

SANJARBEK SOBIRJONOV

PHP7 VA OBYEKTGA YO'NALTIRILGAN DASTURLASH



**MYSQL VA PHP7 BO'YICHA
AMALIYOTLARGA ASOSLANGAN VEB
-DASTURLASH BO'YICHA ELEKTRON
KITOB**

Mundarija

Kirish	15
PHP bilan nimalarni qilish mumkin?.....	15
PHP ning boshqa tillardan ustunligi	16
Boshladik!	17
Lokal Veb Serverni sozlab olamiz.....	17
Birinchi PHP skriptimizni yaratamiz.....	18
PHP sintaksis	19
Standard PHP sintaksis.....	20
HTML ichida PHP-ni biriktirish.....	20
PHP izohlar.....	21
PHP-da katta va kichik registrni sezish qobiliyati	22
O'zgaruvchilar	23
O'zgaruvchi nima?.....	23
O'zgaruvchilarni nomlash konvensiyasi.....	24
O'zgarmaslar	24
O'zgarmaslar nima?.....	24
Konstantalarni nomlash konvensiyasi	25
echo va print instruksiyalari	26
echo instruksiyasi ichida matn.....	26
echo instruksiyasi ichida HTML kodni ishlatalish	26
echo instruksiyasi orqali o'zgaruvchi qiymatini chiqarish	27
print instruksiyasi	27
Matn satrlarini "print" ichida yozish	28
"print" ichida HTML kodni ishlatalish.....	28
"print" orqali o'zgaruvchi qiymatlarini ko'satish.....	28
Ma'lumot turlari	29
Butun sonlar.....	29
Satrlar.....	30
Kasr sonlar.....	30
Boolean.....	31

Massivlar	31
Obyektlar	32
NULL.....	33
Resurslar	34
Satrlar	34
Satrlarni manipulatsiya qilish	35
Satrni uzunligi hisoblash.....	36
Satrdagi so'zlar sonini hisoblash	36
Satr ichidagi matnni boshqasi bilan almashtirish	36
Satrni teskarisiga o'girish	37
Operatorlar	37
Operatorlar nima?	37
Belgilash operatorlari.....	38
Taqqoslash operatorlari	39
Oshirish va kamaytirish operatorlari.....	40
Mantiqiy operatorlar	41
Satr operatorlari	42
Massiv operatorlari	43
Kosmik kema operatori PHP 7	44
Shart konstruktori	45
If instruksiyasi	45
If...else shart konstruktori	46
If...elseif...else konstruktori	46
Uchlik operatori	47
NULL koordinatsion operatori	48
switch...case konstruktori.....	49
Massivlar	50
Massiv turlari.....	51
Indekslangan massivlar.....	51
Bog'langan massivlar	52
Ko'p o'lchamli massivlar	52
Massiv strukturasini va qiymatlarini ko'rish	53
Massivlarni saralash.....	53

Indekslangan massivlarni o'sish tartibida saralash.....	54
Indekslangan massivni kamayish tartibida saralash	55
Bog'langan massivlarni qiymati bo'yicha o'sish tartibida saralash	55
Bo'glangan massivlarni kaliti bo'yicha o'sish tartibida saralash	56
Bog'langan massivlarni kalitlari bo'yicha kamayish tartibida saralash	57
Sikllar	58
While sikli.....	58
Do...while sikli.....	59
while va do-while o'rtasidagi farq.....	59
For sikli.....	59
foreach sikli	60
Funksiyalar.....	61
Funksiyalarning turlari.....	62
Foydalanuvchi yaratgan funksiyalar.....	62
Funksiyani yaratish va uni chaqirish	63
Parametrli funksiyalar.....	64
Ixtiyoriy parameter va boshlang'ich(defolt) qiymatli funksiyalar.....	65
Funksiya ichidan qiymat qaytarish	65
Havola orqali funksiyaga argument uzatish.....	66
O'zgaruvchi muhiti haqida	67
Anonim funksiyalar	68
"Global" kalit so'zi	69
Rekursiv fuksiyalar yaratish	70
Matematik amallar	71
Sonnig absolyut qiymatini topish	72
Kasrning katta yoki kichik qismini yaxlitlash	73
Sonning kvadrat ildizini topish	73
Tasodifiy sonlarni generatsiya qilish	73
O'nlik sonlarni ikkilikka o'girish yoki teskarisi	74
O'nlik sonlarni o'n otilik sonlarga o'girish yoki aksi.....	74
O'nlik sonlarni sakkizlikka o'girish yoki aksi	75
Sonni bir sanoq tizimidan boshqasiga o'girish.....	75
GET va POST metodlari	76
GET metodi	76

GET metodining afzalliklari va kamchiliklari	77
POST metodi	78
POST metodining afzalliklari va kamchiliklari	78
\$_REQUEST o'zgaruvchisi.....	79
Sana va vaqt	79
Time funksiyasi, timestamp formati	79
Nega timestamp kerak?.....	79
mktime funksiyasi.....	80
Date funksiyasi	81
Date funksiyasining ikkinchi parametric	82
strtotime funksiyasi.....	83
Sanani qanday qo'shish va ayirish.....	84
Include va require	85
Sintaksislari	86
include va require ko'rsatmalari orasidagi farq	86
include_once va require_once ko'rsatmalari	87
Fayl sistemasi.....	89
fopen() funksiyasi	89
fclose() funksiyasi.....	91
fread() funksiyasi	91
Fayl kontentini butunlay o'qish	93
fwrite() funksiyasi.....	95
rename() funksiyasi	97
unlink() funksiyasi	98
Fayl tizimiga oid bo'lgan ba'zi funksiyalar	99
Kataloglarni saralash.....	99
Yangi katalog yaratish	100
Faylni bir joydan boshqa joyga ko'chirish	100
Katalog ichidagi fayllarni ro'yhatlash	101
Ma'lum turdag'i fayllarni ro'yhatlash.....	103
Fayl yuklash	104
Birinchi qadam: Fayl yuklash uchu HTML forma	104
Ikkinchi qadam: Yuklangan faylni qayta ishslash.....	105
Yuqoridagi kodning ta'rifi	107

Cookie'lar bilan ishlash	108
Ta'rif.....	108
Kuki(cookie) nima?	108
Kukini sozlash	108
Kuki qiymatlariga kirish.....	109
Kukini o'chirish.....	111
Sessiyalar	111
Sessiya bilan ishlaymiz.....	112
Sessiya ma'lumotini olish va ma'lumotni saqlash.....	112
Sessiyani o'chirish.....	113
Elektron xat yuborish.....	114
Oddiy matnli pochta yuborish	115
Formatlangan pochta yuborish	116
Filterlar.....	117
Satrni tozalash.....	118
Butun sonni tekshirish	118
IP manzilni tekshirish	119
Elektron pochtani tekshirish va keraksiz belgilardan tozalash	119
URL manzillarni tekshirish va keraksiz belgilardan tozalash.....	120
Oraliqdagi butun sonlarni tekshirish.....	121
Formalar bilan ishlash.....	121
HTMLda forma qanday yaratiladi	122
Forma maydonlari.....	122
Forma namunasi	122
Formadi ma'lumot PHPda qanday olinadi.....	123
Yuborilgandan keyin forma maydonchalarining qiymatlarini saqlash.....	123
Muntazam ifodalar	124
Harflar, sonlar va turli belgilar	125
Belgilarni takrorlash operatori(*,+,:)	126
Guruhash qavslari	127
Maxsus belgilarni ekranlashtirish	128
Cheklovchilar.....	128
Oddiy va maxsus belgilar ro'yhati.....	129

Muntazam ifodalar – II	129
Belgilar guruhi \s, \S, \w, \W, \d, \D	131
Turli xil belgilar guruhining ta'riflari:	132
Kvadrat qavslar '[' va ']'	132
Xususiyatlari:	133
Kiril yozuvining xususiyatlari	134
Satrning boshlanishi '^' va oxiri '\$'	135
Vertikal chiziq yoxud YOKI	135
Bekslesh muammosi	135
Preg_replace da almashinuvlar soni	136
Muntazam ifodalar – III	136
preg_match_all funksiyasi	137
Uchinchi parametr	138
Kartmon	138
Saqlamaydigan qavslar	139
Muntazam ifodalar – IV	140
preg_replace_callback funksiyasi	141
Muntazam ifodalar bilan ishlovchi qo'shimcha funksiyalar	142
preg_grep	142
preg_split	143
Modifikatorlar	143
Ma'lumotlar bazasi bilan ishlash	145
Ma'lumotlar bazasi nima?	145
PhpMyAdmin	145
PhpMyAdmin ga oid masalalar	145
AUTO_INCREMENT	146
O'zgaruvchi turlari	147
PHP da MySql bilan qanday ishlaydi?	147
MB bilan aloqa o'rnatish	147
MB ga so'rov yuborish	148
Ma'lumot bazasidagi xatoliklarni ushlash	149
Kodirovka bilan muammolar	149
Ishlashni sinab ko'ramiz	150
Qanday qilib natijani olish mumkin?	151

MB buyruqlar.....	151
SELECT buyrug'i.....	151
Shartga yana nima yozish mumkin?	153
SELECT bilan ishlash namunalar.....	153
Murakkabroq amallar: OR yoki AND	153
Shartlarni guruhlash.....	154
Ustunlarni tanlash.....	154
INSERT buyrug'I – Bazaha ma'lumot joylashtirish	155
INSERTning boshqacha sintaksisi	155
INSERT orqali ma'lumotni ommaviy kiritish	155
DELETE komandasi.....	156
ORDER BY, LIMIT, COUNT, LIKE buyruqlari	157
ORDER BY – saralash	157
LIMIT – miqdorni cheklash.....	157
COUNT – sonini hisoblaymiz	158
LIKE – qidiruv bilan ishlaymiz	159
Iqtibos belgilari ``	159
Ma'lumot bazasi bilan ishlash bo'yicha mashg'ulot ishlari	160
1-amaliyat.....	160
2-amaliyat.....	164
3-amaliyat.....	173
Tizimga kirish va ro'yhatdan o'tish.....	180
Avtorizatsiya va registratsiya haqida ma'lumot	180
MB orqali sodda avtorizatsiya qilish	181
Topshiriq 1:.....	183
Topshiriq 2:.....	183
Topshiriq 3:.....	183
Avtorizatsiyaga sessiya qo'shish	183
Topshiriq 1:.....	187
Topshiriq 2:.....	187
Topshiriq 3:.....	187
Logaut(saytdan chiqish).....	188
Topshiriq 4:.....	188
Topshiriq 5:.....	188

Ro'yhatdan o'tish yoki registratsiya	189
Topshiriq 1:.....	190
Topshiriq 2:.....	190
Topshiriq 3:.....	191
Topshiriq 5:.....	193
Topshiriq 6:.....	195
Topshiriq 7:.....	198
Ma'lumotlarni tekshirish	199
Topshiriq 1:.....	199
Topshiriq 2:.....	199
Topshiriq 3:.....	200
Topshiriq 4:.....	200
Topshiriq 5:.....	200
Topshiriq 6:.....	200
Topshiriq 7:.....	200
Topshiriq 8:.....	200
Parolni heshlash.....	200
Topshiriq 1:.....	202
Topshiriq 2:.....	202
Registratsiyaga "tuz" qo'shamiz.....	202
Topshiriq 3:.....	205
Avtorizatsiyaga tuz qo'shamiz	205
Topshiriq 4:.....	207
password_hash funksiyasidan foydalanamiz	207
Topshiriq 5:.....	210
Profil va shaxsiy kabinet.....	210
Topshiriq 1:.....	211
Topshiriq 2:.....	211
Shaxsiy kabinet.....	211
Topshiriq 4:.....	213
Parolni almashtirish	213
Topshiriq 5:.....	214
Parolni tasdiqlash.....	215
Topshiriq 6:.....	215
Akkauntni o'chirish	215

Topshiriq 7:.....	216
Kirish huquqi	216
Masalalar yechish	219
Topshiriq 1:.....	219
Topshiriq 2:.....	219
Topshiriq 3:.....	219
Topshiriq 4:.....	219
Topshiriq 5:.....	219
Topshiriq 6:.....	220
Topshiriq 7:.....	220
Topshiriq 8:.....	220
Topshiriq 9:.....	220
Foydalanuvchilarni bloklash.....	220
Ma'lumotlar bazasini normallashtirish	223
Obyektga yo'naltirilgan dasturlash – kirish	225
Obyekt xossalari bilan ishlash	230
Topshiriq 1:.....	232
Topshiriq 2:.....	233
Topshiriq 3:.....	233
Topshiriq 4:.....	233
Topshiriq 5:.....	233
Metodlar bilan ishlash.....	233
Topshiriq 1:.....	234
Metodning parametrlari	234
Topshiriq 2:.....	235
Sinfning xossalariiga \$this orqali murojaat qilish	235
Xossaga ma'lumot kiritish	238
Topshiriq 7:.....	239
Topshiriq 8:.....	239
Topshiriq 9:.....	239
Topshiriq 10:.....	239
Topshiriq 12:.....	239
Topshiriq 13:.....	239
Topshiriq 14:.....	240
Metodga \$this orqali murojaat qilish.....	240

Topshiriq 1:	245
Topshiriq 2:	245
Topshiriq 3:	245
public va private kirish modifikatorlari	245
Amaliyotda qo'llanilishi	247
Topshiriq 1:	251
Topshiriq 2:	251
Topshiriq 3:	251
Topshiriq 4:	251
Topshiriq 5:	251
Topshiriq 6:	251
Obyekt konstruktori	252
Topshiriq 1:	255
Topshiriq 2:	255
Topshiriq 3:	255
Topshiriq 4:	255
Getter(oluvchi) va setter(o'rnatuvchi)lar bilan ishlash	255
Topshiriq 1:	261
Topshiriq 2:	261
Topshiriq 3:	261
Topshiriq 4:	261
Faqat o'qish uchun mo'ljallangan xossalar	261
Topshiriq 1:	265
Topshiriq 2:	265
Sinflarni alohida fayllarda saqlash	265
Topshiriq 1:	267
Topshiriq 2:	267
Obyektlarni massivlarda saqlash	267
Topshiriq 1:	270
Topshiriq 2:	270
Topshiriq 3:	270
Konstruktorda xossalarning boshlang'ich qiymati	270
Topshiriq:	275
Xossani e'lon qilishda uning boshlang'ich qiymati	275
O'zgaruvchan xossa nomlari	283

Xossalar massivi	284
Bog'langan massivda xossa nomlari.....	286
Metodlarning o'zgaruvchan nomlari	287
Obyekt yaratilishi bilanoq metodni chaqirish.....	288
Metodlar zanjiri	290
Sinf metodlar to'plami sifatida	294
Sinfdan meros olish	303
Topshiriq 1:.....	308
Bir nechta avlod sinflar.....	308
Topshiriq 2:.....	310
Avlod sinfdan meros olish	310
Topshiriq 3:.....	311
Topshiriq 4:.....	311
protected – kirish modifikatori	311
Avlod sinfda ota sinfning metodlarini qayta yozish	318
Parent bilan ishslash.....	323
Topshiriq 1:.....	324
Topshiriq 2:.....	325
Avlod sinfda ota sinfning konstruktorini qayta yozish	325
Topshiriq 1:.....	330
Ota sinfning konstruktoridan foydalanamiz	330
Topshiriq 2:.....	334
Topshiriq 3:.....	334
Topshiriq 4:.....	334
Topshiriq 5:.....	334
Obyektni havola orqali uzatish	335
Bir sinfni boshqa sinf ichida qo'lllash	337
Obyektlarni taqqoslash	341
Obyektni sinfga tegishliligini aniqlash	345
Instanceof operatori va meros olish	346
Obyektlar bilan ishlashda ma'lumot turini nazorat qilish	348
Statik metodlar	351
Sinf ichida static metod	352

Amaliyot	353
Topshiriq 1:.....	355
Topshiriq 2:.....	355
Statik xossalar	355
Sinf ichida statik xossa	356
Konstant sinflar	357
Sinf ichida konstantaga murojaat qilish.....	358
Obyekt va sinflar bilan ishlash uchun funksiyalar	359
Polimorfizm	362
Abstrakt sinflar	363
Abstrakt metodlar	366
Obyektga yo'naltirilgan dasturlashda interfeyslar.....	372
Interfeyslarning qo'llanishi.....	375
Topshiriq 1:.....	379
Topshiriq 2:.....	379
Interfeys metodlarida parametrlar.....	379
Topshiriq:.....	382
Interfeysda konstruktor e'lon qilish.....	383
Topshiriq 1:.....	384
Topshiriq 2:.....	385
Interfeyslarning bir-biridan meros olishi.....	385
Interfeyslar va instanceof	386
Bir nechta interfeyslar.....	387
Sinfdan meros olish va interfeysni realizatsiya qilish	391
Interfeyslarda konstanta	395
Interfeyslar bilan ishlash uchun ba'zi kerakli funksiyalar.....	396
Treytlar bilan ishlash	397
Topshiriq 1:.....	401
Bir nechta treytlar	401
Topshiriq 2:.....	401
Topshiriq 3:.....	402

Treytlarda konfliktlarni hal qilish	402
Topshiriq 1:.....	405
Topshiriq 2:.....	406
Kirish modifikatori va treytlar	406
Treytning metodini kirish modifikatorini o'zgartirish.....	407
Metodlarning ustunligi.....	408
Treytlarda abstrakt metodlar	410
Topshiriq:.....	412
Treytlarni treylarda qo'llash	412
Topshiriq:.....	414
Treytlar bilan ishslash uchun maxsus funksiyalar	415
__toString sehrli metodi.....	417
__get sehrli metodi.....	421
__set sehrli metodi	422
Nom maydonlari – kirish	426
Quyi nomlar maydoni	428
Nom maydonlariga murojaat qilishning qisqa usuli.....	430
Nisbiy manzil.....	431
use buyrug'i va nom maydonlari	432
Bir nechta sinflarni ularash	434
use komandasi va nisbiy sinf manzili	435
Topshiriq 1:.....	436
Topshiriq 2:.....	436
Topshiriq 3:.....	437
Topshiriq 4:.....	438
Nom maydonlarida taxallusli sinflar	438
Topshiriq 1:.....	441
Topshiriq 2:.....	441
Sinflarni avtoyuklash	442
Xotima.....	444

Kirish

PHP ingliz tilidan o'zbek tiliga tarjima qilinsa, “*Gipermatnli preprocessor*” degan ma'noni anglatadi. Oldin, PHP qisqartmasi ingliz tilidan **Personal Home Page**(Shaxsiy Bosh Sahifa) so'zidan olingan bo'lib, keyinchalik **PHP:Hypertext Preprocessor**, PHP FAQ-da tasvirlanishicha, bu rekursiv qisqartma ekan. PHP bu dinamik veb sahifalarni yaratish uchun xizmat qiladigan juda mashhur va keng tarqalgan ochiq manbali server tomon skriptlash tili hisoblanadi. PHP dastlab Rasmus Lerdorf tomonidan 1994-yilda yaratilgan. PHP skriptlar serverda ishga tushiriladi va natija sodda HTML sifatida veb-brauzerga yuboriladi. PHP ko'pgina mashhur ma'lumotlar bazasi bilan integratsiya qilinishi mumkin, shu jumladan, MySQL, PostgreSQL, Oracle, Microsoft SQL Server, Sybase va boshqalar. Hozirda PHP'ning eng katta versiyasi bu 7.4x. Lekin ko'p ishlatiladigan versiyasi bu PHP 7. Kitobimizdagi barcha kodlar PHP 7 ning eng so'nggi versiyasiga nisbatan olib yozilgan. PHP hali hanuzgacha eng kuchli va o'rghanish hamda ishlatish uchun oson til hisoblanadi.

Eslatma

Bizning PHP darsimiz sizga PHP skriptlash tilining fundamentallarini o'rghanishga ko'maklashadi va qadamma-qadam murakkabgacha o'rGANAMIZ. Agar siz yangi bo'lsangiz, birinchi boshidan o'rGANISHNI boshlang va har kuni oz-ozdan o'rGANIB oldinga harakat qiling.

PHP bilan nimalarni qilish mumkin?

PHP bilan ko'pgina vazifalarni amalga oshirishingiz mumkin:

- Veb sahifa va fayllarni dinamik ravishda yaratish;
- Serverdagи fayllarni ochish, o'qish,yozish va yopish;
- Veb forma orqali foydalanuvchi haqidagi ma'lumotlar, email, teleraqam va boshqalarni yig'ish;

- Vebsayt foydalanuvchisiga email yuborish;
- Vebsaytingizdan foydalanayotgan tashrif buyuruvchini harakatlarini saqlash, kuzatish;
- Bazadagi ma'lumotni o'zgartirish, o'chirish, saqlash;
- Vebsaytga avtorizatsiyadan o'tmaganlarni cheklash;
- Internet bo'y lab ma'lumotni xavfsiz tashish uchun uni shifrlash.

Bu ro'yhat hali tugamaydi, PHP bilan ko'pgina vazifalarni va ishlarni bajarishingiz mumkin.

PHP ning boshqa tillardan ustunligi

Agar siz server tomon ASP.NET yoki Java kabi tillar bilan tanish bo'lsangiz, siz PHP-ni nima shunchalik obro'liroq qilishi haqida hayratlanayotgan bo'lsangiz kerak. Quyida, nima uchun PHP-ni tanlash kerak degan savolga javoblar keltirilgan:

- **O'r ganish oson:** PHP-ni o'r ganish va ishlatish oson. Veb sohasiga kirib kelgan boshlovchi dasturchilar uchun, PHP eng ma'qul til hisoblanadi. Shuningdek, PHP startap loyihalar uchun ham afzal.
- **Ochiq manbali:** PHP ochiq manbali til. U dunyo bo'y lab barcha dasturchilar tomonidan rivojlantiriladi va ishlab chiqiladi.
- **Portativlik:** PHP-ni turli xil platformalarda, shuningdek Microsoft Windows, Linux, Mac OS va boshqalarda ishga tushirish imkonim mavjud. Hamda u ko'pgina serverlar Apache, IIS va boshqalar bilan mutanosib.
- **Tezkorlik:** PHP-ning eng so'nggi versiyalarida yozilgan skriptlar boshqa skriptlash tillari ASP, RUBY, Python, Java kabi tillardan ko'ra tezroq ish bajaradi.

- **Katta hamjamiyat:** PHP dunyo dasturchilar hamjamiyati tomonidan qo'llab-quvvatlanadi va PHP to'g'risida ma'lumot izlash ancha oson hamda tez.

Boshladik!

Bu yerda siz PHP yordamida dinamik veb-sahifalarni yaratish qanchalik osonligini bilib olasiz. Ishni boshlashdan oldin kod muharririga va HTML va CSS haqida ba'zi bilimlarga ega ekanligingizga ishonch hosil qiling. Agar siz veb sohasini endigina o'rganayotgan bo'lsangiz, unda shu <https://tutorials.uz/tutorial/html> saytdan o'rganishni boshlang. Xo'sh, keling, ko'p gapirgandan ko'ra tezroq boshlaymiz.

Lokal Veb Serverni sozlab olamiz

PHP skript veb serverda ishga tushiriladi. Shuning uchun avvalo har qanday PHP dastur yozishdan oldin siz kompyuteringizga quyidagi dasturlarni o'rnatganingizni tekshirib olishingiz kerak.

- Apache Veb serveri
- PHP tizimi
- MySQL ma'lumotlar serveri

Siz ularni individual tarzda o'rnatishingiz yoki oldindan ozlab qo'yilgan paketlarini Linux va Windows kabi operatsion tizimingizga o'rnatishingiz mumkin bo'ladi. Mashhur oldindan

sozlab qo'yilgan paketlardan biri bu XAMPP, WampServer, OSPanel va Denwer(hozirda ishlatalmaydi, funksionalligi juda chegaralangan).

Bilasizmi?

Facebook, Yahoo, Flickr va Wikipedia PHP dan foydalanib yaratilgan. Katta CMS lar Wordpress, Drupal, Joomla va Magento ham PHP da yaratilgan.

OSPanel - Windows veb dasturlash muhiti. U sizga Apache, PHP va MySQL bilan veb ilovalar yaratish imkonini beradi. Shu bilan birgalikda MySQL boshqaruvi paneli bo'lmish PhpMyAdmin bilan ta'minlaydi. PhpMyAdmin bu ma'lumotlar bazasini veb brauzerda turgan holda oson boshqarish uchun veb ilova hisoblanadi. OSPanelni yuklab olish va o'rnatish bo'yicha ko'rsatmalarni, quyidagi rasmiy veb-saytidan olishingiz mumkin:

<https://ospanel.io>

Birinchi PHP skriptimizni yaratamiz

OSPanelni kompyuteringizga muvaffaqiyatli o'rnatib bo'lganingizdan so'ng, endi keyingi qiladigan ishimiz bu brauzer oynasida oddiygina "Salom, koronavirus!" matnini ko'rsatuvchi PHP skriptimizni yaratish. Yaxshi, endi OSPanelni qayerga o'rnatgan bo'lsangiz o'sha papkaga kiring va domains papkasida o'zingizni lokal veb saytingiz uchun domen papka oching. Masalan, **tutorials.uz**. Yaratib bo'lganingizdan keyin, serverni qayta ishga tushirishingiz zarur.

Endi o'zingizga yoqadigan kod muharririni olib, quyidagi kodni yangi PHP fayl yaratib yozing:

```
<?php  
// Salomlashishni ekranga chiqarish  
echo "Salom, koronavirus!";  
?>
```

Endi faylingizni “koronavirus.php” nomi bilan localhost domenlar papkasidagi o'zingizni domen papkangizga saqlang.

Natijani brauzerda ko'rish uchun, OSPanelni o'rnatgan papkangizda, domains papkasi ichidagi yaratgan domen papkangiz nomini brauzerning URL kiritiladigan joyiga <http://domennomi.uz/koronavirus.php> kirtsangiz kifoya.

PHP odatiy HTML veb sahifa ichida ham biriktirilishi mumkin. Bu HTML dokumentingiz ichida siz PHP instruksiyalarini yozishingiz mumkinligini anglatadi, xuddi quyidagi tartibda:

```
<html>  
<head>  
    <title>Tutorials.uz - onlayn ta'lim platformasi</title>  
</head>  
<body>  
<?php  
// Salomlashishni ekranga chiqarish  
echo "Salom, koronavirus!";  
?>  
</body>  
</html>
```

Keyingi mavzularda yuqoridagi instruksiyalar(PHP sintaksislari nazarda tutilyapti) nimani anglatishini bilib olasiz!

PHP sintaksis

Sintaksis asli qadimgi grek tilidan olingan bo'lib, lingvistikada “*qidalar to'plami*” degan ma'noni anglatadi.

Standard PHP sintaksis

PHP skriptini yozishdan oldin biz uni `<?php` bilan boshlab va oxirini `?>` bilan tugatamiz. Ushbu PHP delimiter `<?php` va `?>` PHP-ga soddagina, “*mana shu delimiterlar orasidagi kod blokini, PHP kod blok deb qabul qil*” deb buyuradi va bu HTML orasida ham bemalol PHP kod yozish imkonini beradi.

```
<?php  
// Salomlashishni ekranga chiqarish  
echo "Salom, koronavirus!";  
?>
```

Har bir PHP instruksiyasi oxirida *nuqtali vergul(;) bilan* tugallanadi – bu PHP dasturiga *ma'lum bir kod vazifasi tugallanganligini* aytadi.

HTML ichida PHP-ni biriktirish

PHP fayllari soda .php kengaytmasiga ega bo'lgan fayllar hisoblanadi. PHP fayli ichida odatiy HTML sahifalarda yozganingiz kabi HTML kod shuningdek, server tomon bajarishi uchun PHP kodlarni yozishingiz mumkin.

```
<html>  
<head>  
    <title>Tutorials.uz – onlayn ta'lim platformasi</title>  
</head>  
<body>  
<?php  
// Salomlashishni ekranga chiqarish  
echo "Salom, koronavirus!";  
?>  
</body>  
</html>
```

Yuqoridagi namunada HTML ichida yaxshi shakllantirilgan dinamik veb sahifa yaratish uchun qanday qilib PHP kodlarni biriktirish ko'rsatilgan. Agar siz brauzer orqali sahifaning ochiq kod manbasini ko'zdan kechiradigan bo'lsangiz, faqat bitta farqni ko'rishingiz mumkin, ya'ni u yerda hech qanday

PHP kod emas balki uni HTML ga o'rin almashtirilganini <p>Salom, koronavirus!</p> ko'rishingiz mumkin. Bu yerda nima bo'ldi? Siz PHP kodingizni ishga tushirganingizda <?php...?> teglar orasidagi kodni bajaradi va qolganlari o'z holicha qoladi. Oxiri veb server to'liq HTML da bo'lgan so'nggi natijani brauzerga qaytarib yuboradi.

PHP izohlar

Izoh bu PHP-ga kerak bo'lmaydigan, ya'ni PHP uni *kod sifatida tan olmaydigan* oddiy matn. Izoh qoldirib ketishning maqsadi bu kodni o'qimishliroq qilish. Bu boshqa dasturchiga(yoki kelajakda kodingizni tahrirlayotganingizda) PHP-da yozgan kodingiz nima vazifani bajaryotganini tushunib olishga yordam beradi.

PHP bir qatorli va ko'p qatorli izohlarni qo'llab quvvatlaydi. Bir qatorli izohni yozish uchun siz satrni ham boshiga ham oxiriga *slesh(//)* yoki *hesh belgi(#)*ni yozish kifoya. Misol uchun:

```
<?php
// Bir qatorli izoh
# Bu ham bir qatorli izoh
echo "Salom, koronavirus!";
?>
```

Biroq ko'p qatorli izohlarni yozish uchun, izohni *sleshdan* keyin *yulduzcha(*)* yozib va izohni oxirini yulduzchadan keyin *slesh(*)* bilan tamomlash kerak, xuddi shunday:

```
<?php
/*
Bu bittadan oshiq
bo'lgan izohlarni
yozishga imkon beradi
*/
echo "Salom, koronavirus!";
?>
```

PHP-da katta va kichik registrni sezish qobiliyati

PHP-da katta registrdagi o'zgaruvchi nomi(*\$rang*), kichik registrdagi o'zgaruvchi nomi(*\$Rang*) bilan bir xil emas. Bu ingliz tilida ***case-sensitivity*** deb ataladi, o'zbek tilida bu “***holat sezgirligi***” deb tarjima qilinadi.

```
<?php
// o'zgaruvchiga qiymatni belgilash
$color = "ko'k";

// o'zgaruvchi qiymatlarini chiqarishga harakat qilamiz
echo "Osmonning rangi " . $color . "<br>";
echo "Osmonning rangi " . $Color . "<br>";
echo "Osmonning rangi " . $COLOR . "<br>";
?>
```

Agar siz yuqoridagi namunaviy kodni ishga tushirishga harakat qilsangiz faqatgina ekranda \$rang o'zgaruvchisining qiymatini(*blue*) ko'rishingiz mumkin va qolgan o'zgaruvchilar \$Rang va \$RANG uchun “*Undefined variable*” warning ogohlantirish beradi. Biroq PHP **kalit so'zlar, funksiyalar va klass nomlari** holatga nisbatan sezgir emas. Ya'ni yuqoridagi natijani aksi. **gettype()** yoki **GETTYPE()** bir xil natijani ya'ni “***string***”(gettype() funksiyasi o'zgaruvchi turini aniqlaydi) ni ko'rsatadi:

```
<?php
// o'zgaruvchiga qiymatni belgilash
$color = "ko'k";

// o'zgaruvchining turini aniqlash
echo gettype($color) . "<br>";
echo GETTYPE($color) . "<br>";
?>
```

O'zgaruvchilar

O'zgaruvchi nima?

O'zgaruvchilar – sonlar, matn satrlariga o'xshash ma'lumotlarni o'zida saqlaydi. O'zgaruvchi qiymatlari script bo'y lab o'zgarishi mumkin. Quyida o'zgaruvchi haqida bilishingiz kerak bo'lgan muhim jihatlar sanab o'tilgan:

- PHP-da, o'zgaruvchi unga qiymat qo'shilmasidan oldin e'lon qilinishi kerak emas. PHP o'zgaruvchini, uning qiymatiga qarab avtomatik tarzda to'g'ri ma'lumot turiga o'giradi.
- O'zgaruvchi e'lon qilinganidan keyin u kod bo'y lab qayta ishlatalishi mumkin.
- Belgilash operatori o'zgaruvchiga qiymat belgilash uchun ishlataladi.

PHP-da o'zgaruvchi **\$ozgaruvchi_nom = qiymat;** kabi e'lon qilinadi.

```
<?php
// o'zgaruvchini e'lon qilish
$txt = "Salom, koronavirus!";
$number = 10;

// qiymatlarni chiqarish
echo $txt; // Natija: Salom, koronavirus!
echo $number; // Natija: 10
?>
```

Yuqoridagi namunada biz 2ta o'zgaruvchi, birinchisi satr qiymat belgilangan, ikkinchisi son qiymat belgilangan o'zgaruvchilar yaratdik. So'ngra biz o'zgaruvchi qiymatlarini **echo** instruksiyasi orqali brauzerga chiqaramiz. PHP **echo** instruksiyasi muntazam brauzerga ma'lumotni chiqarish uchun ishlataladi. Biz bu haqida ko'proq keying darslarimizda to'xtalib o'tamiz.

Yodda tuting!

O'zgaruvchi nomlari holatga sezgir bo'ladi, ya'ni \$x va \$X bular bir xil emas. Shunday ekan, o'zgaruvchi nomlashda ehtiyoj bo'ling.

O'zgaruvchilarni nomlash konvensiyasi

Quyida PHP o'zgaruvchilarni nomlashda amal qilinishi kerak bo'lgan shartlar keltirilgan:

- Barcha o'zgaruvchilar *dollar(\$)* belgisi bilan boshlanib, undan keyin o'zgaruvchi nomi yoziladi.
 - O'zgaruvchi nomi harf bilan yoki *tagchiziq(_)* belgisi bilan boshlanishi shart.
 - O'zgaruvchi nomi raqam bilan boshlanmaydi!
 - O'zgaruvchini nomlashda alfa-raqamli belgilar va *tagchiziq(A-z,0-9, _)* belgilar ishlatilishi mumkin.
 - O'zgaruvchi nomlari orasida bo'shliq bo'lmasligi kerak
-

O'zgarmaslar

O'zgarmaslar nima?

O'zgarmas ya'ni konstanta(constant) bu belgilab qo'yilgan qiymat uchun nom yoki identifikator hisoblanadi. O'zgarmaslar o'zgaruvchilarga o'xshaydi, yagona farqi bu ular belgilangandan keyin ularni o'zgartirib bo'lmaydi.

Konstantalardan qachonki kod yozish jarayonida skript bo'ylab o'zgarmaydigan ma'lumotni saqlash kerak bo'lsa, unda foydalanish foydali. Odatiy foydalanishdagi namunalardan keltirsak, bular ma'lumotlar bazasi

foydalananuvchisi, paroli, veb-sayt manzili, kompaniya yoki veb-sayt nomi va hakozolarni o'z ichiga oladi.

Konstantalar PHP-ning *define()* funksiyasidan foydalaniб yoki *const* kalit so'zidan foydalaniб yaratish mumkin. *define()* funksiyasi 2ta argument qabul qiladi: konstanta nomi va uning qiymati. Konstanta qiymati belgilanishi bilanoq unga konstanta nomi bilan xohlagan paytimiz murojaat qilishimiz mumkin.

```
<?php
// konstantani belgilash
define("SITE_URL", "https://tutorials.uz/");
const SITE_URL = "https://tutorials.uz/";

// konstantadan foydalanish
echo 'Saytimizga xush kelibsiz - ' . SITE_URL;
?>
```

Natija veb-sayt manzili bo'ladi.

Yodda tuting: Konvensiyaga ko'ra, konstantalar nomi odatda katta harflarda yoziladi. Bu ularni kodda o'zgaruvchilardan farqlashga yordam beradi.

Ko'rsatma: O'zgaruvchi o'rniغا konstantaga qiymat belgilasangiz, u qiymat hech qachon o'zgarmasligiga bema'lol ishonch hosil qilishingiz mumkin.

Konstantalarni nomlash konvensiyasi

Konstantalarni nomlash qoidasi ham xuddi o'zgaruvchilarnikiga mos keladi. Lekin bu yerda bitta istisno, konstantalar \$ prefiksini talab qilmaydi.

echo va print instruksiyalari

echo instruksiyasi bitta yoki undan ortiq satrlarni ekranga chop etishi mumkin. Umuman olganda, echo instruksiyasi brauzerga chiqarib bo'ladigan har qanday ma'lumotni chop etoladi, shuningdek satrlar, sonlar, o'zgaruvchi qiymatlari va boshqalar.

echo til konstruksiyasi bo'lib, u odatiy funksiya emas(if instruksiyasi kabi), siz undan qavslarsiz ham foydalanishingiz mumkin, masalan: **echo** yoki **echo()**. Biroq, agar siz **echo** ga bittadan ortiq parametrni uzatmoqchi bo'lsangiz, parametrlar qavslar bilan yopilgan bo'lishi kerak emas.

echo instruksiyasi ichida matn

Quyidagi namunada siz matnni **echo** instruksiyasi yordamida ekranga namoyish qilishni ko'rishingiz mumkin:

```
<?php  
// Salomlashishni ekranga chiqarish  
echo "Salom, koronavirus!";  
?>
```

Yuqoridagi PHP kodning natijasi, "Salom, koronavirus!" bo'ladi.

echo instruksiyasi ichida HTML kodni ishlatalish

Quyidagi namunada esa siz **echo** instruksiyasi yordamida HTML kodni brauzerga chop etishni ko'rishingiz mumkin:

```
<?php  
// HTML kodni namoyish etish  
echo "<h4>Bu soda sarlavha.</h4>";  
echo "<h4 style='color: red;'>Bu ishlov berilgan  
sarlavha.</h4>";  
?>
```

Yuqoridagi PHP koddan quyidagicha natijani olishimiz mumkin:

Bu oddiy sarlavha.

Bu ishlov berilgan sarlavha.

echo instruksiyasi orqali o'zgaruvchi qiymatini chiqarish

Quyidagi namunada echo instruksiyasi yordamida o'zgaruvchining qiymatini ekranga qanday qilib chop etishni bilib olasiz:

```
<?php
// o'zgaruvchilar yaratamiz
$txt = "Salom, koronavirus!";
$num = 123456789;
$colors = ["Qizil", "Yashil", "Ko'k"];

// o'zgaruvchilarni ko'rsatish
echo $txt;
echo "<br>";
echo $num;
echo "<br>";
echo $colors[0];
?>
```

Yuqoridagi PHP koddan quyidagicha natijani olamiz:

```
Salom, koronavirus!
123456789
Qizil
```

print instruksiyasi

Siz print instruksiyasidan foydalangan holda (**echo** ning muqobili) brauzerga natijani chiqarishingiz mumkin bo'ladi. **echo** kabi print ham til konstruksiyasi bo'lib, haqiqiy funksiya emas. Shuningdek, **print** ni ham qavslarsiz ishlatishingiz mumkin: **print** yoki **print()**.

Ikkala **echo** va **print** instruksiyasi ham bir xil vazifani bajaradi faqatgina **print** instruksiyasi bitta satrni chiqaradi va har doim

“1” natija qaytaradi. **echo** hech qanday natija qaytarmaganligi sababli **echo** instruksiyasi **print** instruksiyasidan biroz tezroq hisoblanadi.

Matn satrlarini “print” ichida yozish

Quyidagi namunada siz **print** instruksiyasi orqali matn satrini ekranga qanday qilib namoyish etishni bilib olasiz:

```
<?php  
// Salomlashishni ekranga chiqarish  
Print "Salom, koronavirus!";  
?>
```

Natija “echo” niki bilan bir xil bo’ladi. Hech qanday farq yo’q.

“print” ichida HTML kodni ishlatish

Bu ham xuddi shunday tartibda, ya’ni **echo** instruksiyasi orqali qanday HTML-ni ishlatgan bo’lsak, **print** da ham xuddi shunday qo’llaymiz:

```
<?php  
// HTML kodni namoyish etish  
print "<h4>Bu soda sarlavha.</h4>";  
print "<h4 style='color: red;'>Bu ishlov berilgan  
sarlavha.</h4>";  
?>
```

Natija ham “echo”niki bilan bir xil.

“print” orqali o’zgaruvchi qiymatlarini ko’satish

```
<?php  
// o’zgaruvchilar yaratamiz  
$txt = "Salom, koronavirus!";  
$num = 123456789;  
$colors = ["Qizil", "Yashil", "Ko’k"];  
  
// o’zgaruvchilarni ko’rsatish  
print $txt;  
print "<br>";  
print $num;  
print "<br>";  
print $colors[0];  
?>
```

Natija ham “echo”niki bilan bir xil.

Ma'lumot turlari

O'zgaruvchiga belgilangan qiymat har qanday ma'lumot turlari bo'lishi mumkin, u satr va sondan iborat turlardan tortib to murakkab ma'lumot turlari: massivlar va obyektlargacha bo'lgan turlarni o'z ichiga olishi mumkin. PHP umumiyligi 8ta oddiy ma'lumot turlarini qo'llab-quvvatlaydi:

1. *integer*(butun son),
2. *floating point number* yoki *float*(kasr son),
3. *string*(satr),
4. *boolean*(mantiqiy),
5. *array*(massiv),
6. *object*(obyekt),
7. *resource*(resurs)
8. *NULL*.

Mana shu ma'lumot turlaridan o'zgaruvchi yaratishda ishlataladi. Endi keling har birini batafsil o'r ganib chiqamiz.

Butun sonlar

Integer bu o'nlik kasr son(...,1.2,3.0)dan iborat bo'limgan butun sonlar hisoblanadi. Butun sonlar *decimal(asosi 10)*, *hexadecimal(asosi 16 – 0x prefiksga ega)* yoki *octal(asosi 8 – 0 prefiksga ega)* sanoq sistemalarida ifodalanishi mumkin va o'zini oldiga manfiy yoki musbat ishoralarni ixtiyoriy yozishimiz mumkin.

```
<?php  
$a = 123; // onlik son  
var_dump($a);
```

```
echo "<br>";  
  
$b = -123; // manfiy son  
var_dump($b);  
echo "<br>";  
  
$c = 0x1A; // o'n oltilik son  
var_dump($c);  
echo "<br>";  
  
$d = 0123; // sakkizlik son  
var_dump($d);  
?>
```

Satrlar

Satrlar - bu har bir belgi 1 baytga teng bo'lgan belgilar ketma-ketligi. Satr o'zida 2GB(2147483647 bayt) gacha bo'lgan o'lchamdagি harflar, sonlar, maxsus belgilarni saqlay olishi mumkin.

Satr yaratishning eng sodda usuli bu yakkatirnoq yoki qo'shtirnoq ichiga belgi kiritishdan iborat, masalan: ‘Salom’), qo'shtirnoqni ham ishlatsak bo'ladi, masalan: “Salom”.

```
<?php  
$a = 'Salom, koronavirus!';  
echo $a;  
echo "<br>";  
  
$b = 'Salom, koronavirus!';  
echo $b;  
echo "<br>";  
  
$c = '0\'rik';  
echo $c;  
?>
```

Kasr sonlar

Floating point number (“kasr son”, “dabl” yoki “haqiqiy sonlar” deb ham ataladi) – bu o'nlik yoki kasr sonlar hisoblanib, quyidagicha bo'ladi:

```
<?php  
$a = 1.234;  
var_dump($a);  
echo "<br>";  
  
$b = 10.2e3;  
var_dump($b);  
echo "<br>";  
  
$c = 4E-10;  
var_dump($c);  
?>
```

Eslatma: PHP 5.4+ versiyasidan boshlab ikkilik (asosi 2) sistemada ham butun sonlarni belgilashingiz mumkin

Boolean

Boolean(mantiqiy operator) – bular xuddi vklyuchatelga o'xshaydi. Uning ikkita qiymati bor: 1(true) va 0(false).

```
<?php  
// o'zgaruvchi Rost qiymatini belgilash  
$show_error = true;  
var_dump($show_error);  
?>
```

Massivlar

Massiv – bu bir vaqtning o'zida bittadan oshiq bo'lgan qiymatni o'zida saqlay oladigan o'zgaruvchi. Bu bir-biriga o'zaro bog'liq bo'lgan ma'lumotlarni bir joyga yig'ishda qo'l keladi. Misol uchun, davlatlar ro'yhati yoki shahar nomlari ro'yhati.

Massiv rasmiy ravishda ma'lumot qiymatlarining indekslangan to'plami sifatida belgilanadi. Har bir massiv indeksi(kaliti deb ham ataladi) unikal bo'ladi va tegishli qiymatga murojaat qiladi. Aniqroq qilib aytadigan bo'lsak, kalit ya'ni indeks o'zida qiymat saqlaydi.

```
<?php  
$colors = array("Red", "Green", "Blue");  
var_dump($colors);
```

```

echo "<br>";
$c = array(
    "Red" => "#ff0000",
    "Green" => "#00ff00",
    "Blue" => "#0000ff"
);
var_dump($c);
#####
$c = ["Red", "Green", "Blue"];
var_dump($c);
echo "<br>";

$c = [
    "Red" => "#ff0000",
    "Green" => "#00ff00",
    "Blue" => "#0000ff"
];
var_dump($c);
?>

```

Yodda tuting: Obyekt ichida saqlanadigan elementlari(o'zgaruvchi, konstanta, massiv va hakozo) uning xossalari deb ataladi, ma'lumotni qanday ishlash kerakligiga ma'sul kod ya'ni funksiya esa obyekt metodi deb ataladi.

Obyektlar

Obyekt – bu faqatgina ma'lumot saqlashga emas balki o'sha ma'lumotni qanday qilib qayta ishlash kerakligi haqidagi ma'lumotni o'zida saqlaydigan ma'lumot turi hisoblanadi. Obyekt bu obyektlar uchun shablon vazifasini o'tab beruvchi sinfning alohida ekzempliyari. Obyektlar shu shablonga asoslanib *new* kalit so'zi yordamida yaratiladi.

Har bir obyekt, uning ota sinfiga tegishli bo'lgan barcha metodlari va xossalariiga ham ega bo'ladi. Har bir obyekt ekzempliyari o'zining xossalari va metodlari bilan birgalikda mutlaqo mustaqil va shuning uchun ham bir xil sinfning boshqa obyektlaridan mustaqil ravishda manipulatsiya qilinishi mumkin.

Quyida sinfning obyektini qanday yaratish mumkinligi haqida namuna keltirilgan:

```
<?php
// sinf yaratish
class greeting{
    // xossalari
    public $str = "Salom, koronavirus!";

    // metodlar
    function show_greeting() {
        return $this->str;
    }
}

// sinfdan obyekt yaratish
$message = new greeting;
var_dump($message);
?>
```

NULL

Bu PHP-dagi maxsus **NULL** qiymat bo'lib, o'zida ma'lumot saqlamaydigan ya'ni bo'sh o'zgaruvchilarni belgilash uchun ishlataladi.

NULL tur o'zgaruvchisi bu hech qanday ma'lumotsiz o'zgaruvchi. **NULL** turining yagona qiymati ham null hisoblanadi. **\$var** nomli qiymatsiz o'zgaruvchi yaratilsa, unga avtomatik tarzda null qiymat belgilanadi. Ko'pgina yangi PHP dasturchilari **\$var1 = NULL;** va **\$var2 = “”;** ni bir xil deb xato qilishadi, aslida bu noto'g'ri. Ikkala o'zgaruvchi har xil, **\$var1** null qiymatga ega, **\$var2** esa hech qanday qiymat kiritilmagan satr ekanligini ko'rsatadi.

```
<?php
$a = NULL;
var_dump($a);
echo "<br>";

$b = "Salom, koronavirus!";
$b = NULL;
var_dump($b);
```

?>

Resurslar

Resurs bu tashqi resursga murojaatni saqlaydigan maxsus o'zgaruvchi. Resurs o'zgaruvchilari odatda ochilgan fayllarni va ma'lumotlar bazasiga ulanishni saqlab turadi. *get_resource_type()* funksiyasi yordamida ushbu resurs o'zgaruvchisining turini aniqlashingiz mumkin bo'ladi.

```
<?php
// faylni o'qish rejimida ochish
$handle = fopen("eslatma.txt", "r");
var_dump($handle);
echo "<br>";

// MySQL ma'lumotlar bazasiga defolt sozlamalar bilan ulanish
$link = mysql_connect("localhost", "root", "");
var_dump($link);
?>
```

Satrlar

Satr bu harflar, sonlar, maxsus belgilar hamda arifmetik qiymatlar ketma-ketligi yoki barchasining kombinatsiyasi hisoblanadi. Satrni yaratish uslubi soda, shunchaki qo'shtirnoq yoki yakkatirnoq orasiga belgi yozish kifoya.

```
$my_string = 'Salom, koronavirus';
```

Qo'shtirnoq ishorasini ham ishlatalishingiz mumkin. Biroq yakka va qo'shtirnoq ishoralari ikki xil usulda ishlaydi. Yakkatirnoq bilan o'ralgan satrlar so'zma-so'z ko'rib chiqiladi. Aniqroq qilib aytadigan bo'lsak, yakkatirnoq ichida yozilgan belgi xoh u o'zgaruvchi bo'lsin xoh massiv, faqatgina o'sha o'zgaruvchining o'zi va massivning o'zi ekranga chiqadi xolos. Aksincha, qo'shtirnoq ichida yozilgan o'zgaruvchi, ana

shu o'zgaruvchining qiymati bilan almashtiriladi, shuningdek, ma'lum escape-ketma-ketligini ham qo'llaydi.

Escape-ketma-ketligi almashinuvlari:

- **|n** – yangi qator belgisi bilan almashadi
- **|r** - carriage-return belgisi bilan almashadi
- **|t** - tab belgisi bilan almashadi
- **|\$** - dollar belgisini o'zi bilan almashadi
- **|'** – yakkatirnoqni o'zi bilan almashadi
- **\|** - bitta beksleshni(\) o'zi bilan almashadi

Quyida, yakkatirnoq va qo'shtirnoqli satrlarni farqi tushuntirilgan:

```
<?php
$my_str = 'koronavirus!';
echo "Salom, $my_str!<br>";           // Ekranda: Salom,
koronavirus!
echo 'Salom, $my_str!<br>';           // Ekranda: Salom, $my_str!

echo '<pre>Salom\tkoronavirus!</pre>'; // Ekranda:
Hello\tkoronavirus!
echo "<pre>Salom\tkoronavirus!</pre>"; // Ekranda: Salom
koronavirus!
echo 'O\'rik';                         // Ekranda: O'rik
?>
```

Satrlarni manipulatsiya qilish

PHP o'zining, satrni uzunligi hisoblash, quyisatr(substring) yoki belgilarni topish, satrni bir qismini boshqa belgi bilan almashtirish, alohida satr ajratib olish va ko'pgina boshqalar kabi ichki funksiyalari mavjud.

Satrni uzunligi hisoblash

Satr ichidagi belgilar sonini hisoblash uchun *strlen()* funksiyasi ishlataladi. U shuningdek satr ichidagi bo'sh joylarni ham o'z ichiga oladi.

```
<?php
$my_str = 'Tutorials.uz ga xush kelibsiz!';
// Natija: 29
echo strlen($my_str);
?>
```

Satrdagi so'zlar sonini hisoblash

Satrdagi so'zlar sonini hisoblash uchun biz *str_word_count()* funksiyasini qo'llaymiz:

```
<?php
$my_str = 'Tutorials.uz ga xush kelibsiz!';
// Natija: 5
echo str_word_count($my_str);
?>
```

Satr ichidagi matnni boshqasi bilan almashtirish

Mo'ljallangan satr ichidagi qidirilayotgan so'z yoki belgi o'rinlarini boshqasi bilan almashtirishda *str_replace()* funksiyasi qo'l keladi.

```
<?php
$my_str = 'Tutorials.uz ga xush kelibsiz!';
// Almashtirilgan satrni ekranga chiqarish
echo str_replace("Tutorial.uz","O'zbekiston",$my_str);
?>
```

Natija: "O'zbekiston ga xush kelibsiz!"

Ixtiyoriy ravishda *str_replace()* funksiyasi ichiga 4-argumentni ham kirgizishingiz mumkin, bu argument sizga

satrlar almashinuvi necha marta amalga oshirilganligi haqidagi ma'lumotni beradi, xuddi shunday:

```
<?php
$my_str = 'Tutorials.uz ga xush kelibsiz!';

// Satr almashtirishni bajarish
echo str_replace("Tutorials.uz", "O'zbekiston", $my_str, $count);
//satr almashinuvlar soni
Echo "Satr almashinuvlar soni: $count";
?>
```

Natija: “Satr almashinuvlar soni: 1”

Satrni teskarisiga o'girish

strrev() funksiyasi satrni teskarisini ekranga chiqaradi.

```
<?php
$my_str = 'Tutorials.uz ga xush kelibsiz!';

// satrni teskarisiga chiqarish
echo strrev($my_str);
?>
```

Yuqoridagi kodning natijasi quyidagicha bo'ladi:

!zisbilek hsux ag zu.slairotuT

Operatorlar

Bu darsda siz PHP dasturlash tilida operatorlardan foydalangan holda qanday qilib o'zgaruvchi va qiymatlar ustida ma'lum operatsiyalar bajarish yoki manipulatsiya qilishni o'rGANASIZ.

Operatorlar nima?

Operatorlar – bu PHP protsessorga ma'lum bir ish-harakatni amalga oshirishni buyuradigan ishoralar hisoblanadi.

Misol uchun, qo'shish(+) operatori PHP ga 2ta o'zgaruvchini yoki qiymatni qo'shish kerakligini buyuradi, holbuki katta(>) ishorasi PHP ga ikkita qiymatni taqqoslash kerakligini aytadigan operatordir.

Operator	Ta'rif	Namuna	Natija
+	Qo'shish	\$x + \$y	\$x va \$y yig'indisi
-	Ayirish	\$x - \$y	\$x va \$y ayirmasi
*	Ko'paytirish	\$x * \$y	\$x va \$y ko'paytmasi.
/	Bo'lish	\$x / \$y	\$x va \$y bo'linmasi
%	Modul	\$x % \$y	\$x ning \$yga bo'lingandagi qoldiqi

Quyidagi namunada yuqorida ko'rsatilga arifmetik operatorlarni natijasi ko'rsatilgan:

```
<?php
$x = 10;
$y = 4;
echo ($x + $y); // Natija: 14
echo ($x - $y); // Natija: 6
echo ($x * $y); // Natija: 40
echo ($x / $y); // Natija: 2.5
echo ($x % $y); // Natija: 2
?>
```

Belgilash operatorlari

Belgilash operatorlari qiymatni o'zgaruvchiga tayinlash uchun ishlatiladi.

Operator	Ta'rifi	Namuna	Bilan
=	Tayinlash	\$x = \$y	\$x = \$y

				\$x
+=	Qo'shish belgilash	va	\$x + \$y	= \$x + \$y \$x
--=	Ayirish belgilash	va	\$x - \$y	= \$x - \$y \$x
*=	Ko'paytirish belgilash	va	\$x * \$y	= \$x * \$y \$x
/=	Bo'lish qoldig'ini belgilash	va	\$x / \$y	= \$x / \$y \$x
%=	Bo'lish modulini belgilash	va	\$x % \$y	= \$x % \$y \$x

Yuqoridagi operatsiyalarning amaliyotdagi natijasi:

```
<?php
$x = 10;
echo $x; // Natija: 10

$x = 20;
$x += 30;
echo $x; // Natija: 50

$x = 50;
$x -= 20;
```

Taqqoslash operatorlari

Taqqoslash operatorlari ikkita qiymatni boolean(mantiqiy) uslubda qiyoslash uchun ishlataladi.

Operator	Nomi	Namuna	Natija
<code>==</code>	Teng	<code>\$x ==</code> <code>\$y</code>	\$x va \$y teng bo'lса true
<code>==</code>	Identik(bir xil)	<code>\$x ===</code> <code>\$y</code>	\$x va \$y teng bo'lса va bir xil turga mansub bo'lса true
<code>!=</code>	Teng emas	<code>\$x !=</code> <code>\$y</code>	\$x teng bo'lmasa \$y ga true
<code><></code>	Teng emas	<code>\$x <></code> <code>\$y</code>	\$x teng bo'lmasa \$y ga true
<code>!==</code>	Identik emas	<code>\$x !==</code> <code>\$y</code>	\$x va \$y teng bo'lса va bir xil turga mansub bo'lmasa true
<code><</code>	Kichkina	<code>\$x <</code> <code>\$y</code>	\$x kichik bo'lса \$y dan true
<code>></code>	Katta	<code>\$x ></code> <code>\$y</code>	\$x katta bo'lса \$y dan true
<code>>=</code>	Katta yoki teng	<code>\$x >=</code> <code>\$y</code>	\$x katta yoki teng bo'lса \$y ga true
<code><=</code>	Kichik yoki teng	<code>\$x <=</code> <code>\$y</code>	\$x kichik yoki teng bo'lса \$y ga true

Yuqoridagi operatsiyalarning amaliyotdagi natijasi:

```
<?php
$x = 25;
$y = 35;
$z = "25";
var_dump($x == $z); // Natija: boolean true
var_dump($x === $z); // Natija: boolean false
var_dump($x != $y); // Natija: boolean true
var_dump($x !== $z); // Natija: boolean true
var_dump($x < $y); // Natija: boolean true
var_dump($x > $y); // Natija: boolean false
var_dump($x <= $y); // Natija: boolean true
var_dump($x >= $y); // Natija: boolean false
?>
```

Oshirish va kamaytirish operatorlari

Oshirish/kamaytirish operatorlari ingliz tilida *increment/decrement* deb ataladi. Ular o'zgaruvchilarning

qiymatlarini oshirish/kamaytirish uchun ishlatiladi. Ularning termin nomlarini ingliz tilida berishga qaror qildik.

Operator	Nomi	Natija
<code>++\$x</code>	Pre-increment	\$x ni birga oshiradi, va \$x qaytaradi
<code>\$x++</code>	Post-increment	Birinchi \$x qaytaradi, keyin \$x ni birga oshiradi
<code>--\$x</code>	Pre-decrement	\$x ni birga kamaytiradi, keyin \$x qaytaradi
<code>\$x--</code>	Post-decrement	\$x qaytaradi, keyin \$x ni birga kamaytiradi

Yuqoridagi operatsiyalarning amaliyotdagi natijasi:

```
<?php
$x = 10;
echo ++$x; // Natija: 11
echo $x;    // Natija: 11

$x = 10;
echo $x++; // Natija: 10
echo $x;    // Natija: 11

$x = 10;
echo --$x; // Natija: 9
echo $x;    // Natija: 9

$x = 10;
echo $x--; // Natija: 10
echo $x;    // Natija: 9
?>
```

Mantiqiy operatorlar

Mantiqiy operatorlar odatda shart operatorlarida shartlar hosil qilish uchun ishlatiladi.

Operator	Nomi	Namuna	Natija
and	Va	\$x and \$y	\$x va \$y ikkovi ham rost bo'lsa true
or	Yoki	\$x or \$y	\$x yoki \$y dan biri rost bo'lsa

			true
xor	Yoki, istisno	\$x xor \$y	\$x yoki \$y ikkovi emas, ulardan biri rost bo'lsa true
&&	Va	\$x && \$y	\$x va \$y ikkovi ham rost bo'lsa true
	Yoki	\$x \$y	\$x yoki \$y dan biri rost bo'lsa true
!	Bo'lmasa	! \$x	Agar \$x rost bo'lsa true

Yuqoridagi operatsiyalarning amaliyotdagi natijasi:

```
<?php
$year = 2014;
// Kabisa yili 400ga yoki 4 ga bo'linadi, 100ga emas.
if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 == 0))) {
    echo "$year - kabisa yili";
} else{
    echo "$year - kabisa yili emas";
}
?>
```

Satr operatorlari

Satrlar uchun mo'ljallangan 2ta operator mavjud.

Operator	Ta'rifi	Namuna	Natija
.	Bog'lash	\$str1 . \$str2	\$str1 va \$str2 larning birikishi
.=	Belgilangan qiymatni bog'lash	\$str1 .= \$str2	\$str2 ni \$str1 ga qo'shadi

Quyidagi namunada satr operatorlarini qo'llanilishi keltirilgan:

```
<?php
$x = "Salom";
$y = " koronavirus!";
echo $x . $y; // Natija: Salom koronavirus!
```

```
$x .= $y;
echo $x; // Natija: Salom koronavirus!
?>
```

Massiv operatorlari

Massiv operatorlari massivlarni taqqoslash va ular ustida amallar bajarish uchun ishlataladi:

Operator	Nomi	Namuna	Natija
+	Uyushma	\$x + \$y	\$x va \$y ning uyushmasi
==	Tenglik	\$x == \$y	Agar \$x va \$y da bir xil kalit/qiymat juftligi bo'lsa true qaytaradi
====	Identity	\$x === \$y	Agar \$x va \$y da bir xil kalit/qiymat juftligi va bir xil turga mansub bo'lsa true qaytaradi
!=	Inequality	\$x != \$y	Agar \$x teng bo'lmasa \$y ga true qaytaradi
<>	Inequality	\$x <> \$y	Agar \$x teng bo'lmasa \$y ga true qaytaradi
!==	Non- identity	\$x !== \$y	Agar \$x identik ya'ni, bir xil turga va kalit/qiymat juftligiga ega bo'lmasa true qaytaradi.

Namuna:

```
<?php
$x = [a" => "Qizil", "b" => "Yashil", "c" => "Ko'k"];
$y = ["u" => "Sariq", "v" => "Sabzirang", "w" => "Pushti"];
$z = $x + $y; // $x va $y ning birlashmasi
var_dump($z);
var_dump($x == $y); // Natija: boolean false
var_dump($x === $y); // Natija: boolean false
var_dump($x != $y); // Natija: boolean true
var_dump($x <> $y); // Natija: boolean true
var_dump($x !== $y); // Natija: boolean true
?>
```

Kosmik kema operatori | PHP 7

PHP-ning yangi 7-chi versiyasida **spaceship**^($<=>$) operatori paydo bo'ldi. Bu operator ingliz tilidan o'zbek tiliga tarjima qilinadigan bo'lsa "kosmik kema" deb tarjima qilinadi. Rus tilida bu "космический корабль" deb tarjima qilinadi. Bu operator 2ta ifodani taqqoslash uchun ishlataladi. Bu operator *aralash taqqoslash operatori* deb ham ataladi.

Kosmik kema operatori agar ikkala operandlar teng bo'lsa **0**, agar chapdagi operand katta bo'lsa **1**, aksincha o'ng tomondagi katta bo'lsa **-1** qiymat qaytaradi.

Operator	\Leftrightarrow ekvivalent
<code>\$x < \$y</code>	<code>(\$x <= \$y) === -1</code>
<code>\$x <= \$y</code>	<code>(\$x <= \$y) === -1 (\$x <= \$y) === 0</code>
<code>\$x == \$y</code>	<code>(\$x <= \$y) === 0</code>
<code>\$x != \$y</code>	<code>(\$x <= \$y) !== 0</code>
<code>\$x >= \$y</code>	<code>(\$x <= \$y) === 1 (\$x <= \$y) === 0</code>
<code>\$x > \$y</code>	<code>(\$x <= \$y) === 1</code>

Kosmik kema operatorining qanday ishlashi quyidagi namunada bat afsil keltirilgan:

```
<?php
// Butun sonlarni taqqoslash
echo 1 <= 1; // Natija: 0
echo 1 <= 2; // Natija: -1
echo 2 <= 1; // Natija: 1

// Kasrlarni taqqoslash
echo 1.5 <= 1.5; // Natija: 0
echo 1.5 <= 2.5; // Natija: -1
echo 2.5 <= 1.5; // Natija: 1

// Satrlarni taqqoslash
echo "x" <= "x"; // Natija: 0
echo "x" <= "y"; // Natija: -1
echo "y" <= "x"; // Natija: 1
?>
```

Shart konstruktorlari

Ko'pgina boshqa dasturlash tillari kabi, PHP da ham siz dastur ishlash jarayonida mantiqiy yoki qiyosiy test shartlarining natijalariga asoslangan turli xil amallarni bajaradigan kod yozishingiz mumkin. Bu, siz ifoda shaklida yorost yo yolg'on natija beradigan test shartlarini yaratishingiz mumkin va aynan o'sha natjalarga tayanib ma'lum amallarni bajarishingiz mumkinligini anglatadi.

Shartlar tuzish uchun PHP da turli xil shart konstruktorlari mavjud:

- If konstruktori
- If...else konstruktori
- If...elseif...else konstruktori
- Switch...case konstruktori

Har bir konstruktorni darsimizning keying qismlarida yoritib o'tamiz.

If instruksiyasi

If konstruktori agar ma'lum bir shart rost deb baholansa o'sha kod blokni ishga tushirish uchun ishlatiladi. Bu PHP'ning eng soda shart konstruktori hisoblanib u quyidagicha yoziladi:

```
if(condition){  
    // bajariladigan kod  
}
```

Agar bugungi kun Juma bo'ladigan bo'lsa, ekranga "*Juma muborak bo'lsin!*" degan natija chiqadi:

```
<?php
$d = date("D");
if($d == "Fri") {
    echo "Juma muborak bo'lsin!";
}
?>
```

If...else shart konstruktori

Siz *if* shart konstruktoringizga *else* konstruktori orqali muqobil variant qo'shish bilan uni takomillashtirishingiz mumkin bo'ladi. Agar birinchi belgilangan shart rost deb baholansa va boshqa kod bloki yolg'on deb baholansa unda *if...else* konstruktori birinchi test natijasini beradi yoki aksincha bo'lsa, *else* konstruktoridagi test natijani beradi.

Quyidagi namunada, agar bugungi kun Juma bo'ladigan bo'lsa, *Juma muborak bo'lsin* deb, aksincha bo'lsa *Hayrli kun* deb ekranga chiqaradi:

```
<?php
$d = date("D");
if($d == "Fri") {
    echo "Juma muborak bo'lsin!";
} else{
    echo "Hayrli kun!";
}
?>
```

If...elseif...else konstruktori

If...else...if maxsus konstruktori bir nechta *if...else* konstruktorilarini birlashtirish uchun ishlatiladi.

```
if(1-shart){
    // agar 1-shart rost bo'lsa ushbu kod bajariladi
} elseif(2-shart){
    // agar 1-shart yolg'on bo'lib, 2-shart rost bo'lsa ushbu kod bajariladi
} else{
    // agar 1 va 2-shartlar yolg'on bo'lsa, ushbu kod bajariladi
}
```

Quyidagi namunada agar bugungi kun juma bo'lsa, *Juma muborak bo'lsin!* Deb, agar Shanba bo'lsa, *Shanbalikka bormadingizmi?* Deb, aks holda *Hayrli kun* deb ekranga chiqaramiz.

```
<?php
$d = date("D");
if($d == "Fri") {
    echo "Juma muborak bo'lsin!";
} elseif($d == "Sun") {
    echo "Shanbalikka bormadingizmi?";
} else{
    echo "Hayrli kun!";
}
?>
```

Uchlik operatori

Uchlik operatori ingliz tilida *ternary operator* deb tarjima qilinadi. Uchlik operatori if...else konstruktorlarini qisqa usulda yozishga imkon beradi. Uchlik operatori so'roq belgisi(?) bilan ifodalanadi va u 3ta operand oladi: tekshiriladigan shart, rost bo'lgandagi qiymat va yolg'on bo'lgandagi qiymat.

Maslahat: Uchlik operatoridan foydalangan holda yozilgan kodni o'qish qiyin bo'ladi. Biroq u if...else konstruktorini yozishning ixcham usulini beradi

Operatorni qanday ishlashini batafsil tushunish uchun, quyidagi namunani ko'rib chiqishingiz mumkin bo'ladi:

```
<?php
if($age < 18){
    echo 'Bola'; // Agar 18 yoshdan kichik bo'lsa Bola natijasi
} else{
    echo 'O\'smir'; // Agar 18 ga teng yoki katta bo'lsa,
O'smir natijasi chiqadi.
}
?>
```

Uchlik operatoridan foydalanib, xuddi shu kodni ixcham usulda yozish mumkin bo'ladi:

```
<?php echo ($age < 18) ? 'Bola' : 'O\'smir'; ?>
```

Namunadagi uchlik operatorini tushuntirsam. Agar shart rost deb baholansa,yuqoridagi ikki nuqtaning chap tomonidagi qiymat tanlanadi, aks xolda o'ng tomondagisi.

NULL koordinatsion operatori

PHP 7 da yangi *null koordinatsion operatori(??)* paydo bo'ldi. Bu atama ingliz tilidan olingan bo'lib, ingliz tiliga **null coalescing operator** deb tarjima qilingan. Bu operator orqali siz *isset()* funksiyasi bilan birgalikda *uchlik operatorini* qisqa usulda ishlatishingiz mumkin.

Buni yaxshiroq tushunish uchun quyidagi kodni ko'rib chiqing. Bu *\$_GET['name']* ning qiymatini oladi, agar u mavjud bo'lmasa yoki **NULL** bo'lsa, *defolt qiymat* ni qaytaradi.

```
<?php  
$name = isset($_GET['name']) ? $_GET['name'] : 'defolt qiymat';  
?>
```

Endi yuqoridagini biz o'rgangan null koordinatsion operatoridan foydalanib quyidagicha yozishimiz mumkin:

```
<?php  
$name = $_GET['name'] ?? 'defolt qiymat';  
?>
```

Ko'rganingizdek keying sintaksis yozish uchun juda qulay va ixchamroq.

switch...case konstruktori

Bu konstruktor *if-elseif-else* konstruktorining muqobil varianti bo'lib, deyarli bir xil narsa. *switch-case* konstruktori bitta o'zgaruvchini bir qator qiymatlar orasidan mosini topgunga qadar test qiladi va keyin o'sha natijaga mos keladigan kod bloki bajariladi.

```
switch(n){  
    case nishon1:  
        // agar n=nishon1 bo'lsa, ushbu kod  
        break;  
    case nishon2:  
        // agar n=nishon2 bo'lsa, ushbu kod  
        break;  
    ...  
    default:  
        // agar nga mos qiymat topilmasa, ushbu kod bajariladi  
}
```

Har bir kun uchun alohida xabarni ko'rsatadigan namunani ko'rib chiqishingiz mumkin:

```
<?php  
$today = date("D");  
switch($today) {  
    case "Mon":  
        echo "Bugun Dushanba. Uyingizni tozalang.>";  
        break;  
    case "Tue":  
        echo "Bugun Seshanba. Bozorga chiqing.>";  
        break;  
    case "Wed":  
        echo "Bugun Chorshanba. Doktorga uchrashing.>";  
        break;  
    case "Thu":  
        echo "Bugun Payshanba. Moshinangizni ta'mirlang.>";  
        break;  
    case "Fri":  
        echo "Bugun Juma. Juma namozini o'qing.>";  
        break;  
    case "Sat":  
        echo "Bugun Shanba. Kino ko'ring.>";  
        break;  
    case "Sun":
```

```
echo "Bugun Yakshanba. Dam oling va ertangi kuningizni  
rejalashtiring.";  
    break;  
default:  
    echo "Ushbu kun uchun hech qanday ma'lumot topilmadi.";  
    break;  
}  
?>
```

Ushbu operator **tanlash konstruktori** deb ham ataladi. Tanlash konstruktori ***if-elseif-else*** konstruktorlaridan bir jihat bilan farq qiladi. Bu konstruktor qatorma-qator ko'rib chiqadi va rost natijaga ega bo'lgan holat(case) tanlab olinadi. Faqatgina holat konstruktoriga mos keladigan kodni bajarish emas balki switch blok tugagunga qadar bo'lgan barcha keyingi ***holat*** konstruksiyalarini ishga tushiradi.

Bunday bo'lmasligi uchun biz ***break*** konstrukrusiyasini har bir holat qismining oxiriga qo'shib qo'yamiz. ***Break*** konstruksiyasi PHP ga birinchi rost qiymat topilishi bilan o'sha holat konstruksiyasini bajarib to'xtatish kerakligini aytadi.

Massivlar

Massivlar – bu yagona o'zgaruvchi nomi ostida bitta qiymatdan ko'p yoki qiymatlar guruhini saqlash imkonini beradigan murakkab o'zgaruvchi hisoblanadi. Tasavvur qiling, siz skriptingizda ranglar nomini saqlamoqchisiz. Har bir rang nomini alohida o'zgaruvchilarga saqlaysiz, quyidagicha:

```
<?php  
$color1 = "Qizil";  
$color2 = "Yashil";  
$color3 = "Ko'k";  
?>
```

Ammo siz davlatlar nomi va shtatlarini saqlamoqchisiz, va bu galda 3ta o'zgaruvchi emas balki yuzlab o'zgaruvchilar

yaratishingizga to'g'ri keladi. Har bir shahar nomini alohida o'zgaruvchiga saqlash bu juda qiyin, zerikarli va yomon fikr. Bu holatda bizga massivlar yordamga keladi.

Yodda tuting: Indeksli(raqamli deb ham ataladi) massivda, indekslar avtomatik tarzda belgilanadi va 0 dan boshlanadi va qiymatlar har qanday ma'lumot turi bo'lishi mumkin.

Massiv turlari

Yaratishingiz mumkin bo'lgan 3ta massiv turi mavjud, bular:

- **Indeksli massiv** – raqamli kalitdan iborat massiv
- **Bog'langan massiv** – o'zining konkret kalitlari mavjud bo'lgan massiv
- **Ko'p o'lchamli massiv** – o'zining ichida yana bir yoki undan ko'p bo'lgan massivlarni saqlaydigan massiv.

Indekslangan massivlar

Indeksli yoki raqamli massiv har bir elementni raqamli indeks bilan saqlaydi. Quyidagi namunada indeksli massivni yaratishning 2 xil usuli ko'rsatilgan, eng oson usuli:

```
<?php  
// indeksli massivni yaratish  
$colors = ["Red", "Green", "Blue"];  
?>
```

Bu quyidagi namunaga ekvivalent, ya'ni indekslar qo'lda belgilanadi:

```
<?php  
$colors[0] = "Qizil";  
$colors[1] = "Yashil";  
$colors[2] = "Ko'k";  
?>
```

Bog'langan massivlar

Bog'langan massivda, qiymatlarga tayinlangan kalitlar ixtiyoriy va foydalanuvchi belgilagan satrlar bo'lishi mumkin. Quyidagi namunada indeks sonlar o'rniga kalitlar ishlatilgan massiv keltirilgan:

```
<?php
// Bog'langan
$ages = ["Said"=>22, "Ahmad"=>32, "Samandar"=>28];
?>
```

Quyidagi namuna esa yuqoridagi namunaga ekvivalent, ammo bog'langan massivni yaratishning boshqa usuli ko'rsatilgan:

```
<?php
$ages["Said"] = "22";
$ages["Ahmad"] = "32";
$ages["Samandar"] = "28";
?>
```

Ko'p o'lchamli massivlar

Ko'p o'lchamli massiv – bu har bir elementi massiv bo'lgan va o'sha element o'rnidagi massiv elementi ham massiv bo'lgan massiv. Aniqroq qilib aytadigan bo'lsak, bu massiv ichida element o'rniga ichma-ich joylashgan massivlar jamlanmasi.

```
<?php
// Ko'p o'lchamli massivlarni yaratish
$contacts = [
    [
        "ism" => "Said Ahmadjanov",
        "pochta" => "saidahmad@gmail.com",
    ],
    [
        "ism" => "Sanjar Sobirjonov",
        "pochta" => "sanjar.sobirjonov@rambler.ru",
    ],
    [
        "ism" => "Muhammadali Hakimov",
        "pochta" => "hakimov@gmail.com",
    ]
];
// Ichki qiymatlarni olish
echo "Muhammadali Hakimovning pochtasi: " .
$contacts[2]["pochta"];
```

?>

Massiv strukturasini va qiymatlarini ko'rish

Siz ushbu 2ta **var_dump()** yoki **print_r()** instruksiyasidan foydalanib har qanday massiv strukturasi va uning qiymatlarini ko'rishingiz mumkin bo'ladi.

```
<?php
// Massiv yaratamiz
$cities = ["London", "Parij", "Nyu York"];

// Shaharlar massivini ko'ramiz
print_r($cities);
?>
```

Natija quyidagicha bo'ladi :

```
Array ( [0] => London [1] => Parij [2] => Nyu York)
```

```
<?php
// Massiv yaratish
$cities = ["London", "Parij", "Nyu York"];
// Shaharlar massivini ko'ramiz
var_dump($cities);
?>
```

Natija quyidagicha bo'ladi :

```
array(3) { [0]=> string(6) "London" [1]=> string(5) "Parij"
[2]=> string(8) "Nyu York" }
```

Massivlarni saralash

O'tgan mavzumizda sizlar massivlar haqidagi asosiy bilimlarni egallagan edingiz ya'ni massivlar nima, ularni qanday yaratamiz va strukturasi hamda ularning elementlarini qanday ko'ramiz degan savollar endi sizga begona emas. Siz massivlar bilan har xil amallar bajarishingiz mumkin. Misol uchun uni o'zingiz xohlagan tartibda saralash. Hozirgi darsimiz ham aynan shu misolni ko'rib chiqishdan iborat.

PHP massivlarni turli xil usullarda, shuningdek alifbo yoki raqamlarni o'sish yoki kamayish tartibida saralash uchun mo'ljallangan ko'pgina ichiki funksiyalari bilan yaratilgan. Hozir sizlar bilan saralash uchun eng ko'p tarqalgan funksiyalar bilan tanishib chiqamiz.

- **sort()** va **rsort()** – indekslangan massivlarni saralash uchun
- **asort()** va **arsort()** – bog'langan massivning qiymati bo'yicha saralash uchun
- **ksort()** va **krsort()** – bo'glangan massivlarning kaliti bo'yicha saralash uchun.

Indekslangan massivlarni o'sish tartibida saralash

sort() funksiyasi indekslangan massivning elementlarini o'sish tartibida(harflar uchun alifbo tartibida va raqamlar uchun sonli tartibda) saralash uchun ishlatiladi.

```
<?php
// Massiv yaratamiz
$colors = ["Qizil", "Yashil", "Ko'k", "Sariq"];

// Saralaymiz va chop etamiz
sort($colors);
print_r($colors);
?>
```

print_r() instruksiyasi quyidagi natijani beradi:

```
Array ( [0] => Ko'k [1] => Yashil [2] => Qizil [3] => Sariq )
```

Shu kabi raqamlarni ham o'sish tartibida saralashingiz mumkin:

```
<?php
// Massiv yaratamiz
$numbers = [1, 2, 3.5, 4, 7, 10];

// Saralaymiz va chop etamiz
sort($numbers);
print_r($numbers);
?>
```

Bu namunamizda ham natija quyidagicha bo'ladi:

```
Array ( [0] => 1 [1] => 2 [2] => 3.5 [3] => 4 [4] => 7 [5] => 10 )
```

Indekslangan massivni kamayish tartibida saralash

rsort() funksiyasi indekslangan massivning elementlarini kamayish tartibida(harflar uchun alifbo tartibida va raqamlar uchun sonli tartibda) saralash uchun ishlataladi.

```
<?php
// Massiv yaratamiz
$colors = ["Qizil", "Yashil", "Ko'k", "Sariq"];

// Saralaymiz va chop etamiz
rsort($colors);
print_r($colors);
?>
```

Natija:

```
Array ( [0] => Sariq [1] => Qizil [2] => Yashil [3] => Ko'k )
```

Raqamlarni ham shu tartibda saralaymiz:

```
<?php
// Massiv yaratamiz
$numbers = [1, 2, 3.5, 4, 7, 10];

// Saralaymiz va chop etamiz
rsort($numbers);
print_r($numbers);
?>
```

Natija:

```
Array ( [0] => 10 [1] => 7 [2] => 4 [3] => 2.5 [4] => 2 [5] => 1 )
```

Bog'langan massivlarni qiymati bo'yicha o'sish tartibida saralash

assort() funksiyasi bo'glangan massivning elementlarini qiymatiga ko'ra o'sish tartibida saralaydi. U xuddi sort() funksiyasi kabi ishlaydi biroq qiymatlari saralanayotgan

paytda unga birikkan kalitlar bo'g'anmasi ham saqlanib qolinadi. Ya'ni kalitlar o'rni ham o'zgaradi biroq saralanmaydi.

```
<?php
// Massiv yaratamiz
$age = ["Muhammad"=>20, "Abdulloh"=>14, "Malik"=>45,
"Akbar"=>35];

// massivni qiymatiga ko'ra saralaymiz va chop etamiz
arsort($age);
print_r($age);
?>
```

Natijani **print_r()** instruksiyasi yordamida tekshiramiz:

```
Array
(
    [Malik] => 45
    [Akbar] => 35
    [Muhammad] => 20
    [Abdulloh] => 14
)
```

Bo'glangan massivlarni kaliti bo'yicha o'sish tartibida saralash

ksort() funksiyasi bog'langan massiv elementlarini ularning kaliti bo'yicha o'sish tartibida saralab chiqadi. Bu funksiya kalitlar va uning qiymatlari bog'lanuvini saralash jarayonida saqlab qoladi, xuddi **asort()** funksiyasi kabi.

```
<?php
// Massiv yaratamiz
$age = ["Muhammad"=>20, "Abdulloh"=>14, "Malik"=>45,
"Akbar"=>35];

// massivni kalitiga ko'ra saralaymiz va chop etamiz
ksort($age);
print_r($age);
?>
```

Natijani **print_r()** instruksiyasi yordamida tekshirib ko'ramiz:

```
Array
(
    [Abdulloh] => 14
    [Akbar] => 35
    [Malik] => 45
    [Muhammad] => 20
)
```

Bog'langan massivlarni kalitlari bo'yicha kamayish tartibida saralash

krsort() funksiyasi bog'langan massivning elementlarini uning kalitlariga ko'ra kamayish tartibida saralab chiqadi. Odadagidek, kalitlar va qiymatlar joylashuvi o'zgarmaydi.

```
<?php
// Massiv yaratamiz
$age = ["Muhammad"=>20, "Abdulloh"=>14, "Malik"=>45,
"Akbar"=>35];

// massivni kalitiga ko'ra saralaymiz va chop etamiz
krsort($age);
print_r($age);
?>
```

Natija esa quyidagicha:

```
Array
(
    [Muhammad] => 20
    [Malik] => 45
    [Akbar] => 35
    [Abdulloh] => 14
)
```

Sikllar

Sikllar bir xil kod bo'lagini qayta va qayta ma'lum bir shart qoniqtirilmaguncha ishga ishga tushirish uchun ishlataladi. Sikllar ortida vaqtni va harakatni tejash uchun dastur ichidagi takroriy kodlarni avtomatlashtirish g'oyasi yotadi.

- **while** – belgilangan shart rost deb baholanmaguncha ma'lum kod bo'lagini iteratsiya qiladi
- **do...while** – kod bo'lagi bir marta ishga tushiriladi va keyin shart baholanadi. Agar shart rost deb baholansa unda kod belgilangan shart rost qiymat bo'lgunga qadar takrorlanadi.
- **for** – hisoblagich(schyotchik) belgilangan miqdorga yetguncha ma'lum kod bo'lagini iteratsiya qiladi
- **foreach** – massivdagi har bir element uchun kod bo'lagini iteratsiya qiladi.

While sikli

while instruksiyasi uning ichidagi belgilangan shart rost deb baholangunga qadar kod bo'lagini iteratsiya qiladi. Uning sintaksi:

```
while(shart){  
    // bajariladigan kod bo'lagi  
}
```

Quyidagi namunada **\$i=1** bilan boshlanadigan sikl yaratilgan. Sikl, **\$i** toki 3dan kichik yoki teng bo'lgunga qadar ish faoliyatini davom ettiradi. Har bir iteratsiyada **\$i** birga oshiriladi.

```
<?php  
  
$i = 1;  
  
while($i <= 3) {
```

```
$i++;
echo "Son - " . $i . "<br>";
}
?>
```

Do...while sikli

Bu sikl har bir sikl iteratsiyasining oxirida shart baholanadigan **while** siklining varianti hisoblanadi. **do-while** sikli bilan kod bo'lagini bir marta ishga tushiriladi keyin esa shart baholanadi. Agar shart rost bo'lsa, instruksiya toki shart to'g'ri deb baholangunga qadar takrorlanaveradi.

```
do {
    // bajariladigan kod
}
while(shart);
```

while va do-while o'rta sidagi farq

while sikli **do-while**'dan muhim bir jihat bilan farq qiladi. While sikli bilan, shart har bir siklning boshida test qilinadi, shuning uchun agar shartli ifoda yolg'on deb baholansa, sikl hech qachon ishga tushmaydi.

Do-while sikli bilan, sikl faqat bir martta ishga tushiriladi hatto shartli ifoda yolg'on bo'lsa ham. Chunki shart har doim bu siklda har bir iteratsiyaning boshida emas aksincha oxirida baholanadi.

For sikli

For siklini ma'lum bir shart qoniqtirguncha kod bo'lagini takroraydi. Bu odatda kod bo'lagini bir necha martta bajarish uchun ishlatiladi. Sintaksisi quyidagicha:

```
for(initsiliziatsiya; shart; oshirish) {
    // bajariladigan kod
}
```

for sikli paramaterlarining ta'riflarining ma'nosi:

- **initsializatsiya** – bunda siklni ishga tushirish uchun schyotchik o'zgaruvchi yoziladi va u hech qanday shartsiz sikl tana qismidagi kod bajarilishidan oldin bir martta ishga tushiriladi.
- **Shart** – har bir iteratsiyaning boshida, shart tekshiriladi. Agar shart rost deb baholansa, sikl davom etadi va ichidagi instruksiyalar ishga tushiriladi. Agar shart yolg'on bo'lsa unda sikl faoliyati to'xtaydi.

Quyidagi namunada **\$i=1** bilan boshlanadigan sikl yaratilgan. Sikl, **\$i** toki 3dan kichik yoki teng bo'lgunga qadar ish faoliyatini davom ettiradi. Har bir iteratsiyada **\$i** birga oshiriladi.

```
<?php
for($i=1; $i<=3; $i++) {
    echo "Son - " . $i . "<br>";
}
?>
```

foreach sikli

Bu sikl massivlarni iteratsiya qilish uchun ishlataladi. Uning sintaksisi:

```
foreach($massiv as $qiymat) {
    // bajariladigan kod
}
```

Quyidagi namuna berilgan massivning qiymatlarini chop etadigan siklni ko'rsatadi:

```
<?php
$colors = ["Red", "Green", "Blue"];

// massiv ranglarini iteratsiya qilish
foreach($colors as $color) {
```

```
    echo $color. "<br>";  
}  
?>
```

foreach siklining yana bir sintaksi quyidagicha:

```
foreach($massiv as $kalit => $qiymat) {  
    // bajariladigan kod  
}
```

```
<?php  
$information = [  
    "ism" => "Sanjarbek Sobirjonov",  
    "telegram" => "@SobirjonovSanjarbek",  
    "yosh" => 18  
];  
  
// information massivi bo'ylab iteratsiya qilamiz  
  
foreach($information as $kalit => $qiymat) {  
    echo $kalit. " : " . $qiymat. "<br>";  
}  
?>
```

Natija quyidagicha bo'ladi:

```
ism : Sanjarbeek Sobirjonov  
telegram : @SobirjonovSanjarbek  
yosh : 18
```

Funksiyalar

Funksiya – bu ma'lum bir vazifani bajaradigan mustaqil kod bo'lagi hisoblanadi. PHP-da alohida bir vazifani bajarish uchun skriptingiz ichida to'g'ridan-tog'ri chaqirishingiz mumkin bo'lgan katta ichki funksiyalar to'plami mavjud. Masalan, **gettype()**, **print_r()**, **var_dump()** va hakozolar.

Funksiyalarning turlari

Funksiyalar turlari 2ga bo'linadi:

- **Ichki(standart) funksiyalar** – PHP-ning o'zini ichki funksiyalari
- **Foydalanuvchi belgilagan funksiyalar** – foydalanuvchi tomonidan yaratiladigan funksiyalar.

Foydalanuvchi belgilagan funksiyalar ham 2turga bo'linadi:

- **Anonim funksiyalar**(PHP-ning 5.3.0 versiyasidan beri mavjud)
- **Chiziqli funksiyalar**(PHP-ning 7.4 versiyasida qo'shilgan.)

Foydalanuvchi yaratgan funksiyalar

Ichki maxsus funksiyalarga qo'shimcha sifatida, PHP yana o'zingizning vazifangizdan kelib chiqqan holda funksiyangizni yaratishingizga imkon beradi. Bu ma'lum vazifani bajarish uchun qayta foydalanadigan kodni yaratishning oson usuli va yana bir jihat, bu asosiy fayl(dastur)dan alohida ravishda saqlanishi ham mumkin. Funksiyalardan foydalanishning afzalliklari:

- **Dastur ichidagi takrorlanuvchi kodlarni kamaytiradi** – funksiya sizga umumiyligi ishlataladigan kod bo'laklarini bitta yagona komponentga ajratish imkonini beradi. O'xshash vazifalarga duch kelib qolsangiz, avval yaratib qo'ygan funksiyangizni chaqirasiz tamom vassalom.
- **Kodni saqlashni juda ham osonlashtiradi** – funksiya bir martda yaratilgandan keyin turli xil fayllarga tegmasdan barcha joyda funksiya ichida hech qanday o'zgarishlar qilmasdan uni ko'p marotaba ishlatishimiz mumkin bo'ladi.

- **Xatoliklarni bartaraf etishni osonlashtiradi** – dastur funksiyalarga ajratilganda, agar qandaydir xatolik paydo bo'ladigan bo'lsa siz qaysi funksiya bu xatolikka sababchi bo'layotganligini bilasiz va uni topishingiz osonlashadi.
- **Funksiyalar boshqa ilovalarda ham qayta foydalaniladi** – funksiya qolgan skriptdan ajratib olinganligi tufayli, uni boshqa ilovalarda ham xuddi o'sha funksiya faylini skriptga biriktirgan holda ishlatish mumkin.

Funksiyani yaratish va uni chaqirish

O'z funksiyangizni yaratishda ushbu asosiy sintaksisdan foydalanasiz:

```
function funksiyaNomi() {  
    // bajariladigan kod  
}
```

Foydalanuvchi yaratgan funksiyalar **function** so'zi bilan e'lon qilinadi. Undan keyin **funksiya nomi** -> **qavslar()** va oxiri, **jingalak qavslar{}** ya'ni kod joylashadigan joy yoziladi. Foydalanuvchi belgilagan funksiyaning namunasi, bu namunada bugungi sana ko'rsatiladi:

```
<?php  
// Funksiya yaratamiz  
function whatIsToday(){  
    echo "Bugun " . date('l', mktime());  
}  
// funksiya chaqiriladi  
whatIsToday();  
#####DIQQAT#####  
Bu sana funksiyalarini tushunmayotgan bo'lsangiz, ikkilanmang.  
Biz hali bularni keying darslarimizda o'tamiz. Shunchaki  
funksiya ishlashini kuzating.  
?>
```

Eslatma: Funksiya nomi harf yoki raqamdan iborat bo'limgan tagchiziq belgisi bilan boshlanishi kerak. Funksiya nomlari harflarning katta-kichikligiga sezgir bo'ladi.

Parametrli funksiyalar

Funksiya yaratganingizda ishga tushgan vaqtida kiruvchi qiymat qabul qilishi uchun siz unga parameter belgilashingiz mumkin. Parametrlar funksiya ichida **to'ldiruvchi o'zgaruvchilar** kabi ishlaydi; funksiya ishga tushganda ular qiymatlar(argument) bilan o'rinni almashadi.

```
function myFunc($birinchiParametr, $boshqaParametr) {  
    // bajariladigan kod  
}
```

Funksiyaga xohlaganiningizcha parameter kiritishingiz mumkin. Ammo har bir belgilagan parametringizga mos keladigan argument funksiya chaqirilganda, funksiyaga yuborilishi kerak. getSum() funksiyasi quyida 2ta parameter qabul qiladi. Va funksiyani chaqirganimizda unga 2ta argument ya'ni 2ta butun son qiymatlarini uzatyapmiz.

Ko'rsatma: Argument – bu funksiyaga uzatiladigan qiymat bo'lib, parameter esa argument qabul qiladigan funksiya ichidagi o'zgaruvchi hisoblanadi. Ammo, bu terminlar umumiyligi muamolada o'zgaruvchan, ya'ni argument bu parameter, parameter bu argument.

```
<?php  
// Funksiyani yaratamiz  
function getSum($num1, $num2) {  
    $sum = $num1 + $num2;  
    echo "$num1 va $num2 sonlarining yig'indisi : $sum";  
}  
  
// Funksiya chaqirilyapti  
getSum(10, 20);  
//Natija  
10 va 20 sonlarining yig'indisi: 30  
?>
```

Ixtiyoriy parameter va boshlang'ich(defolt) qiymatli funksiyalar

Ixtiyoriy parameterli funksiyalar ham yaratishingiz mumkin – shunchaki parameter nomidan keyin uni boshlang'ich qiymatga tenglashtirib qo'ying, xuddi shunday:

```
<?php
// Funksiya yaratish
function customFont($font, $size=1.5){
    echo "<p style=\"font-family: $font; font-size:
{$size}em;\">Salom, koronavirusdan qutulgan dunyo!</p>";
}

// Funksiya chaqirilayapti
customFont("Arial", 2);
customFont("Times", 3);
customFont("Courier");
?>
```

Ko'rganingizdek **customFont()** ga bo'lgan 3-chaqiriq ikkin argumentni o'z ichiga olmagan. Bu PHP-ni boshlang'ich etib belgilangan **\$size** parametrining 1.5 qiymatini ishlatalishiga sabab bo'ladi.

Funksiya ichidan qiymat qaytarish

Funksiya return instruksiyasidan foydalanayotgan funksiya chaqirilganda scriptga javoban qiymat qaytaradi. Qiymat har qanday turga mansub bo'lishi mumkin, massivlar va obyektlar ham ichiga kiradi.

```
<?php

function getSum($num1, $num2) {
    $total = $num1 + $num2;
    return $total;
}

echo getSum(5, 10); // Natija: 15
?>
```

Funksiya bir vaqtning o'zida bittadan oshiq qiymat qaytarolmaydi. Ammo siz bir xil bo'lgan natijalarni massiv qaytarish orqali olishingiz mumkin, quyidagi namunadagidek:

```
<?php
function divideNumbers($dividend, $divisor) {
    $quotient = $dividend / $divisor;
    $array = array($dividend, $divisor, $quotient);
    return $array;
}

// Masssivdagi elementlarni o'zgaruvchilarga tayinlash
list($dividend, $divisor, $quotient) = divideNumbers(10, 2);
echo $dividend; // Natija: 10
echo $divisor; // Natija: 2
echo $quotient; // Natija: 5
?>
```

Havola orqali funksiyaga argument uzatish

PHP-da funksiyaga argument uzatishning ikki xil yo'li mavjud, ular:

1. Qiymat orqali
2. Havola orqali

Odatda, funksiya argumentlari qiymat orqali uzatiladi . Agar argument qiymati funksiya ichida o'zgartirilsa, u funksiya tashqarisida ta'sirlanmaydi. Biroq, funksiyaga uning qiymatlarini o'zgartirish uchun imkon berish uchun, ular havola orqali uzatilishi kerak.

Argumentni havola orqali uzatish – funksiya ichida argument nomi oldiga **ampersand(&)** qo'shish orqali amalga oshiriladi. Quyidagidek:

```
<?php

function selfMultiply(&$number) {
    $number *= $number;
    return $number;
```

```
}

$mynum = 5;
echo $mynum; // Natija: 5

selfMultiply($mynum);
echo $mynum; // Natija: 25
?>
```

O'zgaruvchi muhiti haqida

O'zgaruvchilarni PHP skriptining istalgan joyida e'lon qilishingiz mumkin. Lekin, e'lon qilinish joyi o'zgaruvchini ko'ra olish ko'lmini belgilaydi, ya'ni o'zgaruvchiga qayerda kirish va ishlatish mumkinligi. Foydalanish imkoniyati ***o'zgaruvchi muhiti*** deb nomlanadi.

Avvalboshdan, funksiya ichida e'lon qilingan o'zgaruvchilar lokal(mahalliy) va ular funksiya tashqarisidan ko'rinxaydi yoki o'zgaritirib bo'lmaydi, quyidagi namunaga qarang:

```
<?php
function test() {
    $greet = "Salom, dunyo!";
    echo $greet;
}

test(); // Natija: Salom, dunyo!

echo $greet; // undefined variable xatoligini chiqaradi
?>
```

Shunga o'xshab, agar siz tashqaridagi o'zgaruvchini funksiya ichidan kirmoqchi bo'lsangiz yoki funksiya ichiga import qilmoqchi bo'lsangiz, baribir *undefined variable error* xatoligini olasiz:

```
<?php
$greet = "Salom, dunyo!";

// Defining function
function test(){
    echo $greet;
}
```

```
test(); // undefined variable xatoligini chiqaradi
echo $greet; // Natija: Salom, dunyo!
?>
```

Yuqoridagi namunada ko'rganingizdek, funksiya ichida e'lon qilingan o'zgaruvchiga tashqaridan kirib bo'lindi, shunga o'xshab, funksiya tashqarisida e'lon qilingan o'zgaruvchiga, funksiya ichidan kirib bo'lindi. Bu farqlanish asosiy dasturdagi o'zgaruvchilar orqali funksiya ichidagi o'zgaruvchilarga ta'sir qilish imkoniyatini kamaytiradi.

Anonim funksiyalar

Anonim funksiyalar **closures** deb ham ataladi. Bu funksiyani yaratishda funksiyaga hech qanday nom berilmaydi. Ular **callback** parametrlar qiymati sifatida ishlatish juda foydali ammo ularning boshqa holda ham foydalanish mumkin. Bu funksiyalar [Closure](#) sinfidan foydalanib bajariladi.

Closures'lar o'zgaruvchi qiymati sifatida ham ishlatilishi mumkin; PHP avtomatik tarzda bunday ifodalarni Closure ichki sinfi obyektiga o'giradi. Closure'ni o'zgaruvchiga tayinlash boshqa tayinlashlar sintaksi bilan bir xil, bu tayinlash oxirida albatta nuqtali vergul qo'yiladi:

```
<?php
$greet = function($name)
{
    printf("Salom %s\r\n", $name);
};

$greet('Dunyo');
$greet('PHP');
?>
```

Closures'lar o'zgaruvchilarni global muhitdan meros qilib olishi mumkin. Bunday o'zgaruvchilar **use** til konstruktoriga uzatiladi.

```
<?php
$message = 'Salom';

// "use" ishlatalmadi
$example = function () {
```

```
    var_dump($message);
};

$example();

// $message dan meros olish
$example = function () use ($message) {
    var_dump($message);
};
$example();

// o'zgaruvchi qiymati funksiya chaqirilganda emas
// balki yaratilganda meros qilib olinadi.
$message = 'dunyo';
$example();

// Xabarni tozalash
$message = 'Salom,';

// Havola orqali meros olish
$example = function () use (&$message) {
    var_dump($message);
};
$example();

// Global muhitda o'zgargan qiymat funksiya chaqiruvi
// ichida ko'rinishi
$message = 'dunyo!';
$example();

// Closure'lar odatiy argumentlar ham qabul qiladi
$example = function ($arg) use ($message) {
    var_dump($arg . ' ' . $message);
};
$example("Salom");
?>
```

“Global” kalit so’zi

Yuqoridagi namunamizning alternativi, ya’ni muqobili. Bu kalit so’zdan foydalanib siz o’zingiz xohlagan o’zgaruvchingizni xohlagan muhitingizda ishlatishingiz mumkin, xoh funksiya ichida, xoh tashqarisida.

```
<?php

$greet = "Salom, dunyo!";

function test(){
    global $greet;
    echo $greet;
```

```

}

test(); // Natija: Salom, dunyo!
echo $greet; // Natija: Salom, dunyo!

// O'zgaruvchiga yangi qiymat tayinlash
$greet = "Hayr!";

test(); // Natija: Hayr!
echo $greet; // Natija: Hayr!
?>

```

Rekursiv fuksiyalar yaratish

Rekursiv funksiya – bu shart qoniqtirilmaguncha qayta va qayta o'zini-o'zi chaqiradigan funksiya. Rekursiv funksiyalar ko'pincha murakkab matematik hisob-kitoblarni yechish uchun ishlataladi yoki chuqur ichki strukturali ma'lumotlarni qayta-ishlash, misol uchun, ichma-ich joylashib ketgan, aniqroq aytadigan bo'lsak ko'p o'lchamli massivlarni barcha elementlarini chiqarish. Namunani ko'ring:

```

<?php
// Rekursiv funksiya yaratilishi
function printValues($arr) {
    global $count;
    global $items;

    // Kiruvchi qiymat massivligini tekshirish
    if(!is_array($arr)){
        die("ERROR: Massiv emas");
    }

    /*
     Agar qiymat massiv bo'lsa iteratsiya davom etadi,
     funksiyani                                                 rekursiv
     chaqiramiz, aks holda topilgan qiymatni items massiviga qo'sh
     Va har bir topilgan qiymat uchun schyotchikni 1ga oshir.
    */
    foreach($arr as $a){
        if(is_array($a)){
            printValues($a);
        } else{
            $items[] = $a;
            $count++;
        }
    }
}

```

```
// Massivda topilgan qiymatlarni va umumiy sanog'ini qaytar
return ['Umumiy' => $count, 'Qiymatlar' => $items];
}

// Ichma-ich joylashgan massiv yarat
$species = [
    "qushlar" => [
        "Burgut",
        "Chumchuq",
        "Laylak"
    ],
    "sut emizuvchilar" => [
        "Odam",
        "mushuk" => [
            "Sher",
            "Yo'lbars",
            "Yaguar"
        ],
        "Fil",
        "Maymun"
    ],
    "sudralib yuruvchilar" => [
        "ilon" => [
            "Kobra" => [
                "Qirol ko'brasi",
                "Misr ko'brasi"
            ],
            "Qora ilon",
            "Anakonda"
        ],
        "Krokodil",
        "Dinozavr" => [
            "T-rex",
            "Alamazavr"
        ]
    ]
];

$result = printValues($species);
echo $result['total'] . ' qiymatlar topildi: ';
echo implode(', ', $result['values']);
?>
```

Matematik amallar

PHP-da har qanday oddiy qo'shish yoki ayirishdan tortib to murakkab hisob-kitoblargacha bo'lgan amallarni bajarishda yordam beradigan turli xil ichki funksiyalari mavjud. Ularni

asosiy arifmetik operatorlar namunalarini PHP operatorlar mavzusida ko'rgan edingiz. Endi keling ko'proq namunalarni ko'rib chiqamiz:

```
<?php
echo 7 + 3; // Natija: 10
echo 7 - 2; // Natija: 5
echo 7 * 2; // Natija: 14
echo 7 / 2; // Natija: 3.5
echo 7 % 2; // Natija: 1
?>
```

Har bir matematik amallarning ma'lum ustunlik darajalari mavjud, odatiy ko'paytirish va bo'lish amallari qo'shish va ayirishdan oldin bajariladi. Biroq, qavslar bu ustunlikni o'zgartirish mumkin; qavslar ichida kiritilgan ifodalar har doim birinchi hisoblanadi, amalning ustunlik darajasidan qat'iy nazar. Quyidagi namunani ko'ring:

```
<?php
echo 5 + 4 * 10; // Natija: 45
echo (5 + 4) * 10; // Natija: 90
echo 5 + 4 * 10 / 2; // Natija: 25
echo 8 * 10 / 4 - 2; // Natija: 18
echo 8 * 10 / (4 - 2); // Natija: 40
echo 8 + 10 / 4 - 2; // Natija: 8.5
echo (8 + 10) / (4 - 2); // Natija: 9
?>
```

Sonnig absolyut qiymatini topish

Butun son yoki kasr sonning absolyut qiymati **abs()** funksiyasi yordamida topiladi:

```
<?php
echo abs(5); // Natija: 5 (integer)
echo abs(-5); // Natija: 5 (integer)
echo abs(4.2); // Natija: 4.2 (double/float)
echo abs(-4.2); // Natija: 4.2 (double/float)
?>
```

Bunda manfiy raqamlar, musbatga aylanadi. Agar son musbot bo'ladigan bo'lsa, unda funksiya shunchaki o'sha sonni o'zini qaytaradi.

Kasrning katta yoki kichik qismini yaxlitlash

ceil() funksiyasi keying eng katta butun qiymatgacha bo'lgan kasr qiymatini yaxlitlash uchun ishlatsa, **floor()** funksiyasi esa keyingi eng kichik butun qiymatgacha bo'lgan kasr qiymatini yaxlitlaydi:

```
<?php

echo ceil(4.2);      // Natija: 5
echo ceil(9.99);     // Natija: 10
echo ceil(-5.18);    // Natija: -5

echo floor(4.2);     // Natija: 4
echo floor(9.99);    // Natija: 9
echo floor(-5.18);   // Natija: -6
?>
```

Sonning kvadrat ildizini topish

Musbat sonning kvadrat ildizini topish uchun siz **sqrt()** funksiyasidan foydalanishingiz mumkin. Bu funksiya manfiy sonlar uchun maxsus qiymat bo'lgan **NAN**-ni qaytaradi. Misol uchun:

```
<?php
echo sqrt(9);        // Natija: 3
echo sqrt(25);       // Natija: 5
echo sqrt(10);       // Natija: 3.1622776601684
echo sqrt(-16);      // Natija: NAN
?>
```

Tasodifiy sonlarni generatsiya qilish

Tasodifiy sonlarni generatsiya qilish uchun **rand()** funksiyasini qo'llaymiz. Ixtiyoriy ravishda unga minimum va maksimumgacha bo'lgan oralig'larni kiritishingiz mumkin, namuna:

```
<?php
// tasodifiy sonlarni generatsiya qilish
echo rand() . "<br>";
```

```
echo rand() . "<br>";  
  
// 1 dan 10 gacha bo'lgan tasodifiy sonlarni generatsiya  
qilish(10 ham //ichiga kiradi)  
echo rand(1, 10) . "<br>";  
echo rand(1, 10) . "<br>";  
?>
```

Agar bu funksiya **min,max** argumentlarsiz chaqirilsa, u 0 dan **getrandmax()** gacha bo'lgan oraliqdagi psevdo-tasodifiy raqamlar qaytaradi. **getrandmax()** funksiyasi eng katta ehtimoliy tasodifiy sonlarni ko'rsatadi, Windows-da bu 32767. Shuning uchun agar sizga 32767 dan katta son kerak bo'lsa, shunchaki **min** va **max** argumentlarni yozing, kifoya.

O'nlik sonlarni ikkilikka o'girish yoki teskarisi

O'nlik sonlarni ikkilikka o'girish uchun **decbin()** funksiyasi ishlataladi. Holbuki, u, ikkilikdan o'nlikka o'giradigan **bindec()** funksiyasining duplikati hisoblanadi.

```
<?php  
// O'nlikdan ikkilikka  
echo decbin(2);      // Natija: 10  
echo decbin(12);     // Natija: 1100  
echo decbin(100);    // Natija: 1100100  
  
// Ikkilikdan o'nlikka  
echo bindec(10);    // Natija: 2  
echo bindec(1100);   // Natija: 12  
echo bindec(1100100); // Natija: 100  
?>
```

O'nlik sonlarni o'n otilik sonlarga o'girish yoki aksi

dechex() funksiyasi o'nlik sonlarni o'n otilik sonlar ko'rinishga o'girish uchun ishlataladi. Holbuki, **hexdec()** funksiyasi o'n otilik satrdan o'nlik songa o'giradi.

```
<?php  
// O'nlikdan o'n otilikka  
echo dechex(255);   // Natija: ff  
echo dechex(196);   // Natija: c4  
echo dechex(0);     // Natija: 0
```

```
// O'n oltilikdan o'nlikka
echo hexdec('ff'); // Natija: 255
echo hexdec('c4'); // Natija: 196
echo hexdec(0); // Natija: 0
?>
```

O'nlik sonlarni sakkizlikka o'girish yoki aksi

decoct() funksiyasi o'nlik sonni sakkizlik ko'rinishiga o'girish uchun ishlataladi. **octdec()** funksiyasi esa sakkizlik sonni o'nlik songa o'giradi.

```
<?php
// O'nlik sonni sakkizlikka
echo decoct(12); // Natija: 14
echo decoct(256); // Natija: 400
echo decoct(77); // Natija: 115

// Sakkizlikni o'nlikka
echo octdec('14'); // Natija: 12
echo octdec('400'); // Natija: 256
echo octdec('115'); // Natija: 77
?>
```

Sonni bir sanoq tizimidan boshqasiga o'girish

base_convert() funksiyasi sonni bitta sanoq tizimidan boshqa sanoq tizimiga o'girish uchun ishlataladi. Misol uchun, siz o'nlik sonni ikkilikka o'girishingiz , o'n oltilikni sakkizlikka , sakkizlikni o'n oltilikka, o'n oltilikni o'nlikk va hakozoga o'girishingiz mumkin.

Bu funksiya 3 ta parameter qabul qiladi: konvertatsiya qilinadigan son, joriy sanoq tizimi, o'giriladigan sanoq tizimi. Sintaksis quyidagicha:

```
base_convert(son, sanoqtizimidan, sanoqtizimiga);
```

Namuna:

```
<?php
// O'nlikni ikkilikka
echo base_convert('12', 10, 2); // Natija: 1100
```

```
// Ikkilikni o'nlikka
echo base_convert('1100', 2, 10); // Natija: 12

// O'nlikni o'n otilikka
echo base_convert('10889592', 10, 16); // Natija: a62978
// O'n otilikdan o'nlikka
echo base_convert('a62978', 16, 10); // Natija: 10889592

// O'nlikdan sakkizlikka
echo base_convert('82', 10, 8); // Natija: 122
// Convert octal to decimal
echo base_convert('122', 8, 10); // Natija: 82

// O'n otilikdan sakkizlikka
echo base_convert('c2c6a8', 16, 8); // Natija: 60543250
// Sakkizlikdan o'n otilikka
echo base_convert('60543250', 8, 16); // Natija: c2c6a8

// Sakkizlikni ikkilikka
echo base_convert('42', 8, 2); // Natija: 100010
// Convert binary to octal
echo base_convert('100010', 2, 8); // Natija: 42

// O'n otilikni ikkilikka
echo base_convert('abc', 16, 2); // Natija: 101010111100
// Ikkilikni o'n otilikka
echo base_convert('101010111100', 2, 16); // Natija: abc
?>
```

Bu funksiya har doim satr qiymat qaytaradi.

GET va POST metodlari

Odatda veb brauzer server bilan 2ta HTTP(HyperText Transfer Protocol – Gipermatnni uzatish protokoli) metodlardan: **GET** va **POST** dan foydalanib bog'lanadi. Ikkala metod ma'lumotni turlicha uzatadi va afzallik hamda kamchiliklari ham mavjud.

GET metodi

GET metodida ma'lumot URL parametrlar sifatida yuboriladi. Odatda nom satrlari va qiymat juftliklari ampersand(&) bilan ajratilib yoziladi. Umuman, GET ma'lumotli URL quyidagi kabi bo'ladi:

<https://www.tutorials.uz/action.php?name=Sanjarbek&age=24>

URL-ning sabzirang bilan bo'yalgan qismlari GET parametrlari va qizil bilan bo'yalgan qismi esa o'sha parametrarning qiymatlari bo'ladi. URL-ga parameter=qiymat ko'rinishida bittadan ko'p yozish uchun ampersand(&) belgisi bilan har birini ajratib turish kerak. GET metodi orqali faqatgina oddiy matn ma'lumotlarini yuborish mumkin.

GET metodining afzalliklari va kamchiliklari

- Ma'lumot GET metodi orqali yuborilgandan keyin u brauzerning URL qismida aks etadi,
- GET metodi shaxsiy ma'lumotlarni, shuningdek login va parol kabilarni yuborish uchun mos emas, chunki har bir login va parol foydalanuvchining brauzeridagi URL qismida ko'rindi va shuningdek uning brauzerida saqlanib qolinishi mumkin.
- GET metodi ma'lumotni server muhit o'zgaruvchisiga tayinlagani bois, URL uzunligi chegaralangan. Umumiyoj natiladigan ma'lumotga chegara mavjud.

PHP-da URL orqali yoki method="get" dan foydalanilgan HTML forma orqali yuborilgan ma'lumotga kirish uchun `$_GET` superglobal o'zgaruvchisi ishlataladi.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>GET metodi</title>
</head>
<body>
<?php
if(isset($_GET["name"])){
    echo "<p>Salom, " . $_GET["name"] . "</p>";
}
?>
<form method="get" action="<?php echo $_SERVER["PHP_SELF"]; ?>">
    <label for="inputName">Ism:</label>
    <input type="text" name="name" id="inputName">
    <input type="submit" value="Yuborish">
```

```
</form>
</body>
```

POST metodi

POST metodi serverga ma'lumotni yaratish/yangilash uchun yuborish uchun ishlataladi. POST metodi orqali yuborilgan ma'lumotlar brauzerning URL qismida ko'rinxaydi.

POST metodining afzalliklari va kamchiliklari

- GET metodidan ko'ra xavfsiz chunki foydalanuvchi kiritgan ma'lumot hech qachon URL qismida yoki server loglarida ko'rinxaydi
- Katta miqdordagi ma'lumotlarni POST metodi orqali uzatish mumkin va matnli hamda ikkilik ma'lumotlarni(faylni yuklash kabi) uzatish imkonim mavjud.

`$_GET` kabi, PHP-da boshqa superglobal **`$_POST`** o'zgaruvchisi mavjud bo'lib, bu HTML **method="post"** forma orqali yuborilgan yoki POST orqali uzatilgan ma'lumotga kirish uchun ishlataladi.

Eslatma: `$_GET` va `$_POST`, `$_REQUEST` superglobal o'zgaruvchilari PHP ning ichki standart funksiyalari bo'lib, u skriptning barcha joyida mavjud bo'ladi.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>PHP 7 darslar</title>
</head>
<body>
<?php
if(isset($_POST["name"])){
    echo "<p>Salom, " . $_POST["name"] . "</p>";
}
?>
<form method="post" action="<?php echo
$_SERVER["PHP_SELF"]; ?>">
```

```
<label for="inputName">Ism:</label>
<input type="text" name="name" id="inputName">
<input type="submit" value="Yuborish">
</form>
</body>
```

\$_REQUEST o'zgaruvchisi

PHP-da yana bir **\$_REQUEST** superglobal o'zgaruvchisi mavjud bo'lib bu o'zgaruvchi **\$_POST** hamda **\$_GET** superglobal o'zgaruvchilarni, shuningdek **\$_COOKIE** superglobal o'zgaruvchisining qiymatini ham o'z ichiga oladi.

Sana va vaqt

Time funksiyasi, timestamp formati

time() funksiyasi *1-yanvar, 1970-yil* va hozirgi vaqt moment oralig'idagi farqni sekundlarda qaytaradi. Vaqtning bunday ko'rinishi **timestamp** formati deb ataladi.

Nega timestamp kerak?

Timestamp formatidagi vaqt bu sanalar orasidagi farqni sekundlarda topish uchun ishlataladi. Misol uchun, menda joriy yilning 1-yanvar holati uchun taymstemp(timestamp) va aynan hozirgi vaqt holati uchun taymstemp mavjud. Endi men ularni bir-biridan ayirsam, o'sha momentlar orasidagi farqni sekund ko'rinishida olaman. Bu chiqqan farqni 60ga bo'laman va farqni minutlarda olaman(1 minutda 60 sekund bor). Yana 60 ga bo'lsam, natijani soatlarda olishim mumkin. Time funksiyasi yordamida biz faqat hozirgi vaqt momentini olishimiz mumkin. Har qanday sana uchun taymstempni olish uchun **mktime** funksiyasi ishlataladi.

mkttime funksiyasi

Bu **mkttime** funksiyasi time funksiyasi bilan o'xshash ishlaydi, lekin undan argument qabul qilishi bilan farq qiladi: **mkttime(soat, daqiqa,sekund,oy,kun,yil);** (e'tibor bering oy va kun o'rirlari almashgan). Ishlash namunasini ko'ring:

```
<?php
/*
    31 yanvar 2020 yil 12 soat, 43 minut, 59 sekund
holatiga
        bo'lgan taymstempni qaytaradi:
*/
echo mkttime(12, 43, 59, 1, 31, 2020);
?>
```

Parametr oxiridan tashlab ketilishi mumkin. Agar biz yilni yozmasak(oxiridagi parameter) – unda avtomtik tarzda joriy yilni oladi, agar kunni ham yozmasak (oxiridan oldingi parameter) – unda hozirgi yil va hozirgi kunni oladi va hakozo:

```
<?php
/*
    23 yanvar joriy yil 12 soat, 43 minut, 59 sekund
holatiga
        bo'lgan taymstempni qaytaradi:
*/
echo mkttime(12, 43, 59, 1, 31);
?>
```

Odatiy masalalarni time va mkttime funksiyalarida yechamiz. Aytaylik bizga hozirgi vaqt momenti va 1-fevral 2000 yil yarim tun oralig'idagi farqni sekundda topish kerak bo'lzin.

```
<?php
/*
    Time funksiyasi hozirgi vaqt momentini taymstemp formatda
qaytaradi
        mkttime funksiyasi esa berilgan sananing taymstempini
        Natijani bir-biridan ayiramiz va farqni sekundda olamiz:
*/
echo time() - mkttime(12, 0, 0, 2, 1, 2000);
?>
```

Sekundlarda olingan farq **640831489** mana shunday ko'rinishga keladi(sahifani yangilasangiz raqamlar o'zgaradi).

Date funksiyasi

date() funksiyasi berilgan format bo'yicha hozirgi sana va vaqt ni chiqaradi. Format esa maxsus boshqaruv buyruqlari bilan beriladi, bunda ular orasiga turli xil ajratuvchi belgilar qo'yish mumkin, masalan: chiziqcha, ikki nuqta va hakozo).

Buyruqlar ro'yhati(katta harfli buyruqlar bilan kichik harflilarini farqi bor!)

- **U** – 1-yanvar 1970-yildan boshlab bo'lgan sekundlar miqdori
- **z** – yil boshidan hozirgacha bo'lgan kunlar soni
- **Y** – yil, 4 xonali
- **y** - yil,2 xonali
- **m** – oy raqami (oldida noli bilan)
- **n** – oy raqami, nol raqamisiz
- **d** – hozirgi oyning nechanchi kuniligini aniqlaydi
- **j** – 0 bilan boshlanmaydigan oydag i kun raqami
- **w** – hafta kuni (0 – yakshanba, 1 – dushanba va boshqalar).
- **h** – 12-soatlik formatdagi soat
- **H** – 24-soatlik formatdagi soat
- **i** – daqiqa
- **s** – soniya
- **L** – agar kabisa yili bo'lsa 1, aks holda 0.
- **W** – yildagi haftaning tartib raqami
- **t** – ko'rsatilgan oydag i kunlar soni

date bilan ishlash:

```
<?php
    //Barcha namunalar 01.06.2013 12.23.59, dushanba sanasi uchun
    ko'rsatilgan

    echo date('Y'); //Natija '2013'
```

```
echo date('y'); // Natija '13'

echo date('m'); // Natija '06' - oy raqami

echo date('d'); // Natija '01' - oydag'i kun raqami

echo date('j'); // Natija '1' - oydag'i kun raqami (nolsiz)

echo date('w'); // Natija '1' - dushanba

echo date('H'); // Natija '12' - soat

echo date('i'); // Natija '23' - daqiqa

echo date('s'); // Natija '59' - soniya

echo date('d-m-Y'); // Natija '01-06-2013'

echo date('d.m.Y'); // Natija '01.06.2013'

echo date('H:i:s d.m.Y'); // Natija '12:23:59 01.06.2013'
?>
```

Date funksiyasining ikkinchi parametric

date funksiyasining ikkinchi shart bo'limgan parametric mavjud bo'lib, u vaqt momentini taymstemp formatda qabul qiladi. Agar bu parameter uzatilgan bo'lsa, date funksiyasi hozirgi vaqt momentini emas balki uzatilgan argumentni formatlaydi.

```
<?php
echo date('d-m-Y', mktime(0, 0, 0, 12, 29, 13)); //Natija
'29-12-2013'
?>
```

Bundan biz belgilangan sananing hafta kuni raqamini bilishimiz mumkin. Bu sanani biz **date** funksiyasining ikkinchi ixtiyoriy parametriga **mktime()** funksiyasini uzatish orqali o'rnatishimiz mumkin.

Quyidagi namunani ko'rishingiz mumkin:

```
<?php
//29-12-2013 qaysi hafta kuni ekanlgini bilamiz:
```

```
echo date('w', mktime(0, 0, 0, 12, 29, 13)); //Natija '0' -  
yakshanba  
?>
```

strtotime funksiyasi

Keyingi foydali ishlatiladigan foydali funksiyalardan biri bu strtotime deb nomlanadi. **strtotime** funksiyasi – mktime funskiyasining analogi hisoblanib(bu ham taymstemp qaytaradi), farqi faqat bu ancha erkin formatda sanani qabul qiladi.

Misol uchun, unga men ‘2025-12-31’ satrni uzatishim mumkin va funksiya uni o’zi qayerda yil, qayerda oy, qayerda kunligini aniqlaydi va taymstemp formatda sanani qaytaradi. Yana nima ish qilish mumkin: **strtotime(‘now’)** yozsak biz hozirgi vaqt momentini olamiz yoki **strtotime(‘next Monday’)** deb yozsak biz keying dushanba kunini olamiz(Monday – inglizchada dushanba degani).

Hamma format shakllarni ushbu saytdan olishingiz mumkin.

Namuna:

```
<?php  
echo strtotime("now");  
  
echo strtotime("10 September 2000");  
  
echo strtotime("+1 day")  
  
echo strtotime("+1 week");  
  
echo strtotime("+1 week 2 days 4 hours 2 seconds");  
  
echo strtotime("next Thursday");  
  
echo strtotime("last Monday");  
  
echo date('d-m-Y', strtotime('last Monday')); //keyingi  
dushanba kunini sanasini qaytaradi.  
?>
```

Sanani qanday qo'shish va ayirish

Sanaga belgilangan vaqt oralig'ini qo'shish yoki ayirish uchun, 3ta funksiya kombinatsiyasi ishlataladi: **date_create** – bu sanani sana nomli obyektni yaratish orqali ishga tayyorlab turadi(sana yil-oy-kun formatida bo'lishi kerak) va keyin sana **date_modify** orqali manipulyatsiya qilinadi. Bu funksiya orqali kun qo'shish yoki ayirish mumkin. **date_format** – bu ko'rsatilgan formatda sanani ekranga chiqaradi. Bularni to'liq ishlatalish uchun quyidagi namunani yaxshilab o'rganing:

Namuna 1

Keling, 2025 – yil, 12 – oy, 31-kunga oid bo'lgan obyekt yaratamiz keyin unga 1 kun qo'shamiz va uni '**kun.oy.yil**' formatda chiqaramiz.

```
<?php
$date = date_create('2025-12-31');
date_modify($date, '1 day');
echo date_format($date, 'd.m.Y');
?>
```

Natija:

01.01.2026

Namuna 2

Keling, 2025 – yil, 12 – oy, 31-kunga oid bo'lgan obyekt yaratamiz keyin unga 3 kun qo'shamiz va uni '**kun.oy.yil**' formatda chiqaramiz

```
<?php
$date = date_create('2025-12-31');
date_modify($date, '3 days');
echo date_format($date, 'd.m.Y');
?>
```

Natija:

03.01.2026

Namuna 3

Keling, 2025 – yil, 12 – oy, 31-kunga oid bo'lgan obyekt yaratamiz keyin unga 3 kun-u 1 oy qo'shamiz va uni '**kun.oy.yil**' formatda chiqaramiz

```
<?php
    $date = date_create('2025-12-31');
    date_modify($date, '3 days 1 month');
    echo date_format($date, 'd.m.Y');
?>
```

Natija:

03.02.2026

Namuna 4

Keling, 2025 – yil, 1 – oy, 1-kunga oid bo'lgan obyekt yaratamiz keyin unga 1 kunni ayiramiz va uni '**kun.oy.yil**' formatda chiqaramiz

```
<?php
    $date = date_create('2025-01-01');
    date_modify($date, '-1 day');
    echo date_format($date, 'd.m.Y');
?>
```

Include va require

include() va **require()** ko'rsatmalari bir PHP fayl ichidagi kodlarni boshqa PHP fayl ichiga qo'shish uchun ishlataladi. Faylni boshqa fayl ichiga olish ya'ni include qilish bu o'sha fayl ichida yozilgan skriptdan nusxa olib u funksiya chaqirilgan joyga qo'yishdek hisoblanadi.

Fayllarni qo'shish – orqali siz ancha vaqtingizni tejaysiz – biror kod blokini alohida fayl ichida saqlab keyin uni o'zingiz xohlagan joylarda include() va require() funksiyasidan foydalanib fayl ichiga o'sha kod blokini ko'p marta yozmasdan qo'yishingiz mumkin bo'ladi.

Odatiy bo'lgan faylni qo'shish namunalaridan bular – saytning tepe qismi ya'ni header, saytning quyi qismi ya'ni footer va menyular joylashgan fayllarni barcha sahifalarga include() funksiyasidan foydalangan holda bir marta yozib ko'p marta ulashingiz mumkin bo'ladi.

Sintaksislari

```
include("faylnomi"); -yoki- include  
"faylnomi";  
require("faylnomi"); -yoki- require  
"faylnomi";
```

Namuna:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <title>Tutorials.uz</title>  
</head>  
<body>  
<?php include "header.php"; ?>  
<?php include "menu.php"; ?>  
    <h1>Saytimizga xush kelibsiz!</h1>  
    <p>Bu yerda siz dasturlashga oid bo'lgan foydali  
ma'lumotlarni topasiz.</p>  
<?php include "footer.php"; ?>  
</body>  
</html>
```

include va require ko'rsatmalari orasidagi farq

Siz o'ylashingiz mumkin, **include()** funksiyasi orqali biz faylni boshqa fayl ichiga olish mumkin bo'lsa, bizga **require()** funksiyasi unda nega kerak? Odatda **require()** funksiyasi **include()** funksiyasiga o'xshab ishlaydi.

Bu **include()** funksiyasining farqi agar qo'shilayotgan fayl topilmasa PHP warning generatsiya qiladi lekin skript ishlashda davom etadi, holbuki **require()** funksiyasi fatal error generatsiya qiladi va skriptni to'xtatadi. Butkul. Faqat xatolik ko'rindi.

```
<?php require "my_functions.php"; ?>
<!DOCTYPE html>
<html lang="en">
<head>
    <title><?php displayTitle($home_page); ?></title>
</head>
```

Maslahat: require() funksiyasini agar siz kutubxona fayllarini yoki dasturingizni ishlashi uchun muhim ahamiyatga ega bo'lgan funksiya va konfigrationsh o'zgaruvchilarni qo'shayotgan bo'lsangiz, ishlatish tavsiya etiladi.

```
<body>
<?php include "header.php"; ?>
<?php include "menu.php"; ?>
    <h1>Saytimizga xush kelibsiz!</h1>
    <p>Bu yerda siz dasturlashga oid bo'lgan qiziqarli
ma'lumotlar topasiz.</p>
<?php include "footer.php"; ?>
</body>
</html>
```

include_once va require_once ko'rsatmalari

Agar tasodifan faylingizga 1 tadan oshiq bo'lgan bir xil fayllarni include yoki require funksiyalaridan foydalangan holda qo'shib yuborsangiz u konfliktlarga sabab bo'lishi mumkin. Bu holatni oldini olish uchun, PHP bizga **include_once** va **require_once** funksiyalarini taqdim etadi. Bu funksiyalar include va require funksiyalari bilan bir xil faqatgina bitta istisno mavjud.

include_once va **require_once** funksiyalari agar fayl 2-martada chaqirilsa ham, u bir marta qo'shiladi. Agar

belgilangan fayl allaqachon qo'shilgan bo'lsa, unda undan keyin qo'shilgan fayl bekor qilinadi. Buni yaxshilab tushunishingiz uchun quyidagi my_function fayli ichidagi kodga diqqatingizni jalb qilamiz:

```
<?php
function multiplySelf($var) {
    $var *= $var;      echo $var;
}
?>
```

Endi bu yerda esa my_function faylidagi kod qo'shilgan skriptga qarang:

```
<?php
// Faylni qo'shyapmiz
require "my_functions.php";
// Funksiyani chaqiryapmiz
multiplySelf(2); // Natija: 4
echo "<br>";

// Yana bir marotaba qo'shyapmiz
require "my_functions.php";
// Funksiyani chaqiryapmiz
multiplySelf(5); // Bajarilmadi
?>
```

Bu kodimiz bizga "**Fatal error: Cannot redeclare multiplySelf()**" nomli xatolik beradi. Chunki bir xil fayl 2 marta belgilandi. Endi require_once() bilan ishlab ko'ramiz:

```
<?php
// Faylni qo'shyapmiz
require_once "my_functions.php";
// Funksiyani chaqiryapmiz
multiplySelf(2); // Natija: 4
echo "<br>";

// Yana bir marotaba qo'shyapmiz
require_once "my_functions.php";
// Funksiyani chaqiryapmiz
multiplySelf(5); // Natija: 25
?>
```

Fayl sistemasi

PHP server tomon dasturlash tili bo'lganligi sababli, u bizga serverda saqlanayotgan fayllar va papkalar bilan ishlashga imkon beradi. Bu mavzuda siz veb serveringizda PHP fayl sistema funksiyalaridan foydalangan holda qanday qilib fayllarni manipulatsiya qilish, kirish va yaratishni o'rganasiz.

Unda kettik!

fopen() funksiyasi

Fayl bilan birinchi ishlashda albatta siz faylni ochib olishingiz kerak bo'ladi. PHP **fopen()** funksiyasidan foydalanib faylni ochish mumkin. Uning soda sintaksi quyida berilgan:

```
fopen(faylnomi, rejim)
```

Bu funksiyaga birinchi uzatiladigan parameter bu siz ochmoqchi bo'lgan faylning nomi uchun va keyingi parameter esa faylni qaysi rejimda ochish kerakligi kiritiladi. Masalan:

```
<?php
$handle = fopen("malumot.txt", "r");
?>
```

Fayl quyidagi rejimlarda ochilishi mumkin:

Rejimlar	Vazifasi
r	Faylni faqat o'qish uchun ochadi
r+	Faylni faqat o'qish va yozish uchun ochadi
w	Faylni faqat yozish uchun ochadi va fayldagi ma'lumotlarni o'chiradi. Agar fayl mavjud bo'lmasa, faylni yaratadi.

Rejimlar	Vazifasi
w+	Faylni o'qish va yozish uchun ochadi va fayldagi ma'lumotlarni o'chiradi. Agar fayl mavjud bo'lmasa, faylni yaratadi.
a	Biriktiradi. Faylni faqat yozish uchun ochadi. Ma'lumotni faylni oxiriga qo'shgani uchun fayldagi ma'lumotlar o'chmaydi. Fayl mavjud bo'lmasa. Uni yaratadi.
a+	O'qiydi va biriktiradi. Faylni o'qish va yozish uchun ochadi. Ma'lumot faylni oxiriga qo'shadi, fayldagi ma'lumotlar o'chmaydi. Fayl mavjud bo'lmasa. Uni yaratadi.
x	Faylni faqat yozish uchun ochadi. Agar fayl mavjud bo'lsa xatolik chiqaradi va FALSE qaytaradi. Mavjud bo'lmasa, uni yaratadi.
x+	Faylni o'qish va yozish uchun ochadi, 'x' rejim bilan bir xil harakat qiladi.

Agar mavjud bo'lмаган faylni ochishga urinsangiz, PHP ogohlantirish xabari yaratadi. Bunday xatolik xabarlarini oldini olish uchun siz har doim unga kirmasdan oldin fayl yoki direktoriya mavjudmi yoki yo'q shuni tekshirib olishingiz kerak bo'ladi. Buni **file_exists()** funksiyasi amalga oshiradi.

```
<?php
$file = "malumot.txt";
// Faylni mavjudligini tekshiring
if(file_exists($file)){
// Ochishga urunib ko'ring
$handle = fopen($file, "r");
} else{
```

```
echo "ERROR: Fayl mavjud emas.";  
}  
?>
```

fclose() funksiyasi

Fayl bilan ish bitgandan so'ng uni albatta yopishingiz kerak(shart emas). PHP'dagi fclose() funksiyasi faylni yopish uchun xizmat qiladi. Namunaga qaranang:

Eslatma: PHP script tugallanganda barcha fayllarni o'zi avtomatik tarzda yopsa ham, lekin tegishli fayl ish faoliyati yakunlangandan so'ng uni yopish bu yaxshi amaliyot hisoblanadi.

```
<?php  
  
$file = "data.txt";  
  
// Fayl mavjudligini tekshiring  
  
if(file_exists($file)) {  
  
// Faylni o'qish uchun oching  
  
$handle = fopen($file, "r") or die("ERROR: Faylni ochib  
bo'lmasdi.");  
  
/* Qanaqadir kod yozasiz */  
  
// Fayl dastagini yoping  
  
fclose($handle);  
  
} else{  
  
echo "ERROR: Fayl mavjud emas.";  
  
}  
?>
```

fread() funksiyasi

Faylni qanday ochish va yopishni tushundingiz. Bu bo'limda siz fayldan ma'lumotni qanday o'qishni o'rganasiz. PHP da fayldan ma'lumotni o'qish uchun turli xildagi funksiyalari

mavjud. Siz butun boshli fayldan bitta belgini yagona operatsiya orqali amalga oshirishingiz mumkin.

Belgilangan miqdorli belgilarni o'qish

fread() funksiyasi belgilangan miqdordan iborat bo'lgan belgilarni fayldan o'qish uchun ishlataladi. Uning asosiy sintaksiyi quyidagicha:

```
fread(fayl dastagi, baytdagi uzunligi)
```

Bu funksiya 2ta parameter oladi – fayl dastagi va o'qilishi kerak bo'lgan baytlar soni. Quyidagi namunada “malumot.txt” faylidan 20 baytini(bo'shliqlar ichiga kiradi) o'qish keltirilgan. Tasavvur qiling, “ma'lumot.txt” fayli “Tutorials.uz” bu O'zbekiston bo'yicha yagona katta o'quv platformasi” matnini o'z ichiga oladi.

```
<?php

$file = "malumot.txt";

// Fayl mavjudligini tekshiring

if(file_exists($file)) {

// faylni o'qish uchun oching

$handle = fopen($file, "r") or die("ERROR: Faylni ochib
bo'lmasdi.");

// Fayldan belgilangan baytli ma'lumotni o'qing

$content = fread($handle, "20");

// Fayl dastagini yoping

fclose($handle);

// Fayl kontentini chiqaring

echo $content;

} else{
```

```
echo "ERROR: Faylni ochib bo'lindi.";  
}  
?>
```

Natija quyidagicha bo'ladi:

```
Tutorials.uz bu O'
```

Fayl kontentini butunlay o'qish

fread() funksiyasi **filesize()** funksiyasi bilan birga ishlatalishi ham mumkin. Bu funksiyalar jamlanmasi bizga fayl ichidagi barcha ma'lumotlarni o'qish uchun imkoniyat beradi.

```
<?php  
  
$file = "malumot.txt";  
  
// Fayl mavjudligini tekshiring  
  
if(file_exists($file)){  
  
// faylni o'qish uchun oching  
  
$handle = fopen($file, "r") or die("ERROR: Faylni ochib  
bo'lindi.");  
  
// Fayldan belgilangan baytli ma'lumotni o'qing  
  
$content = fread($handle, filesize($file));  
  
// Fayl dastagini yoping  
  
fclose($handle);  
  
// Fayl kontentini chiqaring  
  
echo $content;  
}  
else{  
  
echo "ERROR: Faylni ochib bo'lindi.";  
}  
?>
```

Yuqoridagi kodimizning natijasi:

Tutorials.uz bu O'zbekiston bo'yicha yagona katta o'quv platformasi

Fayl kontentini to'liq o'qishning oson usuli bu **readfile()** funksiyasidan foydalanishdir. Bu funksiya sizga faylni ochmasdan turib uni ichidagi barcha ma'lumotlarni o'qishga imkon beradi.

```
<?php  
  
$file = "malumot.txt";  
  
// Fayl mavjudligini tekshiring  
  
if(file_exists($file)) {  
  
    readfile($file) or die("ERROR: Faylni ochib bo'lmadi.");  
  
} else{  
  
echo "ERROR: Faylni ochib bo'lmadi.";  
  
}  
  
?>
```

Buning natijasi ham yuqoridagi bilan bir xil.

Yana bir boshqa yo'li bu fayldagi ma'lumotni uni ochmasdan turib **file_get_contents()** funksiysi bilan olish mumkin. Funksiya fayl nomi yoki fayl manbasini qabul qiladi va uni to'laligicha satr o'zgaruvchisiga o'qiydi. Misol uchun:

```
<?php  
  
$file = "malumot.txt";  
  
// Fayl mavjudligini tekshiring  
  
if(file_exists($file)) {  
  
    $content = file_get_contents($file) or die("ERROR: Faylni  
ochib bo'lmadi.");  
  
    echo $content;  
  
} else{
```

```
echo "ERROR: Faylni ochib bo'lmedi.";  
}  
?>
```

Fayldagi barcha ma'lumotlarni o'qish uchun yana bir metod mavjud bo'lib, bu metod **file()** funksiyasi hisoblanadi. Bu ham **file_get_contents()** funksiyasi kabi bo'lib, biroq bu fayldan olingan ma'lumotlarni massivini qaytaradi.

```
<?php  
  
$file = "malumot.txt";  
  
// Fayl mavjudligini tekshiring  
  
if(file_exists($file)) {  
  
    $arr = file($file) or die("ERROR: Faylni ochib bo'lmedi.");  
  
    foreach($arr as $line) {  
  
        echo $line;  
  
    }  
}  
else{  
  
    echo "ERROR: Faylni ochib bo'lmedi.";  
}  
?>
```

fwrite() funksiyasi

Yuqoridagilar kabi, siz PHPning **fwrite()** funksiyasidan foydalangan holda mavjud faylga ma'lumotni qo'shishingiz mumkin. Uning sintaksi quyidagicha:

fwrite(fayl dastagi, satr)

Bu funksiya 2ta parameter qabul qiladi – fayl dastagi va faylga qo'shiladigan ma'lumot satri, quyidagi namunadagidek:

```
<?php

$file = "eslatma.txt";

// Yozilishi kerak bo'lgan satr

$data = "Aytgancha, saytda pullik kurslar va kitoblar ham bor
☺";

// faylni yozish uchun oching

$handle = fopen($file, "w") or die("ERROR: Faylni ochib
bo'lmedi.");

// Faylga ma'lumotni joylang

fwrite($handle, $data) or die ("ERROR: Faylga ma'lumot yozib
bo'lmedi.");

// Fayl dastagini yoping

fclose($handle);

echo "Ma'lumot faylga muvaffaqiyatli yozildi.";

?>
```

Yuqoridagi namunada, agar “eslatma.txt” fayl mavjud bo'lmasa, PHP uni avtomatik tarzda yaratadi va ma'lumotni yozadi. Ammo, agar “eslatma.txt” fayl mavjud bo'lsa, PHP undagi ma'lumotlarni o'chiradi va o'rniغا yangi kiritilgan ma'lumotni qo'shami. Agar siz o'chirilmasligini xohlasangiz, yuqorida o'tilgan rejimlarni keraklisini topib qo'llashingiz mumkin.

Muqobil variant ham mavjud. Bu **file_put_contents()** funksiyasıdir. Bu funksiya **file_get_contents()** funksiyasining o'xshash varianti va bu faylga ma'lumotni uni ochmasdan yozish imkoniyatini beradi.

```
<?php

$file = "eslatma.txt";

// Yozilishi kerak bo'lgan satr
```

```
$data = "Aytgancha, saytda pullik kurslar va kitoblar ham bor  
☺";  
  
// Faylga ma'lumotni joylang  
  
file_put_contents($file, $data) or die ("ERROR: Faylga ma'lumot  
yozib bo'lindi.");  
  
echo "Ma'lumot faylga muvaffaqiyatli yozildi.";  
  
?>
```

Agar file_put_contents() funksiyasi ichida belgilangan fayl mavjud bo'lsa, PHP uni standart qayta yozadi. Agar siz fayl kontentlarini saqlab qolishni istasangiz, siz maxsus **FILE_APPEND** flegini uchinchi parameter sifatida file_put_contents() funksiyasiga uzatishingiz mumkin. U yangi ma'lumotni oddiygina faylga uni shunchaki qo'shadi. Namuna:

```
<?php  
  
$file = "eslatma.txt";  
  
// Yozilishi kerak bo'lgan satr  
  
$data = "Aytgancha, saytda pullik kurslar va kitoblar ham bor  
☺";  
  
// Faylga ma'lumotni joylang  
  
file_put_contents($file, $data, FILE_APPEND) or die ("ERROR:  
Faylga ma'lumot yozib bo'lindi.");  
  
echo "Ma'lumot faylga muvaffaqiyatli yozildi.";  
  
?>
```

rename() funksiyasi

Fayl yoki direktoriya nomini PHP'ning rename() funksiyasi bilan o'zgartirishingiz mumkin, xuddi shu kabi:

```
<?php  
  
$file = "fayl.txt";
```

```
// Faylni mavjudligini tekshirish  
  
if(file_exists($file)) {  
  
    // Faylni nomini o'zgartirishga urunish  
  
    if(rename($file, "yangifayl.txt")) {  
  
        echo "Faylni qayta nomlandi.";  
  
    } else {  
  
        echo "ERROR: Faylni nomlab bo'lmadi.";  
  
    }  
  
} else {  
  
    echo "ERROR: Fayl mavjud emas.";  
  
}  
  
?>
```

unlink() funksiyasi

Fayl yoki direktoriyalarni PHP'ning **unlink()** funksiyasi yordamida o'chirishingiz mumkin:

```
<?php  
  
$file = "fayl.txt";  
  
// Faylni mavjudligini tekshirish  
  
if(file_exists($file)) {  
  
    // Faylni nomini o'zgartirishga urunish  
  
    if(unlink($file, "yangifayl.txt")) {  
  
        echo "Fayl o'chirildi.";  
  
    } else {  
  
        echo "ERROR: Faylni o'chirib bo'lmadi.";  
  
    }  
  
} else {  
  
    echo "ERROR: Fayl mavjud emas.";  
  
}
```

?>

Fayl tizimiga oid bo'lgan ba'zi funksiyalar

Funksiya	Vazifasi
fgetc()	Bir vaqtning o'zida bitta belgini o'qiydi
fgets()	Bir vaqtning o'zida bir qator ma'lumotni o'qiydi.
fgetcsv()	Vergul bilan ajratilgan qiymatlarni o'qiydi
filetype()	Fayl turini qaytaradi
feof()	Fayl oxiriga yetganligini tekshiradi
is_file()	Fayl doimiy fayl ekanligini tekshiradi
is_dir()	Fayl direktoriya ekanligini tekshiradi
is_executable()	Fayl bajariladiganligini tekshiradi
realpath()	Kanoniklashtirilgan absolyut fayl manzili
rmdir()	Bo'sh direktoriyani o'chiradi

Kataloglarni saralash

O'tgan darsimizda siz PHP da fayllar bilan qanday ishlashlikni o'rgangan edingiz. Shu kabi fayl tizimida direktoriyalar ya'ni kataloglar bilan ham ishлаshingiz mumkin bo'ladi. Misol uchun, siz katalogni ochasiz va uni ichidagi kontentlarni o'qiysiz, u katalogni o'chirasiz yoki qayta yaratasziz, catalog ichidagi fayllar ro'yhatini olasiz va hakozo vazifalarni bajarasiz. Bu kataloglarni saralashga kiradi.

Yangi katalog yaratish

Yangi katalog yaratish uchun PHP'ning **mkdir()** funksiyasi ishlataladi. Funksiya o'ziga parameter sifatida, fayl manzili va yaratilishi kerak bo'lган katalog nomini qabul qiladi. Namuna:

```
<?php

// Katalog manzili

$dir = "testdir";

// Katalogni mavjudligini tekshiramiz

if(!file_exists($dir)) {

// Katalogni yaratib ko'ramiz

if	mkdir($dir) {

    echo "Yaratildi.";

} else{

    echo "ERROR: Yaratib bo'lmadi.';

}

} else{

echo "ERROR: Katalog allaqachon mavjud.';

}

?>
```

mkdir() funksiyasi ishlashi uchun, katalog manzili parametridagi ota katalog mavjud bo'lishi kerak, ya'ni biz **mkdir('testdir/subdir')** deb katalog belgilaganmiz, **testdir** katalogi mavjud bo'lmasa subdir katalogi ham yaratilmaydi.

Faylni bir joydan boshqa joyga ko'chirish

PHP dasturlash tilida faylni bir joydan boshqa joyga ko'chirish imkoniyati ham mavjud bo'lib, bu imkoniyatdan biz **copy()** funksiyasini ishlatalish orqali foydalanishimiz mumkin bo'ladi. Bu funksiya birinchi parametr sifatida fayl manbasi va

ikkinchi parametr sifatida esa ko'chirilib qo'yiladigna joyi uzatiladi.

```
<?php

// Fayl manbasi manzili

$file = "example.txt";

// Mo'ljal

$newfile = "backup/example.txt";

// Mavjudligi tekshirildi

if(file_exists($file)) {

// Nusxalandi

if(copy($file, $newfile)) {

    echo "Ajoyib, nusxalandi!";

} else{

    echo "ERROR: Nusxalab bo'lmadi.';

}

} else{

    echo "ERROR: Fayl mavjud emas.';

}

?>
```

Bu namuna ishlashi uchun, nishondagi katalog ya'ni **backup** katalogi va manba fayli ya'ni **example.txt** mavjud bo'lishi kerak, aks holda PHP xatolik qaytaradi.

Katalog ichidagi fayllarni ro'yhatlash

Belgilangan manzil ichidagi katalog va fayllarni ro'yhatini chiqarish uchun biz **scandir()** PHP funksiyasidan foydalananamiz. Endi biz o'zimizning maxsus funksiyamizni yaratamiz. Bu funksiyani vazifasi katalog ichidagi barcha

fayllarni rekursiv ravishda chiqarishdan iborat. Skript katalogma-katalog joylashib ketgan fayllar bilan ishlash uchun foydali.

```
<?php

function outputFiles($path) {

    if(file_exists($path) && is_dir($path)) {

        $result = scandir($path);

        // Hozirgi va ota katalogni filterlayapmiz

        $files = array_diff($result, array('.', '..'));


        if(count($files) > 0) {

            // Qaytgan massivni sikllayapmiz

            foreach($files as $file) {

                if(is_file("$path/$file")){

                    // Fayl nomini ko'rsatayapmiz

                    echo $file . "<br>";

                } else if(is_dir("$path/$file")){

                    // agar katalog mavjud bo'lsa rekursiv

                    chaqiryapmiz

                    outputFiles("$path/$file");

                }

            }

        } else{

            echo "ERROR: katalogda fayl topilmadi.';

        }

    } else {

        echo "ERROR: Katalog mavjud emas.';

    }

}
```

```
}
```

```
outputFiles("temp");
```

```
?>
```

Ma'lum turdag'i fayllarni ro'yhatlash

Katalog va fayl tuzilmalarida ishslash mobaynida, ba'zida sizga katalog ichidagi ma'lum turdag'i fayllarni topish kerak bo'ladi. Misol uchun sizga faqat .text yoki .png fayllar va hakozolar kerak bo'lishi mumkin. Buni siz PHP bilan osongina **glob()** funksiyasi orqali qilishingiz mumkin. Bu funksiya andozaga asoslanib fayllarni taqqoslaydi.

```
<?php
```

```
/* Katalogni qidiramiz va taqqoslangan
```

```
fayllar massivini siklga qo'yamiz */
```

```
foreach(glob("documents/*.txt") as $file){
```

```
    echo basename($file) . " (O'lchami: " . filesize($file) . "
```

```
bayt) " . "<br>";
```

```
}
```

```
?>
```

glob() funksiyasi orqali katalog ichidagi yoki uning quyi kataloglari ichidagi barcha fayllarni topish ham mumkin. Quyidagi keltirilgan namunada funksiya rekursiv tarzda barcha fayllarni tekshiradi:

```
<?php
```

```
function outputFiles($path) {
```

```
    if(file_exists($path) && is_dir($path)) {
```

```
        $files = glob($path ."/*");
```

```
        if(count($files) > 0) {
```

```
            foreach($files as $file) {
```

```
if(is_file("$file")){
    echo basename($file) . "<br>";
} else if(is_dir("$file")){
    outputFiles("$file");
}
} else{
    echo "ERROR: Katalogda bunday fayl yo'q.";
}
} else {
    echo "ERROR: Katalog mavjud emas.";
}
}

outputFiles("mydir");
?>
```

Fayl yuklash

Bu mavzuda biz oddiy HTML forma va PHP dan foydalangan holda masofadagi serveringizga qanday qilib fayllarimizni yuklashni o'rGANAMIZ. Har qanday turdagি faylni yuklashingiz mumkin: rasmlar, videolar, ZIP fayllar, MS Hujjatlar, PDFlar va shuningdek boshqa fayllarni ham.

Birinchi qadam: Fayl yuklash uchu HTML forma

Quyidagi namunada faylni yuklash uchun ishlataladigan HTML forma ko'rsatilgan:

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8">

<title>Fayl yuklash formasi</title>

</head>

<body>

    <form action="upload-manager.php" method="post"
enctype="multipart/form-data">

        <h2>Faylni yuklash</h2>

        <label for="fileSelect">Fayl nomi:</label>

        <input type="file" name="photo" id="fileSelect">

        <input type="submit" name="submit" value="Upload">

        <p><strong>Eslatma:</strong> Faqatgina .jpg, .jpeg,
.gif, .png formatlar qo'llanadi, Hajmi 5 mb dan oshmasin.</p>

    </form>

</body>

</html>
```

Ikkinci qadam: Yuklangan faylni qayta ishslash

Bu yerda endi bizda **upload-manager.php** fayli mavjud bo'lib bu fayl ba'zi asosiy xavfsizlik choralarini masalan, foydalanuvchi yuklayotgan faylini uning hajmi va turi biz belgilagan qoidaga mos kelayotganligi tekshirilgandan so'ng uni **upload** papkasiga saqlaydi.

Eslatma: file-select maydoniga qo'shimcha sifatida POST metodi va enctype="multipart/form-data" atributi ishlatilishi kerak. Bu atribut forma ma'lumotini ko'p qisqli MIME ma'lumoti ya'ni binary ma'lumotlar: rasm, video, audio va hakozolar sifatida shifrlashni ta'minlaydi

```
<?php
// Fayl yuborilganmi?
```

```
if($_SERVER["REQUEST_METHOD"] == "POST") {  
  
    // Fayl xatolarsiz yuklandimi?  
  
    if(isset($_FILES["photo"])) && $_FILES["photo"]["error"] ==  
0) {  
  
        $allowed = ["jpg" => "image/jpg", "jpeg" =>  
"image/jpeg", "gif" => "image/gif", "png" => "image/png"];  
  
        $filename = $_FILES["photo"]["name"];  
  
        $filetype = $_FILES["photo"]["type"];  
  
        $filesize = $_FILES["photo"]["size"];  
  
  
        // Fayl kengaytmasi tasdiqlandi  
  
        $ext = pathinfo($filename, PATHINFO_EXTENSION);  
  
        if(!array_key_exists($ext, $allowed)) die("Error:  
Iltimos mavjud fayl formatida yuklang.");  
  
  
        // Fayl hajmi tasdiqlandi? (5mb max)  
  
        $maxsize = 5 * 1024 * 1024;  
  
        if($filesize > $maxsize) die("Error: Ruxsat etilgan  
hajmdan oshib ketti");  
  
  
        // Faylning MIME turi tekshirildimi?  
  
        if(in_array($filetype, $allowed)) {  
  
            // Uni yuklashdan oldin mavjudligi tekshirildimi?  
  
            if(file_exists("upload/" . $filename)) {  
  
                echo $filename . " fayl mavjud.";  
  
            } else{  
  
move_uploaded_file($_FILES["photo"]["tmp_name"], "upload/" .  
$filename);  
  
                echo "Fayl yuklandi";  
            }  
        }  
    }  
}
```

```
        }

    } else{

        echo "Error: Faylni yuklashda xatolik. Qayta urunib
ko'ring";

    }

} else{

    echo "Error: " . $_FILES["photo"]["error"];

}

?>
```

Eslatma: Yuqoridagi script bir xil nomli fayl yuklanishini oldini oladi. Biroq agar xohlasangiz siz fayl nomi oldiga taymstemp formatini qo'shib qo'yishingiz ham mumkin:

```
$filename = time() . '_' . $_FILES["photo"]["name"];
```

Yuqoridagi kodning ta'rifi

Fayl yuklangandan so'ng u faylga biz PHP ning superglobal massiv turiga mansub bo'lgan **\$_FILES** massivi orqali kirishimiz mumkin. Misol uchun foto nomli fayl tanlash maydon bor, formamizda. Agar foydalanuvchi shu forma yordamida fayl yuklasa biz u faylning turini, hajmini, vaqtinchalik nomi yoki uni yuklash jarayonida sodir bo'lgan xatoliklarini, nomini **\$_FILES['foto']** bog'langan massivi orqali olishimiz mumkin:

- **\$_FILES["foto"]["name"]** — Bu massiv qiymati faylning original nomini belgilaydi, faylning kengaytmasini ham o'z ichiga oladi. Fayl manzili ko'rsatilmaydi.
- **\$_FILES["foto"]["type"]** — Bu massiv qiymati faylning MIME turini aniqlaydi.
- **\$_FILES["foto"]["size"]** — Bu massiv turi faylning hajmini baytlarda ifodalaydi.

- `$_FILES["foto"]["tmp_name"]` — Bu massiv qiymati fayl serverga yuklangan zahoti unga biriktirilgan fayl manzili va vaqtinchalik nomini belgilaydi.
 - `$_FILES["foto"]["error"]` — Bu massiv qiymati fayl yuklash bilan bog'liq bo'lган xatolik yoki status kodini ifodalaydi. Agar 0 bo'lsa, hech qanday xatolik bo'lмаган bo'ladi.
-

Cookie'lar bilan ishlash

Ta'rif

Cookie(kuki) - bu Internet server tomonidan brauzerga kirilganda har safar brauzer tomonidan qaytariladigan malumot paketi bo'lib, u keyinchalik foydalanuvchini serverga kirishini kuzatish yoki aniqlash uchun ishlataladi.

Kuki(cookie) nima?

Kuki bu **4 KB** ga yaqin bo'lган kichik miqdordagi ma'lumot bo'lib, foydalanuvchining kompyuterida saqlanadi. Ular odatda foydalanuvchining masalan, logini, ismi, nima yoqqani kabi ma'lumotlarni saqlab, uning veb-saytga keyingi tashrifida sayt ma'lumotga qarab personalizatsiya qilinadi.

Kukini sozlash

PHP'da kukini o'rnatish uchun **setcookie()** funksiyasi ishlataladi. Bu **setcookie()** funksiyani skriptingizni eng yuqori qismida yozganingizni tekshiring, aks holda kuki ishlamaydi. Bu funksianing asosiy sintaksisi quyidagicha:

```
setcookie(nom, qiymat, muddat, manzil, domen, xavfsizlik);
```

`setcookie()` funksiyasining parametrлари quyidagi qiymatlarni oladi:

Parametr	Ta'rif
nom	Kukining nomi
qiymat	Kukining qiymati. Shaxsiy ma'lumotlarni saqlamang. Chunki bu foydalanuvchining kompyuterida saqlanadigan ma'lumot hisoblanadi
muddati	Tugash muddati UNIX taymstemp formatida bo'ladi. Bu vaqt tugagandan keyin kukiga kirib bo'lmaydi. Standart qiymati 0.
manzili	Kuki mavjud bo'ladigan serverdagи manzil belgilanadi. Agar / o'rnatilgan bo'lsa, kuki domenning hamma joyida mavjud bo'ladi.
domen	Qaysi domen uchun kuki mavjud, masalan: www.tutorials.uz
xavfsizlik	Agar bu maydon mavjud bo'lsa, kuki faqatgina xavfsiz HTTPS aloqa mavjud bo'lganda jo'natilishi kerakligini bildiradi.

tutorialsuz nomli kuki yaratamiz va unga saytni qiymat sifatida beramiz. Shu bilan birgalikda kuki amal qilish muddatini ham kiritamiz, masalan 30 kun(30kun * 24 soat * 60 daq * 60 sek).

```
<?php
setcookie("tutorialsuz", "https://tutorials.uz",
time() + 30 * 24 * 60 * 60);
?>
```

Kuki qiymatlariga kirish

Yodda tuting: Kukining nomidan tashqari barcha argumentlar ixtiyoriy sanaladi. Xohlasangiz argumentni kiritishni o'rniga bo'sh

satr("") kiritib ketsangiz bo'ladi, amal qilish muddatini kiritishni xohlamasangiz 0 kiritib qo'yasiz.

PHP da **\$_COOKIE** superglobal o'zgaruvchisi mavjud bo'lib, bu o'zgaruvchi orqali kukining qiymatini olishimiz mumkin bo'ladi. U odatda brauzer tomonidan yuborilgan kukilarning qiymatini ro'yhatini o'z ichiga olgan bog'langan massiv hisoblanadi.

Ogohlantirish: Kukida hech qanday shaxsiy ma'lumotlarni saqlamang. Shaxsiy ma'lumotlar deganda, parol, token, telefon nomer va hakozolar nazarda tutulyapti. Bunday ma'lumotlarni sessiyalarda saqlaganingiz ma'qul. Sessiyalar keying mavzuda o'tiladi.

```
<?php  
echo $_COOKIE["tutorialsuz"];  
?>
```

Yodda tuting: Agar kukining amal qilish muddati 0 ga o'rnatilgan bo'lsa yoki qoldirilib ketilgan bo'lsa, kuki sessiya tugaganda tugaydi, masalan brauzer yopilganda.

Kukining qiymatini olishdan avval u kuki o'rnatilganmi yoki yo'q shuni tekshirib olishimiz kerak emas-u, lekin bu yaxshi amaliyotdan darak beradi. Yaxshisi hamma yurgan yo'ldan yuravering :

```
<?php  
if(isset($_COOKIE["username"])) {  
    echo "Ooo, bu ajoyib - " . $_COOKIE["username"];  
} else{  
    echo "Xush kelibsiz saytjon!";  
}  
?>
```

Kukini strukturasini ko'rish uchun siz **print_r()** funksiyasini quyidagicha foydalanishingiz kerak: **print_r(\$_COOKIE);**

Kukini o'chirish

Kukini o'chirish uchun **setcookie()** funksiyasi kuki nomi va bo'sh satr bilan chaqiriladi, ammo o'chirish uchun kerak bo'ladigan argument bu tugagan muddatni o'rnatishdan iborat. Quyidagicha:

```
<?php  
setcookie("tutorialsuz", "", time() -3600);  
?>
```

Sessiyalar

Ma'lumotni kukilarda saqlay olishingiz mumkin, lekin bu yerda jiddiy masala: xavfsizlik masalasi mavjud. Kuki foydalanuvchining kompyuterida saqlanganligin tufayli buzg'unchilar dasturingizni ishdan chiqaradigan potensial zararli kontentni kukini o'zgartirgan holda joylashtirishi mumkin.

Bundan tashqari har safar brauzer serverdan URL manzilini so'raganda veb-saytga tegishli bo'lgan barcha kuki ma'lumotlar avtomatik tarzda so'rovning ichida birga serverga yuboriladi. Bu degani siz foydalanuvchining tizimida har biri 4KB hajmdagi 5ta kukini saqlamoqchi bo'lsangiz, har safar foydalanuvchi sahifani ko'rganda 20KB ma'lumotni brauzer yuklashi kerak, bu saytingiz ishlashiga salbiy ta'sir qiloladi.

Siz ikkala muammoni PHP'ning sessiya funksiyasi bilan yechishingiz mumkin. Sessiya ma'lumotni foydalanuvchining kompyuterida emas balki serverda saqlaydi. Sessiyaga

asoslangan muhitda, har bir foydalanuvchi sessiya identifikator yoki SID deb nomlanadigan unikal maxsus raqam orqali aniqlanadi. Bu unikal sessiya ID har bir foydalanuvchini unga tegishli ma'lumot(email, postlari, raqami)larni bog'lash uchun ishlatiladi.

Yodda tuting: Sessiya identifikatorlari PHP tizimi tomonidan deyarli taxmin qilib bo'lmaydigan darajadagi tasodifiy ravishda yaratiladi.

Sessiya bilan ishlaymiz

Ma'lumotni sessiya o'zgaruvchisida saqlashdan avval siz sessiyani boshlab olishinigz kerak. Yangi sessiyani boshlash uchun, oddiygina **session_start()** funksiyasini chaqirish kifoya. Bu funksiya yangi sessiya yaratadi va foydalanuvchi uchun unikal sessiya ID generatsiya qiladi.

```
<?php  
session_start();  
?>
```

session_start() funksiyasi birinchi sessiya ushbu identifikator bilan avval yaratilganmi yoki yo'q, shuni tekshiradi. Agar sessiya shu id bilan topilsa, sessiya o'zgaruvchilarini o'rnatadi, agar topilmasa yangi sessiya ID bilan sessiya yaratadi.

Sessiya ma'lumotini olish va ma'lumotni saqlash

Siz **\$_SESSION[]** superglobal massiviga sessiya ma'lumotlarni kalit-qiymat juftligi kabi saqlashingiz mumkin. Saqlangan ma'lumotga sessiya mavjud bo'lgan vaqtgacha kirishga imkon bo'ladi:

```
<?php  
session_start();
```

```
//Ma'lumotni saqlayapmiz  
  
$_SESSION["firstname"] = "Sanjarbek";  
$_SESSION["lastname"] = "Sobirjonov";  
?>
```

Ushbu sessiya ma'lumotimizga aynan sessiya yaratgan domenimizning boshqa sahifasidan kirish uchun, oddiygina **session_start()** funksiyasini qayta chaqiramiz va undan keyin tegishli **\$_SESSION** bog'langan massiv kalitlarini uzatamiz:

```
<?php  
  
session_start();  
  
echo 'Salom, ' . $_SESSION["firstname"] . ' ' .  
$_SESSION["lastname"];  
  
?>
```

Eslatma: Bir xil domen va bir xil sahifadagi sessiya ma'lumotiga kirish uchun hech qanday qayta sessiya yaratish talab etilmaydi. Shunchaki sessiya o'zgaruvchilarini ishlatalish kifoya.

Sessiyani o'chirish

Agar biror sessiya ma'lumotini o'chirishni xohlasangiz, tegishli sessiya bog'langan massivini element kalitini **unset()** funksiyasi bilan ishlatalish kifoya:

```
<?php  
  
session_start();  
  
if(isset($_SESSION["lastname"])) {  
    unset($_SESSION["lastname"]);  
}  
?>
```

Ammo, sessiyani butunlay o'chirish uchun **session_destroy()** funksiyasi ishlataladi. Bu funksiyaga hech qanday argument kerak emas va yakka o'zi chaqirilganda barcha sessiya ma'lumotlarini o'chirishga qodir.

```
<?php  
// Sessiyani yaratish  
session_start();  
  
//Sessiyani o'chirish  
session_destroy();  
?>
```

Eslatma: `session_destroy()` funksiyasini ishlatishdan avval sessiyani qayta yaratishingiz kerak, qachonki u yerda avval sessiya ishlatilmagan bo'lsa.

Har bir sessiyaning tugash qiymati mavjud – davomiyligi, bu sekundlarda o'lchanadi. Bu bizga foydalanuvchi harakati mavjud bo'limganda sessiya qancha muddatgacha mavjud bo'lib turish kerakligini aniqlashga yordam beradi. Sessiya davomiyligini PHP konfigratsiya fayli(`php.ini`) ichidagi **session.gc_maxlifetime** o'zgaruvchisining qiymatini o'zgartirish orqali tartibga solishingiz mumkin.

Elektron xat yuborish

Elektron xabarlar yuborish veb ilovalar uchun notanish tushunchasi emas. Elektron xabar qachon yuboriladi? Masalan, saytingizda yangi foydalanuvchi o'z hisob raqamini yaratса “Xush kelibsiz, falonchi falonchi” deb xabar, ro'yhatdan o'tgan foydalanuvchining pochtasiga reklama, yangiliklar yoki

undan vebsayt kontakt formasi orqali sayt haqidagi fikrini so'rash kabi pochtalar yuborishingiz mumkin.

Siz o'zingizning PHP ilovangizdan bir yoki undan ortiq bo'lган qabul qilib oluvchilarga xoh oddiy matn xoh formatlangan HTML ko'rinishda **mail()** PHP ichki funksiyasi yordamida elektron xabar yaratib, jo'natishingiz mumkin.

mail(kimga, mavzu, xabar matni, sarlavhalar, parametrlar)

Bu **mail()** funksiyasining ba'zi argumentlari ixtiyoriy, ba'zilari esa majburiy ya'ni yozilishi shart bo'lganlarga bo'linadi. Keling ularni yaxshilab o'rganamiz:

Parametr	Ta'rif
Majburiy — Quyidagi parametrlarni yozish shart	
kimga	Qabul qilib oluvchining elektron pochtasi
mavzu	Jo'natiladigan pochtaning mavzusi. Bu parameter ya'ni mavzu qatori hech qanday yangi qator belgilari(\n)dan iborat bo'lmasisligi kerak
xabar matni	Jo'natiladigan xabar matni. Qatorlar 70 ta belgidan oshmasligi kerak.
Ixtiyoriy — Quyidagi parametrlarni yozish ixtiyoriy	
sarlavhalar	"From", "Cc", "Bcc" kabi qo'shimcha headerlar qo'shish uchun ishlataladi.
parametrlar	Qo'shimcha parametrlar uzatish uchun ishlataladi

Oddiy matnli pochta yuborish

PHPda elektron pochta yuborishning eng sodda usuli bu matnli pochta yuborish hisoblanadi. Quyidagi namunada biz— qabul qilib oluvchining elektorn pochta manzilini, mavzu qismi va xabar matni uchun o'zgaruvchilar e'lon qildik keyin o'zgaruvchilarni **mail()** funksiyasiga uzatdik.

```
<?php

$to = 'tutorialsuz@gmail.com';

$subject = 'A`zolik';

$message = 'Assalomu alaykum, siz "BEPUL" tarifga ulandingiz';

$from = 'sobirjonovs@gmail.com';

if(mail($to, $subject, $message)){
    echo 'Xabaringiz muvvaffaqiyatli yuborildi.';
} else{
    echo 'Xabarni jo`natib bo`lmadi.';
}

?>
```

Formatlangan pochta yuborish

PHPdan foydalanib matnli xabar yuborganingizda, barcha content oddiy tekst sifatida hisoblanadi. Endi biz oddiylikdan murakkablikga o'tamiz, pochtamizni xabarini HTML orqali formatlaymiz.

```
<?php

$to = 'tutorialsuz@gmail.com';

$subject = 'A`zolik';

$from = 'sobirjonovs@gmail.com';

// HTML pochta yuborish uchun Content type headeri sozlanishi kerak

$headers = 'MIME-Version: 1.0' . "\r\n";
$headers .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
// Email headerlarini yaratish
$headers .= 'From: '.$from."\r\n".
'Reply-To: '.$from."\r\n" .
```

```
'X-Mailer: PHP/' . phpversion();

// Oddiy HTML xabar yozamiz

$message = '<html><body>';

$message .= '<h1 style="color:#f40;">Assalomu alaykum!</h1>';

$message .= '<p style="color:#080;font-size:18px;">Siz "BEPUL"
tarifga ularningiz</p>';

$message .= '</body></html>';

if(mail($to, $subject, $message)) {

    echo 'Xabaringiz muvvaffaqiyatli yuborildi.';

} else{

    echo 'Xabarni jo`natib bo`lmadi.';

}

?>
```

Eslatma: mail() funksiyasi PHP tiziming bir ichki funksiyasi hisoblanadi lekin xabaringiz aniq yuborilishi uchun siz mashinangiz/kompyuteringizda pochta serverini sozlashingiz kerak bo'ladi.

Filterlar

Foydalanuvchi kiritgan ma'lumotni tekshirish va tozalash bu veb ilovalarda keng tarqalgan vazifalardan biri hisoblanadi. Bu vazifani bajarish uchun PHP da o'zining maxsus filter kengaytmasi mavjud bo'lib, bu sizga elektron pochta manzillari, URL manzillar, IP manzillar va hakozolar kabi ma'lumotlarni tekshirish va tozalashga yordam beradi.

Filter kengaytmasidan foydalanib ma'lumotni yaroqlilagini tekshirish uchun siz filter_var() funksiyasidan foydalanishingiz kerak. Uning asosiy sintaksi quyidagidan iborat:

filter_var(o'zgaruvchi, filter, qo'shimcha parametrlar)

Bu funksiya 3 ta parameter qabul qiladi, ulardan ikkitasi ixtiyoriy. Birinchi parameter filterlanishi kerak bo'lgan qiymati, ikkinchisi qo'llaniladigan filter ID'si va uchinchi parameter filterga tegishli bo'lgan qo'shimcha parametrlar massivi. Keling buni qanday ishlashini ko'ramiz.

Satrni tozalash

Quyidagi namunada satrni HTML teglardan tozalash ko'rsatilgan:

```
<?php  
  
$website = "<h1>Assalomu alaykum, tutorials.uz</h1>";  
  
$sanitizedWebsite = filter_var($website,  
FILTER_SANITIZE_STRING);  
  
echo $sanitizedWebsite;  
  
?>
```

Natija:

Assalomu alaykum, tutorials.uz

Butun sonni tekshirish

```
<?php  
  
$int = 20;  
  
if(filter_var($int, FILTER_VALIDATE_INT)) {  
  
    echo "<b>$int</b> butun son";  
  
} else{  
  
    echo "<b>$int</b> butun son emas";  
  
}  
  
?>
```

Yuqoridagi namunada, agar \$int o'zgaruvchisi 0ga o'rnatilsa, namunaviy kod butun son emas xabarini chiqaradi. Bu muammoni bartaraf etish uchun, siz 0 qiymati uchun tekshiruv kiritishigiz kerak:

```
<?php  
  
$int = 20;  
  
if(filter_var($int, FILTER_VALIDATE_INT) === 0 ||  
filter_var($int, FILTER_VALIDATE_INT)) {  
  
    echo "<b>$int</b> butun son";  
  
} else{  
  
    echo "<b>$int</b> butun son emas";  
  
}  
  
?>
```

IP manzilni tekshirish

```
<?php  
  
$ip = "127.0.0.1";  
  
if(filter_var($ip, FILTER_VALIDATE_IP)) {  
  
    echo "<b>$ip</b> yaroqli IP manzil";  
  
} else{  
  
    echo "<b>$ip</b> bu IP manzil yaroqsiz";  
  
}  
  
?>
```

Elektron pochtani tekshirish va keraksiz belgilardan tozalash

```
<?php  
  
$email = "someone@example.com";  
  
$sanitizedEmail = filter_var($email, FILTER_SANITIZE_EMAIL);  
  
if($email == $sanitizedEmail && filter_var($email,  
FILTER_VALIDATE_EMAIL)) {
```

```
echo "<b>$ip</b> yaroqli Email manzil";
} else{
    echo "<b>$ip</b> bu Email manzil yaroqsiz";
}
?>
```

Eslatma: FILTER_SANITIZE_EMAIL filteri harflar va raqamlardan tashqari barcha yaroqsiz belgilar !#\$%&'*+-=?^`{|}~@.[] ni email manzil satridan tozalaydi.

URL manzillarni tekshirish va keraksiz belgilardan tozalash

```
<?php
$url = "http://tutorials.uz";
$sanitizedUrl = filter_var($url, FILTER_SANITIZE_URL);
if($url == $sanitizedUrl && filter_var($url,
FILTER_VALIDATE_URL)) {
    echo "<b>$url</b> yaroqli URL manzil";
} else{
    echo "<b>$url</b> bu URL manzil yaroqsiz";
}
?>
```

Eslatma: FILTER_SANITIZE_URL filteri harflar va raqamlardan tashqari barcha yaroqsiz belgilar \$-_.+!*'(),{}|\|^~[]`<>#%";/?:@&= ni email manzil satridan tozalaydi

Shuningdek URL manzilda so'rov satri bor yoki yo'qligini ham tekshirishingiz mumkin. Buning uchun FILTER_FLAG_QUERY_REQUIRED flegi ishlataladi:

```
<?php
$url = "http://tutorials.uz?bolim=texnologiya";
$sanitizedUrl = filter_var($url, FILTER_SANITIZE_URL);
if($url == $sanitizedUrl && filter_var($url,
FILTER_VALIDATE_URL)) {
    echo "<b>$url</b> yaroqli URL manzil";
} else{
```

```
echo "<b>$url</b> bu URL manzil yaroqsiz";  
}  
?>
```

Oraliqdagi butun sonlarni tekshirish

Quyidagi namunada kiritilgan qiymat butun son yoki yo'qligi hamda u 0 dan 100 gacha bo'lgan sonlar orasidagi son ekanligi yoki yo'qlini tekshiradigan kod yozilgan:

```
<?php  
  
$int = 75;  
  
if(filter_var($int, FILTER_VALIDATE_INT, ["options" =>  
["min_range" => 0, "max_range" => 100]])){  
  
    echo "<b>$int</b> 0 dan 100 gacha bo'lgan sonlar orasida  
bor";  
  
} else{  
  
    echo "<b>$int</b> 0 dan 100 gacha bo'lgan sonlar orasida  
yo'q";  
  
}  
?>
```

Formalar bilan ishslash

Forma – bu HTML sahifasining foydalanuvchi o'ziga tegishli bo'lgan ma'lumotlari, masalan, ismi, familiyasi, yoshi, login va parol va boshqalarni yozishi mumkin bo'lgan elementi hisoblanadi.

Bu ma'lumotlar PHP orqali olinishi va kerakli usulda qayta ishlanishi mumkin. Misol uchun foydalanuvchi forma ichiga o'z familiyasi va ismini kiritdi, “Yuborish” tugmasiga bosadi va kiritilgan ma'lumotlar PHPdagi serverga yuboriladi. U

yerda biz kiritilgan ism va familiyasini olishimiz va misol uchun ularni bazaga saqlab qo'yishimiz mumkin.

HTMLda forma qanday yaratiladi

Forma **form** tegi yordamida yaratiladi. Bu tegning 2ta asosiy atributi mavjud.

action atributi kiritilgan ma'lumotlar yuboriladigan sahifa manzili ko'rsatiladi. Agar bu atribut bo'sh qoldirilsa – forma saytning joriy sahifasiga jo'natiladi.

method atributi formani qay usulda yuborish kerakligini ifodalaydi. GET qiymat yoki POST qiymat qabul qilishi mumkin. **GET** metodi bilan yuborilgan forma ma'lumoti brauzerning vebsayt manzili yoziladigan maydonchasida ko'rinish turadi, **POST** bilan yuborilganida esa yo'q.

Yodda tutingki, formalı PHP kod sahifasi ko'ringanidek bir marta bajarilmaydi, balki 2 marta – birinchi martasida foydalanuvchi sayt sahifasiga kiradi, formani to'ldirib yuborish tugmasini bosadi, formadagi ma'lumot serverga yuboriladi va PHP kod sahifasi yana qaytadan ishga tushishni boshlaydi.

Forma maydonlari

form tegi ichida turli xil forma elementlari joylashishi mukmin: **input** kiritish maydonchasi, formani yuborish tugmasi, ko'p satrli **textarea** kiritish maydonchasi va turli xildagi bizga unchalik kerak bo'limgan elementlar.

Forma namunasi

```
<form action="" method="GET">
    <input type="text" name="user"><br><br>
    <textarea name="message"></textarea><br><br>
    <input type="submit">
```

```
</form>
```

A screenshot of a web form. It contains two input fields: one for text input and one for file upload. Below the fields is a submit button labeled "Отправить запрос".

Formadi ma'lumot PHPda qanday olinadi

Formadagi foydalanuvchi kiritgan ma'lumotni PHP kodda olishimiz mumkin. Bu ish **\$_GET**, **\$_POST**, **\$_REQUEST** superglobal massivlari orqali amalga oshadi.

\$_GET dan GET metodi orqali yuborilgan ma'lumotlar olinadi, **\$_POST** da POST metodi orqali jo'natilgan ma'lumotlar joylashadi. **\$_REQUEST** da esa bir vaqtning o'zida turli metodlar orqali yuborilgan ma'lumotlarni olish mumkin.

Bu aynan qanday qilinadi: name="user" atributli formamiz bor deylik. Unda bu formani yuborganimizdan keyin biz bu ma'lumotni quyidagi usulda olishimiz mumkin:

\$_GET['user'] (yoki **\$_POST['user']**, yoki **\$_REQUEST['user']**).

Yuborilgandan keyin forma maydonchalarining qiymatlarini saqlash

Keling, forma yuborilganda inputlardagi qiymatlar o'chib ketmaydigan qilamiz:

```
<form action="" method="GET">
  <input type="text" name="user" value=<?php echo
$_REQUEST['user']; ?>">
  <input type="submit">
</form>
```

Yuqoridagi namunada PHP ogohlantirish generatsiya qiladi. Sababi hali forma yuborilmasidan biz yo'q ma'lumotni chiqaryapmiz. Buni quyidagicha to'g'irlashimiz mumkin:

```
<form action="" method="GET">
    <input name="user" value=<?php if(isset($_REQUEST['user'])) echo $_REQUEST['user']; ?>>
    <input type="submit">
</form>
```

Muntazam ifodalar

Muntazam ifodalar – bu murakkab qidiruv va almashtiruv (yoki shunchaki qidiruv) vazifasini amalga oshiradigan buyruqlar. PHP'da muntazam ifodalar bilan ishlash uchun bir necha xil funksiyalar mavjud. Birinchi *preg_replace* misolida ular bilan tanishishni boshlaymiz: bu funksiya *str_replace* funksiyasiga juda o'xshash bo'lib, bunda ham qidiruv va almashtirish mavjud, faqat birinchi parametri oddiy satr emas balki *muntazam ifoda* qabul qiladi. Farqni ko'ring:

```
<?php
str_replace(nimani, nimaga, qayerni almashtiramiz)
preg_replace(nimani, nimaga, qayerni almashtiramiz)

echo str_replace('a', '!', 'aabbaa'); //natija !!!bb!!!
echo preg_replace('#a#', '!', 'aabbaa'); //natija !!!bb!!!
?>
```

Panjara, zamonaviy til bilan aytganda heshtegga e'tiboringizni qarating. Bu heshteglar *cheklovchi* muntazam ifoda deb ataladi.

Cheklovchilar ulardan keyin **modifikator** – muntazam ifodaning umumiy xususiyatini o'zgartiradigan buyruqni yozish uchun kerak. Misol uchun, **i** modifikatori belgilarning katta-kichikligini tekshirishni rad etishni talab qiladi.

```
<?php  
  
echo preg_replace('#A#!', '!', 'aAb'); //natija 'a!b'  
echo preg_replace('#A#i', '!', 'aAb'); //natija '!!b'  
  
//Ikkinchchi holatda registr e'tiborga olinmayapti  
  
?>
```

Harflar, sonlar va turli belgilar

Muntazam ifodalarda 2ta turdag'i belgilar mavjud: **o'zini** va **buyruqli(ya'ni buyruq kuchiga ega bo'lgan)** belgini ifodalaydigan.

Harflar va raqamlar bu o'zini ifodalaydi, nuqta esa maxsus belgi bo'lib u "har qanday turdag'i belgi"ni ifodalaydi. Namunaga qarang:

```
<?php  
  
echo preg_replace('#xax#!', '!', 'xax xaax'); //natija '! xaax'  
echo preg_replace('#123#!', '!', '123 xaax'); //natija '! xaax'  
echo preg_replace('#x3x#!', '!', 'x3x xaax'); //natija '! xaax'  
  
//E'tibor bering, registr bu yerda ahamiyatga ega:  
echo preg_replace('#A3B#!', '!', 'a3b A3B'); //natija 'a3b !'  
?>
```

Maxsus nuqta belgisi ishlatilgan quyidagi namunaga qarang(str_replace da bunday qilib bo'lmaydi):

```
<?php  
  
echo preg_replace('#x.x#', '!', 'xax xsx x&x x-x xaax');  
//Natija '! ! ! ! xaax'  
  
?>
```

Nuqta – bu har qanday turdagи belgi. Yuqoridagi namunaga ta'rif: x harfi bilan boshlangan va o'rtasida bitta har qanday turdagи belgidan iborat va x bilan tugagan shablonni almashtirish kerakligini bildiradi.

```
<?php  
  
echo preg_replace('#x..x#', '!', 'xax xabx'); //natija 'xax !'  
  
?>
```

Nuqta – bu har qanday turdagи belgi. Yuqoridagi namunaga ta'rif: x harfi bilan boshlangan va o'rtasida 2ta har qanday turdagи belgidan iborat va x bilan tugagan shablonni almashtirish kerakligini bildiradi.

Demak, eslab qoling: harf va raqam bu faqat o'ziga teng ya'ni o'zini almashtiradi, nuqta esa har qanday belgini.

Belgilarni takrorlash operatori(*,+,?)

Shunday holat bo'ladi, qanaqadir belgi ko'rsatilgan miqdorda takrorlanishi kerak bo'lishi mumkin. Agar biz takrorlanishning aniq midorini bilsak, unda o'sha belgini o'zini o'shancha miqdorda yozishimiz mumkin. Lekin buni “bir yoki undan ortiq marta takrorla” demoqchi bo'lsak nima qilamiz?

Buning uchun **takrorlash operatori(kvantifikator** deb ham ataladi): ‘+’ plyus (bir yoki undan ortiq marta), yulduzcha ‘*’ (nol yoki undan ortiq marta) va so'roq ‘?’ (nol yoki bir marta, qisqa qilib aytsam bo'lishi mumkin, bo'lishi mumkin emas) takrorlaydigan operator mavjud.

Ushbu operatorlar ulardan oldin turgan belgi uchun amal qiladi. Namunani ko'ring:

```
<?php  
  
echo preg_replace('#xa+x#', '!', 'xx xax xaax xaaax xbx');  
//natija 'xx ! ! ! xbx'  
  
?>
```

Bu holatda, qidiruv shabloni quyidagiga o'xshaydi: 'x' harfi, 'a' harfi bir yoki ko'p marta, 'x' harfi

```
<?php  
  
echo preg_replace('#xa*x#', '!', 'xx xax xaax xaaax xbx');  
//natija '! ! ! ! xbx'  
  
?>
```

Bu holatdagi qidiruv shabloni quyidagiga o'xshaydi: 'x' harfi, 'a' harfi nol yoki ko'p marta, 'x' harfi. Boshqa qilib aytganda: 'a' harfi yoki yo'q, yoki bir marta yoki ko'p marta takrorlanadi.

```
<?php  
  
echo preg_replace('#xa?x#', '!', 'xx xax xaax xbx'); //natija  
' ! ! xaax xbx'  
  
?>
```

Bu holatdagi qidiruv shabloni quyidagiga o'xshaydi: 'x' harfi, va undan keying 'a' harfi bo'lishi mumkin yoki yo'q, keyin 'x' harfi

Guruhash qavslari

Oldingi namunalaridagi takrorlash operatorlari faqatgina bitta belgi uchun amal qiladi. Agar 1ta takrorlash operatori bir nechta belgiga ishlatalishini xohlasak-chi? Nima qilamiz?

Buning uchun guruhash qavslari '(' va ')' mavjud:

```
<?php  
  
echo preg_replace('#x(ab)+x#', '!', 'xabx xababx xaabbx');  
//natija '! ! xaabbx'  
  
?>
```

Bu holatda qidiruv shabloni quyidagi o'xshaydi: 'x' harfi, undan keyin 'ab' harflari bir yoki ko'p marta, keyin 'x' harfi.

Ya'ni, agar guruhlash qavslari ichida belgilar turgan bo'lsa, takrorlash operatori o'sha guruhlash qavslari ichidagi belgilar uchun bemalol ishlaydi.

Maxsus belgilarni ekranlashtirish

Aytaylik, biz maxsus belgilarni oddiy belgi sifatida ishlatmoqchimiz. Misol uchun, quyidagi shablon bo'yicha qidiryapmiz: 'a' harfi, keyin plus '+' , keyin 'x' harfi. Quyidagi kod biz xohlagandek ishlamaydi:

```
<?php
    echo preg_replace('#a+x#', '!', 'a+x ax aax aaax'); //natija
    'a+x ! ! !'
?>
```

Muntazam ifodadan biz 'a', keyin plus '+', keyin 'x' ni qidirishini xohlayapmiz. Aslida esa u bunday qidirmayapti, aksincha: 'a' harfi bir marta yoki ko'p marta, keyin 'x' harfini qidiryapti.

Maxsus belgini o'zini ham oddiy belgi sifatida qoldirish uchun uni orqa slesh(\) belgisi yordamida ekranlashtirish kerak. Mana bunday:

```
<?php
    echo preg_replace('#a\+x#', '!', 'a+x ax aax aaax'); //natija
    'a+x ! ! !'
?>
```

Mana endi muntazam ifoda qidiruvni quyidagicha ko'radi: 'a' harfi, keyin plus '+', keyin esa 'x' harfi sifatida.

Cheklovchilar

Cheklovchi sifatida faqatgina # emas balki turli xil belgilar(faqatgina harf va raqam emas) ishlatilishi mumkin. Agar qavslar ishlatilayotgan bo'lsa, unda chap cheklovchisi – bu ochiluvchi qavs, o'ngi esa – yopiluvchi bo'ladi.

```
<?php
```

```
echo preg_replace('&a+&', '!', 'satr'); //cheklovchi  
ampersand  
  
echo preg_replace('(a+)', '!', 'satr'); //cheklovchi qavs  
?>
```

Oddiy va maxsus belgilar ro'yhati

Agar oddiy belgini ekranlashtirsa – hech qanday vahimali narsa bo'lmaydi- hammasi odatdagidek ishlaydi.

