun, bir Web Sunucu, ziyaretçinin (Internet istemci programının, Browser'ın) talep ettiği dosya adının uzatması olarak ".PHP," ".PHP2," ".PHP3," veya ".PHP4" görünce, kendine kendine, "Yine bana iş çıktı!" diye düşünür! Çünkü Web Server, uzatması ".htm" veya ".html" olan bir dosyayı, kendi sabit disklerinden birinde bulur ve zahmetsizce, ziyaretçinin bilgisayarına gönderir. Fakat, ".asp," .pl," ".cfm" gibi, ".PHP" uzatması, Web Sunucu'ya bu sayfayı olduğu gibi, alıp ziyaretçinin Browser'ına göndermek yerine, önce PHP yorumlayıcıyı çağırarak, ona teslim etmesi gerektiğini bildirir. PHP yorumlayıcısı (yani Windows ortamında PHP.exe veya 4'ncü sürümle gelen PHP4isapi.dll, Unix ve türevlerinde çalıştırılabilir PHP dosyası), kendisine teslim edilen bu belgede önce "<?PHP" ve "?>" ayraçları arasındaki PHP kodlarını seçerek alır ve gereği ne ise onu yapar. Bu ayraçların içinde kalan kodlar, bizim yapılmasını istediğimiz işlemin komutlarıdır. HTML ile yetinmeyip, PHP kullanmamızın sebebi, bu komutların HTML'in yapayacağı şeyleri yapmasıdır.

HTML'in yapamayacağı şeylerin başında, Web Sunucusu'nda yapılacak işler vardır. Sözgelimi HTML etiketlerini kullanarak, Web Sunucusu'ndaki bir veritabanı dosyasını açıp, içindeki verileri okuyamayız. HTML yoluyla Web Sunucusu'nun sabit disklerindeki düz yazı dosyalarını da okuyamayız ve bu disklere dosya yazamayız. HTML etiketleri ile Web ziyaretçimizden bilgi edinebilir ve bunları Sunucu'ya yollayabiliriz; ama o kadar! Bu bilgileri işe yarar şekle sokmak için bize Server'da çalışan bir program gerekir. Server'da çalışan

program, EXE ve DLL veya JSP (Java Server Pages) gibi bir "gerçek program"; veya Perl, ASP ve PHP gibi Script diliyle yazılmış bir program olabilir. (Gerçek programları yazanlar, tabiî, sizin Script diliyle yazılmış programlara "program" demenize çok kızarlar! Ama bu Script pogramlarının program olmasını hiç engellemez.)

Ne diyorduk? Web Sunucu, ".PHP" uzatmasını görünce telaşa kapılır, bu dosyayı PHP yorumlayıcısına verir ve o da "<?PHP" ve "?>" ayraçları içindeki kodları icra eder, demiştik. Bu icraatın sonunda ortaya çıkan ürün eğer ziyaretçiye gönderilecekse, HTML etiketleri içinde gönderilir; yani ziyaretçi asla PHP kodlarını göremez. Kimi zaman PHP programımızın çalışması sonucu ortaya ziyaretçiye gönderilecek bir ürün çıkmaz; elde edilen sonuç ya başka bir programa (örneğin elektronik ileti gönderen bir Sendmail progamına) hammadde olarak verilir, ya da Server'da sabit diske yazılır. Fakat her durumda, Web ziyaretçimiz bizim PHP kodlarımızı göremez; bu Web sayfalarımızda PHP kodu kullanmamız halinde, sayfalarımızın Browser tarafından tanınması veya tanınmaması gibi bir durumun ortaya çıkmaması demektir. Ziyaretçi ne tür Browser kullanırsa kullansın, kendisine Sunucu'dan daima saf HTML kodu geleceği için, sayfalarımızı rahatça görebilecektir. Tabiî bir nokta var: Sayfalarımızda herhangi bir Browser'ın arızalı yorumladığı veya hiç yorumlayamadığı HTML etiketleri bulunabilir; ya da söz gelimi ziyaretçiye giden HTML'in içinde Browser'da çalıştırılacak olan başka Script kodları bulunabilir; ve Browser bu kodları anlamayabilir. Örneğin, HTML sayfanızda VBScript kodları kullanmış iseniz, Netscape ile sitenize bağlanmış olan kişiler bu kodun oluşturmasını istediğiniz etkiyi göremeyeceklerdir; çünkü Netscape VBScript anlamaz! PHP ile yazacağınız Web programlarının (eski deyimiyle "sayfaların") oluşturacağı HTML belgesinin Browser türleri ve sürümleri ile uyarlılığını sağlamak yine Web programcısı olarak sizin sorumluluğunuzdadır.

Özetlersek, PHP bir CGI programlama dilidir; bu dille, Web Sunucusu ile Web ziyaretçisi arasındaki buluşma noktası olan CGI'da bilgi alışverişi yapılabilir; Sunucu'da

bulunan başka programlar çalıştırılabilir ve böylece Web sayfalarımıza HTML'in sınırlamalarının ötesinde hareket ve ziyaretçi ile etkileşme olanağı kazandırılabilir.

Sanırım iyi bir Web programcısı olarak şu anda ekrana "Merhaba Dünya!" yazdırmak için sabırsızlanıyorsunuz. Ama bunun için, PHP sayfanızı ya PHP-uyumlu bir Web Sunucusu'ndaki Web sitenize yüklemek, ya da kişisel bilgisayarınızdaki kişişel Web Sunucusu'na PHP öğretmek zorundasınız. Bu ikinci yöntemi kuvvetle tavsiye ederim; çünkü yazacağınız PHP programlarını gerçek Internet Sunucusunda sınamak, bazen arzu edilmeyen durumlara yol açabilir. Oysa kişinin bu denemeleri, evinde ve işyerinde kendi bilgisayarında yapması, çoğu zaman daha az başağrısı ve gözyaşına yol açar! En azından, PHP hatalarınızı ve gözyaşlarınızı sizden başka gören olmaz!

PHP ve Veritabanı

Kişisel bilgisayarınızdaki kişisel Web sunucusuna PHP öğretme işlemine geçmeden önce, hemen hemen PHP'nin ikiz kardeşi sayılan MySQL'den ve onun rakibi PostgreSQL'den kısaca söz edelim.

PHP'nin varlık sebeplerinin başında, ticarî Unix ve Microsoft Windows işletim sistemlerinden ayrı, ticarî olmayan Açık Sistem ortamlarında çalışan bir Script dili ile bu dille kullanılabilecek yine Açık Sistem ürünü bir veritabanı erişim ve yönetim aracına sahip olmak gelir. Perl, gerçi PHP'den çok önce Açık Sistem ürünü olarak, isteyen Web Programcısı'nın ücretsiz edineceği bir CGI programlama dili olarak Web'in ilk gününden beri Web Programcısının emrinde idi. Ama Perl, metin dosyalarından yazı çekmek ve bunları biçimlendirerek rapor haline getirmek amacıyla geliştirilen bir dil olduğu için, özellikle veritabanına dayalı işlemlerde programcıya çok güçlük çıkartıyordu. ASP ise (ASP desteği sağlayan ve ücretsiz Web sitesi veren firmaların giderek artmasına ve veritabanı konusunda hem kolaylık, hem de büyük bir etkinlik sağlamasına rağmen) çoğunlukla ücretli Web sitesi aldığımız ticarî amaçlı evsahibi (Hosting) firmalarının sağladığı bir teknoloji olarak görüldü. PHP, tasarımcılarının MySQL ve PostgreSQL teknolojilerini

geliştiren kurum ve firmalarla yaptığı yakın işbirliği sonucu ikinci sürümünden itibaren veri-yönlendirmeli Web uygulaması alanında önemli bir araç olarak belirdi. Dolayısıyla, PHP kurulumu dendiği zaman, kendiliğinden, MySQL kurulumu da kastedilmiş olur. Şimdi PostgreSQL da ekleyebiliriz.

PHP, bir dildir; MySQL ve PostgreSQL ise bir demet sürücüdür. PHP ile, Web Server'a sayfanızı Internet ziyaretçisine göndermeden önce bir takım komutları icra etmesini, bir takım veri dosyalarını açıp içindeki bilgileri alıp bunları HTML kodlarıyla bezeyip, Browser'a bildiğimiz klasik HTML sayfası olarak göndermesi talimatını veririz. MySQL ve PostgreSQL ise, Windows sistemlerinde Denetim Masası'nda gördüğünüz ODBC Veri Kaynağı Yönetmeni adlı araç gibi, bir vertabanı sunucusu, yani Database Server pprogramlarıdır. Yakın zamana kadar bu iki veritabanı yönetim aracı arasında fazla bir çekişme yoktu. Ancak şimdi özellikle Linux kullanıcıları topluluklarında iki program arasında verimlilik testleri yapılıyor; hangisinin daha iyi olduğnuna ilişkin yoğun bir tartışma sürüyor. PHPBuilder.com uzmanları, yaptıkları bir çok denemede, MySQL'in daha hızlı ve daha etkin çalıştıkları sonucuna varıyorlar. (Bu konuda daha geniş bilgi edinmek için Internet'te www://PHPbuilder.com/columns/tim20000705.PHP3 adresine bakabilirsiniz.)

MySQL, bir kullanıcı lisansı 200 Dolar olan ticarî bir program iken 2000 yılı ortalarında GPL (GNU General Public License/GNU Kamusal Lisans) uygulamasına geçmiş ve ücretsiz dağtılır hale gelmiş bulunuyor. Buna göre MySQL'i bir ticarî uygulamanın içinde motor olarak kullanacaksanız üreteci firmaya lisans ücreti ödemeniz gerekir; bunun dışındaki uygulamalar için programı Internet'ten indirerek kullanabilirsiniz. PHP için MySQL'e gerek yoktur. PHP, bir NT veya Win9x tabanlı Web Server'da çalıştırılıyorsa, Microsoft'un ODBC sürücüleri ile çalışabilir; ve ODBC'nin okuduğu bütün veritabanlarından veri çekebilir. (ODBC'nin Unix-Linux sürümü de vardır.) Ayrıca PHP için Adabas, dBase, Empress, FilePro, Informix, InterBase, mSQL, Oracle, Solid, Sybase, Velocis ve bir çok Unix veritabanı yöneticisi için geliştirilmiş add-on (sonradan eklenen) modüller vardır. PHP,

bir çok LDAP istemci programı için yazılmış API'lere de sahiptir. PHP ile IMAP, SMTP gibi Internet elektronik mektup protokollerini kullanmak da mümkündür.

Biz bu kitapçıkta örneklerimizde ve uygulamalarımızıda MySQ'den yararlanacağız. PHP-MySQL ikilisini kişisel bilgisayarımıza kurmak için önce gerekli programları Internet'ten indirmemiz gerekir. Bunun için şu adreslerden yararlanabilirsiniz:

http://www.PHP.net/downloads.PHP

http://www.mysql.com/downloads/index.html

PHP ve Web Sunucusu

Bizim açımızdan PHP, bir Web sunucu programıdır; kurulacağı sistemdeki Web sunucusu ile uyumlu olması gerekir. 4'ncü sürümü itibariyle, PHP bütün Unix-türevi sistemler (örneğin Linux) ve Microsoft Windows sistemleri ile uyumludur. Birlikte çalışabileceği Web sunucu programları arasında Apache, IIS, FHTTP, Omni HTTPd, Xitami ve Windows 95/98 için PWS (Personal Web Server) vardır. Gerçek Internet ortamında sunucu programlara PHP yeteneği kazandırma işlemi, Web Pnogramcısı olarak bizi ilgilendirmez; bizim için Internet sitemize evsahipliği yapan Hosting firmasının PHP desteği verip vermediğini bilmek yeter. Fakat tasarımlarımızı sınayabilmek için kendi bilgisayarımıza bir kişisel Web Server kurmak ve bu programı PHP-uyumlu hale getirmek gerekir. Burada sadece iki örnek ele alacağız: Linux için Apache sunucu programı ve MS-Windows için PWS. Bu iki programdan hangisi sizin sisteminize uygunsa onu kendi bilgisayarınıza kuracağınızı varsayarak, şimdi PHP ve MySQL kurulumu ile uğraşabiliriz.

Unix/Linux Sistemleri

Kurma işlemine Unix ortamından başlayalım. Burada Unix sistemi dediğimiz zaman Unix'in tekrar bilişim haritasında yerini almasını sağlayan Linux'u kastediyorum. Bunun için önce http://www.PHP.net/ adresini ziyaret etmemiz gerekiyor. Eğer PHP uzmanı olacaksanız, burası sizin ikinci eviniz olabilir. Downloads köprüsünü tıklayarak açılacak sayfada Unix/Linux için kaynak kodunu bilgisayarınıza aktarın. PHP'yi kuracağınız

bilgisayarda bir Web sunucu programı yoksa, http://www.apache.org/dist/ adresinden apache_1.3.12.tar.gz (veya daha ileri) dosyasını da indirmeniz gerekir. Unix-Linux sistemleri için bilgisayarınızda "gcc C derleyici" bulunması gerekir. Linux sisteminizde örneğin Red Hat Linux sürüm 5.2 gibi eski bir gcc derleyici varsa, gcc sürüm 2.8.1'i Internet'ten bulmanız gerekir.

Linux'ta program derlediniz mi?

Daha önce hiç Linux programı derlemediyseniz, önce bazı kavramları açıklığa kavuşturalım:

tar: (tape archiver) birden fazla dosyayı tek dosya haline getirir ve sıkıştırır. Teyplere yedekleme amacıyla geliştirilmiştir. tar dosyaları sıkıştırılmışsa uzatması .gz, değilse .tar olur.

gcc: GNU'nun C derleyicisidir. İnsan tarafından okunabilen kaynak kodlarını makina tarafından okunabilen nesne (object) dosyalarına dönüştürür. C programlama diliyle yazılmış kaynak kodları içeren dosyaların uzantını .c olur. Nesne dosyalarının adının uzantısı genellikle .o olur. Derleyici başlık (header, uzantısı .h olan) dosyalarını bulamazsa derleme hatası verir.

make: henüz derlenmemiş kaynak dosyalarını derleyen yardımcı bir programdır. make, derleme talimatlarını Makefile adlı bir dosyadan alır.

ld: GNU'nun bağlayıcı (linker) programıdır. Nesne dosyalarını ve kitaplık denen dosyaları birbirine bağlayarak ortaya çalıştırılabilir bir program dosyası çıkartır. Genellikle Makefile'ın içinde linker'ın çalıştırılması talimatı bulunur ve sizin bu programı kullanmanız gerekmez.

ldconfig : paylaşılan kitaplık dosyalarını (shared librari) arar. Bu dosyalardan birden fazla program tarafından paylaşılır, ve dosya adlarında .so bulunur.

./configure : C derleyecisi ve header dosyalarının nerede olduğunu araştırır, ve vardığı sonuca göre Makefiles dosyasında değişiklik yapar.

PHP, bir Unix/Linux sisteminde müstakil yorumlayıcı veya Apache Web sunucusunun bir modülü olarak çalışmak üzere derlenebilir. PHP, müstakil yorumlayıcı olarak kurulduğunda, Web sunucusu, her ihtiyaç halinde PHP yorumlayıcısının bir örneğini çalıştırır. Bu, bir süre sonra Web sunucusunun bulunduğu işletim sisteminin yükünün artmasına ve işleyişinde yavaşlamaya yol açabilir. Oysa PHP, sunucunun bir modülü olarak kurulduğunda, bu sakınca ortaya çıkmaz. (Bu özelliği ile PHP, klasik CGI/Perl'ün çalışma tarzından ayrılır, ASP'nin ve ColdFusion'ın çalışma tarzına yaklaşır.) Bununla birlikte kötü niyetle veya beceriksizce yazılmış PHP programları modül olarak kurulmuş PHP'yi çökertirlerse, Web sunucusunu da çökertmiş olurlar; oysa müstakil PHP yorumlayıcı çökerse, Web sunucusuna bir şey olmaz.

Burada yapacağımız kurulum işlemi sırasında Linux kurulu olan bilgisayarımıza Apache Web sunucu programını, bu sunucunun kullanacağı güvenli Socket katmanını sağlayacak Mod_SSL programını, bu programın gerektirdiği OpenSSL aracını, MySQL veritabanı sunucusunu ve PHP Scripting dilini kurmuş olacağız.

PHP'yi Linux ortamında Apache Web sunucusuna modül olarak kurmak için sırasıyla şu işlemleri yapmamız gerekir (Sisteminizde gzip veya gunzip ile gcc and GNU make kurulu olduğunu varsayıyorum):

1. Sisteminizi root olarak başlatın ve sıkıştırılmış olarak alacağınız PHP ve Apache dosyalarını geniş yer olan geçici bir dizinde açın, ve o dizine gidin. Sözgelimi, Apache'nin tar-zip olarak sıkıştırılmış kaynak dosyalarının tmp adlı dizinde olduğunu varsayalım:

```
# cd /tmp
# gunzip –c apache_1.3.12.tar.gz  | tar xf –
```

Burada, indirdiğiniz Apache sürümü farklı ise onun adını yazacaksınız. Şimdi Apache'yi açtığınız klasöre gidin ve buradaki configure dosyasını çalıştırın:

```
# cd apache_1.3.12
# ./configure --prefix=/usr/local/apache
```

Buradaki örnekte Apache'yi /usr/local/apache dizinine koyduğumuza dikkat edin. Bu dizini, bir çok kurulumun varsayım olarak arayacağı dizin olduğu için tercih ediyoruz. Ama isterseniz siz Apache'yi başka dizine de kurabilirsiniz.

