PHP Programlamaya Giriş

Doruk Fişek

(dfisek@fisek.com.tr)

Seminer Notları -- http://seminer.linux.org.tr

PHP NEDİR?

- PHP, özellikle web uygulamaları geliştirilmesinden kullanılan, HTML içine gömülebilen açık kaynak kodlu bir script dilidir.
- C, Java ve Perl dillerinin karışımı bir söz dizimine (syntax) sahiptir.
- Ağustos 2003 itibarıyla internette 13 milyonun üzerinde web sitesinde kullanmaktadır.
- Zaman içinde artan popülerliği ile pratikte bir script dili olmaktan çıkarak bir programlama dili haline gelmiştir.

PHP'nin Gelişimi

- 1994 sonbaharında Rasmus Lerdorf, kendi web sayfasındaki özgeçmişine kimlerin eriştiğini takip etmek için basit birtakım Perl scriptleri kullanıyordu. Daha sonra gereksinimleri büyüdükçe, daha geniş bir C yazılımı haline getirdi -- veritabanına bağlanabiliyor ve basit dinamik web uygulamaları yapılabiliyordu. Bunları 1995'te PHP/FI : Personal Home Page Tools / Forms Interpreter adıyla yayınladı.

PHP'nin Gelişimi

- 1997'de PHP/FI 2.0 duyuruldu. Binlerce kullanıcı ve 50000 alan adına ulaştı. Koda katkıda bulunan birçok insan vardı ancak hala tek bir insan tarafından geliştirilen büyük bir projeydi.

PHP'nin Gelişimi

- 1997 sonlarında bir elektronik ticaret uygulaması geliştirmek isteyen Zeev Suraski ve Andi Gutmans, PHP/FI 2.0'ın kısıtlarını farkederek baştan aşağı tekrar yazdılar. Orjinal projenin yaratıcısı Rasmus Lerdorf da onlara katıldı ve PHP 3.0, PHP/FI projesinin resmen devamı oldu. İsim değişikliğine de giden proje PHP : Hypertext Preprocessor adını aldı.

PHP'nin Gelişimi

- 9 aylık bir test süreci sonunda Haziran 1998'de PHP 3.0 piyasaya çıktı. PHP o günden sonra ciddi bir yaygınlaşma ve gelişme sürecine girdi. PHP 3'ün getirdiği modüler yapı, birçok programcının koda kendi istedikleri özellikler için modüller yazmasını sağladı. PHP 3, ideal bir web programlama dili haline geldi.

PHP'nin Gelişimi

- PHP 3.0 çıktıktan bir süre sonra Zeev Suraski - Andi Gutmans ikilisi PHP'yi oluşturan çekirdeğin yeniden yazılmasına başladılar. Daha yüksek performans, daha modüler bir yapı, daha karmaşık programlar ve Linux/Apache web ortamından bağımsız da çalışabilen bir PHP için yola çıkıldı.

PHP'nin Gelişimi

- Yeni yazılan motora Zend (Zeev ve Andi isimlerinin birleşimi) ismi verildi. Mayıs 2000'de piyasaya çıkan PHP 4.0 ile beraber artık PHP sadece bir script dili olmaktan çıkıp bir programlama dili olma yolunda emin adımlarla ilerliyordu.

PHP 4'te PHP birçok yeni fonksiyona sahip oldu ve birçok yazılım ile bağlantısı sağlandı. Bunun yanı sıra; PEAR, Smarty, PHP-GTK gibi birçok yan proje geliştirilmeye başlandı.

- Halen geliştirilmekte olan PHP 5'in özellikle nesneye yönelik programlama (OOP) konusunda ciddi gelişmeleri de beraberinde getirmesi bekleniyor.

PHP ile Neler Yapılabilir?

