Міністерство освіти і науки України
ВСП «Ковельський промислово-економічний фаховий коледж
Луцького національного технічного університету»

«Web-технології та Web-дизайн»

Конспект лекцій для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр IV курсу спеціальності 122 Комп'ютерні науки денної форми навчання

(Частина друга)

Ковель 2019

УДК 004.4(07)
В 26

До друку _____Голова Навчально-методичної ради Луцького НТУ
(підпис)

Електронна копія друкованого видання передана для внесення в репозитарій Луцького НТУ _____ директор бібліотеки.
(підпис)

Затверджено Навчально-методичною радою Луцького НТУ,
протокол №_____від_____20_____року.

Рекомендовано до видання методичною радою КПЕК Луцького НТУ,
протокол №_____від_____20___року.

_____ Голова методичної ради КПЕК Луцького НТУ
(підпис)

Розглянуто і схвалено на засіданні випускної методичної комісії зі спеціальності 122 Комп'ютерні науки ВСП «Ковельського промислово-економічного фахового коледжу Луцького національного технічного університету», протокол №___від «___»_____2019 року

Укладач: _____Л.В.Мелещук, викладач КПЕК Луцького НТУ
(підпис)

Рецензент: _____О.О.Герасимчук, директор ЦТДН, доцент
(підпис)

Відповідальний
за випуск: _____Л.В. Борисюк, методист КПЕК Луцького НТУ
(підпис)

Web-технології та Web-дизайн [Текст]: частина друга: конспект лекцій для студентів 4-го курсу спеціальності 122 Комп'ютерні науки денної форми навчання / уклад. Л.В.Мелещук – Ковель: КПЕК Луцького НТУ, 2019. – 95 с.

У конспекті лекцій розглянуті питання, пов'язані з графічним дизайном Web-ресурсів, основні рекомендації з HTML верстки сайтів. Розглядається розробка інтерактивних сценаріїв на мові PHP.

Конспект лекцій передбачений для засвоєння теоретичних основ курсу і представляє собою короткий зміст основних матеріалів, що викладаються на відповідних лекціях.

© Л.В.Мелещук, 2019

ЗМІСТ

ВСТУП

Конспект лекцій представляє собою короткий письмовий виклад матеріалів з дисципліни «Web-технології та Web-дизайн». Конспект лекцій використовується студентами за напрямом підготовки 122 Комп'ютерні науки спеціальності при підготовці до складання іспиту і ККР, при узагальненні та систематизації отриманої інформації, при закріпленні теоретичного матеріалу, а також при вивченні матеріалів інших дисциплін, пов'язаних з Інтернет програмуванням.

Даний курс базується на вивченні наступних тематичних напрямків:

- Основи Web - дизайну та графіка для Web.
- Теоретичні основи HTML, включаючи особливості HTML 5.0.
- Теоретичні основи CSS, включаючи особливості CSS 3.0.
- Базовий синтаксис JavaScript і бібліотеки JQuery.
- Технологія AJAX.
- Використання об'єкта XMLHttpRequest.

Метою дисципліни є формування і узагальнення спеціальних знань та навичок студентів з питань створення програм для глобальної мережі Інтернет. Програма розрахована на студентів, які засвоїли дисципліни «Основи програмування та алгоритмічні мови», «Алгоритми та структури даних», «Бази даних», «Комп'ютерні мережі».

У результаті вивчення дисципліни студент повинен знати:

- основні принципи створення Web-сторінок;
- засоби макетування та верстки засобами HTML;
- засоби макетування та верстки засобами CSS;
- правила запису HTML, CSS та PHP кодів;
- основи роботи серверних програм;
- схему роботи мереж клієнт/сервер.

У результаті вивчення дисципліни студент повинен вміти:

- створювати окремі програми для динамічних сайтів на базі HTML, стилів CSS та PHP;
- обробляти строкові дані;
- створювати програми, що можуть управляти зовнішнім виглядом сайту;
- інсталювати програми PHP Edit, Denver;
- розробляти алгоритми та програми на мові програмування PHP;
- оперувати базами даних на сервері;

Самостійне вивчення матеріалу з дисципліни, передбачене програмою, є запорукою того, що студент зможе самостійно використовувати знання для розробки і розгортання складних WEB-застосувань.

У процесі викладання дисципліни необхідно навчати зацікавленість студентів застосовувати комп'ютерні технології в обраній спеціальності, організовувати інформаційні дані різних видів, широко налагоджувати міждисциплінні зв'язки, особливо зі спецдисциплінами.

Тема. Знайомство з PHP. Основний синтаксис PHP

План

1. Короткий огляд можливостей PHP. Вбудовування PHP в HTML.
2. Основний синтаксис PHP.
3. Змінні в PHP.

1. Короткий огляд можливостей PHP. Вбудовування PHP в HTML

Тепер PHP використовується сотнями тисяч розроблювачів. Кілька мільйонів сайтів написані на PHP, що складає більше 20% доменів Internet.

«PHP може все», – заявляють його розробники. У першу чергу PHP використовується для створення скриптів, що працюють на стороні сервера, для цього його, власне, і придумали. PHP здатний розв'язувати ті ж задачі, що і будь-які інші CGI-скрипти, у тому числі обробляти дані html-форм, динамічно генерувати html сторінки і т.п. Але є й інші області, де може використовуватися PHP. Всього виділяють три основні області застосування PHP.

- Перша область – це створення додатків (скриптів), що виконуються на стороні сервера.
- Друга область – це створення скриптів, що виконуються в командному рядку. Тобто за допомогою PHP можна створювати такі скрипти, що будуть виконуватися, незалежно від web-сервера і браузера, на конкретній машині. Цей спосіб роботи підходить, наприклад, для скриптів, що повинні виконуватися регулярно за допомогою різних планувальників задач або для розв'язання задач простої обробки тексту.
- Третя область – це створення GUI-додатків (графічних інтерфейсів), що виконуються на стороні клієнта.

В принципі це не найкращий спосіб використовувати PHP, особливо для початківців, але якщо ви вже досконально вивчили PHP, то такі можливості мови можуть виявитися досить корисними. Для застосування PHP у цій області буде потрібно спеціальний інструмент – PHP-GTK, що є розширенням PHP.

У PHP поєднуються дві найпопулярніші парадигми програмування – об'єктна і процедурна. У PHP4 більш повно підтримується процедурне програмування, але існує можливість писати програми й в об'єктному стилі. Вже в перших пробних версіях PHP5 більшість недоліків у реалізації объектно-орієнтованої моделі мови, що існують у PHP4, усунуті.

Якщо говорити про можливості сьогоднішнього PHP, то вони виходять далеко за рамки тих, що були реалізовані в його перших версіях. За допомогою PHP можна створювати зображення, PDF-файли, флеш-ролики, у нього включена підтримка великої кількості сучасних баз даних, вбудовані функції для роботи з текстовими даними будь-яких форматів, включаючи XML, і функції для роботи з файловою системою. PHP підтримує взаємодію з різними сервісами за допомогою відповідних протоколів, таких як протокол

керування доступом до директорій LDAP, протокол роботи з мережним устаткуванням SNMP, протоколи передачі повідомлень IMAP, NNTP і POP3, протокол передачі гіпертексту HTTP і т.д.

Звертаючи увагу на взаємодію між різними мовами, варто згадати про підтримку об'єктів Java і можливості їхнього використання як об'єктів PHP. Для доступу до вилучених об'єктів можна використовувати розширення CORBA.

Для роботи з текстовою інформацією PHP успадкував (з невеликими змінами) механізми роботи з регулярними виразами з мови Perl і UNIX-систем. Для обробки XML-документів можна використовувати як стандарти DOM і SAX, так і API для XSLT-трансформацій.

Для створення додатків електронної комерції існує ряд корисних функцій, таких як функції здійснення платежів Cybercash, CyberMUT, VeriSign Payflow Pro і CCVS.

Зараз більш докладно розповімо про те, що являє собою PHP-програма і чим вона відрізняється від програм на мовах C, Perl і JavaScript. Розглянемо приклад

```
<html>
 <head>
   <title>Приклад</title>
 </head>
 <body>
  <?php
   echo "<p>Привіт, я – скрипт PHP!</p>";
  ?>
 </body>
</html>
```

Приклад. Простий html-файл з вбудованим кодом на PHP

Це простий HTML-файл, у який вбудований за допомогою спеціальних тегів код, написаний мовою PHP.

PHP-скрипти – це програми, що виконуються й обробляються сервером. Так що порівнювати його зі скриптовими мовами типу JavaScript неможна, тому що написані на них скрипти виконуються на машині клієнта. У чому відмінність скриптів, виконуваних на сервері і на клієнті? Якщо скрипт обробляється сервером, клієнтові посилаються тільки результати роботи скрипта. Наприклад, якщо на сервері виконувався скрипт, подібний наведеному вище, клієнт одержить згенеровану HTML-сторінку вигляду:

```
<html>
  <head>
   <title>Приклад</title>
  </head>
  <body>
   <p>Привіт, я – скрипт PHP!</p>
  </body>
</html>
```

У цьому випадку клієнт не знає, який код виконується. Можна навіть сконфігурувати свій сервер таким чином, щоб HTML-файли оброблялися

процесором PHP, так що клієнти навіть не зможуть довідатися, чи одержують вони звичайний HTML-файл або результат виконання скрипта. Якщо ж скрипт обробляється клієнтом (наприклад, це програма мовою JavaScript), то клієнт одержує сторінку, що містить код скрипта.

Як видно з наведеного прикладу PHP, скрипти вбудовуються в HTML-код. Виникає питання, яким чином? Є кілька способів. Один з них наведений у найпершому прикладі – за допомогою відкриваючого тега <?php і закриваючого тега ?>. Такого вигляду спеціальні теги дозволяють переключатися між режимами HTML і PHP. Цей синтаксис найбільш кращий, оскільки дозволяє задіяти PHP у XML-сумісних програмах (наприклад, написаних мовою XHTML), але проте можна використовувати наступні альтернативні варіанти (команда echo "Some text"; виводить на екран текст «Some text».):

- <? echo "Це найпростішаінструкція для обробки PHP"; ?>;
- <script language="php">
- echo "Деякі редактори(FrontPage) надають перевагу цьому способу"
  </script>;
- < % echo "Можна використовувати теги у стилі ASP "; %>.

Перший з цих способів не завжди доступний. Щоб ним користуватися, потрібно включити короткі теги або за допомогою функції short_tags() для PHP 3, або включивши налаштування short_open_tag у конфігураційному файлі PHP, або скомпілювавши PHP з параметром -enable-short-tags. Навіть якщо це включено за замовчуванням у php.ini-dist, використання коротких тегів не рекомендується.

Другий спосіб аналогічний налаштуванню, наприклад, JavaScript-коду і використовує для цього відповідний html тег. Тому використовувати його можна завжди, але це робиться рідко через його громіздкість.

Третій спосіб можна застосувати, тільки якщо теги в стилі ASP були включені, використовуючи конфігураційне налаштування asp_tags.

Коли PHP обробляє файл, він просто передає його текст, поки не зустріне один з перерахованих спеціальних тегів, що повідомляє йому про необхідність почати інтерпретацію тексту як коду PHP. Потім він виконує весь знайдений код до закриваючого тега, що говорить інтерпретаторові про те що далі знову йде просто текст. Цей механізм дозволяє впроваджувати PHP-код у HTML – усе за межами тегів PHP залишається незмінним, тоді як у середині інтерпретується як код.

2. Основний синтаксис PHP

Перше, що потрібно знати про синтаксис PHP, – це те, як він вбудовується в HTML-код, як інтерпретатор довідається, що це код мовою PHP. Вище вже говорилося про це. Повторюватися не будемо, відзначимо лише, що в прикладах ми найчастіше будемо використовувати варіант <?php ?>, іноді скорочений варіант <? ?>.

Програма на PHP (та й на будь-якій іншій мові програмування) – це набір команд (операторів). Оброблювачеві програми необхідно якось відрізняти одну команду від іншої. Для цього використовуються спеціальні символи – роздільники. У PHP оператори розділяються так само, як і в C Perl або Pascal, – кожен вираз закінчується крапкою з комою.

Закриваючий тег «?>» розуміється як кінець інструкції, тому перед ним крапку з комою не ставлять. Наприклад, два наступні фрагменти коду еквівалентні:

```
<?php
 echo "Hello, world!"; // крапка з комою
                      // наприкінці команди
                      // обов'язкова
?>
<?php
 echo "Hello, world!"
?>
```

Часто при написанні програм виникає необхідність робити які-небудь коментарі до коду, які ніяк не впливають на сам код, а тільки пояснюють його. Це важливо при створенні великих програм і у випадку, якщо кілька людей працюють над однією програмою. При наявності коментарів у програмі в її коді розібратися набагато простіше. В усіх мовах програмування передбачена можливість включати коментарі в код програми. PHP підтримує кілька видів коментарів: у стилі C, C++ і оболонки Unix. Символи // і # позначають початок однорядкових коментарів, /* і */ – відповідно початок і кінець багатострічкових коментарів.

```
<?php
 echo "Мене звуть Андрій";
     // Це однорядковий коментар
     // у стилі C++
 echo "Прізвище моє Вельгач";
     /* Це багатострічковий коментар.
      Тут можна написати кілька рядків.
      При виконанні програми усе, що
      знаходиться тут (закоментовано),
      буде ігноровано. */
echo"Я вивчаю PHP";
     # Це коментар у стилі
     # оболонки Unix
?>
```

Важливим елементом кожної мови є змінні, константи й оператори. Розглянемо, як виділяються й обробляються ці елементи в PHP.

3. Змінні в PHP

Змінна в PHP позначається знаком долара, за яким слідує її ім'я.

Наприклад:

```
$my_var
```

Ім'я змінної чуттєве до регістра, тобто змінні $my_var і $My_var різні.

Імена змінних формуються за таким ж правилами, що й інші імена в PHP: правильне ім'я змінної повинно починатися з букви або символу підкреслення з наступною послідовністю в будь-якій кількості буквами, цифрами або символами підкреслення.

У PHP змінні завжди присвоюються за значенням. Тобто, коли ви присвоюєте вираз змінній, усі значення оригінального виразу копіюють в цю змінну. Це означає, наприклад, що після присвоєння однієї змінної, значення іншої змінної, зміна однієї з них не впливає на значення іншої.

```
<?php
$first = ' Text '; // Присвоюємо $first значення ' Text '
$second = $first; // Присвоюємо $second значення змінної $first
$first = ' New text '; // Змінюємо значення $first на ' New text '
echo "Змінна з ім'ям first "."дорівнює $first <br>";
      // виводимо значення $first
echo "Змінна з ім'ям second ". "дорівнює $second";
      // виводимо значення $second
?>
```

Результат роботи цього скрипта буде наступним:

```
Змінна з ім'ям first дорівнює New text
Змінна з ім'ям second дорівнює Text
```

PHP 4, крім цього, пропонує ще один спосіб присвоєння значень змінним: присвоєння за посиланню. Для того, щоб присвоїти значення змінної за посиланням, це значення повинне мати ім'я, тобто воно має бути представлено якою-небудь змінною. Щоб вказати, що значення однієї змінної присвоюється іншій змінній за посиланням, потрібно перед ім'ям першої змінної вставити знак амперсанд &.

Розглянемо той же приклад, що і вище, тільки будемо присвоювати значення змінної first змінній second за посиланням:

```
<?php
$first = ' Text '; // Привласнюємо $first значення ' Text '
$second = &$first;
/* Робимо посилання на $first через $second.
  Тепер значення цих змінних будуть завжди збігатися */
// Змінимо значення $first на ' New text '
$first = ' New text ';
echo "Змінна з ім'ям first "."дорівнює $first <br>";
// виведемо значення обох змінних
echo "Змінна з ім'ям second "."дорівнює $second";
?>
```

Цей скрипт виведе наступне:

```
Змінна з ім'ям first дорівнює New text.
Змінна з ім'ям second дорівнює New text.
```

Тобто разом із змінною $first змінилася і змінна $second.

Питання для самоконтролю:

1. Для чого призначена PHP?
2. Охарактеризуйте синтаксис PHP.

3. Які Ви знаєте змінні в PHP?

Тема. Основні елементи мови PHP

План

1. Константи в PHP.
2. Оператори в PHP.
3. Типи даних в PHP.
- **a.** boolean (логічний);
- **b.** integer (цілий);
- **c.** float (з плаваючою крапкою);
- **d.** string (стрічковий);
- **e.** array (масив);

1. Константи в PHP

Для збереження сталих величин, тобто таких величин, значення яких не змінюється в ході виконання скрипта, використовуються константи. Такими величинами можуть бути математичні константи, паролі, шляхи до файлів і т.п. Основна відмінність константи від змінної полягає в тому, що їй не можна присвоювати значення більше одного разу і її значення не можна анулювати після її визначення. Крім того, у константи немає приставки у вигляді знака долара і її не можна задати простим присвоюванням значення. Як же тоді можна визначити константу? Для цього існує спеціальна функція define().

Її синтаксис такий:

```
define("Ім'я_константи","Значення_константи",[Нечутливість_до_регістра])
```

За замовчуванням імена констант чуттєві до регістра. Для кожної константи це можна змінити, вказавши як значення аргументу Нечутливість_до_регістра значення True. Існує домовленість, видно з якого імена констант завжди пишуться у верхньому регістрі.

Одержати значення константи можна, вказавши її ім'я. На відміну від змінних, не потрібно ставити перед ім'ям константи символ $. Крім того, для одержання значення константи можна використовувати функцію constant() з ім'ям константи як параметр.

```
<?php
  // визначаємо константу PASSWORD
define("PASSWORD","qwerty");
  // визначаємо регістронезалежну константу PI зі значенням 3.14
define("PI","3.14", True);
  // виводимо значення константи PASSWORD, тобто qwerty
echo (PASSWORD);
  // теж виведе qwerty
echo constant("PASSWORD");
echo (password);
  /* виведе password і попередження,
     оскільки ми ввели регістрозалежну
     константу PASSWORD */
```

```
// виводимо 3.14, оскільки константа PI регістронезалежна за визначенням
echo pi;
?>
```

Крім змінних, що задаються користувачем, про які ми тільки що розповіли, у PHP існує ряд констант, обумовлених самим інтерпретатором. Наприклад, константа __FILE__ зберігає ім'я файлу програми (і шлях до нього), що виконується в даний момент,___FUNCTION__містить ім'я функції, __CLASS__ – ім'я класу, PHP_VERSION – версія інтерпретатора PHP. Повний список визначених констант можна одержати, прочитавши посібник з PHP.

2. Оператори в PHP

Оператори дозволяють виконувати різні дії із змінними, константами і виразами. Ми ще не згадували проте, що таке вираз. Вираз можна визначити як все, що завгодно, що має значення. Змінні і константи – це основні і найбільш прості форми виразів. Існує безліч операцій (і відповідних їм операторів), з допомогою яким можна створювати вирази. Розглянемо деякі з них докладніше.

Таблиця 2.1. Арифметичні оператори

Позначення	Назва	Приклад
+	Додавання	$a + $b
-	Віднімання	$a - $b
*	Множення	$a * $b
/	Ділення	$a / $b
%	Остача від ділення	$a % $b

Таблиця 2.2. Стрічкові оператори

Позначення	Назва	Приклад
.	Конкатенація (додавання рядків)	$c=$a.$b (це рядок, що складається з $a і $b)

Таблиця 2.3. Оператори присвоювання

Позначення	Назва	Опис	Приклад
=	Присвоювання	Змінній ліворуч від оператора буде присвоєне значення, отримане в результаті виконання яких-небудь операцій або змінної/константи з правої сторони	$a = ($b = 4) +5; ($a буде дорівнює 9, $b буде дорівнює 4)
+=		Скорочення. Додає до змінної число і потім привласнює їй	$a += 5; (еквівалентно $a = $a + 5;)

		отримане значення	
.=		Скорочено позначає комбінацію операцій конкатенації і присвоєння (спочатку додається рядок, потім отриманий рядок записується в змінну)	$b = "Привіт "; $b .= "усім"; (еквівалентно $b = $b . "усім";) У результаті: $b="Привіт усім"

Таблиця 2.4. Логічні оператори

Позначення	Назва	Опис	Приклад
and	І	$a і $b істинні (True)	$a and $b
&&	І		$a && $b
or	Або	Хоча б одна з змінних $a або $b істинна (можливо, що й обидві)	$a or $b
||	Або		$a || $b
xor	виключає або	Одна з змінних істинна. Випадок, коли вони обидві істинні, виключається	$a xor $b
!	Інверсія (NOT)	Якщо $a=True, то !$a=False і навпаки	! $a

Таблиця 2.5. Оператори порівняння

Позначення	Назва	Приклад	Опис
==	Рівність	Значення змінних рівні	$a == $b
===	Еквівалентність	Рівні значення і типи змінних	$a === $b
!=	Нерівність	Значення змінних не рівні	$a != $b
<>	Нерівність		$a <> $b
!==	Нееквівалентність	Змінні не еквівалентні	$a !== $b
<	Менше		$a < $b
>	Більше		$a > $b
<=	Менше або дорівнює		$a <= $b
>=	Більше або дорівнює		$a >= $b

Таблиця 2.6. Оператори інкремента і декремента

Позначення	Назва	Опис	Приклад
++$a	Пре-інкремент	Збільшує $a на одиницю і повертає $a	<? $a=4; echo "Повинно бути 4:".$a++; echo "Повинно бути 5:".$a; ?>
$a++	Пост-	Повертає $a, потім збільшує $a на	

	інкремент	одиницю	
--$a	Пре-декремент	Зменшує $a на одиницю і повертає $a	
$a--	Пост-декремент	Повертає $a, потім зменшує $a на одиницю	

3. Типи даних

PHP підтримує вісім простих типів даних.

1. Чотири скалярних типи:
   - boolean (логічний);
   - integer (цілий);
   - float (з плаваючою крапкою);
   - string (стрічковий).
2. Два змішаних типи:
   - array (масив);
   - object (об'єкт).
3. І два спеціальних типи:
   - resource (ресурс);
   - NULL.

У PHP не прийняте явне визначення типів змінних. Переважно це робить сам інтерпретатор під час виконання програми в залежності від контексту, в якому використовується змінна. Розглянемо всі перераховані типи даних.

Тип boolean (булевий або логічний тип)

Цей найпростіший тип виражає істинність значення, тобто змінна цього типу може мати тільки два значення – істина TRUE або неправда FALSE.

Щоб визначити булевий тип, використовують ключове слово TRUE або FALSE. Обидва регістронезалежні.

```
<?php
  $test = True;
?>
```

Логічні змінні використовуються в різних керуючих конструкціях (циклах, умовах і т.п). Мати логічний тип, тобто приймати тільки два значення, істину або неправду, можуть також і деякі оператори (наприклад, оператор рівності). Вони також використовуються в керуючих конструкціях для перевірки яких-небудь умов. Наприклад, в умовній конструкції перевіряється істинність значення оператора або змінної й у залежності від результату перевірки виконуються ті або інші дії. Тут умова може бути істинною або хибною, що саме і відображають змінна й операторів логічного типу.

Тип integer (цілі)

Цей тип задає число з множини цілих чисел Z = {..., -2, -1, 0, 1, 2, ...}. Цілі числа можуть бути зазначені в десяткової, шістнадцятковій або вісімковій системах числення, за бажанням з попереднім знаком «-» або «+».

Якщо ви використовуєте вісімкову систему числення, то маєте випередити число 0 (нулем), для використання шістнадцяткової системи потрібно поставити перед числом 0x.

```
<?php
# десяткове число
$a = 1234;
# від'ємне число
$a = -123;
# вісімкове число (еквівалентне 83 у десятковій системі)
$a = 0123;
# шістнадцяткове число (еквівалентно 26 у десятковій системі)
$a = 0x1A;
?>
```

Розмір цілого залежить від платформи, хоча, як правило, максимальне значення близько двох мільярдів (це 32-бітне знакове). Беззнакові цілі PHP не підтримує.

Якщо ви визначите число, що перевищує межі цілого типу, то воно буде інтерпретоване як число з плаваючою крапкою. Також якщо ви використовуєте оператор, результатом роботи якого буде число, що перевищує межі цілого, то замість нього буде повернуте число з плаваючою крапкою.

У PHP не існує оператора ділення націло. Результатом 1/2 буде число з плаваючою крапкою, 0.5. Ви можете звести значення до цілого, що завжди округляє його в меншу сторону, або використати функцію round(), що округляє значення за стандартними правилами. Для перетворення змінної до конкретного типу потрібно перед змінною вказати в дужках потрібний тип. Наприклад, для перетворення змінної $a=0.5 до цілого типу необхідно написати (integer)(0.5) або (integer) $a або використовувати скорочений запис (int)(0.5). Можливість явного визначення типів за таким принципом існує для всіх типів даних (зазвичай, не завжди значення одного типу можна перевести в інший тип). Ми не будемо заглиблюватися в усі тонкощі визначення типів, оскільки PHP робить це автоматично в залежності від контексту.