2. Sıra MySQL veritabanı sunucusunda. Linux'ta root oturumunda iken sıkıştırılmış dosyaları indirdiğiniz geçici dizine geçin ve dosyaları açın; sonra oluşturulan dizine giderek, configure komutu ile veritabanı sürücüsünü yapılandırın:

```
# cd /tmp
# gunzip -dc mysql-3.22.32.tar.gz | tar xvf -
# cd mysql-3.22.32
# configure --prefix=/usr/local/mysql
```

Şimdi make komutu ile binary dosyalarını oluşturarak, ve sisteme yükleyelim:

```
# make
# make install
```

MySQL'in kullanım izinlerini belirlememiz gerekir. Aşağıdaki komutları yazarken "parola" kelimesi yerine vermek istediğiniz parola kelimesini yazmanız gerekir:

```
# scripts/mysql_install_db
# cd /usr/local/mysql/bin
# ./safe_mysqld &
# ./mysqladmin -u root password 'parola'
```

Kurulum işleminin başarılı olup olmadığını sınamak için, kurulum sırasında oluşturlan örnek veritabanı dosyalarını bulunup bulunmadığını soruşturabilir ve kendimiz yeni bir veritabanı oluşturabiliriz. Bunun için şu komutu yazalım:

```
# BINDIR/mysqlshow -p
# mysql -u root -p
```

Sistem "Enter password:" karşılığını verdiğinde, yukarıda kullanım izinlerini belirlerken yazdığınız parolayı yazın; MySQL sunucusu çalışacak ve size kullanıcı izinlerinin bulunduğu "mysql" ve sınama amacıyla oluşturulan "test" adlı iki veritabanının bulunduğunu bildirecektir. Şimdi kendi veritabanımızı oluşturalım. MySQL sunucu komut

istemcisi satırına şunu yazın (Bilgisayarın vereceği karşılıklardan ayırt etmek için kendi yazdıklarımızı koyu renkle gösteriyoruz):

```
mysql> create database deneme;
```

Sunucu "Query OK, 1 row affected (0.00 sec)" şeklinde karşılık verecektir. Şimdi bu veritabanını seçerek içinde deneme adlı ve içinde "no" ve "isim" adlı iki alan bulunan bir tablo oluşturalım (MySQL komutları büyük harf-küçük harf ayrımı yapmaz; komutlardan sonra noktalı virgül konur). Burada koyu renkle gösterilen satırlar ve işaretler bize aittir:

```
mysql> use deneme;
Database changed
mysql> CREATE TABLE books (
-> no int(3) not null auto_increment,
-> isim char(50) not null,
-> unique(no),
-> primary key(no)
-> );
Query OK, 0 rows affected (0.00 sec)
```

Bakalım tablomuz başarılı şekilde oluşmuş mu?

```
mysql> show tables;
mysql> describe deneme;
Field Type Null Key Default Extra
no int(3) PRI 0 auto_increment
isim char(50)
2 rows in set (0.00 sec)
```

Çok güzel! Hem MySQL'i kurmuş, hem de ilk tablomuzu oluşturmuş bulunuyoruz. MySQL komutlarını daha ayrıntılı öğrenebilmek için MySQL'in http://www.mysql.com adresindeki sitesini ziyaret edebilirsiniz. Şimdilik MySQL'den çıkalım ve kuruluma devam edelim.

3. Sıra geldi PHP'ye! PHP kaynak dosyasının bulunduğu geçici dizine geçin, ve dosyaları açın; sonra açılan dosyaların bulunduğu dizine gidin:

```
# gunzip -c PHP-4.0.1pl2.tar.gz | tar xf -
# cd PHP-4.0.1pl2
```

Şimdi configure dosyasını çalıştıracağız; bu komutun bir çok opsiyonu vardır; bunların tümünü görmek istiyorsanız, "configure --help" komutunu verebilirsiniz. Biz sadece PHP'ye MySQL ve Apache opsiyonlarını eklemekle yetineceğiz:

```
# ./configure --with-mysql=/usr/local/mysql \
--with-xml \
--with-apache=../apache_1.3.12 \
--enable-track-vars
```

Oluşan binary dosyalarını program dosyasına çevirerek, yükleyelim:

```
# make
# make install
```

Oluşan ini dosyasını lib dizinine koplayalım:

```
# cp PHP.ini-dist /usr/local/lib/PHP.ini
```

PHP.ini dosyasındaki bir çok özelliği değiştirerek, PHP'nin çalışma tarzını ayarlayabilirsiniz; örneğin şu satırı ekleyerek, PHP'nin azamî çalışma süresini arttırabilirsiniz:

```
max_execution_time = 60;
```

Güvenli socket ve Internet Erişimi Katmanı

Kuracağınız Web sunucusu gerçek Internet'e açılacaksa, sisteminize Mod_SSL yeteneğini kazandırmanız gerekir. Bu sistemin rsaref-2.0 adında bir eki vardır; ancak Amerika dışında kurulan Web Sunucularda rsaref-2.0 ekinin kurulması gerekmez. Apache sunucusunu sadece PHP dosyalarınızı sınamak amacıyla kendi bilgisayarınıza kuruyorsanız, Mod_SSL kurmak şart değildir.

Bu işlemi yapmak zorunda iseniz, önce http://www.openssl.org/ adresinden openssl-0.9.1c.tar.gz adlı dosyayı, http://www.modssl.org/ adresinden de mod_ssl-2.6.5-1.3.12.tar.gz dosyasını indirin. Sonra sırasıyla şu işlemleri yapabilirsiniz:

```
# gunzip -dc openssl-0.9.5c.tar.gz | tar xvf -
# cd openssl-0.9.x
#./config --prefix=/usr/local/ssl \
-L`pwd`/../rsaref-2.0/local/rsaref -fPIC
```

```
# make
# make test
# make install
# cd ..
```

Şimdi Mod_SSL modülünü Apache ile kullanılacak şekilde kuralım:

```
# gunzip -dc mod_ssl-2.6.4-1.3.12.tar.gz |tar xvf -
# cd
# ./configure --with-apache=../apache_1.3.12
# cd ..
# cd apache_1.3.12
# SSL_BASE=../openssl-0.9.x \
RSA_BASE=../rsaref-2.0/local \
./configure \
--enable-module=ssl \
--activate-module=src/modules/PHP4/libPHP4.a \
--enable-module=PHP4 \
--prefix=/usr/local/apache \
--enable-shared=ssl
[...ISS'ler buraya arzu ettikleri diğer opsiyonları ekleyebilirler..]
# make
```

Bu noktada sistem binary dosyaların oluşturulduğunu ve kurulumun gerekli güvenlik sertifikalarını istediğini bildirecektir. Sertifika oluşturma konusunda fazla bilgi için, http://www.modssl.org/docs/2.6/ssl_faq.html#ToC25 adresine bakabilirsiniz.) Diyelim ki sadece ISS'in yeri, firma adı ve bir iki diğer bilgiden oluşan özel bir sertifika oluşturmakla yetineceksiniz. Bunun için şu komutu girin ve sistemin soracağı sorulara cevap verin:

```
# make certificate TYPE=custom
```

Şimdi Apache'yi yeniden kurabiliriz:

```
# make install
```

Herşey yolunda gittiyse, sistem "You now have successfully built and installed the Apache 1.3 HTTP server," şeklinde başlayan uzun bir mesaj verecektir. Bu durumda Apache sunucusu şu komutla başlatılabilir:

```
/usr/local/apache/bin/apachectl start
```

Apache'yi güvenli iletişim katmanı desteği ile başlatmak istiyorsanız şu komutu yazın:

```
/usr/local/apache/bin/apachectl startssl
```

Bu işlemleri başarıyla bitirmiş olsak bile sistemin çalıştığını kontrol etmemiz gerekir. Önce Apache'nin çalıştığından emin olalım. Bunun için önce Apache'nin yapılandırılmasına PHP'nun eklendiğinden emin olmamız gerekir. Kurulumu buradaki gibi yaptı iseniz /usr/local/apache/conf dizininde bulunan httpd.conf dosyasını açın; ve "addtype for PHP4" satırların önündeki yorum (#) işaretlerini kaldırın; yani bu satır yorum olmaktan çıkıp yapılandırma komutu olsun:

```
    #AddType application/x-httpd-PHP .PHP
    #AddType application/x-httpd-PHP-source .PHPs
```

şeklindeki satırları şu şekle getirin:

```
    AddType application/x-httpd-PHP .PHP
    AddType application/x-httpd-PHP-source .PHPs
```

Şimdi Apache sunucusunu çalıştırabilir ve PHP desteğinin bulunup bulunmadığını sınayabilirsiniz:

```
# cd /usr/local/apache/bin
# ./apachectl configtest
Syntax OK
# ./apachectl start
./apachectl start: httpd started
```

"httpd started" ifadesi bize Apache'nin çalıştığını gösteriyor. Şimdi Browser'ınızı açın ve HTTP sunucusu olarak Apache'nin adresini yazın. Bu, bilgisayarınızın Domain adıdır; Apache'nin kurulu olduğu bilgisayarın IP numarasını da yazabilirsiniz.

Şimdi, bir düzyazı dosyasına şunları yazın:

```
<? PHPinfo(); ?>
```

Dosyayı PHP.PHP adıyla Apache'nin Web sunucu için kök dizin saydığı klasöre kaydedin. Browser'ınızın URL adresi alanına sunucu adıyla birlikte dosyanın adını yazın.

Aldığınız sonuç, PHP'nin sistem ve sunucu hakkında ayrıntılı raporu ise, kendi kendinizi kutlayabilirsiniz: hem Apache sunucu, hem de PHP çalışıyor!

Windows-PWS ve IIS

PHP'yi Windows 95/98/NT/2000 işletim sistemlerinden biri ile çalışan bilgisayara kurmak için önce http://www.PHP.org adresinden PHP-4.0.1pl2-Win32.zip veya daha ileri sürümdeki zip dosyasını almamız gerekir. Bu dosyayı açın ve içindekileri sözgelimi c:\PHP dizinine kopyalayın. (Dizini oluştururken "PHP," "PHP" veya "PHP" yazdığınıza dikkat edin!)

Daha önce Wihdows sistemine PHP 3.x kurmuş olanlar PHP 4.0'ün kurulumumu biraz farklı bulacaklardır. Bu, daha önce PHP kişisel ve Internet amaçlı Web sunucuda (PWS veya IIS) müstakil bir CGI programı olarak çalışırken şimdi buna ek olarak (tıpkı ASP gibi) DLL yoluyla, bir ISAPI modülü olarak çalışmasının sağlanmış olmasından kaynaklanıyor.

PHP'yi Windows 95 veya NT4.0 sisteminde kurulu bir PWS veya IIS için kuracaksınız, önce bazı önlemler almanız gerekiyor. NT 4.0 sistemlerinde IIS 4.0'ün bulunduğu Option Pack'in kurulması şart. Windows 95 sistemlerinde ise önce DCOM bileşenlerinin günceleştirilmesi gerekiyor. Bu dosyayı, http://download.microsoft.com/msdownload/dcom/95/x86/en/dcom95.exe adresinden alabilirsiniz.

PHP'yi müstakil çalışan CGI programı olarak kullanmayı düşünseniz bile daha önceki sürümlerinden farklı olarak bazı DLL dosyalarının Windows dizininde sistem alt-dizinine kopyalanması gerekir. Açtığınız Zip dosyasından çıkan ve C:\PHP dizinine kopyaladığınız dosyalar arasında bulunan MSVCRT.DLL ve PHP4TS.DLL adlı dosyaları Windows 9x sistemlerinde WINDOWS dizininde SYSTEM, NT/2000'de ise WINNT dzininde SYSTEM32 dizinine kopya edin. (Windows, MSVCRT.DLL dosyasının kullanılmakta olduğu bildirerek, kopyalamayı reddederse, kopyalamaya gerek yok demektir.)

IIS 4.0 VE 5.0

Windows NT/2000'e kurulu olan IIS 4.0 veya 5.0'e PHP desteği kazandırmak için, önce yukarıdaki gibi DLL dosyalarını gerekli yere kopyalayın. Açtığınız Zip dosyasından çıkmış olan PHP.ini-dist isimli düz yazı dosyasını Notepad ile açın ve "extension_dir=" satırını bularak, karşısındaki "./" yazısını silip yerine PHP dosyalarını kopyaladığınız dizinin adını (örneğin C:\PHP) yazın. PHP dizinin adını doğru yazdığınızı kontrol etmekte daima yarar vardır! Sonra ";browscap =" satırını bulun ve önündeki noktalı virgülü kaldırarak karşısındaki "extra/browscap.ini" kelimelerini c:\Winnt\browscap.ini olarak değiştirin. Bu dosyayı, PHP.ini adıyla Winnt dizinine kaydedin. (Windows dizininin Winnt adını taşıdığını varsayıyorum. Sizin Windows dizininiz başka ise, gerekli düzeltmeyi yapmanız gerekir.)

Browser Yetenek Dosyası: Browscap.ini

Windows sistemine kurulu bir Web Sunucu programına (PWS, IIS veya Apache Win32) PHP yeteneği kazandırabilmek için PHP'nin talep sahibi Browser'ın yeteneklerini bir dosyadan okuması gerekir. Bu dosyanın adı browscap.ini dir.

Internet'te http://asptracker.com/browscap.zip dosyasını alarak açın ve çıkacak "browscap.txt" dosyasının adını "browscap.ini" olarak değiştirerek Windows/Winnt klasörüne kopyalayın.

Internet Services Manager'ı açın; bu araç, NT 4.0'te Start menüsünde Programs/Option Pack'te, 2000'de ise Control Panel/Administrative Tools'dadır. Açılacak pencerede muhtemelen Default Web Server adını taşıyan Web sunucusunun adını sağ tıklayın.

Açılan listede Properties maddesini; açılan kutuda Home Directory sekmesini seçin; Configuration düğmesini tıklayın ve yeni bir Application Mappings satırı eklemek üzere, Executable kutusuna PHP4isapi.dll dosyasının tam yolunu ve adını (örneğin C:\PHP\

PHP4isapi.dll); Extension kutusuna ise .PHP yazın (IIS 4.0'te Method exclusions kutusunu boş bırakın) ve Script engine kutusuna işaret koyun.

ISAPI Filters sekmesini seçin. Add düğmesini tıklayarak yeni bir ISAPI filtresi kaydı yapın. Açılacak kutuda filtre adı olarak PHP, yol olarak da PHP4isapi.dll dosyasının bulunduğu dizinin yolunu (örneğin C:\PHP\ PHP4isapi.dll) yazın.

IIS'i tamamen durdurun; ama bunun için sadece Internet Services Manager'daki Stop düğmesini tıklamak yetmez; komut istemci (MSDOS Prompt) penceresinde net stop iisadmin yazın. Windows, bu servisin bağlı olduğu diğer bazı hizmetlerin de durdurulacağını bildirince "Y" yazın ve Enter'a basın. Şimdi IIS'i tekrar başlatmak için aynı pencerede net start w3svc yazın. Artık IIS'iniz PHP anlar hale gelmiş olmalı. Bunu, hemen sınayalım. Notepad veya bir başka düzyazı programıyla şunları yazın:

```
<? PHPinfo(); ?>
```

Dosyayı PHP.PHP adıyla IIS'in kök dizin saydığı klasöre kaydedin. Browser'ınızın URL adresi alanına sunucu adıyla birlikte dosyanın adını yazın.

Bravo! İşte yeni bir dil öğrenmiş Web sunucuya sahip oldunuz.

PWS 4.0

Windows'un 95/98/SE/Me türlerine kişisel Web sunucu programı olarak Personal Web Server (PWS 4.0) kurulmalıdır. PHP Zip dosyasından çıkacak DLL'leri Windows dizininde System klasörüne kopyaladıktan sonra, Zip dosyasından çıkmış olan PHP.ini-dist isimli düzyazı dosyasını Notepad ile açın ve "extension_dir=" satırını bularak, karşısındaki "./" yazısını silip yerine PHP dosyalarını kopyaladığınız dizinin adını (örneğin C:\PHP) yazın. PHP dizinin adını doğru yazdığınızı kontrol edin! Sonra ";browscap =" satırını bulun ve önündeki noktalı virgülü kaldırarak karşısındaki "extra/browscap.ini" kelimelerini c:\Windows\browscap.ini olarak değiştirin. Bu dosyayı, PHP.ini adıyla Windows dizinine kaydedin. Daha önceki sayfalarda göreceğiniz şekilde Browscap.ini dosyasının dosyasını Windows dizinine kopyalamış olmanız gerekir. (Windows dizininin Windows adını taşıdığını varsayıyorum. Sizin Windows dizininiz başka ise, gerekli düzeltmeyi yapabilirsiniz.)

Şimdi, açtığınız PHP Zip dosyasının içinden çıkan PWS-PHP4.reg adlı dosyayı bir düzyazı programıyla, örneğin Notepad ile açın. ".PHP"="[PUT PATH HERE]\\PHP4isapi.dll" yazılı satırı PHP dosyalarını koyduğunuz dizine göre, örneğin şöyle değiştirin:

```
".PHP" = "C:\\PHP\\PHP4isapi.dll"
```

Burada çok dikkat etmeniz gereken nokta, ters-bölü işaretlerinin başka yerlerde normal olarak yazdığımız gibi bir adet değil, iki adet olmasıdır. Dosyayı kaydettikten sonra, iki kere tıklayın ve Windows'un Registry'de değişiklik yapmayı isteyip, istemediğinize ilişkin sorusuna Evet yanıtını verin. Tamaam! Sizin de PHP bilen bir PWS'iniz oldu. Ama bunu bir sınayalım. Ve bunu bir Merhaba Dünya programı ile yapalım ki, gelenek bozulmasın!

Düzyazı programınızla şu metni merhaba.PHP adıyla PWS'in Web için kök dizin saydığı (muhtemelen C:\Inetpub\wwwroot) klasöre kaydedin:

```
<html>
<?PHP
      echo ("Merhaba Dünya!");
?>
</html>
```

Sonra Personal Web Manager kutusunda, Ana başlıklı bölümde, Yayımlama kısmındaki "http://" diye başlayan ve kişisel Web sunucunuzun adı ile devam eden köprüyü tıklayın. Varsayılan Browser'ınız, kişisel Web sunucunuzun ana sayfasını açacaktır. Browser'ın URL adres bölümüne, merhaba.PHP yazın ve klavyede Enter tuşuna basın. Eğer gerçekten PWS'iniz, PHP öğrenmişse, size ve dünyaya merhaba diyecektir.