Istisno – raqamlar, ular kartmonda qoladi, **muntazam ifodalar mavzusining 2-qismiga qarang.**

Yana istisno – ‘X’ modifikatori: agar u belgilangan bo’lsa, unda oddiy belgilarni ekranlashtirishda xatolik kelib chiqadi, **muntazam ifodalar mavzusining 3-qismiga qarang.**

Maxsus belgilar hisoblanadi: \$ ^ . * + ? \ { } [] () |

Maxsus belgilar hisoblanmaydi: @ : , ‘ “ ; - _ = < > % # ~ ` & ! /

Bundan tashqari: maxsus belgi **tanlangan cheklovchi** bo'ladi.

Muntazam ifodalar – II

+,* va ? operatorlari unchalik ham universal emas(ko'p holatlarda ishlatilishiga qaramay). Agar biz aniq bo'lgan takrorlanishlar sonini belgilamoqchi bo'lsak nima qilishimiz kerak?

{} operatori takrorlanishlar sonini ko'rsatish uchun qo'llaniladi va bu bizga ancha qo'l keladi. U quyidagicha ishlaydi: {5} – 5 ta takrorlish, {2,5} – 2 dan 5 tagacha bo'lgan takrorlanishlar, {2,} – 2 marta yoki undan ko'p bo'lgan takrorlanishlar.

Iltimos, shuni yodda tutingki, bizda **{2,}** variant mavjud bo'lgani bilan, **{,2}** variant mavjud emas:

```
<?php
    echo preg_replace('#xa{1,2}x#', '!', 'xx xax xaax
xaax'); //natija 'xx ! ! xaaax'
?>
```

Bu holatda, qidiruv shabloni quyidagilarni ko'radi: 'x' harfi bitta o'zi, 'a' harfi bir yoki 2 marta qatnashadi, 'x' harfi.

```
<?php
    echo preg_replace('#xa{2,}x#', '!', 'xx xax xaax
xaax'); //natija 'xx xax ! !
?>
```

Bu holatda, qidiruv shabloni quyidagicha qidiradi: bitta 'x' harfi qatnashgan, 'a' harfi 2 yoki undan ko'p marta qatnashgan, va yana 'x' harfi qatnashgan

```
<?php
    echo preg_replace('#xa{2}x#', '!', 'xx xax xaax xaaax');
//natija 'xx xax ! xaaax'
?>
```

Bu holatda, qidiruv shabloni quyidagilarni ko'radi: 'x' harfi qatnashgan, 'a' harfi 2marta qatnashgan, 'x' harfi qatnashgan.

Bu masala, quyidagicha ham hal etilishi mumkin, albatta(hatto qisqaroq ham):

```
<?php
    echo preg_replace('#xaax#', '!', 'xx xax xaax xaaax');
//natija 'xx xax ! xaaax'
?>
```

Lekin, ikkinchi variantda har doim ham qisqa bo'lmasligi mumkin – ikkovini taqqoslab ko'ring:

```
<?php
    echo preg_replace('#aaaaaaaaaaax#', '!',
'saaaaaaaaatr');
    echo preg_replace('#xa{10}x#', '!', 'aaaaaaaaaaaatr');
?>
```

Endi esa, ikkinchi ancha osonroq ☺

Biroq, yuqorida aytganimdek bu ishlar maydi:

```
<?php
```

```
echo preg_replace('#xa{,2}x#', '!', 'xax xaax xaaax');
//ktuilgan natija '! ! xaaax'
?>
```

Biz bu shablondan: 'x' harfi qatnashgan, 'a' harfi 2 yoki undan kam marta qatnashgan, 'x' harfi oxirida ham qatnashganni qidirishini xohlagandik, afsuski, bu ishlamadi.

Aniq belgilashimiz kerak:

```
<?php
echo preg_replace('#xa{1,2}x#', '!', 'xx xax xaax
xaaax'); //natija '! ! xaaax'
?>
```

Endi kutganimizdek bo'ldi

Yodda tuting, nol(0) ham mavjud:

```
<?php
echo preg_replace('#xa{0,2}x#', '!', 'xx xax xaax
xaaax'); //natija '! ! ! xaaax'
?>
```

Bu holatda, qidiruv shabloni quyidagicha ko'radi: 'x' harfi qatnashgan, 'a' harfi 0(majud bo'limgan) yoki 2 marta qatnashgan, 'x' harfi oxirida qatnashgan.

Belgilar guruhi \s, \S, \w, \W, \d, \D

Bir vaqtning o'zida butun belgilar guruhini tanlash uchun maxsus buyruqlar mavjud. Misol uchun, \d ' 0 dan 9 gacha bo'lgan sonlarni' ni anglatsa, \D esa teskarisi, 'raqam bo'limgan' ni anglatadi. Namunaga qarang:

```
<?php
echo preg_replace('#\d+#', '!', '1 12 123 abc @@@');
//natija '! ! ! abc @@@'
?>
```

Bu namunada, qidiruv shabloni quyidagilarni ko'radi: 0 dan 9 gacha bo'lgan bir yoki ko'p marta ishlatalgan sonlar sifatida. Esda tuting, + operatori bu buyruqlarni bitta belgi sifatida ko'radi(guruhash qavslari kerak emas). Bu barcha takroriy buyruqlarga amal qiladi.

```
<?php
echo preg_replace('#\D+#', '!', '123abc3@@'); //natija
'123!3!'
?>
```

Endi hammasi o'zgardi: Odan 9 gacha bo'limgan 1 yoki ko'p marta ishlatalgan har qanday belgi qidiriladi.

Turli xil belgilar guruhining ta'riflari:

Belgi

\s

Bo'shliq yoki bo'shliq belgilari(tab klavishi, yangi qator va hakozolar)ni bildiradi

\S

Bo'shliq emas, ya'ni \s ning o'z ichiga olganlaridan tashqari bo'lgan har qanday belgi

\w

Son yoki harf(eslatma: bu krill harflarini o'z ichiga olmaydi! Buni setlocale funksiyasi yordamida to'g'irlashingiz mumkin. Xohlasangiz albatta.

\W

Raqam yoki harf bo'limgan

\d

0 dan 9 gacha bo'lgan sonlar

\D

0 dan 9 gacha bo'limgan sonlar

Uning ma'nosi

Ko'rganingizdek, katta harf "rad" ga ishora: agar \s bo'shliq bo'lsa, unda \S bo'shliq emas, va hakozo. Namunalarga e'tibor bering:

```
<?php
    echo preg_replace('#\s#', '!', '1 12 123 abc @@@');
//natija '1!12!123!abc!@@@'
?>
```

Bu namunada, qidiruv shabloni quyidagilarni qidiradi: bir marta ishlataligan bo'shliq.

```
<?php
    echo preg_replace('#\S+#', '!', '1 12 123 abc @@@');
//natija '! ! ! ! ! !'
?>
```

Bu yerda qidiruv shabloni bir yoki undan ko'p marta takrorlangan bo'shliq mavjud bo'limgan belgilarni qidiradi.

Kvadrat qavslar '[' va ']'

\s, \S, \w, \W, \d, \D belgilar guruhi juda moslashuvchan emas. Barcha harflarni topadigan oddiy topshiriqlarni ham bular bilan yechib bo'lmaydi. Keling, muntazam ifodalar yana nimaga qodirligini ko'ramiz. Muntazam ifodalarda [va] kvadrat qavslari mavjud bo'lib, u 'yoki' shartini o'rnida ishlataladi:

```
<?php
```

```
<?php
echo preg_replace('#[abc]xx#', '!', 'axx bxx cxx exx');
//natija '! ! ! exx'
?>
```

Qidiruv shabloni: birinchi belgisi 'a','b' yoki 'c' bo'lgan harfni, keying 2ta belgisi 'x' harfi bo'lganni qidiradi.

Lekin faqat bu bilan chegaralnib qolmaganmiz. ^ belgisini ishlatgan holda biz inkor ishorasini berishimiz mumkin:

```
<?php
echo preg_replace('#[^abc]xx#', '!', 'axx bxx cxx exx');
//natija 'axx bxx cxx !'
?>
```

Bu qidiruv shabloni: birinchi belgisi 'a', 'b' yoki 'c' **BO'LMAGAN** (ulardan iborat bo'lmanan har qanday belgi), undan keying 2 ta harfi 'x' bo'lganni qidiradi.

Yana nima qilish mumkin:

- Guruhlar belgisini belgilash mumkin: **[az]** kichik lotin harflarini, **[AZ]** – katta, **[0-9]** – 0 dan 9 gacha bo'lgan sonlarni ifodalaydi.
- Murakkabroq: **[a-zA-Z]** – katta va kichik lotin harflarni, va huddi o'zi + raqamlar – **[a-zA-Z0-9]**, va hakozo. Tartib ahamiyatga ega emas, **[a-zA-Z]** yoki **[A-Za-z]** ni farqi yo'q. Bir xil.
- Yanada murakkabroq: **[2-5]** – 2 dan 5 gacha bo'lgan sonlar, **[ac]** – 'a' dan 'c' gacha bo'lgan harflar(alifbo bo'yicha).

Xususiyatlari:

№1. Shlyapacha(^) – bu [] kvadrat qavsning ichida maxsus belgi hisoblanadi(tashqarisida ham). Agar siz uni oddiy belgi sifatida ishlatmoqchi bo'lsangiz, uni kvadrat qavsning ichining boshiga qo'ymang(kvadrat qavsning boshida bu har doim maxsus belgi deb qaraladi: **[^d]** – bu maxsus belgi, va bu **[d^]** esa unday emas.)

```
<?php
```

```
echo preg_replace('#[^d]xx#', '!', 'axx bxx ^xx dxx');
//natija '! ! ! dxx'
?>
```

Va

```
<?php
echo preg_replace('#[d^]xx#', '!', 'axx bxx ^xx dxx');
//natija 'axx bxx ! !'
?>
```

Birinchi namunada shlyapacha maxsus belgi – inkor belgisi rolini o'ynayotgan bo'lsa, ikkinchisida esa oddiy belgi rolini o'ynamoqda.

Birinchi namunadagi shlyapaning xususiyatini ekranlashtirishingiz ham mumkin:

```
<?php
echo preg_replace('#[\^d]xx#', '!', 'axx bxx ^xx dxx');
//natija 'axx bxx ! !'
?>
```

Kiril yozuvining xususiyatlari

Birinchidan: Kiril yozuvi \w da kiritilmagan. Bunday qilish kerak: **[а-яА-Я]**

Ikkinchidan: [а-яА-Я] bunday qilish yetarli emas – bu yerga ё harfi kirmaydi, uning uchun bunday qilish kerak: **[а-яА-ЯЁё]**

Uchinchidan: PHP kiril yozuvi bilan ishlashni yoqtirmaydi, shuning uchun hammasi to'g'ri ishlashi uchun – ‘u’ modifikatorini qo'yish kerak(aynan kichkina harfni)

```
<?php
echo preg_replace('#[а-яА-ЯЁё] яя#u', '!', 'аяя ёяя
2яя'); //natija '! ! 2яя'
?>
```

Kiril yozuvi u modifikatorisiz ham to'g'ri ishlay oladi, ba'zida esa yo'q. PHP shunaqa ☺

Satrning boshlanishi '^' va oxiri '\$'

Satrning boshi '^' va oxiri '\$' ni ko'rsatadigan maxsus belgilar mavjud:

```
<?php
    echo preg_replace('#^aaa#', '!', 'aaa aaa aaa');
//natija '! aaa aaa'
?>
```

Qidiruv shabloni faqatgina qatorning boshidagi 'aaa' ni '!' ga almashtiradi

Vertikal chiziq yoxud YOKI

Kvadrat qavslar 'yoki' shartini yaratishning yagona yechimi emas: vertikal chiziq '|' orqali ham 'yoki' shartini yaratishimiz mumkin:

```
<?php
    echo preg_replace('#a|b+c#', '!', 'bbbb'); //natija '!'
?>
```

Qidiruv shabloni: agar butun qator 'a', yoki butun qator bitta yoki undan ko'p 'b', yoki butun qator 'c' dan iborat bo'ladi bo'lsa, unda ularni '!' ga almashtiradi.

Bekslesh muammosi

Bekslesh()ni o'zi PHPning maxsus belgisi hisobalanadi. Bu qanday holatlarda kerak: **\$var = 'd'ombira'** – satrni ichiga yakka tirnoqni yozmoqchiman, lekin satrni o'zi yakka tirnoqli. Bunda hech nima ishlamaydi – PHP xatolik beradi.

Yakka tirnoqli belgini belgi sifatida ishlatish uchun – men uni bekslesh bilan ekranlashtirishim kerak: **\$var = 'd\ombira'** – bu ishlaydi. Agar beksleshni satrni ichida oddiy belgi sifatida qo'ymoqchi bo'lsamchi? Unda uni o'zini ham ekranlashtirishga to'g'ri keladi: **\$var = 'bekslesh \\ !'**. Natijada, o'zi qoladi.

Endi muntazam ifodalarga o'tamiz: bekslesh PHP ning maxsus belgisi hisoblanadi va shuningdek muntazam ifodalarda ham. Har doim ham sleshni ikki marta yozmaymiz (masalan, \s deb

yozamiz, `\s` emas), lekin, agar biz beksleshni o'zini belgi sifatida ishlatmoqchi bo'lsak, unda uni **4** martagacha yozishimiz kerak(2 martasi PHPning maxsus belgisi bo'lganligi tufayli uni ekranlashtirish uchun va yana 2martasi muntazam ifodalarning ham maxsus belgisi bo'lganligi tufayli uni ekranlashtirish uchun)

```
<?php
    echo preg_replace('#\\\\\\#', '!', '\\ \\ \\\\'); //natija
'! ! ! !'
?>
```

Qidiruv shabloni bir marta yozilgan beksleshni qidiradi.
“\\\\\\\\” ga e'tibor bering – biz PHP da beksleshni o'zini yozish uchun uni ikki marta takrorlashimiz kerak va rostakamida “\\ \\” ko'rinishda bo'ladi, shuning uhcun natija “! ! !”, “!! !! !!!” emas!

Preg_replace da almashinuvlar soni

preg_replace funksiyasining to'rtinchi ixtiyoriy parametri mavjud. Bu parametr nechta almashtirish qilish kerakligini ko'rsatadi:

```
<?php
    echo preg_replace('#a+#', '!', 'a aa aaa aaaa', 2);
//natija '! ! aaa aaaa'
?>
```

Funksiyamiz 2ta almashtirishni amalga oshirdi, e'tibor bering, hammasi almashmadi!

Muntazam ifodalar – III

Hozircha biz muntazam ifodalar bilan ishlaydigan bitta funksiya bilan tanishdik – preg_replace. Lekin biz faqatgina preg_replace funksiyasi bilan to'xtab qolmaymiz.

preg_match(muntazam ifoda, qayerni qidirish kerak) funksiyasi ham mavjud bo'lib, u satrni muntazam ifoda bilan mos kelishini tekshiradi.

Agar mutanosiblik ko'p bo'lsa – unda funksiya o'z ishini tugatish uchun faqat birinchi natijani chiqaradi.

Shuning uchun **preg_match** yoki 1, yoki 0 natija qaytaradi(true yoki false emas) va '**qidiriluvchi, satrda bormi yoki yo'q**' savoliga javob berish uchun ishlatiladi. 1 qaytarsa demak bor, 0 qaytarsa demak yo'q.

```
<?php
echo preg_match('#a+#', 'eee aaa bbb'); //natija 1
?>
```

preg_match funksiyasi ham false qaytarishi mumkin. Qaysi holatda? Biror xatolik yuz berganda.

Ko'pincha bu funksiya butun satrning muntazam ifoda bilan mosligini tekshirish uchun ishlatiladi. Misol uchun, joriy satr yaroqli email manzilmi yoki yo'qligini bilmoqchimiz:

```
<?php
echo preg_match('#^ [a-zA-Z-.]+@[a-z]+\.[a-z]{2,3} $#',
'my-mail@umail.uz');
?>
```

Qidiruv shabloni qidiradi:

[a-zA-Z-.]+ kichik yoki katta lotin harflar, nuqta yoki '-' bir yoki ko'p marta, keyin @

[a-z]+ keyin kichik lotin harflari bir yoki ko'p marta

\. Keyin nuqta

[a-z]{2,3} keyin kichik lotin harflari ikki yoki uch marta ishlatilganni(ru, by, com va hakozo)

Natijada biz 1 – ya'ni satr yaroqli email ekanligini ko'ramiz.

preg_match_all funksiyasi

preg_match faqat birinchi mosini topadi. Barcha mutanosiblikni topish uchun **preg_match_all(muntazam ifoda, qayerni qidirish kerak, topilganlar)** funksiyasi ishlatiladi.

```
<?php
echo preg_match_all('#a+#', 'eee aaa bbb', $m); //natija
1
?>
```

2 – holat

```
<?php
echo preg_match_all('#a+#', 'eee aaa aa bbb', $m);
//natija 2
?>
```

Funksiya barcha moslikni qidirdi va 2ta moslikni topdi

Uchinchi parametr

`preg_match_all` funksiyasining uchinchi parametri mavjud: u yerda ya'ni o'zgaruvchida barcha topilgan natijalar massivi bo'ladi:

```
<?php
echo preg_match_all('#a+#', 'eee aaa aa bbb a',
$matches); //natija 3
var_dump($matches);

//Topilgan natijalar massivi:
[0=>'aaa', 1=>'aa', 2=>'a']
?>
```

Kartmon

Biz siz bilan guruhash uchun oddiy qavs()larni ishlatib ko'rgandik. Endi ularning qo'shimcha xususiyati – kartmon sifatida ishlatilishini ham ko'rib chiqamiz:

```
<?php
echo preg_match_all('#x(a+)x#', 'xax xaax xaaax',
$matches); //natija 3
var_dump($matches);
?>
```

() ushbu kartmon ichidagi - \$matches massiviga ham kirgan bo'ladi:

```
<?php
[
```

```
0=>[0=>'xax', 1=>'xaax', 2=>'xaaax'], //bu  
topilgan qism satr  
1=>[0=>'a', 1=>'aa', 2=>'aaa'] //bu esa  
kartmonning tarkibi  
]  
?>
```

Kartmon nima deyotgan bo'lsangiz kerak. Kartmon bu biz qidirayotgan narsaning kerakli qismini saqlab qo'yishning bir usuli hisoblanadi. Misol uchun, biz domain.uz ko'rinishidagi domenni qidiryapmiz, lekin, biz faqatgina domen zonasini bilmoxchimiz(uz,ru,com yoki boshqa).

Muntazam ifodada bizga kerak bo'lgan 'domain.uz' ko'rinishidagi satr ko'rsatiladi, lekin bizni qiziqtirgan asosiy qismini – kartmonga joylashtiramiz. Keling yaxshisi namunada ko'rsataman:

```
<?php  
preg_match_all('#[a-z]+\.[a-z]{2,3}#', 'domain.uz  
site.com hello.by', $matches);  
?>
```

Matches o'zgaruvchisining tarkibida o'zgarish bo'ldi:

```
<?php  
var_dump($matches);  
[  
    0=>[0=>'domain.uz', 1=>'site.com', 2=>'hello.by'],  
    1=>[0=>'ru', 1=>'com', 2=>'by'] //bu  
kartmonimizning tarkibi  
]  
?>
```

Saqlamaydigan qavslar

Afsuski, qavslar () 2ta vazifani bajaradi – belgilarni guruhash va kartmon sifatida. Ammo bizga guruhash kerag-u, lekin kartmon kerak bo'lmasa-chi? Buni amalga oshirish uchun maxsus **saqlamaydigan qavslar**(?:belgi) mavjud. Ular guruhashlaydi, lekin kartmonga joylamaydi:

```
<?php
```

```
<?php
echo preg_replace('#(?:ab)+([a-z])#', '!$1!', 'ababx
abe'); //natija '!x! !e!'
?>
```

Bu yerda saqlamaydigan qavs ichida joylashgan ab belgilari kartmon ichida saqlanmaydi. Lekin keying muntazam ifodamiz, ya'ni [a-z] kartmonda topiladi(agar moslik topilsa).

Muntazam ifodalar – IV

Pozitiv va negative ko'rish

Ba'zida shunday turdag'i masalani yechish kerak: '**'aaa'** satrni top va uni '**'!'** ga almashtir, lekin faqat 'aaa' ni olida 'x' turgan bo'lsa(bunda 'x' o'zgarmas). Agar biz bu masalani oddiy yechaydigan bo'lsak:

```
<?php
echo preg_replace('#aaa#!', '!', 'aaa baaa'); //natija
'! baaa', 'x! baaa' ni kutgandik
?>
```

Bu shunchaki ko'rish va aynan o'sha belgini o'zini olmaslik uchun ishlatiladigan maxsus qavs(?=<) mavjud.

```
<?php
echo preg_replace('#(?<=x)aaa#!', '!', 'aaa baaa');
//natija 'x! baaa'
?>
```

(?=<) qavslar **ortga pozitiv nazar** deb ataladi. Pozitiv – 'x'(bizning holatda)mavjud bo'lsagina almashinuv amalga oshadi. Ortga negativ nazar ham mavjud bo'lib, u aksincha, amalga oshmasligini ifodalaydi:

```
<?php
//Agar 'aaa' oldida turgani 'x' bo'lmasa, unda '!' ga
almashadi:
echo preg_replace('#(?<!x)aaa#!', '!', 'aaa baaa');
//natija 'aaa b!'
?>
```

Muqobil variant sifatida yuqorida ta'kidlangan ikkita operatsiyaning quyidagilarni keltirishimiz mumkin: ortga pozitiv nazar (?:) va negative nazar (?!):

```
<?php
    //agar 'aaa' oldida turgani 'x' bo'lsa, unda '!' ga
    al mashadi:
    echo preg_replace('#aaa(?:=x)#', '!', 'aaax aaab');
    //natija '!x aaab'

    //agar 'aaa' oldida turgani 'x' bo'lmasa, unda uni
    '!' ga almashtiramiz:
    echo preg_replace('#aaa(?:!x)#', '!', 'aaax aaab');
    //natija 'aaax !b'

?>
```

preg_replace_callback funksiyasi

Ba'zida kartmonlar bilan qanaqadir operatsiyalarni amalga oshirishga to'g'ri kelib qoladi. Misol uchun, *raqamlardan tashkil topgan ikkita kartmonni qo'shish* yoki **barcha so'zning birinchi harfini topib** ularni katta registrga o'girish kabilar.

Bu holatda bizga preg_replace funksiyasiga o'xshagan **preg_replace_callback(muntazam ifoda, 'funksiya nomi', qayerni almashtirish)**, ishlatalamiz, ikkinchi parametr nimani almashtirish kerakligini qaytaradigan funksiya nomini qabul qiladi. Ushbu funksiyaning ichida kartmon mavjud bo'ladi.

Namuna

Topshiriq: '2+3=' ko'rinishidagi satr berilgan, uni xuddi shunga lekin tenglikdan keyin yig'indi natijasi yozilishi kerak. Sinab ko'ramiz:

```
<?php
    echo preg_replace_callback('#(\d+)\+(\d+)=#', 'sum',
    '2+3=');
    function sum($matches)
    {
```

```
        var_dump($matches);  
    }  
?>
```

Var_dump funksiyasi [**0=>'2+3', 1=>'2', 2=>'3'**] massivni qaytaradi – ya'ni u yerda topilgan sat va kartmon joylashgan bo'ladi. O'zgaruvchining nomini \$matches deb nomlash shart emas, u har xil nomga ega bo'lishi mumkin, asosiysi o'sha o'zgaruvchi biz chaqiradigan funksiyaning parametric bo'lishi kerak.

Kartmonga qanday kirish mumkinligini bilamiz, unda masalani yechishni davom etamiz:

```
<?php  
echo preg_replace_callback('#(\d+)\+(\d+)=#', 'sum',  
'2+3=');  
function sum($matches)  
{  
    $sum = $matches[1] + $matches[2]; //yig'indini  
olamiz  
    $result = $matches[0].$sum; // $matches[0] – bu  
'2+3=' satr  
    return $result;  
}  
//Natiжada '2+3=5' shuni olamiz  
?>
```

Muntazam ifodalar bilan ishlovchi qo'shimcha funksiyalar preg_quote(satr,[qo'shimcha belgilar]) – satrdagi muntazam ifodaning maxsus belgilarini ekranlashtiradi. Bu nima uchun kerak? Misol uchun, siz dinamik ravishda '#.\$var.'# muntazam ifodani yaratyapsiz va bu yerga maxsus belgilar tushmasligiga ishonchingiz komil emas, mana shu paytda ushbu funksiya qo'l keladi. Ikkinchi parametr ekranlashtirish uchun o'zingizni belgilaringizni qo'shishingiz mumkin

preg_grep

preg_grep(muntazam ifoda, massiv) – massiv qabul qiladi va muntazam ifoda bilan mos kelgan elementlar massivini qaytaradi.

preg_split

preg_split – muntazam ifodaga muvofiq satrni massivga ajratadi(explodega o'xshaydi, faqat muntazam ifoda bilan)

Modifikatorlar

Modifikatorlar – bu ('#.+#iu' – 'iu' – modifikator) ikkinchi cheklovchidan keyin yoziladigan buyruq. Ular muntazam ifodalarning xususiyatini o'zgartirishga imkon beradi.

Modifikator x

x – bu modifikator ishlatilsa muntazam ifodani probel ya'ni bo'sh joy tashalganni hisoblashini to'xtatadi va heshtegdan keyin izoh yozish(bir qatorli izoh) imkonini beradi:

```
<?php
    //Izoh bilan tushunarliroq bo'ladi lekin uzayib ham
ketadi:
    echo preg_replace('&
        .+? #har qanday bir yoki ko'p marta yozilgan belgi
        a #keyin a harfi
    &x', '!', 'satr');
?>
```

Yodda tuting: heshteg izoh yozish uchun ishlatilganda uni bir vaqtning o'zida cheklovchi sifatida ham ishlatish mumkin emas. O'rniga boshqa cheklovchi, ampersand yozish yoki boshqasini yozishingiz mumkin.

m va s modifikatorlari

Muntazam ifoda faqatgina bir-qatorli satrlarni emas balki ko'p-qatorli satrlarni ham qayta ishlay oladi:

```
<?php
$str = '
    Birinchi satr
    Ikkinci satr va ikkinchi qator
';
?>
```

Yuqoridagi satrni ko'p-qatorli satr sifatida ishlatamiz desak muntazam ifodada qo'llash uchun biz cheklovchidan keyin **m**

modifikatori yoxud bir-qatorli satr uchun **s** **modifikatorini** ishlatalamiz.

Ko'p qatorli satr

```
<?php
$str = '
    ^Birinchi satr$
    ^Ikkinci satr va ikkinchi qator$
';
?>
```

^ shlyapa belgisi har bir qism satrning boshiga, \$ dollar belgisi esa har bir qism satrga to'g'ri keladi.

Barcha satrlarning boshi uchun ^ o'rniga, \A, barcha satrlarning oxiri uchun \$ o'rniga, \z ishlataladi.

Bir qatorli satr

```
<?php
$str = '^
    Birinchi satr
    Ikkinci satr va ikkinchi qator
$';
?>
```

^ shlyapa belgisi barcha satrning boshiga, \$ esa oxiriga to'g'ri keladi.

Modifikator i

i – registr, ya'ni harflarning katta-kichikligini hisobga olishni rad etadi.

Modifikator u

u – utf-8 kadirovkasi bilan to'g'ri ishslash uchun buni ishlatalish kerak. Masalan, kiril yozushi bilan muammo bo'lmasligi uchun.

Ma'lumotlar bazasi bilan ishslash

Ma'lumotlar bazasi nima?

Ma'lumotlar bazasi(qisqartmasi MB) – bu sayt ma'lumotlari saqlanadigan joy. Bu sahifa matni, foydalanuvchining login va parollari ro'yhati, mahsulotlari katalogi va boshqalar bo'lishi mumkin. Ma'lumotlar bazasi jadvallardan tashkil topadi. Jadval nima: bu qator va ustunlar. Qator va ustunlar kesishmasida yacheyka ya'ni kataklar joylashgan bo'ladi. Ma'lumotlar bazasida odatda ular maydonlar deb ataladi. Buni oson qilib Excel misolida keltirishim mumkin. Ma'lumot bazasi hujjat(kitob)ga o'xshaydi, jadval esa kitobning har bir sahifasi.

PhpMyAdmin

Ma'lumot bazasini tahrirlash uchun ko'pincha PhpMyAdmin dasturi ishlataladi. **PhpMyAdmin** (Pieychpimayadmin deb o'qiladi, ko'pincha PMA deb qisqartirilib yoziladi) – bu to'g'ridan-to'g'ri brauzerning o'zida ma'lumotlar bazasi bilan ishslash uchun dastur. SQL tilini bilmasdan turib, veb-interfeys orqali jadvalni tarkibini o'zgartirish yoki yangi baza va jadval hosil qilishingiz mumkin.

PhpMyAdmin ga oid masalalar

PhpMyAdminni ochasiz va interfeysni o'rghanishni boshlaymiz. Agar siz OpenServer ishlata yotgan bo'lsangiz, topshiriqlar panelida ko'rinish turgan yashil bayroqchani ustiga sichqonchaning o'ng tomonini bosasiz va "Дополнительно" kontekst menyusini bosib, "PhpMyAdmin" menyusini tanlaysiz. Va boshlaymiz:

1. Test nomli ma'lumot bazasi yaratish

2. Uni ichida users nomli jadval hosil qiling
3. Bu jadval ichiga 4 ta maydon qiling:
 - **id**(unga AUTO_INCREMENT yoki A.I deganiga galochka qo'yish kerak), turi – integer
 - **name**, tur – varchar, 32ta belgi
 - **age**, tur – integer
 - **birthday**(inglizcha tug'ilgan kun), tur – date
4. ‘вставить’ vkadkasini toping va u yordamida bir necha qatorlarni jadvalning ichiga joylashtiring. id maydoni to'ldirilishi kerak emas! Ular avtomatik to'ldiriladi.
5. Qanaqadir ma'lumotni qayta tahrirlang
6. Yana qanaqadirini o'chiring
7. Jadval va barcha ma'lumot bazalari uchun kodirovkasini o'zgartiring(utf8_general_ci ga)
8. Jadval nomini qayta boshqasiga nomlang
9. Ma'lumot bazasi nomini ham qayta nomlang

AUTO_INCREMENT

Yaratgan id maydonimizga e'tiboringizni qarating. Biz unga **AUTO_INCREMENT** ga galochka qo'yganmiz. Bu juda muhim bosqich! Endi biz bu maydonga yangi ma'lumot kiritadigan bo'lsak uning aydisi avtomatik tarzda unikal nomer bilan nomerlanadi. Agar biz 30 raqamli id maydonini o'chiradigan bo'lsak, keying safar, umuman bunaqa raqam uchramaydi. Hammasi unikal!

Nimaga id maydon kerak?

Chunki, biz har bir ma'lumotni biror usul bilan topishimiz kerak, bu bizga unikal hamda oson bo'lishi kerak. Albatta bizga bu unikal raqam ya'ni id qo'l keladi.

O'zgaruvchi turlari

SQL da yetarlicha o'zgaruvchi turlari mavjud, ammo ko'pincha quyidagilar ishlatiladi:

- **integer** – butun son
- **text** – katta matnli maydon
- **varchar** – katta bo'limgan matnli maydon, bunda biz uning o'lchamini bilishimiz kerak (u sonning ikkinchi darajasi ya'ni: 8,16,32,64,128,256 va hakozo)
- **date** – sanani saqlash uchun maydon(SQL formatda sana quyidagicha saqlanadi: yil-oy-kun, misol uchun: 2013-06-24).

PHP da MySql bilan qanday ishlaydi?

PHP da ma'lumot bazasi bilan ishlash uchun quyidagi uchta funksiyadan foydalaniladi:

- **mysqli_connect** – ma'lumot bazasi va server bilan ulanish
- **mysqli_query** – MB ga so'rov yuborishning universal funksiyasi, u yordamida hamma narsani qilish mumkin
- **mysqli_error** – xatolik chiqarish+

Dars mobaynida har bir funksiya haqida batafsil o'rghanasiz.

MB bilan aloqa o'rnatish

PHP da ma'lumot bazasi bilan ishlashdan oldin baza joylashgan server bilan aloqani o'rnatish kerak.

Bu **mysqli_connect** funksiyasi yordamida qilinadi. Bu funksiya uchta parametr qabul qiladi: 1-host nomi(server), foydalanuvchi nomi, parol va ma'lumot bazasi nomi.

Agar kompyuteringizda ishlayotgan bo'lsangiz, ya'ni openserverda unda bu '**localhost**', '**root**' va parol bo'sh satr

bo'ladi. Agar ma'lumot bazasi internetda bo'lsa unda bu ma'lumotlarni sizga sizning serveringiz berishi kerak.

Ma'lumot bazasi bilan aloqani o'rnatamiz:

```
<?php
    //MB bilan aloqa o'rnatamiz:
    $host = 'localhost'; //host nomi, lokal
kompyuterda bu localhost
    $user = 'root'; //foydalanuvchi nomi, standart bu
root
    $password = ''; //parol, standart bo'sh bo'ladi
    $db_name = 'test'; //MB nomi

    //Shaxsiy ma'lumotlarimizdan foydalaniib, MB ga ulanamiz:
    $link = mysqli_connect($host, $user, $password,
$db_name);

    /*
        Ulanish $link o'zgaruvchisiga yozildi(bu
resource),
    */
?>
```

MB ga so'rov yuborish

Ma'lumotlar bazasiga so'rovlari bu mysqli_query funksiyasi ichiga uzatiladigan oddiy satrlar.

```
<?php
    //mysqli_query funksiyasi ichida oddiy satr bo'ladi:
    $link = mysqli_query($link, "SELECT*FROM workers WHERE
id>0");

    //Bu qatorni o'zgaruvchi yordamida formatlash mumkin:
    $table = 'workers'; //jadval nomini o'zgaruvchiga
yozamiz
    mysqli_query($link, "SELECT*FROM ".$table." WHERE
id>0");
?>
```

SQL buyruqlar hamisha katta registrda yozilishi kerak, qolganlari kichkina. Bu **SELECT, UPDATE, FROM, DELETE, WHERE** va boshqa shu turdag'i komandalarga tegishli. Agar bularni kichik registrda yozsangiz xato

bo'lmaydi, lekin katta registrda yozish umum qoida sifatida qabul qilingan.

Ma'lumot bazasidagi xatoliklarni ushslash

Ko'pgina boshlovchi dasturchilar ma'lumot bazasidan qaytgan xatoliklarni ushslashga odatlanishmaydi. Shuning uchun, ma'lumotlar bazasi bilan ishslashda ular doimo qiyinchilikka duch kelishadi. Tushunarsiz bir nima ishlamasa, xatolikni ular ko'rmaydi va PHP mysql xatolikni chiqarmaydi(agar belgilanmagan bo'lsa).

Xatolikni chiqarish uchun or die(mysqli_error(\$link)) konstruksiyasiga amal qilinadi. Buni har bir mbga so'rov yuborilganda qo'shish kerak.

Misol uchun:

```
mysqli_query($link, $query( or die(mysqli_error($link));
```

Bu usulda SQL sintaksisida vujudga kelgan xatoliklar haqida xabardor bo'lasiz.

Kodirovka bilan muammolar

Ko'pincha boshlovchi dasturchilarda kodirovka bilan muammolar ko'p bo'ladi, agar rus tilida bo'lsa. Bazaga ma'lumot olinganda qanaqadir abracadabra matnlar yoki so'roqchalar chiqib ketadi.

Bunday muammo bo'lmasligi uchun quyidagi qoidaga amal qiling:

- Ma'lumot bazasini utf8_general_ci kodirovkasida yaratish
- PHP skripti utf8 kodirovkasida bo'lish
- MB dagi jadval utf8_general_ci kodirovkasida bo'lishi

- Har ehtimolga qarshi mysqli_connect kommandasidan keyin quyidagi so'rovni qo'shish: **mysqli_query(\$link, "SET NAMES 'utf8'");**

Ishlashni sinab ko'ramiz

Ishni boshlashdan oldin uni bazaga qanaqadir so'rov qilish orqali sinab ko'rish kerak:

```
<?php

//MB bilan aloqa o'rnatamiz:

$host = 'localhost'; //host nomi, lokal kompyuterda
bu localhost

$user = 'root'; //foydalanuvchi nomi, standart bu
root

$password = ''; //parol, standart bo'sh bo'ladi

$db_name = 'test'; //MB nomi


//Shaxsiy ma'lumotlarimizdan foydalanib, MB ga ulanamiz:

$link = mysqli_connect($host, $user, $password,
$db_name);

//Kodirovkani o'rnatamiz:

mysqli_query($link, "SET NAMES 'utf8'")


//Matnli so'rovni yaratamiz:

$query = "SELECT * FROM workers WHERE id > 0";


//MB ga so'rov yuboramiz uni $result ga yozamiz:

$result = mysqli_query($link, $query) or
die(mysqli_error($link));
```

```
//Tekshiramiz:  
var_dump($result);  
?>
```

Qanday qilib natijani olish mumkin?

Bazaga so'rov qilganimizdan keyin \$result o'zgaruvchisida bazadan qaytgan natija bo'ladi. Ammo u bizga kerak bo'lgan PHP shaklda bo'lмаган bo'ladi. Natijani normal ko'rinishda olish uchun ushbu koddan foydalanib ko'ring:

```
<?php  
//So'rov yaratamiz va $resultga yozamiz:  
$result = mysqli_query($link, $query) or  
die(mysqli_error($link));  
  
//natija  
for ($data = []; $row =  
mysqli_fetch_assoc($result); $data[] = $row);  
?>
```

Keyingi qator qanday ishlayapti?

mysqli_fetch_assoc funksiyasi ma'lumotlar bazasi bizga yuborgan natijalarni har bir qatorini ketma-ket o'qiydi. For siklida ma'lumotlar bazasidan olingan natijani qatorma-qator o'qiymiz.

Sikl oxirgi qatorga yetganda mysqli_fetch_assoc false qaytaradi va for sikli o'z ishini yakunlaydi.

MB buyruqlar

SELECT buyrug'i

Avvalgi kodlarimizda ko'rib ulgurgran kodingiz, **SELECT** bilan tanishamiz. **SELECT** jadvalimizdan bir nechta qatorlarni olishga imkon beradi. Uning sintaksisi quyidagicha:

//TANLA barcha_ustunlarni falonchi jadvaldan QAYERDAKI qanaqadir_shart
SELECT * FROM имя_таблицы WHERE qatorni_tanlash_bo'yicha_shart;

Skriptingizda bo'lishi kerak bo'lgan butun kodlarni ko'rsataman. Quyidagi kodni bir qismini baza bilan ulanishni sinab ko'rishda sinab ko'rganmiz. Demak, qandaydir shart asosida bazadan tanlaymiz, bizning holatda bu id > 0 ya'ni id 0 dan katta bo'lgan barcha ma'lumotlarni tanlanadi.

```
<?php

//MB bilan aloqa o'rnatamiz:

$host = 'localhost'; //host nomi, lokal kompyuterda
bu localhost

$user = 'root'; //foydalanuvchi nomi, standart bu
root

$password = ''; //parol, standart bo'sh bo'ladi

$db_name = 'test'; //MB nomi

//Shaxsiy ma'lumotlarimizdan foydalanib, MB ga ulanamiz:

$link = mysqli_connect($host, $user, $password,
$db_name);

//Kodirovkani o'rnatamiz:

mysqli_query($link, "SET NAMES 'utf8'");

//Matnli so'rovni yaratamiz:

$query = "SELECT * FROM workers WHERE id > 0";

//MB ga so'rov yuboramiz uni $result ga yozamiz:
```

```
$result = mysqli_query($link, $query) or  
die(mysqli_error($link));  
  
//Tekshiramiz:  
  
var_dump($result);  
  
?>
```

So'rovni **workers** jadvaliga qildik. var_dump funksiyasi \$data massividagi jadvalimizning barcha qatorlarini chiqarmoqda.

Shartga yana nima yozish mumkin?

Minimum kerak bo'ladigan komandalar: = tenglik, != teng emas, <> teng emas, > katta, < kichik, >= katta yoki teng, <= kichik yoki teng.

SELECT bilan ishlash namunalar

Quyida biz workers jadvalidan id si 2 ga teng bo'lgan ishchini tanlayapmiz:

```
<?php  
 // $data o'zgaruvchisida Ahmadga oid bo'lgan bitta qator  
 // mavjud bo'ladi:  
 $query = "SELECT * FROM workers WHERE id=2";  
?>
```

Jadvaldan endi ism bo'yicha tanlaymiz:

```
<?php  
 // $data o'zgaruvchida Javlonga oid ma'lumot bor:  
 $query = "SELECT * FROM workers WHERE name='Javlon'";  
?>
```

Murakkabroq amallar: OR yoki AND

Shartli tanlashda **OR** va **AND** so'zlari yordamida yanada qiyinroq bo'lgan va kerakli bo'lgan shart tuzishingiz mumkin. if konstruksiyasidagi kabi ishlaydi.

Workers jadvalidan **oylik maoshi 500\$ va yoshi 23** ga teng bo'lgan ishchini tanlaymiz:

```
<?php
$query = "SELECT * FROM workers WHERE salary=500 AND
age=23";
?>
```

Workers jadvalidan oylik maoshi 450 \$ dan 900\$ gacha bo'lganlarni tanlab olamiz:

```
<?php
$query = "SELECT * FROM workers WHERE salary>450 AND
salary<900";
?>
```

Shartlarni guruhash

Agar sizga OR va AND ishlataligan qiyinroq tanlash shartlari kerak bo'ladiga bo'lsa, ularni () qavslar yordamida guruhash olishingiz mumkin. Guruhashlangan shart ustuvorligi yuqori bo'ladi. Keling, workers jadvalidan yoshi 20 dan 27 gacha bo'lganlar yoki 100\$ oylik maoshdan 300\$ gacha bo'lganlarni tanlaymiz

```
SELECT * FROM workers WHERE (age>20 AND age<27) OR
(salary>300)
```

Ustunlarni tanlash

SELECT orqali biz hamma ustunlarni tanlab olmasdan, o'zimizga kerak bo'lgan va ishlataligan ustunlarni tanlasak ham bo'ladi. Masalan, workers ya'ni ishchilar jadvali kerak menga, lekin men bazamdag'i hamma, foydalanuvchilar, maqolalar kabi jadvallarni ham tanlab olmoqdaman. Bu esa ishni susaytirishi mumkin.

```
<?php
$query = "SELECT name, age FROM workers WHERE id>0";
?>
```

INSERT buyrug'I – Bazaha ma'lumot joylashtirish

INSERT buyrug'I yordamida jadvalga yangi ma'lumot qo'shish mumkin. Sintaksisi quyidagicha:

```
//jadval_nomi jadvalida maydon1ga qiymat1 ni,  
maydon2=qiymat2 ini qo'sh  
INSERT INTO jadval_nomi SET maydon1=qiymat1, maydon2=  
qiymat2, maydon3=qiymat3...;
```

Workers jadvalimizga yangi ishchi qo'shib ko'ramiz:

```
<?php  
  
$query = "INSERT INTO workers SET name='Otabek', age=30,  
salary=1000";  
  
?>
```

INSERTning boshqacha sintaksisi

INSERT buyrug'inining alternative sintaksisi mavjud:

```
<?php  
  
//jadval_nomi dagi (maydon1, maydon2) ga (qiymat1,  
qiymat2) qiymatlarni qo'sh  
  
$query = "INSERT INTO workers (maydon1, maydon2...) VALUES  
(qiymat1, qiymat2...);  
  
?>
```

Workers jadvalimizga yangi ishchi qo'shib ko'ramiz:

```
<?php  
  
$query = "INSERT INTO workers (name, age, salary) VALUES  
('Otabek', 30, 1000)";  
  
?>
```

INSERT orqali ma'lumotni ommaviy kiritish

INSERT yordamida faqatgina bitta ma'lumot emas balki bir nechta ma'lumotni bir vaqtning o'zida qo'shishingiz mumkin. Sintaksis quyidagicha:

```
<?php  
  
$query = "INSERT INTO имя_таблицы (maydon1, maydon2...)  
VALUES (qiyamat1, qiyamat2...), (qiyamat1,  
qiyamat2...)...";  
  
?>
```

Keling bir vaqt ni o'zida 3ta ishchi qo'shib ko'ramiz:

```
<?php  
  
$query = "INSERT INTO workers (name, age, salary)  
VALUES ('Otabek', 30, 1000), ('Jahon', 25, 500),  
('Tohir', 27, 1500)";  
  
?>
```

DELETE komandasi

DELETE buyrug'i yordamida jadvaldan ma'lumotlarni o'chirish mumkin. Uning sintaksisi ham SELECT nikiga juda o'xshash:

```
<?php  
  
//jadval_nomi dan shu shart bo'yicha ma'lumotni o'chir  
"DELETE FROM jadval_nomi WHERE  
qatorlarni_ochirish_boyicha_shart";  
  
?>
```

Unda, 6 id dagi ishchimizning ma'lumotlarini o'chirib ko'ramiz:

```
<?php  
  
$query = "DELETE FROM workers WHERE id=6";  
  
?>
```

ORDER BY, LIMIT, COUNT, LIKE buyruqlari

Ushbu dars – SQL ma'lumotlar bazasi bilan ishlash to'g'risidagi o'tkan mavzuning davomi hisoblanadi. Hozir biz siz bilan, ma'lumotlarni saralash, sonini cheklash, ularni hisoblash hamda ma'lumotlararo qidiruvni amalga oshiramiz

ORDER BY – saralash

ORDER BY buyrug'i yordamida oladigan natijamizni oldindan saralashimiz mumkin. **workers** jadvalimizdan barcha ishchilarni tanlaymiz va ularni yoshlari bo'yicha saralab olamiz:

```
<?php
 // $data o'zgaruvchisida qatorlar 0 dan katta bo'lgan
 barchasi yosh bo'yicha va o'sish tartibida saralanadi:
 $query = "SELECT * FROM workers WHERE id>0 ORDER BY
 age";
?>
```

Agar biz ularni aksincha, kamayish tartibida saralamoqchi bo'lsak, bunday qilamiz:

```
<?php
 // DESC – kamayish tartibida saralaydi, ASC – o'sish
 tartibida. ASC yozilmasa //ham standart ravishda ASC deb
 ko'rilib:
 $query = "SELECT * FROM workers WHERE id>0 ORDER BY age
 DESC";
?>
```

LIMIT – miqdorni cheklash

LIMIT buyrug'i yordamida natijadagi qatorlar sonini cheklashimiz mumkin. Quyidagi namunada qatorlar sonini 2tagacha cheklaymiz:

```
<?php
 // $data o'zgaruvchisida faqat 2ta birinchi qatorlar
 bo'ladi:
 $query = "SELECT * FROM workers WHERE id>0 LIMIT 2";
?>
```

Ammo, bu hammasi emas! LIMIT yordamida rezultat ya'ni bazadan olgan qatorlarimiz orasidan bir nechtasini tanlab olish mumkin. Quyidagi namunada biz 2-qatordan 5ta ma'lumotni tanlab olayapmiz:

```
<?php
$query = "SELECT * FROM workers WHERE id>0 LIMIT 2,5";
?>
```

LIMIT va ORDER birligida:

```
<?php
// $data o'zgaruvchisida 2-qatordan 5ta ma'lumot
// joylanadi, id o'sish tartibi bo'yicha saralanadi
$query = "SELECT * FROM workers WHERE id>0 ORDER BY id
DESC LIMIT 2,5";
?>
```

COUNT – sonini hisoblaymiz

COUNT buyrug'i yordamida tanlovdagi qatorlar sonini hisoblash mumkin. Sintaksisi:

```
<?php

// $result da qatorlar soni yotadi:

$query = "SELECT COUNT(*) as count FROM workers WHERE
id>0";

$result = mysqli_query($link, $query) or die(
mysqli_error($link) );

$count = mysqli_fetch_[$result];

// $count da ['count'=>soni] massivi yotadi:
var_dump($count);

?>
```

COUNT bilan ko'pchilik yo'l qo'yadigan xatolik: COUNT(*) konstruksiyasi yopishtirib yozilishi kerak, uning ichida(yulduzcha atrofi) hech qanday bo'shliq bo'lmasligi kerak. Aks holda, ishlamaydi

LIKE – qidiruv bilan ishlaymiz

LIKE (inglizcha. O'xshash) buyrug'i yordamida ma'lumotlar bazasida qidiruvni amalga oshirishimiz mumkin. Namunaga qarang:

```
<?php

// jadvaldan name ustunida ya bilan tugaydigan har qanday
qatorni barchasini tanla

$query = "SELECT * FROM workers WHERE name LIKE '%я'";

/*
Misol uchun bazada Имя, Фамилия, Оливия satrlar bor.
Natijada bizga o'sha qatorlar chiqariladi.

*/
?>
```

E'tibor bering **LIKE** dan keyin satrda yozilgan **%** belgisi, har qanday so'zning oxirida foiz belgisidan keying belgi bo'lsa degan ma'noni beradi. Agar **_** belgisi yozilgan bo'lsa, LIKE '**2_**', unda boshlanishi 2 bilan boshlanadigan har qanday raqam yoki satr degan ma'noni beradi.

Iqtibos belgilari ``

SQL so'rovining nozik jihatiga e'tibor bering: quyidagi so'rovimiz ishlamaydi sababi **from** jadval nomi **FROM** buyrug'i bilan mos:

```
<?php
$query = "SELECT * FROM from";
?>
```

Bunday muammoli vaziyatdan biz u so'zni rus tilida kavichka, o'zbek tilida iqtibos belgisi ichiga yozish bilan chiqib ketishimiz mumkin, mana bunday:

```
<?php
$query = "SELECT * FROM `from`";
?>
```

Ma'lumot bazasi bilan ishlash bo'yicha mashg'ulot ishlari

1-amaliyot

id	name	age	salary
1	Sanjar	23	400
2	O'ktam	24	500
3	Qodir	25	600

Bazada yuqoridagidek ishchilar va ularning yoshi hamda maoshi haqidagi ma'lumotlar jadvali bor deylik. Keling endi o'sha jadvalni brauzerga xuddi shu ko'rinishda chiqaramiz. Buning uchun birinchi HTML kodimizni yozib olishimiz kerak:

```
<table>
<tr>
    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
</tr>
<tr>
    <td>1</td>
    <td>Sanjar</td>
    <td>23</td>
    <td>400</td>
</tr>
```

```
<tr>
    <td>2</td>
    <td>0' ktam</td>
    <td>24</td>
    <td>500</td>
</tr>

<tr>
    <td>3</td>
    <td>Qodir</td>
    <td>25</td>
    <td>600</td>
</tr>
</table>
```

Qanday qilib buni amalga oshiramiz: HTML kod qismini qo'lida yozib chiqdik, keying qismini PHP o'zi yaratadi. HTML static qismini qo'lida yozamiz – table tegi va birinchi tr tegi(jadval nomlari bilan). Mana endi jadvalning asl ma'lumotlarini PHP bazadan olib o'zi yaratadi.

```
<table>
<tr>
    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
</tr>
<?php
    //PHP kod yozamiz,
```

```
?>  
</table>
```

Biz yozadigan PHP kod ma'lumotlar bazasiga so'rov yuborishi kerak, ishchilar massivini olib undan tegishli miqdordagi td li tr larni yaratish kerak. Keling workers jadvalidan barcha ishchilarni olamiz va uni \$data massiviga yozamiz(ma'lumot bazasi bilan ulanishni biz avvalgi mavzularimizda yozib o'tgandik, shuning uchun buni qisqa qilish uchun yozmay ketamiz):

```
<table>  
  
<tr>  
    <th>id</th>  
    <th>name</th>  
    <th>age</th>  
    <th>salary</th>  
  
</tr>  
  
<?php  
  
    $query = "SELECT * FROM workers";  
  
    $result = mysqli_query($link, $query) or  
        die( mysqli_error($link) );  
  
    for ($data = []; $row = mysqli_fetch_assoc(  
        $result); $data[] = $row)  
        var_dump($data);  
  
    ?>  
</table>
```

Biz olgan massiv quyidagi ko'rinishda bo'ladi:

[

```
[ 'id' => '1', 'name' =>
    'Sanjar', 'age' => '23', 'salary' => '400'],
[ 'id' => '2', 'name' =>
    'O\'ktam', 'age' => '24', 'salary' => '500'],
[ 'id' => '3', 'name' =>
    'Qodir', 'age' => '25', 'salary' => '600'],
]
```

Foreach sikli yordamida uni siklga qo'yamiz hamda tr qator va td ma'lumot ustunlarini hosil qilamiz:

```
<table>
<tr>
    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
</tr>
<?php
    $query = "SELECT * FROM workers";
    $result = mysqli_query(
        $link, $query) or die(mysqli_error($link));
    for ($data = []; $row = mysqli_fetch_assoc(
        $result); $data[] = $row);

    $result = '';
    foreach ($data as $elem) {
        $result .= '<tr>';
        $result .= '<td>' . $elem['id'] . '</td>';
        $result .= '<td>' . $elem['name'] . '</td>';
        $result .= '<td>' . $elem['age'] . '</td>';
        $result .= '<td>' . $elem['salary'] . '</td>';
        $result .= '</tr>';
    }
    echo $result;
}</?php
```

```
$result .= '<td>' . $elem['id'] . '</td>';
$result .= '<td>' . $elem['name'] . '</td>';
$result .= '<td>' . $elem['age'] . '</td>';
$result .= '<td>' .
    $elem['salary'] . '</td>';

$result .= '</tr>';
}

echo $result;
?>
</table>
```

Topshirig'imiz yechildi. Agar bu kodni(MB ga ulanishni amalga oshirgan holda) ishga tushirsangiz tegishli jadvalni ko'rishingiz mumkin.

2-amaliyot

Yuqoridagi amaliyotimizda ischilarning ro'yhatini HTML jadval ko'rinishda bazadan olib qo'ygan edik. Endi ushbu amaliyotimizga o'zgartirish kiritamiz, ya'ni, ishchini brauzerimizdan turib bazadan o'chirish imkonini qo'shamiz. Buning uchun biz jadvalimizga yangi ustun qo'shamiz, ya'ni ishchini o'chirish uchun havola:

```
<table>
<tr>
    <th>id</th>
    <th>name</th>
```

```
<th>age</th>
<th>salary</th>
<th>delete</th>
</tr>
<tr>
<td>1</td>
<td>Sanjar</td>
<td>23</td>
<td>400</td>
<td><a href="">o'chirish</a></td>
</tr>
<tr>
<td>2</td>
<td>O'ktam</td>
<td>24</td>
<td>500</td>
<td><a href="">o'chirish</a></td>
</tr>
<tr>
<td>3</td>
<td>Qodir</td>
<td>25</td>
<td>600</td>
<td><a href="">o'chirish</a></td>
</tr>
</table>
```

Havolaga bosganda hozirgi turgan sahifamizda qolamiz lekin so'rovni GET orqali yuboramiz. Uning qiymati sifatida ishchining ID si kiritiladi:

```
<table>
<tr>
    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
    <th>delete</th>
</tr>
<tr>
    <td>1</td>
    <td>Sanjar</td>
    <td>23</td>
    <td>400</td>
    <td><a href="?del=1">o'chirish</a></td>
</tr>
<tr>
    <td>2</td>
    <td>o'ktam</td>
    <td>24</td>
    <td>500</td>
    <td><a href="?del=2">o'chirish</a></td>
</tr>
<tr>
    <td>3</td>
```

```
<td>Qodir</td>
<td>25</td>
<td>600</td>
<td><a href="?del=3">o'chirish</a></td>
</tr>
</table>
```

Bu qanday ishlaydi: agar biz havolani bossak, masalan Sanjar uchun belgilangan havolani. Keyin biz bazadagi Sanjar ning ID raqamiga muvofiq tarzda del parametri qiymati 1 ga teng bo'lган GET so'rovini yuboramiz.

PHP kodda biz o'chirish uchun ishchining id sini `$_GET['del']` ni yozish orqali olishimiz mumkin va uni quyidagicha o'chiramiz:

```
<?php
$del = $_GET['del']; // o'chirish uchun id ni olvolamiz
$query = "DELETE FROM workers WHERE
          id=$del"; // o'chirish uchun moslaymiz
mysqli_query($link,
              $query) or die(mysqli_error($link)); // o'chiramiz
?>
```

Bu operatsiyani har sahifaga kirganda bajarmaganligimiz tufayli, GET parametri brauzerning manzil qismida mavjud bo'lmasligi mumkin. Shuning uchun uni avval isset funksiyasi yordamida mavjudmi yoki yo'q shuni tekshirib olamiz va undan keyin o'chirishni amalgalash oshiramiz.

```
<?php
if (isset($_GET['del'])) {
```

```
$del = $_GET['del'];

$query = "DELETE
          FROM workers WHERE id=$del";

mysqli_query($link,
              $query) or die(mysqli_error($link));

}

?>
```

Keling endi avvalgi amaliyotimizda ma'lumotni HTML jadval sifatida chiqqarganimizni eslab olamiz:

```
<table>

<tr>
    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
</tr>

<?php

$query = "SELECT * FROM workers";
$result = mysqli_query(
    $link, $query) or die(mysqli_error($link));
for ($data = []; $row = mysqli_fetch_assoc(
    $result); $data[] = $row);

$result = '';
foreach ($data as $elem) {
    $result .= '<tr>' .
```

```
$result .= '<td>' . $elem['id'] . '</td>';
$result .= '<td>' . $elem['name'] . '</td>';
$result .= '<td>' . $elem['age'] . '</td>';
$result .= '<td>' .
    $elem['salary'] . '</td>';

$result .= '</tr>';
}

echo $result;
?>
</table>
```

Endi jadvalga yana bitta yacheyka, o'chirish uchun havola yacheykasini qo'shamiz:

```
<table>

<tr>

    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
    <th>delete</th>

</tr>

<?php

    $query = "SELECT * FROM workers";
    $result = mysqli_query($link, $query) or
```

```
die( mysqli_error($link) );

for ($data = []; $row = mysqli_fetch_assoc(
    $result); $data[] = $row);

$result = '';
foreach ($data as $elem) {

    $result .= '<tr>';

    $result .= '<td>' . $elem['id'] . '</td>';
    $result .= '<td>' . $elem['name'] . '</td>';
    $result .= '<td>' . $elem['age'] . '</td>';
    $result .= '<td>' .
        $elem['salary'] . '</td>';

    $result .= '<td>' .
        $elem['salary'] . '</td>';

    $result .= '<td>
        <a href="">удалить</a></td>';

    $result .= '</tr>';

}

echo $result;

?>

</table>
```

Endi havolaga bosganda biz o'chirmoqchi bo'lgan ishchining id si mavjud bo'lgan GET parametri uzatiladi. Hali o'chirish

uchun SQL so'rovni amalga oshirmaganimi uchun bu o'chirish tugmasi ishlamaydi.

Keling, darsning boshida yozilgan o'chirish kodimizni qo'shamiz. E'tibor bering, bu ma'lumotlar bazasidan ishchilarni olishdan oldin qo'shilishi kerak, shunda ishchini o'chirganingizda, u bazadan yo'qoladi sshundan keyingina biz qolgan ishchilarni olib HTML jadvalimizda ko'rsatishimiz mumkin:

```
<table>
<tr>
    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
    <th>delete</th>
</tr>
<?php
    // Ma'lumotni olgunimizga qadar ishchini o'chirib
    bo'lamiz:
    if (isset($_GET['del'])) {
        $del = $_GET['del'];
        $query = "DELETE
                    FROM workers WHERE id=$del";
        mysqli_query($link,
                    $query) or die(mysqli_error($link));
    }
}
```

```
// Barcha ishchilarni olamiz:  
  
$query = "SELECT * FROM workers";  
  
$result = mysqli_query(  
    $link, $query) or die(mysqli_error($link));  
  
for ($data = []; $row = mysqli_fetch_assoc(  
    $result); $data[] = $row);  
  
  
// Ekranga chiqaramiz:  
  
$result = '';  
  
foreach ($data as $elem) {  
  
    $result .= '<tr>';  
  
  
    $result .= '<td>' . $elem['id'] . '</td>';  
    $result .= '<td>' . $elem['name'] . '</td>';  
    $result .= '<td>' . $elem['age'] . '</td>';  
    $result .= '<td>' .  
        $elem['salary'] . '</td>';  
    $result .= '<td>' .  
        $elem['salary'] . '</td>';  
    $result .= '<td><a href="?del=' .  
        $elem['id'] . '">o`chirish</a></td>';  
  
  
    $result .= '</tr>';  
}  
  
  
echo $result;  
?>
```

```
</table>
```

Agar bu kodni ishga tushirsangiz, barcha ishchilarning ro'yhatini brauzerda ko'rasiz. Agar qaysidir xodimni o'chirish uchun tegishli havolaga o'tsangiz, brauzer sahifasi yangilanadi, xodim esa baza va jadvaldan o'chirilib, sizga yangilangan jadval brauzerda ko'rindi.

3-amaliyot

Keling, endi yana qo'shimcha kiritamiz. Endilikda HTML forma yordamida yangi xodimni bazaga qo'shamiz:

```
<form action="" method="POST">

<input name="name">

<input name="age">

<input name="salary">

<input type="submit"

       value="Xodim qo'shish">

</form>
```

Endi formadan olingan ma'lumotlarni ma'lumotlar bazasiga INSERT so'rovi yordamida saqlash kodini yozamiz:

```
<?php

$name = $_POST['name'];

$age = $_POST['age'];

$salary = $_POST['salary'];

$query = "INSERT INTO workers SET name= '$name', age='$age', salary='$salary'";

mysqli_query($link,
$query) or die(mysqli_error($link));
```

```
?>
```

Keling endi formamiz va undagi ma'lumotni qayta ishlaydigan kodlarimizni bitta faylga birlashtiramiz. Qayta eslataman, bizning script 2 marta bajariladi: birinchi martada foydalanuvchi sahifaga kiradi va formani to'ldirib uni jo'natadi. Jo'natilganidan keyin biz xuddi formani jo'natgan sahifamizni ko'ramiz va script qayta ishga bajariladi, ammo formadagi ma'lumot bazaga yuboriladi.