Тип float (числа з крапкою, що плаває)

Числа з плаваючою крапкою, (вони ж числа подвійної точності або дійсні числа) можуть бути визначені з допомогою будь-якого з наступних синтаксисів:

```
<?php
  $a = 1.234;
  $b = 1.2e3;
  $c = 7E-10;
?>
```

Розмір цілого залежить від платформи, хоча максимум, як правило, ~1.8e308 з точністю близько 14 десяткових цифр.

Тип string (стрічка)

Стрічка – це набір символів. У PHP символ – це те ж саме, що байт, це означає, що існує рівно 256 різних символів. Це також означає, що PHP не має вбудованої підтримки Unicode. У PHP практично не існує обмежень на розмір рядків, тому немає абсолютно ніяких причин турбуватися про їхню довжину.

Стрічка у PHP може бути визначена трьома різними способами:

- за допомогою одинарних лапок;
- за допомогою подвійних лапок;
- heredoc-синтаксисом.

Одинарні лапки

Найпростіший спосіб визначити стрічку – це вказати її в одинарних лапках «'». Щоб використовувати одинарні лапки всередині рядка, як і в багатьох інших мовах, перед ними необхідно поставити символ зворотної косої риски «\», тобто екранувати її. Якщо зворотна коса риска повинна йти перед одинарними лапками, або бути наприкінці рядка, то необхідно продублировать її «\\'».

Якщо в середині стрічки, вказаної в одинарних лапках, зворотній слеш «\» зустрічається перед будь-яким іншим символом (відмінним від «\» і «'»), то він розглядається як звичайний символ і виводиться, як і всі інші. Тому зворотну косу риску необхідно екранувати, тільки якщо вона знаходиться наприкінці рядка, перед закриваючими лапками.

У PHP існує ряд комбінацій символів, що починаються із символу зворотної косої риси. Їх називають керуючими послідовностями, і вони мають спеціальні значення, про які ми розповімо набагато пізніше. Отож, на відміну від двох інших синтаксисів, змінні і керуючі послідовності для спеціальних символів, що зустрічаються в рядках, вказаних в одинарних лапках, не обробляються.

```
<?php
echo 'Також ви можете вставляти в стрічки
     символ нового рядка';
// Виведе: Щоб вивести ' треба перед нею поставити \
echo 'Щоб вивести \' треба перед'.'нею поставити \\';
// Виведе: Ви хочете вивести C:\*.*?
echo 'Ви хочете вивести C:\\*.*?';
// Виведе: Ви хочете вивести C:\*.*?
echo 'Ви хочете видалити C:\*.*?';
// Виведе: Це не вставить: \n
// новий рядок
echo 'Це не вставить: \n новий рядок';
// Виведе: Змінні $expand також
// $either не підставляються
echo 'Змінні $expand також $either'.'не підставляються';
?>
```

Подвійні лапки

Якщо стрічка вказана у подвійних лапках «"», то PHP розпізнає більшу кількість керуючих послідовностей як спеціальні символи. Деякі з них наведені в таблиці 2.6.

Таблиця 2.6. Керуючі послідовності

Послідовність	Значення
\n	Новий рядок (LF або 0x0A (10) у ASCII)
\r	Повернення каретки (CR або 0x0D (13) у ASCII)
\t	Горизонтальна табуляція (HT або 0x09 (9) у ASCII)
\\	Зворотна коса риска
\$	Знак долара
\"	Подвійні лапки

Повторюємо, якщо ви захочете екранувати будь-який інший символ, то зворотна коса риска також буде надрукована!

Найважливішою властивістю стрічок у подвійних лапках є обробка змінних.

Heredoc

Інший спосіб визначення стрічок – це використання heredoc-синтаксису. У цьому випадку стрічка повинна починатися із символу <<<, після якого йде ідентифікатор. Закінчується стрічка цим же ідентифікатором. Закриваючий ідентифікатор повинен починатися в першому стовпці рядка. Крім того, ідентифікатор повинен відповідати тим же правилам іменування, що і всі інші мітки в PHP: містити тільки буквено-цифрові символи та знак підкреслення і починатися не з цифри або знака підкреслення.

Heredoc-текст поводиться так само, як і стрічка у подвійних лапках, при цьому їх не маючи. Це означає, що вам немає необхідності екранувати лапки у heredoc, але ви як і раніше можете використовувати перераховані вище керуючі послідовності. Змінні всередині heredoc теж обробляються.

```
<?php
$str = <<<EOD
Приклад рядка, що охоплює кілька
рядків, з використанням
heredoc-синтаксису
EOD;
// Тут ідентифікатор – EOD. Нижче
// ідентифікатор EOT
$name = 'Вася';
echo <<<EOT
Мене кличуть "$name".
EOT;
// це виведе "Мене кличуть "Вася"."
?>
```

Зауваження: Підтримка heredoc була додана в PHP 4.

Тип array (масив)

Масив у PHP являє собою впорядковану карту – тип, що перетворює значення в ключі. Цей тип оптимізовано у декількох напрямках, тому ви можете використовувати його як масив, список (вектор), хеш-таблицю (що є реалізацією карти), стек, черга і т.д. Оскільки ви можете мати як значення інший масив PHP, можна також легко емулювати дерева.

Задати масив можна за допомогою конструкції array() або безпосередньо задаючи значення його елементам.

Визначення за допомогою array()

```
array ([key] => value,[key1] => value1, ... )
```

Мовна конструкція array() приймає як параметри пари **ключ** => значення, розділені комами. Символ => встановлює відповідність між значенням і його ключем. Ключ може бути як цілим числом, так і рядком, а значення може бути будь-якого ніснуючого в PHP типу. Числовий ключ масиву часто називають індексом. Індексування масиву в PHP починається з нуля. Значення елемента масиву можна одержати, вказавши після імені масиву в квадратних дужках ключ шуканого елемента. Якщо ключ масиву являє собою стандартний запис цілого числа, то він розглядається як число, в протилежному випадку – як рядок. Тому запис $a["1"] рівносильний запису $a[1], так само як і $a["-1"] рівносильне $a[-1].

```
<?php
$books = array ("php" =>"PHP users guide",12 => true);
echo $books["php"];
//виведе "PHP users guide"
echo $books[12];     //виведе 1
?>
```

Якщо для елемента ключ не заданий, то в якості ключа береться максимальний числовий ключ, збільшений на одиницю. Якщо вказати ключ, якому вже було присвоєно якесь значення, то воно буде перезаписано. Починаючи з PHP 4.3.0, якщо максимальний ключ – від'ємне число, то наступним ключем масиву буде нуль (0).

```
<?php
// масиви $arr і $arr1 еквіваленти
$arr = array(5 => 43, 32, 56, "b" => 12);
$arr1 = array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Якщо використовувати як ключ TRUE або FALSE, то його значення переводиться відповідно в одиницю і нуль типу integer. Якщо використовувати NULL, то замість ключа одержимо порожню стрічку. Можна використовувати і сам порожню стрічку як ключ, при цьому її треба брати в лапки. Так що це не те ж саме, що використання порожніх квадратних дужок. Не можна використовувати як ключ масиви й об'єкти.

Визначення за допомогою синтаксису квадратних дужок

Створити масив можна, просто записуючи в нього значення. Як ми вже говорили, значення елемента масиву можна одержати за допомогою квадратних дужок, всередині яких потрібно вказати його ключ наприклад, $book["php"]. Якщо вказати новий ключ і нове значення наприклад,

$book["new_key"]="new_value", то в масив додасться новий елемент. Якщо ми не вкажемо ключ, а тільки привласнимо значення $book[]="new_value", то новий елемент масиву буде мати числовий ключ, на одиницю більший максимального існуючого. Якщо масив, у який ми додаємо значення, ще не існує, то він буде створений.

```
<?
$books["key"]= value; // додали в масив
                      // $books значення
                      // value із ключем key
$books[] = value1; /* додали в масив
                      значення value1 з
                      ключем 13, оскільки
                      максимальний ключ у
                      ми був 12 */
?>
```

Для того щоб змінити конкретний елемент масиву, потрібно просто присвоїти йому з його ключем нове значення. Змінити ключ елемента не можна, можна тільки знищити елемент (пару ключ/значення) і додати нову. Щоб видалити елемент масиву, потрібно використовувати функцію unset().

```
<?php
$books = array ("php" =>"PHP users guide",12 => true);
$books[] ="Book about Perl"; // додали елемент
                  // з ключем (індексом)
                  // 13 це еквівалентно
                  // $books[13] =
                 // "Book about Perl";
$books["lisp"] =123456; /* Це додає до масиву новий
          елемент із ключем "lisp" і
          значенням 123456 */
unset($books[12]); // Це видаляє елемент
                // с ключем 12 з масиву
unset ($books); // видаляє масив цілком
?>
```

Помітимо, що, коли використовуються порожні квадратні дужки, максимальний числовий ключ шукається серед ключів, що існують у масиві з моменту останнього переіндексування. Переіндексувати масив можна за допомогою функції array_values().

```
<?php
$arr =array ("a","b","c"); /* Створюємо масив зі значеннями "a", "b" і "c".
                        Оскільки ключі не зазначені, вони будуть
                        0,1,2 відповідно */
print_r($arr); // виводимо масив (і ключі, і значення)
unset($arr[0]);
unset($arr[1]);
unset($arr[2]);
       // знищуємо з нього всі значення
print_r($arr);  // виводимо масив (і ключі, і значення)
$arr[] = "aa";  // додаємо новий елемент
             // у масив.
             // Його індексом (ключем)
```

```
                // буде 3, а не 0
print_r($arr);
$arr =array_values($arr); // переіндексуємо масив
$arr[] = "bb"; // ключем цього елемента буде 1
print_r($arr);
?>
```

Результатом роботи цього скрипта буде:

```
Array ( [0] => a [1] => b [2] => c )
Array ( )
Array ( [3] => aa )
Array ( [0] => aa [1] => bb )
```

Тип object (об'єкти)

Об'єкти – тип даних, що прийшов з об'єктно-орієнтованого програмування (ООП). Відповідно до принципів ООП, клас – це набір об'єктів, що володіють визначеними властивостями і методами роботи з ним, а об'єкт відповідно – представник класу. Наприклад, програмісти – це клас людей, що пишуть програми, вивчають комп'ютерну літературу і, крім цього, як усі люди, мають ім'я і прізвище. Тепер, якщо взяти одного конкретного програміста Васю Іванова, то можна сказати, що він є об'єктом класу програмістів, має ті ж властивості, що й інші програмісти, теж має ім'я, пише програми і т.п.

У PHP для доступу до методів об'єкта використовується оператор ->. Для ініціалізації об'єкта використовується вираз new, що створює для змінної екземпляр об'єкта.

```
<?php
  //створюємо клас людей
class Person
{
// метод, що навчає людину PHP
   function know_php()
      {
         echo "Тепер я знаю PHP";
      }
}
$bob = new Person; // створюємо об'єкт
                  // класу людини
$bob -> know_php(); // навчаємо його PHP
?>
```

Тип resource (ресурси)

Ресурс – це спеціальна змінна, що отримує посилання на зовнішній ресурс (наприклад, з'єднання з базою даних). Ресурси створюються і використовуються спеціальними функціями (наприклад, mysql_connect(), pdf_new() і т.п.).

Тип Null

Спеціальне значення NULL говорить про те, що змінна не має значення.

Змінна вважається NULL, якщо:

- їй була присвоєна константа NULL ($var = NULL);
- їй ще не було надане яке-небудь значення;
- вона була вилучена за допомогою unset().

Існує тільки одне значення типу NULL – регістронезалежне ключове слово NULL.

Отже, у цій лекції ми познайомилися з основами синтаксису мови PHP, навчилися працювати із змінними різних типів, константами, операторами, познайомилися зі всіма існуючими в PHP типами даних. Говорячи про такі типи даних, як стрічки і масиви, ми розібрали тільки самі основи.

Питання для самоконтролю:

1. Як записуються константи в PHP?
2. Які Ви знаєте оператори присвоєння в PHP?
3. Які Ви знаєте оператори порівняння в PHP?
4. Які Ви знаєте оператори арифметичні в PHP?
5. Які Ви знаєте стрічкові оператори в PHP?
6. Дайте характеристику типам даних в PHP.

Тема. Керуючі конструкції мови PHP

План

1. Умовні оператори:
   - оператор if;
   - оператор else if;
   - альтернативний синтаксис;
   - оператор switch.
2. Оператори циклу:
   - цикл з пердумовою while ;
   - цикл з післяумовою do...while;
   - цикл з лічильником for;
   - цикл для роботи з масивами foreach.

1. Умовні оператори

Оператор if

Це один з найважливіших операторів багатьох мов, включаючи PHP. Він дозволяє виконувати фрагменти коду в залежності від умови. Структуру оператора if можна представити наступним чином:

```
if (вираз) блок_виконання
```

Тут виразом є будь-який правильний PHP-вираз (тобто все, що має значення). У процесі обробки скрипта вираз набуде логічного типу. Якщо в результаті перетворення значення виразу істинне (True), то виконується блок_виконання. В протилежному випадку блок_виконання ігнорується.

Якщо блок_виконання містить кілька команд, то він повинен бути вкладений у фігурні дужки { }.

Правила перетворення виразу до логічного типу:

1. У FALSE перетворюються наступні значення:
   - логічне False;
   - цілий нуль (0);
   - дійсний нуль (0.0);
   - порожній рядок і рядок "0";
   - масив без елементів;
   - об'єкт без змінних (докладно про об'єкти буде викладено в одній з наступних лекцій);
   - спеціальний тип NULL.
2. Всі інші значення перетворяться в TRUE.

```
<?
$names = array("Іван","Петро","Семен");
if ($names[0]=="Іван") {
   echo "Привіт, Ван!";
   $num = 1;
   $account = 2000;
}
if ($num) echo "Іван перший у списку!";
$bax = 30;
if ($account > 100*$bax+3)
   echo "Цей рядок не з'явиться на екрані, тому що умова не виконана";
?>
```

Ми розглянули тільки одну, основну частину оператора if. Існує кілька розширень цього оператора. Оператор else розширює if на випадок, якщо вираз, що перевіряється в if, є невірним, і дозволяє виконати які-небудь дії за таких умов.

Структуру оператора if, розширеного за допомогою оператора else, можна представити наступним чином:

```
if (вираз) блок_виконання
else блок_виконання1
```

Цю конструкцію if...else можна інтерпретувати приблизно так: якщо виконано умову (тобто вираз=true), то виконуються дії з блоку_виконання, інакше – дії з блоку_виконання1. Використовувати оператор else не обов'язково.

Оператор elseif

Ще один спосіб розширення умовного оператора if – використання оператора elseif. elseif – це комбінація else і if. Як і else, він розширює if для виконання різних дій у тому випадку, якщо умова, що перевіряється в if, невірна. Але на відміну від else, альтернативні дії будуть виконані, тільки якщо elseif-умова є вірною. Структуру оператора if, розширеного за допомогою операторів else і elseif, можна представити наступним чином:

```
if (вираз) блок_виконання
elseif(вираз1) блок_виконання1
...
```

```
else блок_виконання
```

Операторів elseif може бути відразу кілька в одному if-блоці. Elseif-твердження буде виконано, тільки якщо попередня if-умова є False, усі попередні elseif-умови є False, а дана elseif-умова – True.

Альтернативний синтаксис

PHP пропонує альтернативний синтаксис для деяких своїх керуючих структур, а саме для if, while, for, foreach і switch. У кожному разі відкриваючи дужку потрібно замінити на двокрапку (:), а закриваючу – на endif;, endwhile; і т.д. відповідно.

Наприклад, синтаксис оператора if можна записати наступним чином:

```
if(вираз): блок_виконання endif;
```

Зміст залишається тим же: якщо умова, записана в круглих дужках оператора if, виявилась істиною, буде виконуватися весь код, від двокрапки «:» до команди endif;. Використання такого синтаксису корисне при вбудовуванні php у html-код.

```
<?php
$names = array("Іван","Петро","Семен");
if ($names[0]=="Іван"):
?>
Привіт, Ван!
<?php endif ?>
```

Оператор switch

Ще одна конструкція, що дозволяє перевіряти умову і виконувати в залежності від цього різні дії, – це switch. На українську мову назва даного оператора перекладається як «перемикач». І зміст у нього такий же. У залежності від того, яке значення має змінна, він виконує ту чи іншу дію. switch дуже схожий на оператор if...elseif...else або набір операторів if. Структуру switch можна записати наступним чином:

```
switch (вираз або змінна){
case значення1:
   блок_дій1
break;
case значення2:
   блок_дій2
break;
...
default:
   блок_дій_за_замовчуванням
}
```

На відміну від if, тут значення виразу не зводиться до логічного типу, а просто порівнюється зі значеннями, перерахованими після ключових слів case (значення1, значення 2 і т.д.). Якщо значення виразу співпадає з якимось варіантом, то виконується відповідний блок_дій – від двокрапки після значення, що співпало, до кінця switch або до першого оператора break, якщо такий знайдеться. Якщо значення виразу не співпало з жодним з варіантів, то виконуються дії за замовчуванням (блок_дій_по_замовчуванні), що знаходяться після ключового слова default. Вираз в switch обчислюється тільки один раз, а в операторі elseif – щоразу, тому, якщо вираз достатньо складний, то switch працює швидше.

```
<?
$names = array("Іван","Петро","Семен");
switch ($names[0]){
case "Іван":
   echo "Привіт, Ван!";
break;
case "Петро":
   echo "Привіт, Петя!";
break;
case "Семен":
   echo "Привіт, Сеня!";
break;
default:
   echo "Привіт, $names[0].
  А як Вас кличуть?";
}
?>
```

Якщо в цьому прикладі опустити оператор break, наприклад, у case "Петро":, то, якщо змінна виявиться рівною рядкові "Петро", після виведення на екран повідомлення "Привіт, Петя!" програма піде далі і виведе також повідомлення "Привіт, Сеня!" і тільки потім, зустрівши break, продовжить своє виконання за межами switch.

Для конструкції switch, як і для if, можливий альтернативний синтаксис, де відкриваюча switch фігурна дужка заміняється двокрапкою, а закриваюча – endswitch; відповідно.

2. **Оператори циклу**

У PHP існує кілька конструкцій, що дозволяють виконувати повторювані дії в залежності від умови. Це цикли while, do..while, foreach і for. Розглянемо їх більш докладно.

Цикл з передумовою while

Структура:

```
while (вираз) { блок_виконання }
```

або

```
while (вираз): блок_виконання end while;
```

while – простий цикл. Він пропонує PHP виконувати команди блоку_виконання доти, поки вираз обчислюється як True (тут, як і в if, відбувається переведення виразу до логічного типу). Значення виразу перевіряється щоразу на початку циклу, так якщо навіть його значення змінилося в процесі виконання блоку_виконання, цикл не буде зупинений до кінця ітерації (тобто поки всі команди блоку_виконання не будуть виконані).

```
<?
//ця програма надрукує всі парні цифри від 1 до 10
   $i = 1;
   while ($i < 10) {
      if ($i % 2 == 0) print $i;
   // друкуємо цифру, якщо вона парна
      $i++;
   // і збільшуємо $i на одиницю
   }
```

```
?>
```

Цикл з після умовою do...while

Цикли do..while дуже схожі на цикли while з тією лише різницею, що істинність виразу перевіряється наприкінці циклу, а не на початку. Завдяки цьому блок_виконання циклу do...while гарантовано виконається хоча б один раз.

Структура:

```
do {блок_виконання} while (вираз);
<?
// ця програма надрукує число 12, незважаючи на те
// що умова циклу не виконується
$i = 12;
   do{
      if ($i % 2 == 0) print $i;
   // якщо число парне, те друкуємо його
      $i++;
   // збільшуємо число на одиницю
   }while ($i<10)
?>
```

Цикл з лічильником for

Це найскладніші цикли в PHP. Вони нагадують відповідні цикли Pascal.

Структура:

```
for (вираз1; вираз2; вираз3) {блок_виконання}
```

або

```
for (вираз1; вираз2; вираз3): блок_виконання endfor;
```

Тут, як ми бачимо, умова складається відразу з трьох виразів. Перший вираз вираз 1 обчислюється, безумовно, один раз на початку циклу. На початку кожної ітерації обчислюється вираз2. Якщо він є True, то цикл продовжується і виконуються всі команди блоку_виконання. Якщо вираз2 обчислюється як False, то виконання циклу зупиняється. Наприкінці кожної ітерації (тобто після виконання всіх команд блоку_виконання) обчислюється вираз3.

Кожен з виразів 1, 2, 3 може бути порожнім. Якщо вираз2 є порожнім, то це означає, що цикл повинен виконуватися невизначену кількість разів (у цьому випадку PHP вважає цей вираз завжди вірним). Це не так непотрібно, як здається, адже цикл можна зупиняти, використовуючи оператор break.

Наприклад, усі парні цифри від 1 до 10 можна вивести з використанням циклу for наступним чином:

```
<?php
for ($i=0; $i<10; $i++){
   if ($i % 2 == 0) print $i;
   // друкуємо парні числа
}
?>
```

Якщо опустити другий вираз (умову $i<10), то таку ж задачу можна розв’язати, зупиняючи цикл оператором break.

```
<?php
for ($i=0; ;  $i++){
   if ($i>=10) break;
     // якщо $i більше або дорівнює 10,
```

```
    // то припиняємо роботу циклу
  if ($i % 2 == 0) print $i;
   // якщо число парне,
    // то друкуємо його
}
?>
```

Можна опустити всі три вирази. У цьому випадку просто не буде задано початкове значення лічильника $i і воно не буде змінюватися щоразу наприкінці циклу. Усі ці дії можна записати у вигляді окремих команд або в блоці_виконання, або перед циклом:

```
<?php
$i=0; // задаємо початкове значення лічильника
for ( ; ; ){
  if ($i>=10) break;
   // якщо $i більше або дорівнює 10,
    // то припиняємо роботу циклу
  if ($i % 2 == 0) print $i;
   // якщо число парне,
    // то друкуємо його
$i++; // збільшуємо лічильник на одиницю
}
?>
```

У третій вираз конструкції for можна записувати через кому відразу кілька найпростіших команд. Наприклад, якщо ми хочемо просто вивести всі цифри, то програму можна записати зовсім просто:

```
<?php
for ($i=0; $i<10; print $i, $i++)
/* Якщо блок_виконання не містить команд
або містить тільки одну команду,
фігурні дужки, у які він укладений,
можна опускати*/
?>
```

Цикл для роботи з масивами foreach

Ще одна корисна конструкція. Вона з'явилася тільки в PHP4 і призначена винятково для роботи з масивами.

Синтаксис:

```
foreach ($array as $value) {блок_виконання}
```

або

```
foreach ($array as $key => $value)
  {блок_виконання}
```

У першому випадку формується цикл по всіх елементах масиву, заданого змінною $array. На кожному кроці циклу значення поточного елемента масиву записується в змінну $value, і внутрішній лічильник масиву пересувається на одиницю (так що на наступному кроці буде видно наступний елемент масиву). Всередині блоку_виконання значення поточного елемента масиву може бути отримане за допомогою змінної $value. Виконання блоку_виконання відбувається стільки разів, скільки елементів у масиві $array.

Друга форма запису на додаток до перерахованого вище на кожному кроці циклу записує ключ поточного елемента масиву в змінну $key, що теж можна використовувати в блоці_виконання.

Коли foreach починає виконання, внутрішній покажчик масиву автоматично встановлюється на перший елемент.

```
<?php
$names = array("Іван","Петро","Семен");
foreach ($names as $val) {
   echo "Привіт, $val <br>";
   // виведе усім вітання
}
foreach ($names as $k => $val) {
   // крім вітання,
   // виведемо номера в списку, тобто ключі
   echo "Привіт, $val ! Ти в списку під номером $k <br>";
}
?>
```

Питання для самоконтролю:

1. Що таке альтернативний синтаксис?
2. Як записується структура оператор if?
3. Охарактеризуйте оператор циклу з пердумовою while?
4. Як записується структура оператор switch?
5. Який цикл використовується при роботі з масивами?

Тема. Керуючі конструкції мови PHP

План

1. Оператори передачі керування:
   1.1 break;
   1.2 continue;
2. Оператори включення:
   2.1 include;
   2.2 require.

1. Оператори передачі керування

Іноді у випадку особливих обставин потрібно негайно завершити роботу циклу і передати керування першої інструкції програми, розташованої за останньою інструкцією циклу. Для цього використовують оператори break і continue.

break

Оператор break закінчує виконання поточного циклу, чи то for, foreach, while, do..while або switch. break може використовуватися з числовим аргументом, що дає вказівку, роботу скількох керуючих структур, що містять його, потрібно завершити.