Windows'da Apache!

NT sisteminizde ISS değil de Apache for Win 32 Web Server programını çalıştırıyorsanız, sürümüne göre httpd.conf veya srm.conf dosyasına şu ekleri yapmanız gerekir:

```
ScriptAlias /PHP3/"c:/path-to-PHP-dir/"
AddType application/x-httpd-PHP3 .PHP3
```

```
Action application/x-httpd-PHP3"/PHP3/PHP.exe"
```

Windows'da MySQL

PHP, Windows ortamında, Windows'un ODBC sürücülerini kullanarak, sürücüsü bulunan bütün veritabanlarına ulaşabilir. Ancak Windows ortamında geliştirseniz bile, Web sitenizi, Unix tabanlı ve ODBC-uyumlu olmayan bir sunucuya gönderebilirsiniz. Bu durumda sayfalarınızda kullanacağınız verileri muhtemelen MySQL aracılığıyla veritabanından çekeceksiniz demektir. Bu yüzden Windows sisteminize MySQL kurmak ve veritabanlarını bu araçla geliştirmeniz yerinde olur. Unix/Linux ortamında ücretsiz olmakla birlikte MySQL, Windows için bedeli paylaşımlı (shareware) olarak edinilebilir. Benioku dosyasında belirtildiğine göre, programı eğitim amacıyla ve 30 günü aşmamak kaydıyla kullanıyorsanız, ücret ödenmesi gerekmiyor. Diğer durumlarda TCX firmasının Internet'te http://www.tcx.se adresindeki sitesindeki formları doldurarak lisans alınması gerekiyor. Bu programı, MySQL'in http://www.mysql.com/downloads/ adresinden edinebilirsiniz. Bu adresteki mysql-shareware-3.22.34-win.zip dosyasını alarak, bir geçici dizinde açın. (Sisteminiz gerçek Internet sunucusu olarak ticarî amaçlı hizmet sağlayacaksa şu dosyayı almanız gerekir: http://www.mysql.com/Downloads/MySQL-3.23/mysql-3.23.21-win-src.zip) Shareware sürümünü içeren dosyayı açtığınızda ortaya çıkacak setup.exe programı MySQL'i kuracaktır.

MySQL Server'ın Windows'da da tıpkı Linux'ta olduğu gibi çalıştırılması ve durdurulması gerekir. MySQL'in paylaşım sürümünü kullanmak için Windows sistemlerinde mysqld-shareware.exe programını çalıştırmanız gerekir. DOS komut istemcisini açın ve "C:/mysql/bin" dizinine giderek, sırasıyla şu komutları verin:

```
mysqld-shareware

mysqlshow

mysqlshow -u root mysql

mysqladmin version status proc
```

Şimdi MySQL server çalışıyor ve biraz sonra PHP yoluyla kişisel Web Server'ınıza hizmet vermeye hazır demektir. Bu komutları icra ettiğiniz sırada bir hata mesajı ile karaşılaşırsanız, MySQL'in kuruluşunda hata yapmış olabilirsiniz. DOS ekrarında test ve MySQLl veri tabanlarının varlığını ve hizmete girdiğini gösteren mesajlar görürseniz, MySQL güzelce çalışıyor demektir. Bunu bir de Browser ile ve PHP yoluyla sınayalım:

Yukarıda IIS'e PHP desteği kazandırma çalışmaları sırasında oluşturmadı iseniz, düz yazı programınızda şu tek satırı yazın; php.php adıyla, kişisel Web'inizin kök dizinine kaydedin:

```
<? PHPinfo(); ?>
```

Browser'ınızı açın; URL hanesine örneğin http://localhost/PHP.PHP3 yazın. Açılacak sayfada iki şeye dikkat edin: MySQL ve ODBC bölümleri var mı? MySQL bölümü varsa; kendinizi yeniden kutlayın! ODBC bölümü yoksa, muhtemelen sisteminizde Microsoft'un Open Database Connectivity (Açık Veribanı Bağlantısı) sürücüleri kurulu değil demektir. (Windows'da Denetim Masası'nda ODBC simgesini görmüyorsanız, MS'un sitesinden gerekli kur programını indirmeniz gerekebilir. Linux'ta ise MyODBC programı ile bu imkanı kazanabilirsiniz.)

Bir kere daha tekrar etmek gerekirse, PHP ile veritabanına dayanan sayfalar yapmak için mutlaka ve sadece MySQL sürücülerini kullanmak gerekmez. Bunun yerine MS'un ODBC sürücülerini kullanabilirsiniz. Ancak Web sitenize evsahipliği yapan firmanın size ODBC desteği verip vermediğini öğrenmeniz gerekir. Aslında aynı şekilde, evsahibi firmanın MySQL desteği verip vermediğini de öğrenmek zorundasınız. (PHP ile MS-ODBC sürücülerini kullanabilmek için Andrew Stopford'un ASPToday sitesindeki http://www.asptoday.com/articles/20000202.htm yazısında geniş bilgi bulabilirsiniz.)

Geçmiş olsun. Bana sorarsanız, PHP ile dolu yeni bir hayata başlamak için en önemli adımı attınız. Bundan sonrası, sadece PHP dilini öğrenmekten ibaret. Dilin üretecilerinden Zeev Suraski, bu yılın başında İsrail'de yapılan ilk uluslararası PHP Konferansı'nda "PHP öğrenmek zor mu?" diye soran bir bilgisayar meraklısına şu cevabı

veriyordu: "Bir haftadan fazla zaman ayırmanız gerekiyorsa, sizin için zor demektir!"

PHP Dili

PHP bir Script dilidir; PHP ile yazdığınız "programlar" birer düzyazı dosyasıdır. Dil ögelerini ayrıntılı olarak ele almadan önce bir iki yazım kuralından söz edelim.

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 yorumlayıcısının kısa ayraç kullanmasını istemiyorsak, PHP.ini dosyasındaki, short_open_tag = On: satırını short_open_tag = Off; yapmak yeter. Bununla birlikte uzun vadede, PHP programlarınızda HTML yerine XML kullanmak istiyorsanız, şimdiden elinizi uzun ayraca alıştırmanız yerinde olur; çünkü XML etiketleri "<?xml" şeklinde başlar ve "?>" şeklinde biter.

ASP ile çalışan Web programcıları "<?" ve ?>" yerine sık sık "<%" ve "%>" yazarlar! PHP dilini gelişterinler., çok yaygın olan bu hatanın PHP programlarının çalışmasına engel olmasını önlemek için bize bir kolaylık sağlıyorlar. Eğer ASP ile çalışan ve daha sonra PHP'ye geçen bir programcı iseniz, PHP.ini dosyasını açarak, asp_tags = Off; satırını short_open_tag = On; yapabilirsiniz. Şimdi artık rahatça ASP ayracını PHP yazmak için kullanabilirsiniz!

PHP komutlarınızı daha uzun olmakla birlikte <SCRIPT> etiketiyle de kullanabilirsiniz. Yukarıdaki örnekte oluşturduğumuz merhaba.PHP'yi açın ve şu şekilde değiştirerek, merhaba2.PHP adıyla kaydedin:

```
<HTML>
<SCRIPT LANGUAGE="PHP">
      print ("Merhaba Dünya!");
</SCRIPT>
```

```
</HTML>
```

Web sunucusunda bu programı açarsanız, aynı sonucu aldığınızı göreceksiniz. Burada bir yanlış anlamaya yol açmayalım. Javascript veya VBScript gibi HTML kodlarının içine gömdüğümüz ve HTML sayfasının içinde, Web ziyaretçimizin bilgisayarına gönderilen ve orada (istemci-tarafından) icra edilen ve yine aynı şekilde SCRIPT> etiketi içine koyduğumuz Script kodları ile burada <SCRIPT> etiketi ile ifad eettiğimiz PHP kodlarının icra yerini karıştırmayalım. PHP, nasıl bir ayraç ve etiket ile yazarsak yazalım, daima sunucuda çalıştırılar ve ziyaretçiye asla PHP kodları gösterillmez. merhaba2.PHP'nin çalıştığı Browser'da kaynağı görüntülersek, istemciye gelen kodların saf-ve-temiz HTML olduğunu göreceğiz.

PHP kodlarımız, oluşturulmasını istediğimiz sayfanın HTML kodları ile tabir yerinde ise içiçe yazılır. Bunun bir örneği olarak yukarıdaki dosyayı şöyle değiştirerek, merhaba3.adıyla kaydedin:

```
<HTML>
<HEAD>
<TITLE>PHP ile Merhaba</TITLE>
</HEAD>
<BODY>
<CENTER>
<B>
<H1>
<?PHP
            print "Merhaba Dünya!";
?>
</H1>
</B>
</CENTER>
</BODY>
</HTML>
```

Sayfamızın görüntüsünde hiç bir değişiklik olmamakla birlikte, Browser'a giden HTML kodları önemli ölçüde değişmiş olacaktır.

Gerçi burada henüz ihtiyacımız yok; çünkü bir bakışta yazdığımız kodun marifetini anlayabiliyoruz; ama ciddî PHP programcılığına başladığımız zaman, hangi satırda neye niyet ettiğimi anlamamız daima mümkün olmayabilir. Biz kendi programımızı satır satır ezberlesek bile, işbirliği yaptığımız diğer Web programcılarının programımızı anlamalarını sağlamamız gerekir. Bunu, yorum satırları ile yapabiliriz. PHP ayraçları içinde iki türlü yorum bulunabilir: Çok-satırlı yorumlar ve tek satırlı yorumlar. Birinci grubu, başına "/*" ve sonuna "*/* işaretleri koyarak, ikinci grubu ise sadece başına *//* işareti koyarak belirtiriz:

```
<HTML>
<!- - Bu satır HTML'in yorum satırı
Buraya istediğimiz kadar yorum yazabiliriz..
Browser 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..
Bunu da PHP yorumcusu asla dikkate almaz
Buraya istediğimiz kadar yorum yazabiliriz.
*/
            print "Merhaba Dünya!";
// Bu ise PHP'nin tek satırlı yorum bölümü
// Bu satırları da PHP yorumcusu dikkate almayacaktır.
?>
</H1>
</B>
</CENTER>
</BODY>
</HTML>
```

PHP'rir bir diğer yorum işareti ise # (Diyez) işaretidir. Bu işaretin başında olrduğu satır da PHP tarafından dikkate alınmaz:

```
<?PHP
# Bu satır PHP'nin çok-satırlı yorum bölümü..
# Bunu da PHP yorumcusu asla dikkate almaz
# Buraya istediğimiz kadar yorum yazabiliriz.
            print "Merhaba Dünya!";
?>
```

Diyez ile yorum haline getirdiğimiz satırların sonuna başka işaret koymaya gerek yoktur.

PHP'nin Yapı Taşları

PHP, bir programlama dili olarak, değişkenler, değişkenlerin değerleriyle bir işlem yapmayı sağlayan işlemciler, işlemcilerle oluşturulan deyimler ve nihayet bunların tümünü toplu olarak kullanmamızı sağlayan işlevlere (fonksiyonlara) sahiptir. PHP, nesne-yönelimli (object-oriented) bir dil olduğu için, nesne oluşturma imkanına ve bunların kullanılmasını sağlayan metodlara da sahiptir. Bütün bu imkanları kullanarak, PHP ile bir veritabanından veri alarak, bunları HTML etiketlerinin değerleri olarak kullanabiliriz; Web sitemizi ziyaret eden kişilerden bilgi alabiliriz, bu bilgilerle işlemler yapabiliriz. PHP'nin çeşitli komutlarını, deyimlerini ve fonksiyonlarını kullanarak, programımızın çalıştığı Web sunucusunun bulunduğu bilgisayara da çeşitli dosya işlemleri yaptırabiliriz.

PHP programını durdurmak

Şimdi belki PHP'yi durdurmayı hiç düşünmüyorsunuz; ama ilerde bir fonksiyonun vereceği sonucun bütün programın durdurulmasını gerektiği zamanlar olabilir. PHP'nin resmî el kitabına bile ilk baskısında konulması unutulmuş olan programı durdurma komutu, exit kelimesidir; hiç bir parametre almadan çalışır.

Bu tür karmaşık uygulamalara geçebilmek için önce, PHP dilinin unsurlarını biraz yakından inceleyelim.

Değişkenler

Programcılıkta işlemlerimizi değişkenlerle yaparız. Değişkeni bir kap gibi düşünebilirsiniz. Sözgelimi "Gün," değişkenin adı ise bu değişkenin değeri Pazar, Pazartesi, Salı, vd., olabilir. Her değişken, türüne göre, ya bir ya da daha fazla değer tutar. Adından

da anlaşılacağı gibi değişkenin değeri değişir! Bu değişikliği biz yapabiliriz; programın kendisi yapabilir.

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;
$123;
$sevdigiRenk;
```

Değişkenler, harf, rakam veya alt çizgi (_) ile başlayabilirler; bu karakterleri içerebilirler; ama içinde boşluk veya diğer işaretler bulunamaz. PHP değişkenleri her türlü değişken tutabilirler: bir değişkenin adının yazılış şekli onun tutabileceği değerin niteliğini belirlemez. Dolayısıyla, "$adi" değişkenin değeri "Reşit" da olabilir, "1255" de olabilir.

PHP'de genellikle değişkenleri değerini atayarak belirleriz:

```
$adi = "Reşit";
$soyadi = "Gülen";
$123 = 123;
$sevdigiRenk = "yeşil";
```

Değişkenler, kullanıldıkları işleme, taşıdıkları değeri verirler:

```
print $adi;
```

komutu, eğer değeri "Reşit" ise "Browser penceresine "Reşit" kelimesini yazdırır.

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

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

Burada Browser penceresine yine "Reşit" 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. Bu, size şu anda karışık görünebilir. Daha sonra döngüleri gördüğümüzde bu tekniği bir satırlık kod ile yüzlerce değeri yazdırmakta kullanacağız. Bu teknik şöyle de kullanılabilir:

```
$degisken  = "adi";
$$degisken = "Reşit";
print "$adi";
print $$degisken;
print "${$degisken}";
print "${'adi'}";
```

Buradaki dört "print" komutu da Browser'a "Reşit" kelimesini yazdıracaktır.

PHP'de bir değişkenin tuttuğu değer, bir başka değişkene atanabilir. Bu ya duragan, ya da dinamik olarak yapılır. Duragan yöntemde:

```
$birinci_degisken  = "Nurcan";
$ikinci_degisken = $birinci_degisken;
$birinci_degisken = "Tülay";
print $ikinci_degisken;
```

Buradaki "print" komutu Browser penceresine "Reşit" kelimesini yazdıracaktır. İki değişken arasındaki değer alış-verişi birinci değişkenin tuttuğu değeri değiştirmeden önce olduğu için, ikinci değişken birinci değişkenin ilk değerini almış oldu. Fakat kimi zaman iki değişken arasındaki değer alış-verişi sürekli (dinamik) olsun isteriz. PHP, buna, ilk atama sırasında & işareti kullanırsak izin verir:

```
$birinci_degisken  = "Nurcan";
$ikinci_degisken = &$birinci_degisken;
$birinci_degisken = "Tülay";
print $ikinci_degisken;
```

Buradaki "print" komutu ise Browser penceresine "Tülay" kelimesini yazdıracaktır. Çünkü iki değişken arasındaki değer alış-verişi birinci değişkenin tuttuğu değeri değiştirmeden önce halde, ikinci değişkene birinci değişkenin değerini dinamik olarak atamış olduk. Atama işlemi sırasında & işareti kullandığımız için, birinci değişkenin değeri değiştiğinde ikinci değişkenin de değeri değişecektir.

Veri Türleri

PHP, değişkenlere, tuttukları değere göre farklı bellek alanı tahsis eder; bu bakımdan verilerimizin türü etkin bir programcılık açısından önem taşır. Ayrıca PHP, diğer

bütün programlama dilleri gibi belirli veri türleri ile belirli işlemleri yapar veya yapamaz. Örneğin, ancak iki sayı arasında aritmedik toplama yapılabilir. İki metin tutan değişken aritmetik olarak toplanamaz; ancak içerdikleri değerler ardarda eklenebilir. Ne var ki, PHP sizin aritmetik toplama mı, yoksa ardarda ekleme mi yapmak istediğinizi bilmek durumunda değildir. (PHP'ye 'muazzam dil' dedik; içinde yapay zekâ var demedik!) Dolayısıyla değerleri 5 ile 6 olan iki değişkeni topladığınız zaman ortaya 11 yerine 56 çıkarsa, kabahati PHP'de değil, değişkenlere veri atayan kişide (büyük bir ihtimalle kendinizde) aramalısınız.

PHP açısından dünyada altı tür değer vardır:

Tamsayı (Integer): 5,124, 9834 gibi

Çift (Double): 3,567 gibi

Alfanümerik (String): "Reşit" gibi

Mantıksal (Boolean): doğru (true)/yanlış (false) gibi

Nesne (Object)

Dizi (Array)

Bazı uzmanlara göre, gerçekte PHP'de mantıksal (boolean) değişken türü yoktur. B kanının sebebi, PHP'nin true (doğru) ya da false (yanlış) olarak atadığımız değerleri 1 ve boş/null değerlere çevirmesi, mantıksal olarak sorgulandığında 0 dışındaki tam ve ondalık değişkenleri TRUE, 0'ı FALSE sayması, ve boş alfanümerik değişkenleri FALSE, dolu alfanümerik değişkenleri ise TRUE olarak değerlendirmesidir. Ancak daha sonra gereceğimiz gibi PHP, Mantıksal/Boolean adında bir değişken değiştirme fonksiyonuna sahiptir ve bu tür değişkenlerin türü sorulduğunda "boolean" karşılığını verir.

Tür Değiştirme

Değişkenlere atadığımız değerlerinin türlerini genellikle biliriz; ama yüzlerce değişkenle uğraştığımız bir Web programında değişken türünü unutmak çok kolaydır. Kimi

zaman da değişkenlere program yoluyla değer atarız; başka bir deyişle programımız bazı değişkenleri kendisi oluşturabilir, ve bunlara kendisi değer atayabilir. Bu tür program yoluyla atanan değerlerin de türünde kuşkuya kapıldığımız anlar olabilir. Bir değişkenin değerinin türü hakkında kuşkunuz varsa, en emin yol bunu PHP'nin kendisine sormaktır. Bu sorgulamayı gettype() fonksiyonu ile yaparız.

Fonksiyon

PHP'nin gettype() ve print() gibi, kendi içinde bulunan kullanılmaya hazır bir çok fonksiyonu vardır. Daha sonra kendi fonksiyonlarımızı yazmanın yollarını da göreceğiz. Programcılık dilinde ister dile ait olsun, ister kendimiz yazmış olalım, bir fonksiyonu kullanmaya "fonksiyon çağırma" denir. Fonksiyonlar da fonksiyon çağırabilirler. Çağrılan bir fonksiyon ya kendisinden beklenen işi yapar, ya da kendisini çağıran işleme veya fonksiyonu bir değer sunar. Buna fonksiyondan dönen değer denir.

Şimdi, bir PHP programı yazalım, bir takım değişkenlere değerler atayalım ve bunların türlerini PHP'ye soralım. Bir düzyazı programı ile şu metni yazın ve turler.php adıyla kaydedin:

<HTML>

