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

BÖLÜM 2

PHP NESNEYE YÖNELİK PROGRAMLAMA

ANAHTAR KAVRAMLAR

Sınıf, Nesne, Kurucu/Yıkıcı, Soyut, Kalıtım, Arayüz, Yineleyiciler

ÖĞRENME HEDEFLERİ

1. Sınıfların ve nesneleri kullanmayı ve etkileşimlerini öğrenmek
2. Erişim belirleyicileri kullanarak sınıf özellik ve yöntemlerine erişimi kontrol etmek
3. Soyut sınıfların tanımlanmasını ve kullanılmasını öğrenmek
4. Statik yöntemlerin ve özelliklerin tanımlanmasını ve kullanılmasını öğrenmek
5. Yineleyicileri kullanarak veri koleksiyonları yönetmek

GİRİŞ

Nesne Yönelimli Programlama (Object-Oriented Programming) (OOP), yazılım geliştirme sürecinde ihtiyaçlar, nesneler ve bu nesneler arasındaki ilişkiler üzerine odaklanır. Bu yaklaşım, gerçek dünyadaki nesnelerin (objelerin) modellenmesi ve bu objeler arasındaki etkileşimlerin programlama dili içinde temsil edilmesi üzerine kuruludur. Nesne yönelimli programlamada, veri ve işlevsellik bir araya getirilerek bir nesne oluşturulur. Bu nesneler, belirli özellikleri (nitelikler veya öznitelikler) ve davranışları (metodlar veya işlevler) içerebilir. Bu sayede, programlar daha modüler, anlaşılabilir ve bakımı kolay hale gelir. Nesne yönelimli programlama, programların yeniden kullanılabilirliğini artırır ve büyük ölçekli yazılım projelerinin yönetimini kolaylaştırır.

Nesne tabanlı programlamada, nesneler özellikleri (Properties), işlevleri (Methods) ve olayları (Events) ile tanımlanır. Bu programlama paradigmalarına örnek olarak C++, C#, Java, Objective-C, PHP, Smalltalk, Python, Ruby, ABAP/4 ve Visual Basic .NET örnek olarak verilebilir.

Prosedürel programlama, işlemleri ve fonksiyonları veriler üzerinde uygulamakla ilgiliyken, nesne yönelimli programlama verileri ve fonksiyonları içeren nesneler oluşturmayı hedefler.

Nesne yönelimli programlamanın prosedürel programlamaya göre birkaç avantajı bulunmaktadır:

- NYP, daha hızlı ve daha kolay bir şekilde yürütülür.
- NYP, programlar için net bir yapı sağlar.
- NYP, PHP kodunun "Kendini Tekrar Etmeme" (DRY) ilkesine uymasına yardımcı olur ve kodun bakımını, değiştirilmesini ve hata ayıklamasını kolaylaştırır.
- NYP, daha az kod yazma ve daha kısa geliştirme süresi ile tamamen yeniden kullanılabilir uygulamaların oluşturulmasını mümkün kılar.

İpucu: "Kendini Tekrar Etmeme" (DRY) prensibi, kod tekrarını azaltmayı hedefler. Ortak kodları uygulamadan ayıklamalı, tek bir yerde tutmalı ve yeniden kullanmalısınız, böylece tekrarı önlemiş olursunuz.

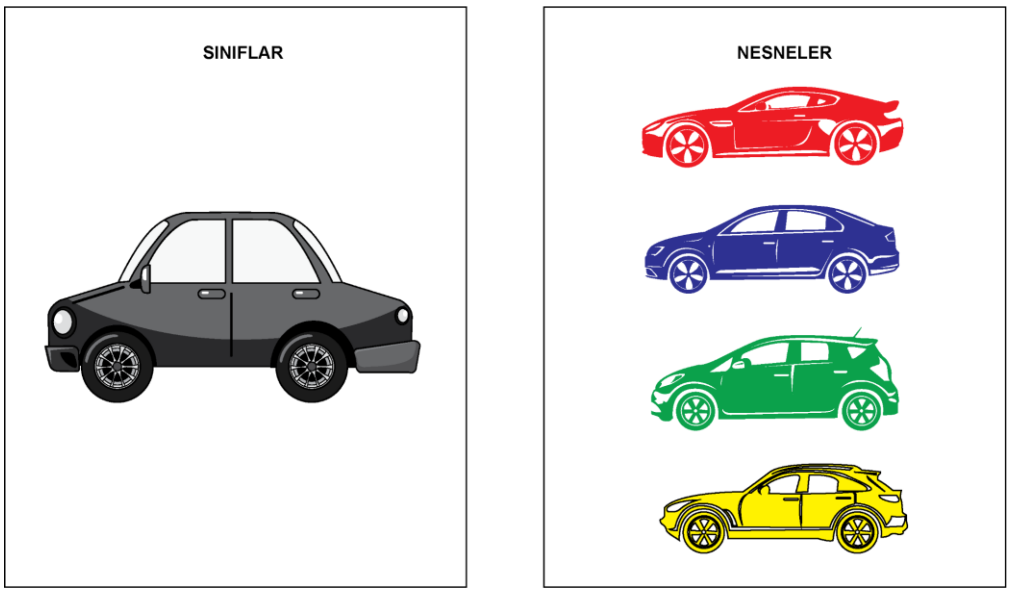
Nesne tabanlı programlama, üç temel prensibe dayanır:

- ***Kapsülleme/Paketleme*** (*Encapsulation*): Kullanıcıya gösterilmeyen kod parçalarının gizlenmesini ifade eder. Kullanıcı, bir aracın istediği sonucu alırken aracın iç işleyişiyle ilgilenmez.
- ***Kalıtım*** (*Inheritance*): Bir sınıftan yeni bir sınıf oluşturmayı ifade eder. Yeni sınıf, özelliklerini ve davranışlarını temel aldığı eski sınıftan miras alır.

- ***Çok Biçimlilik (Polymorphism):*** Aynı işlemin farklı nesneler veya ortamlar üzerinde farklı sonuçlar üretebilme yeteneğidir. Örneğin, konuşma işlemi yüksek sesle, normal ses tonuyla veya kısık sesle gerçekleştirilebilir.

SINIFLAR VE NESNELER

PHP'de, modern programlama dillerinde olduğu gibi sınıf tanımlamaları yapılabilir. PHP'nin 5. versiyonuyla birlikte sınıf tanımlama konusunda önemli değişiklikler yapılmış ve tamamen yeni bir Nesne Modeli oluşturulmuştur. Programlama dillerinde sınıflar, kendi içinde tutarlılığı olan özel işleri modellemek için kullanılır. Bu işlerin yürütülmesinden tamamen tanımlanan bu sınıf sorumlu olacaktır. Böylece, programlar gerçek hayatta olduğu gibi sınıflar aracılığıyla modellenir ve daha anlaşılır hale gelir. Sınıf tanımlamalarıyla kodlar düzenli bir şekilde yazılır ve tekrar tekrar kullanılabilir hale gelir. Örneğin, veritabanı işlemleri için bir sınıf tanımlaması yapılabilir ve veritabanıyla ilgili tüm işlemler bu sınıf üzerinden gerçekleştirilebilir. Benzer şekilde, web sayfasında kullanıcı işlemleriyle ilgili bir sınıf tanımlaması yapılabilir ve kullanıcı girişi, çıkışı, yeni kullanıcı tanımlama gibi birçok işlem tamamen bu sınıf üzerinden gerçekleştirilebilir. Aşağıdaki Resim 1'de Araba sınıfından türetilmiş fakat kendine özgü farklı özellikleri bulunan arabalar sınıf-nesne ilişkisine örnek gösterilebilir.



Resim 1: Sınıf- Nesne ilişkisi

Sınıflar içerisinde sabitler, değişkenler ve fonksiyonlar bulunabilir. Bunlara sınıfın üyeleri denilmektedir. Sınıf tanımlanırken class kelimesi kullanılır. Sınıftan türetilen her bir örnek ise nesne olarak tanımlanmaktadır. Program gereği yapılan işlemler genelde sınıfın bir örneği olan nesneler üzerinden yürütülmektedir.

Sınıf tanımı, **class** anahtar kelimesiyle başlamalıdır. Sınıf adı, **class** kelimesinden sonra gelir ve ardından iki süslü parantez "{}" içerisinde sınıfa ait özelliklerin ve yöntemlerin tanımları bulunur.

Sınıf adı olarak herhangi bir etiket kullanılabilir, ancak bu PHP için özel olarak ayrılmış bir kelime olmamalıdır. Geçerli bir sınıf adı, bir harf veya alt çizgi ile başlar ve ardından sayılar,

harfler veya alt çizgiler içerebilir. Türkçe karakterlerin kullanımında bir kısıtlama olmasa da programlamada mümkün olduğunca kullanmamak doğru olacaktır. Örnek: `Araba, _Araba, _Araba2`

PHP'de sınıf tanımlamalarında geçerli kabul edilmeyen tanımlamalar şunlardır.

- Rakamla Başlayan Sınıf İsmi: class 1Sınıf { } // Hatalı
- Özel Karakterler İçeren Sınıf İsmi: class #Sınıf { } // Hatalı
- Boşluk İçeren Sınıf İsmi: class Uzun Sınıf { } // Hatalı
- Ayrılmış PHP Anahtar Kelimeleri Kullanan Sınıf İsmi: class function { } // Hatalı

Aşağıdaki örnekte bir sınıf ve metotlar yardımıyla bir nesnenin özellikleri ekranda gösteriliyor.

```php
<?php
class Araba
{
    public $marka;
    public $model;
    public $renk;

    //kurucu metot oluşturuluyor
    public function __construct($marka, $model, $renk)
    {
        $this->marka = $marka;
        $this->model = $model;
        $this->renk = $renk;
    }
    //bir metot yardımıyla gösterim sağlanıyor
    function ozellikleriGoster()
    {
        echo "Araba Özellikleri: Marka -" . $this->marka . "Model - " . $this-
>model . "Renk -" . $this->renk;
    }
}
$araba1 = new Araba("Toyota", "Corolla", "Beyaz");
$araba1->ozellikleriGoster();
?>
```

$this Anahtar Kelimesi

$this anahtar sözcüğü geçerli nesneyi ifade eder ve yalnızca yöntemlerin içinde kullanılabilir.

Örnek:

```php
<?php
    class Meyve{
        public $meyveAdi;
    }
    $elma = new Meyve();
?>
```

Örnekte yer alan $meyveAdi özelliğinin değerini iki yöntemle değiştirebiliriz.

Yöntem 1: Sınıfın içinde (set_meyveAdi() yöntemini ekleyerek ve $this'i kullanarak):

```php
<?php
    class Meyve {
       public $meyveAdi;
       function set_meyveAdi($meyveAdi){
         $this -> meyveAdi = $meyveAdi;
       }
}
$elma = new Meyve();
$elma->set_meyveAdi("Elma");

echo $elma->meyveAdi;
?>
```

Yöntem 2: Sınıfın dışında (özellik değerini doğrudan değiştirerek):

```php
<?php
    class Meyve {
       public $meyveAdi;
}
$elma = new Meyve();
$elma->meyveAdi="Elma";

echo $elma->meyveAdi;
?>
```

Bir nesnenin belirli bir sınıfa ait olup olmadığını kontrol etmek için ***instanceof*** anahtar sözcüğü kullanılır.

```php
<?php
    class Meyve {
       public $meyveAdi;
}
$elma = new Meyve();
$elma->meyveAdi="Elma";

//echo $elma->meyveAdi;

var_dump($elma instanceof Meyve);
?>
```

new Anahtar Sözcüğü

Bir sınıfın örneğini oluşturmak için new anahtar kelimesi kullanılmalıdır. Örneğin: $ornek = new Sinif();

Nesnenin oluşturulması sırasında bir hata oluşturmaması için yapıcı metot, istisna (exception) oluşturmayacak şekilde yapılandırılmış olması gerekir. Ayrıca sınıflardan örnek

oluşturulmadan önce tanımlanmış olması gereklidir. new kelimesi ile tanımlanmış olan nesnenin özelliklerinde bellekte yeni bir nesne oluşturulur.

PHP - Kurucular (Constructor)

Sınıflar (nesneler) oluşturulduğunda otomatik çalışan metotlar yapıcı veya kurucu metot (constructor method) olarak isimlendirilirler. Her yeni örnek oluşturulduğunda kurucu metot çalıştırılır ve ilklendirme işlemleri bu yöntemle gerçekleştirilir.

PHP'de, bir sınıfın kurucusunu tanımlamak için **__construct()** metodunu kullanılır. Bu metod, sınıf adından önce çift alt çizgi ile belirtilir ve sınıfın özelliklerini başlatmak veya diğer başlatma işlemlerini gerçekleştirmek için kullanılır.

```
    //kurucu metodumuz (bir kere çalışacak)
public function __construct()
{
    echo "Lastik sayısı: " . $this->lastik . ", Vites sayısı: " . $this->vites
. "<br>";
}
```

İki alt çizgi ile başlayan kurucu metotların tanımlanmasında, zorunlu olmayan fakat ihtiyaç duyulduğu kadar parametre kullanılabilir.

```
<?php
    class Bisiklet {
        public $lastik = 2;
        public $vites = 6;
    //kurucu metodumuz (bir kere çalışacak)
    public function __construct()
    {
        echo "Lastik sayısı: " . $this->lastik . ", Vites sayısı: " . $this-
>vites . "<br>";
    }
    public function bisiklet_sur(){
        if($this->lastik >= 2 && $this->vites >= 1){
           //En az iki lastik ve bir vites varsa
            $this->lastik -=2;
            $this->vites -= 1;
            echo   "Bisiklet Sürüldü";
        }else{
            echo "Bisikletin lastik ve/veya vitesini kontrol ediniz";
        }
    }
}
$bisiklet = new Bisiklet(); // Nesne olusturduk
$bisiklet->bisiklet_sur(); // Hesne ile Slnlf icerisindeki bir metoda
ilerledik
?>
```

PHP - Yıkıcılar (Destructors)

Yıkıcılar (destructors) bir sınıftan örnek alınan nesnenin kullanımı sona erdiğinde, örneğin bellekten kaldırıldığında çalıştırılan özel metodlardır. Yıkıcı metotlar, PHP'de __destruct() olarak adlandırılır ve sınıf içinde tanımlanırlar.

Yıkıcı metotlar, örneğin sonlandırılması gereken bazı kaynakları serbest bırakmak, dosyaları kapatmak, veritabanı bağlantılarını sonlandırmak gibi işlemleri gerçekleştirmek için kullanılır. Bu, nesnenin artık ihtiyaç duyulmadığı ve programın bellek kullanımını optimize etmek için bellekten kaldırıldığı durumlarda önemli olabilir.

Bir sınıf içinde tanımlanan bir yıkıcı metot, nesnenin kullanımı sona erdiğinde otomatik olarak çağrılır ve her örneklemede yalnızca bir kez çalışır. Yıkıcı metotlar genellikle başlatıcı metotlar (constructors) ile çift olarak kullanılır. Başlatıcı metod nesnenin oluşturulduğu zaman, yıkıcı metot ise nesnenin bellekten kaldırıldığı zaman çalışır.

```php
<?php
    class Araba {
        public function __construct(){ //kurucu metot tanımı
            echo "Araba nesnesi oluşturuldu.";
        }
        public function araba_kullan(){
            echo "Araba kullanılıyor.";
        }
        public function __destruct(){//yıkıcı metot tanımı
            echo "Araba nesnesi bellekten kaldırıldı.";
        }
}
$araba = new Araba(); // Araba nesnesi oluşturuldu.
$araba->araba_kullan(); // Araba kullanılıyor.
unset($araba); // Araba nesnesi bellekten kaldırıldı.
?>
```

İpucu: Yapıcılar ve yıkıcılar kod miktarını azaltmaya yardımcı olduğundan çok faydalıdırlar!

PHP - Erişim Belirleyiciler (Access Modifiers)

Erişim belirleyiciler, nesne tabanlı programlamada kapsülleme prensibini uygulamak için kullanılan anahtar kelimelerdir. Bir sınıfın üyelerine (özellikler ve yöntemler) nerelerden erişilebileceğini belirlerler.

PHP'de 3 temel erişim belirleyici vardır:

Public: Public olarak tanımlanan bir üyeye, sınıfın kendisinden, oluşturulan nesnelerden ve bu sınıftan miras alan alt sınıflardan erişilebilir.

Private: Private olarak tanımlanan bir üyeye sadece sınıfın kendisinden erişilebilir. Dışarıdan, nesnelerden ve alt sınıflardan erişilemez.

Protected: Protected olarak tanımlanan bir üyeye, sınıfın kendisinden ve bu sınıftan miras alan alt sınıflardan erişilebilir. Dışarıdan ve nesnelerden erişilemez.

Erişim belirleyicilerde bilinmesi gereken bazı önemli noktalar ise şunlardır.

- **Varsayılan erişim belirleyici:** Bir üyenin erişim belirleyicisi belirtilmezse, varsayılan olarak **public** olur.
- **Erişim belirleyiciler ve kalıtım:** Bir sınıftan miras alan alt sınıf, üst sınıftaki **protected** üyelere erişebilir.

Erişim belirleyicileri kullanmanın faydaları:

- **Kod güvenliği:** Kapsülleme prensibini sağlayarak kodun daha güvenli ve sağlam olmasını sağlar.
- **Değişiklik yönetimi:** Kodda değişiklik yapmayı kolaylaştırır.
- **Bakım kolaylığı:** Kodun daha kolay anlaşılmasını ve bakımı yapılmasını sağlar.

```php
<?php
    class Kişi {
        public $ad; // Her yerden erişilebilir
        private $yaş; // Sadece Kişi sınıfından erişilebilir
        protected $adres; // Kişi sınıfından ve alt sınıflardan erişilebilir

        // Ad özelliğine erişim sağlayan metot
        public function getAd()
        {
            return $this->ad;
        }

        // Yaş özelliğine erişim sağlayan metot
        private function getYaş()
        {
            return $this->yaş;
        }

        // Adres özelliğine erişim sağlayan metot
        protected function getAdres()
        {
            return $this->adres;
        }
    }

    $kişi = new Kişi();
    echo $kişi->ad; // Çalışır
    echo $kişi->yaş; // Hata: 'yaş' özelliği erişilebilir değil
    echo $kişi->adres; // Hata: 'adres' özelliği erişilebilir değil
?>
```

PHP'de Kalıtım Nedir?

PHP'de kalıtım, Nesneye Dayalı Programlama'da (OOP) yeni sınıflar (alt sınıflar) oluşturmanıza ve mevcut sınıfların (üst sınıflar) özelliklerini ve yöntemlerini yeniden kullanmanıza olanak tanıyan temel bir kavramdır. Bu, kodun yeniden kullanılabilirliğini teşvik eder, yedekliliği azaltır ve kodunuzu hiyerarşik bir şekilde düzenlemenize yardımcı olur.

Kalıtımın Temel Noktaları:

- Bir alt sınıf, üst sınıfından tüm halka açık ve korunan üyeleri (özellikler ve yöntemler) devralır.
- Alt sınıf, devralınan üyelere ek olarak kendi özel üyelerine de sahip olabilir.
- Bir alt sınıfı, bir üst sınıftan devralmak için **extends** anahtar kelimesi kullanılır.

Kalıtımın Faydaları:

- ***Kod Yeniden Kullanılabilirliği***: Bir üst sınıftan ortak işlevleri devralarak aynı kodu tekrar tekrar yazmaktan kaçınabilirsiniz.
- ***Azaltılmış Yedeklilik:*** Kalıtım, kod kopyalamayı önler ve kod tabanınızı temiz tutar.
- ***Hiyerarşik İlişkiler:*** Sınıf hiyerarşileri oluşturarak nesneler arasındaki gerçek dünya ilişkilerini modellemenize olanak tanır.
- ***Polimorfizm:*** Kalıtım, farklı sınıflardaki nesneleri benzer şekilde ele almanıza olanak tanıyan polimorfizm için bir temel oluşturur.

```php
<?php
    class Hayvan
    {
        public $isim;
        public function sesCikar()
        {
            echo "Genel hayvan sesi";
        }
    }

    class Kopek extends Hayvan
    {
        public function havla()
        {
            echo "Hav!";
        }
    }

    $kopek = new Kopek();
    $kopek->sesCikar(); // Hayvan'dan devralındı
    $kopek->havla(); // Kopek'e özgü yöntem
?>
```

EK BİLGİ

PHP, **tek kalıtımı** destekler, yani bir alt sınıf doğrudan yalnızca bir üst sınıftan miras alabilir.

Miras alma zinciri oluşturarak (örneğin, torun sınıfın alt sınıftan miras alması) çok seviyeli bir miras alma yapısı oluşturabilirsiniz.

Kalıtım, alt sınıfta devralınan yöntemlerin **geçersiz kılınmasına** izin verir. Bu, devralınan bir yöntemin davranışını, alt sınıfın özel gereksinimlerine uyacak şekilde yeniden tanımlayabileceğiniz anlamına gelir.

PHP'de Kalıtım ve Korumalı Erişim Değiştirici

Korumalı Erişim Değiştirici: protected erişim değiştirici, bir sınıfın üyelerine (özellikler ve yöntemler) **sadece sınıfın kendisinden ve bu sınıftan miras alan alt sınıflardan** erişilebilmesini sağlar.

Kalıtım ile İlişkisi:

- Bir alt sınıf, bir üst sınıftan devralınan korumalı üyelere **doğrudan erişebilir**.
- Dışarıdan ve nesnelerden korumalı üyelere erişilemez.

Faydaları:

- **Kapsülleme:** Sınıfın iç işleyişini gizleyerek kodun daha güvenli ve sağlam olmasını sağlar.
- **Değişiklik Yönetimi:** Kodda değişiklik yapmayı kolaylaştırır.
- **Bakım Kolaylığı:** Kodun daha kolay anlaşılmasını ve bakımı yapılmasını sağlar.

```php
<?php
    class UstSinif
    {
        protected $isim;
        protected function getIsim()
        {
            return $this->isim;
        }
    }

    class AltSinif extends UstSinif
    {
```

```
        public function getUstSinifIsmi()
        {
            return $this->isim; // Çalışır, isim devralındı
        }
    }

    $ustSinif = new UstSinif();
    echo $ustSinif->isim; // Hata: 'isim' özelliği erişilebilir değil
    $altSinif = new AltSinif();
    echo $altSinif->isim; // Hata: 'isim' özelliği erişilebilir değil
    echo $altSinif->getUstSinifIsmi(); // Çalışır, getIsim metodu aracılığıyla
erişim
?>
```

Dikkat Edilmesi Gerekenler:

- Korumalı üyeler, alt sınıflarda **değiştirilebilir**.
- Korumalı üyeler, **sadece miras yoluyla erişilebilir**, doğrudan erişim mümkün değildir.

PHP'de Miras Yöntemleri Geçersiz Kılma

PHP'de, bir alt sınıfta **üst sınıftan devralınan bir yöntemin davranışını yeniden tanımlamak** için **geçersiz kılma** (override) işlemini kullanabilirsiniz.

Nasıl Yapılır:

- Alt sınıfta, aynı ada ve parametre listesine sahip bir yöntem tanımlanır.
- Bu yöntem, üst sınıftan devralınan yöntemi **geçersiz kılar** ve onun yerine geçer.

Aşağıda yer alan kod bir "Meyve" sınıfı oluşturur ve ondan türetilmiş bir "Çilek" sınıfı içerir. "Meyve" sınıfı meyve özelliklerini temsil ederken, "Çilek" sınıfı, meyve özelliklerine ek olarak ağırlığı da temsil eder.

```
<?php
// Meyve sınıfı tanımlanıyor
class Meyve
{
    // Meyve özellikleri
    public $isim;
    public $renk;

    // Meyve sınıfının yapıcı metodunu tanımlıyoruz
    public function __construct($isim, $renk)
    {
        // Kurucu metotta meyve özelliklerini atıyoruz
        $this->isim = $isim;
        $this->renk = $renk;
    }

    // Meyveyi tanıtan metot
```

```php
    public function tanitim()
    {
        // Meyvenin adını ve rengini ekrana yazdırıyoruz
        echo "Meyvenin adı {$this->isim} ve rengi {$this->renk}.";
    }
}

// Çilek sınıfı Meyve sınıfından türetiliyor
class Cilek extends Meyve
{
    // Çileğin ek özelliği: ağırlık
    public $agirlik;

    // Çilek sınıfının yapıcı metodunu tanımlıyoruz
    public function __construct($isim, $renk, $agirlik)
    {
        // Çilek sınıfının yapıcı metodu, Meyve sınıfının yapıcı metodunu
çağırır
        parent::__construct($isim, $renk);
        // Çileğin ağırlığını atıyoruz
        $this->agirlik = $agirlik;
    }

    // Çileği tanıtan metot
    public function tanitim()
    {
        // Çileğin adını, rengini ve ağırlığını ekrana yazdırıyoruz
        echo "Meyvenin adı {$this->isim}, rengi {$this->renk} ve ağırlığı
{$this->agirlik} gram.";
    }
}

// Çilek nesnesi oluşturuluyor
$cilek = new Cilek("Çilek", "kırmızı", 50);
// Çileği tanıtan metot çağrılıyor
$cilek->tanitim();
?>
```

Dikkat Edilmesi Gerekenler:

- Geçersiz kılma işleminde, alt sınıftaki yöntemin **imzası** (adı ve parametre listesi) **üst sınıftaki yöntemin imzasıyla tam olarak eşleşmelidir**.
- Geçersiz kılınan yöntem, **override anahtar kelimesiyle** açıkça belirtilebilir. Bu, kodun daha okunabilir ve anlaşılır olmasını sağlar.

Geçersiz Kılmanın Faydaları:

- **Kodun Esnekliği:** Farklı alt sınıflar için farklı davranışlar sağlayarak kodun daha esnek olmasını sağlar.
- **Kod Yeniden Kullanılabilirliği:** Genel işlevselliği üst sınıfta tanımlayarak ve alt sınıflarda özelleştirerek kodun yeniden kullanılabilirliğini artırır.

PHP'de final Anahtar Kelimesi

PHP'de final anahtar kelimesi, kalıtımı ve yöntem geçersiz kılmayı kısıtlayarak kodun netliğini ve istenmeyen davranışları önler. İşlevleri şunlardır:

1. final Sınıflar:

- Bir sınıf final olarak bildirildiğinde, **başka sınıflar tarafından türetilemez**.
- Bu, temel bir kavramı temsil eden veya miras yoluyla değiştirilmemesi gereken iyi tanımlanmış bir uygulaması olan sınıflar için kullanışlıdır.

```php
<?php
    final class Matematik {
        public static function toplama($a, $b)
        {
            return $a + $b;
        }
    }

    /* final olarak tanımlanmış bir sınıftan yeni bir
    sınıf türetme işlemi gerçekleştirilemez. Bu kod hata verir.
    */
    class Geometri extends Matematik {
    }
?>
```

PHP - Sınıf Sabitleri

Sınıf sabitleri, bir sınıfın tüm nesneleri tarafından paylaşılan ve değiştirilemeyen değerlerdir. Sabitler, **const** anahtar kelimesi kullanılarak tanımlanır.

- Sınıf adı ve :: operatörü kullanılarak sabite erişebilirsiniz.
- Bir nesne örneği aracılığıyla da sabite erişebilirsiniz.

Sınıf Sabitlerinin Özellikleri:

- Sabitler, bir kez tanımlandıktan sonra değiştirilemez.
- Sabitler tüm nesneler tarafından paylaşılır.
- Sabitler, statik (static) olarak da bilinir.
- Sınıf sabitleri büyük/küçük harfe duyarlıdır. Ancak sabitlerin tamamının büyük harflerle adlandırılması önerilir.
- Sabitler, basit veri türleri (string, integer, float) veya null değer alabilir.

Sınıf Sabitlerini Kullanmanın Avantajları:

- Kodun daha okunabilir ve anlaşılır olmasını sağlar.
- Tekrarlayan değerleri tek bir yerde tanımlayarak kodun daha az yer kaplamasını sağlar.
- Değişiklikleri tek bir noktadan yaparak kodun daha kolay bakımı yapılmasını sağlar.

self veya, aşağıdaki gibi , anahtar kelimeyi, ardından kapsam çözümleme operatörünü (::) ve ardından sabit adını kullanarak sınıf içinden bir sabite erişebiliriz :

```php
<?php
    class Veritabani
    {
        const HOST = "localhost";
        const KULLANICI_ADI = "admin'";
        const SIFRE = "1234";
        const VERITABANI_ADI = "phpDers";

        public function baglan()
        {
            echo "HOST : " . self::HOST . "\n";
            echo "KULLANICI_ADI: " . self::KULLANICI_ADI . "\n";
            echo "SIFRE : " . self::SIFRE . "\n";
            echo "VERITABANI_ADI: " . self::VERITABANI_ADI . "\n ";
        }
    }
    $baglanti = new Veritabani();
    $baglanti->baglan( ) ;
?>
```

PHP - Soyut Sınıflar ve Yöntemler

Soyut sınıflar ve yöntemler, PHP'de nesne yönelimli programlamada kullanılan bir kavramdır. Bir sınıf soyut olarak işaretlenebilir, yani soyut bir sınıf, kendisi örneklenemez ancak diğer sınıfların temelini oluşturabilir. Soyut sınıflar, genellikle benzer özelliklere ve davranışlara sahip nesneler için bir şablon veya arayüz sağlamak için kullanılır. Bir sınıfı soyut yapmak için **abstract** anahtar kelimesi kullanılır. Soyut sınıflar içinde soyut yöntemler tanımlanabilir. Soyut yöntemler, sınıfların bu yöntemi uygulamak zorunda olduğunu belirtir, ancak soyut sınıf içinde bu yöntemlerin gerçek bir uygulaması bulunmaz. Alt sınıflar, soyut sınıftan kalıtım alırken soyut yöntemleri uygulamak zorundadır. Bu sayede, soyut sınıflar ve yöntemler, kodun daha modüler ve esnek olmasını sağlar, çünkü alt sınıflar aynı temel davranışı paylaşabilir ve bu davranışı özelleştirebilir.

