ЎЗБЕКИСТОН АЛОҚА ВА АХБОРОТЛАШТИРИШ АГЕНТЛИГИ
ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ
САМАРҚАНД ФИЛИАЛИ

Ахборот ва педагогик технологиялар факультети

«Информатика ва ахборот технологиялари» кафедраси
5521900-«Информатика ва ахборот технологиялари»,
4-курс талабалари учун

PHP ТЕХНОЛОГИЯСИ

Фани бўйича

Юсупов Р.А., Муродов Ў.М.

АМАЛИЙ МАШҒУЛОТЛАРИНИ БАЖАРИШ УЧУН УСЛУБИЙ КЎРСАТМАЛАР

САМАРҚАНД-2008

Юсупов Р., Мурадов У. PHP технологияси фанидан амалий машғулотларини бажариш учун услубий кўрсатмалар -Самарқанд, ТАТУ СФ, 2008. – 27 бет

Тақризчилар: Абдукаримов А. – ТАТУ СФ «Умумкасбий фанлар» кафедраси доценти
Абдуллаев А. – СамДУ «Ахборот технологиялари» кафедраси катта ўқитувчиси

Услубий кўрсатма ТАТУ Самарқанд филиали ўқув-услубий кенгашининг 2008 й. _____ № _____ қарори билан нашрга тавсия қилинган.

Мундарижа

Кириш

Web дастурлаш технологиясида мижоз-сервер технологияси кенг ишлатилади. Мижоз махсус сўровлар ёрдамида сервердан керакли ресурсларни сўрайди. Сервер мижоз талабига кўра унга керакли ресурс ёки маълумотларни беради. Бунда мижоз статик саҳифани сўраган бўлса, сервер статик тузилган саҳифани беради. Бу усул ҳамма вақт ҳам қулай ҳисобланмайди. Фойдаланувчиларнинг аксарият қисми динамик равишда ўзгариб турувчи маълумотлардан иборат саҳифани олишни ҳоҳлашади. Бунинг учун бизга PHP тили ёрдам қилиши мумкин. PHP дан ташқари махсус CGI дастурлари ҳам мавжуд бўлиб, улар C++, JAVA ва PERL тилларида тузилади. PHP нинг улардан асосий фарқи шундаки, PHP сценарийлари HTML кодининг ичида ёзилади. Бундан ташқари сценарийни ёзиш учун фақат PHP ишлатилмайди, яъни сценарийларни JavaScript ёки VBScript тилларидан фойдаланиб ҳам яратиш мумкин. Бу тилларда ёзилдан сценарийлар PHP да ёзилган сценарийлардан шуниси билан фарқ қиладики, улар мижоз томонида бажарилади.

PHP билан ишлаш учун сервер дастурлари керак. Бугунги кунда сервер дастурларнинг энг кўп тарқалганларидан бири APACHE дастуридир. Сервер дастурни ўрнатиш ва унга PHP модулини қўшиш одатдаги дастурларни ўрнатиш каби амалга ошади. PHP кодини ёзишда одатдаги матн муҳарирлардан фойданилади.

1-амалий машғулот. Мавзу: Apache Web-сервери ва PHP модулини ўрнатиш

Бу амалий дарсда Web-серверлардан энг кўп тарқалганларидан бири Apache Web-сервер дастурини Windows операцион тизимида ўрнатиш ва PHP модулини ўрнатиб, уни Apache Web-сервер дастури билан ишлаш учун созлаш қаралади. PHP да оддий сценарийлар яратилиб, уларни қайта ишлаш ва уни броузерда тасвирлашлари қандай бажарилиши кўриб чиқилади:

Apache серверини ўрнатиш ва созлаш

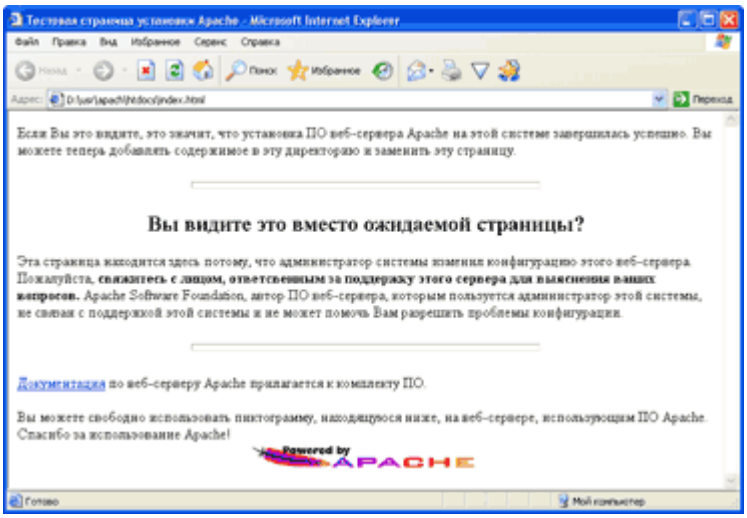
Бугунги кунда энг кенг тарқалган Web-серверлардан бири Apache ҳисобланади. Apache Web-серверини Windows операцион тизимига ўрнатиш ва созлаш ҳеч қандай муҳим муаммони келтириб чиқармайди. Қуйидаги ҳаракатларни бажариш керак.

1. Махсус http://www.apache.org Web–тугунидан Apacheнинг ўрнатувчи (масалан, apache_setup.exe) файлини юкланг.
2. Юкланган файлни ишга туширинг ва Web–серверни Windowsнинг стандарт иловаси каби ўрнатинг (масалан, C:/Apache каталогига). Ўрнатиш жараёни муваффақиятли якунлангандан кейин, ҳамма зарурий ўзгартиришлар ўз ўрнида қолиши учун компьютерни қайта юкланг.
3. Apache серверини созлаш учун қуйидаги ҳаракатлар бажарилиши керак.
   - C:\Apache\conf каталогига кириб ва у ерда жойлашган конфигурацияли http.conf файлини очинг.
   - ServerAdmin директиваси сатрида, сервердаги хатолар тўғрисидаги хабарлар қўшилиб борадиган электрон почтанинг манзилини кўрсатинг. Масалан, ServerAdmin my@email.com.
   - ServerName директиваси сатрини изоҳдан ҳоли қилинг (#белгисини ўчиринг) ва унга талаб қилинган қийматни ўрнатинг, масалан, ServerName localhost.
   - DocumentRoot директиваси сатрида .html файллари сақланиши керак йўлни кўрсатинг, масалан, DocumentRoot “C:/Apache/htdocs”.
   - DirectoryIndex директиваси сатрида Web-сервернинг қайсидир каталогга html–ҳужжат номини аниқ кўрсатмаган ҳолда мурожаат қилганда автоматик бажарадиган файл номини кўрсатинг, масалан, DirectoryIndex index.htm index.html.

- ScriptAlias директиваси сатрида CGI-сценарийлари жойлашадиган йўлни кўрсатинг, масалан, ScriptAlias /cgi-bin/ “C:/Apache/cgi-bin/”.
- AddHandler директиваси сатрида CGI–сценарийлари каби қараладиган файл кенгайтмасини киритинг, масалан, AddHandler cgi-script .bat .exe .pl .cgi.

Барча талаб этилган ўзгартиришлар киритилгандан кейин Apache Web-серверини ишга тушириш учун Пуск ⇨ Все программы ⇨ Apache Web Server ⇨ Start Apache as Console app (Start ⇨ All Programs ⇨ Apache Web Server ⇨ Start Apache as Console app). Бунда экранда консолли мулоқот ойнаси ҳосил бўлади.

Web-сервернинг иш ҳолатдалигини ва тўғри ўрнатилганлигига ишонч ҳосил қилиш учун броузерни ишга туширинг ва адреслар сатрида http://localhost URL адресини киритинг. Агар ўрнатилиш мувафаққиятли тугаса броузер ойнасида 1-расмдагига ўхшаш саҳифа ҳосил бўлади.



1-расм. Apache Web-серверини ўрнатиш мувафаққиятли амалга ошганлиги ҳақида хабар берадиган саҳифа

PHP модулини ўрнатиш ва созлаш

Apache Web-сервери ўрнатилгандан кейин модулини ўрнатиш мумкин. Windows операцион тизимида PHP модулини ўрнатиш автоматик (бунда модулнинг керакли кўп ишлатиладиган қисмлари ўрнатилмайди) ва қўлда амалга оширилиши мумкин.

PHP модулини автоматик ўрнатиш

1. http://www.php.net Web-тугунига киринг, Downloads ва Windows Binaries бўлимларини очинг ва номида –installer сатри бўлган .exe кенгайтмали файлини ишга туширинг (масалан, php406-installer.exe).
2. Юкланган файлни ишга туширинг ва PHP модулини Windows стандарт иловасидек ўрнатинг (масалан, C:\Apache\php катологига). Ўрнатиш жараёнида Apache ҳолатидаги каби фойдаланиладиган Web-сервер ҳам кўрсатилади.

PHP модулини қўлда ўрнатиш

1. http://www.php.net Web-тугунига киринг, Downloads ва Windows Binaries бўлимларини очинг ва .zip кенгайтмали файл-архивни юкланг (масалан, php-4.0.6-Win32.zip).
2. Юкланган файлни талаб этилган каталогга архивдан очинг, масалан, C:\Apache\php.

Бу босқичда PHP ни ўрнатилди деб ҳисоблаш мумкин. Фақат Apache Web-серверини созлаш қолди. Бу қандай амалга ошишини кўрамиз.

PHP билан ишлаш учун Apache Web-серверини созлаш

Бунинг учун, Apache Web-сервери PHP сценарийсини корректни қайта ишлаши учун керакли мос ўзгартиришларни http.conf конфигурацион файлига киритиши зарур бўлади. Бунинг учун қуйидаги ҳаракатлар амалга оширилади.