```
<HEAD>
<TITLE>PHP'de Degisken Turleri</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>
<B>
<H2>
<?php
            $sayi  = 5;
            print("Birinci değişkenin adı: \$sayi<br>");
```

```
            print("Değeri : ");
            print "$sayi<br>";
            print("Türü : ");
            print gettype( $sayi ) ; //tamsayi/integer
            print "<br>";
            print "<br>";

            $alfanumerik  = "Reşit";
            print "İkinci değişkenin adı: \$alfanumerik<br>";
            print "Değeri : ";
            print "$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>";
?>
</H2>
</B>
</BODY>
```

```
</HTML>
```

Bu programı çalıştığımızda karşımıza şöyle bir görüntünün gelmesi gerekir:

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

turler.php dosyasını yazdıysanız, şu satırdaki, ters-bölü işareti dikkatinizden kaçmamış olmalı:

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

Daha önce gördük ki, tek veya çift tırnak içine de alsak, PHP, bir değişken adını gördüğü zaman onun yerine o değişkenin tuttuğu değeri yazar. PHP bunu, değişken adının başındaki $ görerek yaparak. $ işareti gibi PHP için özel anlamı olan işaretlerin anlamlandırılmasını önlemek ve bu işaretleri düz metin saymasını sağlamak için bu işaretlerin önüne ters-bölü işareti koyarız. Buna o karakteri kurtarma veya ESCaping denir. PHP'nin anlamlı işaretleri ve bunların ESCape-yazılışı 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. Elimizde böyle bir imkan var diye, bir değişkenin alfanümerik (String) olan değerini sayıya çevirebileceğini sanmayın. PHP buna çok kızar! Şu örneği de tur_degistir.php adıyla kaydedelim:

```
<HTML>
<HEAD>
<TITLE>PHP'de Degisken Turleri</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>
<B>
<H2>
<?php
            $degisken  = 5.67890;
            print("Değişkenin  değeri : ");
            print "$degisken<br>";
            print("Türü : ");
            print gettype( $degisken ) ; //çift,ondalık/double
            print "<br>";
            print "<br>";

            print "İlk değiştirme işlemi: Alfanümerik/String:<br>";
            settype( $degisken, string ); //alfanümerik/string
            print "Değeri : ";
            print "$degisken<br>";
            print("Türü : ");
            print gettype( $degisken ) ; //alfanümerik/string
            print "<br>";
            print "<br>";

            print "İkinci değiştirme işlemi: Tamsayı/Integer:<br>";
            settype( $degisken, integer ); //Tamsayı/Integer
            print "Değeri : ";
            print "$degisken<br>";
            print("Türü : ");
```

```
            print gettype( $degisken ) ; //Tamsayı/Integer
            print "<br>";
            print "<br>";

            print "Üçüncü değiştirme işlemi: Ondalık/Double:<br>";
            settype( $degisken, double ); //çift,ondalık/double
            print "Değeri : ";
            print "$degisken<br>";
            print("Türü : ");
            print gettype( $degisken ) ; //çift,ondalık/double
            print "<br>";
            print "<br>";

            print "Dördüncü değiştirme işlemi: Mantıksal/Boolean:<br>";
            settype( $degisken, boolean ); // Mantıksal/Boolean
            print "Değeri : ";
            print "$degisken<br>";
            print("Türü : ");
            print gettype( $degisken ) ; // Mantıksal/Boolean
            print "<br>";
            print "<br>";
?>
</H2>
</B>
</BODY>
</HTML>
```

Bu programı çalıştırdığımızda, ilk atadığımız değerin ondalık/double olarak belirlendiğine dikkat edin. Bu değerin türünü alfanümerik/string olarak değiştirdiğimizde değişmediğini görüyoruz. Fakat tamsayı/integer'a çevirdiğimizde PHP değerin tamsayı bölümünü alıyor; ondalık-kesir bölümünü atıyor. Nitekim daha sonra değişkerin türünü yeniden ondalık yaptığımız halde, atılan bu bölüm geri gelmiyor. Son olarak değişkeni mantıksal ifade olarak değiştirdiğimizde, değerin doğru/true olduğunu anlıyoruz; değer olarak bize 1 dönüyor. PHP, sıfır dışındaki tüm değerleri doğru/true'ya çevirir; sıfır'ı ise yanlış/false olarak atar ve boş/null değer verir.

Şu anda PHP'yi tam bilemediğimiz için nerede kullanılır kestiremiyoruz, ama diyelim ki burada olduğu gibi, değişkeni bir daha düzelmeyecek şekilde değişikliğe uğratmak istemeyebiliriz; fakat yine de değerinin türünü değiştirmek isteyebiliriz. Bunu da nerede kullanabiliriz, şu anda bilmiyoruz. Ama diyelim ki, böyle bir işlem yapmaya ihtiyacımız oldu! Hatırlıyor musunuz, biraz önce özel bir şekilde bir değişkene başka bir değişkenin adını değer olarak vererek, aslında bu değişkenin ilk değişkenin değerini almasını sağlamıştık. Burada da aynı tekniği kullanabiliriz. Bu kodu da tur_kopyala.php adıyla kaydedin:

```
<HTML>
<HEAD>
<TITLE>PHP'de Degisken Turleri Kopyalama (Casting)</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>
<B>
<H2>
<?php
            $degisken  = 3.1418;
            print("Değişkenin  değeri : ");
            print "$degisken<br>";
            print("Türü : ");
            print gettype( $degisken ) ; //çift,ondalık/double
            print "<br>";
            print "<br>";

            print "İlk kopyalama işlemi: Alfanümerik/String:<br>";
            $kopya_degisken =  ( string ) $degisken; //alfanümerik/string
            print "Değeri : ";
            print "$kopya_degisken<br>";
            print("Türü : ");
            print gettype( $kopya_degisken ) ; //alfanümerik/string
            print "<br>";
```

```
            print "<br>";

            print "İkinci kopyalama işlemi: Tamsayı/Integer:<br>";
            $kopya_degisken =  ( integer ) $degisken; //Tamsayı/Integer
            print "Değeri : ";
            print "$kopya_degisken<br>";
            print("Türü : ");
            print gettype( $kopya_degisken ) ; //Tamsayı/Integer
            print "<br>";
            print "<br>";

            print "Üçüncü değiştirme işlemi: Ondalık/Double:<br>";
            $kopya_degisken =  ( double ) $degisken; //çift,ondalık/double
            print "Değeri : ";
            print "$kopya_degisken<br>";
            print("Türü : ");
            print gettype( $kopya_degisken ) ; //çift,ondalık/double
            print "<br>";
            print "<br>";

            print "Dördüncü kopyalama işlemi: Mantıksal/Boolean:<br>";
            $kopya_degisken =  ( boolean ) $degisken; // Mantıksal/Boolean
            print "Değeri : ";
            print "$kopya_degisken<br>";
            print("Türü : ");
            print gettype( $kopya_degisken ) ; // Mantıksal/Boolean
            print "<br>";
            print "<br>";
?>
</H2>
</B>
</BODY>
</HTML>
```

Bu programın can alıcı noktası örneğin "$kopya_degisken = (boolean) $degisken;" şeklindeki komut; burada $kopya_degisken adlı değişkene $degisken adlı değişkenin değerini (eşittir işareti ile) kazandırırken, arada yeni değişkenin kazanmasını

istediğimiz türün adını parantez içinde yazıyoruz. Burada gerçekte, $degisken adlı değişkenin değerine hiç bir şey olmuyor. Bu programı çalıştırırsanız, ilk değerin ontalık kısmının önceki örnekte olduğu gibi değişim işlemi sonunda tamamen yok olmadığını görebilirsiniz.

$degisken adlı değişkenin değeri değişmeden durduğu için üçüncü adımda yıptığımız ondalık tür olarak kopyalama sonucu yeni değişkenin değeri yine kesiri ile karşımıza çıkıyor.

Dört yararlı fonksiyon

PHP'de değişken değer türü değiştirmekte kullanabileceğimiz kullanılmaya hazır fonksiyon vardır. ilerde döngüleri ele aldığımızda daha çok anlam kazanacak olan bu değişkenleri şimdilik bir kenara kaydedin:

isset() ve **unset()**

PHP, bir değişken tanımlandığı anda, sanki içinde değer varmış gibi, bilgisayarın belleğinde yer ayırır. Bu bazen bilgisayarın kıt kaynaklarını zorlayabilir. isset() fnksiyonu, PHP'nin bir değişkenin içinde değer bulunup bulunmadığını sınamasını sağlar. unset() ise varolan bir değişkeni yok eder. Bu iki fonksiyonu birlikte kullanabiliriz ve boş değişkenleri yok ederek, bilgisayarın belliğinde yer açabiliriz:

```
if (isset($bir_degisken)) {
print( $bir_degisken );
}
else {
unset($bir_degisken);
}
```

Bu kod parçası, $bir_degisken 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);
```

Bu program parçası Browser penceresine sadece empty() fonksiyonun doğru sonuç verdiği birinci echo() deyiminde 1 yazdıracaktır.

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>");
            }
            $b = "Reşit";
if (is_double($a)) {
            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.

intval(), **doubleval()**, **doubleval()**

Bir değişkenin değerinin türünü başka bir türe çevirmeye yararlar. intval(), ortaya çıkartacağı tamsayı değişkenin 10 tabanlı (decimal) veya 16 tabanlı (hexadecimal) olmasını sağlayan ikinci bir argüman alabilir:

```
<?php
            $a = "123ABC";
            $b = inval($a, 16)
            print ($b);
?>
```

Bu program Browser penceresine, 123'ün hexadecimal ifadesi olan "1194666" yazdıracaktır.

İşlemciler (Operatörler)

Yukarıdaki örnekleri birlikte yaptıysak, = işaretini bol bol kullandık ve böylece PHP'nin bir çok operatöründen biriyle tanıştık. Buna PHP dilinde "atama operatörü" denir. Bu işlemci sadece yukarıda kullandığımız gibi, bir değişkenin bir değere eşitliğini ilan etmek amacıyla kullanılmaz. Örneğin:

```
print ( $sayi = 123 ) ;
```

ifadesinde olduğu gibi, bir fonksiyonun içinde de hem atama işlemi yapılabilir; hem de işlem yürütülür. PHP'nin aritmetik, birleşik-atama, birleştirme, karşılaştırma ve mantıksal sınama operatörleri vardır.

Aritmetik işlemciler:

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

Aritmetik işlemleri gerçek sayılarla yaptığımız gibi, tuttuğu değer sayı olan değişkenlerle de yapabiliriz.

Daha önce, PHP'de değişken tanımlar ve bunlara değer atarken sayı olan değişkenleri tırnak işareti kullanmadan, alfanümerik değişkenleri ise tırnak içinde yazdığımızı hatırlayacaksınız. Bu, geleneksel hale gelmiş olan bir programcılık alışkanlığıdır.

PHP, tırnak kullanmasanız da bir değişkenin sayı olup olmadığını anlar; daha sonra bu değişkenlerle aritmetik işlem yapabilir. PHP ayrıca rakamla başlayan alfanümerik değişkenleri sayı olarak kullanmaya kalktığınızda, bu değerin başındaki bütün rakamları sayı olarak kabul eder. PHP açısından harf veya işaretle başlayan alfanümerik değişkenlerin sayı olarak değeri sıfırdır. Örnek:

```
$a = 6;
$b = "22/A"
print ($a + $b)
```

sonuç olarak 28'i verecektir; veya

```
$a = 6;
$b = "A/22"
print ($a + $b)
```

sonuç olarak 6 çıkacaktır. Bu küçük deneyi yaparsanız, PHP ile program yazarken hangi değişkenin ne sonuç vermesi gerektiğini dikkatle düşünmek zorunda olduğunuzu görürsünüz. Bir kişinin geliri ile adresini toplamak, programcı olarak arzu ettiğiniz bir işlem olmayabilir!

Bununla birlikte elimizde bir kişinin adı ve soyadı ayrı değişkenler halinde ise, bunları birbirine ekleyerek (concatenation) ortaya yeni bir alfanümerik değişken çıkartabiliriz:

```
<?php
            $adi = "Reşit";
            $soyadi = "Muhtar";
            $adi_soyadi = $adi . " " . $soyadi;
            print ("<br>");
            print ($adi_soyadi);
            print ($adi . "'nın soyadı " . $soyadi . "'dir.")
?>
```

Bu örnekte, sadece ekleme yoluyla yeni bir değişken oluşturmakla kalmıyoruz; fakat alfanümerik değişkenleri, başka metinlerle de birbirine ekleyebildiğimizi görüyoruz. Bu işlemi nokta işaretiyle (.) yapıyoruz. Bu işlemciyle sadece değişkenlerin değerlerini değil fakat metinleri de birbirine ekleyebiliriz:

```
<?php
print ("İyileri iyikleri ile alkışla; " . "..." . "ve sen soluklarında daima Mesîh ol..! ")
?>
```

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"

Bu işlemi zihnimizde, "A değişkeninin mevcut değerine 5 ekle, çıkan sonucu A değişkenin yeni değeri yap!" şeklinde canlandırabiliriz.

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.

++ veya -- işaretinin değişken adından önce veya sonra olmasının sonucu farklı olur. İşaretler değişkenin adından önce ise PHP önce değişkenin değerini bir arttırır veya eksiltir, sonra bu yeni değeri işleme katar; işaretler değişken adından sonra ise, PHP değişkenin o andaki değerini işleme kadar, sonra değeri bir arttırır veya 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. Söz gelimi, değeri 6 olan değişkenin 5'ten büyük olup olmadığını sınarsak, sonuç doğru; 5'ten küçük olup olmadığını sınarsak sonuç yanlış çıkacaktır. PHP'nin karşılaştırma işlemcileri şunlardır:

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 veya eşitse	$a >= 5	Doğru/True

PHP'nin karşılaştırma işlemcileri hem tam ve ondalık sayı türü değerlerle, hem de alfanümerik değerlerle kullanılabilir.

Daha ileri aşamalarda, bazen değişkenlerimizin karşılaştırmasının bu kadar basit olamayacağını; kimi zaman iki grup değişkenin karşılaştırılması gerektiği durumların bulunduğu göreceğiz. Diyelim ki, bir öğrencinin başarılı sayılması için hem vize notunun, hem de yılsonu sınav notunun belirli sayılardan büyük olması gerekiyor. Bu durumu ancak karmaşık-mantıksal sınamadan geçirmemiz gerekir. Bu sınamayı iki ayrı sınama sonucu ile birlikte kullanırız. Örneğin:

vize notu yeterli - final notu yetersiz = kalır (yanlış/false)

vize notu yeterli - final notu yeterli = geçer (doğru/true)

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

"veya" ve "ve" sınamaları için iki ayrı işlem işareti bulunmasının sebebi, PHP'nin işlem sırasıyla ilgilidir. Birazdan oraya geliyoruz. Şimdi, kavranması kolay olmayan ikili mantıksal sınamalara bir örnek verelim; yukarıda öğrencinin durumunu PHP komutu olarak yazalım.

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

Mantıksal karşılaştırma işlemleri daha da karmaşık olabilir. Örneğin, öğrencinin geçmiş sayılması için yılsonu vize ve final notlarının 45'dan yüksek, ve ya birinci sömester notunun ya da vize ve final notlarının ortalamasının 65'den yüksek olması şartı aranabilir. (Kabul; kötü bir örnek oldu. Ama sırf alıştırma için!) Bu durumda kodu şöyle yazmamız gerekirdi:

```
<?php
  $vize = 45;
  $final = 65;
  $ilk_sem = 45;
  $ortalama = ($vize + $final) / 2;
  if (($vize >= 45 && $final >= 45) and ($ilk_se >= 65 || $ortalama >= 65)) {
  print ("Öğrenci geçti!");
  }
```

```
   else {
   print ("Öğrenci kaldı!");
   }
?>
```

Burada, PHP önce || veya && işaretinin sonucunu bulacak sonra and veya or işlemcisinin gereğini yapaçaktır. Bu kodun vereceği sonucu, bilgisayarda çalıştırmadan tahmin edebilir misiniz? İpucu: Bu öğrencinin yerinde olmak istemezdim!

İşlem önceliği deyince.. Bütün programlama dilleri gibi PHP için de işlemlerin hangi sırada yapıldığı önem taşır. "5 + 4 * 2" işleminin sonucu, önce 5 ile 4'ün toplanması ve sonucun 2 ile çarpılması halinde başka, 4'ün 2 ile çarpılması ve sonucun 5 ile toplanması halinde başka olacaktır. Bu bakımdan, PHP'nin hangi işlemi önce yaptığı, hangi işlemi sonra yaptığını bilmemiz gerekir. PHP'deki işlemlerin yapılış sırası şöyledir:

++ ve --

/ * %

+ -

< <= => >

== === | =

&&

||

= += -= /= %= .=

and

xor

or

r.