Soyut Sınıfları Tanımlama:

- **abstract** anahtar kelimesi kullanılarak tanımlanır.

- En az bir soyut yöntem içerir.
- Soyut yöntemler, gövdesiz (içeriği olmayan) olarak tanımlanır.

Soyut Yöntemler:

- Soyut yöntemler, alt sınıflarda **mutlaka** geçersiz kılınmalıdır (override).
- Geçersiz kılınan yöntemlerde, soyut yöntemin imzası (ad ve parametre listesi) eşleştirilmelidir

Aşağıdaki örnekte soyut bir sınıf Hayvan ve içerisinde soyut bir yöntem sesCikar() oluşturulmuştur. Hayvan sınıfından türetilen Kopek sınıfında, üst sınıfta yer alan soyut yöntem sesCikar() geçersiz kılınarak (override) kullanılmıştır.

```php
<?php
    abstract class Hayvan {
      abstract public function sesCikar();
      public function hareketEt () {
        echo "Genel hayvan hareketi";
      }
    }

    class Kopek extends Hayvan
    {
        public function sesCikar()
        {
            echo "Hav!";
        }
    }
    $kopek = new Kopek();
    $kopek->sesCikar() ; // "Hav!" yazdırır
?>
```

Soyut Sınıfların Avantajları:	Soyut Sınıfların Dezavantajları:
• Kodun yeniden kullanılabilirliğini artırır. • Ortak işlevselliği tek bir yerde tanımlayarak kodun daha az yer kaplamasını sağlar. • Kodun daha organize ve tutarlı olmasını sağlar.	• Somut nesneler oluşturamazlar. • Daha karmaşık bir kod yapısı oluşturabilirler.

Soyut Sınıfların Kullanım Alanları:

- Ortak bir arayüze sahip nesneler oluşturmak için.
- Kalıtım yoluyla kodun yeniden kullanılabilirliğini artırmak için.

- Kodun daha organize ve tutarlı olmasını sağlamak için.

Örnek Kullanım:

- Hayvan, Bitki gibi kategorileri soyut sınıflar ile tanımlamak.
- Veritabanı bağlantısı, dosya işlemleri gibi genel işlevleri soyut sınıflar ile implemente etmek

PHP Arayüzler

Arayüzler (interfaces), bir sınıfın belirli bir davranışı tanımlayan bir sözleşme veya şablon sağlar. Arayüzler, bir sınıfın belirli bir davranış kümesine sahip olmasını garantileyen yöntem bildirimlerini içerir. Bir sınıf, birden fazla arayüzü uygulayabilir.

Arayüzler, bir sınıfın belirli bir davranışı nasıl gerçekleştireceğini tanımlamaz, sadece hangi yöntemleri uygulaması gerektiğini belirtir. Arayüzler, kodun modülerliğini artırır ve farklı sınıfların belirli bir davranışı nasıl gerçekleştireceğini önceden belirlemesine olanak tanır.

Arayüzler, **interface** anahtar kelimesiyle tanımlanır ve sınıflar tarafından **implements** anahtar kelimesiyle uygulanır. Bir sınıf, bir veya daha fazla arayüzü uygulayabilir. Arayüzlerdeki tüm yöntemler, uygulayan sınıflar tarafından aynı isimde ve imzada (parametrelerle birlikte) uygulanmalıdır.

Arayüzleri Tanımlama:

- **interface** anahtar kelimesi kullanılarak tanımlanır.
- Yalnızca **soyut yöntemler** içerir.
- Yöntem gövdeleri tanımlanamaz.

Aşağıdaki örnekte, SesCikarabilir adında bir arayüz tanımlanıyor. Bu arayüz, sesCikar() adında bir soyut yöntem içeriyor. Hayvan adında bir sınıf SesCikarabilir arayüzünü uyguluyor. Bu sınıf, sesCikar() yöntemini "Genel hayvan sesi" yazdıracak şekilde özelleştiriyor. Köpek adında bir sınıf Hayvan sınıfından miras alıyor. Bu sınıf, sesCikar() yöntemini "Hav!" yazdıracak şekilde yeniden tanımlıyor. $kopek adında bir Köpek nesnesi oluşturuluyor. $kopek->sesCikar() fonksiyonu çağrıldığında, Köpek sınıfındaki sesCikar() yöntemi çalıştırılıyor ve "Hav!" yazdırılıyor.

```php
<?php
    interface SesCikarabilir
    {
        public function sesCikar();
    }

    class Hayvan implements SesCikarabilir
    {
        public function sesCikar()
```

```
        {
            echo "Genel hayvan sesi";
        }
    }

    class Köpek extends Hayvan
    {
        public function sesCikar()
        {
            echo "Hav!";
        }
    }
    $kopek = new Köpek();
    $kopek->sesCikar(); // "Hav!" yazdırır
?>
```

Arayüzlerin Avantajları	Arayüzlerin Dezavantajları
• Kodun yeniden kullanılabilirliğini artırır. • Sınıflar arasında tutarlı bir davranış sağlar. • Kodun daha organize ve tutarlı olmasını sağlar.	• Somut nesneler oluşturamazlar. • Daha karmaşık bir kod yapısı oluşturabilirler.

Arayüzlerin Kullanım Alanları:

- Farklı tiplerde nesneler arasında ortak bir davranış tanımlamak için.
- Kalıtım yoluyla kodun yeniden kullanılabilirliğini artırmak için.
- Kodun daha organize ve tutarlı olmasını sağlamak için.

Örnek Kullanım Alanı:

- Ödeme, kimlik doğrulama gibi genel işlevleri arayüzler ile implemente etmek.
- Farklı veri kaynaklarına erişmek için arayüzler ile soyutlama katmanı oluşturmak.

Arayüzler ve Soyut Sınıflar Arasındaki Farklar:

- Arayüzler yalnızca soyut yöntemler içerirken, soyut sınıflar hem soyut hem de somut yöntemler içerebilir.
- Bir sınıf birden fazla arayüzü uygulayabilirken, tek bir soyut sınıftan miras alabilir.

PHP - Nitelikler (Traits)

PHP'de nitelikler, birden fazla sınıf tarafından **paylaşılabilen** bir dizi **yöntem ve özelliği** tanımlayan bir yapıdır. Özellikler, kodun **yeniden kullanılabilirliğini** ve **tutarlılığını** artırmak için kullanılır.

Nitelikler, **trait** anahtar kelimesiyle tanımlanır ve **use** anahtar kelimesiyle sınıflara eklenir. Bir sınıf, bir veya daha fazla trait kullanabilir.

Aşağıdaki örnekte, Yuzme ve Kosma adında iki trait tanımlanmıştır. Her biri birer metod içerir. Insan sınıfı, bu iki traiti kullanır ve böylece yuz() ve kos() metodlarına erişebilir. Traitler, kodun parçalanmasını ve yeniden kullanılabilirliğini artırır, böylece kod tekrarından kaçınmamıza yardımcı olur.

```php
<?php
    trait Yuzme {
        public function yuz() {
            echo "Yüzüyorum...";
        }
    }

    trait Kosma {
        public function kos() {
            echo "Koşuyorum...";
        }
    }

    class Insan {
        use Yuzme, Kosma;
    }

    $insan = new Insan();
    $insan->yuz(); // Yüzüyorum...
    $insan->kos(); // Koşuyorum...
?>
```

Niteliklerin Avantajları	Niteliklerin Dezavantajları
• Kodun yeniden kullanılabilirliğini artırır. • Sınıflar arasında tutarlı bir davranış sağlar. • Kodun daha organize ve tutarlı olmasını sağlar.	• Karmaşıklığı artırabilir. • İsimlendirme çakışmalarına yol açabilir.

Niteliklerin Kullanım Alanları:

- Ortak işlevselliği birden fazla sınıf arasında paylaşmak için.

- Kodun daha modüler ve esnek olmasını sağlamak için.
- Sınıflar arasında kod tekrarını önlemek için.

Nitelikler ve Sınıflar Arasındaki Farklar:

- Nitelikler, doğrudan nesne oluşturamazken, sınıflar nesne oluşturabilir.
- Nitelikler, yalnızca yöntem ve özellikler içerirken, sınıflar bunlara ek olarak sabitler ve yapıcılar da içerebilir.
- Nitelikler, birden fazla sınıf tarafından kullanılabilirken, bir sınıf tek bir sınıftan miras alabilir.

PHP Statik Yöntemler

Statik yöntemler, **sınıfın bir örneği oluşturulmadan** doğrudan **sınıf adı üzerinden** çağrılabilen yöntemlerdir. Statik yöntemler, genellikle **nesneye ait olmayan** ve **sınıfın genel işlevselliğini** temsil eden işlemler için kullanılır.

Statik yöntemler, **static** anahtar kelimesiyle tanımlanır ve normal sınıf yöntemlerinden farklı olarak **$this** anahtar kelimesine erişemezler. Bunun yerine, sınıf adını (**self** anahtar kelimesiyle) veya mevcut sınıfın adını (**static** anahtar kelimesiyle) kullanırlar.

Bir sınıf hem statik hem de statik olmayan yöntemlere sahip olabilir. Statik bir yönteme aynı sınıftaki bir yöntemden **self** anahtar sözcüğü ve iki nokta üst üste (**self::**) kullanılarak erişilebilir. Bir alt sınıftan statik bir yöntem çağırmak için, alt sınıfın içinde **parent** anahtar sözcüğü kullanılır (**parent::**).

Aşağıda verilen örnekte; Islemci::toplama() ve Islemci::daireAlani() statik yöntemleri, nesne örneği oluşturmadan doğrudan sınıf adı üzerinden çağrılır. Islemci::$pi statik özelliği, self anahtar kelimesi kullanılarak statik yöntemler içerisinde erişilebilir. $x ve $y değişkenleri, statik yöntemlere parametre olarak gönderilir. Örnekteki statik yöntemler ve statik özellikler public olarak tanımlanmıştır. Private veya protected olarak da tanımlanabilirler.

```php
<?php
    class Islem {
    static public $pi = 3.14;
    static public function toplama($a, $b) {
      return $a + $b;
    }

    static public function cikarma($a, $b) {
      return $a - $b;
    }

    static public function daireAlani($r) {
      return self : :$pi * $r * $r;
    }

    echo Islem::toplama (5, 10); // 15 yazdırır
    echo "\n" ; // alt satıra geçirir
    echo Islem::cikarma (10, 5); // 5 yazdırır
```

```php
    echo "\n" ; // alt satıra geçirir
    $x = 10;
    $y = 20 ;
    echo Islem::toplama ($x, $y); // 30 yazdırır
?>
```

Statik Yöntemlerin Avantajları	Statik Yöntemlerin Dezavantajları
• Nesne oluşturma ihtiyacını ortadan kaldırarak kodun daha az yer kaplamasını sağlar. • Sınıfın genel işlevselliğini daha organize bir şekilde sunar. • Bellek kullanımını optimize etmeye yardımcı olabilir.	• Nesneye ait olmayan işlemler için uygun değildir. • Kalıtım ile aktarılamaz. • $this değişkenine erişemedikleri için daha az esnektir.

Statik Yöntemlerin Kullanım Alanları:

- Yardımcı fonksiyonlar ve matematiksel işlemler için.
- Sınıfın genel ayarlarını ve konfigürasyonlarını yönetmek için.
- Fabrika desenleri gibi tasarım örüntülerinde.

Statik Özellikler:

- Statik yöntemler gibi, statik özellikler de nesne örneği oluşturulmadan kullanılabilir.
- Sınıfın tüm nesneleri tarafından paylaşılan bir veri saklamak için kullanılır.

PHP - Statik Özellikler

Statik özellikler, **sınıfın tüm nesneleri tarafından paylaşılan** ve **nesne örneği oluşturmadan** erişilebilen veri değerleridir. Statik özellikler, **sınıfın genel durumunu** veya **ayarlarını** temsil eden verileri saklamak için kullanılır.

Statik bir özelliğe, aynı sınıftaki bir yöntemden **self** anahtar sözcüğü ve iki nokta üst üste (**self::**) kullanılarak erişilebilir. Bir alt sınıftan statik bir özellik çağırmak için, alt sınıfın içinde **parent** anahtar sözcüğü kullanılır (**parent::**).

Aşağıdaki örnekte; Islemci sınıfında $pi adında bir **statik özellik** tanımlanmıştır. Bu özellik, public olarak belirlenmiş ve 3.14 değerine atanmıştır. daireAlani() adında bir **statik yöntem** tanımlanmıştır. Bu yöntem, $r parametresini kullanarak bir dairenin alanını hesaplar. Islemci::$pi ifadesi, Islemci sınıfının $pi **statik özelliğine** doğrudan erişmek için kullanılır. Islemci::daireAlani(5) ifadesi, Islemci sınıfının daireAlani() **statik yöntemini** çağırmak için kullanılır. Yöntem, 5 değerini parametre olarak alır. $daireAlani değişkeni, daireAlani() yönteminin **dönüş değerini** saklamak için kullanılır.

```php
<?php
    class Islem
    {
        static public $pi = 3.14;
        static public function daireAlani($r)
        {
            return self::$pi * $r * $r;
        }
    }
    echo Islem::$pi; // 3.14 yazdırır
    echo "\n" ; // alt satıra geç
    $daireAlani = Islem::daireAlani(5) ;
    echo $daireAlani; // 78.5 yazdırır
?>
```

Statik Özelliklerin Avantajları	Statik Özelliklerin Dezavantajları
• Sınıfın tüm nesneleri arasında veri paylaşımı sağlar. • Kodun daha az yer kaplamasını sağlar. • Sınıfın genel durumunu ve ayarlarını yönetmek için daha kolay bir yol sunar.	• Nesneye ait olmayan veriler için uygun değildir. • Kalıtım ile aktarılamaz.

Statik Özelliklerin Kullanım Alanları:

- Sınıfın genel ayarlarını ve konfigürasyonlarını saklamak için.
- Sabit değerleri saklamak için.
- Sayaç ve istatistik gibi sınıf genelinde paylaşılan verileri saklamak için.

Statik Yöntemlerle İlişkisi:

- Statik özellikler, statik yöntemler tarafından **self** anahtar kelimesi kullanılarak erişilebilir.
- Statik yöntemler ve statik özellikler birlikte kullanılarak, sınıfın genel işlevselliği daha organize bir şekilde sunulabilir.

PHP Ad Alanları

PHP'de **ad alanları (namespaces)**, bir kodun içindeki sınıf, fonksiyon ve sabit isimlerini organize etmek için kullanılan bir mekanizmadır. Ad alanları, farklı kütüphanelerin veya bileşenlerin bir arada kullanılmasını kolaylaştırır ve çakışmaları önler. Ad alanları, kodu sınıflar, arayüzler, işlevler ve sabitleri kategorilere ayırır.

Neden Ad Alanı Kullanmalıyız?

- **Kod Organizasyonu:** Kodunuzu mantıksal birimlere ayırarak daha okunabilir ve bakımı kolay hale getirebilirsiniz.
- **İsim Çakışmalarını Önleme:** Farklı kütüphaneler veya framework'ler aynı sınıf, işlev veya sabit isimlerini kullanabilir. Ad alanları ile kendi kodunuzun isimlerini, bu kütüphanelerin isimlerinden ayırt edebilirsiniz.
- **Kod Yeniden Kullanımı:** Ad alanları ile kodunuzu hiyerarşik bir şekilde organize ederek, kod parçalarını farklı projelerde daha kolay yeniden kullanabilirsiniz.

Ad Alanı Tanımlama:

- namespace anahtar kelimesi kullanılarak tanımlanır.
- Ad alanları birbirlerine nokta (.) ile bağlanarak hiyerarşik bir yapı oluşturabilirsiniz.

Aşağıdaki örnekte; geometri isimli bir ad alanı oluşturulmuştur. Kare adında bir sınıf ve alanHesapla adında bir metot kullanarak bir karenin alanını hesaplar. pi özelliği ise tüm nesneler tarafından paylaşılan sabit bir değerdir. use anahtar kelimesi ile tanımlanan ad alanından bir sınıf çağrılmıştır.

```php
<?php
    namespace geometri;

    class Kare
    {
        static public $pi = 3.14;

        public function alanHesapla($kenar)
        {
            return self::$pi * $kenar * $kenar;
        }
    }
    use geometri\Kare;

    $kare = new Kare();
    $alan = $kare->alanHesapla(S);

    echo "Alan : " . $alan . " (pi : " . Kare::$pi . ")"; //78.5 (pi:
3.14)yazdırır
?>
```

Ad Alanı Takma Adı

Yazmayı kolaylaştırmak için bir ad alanına veya sınıfa takma ad vermek yararlı olabilir. Bu **use** anahtar sözcüğü ile yapılır. Alan adı yazıldıktan sonra **as** anahtar kelimesi ile takma ad verilir.

```php
<?php
    use Html as H;
    $table = new H\Table();
?>
```

PHP Yineleyiciler (Iterable)

PHP 7.1'den itibaren, **iterable** (yineleyiciler) pseudo-tipi, **foreach** döngüsü ile döngüye girebileceğiniz herhangi bir değeri temsil eder. Bu, diziler, nesneler, generatorler ve Traversable (dizilere benzer şekilde yinelemeye izin veren nesneler) arayüzünü uygulayan herhangi bir şey dâhil olmak üzere birçok veri türünü kapsar.

Iterable Oluşturma:

- **Diziler:** Herhangi bir dizi iterable'dır.
- **Nesneler:** Bir nesnenin iterable olması için Traversable arayüzünü uygulaması veya __iterator yöntemini tanımlaması gerekir.
- **Generatorler:** yield anahtar sözcüğünü kullanarak generator fonksiyonları iterable oluşturur.

Iterable Kullanımı:

- **foreach** döngüsü ile iterable'ları kolayca döngüye alabilirsiniz.
- **iterable** pseudo-tipini fonksiyon parametresi veya dönüş değeri olarak kullanabilirsiniz.
- **is_iterable()** fonksiyonu, bir değerin iterable olup olmadığını kontrol etmek için kullanılabilir.
- **count()** fonksiyonu, iterable'daki öğelerin sayısını döndürmek için kullanılabilir.
- **foreach** döngüsü, iterable'daki öğeleri key-value çiftleri olarak da döndürebilir.

Iterables, Iterator arayüzünü uygulayan veya __iterator yöntemini tanımlayan nesnelerdir. Bu arayüz, iterasyon (döngü) için gerekli olan bazı yöntemleri tanımlar.

Iterator Yöntemleri:

Her iterable, aşağıdaki temel Iterator arayüzü yöntemlerini uygulayarak verilere erişebilir:

- **current()**: Döngünün şu anki öğesini döndürür.
- **key()**: Döngünün şu anki öğesinin anahtarını döndürür (diziler için indeks).
- **next()**: Döngüyü bir sonraki öğeye taşır.
- **valid()**: Döngünün geçerli bir öğeye işaret edip etmediğini kontrol eder (true veya false döndürür).
- **rewind()**: Başlangıca geri döndürür.

Iterable'ların Avantajları	Iterables'ın Dezavantajları
• Daha genel bir kod yazımı sağlar. • Farklı veri türleri üzerinde aynı döngü yapısını kullanabilirsiniz. • Kodunuzu daha okunabilir ve anlaşılır hale getirir.	• Performans: Bazı durumlarda, iterable'lar doğrudan döngüye alınan veri türlerinden daha az performanslı olabilir. • Geriye dönük uyumluluk: PHP 7.1'den önce yazılmış kodlar iterable'ları desteklemez. • Hata ayıklama: Iterable'larda hata ayıklamak, doğrudan döngüye alınan veri türlerine göre daha zor olabilir.

Aşağıdaki örnekte: Sinif adında bir sınıf, Iterator arayüzünü kullanarak iterable hale getirilir. degerler dizisi, pozisyon ile takip edilir. __construct metodu, degerleri başlatır. current, key, next ve valid metotları, foreach döngüsü ile degerler üzerinde döngüye girilmesini sağlar. rewind metodu opsiyoneldir. foreach döngüsü, her öğenin anahtarını ve değerini ekrana yazdırır.

```php
<?php
    class Sinif implements Iterator {
        private $degerler = [];
        private $pozisyon = 0;

        public function _construct () {
           $this->degerler = [1, 2, 3, 4, 5];
        }

        public function current() {
```

```
            return $this->degerler [$this->pozisyon];
        }

        public function key() {
            return $this->pozisyon;
        }

        public function next() {
            $this->pozisyon++ ;
        }

        public function valid () {
            return isset($this->degerler[$this->pozisyon]) ;
        }

        public function rewind() {
            $this->pozisyon = 0;
        }
    }
  $sinif = new Sinif ();
  foreach ($sinif as $key => $deger) {
    echo "Değer ($key): " . $deger , "<br>";

?>
```

BÖLÜM ÖZETİ

Nesne tabanlı programlama yani NYP, nesnelerin özellikleri, işlevleri ve olayları ile tanımlanır ve si plus plus, Java gibi dillerde kullanılır. Prosedürel programlamadan farklı olarak, veri ve fonksiyonları bir arada tutarak daha düzenli ve yeniden kullanılabilir kod yapısı sağlar. NYP'nin avantajları arasında hızlı yürütme, net yapı, "Kendini Tekrar Etmeme" ilkesine uyum ve geliştirme süresinin kısalığı yer alır. Temel prensipleri kapsülleme, kalıtım ve çok biçimliliktir. bunlar sırasıyla kodun gizlenmesi, sınıflar arası özellik aktarımı ve aynı işlemin farklı nesnelerde farklı sonuçlar vermesini sağlar. PHP'de sınıflar, belirli işlevleri yerine getiren ve kodun düzenli ve tekrar kullanılabilir olmasını sağlayan yapılar olarak tanımlanır. PHP'5 ile geliştirilen yeni Nesne Modeli sayesinde, veritabanı işlemleri gibi spesifik görevler için sınıflar oluşturulabilir. Sınıflar, sabitler, değişkenler ve fonksiyonlar içerir ve class anahtar kelimesiyle tanımlanır. Her sınıf örneği bir nesne olarak adlandırılır ve new anahtar kelimesiyle oluşturulur. Sınıf adları, PHP için ayrılmış kelimeler olmamalı ve belirli kurallara uygun olmalıdır. dolar this anahtar sözcüğü, yöntemler içinde geçerli nesneyi temsil eder. Sınıfların oluşturulmasıyla otomatik olarak çalışan yapıcı metotlar çoklu parantez içinde construct ve nesnelerin bellekten kaldırılmasıyla çalışan yıkıcı metotlar çoklu parantez içinde destruct bulunur. Yapıcı metotlar, sınıf örneklerinin başlatılması için kullanılırken, yıkıcı metotlar kaynakların serbest bırakılması ve belleğin temizlenmesi işlemleri için devreye girer. Her ikisi de, sınıf tanımlamalarının verimliliğini ve kodun yönetilebilirliğini artırır. Erişim belirleyiciler yani public, private, protected, sınıf üyelerinin erişilebilirlik düzeyini tanımlar. Public üyelere her yerden, private üyelere sadece tanımlandıkları sınıf içinden, protected üyelere ise tanımlandıkları sınıf ve miras alan sınıflardan erişilebilir. Varsayılan olarak, belirleyici belirtilmezse üyeler public olur. Erişim belirleyicileri, kod güvenliğini artırır, değişiklik yönetimini ve bakımını kolaylaştırır.

PHP'de kalıtım, yeni sınıfların mevcut sınıfların özelliklerini ve yöntemlerini devralmasını sağlayan bir OOP kavramıdır. Kalıtım, kodun yeniden kullanılabilirliğini artırır, yedekliliği azaltır ve hiyerarşik düzenlemeyi kolaylaştırır. PHP, tek kalıtımı destekler ve extends anahtar kelimesiyle alt sınıflar üst sınıflardan türetilir. Kalıtım, kod güvenliği ve bakım kolaylığı sağlar, değişiklik yönetimini kolaylaştırır ve polimorfizm için temel oluşturur. Korumalı erişim değiştiricisi yani protected, sınıf üyelerinin sadece sınıfın kendisinden ve miras alan alt sınıflardan erişilebilir olmasını sağlar. Miras yöntemleri geçersiz kılma, alt sınıfların üst sınıftan devralınan yöntemlerin davranışını yeniden tanımlamasına olanak tanır. final anahtar kelimesi, bir sınıfın veya yöntemin başka sınıflar tarafından türetilmesini veya geçersiz kılınmasını engeller. Sınıf sabitleri, const anahtar kelimesi ile tanımlanan ve sınıfın tüm nesneleri tarafından paylaşılan değiştirilemez değerlerdir. Sabitlere, sınıf adı ve çift iki nokta üst üste operatörü kullanılarak veya bir nesne örneği üzerinden erişilebilir. Sabitler, değiştirilemez, tüm nesneler tarafından paylaşılan ve genellikle büyük harflerle adlandırılan statik değerlerdir. Sabitlerin kullanımı, kodun okunabilirliğini ve anlaşılırlığını artırır, tekrarlayan değerlerin merkezi bir noktada yönetilmesini sağlar ve kod bakımını kolaylaştırır. Sabitlere sınıf içinden self anahtar kelimesi ve kapsam çözümleme operatörü parantez içinde çift iki nokta üst üste ile erişilir. Soyut sınıflar ve yöntemler, nesne yönelimli programlamada kullanılır.

Soyut sınıflar, abstract anahtar kelimesiyle tanımlanır ve kendileri örneklenemezler ancak diğer sınıflar için bir temel oluştururlar. Soyut yöntemler, alt sınıfların uygulaması gereken, gövdesiz yöntemlerdir. Alt sınıflar, soyut sınıflardan miras alırken bu soyut yöntemleri gerçekleştirmek zorundadır, böylece kod daha modüler ve esnek hâle gelir. Arayüzler, sınıfların uygulaması gereken yöntemleri tanımlayan sözleşmelerdir. Inter face anahtar kelimesiyle tanımlanır ve sınıflar tarafından implements ile uygulanır. Arayüzler, bir sınıfın belirli davranışları nasıl gerçekleştireceğini değil, hangi yöntemleri içermesi gerektiğini belirtir, böylece kodun modülerliğini ve esnekliğini artırır. Arayüzler yalnızca soyut yöntemler içerir ve bu yöntemlerin gövdeleri tanımlanamaz; uygulayan sınıflar bu yöntemleri kendi içerikleriyle doldurmalıdır.

PHP'de, nitelikler yani traits, birden fazla sınıf arasında paylaşılan yöntem ve özellikleri tanımlayan yapılar olarak kullanılır. trait anahtar kelimesiyle tanımlanır ve use ile sınıflara dâhil edilir. Nitelikler, kodun modülerliğini ve esnekliğini artırırken, sınıflar arasında işlevsellik paylaşımını sağlar ve kod tekrarını azaltır. Nitelikler doğrudan nesne oluşturamaz ve yalnızca yöntem ve özellikler içerir, sınıflar ise sabitler ve yapıcılar da dâhil olmak üzere daha geniş bir yelpazede özelliklere sahip olabilir. Bir sınıf, birden fazla niteliği kullanabilirken, yalnızca bir sınıftan miras alabilir. Statik yöntemler ve özellikler, sınıf örnekleri olmadan kullanılabilen sınıf bileşenleridir. Statik yöntemler, dolar this yerine "self" veya "static" anahtar kelimeleri ile sınıfın kendisine erişirken, statik özellikler sınıfın tüm örnekleri arasında paylaşılan verileri saklar. Her ikisi de self çift iki nokta üst üste veya parent çift iki nokta üst üste kullanılarak çağrılabilir ve sınıfın genel işlevselliği için kullanılır. PHP'de ad alanları, kodun düzenli ve çakışmasız bir şekilde organize edilmesini sağlayan bir yapıdır. Ad alanları sayesinde, farklı kütüphanelerden gelen benzer isimlerdeki sınıf, fonksiyon ve sabitler arasında ayrım yapılabilir. Kodun okunabilirliğini ve bakımını kolaylaştırır, aynı zamanda kodun farklı projelerde yeniden kullanılmasına olanak tanır. Ad alanları namespace anahtar kelimesiyle tanımlanır ve use ile takma ad verilebilir, bu da kod yazımını daha da basitleştirir. PHP'de yinelebilirler yani iterable, PHP'7.1 ile tanıtılmış olup, foreach döngüsünde kullanılabilen her türlü değeri ifade eder.

GÖZDEN GEÇİRELİM

1. Aşağıdaki kod parçasıyla ilgili olarak hangi ifade doğrudur?

```php
<?php
class Hayvan {
	public $name;
	public function __construct($name) {
	$this->name = $name;
	}
	public function sound() {
	return "hav hav!";
	}
}

class Kopek extends Hayvan {
	public function sound() {
	return "hav!";
	}
}
$dog = new Dog("Karabaş");
echo $dog->name . $dog->sound()." dedi." ;
```

A) "Karabaş hav hav! dedi." çıktısı verir.

B) "Karabaş hav! dedi." çıktısı verir.

C) Kod hata verir.

D) "dedi. hav! Karabaş" çıktısı verir.

E) "hav! dedi. Karabaş" çıktısı verir.