- PHP, sunucu tarafında programlamaya odaklanmış bir dildir. Web üzerinde çalışan diğer programlama dilleri gibi form verisi alabilir, dinamik sayfa içeriği üretebilir, çerez (cookie) alıp verebilir.
- Web üzerinden çalıştırılabilmesinin yanı sıra, komut satırından da çalıştırılabilir. Özellikle zaman ayarlı / belirli aralıklarla çalıştırılması gereken programlar için kullanılabilir.
- PHP ile, sadece HTML çıktı vermekle kısıtlı değilsiniz. PHP programlarınızla çalışma esnasında aldığınız verilere göre resimler, PDF dosyaları, hatta Flash filmleri yapabilirsiniz.

PHP ile Neler Yapılabilir?

- PHP'nin en güçlü ve en çarpıcı özelliklerinden biri birçok veritabanı sunucusu desteklemesidir. Veritabanı kullanan bir web sayfası yazmak çok kolaydır. 20 farklı veritabanı sunucusuna direk bağlanabilirken, diğerleri için de ODBC üzerinden bağlantı olanağı tanır.
- PHP ile veritabanı dışındaki servislerle de kolay iletişim kurabilirsiniz. LDAP, SNMP, POP3, SMTP, IMAP, HTTP ve daha birçok protokol desteği vardır. Desteklenmeyen protokoller için ise, raw network soketi açarak iletişim kurmak mümkündür.
- Java objelerini alarak saydam olarak PHP objeleri olarak kullanabilirsiniz.

PHP ile Neler Yapılabilir?

- Güçlü metin işleme ve ayrıştırma fonksiyonlarının yanı sıra düzenli ifadeler (regular expressions) de kullanılabilir.

- Takvim çevirimlerinden SSL şifrelemeye, çeşitli sıkıştırma algoritmalarından süreç (process) yönetimine kadar birçok değişik konuda PHP fonksiyon kütüphanelerini bulmak mümkün.

Temel Söz Dizimi

- PHP söz dizimi, <?php ifadesi ile başlar ve ?> ifadesi ile biter. Öncesinde ya da sonrasında HTML dilinde satırlar yer alabilir.

- Bir komutun bittiğini noktalı virgül ; işareti belirler.

- HTML içine PHP Gömülmesi

Temel Söz Dizimi

Örnek 1 :

```
<html>
<head>
<title>Merhaba Dünya</title>
</head>
<body>
<?php
echo "Merhaba Dünya!";
?>
</body>
</html>
```

Temel Söz Dizimi

Örnek 2:

```
<?php
if ($ifade) {
?>
<i>Doğru</i>
<?php
} else {
?>
<i>Yanlış</i>
<?php
}
?>
```

Temel Söz Dizimi

PHP içine HTML Gömülmesi

Örnek 1:

```
<?php
echo "<html><head><title>Merhaba Dünya</title></head>
<body>Merhaba Dünya! </body></html>";
?>
```

Temel Söz Dizimi

PHP içine HTML Gömülmesi

Örnek 2:

```
<?php
if ($ifade)
echo "<i>Doğru</i>";
else
echo "<strong>Yanlış</strong>";
?>
```

Temel Söz Dizimi

Yorum (Comment)

- PHP; C, C++ ve Unix kabuğu stillerinde yorum (comment) eklenmesine olanak tanır.

- // ve # kullanıldığında, bu işaretten sonra satır sonuna kadar yer alan kısı yorum kabul edilir.

- /* işareti ile */ arasındaki tüm satırlar yorum olarak kabul edilir.

Temel Söz Dizimi

Yorum (Comment)

Örnek :

```
<?php
echo "Ses bir-iki..."; // Gökten üç yorum düşmüş...
/* Birden fazla satırdan
oluşan bir yorum satırı */
echo "Sesim geliyor mu?";
echo "Gelmiyor galiba."; # Bu da son yorum şekli
?>
```

Veri Tipleri

- PHP dört temel veri tipinden oluşur :

- Boolean
- Integer
- Float (Double)
- String

İki tane birleşik (birçok veriden oluşan) veri tipi :

- Dizi (Array)
- Nesne

Veri Tipleri

İki tane de özel veri tipi :

- Kaynaklar : Dış bir kaynak ile kurulan ilişkiyi belirten özel bir değişkendir. Veritabanı bağlantısı, açılmış bir dosya, ...