Sabit Değerler

İster Web, ister Web-dışı amaçlı olsun, bir programın bazen başındanh sonuna kadar değeri değişmeyen değişkene ihtiyacı olabilir. "Değeri değişmeyen değişken"

teriminin anlamsız olduğu kanısında iseniz, bunlara sabit değerler adını da verebilirsiniz! PHP'de sabit değerler, Script boyunca değişmeden kalır.

Sabit değerler, değişkenlerden farklı şekilde oluşturulur. Bunun için PHP'nin define() fonksiyonunu kullanırız. Bu fonksiyonun yazım kuralı şöyledir:

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);
?>
```

Burada tanımladığımız DOLAR_KURU sabit-değeri, tıpkı bir değişken gibi kullanılabilir. Fakat bir sabit değeri tanımladıktan sonra içeriğini değiştiremezsiniz. Sabit değer ile aynı ismi taşıyan değişken oluşturmak mümkündür; ancak aynı ismi taşısa da bir sabit ile değişkeni (birinin adının önünde $ işareti bulunduğu için) karıştırmak kolay olmasa gerek.

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

Bu programı, bu şekilde ve define() komutunu en üst satıra alarak çalıştırırsanız, ikinci çalıştırışınızda programın sabit değeri yeniden oluşturmayı reddettiğini görecektiniz. Sabit değerleri, Web sitesinde ziyaretçilerin girdiği verilerden alarak kendiliğinden oluşturan bir program yazdığınızda, defined() fonksiyonu mantıksal hata yapmanızı önler.

PHP'nin kullanılmaya hazır bir kaç sabid değeri vardır. True (doğru) değeri 1 olan, False (yanlış) ise değeri sıfır sayısı veya boş alfanümerik olan iki sabittir. Diğer sabitler şunlardır:

__FILE__ : O anda çalışmakta olan PHP dosyasının adı (kelimenin önünde ve sonunda iki alt-çizgi var);

__LINE__ : Bu ifadenin yer aldığı satırın sayısı (kelimenin önünde ve sonunda iki alt-çizgi var);

PHP_VERSION: PHP'nin sürümü (İki kelimenin arasında bir alt-çizgi var);

PHP_OS: PHP'nin çalıştığı işletim sistemi (İki kelimenin arasında bir alt-çizgi var);

PHP_VERSION : PHP'nin sürümü (İki kelimenin arasında bir alt-çizgi var);

Bunlara ek olarak, E_ERROR, E_WARNING, E_NOTICE ve E_PARSE şeklinde hata durumunda hatanın çeşitli özelliklerini bildiren sabit değerleri de kullanabiliriz.

PHP hakkında çok daha geniş bilgiyi phpinfo() fonksiyonunu çağırak alabilirsiniz.

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 şxöyle 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

PHP'de Program Denetimi

Bilgisayar programı, belirli değerlere göre kendi izleyeceği yolu kendisi bulmalıdır. Bir Web programı da aynı mantıkla tasarlanır. PHP, bir kısmını daha önceki bölümlerde, bir kısmını da şimdi ele aldığımız araçları kullanarak, belirli durumlarda belirli istikametlere gidecek, yeri geldiğinde geri dönecek, yeri geldiğinde başka bölümlere sıçrayacak bir akış planı tasarlamamıza imkan verir.

Web programlarımızda, ziyaretçilerimizin sitemizde yaptıkları veya yapmak istediklerini bildirdikleri işe göre sayfalar, bu işlere göre sayfa içerikleri sunarız. Web programımız ziyaretçimizin hareket tarzına ve amacına uygun olarak ilerler. Bir programın akış planını kontrol etmemizi sağlayan unsurlar çoğu zaman programın kendi kendine karar vermesini sağlayan araçlardır. PHP'de program denetim unsurları, programın gidiş yönünü değiştirmemizi veya olduğu yerde durmasını ve beklemesini veya bir işi biteviye yapmasını sağlar. Bu bölümde bu araçları ele alacağız.

if Deyimi

İnsan olarak kararlarımızı hemen hemen daima "bir şey öyle ise böyle, öyle değilse şöyle davranmak" üzere almaz mıyız? PHP programında if deyimi bunu sağlar. Kelime anlamı eğer olan if deyimi ile programımızı karar almaya zorlarız. Ne var programımız bizim kadar akıllı olamayacağı için if şartının doğru veya yanlış olması durumunda ne yapacağını da ona biz söyleriz. if deyimi şöyle yazılır:

```
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 ifadesinin doğru olması halinde, ifadeye ait ilk süslü parantezin içindeki komutları icra eder; bu şartlar doğru değilse, elseif deyimi ile ileri sürebileceğimiz diğer bir grup şartın doğru olup olmadığını sınayabiliriz. Bu şartlar yerine geliyorsa, PHP, elseif bölümündeki işleri icra eder. Nihayet diğer her hangi bir durumda icra edilmesini istediğimiz komutları else bölümünde belirtiriz. PHP, if deyimi doğru değilse, deyime ait süslü parantezin içindeki işleri yapmadan yoluna devam eder; varsa elseif deyiminin şartlarını sınar; elseif bölümü yok fakat else bölümü varsa bu bölümdeki işleri icra eder. Deyimde else bölümü de yoksa, PHP yoluna if deyiminden sonraki ilk adımdan devam eder. Bunu bir örnekle belirtelim:

```
<?php
            $sarki = "Güneş Doğacak";
            if ( $sarki == "Güneş Doğacak" ) {
            echo ("Seçtiğiniz şarkının adı: Güneş Doğacak!");
            }
            elseif ($sarki == "Beklerim Ben") {
            echo ("Seçtiğiniz şarkının adı: Beklerim Ben!");
            }
            else {
            echo ("Lütfen bir şarkı seçiniz!");
            }
?>
```

Çoğu zaman if deyimlerini birden fazla koşulu sınayacak şekilde yazarız; bununla birlikte elseif ve else bölümleri olmadan da yazılabilir:

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

Buradaki if deyimi $parola değişkeninin boş alfanümerik olması halinde ziyaretçiyi uyarak ve görevi bitecektir. Ziyaretçi bir parola yazmışsa, daha sonraki komutlar bu parolanın doğru olup olmadığını sınayabiliriz.

switch deyimi

PHP'de program akışını yönlendirmekte kullandığımız bir diğer unsur, switch deyimidir. Adı anahtar anlamına gelen switch deyimi, verilen bir değişkenin değerinin sıraladığımız koşullardan hangisine uygun olduğunu sınar ve o koşula ilişkin komutları icra eder. PHP'nin yaptığı işi bitirdikten sonra switch deyiminin dışına çıkmasını sağlayan, break komutu vardır:

```
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 deyimini yazarken mutlaka break ifadesini kullanmamız gerekmez; switch deyiminin akışı break ile kesilmemişse, program ilk şart yerine gelmiş olsa bile diğer şartları da sınamaya devam eder ve en sondaki default deyimine ulaşarak (default'tan sonra noktalı virgül değil iki nokta üstüste işareti olduğuna dikkat edin), bu bölümdeki komutları icra eder. Ne var ki programcılık mantığına aykırı olabilir: bir değişkenin içeriği

verdiğimiz şartlardan birini tutmuşsa, muhtemelen diğer şartları yerine getirmeyecek demektir. Bu durumda diğer koşulların sınanması gereksiz e default işlerin icra edilmesi ise hata olur. Bu deyimi de bir örnekle görelim:

```
switch ( $sarki ) {
case "Güneş Doğacak" ;
            echo ("Seçtiğiniz şarkının adı: Güneş Doğacak<br>") ;
break;
case "Derdime Derman" ;
            echo ("Seçtiğiniz şarkının adı: Derdime Derman<br>") ;
break;
case "Haberi Yok" ;
            echo ("Seçtiğiniz şarkının adı: Haberi Yok<br>") ;
break;
case "Sonsuza kadar" ;
            echo ("Seçtiğiniz şarkının adı: Sonsuza kadar <br>") ;
break;
default:
            echo ("Seçtiğiniz şarkının adı bilinmiyor<br>") ;
}
```

Programımız, bu kodu icra ederken,, $sarki adlı değişkenin değerini, buradaki dört isimle karşılaştıracaktır. Şarkının adı, "Güneş Doğacak" ise ilk durum gerçekleşmiş olacak; program Browser pencerisine şarkının adını yazacak (ve muhtemelen bu sözlerini görüntüleyecek!) ve ikinci adıma break ifadesine geçecektir. break, switch deyiminin sona ermesini sağlar; PHP yoluna, switch deyiminin kapanan süslü parantezinden sonra devam eder. Bu örnekte break ifadesi kullanılmamış olsaydı, PHP, diğer şartları da sınacak ve boşun azaman kaybedecekti; daha da garibi, en sondaki default komutu da icra edileceği için programımız Browser pencerisine hem seçilen şarkının adını, hem de "Seçtiğiniz şarkının adı bilinmiyor" yazacaktı. Bunu gözönünde tutarak, switch deyimini daima break ile birlikte yazmamız gerekir.

switch için kısa yol

Kimi zaman uzun uzadıya switch deyimi yazmamız gerekmeyebilir; bir değişkenin değerini sadece bir şarta uygunluk açısından sınamamız gerekir. Bunu soru ve iki nokta üstüste işaretleri (? ... :) olarak yazılan işlemci ile yaparız. Diyelim ki, Web sitemizi ziyaret eden kişinin parola yazıp yazmadığını sıramak istiyoruz:

```
<?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.

Döngüler

Yukarıdaki iki denetim ögesi de programın ileri doğru akışını bozmayan araçlardır. Oysa bazen programımızın bir koşul gerçekleşincele kadar yerinde sayması ve koşul gerçekleşmedikçe ilerlememesi gerekebilir. Kimi zaman da programımıza bir işi bitirmeden ileri gitmemesini biz bildiririz. Program akışının bu tür denetimini döngü deyimleri sağlar. PHP'de iki tür döngü vardır: bir durum devam ettiği veya etmediği sürece yapılan ve kaç kere tekrar edeceği bilinemeyen while döngüleri, ve tekrar sayısı belirli for döngüleri.

while döngüsü

Bir değişkenin içeriğinin belirli bir şartı karşılaması veya karşılamaması halinde icra edilir. Burada dikkat edeceğimiz nokta, programın icrası sırasında değişkenin içeriğinin veya koşulun değişmesinin sağlanmasıdır. Aksi taktirde pıgramımız sonsuz döngüye girer ve muhtemelen çöker. while döngüsü şöyle yazılır:

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

Bu yöntemin yaygın kullanıldığı alan, bir sayaçla yaptırılan işlerdir. Aşağıdaki programı while.php adılma kaydedip, çalışrırsanız, PHP, Browser'ın yazı türü boyutunu 1'den 7'ye kadar büyüterek, aynı metni yazdıracaktır:

```
<HTML>
<HEAD>
<TITLE>PHP'de Döngü</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>
<?php
            $sayac = 1;
            while ( $sayac <= 7 ) {
            print ("<font size= $sayac >");
            print ("<b><p>İyileri iyilikleri ile alkışlayınız!</b></p>");
            print ("</font>");
            $sayac ++;
            }
?>
</BODY>
</HTML>
```

Programımız, while döngüsü içindeki iken, sonsuz döngüden kurtulabilmek için, yazıyı yazdırdıktan sonra, $sayac değişkenini bir arttırıyor. Aksi taktirde, değişken hiç bir zaman while döngüsünün beklediği şarta kavuşarak 7'ye eşit hale gelemeyecektir. (Programı, sondaki "$sayac ++;" satırını kaldırarak denerseniz, zararsız bir sonsuz

döngünün ne olduğunu görebilirsiniz! Ama önce Browser'ın dur düğmesinin yerinde durup durmadığna bir bakın!)

while döngüsün aradığı şartın yerine gelmesi, her zaman burada olduğu gibi, tabir yerinde ise "elle" yapılmaz; döngünün icra ettiği komutlar veya fonksiyonlar koşulu içeren değişkeni değiştirebilirler. Sözgelimi elektronik ticaret sitemizde Web ziyaretçimizin while içinde yaptığı alışverişin tutarını koşul olarak kullanabiliriz ve bu koşulu içeren değişken alışveriş ile değişebilir.

do..while

while döngüsü farkettiğiniz gibi, ileri sürdüğümüz şartı, iyi yapmadan önce sınar; ve bu şart ortadan kalkmamışsa (yani henüz doğru/true ise) yapacağı işi yapar; başka bir deyişle while döngüsünün yapacağı iş hiç yapılmayabilir. Fakat sınamanın iş yapıldıktan sonra yerine getirildiği bir şekli de vardır: do..while.. Bu döngü ise şöyle yazılır:

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

Burada gördğünüz gibi do..while döngüsü en az bir kere icra edilir; çünkü şartın sınanması yapılacak işe ilişkin komutlardan sonra gelmektedir. Bu döngünün aradığı şartın döngünün yaptığı işlerin sonucu veya kod bölümünde bizim tarafımızdan gerçekleştirilmesi için igerekli komutların bulunmasına dikkat etmelisiniz. Yoksa, bu döngü de sonsuzluğa erişir! Yukarıda örneği, bu yöntemle yazalım:

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

```
?>
```

Bu döngüyü yazarken, while satırının sorunda noktalı virgül bulunduğuna dikkat edin.

for döngüsü

PHP'de Web programlarımız, döngünün belirli bir sayıda olmasını ve mesela bu sayının bizim istediğimiz basamaklarda artmasını gerektiriyorsa, döngüyü for deyimiyle kurabiliriz. Ayrıca for deyimi sayaç gibi şartın yerine gelmesini sağlayacak arttırma veşa eksiltme işlemlerini kendisi yapacağı için, sonsuz döngüye girme tehlikesi de hemen hemen yoktur. Bu döngü şöyle yazılır:

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

for deyiminin ilk ögesi, döngü sayacı olarak kullanılacak bir değişken oluşkturulması ve buna değer atanmasına ilişkindir; sonra bir noktalı vrigül ile bu sayacın sınırı olan koşul belirtilir. Yine noktalı virgül ile ayırarak, sayaç değişkeninin artış hızı verilir. Bunu da yine yukarıdaki örnekle gösterelim:

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

Buradaki kod, while ve do..while göre çok daha kısa; ve while döngüsü sayacı arttırmayı sağlayan satır unutulsa bile çalışacağı halde, for döngüsü unsurlara ilişkin noktalı virgüller eksik olursa çalışmayacak ve bize böylece sonsuz döngü tehlikesini hatırlatmış olacaktır. for döngüsü kendi sayacını verdiğiniz şartı karşılayıncaya kendisi arttıracak, yani buradaki örneği göre $sayaç değişkeni 7 oluncaya kadar çalışacak ve şart karşılandığı anda, sona erecektir.

foreach

PHP'nin programcıya çok kolaylık sağlayan bir diğer döngüsü, foreach döngüsüdür. Ancak bir değişkenin bütün elemanlarını tekrar sayısında esas alan bu döngünün çalışma ilkelerini görebilmek için dizi değişken kavramını ele almış olmamız gerekiyor. Şimdilik bu döngünün varlığını bir kenara yazıp, dizi değişkenlere kadar, unutabilirsiniz.

Döngüyü sona erdirmek için: break

Programımızın akışı, bazen döngünün durdurulmasını gerektirir. Örneğin, sayaç değişkenini ziyaretçinin vermesi gerektiği ve vermediği durumda, döngünün başlamaması yerinde olur. Programın gerektirdiği böyle bir sebeple döngüyü break komutuyla durduruz.

Sözgelimi aşağıdaki kodda kullandığımız $tekrar değişkenini değerini ziyaretçiden form yoluyla aldığımızı varsayalım:

```
<?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>");
            }
?>
```

Bu kod parçasını, bir sayfa kodunun içine yerleştirir ve önce bu şekliyle, danha sonra $tekrar= 0 yazarak sınarsanız, break komutunun işe yaradığını göreceksiniz. ($tekrar değişkeninin değerini, Form yoluyla ziyaretçiden değişken değeri almayı öğreninceye kadar şimdilik kaydıyla biz yapıyoruz. Ayrıca buradaki if deyiminin süslü parantez olmadan yazıldığına dikkat edin.

