Php 5 ile Filtreleme İşlemleri

Yazan: altayalp (izzet ögetürk)
E-Posta: altayalp [a-t] gmail.com

Belgede eksik ve yanlış gördüğünüz yerler için ya da görüşlerinizi belirtmek için bana e-posta adresimden ulaşabilir ve belgeyi istediğiniz platformlarda dağıtabilirsiniz.

Php 5 Filter Kütüphanesi

Php'nin filter eklentisi, 5.2.0 sürümü ile birlikte gelen kullanışlı bir filtreleme kütüphanesi sunar. Bu eklenti ile filtreleme işlemleri daha kolay ve pratik yoldan yapılabilir. **filter_var()** fonksiyonu Php'nin filter eklentisini kullanarak filtreleme işlemlerini yapmamızı sağlar. filter_var() fonksiyonu veri doğrulanırsa verinin kendisini, doğrulanamazsa false döndürür.

Filtreleme işlemi iki ana tipte gerçekleşir: Doğrulama ve temizleme. (validation ve sanitization)

Doğrulama (validation): Verinin belirli bir kıstasa uygun olup olmadığını kontrol eder. Veriler üzerinde herhangi bir değişiklik yapmaz.

Temizleme (sanitization): Veriyi temizler, istenmeyen karakterleri yok ederek veriyi uygun kritere sokar. Yapılan işleme göre bazı bölümlerde "ayıklama" kelimesini de kullandığımı belirtmek istiyorum.

Bayraklar (flags) isteğe bağlı olarak filtreleme işleminin davranışını değiştirmek için kullanılabilirler. Eğer yapılan işlem url filtrelemek ise FILTER_FLAG_PATH_REQUIRED eklenerek url adresinde bir path geçmesi gerektiği belirtilebilir. (Örnek: http:altayalp.com**/php**)

Filtreler İçin Doğrulama Sabitleri Listesi

FILTER_VALIDATE_BOOLEAN : Boolean veri tipini kontrol etmek için kullanılır. '1', 'true', 'on' ve 'yes' için true döndürür, diğerleri için false döndürür. Eğer FILTER_NULL_ON_FAILURE bayrak olarak geçirilirse sadece '0', 'off', 'no', '' için false ve boolean olmayan tüm değerler için NULL döner. Yukarıda filter_var() fonksiyonunun veri doğrulanırsa verinin kendisini döndüreceğini belirtmiştim, bu istisna dışında öyledir. FILTER_VALIDATE_BOOLEAN yaptığı iş gereği veri doğrulanırsa true döndürür. Fakat bunun istisna olduğunu ve diğer işlemlerde veri doğrulandığı takdirde verinin kendisinin döneceğini hatırlatırım.

FILTER_VALIDATE_EMAIL : E-posta adresi doğrulamak için kullanılır.

FILTER_VALIDATE_FLOAT : Float tibi değerleri doğrular.

FILTER_VALIDATE_INT : Int tipi değerleri doğrular.

FILTER_VALIDATE_IP : Ip adresi doğrulamak için kullanılır.

FILTER_VALIDATE_REGEXP : Perl uyumlu düzenli ifade deseninizle doğrulama yapmanızı sağlar.

FILTER_VALIDATE_URL : Url adresleri doğrulamak için kullanılır.

Daha detaylı bilgi için kılavuz sayfasına bakmanızı öneririm http://tr.php.net/manual/tr/filter.filters.validate.php

Filtreler İçin Temizleme Sabitleri Listesi

FILTER_SANITIZE_EMAIL : Rakamlar, harfler ve !#$%&'*+-/=?^_`{|}~@.[]. dışındaki tüm karakterleri temizler.

FILTER_SANITIZE_ENCODED : Bayrak geçirilmezse hemen, hemen urlencode() fonksiyonu ile aynı işi yapar. Url adresini encode eder.

FILTER_SANITIZE_MAGIC_QUOTES : Verilere addslashes() uygular.

FILTER_SANITIZE_NUMBER_FLOAT : Sayılar +- ve isteğe bağlı olarak .,eE dışındaki karakterleri temizler.

FILTER_SANITIZE_NUMBER_INT : Rakam + ve - dışında tüm karakterleri temizler.

FILTER_SANITIZE_SPECIAL_CHARS : Html etiketlerini ascii 32 karakterlerle encode eder ve isteğe bağlı olarak özel karakterleri temizler ya da encode eder.

FILTER_SANITIZE_STRING : Etiketleri temizler ve isteğe bağlı olarak özel karakterleri encode eder.

FILTER_SANITIZE_URL : Rakamlar, harfler ve $-_.+!*'(),{}|\\^~[]`<>#%";/?:@&=. dışındaki tüm karakterleri kaldırır.

FILTER_UNSAFE_RAW : Bayrak geçirilmezse tek başına hiçbir şey yapmaz. İsteğe bağlı olarak özel karakterleri siler ya da encode eder.

Daha fazla bilgi için kılavuza bakmanızı öneririm http://tr.php.net/manual/tr/filter.filters.sanitize.php .

Ayrıca **FILTER_CALLBACK** ile kendi filtreleme fonksiyonunuzu çağırabiliriz.

filter_var() Fonksiyonu

filter_var() fonksiyonunun prototipi şu şekildedir:

mixed filter_var (mixed $variable [, int $filter = FILTER_DEFAULT [, mixed $options]])

İkinci parametre yani uygulanacak filtre isteğe bağlı olsa da kullanmak gerekir zira kullanılmadığında veriye hiçbir işlem yapılmaz. Bu parametre int olarak da verilebilir. Fakat biz işlemin int id'si yerine id değerine karşılık gelen sabitleri kullanacağız. Örnek olarak 258 ile FILTER_VALIDATE_BOOLEAN aynıdır ve aynı işlemi yapar.

Options Eklemek

Bazı doğrulama ve temizleme işlemlerine ek olarak bir “options” parametresi eklenebilir. Options'ları eklerken diziler kullanılır.

```
<?php

$veri = '';

$regex = '';

filter_var($veri, FILTER_VALIDATE_REGEXP,
array('options'=>array('regexp'=>$regex)));

?>
```

Yukarıdaki kod options eklemenin basit bir örneğidir. Bu örnek sadece options ekleme işlemini göstermek için yazıldığından bu şekilde kullanılınca Php bir hata mesajı verecektir. Burada bizi ilgilendiren 3. parametredir. 3. Parametre olarak bir dizi tanımladık 'options' anahtarına değer olarak yine bir dizi atadık, atadığımız dizide ilk anahtar seçeneği belirliyor ve bu anahtar seçeneğin kullanacağı değeri alıyor. Benzer örnekleri ilerleyen sayfalarda göreceksiniz.

Filter Kütüphanesi ile Doğrulama İşlemleri

Boolean Doğrulama:

```
<pre>
<?php

var_dump(filter_var('true', FILTER_VALIDATE_BOOLEAN));

var_dump(filter_var('elma', FILTER_VALIDATE_BOOLEAN, FILTER_NULL_ON_FAILURE)
);

?>
</pre>
```

Örneğin ikinci satırı gördüğünüz gibi NULL döndürdü. Yukarıda açıkladığımız gibi eğer FILTER_NULL_ON_FAILURE bayrak olarak geçirilmeseydi false dönecekti.

E-posta Adresi Doğrulama:

```
<?php

$eposta1 = 'altayalp@mailsite.com';
$eposta2 = 'altayalp[at]mailsite.com';
$eposta3 = 'altayalp.site.com';

echo filter_var($eposta1, FILTER_VALIDATE_EMAIL) ? "$eposta1 doğru<br
/>" : "$eposta1 yanlış<br />";

filter_var($eposta2, FILTER_VALIDATE_EMAIL) ? print "$eposta2 doğru" :
print "$eposta2 yanlış<br />";

filter_var($eposta3, FILTER_VALIDATE_EMAIL) ? print "$eposta3 doğru" :
print "$eposta3 yanlış<br />";

var_dump(filter_var($eposta2, FILTER_VALIDATE_EMAIL));

?>
```