- NULL : Değeri olmayan değişkenler bu gruba girer. Bir değişken üç durumda NULL'dır :

• Sabit olarak NULL değeri atanmıştır.

• Henüz herhangi bir değer atanmamıştır.

• unset() fonksiyonu ile değeri yok edilmiştir.

Bir değişkenin veri tipi çoğunlukla programcı tarafından belirtilmez, programın çalışması esnasında değişkenin içeriğine göre PHP tarafından ayarlanır.

Değişkenler

- Değişkenler PHP'de dolar ($) işareti ile ifade edilir. Dolar işaretinden sonra değişkenin ismi gelir ($degisken).

- Değişken isimlerinde küçük-büyük harf ayrımı yapılır ($degisken != $Degisken).

- Değişken isimleri; harfler, rakamlar ve alttançizgi (_) işaretinden oluşabilir.

- Harfler; A-Z, a-z ve 127 ile 255 arasındaki ASCII karakterlerinden oluşabilir.

- Değişken isimleri bir rakam ile başlayamaz.

Değişkenler

Örnek :

```
<?php
$degisken = "Koray";
$degisKen = "Ekin";
echo "$degisken, $degisKen";

$5kardesler = 'El'; // geçersiz, rakam ile başlıyor
$_yaz1ilim = 'Matematik'; // geçerli, alttan çizgi ile başlıyor
$ütü = 'prize takılı'; // geçerli, 'ü' 129 numaralı ASCII karakteri
?>
```

Değişkenler

- Değişkenlere normalde her zaman değer atanır. Farklı bir yöntem olarak PHP, bir değişkene değeri olan başka bir değişkene referans olarak atanmasına olanak tanır. Birbirine bağlanan iki değişkenden birinin değeri değiştiğinde otomatik olarak diğer değişkenin değeri de aynı şekilde değişir.

Örnek :

```
<?php
$a = 'Koray';
$b = &$a;
$b = 'Ekin';
echo $b;
echo $a;
?>
```

Değişkenler

- String değişkenlerinin atanmasında iki farklı yöntem vardır.

```
<?php
$a = 'Koray';
$b = "$a Löker";
$c = '$a Löker';
echo "$a, $b, $c";
?>
```

Değişkenler

* Çift tırnak kullanıldığında, PHP string'in içinde başka değişkenlerin bulunup bulunmadığına bakar; eğer bulursa onları kendi değerleri ile değiştirir. Tek tırnak kullanıldığında ise PHP, içeriğine bakmaksızın olduğu gibi değişkeni atar.

Değişkenler

- Çift tırnak kullanılarak yapılan bir atamada, değişkene atanan değerin içerisinde çift tırnak işareti yerleştirilmek istenirse, çift tırnak işaretinin önüne \ (ters bölü) işareti yerleştirilmelidir. Aynı durum tek tırnak kullanılarak yapılan bir atama ile atanan değerde tek tırnak bulunması durumunda da geçerlidir.

```
<?php
$a = 'Koray\'ın bisikleti';
$b = "Zor bir \"test\" oldu";
echo "$a<br>$b";
?>
```

Değişkenler

- Bazı durumlarda değişkenin isminin de değişken olması işimize gelir.

```
<?php
$a = "koray";
$$a = "löker";
echo "$a$koray";
echo "$a${$a}";
?>
```

Önceden Tanımlanmış Değişkenler

- PHP, çeşitli önceden tanımlanmış, duruma göre değerler alan değişkenlere sahiptir. Bu değişkenler türlerine ilişkilendirilmiş dizilerde bulunurlar. Bunlardan bazıları :

• $_GET : HTTP GET metodu ile programa iletilen değişkenler

• $_POST : HTTP POST metodu ile programa iletilen değişkenler

• $_SERVER : Web sunucu, web istemcisi ve programın çalıştığı ortamla ilgili bilgileri içeren değişkenler