Döngüyü sürdürmek için: continue

break ifadesinin döngüyü tamamen durdurduğunu farkettiniz mi? Oysa programımızın akışı, kimi zaman da döngünün sadece belirli bir durumda kendisinden beklenen işi yapmamakla birlikte böyle tümüyle kesilmesini de gerektirmez. Yine yukarıdaki örneğe dönersek, sayaç değişkenini ziyaretçinin vermesi gerektiği ve vermediği durumda, döngünün tümüyle kesilmesini değil de, sadece sayacı gösteren $sayac değişkeninin sıfırdan büyük bir değere ulaşlmasını beklememiz yerinde olabilir. Bu devamlılığı continue komutuyla sağlarız:

Yine bu örnekte kullandığımız $sayaçc değişkeninin değerini ziyaretçiden form yoluyla aldığımızı varsayarak, kodu şöyle değiştirebiliriz:

```
<?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.)

Buraya kadar, PHP programı oluşturabilmek için gerekli temel yazım kurallarını, yapıtaşlarını ve program akışında yararlanacağımız kontrol ögelerini ele almış olduk. Şimdi bu unsurların hepsini bir arada kullanabileceğimiz grupları, fonksiyonları ele alabiliriz.

Fonksiyonlar

Bir bilgisayar programının kalbi, fonksiyonlardır. Fonksiyonlar, bir kere tanımlanır ve bir çok kereler, ihtiyaç olduğu noktada hizmete çağrılır. Fonksiyonlar genellikle bir iş veya hesaplama yaparlar ve bunun sonucu olarak ortaya yeni bir değer çıkartırlar. Bu değere "fonksiyondan dönen değer" denir.

Fonksiyonun bir program için ne kadar önemli bir unsur olduğunu bir örnekle belirtelim. Diyelim ki Web'de öğrencilerimizin yıl içinde aldıkları notları gösteren bir programımız var. Bu siteyi ziyaret eden öğrencilerin, iskterlerse, bir düğmeyi tıklayarak, o andaki not ortalamalarını hesap ettirmelerini sağlamak istiyoruz. Bu hesabı yapacak olan işlem, bir fonksiyondur. Web ziyaretçisi düğmeyi tıkladığı anda, program, ortalama hesabını yapacak olan fonksiyonu, o sayfada bulunan notları bildirecek; fonksiyon da kendisine verilen bu değerlerle yapması gereken hesabı yapacak; ve elde edeceği sonucu kendisini göreve çağıran program satırına bildirecektir. Bildirilen bu değerle ne yapılacağı, fonksiyonu göreve çağıran komuta veya işleme aittir. Fonksiyon kendisinden isteneni yapmakla yetinir.

PHP'nin kullanılmaya hazır bir çok fonskiyonu vardır. Şu ana kadar sürekli kullandığımız echo() ve print() bu tür PHP fonksiyonlarıdır. PHP'nin kendi fonksiyonlarını göreve çağırmak, yine yukarıdaki örneklerde olduğu gibi, program içinde gerektiği yere adını ve kullanacağı değeri yazmakla olur:

```
print ("Bu bir metindir!");
```

PHP bizim kendi fonksiyonlarımızı kullanmamıza da izin verir. Fakat bunun için önce fonksiyonu tanımlamamız gerekir.

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
 gibi etiketler koydurduğumuza dikkat etmiş olmalısınız. Bir programda bu işi sık sık yapacaksak, yani metinlerimiz
 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:

```
<HTML>
<HEAD>
<TITLE>PHP'de Fonksiyon</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>
<?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. Bu programı çalıştırırsanız, Browser'da bu görüntüyü elde edebilirsiniz:

Şimdi, bu penceredeki belgenin kaynağını görüntüleyelim. Karşımıza şuna benzer bir metin çıkmış olması gerekir:

```
<HTML>
<HEAD>
<TITLE>PHP'de Fonksiyon</TITLE>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
<BODY>
<h1>Bu H1 Başlık</h1>
<h2>Bu H2 Başlık</h2>
<h3>Bu H3 Başlık</h3>
<h4>Bu H4 Başlık</h4>
Bu kendisinden sonra BR olan birinci metin.<br>
Bu kendisinden sonra BR olan ikinci metin.<br>
<p>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.</p>
<p>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.</p>
</BODY>
</HTML>
```

Bu, bize, oluşturduğumuz bütün fonksiyonların kendilerinden bekleneni yaptığını ve kendilerine verdiğimiz metinleri, tıpkı istediğimiz gibi, arzu edilen HTML etiketleri ile birlikte Browser'a gönderdiklerini gösteriyor.

Buradaki fonksiyon, kendisini bir iş yapan türden fonksiyondur. Oysa kimi zaman fonksiyonlar, kendilerini göreve çağıran komuta bir değer gönderirler. Şimdi de böyle bir fonksiyon örneği yazalım. Aşağıdaki kodu, hesapla.php adıyla kaydedin:

<HTML>

```
<HEAD>
<TITLE>PHP'de Fonksiyon</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>
<?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>");
?>
</BODY>
</HTML>
```

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, fonksiyonun 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

Bu sayfanın kaynağındaki HTML kodları ise şöyla olacaktır:

```
<HTML>
<HEAD>
<TITLE>PHP'de Fonksiyon</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>
17<br>7<br>60<br>2.4<br>
</BODY>
</HTML>
```

Fonksiyonlarımız, kendilerinden bekleneni yaparak, gereken hesaplamanın sonucunu, kendilerini göreve çağıran komuta, burada PHP'nin print() fonksiyona göndermiş; bu fonksiyon da sadece bu sonuçları Browser'a iletmiş bulunuyor.

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. Şu örneği yazdir02.php adıyla kaydedin:

```
<HTML>
<HEAD>
<TITLE>PHP'de Fonksiyon</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>
<?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.");
?>
```

```
</BODY>
</HTML>
```

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.

Bunu, görüntülenen sayfanın kaynağında rahatça görebiliriz:

```
<HTML>
<HEAD>
<TITLE>PHP'de Fonksiyon</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>

<font size="5">Bu Başlık</font><br>
<font size="2">Bu küçük boyutta bir metin</font><br>
<font size="3">Bu varsayılan boyutta bir metin</font><br>
<font size="8">Bu çok büyük Başlık</font><br>
<font size="3">Bu uzun bir paragraf metni. Boyutu varsayılan ölçüde. Bu uzun
bir paragraf metni. Boyutu varsayılan ölçüde. Bu uzun bir paragraf metni.
Boyutu varsayılan ölçüde. </font><br>
</BODY>
</HTML>
```

Programda, yazdir() fonksiyonunun ikinci ve dördüncü kullanımında ikinci argümanın bulunmadığına, yukarıdaki HTML'de bu satırlar için font size="3" etiketinin kullanıldığına dikkat edin.

Değişkenlerin kapsamı: global ve static

Yukarıda hesapla.php örneğini birlikte yaptıysak, bir nokta dikkatinizi çekmiş olmalı: 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. Bunu hemen bir örnekle görelim. Şu kodu kapsam01.php adıyla kaydedin:

```
<HTML>
<HEAD>
<TITLE>PHP'de Fonksiyon</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>
<?php
$metin = "Başkalarına yararlı olmanın sınırı yoktur!";
function yazdir () {
            print ("<h1>İşte metin: $metin </h1>");
            }
// Başka kodlar buraya girebilir
            yazdir();
?>
</BODY>
</HTML>
```

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:

Aynı kodun fonksiyon bölümünü şöyle değiştirerek, kapsam02.php adıyla kaydedin:

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

Programımızı çalıştırdığımızda, bu kez metnin arzu ettiğimiz gibi görüntülendiğini göreceğiz.

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; ama daha basit bir örnek olarak bu kodu static.php adıyla kaydedelim:

```
<HTML>
<HEAD>
<TITLE>PHP'de Fonksiyon</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>
<?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();
?>
</BODY>
</HTML>
```

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.

Dizi-Değişkenler, Nesneler

PHP'de, diğer programlama dillerinde olduğu gibi, kullandığımız bilgileri, verileri değişken adını verdiğimiz kaplarda tuttuğumuzu biliyoruz. Bu kaplarla ne işler yapabileceğimizi de gördük. Fakat bu kapların sadece bir birim bilgi tuttuğunu farketmiş olmalısınız. Oysa programlarımızda genellikle, sözgelimi "öğrencilerin adları" dediğimiz zaman yüzlerce ismi tutabilen kaplara ihtiyacımız olur. Aksi taktirde, öğrenci sayısı kadar değişken oluşturmamız gerekir ki, bu durumda değişken oluşturma zahmetinden vazgeçmek daha uygun olur; öğrencilerin isimleri, değişkene koymadan da kullanabiliriz. İşte, bu tür, bir grup bilgili topluca tutan değişkenlere dizi-değişken (array) denir; ve program boyunca işe yarayan değişken türü de budur.

PHP'nin diğer bir çok dilden farkını sağlayan bir özelliği, büyük ölçüde dizi değişkenlere dayanan, içinde kendi değişkenleri ve kendi fonksiyonları olan nesne (object) oluşturma imkanına sahip bulunmasıdır. Bu açıdan PHP, nesne-yönelimli (Object-oriented) diller grubuna girer. Nesne oluşturmak ve kullanmak, bu yönteme alışık olmayan programcılar için önceleri çok zor görünür. Ancak bir kere nesne ile programlamanın özünü kavrayanların bir daha kolay kolay nesne fikrinden vazgeçtikleri de görülmemiştir.

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:

Öğrenci No	Adı	Soyadı	Sınav-1	Sınav-2	Not
1	Özbay	Altun	..	..	..
2	Muharrem	Taç	..	..	..

3	Erhan	Mataracı	..	..	..
4	Hasan	Civelek	..	..	..
5	Şahika	Tabak	..	..	..

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. Bu kodu dizi_degiskenler01.php adıyla kaydedin:

```
<HTML>
<HEAD>
<TITLE>PHP'de Degiskenler</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>
<B>
<H2>
<?php
$ogrenciler = array (
array ( adi => "Özbay", soyadi => "Altun", sinav1 =>"", sinav2 =>"", not =>""),
array ( adi => "Muharrem", soyadi => "Taç", sinav1 =>"", sinav2 =>"", not =>""),
array ( adi => "Hasan", soyadi => "Civelek", sinav1 =>"", sinav2 =>"", not =>""),
array ( adi => "Şahika", soyadi => "Tabak", sinav1 =>"", sinav2 =>"", not =>""),
);
// Buraya başka kodlar girecek
print $ogrenciler[0][adi];
?>
</H2>
</B>
</BODY>
</HTML>
```

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 => "Özbay", soyadi => "Altun", "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. dizi_degiskenler01.php dosyasında şu değişikliği yaparak, dizi_degiskenler02.php adıyla kaydedin

```
<?php
                    $ogrenciler[0][adi] = "Özbay";
                    $ogrenciler[0][soyadi] = "Altun";
                    $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 ("Özbay", "Muharrem", "Hasan", "Şahika");
```

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, dizi_degiskenler03.php adıyla kaydedin:

```
<?php
                    $ogrenciler[] = "Özbay";
                    $ogrenciler[] = "Muharrem";
                    $ogrenciler[] = "Hasan";
                    $ogrenciler[] = "Şahika";
// 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 "Özbay" 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 ve dizi_degiskenler04.php adıyla kaydedelim:

```
<?php
                    $ogrenciler[] = "Özbay";
                    $ogrenciler[] = "Muharrem";
                    $ogrenciler[] = "Hasan";
```

```
                    $ogrenciler[] = "Şahika";
// Buraya başka kodlar girecek
                    $ogrenciler[0] = "Emre";
                    $ogrenciler[15] = "Özbay";

             print ("Dizideki 1'nci isim: $ogrenciler[0] <br>");
             print ("Dizideki 2'nci isim: $ogrenciler[1] <br>");
             print ("Dizideki 3'ünci 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 Özbay değil Emre yazdığını göreceğiz.

Bunun sebebi, diziyi oluşturan ilk grup deyimden sonra,

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

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. Daha önce kaydettiğimiz dosyada şu değişikliği yapalım ve dizi_degiskenler05.php adıyla kaydedelim:

```
<?php
             $ogrenci[adi] = "Özbay";
             $ogrenci[soyadi] = "Altun";
             $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. Daha önce kaydettiğimiz dizi_degiskenler04.php dosyasını açın ve PHP bölümünde şu değişikliği yaparak dizi_degiskenler06.php adıyla kaydedin:

```
<?php
                    $ogrenciler[] = "Özbay";
                    $ogrenciler[] = "Muharrem";
                    $ogrenciler[] = "Hasan";
                    $ogrenciler[] = "Şahika";
// Buraya başka kodlar girecek
print ("\$ogrenciler adlı dizide ". count($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, dizi_degiskenler06a.php adıyla kaydelim.

```
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. $ogrenci değişkenin "Özbay," "Muharrem," "Hasan" ve "Şahika" değerleri bulunduğuna göre, bu değişkenin count()'u 4 olacaktır. ("Ozbay" = 0, .. "Şahika" = 3 olmak üzere..) Bu print() komutu, Browser penceresine tahmin ettiğiniz gibi "$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 "Şahika" 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. Yukarıdaki örnekteki bütün print() ve for satırlarını atarak yerine şunları yazıp ve dizi_degiskenler06b.php adıyla kaydelim.

```
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 gönöderecektir. 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. Daha önce yazdığımız dizi_degiskenler05.php adlı dosyayı açarak mevcut tek satırlık print() kodunun yerine şunları yazın ve dizi_degiskenler07.php adıyla kaydedin:

```
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! Şimdi bu bölümün başında yazdığımız dizi_degiskenler0i.php adlı

dosyayı açalım tek satırlık print() komutunun yerine şu satırları yazarak dizi_degiskenler08.php adıyla kaydedelim:

```
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 ( "Özbay" , "Muharrem" , "Hasan" , "Şahika" );
$ikinci_dizi = array ( "Altun" , "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 ( "Özbay" , "Muharrem" , "Hasan" , "Şahika" );
$yeni = array_push ($birinci_dizi, "Altun" , "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 ( "Özbay" , "Muharrem" , "Hasan" , "Şahika" );
$silinen = array_shift ($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 ( "Özbay" , "Muharrem" , "Hasan" , "Şahika", "Altun" ,
"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:

```
$birinci_dizi = array ( "Özbay" , "Muharrem" , "Hasan" , "Şahika", "Altun" ,
"Taç" , "Civelek" , "Tabak");
sort ($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:

```
$birinci_dizi = array ( ogr_01=>"Özbay", ogr_02=>"Muharrem" , ogr_013>"Hasan" , ogr_04=>"Şahika");
asort ($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:

```
$birinci_dizi = array ( ogr_01=>"Özbay", ogr_02=>"Muharrem" , ogr_013>"Hasan" , ogr_04=>"Şahika");
ksort ($birinci_dizi);
```

PHP, şimdi bu diziyi anahtarlara göre alfabetik sıraya sokacaktır.

Nesneler

Nesne yönelimli (Object-oriented) programlama icad edildiğinde, geleneksel programlamaya aşina olanlar önce tereddüt ettiler; sonra bunun ne harika bir teknik olduğnunu gördüler ve nesnelerden vazgeçmez oldular. Kabaca tanımlarsak, nesne, kendi değişkenleri ve icra edeceği komutlardan oluşan fonksiyonları ile bir bütündür. Nesneyi bir kere tanımladıktan sonra istediğimiz kadar örneğini oluşturabiliriz. Bir nesnenin yapacağı işten, o nesnenin metodu diye söz ederiz. Bu açıdan bakarsanız, programlarımızda nesneler sadece metodları için işe yararlar.

Diyelim ki programımızda bir "öğrenci" nesnesi oluşturmak istiyoruz. Bu nesnenin içinde adı, soyadı ve notlar gibi değişkenler, ve bu değişkenlerle yapılan bir takım işler bulunsun. İşe nesnenin tabir yerinde ise şablonu olan class'ı oluşturmakla başlayalım; gerisini de yaptıkça görelim.

Bir Nesne Oluşturalım

Bir nesne oluşturmak için önce onu tanımlamamız gerekir. Bunu PHP'nin class deyimini kullanarak yapabiliriz. Bir nesnenin özellikleri (properties) ve metodları (methods) vardır. Şu kodu nesneler01.php adıyla kaydedin:

```
<HTML>
<HEAD>
<TITLE>PHP'de Degiskenler</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>
<B>
<H2>
<?php
class ogrenci {
// özellikleri tanımlayalım
            var $adi;
            var $soyadi;
            var $sinav1;
            var $sinav2;
            var $not;
// metodları tanımlayalım
function adi_belirle ($n) {
            $this->adi = $n;
            }
function soyadi_belirle ($n) {
            $this->soyadi = $n;
            }
function sinav1_belirle ($n) {
            $this->sinav1 = $n;
```

```
            }
function sinav2_belirle ($n) {
            $this->sinav2 = $n;
            }
function not_hesapla() {
            $this->not = ($this->sinav1 + $this->sinav2)/2;
print ($this->adi. " " . $this->soyadi . " için not ortalaması: ". $this->not);
            }
}
//Buraya başka kodlar girecek
$ogr1 =  new ogrenci();
$ogr1 -> adi_belirle("Şahika");
$ogr1 -> soyadi_belirle("Tabak");
$ogr1 -> sinav1_belirle(7);
$ogr1 -> sinav2_belirle(10);
$ogr1 -> not_hesapla();
?>
</H2>
</B>
</BODY>
</HTML>
```

Bu programda ogrenci adlı bir nesne tanımlıyoruz; ve daha sonra bir değişken adına new komutu ile bu nesnenin bir örneğini oluşturuyoruz. Nesnelerin yeni bir örneği veya kopyasını çıkartmak ifadeleri aslında yaptığımızı tam anlatmıyor. Nesnenin tanımı bir adet; her new komutu ile bu nesnenin özelliklerine ve metodlarına sahip yeni bir nesne yapmış oluyoruz. Nitekim bu işe insanın dilini dolayan İngilizce bir kelimeyle Instantiation (yeni bir varlığını oluşturma) deniyor. Bu örnekte, $ogr1 adlı değişken gerçekte, ogrenci nesnesinin tam bir örneği: içinde beş değişken ve altı metod var. Nesneyi bir kere tanımladıktan sonra programın daha ileri aşamalarında bu istediğimiz kadar örneğini farklı isimler vererek oluşturabiliriz. Şimdi şu satıra dikkat edelim:

```
$ogr1 =  new ogrenci();
$ogr1 -> adi_belirle("Şahika");
```

Burada $ogr1'in parametrelerine nasıl değer yazdırdığımızı görüyorsunuz. Nesnenin metodlarından biri olan adi_belirle fonksiyonuna bir değer veriyoruz: "Şahika"; nesne oluşturulurken yazılmış olan bu fonksiyon ise aldığı değeri, kendi ait olduğu nesnenin bir değişkenine kaydediyor:

```
function adi_belirle ($n) {
            $this->adi = $n;
            }
```

Bu ve diğer fonksiyonlarda kullandığımız "$this->" ifadesi, kendisine ulaştırılan değeri bir parametre olarak kullanıyor ve "$adi" değişkenine yazıyor. "this" (bu) kelimesi o anda nesnenin o anda oluşturulmakta olan örneğine göndermede bulunur. "->" işlemcisini kullanarak, istediğimiz nesnenin istediğimiz metoduna veya parametresine değer gönderebiliriz. Bir nesnenin yeni bir örneğini oluşturduğumuz zaman, bu örneğin bütün parametrelerini sağlamak veya bütün metodlarını kullanmak zorunda değiliz.

Yukarıdaki örneğe göre yeni örnekler oluşturarak en az dört öğrencinin not ortalamasını hesabedebilir misiniz?