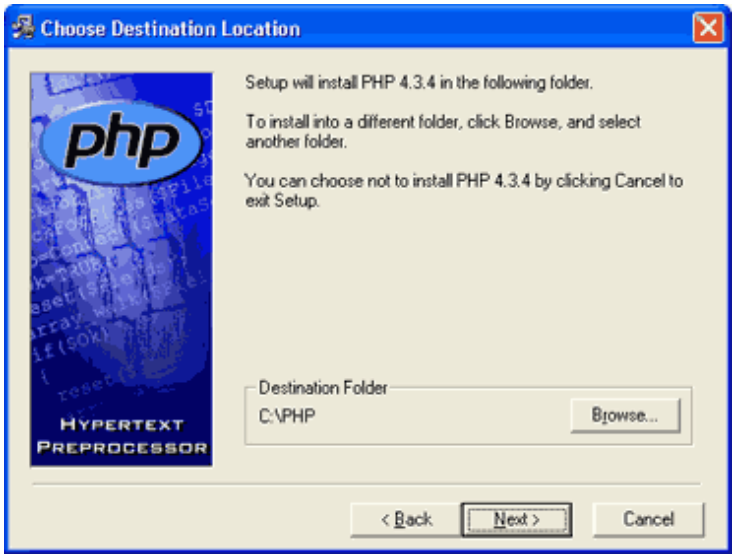
1. AddType application/x-httpd-php .php сатри топилади ва изоҳдан ҳоли қилинади. Шундан кейингина .php кенгайтмали файлларни қайта ишлашда Apache Web-сервери интерпретаторига мурожаат қилинади.
2. ScriptAlias директиваси сатрида ScriptAlias /php/ “PHP_модулига_йул” қийматларини кўрсатинг (масалан, ScriptAlias /php/ “C:/Apache/php/”). Бу директива ёрдамида PHP модулига йўл берилади.
3. Action application/x-httpd-php “php/php.exe” директиваси учун қийматларини кўрсатинг. Бу параметр қайта ишловчи application/x-httpd-php турдаги барча файлларни боғлаш имкониятини беради.
4. httpd.conf файлини сақлаб Web-сервери қайтадан юкланади. Бу эса киритилган ўзгаришларни кучга киришига олиб келади.

Модулининг ишга яроқлигини текшириш учун қўйидаги ҳаракатларни амалга оширинг.

1. C:/Apache/htdocs каталогда қуйидаги код билан test.php файлини яратинг.

```
<?
phpinfo();
?>
```

2. Шундан кейин броузерни қайтадан юкланг URL http://localhost/test.php адресини киритинг.
3. Агар Web-сервер ва PHP модуллари тўғри ўрнатилган бўлса, у ҳолда сиз мониторда PHP тили тўғрисидаги маълумотни кўрасиз. Қайсики бу маълумот 2-расмда кўрсатилган.



2-расм. PHP модули ҳақида маълумотлар браузер ойнасида

Топшириқлар:

1. Apache серверини Windows операцион тизими учун ўрнатиш.
2. Apache серверини Windows операцион тизими учун созлаш.
3. PHP моделини Windows операцион тизими учун автоматик ўрнатиш.
4. PHP моделини Windows операцион тизими учун қўлда ўрнатиш.
5. Windows операцион тизимида PHP моделини Apache сервери билан ишлаш учун созлаш.
6. Apache серверини Unix операцион тизими учун ўрнатиш.
7. PHP моделини Unix операцион тизими учун автоматик ўрнатиш.
8. Unix операцион тизими учун Apache ва PHP ни созлаш.
9. Ихтиёрий матнни экранга чоп этадиган сценарий яратинг.
10. Ихтиёрий матнни ўлчами ва рангини ўзгартириб экранга чоп этадиган сценарий яратинг.

2-амалий машғулот. Мавзу: Бошқарувчи операторлар қатнашган сценарийлар яратиш.

PHP нинг ихтиёрий сценарийси ифодадан иборат. Ифода таркибига ўзлаштириш амаллари, функцияни чақириш, яна бошқарувчи операторлар (шартли операторлар, цикллар, тармоқланувчи ва бошқалар) кириши мумкин. Қоидага кўра, ҳар бир ифода нуқта вергул (;) билан тугайди. Ифодалар группалаштирилган ва мураккаб ифодалар кўринишида бўлиши мумкин.

Шартли операторлар. Бу бошқарувчи оператор берилган шартга кўра аниқланган код фрагментини (бўлагини) бажариш ёки ташлаб кетишни билдиради. Қуйида if операторининг умумий кўриниши келтирилган.

```
if (шарт1) {
1-код блоки
}
elseif (шарт2) {
…
}
elseif (шартN) {
N – код блоки
}
else {
альтернатив код блоки
}
```

Шарт1 чин бўлганда (унинг натижаси true қиймат ҳисобланса), 1-код блоки бажарилади, акс ҳолда шарт2 текширилади ва ҳ.к.. агар ҳамма шартли ифодалар ёлғон бўлса, else операторида аниқланган альтернатив код блоки бажарилади.

Масалан,

```
if ($a>$b) {
print “a is bigger than b”;}
elseif ($a==$b) {
print “a is equal b”;}
else {
print “b is bigger than a”;}
```

PHP тилида яна қуйидаги синтаксисли ?: тернар амали мавжуд.

((ифода1)?(ифода2):(ифода3))

Бу конструкция қуйидаги кўринишда бажарилади. Агар ифода1 чин бўлса, ҳамма ифодалар қиймати ифода2 га тенг бўлади. Акс ҳолда (ифода1 ёлғон бўлганда) барча ифода қиймати ифода3 ҳисобланади.

Цикл операторлари.

While цикли. Олд шартли цикл қуйидагича тавсифланади.

```
While (шарт) {
код блоки;
}
```

код блоки шарт чин бўлганда кўп марта бажарилади. Ҳар сафар шарт цикл бошида текширилади. Агар биринчи қадамдаёқ шарт false қиймат қабул қилса, у ҳолда циклнинг код блоки (цикл танаси) бир марта ҳам бажарилмайди.

Қуйида 1 дан 10 гача сонларни чиқариш учун while операторидан фойдананилган мисол келтирилган.

```
<?php
$i=1;
while ($i<=10) {
echo $i++;
?>
```

Do…while цикли. Бу циклда ҳар қадамда циклнинг операторлар танаси бажарилгандан сўнг шарт текширилади. Бошқача айтганда циклнинг операторлар танаси камида бир марта бажарилишини кафолатланади.

```
do {
код блоки;
} while (шарт);
```

Do-while циклнинг ишлатилишига мисол.

```
$i=10;
do {
print $i;
} while ($i>20);
```

Келтирилган мисолда цикл шарти бажарилмаганда ҳам $i ўзгарувчи бир марта чоп этилаяпти.

For цикли. PHP тилида анча мураккаб цикларни for оператори ёрдамида яратиш мумкин. Бунинг учун қуйидаги синтаксисдан фойданилади.

```
for (ифода1;ифода2;ифода3) {
код блоки;}
```

Ифода1 ҳеч қандай шартни текширилмасдан 1-қадам бошланишидаёқ бир марта бажарилади. Одатда, бу ифодадан циклнинг ўзгарувчи – ҳисоблагичини яратиш (инициализация) учун фойданилади. Ҳар бир қадам бошланишида ифода2 текширилади. Агар у чин бўлса, у ҳолда код блоки бажарилади, акс ҳолда цикл бажарилиши тўхтатилади. Ҳар бир қадам охирида ифода3 бажарилади. Ҳар бир ифода бўш бўлиши мумкин. Кенг тарқалган мисолларни қараймиз, for оператори ёрдамида ҳар хил усуллар билан 1 дан 10 гача бўлган сонларни чиқариш.

1-мисол (“классик” усул)

```
for ($i=1;$i<10;$i++) {
echo $i;}
```

2-мисол.

```
for ($i=1;;$i++) {
 if ($i>10) break;
echo $i;}
```

3-мисол.

```
$i=1;
for(;;) {
if ($i>10) break;
echo $i;
$i++; }
```

4-мисол.

```
for ($i=1;$i<=10; echo $i,$i++) {}
```

Турли кўринишларига қарамасдан, 1-мисолда келтирилган “классик” усулни ишлатиш тавсия этилади.

Foreach оператори. Бу оператор PHP нинг 4 – версиясида пайдо бўлди. У массивнинг барча элементлари устида берилган иш-ҳаракатни бажарилишини билдиради. Foreach операторидан фойдаланиш учун қуйидаги синтаксис қабул қилинган.

```
Foreach (массив as $value) {}
Foreach (массив as $key=>$value) {}
```

1-ҳолда ҳар бир қадамда массивнинг жорий элементи қиймати $value ўзгарувчига таъминланади ва массивнинг жорий кўрсаткичи битта элемент олдинга ўзгартирилади (кейинги қадамда берилган ўзгарувчига кейинги элемент қиймати таъминланади). 2-ҳолда ҳам худди шу жараён бажарилади, лекин массив калити қиймати $key ўзгарувчига таъминланади. Масалан,

```
<?php
```

```
$arr=array(‘one’,‘two’,‘three’);
foreach($arr as $value) {
echo ‘Value: ’.$value.‘<br/>’;}
?>
```

Бу мисолда массив элементлари қийматлари чоп этилаяпти.