2. PHP'de bir sınıfın constructor metodu tanımlanırken hangi isim kullanılır?

A) setup()

B) __init()

C) constructor()

D) __construct()

E) create()

3. Aşağıdakilerden hangisi PHP’de kalıtımı engelleyen anahtar kelimedir?

A) final

B) static

C) private

D) protected

E) public

4. PHP’de bir sınıfın başka bir sınıftan türemesi için hangi anahtar kelime kullanılır?

A) extends

B) implements

C) inherits

D) uses

E) requires

5. Bir abstract sınıf içindeki abstract metodlar ne tür erişim belirleyicilere sahip olabilir?

A) Yalnızca private

B) Yalnızca public veya protected

C) Yalnızca public

D) Her türlü erişim belirleyiciye sahip olabilir

E) Yalnızca protected

6. PHP’de bir arayüz tanımlamak için hangi anahtar kelime kullanılır?

A) class

B) struct

C) interface

D) function

E) implement

7. Aşağıdaki kod parçasında, $obj nesnesi hangi sınıfın trait'ini kullanır?

```php
<?php
trait MyTrait {
    public function sayHello() {
    echo "Hello!";
    }
}

class MyClass {
use MyTrait;
}
$obj = new MyClass();
$obj->sayHello();
?>
```

A) MyTrait

B) MyClass

C) MyClassTrait

D) sayHello

E) HelloTrait

8. Bir namespace içindeki sınıf veya fonksiyonlara erişmek için hangi operatör kullanılır?

A) ::

B) .

C) ->

D) =>

E) ::

CEVAP ANAHTARI

1	2	3	4	5	6	7	8
B	D	A	A	B	C	A	A

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 - OOP (n.d.). https://www.w3schools.com/php/php_oop_what_is.asp

PHP - Class/Object Information (n.d.). https://www.php.net/manual/en/book.classobj

PHP - Construct Functions (n.d.). https://www.w3schools.com/php/php_oop_constructor.asp

PHP - Destruct Functions (n.d.). https://www.w3schools.com/php/php_oop_destructor.asp

PHP - Access Modifiers (n.d.). https://www.w3schools.com/php/php_oop_access_modifiers.asp

PHP - Inheritance (n.d.). https://www.w3schools.com/php/php_oop_inheritance.asp

PHP - Traits? (n.d.). https://www.w3schools.com/php/php_oop_traits.asp

PHP - Static Methods (n.d.). https://www.w3schools.com/php/php_oop_static_methods.asp

PHP - Static Properties (n.d.). https://www.w3schools.com/php/php_oop_static_properties.asp

PHP - Namespaces (n.d.). https://www.w3schools.com/php/php_namespaces.asp

PHP - Iterables (n.d.). https://www.w3schools.com/php/php_iterables.asp

BÖLÜM 3

PHP'DE MYSQL İLE VERİTABANI İŞLEMLERİ

ANAHTAR KAVRAMLAR

Veri tabanı, MySQL, Tablo, Alan, Kayıt

ÖĞRENME HEDEFLERİ

1. MySQL veri tabanı hakkında bilgi sahibi olma
2. Veri tabanı bağlantısı oluşturma
3. Veri tabanında tablo oluşturma
4. Veri tabanından istenilen verileri getirme
5. Verileri sıralı bir şekilde getirme
6. Veri ekleme, silme, güncelleme

GİRİŞ

Veri tabanı, organize edilmiş bilgi koleksiyonu olarak tanımlanmaktadır. Bu bilgiler genellikle belirli bir konuyla ya da bir mimari ile ilgilidir. Söz konusu veriler bir sistem tarafından erişilip yönetilir. Veri tabanları, verileri saklamak, düzenlemek, güncellemek ve sorgulamak için kullanılmaktadır. Büyük veya küçük ölçekte birçok sektör için önemli olan veri tabanlarının uygulama alanları ile ilgili bankalar, e-ticaret siteleri, müşteri ilişkileri yönetimi (CRM) yazılımları gibi birçok örnek verilebilir. Veri tabanı sistemleri, bilgilerin organize edilmesi, saklanması, erişilmesi, güncellenmesi ve yönetilmesi için günümüzün vazgeçilmez mimarileridir. Şekil 1 veri tabanı yönetim sistemlerinin önemi, özellikleri ve kullanılmasının temel nedenleri başlıklar halinde sunmaktadır.

Şekil 1. Veri tabanı yönetim sistemlerinin özellikleri

Veri yönetimi bakımından incelemek gerekirse, veri tabanları yapılandırılmış bir şekilde verilerin saklanmasını sağlamaktadır. Dolayısıyla verilerin tutarlılığı ve bütünlüğü korunmaktadır. Tablolar, ilişkiler ve kısıtlamalar aracılığıyla veriler düzenlenir ve bu da verilerin daha kolay yönetilmesini olanak tanımaktadır. Diğer bir durum ise söz konusu sitemler sayesinde aynı anda birden fazla kullanıcının verilere erişmesi ve güncellemesi mümkün olmaktadır. Bahsi geçen durum ise iş süreçlerinin verimliliğini artırmaktadır. Güvenlik ise veri tabanı sistemlerinin kullanımını zorunluluk haline getiren hayati konulardan biridir. Veri tabanlarında bulunan yetkilendirme, kimlik doğrulama ve şifreleme gibi önemli güvenlik özelliklerinin bulunması yetkisiz erişimlerin engellenmesini sağlamaktadır ve verilerin gizliliği korumaktadır. İlaveten, veri tabanları verilerin yedeklenmesini ve felaket durumlarında kurtarılmasını sağlayan mekanizmalar bulundurmaktadırlar. İlgili durum veri kaybını önler ve iş sürekliliğini sağlar. Veri analizi ise incelenmesi gereken önemli konulardan bir tanesidir. Veri tabanları sorgu dilleri aracılığıyla verilerin analiz edilmesine ve raporların oluşturulmasına olanak tanımaktadır. Böylece iş karar ve süreçlerinin veriye dayalı olarak uygulanmasını sağlamaktadır. Farklı kaynaklardan gelen verilerin birleştirilmesi ve entegrasyonu da veri tabanlarının önemli bir kullanım alanıdır; örneğin, bir şirketin farklı departmanlarından gelen verilerin bir araya getirilmesi ve ortak bir platformda kullanılması mümkündür. Bu nedenlerle, veri tabanları işletmelerde, web sitelerinde, mobil uygulamalarda, bilimsel araştırmalarda ve birçok başka alanda yaygın olarak kullanılmaktadır, veri yönetimi ve iş süreçlerinin etkin bir şekilde yürütülmesine yardımcı olmaktadır.

Farklı gereksinimlere ve kullanım senaryolarına yönelik çeşitli veri tabanı yönetim sistemleri bulunmaktadır. Hangi veri tabanı yönetim sisteminin sizin için en uygun olduğunu belirlemek, ihtiyaçlarınıza ve projenizin gereksinimlerine bağlı olarak değişkenlik gösterecektir. Veri

tabanı yönetim sistemleri için MySql, Oracle Database, Microsoft SQL Server, PostgreSQL, MongoDB, SQLite gibi popüler örnekler verilebilir. Bu bölümde veri tabanı bağlantısı oluşturma, veri tabanı ve tablo oluşturma, veri ekleme, silme seçme, güncelleme gibi işlemler anlatılacak ve örnekler verilecektir. Ancak öncelikle kısaca MySql veri tabanı yönetim sistemi ile ilgili kısaca bilgi verelim.

MySQL Veri Tabanı

MySql, web geliştirme alanında önemli bir rol oynayan popüler bir ilişkisel veritabanı yönetim sistemidir (RDBMS). MySql'in, ücretsiz ve açık kaynaklı olması küçük işletmelerden büyük kuruluşlara kadar birçok kullanıcı için onu tercih edilebilir kılmaktadır. Açık kaynaklı doğası, geniş bir topluluğun katkıda bulunmasını sağlamaktadır ve hızla gelişmesine olanak tanımaktadır. Hızlı ve verimli bir veritabanı yönetim sistemi olarak bilinen MySql, optimize edilmiş sorgu performansı ve düşük sistem kaynakları gereksinimi sayesinde web uygulamaları için ideal bir seçenek olarak öne çıkmaktadır. MySql, web uygulamaları, içerik yönetim sistemleri (CMS) ve e-ticaret platformları gibi çeşitli web tabanlı projelerde yaygın olarak kullanılmaktadır. Ayrıca, MySql Linux, Windows, macOS ve diğer birçok işletim sistemi üzerinde çalışabilir, bu da farklı platformlarda çalışan web uygulamaları için geniş bir uyumluluk sağlamaktadır. MySql, güvenlik duvarları, şifreleme ve yetkilendirme gibi bir dizi güvenlik özelliği sunmaktadır, böylece veri güvenliği sağlanır. Ölçeklenebilirlik açısından, MySql küçük projelerden büyük ölçekli kurumsal uygulamalara kadar geniş bir ölçeklenebilirlik aralığını destekler. Yüksek trafikli web siteleri ve uygulamaları için bile uygun performansı ve ölçeklenebilirliği sağlar. MySql'in geniş bir dokümantasyon ve aktif bir topluluğu vardır, kullanıcılar belgelerden öğrenme kaynaklarına kadar çeşitli kaynaklardan yararlanabilir ve sorunlarını çözmek için topluluğun desteğinden faydalanabilirler. Bu nedenlerle, MySql web geliştirme alanında yaygın olarak kullanılan bir veri tabanı yönetim sistemidir, hem yeni başlayanlar hem de deneyimli geliştiriciler tarafından tercih edilir ve çeşitli web tabanlı projelerde güvenilir bir veri tabanı çözümü olarak kabul edilmektedir. Bu bölümde aşağıdaki başlıklarda incelenen konuların tümü için MySql veri tabanı yönetim sistemi kullanılmıştır.

Veri Tabanı Bağlantısı

PHP ile veritabanına bağlanmak için iki yaygın yöntem vardır: MySqli (MySql Improved) ve PDO (PHP Data Objects). MySqli, PHP'nin MySql veritabanlarıyla etkileşim kurmak için sunulan geliştirilmiş bir API'dir. MySqli, PHP 5.0'dan itibaren kullanılabilir hale gelmiştir. MySqli'nin özellikleri arasında prosedürel ve nesne yönelimli programlama (OOP) tarzında kullanım, hazırlanmış ifadeler (prepared statements), işlem (transactions) ve çıktı parametreleri gibi özellikler bulunur. Kitabın ilerleyen bölümlerindeki örnekler sırasıyla Örnek 1:Mysqli nesne yönelimli, Örnek 2: Mysqli prosedürel ve Örnek 3: PDO kullanımlarını içermektedir. Aşağıda verilen örnek bir MySqli bağlantısını göstermektedir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// MySQLi bağlantısı oluşturma
$conn = new mysqli($servername, $username, $password);
// Bağlantıyı kontrol etme
if ($conn->connect_error) {
    die("Bağlantı hatası: " . $conn->connect_error);
} else {
    echo "Bağlantı başarılı";
}
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// MySQLi bağlantısı oluşturma
$conn = mysqli_connect($servername, $username, $password);
// Bağlantıyı kontrol etme
if (!$conn) {
    die("Bağlantı hatası: " . mysqli_connect_error());
} else {
    echo "Bağlantı başarılı";
}
?>
```

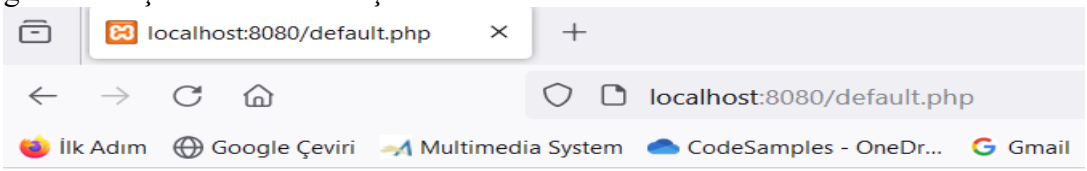
Bu kod, MySql veritabanına MySqli (MySql Improved Extension) kullanarak bağlanmayı denemektedir. İlk olarak, bağlantı parametreleri ($servername, $username, $password,) tanımlanır. Daha sonra, bu parametreler kullanılarak new mysqli() ile bir MySqli bağlantısı oluşturulur.

Bağlantının başarılı olup olmadığı if koşulu ile kontrol edilir. Eğer bağlantı hatası varsa ($conn->connect_error true ise), die() fonksiyonu ile birlikte bir hata mesajı gösterilir ve kod çalışması durdurulur. Aksi takdirde (else bloğu), "Bağlantı başarılı" mesajı ekrana yazdırılır.
if bloğunun içinde die() fonksiyonu kullanılır. Bu fonksiyon, belirtilen bir mesajı ekrana yazdırır ve işlemin çalışmasını durdurur. Yani, eğer bağlantı hatası varsa, kullanıcıya "Bağlantı hatası: [hata_mesajı]" şeklinde bir mesaj gösterilir ve kod çalışmayı durdurur. Eğer bağlantı hatası yoksa, else bloğu çalıştırılır. Bu durumda, echo komutuyla ekrana "Bağlantı başarılı" mesajı yazdırılır. Yani, eğer bağlantı başarılıysa, kullanıcıya bu bilgi verilir. Bu yapı, MySql veritabanıyla bağlantı kurarken oluşabilecek hataları kontrol ederek kullanıcıya uygun bir geri bildirim sağlar. PDO ise farklı veri tabanı türleriyle etkileşim kurmak için kullanılan bir erişim katmanıdır. PDO, çeşitli veritabanlarına (MySql, PostgreSQL, SQLite vb.) bağlanmak için kullanılabilir. Örnek bir PDO bağlantısı aşağıda verilmiştir.

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
try {
    // PDO bağlantısı oluşturma
    $conn = new PDO("mysql:host=$servername;dbname=$database", $username,
$password);
    // Hata modunu ayarlama
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Bağlantı başarılı";
} catch(PDOException $e) {
```

PDO yöntemi kullanılarak yapılan bağlantının bazı farkları bulunmaktadır. `$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION)` satırı PDO'nun hata işleme modunu ayarlar. setAttribute() yöntemi kullanılarak PDO'nun belirli bir özelliği ayarlanır. İlk parametre olarak PDO::ATTR_ERRMODE sabiti seçilir, bu sabit PDO'nun hata işleme modunu ayarlamak için kullanılır. İkinci parametre olarak PDO::ERRMODE_EXCEPTION sabiti kullanılır; bu sabit, PDO'nun hata işleme modunun istisna fırlatma modu olarak ayarlandığını belirtir. Yani, eğer bir PDO işlemi başarısız olursa (örneğin, bir sorgu hatası), bir istisna fırlatılır ve bu istisna, kodun işleyişi normalden farklı bir şekilde devam etmesini sağlar. Bu kod parçası, PDO'nun hata işleme modunu ayarlayarak, PDO'nun hata durumlarını ele almasını ve kodun daha güvenli ve kontrol edilebilir olmasını sağlar. Özellikle geliştirme aşamasında, bu modu kullanmak hata ayıklamayı kolaylaştırabilir ve uygulamanın güvenilirliğini artırabilir. Verilen kodlarda görüldüğü üzere her iki bağlantı çeşidi de sunucu adı (servername), kulanıcı adı (username), şifre (password) ve veritabanı (database) olmak üzere 4 adet parametre almaktadır. Varsayılan olarak sunucuadı=”localhost”, username=”root” değerleri almıştır Yukarıda verilen her iki kodun çalıştırılması sonucu elde edilen sayfa görüntüsü Şekil 2’de verilmiştir.



Şekil 2. MySqli ve PDO örneklerinin ekran görüntüsü

Ek Bilgi: PDO ve MySQLi arasındaki tercih, projenin ihtiyaçlarına ve kişisel tercihlere bağlıdır. PDO, farklı veritabanı motorlarını desteklediği için genellikle daha esnek ve taşınabilir bir seçenektir. Ancak, MySQLi'nin MySQL'e özgü optimizasyonlar ve özellikler sağladığı da unutulmamalıdır.

Veri Tabanı Oluşturma

Bildiğiniz üzere veri tabanı birçok tablodan meydana gelir. Bahsi geçen tabloları ilişkisel olarak tutmak ve tablolara veri ekleyebilmek için öncelikle bu tabloların içerisinde olduğu bir

veri tabanı gerekmektedir. MySql veri tabanı yönetim sisteminde veri tabanı yaratmak için ***create*** deyimi kullanılmaktadır. Örneğin Ankuzef isimli bir veri tabanı oluşturmak için;

```
CREATE DATABASE Ankuzef;
```

şeklinde kod yazmak gerekmektedir. Söz konusu veri tabanı oluşturma komutu MySql platformunda yazıldığı gibi bir php kodu ile de yazılıp sunucuda bir veri tabanı oluşturulması sağlanabilir. MySql veritabanı oluşturmak için hem MySqli hem de PDO kullanılabilmektedir. Sırasıyla her iki yöntemle de MySql veritabanı oluşturmanın kodları aşağıda verilmiştir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// bağlantı oluşturmak
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
// veri tabanı oluşturmak
$sql = "CREATE DATABASE Ankuzef";
if ($conn->query($sql) === TRUE) {
  echo "Veri tabanı Başarılı Bir Şekilde Oluşturuldu!!";
} else {
  echo "Veri Tabanı Oluşturulamadı: " . $conn->error;
}
$conn->close();
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
// bağlantı oluşturmak
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
// veri tabanı oluşturmak
$sql = "CREATE DATABASE Ankuzef";
if (mysqli_query($conn,$sql)) {
  echo "Veri tabanı Başarılı Bir Şekilde Oluşturuldu!!";
} else {
  echo "Veri Tabanı Oluşturulamadı: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
try {
  $conn = new PDO("mysql:host=$servername", $username, $password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "CREATE DATABASE Ankuzef";
  // hiçbir sonuç döndürülmediği için exec() kullanın
  $conn->exec($sql);
  echo "Veri Tabanı Başarılı Bir Şekilde Oluşturuldu<br>";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}
$conn = null;
?>
```

Tablo Oluşturma

Tablo oluşturmak, ilişkisel veritabanlarında veri saklamak için temel bir işlemdir. MySql gibi ilişkisel veritabanı yönetim sistemlerinde bir tablo, belirli bir veri tipine sahip bir dizi sütun ve bu sütunlarla ilişkilendirilmiş veri kayıtlarından oluşur. Bir tablo oluşturmak için bazı bileşenlere dikkat etmek gerekmektedir. Her tablo, veri tabanında benzersiz bir ad ile tanımlanması gerekmektedir. Bir tablonun her bir sütunu, sakladığı veri türünü belirtir. Örneğin, bir kullanıcı tablosu oluştururken, ad, soyad, e-posta gibi bilgileri saklayan sütunlar tanımlanabilir. Her sütunun belirli bir veri tipi olmalıdır.

Tablo oluşturmak için MySql'de CREATE TABLE ifadesi kullanılır. kullanicilar isimli bir tablo oluşturmak için SQL kodu aşağıdaki gibidir.

```sql
CREATE TABLE Kullanicilar (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    ad VARCHAR(30) NOT NULL,
    soyad VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    kayit_tarihi TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

Hem MySqli hem de PDO kullanarak MySql tablosu oluşturmak oldukça basittir. İlk olarak, her iki yöntemle de bağlantı kurmamız gerekmektedir. Ardından, SQL sorgularını kullanarak tabloyu oluşturabiliriz. Bahsi geçen örnekler aşağıda verilmiştir.

Örnek 1:

```php
<?php
// MySQL bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = " Ankuzef";
$conn = new mysqli($servername, $username, $password, $dbname);
// Bağlantı kontrolü
if ($conn->connect error) {
```

Örnek 2:

```php
<?php
// MySQL bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = " Ankuzef";

$conn = mysqli_connect($servername, $username, $password, $dbname);
// Bağlantı kontrolü
if (!$conn) {
    die("Bağlantı hatası: " . mysqli_connect_error());
}
// Tablo oluşturma sorgusu
$sql = "CREATE TABLE kullanicilar (
    id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    ad VARCHAR(30) NOT NULL,
    soyad VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    kayit_tarihi TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
)";
// Sorguyu çalıştırma
if (mysqli_query($conn,$sql)) {
    echo "Tablo başarıyla oluşturuldu.";
} else {
    echo "Tablo oluşturma hatası: " . mysqli_error($conn);
}
// Bağlantıyı kapatma
mysqli_close($conn);
?>
```

Örnek 3.

```php
<?php
// PDO bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // Hata modunu ayarlama: PDOException fırlatma
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // Tablo oluşturma sorgusu
    $sql = "CREATE TABLE kullanicilar (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        ad VARCHAR(30) NOT NULL,
        soyad VARCHAR(30) NOT NULL,
        email VARCHAR(50),
        kayit_tarihi TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
    )";
    // Sorguyu çalıştırma
    $conn->exec($sql);
    echo "Tablo başarıyla oluşturuldu.";
} catch(PDOException $e) {
    echo "Tablo oluşturma hatası: " . $e->getMessage();
}
// Bağlantıyı kapatma
$conn = null;
?>
```

Veri Ekleme

MySql'de INSERT ifadesi, bir tabloya yeni bir kayıt eklemek için kullanılır. INSERT INTO ifadesi, veri eklenmek istenen tabloyu belirtir ve VALUES ifadesi, eklenen verilerin değerlerini belirtir. kullanicilar isimli tabloya herhangi bir kayıt eklemek için gerekli sorgu aşağıda verilmiştir.

```
INSERT INTO kullanicilar (ad, soyad, email) VALUES ('mehmet', 'nallıhan',
'nallihan@example.com');
```

Yukarıda verilen kullanicilar tablosuna veri eklemek için gerekli olan MySqli ve PDO kodları ise aşağıda verilmiştir.

Örnek 1:

```php
<?php
// MySQL bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
$conn = new mysqli($servername, $username, $password, $dbname);
// Bağlantı kontrolü
if ($conn->connect_error) {
    die("Bağlantı hatası: " . $conn->connect_error);
}
// Veri ekleme sorgusu
$sql = "INSERT INTO kullanicilar (ad, soyad, email) VALUES ('mehmet',
'nallihan', 'nallihan@example.com')";
// Sorguyu çalıştırma
if ($conn->query($sql) === TRUE) {
    echo "Yeni kayıt başarıyla eklendi.";
} else {
    echo "Veri ekleme hatası: " . $conn->error;
}
// Bağlantıyı kapatma
$conn->close();
?>
```

Örnek 2:

```php
<?php
// MySQL bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Bağlantı kontrolü
if (!$conn) {
    die("Bağlantı hatası: " . mysqli_connect_error());
}
// Veri ekleme sorgusu
$sql = "INSERT INTO kullanicilar (ad, soyad, email) VALUES ('mehmet',
'nallihan', 'nallihan@example.com')";
// Sorguyu çalıştırma
if (mysqli_query($conn,$sql)) {
    echo "Yeni kayıt başarıyla eklendi.";
} else {
    echo "Veri ekleme hatası: " . mysqli_error($conn);
}
// Bağlantıyı kapatma
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
// PDO bağlantısı
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
    // Hata modunu ayarlama
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // Veri ekleme sorgusu
    $sql = "INSERT INTO kullanicilar (ad, soyad, email) VALUES ('mehmet',
'nallihan', 'nallihan@example.com')";

    // Sorguyu hazırlama ve çalıştırma
    $stmt = $conn->prepare($sql);
    $stmt->execute();

    echo "Yeni kayıt başarıyla eklendi.";
} catch(PDOException $e) {
    echo "Veri ekleme hatası: " . $e->getMessage();
}
// Bağlantıyı kapatma
$conn = null;
```

Çoklu Kayıt Ekleme

Birden fazla SQL ifadesi mysqli_multi_query() fonksiyonu ile yürütülmelidir. Aşağıdaki örnekler kullanicilar isimli tabloya 3 yeni kayıt eklemektedir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
// Bağlantı Oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// Bağlantıyı Kontrol Et
if ($conn->connect_error) {
  die("Başarısız Bağlantı: " . $conn->connect_error);
}
$sql = "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('ahmet', 'ahmet', 'ahmet@example.com');";
$sql .= "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('mehmet', 'mehmet', 'mehmet@example.com');";
$sql .= "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('hasan', 'hasan', 'hasan@example.com')";
if ($conn->multi_query($sql) === TRUE) {
  echo "Yeni kayıtlar eklendi";
} else {
  echo "Hata: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// Bağlantı Oluştur
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Bağlantıyı Kontrol Et
if (!$conn) {
  die("Başarısız Bağlantı: " . mysqli_connect_error());
}

$sql = "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('ahmet', 'ahmet', 'ahmet@example.com');";
$sql .= "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('mehmet', 'mehmet', 'mehmet@example.com');";
$sql .= "INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('hasan', 'hasan', 'hasan@example.com')";

if (mysqli_multi_query($conn, $sql)) {
  echo "Yeni kayıtlar başarılı bir şekilde eklendi";
} else {
  echo "Hata: " . $sql . "<br>" . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // işlemi başlatın
  $conn->beginTransaction();
  // Sql Sorguları
  $conn->exec("INSERT INTO kullanicilar (ad, soyad, email)
VALUES ('ahmet', 'ahmet', 'ahmet@example.com');");
  $conn->exec("INSERT INTO kullanicilar (ad, soyad, email)
  VALUES ('mehmet', 'mehmet', 'mehmet@example.com')");
  $conn->exec("INSERT INTO kullanicilar (ad, soyad, email)
  VALUES ('hasan', 'hasan', 'hasan@example.com')");
  // işlemi gerçekleştirin
  $conn->commit();
  echo "Yeni kayıtlar başarılı bir şekilde eklendi!!!";
} catch(PDOException $e) {
  // İşlemin başarısız olduğu durumda geri alma
  $conn->rollback();
  echo "Hata: " . $e->getMessage();
}
$conn = null;
?>
```

Yukarıda örneği verilen PDO kullanımını açıklamak yerinde olacaktır. PDO kullanarak gerçekleştirilen bir işlemi (transaction) tamamlamak için **commit()** yöntemini kullanır. Bir işlem (transaction) başarıyla tamamlandığında, veritabanındaki tüm değişiklikler kalıcı hale gelir ve veritabanı kilidi kaldırılır. Ardından, kullanıcıya "Yeni kayıtlar başarılı bir şekilde eklendi" mesajı gösterilir. Ancak, **try** bloğundaki kodda bir hata oluştuğunda, bu kod **catch** bloğuna girer. Bu durumda, **rollback()** yöntemi çağrılarak işlem (transaction) geri alınır ve yapılan tüm değişiklikler geri alınır, böylece veritabanı önceki durumuna döner. Son olarak, kullanıcıya bir hata mesajı gösterilir. Bu mesaj, **PDOException** sınıfından türetilen hatanın **getMessage()** yöntemi aracılığıyla alınır ve kullanıcıya gösterilir.

Hazırlanan İfadeler ve Bağlı Parametreler

Hazırlanan ifadeler (prepared statements) ve bağlı parametreler, veritabanı sorgularını güvenli ve etkili bir şekilde çalıştırmak için kullanılan bir tekniktir. Bu teknik, SQL enjeksiyon saldırılarına karşı koruma sağlar ve sorguların daha verimli bir şekilde çalışmasına olanak tanır.

Hazırlanan İfadeler (Prepared Statements):

Hazırlanan ifadeler, bir veritabanı sorgusunun önceden derlenmiş bir kalıba (template) sahip olduğu ve bu kalıbın veritabanına gönderilmeden önce doldurulduğu sorgu türüdür. Bu sayede,

sorgunun güvenliği artırılır ve aynı sorgu farklı parametrelerle tekrar tekrar kullanılabilir. Hazırlanan ifadeler genellikle veritabanı sunucusunda sorgunun ön işlemesi (pre-processing) yapılır, böylece sorgular daha hızlı çalışabilir. Hazırlanan ifadeler genellikle iki kısımdan oluşur:

1. **Sorgu kalıbı**: Sorgunun sabit kısmı, değişmeyen kısmıdır. Genellikle parametre yerine bir placeholder (yer tutucu) olarak kullanılır.
2. **Parametreler**: Sorgunun değişken kısmı, her sorguda değişen kısımdır. Bu parametreler sorgu gönderilmeden önce sorgu kalıbına yerleştirilir.

Bağlı Parametreler (Bound Parameters):

Bağlı parametreler, hazırlanan ifadelerin parametrelerine değer atanmasında kullanılan değerlerdir. Bağlı parametreler, sorgu gönderilmeden önce sorgu kalıbına yerleştirilir ve sorgunun çalıştırılması sırasında bu parametreler sorguya bağlanır. Bağlı parametreler, veritabanı sunucusuna değerlerin güvenli bir şekilde aktarılmasını sağlar ve SQL enjeksiyon saldırılarını önler.

Bağlı parametreler genellikle kullanılan programlama dili ve veritabanı sorgu arayüzüne göre farklılık gösterebilir. Örneğin, PHP ile PDO veya MySQLi kullanırken bağlı parametreler "?" veya ":name" gibi bir placeholder olarak ifade edilir ve sorgu çalıştırılırken parametreler bu placeholder'lara bağlanır. Hazırlanan ifadeler ve bağlı parametreler, veritabanı sorgularını güvenli hale getirirken aynı zamanda performansı da artırır. Bu teknikler, modern web uygulamalarında yaygın olarak kullanılan güvenlik önlemlerindendir. Hazırlanmış ve bağlı ifadelere ait örnekler sırasıyla MySqli ve PDO şeklinde aşağıda verilmiştir.

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
// Bağlantı yarat
$conn = new mysqli($servername, $username, $password, $dbname);
// Bağlantıyı kontrol et
if ($conn->connect_error) {
  die("Başarısız Bağlantı: " . $conn->connect_error);
}
// hazırlayın ve bağlayın
$stmt = $conn->prepare("INSERT INTO kullanicilar (ad, soyad, email) VALUES
(?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);
// parametreleri ayarlayın ve çalıştırın
$firstname = "ahmet";
$lastname = "ahmet";
$email = "ahmet@example.com";
$stmt->execute();

$firstname = "mehmet";
$lastname = "mehmet";
$email = "mehmet@example.com";
$stmt->execute();

$firstname = "hasan";
$lastname = "hasan";
$email = "hasan@example.com";
$stmt->execute();

echo "yeni kayıtlar başarılı bir şekilde eklendi.";
$stmt->close();
$conn->close();
?>
```

$stmt->bind_param("sss", $firstname, $lastname, $email); bu kod satırı parametleri SQL sorgusuna bağlar ve veritabanına parametrelerin ne olduğunu söyler.”sss” argümanı, parametrelerin olduğu veri türlerini listeler. S karakteri MySql’e parametrenin bir string olduğunu ifade eder. Argüman dört türden biri olabilir:

- i-integer
- d-double
- s-string
- b-BLOB

Aşağıdaki örnekte PDO'da hazır deyimler ve bağlı parametreler kullanılmaktadır:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // PDO hata modunu istisna olarak ayarlayın
  $stmt = $conn->prepare("INSERT INTO kullanicilar (ad, soyad, email)
  VALUES (:ad, :soyad, :email)");
  $stmt->bindParam(':ad', $firstname);
  $stmt->bindParam(':soyad', $lastname);
  $stmt->bindParam(':email', $email);

  // satır ekle
  $firstname = "mehmet";
  $lastname = "mehmet";
  $email = "mehmet@example.com";
  $stmt->execute();

  // satır ekle
  $firstname = "ahmet";
  $lastname = "ahmet";
  $email = "ahmet@example.com";
  $stmt->execute();

  // satır ekle
  $firstname = "hasan";
  $lastname = "hasan";
  $email = "hasan@example.com";
  $stmt->execute();

  echo "Yeni kayıtlar başarılı bir şekilde eklendi!!!!!!";
} catch(PDOException $e) {
  echo "Hata: " . $e->getMessage();
}
$conn = null;
?>
```

Veri Seçme (Select), Şart İfadesi (Where),Veri Sıralama (Order By)

"**SELECT**" ifadesi, bir veya daha fazla sütunu seçmek veya belirli bir koşulu karşılayan satırları seçmek için kullanılır. Temel kullanım;

SELECT sütun1,sütun2, …. FROM tablo_adı

şeklindedir. Bu ifade, tablodaki belirtilen sütunları seçer. Eğer tüm sütunları seçmek istiyorsak, "*" karakteri kullanılır. **SELECT** ifadesi ayrıca **WHERE** koşulu ile birlikte kullanılarak belirli bir koşulu karşılayan satırları seçmek için kullanılabilir. Örnek bir where sorgusu aşağıda verilmiştir.

SELECT sütun1,sütun2, …. FROM tablo_adı WHERE koşul ifadesi

Burada "**where**", seçilen satırları filtrelemek için kullanılır. Koşul, bir veya daha fazla sütunun belirli bir değere eşit olması, belirli bir aralıkta olması, bir örüntü ile eşleşmesi veya başka bir mantıksal ifade olabilir. Seçilen verileri belirli bir sütuna göre sıralamak için ise **ORDER BY** ifadesi kullanılır. Örneğin:

SELECT sütun1, sütun2, ... FROM tablo_adı ORDER BY sütun3 ASC;

Burada "sütun3" sütununa göre sıralama yapılır ve "**ASC**" parametresi küçükten büyüğe (artan) sıralama yapılmasını belirtir. "**DESC**" parametresi ise büyükten küçüğe (azalan) sıralama yapılmasını sağlar.

SELECT id, soyad, yaş FROM kullanicilar WHERE soyad ='yıldırım'ORDER BY yaş DESC örneğini inceleyelim. Bu örnekte, "**kullanicilar**" tablosundan "**ad**", "**soyad**" ve "**id**" sütunları seçilir. Ancak, sadece "yaş" sütunu 30'dan büyük olan kullanıcılar seçilir. Son olarak, "yaş" sütununa göre büyükten küçüğe sıralama yapılır. Bahsi geçen sorgu ile ilgili örnekler aşağıda verilmiştir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
// bağlantı oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if ($conn->connect_error) {
  die("başarısız bağlantı: " . $conn->connect_error);
}
$sql = "SELECT id, ad, soyad FROM kullanicilar WHERE soyad='yıldırım' ORDER
BY id DESC";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  // her satırın çıktı verisi
  while($row = $result->fetch_assoc()) {
    echo "id: " . $row["id"]. " - Ad: " . $row["ad"]. " " . $row["soyad"].
"<br>";
  }
} else {
  echo "0 results";
}
$conn->close();
?>
```

Yukarıda verilen örneği açıklayalım. Öncelikle kullanicilar tablosundan id, ad ve soyad sütunlarını seçen bir SQL sorgusu oluşturuyoruz. SQL sorgusu ilgili sütunları seçerken soyadı “yıldırım” olan kayıtları getiriyor ve id sütununa göre azalan sıralama yapmaktadır. Sonraki

kod satırı sorguyu çalıştırır ve elde edilen verileri **$result** adlı bir değişkene koyar. Daha sonra işlev, **num_rows()** ile döndürülen sıfırdan fazla satır olup olmadığını kontrol eder. Döndürülen sıfırdan fazla satır varsa, işlev **fetch_assoc()** tüm sonuçları döngü yapabileceğimiz ilişkisel bir diziye yerleştirir. Yanı sıra **fetch_both()** ve **fetch_num()** ile de sonuçlar döndürülebilir. fetch_both(), sorgudan dönen verileri hem bir ilişkisel dizi (assoc array) hem de bir sayısal dizi (numeric array) olarak alır. fetch_num() ise sorgudan dönen verileri bir sayısal dizi olarak alır. Her bir satır, sütunların sıra numaralarıyla ilişkilendirilir. Yani, sorgudan alınan her bir satır bir sayısal dizi olarak döner. **while()** dögüsü, sonuç kümesi boyunca döngü yapar ve id, ad ve soyadı sütunlarındaki verileri getirir.

Aşağıdaki örnek, MySQLi yordamsal yöntemiyle yukarıdaki örneğin aynısını göstermektedir:

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";
// bağlantı yarat
$conn = mysqli_connect($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if (!$conn) {
  die("başarısız bağlantı: " . mysqli_connect_error());
}
$sql = " SELECT id, ad, soyad FROM kullanicilar WHERE soyad='yıldırım' ORDER
BY id DESC ";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // her satırın çıktı verisi
  while($row = mysqli_fetch_assoc($result)) {
    echo "id: " . $row["id"]. " - Ad: " . $row["ad"]. " " . $row["soyad"].
"<br>";
  }
} else {
  echo "0 results";
}

mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// bağlantı oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if ($conn->connect_error) {
  die("Başarısız bağlantı: " . $conn->connect_error);
}

$sql = " SELECT id, ad, soyad FROM kullanicilar WHERE soyad='yıldırım' ORDER
BY id ASC ";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
  echo "<table><tr><th>ID</th><th>Name</th></tr>";
  // her satırın çıktı verisi
  while($row = $result->fetch_assoc()) {
    echo "<tr><td>".$row["id"]."</td><td>".$row["ad"]."
".$row["soyad"]."</td></tr>";
  }
  echo "</table>";
} else {
  echo "0 results";
}
$conn->close();
?>
```

Veri Silme (Delete)

Herhangi bir kaydı silmek için delete ifadesi kullanılmaktadır. Delete ifadesinin genel kullanımı:

DELETE from tablo_adı where koşul ifadesi

şeklindedir.

Dikkat Edelim: DELETE söz dizimindeki WHERE yan tümcesine dikkat edin: WHERE yan tümcesi, hangi kaydın veya kayıtların silinmesi gerektiğini belirtir. WHERE deyimini atlarsanız tüm kayıtlar silinecektir!

Delete ifadesi ile ilgili örnekler aşağıda verilmiştir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// bağlantı oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if ($conn->connect_error) {
  die("başarısız bağlantı: " . $conn->connect_error);
}
// sql to delete a record
$sql = "DELETE FROM kullanicilar WHERE id=3";

if ($conn->query($sql) === TRUE) {
  echo "Kayıt başarılı bir şekilde silindi";
} else {
  echo "kayıt silinemedi: " . $conn->error;
}
$conn->close();
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
  die("Bağlantı Hatası: " . mysqli_connect_error());
}
// sql to delete a record
$sql = "DELETE FROM kullanicilar WHERE id=3";

if (mysqli_query($conn, $sql)) {
  echo "kayıt başarılı bir şekilde silindi";
} else {
  echo "hata: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  // kayıt silmek için sql sorgusu
  $sql = "DELETE FROM kullanicilar WHERE id=3";

  // kayıt silmek için sql
  $conn->exec($sql);
  echo "Kayıt başarılı bir şekilde silindi";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}

$conn = null;
?>
```

Veri Güncelleme (Update)

Herhangi bir kaydı güncellemek için update ifadesi kullanılmaktadır. Update ifadesinin genel kullanımı:

UPDATE tablo_adı SET sütun1=değer, sütun2=değer … where koşul ifadesi şeklindedir.

Dikkat Edelim: UPDATE söz dizimindeki WHERE yan tümcesine dikkat edin: WHERE yan tümcesi, hangi kaydın veya kayıtların güncellenmesi gerektiğini belirtir. WHERE deyimini atlarsanız tüm kayıtlar güncelenecektir!

Update ifadesi ile ilgili örnekler aşağıda verilmiştir.

Örnek 1:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "mahmut";

// bağlantı oluştur
$conn = new mysqli($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if ($conn->connect_error) {
  die("başarısız bağlantı: " . $conn->connect_error);
}

$sql = "UPDATE kullanicilar SET soyad='kılıçaslan' WHERE id=2";

if ($conn->query($sql) === TRUE) {
  echo "kayıt başarılı bir şekilde güncellendi";
} else {
  echo "hata: " . $conn->error;
}

$conn->close();
?>
```

Örnek 2:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

// bağlantı oluştur
$conn = mysqli_connect($servername, $username, $password, $dbname);
// bağlantıyı kontrol et
if (!$conn) {
  die("başarısız bağlantı: " . mysqli_connect_error());
}

$sql = "UPDATE kullanicilar SET soyad='kılıçaslan' WHERE id=2";

if (mysqli_query($conn, $sql)) {
  echo "Kayıt başarılı bir şekilde güncellendi!";
} else {
  echo "hata: " . mysqli_error($conn);
}
mysqli_close($conn);
?>
```

Örnek 3:

```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "Ankuzef";

try {
  $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username,
$password);
  // PDO hata modunu istisna olarak ayarlayın
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $sql = "UPDATE kullanicilar SET soyad='kılıçaslan' WHERE id=2";
  //
  $stmt = $conn->prepare($sql);

  //
  $stmt->execute();
  //
  echo $stmt->rowCount() . " Kayıt başarılı bir şekilde güncellendi";
} catch(PDOException $e) {
  echo $sql . "<br>" . $e->getMessage();
}
$conn = null;
?>
```

Sınırlı Veri Seçme (Limit Data)

MySQL, döndürülecek kayıt sayısını belirtmek için kullanılan bir LIMIT cümleciği sağlar. LIMIT ifadesi, çok sayfalı sonuçları veya sayfalandırmayı SQL ile kodlamayı kolaylaştırır ve büyük tablolarda çok kullanışlıdır. Çok sayıda kaydın döndürülmesi performansı etkileyebilir.

"kullanicilar" adlı bir tablodan 1'den 30'a kadar (dahil) tüm kayıtları seçmek istediğimizi varsayalım. SQL sorgusu daha sonra şöyle görünecektir:

$sql = "SELECT * FROM kullanicilar LIMIT 30";
Yukarıdaki SQL sorgusu çalıştırıldığında ilk 30 kaydı döndürecektir. Ancak 16 - 25 (dahil) arasındaki kayıtları seçmek istersek Mysql'de bu durum için OFFSET kullanılmaktadır. Aşağıdaki SQL sorgusu 16. kayıttan başlayarak yalnızca 10 kayıt döndür.

$sql = "SELECT * FROM kullanicilar LIMIT 10 OFFSET 15";
Aynı sonucu elde etmek için daha kısa bir sözdizimi de kullanabilirsiniz:

$sql = "SELECT * FROM kullanicilar LIMIT 15, 10;

PHP MYSQL KULLANIMI İÇİN ÖRNEK BİR UYGULAMA

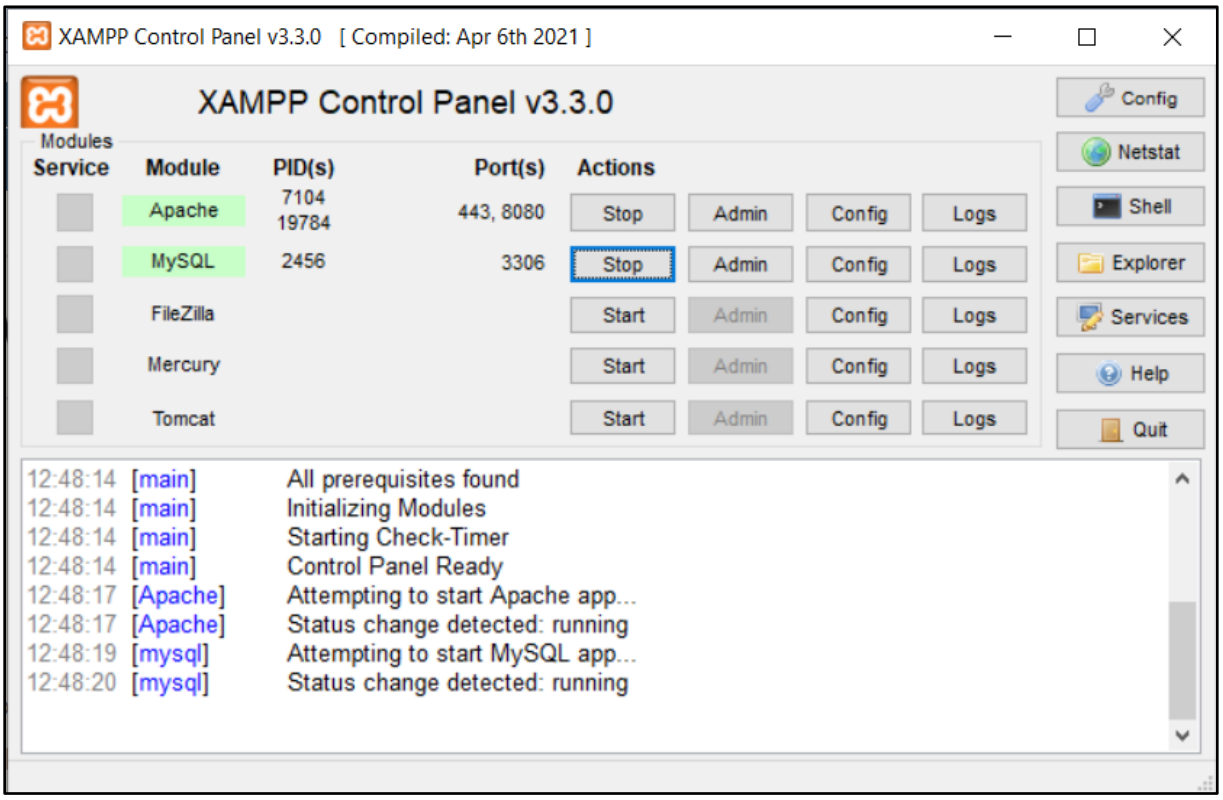
Bölümümüzün bu kısmında ekle/sil/güncelle ve veri çekme üzerine basit bir web uygulaması geliştireceğiz. Kayıt olarak da öğrenciye ait numara (ogrno), isim (ograd, ogrsoyad), kadın/erkek cinsiyet durumu (cinsiyet), doğum tarihi (dtarih), okuduğu sınıf (sinif) ve aldığı puan (puan) bilgilerini tutulmaktadır. Projemizin **bağlan.php, index.php, ekle.php, güncelle.php ve detay.php** olmak üzere beş sayfası bulunmaktadır. Proje için gerekli olan sql

ve php dosyaları sistem üzerinden indirilebilir. Aşağıda tablomuz için oluşturulan alanların isimleri, veri türleri, indeks bilgisi verilmektedir:

```
CREATE TABLE `ogrenci` (
  `ogrno` int(11) NOT NULL AUTO_INCREMENT,
  `ograd` varchar(30) NOT NULL,
  `ogrsoyad` varchar(30) NOT NULL,
  `cinsiyet` varchar(30) NOT NULL,
  `dtarih` date NOT NULL,
  `sinif` varchar(30) NOT NULL,
  `puan` int(11) NOT NULL
)
```

Görüldüğü üzere ogrno alanı indeks için kullanılmaktadır. Ayrıca otomatik artış değeri belirlendiği için bu alana veri girişi yapılmayacaktır. Her kayıt oluştuğunda ilgili yere 1'den başlayıp bir artarak öğrenci numarası verilmektedir.

Projenin çalıştırılması için sisteminize XAMPP paket programını yüklemeniz gerekmektedir (https://www.apachefriends.org/tr/). Program kurulduktan sonra ise XAMPP kontrol panelinden Apache ve MySQL modüllerinin başlatılması gerekir (Şekil 1).

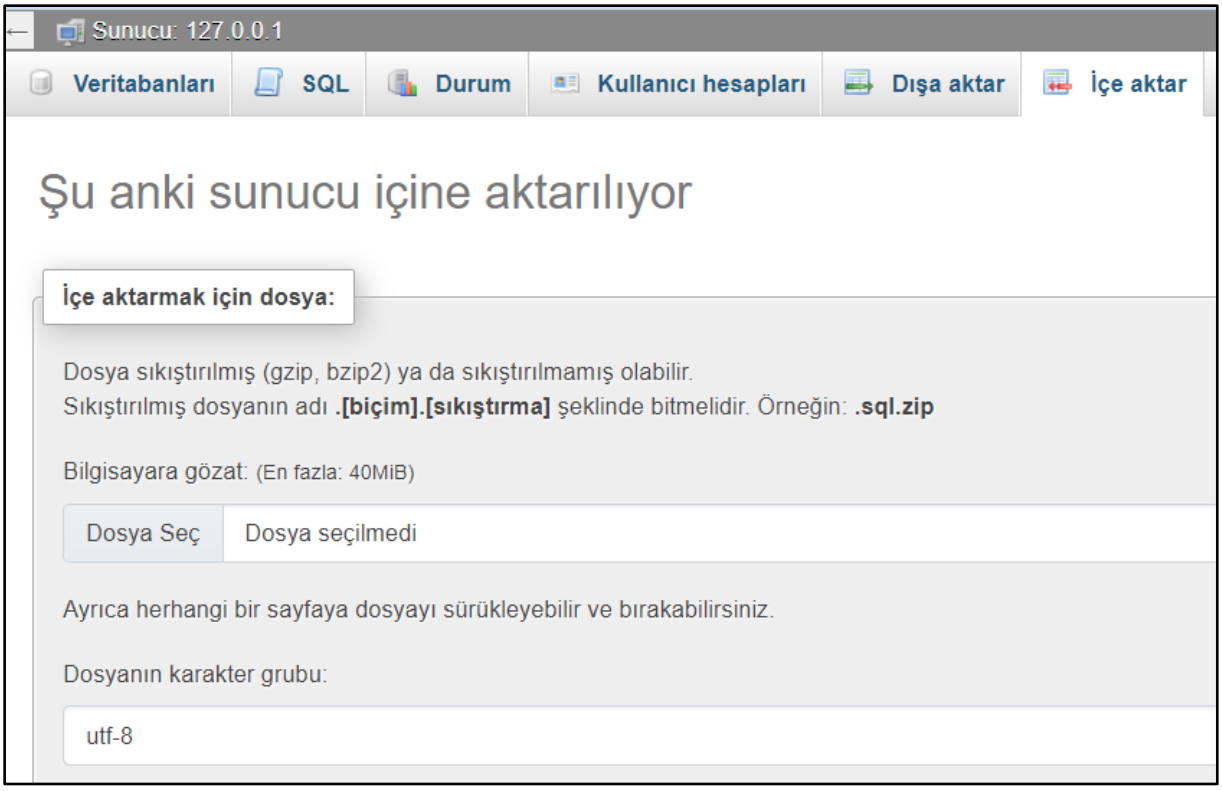


Şekil 1. XAMPP Kontrol Paneli

Bu işlemlerden sonra herhangi bir tarayıcı penceresinde localhost yazıp XAMPP'ın modüllerine bağlantı sağlayan pencereye (dashboard) geçilmelidir. Ardından menüden phpMyAdmin bağlantısı kullanılacaktır. Aşağıda phpMyAdmin'den kısmi bir ekran görüntüsü paylaşılmaktadır. İlgili yerde "İçe Aktar" ile ders sisteminizde yüklü bulunan "kutuphane.sql" dosyasını içe aktarmak gerekmektedir (Şekil 2). Proje için gerekli olan php dosyaları ise,

C:\xampp\htdocs

yolunda belirleyeceğiniz bir klasöre aktarılmalıdır. Burada oluşturulması gereken klasörü "proje" olarak kabul edelim.



Şekil 2. phpMyAdmin Ekran Görüntüsü

Projenizi başlatmak için tarayıcı penceresine "lolcahost/proje" yazmanız yeterlidir. Ana sayfanın (index.php) ekran görüntüsü aşağıdaki gibidir (Şekil 3):

Şekil 3. Ana Sayfa Ekran Görüntüsü

Ana sayfadaki Öğrenci Ekle düğmesi ile buraya daha önceden iki tane örnek kayıt getirilmiştir. Sizlerde bu kısım öğrenci eklenene kadar boş gözükecektir. Aşağıda öğrenci ekle işleminin (ekle.php) ekran görüntüsü verilmektedir (Şekil 4):

Şekil 4. Öğrenci Ekle İşlemi Ekran Görüntüsü

Ekle işlemi için bölümümüzün veri ekleme kısmında anlatılandan farklı olarak eklenecek veriler form öğesi içinde bulunan input öğelerinden alınmaktadır. Form öğesinden gönderilen veriler POST metodu ile aktarıldığından aşağıdaki gibi bir dizi değişken oluşturulmaktadır:

```
$dizi =[
        $_POST["ad"],
        $_POST["sad"],
        $_POST["cins"],
        $_POST["dtarih"],
        $_POST["sinif"]
    ];
```

Kayıt silme işlemi index.php dosyasında yer almaktadır:

```
<a href="?sil=<?=$satir['ogrno']?>" onclick="return confirm('Silinsin mi?')" class="btn btn-danger">Kaldır</a>
```

Burada silme işlemi sorgulu URL kodu ile a öğesinden gerçekleştirilmektedir. Kayıt silme (Şekil 5), listeleme (detay.php) (Şekil 6), güncelleme (guncelle.php) (Şekil 7) işlemlerinin ekran görüntüsü aşağıda sırasıyla verilmektedir:

Şekil 5. Öğrenci Silme İşlemi Ekran Görüntüsü

Şekil 6. Öğrenci Listeleme İşlemi Ekran Görüntüsü

Şekil 7. Öğrenci Güncelleme İşlemi Ekran Görüntüsü

Projedeki form öğelerinin daha etkili gösterilmesi için bootstrap.css dosyasından yararlanılmaktadır:

> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="anonymous">

İlgili satır form öğelerinin çağrıldığı sayfalarda <head></head> bloğu içerisinde yazılmaktadır. Bahsi geçen CSS dosyasındaki kodlar ise öğelerde class özniteliğinde değer atanarak kullanılmaktadır. Bağlantının oluşturulması/kapatılması ve tüm sorguların çalıştırılmasında PDO söz dizimi benimsenmektedir.

BÖLÜM ÖZETİ

My eskuÊL, ilişkisel veritabanı yönetim sistemleri arasında en popüler olanlardan biridir ve geniş kullanım alanlarına sahiptir. Veritabanı yönetimi, web uygulamaları, e-ticaret platformları, mobil uygulamalar ve veri analizi gibi çeşitli alanlarda yaygın olarak kullanılır. My eskuÊL, web sitelerinin ve uygulamalarının arkasında veri tabanı olarak kullanılarak, kullanıcı verilerinin saklanması, işlenmesi ve geri alınması için önemli bir rol oynar. PHP ise sunucu tarafı web geliştirme için özel olarak tasarlanmış güçlü bir programlama dilidir. Web sitelerinin dinamik ve etkileşimli özelliklerini geliştirmek için yaygın olarak kullanılır. Web tabanlı uygulamaların geliştirilmesi, veri tabanı işlemleri, form işlemleri, kullanıcı oturumları yönetimi ve sunucu tarafı programlama gibi birçok alanda PHP tercih edilir. Ayrıca, PHP'nin içerik yönetim sistemlerinde ve e-ticaret platformlarında kullanımı da oldukça yaygındır. My eskuÊL, verilerin saklanması, düzenlenmesi ve erişilmesi için kullanılır. eskuÊL dilini kullanarak veritabanı işlemleri gerçekleştirilir. Veri tabanı bağlantısı oluşturma, PHP gibi programlama dilleri kullanılarak My eskuÊL veritabanına bağlanmak için My eskuÊLi veya PDO gibi uzantılar kullanılır. Bağlantı bilgileri, (sunucu adı, kullanıcı adı, şifre vb.) belirtilerek bağlantı oluşturulur. Veri tabanında tablo oluşturma, veritabanında verilerin organize edildiği yapılardır. “CREATE TABLE” ifadesi kullanılarak yeni bir tablo oluşturulur. Tablonun adı ve sütunların adları ve veri türleri belirtilerek tablo oluşturulur. Veri tabanından istenilen verileri getirme, “SELECT” ifadesi kullanılarak belirli sütunlar veya tüm sütunlar seçilir. “WHERE” koşulu ile belirli bir koşulu karşılayan satırlar seçilebilir. Verileri sıralı bir şekilde getirme, “SELECT” ifadesinin sonuna “ORDER BY” ifadesi eklenerek veriler belirli bir sütuna göre sıralanabilir. “ASC” (artan) veya “DESC” (azalan) parametreleri kullanılarak sıralama yöntemi belirlenir. Veri ekleme, silme, güncelleme, “insert into” ifadesi kullanılarak yeni veri eklenir. “DELETE FROM” ifadesi kullanılarak belirli bir koşulu karşılayan veriler silinir. “UPDATE” ifadesi kullanılarak belirli bir koşulu karşılayan veriler güncellenir. Bu işlemler, veritabanındaki verilerin yönetilmesi ve güncellenmesi için kullanılır.

GÖZDEN GEÇİRELİM

1. PHP'de MySql söz dizim yöntemleri nelerdir?

2. Bağlantı cümlesinin conn isimli değişkende tutulduğu kabul edilsin. Bu durumda PDO söz diziminde ilgili bağlantı nasıl kapatılır?

3. bind_param() fonksiyonu aldığı parametreleri sql sorgusuna bağlar ve veritabanına parametrelerin ne olduğunu söyler. Söz konusu parametreler nelerdir?

4.

```
$stmt = $conn->prepare("INSERT INTO uye (ad, soyad, yas) VALUES (?, ?, ?)");
$stmt->bind_param("___", $firstname, $lastname, $email);
```

Yukarıdaki kodda insert deyimiyle hazırlanmış ifade oluşturulmuştur. Söz konusu uye tablosunun 3 adet string alanı bulunmaktadır. Bu durumda bind_param() fonksiyonunun alması gereken ilk parametre ne olmalıdır?

5. Bağlantı hatasını döndüren MySqli prosedürel söz dizimi nedir?

Cevaplar:

1. MySqli Nesne Yönelimli, MySqli Prosedürel, PDO

2. `$conn = null`

3. i-integer, d-double, s-string, b-BLOB

4. ssi

5.

```
$baglanti = mysqli_connect($servername, $username, $password, $dbname);
mysqli_error($baglanti)
```

KAYNAKÇA

[1] PHP ve MySQL, Pusula Yayıncılık, Erkan Balaban, İstanbul, 2010

[2] PHP ve AJAX, Seçkin Yayıncılık, Haydar Tuna, Ankara, 2010

FAYDALI KAYNAKLAR VE INTERNET KAYNAKLARI

https://www.w3schools.com/php/default.asp
https://www.php.net/manual/tr/book.mysql.php

BÖLÜM 4

PHP'DE XML VE AJAX

ANAHTAR KAVRAMLAR

XML, XML Ayrıştırıcı, AJAX

ÖĞRENME HEDEFLERİ

1. XML ve AJAX ile ilgili temel kavramları bilme
2. XML ayrıştırıcılarını kullanarak XML analiz edebilme
3. AJAX ile veritabanına bağlanabilme
4. AJAX ile canlı arama yapabilme
5. AJAX ile anket hazırlayabilme

PHP XML

PHP’de XML konusunu anlayabilmek için önce XML nediri bilmek gerekmektedir. XML dili, web siteleri arasında paylaşım için verileri yapılandırmanın bir yoludur. RSS Akışları ve Podcast'ler gibi çeşitli web teknolojileri XML'de yazılmıştır. XML'in oluşturulması kolaydır. Ayrıca, kendi etiketlerinizi oluşturmanız dışında HTML'ye çok benzemektedir. PHP’de XML kullanmak için XML ayrıştırıcılarına ihtiyaç vardır.

XML Ayrıştırıcılar

PHP’de bir XML belgesini okumak, güncellemek, oluşturmak ve değiştirmek için bir XML ayrıştırıcı kullanılmaktadır. PHP'de iki ana XML ayrıştırıcı türü vardır:

- Ağaç Tabanlı Ayrıştırıcılar
- Olay Tabanlı Ayrıştırıcılar

Ağaç Tabanlı Ayrıştırıcılar

Ağaç tabanlı ayrıştırıcılar belgenin tamamını bellekte tutar ve XML belgesini Ağaç yapısına dönüştürür. İlgili ayrıştırıcı, belgenin tamamını analiz eder ve oluşturulan Ağaç öğelerine (DOM) erişim sağlar.

Bu tür ayrıştırıcı, daha küçük XML belgeleri için iyi bir seçenektir. Büyük XML belgeleri için tercih edildiğinde ise performans sorunlarına neden olur.

Ağaç tabanlı ayrıştırıcılara örnek:

- SimpleXML
- DOM

Olay Tabanlı Ayrıştırıcılar

Olay tabanlı ayrıştırıcılar Ağaç tabanlının aksine belgenin tamamını bellekte tutmaz; bunun yerine, her seferinde bir düğümde okur ve gerçek zamanlı olarak etkileşim kurmanıza olanak tanır. Bir sonraki düğüme geçtiğinizde ise eskisi atılmaktadır.