Önceden Tanımlanmış Değişkenler

- $_COOKIE : HTTP çerezleri (cookie) tarafından iletilen değişkenler

- $_SESSION : Oturum (session) tarafından iletilen değişkenler

- $_FILES : HTTP dosya gönderme (upload) işlemi sırasında belirlenen değişkenler

- $GLOBALS : Ana programda var olan değişkenler

Bazı $_SERVER Değişkenleri

- 'PHP_SELF' : Şu anda çalışmakta olan sayfanın kök dizine göre dosya ismi. Örneğin, http://linux.fisek.com.tr/dfisek/test.php ise çalışmakta olan sayfa, değeri /dfisek/test.php olacaktır.

- 'HTTP_ACCEPT_LANGUAGE' : Web istemcisinin kabul ettiği / tercih ettiği dil. Örneğin 'tr'.

Bazı $_SERVER Değişkenleri

- HTTP_REFERER : Web istemcisinin kayıtlarına göre bulunulan sayfaya hangi sayfadan gelindiği. Bazı istemciler bu değişkeni hiç vermeyebilir, bazıları da kullanıcılarına bu değeri elle düzeltmek için seçenek sunar.

- HTTP_USER_AGENT : Bağlanan kullanıcının kullandığı web istemcisinin ismi.

- REMOTE_ADDR : Kullanıcı sayfaya bağlandığı bilgisayarın IP numarası.

Bazı $_SERVER Değişkenleri

- PHP_AUTH_USER : HTTP doğrulaması için web istemcisinin gönderdiği kullanıcı ismi.

- PHP_AUTH_PW : HTTP doğrulaması için web istemcisinin gönderdiği şifre.

Değişkenlerin Kapsamı

- Standart olarak her değişken sadece kendi kapsamında (scope) geçerlidir. O kapsamın dışına değeri aktarılmaz.

```
<?php
$a = 5; // global kapsam

function test()
{
echo "a : $a"; // ilgili fonksiyonun kapsamı
$b = 8;
}

test();
echo "<br>b : $b";
?>
```

Değişkenlerin Kapsamı

- Global ve static tanımları bu kapsamları gereksinimlerimize göre esnetmemizi sağlar.

```
<?php
$a = 5;
$b = 8;

function toplagel()
{
global $a, $b;


$b = $a + $b;
}

toplagel();
echo $b;
?>
```

Değişkenlerin Kapsamı

- Global ile değişkenleri tanımlamak yerine önceden tanımlanmış değişkenlerden $GLOBALS'ı da kullanabiliriz.

```
<?php
$a = 5;
$b = 8;

function toplagel()
{
$GLOBALS[b] = $GLOBALS[a] + $GLOBALS[b];
}

toplagel();
echo $b;
?>
```

Değişkenlerin Kapsamı

- Static tanımı ile bir kapsam içinde tanımlanan değişkenin, kapsam dışına çıkıldığında değerini kaybetmemesini ve daha sonra tekrar o kapsama girildiğinde aynı değeri devam ettirmesini sağlar. Özellikle birden fazla kez çağırılan veya kendi kendini yineleyen (recursive) fonksiyonlarda yararlıdır.

```
<?php
function test()
{
static $a = 0;

$a++;
echo $a;
if ($a < 10)
test();
$a--;
echo $a;
}
?>
```

SABİTLER

- define() fonksiyonu kullanılarak tanımlanırlar.
- Bir sabit bir kez tanımlandıktan sonra, değeri programın çalıştırılması boyunca değiştirilemez.
- Sabitler sadece temel veri tiplerinden (boolean, integer, float, string) oluşabilir.
- Sabitler tanımlanırken önlerinde bir dolar işareti bulunmaz.

SABİTLER

- Sabitler kapsam kurallarından bağımsız olarak programın herhangi bir yerinden erişilebilirler.

