ՖԱՅԼԱՅԻՆ ՀԱՄԱԿԱՐԳԻ ԳՈՐԾԱՌՈՒՅԹՆԵՐԸ

Վեբ հավելվածի համար ընտրված տվյալների պահպանման վայրը տվյալների բազան է: Դա չի նշանակում, որ լիովին հրաժարվում ենք սովորական տեքստային ֆայլերի հետ գործ ունենալուց: Պարզ տեքստային ֆայլերը դեռեւս հարմար եւ բազմակողմանի միջոց են որոշ տեսակի տեղեկատվություն փոխանակելու և պահպանելու համար:

Մենք կարող ենք հեշտությամբ մշակել տեքստային ֆայլում պահված վեբ էջի HTML կաղապարի (շաբլոնի) կոդը: Ֆայլերը նաեւ հարմար են աղյուսակային տվյալների փոխանակման համար: PHP ծրագրերում կարող ենք հեշտությամբ կարդալ եւ գրել CSV ֆայլեր:

Ֆայլերի հետ աշխատանքը ներառում է 3 քայլ:

- Ֆայլի բացում:
- Մշակում (կարդալ, գրել):
- Ֆայլի փակում:

Ֆայլի բացումը կատարում է fopen() ֆունկցիան: Նրան պետք է փոխանցել երկու պարամետրերը՝ առաջինը - ֆայլի անունը (տող), երկրորդը - ռեժիմ (նույնպես տող): Այս ֆունկցիան վերադարձնում է ռեսուրս տիպի արժեք: Ապագայում այն կօգտագործվի այլ ֆունկցիաների կողմից, որոնք աշխատում են ֆայլերի հետ:

Եթե ֆայլը գտնվում է ընթացիկ պանակում, ապա բավական է նշել միայն իր անունը (առանց ուղին նշելու): Եթե նա ընթացիկ պանակում չէ, պետք է նշել ուղին: Ընթացիկ պանակը փոխելու համար օգտագործվում է chdir() ֆունկցիան, որին պետք է տրվի այն պանակի անունը, որը մենք ցանկանում ենք դարձնել ընթացիկ: Եթե պանակը փոխել չի հաջողվում, ֆունկցիան վերադարձնում է false:

Պարզելու համար, թե այս պահին որն է ընթացիկ պանակը, օգտագործվում է getcwd() ֆունկցիան:

Ֆայլի բացման fopen() ֆունկցիայի ռեժիմները.

Ռեժիմ	Կարդալ	Գրել	Ֆայլային ցուցիչ	Մաքրել ֆայլը	Ստեղծել եթե ֆայլը չկա	Սխալ եթե ֆայլը կա
r	այո	ոչ	սկզբում	ոչ	ոչ	ոչ
r+	այո	այո	սկզբում	ոչ	ոչ	ոչ
w	ոչ	այո	սկզբում	այո	այո	ոչ
w+	այո	այո	սկզբում	այո	այո	ոչ
a	ոչ	այո	վերջում	ոչ	այո	ոչ
a+	այո	այո	վերջում	ոչ	այո	ոչ

R, r+, w, w+, a եւ a+ սիմվոլներից հետո կարող է լինել եւս մեկ ոչ պարտադիր սիմվոլ՝ b կամ t : Եթե նշված է b-ն, ապա ֆայլը հասկացվում է որպես երկուական, իսկ եթե նշված է t -ն, ապա ֆայլը հասկացվում է որպես տեքստային:

Ֆայլի մշակումը սովորաբար ներառում է կարդալ եւ/կամ գրել գործառույթները:

Ֆայլից կարդալու համար օգտագործվում են հետեւյալ ֆունկցիաները:

fgets () - կարդում է տող ֆայլից: Առաջին պարամետրը ռեսուրս ցուցիչ է, որը մեզ վերադարձրեց fopen() ֆունկցիան: Երկրորդ պարամետրը՝ ոչ պարտադիր, կարդացվող բայթերի քանակն է: Ֆունկցիան կարդում է նշված քանակով բայթեր կամ պակաս, եթե այն ավելի վաղ է հանդիպում տողի ավարտին կամ ֆայլի ավարտին:

file_get_contents() - ընդունում ֆայլի անունը եւ վերադարձնում դրա բովանդակությունը մեկ տողով: Ձախողման դեպքում, file_get_contents() ֆունկցիան կվերադարձնի false:

file() - կարդում է ֆայլի պարունակությունը և տեղադրում այն զանգվածի մեջ: Ձախողման դեպքում, file() ֆունկցիան կվերադարձնի false:

Վերջին երկու ֆունկցիաներն օգտագործելիս հարկավոր չէ ֆայլը բացել fopen() ֆունկցիայով: Նրանք ամեն ինչ իրենք կանեն:

Ֆայլում գրելու համար օգտագործվում են հետեւյալ ֆունկցիաները:

fputs() - fwrite() ֆունկցիաի կեղծանունը: Առաջին պարամետրը ռեսուրսի ցուցիչն է, երկրորդը՝ այն տողը, որը մենք գրում ենք:

file_put_contents() – գրում է տողը ֆայլում: Այս ֆունկցիան վերադարձնում է ֆայլում գրված բայթերի քանակը:

Օրինակ.

Ընթացիկ պանակում ստեղծել **testfile.php** անունով ֆայլ երեք տողով.

```
Տող 1
Տող 2
Տող 3
```

```
<?php // testfile.php
    $fh = fopen("testfile.txt", 'w') or die("Ֆայլ ստեղծել չհաջողվեց");
    $text = <<<_END
    Տող 1
    Տող 2
    Տող 3
```

```
    _END;
    fwrite($fh, $text) or die("Ձախողում, ֆայլը չգրվեց");
    fclose($fh);
    echo " 'testfile.txt' ֆայլը հաջողությամբ գրվեց ";
?>
```

Օրինակ.

Կարդալ եւ գրել ֆայլերը միանգամից ամբողջությամբ

Այս բաժինը ցույց է տալիս, թե ինչպես պետք է աշխատել անմիջապես ամբողջ ֆայլի հետ, այլ ոչ թե ֆայլի մի քանի տողերի:

Կարդալ ֆայլը: Ֆայլի բովանդակությունը տողի մեջ կարդալու համար օգտագործվում է file_get_contents() ֆունկցիան: Նրան որպես արգումենտ փոխանցվում է ֆայլի անունը, եւ այն վերադարձնում է ֆայլի պարունակությունը պարունակող տող:

Օրինակ 2-ը կարդում է օրինակ 1-ի ֆայլը file_get_contents()-ի օգնությամբ, փոխում է այն, օգտագործելով str_replace(), այնուհետեւ արդյունքը տպում է:

Օրինակ 1: page-template.html-ը օրինակ 2-ի համար

```
<html>
  <head>
    <title>{page_title}</title>
  </head>
  <body bgcolor="{color}">
    <h1>Hello, {name}</h1>
  </body>
</html>
```

Օրինակ 2. file_get_contents() ֆունկցիայի օգտագործումը էջի կաղապարի հետ

```
// բեռնել շաբլոն ֆայլը նախորդ օրինակից
$page = file_get_contents('page-template.html');
// ներդնել էջի վերնագիրը
$page = str_replace('{page_title}', 'Welcome', $page);
// էջը դարձնել կապույտ կեսօրին և կանաչ՝ առավոտյան
if (date('H' >= 12)) {
   $page = str_replace('{color}', 'blue', $page);
} else {
```

```
    $page = str_replace('{color}', 'green', $page);
}
// ստանալ նախորդ սեսիայի փոփոխականում պահված օգտագործողի անունը
$page = str_replace('{name}', $_SESSION['username'], $page);
//տպել արդյունքը
print $page;
```