```
<?php
$i=1;
while ($i) {
   $n = rand(1,10);
   // генеруємо довільне число від 1 до 10
   echo "$i:$n ";
   // виводимо номер ітерації і згенероване число
   if ($n==5) break;
```

```
/* Якщо було згенеровано число 5,
то припиняємо роботу циклу. У цьому випадку
усе, що знаходиться після цього рядка
у середині циклу, не буде виконане */
   echo "Цикл працює <br>";
   $i++;
}
echo "<br>Число ітерацій циклу $i ";
?>
```

Результатом роботи цього скрипта буде приблизно наступне:

```
1:7 Цикл працює
2:2 Цикл працює
3:5
Число ітерацій циклу 3
```

continue

Іноді потрібно не цілком припинити роботу циклу, а тільки почати його нову ітерацію. Оператор continue дозволяє пропустити подальші інструкції з блоку_виконання будь-якого циклу і продовжити виконання з початку. continue можна використовувати з числовим аргументом, що вказує, скільки утримуючих його керуючих конструкцій повинні завершити роботу.

Замінимо в прикладі попереднього параграфа оператор break на continue. Крім того, обмежимо кількість кроків циклу чотирма.

```
<?php
$i=1;
while ($i<4) {
   $n = rand(1,10);
  // генеруємо довільне число від 1 до 10
   echo "$i:$n ";
  // виводимо номер ітерації і згенероване число
   if ($n==5) {
      echo "Нова ітерація ";
      continue;
/* Якщо було згенеровано число 5,
то починаємо нову ітерацію циклу,
$i не збільшується */
   }
   echo "Цикл працює <br>";
   $i++;
}
echo "<br>Число ітерацій циклу $i ";
?>
```

Результатом роботи цього скрипта буде

```
1:10 Цикл працює
2:5 Нова ітерація 2:1 Цикл працює
3:1 Цикл працює
Число ітерацій циклу 4
```

2. Оператори включення

include

Оператор include дозволяє включати код, що міститься в зазначеному файлі, і виконувати його стільки разів, скільки разів програма зустрічає цей оператор. Включення може використовуватися кожним з перерахованих способів:

```
include 'ім'я_файла';
include $file_name;
include ("ім'я_файла");
```

Приклад. Нехай у файлі params.inc у нас зберігається набір якихось параметрів і функцій. Щоразу, коли нам потрібно буде використовувати ці параметри (функції), ми будемо вставляти в текст нашої основної програми команду include 'params.inc'.

params.inc

```
<?php
$user = "Вася";
$today = date("d.m.y");
/* функція date() повертає дату
і час (тут – дату у форматі
день.місяць.рік) */
?>
```

include.php

```
<?php
include ("params.inc");
/* змінні $user і $today задані у файлові
params.inc. Тут ми теж можемо ними
користуватися завдяки команді
include("params.inc") */
echo "Привіт, $user!<br>";
   // виведе "Привіт, Вася!"
echo "Сьогодні $today";
   // виведе, наприклад, "Сьогодні 7.07.05"
?>
```

Помітимо, що використання оператора include еквівалентне простій вставці змістовної частини файла params.inc у код програми include.php. Можливо, тоді можна було б в params.inc записати простий текст без всяких тегів, що вказують на те, що це php-код? Не можна! Справа в тому, що в момент вставки файла відбувається переключення з режиму обробки PHP у режим HTML. Тому код всередині файла, що включається, який потрібно обробити як PHP-скрипт, повинен бути вкладений у відповідні теги.

Пошук файла для вставки відбувається за наступними правилами.

- Спочатку ведеться пошук файла в include_path щодо поточного робочого каталогу.
- Якщо файл не знайдений, то пошук ведеться в include_path щодо каталога поточного скрипта.
- Параметр include_path, обумовлений у файлі налаштувань PHP, задає імена каталогів, у яких потрібно шукати файли, що включаються.

Наприклад, ваш include_path це (тобто поточний каталог), поточний робочий каталог це /www/. В основний файл include.php ви включаєте файл my_dir/a.php, що у свою чергу включає b.php. Тоді парсер першою справою шукає файл b.php у каталозі /www/, і якщо такого немає, то в каталозі /www/my_dir/.

Крім локальних файлів, за допомогою include можна включати і зовнішні файли, вказуючи їхні url-адреси. Дана можливість контролюється каталогом url_fopen_wrappers у файлі налаштувань PHP і за замовчуванням, як правило, включена. Але у версіях PHP для Windows до PHP 4.3.0 ця можливість не підтримується зовсім, поза залежністю від url_fopen_wrappers.

include() – це спеціальна мовна конструкція, тому при використанні всередині умовних блоків її потрібно взяти у фігурні дужки.

```
<?php
/*  Це невірний запис. Одержимо помилку.
    Ми ж вставляємо не одну команду,
   а декілька, вони лиш записані в іншому файлі */
if ($condition) include("first.php");
else include("second.php");
// А от так правильно.
if ($condition){ include("first.php"); }
else { include("second.php"); }
?>
```

При використанні include можливі два види помилок – помилка вставки (наприклад, не можливо знайти зазначений файл, невірно написана сама команда вставки і т.п.) або помилка виконання (якщо помилка міститься у файлі, що вставляється,). У будь-якому випадку при помилці в команді include виконання скрипта не завершується.

require

Цей оператор діє приблизно так само, як і #include у C++. Усе, що ми говорили про include, лише за деякими винятком, справедливо і для require. Require також дозволяє включати в програму і виконувати який-небудь файл. Основна відмінність require і include полягає втому, як вони реагують на виникнення помилки. Як вже говорилося, include видає попередження, і робота скрипта продовжується. Помилка в require викликає фатальну помилку роботи скрипта і припиняє його виконання.

Умовні оператори на require() не впливають. Хоча, якщо рядок, у якому з'являється цей оператор, не виконується, то жоден рядок коду з файлу, що вставляється, теж не виконується. Цикли також не впливають на require(). Хоча код, який міститься у файлі, що вставляється, є об'єктом циклу, але вставка сама по собі відбувається лише один раз.

Питання для самоконтролю:

1. Для чого призначений оператор break?
2. Як працює оператор continue?
3. Яке призначення оператора include?

4. Опишіть роботу оператора require?

Тема. Створення функцій в PHP

План

1. Функції, задані користувачем.
2. Аргументи функцій.
3. Списки аргументів змінної довжини.

1. Функції, задані користувачем

Зараз ми розглянемо питання створення і використання функцій у PHP. Говорячи «функції», ми не маємо на увазі всі існуючі в PHP функції, а лише функції, задані користувачем. Ми розглянемо, завдання таких функцій, методи передачі аргументів, використання аргументів зі значенням за замовчуванням і значення, що повертаються функцією.

Для чого потрібні функції? Щоб відповісти на це запитання, потрібно зрозуміти, що взагалі являють собою функції. У програмуванні, як і в математиці, функція є відображенням безлічі її аргументів на безліч її значень. Тобто функція для кожного набору значень аргументу повертає якісь значення, що є результатом її роботи. Навіщо потрібні функції? Це спробуємо пояснити на прикладі. Класичний приклад функції в програмуванні – це функція, що обчислює значення факторіала числа. Тобто ми задаємо їй число, а вона повертає нам його факторіал. При цьому не потрібно для кожного числа, факторіал якого ми хочемо одержати, повторювати той самий код – досить просто викликати функцію з аргументом, рівним цьому числу.

Функція обчислення факторіала натурального числа:

```
<?php
function fact($n){
  if ($n==0) return 1;
  else return $fact = $n * fact($n-1);
}
?>
```

Таким чином, коли ми здійснюємо дії, в яких простежується залежність від якихось даних, і при цьому, можливо, нам знадобиться виконувати такі ж дії, але з іншими вихідними даними, зручно використовувати механізм функцій – оформити блок дій у вигляді тіла функції, а змінні дані – у якості її параметрів.

Подивимося, як у загальному виглядає задання функції. Функція може бути визначена за допомогою наступного синтаксису:

```
function Ім'я_функції (параметр1, параметр2,... параметр){
  Блок_дій
  return "значення, що повертаються функцією";}
```

Де *Ім'я_функції* й імена параметрів функції (параметр1, параметр2 і т.д.) повинні відповідати правилам найменування в PHP. Імена функцій

нечуттєві до регістра. *Параметри функції* – це змінні, тому перед назвою кожної з них повинен стояти знак $. *Блок_дій* у тілі функції - будь-який правильний PHP-код (не обов'язково залежний від параметрів). І нарешті, після ключового слова *return*повинен йти коректний php-вираз (що-небудь, що має значення). Крім того, у функції може і не бути параметрів, як і значення, що повертається. Приклад правильного оголошення функції – функція обчислення факторіала, наведена вище.

Як відбувається виклик функції? Вказується ім'я функції й у круглих дужках список значень її параметрів, якщо такі існують:

```
<?php
Ім'я_функції ("значення_для_параметра1", "значення_для_параметра2",...);
// приклад виклику функції – виклик функції
// обчислення факторіала приведений вище,
// там для обчислення факторіала числа 3
// ми писали: fact(3);
// де fact – ім'я викликуваної функції,
// а 3 – значення її параметра з ім'ям $n
?>
```

Коли можна викликати функцію? Здавалося б, дивне питання. Функцію можна викликати після її визначення, тобто в будь-якому рядку програми нижче блоку function f_name(){...}. У PHP3 це було дійсно так. Але вже в PHP4 такої вимоги немає. Уся справа в тому, як інтерпретатор обробляє одержуваний код. Єдиним винятком становлять функції, обумовлені умовно (всередині умовних операторів або інших функцій). Коли функція визначається таким чином, її визначення повинне передувати її викликові.

```
<?
$make = true;
/* тут не можна викликати Make_event();
тому що вона ще не існує, але можна
викликати Save_info() */

Save_info("Вася","Іванов", "Я вибрав курс по PHP");
if ($make){
// визначення функції Make_event()
  function Make_event(){
    echo "<p>Хочу вивчати Python<br>";
  }
}
// тепер можна викликати Make_event()
Make_event();
// визначення функції Save_info
function Save_info($first, $last, $message){
  echo "<br>$message<br>";
  echo "Ім'я: ". $first . " ". $last . "<br>";
}
Save_info("Федя","Федоров", "А я вибрав Lisp");
// Save_info можна викликати і тут
?>
```

Якщо функція один раз визначена в програмі, то перевизначити або видалити її пізніше не можна. Незважаючи на те, що імена функцій нечуттєві до регістра, краще викликати функцію по тому ж імені, яким вона була задана у визначенні.

Розглянемо докладніше аргументи функцій, їхнє призначення і використання.

2. Аргументи функцій

У кожної функції може бути, як ми вже говорили, список аргументів. За допомогою цих аргументів у функцію передається різна інформація (наприклад, значення числа, факторіал якого треба підрахувати). Кожен аргумент представляє собою змінну або константу.

За допомогою аргументів дані у функцію можна передавати трьома різними способами. Це передача аргументів за значенням (використовується за замовчуванням), за посиланням і задання значення аргументів за замовчуванням. Розглянемо ці способи докладніше.

Коли аргумент передається у функцію за значенням, зміна значення аргументу всередині функції не впливає на його значення поза функцією. Щоб дозволити функції змінювати її аргументи, їх потрібно передавати за посиланням. Для цього у визначенні функції перед ім'ям аргументу варто написати знак амперсанд «&».

```
<?php
// напишемо функцію, яка б додавала до рядка слово checked
function add_label(&$data_str){
  $data_str .= "checked";
}
$str = "<input type=radio name=article ";
  // нехай є такий рядок
echo $str .="><br>";
    // виведе елемент форми – не відзначену радіо кнопку
add_label($str);
    // викличемо функцію
echo $str .="><br>";
    // це виведе уже відзначену радіо кнопку
?>
```

У функції можна визначати значення аргументів, використовувані за замовчуванням. Саме значення за замовчуванням повинне бути сталим виразом, а не змінною і не представником класу або викликом іншої функції.

У нас є функція, що створює інформаційне повідомлення, підпис до якого змінюється в залежності від значення переданого їй параметра. Якщо значення параметра не задано, то використовується підпис "Оргкомітет".

```
<?php
function Message($sign="Оргкомітет"){
// тут параметр sign визначається за замовчуванням
// значення "Оргкомітет"
  echo "Наступні збори відбудуться завтра.";
```

```
  echo "$sign .<br>";
}
Message();
   // викликаємо функцію без параметра.
   // У цьому випадку підпис – це Оргкомітет
Message("З повагою, Вася");
   // У цьому випадку підпис
   // буде "З повагою, Вася"
?>
```

Якщо у функції кілька параметрів, то ті аргументи, для яких задаються значення за замовчуванням, повинні бути записані після всіх інших аргументів у визначенні функції. У протилежному випадку з'явиться помилка, якщо ці аргументи будуть опущені при виклику функції.

Наприклад, ми хочемо внести опис статті в каталог. Користувач повинен ввести такі характеристики статі, як її назва, автор і короткий опис. Якщо користувач не вводить ім'я автора статі, вважаємо, що це Іванов Іван.

```
<?php
function Add_article($title, $description, $author="Іванов Іван"){
  echo "Заносимо в каталог статтю: $title,";
  echo "автор $author";
  echo "<br>Короткий опис: ";
  echo "$description <hr>";
}
Add_article("Інформатика і ми","Це стаття про інформатику ...","Петров
Петро");
Add_article("Хто такі хакери", "Це стаття про хакерів ...");
?>
```

У результаті роботи скрипта одержимо наступне:

```
Заносимо в каталог статтю: Інформатика і ми, автор Петров Петро.
Короткий опис:    Це стаття про інформатику...
Заносимо в каталог статтю: Хто такі хакери, автор Іванов Іван.
Короткий опис: Це стаття про хакерів...
```

Якщо ж ми напишемо от так:

```
<?php
function Add_article($author="Іванов Іван", $title, $description){
// ...дії як у попередньому прикладі
}
Add_article("Хто такі хакеры", "Це стаття про хакеров...");
?>
```

То в результаті одержимо:

```
Warning: Missing argument 3 for
add_article() in c:\users\nina\tasks\func\def_bad.php
   on line 2
```

3. Списки аргументів змінної довжини

У PHP4 можна створювати функції із змінною кількістю аргументів. Тобто ми створюємо функцію, не знаючи заздалегідь, із скількома аргументами її викликають. Для написання такої функції ніякого спеціального синтаксису не потрібно. Усе робиться за допомогою вбудованих функцій func_num_args(), func_get_arg(), func_get_args().

Функція func_num_args() повертає кількість аргументів, переданих у поточну функцію. Ця функція може використовуватися тільки всередині визначення користувацької функції. Якщо вона з'явиться поза функцією, то інтерпретатор видасть попередження.

```
<?php
function DataCheck(){
  $n = func_num_args();
  echo "Число аргументів функції $n";
}
DataCheck();
   // виведе рядок "Число аргументів функції 0"
DataCheck(1,2,3);
   // виведе рядок "Число аргументів функції 3"
?>
```

Функція func_get_arg (ціле номер_аргументу) повертає аргумент зі списку переданих у функцію аргументів, порядковий номер якого заданий параметром номера_аргументу. Аргументи функції рахуються, починаючи з нуля. Як і func_num_args(), ця функція може використовуватися лише в середині визначення якої-небудь функції.

Номер_аргументу не може перевищувати кількості аргументів, переданих у функцію. Інакше буде згенеровано попередження, і функція func_get_arg() поверне False.

Створимо функцію для перевірки типу даних, її аргументів. Вважаємо, що перевірка пройшла успішно, якщо перший аргумент функції – ціле число, другий – стрічка.

```
<?
function DataCheck(){
  $check =true;
  $n = func_num_args();
     // число аргументів, переданих у функцію

  /* перевіряємо, чи є перший
  переданий аргумент цілим числом */
  if ($n>=1) if (!is_int(func_get_arg(0)))
      $check = false;

  /* перевіряємо, чи є другий
  переданий аргумент стрічкою */

  if ($n>=2)
     if (!is_string(func_get_arg(1)))
        $check = false;
return $check;
```

```
}
if (DataCheck(a123,"text"))
  echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовільняють умов<br>";
if (DataCheck(324))
  echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовільняють умов<br>";
?>
```

Результатом роботи буде наступне.

```
Дані не задовільняють умов
Перевірка пройшла успішно
```

Функція func_get_args() повертає масив, що складається зі списку аргументів, переданих функції. Кожен елемент масиву відповідає аргументові, переданому функції. Якщо функція використовується поза визначенням користувацької функції, то генерується попередження.

Перепишемо попередній приклад, використовуючи цю функцію. Будемо перевіряти, чи є цілим числом кожен парний аргумент, переданий функції:

```
<?
function DataCheck(){
 $check =true;
 $n = func_num_args();
 // число аргументів, переданих у функцію
 $args = func_get_args();
 // масив аргументів функції
 for ($i=0;$i<$n;$i++){
  $v =  $args[$i];
  if ($i % 2 == 0){
   if (!is_int($v)) $check = false;
   // перевіряємо, чи є парний аргумент цілим
  }
 }
return $check;
}
if (DataCheck("text", 324))
  echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовільняють умов<br>";
?>
```

Як бачимо, комбінації функцій func_num_args(), func_get_arg() і func_get_args() використовуються для того, щоб функції могли мати змінний список аргументів. Ці функції були додані тільки в PHP 4. У PHP3 для того, щоб домогтися подібного ефекту, можна використовувати як аргумент функції масив. Наприклад, от так можна написати скрипт, що перевіряє, чи є кожен непарний параметр функції цілим числом:

```
<?
function DataCheck($params){
   $check =true;
```

```
    $n = count($params);
  // число аргументів,переданих у функцію
  for ($i=0;$i<$n;$i++){
    $v = $params[$i];
    if ($i % 2 !== 0){
      // перевіряємо, чи є непарний аргумент цілим
if (!is_int($v)) $check = false;
    }
  }
return $check;
}
if (DataCheck("text", 324))
    echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовільняють умов<br>";
```

Питання для самоконтролю:

1. Для чого потрібні функції?
2. Як записується загальний вигляд функції?
3. Що таке аргумент функції?
4. Які Ви знаєте способи передачі функції за допомогою аргументів?

Тема. Внутрішні (вбудовані) функції

План

1. Використання змінних всередині функції:
   - глобальні змінні;
   - статичні змінні;
   - значення, що повертаються.
2. Внутрішні (вбудовані) функції.

1. Використання змінних всередині функції

Глобальні змінні

Щоб використовувати всередині функції змінні, задані поза нею, ці змінні потрібно оголосити як глобальні. Для цього в тілі функції варто перелічити їхні імена після ключового слова global: global $var1, $var2;

```
<?
$a=1;
function Test_g(){
global $a;
  $a = $a*2;
  echo 'у результаті роботи функції $a=',$a;
}
echo 'поза функцією $a=',$a,', ';
Test_g();
echo "<br>";
echo 'поза функцією $a=',$a,', ';
Test_g();
?>
```

У результаті роботи цього скрипта одержимо:

```
поза функцією $a=1, у результаті роботи функції $a=2
поза функцією $a=2, у результаті роботи функції $a=4
```

Статичні змінні

Щоб використовувати змінні тільки всередині функції, при цьому зберігаючи їхні значення і після виходу з функції, потрібно оголосити ці змінні як статичні. Статичні змінні видно тільки всередині функції вони не втрачають свого значення, якщо виконання програми виходить за межі функції. Оголошення таких змінних здійснюється за допомогою ключового слова static: static $var1, $var2;

Статичній змінній може бути присвоєне будь-яке значення, але не посилання.

```
<?
function Test_s(){
static $a = 1;
// не можна присвоїти вираз або посилання
  $a = $a*2;
  echo $a;
}
Test_s(); // виведе 2
echo $a;  // нічого не виведе, тому що $a доступна тільки
        // усередині функції
Test_s(); // всередині функції $a=2, тому результатом роботи функції
        // буде число 4
?>
```

Значення, що повертаються

Усі функції, наведені вище як приклади, виконували які-небудь дії. Крім подібних дій, будь-яка функція може повертати як результат своєї роботи яке-небудь значення. Це робиться за допомогою твердження return. Значення, що повертається, може бути будь-якого типу, включаючи списки й об'єкти. Коли інтерпретатор зустрічає команду return у тілі функції, він негайно припиняє її виконання і переходить на той рядок, з якого була викликана функція.

Наприклад, складемо функцію, що повертає вік людини. Якщо людина не вмерла, то вік вважається щодо поточного року.

```
<?php
/* якщо другий параметр обчислюється
як true, то він розглядається як
дата смерті, */
function Age($birth, $is_dead){
  if ($is_dead) return $is_dead-$birth;
  else return date("Y")-$birth;
}
echo Age(1971, false); // виведе 33
echo Age(1971, 2001); // виведе 30?>
```

У цьому прикладі можна було і не використовувати функцію return, а просто замінити її функцією виведення echo. Однак якщо ми все-таки робимо

так, що функція повертає якесь значення (у даному випадку вік людини), то в програмі ми можемо присвоїти будь-якій змінній значення цієї функції:

```
$my_age = Age(1981, 2004).
```

У результаті роботи функції може бути повернуте тільки одне значення. Кілька значень можна одержати, якщо повертати список значень (одномірний масив). Припустимо, ми хочемо одержати повний вік людини з точністю до дня.

```
<?php
function Full_age($b_day, $b_month, $b_year){
  if (date("m")>$b_month && date("d")>$b_day)
  {
    $day = date ("d") - $b_day;
    $month = date("m") - $b_month;
    $year = date("Y") - $b_year;
  } else {
    $year = date("Y") - $b_year - 1;
    $day = 31 - ($b_day - date ("d"));
    $month = 12 - ($b_month - date("m"));
  }
return array ($day,$month,$year);
}
$age = Full_age("07","08","1974");
echo "Вам $age[2] років, $age[1] місяців і $age[0] днів";
// виведе "Вам 29 років, 11 місяців і 5 днів"
?>
```

Коли функція повертає кілька значень для їхньої обробки в програмі, то зручно використовувати мовну конструкцію list(), що дозволяє однією дією присвоїти значення відразу декільком змінним. Так, у попередньому прикладі, залишивши без зміни функцію, обробити значення, що повертаються з неї, можна було так:

```
<?
// завдання функції Full_age()
list($day,$month,$year) = Full_age("07","08","1974");
echo "Вам $year років, $month місяців і $day днів";
?>
```

Взагалі конструкцію list() можна використовувати для присвоєння змінним значень елементів будь-якого масиву.

```
<?
$arr = array("first","second");
list($a,$b) = $arr;
  // змінної $a привласнюється перше
  // значення масиву, $b – друге
echo $a," ",$b;
  // виведе рядок «first second»?>
```

2. Внутрішні (вбудовані) функції

Кажучи про функції, створювані користувачем, все-таки не можна не сказати пару слів про вбудовані функції. З деякими з вбудованих функцій, такими як echo(), print(), date(), include(), ми вже познайомилися. Насправді всі перераховані функції, крім date(), є мовними конструкціями. Вони входять у ядро PHP і не вимагають ніяких додаткових налаштувань і модулів. Функція date() теж входить до складу ядра PHP і не вимагає налаштувань. Але є і функції, для роботи з якими потрібно встановити різні бібліотеки і підключити відповідний модуль. Наприклад, для використання функцій роботи з базою даних MySql варто скомпілювати PHP з підтримкою цього розширення. Останнім часом до складу PHP так, щоб з ними можна працювати без будь-яких додаткових налаштувань інтерпретатора.

Питання для самоконтролю:

1. Що таке глобальні змінні?
2. Що таке статистичні змінні?
3. Як записується функція, значення якої повертається?
4. Які функції називаються внутрішніми?

Тема. Обробка запитів з допомогою PHP. Використання HTML форм для передачі даних на сервер

План

1. Протокол HTTP і способи передачі даних на сервер.
2. Форма запиту клієнта.
3. Методи передачі даних.
   - GET;
   - HEAD;
   - POST;
4. Використання HTML-форм для передачі даних на сервер.
5. Обробка запитів за допомогою PHP.

Internet побудований за багаторівневим принципом, від фізичного рівня, пов'язаного з фізичними аспектами передачі двійкової інформації, і до прикладного рівня, що забезпечує інтерфейс між користувачем і мережею.

1. Протокол HTTP і способи передачі даних на сервер

HTTP (HyperText Transfer Protocol, протокол передачі гіпертексту) – це протокол прикладного рівня, розроблений для обміну гіпертекстовою інформацією в Internet.

HTTP надає набір методів для вказання цілей запиту, що відправляється серверові. Ці методи засновані на правилах посилань, де для вказівки ресурсу, до якого повинен бути застосований даний метод, використовується універсальний ідентифікатор ресурсів (Universal Resource Identifier) у вигляді місцезнаходження ресурсу (Universal Resource Locator,

URL) або у вигляді його універсального імені (Universal Resource Name, URN).

Протокол реалізує принцип запит/відповідь. Запитуюча програма – клієнт ініціює взаємодію з програмою, що відповідає сервером і надсилає запит, що містить:

- метод доступу;
- адресу URL;
- версію протоколу;
- повідомлення (схоже за формою на MIME) з інформацією про тип переданих даних, інформацією про клієнта, що послав запит, і, можливо, зі змістовною частиною (тілом) повідомлення.

Відповідь сервера містить:

- стрічку стану, в яку входять версія протоколу і код повернення (успіх або помилка);
- повідомлення (у формі, схожій на MIME), в яке входить інформація сервера, метаінформація (тобто інформація про зміст повідомлення) і тіло повідомлення.

У протоколі не вказується, хто повинен відкривати і закривати з'єднання між клієнтом і сервером. На практиці з'єднання, як правило, відкриває клієнт, а сервер після відправлення відповіді ініціює його розрив.

Давайте розглянемо більш детально, у якій формі відправляються запити на сервер.

2. Форма запиту клієнта

Клієнт відсилає серверові запит в одній із двох форм: у повній або скороченій. Запит у першій формі називається відповідно повним запитом, а в другій формі – простим запитом.

Простий запит містить метод доступу й адресу ресурсу. Формально це можна записати так:

<Простий-Запит>:=<Метод><символ пробіл><Запитуваний-URL><символ нового рядка>

Як метод можуть бути зазначені GET, POST, HEAD, PUT, DELETE та інші. Про найбільш поширені з них ми поговоримо дещо пізніше. В якості запитуваного URL найчастіше використовується URL-адреса ресурсу.

Приклад простого запиту:

```
GET http://phpbook.info/
```

Тут GET – це метод доступу, тобто метод, що повинен бути застосований до запитуваного ресурсу, а http://phpbook.info/ – це URL-адреса запитуваного ресурсу.

Повний запит містить рядок стану, кілька заголовків (заголовок запиту, загальний заголовок або заголовок змісту) і, можливо, тіло запиту. Формально загальний вид повного запиту можна записати так:

<Повний запит> := <Рядок Стану>
(<Загальний заголовок>|<Заголовок запиту>|<Заголовок змісту>)

<символ нового рядка>
[<зміст запиту>]

Квадратні дужки тут позначають необов'язкові елементи заголовка, через вертикальну риску перераховані альтернативні варіанти. Елемент <Рядок стану> містить метод запиту і URL ресурсу (як і простий запит) і, крім того, використовувану версію протоколу HTTP.

Наприклад, для виклику зовнішньої програми можна задіяти наступний рядок стану:

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

У даному випадку використовують метод POST і протокол HTTP версії 1.0.

В обох формах запиту важливе місце займає URL запитуваного ресурсу. Найчастіше URL використовується у вигляді URL-адреси ресурсу. При звертанні до сервера можна застосовувати як повну форму URL, так і спрощену.

Повна форма містить тип протоколу доступу, адресу сервера ресурсу й адресу ресурсу на сервері (рис 1.).

У скороченій формі опускають протокол і адресу сервера, вказуючи лише місце розташування ресурсу від кореня сервера. Повну форму використовують, якщо можливе пересилання запиту іншому серверові. Якщо ж робота відбувається тільки з одним сервером, то частіше застосовують скорочену форму.

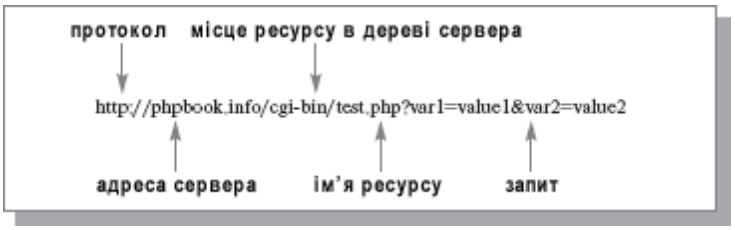

Рис.1. Повна форма URL

Далі ми розглянемо найбільш розповсюджені методи відправлення запитів.

3. Методи передачі даних

Як вже говорилося, будь-який запит клієнта до сервера повинен починатися з вказівки методу. Метод повідомляє про мету запиту клієнта. Протокол HTTP підтримує досить багато методів, але реально використовуються тільки три:

- GET
- HEAD
- POST

GET

Метод GET дозволяє одержати будь-які дані, ідентифіковані за допомогою URL у запиті ресурсу. Якщо URL вказує на програму, то повертається результат роботи програми, а не її текст (якщо, звичайно, текст

не є результатом її роботи). Додаткова інформація, яка необхідна для обробки запиту, вбудовується в сам запит (у рядок статусу). При використанні методу GET у поле тіла ресурсу повертається власне викликана інформація (текст HTML-документа, наприклад).

Існує різновид методу GET – умовний GET. Цей метод повідомляє серверові про те, що на запит потрібно відповісти тільки якщо виконано умову, що міститься в полі if-Modified-Since заголовка запиту. Якщо говорити більш точно, то тіло ресурсу передається у відповідь на запит, якщо цей ресурс змінювався після дати, зазначеної в if-Modified-Since.

HEAD

Метод HEAD аналогічний методові GET, тільки він не повертає тіло ресурсу і не має умовного аналога. Метод HEAD використовують для одержання інформації про ресурс. Це може знадобитися, наприклад, при розв'язанні задачі тестування гіпертекстових посилань.

POST

Метод POST розроблений для передачі на сервер такої інформації, як анотації ресурсів, новинні і поштові повідомлення, дані для додавання в базу даних, тобто для передачі інформації великого обсягу і досить важливої. На відміну від методів GET і HEAD, у POST передається тіло ресурсу, що і є інформацією, одержуваної з полів форм або інших джерел введення.

4. Використання HTML-форм для передачі даних на сервер

Як передавати дані серверові? Для цього в мові HTML є спеціальна конструкція – форми. Форми призначені для того, щоб отримувати від користувача інформацію. Наприклад, вам потрібно знати логін і пароль користувача для того, щоб визначити, на які сторінки сайту його можна допускати. Або вам необхідні особисті дані користувача, щоб була можливість з ним зв'язатися. Форми саме і застосовуються для введення такої інформації. У них можна вводити текст або вибирати потрібні варіанти зі списку. Дані, записані у форму, відправляються для обробки спеціальній програмі (наприклад, скрипту на PHP) на сервері. У залежності від введених користувачем даних ця програма може формувати різні web-сторінки, відправляти запити до бази даних, запускати різні додатки і т.п. Із синтаксисом HTML-форм ми розбиралися поверхово в одній із попередніх лекцій, але зараз ми все-таки повторимо основні моменти, оскільки це важливо.

Отже, для створення форми в мові HTML використовується тег FORM. У середині нього знаходиться одна або кілька команд INPUT. За допомогою атрибутів action і method тега FORM задаються ім'я програми, що буде обробляти дані форми, і метод запиту, відповідно. Команда INPUT визначає тип і різні характеристики запитуваної інформації. Відправлення даних форми відбувається після натискання кнопки input типу submit.

Отже, у формі можна вказувати метод передачі даних. Подивимося, що буде відбуватися, якщо вказати метод GET або POST, і в чому буде різниця.

Для методу GET

При відправленні даних форми за допомогою методу GET вміст форми додається до URL після знака питання у вигляді пар *ім'я=значення*, об'єднаних за допомогою амперсанта &:

action?name1=value1&name2=value2&name3=value3

Тут action – це URL-адреса програми, що повинна обробляти форму (це або програма, задана в атрибуті action тега form, або сама поточна програма, якщо цей атрибут опущений). Імена name1, name2, name3 відповідають іменам елементів форми, а value1, value2, value3 – значенням цих елементів. Усі спеціальні символи, включаючи = і &, в іменах або значеннях цих параметрів будуть опущені. Тому не варто використовувати в назвах або значеннях елементів форми ці символи і символи кирилиці в ідентифікаторах.

Якщо в поле для введення ввести який-небудь службовий символ, то він буде переданий у його шіснадцятковому коді, наприклад, символ $ заміниться на %24. Так само передаються і кириличні літери.

Для полів введення тексту і пароля (це елементи input з атрибутом type=text і type=password), значенням буде те, що введе користувач. Якщо користувач нічого не вводить у таке поле, то в рядку запиту буде присутній елемент name=, де name відповідає імені цього елемента форми.

Для кнопок типу checkbox і radio button значення value визначається атрибутом VALUE у тому випадку, коли кнопка відзначена. Не відзначені кнопки при складанні рядка запиту ігноруються цілком. Кілька кнопок типу checkbox можуть мати один атрибут NAME (і різні VALUE), якщо це необхідно. Кнопки типу radio button призначені для одного з усіх запропонованих варіантів і тому повинні мати однаковий атрибут NAME і різні атрибути VALUE.

У принципі створювати HTML-форму для передачі даних методом GET не обов'язково. Можна просто додати в рядок URL потрібні змінні і їхні значення.

http://phpbook.info/test.php?id=10&user=pit

У зв'язку з цим в передачі даних методом GET є один істотний недолік – кожен може підробити значення параметрів. Тому не радимо використовувати цей метод для доступу до захищених паролем сторінок, для передачі інформації, що впливає на безпеку роботи програми або сервера. Крім того, не варто застосовувати метод GET для передачі інформації, що не дозволено змінювати користувачеві.

Незважаючи на всі ці недоліки, використовувати метод GET досить зручно при налаштуванні скриптів (тоді можна бачити значення й імена переданих змінних) і для передачі параметрів, що не впливають на безпеку.

Для методу POST

Вміст форми кодується точно так само, як для методу GET (див. вище), але замість додавання рядка до URL вміст запиту посилається блоком даних як частина операції POST. Якщо є присутній атрибут ACTION, то значення URL, що там знаходиться, визначає, куди посилати цей блок даних. Цей метод, як вже відзначалося, рекомендується для передачі великих за обсягом блоків даних.

Інформація, введена користувачем і відправлена серверові за допомогою методу POST, подається на стандартне введення програмі, зазначеної в атрибуті action, або поточному скрипту, якщо цей атрибут опущений. Довжина файлу, що посилається, передається в змінній CONTENT_LENGTH, а тип даних - у змінній CONTENT_TYPE.

Передати дані методом POST можна тільки за допомогою HTML-форми, оскільки дані передаються в тілі запиту, а не в заголовку, як у GET. Відповідно і змінити значення параметрів можна, тільки змінивши значення, введене у форму. При використанні POST користувач не бачить переданих серверові даних.

Основна перевага POST-запитів – це їхня велика безпека і функціональність у порівнянні з GET-запитами. Тому метод POST частіше використовують для передачі важливої інформації, а також інформації великого обсягу. Проте не варто цілком покладатися на безпеку цього механізму, оскільки дані POST-запиту також можна підробити, наприклад створивши html-файл на своїй машині і заповнивши його потрібними даними. Крім того, не всі клієнти можуть застосовувати метод POST, що обмежує варіанти його використання.

При відправленні даних на сервер будь-яким методом передаються не тільки самі дані, введені користувачем, але і ряд змінних, названих змінними оточення, що характеризують клієнта, історію його роботи, шляхи до файлів і т.п. Нижче наведені деякі змінні оточення:

- REMOTE_ADDR – IP-адреса хоста (комп'ютера), що відправляє запит;
- REMOTE_HOST – ім'я хоста, з якого відправлений запит;
- HTTP_REFERER – адреса сторінки, що посилається на поточний скрипт;
- REQUEST_METHOD – метод, що був використаний при відправленні запиту;
- QUERY_STRING – інформація, що знаходиться в URL після знака питання;
- SCRIPT_NAME – віртуальний шлях до програми, що повинна виконуватися;
- HTTP_USER_AGENT – інформація про браузер, що використовує клієнт;

5. Обробка запитів за допомогою PHP

Дотепер ми говорили лише про те, що запити клієнта обробляються на сервері за допомогою спеціальної програми. Насправді цю програму ми

можемо написати самі, в тому числі і мовою PHP, і вона буде робити з отриманими даними все, що ми захочемо. Для того щоб написати цю програму, необхідно познайомитися з деякими правилами й інструментами, запропонованими для цих цілей PHP.

У середині PHP-скрипта існує кілька способів одержання доступу до даних, переданих клієнтом за протоколом HTTP. До версії PHP 4.1.0 доступ до таких даних здійснювався за іменами переданих змінних (нагадаємо, що дані передаються у вигляді пар «ім'я змінної, символ «=», значення змінної»). Таким чином, якщо, наприклад, було передано first_name=Nina, то в середині скрипта з'являлася змінна $first_name зі значенням Nina. Якщо було потрібно розрізняти, яким методом були передані дані, то використовувалися асоціативні масиви $HTTP_POST_VARS і $HTTP_GET_VARS, ключами яких були імена переданих змінних, а значеннями – відповідно значення цих змінних. Таким чином, якщо пара first_name=Nina передана методом GET, то $HTTP_GET_VARS["first_name"]="Nina".

Використовувати в програмі імена переданих змінних прямо (безпосередньо) небезпечно. Тому було вирішено, починаючи з PHP 4.1.0, задіяти для звертання до змінних, переданих за допомогою HTTP-запитів, спеціальний масив – $_REQUEST. Цей масив містить дані, передані методами POST і GET, а також за допомогою HTTP cookies. Це суперглобальний асоціативний масив, тобто його значення можна одержати в будь-якому місці програми, використовуючи як ключ ім'я відповідної змінної (елемента форми).

Приклад. Припустимо, ми створили форму для реєстрації учасників заочної школи програмування. Тоді у файлі 1.php, що обробляє цю форму, можна написати наступне:

```
<?php
$str = "Привіт,".$_REQUEST["first_name"]."  ".$_REQUEST["last_name"]."!
<br>";
$str .="Ви вибрали для вивчення курс по".$_REQUEST["kurs"];
echo $str;
?>
```

Тоді, якщо у форму ми ввели ім'я «Вася», прізвище «Петров» і вибрали серед всіх курсів курс по PHP, на екрані браузера одержимо таке повідомлення:

Привіт, Вася Петров!
Ви вибрали для вивчення курс по PHP

Після введення масиву $_REQUEST масиви $HTTP_POST_VARS і $HTTP_GET_VARS для одностайності були перейменовані в $_POST і $_GET відповідно, але самі вони з використання не зникли з розумінь сумісності з попередніми версіями PHP. На відміну від своїх попередників, масиви $_POST і $_GET стали суперглобальними, тобто доступними прямо і в середині функцій й методів.

Наведемо приклад використання цих масивів. Припустимо, нам потрібно обробити форму, що містить елементи введення з іменами first_name, last_name, kurs. Дані передані методом POST, і дані, передані іншими методами, ми обробляти не хочемо. Це можна зробити наступним чином:

```
<?php
$str = "Привіт, ".$_POST ["first_name"]." ".$_POST ["last_name"] ."! <br>";
$str .= "Ви вибрали для вивчення курс по ".$_POST["kurs"];
echo $str;
?>
```

Тоді на екрані браузера, якщо ми ввели ім'я «Вася», прізвище «Петров» і вибрали серед усіх курсів курс з PHP, побачимо повідомлення, як у попередньому прикладі:

```
Привіт, Вася Петров!
Ви вибрали для вивчення курс з PHP
```

Для того, щоб зберегти можливість обробки скриптів більш ранніх версій, ніж PHP 4.1.0, була введена директива register_globals, що дозволяє або забороняє доступ до змінних безпосередньо за їх іменами. Якщо у файлі налаштування PHP параметр register_globals=On, то до змінної, переданої серверу методами GET і POST, можна звертатися просто за їх іменами (тобто можна писати $first_name). Якщо ж register_globals=Off, то потрібно писати $_REQUEST["first_name"] або $_POST["first_name"], $_GET["first_name"], $HTTP_POST_VARS["first_name"], $HTTP_GET_VARS["first_name"]. З погляду безпеки цю директиву краще відключати (тобто register_globals=Off). При включеній директиві register_globals перераховані вище масиви також будуть містити дані, передані клієнтом.

Іноді виникає необхідність довідатися значення якої-небудь змінної оточення, наприклад метод, що використовувався при передачі запиту або IP-адреса комп'ютера, що відправив запит. Одержати таку інформацію можна за допомогою функції getenv(). Вона повертає значення змінної оточення, ім'я якої передане їй як параметр.

```
<?
getenv("REQUEST_METHOD");
   // поверне використаний метод
echo getenv ("REMOTE_ADDR");
   // виведе IP-адресу користувача, що послав запит
?>
```

Як ми вже говорили, якщо використовується метод GET, то дані передаються дописуванням рядка запиту у вигляді пар «ім'я_змінної=значення до URL-адреси ресурсу». Усе, що записано в URL після знакау запитання, можна одержати за допомогою команди getenv("QUERY_STRING").

Завдяки цьому можна за методом GET передавати дані в якому-небудь іншому вигляді. Наприклад, вказувати лише значення декількох параметрів через знак плюс, а в скрипті розбирати рядок запиту на частини або можна передавати значення лише одного параметра. У цьому випадку в масиві $_GET з'явиться порожній елемент із ключем, рівним цьому значенню (всьому рядкові запиту), причому символ «+», що зустрівся в рядку запиту, буде замінений на підкреслення «_».

Методом POST дані передаються лише за допомогою форм, і користувач (клієнт) не бачить, які саме дані відправляються серверові. Щоб їх побачити, хакер повинен підмінити нашу форму своєю. Тоді сервер

відправить результати обробки неправильної форми не туди, куди потрібно. Щоб цього уникнути, можна перевіряти адресу сторінки, з якої були послані дані. Це можна зробити за допомогою функції getenv(): getenv("HTTP_REFERER").

Питання для самоконтролю:

1. Які Ви знаєте способи передавачі даних на сервер?
2. Охарактеризуйте метод GET.
3. Охарактеризуйте метод HEAD.
4. Охарактеризуйте метод POST.
5. Що таке форма і для чого вона призначена?

Тема. Функції роботи із стрічками в PHP

План

1. Стрічки. Пошук елемента в стрічці.
2. Вирізання підстрічки:
   - Функція strstr;
   - Функція substr;
   - Функція strip_tags.
3. Заміна входження підстрічки:
   - Функція str_replace;
   - Функція substr_replace.
4. Поділ і з'єднання стрічки.

1. Стрічки. Пошук елемента в стрічці

В одній з попередніх лекцій ми наводили три способи завдання *стрічок*: за допомогою одинарних лапок, подвійних лапок і за допомогою heredoc–синтаксису. Відзначали ми й основні відмінності між цими способами. В основному вони стосуються обробки змінних і керуючих послідовностей в середині стрічки.

```
<?php
echo 'У такій стрічці НЕ обробляються змінні і більшість послідовностей';
echo "Тут змінні і послідовності обробляються";
echo <<<EOT
Тут теж обробляються як змінні, так і керуючі послідовності.
І крім того, можна вводити символи лапок
без їхнього екранування зворотним слешем.
EOT;
?>
```

Не одноразово, починаючи з найпершої лекції з PHP, ми використовували функцію echo. Насправді, echo – не функція, а мовна конструкція, тому використовувати при її виклику круглі дужки не обов'язково. Echo дозволяє виводити на екран стрічки, передані їй як параметри. Параметрів у echo може бути скільки завгодно. Їх розділяють

комами або поєднують за допомогою оператора конкатенації і ніколи не беруть у круглі дужки.

```
<?
echo "Прийшов ", "побачив ", "переміг ";
// виведе Стрічка "Прийшов побачив переміг"
// багато хто воліє передавати більше
// параметрів у echo за допомогою конкатенації
echo "Прийшов " . "побачив " . "переміг ";
// теж виведе Стрічку
// "Прийшов побачив переміг"
echo ("Прийшов ", "побачив ", "переміг ");
// видасть помилку: unexpected ','
?>
```

Існує скорочений синтаксис для команди echo:

```
<?=Стрічка_для_виведення ?>
```

Тут параметр Стрічка_для_виведеня містить стрічку, задану будь-яким з відомих способів, що повинна бути виведена на екран.

Наприклад, такий скрипт виведе на екран червоним кольором "Мене кличуть Вася":

```
<? $name="Вася" ?>
<font color=red>Мене кличуть <?=$name?></font>
```

Крім мовної конструкції echo існує ряд функцій для виведення стрічок. Це в першу чергу функція print і її різновиди printf, sprintf і т.п.

Функція print дозволяє виводити на екран тільки одну стрічку і, як і echo, не може бути викликана за допомогою змінних функцій, оскільки є мовною конструкцією.

Функція print_r не належить до стрічкових функцій, як можна було б подумати. Вона відображає інформацію про змінну у формі, зрозумілій користувачеві.

Функції sprintf і printf обробляють передану їм стрічку відповідно до заданого формату. Але про них ми говорити не будемо. А поговоримо про те, як можна здійснювати пошук у тексті, представленому у вигляді стрічки.

Для того щоб визначити, чи входить дана підстрічка до складу стрічки, використовується функція strpos(). Синтаксис strpos() такий:

strpos (задана стрічка,стрічка для пошуку [,з якого символу шукати])

Вона повертає позицію входження шуканої стрічки у задану стрічку або повертає логічне false, якщо входження не знайдене. Додатковий аргумент дозволяє задавати символ, починаючи з якого буде виконуватися пошук. Крім логічного false ця функція може повертати й інші значення, що зводяться до false (наприклад, 0 або ""). Тому для того, щоб перевірити, чи знайдена шукана стрічка, рекомендують використовувати оператор еквівалентності «===».

```
<? $str = "Ідея наносити дані на перфокарти і потім зчитувати й обробляти їх
автоматично належала Джону Биллингсу,а її технічне рішення здійснив
Герман Холлерит.
```

```
Перфокарта Холлерита виявилася настільки вдалою, що без найменших змін проіснувала
до наших днів.";
$pos = strpos($str,"Холлерит");
if ($pos !== false) echo "Шукана стрічка зустрінута в позиції номер $pos ";
else echo "Шукана стрічка не знайдена";
/* помітимо, що ми перевіряємо значення $pos на еквівалентність з false.
Інакше Стрічка, що знаходиться в першій позиції,не була б знайдена, тому що інтерпретується як false. */
?>
```

Якщо значення параметра стрічки_для_пошуку не є стрічкою, то воно перетворюється в цілий тип і розглядається як ASCII-код символу. Щоб одержати ASCII-код будь-якого символу в PHP, можна використати функцію ord("символ")

Наприклад, якщо ми напишемо $pos = strpos($str,228); то інтерпретатор буде вважати, що ми шукаємо символ «д». Якщо додати цей рядок у наведений вище приклад і вивести результат, то одержимо повідомлення, що шукана стрічка знайдена у позиції 1.

Функція, зворотна за змістом ord, – це chr (код символу). Вона по ASCII-коду виводить символ, що відповідає цьому кодові.

За допомогою функції strpos можна знайти номер тільки першого входження стрічки у задану стрічку. Природно, існують функції, що дозволяють обчислити номер останньої появи стрічки у заданій стрічці. Це функція strrpos().

Її синтаксис такий:

```
strrpos (задана стрічка, символ для пошуку)
```

На відміну від strpos() ця функція дозволяє знайти позицію останнього входження в стрічку зазначеного символу. Не можна шукати позицію стрічки, лише символа.

Бувають ситуації, коли знати позицію, де знаходиться шукана стрічка, не обов'язково, а потрібно просто одержати всі символи, що розташовані після входження цієї стрічки. Можна, звичайно, скористатися і наведеними вище функціями strpos() і strrpos(), але можна зробити і простіше – вирізати підстрічку за допомогою призначених саме для цього функцій.

2. Вирізання стрічки

Функція strstr

Говорячи про вирізання підстрічки із шуканої стрічки в мові PHP, у першу чергу варто відзначити функцію strstr():

```
strstr (задана стрічка, стрічка для пошуку)
```

Вона знаходить перше входження шуканої стрічки і повертає підстрічку, починаючи з цієї шуканої стрічки до кінця заданої стрічки.

Якщо стрічка для пошуку не знайдена, то функція поверне false. Якщо стрічка для пошуку не належить стрічковому типові даних, то вона перетворюється в ціле число і розглядається як код символу. Крім того, ця функція чуттєва до регістра, тобто, якщо ми будемо паралельно шукати

входження слів «Ідея» і «ідея», то результати будуть різними. Замість strstr() можна використовувати абсолютно ідентичну їй функцію strchr().

Приклад. Виріжемо із стрічки, що містить назву й автора дослідження, підстрічку, що починається зі слова «Назва»:

```
<?
$str = "Автор: Іванов Іван (<a href=mailto:van@mail.ru>написати лист</a>),
  Назва: 'Дослідження мов програмування' ";
echo "<b>Задана стрічка: </b>",$str;
if (!strstr($str, "Назва"))
  echo "Стрічка не знайдена<br>";
else echo "<p><b>Отримана підстрічка: </b>",strstr($str, "Назва");
?>
```

У результаті отримаємо:

```
Задана стрічка:
Автор: Іванов Іван (написати лист), Назва: 'Дослідження мов програмування'
Отримана підстрічка: Назва: 'Дослідження мов програмування'
```

Для реалізації регістронезалежного пошуку підстрічки існує відповідний аналог цієї функції – функція stristr (задана стрічка, шукана стрічка). Діє і використовується вона аналогічно, як і strstr(), за винятком того, що регістр, у якому записані символи шуканої стрічки, не відіграє ролі при пошуку.

Очевидно, що функція strstr() не надто часто використовується – на практиці рідко буває потрібно отримувати підстрічку, що починається з визначеного слова або стрічки. Але в деяких випадках і вона може знадобитися. Крім того, у PHP є і більш зручні функції для пошуку входжень. Найбільш могутні з них, зазвичай, зв'язані з регулярними виразами.

Функція substr

Іноді ми не знаємо, з яких символів починається шукана стрічка, але знаємо, наприклад, що починається вона з п'ятого символу і закінчується за два символи до кінця заданої стрічки. Як вирізати підстрічку за такими вимогами? Дуже просто, за допомогою функції substr().

Її синтаксис наступний:

substr (задана стрічка, позиція початкового символу [, довжина])

Ця функція повертає частину стрічки довжиною, заданою параметром довжини, починаючи із символу, зазначеного параметром позиції початкового символу. Позиція, з якої починається вирізувана підстрічка, може бути як додатнім цілим числом, так і від’ємним. В останньому випадку відлік елементів починається з кінця стрічки. Якщо параметр довжини опущений, то substr() повертає підстрічку від зазначеного символу і до кінця заданої стрічки. Довжина вирізуваної підстрічки теж може бути задана від’ємним числом. Це означає, що зазначене число символів відкидається з кінця стрічки.

Приклад. Припустимо, в нас є фраза, виділена жирним шрифтом за допомогою тега <b> мови HTML. Ми хочемо одержати цю ж фразу, але в звичайному стилі. Напишемо таку програму:

```
<?php
$word = "<b>Hello, world!</b>";
```

```
echo $word , "<br>";
$pure_str = substr($word, 3, -4);
/* виділяємо підстрічку, починаючи з 3-го символу,
  не включаючи 4 символи з кінця Стрічки */
echo $pure_str;
?>
```

Функція strip_tags

Насправді розв'язати таку задачу можна набагато простіше, за допомогою функції strip_tags:

strip_tags (стрічка [, припустимі теги])

Ця функція повертає стрічку, з якої вилучені всі html і php-теги. За допомогою додаткового аргументу можна задати теги, що не будуть вилучені з стрічки. Список з декількох тегів вводиться без яких-небудь розділових знаків. Функція видає попередження, якщо зустрічає неправильні або неповні теги.