PHP ile Web programcılığı yolunda hızlı adımlarla yol alıyoruz. Bu bölümde gördüğümüz dizi değişkenler ve nesneler, Web sayfalarımızda bir çok işi adeta otomatik hale getirecek unsurlar olarak kullanılacak.

PHP İşbaşında

PHP'de bir programı oluşturmaya ve programın akışını kontrol etmeye yetecek kadar bilgi sahibi olduk. Bu bölümde, bu bilgilerimizi uygulamaya koyacağız; ve bu arada halâ öğrenmemiz gereken bir iki noktaya ayrıntılı olarak değineceğiz. Özellikle metinlerin düzenlenmesi konusuna bakmamız ve ve Düzenli İfadeler (Regular Expressions) denen kavramla tanışmak ve bunun araçlarını öğrenmek zorundayız. PHP'nin işbaşında olması demek, PHP bilgimizi HTML sayfalarımızı PHP ile oluşturma, Form bilgilerini derleme, bir veritabanından veri alarak bunları sayfalarımızda kullanma ve ziyaretçinin vereceği bilgileri veritabanına işleyebilmek demektir. Şimdi kolları yeniden sıvayalım ve PHP'yi uygulamaya koyulalım.

Formlar

Web sunucusu ve istemcisi (Browser) arasındaki etkileşme ve alışverişin nasıl işlediğine ilişkin bilgilerimiz olduğunu varsayarak, sadece Internet'te Form, bir Web sayfasının ziyaretçiden veri alabildiği ve bunları Web sunucusuna ulaştırabildiği başlıca araç olduğunu hatırlayalım. Form, ziyaretçinin bizim istediğimiz bilgilerin yanı sıra ziyaretçinin bilgisayarından Web sunucusu bilgisayara, daha bir çok bilgiyi de beraberinde getirir. Web programcısı ve Web tasarımcısı olarak bu bilgileri bilmeye daima ihtiyacımız var. Söz gelimi, ziyaretçimizin Browser türünü ve sürümü belirleyerek, onu, uygun sayfaya yönlendirmekten tutun, ziyaretçiden istediğimiz bilgilerin Sunucu'ya ulaştığında nerede ve hangi değişkende tutulduğuna kadar, gerekli bir çok bilgi Sunucu çievre değişkenleri ve Sunucu değişkenleri dediğimiz dizilerde bulunur. Form tasarımına ve bir Form'un bilgilerini PHP ile yakalamaya ve işlemeye geçmeden önce Web Server'ı ve bize verdiği bilgileri daha yakından tanınamız yararlı olur.

Bunun için önce şu programı formlar01.php adıyla kaydedin ve çalıştırın:

```
<HTML>
<HEAD>
<TITLE>PHP'de Nesneler</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>
<?php
foreach ($GLOBALS as $anahtar=>$deger ) {
            print ($anahtar . " =  " . $deger . "<br>");
            }
?>
</BODY>
</HTML>
```

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 http://www.mycgiserver.com/~ocal/formlar01.php adresinde görebilirsiniz. Sunucu'nun oluşturduğu diğer değişkenleri ise http://www.mycgiserver.com/~ocal/php.php adresinde inceleyebilirsiniz.)

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. Aşağıdaki kodları formlar02.htm adılyla kaydedin:

```
<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=Muharrem+Ta%E7&adres=muharremtac@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. Şu aşağıdaki kodları formlar02_isle.php adıyla kaydedin:

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

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.

Önce Form'umuzu değiştirelim ve formlar03.htm adıyla kaydedelim:

```
<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:

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

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=Şahika+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.

URL Kodları

HTTP protokolüne göre, temel ASCII listesi içinde yer almayan karakterler, ve tabiî bu arada sadece Türkçe'de bazı diğer alfabelerde bulunan harfler, bir Form'da yer aldığı taktirde, Browser tarafından URL şemasına göre kodlanarak gönderilir. Gerçi bir çok HTTP sunucu programıbu karakterlere tanıyabilir ve bir dosyaya yazarken doğru şekilde yazabilirler; ama bu çevirme işleminin bazen program yardımıyla yapılması gerekebilir. Bizim için önemli karakterler ve URL kodları şöyle:

ü = %FC

Ü = %DC

ö = %F6

Ö = %D6

ı = %FD

İ = %DD

ğ = %F0

Ğ = %D0

ş = %FE

Ş = %DE

ç = %E7

Ç = %C7

% = %25

& = %26

[= +%5B

] = %5D

{ = %7B

} = %7D

? = %3F

= = %3D

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. Biraz önceki kaydettiğimiz dosyanın sadece Form bölümünü aşağıdaki gibi geliştirerek, formlar03a.htm adıyla kaydedelim:

```
<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=Şahika 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 değiştirir ve formlar03c_isle.php adıyla kaydedersek, 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; ve bunu formlar04.php adıyla kaydedelim:

```
<?php
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?)

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. Önce şu dosyayı, dosya_gonder.php adıyla kaydedin:

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

```
<TITLE>PHP'de Dosya Gönderme</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>
<?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");
      }
}
?>
</BODY>
<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>
</BODY>
</HTML>
```

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:

Dosya İşlemleri

Web programıcısı olarak ziyaretçinin bize göndereceği bilgileri yakalamayı öğrendik. Şimdilik sadece ziyaretçinin Browser'ına geri gönderdiğimiz bu bilgileri, sunucu tarafından çeşitli işlemlerde kullanabiliriz. Bu işlemlerin başında dosya girdi/çıktı işlemleri gelir. Başka bir ifade ile ziyaretçiden aldığımız bu bilgileri sunucuda bir dosyaya yazdırabiliriz. Bu bölümde PHP ile sunucuda yapabileceğimiz dosya işlemlerine bakacağız. Tabiî dosya işlemleri dendiğinde sadece ziyaretçiden aldığımız bilgileri bir dosyaya yazdırmakla yetinmeyeceğiz; PHP programlarımıza sunucuda bir dosyada bulunan bilgileri okumasını da öğreteceğiz.

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:

```
<HTML>
<HEAD>
<TITLE>PHP'de Harici Dosya Ekleme</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>
<?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>");
?>
</BODY>
</BODY>
</HTML>
```

Buradaki include komutu biraz önce yazdığımız harici dosyayı alacak, içeriğini aynen kendi bulunduğu noktada, PHP'nin oluşturacağı HTML dosyasına katacaktır. Bu programı çalıştırdığınızda görüntü şöyle olacaktır:

Bu görüntüyü aldığınız sırada Browser'ınızda kaynağı görüntülerseniz, iki metnin adeta birleştirilmiş olduğunu göreceksiniz. Fakat dışardan PHP programına sadece düz metin eklemeyiz; program veya fonksiyon da ekleyebiliriz. Harici dosyada şu değişikliği yaparak, harici_dosya02.x adıyla kaydedin.

```
<?php
print ("Ben de harici dosyayım.!<br>");
print ("Ama ben hesap da yaparım.. Örneğin iki iki daha ". (2 + 2). " eder!");
?>
```

Haricî dosya okuyan programı da programı içinde, okunacak dosya adını düzelttikten sonra dosya_ekle02.php adıyla kaydedin; ve çalıştırın.

Haricî dosyamızda yer alan ...iki iki daha ". (2 + 2). " eder!"); şeklindeki ifadenin programa gelirken "..iki iki daha 4 eder!" şekline gelmesinin izahı, PHP'nin dışarıdan aldığı

dosyanın içinde PHP programı olduğunu gördüğü anda, bu dosyanın içeriğini metin olarak değil, program olarak ele almasıdır. Bu yolla, PHP programlarımıza bir değerin dönmesini de sağlayabiliriz. Haricî dosyamızda şu değişikliği yapalım ve harici_dosya03.x adıyla kaydedelim:

```
 <?php
$sonuc = ( 2 + 2 );
return $sonuc;
?>
Bu satır bakalım programa dahil olacak mı?
```

Sonra, program metninde şu değişlikliği yaparak, dosya_ekle03.php adıyla kaydedin ve çalıştırın:

```
<?php
$donen_sonuc = include("harici_dosya03.x");
print ("\n<p> Harici dosyadan dönen sonuç: $donen_sonuc </p>");
?>
```

Haricî dosyamızın içindeki program, return komutu ile sadece kendi içinde elde ettiği sonucu, onu çağıran programa verdi; ve bu sonucu alan programımız, sadece kendi içindeki bir değişkenin değerini Browser'a gönderdiği halde, bu değer haricî dosyadan alınmış oldu.

Haricî dosyaların include yoluyla PHP programlarımıza katılması, özellikle bir sitenin bir çok programında sık sık kullanılan metinlerin ve hesaplamaların bir kere yazılmasını ve tek satırla çağrılmasını sağladığı için kolaylık sağlar ve hata ihtimalini azaltır.

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!");
?>
</BODY>
</HTML>
```

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
 işaretini Browser penceresine gönderiyor. Programı bu şekliyle sınarsanız, dosyadaki metnin tek karakter olarak Browser penceresinin soluna dizildiğini göreceksiniz. Programı
 kodunu silerek çalıştırırsanız, bu kez dosyadaki metinde var olan satır sonu işaretlerinin de kaldırıldığını ve metnin bir paragraf oülarak görüntülendiğiini 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. (Bu programı geliştirerek, önce dizinleri, sonra dosyaları yazabilen biçime getirmek mümkündür. Bunun bir örneği bu kitapçığı örnek kod dosyaları arasında dizin_oku01a_server.php ve dizin_oku01a_pws.php aadıyla bulhabilirsiniz.)

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");
```

Bir Dosya İşlemi Örneği: Konuk Defteri

Böylece PHP ile "iş yapmak" için gerekli araçlarımıza bir yenisini ekledik. Artık PHP programlarımızdan sabit diskteki dosyalara ulaşmayı ve dosyalarla çalışmayı biliyoruz. Buraya kadar öğrendiklerimizi bir PHP programı üzerinde uygulayabiliriz. Bunun için birlikte bir konuk defteri yapalım.

Diyelim ki konuk defterimize sadeceziyaretçimizin adını, elektronik posta adresini ve kısa bir mesajını isteyeceğiz. Öyle bir sayfa yapalım ki; ziyaretçi bu sayfayı ilk kez açtığında, yani henüz Form'dan sunucuya bir veri gelmemiş dolayısıyla $HTTP_POST_VARS dizi değişkenin içinde değer bulunmadığı sırada, programımız bu durumu belirleyerek ziyaretçiye konuk defteri Form'nu sunmalı; ziyaretçi formu doldurup gönder düğmesini tıkladığı zaman programımız bir yandan ziyaretçinin gönderdiği bilgileri kendisine gösterir ve teşekkürlerimizi aktarırken, diğer yandan da bu bilgileri bir düzyazı programı girmeli. Bu sırada ziyaretçimize isterse defterimizdeki diğer mesajları okuma imkanı da sunabilmeliyiz. Biraz uzunca olan şu programı kd_01.php adıyla kaydedelim. (Bu dosyayı,

kitapçığın örnek kodlar arasında bulabilirsiniz; fakat bu alıştırmayı bir düz yazı programı ile buradaki örnek ifadeler ve alan/değişken adlarını değiştirerek tekrar etmeniz yerine olur:

```
<?php
if ( isset ( $HTTP_POST_VARS )) {
// Form doldurulmuş ise
echo ("
    <HTML>
    <HEAD>
    <TITLE>PHP'de Misafir Defteri</TITLE>
    <meta http-equiv='content-type' content='text/html; charset=ISO-8859-9'>
    <meta http-equiv='Content-Type' content='text/html; charset=windows-1254'>
    </HEAD>
    <BODY>
    <P><font size='3' class='s4' face='Arial, Helvetica'>
    ");
```

Programımızın buraya kadar olan bölümü, if ile $HTTP_POST_VARS dizi değişkeninde değer bulunup bulunmadığına ilişkin sınamanın olumlu sonuç vermesi halinde uygulanan bölümüne dahil. Burada dikkatini echo() fonksiyonu ile birden fazla satır yazdırdığımız çekmiş olmalı. echo(), fonksiyonuna Browser penceresine göndereceğimiz değerleri çift-tırnak içinde yazacağımız için, HTML etiketlerinin çift-tırnak gerektiren etiketlerinde tek tırnak kullandığımıza dikkat edin. if sınavının olumlu sonucuna ilişkin bölümü yazmaya devam edelim:

```
$dosya_adi = "mesajlar.txt";
if (file_exists("$dosya_adi")) {
      $yazilacak_dosya = fopen($dosya_adi,'a');
      }
else {
      $yazilacak_dosya = fopen($dosya_adi,'w');
      }
foreach ($HTTP_POST_VARS as $anahtar=>$deger ) {
       $yeni_anahtar = strtoupper($anahtar);
       print ("$yeni_anahtar: $deger<br>");
       $metin = "$yeni_anahtar: $deger<br>";
       fputs($yazilacak_dosya, $metin);
```

```
        }
        fputs($yazilacak_dosya, date("D d M Y h:i:s"));
        fputs($yazilacak_dosya, "<br>\n");
        fputs($yazilacak_dosya, "<center>
        <img src='nokta.gif' width=100% height=1 border=0></center>\n");
fclose($yazilacak_dosya);
```

Konuk defteri programımızHTML için gerekli temel etiketleri ürettikten sonra bu bölümde veritabanı dosyası olarak kullandığımız mesajlar.txt isimli dosyanın bulunup bulunmadığını sınıyor; dosya varsa ek yapmak için ('a'),dosya yoksa yazmak için ('w'), dosyayı açıyoruz. Sonra, $HTTP_POST_VARS dizi-değişkenine Form'dan gelmiş olan anahtar=değer çiftlerinin her birini (yani, Browser'dan gelen örneğin "adi" değişkeni ve bunun değeri olan ismi) önce print() ile ziyaretçinin Browser'ına, sonra da fputs() ile biraz önce açtığımız veritabanı dosyasına yazdırıyoruz. Fakat burada bundan önce Form'dan küçük harf olarak gelen değişken adlarını (anahtarları) strtoupper() fonksiyonu ile büyük harfe çeviriyoruz.Şimdilik bu fonksiyonun üzerinde durmayın; biraz sonra metin düzenleme ve düzenli ifadeler konusuna geleceğiz. Ancak bu sırada ziyaretçilerin mesajlarını daha sonra güzel görüntelemek için aralarına çizgi olarak kullanmak üzere bir resim dosyasını IMG etiketi ile yerleştirdiğimize de dikkat edin. Bu grafik dısyasının eni-boyu 1 piksel olduğu halde burada ziyaretçinin Browser penceresinin eni kadar genişleterek çizgi üretmiş oluyoruz! Evet, koda devam:

```
echo ("
    <HTML>
    <HEAD>
    <TITLE>PHP'de Misafir defteri</TITLE>
    <meta http-equiv='"content-type' content='text/html; charset=ISO-8859-9'>
    <meta http-equiv='Content-Type' content='text/html; charset=windows-1254'>
    </HEAD>
    <BODY>
    <P><font size='3' class='s4' face='Arial, Helvetica'>
    <center><img src='nokta.gif' width=100% height=1 border=0></center>
    Sayın: $adi<br>
    Defterimi imzaladığınız için teşekkür ederim</P>
```

```
    <p>Ana Sayfaya Dönmek için <A HREF=\"index.htm\"> tıklayın</A><BR>
    Mesajları okumak için<A HREF=\"mesajlar.php\"> tıklayın</A>
    </BODY>
    </HTML>
    ");
}
```

Programımızın bu bölümünde, ziyaretçiye teşekkür ediyoruz; ve isterse diğer mesajları okumak üzere tıkllayacağı köprüyü sağlıyoruz. Mesajların çok uzaması halinde bu köprüyü bir sayfanın üstünde, bir de altında vermekte yarar var. Şimdi, programımızın Form doldurulduktan sonra işleyen bölümünü yazmış olduk. Taa en başta yaptığımız $HTTP_POST_VARS dizi-değişkeninin değer içerip içermediğine ilişkin sınamanın olumsuz olması halinde, prıogramımız, işe buradan başlayacaktır:

```
else {
      echo ("
      <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=\"$PHP_SELF\" METHOD=\"POST\">
      <P><font size='3' face='verdana, Arial, Helvetica'>
      Defteri imzalamadan okumak için <a href='mesajlar.php'>
      burayı tıklayın</a></FONT></P>
      <P><font size='3' face='Arial, Helvetica'>
      Adınız, Soyadınız: <br><INPUT TYPE=\"TEXT\" NAME=\"adi\"><BR>
      Elektronik Adresiniz: <br><INPUT TYPE=\"TEXT\" NAME=\"adres\"><BR>
      Mesajınız:<BR>
      <TEXTAREA NAME=\"mesaj\" ROWS=10 COLS=30 WRAP=virtual></TEXTAREA>
      <BR><BR><BR>
      <INPUT TYPE=\"SUBMIT\" VALUE=\"Defteri imzala!\">
      <INPUT TYPE=\"RESET\" VALUE=\"Tümünü sil!\">
      </FORM>
      </BODY>
```

```
        </HTML>
        ");
}
?>
```

Bu bölümde çift-tırnak işaretlerini değiştirmek yerine, bunları ESC karakteri ile özel işaret değil, metin olarak ele alınacak hale getirdik. Programımız burada yine echo() ile düz ve basit bir HTML Formu oluşturuyor. Programın akış planını anlamış olmalısınız: ziyaretçi önce Form'u dolduracak ve gönder düğmesini tıklayacak.

Bu ACTION, Form'da INPUT alanlarındaki değerleri POST yöntemiyle yine kendisine gönderecek; böylece PHP, programı yeniden çalıştırırken, bu kez en baştaki şartın yerine geldiğini (yani $HTTP_POST_VARS dizi-değişkeninin değer içerip içermediğine ilişkin sınamanın olumlu sonuç verdiğini) görerek, işe baştan baştan başlayacaktır. Bu, INPUT etiketinin alanlarından oluşturulan değişkenlerden yararlanarak, ziyaretçiye bize ulaştırdığı bilgileri ve bizim teşekkür mesajımızı yollamaktan ibarettir.