Вариант оператори. Switch оператори. Берилган оператор бир нечта if инструкциясига ўхшаш. Битта ва айнан шу ўзгарувчининг қийматини турли хил ҳаракатлар бажарилишига боғлиқ ҳолда бошқа бир нечта қийматлар билан солиштириш талаб этилганда, айнан switch операторидан фойдаланиш яхши бўлади.

```
<?php
switch($value) {
 case ‘apple’:
   echo ‘This is a an apple!’;
    break;
 case ‘pear’:
    echo ‘This is a pear!’;
    break;
  default:
     echo ‘Hm… Who knows…’;
 }
?>
```

Агар $value ўзгарувчи қиймати apple ёки pear га тенг бўлса, мос код блоки бажарилади ва switch операторидан чиқилади (break оператори ёрдамида). Агар қиймат санаб ўтилганларининг биронтасига мос келмаса, у ҳолда default калит сўзи ёрдами билан берилган инструкция блоки бажарилади.

Бошқарувни ўзатиш операторлари.

Break оператори. Break оператори for, foreach, while, do-while, switch операторлари ёрдами билан ташкил этилган цикл ва бошқа бошқарувчи конструкцияларнинг бажарилишини тўхтатишда қўлланилади.

Continue оператори. Continue операторидан бажарилаётган цикллар ичида қадамнинг бажарилишини тўхтатиб кейинги қадамга ўтишда фойдаланиш мумкин.

Агар ўзгарувчи қиймати case блоклардан бирининг қиймати билан мос тушса, у ҳолда яқин break операторигача бўлган барча инструкциялар кетма-кетлиги бажарилади. Шунга ўхшаш бу операторни ташлаб кетиш натижасида switch блокидаги ёпувчи қавсгача бўлган барча операторлар кетма-кет бажарилади.

```
Switch ($i) {
 Case 0:
    Echo $i;
  Case 1:
     Echo $i;
   Case 2:
      Echo $i;
}
```

Агар $i ўзгарувчи қиймати 0 дан иборат бўлса, у ҳолда бу фрагментнинг бажарилишида $i ўзгарувчининг қиймати уч марта чиқарилади. Бундай ҳол содир бўлмаслиги учун break операторидан фойдаланиш кеарк.

```
Switch ($i) {
 Case 0:
    Echo $i;
    Break;
  Case 1:
     Echo $i;
```

```
    Break;
  Case 2:
    Echo $i;
    Break;
}
```

return оператори. Функциялар ичида return(arg) ифодани ишлатишда, унинг бажарилиши натижасида тезда ишнинг якунланишига олиб келади ва arg аргументидан қайтарилаётган қиймат сифатида фойданилади.

Бошқарувчи операторлар ёзилишининг альтернатив кўриниши

if, switch, for ва while бошқарувчи оператордан фойдаланишда, ёзувнинг альтернатив формасини ишлатиш мумкин. Бошқарувчи операторлар танасини аниқлайдиган фигурали қавс ўрнига очилувчи (☺ ажратиш-белгисидан фойдаланиш мумкин. Ҳар тўрттала бошқарувчи операторларнинг ҳар бири мос ёпувчи операторларига эга – endif, endswitch, endfor, endwhile. Масалан, қуйидаги иккита while оператори эквивалент.

```
While ($b<10) {
$a=$a+$b; ++$b;
}
while ($a<10):
$a=$a+$b; ++$b;
endwhile;
```

Бундай форманинг асосий қулайлиги дастурий кодни ўқишни енгиллаштиради.

Агар таркибида else оператори бўлган if операторини ишлатишда альтернатив формани ишлатсак, у ҳолда if оператори танасини икки нуқта ёрдами билан очсак, else ёпилувчи оператор бўлади. Ўз навбатида, else оператор танаси икки нуқтадан фойдаланиб очсак, endif оператори ёрдами билан ёпилади. Масалан,

```
If ($a>$b):
$maximum=$a;
print (“maximum is $a”);
else:
 $large=$b;
print (“maximum is $b”);
endif;
```

Топшириқлар:

1. Учта сонни ўсиш (камайиш) тартибида саралаб чиқарадиган сценарий яратинг.
2. Тонер оператори билан учта сондан каттасини (кичигини) топадиган сценарий яратинг.
3. Ойларнинг тартиб номерига кўра номларини чоп этадиган сценарий яратинг.
4. Ҳафта кунларининг тартиб номерига кўра номларини чоп этадиган сценарий яратинг.
5. Сонни туб эканлигини аниқлайдиган сценарий яратинг.
6. 1 дан 100 гача бўлган сонлар ичидан тоқ сонлар йиғиндисини ҳисоблайдиган сценарий яратинг.
7. 1 дан 100 гача бўлган сонлар ичидан жуфт сонлар йиғиндисини ҳисоблайдиган сценарий яратинг.
8. 1 дан 100 гача бўлган сонлар ичидан туб сонларни устун шаклда чоп этадиган сценарий яратинг.
9. 10 дан 500 гача бўлган сонлар ичидан мураккаб сонларни жадвал шаклда чоп этадиган сценарий яратинг.
10. Товарларнинг миғдорлари, нархи ва жами суммасини жадвал шаклда чоп этадиган сценарий яратинг.

3-амалий машғулот. Мавзу: PHP да сўровларни қайта ишлаш.

HTTP баённомаси ва серверга маълумотларни ўзатиш усуллари. Internet иккилик ахборотларни ўзатувчи физик аспектлар билан боғланган физик даражадан тортиб, фойдаланувчи ва тармоқ орасидаги интерфейсни таъминлаб берувчи Амалий даражагача бўлган кўп даражали принцип асосида қурилган.

HTTP (Hyper Text Transfer Protocol-гиперматнларни ўзатиш баённомаси) – бу амалий даражадаги баённома бўлиб, Internetда гиперматнли ахборотларни алмашиш учун яратилган.

HTTP серверга юбориладиган сўровлар мақсадини кўрсатиш учун ишлаб чиқилган усуллар тўпламини кўрсатади. Бу усуллар манба (ресурс)ларнинг топилган жойи (Unisersal Resourse Lacator, URL) ёки уларнинг универсал номи (Unisersal Resourse Name, URN) кўринишидаги ресурсларнинг универсал идентификатори (Unisersal Resourse Identifier) ишлатиладиган берилган усулн қўллаш мумкин бўлган ресурсни кўрсатиш учун ссылка (ҳавола) тартиби бўйига асосланган.

HTTP баённомасини ишлатиш давомида тармоқ орқали хабар Internetнинг почта хабари (RFC-822) форматига ёки хабари (Multiproprose Internet Mail Exchange) форматига ўхшаш форматда жўнатилади.

HTTP SMTP (электрон почта баённомаси), NNTP (ахборотларни ўзатиш баённомаси), FTP (файллани ўзатиш баённомаси), Gopher ва WAIS каби мавжуд Internet – баённомаларига мурожаат қиладиган турли дастурлар ва дастур – шлюзлар орасида алоқа қилиш учун ишлатилади. HTTP бундай шлюзларда ичма-ич дастур-серверлар (proxy) орқали маълумотларни йўқотмасдан жўнатиш учун ишлаб чиқилган.

Баённома сўров/жавоб принципи асосида ишлайди. Сўровчи дастур – мижоз

- мурожаат усули;
- URI манзили;
- баённома версияси;
- жўнатиладиган хабарлар тури ҳақидаги ахборотли хабар (MIME хабарига ўхшаш);

сўровлардан ташкил топган жавоб берувчи дастур-сервер билан ўзаро алоқани таъминлайди.

Сервер жавоби

- баённома версияси ва қайтиш коди (тўғри ёки хато) кирадиган ҳолат сатри;
- сервер ахбороти, метаахборот (яъни хабар тўғрисидаги ахборот) ва хабар танаси кирувчи хабар (MIME га ўхшаш шаклда) дан иборат бўлади.

Баённомада мижоз ва сервер орасидаги боғланишни ким очиши ёки ёпиш кераклиги кўрсатилмаган. Амалиётдан маълумки, боғланишни мижоз очади, сервер эса жавоб юборилгандан кейин ишни тугатади.

Серверга сўровлар қандай формада юборилишини кўриб чиқамиз.

Мижознинг сўров формаси. Мижоз серверга иккита: тўлиқ ёки қисқартирилган формалардан бирида сўров юборади. Биринчи формадаги сўров тўлиқ сўров, иккинчи формадаги эса – оддий сўров дейилади.

Оддий сўров мурожаат усули ва ресурс усулидан иборат. Уни қуйидагича ёзиш мумкин:

<Оддий-Сўров> := <Усул> <Бўш жой белгиси> <Сўралаётган-URI> <янги сатр белгиси>

Усул сифатида GET, POST, HEAD, PUT, DELETE ва х.к. ларни кўрсатишимиз мумкин. Уларнинг энг кўп тарқалганлари билан кейинчалик танишамиз. Сўраладиган URI сифатида кўпинча ресурснинг URL манзили ишлатилади.

Оддий сўровга мисол:

GET http://phpbook.info/

Бу ерда GET – мурожаат усули, яъни сўраладиган ресурсга қўллаш мумкин бўлган усул, http://phpbook.info/ эса сўраладиган ресурснинг URL-манзили.

Тўлиқ сўров ҳолат сатри, бир нечта сарлавҳа (сўров сарлавҳаси, умумий сарлавҳа ёки мундарижа сарлавҳаси) ва сўров танасидан иборат.

Тўлиқ сўровнинг умумий кўринишини қўйидагича ёзиш мумкин:

<Тўлиқ сўров> := <Ҳолат сатри>(<Умумий сарлавҳа>|<Сўров сарлавҳаси>| <Мундарижа сарлавҳаси>|) <янги сатр белгиси> [<сўров мундарижаси>]

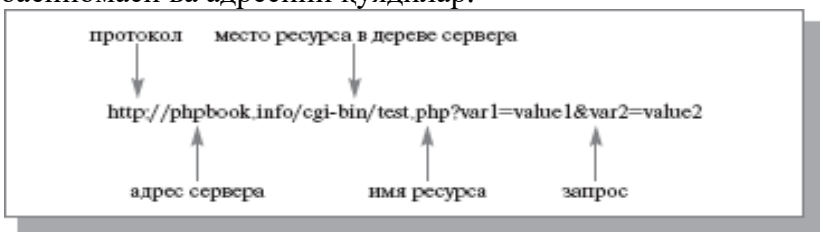
Квадрат қавслар бу ерда сарлавҳанинг зарур бўлмаган элементларини тасвирлайди, вертикал чизиқ эса алтернатив вариантларни кўрсатади. <Ҳолат сатри> элементи сўров усули ва URI ресурси (оддий сўров каби) ва бундан ташқари HTTP баённоманинг мавжуд версияларини ўзида сақлайди. Масалан, ташқи дастурни чақириш учун қуйидаги ҳолат сатрини бажариш керак:

POST http://phpbook.info/cgi-bin/test HTTP/1.0

Бунда POST усули ва HTTP баённомасининг 1.0 версияси ишлатилмоқда.

Сўровнинг ҳар иккала шаклида ҳам сўралаётган ресурснинг URI асосий ўрин эгаллайди. Кўпинча URI ресурснинг URL-манзили кўринишда ишлатилади. Серверга мурожаатда URL нинг тўлиқ шакли каби қисқартирилган шаклини ҳам ишлатиш мумкин (5.2 расм).

Қисқартирилган шаклда сервер асосида ресурсларнинг жойлашган ўрнини кўрсатувчи сервер баённомаси ва адресини қўядилар.



3- расм. URL нинг тўлиқ формаси

Кейинчалик кўп тарқалган сўровларни ўзатиш усулларини кўриб чиқамиз.

Усуллар. Аввал айтилганидек, серверга мижознинг ихтиёрий сўрови усулни кўрсатишдан бошланади. Усул мижохнинг сўров мақсади ҳақида маълумот беради. HTTP баённомаси кўп усуллардан иборат бўлиб, шулардан учтаси POST, GET ва HEAD лар кўпроқ ишлатилади. GET усули ресурс сўровида URI ёрдамида идентификацияланган ихтиёрий маълумотларни олишга ёрдам беради. Агар URI дастурга кўрсатса, у ҳолда у дастурнинг ишлаш натижасини қайтаради, унинг матнини эмас (агар, матн унинг ишини натижаси бўлмаса). Сўровни қайта ишлаш учун керак бўладиган қўшимча ахборот сўровнинг ўзида қурилади (статус старида). Ресурс танаси майдонида GET методини ишлатиш учун махсус талаб қилинган ахборотлар қайтарилади (масалан, HTML-ҳужжатнинг матни).

Топшириқлар:

1. Сервер ишини тушунтириб беринг.
2. Сервер/мижоз деганда нима тушунасиз.
3. Фойдаланувчи логин ва паролини тўғрилигини текширадиган сценарий яратинг.
4. Фойдаланувчи киритган маълумотларни қайта ишлайдиган сценарий яратинг.
5. Фойдаланувчи ёшини таҳлил қиладиган сценарий яратинг.
6. Фойдаланувчи киритган логин ва парол тўғри бўлса, кейинги саҳифани очиб берадиган сценарий яратинг.
7. Фойдаланувчи киритган логин ва паролларидаги ортиқча бўш жойларни олиб ташлайдиган сценарий яратинг.
8. Фойдаланувчи формадаги тўлдириши зарур бўлган майдонларни тўлдирганлигини текширадиган сценарий яратинг.
9. Фойдаланувчи қайси машғулотга аъзо бўлганлигини текширадиган сценарий яратинг.
10. Фойдаланувчи билан итерактив мулоқот қиладиган сценарий яратинг.

4-амалий машғулот. Мавзу: Массивлар. Массивларни тартиблаш. Калитлар буйича массивларни тартиблаш.

PHP тилидаги массивлар бошқа ихтиёрий дастурлаш тилларидаги аналог конструкциялардан фарқ қилади. Улар яхши традицион массивлар ва Perl нинг хэш жадваллари комбинацияси сифатида қаралиши мумкин. Бу уларни жудаям мосланувчан қилади. Массивнинг алоҳида элементлари Perl тилининг хэш-жадвал элементларига ўхшаш: уларнинг ҳар бири “калит-қиймат” жуфтлигини ташкил қилади. Агар PHP массивининг мантиқий структураси бошқа ихтиёрий тиллар массивига ўхшаса, у ҳолда мос калитлар оддий мусбат сонлар ҳисобланади. Бу сонлар ҳар доим ўсиш тартибида жойлашади. Агар PHP массивининг мантиқий структураси Perl хэш-жадвалига ўхшасада, у ҳолда калитлар сатрлардан иборат бўлади, массив элементлар тартиби хэш функцияларни ишлатиш билан аниқланади.

Массивларни яратиш. PHP тилида массивларни икки усул билан яратиш мумкин. Скаляр ўзгарувчиларни яратиш учун ўзлаштириш амалидан фойданилади. Уни массивни яратиш учун қўллаш мумкин. Массив элементига қиймат бериш, ҳали мавжуд бўлмаган массивни яратишга сабабчи бўлади. Масалан, айни вақтда $mass мавжуд эмас. У ҳолда қуйидаги ифода уни яратишни билдиради.

```
$mass[0]=5;
```

Шунга эътибор қилиш керакки PHP массивлари Perl нинг хэш-жадвалига ўхшасада, уларнинг номи бошқа PHP ўзгарувчилари каби $ белгиси билан бошланади. Шунинг учун ўзлаштириш оператори бажарилишига қадар $mass ўзгарувчиси мавжуд бўлса, бу амал бажарилгандан кейин у массивга айланади. Массив элементини қиймат билан таъминлашда унинг индекси кўрсатилмаса, у ноаниқ кўринишда танланади. Бунда индекс бирлиги олдин фойданилган индекс қийматидан биттага кўп бўлади. Агарда массив элементига таъминлаш қўлланилишидан олдин калитли элементлар мавжуд бўлмаса, унда нол калити ишлатилади. Масалан, қуйидаги қатордаги иккинчи калит сифатида икки қиймати қўлланилади:

```
$mass[1]=”Today is my birthday!”;
$mas[]=50;
```

Массив яратишнинг иккинчи усули array операторидан фойдаланиб яратишдир. Бу операторнинг параметри янги массив яратишдаги берилаётган қийматлар ва талаб этилаётган мос калитдан иборатдир. Агар массив традицион усулда яратиш режалаштирилаётган бўлса (калитларсиз), у ҳолда array операторида фақат қийматлар берилиши мумкин (бунда PHP интерпретатори бутун сонли қийматларни калитларга таъминлайди).

Масалан,

```
$mass=array(10,20,30,40);
```

Бундай таъминлашда тўртта элементли ва 0, 1, 2 ва 3 калитли оддий массив яратилади. Агар бошқа калитларни кўрсатиш керак бўлса, у ҳолда қуйидаги кўринишда тасвирлаш керак.

```
$mass=array(1=>10, 2=>20, 3=>30, 4=>40);
```

Агар array операторидан фойдаланиб массив яратишда параметрлар кўрсатилмаса, бўш массив яратилади.

```
$mass=array();
```

Қуйидаги сатрда Perl хэш-жадвалига тўлиқ мос келадиган массив яратилган.

```
$ages=array(“Ivan”=>40, “Mary”=>17,“Alex”=>25);
```

PHP тилидаги массивлар на традицион массивга, на хэш-жадвалга мос келмаслиги мумкин. Улар ўзларида юқоридаги конструкциялар “аралашмасини” мужассам этади. Масалан, қуйидаги ёзув синтаксис жиҳатдан тўғри.

```
$auto=array(“make”=>“AZIK”, “model”=>2110,“year”=>1992, 3=>“Sold”);
```

Массив элементларига мурожаат. Массивнинг алоҳида элементларига мурожаат бошқа дастурлаш тилларидаги каби индекс орқали амалга оширилади. Бунда квадрат

қавсга олинган индекс изланаётган қиймат калити ҳисобланади. Қавслар бутун сонли ёки сатрий калитлар ишлатилишига боғлиқмас ҳолда қўйилади. Масалан, $ages массив элементи қийматини “Mary” калити билан қуйидаги кўринишда олиш мумкин.

Print(“Mary is $ages[‘Mary’] years ild
”);

Массивнинг бир нечта элементлари қийматини битта оператор билан скаляр ўзгарувчиларга таъминлаш мумкин. Масалан,

$trees=array(“oak ”,“pine”,“binary”);

$list($hardwood, $softwood, $data_structure)=$trees;

Бу фрагментда $hardwood, $softwood ва $data_structure ўзгарувчиларига мос равишда массив элементларининг (“oak ”,“pine” ва “binary”) қийматлари таъминланаяпти.

PHP ички функцияларидан фойдаланиб массивдан алоҳида калитлар тўпламини ва қийматлар тўпламини ажратиб олиш мумкин. Array_keys функция параметри сифатида массив берилади, натижа массив калитлари қайтади. Бу ҳолда натижавий массив калитлари 0, 1 ва ҳ.к. сонлар ҳисобланади. Array_values функцияси массивдан массив-параметр қийматларини ажратиш учун ишлатилади. Масалан,

$mass=array(“one”=>74,“two”=>70,“three”=>67,“four”=>62,“five”=>65);

$numbers=array_keys($mass);

$values=array_values($mass);

Энди $numbers массиви“one”, “two”, “three”, “four” ва “five қийматлардан, $values ” массиви эса – 74, 70, 67, 62 ва 65 қийматлардан иборат бўлади. Бу массивларнинг калитларини 0, 1, 2, 3 ва 4 лар ташкил этади.

