**PHP Web Proqramlaşdırma dili**

**Şükür Hüseynov 09.04.2016**

**Ön söz**

Kitab PHP dilini 0-dan öyrənmək istəyənlər üçün nəzərdə tutulub. Bu kitab vasitəsilə PHP dilini müəyyən səviyyədə öyrənə bilərsiniz.

Müəlliflə əlaqə: [Programmer\_And\_Developer@mail.ru](mailto:Programmer_And_Developer@mail.ru)

**Mündəricat**

[Php dilinə giriş 4](#_TOC_250024)

[Php dilində şərhlər 5](#_TOC_250023)

[Hesablamalar 6](#_TOC_250022)

[Dəyişənlər 11](#_TOC_250021)

[Şərt Operatorları 13](#_TOC_250020)

[Dövr Operatorları 18](#_TOC_250019)

[Massivlər 22](#_TOC_250018)

[Funksiyalar 35](#_TOC_250017)

Empty(), isset() və unset() funksiyaları 39

[Form daxili əməliyyatlar. Post və Get metodları 40](#_TOC_250016)

[Filter funksiyaları 42](#_TOC_250015)

[Cookies 46](#_TOC_250014)

[Sessiyalar 47](#_TOC_250013)

[Zaman funksiyaları 48](#_TOC_250012)

[String funksiyaları 50](#_TOC_250011)

[Php dilində $\_SERVER massivi 56](#_TOC_250010)

[Fayl funksiyaları 58](#_TOC_250009)

[Php fayl upload əməliyyatı 63](#_TOC_250008)

[Php ilə şəkil hazırlamaq. 65](#_TOC_250007)

[Php və XML 67](#_TOC_250006)

[Mail funksiyaları 68](#_TOC_250005)

[HTTP funksiyaları 69](#_TOC_250004)

[Include və require funksiyaları 70](#_TOC_250003)

[Php və MySql arasında əlaqə 71](#_TOC_250002)

[Obyektyönümlü proqramlaşdırma 74](#_TOC_250001)

Php Data Obejcts 81

[Php və Ajax. 83](#_TOC_250000)

# Php dilinə giriş

Php kodları html kodları arasına yerleşdirilir. Php kodları <?php … ?> arasında yazılır. Başlanğıcda

<?php yerinə sadəcə <? yaza bilərik. Hər iki variant doğrudur.

<?php

…

…

…

?>

Nöqtələrin yerinə php əməliyyatları yazılır.

Php dilində yazını, dəyişəni və s. çap etmək üçün echo funksiyasından istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

<?php

echo “Php dilində ilk yazı";

?>

Echo yazdıqdan sonra boşluq qoyub dırnaq işarəsi açırıq və

dırnaq işarəsi arasında yazmaq

istədiyimiz sözləri yazırıq. Php dilində hər sətrin sonunda ; qoyulmalıdır. Bu kodun nəticəsi

olaraq brauzerdə

yalnızca “Php dilində

ilk yazı" sözləri olacaq. Rəqəmlər yazıldıqda dırnaq

işarəsi qoymağa ehtiyac olmur. Məsələn:

<?php echo 5;

?>

Bu halda ekrana 5 çıxacaq. Eyni zamanda echo daxilində html teqləri də yazıla bilər. Məsələn:

<?php

echo “Php";

echo “<br><font color=red>Php</font>";

?>

Bu kod nəticəsində ekrana Php sözü və növbəti sətirdə qırmızı rəngdə Php sözü yazılacaq.Html teqləri dırnaq içində yazılmalıdır.

# Php dilində şərhlər

Əgər kodlarımız çox uzundursa kodu oxumaq çətin olur. Bu səbəbdən kodların yanında əlavə

şərhlər yaza bilərik. Bu şərhlər kod kimi nəzərə qoyulur. Məsələn:

alınmır. Tək sətirli şərhlər // işarələri ilə

<?php

echo “Şərhlər"; //Verilən şərh

?>

Bu yalnız tək sətirli şərhlər üçündür. 2 və daha çox sətir üçün bu üsuldan istifadə edə bilmərik. İki və daha çox sətirli şərhlər /\*...\*/ işarələri arasında yazılır. Məsələn:

<?php

echo “Çox sətirli şərhlər";

/\* Çox sətirli şərh

.........

.........

................ \*/

?>

Tək sətirli şərhlər daha çox istifadə olunur.

# Hesablamalar

Php dilində müxtəlif riyazi hesablamaları aparmaq mümkündür. Məsələn:

<?php echo 5+7; echo 5\*7; echo 5-7; echo 5/7;

?>

Bu kodun nəticəsi olaraq ekrana hesablamaların nəticəsi çıxacaq. Diqqət edilməli əsas məqam odur ki, yazılan ifadə hesablama olduğu üçün dırnaq işarəsi içində yazılmır. Əgər dırnaq işarəsi içində yazılsa, hesablamanın nəticəsi yox, sadəcə söz kimi “5+7" yazılacaq.Hesablama bir neçə mərhələdən ibarətdirsə, ilk öncə riyaziyyatda olduğu kimi vurma və bölmə, sonra isə toplama və çıxma yerinə yetirilir.Məsələn:

<?php

echo 5+8/4;

?>

Burada nəticə olaraq, 7 çıxacaq əgər biz ilk öncə toplamanın yerinə yetirilməyin istəyiriksə 5+8 hissəsini mötərizə daxilində yazmalıyıq. Məsələn:

<?php

echo (5+8)/4;

?>

Burada ilk öncə 5 və 8 toplanacaq, sonra isə 4-ə bölünəcək.

Əsas 4 hesablama olduğu kimi, digər mürəkkəb hesablamaları da yerinə yetirmək olar**.** Məsələn sqrt funksiyasını işlədərək ədədin kök altısın tapa bilərik. Məsələn:

<?php

echo sqrt(9);

?>

Burada ekrana 3 çıxacaq. Kök altından funksiyalar aşağıda göstərilir:

əlavə

php dilində

bir çox riyazi funksiyalar var. Bu

1. Abs funskiyası-Bu funksiya ədədin modulun tapır.İstifadə qaydası aşağıdakı kimidir:

<?php

echo abs(-5); // Nəticədə ekrana 5 çıxacaq,

?>

1. Acos funksiyası-Bu funksiya riyaziyyatdakı arccos funksiyasıdır.İşlənmə şəkildədir:

qaydası aşağıdakı

<?php

echo acos(0.5);

?>

Proqramlaşdırma dillərində

triqonometrik funksiyalarda vahid olaraq radiandan istifadə

olunur.

Yuxarıdakı funksiyanın nəticəsi də radianla veriləcək. Nəticəni dərəcəyə çevirmək üçün lazım

olan funksiya ilə irəlidə tanış olacağıq.

1. Acosh funksiyası-Hiperbolik kosinus funksiyasıdır. İşlənmə qaydası digər riyazi funksiyalarla eynidir.
2. Asin funksiyası-Arcsin funksiyasıdır. Nəticə radianla verilir. Məsələn:

<?php

echo asin(0.5);

?>

1. Asinh funksiyası-Hiperbolik sinus funksiyasıdır. İşlənmə eynidir.

qaydası digər riyazi funksiyalarla

1. Atan funksiyası-Arctg funksiyasıdır. Daxiletmə radianladır. Məsələn:

<?php

echo atan(21);

?>

1. Atanh funksiyası-Hiperbolik tg funksiyasıdır. İşlənmə qaydası digər riyazi funksiyalarla eynidir. 8)Base\_convert funksiyası-Bu funksiya bir say sistemində olan ədədi digərinə çevirir. Məsələn:

<?php

$hexadecimal = 'A37334';

echo Base\_convert($hexadecimal, 16, 2);

?>

Burada 16-lıq say sistemindəki ədəd 2-lik say sistemindəki ədədə çevrilir. 9)Ceil funksiyası-Bu funksiya ədədi yuxarı yuvarlaqlaşdırır. Məsələn:

<?php

echo ceil(4.3); echo ceil(2.7);

?>

Nəticədə uyğun olaraq 5 və 3 çıxacaq.

1. Cos funksiyasıdır.-Kosinus funksiyasıdır. Nəticə radianladır. Məsələn:

<?php

echo cos(30); // 0.15425144988758

?>

1. Cosh funksiyası-Hiperbolik kosinus funksiyasıdır. İşlənmə qaydası digər riyazi funksiyalarla eynidir.
2. Deg2rad funksiyası-Bu funksiya dərəcəni radiana çevirir. Məsələn:

<?php

echo deg2rad(45); // 0.785398163397

?>

1. Exp funksiyası-Bu funksiya e üstü hərhansı bir ədədi tapmaq üçündür. Məsələn:

<?php

echo exp(2); // Nəticədə 7.3890560989307 çıxacaq

?>

1. Floor funksiyası- Bu funksiya ədədi aşağı yuvarlaqlaşdırır.Məsələn:

<?php

echo floor(4.8); // Nəticədə ekrana 4 çıxacaq.

?>

1. Log10 funksiyasə- Bu funksiya 10 əsasdan loqarifmanı tapmaq üçündür. Məsələn:

<?php

echo log10(1000); // Nəticədə 3 çıxacaq

?>

1. Log funksiyası-Natural loqarifma funksiyasıdır. Məsələn:

<?php

echo Log(10); // Nəticədə 2.302585092994 çıxacaq.

?>

1. Max funksiyası-Daxil edilmiş ədədlərdən ən böyüyünü tapmaq üçündür. Məsələn:

<?php

echo max(4,7); // Nəticədə 7 çıxacaq

echo max(7,4,8,9,1); // Nəticədə 9 çıxacaq

?>

Bu funksiyada ədədlərlə sözlərdə müqayisə oluna bilər. Məsələn:

<?php

echo max('word',-3);

?>

Burada -3 rəqəmi 0-dan kiçik olduğu üçün maksimum olaraq word sözü olacaq.

1. Min funksiyası-Verilmiş ədədlərdən ən kiçiyini tapmaq üçün istifadə olunur. İstifadə qaydası max funksiyası ilə eynidir.
2. Pi funksiyası-Pi ədədinin qiymətini verir. Məsələn:

<?php echo pi();

?>

Bu funksiyanın yerinə M\_Pİ işlədilə bilər. Məsələn:

<?php

echo M\_PI; // Nəticədə 3.14159265358979323846 alınır echo M\_PI\_2; // Nəticədə Pi/2 alınır

echo M\_Pİ\_4; // Nəticədə Pi/4 alınır echo M\_1\_Pİ;// Nəticədə 1/Pi alınır echo M\_2\_PI; // Nəticədə 2/Pi alınır

?>

1. Pow funksiyası-Üstlü funksiyaları hesablamaq üçündür. Məsələn:

<?php

echo pow(2,8); // Nəticədə 256 alınacaq echo pow (-1,10); // Nəticədə 1 alınacaq

?>

1. Rad2deg funksiyası-Radianı dərəcəyə çevirmək üçün istifadə olunur. Məsələn:

<?php

echo rad2deg(1); // Nəticədə 57.295779513082 alınacaq

?>

1. Rand funksiyası-Təsadüfi ədəd istehsal etmək üçün istehsal olunur. Məsələn:

<?php

echo rand();

?>

Nəticədə təsadüfi bir ədəd çap olunacaq. Bu ədəd müəyyən aralıqdan da seçilə bilər. Məsələn:

<?php

echo rand(5,15);

?>

Nəticədə 5 və 15 aralığından təsadüfi bir ədəd seçiləcək.

1. Round funksiyası-Bu funksiya ədədi yuvarlaqlaşdırmaq üçün istifadə olunur. Məsələn:

<?php

echo round(5.7); //Nəticədə 6 alınacaq echo round(5.3); //Nəticədə 5 alınacaq

?>

1. Sin funksiyası-Sinus funksiyasıdır. Daxiletmə radianladır. Məsələn:

<?php

echo sin(2); // Nəticədə 0.90929742682568 alınacaq

?>

Bir neçə funksiyanı bir yerdə işlətmək mümkündür. Yuxarıda göstərilmiş deg2rad funksiyası ilə sin funksiyasını bir yerdə işlədərək sinusu dərəcə ilə hesablaya bilərik. Məsələn:

<?php

echo sin(deg2rad(30)); Nəticədə 0.5 alınacaq

?>

Burada ilk öncə içəridəki deg2rad funksiyası öz işini görür və 30 dərəcəni radiana çevirir. Sinus isə alınmış radianla qiyməti hesablayır.

1. Sinh funksiyası-Hiperbolik sinus funksiyasıdır. İstifadə eynidir.

qaydası digər riyazi funksiyalarla

1. Tan funksiyası-Tg funksiyasıdır.Daxiletmə radianladır. Məsələn:

<?php

echo tan(deg2rad(45)); // Nəticədə 1 alınacaq

?>

1. Tanh funksiyası- Hiperbolik tg funksiyasıdır. İstifadə qaydası digər riyazi funksiyalarla eynidir.

Dəyişənlər proqramlşadırmanın ən

# Dəyişənlər

əsas elementlərindən biridir. Dəyişənlərin qiymətləri

proqramçı tərəfindən təyin oluna bilər və kod emal olunduqdan sonra istifadəçinin seçimi ilə dəyişə bilər.Php dilində dəyişənlər $ işarəsi ilə elan olunur. Məsələn:

<?php

$a=5;

?>

Burada sadəcə

a dəyişəni elan olundu.

$ işarəsi a-nın dəyişən

olduğunu göstərir.

Burada

dəyişənin adı isə a-dır. Dəyişənin adı üçün aşağıdakı qaydalar var: 1)Dəyişən adları rəqəmlə başlaya bilməz;

2)Dəyişən adları daxilində yalnız hərflər,rəqəmlər və \_ işarəsi ola bilər; 3)Dəyişən adlarında böyük və kiçik hərflər ayrı nəzərdə tutulur. Məsələn ayrı-ayrı dəyişənlər sayılır. Məsələn:

<?php

$a=5;

$A=7;

?>

Burada $a dəyişəninin qiyməti 5, $A dəyişəninin qiyməti isə 7 olur. Dəyişənlər echo funksiyası vasitəsilə çap oluna bilər. Məsələn:

<?php

$start=15; echo $start;

?>

$a və $A dəyişəni

Burada $start dəyişəni çap olunaraq ekrana 15 çıxacaq. Digər proqramlaşdırma dillərindən fərqli olaraq, Php dilində dəyişənin tipi dəyişən elan olunarkən göstərilmir. Yuxarıdakı $start dəyişəni ədəd tipli dəyişən sayılır. Bundan əlavə dəyişənlər yazı və məntiqi tipdə ola bilərlər. Məsələn:

<?php

$start="Yazılar"; echo $start;

?>

Burada $start dəyişəni yazı tipindədir.Yazı tipində olduğuna görə aldığı qiymət, yəni “Yazılar" sözü dırnaq işərisində yazılmalıdır. Məntiqi tiplər isə True(Doğru) və False(Yalnış) olmaqla iki cür olur. Məsələn:

<?php

$tip=true;

$next=false;

?>

Məntiqi tiplər şərtlərin yoxlanılması və başqa yerlərdə istifadə olunur.Dəyişənin qiyməti olaraq, aldığı ən son qiymət nəzərdə tutulur. Məsələn:

<?php

$a=4;

$a=8; echo $a;

?>

Burada $a dəyişəninin qiyməti 8 sayılır və

ekrana 8 çap olunur. Adi

ədədlərlə

hesablama

aparıldığı kimi dəyişənlərlə də aparıla bilər. Məsələn:

<?php

$a=8;

$b=3;

$c=$a+$b; echo $c;

?>

Burada $a dəyişəninə 8,

$b dəyişəninə

isə

3 qiymətini veririk. $c dəyişəninə

isə $a və $b

dəyişənlərini cəmini veririk və $c dəyişənini çap edirik. Nəticədə funksiyaların qiymətlərinidə verə bilərik. Məsələn:

<?php

$a=sqrt(25);

$b=pow(2,8);

$c=sin(deg2rad(30)); echo $a, $b, $c;

?>

11 alınır. Dəyişənlərə

Bir neçə dəyişəni bir yerdə çap edərkən yuxarıda olduğu kimi vergüllə ayıraraq tək bir echo

funksiyası ilə

çap edə

bilərik.Nəticədə

qiymətlər yan-yana yazıldığı üçün qarışıqlıq alınacaq.

Buna görə də <br> teqini də işlədərək qiymətləri alt-alta yaza bilərik. Məsələn:

<?php

$a=sqrt(25);

$b=pow(2,8);

$c=sin(deg2rad(30));

echo $a."<br>".$b."<br>".$c;

?>

# Şərt operatorları

Php dilində əməliyyatları müəyyən şərt daxilində yerinə yetirmək üçün şərt operatorlarından

istifadə olunur. İlk öyrənəcəyimiz şərt operatoru if operatorudur. If operatoruna başlamazdan

əvvəl if operatoru ilə işlənən müqayisə işarələrini göstərək:

== Bərabərlik, != Bərabər deyil, > Böyükdür, < Kiçikdir, >= Böyükdür və ya bərabərdir, <= Kiçikdir və ya bərabərdir.

If iperatorunun istifadə qaydası aşağıdakı kimidir:

<?php if(şərt){

…

…

...

}

?>

Burada '{' və '}' işarələri arasına şərt ödəndiyi halda yerinə

yetiriləcək

əməliyyatlar yazılır.

Məsələn 5-in 4-dən böyük olması halını yoxlayaraq bir əməliyyatı icra edək:

<?php if(5>4){

echo “Şərt ödənir";

}

?>

Burada 5 həqiqətən də 4-dən böyük olduğu üçün

'{' və ' }' işarələri arasındakı əməliyyatlar

yerinə yetirilir. Burada əməliyyat yalnız echo funksiyası vasitəsilə ekrana yazı yazdırmaqdır. Biz bu şərti 5<4 olaraq dəyişdirsək şərt ödənmədiyi üçün əməliyyat yerinə yetirilməyəcək.

<?php if(5<4){

echo “Şərt ödənmir";

}

?>

Şərt ödənmədiyi üçün

'{' və '}' işarələri arasındakı

əməliyyatlar yerinə

yetirilmir. İndi isə

dəyişənlərdən istifadə edərək şərt if operatoruna daha bir nümunə göstərək:

<?php

$a=8;

$b=10; if($b>$a){

echo “Şərt ödəndi. Əməliyyatlar yerinə yetirilir."; echo “Şərt doğrudur.";

}

?>

Burada a və

b dəyişənlərinə

qiymət verdik. Sonra isə b dəyisşəninin a dəyişənindən böyük

olub-olmamasını yoxladıq. Şərt doğru olduğu üçün

'{' və '}' işarələri arasındakı

əməliyyatlar

yerinə

yetiriləcək.

'}' işarəsindən sonra yazılan

əməliyyatlar isə

şərt operatorunun

əməliyyatlarına daxil olmur. Məsələn:

<?php

$a=8;

$b=10; if($b>$a){

echo “Şərt ödəndi. Əməliyyatlar yerinə yetirilir."; echo “Şərt doğrudur.";

}

echo “Bu sətir və növbəti sətirlər isə yuxarıdakı şərtə daxil olmadığı üçün yuxarıdakı şərtdən asılı olmayaraq yerinə yetiriləcək.";

?>

İndi isə 2 şərt operatorundan istifadə edək:

<?php

$a=8;

$b=10; if($b>$a){

echo “Şərt ödəndi. Əməliyyatlar yerinə yetirilir."; echo “Şərt doğrudur.";

}

if($b<$a){

echo “Şərt ödənmir.";

}

?>

Burada iki dəfə if operatorundan istifadə etdik. İkinci şərtdə birincinin əksini göstərdik. Ancaq

əksini göstərmək üçün iki dəfə

if operatorundan istifadə

etməyə

ehtiyac yoxdur. Şərt

ödənmədikdə digər bütün hallarda olacaq əməliyyatları göstərmək üçün else yazılır. Məsələn:

<?php

$a=8;

$b=10; if($b>$a){

echo “Şərt ödəndi. Əməliyyatlar yerinə yetirilir."; echo “Şərt doğrudur.";

}

else{

echo “Burada isə yuxarıdakı şərt ödənməsə qalan bütün hallar üçün olacaq əməliyyatlar yazılır.";

}

?>

Şərtin ödənmədiyi bütün digər hallar üçün əməliyyatlar yerinə yetirmək istəyiriksə if ilə bir

yerdə

else istifadə

etməliyik.Yuxarıdakı şərtin ödənmədiyi hallar

$b dəyişəninin $a

dəyişənindən kiçik olması və $b dəyişənin $a dəyişəninə bərabər olması hallarıdır. Ancaq biz böyük olanda bir əməliyyat, bərabər olanda başqa bir əməliyyat, digər hallarda isə yenə başqa bir əməliyyatı yerinə yetirmək istəyiriksə else if istifadə etməliyik. Məsələn:

<?php

$a=8;

$b=10; if($a>$b){

echo “a dəyişəninin b dəyişənindən böyük olduğu hal";

}

else if($a==$b){

echo “a dəyişəninin b dəyişəninə bərabər olduğu hal";

}

else{

echo “Qalan bütün hallar.";

}

?>

Burada əvvəlcə $a dəyişəninin $b dəyişənindən böyük olma halı yoxlanılır. Şərt ödənmədiyi üçün şərt daxilindəki əməliyyatlara baxılmadan ikinci şərtin, yəni, $a dəyişəninin $b dəyişəninə bərabər olma halı yoxlanılır. Bu hal da doğru olmadığına görə bu şərt daxilindəki əməliyyatlara

baxılmadan else hissəsinə

keçilir. Burada isə

heç bir şərt yoxlanılmadan else daxilindəki

əməliyyatlar yerinə

yetirilir. Ancaq burada şərt sadə

olduğuna görə

yoxlanılmayan son halın

$a<b$ olduğu bizə məlumdur. Eyni zamanda çox sayda else if istifadə edə bilərik. Məsələn:

<?php

$a=100;

if($a==1){

echo “Dəyişənin qiyməti 1-dir.";

}

else if($a==2){

echo “Dəyişənin qiyməti 2-dir.";

}

else if($a==3){

echo “Dəyişənin qiyməti 3-dür.";

}

else if($a==4){}(

echo “Dəyişənin qiyməti 4-dür.";

)

else{

echo “Dəyişənin qiyməti fərqli bir ədəddir.";

}

?>

Burada nəticə olaraq “Dəyişənin qiyməti fərqli bir ədəddir." sözü ekrana çıxacaq. İndi isə daha bir kodu nəzərdən keçirək:

<?php

$a=100;

if($a==1){

echo “Dəyişənin qiyməti 1-dir.";

}

else if($a==2){

echo “Dəyişənin qiyməti 2-dir.";

}

else{

echo “Dəyişənin qiyməti fərqli bir ədəddir.";

} if(5>3){

echo “Bu yeni şərt operatorudur və şərt doğrudur.";

}

?>

Burada axırda işlədilən if operatorunun yuxarıdakı if, else və else if ilə heç bir əlaqəsi yoxdur.

Yuxarıda işlədilən if, else if və else bir-birilə əlaqəlidir. Else if və else ilk işlədilən if-in əks

hallarıdır. Ancaq, sonda işlədilən if operatoru daxilində yeni şərt və yeni əməliyyatlardır və bu

şərt və əməliyyatlar yuxarıdakı operator və baxaq:

şərtlərdən asılı deyil. Daha bir kod nümunəsinə

<?php

$a=100;

if($a==1){

echo “Dəyişənin qiyməti 1-dir.";

}

else if($a==2){

echo “Dəyişənin qiyməti 2-dir.";

}

else{

echo “Dəyişənin qiyməti fərqli bir ədəddir.";

} if(5>3){

echo “Bu yeni şərt operatorudur və şərt doğrudur.";

}

$a=7; if($a>4){

echo “Digər şərt.";

}

?>

Burada sonda $a dəyişəninə yeni qiymət verilmiş və şərtlə yoxlanılmışdır. Kodun ilk əvvəlində

$a dəyişəninə 100 qiyməti verilmişdir. Sonra if operatoru işlədilərək $a dəyişəni yoxlanılmışdır.

$a dəyişənini 100 etdikdən sonra yazdığımız ilk şərtlər a dəyişəninin qiymətini 100 olaraq yoxlayır. Sonda isə a dəyişəninə 7 qiymətini veririk. 7 qiymətini verdikdən sonra yazdığımız şərt

isə $a dəyişəninin qiymətini 7 olaraq yoxlayır. Yəni, Yuxarıda yazdığımız ilk operatorda $a

dəyişəninin qiyməti 7 olaraq görülmür. Ümümiyyətlə proqramlaşdırmada operator öz işini

görərkən özündən sonrakı sətirləri nəzərə almır.Yalnız özündən əvvəlki sətirlər nəzərə alınır. Məsələn 2-ci sətirdə $a dəyişəninə 5 qiyməti verilmişdir. 8-ci sətirdə $a dəyişəni if operatoru

vasitəsilə

yoxlanılmışdır və əməliyyatlar yerinə

yetirilmişdir. 12-ci sətirdə

isə

yenidən $a

dəyişəninə 10 qiyməti verilmişdirsə 8-ci sətirdəki if operatoru a dəyişəninin yalnız 2-ci sətirdəki qiymətinə baxır. 12-ci sətir öz sətrindən sonra gəldiyinə görə operator bu sətrə baxmır.

Real layihələrdə

operator daxilində

onlarla, hətta yüzlərlə

şərt yazmaq lazım gələ

bilər. Bu

zaman if operatorundan istifadə etmək çətin olur. Şərtlərin sayı çox olduqda digər şərt operatoru olan switch operatorundan istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

<?php switch($a){

case 'hal1': {Bu halda}; break;

case 'hal2': {Digər hal}; break;

case 'hal3': {Digər hal}; break;

default: {Qalan bütün hallar};

}

?>

Burada yoxlanılacaq dəyişən

$a dəyişənidir.

Switch yazıldıqdan sonra mötərizə

açırıq və

yoxlanılacaq dəyişəni yazırıq.Case sözündən sonra halı yazıb iki nöqtə qoyuruq. Sonra isə yerinə yetiriləcək əməliyyatları yazırıq. Növbəti sətirdə break yazmalıyıq. Əgər break yazmasaq, şərt ödənsə belə yenə digər hallara baxılacaq. Bu səbəbdən break yazmaq vacibdir. Sonda isə default yazırıq və qarşısında digər bütün hallarda yerinə yetiriləcək əməliyyatları yazırıq. Məsələn:

<?php

$a=3; switch($a){

case 1: echo "Doğru olmayan hal"; break;

case 2: echo "Doğru olmayan hal"; break;

case 3: echo "Doğru hal"; break;

default: echo "Digər hallar";

}

?>

Burada $a dəyişəni 3 olduğu üçün 3-cü hal doğru olur və müvafiq əməliyyat yerinə yetirilir.İndi isə switch operatoruna aid daha bir nümunə yazaq. Fərz edək ki, a dəyişəni istifadəçinin forma daxil etdiyi sözdür(Bunu irəlidə öyrənəcəyik). Həmin söz ingiliscə daxil olunub və biz həmin sözün Azərbaycan dilində qarşılığın yazmalıyıq. O zaman kod aşağıdakı kimi olar:

<?php

$a="apple"; switch($a){

case "apple": echo "Alma"; break;

case "Orange": echo "Protağal"; break;

default: echo "Axtarılan söz tapılmadı.";

}

?>

Burada düzgün hala uyğun müvafiq nəticə çıxacaq.

# Dövr operatorları

Eyni bir

əməliyyatı bir neçə

dəfə

yerinə

yetirmək üçün dövr operatorlarından istifadə

olunur.Əgər

əməliyyatların sayı məlumdursa for dövr operatorundan istifadə

olunur. İstifadə

qaydası aşağıdakı şəkildədir:

<?php

for($i=ilk qiymət;$i<son qiymət, artım){ echo “Təkrarlanacaq əməliyyat";

}

?>

For operatoruna aid bir nümunə göstərək:

<?php for($a=0;$a<10;$a=$a+1){ echo "Əməliyyat <br>";

}

?>

Burada ilk öncə operator daxilində ilk parametr olaraq

$a dəyişəninə 0 qiyməti veririk. İkinci

paramatrdə dəyişənin 10-dan kiçik olduğu halda əməliyyatların yerinə yetirilməsini təmin edirik. Üçüncü parametrdə isə artım olaraq $a dəyişəninin hər dövrdə 1 vahid artmasını təmin edirik.

Əvvəlcə

dövr daxilində $a dəyişəninin 0 qiymətinə

baxılır. Bu dəyişənin qiymətinin 10-dan

kiçik olması yoxlanılır.

Əgər dəyişən həqiqətən də

10-dan kiçikdirsə əməliyyatlar bir dəfə

yerinə

yetirilir

və a dəyişəninin qiyməti 1 vahid artır. Artıq

$a dəyişəninin qiyməti 1 vahid

artdığına görə yeni qiyməti 1-dir. Yenidən bu qiymətin 10-dan kiçik olması yoxlanılır. 1

həqiqətən də

10-dan kiçik olduğuna görə əməliyyatlar ikinci dəfə

yerinə

yetirilir və $a

dəyişəninin qiyməti yenidən 1 vahid artırılır. Bu proses 9-a kimi davam edir. Dəyişənin qiyməti 9-a çatdıqda 9-un 10-dan kiçik olması yoxlanılır. Bu dəfə də dəyişən 10-dan kiçik olduğuna görə

əməliyyat təkrar yerinə

yetirilir və

dəyişənin qiyməti bir vahid artırılaraq 10 olur. Bu dəfə

dəyişənin qiymətinin 10-dan kiçik olması yoxlanılır. Dəyişənin qiyməti 10-dan kiçik olmadığına

görə

heç bir

əməliyyat yerinə

yetirilmədən dövr başa çatır.

If operatorunda olduğu kimi for

operatoru daxilində də digər müqayisə əməlləri işlədilə bilər.Eyni zamanda artan dəyişəni də dövr daxilində işlədə bilərik. Məsələn:

<?php for($a=0;$a<100;$a=$a+1){

echo $a."-nın kvadratı: ".$a\*$a."<br>";

}

?>

For operatorunun daxilində digər operatorlardan istifadə etmək olar. Məsələn:

<?php for($a=0;$a<=100;$a=$a+1){ if($a!=8){

echo $a."-nın kvadratı: ".$a\*$a."<br>";

}

}

?>

a dəyişəninin qiyməti ilk öncə 0-dır. Bu qiymətin 100-dən kiçik olması yoxlanılır. Hal doğru

olduğu üçün

əməliyyatlara başlanılır.

Əməliyyat isə

həmin dəyişəni if operatoru vasitəsilə

yoxlayıb ekrana həmin dəyişənin qiymətinin kvadratını yazmaqdır. Yoxlama aparılır və 0 rəqəmi 8-dən fərqli olduğu üçün if operatorunun əməliyyatı da yerinə yetirilir. Dövr əməliyyatları bitir və a dəyişəninin qiyməti bir vahid artır. Yenidən eyni yoxlanışlar olur. Dəyişənin qiyməti 8-ə çatdıqda isə if operatorunda yoxlama səhv olduğu üçün kvadratı çap olunmur və dəyişən bir vahid artaraq yenidən dövrə davam edir.

İndi isə iç-içə dövrlərdən istifadə edək. İç-içə dövrlərin istifadəsi daha qəlizdir. Belə ki iç-içə istənilən qədər dövr yaza bilərik. Məsələn:

<?php for($a=0;$a<5;$a=$a+1){ for($b=0;$b<8;$b=$b+1){ echo "Ok <br>";

}

}

?>

İlk öncə ilk dövrdə $a dəyişəninin qiyməti 0-dır. 5-dən kiçik olduğu üçün ilk dövrün

əməliyyatlarına başlanılır.Sonra içəridəki dövrə başlanır. İçəridəki dövrün əməliyyatları 8 dəfə yerinə yetiriləcək. İçəridəki dövr tam bitdikdən sonra birinci dövrdə $a dəyişəninin qiyməti bir

vahid artırılacaq və

yenə

içəridəki dövr

əməliyyatları 8 dəfə

yerinə

yetiriləcək. Bu dövr

bitdikdən sonra yenidən

əvvələ

qayıdılacaq və

a dəyişəninin qiyməti bir vahid artacaq. $a

dəyişəninin qiymətinin 5-dən kiçik olması yoxlanıldıqdan sonra içəridəki dövrə keçiləcək. Bu dəfə də içəridəki dövr yerinə yetiriləcək. Beləliklə ekrana 40 dəfə yazı yazılacaq.

Yazdığımız $a=$a+1 yerinə $a++ yaza bilərik. Məsələn:

<?php for($a=0;$a<5;$a++){ echo "Ok <br>";

}

?>

For operatoruna aid bir neçə görülür.

nümunə

yazıb sonra növbəti operatorlara keçməyiniz məsləhət

Növbəti öyrənəcəyimiz operator while operatorudur. Bir şərt doğru olduğu müddətcə

əməliyyatları yerinə aşağıdakı şəkildədir:

yetirmək üçün

while operatorundan istifadə

olunur. İstifadə

qaydası

<?php while(Şərt){

// Əməliyyatlar

}

?>

Şərt doğru olduğu müddətcə əməliyyatlar yerinə yetirilir. Məsələn:

<?php

$a=0; while($a<10){ echo "Ok <br>";

}

?>

Burada $a dəyişəni 10-dan kiçik olduğu müddətcə

dövr

əməliyyatları yerinə

yetirilir. Ancaq

burada $a dəyişəni həmişə 10-dan kiçik olduğu üçün daima ekrana yazı yazılacaq. Bu isə sonsuz

dövr adlanır. Sonsuz dövr olmaması üçün artırmalıyıq. Məsələn:

əməliyyatların sonunda

$a dəyişəninin qiymətini

<?php

$a=0; while($a<10){ echo "Ok <br>";

$a++;

}

?>

Burada $a dəyişəninin ilk qiyməti əvvəlvədən 0 təyin olunub. A dəyişəninin 10-dan kiçik olması yoxlanılır. Şərt doğru olduğu üçün əməliyyatlara başlanılır. Yazı bir dəfə yazılır, $a dəyişəninin qiyməti bir vahid artır və yenidən əvvələ qayıdır. $a dəyişəninin yeni qiyməti də yoxlanılır və

əməliyyatlar yenidən yerinə

yetirilir. Bu proses

$a dəyişəni 10 olana qədər davam edir. Bu

operatorunda iç-içə işlədə bilərik. Məsələn:

<?php

$a=0; while($a<10){

$b=0; while($b<10){ echo "Ok <br>";

$b++;

}

$a++;

}

?>

Burada $b=0; sətrini birinci

while operatorunun içində

yazmalıyıq. Çünki içəridəki

while

operatoru öz işini qurtardıqdan sonra $b dəyişəninin qiyməti 10 olur. Belə olduqda birinci while operatorunun növbəti dövrlərində içəridəki dövr əmməliyyatları yerinə yetirilməyəcək.

Növbəti öyrənəciyimiz dövr operatoru do aşağıdakı kimidir:

while operatorudur. Bu operatorun istifadə

qaydası

<?php do{

// Əməliyyatlar

}while(Şərt);

?>

Bu operatorun while operatorundan fərqi odur ki, şərt doğru olmasa belə əməliyyatlar ən əz bir əfə yerinə yetirilir. Məsələn:

bu operatorda

<?php

$a=0; do {

echo "Ok <br>";

$a++;

} while($a<10);

?>

Burada ilk öncə əməliyyatlar yerinə yetirilir, sonra isə şərt yoxlanır.