Agar formangiz yuborish metodi POST bo'lsa unda bu ma'lumotlarni \$_POST superglobal o'zgaruvchisida saqlanadi. Bizning PHP kod foydalanuvchi sahifaga kirgan zahoti emas balki forma jo'natilgan ondayoq ishlashi kerak. Bunga erishish uchun IF konstruksiyasidan foydalanamiz. Masalan, shart qo'yishimiz mumkin POST bo'shmasmi, agar bo'sh bo'lmasa unda kodimizni saqlash uchun ishga tushirishni boshla degani kabi:

```
<?php

if (!empty($_POST)) {
    $name = $_POST['name'];
    $age = $_POST['age'];
    $salary = $_POST['salary'];

    $query = "INSERT INTO workers SET name=
        '$name', age='$age', salary='$salary'";
    mysqli_query($link,
        $query) or die(mysqli_error($link));
}

?>
```

Keling formamiz va PHP kodimizni birlashtiramiz:

```
<?php

if (!empty($_POST)) {

    $name = $_POST['name'];
    $age = $_POST['age'];
    $salary = $_POST['salary'];

    $query = "INSERT INTO workers SET name=
        '$name', age='$age', salary='$salary'";
    mysqli_query($link,
        $query) or die(mysqli_error($link));

}

?>

<form action="" method="POST">

<input name="name">

<input name="age">

<input name="salary">

<input type="submit"
    value="добавить работника">

</form>
```

PHP kodni formaning pastiga joylayman, ammo skriptimiz 2 marta bajarilganligi bois php kodimizni faylning istalgan yeriga qo'yishimiz mumkin.

Odatda, ma'lum muammoni yechadigan kod alohida faylda joylanadi yoki HTML fayl ichida ham yoziladi. Birinchi variantni sinab ko'rdik – php kodni alohida faylga joylab va

uni ishlashini tekshirib ko'rib(kodimizda baza bilan ulanish ko'rsatilmagan, qisqa bo'lishi uchun).

Keling endi xodimni yangi qo'shish uchun mo'ljallangan va xodimlarni ro'yhatini chiqaradigan jadvalimizni birlashtiramiz. Jadvalni chiqaradigan va xodimni o'chiradigan kodimizni esga olamiz:

```
<table>
<tr>
    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
    <th>delete</th>
</tr>
<?php
    // Ma'lumotni olganimizga qadar ishchini o'chirib
    bo'lamiz:
    if (isset($_GET['del'])) {
        $del = $_GET['del'];
        $query = "DELETE
                    FROM workers WHERE id=$del";
        mysqli_query($link,
                    $query) or die(mysqli_error($link));
    }
    // Barcha ishchilarни olamiz:
    $query = "SELECT * FROM workers";
```

```
$result = mysqli_query(
    $link, $query) or die(mysqli_error($link));
for ($data = []; $row = mysqli_fetch_assoc(
    $result); $data[] = $row);

// Ekranga chiqaramiz:
$result = '';
foreach ($data as $elem) {
    $result .= '<tr>';
    $result .= '<td>' . $elem['id'] . '</td>';
    $result .= '<td>' . $elem['name'] . '</td>';
    $result .= '<td>' . $elem['age'] . '</td>';
    $result .= '<td>' .
        $elem['salary'] . '</td>';
    $result .= '<td>' .
        $elem['salary'] . '</td>';
    $result .= '<td><a href="?del=' .
        $elem['id'] . '">o`chirish</a></td>';
    $result .= '</tr>';
}
echo $result;
?>
</table>
```

Endi xodim qo'shish uchun mo'ljallangan formamizni yuqoridagi kodimiz bilan bog'laymiz. Formamizni jadvalning tagiga joylaymiz:

```
<table>
<tr>
    <th>id</th>
    <th>name</th>
    <th>age</th>
    <th>salary</th>
    <th>delete</th>
</tr>
<?php
if (!empty($_POST)) {
    $name = $_POST['name'];
    $age = $_POST['age'];
    $salary = $_POST['salary'];

    $query = "INSERT INTO workers SET name=
        '$name', age='$age', salary='$salary'";
    mysqli_query($link,
        $query) or die(mysqli_error($link));
}

if (isset($_GET['del'])) {
    $del = $_GET['del'];
    $query = "DELETE
        FROM workers WHERE id=$del";
```

```
mysqli_query($link,  
    $query) or die(mysqli_error($link));  
  
}  
  
  
$query = "SELECT * FROM workers";  
$result = mysqli_query(  
    $link, $query) or die(mysqli_error($link));  
for ($data = []; $row = mysqli_fetch_assoc(  
    $result); $data[] = $row);  
  
  
$result = '';  
foreach ($data as $elem) {  
    $result .= '<tr>';  
  
  
    $result .= '<td>' . $elem['id'] . '</td>';  
    $result .= '<td>' . $elem['name'] . '</td>';  
    $result .= '<td>' . $elem['age'] . '</td>';  
    $result .= '<td>' .  
        $elem['salary'] . '</td>';  
    $result .= '<td>' .  
        $elem['salary'] . '</td>';  
    $result .= '<td><a href="?del=' .  
        $elem['id'] . '">o`chirish</a></td>';  
  
  
    $result .= '</tr>';  
}
```

```
echo $result;  
?  
</table>  
  
<form action="" method="POST">  
  
<input name="name">  
  
<input name="age">  
  
<input name="salary">  
  
<input type="submit"  
      value="xodim qo'shish">  
</form>
```

Tizimga kirish va ro'yhatdan o'tish

Avtorizatsiya va registratsiya haqida ma'lumot

Menimcha ko'pchilik internet foydalanuvchilari ozmi-ko'pmi avtorizatsiya va registratsiya haqida tushunchaga ega.

Avtorizatsiya – bu sayt foydalanuvchisini aniqlash(identifikatsiya) jarayoni. Buning uchun odatda foydalanuvchi tizimga o'zinig parol va loginini aytadi. Albatta u foydalaувчи birinchi o'sha saytdan registratsiyadan o'tishi kerak: o'zi uchun login(ya'ni saytdagi unikal ismi), va parol yozadi.

Odatda **login** – bu saytga tashrif buyuruvchilarning barchasiga ko'rinish turadigan ochiq ma'lumot. **Parol** esa – saytdagi akkauntiga kirish, va o'zining shaxsiy ma'lumotlarini o'qish uchun ishlatalidiga yopiq ma'lumot hisoblanadi. Ko'pincha bunay narsalar *profayl* deb ham ataladi. **Profayl** – foydalanuvchi haqidagi ma'lumotlarni o'qiy olish imkonи

bo'lgan alohida sahifa hisoblanadi. Ko'pincha profayl barcha uchun faqat o'qish uchun mavjud bo'ladi, lekin o'sha sahifaning egasi uni tahrirlay olishi va boshqa maxsus amallarni bajarolishi mumkin.

MB orqali sodda avtorizatsiya qilish

Keling, registratsiyasiz sodda avtorizatsiyani qilib ko'ramiz. Foydalanuvchini ro'yhatga olishni o'rniliga biz u foydalanuvchilarni oldindan ma'lumotlar bazasiga kirgizib qo'yamiz. PhpMyAdmin yordamida **login** va **password** maydonlari mavjud bo'lgan **users** jadvalini yarating. Endi, keling 2ta foydalanuvchini jadvalimizga qo'lda qo'shamiz: birinchi **user** loginli va **12345** parolli va ikkinchisi **admin** loginli va **123** parolli.

Buni amalga oshirish uchun avval login va parol yozish mumkin bo'lgan formani yaratish kerak. Keyin kiritilgan forma ma'lumotini bazada mavjdumi yoki yo'qligini tekshiradigan PHP kod yozamiz. Agar mavjud bo'lsa, unda uni avtorizatsiyadan o'tkazamiz- yo'q bo'lsa login yoki parol xato ekanligi haqida xabar beramiz.

Hamma yozgan kodingizni login.php fayliga joylang. Quyidagi kodni yozamiz(*topshiriqlarni yechishda bu kodni nusxalab olmang – uni mustaqil yozishni o'rganing, aks holda o'rganolmaysiz!*):

```
<?php  
/* dastavval bazaga ulanish kerak,  
 qisqa bo'lishlik uchun yozmadim  
  
Agar formamiz jo'natilsa...  
 181
```

```
*/  
  
if (!empty($_POST[  
    'password'])) and !empty($_POST['login'])) {  
  
    /*  
     * Login va parollarni o'zgaruvchilarga  
     * yozib olamiz, qulaylik uchun  
     */  
  
    $login = $_POST['login'];  
    $password = $_POST['password'];  
  
    // SQL so'rov yaratamiz va bajaramiz:  
    $query = "SELECT * FROM users WHERE login=  
        '$login' AND password='$password';  
    $result = mysqli_query($link, $query);  
  
    // Natijani shakllantiramiz  
    $user = mysqli_fetch_assoc($result);  
  
    if (!empty($user)) {  
        /* Foydalanuvchi avtorizatsiyadan o'tdi  
         * qanaqadir maxsus sahifa uchun ruxsat beriladi  
         * yoki h.z  
        */  
    } else {  
        /*  
         * Foydalanuvchi ma'lumotni noto'g'ri kiritdi  
        */  
    }  
}
```

```
Xatolik haqida xabar beramiz yoki boshqa  
*/  
}  
}  
?>  
<form action="" method="POST">  
<input name="login">  
<input name="password" type="password">  
<input type="submit" value="Jo'natish">  
</form>
```

Topshiriq 1:

Yuqoridagi avtorizatsiyani o'zingiz qayta yozing. Shunday qilingki, foydalanuvchi avtorizatsiyadan o'tsa bu haqidagi xabar unga ko'rsatilsin, aks holda o'tolmasa sababi haqida chiqarsin

Topshiriq 2:

Kodni shunday o'zgartiringki, foydalanuvchi avtorizatsiyadan muvaffaqiyatli o'tgandan keyin login va parol yoziladigan forma ekranda ko'rsatilmasin.

Topshiriq 3:

Agar avtorizatsiya muvaffaqiyatli o'tsa, unda foydalanuvchini index.php sahifaga yo'naltiring.

Avtorizatsiyaga sessiya qo'shish

Bizning avtorizatsiyamiz shunday ishlashi kerak: saytdan avtorizatsiyadan o'tmoqchi bo'lgan foydalanuvchi, **login.php** sahifasiga kiradi, to'g'ri login va parollarini kiritib keyin

avtorizatsiyadan o'tganlar uchun mo'ljallangan maxsus sahifaga kiradi.

Boshqa sahifa ushbu foydalanuvchini avtorizatsiyadan o'tganligini bilishi uchun biz buni sessiyada qayd qilib qo'yishimiz darkor.

Hozircha bizning avtorizatsiya unchalik ishlamaydi, ya'ni sessiyani hali o'rnatganimiz yo'q. Shuning uchun boshqa sahifaga o'tganda foydalanuvchini avtorizatsiyadan o'tgan yoki yo'qliginni aniqlab bo'lmaydi.

Avtorizatsiyadan o'tganligi haqidagi qaydni **\$_SESSION['auth']** sessiya o'zgaruvchisida saqlaymiz – agar **true** yozilgan bo'lsa, unda foydalanuvchi haqiqatan ham avtorizatsiyadan o'tgan, aksincha **null** bo'lsa avtorizatsiyadan o'tmaganligini ifodalaydi.

Foydalanuvchi avtorizatsiyagunga qadar **\$_SESSION['auth']** boshida null qiymatga ega bo'ladi(chunki **\$_SESSION['auth']** hali aniqlanmagan bo'ladi).

Endi keling, kodimizga tegishli o'zgartirishlarni kiritamiz:

```
<?php  
...  
// Agar avtorizatsiya formasi yuborilgan bo'lsa...  
if (!empty($_POST[  
    'password']) and !empty($_POST['login'])) {  
  
    $login = $_POST['login'];  
    $password = $_POST['password'];  
  
    $query = "SELECT * FROM users WHERE login="
```

```
'$login' AND password='$password"';  
  
$result = mysqli_query($link, $query);  
  
$user = mysqli_fetch_assoc($result);  
  
  
if (!empty($user)) {  
  
    // Foydalanuvchi avtorizatsiyadan o'tdi,  
    // biz buni sessiyaga belgilab qo'yamiz:  
  
    $_SESSION['auth'] = true;  
  
} else {  
  
    // Foydalanuvchi noto'g'ri ma'lumot kiritdi  
    // qanaqadir hodisa bajariladi  
  
}  
  
}  
  
...  
  
?>
```

Shunday qilib, foydalanuvchi to'g'ri avtorizatsiyadan o'tgunga qadar, **\$_SESSION['auth']** belgilanmagan bo'ladi, ya'ni **null** qiymatga ega bo'ladi, agar foydalanuvchi avtorizatsiyadan o'tsa, **\$_SESSION['auth'] true** qiymatga ega bo'ladi.

Endi har qanday sahifaga kirganda, foydalanuvchini avtorizatsiyadan o'tgan yoki yo'qligini tekshirishimiz mumkin, mana bunday:

```
<?php  
  
if (!empty($_SESSION['auth'])) {  
  
}  
  
?>
```

Eslatma!

Shunchaki **if(\$_SESSION['auth'] == true)** shaklida tekshirish xato, chunki agar **\$_SESSION['auth']** da hech nima bo'lmaydigan bo'lsa unda PHP xatolik chiqaradi. Bu shartdagi ma'no: agar sessiyaning auth qiymati bo'sh bo'lmasa...

Empty funksiyasi quyidagilarni bo'sh deb hisoblaydi:

- **""** (bo'sh satr)
- **0** (butun son)
- **0.0** (kasr son)
- **"0"** (satr)
- **NULL**
- **FALSE**
- **array()** (bo'sh massiv)

Shuningdek, siz avtorizatsiyadan o'tmagan foydalanuvchi uchun yagona sahifadagi qanaqadir matnni u uchun ruxsat etmasligingiz mumkin. Buning uchun shunchaki IF shartimiz ichiga matn yozamiz:

```
<?php

if (!empty($_SESSION['auth'])) {

?>

<!DOCTYPE html>

<html>

    <head>

        </head>

    <body>

        <p>Faqat avtorizatsiyadan o'tgan foydalanuvchilar uchun</p>

    </body>

</html>

<?php

} else {
```

```
?>  
<p>Iltimos avtorizatsiyadan o'ting</p>  
<?php  
}  
?>
```

Topshiriq 1:

login.php sahifamizdan tashqari saytimizda yana boshqa, **1.php**, **2.php** va **3.php** sahifalar bor deylik. Shunday qilingki, aynan 1,2 va 3 sahifalarga faqat avtorizatsiyadan o'tgan foydalanuvchilargina kiraolsin. Agar foydalanuvchi avtorizatsiyadan o'tmagan bo'lsa unda unga bu haqida xabar chiqaring va **login.php** sahfasi uchun havola berib qo'ying.

Topshiriq 2:

login.php sahifamizdan tashqari saytimizda yana boshqa **index.php** nomli sahifa mavjud deylik. Shunday qilingki, **index.php** sahifasining bir qismi barcha foydalanuvchilar uchun, ikkinchi qismi esa faqatgina avtorizatsiyadan o'tganlar uchun ochiq qilib qo'yilsin.

Topshiriq 3:

Kodni shunday o'zgartiringki, foydalanuvchi avtorizatsiyadan muvaffaqiyatli o'tgandan so'ng sessiyaga uning loginini ham yozib qo'ying. Saytning har qanday sahifasiga kirganda, avtorizatsiyadan o'tgan foydalanuvchi uchun "Siz user nom ostida saytga kirdingiz" degan xabar chiqsin, avtorizatsiyadan o'tmaganlar uchun esa "Iltimos, avtorizatsiyadan o'ting" degan va avtorizatsiyadan o'tish uchun mo'ljallangan sahifa uchun havola xabarini yozib qo'ying.

Logaut(saytdan chiqish)

Avtorizatsiyadan o'tgan foydalanuvchi saytdagi sessiyasini to'xtatish, ya'ni o'z akkauntidan chiqib ketishi ham mumkin, yoki inglizchalab **logout** qilishi mumkin.

Buning uchun alohida sahifa qilib, uni nomini **logout.php** deb nomlashimiz va avtorizatsiyadan o'tganligi to'g'risidagi qaydni o'chirish orqali amalga oshirish mumkin.

```
<?php  
  
session_start(); // albatta sessiyani birinchi boshlash  
kerak  
  
$_SESSION['auth'] = null;  
  
?>
```

Sessiyaga null qiymatini belgilashning o'rнига, **unset(\$_SESSION['auth'])** qilishimiz ham mumkin. Ular bir xil vazifa bajaradi. Lekin, unset PHP 7 da eskirgan hisoblanib va PHP 8 versiyasida o'chiriladi, uni ishlatmagan ma'qul.

Yana bir nozik tomon: sessiyani **session_destroy()** orqali o'chirishimiz ham mumkin, lekin bu funksiya sessiyadagi barcha ma'lumotlarni o'chirib yuboradi, xolbuki bizning sessiyamizda boshqa ma'lumotlar ham saqlanishi mumkin.

Topshiriq 4:

logout.php sahifasini yarating, unga kirgan foydalanuvchi saytdagi akkauntidan chiqib ketishi lozim.

Topshiriq 5:

Yuqoridaagi topshiriqnı sal o'zgartiramiz: logout.php sahifasi o'z vazifasini bajarganidan so'ng u foydalanuvchini index.php sahifasiga yo'naltirib yuborsin va foydalanuvchiga saytdan chiqqanligi haqidagi xabar ko'rinsin.

Ro'yhatdan o'tish yoki registratsiya

Endi keling bиргаликда ro'yhatdan o'tish sahifasini ham ishlab chiqamiz. Buning uchun alohida register.php sahifasini yaratib olamiz.

Bu sahifaga kirgan foydalanuvchi, o'zi xohlagan login va parolini yozishi mumkin bo'lgan formani ko'rishi kerak. Formaning jo'natish tugmasini bosganda foydalanuvchining login va parollari **INSERT** so'rovi yordamida bazaga yuklanishi kerak, taxminan shunday:

```
<?php  
...  
// Agar reg formasi yuborilsa...  
if (!empty($_POST[  
    'login']) and !empty($_POST['password'])) {  
  
    $login = $_POST['login'];  
    $password = $_POST['password'];  
  
    $query = "INSERT INTO users SET login=  
        '$login', password='$password';  
    mysqli_query($link, $query);  
}  
?  
<form action="" method="POST">  
    <input name="login">  
    <input name="password" type="password">  
    <input type="submit" value="Jo'natish">
```

```
</form>
```

Keyin foydalanuvchi registratsiyada kiritgan login va parollari asosida avtorizatsiya sahifasiga kirib, avtorizatsiyadan o'tishi mumkin.

Topshiriq 1:

Yuqorida yozilgan registratsiyani mustaqil ishlab chiqing. Yangi foydalanuvchi ro'yhatdan o'tgandan so'ng uni avtorizatsiyadan ham o'tkazing. Hammasi to'g'ri ishlayotganiga ishonch hosil qiling.

Topshiriq 2:

Yuqoridagi kodni shunday o'zgartiringki, foydalanuvchi login va parol kiritishdan tashqari, o'zining tug'ilgan sanasini va email manzilini ham kiritolsin. Ushbu ma'lumotlar bazaga saqlansin.

Ro'yhatdan o'tganlik sanasi

Keling endi foydalanuvchi ro'yhatdan o'tganidan so'ng uning ro'yhatdan qachon o'tganligi to'g'risidagi sanani ham bazaga saqlaymiz. Bu ma'lumot albatta kerak bo'ladi.

Buning uchun MB ga biz yana bitta sana uchun maydon yaratib uni **registration_date** deb nomlaymiz. SQL so'rovga o'zgartirish kiritamiz:

```
<?php  
  
$date = date('Y-m-d'); //  
//joriy sanani olamiz  
  
$query = "INSERT INTO users SET  
login='$login', password='$password',  
'registration_date'='$date'" ;
```

```
...  
?>
```

Topshiriq 3:

Yuqoridagi yangi registratsiya shaklini mustaqil yarating

Registratsiyadan o'tgan zahoti avtorizatsiya qilish

Hozir bizning registratsiyamiz bunday tarzda qilingan, foydalanuvchi birinchi martasida registratsiyadan o'tish uchun o'zi xohlagan login va parollarini yozadi, keyin esa avtorizatsiyadan o'tmoqchi bo'lsa o'sha sahifaga kirib qaytadan login va parollarini yozadi.

Bu unchalik qulay emas va foydalanuvchini kuttirib qo'yadi. Yaxshisi registratsiya muvaffaqiyatli amalga oshganidan so'ng foydalanuvchini darrov avtorizatsiyadan o'tkazish kerak.

Buni qilish qiyin emas – muvaffaqiyatli registratsiyadan so'ng sessiyaga avtorizatsiyadan o'tganligi haqida belgi qo'yib qo'yamiz, bo'ldi:

```
<?php  
...  
if (!empty($_POST[  
    'login']) and !empty($_POST['password'])) {  
  
    $login = $_POST['login'];  
    $password = $_POST['password'];  
  
    $query = "INSERT INTO users SET login=  
        '$login', password='$password'";
```

```
mysqli_query($link, $query);

$_SESSION['auth'] = true;

}

?>

<form action="" method="POST">

<input name="login">

<input name="password" type="password">

<input type="submit" value="Jo'natish">

</form>
```

Sessiyaga foydalanuvchi idsini qo'shamiz

Avtorizatsiyadan o'tganligi to'g'risida belgi qo'shib qo'yishdan tashqari sessiyamizga foydalanuvchining idsini ham qo'shishni ham xohlaymiz. Bu holatda biz bu idni mysqli_insert_id funksiyasi yordamida olishimiz mumkin. Bu funksiya ushbu script orqali qo'shilgan so'nggi ma'lumotni id raqamini oladi va bu bizga kerakli bo'lgani:

```
<?php

...

if (!empty($_POST[
    'login']) and !empty($_POST['password'])) {
```

переменные для удобства работы:

```
$login = $_POST['login'];
$password = $_POST['password'];
```

```
$query = "INSERT INTO users SET login=
```

```
'$login', password='$password"';  
mysqli_query($link, $query);  
  
$_SESSION['auth'] = true;  
  
$_SESSION['id'] = $id;  
}  
?  
<form action="" method="POST">  
<input name="login">  
<input name="password" type="password">  
<input type="submit" value="Jo'natish">  
</form>
```

Topshiriq 5:

Mustaqil ravishda registratsiyadan muvaffaqiyatli o'tgan foydalanuvchining id raqamini sessiyaga yozib qo'ying.

Parolni yashirish

Odatda parol kiritiladigan maydon **password** tipida bo'ladi. Uning vazifasi foydalanuvchi o'zining maxfiy parolini registratsiyadan o'tishda boshqa birovning bu haqida bilib olmasligi tufayli uni yulduzcha shaklda ko'rsatib uni yashirishdan iborat.

Parolni yashirish albatta yaxshi, lekin bitta muammo bor – foydalanuvchi nima kiritayotganin bilmaydi. Registratsiyadan o'tayotganda foydalanuvchi o'zi bilmagan holda xatoga yo'l qo'yib, noto'g'ri parol kiritib qo'yishi ehtimoldan holi emas.

Shunning uchun bizda bir standart mavjud: foydalanuvchi parolni bir marta emas, ikki marta kiritadi. Birinchisida foydalanuvchi parolni kirlitsa, ikkinchi maydonda esa avvalgi kiritgan parolini to'g'rilini tasdiqlaydi:

```
<form action="" method="POST">

<input name="login">

<input name="password" type="password">

<input name="password" type="confirm">

<input type="submit" value="Jo'natish">

</form>
```

Bizning ishimiz kiritilgan parol ikkinchi parol bilan mosligini tekshirishdan iborat:

```
<?php

...

if (!empty(
    $_POST['login']) and
    !empty($_POST[
        'password']) and
    !empty($_POST['confirm'])) {
    if ($_POST['password'] == $_POST['confirm']) {
        // birinchi va ikkinchi parol to'g'rilingin
        tasdiqlandi - ruxsat etildi
    } else {
        // tasdiqlanmadi va xatolik chiqarildi
    }
}
```

```
<form action="" method="POST">  
  
<input name="login">  
  
<input name="password" type="password">  
  
<input name="password" type="confirm">  
  
<input type="submit" value="Jo'natish">  
  
</form>
```

Topshiriq 6:

Kodingizni shunday o'zgartiringki, formani yuborganingizda parol va uning ikkinchi tasdiq parolini mosligi tekshirilsin, agar ular mos kelsa, registratsiyani davom ettiring aks holda muammoni xabarini chiqaring.

Loginni bandligini tekshirish

Hozir bizning registratsiyamizda bitta muammo qoldi – saytimizdan yangi foydalanuvchi, saytimizdan avval masalan sobirjonovs logini ostida ro'yhatdan o'tgan foydalanuvchining logini bilan ham ro'yhatdan o'tishi mumkin. Lekin, yuqorida aytgan gapimiz: **login foydalanuvchining saytdagi unikal ismi**, ga mos kelmayapti. Shuning uchun bu muammoni yechishda biz **INSERT** so'rovidan oldin **SELECT** so'rovini amalga oshiramiz. Bu so'rov orqali kiritilgan login bazada mavjudmi yoki yo'q shuni tekshiramiz. Agar band bo'lmasa foydalanuvchini ro'yhatga olamiz aks holda login bandligi to'g'risidagi xatolikni ko'rsatamiz:

```
<?php  
  
...  
  
if (!empty($_POST[  
    'login'])) and !empty($_POST['password'])) {
```

```
$login = $_POST['login'];
$password = $_POST['password'];

// Shunday loginli foydalanuvchini bazadan olishga
urunib ko'ramiz:

$query = "SELECT *
FROM users WHERE login='$login'";

$user = mysqli_fetch_as
soc(mysqli_query($link, $query));

// Agar bunday loginli foydalanuvchi bo'lmasa:
if (empty($user)) {
    // Формируем и отсылаем SQL запрос:
    $query = "INSERT INTO users SET login=
'$login', password='$password'";
    mysqli_query($link, $query);

    $_SESSION['auth'] = true;
}

$id = mysqli_insert_id($link);
$_SESSION['id'] = $id;

} else {
    // Login bandligi haqida xabar beramiz
}

?>
```

```
<form action="" method="POST">  
  
<input name="login">  
  
<input name="password" type="password">  
  
<input type="submit" value="Jo'natish">  
  
</form>
```

Keling endi parolni mosligini tekshirishni ham qo'shamiz:

```
<?php  
  
...  
  
if (!empty(  
    $_POST['login']) and  
    !empty($_POST [  
        'password']) and  
    !empty($_POST ['confirm'])) {  
  
    if ($_POST ['password'] == $_POST ['confirm']) {  
  
        в переменные для удобства работы:  
  
        $login = $_POST ['login'];  
        $password = $_POST ['password'];  
  
        $query = "SELECT *  
                  FROM users WHERE login='$login'";  
        $user = mysqli_fetch_a  
                ssoc(mysqli_query($link, $query));  
  
        if (empty($user)) {
```

```
$query = "INSERT INTO users SET login=
    '$login', password='$password'";
mysqli_query($link, $query);

$_SESSION['auth'] = true;

$id = mysqli_insert_id($link);
$_SESSION['id'] = $id;
} else {
    // login bandligi to'g'risdagi xatolik
}
} else {
    // parol va ikkinchi tasdiq parol mos emasligi
    haqidagi xatolik
}
}

?>

<form action="" method="POST">
<input name="login">
<input name="password" type="password">
<input name="password" type="confirm">
<input type="submit" value="Jo'natish">
</form>
```

Topshiriq 7:

Yuqoridagi kodni shunday o'zgartiringki, ro'yhatga o'tishda urungan foydalanuvchining logini band yoki bandmasligi

tekshirilsin agar u band bo'lsa unda bu haqida xabar chiqarilsin va boshqa login tanlashligi aytilsin.

Ma'lumotlarni tekshirish

Hozirda biz foydalanuvchi login/parole uchun hech qanday cheklovlar mavjud emas, bu esa noto'g'ri. Misol uchun, hozir foydalanuvchi tasodifan yoki ataylab hech qanday login yozmasdan ya'ni 1ta belgidan iborat login yoki parol yozib ro'yhatdan o'tishi mumkin. Bunday parol juda ham sodda va xavfsizligi o'ta past.

Shuni yodda tutingki, agar formaning bitta maydonida xatolik yuz bersa, butun formaga kiritilgan ma'lumotni o'chirish shart emas, bu foydalanuvchiga noqulaylik tug'diradi: u ma'lumotlarni kiritdi-kiritdi-da, keyin yubordi va hammasi yo'qoldi, hatto bitta belgida xato qilgan bo'lsa ham.

Shu tufayli foydalanuvchi saytdan ro'yhatdan o'tmasligi mumkin, bu esa bizning saytimiz foydalanuvchilarini oshishiga halaqit qiladi.

Quyidagi barcha topshiriqlarni yeching, forma yuborilgandan keyin agar birorta formada xatolik yuz bersa ham lekin formada kiritilgan ma'lumotlar saqlab qolinsin:

Topshiriq 1:

Shunday kod yozingki, foydalanuvchi bo'sh login va parol kirtska ro'yhatga olinmasin

Topshiriq 2:

Shunday kodni o'zgartiringki, login faqat lotin harflari va sonlaridan iborat bo'lsin. Bunday bo'lмаган taqdirda, forma tagida bu haqida xabar chiqarilsin.

Topshiriq 3:

Kodni shunday o'zgartiringki, login 4 tadan 10 tagacha bo'lган belgilardan iborat bo'lsin. Bunday bo'lмаган taqdirda, bu haqida forma tagida xabar chiqarilsin.

Topshiriq 4:

Shunday kodni o'zgartiringki, parol 6 tadan 12 tagacha bo'lган belgilardan iborat bo'lsin. Bunday bo'lмаган taqdirda, bu haqida forma tagida xabar chiqarilsin.

Topshiriq 5:

Foydalanuvchini ro'yhatga olishda uning email manzilini ham so'rang va uni bazaga saqlang. Email manzil to'g'ri kiritilganligini tekshiring, agar bunday bo'lmasa formadagi tegishli maydonning tagiga bu haqida xabar bering.

Topshiriq 6:

Kodni shunday tuzingki, agar parol yoki login xato kiritilsa, bu haqida tegishli maydonlar ostida xabar chiqarilsin.

Topshiriq 7:

Foydalanuvchini ro'yhatga olishda uning tug'ilgan sanasini ham kun.oy.yil formatda so'rang. Uni bazaga saqlang. Sananing to'g'riliгини tegishli formatda tekshiring.

Topshiriq 8:

Foydalanuvchini ro'yhatga olishda yashash joyini ham so'rang. Qaysidir hududni select ro'yhatidan tanlasin.

Parolni heshlash

Parolni ochiq ko'rinishda bazada saqlash – bu noto'g'ri. Yovuz hakkerlar bazangizga kirib olgan taqdirda, parollarni

bemalol o'qiy olishi va foydalanuvchilaringizning ma'lumotlariga egalik qilishi mumkin.

Shuning uchun loginni ochiq ko'rinishda saqlash, parolni esa **md5** funksiyasi yordamida heshlab saqlash kerak. **Md5** bilan heshlangan parolni qayta tiklab bo'lmaydi. Keling, '12345' satrni heshlab ko'ramiz:

```
<?php  
echo md5('12345');  
//natija '827ccb0eea8a706c4c34a16891f84e7b'  
?>
```

Hozir biz registratsiya va avtorizatsiyamizni qayta ko'rib, ishlab chiqishimiz kerak. Boshlanishiga foydalanuvchilar jadvali ichidagi ma'lumotlarni o'chirishni tavsiya qilaman sababi u yerda hozir parol ochiq ko'rinishda saqlanayapti. Bizga esa uni heshlab saqlash kerak. Endi registratsiyamizni yangi foydalanuvchini bazaga qo'shishda parolni emas balki uni heshini saqlaydigan qilib o'zgartiramiz:

```
<?php  
...  
$login = $_POST['login'];  
$password = md5($_POST['password']); //parolni heshlaymiz  
  
$query = "INSERT INTO users SET login=  
'$login', password='$password';  
...  
?>
```

Avtorizatsiyaga ham shunga o'xshash o'zgartirish kiritamiz:

```
<?php  
...  
$login = $_POST['login'];  
$password = md5($_POST['password']);  
  
$query = "SELECT * FROM users WHERE login=  
        '$login' AND password='$password'";  
...  
?>
```

Topshiriq 1:

Ro'yhatdan o'tishni hesh asosida o'zgartiring. Bir nechta yangi foydalanuvchilar qo'shing va ularning parollari bazaga heshlanganligiga ishonch hosil qiling.

Topshiriq 2:

Avtorizatsiyanni hesh asosida o'zgartiring. Avvalroq ro'yhatdan o'tgan foydalanuvchilarning login va parollari bilan avtorizatsiyadan o'tishni sinab ko'ring.

Registratsiyaga “tuz” qo'shamiz

Md5 bilan heshlash bilasizki – bu orqaga qaytarib bo'lmas jarayon va parolni o'g'irlamoqchi bo'lgan hakker ham aniq parolni bilolmaydi.

Lekin aslida bu unchalik ham to'g'ri emas – hozirgi vaqtida yovuz niyatli hakerlar unchalik ko'p bo'lmanan mashhur heshlar kutubxonasini tuzishgan va har qanday omi(e'tibor bering: albatta haker emas) heshni gugl qilish orqali parolni bilib olishi ham mumkin.

Gap yetarlicha sodda, mashhur parollar haqida ketyapti. Masalan, Googlega `827ccb0eea8a706c4c34a16891f84e7b` heshni yozib ko'ring va qidiruv natijasi sifatida 12345 parolini ko'rasiz. Yetarlicha murakkab parollarning heshini bunday tarzda topib bo'lmaydi. Aytishingiz mumkin, muammo nimada – keling unda registratsiyadan o'tishda murakkab parol yozib o'tamiz deb. Lekin, bitta muammo – ko'pchilik foydalanuvchilar o'zining shaxsiy ma'lumotlarini xavfsizligi haqida qayg'urishmaydi va yetarlicha sodda parollardan foydalanishadi.

Registratsiyadan o'tishda ancha uzunlikdagi parolni foydalanuvchidan uni cheklov orqali so'rashimiz mumkin, masalan eng kam belgilar soni 6ta yoki 8ta bo'lsin deb, lekin ular oddiygina 123456 yoki 12345678 ko'rinishda parolni yozishi mumkin. Parolni murakkabligi bo'yicha aqli tekshirish algoritmini o'ylab chiqishimiz mumkin lekin boshqa yechim bor.

Ushbu yechimning mohiyati shunday iborat: parolni tuzlash kerak. **Tuz(ing. Salt)** – bu ro'yhatdan o'tish mobaynida parolga qo'shiladigan maxsus tasodifiy satr bo'lib, hesh oddiy kiritilgan paroldan yasalmasdan balki tuz+parol satri, ya'ni tuzlangan paroldan yasaladi.

Ro'yhatdan o'tishni taxminan shu ko'rinishda qilasiz:

```
<?php  
  
$salt = '1sJg3hfdf'; // tuz - murakkab tasodifiy satr  
  
$password = md5($salt . $_POST['password']); // parolni  
tuzlangan heshga aylantiramiz  
  
?>
```

Bunda albatta tuz har bir foydalanuvchi uchun har xil bo'lishi kerak, uni registratsiya mobaynida tasodifiy ravishda generatsiya qilamiz. Mana tayyor funksiya, aynan shu vazifaga mo'ljallangan:

```
<?php

function generateSalt()
{
    $salt = '';
    $saltLength = 8; // tuzning uzunligi

    for($i = 0; $i < $saltLength; $i++) {
        $salt .= chr(mt_rand(33,
126)); // ASCII jadvalidagi belgi
    }

    return $salt;
}

?>
```

Yana bir bor takrorlayman, bu registratsiya paytidagi qilinadigan ishlar – bazaga oddiy parolni heshini emas balki tuzlangan parolni heshini saqlaymiz.

Bu hali hammasi emas: foydalanuvchilar jadvalida login va password maydonlaridan tashqari yana bitta salt nomli maydon qo'shamiz. Bu maydonda har bir foydalanuvchining tuzi ochiq ko'rinishda saqlanadi. Ya'ni parol yopiq ko'rinishda – heshlangan shaklda saqlansa, tuz esa ochiq ko'rinishda saqlanadi.

Topshiriq 3:

Yuqoridagi registratsiyani tuzlangan parolli ko'rinishdagisin ishlab chiqing.

Avtorizatsiyaga tuz qo'shamiz

Endi navbat avtorizatsiyani o'zgartirishga. Bu yerdagi o'zgarishlar yanada muhimroq rol o'ynaydi. Endi login parol juftligi to'g'rilibni bitta so'rovda birdan tekshirib bo'lmaydi. Nima uchun: sababi, parolni tekshirish uchun uning tuzlangan heshini olish kerak, tuz esa bazada saqlanadi va har bir foydalanuvchi uchun unikal hisoblanadi. Avvaliga ma'lumotni login bo'yicha olamiz, tuzni o'qib, kiritilgan parolni tuzlab va uni bazadagi tuzlangan parol bilan taqqoslaymiz va faqatgina ular mos kelishsagina – foydalanuvchini avtorizatsiyadan o'tkazamiz.

Shuni yodda tutingki, login noto'g'ri kiritilishi mumkin bu holatda siz parolni tekshira olmaysiz, shunda darrov avtorizatsiya ma'lumotlari noto'g'ri ekanligi haqidagi xabarni chiqaring:

```
<?php  
...  
$login = $_POST['login'];  
  
$query = "SELECT * FROM users WHERE login=  
        '$login"'; // foydalanuvchini login bo'yicha olamiz  
$result = mysqli_query($link, $query);  
$user = mysqli_fetch_assoc($result);  
  
if (!empty($user)) {
```

```
// agar foydalanuvchi bu login bo'yicha topilsa,  
parolni tekshiramiz  
  
} else {  
  
    // topilmasa xatolik xabarini chiqaramiz  
  
}  
  
?>
```

Keling endi parolni tekshirishni ham tuzamiz:

```
<?php  
  
...  
  
$login = $_POST['login'];  
  
  
$query = "SELECT * FROM users WHERE login=  
        '$login'";  
  
$result = mysqli_query($link, $query);  
  
$user = mysqli_fetch_assoc($result);  
  
  
if (!empty($user)) {  
  
  
    $salt = $user['salt']; // bazadagi tuz  
    $hash = $user[  
        'password']; // bazadagi tuzlangan parol  
  
  
    $password = md5($salt,  
                    $_POST['password']); // foydalanuvchi tomonidan  
    kiritilgan tuzlangan parol  
  
  
    // tuzlangan heshlarni tekshirish
```

```
if ($password == $hash) {  
    // Hammasi joyida, davom etamiz  
}  
else {  
    // Parol mos kelmadi, xatolik  
}  
else {  
    // Bu foydalanuvchi ham topilmadi  
}  
?  
?
```

Muhim vaziyat: xavfsizlik nuqtayi nazaridan foydalanuvchi avtorizatsiy a ma'lumotlarini noto'g'ri kiritgan taqdirda, qaysi biri login yo parol noto'g'ri kiritilganligi haqidagi ma'lumotni chiqarmang. Shunchaki, parol yoki login noto'g'ri deb xatolik ko'rsating

Topshiriq 4:

Tuzlangan parol qo'shilgan avtorizatsiyani ishlab chiqing.
Ro'yhatdan o'ting, avtorizatsiya qiling, hammasi
ishlayotganligiga amin bo'ling.

password_hash funksiyasidan foydalanamiz

Aslida md5 funksiyasi va tuzlangan paroldan foydalanish eskirgan hisoblanadi. Biz ularni shunchaki qo'shimcha tajriba sifatida o'rgattik. Balki shu uslubda ishlagan eski loyihalardagi vaziyatlarni guvohi bo'lib qolishingiz mumkin, ana shu holatda bu sizga tushunarli bo'lmasligi uchun harakat qildik.

Tuzlangan parolni olishning zamonaviy usuli mavjud bo'lib, buning uchun **password_hash** funksiyasi ishlatiladi. Birinchi parametr sifatida u satrni, ikkinchi parametr sifatida esa shifrlash algoritmini(ular haqida keyinroq gaplashamiz), qabul qiladi va satrni tuzi bilan birga heshini qaytaradi. Misol uchun:

```
<?php  
echo password_hash(  
    '12345', PASSWORD_DEFAULT);  
?>
```

Har funksiya ishga tushgan vaqtida har xil natija olasiz – bu natijadagi satrning birinchi qismi tuz hisoblanadi, ikkinchisi esa tuzlangan parol.

Aytaylik bizda password_hash funksiyasidan olingan hesh bor va qanaqadir parol ham. U parolning heshi ekanligini tekshirish uchun **password_verify** funksiyasi ishlataladi – birinchi parametr sifatida parolni, ikkinchi parametr sifatida heshni qabul qiladi va true yoki false qaytaradi.

```
<?php  
  
$password = '12345'; // parol  
  
$hash = '$2y$10$xoYFX1mFPxBSyxaRe3iIRutxkIWhxGShz  
        EhjYUVd3qpCUKfJE1k7a'; // hesh  
  
  
if (password_verify($password, $hash)) {  
    // bu parolning heshi  
}  
else {  
    // bu parolning heshi emas  
}  
?>
```

Bu bizga amaliyotda nima beradi: biz ma'lumotlar bazasida alohida parolning tuzini saqlash uchun maydon hosil qilmaymiz, PHP biz uchun buni hammasini o'zi amalgaloshiradi!

Ya'ni ma'lumotlar bazasidagi password maydonini o'zida ham tuzlangan parol ham uning tuzini saqlanadi:

```
<?php

. . .

$login = $_POST['login'];

$query = "SELECT * FROM users WHERE login=
'$login';

$result = mysqli_query($link, $query);
$user = mysqli_fetch_assoc($result);

if (!empty($user)) {
    $hash = $user[
        'password'];

    if (password_verify(
        $_POST['password'], $hash) ) {
        // hammasi OK.

    } else {
        // xatolik
    }
} else {
    // xatolik
}
?>
```

Topshiriq 5:

Avtorizatsiya va registratsiya qismlarni yangi o'rgangan funksiyangiz asosida qayta ishlab chiqing.

Keling endi **password_hash** funksiyasining ikkinchi parametrini ko'rib chiqamiz. Bunda biz shifrlash algoritmini ko'rsatamiz. Bizni vaziyatimizdagi standart algoritm bu **BCrypt**(md5 ni analogi lekin undan ishonchli), va bu bizga 60ta belgidan iborat bo'lgan tuz+hesh satrini qaytaradi.

Bu ma'lumotlar bazasidagi **password** maydonini o'lchamini **60taga** o'rnatish kerakligini va bunday bo'lmagan holda funksiya ishlamasligini anglatadi.

BCrypt algoritmi **password_hash** funksiyasining ikkinchi parametrida **PASSWORD_BCRYPT** deb yozish orqali ochiq ko'rsatilishi mumkin.

Agar u yerga **PASSWORD_DEFAULT** deb yozadigan bo'lsangiz ham PHP uni standart algoritm **BCRYPT** deb qabul qiladi. Kelajakda **BCrypt** ham eskirishi mumkin, **md5** kabi boshqa boshqa kuchliroq algoritmga almashadi.

Profil va shaxsiy kabinet

Endi foydalanuvchi o'z profilini ko'rish imkoniyatini yaratamiz. Profil bilan foydalanuvchi registratsiyada kiritgan ma'lumotlarini tahrirlashi, o'chirishi yoki h.k ishlarni amalga oshirishi mumkin.

Shunday qilaylik, har qanday foydalanuvchi o'zini profilini osonlik bilan ko'rolsin. Buning uchun biz ko'rmoqchi bo'lган foydalanuvchining id raqami GET parametrda ko'rsatiladigan profile.php sahifasini yaratamiz, mana bunday: **profile.php?id=0** – Oni ornida har qanday son bo'lishi mumkin, ya'ni foydalanuvchining id raqami.

Fikrimcha, profil sahifasida foydalanuvchi o'zi haqida kiritgan barcha ma'lumotni ham ko'rsatmasligimizni tushundingiz. Misol uchun, parolni u yerda ko'rsatib bo'lmaydi. Undan tashqari, balkim email manzilni ham ko'rsatish oshiqcha

bo'lishi mumkin, shunday holat bo'lishi mumkin spammer foydalanuvchi email manzillarini parser qilib, spam xabarlarni tarqatishi mumkin.

Topshiriq 1:

Registratsiyada foydalanuvchidan login, parol, ism, sharif, familiya, tug'.sanasini kiritishini so'rang. Uni foydalanuvchi profilida paroldan tashqari barcha ma'lumotlarni ko'rsating.

Topshiriq 2:

users.php sahifasini yarating, har qanday sahifaga tashrif buyurgan foydalanuvchi, saytimizning foydalanuvchilari ro'yhatini havola ko'rishida ko'rsin. Har bir havola tegishli profilga yo'naltirilgan bo'lsin.

Shaxsiy kabinet

Shaxsiy kabinetga foydalanuvchi o'z ma'lumotlarini tahrirlaydigan joy deb qaralishi kerak.

personalArea.php sahifasini qilamiz, unga kirgan foydalanuvchi o'z profil ma'lumotlarini tahrirlashi uchun kerak bo'lgan formani ko'radi(login va paroldan tashqari).

E'tibor bering, biz har bir foydalanuvchi o'z profil ma'lumotlarini personalArea.php sahifasida ko'rishi uchun, foydalanuvchi id raqamini GET parametri asosida yubormaymiz.

Buni qilish uchun biz: avtorizatsiyadan o'tgan foydalanuvchining id raqamini sessiyaga yozib olamiz, mana bunday:

```
<?php
```

```
...
```

```
$login = $_POST['login'];

$query = "SELECT * FROM users WHERE login=
'$login';

$result = mysqli_query($link, $query);
$user = mysqli_fetch_assoc($result);

if (!empty($user)) {
    $hash = $user[
        'password'];

    // пароль из базы введенному паролю
    if (password_verify(
        $_POST['password'], $hash)) {
        $_SESSION['auth'] = true;
        $_SESSION['id'] = $user['id'];
    } else {
        // пароль не совпал
    }
} else {
    // пользователь не найден
}
?>
```

Keyin, **personalArea.php** sahifasiga kirishi bilanoq o'sha id ga ega bo'lgan foydalanuvchining tegishli ma'lumotlarini olish uchun **SELECT** so'rovini yaratamiz:

```
<?php  
  
$id = $_SESSION['id'];  
  
$query = "SELECT *  
          FROM users WHERE login='$id';  
  
?>
```

Formani yuborish tugmasiga bosganda biz **UPDATE** so'rovi ichida **WHERE** shartida o'sha id ni ko'rsatamiz va **UPDATE** so'rovini ishga tushiramiz.

Topshiriq 4:

Yozilgan shaxsiy kabinetni yarating.

Parolni almashtirish

Yuqorida eslatib o'tganimdek, shaxsiy kabinetda parolni shunchaki almashtirish mumkin emas.

Gap shundaki, foydalanuvchi kompyuterini nazoratsiz qoldirib ketishi mumkin(misol uchun, ofisda). Bunday holatda agar shunchaki parolni almashtirishga ruxsat berib qo'yilgan bo'lsa, unda yovuz niyatli kimsa parolni boshqasiga almashtirib o'zining g'arazli niyatlarini amalga oshirishi mumkin. Bu esa yomon.

Shuning uchun, **changePassword.php** sahifasini yaratib olamiz, unga kirgan foydalanuvchi 2ta inputli forma ko'rabi birinchisida eski parolini kiritadi, ikkinchisida esa yangisini.

Yuborish tugmasi bosilganda biz quyidagilarni qilishimiz kerak:

```
<?php  
  
...  
  
$id = $_SESSION['id'];
```

```
$query = "SELECT * FROM users WHERE id='$id'";

$result = mysqli_query($link, $query);
$user = mysqli_fetch_assoc($result);

$hash = $user['password'];

if (password_verify(
    $_POST['old_password'], $hash)) {

    $newPasswordHash = password_hash($_POST[
        'new_password'], PASSWORD_DEFAULT);

    $query = "UPDATE users SET password=
        '$newPasswordHash' WHERE id='$id'";

    mysqli_query($link, $query);
} else {
    //eski parol notog'ri
}

?>
```

Albatta bu sahifaga kirishdan oldin foydalanuvchi avtorizatsiyadan o'tganligini tekshirish shart.

Topshiriq 5:

Yuqorida yozilgan parolni almashtirish sahifasini ishlab chiqing.

Parolni tasdiqlash

Yana bir nozik tomoni bor: eski va yangi parollar password tipli inputlarga kiritiladi. Bu esa foydalanuvchi yangi paroliga qanday parol yozganligini bilmasligini anglatadi. Endi yangi parolni ham tasdiqlashni qo'shish kerak.

Topshiriq 6:

Yuqorida yozilgan bo'yicha kodingizga o'zgartirish kriting.

Akkauntni o'chirish

Endi keling, foydalanuvchiga o'z akkauntini o'chirish imkoniyatini beramiz. Buning uchun alohida PHP sahifa yaratamiz. Unga kirgan foydalanuvchi parol kiritiladigan formani ko'radi. Parol to'g'ri kiritilgan taqdirdagina akkaunt o'chiriladi. Gap shundayki, akkauntni o'chirish – bu muhim ishlardan biri, bunday turdag'i barcha operatsiyalarda parolni so'rash shart. Sababi bu haqiqatan ham akkaunt egasimi yoki yovuz niyatli bo'lган kimsa.

Kodning kalit qismini ko'rsataman:

```
<?php  
...  
$id = $_SESSION['id'];  
$query = "SELECT * FROM users WHERE id='$id"';  
  
$result = mysqli_query($link, $query);  
$user = mysqli_fetch_assoc($result);  
  
$hash = $user['password'];
```

```
хеша из базы введенному старому паролю

if (password_verify(
    $_POST['password'] , $hash)) {

    $query = "DELETE
        FROM users WHERE id='$id'";
    mysqli_query($link, $query);

} else {
    // пароль не тот
}

...
?>
```

Topshiriq 7:

Akkauntni o'chirish qismini ishlab chiqing.

Kirish huquqi

Saytda bitta foydalanuvchi turidan tashqari yana boshqa turlar mavjud bo'ladi. Misol uchun, sizda odatiy foydalanuvchi va admin bo'lishi mumkin. Admin odatiy foydalanuvchilarga qaraganda ko'proq huquqga ega bo'ladi.

Bu qanday amalga oshiriladi: users jadvalida yana bir maydon qo'shamiz, uni status deb nomlaymiz va har bir foydalanuvchining statusi saqlanadi: aytaylik administratorlar uchun admin so'zidan, odatiy foydalanuvchilar uchun – user so'zidan foydalanamiz.

Endi foydalanuvchini avtorizatsiyadan o'tkazishda foydalanuvchi statusini **\$_SESSION['status']** sessiyaga yozib olamiz:

```
<?php

...
$login = $_POST['login'];

$query = "SELECT * FROM users WHERE login='$login'";
$result = mysqli_query($link, $query);
$user = mysqli_fetch_assoc($result);

if (!empty($user)) {
    $hash = $user['password'];
    if (password_verify(
        $_POST['password'], $hash)) {
        $_SESSION['auth'] = true;
        $_SESSION['id'] = $user['id'];
        $_SESSION['status'] = $user['status'];
    } else {
        // parol togri kelmadi
    }
} else {
    // foydalanuvchi topilmadi
}
?>
```

Aytaylik, saytimizda faqat adminka ruxsat berilgan sahifa bor deylik. Buni kontentini faqat admin ko'rolishi uchun, bunday qilamiz:

```
<?php
```

```
if (!empty($_SESSION['auth']) and $_SESSION['auth'] == 'admin') {  
    // faqat admin bo'lsa sahifani ko'rsatamiz  
}  
?>
```

Registratsiyada o'zgarish

Shuningdek endi registratsiyamizga ham o'zgartirish kiritishimiz kerak. Endi registratsiyadan o'tayotgan foydalanuvchining statusini **INSERT** so'rovida belgilab qo'yishimiz kerak.

Odatda boshlang'ich registratsiyada barcha foydalanuvchi eng past statusni olishadi, ya'ni bizni holatda bu **user**:

```
<?php  
...  
$query = "INSERT  
    INTO users SET login='$login', password=  
        '$password', 'status'='user'";  
...  
?>
```

Eng yuqori status albatta administratorga beriladi. U admin panelda turib foydalanuvchilarning ro'yhatini ko'rishi va har qanday ishlarni, masalan, foydalanuvchini o'chirish, foydalanuvchini ismini o'zgartirib qo'yish, foydalanuvchining balansiga pul qo'shish va h.klarni bajaroladi

Xo'sh, hamma foydalanuvchi oddiy user maqomida ro'yhatdan o'tsa, unda birinchi administrator qanday paydo bo'ladi? Javob juda ham oddiy: odatiy foydalanuvchi sifatida

ro'yhatdan o'tiladi va uni bazada statusiga admin yozilishi orqali amalga oshiriladi.

Murakkabrog'i: saytni birinchi hostingga qo'yish, o'rnatish paytida forma orqali login va parolni so'rab, birinchi foydalanuvchini admin qilib qo'yish orqali ham bajarilishi mumkin. **Wordpress CMS** i kabi.

Masalalar yechish

Saytda huquqlarni taqsimlashning umumiy mohiyatini tushunib oldingiz deb o'ylayman. Unda keling, topshiriqlarni yechamiz:

Topshiriq 1:

Registratsiya kodimizga o'zgartirish kriting: unda foydalanuvchini statusini o'rnatish qo'shilgan bo'lsin.

Topshiriq 2:

Qaysidir foydalanuvchining statusini admin deb qo'lda belgilang va o'sha foydalanuvchi nomidan avtorizatsiyadan o'ting.

Topshiriq 3:

admin.php sahifasini yarating. U yerga faqat admin maqomiga ega bo'lган foydalanuvchilar kiolsin.

Topshiriq 4:

admin.php sahifasi ichida saytning barcha foydalanuvchilari ro'yhatini table jadvali ko'inishida chiqaring. Jadvalda 2ta ustun bo'lsin: biri **login**, ikkinchisi **status**.

Topshiriq 5:

Yuqoridagi topshiriqqa o'zgartirish kriting: uchinchi ustun sifatida "o'chirish" ni kriting, bu orqali har bir

foydalanuvchini id raqami asosida o'chirish imkonи mavjud bo'lsein.

Topshiriq 6:

Yuqoridagi topshiriqqa o'zgartirish kriting: admin maqomiga ega bo'lgan foydalanuvchilar qatori qizil rangda, odatiy foydalanuvchilar – yashil rangda chiqsin.

Topshiriq 7:

Yuqoridagi topshiriqqa o'zgartirish kriting: yana bir ustun qo'shing. Bu ustunda administrator foydalanuvchining maqomini o'zgartirish mumkin bo'lsein. Shunday qilingki, barcha odatiy foydalanuvchilar uchun u ustunda none deb chiqib tursin, admin uchun esa "admin qilish" nomli havola tursin. U havolaga bosganda aynan o'sha id raqamli foydalanuvchi admin maqomiga ega bo'lsein.

Topshiriq 8:

Shunday qilingki, saytning barcha sahifalarining header, ya'ni bosh qismida foydalanuvchining login va statusi ko'rinish tursin.

Topshiriq 9:

Yuqoridagi topshiriqni shunday o'zgartiringki, adminlar uchun saytning bosh qismida admin panel uchun havola tursin.

Foydalanuvchilarni bloklash

Keling, endi foydalanuvchilarni administrator tomonidan bloklash imkonini yaratamiz. Bloklash nima uchun kerak: agar foydalanuvchi saytning qoidasini buzsa uni cheklash uchun albatta. Masalan, spam qilgani, so'kingani va h.k uchun.

Blok(ing. ban | o'zb. cheklash) – bu foydalanuvchi akkaunti turadi, lekin muzlatiladi. Foydalanuvchi akkauntiga kirmoqchi bo'lganda unga rad javobi beriladi va nima uchun ban olganligining sababi ko'rsatiladi.

Banning turli xil ko'rinishlari mavjud: misol uchun, foydalanuvchini butun umrga, yoki qaysidir muddatgacha, yoki belgilangan vaqtgacha, masalan bir sutkaga yoki 2 kunga, undan keyin foydalanuvchi akkauntidan avtomatik tarzda cheklov olib tashlanadi.

Shunday qilib boshlaymiz.

Birinchi qilinadigan ish – bu administrator panelda administratorning ban qilish imkoniyatini yaratish. Buning uchun foydalanuvchilar ro'yhatiga yana bir ustun “bloklash” havolasini qo'shish kerak. U havolada tegishli foydalanuvchining id raqami GET so'rovi asosida yuboriladi va bosilgandan kerakli so'rov asosida bazadan o'sha foydalanuvchiga taqiq qo'yiladi.

Qanday usulda bloklaymiz: ma'lumotlar bazasidagi users jadvaliga yana bir maydon **banned** nomli maydonni qo'shamiz. Agar bu maydonda 1 bo'lsa unda foydalanuvchi bloklangan, aksincha 0 tursa foydalanuvchi bloklanmaganligini bildiradi.

GET so'rovimiz ban_id nomli bo'lsin va bizga ushbu id bo'yicha UPDATE so'rovi ham qo'shilsin:

```
<?php  
...  
$id = $_GET['ban_id'];
```

```
$query = "UPDATE  
    users SET login=  
        '$login', password=  
        '$password',  
        banned='0' WHERE id='$id'";  
...  
?>
```

Foydalanuvchilarni ro'yhatga olishda ham to'g'irlash kiritishimiz kerak: har bir foydalanuvchini ro'yhatga olganda banned ustuni qiymatini 0ga tenglab qo'yish kerak.

```
<?php  
...  
$query = "INSERT INTO users SET  
    login='$login',  
    password='$password', banned='0'";  
...  
?>
```

Eng asosiy o'zgartirish avtorizatsiyada qilinadi: foydalanuvchi to'g'ri login-parol kiritganidan keyin, foydalanuvchini bloklangan yoki bloklanmaganligini tekshirish kerak, agar bloklangan bo'lsa u haqida xabar e'lon qilib, avtorizatsiyani bekor qilish kerak:

```
<?php  
...  
$login = $_POST['login'];  
  
$query = "SELECT * FROM users WHERE login='$login'";
```

```
$result = mysqli_query($link, $query);

$user = mysqli_fetch_assoc($result);

if (!empty($user)) {

    $hash = $user['password'];

    if (password_verify(
        $_POST['password'], $hash) ) {

        if ($user['banned' != 1]) {

            // OK

        } else {

            // BANNED

        }

    } else {

        // parol xato

    }

} else {

    // foydalanuvchi yo'q

}

?>
```

Ma'lumotlar bazasini normallashtirish

Hozir biz foydalanuvchilarning statusini aynan o'sha foydalanuvchilar jadvalining o'zida saqlamoqdamiz. Lekin bu xato – bizda normallashmagan jadval vujudga kelib qoldi, aniq

qilib aytganda user va admin so'zlari ko'p marta takrorlanayapti.

Normallashtirishimiz kerak – statuslarimizni alohida **statuses** nomli jadvalga yozamiz, **users** jadvalida esa status ustunini o'rniغا **status_id** ustunini yaratamiz.

Endi esa statuses jadvalimizda faqat ikkita ma'lumot kiritamiz: birinchi **user** matnli, ikkinchisi **admin** matnli.

	id	name
1		user
2		admin

Endi registratsiyada **status** ustuniga yoziladigan **user** matnini o'rniغا, **status_id** ustuniga **statuses** jadvalidagi foydalanuvchining status id raqami yoziladi:

```
<?php  
...  
$query = "INSERT  
    INTO users SET login='$login', password=  
        '$password', status_id='1';  
...  
?>
```

Eng qiyin ish avtorizatsiyada: foydalanuvchining statusini olish uchun LEFT JOIN kommandasini yozishimiz kerak:

```
<?php  
...  
$login = $_POST['login'];  
  
// Foydalanuvchini va unga birikkan statusni olamiz
```

```
$query = "SELECT *, statuses.name as status FROM users
LEFT JOIN statuses ON users.status_id=
statuses.id WHERE login='$login'";

$result = mysqli_query($link, $query);
$user = mysqli_fetch_assoc($result);

if (!empty($user)) {
    $hash = $user['password'];

    if (password_verify(
        $_POST['password'], $hash)) {
        // OK

        $_SESSION['auth'] = true;
        $_SESSION['status'] = $user['status'];
    } else {
        // parol xato
    }
} else {
    // foydalanuvchi yo'q
}
?>
```

Obyektga yo'naltirilgan dasturlash – kirish

Hozir biz sizlar bilan endi PHP dasturlash tilida **OOP**(ing. Object Oriented Programming) ni o'rganamiz. Keling,

hayotdagি namunalarni ko'rib chiqamiz va uni PHP da tasvirlaymiz.

Namuna sifatida avtomobilni olaman. Unda g'ildirak, rang, kuzovi, dvigatel hajmi va h.k bor. Undan tashqari, haydovchi unga buyruq berishi mumkin: yur, to'xta, o'nga buril, chapga buril va h.z.

Umumiy xususiyatga ega bo'lgan ma'lum avtomashinalar sinfi mavjud deylik. Barchasida g'ildirak mavjud va ularga buyruq berishingiz mumkin.

Ko'chada turgan biror avtomobil o'sha klassning vakili hisoblanadi, yoki boshqacha qilib aytganda, klassning obyekti bo'ladi. Avtomobil klassining barcha obyektlarida: g'ildiraklar, ranglar, turi, va metodlari: haydash, to'xtash, o'nga yoki chapga burilish kabilar bo'ladi.

Boshqacha qilib aytganda, sinfning o'zi – bu zavodda ishlab chiqariladigan moshinalarning andozasi. Obyekt bu andozaga muvofiq tayyorlangan moshina.