- Geleneksel olarak sabit isimleri tamamen büyük harfle yazılır.

```
<?php
define(SPOR, Futbol);
echo SPOR;
?>
```

TEMEL OPERATÖRLER

- Aritmetik Operatörleri : Sayısal değerlerle toplama(+), çıkarma(-), çarpma(*) ve bölme(/) işlemleri yapılabilir. İşlemlerin önceliğini belirlemek için parantez () kullanılabilir.

```
<?php
$a = (9*(1+3))/2
$a = $a - 5
?>
```

TEMEL OPERATÖRLER

* Bir değişkenin değerini arttırmak/azaltmak için daha basit ifadeler de kullanılabilir.

```
<?php
$a++; // $a = $a +1
$a--; // $a = $a - 1
$a += 8; // $a = $a +8
?>
```

TEMEL OPERATÖRLER

* Dört işlem dışında, mod işlemi de % işareti ile yapılabilir. $a % $b, $a'nın $b'ye bölümünden artan sayıyı verir.

- Atama Operatörü : Temel atama operatörü '=' dir. Genel kanının aksine, iki değişkenin eşit olduğunu göstermez. Sağdaki değişkenin değerinin soldaki değişkene atandığını ifade eder.

TEMEL OPERATÖRLER

- Karşılaştırma Operatörleri : İki değeri karşılaştırmak için kullanılırlar.

İFADE	ANLAMI
$a == $b	Eşit mi?
$a === $b	Aynısı mı?
$a != $b	Eşit değil mi?
$a <> $b	Eşit değil mi?
$a !== $b	Aynısı değil mi?
$a < $b	Daha küçük mü?
$a > $b	Daha büyük mü?
$a <= $b	Daha küçük ya da eşit mi?
$a >= $b	Daha büyük ya da eşit mi?

TEMEL OPERATÖRLER

* Aynılık, iki değişkenin eşit olmasının yanı sıra veri tiplerinin de aynı olmasını gerektirir.

- String Opeatörleri : İki string'i birleştirmek için nokta (.) operatörü kullanılır.

```
<?php
$a = 'Koray';
$b = 'Löker';
$c = $a . ' ' . $b;
?>
```

TEMEL OPERATÖRLER

Mevcut bir string'e ekleme yapılmak isteniyorsa, sayılarda kullandığımız basit ifadeye başvurulabilir.

```
<?php
$a = 'Koray';
$a .= ' Löker';
?>
```

TEMEL OPERATÖRLER

- Mantıksal Operatörler

İfade	İsim	Sonuç
$a and $b	AND	$a ve $b doğruysa doğru
$a && $b	AND	$a ve $b doğruysa doğru
$a or $b	OR	$a ya da $b doğruysa doğru
$a || $b	OR	$a ya da $b doğruysa doğru
!$a	NOT	$a doğru değilse doğru
$a xor $b	XOR	$a ya da $b doğruysa, ama her ikisi birden doğru değilse doğru

Denetim Yapıları - if

- *if*, birçok dilde olduğu gibi PHP'de de çok önemli bir deyimdir. PHP, C'ye benzer bir *if* yapısı sunar :

```
if (ifade)
deyim;
```

- İfade, Boole olarak değerlendirilir. Eğer ifade doğru ise, PHP deyimi uygulayacaktır. Eğer ifade yanlışsa deyimi dikkate almayacaktır.

Denetim Yapıları - if

- Çoğu zaman duruma bağlı olarak birden fazla deyimin birden uygulanmasını isteyebiliriz. Bu durumda deyimleri küme parantezleri ile gruplayabiliriz.

```
<?php
if ($a != $b)
{
$a = $b;
echo $a;
}
?>
```

- İçiçe birçok *if* deyimi yerleştirilebilir.

Denetim Yapıları - else

- Belirli bir koşula uyulduğunda bir deyimin, koşula uyulmadığında ise başka bir deyimin yürütülmesini istiyorsanız; iki ayrı *if* deyimi kullanmanız gerekmiyor.

```
<?php
if ($a > $b)
echo "$a, $b 'den büyüktür";
else
echo "$a, $b 'den küçüktür";
?>
```

- *else* bloğunda bulunan ifadeler, sadece *if* deyiminin sonucunun yanlış olması durumunda yürütülür.

Denetim Yapıları - elseif

- *Elseif* tam anlamıyla *else* ve *if* deyimlerinin birleşiminden oluşan bir deyimdir. *else* deyimi ve hemen arkasından *if* deyimi kullanacaksanız, bunun yerine *elseif* kullanabilirsiniz.

```
<?php
if ($a > $b)
echo 'a değeri b değerinden daha büyüktür';
elseif ($a == $b)
print 'a değeri b değerine eşittir.';
?>
```

Denetim Yapıları - while

```
while (ifade)
   deyim;
```

- İfade doğru olduğu sürece, yerleştirilmiş olan deyim(ler)i tekrar tekrar yürütür.

- İfadenin doğruluğu, döngünün başında kontrol edilir. Eğer daha en başta ifadenin boole değeri yanlissa, deyim bir kez bile yürütülmeyebilir.

Denetim Yapıları - while

- Döngü içerisinde, ifadenin boole değerini değiştiren deyim(ler) olmalıdır. Aksi takdirde program sonsuz bir döngüye girebilir.

```
<?php
$i = 1;
while ($i <= 10)
{
$i++;
echo "$i<br>";
}
?>
```

Denetim Yapıları - do..while

• do..while döngüleri, while döngülerine çok benzer biçimde çalışır. Aralarındaki tek fark, while'da döngünün başında ifadenin doğruluğunun kontrol edilmesi, do..while döngüsünde ise döngü sonunda kontrol etmesi.

- do..while döngüsünde, ilk yineleme mutlaka gerçekleşecektir.

```
<?php
$i = 1;
do
{
$i++;
echo "$i<br>";
} while ($i <= 10)
?>
```

Denetim Yapıları - for

for (ifade1; ifade2; ifade3)
  deyim

- ifade1, for döngüsünun başlangıcında bir defaya mahsus olmak üzere yürütülür.
- ifade2, döngünün her yinelenmesinde bu ifade tekrar değerlendirilir. Eğer doğru ise döngü devam eder, yanlış ise döngü durdurulur.
- ifade3, for döngüsü devam ettiği sürece yinelenerek yürütülür.

Denetim Yapıları - for

- Bu ifadelerin herhangi biri boş olabilir. ifade2'nin boş olması durumunda döngü sonsuza dek çalışacaktır (PHP öntanımlı olarak ifadeyi doğru kabul edecektir).

```
<?php
for ($i = 1; $i <= 10; $i++)
echo $i;
?>
```

Denetim Yapıları - foreach

- Diziler için for döngüsü olarak da adlandırılabilir.
- Diziler dışında başka değişkenler için kullanılamaz.

```
foreach ($dizi as $degisken)
echo "Dizi elemanı: $degisken<br>\n";
```

Denetim Yapıları - break

- Yürütülmekte olan for, foreach, while, do..while veya switch denetim yapısını durdurur.
- break'e opsiyonel olarak kaç tane içice yapının durdurulacağı parametresi verilebilir. Bu değer öntanımlı olarak 1'dir.

```
<?php
$i = 7;


while ($i--)
{
echo "$val<br>\n";
if ($i == '2')
break;
}
?>
```

Denetim Yapıları - continue

- continue, döngü yapılarında mevcut yinelemenin eş geçilerek bir sonraki yineleme ile döngünün yürütülmesini devam edilmesini sağlar.

- Aynı break deyiminde olduğu gibi içice kaç döngünün 'ileriye sarılacağını' opsiyonel olarak belirtebilirsiniz. Öntanımlı değeri 1'dir.

```
<?php
$i = 7;
while ($i--)
{
if ($i % 3)
continue;

echo "$i<br>";
}
?>
```

Denetim Yapıları - switch

- switch deyimini, aynı ifade üzerine uygulanan bir seri if deyimine benzetebilirsiniz.