```
<?php
$string = "<b>Bold text</b>
        <i>Italic text</i>";
$str = strip_tags($string);
  // видаляємо всі теги з Стрічки
$str1 = strip_tags($string, '<i>');
  // видаляємо всі теги крім тега <i>
$str2 = strip_tags($string, '<i><b>');
  // видаляємо всі теги крім тегов <i> і <b>
echo $str,"<br>",$str1,"<br>", $str2;
?>
```

Наведемо інший приклад використання функції substr(). Припустимо, в нас є якесь повідомлення з вітанням і підписом автора. Ми хочемо вирізати спочатку вітання, а потім і підпис, залишивши тільки змістову частину повідомлення.

```
<?php
$text = "Привіт! Сьогодні ми вивчаємо роботу з Стрічками. Автор.";
$no_hello = substr($text, 8);
   // забираємо вітання
$content = substr($text, 8, 39);
   // те ж саме, що substr($text, 8, -6).
   // Забираємо підпис.
echo $text, "<br>", $no_hello,
    "<br>", $content;
?>
```

У результаті одержимо:

```
Привіт! Сьогодні ми вивчаємо роботу з Стрічками. Автор.
Сьогодні ми вивчаємо роботу з Стрічками. Автор.
Сьогодні ми вивчаємо роботу з Стрічками.
```

Якщо нам потрібно отримати один конкретний символ із стрічки, знаючи його порядковий номер, то не слід використовувати функції типу substr. Можна скористатися більш простим синтаксисом – записуючи номер символу у фігурних дужках після імені стрічкової змінної. У контексті

попереднього прикладу букву «р», розташовану другою за рахунком, можна отримати так:

```
echo $text{1}; // виведе символ "р"
```

Відмітимо, що номером цього символу є число один, а не два, тому що нумерація символів стрічки починається з нуля.

Оскільки ми почали говорити про символи в стрічці і їх нумерацію, то мимоволі виникає запитання, скільки всього символів у стрічці і як це обчислити. Кількість символів у стрічці – це довжина стрічки. Обчислити довжину стрічки можна за допомогою функції **strlen (Стрічка)**. Наприклад, довжина стрічки «Розробка інформаційної моделі» обчислюється за допомогою команди: strlen("Розробка інформаційної моделі"); і дорівнює 32 символам.

Отже, як вирізати і знаходити підстрічки, ми розглянули. Тепер навчимося заміняти стрічки, що входять до складу заданої стрічки, на іншу стрічку за нашим вибором.

3. Заміна входження підстрічки

Функція str_replace

Для заміни підстрічки можна використовувати функцію str_replace(). Це проста і зручна функція, що дозволяє розв’язати безліч задач, що не вимагають особливих тонкостій при виборі замінимої підстрічки. Для того щоб робити заміни з більш складними умовами, використовують механізм регулярних виразів і відповідні функції ereg_replace() і preg_replace(). Синтаксис функції str_replace() такий:

```
str_replace(шукане значення, значення для заміни, об'єкт)
```

Функція str_replace() шукає в розглянутому об'єкті значення і заміняє його значенням, призначеним для заміни. Чому ми говоримо тут не про стрічки для пошуку і заміни та вихідну стрічку, а про значення й об'єкт, в якому відбувається заміна? Справа в тому, що, починаючи з PHP 4.0.5, будь-який аргумент цієї функції може бути масивом.

Якщо об'єкт, в якому виконуються пошук і заміна, є масивом, то ці дії виконуються для кожного елемента масиву й у результаті повертається новий масив.

```
<?php
$greeting = array("Привіт", "Привіт усім!","Привіт, дорога!"); // об'єкт
$new_greet = str_replace("Привіт", "Добрий ранок", $greeting);
  // робимо заміну
print_r($new_greet);
/* одержимо: Array ([0]=>Добрий ранок [1]=>Добрий ранок усім!
[2]=>Добрий ранок, дорога!) */
?>
```

Якщо шукане значення і значення для заміни – масиви, то береться по одному значенню з кожного масиву і виконується їхній пошук і заміна в об'єкті. Якщо значень для заміни менше, ніж значень для пошуку, то як нові значення використовується порожня стрічка.

```
<?php
```

```
$greeting = array("Привіт", "Привіт усім!","Привіт, дорога!",
"Здрастуйте","Здрастуйте, товариші", "Ні");    // об'єкт
$search = array ("Привіт", "Здрастуйте", "Ні"); // значення, що будемо заміняти
$replace = array ("Добрий ранок", "День добрий"); // значення, якими будемо заміняти
$new_greet = str_replace($search, $replace, $greeting);// робимо заміну
print_r($new_greet); //виводимо отриманий масив
?>
```

У результаті одержимо такий масив:

```
Array ([0] => Добрий ранок
    [1] => Добрий ранок усім!
    [2] => Добрий ранок, дорога!
    [3] => День добрий
    [4] => День добрий, товариші
    [5] =>)
```

Якщо значення для пошуку – масив, а значення для заміни – стрічка, то ця стрічка буде використана для заміни всіх знайдених значень.

```
<?php $greeting = array("Привіт", "Привіт усім!", "Привіт, дорога!", "Здрастуйте",
  "Здрастуйте, товариші");    // об'єкт
$search = array ("Привіт","Здрастуйте"); // значення, що будемо заміняти
$replace = "День добрий"; // значення, яким будемо заміняти
$new_greet = str_replace($search, $replace, $greeting); // робимо заміну
print_r($new_greet);//виводимо отриманий масив
?>
```

Одержимо:

```
Array ([0] => День добрий
    [1] => День добрий усім!
    [2] => День добрий, дорога!
    [3] => День добрий
    [4] => День добрий, товариші)
```

Функція str_replace() чуттєва до регістра, але існує її регістронезалежний аналог – функція str_ireplace(). Однак ця функція підтримується не в усіх версіях PHP.

Функція substr_replace

Ця функція сполучає у собі властивості двох вже розглянутих нами функцій – функції str_replace() і substr(). Її синтаксис такий:

substr_replace (вихідна стрічка, стрічка для заміни, позиція початкового символу [, довжина])

Ця функція заміняє частину стрічки стрічкою, призначеною для заміни. Заміняється та частина стрічки (тобто підстрічка), що починається з позиції, зазначеної параметром позиції початкового символу. За допомогою додаткового аргументу довжина можна обмежити кількість замінимих символів. Тобто, фактично, ми не вказуємо конкретно стрічку, яку потрібно замінити, ми тільки описуємо, де вона знаходиться і, можливо, яку довжину має. У цьому відмінність функції substr_replace() від str_replace().

Як і у випадку з функцією substr() аргументи позиція початкового символу і довжини можуть бути від'ємними. Якщо позиція початкового символу від'ємна, то заміна виконується, починаючи з цієї позиції щодо кінця стрічки. Від'ємна довжина задає, скільки символів від кінця стрічки не повинна бути заміненою. Якщо довжина не вказується, то заміна відбувається до кінця стрічки.

```
<?php
$text = "Мене кличуть Вася.";
echo "Задана стрічка: $text<hr>\n";
/* Наступні два рядки замінять усю
задану стрічку рядком 'А мене – Петя' */
echo substr_replace($text, 'А мене – Петя', 0) . "<br>\n";
echo substr_replace($text, 'А мене – Петя', 0, strlen($text)) . "<br>\n";
// Наступна стрічка додасть слово 'Привіт! '
// у початок заданої стрічки
echo substr_replace($text, 'Привіт! ', 0, 0). "<br>\n";
// Наступні два рядки замінять ім'я Вася
// на ім'я Іван у заданій стрічці
echo substr_replace($text, 'Іван', 11, -1) . "<br>\n";
echo substr_replace($text, 'Іван', -5, -1) . "<br>\n";
?>
```

У результаті роботи цього скрипта одержимо:

```
Вихідна стрічка: Мене кличуть Вася.
А мене – Петя
А мене – Петя
Привіт! Мене кличуть Вася.
Мене кличуть Іван.
Мене кличуть Іван.
```

4. Поділ і з'єднання стрічок

Дуже корисні функції – функція поділу стрічки на частини і зворотна їй функція об'єднання стрічок в одну стрічку. Чому дуже корисні? Наприклад, якщо ви динамічно генеруєте форму за бажанням користувача, то можна запропонувати йому вводити елементи для створення списку вибору, розділяючи їх яким-небудь символом. І для того щоб обробити отриманий список значень, саме і знадобиться вміння розбивати стрічку на частини. Для реалізації такої розбивки в PHP можна використовувати кілька функцій:

```
explode(роздільник, задана стрічка [,максимальна кількість елементів])
split (шаблон, задана стрічка [, максимальна кількість елементів])
preg_split (шаблон, задана стрічка [,максимальна кількість елементів     [,прапори]])
```

Останні дві функції працюють з регулярними виразами, тому в даній лекції ми їх розглядати не будемо. Розглянемо більш просту функцію – explode().

Функція explode() поділяє задану стрічку на підстрічки, кожна з яких відділена від сусідньої за допомогою зазначеного роздільника і повертає масив отриманих стрічок. Якщо задано додатковий параметр максимальної кількості елементів, то кількість елементів у масиві буде не більшою цього

параметра, в останній елемент записується весь залишок стрічки. Якщо як роздільник зазначено порожню стрічку «""», то функція explode() поверне false. Якщо символу роздільника у заданій стрічці немає, то повертається задана стрічка без змін.

Приклад. Ми хочемо створити елемент форми – список, що випадає, і значення для цього списку повинен ввести користувач, не знайомий з мовою html. Створимо таку форму:

```
<form action=exp.php>
Введіть варіанти для вибору автора статті через двокрапку (":"):<br>
  <input type=text name=author size=40>
  <br>
  <input type=submit value=Створити елемент>
</form>
```

Скрипт, що буде її обробляти (exp.php), може бути таким:

```
<?php
$str = $_GET["author"];
$names = explode(":",$str);
   // розбиваємо стрічку введену, користувачем за допомогою ":"
$s = "<select name=author>";
   // створюємо список, що випадає
foreach ($names as $k => $name) {
  $s .= "<option value=$k>$name";
   // додаємо елементи до списку
}
$s .= "</select>";
echo $s;
?>
```

Крім поділу стрічки на частини, іноді, навпаки, виникає необхідність об'єднання декількох стрічок в одне ціле. Функція, пропонована для цього мовою PHP, називається implode():

implode (масив стрічок, об’єднуючий елемент)

Ця функція об’єднує елементи масиву за допомогою переданого їй об'єднуючого елемента (наприклад, коми). На відміну від функції explode(), порядок аргументів у функції implode() не має значення.

Приклад. Припустимо, ми зберігаємо ім'я, прізвище і по батькові людини окремо, а виводити їх на сторінці потрібно разом. Щоб з'єднати їх в одну стрічка, можна використовувати функцію implode():

```
<?php
$data = array("Іванов","Іван","Іванович");
$str = implode($data," ");
echo $str;
?>
```

У результаті роботи цього скрипта одержимо стрічку:

Іванов Іван Іванович

У функції implode() існує аналог – функція join(), тобто ці дві функції відрізняються лише іменами.

Отже, ми завершили знайомство з функціями роботи із стрічкими мови PHP. Звичайно ж, ми торкнулися далеко не всіх існуючих функцій, а розглянули лише малу частину. Ми вивчили функції, що дозволяють знайти

набір символів у стрічці, функції, що заміняють усі входження однієї стрічки на іншу, функції поділу стрічки на частини і з'єднання декількох стрічок в одну.

Питання для самоконтролю:

1. Що таке стрічка?
2. Назвіть основні способи її представлення.
3. Як відбувається пошук елемента в стрічці?
4. Назвіть функції вирізання підстрічки.
5. Як відбувається поділ і з’єднання стрічки?

Тема. Функції роботи з файловою системою PHP

План

1. Функція fopen.
2. Закриття з'єднання з файлом.Запис даних у файл.
3. Читання даних з файлу:
   - Функція fread;
   - Функція fgets;
   - Функція fgetss;
   - Функція fgetc;
   - Функція readfile;
   - Функція file;
   - Функція file_get_contents.
4. Перевірка існування файлу:
   - Функція file_exists;
   - Функція is_writable;
   - Функція is_readable.

1. Функція fopen

Взагалі, у PHP не існує функції, призначеної саме для створення файлів. Більшість функцій працюють із вже існуючими файлами у файловій системі сервера. Є кілька функцій, що дозволяють створювати тимчасові файли, або, що те ж саме, файли з унікальним для поточної директорії ім'ям. А от для того, щоб створити сам файл, потрібно скористатися функцією, що відкриває локальний або вилучений файл. Називається ця функція fopen(). Що означає «відкриває файл»? Це означає, що fopen пов'язує даний файл із потоком керування програми. Причому зв'язок буває різним у залежності від того, що ми хочемо робити з цим файлом: читати його, записувати в нього дані або робити і те й інше. Синтаксис цієї функції такий:

resource fopen (ім'я_файлу, тип_доступу[, use_include_path])

У результаті роботи ця функція повертає вказівник на відкритий нею файл. Як параметри цієї функції передаються: ім'я файлу, який потрібно відкрити, тип доступу до файлу (визначається тим, що ми збираємося робити з ним) і, можливо, параметр, що визначає, чи шукати зазначений файл у

include_path. Є ще один параметр, але про нього ми говорити не будемо, щоб не ускладнювати виклад. Обговоримо докладніше кожний з цих трьох параметрів.

Параметр ім'я_файлу повинен бути рядком, що містить вірне локальне ім'я файлу або URL-адреса файлу в мережі. Якщо ім'я файлу починається з вказівки протоколу доступу (наприклад, http://... або ftp://...), то інтерпритатор вважає це ім'я адресою URL і шукає оброблювач зазначеного в URL протоколу. Якщо оброблювач знайдений, то PHP перевіряє, чи дозволено працювати з об'єктами URL як зі звичайними файлами (директива allow_url_fopen). Якщо allow_url_fopen=off, то функція fopen викликає помилку і генерується попередження. Якщо ім'я файлу не починається з протоколу, то вважається, що зазначено ім'я локального файлу. Щоб відкрити локальний файл, потрібно, щоб PHP мав відповідні права доступу до цього файла.

Параметр use_include_path, встановлений із значенням 1 або TRUE, змушує інтерпретатор шукати зазначений у fopen() файл у include_path. Нагадаємо, що include_path – це директива з файла налаштувань PHP, що задає список директорій, в яких можуть знаходитися файли для включення. Крім функції fopen(), вона використовується функціями include() і require().

Параметр тип_доступу може приймати одне з наступних значень (див. таб.).

Таблиця. Значення, що можуть прийматися параметром тип доступу

Тип доступу	Опис
r	Відкриває файл тільки для читання; установлює покажчик позиції у файлі на початок файла.
r+	Відкриває файл для читання і запису; установлює покажчик файлу на його початок.
w	Відкриває файл тільки для запису; установлює покажчик файлу на його початок і усікає файл до нульової довжини. Якщо файл не існує, то намагається створити його.
w+	Відкриває файл для запису і для читання; встановлює покажчик файла на його початок і усікає файл до нульової довжини. Якщо файл не існує, то намагається створити його.
a	Відкриває файл тільки для запису; установлює покажчик файла в його кінець. Якщо файл не існує, то намагається створити його.
a+	Відкриває файл для запису і для читання; установлює покажчик файлу в його кінець. Якщо файл не існує, то намагається створити його.
x	Створює і відкриває файл тільки для запису; поміщає покажчик файла на його початок. Якщо файл вже існує, то fopen() повертає false і генерується попередження. Якщо файл не існує, то робиться спроба створити його. Цей тип доступу підтримується починаючи з версії PHP 4.3.2 і працює тільки з локальними файлами.
x+	Створює і відкриває файл для запису і для читання; поміщає покажчик файлу на його початок. Якщо файл вже існує, то fopen() повертає false і генерується попередження. Якщо файл не існує, то робиться спроба створити його. Цей тип доступу підтримується, починаючи з версії PHP 4.3.2, і працює тільки з локальними файлами.

Отже, щоб створити файл, потрібно, відкрити неіснуючий файл на запис.

```
<?php
$h = fopen(“my_file.html”,”w”);
/* відкриває на запис файл my_file.html,
якщо він існує, або створює порожній файл із таким ім'ям, якщо його ще немає */
$h = fopen(“dir/another_file.txt”,”w+”);
/* відкриває на запис і читання або створює файл another_file.txt у директорії dir */
$h = fopen(“http://www.server.ru/dir/file.php”,”r”);
/* відкриває на читання файл, що знаходиться за зазначеною адресою*/
?>
```

Створюючи файл, потрібно враховувати, під якою операційною системою ви працюєте, і під якою ОС приблизно цей файл буде читатися. Справа в тому, що різні операційні системи по-різному позначають кінець рядка. У Unix-подібних ОС кінець рядка позначається \n, у системах типу Windows – \r\n. Windows пропонує спеціальний прапор t для перекладу символів кінця рядка систем типу Unix у свої символи кінця рядка. На противагу цьому існує прапор b, використовується найчастіше для бінарних файлів, завдяки якому такої трансляції не відбувається. Використовувати ці прапори можна, просто дописавши їх після останнього символу обраного типу доступу до файла. Наприклад, відкриваючи файл на читання, замість r варто використовувати rt, щоб перекодувати всі символи кінця рядка в \r\n. Якщо не використовувати прапор b при відкритті бінарних файлів, то можуть з'являтися помилки, пов'язані зі зміною вмісту файлу. З розумінь перенесення програми на різні платформи рекомендується завжди використовувати прапор b при відкритті файлів за допомогою fopen().

Що відбувається, якщо відкрити або створити файл за допомогою fopen не вдається? У цьому випадку PHP генерує попередження, а функція fopen повертає як результат своєї роботи значення false. Такого роду попередження можна «придушити» (заборонити) за допомогою символу @.

Наприклад, така команда не виведе попередження, навіть тоді якщо відкрити файл не вдалося:

```
$h = @fopen(“dir/another_file.txt”,”w+”);
```

Таким чином, функція fopen() дозволяє створити лише порожній файл і зробити його доступним для запису. Як же записати дані в цей файл? Як прочитати дані з вже існуючого файла?

Перш ніж відповісти на ці питання, розглянемо, як закрити встановлене за допомогою fopen() з'єднання.

2. Закриття з'єднання з файлом

Після виконання необхідних дій з файлом, чи то читання або запис даних або яке-небудь інше, з'єднання, встановлене з цим файлом функцією fopen(), потрібно закрити. Для цього використовують функцію fclose(). Синтаксис якої наступний:

fclose (покажчик на файл)

Ця функція повертає TRUE, якщо з'єднання успішно закрито, і FALSE – у протилежному випадку. Параметр цієї функції повинен вказувати на файл, успішно відкритий, наприклад, за допомогою функції fopen().

```
<?php
$h = fopen("my_file.html","w");
fclose($h);
?>
```

Звичайно, якщо не закривати з'єднання з файлом, то ніяких помилок виконання скрипта не відбудеться. Але в цілому для сервера це може мати серйозні наслідки. Наприклад, хакер може скористатися відкритим з'єднанням і записати у файл вірус, не говорячи вже про зайву витрату ресурсів сервера. Так що радимо завжди закривати з'єднання з файлом після виконання необхідних дій.

Запис даних у файл

Функція fwrite

Для того щоб записати дані у файл, доступ до якого відкритий функцією fopen(), можна використовувати функцію fwrite(). Синтаксис якої наступний:

int fwrite (покажчик на файл, рядок [, довжина])

Ця функція записує вміст рядка стрічкою у файл, на який вказує покажчик на файл. Якщо зазначено додатковий аргумент довжини, то запис закінчується після того, як записана кількість символів, рівна значенню цього аргументу, або коли буде досягнутий кінець рядка.

У результаті своєї роботи функція fwrite() повертає кількість записаних байтів або false у випадку помилки.

Приклад. Нехай у нашій робочій директорії немає файла my_file.html. Створимо його і запишемо в нього рядок тексту:

```
<?php
$h = fopen("my_file.html","w");
$text = "Цей текст запишемо у файл.";
if (fwrite($h,$text))
  echo "Запис пройшов успішно";
else
  echo "Відбулася помилка при записі даних";
fclose($h);
?>
```

У результаті роботи цього скрипта в браузері ми побачимо повідомлення про те, що запис пройшов успішно, а у файлі my_file.html з'явиться рядок “Цей текст запишемо у файл.”. Якби цей файл існував до того, як ми виконали цей скрипт, усі дані, що знаходилися в ньому були б вилучені.

Якщо ж ми напишемо такий скрипт:

```
<?php
$h = fopen("my_file.html","a");
$add_text = "Додамо текст у файл.";
if(fwrite($h,$add_text,7))
```

```
  echo "Додавання тексту пройшло успішно<br>";
else echo "Відбулася помилка при додаванні даних<br>";
fclose($h);
?>
```

то до рядка, що вже існує у файлі my_file.html, додасться ще сім символів з рядка, що міститься в змінній $add_text, тобто слово «Додамо»

Далі ми розглянемо, які методи читання даних з файлу пропонує мова PHP.

3. Читання даних з файла

Якщо ми хочемо прочитати дані з існуючого файла, однієї функції fopen(), як і у випадку з записом даних, недостатньо. Вона лише повертає покажчик на відкритий файл, але не зчитує ні одного рядка з цього файлу. Тому для того, щоб прочитати дані з файлу, потрібно скористатися однією зі спеціальних функцій: file, readfile, file_get_contents, fread, fgets і т.п.

Функція fread

Ця функція здійснює читання даних з файла. Її можна використовувати і для читання даних з бінарних файлів, не боюючись їхнього пошкодження. Синтаксис fread() такий:

string fread (покажчик на файл, довжина)

При викликанні цієї функції відбувається читання даних довжиною (у байтах), визначеною параметром довжини, з файла, на який вказує покажчик на файл. Параметр покажчик на файл повинен бути реально існуючою змінною типу ресурс, що містить у собі зв'язок з файлом, відкритим, наприклад, за допомогою функції fopen(). Читання даних відбувається доти, поки не зустрінеться кінець файлу або поки не буде прочитане зазначене параметром довжини число байтів.

У результаті роботи функція fread() повертає рядок з зчитаною з файла інформацією.

Як ви помітили, у цій функції параметр довжини – обов'язковий. Отже, якщо ми хочемо зчитати весь файл у стрічку, потрібно знати її довжину. PHP може самостійно обчислити довжину зазначеного файла. Для цього потрібно скористатися функцією filesize(ім'я файла). У випадку помилки ця функція поверне false. На жаль, її можна використовувати лише для одержання розміру локальних файлів.

Приклад. Прочитаємо вміст файла my_file.html

```
<?php
$h = fopen("my_file.html","r+");
// відриваємося файл на запис і читання
$content = fread($h,filesize(“my_file.html”));
// зчитуємо вміст файла в стрічку
fclose($h); // закриваємо з'єднання з файлом
echo $content;
// виводимо вміст файла на екран браузера
?>
```

Для того щоб зчитати вміст бінарного файла, наприклад зображення, у таких системах, як Windows, рекомендується відкривати файл за допомогою прапора rb або йому подібного, що містять символ b наприкінці.

Функція filesize() кешує результати своєї роботи. Якщо змінити вміст файлу my_file.html і знову запустити наведений вище скрипт, то результат його роботи не зміниться. Більше того, якщо запустити скрипт, що зчитує дані з цього файла за допомогою іншої функції (наприклад, fgetss), то результат може виявитися таким, як ніби файл не змінився. Щоб цього уникнути, потрібно очистити статичний кэш, додавши в код програми команду clearstatcache();

Функція fgets

За допомогою функції fgets() можна зчитати з файла стрічку тексту. Синтаксис цієї функції практично такий самий, як і в fread(), за вийнятком того, що довжину стрічки, яка зчитується, вказувати необов'язково:

string fgets (покажчик на файл [, довжина])

У результаті роботи функція fgets() повертає рядок довжиною (довжина–1) байт із файлу, на який вказує покажчик на файл. Читання закінчується, якщо прочитано (довжину–1) символів і зустрівся символ кінця стрічки або кінець файла. Нагадаємо, що в PHP один символ – це один байт. Якщо довжина стрічки, що зчитується, не зазначена (дана можливість з'явилася, починаючи з PHP 4.2.0), то зчитується 1 Кбайт (1024 байт) тексту або, що те саме, 1024 символу. Починаючи з версії PHP 4.3, якщо параметр довжина не задана, зчитується стрічка цілком. У випадку помилки функція fgets() повертає false. Для версій PHP, починаючи з 4.3, ця функція безпечна для двійкових файлів.

```
<?php
$h = fopen("my_file.html","r+");
$content = fgets($h,2);
// зчитує перший символ з першого рядка файла my_file.html
fclose($h);
echo $content;
?>
```

Обидві функції, fread() і fgets(), припиняють зчитування даних з файла, якщо зустрічають кінець файла. У PHP є спеціальна функція, що перевіряє, чи досягнув вказівник позиції файла на кінець файла. Це булева функція **feof()**, як параметр якої передається вказівник на з'єднання з файлом. Наприклад, ось так можна зчитати всі рядки файла my_file.html:

```
<?php
$h = fopen("my_file.html","r");
while (!feof ($h)) {
   $content = fgets($h);
   echo $content,"<br>";
}
fclose($h);
?>
```

Функція fgetss

Існує різновид функції fgets() – функція fgetss(). Вона теж дозволяє зчитувати стрічку із зазначеного файла, але при цьому видаляє з нього всі html-теги, що зустрілися, за винятком, можливо, деяких. Синтаксис fgetss() такий:

string fgetss(покажчик на файл, довжина [, припустимі теги])

Зверніть увагу, що тут аргумент довжина обов'язковий.

Приклад. Нехай у нас існує файл my_file.html наступного змісту:

```
<h1>Без праці не виймеш і рибку зі ставка</h1>
<b>Тихіше їдеш – далі будеш</b>
У семи няньок<i> дитя без ока</i>.
Виведемо на екран усі стрічки файла my_file.html,
видаливши з них всі теги, крім <b> і <i>:
<?php
$h = fopen("my_file.html","r");
while (!feof ($h)) {
   $content = fgetss($h,1024,'<b><i>');
   echo $content,"<br>";
}
fclose($h);
?>
```

У результаті роботи цього скрипта одержимо:

```
Тихіше їдеш – далі будеш
Без праці не виймеш і рибку зі ставка.
У семи няньок дитя без ока.
```

Функція fgetc

Природньо, якщо можна зчитувати інформацію з файла пострічково, то можна зчитувати її і посимвольно. Для цього призначена функція fgetc(). Легко здогадатися, що синтаксис у неї наступний:

string fgetc (покажчик на файл)

Ця функція повертає символ з файла, на який посилається вказівник на файл, і значення, що обчислюється як FALSE, якщо зустрінуто кінець рядка.

От так, наприклад, можна зчитати файл по одному символу:

```
<?php
$h = fopen("my_file.html","r");
while (!feof ($h)) {
   $content = fgetc($h);
   echo $content,"<br>";
}
fclose($h);
?>
```

Насправді для того, щоб прочитати вміст файла, відкривати з'єднання з ним за допомогою функції fopen() зовсім не обов'язково. У PHP є функції, що дозволяють робити це, використовуючи лише ім'я файла. Це функції readfile( ), file( ) і file_get_contents( ). Розглянемо кожну зних докладніше.

Функція readfile

Синтаксис:

int readfile (ім'я_файла [, use_include_path])

Функція readfile() зчитує файл, ім'я якого передане їй як параметр ім'я_файла, і виводить його вміст на екран. Якщо додатковий аргумент

use_include_path має значення TRUE, то пошук файла з заданим ім'ям виконується і по директоріях, що входять у include_path.

У програму ця функція повертає кількість зчитатих байтів (символів) файла, а у випадку помилки – FALSE. Повідомлення про помилку в цій функції можна ”придушити” оператором @ .

Приклад. Наступний скрипт виведе на екран вміст файла my_file1.html і розмір цього файла, якщо він існує. У протилежному випадку виведеться нам повідомлення про помилку – рядок "Error in readfile".

```
<?php
$n = @readfile ("my_file1.html");
/* виводить на екран вміст файла і записує його розмір у змінну $n */
if (!$n) echo "Error in readfile";
/* якщо функція readfile() виконалася с помилкою, то $n=false і виводимо
повідомлення про помилку */
else echo $n;
  // якщо помилки не було, то виводимо число кількості символів
?>
```

За допомогою функції readfile() можна читати вміст віддалених файлів, вказуючи їхню URL-адресу як ім'я файла, якщо ця опція не відключена в налаштуваннях сервера.

Відразу ж виводити вміст файла на екран не завжди зручно. Інколи потрібно записати інформацію з файла в змінну, щоб надалі зробити з нею які-небудь дії. Для цього можна використовувати функцію file() або file_get_contents().

Функція file

Функція file() призначена для зчитування інформації з файла в змінну типу масив. Синтаксис у неї такий же, як і у функції readfile(), за винятком того, що в результаті роботи вона повертає масив:

array file (ім'я_файла [, use_include_path])

Що ж за масив повертає ця функція? Кожен елемент даного масиву є стрічкою у файлі, інформацію з якого ми зчитуємо (його ім'я задане аргументом ім'я_файла). Символ нового рядка теж включається в кожен з елементів масиву. У випадку помилки функція file(), як і всі вже розглянуті, повертає false. Додатковий аргумент use_include_path знову ж визначає, чи шукати даний файл у директоріях include_path. Відкривати віддалені файли за допомогою цієї функції теж можна, якщо не заборонено сервером. Починаючи з PHP 4.3 робота з бінарними файлами за допомогою цієї функції стала безпечною.

Наприклад, маємо файл my_file.html наступного змісту:

```
<h1>Без праці не виймеш і рибку зі ставка</h1>
<b>Тихіше їдеш – далі будеш</b>
```

Прочитаємо його вміст за допомогою функції file():

```
<?php
$arr = file ("my_file.html");
foreach($arr as $i => $a) echo $i,": ", htmlspecialchars($a), "<br>";
?>
```

У результаті на екран буде виведене наступне повідомлення:

```
0: <h1>Без праці не виймеш і рибку зі ставка</h1>
1: <b>Тихіше їдеш – далі будеш</b>
```

Функція file_get_contents

У версіях PHP починаючи з 4.3 з'явилася можливість зчитувати вміст файла в стрічку. Робиться це за допомогою функції file_get_contents(). Як і дві попередні функції, як параметри вона приймає значення імені файла і, можливо, дати вказівку шукати його в директоріях include_path. Наведемо її синтаксис:

```
string file_get_contents (ім'я_файла [, use_include_path])
```

Ця функція абсолютно ідентична функції file(), тільки повертає вона вміст файла у вигляді стрічки. Крім того, вона безпечна для обробки бінарних даних і може зчитувати інформацію з віддалених файлів, якщо це не заборонено налаштуваннями сервера.

4. Перевірка існування файла

Отже, створювати файл ми навчилися, записувати дані в нього – навчилися, зчитувати дані з файла – теж навчилися. Але от питання: а що якщо файла, з яким ми намагаємося проробити всі ці операції, не існує? Або він недоступний для читання чи запису? Очевидно, що в такому випадку жодна з вивчених нами функцій працювати не буде і PHP видасть повідомлення про помилку. Щоб відстежувати такого роду помилки, можна використовувати функції file_exists(), is_writable(), is_readable().

Функція file_exists

Синтаксис:

```
bool file_exists (ім'я файла або директорії)
```

Функція file_exist() перевіряє, чи існує файл або директорія, ім'я якої передане їй як аргумент. Якщо директорія або файл у файловій системі сервера існує, то функція повертає TRUE, у протилежному випадку – FALSE. Результат роботи цієї функції кешується. Відповідно очистити кэш можна, як вже відзначалося, за допомогою функції clearstatcache(). Для нелокальних файлів використовувати функцію file_exists() не можна.

```
<?php
$filename = 'c:/users/files/my_file.html';
if (file_exists($filename)) {
  print "Файл <b>$filename</b> існує";}
else {  print "Файл <b>$filename</b> НЕ існує";
}
?>
```

Функція is_writable

Якщо, крім перевірки існування файла потрібно довідатися ще, чи дозволено записувати інформацію в цей файл, варто використовувати функцію is_writable() або її аналог – функцію is_writeable().

Синтаксис:

```
bool is_writable (ім'я файла або директорії)
```

Ця функція повертає TRUE, якщо файл (або директорія) існує і доступний для запису. Доступ до файла здійснюється під тим обліковим записом користувача, під яким працює сервер (найчастіше це користувач nobody або www). Результати роботи функції is_writable кешуються.

Функція is_readable

Якщо, крім перевірки існування файла потрібно довідатися ще, чи дозволено читати інформацію з нього, слід використовувати функцію is_readable().

Синтаксис:

```
bool is_readable (ім'я файла)
```

Ця функція працює подібно функції is_writable().

```
<?php
$filename = 'c:/users/files/my_file.html';
if (is_readable($filename)) {
  print "Файл <b>$filename</b> існує і доступний для читання";}
else {  print "Файл <b>$filename</b> НЕ існує або НЕ доступний для читання";
}
?>
```

Підіб'ємо підсумки. У цій лекції ми вивчили, як створювати файли за допомогою мови PHP, як записувати дані у файли за допомогою PHP, як зчитувати з них інформацію різними способами, як перевіряти існування і доступність файла для запису і читання.

Питання для самоконтролю:

1. Дайте характеристику функції fopen.
2. Дайте визначення функціям читання даних з файла.
3. Як відбувається запис даних у файл.
4. Охарактеризуйте функції перевірки існування файла.

Тема. Авторизація доступу за допомогою сесій

План

1. Авторизація доступу.
2. Механізм сесій.
3. Створення сесії.
4. Реєстрація змінних сесії. Знищення змінних сесії.

Ця частина лекції присвячена вивченню питань забезпечення безпеки в мережі і використанню для цих цілей механізму сесій. Розглядаються: ініціалізація сесій, передача ідентифікатора користувача, реєстрація змінних сесії, знищення сесії. На завершення наводиться приклад авторизації користувача за допомогою механізму сесій.

Також ми розглянемо, що таке сесії й у чому їхня специфіка в PHP, розв'яжемо одну з основних задач, що виникає при побудові більш-менш складних інформаційних систем (сайтів) - завдання авторизації доступу користувачів до ресурсів системи, а також обговоримо безпеку побудованого розв'язку.

1. Авторизація доступу

Що таке авторизація доступу? Спробуємо пояснити на прикладі зі звичайного життя. Ви хочете взяти в бібліотеці книгу. Але ця послуга доступна тільки тим, у кого є читацький квиток. Можна сказати, що за допомогою цього квитка виконується "авторизація доступу" до бібліотечних ресурсів. Бібліотекар після пред'явлення йому читацького квитка знає, хто бере книгу, і в разі потреби (наприклад, книгу довго не повертають) може вжити заходів (подзвонити боржникові додому). Бібліотекар має набагато більше прав, ніж звичайний відвідувач: він може давати або не давати книги визначеному відвідувачеві, може виставляти на показ новинки і забирати в архів книги, що рідко читаються, і т.п.

В інформаційних технологіях усе приблизно так само. У мережі існує величезна кількість ресурсів, тобто безліч "бібліотек". У кожної з них свій "бібліотекар", тобто людина або група людей, що відповідають за зміст ресурсу і надання користувачам інформації. Їх називають адміністраторами. Функції адміністратора, як правило, включають додавання нової інформації, знищення і редагування існуючої, налаштування способів відображення інформації користувачеві. А в функції користувача (простого відвідувача ресурсу) входить тільки пошук і перегляд інформації.

Як же відрізнити користувача від адміністратора? У реальній бібліотеці це очевидно, але якщо ролі бібліотекаря і відвідувача бібліотеки перенести у віртуальну реальність, то ця очевидність зникає. Бібліотекар, як і відвідувач, має доступ до бібліотечних ресурсів через Internet. А відповідно до протоколу HTTP усі клієнти абсолютно рівноправні. Як же зрозуміти, хто зайшов на сайт? Звичайний користувач (відвідувач) чи адміністратор (бібліотекар)? Якщо це простий користувач, то як зберегти цю інформацію, щоб не допустити відвідувача в закриті архіви сайту? Тобто виникає запитання, як ідентифікувати клієнта, що надіслав запит, і зберігати відомості про нього, поки він знаходиться на сайті?

Найпростіший варіант, що приходить у голову, - це реєстрація людини в системі і видача йому аналога читацького квитка, а саме логіна і пароля для входу в адміністративну частину системи. Ця інформація зберігається на комп'ютері-сервері, і при вході в систему перевіряється відповідність введених користувачем логіна і пароля тим, що зберігаються в системі. Правда, тут у порівнянні з реальною бібліотекою ситуація змінюється: читацький квиток потрібно бібліотекареві для входу в закриту частину системи, а читач може заходити на сайт вільно. У принципі можна реєструвати і простих відвідувачів. Тоді всіх зареєстрованих користувачів потрібно розділити на групи: бібліотекарі (адміністратори) і читачі (прості користувачі), наділивши їх відповідними правами. Ми не будемо вдаватися в ці тонкості і скористаємося найпростішим варіантом, коли введення логіна і пароля потрібно для доступу до деяких сторінок сайта.

Приклад.

У нас є файл index.html - домашня сторінка Васі Петрова

```
<html>
<head><title>My home page</title></head>
<body>
```

```
Привіт усім!
Мене звуть Вася Петров і це моя домашня сторінка.
<a href="secret_info.html">Для Петра</a>
</body></html>
```

і файл secret_info.html, що містить секретну інформацію, читати яку дозволено тільки другові Василя - Петру.

```
<html>
<head><title>Secret info</title></head>
<body>
Тут я хочу поділитися секретами з другом Петром.</p>
</body></html>
```

Якщо залишити обидва ці файли такими як є, то будь-який відвідувач, кликнувши на посилання "Для Петра", потрапить на секретну сторінку. Щоб цього уникнути, потрібно додати проміжний скрипт, що буде перевіряти, чи дійсно Петя хоче потрапити на секретну сторінку. І зробити так, щоб головний файл посилався не відразу на secret_info.html, а спочатку на цей скрипт.

```
<html>
 <head><title>My home page</title></head>
 <body>
  <p>Привіт усім!
  Мене звуть Вася Петров і це моя домашня сторінка.
  </p>
  <a href="authorize.php">Для Петра</a>
 </body>
</html>
```

Сам скрипт авторизації повинен містити форму для введення логіна і пароля, перевіряти їхню правильність і перенаправляти на секретну сторінку, якщо перевірка пройшла успішно, і видавати повідомлення про помилку в протилежному випадку.

```
<?
if (!isset($_GET['go'])){
   // перевіряємо, чи відправлені дані формою
   echo "<form>
   // форма для авторизації (введення логіна і пароля)
   Login: <input type=text name=login>
   Password: <input type=password name=password>
   <input type=submit name=go value=Go>
   </form>";}
else {
// якщо форма заповнена, то порівнюємо логін
// і пароль із правильними логіном і паролем
   if ($_GET['login']=="pit" && $_GET['passwd']=="123") {
   Header("Location: secret_info.html");
   //і перенаправляємо на секретну сторінку }
else echo "Невірне введення, спробуйте ще раз<br>";}
?>
```

Начебто усе досить просто. Але припустимо, що в нас не одна секретна сторінка, а декілька. Причому вони пов'язані між собою перехресними посиланнями. Тоді виникає необхідність постійно пам'ятати пароль і логін

відвідувача сайта (якщо він такий має). Щоб розв’язати цю проблему, можна в кожну сторінку вмонтувати скрипт, що буде передавати логін і пароль від сторінки до сторінки як сховані параметри форми. Але такий спосіб не зовсім безпечний: ці параметри можна перехопити і підробити. У PHP існує більш зручний і безпечний метод розв’язання такої проблеми збереження даних про відвідувача протягом сеансу його роботи із сайтом - це механізм сесій.

2. Механізм сесій

Сесії - це механізм, що дозволяє створювати і використовувати змінні, що зберігають своє значення протягом усього часу роботи користувача із сайтом.

Ці змінні для кожного користувача мають різні значення і можуть використовуватися на будь-якій сторінці сайта до виходу користувача із системи. При цьому щоразу, заходячи на сайт, користувач одержує нові значення змінних, що дозволяють ідентифікувати його протягом цього сеансу або сесії роботи із сайтом. Звідси і назва механізму - сесії.

Завдання ідентифікації користувача вирішується шляхом присвоєння кожному користувачеві унікального номера, так званого ідентифікатора сесії (SID, Session IDentifier). Він генерується PHP у той момент, коли користувач заходить на сайт, і знищується, коли користувач іде із сайта, і представляє собою рядок з 32 символів (наприклад, ac4f4a45bdc893434c95dcaffb1c1811). Цей ідентифікатор передається на сервер разом з кожним запитом клієнта і повертається назад разом з відповіддю сервера.

Існує кілька способів передачі ідентифікатора сесії:

- *За допомогою cookies.*

Cookies були створені спеціально як метод однозначної ідентифікації клієнтів і являють собою розширення протоколу HTTP. У цьому випадку ідентифікатор сесії зберігається в тимчасовому файлі на комп'ютері клієнта, що послав запит. Метод, безсумнівно, гарний, але багато користувачів відключають підтримку cookies на своєму комп'ютері через проблеми з безпекою.

- *За допомогою параметрів командного рядка.*

У цьому випадку ідентифікатор сесії автоматично вбудовується в усі запити (URL), передані серверові, і зберігається на стороні сервера.

Наприклад: адреса

http://green.nsu.ru/test.php перетворюється на адресу
http://green.nsu.ru/test.php?PHPSESSID=ac4f4a45bdc893434c95dcaffb1c1811

Цей спосіб передачі ідентифікатора використовується автоматично, якщо в браузера, що відправив запит, вимкнені cookies. Він досить надійний - передавати параметри в командному рядку можна завжди. З іншого боку, ідентифікатор сесії можна підглянути, скористатися збереженим варіантом у рядку браузера або підробити. Хоча, звичайно, усі ці проблеми або надумані або їх можна розв’язати. Наприклад, хто зможе запам'ятати рядок з 32 різних символів? А якщо правильно організувати роботу із сесіями (вчасно їх

знищувати), то навіть збережений у браузері номер сесії нічого не дасть. До питань безпеки ми ще повернемося наприкінці лекції.

Крім перерахованих варіантів передачі ідентифікатора сесії, відомо ще декілька, але ми їх розглядати не будемо через їхню складність.

3. Створення сесії

Перше, що потрібно зробити для роботи із сесіями, це запустити механізм сесій. Якщо в налаштуваннях сервера змінна session.auto_start встановлена в значення "0" (якщо session.auto_start=1, то сесії запускаються автоматично), то будь-який скрипт, у якому потрібно використовувати дані сесії, повинен починатися з команди

```
session_start();
```

Одержавши таку команду, сервер створює нову сесію або відновлює поточну, ґрунтуючись на ідентифікаторі сесії, переданому по запиті. Як це робиться? Інтерпретатор PHP шукає змінну, в якій зберігається ідентифікатор сесії (за замовчуванням це PHPSESSID) спочатку в cookies, потім у змінних, переданих за допомогою POST- і GET-запитів. Якщо ідентифікатор знайдений, то користувач вважається ідентифікованим, виконується зміна всіх URL і виставляння cookies. У протилежному випадку користувач вважається новим, для нього генерується новий унікальний ідентифікатор, потім виконується зміна URL і виставляння cookies.

Команду session_start() потрібно викликати у всіх скриптах, у яких мають бути використані змінні сесії, причому до виведення яких-небудь даних у браузер. Це пов'язано з тим, що cookies виставляються тільки до виведення інформації на екран.

Одержати ідентифікатор поточної сесії можна за допомогою функції session_id().

Для наочності сесії можна задати ім'я за допомогою функції session_name([ім'я_сесії]). Робити це потрібно ще до ініціалізації сесії. Одержати ім'я поточної сесії можна за допомогою цієї ж функції, викликаної без параметрів: session_name();

Приклад. Створення сесії

Переіменуємо наш файл index.html, щоб оброблялися php-скрипти, наприклад у Index.php, створимо сесію і подивимося, який вона одержить ідентифікатор та ім'я.

```
<? session_start();
   // створюємо нову сесію або відновлюємо поточну
echo session_id();
 // виводимо ідентифікатор сесії
?>
<html>
  <head><title>My home page</title></head>
... // домашня сторінка
</html>
<?
  echo session_name();
    // виводимо ім'я поточної сесії.
    // У даному випадку це PHPSESSID
```

```
?>
```

Якщо проробити те ж саме з файлом authorize.php, то значення виведених змінних (id сесії і її ім'я) будуть такими ж, якщо перейти на нього з index.php і не закривати перед цим вікно браузера (тоді ідентифікатор сесії зміниться).

4. Реєстрація змінних сесії

Однак від самих ідентифікатора й імені сесії нам користі для розв'язання наших задач небагато. Ми ж хочемо передавати і зберігати протягом сесії наші власні змінні (наприклад, логін і пароль). Для того, щоб цього домогтися, потрібно просто зареєструвати свої змінні:

session_register(ім'я_змінної1, ім'я_змінної2, ...);

Помітимо, що реєструються не значення, а імена змінних. Зареєструвати змінну досить один раз на будь-якій сторінці, де використовуються сесії. Імена змінних передаються функції session_register() без знака $. Усі зареєстровані в такий спосіб змінні стають глобальними (тобто доступними з будь-якої сторінки) протягом даної сесії роботи із сайтом.

Зареєструвати змінну також можна, просто записавши її значення в асоціативний масив $_SESSION, тобто написавши

$_SESSION['ім'я_змінної'] = 'значення_зміної';

У цьому масиві зберігаються всі зареєстровані (тобто глобальні) змінні сесії.

Доступ до таких змінних здійснюється за допомогою масиву $_SESSION['ім'я_змінної'] (або $HTTP_SESSION_VARS['ім'я_змінної'] для версії PHP 4.0.6 і більш ранніх). Якщо ж у налаштуваннях php вімкнена опція **register_globals**, то до сесійних змінних можна звертатися ще і як до звичайних змінних, наприклад так: $ім'я_змінної.

Якщо **register_globals=off** (відключені), то користуватися session_register() для реєстрації змінних переданих методами POST або GET, не можна, тобто це просто не працює. І взагалі, не рекомендується одночасно використовувати обидва методи реєстрації змінних, $_SESSION і session_register().

Приклад. Реєстрація змінних

Зареєструємо логін і пароль, що вводяться користувачем на сторінці авторизації.

```
<?
session_start();
   // створюємо нову сесію або відновлюємо поточну
if (!isset($_GET['go'])){
  echo "<form>
    Login: <input type=text name=login>
    Password: <input type=password name=passwd>
    <input type=submit name=go value=Go>
  </form>";}
else {  $_SESSION['login']=$_GET['login'];
   // реєструємо змінну login
   $_SESSION['passwd']=$_GET['passwd'];
```

```
  // реєструємо змінну passwd
// тепер логін і пароль - глобальні змінні для цієї сесії
   if ($_GET['login']=="pit" && $_GET['passwd']=="123") {
      Header("Location: secret_info.php");
      // перенаправляємо на сторінку secret_info.php}
else echo "Невірне введення,спробуйте ще раз<br>";
}
print_r($_SESSION);
   // виводимо всі змінні сесії
?>
```

Тепер, потрапивши на сторінку secret_info.php, та й на будь-яку іншу сторінку сайта, ми зможемо працювати з введеними користувачем логіном і паролем, що будуть зберігатися в масиві $_SESSION. Таким чином, якщо змінити код секретної сторінки (помітьте, ми перейменували її в secret_info.php) так:

```
<?php
session_start();
   // створюємо нову сесію або відновлюємо поточну
print_r($_SESSION);
   // виводимо всі змінні сесії
?>
<html>
<head><title>Secret info</title></head>
<body>
<p>Тут я хочу поділитися секретами з другом Петром.
</body>
</html>
```

То ми отримаємо в браузері на секретній сторінці наступне:

```
Array ( [login] => pit [passwd] => 123 )
Тут я хочу поділитися секретами з другом Петром.
```

У підсумку отримаємо список змінних, зареєстрованих на authorize.php і, власне, саму секретну сторінку.

Що це нам дає? Припустимо, хакер хоче прочитати секрети Васі і Петра. І він якось довідався, як називається секретна сторінка (або сторінки). Тоді він може спробувати просто ввести її адресу в рядку браузера, минаючи сторінку авторизації (введення пароля). Щоб уникнути такого проникнення в наші таємниці, потрібно дописати усього пару рядків у код секретних сторінок:

```
<?php
session_start();
   // створюємо нову сесію або відновлюємо поточну
print_r($_SESSION);
   // виводимо всі змінні сесії
if (!($_SESSION['login']=="pit" &&  $_SESSION['passwd']==123))
   // перевіряємо правильність пароля-логіна
   Header("Location: authorize.php");
   // якщо помилка, то перенаправляємо на сторінку авторизації
?>
<html>
<head><title>Secret info</title></head>
```

```
... // тут розташовується секретна інформація :)
</html>
```

Знищення змінних сесії

Крім вміння реєструвати змінні сесії (тобто робити їх глобальними протягом усього сеансу роботи), корисно також вміти знищувати такі змінні і сесію в цілому.

Функція session_unregister (ім'я_змінної) знищує глобальну перемінну з поточної сесії (тобто знищує її зі списку зареєстрованих змінних). Якщо реєстрація виконується за допомогою $_SESSION ($HTTP_SESSION_VARS для версії PHP 4.0.6 і більш ранніх), то використовують мовну конструкцію unset(). Вона не повертає ніякого значення, а просто знищує зазначені змінні.

Де це може знадобитися? Наприклад, для знищення даних про відвідувача (зокрема, логіна і пароля) після його виходу із секретної сторінки. Якщо правильні логін і пароль зберігаються і вікно браузера після відвідування сайта не закрили, то будь-який інший користувач цього комп'ютера зможе прочитати закриту інформацію.

Приклад. Знищення змінних сесії

У файл secret_info.php додамо рядок для виходу на головну сторінку:

```
<?php
// ... php код
?>
<html>
<head><title>Secret info</title></head>
... // тут розташовується секретна інформація :)
<a href="index.php">На головну</a>
</html>
```

У Index.php знищимо логін і пароль, введені раніше:

```
<?
session_start();
session_unregister('passwd');
   // знищуємо пароль
unset($_SESSION['login']);
   // знищуємо логин
print_r($_SESSION);
   // виводимо глобальні змінні сесії
?>
<html>
<head><title>My home page</title></head>
... // домашня сторінка
</html>
```

Тепер, щоб потрапити на секретну сторінку, потрібно буде знову вводити логін і пароль.

Для того, щоб скинути значення всіх змінних сесії, можна використовувати функцію session_unset();

Знищити поточну сесію цілком можна командою session_destroy(); Вона не скидає значення глобальних змінних сесії і не видаляє cookies, а знищує всі дані, асоційовані з поточною сесією.

```
<?
session_start(); // ініціалізуємо сесію
$test = "змінна сесії";
$_SESSION['test']= $test;
// реєструємо змінну $test.
// якщо register_globals=on, то можна використовувати
// session_register('test');
print_r($_SESSION);
// виводимо всі глобальні змінні
echo session_id();
// виводимо ідентифікатор сесії
echo "<hr>";
session_unset();
// знищуємо всі глобальні змінні сесії
print_r($_SESSION);
echo session_id();
echo "<hr>";
session_destroy(); // знищуємо сесію
print_r($_SESSION);
echo session_id();
?>
```

У результаті роботи цього скрипта будуть виведені три рядки: у першому - масив з елементом test і його значенням, а також ідентифікатор сесії, у другому - порожній масив і ідентифікатор сесії, у третьому - порожній масив. Таким чином, видно, що після знищення сесії знищується і її ідентифікатор, і ми більше не можемо ні реєструвати змінні, ні взагалі робити які-небудь дії із сесією.

Безпека

Варто розуміти, що використання механізму сесій не гарантує повної безпеки системи. Для цього потрібно вживати додаткові заходи. Звернемо увагу на проблеми з безпекою, що можуть виникнути при роботі із сесіями і, зокрема, з тими програмами, що ми написали.

По-перше, небезпечно передавати сюди пароль, його можуть перехопити. Крім того, ми зареєстрували його як глобальну змінну сесії, значить, він зберігся в cookies на комп'ютері-клієнті. Це теж погано. І взагалі, паролі і логіни краще повинні зберігатися в базі даних. Нехай інформація про користувачів зберігається в базі даних "test" (у таблиці "users"), а ми маємо до неї доступ під логіном my_user і паролем my_passwd.

По-друге, що робити, якщо хтось написав скрипт підбору пароля для секретної сторінки? У цьому випадку на сторінку авторизації багато разів повинен стукатися якийсь сторонній скрипт. Тому потрібно просто перевіряти, чи з нашого сайту прийшов запит на авторизацію, і якщо ні, то не пускати його далі. Адреса сторінки, з якої надійшов запит, можна одержати за допомогою глобальної змінної $_SERVER['HTTP_REFERER']). Хоча, звичайно, якщо за ламання сайта взялися серйозно, то значення цієї змінної теж підмінять (наприклад, за допомогою того ж PHP). Проте перевірку її значення можна вважати одним з найважливіших кроків на шляху до забезпечення безпеки свого сайта.

Начебто б перші дві проблеми розв'язані. Але є ще одна. Що робити, якщо хакер просто допише в рядок запиту значення якої-небудь глобальної змінної (наприклад, логіна)? Взагалі це можливо, тільки якщо register_globals=On. Просто інакше ми використовуємо для роботи з глобальними змінними масив $_SESSION і з ним такі фокуси не проходять. Усе-таки спробуємо розв'язати і цю проблему. Для цього потрібно очистити рядок запиту перед тим, як порівнювати значення параметрів. Тобто спочатку скинемо значення $user_login. Потім дану змінну потрібно знову зареєструвати, але не як нову, а як вже існуючу. Для цього знак долара при реєстрації НЕ опускається. От що вийшло:

```
<?php
unset($user_login); // знищуємо змінну
session_start(); // створюємо нову сесію або відновлюємо поточну
session_register($user_login);
   // реєструємо змінну як вже існуючу
if (!($user_login=="pit")) // перевіряємо логін
   Header("Location: authorize.php");
   // якщо помилка, то перенаправляємо на сторінку авторизації
?>
<html>
<head><title>Secret info</title></head>
... // тут розташовується секретна інформація :)
</html>
```

Отже, ми познайомилися із сесіями й основними способами роботи з ними, проблемами, що виникають при їхньому використанні, і можливими розв'язками цих проблем. Сподіваюся, що після прочитання лекції слухачам стало зрозуміло, наскільки зручні і прості у використанні сесії, а наведені приклади знадобляться на практиці.

Питання для самоконтролю:

1. Що таке сесія?
2. Що таке авторизація доступу?
3. Як можна реєструвати змінні сесії?
4. Які методи Ви знаєте для знищення змінних сесій?
5. Які знаєте способи передачі ідентифікатора сесії?

Тема. Взаємодія PHP із MySGL

План

1. Встановлення з'єднання з БД. Вибір бази даних.
2. Одержання списку полів таблиці.
3. Запис даних у базу даних.
4. Відображення даних, що зберігаються в MySQL.

Ця частина лекції познайомить зі способами взаємодії PHP і СУБД MySql. Основна увага тут приділяється встановленню з'єднання з базою даних, функціям відправлення запитів і обробці відповідей (mysql_connect,

mysql_query, mysql_result, mysql_num_rows, mysql_close). Приклад - створення web -інтерфейсу для адміністрування бази даних віртуального музею історії.

У дистрибутив PHP входить розширення, що містить вбудовані функції для роботи з базою даних MySQL. У цій частині лекції ми познайомимося з деякими основними функціями для роботи з MySQL, що будуть потрібні для розв’язання задач побудови web-інтерфейсів з метою відображення і наповнення бази даних. Виникає питання: навіщо будувати такі інтерфейси? Для того щоб вносити інформацію в базу даних і переглядати її вміст могли люди, не знайомі з мовою запитів SQL. При роботі з web-інтерфейсом для додавання інформації в базу даних людині потрібно просто ввести ці дані в html-форму і відправити їх на сервер, а наш скрипт зробить все інше. А для перегляду вмісту таблиць досить просто клацнути по посиланню і зайти на потрібну сторінку.

Для наочності будемо будувати ці інтерфейси для таблиці Artifacts, у якій міститься інформація про експонати віртуального музею інформатики. У попередній частині лекції ми вже наводили структуру цієї колекції, а також її зв'язки з колекціями опису персон (Persons) і зображень (Images). Нагадаємо, що кожен експонат у колекції Artifacts описується за допомогою наступних характеристик:

- назва (title);
- автор (author);
- опис (description);
- альтернативна назва (alternative);
- зображення (photo).

Назва й альтернативна назва є рядками менш ніж 255 символів довжиною (тобто мають тип VARCHAR(255)), опис - текстове поле (має тип TEXT), а в полях "автор" і "зображення" містяться ідентифікатори автора з колекції Persons і зображення експоната з колекції Images відповідно.

Отже, у нас є якась таблиця в базі даних. Щоб побудувати інтерфейс для додавання інформації в цю таблицю, потрібно її структуру (тобто набір її полів) відобразити в html-форму.

Розіб'ємо цю задачу на наступні підзадачі:

- установка з'єднання з БД;
- вибір робочої БД;
- одержання списку полів таблиці;
- відображення полів у html-форму.

Після цього дані, введені у форму, потрібно записати в базу даних. Розглянемо всі ці задачі одну за одною.

1. Встановлення з'єднання з БД

Отже, перше, що потрібно зробити, - це встановити з'єднання з базою даних. Скористаємося функцією mysql_connect.

Синтаксис mysql_connect

mysql_connect ([рядок server [, рядок username [, рядок password [, логічне new_link [, ціле client_flags]]]]])

Дана функція встановлює з'єднання із сервером MySQL і повертає вказівник на це з'єднання або FALSE у випадку невдачі. Для відсутніх параметрів встановлюються наступні значення за замовчуванням:

server = 'localhost:3306'
username = ім'я користувача власника процесу сервера
password = порожній пароль

Якщо функція викликається двічі з тими самими параметрами, то нове з'єднання не встановлюється, а повертається посилання на старе з'єднання. Щоб цього уникнути, використовують параметр new_link, що змушує в будь-якому випадку відкрити ще одне з'єднання.

Параметр client_flags - це комбінація наступних констант: MYSQL_CLIENT_COMPRESS (використовувати протокол стиску), MYSQL_CLIENT_IGNORE_SPACE (дозволяє вставляти пробіли після імен функцій), MYSQL_CLIENT_INTERACTIVE (чекати interactive_timeout секунд - замість wait_timeout - до закриття з'єднання).

Параметр new_link з'явився в PHP 4.2.0, а параметр client_flags - у PHP 4.3.0.

З'єднання із сервером закривається при завершенні виконання скрипта, якщо воно до цього не було закрито за допомогою функції mysql_close().

Отже, встановлюємо з'єднання з базою даних на локальному сервері для користувача nina з паролем "123":

```
<?
$conn = mysql_connect("localhost", "nina","123")
or die("Неможливо встановити з'єднання: ". mysql_error());
echo "З'єднання встановлене";
mysql_close($conn);
?>
```

Дія mysql_connect рівносильна команді shell>mysql -u nina -p123

Вибір бази даних

Після встановлення з'єднання потрібно вибрати базу даних, з якою будемо працювати. Наші дані зберігаються в базі даних book. У MySQL вибір бази даних здійснюється за допомогою команди use:

```
mysql>use book;
```

У PHP для цього існує функція mysql_select_db.

Синтаксис mysql_select_db:

логічне mysql_select_db (рядок database_name [, ресурс link_identifier])

Ця функція повертає TRUE у випадку успішного вибору бази даних і FALSE - у протилежному випадку.

Зробимо базу даних book робочою:

```
<?
$conn = mysql_connect("localhost","nina","123")
or die("Неможливо установити з'єднання: ". mysql_error());
echo "З'єднання встановлене";
```

```
mysql_select_db("book");
?>
```

2. Одержання списку полів таблиці

Тепер можна зайнятися власне розв'язанням задачі. Як одержати список полів таблиці? Дуже просто, у PHP і на цей випадок є своя команда - mysql_list_fields.

Синтаксис mysql_list_fields

ресурс mysql_list_fields (рядок database_name,рядок table_name[, ресурс link_identifier])

Ця функція повертає список полів у таблиці table_name у базі даних database_name. Отже, вибирати базу данних нам було необов'язково, але це знадобиться пізніше. Як можна помітити, результат роботи цієї функції - змінна типу ресурс. Тобто це не зовсім те, що ми хотіли отримати. Це посилання, яке можна використовувати для одержання інформації про поля таблиці, включаючи їх назви, типи і прапори.

Функція mysql_field_name повертає ім'я поля, отриманого в результаті виконання запиту. Функція mysql_field_len повертає довжину поля. Функція mysql_field_type повертає тип поля, а функція mysql_field_flags повертає список прапорів поля, записаних через пробіл. Типи полів можуть бути int, real, string, blob і т.д. Прапори можуть бути not_null, primary_key, unique_key, blob, auto_increment і т.д.

Синтаксис у всіх цих команд однаковий:

```
рядок mysql_field_name (ресурс result, ціле field_offset)
рядок mysql_field_type (ресурс result, ціле field_offset)
рядок mysql_field_flags (ресурс result, ціле field_offset)
рядок mysql_field_len (ресурс result, ціле field_offset)
```

Тут result - це ідентифікатор результату запиту (наприклад, запиту, відправленого функціями mysql_list_fields або mysql_query (про неї буде розказано пізніше)), а field_offset - порядковий номер поля в результаті.

Взагалі кажучи, те, що повертають функції типу mysql_list_fields або mysql_query, являє собою таблицю, а точніше, вказівник на неї. Щоб одержати з цієї таблиці конкретні значення, потрібно задіяти спеціальні функції, що пострічково читають цю таблицю. До таких функцій і належать mysql_field_name і т.п. Щоб перебрати всі рядки в таблиці результату виконання запиту, потрібно знати кількість рядків у цій таблиці. Команда mysql_num_rows(ресурс result) повертає кількість рядків у безліч результатів result.

А тепер спробуємо одержати список полів таблиці Artifacts (колекція експонатів).

```
<?
$conn = mysql_connect("localhost","nina","123")
or die("Неможливо встановити з'єднання: ". mysql_error());
echo "З'єднання встановлене";
mysql_select_db("book");
$list_f = mysql_list_fields ("book","Artifacts",$conn);
$n = mysql_num_fields($list_f);
```

```
for($i=0;$i<$n; $i++){
  $type = mysql_field_type($list_f, $i);
  $name_f = mysql_field_name($list_f,$i);
  $len = mysql_field_len($list_f, $i);
  $flags_str = mysql_field_flags ($list_f, $i);
echo "<br>Ім'я поля: ". $name_f;
echo "<br>Тип поля: ". $type;
echo "<br>Довжина поля: ". $len;
echo "<br>Рядок прапорів поля: ". $flags_str . "<hr>";
}
?>
```

У результаті повинно вийти приблизно от що (якщо в таблиці всього два поля, звичайно):

```
Ім'я поля: id
Тип поля: int
Довжина поля: 11
Рядок прапорів поля: not_null primary_key auto_increment
Ім'я поля: title
Тип поля: string
Довжина поля: 255
Рядок прапорів поля:
```

Відображення списку полів у html-формі

Тепер дещо підкоректуємо попередній приклад. Будемо не просто виводити інформацію про поле, а відображати його як потрібний елемент html-форми. Так, елементи типу BLOB переведемо в textarea (помітимо, що поле description, що ми створювали з типом TEXT, відображається як таке, що має тип BLOB), числа і рядки відобразимо як текстові рядки введення <input type=text>, а елемент, що має мітку автоінкремента, взагалі не будемо відображати, оскільки його значення встановлюється автоматично. Усе це розв'язується досить просто, за винятком виділення зі списку прапорів прапора auto_increment. Для цього потрібно скористатися функцією explode.

Синтаксис explode:

масив explode(рядок separator, рядок string [, int limit])

Ця функція розбиває рядок string на частини за допомогою роздільника separator і повертає масив отриманих рядків.

У нашому випадку як роздільник потрібно взяти пробіл " ", а як задану стрічку для розбивки - рядок прапорів поля.

Отже, створимо форму для введення даних у таблицю Artifacts:

```
<?
$conn=mysql_connect("localhost","nina","123");  // встановлюємо з'єднання
$database = "book";
$table_name = "Artifacts";
mysql_select_db($database); // обираємо базу даних для роботи
$list_f = mysql_list_fields($database,$table_name); // отримуємо список полів у базі
$n = mysql_num_fields($list_f);
                    // кількість стрічок в результаті попереднього запиту
                    //тобто скільки всього полів в таблиці Artifacts)
```

```
echo "<form method=post action=insert.php>";
                    // створюємо форму для введення даних
echo "<TABLE BORDER=0 CELLSPACING=0 width=50%><tr>
   <TD  BGCOLOR='#005533' align=center>
    <font color='#FFFFFF'>
      <b> Add new row in $table_name</b>
   </font></td></tr>
   <tr><td></td></tr></TABLE>";
echo "<table border=0 CELLSPACING=1 cellpadding=0 width=50% >";
// для кожного поля отримуємо його ім'я, тип, довжину і прапори
for($i=0;$i<$n; $i++){
   $type = mysql_field_type($list_f, $i);
   $name_f = mysql_field_name ($list_f,$i);
   $len = mysql_field_len($list_f, $i);
   $flags_str = mysql_field_flags ($list_f, $i);
// із стрічки прапорів робимо масив, де кожен елемент масиву – прапор поля,
   $flags = explode(" ", $flags_str);
   foreach ($flags as $f){
      if ($f == 'auto_increment') $key = $name_f;
         // запам'ятовуємо ім'я інкремента
   }
/* для кожного поля, що не автоінкрементом, в
залежності від його типу виводимо потрібний елемент форми */
if ($key <> $name_f){
echo "<tr><td align=right bgcolor='#C2E3B6'>
     <font size=2><b> ". $name_f ."</b></font>
     </td>";
switch ($type){
       case "string":
           $w = $len/5;
           echo "<td><input type=text name=\"$name_f\" size = $w ></td>";
       break;
       case "int":
          $w =  $len/4;
          echo "<td><input type=text name=\"$name_f\" size = $w ></td>";
       break;
       case "blob":
        echo "<td><textarea rows=6 cols=60 name=\"$name_f\"></textarea></td>";
       break;
       }
   }
   echo "</tr>";
echo "</table>";
echo "<input type=submit name='add' value='Add'>";
echo "</form>";?>
```

3. Запис даних у базу даних

Отже, форма створена. Тепер потрібно зробити найголовніше - відправити дані з цієї форми в нашу базу даних. Як ви вже знаєте, для того щоб записати дані в таблицю, використовується команда INSERT мови SQL. Наприклад:

```
mysql> INSERT INTO Artifacts SET title='Петров';
```

Виникає питання, як можна скористатися такою командою (або будь-якою іншою командою SQL) у PHP скрипті. Для цього існує функція mysql_query().

Синтаксис mysql_query

ресурс mysql_query (рядок query [, ресурс link_identifier])

mysql_query() посилає SQL-запит активній базі даних MySQL сервера, що визначається за допомогою вказівника link_identifier (це посилання на якесь з'єднання із сервером MySQL). Якщо параметр link_identifier опущений, використовується останнє відкрите з'єднання. Якщо відкриті з'єднання відсутні, функція намагається з'єднатися із СУБД, аналогічно функції mysql_connect() без параметрів. Результат запиту буферизується.

Зауваження: рядок запиту НЕ повинен закінчуватися крапкою з комою.

Тільки для запитів SELECT, SHOW, EXPLAIN, DESCRIBE, mysql_query() повертає вказівник на результат запиту, або FALSE, якщо запит не був виконаний. В інших випадках mysql_query() повертає TRUE, якщо запит виконаний успішно, і FALSE - у випадку помилки. Значення, не рівне FALSE, свідчать про те, що запит був виконаний успішно. Воно не говорить про кількість порушених або повернутих рядів. Цілком можлива ситуація, коли успішний запит не торкнеться жодного ряду. mysql_query() також вважається помилковим і поверне FALSE, якщо в користувача недостатньо прав для роботи з зазначеною в запиті таблицею.

Отже, тепер ми знаємо, як відправити запит на вставку рядків у базу даних. Помітимо, що в попередньому прикладі елементи форми ми назвали іменами полів таблиці. Тому вони будуть доступні в скрипті insert.php, що обробляє дані форми, як змінні виду:

```
$_POST['ім'я_поля'].
<?
$conn=mysql_connect("localhost","nina","123");// встановлюємо з’єднання
$database = "book";
$table_name = "Artifacts";
mysql_select_db($database); // обираємо базу даних
$list_f = mysql_list_fields($database,$table_name);
          // отримуємо список полів в базі
$n = mysql_num_fields($list_f);
            // кількість стрічок в результаті попереднього запиту
            // створимо один запит відразу для всіх полів таблиці
$sql = "INSERT INTO $table_name SET ";
   // починаємо створювати запит, перебираємо всі поля таблиці
for($i=0;$i<$n; $i++){
   $name_f = mysql_field_name ($list_f,$i); // визначаємо ім’я поля
   $value = $_POST[$name_f]; // обчислюємо значення поля
   $j = $i + 1;
   $sql = $sql . $name_f." = '$value'";
              // дописуємо в стрічку $sql пару ім’я=значення
   if ($j <> $n) $sql = $sql . ", ";
             // якщо поле не останнє в списку, то ставимо кому
}
// перед тим, як записувати що небуть у базу можна подивитися,
//який запит отримаєтьмя
```

```
//echo $sql;
$result = mysql_query($sql,$conn); // відправляємо запит
// виводимо повідомлення чи успішно виконано запит
if (!result) echo " Can't add ($table_name) ";
   else echo "Success!<br>";
?>
```

Отже, задачу додавання даних за допомогою web-інтерфейсу ми розв'язали. Однак тут є один нюанс. При розв'язанні ми не враховували той факт, що значення деяких полів (author, photo) повинні братися з інших таблиць (Persons, Images). Оскільки MySQL із зовнішніми ключами не працює, цей момент залишається на совісті розроблювачів системи. Потрібно дописати програму таким чином, щоб була можливість вводити в такі поля правильні значення. Але ми робити цього не будемо, оскільки завдання лекції полягає в тому, щоб познайомити читача з елементами технології, а не в тому, щоб створити працюючу систему. Тепер звернемося до іншої задачі - відображення даних, що зберігаються в базі даних СУБД MySQL.

4. Відображення даних, що зберігаються в MySQL

Щоб відобразити якісь дані, в браузер за допомогою PHP, потрібно спочатку одержати ці дані у вигляді змінних PHP. При роботі з MySQL без посередника (такого, як PHP) вибірка даних робиться за допомогою команди SELECT мови SQL:

```
mysql> SELECT * FROM Artifacts;
```

У попередньому розділі ми говорили, що будь-який запит, у тому числі і на вибірку, можна відправити на сервер за допомогою функції mysql_query(); Там у нас було дещо інше завдання - одержати дані з форми і відправити їх за допомогою запиту на вставку в базу даних. Результатом роботи mysql_query() там міг бути лище один з виразів, TRUE або FALSE. Тепер же потрібно відправити запит на вибірку всіх полів, а результат відобразити в браузері. І тут результат - це ціла таблиця значень, а точніше, вказівник на цю таблицю. Так що потрібні якісь аналоги функції mysql_field_name(), тільки щоб вони витягували з результату запиту не ім'я, а значення поля. Таких функцій у PHP декілька. Найбільш популярні - mysql_result() і mysql_fetch_array().

Синтаксис mysql_result

```
змішане mysql_result (ресурс result, ціле row [, змішане field])
```

mysql_result() повертає значення однієї комірки результату запиту. Аргумент field може бути порядковим номером поля, в результаті, ім'ям поля або ім'ям поля з ім'ям таблиці через крапку tablename.fieldname. Якщо для імені поля в запиті застосовувався аліас ('select foo as bar from...'), то використовуйте його замість реального імені поля.

Працюючи з великими результатами запитів, варто задіяти одну з функцій, що обробляє відразу цілий ряд результатів (наприклад, mysql_fetch_row(), mysql_fetch_array() і т.д.). Тому що ці функції повертають значення декількох комірок відразу, вони НАБАГАТО швидші mysql_result(). Крім того, потрібно врахувати, що вказівка чисельного зсуву (номера поля)

працює набагато швидше, ніж вказівка стовпчика або стовпчиків і таблиці через крапку.

Виклики функції mysql_result() не повинні змішуватися з іншими функціями, що працюють з результатом запиту.

Синтаксис mysql_fetch_array

масив mysql_fetch_array (ресурс result [, ціле result_type])

Ця функція обробляє ряд результатів запиту, повертаючи масив (асоціативний, чисельний або обидва) з обробленим рядом результатів запиту, або FALSE, якщо рядів більше немає.

mysql_fetch_array() - це розширена версія функції mysql_fetch_row(). Крім збереження значень у масиві з чисельними індексами, функція повертає значення в масиві з індексами за назвою стовпчиків.

Якщо декілька колонок у результаті будуть мати однакові назви, буде повернутий останній стовпчик. Щоб одержати доступ до перших, варто використовувати чисельні індекси масиву або аліаси в запиті. У випадку аліасів саме їх ви не зможете використовувати дійсні імена стовпчиків, як, скажімо, не зможете використовувати "photo" в описаному нижче прикладі.

```
select Artifacts.photo as art_image, Persons.photo as pers_image from Artifacts, Persons
```

Важливо зауважити, що mysql_fetch_array() працює НЕ повільніше, ніж mysql_fetch_row(), і надає більш зручний доступ до даних.

Другий опційний аргумент result_type у функції mysql_fetch_array() є константою і може приймати наступні значення: MYSQL_ASSOC, MYSQL_NUM і MYSQL_BOTH. Ця можливість додана в PHP 3.0.7. Значенням за замовчуванням є: MYSQL_BOTH.

Використовуючи MYSQL_BOTH, одержимо масив, що складається як з асоціативних індексів, так і з чисельних. MYSQL_ASSOC, він поверне тільки асоціативні відповідності, а MYSQL_NUM - тільки чисельні.

Зауваження: імена полів, що повертаються цією функцією, регістронезалежні.

У цій лекції ми розглянули дві задачі: додавання даних у базу даних і їхнє відображення в браузері за допомогою мови PHP. Для цього ми розглянули ряд функцій, що дозволяють відправляти SQL-запити до бази даних і обробляти отримані відповіді. Використовуючи наведену тут технологію, можна розв’язати цілий ряд подібних задач, таких як задачі зміни і знищення даних, задачі маніпулювання таблицями бази даних (тобто їхнє створення, зміна та знищення) і т.п. Усе це типові задачі, що виникають при розробці систем керування даними, і вміння їх розв’язувати, як і вміння працювати з базами даних у цілому, дуже важливі для web-програміста.

Питання для самоконтролю:

1. Яким чином відбувається з’єднання з базою даних?
2. Як відбувається вибір бази даних?
3. Як одержати список полів таблиці?
4. Як записати дані у базу даних?

5. Запишіть синтаксис функції mysql_fetch_array.

Тема. СУБД MySQL. Основні оператори мови SQL

План

1. Створення, знищення, зміна бази даних (БД):
- Оператор CREATE TABLE
- Оператор DROP TABLE
- Оператор ALTER TABLE

2. Внесення, знищення і зміна даних у БД (таблиць і записів):
- Оператор INSERT
- Оператор UPDATE
- Оператор DELETE

3. Вибірка даних із БД.

MySQL – це реляційна система керування базами даних. Тобто дані в її базах зберігаються у вигляді логічно зв'язаних між собою таблиць, доступ до яких здійснюється за допомогою мови запитів SQL. MySQL – вільно поширена система, тобто платити за її використання не потрібно. Крім того, це досить швидка, надійна і, головне, проста у використанні СУБД, що цілком підходить для не надто глобальних проектів.

Працювати з MySQL можна не лише в текстовому режимі, але й у графічному. Існує дуже популярний візуальний інтерфейс (до речі, написаний на PHP) для роботи з цієї СУБД. Називається він PhpMyAdmin. Цей інтерфейс дозволяє значно спростити роботу з базами даних у MySQL.

1. Створення, знищення, зміна бази даних (БД)

До перших двох функцій мають доступ лише адміністратори СУБД або привілейовані користувачі. Розглянемо, як розв’язуються останні дві задачі (насправді це сім задач).

Перш ніж що-небудь робити з даними, потрібно створити таблиці, у яких ці дані будуть зберігатися, навчитися змінювати структуру цих таблиць і їх знищувати, якщо буде потрібно. Для цього в мові SQL існують оператори CREATE TABLE, ALTER TABLE і DROP TABLE.

Оператор CREATE TABLE

Оператор CREATE TABLE створює таблицю із заданим ім'ям у поточній базі даних. Правила для допустимих імен таблиці наведені в документації. Якщо немає активної поточної бази даних або зазначена таблиця вже існує, то виникає помилку виконанні команди.

У версії MySQL 3.22 і більш пізніх ім'я таблиці може бути зазначене як ім'я_бази_даних.ім'я_таблиці. Ця форма запису працює незалежно від того, чи є зазначена база даних поточною.

У версії MySQL 3.23 при створенні таблиці можна використовувати ключове слово TEMPORARY. Тимчасова таблиця автоматично знищується по завершенні з'єднання, а її ім'я дійсне лише протягом даного з'єднання. Це

означає, що в двох різних з'єднаннях можуть використовуватися тимчасові таблиці з однаковими іменами не конфліктуючи одна з одною або з існуючою таблицею з тим же ім'ям (існуюча таблиця схована, поки не знищена тимчасова таблиця). У версії MySQL 4.0.2 для створення тимчасових таблиць необхідно мати привілею CREATE TEMPORARY TABLES.

У версії MySQL 3.23 і більш пізніх можна використовувати ключові слова IF NOT EXISTS для того, щоб не виникала помилка, якщо зазначена таблиця вже існує. Варто враховувати, що ідентичність структур цих таблиць не перевіряється.

Кожна таблиця представлена набором визначених файлів у директорії бази даних. Синтаксис

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] ім'я_таблиці [(визначення_стовпця,...)]
[опції_таблиці] [select_вираз]

У виразі визначення_стовпця перелічують, які стовпці повинні бути створені в таблиці. Кожен стовпець таблиці може бути порожнім (NULL), мати значення за замовчуванням, бути ключовим або автоінкрементом. Крім того, для кожного стовпця обов'язково вказується тип даних, що будуть у ньому зберігатися. Якщо не вказується ні NULL, ні NOT NULL, то стовпець інтерпретується так, начебто зазначено NULL. Якщо поле позначають як автоінкремент (AUTO_INCREMENT), то його значення автоматично збільшується на одиницю щоразу, коли відбувається додавання даних у таблицю й у це поле записується порожнє значення (NULL, тобто нічого не записується) або 0. Автоінкремент у таблиці може бути тільки один, і при цьому він обов'язково повинен бути проіндексований. Послідовність AUTO_INCREMENT починається з 1. Наявність автоінкремента є однією з особливостей MySQL. Формальний опис стовпця (визначення_стовпця) виглядає так:

ім'я_стовпця тип [NOT NULL | NULL]|DEFAULT значення_по_замовчуванні]
[AUTO_INCREMENT][PRIMARY KEY][reference_definition]

Тип стовпця (тип у виразі визначення_стовпця) може бути одним з наступних:

- цілий: INT[(length)] [UNSIGNED] [ZEROFILL]
- дійсний: REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
- символьний: CHAR(length) [BINARY] і VARCHAR(length) [BINARY]
- дата і час: DATE і TIME
- для роботи з великими об'єктами: BLOB
- текстовий: TEXT
- перераховний: ENUM(value1,value2,value3,...) і SET(value1,value2,value3,...)

Повний список типів дивіться в документації MySQL.

Крім всього перерахованого, при створенні таблиці можна вказати деякі її властивості (опції_таблиці), наприклад такі:

- тип таблиці: TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM }
- початкове значення лічильника автоінкремента: AUTO_INCREMENT = число
- середня довжина рядків у таблиці: AVG_ROW_LENGTH = число
- коментарі до таблиці (рядок з 60 символів): COMMENT = "рядок"
- максимальна і мінімальна передбачувана кількість рядків: MAX_ROWS = число і MIN_ROWS = число

І останній опційний елемент команди CREATE – це вираз SELECT (select_вираз).

Синтаксис такий:

[IGNORE | REPLACE] SELECT ... (будь-який коректний вираз SELECT)

Якщо при створенні таблиці в команді CREATE вказується вираз SELECT, то всі поля, отримані вибіркою, додаються в створювану таблицю.

Приклад. Створимо таблицю Persons

```
mysql>CREATE TABLE Persons (
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name  VARCHAR(50),
    last_name VARCHAR(100),
    death_date INT,
    description TEXT,
    photo INT,
    citienship CHAR(50) DEFAULT 'Russia'
    );
```

За допомогою специфічної для MySql команди SHOW можна переглянути існуючі бази даних, таблиці в базі даних і поля в таблиці.

Показати всі бази даних:

```
mysql>SHOW databases;
```

Зробити поточною базу даних book і показати всі таблиці в ній:

```
mysql>use book;
mysql>show tables;
```

Показати всі стовпці в таблиці Persons:

```
mysql> show columns from Persons;
```

Оператор DROP TABLE

Оператор DROP TABLE знищує одну або декілька таблиць. Усі табличні дані і значення знищуються, так що при роботі з цією командою слід дотримуватися обережності. Синтаксис:

DROP TABLE [IF EXISTS] ім'я_таблиці [, ім'я_таблиці,...] [RESTRICT | CASCADE]

У версії MySQL 3.22 і більш пізніх можна використовувати ключові слова IF EXISTS, щоб запобігти помилці, якщо зазначені таблиці не існують.

Опції RESTRICT і CASCADE дозволяють спростити перенесення програми з інших СУБД.

mysql> DROP TABLE IF EXISTS Persons,Artifacts, test;

Оператор ALTER TABLE

Оператор ALTER TABLE забезпечує можливість змінювати структуру існуючої таблиці. Наприклад, можна додавати або знищувати стовпці, створювати або знищувати індекси або переіменовувати стовпці чи саму таблицю. Можна також змінювати коментар для таблиці і її тип. Синтаксис:

ALTER [IGNORE] TABLE ім'я_таблиці alter_specification [,alter_specification...]

Можна робити наступні зміни в таблиці (усі вони записуються в alter_specification):

- додавання поля:
  ADD [COLUMN] визначення_стовпця [FIRST | AFTER ім'я_стовпця]
  або
  ADD [COLUMN] (визначення_стовпця, визначення_стовпця,...)
  Тут, як і далі, визначення_стовпця записується так само, як при створенні таблиці.
- додавання індексів:
  ADD INDEX [ім'я_індексу] (ім'я_індексного_стовпця,...)
  або
  ADD PRIMARY KEY (ім'я_індексного стовпця,...)
  або
  ADD UNIQUE [ім'я_індексу] (ім'я_індексного стовпця,...)
  або
  ADD FULLTEXT [ім'я_індексу] (ім'я_індексного стовпця,...)
- зміна поля:
  ALTER [COLUMN] ім'я_стовпця {SET DEFAULT literal | DROP DEFAULT}
  або
  CHANGE [COLUMN] старе_ім'я_стовпця визначення_стовпця
  або
  MODIFY [COLUMN] визначення_стовпця

- зничення поля, індексу, ключа:
  DROP [COLUMN] ім'я_стовпця
  DROP PRIMARY KEY
  DROP INDEX ім'я_індексу
- перейменування таблиці:
  RENAME [TO] нове_ім'я_таблиці
- переупорядкування полів таблиці:
  ORDER BY поле
  Або
  опції_таблиці

Якщо оператор ALTER TABLE використовується для зміни визначення типу стовпця, але DESCRIBE ім'я_таблиці показує, що стовпець не змінився, то, можливо, MySQL ігнорує дану модифікацію по одній із причин, описаних у спеціальному розділі документації. Наприклад, при спробі змінити

стовпець VARCHAR на CHAR MySQL буде продовжувати використання VARCHAR, якщо дана таблиця містить інші стовпці із змінною довжиною.

Оператор ALTER TABLE під час роботи створює тимчасову копію вихідної таблиці. Необхідна зміна виконується на копії, потім вихідна таблиця знищується, а нова перейменовується. Це робиться для того, щоб у нову таблицю автоматично, потрапляли усі відновлення, крім невдалих. Під час виконання ALTER TABLE вихідна таблиця доступна для читання іншими клієнтами. Операції відновлення і запису в цій таблиці припиняються, поки не буде готова нова таблиця. Слід зазначити, що при використанні будь-якої іншої опції для ALTER TABLE, крім RENAME, MySQL завжди буде створювати тимчасову таблицю, навіть якщо дані, строго кажучи, і не мають потреби в копіюванні (наприклад, при зміні імені стовпця).

Приклад. Додамо в створену таблицю Persons поле для запису року народження людини:

```
mysql> ALTER TABLE Persons ADD bday INTEGER AFTER last_name;
```

Таким чином, ми навчилися працювати з таблицями: створювати, знищувати і змінювати їх. Тепер з’ясуємо, як робити те ж саме з даними, що у цих таблицях зберігаються.

2. Внесення, знишення і зміна даних у БД (таблиць і записів)

Оператор INSERT

Оператор INSERT вставляє нові рядки в існуючу таблицю. Оператор має кілька форм. Параметр ім'я_таблиці у всіх цих формах задає таблицю, в яку повинні бути внесені рядки. Стовпці, для яких задаються значення, вказуються в списку імен стовпців (ім'я_стовпця) або в частині SET.

Синтаксис:

```
INSERT[LOW_PRIORITY|DELAYED][IGNORE][INTO]ім'я_таблиці
        [(ім'я_стовпця,...)] VALUES (вираз,...),(...),...
```

Ця форма команди INSERT вставляє рядки з відповідно точно зазначеними в команді значеннями. У дужках після імені таблиці перелічуються стовпці, а після ключового слова VALUES – їх значення.

Наприклад:

```
mysql> INSERT INTO Persons (last_name, bday) VALUES ('Іванов', '1934');
```

вставить у таблицю Persons рядок, у якому значення прізвища (last_name) і дати народження (bday) будуть задані відповідно як «Іванов» і «1934».

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE][INTO] ім'я_таблиці
        [(ім'я_стовпця,...)] SELECT ...
```

Ця форма команди INSERT вставляє рядки, обрані з іншої таблиці або таблиць.

Наприклад:

```
mysql> INSERT INTO Artifacts (author)SELECT id FROM Persons WHERE
        last_name='Іванов' AND bday='1934';
```

вставить у таблицю Artifacts у поле «автор» (author) значення ідентифікатора, обраного з таблиці Persons за умовою, що прізвище людини Іванов.

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE][INTO] ім'я_таблиці
        SET ім'я_стовпця=вираз, ім'я_стовпця=вираз, ...
```

Наприклад:

```
mysql> INSERT INTO Persons SET last_name='Петров', first_name='Іван';
```

Ця команда вставить у таблицю Persons у поле last_name значення «Петров», а в поле first_name – рядок «Іван».

Форма INSERT ... VALUES зі списком з декількох значень підтримується у версії MySQL 3.22.5 і більш пізніх. Синтаксис виразу ім'я_стовпця=вираження підтримується у версії MySQL 3.22.10 і більш пізніх.

Діють наступні домовленості.

- Якщо не зазначений список стовпців для INSERT ... VALUES або INSERT ... SELECT, то величини для всіх стовпців повинні бути визначені в списку VALUES() або в результаті роботи SELECT. Якщо порядок стовпців у таблиці невідомий, для його одержання можна використовувати DESCRIBE ім'я_таблиці.
- Будь-який стовпець, для якого явно не зазначене значення, буде встановлений у своє значення за замовчуванням. Наприклад, якщо в заданому списку стовпців не зазначені всі стовпці в даній таблиці, то не зазначені стовпці встановлюються у свої значення за замовчуванням.
- Вираз expression може відноситися до будь-якого стовпця, що раніше був внесений у список значень.

Наприклад, можна вказати наступне:

```
mysql> INSERT INTO ім'я_таблиці (col1,col2) VALUES(15,col1*2);
```

Але не можна вказати:

```
mysql> INSERT INTO ім'я_таблиці (col1,col2)VALUES(col2*2,15);
```

Ми ще не прокоментували три необов'язкових параметри, що є присутніми у всіх трьох формах команд: LOW_PRIORITY, DELAYED і IGNORE.

Параметри LOW_PRIORITY і DELAYED використовуються, коли з таблицею працює велика кількість користувачів. Вони пропонують встановлювати пріоритет даної операції перед операціями інших користувачів. Якщо вказується ключове слово LOW_PRIORITY, то виконання даної команди INSERT буде затримано доти, поки інші клієнти не завершать читання цієї таблиці. У цьому випадку клієнт повинен очікувати, поки дана команда вставки не буде завершена, що у випадку інтенсивного використання таблиці може зажадати значного часу. На противагу цьому команда INSERT DELAYED дозволяє даному клієнтові виконати операцію відразу ж, незалежно від інших користувачів.

Якщо в команді INSERT вказується ключове слово IGNORE, то всі рядки, які мають ключі, що дублюються, PRIMARY або UNIQUE у цій таблиці, будуть зігноровані і не внесені в таблицю. Якщо не вказувати IGNORE, то дана операція вставки припиняється при виявленні рядка, що має значення, яке дублюється, для існуючого ключа.

Оператор UPDATE

Синтаксис:

UPDATE [LOW_PRIORITY] [IGNORE] ім'я_таблиці SET ім'я_стовпця1=вираз1
[, ім'я_стовпця2=вираз2, ...][WHERE where_definition][LIMIT число]

Оператор UPDATE обновляє значення існуючих стовпців таблиці відповідно до введених значень. У виразі SET вказується, які саме стовпці варто модифікувати і які величини повинні бути в них встановлені. У виразі WHERE, якщо воно присутнє, задається, які рядки підлягають відновленню. В інших випадках обновляються всі рядки. Якщо задано вираз ORDER BY, то рядки будуть обновлятися в зазначеному в ньому порядку.

Якщо вказується ключове слово LOW_PRIORITY, то виконання даної команди UPDATE затримується доти, поки інші клієнти не завершать читання цієї таблиці.

Якщо вказується ключове слово IGNORE, то команда відновлення не буде перервана, навіть якщо виникне помилка дублювання ключів. Рядки, через які виникають конфліктні ситуації, обновлені не будуть. Якщо у виразі, що задає нове значення стовпця, використовується ім'я цього поля, то команда UPDATE використовує для цього стовпця його поточне значення. Наприклад, наступна команда встановлює стовпець death_date у значення, на одиницю більше його поточної величини:

```
mysql> UPDATE Persons SET death_date=death_date+1;
```

У версії MySQL 3.23 можна використовувати параметр LIMIT #, щоб переконатися, що було змінено лише задану кількість рядків.

Наприклад, така операція замінить у першому рядку нашої таблиці експонатів назву title на рядок «Лампова ЕОМ»:

```
mysql> UPDATE Artifacts SET title='Лампова ЕОМ' Limit 1;
```

Оператор DELETE

Оператор DELETE знищує з таблиці рядки, що задовольняють задані у where_definition умови, і повертають кількість вилучених записів.

Якщо оператор DELETE запускається без визначення WHERE, то знищуються всі рядки.

Синтаксис:

DELTE [LOW_PRIORITY] FROM ім'я_таблиці [WHERE where_definition] [LIMIT rows]

Наприклад, наступна команда знищить з таблиці Persons усі записи, у яких поле «рік народження» (bday) більше 2003:

```
mysql> DELETE FROM Persons WHERE bday>2003;
```

Знищити всі записи в таблиці можна ще і за допомогою такої команди:

```
mysql> DELETE FROM Persons WHERE 1>0;
```

Але цей метод працює набагато повільніше, ніж використання тієї ж команди без умови:

```
mysql> DELETE FROM Persons;
```

Специфічна для MySQL опція LIMIT для команди DELETE вказує серверові максимальну кількість рядків, які варто знищити до повернення

керування клієнтові. Ця опція може використовуватися для гарантії того, що дана команда DELETE не зажадає занадто багато часу для виконання.

Отже, ми розібралися з основами реляційних баз даних, навчилися створювати прості і не дуже SQL-запити. Сподіваюся, що велика кількість технічних деталей не нашкодила слухачам одержати уявлення про базові елементи мови, оскільки все це напевно знадобиться для розв'язання практичних задач.

3. Вибірка даних із БД

Оператор SELECT

Оператор SELECT застосовується для витягнення рядків, обраних з однієї або декількох таблиць. Тобто з його допомогою ми задаємо стовпці або вирази, які треба витягнути (select_вираження), таблиці (table_references), з яких повинна робитися вибірка, і, можливо, умова (where_definition), якій повинні відповідати дані в цих стовпцях, і порядок, у якому ці дані потрібно видати.

Крім цього, оператор SELECT можна використовувати для витягнення рядків, обчислених без посилання на яку-небудь таблицю.

Наприклад, щоб обчислити, чому дорівнює 2*2, потрібно просто написати

```
mysql> SELECT 2*2;
```

Спрощено структуру оператора SELECT можна представити в такий спосіб:

```
SELECT select_вираз1, select_вираз2,...
  [FROM table_references [WHERE where_definition]
    [ORDER BY {число | ім'я_стовпця | формула}
      [ASC | DESC], ...]]
```

Квадратні дужки [ ] означають, що використання оператора, який знаходиться в них, необов'язкове, вертикальна риска | означає перерахування можливих варіантів. Після ключового слова ORDER BY вказують ім'я стовпця, число (ціле беззнакове) або формулу і спосіб впорядкування (по зростанням – ASC, або за спаданням – DESC). За замовчуванням використовується впорядкування по зростанню.

Коли в select_виразі ми пишемо «*», то це значить вибрати всі стовпці. Крім «*» у select_виразах можуть використовуватися функції типу max, min і avg.

Приклад. Вибрати з таблиці Persons усі дані, для яких поле first_name має значення 'Олександр':

```
mysql> SELECT * FROM Persons WHERE first_name='Олександр';
```

Вибрати назву й опис (title, description) артефакту під номером 10:

```
mysql> SELECT title,description FROM Artifacts WHERE id=10;
```

Питання для самоконтролю:

1. Які оператори використовуються для створення бази даних (БД).
2. Які оператори використовуються для знищення бази даних.
3. Які оператори використовуються для зміни бази даних.
4. Як відбувається вибірка даних із БД.

СПИСОК ЛІТЕРАТУРИ

1. Пасічник О. В., Пасічник В. В. Веб-дизайн: Підручник. – Львів: Магнолія 2006, 2017. – 570 с.

2. Пасічник О. В., Пасічник В. В., Угрин Д. І. Веб-технології: Підручник. – Львів: Магнолія 2006, 2015. – 336 с.

3. Web-технології та Web-дизайн [Текст]: частина друга: конспект лекцій для студентів 4-го курсу спеціальності 5.05010101 "Обслуговування програмних систем і комплексів" денної форми навчання / уклад. Л.В.Мелещук – Ковель: КПЕК Луцького НТУ, 2015. – 92 с.

4. Мержевич В. HTML и CSS на прикладах. – СПб., 2004. – 441 с.

5. Дари К., Бринзаре Б., Черчез-Тоза Ф., Бусика М. AJAX и PHP. Разработка динамических веб-приложений. – М.: Солон-Пресс, 2006. – 336 с.

6. Уилсон П. Основы JavaScript. - СПб.: Символ, 2002.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Web-технології та Web-дизайн [Текст]: Конспект лекцій для здобувачів освітньо-кваліфікаційного рівня фаховий молодший бакалавр IV курсу спеціальності комп'ютерні науки денної форми навчання. /уклад. Л.В. Мелещук. – Ковель : ВСП «КФПЕК Луцький НТУ», 2019. – 95.

Комп'ютерний набір і верстка: Л.В.Мелещук

Редактор: Л.В.Мелещук

Підп. до друку _____2019. Формат 60x84/16.Папір офс. Гарн. Таймс.
Ум. друк. арк.3,4.
Обл.-вид. арк. 7,4. Тираж _____прим. Зам.188.

ВСП «Ковельський промислово-економічний фаховий коледж
Луцького національного технічного університету»
45000 м. Ковель, вул. Заводська, 23
Друк – ВСП «КПЕФК Луцького НТУ»