PHP da sinf **class** kalit so'zi yordamida yaratiladi. Class kalit so'zidan keyin sinf nomi yoziladi. Keling Car sinfini yaratamiz:

```
<?php  
  
class Car  
  
{  
    // bu yerda kodlar  
}  
?>
```

Andozamizda endi ushbu andoza bo'yicha tayyorlangan har qanday moshinada rang uchun va yoqilg'i miqdori uchun xususiyatlar bo'ladi. Xususiyatlar – maydonlar deb ham ataladi. Buning uchun biz sinf ichiga **\$color(rang)** va **\$fuel(yoqilg'i)** xususiyatlarini yozamiz:

```
<?php  
  
class Car  
  
{  
  
    // xususiyatini yozamiz , asli olib qaralganda bu  
    // sinfning o'zgaruvchilari  
  
    public $color; // avtomobil rangi  
  
    public $fuel; // yoqilg'i miqdori  
  
}  
  
?>
```

public kalit so'zi bizning xususiyatlarimiz, ya'ni xossalaramizning ko'rinish maydonini tartibga soladi. Bu haqida to'liqroq keying darslarimizda o'rghanasiz.

Endi keling, sinfimiz ichiga metod qo'shamiz. PHP da metod – bu **function** kalit so'zidan oldin **public** kalit so'zi yozilib e'lon qilinadigan odatiy funksiyalar.

Yuqoridagi gaplarimni eslagan bo'lsangiz, avtomobil yurishi mumkin, burilishi mumkin yoki bo'lmasam to'xtashi mumkin. Endi sinfimizda ham shunga mos metodlar yartamiz:

```
<?php  
  
class Car  
  
{  
  
    public $color;  
  
    public $fuel;
```

```
// Yurish uchun buyruq:  
  
public function go()  
{  
    // qanaqadir kod bajariladi  
}  
  
  
// Burilish uchun buyruq:  
  
public function turn()  
{  
    // qanaqadir kod bajariladi  
}  
  
  
// To'xtash uchun buyruq:  
  
public function stop()  
{  
    // qanaqadir kod bajariladi  
}  
  
}  
  
?>
```

Hozir siz bilan biz avtomobilimizning andozasini qilib oldik. Endi buni zavodga yuborib o'sha sinfning obyektini tayyorlashimz kerak(ya'ni konkret avtomobil).

```
<?php  
  
new Car; // avtomobilni ishlab chiqarish-chun zavodga  
buyruq beramiz  
  
?>
```

Lekin, sinfning obyektini shunchaki yaratib qo'yish bizga hech nima bermaydi. Ushbu obyektni ishlatalish uchun uni o'zgaruvchiga saqlashimiz kerak.

```
<?php  
$myCar = new Car;  
?>
```

Avtomobilni yaratganimizdan so'ng uning xossalariga murojaat qilishimiz mumkin. Murojaat berish uchun biz -> o'q chiziqdan foydalanamiz, mana bunday: *obyektNomi->uningXossasi*

```
<?php  
$myCar = new Car;  
  
// Avtomobilning xossalarini belgilaymiz:  
$myCar->color = 'red'; // avtomobilimiz qizil rangda bo'lsin  
$myCar->fuel = 50; // yoqilg'isi esa 50  
?>
```

Bo'ldi, bizning avtomobilimiz zavodda bo'yaldi va yoqilg'i to'ldirilib chiqarildi. Endi biz unga ushbu avtomobilning metodlari orqali buyruq berishimiz mumkin. Metodga murojaat qilish ham xuddi xossaga murojaat qilishga o'xshaydi lekin metodning nomidan keyin biz qavs yozishimiz kerak. *obyektningNomi->uningMetodi();*

```
<?php  
// yaratilgan avtomobilga buyruq beramiz:  
$myCar->go(); // avtomobil->yur
```

```
$myCar->turn(); // avtomobil->burul  
$myCar->stop(); // avtomobil->to'xta  
?>
```

Obyekt xossalari bilan ishlash

Hozir biz siz bilan obyekt va uning xossalari bilan ishlashni amaliyotda o'rjanamiz.

Keling, saytimizning foydalanuvchisini tasvirlab beradigan **User** nomli klass yaratamiz. Foydalanuvchida 2ta xossa bo'lzin: ism va yosh.

```
<?php  
  
class User  
  
{  
  
    public $name; // ism xossasi  
    public $age; // yosh xossasi  
  
}  
  
?>
```

Hozircha sinfimiz hech qanday ish qilmayapti – shunchaki ushbu sinfning obyektida nimalar bo'lishi tasvirlayapti. Endi **User** sinfining vakili bo'lmish birinchi obyektimizni yaratamiz:

```
<?php  
  
class User  
  
{  
  
    public $name;  
    public $age;  
  
}
```

```
$user = new User;  
?>
```

Yodda saqlang: hamisha sinfning nomi katta harf bilan boshlanadi, lekin uning obyektlari kichik. Undan tashqari sinfni xoh **new User**, xoh **new User()** shaklida yozishimiz mumkin. Farqi yo'q.

Keling endi obyektimizning xossasiga biror nima yozamiz, keyin ushbu ma'lumotlarni ekranga chiqaramiz:

```
<?php  
  
class User  
  
{  
  
    public $name;  
  
    public $age;  
  
}  
  
  
$user = new User; // klassning obyekti  
$user->name = 'Sanjar'; // name xossasiga ism kiritamiz  
$user->age = 25; // age xossasiga yosh  
  
  
echo $user->name; // yozilgan ismni ekranga chiqaramiz  
echo $user->age; // yoshni ham  
?>
```

Tushunganingizdek, obyekt xossalariiga xohlagan ma'lumotimizni yozib keyin uni ko'rishimiz ham mumkin. Xo'sh, endi 2ta obyekt yaratamiz: 'Sanjar' va 'Abdurasul', ularning ma'lumotlarini to'ldirib, yoshlaringning yig'indisini ekranga chiqaramiz:

```
<?php
```

```
class User

{
    public $name;
    public $age;
}

// Birinchi obyekt
$user1 = new User;
$user1->name = 'Sanjar';
$user1->age = 25;

// Ikkinchchi obyekt
$user2 = new User;
$user2->name = 'Ogabek';
$user2->age = 30;

// Yoshlarning yig'indisi
echo $user1->age +
    $user2->age; // natija 55
?>
```

Biz bitta sinfdan xohlagancha obyekt yaratishimiz mumkin.
Bu chegaralanmagan.

Topshiriq 1:

Employee(ishchi) nomli sinf yarating, sinfning 3ta xossasini belgilang: **name**(ism), **age**(yosh), **salary**(maoshi).

Topshiriq 2:

Employee sinfning obyektini yarating va uning xossasiga quyidagi qiymatni kriting: ‘Otabek’ ism sifatida va 25 yosh sifatida hamda 1000 maoshi sifatida.

Topshiriq 3:

Employee sinfining ikkinchi obyektini yarating va unga ism sifatida ‘Og’abek’, yosh sifatida 26, maosh sifatida 2000 qiymatlarini kriting.

Topshiriq 4:

Otabek va Og’abeklarning maoshlari yig’indisini ekranga chiqaring.

Topshiriq 5:

Otabek va Og’abeklarning yoshlari yig’indisini ekranga chiqaring.

Metodlar bilan ishlash

Endi metodlarga o’tamiz.

Metodlar – bu aslida har bir obyekt uni chaqirishi mumkin bo’lgan funksiyalar(siz o’rganib qolgan odatiy function funksiyasi).

Metod va xossalari orasidagi farq bu xossalarni qavslarsiz chaqiramiz, metodlarni esa qavslar bilan chaqiramiz. Misol uchun: **\$user->name** – xossa, **\$user->getName()** – biror bir vazifani bajaradigan metod.

```
<?php  
  
class User  
{  
    public $name;
```

```
public $age;

// Metod

public function show()
{
    return '!!!!';
}

}

$user = new User;
$user->name = 'Sanjar';
$user->age = 25;

// Metodni chaqiramiz:

echo $user->show(); // natija '!!!!'
?>
```

Topshiriq 1:

Yuqoridagi kodga qaramasdan mustaqil ravishda aynan show metodiga ega bo'lgan **User** sinfi yarating.

Metodning parametrlari

Metod asli odatiy funksiya bo'lgani sababli, u ham barcha funksiyalar qatori parametr qabul qilishi mumkin. Keling, **show** metodimiz qanaqadir satr qabul qiladigan parametrga ega bo'lsin va uni chaqirganimizda argumentni va oxiriga ‘!!!’ ni qo'shib ekranga chiqarsin.

```
<?php
class User
```

```
{  
    public $name;  
    public $age;  
  
    public function show($str)  
    {  
        return $str . '!!!!';  
    }  
  
}  
  
$user = new User;  
$user->name = 'Коля';  
$user->age = 25;  
  
echo $user->show('hello'); // natija 'hello!!!!'  
?>
```

Topshiriq 2:

Xuddi shu sinfni mustaqil ravishda yarating.

Sinfning xossalariga **\$this** orqali murojaat qilish

Aytaylik, **show()** metodimiz sinfimizning name xossasida saqlangan qiymatni ekranga chiqaradi deylik. Sinf ichidagi metodda sinfning xossasiga murojaat qilish uchun, obyektni nomini yozishni o'rniga maxsus **\$this** o'zgaruvchisi yoziladi. Ya'ni, bizning holatda sinf ichida foydalanuvchini ismini ekranga chiqarishimiz uchun **\$this->name** kabi yozishimiz kerak:

```
<?php
```

```
class User

{
    public $name;
    public $age;

    public function show()
    {
        return $this->name; // ushbu sinfning name
xossasini qaytaramiz
    }
}

?>
```

Nima uchun sinf ichida **\$user->name** kabi yozib bo'lmaydi? Chunki bu sinfdan tashqarida joylashgan o'zgaruvchi va sinfning o'zi bu o'zgaruvchini tanimaydi. Keling sinfimizning obyektini yaratamiz va metodimizning ishlashini tekshirib ko'ramiz:

```
<?php

class User
{
    public $name;
    public $age;

    public function show()
    {
        // Ismni qaytaramiz:
```

```
        return $this->name;  
    }  
  
}  
  
  
$user = new User;  
  
$user->name = 'Sanjar';  
  
$user->age = 25;  
  
  
echo $user->show(); // natija 'Sanjar'  
?>
```

Topshiriq 1:

name(ism), **age**(yosh), **salary**(maosh) xossalari mavjud bo'lgan Employee(ishchi) sinfini yarating

Topshiriq 2:

Employee sinfining ichida ishchining ismini qaytaradigan **getName** metodini hosil qiling.

Topshiriq 3:

Employee sinfining ichida ishchining yoshini qaytaradigan **getAge** metodini hosil qiling.

Topshiriq 4:

Employee sinfining ichida ishchining maoshini qaytaradigan **getSalary** metodini hosil qiling.

Topshiriq 5:

Employee sinfining ichida agar ishchining yoshi 18 dan katta bo'lsa true, aks holda false qaytaradigan **checkAge** metodini hosil qiling.

Topshiriq 6:

Employee sinfining 2ta obyektini yarating va uning xossalariga tegishli qiymatlarni yozing. **getSalary** metodi yordamida ularning maoshlari yig'indisini chiqaring.

Xossaga ma'lumot kiritish

\$this yodamida faqat o'qish emas balki ma'lumot yozish ham mumkin. Keling, foydalanuvchini ismini qabul qiladigan va uni sinfning name xossasiga yozib qo'yadigan **setName** metodini hosil qilamiz.

```
<?php

class User

{

    public $name;

    public $age;

    public function setName($name)

    {

        $this->name = $name;

    }

}

$user = new User;

$user->name = 'Sanjar';
```

```
$user->age = 25;  
  
$user->setName ('Abdurasul');  
  
echo $user->name; // natija 'Abdurasul'  
?>
```

Topshiriq 7:

name(ism), **age**(yosh) xossalariiga ega bo'lgan **User** sinfini yarating.

Topshiriq 8:

Foydalanuvchining yoshini o'zgartiradigan **setAge** metodini hosil qiling.

Topshiriq 9:

Ismi "Sanjar" va yoshi 25 bo'lgan **User** sinfining obyektini yarating. **setAge** metodi yordamida yoshti 30ga o'zgartiring. Yangi qiymatni ekranga chiqaring.

Topshiriq 10:

name(ism), **salary**(maosh) xossalari mavjud bo'lgan **Employee**(ishchi) sinfini hosil qiling. Maoshni 2 barobarga oshiradigan **doubleSalary** metodini yarating.

Topshiriq 12:

Eni va bo'yisi xossalari belgilangan **Rectangle**(to'rtburchak) sinfini yarating.

Topshiriq 13:

Rectangle sinfiga to'rtburchakning yuzini topadigan **getSquare** metodini yarating.

Topshiriq 14:

Rectangle sinfida to'rtburchakning perimetrini topadigan **getPerimeter** metodini yarating.

Metodga \$this orqali murojaat qilish

\$this orqali nafaqat obyektning xossasiga balki uning metodlariga ham murojaat qilish mumkin. O'ylaymanki, bu ozmi-ko'pmi sizga tushunarli bo'lib qoldi, unda keling darrov buni amaliyotda sinab ko'ramiz.

Bizda User sinfi va foydalanuvchining yoshini o'zgartirish uchun setAge metodi ham mavjud deylik:

```
<?php  
  
class User  
  
{  
  
    public $name;  
  
    public $age;  
  
  
    public function setAge($age)  
    {  
  
        $this->age = $age;  
  
    }  
  
}  
  
?>
```

Keling, endi kiritilgan yoshni tekshiramiz: agar u 18 dan 60 yoshgacha bo'lgan bo'lsa, unda yoshni yangisiga o'zgartiramiz, unday bo'lmasa tegmaymiz:

```
<?php  
  
class User
```

```
{  
  
    public $name;  
  
    public $age;  
  
  
    public function setAge($age)  
    {  
        if ($age >= 18 and $age <= 60) {  
  
            $this->age = $age;  
  
        }  
  
    }  
  
?>
```

Bizga yana bir metod ham kerak bo'ladi, bu metod hozirgi foydalanuvchining yoshiga qandaydir miqdorda yana yosh qo'shami. Uni **addAge** deb nomlaymiz:

```
<?php  
  
class User  
  
{  
  
    public $name;  
  
    public $age;  
  
  
    public function setAge($age)  
    {  
        if ($age >= 18 and $age <= 60) {  
  
            $this->age = $age;  
  
        }  
  
    }  
  
}
```

```
public function addAge($years)
{
    $this->age = $this->age + $years;
}
?>
```

Yoshni tekshirish **addAge metodida ham qo'llanilishi kerak:**

```
<?php
class User
{
    public $name;
    public $age;

    public function setAge($age)
    {
        if ($age >= 18 and $age <= 60) {
            $this->age = $age;
        }
    }

    public function addAge($years)
    {
        $newAge = $this->age + $years;
        // Agar yangi kiritilgan qiymat 18dan 60 gacha
        bo'lsa:
```

```
    if ($newAge >= 18 and $newAge <= 60) {  
  
        $this->age = $newAge; //unda yoshni  
        o'zgartiramiz  
  
    }  
  
}  
  
?>
```

Hozir yoshni tekshirishni 2ta joyda kiritdik(**setAge** funksiyasi hamda **addAge** funksiyasi ichida), bu esa unchalik yaxshi emas: agar biz cheklowni 18 emas 19dan boshlanishini xohlasak, o'zgartirmoqchi bo'lsak, bizga buni 2ta joyda o'zgartirishimizga to'g'ri keladi – bu esa noqulaylik tug'diradi va biz qayergadir o'zgartirish kiritishni unutib qo'yishimiz mumkin va bu juda xavfli(agar bu forma ma'lumotlarini tekshirishda bo'lsa va h.z)

Keling, yoshni to'g'riliini tekshirishni alohida metodga olib o'tamiz va bu metodni **isAgeCorrect** deb nomlaymiz, bu metod yoshni qabul qiladi va un tekshiradi. Bu metodni bemalol sinf ichida qo'llashimiz mumkin:

```
<?php  
  
class User  
  
{  
  
    public $name;  
  
    public $age;  
  
  
    public function isAgeCorrect($age)  
    {  
  
        if ($age >= 18 and $age <= 60) {
```

```
        return true;

    } else {

        return false;
    }

}

public function setAge($age)

{
    if ($this->isAgeCorrect($age)) {

        $this->age = $age;
    }
}

public function addAge($years)

{
    $newAge = $this->age + $years;

    if ($this->isAgeCorrect($newAge)) {

        $this->age = $newAge;
    }
}

?>
```

Endi agar shartni o'zgartirmoqchi bo'lsak, bitta isAgeCorrect metodi ichidan o'zgartirish orqali kerak bo'lsa millionta joydagi shartlarni o'zgartirish uchun ketadigan vaqtimizni tejab qolgan bo'lamiz.

Aytmoqchi, **isAgeCorrect** metodimizni yanada chiroyliroq qilishga harakat qilib ko'raylik?!:

```
<?php  
  
public function isAgeCorrect($age)  
{  
    return $age >= 18 and $age <= 60;  
}  
?>
```

Ko'rningizmi? Mantiqiy ishoralar orqali 4-5 qatorli yoziladigan kodimizni 1 qatorda yozib qo'ydik. Bunaqa usullarni katta loyihalarda qo'llash ancha vaqtni tejaydi.

Topshiriq 1:

Yuqoridagi yaratgan **User** sinfimizni barcha metodlari bilan mustaqil yaratishga harakat qiling.

Topshiriq 2:

User sinfining obyektini yarating va **setAge** hamda **addAge** metodlari ishlashini tekshiring

Topshiriq 3:

User sinfida foydalanuvchining yoshini ma'lum qiymatga kamaytiradigan **subAge** metodini hosil qiling.

public va private kirish modifikatorlari

Keling, endi sinfimizning metodlari va xossalari oldidan yozgan **public** kalit so'zimiz nima qilishi haqida o'rganamiz.

public kalit so'zi xossa va metodning sinfdan tashqarida ham qo'llanilishi mumkinligini ko'rsatadi. Uning aksi bo'lmish, **private** kalit so'zi xossa va metodga tashqaridan murojaat

qilib bo'lmasligini bildiradi. Ya'ni buni ingliz tilidan tarjima qilsak ham aniq bo'ladi. **Public** – inglizcha, ommaviy, ochiq degan ma'noni bildiradi. **Private** – inglizcha, maxfiy, yopiq degan ma'noni bildiradi.

Bu nimaga kerak? Misol uchun, ayrim funksionallarga ega bo'lgan sinfimiz mavjud. Metodlar mavjud, ammo metodlarning ba'zilari yordamchi metodlar hisoblanadi.

Ana shu metodlarni sinfdan tashqarida ishlatib bo'lmaydigan qilib qo'yish afzal.

Bu holatda biz ushbu yordamchi metod kodlarini oson tahrirlashimiz va ushbu metodlarimizga tashqaridan hech qanday ta'sir qilmasligiga ishonch hosil qilishimiz mumkin.

Bunday yondashuv – **inkapsulatsiya** deb ataladi. Ya'ni tashqarida mavjud bo'lmasligi kerak bo'lgan metodlarni sinfning ichidan boshqa yerda ishlatalmasligi. Bunday holatda dasturchining hayoti ancha osonlashadi.

Xossalar ham ana shunday. Ba'zi xossalar faqat yordamchi funksiyalarga tegishli bo'ladi va uni sinfning tashqarida ochiq qoldirish kodimizni ishdan chiqishiga ham olib kelishi mumkin.

Tashqaridan murojaat qilib bo'lmaydigan bunday xossa va metodlar maxfiy yoki yopiq metod va xossalar deb ataladi va **private** kalit so'zi yordamida e'lon qilinadi.

Keling, unda **\$name** va **\$age** xossalarimizni yopiq deb e'lon qilamiz va unga tashqaridan murojaat qilib ko'ramiz:

```
<?php  
class User
```

```
{  
    private $name;  
    private $age;  
}  
  
$user = new User;  
  
// name xossasi yopiqligi uchun xato chiqadi  
$user->name = 'Sanjarbek';  
?>
```

Amaliyotda qo'llanilishi

O'tgan darsimizda yozgan User sinfimiz misolida ko'ramiz:

```
<?php  
  
class User  
  
{  
    public $name;  
    public $age;  
  
    public function isAgeCorrect($age)  
    {  
        return $age >= 18 and $age <= 60;  
    }  
  
    public function setAge($age)  
    {
```

```
if ($this->isAgeCorrect($age)) {  
    $this->age = $age;  
}  
  
}  
  
public function addAge($years)  
{  
    $newAge = $this->age + $years;  
  
    if ($this->isAgeCorrect($newAge)) {  
        $this->age = $newAge;  
    }  
}  
}  
?>
```

Bilganingizdek, `isAgeCorrect` metodimiz yordamchi hisoblanadi va biz bu metodni sinfning tashqarisida ishlatalish uchun yaratmaganmiz.

Boshqa dasturchi proyektimiz ustida ish olib borayotganda to'satdan bu metodni ishlatab yuborib, xatolikka yo'l qo'myasligi uchun, mantiqan biz buni maxfiy/yopiq mteod deb belgilashimiz kerak.

```
<?php  
  
class User  
{  
    public $name;  
    public $age;
```

```
private function isAgeCorrect($age)
{
    return $age >= 18 and $age <= 60;
}

public function setAge($age)
{
    if ($this->isAgeCorrect($age)) {
        $this->age = $age;
    }
}

public function addAge($years)
{
    $newAge = $this->age + $years;

    if ($this->isAgeCorrect($newAge)) {
        $this->age = $newAge;
    }
}

?>
```

Odatda barcha maxfiy bo'lgan metodlar sinfning eng oxirgi qismiga qo'yiladi, keling metodimizni eng pastga olib tushamiz:

```
class User

{
    public $name;
    public $age;

    public function setAge($age)
    {
        if ($this->isAgeCorrect($age)) {
            $this->age = $age;
        }
    }

    public function addAge($years)
    {
        $newAge = $this->age + $years;

        if ($this->isAgeCorrect($newAge)) {
            $this->age = $newAge;
        }
    }

    private function isAgeCorrect($age)
    {
        return $age >= 18 and $age <= 60;
    }
}

?>
```

Maxus qoida mavjud: agar siz yangi metod yaratsangiz-u, uni yopiqmi yo ochiqmi, qanday e'lon qilishni bilmasangiz, unda uni yopiq metod qilib e'lo qiling. Kelgusida, agar u tashqarida ham kerak bo'ladigan bo'lsa u holda uni **public**, ya'ni ochiq metod deb belgilab qo'yasiz.

Xulosa sifatida: **public** va **private** dastur logikasini amalga oshirish uchun emas, balki dasturchini turli xil xatolardan himoya qilish uchun kerak.

Topshiriq 1:

User sinfini mustaqil ravishda ishlab chiqing.

Topshiriq 2:

Sinfning tashqarisdan **isAgeCorrect** metodini chaqiring.
Xatolik chiqqanligiga ishonch hosil qiling.

Topshiriq 3:

\$name va **\$course**(kurs 1dan 5 gacha bo'ladi) xossali **Student** sinfini yarating.

Topshiriq 4:

Student sinfi ichida studentni keying kursga o'tkazadigan ochiq **transferToNextCourse** metodini yarating

Topshiriq 5:

transferToNextCourse metodida keyingi kurs 5 dan katta emasligini tekshiring.

Topshiriq 6:

Sinf ichida alohida yopiq **isCourseCorrect** metodi ichida kursni to'g'rilibini tekshiring tekshiring.

Obyekt konstruktori

Quyidagi kodga e'tibor bering:

```
<?php

class User

{
    public $name;
    public $age;
}

$user = new User;

$user->name = 'Sanjar';
$user->age = 25;

echo $user->name;
echo $user->age;

?>
```

Hozirgi kodimizda obyektning qanaqadir xossalariq ma'lumot kiritishni unutib qo'yishimiz oson, ayniqsa bunday xossalari ko'p bo'lsa...

Bu kodni:

```
<?php

$user = new User;

$user->name = 'Sanjar';
$user->age = 25;
```

?>

Bunga almashtiramiz:

```
<?php  
  
$user = new User('Sanjar', 25); // birdan ma'lumotni  
to'ldirgan xolda obyekt yaratamiz  
  
?>
```

Ya'ni, obyektning kerakli xossalari to'ldirilganligiga ishonch hosil qilish uchun bunday usulni ishlatalimiz.

Muammoni bartaraf etish uchun bizga **__construct**(boshida 2ta tagchiziq mavjud) nomli *konstruktor metodimiz* yordam beradi. Agar bu metod sinf ichida yozilgan bo'lsa, unda obyekt yaratilgan paytdan darrov chaqiriladi. Mohiyat ham ana shunda:

```
<?php  
  
class User  
  
{  
  
    public $name;  
  
    public $age;  
  
  
    public function __construct()  
    {  
        echo '!!!!';  
    }  
}  
  
  
$user = new User; // natija '!!!!'  
?>
```

Konstruktor ham boshqa metodlar kabi parametr qabul qilishi mumkin, namuna:

```
<?php

class User

{

    public $name;

    public $age;

    public function __construct($var1, $var2)
    {
        echo $var1 + $var2;
    }

}

$user = new User(1, 2); // natija 3

?>
```

Keling, endi kodimizga sinfimiz nomidan kelib chiqqan holda, konstruktordan foydalanib boshqacharoq ishlov beramiz:

```
<?php

class User

{

    public $name;

    public $age;

    public function __construct($name, $age)
    {
```

```
    $this->name = $name; // name xossasiga $name  
argumetini yozamiz  
  
    $this->age = $age; // age xossasiga $age  
argumetini yozamiz  
}  
  
}  
  
  
$user = new User('Sanjar', 25);  
  
  
echo $user->name; // natija 'Sanjar'  
echo $user->age; // natija 25  
?>
```

Topshiriq 1:

name, age va salary ochiq xossali Employee sinfini yarating. Shunday qilingki, uni obyektini yaratayotganingiz bilanoq **__construct** metodi orqali uning xossalari to'ldirilsin.

Topshiriq 2:

Ismi ‘Sanjar’, yoshi 25, maoshi 1000 bo’lgan **Employee** sinfining obyektini yarating.

Topshiriq 3:

Ismi ‘Abdurasul’, yoshi 30, maoshi 2000 bo’lgan **Employee** sinfining ikkinchi obyektini yarating

Topshiriq 4:

Sanjar va Abdurasullarning maoshlari yig'indisini ekranga chiqaring.

Getter(oluvchi) va setter(o'rnatuvchi)lar bilan ishlash

Kodga qarang:

```
<?php

class User

{

    public $name;

    public $age;

    public function setAge($age)

    {

        if ($this->isAgeCorrect($age)) {

            $this->age = $age;

        }

    }

    private function isAgeCorrect($age)

    {

        return $age >= 18 and $age <= 60;

    }

}

?>
```

Ko'rganingizdek, bizda \$name va \$age ochiq xossalari hamda setAge ochiq metodi mavjud bo'lib, bu metod yoshni o'zgartirish va isAgeCorrect metodi yoshni tekshirish uchun yopiq metod sanaladi.

Ma'lumki, yoshni o'zgartirish bizning setAge metodimiz orqali amalga oshiriladi, sababi bu metodda kiritilgan yoshni tekshirish sharti mavjud. Lekin, bizga yosh xossasini o'zgartirish uchun aynan setAge metodi kerak emas:

```
<?php
```

```
$user = new User;

// setAge metodini o'rniغا boshqa noto'g'ri yoshni belgilaymiz

$user->age = 100500; // va bu ishlayveradi!

?>
```

Attang... Biz yoshni har doim setAge bilan o'zgartirilishini xohlaymiz-u, lekin biror boshqa dasturchi tasodifan bu xossani ishlataladigan bo'lsa, tizimda ancha katta xatolik ro'y berishi mumkin. Bizni kesimimizdagi bu dasturda hozirchalik katta xato yo'q. Agar katta loyihalarda ishlasangiz, albatta bu ayanchli oqibatlarga olib kelishi mumkin.

Bu muammoni yechish uchun yoshni yopiq xossali deb e'lon qilishimiz darkor:

```
<?php

class User

{

    public $name;

    private $age; // yopiq xossa


    public function setAge($age)

    {

        if ($this->isAgeCorrect($age)) {

            $this->age = $age;

        }

    }

}
```

```
private function isAgeCorrect($age)
{
    return $age >= 18 and $age <= 60;
}

?>
```

Endi yoshni to'g'ridan-to'g'ri xossa orqali yana bir marta o'rnatib ko'ramiz:

```
<?php
$user = new User;

$user->age = 100500; // ups... xatolik! Yashavor Vasya!
?>
```

Barakalla, istaganimizni amalga oshirdik. Biroq, endi boshqa muammo vujudga keldi: yosh xossasi yopiqligi uchun uni tashqaridan o'qiy olmaymiz:

```
<?php
$user = new User;

$user->setAge(50);

echo $user->age; // xatolik
?>
```

Muammoni yechish uchun **\$age** xossasini qiymatini o'qish uchun yana bir **getAge** nomli metodini hosil qilamiz:

```
<?php
```

```
class User

{
    public $name;

    private $age;

    // Foydalanuvchini yoshini o'qish uchun maxsus metod
    public function getAge()
    {
        return $this->age;
    }

    public function setAge($age)
    {
        if ($this->isAgeCorrect($age)) {
            $this->age = $age;
        }
    }

    private function isAgeCorrect($age)
    {
        return $age >= 18 and $age <= 60;
    }
}

?>
```

Endilikda, foydalanuvchining yoshini xohlasak tahrirlashimiz, xohlasak o'qishimiz mumkin:

```
<?php  
  
$user = new User;  
  
  
$user->setAge(50);  
  
  
echo $user->getAge(); // natija 50  
  
?>
```

Eslatma: nima uchun xossani o'zidan to'g'ridan-to'g'ri foydalanib uni qiymatini olmaymiz yoki unga qiymat kiritmaymiz, sababi xossaga ma'lumot kiritayotganda biz uni qanaqadir shart bilan tekshirishimiz kerak.

Hozirgi biz standart bo'yicha yondashdik. Bu yondashuv terminda, bizning holatdagi getAge metodi - **getter**(ing. oluvchi), setAge metod – **setter**(ing. o'rnatuvchi) deb ataladi.

Bunday yondashuv bizga setterda qanaqadir shart bajarilishi kerak bo'lganda asqotadi.

Ko'p holatlarda hech qanday shart, tekshiruvlar bajarilmasa ham xossa yoki metodlar yopiq, ya'ni private qilib e'lon qilinadi, ularga kirish uchun esa getter va setterlar ishlataladi.

Nima uchun? Sababi, kelgusida biz unga shart yoki tekshiruv qo'shishni istagan bo'lishimiz mumkin, agar biz faqat bitta setter orqali qiymat kiritadigan bo'lsak, uni bitta metod orqali amalga oshirishimiz mumkin va kelgusida o'zgartirish ham kiritishimiz mumkin, Ya'ni, siz xossalarni hammasiga bittama-bitta qiymat berib ketmasdan, bitta metod orqali hammasiga qiymat kiritishingiz mumkin bo'ladi.

Topshiriq 1:

name, age va salary yopiq xossalari Employee sinfini yarating.

Topshiriq 2:

Employee sinfining barcha xossalari uchun getter va setterlarni yarating

Topshiriq 3:

Employee sinfiga yoshni 1dan 100 gacha bo'lgan o'lchamda tekshiradigan `isAgeCorrect` yopiq metodini qo'shing. Bu metod yangi yoshni kiritishdan oldin `setAge` setterida qo'llanilishi kerak(agar yosh noto'g'ri kiritilgan bo'lsa o'zgarmasligi kerak).

Topshiriq 4:

Xodimlarimizning maoshi dollarda saqlanadi deylik. Shunday qilingki, `getSalary` getteri orqali maoshning oxiriga dollar belgisi qo'shilsin. Boshqacha qilib aytganda, salary xossamiz oddiy raqamni o'zida saqlaydi, lekin `getSalary` orqali unga qo'shimcha qiymat qo'shib, shaklini o'zgartirasiz.

Faqat o'qish uchun mo'ljallangan xossalari

Hozir biz siz bilan obyektda qanaqadir xossani ma'lumot kiritish uchun emas, balki faqat o'qish uchun qilamiz.(ing. `read-only`)

Bu quyidagicha bajariladi: bunday xossa uchun setter emas, faqat getter yaratilishi kerak.

Bunday holatda xossa faqat getter orqali o'qiladi lekin setter bo'limgani uchun unga ma'lumot kiritib bo'lmaydi. Xossaning boshlang'ich qiymati obyektni yaratish paytida

konstruktorda beriladi. Keling yozilganni endi amalda tadbiq qilamiz. User nomli sinf mavjud:

```
<?php  
  
class User  
  
{  
  
    private $name;  
  
    private $age;  
  
}  
  
?>
```

Shunday qilaylikki, **name** xossasi faqat o'qish uchun, **age** xossasi esa o'qish va yozish uchun bo'lsin. Buning uchun **name** xossasi uchun bitta **getter**, **age** xossasi uchun esa **getter** va **setterlar** belgilaymiz:

```
<?php  
  
class User  
  
{  
  
    private $name;  
  
    private $age;  
  
  
    public function getName()  
    {  
        return $this->name;  
    }  
  
  
    public function getAge()  
    {  
        return $this->age;  
    }
```

```
}

public function setAge($age)
{
    $this->age = $age;
}

?>
```

Keling, endi xossalaramizning boshlang'ich qiymatini obyektni yaratishda konstruktorga uzatish uchun, konstruktor yaratamiz:

```
<?php

class User
{

    private $name;
    private $age;

    public function __construct($name, $age)
    {
        $this->name = $name;
        $this->age = $age;
    }

    public function getName()
    {
        return $this->name;
    }
}
```

```
public function getAge()  
{  
    return $this->age;  
}  
  
public function setAge($age)  
{  
    $this->age = $age;  
}  
}  
?>
```

Hammasi hal – muammomiz yechildi:

```
<?php  
  
$user = new User('Sanjar',  
    25); // boshlang'ich qiymatli obyektni yaratamiz  
  
// ismni faqat o'qib bo'ladi, yozib bo'lmaydi  
echo $user->getName(); // natija 'Sanjar'  
  
// Yosh o'qish va yozish uchun  
echo $user->getAge(); // natija 25  
echo $user->setAge(30); //  
    установим возраст в значение 30  
echo $user->getAge(); // natija 30  
?>
```

Topshiriq 1:

name(ism), **surname**(familiya) va **salary**(maosh) xossalariga ega bo'lган Employee sinfini yarating.

Topshiriq 2:

Shunday qiling, **name** va **surname** xossalar faqat o'qilishi mumkin bo'lsin, **salary** xossasi esa ham o'qilishi, ham yozilishi mumkin bo'lsin.

Sinflarni alohida fayllarda saqlash

Bu darsimizga qadar biz sinflarimizni bitta faylda yozdik hamda ularni aynan o'sha fayllarning o'zida ishlattik. Real hayotda sinflar odatda alohida fayllarda saqlanadi, har bir sinf alohida faylda. Shu bilan birga fayl nomi qanday bo'lsa sinfning nomi ham ana shunday bo'lishi kerak. Misol uchun, bizda User sinfi mavjud, uni User.php faylida saqlaymiz. Tushunarli-a?

Keling, amaliyotda ko'ramiz. User sinfli User.php faylini yaratamiz:

```
<?php  
  
// User.php  
  
  
class User  
  
{  
  
  
}  
?>
```

Aytaylik, User sinfini ishlatmoqchi bo'lган index.php faylimiz bor. Biz bu faylda shunchaki olib User sinfini obyektini

yaratish xatolikka sabab bo'ladi. Chunki PHP u sinfning kodini topishga imkonи bo'lmaydi:

```
<?php  
// index.php  
  
$user = new User; // bu xatolik beradi  
?>
```

Buning uchun biz index.php faylimizga ushbu sinf mavjud bo'lgan faylni ulashimiz kerak. Bu require buyrug'i asosida qilinadi.

```
<?php  
// index.php  
  
  
require 'User.php'; // sinfni ulaymiz  
  
$user = new User; // endi xatolik bo'lmaydi  
?>
```

Lekin require funksiyasi bilan ishlash unchalik qulay emas, agar siz bir xil nomli ko'p fayllarni bir xil joyda ulamoqchi bo'lsangiz, unda PHP xatolik beradi – bu fayl allaqachon ulangan kabi.

Qulayroq usul bu **require_once** buyrug'ini ishlatish, bu funksiya bilan avvalgi darslarimizda tanishgansiz:

```
<?php  
// index.php  
  
  
require_once 'User.php';  
  
$user = new User;
```

?>

Topshiriq 1:

name(ism), surname(familiya), patronymic(sharif) va salary(maosh) xossalariiga ega bo'lgan Employee sinfini yarating. Bu sinf alohida faylda saqlansin.

Topshiriq 2:

Index.php faylida Employee sinfini ulab uning 2 ta obyektini yarating va ixtiyoriy ma'lumotlar kiritib ularning maoshlari yig'indisini ekranga chiqaring.

Obyektlarni massivlarda saqlash

User nomli sinf berilgan bo'lsin:

```
<?php  
  
class User  
  
{  
  
    public $name;  
  
    public $age;  
  
  
    public function __construct($name, $age)  
    {  
        $this->name = $name;  
        $this->age = $age;  
    }  
}  
  
?>
```

Keling endi ushbu sinfning 3ta obyektini yaratamiz:

```
<?php
```

```
$user1 = new User('Sanjar', 21);  
  
$user2 = new User('Otabek', 22);  
  
$user3 = new User('Ogabek', 23);  
  
?>
```

Endi yaratgan obyektlarimizni **\$users** massiviga joylaymiz:

```
<?php  
  
$user1 = new User('Sanjar', 21);  
  
$user2 = new User('Otabek', 22);  
  
$user3 = new User('Ogabek', 23);  
  
  
$users[] = $user1;  
$users[] = $user2;  
$users[] = $user3;  
  
  
var_dump($users);  
?>
```

Umuman o'zi biz obyektni yaratib uni saqlagan o'zgaruvchimiz kerak emas. Kodimizni qisqartirsak bo'ladi:

```
<?php  
  
$users[] = new User('Sanjar', 21);  
$users[] = new User('Otabek', 22);  
$users[] = new User('Ogabek', 23);  
  
  
var_dump($users);  
?>
```

Endi kodimizni boshqacha usulda qayta yozamiz – elementni yaratilishi bilanoq massivga qo'shamiz:

```
<?php  
  
$user1 = new User('Sanjar', 21);  
  
$user2 = new User('Otabek', 22);  
  
$user3 = new User('Ogabek', 23);  
  
  
$users = [$user1, $user2, $user3];  
  
var_dump($users);  
  
?>
```

Bu yerda ham o'rtadagi o'zgaruvchilardan qutulishimiz mumkin:

```
<?php  
  
$users = [  
  
    new User('Sanjar', 21),  
  
    new User('Otabek', 22),  
  
    new User('Ogabek', 23)  
  
];  
  
  
var_dump($users);  
  
?>
```

Obyektlarni massivga qay usulda saqlashimiz yoki yaratishimizning farqi yo'q, asosiysi bu mohiyati: obyekt massivlarda saqlanishi mumkin.

Keyin ushbu obyektlarni siklga qo'yishimiz mumkin:

```
<?php
```

```
$users = [  
    new User('Sanjar', 21),  
    new User('Otabek', 22),  
    new User('Ogabek', 23)  
];  
  
foreach ($users as $user) {  
    echo $user->name .  
        ' ' . $user->age . '<br>';  
}  
?>
```

Topshiriq 1:

name(shahar nomi), **population**(aholi soni) xossalariiga ega bo'lган City(shahar) sinfini yarating.

Topshiriq 2:

City sinfining 5 obyektini yarating va ularni massivga joylang.

Topshiriq 3:

Massivni siklga qo'ying va massiv ichidagi barcha obyektlarning shahar va aholi sonlarini ekranga chiqaring.

Konstruktorda xossalarning boshlang'ich qiymati

2 ta xossali qanaqadir sinfga egamiz deylik:

```
<?php  
  
class Test  
{  
    public $prop1;  
    public $prop2;
```

```
}
```

```
?>
```

Keling, endi shunday qilamiz, ushbu sinfning obyekti yaratilishi bilanoq sinfning xossalarda qanaqadir qiymatlar mavjud bo'lsin.

Allaqachon bilganingizdek, obyektni yaratganingizda avtomatik tarzda **__construct** metodi chaqiriladi. Shundan foydalanib, ushbu metodning ichida xossalarning boshlang'ich qiymatlarini berib qo'yamiz:

```
<?php
```

```
class Test
```

```
{
```

```
    public $prop1;
```

```
    public $prop2;
```



```
    public function __construct()
```

```
    {
```

```
        $this->prop1 =
```

```
            'value1';
```

```
        $this->prop2 =
```

```
            'value2';
```

```
    }
```

```
}
```



```
$test = new Test;
```

```
echo $test->prop1; // natija 'value1'  
echo $test->prop2; // natija 'value2'  
?>
```

Qo'llanilishi

Aytaylik, bizda **name** va **course** xossali **Student** sinfi mavjud. Shunday qilamiz, obyekt yaratilishi paytida studentning ismi kiritiladi, kurs esa avtomatik tarzda 1 qiymat ega bo'ladi:

```
<?php  
  
class Student  
  
{  
  
    private $name;  
  
    private $course;  
  
  
    public function __construct($name)  
    {  
  
        $this->name = $name;  
  
        $this->course = 1;  
  
    }  
  
}  
?>
```

Xossalaramiz uchun getter hosil qilamiz:

```
<?php  
  
class Student  
  
{  
  
    private $name;  
  
    private $course;
```

```
public function __construct($name)
{
    $this->name = $name;
    $this->course = 1;
}

// Studentning ismini oluvchi metod:
public function getName()
{
    return $this->name;
}

// Studentning o'qiyotgan kursini oluvchi metod:
public function getCourse()
{
    return $this->course;
}
}

?>
```

Yaratilgan studentning ismi o'zgarmaydi va faqat o'qish uchun bo'ladi, lekin kursi uchun uni keying kursga o'tkazadigan metod qilamiz:

```
<?php
class Student
{
    private $name;
    private $course;

    public function __construct($name)
```

```
{  
    $this->name = $name;  
    $this->course = 1;  
}  
  
public function getName()  
{  
    return $this->name;  
}  
  
public function getCourse()  
{  
    return $this->course;  
}  
  
public function transferToNextCourse()  
{  
    $this->course++;  
}  
}  
?>
```

Sinfimizning ishlashini tekshiramiz:

```
<?php  
$student = new Student(  
    'Sanjar'); // sinfni obyekti yaratiladi
```

```
echo $student->  
    getCourse(); // natija 1 - boshlang'ich qiymat  
  
$student->transferToNextCourse(); // studentni keyingi  
kursga o'tkazamiz  
  
echo $student->getCourse(); // natija 2  
  
?>
```

Topshiriq:

Student sinfini mustaqil ishlab chiqing.

Xossani e'lon qilishda uning boshlang'ich qiymati
Keyingi sinfni ko'zdan kechiramiz:

```
<?php  
  
class Test  
  
{  
  
    public $prop1;  
  
    public $prop2;  
  
  
    public function __construct()  
    {  
  
        $this->prop1 =  
            'value1'; // prop1 xossasining boshlang'ich  
qiymati  
  
        $this->prop2 =  
            'value2'; // prop2 xossasining boshlang'ich  
qiymati  
    }  
  
}
```

```
$test = new Test;  
  
echo $test->prop1; // natija 'value1'  
  
echo $test->prop2; // natija 'value2'  
?>
```

Yuqoridagi kodimizni ko'rganingizdek, obyekt konstruktorida biz xossalarning boshlang'ich qiymatlarini berayapmiz. Lekin, aslida kodimizni yanada qisqartirishimiz mumkin, bu xossalarning boshlang'ich qiymatini uni e'lon qilish bilanoq belgilab qo'yishimiz mumkin:

```
<?php  
  
class Test  
  
{  
  
    public $prop1 =  
        'value1';  
  
    public $prop2 =  
        'value2';  
  
}  
  
  
$test = new Test;  
  
echo $test->prop1; // natija 'value1'  
  
echo $test->prop2; // natija 'value2'  
?>
```

Barcha xossalarga boshlang'ich qiymat berish shart emas, albatta:

```
<?php  
  
class Test  
  
{
```

```
public $prop1 =  
    'value1'; // boshlang'ich qiymat beramiz  
public $prop2; // qiymat bermaymiz  
}  
?>
```

Xossaga boshlang'ich qiymat berishda, turli xil amallarni bajarishimiz ham mumkin bo'ladi:

```
<?php  
  
class Test  
{  
  
    public $prop = 1 +  
        2; // sonlarning yig'indisini topamiz  
}  
  
  
$test = new Test;  
  
echo $test->prop; // natija 3  
?>
```

Faqatgina shunaqa primitiv amallar bajarishimiz mumkin, lekin funksiya va shunga o'xshashlarni xossaga qiymat sifatida yozolmaymiz

Qo'llanilishi

Aytaylik, bizda course xossasiga konstruktorda boshlang'ich qiymat berilgan Student sinfi mavjud:

```
<?php  
  
class Student  
{  
  
    private $name;
```

```
private $course;

public function __construct ($name)
{
    $this->name = $name;
    $this->course =
        1; // kursning boshlang'ich qiymati
}

// Ism uchun getter:
public function getName()
{
    return $this->name;
}

// Kurs uchun getter:
public function getCourse()
{
    return $this->course;
}

// Studentni keyingi kursga o'tkazish:
public function transferToNextCourse()
{
    $this->course++;
}

}
```

?>

Keling, endi xossalarning boshlang'ich qiymatini konstruktorda emas balki xossani e'lon qilishni o'zida belgilab qo'yamiz:

```
<?php

class Student

{
    private $name;

    private $course =
        1; // kursning boshlang'ich qiymati


    public function __construct($name)
    {
        $this->name = $name;
    }

    // Ism uchun getter:
    public function getName()
    {
        return $this->name;
    }

    // Kurs uchun getter:
    public function getCourse()
    {
        return $this->course;
    }
}
```

```
// Studentni keyingi kursga o'tkazish:  
  
public function transferToNextCourse()  
{  
    $this->course++;  
}  
}  
?>
```

Qo'llanilishi

Aytaylik, bizda son qo'shadigan **add** metodi va barcha qo'shilgan sonlarning yig'indisin oladigan **getSum** metodiga ega bo'lgan **Arr** sinfi mavjud:

```
<?php  
  
class Arr  
  
{  
    // sonlarni saqlash uchun massiv:  
    private $numbers;  
  
    // sonni massivga qo'shish:  
    public function add($num)  
    {  
        $this->numbers[] = $num;  
    }  
  
    // Tanlovdagi sonlarning yig'indisini topish:  
    public function getSum()  
    {
```

```
        return array_sum($this->numbers);  
    }  
}  
?>
```

Keling, endi **Arr** sinfimizni ishlatalamiz va unga bir nechta sonlar qo'shib ularning yig'indisini topib ko'ramiz:

```
<?php  
  
$arr = new Arr;  
  
  
$arr->add(1);  
$arr->add(2);  
$arr->add(3);  
  
  
echo $arr->getSum(); // natija 6  
?>
```

Hozirchalik hammasi yaxshi ishlayapti, lekin agar obyektni yaratgandan so'ng darrov getSum metoddini chaqirsak nima bo'ladi? Bunday tarzda:

```
<?php  
  
$arr = new Arr;  
  
echo $arr->getSum(); // xato vujudga keladi!  
?>
```

Bunday kod xatolik beradi! Nima uchun: sababi **array_sum** **\$this->numbers** massividagi sonlarning yig'indisini topishga urunadi, lekin u yerda hech qanday element mavjud emas.

Sababi, biz numbers xossasiga hech nima yozmadik, u yerda null qiymat yotibdi va biz aslida array_sum(null) kodini chaqiryapmiz, keyin albatta xato bo'ladida ☺. Array_sum funksiyasi bizdan massiv kutsa-yu, biz unga mana senga deb null qiymat bersak-a? ☺

Keling, bu xatolikni to'g'irlaymiz, numbers ni bo'sh massiv sifatida e'lon qilamiz:

```
<?php

class Arr

{

    private $numbers =


        [] ; // bu xossaga boshlang'ich qiymat sifatida
bo'sh massiv beramiz, ya'ni []


    public function add($num)

    {

        $this->numbers[] = $num;

    }


    public function getSum()

    {

        return array_sum($this->numbers);

    }

}

?>
```

Tekshiramiz:

```
<?php
```

```
$arr = new Arr;  
  
echo $arr->getSum(); // natija 0  
  
?>
```

O'zgaruvchan xossa nomlari

Aytaylik, bizda **User** nomli sinf bor:

```
<?php  
  
class User  
  
{  
  
    public $name;  
  
    public $age;  
  
  
    public function __construct($name, $age)  
    {  
  
        $this->name = $name;  
  
        $this->age = $age;  
  
    }  
  
}  
  
  
$user = new User('Sanjar', 21);  
  
echo $user->name; // natija 'Sanjar'  
  
?>
```

Ushbu sinf misolida biz siz bilan hozir xossa nomlarini o'zgaruvchilarda saqlash mumkinligini ko'rib chiqamiz.

Misol uchun, bizda 'name' satri mavjud bo'lgan \$prop o'zgaruvchisi mavjud deylik.

Unda biz \$user->\$prop ravishda murojaat qilishimiz mumkin. Aslida bu \$user->name bilan ekvivalent ya'ni bir xil. Bunday xossaga ayyorona murojaat onda-sonda ishlataladi, lekin ba'zida kerak bo'lib qolishi mumkin. Namunada ko'rib chiqamiz:

```
<?php  
  
$user = new User('Sanjar', 21);  
  
  
$prop = 'name';  
  
echo $user->$prop; // natija 'Sanjar'  
  
?>
```

Xossalari massivi

Endi, mana bunday shakldagi **User** sinfi berilgan bo'lsin:

```
<?php  
  
class User  
  
{  
  
    public $surname; // familiya  
  
    public $name; // ism  
  
    public $patronymic; // sharif  
  
  
    public function __construct($surname, $name,  
$patronymic)  
  
    {  
  
        $this->surname = $surname;  
  
        $this->name = $name;  
  
        $this->patronymic = $patronymic;  
  
    }  
}
```

```
}
```

```
?>
```

Va xossalari massivi berilgan bo'lzin:

```
<?php
```

```
$props = ['surname', 'name', 'patronymic'];
```

```
?>
```

Endi, nolinchini o'rinda saqlanayotgan massiv elementining qiymatini chiqarib ko'ramiz. Aslida, agar **\$user->\$props[0]** shaklida yozadigan bo'lsa, bu ishlamaydi:

```
<?php
```

```
$user = new User('Sobirjonov', 'Sanjar', 'Safarovich');
```



```
$props = ['surname', 'name', 'patronymic'];
```

```
echo $user->$props[0];
```

```
    // bu ishlamaydi
```

```
?>
```

Bunday murakkab tarzda berilgan xossaning nomi ishlashi uchun uni jingalak qavsga olishimiz kerak, mana bunday:

```
<?php
```

```
$user = new User('Sobirjonov', 'Sanjar', 'Safarovich');
```



```
$props = ['surname', 'name', 'patronymic'];
```

```
echo $user->{$props[0]};
```

```
    // bu ishlaydi
```

```
?>
```

Bog'langan massivda xossa nomlari

Berilgan massiv, bog'langan massiv bo'lishi ham mumkin:

```
<?php

$user = new User('Sobirjonov', 'Sanjar', 'Safarovich');

$props = [ 'prop1' => 'surname', 'prop2' =>
    'name', 'prop3' => 'patronymic'];

echo $user->{$props['prop1']} ;

// bu ishlaydi

?>
```

Funksiyadagi xossa nomi

Xossa nomini funksiyadan ham olishimiz mumkin:

```
<?php

function getProp()
{
    return 'surname';
}

$user = new User('Sobirjonov', 'Sanjar', 'Safarovich');

echo $user->{getProp()} ;

// bu ishlaydi

?>
```

Shunday qilib, biz misol uchun, **Worker** sinfning value xossasiga **Student** sinfimizning age xossasining nomini berib qo'yishimiz mumkin. Va ikkala sinfni obyektini yaratib,

Worker sinfining value xossasidagi qiymatni Student sifning age xossasini o'rniliga ishlatsishimiz mumkin.

Metodlarning o'zgaruvchan nomlari

Xossa nomlarini o'zgaruvchilarda saqlaganimiz kabi, metod nomlarini ham o'zgaruvchida saqlashimiz mumkin. Keling, buni namunada ko'rib chiqamiz. Getter xossali **User** sinfi berilgan bo'lsin:

```
<?php

class User

{

    private $name;

    private $age;

    public function __construct($name, $age)

    {

        $this->name = $name;

        $this->age = $age;

    }

    public function getName()

    {

        return $this->name;

    }

    public function getAge()

    {

        return $this->age;

    }

}
```

```
}
```

```
}
```

```
?>
```

\$method o'zgaruvchisiga metod nomini saqlaymiz. Keling, endi metodni bunday usulda chaqirib ko'ramiz:

```
<?php
```

```
$user = new User('Sanjar', 21);
```



```
$method = 'getName';
```

```
echo $user->$method(); // natija 'Sanjar'
```

```
?>
```

Agar metod nomi massivdan olinadigan bo'lsa, metodga murojaatda biz uni jingalak qavslarga olishimiz kerak bo'ladi:

```
<?php
```

```
$user = new User('Sanjar', 21);
```



```
$methods = ['getName', 'getAge'];
```

```
echo $user->{$methods[0]}(); // natija 'Sanjar'
```

```
?>
```

Obyekt yaratilishi bilanoq metodni chaqirish

O'zida sonlardan iborat bo'lgan massiv va ushbu sonlarning yig'indisini topadigan **getSum** metodli **Arr** sinfi berilgan bo'lsin. Raqamlar obyektning konstruktor orqali massiv ko'rinishida kiritiladi, hamda **add** metodi orqali bittalab qo'shish ham mumkin:

```
<?php
```

```
class Arr
```

```
{  
  
    private $numbers = [];  
    // sonlar massivi  
  
    public function __construct($numbers)  
    {  
        $this->numbers = $numbers;  
    }  
  
    public function add($number)  
    {  
        $this->numbers[] = $number;  
    }  
  
    public function getSum()  
    {  
        return array_sum($this->numbers);  
    }  
}  
?>
```

Arr sinfining ishlatalishi bo'yicha namuna:

```
<?php  
  
$arr = new Arr([1, 2, 3]);  
  
$arr->add(4); // massivni oxiriga 4 raqam qo'shilyapti  
$arr->add(5); // massivni oxiriga 5 raqam qo'shilyapti  
  
// Massiv elementlari yigindisi
```

```
echo $arr->getSum(); // natija 15  
?>
```

Obyektni yaratganimizda unga kerakli sonlar massivini konstruktoriga uzatamiz, keyin darrov uning yig'indisini topamiz:

```
<?php  
  
$arr = new Arr([1, 2, 3]);  
  
echo $arr->getSum(); // natija 6  
?>
```

Agar biz obyekt bilan kelgusida hech qanday manipulatsiya qilishni rejalashtirmagan bo'lsak, unda yuqoridagi kodimizni yanada qisqaroq yozishimiz mumkin: obyektni yaratib darrov **getSum()** metodini chaqirish mumkin:

```
<?php  
  
echo (new Arr([1,  
              2, 3]))->getSum(); // natija 6  
?>
```

Metodlar zanjiri

O'zida sonlardan iborat bo'lgan massiv va ushbu sonlarning yig'indisini topadigan **getSum** metodli **Arr** sinfi berilgan bo'lsin. Raqamlar obyektning konstruktor orqali massiv ko'rinishida kiritiladi, hamda **add** metodi orqali bittalab qo'shish ham mumkin:

```
<?php  
  
class Arr  
  
{  
  
    private $numbers = []; // sonlar massivi
```

```
public function __construct($numbers)
{
    $this->numbers = $numbers;
}

public function add($number)
{
    $this->numbers[] = $number;
}

public function getSum()
{
    return array_sum($this->numbers);
}

?>
```

Arr sinfining ishlatalishi bo'yicha namuna:

```
<?php

$arr = new Arr([1, 2, 3]);

$arr->add(4); // massivni oxiriga 4 raqam qo'shilyapti
$arr->add(5); // massivni oxiriga 5 raqam qo'shilyapti

// Massiv elementlari yigindisi
echo $arr->getSum(); // natija 15

?>
```

Endi biz har bir sonni alohida qo'shib o'tirishni xohlamaymizda, aksincha bitta qatorni o'zida hammasini kiritmoqchimiz, mana bunday:

```
<?php  
  
$arr = new Arr;  
  
echo $arr->add(1)->add(2)->add(3)->getSum(); // hozir bu  
ishlamaydi chunki kodimiz moslashtirilmagan  
  
?>
```

Mana shunday zanjirni hosil qilish uchun unda qatnashayotgan barcha metod **\$this** qaytarishi kerak.

Bu qanday ishlaydi: \$arr->add(1) \$thisga teng bo'ladi, ya'ni \$arr ni o'ziga. Va har bir metoddagi qiymat aynan o'sha sinfga teng shuningdek, qiymatlar ham saqlangan holatda bo'ladi. Ya'ni bunday zanjir:

```
<?php  
  
echo $arr->add(1)->add(2)->add(3);  
  
?>
```

Aslida bunga teng:

```
<?php  
  
$arr->add(1); $arr->add(2); $arr->add(3);  
  
?>
```

Yanada qisqaroq yozishimiz ham mumkin:

```
<?php  
  
echo (new Arr)->add(1)->add(2)->  
    add(3)->getSum(); // natiaj 6  
  
?>
```

Yodda tuting: zanjirda hozir bizdagidek faqat bitta metod qatnashmasligi mumkin albatta, u yerda turli xil metodlar bo'lishi mumkin. Lekin eng asosiysi ular **\$this** qaytarsa bo'lgani. Ishlaydi.

Topshiriq 1:

Kodga qaramasdan, zanjirli ishlatilishi mumki bo'lgan Arr sinfini mustaqil ishlab chiqing.

Topshiriq 2:

Arr sinfiga yana bir, sonlar saqlanadigan massivni oxiriga sonlar qo'shadigan append metodini qo'shing. Bu add va append metodlar turli xil tartibda ishlatilishi mumkin:

```
<?php  
echo (new Arr)->  
    add(1)->append([2, 3, 4])->add(5)->getSum();  
?>
```

Topshiriq 3:

surname(familiya), **name**(ism) va **patronymic**(sharif) xossalni **User** sinfini yarating. Bu xossalarga qiymatlar setterlar orqali berilsin. Shunday qilingki, ushbu setterlar turli xil tartibda zanjir ko'rinishida chaqirilsin va ushbu zanjirning oxirida foydalanuvchining F.I.Shini qaytaradigan **getFullName** metodini chaqiring:

```
<?php  
echo (new User)->  
    setName('Sanjar')->setPatronymic('Safarovich')  
    ->setSurname('Sobirjonov')->
```

```
?> getFullName(); // natija 'SSS'
```

Sinf metodlar to'plami sifatida

Ko'pincha sinflar birgalikda guruhlangan ba'zi metodlar to'plami sifatida ishlataladi. Bunday holatda bizga ushbu sinfning bir nechta obyektlarini yaratish kerak emas, bittasi yetadi.

Misol uchun, massivlar bilan ishlash uchun metodlar to'plamini taqdim etgan **ArraySumHelper** sinfini yaratamiz. Sinfimizning har bir metodi parametr qabul qiladi va nimadir vazifani bajarib, natija qaytaradi. Keling, bizda quyidagi metodlar mavjud bo'lsin:

```
<?php  
  
class ArraySumHelper  
{  
  
    // massiv elementlarning yig'indisi:  
    public function getSum1($arr)  
    {  
  
    }  
  
    // massiv elementlarining kvadrati yig'indisi:  
    public function getSum2($arr)  
    {  
  
    }  
  
    // msasiv elementlarining kubi yig'indisi:  
    public function getSum3($arr)
```

```
{  
  
}  
// massiv elementlarining 4chi darajasi yig'indisi  
public function getSum4($arr)  
{  
  
}  
  
}  
?  

```

Endi sinfimizdan qanday foydalanishni ko'rib chiqaylik:

```
<?php  
  
$arraySumHelper =  
    new ArraySumHelper; // obyekt yaratamiz  
  
  
$arr = [1, 2, 3];  
  
echo $arraySumHelper->  
    getSum1($arr); // elementlar yig'indisini topamiz  
  
echo $arraySumHelper->  
    getSum2($arr); // elementlarning kvadrati  
    yig'indisini topamiz  
  
echo $arraySumHelper->  
    getSum3($arr); // elementlarning kub yig'indisin  
    topamiz  
  
echo $arraySumHelper->getSum4($arr); // elementlarning  
4chi darajasi yig'indisini topamiz  
  
?>
```

Mana, yana bir namuna – massiv elementlarining kvadrat yig'indisini hamda kubining yig'indisini topamiz va ikkovilarini qo'shamiz:

```
<?php  
  
$arraySumHelper = new ArraySumHelper;  
  
  
$arr = [1, 2, 3];  
  
echo $arraySumHelper->getSum2($arr) + $arraySumHelper-  
>getSum3($arr);  
  
?>
```

Ya'ni, biz aslida bitta sinfdagi guruhlangan metodlar to'plamini olayapmiz. Lekin, odatiy metodlar to'plamidan farqi, biz **OOP**(o'zb. OYD)ning qulayliklaridan foydalanishimiz mumkin - misol uchun, yordamchi funksiyani sinfdan tashqarida mavjud bo'lmasligi uchun uni yopiq ya'ni maxfiy qilib e'lon qilamiz. Keling, sinfimizning yozilgan kodiga e'tibor qaratamiz. Agar metodlarni yaratish bo'yicha fikrlaydigan bo'lsak, bitta narsa aniqki, ular aslida aynan bitta, lekin ularning farqi darajalarida ekanligini bilamiz. Kodga qarang:

```
<?php  
  
class ArraySumHelper  
{  
  
    public function getSum1($arr)  
    {  
  
        $sum = 0;
```

```
foreach ($arr as $elem) {  
    $sum += $elem; // elementning 1 chi darajasi  
    yig'indisi  
}  
  
return $sum;  
}  
  
  
public function getSum2($arr)  
{  
    $sum = 0;  
  
    foreach ($arr as $elem) {  
        $sum += pow($elem,  
                    2); // elementning 2 chi darajasi  
        yig'indisi  
    }
  
  
    return $sum;  
}
  
  
  
public function getSum3($arr)  
{  
    $sum = 0;  
  
    foreach ($arr as $elem) {  
        $sum += pow($elem,
```

```
            3); // elementning 3 chi darajasi
yig'indisi

        }

return $sum;

}

public function getSum4($arr)
{
    $sum = 0;

    foreach ($arr as $elem) {
        $sum += pow($elem,
            4); // elementning 4 chi darajasi
yig'indisi

    }

return $sum;

}

?>
```

Har bir metodni alohida yaratmasdan, keling yaxshisi bitta, daraja hamda massiv parametrlar qabul qiladigan hamda massiv elementlarining darajasi yig'indisini qaytaradigan maxfiy **getSum** metodini yaratamiz:

```
<?php
private function getSum($arr, $power) {
```

```
$sum = 0;

foreach ($arr as $elem) {

    $sum += pow($elem, $power);

}

return $sum;

}

?>
```

Endi esa sinfimizning boshqa metodlarini **getSum** metodidan foydalanib o'zgartirib chiqamiz:

```
<?php

class ArraySumHelper

{

    public function getSum1($arr)

    {

        return $this->getSum($arr, 1);

    }

    public function getSum2($arr)

    {

        return $this->getSum($arr, 2);

    }

    public function getSum3($arr)

    {
```

```
        return $this->getSum($arr, 3);  
    }  
  
    public function getSum4 ($arr)  
    {  
        return $this->getSum($arr, 4);  
    }  
  
    private function getSum($arr, $power) {  
        $sum = 0;  
  
        foreach ($arr as $elem) {  
            $sum += pow($elem, $power);  
        }  
  
        return $sum;  
    }  
}  
?>
```

Bizning **ArraySumHelper** sinfimiz avvalgidan ancha o'qimishli, ya'ni inson tushunadigan ko'rinishga aylandi. Ammo bitta tamoyilni tushunishingiz muhim – ba'zi sinflarni shunchaki metodlar to'plami sifatida ishlatsishimiz mumkin aynan hozirgidek va bunda sinfning faqat bitta obyekti yaratiladi. Keyingi darslarimizda yanada hayotiy misollar bilan buni tushuntirishga harakat qilaman.