- Çeşitli durumlarda, bir değişkenin aldığı farklı değerlere göre farklı kod parçacıkları işleme koymak isteyebilirsiniz. İşte switch bunun için biçilmiş kaftandır.

```
<?php
$semt = 'Sihhiye';

if ($semt == 'Gaziosmanpasa')
echo 'Luks semt';
elseif ($semt == 'Batikent')
echo 'Metro serinletir';
elseif ($semt == 'Sihhiye')
echo 'Tren kalkiyoor...';
```

Denetim Yapıları - switch

```
switch ($semt)
{
case 'Gaziosmanpasa':
echo 'Luks semt';
break;
case 'Batikent':
echo 'Metro serinletir';
break;
case 'Sihhiye':
echo 'Tren kalkiyoor...';
break;
default:
echo 'Kizilay';
}
?>
```

- break deyimi, switch denetim yapısının önemli bir parçasıdır. break deyimi kullanılmazsa, switch eşleyen değeri bulduktan sonra sona ermez ve diğer ifadeleri de yürütür.

Denetim Yapıları - return

- Bir fonksiyon içinden çağrılırsa, fonksiyonun yürütülmesini durdurur ve fonksiyonun değeri olarak kendisine verilen parametreyi dondurur.

- Eğer global kapsamda çağrılırsa, yürütülmekte olan script'in yaşamını sona erdirir.

```
return 5;
```

Denetim Yapıları - require, include

- Başka bir dosyayı çalışmakta olan script'in içine çalıştırıldığı yerden itibaren katar.

- require ile katılmaya çalışılan dosya eğer yerinde yoksa, script hata mesajı verir ve çalışmayı durdurur.

- include ile katılmaya çalışılan dosya eğer yerinde yoksa, script bir uyarı mesajı verir (çoğu kez görünmeyebilir PHP ayarlarına bağlı olarak) ve çalışmaya devam eder.

- Bir dosya, başka bir dosyanın içine katıldığı zaman; ikisi (ya da daha fazlası) sanki tek bir bütün dosyaymış gibi davranır.

Denetim Yapıları - require_once, include_once

- require ve include 'dan farklı olarak, eğer eklenmek istenen dosya daha önce script'e eklendiyse; tekrar eklenmeyecektir.

- Script'in belirli bir bölümü birden fazla kez yürütülecekse, çeşitli problemlerin önüne geçmek için kullanılır (fonksiyonların tekrar tanımlanması, değişkenlerin değerlerinin tekrar atanması ve benzeri).

Fonksiyonlar

- function test ($parametre_1, $parametre_2, ... , $parametre_n)

{

echo "Test fonksiyonu";

return 1;

}

- Bir koşullu ifadenin içinde tanımlanmadığı sürece; bir fonksiyonun çağrılmadan önce tanımlanmış olması gerekmez.

Fonksiyonlar

- Fonksiyon isimlerinde büyük-küçük harf ayrımı yapılmaz. Ancak ne şekilde tanımlandıysa o şekilde çağrılması her zaman için yararlıdır :-)

- Bir fonksiyon için birden fazla değişken sonuç olarak döndürülemez. Ancak birleşik bir değişkene (örneğin dizi), birden fazla değişken yerleştirilerek topluca sonuç olarak döndürülebilir.

Kaynaklar

- PHP EL KİTABI -- http://tr.php.net/manual/en/
- http://www.devshed.com/Server_Side/PHP
- http://www.hotscripts.com/PHP/
- http://www.php.org.tr
- http://www.phpbuilder.com
- http://php.resourceindex.com
- http://www.wdvl.com/Authoring/Languages/PHP/
- http://www.weberdev.com
- http://codewalkers.com
- http://px.sklar.com
- http://www.webmasterbase.com

SORULAR?