Bu tür ayrıştırıcı büyük XML belgeleri için uygundur. Daha hızlı ayrıştırır ve daha az bellek tüketir.

Olay tabanlı ayrıştırıcılara örnek:

- XMLReader
- XML Expat Parser

SimpleXML Ayrıştırıcısı

SimpleXML, XML verilerini kolayca işlememize ve almamıza olanak tanıyan bir PHP uzantısıdır. SimpleXML yukarıda da bahsedildiği üzere ağaç tabanlı bir ayrıştırıcıdır.

SimpleXML, XML belgesinin yapısı veya düzeni biliniyorsa, bir öğenin adını, niteliklerini ve metin içeriğini almanın kolay bir yolunu sağlamaktadır.

SimpleXML, bir XML belgesini, diziler ve nesneler koleksiyonu gibi yineleyebileceğiniz bir veri yapısına dönüştürür.

DOM veya Expat ayrıştırıcıyla karşılaştırıldığında SimpleXML, bir öğeden metin verilerini okumak için daha az kod satırı kullanır.

SimpleXML Ayrıştırıcısı Kurulumu

PHP 5 sürümünden itibaren SimpleXML işlevleri PHP çekirdeğinin bir parçası olarak karşımıza çıkmaktadır. Bu bağlamda ayrıştırıcının işlevlerini kullanmak için herhangi bir kurulum gerekmez.

PHP SimpleXML - Dizeden Okuma

PHP simplexml_load_string() işlevi bir dizeden XML verilerini okumak için kullanılır.

Aşağıda verilen kod bloğunda olduğu gibi XML verilerini içeren bir değişkenimiz olduğunu varsayalım:

```
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Öğrenci 1</to>
<from>ANKUZEF</from>
<heading>Hatırlatıcı</heading>
<body>Bu hafta kayıt olmayı unutmayınız!</body>
</note>";
```

Aşağıdaki örnek kod bloğunda da, bir dizeden XML verilerini okumak için simplexml_load_string() işlevinin nasıl kullanılacağı gösterilmektedir:

```
<?php
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Öğrenci 1</to>
<from>ANKUZEF</from>
<heading>Hatırlatıcı</heading>
<body>Bu hafta kayıt olmayı unutmayınız!</body>
</note>";

$xml=simplexml_load_string($myXMLData) or die("Error: hatası ile nesne oluşturulamadı");
print_r($xml);
?>
```

Yukarıdaki kodun çıktısı şöyle olacaktır:

SimpleXMLElement Object ([to] => Öğrenci 1 [from] => ANKUZEF [heading] => Hatırlatıcı [body] => Bu hafta kayıt olmayı unutmayınız!)

Belgeyi yüklerken tüm XML hatalarını almak ve ardından hataları yinelemek için de libxml işlevi kullanılmaktadır. Aşağıdaki örnek, bozuk bir XML dizesini yüklemeye çalışır:

```php
<?php
libxml_use_internal_errors(true);
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<document>
  <user>Öğrenci 1</wronguser>
  <email>ogrenci1@ankara.edu.tr</wrongemail>
</document>";

$xml = simplexml_load_string($myXMLData);
if ($xml === false) {
    echo "XML yükleme başarısız: ";
    foreach(libxml_get_errors() as $error) {
        echo "<br>", $error->message;
    }
} else {
    print_r($xml);
}
?>
```

Yukarıdaki kodun çıktısı şöyle olacaktır:

XML yükleme başarısız:
Opening and ending tag mismatch: user line 3 and wronguser
Opening and ending tag mismatch: email line 4 and wrongemail

Görüldüğü üzere 3. ve 4. Satırdaki user ve email etiketleri yanlış kapatılmış ve ilgili hata libxml_get_errors() işlevi ile ekrana yazdırılmıştır.

PHP SimpleXML - Dosyadan Okuma

PHP simplexml_load_file() işlevi bir dosyadan XML verilerini okumak için kullanılır.

Aşağıda verildiği gibi "note.xml " adında bir XML dosyamız olduğunu varsayalım:

```xml
$myXMLData =
"<?xml version='1.0' encoding='UTF-8'?>
<note>
<to>Öğrenci 1</to>
<from>ANKUZEF</from>
<heading>Hatırlatıcı</heading>
<body>Bu hafta kayıt olmayı unutmayınız!</body>
</note>";
```

Aşağıdaki kod bloğunda ise, bir dosyadan XML verilerini okumak için simplexml_load_file() işlevinin nasıl kullanılacağı verilmektedir:

```php
<?php
$xml=simplexml_load_file("note.xml") or die("Error: hatası ile nesne oluşturulamadı");
print_r($xml);
?>
```

Yukarıdaki kodun çıktısı şöyle olacaktır:

> SimpleXMLElement Object ([to] => Öğrenci 1 [from] => ANKUZEF [heading] => Hatırlatıcı [body] => Bu hafta kayıt olmayı unutmayınız!)

PHP SimpleXML - Düğüm Değerlerini Alma

SimpleXML'in, XML verilerini kolayca işlememize ve almamıza olanak tanıyan bir PHP uzantısı olduğundan daha önce bahsetmiştik. İlgili ayrıştırıcı ile düğüm değerlerini almak için aşağıdaki gibi kod yazılmaktadır. Bu betik yukarıda daha önce bahsi geçen note.xml dosyasına erişmektedir:

```php
<?php
$xml=simplexml_load_file("note.xml") or die("Error: hatası ile nesne oluşturulamadı");
echo $xml->to . "<br>";
echo $xml->from . "<br>";
echo $xml->heading . "<br>";
echo $xml->body;
?>
```

Yukarıdaki kodun çıktısı şöyle olacaktır:

> Öğrenci 1
> ANKUZEF
> Hatırlatıcı
> Bu hafta kayıt olmayı unutmayınız!

Benzer şekilde bir de kitaplar için tutulan bir xml dosyamıza erişim sağlayalım. İlgili dosya aşağıdaki gibidir (books.xml):

```xml
<?xml version="1.0" encoding="utf-8"?>
<bookstore>
  <book category="Masaüstü">
    <title lang="en">Kitap 1</title>
    <author>Yazar 1</author>
    <year>2020</year>
    <price>130.00</price>
  </book>
  <book category="Mobil">
    <title lang="en">Kitap 2</title>
    <author>Yazar 2</author>
    <year>2022</year>
    <price>229.99</price>
  </book>
  <book category="WEB">
    <title lang="en-us">Kitap 3</title>
    <author>Yazar 3</author>
    <year>2024</year>
    <price>249.99</price>
  </book>
  <book category="WEB">
    <title lang="en-us">Kitap 4</title>
    <author>Yazar 4</author>
    <year>2018</year>
    <price>139.95</price>
  </book>
</bookstore>
```

PHP SimpleXML - Belirli Öğelerin Düğüm Değerlerini Alma

Aşağıdaki kod parçası ile "books.xml" dosyasının birinci ve ikinci <book> öğelerindeki <title> öğesinin düğüm değerini alınmaktadır:

```php
<?php
$xml=simplexml_load_file("books.xml") or die("Error: hatası ile nesne oluşturulmadı");
echo $xml->book[0]->title . "<br>";
echo $xml->book[1]->title;
?>
```

Yukarıdaki kodun çıktısı:

Kitap 1
Kitap 2

şeklindedir.

PHP SimpleXML - Düğüm Değerlerini Alma – Döngü Kullanımı

Aşağıdaki örnek, "books.xml" dosyasındaki tüm <book> öğeleri arasında döngü yapar ve <title>, <author>, <year> ve <price> öğelerinin düğüm değerlerini alır:

```php
<?php
$xml=simplexml_load_file("books.xml") or die("Error: hatası ile nesne oluşturulmadı");
foreach($xml->children() as $books) {
  echo $books->title . ", ";
  echo $books->author . ", ";
  echo $books->year . ", ";
  echo $books->price . "<br>";
}
?>
```

Yukarıdaki kodun çıktısı şöyle olacaktır:

Kitap 1, Yazar 1, 2020, 130.00
Kitap 2, Yazar 2, 2022, 229.99
Kitap 3, Yazar 3, 2024, 249.99
Kitap 4, Yazar 4, 2018, 139.95

PHP SimpleXML - Öznitelik Değerlerini Alma

SimpleXML ile ilk <book> öğesinin "category" niteliğinin öznitelik değerini ve ikinci <book> öğesindeki <title> öğesinin "lang" niteliğinin öznitelik değerini almak istersek kod bloğumuz aşağıdaki gibi olmalıdır:

```php
<?php
$xml=simplexml_load_file("books.xml") or die("Error: hatası ile nesne oluşturulmadı");
echo $xml->book[0]['category'] . "<br>";
echo $xml->book[1]->title['lang'];
?>
```

Yukarıdaki kod çalıştırıldığında,

Masaüstü
en

ekrana yazar.

PHP SimpleXML - Öznitelik Değerlerini Alma – Döngü

Aşağıdaki örnek, "books.xml" dosyasındaki <title> öğelerinin nitelik değerlerini bir döngü içerisinde almaktadır:

```php
<?php
$xml=simplexml_load_file("books.xml") or die("Error: hatası ile nesne oluşturulmadı");
foreach($xml->children() as $books) {
  echo $books->title['lang'];
  echo "<br>";
}
?>
```

Yukarıdaki kodun çıktısı:

en
en
en-us
en-us

şeklinde olacaktır.

PHP XML Expat Ayrıştırıcı

XML Expat Ayrıştırıcısı, SimpleXML de olduğu gibi XML belgelerinin PHP'de işlenmesini mümkün kılar. Expat ayrıştırıcısı olay tabanlı bir ayrıştırıcıdır.

Aşağıdaki XML öğesine bakın:

<from>ANKUZEF</from>

Olay tabanlı bir ayrıştırıcı, yukarıdaki XML'i üç olaydan oluşan bir dizi olarak bildirir:

Başlangıç öğesi: from
CDATA bölümünü başlat, değer: ANKUZEF
Kapanış öğesi: from

XML Expat Ayrıştırıcı işlevleri PHP çekirdeğinin bir parçasıdır. Bu işlevleri kullanmak için herhangi bir kuruluma gerek yoktur.

XML Expat Ayrıştırıcısını Başlatma

PHP'de XML Expat Ayrıştırıcısını başlatmak, farklı XML olayları için bazı işleyiciler tanımlamak ve ardından XML dosyasını ayrıştırmak istiyoruz. Bu duruma uygun kod parçacağı aşağıdaki gibi olacaktır:

```php
<?php
//XML ayrıştırıcıyı başlatma
$parser=xml_parser_create();
//Bir öğenin başlangıcında kullanılacak işlev
function start($parser,$element_name,$element_attrs) {
    switch($element_name) {
        case "NOTE": echo "-- Note --<br>";
    break;
        case "TO": echo "To: ";
    break;
        case "FROM": echo "From: ";
    break;
        case "HEADING": echo "Heading: ";
    break;
        case "BODY": echo "Message: ";
    }
}
//Bir öğenin sonunda kullanılacak işlev
function stop($parser,$element_name) { echo "<br>"; }
//Karakter verilerini bulurken kullanılacak işlev
function char($parser,$data) { echo $data; }
//Öğe işleyicisini belirtme
xml_set_element_handler($parser,"start","stop");
//Veri işleyiciyi belirtme
xml_set_character_data_handler($parser,"char");
//XML dosyasını açma
$fp=fopen("note.xml","r");
//Veriyi okuma
while ($data=fread($fp,4096)) {
    xml_parse($parser,$data,feof($fp)) or
    die (sprintf("XML Error: %s at line %d",
    xml_error_string(xml_get_error_code($parser)),
    xml_get_current_line_number($parser)));
}
//XML ayrıştırıcıyı serbesty bırakma
xml_parser_free($parser);
?>
```

Yukarıdaki kod bloğunun aşağıdaki gibi bir sıralaması olmaktadır:

1. XML ayrıştırıcısını xml_parser_create() işlev ile başlatın
2. Farklı olay işleyicileriyle kullanılacak işlevler oluşturun
3. Ayrıştırıcı açılış ve kapanış etiketleriyle karşılaştığında hangi işlevin yürütüleceğini belirtmek için xml_set_element_handler() işlevi ekleyin
4. Ayrıştırıcı karakter verileriyle karşılaştığında hangi işlevin yürütüleceğini belirtmek için xml_set_character_data_handler() işlevi ekleyin
5. "note.xml" dosyasını xml_parse() fonksiyonuyla ayrıştırın
6. Hata durumunda, XML hatasını metinsel açıklamaya dönüştürecek xml_error_string() işlevi ekleyin
7. xml_parser_create() işleviyle ayrılan belleği serbest bırakmak için xml_parser_free() işlevini çağırın

PHP XML DOM Ayrıştırıcı

DOM ayrıştırıcısı ağaç tabanlı bir ayrıştırıcıdır.

Aşağıdaki XML belge bölümüne bakın:

<?xml version="1.0" encoding="UTF-8"?>
<from>ANKUZEF</from>

DOM yukarıdaki XML'i bir ağaç yapısı olarak görür:

Seviye 1: XML Belgesi
Seviye 2: Kök öğe: <from>
Seviye 3: Metin öğesi: "ANKUZEF"

Kurulum

DOM ayrıştırıcı işlevleri PHP çekirdeğinin bir parçasıdır. Bu işlevleri kullanmak için herhangi bir kuruluma gerek yoktur.

XML'i Yükleme ve Çıkış

XML ayrıştırıcısını başlatmak, xml'i yüklemek ve çıktısını almak için,

```php
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

print $xmlDoc->saveXML();
?>
```

Yukarıdaki kodun çıktısı şöyle olacaktır:

Öğrenci 1 ANKUZEF Hatırlatıcı Bu hafta kayıt yapmayı unutmayınız!

Ayrıca yukarıdaki kod bloğunu herhangi tarayıcı penceresinde sağ tıklayıp “kaynağı görüntüle” seçildiğinde,

<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Öğrenci 1</to>
<from>ANKUZEF</from>
<heading>Hatırlatıcı</heading>
<body>Bu hafta kayıt yapmayı unutmayınız!</body>
</note>

şeklinde bir XML yapısı gibi gözüktüğü görülecektir. Bu kod, bir DOM Document-Object oluşturur ve XML'i "note.xml"den ona yükler.

Daha sonra saveXML() işlevi ile dâhili XML belgesini bir dizeye yerleştirir, böylece çıktısını alabiliriz.

XML'de Döngü Yapma

XML DOM ayrıştırıcısını başlatmak, XML'i yüklemek ve <note> öğesinin tüm öğeleri arasında döngü yapmak istediğimizde,

```php
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item) {
  print $item->nodeName . " = " . $item->nodeValue . "<br>";
}
?>
```

şeklinde kod yazmamız gerekmektedir. foreach ile DOM ayrıştırıcısı tabanlı öğelerin isimleri ve değerleri çekilmektedir. Yukarıdaki kodun çıktısı şöyle olacaktır:

#text =
to = Öğrenci 1
#text =
from = ANKUZEF
#text =
heading = Hatırlatıcı
#text =
body = Bu hafta kayıt olmayı unutmayınız!
#text =

Yukarıdaki örnekte her öğe arasında boş metin düğümleri olduğunu görüyorsunuz.

XML oluşturulurken genellikle düğümler arasında boşluklar bulunur. XML DOM ayrıştırıcısı bunları sıradan öğeler olarak ele alır. Ancak sizler bunların farkında değilseniz bazen sorunlara neden olmaktadırlar.

PHP - AJAX

AJAX, bir web sayfasının bazı bölümlerinin, tüm sayfayı yeniden yüklemeden güncellenmesiyle ilgilidir. Bu kısımda AJAX'ın ne olduğu ve PHP'deki kullanımını göreceğiz.

AJAX nedir?

AJAX = Eşzamansız JavaScript ve XML (Asynchronous JavaScript and XML).

AJAX hızlı ve dinamik web sayfaları oluşturmaya yönelik bir tekniktir.

AJAX, arka planda sunucuyla küçük miktarlarda veri alışverişi yaparak web sayfalarının eşzamansız olarak güncellenmesine olanak tanır. Bu, bir web sayfasının bazı kısımlarını, sayfanın tamamını yeniden yüklemeden güncellemenin mümkün olduğu anlamına gelir.

Klasik web sayfaları (AJAX kullanmayanlar), içeriğin değişmesi durumunda sayfanın tamamını yeniden yüklemelidir.

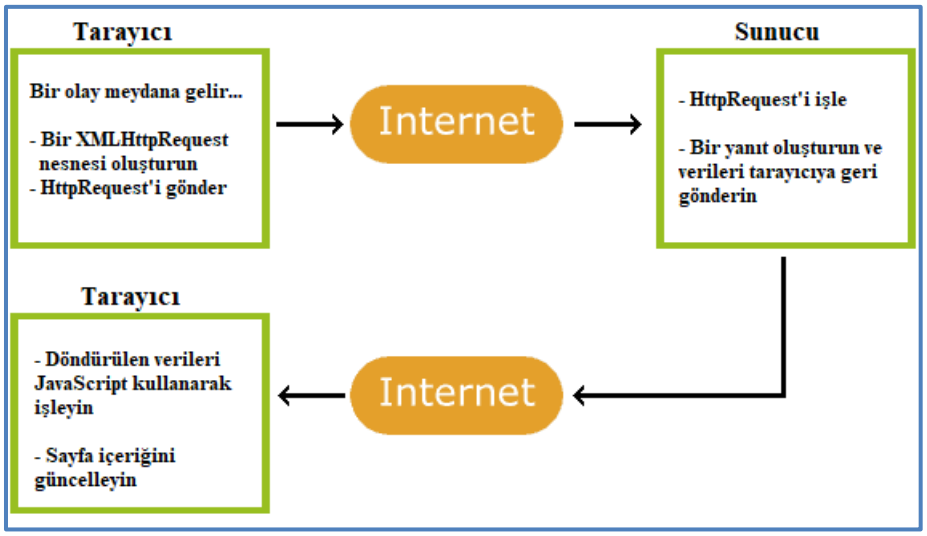
AJAX kullanan uygulamalara örnekler: Google Haritalar, Gmail, Youtube ve Facebook sekmeleri.

AJAX Nasıl Çalışır?

AJAX internet standartlarını temel alır ve aşağıdakilerin bir birleşim unu kullanır:

- XMLHttpRequest nesnesi (bir sunucuyla eşzamansız olarak veri alışverişi yapmak için)
- JavaScript/DOM (bilgileri görüntülemek/etkileşimde bulunmak için)
- CSS (verilere stil vermek için)
- XML (genellikle veri aktarma formatı olarak kullanılır)

AJAX uygulamaları tarayıcı ve platformdan bağımsızdır. Şekil 1 Sunucu ile İstemcinin tarayıcısı arasında gerçekleşen AJAX yapısını göstermektedir:



Şekil 1. AJAX Çalışma Prensibi

AJAX daha etkileşimli uygulamalar oluşturmak için kullanılır.

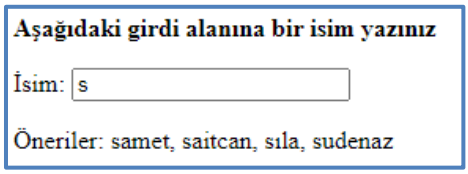
AJAX PHP

Aşağıdaki örnek, kullanıcı giriş alanına karakterleri yazarken (onkeyup) bir web sayfasının sunucuyla nasıl iletişim kurabileceğini göstermektedir. Bunun için arama işleminin yapıldığı index.php ve arama işlemi için kullanılan gethint.php dosyaları aşağıdaki gibidir:

index.php dosyası;

```
<html>
<head><meta charset="UTF-8">
<script>
function showHint(str) {
  if (str.length == 0) {
    document.getElementById("txtHint").innerHTML = "";
    return;
  } else {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("txtHint").innerHTML = this.responseText;
      }
    };
    xmlhttp.open("GET", "gethint.php?q=" + str, true);
    xmlhttp.send();
  }
}
</script>
</head>
<body>
<p><b>Aşağıdaki girdi alanına bir isim yazınız</b></p>
<form action="">
  <label for="fname">İsim:</label>
  <input type="text" id="fname" name="fname" onkeyup="showHint(this.value)">
</form>
<p>Öneriler: <span id="txtHint"></span></p>
</body>
</html>
```

şeklindedir. Çıktısı ise aşağıdaki gibidir:



Burada XMLHttpRequest, web sayfalarının sunucu ile iletişim kurmasını sağlayan bir API'dir.

this.readyState XMLHttpRequest nesnesinin durumunu belirtir ve 0 ile 4 arasında bir tamsayı değeri alır. Bu değerler şunları ifade eder:

0: İstek henüz başlatılmadı
1: Bağlantı kuruldu
2: İstek alındı
3: İstek işleniyor
4: İstek tamamlandı ve yanıt alındı

this.status ise sunucudan dönen HTTP yanıt kodunu belirtir. Genellikle, 200 bir "Başarılı" yanıt kodunu ve isteğin başarılı olduğunu gösterir.

Yani, this.readyState == 4 && this.status == 200 ifadesi, bir XMLHttpRequest nesnesinin durumunun 4 (tamamlanmış) ve sunucudan gelen yanıt kodunun 200 (başarılı) olduğu durumu kontrol eder. Bu durum genellikle bir AJAX isteğinin başarıyla tamamlandığını gösterir. Gelen istek gethint.php dosyasına q parametresi ile *gethint.php?q=" + str* şeklinde gönderilmektedir. str değişkeni giriş alanına yazılan değerleri tutmaktadır.

gethint.php dosyası;

```php
<?php
// İsimler ile bir dizi
$a[]="samet";
$a[]="saitcan";
$a[]="sıla";
$a[]="sudenaz";
// URL'den q parametresini alma
$q = $_REQUEST["q"];
$hint = "";
// $q "" öğesinden farklıysa dizideki tüm ipuçlarını arayın
if ($q !== "") {
  $q = strtolower($q);
  $len=strlen($q);
  foreach($a as $name) {
    if (stristr($q, substr($name, 0, $len))) {
      if ($hint === "") { $hint = $name; }
      else { $hint .= ", $name"; }
    }
  }
}
// Hiçbir ipucu bulunamazsa "öneri yok" çıktısı alın
echo $hint === "" ? "öneri yok" : $hint;
?>
```

AJAX Veritabanı

AJAX, bir veritabanıyla etkileşimli iletişim için kullanılabilir. Aşağıdaki örnek, bir web sayfasının AJAX ile bir veritabanından nasıl bilgi alabileceğini gösterecektir. Bunun için ajax_demo isimli veritabanı ve user isimli aşağıdaki gibi bir tabloya ihtiyaç vardır:

user tablosu;

id	ad	soyad	yas	sehir	meslek
1	isim1	soyad1	30	Ankara	Öğretmen
2	isim2	soyad2	20	İzmir	Mühendis
3	isim3	soyad3	25	İstanbul	Tekniker

Veritabanından Ajax ile verilere erişmek için index.php ve uye.php olmak üzere iki dosya daha oluşturulmalıdır.

index.php dosyası;

```html
<html>
<head>
<script>
function showUser(str) {
  if (str == "") {
    document.getElementById("txtHint").innerHTML = "";
    return;
  } else {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("txtHint").innerHTML = this.responseText;
      }
    };
    xmlhttp.open("GET","uye.php?q="+str,true);
    xmlhttp.send();
  }
}
</script>
</head>
<body>
<form>
<select name="users" onchange="showUser(this.value)">
  <option value="">Kişi seç:</option>
  <option value="1">isim1 soyisim1</option>
  <option value="2">isim2 soyisim2</option>
  <option value="3">isim3 soyisim3</option>
</select>
</form>
<br>
<div id="txtHint"><b>Kişinin bilgisi burada listelenmektedir:</b></div>
</body>
</html>
```

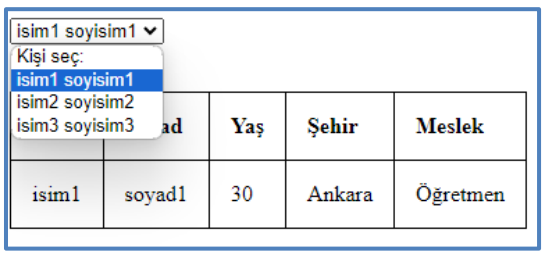
Yukarıdaki kod bloğunda, kullanıcı açılır listeden bir kişiyi seçtiğinde, çağrılan showUser() isimli bir işlev yürütülmektedir. İlgili işlev onchange olayı tarafından tetiklenmektedir.

onchange olayı, bir HTML form elemanının değeri değiştiğinde tetiklenen bir JavaScript olayıdır. Bu olay genellikle <input>, <select>, ve <textarea> gibi form elemanlarında kullanılır.

Örneğin, bir <select> elemanında seçilen öğe değiştirildiğinde veya bir <input type="checkbox"> veya <input type="radio"> elemanındaki seçim değiştirildiğinde onchange olayı devreye girmektedir.

JavaScript'te bu olaya bir işlev atanarak, kullanıcı bir değer seçtiğinde veya girdiğinde yapılması gereken işlemleri tanımlayabilirsiniz. Örneğin, bir seçim yapıldığında bir fonksiyon çağrılabilir veya belirli bir kontrolün durumu değiştiğinde bir işlem gerçekleştirilebilir.

index.php dosyasının çıktısı aşağıdaki gibidir:



Burada öncelikle bir kişinin seçilip seçilmediği kontrol edilmektedir. Hiç kimse seçilmediyse (str == "") txtHint id'li div öğesi temizlenmektedir (document.getElementById("txtHint").innerHTML = "";). Eğer bir kişi seçilirse XMLHttpRequest nesnesi oluşturulmakta ve seçim URL'ye q parametresi ile gönderilmektedir.

uye.php dosyası ise aşağıdaki gibi olup seçilen kişi bir tablo içerisinde listelenmektedir.

```
<html>
<head>
    <style>table,tr,th,td{
        border:1px solid black;
        border-collapse:collapse;
        text-align:left;
        padding:15px;}</style>
</head>
<body>
<?php
$q = intval($_GET['q']);
$con = mysqli_connect('localhost','root','');
if (!$con) {
  die('Bağlantı kurulmadı: ' . mysqli_error($con));
}
mysqli_select_db($con,"ajax_demo");
$sql="SELECT * FROM user WHERE id = '".$q."'";
$result = mysqli_query($con,$sql);
echo "<table>
<tr>
<th>Ad</th><th>Soyad</th><th>Yaş</th><th>Şehir</th><th>Meslek</th>
</tr>";
while($row = mysqli_fetch_array($result)) {
  echo "<tr>";
  echo "<td>" . $row['ad'] . "</td>";
  echo "<td>" . $row['soyad'] . "</td>";
  echo "<td>" . $row['yas'] . "</td>";
  echo "<td>" . $row['sehir'] . "</td>";
  echo "<td>" . $row['meslek'] . "</td>";
  echo "</tr>";
}
echo "</table>";
mysqli_close($con);
?>
</body></html>
```

AJAX XML

AJAX, bir XML dosyasıyla etkileşimli iletişim için kullanılabilir. Aşağıdaki örnek, bir web sayfasının AJAX ile bir XML dosyasından nasıl bilgi getirebileceğini gösterecektir. Bu bağlamda uygulama için cd_catalog.xml isimli sanatçı albümlerinin tutulduğu bir dosya ve XML analizini yapabilmek adına index.php, getcd.php isimli dosyalar oluşturulmaktadır.

cd_catalog.xml dosyası,

```xml
<?xml version="1.0" encoding="utf-8"?>
<KATALOG>
  <CD>
    <ALBUM>Başlık 1</ALBUM>
    <ARTIST>Sanatçı 1</ARTIST>
    <ULKE>Ülke 1</ULKE>
    <SIRKET>Şirket 1</SIRKET>
    <FIYAT>Fiyat 1</FIYAT>
    <YIL>Yıl 1</YIL>
  </CD>
  <CD>
    <ALBUM>Başlık 2</ALBUM>
    <ARTIST>Sanatçı 2</ARTIST>
    <ULKE>Ülke 2</ULKE>
    <SIRKET>Şirket 2</SIRKET>
    <FIYAT>Fiyat 2</FIYAT>
    <YIL>Yıl 2</YIL>
  </CD>
    <CD>
    <ALBUM>Başlık 3</ALBUM>
    <ARTIST>Sanatçı 3</ARTIST>
    <ULKE>Ülke 3</ULKE>
    <SIRKET>Şirket 3</SIRKET>
    <FIYAT>Fiyat 3</FIYAT>
    <YIL>Yıl 3</YIL>
  </CD>
</KATALOG>
```

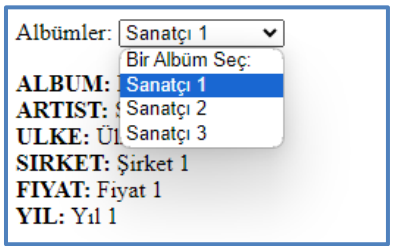
index.php dosyası,

```php
<html>
<head>
<script>
function showCD(str) {
  if (str=="") {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  var xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange=function() {
    if (this.readyState==4 && this.status==200) {
      document.getElementById("txtHint").innerHTML=this.responseText;
    }
  }
  xmlhttp.open("GET","getcd.php?q="+str,true);
  xmlhttp.send();
}
</script>
</head>
<body>
<form>
Albümler:
<select name="cds" onchange="showCD(this.value)">
  <option value="">Bir Albüm Seç:</option>
  <option value="Sanatçı 1">Sanatçı 1</option>
  <option value="Sanatçı 2">Sanatçı 2</option>
  <option value="Sanatçı 3">Sanatçı 3</option>
</select>
</form>
<div id="txtHint"><b>Albüm bilgisi burada listelenecektir...</b></div>
</body>
</html>
```

getcd.php dosyası,

```php
<?php
$q=$_GET["q"];
$xmlDoc = new DOMDocument();
$xmlDoc->load("cd_catalog.xml");
$x=$xmlDoc->getElementsByTagName('ARTIST');
for ($i=0; $i<=$x->length-1; $i++) {
  //Yalnızca öğe düğümlerini işle
  if ($x->item($i)->nodeType==1) {
    if ($x->item($i)->childNodes->item(0)->nodeValue == $q) {
      $y=($x->item($i)->parentNode);
    }
  }
}
$cd=($y->childNodes);
for ($i=0;$i<$cd->length;$i++) {
  //Yalnızca öğe düğümlerini işle
  if ($cd->item($i)->nodeType==1) {
    echo("<b>" . $cd->item($i)->nodeName . ":</b> ");
    echo($cd->item($i)->childNodes->item(0)->nodeValue);
    echo("<br>");
  }
}
?>
```

şeklindedir. Uygulamanın çıktısı ise,



Kullanıcı yukarıdaki açılır listeden bir CD seçtiğinde "showCD()" adı verilen bir işlev yürütülür. İşlev "onchange" olayıyla tetiklenmektedir. showCD() işlevi aşağıdakileri yapar:

- Bir CD'nin seçilip seçilmediğini kontrol eder,
- XMLHttpRequest nesnesi oluşturur,
- Sunucu yanıtı hazır olduğunda yürütülecek işlevi belirler,
- İsteği sunucudaki bir dosyaya gönderir,
- URL'ye (getcd.php) açılır listeden seçilen değeri q isimli bir parametre ile gönderir.

getcd.php içindeki betik ile "cd_catalog.xml" adlı bir XML belgesi yüklenmekte ve XML dosyasında bir sorgu çalıştırılmaktadır. Sorgunun sonucu HTML olarak dönmektedir.

Albüm sorgusu JavaScript'ten PHP sayfasına gönderildiğinde aşağıdakiler gerçekleşir:

- PHP bir XML DOM nesnesi oluşturur,
- JavaScript'ten gönderilen <artist> etiketi ile eşleşen tüm öğeleri bulur,

- Albüm bilgilerinin çıktısı alınır. Yani "txtHint" id'li div öğesine gönderilir.

AJAX Canlı Arama

AJAX, daha kullanıcı dostu ve etkileşimli aramalar oluşturmak için kullanılabilir. Aşağıdaki örnek, yazarken arama sonuçlarını alacağınız canlı bir aramayı göstermektedir.

Canlı aramanın geleneksel aramaya kıyasla birçok avantajı vardır:

- Siz yazdıkça sonuçlar gösterilir
- Siz yazmaya devam ettikçe sonuçlar daralır
- Sonuçlar çok dar olursa daha geniş bir sonuç görmek için karakterleri silmek gerekir.

İlgili uygulama için, links.xml isimli dosyada ANKUZEF'e bağlı Açıköğretim Lisans Programlarının isimleri ve url'leri tanımlanmaktadır. İlaveten index.php ve livesearch.php isimli dosyalara ihtiyaç duyulmaktadır.

links.xml dosyası,

```
<?xml version="1.0" encoding="utf-8"?>
<sayfalar>
  <baglanti>
    <baslik>Elektronik Ticaret ve Yönetimi Açıköğretim Lisans Programı</baslik>
    <url>https://ankuzef.ankara.edu.tr/programlar/acikogretim-programlari/
    lisans-programlari/47-ankuzef/programlar/acikogretim/lisans-programlari/
    310-elektronik-ticaret-ve-yoenetimi-ac-koegretim-lisans-program</url>
  </baglanti>
  <baglanti>
    <baslik>Yönetim Bilişim Sistemleri Açıköğretim Lisans Programı</baslik>
    <url>https://ankuzef.ankara.edu.tr/programlar/acikogretim-programlari/
    lisans-programlari/47-ankuzef/programlar/acikogretim/lisans-programlari/
    311-yoenetim-bilisim-sistemleri-ac-koegretim-lisans-program</url>
  </baglanti>
</sayfalar>
```

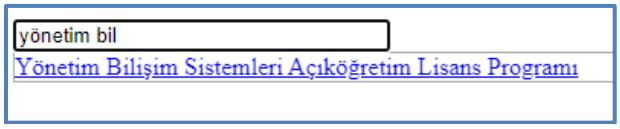
index.php dosyası,

```
<html>
<head>
<script>
function showResult(str) {
  if (str.length==0) {
    document.getElementById("livesearch").innerHTML="";
    document.getElementById("livesearch").style.border="0px";
    return;
  }
  var xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange=function() {
    if (this.readyState==4 && this.status==200) {
      document.getElementById("livesearch").innerHTML=this.responseText;
      document.getElementById("livesearch").style.border="1px solid #A5ACB2";
    }
  }
  xmlhttp.open("GET","livesearch.php?q="+str,true);
  xmlhttp.send();
}
</script>
</head>
<body>
<form>
<input type="text" size="30" onkeyup="showResult(this.value)">
<div id="livesearch"></div>
</form>
</body>
</html>
```

livesearch.php dosyası,

```php
<?php
$xmlDoc=new DOMDocument();
$xmlDoc->load("links.xml");
$x=$xmlDoc->getElementsByTagName('baglanti');
// URL'den q parametresini al
$q=$_GET["q"];
// uzunluğu q>0 ise xml dosyasındaki tüm bağlantıları arayın
if (strlen($q)>0) {
  $hint="";
  for($i=0; $i<($x->length); $i++) {
    $y=$x->item($i)->getElementsByTagName('baslik');
    $z=$x->item($i)->getElementsByTagName('url');
    if ($y->item(0)->nodeType==1) {
      // arama metniyle eşleşen bir bağlantı bulun
      if (stristr($y->item(0)->childNodes->item(0)->nodeValue,$q)) {
        if ($hint=="") {
          $hint="<a href='" .
          $z->item(0)->childNodes->item(0)->nodeValue .
          "' target='_blank'>" .
          $y->item(0)->childNodes->item(0)->nodeValue . "</a>";
        } else {
          $hint=$hint . "<br /><a href='" .
          $z->item(0)->childNodes->item(0)->nodeValue .
          "' target='_blank'>" .
          $y->item(0)->childNodes->item(0)->nodeValue . "</a>";
        }
      }
    }
  }
}
// İpucu bulunamazsa çıktıyı "öneri yok" olarak ayarlayın
// veya doğru değerlere
if ($hint=="") { $response="öneri yok"; }
else { $response=$hint; }
// yanıtı göster
echo $response;
?>
```

Uygulamanın ara yüzü ise aşağıdaki gibidir:



Yukarıdaki giriş alanında bir ANKUZEF Açıköğretim lisans programı sayfası aranmaktadır. Gerekli sonuçlar bir XML dosyasında (links.xml) bulunmaktadır. Bu örneği küçük ve basit hale getirmek için yalnızca iki sonuç mevcuttur. Daha fazla sonuç için xml dosyasına daha fazla kayıt girilmesi gerekir.

index.php dosyasında tanımlı <input> öğesine bir karakter yazdığında "showResult()" isimli işlev yürütülür. İşlev "onkeyup" olayıyla tetiklenir. Giriş alanı boşsa (str.length==0), işlev canlı arama yer tutucusunun içeriğini temizler ve işlevden çıkar.

Giriş alanı boş değilse showResult() işlevi aşağıdakileri yürütür:

- XMLHttpRequest nesnesi oluşturur,
- Sunucu yanıtı hazır olduğunda yürütülecek işlevi oluşturur,
- İsteği sunucudaki bir dosyaya gönderir,
- URL'ye (giriş alanının içeriğiyle birlikte) bir parametre (q) ekler.

"livesearch.php" dosyasındaki kaynak kod ise arama dizesiyle eşleşen başlıkları bir XML dosyasında arar ve sonucu döndürür. JavaScript'ten gönderilen herhangi bir metin varsa (strlen($q) > 0), aşağıdakiler gerçekleşir:

- XML dosyasını yeni bir XML DOM nesnesine yükler,
- JavaScript'ten gönderilen metindeki eşleşmeleri bulmak için tüm <baslik> öğeleri arasında dolaşır,
- "$response" değişkeninde doğru URL'yi ve başlığı ayarlar. Birden fazla eşleşme bulunursa tüm eşleşmeler değişkene eklenir,
- Hiçbir eşleşme bulunamazsa $response değişkeni "öneri yok" olarak ayarlanır.

AJAX Anket

Buradaki örnek, sonucun yeniden yüklenmeden gösterildiği bir anket uygulamasıdır. Kullanıcı seçeneklerden birini seçtiğinde "getVote()" adı verilen bir işlev yürütülür. İşlev "onclick" olayı tarafından tetiklenmektedir. Uygulama için index.php ve poll_vote.php isimli iki temel dosya kullanılmaktadır. Anket sonuçları ise php kodları çalıştırıldığında poll_result.txt dosyasına kaydedilmektedir. İlaveten anket sonuçlarına görsellik katmak adına poll.gif isimli bir görüntü dosyasına ihtiyaç vardır.

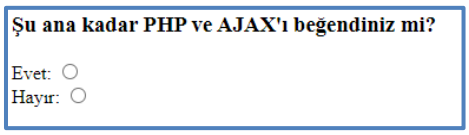
index.php dosyası,

```
<html>
<head>
<script>
function getVote(int) {
  var xmlhttp=new XMLHttpRequest();
  xmlhttp.onreadystatechange=function() {
    if (this.readyState==4 && this.status==200) {
      document.getElementById("poll").innerHTML=this.responseText;
    }
  }
  xmlhttp.open("GET","poll_vote.php?vote="+int,true);
  xmlhttp.send();
}
</script>
</head>
<body>
<div id="poll">
<h3>Şu ana kadar PHP ve AJAX'ı beğendiniz mi?</h3>
<form>
Evet: <input type="radio" name="vote" value="0" onclick="getVote(this.value)"><br>
Hayır: <input type="radio" name="vote" value="1" onclick="getVote(this.value)">
</form>
</div>
</body>
</html>
```

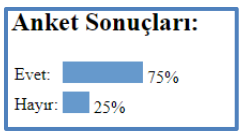
poll_vote.php dosyası,

```php
<?php
$vote = $_REQUEST['vote'];
// metin dosyasının içeriğini al
$filename = "poll_result.txt";
$content = file($filename);
// içeriği diziye koy
$array = explode("||", $content[0]);
$yes = $array[0];
$no = $array[1];
if ($vote == 0) { $yes = (int)$yes + 1; }
if ($vote == 1) { $no = (int)$no + 1; }
// txt dosyasına oyları ekleme
$insertvote = $yes."||".$no;
$fp = fopen($filename,"w");
fputs($fp,$insertvote);
fclose($fp);
?>
<h2>Anket Sonuçları:</h2>
<table>
<tr>
<td>Evet:</td>
<td><img src="poll.gif"
width='<?php echo(100*round((int)$yes/((int)$no+(int)$yes),2)); ?>'
height='20'>
<?php echo(100*round((int)$yes/((int)$no+(int)$yes),2)); ?>%
</td>
</tr>
<tr>
<td>Hayır:</td>
<td><img src="poll.gif"
width='<?php echo(100*round((int)$no/((int)$no+(int)$yes),2)); ?>'
height='20'>
<?php echo(100*round((int)$no/((int)$no+(int)$yes),2)); ?>%
</td>
</tr>
</table>
```

Uygulama çalıştırıldığında,



ekrana oylama anketi gelir. Seçeneklerden hangisinin işaretlendiğine göre, evet'lerin sayısı toplam seçim sayısına ve hayır'ların sayısı toplam seçim sayısına bölünerek yüzde değerler hesaplanmaktadır.



getVote() işlevi aşağıdakileri yapar:

- XMLHttpRequest nesnesi oluşturur,
- Sunucu yanıtı hazır olduğunda yürütülecek işlevi oluşturur,
- İsteği sunucudaki bir dosyaya gönderir,
- URL'ye (poll_vote.php) evet veya hayır seçeneğinin değeriyle birlikte bir parametre (vote) gönderir.

BÖLÜM ÖZETİ

Bu bölümde X emel ve ajax'ın PHP ile kullanımı üzerine bilgiler verilmektedir. X emel, verilerin yapısal olarak organize edildiği bir formattır ve PHP'de bu verileri işlemek için bir dizi X emel ayrıştırıcı bulunur. Bu ayrıştırıcılar sayesinde X emel belgelerindeki verileri kolayca okuyabilir, değiştirebilir ve işleyebiliriz. X emel belgesini yükleyerek, X Path veya dom gibi yöntemlerle belirli öğeleri seçebilir, veri çekebilir ve manipüle edebiliriz. Simple X emel, dom Dokument ve X emel Reader gibi PHP'nin yerleşik sınıfları, X emel belgeleriyle etkileşim sağlamak için sıkça kullanılan araçlardır. ajax ise Asenkron Java Script ve X emel'in kısaltmasıdır ve web uygulamalarında kullanıcı etkileşimini geliştirmek için kullanılan bir tekniktir. Sayfayı yenilemeden sunucu ile veri alışverişi yapmayı sağlar, böylece kullanıcılar daha hızlı ve daha akıcı bir deneyim yaşarlar. ajax, tarayıcı tarafında Java Script kullanarak sunucuya http istekleri yapar ve gelen yanıtları işler. Ji SON veya X emel formatında veri alışverişi yapabilir ve bu verileri dinamik olarak sayfaya ekleyebiliriz. ajax, kullanıcı dostu ve etkili web uygulamaları oluşturmanın temel taşlarından biridir ve modern web geliştirme süreçlerinde yaygın olarak kullanılmaktadır.

GÖZDEN GEÇİRELİM

1. PHP'de XML dosyasından veri çekmek için hangi yöntemler kullanılabilir?

2. AJAX nedir ve nasıl kullanılır?

3. SimpleXML nedir ve ne işe yarar?

4. DOMDocument ile XML ayrıştırma nasıl yapılır?

5. AJAX ile XML verisi nasıl alınır ve işlenir?

Cevaplar:

1. PHP'de XML dosyasından veri çekmek için SimpleXML, DOMDocument, ve XMLReader gibi yerleşik XML ayrıştırıcıları kullanılabilir. Bu sınıflar, XML belgelerini yüklemek, belirli öğeleri seçmek, veri çekmek ve manipüle etmek için kullanılır.

2. AJAX, Asenkron JavaScript ve XML'in kısaltmasıdır ve web uygulamalarında kullanıcı etkileşimini geliştirmek için kullanılan bir tekniktir. AJAX kullanılarak, tarayıcı tarafında JavaScript kullanılarak sunucuya HTTP istekleri yapılır ve gelen yanıtlar işlenir. Bu şekilde, sayfa yenilenmeden sunucu ile veri alışverişi yapılabilir, böylece daha hızlı ve daha akıcı bir kullanıcı deneyimi sağlanır.

3. SimpleXML, PHP'de XML belgelerini işlemek için kullanılan basit ve kullanıcı dostu bir ayrıştırıcıdır. XML belgelerini yüklemek, belirli öğeleri seçmek ve veri çekmek için kullanılır.

4. DOMDocument, PHP'de XML belgelerini yüklemek ve işlemek için kullanılan bir ayrıştırıcıdır. XML belgesini yükledikten sonra, belirli öğeleri seçmek için XPath veya DOM yöntemlerini kullanabilirsiniz. Bu sayede, belgedeki verileri çekebilir ve manipüle edebilirsiniz.

5. AJAX kullanılarak, tarayıcı tarafında JavaScript ile sunucuya HTTP istekleri yapılır ve gelen yanıtlar işlenir. XML verisi almak için genellikle XMLHttpRequest objesi kullanılır. Gelen XML verisi JavaScript'te işlenerek, istenen verilere erişilir ve kullanıcı arayüzünde gösterilir veya manipüle edilir. Bu sayede, sayfa yenilenmeden dinamik içerikler görüntülenebilir.

KAYNAKÇA

Anbiah, R. R. J., Bhattarai, R., & Sedliak, M. (2011). *PHP ajax cookbook*. Packt Pub.

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

Vaswani, V. (2002). *XML and PHP*. Sams Pearson Education distributor.

FAYDALI KAYNAKLAR VE INTERNET KAYNAKLARI

PHP - ajax and mysql. PHP AJAX and MySQL. (n.d.). https://www.w3schools.com/php/php_ajax_database.asp

PHP - Ajax and php. (n.d.). https://www.w3schools.com/php/php_ajax_php.asp

PHP - Ajax introduction. (n.d.). https://www.w3schools.com/php/php_ajax_intro.asp

PHP example - ajax and XML. PHP AJAX and XML. (n.d.). https://www.w3schools.com/php/php_ajax_xml.asp

PHP example - ajax live search. PHP AJAX Live Search. (n.d.). https://www.w3schools.com/php/php_ajax_livesearch.asp

PHP example - ajax poll. PHP AJAX Poll. (n.d.). https://www.w3schools.com/php/php_ajax_poll.asp

PHP SimpleXML - get node/attribute values. (n.d.). https://www.w3schools.com/php/php_xml_simplexml_get.asp

PHP simplexml parser. (n.d.). https://www.w3schools.com/php/php_xml_simplexml_read.asp

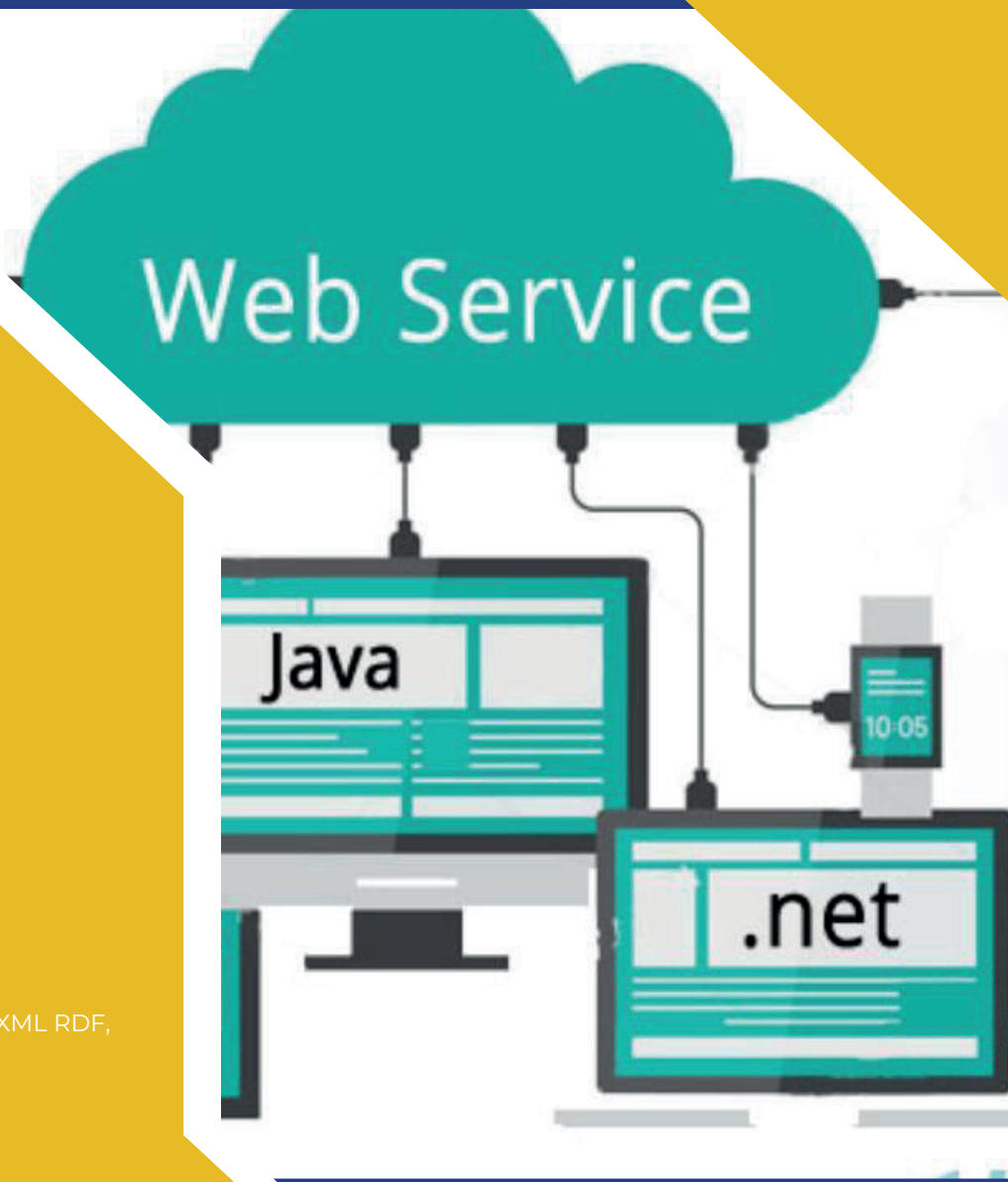
PHP XML Dom Parser. (n.d.). https://www.w3schools.com/php/php_xml_dom.asp

PHP XML expat parser. (n.d.). https://www.w3schools.com/php/php_xml_parser_expat.asp

PHP XML parsers. (n.d.). https://www.w3schools.com/php/php_xml_parsers.asp

BÖLÜM 5

WEB SERVİSLER



ANAHTAR KAVRAMLAR

XML Web Servisleri, WSDL, SOAP, XML RDF, RSS

ÖĞRENME HEDEFLERİ

1. XML Web Servislerinin temel kavramlarını ve çalışma prensiplerini anlamak
2. XML Web Servislerinin avantajlarını ve dezavantajlarını değerlendirebilmek
3. SOAP protokülünün temel bileşenlerini ve çalışma prensiplerini anlamak
4. RDF'nin temel kavramlarını ve çalışma prensiplerini anlamak
5. RSS'in temel kavramlarını ve çalışma prensiplerini anlamak
6. XML Web Servislerini RSS kullanarak yayınlayabilmek

GİRİŞ

Günümüzün dijital dünyasında, birbirleriyle iletişim kuran ve veri alışverişi yapan çok sayıda uygulama bulunmaktadır. Farklı platformlarda ve programlama dillerinde yazılmış bu [illegible] alışverişini kolaylaştırmak ve standart hale getirmek için **XML Web Servisleri** devreye girer.

XML Web Servisleri, tıpkı internette bir web sitesine erişmek gibi, farklı uygulamaların birbirleriyle iletişim kurmasına ve veri alışverişi yapmasına olanak tanıyan bir teknolojidir. Bu sayede, farklı sistemler arasında veri aktarımı manuel olarak yapılmadan otomatik hale gelir ve hatalar en aza indirilir.

Bu bölümde, XML Web Servislerinin temel kavramlarını, çalışma prensiplerini ve kullanım alanlarını adım adım inceleyeceğiz. Ayrıca, WSDL, SOAP, RDF ve RSS gibi XML Web Servisleri ile ilgili önemli teknolojileri tanımaya çalışacağız.

Hayatın içinden bir örnek vererek başlayalım: Bir kitap satıcısı olduğunuzu hayal edin: Müşterilerinizin internet üzerinden stoklarınızdaki tüm kitapları kolayca sorgulayıp, diledikleri gibi sipariş verebilmelerini ister misiniz? Üstelik bu müşteriler farklı platformlarda ve farklı dillerde yazılmış uygulamalar kullanıyor olsalar bile...

İşte XML Web Servisleri tam da bu noktada devreye giriyor! Bu güçlü teknoloji sayesinde, kitap stoklarınızı farklı uygulamalara açarak, müşterilerinizin platform ve dil sınırlamaları olmadan size ulaşmalarını sağlayabilirsiniz. Bunun için aşağıdaki yöntem izlenebilir.

1. **Kitap Listesi Alma (XML Web Servisleri):** Kitap satış sisteminizde, müşterilerin istek gönderdiği bir web servisi bulunur. Bu web servisi, müşterilerin sorması halinde mevcut kitapların bir listesini XML formatında geri döndürür. Örneğin, müşteri "Tüm kitapları listele" diye bir istek gönderdiğinde, sistem XML formatında bir cevap gönderir. Bu XML dosyasında kitapların isimleri, yazarları, fiyatları gibi bilgiler yer alır.
2. **Müşteri Sorgusu (XML):** Müşteri, kendi uygulamasında bir sorgu oluşturur. Örneğin, "Romanları listele" gibi bir sorgu olabilir. Bu sorgu, XML formatında bir istek olarak hazırlanır. Örneğin:

```
<query>
    <type>roman</type>
</query>
```

3. **Web Servisi İşlemi (XML Web Servisi):** Müşterinin oluşturduğu bu XML isteği, web servisine gönderilir. Web servisi bu isteği alır, içindeki bilgilere bakar ve uygun yanıtı hazırlar. Örneğin, bu sorgu sonucunda XML formatında bir cevap hazırlanır:

```
<books>
    <book>
        <title>Şeker Portakalı</title>
        <author>Jose Mauro de Vasconcelos</author>
        <price>25.00</price>
    </book>
    <book>
        <title>Yaban</title>
        <author>Halit Ziya Uşaklıgil</author>
        <price>20.00</price>
    </book>
    <!-- Diğer kitaplar -->
</books>
```

4. **Müşteriye Yanıt (XML):** Web servisi, müşterinin isteğine uygun olarak hazırladığı XML yanıtını müşteriye gönderir. Müşteri bu yanıtı alır, XML dosyasını işleyerek istediği bilgilere ulaşır.

Bu basit örnekte, XML Web Servisleri sayesinde farklı sistemler arasında iletişim kurulabiliyor. XML formatı, verilerin bir sistemden diğerine taşınmasını sağlayan standart bir formattır ve bu sayede farklı platformlar arasında sorunsuz bir iletişim sağlanabilir.

XML Web Servis Tanım Dili (WSDL)

XML Web Servis Tanım Dili (WSDL), XML tabanlı web servislerinin nasıl kullanılacağını tanımlayan bir standarttır. Şimdi basit bir örnek üzerinden WSDL'yi anlatalım:

Yukarıdaki örnekte verdiğimiz gibi bir kitap satıcısının müşterilerinin XML tabanlı web servisini kullanarak kitapları sorgulaması istendiğini varsayalım. WSDL, bu web servisini tanımlamak için kullanılır.

1. **Servis Tanımı (WSDL):** İlk adım olarak, web servisinin ne tür işlemleri kabul edeceğini ve nasıl kullanılacağını tanımlayan bir WSDL belgesinin oluşturulması gerekir. Bu belge, XML formatında yazılır ve web servisinin işlevselliğini ve erişim yöntemlerini açıklar. Örneğin, bir kitap listesi alma servisinin WSDL belgesi şöyle olabilir:

```
<definitions name="BookService" targetNamespace="http://example.com/books">
    <service name="BookService">
        <port name="BookPort" binding="tns:BookBinding">
            <soap:address location="http://example.com/books/service"/>
        </port>
    </service>
    <binding name="BookBinding" type="tns:BookPortType">
        <soap:binding                                                style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="getBookList">
            <soap:operation soapAction="getBookList"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
    <portType name="BookPortType">
        <operation name="getBookList">
            <input message="tns:getBookList"/>
            <output message="tns:getBookListResponse"/>
        </operation>
    </portType>
    <message name="getBookList"/>
    <message name="getBookListResponse"/>
</definitions>
```

Bu WSDL belgesi, "BookService" adında bir servisi ve "getBookList" adında bir işlemi tanımlar. Ayrıca, bu servise SOAP (Simple Object Access Protocol) protokolünü kullanarak erişilebileceğini ve bu işlemin giriş ve çıkış parametrelerinin nasıl olacağını belirtir.

