WEB TASARIM I

Öğr. Gör. M. Mutlu YAPICI

Ankara Üniversitesi
Elmadağ Meslek Yüksekokulu

Öğr. Gör. M. Mutlu YAPICI

Ders İzlencesi

Hafta	Modüller/İçerik/Konular
1. Hafta	PHP Tanımı ve Sunucu Kurulumları
2. Hafta	PHP Yazım Notasyonu ve Değişkenler
3. Hafta	PHP de Karar kontrol yapıları ve Döngüler
4. Hafta	Dizi ve Dizi işlemleri
5. Hafta	Fonksiyon, Sınıf ve Nesne Kavramları
6. Hafta	HTML ve PHP
7. Hafta	PHP ile Veritabanı İşlemleri
8. Hafta	MYSQL, MYSQLİ
9. Hafta	AJAX
10. Hafta	
11. Hafta	
12. Hafta	
13. Hafta	
14. Hafta	

Bu Ünitede Ele Alınan Konular

- PHP dilinin genel yapısı
- PHP de fonksiyonlar
- Kullanıcı Tanımlı Fonksiyonlar
- Dönüş değerine sahip Fonksiyonlar
- Varsayılan Parametreye Sahip Fonksiyonlar
- Referansa Bağlı Çağırılan Fonksiyonlar
- Öz Yinelemeli (Recursive) Fonksiyonlar
- Parametre Sayısı Değişen Fonksiyonlar
- Dışarıdaki Bir Dosyadan Fonksiyon Çağırmak

Ders Kazanımları

Bu bölümü Bitirdiğimizde,

- PHP dilinin genel yapısı
- PHP de fonksiyonlar
- Kullanıcı Tanımlı Fonksiyonlar
- Dönüş değerine sahip Fonksiyonlar
- Varsayılan Parametreye Sahip Fonksiyonlar
- Referansa Bağlı Çağırılan Fonksiyonlar
- Öz Yinelemeli (Recursive) Fonksiyonlar
- Parametre Sayısı Değişen Fonksiyonlar
- Dışarıdaki Bir Dosyadan Fonksiyon Çağırmak

öğrenmiş olacaksınız.

Fonksiyonlar

Profesyonel programlamada Brute Force dediğimiz tek düze programlama yetersiz kalmaktadır. Geliştirdiğimiz yazılımların daha kolay analiz edilmesi ve modüler bir yapıya kavuşabilmesi ve hatta en önemlisi Nesnel Programlamaya (OOP) uygun olabilmesi için fonksiyon oluşturulması büyük önem taşımaktadır.

Fonksiyonlar sayesinde yapacağımız programları görevlerine göre parçalara ayırıp her bir göreve bir fonksiyon tanımlayarak aynı göreve başka yerde ihtiyacımız olduğunda tekrar tekrar aynı kodları yazmaktan kurtuluruz.

Fonksiyonların programlamadaki önemini sayfalarca anlatsak azdır. Daha fazla bilgiyi çeşitli kaynaklardan edinebilirsiniz.

Fonksiyonlar

PHP de Fonksiyon tanımlama işlemi function anahtar kelimesi ile yapılmaktadır.

Örnek :

```
function Fonksiyon Adı () {

.....................
        Yapılacak İşlemler.
.....................
}
```

Fonksiyonlar

PHP de Fonksiyon tanımlama işlemi function anahtar kelimesi ile yapılmaktadır.

Örnek :

```
function merhabaYaz () {
        Echo 'Merhaba';
}

merhabaYaz();//Fonksiyonu ismi ile çağırıyoruz.
```

Fonksiyonlar

Fonksiyonlara dışarıdan değer gönderebilmek için fonksiyon argümanları kullanılmaktadır. Fonksiyon parantezleri arasına tanımlanan değişkenlere fonksiyon argümanı yada parametre denir. Bu değişkenler sayesinde fonksiyonlara dışarıdan bilgi yüklemesi yapılır.

Örnek :

```
function merhabaYaz ($adi) {
    Echo 'Merhaba '.$adi;
}

merhabaYaz('Mutlu');//Fonksiyonu ismi ile çağırıyoruz.
```

Fonksiyonlar

Fonksiyonlara dışarıdan değer gönderebilmek için fonksiyon argümanları kullandığımızı söyledik peki ama fonksiyon içerisinden dışarıya nasıl veri göndereceğiz. Bu işlem için de fonksiyonların değer dönderme özelliklerini kullanacağız. Fonksiyonlar return anahtar kelimesi ile çağırıldıkları yere sonuç dönderirler.

Örnek :

```
function ToplamaIslemi ($sayi1, $sayi2) {
    return $sayi1 + $sayi2;
}

Echo ToplamaIslemi(5, 10);//toplama işlemi yapıyor.
```

Fonksiyonlar

Örnek; $sayi ve $islemTuru diye iki argümanı bulunan bir fonksiyon tanımlayınız. Bu fonksiyonun $islemTuru değeri ‘tek’ ise 1 den itibaren $sayi ya kadar olan tek sayıları, ‘çift’ ise 1 den itibaren $sayi ya kadar olan çift sayıları, ‘tümü’ ise 1 den itibaren $sayi ya kadar olan tüm sayıları, toplasın.

Fonksiyonlar

Örnek; Bir önceki sorudan $islemTurunu çıkartın. $sayı ya kadar olan tek , çift ve tum toplamları bulup bunları dizi şeklinde geri dönderin.

Varsayılan Parametreli Fonksiyonlar

Fonksiyonları çağırırken eğer bir parametresine değer göndermezseniz hata mesajı ile karşılaşırsınız. Eğer bazı parametrelere her zaman değer göndermeyelim onların varsayılan bir değeri olsun göndermediğimizde o değer geçerli olsun diyorsanız. Default yani varsayılan değerlerini tanımlamalısınız.

Örnek :

```
function ToplamaIslemi ($sayi1, $sayi2) {
    return $sayi1 + $sayi2;
}

Echo ToplamaIslemi(10);//parametre eksi hatası alırsınız.
```

Örnek :

```
function ToplamaIslemi ($sayi1, $sayi2=5) {
    return $sayi1 + $sayi2;
}

Echo ToplamaIslemi(10);//işlem başarılı çünkü ikinci varsayılan değere sahip.
```

Varsayılan Parametreli Fonksiyonlar

Faktöriyel hesabı yapan bir fonksiyon oluşturun. Eğer fonksiyona değer gönderilmiyorsa 10 faktöriyeli hesaplasın. Gönderiliyorsa gönderilen sayının faktöriyelini hesaplasın.

Referansla Çağrılan Fonksiyonlar

Şimdiye kadar kullandığımız fonksiyonları değer gönderme ve değeri geri alma şeklinde tanımladık. Bu tip fonksiyonlarda tanımlanan her değişken aynı adla bile olsa yeni bir değişkeni ifade eder. Daha önceki konularda değişken tanımlarken değişkenleri referansları ile tanımlayıp birbirlerine bağlamıştık. Fonksiyonlara gönderdiğimiz parametre değişkenleri de aynı şekilde referansla birbirlerine bağlayarak return işlemine gerek kalmadan değer alabiliriz.

Örnek :

```
function Karesi (&$sayi) {
    $sayi *=$sayi;
}
$a=6;
ToplamaIslemi($a);
Echo $a;//a değeri 6 değil artık 36 dır çünkü $sayi anın adresine işlem sonucunu yükledi. Böylece retun e gerek kalmadan sonucu alabildik
```

Parametre Sayısı Değişen Fonksiyonlar

Bazen fonksiyona göndereceğimiz parametre sayıları değişebilir ve biz başlangıçta kaç adet veri göndereceğimizi bilmeyebiliriz. Bu gibi durumlarda parametre sayısı değişen fonksiyon mantığını kullanabiliriz.