Yukarıdaki kod

```
altayalp@mailsite.com doğru
altayalp[at]mailsite.com yanlış
altayalp.site.com yanlış
bool(false)
```

Şeklinde çıktı verir. Gördüğünüz gibi e-posta adresi doğrulamak kolay ve zahmetsiz. Sizin de herkes gibi kendinize ait bir regex deseniniz olabilir fakat Php, filter kütüphanesi ile bu işi kısmen standartlaştırmış oluyor.

Float Doğrulama:

```
<?php

$sayi = 12.43;

if(filter_var($sayi, FILTER_VALIDATE_FLOAT) === false) {
    echo 'veri float tipinde değil';
} else {
    echo $sayi;
}
?>
```

Kodun çıktısı "12.43" olacaktır. Eğer veri float tipinde olmasaydı "veri float tipinde değil" şeklinde çıktı verecekti. Float doğrularken eğer . (nokta) yerine , (virgül) kullanırsanız false dönecektir. Bunu aşmak için options ekleyebiliriz.

```
<?php

$sayi = '12,43';

if(filter_var($sayi, FILTER_VALIDATE_FLOAT,
array('options'=>array('decimal'=>','))) === false) {
    echo 'veri float tipinde değil';
} else {
    echo $sayi;
}
?>
```

Yukarıdaki kodun çıktısı "12,43" olacaktır. Decimal options ile . (nokta) yerine , (virgül) kullanabiliyoruz.

"options" eklemeyi yukarıda göstermiştik. Bu örnek "options" ekleme işlemine somut bir örnek oldu.

Integer Doğrulama:

```
<?php

$sayi1 = '1234';

$sayi2 = 4321;

$sayi3 = 1.4;

filter_var($sayi1, FILTER_VALIDATE_INT) ? print "$sayi1 int<br />" :
print "$sayi1 int değil";

filter_var($sayi2, FILTER_VALIDATE_INT) ? print "$sayi2 int<br />" :
print "$sayi2 int değil";

filter_var($sayi3, FILTER_VALIDATE_INT) ? print "$sayi3 int" :
print "$sayi3 int değil";

?>
```

Yukarıdaki kodun çıktısı:

1234 int
4321 int
1.4 int değil

Şeklinde olacaktır. $sayi3 integer olmadığından false döndürür.

Int doğrularken "options" ekleyerek en küçük ve en büyük sayıyı belirleyebiliriz.

```
<?php

$enaz = 5;

$encok = 10;

$sayi = 15;

if(filter_var($sayi, FILTER_VALIDATE_INT, array('options' =>
array('min_range' => $enaz, 'max_range'=> $encok ))) === false) {
    echo 'Girdiğiniz sayı 5 ile 10 arasında olmalıdır';
} else {
    echo 'Doğru sayı girdiniz';
}

?>
```

min_renge ve max_renge ile girilecek en küçük ve en büyük sayıyı belirledik. Bu şekilde 5 ve 10 doğru olarak kabul edilir 5'den küçük ve 10'dan büyük sayılar false döndürür.

Ip Adresi Doğrulama:

```
<?php

$ip1 = '127.0.0.1';

if(filter_var($ip1, FILTER_VALIDATE_IP) === false) {
    echo 'Ip adresi yanlış';
} else {
    echo 'Ip adresi doğru';
}

?>
```

FILTER_VALIDATE_IP, 4 adet isteğe bağlı bayrak alabilir ve bu şekilde doğrulama yapabilir. Bu bayraklar şunlardır:

FILTER_FLAG_IPV4
FILTER_FLAG_IPV6
FILTER_FLAG_NO_PRIV_RANGE
FILTER_FLAG_NO_RES_RANGE

FILTER_FLAG_IPV6 bayrağı ile ipv6 ip adresi kontrol edelim.

```
<?php

$ip2 = '2001:cdba:0000:0000:0000:0000:3257:9652';

if(filter_var($ip2, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6) === false) {
    echo 'Ip adresi yanlış';
} else {
    echo 'Ip adresi doğru';
}

?>
```

Ip adresi geçerli bir ipv6 ip adresi olduğundan kod “Ip adresi doğru” şeklinde çıktı verir.

Regex (Düzenli ifadeler) ile Doğrulama Yapmak:

```
<?php

$string = 'Ekmek aldım eve gidiyorum.';

if(filter_var($string, FILTER_VALIDATE_REGEXP,
array('options'=>array('regexp'=>'/^Ekmek(.*)/'))) === false) {
    echo 'Cümleniz ekmek ile başlamalıdır.';
} else {
    echo 'Doğru kelime';
}

?>
```

Düzenli ifade kullanarak doğrulama yapmak için “options” eklemek zorunludur. Burada “options” olarak regexp kullandık ve değer olarak basit bir düzenli ifade kullandık. Yukarıda $string değişkeninde depolanan cümle Ekmek ile başlamazsa false döndürür ve “Cümleniz ekmek ile başlamalıdır” şeklinde çıktı verir.

Regex ile e-posta adresi doğrulama:

```
<?php

$eposta = 'altayalp@mailsite.com';

if(filter_var($eposta, FILTER_VALIDATE_REGEXP,
array('options'=>array('regexp'=>'/^([a-z0-9_]|\\-|\\.)+@(([a-z0-9_]|\\-
)+\\.)+[a-z]{2,4}$/'))) === false) {
    echo 'e-posta adresi yanlış';
} else {
    echo 'e-posta adresi doğru';
}

?>
```

Url Doğrulama:

```php
<?php

$url = 'http://www.altayalp.com';

if(filter_var($url, FILTER_VALIDATE_URL) === false) {
    echo 'Url yanlış';
} else {
    echo 'Url doğru';
}

?>
```

Url doğru formatta olduğu için ekranda “Url doğru” yazacaktır.

FILTER_VALIDATE_URL isteğe bağlı 2 bayrak alabilir:

FILTER_FLAG_PATH_REQUIRED
FILTER_FLAG_QUERY_REQUIRED

```php
<?php

$url = 'http://www.altayalp.com/php';

if(filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_PATH_REQUIRED)
=== false) {
    echo 'Url\'ye bir yol girilmesi gerekir';
} else {
    echo 'Url doğru';
}

?>
```

Eklediğimiz FILTER_FLAG_PATH_REQUIRED bayrağı url'ye bir yol tanımlamak gerektiğini belirtir. Eğer url'nin sonunda / varsa da url'yi doğru olarak kabul eder. FILTER_FLAG_QUERY_REQUIRED bayrağı ise url'de bir sorgunun olmasını zorunlu kılar.

```php
<?php

$url = 'http://www.altayalp.com/index.php?s=';

if(filter_var($url, FILTER_VALIDATE_URL, FILTER_FLAG_QUERY_REQUIRED)
=== false) {
    echo 'Url\'ye bir sorgu eklemeniz gerekir';
} else {
    echo 'Url doğru';
}

?>
```

Url’de sorgu olduğu için doğru kabul edilecektir.

Filter Kütüphanesi ile Temizleme İşlemleri

E-posta Adresi Temizleme:

```
<?php

$string = 'altayalp()@/birsite.com';

echo filter_var($string, FILTER_SANITIZE_EMAIL);

?>
```

Kod dizge içindeki gereksiz harfleri temizledi ve geriye düzgün olarak altayalp@birsite.com e-posta adresini döndürdü. FILTER_SANITIZE_EMAIL dizge içindeki ()/ karakterlerini temizledi.

Url Adresi Encode Etme:

```
<?php
$url = 'http://www.altyalp.com/?s=aranan kelime';

echo filter_var($url,  FILTER_SANITIZE_ENCODED);
?>
```

Yukarıdaki kodun çıktısı http%3A%2F%2Fwww.altyalp.com%2F%3Fs%3Daranan%20kelime şeklinde olacaktır.

Sihirli Tırnaklar (Magic Quotes) ile addshashes() Uygulama:

```
<?php

$string = 'yazi "kelime" "elma"portakal"armut" \'karpuz\'';

echo filter_var($string, FILTER_SANITIZE_MAGIC_QUOTES);

echo '<br />' . addslashes($string);

?>
```

Yukarıdaki kodu çalıştırdığınızda iki satırında aynı olduğunu göreceksiniz. FILTER_SANITIZE_MAGIC_QUOTES addslashes() ile aynı işi yapmaktadır.

Float Ayıklama:

```
<?php

$float = 'xyz12.,34%r3';

echo filter_var($float, FILTER_SANITIZE_NUMBER_FLOAT);

?>
```

Yukarıdaki kodu çalıştırdığınızda çıktısının 12343 olduğunu görürsünüz. Dikkat ettiyseniz dizge içerisinde yer alan . (nokta) ve , (virgül) de silinmiştir. İsteğe bağlı olan FILTER_FLAG_ALLOW_FRACTION bayrağı ile noktanın silinmesini önleyebilirsiniz.

```
<?php

$float = 'xyz12.,34%r3e';

echo filter_var($float, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_FRAC
TION);

?>
```

Gördüğünüz gibi nokta silinmedi ve kodun çıktısı 12.343 oldu. Eğer nokta yerine virgül kullanmak isterseniz FILTER_FLAG_ALLOW_THOUSAND bayrağını kullanmalısınız.

```
<?php

$float = 'xyz12.,34%r3e';

echo filter_var($float, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_THOU
SAND);

?>
```

Yukarıdaki kod bir öncekinin aksine . (nokta) işaretini siler ve , (virgül) çıktıda görünür. Bu işlem için kullanabileceğimiz son bayrak da FILTER_FLAG_ALLOW_SCIENTIFIC bayrağıdır. Bu bayrak dizge içerisindeki e ve E harflerinin silinmesini engeller.

```
<?php

$float = 'xyz12.,34%r3e';

echo filter_var($float, FILTER_SANITIZE_NUMBER_FLOAT, FILTER_FLAG_ALLOW_SCIE
NTIFIC);

?>
```

Integer Ayıklama:

```
<?php

$sayi = 'x-yz12.,34%r3+';

echo filter_var($sayi, FILTER_SANITIZE_NUMBER_INT);

?>
```

Gördüğünüz gibi FILTER_SANITIZE_NUMBER_INT + - ve rakam dışındaki karakterleri temizledi.

Html ve Özel Karakterleri Temizleme:

```
<?php

$string = '<script>alert("javascript");</script> ŞıİçöüĞ';

echo filter_var($string, FILTER_SANITIZE_SPECIAL_CHARS);

?>
```

Yukarıdaki kodu çalıştırdığınızda ekranda **<script>alert("javascript");</script> ŞıİçöüĞ** yazdığını görürsünüz. Tarayıcınızdan sayfanın kaynağına baktığınızda ise **<script>alert("javascript");</script> ŞıİçöüĞ** şeklinde Html etiketlerinin encode edildiğini göreceksiniz. Fakat Türkçe karakterlere dokunulmamıştır. Silme ya da encode etme yoktur.

Bu işlem için kullanabileceğimiz 3 bayrak bulunur. Bu bayraklar:

FILTER_FLAG_STRIP_LOW
FILTER_FLAG_STRIP_HIGH
FILTER_FLAG_ENCODE_HIGH

İsimlerinden ne iş yaptıkları az çok anlaşılıyor. Örneğin FILTER_FLAG_STRIP_HIGH bayrağını kullanırsanız dizgede bulunan Türkçe karakterlerin silindiğini görecekseniz.

```
<?php

$string = '<script>alert("javascript");</script> ŞıİçöüĞ';

echo filter_var($string, FILTER_SANITIZE_SPECIAL_CHARS, FILTER_FLAG_STRIP_HIGH);

?>
```

Eğer Türkçe karakterleri silmeden encode etmek istiyorsanız FILTER_FLAG_ENCODE_HIGH bayrağını kullanmalısınız. Bu bayrak Türkçe karakterleri silmek yerine Ascii 32 olarak kodlayacaktır.

```
<?php

$string = '<script>alert("javascript");</script> ŞıİçöüĞ';

echo filter_var($string, FILTER_SANITIZE_SPECIAL_CHARS, FILTER_FLAG_ENCODE_HIGH);

?>
```

Kodu çalıştırıp tarayıcıdan kaynağına baktığınızda Türkçe karakterlerin kodlandığını göreceksiniz.

Dizge Temizlemek:

Bu işlem için FILTER_SANITIZE_STRING kullanılır. Bu işlem FILTER_SANITIZE_SPECIAL_CHARS işlemine benzemektedir. Fakat fark olarak Html karakterlerini encode etmez, siler ve özel karakterleri encode eder.

```
<?php

$string = '<script>alert("javascript");</script>';

echo filter_var($string, FILTER_SANITIZE_STRING);

?>
```

Yukarıdaki kodu tarayıcıda çalıştırdığınızda <script> etiketlerinin silindiğini fakat alert("javascript"); kodunun silinmediğini görürsünüz. Tarayıcının kaynağına baktığınızda ise parantez içindeki çift tırnakların encode edildiğini göreceksiniz. Bu işlem isteğe bağlı 6 bayrak alabilir:

FILTER_FLAG_NO_ENCODE_QUOTES
FILTER_FLAG_STRIP_LOW
FILTER_FLAG_STRIP_HIGH
FILTER_FLAG_ENCODE_LOW
FILTER_FLAG_ENCODE_HIGH
FILTER_FLAG_ENCODE_AMP

```
<?php

$string = '<script>alert("javascript");</script>';

echo filter_var($string, FILTER_SANITIZE_STRING, FILTER_FLAG_NO_ENCODE_QUOTE
S);

?>
```

Yukarıdaki kodda bayrak olarak FILTER_FLAG_NO_ENCODE_QUOTES geçiriyoruz. Kodu çalıştırdığınızda bir önceki kod ile aynı çıktıyı verdiğini görürsünüz. Fakat sayfa kaynağına baktığınızda parantez içindeki tırnak işaretlerinin encode edilmediğini göreceksiniz. Eğer & işaretini de silmek isterseniz FILTER_FLAG_ENCODE_AMP bayrağını kullanabilirsiniz.

```
<?php

$string = '<b><script>&</script></b><br />';

echo filter_var($string, FILTER_FLAG_ENCODE_AMP);

?>
```

Url Temizlemek:

```
<?php

$url = "http://www.altâayalp.comşğ";
```

```
echo filter_var($url,  FILTER_SANITIZE_URL);

?>
```

Kodun çıktısı http://www.altayalp.com olacaktır. Cümle içerisindeki özel karakterleri (Türkçe karakterler dahil) temizleyerek temiz bir url çıktılar.

İsteğe Bağlı Temizlik:

Bu işlem için FILTER_UNSAFE_RAW kullanılır. Bu filtre bayraklar olmadan hiçbir şey yapmaz. İsteğe bağlı 5 adet bayrakla kullanılabilir:

FILTER_FLAG_STRIP_LOW
FILTER_FLAG_STRIP_HIGH
FILTER_FLAG_ENCODE_LOW
FILTER_FLAG_ENCODE_HIGH
FILTER_FLAG_ENCODE_AMP

```
<?php

$string = '<script>alert("javascript");</script> ŞıİçöüĞ';

echo filter_var($string, FILTER_UNSAFE_RAW, FILTER_FLAG_STRIP_LOW);

?>
```

Yukarıdaki kodu çalıştırdığınızda javascript kodlarının çalıştığını ve Türkçe karakterlerin ekranda yazdığını görürsünüz.

```
<?php

$string = '<script>alert("javascript");</script> ŞıİçöüĞ';

echo filter_var($string, FILTER_UNSAFE_RAW, FILTER_FLAG_STRIP_HIGH);

?>
```

2. kodu çalıştırdığınızda ise javascript'in çalıştığını fakat Türkçe karakterlerin silindiğini göreceksiniz. Aşağıdaki kodları çalıştırdığınızda ise javascript kodlarının çalıştığını ve Türkçe karakterlerin encode edildiğini görürsünüz.

```
<?php

$string = '<script>alert("javascript");</script> ŞıİçöüĞ';

echo filter_var($string, FILTER_UNSAFE_RAW, FILTER_FLAG_ENCODE_HIGH);

?>
```

Kendi Filtremizi Yazalım:

Daha önceden belirttiğimiz gibi FILTER_CALLBACK kullanarak kendi filtreleme işlemimizi yazabiliriz. Php'nin size sunduğu filtreleme seçenekleri çok çeşitli olsa da tam olarak sizin ihtiyacınıza cevap veren bir filtreleme türü olmayabilir. Örneğin bir küfür süzgeci yazmak ve girilen verileri bu süzgeçten geçirmek isteyebilirsiniz. Ya da arama motorlarıyla uyumlu seo url kullanmak isteyebilirsiniz. Örneklerimiz de tam böyle bir küfür süzgeci ve seo link fonksiyonu olacak.

Küfür Süzgeci

```
<?php

$string = 'Sen salak mısın desem, bana manyak der misin? Yoksa aptallaşır
kalır mısın?';

$kufur = array('salak', 'manyak', 'aptal');

function degistir($dizge) {
    global $kufur;
    return str_replace($kufur,'***',$dizge);
}

echo filter_var($string, FILTER_CALLBACK, array('options'=>'degistir'));

?>
```

$string değişkeni işlem yapacağımız veriyi tutuyor, $kufur değişkeni ise elemek istediğimiz kötü sözcükleri. Öncelikle kendi fonksiyonumuzu yazıyoruz. Global $kufur ile fonksiyon dışında tanımladığımız değişkene fonksiyon içinden erişebiliyoruz. str_replace() ile $string değişkeninde bulunan $kufur ile belirlediğimiz sözcükleri *** (üç yıldız) ile değiştiriyoruz. FILTER_CALLBACK sabiti ile options kullanarak fonksiyonumuzu filtreleme işlemi için kullanıyoruz. Kendi fonksiyonumuzu options'a dahil ederken sadece fonksiyon adını yazdığımıza ve parantez kullanmadığımıza dikkat edin.

Seo Linkler

```
<?php
$string = 'Uyusunda büyüsün nenni';

function seolink($baslik) {
$baslik = strtolower(strtr($baslik,'ÜŞÇİĞÖüöşçiğı ','USCIGOuoscigi-'));
return $baslik;
}

$seobaslik = filter_var($string, FILTER_CALLBACK,
array('options'=>'seolink'));

echo $string . '<br />';

echo $seobaslik . '<br />';

echo '<a href="'.$seobaslik.'.html">'.$string.'</a>';
?>
```

Kodumuz küfür süzgeci için yazdığımız koda çok benziyor. **seolink()** fonksiyonumuzda yaptığımız işlem harfleri **strtolower()** fonksiyonu ile küçük harfe çevirmek ve **strtr()** fonksiyonu ile Türkçe karakterleri değiştirmek. Kodu çalıştırdığınızda ilk satırda normal cümle, ikinci satırda değiştirilmiş cümle ve son satırda da seo link olarak çıktı verecektir.

Yukarıdaki kodda görüyorsunuz ki kendi filtreleme fonksiyonlarımızı kolaylıkla **filter_var()** ile kullanabiliyoruz.

Elimden geldiğince sade anlatmaya ve anlaşılması kolay basit kodlarla örneklemeye çalıştım. Herhangi bir hata tespit ederseniz e-posta adresimden iletişime geçebilirsiniz.

Saygılarımla **altayalp**