Массивлар устида амаллар бажариш. Ихтиёрий бошқа скаляр ўзгарувчилар каби массивни ўчириш unset() функцияси ёрдамида амалга оширилади. Бу функциядан фойдаланиб, массивнинг алоҳида элементини ҳам ўчириш мумкин. Масалан, қуйидаги кўринишли

$mass=array(2,4,6,8);

$unset($mas[2]);

Фрагментда, энди $mass массивида 2, 4 ва 8 элементлар, 0, 1 ва 3 калитлар билан учта элемент қолди.

Is_array функцияси is_int функциясига ўхшаш: унинг параметри сифатида ўзгарувчи ўзатилади, агар бу ўзгарувчи массив бўлса true қиймат, акс ҳолда false қиймат қайтади. Иккита count ва sizeof функциялари параметр сифатида массивни олади ва унинг элементлар сонини қайтаради. In_array функцияси иккита кириш ифода ва массив параметридан иборат, агар ифода қиймати массив элементлари қийматларидан бири ҳисобланса true қиймат, акс ҳолда false қийматларни қайтаради.

Foreach оператори массивнинг ҳамма элементларини қайта ишлашдаги цикларни ташкил этиш учун махсус яратилган. Бу операторнинг иккита кўринишда ишлатиш мумкин.

Foreach(<массив> as <скаляр ўзгарувчи>) <цикл танаси>;

Foreach(<массив> as <калит>=><қиймат>) <цикл танаси>;

Foreach операторининг биринчи кўринишидан фойдаланилганда циклнинг ҳар бир итерациясида скаляр ўзгарувчи массив қийматларининг бирини ўзлаштиради. Биринчи қадам бажарилгандан кейин массивнинг жорий кўрсаткичи reset функциясидан фойданилган каби ошкормас ўрнатилади (инициализация қилинади).

Масалан,

Foreach($mass as $temp) {

Print(“$temp
”);

…}

Бу фрагмент бажарилгандан кейин $mass массивининг барча элементларининг қийматлари олинади.

Foreach операторининг иккинчи кўринишидан массивнинг ҳамма элементлари қийматлар каби ва калит каби олишда фойданилади. Масалан, қуйидаги кўринишда.

```
$temperature=array(“January”=>-23, “February”=>-18, “March”=>10);
foreach($temperature as $month=>$temp) {
print(“The low temperature on $month was $temp<br/>”);
…}
```

Массивларни саралаш. Sort функциясининг параметри сифатида массив номи берилади, бу функция массивда сақланаётган қийматларни саралаш бажарилади, кейин эса калитлар ўчирилади. Массивларда ҳам сатрли, ҳам сонли қийматлар сақланиши мумкин. Бу ҳолда сатрли қийматлар массивнинг бошларида алфавитли тартибда жойлаштирилади. Сўнгра сонли қийматлар ўсиш тартибида жойлашади. Бунда берилган массив калитларга боғлиқмас ҳолда тартибланган массив калитлари 0,1,2 ва ҳ.к. лардан иборат бўлади. Sort функциясини фақат сатрли ёки сонли қийматлардан иборат оддий массивларни тартиблашда фойдаланган маъқул.

Sort функция иккинчи SORT_NUMERIC параметрига эга. Ундан берилган массивнинг сонли қийматларини ўсиш тартибида саралашда фойданилади, сатрли қийматлар ўзгармасдан қолади. Барибир сатрли қийматлар массивнинг бошларида жойлаштирилади.

Asort функциясидан Perl тилининг хэш-жадвалига ўхшайдиган массивларни тартиблашда ишлатилади. Бу функция берилган массив элементларини бошланғич “калит-қиймат” алоқаларини сақлаган ҳолда тартиблайди. Sort функцияси каби asort функцияси ҳам сатрли қийматларни массивнинг бошларига жойлаштиради, кейин эса сонли қийматларни ўсиш тартибида жойлаштиради. Asort функциясида ҳам SORT_NUMERIC параметридан фойдаланиш мумкин.

Rsort ва arsort функциялари мос ҳолда sort ва asort ларга ўхшаш. Ягона фарқи шундаки, улардан тескари тартибда саралашда фойданилади.

Масалан.

Қуйидаги сценарий sort, rsort, asort ва arsort функцияларини ишлашини намойиш этади. Бунда шуни ёдда сақлаш керакки, сценарий файли жимлик ҳолда ҳужжатларни сақлаш учун фойданиладиган –сервер катологида жойлашиши керак. Бундан ташқари унутмангки...

```
<!DOCTYPE html PUBLIC “//w3c//DTD XHTML 1.0 strict//EN” “http://www.w3c.org/TR/xhtml/DTD/xhtml-strict.dtd”>
<!--example.php – Simple example to illustrate several sorting functions-->
<html>
<head>
<title>example.php (Sorting)</title>
</head>
<body>
<?php
$original=array(“one”=>10, “two”=>20, “three”=>30, “four”=>“tree”);
?>
<h4>Orginal Array</h4>
<?php
foreach($original as $key=>$value){
print(“[$key]=>$value<br/>”);}
$new_array=$original;
sort($new_array);
?>
<h4>Array sorted with sort function</h4>
<?php
foreach($new_array as $key=>$value){
print(“[$key]=>$value<br/>”);}
$new_array=$original;
```

```
sort($new_array, SORT_NUMERIC);
?>
<h4>Array sorted with sort function and SORT_NUMERIC parameter</h4>
<?php
foreach($new_array as $key=>$value){
print("[$key]=>$value<br/>");}
$new_array=$original;
rsort($new_array);
?>
<h4>Array sorted with rsort function</h4>
<?php
foreach($new_array as $key=>$value){
print("[$key]=>$value<br/>");}
$new_array=$original;
asort($new_array);
?>
<h4>Array sorted with asort function</h4>
<?php
foreach($new_array as $key=>$value){
print("[$key]=>$value<br/>");}
$new_array=$original;
arsort($new_array, SORT_NUMERIC);
?>
<h4>Array sorted with arsort function</h4>
<?php
foreach($new_array as $key=>$value){
print("[$key]=>$value<br/>");}
?>
</body>
</html>
```

Топшириқлар:

1. PHP да массивларнинг яратиш усуллари.
2. Массив элементларига мурожаат қандай амалга оширилади.
3. Foreach оператори нима учун ишлатилади.
4. Массивларни саралашда қанақа функциялардан фойданилади.
5. Берилган массивда, берилган қиймат неча марта такрорланишини аниқлайдиган сценарий яратинг.
6. Фойдаланувчи нечта машғулотга аъзо бўлганлигини аниқлайдиган сценарий яратинг.
7. Хэш жадвалдаги маълумотларни калит ва қийматларни алоҳида ажратувчи сценарий яратинг.
8. Хэш жадвалдаги маълумотларни фақат қийматларини тартиблайдиган сценарий яратинг.
9. Хэш жадвалдаги маълумотларни калит-қиймати сақланган ҳолда тартиблайдиган сценарий яратинг.
10. Берилган массивни барча саралаш усуллари билан саралаб натижаларни кетма-кет экранга чоп этадиган сценарий яратинг.

5-амалий машғулот. Мавзу: Каторлар. Катор ичида элементларини кидириш.

PHP да сатр – бу байтлар кетма-кетлигидир. PHP да Unicode ни қўлловчи ички функциялар мавжуд эмас. Сатрли ўзгармаслар (литераллар) битталик ёки иккиталик тирноқлар ёрдами билан аниқланиши мумкин.

Махсус белгилар (масалан, \' ва \\) сатрларда битта тирноққа (апостриф) олинган бўлса, ажратилмайди. Бу ўзгарувчиларга ҳам таалуқли. Бундай ўзгармаслар чиқариш потокларида ҳар қандай ўзгартиришда ҳам чиқарилади.

Қўштирноққа олинган сатрий ўзгармасларни қайта ишлаш умуман бошқача. Айрим ҳолларда PHP интерпретатори ўзгарувчиларни танийди ва уларнинг ўрнига жорий қиймати чиқарилади.

Масалан,

'The age is: $age' сатри ҳеч қандай ўзгартиришларсиз чиқарилади. Агарда $age ўзгарувчининг жорий қиймати 12 дан иборат бўлса, у ҳолда

"The age is: $age"

сатрини чоп этишда қуйидаги кўринишда ўзгартирилади:

The age is: 12

Бир нечта мисоллар келтирилган.

Echo 'Бу сатр'; //Бу сатр чиқарилади

Echo 'Бу \'сатр\'даги "сатр"';// Бу 'сатр'даги "сатр" чиқарилади

Echo 'a\nb';//a\nb чоп этилади

Биринчи қаторда келтирилган фрагментга изоҳ керак эмас. Иккинчи мисолдаги сатр битталик тирноққа олинган. Айнан шунинг учун бу сатрнинг ичида бемалол қўштирноқни ишлатиш мумкин, битталик тирноқни эса маскировка қилишга тўғри келади. Учинчисида битталик тирноққа олинган сатрда махсус белгини чиқариш намойиш қилинаяпти. Келтирилган мисолдан кўриб турганингиздек кутилаётган сатрни кўчириш белгисини ўрнига \n сатри чиқарилаяпти.

Агар сатр қўштирноққа олинган бўлса, у ҳолда унда қуйидаги махсус белгиларни ишлатиш мумкин:

\n - сатрни кўчириш;

\r - кареткани қайтариш;

\t - табуляция белгиси;

\$ - доллар белгиси;

\" - қўштирноқ.