Bu fonksiyon türündeki ana mantı, öncelikle fonksiyon içerisinden fonksiyona o anda kaç parametre geldiğini öğrenmeliyiz ve bunun için func_num_args(); fonksiyonunu kullanacağız. Daha sonra her parametre değerini döngü ile sırasıyla alıp işleme katacağız. Parametreler gelirken sanki bir dizi elemanıymış gibi geliyor yani 0. indisten başlayarak sıralı bir şekilde geliyor. İndis numarası ile sıradaki elemana ulaşabilmek için de func_get_args(); fonksiyonunu kullanacağız.

Örnek :

```
function carpim(){
        $carpim=1;
        $par_say= func_num_args();///gelen parametre sayısını aldık
        for($i=0;$i<$par_say;$i++)
        {
                $carpim *=func_get_arg($i);//$i . sıradaki parametreyi alıp işleme katıyoruz
        }

        return $carpim;

}

echo carpim(2,5,7,8,10);//5600
echo carpim(22,5);//110
```

Recursive Fonksiyonlar

Bazı işlemler aslında hep kendini tekrar eder sadece işleme gelen parametre değerleri değişir. Bu gibi durumlarda işlemi sürekli çağırmak gerekir. İşte bu gibi durumlarda kendi kendini çağıran fonksiyonlar kullanılır. Bir çok sıralama algoritmasında ve gelişmiş algoritmalarda öz yinelemeli yani recursive fonksiyonlar çok sık kullanılır. Örneğin faktöriyel hesaplamalarında aslında 1 e kadar olan sayılar hep toplamcarpimla çarpılır yani yapılan işlem hep değeri bir azaltıp çarpmadır. Bu faktöriyel hesaplama işlemine recursive fonksiyonlar çok uygundur.

Örnek :

```
function faktoriyel($sayi){
        if($sayi<2)
        {
            return 1;//eğer gelen değer 2 den küçükse 1 döndürelim
        }

        return  $sayi * faktoriyel($sayi-1); ///1 den büyükse şimdiki değerle çarpılmak üzere değerin 1 eksiğini tekrar fonksiyona gönderelim. Yani fonksiyonu fonksiyon içinden tekrar çağıralım
}
echo faktoriyel(5);//120
```

Farklı Dosyadan Çağırılan Fonksiyonlar

Programlamada tek bir dosyaya tüm kodları yazmak, tüm fonksiyonları oluşturmak kod karmaşasına ve düzensizliğe sebep olur ve kodları yorumlamak analiz etmek zorlaşır. Bu gibi durumlarla karşılaşmamak için aslında biz programcılar her kodu her fonksiyonu ilgili dosyaya kaydedip gerektiğinde o dosyayı kullanacağımız yere import ederek yani yükleyerek kullanırız. Örneğin veritabanı işlemleri ile ilgili fonksiyonlar bir dosyada, kullanıcı işlemleri ile ilgili fonksiyonlar başka bir dosyada , oturum işlemleri ile ilgili fonksiyonlar başka bir dosyada tanımlanarak daha düzenli bir yapı oluşturulur. İşte bu gibi durumlarda kullanılacak fonksiyona ait dosya gereklidir ve kullanacağımız yere yüklenmelidir. Yükleme işlemi için PHP de require(dosya adi); fonksiyonu kullanılır.

Örnek : bu kod faktoriyel.php dosyasında olsun

```
function faktoriyel($sayi){
        if($sayi<2)
        {
            return 1;//eğer gelen değer 2 den küçükse 1 döndürelim
        }

        return   $sayi * faktoriyel($sayi-1); ///1 den büyükse şimdiki değerle çarpılmak üzere değerin 1 eksiğini tekrar fonksiyona gönderelim. Yani fonksiyonu fonksiyon içinden tekrar çağıralım
}
```

Ve şimdi biz onu anasayfa.php den kullanmak isteyelim. Öncelikle faktöriyel.php yi yüklemeliyiz.

```
require('faktöriyel.php'); // faktöriyel.php dosyası yükleniyor.
echo faktoriyel(5);//120    fonksiyon kullanılıyor.
```

KAYNAKLAR

- İnternet ortamı
- PHP ve AJAX Haydar TUNA
- A'dan Z'ye PHP Rıza ÇELİK