Bir adda bir neçə

# Massivlər

dəyişən elan etmək üçün massivlərdən istifadə

olunur. Massivlər Array

funksiyası vasitəsilə yaradılır. Məsələn:

<?php

$massiv=array("start","end","game");

?>

Burada massivi yaratdıq. İndi isə massiv elementinə müraciət edərək ekrana yazdıraq.

<?php

$massiv=array("start","end","game"); echo $massiv[0];

?>

Burada $massiv[0] yazdıqda 1-ci element nəzərdə

tutulur. Proqramlaşdırmada sıralama 0-dan

başlanır. Biz massiv elementinə

müraciət edərkən sıra nömrəsinə

görə

müraciət etdik. Sıra

nömrəsindən əlavə biz massiv elementinə açar sözlə də müraciət edə bilərik.Bunun üçün biz massivi yaradarkən elementləri açar sözlə yaratmalıyıq. Məsələn:

<?php

$massiv=array("birinci" => "start","ikinci" => "end"); echo $massiv["birinci"];

?>

Burada hər massiv elementinə bir açar söz verdik.

Massivləri array funksiyasını işlətmədən də yarada bilərik. Məsələn:

<?php

$massiv[0]="start";

$massiv["yeni"]="end"; echo $massiv[0];

echo $massiv["yeni"];

?>

Burada massivi array funksiyası işlətmədən yaratdıq. Massivin bütün elementlərini ekrana çıxarmaq üçün print\_r funksiyasından istifadə olunur. Məsələn:

<?php

$massiv=array("one" => "value", "two" => "value2"); print\_r($massiv);

?>

Burada massiv çap olunur. Aşağıdakı şəkildə də yazmaq olar:

<?php

$massiv[0]="value1";

$massiv["yeni"]="value2"; print\_r($massiv);

?>

Burada da massiv çap olunur. Massivdəki element sayını tapmaq üçün count() funksiyasından istifadə olunur. Məsələn:

<?php

$massiv=array("one","two","three"); echo count($massiv);

?>

Nəticədə

ekrana massiv elementlərinin sayı çıxacaq. Massivi silmək üçün isə

unset()

funksiyasından istifadə olunur. Məsələn:

<?php

$massiv=array("one","two","three"); unset($massiv);

print\_r($massiv);

?>

Massiv silindiyi heç nə çap olunmayacaq.

Massiv funksiyaları vasitəsilə Bu funksiyalar aşağıdakılardır:

massivlər üzərində müxtəlif

əməliyyatlar aparmaq mümkündür.

1. array\_change\_key\_case() funksiyası- Massivdəki açar sözlərin hərflərini böyütmək və ya kiçiltmək üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$massiv=array("one" => "element1","two" => "Element2","three" => "Element3"); print\_r(array\_change\_key\_case($massiv,CASE\_UPPER));

?>

Burada nəticə olaraq massivin açar sözlərinin hərfləri böyük hərflərə çevrilir. CASE\_UPPER

əvəzinə CASE\_LOWER yazsaq hərflər kiçik olacaq.

1. array\_chunk() funksiyası-Massivləri bir neçə qaydası aşağıdakı şəkildədir:

massivə

bölmək üçün istifadə

olunur. İstifadə

<?php

$cars=array("Volvo","BMW","Toyota","Honda","Mercedes","Opel"); print\_r(array\_chunk($cars,2));

?>

Nəticədə massivi 2 massivə böldük.Ancaq əgər biz elementləri açar sözlə yazmışıqsa yuxarıdakı kimi yazsaq açar sözlər müvafiq ədədlərlə əvəz olunacaq. Bu səbəbdən əgər elementləri açar sözlərlə daxil etmişiksə funksiyanı aşağıdakı şəkildə işlətməliyik:

<?php

$fname=array("ad" => "Peter","ad2" => "Ben", "ad3" => "Joe"); print\_r(array\_chunk($fname,2, true));

?>

Belə yazdıqda açar sözlər saxlanacaq.

1. array\_column() funksiyası-Funksiya verilənlər bazası mövzusunda ətraflı izah olunacaq.
2. array\_combine() funksiyası- Bu funksiya massivləri birləşdirərək yeni massiv yaratmaq üçün istfadə olunur. Məsələn:

<?php

$fname=array("Peter","Ben","Joe");

$age=array("35","37","43");

$c=array\_combine($fname,$age); print\_r($c);

?>

Burada birinci massivin elementləri yeni massivin açar sözləri olur. İkinci massivin elementləri isə yeni massivin elementləri olur.

1. array\_count\_values() funksiyası- Bu funksiya massivdəki bütün elementləri saymaq üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$a=array("A","Cat","Dog","A","Dog"); print\_r(array\_count\_values($a));

?>

Nəticə olaraq hər elementdən neçə ədəd olduğu çap olunur.

1. array\_diff() funksiyası- Bu funksiya massivləri müqayisə etmək üçün istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");

$a2=array("e"=>"red","f"=>"green","g"=>"blue");

$result=array\_diff($a1,$a2); print\_r($result);

?>

Nəticədə $a1 massivində olan, ancaq digər massivdə olmayan elementlər çap olunacaq. Başqa bir nümunə:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");

$a2=array("e"=>"red","f"=>"black","g"=>"purple");

$a3=array("a"=>"red","b"=>"black","h"=>"yellow");

$result=array\_diff($a1,$a2,$a3); print\_r($result);

?>

Nəticədə $a1 massivində olan və digər massivlərdə olmayan elemntlər çap olunacaq.

1. array\_diff\_key() funksiyası- Massivləri açar sözlərinə olunur. İstifadə qaydası aşağıdakı şəkildədir:

görə

müqayisə

etmək üçün isitfadə

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue");

$a2=array("a"=>"red","c"=>"blue","d"=>"pink");

$result=array\_diff\_key($a1,$a2);

print\_r($result);

?>

Yalnız $a1 massivində “b” açar sözü olduğuna görə yalnız bu açar söz və elementi çap olunacaq. Aşağıdakı kodda massivddə açar sözlər təyin olunmayıb:

<?php

$a1=array("red","green","blue","yellow");

$a2=array("red","green","blue");

$result=array\_diff\_key($a1,$a2); print\_r($result);

?>

Nəticədə yellow çap olunacaq, çünki 3-cü nömrə(Proqramlaşdırmada sıralama 0-dan başlayır)

yalnız $a1 massivində var. Başqa bir nümunə:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue");

$a2=array("c"=>"yellow","d"=>"black","e"=>"brown");

$a3=array("f"=>"green","c"=>"purple","g"=>"red");

$result=array\_diff\_key($a1,$a2,$a3); print\_r($result);

?>

Nəticədə a və b açar sözlərinin elementləri çap olunacaq.. Çünki a və b açar sözləri yalnız birinci massivdə var.

1. array\_flip() funksiyası- Massivdəki açar sözlərlə elementlərinin yerini dəyişmək üçün istifadə olunur. İsitfadə qaydası aşağıdakı şəkildədir.

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");

$result=array\_flip($a1); print\_r($result);

?>

Nəticədə massivin açar sözləri ilə elementlərinin yerləri dəyişəcək.

1. array\_intersect() funksiyası- Massivləri müqayisə etmək üçün istifadə olunur. Digər müqayisə funksiyalarından fərqli olaraq burada alınmış yeni massiv müqayisə olunmuş massivlərin ortaq elementlərindən təşkil olunacaq.

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");

$a2=array("e"=>"red","f"=>"green","g"=>"blue");

$result=array\_intersect($a1,$a2); print\_r($result);

?>

Nəticədə alınmış massiv: Array ( [a] => red [b] => green [c] => blue )

Nəticədə

alınmış yeni massivdə

yuxarıdaki iki massivin ortaq

elementləri

olacaq. Başqa bir

nümunə göstərək:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");

$a2=array("e"=>"red","f"=>"black","g"=>"purple");

$a3=array("a"=>"red","b"=>"black","h"=>"yellow");

$result=array\_intersect($a1,$a2,$a3); print\_r($result);

?>

Nəticədə alınmış massiv: Array ( [a] => red )

Əlavə olaraq deyək ki, müqayisə olunarkən yalnız elementlər müqayisə olunur və açar sözlərə baxılmır.

1. array\_intersect\_assoc()- Bu funksiya massivləri eyni zamanda həm elementlərə, həm də

elementlərin açar sözlərinə

görə

müqayisə

etmək üçün istifadə

olunur. Nəticədə

ortaq

elementlər seçilir. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");

$a2=array("a"=>"red","b"=>"green","c"=>"blue");

$result=array\_intersect\_assoc($a1,$a2); print\_r($result);

?>

Nəticədə alınmış massiv: Array ( [a] => red [b] => green [c] => blue ) Başqa bir nümunə göstərək:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow");

$a2=array("a"=>"red","b"=>"green","g"=>"blue");

$a3=array("a"=>"red","b"=>"green","g"=>"blue");

$result=array\_intersect\_assoc($a1,$a2,$a3); print\_r($result);

?>

Nəticədə alınmış massiv: Array ( [a] => red [b] => green )

Massivlərdəki 3-cü elementlər eyni olsalar da, açar sözlər fərqli olduğuna görə 3-cü element yeni massivə daxil olmur.

1. array\_intersect\_key() funksiyası- Bu funksiya massivləri açar sözlərinə görə müqayisə etmək üçün istifadə olunur. Burada da ortaq elementlər seçilir. Məsələn:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue");

$a2=array("a"=>"red","c"=>"blue","d"=>"pink");

$result=array\_intersect\_key($a1,$a2); print\_r($result);

?>

Nəticədə alınmış yeni massiv: Array ( [a] => red [c] => blue )

Başqa bir nümunə:

<?php

$a1=array("a"=>"red","b"=>"green","c"=>"blue");

$a2=array("c"=>"yellow","d"=>"black","e"=>"brown");

$a3=array("f"=>"green","c"=>"purple","g"=>"red");

$result=array\_intersect\_key($a1,$a2,$a3); print\_r($result);

?>

Nəticədə alınmış massiv: Array ( [c] => blue )

Açar sözlər olmadıqda isə massivlər sıra nömrələrinə görə müqayisə olunur:

<?php

$a1=array("red","green","blue","yellow");

$a2=array("red","green","blue");

$result=array\_intersect\_key($a1,$a2); print\_r($result);

?>

Nəticədə alınmış massiv: Array ( [0] => red [1] => green [2] => blue )

1. array\_key\_exists() funksiyası- Bu funksiya massivdə

açar sözə

görə

bir elementin olub-

olmadığını yoxlamaq üçün istifadə istifadə edəcəyik. Məsələn:

olunur.

Funksiyanı istifadə

edərkən şərt operatorundan da

<?php

$a=array("Volvo"=>"XC90","BMW"=>"X5");

if (array\_key\_exists("Volvo",$a))

{

echo "Açar söz mövcuddur!";

}

else

{

echo "Açar söz mövcud deyil!";

}

?>

Burada ilk öncə massiv yaradılır. Sonra isə şərt operatoru vasitəsilə $a massivində Volvo açar sözünün olub-olmadığı yoxlanılır.Əgər şərt doğru olsa “Açar söz mövcuddur” çap olunacaq, əks halda isə “Açar söz mövcud” deyil sözü çap olunacaq. İndi isə açar söz olmadan elementin olub- olmamağını yoxlayaq. Bunun üçün elementin sıra nömrəsindən istifadə edəcəyik. Məsələn:

<?php

$a=array("Volvo","BMW"); if (array\_key\_exists(0,$a))

{

echo "Açar mövcuddur!";

}

else

{

echo "Açar mövcud deyil!";

}

?>

Burada isə

0-cı elementin(Proqramlaşdırmada sayma 0-dan başlayır) olub-olmamağı yoxlanılır.

Şərt doğru olduğu üçün “Açar mövcuddur” sözü çap olunacaq.

1. array\_keys() funksiyası- Mövcud massivin açar sözlərindən ibarət yeni massiv yaratmaq üçün istifadə olunur. Məsələn:

<?php

$a=array("key1"=>"element1","key2"=>"Element2","key3"=>"element3");

$b=array\_keys($a); print\_r($b);

?>

Burada $b massivi $a massivinin açar sözlərindən ibarət yeni massivdir.

1. array\_merge() funkasiyası- Bu funksiya iki və ya daha çox massivi birləşdirərək yeni massiv yaratmaq üçün istifadə olunur. Məsələn:

<?php

$a1=array("red","green");

$a2=array("blue","yellow");

$a3=array\_merge($a1,$a2); print\_r($a3);

?>

Burada $a3 massivi $a1 və $a2 massivlərinin elementlərindən ibarət yeni massiv olur.

1. array\_multisort() funksiyası- Massiv elementlərini sıralamaq üçün istifadə olunur. Elementləri müxtəlif cür sıralamaq olar.İstifadə qaydası aşağıdakı şəkildədir:

<?php

$a=array("Dog","Cat","Horse","Bear","Zebra"); array\_multisort($a);

print\_r($a);

?>

Burada massiv elementləri böyüyə sıralanır. Məsələn

əlifba sırasına görə sıralanır. Elementlər

ədədlər olduqda kiçikdən

<?php

$a=array(10,8,15,11,20);

array\_multisort($a); print\_r($a);

?>

Burada ədədlər kiçikdən böyüyə sıralanacaq.

1. array\_pop() funksiyası- Massivin son elementini silmək üçün istifadə olunur. Məsələn:

<?php

$a=array("red","green","blue");

array\_pop($a); print\_r($a);

?>

Nəticədə son element olan “blue” elementi massivdən çıxarılır.

1. array\_product() funksiyası- Bu funksiya massiv elementlərini bir-birinə etmək üçün istifadə olunur. Məsələn:

vurub nəticə əldə

<?php

$a=array(5,5,8); echo(array\_product($a));

?>

Nəticədə 200 çap olunacaq. Burada massiv yox, sadəcə bir ədəd çap olunduğuna görə print\_r funksiyasından yox, echo funksiyasından istifadə olunur.

1. array\_push() funksiyası- Massivə yeni elementlər əlavə etmək üçün istifadə olunur. Məsələn:

<?php

$a=array("red","green"); array\_push($a,"blue","yellow"); print\_r($a);

?>

Burada massivə “blue” və “yellow” elementləri əlavə olunur.

1. array\_rand() funksiyası- 20)array\_replace() funksiyası-

21)array\_reverse() funksiyası-Bu funksiya massiv elementlərini tərs qaydada düzmək üçün istifadə olunur. Məsələn:

<?php

$a=array("a"=>"Volvo","b"=>"BMW","c"=>"Toyota","d"=>"Honda");

$b=array\_reverse($a); print\_r($b);

?>

Nəticədə alınmış massiv: Array ( [d] => Honda [c] => Toyota [b] => BMW [a] => Volvo ) 22)array\_search()-Massivdəki elementin açar sözünü tapmaq üçün istifadə olunur. Məsələn:

<?php

$a=array("a"=>"red","b"=>"green","c"=>"blue"); echo array\_search("red",$a);

?>

“Red” elementinin açar sözü “a” olduğu üçün “a” çap olunacaq.

1. array\_shift() funksiyası-Massivdəki ilk elemnti silmək üçün istifadə olunur. Məsələn:

<?php

$a=array("a"=>"red","b"=>"green","c"=>"blue"); array\_shift($a);

print\_r ($a);

?>

Nəticədə alınmış massiv: Array ( [b] => green [c] => blue ) Funksiyanı indeks nömrələri ilə olan massiv üçün istifadə edək:

<?php

$a=array(0=>"red",1=>"green",2=>"blue"); array\_shift($a);

print\_r ($a);

?>

Nəticədə alınmış massiv: Array ( [0] => green [1] => blue )

Eyni zamanda ilk elementi silərkən həmin elementi ekrana çap edə bilərik. Məsələn:

<?php

$a=array("a"=>"red","b"=>"green","c"=>"blue"); echo array\_shift($a);

print\_r ($a);

?>

Burada massivlə yanaşı silinən elementdə ekrana çap olunur.

1. array\_slice() funksiyası-Hazırki massivin bir hissəsindən ibarət yeni massiv yaratmaq üçün istifadə olunur. Məsələn:

<?php

$a=array("red","green","blue","yellow","brown");

$b=array\_slice($a,2); print\_r($b);

?>

Burada $b massivi $a massivinin 2-ci elementi də daxil olmaqla(proqramlaşdırmada sayma 0-dan

başladığı üçün 2-ci element dedikdə

3-cü yerdə

duran element nəzərdə

tutulur) növbəti

elementlərindən ibarət yeni massivdir.Yuxarıdakı kimi yazdıqda $b massivinə $a massivinin 2- cidən başlayaraq bütün elementləri daxil ola bilər. Ancaq biz 2-ci elementdən başlayaraq istədiyimiz sayda elementi yeni massivə daxil edə bilərik. Məsələn:

<?php

$a=array("red","green","blue","yellow","brown");

$b=array\_slice($a,2,1); print\_r($b);

?>

Belə yazdıqda 2-ci elementdən başlayaraq yalnız 1 element yeni massivə daxil olacaq. Sıralamağa əvvəldən başladığımız kimi axırdan da başlaya bilərik Məsələn əgər biz axırdan 2-ci elementdən başlayaraq elementləri seçmək istəyiriksə onda aşağıdakı kimi yazmalıyıq

<?php

$a=array("red","green","blue","yellow","brown");

$b=array\_slice($a,-2,2); print\_r($b);

?>

Nəticədə $b massivi $a massivinin sondan 2-ci elementindən başlayaraq 2 elementindən ibarət

olur. -1-ci element dedikdə sonuncu element, -2-ci element dedikdə nəzərdə tutulur.

isə

sondan ikinci element

1. array\_sum() funksiyası-Ədədlərdən ibarət massivin elementlərinin cəmini tapmaq üçün istifadə olunur. Məsələn:

<?php

$a=array(5,15,25); echo array\_sum($a);

?>

Nəticədə ekrana 45 çıxacaq.

1. array\_unique() funksiyası-Massivdə olan eyni elementləri silmək üçün istifadə olunur massivdə bir neçə eyni element varsa birincisi saxlanılır,digərləri isə silinir. Məsələn:

Əgər

<?php

$a=array("a"=>"red","b"=>"green","c"=>"red","e"=>"red");

$b=array\_unique($a); print\_r($b);

?>

Nəticədə alınmış $b massivi: Array ( [a] => red [b] => green )

1. array\_unshift() funksiyası-Massivə yeni element daxil etmək üçün istifadə olunur. Məsələn:

<?php

$a=array("a"=>"red","b"=>"green"); array\_unshift($a,"blue"); print\_r($a);

?>

Nəticədə $a massivinə

“blue” elementi daxil olur.

Aşağıdakı kimi yazdıqda isə

element a

massivinə daxil olduqdan sonra a massivinin element sayı çap olunur:

<?php

$a=array("a"=>"red","b"=>"green"); echo array\_unshift($a,"blue");

echo "<br>"; print\_r($a);

?>

Son sətirdə

isə

print\_r() funksiyası vasitəsilə

yeni a massivi çap olunur. Funksiyanı istifadə

etdikdə yeni element massivin ilk elementi olur. Elementlər indeks nömrələri ilə olduqda yenə

yeni element massivin ilk elementi olur və 0 indeksin alır. Məsələn:

<?php

$a=array(0=>"red",1=>"green"); array\_unshift($a,"blue"); print\_r($a);

?>

Burada “blue” elementi 0 indeksini aldı və digər elemntlərin indeksləri bir vahid artdı.

1. array\_values() funksiyası-Massiv elementlərini indeks nömrələri ilə almaq üçün istifadə olunur Məsələn:

<?php

$a=array("Name"=>"Peter","Age"=>"41","Country"=>"USA");

$b=array\_values($a); print\_r($b);

?>

Nəticədə $a massivindəki elemntlərin açar sözləri olsa da, $b massivindəki elementlər indeks nömrələri ilə olur.

1. Current() funksiyası-Massivdəki ilk elementi seçmək üçün istifadə olunur, Məsələn:

<?php

$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo current($people) . "<br>";

?>

Nəticədə massivin ilk elementi çap olunur,

1. End() funksiyası-Massivin son elementini seçmək üçün istifadə olunur, Məsələn:

<?php

$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo end($people);

?>

Nəticədə massivin son elementi çap olunacaq,

1. Extract() funksiyası-Massivdəki açar sözlərlə adlanan, uyğun olaraq massiv elementlərinə

bərabər olan dəyişənlər yaratmaq üçün istifadə olunur, Məsələn:

<?php

$array=array("key1"=>"ilkelement","key2"=>5,"key3"=>"Sonuncu"); extract($array);

echo $key1."<br>".$key2."<br>".$key3;

?>

Burada $key1,$key2,$key3 adlı dəyişənlər yaranır və uyğun dəyərlərinə bərabər olur. 32)In\_array() funksiyası-Bir elementin massivdə olub olmadığını yoxlamaq üçün istifadə olunur.

Məsələn:

<?php

$people = array("Peter", "Joe", "Glenn", "Cleveland"); if (in\_array("Joe", $people))

{

echo "Element tapıldı";

}

else

{

echo "Element tapılmadı";

}

?>

Burada “Joe” elementi massivdə olduğu üçün Ekrana “Element tapıldı” yazısı çap olunacaq. 33)List() funksiyası-Massiv elementlərini dəyişənlərə köçürmək üçün istifadə olunur. Məsələn:

<?php

$people = array("Peter", "Joe", "Glenn", "Cleveland"); list($a,$b,$c,$d)=$people;

echo $a."<br>".$b."<br>".$c."<br>".$d;

?>

Burada uyğun olaraq massiv elementlərinə bərabər olan $a, $b, $c, $d dəyişənləri yaradıldı.

Massivin sadəcə bir neçə elementinə aid də dəyişənlər yaratmaq mümkündür. Məsələn:

<?php

$people = array("Peter", "Joe", "Glenn", "Cleveland"); list($a,$b)=$people;

echo $a."<br>".$b;

?>

Burada yalnız massivin ilk iki elementinə uyğun dəyişənlər yaradıldı və çap olundu. 34)Next() funksiyası-Massivin növbəti elementlərini seçmək üçün istifadə olunur. Məsələn:

<?php

$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo next($people);

?>

Burada ikinci elemnt çap olunacaq. Məsələn 4-cü elementi çap etmək istəsək, 3 dəfə funksiyasından istifadə etməliyik:

next

<?php

$people = array("Peter", "Joe", "Glenn", "Cleveland"); next($people);

next($people); echo next($people);

?>

Nəticədə 4-cü element çap olunacaq.

1. Prev() funksiyası-əvvəlki elementi seçmək üçün istifadə olunur. Məsələn:

<?php

$people =array("Peter", "Joe", "Glenn", "Cleveland"); echo next($people) . "<br>";

echo prev($people);

?>

Burada ilk öncə next() funksiyası vasitəsilə növbəti element, yəni, ikinci element seçilərək çap

edilir Növbəti addımda isə element çap edilir.

prev() funksiyası vasitəsilə

bir vahid

əvvəlki element ,yəni, 1-ci

1. Range() funksiyası-seçilmiş iki ədəd arasındakı ədələrdən ibarət yeni massiv yaratmaq üçün istifadə olunur. Məsələn:

<?php

$number = range(0,5); print\_r ($number);

?>

Nəticə: Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 )

1. Reset() funksiyası-Massivi yeniləmək üçün istifadə olunur. Məsələn:

<?php

$people = array("Peter", "Joe", "Glenn", "Cleveland"); echo next($people) . "<br>";

echo next($people) . "<br>"; echo reset($people);

?>

Burada iki dəfə next() funksiyasını işlədərək 3-üncü elementə getmiş oluruq. Ancaq sonra reset() funksiyasını işlədərək yenidən 1-inci elementə qayıdırıq.

1. Shuffle() funksiyası-Massiv elementlərini təsadüfi sıralamaq üçün istifadə olunur. Məsələn:

<?php

$my\_array = array("red","green","blue","yellow","purple"); shuffle($my\_array);

print\_r($my\_array);

?>

Burada hər dəfə səhifə yeniləndikdə massiv elementləri təsadüfi olaraq yerini dəyişir.

# Funksiyalar

Yuxarıdakı mövzularda biz bir çox riyazi funksiyalar, massiv funksiyaları ilə tanış olduq. Növbəti

mövzularda da bir çox funksiyalarla tanış olacağıq. Bu funksiyalardan

əlavə

biz özümüz də

funksiyalar yarada bilərik. Yaradacağımız yeni funksiyalar çox sadə və ya çox mürəkkəb

əməliyyatları yerinə yetirəcək. Funksiyalar proqramçı üçün çox əlverişlidir. Fərz edək ki, saytda bir eyni bir kodu bir neçə səhifədə yazmalıyıq. Əgər bu kod uzundursa çətinlik yaranacaq. Ancaq

funksiya vasitəsilə

həmin kodu bir dəfə

yazıb bir çox yerlədrə

işlədə

bilərik. Funksiyanın

yaradılma qaydası aşağıdakı kimidir:

<?php

function funksiya\_adı(arqumentlər){

//Əməliyyatlar

}

?>

Burada arqumentlər hissəsinə istifadə olunacaq arqumentlər, əməliyyatlar hissəsinə isə lazımi əməliyyatları yazmalıyıq. İlk öncə arqumentsiz funksiyalarla tanış olaq. Sadəcə ekrana yazı yazan bir funksiya yaradaq:

<?php

function funksiya(){ echo "First function";

}

?>

Burada artıq funksiya yaradıldı. Ancaq burada ekrana heç yazılmayacaq çünki biz funksiyanı çağırmamışıq. Funksiyanı işlətmək üçün əvvəlcə funksiyanı çağırmalıyıq:

<?php

function funksiya(){ echo "First function";

}

funksiya();

?>

Nəticədə funksiya öz işini görür və ekrana yazı yazılır. Burada funksiyanı bir neçə dəfə çağıraraq funksiya ənəliyyatlarını bir neçə dəfə yerinə yetirə bilərik.

Bu yazdığımız arqumentsiz funksiya idi. İndi isə arqumentli funksiyalarla tanış olaq. Arqumentli

funksiya yaradarkən funksiyada mötərizə

daxilində

arqumentləri daxil edirik və

funksiya

daxilindən həmin arqumentlər üzərində

müəyyən

ənəliyyatlar aparırırıq. Funksiyalar

çağırıldıqda isə arqumentlər müəyyən dəyərlərlə əvəz olunur və əməliyyatlar həmin dəyərlər üzərində aparılır. Məsələn:

<?php

function funksiya($a){ echo $a;

}

funksiya("First variable");

?>

Burada $a arqument olur və sonra funksiya çağırıldıqda arqument dəyərlə əvəzlənir. Nəticədə arqumentin yerinə yazdığımız dəyər çap olunur. İndi isə bir neçə arqumentli funksiya yazaq:

<?php

function funksiya($a,$b,$c){ echo $a+$b+$c;

}

funksiya(5,4,8);

?>

Burada $a,$b və

$c dəyişən olur. Sonra funksiya çağırıldıqda Arqumentlər

ədədlərlə əvəz

olunur. Indi isə bu funksiyanı müxtəlif dəyərlərlə bir neçə dəfə işlədək:

<?php

function funksiya($a,$b,$c){ echo $a+$b+$c."<br>";

}

funksiya(4,5,8);

funksiya(1,2,3);

funksiya(7,8,5);

?>

Nəticədə müxtəlif qiymətlər çap olunur. Yazdığımız funksiya daxilində digər hazır funksiyalarıda işlədə bilərik. Məsələn:

<?php

function funksiya($a){ echo sqrt($a);

}

funksiya(4);

?>

Nəticədə

2 çap olunur. Ancaq bir şeyi nəzərə

almaq lazımdır ki funksiya daxilində

echo ilə

ekrana nəticəni yazdığımızda, sonra biz həmin funksiyanı çağırdıqda nəticəni ədəd kimi istifadə

edə

bilməyəcəyik. Bunun səbəbi echo funksiyasının nəticəni sadəcə

ekrana yazmasıdır.

Aşağıdakı koda baxaq:

<?php

function funksiya($a,$b){ echo $a+$b;

}

$a=funksiya(5,4);

$b=funksiya(3,2); echo $a+$b;

?>

Burada biz düzgün nəticə ala bilməyəcyik, çünki, nəticədə sadəcə 9 və 5 yan-yana yazılacaq.

Əgər funksiya daxilindəki əməliyyatlardan sonra alınmış nəticə üzərində əməliyyatlar aparmaq

istəyiriksə, funksiya daxilində

echo yox, return istifadə

edərək müəyyən bir dəyər

qaytarmalıyıq. Bu halda nəticə üzərində əməliyyat aparmaq mümkün olacaq. Məsələn:

<?php

function funksiya($a,$b){ return $a+$b;

}

$a=funksiya(5,4);

$b=funksiya(3,2); echo $a+$b;

?>

Burada isə funksiyadan alınan qiymətlər üzərində əməliyyatlar aparmaq mümkün olur, yəni

funksiya konkret bir qiymətə bərabər olur. Funksiyaya aid bir neçə nümunə yazaq.

Nümunə 1: Əgər bir arqumentli funksiyada arqument müsbətdirsə 1 dəyərini, əks halda isə 0

dəyərini qaytaraq, Sonra isə olacaq:

bu dəyəri yoxlayaraq ekrana yazı yazdıraq. Kod aşağıdakı kimi

<?php

function funksiya($a){ if($a>0){

return 1;

}

else{ return 0;

}

}

$check=funksiya(5); if($check==1){ echo "Musbet";

}

else{

echo "Menfi ve ya sifir";

}

?>

Kodu izah edək. İlk öncə funksiyanı elan etdik və $a arqumentini təyin etdik. Funksiya daxilində if operatoru vasitəsilə bu arqumenin 0-dan böyük olub-olmamasını yoxladıq. Əgər şərt doğru olsa

1 qiyməti qayıdacaq.

Əks halda isə

0 qiyməti qayıdaaqdı. Funksiya sona çatır. Deməli

əgər

arqument 0-dan böyük olsa 1, 0-dan kiçik olsa 0 qiyməti qayıdacaq, Növbəti addımda isə

funksiyaya 5 arqumentini verərək $check dəyişəninə bilərik ki, 5 ədədi

mənimsədirik. Buradan asanlıqla görə

1. dan böyük olduğu üçün funksiyada 1 dəyəri qayıdacaq və $check dəyişəni 1-ə bərabər olacaq. Növbəti olaraq isə $check dəyişənini yoxlayaraq ekrana müvafiq yazını yazırıq.

Nümunə

2: Fərz edək ki funksiya dəyəri olaraq 1-7 arası

ədədlərdən biri verilir və

biz bu

qiymətə əsasən həftənin gününü ekrana yazmalıyıq. Burada şərt çox olduğu üçün switch

operatorundan istifadə

edəcəyik. Burada sadəcə

ekrana yazma olacaq deyə

return istifadə

etməyəcəyik. Kod aşağıdakı kimi olacaq:

<?php

function funksiya($a){ switch($a){

case 1: echo "Birinci gün"; break;

case 2: echo "İkinci gün"; break;

case 3: echo "Üçüncü gün"; break;

case 4: echo "Dördüncü gün"; break;

case 5: echo "Beşinci gün"; break;

case 6: echo "Altıncı gün"; break;

case 7: echo "Bazar günü"; break;

default: echo "Daxiletme sehvdir!"; break;

}

}

funksiya(3);

?>

Nəticədə "Üçüncü gün" sözü çap olunacaq.

Nümunə

3: Funksiya vasitəsilə

1. dən arqumentə

qədər

ədələri çap edək. Bunun üçün for

operatorundan istifadə edəcəyik. Kod aşağıdakı kimi olacaq:

<?php

function funksiya($a){ for ($i=1;$i<=$a;$i++) echo $i."<br>";

}

funksiya(10);

?>

Nəticədə 1-dən 10-a kimi ədələr alt-alta çap olunacaq.

**Isset(), Empty() və Unset() funksiyaları**

Bir dəyişənin dəyərə malik olub-olmadığını yoxlamaq üçün isset() funksiyasından istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$a=5; if(isset($a)){

echo "Dəyişən mövcuddur!";

}

?>

Isset() funksiyasının tam əksi isə unset() funksiyasıdır. Əgər dəyişən dəyərə malik deyilsə doğru sayılır.

<?php

$a=5; if(empty($a)){

echo "Dəyişən mövcud deyil!";

}

else{

echo "Dəyişən mövcuddur!";

}

?>

Dəyişəni silmək üçün unset() funksiyasından istifadə olunur.

<?php

$a=5; unset($a); if(empty($a)){

echo "Dəyişən mövcud deyil";

}

else{

echo "Dəyişən mövcuddur";

}

?>

Burada unset() funksiyası ilə dəyişən silindiyinə görə şərt doğru olacaq və «Dəyişən mövcud deyil» yazısı çap olunacaq.

# Form daxili əməliyyatlar. Post və Get metodları

Yaradılmış formlara istifadəçi tərəfindən daxil edilmiş məlumatları götürmək və başqa vacib

məlumatları əldə etmək üçün Post və Get metodlarından istifadə olunur. İlk öncə Get metodu ilə tanış olaq.

Get metodu. Bu metodla formdan məlumatları əldə etmək üçün yaratdığımız formanın method

atributuna «Get» yazmalıyıq. Əvvəlcə formanı yaradaq:

<form method="Get">

Adınız: <input type="text" name="ad">

<br>

<input type="submit" value="Gonder">

</from>

Formanı hazırladıq. Əməliyyatlar eyni səhifədə baş verəcəyi üçün formada action daxil etmirik. Burada yaratdığımız inputların adlarına xüsusi fikir vermək lazımdır, çünki, inputlara bu adlarla müraciət edəcəyik. Formanı yaratdıq. İndi isə inputa daxil olunmuş yazını ekrana yazmağa çalışaq. İnputa daxil olunmuş yazı, Get metodu ilə aşağıdakı şəkildə götürülür:

$a=$\_GET['inputun\_adi'];

Burada inputa daxil edilmiş yazı $a dəyişəninə mənimsədilmiş olur. Ümumi kod isə bu formada olacaq:

<form method="Get">

Adınız: <input type="text" name="ad">

<br>

<input type="submit" value="Gonder">

</from>

<?php

$a=$\_GET['ad']; echo $a;

?>

Nəticədə inputa daxil olunmuş yazı çap olunacaq. Biz burada birbaşa yazını çap etdik. Ancaq

əslində

göndər düyməsinə

basıldıqdan sonra inputun dəyərini götürməliyik. Bunun üçün isə

isset() funksiyası vasitəsilə göndər düyməsinə basılmağını yoxlamalıyıq. Yuxarıdıkı php kodunda aşağıdakı dəyişikliyi edək:

<?php if(isset($\_GET['daxil'])){

$a=$\_GET['ad']; echo $a;

}

?>

Burada artıq inputa daxil edilmiş yazı Göndər basıldıqdan sonra əldə olunacaq və çap olunacaq.

Get metodunun mənfi bir xüsusiyyəti var ki, get metodundan istifadə

etdikdə

formaya daxil

edilmiş dəyərlər brauzerin adres hissəsində görünür. Məsələn, yuxarıdakı kodda Göndər

düyməsinə basdıqda brauzerdə adres hissəsi aşağıdakı şəkildə olur:

**1.php?ad=Daxiletme&daxil=Gonder**

Belə

olduqda isə

təhlükəsizlik cəhətdən problemlər yaranar bilər. Bu səbəbdən formalarda

digər metod olan post metodundan istifadə

etmək daha

əlverişlidir. Ancaq bu yazıların adres

hissəsinə yazılmağından biz başqa yerlərdə də istifadə edə bilərik, Məsələn fərz edək ki, sayta

3 xəbər çap etmişik. Bu xəbərləri silmək üçün bizə əlavə

iki səhifə

lazım olacaq. Birinci

səhifədə xəbərlərin siyahısı və silmə linki olacaq. İkinci səhifədə isə silmə əməliyyatı yerinə yetiriləcək. Birinci səhifəmiz:

<a href="sil.php?xeber=1">Sil</a>

<br>

<a href="sil.php?xeber=2">Sil</a>

<br>

<a href="sil.php?xeber=3">Sil</a>

İkinci səhifəmizin adı isə sil.php-dir. Sil.php səhifəsinə silmək istədiyimiz xəbərə uyğun ədəd göndəririk. Sil.php səhifəsində isə aşağıdakı kod yazılacaq və ədədə uyğun xəbər silinəcək:

<?php

echo $\_GET['xeber'];

// Silmə əməliyyatları

?>

Bu cür əməliyyatlarla irəlidəki mövzularda tanış olacağıq. İndi isə Post metodu ilə tanış olaq.

Post metodu. Bu metodun istifadə qaydası Get metodu ilə eynidir. Üstün cəhəti isə formalarda

daxil edilmiş məlumatlar brauzerin adres hissəsində problemləri yaranmır. Post metoduna aid bir nümunə:

göstərilmir. Bu zaman istə

təhlükəsizlik

<form method="POST">

Adınız: <input type="text" name="ad">

<br>

<input type="submit" value="Gonder">

</from>

<?php

$a=$\_POST['ad']; echo $a;

?>

Burada məlumat adres hissəsinə yazılmadan səhifədə çap olunacaq.

Dəyişənə

# Filter funksiyaları

hər hansı bir məlumatı ötürdükdən sonra həmin məlumatın doğruluğunu yoxlamaq

lazımdır. Bu təhlükəsizlik baxımından vacıbdir. Bir sıra filter funksiyaları mövcuddur. Bunlar aşağıdakılardır:

1. filter\_var() funksiyası-Hər hansı bir məlumatın doğru olub-olmadığını yoxlamaq üçün istifadə olunur. Məsələn aşağıda emailin doğruluğu yoxlanılır:

<?php

$email = ["exa](mailto:example@example.com)m[ple@example.com](mailto:example@example.com)";

if (filter\_var($email, FILTER\_VALIDATE\_EMAIL)) { echo "Valid!";

}

else{

echo "Invalid!";

}

?>

Funksiyada emaili yoxlamaq üçün FILTER\_VALIDATE\_EMAIL ifadəsi işlənir. filter\_var() funksiyası ilə emaildən əlavə bir çox ifadələri yoxlamaq mümkündür.

İfadənin tam ədəd olub-olmadığını yoxlamaq üçün:

<?php

$a=123;

if (filter\_var($a, FILTER\_VALIDATE\_INT)) { echo("True");

} else { echo("False");

}

?>

Burada $a dəyişəni tam ədəd olduğu üçün True sözü çap olunacaq. 2.İfadənin məntiqi tip olub-olmadığını yoxlamaq üçün:

<?php

$a=true;

if (filter\_var($a, FILTER\_VALIDATE\_BOOLEAN)) { echo("True");

} else { echo("False");

}

?>

İfadə məntiqi olduğu üçün true sözü çap olunur. 3.İfadənin onluq ədəd olub-olmadığını yoxlamaq üçün:

<?php

$a=5.4;

if (filter\_var($a, FILTER\_VALIDATE\_FLOAT)) {

echo("True");

} else { echo("False");

}

?>

Nəticədə true sözü çap olunacaq. Tam ədədlər də buraya daxildir. 4.İfadənin URL ünvan olub-olmadığını yoxlamaq üçün:

<?php

$a=["ht](http://example.com/)t[p://example.com/](http://example.com/)";

if (filter\_var($a, FILTER\_VALIDATE\_URL)) { echo("True");

} else { echo("False");

}

?>

5.İfadənin Email olub-olmadığını yoxlamaq üçün:

<?php

$a=["e](mailto:email@email.com)m[ail@email.com](mailto:email@email.com)";

if (filter\_var($a, FILTER\_VALIDATE\_EMAIL)) { echo("True");

} else { echo("False");

}

?>

6.İfadənin İP ünvan olub-olmadığını yoxlamaq üçün:

<?php

$a="1.1.1.1";

if (filter\_var($a, FILTER\_VALIDATE\_IP)) { echo("True");

} else { echo("False");

}

?>

1. Email olaraq təqdim olunan ifadədən uyğun olmayan simvolları silmək üçün:

<?php

$email = "john(.doe)@exa//mple.com";

$email = filter\_var($email, FILTER\_SANITIZE\_EMAIL); echo $email;

?>

Nəticə: [john.doe@example.com](mailto:john.doe@example.com)

1. Aşağıdakı filterləmədə addslashes() funksiyası ilə eyni iş görülür:

<?php

$var="Peter's here!";

echo(filter\_var($var, FILTER\_SANITIZE\_MAGIC\_QUOTES));

?>

Nəticə: Peter\'s here!

1. Onluq ifadədən uyğun olmayan simvolları silmək üçün:

<?php

$number="5-2f+3.3pp";

echo(filter\_var($number, FILTER\_SANITIZE\_NUMBER\_FLOAT, FILTER\_FLAG\_ALLOW\_FRACTION));

?>

Nəticə: 5-2+3.3

Funksiyada üçüncü parametr olaraq FILTER\_FLAG\_ALLOW\_FRACTION daxil etdik. Bu parametr ifadədə nöqtələrin qalmasına icazə verir. Bu ifadənin yerinə başqa iki ifadə də yaza

bilərik. FILTER\_FLAG\_ALLOW\_THOUSAND ifadəsi vergülləri saxlamağa icazə Məsələn:

verir.

<?php

$number="5-2f+4,3pp";

echo(filter\_var($number, FILTER\_SANITIZE\_NUMBER\_FLOAT, FILTER\_FLAG\_ALLOW\_THOUSAND));

?>

Nəticə: 5-2+4,3

FILTER\_FLAG\_ALLOW\_SCIENTIFIC ifadəsi isə ifadədə e və E simvollarını saxlamağa icazə verir. Məsələn:

<?php

$number="5-2f+4epp";

echo(filter\_var($number, FILTER\_SANITIZE\_NUMBER\_FLOAT, FILTER\_FLAG\_ALLOW\_SCIENTIFIC));

?>

Nəticə: 5-2+4e

1. Tam ifadədən uyğun olmayan simvolları silmək üçün:

<?php

$number="5-2+3.5pp";

echo(filter\_var($number, FILTER\_SANITIZE\_NUMBER\_INT));

?>

Nəticə: 5-2+35

1. Aşağıdakı filtrləmədə isə ifadədə olan xüsusi xarakterlər(məsələn html teqləri) adi söz kimi nəzərə alınır:

<?php

$url="Is Peter <smart> & funny?"; echo(filter\_var($url,FILTER\_SANITIZE\_SPECIAL\_CHARS));

?>

Nəticə: Is Peter <smart> & funny?

1. Aşağıdakı filtrləmədə mətndə olan html teqlər silinir:

<?php

$str = "<h1>Hello World!</h1>";

$newstr = filter\_var($str, FILTER\_SANITIZE\_STRING); echo $newstr;

?>

Nəticə: Hello World!

1. URL ünvandan uyğun olmayan ifadələri silmək üçün:

<?php

$var="http://www.w3schoo��ls.co�m"; echo(filter\_var($var, FILTER\_SANITIZE\_URL));

?>

Nəticə: [http://www.w3schools.com](http://www.w3schools.com/)

1. filter\_list() funksiyası-Filter funksiyalarında dəstəklənən ifadələri özündə verir. Məsələn:

saxlayan massivi

<?php print\_r(filter\_list());

?>

1. filter\_id() funksiyası-Xüsusi filter adının İD nömrəsini göstərən funskiyadır. Məsələn:

<?php

$echo(filter\_id("validate\_email"));

?>

Nəticədə 274 çap olunur.

# Cookies

Php dilində

bir məlumatı istifadəçinin brauzerinə

yazaraq oradan oxumaq mümkündür. Bu

əməliyyatlar cookie-lər vasitəsilə yerinə yetirilir. Belə ki, cookie yaradılarkən cookie-nin qalma vaxtı təyin olunur və həmin müddət boyunca cookie istifadəçinin brauzerində qalır. Cookie-lər ən çox saytın giriş bölməsində «yadda saxla» əməliyyatı üçün istifadə olunur. Bir şeyi də nəzərə almaq lazımdır ki, istifadəçi istədiyi vaxt brauzerdən həmin cookie-ləri silə bilər. Eyni zamanda

əgər cookie A brauzerinə

yazılıbsa, B brauzeri ilə

sayta daxil olduqda həmin cookie B

brauzerində kimidir:

olmadığı üçün istifadə oluna bilməyəcək. Cookie-nin yaradılma qaydası aşağıdakı

<?php setcookie("cookie","value",time()+3600);

?>

Burada ilk parametr yaradacağımız cookie-nin adıdr. İkinci parametr cookie-nin dəyəridir. Üçüncü parametr isə cookie-nin brauzerdə qalacağı vaxtdır. Üçüncü parametr saniyə ilə təyin olunur.

Time()+3600 yazaraq cookie-nin qalma vaxtını 1 saat təyin edirik(1 saat=3600 saniyə). Əgər

cookie-nin bir gün aktiv qalmağını istəyiriksə onda time()+3600\*24 yazmağımız kifayətdir. Eynilə olaraq cookie-nin qalma vaxtını daha da uzada bilərik. Yaradılmış cookie aşağıdakı qaydada çağırılır:

<?php setcookie("cookie","value",time()+3600); echo $\_COOKIE['cookie'];

?>

Nəticədə cookie-nin dəyəri çap olunur. $\_COOKİE[''] daxilində cookie-nin adı olmalıdır.

Əvvəlcədən yaradılmış cookie-ni silmək üçün zaman hissəsinə Beləliklə cookie-nin vaxtı dolmuş sayılacaq.

keçmiş saatı yazmalıyıq.

<?php

setcookie("cookie", "value", time() - 3600);

?>

Beləliklə yaradılmış cookie-nin vaxtı dolmuş sayılır. Bir cookie-nin olub olmadığını isset()

funksiyası ilə yoxlaya bilərik:

<?php setcookie("cookie","value",time()+3600); if(isset($\_COOKIE['cookie'])){

echo "Cookie mövcuddur!";

}

?>

# Sessiyalar

Sessiyalar istifadəçinin brauzeri açıq saxladığı müddətcə

aktiv olur və

istifadə

oluna bilər.

Brauzer bağlandıqda sessiya itir. Sessiyanın yaradılma və çağırılma qaydası aşağıdakı kimidir:

<?php session\_start();

$\_SESSION['name']="Php"; echo $\_SESSION['name'];

?>

Sessiya istifadə olunan səhifənin əvvəlində mütləq session\_start(); yazılmalıdır. Sessiyanı silmək üçün unset() funksiyasından istifadə olunur.

<?php session\_start();

$\_SESSION['name']="Php"; unset($\_SESSION['name']); if(isset($\_SESSION['name'])){ echo "Sessiya mövcuddur!";

}

else{

echo "Sessiya mövcud deyil!";

}

?>

Nəticədə sessiya boş olduğu üçün heç nə çap olunmayacaq. Sessiyanın olub-olmadığını yoxlamaq üçün isset() funksiyasından istifadə etməliyik:

<?php session\_start();

$\_SESSION['name']="Php"; if(isset($\_SESSION['name'])){ echo "Sessiya mövcuddur!";

}

?>

Sessiyalar ən əsas saytda istifadəçi girişinin proqramlaşdırılmasında istifadə olunur və ən vacib mövzulardan biridir. İrəlidəki mövzularda sessiyaların geniş istifadəsilə tanış olacağıq.

# Zaman funksiyaları

Php dilində zaman funksiyaları mövcuddur. Bu funksiyalar vasitəsilə saatı, tarixi əldə etmək və s.

Müxtəlif formalarda

əldə

etmək mümkündür. Bəzi zaman funksiyalarını işlətməzdən

əvvəl

date\_default\_timezone\_set('Asia/Baku'); funksiyasını işlətmək lazımdır. Yoxsa nəticə səhv olacaq.

Ən vacib zaman funksiyaları aşağıdakılardır:

1. Date() funksiyası-Bu funksiya vasitəsilə

hal hazırki tarixi və

saatı

əldə

etmək

mümkündür.Məsələn funksiya daxilində «d» yazmaqla hal-hazırkda ayın neçəsi olduğunu əldə

edə bilərik:

<?php date\_default\_timezone\_set('Asia/Baku'); echo date("d");

?>

Nəticədə ayın neçəsi olduğu çap olunur. Gündən əlavə digər vaxtları da əldə etmək mümkündür. Bunun üçün funksiya daxilində nəyi əldə etmək istəyiriksə ona müvafiq hərfi yazmalıyıq. Ancaq burada böyük və kiçik hərflər fərqlənir. Bunlar aşağıdakılardır:

|  |  |  |
| --- | --- | --- |
| Hərf | İzahı | Nümunə |
| d | Ayın gününü əldə etmək üçün istifadə olunur. | 27 |
| m | Hansı ayda olduğumuzu göstərir(rəqəmlə). | 3 |
| Y | Hansı ildə olduğumuzu göstərir. | 2015 |
| H | Saatı göstərir. | 14 |
| i | Dəqiqəni göstərir. | 21 |
| s | Saniyəni göstərir | 51 |
| l | Həftənin gününün adını ingiliscə verir. | Saturday |
| F | Ayın adını ingiliscə verir. | Oct |
| D | Günün adını ilk 3 hərfini verir(İngiliscə) | Sat |
| N | Həftədəki günün nömrəsini verir. Məsələn Bazar ertəsi üçün 1, şənbə üçün 6 | 1.5 |
| U | 1970-ci ildən günümüzə qədər keçən vaxtı saniyə ilə verir. | 1445084708 |

Bunlardan

əlavə

hərflər də

mövcuddur, ancaq

ən əsasları yuxarıdakı cədvəldə

göstərilib.

Funksiyanın müxtəlif hərflərlə işlənmə qaydası aşağıda göstərilib:

<?php date\_default\_timezone\_set('Asia/Baku'); echo date("d-m-y H:i:s");

?>

1. time() funksiyası-1970-ci ildən indiyə qədər keçən vaxtı saniyə ilə verir.

<?php date\_default\_timezone\_set('Asia/Baku'); echo time();

?>

1. getdate()-əsas zaman parametrləri daxil olan bir massiv yaradır. Bu massivdən lazımi tarixi, saatı və s. Əldə edə bilərik.

<?php date\_default\_timezone\_set('Asia/Baku');

$a=getdate();

echo $a['seconds'];

?>

$a['seconds'] yazaraq hal-hazırki saniyəni əldə edirik. Uyğun sözü yazaraq digər tarixləridə əldə

edə bilərik. Bu sözlər aşağıdakılardır:

|  |  |
| --- | --- |
| Hərf | Funksiyası |
| seconds | Hal-hazırki saniyəni əldə etmək üçündür. |
| minutes | Hal-hazırki dəqiqəni əldə etmək üçündür. |
| hours | Hal-hazırki saatı əldə etmək üçündür. |
| mday | Ayın gününü əldə etmək üçündür. |
| wday | Həftənin gününü əldə etmək üçündür. |
| mon | Ayı əldə etmək üçün istifadə olunur. |
| year | İli əldə etmək üçün istifadə olunur. |
| yday | İlin gününü əldə etmək üçün istifadə olunur. |
| weekday | Həftənin günrıldə etmək üçün istifadə olunur. |
| month | Ayın adını əldə etmək üçün istifadə olunur. |
| 0 | 1970-ci ildən günümüzə qədər olan vaxtı saniyə ilə verir. |

Php dilində

yazılar üzərində

# String funksiyaları

dəyişikliklər aparmaq üçün bir sıra funksiyalar mövcuddur. Bu

funksiyalar aşağıdakılardır:

1. addslashes() funksiyası-Bu funksiya vasitəsilə

yazıdakı dırnq işarəsinin

əvvəlinə

avtomatik

olaraq \ işarəsi qoyulur. Bazaya məlumat göndərərkən təhlükəsizlik məqsədilə bu funksiyadan istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$str = addslashes('What does "yolo" mean?'); echo($str);

?>

Nəticə:

What does \"yolo\" mean?

1. bin2hex() funksiyası-Yazını 16-lıq say sisteminə çevirir. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$str = bin2hex("Hello World!"); echo($str);

?>

1. chunk\_split() funksiyası-Yazıdakı xarakterləri bir neçə olunur. Məsələn:

simvoldan bir ayırmaq üçün istifadə

<?php

$str = "Hello world!";

echo chunk\_split($str,1,".");

?>

Nəticə:

H.e.l.l.o. .w.o.r.l.d.!.

Burada hər bir simvoldan bir araya nöqtə Qoyulur. Nöqtənin əvəzinə digər simvollar da ola bilər.

1. htmlspecialchars() funksiyası-Bu funksiya vasitəsilə yazı daxilindəki html kodlar kod kimi yox sadəcə yazı kimi nəzərə alınır. Məsələn:

<?php

$str = "This is some <b>bold</b> text."; echo htmlspecialchars($str);

?>

Nəticə:

This is some <b>bold</b> text.

1. lcfirst() funksiyası-Bu funksiya yazının ilk hərfini kiçiltmək üçün istifadə olunur. Məsələn:

<?php

echo lcfirst("Hello world!");

?>

Nəticə: hello world!

1. ltrim() funksiyas-Yazının sol tərəfindən qeyd olunmuş hissəni silmək üçün istifadə Məsələn:

olunur.

<?php

$str = "Hello World!"; echo $str . "<br>";

echo ltrim($str,"Hello");

?>

Nəticə: World!

1. rtrim() funksiyası-Yazının sağ tərəfindən qeyd olunmuş hissəni silmək üçün istifadə olunur. Məsələn:

<?php

$str = "Hello World!"; echo $str . "<br>";

echo rtrim($str,"World!");

?>

Nəticə: Hello

1. md5() funksiyası-Yazını mdt şifrələmə alqoritmi ilə şifrələmək üçün istifadə olunur. Məsələn:

<?php

$str = "Hello"; echo md5($str);

?>

Nəticə: 8b1a9953c4611296a827abf8c47804d7

1. nl2br() funksiyası-\n ifadəsi olan hissədə növbəti sətirə keçir. Məsələn:

<?php

echo"One line.\nAnother line.";

?>

Nəticədə iki tərəf ayrı-ayrı sətirlərdə olur. Əsasən formlarda textarea daxilində istifadəçi yazı

yazdıqda ortada enter basaraq növbəti sətrə

keçsə

də həmin yazılar yan-yana olur. Bu zaman

yazıları yeni sətirdə etmək üçün nl2br() funksiyasından istifadə olunur.

1. sha1() funksiyası-Yazıları sha-1 alqoritmi ilə şifrələmək üçün istifadə olunur. Məsələn:

<?php

$str = "Hello"; echo sha1($str);

?>

Nəticə: f7ff9e8b7bb2e09b70935a5d785e0cc5d9d0abf0

1. similar\_text() funksiyası-İki mətndə Məsələn:

eyni olan simvol sayını tapmaq üçün istifadə

olunur.

<?php

echo similar\_text("Hello World","Hello Peter");

?>

Nəticə: 7

Eyni zamanda nəticəni faizlə istifadə etmək lazımdır:

də hesablamaq olar. Bunun üçün funksiyanı aşağıdakı şəkildə

<?php

similar\_text("Hello World","Hello Peter",$percent); echo $percent;

?>

Nəticə: 63.636363636364

1. str\_pad() funksiyası-Yazını qeyd olunmuş simvol sayına qeyd olunmuş xarakterlə tamamlayır. Məsələn:

<?php

$str = "Hello World"; echo str\_pad($str,20,".");

?>

Nəticə:Hello World.........

Burada 20 simvol tamam olana qədər mətnin sonuna nöqtə yazır.

Eyni zamanda yazını tamamlarkən simvolları mətnin sol tərəfinə də yazmaq olar. Məsələn:

<?php

$str = "Hello World";

echo str\_pad($str,20,".",STR\_PAD\_LEFT);

?>

Nəticə: .........Hello World

Eyni zamanda simvolları yazının hər iki tərəfinə də yerləşdirmək olar. Məsələn:

<?php

$str = "Hello World";

echo str\_pad($str,20,".:",STR\_PAD\_BOTH);

?>

Nəticə: .:.:Hello World.:.:.

1. str\_repeat() funksiyası-Müəyyən bir yazını bir neçə dəfə təkrar yazmaq üçün istifadə olunur.

Məsələn:

<?php

echo str\_repeat(".",13);

?>

Nəticə: .............

1. str\_replace() funksiyası-Bir yazıdakı müəyyən hissəni başqa bir yazı ilə əvəz etmək üçün istifadə olunur. Məsələn:

<?php

echo str\_replace("world","Peter","Hello world!");

?>

Nəticə: Hello Peter!

Nəticə «Hello world» ifadəsində «world» sözü «Peter» sözü ilə əvəz olunur.

1. str\_shuffle() funksiyası()-Yazıdakı simvolları təsadüfi sıralamaq üçün istifadə olunur. Məsələn:

<?php

echo str\_shuffle("Hello World");

?>

Burada hər dəfə müxtəlif nəticələr çıxır. Nəticə 1: HoreWl olld Nəticə 2: d erloHWoll Nəticə 3: eldo HWroll

1. str\_split() funksiyası-Bu funksiya yazının simvollarında ibarət olan massiv düzəldir. Məsələn:

<?php print\_r(str\_split("Hello"));

?>

Nəticə: Array ( [0] => H [1] => e [2] => l [3] => l [4] => o )

Funksiyanın başqa cür istifadəsi də mövcuddur:

<?php print\_r(str\_split("Hello",3));

?>

Burada massiv elementlərində 3 simvol olur.

1. str\_word\_count() funksiyası-Cümlədəki sözlərin sayını tapmaq üçün bu funksiyadan istifadə olunur. Məsələn:

<?php

echo str\_word\_count("Hello world!");

?>

Nəticə: 2

1. strip\_tags() funksiyası-Mətn daxilindəki html teqlərini silmək üçün istifadə olunur. Məsələn:

<?php

echo strip\_tags("Hello <b>world!</b>");

?>

Nəticə: Hello world!

Bu funksiyadan istifadə edərkən bəzi teqlərin işlənməsinə icazə vermək olar. Məsələn:

<?php

echo strip\_tags("Hello <b><i>world!</i></b>","<b>");

?>

Nəticə: Hello **world!**

1. stripslashes() funksiyası-Mətn daxilindən \ işarəsini silmək üçün istifadə olunur. Məsələn:

<?php

echo stripslashes("Who\'s Peter Griffin?");

?>

Nəticə: Who's Peter Griffin?

1. stripos() funksiyası-Mətn daxilində işlənmiş bir sözün ilk yerini tapmaq üçün istifadə olunur. Məsələn:

<?php

echo stripos("I love php, I love php too!","PHP");

?>

Nəticə: 7

1. strlen() funksiyası-Mətn daxilindəki simvol sayını tapmaq üçün istifadə olunur. Məsələn:

<?php

echo strlen("Hello");

?>

Nəticə: 5

1. strstr() funksiyası()-Mətn daxilində müəyyən sözün olub-olmadığını yoxlamaq üçün istifadə olunur. Məsələn:

<?php

if(strstr("String functions","functions")){ echo "Var";

}

?>

Nəticədə şərt doğru olduğuna görə Var sözü çap olunur. Bir şeyi də nəzərə almaq lazımdır ki bu

funksiyada böyük və

kiçik hərflər fərqlənir. Yəni yuxarıda «functions» yerinə

«Functions»

yazsaydıq şərt doğru olmayacaqdı. Böyük və stristr() funksiyasından istifadə edə bilərik.

kiçik hərflərin nəzərə

alınmasını istəmiriksə

1. stristr() funksiyası-Bu funksiya da mətn də müəyyən sözün olub-olmadığını yoxlamaq üçün istifadə olunur. Strstr() funksiyasından fərqli olaraq bu funksiyada böyük və kiçik hərflərin fərqi yoxdur. Məsələn:

<?php

if(stristr("String functions","Functions")){ echo "Var";

}

?>

Burada şərt doğru olduğu üçün «Var» sözü çap olunacaq.

1. ucfirst() funksiyası-yazının ilk xarakterini böyültmək üçün istifadə olunur. Məsələn:

<?php

echo ucfirst("hello world!");

?>

Nəticə: Hello world!

1. ucwords() funksiyası-Yazıdakı hər bir sözün ilk hərfini böyültmək üçün istifadə Məsələn:

olunur.

<?php

echo ucwords("hello world");

?>

Nəticə: Hello World

1. lcfirst() funksiyası-Yazının ilk xarakterini kiçiltmək üçün istifadə olunur. Məsələn:

<?php

echo lcfirst("Hello world!");

?>

Nəticə: hello world!

1. strtolower() funksiyası-Yazının bütün hərflərini kiçiltmək üçün istifadə olunur. Məsələn:

<?php

echo strtolower("Hello WORLD.");

?>

Nəticə: hello world.

**Məlumatların şifrələnməsi**

Təhlükəsizlik baxımından bazaya göndərilən məlumatların şifrələnməsi vacibdir. Php dilində şifrələmə üçün bir neçə funksiya mövcuddur.

1. md5() funksiyası-Bu şifrələmə funksiyasını istifadə etdikdə 32 simvolluq bir şifrə əldə olunur. Bu şifrəni yenidən açmaq üçün bir funksiya mövcud deyil. Nümunə:

<?php

$x = 123456;

echo md5($x);

?>

Nəticə: e10adc3949ba59abbe56e057f20f883e

1. sha1() funksiyası-Bu funksiya ilə şifrələmə nəticəsində 40 simvolluq şifrə əldə olunur. Bu Bu şifrəni yenidən açmaq üçün bir funksiya mövcud deyil. Nümunə:

<?php

$x = 123456;

echo sha1($x);

?>

Nəticə: 7c4a8d09ca3762af61e59520943dc26494f8941b

1. crc32() funksiyası-Bu şifrələmə metodu ilə məlumat şifrələnərək bir tam ədəd əldə olunur. Bu şifrəni yenidən açmaq üçün bir funksiya mövcud deyil. Nümunə:

<?php

$x = 123456;

echo crc32($x);

?>

Nəticə: 158520161 4)Base64 şifrələmə

alqoritmində

digərlərindən fərqli olaraq şifrələnmiş məlumatı yenidən

deşifrə etmək üçün funksiya mövcuddur. Şifrələmək üçün base64\_encode() funksiyasından

istifadə

olunur. Şifrələnmiş məlumat deşifrə

etmək üçün isə

base64\_decode() funksiyasından

istifadə olunur. Nümunə:

<?php

$x = 123456;

$a = base64\_encode($x);

$b= base64\_decode($a);

echo 'Şifrələnmiş məlumat:'.$a.'<br />'; echo 'Deşifrə olunmuş məlumat:'.$b;

?>

Burada $a dəyişəni şifrələnmiş qiymətə

bərabər olur, $b dəyişənində

isə

məlumat yenidən

deşifrə

olunur. Tam təhlükəsizlik üçün yuxarıdakı funksiyalardan bir neçə

dəfə istifadə oluna

bilər. Məsələn bir qiyməti 3 dəfə şifrələmək olar. Məsələn:

md5() funksiyası ilə, iki dəfə

isə

sha1() funksiyası ilə

<?php

$a="Php";

$b=sha1(sha1(md5(md5(md5($a))))); echo $b;

?>

Artıq burada mıəlumat mürəkkəb şəkildə şifrələnir.

# Php dilində $\_SERVER massivi

$\_SERVER massivi serverə

dait məlumatları özündə

saxlayan bir massivdir. Bu massivdəki

məlumatlar server tərəfindən emal olunur. Massivdəki məlumatlar aşağıdakılardır:

1)$\_SERVER['PHP\_SELF'] — Olduğumuz səhifənin yolunu göstəriri. Məsələn:

<?php

echo $\_SERVER[‘PHP\_SELF’];

?>

2)$\_SERVER['GATEWAY\_INTERFACE'] - Serverin istifadə etdiyi CGİ versiyasını göstərir. 3)$\_SERVER['SERVER\_ADDR'] - Sayta aid İP adresini göstərir.

4)$\_SERVER['SERVER\_NAME'] - Saytın domenini göstərir. 5)$\_SERVER['SERVER\_PROTOCOL'] — Serverin http protokolunu yoxlayır. 6)$\_SERVER['HTTP\_ACCEPT\_CHARSET'] — Saytın xarakter dilini göstərir. 7)$\_SERVER['HTTP\_ACCEPT\_LANGUAGE'] — istifadəçinin brauzerinin dilini yoxlayır. 8)$\_SERVER['REMOTE\_ADDR'] — İstifadəçinin İP adresini yoxlayır.

9)$\_SERVER[‘QUERY\_STRING’] - Səhifə adından sonra gələn məlumatları göstərir.

Php dilində

fayllar yaratmaq və

# Fayl funksiyaları

onlar üzərində əməliyyatlar aparmaq mümkündür.

Əvvəlcə

faylların yaradılmasına baxaq. Php dilində olunur. Aşağıda faylın yaradılması göstərilir:

fayl yaratmaq üçün touch() funksiyasından istifadə

<?php touch("fayl.txt");

?>

Bu kod vasitəsilə fayl adlı mətn faylı yaradılır. Yaradılmış bu faylı unlink() funksiyası ilə silə bilərik:

<?php unlink("fayl.txt");

?>

Nəticədə fayl silinir. Yaradılmış faylı açaraq həmin fayla məlumat yaza bilərik. Bu zaman fopen() funksiyasından istifadə olunur.

<?php

$fp=fopen("1.txt","w"); fwrite($fp,"Php"); fclose($fp);

?>

Fopen() funksiyası ilə faylı açırıq. Fwrite() funksiyası ilə fayla yazı yazdıqdan sonra isə fclose() funksiyası ilə fayl əməliyyatını bağlayırıq. Fayla yazı yazarkən yazını növbəti sətirə keçirmək

üçün \n ifadəsini yaza bilərik. \t ifadəsini yazaraq yazılar arasında 8 xarakter məsafə qoya

bilərik.Fopen() funksiyasında ikinci arqument olaraq w yazdıq. w Faylı yazmaq üçün açır. Faylı digər əməliyyatlar üçün də aça bilərik. Bunun üçün w əvəzinə digər müvafiq simvolları yazmaq lazımdır.

|  |  |
| --- | --- |
| Simvol | Yerinə yetirdiyi funksiya |
| r | Faylı oxumaq üçün istifadə olunur. |
| r+ | Faylı həm açmaq həm də fayla məlumat yazmaq üçün istifadə olunur. |
| w | Fayla məlumat yazmaq üçün istifadə olunur. Yeni məlumat yazılarkən fayldakı köhnə məlumatlar silinir. |
| w+ | Faylı həm oxumaq həm də yazmaq üçün açır. Fayldakı köhnə məlumatlar silinmir. |
| a | Fayla məlumat yazmaq üçün istifadə olunur. Bu zaman fayldakı köhnə məlumatlar silinir. |
| a+ | Fayla həm məlumat yazmaq həm də oxumaq üçün istifadə olunur. Məlumat yazılarkən fayldakı köhnə məlumatlar silinmir |
| x | Faylı yaradır və oxumaq üçün açır. Əgər fayl mövcuddursa false dəyərini qaytarır. |
| x+ | Faylı yaradır, oxumaq və yazmaq üçün istifadə olunur. Əgər fayl mövcuddursa false dəyərini qaytarır. |

Bir çox fayl funksiyaları mövcuddur. Bunlar aşağıdakılardır:

1. copy() funksiyası-Bir faylın tərkibini başqa fayla kompyalamaq üçün istifadə olunur. Məsələn:

<?php copy("1.html","1.php");

?>

Burada 1.html faylında olan məlumat 1.php faylına kopyalanır.

1. file() funksiyası-Bu funksiya faylı sətir-sətir massivə yazır. Məsələn:

<?php

$massiv=array();

$massiv=file("1.txt"); print\_r($massiv);

?>

1. file\_exists() funksiyası-Faylın mövcudluğunu yoxlamaq üçün istifadə olunur. Məsələn:

<?php if(file\_exists("1.php")){ echo "Fayl mövcuddur!";

}

?>

1. file\_get\_contents() funksiyas-Fayldakı məlumatı əldə etmək üçün istifadə olunur. Məsələn:

<?php

echo file\_get\_contents("1.txt");

?>

1. file\_put\_contents() funksiyası-Fayla məlumat yazmaq üçün istifadə olunur. Bu funksiya ilə fayla məlumat yazılarkən fayldakı köhnə məlumatlar silinir. Məsələn:

<?php file\_put\_contents("1.html","Information");

?>

1. filesize() funksiyası-Faylın ölçüsünü göstərir. Məsələn:

<?php

echo filesize("test.txt");

?>

Nəticəni byte olaraq göstərir.

1. is\_dir() funksiyası-Bir qovluğun mövcud olub-olmadığını yoxlamaq üçün istifadə Məsələn:

olunur.

<?php if(is\_dir("build")){ echo "Qovluq var!";

}

else{

echo "Movcud deyil!";

}

?>

1. is\_readable() funksiyası-Faylı oxumağın mümkünlüyünü yoxlamaq üçün istifadə Məsələn:

olunur.

<?php if(is\_readable("1.html")){ echo "Oxuna bilər!";

}

else{

echo "Oxunması mümkün deyil!";

}

?>

1. is\_writeable()funksiyası-Faylın yazıla biləcəyini yoxlamaq üçün istifadə olunur. Məsələn:

<?php if(is\_writeable("test.txt")){ echo "Yazıla bilər!";

}

else{

echo "Yazılması mümkün deyil!";

}

?>

1. is\_executable() funksiyası-Faylın işləyə biləcəyin yoxlamaq üçün istifadə olunur. Məsələn:

<?php if(is\_executable("end.exe")){ echo "İşləyə bilər!";

}

else{

echo "İşləyə bilməz!";

}

?>

1. mkdir() funksiyası-Yeni qovluq yaratmaq üçün istifadə olunur. Məsələn:

<?php mkdir("Qovluq");

?>

Nəticədə yeni qovluq yaranır.

1. rename() funksiyası-Bu funkksiya faylın adını dəyişmək üçün istifadə olunur. Məsələn:

<?php rename("1.txt","yeni.txt");

?>

Nəticədə 1.txt adlı faylın adı dəyişdirilərək yeni.txt olur.

1. stat() funksiyası-Bu funksiya ilə fayla aid məlumatlardan ibarət massiv yaradılır. Məsələn:

<?php

$stat=stat("1.php"); echo $stat['size'];

?>

Burada faylın ölçüsü göstərilir. Bu funksiya vasitəsilə

ölçüdən

əlavə

fayla aid digər

xüsusiyyətlərə

də baxmaq olur. Burada size sözünün

əvəzinə

uyğun sözü yazaraq digər bir

xüsusiyyətə baxmaq olar. Bu sözlər aşağıdakılardır:

* + dev
  + ino
  + mode
  + nlink
  + uid
  + gid
  + rdev
  + size
  + atime
  + mtime
  + ctime
  + blksize
  + blocks

# Php fayl upload əməliyyatı

Php vasitəsilə faylı serverə yükləmək mümkündür. Bu əməliyyat üçün form aşağıdakı şəkildə yaradılmalıdır:

<form method="POST" enctype="multipart/form-data">

<input type="file" name="file" />

<button type="submit" name="submit">Yüklə</button>

</form>

Bu formda enctype="multipart/form-data" yazılmağı vacibdir. Fayl yükləmə əməliyyatı üçün bir çox hazır funksiyalar var. Bunlar aşağıdakılardır:

1)$\_FILES['fayl']['name']-Faylın adını göstərir. 2)$\_FILES['fayl']['type']-Faylın MIME tipini göstərir. 3)$\_FILES['fayl']['size']-Faylın baytlarla ölçüsünü göstərir.

4)$\_FILES['fayl']['tmp\_name']-Serverdə saxlanacaq faylın müvəqqəti adını göstərir. 5)$\_FILES['fayl']['error']-Fayl yükləmə zamanı alınan xəta kodudur.

move\_uploaded\_file() funksiyası faylı serverə daşıyır. İndi isə bu funksiyalardan istifadə etməklə yükləmə əməliyyatına baxaq:

<form method="POST" enctype="multipart/form-data">

<input type="file" name="file" />

<button type="submit" name="submit">Yüklə</button>

</form>

<?php if(isset($\_POST['submit'])){

$file = $\_FILES['file'];

$qovluq = "images/";

$upload\_file = $qovluq.$file['name']; if(move\_uploaded\_file($file['tmp\_name'], $upload\_file)){ echo "<h3>Şəkil əlavə olundu!</h3>";

}

else{

echo "<h3>Şəkil əlavə olunmadı!</h3>";

}

}

?>

Burada şəkil images qovluğuna yüklənir. Ancaq burada heç bir yoxlama aparılmır. Biz şəklin ölçüsünü, şəklin tipini yoxlayaraq şəkli serverə əlavə edə bilərik:

<form method="POST" enctype="multipart/form-data">

<input type="file" name="file" />

<button type="submit" name="submit">Yüklə</button>

</form>

<?php if(isset($\_POST['submit'])){

$file = $\_FILES['file'];

$uzantilar = array("jpg", "png", "images/jpeg", "images/png");

$dizin="images/";

$upload\_file = $dizin.basename($file['name']);

$size = $file['size'];

$uzanti = explode(".", $file['name']);

$uzanti = $uzanti[count($uzanti)-1];

$tip = $file['type']; if($file['name'] != ""){

if(in\_array($tip, $uzantilar) || in\_array($uzanti, $uzantilar)) if($size < (1024\*1024\*3)){ if(move\_uploaded\_file($file['tmp\_name'], $upload\_file)){ echo "<h3>Şəkil əlavə olundu!</h3>"; // olumlu

}

else{

echo "<h3>Şəkil əlavə olunmadı!</h3>"; // hata

}

}else{

echo "<h3>Şəklin ölçüsü 3MB-dan çox olmamalıdır.</h3>"; // hata

}

}else{

echo "<h3>Yalnız Jpg və Png formatlar qəbul edilir.</h3>"; // hata

}

}

?>

Burada yalnız png və

jpg formatında olan şəkillər

əlavə

oluna bilər və eyni zamanda şəklin

ölçüsü maksimum 3MB ola bilər.

# Php ilə şəkil hazırlamaq

Php vasitəsilə

şəkillər hazırlamaq mümkündür. Php ilə

şəkil yaratmaq

əsasən saytlarda olan

təsdiləmə kodunu hazırlayarkən və başqa yerlərdə istifadə olunur. Şəkil hazırlamaq üçün hazır funksiyalar yaradılmışdır. Bu funksiyalarla tanış olaq:

ImageCreate() funskiyası-Bu funksiya şəkli yaratmaq üçündür. İki parametr daxil edilməlidir. Birinci parametr olaraq şəklin eni ikinci parametr olaraq isə şəklin uzunluğu daxil edilir.

İmageColorAllocate() funksiyası-Bu funksiya rəngi seçmək üçündür. Dörd parametr qeyd olunmalıdır. Birinci parametr olaraq ImageCreate() funksiyası ilə yaradılan şəklin dəyişəni ;İkinci parametr olaraq qırmızı rəngin dəyəri; Üçüncü parametr olaraq yaşıl rəngin dəyəri; dördüncü parametr olaraq isə göy rəng dəyəri daxil olunmalıdır. Bu üç rəng vasitəsilə bütün rənglər əldə oluna bilər. Dəyərlər 0-255 arası olur.

ImageString() funksiyası-Bu funksiya şəklin üzərinə yazı yazmaq üçündür. Altı parametr daxil

edilməlidir. Birinci parametr İmageCreate() ilə yaradılan şəklin dəyişəni;İkinci parametr olaraq yazının böyüklüyü;üçüncü parametr olaraq yazının x koordinatı;dördüncü parametr olaraq yazının y koordinatı;beşinci parametr olaraq yazılacaq yazı;altıncı parametr olaraq isə İmageColorAllocate() ilə yaradılan rəngin dəyəri daxil edilməlidir.

ImageJpeg() funksiyası-Hazırlanan şəklin Jpeg formada olması üçündür. Birinci parametr olaraq

İmageCreate() ilə

yaradılan şəklin dəyişəni; İkinci parametr olaraq

əgər şəkil yaddaşda

saxlanacaqsa saxlanılacaq şəklin adı; üçüncü parametr olaraq isə edilməlidir.

şəklin keyfiyyəti daxil

ImageGif() funksiyası-Haz;rlanan şəklin gif olması üçündür. İlk parametr olaraq İmageCreate() ilə yaradılan şəklin dəyişəni; ikinci parametr olaraq isə əgər şəkil yadda saxlanacaqsa saxlanacaq şəklin adı daxil edilməlidir.

ImagePng() funksiyası-Hazırlanan şəklin png formatda olması üçündür. Dörd parametr daxil oluna bilər. Birinci parametr olaraq İmageCreate() ilə yaradılan şəklin dəyişəni İkinci parametr olaraq əgər şəkil yadda saxlanılacaqsa saxlanılacaq şəklin adı üçüncü parametr olaraq şəklin keyfiyyəti dördüncü parametr olaraq isə istifadə olunan filtr. Bunlardan birincisi mütləq daxil edilməlidir.

İmageDestroy() funksiyası-Yaddaşı təmizləyir. Faylın sonuna əlavə edilməlidir. İndi isə bu funksiyaları birləşdirərək şəklin hazırlanmasına baxaq:

<?php

header ("Content-type: image/jpeg");

$image= ImageCreate (350,350);

$arxaplan=ImageColorAllocate($image , 30 , 144 , 255);

$white= ImageColorAllocate ($image , 255 , 255 , 255);

ImageString($image, 30 , 40 , 160 , "Image", $white); ImageJpeg($image,NULL,100); ImageDestroy($image);

?>

Yuxarıda göstərilən funksiyalarla şəkli hazırladıq. Ilk sətirdə

header funksiyasını işlədərək

səhifənin şəkil olduğunu bildirdik. İndi isə bu kodları yazdığımız səhifəni image.php adı ilə

yadda saxlayaq və başqa bir səhifədə bu şəkli işlədək:

<img src="image.php">

Bu şəkildə hazırladığımız şəkli səhifədə istifadə edə bilərik.

# Php və XML

Php və XML arasında əlaqə yaratmaq mümkündür. Bunun üçün hazır funksiyalar mövcuddur. İlk öncə xml faylını yaradaq:

<?xml version="1.0" encoding="utf-8"?>

<users>

<user>

<login>Login1</login>

<password>Password1</password>

</user>

<user>

<login>Login2</login>

<password>Password2</password>

</user>

<user>

<login>Login3</login>

<password>Password3</password>

</user>

</users>

Php ilə

XML faylını oxumaq üçün simplexml\_load\_file() funksiyasından istifadə

olunur. Bu

funksiya ilə xml faylının elementlərindən ibarət massiv yaradılır:

<?php

$xml=simplexml\_load\_file("users.xml"); print\_r($xml);

?>

Bu şəkildə xml faylını oxuya bilərik. Faylı oxuyarkən foreach operatorundan istifadə edə bilərik:

<?php

$xml=simplexml\_load\_file("users.xml"); foreach($xml->user as $user){

echo $user->login."<br>"; echo $user->password."<br>";

}

?>

Bu şəkildə xml faylını oxuya bilərik.

# Mail funksiyaları

Php dilində mail göndərmək üçün funksiyalar mövcuddur. Mail() funksiyası vasitəsilə hər hansı bir emailə məktub göndərə bilərik. İstifadə qaydası aşağıdakı şəkildədir:

<?php mail("Email","Subject","Message");

?>

Eyni zamanda funksiyanı aşağıdakı şəkildə də işlətmək olar:

<?php

$to = ["som](mailto:somebody@example.com)e[body@example.com](mailto:somebody@example.com)";

$subject = "My subject";

$txt = "Hello world!";

$headers = "From: [webmaster@example.com](mailto:webmaster@example.com)" . "\r\n" ."CC: [somebodyelse@example.com](mailto:somebodyelse@example.com)"; mail($to,$subject,$txt,$headers);

?>

Eyni zamanda məktub daxilinə html kodları da yaza bilərik. Məsələn:

<?php

$to = ["som](mailto:somebody@example.com)e[body@example.com,](mailto:somebody@example.com) [somebodyelse@example.com](mailto:somebodyelse@example.com)";

$subject = "HTML email";

$message = "

<html>

<head>

<title>HTML email</title>

</head>

<body>

<p>This email contains HTML Tags!</p>

<table>

<tr>

<th>Firstname</th>

<th>Lastname</th>

</tr>

<tr>

<td>John</td>

<td>Doe</td>

</tr>

</table>

</body>

</html> ";

$headers = "MIME-Version: 1.0" . "\r\n";

$headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

$headers .= 'From: [<webmaster@example.com](mailto:webmaster@example.com)>' . "\r\n";

$headers .= 'Cc: myboss@example.com' . "\r\n"; mail($to,$subject,$message,$headers);

?>

HTTP funksiyaları aşağıdakılardır:

# HTTP funksiyaları

1. header() funksiyası- Səhifəni başqa bir səhifəyə yönləndirmək və digər əməliyyatlar üçün bu funksiyadan istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

<?php

header("Location: 1.php");

?>

Bu zaman səhifə

avtomatik olaraq 1.php səhifəsinə

yönləndirilir. Eyni zamanda səhifəni

müəyyən bir vaxtdan sonra da yönləndirə bilərik. Məsələn:

<?php

echo "Php"; header("refresh:2");

?>

Burada hər iki saniyədən bir səhifə yenilənəcək. Eyni zamanda məyyən bir vaxtdan sonra da səhifəni başqa bir səhifəyə yönləndirə bilərik. Məsələn:

<?php

header("Refresh:2; url=page2.php");

?>

Burada 2 saniyədən sonra səhifə

göstərilən səhifəyə

yönləndirilir. Eyni zamanda header()

funksiyası ilə səhifə tərkibini də müəyyən etmək olar. Məsələn:

<?php

header("Content-type: text/plain; charset=utf-8");

?>

Funksiya yuxarıdakı formada istifadə oluna bilər.

1. headers\_list() funksiyası-Səhifədə istifadə olunan başlıqları massiv şəklində göstərir. Məsələn:

<?php

header("Content-type: text/plain; charset=utf-8"); print\_r(headers\_list());

?>

Nəticə:

Array

(

[0] => X-Powered-By: PHP/5.2.17

[1] => Content-type: text/plain; charset=utf-8

)

1. headers\_sent() funksiyası()-

# Include və require funksiyaları

Bir səhifəyə başqa səhifəni çağırmaq üçün include və require funksiyalarından istifadə olunur. Bu

iki funksiya eyni işi görsə də

müəyyən fərqləri var. İlk öncə

include funksiyasına baxaq.

Aşağıdakı səhifə 1.php səhifəsidir.

<?php

$a=5;

echo "1.php";

?>

1.php səhifəsini 2.php səhifəsinə çağıraq:

<?php include("1.php"); echo $a;

?>

Göründüyü kimi 1.php səhifəsi 2.php səhifəsinə çağrılır. İnclude funksiyasında çağırılan faylın

olub-olmaması vacib deyil. davam edəcək.

Əgər həmin fayl olmasa belə

kodlar növbəti sətridən işləməyə

Require funksiyasının da istifadə include funksiyası ilə eynidir:

<?php require("1.php"); echo $a;

?>

Require funksiyasını işlədərkən əgər çağırılan fayl mövcud deyilsə bu funksiya digər kodların da

işləməyinə

icazə

verməyəcək. Bu funksiyalara aid daha bir nümunə

göstərək. Məsələn

functions.php səhifəsində funksiyalarımız var:

<?php

function toplama($a,$b){ return $a+$b;

}

?>

İndi isə 1.php səhifəsinə functions.php səhifəsini çağıraq və toplama funksiyasını istifadə edək:

<?php include("functions.php"); echo toplama(4,5);

?>

functions.php səhifəsini 1.php səhifəsinə çağırdıq və oradakı funksiyanı istifadə etdik.

# Php və MySql arasında əlaqə

Saytda istifadəçi məlumatlarını və digər məlumatları saxlamaq üçün verilənlər bazasına ehtiyac duyulur. Bu mövzuda PHP və MySql verilənlər bazası ilə əlaqə göstərilir. Mövzuya başlamazdan öncə Mysql verilənlər bazası, Phpmyadminlə tanış olmağınız tövsiyyə olunur.

MySql verilənlər bazası ilə əlaqə yaratmaq üçün mysqli\_connect() funksiyasından istifadə olunur. İstifadə qaydası aşağıdakı şəkildədir:

mysqli\_connect(host,username,pass,database);

Verilənləri yazdıqda baza ilə əlaqə yaradılmış olur. Real baza ilə əlaqə yaradaq:

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","parol","u372209141\_ye ni");

?>

Bu şəkildə

bazayla

əlaqə

yaradılır.

Burada oxucuya tam aydın olması üçün baza məlumatları

olduğu şəkildə yazılıb. Siz burada öz məlumatlarınızı yazmalısınız. Əlaqədə hər-hansı bir səhvin olub-olmamasını yoxlamaq üçün mysqli\_connect\_errno() funksiyasından aşağıdakı şəkildə istifadə edə bilərik:

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","parol","u372209141\_ye ni");

if (mysqli\_connect\_errno())

{

echo "Error!: " . mysqli\_connect\_error();

}

?>

Səhv olduqda şərt ödənir və səhv olduğu haqda məlumat verilir.

Baza ilə əlaqəni bağlamaq üçün mysqli\_close() funksiyasından istifadə olunur.

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","parol","u372209141\_ye ni");

if (mysqli\_connect\_errno())

{

echo "Error!: " . mysqli\_connect\_error();

}

mysqli\_close($connect);

?>

Burada baza ilə əlaqə yaradılır və sonda əlaqə bağlanır.

Bazada müəyyən

əməliyyatı aparmaq üçün mysqli\_query() funksiyasından istifadə

olunur.

Məsələn bazaya aşağıdakı şəkildə məlumat əlavə edə bilərik:

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","parol","u372209141\_ye ni");

mysqli\_query($connect,"insert into news(id,title) values ('5','Php')");

mysqli\_close($connect);

?>

Burada news cədvəlinə yeni sətir əlavə etdik və id və title sütunlarına məlumat yazdıq. Eynilə aşağıdakı şəkildə cədvəldəki məlumatları silə bilərik:

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","parol","u372209141\_ye ni");

mysqli\_query($connect,"delete from news where id=5"); mysqli\_close($connect);

?>

Burada id dəyəri 5 olan sətirlər silinir.

Bazadan məlumatı oxumaq əlavə etmə və silmə əməliyyatlarına nisbətən mürəkkəb olur. Çünki

sətirlərin sayı çox olduğuna görə

bir dövr yaratmalı və

bütün sətirləri oxumaq lazımdır.

Məlumatları sətir-sətir oxumaq üçün mysqli\_fetch\_array() funksiyasını da işlədəcəyik.

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","parol","u372209141\_ye ni");

$sql=mysqli\_query($connect,"select \* from news"); while($read=mysqli\_fetch\_array($sql)){

echo $read['id'];

echo "-";

echo $read['title']; echo "<br>";

mysqli\_close($connect);

?>

Bu şəkildə bazadakı məlumatları oxuya bilərik.

Əsas əməliyyatlar yuxarıdakı şəkildədir. Bunlardan əlavə Php dilində bir çox mysqli funksiyaları mövcuddur. Bunlar aşağıdakılardır:

1. mysqli\_change\_user() funksiyası-MySql istifadəçisini dəyişmək üçün istifadə olunur. Məsələn:

<?php

$con=mysqli\_connect("localhost","my\_user","my\_password","my\_db"); mysqli\_change\_user($con, "my\_user", "my\_password", "my\_test"); mysqli\_close($con);

?>

Bu funksiya verilənlər bazası əlaqəsinin istifadəçisini dəyişir.

1. mysqli\_character\_set\_name() funksiyası-Verilənlər bazasının xarakter setini öyrənmək üçün istifadə olunur. Məsələn:

<?php

$con=mysqli\_connect("localhost","my\_user","my\_password","my\_db");

$charset=mysqli\_character\_set\_name($con); echo "Default character set is: " . $charset; mysqli\_close($con);

?>

1. mysqli\_get\_host\_info() funksiyası-Serverin host adı qoşulma növü haqqında məlumat verir. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","12345","u372209141\_y eni");

echo mysqli\_get\_host\_info($connect);

?>

1. mysqli\_get\_server\_info() funksiyası-MySql serverin versiyasını göstərir. İstifadə aşağıdakı şəkildədir:

qaydası

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","servant12","u37220914 1\_yeni");

echo mysqli\_get\_server\_info($connect);

?>

1. mysqli\_num\_fields funksiyası-Cədvəldəki sütun sayını göstərir. İstifadə şəkildədir:

qaydası aşağıdakı

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","12345","u372209141\_y eni");

$sql=mysqli\_query($connect,"Select \* from news");

$fields=mysqli\_num\_fields($sql); print\_r($fields);

?>

1. mysqli\_num\_rows() funksiyası-Cədvəldəki sətirlərin sayını göstərir. İstifadə qaydası aşağıdakı şəkildədir:

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","12345","u372209141\_y eni");

$sql=mysqli\_query($connect,"Select \* from news");

$fields=mysqli\_num\_rows($sql); print\_r($fields);

?>

1. mysqli\_stat() Funksiyası-Bazanın statusunu göstərir. Istifadə qaydası aşağıdakı şəkildədir:

<?php

$connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","12345","u372209141\_y eni");

echo mysqli\_stat($connect);

?>

# Obyektyönümlü proqramlaşdırma

Əsasən böyük layihələrdə obyektyönümlü proqramlaşdırmadan istifadə olunur. Obyektyönümlü

proqramlaşdırmada dəyişənlər və aşağıdakı şəkildə elan olunur:

funksiyalar sinif daxilində

yazılır və

istifadə

olunur. Sinif

<?php class sinif{

//Operations...

}

?>

Sinif bu şəkildə yaradılır. Sinif daxilində dəyişənlər var açar sözü ilə elan olunur.

<?php class sinif{ var $a;

var $b;

}

?>

Burada sinif daxilində iki dəyişən elan olundu. İndi isə bu sinifdə olan dəyişənlərdən istifadə edək. Bunun üçün obyekt yaradırıq:

<?php class sinif{ var $a;

}

$obyekt=new sinif();

$obyekt->a=5; echo $obyekt->a;

?>

«new» açar sözü ilə obyekti yaradırıq. Dəyişənə müraciət üçün $obyekt->a yazırıq. Bu dəyişənə müraciət zamanı $ işarəsini yazmırıq. Başqa bir nümunə də göstərək:

<?php class sinif{ var $a;

var $b;

}

$obyekt=new sinif();

$obyekt->a=5;

$obyekt->b=10;

echo $obyekt->a+$obyekt->b;

?>

Burada sinif dəyişənlərinə qiymət verib onların cəmini tapırıq. Sinif daxilində sabitlərdən də istifadə edə bilərik. Məsələn:

<?php class sinif{

const sabit="Php";

}

echo sinif::sabit;

?>

Const açar sözü ilə sabiti elan edirik sonra isə sinif::sabit ifadəsi ilə sabitə müraciət edirik. İndi isə sinif daxilində funksiyaların elan olunmasına baxaq:

<?php class sinif{

function funksiya(){ echo "Class";

}

}

$obyekt=new sinif();

$obyekt->funksiya();

?>

Burada isə sinif daxilində funksiyanı yaratdıq və sonra istifadə etdik. İndi isə eyni sinif daxilində həm funksiyadan həm də dəyişənlərdən istifadə edək:

<?php class sinif{ var $ad;

function funksiya($name){

$this->ad=$name;

}

}

$obyekt=new sinif();

$obyekt->funksiya("Name"); echo $obyekt->ad;

?>

Burada ilk öncə dəyişəni elan edirik. Dəyişənri elan etdikdən sonra funksiyanı elan edirik.

Funksiyada 1 parametr var. Sinif daxilində

dəyişənlərə

müraciət etmək üçün $this->ad kimi

yazırıq. Funksiya daxilində isə sinif dəyişənini funksiyanın parametrinə bərabər edirik. Sonda isə funksiyanı istifadə edirik və dəyişənə qiymət vermiş oluruq. Daha bir nümunəyə baxaq:

<?php class sinif{ var $a;

var $b;

function toplama(){ echo $this->a+$this->b;

}

}

$obyekt=new sinif();

$obyekt->a=5;

$obyekt->b=10;

$obyekt->toplama();

?>

Burada da sinif vasitəsilə

toplama

əməliyyatı yerinə

yetirilir. Sinif daxilində

sabitə

aşağıdakı

şəkildə müraciət olunur:

<?php class sinif{

const sabit="Php"; function funksiya(){ echo self::sabit;

}

}

$obyekt=new sinif();

$obyekt->funksiya();

?>

Burada self açar sözü ilə sinif daxilində sabitə müraciət etdik.

Sinif daxilində construct adı ilə çağrılır. Məsələn:

yaradılmış funksiya obyekt yaradılanda avtomatik şəkildə

<?php class sinif{

function construct(){ echo "Function";

}

}

$obyekt=new sinif();

?>

Burada obyekt yaradılanda artıq funksiya avtomatik şəkildə çağrılır. Bu funksiyaya aid başqa bir nümunəyə də baxaq:

<?php class sinif{

function construct(){ echo "Function";

}

function funksiya(){ echo "Funksiya";

}

}

$obyekt=new sinif();

$obyekt->funksiya();

?>

Burada görə çağrıldı.

bilərik ki construct funksiyası biz çağırdığımız funksiyadan

əvvəl avtomatik

Sinif daxilində destruct adı ilə avtomatik çağrılır. Məsələn:

yaradılmış funksiya çağrılmış bütün funksiyalardan sonra

<?php class sinif{

function destruct(){ echo "Function";

}

function funksiya(){ echo "Funksiya";

}

}

$obyekt=new sinif();

$obyekt->funksiya();

?>

Burada isə çağırdığımız funksiyadan sonra destruct funksiyası avtomatik olaraq çağrıldı.

İndi isə construct və destruct funksiyalarından da istifadə yaradaq:

etmək MySql baza ilə əlaqə

<?php class baza{

var $connect;

function construct(){

$this>connect=mysqli\_connect("mysql.1freehosting.com","u372209141\_yeni","12345","u3722091 41\_yeni");

}

function query($query){ mysqli\_query($this->connect,$query);

}

function destruct(){ mysqli\_close($this->connect);

}

}

$obyekt=new baza();

$obyekt->query("Delete from news");

?>

Burada construct funksiyasında baza ilə əlaqə yaradırıq. Obyekt yaradılanda funksiya avtomatik

çağrılır və

baza ilə əlaqə

yaradılır. Növbəti olaraq sorğunu yazmaq üçün query funksiyasını

yaratdıq və

çağırdıq. Növbəti olaraq isə destruct funksiyasında

əlaqəni bağladıq. Sonda

destruct funksiyası avtomatik çağrılacağı üçün baza ilə əlaqə avtomatik olaraq bağlanacaq.

İndi isə

private və

public anlayışları ilə

tanış olaq. Public olaraq elan edilən dəyişən və

funksiyalar ümumi olur yəni bu formada elan olunan dəyişənlərə kənardan qiymət vermək

mümkün olur. Bizim yuxarda yazdığımız kodlarda bütün dəyişən və funksiyalar biz public

yazmağımızdan asılı olmayaraq public hesab olunurdu. Məsələn aşağıdakı kodlar eynidir:

<?php class sinif{ public $a;

}

?>

<?php class sinif{ var $a;

}

?>

Yəni burada public yazmasaq da bu dəyişən public sayılır. Yuxarıda qeyd etdiyimiz kimi public

dəyişənlərə

kənardan qiymət vermək mümkündür. Private olan dəyişənlərə

isə

yalnız sinif

daxilində qiymət verilə bilər. Aşağıdakı koda baxaq:

<?php class sinif{ public $a;

}

$obyekt=new sinif();

$obyekt->a=5; echo $obyekt->a;

?>

Burada dəyişəni elan edərkən public yerinə private yazsaq:

<?php class sinif{ private $a;

}

$obyekt=new sinif();

$obyekt->a=5; echo $obyekt->a;

?>

Artıq burada səhifəni işlədən zaman səhv çıxacaq. Bunun səbəbi yuxarıda da qeyd etdiymiz kimi private olan dəyişənlərə yalnız sinif daxilində qiymət verilə bilər. Aşağıdakı kod isə doğrudur:

<?php class sinif{ private $a;

public function funksiya($var){ echo $this->a=$var;

}

}

$obyekt=new sinif();

$obyekt->funksiya(5);

?>

Burada dəyişənə funksiyanın qiyməti verildi.

Sinifləri genişləndirmək üçün extends açar sözündən istifadə olunur. Nümunə:

<?php

class sinif1{ public $a;

}

class sinif2 extends sinif1{ function funksiya($var){

$this->a=$var; echo $this->a;

}

}

$obyekt=new sinif2();

$obyekt->funksiya("Hello World!");

?>

Burada sinifi genişləndirərək sinif2-də sinif1-in dəyişənindən istifadə etdik.

Mücərrəd siniflər anlayışı mövcuddur. Mücərrəd sinif daxilində dəyişənlər və funksiyalar elan olunur, əməliyyatlar isə genişləndirilən siniflərdə yazılır. Mücərrəd siniflər abstract açar sözü ilə yazılır. Nümunə:

<?php

abstract class sinif1{ public $a;

abstract public function funksiya($var);

}

class sinif2 extends sinif1{ public function funksiya($var){

$this->a=$var; echo $this->a;

}

}

$obyekt=new sinif2();

$obyekt->funksiya("Hello World!");

?>

Mücərrəd sinifdən bu şəkildə istifadə etdik.

Mücərrəd siniflərdən

əlavə

onlara çox oxşar olan interface siniflər də

mövcuddur. Bu tip

siniflərdə

funksiyalar elan olunur.

Əməliyyatlar isə

genişləndirilmiş sinifdə

yazılır.Bu tip

siniflərdə

dəyişənlər və

funksiyalar public olmalıdır.Bu tip siniflər interface açar sözü ilə

yaradılır və siniflər genişləndirildikdə implements açar sözündən istifadə olunur. Nümunə:

<?php

interface sinif1{

public function funksiya();

}

class sinif2 implements sinif1{ public function funksiya(){ echo "Hello World!";

}

}

$obyekt=new sinif2();

$obyekt->funksiya();

?>

Mücərrəd və interface sinifləri bir yerdə işlədə bilərik. Məsələn:

<?php

abstract class sinif1{ public $var;

}

interface sinif2{

public function funksiya($var);

}

class sinif3 extends sinif1 implements sinif2{ public function funksiya($var){

$this->var=$var; echo $this->var;

}

}

$obyekt=new sinif3();

$obyekt->funksiya("Hello World!");

?>

Burada mücərrəd və interface siniflərini bir yerdə istifadə etdik.

**Php Data Objects**

PDO(Php Data Objects) verilənlər bazası ilə əməliyyatlar aparmaq üçün yardılmış sinifdir. PDO bir çox verilənlər bazasını dəstəkləyir:

* + Cubrid
  + FreeTDS / Microsoft SQL Server / Sybase
  + Firebird/Interbase 6
  + IBM DB2
  + IBM Informix Dynamic Server
  + MySQL 3.x/4.x/5.x
  + Oracle Call Interface
  + ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
  + PostgreSQL
  + SQLite 3 and SQLite 2
  + Microsoft SQL Server / SQL Azure

Bu səbəbdən PDO ilə yaradırıq:

<?php

işləmək daha

əlverişlidir. MySql bazası ilə

aşağıdakı şəkildə əlaqə

$db = new PDO('mysql:host=localhost;dbname=test', $user, $pass);

?>

Birinci parametrdə host də bazanı yazırıq ikinci parametrdə istifadəçi adı üçüncü parametr olaraq isə parol yazırıq. Sorğu yazmaq üçün isə Pdo sinifinin daxilindəki exec() funksiyasından istifadə edirik:

<?php

$db = new PDO('mysql:host=mysql.1freehosting.com;dbname=u372209141\_yeni', "u372209141\_yeni", "servant12");

$db->exec("Delete from news");

?>

Bu şəkildə cədvəldən məlumatları silmiş oluruq. Adətən Insert, Delete və Update sorğuları üçün exec(), Select sorğusu üçün isə query() funksiyasından istifadə olunur.

Pdo sinifinin əsas xüsusiyyətlərindən biri olan binding üsulu ilə dəyişənləri sorğuda təhlükəsiz

və düzgün bir şəkildə

istifadə

edə

bilərik. Bunun üçün prepare metodu ilə

sorğu yazılır və

sorğuda dəyişənlərin yerinə

? Işarəsi qoyulur. Sonra isə

execute() metodu ilə

? işarəsi olan

yerlərə lazımi dəyişənləri yazırıq. Nümunə:

<?php

$db = new PDO('mysql:host=mysql.1freehosting.com;dbname=u372209141\_yeni',

"u372209141\_yeni", "servant12");

$query=$db->prepare("insert into news(Name,title) values(?,?)");

$query->execute(array("Ad",Başlıq));

?>

Nəticədə sorğu aşağıdakı şəkildə olacaq: insert into news(Name,title) values('Ad','Başlıq') Beləliklə dəyişənlərdən təhlükəsiz və düzgün şəkildə istifadə etmiş oluruq.

Pdo sinifi ilə cədvəldəki məlumatları oxuyarkən query() metodundan istifadə edəcəyik:

<?php

$db = new PDO('mysql:host=mysql.1freehosting.com;dbname=u372209141\_yeni', "u372209141\_yeni", "servant12");

foreach($db->query("Select \* from news") as $row){ echo $row['name']."-".$row['title']."<br>";

}

?>

Bu şəkildə cədvəldəki məlumatları oxumuş oluruq. Pdo əlaqəsini aşağıdakı şəkildə bağlamaq olar:

<?php

$db = new PDO('mysql:host=mysql.1freehosting.com;dbname=u372209141\_yeni', "u372209141\_yeni", "servant12");

$db=null;

?>

Bu şəkildə Pdo əlaqəsini bağlamış oluruq.

Sorğunu göndərən zaman hər-hansı bir xəta yarana bilər. Bu xətanı öyrənmək üçün errorInfo() metodundan istifadə edə bilərik:

<?php

$db = new PDO('mysql:host=mysql.1freehosting.com;dbname=u372209141\_yeni', "u372209141\_yeni", "servant12");

if($db->exec("Delete from news Where")){ echo "Cədvəldən məlumatlar silindi!";

}

else{

echo "Xəta yarandı! <br>"; echo $db->errorInfo();

}

?>

Yuxarıdakı kodda sorğunu səhv yazdığımıza görə(where operatorundan sonra heç bir şərt yazmamışıq) şərt ödənməyəcək və xəta mesajını alacağıq.

# Php və Ajax

Ajax Javascript vasitəsilə

səhifəni yeniləmədən

əməliyyatlar aparmaq üçün yaradılmış

texnologiyadır. Ajax ilə

səhifə

yenilənmədən

əməliyyatlar aparmaq mümkündür. Bu

texnologiyanı istifadə etmək üçün bir çox hazır kitabxanalar mövcuddur. Biz bu dərsdə Jquery

kitabxanasından istifadə istifadəsinə baxaq:

edəcəyik. Bunu üçün səhifəyə

Jquery faylını çağırmalıyıq. Ajax

<html>

<head>

<script src=["ht](http://code.jquery.com/jquery-2.2.0.js)t[p://code.jquery.com/jquery-2.2.0.js](http://code.jquery.com/jquery-2.2.0.js)"></script>

<script>

$(function(){

$.ajax({

type: "post",

url: "ajax.php",

data: {"name":"Ad"}, success: function(e){ alert(e);

}

})

})

</script>

</head>

<body>

</body>

</html>

ajax.php səhifəsi:

<?php

$ad=$\_POST['name']; echo $ad;

?>

İndi isə izahına keçək. İlk səhifədə jquery faylını çağırdıqdan sonra ajax metodundan istifadə

edirik. İlk parametr type parametridir. Php səhifəsində

post metodundan istifadə

edəcəyimizə

görə burada post yazırıq. İkinci parametr url parametridir. Burada məlumatı göndərəcəyimiz

səhifəni yazırıq. Üçüncü parametr data parametridir. Burada isə göndərəcəyimiz məlumatları

yazırıq. Dördüncü parametr isə success parametridir. Məlumat göndərildikdən sonra yerinə

yetiriləcək funksiyanı buraya yazırıq. Ajax.php səhifəsində

isə

post metodu ilə

məlumatı

götürürük və çap edirik. Nəticədə ilk səhifədə alert mesajı ilə ajax.php səhifəsində çap olunan

məlumat göstərilir. İndi isə

Formdan gələn məlumatları səhifə

yenilənmədən bazaya

əlavə

edək. Əvvəlcə ilk səhifəni hazırlayaq:

<html>

<head>

<script src=["ht](http://code.jquery.com/jquery-2.2.0.js)t[p://code.jquery.com/jquery-2.2.0.js](http://code.jquery.com/jquery-2.2.0.js)"></script>

<script>

$(function(){

$("#button").click(function(){ var ad=$("#text").val();

var soyad=$("#text2").val();

$.ajax({

type: "post",

url: "ajax.php",

data: {"name":ad,"surname":soyad}, success: function(e){

alert(e);

}

})

})

})

</script>

</head>

<body>

Ad: <input type="text" name="ad" id="text">

<br>

Soyad: <input type="text" name="soyad" id="text2">

<br>

<button id="button">Send</button>

</body>

</html>

ajax.php səhifəsi:

<?php

$ad=$\_POST['name'];

$soyad=$\_POST['surname']; if($ad!=NULL && $soyad!=NULL){

$db = new PDO('mysql:host=mysql.1freehosting.com;dbname=u372209141\_yeni', "u372209141\_yeni", "servant12");

$query=$db->prepare("insert into news(Name,title) values(?,?)");

$query->execute(array($ad,$soyad)); echo "Məlumat bazaya əlavə olundu!";

}

else{

echo "Xahiş olunur məlumatları tam doldurun!";

}

?>

Beləliklə məlumatları bazaya əlavə edirik.