Қолган барча ҳолларда тескари слэш ҳеч қандай белгини аниқламайди ва оддий белги сифатида намоён бўлади.

Ўзгарувчи номини ({}) қавс билан чегаралаш мумкин. Масалан, белгисини идентификаторда ишлатиш мумкин бўлмасада, қуйидаги фрагмент

$beer='Heineken';

echo "$berr's taste is great";

бажарилиш натижаси қуйидагича бўлади

Heineken's taste is great

Мана яна битта мисол

Echo "He drunk some $beers";

Echo "He drunk some ${beer}s";

Биринчи сатрни қайта ишлаш натижасида $beer ўзгарувчи қиймати чиқарилмайди, чунки s белгиси идентификаторга тегишли бўлаяпти ва интерпретатор тўғри ечимни аниқлай олмайди. Иккинчи сатрни қайта ишлашда кутилаётган натижага келинади: яъни beer ўзгарувчиси интерполяция қилинади ва унинг қиймати натижавий сатрга қўшилади.

Изоҳ. Сценарий ишини самарадорлигини ошириш учун битталик тирноқни ишлатган маъқул. Бундай ҳолда PHP интерпретатори бошқа тилларга тегишли воситаларни қидириб ўтирмайди.

Сатрлар устида амаллар бажариш. PHP тилида . (нуқта) амалини ишлатиш билан сатрларни бирлаштирувчи (конкатенация) ягона амали аниқланган. Масалан,

$NewString=$aString.$bString;

Сатрларни бирлаштиришда ихтиёрий сондаги сатрлар келиши мумкин. Яъни сатрни қисми сифатида сатрга сонларни ҳам қўшиш мумкин:

$NewString=$aString.$bString.$cNumber;

Бу шунинг учун ҳам ишлайдики, яъни PHP кучли типлашган тил бўлмаганлиги сабабли, бунда $cNumber сон ўзгарувчи автоматик сатрга ўтказилади ва сатрга қўшилади. Сатрни номерлари нолдан бошланувчи белгилар массиви сифатида қараш мумкин ва фигурали қавсда унинг номерини кўрсатиб алоҳида белги сифатида мурожаат қилиш мумкин. Масалан, агар $str ўзгарувчи “abcdefgh” сатрини ўзида сақласа, у ҳолда қуйидаги икки сатрни қайта ишлаш

$first=$str{0}; // ‘a’

$last=$str{strlen($str)-1}; // ‘h’

қуйидаги натижага олиб келади

a

h

Web-браузердан ортиқча бўш жойларни жўнатиш, сценарий ёрдамида маълумотларни қайта ишлашда кўнгилсизликларга олиб келиши мумкин. Масалан, агар паролнинг охирида бўш жой белгиси бўлса, унда у ҳақиқий ҳисобланмайди.

Trim() функцияси сатр бошидаги ва охиридаги (ўртадагиси эмас), ортиқча бўш жойларни автоматик равишда олиб ташлайди. Масалан,

$string=“ extra space before and after text”;

$string=trim($string);

Натижа “extra space before and after text” дан иборат бўлади.

Фақат сатр бошидаги ёки сатр охиридаги бўш жойларни ўчириш учун, мос равишда ltrim() ва rtrim() функциялари ишлатилади.

Сатр қисмини ажратиш. Узун сатрдан кичик қисмини ажратиш мумкин. Бу эса иккита функция ёрдамида амалга оширилади.

Strok() функцияси сатрдан ажратувчи (одатда вергул ёки бўш жой) асосида қисм сатрни ажратади. Масалан, фойдаланувчи битта майдонда исми ва фамилиясини бўш жой билан киритган бўлса, қуйидаги код ёрдамида унинг исмини ажратиш мумкин:

$FirstName=strok($Name,“ ”);

Иккинчи усул – сатр ичидаги позиция индексга мурожаат қилиб. Бунинг учун substr() функцияси ишлатилади. Масалан,

$substring=substr($String,0,10);

Бунда биринчи параметр сатр келтирилади, иккинчи параметр ажратувчи қисм сатрнинг биринчи ўрни, учинчи параметр эса ажратилувчи белгилар сони бўлади. Агар сатрдаги белгилар сони ажратилувчи белгилар сонидан кам бўлса, қисм сатр сатр охири билан тугайди.

Массив ва сатрлар ўртасида ўзгартиришларни бажариш кўпинча фойдали бўлади. Буни implode() ва explode() функциялари ёрдамида амалга ошириш мумкин. Explode() функцияси биринчи параметр ажратувчи белги сифатида қаралиб, сатрни қисм сатрларга ажратади ва уларни массив элементларига жойлаштиради. Иккинчи параметр эса сатр ҳисобланади. Масалан,

$str=‘April in Kiev, Sevastopol is nice’;

$words=explode(“ ”,$str);

Бу фрагмент бажарилгандан кейин $words массиви ‘April’, ‘in’, ‘Kiev,’, ‘Sevastopol’, ‘is’ ва ‘nice’ қийматлардан иборат бўлади.

Implode() функциясининг ҳаракати explode() функциясига қарама-қаршидир. Параметрлар сифатида ажратувчи-белги ва массив олинади, бу функция массив

элементларини бириктиришни бажаради. Бунда элементлар ўртасида ажратувчи-белги қўйилади. Қайтариладиган қиймат сатр бўлади. Масалан,

$words=array(“Are”, “you”, “working”, “today”);

$str=implode(“ ”,$words);

Бу фрагмент қайта ишлангандан кейин $str сатри Are you working today қийматдан иборат бўлади.

Топшириқлар:

1. PHP да сатр тушунчаси нима.
2. Сатрлар устида қандай амаллар мавжуд.
3. Сатр қисмини ажратиш қандай амалга оширилиши мумкин.
4. Implode() функциясининг вазифаси нимадан иборат.
5. Берилган сатрни, берилган белги бўйича ажратиб массив ташкил қилувчи сценарий яратинг.
6. Массив элементларини берилган белги бўйича бирлаштириб сатр ҳосил қилувчи сценарий яратинг.
7. Берилган сатрда, берилган қисм сатр мавжудлигини текширувчи сценарий яратинг.
8. Берилган сатрдаги барча бош ҳарфларни кичик ҳарфларга айлантириб берувчи сценарий яратинг.
9. Берилган сатрни кодлайдиган сценарий яратинг.
10. Кодланган сатрни очадиган сценарий яратинг.

6-амалий машғулот. Мавзу: Фойдаланувчи тамонидан аникладиган функциялар. Функция аргументлари.

Функциялар фавқулотда муҳим ва аниқ механизм битта катта масалани ечишда майда қисм масалаларга бўлаклашда фойданилади. Функция – бу номланган дастурлаш кодининг фрагменти бўлиб, яратувчи томонидан берилган аниқ ҳаракатларни бажаради.

Функцияни тавсифлаш. Функцияларни тавсифлаш учун қуйидаги синтаксис ишлатилади.

```
function Функция_номи ([параметрлар])
{
//Функция танаси
…
}
```

Квадрат қасвга олинган ифодани маъноси шулки, функция параметрлар рўйхати бўш бўлиши мумкин. Агарда сценарийда битта функциянинг иккинчи тавсифи ҳам мавжуд бўлса, у ҳолда хато ҳақида хабар генерация қилинади. Битта функциянинг ўзини иккинчи марта тавсифлаш, қайта аниқлаш ёки қўшимча параметрлар билан тўлдириш тақиқланади, яъни бир хил ном билан ҳар хил параметрли функцияларни яратиш мумкин эмас.

Функция танаси ихтиёрий код бўлиши, ўз навбатида фойдаланувчининг янги функциялари ёки класслари тавсифланиши келиши мумкин. Бошқача айтганда, функция ичма-ич тавсифланади.

Фойдаланувчи функцияларини тавсифи дастурнинг ихтиёрий жойида, ҳатто инструкциялар орасида ҳам келиши мумкин. Бу дастур кодини тушуниш қийинлаштирганлиги сабабли тавсия этилмайди. Масалан,

```
Инструкция1;
Function MyFunction(…) {…}
Инструкция2;
```

PHP тилида функциялар ихтиёрий турдаги қийматларни қайтариши ёки ҳеч нима (жимлик ҳолда қайтадиган қиймат false га тенг бўлади) қайтармайди. Функция бирданига бир нечта қиймат сифатида бир нечта элементдан иборат бўлган массивни кўрсатиш орқали қайтаради. Функциядан чиқиш, ёки return операторидан фойдаланиб, ёки танасидаги барча операторлари бажарилиши натижасида амалга оширилади. Масалан,

```
//Бу функция ҳеч қандай қиймат қайтармайди.
//Функциянинг бижарилиши, функция танасидаги барча
//инструкциялар бажарилгандан кейин тугайди.
function Func1()
{
echo “Wonderful function”;
}
//Бу функция ҳам ҳеч қандай қиймат қайтармайди.
// Функциянинг бажарилишининг тугаши return оператори
//билан амалга оширилади. Бу шуни билдирадиики,
//охирги сатр бажарилмайди.
function Func2()
{
echo “Other wonderful function”;
return;
echo “Useless string”;
}
//Бу функция ҳар доим 0 қийматни қайтаради.
function Func3()
{
return 0;
}
```

Функцияни чақириш қуйидаги кўринишда амалга оширилади:

```
функция_номи($arg1, $arg2,…);
```

PHP тилида функция ўзгарувчилари (variable function) деб номланувчи концепция киритилган. Бу имкониятдан қуйидагича фойдаланилади. Ўзгарувчи яратилади, кейин унинг қиймати сифатида мавжуд функция номи кўрсатилади. Энди бу ўзгарувчи ёрдами билан функцияни чақириш мумкин. Бундай конструкция функцияга мурожаатни беради. Масалан,

```
Function DefinedFunction() {..}
$pseudoname=“DefinedFunction”;
$pseudoname(); //DefinedFunction() функцияга мурожаат
```

Юқорида айтганимиздек, функция (ички) бошқа функциянинг (ташқи) ичида ҳам аниқланиши мумкин. Бунда ички функцияни, фақат ташқи функцияни чақиргандан кейин, дастурнинг ихтиёрий қисмидан чақириш мумкин. Акс ҳолда интерпретатор аниқланмаган функцияни чақиришга ҳаракат қилинаяпти деб “қарор” қабул қилади. Бундан ташқари, бош функция бир мартадан ортиқ чақирилса, бу албатта хатоликка олиб келади, чунки PHP интерпретатори буни қуйи функцияни бир мартадан ортиқ тавсифлашга ҳаракат бўлаяпти деб “ҳисоблайди”. Бундай ҳол бўлишига сабаб шуни, одатда қуйи функция бош функциянинг ичида тавсифланади.

Қуйидаги имкониятлар бўлиши мумкин.

```
if (шарт)
{
F функцияни тавсифлаш
}
else
{
```

```
Бошқа F функцияни тавсифлаш
}
```

Яъни F функция, амалда бажарилиш босқичида тавсифланади. Нима бўлгандаям, биринчидан, ҳар қандай чақириладиган функциянинг тавсифланганлигига, иккинчидан, ҳеч бир функциянинг икки марта тавсифланмаслигига эътибор бериш зарур.

Функцияга параметрларни узатиш. Параметрлар рўйхати вергул билан ажратилган бир нечта ўзгарувчилардан иборат бўлади. Бошқа дастурлаш тиллари каби, функцияларни чақиришдаги параметрлар фактик (actual parametr), функцияларни тавсифлашдаги параметр – формал (formal parametr) деб номланади. Фактик параметрлар сифатида ихтиёрий ифодалар ишлатилиши мумкин. Формал параметрлар ҳар доим ўзгарувчиларнинг исми бўлиши лозим.

Функцияларни чақиришдаги фактик параметрлар миғдори унинг формал параметрлар сони билан мос келмаслиги мумкин. Агар функция чақирилишида аниқ параметрлар сони формал параметрларда кўрсатилган параметрлар сонидан кам бўлса, “ортиқча” формал параметрлар мослигини ўрнатишда бу параметрлар алоқасиз ўзгарувчилар сифатида қаралади. Агар функцияни чақирилишида аниқ параметрлар сони кўрсатилган формал параметрлар сонидан кўп бўлса, унда ортиқча аниқ параметрлар эътиборга олинмайди. Функцияларга бундай яқинлашиш функцияларни ўзгарувчан сонли параметрлар билан ишлатишга имкон яратади.

PHP тилининг 4-версиясида функцияларга параметрлар узатишда қийматлари бўйича, мурожаат усулида, ҳамда жимлик қоидаси асосида беришни таъминлайди.

Функцияни учта параметр билан аниқлаш қуйидаги кўринишда бўлади.

```
Function A($arg1,$arg2,$arg3);
```

Бу функцияга мурожаат эса қуйидагичадир.

```
A(2,$b, $c);
```

Функцияга параметрларни узатишнинг бундай усули *параметрларни қийматлари бўйича узатиш* деб номланади. Бу эса функция параметрлари унинг локал ўзгарувчилари бўлишини билдиради. Улар устида бажарилаётган ишларда қийматлари фактик параметрларда берилган глобал ўзгарувчиларга ҳеч қандай таъсир кўрсатилмайди. Масалан, қуйидагича.

```
<?php
$a=5; //глобал $a ўзгарувчисини тавсифлаш
function Inc($a)
{
$a++; //функция локал ўзгарувчининг қийматини ўзгартиради ва
return $a; //унинг қийматини қайтаради
}
$b=Inc($a);    // Яна битта $b ўзгарувчиси
echo “$a $b”; // 5 ва 6 қийматлари чиқарилади, чунки Inc функцияси $a глобал
                //ўзгарувчи билан эмас, балки унинг $a нусхаси билан ишлайди
?>
```

Мисолдан кўриниб турибдики, бундай кўринишда тавсифланган функциядан фойдаланилганда ҳар бир фактик параметрлар учун унинг локал нусхаси яратилади.

Умуман олганда, PHP 4 тилида жимлик қоидасига асосан параметрлар ва функциялар қийматлари бўйича берилади (5-версияда аниқлигигача эътиборга олинади). Бу шуни билдирадики, мазмунан, фактик параметрлар қийматлари функциянинг формал параметрларига боғлиқ ҳолда оператив хотира соҳасига нусхаланади. Шу билан биргаликда формал параметр қийматлари ҳеч қачон фактик параметр сифатида берилган соҳага нусхаланмайди. Шундай қилиб айтиш мумкинки, параметрларни узатишда функция билан ўзаро алоқада қийматлар бўйича бир йўналишли боғлиқлик амалга оширилади. Параметрларни узатишнинг бу усули жуда кўп ишлатилади. Аммо айрим ҳолларда функциялардан фойдаланишда икки йўналишли алоқадан фойдаланиш

зарурияти пайдо бўлади. Масалан, бу усул функция битта қийматдан кўп қийматларни қайтарганда жуда фойдали бўлади. Бундай усулдан фойдаланишда параметрларни узатишда фактик қийматларни узатмасдан уларнинг адресларини узатиш тавсия этилади. Бунда фактик параметрларнинг ўзгариши мос равишда формал параметрларнинг ҳам ўзгаришига олиб келади. Функциялардан бундай фойдаланиш усули параметрларни мурожаат (по ссылка) бўйича узатиш дейилади.

Айрим ҳолларда функция параметрларини нусхалаш керак эмас, агар узатилаётган параметрлар катта ҳажмли объектлардан ташкил топса, чунки уларни нусхалаш кўп вақт ва сезиларли даражада хотирани банд қилади. Параметрларни мурожаат бўйича узатишни амалга оширишнинг иккита усули мавжуд. Биринчи усулга фактик параметр номи олдига & белгисини қўшиш киради. Бу белгини функцияда тавсифланаётган формал параметр номининг олдига қўшиш ҳам мумкин. Иккала усуллар ўзаро эквивалентдир. Аниқ усулни танлаш яратувчининг ўзига боғлиқ.

Масалан,

```
function add_one ($first, $second, &$third)
{
$first++;  $second++;  $third++;
}
$a=3;  $b=3;  $c=3;
add_one($a, &$b, $c);
print("\$a, \$b and \$c are now: $a, $b and $c<br/>");
```

Бу фрагмент бажарилгандан кейин қуйидаги натижа ҳосил бўлади.

```
$a, $b and $c are now: 3, 4 and 4
```

Яна бир мисол келтирамиз.

```
$x=5; //глобал ўзгарувчини яратиш
//оддий функцияни аниқлаш
function A($arg1) {…}
//функцияга $x глобал ўзгарувчининг мурожаатини узатиш
A(&$x);
//ҳар доим мурожаатни қабул қиладиган
//функцияни аниқлаш
function B(&$arg) {…}
//$x ўзгарувчининг нусхаси эмас, мурожаати узатилади
B($x);
```

Параметрларни мурожаат бўйича узатиш функция ичида глобал ўзгарувчиларни ишлатишга жуда ўхшаш. Бу шуни билдирадики, мурожаат бўйича узатилган параметрларни ўзгариши, унга мос келувчи қийматни ўзгартириши мумкин. Ёддга олиш керакки, формал параметрлар олдидан амперсант (&) белгиси ишлатилса ҳар доим функция мурожаатни узатишни кутади. Шунинг учун унинг аргументлари сифатида ўзгармасларни узатиш мумкин эмас. Бу ерда ҳам, агар функция …

Жимлик қоидаси бўйича қийматлар кўриниши фақат ўзгармаслар бўлиши керак. Қуйида бир нечта мисоллар келтирилган.

```
<?php
$x=5;
//одатдаги тавсифлаш …
function A($arg) {…}
//бу функция ҳар доим мурожаатни кутади
function B(&$arg) {…}
//нотўғри, 3 ўзгармаси мурожаат ҳисобланмайди
function A(&$arg=3) {…}
A($x); //$x аргументини қиймати бўйича узатиш
A(&$x); //$x аргументни мурожаат бўйича узатиш
```

B($x); //$x аргументни мурожаат бўйича узати
B(&$x); //$x аргументни яъна мурожаат бўйича узати
B(5); //нотўғри, чунки функция мурожаатни кутади
?>

Жимлик бўйича қиймат сифатида фақат ўзгармасдан фойданилади. …

Масалан,

function A($arg) {…}
//$arg параметр учун жимлик қоидаси бўйича берилмаган
function B($arg=5) {…}
//$arg параметр учун жимлик қоидаси бўйича 5 берилган
function C($arg=$def) {…}
//нотўғри, жимлик бўйича қиймат фақат ўзгармас бўлиши керак

Қуйидаги мисолда B() функцияни чақиргандаги биринчи иккитаси бир-бирига тенг кучли.