Program bu şekliyle ne güvenlik önlemlerine sahip, yani kötüniyetli kişilerin INPUT alanlarına kötü niyetli kodlar yazmaları halinde bunları ayıklıyor; ne de ziyaretçinin boş Form gönderip göndermediğini sınıyor. Bu gibi işleri, PHP ile metin düzenlemeyi öğrendikten ve Düzenli İfadeler denen kavramı ile tanıştıktan sonra yapacağız.

Metin Düzenleme ve Düzenli İfadeler

Web programlarımızda bir ziyaretçinin bize ulaştırdığı Form bilgilerini ve bir veritabanı dosyasından aldığımız ham verileri yeniden düzenlememiz gerekir. PHP bunu yapabilmek için bize 50'den fazla fonksiyon sağlar. Bunlardan sık kullanmamız gerekenleri ele alacağız. Daha sonra, özellikle Unix işletim sistemine, Perl veya Javascript diline aşina olanların hatırlayacağı Düzenli İfadedeler denen metin ve sıralanış (pattern) eşleştirme araçlarının üzerinde duracağız. Bu iki araçla, elimize aldığımız berhangi metne istediğimiz biçimi vermek veya metnin içinden istediğimiz bölümü hemen hemen tamamen seçip biçimlendirmek ve kullanmak mümkün olur.

Temel Alfanümerik Fonksiyonları

PHP'nin metin olarak gördüğü değişkenlere alfanümerik (String) türü değişkenler adını vermiştik. PHP'nin alfanümerik fonksiyonları, bu tür değişkenlerin değerlerinin içinden bir bölümü alma, atma veya değiştirme imkanı sağlar. Programlarımızda sık kullanacağımız bir kaç fonksiyonu ele alalım. Bu fonksiyonların tam listesini bu kitapçığın örnek kodları arasında alfa_fonksiyonlar.htm adlı dosyada bulabilirsiniz. (Aşağıdaki örnekler de alfa_fonksiyonlarXX.php adlı dosyalarda bulunabilir.)

substr()

Bir alfanümerik değişkenin değerinin veya bir metnin tanımladığınız bölümünü verir. İkisi zorunlu, biri seçmeli üç parametre ile kullanılır. Yazılışı:

```
substr($degisken, başla, [boyut] );
```

Burada, $degisken yerine içinden bir bölümü seçmek istediğiniz metni tutan değişkenin adını; başla yerine seçme işleminin başlayacağı karakterin baştan itibaren

kaçıncıdan sonra olduğunu ve isterseniz boyut yerine de seçilmesini istediğimiz karakter sayısını yazarız. Kaç karakter seçileceğini gösteren boyut parametresini yazmazsanız, PHP değişkenin veya metnin sonuna kadar seçer. Örnek

```
$secilen = substr ("abcdef", 1);
```

Burada $secilen değişkeninin değeri "bcdef" olacaktır; çünkü birinci karakterden sonra seçime başlatıyoruz ve seçimin boyutunu belirlemiyoruz. PHP, verdiğimiz metnin sonuna kadar seçecektir.

```
$degisken = "Olumsuz düşünce ve mekanizmalar üzerine olumlu şeyler bina edilemez!";
$secilen = substr ($degisken, 8, 20);
```

Burada ise seçime 8'nci karakterden sonra başladığımız ve 20 karakterlik seçim yapılmasını belirttiğimiz için seçilen değer "düşünce ve mekanizma" olacaktır.

Başlanıç olarak verdiğiniz sayı negatif ise (önünde - işareti varsa) seçim işlemine değişkenin sonundan itibaren başlanır. Yukarıdaki substr() fonksiyonunu:

```
$secilen = substr($degisken, -9);
```

şeklinde yazarsak, dönen değer, "edilemez!" olacaktır.

trim()

Bir alfanümerik değişkenin değerinin baş ve son tarafındaki boşlukları atar. Yazılışı:

```
trim($degisken);
```

Bu fonksiyon değişken değerinin veya metnin başındaki ve sonundaki "\n", "\r", "\t", "\v", "\0", ve aralık işaretini boşluk sayar. Örnek:

```
$degisken = "\0   Olumsuz düşünce ve mekanizmalar üzerine olumlu şeyler bina edilemez!\n\r\t";
$secilen = trim ($degisken);
```

Burada seçilen metinde, baştaki boşluk, sondaki yeni satır, satır başı ve sekme karakterleri olmayacaktır.

chr()

Parametre olarak ASCII değerini belirttiğimiz karakteri sağlar. Örnek:

```
echo (chr(34);
```

Bu komutla Browser penceresinde çift-tırnak işareti görüntülenir.

ord()

Parametre olarak yazdığınız karakterin ASCII değerini sağlar. Örnek:

```
echo (ord("A");
```

Bu komutla Browser penceresinde 65 rakamı görüntülenir. bu fonksiyonun başlıca kullanım alanı, bir değişkenin değerini arzu ettiğimiz bir karakterin ACSII koduyla karşılaştırmak olabilir. Örneğin:

```
if ( ord($karakter) ==  60 { // karakter "<" işareti ise
            $karakter = "" ;
            }
```

koduyla, bir Form'dan gelen verileri karakter-karakter sınayabilir ve içinde normal olarak bulunmaması gereken, zararlı kod parçası olması ihtimali bulunan karakterleri ayıklatabiliriz.

strlen()

Bir alfanümerik değişkenin değerinin kaç karakter içerdiğini bildirir. Yazılışı:

```
strlen($degisken);
```

Bu fonksiyondan dönen değer tam sayıdır; aritmetik işlemlerde kullanılabilir. Örneğin, ziyaretçinin bir INPUT alanına yazdığı yazının uzunluğunun arzu ettiğimiz ölçülere uygun olup olmadığını sınamakta bu fonksiyondan yararlanabiliriz:

```
if ( strlen($degisken) >  30 { // girilen değer 30 karakterden uzun ise
            echo ("Lütfen 30 karakterden uzun yazmayınız!);
            }
```

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
```

number_format()

Türü sayı olan değişken değerlerini bin-basamakları şeklinde biçimlendirmekte kullanılır. Parametre olarak sayı içeren değişkenin adını, ondalık bölümün kaç haneli olacağını, ondalık hanesini ve binler basamaklarını ayırmakta kullanılacak karakteri kabul eder. Örnek:

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

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

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);
```

Düzenli İfadeler

Unix işletim sistemini geliştirenlerin bilişim teknolojisine kazandırdığı bir diğer değer biçilemez araç olan Düzenli İfadeler (Regular Expressions), belki de işaretlerinin karmaşıklığı sebebiyle programcıların fazla dokunmak istemedikleri bir alandır. Bu kavramla daha önce tanışmış olmayanlara, CGI veya Javascript kodlarında gördükleri,

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

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. Düzenli İfade oluşturarak, PHP'ye "İşte buradaki gibi sıralanmış karakterleri bul!" demiş oluruz. Düzenli İfade, PHP'ye verdiğimiz sıralanışa uygun metnin nerede ve kaçıncı kez bulunmasını istediğimizi de belirtir. Ayrıca, bu ifade ile, PHP'nin bulduğu metni (veya metinleri) ne yapmasını istediğimizi de söyleriz.

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. ^işareti, onu izleyen karakterin sadece alfanümerik değişkenin başında olması gerektiğine işaret eder. Bu deyimle, "Hakkı Öcal bugün buradaydı!" veya "Hakkıdır hakka tapan, milletimin istiklâl!" eşleşebilir; fakat "Anaların hakkı ödenmez!" eşleşmez; çünkü PHP açısından ^işaretinin anlamı, "Başlangıcında 'hakk' karakterlerinin sıralanıyla aynı sıralanıştaki alfanümerik değeri bul!" demektir.

```
edilemez$
```

Bu deyim ise PHP'ye "edilemez" ile biten bütün kelimeleri bulur. Sözgelimi, "Kötümser insanın yükselişinden bahsedilemez" cümlesi bu deyimle bulunabilir; fakat sonunda nokta olduğu için ".. edilemez." diye biten hiç bir cümle bu deyimle eşleşemez.

```
^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.

PHP'nin kullanılmaya hazır özel Düzenli İfade eşleştirme grupları da vardır:

[[:alpha:]] Herhangi bir harf

[[:digit:]] Herhangi bir rakam

[[:alnum:]] Herhangi bir harf veya rakam

[[:space:]] Herhangi bir boş karakter

[[:upper:]] Herhangi bir büyük harf

[[:lower:]] Herhangi bir küçük harf

[[:punc:]] Herhangi bir noktalama işareti

[[:xdigit:]] Herhangi bir Hexadecimal karakter. [0-9a-fA-F]

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. Sözgelimi bir ziyaretçimiz "@@@@.@@@" yazarsa, bu deyim bu değeri geçerli bir elektronik adres sayacaktır.

PHP programlarımızda ziyaretçilerimizin verdiği değerleri çeşitli bakımlardan sınamak ve seçmek mümkündür; ancak hiç bir zaman yazılanların doğruluğunu garanti edemeyiz. Fakat özellikle bir metinde bulunmaması gereken işaretlleri PHP'ye aratabiliriz. Bunu Düzenli İfade Fonksiyonları sağlar.

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. (Burada, yaptığı hayatı düzeltmesi yani geçerli bir elektronik posta adresi vermesini sağlayan yeni sayfanın, include komutu ile ziyaretçiye gönderildiğine dikkat edin. Bu programın yeni biçimi kitapçığın örnek kodları arasında kd_02.php adıyla bulunabilir.)

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. PHP'de güvenlik bölümünde bu konuda daha ayrıntılı bilgi bulacaksınız. 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. (Bu örneği içeren konuk defteri, bu kitapçığın örnek kodları arasında kd_03.php adıyla bulunabilir.)

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:

```
<?php
$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]

PHP ile Veritabanı

PHP'yi bir veritabanından veri çekecek, bunlarla HTML etiketlerinin içeriğini oluşturacak ve ziyaretçimizden alacağımız bilgileri bir veritabanına işleyecek işlemleri öğrenebilecek kadar biliyoruz artık. Bu kitapçığın baş tarafındaki kurma işlemlerini birlikte yaptıysak, şu anda PHP programlarını geliştirdiğiniz kişisel bilgisayardaki kişisel Web sunucusunda MySQL sürücüleriniz ve veritabanı yönetim programınız çalışıyor olmalı. Bir kere daha tekrar etmek gerekirse, PHP ile veritabanına dayanan sayfalar yapmak için mutlaka ve sadece MySQL sürücülerini kullanmak gerekmez. Bunun yerine sözgelimi Microsoft'un ODBC sürücülerini kullanabilirsiniz. Ancak Web sitenize evsahipliği yapan firmanın size ODBC desteği verip vermediğini öğrenmeniz gerekir. Aslında aynı şekilde, evsahibi firmanın MySQL desteği verip vermediğini de öğrenmek zorundasınız. (PHP ile MS-ODBC sürücülerini kullanabilmek için Andrew Stopford'un ASP Today sitesinde http://www.asptoday.com/articles/20000202.htm adresindeki yazısında daha geniş bilgi bulunmaktardır.)

Web sitemizde, HTML sayfalarımızı ve PHP programlarımızı neden veriya dayalı hale getirmeliyiz? Bu sorunun cevabı, "Başka türlü elde edilemeyecek bir otomasyon, ziyaretçilerimizle gerçek anlamıyla etkili biralışveriş sağlamak için!" diye yanıtlayabiliriz. Yukarıda, bir PHP programının ziyaretçiden Frm ile alacağı bilgileri, yine kendisine gönderebileceğini ve bu bilgileri işleyebileceğini gördük. PHP programımızın bu bilgileri bir veritabanına işlediğini ve ziyaretçi bir başka sayfaya gitmek istediğinde o sayfanın içeriğinin, ziyaretçinin birkaç saniye önce yaptığı tercihlere göre belirlendiğini düşünün. Bir başka örnek, Web sitemizdeki sayfalarda yer alan unsurların güncelleştirilmesi için her güncelleştirmede çok sayıda programı yeniden yazmak veya sayfayı yeniden yapmak zorunluğu varsa, bu program ve sayfaların içeriklerini bir veritabanından alıyor olması

halinde, sadece veritabanını güncelleştirmekle onlarca, belki de yüzlerce, HTLL ve PHP belgesini yeniden yazmış gibi olabiliriz.

Bu bölümde önce bir örnek veritabanı dosyası hazırlayacağız; daha sonra bunun içindeki verileri PHP programı ile sayfalarımıza katacağız. Bu dizinin ikinci kitabında Unix/Linux sistemlerinde veritabanı uygulamaları üzerinde daha geniş duracağız. Burada Windıows sisteminde bir MySQL uygulamasını ele almakla yetineceğiz.

MySQL Veritabanı

Gerçi bu kitapçığın konusu kendisi başlı başına bir dil olan SQL değil, ama burada sadece MySQL programı ile veritabanı oluşturmakla kalmayalım, bir anlamda SQL diline de çok kısa bir giriş yapalım. SQL çook geniş bir dildir. Burada yapabileceğimiz giriş ancak ilk alıştırmamaızı yapmamıza yetebilir.

SQL, (Structured Query Language, Yapısal Sorgu Dili) veritabanı endüstrisinin standart dilidir. Fakat HTML konusunda farklı firmaların sadece kendi Browserlarının tanıdığı türden etiketler geliştirmeleri gibi, SQL alanında da farklı firmaların ekleri ve dilde yaptıkları değişiklikler vardır.

MySQL, teknik tabiriyle sunucuda daemon olarak çalışır ve arzu eden programa, bildireceği (ve erişim hakkı bulunan) veritabanı dosyasından veri çekerek, sunar. Bir veritabanında farklı sayıda tablolar olabilir. Bir veritabanı dosyasında yer alan tablolarda, sütunlar ve satırlar vardır: bunların kesiştikleri yerlere biz verilerimizi yazarız. Bir SQL veritabanı dosyasında her bir kutunun ne tür veri kabul edeceği önceden belirlenir; dolayısıyla türü sayı olan verinin yazılacağı kutuya, alfanümerik değer yazılmaz. MySQL, sayısal verilerin türünü INT, belirli bir sınıra kadar değişen sayıda karakterden oluşacak verilerin türünü VARCHAR komutuyla belirler.

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 ('Muharrem','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 = "Şahika" ;
```

Bu komut, veritabanındaki bütün satırlarda, birinci sütundaki değerleri "Şahika" 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 = "Şahika" 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;
```

Vveritabanında 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 , 'Muharrem' , '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:

```
<HTML>
<TITLE>PHP ile Veri Örnegi</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>
<?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"));
?>
</BODY>
</HTML>
```

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'de Güvenlik

Günümüzde Internet'te ne kadar güvenlik önlemi alınsa, yeridir. Sadece iyi niyetli olmayan kişilere karşı değil, fakat daha çok ne yaptığını tam bilmeyen Web ziyaretçilerini de düşünmek ve PHP programlarınızın güvenliğini arttırmak zorundasınız.

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

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

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

Bir Form'un içerdiği bilgilerin denetimi ve doğrulanması iki yerde yapılabilir: istemci-tarafında, sunucu-tarafında. İstemci tarafında, yani ziyaretçiye göndereceğimiz HTML sayfasının içinde yer alan kodlarla yapacağımız denetim-doğrulama, hızlı çalışır; ve

ziyaretçi ile sunucu arasında iletişim kurulmadan önce yapılır; böylece ziyaretçinin sözgelimi gönder düğmesini tıkladıktan sonra çok beklemesi gerekmez. Ne var ki bu yöntemde denetim-doğrulama kodu ziyaretçiye gönderilmiştir; kötü niyetli kişi neyi denetlediğinizi görecek ve isterse bu denetimi kaldıracak size zararlı kod parçacıkları gönderebilecektir. Denetimin sunucu tarafında yapılması, belki biraz daha fazla zaman harcamayı gerektirir, fakat güvenlik açısından daha etkili olabilir. Elektronik posta adresi doğırulama ve mesajların içinden zararlı kodları ayıklama yöntemlerinden Düzenli İfadeler bölümünden söz ettik. Orada ele aladığımız örnek kodlar geliştirilerek ziyaretçilerin girdilerinde daha sıkı aramalar yapılabilir.

Parola ve SSL

PHP gibi, kendisi ziyaretçinin Browser'ına gönderilmeyen bir dille, sitelerimi parola ile korumak mümkün ve kolaydır. Sitenize sadece sizin vereceğiniz parola ile girilmesini istiyorsanız, ana sayfanın şöyle başlaması yeter:

```
<?php
if (!isset($parola) || $parola != "gizli_kelime") {
?>
<FORM ACTION="parola_igir.php" METHOD="POST">
Parola: <INPUT NAME="parola" TYPE"password"><BR>
<INPUT TYPE="submit">
</FORM>
<?php
}
else {
// Buraya gizlenen sayga girecek
?>
```

PHP ayrıca mevcut hemen hemen bütün sunucu kullanıcı kimlik doğrulama protokolleri ile uyumludur; ve sunucu yönetimi ile ilişki kurularak daha karmaşık kullanıcı kimlik doğrulama yöntemleri uygulanabilir. Bu yöntemler arasında Secure Socket Layer (SSL) yöntemlerini, ve OpenSSL ve ModSSL teknikleri sayabiliriz.

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)
```

Bu dizinin ikinci kitabında güvenlik konusuna daha geniş yer vereceğiz.

PHP Kaynakları

Güvenlik dahil, PHP konusunda başvurulacak kaynakların bir kısmı şöyle sıralanıyor. Ayrıca bu kitapçığın örnek kodları arasında, PHP Hosting Companies Search the Directory.htm adlı belgede çık sayıda kurum ve firmasın listesini bulabilirsiniz:

http://www.php.net/FAQ.php

http://www.php.net/manual/config-odbc.html

http://www.techdevelopers.com

http://www.lilali.net

http://www.sqlcourse.com/

ttp://www.apache.org/info/apache_users.html

W/O: 214175

W/O: 145821

W: 167316