BÖLÜM 1

İLERİ WEB PROGRAMLAMAYA GİRİŞ

ANAHTAR KAVRAMLAR

Tarih/Saat, Dosya İşlemleri, Çerez/Oturum, Filtreler, JSON, İstisnalar

ÖĞRENME HEDEFLERİ

1. Normalizasyon ve standardizasyon işlemlerini yapabilir.
2. Karar matrisi yapısını tanımlayabilir.
3. Çok kriterli karar vermede gerekli olan temel yöntemleri açıklayabilir.
4. TOPSIS yöntemi ile karar verme hesaplaması yapabilir.
5. Analitik Hiyerarşi Süreci yöntemi ile kriter ve alternatif ağırlıklarını hesaplayabilir.

GİRİŞ

İnternet programcılığı II dersimiz İnternet Programcılığı I dersindeki temel komut ve işlevlerinin üzerine inşa edilmiş ileri seviye web programlama bilgilerini barındırmaktadır. Bu bağlamda ilk bölümümüzde PHP'de tarih/zaman işlemlerine, dosya üzerinde yapılan dahil etme, açma, okuma, oluşturma, yazma ve yükleme süreçlerine, çerez ve oturum kullanmaya, filtrelere, işlevlerde geri çağırmaya, verileri depolamak ve değiştirmek için kullanılan JavaScript Nesne Gösterimi'ne (JSON) ve son olarak bir PHP betiğinin hatasını ve beklenmeyen davranışını tanımlayan istisnalar konusuna değinilmektedir. Ayrıca her bir konu için PHP betik örnekleri verilmektedir. Söz konusu örnekleri sizlerinde yazıp çalıştırmanız faydanıza olacaktır.

TARİH VE ZAMAN

PHP'de date() işlevi tarih ve saat için kullanılmaktadır. Zaman damgasını daha okunabilir bir tarih ve saate biçimlendirir. Söz dizimi:

date(biçin,zaman damgası)

Burada biçim zaman damgası kalıbını belirlemek için kullanılır ve date() işlevinin zorunlu bir parametresidir. Zaman daması ise seçimlik bir parametre olup, mevcut tarih ve zaman bilgisini getirmektedir. Tarihlerde kullanılan bazı karakterler vardır:

- d - Ayın gününü temsil eder (01-31)
- m - Bir ayı temsil eder (01-12)
- Y – Dört haneli bir yılı temsil eder
- l (küçük harf 'L') - Haftanın gününü temsil eder

Ayrıca tarihlere ek biçimlendirme eklemek için karakterlerin arasına "/", "." veya "-" gibi diğer karakterler de eklenebilir. Aşağıdaki örnek, bugünün tarihini üç farklı şekilde biçimlendirmek için yazılmıştır:

```php
<?php

  echo "Bugün " . date("d/m/Y") . "<br>"; // Bugün 24/02/2024
  echo "Bugün " . date("d.m.Y") . "<br>"; // Bugün 24.02.2024
  echo "Bugün " . date("d-m-Y") . "<br>"; // Bugün 24.02.2024
  echo date("l"); // Günün İngilizce adı döner
?>
```

Gün ve tarih bilgisi İngilizce olarak dönmektedir. Eğer Türkçe olarak bir sonuç elde etmek istiyorsak, setlocale() ve strftime() işlevlerini kullanmamız gerekir:

```php
 setlocale(LC_TIME, 'tr_TR');
 // ya da setlocale(LC_TIME, 'tr_TR.UTF-8');
 echo strftime('%e %B %Y %A %H:%M:%S');
 // Örnek Sonuç: 24 Şubat 2024 Cumartesi 02:05:42
```

Telifleri gösterirken de date() işlevi tercih edilmektedir:

```php
&copy; 2024-<?php echo date("Y"); // © 2010-2024 ?>
```

Zaman gösterimlerinde de kullanılan karakterler vardır:

- H - Bir saatin 24 saatlik biçimi (00'dan 23'e)
- h - Başında sıfırlar bulunan bir saatin 12 saatlik biçimi (01'den 12'ye)
- i - Başında sıfır bulunan dakikalar (00 - 59)
- s - Başında sıfır bulunan saniyeler (00 - 59)
- a - Küçük Harf Ante meridiem ve Post meridiem (am veya pm)

Aşağıdaki örnekte zamanın karakterleri kullanılarak geçerli saatin belirtilen biçimde çıktısı alınmaktadır:

```php
<?php
   echo "Saat " . date("h:i:sa"); // Saat 07:51:27pm
?>
```

Saat Dilimini Alma

Kodunuzdan aldığınız zaman bilgisi yanlışsa, sunucu başka bir ülkede veya farklı bir saat dilimindedir. Bu durumda saat dilimini ayarlayabilirsiniz. Aşağıdaki örnek, saat dilimini "Europe/Istanbul" olarak ayarlar ve ardından geçerli saati belirtilen biçimde verir:

HATIRLATMA: PHP date() fonksiyonunun sunucunun geçerli tarih/saatini döndüreceğini unutmayın!

```php
<?php
  date_default_timezone_set("Europe/Istanbul");
  echo "Saat " . date("h:i:sa"); // Saat 10:57:14pm
?>
```

mktime() ile Tarih Oluşturun

date() işlevindeki isteğe bağlı zaman damgası parametresinin bir zaman damgasını belirttiği daha önce anlatılmıştı. Eğer bu değer girilmezse, yukarıdaki örneklerde olduğu gibi sistemin güncel tarih ve saati kullanılacaktır.

PHP mktime() işlevi bir tarih için Unix zaman damgası olarak bilinen bir değer döndürür. Unix zaman damgası ise, Unix Dönemi (1 Ocak 1970 00:00:00 GMT) ile belirtilen zaman arasındaki saniye sayısını içerir. mktime() işlevinin söz dizimi:

mktime(saat, dakika, saniye, ay, gün, yıl)

şeklindedir. Aşağıdaki örnek, mktime() işlevindeki bir dizi parametreden date() işleviyle birlikte bir tarih ve saat oluşturur:

```php
<?php
  $d=mktime(11, 14, 54, 8, 12, 2024);
  // Oluşturulan tarih 12-08-2024 11:14:54am
  echo "Oluşturulan tarih " . date("d-m-Y h:i:sa", $d);
?>
```

strtotime() ile Bir Dizeden Tarih Oluşturma

PHP strtotime() işlevi, insanlar tarafından okunabilen bir tarih dizesini Unix zaman damgasına (1 Ocak 1970 00:00:00 GMT'den bu yana geçen saniye sayısı) dönüştürmek için kullanılır. Söz dizimi:

strtotime(zaman, şimdi)

Aşağıdaki örnek strtotime() işleviyle bir tarih ve saat oluşturur:

```php
<?php
  $d=strtotime("10:30pm April 15 2024");
  // Oluşturulan tarih 15-04-2024 10:30:00pm
  echo "Oluşturulan tarih " . date("d-m-Y h:i:sa", $d);
?>
```

PHP bir dizeyi tarihe dönüştürme konusunda oldukça faydalıdır, bu nedenle çeşitli değerler girebilirsiniz. Ancak strtotime() işlevi yeterince mükemmel değildir, bu nedenle oraya koyduğunuz dizeleri kontrol etmeyi unutmamak gerekir. Aşağıda bu duruma örnek verilmektedir:

```php
<?php
  $d=strtotime("tomorrow"); // yarının zaman bilgisi döner
  echo date("d-m-Y h:i:sa", $d) . "<br>";

  $d=strtotime("next Saturday"); //gelecek cumartesinin bilgisi döner
  echo date("d-m-Y h:i:sa", $d) . "<br>";

  $d=strtotime("+3 Months"); // 3 ay sonrasının zaman bilgisi döner
  echo date("d-m-Y h:i:sa", $d) . "<br>";
?>
```

Aşağıdaki örnekte sonraki altı cumartesi gününün tarihleri verilmektedir:

```php
<?php
$baslangic_tarihi=strtotime("Saturday");
$bitis_tarihi=strtotime("+6 weeks", $baslangic_tarihi);

while ($baslangic_tarihi < $bitis_tarihi) {
  echo date("M d", $baslangic_tarihi) . "<br>";
  $baslangic_tarihi = strtotime("+1 week", $baslangic_tarihi);
}
?>
```

Aşağıdaki örnekte 4 Temmuz'a kadar olan gün sayısı verilmektedir:

```php
<?php
  $d1=strtotime("July 04");
  $d2=ceil(($d1-time())/60/60/24);
  //ceil() sayının ondalık kısmı ne olursa olsun yukarı yuvarlar.
  echo "4 Temmuza kadar ". $d2 ." gün vardır";
?>
```

Daha fazla tarih ve zaman işlevi için https://www.w3schools.com/php/php_ref_date.asp adresi incelenebilir.

PHP DOSYA İŞLEMLERİ

Bu kısımda PHP'de dosyalar ile ilgili çeşitli işlemlere yer verilmektedir.

Dosyaları Dahil Etme

Dosya dahil etmek ortak bir dosyayı diğerlerinde kullanmak adına yapılmaktadır. Söz konusu durumda içerme/gerektirme (include/require) ifadeleri kullanılır. İlgili ifadeler, seçilen dosyadaki tüm metni/kodu/işaretlemeyi onu çağırmış olduğunuz dosyaya kopyalar. Benzer PHP, HTML veya metni bir web sitesinin birden fazla sayfasına eklemek istediğinizde dosyaları dahil etmek çok kullanışlıdır. İlaveten, include ve require ifadeleri, başarısızlık durumu dışında aynıdır. Bu başarısızlık durumları olduğunda,

- require önemli bir hata (E_COMPILE_ERROR) üretecek ve betiği durduracaktır
- include ise yalnızca bir uyarı (E_WARNING) üretecek ve komut dosyası çalışmaya devam edecektir

Bu nedenle, yürütmenin devam etmesini ve kullanıcılara çıktıyı göstermesini istiyorsanız, include ifadesi kullanılır. Aksi takdirde, FrameWork, CMS veya karmaşık PHP uygulama kodlaması varsa, yürütme akışına bir anahtar dosya eklemek için her zaman require ifadesi tercih edilir. Bu, bir anahtar dosyanın yanlışlıkla kaybolması durumunda uygulamanızın güvenliğinden ve bütünlüğünden ödün verilmesini önlemeye yardımcı olacaktır.

Dosyaları dahil etmek birçok iş tasarrufu sağlar. Genelde, tüm web sayfalarınız için standart bir üstbilgi, altbilgi veya menü dosyası oluşturmak istediğimizde kullanılmaktadır. Böylece güncelleme gerekirse yalnızca dahil edilen dosyanın değiştirilmesi yeterli olacaktır.

Söz Dizimi:

```php
include 'filename';
```

ya da

```php
require 'filename';
```

şeklindedir. Örnek olarak "footer.php" adında standart bir altbilgi dosyamız olduğunu varsayalım. footer.php şuna benzer:

```php
<?php
  echo "<p>Telif hakkı &copy; 1999-" . date("Y") . " ANKUZEF</p>";
?>
```

Altbilgi dosyasını bir sayfaya eklemek için

```php
<html>
    <body>
        <h1>Sayfama hoşgeldiniz</h1>
        <p>Kendiniz hakkında bir bilgi...</p>
        <p>Kendiniz hakkında bir bilgi...</p>
        <?php include 'footer.php';?>
    </body>
</html>
```

Bir diğer örnek için de menü uygulaması yapalım. Öncelikle menu.php isminde bir dosyamız olsun:

```php
<?php
    echo '<a href="/default.php">Ana Sayfa</a> -
    <a href="/html/default.php">HTML Ders Notu</a> -
    <a href="/css/default.php">CSS Ders Notu</a> -
    <a href="/js/default.php">JavaScript Ders Notu</a> -
    <a href="default.php">PHP Ders Notu</a>';
?>
```

Web sitesindeki tüm sayfaların bu menü dosyasını kullanması için,

```php
<html>
    <body>
    <?php include 'menu.php';?>
        <h1>Sayfama hoşgeldiniz</h1>
        <p>Kendiniz hakkında bir bilgi...</p>
        <p>Kendiniz hakkında bir bilgi...</p>
    </body>
</html>
```

şeklinde yapılır. Bir diğer örnek için de bir dosya içerisindeki değişkenleri başka bir dosyada kullanmaya bakalım. Bu bağlamda vars.php isminde bir dosyamız olduğunu varsayalım. Ve içerisinde renk ve araba isminde değişkenlerimiz olsun:

```php
<?php
  $renk='kırmızı';
  $araba='BMW';
?>
```

Daha sonra "vars.php" dosyasını dahil edersek değişkenler çağıran dosyada kullanılabilir:

```
<html>
<body>
<h1>Web sayfama hoşgeldiniz</h1>
    <?php include 'vars.php';
    echo "Benim $renk renginde $araba arabam var.";
    ?>
</body>
</html>
```

Tüm yukarıda verilen örneklerde include yerine require ifadesi de yazılabilirdi. Ancak dosya bulunamadığında betik ölümcül hata dediğimiz bir hata türünü üretecek ve betiğin çalışmasını durduracaktır.

EK BİLGİ:
Dosyaları yönetirken çok dikkatli olmalısınız. Yanlış bir şey yaparsanız çok fazla zarar verebilirsiniz. Yaygın hatalar şunlardır: Yanlış dosyayı düzenlemek, sabit sürücüyü gereksiz verilerle doldurmak ve bir dosyanın içeriğini kazara silmek.

Dosya işleme herhangi bir web uygulamasının önemli bir parçasıdır. Dosya dahil etme işlemi dışında genellikle farklı görevler için bir dosyayı açmanız ve işlemeniz gerekir. PHP'nin dosyaları oluşturmak, okumak, yüklemek ve düzenlemek için çeşitli işlevleri vardır.

PHP readfile() İşlevi

Bu işlev bir dosyayı okur ve onu çıktı ara belleğine yazar. Sunucuda depoladığımız "webdictionary.txt" adında bir metin dosyamız olduğunu varsayalım. Dosya şuna benzer:

AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language

Dosyayı okumak ve çıktı arabelleğine yazmak için kullanılan PHP kodu aşağıdaki gibidir (readfile() işlevi, başarı durumunda okunan bayt sayısını döndürür):

```
<?php
  echo readfile("webdictionary.txt");
?>
```

Bir diğer dosya işlemi ise sunucudaki bir dosyayı açma, okuma ve kapatmadır.

PHP Dosya Açma - fopen()

Dosyaları açmanın daha iyi bir yöntemi fopen() işlevini kullanmaktır. Bu işlev size readfile() işlevinden daha fazla seçenek sunmaktadır. Dosyalama örneklerinde yine yukarıda bahsedilen "webdictionary.txt" isimli metin dosyasını kullanacağız.

fopen() işlevinin ilk parametresi açılacak dosyanın adını içerir, ikinci parametresi ise dosyanın hangi modda açılması gerektiğini belirtir. Aşağıdaki örnekte, fopen() işlevi belirtilen dosyayı açamadığında da bir mesaj oluşturur:

```php
<?php
    $dosyam = fopen("webdictionary.txt", "r") or die("Dosya açılamıyor!");
    echo fread($dosyam,filesize("webdictionary.txt"));
    fclose($dosyam);
?>
```

Burada die() işlevi dosya açılamadığında devreye girer ve “Dosya açılamıyor!” şeklinde bir uyarı verir. filesize() işlevi ise dosyanın boyutunu döndürmektedir. Son olarak fclose() işlevi ile açılan dosya ile işlem bittikten sonra dosyayı kapatmak için kullanılır. Dosya açma modları Çizelge 1’de verilmektedir:

Çizelge 1. Dosya Modları ve Açıklamaları

Modlar	**Tanım**
r	Bir dosyayı salt okunur olarak açar. Dosya işaretçisi dosyanın başlangıcında başlar
w	Bir dosyayı yalnızca yazmak için açar. Dosyanın içeriğini siler veya mevcut değilse yeni bir dosya oluşturur. Dosya işaretçisi dosyanın başlangıcında başlar
a	Bir dosyayı yalnızca yazmak için açar. Dosyadaki mevcut veriler korunur. Dosya işaretçisi dosyanın sonunda başlar. Dosya mevcut değilse yeni bir dosya oluşturur
x	Yalnızca yazmak için yeni bir dosya oluşturur. Dosya zaten mevcutsa FALSE değerini döndürür ve bir hata verir
r+	Okuma/yazma için bir dosya açar. Dosya işaretçisi dosyanın başlangıcında başlar
w+	Okuma/yazma için bir dosya açar. Dosyanın içeriğini siler veya mevcut değilse yeni bir dosya oluşturur. Dosya işaretçisi dosyanın başlangıcında başlar
a+	Okuma/yazma için bir dosya açar. Dosyadaki mevcut veriler korunur. Dosya işaretçisi dosyanın sonunda başlar. Dosya mevcut değilse yeni bir dosya oluşturur
x+	Okuma/yazma için yeni bir dosya oluşturur. Dosya zaten mevcutsa FALSE değerini döndürür ve bir hata verir

PHP Dosya Okuma - fread()

fread() işlevi açık bir dosyadan okur. İlk parametresi okunacak dosyanın adını içerir ve ikinci parametresi ise okunacak dosyanın maksimum bayt sayısını belirtir. Aşağıdaki örnek PHP kodu "webdictionary.txt" dosyasını sonuna kadar okur:

```
fread($dosyam,filesize("webdictionary.txt"));
```

PHP Dosya Kapatma - fclose()

Bu fonksiyon açık bir dosyayı kapatmak için kullanılır. İşiniz bittikten sonra tüm dosyaları kapatmak iyi bir programlama uygulamasıdır. Sunucunuzda açık bir dosyanın dolaşıp kaynakları kaplaması istenen bir durum değildir. fclose() işlevi kapatmak istediğimiz dosyanın adını (veya dosya adını tutan bir değişkeni) gerektirir:

```
<?php
    $dosyam = fopen("webdictionary.txt", "r");
    // yürütülecek diğer dosya kodlarınız....
    fclose($dosyam);
?>
```

PHP Dosyadan Tek Satır Okuma - fgets()

fgets() fonkssyonu bir dosyadan tek bir satırı okumak için kullanılır. Aşağıdaki örnek "webdictionary.txt" dosyasının ilk satırını verir:

```
<?php
    $dosyam = fopen("webdictionary.txt", "r") or die("Dosya açılamıyor!");
    echo fgets($dosyam);
    fclose($dosyam);
?>
```

Burada fgets() işlevi çağrısından sonra dosya işaretçisi bir sonraki satıra taşınmıştır.

PHP Dosya Sonunu Kontrol Etme - feof()

feof() işlevi, dosya sonuna ulaşılıp ulaşılmadığını kontrol eder. Bu işlev bilinmeyen uzunluktaki veriler arasında döngü yapmak için kullanışlıdır. Aşağıdaki örnekte "webdictionary.txt" dosyası dosya sonuna ulaşılıncaya kadar satır satır okunur:

```
<?php
    $dosyam = fopen("webdictionary.txt", "r") or die("Dosya açılamıyor!");
    // Dosya sonuna kadar tek satır çıktı üretme işlemi
    while(!feof($dosyam))
     {
        echo fgets($dosyam) . "<br>";
     }
    fclose($dosyam);
?>
```

PHP Dosyadan Tek Karakter Okuma - fgetc()

fgetc()bir dosyadan tek bir karakteri okumak için kullanılır. İşlev çağrısından sonra dosya işaretçisi bir sonraki karaktere geçer. Aşağıdaki örnekte "webdictionary.txt" dosyası dosya sonuna ulaşılana kadar karakter karakter okunur:

```
<?php
    $dosyam = fopen("webdictionary.txt", "r") or die("Dosya açılamıyor!");
    // Dosyanın sonuna kadar bir karakter çıktısı alma
    while(!feof($dosyam)) {
      echo fgetc($dosyam);
    }
    fclose($dosyam);
?>
```

PHP Dosya Oluşturma - fopen()

fopen() işlevi aynı zamanda bir dosya oluşturmak için de kullanılır. Var olmayan bir dosya üzerinde kullanırsanız fopen(), dosyanın yazılması (w) veya eklenmesi (a) için açılması koşuluyla, onu oluşturacaktır. Aşağıdaki örnek "testfile.txt" adında yeni bir dosya oluşturur. Dosya, PHP kodunun bulunduğu dizinde oluşturulacaktır:

```
$dosyam = fopen("testfile.txt", "w");
```

Bu kodu çalıştırmaya çalışırken hatalarla karşılaşıyorsanız, PHP dosyanıza sabit sürücüye bilgi yazması için erişim izni verdiğinizi kontrol etmeniz gerekir.

PHP Dosyaya Yazma - fwrite()

fwrite() işlevi bir dosyaya yazmak için kullanılır. İlk parametresi yazılacak dosyanın adını içerir ve ikinci parametresi yazılacak dizedir. Aşağıdaki örnek, "yenidosya.txt" adlı yeni bir dosyaya birkaç bölüm ismi yazar:

```
<?php
    $dosyam = fopen("yenidosya.txt", "w") or die("Dosya açılamıyor!");
    $txt = "Bilgisayar Teknolojisi\n";
    fwrite($dosyam, $txt);
    $txt = "Elektronik Teknolojisi\n";
    fwrite($dosyam, $txt);
    fclose($dosyam);
?>
```

"yenidosya.txt" dosyasına iki kez yazdığımıza dikkat edin. Dosyaya yazma işleminde, ilk olarak "Bilgisayar Teknolojisi" ve ikinci olarak "Elektronik Teknolojisi" içeren $txt dizesini gönderdik. Yazmayı bitirdikten sonra fclose() fonksiyonunu kullanarak dosyayı kapattık. "yenidosya.txt" dosyasını açarsak şöyle görünecektir:

Bilgisayar Teknolojisi
Elektronik Teknolojisi

PHP Dosya Üzerine Yazma

Yukarıdaki örnek çalıştırıldığında "yenidosya.txt" adında bir dosyamız ve içerisinde bazı verilerimiz bulunmaktadır. Söz konusu dosya bu durumda iken üzerine yazma işleminde kullanabiliriz. Ancak, işlem gerçekleştiğinde mevcut tüm veriler silinecek ve yerine yeni veriler gelecektir.

Aşağıdaki örnekte mevcut "yenidosya.txt" dosyamızı açıyoruz ve içine bazı yeni veriler yazıyoruz:

```php
<?php
    $dosyam = fopen("yenidosya.txt", "w") or die("Dosya açılamıyor!");
    $txt = "Bilgisayar Programcılığı\n";
    fwrite($dosyam, $txt);
    $txt = "İnternet ve Ağ Teknolojileri\n";
    fwrite($dosyam, $txt);
    fclose($dosyam);
?>
```

Şimdi "yenidosya.txt" dosyasını açarsak hem “Bilgisayar Teknolojisi” hem de “Elektronik Teknolojisi” yazıları kaybolmuştur ve sadece az önce yazdığımız veriler mevcuttur:

Bilgisayar Programcılığı
İnternet ve Ağ Teknolojileri

PHP Dosyaya Metin Ekleme

"a" modunu kullanarak bir dosyaya veri ekleyebilirsiniz. "a" modu metni dosyanın sonuna eklerken, "w" modu dosyanın eski içeriğini geçersiz kılar ve siler. Aşağıdaki örnekte mevcut "yenidosya.txt" dosyamızı açıyoruz ve ona bir miktar metin ekliyoruz:

```php
<?php
    $dosyam = fopen("yenidosya.txt", "a") or die("Dosya açılamıyor!");
    $txt = "Sosyal Güvenlik\n";
    fwrite($myfile, $txt);
    $txt = "Muhasebe ve Vergi Uygulamaları\n";
    fwrite($dosyam, $txt);
    fclose($dosyam);
?>
```

Şimdi "yenidosya.txt" dosyasını açarsak, “Sosyal Güvenlik” ve “Muhasebe ve Vergi Uygulamaları” verilerinin dosyanın sonuna eklendiğini göreceğiz:

Bilgisayar Programcılığı
İnternet ve Ağ Teknolojileri
Sosyal Güvenlik
İnternet ve Ağ Teknolojileri

PHP Dosya Yükleme

PHP ile dosyaları sunucuya yüklemek kolaydır. Ancak kolaylıkla tehlike de beraberinde gelebilir, bu nedenle dosya yüklemelerine izin verirken her zaman dikkatli olmak gerekmektedir. Bu bağlamda öncelikle PHP'nin dosya yüklemelerine izin verecek şekilde yapılandırılır. Bunun için "php.ini" dosyasında file_uploads açık olarak ayarlanır:

file_uploads = On

Dosya yükleme işlemi için resim dosyaları atılacak örnek bir form oluşturalım:

```
<!DOCTYPE html>
<html>
  <body>
    <form action="upload.php" method="post" enctype="multipart/form-data">
    Yüklenecek görseller:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Görüntü Yükle" name="submit">
    </form>
  </body>
</html>
```

Yukarıdaki HTML formu için uyulması gereken bazı kurallar:

- Formun metot için post kullandığından emin olun.
- Formun ayrıca şu özniteliğe ihtiyacı vardır: enctype="multipart/form-data". İlgili öznitelik, formu gönderirken hangi içerik türünün kullanılacağını belirtir

Yukarıdaki gereksinimler olmadan dosya yükleme işlemi çalışmaz. Dikkat edilmesi gereken diğer şeyler:

- <input> etiketinin type = "file" özelliği, giriş alanını, giriş kontrolünün yanında bir "Gözat" düğmesiyle birlikte bir dosya seçme kontrolü olarak gösterir.

Yukarıdaki form örneğimiz, verileri daha sonra oluşturacağımız "upload.php" adlı dosyaya göndermektedir. “upload.php” dosyası şuna benzer:

```
<?php
    $hedef_dizin = "uploads/";
    $hedef_dosya = $hedef_dizin . basename($_FILES["fileToUpload"]["name"]);
    $yukleme_durumu = 1;
    $goruntuDosyaTuru = strtolower(pathinfo($hedef_dosya,PATHINFO_EXTENSION));
    // Yüklenecek dosyanın gerçek bir görüntü olup olmadığı kontrol
edilmelidir
    if(isset($_POST["submit"])) {
    $kontrol = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
        if($kontrol !== false) {
            echo "Dosya - " . $check["mime"] . "şeklinde bir görüntüdür.";
            $yukleme_durumu = 1;
```

```
        } else {
            echo "Dosya bir görüntü değildir.";
            $yukleme_durumu = 0;
        }
    }
?>
```

Yukarıdaki betikte:

$hedef_dizin = "uploads/" - dosyanın yerleştirileceği dizini belirtir
$hedef_dosya yüklenecek dosyanın yolunu belirtir
$yukleme_durumu = 1 henüz kullanılmıyor (daha sonra kullanılacak)
$goruntuDosyaTuru dosyanın dosya uzantısını tutar (küçük harflerle)

Bu form örneğini çalıştırırken uploads isminde bir dizin oluşturmayı unutmayınız!

Dosyanın var olup olmadığını kontrol etmek için,

```
if (file_exists($hedef_dosya)) {
    echo "Özür dileriz, bu isimde dosya mevcut.";
    $yukemeDurumu = 0;
  }
```

Dosyanın boyutunu sınırlandırmak için,

```
if ($_FILES["fileToUpload"]["size"] > 500000) {
    echo "Özür dileriz dosyanız 500Kb den büyük.";
    $yukleme_durumu = 0;
  }
```

Dosya türünü sınırlamak için,

```
if($goruntuDosyaTuru != "jpg" && $goruntuDosyaTuru != "png" &&
   $goruntuDosyaTuru != "jpeg" && $goruntuDosyaTuru != "gif" )
  {
    echo "Sadece JPG, JPEG, PNG & GIF dosyalarına izin verilmektedir.";
    $yuklemeDurumu = 0;
  }
```

Dosya yükleme işlemi için gerekli olan upload.php dosyasının tamamı aşağıdadır:

```php
<?php
$hedef_dizin= "uploads/";
$hedef_dosya = $hedef_dizin . basename($_FILES["fileToUpload"]["name"]);
$yuklemeDurumu = 1;
$goruntuDosyaTuru=strtolower(pathinfo($hedef_dosya,PATHINFO_EXTENSION));

    // Bir görüntü dosyası olup olmadığının kontrolü
    if(isset($_POST["submit"])) {
       $kontrol = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
        if($kontrol !== false) {
            echo "Dosya - " . $check["mime"] . "şeklinde bir görüntüdür.";
            $yukleme_durumu = 1;
        } else {
            echo "Dosya bir görüntü değildir.";
            $yukleme_durumu = 0;
        }
    }
    // Dosyanın mevcut olup olmadığının kontrolü
    if (file_exists($hedef_dosya)) {
      echo "Özür dileriz dosya mevcut.";
      $yukleme_durumu = 0;
    }
    // Dosya boyutunun kontrolü
    if ($_FILES["fileToUpload"]["size"] > 500000) {
      echo "Özür dileriz dosya boyutu 500Kb den büyük.";
      $yukleme_durumu = 0;
    }
    // Belirli türde dosyalara izin verme
    if($goruntuDosyaTuru != "jpg" && $goruntuDosyaTuru != "png" &&
       $goruntuDosyaTuru != "jpeg"  && $goruntuDosyaTuru != "gif" ) {
        echo "Sadece JPG, JPEG, PNG & GIF dosyalarına izin verilmektedir.";
        $yuklemeDurumu = 0;
       }

    // $yukleme_durumu değişkeni ile yükleme gerçekleşmezse bir hata üretme
    if ($yukleme_durumu == 0) {
    echo "Özür dileriz dosya yüklenmedi.";
    // Herşey yolunda ise yükleme işlemi
    } else {
        if(move_uploaded_file($_FILES["fileToUpload"]["tmp_name"],
$hedef_dosya)) {
         echo "Dosya ".htmlspecialchars(
basename($_FILES["fileToUpload"]["name"])). " yüklendi.";
            } else {
            echo "Özür dileriz dosya yüklenirken bir hata oluştu.";
        }
    }
?>
```

Dosyalama işlemleri ile ilgili daha fazla işlev için,
https://www.w3schools.com/php/php_ref_filesystem.asp adresini ziyaret edebilirsiniz.

ÇEREZLER

Bir çerez (cookie) genellikle bir kullanıcıyı tanımlamak için kullanılan ve sunucu tarafından istemci bilgisayarına yerleştirilen küçük bir dosyadır. Her seferinde aynı bilgisayar, tarayıcı ile bir sayfa istediğinde oluşturulan çerez de gönderilir. PHP ile ilgili çerez değerlerini hem oluşturabilir hem de alabilirsiniz.

HATIRLATMA:
Sunucu tarafından tarayıcının belleğinde tutulan bir çerez, httpOnly parametresi içeriyorsa bu çerez yalnızca HTTP isteklerine eklenir ve Javascript tarafından okunamaz, değiştirilemez!

Çerez Oluşturma

setcookie() işlevi çerezi oluşturur. Söz dizimi:

setcookie(ad, değer, son kullanma tarihi, yol, etki alanı, güvenli, httponly);

Bu işlevde sadece ad parametresi zorunludur. Diğerleri ihtiyaca bağlı olarak eklenir.

UYARI:
setcookie() işlevi <html> öğesinden önce kullanılmalıdır!

Aşağıdaki örnek, "ANKUZEF" değerine sahip "user" adında bir çerez oluşturur. Çerezin süresi 30 gün sonra, yani 86400 saniye x 30'da sona erecektir. Örnekte "/" parametresi çerezin web sitesinin tamamında mevcut olmasını sağlar. Aksi takdirde tercih ettiğiniz dizini belirlememiz gerekir. $_COOKIE global değişkeni ise "user" çerezinin değerini almaya yarar. Ayrıca çerezin tanımlı olup olmadığını öğrenmek için de isset() işlevi kullanılmaktadır:

```php
<?php
    $cerez_ismi = "user";
    $cerez_degeri = "ANKUZEF";
    setcookie($cerez_ismi, $cerez_degeri, time() + (86400 * 30), "/");
        // 86400 = 1 gün
?>
<html>
<body>

<?php
    if(!isset($_COOKIE[$cerez_ismi])) {
      echo " ' " .$cerez_ismi. " ' isminde bir çerez yoktur!";
    } else {
    echo "Çerez ' " . $cerez_ismi. " ' isminde tanımlanmıştır!<br>";
    echo "Değeri ise " . $_COOKIE[$cerez_degeri] . "dir";
    }
?>

</body>
</html>
```

Örneğimizde çerezin değeri, çerez gönderildiğinde otomatikman URL olarak kodlanır ve alındığında otomatik olarak kodu çözülür. URL kodlamasını önlemek için setrawcookie() işlevi kullanılır.

Çerez Değerini Değiştirme

Bir çerezi değiştirmek için, setcookie() işlevi kullanarak çerezi tekrar ayarlamanız yeterlidir:

Çerez Silme

Bir çerezi silmek için geçmişten bir tarih ile setcookie() işlevi yürütülür.

```
<?php
  // çerezin son kullanma tarihini 1 saat öncesine ayarlama
  setcookie("user", "", time() - 3600);
?>
<html>
<body>
<?php
  echo "Kullanıcı çerezi silindi.";
?>
</body>
</html>
```

Çerezin Etkinliğini Kontrol Etme

Aşağıdaki örnek, çerezlerin etkin olup olmadığını kontrol eden küçük bir komut dosyası oluşturur. Öncelikle setcookie() fonksiyonuyla bir test çerezi oluşturulup, ardından $_COOKIE dizi değişkeni saydırılır. Eğer değer sıfırdan büyükse çerez etkindir.

```
<?php
    setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
    if(count($_COOKIE) > 0) {
    echo "Çerezler etkin.";
    } else {
    echo "Çerezler etkin değil.";
    }
?>

</body>
</html>
```

OTURUMLAR

Oturum, birden fazla sayfada kullanılacak bilgileri (değişkenler halinde) saklamanın bir yoludur. Çerezlerden farklı olarak bilgiler kullanıcının bilgisayarında saklanmaz.

PHP Oturumu nedir?

Bir uygulamayla çalışırken açıp, bazı değişiklikler yapıp sonra kapatırız. Bu durum bir Oturuma çok benzemektedir. Normal olarak, bilgisayar senin kim olduğunu, uygulamayı ne zaman başlattığını ve ne zaman sonlandıracağını bilir. Ancak internette bir sorun varsa HTTP adresi durumu korumadığından web sunucusu sizin kim olduğunuzu veya ne yaptığınızı bilemez. Oturum değişkenleri, birden fazla sayfada kullanılacak kullanıcı bilgilerini (örn. kullanıcı adı, favori renk vb.) depolayarak bu sorunu çözer. Varsayılan olarak oturum değişkenleri, kullanıcı tarayıcıyı kapatana kadar sürer. Bu bağlamda; oturum değişkenleri tek bir kullanıcı hakkındaki bilgileri tutar ve bunlar tek bir uygulamadaki tüm sayfalarda kullanılabilir.

PHP Oturum Başlatma

session_start() işlevi oturum başlatır. Oturum değişkenleri ise PHP $_SESSION ile ayarlanır. Şimdi "demo_session1.php" adında yeni bir sayfa oluşturalım. Bu sayfada yeni bir PHP oturumu başlatıyoruz ve bazı oturum değişkenlerini ayarlıyoruz:

HATIRLATMA:
session_start() işlevi, herhangi bir HTML etiketinden önce belgenizdeki ilk şey olmalıdır.

```php
<?php
    // Oturum başlatma
    session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
    // Oturum değişkenleri ayarlama
    $_SESSION["favori_renk"] = "yeşil";
    $_SESSION["favori_hayvan"] = "kedi";
    echo "Oturum değişkenleri ayarlandı.";
?>
</body>
</html>
```

Oturum Değişken Değerlerini Alma

Yukarıdaki oturum başlatma sayfasına ek olarak "demo_session2.php" adında başka bir sayfa oluşturuyoruz. Bu sayfadan ilk sayfada belirlediğimiz oturum bilgilerine ("demo_session1.php") ulaşacağız.

```php
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
    // Önceki sayfada (demo_session1.php) ayarlanan oturum değişkenlerini
    // yazdırma
    echo "Favori rengim " . $_SESSION["favori_renk"] . ".<br>";
    echo "Favori hayvanım " . $_SESSION["favori_hayvan"] . ".";
?>

</body>
</html>
```

Burada oturum değişkenlerinin her yeni sayfaya ayrı ayrı aktarılmadığını, bunun yerine her sayfanın başında açtığımız oturumdan session_start() işlevi ile alındığını bilmemiz gerekir. Ayrıca tüm oturum değişkeni değerleri $_SESSION değişkeninde saklanmaktadır. Bir kullanıcı oturumuna ilişkin tüm oturum değişkeni değerlerini göstermenin başka bir yolu da aşağıdaki kodu çalıştırmaktır:

EK BİLGİ:
Çoğu oturum, kullanıcının bilgisayarında bir kullanıcı anahtarı belirler: Örneğin, 765487cf34ert8dede5a562e4f3a7e12. Daha sonra başka bir sayfada oturum açıldığında, kullanıcı anahtarı için bilgisayarı tarar. Eşleşme varsa o oturuma erişir, yoksa yeni bir oturum başlatır. Bu şekilde kim olduğumuzu sistem bilmektedir.

```php
<?php
  session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
  print_r($_SESSION);
?>

</body>
</html>
```

Oturum Değişkenini Değiştirme

Bir oturum değişkenini değiştirmek için üzerine yazmanız yeterlidir. Örnek kod aşağıdaki gibidir:

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
    // oturum değişkenini üzerine yazarak değiştirme
    $_SESSION["favori_renk"] = "sarı";
    print_r($_SESSION);
?>

</body>
</html>
```

Oturum Yok Etme

Tüm genel oturum değişkenlerini kaldırmak ve oturumu yok etmek için sırasıyla session_unset() ve session_destroy() işlevleri kullanılmaktadır:

```
<?php
  session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
  // oturum değişkenlerini yok etme
  session_unset();

  // oturumu yok etme
  session_destroy();
?>

</body>
</html>
```

FİLTRELER

PHP filtreleri harici girişi doğrulamak ve temizlemek için kullanılır. Bu bağlamda, PHP filtre uzantısı, kullanıcı girişini kontrol etmek için gereken işlevlerin çoğuna sahiptir ve veri doğrulamayı daha kolay ve hızlı hale getirmek için tasarlanmıştır.

filter_list() işlevi, PHP filtre uzantısının neler sunduğunu listelemek için kullanılabilir:

```
<table>
  <tr>
    <td>Filter Name</td>
    <td>Filter ID</td>
  </tr>
  <?php
  foreach (filter_list() as $id =>$filter) {
    echo '<tr><td>' . $filter . '</td><td>' . filter_id($filter) .
'</td></tr>';
  }
  ?>
</table>
```

Neden Filtre Kullanmalı?

Birçok web uygulaması harici girdi alır. Harici giriş/veriler şunlar olabilir:

- Bir formdan kullanıcı girişi
- Çerez
- Web hizmetleri verileri
- Sunucu değişkenleri
- Veritabanı sorgu sonuçları

EK BİLGİ:
Verilerin doğrulanması = Verilerin uygun biçimde olup olmadığını belirleme işlemidir.

Verilerin temizlenmesi = Verilerden yasa dışı karakterleri kaldırmaktır.

Geçersiz gönderilen veriler güvenlik sorunlarına yol açabilir ve web sayfanızın bozulmasına neden olabilir! PHP filtrelerini kullanarak uygulamanızın doğru girişi aldığından emin olabiliriz.

PHP filter_var() İşlevi

filter_var() işlevi, verileri hem doğrular hem de sterilize eder. Ayrıca, tek bir değişkeni belirtilen filtreyle filtreler. Kontrol etmek istediğiniz değişken ve kullanılacak kontrol türü olmak üzere iki parça veri alır.

Bir Dizeyi Sterilize Etme

Aşağıdaki örnekte, bir dizeden tüm HTML etiketlerini kaldırmak için filter_var() işlevinin nasıl kullanıldığı gösterilmektedir:

```php
<?php
    $metin = "<h1>ANKUZEF</h1>";
    $yeni_metin = filter_var($metin, FILTER_SANITIZE_STRING);
    echo $yeni_metin;
?>
```

Bir Tam Sayıyı Doğrulama

Aşağıdaki örnek, $int değişkeninin bir tamsayı olup olmadığını kontrol etmek için filter_var() işlevini kullanır. Eğer $int bir tamsayı ise, "Tamsayı geçerli" değilse "Tamsayı geçerli değil" çıktısı üretilmektedir:

```php
<?php
    $int = 100;

    if (filter_var($int, FILTER_VALIDATE_INT) === 0 ||
        !filter_var($int, FILTER_VALIDATE_INT) === false)
    {
      echo("Tamsayı geçerli");
    } else {
      echo("Tamsayı geçerli değil");
    }
?>
```

Burada tamsayı değerinde sıfır kontrolü de yapılmaktadır. Aksi takdirde filtervar() işlevi 0 için tam sayı gereçli değil çıktısı üretmektedir.

IP Adresini Doğrulama

Aşağıdaki örnek, $ip değişkeninin geçerli bir IP adresi olup olmadığını kontrol etmek için filter_var() işlevini kullanır:

```php
<?php
    $ip = "127.0.0.1";

    if (!filter_var($ip, FILTER_VALIDATE_IP) === false) {
    echo("$ip adresi geçerlidir");
    } else {
    echo("$ip adresi geçerli değildir");
    }
?>
```

Bir E-posta Adresini Temizleme ve Doğrulama

Aşağıdaki örnek, önce $email değişkenindeki tüm geçersiz karakterleri kaldırmak, ardından bunun geçerli bir e-posta adresi olup olmadığını kontrol etmek için filter_var() işlevini kullanır:

```php
<?php
    $email = "ankuzef@ankara.edu.tr";

    // tüm geçersiz karakterleri kaldırma
    $email = filter_var($email, FILTER_SANITIZE_EMAIL);

    // eposta adresini doğrula
    if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
      echo("$email adresi geçerli");
    } else {
      echo("$email adresi geçerli değil");
    }
?>
```

Bir URL'yi Temizleme ve Doğrulama

Aşağıdaki örnek, öncelikle bir URL'deki tüm yasa dışı karakterleri kaldırmak, ardından $url'nin geçerli bir URL olup olmadığını kontrol etmek için filter_var() işlevini kullanır:

```php
<?php
    $url = "https://ankuzef.ankara.edu.tr/";

    // tüm yasa dışı karakterleri kaldırma
    $url = filter_var($url, FILTER_SANITIZE_URL);

    // URL doğrulama
    if (!filter_var($url, FILTER_VALIDATE_URL) === false) {
      echo("$url geçerli bir URL'dir");
    } else {
      echo("$url geçerli bir URL değildir");
    }
?>
```

Bir Aralık İçinde Bir Tam Sayıyı Doğrulama

Aşağıdaki örnek, bir değişkenin hem INT türünde hem de 1 ile 200 arasında olup olmadığını kontrol etmek için filter_var() işlevini kullanır:

```php
<?php
    /* kontrol edilecek değişken */
    $int = 122;

    /* minimum değer */
    $min = 1;
    /* maksimum değer */
    $max = 200;
```

```
    if (filter_var($int, FILTER_VALIDATE_INT, array("options" =>
array("min_range"=>$min, "max_range"=>$max))) === false) {
      echo("Değişkenin değeri istenen aralıkta değil");
    } else {
      echo("Değişkenin değeri $int, $min ile $max aralığında");
    }
?>
```

IPv6 Adresini Doğrulama

Aşağıdaki örnek, $ip değişkeninin geçerli bir IPv6 adresi olup olmadığını kontrol etmek için filter_var() işlevini kullanır:

```
<?php
  $ip = "2001:0db8:85a3:08d3:1319:8a2e:0370:7334";

    if (!filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6) === false) {
      echo("$ip geçerli bir IPv6 adresidir");
    } else {
      echo("$ip geçerli IPv6 adresi değildir");
    }
?>
```

Sorgu Dizesi İçeren URL'yi Doğrula

Aşağıdaki örnek, $url değişkeninin sorgu dizesi içeren (örneğin https://www.domain.com/url?degisken1=deger&degisken2=value) şeklinde bir URL olup olmadığını kontrol etmek için filter_var() işlevini kullanır:

```
<?php
    $url = "https://oys.ankara.edu.tr/mod/lti/view.php?id=44029";

   if (!filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_REQUIRED) ===
false) {
      echo("$url sorgu dizesi içeren bir URL'dir");
    } else {
      echo("$url sorgu dizesi içeren bir URL değildir");
    }
?>
```

ASCII Değerine Bakarak Dize Temizleme

Bir dizeyi ASCII tablosundaki verileri kullanarak temizlemek için filter_var() işlevi kullanılır. Aşağıdaki örnek, hem tüm HTML etiketlerini hem de ASCII değeri > 127 olan (ÆØÅ) tüm karakterleri dizeden kaldıracaktır:

```php
<?php
 $metin = "<h1>Hello WorldÆØÅ!</h1>";

 $yeni_metin=filter_var($metin,FILTER_SANITIZE_STRING,FILTER_FLAG_STRIP_HIGH);
 echo $yeni_metin;
?>
```

GERİ ÇAĞIRMA İŞLEVLERİ

Geri çağırma işlevi, başka bir işleve argüman olarak iletilen işlevleri ifade etmektedir. Mevcut herhangi bir işlev geri çağırma işlevi olarak kullanılabilir. Bir işlevi geri çağırma işlevi olarak kullanmak için, işlevin adını içeren bir dizeyi başka bir işlevin argümanı olarak iletiriz.

Aşağıdaki örnek, bir dizideki her dizenin uzunluğunu hesaplamak için PHP'nin array_map() işlevine bir geri çağrı iletmektedir:

```php
<?php
    function uzunlukHesapla($oge) {
      return strlen($oge);
    }

    $meyveler = ["elma", "portakal", "muz", "hindistan cevizi"];
    $uzunluklar = array_map("uzunlukHesapla", $meyveler);
    print_r($uzunluklar); // Array ( [0] => 4 [1] => 8 [2] => 3 [3] => 16 )
?>
```

İlgi örnek çalıştırıldığında array_map() işlevi üzerinden $meyveler isimli dizi değişkenin içerisindeki her bir elemanın (*["elma", "portakal", "muz", "hindistan cevizi"]*) strlen() aracılığıyla uzunlukları alınacak ve bu işlem uzunlukHesapla() işlevine dizi öğesi bitene kadar geri çağrı yapılarak tekrarlanacaktır.

PHP'nin 7.x sürümünden itibaren anonim olan işlevler de geri çağırma işlevleri olarak iletilebilmektedir. Aşağıda, array_map() işlevi için geri çağırma olarak anonim bir işlev kullanılmaktadır:

```php
<?php
$meyveler = ["elma", "portakal", "muz", "hindistan cevizi"];
$uzunluklar = array_map( function($oge) { return strlen($oge); }, $meyveler);
print_r($uzunluklar);
?>
```

Kullanıcı Tanımlı Fonksiyonlarda Geri Çağırma

Kullanıcı tanımlı işlevler, geri çağırma işlevlerini değişken olarak alabilmektedir. Bunun için, öncelikle değişkene parantez ekleyerek çağrı yapılır. Ardından ilgili değişkenleri normal işlevlerde olduğu gibi iletiriz:

```php
<?php
    function ornek1($a) {
       return $a . "! ";
    }

    function ornek2($a) {
       return $a . "? ";
    }
```

```
    function cikti($a, $b) {
      // $b değişkenini geri çağırma işlevi olarak kullanma
      echo $b($a);
    }

    // cikti() isimli işleve “ornek1” ve “ornek2” yi geri
    // çağırma işlevi olarak gönderme
    cikti("ANKUZEF", "ornek1"); // ANKUZEF!
    cikti("ANKUZEF", "ornek2"); // ANKUZEF?
?>
```

İlgili kod yürütüldüğünde cikti() isimli işlevde “ANKUZEF” kelimesi parametre olarak ornek1() ve ornek2() isimli işlevlere değişken olarak atanmakta ve geri çağırma işlevi elde edilmektedir.

PHP ve JSON

JSON, JavaScript Nesne Gösterimi anlamına gelir ve verileri depolamak ve değiştirmek için kullanılan bir sözdizimidir.

JSON formatı metin tabanlı bir biçime sahip olduğundan, bir sunucudan veya sunucuya kolayca gönderilebilir ve herhangi bir programlama dili tarafından veri olarak kullanılabilir. PHP'nin JSON'u işlemek için bazı yerleşik işlevleri vardır. Bu bağlamda, aşağıdaki iki fonksiyona bakacağız:

- json_encode()
- json_decode()

json_encode()

json_encode() işlevi, bir değeri JSON biçimine kodlamak içindir. Aşağıdaki örnek, ilişkisel bir dizinin bir JSON nesnesine nasıl kodlanacağını gösterir:

```php
<?php
    $ogrenci_sayilari = array("Bilgisayar Teknolojisi"=>150,
                              "Elektronik Teknolojisi"=>200);
    echo json_encode($ogrenci_sayilari);
?>
```

Yukarıdaki betik çağrıldığında bir JSON nesnesi elde edilmektedir:

{"Bilgisayar Teknolojisi":150,"Elektronik Teknolojisi":200}

Benzer şekilde indeksli bir dizi üzerinde yapılan örnek aşağıda verilmektedir:

```php
<?php
    $arabalar = array("Volvo", "BMW", "Toyota");
    echo json_encode($arabalar); // ["Volvo","BMW","Toyota"]
?>
```

json_decode()

json_decode () işlevi, bir JSON nesnesinin kodunu bir PHP nesnesine veya ilişkisel diziye dönüştürmek için kullanılır. Aşağıdaki örnek, JSON verilerinin kodunu bir PHP nesnesine dönüştürür:

```php
<?php
    $jsonobj = '{"Bilgisayar Teknolojisi":150,"Elektronik Teknolojisi":200}';
    var_dump(json_decode($jsonobj));
?>
```

İlgili betiğin çıktısı,

```
object(stdClass)#1 (2) { ["Bilgisayar Teknolojisi"]=> int(150) ["Elektronik
Teknolojisi"]=> int(200) }
```

şeklindedir. json_decode () işlevi varsayılan olarak bir nesne döndürür. json_decode () işlevinin ikinci bir parametresi vardır ve true olarak ayarlandığında JSON nesnelerinin kodu ilişkisel diziler halinde çözülür:

```
object(stdClass)#1 (2) { ["Bilgisayar Teknolojisi"]=> int(150) ["Elektronik
Teknolojisi"]=> int(200) } bool(true)
```

Kodu Çözülmüş Değerlere Erişim

Burada bir nesneden ve ilişkisel bir diziden kodu çözülmüş değerlere nasıl erişileceğine dair iki örnek verilmiştir. Aşağıdaki örnek, bir PHP nesnesinden değerlere nasıl erişileceğini gösterir:

```php
<?php
    $jsonobj = '{"Bilgisayar Teknolojisi":150,"Elektronik Teknolojisi":200}';
    $obj = json_decode($jsonobj);

    echo $obj->{"Bilgisayar Teknolojisi"} . '<br>';
    echo $obj->{"Elektronik Teknolojisi"};
?>
```

Betik çıktı olarak ilişkisel dizinin değerlerini ekrana yazar:

150
200

Diğer örnek ise ilişkisel diziden değerlere nasıl erişileceğini gösterir:

```php
<?php
    $jsonobj = '{"Bilgisayar Teknolojisi":150,"Elektronik Teknolojisi":200}';

    $arr = json_decode($jsonobj, true);

    echo $arr["Bilgisayar Teknolojisi"].'<br>';
    echo $arr["Elektronik Teknolojisi"];
?>
```

Değerler Arasında Döngü

Değerler arasında foreach() döngüsüyle geçiş yapılmaktadır:

```php
<?php
    $jsonobj = '{"Bilgisayar Teknolojisi":150,"Elektronik Teknolojisi":200}';
    $obj = json_decode($jsonobj);

    foreach($obj as $key => $value) {
      echo $key . " => " . $value . "<br>";
    }
?>
```

Elde edilen çıktı:

```
Bilgisayar Teknolojisi => 150
Elektronik Teknolojisi => 200
```

şeklindedir. Aşağıdaki örnek ise ilişkisel dizinin değerleri arasında nasıl döngü yapılacağını gösterir:

```php
<?php
    $jsonobj = '{"Bilgisayar Teknolojisi":150,"Elektronik Teknolojisi":200}';
    $arr = json_decode($jsonobj, true);

    foreach($arr as $key => $value) {
      echo $key . " => " . $value . "<br>";
    }
?>
```

PHP İSTİSNALARI

İstisna, bir PHP betiğinin hatasını veya beklenmeyen davranışını tanımlayan bir nesnedir. Birçok PHP işlevi ve sınıfı tarafından istisnalar oluşturulur. Kullanıcı tanımlı işlevler ve sınıflar da istisnalar oluşturabilir. İstisnalar, bir işlevin kullanamayacağı verilerle karşılaştığında onu durdurmanın iyi bir yoludur.

Bir İstisna Oluşturmak

Throw ifadesi, kullanıcı tanımlı bir işlevin veya yöntemin bir istisna oluşturmasına izin verir. Bir istisna oluştuğunda onu takip eden kod çalıştırılmaz. Bir istisna yakalanmazsa "Yakalanmayan İstisna" mesajıyla ölümcül bir hata meydana gelecektir. Yakalamadan bir istisna oluşturma örneği aşağıda verilmektedir:

```php
<?php
    function bolme($bolum, $bolen) {
        if($bolen == 0) {
        throw new Exception("Bölenin 0 olma hatası");
        }
        return $bolum / $bolen;
    }

    echo bolme(5, 0);
?>
```

Betiğin çıktısı şuna benzemektedir:

```
Fatal error: Uncaught Exception: Bölenin 0 olma hatası in
C:\xampp\htdocs\index.php:4 Stack trace: #0 C:\xampp\htdocs\index.php(9):
bolme(5, 0) #1 {main} thrown in C:\xampp\htdocs\index.php on line 4
```

try...catch İfadesi

Yukarıdaki örnekteki hatayı önlemek, istisnaları yakalamak ve işleme devam etmek için try...catch ifadesini kullanabiliriz. Söz dizimi:

```php
try
 {
    // istisnalar atabilecek kod
 }
catch(Exception $e)
 {
    // bir istisna yakalandığında çalışan kod
 }
```

Aşağıda bir istisna oluştuğunda mesaj atan bir örnek vardır:

```php
<?php
    function bolme($bolunen, $bolen) {
        if($bolen == 0) {
            throw new Exception("Bölenin 0 olma hatası!");
        }
        return $bolunen / $bolen;
    }

  try
  {
    echo bolme(5, 0);
  }
  catch(Exception $e)
  {
    echo "Bölme işlemi tanımsızdır.";
  }
?>
```

Betik yürütüldüğünde ekrana "Bölme işlemi tanımsızdır" yazar. Catch bloğu, ne tür bir istisnanın yakalanması gerektiğini ve istisnaya erişmek için kullanılabilecek değişkenin adını belirtir. Yukarıdaki örnekte istisnanın türü Exception ve değişkenin adı ise $e dir.

try...catch...finally İfadesi

try...catch...finally ifadesi istisnaları yakalamak için kullanılabilir. finally bloğundaki kod, bir istisnanın yakalanıp yakalanmadığına bakılmaksızın her zaman çalışacaktır. finally mevcutsa catch bloğu isteğe bağlıdır. Söz dizimi:

```
try
{
    // istisnalar atabilecek kod
}
catch(Exception $e)
{
    // bir istisna yakalandığında çalışan kod
}
finally
{
    // bir istisnanın yakalanıp yakalanmamasına bakılmaksızın
    //her zaman çalışan kod
}
```

Aşağıda bir istisna oluştuğunda bir mesaj gösteren ve ardından işlemin sona erdiğini belirten örnek bulunmaktadır:

```php
<?php
    function bolme($bolunen, $bolen) {
        if($bolen == 0) {
            throw new Exception("Bölenin 0 olma hatası!");
        }
    return $bolunen / $bolen;
    }

    try {
        echo bolme(5, 0);
    }
    catch(Exception $e) {
        echo "Bölünemiyor. ";
    }
    finally {
     echo "İşlem tamamlandı.";
    }
?>
```

Bir istisna yakalanmasa bile bir dize çıktısı almak için

```php
<?php
    function bolme($bolunen, $bolen) {
        if($bolen == 0) {
            throw new Exception("Bölenin 0 olma hatası!");
        }
      return $bolunen / $bolen;
     }

    try {
       echo bolme(5, 0);
    }
    finally {
       echo "İşlem tamamlandı.";
    }
?>
```

yazılmalıdır.

İstisna Nesnesi

İstisna Nesnesi, işlevin karşılaştığı hata veya beklenmeyen davranış hakkında bilgi içerir. Söz dizimi:

new Exception(mesaj, kod, önceki)

Burada üç parametrenin kullanımı da isteğe bağlı olup,

- “mesaj” istisnanın neden atıldığını göstermek,
- “kod”, istisnayı aynı türdeki diğerlerinden kolayca ayırt etmek için kullanılabilecek bir tam sayı üretmek ve
- “önceki” de istisna başka bir istisnanın catch bloğunda atılmışsa, bu istisnanın bu parametreye iletilmesini sağlamak

için kullanılmaktadır.

Yöntemler

Çizelge 2’de PHP istisnalar için kullanılan diğer yöntemler ve kullanım amacı verilmektedir:

Çizelge 2. İstisna Yöntemleri

Yöntem	**Tanım**
getMessage()	İstisnanın neden atıldığını açıklayan bir dize döndürür
getPrevious()	Bu istisna başka bir istisna tarafından tetiklendiyse, bu yöntem önceki istisnayı döndürür. Değilse, null değerini döndürür
getCode()	İstisna kodunu döndürür
getFile()	İstisnanın oluşturulduğu dosyanın tam yolunu döndürür
getLine()	İstisnayı atan kod satırının satır numarasını döndürür

Aşağıda bir istisna için Çizelge 2’de verilen yöntemler kullanılmaktadır:

```php
<?php
    function bolme($bolunen, $bolen) {
        if($bolen == 0) {
            throw new Exception("Bölenin 0 olma hatası!", 1);
        }
    return $bolunen / $bolen;
    }

    try {
        echo bolme(5, 0);
    }
    catch(Exception $ex) {
        $code = $ex->getCode();
        $message = $ex->getMessage();
        $file = $ex->getFile();
        $line = $ex->getLine();
        echo "$line. satırda $file dosyasında istisna oluşturuldu: [Code
              $code] $message";
    }
?>
```

Yukarıdaki betiğin çıktısı,

```
4. satırda C:\xampp\htdocs\index.php dosyasında istisna oluşturuldu: [Code 1]
Bölenin 0 olma hatası!
```

şeklindedir.

BÖLÜM ÖZETİ

İleri internet programcılığı, web geliştirme alanında daha karmaşık ve ileri düzey becerilerin gerektiği bir konudur. İlgili alanda uzmanlaşmak isteyenler, genellikle İnternet Programcılığı 1, dersindeki temel web geliştirme bilgisine sahip olduktan sonra ileri düzey teknolojiler ve teknikler öğrenmeye başlarlar. Bu bağlamda, sunucu taraflı programlama ve uygulama arayüzü hazırlama bilgilerine ihtiyaç duyulmaktadır. Sunucu taraflı programlama genellikle veritabanı yönetimi, sunucu iş mantığı ve uygulama programlama arabirimleri yani api'ler oluşturmayı içerir. Ayrıca, kullanıcı deneyimini iyileştirmek, tarayıcı uyumluluğunu sağlamak ve görsel tasarımı optimize etmek adına etkileşimli arayüzlere ihtiyaç vardır. Bu nedenle html, si es es, Java Script, PHP, My es q eL, gibi yazılımlarda kendimizi geliştirmemiz gerekir. İleri düzey web uygulamaları genellikle büyük miktarda veri ile çalışır. Veritabanı tasarımı, optimize edilmiş sorguların yazılması ve veritabanı yönetimi becerileri, ileri internet programcılığının önemli bir parçasıdır. Güvenlik ve performans optimizasyonu ise yine bu alanda dikkate alınması gereken bir diğer konudur. Bizler kitabımızın birinci bölümünde işlevler, dosyalama, filtreleme ve ji son objeleri ile çalışarak ileri düzey web uygulamalarına giriş yaptık. İlerleyen bölümlerde nesneye yönelik programlama, My es q eL, ile veritabanı işlemleri x emel, Ajax ve web servislerini öğrenerek bu alandaki ihtiyaçlarımızı karşılamaya çalışacağız. İleri internet programcılığı, sürekli olarak değişen ve gelişen bir alandır. Bu nedenle, bu alanda çalışanlar sürekli olarak yeni teknolojileri ve en iyi uygulamaları takip etmeli ve kendilerini sürekli olarak güncellemelidirler.

GÖZDEN GEÇİRELİM

1. Günün (pazartesi, salı vb.) çıktısını nasıl alırız?

2. Veritabanı bağlantısı için conn.php isimli bir dosyanızın olduğunu varsayın. İlgili dosyayı veritabanına bağlantı yapmak istediğiniz diğer web sayfalarında da kullanmanız gerekmektedir. Bahsi geçen dosyayı sayfalara dahil eden PHP kodu nedir?

3. Kullanıcı bilgilerini tanımlamak ve ilgili değerleri diğer sayfalara da taşımak adına ne kullanılmalıdır?

4. PHP'de dosya işlemleri için kullanılan işlevler nelerdir?

5. JSON verilerini işlemek için hangi işlevleri kullanırız?

Cevaplar:

1. echo date(l);

2. <?php include conn.php';?>

3. çerez ya da oturum

4. readfile(), fopen(), fread(), fwrite(), fclose(), fgets(), feof(), fgetc()

5. json_encode(), json_decode()

KAYNAKÇA

[1] Learn to code. W3Schools Online Web Tutorials. (n.d.). https://www.w3schools.com/

FAYDALI KAYNAKLAR VE INTERNET KAYNAKLARI

Dimes, T., & L., C. (2017). PHP. Babelcube Inc.

PHP and JSON. (n.d.). https://www.w3schools.com/php/php_json.asp

PHP Basics. (n.d.). Beginning PHP and MySQL, 55–112. https://doi.org/10.1007/978-1-4302-0299-8_3

PHP callback functions. (n.d.). https://www.w3schools.com/php/php_callback_functions.asp

PHP Date/Time Functions. (n.d.). https://www.w3schools.com/php/php_ref_date.asp

PHP exception reference. (n.d.). https://www.w3schools.com/php/php_ref_exception.asp

PHP filesystem functions. (n.d.). https://www.w3schools.com/php/php_ref_filesystem.asp

PHP filter functions. (n.d.). https://www.w3schools.com/php/php_ref_filter.asp

PHP include files. PHP include and require. (n.d.). https://www.w3schools.com/php/php_includes.asp

PHP network functions. (n.d.). https://www.w3schools.com/php/php_ref_network.asp