Topshiriq :

Quyida ArrayAvgHelper sinfining metodlar qolipi berilgan.
Ushbu metodlarni vazifasiga ko'ra davom ettiring.

```
<?php

class ArraySumHelper

{

    /*
        Massiv elementlarining birinchi
        darajasi yig'indisini topadi
    */

    public function getAvg1($arr)
    {

    }

    /*
        Massiv elementlarining ikkinchi darajasi
        yig'indisini topadi va uning kvadrat ildizni
        chiqaradi:
    */

    public function getAvg2($arr)
    {

    }

    /*
        Massiv elementlarining uchinchi darajasi
    
```

```
    yig'indisini topadi va uning kub ildizni  
    chiqaradi:  
  
    */  
  
    public function getAvg3($arr)  
  
    {  
  
    }  
  
/*  
Massiv elementlarining to'rtinchi darajasi  
yig'indisini topadi va uning to'rtinchi ildizni  
chiqaradi:  
  
*/  
  
public function getAvg4($arr)  
  
{  
  
}  
  
/*  
Massiv va daraja parametrlarini qabul qilib  
massiv elementlarining darajalari yig'indisini  
chiqaradigan yordamchi metod  
  
*/  
  
private function getSum($arr, $power)  
  
{  
  
}
```

```
/*
    Butun son va daraja parametrlarini qabul qilib
    sonning berilgan darajadagi ildizini chiqaradi
*/
private function calcSqrt($num, $power)
{
}

?>
```

Matematik yordam: birinchi darajaning ildizi – bu o'sha sonning o'zi. Ya'ni, **calcSqrt**(son,1) shunchaki o'sha sonni o'zini qaytarishi kerak.

A har qanday darajaning ildizini esa **pow** funksiyasi yordamida topish mumkin, shunchaki unga parametr sifatida slash qo'shish orqali. Misol uchun, **pow(son, 1/3)** – 3 darajaning ildizini topib beradi.

Sinfdan meros olish

Tasavvur qiling, bizda **User** sinfi mavjud. U sizga qandaydir maqsadlarda ishlatalish uchun kerak va u sinf sizni to'laqonli qoniqtiradi – ortiqcha o'zgartirishlar shart emas. O'sha sinf, mana:

```
<?php
class User
{
    private $name;
```

```
private $age;

public function getName()
{
    return $this->name;
}

public function setName($name)
{
    $this->name = $name;
}

public function getAge()
{
    return $this->age;
}

public function setAge($age)
{
    $this->age = $age;
}

?>
```

Endi, tasavvur qiling bizga **Employee** sinfini ham yaratishimiz kerak. **Employee** sinfining ma'nosi xodim degan ma'noni anglatadi. Xodim , foydalanuvchi(user) sinfiga juda o'xshash, bir xil metod va xossalarga ega, lekin unga **salary** ya'ni maosh

xossasini qo'shish kerak, shuningdek, unga muvofiq **getter** va **setter** ham:

```
<?php

class Employee

{

    private $name;

    private $age;

    private $salary; // зарплата


    public function getSalary()

    {

        return $this->salary;

    }


    public function setSalary($salary)

    {

        $this->salary = $salary;

    }


    public function getName()

    {

        return $this->age;

    }


    public function setName($name)

    {
```

```
    $this->name = $name;  
}  
  
public function getAge()  
{  
    return $this->age;  
}  
  
public function setAge($age)  
{  
    $this->age = $age;  
}  
}  
?>
```

Ko'rib turganimizdek, **User** va **Employee** sinflari amaliyot jihatdan bir-biriga juda mos. Sinflarning umumiy qismini bir joyga yozib qo'ysak, nur ustiga a'lo nur bo'lardi.

Agar bu haqida yaxshilab o'ylab ko'rsangiz, **Employee** sinfi bu **User** sinfi, lekin kengaytirilganligini anglaysiz.

Meros olish yordamida, biz **Employee** sinfimizni **User** sinfining xossa va metodlaridan foydalanishga majburlaymiz va shunchaki **User** sinfida mavjud bo'lмаган metod va xossalar bo'lsa, ya'ni **Employee** ga oid bo'lган, unda yozib qo'yamiz.

Meros olish **extends** kalit so'zi yordamida amalga oshiriladi. **Employee** sinfi, **User** sinfidan meros olishi uchun, uni e'lon

qilayotganda **class Employee** deb yozishni o'rniga, **class Employee extends User** deb yozishimiz kerak.

Meros berayotgan sinf – **ota sinf** deyiladi va meros olayotgan sinf esa **avlod sinf** deyiladi.

Avlod sinf faqatgina ommaviy, ya'ni ochiq zonadagi metod va xossalarni meros qilib ola oladi, maxfiy, yopiqlarini esa yo'q. Keling, **Employee** sinfimizga **User** sinfimizdan meros olib beramiz. Kodimiz ancha qisqaroq bo'ladi:

```
<?php

class Employee extends User

{
    private $salary;

    public function getSalary()
    {
        return $this->salary;
    }

    public function setSalary($salary)
    {
        $this->salary = $salary;
    }
}

?>
```

Yangi Employee sinfimizning ishlashini tekshiramiz:

```
<?php

$employee = new Employee;

$employee->setSalary(
    1000); // Employee sinfi metodi

$employee->setName(
    'Sanjar'); // ota sinfdan olingan metod

$employee->setAge(25); // ota sinfdan olingan metod


echo $employee->
    getSalary(); // Employee sinfi metodi

echo $employee->
    getName(); // ota sinfdan olingan metod

echo $employee->
    getAge(); // ota sinfdan olingan metod

?>
```

Quyidagiga e'tibor bering: avlod sinf, ota sinfning **name** va **age** yopiq xossalariini meros qilib olmadi, unga murojaat esa xatolikka sabab bo'ladi. Lekin, avlod sinfda **name** va **age** xossalarning **getter** va setterlari mavjud, sababi ular ochiq zonada belgilangan.

Topshiriq 1:

User va Employee sinflarini mustaqil ravishda yozib chiqing.

Bir nechta avlod sinflar

Meros olishning foydali jihatlaridan biri bu har bir sinfning ko'pgina avlodlari bo'lishi mumkin. Namunaga qarang.

Faraz qiling, employee sinfiga qo'shimcha sifatida yana Student sinfini ham qo'shmoqchimiz, keling buni ham User sinfidan meros olib beramiz:

```
<?php

class Student extends User

{

    private $course; // kurs


    public function getCourse()

    {

        return $this->course;

    }

    public function setCourse($course)

    {

        $this->course = $course;

    }

}

?>

#####
#####

<?php

$student = new Student;



$student->setCourse(3); // Student sinfi metodi

$student->setName('Sanjar');

// ota sinfdan olingan metod
```

```
$student->setAge(25); // ota sinfdan olingan metod

echo $student->

    getCourse(); // Student sinfdan olingan metod

echo $student->

    getName(); // ota sinfdan olingan metod

echo $student->getAge(); // ota sinfdan olingan metod

?>
```

Topshiriq 2:

Kodga qaramadan, Student hamda User sinflarini ishlab chiqing.

Avlod sinfdan meros olish

Aytaylik bizda ota sinf hamda avlod sinf mavjud. Ushbu avlod sinfdan ham boshqa sinfga meros olib berish mumkin, uning avlod sinfidan esa boshqa sinfga va hakozo... Namunada ko'ramiz.

Student sinfidan **StudentBSU** sinfiga meros olib beramiz, deylik:

```
<?php

class StudentBSU extends Student

{

    // kod

}

?>
```

Student sinf **User** sinfidan meros oladi, **StudentBSU** o'z navbatida **Student** sinfidan meros oladi. **StudentBSU** ham endi qanaqadir sinfga meros berishi ham mumkin...

Topshiriq 3:

Employee(xodim) sinfidan meros oladigan **Programmer** sinfini yarating. Yangi sinfimizda, dasturchi o'zi biladigan tillar massivini saqlaydigan **langs** xossasi bo'lsin. Shuningdek, ushbu xossa uchun getter va setterlarni ham yarating.

Topshiriq 4:

Employee sinfidan meros oladigan **Driver**(haydovchi) sinfini yarating. Yangi sinfga quyidagi xossalarni qo'shing: **haydovchilik stoji**, **haydovchi kategoriyasi(A,B,C,D)** hamda ular uchun getter hamda setterlarini ham yarating.

protected – kirish modifikatori

Avvalgi darslardan bizga ma'lumki, yopiq xossalarni va metodlar boshqa sinfga meros bo'lib o'tmaydi.

Agar biz metod yoki xossani avlod sinfda ham mavjud bo'lishini xohlasak, siz ularni public sifatida belgilashingiz kerak. Muammo bor: ochiq xossa va metodlar sinfning tashqarisida ham mavjud bo'ladi, biz xohlamagan taqdirda ham. Boshqacha qilib aytganda, biz ota sinf o'zining ba'zi metod va xossalarni avlod sinfga meros qilib berisin, lekin sinfning tashqarida esa ular ko'rinxas, ya'ni yopiq bo'lsin.

Muammoni yechish uchun maxsus protected modifikator mavjud va aynan shu modifikator bizga qo'l keladi. Keling, uning ishlashini real misollarda ko'rib chiqamiz.

name va **age** yopiq xossalariiga ega User sinfi berilgan bo'lsin:

```
<?php  
  
class User  
  
{
```

```
private $name;  
private $age;  
  
public function getName()  
{  
    return $this->name;  
}  
  
public function setName($name)  
{  
    $this->name = $name;  
}  
  
public function getAge()  
{  
    return $this->age;  
}  
  
public function setAge($age)  
{  
    $this->age = $age;  
}  
}  
?>
```

User sinfidan Student sinfi meros olsin:

```
<?php
```

```
class Student extends User

{
    private $course;

    public function getCourse()
    {
        return $this->course;
    }

    public function setCourse($course)
    {
        $this->course = $course;
    }
}

?>
```

Hozircha barchasi a'lo va yaxshi ishlayapti.

Endi Student sinfiga age xossasiga 1 yil qo'shadigan addOneYear metodini qo'shib ko'ramiz. Keling, bu metodni yaratilishi bilan tarnishing:

```
<?php

class Student extends User
{
    private $course;

    // Metod

    public function addOneYear()
```

```
{  
    $this->age++;  
  
}  
  
public function getCourse()  
{  
    return $this->course;  
}  
  
public function setCourse($course)  
{  
    $this->course = $course;  
}  
}  
?>
```

Muammo nimada? Muammo, agar age xossasini yopiq deb qoldirsak, unda biz avlod sinfdan murojaat qilmoqchi bo'lib, addOneYear metodini chaqiradigan bo'lsak xatolik chiqadi:

```
<?php  
  
$student = new Student();  
  
$student->setAge(25);  
  
$student->addOneYear(); // xatolik chiqadi  
?>
```

Xatolikni bartaraf etish uchun User sinfidagi age xossani private emas balki protected sifatida e'lon qilib qo'yamiz:

```
<?php

class User

{

    private $name;

    protected $age;

        // protected deb belgilaymiz

    ...

}

?>
```

Endi avlod sinfning **addOneYear** metodi **age** xossasini bexato o'zgartiradi lekin u sinfimizning tashqarisida mavjud bo'lmaydi. Keling, hammasini birlashtiramiz:

```
<?php

class User

{

    private $name;

    protected $age; // protected deb belgilaymiz


    public function getName()

    {

        return $this->name;

    }


    public function setName($name)

    {

        $this->name = $name;

    }

}
```

```
public function getAge()

{
    return $this->age;
}

public function setAge($age)
{
    $this->age = $age;
}

class Student extends User

{
    private $course;

    // yoshga 1 qo'shish uchun metod
    public function addOneYear()
    {
        $this->age++;
    }

    public function getCourse()
    {
        return $this->course;
    }
}
```

```
public function setCourse($course)
{
    $this->course = $course;
}

?>
```

Student sinfining ishlashini tekshiramiz:

```
<?php

$student = new Student();

$student->setName('Коля'); // ism kiritamiz
$student->setCourse(3); // kursni kiritamiz
$student->setAge(25); // yoshni 25ga o'rnatamiz

$student->addOneYear(); // yoshni birlikka oshiramiz
echo $student->getAge(); // natija 26

?>
```

Bizga muhim va kerakli bo'lgan narsa bu age xossasiga murojaat bo'lsa(sinfdan tashqarisida) xatolik chiqarishi kerak :

```
<?php

$student = new Student();

$student->age = 30; // xatolik

?>
```

☺Hammasi ajoyib, eng asosiysi, bizga kerakli bo'lgani ishlayapti.

Avlod sinfda ota sinfning metodlarini qayta yozish

Yopiq **name** va **age** xossali **User** sinfi berilgan, shu jumladan ularning **getter** va **setterlari** ham. Yoshning **setter** kiritilgan yoshni 18 ga teng yoki kattaligini tekshiradi:

```
<?php

class User

{

    private $name;

    private $age;

    public function getName()

    {

        return $this->name;

    }

    public function setName($name)

    {

        $this->name = $name;

    }

    public function getAge()

    {

        return $this->age;

    }

    public function setAge($age)

    {
```

```
// tekshiruv

if ($age >= 18) {

    $this->age = $age;

}

}

?>
```

User sinfidan yopiq course xossaga ega bo'lgan va aynan shu xossaga tegishli bo'lgan **setter** hamda **getterli** Student sinfi meros oladi:

```
<?php

class Student extends User

{

    private $course;

    public function getCourse()

    {

        return $this->course;

    }

    public function setCourse($course)

    {

        $this->course = $course;

    }

}

?>
```

Endi tasavvur qilamiz, **Student** sinfi **User** sinfidan **setAge** metodini meros qilib olayapti, bizga mos kelmaydigan jihatni, bu **setAge** metodi yoshni 18 likkga tekshiradi, bizga esa 25 likni tekshirishi kerak.

Ya'ni, yosh agar 18 ga teng bo'lsa yoki katta bo'lsa ota sinfdagi **setAge** metodni bu qanoatlantiryapti, lekin biz unga qo'shimcha shart ham kiritmoqchimiz: ham 25dan kichik bo'lsa degan.

Muammoni yechish uchun, PHP da avlod sinfda aynan shu masalada, ota sinfdagi metodni o'ziga moslab qayta yozish imkonи mavjud. Bu esa bizga kerakli bo'lgan narsa. Shunday qilib, keling **Student** sinfida **setAge** metodini yozamiz. Bizning setAge yoshni 18 dan katta 25dan kichikligini tekshiradi:

```
<?php

class Student extends User

{
    private $course;

    // ota sinfni metodini qayta yozamiz
    public function setAge($age)
    {
        if ($age >= 18 and $age <= 25) {
            $this->age = $age;
        }
    }
}
```

```
public function getCourse()  
{  
    return $this->course;  
}  
  
public function setCourse($course)  
{  
    $this->course = $course;  
}  
}  
?>
```

Tegishli ravishda, **setAge** ishlatalayotgan ota sinfdagi **age** xossani **protected**(himoyalangan) deb e'lon qilib qo'yishimiz kerak:

```
<?php  
  
class User  
{  
  
    private $name;  
  
    protected $age;  
        // protected  
  
  
    public function getName()  
    {  
        return $this->name;  
    }  
  
  
    public function setName($name)
```

```
{  
    $this->name = $name;  
}  
  
public function getAge()  
{  
    return $this->age;  
}  
  
public function setAge($age)  
{  
    if ($age >= 18) {  
        $this->age = $age;  
    }  
}  
}  
?>
```

Keling endi qayta belgilangan **setAge** metodini ishlashini tekshirib ko'ramiz:

```
<?php  
$student = new Student;  
  
$student->setAge(24); // to'g'ri yoshni ko'rsatamiz  
echo $student->getAge(); // yosh o'zgardi - 24  
  
$student->setAge(30); // noto'g'ri yoshni ko'rsatamiz
```

```
echo $student->getAge(); // yosh o'zgarmadi - 24 natija  
chiqdi  
?>
```

Parent bilan ishlash

Hozir biz siz bilan **setAge** metodida yoshni 18dan 25gachaligini tekshirdik.

Lekin, 18 ga to'lganligini tekshirishni **setAge** metod ham amalga oshiradi. Bu shuni anglatadiki, agar biz yoshning chegarasini o'zgartirmoqchi bo'lsak, bizga uni 2ta joyda qilishga to'g'ri keladi: ota sinfda va avlod sinfda.

Agar avlod sinf, ota sinfdagi **setAgedan** foydalananayotgan bo'lsa, bu oson bo'ladi. Chunki, ota sinfdagi u metod bizga mos keladigan kodning ma'lum qismini o'z ichiga olgan va bu kodni biz avlod sinfda takrorlamoqchi emasmiz.

Buni biz ota sinfda ishora qiluvchi **parent** kalit so'zi yordamida amalga oshirolamiz.

U yordamida ota sinf metodiga quyidagicha kirish mumkin: **parent::setAge()**, ya'ni parent kalit so'zidan keyin ikki nuqta va metodning nomi. Keling, Student sinfimizni qayta yozamiz:

```
<?php  
  
class Student extends User  
{  
  
    private $course;  
  
  
    public function setAge($age)  
    {  
  
        // agar yosh 25ga teng yoki katta bo'lsa
```

```
if ($age <= 25) {  
    // ota sinfdagi ushbu metodni chaqiramiz  
    parent::setAge($age); // ota sinf esa buni  
18ga to'lganligini tekshiradi  
}  
  
}  
  
public function getCourse()  
{  
    return $this->course;  
}  
  
public function setCourse($course)  
{  
    $this->course = $course;  
}  
}  
?>
```

Biz o'zimiz xohlagan ishni bajaroldik. Undan tashqari, avlod sinfdagi **setAge** metodi age xossasini to'g'ridan-to'g'ri foydalanmayapti. Bu shuni anglatadiki, ota sinfdagi **age** xossaning kirish modifikatorini **protected** emas balki **private** ga qaytib o'zgartirib qo'yishimiz mumkin.

Topshiriq 1:

Hozirgi User sinfimizdagi setAge metodini ismdagi belgilar soni 3 tadan kam emasligini tekshiradigan qilib o'zgartiring.

Topshiriq 2:

Student sinfida setName metodini ismdagi belgilar soni 3 tadan kam va 10 tadan ko'p ekanligini tekshiring.

Student sinfning **setName** metodi ota sinfning **setName** metodidagi tekshiruvni takrorlamasin.

Avlod sinfda ota sinfning konstruktorini qayta yozish

Konstruktorda kiritiladigan name va age va kelgusida faqat o'qish uchun(ya'ni, maxfiy xossalarning faqat getteri bo'ladi, setter esa yo'q) mavjud bo'lgan xossaga ega **User** sinfi bor bo'lsin.

```
<?php  
  
class User  
  
{  
  
    private $name;  
  
    private $age;  
  
  
    public function __construct($name, $age)  
    {  
        $this->name = $name;  
        $this->age = $age;  
    }  
  
  
    public function getName()  
    {  
        return $this->name;  
    }  
}
```

```
}

public function getAge()
{
    return $this->age;
}

?>
```

Ushbu sinfdan **Student** sinfi meros oladi:

```
<?php

class Student extends User
{
    private $course;

    public function getCourse()
    {
        return $this->course;
    }
}

?>
```

Avlod sinfning o'zini konstruktori bo'lmaydi – bu sinf obyekti yaratilganda ota sinf obyekti ishlashini anglatadi:

```
<?php

$student = new Student(
    'Sanjar',    19); // ota sinfning konstruktori
ishlatilayapti
```

```
echo $student->  
      getName(); // natija 'Sanjar'  
echo $student->getAge(); // natija 19  
?>
```

Hammasi zo'r, lekin bitta muammo bor: Student sinfining obyektini yaratganimizda biz uning konstruktoriga uchinchi parametr ham kiritmoqchimiz, ya'ni nechanchi kursligini, mana bunday:

```
<?php  
$student = new Student(  
    'Sanjar', 19, 2); // bu hozircha ishlamaydi  
?>
```

Buni qilishning eng sodda usuli bu ota sinfning konstruktoridagi kodni olib o'zingizni konstruktoringiz bilan birga qayta belgilash:

```
<?php  
class Student extends User  
{  
    private $course;  
  
    // obyekt konstruktori:  
    public function  
        __construct($name, $age, $course)  
    {  
        // ota sinfning konstruktorini ko'paytiramiz:  
        $this->name = $name;  
        $this->age = $age;
```

```
// Kodimiz:  
$this->course = $course;  
}  
  
public function getCourse()  
{  
    return $this->course;  
}  
}  
?>
```

Bunda avlod sinfda ota sinfning **name** va **age** maxfiy xossalariiga murojaat qilish, albatta biz xohlagandek ishlamaydi. Uni **protected** ga almashtiramiz:

```
<?php  
class User  
{  
    protected $name;  
        // xossani himoyalangan rejimda e'lon qilamiz  
    protected $age;  
        // xossani himoyalangan rejimda e'lon qilamiz  
  
    // Obyekt konstruktori:  
    public function __construct($name, $age)  
    {  
        $this->name = $name;  
        $this->age = $age;
```

```
}

public function getName()
{
    return $this->name;
}

public function getAge()
{
    return $this->age;
}

?>
```

Endi student obyektini yaratganimiz bilanoq, uchinchi, kurs parametrini uzatishimiz mumkin bo'ladi:

```
<?php

$student = new Student(
    'Sanjar', 19, 2); // endi bu ishlaydi

echo $student->
    getName(); // natija 'Sanjar'
echo $student->getAge(); // natija 19
echo $student->getCourse(); // natija 2

?>
```

Topshiriq 1:

User sinfidan meros oladigan Student sinfini, kodimga qaramasdan mustaqil ishlab chiqing.

Ota sinfning konstruktoridan foydalanamiz

Avlod sinfda, ota sinfning konstruktoridagi kodni nusxalab yozish, tushunarli bo'lib turganidek – unchalik yaxshi emas.

Keling, siz bilan avlod sinfda kodni dublirovka qilishni o'rniga, ota sinfning konstruktorini chaqirib qo'yamiz. Aniqroq qilish uchun, sizga qadamma-qadam tushuntiraman.

User sinfi konstruktori shunday bo'lsin: u 2ta, **name** va **age** parametrlarini qabul qiladi va tegishli xossalarga yozadi:

```
<?php  
  
// User sinfning konstruktori:  
  
public function __construct($name, $age)  
{  
  
    $this->name = $name;  
  
    $this->age = $age;  
  
}  
  
?>
```

Mana biz qayta yozmoqchi bo'lgan Student sinfining konstruktori:

```
<?php  
  
// Student sinfining konstruktori  
  
public function __construct($name, $age, $course)  
{
```

```
// Bu kodni ota sinfni konstruktorini chaqirish bilan  
o'zgartiramiz  
  
    $this->name = $name;  
  
    $this->age = $age;  
  
  
    // Kodimiz:  
  
    $this->course = $course;  
  
}  
  
?>
```

Avlod sinfning ichida ota sinfning konstruktorini qanday chaqirish mumkin? Allaqachon bilganingizdek, **parent** orqali. Ya'ni, *parent::__construct*.

Bunda ota sinfning konstruktori bиринчи parametr sifatida ismni, ikkinchi parametr sifatida esa yoshni qabul qilishni kutadi va biz uni mana bunday tarzda uzatishimiz kerak bo'lad: *parent::__construct(\$name, \$age);*.

Keling, buni amalda sinaymiz:

```
<?php  
  
// Student obyektining konstruktori:  
  
public function __construct($name, $age, $course)  
{  
  
    // Ota sinfning konstruktoriga 2ta  
    //parametr uzatib, uni chaqiramiz  
    parent::__construct($name, $age);  
  
  
    // course xossasini yozamizz:  
  
    $this->course = $course;
```

```
}
```

```
?>
```

Student sinfining to'liq kodini yozamiz:

```
<?php

class Student extends User

{

    private $course;

    // Obyekt konstruktori:

    public function

        __construct($name, $age, $course)

    {

        parent::__construct(

            $name, $age); // ota sinfning konstrukturini

        chaqiramiz

        $this->course = $course;

    }

    public function getCourse()

    {

        return $this->course;

    }

}
```

```
?>
```

Hammasi ishlayotganligini tekshirib ko'ramiz:

```
<?php
```

```
$student = new Student('Sanjar', 19, 2);

echo $student->
    getName(); // natija 'Sanjar'
echo $student->getAge(); // natija 19
echo $student->getCourse(); // natija 2
?>
```

Student sinfi endi ota sinfning **name** va **age** xossasiga to'g'ridan-to'g'ri murojaat qilmayotganligi tufayli, biz ularni yana maxfiy rejimda e'lon qilishimiz mumkin bo'ladi:

```
<?php

class User
{

    private $name; // maxfiy
    private $age; // maxfiy


    public function __construct($name, $age)
    {
        $this->name = $name;
        $this->age = $age;
    }

    public function getName()
    {
        return $this->name;
    }
}
```

```
public function getAge()  
{  
    return $this->age;  
}  
}  
?>
```

Topshiriq 2:

User nomli, quyidagi **name**(ism), **surname**(familiya) faqat o'qish uchun mavjud xossalarga ega bo'lgan sinf yarating. Ushbu xossalarning boshlang'ich qiymati konstruktorda o'rnatilsin. Shuningdek, ushbu xossalarning getterlarini ham yarating.

Topshiriq 3:

Shunday qiling, konstruktor uchinchi, xodimning tug'ilgan kunini yil-oy-kun formatda uzatadigan parametrga ega bo'lsin. Uni birthday xossasiga yozing. Ushbu xossa uchun getter ham yarating.

Topshiriq 4:

Tug'ilgan sanani qabul qiladigan **calculateAge** maxfiy metod yarating. Bu metodning vazifasi yosh bilan birga uning tug'ilgan kuni ushbu yilda bo'lib o'tdimi yoki yo'qligini qaytarsin.

Topshiriq 5:

Shunday qilingki, obyekt konstruktorida **calculateAge** chaqirilsin, foydalanuvchi yoshi hisoblanib uni age xossasiga yozib qo'ysin. Ushbu xossa uchun getter ham yaratilsin.

Obyektni havola orqali uzatish

Bizda mana bunday **User** sinfi berilgan bo'lsin:

```
<?php

class User

{

    public $name;

    public $age;

    public function __construct($name, $age)
    {
        $this->name = $name;
        $this->age = $age;
    }
}

?>
```

Biz ushbu sifning obyektini yaratamiz:

```
<?php

$user = new User('Sanjar', 30);

?>
```

Endi ushbu vaziyatni tasavvur qilib ko'ring: siz \$user o'zgaruvchining qiymatini qanaqadir o'zgaruvchiga belgilamoqchisiz, masalan \$test o'zgaruvchisiga.

Gap obyektlar emas, balki primitivlar haqida ketganda, ya'ni satrlar, raqamlar, massivlar va hakozolar, \$test o'zagruvchisiga \$user o'zgaruvchisining qiymatini nusxasi tushadi.

Nusxalash nimani anglatadi: bu, kelgusida biron bir o'zgaruvchiga o'zgartirish boshqa o'zgaruvchining qiymatini o'zgartirmaydi degan ma'noni anglatadi:

```
<?php  
$user = 1;  
  
$test = $user; //  
    $test o'zgaruvchisida endi 1  
  
$test = 2; // $test o'zgaruvchisida 2, lekin $user da hali  
hanuz 1  
?>
```

Obyektlar bilan esa hammasi boshqacha – obyektni boshqa o'zgaruvchiga yozish uni nusxalab qo'ymaydi, balki *havola orqali uzatadi*: bu, ushbu o'zgaruvchining qiymati, obyektning qiymati bilan proporsional. Biri o'zgarsa, ikkinchisiga ta'sir qiladi:

```
<?php  
$user = new User('Sanjar', 30);  
  
$test = $user; // $user va $test bitta obyektga murojaat  
qilayapti  
  
$test->name = 'Otabek'; // $test o'zgaruvchisini  
o'zagrtirdik  
  
//lekin $user ham o'zgardi  
  
// $user o'zgaruvchisining xossasini ko'ramiz  
echo $user->name; // natija 'Otabek'  
?>
```

Bir sinfni boshqa sinf ichida qo'llash

Shunday holat bo'ladiki, biz bir sinfning qandaydir metodini boshqa sinf ichida qo'llashimizga to'g'ri keladi lekin undan meros olishni xohlamaymiz.

Nima uchun meros olmaymiz?

Birinchidan, foydalanayotgan sinfimiz yordamchi bo'lishi va mantiqan bizning kodimiz ota sinf roliga mos tushmasligi mumkin.

Ikkinchidan, biz boshqa sinf ichida bir nechta sinfni ishlatalishimiz mumkin, lekin, meros olish mumkin emas. PHP da har bir sinfning otasi bitta bo'ladi.

Keling, endi amaliy namunada ko'rib chiqamiz. Bizda quyidagi Arr sinfi berilgan bo'lsin, obyektda biz add metodi orqali unga son qo'shishimiz mumkin:

```
<?php

class Arr

{

    private $nums = [];// sonlar massivi

    // Sonni massivga qo'shamiz:
    public function add($num)

    {

        $this->nums[] = $num;

    }

}

?>
```

Keling endi sinfimizga elementlarning kvadratlar yig'indisini topib va unga elementlarning kub yig'indisini qo'shadigan metod qo'shamiz.

Bizda **SumHelper** sinfi mavjud bo'lsin:

```
<?php

class SumHelper {

    // Kvadratlar yig'indisi:
    public function getSum2($arr)
    {
        return $this->getSum($arr, 2);
    }

    // Kublar yig'indisi:
    public function getSum3($arr)
    {
        return $this->getSum($arr, 3);
    }

    // Yig'indisini topadigan yordamchi funksiya:
    private function getSum($arr, $power) {
        $sum = 0;

        foreach ($arr as $elem) {
            $sum += pow($elem, $power);
        }
    }
}
```

```
        return $sum;  
    }  
}  
?>
```

Arr sinfidagi mavjud metodni qaytadan yana **SumHelperda** yaratish emas, balki uni **Arr** sinfidan olib ishlatish mantiqan to'g'ri keladi.

Buning uchun **Arr** sinfida konstruktor ichida **SumHelper** obyektini yaratamiz va uni **sumHelper** xossasiga yozamiz:

```
<?php  
  
class Arr  
  
{  
  
    private $nums = [];  
    // sonlar massivi  
  
    private $sumHelper;  
    // bu yerga SumHelper obyekti yoziladi  
  
    // Sinf konstruktori:  
    public function __construct()  
    {  
        // yordamchi sinfning obyektini yozamiz:  
        $this->sumHelper = new SumHelper;  
    }  
  
    // Massivga son qo'shamiz:  
    public function add($num)
```

```
{  
    $this->nums[] = $num;  
}  
}  
?>
```

Endi Arr ichida, SumHelper va uning ochiq xossa va metodlari mavjud bo'lgan \$this->sumHelper xossasi ishlatalishga tayyor. Arr sinfida endi getSum23 metodini yaratamiz, bu metod elementlarning kvadratlar yig'indisini topadi va uni elementlarning kub yig'indisiga qo'shamiz, SumHelper sinfining metodidan foydalanib:

```
<?php  
  
class Arr  
  
{  
  
    private $nums = [];  
    private $sumHelper;  
  
  
    public function __construct()  
    {  
        $this->sumHelper = new SumHelper;  
    }  
  
  
    public function getSum23()  
    {  
        $nums = $this->nums;  
  
        return $this->sumHelper->getSum2($nums) +
```

```
$this->sumHelper->getSum3($nums);  
}  
  
public function add($number)  
{  
    $this->nums[] = $number;  
}  
}  
?>
```

Keling endi yaratilgan Arr sinfidan foydalanib ko'ramiz:

```
<?php  
  
$arr = new Arr(); // obyekt yaratamiz  
  
$arr->add(1); // massivga 1 qo'shamiz  
$arr->add(2); // massivga 2 qo'shamiz  
$arr->add(3); // massivga 3 qo'shamiz  
  
// Kvadratlar yig'indisi hamda kublar yig'indisini  
topamiz:  
echo $arr->getSum23();  
?>
```

Obyektlarni taqqoslash

Hozir biz siz bilan obyektlarni == va === operatorlari yordamida taqqoslashni o'rjanamiz.

Siz primitivlar uchun(obyekt uchun emas) == operatori uni qiymati bo'yicha taqqoslasa, === operatori esa uni qiymati hamda turi bo'yicha taqqoslaydi:

```
<?php  
  
var_dump(3 == 3); // natija true  
var_dump(3 == '3'); // natija true  
  
var_dump(3 === 3); // natija true  
var_dump(3 === '3'); // natija false  
?>
```

Keling endi obyektlarni taqqoslash qanday bo'lishini ko'ramiz. 2ta obyektni taqqoslash uchun == operatoridan foydalanganimizda, obyektlarning xossalalarini taqqoslash amalga oshiriladi: agar ular bir xil xossaga va qiymatga teng bo'lsa va bir xil sinfning ekzempliyari bo'lsa, ular teng hisoblanadi.

==== operatoridan foydalanganimizda, tarkibida obyekt mavjud bo'lgan o'zgaruvchi, qachonki ular sinfning bir xil ekzempliyariga murojaat qilayotgan bo'lsagina ular teng bo'ladi.

Keling, endi ularni amaliyotda ko'ramiz:

```
<?php  
  
class User  
  
{  
    private $name;  
    private $age;
```

```
public function __construct($name, $age)
{
    $this->name = $name;
    $this->age = $age;
}

public function getName()
{
    return $this->name;
}

public function getAge()
{
    return $this->age;
}

?>
```

Sinfimizning 2ta xossalari bir xil qiymatga ega bo'lgan obyektini yaratamiz va ularni taqqoslaymiz:

```
<?php
$user1 = new User('Sanjar', 30);
$user2 = new User('Sanjar', 30);

var_dump($user1 == $user2); // natija true
?>
```

Endi, xossalari qiymati bir xil, lekin turli xil turda bo'lzin:

```
<?php  
  
$user1 = new User('Sanjar', 30);  
  
$user2 = new User('Sanjar', '30');  
  
  
var_dump($user1 == $user2); // natija true  
?>
```

Endi xossalari qiymati ham har xil bo'lsin:

```
<?php  
  
$user1 = new User('Sanjar', 25);  
  
$user2 = new User('Sanjar', '30');  
  
  
var_dump($user1 == $user2); // natija false  
?>
```

Keling, endi 2ta obyektni === orqali taqqoslab ko'ramiz:

```
<?php  
  
$user1 = new User('Sanjar', 30);  
  
$user2 = new User('Sanjar', 30);  
  
  
var_dump($user1 === $user2); // natija false  
?>
```

==== bilan taqqoslayotganimizda 2ta obyekt haqqatan ham teng bo'lishi uchun, ular bir xil obyektni ko'rsatishi kerak. Keling, ularni quyidagicha taqqoslaymiz:

```
<?php  
  
$user1 = new User('Sanjar', 30);  
  
$user2 = $user1;
```

```
var_dump($user1 === $user2); // natija true  
?>
```

Obyektni sinfga tegishlilagini aniqlash

Biz endi siz bilan **instanceof** operatori bilan tanishamiz. Ushbu operator joriy obyekt ko'rsatilgan sinfning ekzempliyari ekanligini aniqlash uchun ishlataladi. Keling, namunada ko'ramiz. 2ta qanaqadir sinf berilgan bo'lsin:

```
<?php  
  
// Birinchi sinf:  
  
class Class1  
  
{  
  
}  
  
  
// Ikkinchchi sinf:  
  
class Class2  
  
{  
  
}  
?>
```

Birinchi sinfning obyektini yaratamiz:

```
<?php  
  
$obj = new Class1;  
?>
```

\$obj o'zgaruvchisidagi obyekt birinchi va ikkinchi sinfga tegishliligini tekshiramiz:

```
<?php

// natija, true:
// Class1 ga tegishli ekan
var_dump($obj instanceof Class1);

// natija, false:
// Class2 ga tegishli emas
var_dump($obj instanceof Class2);
?>
```

Instanceof operatori va meros olish

Bizda ota sinf hamda avlod sinf berilgan bo'lsin:

```
<?php

// Ota sinf:
class ParentClass
{

}

// Avlod sinf:
class ChildClass extends ParentClass
{



}

?>
```

Avlod sinfning obyektini yaratamiz:

```
<?php  
$obj = new ChildClass;  
?>
```

Endi instanceof orqali obyektimiz **ParentClass** va **ChildClass** larimizga tegishli ekanligini tekshirib ko'ramiz:

```
<?php  
var_dump($obj instanceof  
         ChildClass); // natija true  
var_dump($obj instanceof  
         ParentClass); // bu ham true  
?>
```

Namunadan ko'rGANINGIZDEK, instanceof operatori ota sinf hamda avlod sinf o'rtasida hech qanday farqni sezmaydi.

Lekin, chalkashmang – agar obyekt haqiqatan ham ota sinf bo'ladigan bo'lsa, obyektni avlod sinfga tegishlilagini tekshirish, false beradi, albatta:

```
<?php  
$obj = new ParentClass; //  
      ota sinf obyekti  
  
var_dump($obj instanceof  
         ParentClass); // natija true  
var_dump($obj instanceof  
         ChildClass); // natija false  
?>
```

Obyektlar bilan ishlashda ma'lumot turini nazorat qilish

Employee nomli sinf mavjud deylik:

```
<?php

class Employee

{

    private $name;

    private $salary;

    public function __construct($name, $salary)

    {

        $this->name = $name;

        $this->salary = $salary;

    }

    public function getName()

    {

        return $this->name;

    }

    public function getSalary()

    {

        return $this->salary;

    }

}

?>
```

Shuningdek, EmployeesCollection nomli sinf ham berilgan, u xodimlar kolleksiyasini saqlash uchun ishlataladi:

```
<?php

class EmployeesCollection

{

    private $employees = [] ; // xodimlar massivi

    // To'plamga xodim qo'shish
    public function add($employee) // parametr Employee sinfi obyektini uzatadi

    {

        $this->employees[] = $employee; // to'plamga obyektni qo'shadi

    }

    // Xodimlarning umumiyligi maoshi olinadi:
    public function getTotalSalary()

    {

        $sum = 0;

        foreach ($this->employees as $employee) {

            $sum += $employee->getSalary();

        }

        return $sum;

    }

}
```

```
}
```

```
?>
```

EmployeesCollection sinfining add metodiga diqqat bilan e'tibor bering: uning parametriga Employee sinfining obyekti uzatilayapti.

Lekin, kodni o'qiyotgan dasturchi add metodiga Employee sinfi va obyekt parametri uzatilishi kerakligini tushinishi qiyin bo'ladi. Qo'pol qilib aytganda, fol ochib o'tirishiga to'g'ri keladi.

Ha, balki holatni tushuntirish uchun kodimizni izohlab ketishimiz mumkindir, biroq bu dasturchini xatoliklardan qutqarmaydi. Agar u biron bir boshqa sinfning obyektini yoki umuman massivni uzatishga harakat qilsa, qanaqadir xatoliklar kelib chiqishi tayin.

Uzatiladigan parametr turini funksiyaning tavsifida ko'rsatib ketish ancha zo'r bo'ladi.

Shuningdek, u yerda biz parametr turi sifatida qaysi sinfning obyekti uzatilishi kerakligi ham aniq qilib yozishimiz mumkin bo'ladi. Buning uchun o'zgaruvchi parametrdan oldin kutilayotgan sinf nomi yoziladi, bizning holatda bu **Employee**. Keling, add metodini qayta yozamiz:

```
<?php
```

```
class EmployeeCollection
```

```
{
```

```
    private $employees = [];
```



```
    // parametr turini aniq ko'rsatamiz:
```

```
public function add(Employee $employee)
{
    $this->employees[] = $employee;
}

public function getTotalSalary()
{
    $sum = 0;

    foreach ($this->employees as $employee) {
        $sum += $employee->getSalary();
    }

    return $sum;
}

?>
```

Endi, agar biz add metodiga boshqa sinfning obyektini uzatadigan bo'lsak, PHP bizga xatolik qaytaradi.

Statik metodlar

Sinflar bilan ishlashda shunday metod qilishimiz mumkinki, uni chaqirishimizda hech qanday obyekt yaratish talab qilinmaydi. Bunday metodlar **static** metodlar deb ataladi.

Metodni static qilib e'lon qilish uchun, kirish modifikatoridan so'ng(ya'ni, public, private yoki protected), **static** kalit so'zini yozish kerak, misol uchun:

```
<?php

class Test

{

    // Statik metod:

    public static function method()

    {

        return '!!!!';

    }

}

?>
```

Statik metodga murojaat qilish uchun sinf nomidan keyin 2 nuqta va metod nomi yoziladi, bunda sinf obyektini yaratib o'tirish shart emas, misol uchun:

```
<?php

echo Test::method(); // natija '!!!!

?>
```

Sinf ichida static metod

Agar statik metodlarni sinf ichida ishlatalishni xohlasangiz, unda unga **\$this->** orqali emas, balki **self::** yordamida murojaat qilishingizga to'g'ri keladi. Misol uchun **Math** sinf yaratamiz, uning **getDoubleSum** metodini yig'indini 2ga ko'paytirilgandagi yechimini topish uchun yaratamiz. Yangi **getDoubleSum** metodi ichida **getSum** metodimizni ham ishlatalamiz:

```
<?php

class Math

{
```

```
// 2ta sonning yig'indisini 2 karrasini topamiz
public static function getDoubleSum($a, $b)
{
    return 2 * self::getSum($a, $b); // boshqa metod
ishlatamiz
}

public static function getSum($a, $b)
{
    return $a + $b;
}

public static function getProduct($a, $b)
{
    return $a * $b;
}

}

// yangi metoddan foydalanamiz:
echo Math::getDoubleSum(1, 2); // javob 6
?>
```

Amaliyot

Bizda avvalgi darslarimizda ko'rgan, **ArraySumHelper** sinfi berilgan:

```
<?php
class ArraySumHelper
{
```

```
public function getSum1($arr)
{
    return $this->getSum($arr, 1);
}

public function getSum2($arr)
{
    return $this->getSum($arr, 2);
}

public function getSum3($arr)
{
    return $this->getSum($arr, 3);
}

public function getSum4($arr)
{
    return $this->getSum($arr, 4);
}

private function getSum($arr, $power) {
    $sum = 0;

    foreach ($arr as $elem) {
        $sum += pow($elem, $power);
    }
}
```

```
        return $sum;  
    }  
}  
?>
```

Topshiriq 1:

ArraySumHelper metodlarini statik metodlarga o'giring.

Topshiriq 2:

Sonlar mavjud bo'lgan massiv berilgan. Ushbu massivning elementlarining kvadratlar yig'indisini **ArraySumHelper** sinfi yordamida toping.

Statik xossalar

Statik metodlardan tashqari xossalarni ham statik qilish mumkin. Bunday xossalarni ham **static** kalit so'zi yordamida e'lon qilinadi:

```
<?php  
  
class Test  
{  
    // Statik xossalarni ham qilish mumkin.  
    public static $property;  
}  
?>
```

Statik xossaga ma'lumot yozish hamda uni o'qish mumkin:

```
<?php  
  
Test::$property = 'test';  
  
echo Test::$property; // natija 'test'  
?>
```

Sinf ichida statik xossa

Statik xossani, statik metodni sinf ichida qanday ishlatsak, xuddi shunday ishlatishimiz mumkin.

```
<?php

class Test

{

    // yopiq statik xossa:
    private static $property;

    // xossaga qiymat berish uchun statik metod:
    public static function setProperty($value)

    {

        self::$property = $value; // malumotni
        statik xossamizga yozamiz

    }

    // xossani qiymatini olish uchun statik metod:
    public static function getProperty()

    {

        return self::$property;

        // yozilgan ma'lumotni o'qiyamiz

    }

}

?>
```

Sinfimizni ishlatib ko'ramiz:

```
<?php
```

```
Test::setProperty('test');

    // xossaga ma'lumot yozamiz

echo Test::getProperty(); // ekranga chiqaramiz

?>
```

Konstant sinflar

Hozir biz siz bilan konstant sinflar bilan ishlaymiz. Konstanta asli o'zi, qiymati o'zgarmaydigan xossa hisoblanadi.

O'zgarmas xossalari – o'zi, doimiy bo'lgan va tasodifan o'zgarib ketmaydigan, qandaydir qiymatni saqlash uchun ishlatiladigan xossa.

Konstanta yaratish uchun uni **const** kalit so'zi yordamida e'lon qilishimiz va darrov unga qiymat berishimiz kerak:

```
<?php

class Test

{

    // Konstanta belgilaymiz:

    const constant = 'test';

}

?>
```

Dasturchilar o'rtasida umumiyligida qabul qilingan qonun-qoidalarga ko'ra, konstanta nomlari katta harflar bilan yozilishi maqsadga muvofiq, ya'ni **const constant** emas, balki **const CONSTANT**. Bu kodda konstantalarni oson ilg'ash uchun yordam beradi. Keling, sinfimizni to'g'irlaymiz:

```
<?php

class Test
```

```
{  
    // Konstanta beramiz:  
    const CONSTANT = 'test';  
}  
?>
```

Keling endi konstanta qiymatlarini qanday o'qishni ko'rib chiqamiz. Shuni aytish kerakki, konstanta sinflar ko'proq oddiy xossalarga emas, balki statikka o'xshaydi. Shuning uchun konstantalarga murojaat qilish statik xossalarga murojaat qilish bilan bir xil: klass nomini yozamiz, ikki nuqta va konstanta nomi:

```
<?php  
echo Test::CONSTANT; // natija 'test'  
?>
```

Sinf ichida konstantaga murojaat qilish

Sinf ichida konstantaga **self::** orqali murojaat qilish mumkin, quyidagidek:

```
<?php  
class Test  
{  
    const CONSTANT = 'test';  
  
    function getConstant() {  
        return self::CONSTANT;  
    }  
}  
?>
```

Metodimizdan foydalanamiz:

```
<?php  
  
$test = new Test;  
  
echo $test->getConstant(); // natija 'test'  
?>
```

Obyekt va sinflar bilan ishlash uchun funksiyalar

1. **get_class** – obyektning sinf nomini qaytaradi
2. **get_class_methods** – sinf metodlarining nomlarini oladi

```
<?php  
  
class myclass {  
  
    // konstruktor  
  
    function __construct()  
    {  
  
        return(true);  
    }  
  
    // metod 1  
  
    function myfunc1()  
    {  
  
        return(true);  
    }  
  
    // metod 2  
  
    function myfunc2()
```

```
{  
    return(true);  
}  
}  
  
$sinf_metodlari = get_class_methods('myclass');  
// yoki  
$sinf_metodlari = get_class_methods(new myclass());  
  
foreach ($class_methods as $metod_nomi) {  
    echo "$metod_nomi\n";  
}  
?>
```

Natija:

```
myclass  
myfunc1  
myfunc2
```

3. **get_class_vars** – sinfning dastlabki(defolt) xossalalarini oladi.

```
<?php  
  
class myclass {  
  
    var $var1; // buning hech qanday boshlang'ich qiymati  
    yo'q...  
    var $var2 = "xyz";  
    var $var3 = 100;
```

```
private $var4;

// konstruktor

function __construct() {
    // change some properties
    $this->var1 = "foo";
    $this->var2 = "bar";
    return true;
}

$my_class = new myclass();

$class_vars = get_class_vars(get_class($my_class));

foreach ($class_vars as $name => $value) {
    echo "$name : $value\n";
}

?>
```

Natija:

```
var1 :
var2 : xyz
var3 : 100
```

4. **class_exists** – sinf e'lon qilinganligini tekshiradi.

```
<?php  
  
// sinfni foydalanishdan avval uni e'lon qilinganlini  
tekshiramiz  
  
if (class_exists('MyClass')) {  
  
    $myclass = new MyClass();  
  
}  
  
?>
```

5. **method_exists** – sinfda belgilangan metod mavjud yoki yo'qligini tekshiradi
6. **property_exists** – sinfda belgilangan xossa mavjud yoki yo'qligini tekshiradi
7. **get_parent_class** – avlod sinfning ota sinf nomini qaytaradi.
8. **is_a** – obyekt belgilangan sinfga tegishli yoki ushbu sinf uning ota sinfidan birilagini tekshiradi.
9. **get_declared_classes** – e'lon qilingan sinflarning nomlari massivini qaytaradi.

Polimorfizm

Polimorfizm so'zi grekcha so'zdan olingan bo'lib, **poly – ko'p, morphism – shakllar** degan ma'noni anglatadi.

Polimorfizmga obyektga yo'naltirilgan dasturlashda misol keltirishimiz mumkin: turli xil sinflardagi bir xil vazifani bajaradigan metodlar bir xil nomga ega bo'lishi kerak. **Polimorfizm** – bu ko'pgina sinflarga turli xil funksional imkoniyatlarni berib umumiy interfeysni ulashadigan yoki bajaradigan OOP andozasining asosiy mohiyati hisoblanadi. Polimorfizmning foydali jihat – turli xil sinflarda yozilgan

kod uni qaysi sinfga tegishli bo'lishiga ta'sir qilmaydi. Chunki ular bir xil usulda ishlataladi. Polimorfizm talablarini bajaradigan sinflarni yozish uchun, biz 2ta muqobil yo'llarning birini tanlashimiz mumkin, yo abstract yo interfeys sinflar.

Abstrakt sinflar

Employee va **Student** sinflariga meros beradigan **User** sinfi mavjud deylik.

Tahmin qilinishicha, **Employee** va **Student** sinflarining obyektini yaratishingiz mumkin deb, lekin **User** sinfining obyektini esa – yo'q, sababi **User** sinfi o'zining avlodlari uchun umumiy xossa va metodlar to'plami uchun ishlataladi.

Bunday holatda, siz yoki boshqa dasturchi tasodifan User sinfining obyektini yaratib qo'ymasligi uchun uni yaratishni taqiqlab qo'yishingiz mumkin.

Buning uchun **abstract sınıf** deb nomlangan sinf mavjud.

Abstrakt sinflar – undan meros olish uchun mo'ljallangan sinflar hisoblanadi. Biroq, bunday sinflardan obyekt yaratib bo'lmaydi. Buning uchun sinflarni abstract deb e'lon qilish kerak, bunda biz **abstract** kalit so'zini yozishimiz kerak:

```
<?php  
abstract class User  
{  
}  
?>
```

Shunday qilib, keling User abstrakt sinfini realizatsiyasini yozamiz. Unda yopiq name xossasi va shu jumladan, uning getter va setterlari ham bo'lzin:

```
<?php

abstract class User

{

    private $name;

    public function getName()

    {

        return $this->name;

    }

    public function setName($name)

    {

        $this->name = $name;

    }

}

?>
```

Endi, **User** sinfini obyektini yaratmoqchi bo'lsak, xatolik beradi:

```
<?php

$user = new User; // xatolik beradi

?>
```

Lekin, sinfimizdan meros olishimiz mumkin. **User** abstrakt sinfidan meros oladigan **Employee** sinfini yaratmiz:

```
<?php

class Employee extends User

{

    private $salary;

    public function getSalary()

    {

        return $this->salary;

    }

    public function setSalary($salary)

    {

        $this->salary = $salary;

    }

}

?>
```

Employee sinfini obyektini yaratamiz – hammasi ishlayapti:

```
<?php

$employee = new Employee;

$employee->setName('Sanjar'); // ota sinf metod, ya'ni
User

$employee->setSalary(

    1000); //o'zining metodi, ya'ni Employee

echo $employee->

    getName(); // natija 'Sanjar'
```

```
echo $employee->  
    getSalary(); // natija 1000  
?>
```

Shunday tarzda **User** abstrakt sinfidan meros olayotgan **Student** sinfining obyektini ham yaratishimiz mumkin:

```
<?php  
  
class Student extends User  
{  
  
    private $scholarship; // stipendiya  
  
    public function getScholarship()  
    {  
        return $this->scholarship;  
    }  
  
    public function setScholarship($scholarship)  
    {  
        $this->scholarship = $scholarship;  
    }  
}  
?>
```

Abstrakt metodlar

Abstrakt sinflar abstrakt metodlardan iborat bo'lishi ham mumkin. Bunday metodlar realizatsiya qilinmaydi, bunday metodlarni avlod sinflarda ishlatish kerak. Aslini olganda bunday metodlarni realizatsiyasi – endi u avlod sinflarni vazifasi.

Buning uchun metodlarni abstrakt qilib e'lon qilish kerak, e'lon qilayotganda abstract kalit so'zini yozish kerak.

Keling, amaliyotda sinaymiz. Barcha User sinfining avlodlarida increaseRevenue(daromadni oshirish) metodi bo'ladi deb tasavvur qilamiz.

Bu metod foydalanuvchining hozirgi daromadini oladi va uni belgilangan miqdorda oshiradi.

User sinfining o'zi avlodni daromadni qanday olishini bilmaydi – xodimda bu maosh, studentda esa bu stipendiya. Shuning uchun har bir avlod sinfda bu metodlar o'zicha realizatsiya qilinadi. Bu yerda asosiy fishka bu User sinfi, unda abstrakt deb e'lon qilingan metodlarni uning avlod sinflarida bajarish kerakligini dasturchidan talab qilishidir. Aks holda, metodlar avlod sinfda bo'lmasa PHP xatolik beradi.

Shuning uchun, siz yoki dasturchi kod bilan ishlayotganda, abstrakt sinfdan meros olayotgan barcha sinflarda bo'lishi kerak bo'lgan metodlarni bajarishni hech qachon unutolmaysiz.

Keling, amaliyotda qo'llab ko'ramiz. User sinfiga **increaseRevenue** abstrakt metodini qo'shamiz:

```
<?php  
  
abstract class User  
  
{  
  
    private $name;  
  
  
    public function getName()  
    {
```

```
        return $this->name;  
    }  
  
    public function setName($name)  
    {  
        $this->name = $name;  
    }  
  
    // Абстрактный метод без тела:  
    abstract public  
        function increaseRevenue($value);  
    }  
?>
```

Bizning Employee sinfi hozircha o'zgarishsiz qolsin. Bunday holatda, hatto agar Employee sinfining obyekti yaratilmasa ham, shunchaki sinflar belgilangan kodni ishga tushirsak ham PHP xatolik beradi:

```
<?php  
  
abstract class User  
  
{  
  
    private $name;  
  
  
    public function getName()  
    {  
        return $this->name;  
    }  
?>
```

```
public function setName($name)
{
    $this->name = $name;
}

abstract public
function increaseRevenue($value);
}

class Employee extends User
{
    private $salary;

    public function getSalary()
    {
        return $this->salary;
    }

    public function setSalary($salary)
    {
        $this->salary = $salary;
    }
}

?>
```

Keling, endi **Employee** sinfida increaseRevenue metodini realizatsiyasini yozamiz:

```
<?php

class Employee extends User

{
    private $salary;

    public function getSalary()
    {
        return $this->salary;
    }

    public function setSalary($salary)
    {
        $this->salary = $salary;
    }

    // metodni realizatsiyasini yozamiz:
    public function increaseRevenue($value)
    {
        $this->salary = $this->salary + $value;
    }
}

?>
```

Sinfning ishlashini tekshiramiz:

```
<?php
$employee = new Employee;
$employee->setName('Sanjar');
```

```
$employee->setSalary(  
    1000);  
  
$employee->increaseRevenue(100); // maoshini oshiramiz  
  
echo $employee->  
    getSalary(); // natija 1100  
?>
```

increaseRevenue metodini Student sinfida realizatsiyasini yozamiz. Faqat bizning metod endi stipendiyani oshiradi:

```
<?php  
  
class Student extends User  
  
{  
  
    private $scholarship; // stipendiya  
  
  
    public function getScholarship()  
    {  
        return $this->scholarship;  
    }  
  
  
    public function setScholarship($scholarship)  
    {  
        $this->scholarship = $scholarship;  
    }  
  
  
    public function increaseRevenue($value)  
    {
```

```
$this->scholarship =  
    $this->scholarship + $value;  
}  
}  
?>
```

Ba'zi bir eslatmalar

Abstrakt sinfdan meros olayotdan sinf, ota sinfda **abstrakt** deb belgilangan barcha metodlar avlod sinfda ham belgilanishi kerak.

Bunda mavjudlik maydoni avlod sinf bilan ota abstrakt sinfniki bilan mos tushishi yoki ota abstrakt sinfda belgilangan kirish modifikatoridan pastroq kuchga ega modifikator yozilishi kerak. Ya'ni, abstrakt sinfda protected sifatida e'lon qilingan abstrakt metod, avlod sinfda ham u metod yo **protected** yo **public** deb e'lon qilinishi kerak, lekin **private** emas.

Obyektga yo'naltirilgan dasturlashda interfeyslar

O'tkan darslarda biz siz bilan abstrakt sinflar bilan sinflarni o'rgangan edik.

Allaqachon bilgанингиздек, abstrakt sinflar o'zining avlod sinflari uchun metodlar to'plamini taqdim etadi. Ushbu metodlarning ayrimlari o'sha sinfning o'zida realizatsiya bo'lishi mumkin, ba'zilari esa abstrakt deb e'lon qilinishi va avlod sinflardan bu metodni bo'lishini talab qilish ham mumkin.

Tasavvur qiling, sizning abstrakt sinfingiz faqat abstrakt ochiq va hech qanday realizatsiya qilinmagan metodlar to'plamini taqdim etadi. Aslida sizning bu ota sinfingiz avlod sinflarini interfeyslarini tasvirlayapti, ya'ni ushbu belgilangan ochiq metodlar, albatta, avlod sinflarda realizatsiya qilinishi kerak.

Nima uchun bunday narsa bizga kerak: dasturlashda kamroq xatolik qilish uchun – ota sinfda barcha kerakli metodlarni belgilab qo'yib, u metodlarni o'sha ota sinfning barcha avlod sinflarida realizatsiya qilinishiga ishonch hosil qilishimiz mumkin.

Qachon bu bizga qo'l keladi: bitta ota sinf va uning bir nechta avlod sinflarini yaratdik deylik. Agar qanchadir vaqt dan so'ng, misol uchun, bir oydan so'ng, yana bir avlod sinf yaratishni xohlab qoldik, lekin avvalgi yozgan kodimizni nimalar tasvirlanishini unutdik va hozir yozayotgan avlod sinfimizda qanaqadir kerakli metodni yozmasdan unutib qoldirib ketishimiz ehtimoli yuqori. Lekin PHP bizga bu metodni qoldirib ketishimizga yo'l qo'ymaydi – xatolik chiqaradi(agar abstrakt bo'lsa).

Yana aytaylik. Loyihangiz bilan birga ishlayotgan boshqa dasturchi ham. Siz ota sinf yozib qo'ygansiz(qaysidir vaqtlar), keyin esa sizning hamkasblaringiz ushbu ota sinfning yana bitta avlod sinfini yaratishmoqchi bo'lishdi. Sizning hamkasblaringiz agar siz ota sinfda barcha kerakli metodlarni yozib qo'yan bo'lsangiz, unda sizning hamkasblaringiz avlod sinfda yozilishi kerak bo'lgan hech qanday metodni tashlab ketolmaydi.

Yana bir muammo bor: aslida biz ota sinfimizni ichida abstrakt ochiq metodlarni yozdik, lekin biz yoki hamkasblarimiz ushbu sinfga ochiq BO'L MAGAN yoki abstrakt BO'L MAGAN metodlarni tasodifan qo'shib yuborishimiz mumkin.

Tasavvur qiling, ota sinfda abstrakt ochiq sinflardan tashqari yana boshqa metod yozishni amaliy nuqtai nazardan ta'qiqlamoqchimiz.

PHP da uning uchun abstrakt sinfini o'rniga **interfeyslar** ishlatalish mumkin.

Interfeyslar o'zi bilan, realizatsiyaga ega bo'l magan va ochiq bo'l gan barcha metodlarni taqdim etadi. Metod kodlari interfeysning avlod sinflarida realizatsiya qilinishi kerak.

Interfeyslar ham oddiy sinflar kabi e'lon qilinadi, lekin, **class** so'zini o'rniga **interface** kalit so'zi ishlataladi.

Interfeyslardan meros olish umuman boshqacha terminologiyaga ega: u sinflar interfeyslardan meros olmaydi, balki ularni realizatsiya qiladi deb aytadi. Shunga muvofiq, extends so'zini o'rniga implements kalit so'zi ishlataladi. Implement so'zi, bajarmoq, realizatsiya qilmoq degani. Ya'ni, biz bo'sh hali hech qanday kod yozilmagan metodlarni interfeyslarda yozamiz va ularga boshqa sinflarda tegishli kodlar yoziladi. Bu realizatsiya deyiladi.