B(); //B(5) деб чақириш билан тенг кучли
B(5); //функцияни аниқ қиймат билан чақириш
B(3); //функцияни 3 параметр билан чақириш

Агар функция аргументларининг бир нечтаси жимлик қоидаси бўйича, бошқалари оддий ҳолатда берилган бўлса, у ҳолда бу параметрлар бошқа параметрлардан кейин туриши керак.

function A($arg1, $arg2=0, $arg3=1) {…} //тўғри
function B ($arg1, $arg2=5, $arg3) {…} //нотўғри
A(-1); // A(-1,0,1) чақириш билан тенг кучли
A(); //нотўғри, чунки бита параметр жимлик қоидаси билан аниқланмаган
//ва уни аниқ кўрсатиш талаб этилади

Топшириқлар:

1. PHP да фойдаланувчи функцияси қандай тавсифланади?
2. Ўзгарувчиларнинг кўриниш соҳаси нима?
3. Ўзгарувчиларнинг яшаш даври нима?
4. Глобал ўзгарувчига функция танасида мурожаат қандай амалга оширилади?
5. Статик ўзгарувчи нима ва нима учун ишлатилади?
6. Функциядан фойдаланиб берилган иккита сонни ЭКУБ ини топадиган сценарий яратинг.
7. Функциядан фойдаланиб массив элементларининг энг каттасини топадиган сценарий яратинг.
8. Функциядан фойдаланиб массив элементларининг йиғиндисини топадиган сценарий яратинг.
9. Функциядан фойдаланиб массив элементларининг ўрта арифметигини топадиган сценарий яратинг.
10. Функциядан фойдаланиб берилган сатрда, берилган сўз неча марта такрорланганлигини аниқлайдиган сценарий яратинг.

7-амалий машғулот. Мавзу: Файлли тизим билан ишлаш. Файл хосил килиш. Файл билан богланишни ёпиш.

Ўзгарувчиларни кўриниш соҳаси. PHP тилида ўзгарувчиларнинг кўриниш соҳасига кўра уч турга бўлинади: глобал, локал ва статистик.

Глобал ўзгарувчиларнинг кўриниш соҳаси – бутун дастурдир.

Локал ўзгарувчилар фақат функция ичида мавжуд. Функция ичида тавсифланган ихтиёрий ўзгарувчилар жимлик ҳолда локал бўлади. Функция параметрлари сифатида олинган параметрлар ҳам локал ҳисобланади. Қуйидаги мисолни қараймиз.

```
<?php
$a=5; //$a – глобал ўзгарувчи
function f() {
echo $a; //нотўғри, чунки $a глобал ўзгарувчи …, лекин локал ўзгарувчи
        //ҳали аниқланмаган
$a=4;
$a+=10; //$a локал ўзгарувчини тавсифлаш
echo $a; //локал ўзгарувчининг қийматини 14 ни чоп этиш
$b=2; //Яна битта локал ўзгарувчини тавсифлаш
}
f(); //f функцияни чақириш
echo $a; //$a глобал ўзгарувчини қийматини чоп этиш, яъни 5 ни 14 эмас!
echo $b; //яна хато, $b-f функция ичидаги локал ўзгарувчи, функциядан
        //ташқарида у аниқланмаган
?>
```

Яна бир мисол.

```
function summer($list)
{
$sum=0;
foreach($list as $value)
  $sum+=$value;
return $sum;
}
$sum=10;
$nums=array(2,4,6,8);
$ans=summer($nums);
print("The sum of the values in \$nums is: $ans<br/>");
print("The value of \$sum is still: $sum<br/>");
```

Бу фрагмент қайта ишлангандан кейин қуйидаги маълумот ҳосил бўлади.

```
The sum of the values in $nums is: 20
The value of $sum is still: 10
```

Бошқача айтганда, $sum локал ўзгарувчи шу номли глобал ўзгарувчига ҳеч қандай таъсир қилмайди. Бундай яқинлашишни тушуниш жуда одиий.

Айрим ҳолларда функция ичида глобал ўзгарувчига мурожаат қилиш керак бўлади. Бу иккита йўл билан амалга оширилади. Биринчи йўл бундай ўзгарувчи функция ичида global каби тавсифланади. Бунда берилган ўзгарувчи функциядан олдин аниқланганлиги кутилади. Масалан, бундай.

```
<?php
$a=5; //$a – глобал ўзгарувчини тавсифлаш
function f()
{
global $a; //f функция ичида $a ўзгарувчи глобал каби тавсифланаяпти
echo $a; //барчаси тўғри, ҳозир бу ўзгарувчи аниқланган
$a++; //$a локал ўзгарувчини тавсифлаш
}
f(); //f функцияни чақириш
echo $a; //$a глобал ўзгарувчини 6 қийматини чиқариш
?>
```

Яна битта мисолни қараймиз.

```
$big_sum=0;
function summer($list) {
global $big_sum;
```

```
$sum=0;
foreach($list as $value)
  $sum+=$value;
$big_sum+=$sum;
return $sum;
}
$ans1=summer($list1);
$ans2=summer($list2);
print("The sum of all array elements is: $big_sum<br/>");
```

Иккинчи йўл ҳамма глобал ўзгарувчилар номи ва қийматидан иборат $GLOBALS массивидан фойдаланиб функция ичида глобал ўзгарувчиларга мурожаат қилиш. Бу массив ҳамма функцияларга глобал ҳисобланади. Мисол учун.

```
<?php
$a=5; //$a глобал ўзгарувчини эълон қилиш
function f()
{
$GLOBALS["a"]+=10; //$a глобал ўзгарувчига мурожаат ва унинг
                   //қийматини ўзгартириш
}
f(); //f функцияни чақириш
echo $a; //$a ўзгарувчи қийматини чиқариш, $a ўзгарувчининг
         //f функцияда ўзгартирилган 15 қиймати чиқарилаяпти
?>
```

Ўзгарувчиларни яшаш вақти. Айрим ҳолларда функция олдинги жараёндаги ҳаракатига боғлиқ ҳолда функция ҳаракат жимлик ҳолда PHP да функцияда локал ўзгарувчиларнинг яшаш вақти функцияни чақиргандан то ишини якунланигача давом этади. Функциянинг олдинги ҳаракатаидаги …

Статик локал ўзгарувчиларнинг яшаш вақтини функцияни биринчи чақирганда бошлайди, сценарий бажарилиши якунланиши билан тугайди.

PHP тилида static калит сўзи ёрдами билан локал ўзгарувчини статик каби аниқлаш мумкин. Масалан, қуйидаги функцияни қараймиз.

```
function do_it($param)
{
static $count=0;
$count++;
print("do_it has now been called $count times<br/>");
…
}
```

Келтирилган мисолдаги функция ўзининг чақиришлар миғдорини кўрсатади. do_it функцияни сценарийнинг ихтиёрий жойидан чақирганда ҳам бу қиймат ўзгаради.

Статик ўзгарувчиларнинг кўриниш соҳаси локал ўзгарувчи каби – функция ичидадир. Функцияни ҳар сафар чақирганда яратиладиган локал ўзгарувчилардан фарқли статик ўзгарувчи бир марта яратилади, кейин жорий қийматни функцияни кейинги чақиргангача сақлайди. Манна яна бир мисол.

```
<?php
function f() {
$a=3; //локал $a ўзгарувчини тавсифлаш
static $b=10; //статик ўзгарувчини тавсифлаш
              //f функцияни биринчи марта чақирганда бу ўзгарувчига 10
              //қиймат таъминланади. f функцияни кейинги чақиришгача $b
              //ўзгарувчи ўзининг охирги қийматинисақлайди
$a++; //локал ўзгарувчини ўзгартириш
```

```
$b++; //статик ўзгарувчини ўзгартириш
}
f(); //функцияга биринчи марта мурожаат, 4 ва 11 қийматлари чиқарилади
f(); // функцияга иккинчи марта мурожаат, 4 ва 12 қийматлари чиқарилади
?>
```

Топшириқлар:

1. Файл дискрептори деганда нима тушунилади.
2. Файл кўрсаткичи нима учун ишлатилади.
3. Фойдаланувчи исмларини файлга ёзиб қўйувчи сценарий яратинг.
4. Фойдаланувчини рўйхатга олувчи сценарий яратинг.
5. Фойдаланувчи логини рўйхатда мавжудлигини текширувчи сценарий яратинг.
6. Файллар устида тегишли амаллар бажарувчи сценарий яратинг.
7. Берилган ном бўйича файл мавжудлигини текширувчи сценарий яратинг.
8. Саҳифага неча марта мурожаат бўлганлигини аниқловчи сценарий яратинг.
9. Бошқа иловалар билан маълумотлар алмашишни ташкиллаштирувчи сценарий яратинг.
10. Каталоглар билан ишни ташкиллаштириб берувчи сценарий яратинг.

АДАБИЁТЛАР

1. Куссуль Н.Н., Шелестов А.Ю. Использование PHP. Самоучитель. – М.: Издательский дом «Вильямс», 2005. – 272 с.: ид. – Парал. тит. англ.
2. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. – Пер. с англ. – СПб: Символ – Плюс, 2003 – 1048 с., ил.
3. Котеров Д.В. Самоучитель PHP4. – СПб.: БХВ – Петербург, 2003. – 576 с.: ил.
4. Ульман Л. Основы программирования на PHP: Пер. с англ. – М.: ДМК Пресс, 2001. – 288 с.: ил. (Самоучитель)
5. Колискиченко Д.Н. PHP 5. В теории и на практики. Самоучитель. Изд-во «Нит» Санк-Петербург, 2007, 631 с.
6. Мазуркевич А. PHP: настольная книга программиста /Александр Мазуркевич, Дмитрей Еловой. – Мн.: Новое знание, 2003. – 480 с.ил.
7. Харрис Э. PHP/MySQL для начинающих. /Пер. с англ. – М.: КУДИЦ – ОБРАЗ, 2005, - 384 с.
8. Александр К. и др. Букварь по PHP и MySQL
9. www.intuit.ru