2. **Servis Kullanımı (XML İsteği):** Müşteriler, WSDL belgesine dayanarak web servisine erişebilirler. Örneğin, "getBookList" işlemini çağırmak için müşteri bir XML isteği oluşturur:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://example.com/books">
   <soapenv:Header/>
   <soapenv:Body>
      <web:getBookList/>
   </soapenv:Body>
</soapenv:Envelope>
```

Bu istek, web servisine gönderilir ve belirtilen işlemi (örneğin, "getBookList") çağırır.

3. **Servis Yanıtı (XML Cevabı):** Web servisi, isteğe yanıt olarak bir XML belgesi döndürür. Örneğin, "getBookList" işlemi çağrıldığında, servis mevcut kitapların bir listesini döndürebilir:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://example.com/books">
   <soapenv:Header/>
   <soapenv:Body>
      <web:getBookListResponse>
         <bookList>
            <book>
               <title>Kitap 1</title>
               <author>Yazar 1</author>
               <price>20.00</price>
            </book>
            <book>
               <title>Kitap 2</title>
               <author>Yazar 2</author>
               <price>25.00</price>
            </book>
            <!-- Diğer kitaplar -->
         </bookList>
      </web:getBookListResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Bu yanıt, istemciye geri döndürülür ve istemci tarafından işlenerek kullanıcıya gösterilir. Bu örnekler, WSDL web servislerinin nasıl kullanılacağını ve müşterilerin bu servislerden nasıl sorgu yapabileceğini anlatmaktadır.

XML Web Servisleri ve WSDL'nin kullanılmasının hem avantajları hem de dezavantajları bulunmaktadır.

Avantajlar:

1. **Platform Bağımsızlık:** XML tabanlı web servislerinin kullanılması, farklı platformlar arasında iletişim kurmayı sağlar. Bu, farklı programlama dilleri ve işletim sistemlerinin birbirleriyle entegre olmasını kolaylaştırır.
2. **İşbirliği Yeteneği:** XML Web Servisleri, farklı şirketlerin ve sistemlerin birlikte çalışmasını sağlar. Bu, işbirliği gerektiren uygulamalar ve sistemler arasında veri paylaşımını kolaylaştırır.
3. **Geniş Kullanılabilirlik:** XML ve WSDL standartları, yaygın olarak desteklenir ve geniş bir kullanılabilirliğe sahiptir. Bu da bu teknolojilerin uygulanmasını ve benimsenmesini kolaylaştırır.
4. **Esneklik:** XML tabanlı web servisleri, veri alışverişinde esneklik sağlar. Veri formatları ve iletişim protokolleri standartlaştırıldığı için, değişiklikler kolayca yapılabilir ve uyum sağlanabilir.

Dezavantajlar:

1. **Performans:** XML Web Servisleri, veri alışverişinde XML kullanması nedeniyle bazen performans sorunlarına neden olabilir. XML'in ağırlığı ve işleme maliyeti, özellikle büyük veri miktarlarıyla çalışırken performansı olumsuz etkileyebilir.
2. **Karmaşıklık:** WSDL belgeleri ve XML yapıları karmaşık olabilir. Bu da uygulamaların geliştirilmesi ve bakımının daha zor olmasına neden olabilir.
3. **Veri Boyutu:** XML tabanlı iletişim, veri boyutunu artırabilir. Bu da ağ trafiğini ve veri transfer maliyetlerini artırabilir.
4. **Alternatif Teknolojiler:** Gelişen teknoloji ile XML tabanlı web servislerine alternatif teknolojiler de ortaya çıkmıştır. Örneğin, JSON tabanlı RESTful servisler, daha hafif ve daha hızlı bir iletişim sağlar.

Bu avantajlar ve dezavantajlar, XML Web Servisleri ve WSDL'nin kullanılmasıyla ilgili karar verirken göz önünde bulundurulması gereken önemli faktörlerdir. Uygulamanın gereksinimleri ve ihtiyaçları doğrultusunda en uygun teknoloji seçilmelidir.

XML SOAP (Simple Object Access Protocol) Basit Nesne Erişim Protokolü

SOAP Nedir?

SOAP, **Simple Object Access Protocol**'ün (Basit Nesne Erişim Protokolü) kısaltmasıdır ve internet üzerinden uygulamalar arasında bilgi alışverişi yapmak için kullanılan bir **protokoldür**. XML tabanlı bir protokoldür, yani mesajlar XML formatında gönderilir ve alınır. SOAP, istemci-sunucu mimarisini kullanır, yani bir istemci sunucudan bilgi ister ve sunucu bu bilgiyi istemciye geri gönderir.
SOAP'ın temel amacı, farklı programlama dillerinde ve platformlarda yazılmış uygulamaların birbirleriyle sorunsuz bir şekilde iletişim kurabilmesini sağlamaktır. Bunu, mesajları standart bir formatta tanımlayarak ve bu mesajları taşımak için standart bir iletişim protokolü kullanarak yapar.
SOAP ile ne yapılabilir?

- Farklı programlama dilleri ve platformları kullanan uygulamalar arasında veri alışverişi yapılabilir.
- Web servisleri aracılığıyla uygulamalara erişilebilir.

- Uzaktan prosedür çağrıları (RPC) yapılabilir.
- Uzaktan veri erişimi sağlanabilir.

SOAP nasıl çalışır?
SOAP ile bir uygulama, bir sunucuya **SOAP mesajı** gönderir. Bu mesaj, bir **istek** (request) veya bir **yanıt** (response) olabilir. İstek mesajları, sunucudan bilgi veya işlem talep eder. Yanıt mesajları ise sunucunun isteğe verdiği cevabı içerir.
SOAP mesajları, **zarf** (envelope) ve **gövde** (body) olmak üzere iki bölümden oluşur. Zarf, mesajın kimden kime gönderildiğini, mesajın türünü ve diğer meta verileri içerir. Gövde ise mesajın içeriğini içerir.

İpucu: *SOAP mesajları, HTTP veya HTTPS gibi bir protokol aracılığıyla taşınır.*

SOAP Örneği:
Kitap satan bir e-ticaret web sitesi olduğunu düşünün. Müşteri, bir ürünü sepete eklemek istediğinde, web sitesi SOAP mesajı kullanarak sunucuya bir istek gönderir. Bu mesaj, ürünün kimliğini ve miktarını içerir. Sunucu, mesajı işler ve müşterinin sepetine ürünü ekler. Ardından, müşteriye bir yanıt mesajı gönderir. Bu mesaj, sepetin içeriğini ve toplam tutarı içerir.

1. **İstek Oluşturma (XML SOAP):** İlk olarak, satın almak istediğin kitapla ilgili bir SOAP isteği oluşturmak gerekir. Bu istek, XML formatında bir SOAP mesajıdır. Örneğin:

```
<SOAP-ENV:Envelope      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:buyBook xmlns:m="https://example.com/bookstore">
            <bookID>123456</bookID>
            <quantity>1</quantity>
        </m:buyBook>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

2. Bu SOAP isteği, "buyBook" adında bir işlemi ve bu işlem için gerekli parametreleri içerir.
3. **SOAP İstek Gönderme:** Şimdi bu SOAP isteğini, kitap satıcısının web servisine göndermek gerekir. SOAP isteği, HTTP(S) protokolü üzerinden gönderilir. Web servisi, bu isteği alır ve içindeki işlemi gerçekleştirir.
4. **İşlem Gerçekleştirme:** Web servisi, aldığı SOAP isteğini işler. Örneğin, verilen kitap ID'sine göre stoklarını kontrol eder ve gerekli işlemleri gerçekleştirir (örneğin, kitabı satışa çıkarır veya stoktan düşer).
5. **Cevap Oluşturma (XML SOAP):** İşlem başarıyla gerçekleştirildikten sonra, web servisi bir SOAP yanıtı oluşturur. Bu yanıt da XML formatında bir SOAP mesajıdır. Örneğin:

```
<SOAP-ENV:Envelope      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <m:buyBookResponse xmlns:m="https://example.com/bookstore">
            <status>success</status>
            <message> Kitabınız başarıyla satın alındı.</message>
        </m:buyBookResponse>
```

```
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

6. **SOAP Cevabını Alma:** Son olarak, istek doğrultusunda oluşturulan SOAP yanıtı alınır. Bu yanıtı işleyerek, kitabın başarıyla satın alındığını veya bir hata oluştuğu öğrenilebilir.

XML SOAP, web servisleri arasında güvenli ve standart bir iletişim sağlar. İstemci ve sunucu arasındaki veri alışverişi, XML formatında yapılandırılmış SOAP mesajları aracılığıyla gerçekleşir. Bu sayede farklı platformlar arasında tutarlı bir iletişim sağlanabilir.
XML SOAP'un bazı avantajları ve dezavantajları vardır:

Avantajlar:

1. **Platform Bağımsızlığı:** XML SOAP, farklı platformlar arasında iletişim kurabilmek için kullanılabilir. Bu, farklı programlama dilleri ve işletim sistemleri arasında uyumluluğu artırır.
2. **Standart Protokol:** XML SOAP, standart bir iletişim protokolüdür. Bu standartlık, farklı sistemler arasında entegrasyonun kolaylaştırılmasına ve uyumluluğun artırılmasına yardımcı olur.
3. **Güvenlik:** XML SOAP, iletişimin güvenliğini sağlamak için çeşitli güvenlik mekanizmalarını destekler. Bu, veri alışverişi sırasında güvenlik risklerini azaltır.
4. **Desteklenen Araçlar:** XML SOAP'u destekleyen birçok geliştirme aracı, kütüphane ve çerçeve bulunmaktadır. Bu araçlar, XML SOAP tabanlı uygulamaların geliştirilmesini kolaylaştırır.

Dezavantajlar:

1. **Karmaşıklık:** XML SOAP mesajları genellikle karmaşık olabilir. Bu, geliştirme sürecini ve mesajların işlenmesini karmaşık hale getirebilir.
2. **Performans:** XML SOAP mesajlarının işlenmesi ve ağ üzerinde iletilmesi bazen diğer daha hafif protokollere göre daha fazla kaynak gerektirebilir. Bu, performans açısından dezavantaj olabilir.
3. **Veri Boyutu:** XML SOAP mesajları genellikle büyük olabilir, çünkü XML formatı metin tabanlıdır ve belirli bir düzende yapılandırılmıştır. Bu, veri alışverişinin ağırlığını artırabilir ve bant genişliği tüketimini artırabilir.
4. **Alternatif Protokollerin Varlığı:** Gelişen teknoloji ile birlikte, XML SOAP'un yerini daha hafif ve daha performanslı protokoller almıştır. Bu alternatif protokoller, özellikle hafif ve dağıtık sistemler için daha uygun olabilir.

XML RDF

RDF (Resource Description Framework), "Kaynak Tanımlama Çerçevesi" anlamına gelir ve web üzerindeki kaynakları tanımlamak için kullanılan bir standarttır. Bu çerçeve, bilgisayarlar tarafından okunacak ve anlaşılacak şekilde özellikle tasarlanmıştır, yani insanlara gösterilmek üzere değildir. RDF, bilgileri XML biçiminde ifade eder ve bu nedenle web tarafından işlenebilir veri olarak kullanılır. Ayrıca, RDF, W3C'nin Anlamsal Web Etkinliğinin bir parçasıdır ve 10 Şubat 2004 tarihli bir W3C Tavsiyesi olarak kabul edilmiştir.

RDF kullanım alanlarına aşağıdaki örnekler gösterilebilir.

- Fiyat ve stok durumu gibi alışveriş öğelerinin özelliklerini açıklama
- Web etkinlikleri için zaman çizelgelerini açıklama
- Web sayfalarına ilişkin bilgilerin açıklanması (içerik, yazar, oluşturulma ve değiştirilme tarihi)
- Web resimleri için içerik ve derecelendirmeyi açıklama
- Arama motorları için içeriği tanımlama
- Elektronik kütüphanelerin tanımlanması

RDF, verilerin anlamını tanımlamak için kullanılır ve genellikle üç temel bileşeni vardır: **kaynaklar** (resources), **özellikler** (properties) ve **değerler** (values). Bu bileşenler şöyle açıklanabilir:

1. **Kaynaklar (Resources):** RDF'de, herhangi bir şeyi temsil eden kaynaklar bulunur. Örneğin, bir kitap, bir web sayfası, bir kişi veya bir etkinlik gibi herhangi bir şey bir kaynak olabilir.
2. **Özellikler (Properties):** Kaynaklar arasındaki ilişkileri tanımlayan özellikler bulunur. Örneğin, bir kitabın başlığı, yazarı veya yayınevi gibi özellikler bir kitap kaynağıyla ilişkilendirilebilir.
3. **Değerler (Values):** Özelliklere atanan değerler bulunur. Örneğin, bir kitabın başlığı "Harry Potter and the Philosopher's Stone" olabilir.

RDF Bildirimleri:

RDF, genellikle Turtle veya RDF/XML gibi formatlarda ifade edilir. Bu formatlar, kaynaklar, özellikler ve değerler arasındaki ilişkileri açıkça belirtmek için kullanılır.

RDF'nin avantajları şunlardır:

- **Esneklik:** RDF, web üzerindeki her türlü bilgiyi temsil etmek için kullanılabilir. Bu, çok çeşitli bilgi türlerini ve ilişkilerini kolayca tanımlama esnekliği sağlar.
- **Bağlantılı Veri:** RDF, veriler arasındaki anlamlı ilişkileri tanımlayarak bağlantılı veri oluşturmayı sağlar. Bu da verilerin daha anlamlı ve işlevsel hale gelmesine yardımcı olur.
- **Makine Okunabilirlik:** RDF, verilerin bilgisayarlar tarafından otomatik olarak işlenebilir ve anlamlandırılabilir olmasını sağlar. Bu, veri entegrasyonu, arama ve analiz gibi işlemleri kolaylaştırır.

Ancak RDF'nin dezavantajları da vardır:

- **Karmaşıklık:** RDF, bazen karmaşık ve anlaşılması zor olabilir, özellikle büyük veri kümeleri için.
- **İşleme Maliyeti:** RDF verilerinin işlenmesi ve depolanması bazen maliyetli olabilir, özellikle büyük ölçekli projelerde.
- **Yetersiz Kullanım:** RDF, genellikle belirli uzmanlık gerektiren alanlarda kullanılır ve genel web uygulamalarında daha az yaygındır. Bu da yaygın kullanımını kısıtlayabilir.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cd="http://www.recshop.fake/cd#">
```

```
<rdf:Description
rdf:about="http://www.recshop.fake/cd/Empire Burlesque">
  <cd:artist>Bob Dylan</cd:artist>
  <cd:country>USA</cd:country>
  <cd:company>Columbia</cd:company>
  <cd:price>10.90</cd:price>
  <cd:year>1985</cd:year>
</rdf:Description>

<rdf:Description
rdf:about="http://www.recshop.fake/cd/Hide your heart">
  <cd:artist>Bonnie Tyler</cd:artist>
  <cd:country>UK</cd:country>
  <cd:company>CBS Records</cd:company>
  <cd:price>9.90</cd:price>
  <cd:year>1988</cd:year>
</rdf:Description>
</rdf:RDF>
```

Yukarıdaki örnekte: RDF belgesinin ilk satırı XML bildirimidir. XML bildiriminin ardından RDF belgelerinin kök öğesi gelir: <rdf:RDF> .

xmlns :rdf ad alanı, rdf önekine sahip öğelerin "http://www.w3.org/1999/02/22-rdf-syntax-ns#" ad alanından olduğunu belirtir.

xmlns :cd ad alanı, cd önekine sahip öğelerin "http://www.recshop.fake/cd#" ad alanından olduğunu belirtir.

< rdf:Description> öğesi, rdf:about özelliği tarafından tanımlanan kaynağın açıklamasını içerir .

Öğeler: <cd:artist>, <cd:country>, <cd:company> vb. kaynağın özellikleridir.

Title	Artist	Country	Company	Price	Year
Empire Burlesque	Bob Dylan	USA	Columbia	10.90	1985
Hide your heart	Bonnie Tyler	UK	CBS Records	9.90	1988

RDF Öğeleri

RDF'nin ana öğeleri, kök öğe olan <RDF> ve bir kaynağı tanımlayan açıklama öğesidir.

<rdf:RDF> Öğesi

<rdf:RDF> bir RDF belgesinin kök öğesidir. XML belgesinin bir RDF belgesi olduğunu tanımlar. Ayrıca RDF ad alanına bir referans içerir:

```
<?xml version="1.0"?>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      ...Açıklama burada yer alır...
    </rdf:RDF>
```

<rdf:Description> Öğesi

XML'de, RDF Description elemanı, bir bilgi kaynağını ve onunla ilgili bilgileri temsil etmek için kullanılır. Bu sayede verilerin anlamını ve bağlamını daha net bir şekilde ifade edebilir.

Neler Yapılabilir?

- **Bilgi Kaynaklarını Tanımlama:** RDF Description elemanını kullanarak bir bilgi kaynağını (örneğin, bir kitap, bir kişi veya bir web sitesi) tanımlayabilirsiniz.
- **Özellikleri Tanımlama:** RDF Description elemanının alt öğeleri aracılığıyla bilgi kaynağına ilişkin özellikleri tanımlayabilirsiniz. Örneğin, bir kitap için yazar, başlık, ISBN ve yayın tarihi gibi bilgileri tanımlayabilirsiniz.
- **İlişkileri Gösterme:** RDF Description elemanındaki özellikler aracılığıyla bilgi kaynakları arasındaki ilişkileri gösterebilirsiniz. Örneğin, bir yazarın yazdığı kitapları veya bir kitap kategorisini ilişkilendirebilirsiniz.

Nasıl Yapılır?

XML'de bir RDF Description elemanı oluşturmak için <rdf:Description> etiketini kullanmanız gerekir. Örnek olarak, bir kitap hakkında bilgi içeren bir XML dosyasında şu şekilde bir kod kullanabilirsiniz:

```
<?xml version="1.0" encoding="UTF-8"?>
<kitaplar xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about="http://www.example.com/kitaplar/1">
        <baslik>Veritabanları ve XML</baslik>
        <yazar>Roger Conklin</yazar>
        <yayinEvi>Addison-Wesley</yayinEvi>
        <yayinTarihi>2003</yayinTarihi>
        <tur>Bilgisayar Bilimi</tur>
        <fiyat>50 TL</fiyat>
        <ozet>Veritabanları ve XML hakkında bir kitap...</ozet>
    </rdf:Description>
</kitaplar>
```

Bu kodda, <rdf:Description> etiketi bir RDF Description elemanı oluşturur. rdf:about özelliği, elemanın temsil ettiği bilgi kaynağının URL'sini belirtir. Bu örnekte, bilgi kaynağı, http://www.example.com/kitaplar/1 URL'sindeki bir kitaptır.

Nitelik Olarak Özellikler

Özellik öğeleri aynı zamanda kaynaklar olarak da tanımlanabilir:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:book="http://www.library.com/book#">
    <rdf:Description rdf:about="http://www.library.com/book/12345">
```

```
        <book:author
rdf:resource="http://www.library.com/author/agatha_christie"/>
        <book:title>Mavi Ekspres'te Cinayet</book:title>
    </rdf:Description>
</rdf:RDF>
```

Yukarıdaki örnekte yazar özelliğinin bir değeri yoktur, ancak yazar hakkında bilgi içeren bir kaynağa referansı vardır.

RDF Konteynerleri

XML'de, RDF konteynerleri, birden fazla değeri tek bir özellikte tutmak için kullanılır. Bu sayede, karmaşık ilişkileri ve verileri daha kolay bir şekilde temsil edebilirsiniz.

RDF'de, bilgi kümelerini ve sıralı veya sırasız ilişkileri temsil etmek için üç konteyner türü kullanılır:

RDF'de, bilgi nesnelerini organize etmek ve ilişkilerini göstermek için konteynerler kullanılabilir. En yaygın kullanılan üç konteyner türü şunlardır:

1. <Bag>:

- **Tanım:** Bir **<Bag>**, sıralı veya sırasız olabilen bir değer kümesi tanımlar.
- **Kullanım:** Bir nesnenin birden fazla değere sahip olduğu durumlarda kullanılır. Örneğin, bir kitap için birden fazla yazar veya bir film için birden fazla tür belirtilebilir.

Örnek:

```
<rdf:Description rdf:about="http://www.example.com/kitaplar/12345">
  <dc:creator rdf:resource="http://www.example.com/yazarlar/agatha_christie"/>
  <dc:creator rdf:resource="http://www.example.com/yazarlar/hercule_poirot"/>
  <dc:title>Mavi Ekspres'te Cinayet</dc:title>
</rdf:Description>
```

2. <Seq>:

- **Tanım:** Bir **<Seq>**, sıralı bir değer kümesi tanımlar.
- **Kullanım:** Bir nesnenin birden fazla değere sahip olduğu ve bu değerlerin sıranın önemli olduğu durumlarda kullanılır. Örneğin, bir kitabın bölümlerini veya bir filmin sahnelerini sıralı olarak listelemek için kullanılabilir.

Örnek:

```
<rdf:Description rdf:about="http://www.example.com/filmler/12345">
    <dc:subject rdf:resource="http://www.example.com/konular/gizem"/>
    <dc:subject rdf:resource="http://www.example.com/konular/macera"/>
    <dc:description>Bir grup arkadaşın ıssız bir adaya düşmesi ve hayatta kalmak
için mücadele etmesi hakkında bir film...</dc:description>
</rdf:Description>
```

3. <Alt>:

- **Tanım:** Bir **<Alt>**, alternatif değerleri temsil eden bir değer kümesi tanımlar.
- **Kullanım:** Bir nesnenin birden fazla değere sahip olduğu ve bu değerlerin hangisinin kullanılacağının belirsiz olduğu durumlarda kullanılır. Örneğin, bir kitabın farklı baskılarının ISBN numaralarını veya bir filmin farklı dillerdeki isimlerini listelemek için kullanılabilir.

Örnek:

```
<rdf:Description rdf:about="http://www.example.com/kitaplar/12345">
  <dc:identifier rdf:resource="http://www.example.com/isbn/978-0-14-035697-5"/>
  <dc:identifier rdf:resource="http://www.example.com/isbn/978-0-14-035698-2"/>
  <dc:title>Mavi Ekspres'te Cinayet</dc:title>
</rdf:Description>
```

RDF Koleksiyonları

RDF'de, bilgi nesnelerini organize etmek ve ilişkilerini göstermek için koleksiyonlar kullanılabilir. Koleksiyonlar, birden fazla bilgi nesnesini tek bir birim halinde gruplandırmak için kullanılır. Bu sayede, bilgi nesneleri arasındaki ilişkileri daha net bir şekilde ifade etmek ve verilerin daha kolay işlenmesini sağlamak mümkün hale gelir.

Kullanım Alanları:

- **Listeler:** Bir nodanın birden fazla sıralı değere sahip olduğu durumlarda kullanılır. Örneğin, bir kitabın yazarlarını veya bir filmin türlerini listelemek için kullanılabilir.
- **Setler:** Bir nodanın birden fazla sırasız değere sahip olduğu durumlarda kullanılır. Örneğin, bir kitabın konularını veya bir filmin oyuncularını listelemek için kullanılabilir.
- **Torbalar:** Bir nodanın birden fazla sırasız ve tekrarlanabilen değere sahip olduğu durumlarda kullanılır. Örneğin, bir kitabın baskılarının ISBN numaralarını veya bir filmin farklı dillerdeki isimlerini listelemek için kullanılabilir.

rdf:parseType="Collection" Özelliği Nedir?

rdf:parseType="Collection", bir RDF noda bağlı bir XML öğesinin birden fazla RDF terimini temsil ettiğini göstermek için kullanılır. Bu, bir nodanın tek bir değer yerine bir koleksiyonla ilişkilendirilmesini sağlar.

Kullanım Alanları:

- **Listeler:** Bir nodanın birden fazla sıralı değere sahip olduğu durumlarda kullanılır. Örneğin, bir kitabın yazarlarını veya bir filmin türlerini listelemek için kullanılabilir.
- **Setler:** Bir nodanın birden fazla sırasız değere sahip olduğu durumlarda kullanılır. Örneğin, bir kitabın konularını veya bir filmin oyuncularını listelemek için kullanılabilir.
- **Torbalar:** Bir nodanın birden fazla sırasız ve tekrarlanabilen değere sahip olduğu durumlarda kullanılır. Örneğin, bir kitabın baskılarının ISBN numaralarını veya bir filmin farklı dillerdeki isimlerini listelemek için kullanılabilir.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
    <rdf:Description rdf:about="http://www.example.com/kitaplar/12345">
        <dc:creator rdf:parseType="Collection">
            <rdf:li
rdf:resource="http://www.example.com/yazarlar/agatha_christie"/>
            <rdf:li
rdf:resource="http://www.example.com/yazarlar/hercule_poirot"/>
        </dc:creator>
        <dc:title>Mavi Ekspres'te Cinayet</dc:title>
    </rdf:Description>
</rdf:RDF>
```

Bu örnekte, dc:creator özelliği, rdf:parseType="Collection" özelliğini kullanarak bir liste tanımlamak için kullanılır. Bu liste, kitabın iki yazarını içerir. rdf:li etiketleri, listenin öğelerini temsil eder.

Faydaları:

- **Veri Esnekliği:** rdf:parseType="Collection", bir nodanın birden fazla değere sahip olmasını sağlayarak veri esnekliği sağlar.
- **Veri Temsili:** rdf:parseType="Collection", bilgi nesnelerini daha net bir şekilde temsil etmenizi sağlar.
- **Veri İşleme:** rdf:parseType="Collection", verilerin daha kolay işlenmesini ve sorgulanmasını sağlar.

RDF Şeması ve Uygulama Sınıfları

RDF Şeması (RDFS):

RDF Şeması (RDFS), RDF'de bilgi nesneleri ve ilişkileri tanımlamak için kullanılan bir meta veri dilidir. RDFS, temel ontoloji kavramlarını ve ilişkilerini tanımlamak için kullanılır. Bunlar şunları içerir:

- **Sınıflar:** Gerçek dünyadaki nesneleri veya kavramları temsil eder.
- **Özellikler:** Sınıflar arasındaki ilişkileri ve sınıfların özelliklerini temsil eder.
- **Alt Sınıflar:** Bir sınıfın, daha genel bir sınıftan türediğini gösterir.
- **Özellik Kısıtlamaları:** Özelliklerin değerlerinin hangi türde olması gerektiğini belirtir.

Uygulama Sınıfları:

Uygulama sınıfları, RDFS'de tanımlanan temel kavramları kullanarak belirli bir alan veya uygulama için bilgi nesnelerini ve ilişkilerini tanımlamak için kullanılır. Uygulama sınıfları, RDFS'nin sunduğundan daha fazla ayrıntı ve özel bilgi sağlayarak daha karmaşık ve nüanslı bilgi modelleri oluşturmak için kullanılabilir.

Sınıflar, Özellikler, Alt Sınıflar ve Özellik Kısıtlamaları Tablosu

Terim	Tanım	Örnek Kullanım

Sınıf	Gerçek dünyadaki nesneleri veya kavramları temsil eder.	Kitap, Film, Kişi gibi
Özellik	Sınıflar arasındaki ilişkileri ve sınıfların özelliklerini temsil eder.	yazar, tür, ad, soyad gibi
Alt Sınıf	Bir sınıfın, daha genel bir sınıftan türediğini gösterir.	Roman sınıfı, Kitap sınıfının alt sınıfıdır.
Özellik Kısıtlaması	Özelliklerin değerlerinin hangi türde olması gerektiğini belirtir.	yazar özelliğinin değeri Kişi sınıfından bir nesne olmalıdır.

Ek Bilgiler:

- Bir sınıf birden fazla alt sınıfa sahip olabilir.
- Bir özellik birden fazla sınıf tarafından kullanılabilir.
- Bir özellik kısıtlaması, bir özelliğin değerinin veri türünü, aralığını veya formatını belirleyebilir.

Örnek:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.animals.fake/animals#">
    <rdf:Description rdf:ID="animal">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
    </rdf:Description>
    <rdf:Description rdf:ID="horse">
        <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
        <rdfs:subClassOf rdf:resource="#animal"/>
    </rdf:Description>
</rdf:RDF>
```

Yukarıdaki örnekte, RDF (Resource Description Framework) ile hayvanları ve sınıflandırmalarını tanımlayan bir şema oluşturmaktadır. "Hayvan" adında bir üst sınıf ve "at" adında bir alt sınıf tanımlanır. Bu, tüm atların hayvan sınıfından türediğini ve hayvanların özelliklerini paylaştıklarını gösterir. Kod, RDF ve RDFS (RDF Şeması) için ad alanları kullanır ve göreceli URI'leri çözmek için bir taban URI belirler.

Aynı Örneğin Daha Kısa Yazılışı:

RDFS sınıfı bir RDF kaynağı olduğundan, yukarıdaki örneği rdf:Description yerine rdfs:Class kullanarak kısaltabilir ve rdf:type bilgisini bırakabiliriz:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
         xml:base="http://www.animals.fake/animals#">
```

```
    <rdfs:Class rdf:ID="animal"/>
    <rdfs:Class rdf:ID="horse">
        <rdfs:subClassOf rdf:resource="#animal"/>
    </rdfs:Class>
</rdf:RDF>
```

RDFS / RDF Sınıfları

Sınıf	Tanım	Alt Sınıfı
rdfs:Class	Gerçek dünyadaki nesneleri veya kavramları temsil eder.	rdfs:Datatype (Veri Tipleri) rdfs:Resource (Tüm Kaynaklar) rdfs:Literal (Değerler - Metin ve Sayılar) rdfs:Container (Konteynerler) rdfs:List (Listeler) rdfs:Property (Özellikler) rdfs:Statement (İfadeler)
rdfs:Datatype	Veri tiplerini (örneğin, string, integer, date) temsil eder.	Yok
rdfs:Resource	Tüm RDF kaynaklarını temsil eder.	rdfs:Literal (Değerler - Metin ve Sayılar) rdfs:Container (Konteynerler)
rdfs:Literal	Metin veya sayısal değerleri temsil eder.	Yok
rdfs:Container	Diğer RDF kaynaklarını (örneğin, listeler, torbalar, diziler) gruplandırmak için kullanılır.	rdfs:List (Listeler) rdfs:Bag (Sırasız Konteynerler) rdfs:Seq (Sıralı Konteynerler)
rdfs:List	Sıralı bir kaynak koleksiyonunu temsil eder.	Yok
rdfs:Property	Sınıflar arasındaki ilişkileri veya bir sınıfın özelliklerini temsil eder.	Yok
rdfs:Statement	Bir özne, yüklem ve nesneden oluşan bir RDF ifadesini temsil eder.	Yok
rdfs:Alt	Bir kaynağın alternatif değerlerini temsil eder.	Yok

rdfs:Bag	Sırasız ve tekrarlanabilen bir kaynak koleksiyonunu temsil eder.	Yok
rdfs:Seq	Sıralı ve tekrarlanmayan bir kaynak koleksiyonunu temsil eder.	Yok
rdfs:XMLLiteral	XML biçimindeki değerleri temsil eder.	Alt sınıfı yoktur, rdfs:Literal sınıfının bir alt türü olarak düşünülebilir.

RDFS / RDF Özellikleri

Terim	Açıklaması	Örnek
rdfs:domain	Özelliğin uygulandığı kaynak sınıfı	Yazar özelliğinin domain'i Kişi sınıfı olabilir.
rdfs:range	Özelliğin değerlerinin ait olduğu sınıf	Yazar özelliğinin range'i Kişi sınıfı olabilir.
rdfs:subPropertyOf	Bir özelliğin başka bir özelliğin alt özelliği olduğunu gösterir	tür özelliği, kategorisi özelliğinin alt özelliği olabilir.
rdfs:subClassOf	Bir sınıfın başka bir sınıfın alt sınıfı olduğunu gösterir	Roman sınıfı, Kitap sınıfının alt sınıfı olabilir.
rdfs:comment	Kaynağın insan tarafından okunabilir açıklaması	Kitap sınıfının yorumu "Farklı konularda yazılmış eserler" olabilir.
rdfs:label	Kaynağın insan tarafından okunabilir etiket (adı)	Kitap sınıfının label'ı "Kitap" olabilir.
rdfs:isDefinedBy	Kaynağın tanımlandığı kaynak	Kitap sınıfının tanımı, bir ontoloji dosyasında bulunabilir.
rdfs:seeAlso	Kaynak hakkında ek bilgi	Kitap sınıfı için Yayıncı sınıfına bakın bilgisi bulunabilir.
rdfs:member	Bir koleksiyonun üyesi	Yazarlar koleksiyonunun bir üyesi Agatha Christie olabilir.
rdf:first	Listenin ilk elemanı	Yazarlar listesindeki ilk yazar Agatha Christie olabilir.

rdf:rest	Listenin geri kalanı	Yazarlar listesinde Agatha Christie'den sonraki yazarlar listenin geri kalanını oluşturur.
rdf:subject	Bir RDF ifadesindeki özne	Agatha Christie yazdı ifadesinde özne Agatha Christie olabilir.
rdf:predicate	Bir RDF ifadesindeki yüklem	Agatha Christie yazdı ifadesinde yüklem yazdı fiilidir.
rdf:object	Bir RDF ifadesindeki nesne	Agatha Christie yazdı ifadesinde nesne yazılan eser olabilir.
rdf:value	Değerler için kullanılan özellik	fiyat özelliği bir ürünün fiyatını tutar.
rdf:type	Kaynağın ait olduğu sınıf	Agatha Christie örneği Kişi sınıfına ait olabilir.

RDF Nitelikleri

Nitelik	Açıklama	Örnek
rdf:about	Tanımlanan kaynağı belirtir.	rdf:about="#kitap1" (kitap1 kimliği)
rdf:Description	Kaynak tanımı için kullanılır.	Her kaynak <rdf:Description> içinde tanımlanır.
rdf:resource	Kaynağı tanımlar.	dc:creator="yazarlar/agatha_christie"
rdf:datatype	Veri türünü tanımlar.	yas xsd:date (tarih formatı)
rdf:ID	Benzersiz kimlik tanımlar.	rdf:ID="kitapX" (kitapX kimliği)
rdf:li	Liste öğesini tanımlar.	<rdf:li>Yazar Adı</rdf:li>
rdf:nodeID	Düğüm kimliğini tanımlar.	rdf:nodeID="yazarY" (yazarY kimliği)
rdf:RDF	RDF belgesinin köküdür.	Her RDF belgesi rdf:RDF ile başlar.
xml:base	XML tabanını tanımlar.	Göreceli URI'lerin çözümü için.
xml:lang	Dil bilgisini tanımlar.	Çok dilli içeriklerde kullanılır.

XML RSS (Really Simple Syndication)

RSS Nedir?

RSS, Really Simple Syndication'ın kısaltmasıdır ve Türkçe'ye Gerçekten Basit Dağıtım olarak çevrilebilir. İnternette sık güncellenen web sitelerinin (haber siteleri, bloglar, podcastler vb.)

yeni içeriklerini takip etmenizi sağlayan bir sistemdir. Kullandığı dosya biçimleri **.rss** ve **.xml**'dir. RSS; web geliştiriciler tarafından RSS akışı oluşturacak .rss veya .xml dosyaların web sitelerinde oluşturulmasının ardından istemci tarafında yani web ziyaretçileri tarafından da RSS okuyucu yazılımlara söz konusu dosyaların bağlantısının eklenmesiyle süreç tamamlanmaktadır. RSS kaynağı sağlayan internet sitelerinde genellikle şu simgeler eklenir.

RSS kısaltmasının açılımı ve zaman içinde gelişimi şöyledir:

- Rich Site Summary (RSS 0.91) (Zengin Site Özeti)
- RDF Site Summary (RSS 0.9 and 1.0) (RDF Site Özeti)
- Really Simple Syndication (RSS 2.0.0) (Çok Basit Besleme)

RSS, XML tabanlı bir dosya formatı kullanır. Bu dosya, sitenin güncel içeriklerinin başlıklarını, özetlerini, bağlantı adreslerini ve diğer bazı bilgilerini içerir. RSS okuyucu adı verilen bir program aracılığıyla bu dosyayı takip ederek, web sitesini ziyaret etmeden yeni içeriklerden haberdar olabilirsiniz.

RSS Nasıl Çalışır?

1. Abone Olma: İlgi duyduğunuz web sitenin RSS beslemesine abone olursunuz. Bunu, sitenin RSS bağlantısını kopyalayıp RSS okuyucunuza yapıştırarak veya sitenin sunduğu abonelik seçeneklerini kullanarak yapabilirsiniz.
2. RSS Beslemesini Oku: RSS okuyucu, abone olduğunuz sitelerin RSS beslemelerini otomatik olarak kontrol eder. Yeni içerikler eklendiğinde, RSS okuyucuda bir bildirim alırsınız.
3. İçeriği Görüntüleme: Bildirime tıkladığınızda, RSS okuyucuda içeriğin başlığını, özetini ve bağlantısını görebilirsiniz. İstediğiniz içeriğin bağlantısına tıklayarak web sitesine gidip içeriği tam olarak okuyabilirsiniz.

RSS Örneği

Aşağıda yer alan örnek RSS Belgesindeki ilk satır - XML bildirimi - belgede kullanılan XML sürümünü ve karakter kodlamasını tanımlar. Örneğimizde verilen belge XML'in 1.0 versiyonuna uygundur ve UTF-8 karakter setini kullanır.

Sonraki satır, bunun bir RSS belgesi olduğunu tanımlayan RSS beyanıdır (Örneğimizde RSS sürüm 2.0).

Sonraki satır <channel> öğesini içerir. Bu öğe RSS beslemesini tanımlamak için kullanılır.

<channel> öğesinin gerekli üç alt öğesi vardır:

<title> - Kanalın başlığını tanımlar (örn. **Örnek Blog**)

<link> - Kanala olan köprüyü tanımlar (örn. **https://www.ornekblog.com**)

<açıklama> - Kanalı açıklar (örn. **Teknoloji ve Bilim Hakkında Bilgiler**)

Her <channel> elemanı bir veya daha fazla <item> elemanına sahip olabilir.

Her <item> öğesi, RSS beslemesindeki bir makaleyi veya "hikayeyi" tanımlar.

<item> öğesinin gerekli alt öğeleri vardır:

<title> - Öğenin başlığını tanımlar (örn. **Yapay Zeka Gelecekte Nasıl Bir Rol Oynayacak?**)

<link> - Öğeye olan köprüyü tanımlar (örn. **https://www.ornekblog.com/yapay-zeka-gelecekte-nasil-bir-rol-oynayacak**)

<açıklama> - Öğeyi açıklar (örn. **Yapay zekanın gelişimi ve gelecekte insanlık üzerinde yaratacağı etkiler hakkında bir makale.**)

<pubDate> Öğenin yayınlanma zamanını tanımlar. (örn. **Mon, 04 May 2024 15:30:00 +0300**)

Son olarak son iki satır <channel> ve <rss> öğelerini kapatır.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
  <title>Örnek Blog</title>
  <link>https://www.ornekblog.com</link>
  <description>Teknoloji ve Bilim Hakkında Bilgiler</description>
  <item>
    <title>Yapay Zeka Gelecekte Nasıl Bir Rol Oynayacak?</title>
    <link>https://www.ornekblog.com/yapay-zeka-gelecekte-nasil-bir-rol-
oynayacak</link>
    <description>Yapay zekanın gelişimi ve gelecekte insanlık üzerinde
yaratacağı etkiler hakkında bir makale.</description>
    <pubDate>Mon, 04 May 2024 15:30:00 +0300</pubDate>
  </item>
  <item>
    <title>Yeni Uzay Teleskobu Başarıyla Fırlatıldı</title>
    <link>https://www.ornekblog.com/yeni-uzay-teleskobu-basariyla-
firlatildi</link>
    <description>Yeni uzay teleskobunun özellikleri ve evreni keşfetmede
nasıl yardımcı olacağı hakkında bir makale.</description>
    <pubDate>Sun, 29 Apr 2024 12:00:00 +0300</pubDate>
  </item>
</channel>
</rss>
```

Bu kodda, ***channel*** öğesi blogun adını, bağlantısını ve açıklamasını içerir. ***item*** öğeleri ise her bir yeni makaleyi temsil eder ve makalenin başlığını, bağlantısını, özetini ve yayınlanma tarihini içerir.

RSS'in Faydaları

- **Zaman Tasarrufu:** RSS sayesinde, takip ettiğiniz tüm sitelerin yeni içeriklerini tek bir yerden takip edebilir ve zamandan tasarruf edebilirsiniz.

- **Güncel Kalma:** RSS, yeni içeriklerden anında haberdar olmanızı sağlayarak güncel kalmanıza yardımcı olur.
- **Bilgi Düzenleme:** RSS, ilgi alanlarınıza göre bilgi almanızı ve gereksiz içeriklerden uzak durmanızı sağlar.

RSS Kullanmanın Dezavantajları

- **Teknik Gereklilik:** RSS kullanmak için bir RSS okuyucu programı kurmanız gerekir.
- **Bakım Zorluğu:** Bazı siteler RSS beslemelerini güncellemeyi bırakabilir veya formatlarını değiştirebilir.
- **Bilgi Kirliliği:** Çok fazla RSS beslemesine abone olmak, bilgi kirliliğine yol açabilir.

RSS Kimler İçin Faydalıdır?

RSS, web sitelerini **sık sık güncelleyen** içerik üreticileri için oldukça faydalı bir araçtır.

- **Haber siteleri:** RSS, haber sitelerinin güncel haberlerini, başlıkları, tarihlerini ve özetlerini takip etmenizi sağlar. Bu sayede, web sitesini ziyaret etmeden bile en son gelişmelerden haberdar olabilirsiniz.
- **Şirketler:** Şirketler, RSS kanalları aracılığıyla yeni ürünlerini, haberlerini ve güncellemelerini duyurabilirler. Bu sayede müşterileri ve iş ortakları ile daha kolay iletişim kurabilirler.
- **Takvimler:** RSS, yaklaşan etkinlikleri, önemli günleri ve randevuları takip etmenizi sağlar. Bu sayede zamanınızı daha iyi planlayabilir ve önemli bir şeyi kaçırma riskini en aza indirebilirsiniz.
- **Bloglar ve forumlar:** Bloglar ve forumlar da RSS kanalları aracılığıyla yeni içeriklerini takipçileriyle paylaşabilirler. Bu sayede içerik üreticileri, takipçileriyle daha fazla etkileşime girebilir ve onları sitenin güncel gelişmelerinden haberdar edebilirler.

Önemli Hatırlatma: RSS kullanmak için bir RSS okuyucu programı kurmanız gerekir. Feedly, Inoreader gibi birçok farklı RSS okuyucu programı mevcuttur.

Hangi RSS Versiyonu Kullanılmalı?

RSS 0.91 ve RSS 2.0, RSS 1.0'a göre daha kolay anlaşılır ve kullanılır. Bu bölümde verilen bilgiler de **RSS 2.0** versiyonuna dayanmaktadır.

RSS 2.0, basit ve katı sözdizimi kurallarına sahiptir. Bu nedenle, yeni başlayanlar için daha uygundur.

RSS, resmi bir web standardı değildir. Fakat, birçok web sitesi ve uygulama tarafından kullanılmaktadır.

Günümüzde kullanılan RSS versiyonları:

- **%50:** RSS 0.91
- **%25:** RSS 1.0
- **%25:** RSS 0.9x ve RSS 2.0 (diğer versiyonlar)

RSS Nasıl Çalışır?

- **RSS kullanımı için:**
    1. Bir RSS belgesi oluşturulur ve .xml uzantısıyla kaydedilir.
    2. Dosya web sitesine yüklenir.
    3. Bir RSS okuyucuyu yüklenir veya çevrimiçi yazılımlara kaydolunur.
    4. Okuyucu, web sitenizi her gün tarar ve yeni RSS belgelerini otomatik olarak bulur.
    5. Takipçiler, RSS toplayıcı aracılığıyla yeni içeriklere bağlantıyı bulabilirler.

RSS XML olduğundan şu hususlar unutulmamalıdır:

- Tüm öğelerin bir kapanış etiketi olmalıdır.
- Öğeler büyük/küçük harfe duyarlıdır.
- Öğeler düzgün bir şekilde iç içe yerleştirilmelidir.
- Özellik değerleri her zaman alıntılanmalıdır.

Yorum satırları

RSS'de yorum yazmanın sözdizimi HTML'deki gibidir.

```
<!-- Bu bir RSS yorum satırıdır -->
```

RSS Öğeleri

<channel> Öğesi:

RSS beslemesinin temelini oluşturan <channel> öğesi, beslemenin başlığını, bağlantısını ve açıklamasını içerir. Bu öğe, beslemeyi tanımlayan ve diğer öğelerin ana çerçevesini oluşturan önemli bir öğedir.

<channel> Öğesinin Gerekli Alt Öğeleri:

1. **<title>:** Bu alt öğe, RSS beslemesinin başlığını tanımlar ve beslemenin içeriğini özetleyen kısa bir metin içerir. Başlık, beslemenin tarayıcıda veya RSS okuyucu programında nasıl gösterileceğini belirler.
2. **<link>:** Bu alt öğe, RSS beslemesinin URL adresini (web sitesi bağlantısını) tanımlar. Kullanıcılar, bu bağlantıyı tıklayarak beslemenin kaynağına, yani web sitesine gidebilirler.
3. **<description>:** Bu alt öğe, RSS beslemesini kısaca açıklar. Açıklama metni, beslemenin içeriğini ve sunduğu konuları hakkında genel bir bilgi verir.

<channel> Öğesinin İsteğe Bağlı Alt Öğeleri:

<channel> öğesi, gerekli alt öğelere ek olarak, besleme hakkında daha fazla bilgi sağlayan çeşitli isteğe bağlı alt öğelere de sahip olabilir. Bu alt öğelerden bazı önemlileri şunlardır:

<category> Öğesi

<category> alt öğesi feed'iniz için bir kategori belirtmek amacıyla kullanılır. Bu öğe ile RSS toplayıcılarının siteleri kategoriye göre gruplandırmasına olanak tanır.

Yukarıdaki RSS belgesinin kategorisi şöyle olabilir:

```
<category>Teknoloji ve Bilim</category>
```

<telif hakkı> Öğesi

<copyright> alt öğesi, telif hakkıyla korunan materyal hakkında bilgi verir.

Yukarıdaki RSS belgesinin telif hakkı şöyle olabilir:

```
<copyright>Tüm hakları Örnek Blog'a aittir. ©2024 </copyright>
```

<image> Öğesi

<image> alt öğesi, toplayıcılar bir yayın sunduğunda bir görselin görüntülenmesine olanak tanır.

<image> öğesinin gerekli üç alt öğesi vardır:

- <url> - Resmin URL'sini tanımlar
- <title> - Görüntünün gösterilememesi durumunda görüntülenecek metni tanımlar
- <link> - Kanalı sunan web sitesine giden köprüyü tanımlar

Yukarıdaki RSS belgesinin resmi şöyle olabilir:

```
<image>
  <url>https://www.ornekblog.com/logo.gif</url>
  <title>Örnek Blog</title>
  <link>https://www.ornekblog.com</link>
</image>
```

<dil> Öğesi

<language> alt öğesi, belgenizi yazmak için kullanılan dili belirtmek için kullanılır.

<language> öğesi, RSS toplayıcılarının siteleri dile göre gruplandırmasına olanak tanır.

Yukarıdaki RSS belgesinin dili şöyle olabilir:

```
<language>tr-TR</language>
```

<item> Öğesi

RSS beslemeleri, <channel> öğesi içinde bir veya daha fazla <item> öğesi içerir. Her <item> öğesi, beslemedeki bir makaleyi veya "hikayeyi" tanımlar ve bu makaleyle ilgili bilgileri içerir.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
  <title>Örnek Blog</title>
  <link>https://www.ornekblog.com</link>
  <description>Teknoloji ve Bilim Hakkında Bilgiler</description>
  <item>
    <title>Yeni Uzay Teleskobu Başarıyla Fırlatıldı</title>
```

```
    <link>https://www.ornekblog.com/yeni-uzay-teleskobu-basariyla-
firlatildi</link>
    <description>Yeni uzay teleskobunun özellikleri ve evreni keşfetmede nasıl
yardımcı olacağı hakkında bir makale.</description>
    <pubDate>Sun, 29 Apr 2024 12:00:00 +0300</pubDate>
  </item>
</channel>
</rss>
```

<item> öğesinin gerekli üç alt öğesi vardır:

- <title> - Öğenin başlığını tanımlar
- <link> - Öğeye olan köprüyü tanımlar
- <açıklama> - Öğeyi açıklar

Ayrıca <item> öğesinin çeşitli isteğe bağlı alt öğeleri vardır. Aşağıda en önemlilerini açıklayacağız.

<yazar> Öğesi

<author> alt öğesi, bir öğenin yazarının e-posta adresini belirtmek için kullanılır.

Not: Spam e-postaları önlemek için bazı geliştiriciler <author> öğesini eklemeyi tercih etmemektedir.

<enclosure> Öğesi

<enclosure> alt öğesi, bir medya dosyasının bir öğeye dahil edilmesini sağlar.

<enclosure> öğesinin üç gerekli özelliği vardır:

- url - Medya dosyasının URL'sini tanımlar
- uzunluk - Medya dosyasının uzunluğunu (bayt cinsinden) tanımlar
- type - Medya dosyasının türünü tanımlar

Yukarıdaki RSS belgesindeki öğenin içerdiği bir medya dosyası şunlar olabilir:

```
<enclosure url="https://www.ornekblog.com/rss.mp3" length="5000" type="audio/mpeg"
/>
```

RSS Okuyucular

RSS Akışlarını okumak için bir RSS Okuyucu kullanmanız gerekir. RSS okuyucular birçok farklı cihaz ve işletim sistemi için mevcuttur.

Farklı RSS Okuyucu Türleri:

- **Web Hizmetleri Olarak Çalışanlar:** Bu okuyuculara internet tarayıcınız üzerinden erişebilirsiniz.
- **Pencerelerle Sınırlı Olanlar:** Bu okuyucular Windows işletim sistemine özeldir.
- **Mac, PDA veya UNIX ile Sınırlı Olanlar:** Bu okuyucular Mac, PDA veya UNIX işletim sistemlerine özeldir.

Bazı Popüler RSS Okuyucular:

- **QuietRSS:** Açık kaynaklı, platformlar arası bir RSS/Atom haber akışı okuyucusu
- **FeedReader:** Çok sayıda feed'i kolayca işleyen basit, anlaşılır bir feed okuyucu

İpucu: Çoğu tarayıcıda yerleşik bir RSS Okuyucu bulunur. RSS beslemeleri sunan bir web sitesine giderseniz, adres çubuğunda veya araç çubuğunda bir RSS simgesi göreceksiniz. Farklı yayınların listesini görüntülemek için simgeye tıklayın. Okumak istediğiniz feed'i seçin.

<channel> Öğesinin alt öğeleri

Öğe	Açıklama	Zorunlu mu?
<category>	RSS beslemesindeki içeriklerin kategorilerini (konularını) belirtir. Örneğin; haber, spor, teknoloji gibi	Hayır
<cloud>	RSS beslemesinde güncelleme olduğunda haberdar edilmek için kullanılabilecek işlemleri tanımlar	Hayır
<copyright>	Telif hakkı alınmış materyaller hakkında bilgi verir	Hayır
<description>	RSS beslemesini kısaca açıklar	Evet
<docs>	RSS beslemesinde kullanılan formatın dokümantasyonuna giden bir bağlantı URL'si belirtir	Hayır
<generator>	RSS beslemesini oluşturmak için kullanılan programı belirtir	Hayır
<image>	Toplayıcılar bir RSS beslemesi gösterirken görüntülenebilecek bir resim sağlar.	Hayır
<language>	RSS beslemesinin yazıldığı dili belirtir	Hayır
<lastBuildDate>	RSS beslemesindeki içeriğin en son değiştirilme tarihini ve saatini tanımlar	Hayır
<link>	RSS beslemesine bağlantı adresini (URL) tanımlar	Evet
<managingEditor>	RSS beslemesindeki içeriklerin editörü ile iletişime geçmek için e-posta adresini belirtir	Hayır
<pubDate>	RSS beslemesindeki içeriklerin en son yayınlanma tarihini ve saatini tanımlar	Hayır
<rating>	RSS beslemesinin PICS (Platform for Internet Content Selection) derecelendirmesini belirtir	Hayır
<skipDays>	Toplayıcıların RSS beslemesini hangi günlerde güncellemeyi atlamasını istediğinizi belirtir	Hayır
<skipHours>	Toplayıcıların RSS beslemesini hangi saatlerde güncellemeyi atlamasını istediğinizi belirtir	Hayır
<textInput>	RSS beslemesi ile birlikte görüntülenmesi gereken bir metin giriş alanı belirtir	Hayır

<title>	RSS beslemesinin başlığını tanımlar	Evet
<ttl>	RSS beslemesinin kaynak yenilenmeden önce önbellekte tutulabileceği dakika sayısını belirtir	Hayır
<webMaster>	RSS beslemesinin web yöneticisi ile iletişime geçmek için e-posta adresini belirtir	Hayır

<item> Öğesinin alt öğeleri

Element	Açıklama
<author>	İsteğe bağlıdır. Öğenin yazarının e-posta adresini belirtir.
<category>	İsteğe bağlıdır. Öğenin ait olduğu bir veya daha fazla kategoriyi tanımlar.
<comments>	İsteğe bağlıdır. Öğenin, o öğe hakkındaki yorumlara bağlantı kurmasını sağlar.
<description>	Zorunludur. Öğeyi açıklar.
<enclosure>	İsteğe bağlıdır. Öğeye bir medya dosyası eklenmesine izin verir.
<guid>	İsteğe bağlıdır. Öğe için benzersiz bir tanımlayıcı tanımlar.
<link>	Zorunludur. Öğeye giden hiperlinki tanımlar.
<pubDate>	İsteğe bağlıdır. Öğe için son yayınlanma tarihini tanımlar.
<source>	İsteğe bağlıdır. Öğe için üçüncü taraf bir kaynak belirtir.
<title>	Zorunludur. Öğenin başlığını tanımlar.

BÖLÜM ÖZETİ

Web servisleri, farklı platformlarda ve programlama dillerinde yazılmış uygulamalar arasında veri alışverişini kolaylaştırmak için kullanılan X emel tabanlı bir teknolojidir. Bu servisler, farklı sistemler arasında iletişim kurmayı sağlar ve veri aktarımını otomatikleştirir, hataları en aza indirir. Temel olarak, X emel Web Servisleri, müşterilerin isteklerini işleyen ve uygun yanıtları döndüren bir yapıya sahiptir. X emel Web Servisleri'nin kullanımıyla, örneğin bir kitap satıcısı olarak, müşterilerinizin stoklarınızdaki kitapları sorgulaması ve sipariş vermesi mümkün hÂle gelir. Bu sayede, farklı platformlarda ve dillerde yazılmış uygulamaları kullanan müşterilere kolayca hizmet verebilirsiniz. Örneğin, müşterilerin bir istekte bulunması durumunda, sistem X emel formatında bir cevap gönderir ve bu cevapta kitapların isimleri, yazarları ve fiyatları gibi bilgiler yer alır. X emel Web Servislerinin kullanımı için X emel Web Servis Tanım Dili (WSDL) önemlidir. WSDL, bir web servisinin işlevselliğini ve erişim yöntemlerini tanımlayan bir standarttır. Örneğin, bir kitap listesi alma servisinin WSDL belgesi, servisin adını, işlemlerini ve erişim protokollerini belirtir. X emel SOAP (Simple Object Access Protocol), internet üzerindeki uygulamalar arasında bilgi alışverişi yapmak için kullanılan bir protokoldür. SOAP, X emel tabanlı bir protokol olup, istemci-sunucu mimarisini kullanır. Müşteriler, SOAP mesajları aracılığıyla sunuculara istek gönderir ve sunucular da bu isteklere yanıt verir. SOAP mesajları, zarf ve gövde olmak üzere iki bölümden oluşur ve http veya https gibi protokoller aracılığıyla taşınır. X emel SOAP'un avantajları arasında platform bağımsızlığı, standart protokol olması ve güvenlik sağlaması bulunurken, karmaşıklık, performans ve veri boyutu gibi dezavantajlar da vardır. Bu nedenle, X emel Web Servisleri ve X emel SOAP'un kullanımı, uygulamanın gereksinimleri ve ihtiyaçlarına bağlı olarak değerlendirilmelidir. RDF (Resource Description Framework), web üzerindeki kaynakları tanımlamak için kullanılan ve bilgisayarlar tarafından işlenebilir veri olarak kullanılan bir standarttır. RDF, W3C'nin Anlamsal Web Etkinliğinin bir parçasıdır ve 2004 yılında kabul edilmiştir. RDF'nin kullanım alanları arasında alışveriş öğelerinin özelliklerini açıklama, web etkinliklerinin zaman çizelgelerini belirtme, web sayfalarına ilişkin bilgilerin açıklanması gibi örnekler bulunmaktadır. RDF'nin temel bileşenleri kaynaklar, özellikler ve değerlerdir. RDF, genellikle Turtle veya RDF/X emel gibi formatlarda ifade edilir ve veri esnekliği, bağlantılı veri oluşturma ve makine okunabilirlik gibi avantajlara sahiptir. Ancak, karmaşıklık, işleme maliyeti ve yetersiz kullanım gibi dezavantajları da vardır. RDF öğeleri arasında RDF Description elemanı, kaynakları ve ilişkili bilgileri temsil etmek için kullanılır. Bu elemanla bilgi kaynaklarını tanımlama, özellikleri belirtme ve ilişkileri gösterme gibi işlemler yapılabilir. RDF'de konteynerler ve koleksiyonlar da bilgi nesnelerini organize etmek ve ilişkilerini göstermek için kullanılır. RDF Şeması (RDFS), RDF'de kullanılan meta veri dilidir ve temel ontoloji kavramlarını tanımlar. Sınıflar, özellikler, alt sınıflar ve özellik kısıtlamaları gibi kavramlar RDFS içinde tanımlanır. Bu kavramlar, belirli bir alan veya uygulama için bilgi nesnelerini ve ilişkilerini tanımlamak için kullanılan uygulama sınıfları tarafından kullanılabilir. RSS (Really Simple Syndication), web sitelerinin güncel içeriklerini takip etmek için kullanılan bir sistemdir.

RSS, web sitelerinin sık güncellenen içeriklerini kullanıcıların RSS okuyucuları aracılığıyla takip etmelerine olanak tanır. Bu sayede kullanıcılar, siteleri ziyaret etmeden yeni içeriklerden haberdar olabilirler. RSS belgeleri, bir web sitesinin güncel içeriğini içeren ve ve gibi öğeleri içeren X emel belgeleridir. öğesi, RSS beslemesinin başlığını, bağlantısını ve açıklamasını içerirken, öğesi, bir makalenin veya içeriğin başlığını, bağlantısını, özetini ve yayınlanma tarihini içerir. RSS'in avantajları arasında zaman tasarrufu, güncel kalma ve bilgi düzenleme bulunurken, dezavantajları arasında teknik gereklilik, bakım zorluğu ve bilgi kirliliği yer alır. RSS, haber siteleri, şirketler, takvimler ve bloglar gibi web siteleri için faydalı bir araçtır.

GÖZDEN GEÇİRELİM

1. XML Web Servisleri, hangi amaçla kullanılır?
A) Müşteri hizmetleri için
B) Veri alışverişi ve iletişim için
C) Oyun geliştirme için
D) Reklam amaçlı kullanım için
E) Haber alma amaçlı kullanım için

2. XML Web Servisleri, farklı uygulamalar arasında hangi veri alışverişini sağlar?
A) Ses dosyaları
B) Görüntü dosyaları
C) Metin dosyaları
D) XML formatında veri
E) Video dosyaları

3. SOAP, hangi amaçla kullanılır?
A) XML dökümanları oluşturmak için
B) İşletim sistemi oluşturmak için
C) Görsel ögeleri oluşturmak için
D) Oyun geliştirmek için
E) Veri transferi için XML tabanlı bir protokol

4. WSDL nedir?
A) Web Sitesi Dizini
B) Web Servis Dökümanı
C) Yazılım Geliştirme Dili
D) Web Servis Tanım Dili
E) XML Tanım Dili

5. RDF Şeması (RDFS) ne için kullanılır?
A) Bir web sayfasının tasarımını oluşturmak için
B) Bir veritabanı şemasını tasarlamak için
C) RDF'de bilgi nesneleri ve ilişkileri tanımlamak için
D) Bir sunucu ağını kurmak için
E) Bir dosya sistemini organize etmek için

6. RSS nasıl çalışır?
A) Web siteleri RSS beslemelerini oluşturur ve kullanıcılar RSS okuyucu yazılımlarıyla bu beslemeleri takip eder.
B) Kullanıcılar RSS beslemelerini oluşturur ve web siteleri bu beslemeleri kullanarak içeriklerini günceller.
C) RSS beslemeleri, web sitelerinin içeriklerini otomatik olarak günceller.
D) RSS beslemeleri, web sitelerinin görünümünü özelleştirmek için kullanılır.
E) RSS beslemeleri, web sitelerinin güvenliğini artırmak için kullanılır.

7. RSS'de hangi versiyonlar kullanılır?

A) RSS 0.91, RSS 1.0 ve RSS 2.0
B) RSS 2.0, RSS 3.0 ve RSS 4.0
C) RSS 1.0, RSS 2.0 ve RSS 3.0
D) RSS 0.91, RSS 2.0 ve RSS 3.2
E) RSS 0.91, RSS 1.0 ve RSS 2.0.1

Cevap Anahtarı

1	2	3	4	5	6	7
B	D	E	D	C	A	A

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

XML – Web Services (n.d.). https://www.w3schools.com/xml/xml_services.asp
XML – WSDL (n.d.). https://www.w3schools.com/xml/xml_wsdl.asp
XML – SOAP (n.d.). https://www.w3schools.com/xml/xml_soap.asp
XML – RDF (n.d.). https://www.w3schools.com/xml/xml_rdf.asp
XML – RSS (n.d.). https://www.w3schools.com/xml/xml_rss.asp