Եթե $_SESSION ['username'] ստանում է Aram արժեքը, օրինակ 2-ը տպում է՝

```
<html>
  <head>
    <title>Welcome</title>
  </head>
  <body bgcolor="green">
    <h1>Hello, Aram </h1>
  </body>
</html>
```

Գրել ֆայլը: Տողի մեջ ֆայլի բովանդակության ընթերցումը նման է տողի գրառումը ֆայլում: Իսկ file_get_contents() ֆունկցիայի անալոգը file_put_contents() ֆունկցիան է:

Օրինակ 3-ը ընդլայնում է օրինակը 2-ը , պահելով HTML կոդը ֆայլում այն տպելու փոխարեն:

Օրինակ 3. Պահպանել ֆայլը, օգտագործելով file_put_contents()

```
// բեռնել նախկինում օգտագործված շաբլոն ֆայլը
$page = file_get_contents('page-template.html');
// ներմուծել էջի վերնագիրը
$page = str_replace('{page_title}', 'Welcome', $page);
// էջը դարձնել կապույտ կեսօրին և կանաչ՝ առավոտյան
if (date('H' >= 12)) {
    $page = str_replace('{color}', 'blue', $page);
} else {
    $page = str_replace('{color}', 'green', $page);
}
// ստանալ նախորդ սեսիայի փոփոխականում պահված օգտագործողի անունը
$page = str_replace('{name}', $_SESSION['username'], $page);
// արդյունքը գրել page.html ֆայլում
file_put_contents('page.html', $page);
```

Օրինակ 3-ը արձանագրում է $page-ի արժեքը (HTML) page.html ֆայլում: file_put_contents () ֆունկցիայի առաջին արգումենտը գրվող ֆայլի անունն է, իսկ երկրորդ արգումենտը՝ այն է, որ պետք է գրվեր ֆայլում:

ֆայլերի մասերի կարդալը եւ գրելը

File_get_contents() եւ file_put_contents() ֆունկցիաները աշխատում են կատարելապես, երբ աշխատում ենք միանգամից ամբողջ ֆայլի հետ: Բայց երբ ժամանակը գա ճշգրիտ աշխատանքի համար, անհրաժեշտ է օգտագործել file() ֆունկցիան ֆայլի յուրաքանչյուր տող մուտք գործելու համար: Օրինակ 4-ը կարդում է ֆայլը, որտեղ յուրաքանչյուր տող պարունակում է անունը եւ էլեկտրոնային փոստի հասցեն, եւ ապա տպում է այս տեղեկատվության HTML ֆորմատավորված ցուցակը:

Օրինակ 4. Մուտք գործել ֆայլի յուրաքանչյուր տող

```
foreach (file('people.txt') as $line) {
   $line = trim($line);
   $info = explode('|', $line);
   print '<li><a href="mailto:' . $info[0] . '">' . $info[1] ."</li>\n";
}
```

Ենթադրենք, people.txt-ը պարունակում է այն, ինչ նշված է Օրինակ 5-ում.

Օրինակ 5. people.txt-ը օրինակ 4-ի համար

```
alice@example.com|Alice Liddell

bandersnatch@example.org|Bandersnatch Gardner

charles@milk.example.com|Charlie Tenniel

dodgson@turtle.example.com|Lewis Humbert
```

Այնուհետեւ 9-4-ի օրինակը տպում է՝

```
<li><a href="mailto:alice@example.com">Alice Liddell</li>

<li><a href="mailto:bandersnatch@example.org">Bandersnatch Gardner</li>

<li><a href="mailto:charles@milk.example.com">Charlie Tenniel</li>

<li><a href="mailto:dodgson@turtle.example.com">Lewis Humbert</li>
```

file () ֆունկցիան վերադարձնում է զանգված: Այս զանգվածի յուրաքանչյուր տարրը մի տող է, որը պարունակում է ֆայլի մեկ տող, ներառյալ նոր տողը: Այսպիսով,

foreach() ցիկլը օրինակ 4-ում այցելում է զանգվածի յուրաքանչյուր տարր, և նրա արժեքը տեղադրում է $line փոփոխականում: Trim() ֆունկցիան հեռացնում է տողի սկզբի և վերջի բացատները: explode()-ը մասնատում է տողը ըստ “|” սիմվոլի, ապա print-ը արտածում է HTML ցուցակի տարրերը:

Թեեւ file()-ը շատ հարմար է, այն կարող է խնդրահարույց լինել շատ մեծ ֆայլերի հետ աշխատելուց: Այն կարդում է ամբողջ ֆայլը տողերի զանգված կառուցելու համար, այդ պատճառով բազմաթիվ տողեր պարունակող ֆայլը կպահանջի մեծ հիշողություն: Այս դեպքում հարմար է կարդալ ֆայլը տող առ տող, ինչպես ցույց է տրված օրինակ 6-ում՝

Օրինակ 6. Ֆայլի ընթերցումը տող առ տող

```
$fh = fopen('people.txt','rb');
while ((! feof($fh)) && ($line = fgets($fh))) {
   $line = trim($line);
   $info = explode('|', $line);
   print '<li><a href="mailto:' . $info[0] . '">' . $info[1] ."</li>\n";
}
fclose($fh);
```

Օրինակ 6 -ում ֆայլերի մատչելիության չորս գործառույթները fopen(),fgets (), feof () եւ fclose (). Նրանք միասին աշխատում են հետեւյալ կերպ՝

• fopen () ֆունկցիան բացում է կապը ֆայլի հետ եւ վերադարձնում է փոփոխական, որը օգտագործվում է հետագայում ֆայլ մուտք գործելու համար ծրագրում:

* fgets () ֆունկցիան կարդում է տողը ֆայլից եւ վերադարձնում է այն:

* PHP շարժիչը պահում է էջանիշ, որտեղ գտնվում է ընթացիկ դիրքը ֆայլում: Էջանիշը սկսվում է ֆայլի սկզբից, այնպես որ fgets () ֆունկցիայի առաջին կանչով կարդում է ֆայլի առաջին տողը: Տողը կարդալուց հետո, էջանշանը ստանում է հաջորդ տողի սկզբի արժեքը:

* feof () ֆունկցիան վերադարձնում է true, եթե էջանիշը գտնվում է ֆայլի վերջից դուրս ("eof" նշանակում է "ֆայլի վերջ"):

* fclose () ֆունկցիան փակում է ֆայլի հետ կապը:

while () ցիկլը օրինակ 9-6-ում շարունակում է կատարվել, քանի դեռ առկա է երկու պայման՝

* feof ($fh)վերադարձնում է false:

* fgets($fh)-ից վերադարձվող $line արժեքը ընդունում է true արժեք:

Ամեն անգամ, երբ սկսվում է fgets ($fh)-ը, PHP շարժիչը կարդում է ֆայլի տողը, էջանշանը տեղափոխում է հաջորդ տողի սկիզբ եւ վերադարձնում է կարդացած տողը: Երբ էջանշանը նշում է ամենավերջին տեղը ֆայլում, feof($fh)-ը դեռեւս վերադարձնում է false: Սակայն այս պահին fgets ($fh)-ը վերադառնում է false, քանի որ նա փորձում է կարդալ տողը եւ չի կարող դա անել: Հետևաբար, ցիկլը ճիշտ ավարտելու համար անհրաժեշտ են նրա վերնագրի երկու պայմանները:

Օրինակ 6-ը օգտագործում է trim()-ը $line-ում, քանի որ fgets ()-ից վերադարձվող տողը ներառում է նոր տողի նշանը. Trim () ֆունկցիան հեռացնում է նոր տողի նշանը:

fopen()-ի առաջին արգումենտը այն ֆայլի անունն է, որը ցանկանում ենք մուտք գործել: Ֆայլի ճանապարհը նշելուց “\” սիմվոլի փոխարեն օգտագործվում է “/” սիմվոլը անգամ windows-ում: