1. КОНСПЕКТ ЛЕКЦІЙ

Лекції 1-3. Основи web-програмування

Тема 1. Огляд програмування на боці клієнта та сервера. Налаштування робочого середовища.

Інтернет та WEB

Інтернет (Internet, скор. від Interconnected Networks – об'єднані мережі) – глобальна телекомунікаційна мережа інформаційних і обчислювальних ресурсів.

Від самого початку архітектура Інтернету мала наступний вигляд.

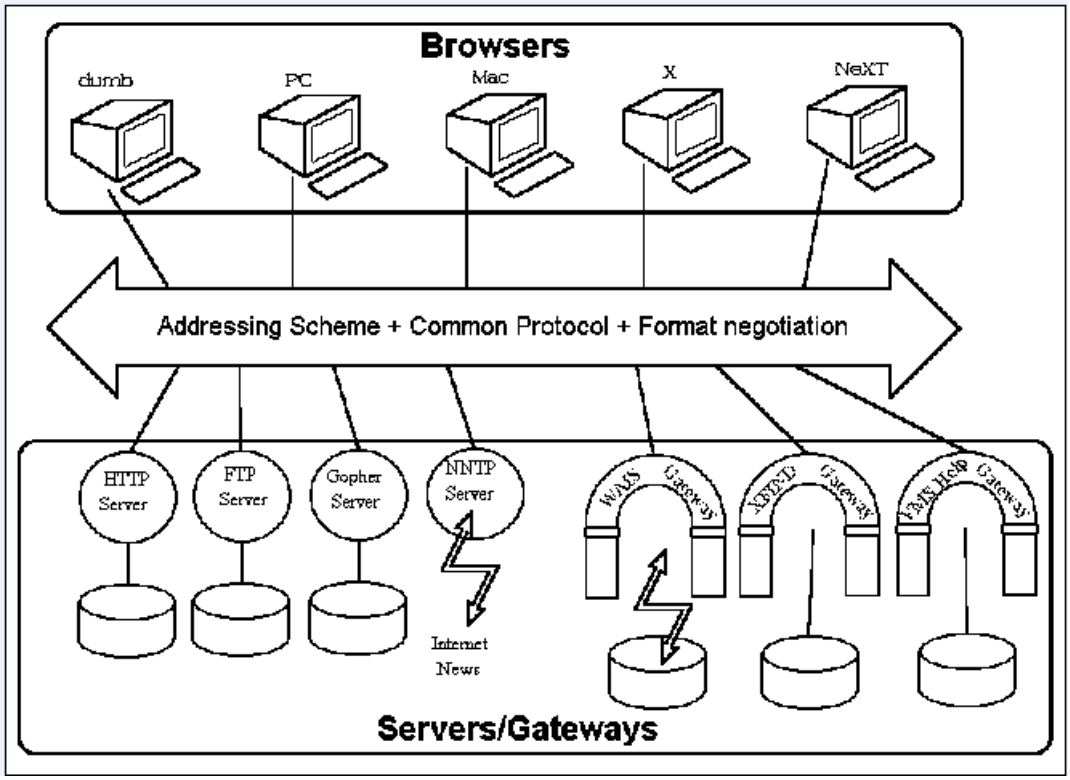

Рис. 3. Базова архітектура мережі Інтернет.

Курс «Програмування Інтернет» може включати всі технології і протоколи, показані на Рис.3. Але в нашому курсі ми більше зосередимося на web-програмуванні.

Web, або Всесвітня мережа (англ. World Wide Web) – розподілена система, що надає доступ до пов'язаних між собою документів, розташованих на різних комп'ютерах, підключених до Інтернету. Основні складові: мова HTML, універсальний спосіб адресації ресурсів у мережі URL, протокол обміну гіпертекстовою інформацією HTTP. Архітектура WWW показана на Рис.4.

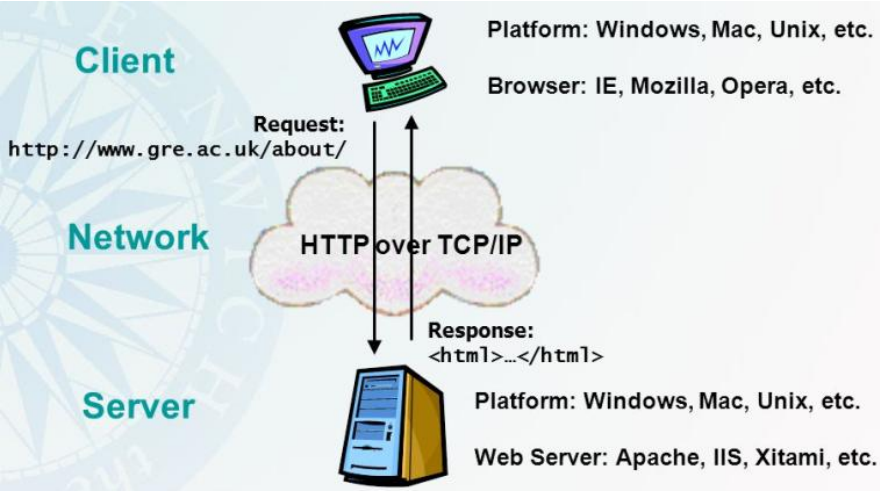

Рис. 4. Архітектура Web

Потік даних та взаємодія протоколів показані на Рис.5.

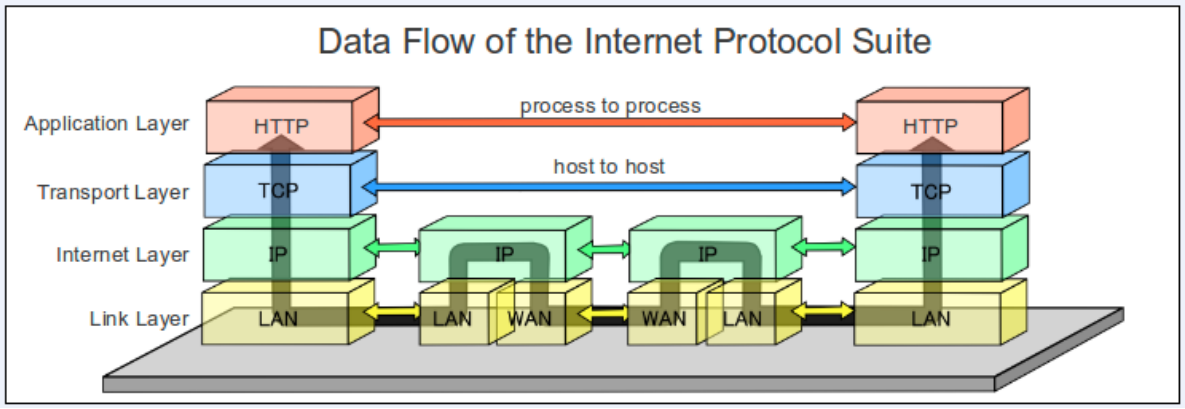

Рис.5. Потік даних та взаємодія протоколів для Web

Web-програмування

Веб-програмування – розділ програмування, орієнтований на розробку динамічних Internet-додатків.

Веб-програмування включає програмування на стороні клієнта (front-end) та на стороні сервера (back-end).

На стороні клієнта в основному використовується мова JavaScript, як показано на Рис.6.

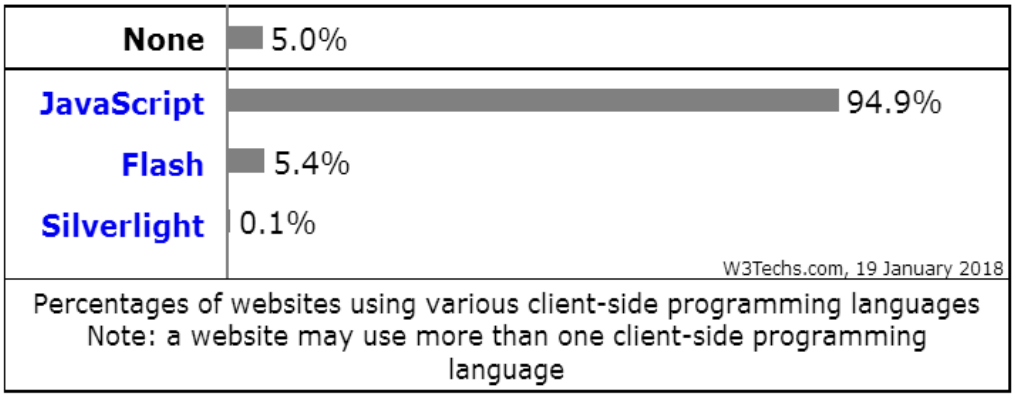

Рис.6. Використання мов програмування на стороні клієнта.

Використання бібліотек JavaScript для сайтів станом на січень 2018 р. показане в Табл.1 (сайти можуть використовувати декілька бібліотек).

Таблиця 1. Використання JavaScript-бібліотек.

Бібліотека	Використання
jQuery	96.2%
Bootstrap	21.8%
Modernizm	15.0%
MooTools	3.5%
ASP.NET Ajax	2.6%
Backbone	0.8%
Lodash	0.7%
React	0.7%
Angular	0.6%
Vue.js	0.1%

Використання різних мов програмування в серверних web-технологіях станом на січень 2018 р. показане на Рис.7 [27].

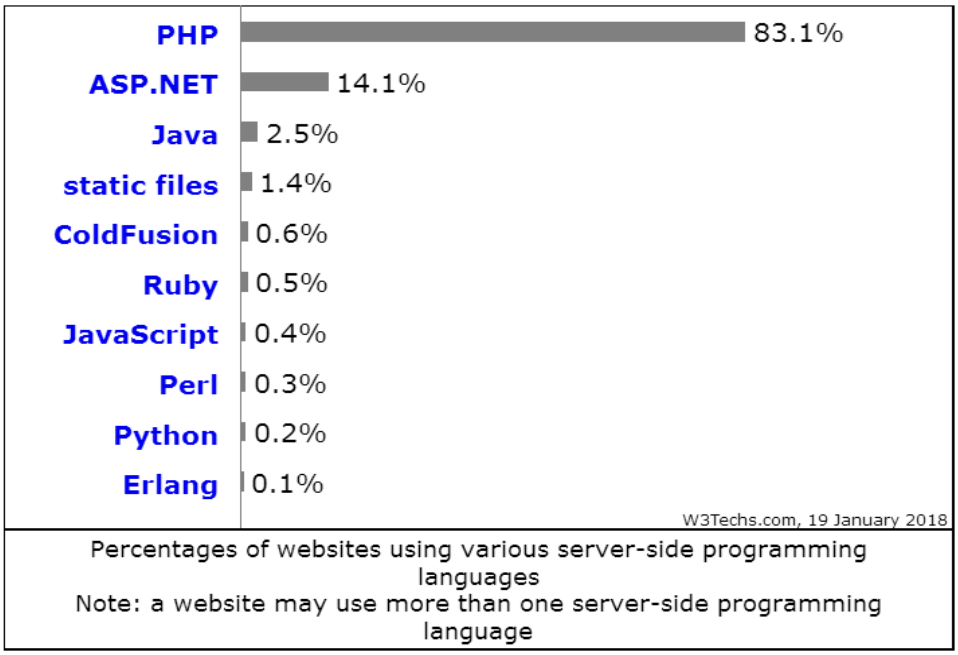

Рис. 7. Мови програмування в серверних web-технологіях

В першій частині курсу ми розглянемо використання мови PHP при розробці серверних веб-застосунків.

Налаштування робочого середовища розробки

Варіант 1. Установка вручну всіх програмних компонентів

- Web-сервер Apache 2.4 (http://httpd.apache.org/download.cgi).
- Система керування базами даних MySQL 5.7 (www.mysql.com).
- Інтерпретатор скриптів PHP 7.2 (www.php.net).

Цей варіант потребує тонкого налаштування всіх взаємодіючих компонентів, і на початковій стадії вивчення технології занадто громіздкий.

Варіант 2. Попередньо упаковані установки

Краще (для початку) використовувати один з готових наборів дистрибутивів (вже налаштованих) – їх називають: WAMP (Windows, Apache, MySQL, PHP), LAMP – для Linux, MAMP – для Mac OS.

- Денвер-3 (Д.н.w.p – джентльменський набір web-розробника, www.denwer.ru) – PHP 5.3.13, MySQL 5.5. Цей набір давно не оновлювався і його використання наразі недоречне.
- XAMPP (X, Apache, MySQL, PHP, Perl) – www.apachefriends.org/ru/ (Includes**:** Apache 2.4.29, MariaDB 10.1.29, PHP 7.2.0, phpMyAdmin 4.7.4, OpenSSL 1.0.2, XAMPP Control Panel 3.2.2, FileZilla FTP Server 0.9.41). Цей набір ми будемо використовувати в нашому курсі. Робочий каталог для сайтів: C:\xampp\htdocs.
- WampServer (http://www.wampserver.com/)
- Open Server (https://ospanel.io/)

Варіант 3. Віртуальні сервери

Віртуальний сервер працює як веб-сервер на іншому комп'ютері. На цьому комп'ютері можна запустити будь-яку операційну систему.

Треба встановити наступне програмне забезпечення [8]:

- Git (https://www.git-scm.com/)
- VirtualBox (https://www.virtualbox.org/)
- Vagrant (https://www.vagrantup.com/)

Цей спосіб використовують професійні веб-розробники, яким треба одночасно підтримувати декілька проектів з різними налаштуваннями.

Варіант 4. Онлайн середовища кодування

- EasyPHP’s Code Classroom (http://www.easyphp.org/) створює онлайн середовище розробки для студентів і викладачів. Включає PHP, Apache, MySQL, Nginx, PhpMyAdmin, Xdebug, PostgreSQL, MongoDB, Python, Ruby for Windows.
- Code Classroom (http://www.codeclassroom.net/) – інтерактивне середовище на допомогу студентам, що вивчають PHP, HTML, CSS, JS, SQL.
- PHP 5 Tutorial (https://www.w3schools.com/php/default.asp)

Редактори PHP та IDE

- Notepad++ (https://notepad-plus-plus.org/)
- Brackets (http://brackets.io/)
- Atom (https://atom.io/)
- Sublime Text (http://www.sublimetext.com/)
- Aptana Studio (http://www.aptana.com/)
- NetBeans IDE (https://netbeans.org/)
- Komodo IDE (https://www.activestate.com/komodo-ide)
- PhpStorm (https://www.jetbrains.com/phpstorm/)

- CodeLobster PHP Edition (http://www.codelobster.com/)
- Zend Studio (http://www.zend.com/en/products/studio)

Перевагу слід надати безкоштовним редакторам і середовищам розробки та тим IDE, для яких можна отримати безкоштовну студентську ліцензію.

Фреймворки PHP

Програмний фреймворк (англ. *software framework*) або каркас застосунку – це готовий до використання комплекс програмних рішень, що включає дизайн, логіку та базову функціональність системи.

Найбільш популярними фреймворками PHP є такі:

- Laravel (https://laravel.com/)
- Symfony (https://symfony.com/)
- Yii (http://www.yiiframework.com/)
- CakePHP (https://cakephp.org/)
- CodeIgniter (https://codeigniter.com/)
- Zend Framework (https://framework.zend.com/)

Програма нашого курсу передбачає розгортання фреймворку Laravel та створення в його середовищі простого сайту.

CMS на базі PHP

Система керування вмістом (СКВ; англ. *Content Management System, CMS*) – це програмне забезпечення організації веб-сайтів. Надає інструменти для додавання, редагування, видалення інформації на сайті (без програмування).

Перерахуємо найбільш популярні CMS на базі PHP.

- WordPress (https://wordpress.org/)
- Joomla! (https://www.joomla.org/)
- Drupal (https://www.drupal.org/)
- Magento (https://magento.com/)
- Bitrix (https://www.bitrix.ua/)

Програма нашого курсу передбачає розгортання системи керування вмістом WordPress та створення в її середовищі простого сайту.

Тема 2. Основи World Wide Web (WWW). Протокол HTTP. Структура HTTP-повідомлення. Основні заголовки. Стандарт MIME.

HTTP (HyperText Transfer Protocol, протокол передачі гіпертексту) – протокол прикладного рівня для передачі даних в першу чергу у вигляді текстових повідомлень. Основою протоколу HTTP є технологія «клієнт-сервер».

Робота за протоколом HTTP відбувається так:

- програма-клієнт установлює TCP-з'єднання із сервером (стандартний номер порту – 80, для https – 443) і видає йому HTTP-запит.

- Сервер обробляє цей запит і
- видає HTTP-відповідь клієнтові.

Структура HTTP-повідомлення

Незалежно від виду повідомлення (запит клієнта чи відповідь сервера) кожне HTTP-повідомлення має один і той же формат, що складається з трьох розділів:

- Стартовий рядок запиту/відповіді.
- HTTP-заголовок.
- HTTP-тіло.

Вміст цих частин залежить від того, чи є повідомлення запитом чи відповіддю.

HTTP-заголовок відділяється від HTTP-тіла порожнім рядком.

HTTP-запит (браузер – web-серверу)

- Стартовий рядок запиту.
- Заголовок запиту.
- Тіло запиту.

Приклад (рядок запиту та заголовок запиту).

```
GET /wiki/HTTP HTTP/1.1
Host: uk.wikipedia.org
User-Agent: firefox/5.0 (Linux; Debian 5.0.8; en-US;
rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7
Connection: close
```

Відповідь:

```
HTTP/1.0 200 OK
Server: Apache
Content-Language: uk
Content-Type: text/html; charset=utf-8
Content-Length: 1234
```

(далі йде текст запитаної сторінки)

Рядок запиту (request line)

Має наступний формат:

Метод_запиту URL_ресурсу Версія_протоколу_HTTP

- Метод_запиту - це HTTP-команда, яку називають метод (частіше за все GET або POST).
- URL_ресурсу - це шлях від сервера до ресурсу, який запитує клієнт.

Метод вказує серверу, як обробляти дані (GET – дані в заголовці, POST – дані в HTTP-тілі).

Є ще методи HEAD, PUT, DELETE, TRACE, CONNECT, OPTIONS – але вони менш поширені.

Заголовок HTTP-запиту

- Відомості про типи документів, які клієнт приймає (Accept).
- Ім’я хоста, з якого запитується ресурс (Host).
- Тип браузера (User-Agent).
- Дата та загальна конфігураційна інформація

Заголовок складається з декількох рядків. Ознака закінчення заголовку – пустий рядок.

Тіло HTTP-запиту

Якщо в рядку запиту HTTP використовується метод POST, то HTTP-тіло містить довільні дані (наприклад дані форми).

Інакше тіло HTTP запиту пусте, як в нашому випадку (див. Приклад вище).

HTTP-відповідь

- Рядок відповіді (версія HTTP, код - успіх або помилка)).
- Заголовок.
- Тіло.

Приклад.

```
HTTP/1.1 200 OK
Date: Sun, 06 Jan 2008 11:19:28 GMT
Server: Apache/2.2.4
Pragma: no-cache
Content-Length: 23341
Content-Type: text/html; charset=windows-1251

<html> . . .
</html>
```

Існують 5 класів відповідей:

100 – 199 - запит ще обробляється;
200 – 299 - успіх;
300 – 399 - запит не виконано через переміщену інформацію;
400 – 499 - клієнтська помилка (запит некоректний);
500 – 599 - серверна помилка (запит був коректним).

Основні заголовки

- **Accept**. Інформує сервер про типи даних, які підтримуються браузером. Перечислення йде через кому. Використовується змінна оточення HTTP_ACCEPT.
- **Content-type**. Ідентифікує тип передаваємих даних. Іменування типів даних вказано в форматі стандарта MIME. Це тот самий формат передачі, який використовується методами GET і POST. Сервер ніяк не інтерпретує цей заголовок, а просто передає його сценарію через змінну оточення CONTENT_TYPE.

- **Content-length**. Цей заголовок містить довжину передаваємих даних в байтах при використанні метода передачі POST. Змінна оточення CONTENT_LENGTH.
- **Cookie**. В цьому заголовку зберігаються всі Cookies. Для установки Cookies застосовується заголовок Set-Cookie. Змінна оточення HTTP_COOKIE.
- **Location**. Отримавши цей заголовок разом з вказаним в ньому URL, браузер негайно переходить по вказаному URL.
- **Pragma**. Даний заголовок використовується для різних цілей, одна из яких – це зоборона кешування документа.
- **Server**. Даний заголовок містить назву і версію програмного забезпечення сервера.
- **Referer**. Містить URL сторінки, звідки клієнт прийшов на нашу. Змінна оточення: HTTP_REFERER.
- **User-Agent**. Містить версію браузера. Змінна оточення: HTTP_USER_AGENT.

Стандарт MIME

MIME (Multipurpose Internet Mail Extensions) – багатоцільові розширення поштового стандарту Інтернету. Спочатку MIME був створений для вказівки, якого типу документ вкладений у повідомлення електронної пошти.
MIME-тип задається у вигляді «тип/підтип». Наприклад: text/html
Стандарт MIME визначає сім типів даних:

- application;
- audio;
- image;
- message;
- multipart;
- text
- video.

Корисні програми для вивчення HTTP

- HTTP Analyzer v7.5.2.454 ($119). HTTP Analyzer is such a handy tool that allows you to monitor, trace, debug and analyze HTTP/HTTPS traffic in real-time. It is used by industry-leading companies including Microsoft, Cisco, AOL and Google.
- Відлагоджувальний проксі Fiddler (free).
- Postman – HTTP-клієнт для тестування сайтів (free).
- Плагіни для браузера Firefox: Firebug, LiveHTTPHeaders, HttpFox.

Змінні оточення

Для зв'язку між web-сервером і додатком використовується стандарт CGI (Common Gateway Interface, загальний інтерфейс шлюзу). Цей зв'язок забезпечується змінними оточення web-сервера, до яких, при необхідності, додаток звертається для отримання даних.

- REMOTE_ADDR - IP-адрес хоста, що відправив запит.
- REMOTE_HOST - Ім'я хоста, с якого відправлено запит.
- REQUEST_METHOD - Метод, що був використаний при відправці запиту.
- QUERY_STRING - Інформація, що знаходиться в URL після знаку питання.
- SCRIPT_NAME - Віртуальний шлях до програми, яка повинна виконуватися.

Отримати значення змінної оточення можна за допомогою функції getenv():

echo getenv(“REMOTE_ADDR”);

Тема 3. Вступ до PHP.

Мова PHP створена спеціально для розробки веб-додатків. Коротко розглянемо історію мови PHP [1].

- 1995 р. данський програміст Расмус Лердорф (англ. *Rasmus Lerdorf*) написав Perl-скрипт для домашньої сторінки. Назвав PHP (Personal Home Page). Потім переписав мовою С.
- 1997 р. Andi Gutmans та Zeev Suraski (Ізраїль) покращили функціоналльність PHP. У 1998 р. був вже PHP 3 і працював на 10% веб-серверів. Назву змінили. PHP – це рекурсивний акронім PHP: Hypertext Preprocessor (препроцесор гіпертексту).
- 2000 р. Andi та Zeev покращили продуктивність PHP 4. Движок PHP назвали Zend Engine.
- 2004 р. PHP 5. Підтримка ООП та XML. Остання стабільна версія PHP 5.6.33 (04 Jan 2018).
- З 2006 р. працювали над PHP 6, планували підтримку Юнікоду. Однак в березні 2010 р. розробка PHP 6 визнана безперспективною через складності з Юнікодом.
- Остання версія PHP: PHP 7.2.2 (01 Feb 2018)

PHP Is Not Part of Your Browser

Як показано на Рис.8, PHP не є частиною вашого браузера.

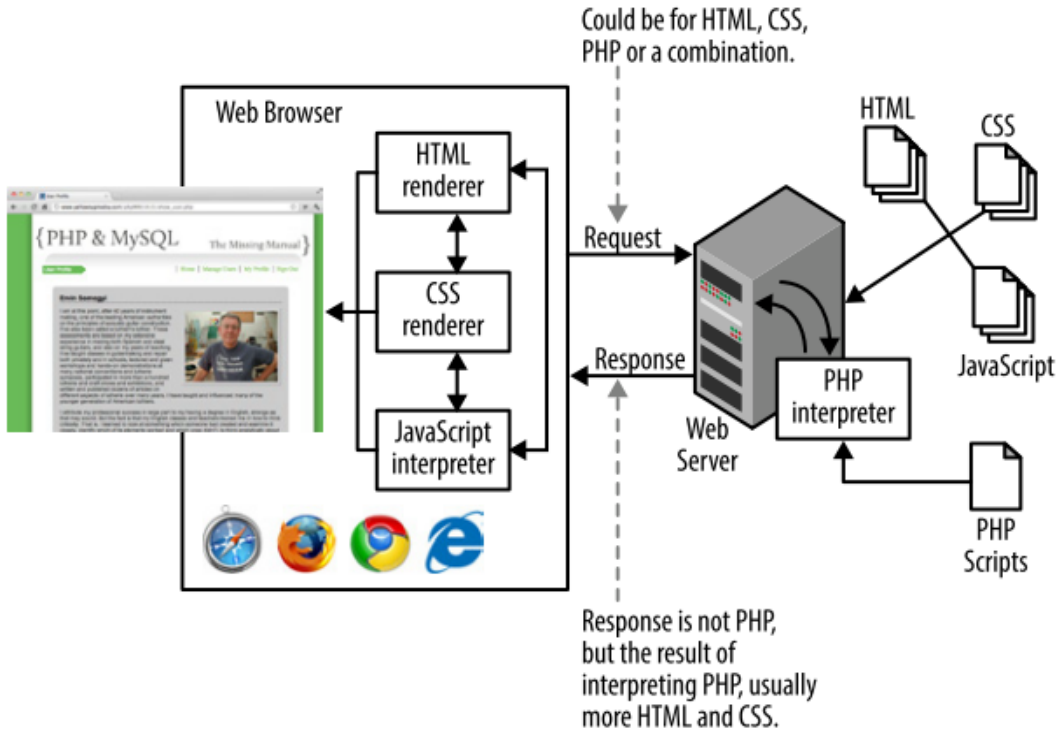

Рис. 8. Схема обробки запиту

Загальне визначення мови PHP

PHP – це мова сценаріїв для Web з відкритим вихідним кодом. PHP застосовується в складі серверного ПЗ та призначена для впровадження в код HTML.

Система підтримки мови PHP сумісна з основними типами web-серверів (Apache, IIS, Nginx та інші).

Переваги: безкоштовна, з відкритим кодом, міжплатформена, стабільна, швидка, чітко спроектована, проста в вивченні, підтримувана провайдерами.

Приклад. Програма “Hello World”

```
<!DOCTYPE html>
<meta charset=utf-8>
<title>PHP Test</title>
<?php echo 'Hello World';
?>
```

Основні можливості мови

- Всі вирази в PHP повинні закінчуватись крапкою з комою. Виняток – останній вираз перед закриваючим тегом.
- Блоки коду позначаються фігурними дужками { }.
- Конструкції мови PHP та назви функцій не чутливі до регістру, а імена змінних та констант – чутливі.

```
<?php
ECHO "Hello World"; // works
$variable = "Hello World";
```

```
echo $VARIABLE; // don't work
```

Приклад обробки форми

Файл formexample.php

```
<html>
<head> <title> PHP Test </title> </head>
<body>
<?php
     if(isset($_POST['submit'])){
     echo "Hi, ".$_POST['name']."!<br/>";
     }
?>
<form action="formexample.php" method="post">
     <p>
     Name: <br/>
     <input  type="text" name="name" size="20"
maxlength="40" value="" />
     </p>
     <input  type="submit" name="submit" value="Go!" />
</form>
</body>
</html>
```

Інша форма “Hello”-програми

Файл hello.php

```
<?php
if  ('POST' == $_SERVER['REQUEST_METHOD']) {
print  "Hello, ". $_POST['my_name'];
} else  {
print<<<_HTML_
<form method="post" action="$_SERVER[PHP_SELF]">
 Your name: <input type="text" name="my_name" >
<br>
<input type="submit" value="Say Hello">
</form>
_HTML_;
}
```

Ілюстрація обробки форми для цього прикладу наведена на Рис.9.

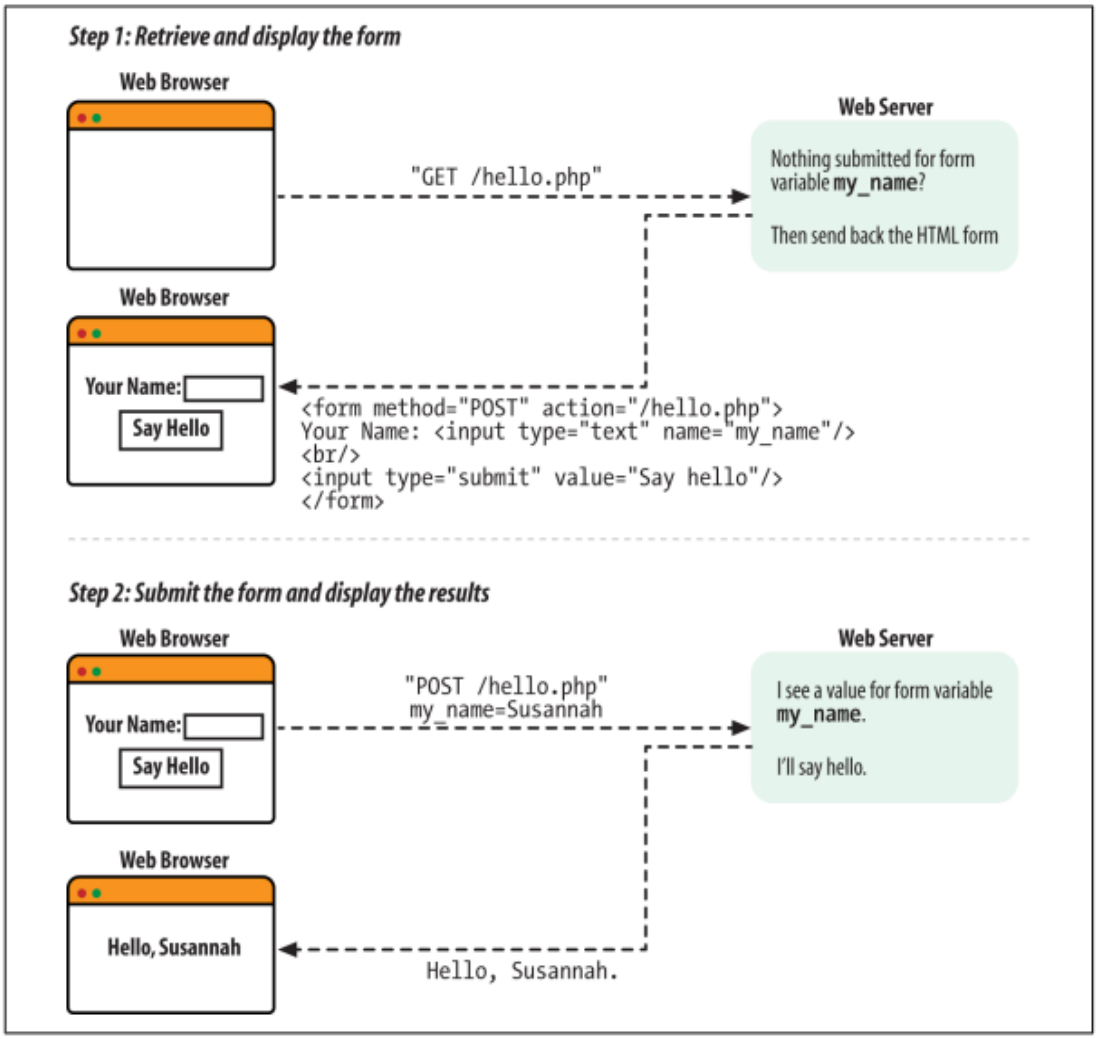

Рис. 9. Ілюстрація обробки форми

Відокремлення PHP коду від HTML розмітки

Команди мови PHP обрамляються спеціальними дескрипторами – тегами мови PHP. Існують наступні стилі написання тегів:

- XML-стиль (рекомендований): <?php код на PHP ?> .
- Echo-стиль: <?= ?> .
- HTML-стиль: <script language="php"> код на PHP </script> . Не використовуйте.
- Скорочений стиль: <? код на PHP ?> . Не підтримується.
- ASP-стиль: <% код на PHP %> . Не підтримується.

Echo-стиль <?= ?>

- Тег echo дозволяє вам легко повторювати змінну PHP і робить ваш HTML документ простішим для читання.
- Він зазвичай використовується у шаблонах, де ви хочете вивести декілька значень в різні позиції на сторінці.

- Його використання простіше зрозуміти, коли відобразити цей стиль разом із еквівалентом у стандартних кодах відкриття. Два наступні теги ідентичні:

  <?= $variable ?>
  <?php echo $variable ?>

Приклад.

Ви можете використовувати логіку PHP між відкриттям і закриттям тегів, як це відбувається в цьому прикладі:

```
Balance:
<?php
if ($bankBalance > 0): ?>
<p class="black">
<?php else: ?>
<p class="red">
<?php endif; ?>
<?= $bankBalance?>
</p>
```

Закриваючий тег

- У PHP-програмах досить часто викидають закриваючий тег ?> у файлі.
- Це прийнятно для аналізатора, і це є корисним способом запобігти проблемам з символами newline, що з'являються після закриваючого тегу.
- Ці символи нового рядка надсилаються в якості результату інтерпретатору PHP і можуть перешкоджати заголовкам HTTP або викликати інші небажані побічні ефекти.
- Не закриваючи скрипт у файлі PHP, ви запобігаєте можливості передачі нових символів.

Коментарі

PHP підтримує три види коментарів: один багаторядковий і два однорядкових. PHP-парсер ніяк не аналізує коментарі, їх просто ігнорують

```
<?php
     /*
     Перший    вид  коментарів Multiline comment
      */
     //   Другий   C style comments
     #    Третій  Perl style comments
?>
```

Лекції 4-6. Мова PHP для створення серверних web-застосунків

Тема 4. Типи даних в PHP. Змінні. Ключові слова global та static. Суперглобальні змінні. Константи. Рядки. Масиви. Функції . Обробка форм.

Змінні в мові PHP

- Всі імена змінних повинні починатися зі знака долара ($).
- Оголошення не є обов'язковим. Змінна починає існувати з моменту присвоєння їй значення або з моменту першого використання. Якщо використання починається раніше присвоєння, то змінна буде містити значення за замовчуванням.
- Змінній не призначається певний тип. Тип визначається значенням, що зберігається і поточною операцією.
- Перша літера a-z, A-Z або _ . Потім можна використовувати цифри.

Локальні змінні

Локальні змінні - змінні, визначені всередині підпрограми (функції). Вони доступні тільки всередині функції, в якій вони визначені.

Приклад:

```
<?php
     $a = 100;
     function  funct() {
          $a = 70;
          echo "<h4>$a</h4>";
     }
     funct();
     echo "<h2>$a</h2>";
?>
```

Сценарій виведе спершу 70, а потім 100.

Ключові слова global та static

Існує спосіб доступу до зовнішніх змінних всередині функцій. Якщо змінна оголошена з ключовим словом global, то вона буде доступна всередині функції:

```
$my_data="Зовнішні дані";
function send_data(){
     global $my_data;
     echo $my_data;
}
send_data();
```

Статичні змінні. Якщо при створенні змінної всередині функції використати ключове слово static, то ця змінна та її значення буде зберігатися між викликами функції.

Посилальні змінні. Жорсткі посилання. Операція &.

Приклад: $a=10; $b=&$a; $b=0;

Суперглобальні змінні

$GLOBALS – Масив, що містить усі глобальні змінні.
$_ENV – Масив змінних оточення.
$_COOKIE – Масив файлів cookie, відправлених на сервер.
$_GET – Масив змінних, відправлених методом GET.
$_POST – Масив змінних, відправлених методом POST.

$_FILES – Масив, що містить інформацію про завантажені файли.

$_REQUEST – Масив, що містить $_GET, $_POST, $_FILES, $_COOKIE.

$_SESSION – Масив змінних, розміщених в сесіях PHP.

$_SERVER – Масив, що містить інформацію про сервер.

Типи даних

PHP підтримує вісім типів даних.Чотири скалярних типи:

- boolean — логічний;
- integer — ціле число;
- float (double) — число с плаваючою точкою;
- string — рядок.

Два змішані типи:

- array — масив;
- object — екземпляр класа.

Два спеціальних типи:

- resource — посилання на зовнішнє по відношенню до скрипта джерело даних (файл на диску, зображення в пам'яті та т.п.);
- NULL — відсутність якого небудь значення

Приклад

```
$test_var;
echo gettype($test_var).”<br/>”;    //покаже “NULL”
$test_var=15;
echo gettype($test_var).”<br/>”;    //покаже “integer”
$test_var=8.23;
echo gettype($test_var).”<br/>”;    //покаже “double”
$test_var=“Hello!”;
echo gettype($test_var).”<br/>”;    //покаже “string”
```

Змінні змінних (Variable Variables). Це змінна, чиє ім'я містить іншу змінну.

```
$name='foo'; $$name='bar';
echo $foo;  // Displays 'bar'
```

Типи даних. Корисні функції

isset (ім'я_змінної) – повідомляє, чи існує змінна;.

unset(ім'я_змінної) – знищує (видаляє) вказану змінну;

empty(ім'я_змінної) – чи присвоєно змінній яке-небудь значення;

gettype(ім'я_змінної) – повертає тип вказаної змінної;

settype(ім'я_змінної, тип) - конвертує змінну в інший тип;

is_bool(ім'я_змінної) – перевіряє, чи є тип змінної логічним;

Функції is_numeric(), is_float(), is_int(), is_string(), is_object(), is_array() працюють за аналогією.

Константи

Визначаються за допомогою функції define(string_name, string_value).

Приклад

```
<?php
define('PI', 3.142);
echo PI;
define('UNITS',['MILES_CONV'=>1.6,'INCHES_CONVERSION'=>
'2.54']);
echo "5km in miles is " . 5 * UNITS['MILES_CONVERSION'];
/*
3.1425km in miles is 8
*/
```

- У констант нема префікса у вигляді знака долара.
- Константам не можна присвоювати значення.

Ключове слово const

Ви також можете використовувати ключове слово const для визначення констант.

Константи можуть містити лише масиви або скалярні значення, а не ресурси або об'єкти.

```
<?php
     const UNITS = ['MILES_CONVERSION' => 1.6,
     'INCHES_CONVERSION' => '2.54'];
     echo "5km in miles is " . 5 *
UNITS['MILES_CONVERSION'];
     /*
     5km in miles is 8
     */
```

Вбудовані константи

__LINE__ - номер поточного рядка.
__FILE__ - повний шлях та ім’я поточного файлу.
__FUNCTION__ - ім’я поточної функції.
__CLASS__ - ім’я поточного класу.
PHP_EXTENSION_DIR - каталог разширень PHP
PHP_OS - операційна система
PHP_VERSION - версія PHP
PHP_CONFIG_FILE_PATH - каталог розміщення php.ini

Рядки в PHP

- PHP-рядки являють собою серію байтів і не містять жодної інформації про те, як ці байти повинні бути переведені на символи.
- PHP зберігає довжину рядка разом із його вмістом і не спирається на закінчуючий символ, щоб позначити кінець рядка. Це допомагає зробити рядки безпечними, оскільки нульові символи в рядку не викликають плутанини.
- На 32-бітних системах рядок може займати до 2 Гб. Не існує конкретної межі щодо довжини рядка у 64-бітній системі PHP.

- Рядки можуть бути укладені в апострофи або в лапки. Це впливає на те, як рядки будуть оброблятися інтерпретатором.
- Рядки в апострофах залишаються майже незмінними.

```
$win_path='C:\\Inetpub\\PHP\\';
echo $win_path;     // Виведе   C:\Inetpub\PHP
```

- Рядки в лапках піддають попередній обробці.
  - Деякі символьні послідовності, що починаються з '\' замінюються спецсимволами
  - Імена змінних (починаються з $) замінюються рядковими представленнями їх значень

```
<?php
$name = 'Bob';
$a = 'Hello $name\n';
$b = "Hello $name\n";
echo $a;        // Hello $name\n
echo $b;        // Hello Bob
```

Рядки Here-документ (heredoc)

```
$name="Білл  Гейтс";
$text=<<<MARKER
Далі іде текст
Можливо зі змінними,
Які інтерпретуються, $name
MARKER;
```

Виклик зовнішньої програми

Останній вид рядка – це рядок в зворотніх апострофах (наприклад, `команда`). Змушує PHP виконати команду ОС і те, що вона вивела, підставити.

Приклад

```
$st=`commamd.com  /c  dir`;
echo "<pre> $st </pre>";
```

(В PHP можна встановити безпечний режим з обмеженнями)

Символічні посилання

Символічне посилання – це рядкова змінна, що містить імя іншої змінної. Щоб дістатися до значення змінної, на яку посилаються, необхідно застосувати операцію розіменування – додатковий знак $ перед іменем посилання.

Приклад.

```
$right = "червона";
$wrong = "синя";
$color = "right";
echo  $$color;  // Виведе "червона"
$$color = "жовта"; // Присвоює $right нове значення
```

Фігурні дужки

PHP дозволяє використовувати фігурні дужки, щоб явно сказати парсеру, що частина рядка повинна бути обчислена.

Це необхідно, наприклад, при виведенні елемента з масиву, де не може бути відразу зрозуміло, що квадратні дужки призначені як пунктуація в рядку або як синтаксис для посилання на елемент масиву.

```
<?php
$burger = "Cheeseburger";
echo "I can haz {$burger}"; // I can haz Cheeseburger
echo "I can haz ${burger}"; // I can haz Cheeseburger
echo "I can haz $burgers";  // no variable $burgers
echo "I can haz {$burger}s"; // I can haz Cheeseburgers
echo  "I  can  haz  {  $burger  }";  //  I  can  haz  {
Cheeseburger }
```

Посилання на символи в рядках

Ви можете вказати позицію в рядку за допомогою квадратних дужок або фігурних дужок, щоб позначити цільову позицію, яку ви хочете вказати.

```
<?php
$hello = "world";
echo $hello[0]; // w
echo $hello{1}; // o
```

Обережно! Пам'ятайте, що рядки є серією байтів, і ви посилаєтесь на позицію байтів. Якщо ваш набір символів використовує більше одного байта на символ, ви не отримаєте очікуваного результату.

Деякі функції для роботи з рядками

strlen(string) - визначає довжину рядка;
ltrim(string) - видаляє символи-розділювачі на початку рядка;
rtrim(string) - видаляє символи-розділювачі в кінці рядка;
strpbrk(string, char) - шукає символ char;
strstr(string, string) - знаходить першу появу рядочка у рядку;
str_replace(. . .) - замінює рядок пошуку на рядок заміни;
strrev(string) - перевертає рядок;
strrpos(string, char) - знаходить останню появу заданого символу у рядку;
explode – розбиває рядок на підрядки;
trim – видаляє пробіли з початку та кінця рядка;
addslashes – екранує спецсимволи в рядку.

Проблема з підтримкою UTF-8

Треба підключити розширення *mbstring* для підтримки багатобайтових кодувань (в файлі *php.ini* розкоментувати рядки

```
extension  dir  =  "ext"
extension=php_mbstring.dll
```

).

Приклад

```
<?php
$str  =  "Привіт, Том!";
echo  "В рядку &quot;$str&quot; ".strlen($str)." байтів<br />"; //21
echo   "В  рядку  &quot;$str&quot;  ".mb_strlen($str)."  символів<br  />";
//12
```

```
?>
```

Масиви в мові PHP

Масиви з числовими індексами. Індекси починаються з нуля.

```
<?php
     $zoo[0] = 'слон';
     $zoo[6] = 'крокодил';
     $zoo[4] = 'жираф';
     $zoo[] = 'осел';// Індекс дорівнює 7
     // або
     $zoo = array('лев', 'ведмідь', 'мавпа');
     echo count($zoo); // Количество элементов массива
$a = [1,2,3]; // PHP 5.4
?>
```

Функції створення масивів

- array([...]) – створює масив з переданих значень.
- array_fill ($start_index, $num, $value) – повертає масив, що містить $num елементів, що мають значення $value. Нумерація індексів при цьому починається зі значення $start_index.
- range($low, $high [, $step]) – створює масив зі значеннями з інтервалу від $low до $high і кроком $step.
- explode($delimiter, $str [, $limit]) – функція повертає масив з рядків, кожен з яких відповідає фрагменту вихідного рядка $str, що знаходиться між роздільником, визначеним аргументом $delimiter. Необов'язковий параметр $limit визначає максимальну кількість елементів у масиві.

Функція array_fill()

```
<?php
// Створюємо масив
$arr = array_fill(5, 6, "Hello world!");
// Виводимо дамп масиву
echo "<pre>";
print_r($arr);
echo "</pre>";
?>
```

Результат:

```
Array
(
[5] => Hello world!
[6] => Hello world!
[7] => Hello world!
[8] => Hello world!
[9] => Hello world!
[10] => Hello world!
)
```

Функція explode()

```
<?php
$str = "Ім'я Прізвище e-mail";
```

```
$arr = explode(" ", $str);
echo "<pre>";
print_r ($arr);
echo "</pre>";
?>
Результат:
Array
(
[0] => Ім’я
[1] => Прізвище
[2] => e-mail
)
```

Випадкові елементи масиву

Для отримання випадкового значення індексного масиву можна використовувати функцію rand (), яка повертає випадкове число із заданого діапазону.

- Функція rand () має наступний синтаксис:
  rand ([$ min, $ max]);

Приклад.

```
<?php
// Оголошуємо массив
$arr = array(0, 1, 2, 3, 4, 5, 6, 7, 8, 9);
// Отримуємо випадковий індекс масиву
$index = rand(0, count($arr) - 1);
// Виводимо випадковий елемент масиву
echo $arr[$index];
?>
```

- array_rand ($arr [, $num_req])) – вибирає одне або декілька випадкових значень з масиву.
- shuffle ($arr)) – перемішує елементи масиву $arr в випадковому порядку.

```
<?php
$arr = array("синій", "жовтий", "зелений", "червоний",
"оранжевий");
$rand_keys = array_rand($arr, 2);
echo "<pre>";
print_r($rand_keys);
echo "</pre>";
?>
```

Асоціативні масиви

В асоціативних масивах використовується не числовий, а рядковий індекс.

```
<?php
     $pets['dog'] = 'Бульдог';
     $pets['cat'] = 'Шиншила';
     $pets['fish'] = 'Золота';
     // або
     $pets = array ('lizard' => 'Ігуана',
```

```
                         'spider' => 'Чорна вдова',
                         'parrot' => 'Apa');
     print_r ($pets);    // Друкування масиву
?>
```

Перегляд асоціативного масиву за допомогою foreach...as

```
<?php
$paper = array('copier' => "Copier & Multipurpose",
               'inkjet' => "Inkjet Printer",
               'laser'  => "Laser Printer",
               'photo'  => "Photographic Paper");
foreach ($paper as $item => $description)
    echo "$item: $description<br>";
?>
Результат:
copier: Copier & Multipurpose
inkjet: Inkjet Printer
laser: Laser Printer
photo: Photographic Paper
```

Перегляд асоціативного масиву за допомогою each та list

```
<?php
$paper = array('copier' => "Copier & Multipurpose",
               'inkjet' => "Inkjet Printer",
               'laser'  => "Laser Printer",
               'photo'  => "Photographic Paper");
while (list($item, $description) = each($paper))
    echo "$item: $description<br>";
?>
Результат:
copier: Copier & Multipurpose
inkjet: Inkjet Printer
laser: Laser Printer
photo: Photographic Paper
```

Багатовимірні масиви

Багатовимірний массив – це масив, елементами якого є не тільки скалярні величини, але й самі масиви, наприклад

```
<?php
     $users = array (
                    0 => array (
                         'login' => 'admin',
                    'password' => 'hskdfuegefdjfdg'
                                  ),
                    1 => array (
                         'login' => 'telo',
                         'password' => 'ppqmcnvkfghye'
                                  )
     );
     echo $users[0]['login']; // admin
```

Приклад.

```
<?php
```

```
     $objects = array(
          'Банка содової' => array(
          'Форма' => Циліндр','Колір' => 'Червоний'
               'Матеріал' => 'Метал'
                         ),
          'Блокнот' => array(
          'Форма' => Прямокутник','Колір' => 'Білий'
               'Матеріал' => 'Папір'
                         )
     );
     foreach   ($objects as $obj_key => $obj){
          echo "$obj_key:<br>";
          while(list($key, $value) = each($obj)){
               echo "$key=$value ";
          };
     echo "<br/>";
}
?>
```

Функція each() повертає ключ та значення поточного елементу. Можна було використати вбудовану функцію var_dump($object).

Приклад. Створення багатовимірного асоціативного масиву.

```
<?php
$products = array(
    'paper' =>  array(
        'copier' => "Copier & Multipurpose",
        'inkjet' => "Inkjet Printer",
        'laser'  => "Laser Printer",
        'photo'  => "Photographic Paper"),
    'pens' => array(
        'ball'   => "Ball Point",
        'hilite' => "Highlighters",
        'marker' => "Markers"),
    'misc' => array(
        'tape'   => "Sticky Tape",
        'glue'   => "Adhesives",
        'clips'  => "Paperclips") );
echo "<pre>";
foreach ($products as $section => $items)
    foreach ($items as $key => $value)
        echo "$section:\t$key\t($value)<br>";
echo "</pre>";
?>
```

Результат:

```
paper:  copier  (Copier & Multipurpose)
paper:  inkjet  (Inkjet Printer)
paper:  laser   (Laser Printer)
paper:  photo   (Photographic Paper)
pens:   ball    (Ball Point)
pens:   hilite  (Highlighters)
pens:   marker  (Markers)
misc:   tape    (Sticky Tape)
misc:   glue    (Adhesives)
```

```
misc:   clips   (Paperclips)
```

Сортування масивів

- **sort** ($array [, $sort_flags])) – сортує масив $array в прямому порядку. Параметр $sort_flags задає параметри сортування: SORT_REGULAR- звичайне сортування; SORT_NUMERIC- порівнювати елементи як числа; SORT_STRING- порівнювати елементи як рядки;
- **rsort** ($array [, $sort_flags]) – сортує масив $array в зворотному порядку.
- **asort** ($array [, $sort_flags]) – сортує масив $array в прямому порядку із збереженням відношення ключ-значення.
- **arsort** ($array [, $sort_flags]) – сортує масив $array в зворотному порядку зі збереженням відношення ключ-значення.

Приклад.

```
<?php
$number = array("2", "1", "4", "3","5"); // вихідний масив
echo «До сортування: <br>";
for($i=0; $i < count($number); $i++){
echo "$number[$i] ";
}
sort($number); // сортування за зростанням
echo "<br> Після сортування: <br>";
for($i = 0; $i < count($number); $i++)
{
echo "$number[$i] ";
}
?>
```

Сортування рядків відбувається за старшинством першої літери в алфавіті

Деякі функції для роботи з масивами

array_chunk - розбити масив на частини.
array_merge - злити два або більшу кількість масивів.
array_pop - витягти останній елемент масиву.
array_push - додати один або кілька елеметов в кінець масиву.
array_rand - вибрати одне або кілька випадкових значень.
array_reverse - повертає масив з елементами в зворотному порядку.
array_shift - витягти перший елемент масиву.
array_splice - видалити послідовність елементів масиву і замінити її іншою послідовністю.
count - порахувати кількість елементів масиву.
in_array - перевірити, чи присутній в масиві значення.

Суперглобальний масив $ _GET

GET-параметри передаються в рядку запиту після символу питання (?)
Приклад 1.

```
http: //localhost/index.php? id = 1
<?php
echo $_GET['id']; // 1
?>
```

Приклад 2.

```
http://localhost/index.php?id =1&number=2&page=10
<?php
echo "<pre>";
print_r($_GET);
echo "</pre>";
?>
```

Результат:

```
Array
(
[id] => 1
[number] => 2
[page] => 10
)
```

Отримання даних форми

Приклад 1. Відправка даних

Файл login_form.html.

```
<html>
  <head>
    <title> Відправка форми </title>
  </head>
  <body>
   <form action="handler.php" method="POST">
Логин: <input type="text" name="login"><br>
Пароль:<input type="password" name="pass"><br>
 <input type="submit" value="Відправити">
    </form>
  </body>
</html>
```

Приклад 1. Обробка форми

Після відправки дані потрапляють в суперглобальні масиви, імена яких відповідають назві методу відправки.

$ _POST - Якщо дані передані методом POST;

$ _GET - Якщо дані передані методом GET;

$ _REQUEST - Якщо дані були передані будь-яким з них.

Файл handler.php

```
<?php
     if(isset($_POST['login']) && $_POST['login'] != '' &&
                    isset($_POST['pass']) && $_POST['pass']
!= ''){
          echo 'Привіт ' . $_POST['login'] . '!';
          echo 'Твій пароль: ' . $_POST['pass'] . '<br>';
     }else{
          echo 'Некоректне імя та пароль!<br>';
     }
?>
```

Як використовувати функцію filter_input

Ця функція введена в PHP 5.2 і є рекомендованою при отриманні даних з масивів $_GET та $_POST

Синтаксис:

`filter_input($type, $variable_name [ , $filter] )` , де

- type – Задає суперглобальний масив для доступу (INPUT_GET, INPUT_POST, INPUT_COOKIE)
- variable_name – Ім'я змінної
- filter – FILTER _VALIDATE_INT, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_EMAIL, FILTER_VALIDATE_URL , FILTER_VALIDATE_BOOLEAN.

Приклад [21].

Перша сторінка (index.html) показана на Рис.10.

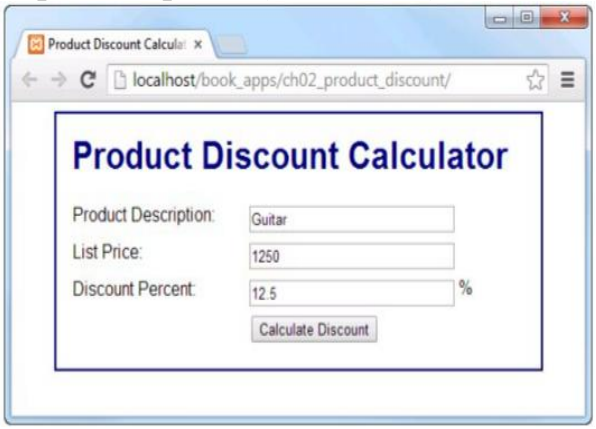

Рис.10. Форма

```
<h1>Product Discount Calculator</h1>
<form action="display_discount.php" method="post">
   <div id="data">
      <label>Product Description:</label>
      <input type="text" name="product_description"><br>
      <label>List Price:</label>
      <input type="text" name="list_price"><br>
      <label>Discount Percent:</label>
      <input type="text" name="discount_percent">
<span>%</span><br>
    </div>
    <div id="buttons">
      <label> </label>
      <input type="submit" value="Calculate Discount"><br>
    </div>
 </form>
```

Друга сторінка (display_discount.php) показана на Рис.11.

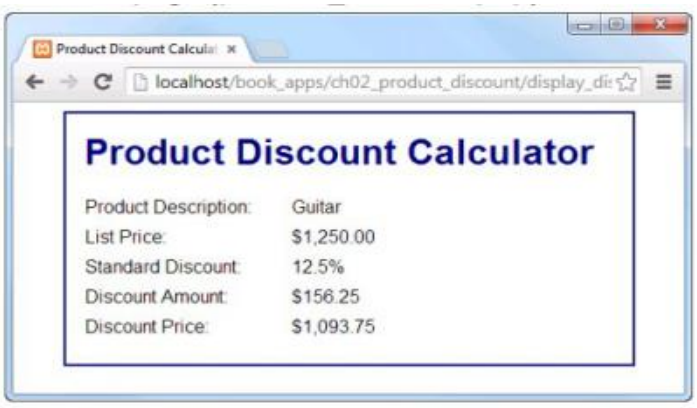

Рис.11. Результат обробки форми

```
<?php
    // get the data from the form
    $product_description = filter_input(INPUT_POST,
'product_description');
    $list_price = filter_input(INPUT_POST, 'list_price');
    $discount_percent = filter_input(INPUT_POST,
'discount_percent');

    // calculate the discount and discounted price
    $discount = $list_price * $discount_percent * .01;
    $discount_price = $list_price - $discount;

    // apply currency formatting to the dollar and percent amounts
    $list_price_f = "$".number_format($list_price, 2);
    $discount_percent_f = $discount_percent."%";
    $discount_f = "$".number_format($discount, 2);
    $discount_price_f = "$".number_format($discount_price,
2);
?>
<!DOCTYPE html>
<html>
<head>
    <title>Product Discount Calculator</title>
    <link rel="stylesheet" type="text/css" href="main.css">
</head>
<body>
    <main>
        <h1>Product Discount Calculator</h1>
        <label>Product Description:</label>
    <span><?php echo htmlspecialchars($product_description);
?></span>
        <br>
        <label>List Price:</label>
        <span><?php echo htmlspecialchars($list_price_f);
?></span>
        <br>
. . . . . .
```

Як отримати дані з radio button

Приклад.

```
<input  type="radio"  name="card_type"  value="visa"  checked>
```

```
Visa<br>
<input type="radio" name="card_type" value="mastercard">
MasterCard<br>
<input type="radio" name="card_type" value="discover">
Discover
```

Рис.12. Форма з radio button

```
<?php
$card_type = filter_input(INPUT_POST, 'card_type' );
?>
```

А якщо нема значення за замовчуванням:

```
<?php
$card_type = filter_input(INPUT_POST, 'card_type' );
if ($card_type == NULL) {
$card_type = ' unknown ';
}
?>
```

Як отримати дані з check box

Приклад.

```
<input type="checkbox" name="pep" checked>
Pepperoni<br>
<input type="checkbox " name="msh"> Mushrooms<br>
<input type="checkbox " name="olv"> Olives
```

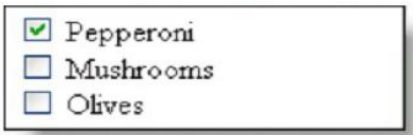

Рис.13. Форма з check box

```
<?php
$pepperoni= isset($_POST ['pep']);
$mushrooms= isset($_POST ['msh']);
$olives= isset($_POST ['olv'] );
?>
```

Як отримати дані з масиву check box

Приклад [21].

```
<input type="checkbox " name="top[]" value="pep">
Pepperoni<br>
<input type="checkbox " name="top[]" value="msh">
Mushrooms<br>
<input type="checkbox" name="top[]" value="olv"> Olives
```

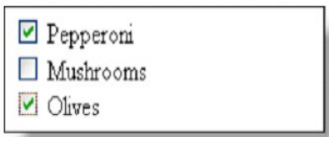

Рис.14. Форма з масивом check box

```
<?php
```

```
$toppings= filter_input(INPUT_POST,    ' top ' ,
FILTER_SANITIZE_SPECIAL_CHARS,   FILTER_REQUIRE_ARRAY) ;
?>
if  ($toppings !==  NULL)    {
$topl  =   $toppings [0];
$top2  =   $toppings [l];
$top3  =   $toppings [2];
}
```

Як отримати дані з drop-down list

Приклад [21].

```
<select  name="card_type">
<option  value="visa">Visa</option>
<option  value="mastercard">MasterCard</option>
<option  value="discover">Discover</option>
< /select>
```

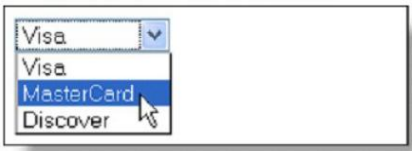

Рис. 15. Форма з випадаючим списком

```
<?php
$card_type  =  filter_input( INPUT_POST,  'card_type' );
?>
```

Як отримати дані з list box

Приклад [21].

```
<select  name="card_type"  size=" 3">
<option  value="visa"> Visa </option>
<option  value="mastercard"> MasterCard </option>
<option  value="discover"> Discover </option>
</select>
```

Рис.16. Форма 1 зі списком [21].

```
<select  name="top[]"   size="3"  multiple>
<option  value="pep"  selected>Pepperoni</option>
<option  value="msh">Mushrooms</option>
<option  value="olv ">Olives</option>
< /select>
```

Рис.17. Форма 2 зі списком

```
<?php
$toppings  =  filter_input (INPUT_ POST,    ' top ' ,
FILTER_SANITIZE_SPECI AL_ CHARS,   FILTER_REQUIRE_ ARRAY );
if  ($toppings  !==  NULL)   {
foreach  ($toppings  as   $key=> $value)    {
```

```
     echo  $key.‘ = '.$value.'<br>' ;          //  '0 = pep'
and  '1 = msh'
     } }  else  {
     echo  ' No   toppings  selected. ' ;
     }
     ?>
```

Як отримати дані з text area

Приклад [21].

```
     < textarea  name="conunent"  rows="4"  cols ="50">Welcome
to  PHP  and MySQL!
     < /textarea>
```

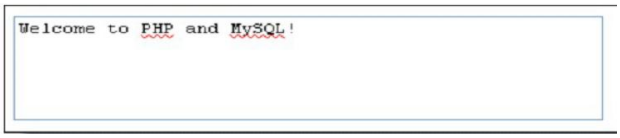

Рис.18. Форма з text area

```
     <?php
     $conunent =  filter _ input(INPUT_POST,    'conunent');
     ? >
```

Функції в PHP

Функція – це частина програми, позначена певним ім'ям, що виконує певне завдання і необов'язково повертає результат.

```
     <?php
          function PrintName($name){
               echo $name;
          }
          $boy = 'Jack';
          PrintName($boy);    // Виведе: «Jack»
          PrintName('Sally'); // Виведе: «Sally»
     ?>
```

Якщо функції потрібно повернути значення, то останнім оператором повинен бути *return* value.

Функція може мати нуль або більше параметрів.

Функція визначається ключовим словом *function*.

Декларування типів аргументів

```
     <?php
     function addToShoppingCart(string $itemName, array $details)
{}
     function requestPayment(PaymentProviderInterface
$paymentObject) {}
     function calculateWage(Employee $employee) {}
     function performCalculation(callable $method) {}
```

Параметри за замовчуванням

```
     <?php
     function sayHi($message = 'world') {
     echo "Hello $message";
     }
     sayHi();
```

Overloading Functions

```
<?php
function myFunc() {
foreach(func_get_args() as $arg => $value) {
echo "$arg is $value" . PHP_EOL;
}
}
myFunc('variable', 3, 'parameters');
/*
0 is variable
1 is 3
2 is parameters
*/
```

Передача параметрів за посиланням

```
<?php
$a1 = "WILLIAM";
$a2 = "henry";
$a3 = "gatES";
echo $a1 . " " . $a2 . " " . $a3 . "<br />";
fix_names($a1, $a2, $a3);
echo $a1 . " " . $a2 . " " . $a3;
function fix_names(&$n1, &$n2, &$n3)
{
    $n1 = ucfirst(strtolower($n1));
    $n2 = ucfirst(strtolower($n2));
    $n3 = ucfirst(strtolower($n3));
}
?>
```

Результат: WILLIAM henry gatES
William Henry Gates

Повернення глобальних змінних

```
<?php
$a1 = "WILLIAM";
$a2 = "henry";
$a3 = "gatES";
echo $a1 . " " . $a2 . " " . $a3 . "<br />";
fix_names();
echo $a1 . " " . $a2 . " " . $a3;
function fix_names()
{
    global $a1; $a1 = ucfirst(strtolower($a1));
    global $a2; $a2 = ucfirst(strtolower($a2));
    global $a3; $a3 = ucfirst(strtolower($a3));
}
?>
```

Результат: WILLIAM henry gatES
William Henry Gates

Змінне число параметрів

Починаючи з PHP 5.6 можна перед параметром вказувати три крапки. Всередині функції такий параметр розглядається як масив.

```
<?php
function parameterTypeExample($required, $optional = null,
...$variadicParams) {
printf('Required: %d; Optional: %d; number of variadic
parameters:
%d'."\n",
$required, $optional, count($variadicParams));
}
f(1);
f(1, 2);
f(1, 2, 3);
f(1, 2, 3, 4);
```

Return Type Declarations

```
<?php
function getFullName(string $firstName, string $lastName):
string {
return 123;
}
$name = getFullName('Mary', 'Moon');
echo gettype($name);  // string
```

Функції. Області видимості

```
<?php
     $a = 1;// Глобальна область видимості
     function Test(){
     echo $a;// Локальна область видимості
     }
     Test();// Не виведе нічого
?>
```

Отримати доступ до глобальних змінних з локального контексту можна наступними способами:

```
<?php
     $a = 1; $b = 2;
     function Sum (){
     global $a, $b;
     return $b += $a;
     }
     echo Sum();
?>
<?php      // Рекомендований варіант
     $a = 1; $b = 2;
     function Sum () {
     return $GLOBALS['b'] += $GLOBALS['a'];
     }
     echo Sum();
?>
```

Анонімні функції (Lambda) та замикання (Closure)

Lambda в PHP – це анонімна функція, яку можна зберегти як змінну.

```
<?php
$lambda = function($a, $b) {
echo $a + $b;
};
echo gettype($lambda); // true
echo (int)is_callable($lambda); // 1
echo get_class($lambda); // Closure
```

Ми бачимо, що в PHP лямбда та замикання реалізовані як об'єкти, створені з класу Closure.

Замикання в PHP - це анонімна функція, яка інкапсулює змінні, таким чином, що їх можна використовувати, коли їх оригінальні посилання недоступні.

Тобто анонімна функція "закриває" змінні, які знаходяться в її області визначення. Наприклад:

```
<?php
$string = "Hello World!";
$closure = function() use ($string) {
echo $string;
};
$closure();
```

Виклик echo $string призведе до попередження, оскільки змінна не існує.

Основне призначення замикань – заміна глобальних змінних.

Генератори

Генератори з’явились в PHP 5.5. Вони дозволяють створювати власні ітератори, які можна використовувати в операторі foreach.

Генератор – це звичайна функція, в якій замість return використовується оператор yield.

```
<?php
function generator() {
for ($i = 0; $i < 99; $i++) {
yield $i;
}
}
foreach (generator() as $value) {
echo $value . " ";
}
```

Делегування генераторів

Починаючи з PHP 7 можна викликати одні генератори з других, використовуючи ключове слово *from* після *yield.*

```
<?php
function generator() {
$a = [1,2,3];
yield from $a;
yield from range(4,6);
yield from sevenAteNine();
}
```

```
function sevenAteNine() {
for ($i=7; $i<10;$i++) {
yield $i;
}
}
$gen = generator();
foreach ($gen as $value) {
echo $value . PHP_EOL;
}
```

Date/Time Functions

Представлення часу в форматі *timestamp*.

int time() – повертає час в секундах з початку “епохи UNIX” (01.01.1970)

Рядкове представлення дати.

string date(string Format) – повертає дату, відформатовану згідно рядку Format, в якому можуть бути такі символи форматування:

Y – рік (чотири цифри)

F – назва місяця

j – номер дня в місяці

l – день тижня

. . .

Розбор timestamp: array getdate() - повертає асоціативний масив, що містить дані поточної дати та часу (seconds, minutes, hours, …, mon, year)

Приклад.

```
<html>
     <head>
     <title>Date and Time Functions </title>
     </head>
     <body>
          <h1>Date and Time Example</h1>
          <?php
               echo date(DATE_RFC822), "<br />";
               $date_format = "l, F jS, Y";
               echo date($date_format), "<br />";
               print_r(getdate());
               $date_array = getdate();
echo "<br />Today is: ", $date_array["weekday"],
                    ", ", $date_array["month"], " ",
$date_array["mday"],
                    ", ", $date_array["year"], "<br />";
//Seconds since 1-1-1970
echo "Today's timestamp is: ", time(), "<br />";
//Seconds in 2 weeks
$twoweeks = (14*24*60*60);
$twoweeksFrNow = time() + $twoweeks;
echo "Two weeks from today is: ",
          date("l, F jS, Y", $twoweeksFrNow), "<br />";
     ?>
</body>
```

```
</html>
```

Результат роботи програми показаний на Рис.19.

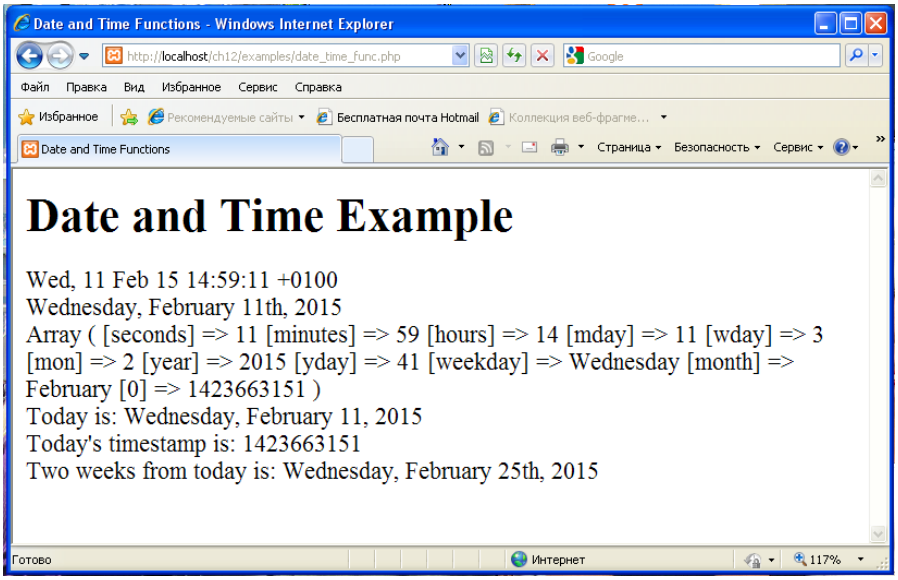

Рис.19. Результат роботи сценарію

Повторне використання коду (Code Reuse)

Щоб підключити в поточний файл код з стороннього файлу, можна використовувати наступні конструкції мови PHP:

include - Підключає код із зазначеного файлу в поточний.

require - Працює аналогічно include. Різниця в тому, що require, не знайшовши зазначеного файлу, виведе помилку і змусить сценарій завершитися, тоді як include виведе лише попередження, а сценарій продовжить роботу.

include_once - Підключить код тільки, якщо він ще не підключений, допоможе уникнути подвійного підключення коду і пов'язаних з ним помилок.

require_once - Працює аналогічно include_once, але з особливостями require.

Приклад.

```
<?php      // Файл Sum.function.php
     function Sum ($a, $b){
          return $a + $b;
     }
?>

<?php      // Файл index.php
     echo Sum (1,2);      // Викличе помилку
     include ('Sum.function.php');// Підключаємо Sum
     echo Sum (1,2);      // Виведе 3
?>
```

Тема 5. Зв'язок PHP з MySQL . Основні клієнтські команди MySQL. Створення бази даних в phpMyAdmin. Основні функції PHP для роботи з MySQL. Використання MySQLi.

Зв'язок PHP з MySQL

- MySQL – вільна система керування реляційними базами даних з відкритим кодом, GNU General Public License.
- Характеризується великою швидкістю, стійкістю і простотою використання.
- Крос-платформова. Підтримка різних мов програмування.
- Повноцінна підтримка Юнікоду (UTF-8 і UCS2)..
- Незалежні типи таблиць (MyISAM для швидкого читання, InnoDB для транзакцій і цілісності посилань).
- Необмежена кількість користувачів, що одночасно працюють із БД.
- Кількість рядків у таблицях може досягати 50 млн.
- Відомі користувачі: Apple, Amazon.com, Google, NASA, Nokia, Вікіпедія, Yahoo!
- Розробник: MySQL AB (підрозділ Oracle).

Визначення прав доступу до БД

- Відвідувач веб-сайту: тільки SELECT.
- Учасник розробки: SELECT, INSERT, можливо UPDATE.
- Редактор вмісту сайту: SELECT, INSERT, UPDATE, можливо DELETE і можливо GRANT.
- Адміністратор БД: SELECT, INSERT, UPDATE, DELETE, GRANT та DROP.

Основні клієнтські команди MySQL

В каталозі C:\xampp\mysql\bin є клієнтська (mysql.exe) та серверна (mysqld.exe) програми MySQL.

Клієнтська програма має інтерфейс командного рядка.

Щоб підключитися до сервера MySQL з використанням клієнта треба ввести команду

```
mysql [-h hostname] [-P portnumber] –u username –p
```

Після підключення до сервера треба вибрати БД

```
USE databasename;
```

Команда

```
SHOW TABLES;
```

дозволяє отримати список усіх таблиць.

Щоб швидко ознайомитися зі структурою однієї з таблиць БД можна скористатися командою

```
SHOW COLUMNS FROM tablename;
```

Для перегляду значень таблиці:

```
SELECT * FROM tablename
```

Інтерфейс командного рядка

```
C:\xampp\mysql\bin\mysql -u root
mysql>SHOW databases;
```

Результат показаний на Рис.20.

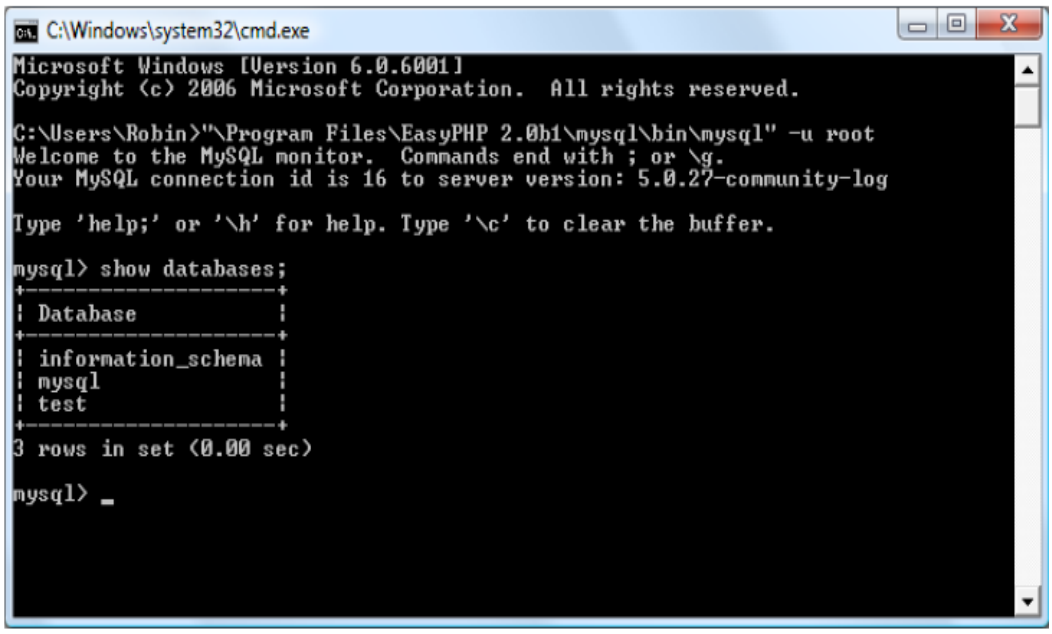

Рис.20. Інтерфейс командного рядка MySQL

Команди MySQL наведено на Рис.21.

Command	Action
ALTER	Alter a database or table
BACKUP	Back up a table
\c	Cancel input
CREATE	Create a database
DELETE	Delete a row from a table
DESCRIBE	Describe a table's columns
DROP	Delete a database or table
EXIT (CTRL-C)	Exit
GRANT	Change user privileges
HELP (\h, \?)	Display help
INSERT	Insert data
LOCK	Lock table(s)
QUIT (\q)	Same as EXIT

Command	Action
RENAME	Rename a table
SHOW	List details about an object
SOURCE	Execute a file
STATUS (\s)	Display the current status
TRUNCATE	Empty a table
UNLOCK	Unlock table(s)
UPDATE	Update an existing record
USE	Use a database

Рис.21. Команди MySQL

Приклад. Створення бази даних.

```
CREATE DATABASE publications;
USE publications;
GRANT ALL ON publications.* TO 'jim'@'localhost'
  IDENTIFIED BY 'mypasswd';
CREATE TABLE classics (
 author VARCHAR(128),
 title VARCHAR(128),
 category VARCHAR(16),
 year CHAR(4),
 isbn CHAR(13)) ENGINE MyISAM;
```

Типи даних MySQL

Типи даних MySQL показані на Рис.22.

Data type	Bytes used	Examples
CHAR(*n*)	Exactly *n* (<= 255)	CHAR(5) *"Hello" uses 5 bytes* CHAR(57) *"Goodbye" uses 57 bytes*
VARCHAR(*n*)	Up to *n* (<= 65535)	VARCHAR(7) *"Morning" uses 7 bytes* VARCHAR(100) *"Night" uses 5 bytes*

Data type	Bytes used	Examples
BINARY(*n*) or BYTE(*n*)	Exactly *n* (<= 255)	As CHAR but contains binary data
VARBINARY(*n*)	Up to *n* (<= 65535)	As VARCHAR but contains binary data

Data type	Bytes used	Attributes
TINYTEXT(*n*)	Up to *n* (<= 255)	Treated as a string with a character set
TEXT(*n*)	Up to *n* (<= 65535)	Treated as a string with a character set
MEDIUMTEXT(*n*)	Up to *n* (<= 1.67e+7)	Treated as a string with a character set
LONGTEXT(*n*)	Up to *n* (<= 4.29e+9)	Treated as a string with a character set

Data type	Bytes used	Attributes
TINYBLOB(*n*)	Up to *n* (<= 255)	Treated as binary data—no character set
BLOB(*n*)	Up to *n* (<= 65535)	Treated as binary data—no character set
MEDIUMBLOB(*n*)	Up to *n* (<= 1.67e+7)	Treated as binary data—no character set
LONGBLOB(*n*)	Up to *n* (<= 4.29e+9)	Treated as binary data—no character set

Data type	Bytes used
TINYINT	1
SMALLINT	2
MEDIUMINT	3
INT / INTEGER	4
BIGINT	8
FLOAT	4
DOUBLE / REAL	8

Data type
DATETIME
DATE
TIMESTAMP
TIME
YEAR

Рис.22. Типи даних MySQL

Графічний веб-інтерфейс для адміністрування бази даних MySQL

Графічним веб-інтерфейсом для адміністрування бази даних MySQL є phpMyAdmin. Це веб-застосунок з відкритим кодом, написаний на мові PHP. Він входе до складу пакета XAMPP і має вигляд, показаний на Рис.23.

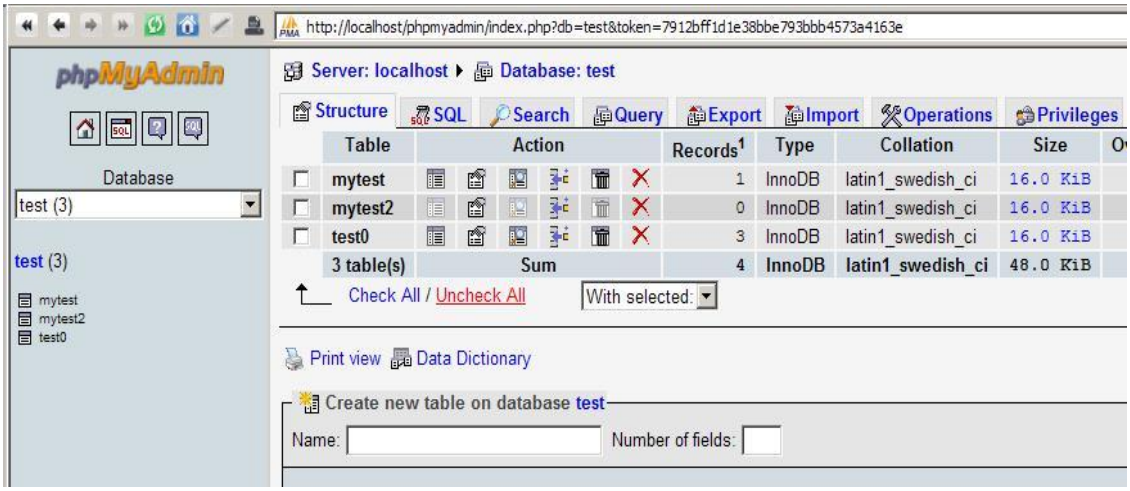

Рис.23. Вікно phpMyAdmin

Зв'язок PHP з MySQL

Існує три способи підключення до сервера MySQL з PHP:

- бібліотека MySQL;
- бібліотека MySQLi;
- бібліотека PDO.

Бібліотека MySQL є найстарішим способом підключення до бази даних і була введена в PHP 2.0. Її функції були мінімальними, і вона була заміщена бібліотекою MySQLi (“MySQL Improved”), починаючи з версії PHP 5.0 (випущеній в 2004 році).

Для підключення та запитування бази даних за допомогою старої бібліотеки MySQL використовувалися такі функції, як mysql_connect() і mysql_query(). Ці функції застаріли - це означає, що їх слід уникати, починаючи з PHP 5.5, і їх повністю вилучено з PHP 7.0.

Одна з основних змін у PHP 5.1 полягала в тому, що вона представила третю бібліотеку, PDO (PHP Data Objects) для підключення до баз даних MySQL (та інших БД).

Використання бібліотеки MySQLi

1) Процедурний стиль
З'єднання з MySQL-сервером:

$dbc = mysqli_connect($server, $username, $password, db_name);

Якщо ви тестуєте сайт на локальному комп'ютері і у вас встановлений пакет XAMPP, то параметри повинні бути наступними
$server = 'localhost';
$username = 'root';
$password = '';

Основні функції бібліотеки MySQLi

mysqli_connect() – З'єднання з сервером MySQL (повертає id з'єднання або NULL);
mysqli_real_escape_string(id, str) – пропукає спецсимволи;

mysqli_query($sql) – відправлення запиту до БД;
mysqli_num_rows($result) – кількість рядків, повернутих запитом;
mysqli_affected_rows(id) – кількість рядків SELECT, INSERT, UPDATE, REPLACE, чи DELETE запиту;
mysqli_result() – отримати потрібний елемент з набору записів;
mysqli_error($dbc) – рядок опису помилки MySQL;
mysqli_fetch_array – занести запис в асоц. масив або в числ. массив;
mysqli_fetch_assoc – занести запис в асоц. массив;
mysqli_fetch_row($result) – занести запис в массив;
mysqli_close(id) – закриття з'єднання з БД;
mysqli_list_dbs([id]) – повертає вказівник на масив з іменами БД;
mysqli_list_tables(ім'я_БД , [id]) – еквівалент SHOW TABLES.

Приклад: З'єднання та створення нової таблиці [5, Ullman]

```
<?php
if ($dbc = @mysqli_connect('localhost', 'root', '', 'myblog')) {
$query = 'CREATE TABLE entries (
id INT UNSIGNED NOT NULL  AUTO_INCREMENT PRIMARY KEY,
title VARCHAR(100) NOT NULL,
entry TEXT NOT NULL,
date_entered DATETIME NOT NULL
) CHARACTER SET utf8';
if (@mysqli_query($query, $dbc))
{
    print '<p> The table has been created.</p>';
} else {
    print '. . . . . .'
}
```

Приклад: Вставка даних в таблицю БД [6]
Форма для додавання даних показана на Рис.24.

Рис.24. Форма вставки даних в таблицю БД

```
<form action="add_entry.php" method="post">
<p>Entry Title: <input type="text" name="title" size="40" maxsize="100"></p>
```

```
     <p>Entry Text: <textarea name="entry" cols="40"
rows="5"></textarea></p>
     <input type="submit" name="submit" value="Post This Entry!">
     </form>
     .....
     if ($_SERVER['REQUEST_METHOD'] == 'POST') {
     $problem = FALSE;
     if (!empty($_POST['title']) && !empty($_POST['entry'])) {
      $title =  mysqli_real_escape_string($dbc,
trim(strip_tags($_POST['title'])));
      $entry = trim(strip_tags($_POST['entry'])); //можлива
ін’єкція! p.366
     } else {
          print '<p style="color: red;">Please submit both a
title and an entry.</p>';
          $problem = TRUE;
     }
     if (!$problem) {
     $dbc = mysqli_connect('localhost', 'root', '', 'myblog');
     . . . .    mysqli_set_charset($dbc, 'utf8');
     $query = "INSERT INTO entries (id, title, entry,
date_entered) VALUES (0, '$title', '$entry', NOW())";
     if (@mysqli_query($dbc, $query)) { . . . }
```

Приклад. Отримання даних з бази даних [6]

```
// Файл view_entries.php
$query = 'SELECT * FROM entries ORDER BY date_entered DESC';
if ($r = mysqli_query($dbc, $query)) {
     while ($row = mysqli_fetch_array($r)) {
     print "<p><h3>{$row['title']}</h3>
     {$row['entry']}<br>
     <a href=\"edit_entry.php?id={$row['id']}\">Edit</a>
     <a href=\"delete_entry.php?id={$row['id']}\">Delete</a>
     </p><hr>\n";
}
. . . . .
```

Результат показано на Рис.25.

Рис.25. Отримання даних з бази даних

Приклад. Видалення даних у базі даних [6]

При виборі Delete на сторінці view_entries.php треба вивести вибраний пост і запропонувати його видалити, як показано на Рис.26.

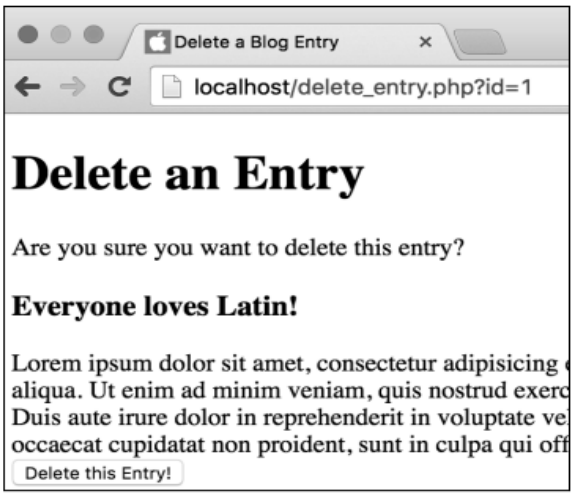

Рис.26. Вибір Delete на сторінці view_entries.php

Після видалення треба вивести сторінку (Рис.27).

Рис.27. Сторінка після видалення

Реалізація має наступний вигляд [6]:

```
     // Файл delete_entry.php
     $dbc = mysqli_connect( . . . . );
     if (isset($_GET['id']) && is_numeric($_GET['id']) ) {
     $query = "SELECT title, entry FROM entries WHERE
id={$_GET['id']}";
     if ($r = mysqli_query($dbc, $query)) {
     $row = mysqli_fetch_array($r);
     print '<form action="delete_entry.php" method="post">
     <p>Are you sure you want to delete this entry?</p>
     <p><h3>' . $row['title'] . '</h3>' .
     $row['entry'] . '<br>
     <input type="hidden" name="id" value="' . $_GET['id'] . '">
     <input type="submit" name="submit" value="Delete this
Entry!"></p>
     </form>';
     } else {     // Друкує помилку  }
     } elseif (isset($_POST['id']) && is_numeric($_POST['id'])) {
     $query = "DELETE FROM entries WHERE id={$_POST['id']} LIMIT
1";
      $r = mysqli_query($dbc, $query);
      if (mysqli_affected_rows($dbc) == 1) {
```

```
     print '<p>The blog entry has been deleted.</p>';
      } else {
          print '<p style="color: red;">Could not delete the blog
entry because:<br>' .
          mysqli_error($dbc) . '.</p><p>The query being run was:
' . $query . '</p>';
     }
     . . . . . .
```

Приклад. Оновлення даних у базі даних [6]

При виборі Edit на сторінці view_entries.php треба вивести вибраний пост і запропонувати його відредагувати (Рис.28).

Рис.28. Редагування даних

Після редагування треба вивести сторінку (Рис.29):

Рис.29. Сторінка після редагування

Реалізація має наступний вигляд [6]:

```
     // Файл edit_entry.php
     $dbc = mysqli_connect('localhost', 'root', '', 'myblog');
     mysqli_set_charset($dbc, 'utf8');
     if (isset($_GET['id']) && is_numeric($_GET['id']) ) {
     $query = "SELECT title, entry FROM entries WHERE
id={$_GET['id']}";
     if ($r = mysqli_query($dbc, $query)) {
     $row = mysqli_fetch_array($r);
     print '<form action="edit_entry.php" method="post">
      <p>Entry Title: <input type="text" name="title" size="40"
maxsize="100" value="' .
     htmlentities($row['title']) . '"></p>
     <p>Entry Text: <textarea name="entry" cols="40" rows="5">' .
htmlentities($row['entry']) . '</textarea></p>
     <input type="hidden" name="id" value="' . $_GET['id'] . '">
```

```
      <input type="submit" name="submit" value="Update this
Entry!">
     </form>';
     } else {   // Помилка }
     } elseif (isset($_POST['id']) && is_numeric($_POST['id'])) {
     $problem = FALSE;
      if (!empty($_POST['title']) && !empty($_POST['entry'])) {
      $title = mysqli_real_escape_string($dbc,
trim(strip_tags($_POST['title'])));
     $entry = mysqli_real_escape_string($dbc,
trim(strip_tags($_POST['entry'])));
     } else {
     print '<p style="color: red;">Please submit both a title and
an entry.</p>';
     $problem = TRUE;
     }
     if (!$problem) {
     $query = "UPDATE entries SET title='$title', entry='$entry'
WHERE id={$_POST['id']}";
      $r = mysqli_query($dbc, $query);
      if (mysqli_affected_rows($dbc) == 1) {
     print '<p>The blog entry has been updated.</p>';
     } else {    // Помилка }
     }
```

Використання бібліотеки MySQLi

2) Об’єктно-орієнтований стиль

З'єднання з MySQL-сервером шляхом створення об’єкта mysqli

```
$dbc = new mysqli('localhost', 'root', '', 'myblog');
```

Далі необхідно викликати методи об’єкта mysqli

Приклад [19]

Результат виведення інформації з бази даних показаний на Рис.30.

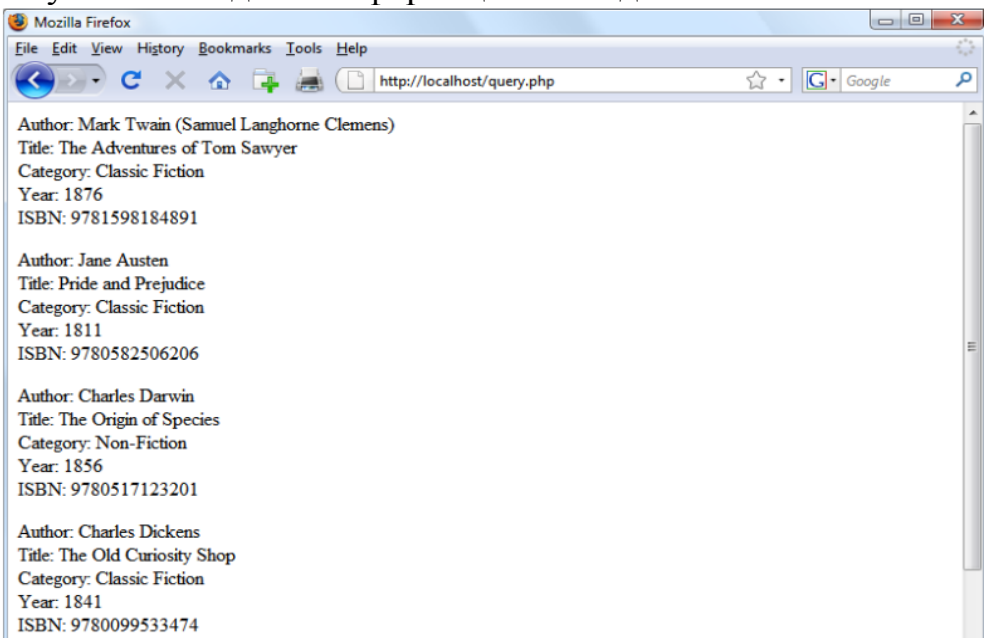

Рис.30. Виведення інформації з бази даних

Реалізація:

```
<?php // login.php
  $hn = 'localhost';
  $db = 'publications';
  $un = 'username';
  $pw = 'password';
?>
<?php //fetchrow.php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);
  $query  = "SELECT * FROM classics";
  $result = $conn->query($query);
  if (!$result) die($conn->error);
  $rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j){
          $result->data_seek($j);
          $row = $result->fetch_array(MYSQLI_ASSOC);
          echo 'Author: '   . $row['author']   . '<br>';
          echo 'Title: '    . $row['title']    . '<br>';
          echo 'Category: ' . $row['category'] . '<br>';
          echo 'Year: '     . $row['year']     . '<br>';
          echo 'ISBN: '     . $row['isbn']     . '<br><br>';
}
$result->close();
  $conn->close();
?>
```

Приклад [19]

Результат виведення інформації з бази даних з можливістю додавати та видаляти інформацію показаний на Рис.31.

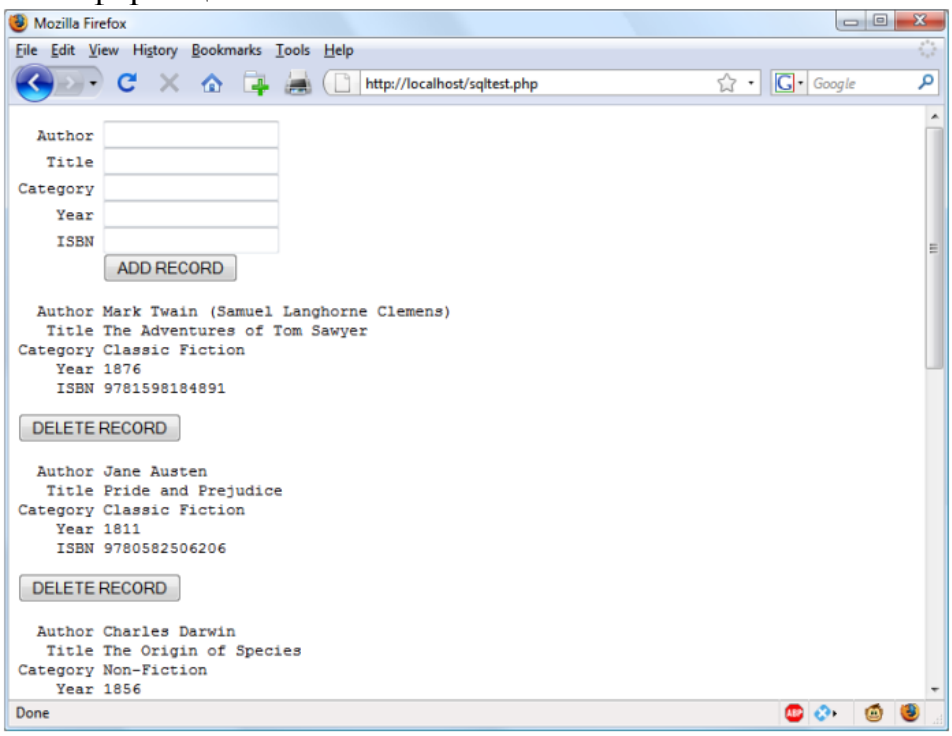

Рис.31. Додавання та видалення інформації

Реалізація:

```
<?php // sqltest.php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);
  if (isset($_POST['delete']) && isset($_POST['isbn']))
  {
    $isbn   = get_post($conn, 'isbn');
    $query  = "DELETE FROM classics WHERE isbn='$isbn'";
    $result = $conn->query($query);
    if (!$result) echo "DELETE failed: $query<br>" .
      $conn->error . "<br><br>";
  }
if (isset($_POST['author'])   &&
      isset($_POST['title'])    &&
      isset($_POST['category']) &&
      isset($_POST['year'])     &&
      isset($_POST['isbn']))
  {
    $author   = get_post($conn, 'author');
    $title    = get_post($conn, 'title');
    $category = get_post($conn, 'category');
    $year     = get_post($conn, 'year');
    $isbn     = get_post($conn, 'isbn');
    $query    = "INSERT INTO classics VALUES" .
      "('$author',    '$title',    '$category',    '$year',
'$isbn')";
    $result   = $conn->query($query);
    if (!$result) echo "INSERT failed: $query<br>" .
      $conn->error . "<br><br>";
  }
echo <<<_END
  <form action="sqltest.php" method="post"><pre>
    Author <input type="text" name="author">
     Title <input type="text" name="title">
  Category <input type="text" name="category">
      Year <input type="text" name="year">
      ISBN <input type="text" name="isbn">
           <input type="submit" value="ADD RECORD">
  </pre></form>
_END;
  $query  = "SELECT * FROM classics";
$result = $conn->query($query);
  if (!$result) die ("Database access failed: " . $conn-
>error);
  $rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)
  {
    $result->data_seek($j);
    $row = $result->fetch_array(MYSQLI_NUM);
    echo <<< _END
  <pre>
    Author $row[0]
```

```
      Title $row[1]
    Category $row[2]
        Year $row[3]
        ISBN $row[4]
    </pre>
    <form action="sqltest.php" method="post">
    <input type="hidden" name="delete" value="yes">
    <input type="hidden" name="isbn" value="$row[4]">
    <input type="submit" value="DELETE RECORD"></form>
  _END;
    }
  $result->close();
    $conn->close();
    function get_post($conn, $var)
    {
      return $conn->real_escape_string($_POST[$var]);
    }
  ?>
```

PHP Data Objects (PDO)

PHP Data Objects (PDO) — розширення для PHP, що надає розробнику простий і універсальний інтерфейс для доступу до різних баз даних.

PDO не використовує абстрактних прошарків для підключення до БД, на зразок ODBC, а використовує для різних БД їх «рідні» драйвери, що дозволяє добитися високої продуктивності.

PDO входить до складу PHP з версії 5.1.

Зв'язок PHP з MySQL через PDO

З’єднання

```
$host = 'localhost';
$db = 'myblog';
$user = 'root';
$pass = '';
$charset = 'utf8';
$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$opt = [
     PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
     PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
PDO::ATTR_EMULATE_PREPARES   => false,
];
$pdo = new PDO($dsn, $user, $pass, $opt);
```

PDO. Виконання запитів

1) Якщо в запит не передаються ніякі змінні, то можна скористатися методом query(). Він виконає запит і поверне спеціальний об'єкт - PDO statement.

Данні можна отримати за допомогою метода fetch()

```
$stmt = $pdo->query('SELECT name FROM users');
while ($row = $stmt->fetch())
{
```

```
        echo $row['name'] . "\n";
    }
```

2) Якщо ж в запит передається хоча б одна змінна, то цей запит повинен виконуватися тільки через підготовлені вирази (prepared statments).

Що це таке? Це звичайний SQL запит, в якому замість змінної ставиться спеціальний маркер - плейсхолдер.

PDO підтримує позиційні плейсхолдери (?), для яких важливий порядок переданих змінних, і іменовані (: name), для яких порядок не важливий. Приклади:

```
$sql = 'SELECT name FROM users WHERE email = ?';
$sql = 'SELECT name FROM users WHERE email = :email';
```

Щоб виконати такий запит, спочатку його треба підготувати за допомогою метода prepare (). Він також повертає PDO statement, але ще без даних.

Щоб їх отримати, треба виконати цей запит (метод execute()), попередньо передавши в нього змінні.

```
$stmt = $pdo->prepare('SELECT name FROM users WHERE email = ?');
$stmt->execute(array($email));
```

Інший варіант:

```
$stmt = $pdo->prepare(
'SELECT name FROM users WHERE email = :email');
$stmt->execute(array('email' => $email));
```

Приклад [26]

1) Створимо базу даних misc

2) Створимо таблицю users

// Файл misc.sql

```
USE misc;
CREATE TABLE users (
   user_id INTEGER NOT NULL AUTO_INCREMENT,
   name VARCHAR(128),
   email VARCHAR(128),
   password VARCHAR(128),
   PRIMARY KEY(user_id),
   INDEX(email)
) ENGINE=InnoDB CHARSET=utf8;
INSERT INTO users (name,email,password) VALUES
('Chuck','csev@umich.edu','123');
INSERT INTO users (name,email,password) VALUES
('Glenn','gg@umich.edu','456');
```

3) Файл pdo.php

```
<?php
$pdo = new PDO('mysql:host=localhost;dbname=misc', 'root',
'');
$pdo->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
```

4) Файл first.php

```
<?php
echo "<pre>\n";
```

```
require_once "pdo.php";
$stmt = $pdo->query("SELECT * FROM users");
while ( $row = $stmt->fetch(PDO::FETCH_ASSOC) ) {
    print_r($row);
}
echo "</pre>\n";?>  або
  $rows = $stmt>fetchAll(PDO::FETCH_ASSOC);
   print_r($rows);
```

Результат:

```
Array(
    [user_id] => 1
    [name] => Chuck
    [email] => csev@umich.edu
    [password] => 123
)

Array(
    [user_id] => 2
    [name] => Glenn
    [email] => gg@umich.edu
    [password] => 456
```

mysql> select * from users;

```
+---------+-------+----------------+----------+
| user_id | name  | email          | password |
+---------+-------+----------------+----------+
|       1 | Chuck | csev@umich.edu | 123      |
|       2 | Glenn | gg@umich.edu   | 456      |
+-------+-------+----------------+------------+
```

4) Файл user1.php

```
<?php
require_once "pdo.php";
if ( isset($_POST['name']) && isset($_POST['email'])
                           && isset($_POST['password'])) {
$sql   =   "INSERT   INTO   users   (name,   email,   password)
              VALUES (:name, :email, :password)";
echo("<pre>\n".$sql."\n</pre>\n");
    $stmt = $pdo->prepare($sql);
    $stmt->execute(array(
        ':name' => $_POST['name'],
        ':email' => $_POST['email'],
        ':password' => $_POST['password']));
}
?><html><head></head><body>
<p>Add A New User</p>
<form method="post">
<p>Name:<input type="text" name="name" size="40"></p>
<p>Email:<input type="text" name="email"></p>
<p>Password:<input type="password" name="password"></p>
<p><input type="submit" value="Add New"/></p>
</form>
</body>
```

Результат показаний на Рис.32

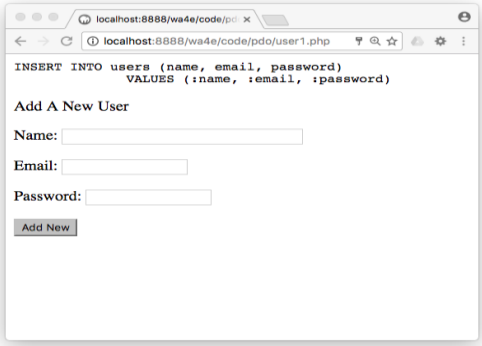

Рис.32. Додавання даних через PDO

Результат після додавання:

```
mysql> select * from users;
+---------+-------+----------------+----------+
| user_id | name  | email          | password |
+---------+-------+----------------+----------+
|       1 | Chuck | csev@umich.edu | 123      |
|       2 | Glenn | gg@umich.edu   | 456      |
|       3 | Sally | sally@uiuc.edu | 123      |
|       4 | Fred  | fred@umich.edu | YO       |
+---------+-------+----------------+----------+
```

5) Файл user2.php

```
. . . . . .
<html>
<head></head><body>
<table border="1">
<?php
$stmt = $pdo->query("SELECT name, email, password FROM
users");
while ( $row = $stmt->fetch(PDO::FETCH_ASSOC) ) {
    echo "<tr><td>";
    echo($row['name']);
    echo("</td><td>");
    echo($row['email']);
    echo("</td><td>");
    echo($row['password']);
    echo("</td></tr>\n");
}
?>
</table>
<p>Add A New User</p> …
```

В результаті отримаємо наступну сторінку (Рис.33):

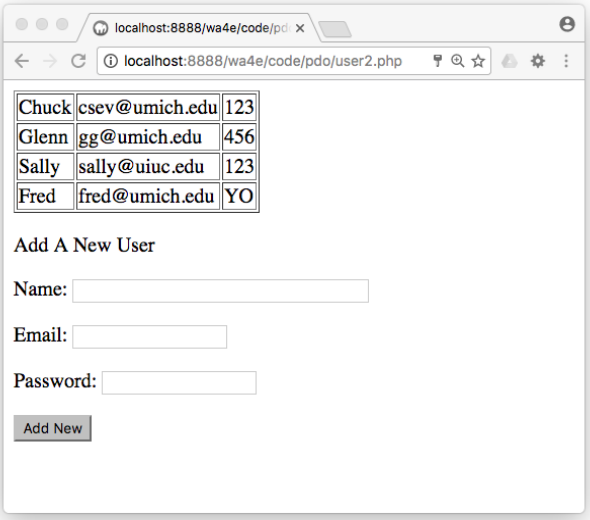

Рис.33. Сторінка після додавання даних

6)Файл user2del.php

Сторінка до видалення показана на Рис.34.

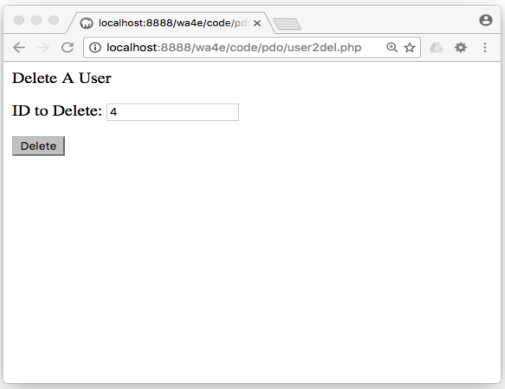

Рис.34. Сторінка до видалення

```
 <?php
require_once "pdo.php";

if ( isset($_POST['user_id']) ) {
    $sql="DELETE FROM users WHERE user_id = :zip";
    echo "<pre>\n$sql\n</pre>\n";
    $stmt = $pdo->prepare($sql);
    $stmt->execute(array(':zip'=>$_POST['user_id']));
}
?>
<p>Delete A User</p>
<form method="post"><p>ID to Delete:
<input type="text" name="user_id"></p>
<p><input type="submit" value="Delete"/></p>
</form>
```

Сторінка після видалення показана на Рис.35.

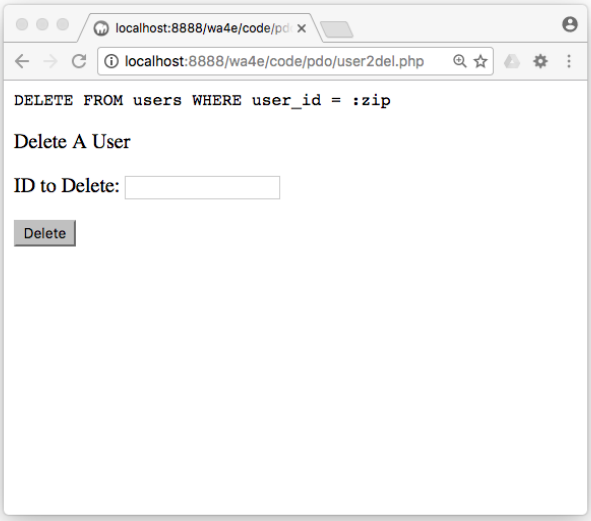

Рис.35. Сторінка після видалення

7)Файл user3.php

Форма з кнопкою видалення показана на Рис.36.

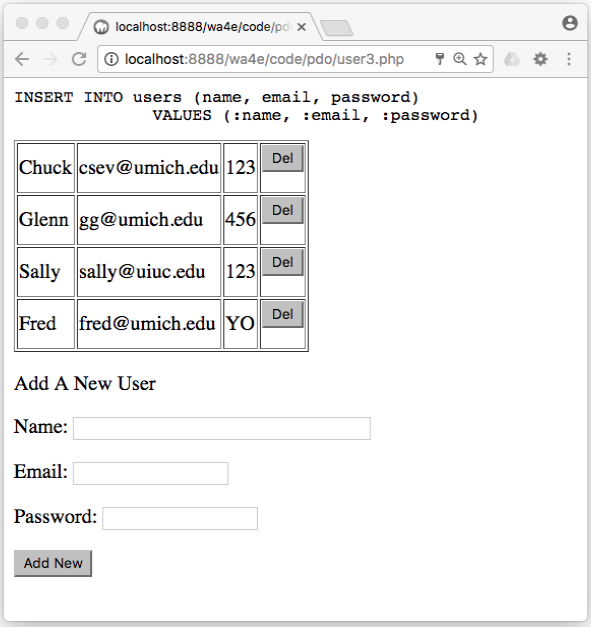

Рис.36. Форма з кнопкою видалення

```
<?php
require_once "pdo.php";
if ( isset($_POST['name']) && isset($_POST['email'])
      && isset($_POST['password'])) {
$sql = "INSERT INTO users (name, email, password)
VALUES (:name, :email, :password)";
echo("<pre>\n".$sql."\n</pre>\n");
$stmt = $pdo->prepare($sql);
$stmt->execute(array(
        ':name' => $_POST['name'],
        ':email' => $_POST['email'],
        ':password' => $_POST['password']));
```

```
    }
    if ( isset($_POST['delete']) && isset($_POST['user_id']) ) {
        $sql = "DELETE FROM users WHERE user_id = :zip";
        echo "<pre>\n$sql\n</pre>\n";
        $stmt = $pdo->prepare($sql);
        $stmt->execute(array(':zip' => $_POST['user_id']));
    }
    $stmt = $pdo->query("SELECT name, email, password, user_id
FROM users");
    $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
    ?>
    <html><head></head><body><table border="1">
    <?php
    foreach ( $rows as $row ) {
        echo "<tr><td>";
        echo($row['name']);
        echo("</td><td>");
        echo($row['email']);
        echo("</td><td>");
        echo($row['password']);
        echo("</td><td>");
        echo('<form method="post"><input type="hidden" ');
echo('name="user_id" value="'.$row['user_id'].'">'."\n");
echo('<input type="submit" value="Del" name="delete">');
echo("\n</form>\n");
        echo("</td></tr>\n");}?>
    </table>
    <p>Add A New User</p>
    <form method="post">
    <p>Name:<input type="text" name="name" size="40"></p>
    <p>Email:<input type="text" name="email"></p>
    <p>Password:<input type="password" name="password"></p>
    <p><input type="submit" value="Add New"/></p>
    </form>
    </body>
```

Тема 6. Робота з Cookies та сесіями. Робота з файлами. Завантаження файлів на сервер.

Cookies

Зберігати деякі відомості про користувачів, наприклад ім'я, число відвідувань, дату останнього відвідування, можна за допомогою cookies.

Cookie (англ. печиво) – являє собою невеликий пакет інформації (текстовий файл), який web-сервер може записати на клієнтській машині.

Життєвий цикл cookie виглядає так:

- Клієнт відправляє HTTP-запит серверу.
- Сервер відправляє HTTP-відповідь, яка серед іншого включає в себе заголовок Set-Cookie: var = value.

- При необхідності, клієнт переходить на іншу сторінку цього ж сервера, шляхом відправки нового HTTP-запиту, що включає в себе заголовок Cookie: var = value.
- Сервер «впізнає» клієнта і відповідним чином реагує на його запит, якщо це передбачено.

Життєвий цикл cookies показаний на Рис.37.

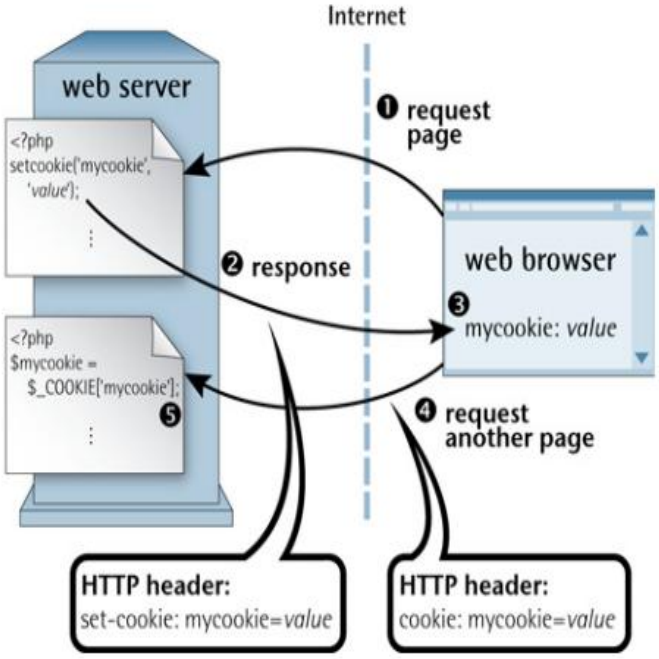

Рис.37. Життєвий цикл cookies

- Для безпеки прочитати cookie можна тільки з домена, в якому вони були створені.
- Крім того, у cookie є дата закінчення строку зберігання, після чого вони видаляються.
- Максимальний об'єм (обсяг) даних – 4Кб.

Установка cookie

setcookie(ім'я, значення, термін_зберіг, шлях, домен, режим)

Тільки перший параметр є обов'язковим.

Якщо термін не вказано, cookie-файл буде дійсним тільки протягом сеансу.

Доступ до cookie

Отримати значення cookie можна двома способами.

- Усі cookies доступні через змінну оточення $_COOKIE, як $_COOKIE[‘ім'я_cookie’]
- Суперглобальна змінна $_SERVER[‘HTTP_COOKIE’] містить значення будь-якого cookie.

Приклад

```
<?php   // Встановлюємо cookie
setcookie ("TestCookie", "value");//До кінця сеансу
setcookie ("TestCookie", "value", time()+3600);//На 1 годину
```

```
     setcookie ("TestCookieArray[1]", "value1");  //Массив
Cookie
     setcookie ("TestCookieArray[2]", "value2");
     ?>
```

Cookie стане доступною тільки після перезавантаження сторінки.

```
     <?php     // Читаємо cookie
          echo $_COOKIE['TestCookie'];
          echo $_COOKIE['TestCookieArray'][1];
     ?>

     <?php     // Видаляємо cookie
          setcookie   ("TestCookie");//Встановлюємо   куку   без
значення
          setcookie ("TestCookieArray[1]");
     ?>
```

Приклад. Використання Cookie при авторизації та реєстрації [20].

Створення БД сайту та таблиці користувачів.

Файл sql.sql

```
     CREATE DATABASE sitename;
     USE sitename;
     CREATE TABLE users (
     user_id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
     first_name VARCHAR(20) NOT NULL,
     last_name VARCHAR(40) NOT NULL,
     email VARCHAR(60) NOT NULL,
     pass CHAR(40) NOT NULL,
     registration_date DATETIME NOT NULL,
     PRIMARY KEY (user_id)
     );
     INSERT INTO users
     (first_name, last_name, email, pass, registration_date)
     VALUES ('Larry', 'Ullman', 'email@example.com',
SHA1('mypass'), NOW());
     INSERT INTO users VALUES
     (NULL, 'Zoe', 'Isabella', 'email2@example.com',
SHA1('mojito'), NOW());
     INSERT INTO users (first_name, last_name, email, pass,
registration_date) VALUES
     ('John', 'Lennon', 'john@beatles.com', SHA1('Happin3ss'),
NOW()),
     ('Paul', 'McCartney', 'paul@beatles.com', SHA1('letITbe'),
NOW()),
     ('George', 'Harrison', 'george@beatles.com ',
SHA1('something'), NOW()),
     ('Ringo', 'Starr', 'ringo@beatles.com', SHA1('thisboy'),
NOW());
```

Головний файл

```
     <?php # Script 3.4 - index.php
     $page_title = 'Welcome to this Site!';
     include ('./includes/header.html');
     ?>
```

```
<h1 id="mainhead">Big Header</h1>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<h2>Subheader</h2>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<?php
include ('./includes/footer.html');
?>
```

Головна сторінка сайту показана на Рис.38.

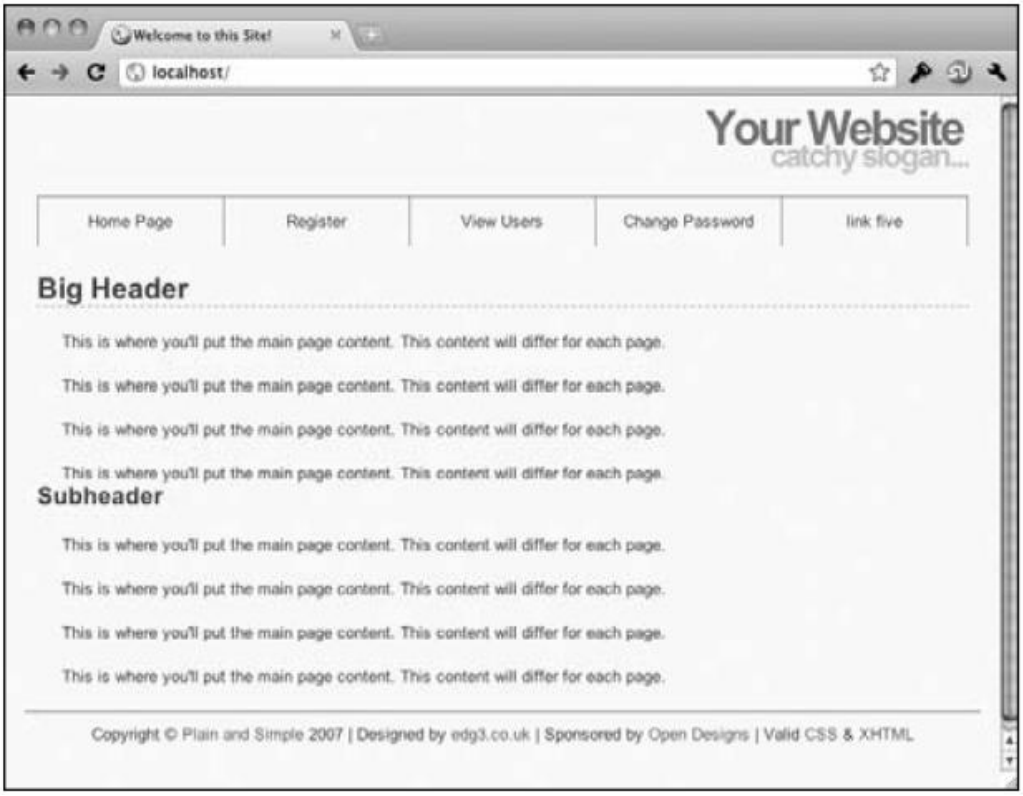

Рис.38. Головна сторінка сайту

Файл header.html

```
<html >
<head><title> <?php echo $page_title; ?> </title>
     <link rel="stylesheet" href="includes/style.css"
type="text/css" media="screen"/>
<meta http-equiv="content-type" content="text/html;
charset=utf-8"/>
</head>
<body>
<div id="header"><h1>Your Website</h1><h2>catchy
slogan...</h2></div>
<div id="navigation">
<ul>
     <li><a href="index.php">Home Page</a></li>
     <li><a href="register.php">Register</a></li>
```

```
        <li><a href="view_users.php">View Users</a></li>
    <li><a href="password.php">Change Password</a></li>
        <li> <?php  // Create a login/logout link:
    if ( (isset($_COOKIE['user_id'])) &&
(basename($_SERVER['PHP_SELF']) != 'logout.php') ){
    echo '<a href="logout.php">Logout</a>';
    } else {  echo '<a href="login.php">Login</a>';}
    ?></li></ul>
    </div>
    <div id="content">
     <!-- Start of the page-specific content. -->
```

Стилі для навігації

```
#navigation {
background:#fafafa;
border-right:1px solid #999;
margin:0 auto;
width:750px;
height:40px;
list-style:none;
}
#navigation li {
border-left:1px solid #999;
float:left;
width:149px;
list-style:none;
}
#navigation a {
color:#555;
display:block;
line-height:40px;
text-align:center;
}
#navigation a:hover {
background:#e3e3e3;
color:#555;
}
#navigation .active {
background:#e3e3e3;
color:#777;
}
```

Форма реєстрації показана на Рис.39

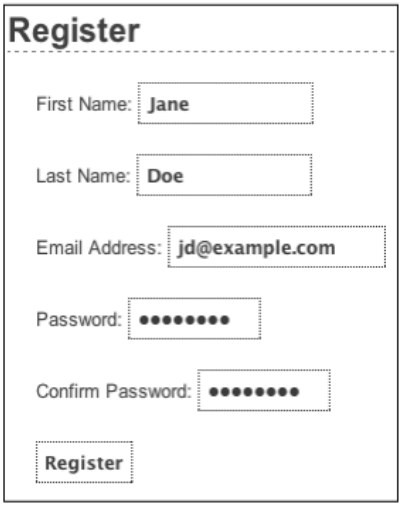

Рис.39. Форма реєстрації

Фрагменти файлу register.php

```
<?php
$page_title = 'Register';
include ('includes/header.html');
require ('mysqli_connect.php');
$errors = array(); // Initialize an error array.
     // Check for a first name:
     if (empty($_POST['first_name'])) {
          $errors[] = 'You  forgot  to  enter  your  first
name.';
     } else {
          $fn = mysqli_real_escape_string($dbc,
trim($_POST['first_name']));
     }
. . .
if (empty($errors)) {    // If everything's OK.
$q = "INSERT  INTO  users (first_name, last_name, email, pass,
registration_date) VALUES ('$fn', '$ln', '$e', SHA1('$p'), NOW() )";

          $r = @mysqli_query ($dbc, $q); // Run the query.
          if ($r) {   // If it ran OK.
                   echo  '<h1>Thank you!</h1><p>You are now
registered. </p>
                     <p><br /></p>';
          } else { // If it did not run OK.
          echo '<h1>System Error</h1>
          <p class="error">You could not be registered due to a
system error.</p>';
          echo '<p>' . mysqli_error($dbc) . '<br /><br />Query:
' . $q . '</p>
          }   // End of if ($r) IF.
          mysqli_close($dbc);    // Close the database
connection.
          include ('includes/footer.html');
          exit();
. . .
```

Файл mysqli_connect.php

```
<?php
DEFINE ('DB_USER', 'username');
DEFINE ('DB_PASSWORD', 'password');
DEFINE ('DB_HOST', 'localhost');
DEFINE ('DB_NAME', 'sitename');
// Make the connection:
$dbc = @mysqli_connect (DB_HOST, DB_USER, DB_PASSWORD,
DB_NAME)
     OR die ('Could not connect to MySQL: ' .
mysqli_connect_error() );
// Set the encoding...
mysqli_set_charset($dbc, 'utf8');
```

Фрагменти файлу login_page.inc.php

```
<?php
$page_title = 'Login';
include ('includes/header.html');
// Print any error messages, if they exist:
```

```
     if (isset($errors) && !empty($errors)) {
     echo '<h1>Error!</h1>
     .  .  .  .  .  .  .
     }
     // Display the form:
     ?>
     <h1>Login</h1>
     <form action="login.php" method="post">
     <p> Email Address: <input type="text" name="email" size="20"
maxlength="60" /> </p>
     <p>Password: <input type="password" name="pass" size="20"
maxlength="20" /></p>
     <p><input type="submit" name="submit" value="Login" /></p>
     </form>
     <?php include ('includes/footer.html'); ?>
```

Фрагменти файлу login.php

```
     <?php
     require ('includes/login_functions.inc.php');
     require ('../mysqli_connect.php');
     list ($check, $data) = check_login($dbc,
$_POST['email'],$_POST['pass']);
     if ($check) {
          setcookie ('user_id', $data['user_id']);
          setcookie ('first_name', $data['first_name']);
          redirect_user('loggedin.php');
     } else {
          $errors = $data;
     }
     mysqli_close($dbc);
     include ('includes/login_page.inc.php');
     ?>
```

Фрагменти файлу login_functions.inc.php

```
     <?php
     function redirect_user($page = 'index.php') {
     $url = 'http://' . $_SERVER['HTTP_HOST']
dirname($_SERVER['PHP_SELF']);
     $url = rtrim($url, '/\\');   $url .= '/' . $page;
     // Redirect the user:
     header("Location: $url");
     exit(); // Quit the script.
     }
     function check_login($dbc, $email = '', $pass = '') {
     $errors = array();
     if (empty($email)) { . . . } if (empty($pass)) { .  .  .}
if (empty($errors)) {
     $q = "SELECT user_id, first_name FROM users WHERE email='$e'
AND pass=SHA1('$p')";
     $r = @mysqli_query ($dbc, $q);
     if (mysqli_num_rows($r) == 1) {
     $row = mysqli_fetch_array ($r, MYSQLI_ASSOC);
     return array(true, $row);
     } else { . . .} } return array(false, $errors); }
```

Робота з сесіями

- Сесії (session) в PHP призначені для зберігання на сервері даних при переходах користувача між різними сторінками веб-сайту.
- Сесією (сеансом) називають період часу, який користувач проводе на сайті.
- При відкритті сесії користувачу присвоюється його унікальний ідентифікатор сеансу (SID), який зберігається на боці клієнта у вигляді cookie.
- Якщо підтримка cookie відключена, то SID автоматично додається до URL на даному сайті.
- У той же час web-сервер зберігає цей SID на своєму боці та записує сесійні дані в файли на своєму жорсткому диску.

Ілюстрація механізму сесій показана на Рис.40.

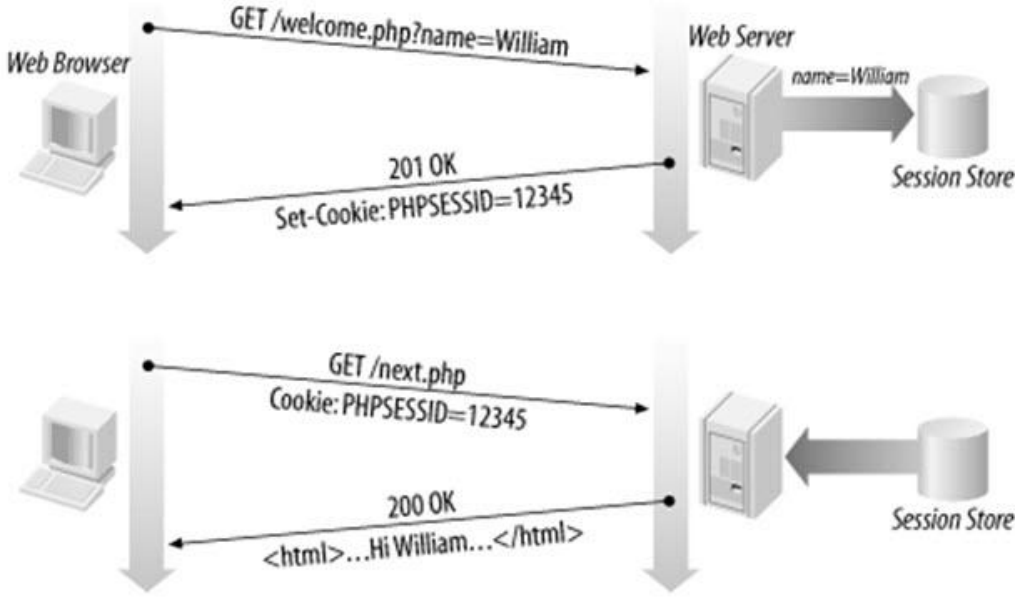

Рис.40. Ілюстрація механізму сесій

Відслідковування сеанса в PHP

Воно не стартує автоматично. Щоб почати сесію, треба застосувати функцію session_start(). Після цього усі сеансові змінні будуть зберігатися в масиві $_SESSION.

Приклад.

```
<?php //Файл index.php
     session_start(); //Начало сесії
     if(isset($_POST['login']) &&
isset($_POST['password'])){
          $_SESSION['auth_user']=htmlspecialchars(
$_POST['login']);
          header("Location: secret.php");
     }else{?>
          <form method="POST" action="index.php">
               <input type="text" name="login"><br>
               <input type="password" name="password"><br>
```

```
                <input type="submit" value=«Відправити">
            </form>
<?   }
?>
<?php //Файл secret.php
     session_start();
     if (!isset($_SESSION['auth_user'])){
          header("Location: index.php");
     } else{
      //Використання сесії
          $user = $_SESSION['auth_user'];
          echo «Привіт, $user! Це секретна частина сайту";
     }
?>
```

Робота з файлами

Для роботи з файлами є дві основні групи функцій: ті, що працюють з файловим ресурсом, і ті, що працюють з іменем файлу.

Пам'ятайте, що ресурс – це тип змінної, який не можна зберігати безпосередньо в PHP. Файловий ресурс – це вказівник (хендл) файлової системи операційної системи.

Всі функції, що стосуються файлових ресурсів, починаються з літери f і потім мають дієслово, що описує їхню функцію. Наприклад, *fopen()* відкриває файловий ресурс.

Функції, які працюють з рядком імені файлу, всі починаються з слова file, після чого йде дієслово, що описує те, що функція робить. Наприклад, *file_get_contents()* бере рядок назви файлу і повертає вміст цього файлу.

Відкриття файлів

Функція fopen() використовується для відкриття файлів. Вона повертає ресурсну змінну, яка є вказівником (хендлом) файлу.

Ви повинні передати два параметри fopen ():

• Назва файлу у вашій файловій системі.

• Режим файлу, з яким ви хочете його відкрити.

Режими файлу

r — відкрити файл лише для читання;

r+ — відкрити файл для читання і запису;

w — відкрити файл тільки для запису. Якщо він існує, то поточний вміст файлу знищується;

w+ — відкрити файл для читання і для запису. Якщо він існує, вміст файлу знищується. Позиція встановлюється в початок;

a — відкрити файл для запису. Позиція встановлюється в кінець;

a+ — відкрити файл для читання і запису. Поточна позиція встановлюється в кінець файлу;

b — обробляти бінарний файл. Цей прапор необхідний при роботі з бінарними файлами в ОС Windows.

Читання файлів

Ви можете читати з файлового ресурсу за допомогою функції fread().

```
<?php
$handle = fopen('info.txt', 'r');
while (!feof($handle)) {
echo fread($handle,1024);
}
```

Ось ще чотири PHP-функції, які полегшують читання файлів:

- fgetcsv() – прочитати рядок із файлу з заданим вказівником та проаналізувати поля для полів CSV.
- file_get_contents() – взяти файл з заданим іменем та прочитайте його вміст в рядок.
- readfile() – прочитати файл з заданим іменем та записати вміст у вихідний буфер.
- file() – прочитати цілий файл у масив.

Запис у файли

Запис у файл виконується за допомогою функції fwrite(), яка є binary-safe. fputs () є псевдонімом для цієї функції.

Функція fwrite() приймає два параметри: файловий ресурс для запису, і рядок для запису в файл.

Існує записуючий аналог для функції fgetcsv(), а саме fputcsv(), яка форматує масив як CSV і записує рядок у файл. Окрім параметрів файлового ресурсу та масиву, потрібні ще необов'язкові параметри для визначення формату CSV.

Якщо ви хочете записати форматовані рядки до файлу, ви повинні використовувати fprintf(), яка працює як команда printf().

Якщо ви хочете скинути вміст файлу до підключеного клієнта, ви можете використовувати fpassthru(). Ця функція почне з поточної позиції файлу та запише решту файлу у вихідний буфер.

Нарешті, є зручна функція, щоб швидко записати рядок у файл. Функція file_put_contents() не вимагає надання файловому ресурсу і просто потребує ім'я файлу та рядок, який ви хочете записати.

Ось простий приклад використання деяких з цих функцій:

```
<?php
$filename = 'test.csv';
$dataString = '1,2,3,4,5';
file_put_contents($filename, $dataString);
$handle = fopen($filename, 'r');
$myData = fgetcsv($handle);
echo gettype($myData); // array
echo count($myData); // 5
```

Функції файлової системи

PHP має великий список функцій, які з'єднують вас із файловою системою. Ми розглянемо декілька з них у цьому розділі.

Директорії

Ця група функцій дозволяє вам створювати, видаляти каталоги та переходити в каталоги.

- chdir() – змінює поточний робочий каталог PHP.
- chroot() – змінює кореневий каталог поточного процесу у вказаний каталог та встановлює робочий каталог PHP на /.
- rmdir() – видаляє каталог.
- readdir() – повертає ім'я наступного запису в хендлі каталогу, переданого як параметр. Записи повертаються в тому порядку, в якому вони зберігаються файловою системою.

Інформація про файл

PHP надає функцію finfo_open(), яка повертає новий екземпляр ресурсу файлової інформації. Ви надаєте йому два параметри - попередньо визначену константу опції та рядок з розташуванням файлу магічної бази даних.

Файл магічної бази даних - це формат, який використовується для опису типів файлів, він також використовується стандартною командою Unix file. Якщо ви не надаєте шлях до магічної бази даних, PHP використовуватиме той, разом з яким він поставляється.

Коли PHP знає, як ідентифікувати файли, ви можете використовувати функцію finfo_file() для отримання інформації про файл. Вона приймає принаймні два параметри – файл файлової інформації, який ви щойно створили, та ім'я рядка файлу, який ви хочете перевірити.

Ось приклад з керівництва PHP:

```
<?php
$finfo = finfo_open(FILEINFO_MIME_TYPE);
foreach (glob("*") as $filename) {
echo finfo_file($finfo, $filename) . "\n";
}
finfo_close($finfo);
```

Обидві функції мають об'єктно-орієнтовані стилі використання, як це показано в цьому прикладі з керівництва PHP:

```
<?php
// finfo will return the mime type
$finfo = new finfo(FILEINFO_MIME, "/usr/share/misc/magic");
/* get mime-type for a specific file */
$filename = "/usr/local/something.txt";
echo $finfo->file($filename);
```

Завантаження файлів на сервер

Знадобиться HTML-форма (index.html) та скрипт upload.php для її обробки.

```
//Файл index.html
<html><head><title> Завантаження файлів на сервер
</title>    </head>
<body>
<h2><b> Форма для завантаження файлів </b></h2>
<form action= "upload.php"  method="post"
```

```
                                enctype="multipart / form-data">
<input  type= "file"  name="filename"><br>
<input  type=" submit" value="Завантажити"><br>
</ form>
< /body>
</html>
```

Будемо мати наступну сторінку (Рис.41).

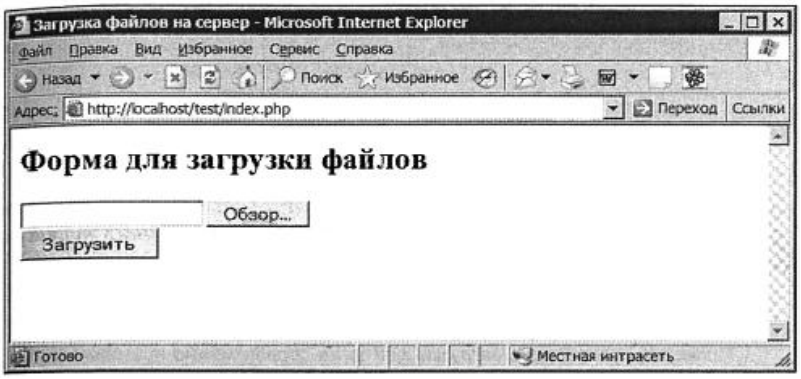

Рис.41. Форма завантаження файлів на сервер

Атрибут форми enctype визначає тип передачі даних. За замовчуванням цей атрибут має значення application/x-www-form-urlencoded, що не дозволяє завантажувати.

Елемент введення цієї форми повинен мати тип file.

Після отримання HTTP-запиту вміст завантажуваного файлу записується в тимчасовий файл, який створюється на сервері в каталозі, заданому за замовчуванням для тимчасових файлів, якщо в файлі php.ini не задано інший каталог (директива upload_tmp_dir).

Характеристики завантаженого файлу доступні через двовимірний суперглобальний масив $_FILES:

$_FILES[‘filename’][‘name’] – ім'я файлу на клієнтській машині.

$_FILES[‘filename’][‘size’] – розмір завантаженого файлу в байтах.

$_FILES[‘filename’][‘type’] – MIME-тип файлу.

$_FILES[‘filename’][‘tmp_name’] – ім'я тимчасового файлу, в якому зберігається завантажений файл.

Код скрипта обробки форми (upload.php)

```
    <html> <head><title>Результат завантаження файлу</title>
</head>
     <body>
    <?php
        if( $_FILES["filename"]["size"] > 1024*3*1024){
          echo ("Розмір файлу перевищує три мегабайти");
    exit;    }
    //  Перевіряємо  чи  завантажений  файл
    if(is_uploaded_file($_FILES["filename"]["tmp_name"])){
    //  Якщо файл завантажений успішно, переміщаємо його
    //  з тимчасової директорії в кінцеву
    move_uploaded_file($_FILES["filename"]["tmp_name"],
     "/path/to/file/".$_FILES["filename"]["name"]);
    } else {
```

```
echo(" Помилка завантаження файлу ");
}
?>
 </body>
</html>
```

Регулярні вирази (regex) в PHP

Regex представляють собою шаблони для пошуку в рядках.
Є два класи regex:

1. Regex, які відповідають стандарту POSIX.
2. Perl-сумісні regex (Perl-Compatible Regular Expressions, PCRE).

Мова PHP дозволяє використовувати Perl-сумісні regex.

При вивченні регулярних виразів корисно знайти сайт для онлайн-тестування регулярних виразів, наприклад https://regex101.com/

Обмежувачі

Регулярні вирази обмежуються символами, які з'являються на початку і в кінці кожного шаблону у вашому виразі. Зазвичай використовується пряма коса риска, але # і ! також є загальними.

Метасимволи в регулярних виразах

\ - Загальний екрануючий символ
^ - Декларує початок даних
$ - Декларує кінець даних
. - Відповідає будь-якому символу, крім нового рядка
[- Початок опису символьного класу
] - Кінець опису символьного класу
| - Початок гілки умовного вибору
(- Початок підмаски
) - Кінець підмаски
? - Ноль або одне входження, квантіфікатор жадібності
* - Квантіфікатор, що означає нуль або більше входжень
+ - Квантіфікатор, що означає одне або більше входжень
{ - Початок кількісного квантіфікатора
} - Кінець кількісного квантіфікатора

Загальні типи символів

Regex пропонує вам спосіб вказувати, що символ у вашому рядку пошуку може бути будь-яким з певного типу. Ви вказуєте їх, використовуючи метасимвол зворотної риски (Escape), а потім надаючи літеру типу. Загальні типи символів наведені в Таблиці 2.

Таблиця 2. Загальні типи символів

Символ	Тип символів
\d	Будь-яка десяткова цифра
\h	Будь-який горизонтальний символ пробілу

\s	Будь-який символ пробілу
\v	Будь-який вертикальний символ пробілу
\w	Будь-який символ, який утворює "слово"
\D	Будь-який символ, який не є десятковою цифрою
\H	Символ, який не є горизонтальним пробілом
\S	Будь-який символ, який не є пробілом
\V	Символ, який не є символом вертикального пробілу
\W	Будь-який символ "не слова"

Межі (Boundaries)

Символ "слово" - це будь-яка літера, цифра або символ підкреслення.

Межа слова - це позиція в рядку, де починається або закінчується слово.

Символи межі наведені в Таблиці 3.

Таблиця 3. Символи межі

Символ	**Межа**
\b	Межа слова
\B	Не межа слова
\A	Початок суб'єкта
\Z	Кінець суб'єкта або рядка на кінці
\z	Кінець суб'єкта
\G	Перше узгоджене положення в суб'єкті

Класи символів

- Ви створюєте клас символів, помістивши його в квадратні дужки.
- Прикладом класу символів є **[A-Z],** що означає будь-яку букву у верхньому регістрі.
- Ви також можете використовувати всі загальні типи в класах символів, тому
  - **[A-Z \d]** відповідатиме будь-якій великій літері, а також будь-якій цифрі.
- Частина шаблону, укладена в квадратні дужки, називається символьним класом.
- Усередині символьних класів використовуються наступні метасимволи:
  \ - Загальний екрануючий символ.
  ^ - Означає заперечення класу, допустимо тільки на початку класу.
  - - Означає символьний інтервал.

Збіг більше ніж один раз

- Шаблон **/[A-Z\d]+/** , застосований для рядка "abc123ABCabc" буде відповідати "**123ABC**“ символам.
- Шаблон [A-Z\d]{3} виділить 123
- Шаблон [A-Z\d]{3,} виділить 123ABC
  Шаблон [A-Z\d]{3,5} виділить 123AB

Виділення груп

- Групи позначають круглими дужками і це дозволяє застосувати квантифікатор до групи.
- Можна також створювати нумеровані групи, які зберігають співставне значення, і на них можна посилатися у будь-якому іншому місці вашого виразу.

```
<?php
$subject = "I can haz Cheeseburgers";
$pattern = "/I can haz (Cheeseburger)?/";
$matches = [ ];
preg_match($pattern, $subject, $matches);
var_dump($matches[0]);
```

Буде виведено рядок "I can haz Cheeseburger"

Жадібність і лінивість (Greed and Laziness)

За замовчуванням, відповідність є "жадібною“

```
<?php
$subject = "Some <strong>html</strong> text";
$pattern = "/<.*>/";
$matches = [];
preg_match($pattern, $subject, $matches);
var_dump($matches[0]);
```

Виведе: "html“

Квантифікатор ***** є “жадібним” і шукає саму довгу послідовність

- Навпаки, лінивий пошук повертає найкоротше можливе співпадання.
- Ви можете змінити квантор, щоб зробити його “лінивим”, додавши до нього знак питання (?).

```
<?php
$subject = "Some <strong>html</strong> text";
$pattern = "/<.*?>/";  // note the pattern has changed
$matches = [];
preg_match($pattern, $subject, $matches);
var_dump($matches[0]);  // string(8) "<strong>"
```

Отримання всіх збігів

```
<?php
$subject = "Some <strong>html</strong> text";
$pattern = "/<.*?>/";
$matches = [];
preg_match_all($pattern, $subject, $matches);
var_dump($matches);
Надрукує
array(1) {
     [0] =>
          array(2) {
               [0] => string(8) "<strong>"
               [1] => string(9) "</strong>"
          }
}
```

Шаблони завжди мають вигляд /pattern/, тобто в мові PHP шаблон треба задавати так: $p = ‘/pattern/’;

Найбільш вживаною функції є preg_match().

Синтаксис:

int preg_match (string pattern , string subject [, array matches])

Ця функція шукає в рядку subject відповідність регулярному виразу pattern. Якщо встановлено необов’язковий параметр matches, то результати пошуку поміщаються в масив matches.

Наприклад:

```
<?php
     $email = 'borysiv@gmail.com';
     if (!preg_match("/^[\ a-z0-9._-]+@[a-z0-9.-]+\.[a-
z]{2,6}$/i", $email)) {
          echo 'Некоректно введений email';
     }
?>
```

Функція preg_replace()

Синтаксис:

preg_replace (pattern , replacement , subject [, limit])

Ця функція шукає в рядку subject відповідності регулярному виразу pattern, і замінює їх на replacement. Необов’язкового параметр limit задає число відповідностей, які треба замінити. Якщо цей параметр не вказаний, або дорівнює -1, то замінюються всі знайдені відповідності.

Наприклад:

```
<? php
$str = "19 травня  1990";
 $pattern = "/([0-9]{4})/i";
 $replacement = "$1 року";
// $1 посилання на результат, знайдений першими круглими дужками
print preg_replace($pattern, $replacement, $str);
?>
```