Eslatma:

1. interfeysdan obyekt yaratib bo'l maydi
2. interfeysning barcha metodlari ochiq ya'ni public deb e'lon qilinishi kerak.

3. Interfeysning barcha metodlari hech qanday realizatsiyaga ega bo'lmanan bo'lishi kerak. – **public function helloWorld();**
4. Undan tashqari interfeyslarda faqatgina metodlar bo'ladi, xossalari esa yo'q.
5. Interfeys nomi va sinf nomi bir xil bo'lmasligi kerak.

Interfeyslarning qo'llanishi

Xo'sh, biz endi interfeyslarni, sinfning barcha kerakli, realizatsiya qilinishi kerak bo'lgan metodlarini nazorat qilishda yaxshi ekanligini bilib oldik. Keling endi ko'proq amaliyotda qo'llab ko'rishga harakat qilamiz. Bizda **FiguresCollection** sinfi berilgan, u o'zida shakllar obyektining massivini saqlaydi:

```
<?php

class FiguresCollection

{

    private $figures = []; // shakllar uchun massiv

}

?>
```

To'plamga, ya'ni kolleksiyamizga obyekt qo'shish uchun `addFigure` metodimizni sinfimiz ichida realizatsiya qilamiz:

```
<?php

class FiguresCollection

{

    private $figures = [];



    public function addFigure($figure)
```

```
{  
    $this->figures[] = $figure;  
}  
}  
?>
```

Shuni aniqki, biz **addFigure** metodi parametriga shakl obyekti uzatilishini bilamiz. Lekin bu yerda hech qanday nazorat mavjud emas!

Keling **Figure** sinfi obyekti turiga mansub bo'lgan turni metodda ko'rsatma sifatida yozib qo'yamiz:

```
<?php  
  
class FiguresCollection  
{  
  
    private $figures = [];  
  
  
    public function addFigure(Figure $figure)  
    {  
        $this->figures[] = $figure;  
    }  
}  
?>
```

Endi yuqorida yozgan kodimizni izohlab beray.

Agar haqiqatdan ham Figure mavjud bo'ladigan bo'lsa unda parametriga faqat o'sha sinfning obyekti uzatilishi mumkin, shuning uning avlodlari ham. Bizda, biroq **Figure** – bu interfeys. Bunday holatda, biz metodda yozib qo'ygan

ishoramiz shuni anglatadiki, metod parametriga faqatgina **Figure** interfeysiini realizatsiya qilayotgan sinfning obyekti uzatilishi mumkin. Keling, FiguresCollection sinfi obyektini yaratib, unga shakl qo'shib ko'ramiz:

```
<?php

$figuresCollection = new FiguresCollection;

// 2ta kvadrat qo'shamiz:
$figuresCollection->add(new Quadrate(2));
$figuresCollection->add(new Quadrate(3));

// 2ta to'rtburchak qo'shamiz:
$figuresCollection->add(new Rectangle(2, 3));
$figuresCollection->add(new Rectangle(3, 4));

?>
```

Lekin boshqa sinfning obyektini qo'shish xatolikka sabab bo'ladi:

```
<?php

$figuresCollection = new FiguresCollection;

class Test {} // qanaqadir sinf
$figuresCollection->
    add(new Test); // xatolik beradi
?>
```

Amaliyotdan biz qanday nazoratni o'rgandik: kolleksiyaga qo'shilayotgan barcha figuralar **Figure** interfeysiini realizatsiya

qiladi va biz ishonishimiz mumkinki, ularning har birida **getSquare** va **getPerimeter** metodlari mavjud bo'ladi.

Kelgusida, kvadrat va to'rtburchaklardan tashqari yana boshqa shakl qo'shishimiz mumkin, masalan, uchburchak. Bunday holatda uchburchakda ham **getSquare** va **getPerimeter** metodlari bo'ladi. Ya'ni uning kvadrati va perimetri.

Biz **FiguresCollection** sinfida yana bitta metod qo'shishimiz mumkin. Bu metodni **getTotalSquare** deb nomlaymiz. Bu metodni vazifasi kolleksiyadagi shakllarning umumiyligini topishdan iborat.

Figure interfeysi realizatsiya qilayotgan har bir sinfning **getSquare** metodi bo'lishiga 100% ishonchimiz komil bo'ladi. Mana metodning realizatsiyasi:

```
<?php

class FiguresCollection

{

    private $figures = [];


    public function addFigure(Figure $figure)
    {

        $this->figures[] = $figure;
    }

    // Umumiyligini topish:

    public function getTotalSquare()
    {
        $sum = 0;
```

```
foreach ($this->figures as $figure) {  
    $sum += $figure->  
        getSquare(); // getSquare metodidan  
foydalanamiz  
}  
  
return $sum;  
}  
}  
?>
```

Topshiriq 1:

Kodimga qaramasdan, **FiguresCollection** sinfini mustaqil ravishda realizatsiya qiling.

Topshiriq 2:

FiguresCollection sinfiga barcha shakllarning umumiylarini topish uchun **getTotalPerimeter** metodini qo'shing.

Interfeys metodlarida parametrlar

Interfeyslarda metodlarni tasvirlayotganda faqat metodlarning nomini ko'rsatish emas, balki ular qabul qiladigan parametrlarni ham ko'rsatish kerak. Namunada ko'ramiz.

Matematik amallar: qo'shish, ayirish, ko'paytirish va bo'lish uchun mo'ljallangan **iMath** interfeysi berilgan. Interfeysimiz quyidagicha ko'rinishga ega:

```
<?php
```

```
interface iMath
{
    public function sum(); // qo'shish
    public function subtract(); // ayirish
    public function multiply(); // ko'paytirish
    public function divide(); // bo'lish
}
?>
```

Hozir interfeysimiz metodlari hech qanday parametr qabul qilmayapti va shuningdek interfeysi realizatsiya qilayotgan sinf metodlari ham hech qanday parametr qabul qilmasligi kerak. Aks holda xatolik chiqadi. Shunday ekan, unda keling interfeysimizdagi metodlarning parametrlarini yozib chiqamiz:

```
<?php
interface iMath
{
    public function sum($a, $b); // qo'shish
    public function subtract($a, $b); // ayirish
    public function multiply($a, $b); // ko'paytirish
    public function divide($a, $b); // bo'lish
}
?>
```

Endi interfeysimizni realizatsiyasini yozamiz:

```
<?php
class Math implements iMath
{
    public function sum($a, $b)
```

```
{  
    return $a + $b;  
}  
  
public function subtract($a, $b)  
{  
    return $a - $b;  
}  
  
public function multiply($a, $b)  
{  
    return $a * $b;  
}  
  
public function divide($a, $b)  
{  
    return $a / $b;  
}  
}  
?>
```

Agar sinfimizda interfeys metodlariga berilgan parametrlar sonidan ko'proq parametr berilsa – bu ishlamaydi: PHP xatolik beradi. Bunday holatda biz hech qanday metod parametrini na unutamiz, na oshiqcha yozib yuboramiz.

Eslatma!

Yuqorida yozilganidek, interfeys va sinf nomi bir xil bo'lishi mumkin emas. Agar bunday bo'lsa, nom bilan muammolar yuzaga kelishi mumkin. Misol uchun, agar biz **User** interfeysi realizatsiya qiladigan **User** sinfini yaratmoqchi bo'lsak, bu yerda nomlar o'rtaida konflikt bo'lishi mumkin. Bunday muammoni oldini olish uchun, sinf va interfeys nomlarini boshqa-boshqa nom bilan atash kerak. Dasturchilar qabul qilgan umumiyligiga qoidalarga ko'ra, bunday holatda interfeys nomlarini kichkina **i** harfini qo'shish maqsadga muvofiq. Bu harf bizga buni interfeys ekanligini ko'rsatib turadi. Ya'ni bizni holatda endi interfeys **User** emas, balki **iUser** bo'ladi, va uni realizatsiya qiladigan sinfimiz **Userligicha** qoladi. Bunday yondashuv barcha darslarimizda qo'llaniladi.

Topshiriq:

Iuser interfeysi berilgan:

```
<?php

interface iUser
{
    public function
        setName($name); // ism berish
    public function
        getName(); // ismni olish
    public function
        setAge($age); // yosh belgilash
    public function
        getAge(); // yoshni olish
}
```

?>

Ushbu interfeysi realizatsiya qiladigan **User** sinfini yarating.

Interfeysda konstruktor e'lon qilish

Interfeyslarda konstruktorlarni ham e'lon qilishimiz mumkin. Keling, buni amaliyotda ko'ramiz.

Konstruktorida 2 ta parametr qabul qiladigan, perimeter va maydonini topadigan metodlarga ega Rectangle sinfini yarataylik. Keling, sinfimizni iRectangle interfeysi yordamida yozamiz:

```
<?php

interface iRectangle

{
    public function __construct($a, $b); // 
        2ta parametrli konstruktor
    public function
        getSquare(); // maydonini topish
    public function
        getPerimeter(); // perimetreni topish
}

?>
```

Keling endi **iRectangle** interfeysi realizatsiyasini yozamiz:

```
<?php

class Rectangle implements iRectangle
{
    private $a;
```

```
private $b;

public function __construct($a, $b)
{
    $this->a = $a;
    $this->b = $b;
}

public function getSquare()
{
    return $this->a * $this->b;
}

public function getPerimeter()
{
    return 2 * ($this->a + $this->b);
}

?>
```

Interfeysda konstruktorni e'lon qilish bizga nima beradi: Birinchidan, biz sinfda konstruktorni realizatsiya qilishni unutmaymiz. Ikkinchidan, niterfeys, sinf konstruktori 2ta parametr olishini aniq ko'rsatadi, kam ham emas, ko'p ham emas. Bu bizni tasodifiy xatoliklardan qutqaradi.

Topshiriq 1:

Kub shaklini tasvirlaydigan **iCube** interfeysi yaratting. Interfeysda kub tomonlarini qabul qiladigan parametr hamda

kub hajmi va maydonining yuzini olish uchun metodlarni tasvirlaydigan konstruktori bo'lsin.

Topshiriq 2:

iCube interfeysini realizatsiy qiladigan **Cube** sinfini yarating.

Interfeyslarning bir-biridan meros olishi

Interfeyslar ham sinflar kbi, bir-biridan **extends** kalit so'zi yordamida meros olishlari mumkin. O'tgan darsimizda siz bilan **iRectangle** interfeysini yaratgandik:

```
<?php

interface iRectangle

{
    public function __construct($a, $b);
    public function getSquare();
    public function getPerimeter();
}

?>
```

Lekin, bizda **iRectangle** interfeysini metodlarini(getSquare va getPerimeter) o'zida tasvirlagan **Figure** interfeysimiz ham bor:

```
<?php

interface Figure

{
    public function getSquare();
    public function getPerimeter();
}

?>
```

Keling, shunday qilamizki, **iRectangle** interfeysimiz **Figure** interfeysidan meros olsin:

```
<?php

interface iRectangle extends Figure

{
    public function __construct($a, $b);
}

?>
```

Mana, endi **Figure** interfeysi metodlarin ham **iRectangle** ishlata oladi.

Interfeyslar va instanceof

instanceof operatori ham xuddi meros olish kabi, interfeyslar bilan ishlay oladi.

Bizda **Figure** interfeysi realizatsiya qilayotgan **Quadrat** sinfi mavjud:

```
<?php

interface Figure

{



}

class Quadrat implements Figure

{



}

?>
```

Ushbu sinfning obyektini yaratamiz va uni **instanceof** operatori bilan tekshirib ko'ramiz:

```
<?php  
  
$quadrate = new Quadrate;  
  
  
var_dump($quadrate  
    instanceof Quadrate); // natija true  
  
var_dump($quadrate  
    instanceof Figure); // natija true  
  
?>
```

Ya'ni, **instanceof** operatori bilan biron-bir sinfni ko'rsatilgan interfeysni realizatsiya qilayotgan yoki yo'qligini tekshirish mumkin ekan.

Bir nechta interfeyslar

PHP dasturlash tilida ko'p meros olish mavjud emas – har bir sinfning faqat bitta ota sinfi mavjud bo'ladi.

Interfeyslarga kelganda, bu qoida o'zgaradi: har bir sinf bitta yoki undan ko'p bo'lgan interfeyslarni realizatsiya qilish mumkin.

Har bir interfeys nomi implements kalit so'zidan so'ng, vergul bilan ajratib yozilishi kerak bo'ladi.

Bunda interfeyslarning abstrakt sinflardan yana bir farqi ko'zga tushib qoladi – demak ko'pgina interfeyslarni realizatsiya qilishimiz mumkin, lekin bir nechta abstrakt sinflardan meros olishimiz esa yo'q ekan.

Keling, amaliyotda sinab ko'ramiz. **Figure** interfeysidan tashqari yana bir **Tetragon**(to'rtburchak) interfeysimiz bor. Ushbu interfeysning metodlarini **Quadrat**(kvadrat) va **Rectangle**(to'g'ri to'rtburchak) sinfi realizatsiya qiladi va shuningdek ularning har birida 4 ta tomon mavjud bo'ladi, lekin **Disk** sinfimizda bunday emas.

Tetragon interfeysi barcha to'rtburchak tomonlari uchun getterni tasvirlagan:

```
<?php

interface Tetragon

{

    public function getA();

    public function getB();

    public function getC();

    public function getD();

}

?>
```

Shuningdek, bizning avval tayyorlab qo'ygan **Figure** interfeysimiz ham mavjud:

```
<?php

interface Figure

{

    public function getSquare();

    public function getPerimeter();

}

?>
```

Shunday qilaylikki, **Quadrat** sinfi ham **Figure** ham **Tetragon** interfeyslarini realizatsiya qilsin. Buning uchun interfeyslar haqida yozganimizni eslang: har bir interfeys implements kalit so'zidan so'ng vergul bilan ajratib yozilishi kerak.

```
<?php

class Quadrat implements Figure, Tetragon

{

    // realizatsiya ketadi

}

?>
```

Endi biz **Tetragon** interfeysi realizatsiya qilayotgan **Quadrat** sinfimizni tugatayapmiz. Kvadratimiz hozir to'rtburchakning buzulgan holati bo'lib qoldi, sababi kvadratning barcha tomonlari teng bo'ladi, bizda esa har xil tomonlar mavjud bo'lganligi uchun har xil metodlar bor. Barcha yangi metodlar bir xil narsani qaytaradi:

```
<?php

class Quadrat implements Figure, Tetragon

{

    private $a;

    public function __construct($a)
    {
        $this->a = $a;
    }

}
```

```
public function getA()
{
    return $this->a;
}

public function getB()
{
    return $this->a;
}

public function getC()
{
    return $this->a;
}

public function getD()
{
    return $this->a;
}

public function getSquare()
{
    return $this->a * $this->a;
}

public function getPerimeter()
{
```

```
    return 4 * $this->a;  
}  
}  
?>
```

To'rtburchakning barcha tomonlari teng bo'lmasligi va ular bir-biriga teskari ekanligi aniq. Bunday holatda yangi metodlarimiz o'z ishini bajaradi. Ba'zi trapetsiyalarda ham 4 tomon teng bo'lmaydi. Lekin, bizning qanday turdag'i shaklni yasashimiz ahamiyatga ega emas – muhimi barcha shakllarning o'zini belgilangan metodlari bo'lishi(xuddi yuqoridagi to'rtburchak va kvadrat kabi) va bir xil ishlashi.

Sinfdan meros olish va interfeysni realizatsiya qilish

Sinf biron bir boshqa sinfdan meros olishi hamda shu bilan birga qaysidir interfeysni realizatsiya qilishi ham mumkin. Bularni siz bilan amaliyotda ko'rib chiqamiz.

Yoshi, maoshi va dasturlash bilan shug'ullanadigan tillar ro'yhatiga ega bo'lgan **Programmer**(dasturchi) sinfini yaratamiz. Hozircha bizning sinfimiz tavsifi juda noaniq: ha, sinfimizning, ya'ni dasturchimizning ismi bo'ladi, maoshi bo'ladi, tillari bo'ladi, - lekin sinfimizda qanday metodlar bo'ladi? Keling, sinfimizni **iProgrammer** interfeysi yordamida aniqroq yozib olamiz:

```
<?php  
  
interface iProgrammer  
{  
    public function __construct($name,  
        $salary); // ism va maosh kiritamiz
```

```
public function  
    getName(); // ismni olamiz  
  
public function  
    getSalary(); // maoshini olamiz  
  
    public function getLangs(); // dasturchi biladigan  
tillar massivi  
        olamiz  
  
    public function addLang($lang); //  
        tillar massiviga til qo'shamiz  
}  
  
?>
```

Yuqoridagi yozgan sinfimizga chuqurroq razm soladigan bo'lsak, shunisi aniq bo'ladiki, bizda shunga o'xshash **Employee** sinfi mavjud ekan. U **Programmer** sinfining barcha metodlarini realizatsiya qilmasa ham, lekin qisman qiladi. Mana, quyida **Employee** sinfimizning kodi keltirilgan:

```
<?php  
  
class Employee  
{  
  
    private $name;  
    private $salary;  
  
    public function  
        __construct($name, $salary)  
    {  
        $this->name = $name;  
        $this->salary = $salary;
```

```
}

public function getName()
{
    return $this->name;
}

public function getSalary()
{
    return $this->salary;
}

?>
```

Bizning holatda, hozirgi yangi **Programmer** sinfimizga **Employee** sinfidan o'ziga kerakli bo'lган ayrim metodlarni meros qilib olib berishimiz mantiqan to'g'ri keladi.

```
<?php

class Programmer extends Employee
{

}

?>
```

Shu vaqtida, **Programmer** sinfimiz **iProgrammer** interfeysini realizatsiya qilishi kerakligini bilamiz va uni amalga oshiramiz:

```
<?php

class Programmer implements iProgrammer
```

```
{  
}  
?>
```

Keling endi **Employee** sinfimizdan meros olishni va **iProgrammer** interfeysiini realizatsiya qilishni birlashtiramiz:

```
<?php  
  
class Programmer  
  
    extends Employee implements iProgrammer  
  
{  
  
}  
  
?>
```

Endi **Programmer** sinfimiz **Employee** sinfidan **__construct**, **getName()** va **getSalary()** metodlarini meros qilib oladi hamda **iProgrammer** interfeysining **getLangs** hamda **addLang** realizatsiya qiladi:

```
<?php  
  
class Programmer  
  
    extends Employee implements iProgrammer  
  
{  
  
    public function addLang($lang)  
  
    {  
  
        // realizatsiya  
  
    }  
  
    public function getLangs()  
}
```

```
{  
    // realizatsiya  
}  
}  
?>
```

__construct, **getName()** va **getSalary()** metodlari **Employee** sinfidan **Programmer** sinfiga meros qilib olib berilganligi uchun ularni **Programmer** sinfida realizatsiya qilishimiz shart bo'lmay qoladi.

Interfeyslarda konstanta

Interfeyslar sinf xossalari o'z ichiga olmaydi, lekin, konstantalar bundan mustasno. Interfeysning konstantalari sinf konstantalari kabi bir xil ishlaydi, bitta istisno bor – ularni meros olayotgan sinfda yoki interfeysda qayta belgilab bo'lmayi.

Misol uchun shar bilan ishlash uchun tasvirlangan **iSphere** interfeysi yaratamiz. Bunda biz sharning hajmi va maydon yuzini topishimiz kerak. Buning uchun bizga **PI** soni kerak bo'ladi. Uni biz intefeyesda konstanta qilib belgilab qo'yamiz. Mantiqan o'ylab qaraganda ham, **Pi** soni, o'zgarmas son, ya'ni konstanta:

```
<?php  
  
interface iSphere  
{  
    const PI = 3.14; // PI soni konstanta shaklida  
  
    // shar konstruktori:
```

```
public function __construct($radius);

// sharning hajmini topish uchun:
public function getVolume();

// sharning maydon yuzini topish uchun:
public function getSquare();

}

?>
```

Interfeyslar bilan ishlash uchun ba'zi kerakli funksiyalar

- **interface_exists** – interfeys belgilan yoki belgilanmaganligini tekshiradi.

```
<?php

// interfeysi ishlashidan oldin uni belgilanganligini
tekshirib olamiz

if (interface_exists('MyInterface')) {

    class MyClass implements MyInterface

    {

        // metodlar

    }

}

?>
```

- **get_declared_interfaces** – barcha e'lon qilingan interfeyslar massivini qaytaradi.

```
<?php
```

```
print_r(get_declared_interfaces());  
?>
```

```
Array  
(  
    [0] => Traversable  
    [1] => IteratorAggregate  
    [2] => Iterator  
    [3] => ArrayAccess  
    [4] => reflector  
    [5] => RecursiveIterator  
    [6] => SeekableIterator  
)
```

Treytlar bilan ishlash

Avvalgi darslarimizdan o'rganganingizdek, PHP da bir nechta sinflardan meros olish mumkin emas, faqat bittadan olish mumkin.

Oldinroq bu muammoni meros olishni o'rniiga biror sinfni obyektini boshqasini ichida ishlatish bilan bartaraf etgan edik. PHP da yana bir standart usuli bor. U **treytlarni** ishlatish usuli hisoblanadi.

Treytlar o'zi bilan boshqa sinfga qo'shilishi mumkin bo'lgan metod va xossalari to'plamini taqdim etadi. Bunda treytning xossa va metodlari sinf tomonidan xuddi o'zinikidek qabul qilinadi.

Treytning sintaksi xuddi sinfni yaratilish sintaksisiga o'xshaydi, bitta istisnosi bor – treytning nomi **trait** kalit so'zi

yordamida ifodalanadi. Treytning ekzemplyarini, ya'ni obyektini yaratib bo'lmaydi – treylar faqat boshqa sinfga ulanish uchun yaratilgan.

Bog'lanish **use** buyrug'i yordamida amalga oshiriladi, bu buyruqdan so'ng bitta bo'shliq tashlash orqali treytning nomi kiritiladi. Bu buyruq sinfning eng yuqori qismida yoziladi. Keling, treylarni amaliyotda qo'llash yo'llarini ko'rib chiqamiz. Bizda name va age yopiq xossalardan hamda uning getterlaridan iborat bo'lgan Helper treyti mavjud:

```
<?php

trait Helper

{

    private $name;

    private $age;

    public function getName()

    {

        return $this->name;

    }

    public function getAge()

    {

        return $this->age;

    }

}

?>
```

Shuningdek bizda yana konstruktorda **name** va **age** xossalariga qiymat beradigan mana bunday **User** sinfi mavjud:

```
<?php

class User

{

    public function __construct ($name, $age)

    {

        $this->name = $name;

        $this->age = $age;

    }

}

?>
```

Keling endi **User** sinfimizning xossasi uchun getter qo'shamiz. Uni faqat sinfda belgilamaymiz, balki shunchaki allaqachon metodlar realizatsiya qilingan **Helper** treytimizni ulab qo'yamiz:

```
<?php

class User

{

    use Helper; // treytni ulaymiz


    public function __construct ($name, $age)

    {

        $this->name = $name;

        $this->age = $age;

    }

}
```

?>

Sinfimizga treytni ulaganimizdan so'ng uning metodlari sinfda ishlatalishi mumkin bo'ladi. Bunda biz ushbu metodlarga har doimgidek murojaat qilishimiz mumkin:

```
<?php

$user = new User('Sanjar', 30);

echo $user->getName(); // natija 'Sanjar'
echo $user->getAge(); // natija 30

?>
```

Treytning xossalari ham sinfimizda mavjud bo'ladi, albatta.

Treytlarning foydali jihatlarini ko'rsatish uchun keling yana bir **City**(shahar) nomli sinf yaratamiz. Shaharning yoshi va ismi bo'ladi, lekin mantiqan shahar(**city**) va foydalanuvchi(**user**) bitta ota sinfdan meros olishi to'g'ri kelmaydi. Unda boshqacharoq xossa va metodlari bo'lishi mumkin, shaharga tegishli bo'lмаган. Shuning uchun **City** sinfimizda o'zimiz yaratgan **Helper** treytimizdan foydalanamiz:

```
<?php

class City

{

    use Helper;

    public function __construct($name, $age)
    {
        $this->name = $name;
        $this->age = $age;
    }
}
```

```
}
```

```
}
```

```
?>
```

Sinfimizni ishlayotganligini tekshirib ko'ramiz:

```
<?php
```

```
$city = new City('Guliston', 59);
```

```
echo $city->getName(); // natija 'Guliston'
```

```
echo $city->getAge(); // natija 59
```

```
?>
```

Topshiriq 1:

Name(nom), **age**(yosh), **population**(aholi soni) va ular uchun getterlardan iborat bo'lgan **Country**(mamlakat) sinfini realizatsiya qiling. Sinfimizning kodini qisqartirish uchun uni xossalari va metodlari avvaldan mavjud bo'lgan **Helper** treytidan foydalaning.

Bir nechta treytlar

Sinf ichida bitta emas, balki bir nechta treytlarni ulashimiz mumkin. Bu ularning meros olishdan ustunligi va afzalligi hisoblanadi. Sinfga kerakli bo'lgan treytlar **use** buyrug'idan keying treyt nomidan keyin, vergul bilan ajratilib, ketma-ket yoziladi.

Topshiriq 2:

Uchta, **Trait1**, **Trait2** va **Trait3** nomli treyt yaratting. Birinchi treytda **method1** metodi bo'lsin va u 1 qaytarsin, ikkinchi treytda **method2** bo'lsin va 2 qaytarsin va uchinchi treytda **method3** bo'lsin va 3 qaytarsin. Ularni hammasini yopiq qilib e'lon qiling.

Topshiriq 3:

Uchala yaratilgan treylardan foydalanadigan **Test** sinfini yarating. Bu sinfda barcha ulangan treylarning metodlarini yig'indisini qaytaradigan `getSum` ochiq metodini ishlab chiqing.

Treytlarda konfliktlarni hal qilish

Bitta sinf bir nechta treylarni ishlatishi mumkin, shu bilan birga bitta treylarda bir xil nomli metodlar ham bo'lishi ehtimoldan holi emas. Bunday holatda, PHP fatal xatolik beradi. Vaziyatdan chiqib ketish uchun, nomlar konfliktini aniq shaklda hal etishimiz kerak. Bu qanday qilinadi – amaliyotda ko'ramiz.

Bizda 2ta treytda bir xil **method** metodi berilgan:

```
<?php

trait Trait1

{
    private function method()
    {
        return 1;
    }
}

trait Trait2
{
    private function method()
    {
}
```

```
        return 2;  
    }  
}  
?>
```

Shuningdek, ikkala treylarni ham ishlatalayotgan **Test** sinfimiz ham bor. Agar treylarni shunchaki sinfga ulab qo'yadigan bo'lsak, treylarda bir xil metodlar bo'lgani uchun PHP xatolik beradi:

```
<?php  
// xatoliky!  
  
class Test  
{  
    use Trait1, Trait2; // treytni ulaymiz  
}  
?>
```

Keling treytlarimizning nomlar konfliktini hal etamiz(bu kontektsda bu so'z, yo'q qilish bilan bir xil qiymatga ega). Buning uchun maxsus **insteadof(-ning o'rniغا)** operatori ishlataladi.

Keling, o'sha operatori bilan mana bunday deb ko'ramiz: **Trait2** treytidagi **method** metodinining o'rniغا, **Treyt1** treytidagi **method** metodini ishlat.

```
<?php  
  
class Test  
{  
    use Trait1, Trait2 {  
        Trait1::method insteadof Trait2;
```

```
    }

}

new Test;

?>
```

Ko'rganingizdek, sintaksis quyidagicha: boshia treytning nomi, keyin 2 nuqta, keyin metod nomi, keyin **insteadof** operatori va ikkinchi treyt nomi. Tekshiramiz:

```
<?php

class Test

{

    use Trait1, Trait2 {

        Trait1::method insteadof Trait2;

    }

    public function __construct()
    {
        echo $this->method(); // natija 1, bu birinchi treytning metodi
    }
}

new Test;

?>
```

Agar biz birinchi treytning metodini ishlatishni ko'rsatganimizda, shunga muvofiq tarzda ikkinchi treytning metodi ko'rinxay qoladi. Lekin, uni ya'ni ikkinchi treytning

metodini ishlatischimiz mumkin, uni as kalit so'zi orqali qayta nomlash orqali, mana bunday tarzda:

```
<?php

class Test

{

    use Trait1, Trait2 {

        Trait1::method insteadof Trait2; //

        birinchi treytdan metodni oldik

        Trait2::method as method2; // ikkin treytning

metodini

        //method2 deb qaytanomlab ishlatalamiz

    }

    public function __construct()

    {

        echo $this->method() +

        $this->method2(); // natija 3

    }

}

new Test;

?>
```

Topshiriq 1:

Trait1, Trait2, Trait3 nomli 3ta treyt yaratting. Birinchi treytimizning metodi **method**, 1 qaytaradi, ikkinchi treytimizning metodi ham shunday, 2 qaytaradi, uchinchi

treytimizning metodi ham yuqoridagilar bilan bir xil, 3 qaytaradi.

Topshiriq 2:

Barcha yaratgan treytlarimizni ishlataladigan **Test** sinfini yarating. Bu sinfda barcha ulangan treylarni metodlar natijasini yig'indisini qaytaradigan **getSum** metodini yarating.

Kirish modifikatori va treylar

Diqqatingizni bir narsaga qaratmoqchiman: treylarni ishlatalish –bu meros olish emas. Meros olishda, yopiq metod va xossalarni meros qilib olib bo'lmaydi.

Treylarda aksincha, hammasi teskari: sinfda foydalanayotgan treytimizning ham ochiq, ham yopiq xossa va metodlari ishlatalishi mumkin. Keling, namunada ko'ramiz:

```
<?php

trait TestTrait

{
    // yopiq metod:
    private function method()
    {
        return '!!!!';
    }
}

class Test
{
    use TestTrait; // treytni ulaymiz
```

```
public function __construct()  
{  
    // treytning yopiq metodini ishlatalamiz:  
    echo $this->method(); // natija '!!!!'  
}  
  
}  
  
new Test;  
?>
```

Treytning metodini kirish modifikatorini o'zgartirish

Treytda metod uchun har qanday kirish modifikatori(public, private yoki protected) ishlatalishi mumkin. Agar kerak bo'lsa, sinfning o'zida ushbu modifikatorlarni boshqasiga alishtirishimiz mumkin. Buning uchun use ning blok qismida as kalit so'zidan so'ng yangi modifikator ko'rsatilishi kerak.

Namuna:

```
<?php  
  
trait TestTrait  
{  
  
    private function method()  
    {  
        return '!!!!';  
    }  
  
}  
  
class Test
```

```
{  
    use TestTrait {  
        TestTrait::method  
            as public; // yopiq metodni, ochiqqa  
            o'zgartiramiz  
    }  
}  
  
$test = new Test;  
echo $test->method(); // natija '!!!'  
?>
```

Metodlarning ustunligi

Agar sinfda ham treytida ham bir xil nomli metod bo'ladigan bo'lsa unda sinf metodi treyt metodini ustidan g'olib chiqadi, ya'ni undan ustunlik qiladi:

```
<?php  
  
trait TestTrait  
{  
    public function method()  
    {  
        return 'trait';  
    }  
}  
  
class TestClass  
{
```

```
use TestTrait;

// bir xil nomli metod:
public function method()
{
    return 'test';
}

}

$test = new TestClass;
echo $test->method(); // natija - test
//sinfni o'zini metodi ishladi
?>
```

Agar avlod sinfda hech qanday unaqa metod bo'lmasa, lekin treyt metodi hamda ota sinfning metodi bilan nomlar konflikti bo'lsa, unda treytning metodi ustunlik qiladi:

```
<?php

trait TestTrait
{
    // method nomli metod
    public function method()
    {
        return 'trait';
    }
}

// Ota sinf:
```

```
class ParentClass

{
    // method nomli metod:
    public function method()
    {
        return 'parent';
    }
}

// ota sinfdan methodni meros qilib olayapti:

class TestClass extends ParentClass
{
    use TestTrait;
}

$test = new TestClass;

echo $test->method(); //natija - trait
//treyt metodi ustun
?>
```

Treytlarda abstrakt metodlar

Treytlarda ham o'zingiz xohlagan metodni abstrakt e'lon qilishingiz mumkin. Ushbu treytdan foydalanayotgan sinf,treytda abstrakt e'lon qilingan metodlarni realizatsiya qilishi shart bo'ladi. Buni namunada ko'rishimiz mumkin. Bizda mana bunday treyt mavjud:

```
<?php
```

```
trait TestTrait

{
    public function method1()
    {
        return 1;
    }

    // abstrakt metod:
    abstract public function method2();
}

?>
```

Eslatma!

Treytdagi abstrakt metodlar **public** va **protected** rejimda bo'lishi mumkin, lekin **private** emas.

Treytimizni **Test** sinfi ichida ishlatayapmiz. Treytda abstrakt metodlarni mavjudligi dasturchini **Test** sinfida ana shu metodlarni realizatsiya qilishga majbur qiladi, aks holdda PHP xatolik bilan jazolaydi ☺.

```
<?php

class Test

{
    use TestTrait; // treyt

    // abstrakt metodni realizatsiya qilamiz:
    public function method2()
    {
```

```
        return 2;  
    }  
  
}  
  
new Test;  
?>
```

Topshiriq:

TestTrait treytidagi kodni va **Test** sinfidagi kodni nusxalab oling va sinfdan **method2** metodini o'chiring. PHP qanaqa xatolik berayotganligiga e'tibor bering.

Treytlarni treytarda qo'llash

Treytlar,sinflarga o'xshaydi ular ham boshqa treytlarni ishlatsishi mumkin. Keling, buni namunada ko'ramiz. Bizda 2ta metodi mavjud bo'lgan treyt bor:

```
<?php  
  
trait Trait1  
  
{  
  
    private function method1()  
  
    {  
  
        return 1;  
  
    }  
  
  
    private function method2()  
  
    {  
  
        return 2;  
  
    }  
}
```

```
}
```

```
?>
```

Va yana bitta treyt mavjud:

```
<?php
```

```
trait Trait2
```

```
{
```

```
    private function method3()
```

```
    {
```

```
        return 3;
```

```
    }
```

```
}
```

```
?>
```

Keling endi **Trait1** treytimizni **Trait2** treytimizga ulaymiz:

```
<?php
```

```
trait Trait2
```

```
{
```

```
    use Trait1; // treytni ulayapmiz
```



```
    private function method3()
```

```
    {
```

```
        return 3;
```

```
    }
```

```
}
```

```
?>
```

Bunday bog'lanishdan so'ng, **Trait2** treytida o'zining metodlaridan tashqari yana bir **Treyt1** metodi paydo bo'ladi.

Buni tekshirib ko'ramiz: Test sinfini yaratamiz, unga Trait2 treytini ulaymiz va birinchi qaysi, ikkinchi qaysi va uchinchi qaysi metodlar paydo bo'lganligini aniqlaymiz:

```
<?php

class Test

{

    use Trait2; // treyt

    public function __construct()
    {

        echo $this->method1(); // birinchi treytning metodi

        echo $this->method2(); // birinchi treytning metodi

        echo $this->method3(); // ikkinchi treytning metodi

    }

}

new Test;

?>
```

Topshiriq:

Yuqoridagidek treylarni mustaqil yarating va ularni **Test** sinfiga ulang. Sinfda, ulangan treyt metodlari qaytargan natijalarini yig'indisini qaytaradigan `getSum` metodini yarating.

Treytlar bilan ishlash uchun maxsus funksiyalar

- **trait_exists** – treyt mavjud yoki yo'qligini tekshiradi

```
<?php

trait World {

    private static $instance;

    protected $tmp;

    public static function World()
    {
        self::$instance = new static();
        self::$instance->tmp = get_called_class() . '
' . __TRAIT__;

        return self::$instance;
    }

}

if ( trait_exists( 'World' ) ) {

    class Hello {
        use World;

        public function text( $str )
        {
            return $this->tmp.$str;
        }
    }
}
```

```
    }

}

echo Hello::World()->text('!!!!'); // Hello World!!!
```

- **get_declared_traits** – barcha yaratilgan treylar massivini qaytaradi

```
<?php

namespace Example;

// Treyt yaratamiz

trait FooTrait
{
}

// Abstrakt sinf yaratamiz

abstract class FooAbstract
{
}

// Sinf yaratamiz

class Bar extends FooAbstract
{
    use FooTrait;
}
```

```
}

// Barcha yaratilgan treytlarni olamiz

$array = get_declared_traits();

var_dump($array);

/**
 * Natija:

* array(1) {
*   [0] =>
*   string(23) "Example\FooTrait"
* }
*/
```

__toString sehrli metodi

PHP da 2ta ostgi chiziq bilan boshlanuvchi metodlar **sehrli metodlar** deb ataladi. Bunday metodlarning sehrli deb atalishiga sabab, ular biron-bir ish-harakat bajarilganda dasturchi ta'sirisiz avtomatik chaqiriladi.

Siz bilan o'rGANADIGAN birinchi sehrli metodimiz **__toString** metodi hisoblanadi. U sinfning obyekti satrga o'girilganda avtomatik chaqiriladi. **Satrga o'girish** tushunchasini yaxshisi amaliyot kesimida ko'rganimiz afzal. Bizda mana bunday **User** sinfi mavjud:

```
<?php

class User
```

```
{  
  
    private $name;  
  
    private $age;  
  
  
    public function __construct($name, $age)  
    {  
  
        $this->name = $name;  
  
        $this->age = $age;  
    }  
  
  
    public function getName()  
    {  
  
        return $this->name;  
    }  
  
  
    public function getAge()  
    {  
  
        return $this->age;  
    }  
  
}  
?>
```

Keling, ushbu sinfning obyektini yaratamiz:

```
<?php  
  
$user = new User('Sanjar',  
                25); // sinfni obyekti yaratildi  
?>
```

Endi ushbu obyektini **echo** orqali chiqarishga harakat qilamiz:

```
<?php  
  
$user = new User('Sanjar', 25);  
  
echo $user; // echo orqali urunib ko'ramiz  
  
?>
```

Obyekt ma'lumot turini echo orqali chiqarish bu **satrga konversiya** qilish hisoblanadi. Bu holatda PHP xatolik chiqaradi, sababi obyekt satrga o'girilmaydi.

Xatolikni oldini olish uchun biz **phpga** obyekt satrga o'girilsa nima qilish kerakligini aniq qilib aytishimiz kerak. Buning uchun bizda sehrli **__toString** metodi mavjud.

Agar sinfimizda mana shunday metod bo'ladigan bo'lsa, bu metodning natijasi(ya'ni return orqali qaytarilgan) obyektning satr ko'rinishi bo'ladi.

Keling, agar obyekt echo orqali chiqarilsa, xatolikni o'rniga shunchaki foydalanuvchining ismi chiqsin. Demak, **__toString** yaratamiz va unda qaytariluvchi natija sifatida name xossasining qiymatini beramiz:

```
<?php  
  
class User  
  
{  
  
    private $name;  
  
    private $age;  
  
  
    public function __construct($name, $age)  
    {
```

```
$this->name = $name;  
  
$this->age = $age;  
  
}  
  
  
// ko'rsatilgan metodni realizatsiya qilamiz  
public function __toString()  
{  
  
    return $this->name;  
  
}  
  
  
public function getName()  
{  
  
    return $this->name;  
  
}  
  
  
public function getAge()  
{  
  
    return $this->age;  
  
}  
  
}  
  
?>
```

Tekshiramiz:

```
<?php  
  
$user = new User('Sanjar', 25);  
  
echo $user; // natija 'Sanjar' - hammasi ishlayapti!  
  
?>
```

__get sehrli metodi

Biz siz bilan o'rghanadigan keying sehrli metodimiz bu **__get** deb nomlanadi. Bu metod yo'q yoki yopiq xossalari(ya'ni **private** yoki **protected**)ning qiymatini o'qishga urunishda chaqiriladi.

Agar qaysidir sinfda **__get** metodi realizatsiya qilinsa, unda mavjud bo'lмаган yoki yopiq xossalarga murojaatlar mana shu metod orqali amalga oshadi.

Bunda PHP avtomatik tarzda so'rالган xossaning nomini ushbu metodning birinchi parametri sifatida uzatadi, ushbu metoddan qaytarilayotgan qiymat xossaning qiymati sifatida qabul qilinadi.

Agar unchalik tushunarli bo'lмаган bo'lsa, unda amaliyotda tushunib olasiz. Bizda mana bunday yopiq va ochiq xossalarga ega bo'lган **Test** sinfi mavjud:

```
<?php

class Test

{
    public $prop1 = 1;
    private $prop2 = 2;

    public function __get($property)
    {
        return $property; // shunchaki xossaning nomi qaytaramiz
    }
}
```

?>

Keling, endi ushbu yaratgan metodimizning ishlashini tekshirib ko'ramiz. Uch turdag'i xosssaga murojaat qilamiz: ochiq, yopiq va mavjud bo'limgan:

```
<?php

$test = new Test;

// Ochiq xosssaga murojaat:
echo $test->prop1; //

natija 1 - ushbu xossanining asl qiymati


// Yopiq xosssaga murojaat qilamiz:
echo $test->prop2; //

natija 'prop2' - ushbu xossanining nomi


// Mavjud bo'limgan xosssaga murojaat:
echo $test->prop3; //

natija 'prop3' - xossanining nomi

?>
```

Yuqorida ko'rganingizdek, bizning sehrli metodimiz yopiq va mavjud bo'limgan xosssaga murojaatga reaksiya berayapti, lekin ochiq xossga esa avval qanday ishlagan bo'lsa, hozir ham xuddi shunday ishlamoqda.

__set sehrli metodi

__set sehrli metodi mavjud bo'limgan yoki yopiq xossanining qiymatini o'zgartirishga harakat qilinganda chaqiriladi.

Parametr sifatida u xossaning nomi va qiymatini qaul qiladi. Keling, uni namunada ko'ramiz. Bizda mana bunday **Test** sinfi mavjud:

```
<?php

class Test

{

    private $prop1;

    private $prop2;

}

?>
```

Keling ushbu sinfda **var_dump** orqali xossaning nomi va o'sha xossaga o'rnatilishi kerak bo'lgan qiymatni chiqaruvchi **__set** sehrli metodini yaratamiz:

```
<?php

class Test

{

    private $prop1;

    private $prop2;

    public function __set($property, $value)
    {
        var_dump($property . ' ' . $value);
    }
}

?>
```

\$property va **\$value** parametrlari har qanday nomlanishi mumkin. Asosiysi, ular mavjud bo'lsa bas. Bu parametrga

PHP bиринчи параметрга хоссанинг номи ва иккинчисига қиymatini yuboradi

Sinfimizni ishlashini tekshirib ko'ramiz:

```
<?php  
  
$test = new Test;  
  
$test->prop = 'value'; //  
    __set metodidagi var_dump natijasi 'prop value'  
  
?>
```

Keling, endi **\$property** o'zgaruvchisida turgan xossaga қiymat kiritamiz:

```
<?php  
  
class Test  
  
{  
  
    private $prop1;  
  
    private $prop2;  
  
  
    public function __set($property, $value)  
    {  
        $this->$property =  
            $value; // қiymat kiritamiz  
    }  
  
}  
  
?>
```

Endi biz sinfning tashqarisidan turgan holda yopiq хоссанинг қiymatini o'zgartirishimiz yoki yangi қiymat berishimiz mumkin:

```
<?php  
$test = new Test;  
  
$test->prop1 = 1; // 1 kiritdik  
$test->prop2 = 2; // 2 kiritdik  
?>
```

Qiymatni kiritishimiz mumkin, lekin, u yopiq xossaga yozildimi yoki yo'q, buni tekshirish kerak. Buni biz yopiq xossa deb belgilaganimiz uchun uni qiymatini sinf tashqarisidan ololmaymiz, shuning uchun yana bitta __get sehrli metodimizni sinfda yozamiz va uni ishlatamiz:

```
<?php  
class Test  
{  
    private $prop1;  
    private $prop2;  
  
    public function __set($property, $value)  
    {  
        $this->$property = $value;  
    }  
  
    public function __get($property)  
    {  
        return $this->$property;  
    }  
}
```

?>

Mana endi sinfimizni ishlayotganligini tekshirish imkoniga ega bo'lrik:

```
<?php  
  
$test = new Test;  
  
  
$test->prop1 = 1; // 1 ni kiritdik  
$test->prop2 = 2; // 2 ni kiritdik  
  
  
echo $test->prop1; // natija 1  
echo $test->prop2; // natija 2  
  
?>
```

Nom maydonlari – kirish

2ta bir xil nomli sinflar mavjud bo'lgan PHP skriptni ishga tushirganimizda, ular fata xatolikka sabab bo'ladigan konfliktga duch kelishadi. Bu o'z navbatida yaxshi emas, shuning uchun har doim unikal nom yozishga harakat qilish kerak.

Namuna sifatida quyidagi holatni keltiramiz: sizda admin va foydalanuvchi mavjud bo'lgan veb-sayt bor. **Users** papkasida foydalanuvchi uchun sinflar, **admin** papkasida esa – admin uchun mavjud bo'ladi.

Admin va foydalanuvchi uchun saytning qaysidir sahifasiga javob beradigan **Page** sinfi kerak deylik. Foydalanuvchilar uchun ham, admin uchun ham bir xil sinf bo'ladi. Bunday holatda bizni nomlar konflikti kutib turadi desak ham bo'ladi.

Bu konfliktni oldini olishning eng oson usuli – sinflarga bir-biriga o'xshamagan nomlar berish kerak, misol uchun, **UsersPage** va AdminPage. Lekin, bu usul asta-sekin sinflarning nomini kengayib ketishiga sabab bo'lib qolishi mumkin(vaziyatga qarab).

PHP da muammoni hal etadigan yana bir usul mavjud – nomlar maydoni. Maqsadi: har bir sinf qanaqadir nom maydoniga tegishli bo'ladi va sinflar nomining unikalligi faqat o'sha nom maydonida kuzatilishi kerak.

Ya'ni, muammoni yechish uchun quyidagicha qilamiz: **Page** sinfini biror nom maydoniga belgilang,misol uchun, **Users**, va ikkinchi **Page** sinfini boshqa nom maydoniga, misol uchun, **Admin**.

Namunada ko'ramiz. Keling, foydalanuvchi uchun va admin uchun belgilangan sinflarni, yuqorida ko'rsatilgan sinflar konfliktini oldini olish uchun turli xil nom maydonlariga toifalaymiz. Misol uchun, **/admin/page.php** faylidagi Page sinfi uchun, **Admin** nom maydonini yaratamiz:

```
<?php  
namespace Admin;  
  
class Page  
{  
  
}  
?>
```

/users/page.php faylidagi Page sinfi uchun **Users** nom maydonini yaratamiz:

```
<?php  
  
namespace Users;  
  
  
class Page  
  
{  
  
  
}  
  
?>
```

End /index.php faylida Page sinfning birinchi va ikkinchi obyektini yaratamiz:

```
<?php  
  
require_once '/admin/page.php';  
  
require_once '/users/page.php';  
  
  
$adminPage = new \Admin\Page;  
  
$usersPage = new \Users\Page;  
  
?>
```

Quyi nomlar maydoni

Endi bizda qiyinroq vaziyat bor: admin uchun 2ta **Page** sinfini yaratish kerak – birinchishi sahifaning ma'lumotlari, ikkinchisi esa u ma'lumotlarning ko'rinishi. Birnichi Page sinfimiz /admin/data/page.php faylida turadi, ikkinchisi esa /admin/view/page.php.

Yuqorida biz admin sinfidagi barcha sinflarni Admin nom maydoniga bog'lab olgan edik. Lekin, endi bunday maydonda

ikkita sinflarning konflikti bo'ladi. Bu muammoni yechish uchun qo'shimcha nom maydonini yaratish kerak. Misol uchun, Admin nom maydonini yaratish mumkin, uni ichida Data va View quyi maydonlarni yaratamiz. Bunday holatda bunday maydonlar nom maydonining original nomidan keyin shunchaki teskari slesh bilan ajratib yoziladi.

Bu yerda bir narsani aniq aytolaman: ichma-ich joylashadigan quyi maydonlarning soni cheklanmagan. Unda, yuqoridagi namunamizni amalda yozib chiqamiz.

/admin/data/page.php faylidagi **Page** sinfi uchun Admin\Data nom maydonini ko'rsatamiz:

```
<?php  
namespace Admin\Data;  
  
class Page  
{  
  
}  
?>
```

/admin/view/page.php faylidagi **Page** sinfi uchun Admin\View nom maydonini ko'rsatamiz:

```
<?php  
namespace Admin\View;  
  
class Page  
{
```

```
}
```

```
?>
```

/index.php faylida ushbu sinflarning obyektlarini yaratamiz:

```
<?php
```

```
require_once '/admin/data/page.php';
```

```
require_once '/admin/view/page.php';
```



```
$adminDataPage = new \Admin\Data\Page;
```

```
$adminViewPage = new \Admin\View\Page;
```

```
//ishladi
```

```
?>
```

Eslatma!

Yuqorida yozilgan namunada nom maydoni nomi fayl saqlanadigan papka nomi bilan mos. Bunga amal qilish yaxshi amaliyot ya'ni ingliz tilida **good practice** deyiladi, lekin majburiy hisoblanmaydi.

Nom maydonlariga murojaat qilishning qisqa usuli
Bizda **Controller** sinfi mavjud:

```
<?php
```

```
namespace Admin;
```



```
class Controller
```

```
{
```



```
}
```

?>

Shuningdek, **Controller** dan meros olayotgan **Page** sinfi ham:

```
<?php  
namespace Admin;  
  
class Page extends \Admin\Controller  
{  
  
}  
?>
```

Ko'rib turganingizdek, meros olishda biz ota sinf nomi bilan birga nom maydonining nomini ham ko'rsatayapmiz. Ushbu namunaning bitta nozik tomoni bor: ikkala sinf ham bitta nom maydoniga tegishli. Bunday holatda sinfga murojaat qilishda uning nom maydonini yozish kerak emas, sababi **Page** sinfi o'zi **Controller** sinfining nom maydonida:

```
<?php  
namespace Admin;  
  
class Page extends Controller  
{  
  
}  
?>
```

Nisbiy manzil

Index.php faylida quyidagicha chaqiruv amalga oshayapti:

```
<?php
```

```
namespace Admin\Data;  
  
new \Core\Controller;  
?>
```

Avval o'rganganingizdek, sinfga murojaat qilishda sinf nomidan oldin uning nom maydoni teskari slesh bilan yozilardi. Lekin bu hart emas. Agar slesh yozilmasa, nom maydoni joriy nom maydoniga nisbatan belgilanadi. Namunaga qarang:

```
<?php  
namespace Admin\Data;  
  
new Core\Controller; // \Admin\Data\Core\Controller ga  
ekvivalent  
?>
```

use buyrug'i va nom maydonlari

Aytaylik bizda **Data** sinfi mavjud:

```
<?php  
namespace \Core\Admin;  
  
class Data  
{  
    public function __construct ($num)  
    {  
  
    }  
}
```

```
}
```

```
?>
```

Va shuningdek o'zida **Data** sinfining obyektini yaratadigan **Page** sinfi ham mavjud:

```
<?php

namespace Users;

class Page
{
    public function __construct()
    {
        $data1 = new \Core\Admin\Data('1');

        $data2 = new \Core\Admin\Data('2');
    }
}

?>
```

Ko'rganingizdek, ikkala sinfimiz ham umuman boshqa-boshqa nom maydonlarida joylashgan, shuning uchun o'tgan darsimizda qilinganidek **Data** sinfini chaqirishni osonlashtirish mumkin emas.

Mana shunday sinfni chaqirishlar, juda uzun nomlanib va noqulay bo'lib qoladi. Har Data sinfini ishlatmoqchi bo'lganingizda siz uni nom maydonini ham yozishingiz kerak bo'ladi, bu ham vaqt ni oladi ham ko'zingizni charchatadi.

Bunday muammoni yechish uchun maxsus use buyrug'i mavjud. Bu buyruq orqali sinfni uning to'liq nomi bilan birga bir marta ulash kifoya, keyin unga faqatgina o'sha sinfning

nomi bilan murojaat qilishingiz mumkin bo'ladi. Namunga qarang:

```
<?php

namespace Users;

use \Core\Admin\Data; // sinfni ulaymiz


class Page extends Controller

{

    public function __construct()

    {

        $data1 = new Data('1');

        // sinf nomi bilan chaqiramiz

        $data2 = new Data('2');

        // sinf nomi bilan chaqiramiz

    }

}

?>
```

Bir nechta sinflarni ulash

Agar siz bir nechta sinflarni ulamoqchi bo'lsangiz, unda birinchi yoki so'nggi ulangan sinfdan bitta pastga tushib, ulamoqchi bo'lgan sinflingizni yozasiz:

```
<?php

namespace Users;

use \Core\Admin\Datas; // birinchi sinf ulandi
use \Core\Admin\Datases; // ikkinchi sinf ulandi


class Page extends Controller
```

```
{  
    public function __construct()  
    {  
        $data1 = new Data1;  
        $data2 = new Data2;  
    }  
}  
?>
```

use komandasi va nisbiy sinf manzili

use komandasidan foydalanayotganimizda nisbiy fayl manzilini ko'rsatishimiz mumkin bo'ladi, xuddi o'tgan darsimizda qilganimizdek. Keling ushbu gapimizni amaliyotda tasdiqlaymiz:

```
<?php  
  
namespace Core\Admin;  
  
use \Core\Admin\Path\Router; // sinfni ulayapimz  
  
  
class Controller extends Router  
{  
  
}  
?>
```

Ko'rganingizdek, ulanayotgan sinfning nom maydoni hozirgi nom maydoni bilan mos tushadi. Bu ushbu qism bilan boshlanadigan joyni olib tashlash mumkinligini anglatadi:

```
<?php  
  
namespace Core\Admin;
```

```
use Path\Router;  
// nisbiy sinf manzili  
  
class Controller extends Router  
{  
  
}  
?>
```

Topshiriq 1:

Quyidagi kodni **use** dan foydalanib soddalashtiring:

```
<?php  
  
namespace Project;  
  
  
class Test  
{  
  
    public function __construct()  
    {  
  
        // bitta sinfning 3ta obyektini yaratamiz:  
  
        $data1 = new \Core\Users\Data('user1');  
  
        $data2 = new \Core\Users\Data('user3');  
  
        $data3 = new \Core\Users\Data('user3');  
  
    }  
  
}  
?>
```

Topshiriq 2:

Quyidagi sinf berilgan:

```
<?php  
namespace Core\Admin;  
  
class Controller  
{  
  
}  
?>
```

```
<?php  
namespace Users;  
  
class Page extends \Core\Admin\Controller  
{  
  
}  
?>
```

use komandasidan foydalanib, sinfdan meros olish kodini soddashtiring.

Topshiriq 3:

use komandasidan foydalanib, quyidagi kodni soddashtiring:

```
<?php  
namespace Project;  
  
class Test  
{
```

```
public function __construct()  
{  
    $model = new \Core\Admin\Model;  
    $data  = new \Core\Users\Storage\Data;  
}  
}  
?>
```

Topshiriq 4:

Quyidagi kodni **use** komandasidan foydalanib soddalashtiring:

```
<?php  
  
namespace Core\Storage;  
  
  
class Model  
{  
    public function __construct()  
    {  
        $database  = new \Core\Storage DataBase;  
    }  
}  
?  
?>
```

Nom maydonlarida taxallusli sinflar

Boshqa-boshqa nom maydonlariga tegishli bo'lgan 2ta **Data** sinfimiz mavjud. Qaysidir sinfimizga ushbu sinflarning obyekti kerak bo'lib qoldi:

```
<?php  
  
namespace Project;
```

```
class Test

{
    public function __construct()
    {
        $data1 = new \Core\Users\Data; // obyekt
        yaratilyapti
        $data2 = new \Core\Admin\Data; // obyekt
        yaratilyapti
    }
}
?>
```

Ho'p, biz sinfni chaqirishni soddalashtirish uchun **use** komandasidan foydalanamiz. Bunday holatda bizni bitta muammo quchoq ochib kutib turibdi: ikkala sinf nomi ham **Data**, bu esa nomlar konfliktiga olib keladi.

```
<?php

namespace Project;

// Nomlar konflikti bo'lad

use \Core\Users\Data; // birinchi sinfni ulaymiz
use \Core\Admin\Data; // ikkinchi sinfni ulaymiz


class Test

{
    public function __construct()
    {
```

```
$data1 = new Data;  
  
$data2 = new Data;  
  
}  
  
}  
  
?>
```

Bu muammoni yechish uchun maxsus as nomli komanda mavjud, bu komanda bir xil nomga ega bo'lgan yoki shunchaki ulanayotgan sinflar uchun taxallus beradi, va aynan o'sha taxallusga ega bo'lgan sind skriptda ishlatalishi mumkin bo'ladi.

Keling birinchi Data sinfini **UsersData** sifatida belgilaymiz, ikkinchisini esa **AdminData** sifatida belgilaymiz:

```
<?php  
  
namespace Project;  
  
use \Core\Users\Data as UsersData;  
  
use \Core\Admin\Data as AdminData;  
  
  
class Test  
  
{  
  
    public function __construct()  
    {  
  
        $data1 = new UsersData;  
  
        $data2 = new AdminData;  
  
    }  
  
}  
  
?>
```

Topshiriq 1:

Quydagi kodni use komandasidan foydalanib soddalashtiring:

```
<?php

namespace Project;

class Test
{
    public function __construct()
    {
        $pageController = new \Resource\Controller\Page;
        $pageModel      = new \Resource\Model\Page;
    }
}

?>
```

Topshiriq 2:

Quydagi kodni use komandasidan foydalanib soddalashtiring:

```
<?php

namespace Project\Data;

class Test
{
    public function __construct()
    {
        $pageController = new
\Project\Data\Controller\Page;
        $pageModel      = new \Project\Data\Model\Page;
    }
}
```

```
}
```

```
?>
```

Sinflarni avtoyuklash

Allaqachon o'rgangansizki, qanaqadir sinfdan foydalanish uchun uni biz require orqali ulashimiz kerak bo'ladi. Loyihada sinflar soni ko'p bo'lganda, require komandasidan ko'p foydalanishga to'g'ri keladi, bu esa juda qiyin va nazorat qilish oson emas.

Bunday muammoni yechish uchun PHP sinflarni avtoyuklashni qo'shgan.

Avtoyuklash bizga kodda qanaqadir sinfga kirishga harakat qilinayotganda sinfning fayli bilan avtomatik yuklashga imkon beradi. Biroq, sinflar oddiy yo'l bilan yuklanmaydi – ular saytning qanaqadir papkasida maxsus yo'l bilan joylashtirilishi kerak. Fayllar va papkalarni nomlashda qanaqadir konvensiyaga amal qilish zarur. Siz PHP da maxsus ichki konvensiyadan foydalanishingiz mumkin yoki o'zingiznikiga mosidan foydalanishingiz ham mumkin.

Keling, PHP ning standart konvensiyasidan foydalanishdan boshlaymiz. Bu konvensiya shundan iboratki, agar bizda ma'lum sinfga ega bo'lgan fayl mavjud bo'lsa unda ushbu faylga olib boradigan papkalar manzili faylning nom maydoni bilan hamda fayl nomi, unda saqlanayotgan sinfning nomi bilan mos kelishi kerak. Bu holatda, papka va fayl nomlari kichik harflarda yoziladi.

Amaliyotda sinab ko'ramiz. Bizda **Core\Admin\PageController** sinfi mavjud. bu ushbu sinf

/core/admin/ papkasida **pagecontroller.php** faylida joylashishi kerakligini anglatadi.

Avtoyuklashni yoqish uchun faylning standart bo'yicha fayldagi sinf chaqirilgan qismidan tepada `spl_autoload_register` funksiyasi chaqiriladi.

Misol uchun, bizda **index.php** fayli bor, uning ichida esa **Core\Admin\PageController** sinfi mavjud. Faylimiz konvensiyaga muvofiq joylashgan. Keling, **index.php** faylidagi ushbu sinfning obyektini yaratamiz, sinfni **require** orqali emas balki **avtoyuklash** xususiyati orqali ulaymiz:

```
<?php  
  
spl_autoload_register(); //avtoyuklashni yoqamiz  
  
  
$obj = new Core\Admin\PageController; // bemalol obyekt  
yaratamiz  
  
?>
```

Eslatma!

spl_autoload_register funksiyasi faylning boshida faqat bir marta yoziladi. Shundan keyin turli xil sinflarning obyektini qancha bo'lsa shuncha marta yaratish mumkin bo'ladi, asosiysi ularning nomlanishi konvensiyaga amal qilingan bo'lsa bas.

Xotima

Assalomu alaykum, aziz kitobxon. Bu dasturlash tili bo'yicha o'zbek tilida chiqarilgan ilk kitob hisoblanadi. Kitobimizning asl maqsadi sizning PHP bo'yicha bilimingizni oshirish va uni takomillashtirishdan iborat. Kitob **MySQL** baza, **PHP 7** dasturlash tili hamda **obyektga yo'naltirilgan dasturlash** kabi mavzularni o'z ichiga oladi. Bu kitobdagi kontentlarning barchasi tekshirilgan hamda aniq ma'lumotlar asosida, eng muhim dasturchi tomonidan yozilgan. Kitob muallifi tutorials.uz onlayn ta'lim platformasi asoschisi – **Sanjarbek Sobirjonov** hisoblanadi. Kitobni yozishdan maqsad, O'zbekistonda dasturlash sohasidagi manbalarni, o'zbek tilining qadrini oshirish va ko'paytirish hisoblanadi. Ajab emas, bu kitobimizdan so'ng boshqa dasturlash tillari bo'yicha ham onlayn elektron kitoblash yaratilsa. Kitobdan o'qiganlaringizni, o'rganganlaringizni barchasini amaliyotda ko'proq sinang. Kitobni shunchaki o'qib, o'rgandim deb olish – bu juda ham katta xatolik. Sizni darajangizni oshiradigan eng muhim kimsa – bu o'zingiz, eng muhim narsa – bu amaliyotdir.

Hurmat ila,

Sanjarbek Sobirjonov

tutorials.uz onlayn ta'lim platformasi asoschisi.