Курс “Програмування Інтернет”

- Лекцій – 24 години

Лабораторних робіт – 24 годин

Самостійна робота – 102 годин

Кредитів - 5

Іспит

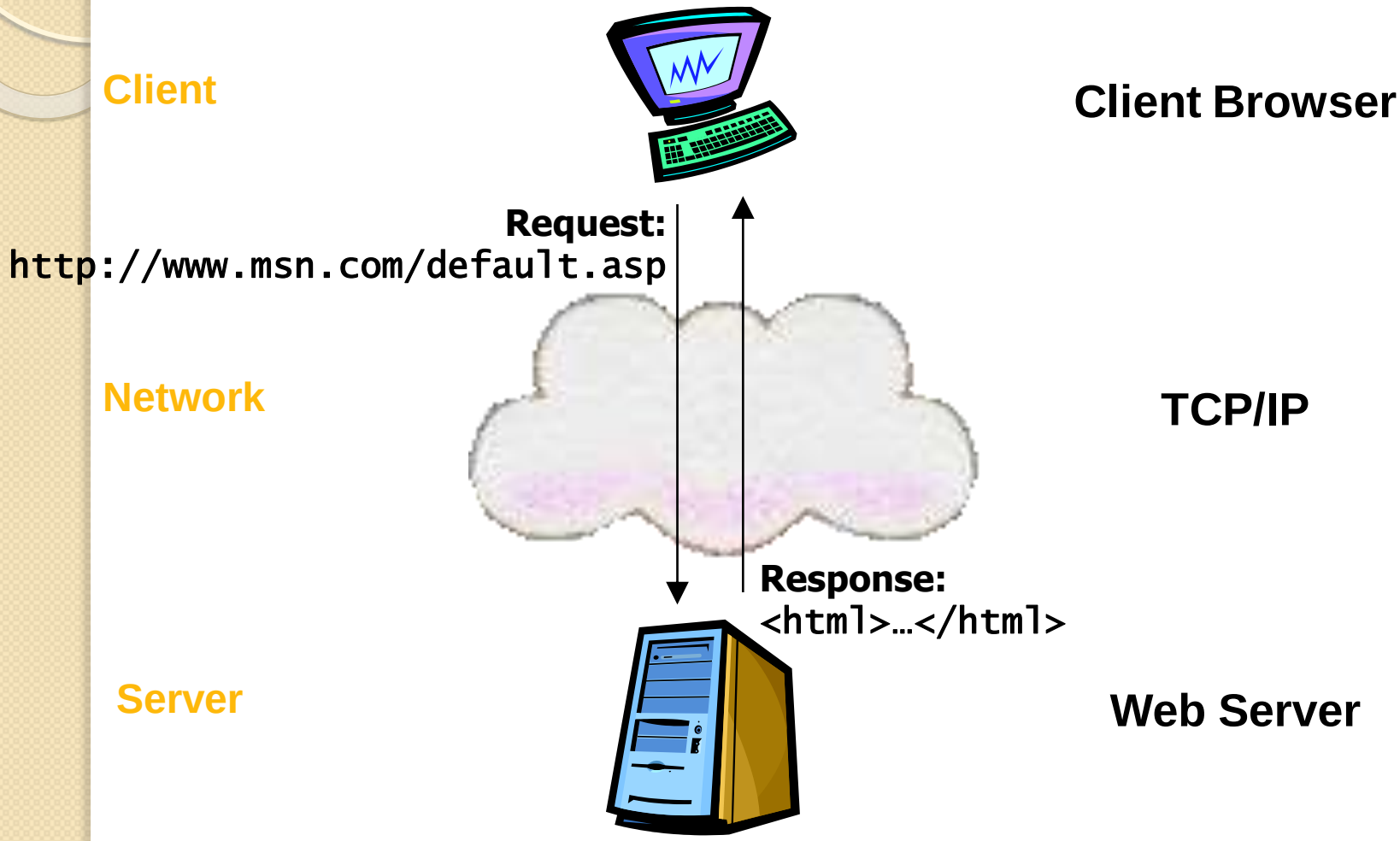
Мета – отримання теоретичних знань і практичних навичок з програмування для Інтернет, вивчення мов PHP, JavaScript, Python, використання сервера баз даних MySQL для розробки Web-додатків.

Лектор: доцент кафедри ПЗАС ЗДІА Попівщий Василь Іванович

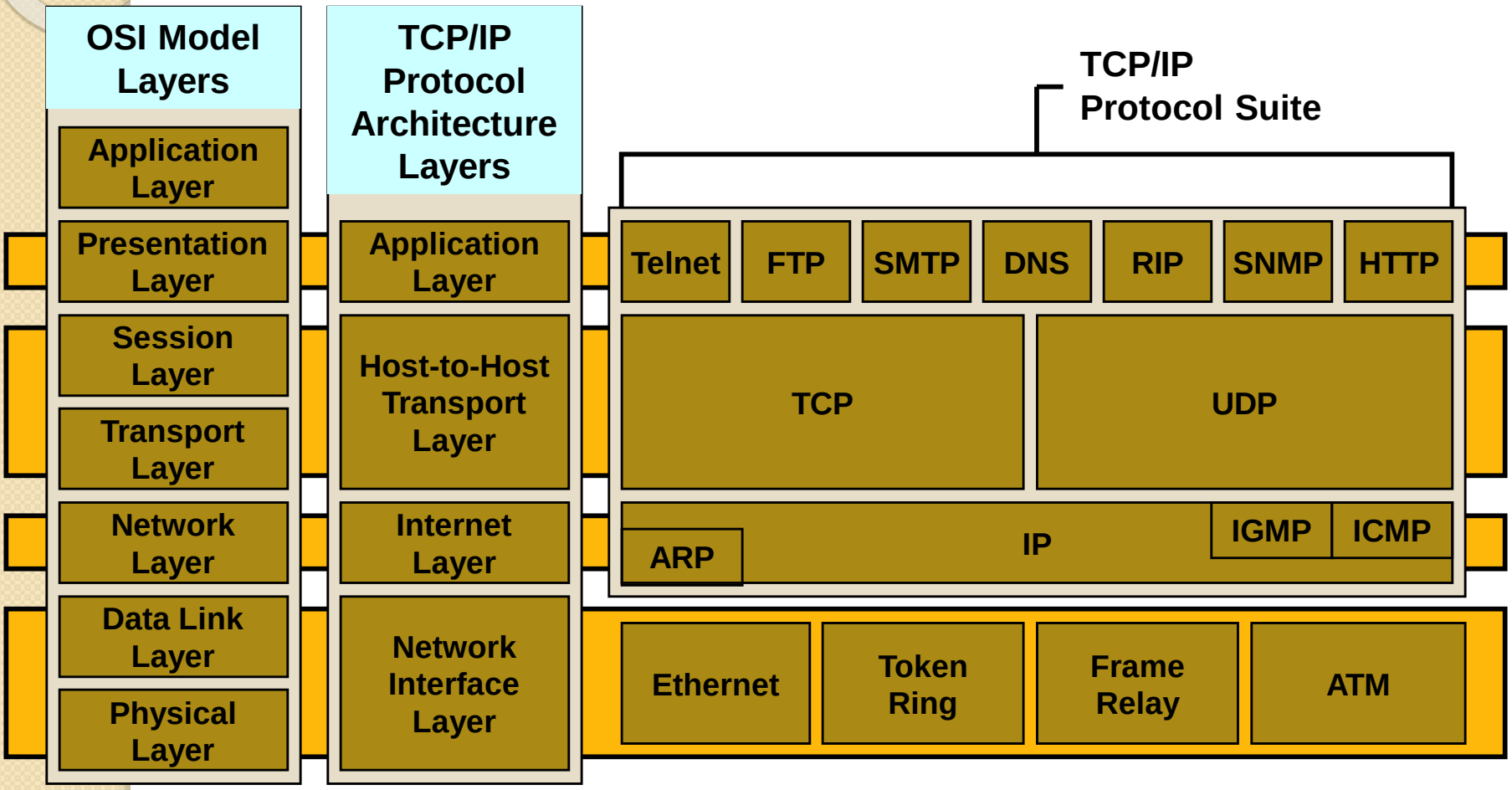
Інтернет та WEB

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

WWW архітектура



Мережні протоколи



Web-програмування

- **Веб-програмування - розділ програмування, орієнтований на розробку динамічних Internet-додатків**

- **Client-Side Programming**
  - **JavaScript**
  - Dynamic HTML
  - .Net controls (ASP.NET), ActionScript, Silverlight

- **Server-Side Programming**
  - **PHP**, Perl, Ruby, Python, . . . script
  - Server components
  - C# code-behind (ASP.NET)
  - JSP (Java)
  - Web controls used on ASPX pages
  - Web services

TIOBE Programming Community Index for January 2014 (www.tiobe.com)

Jan 2014	Jan 2013	Change	Programming Language	Ratings	Change
1	1		C	17.871%	+0.02%
2	2		Java	16.499%	-0.92%
3	3		Objective-C	11.098%	+0.82%
4	4		C++	7.548%	-1.59%
5	5		C#	5.855%	-0.34%
6	6		PHP	4.627%	-0.92%
7	7		(Visual) Basic	2.989%	-1.76%
8	8		Python	2.400%	-1.77%
9	10	↑	JavaScript	1.569%	-0.41%
10	22	⇈	Transact-SQL	1.559%	+0.98%
11	12	↑	Visual Basic .NET	1.558%	+0.52%
12	11	↓	Ruby	1.082%	-0.69%
13	9	⇊	Perl	0.917%	-1.35%
14	14		Pascal	0.780%	-0.15%
15	17	↑	MATLAB	0.776%	+0.14%
16	45	⇈	F#	0.720%	+0.53%

TIOBE Index for February 2016

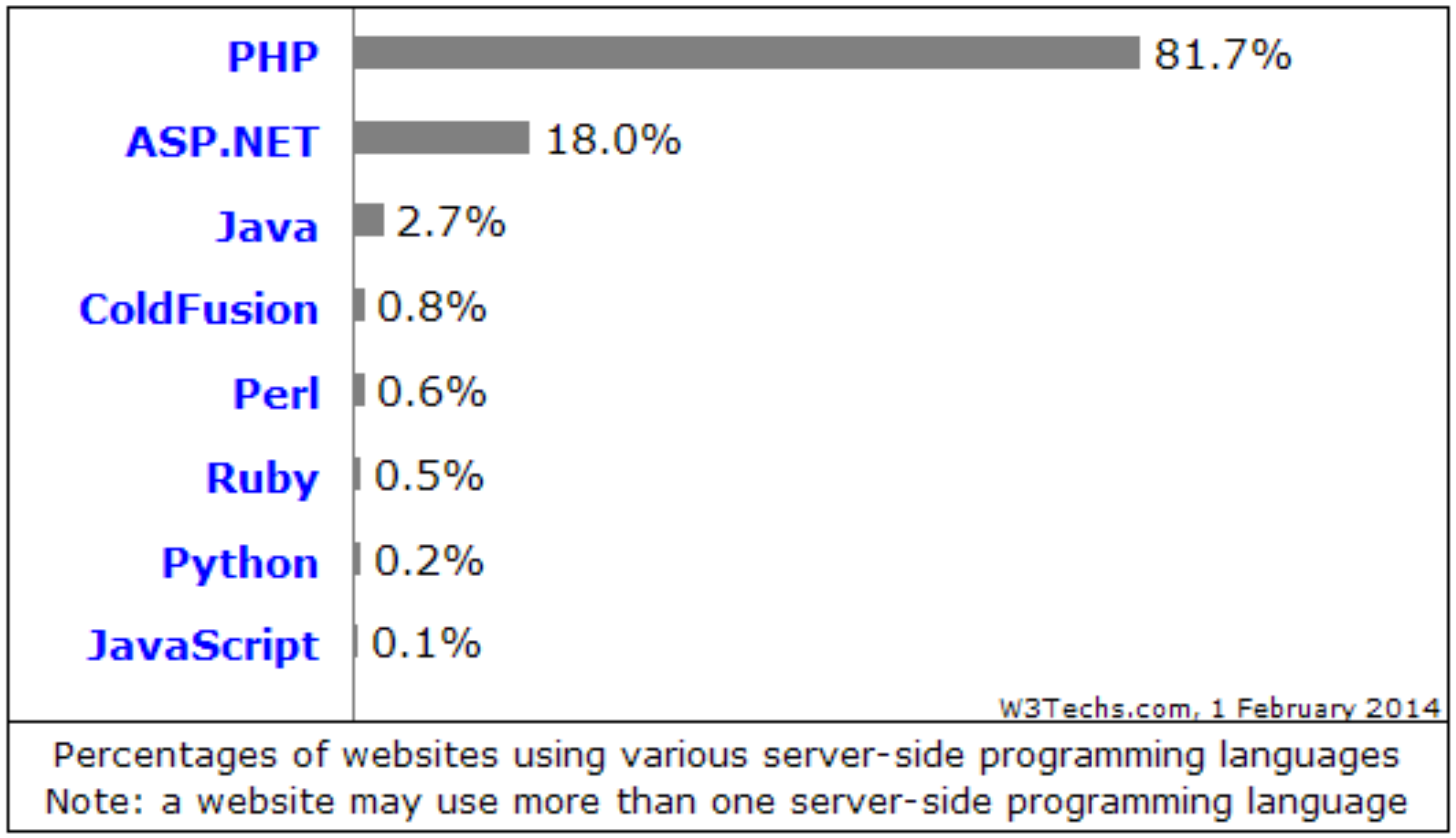
Feb 2016	Feb 2015	Change	Programming Language
1	2	↑	Java
2	1	↓	C
3	3		C++
4	5	↑	C#
5	8	↑	Python
6	7	↑	PHP
7	9	↑	Visual Basic .NET
8	12	⇈	Perl
9	6	↓	JavaScript
10	11	↑	Delphi/Object Pascal
11	**20**	⇈	**Ruby**
12	10	↓	Visual Basic
13	26	⇈	Assembly language
14	4	⇊	Objective-C
15	30	⇈	D
16	27	⇈	Swift

TIOBE Index for January 2018

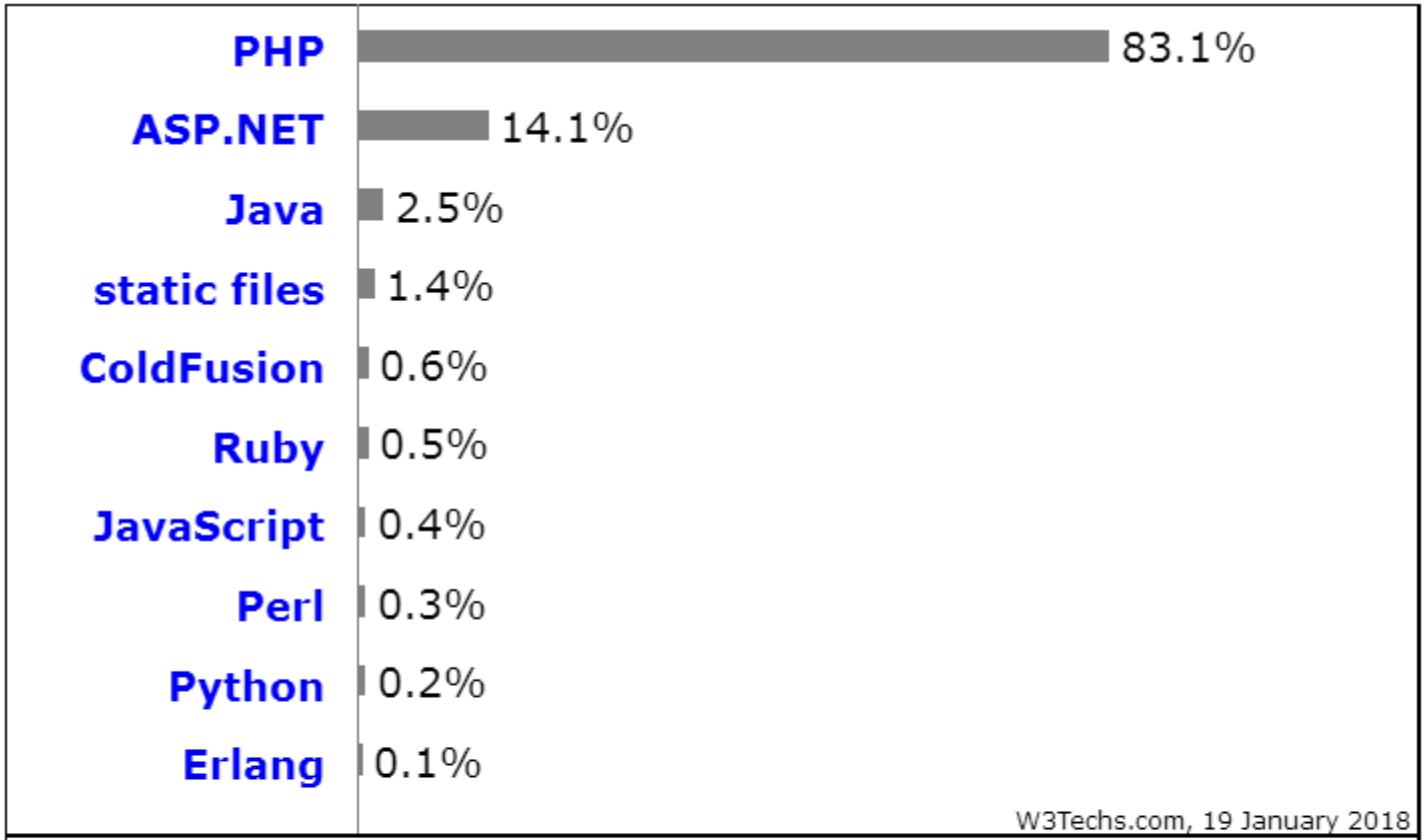
Jan 2018	Jan 2017	Change	Programming Language
1	1		Java
2	2		C
3	3		C++
4	5	↑	Python
5	4	↓	C#
6	7	↑	JavaScript
7	6	↓	Visual Basic .NET
8	16	⇈	R
9	10	↑	PHP
10	8	↓	Perl
11	12	↑	Ruby
12	14	↑	Swift

Серверні web-технології в світі

(http://w3techs.com/technologies/overview/programming_language/all)



Дані, отримані 01.02.2014



Percentages of websites using various server-side programming languages
Note: a website may use more than one server-side programming language

Дані, отримані 19.01.2018

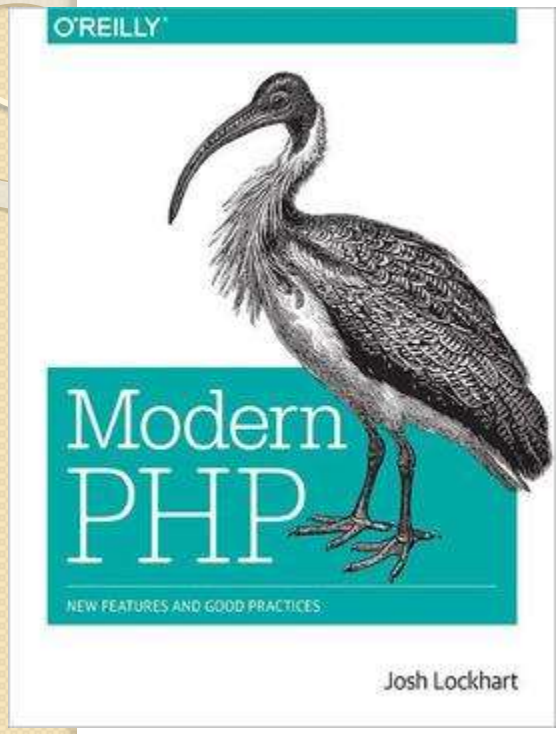
Рекомендована література

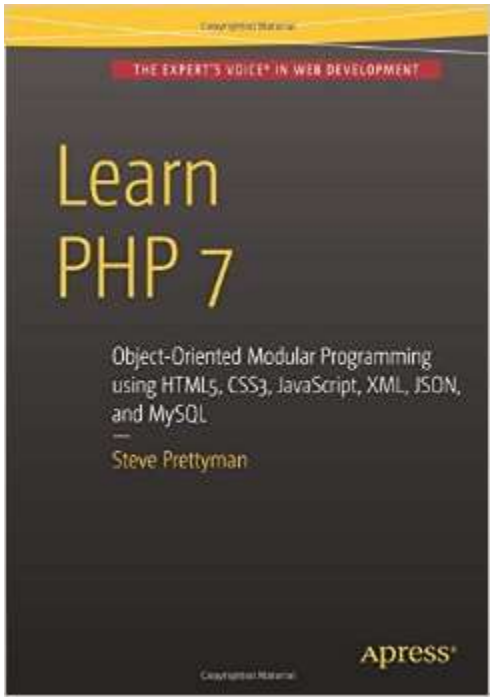
1. **Веллинг Л., Томсон Л.** Разработка веб-приложений с помощью PHP и MySQL, 4-е издание. – М.: Вильямс, 2010. – 848 с.
2. **Прохоренок Н.А.** HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера, 2010
3. **Кузнецов М.В., Симдянов И.В.** PHP. Практика создания Web-сайтов, 2-е изд., 2009
4. **Котеров Д., Костарев А.** PHP 5, 2-е изд., 20008
5. PHP 5 для профессионалов, 2006
6. **Кузнецов М.В., Симдянов И.В**. PHP на примерах, 2012
7. **Кузнецов М.В., Симдянов И.В**. Объектно-ориентированное программирование на PHP, 2008
8. **Д. Колисниченко** - PHP и MySQL. Разработка Web-приложений 2013

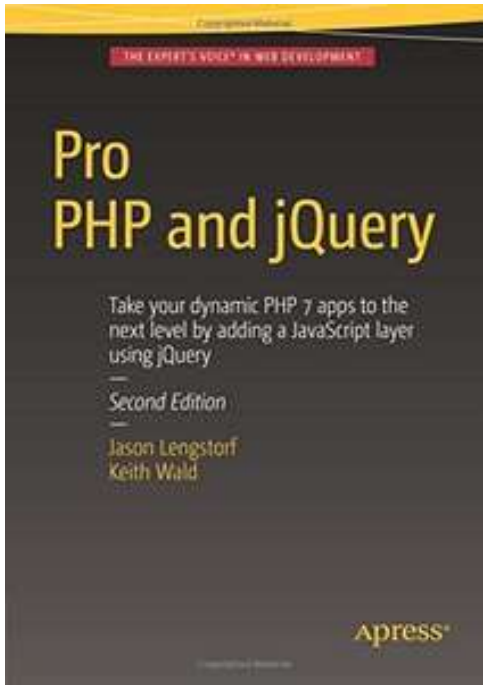
Рекомендована література

9. **PHP и MySQL. Создание Интернет-магазина**, 2011
10. **Larry Ullman** PHP and MySQL for Dynamic Web Sites, 2011
11. **Larry Ullman** PHP Advanced and Object-oriented Programming, 2012
12. **Learning PHP & MySQL** (Davis M., Phillips J.), 2012
13. **Sams Teach Yourself PHP, MySQL** and Apache All in One (5th Edition), 2012
14. **Сайт на AJAX под ключ**. Готовое решение для интернет-магазина, 2012

New Books


O'REILLY
Modern
PHP
NEW FEATURES AND GOOD PRACTICES
Josh Lockhart


THE EXPERT'S VOICE® IN WEB DEVELOPMENT
Learn
PHP 7
Object-Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL
Steve Prettyman
Apress


Pro
PHP and jQuery
Take your dynamic PHP 7 apps to the next level by adding a JavaScript layer using jQuery
Second Edition
Jason Lengstorf
Keith Wald
Apress

Server Setup 1: Manually Installing All the Software Components

- **Web-сервер Apache** (https://httpd.apache.org/)
- **СКБД MySQL** (https://www.mysql.com/)
- **Інтерпретатор скриптів PHP** (https://www.php.net/)

Server Setup 2: Pre-packaged Installations

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

1. XAMPP (X, Apache, MySQL, PHP, Perl) – https://www.apachefriends.org (**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)

Робочий каталог : C:\xampp\htdocs

3. WampServer (http://www.wampserver.com/) і т.д.
4. Open Server - https://ospanel.io/

Робочий каталог

- Робочий каталог

для XAMPP – C:\xampp\htdocs)

Застереження

- Перш ніж встановлювати програми, необхідно перевірити мережеві настройки і відсутність програм, що займають порти 80 і 3306, тому що ці порти використовують Web-сервер Apache і сервер MySQL
- `ping 127.0.0.1`
- `netstat –anb`

У списку не повинно бути рядків з портами 80 і 3306. Якщо вони є, то **Apache** або **MySQL** не зможуть запуститися.

Зазвичай ці порти займають програми **Skype** і Web-сервер **IIS**. Перед установкою і використанням Apache і MySQL ці програми не слід запускати

Server Setup 3: Virtual Servers

- Віртуальний сервер працює як веб-сервер на іншому комп'ютері. На цьому комп'ютері може бути запущена будь-яка операційна система.

Треба встановити наступне ПЗ:

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

(Дивись: PHP & MySQL: Novice to Ninja, 6th Edition, 2017)

Online coding environments

- EasyPHP's Code Classroom EasyPHP (www.easyphp.org) now has an online coding environment for students and teachers.
- This environment allows you to **enter code** (black window), click a **Submit** button (red button), and **see your results** on the right (white window).
- www.codeclassroom.net
- **PHP 5 Tutorial**
  https://www.w3schools.com/php/default.asp

Редактори PHP та IDE

- **Notepad++ - free**
- **Brackets (http://brackets.io/)**
- **Visual Studio Code**
- **CodeLobster PHP Edition - free**
- Sublime Text (http://www.sublimetext.com/)
- JetBrains **PhpStorm** - $199 (Free individual licenses for students)
- **phpDesigner** 8.0.0.145
- **Eclipse**-php-helios (www.eclipse.org/pdt/) - free
- **NetBeans**-7.2.1-ml-php - free
- **Atom** (https://atom.io/)
- Aptana Studio (aptana.com) - free
- **Zend Studio** - $189
- Komodo IDE - $382

Фреймворки PHP

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

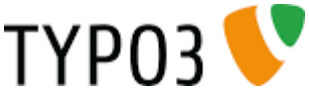
1. Zend Framework
2. CakePHP
3. CodeIgniter
4. Kohana
5. **Symfony**
6. **Yii**
7. **Laravel**

CMS на базі PHP

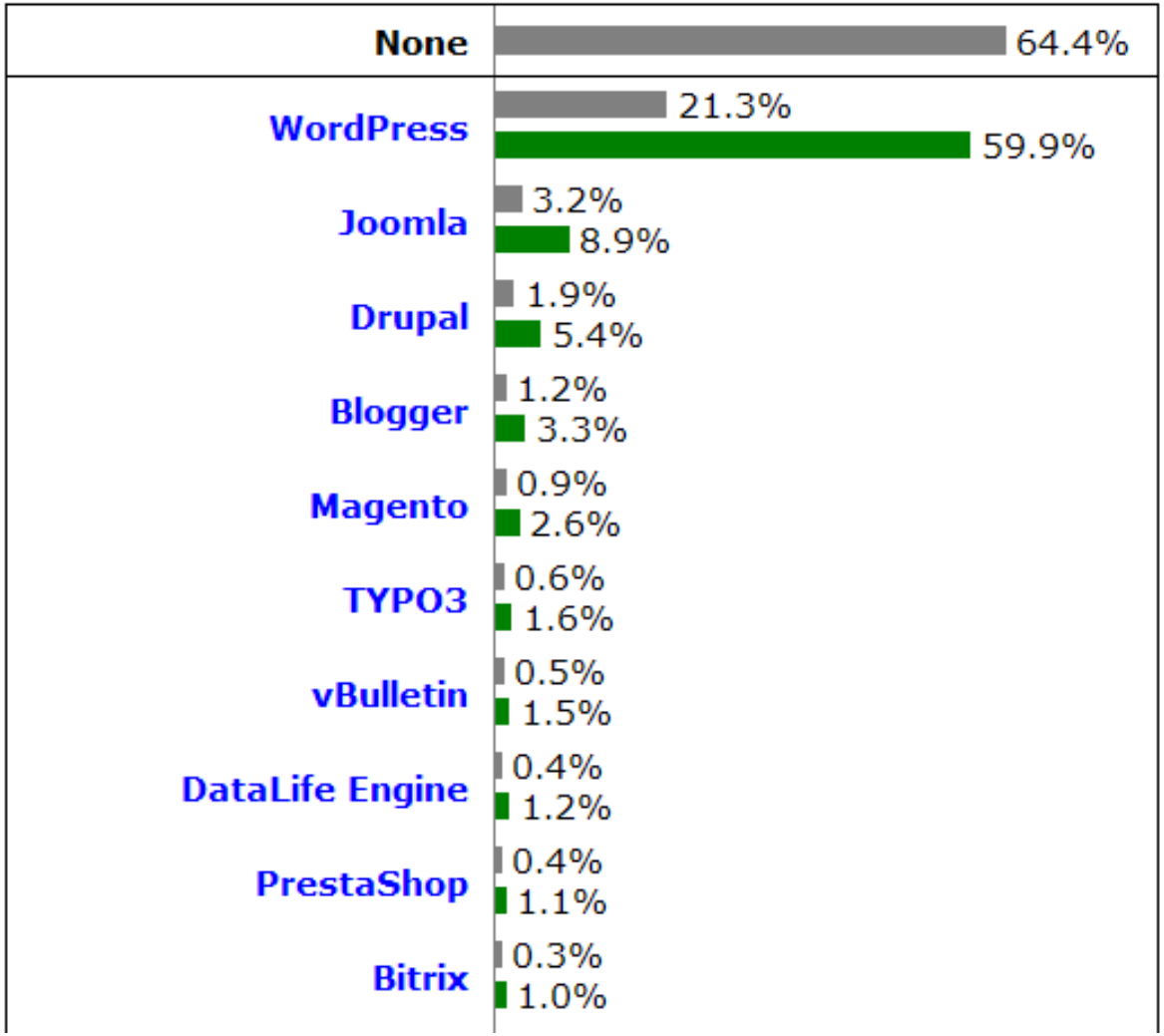
Система керування вмістом (СКВ; англ. Content Management System, CMS) — програмне забезпечення для організації веб-сайтів. Надає інструменти для додавання, редагування, видалення інформації на сайті (без програмування).

1. Joomla!
2. Drupal
3. **WordPress**
4. XOOPS
5. TYPO3
6. 1С-Битрикс

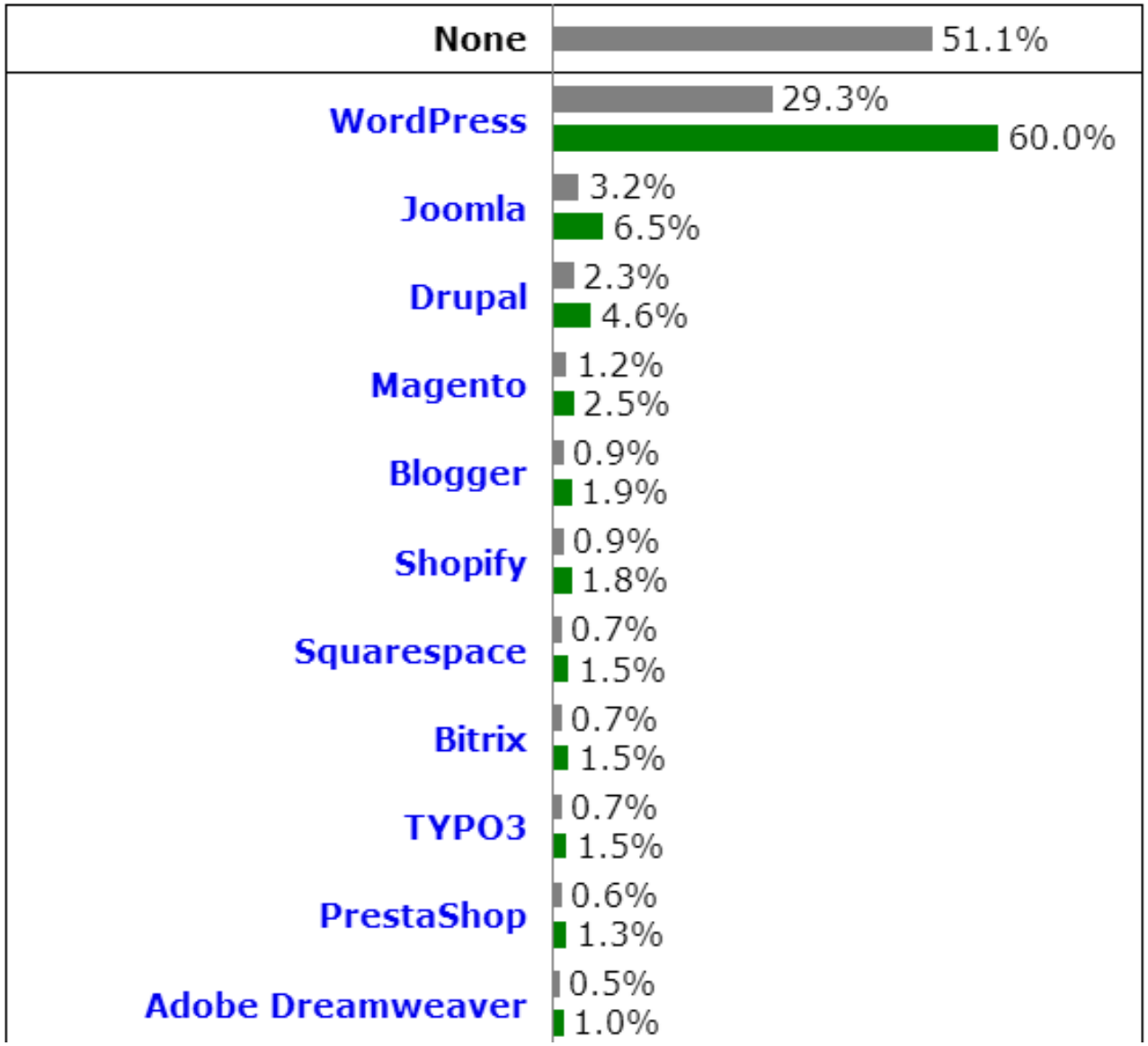
. . .



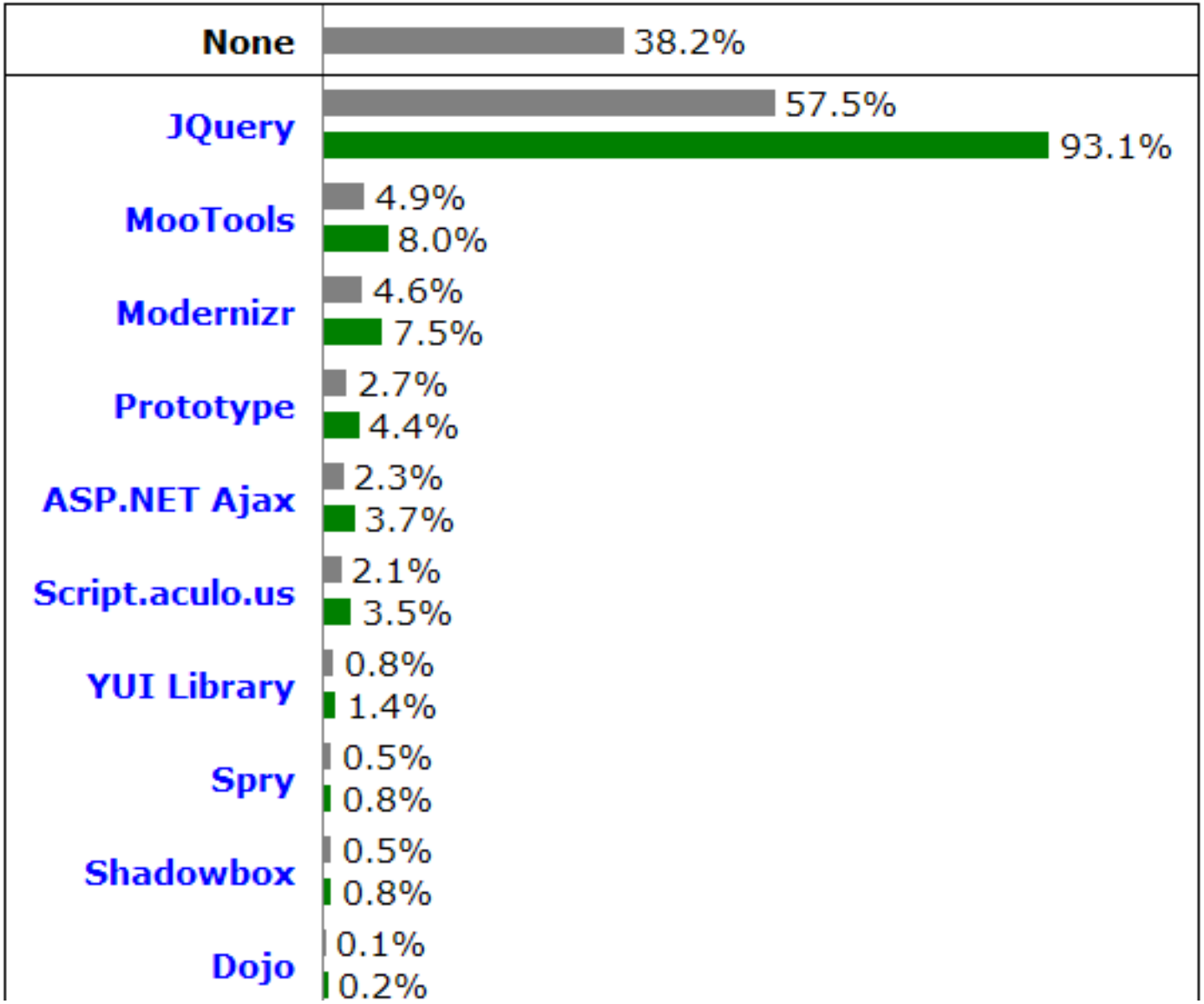
Usage of content management systems for websites (2014)



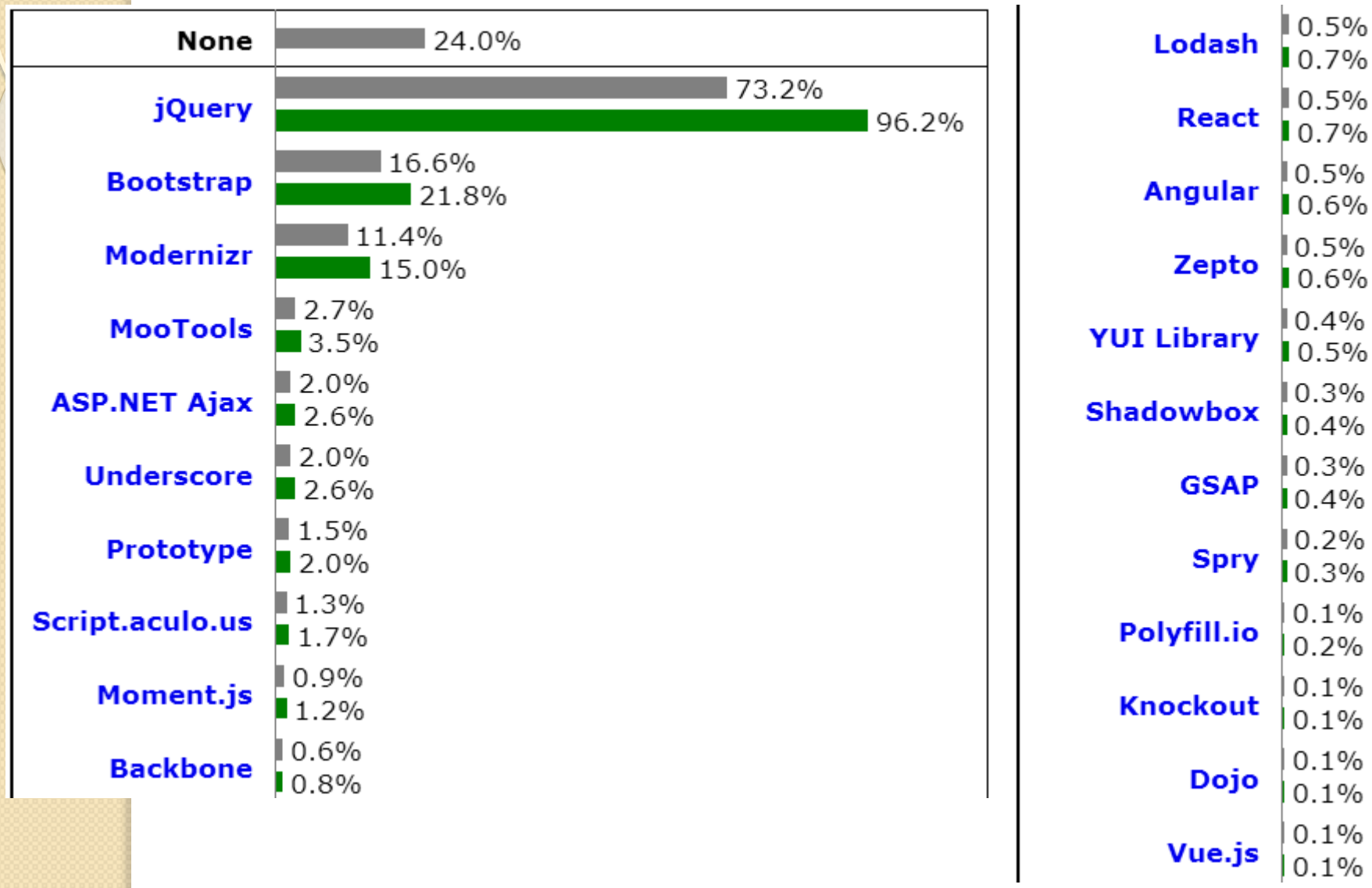
Usage of content management systems for websites (Jan 2018)



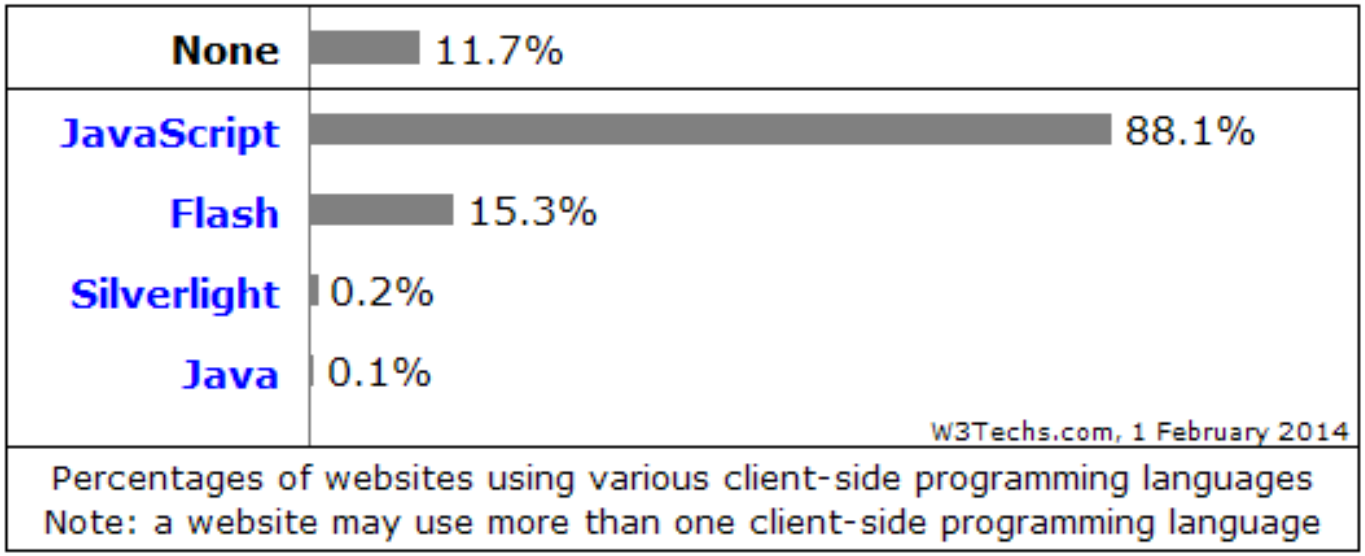
Usage of JavaScript libraries for websites (2014)



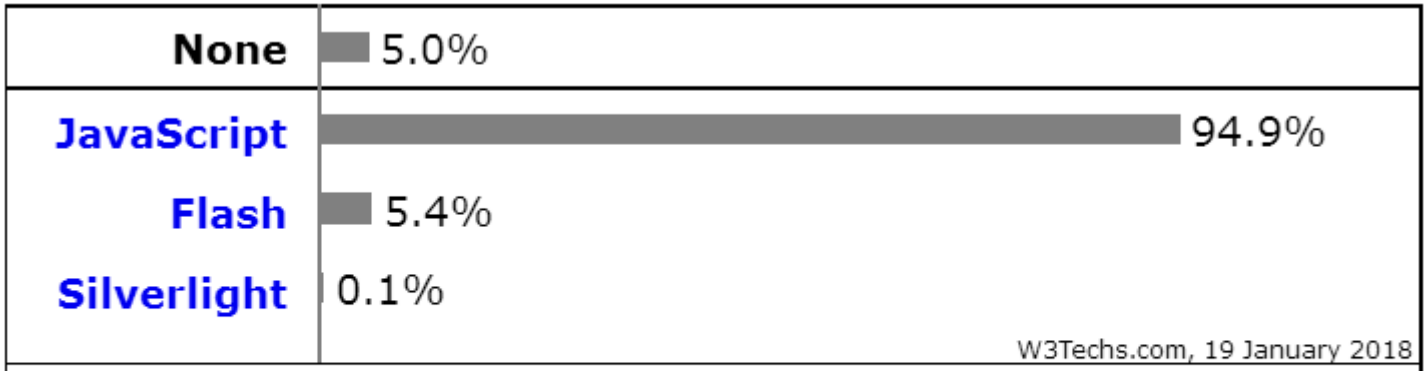
Usage of JavaScript libraries for websites (Jan 2018)



Usage of client-side programming languages for websites

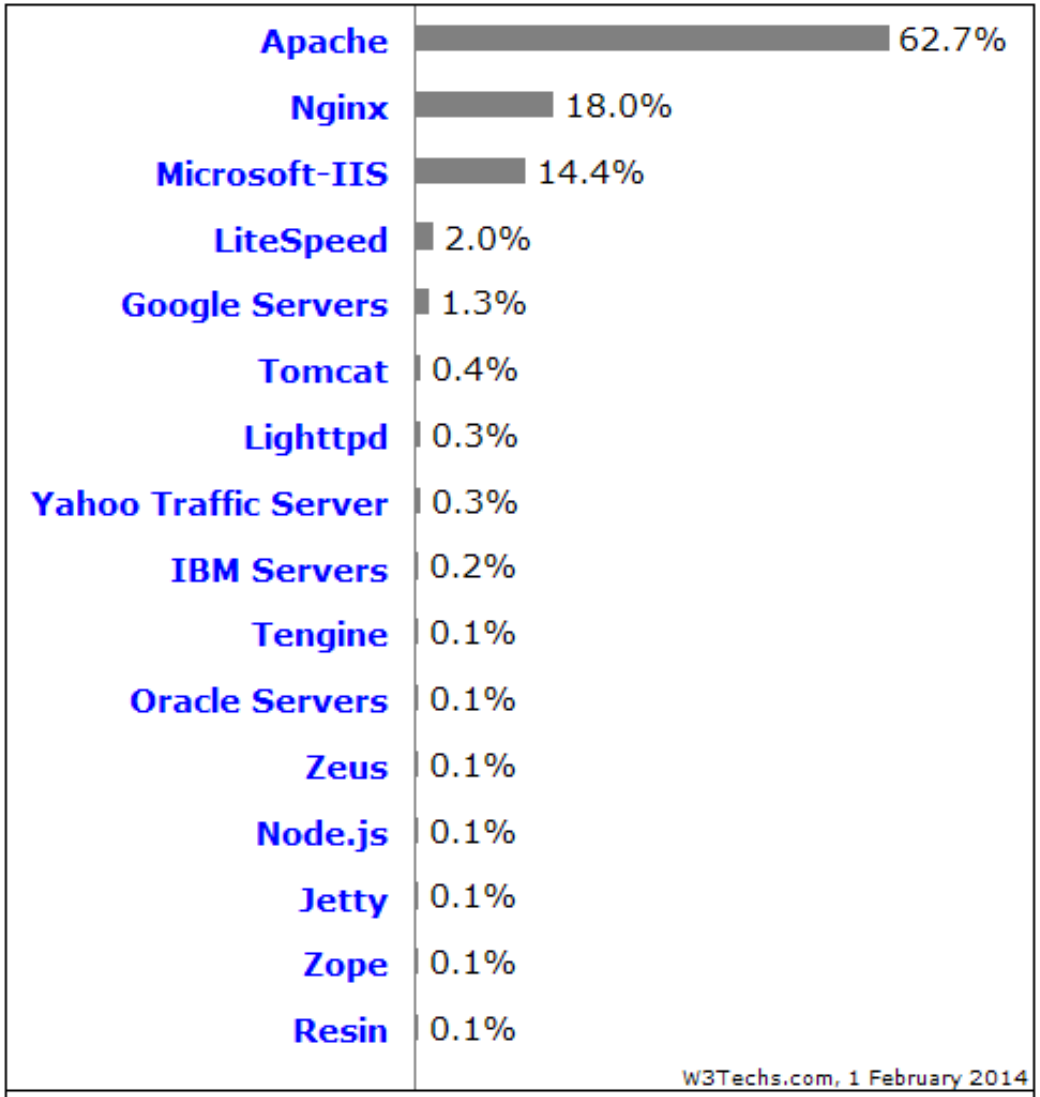


Percentages of websites using various client-side programming languages
Note: a website may use more than one client-side programming language

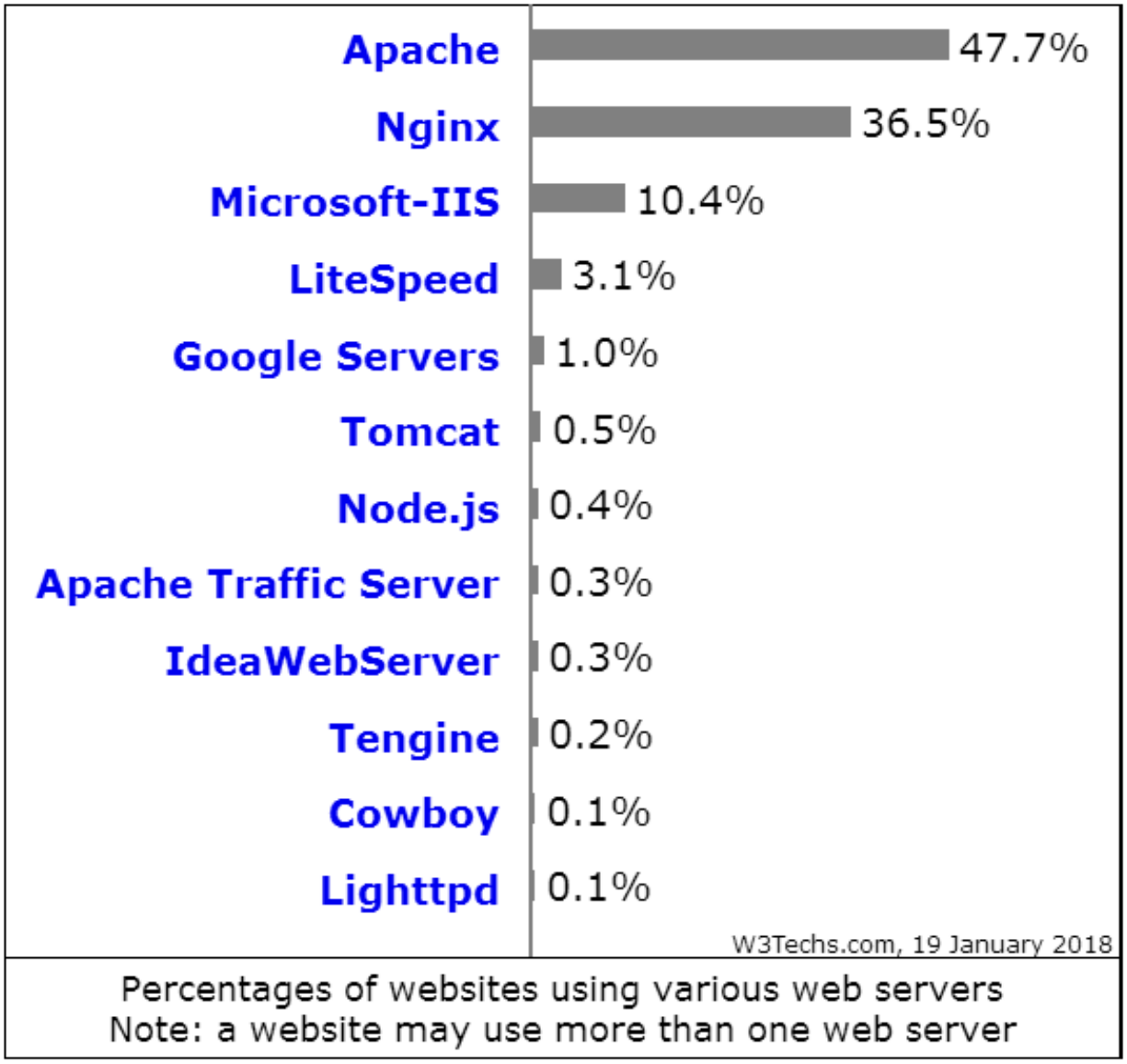


Percentages of websites using various client-side programming languages
Note: a website may use more than one client-side programming language

Usage of web servers for websites (2014)

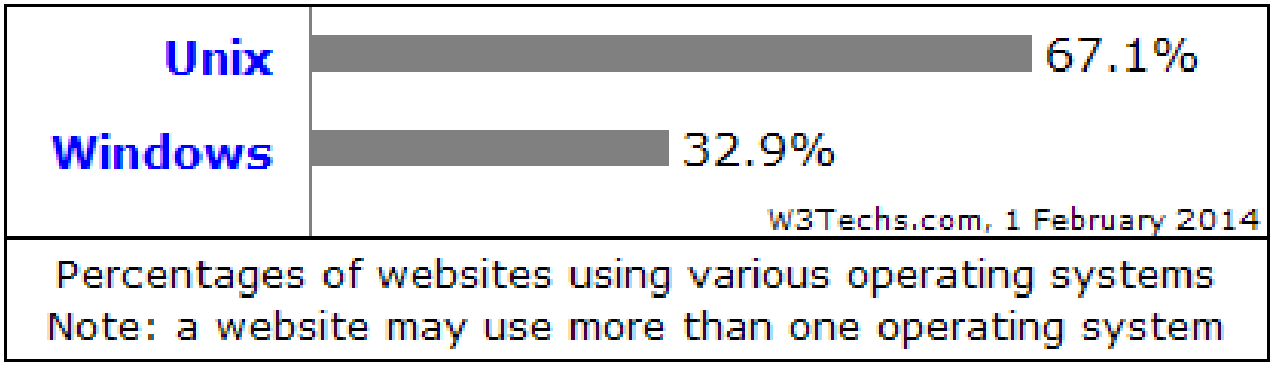


Usage of web servers for websites (Jan 2018)

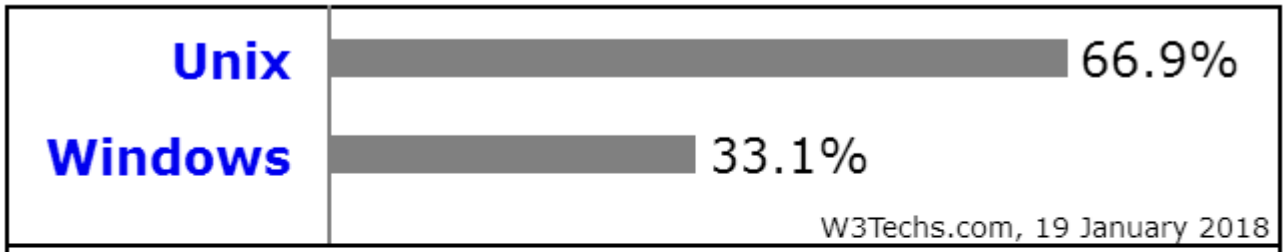


Percentages of websites using various web servers
Note: a website may use more than one web server

Usage of operating systems for websites



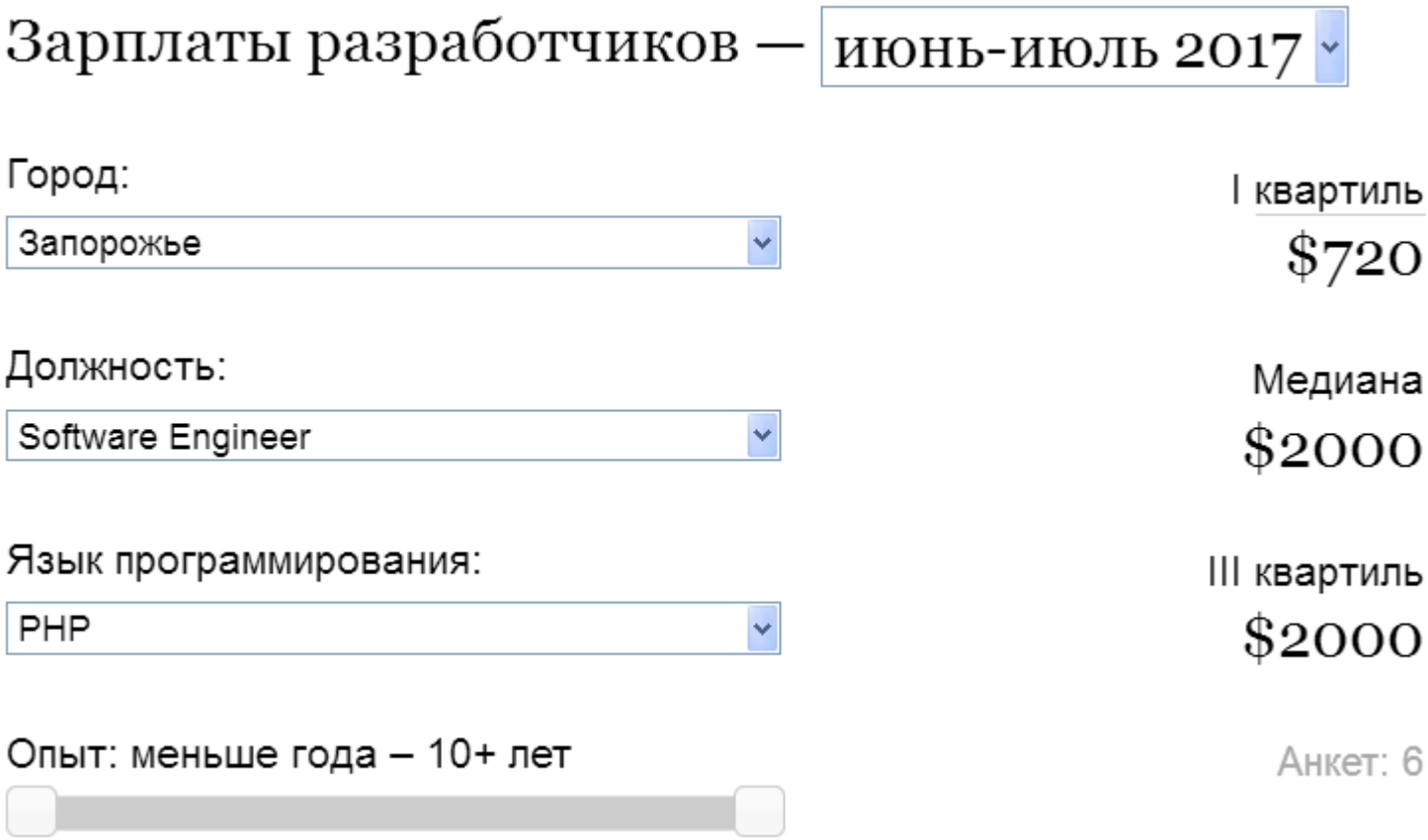
Percentages of websites using various operating systems
Note: a website may use more than one operating system



Percentages of websites using various operating systems
Note: a website may use more than one operating system

Зарплати розробників

(https://jobs.dou.ua/salaries/#period=jun2017&city=Zaporizhia&title=Software%20Engineer&language=PHP&spec=&exp1=0&exp2=10)



Корисні посилання

- www.php.net/manual/ru
- www.phpclub.ru
- www.php.su
- www.softtime.ru
- px.sklar.com
- http://www.sitepoint.com/
- www.hotscripts.com
- www.spravkaweb.ru/mysql/
- Phpfaq.ru
- **https://www.w3schools.com/php/**

PHP – це акронім

Означає: “PHP: Hypertext Preprocessor”

(PHP: Гіпертекстовий препроцесор)

- Хостинг https://www.ho.ua/

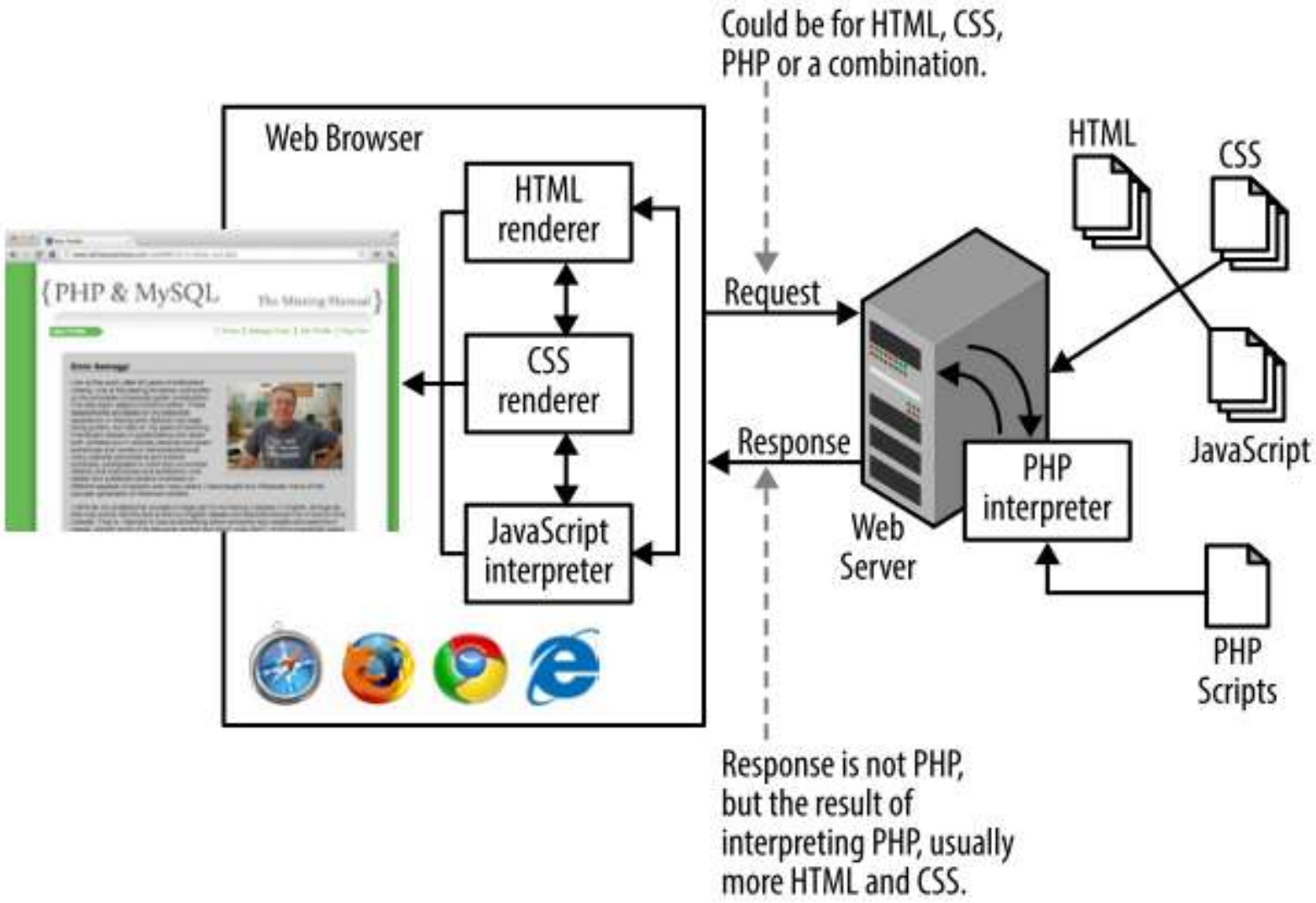
Історія PHP

- 1995 р . Датський програміст Rasmus Lerdorf написав Perl-скрипт для домашньої сторінки. Назвав PHP (Personal Home Page Tools). Потім переписав на мові C.
- **1997 р**. Andi Gutmans та Zeev Suraski (Ізраїль) покращили функціональність. В 1998 р. був вже PHP 3, працював на 10% веб-серверів. Назву змінили – PHP 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



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, . . .)

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

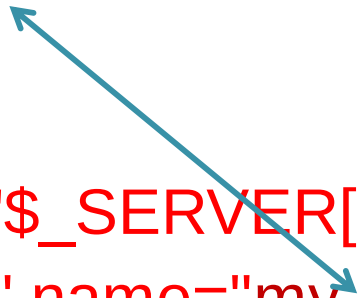
Файл formexample.php

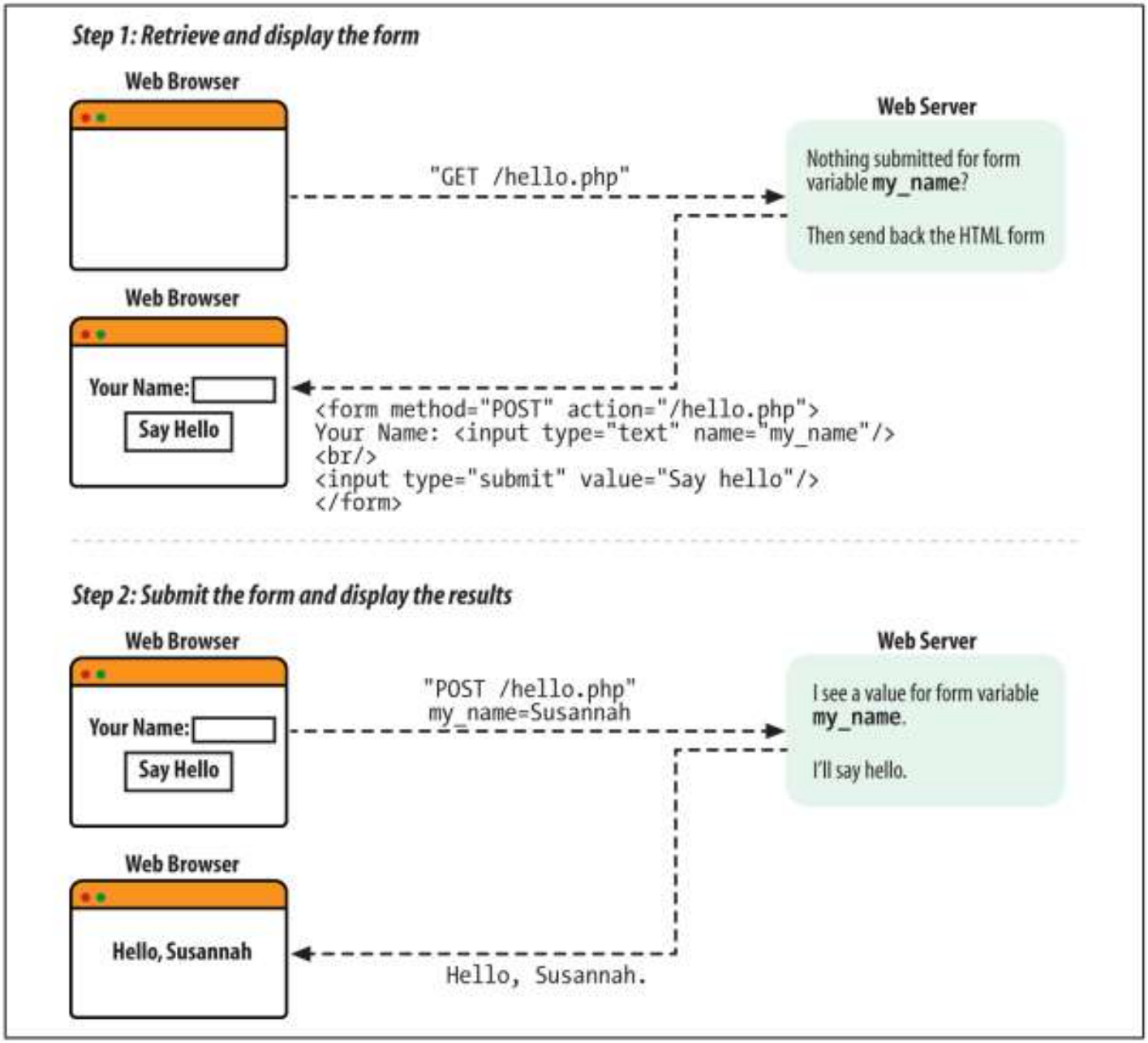
```
<html>
<head> <title> PHP Test </title> </head>
<body>
<?php
        if(isset($_POST['submit'])){
        echo "Hi, ".$_POST['name']."!<br/>";
        }
?>
<form action="formexample.php" method="post">
        <p>
        Name: <br/>
        <input  type="text" name="name" size="20" maxlength="40"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




Step 1: Retrieve and display the form
Web Browser
Web Server
"GET /hello.php"
Nothing submitted for form variable my_name?
Then send back the HTML form
Web Browser
Your Name:
Say Hello
<form method="POST" action="/hello.php">
Your Name: <input type="text" name="my_name"/>

<input type="submit" value="Say hello"/>
</form>
Step 2: Submit the form and display the results
Web Browser
Web Server
"POST /hello.php"
my_name=Susannah
I see a value for form variable my_name.
I'll say hello.
Your Name:
Say Hello
Web Browser
Hello, Susannah
Hello, Susannah.

Відокремлення PHP коду від HTML розмітки

Команди мови PHP обрамляються спеціальними дескрипторами – тегами мови PHP. Підтримуються 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 воно відображається 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
?>
```

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 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

Типи даних

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) – повідомляє, 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_CONVERSION' =>
  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 в 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
  1) Деякі символьні послідовності, що починаються з '\' замінюються спецсимволами
  2) 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

Функція	Опис
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 [, $limit])	Функція повертає масив з рядків, кожен з яких відповідає фрагменту вихідної рядка **$str**, що знаходиться між роздільниками, 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];
?>
```

Випадкові елементи масиву

Функція	Опис
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

  $pets = array ('lizard' => 'Ігуана',
                 'spider' => 'Чорна вдова',
                 'parrot' => 'Ара');
  print_r ($pets); // Друкування масива
?>
```

Walking through an associative array using 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

Walking through an associative array using each and 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

Це масив, елементами якого є не тільки скалярні величини, але й самі масиви

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

Багатовимірні масиви

```
<?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 ф-ю var_dump($object)

Creating a multidimensional associative array

```
<?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

Функція	Опис
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.

sort - Відсортувати масив.

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
)
```

Протокол HTTP

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) 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:

- **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)

- 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

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. Детальніше – RFC 2068 (www.rfc.net)

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

HTTP-відповідь

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

```
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>
```

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.

Основні заголовки

- **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
- Postman – HTTP-клієнт для тестування сайтів
- 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");
```

Приклад 1. Передача форми

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])

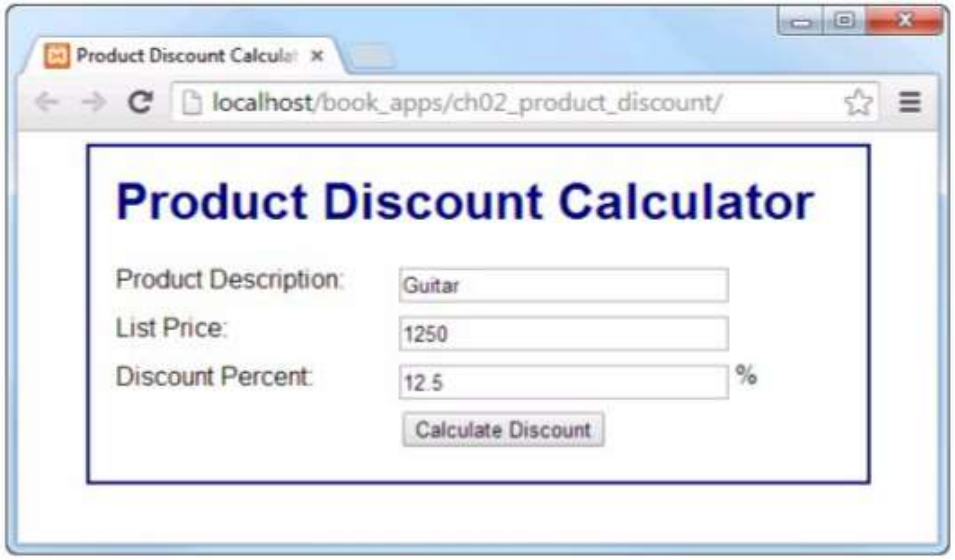
type – Задає суперглобальний масив для доступу (INPUT_GET, INPUT_POST, INPUT_COOKIE)

variable_name – Ім'я змінної

filter – FILTER _VALIDATE_INT, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_EMAIL, FILTER_VALIDATE_URL , FILTER_VALIDATE_BOOLEAN

Приклад (Murach's PHP and MySQL, 2nd Ed, p. 69)

- The first page (index.html)



- The second page (product_discount.php)

index.html

```
<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>
```

display_discount.php (частина 1)

```
<?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);
?>
```

display_discount.php (частина 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>
. . . . . .
```

Murach's PHP and MySQL, 2nd Edition

Як отримати дані з radio button

```
<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
```

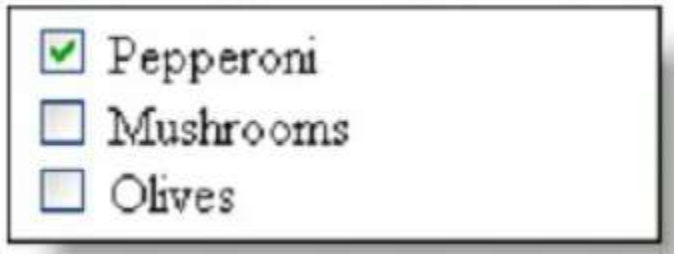

```
<?php
$card_type = filter_input(INPUT_POST, 'card_type' );
?>
```

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

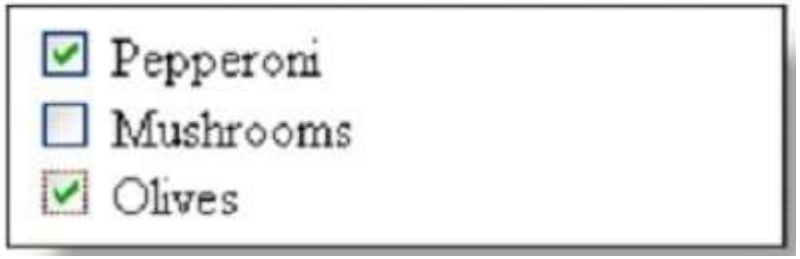
```
<input type="checkbox" name="pep" checked>
  Pepperoni<br>
<input type="checkbox " name="msh"> Mushrooms<br>
<input type="checkbox " name="olv"> Olives
```




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

Як отримати дані з масиву check box

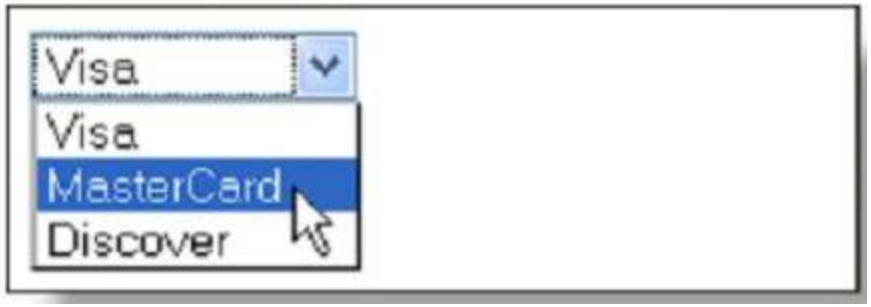
```
<input type="checkbox " name="top[]" value="pep">
  Pepperoni<br>
<input type="checkbox " name="top[]" value="msh">
  Mushrooms<br>
<input type="checkbox" name="top[]" value="olv"> Olives
```




```
<?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

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




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

Як отримати дані з list box

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


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


```
<?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

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

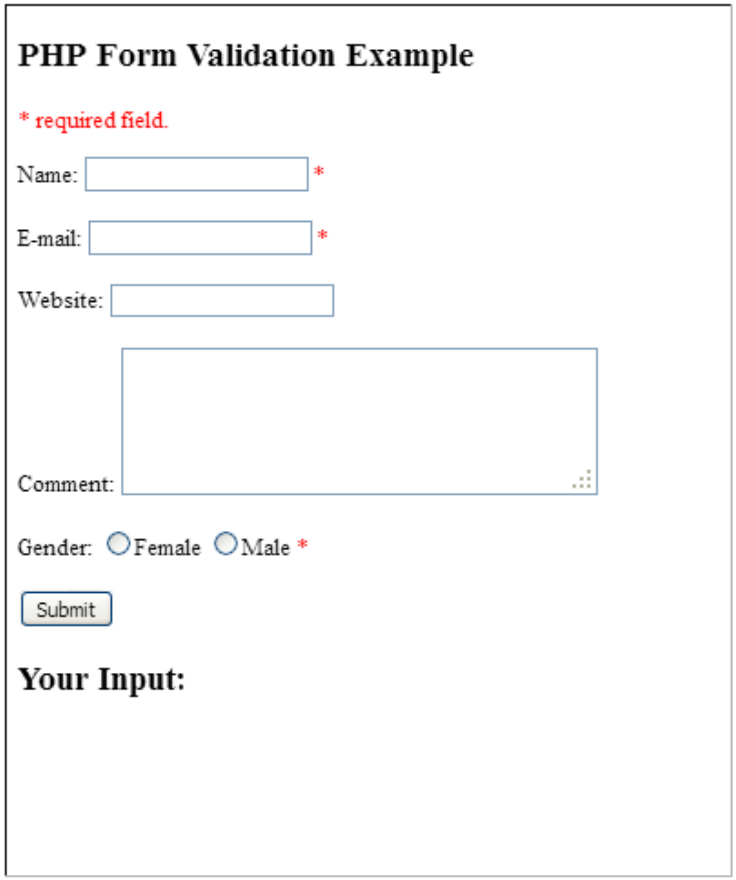

```
<?php
$conunent =  filter _ input(INPUT_POST,    'conunent');
? >
```

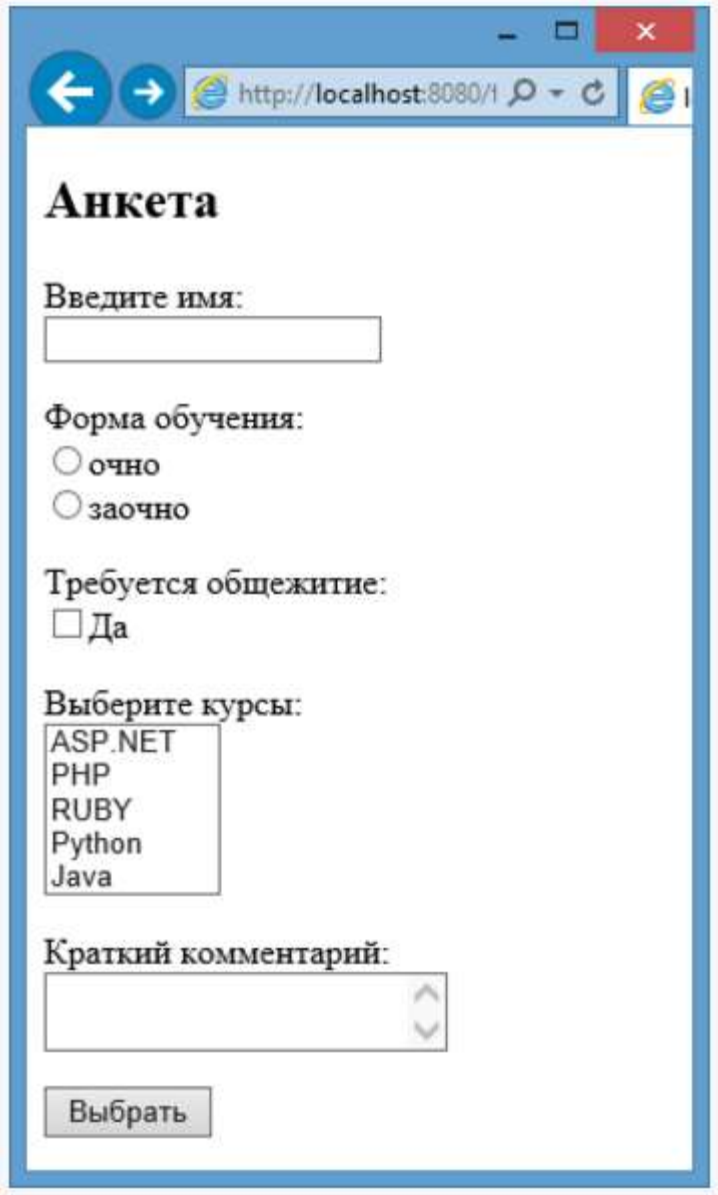
Приклад обробки форми

(http://metanit.com/web/php/3.4.php)

PHP Form Validation

https://www.w3schools.com/php/php_form_complete.asp





Функції в PHP

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

```php
<?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
<?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
<?php
     $a = 1; $b = 2;
     function Sum (){
     global $a, $b;
     return $b += $a;
     }
     echo Sum();
?>
```

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

Lambda and Closure

(Анонімні функції та замикання)

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.

Lambda and 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
- . . .

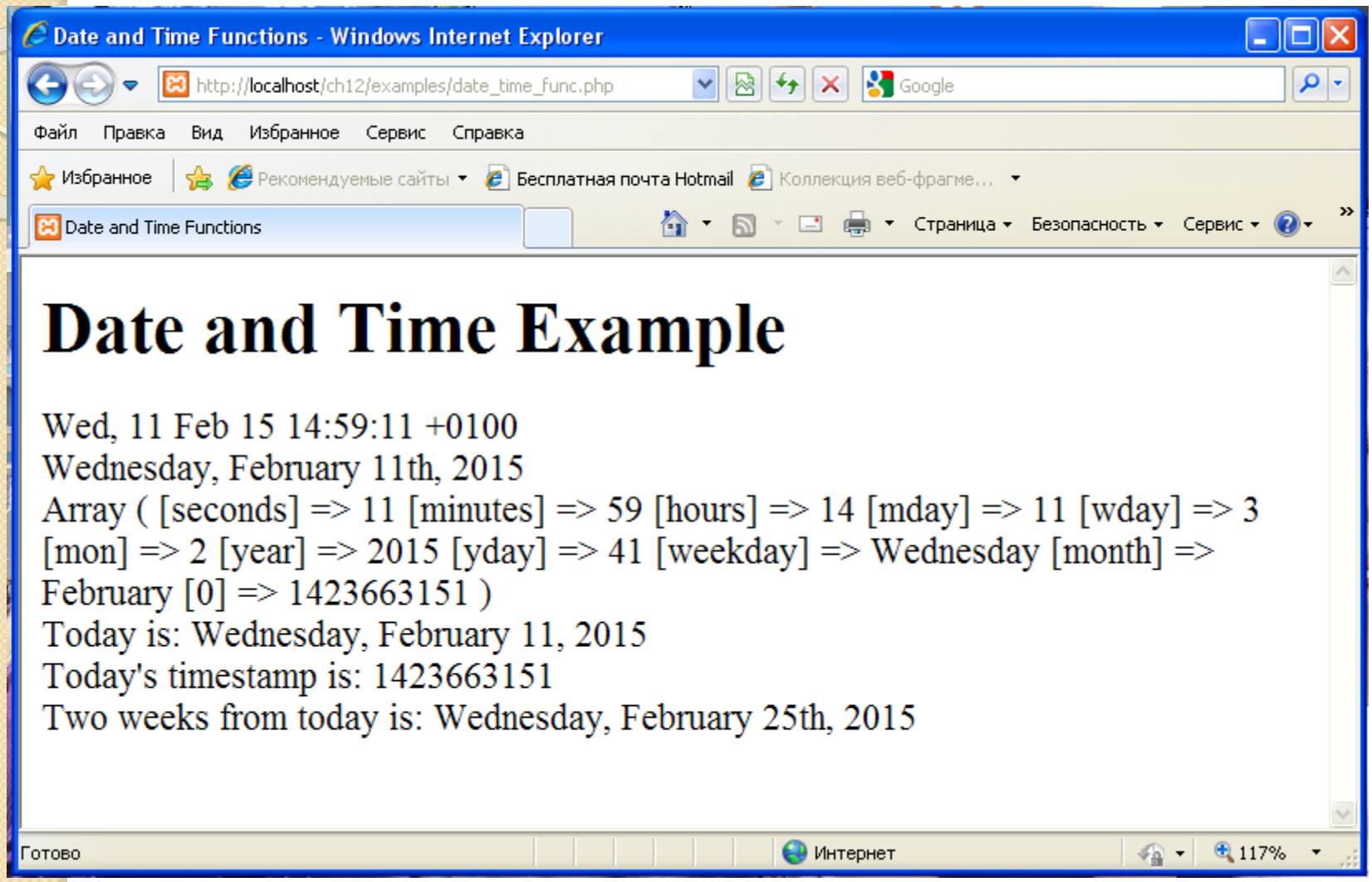
Розбор timestamp: **array getdate()** - повертає асоціативний масив, що містить дані поточної дати та часу (seconds, minutes, hours, …, mon, year)

Приклад. (файл date_time_func.php)

```
<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


Date and Time Functions - Windows Internet Explorer
http://localhost/ch12/examples/date_time_func.php
Google
Файл Правка Вид Избранное Сервис Справка
Избранное
Рекомендуемые сайты
Бесплатная почта Hotmail
Коллекция веб-фрагме...
Date and Time Functions
Страница
Безопасность
Сервис
Date and Time Example
Wed, 11 Feb 15 14:59:11 +0100
Wednesday, February 11th, 2015
Array ([seconds] => 11 [minutes] => 59 [hours] => 14 [mday] => 11 [wday] => 3 [mon] => 2 [year] => 2015 [yday] => 41 [weekday] => Wednesday [month] => February [0] => 1423663151)
Today is: Wednesday, February 11, 2015
Today's timestamp is: 1423663151
Two weeks from today is: Wednesday, February 25th, 2015
Готово
Интернет
117%

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.

Повторне використання коду (Code Reuse)

```php
<?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
?>
```

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

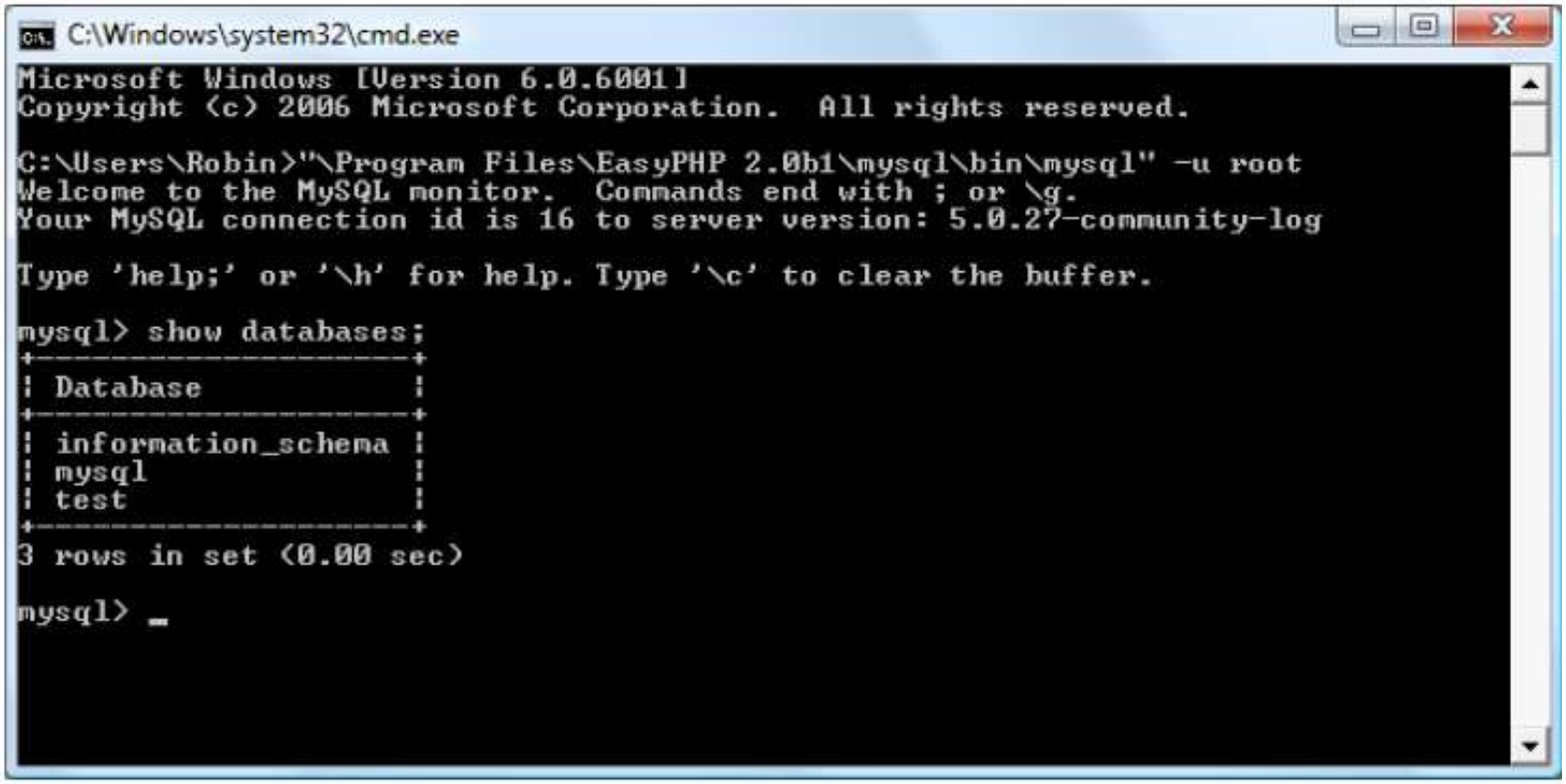
SHOW COLUMNS FROM tablename;

Для перегляду значень таблиці: **SELECT** * **FROM tablename**

Інтерфейс командного рядка

C:\xampp\mysql\bin\mysql -u root

mysql>SHOW databases;



MySQL Commands

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

Creating a database

```
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;
```

Data Types

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

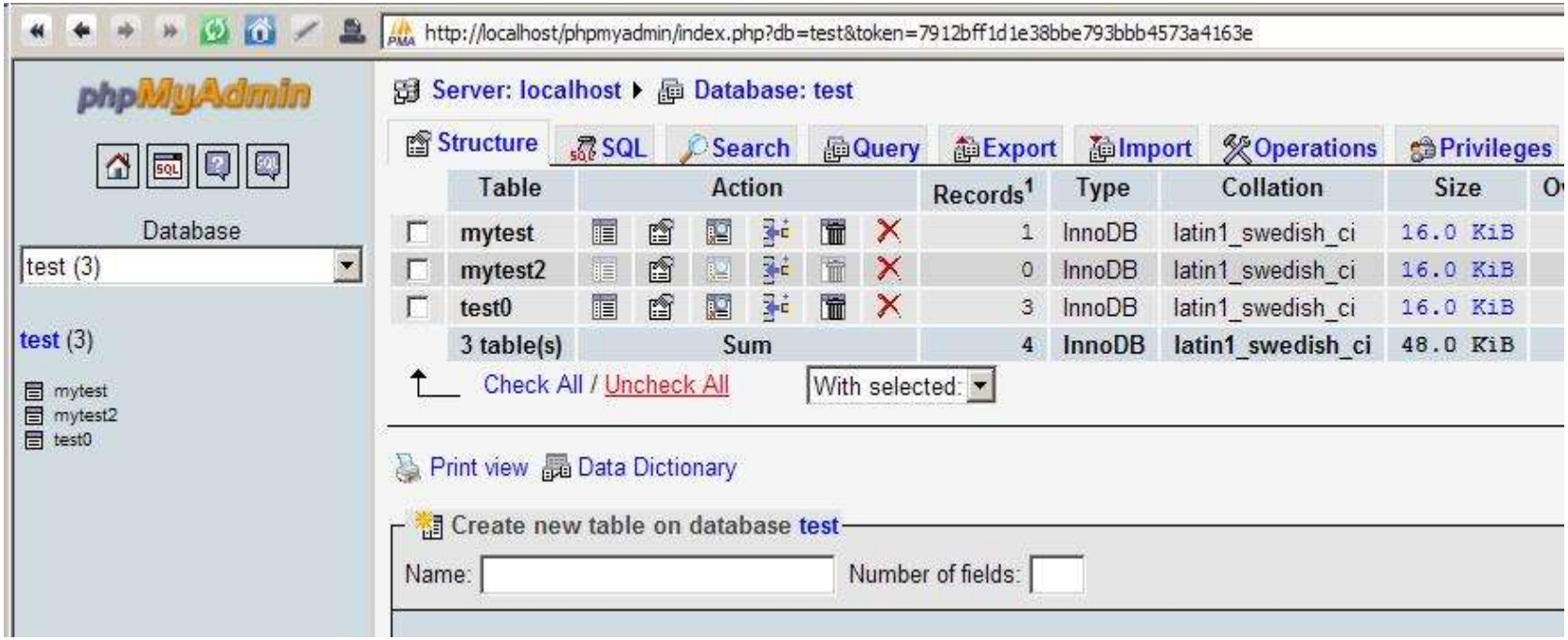
Data type	Bytes used	Attributes
TINYBLOB(*n*)	Up to *n* (<= 255)	Treated as binary data—no character set
BLOB(*n*)	Up to *n* (<= 65535)	Treated as binary data—no character set
MEDIUMBLOB(*n*)	Up to *n* (<= 1.67e+7)	Treated as binary data—no character set
LONGBLOB(*n*)	Up to *n* (<= 4.29e+9)	Treated as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

Зв'язок PHP з MySQL

phpMyAdmin — веб-застосунок з відкритим кодом, написаний на мові PHP, представляє собою веб-інтерфейс для адміністрування СКБД MySQL



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.

Зв'язок PHP з MySQL

- 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. масив

mysqli_fetch_assoc - Занести запис в асоц. масив

mysqli_fetch_row($result) - Занести запис в масив

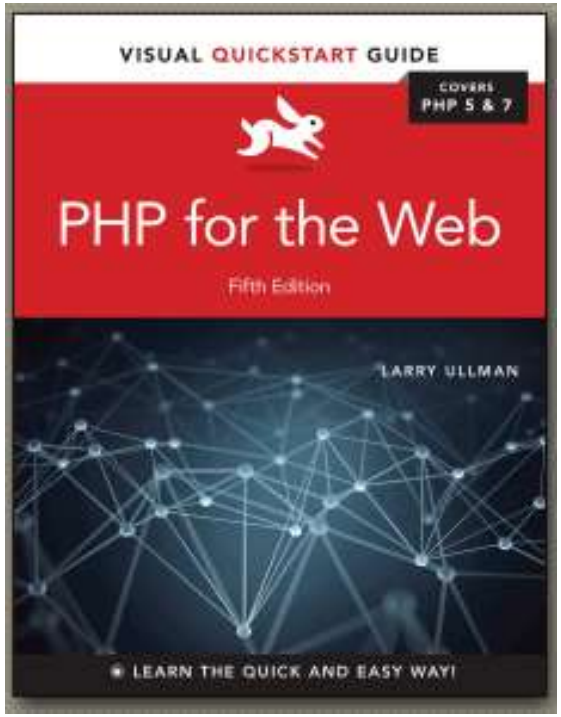
mysqli_close(id) - Закриття з'єднання з БД

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

```
<?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 ‘. . . . . .’
}
```



(**Larry Ullman PHP for the Web - Visual Quickstart Guide, 2016, p.357**)

Приклад: Вставка даних в таблицю БД (Larry Ullmann, p.360)


```
<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>
```

Приклад: Вставка даних в таблицю БД

```
. . . . .
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

```
// 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";
}
. . . . .
```

Приклад. Видалення даних у базі даних (1)

При виборі Delete на сторінці view_entries.php треба вивести вибраний пост і запропонувати його видалити.

А після видалення вивести:

Приклад. Видалення даних у базі даних (2)

```
// 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  }
```

Приклад. Видалення даних у базі даних (3)

```
} 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>';
}
. . . . . .
```

Оновлення даних у базі даних (1)

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

А після редагування вивести:

Оновлення даних у базі даних (2)

```
// 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'])) {
```

Оновлення даних у базі даних (3)

```
$problem = FALSE;
 if (!empty($_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

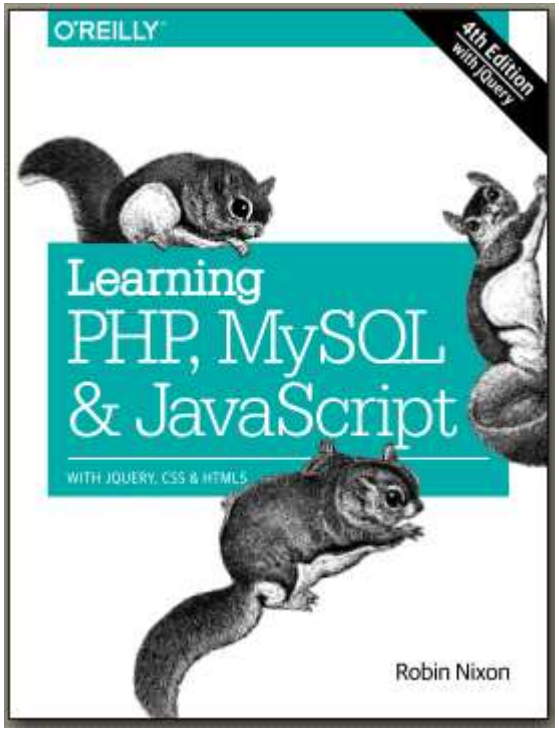
2) Об’єктно-орієнтований стиль

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

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

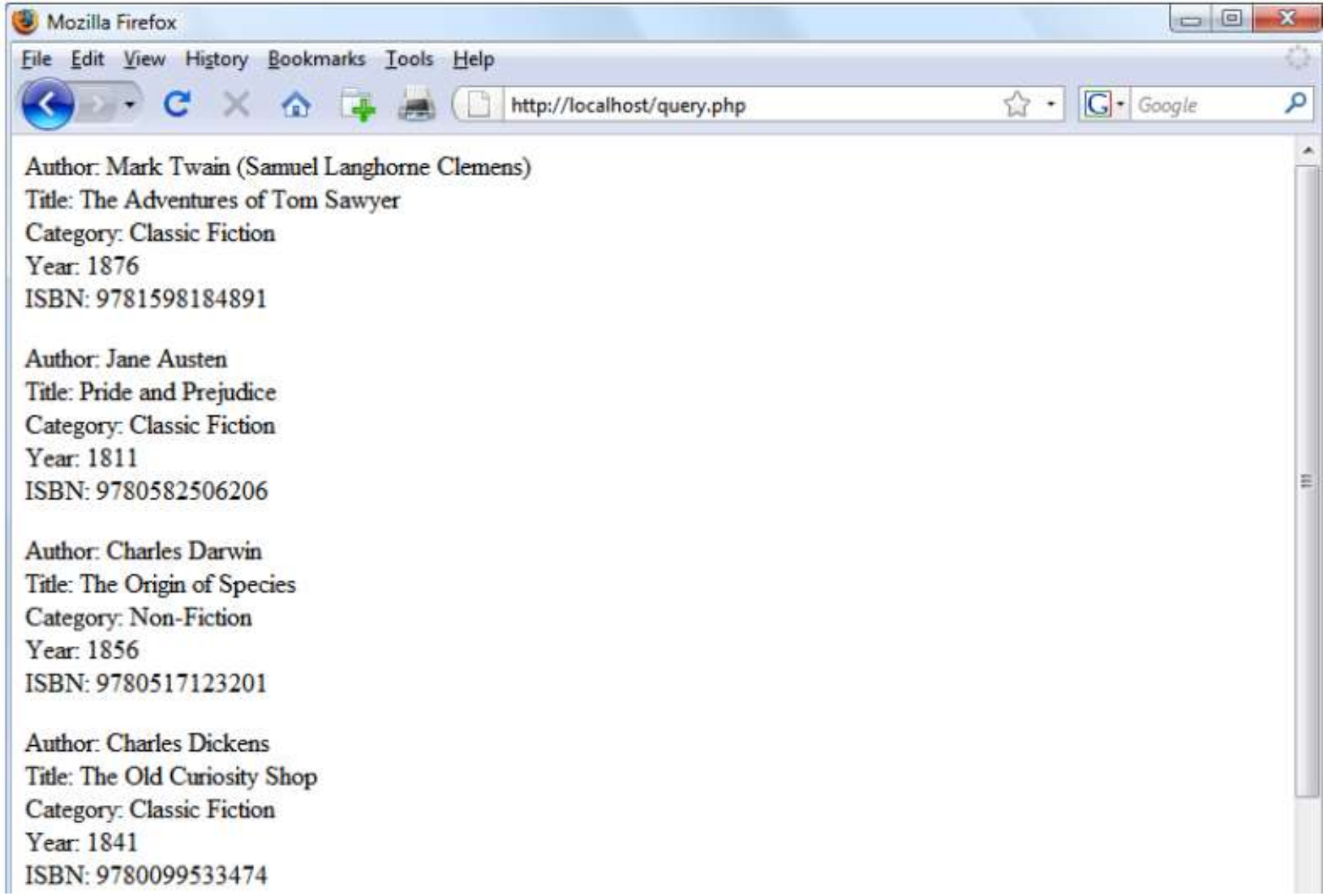
Далі необхідно викликати методи об’єкта **mysqli**

Приклад (Learning PHP, MySQL & JavaScript With jQuery, CSS & HTML5, 4th Edition 2015, Chapter 10)




```php
<?php // login.php
  $hn = 'localhost';
  $db = 'publications';
  $un = 'username';
  $pw = 'password';
?>
<?php // query.php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db); // ООП-стиль
  if ($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);   //переміщення на рядок $j
    echo 'Author: '    . $result->fetch_assoc()['author']   . '<br>';
    $result->data_seek($j);
    echo 'Title: '     . $result->fetch_assoc()['title']    . '<br>';
    $result->data_seek($j);
    echo 'Category: ' . $result->fetch_assoc()['category'] . '<br>';
    $result->data_seek($j);
    echo 'Year: '      . $result->fetch_assoc()['year']     . '<br>';
    $result->data_seek($j);
    echo 'ISBN: '      . $result->fetch_assoc()['isbn']     . '<br><br>';
  }                          //Повертає рядок у вигляді асоц. масиву
  $result->close();
  $conn->close();
?>
```

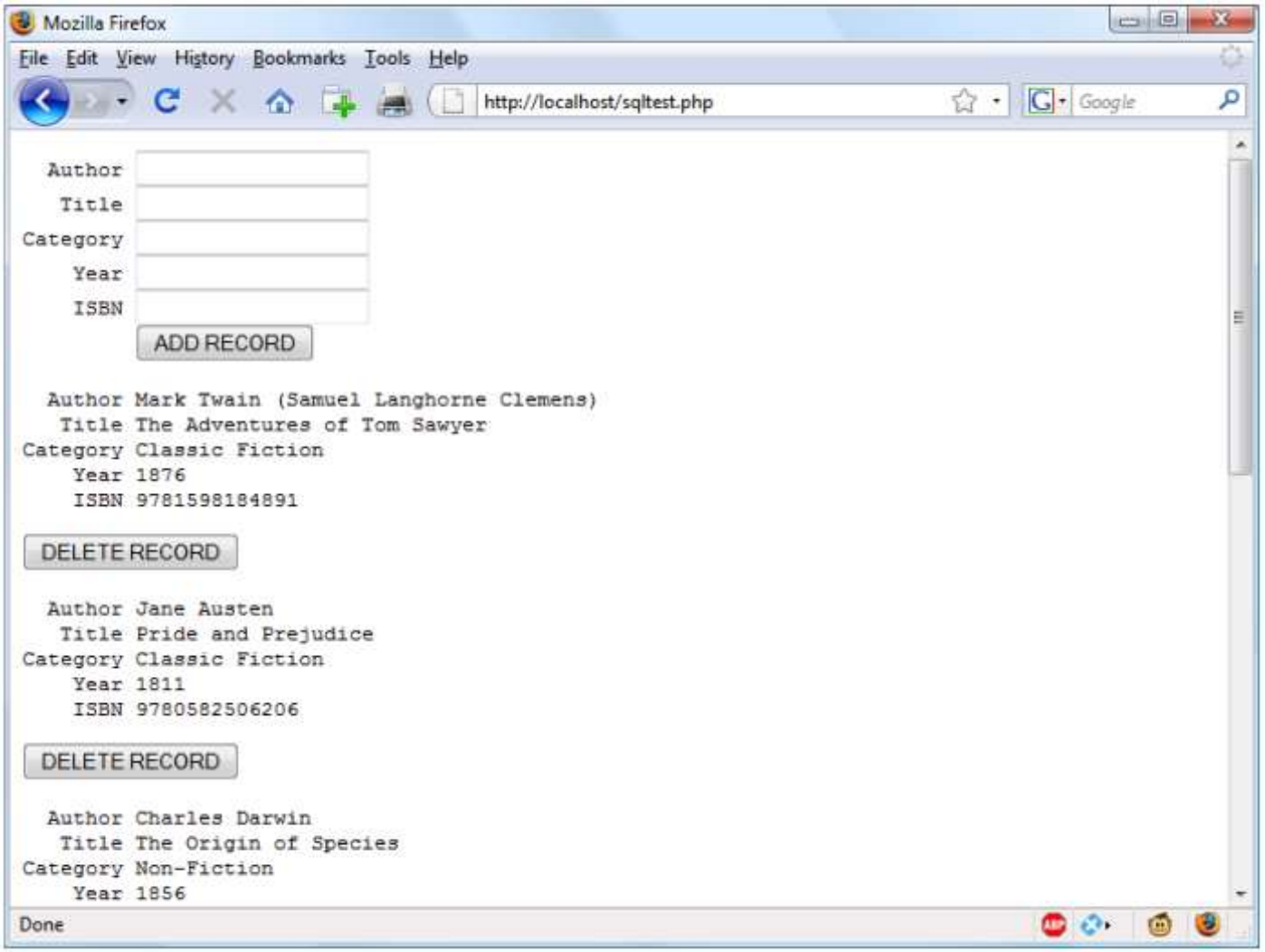

Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/query.php
Google
Author: Mark Twain (Samuel Langhorne Clemens)
Title: The Adventures of Tom Sawyer
Category: Classic Fiction
Year: 1876
ISBN: 9781598184891
Author: Jane Austen
Title: Pride and Prejudice
Category: Classic Fiction
Year: 1811
ISBN: 9780582506206
Author: Charles Darwin
Title: The Origin of Species
Category: Non-Fiction
Year: 1856
ISBN: 9780517123201
Author: Charles Dickens
Title: The Old Curiosity Shop
Category: Classic Fiction
Year: 1841
ISBN: 9780099533474

Fetching results one row at a time

```
<?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 (Learning PHP, MySQL & JavaScript With jQuery, CSS & HTML5, 4th Edition 2015, Chapter 10)



Example 10-6. Inserting and deleting using sqltest.php

```
<?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>";
}
```

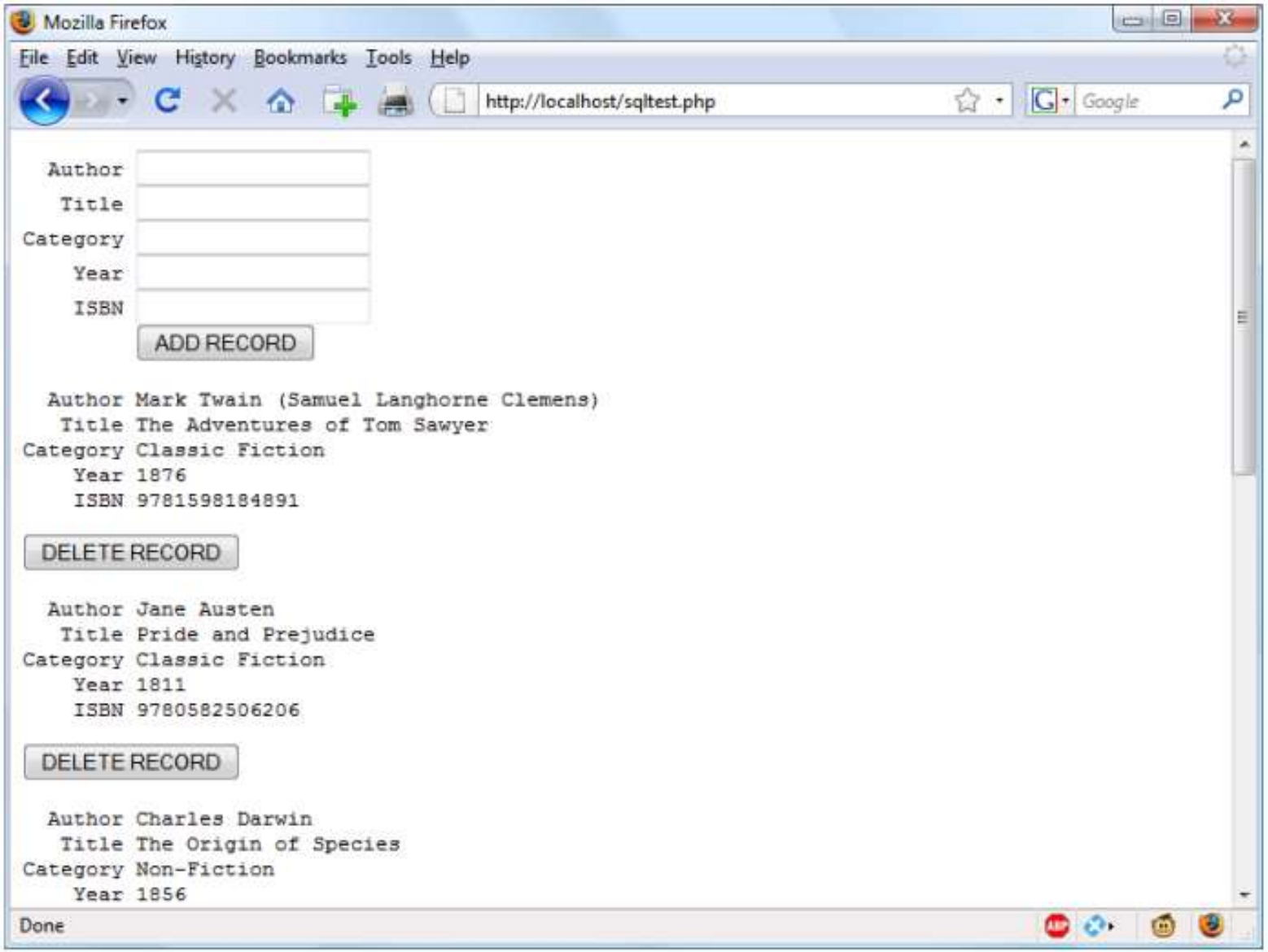
```
if (isset($_POST['author'])   &&
    isset($_POST['title'])    &&
    isset($_POST['category']) &&
    isset($_POST['year'])     &&
    isset($_POST['isbn']))
  {
    $author   = get_post($conn, 'author');
    $title    = get_post($conn, 'title');
    $category = get_post($conn, 'category');
    $year     = get_post($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;
```

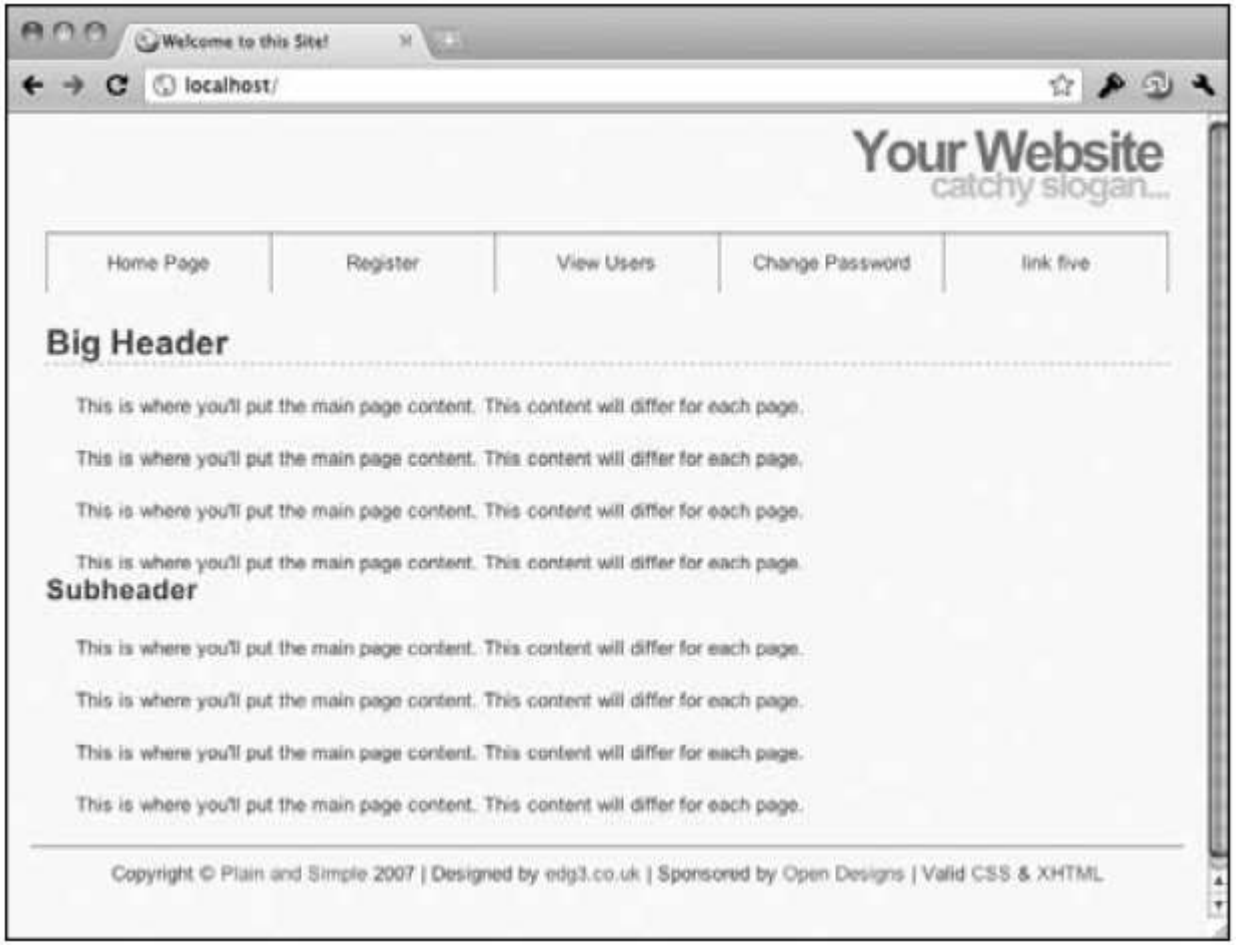
```
for ($j = 0 ; $j < $rows ; ++$j)
 {
   $result->data_seek($j);
   $row = $result->fetch_array(MYSQLI_NUM);
   echo <<< _END
  <pre>
   Author $row[0]
    Title $row[1]
  Category $row[2]
     Year $row[3]
     ISBN $row[4]
  </pre>
  <form action="sqltest.php" method="post">
  <input type="hidden" name="delete" value="yes">
  <input type="hidden" name="isbn" value="$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]);
  }
?>
```

Результат


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/sqltest.php
Google
Author
Title
Category
Year
ISBN
ADD RECORD
Author Mark Twain (Samuel Langhorne Clemens)
Title The Adventures of Tom Sawyer
Category Classic Fiction
Year 1876
ISBN 9781598184891
DELETE RECORD
Author Jane Austen
Title Pride and Prejudice
Category Classic Fiction
Year 1811
ISBN 9780582506206
DELETE RECORD
Author Charles Darwin
Title The Origin of Species
Category Non-Fiction
Year 1856
Done

Приклад. (Larry Ullman “PHP and MySQL for Dynamic Web Sites”, 2012, Chapter 9 - 12)



Створення БД сайту та таблиці користувачів

З файлу 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');
?>
```

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. -->
```

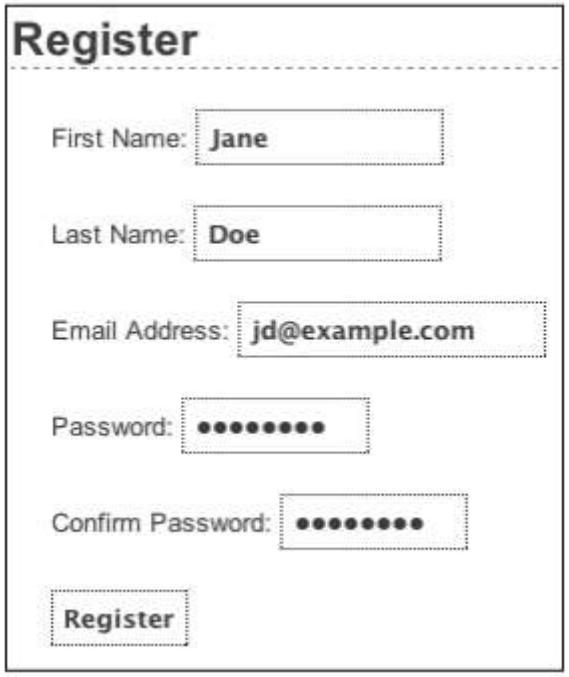
Стилі для навігації

```
#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



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']));
    }
. . .
```

Фрагменти файлу register.php

```
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

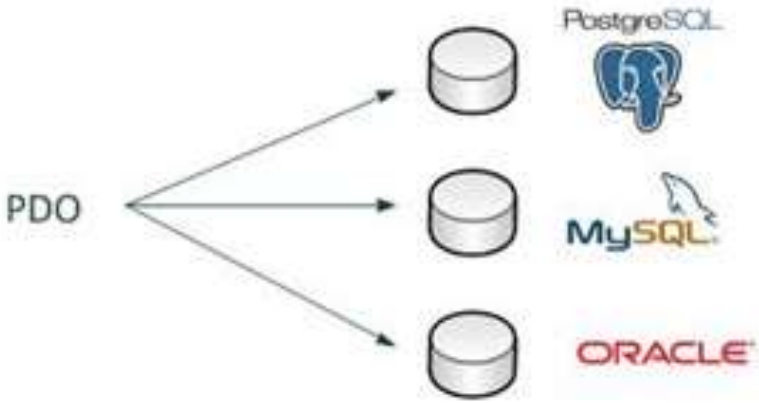
```
<?php
require ('includes/login_functions.inc.php');
require ('../mysqli_connect.php');
list ($check, $data) = check_login($dbc,
    $_POST['email'],$_POST['pass']);
if ($check) {
    setcookie ('user_id', $data['user_id']);
    setcookie ('first_name', $data['first_name']);
    redirect_user('loggedin.php');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

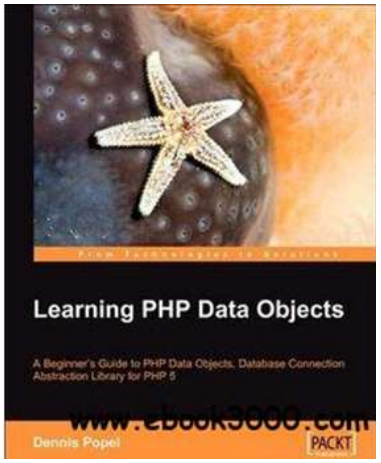
Фрагменти файлу login_functions.inc.php

```
<?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); }
```

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)

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";
    }
```

PDO. Виконання запитів (2)

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.

PDO. Виконання запитів (3)

```
$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));
```

PDO. Приклад (1) http://www.wa4e.com

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');
```

PDO. Приклад (2)

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);
```

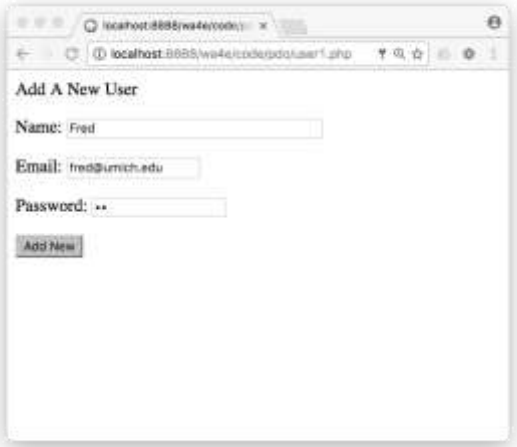
PDO. Приклад (3)

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
)
```

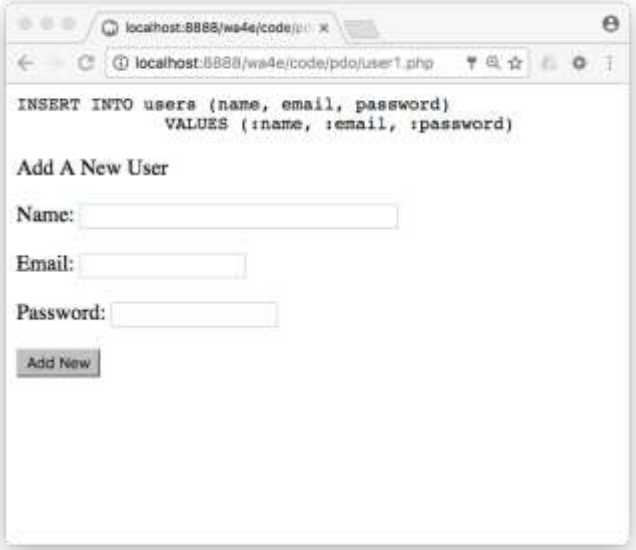
```
mysql> select * from users;
+---------+-------+---------------+----------+
| user_id | name  | email         | password |
+---------+-------+---------------+----------+
|       1 | Chuck | csev@umich.edu | 123      |
|       2 | Glenn | gg@umich.edu   | 456      |
+---------+-------+---------------+----------+
```



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>
```



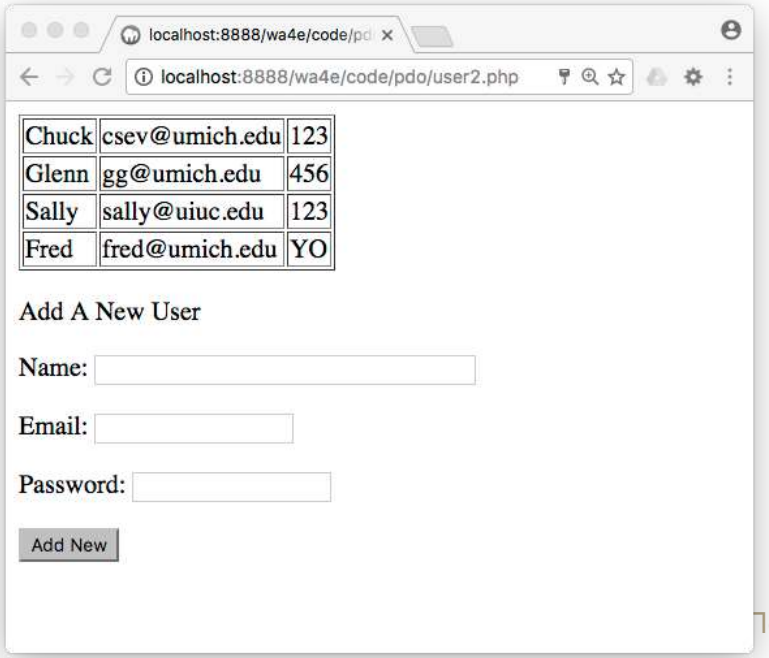
PDO. Приклад (4)

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> …
```



PDO. Приклад (5)

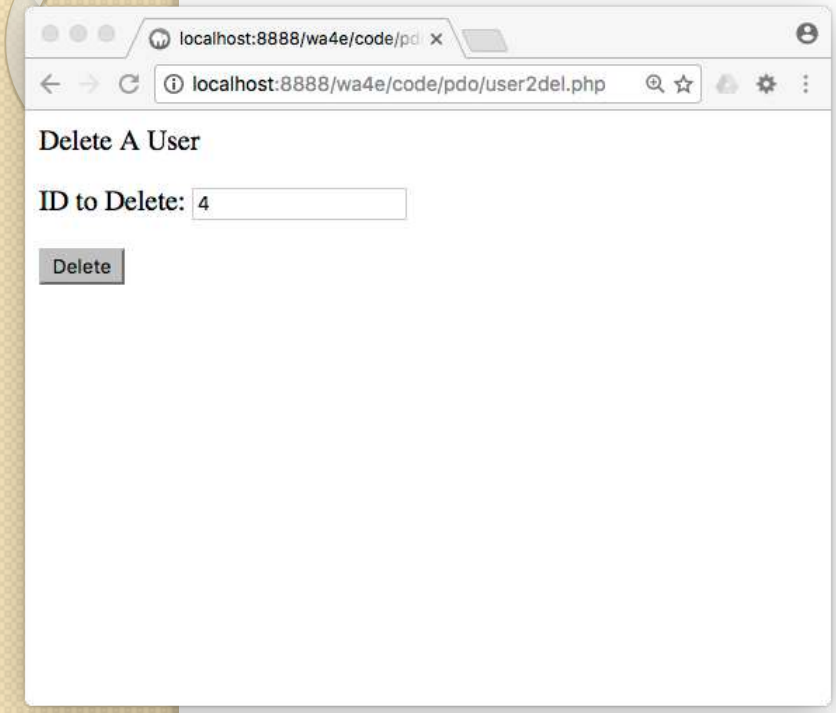
6)**Файл user2del.php**

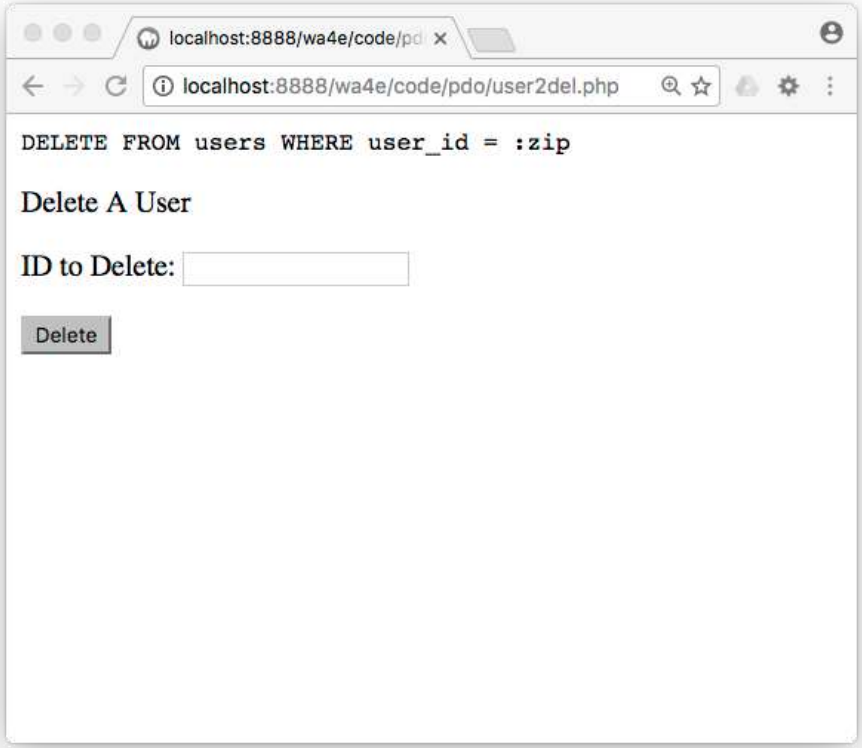
```
 <?php
require_once "pdo.php";

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>
```

PDO. Приклад (6)

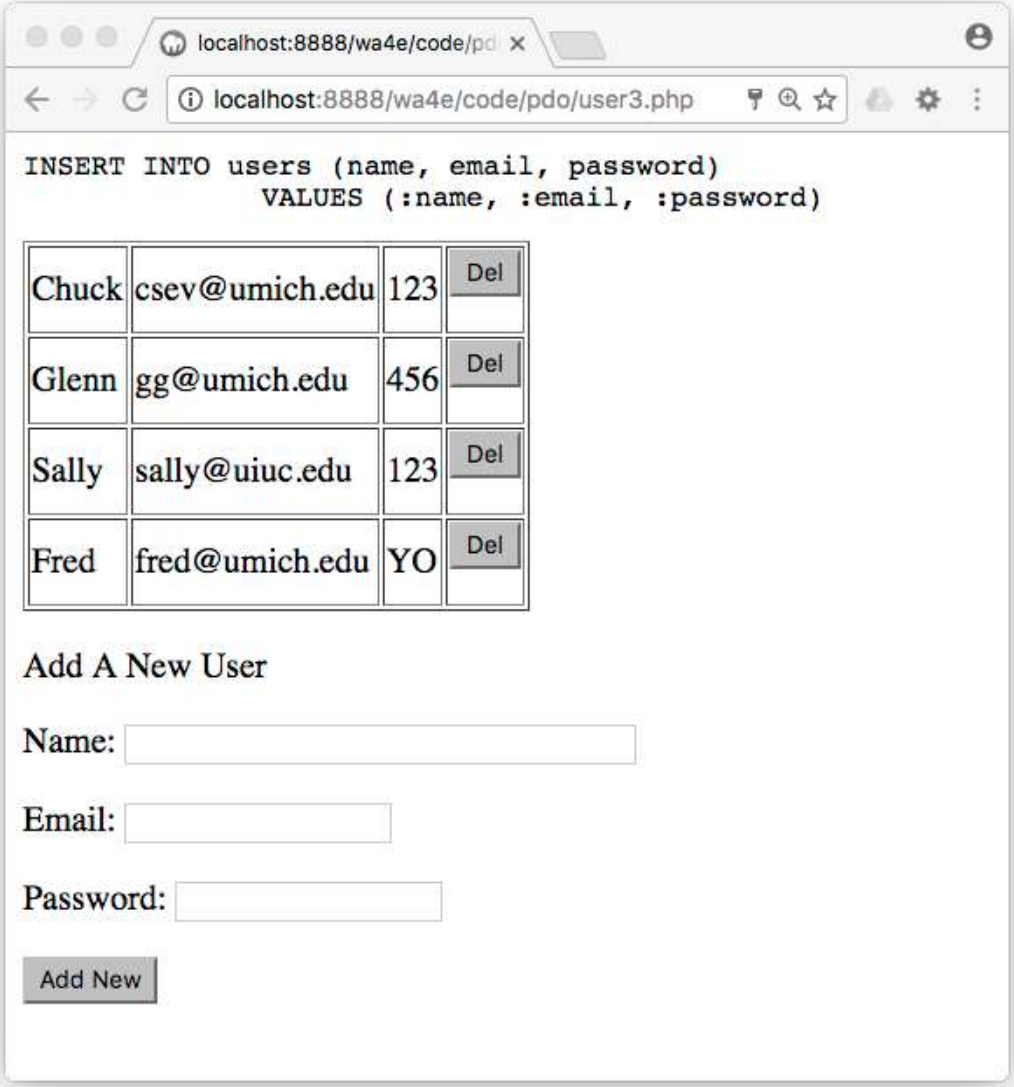
Результат:





PDO. Приклад (7)

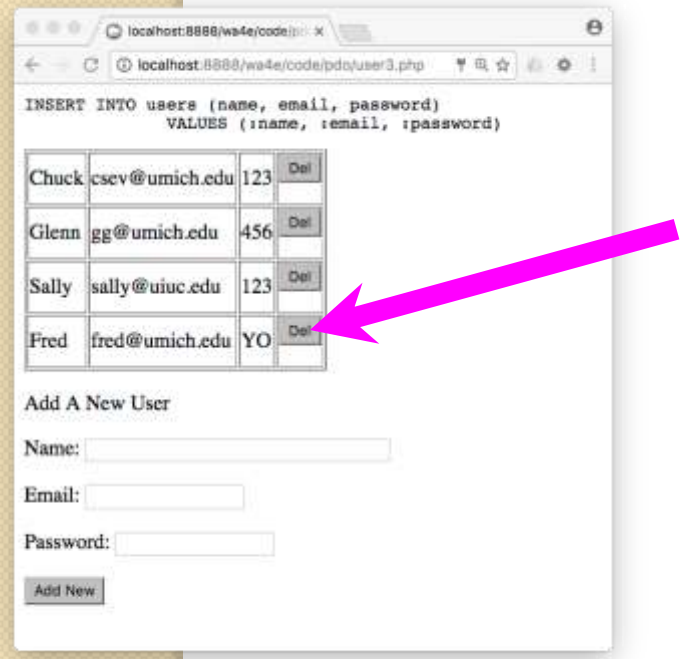
7)Файл user3.php

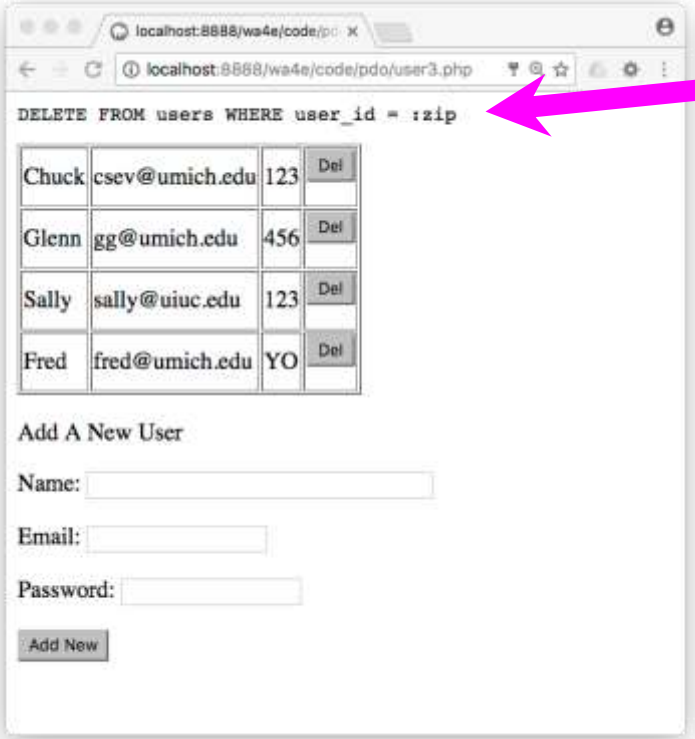


```php
<?php
require_once "pdo.php";
if ( isset($_POST['name']) && isset($_POST['email'])
      && isset($_POST['password'])) {
$sql = "INSERT INTO users (name, email, password)
    VALUES (:name, :email, :password)";
echo("<pre>\n".$sql."\n</pre>\n");
$stmt = $pdo->prepare($sql);
$stmt->execute(array(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





Cookies

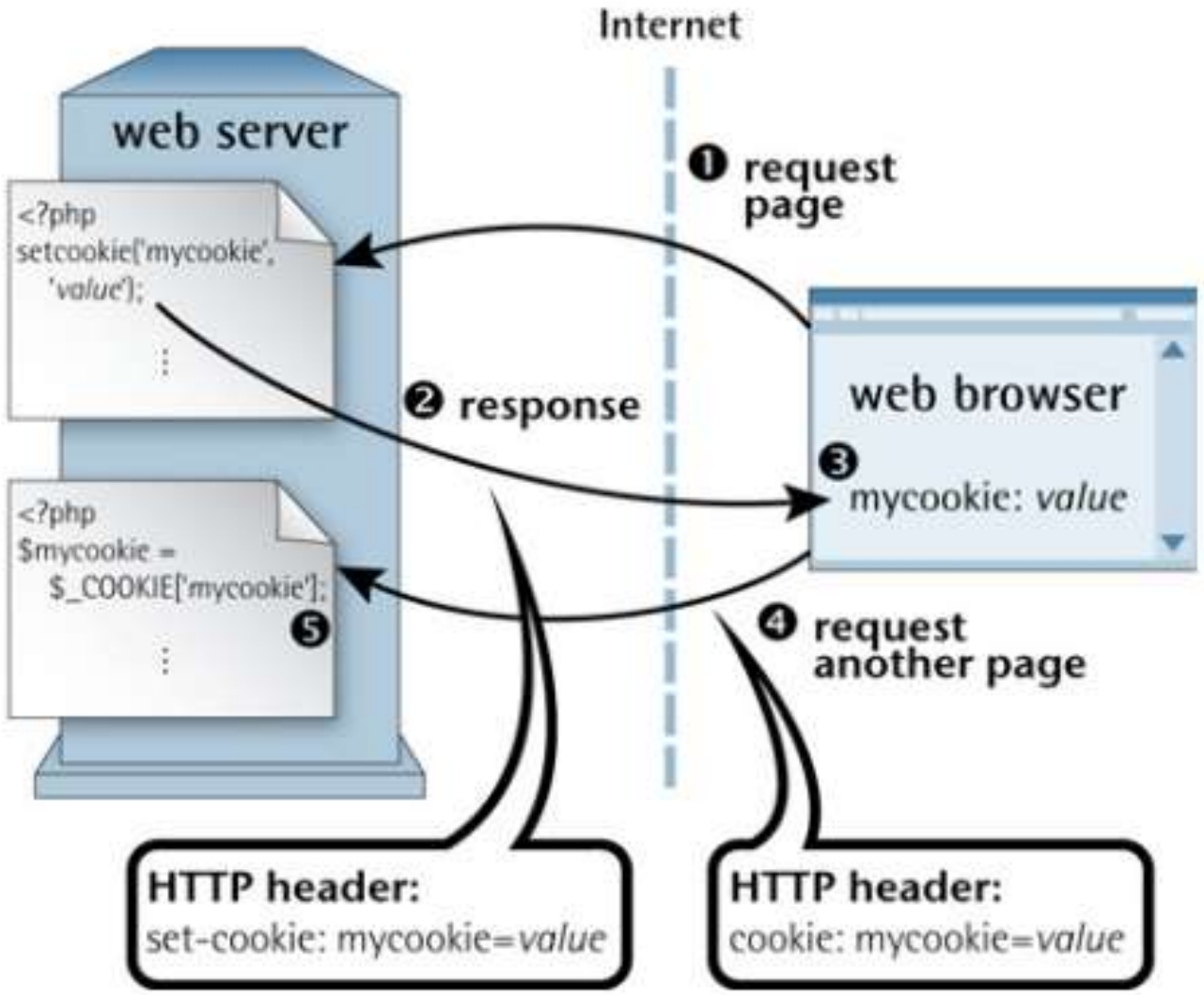
Зберігати деякі відомості про користувачів, наприклад ім'я, число відвідувань, дату останнього відвідування, можна за допомогою cookies.

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

Життєвий цикл cookie виглядає так:

1. Клієнт відправляє HTTP-запит серверу.
2. Сервер відправляє HTTP-відповідь, яка серед іншого включає в себе заголовок Set-Cookie: var = value.
3. При необхідності, клієнт переходить на іншу сторінку цього ж сервера, шляхом відправки нового HTTP-запиту, що включає в себе заголовок Cookie: var = value.
4. Сервер «впізнає» клієнта і відповідним чином реагує на його запит, якщо це передбачено

Життєвий цикл cookie



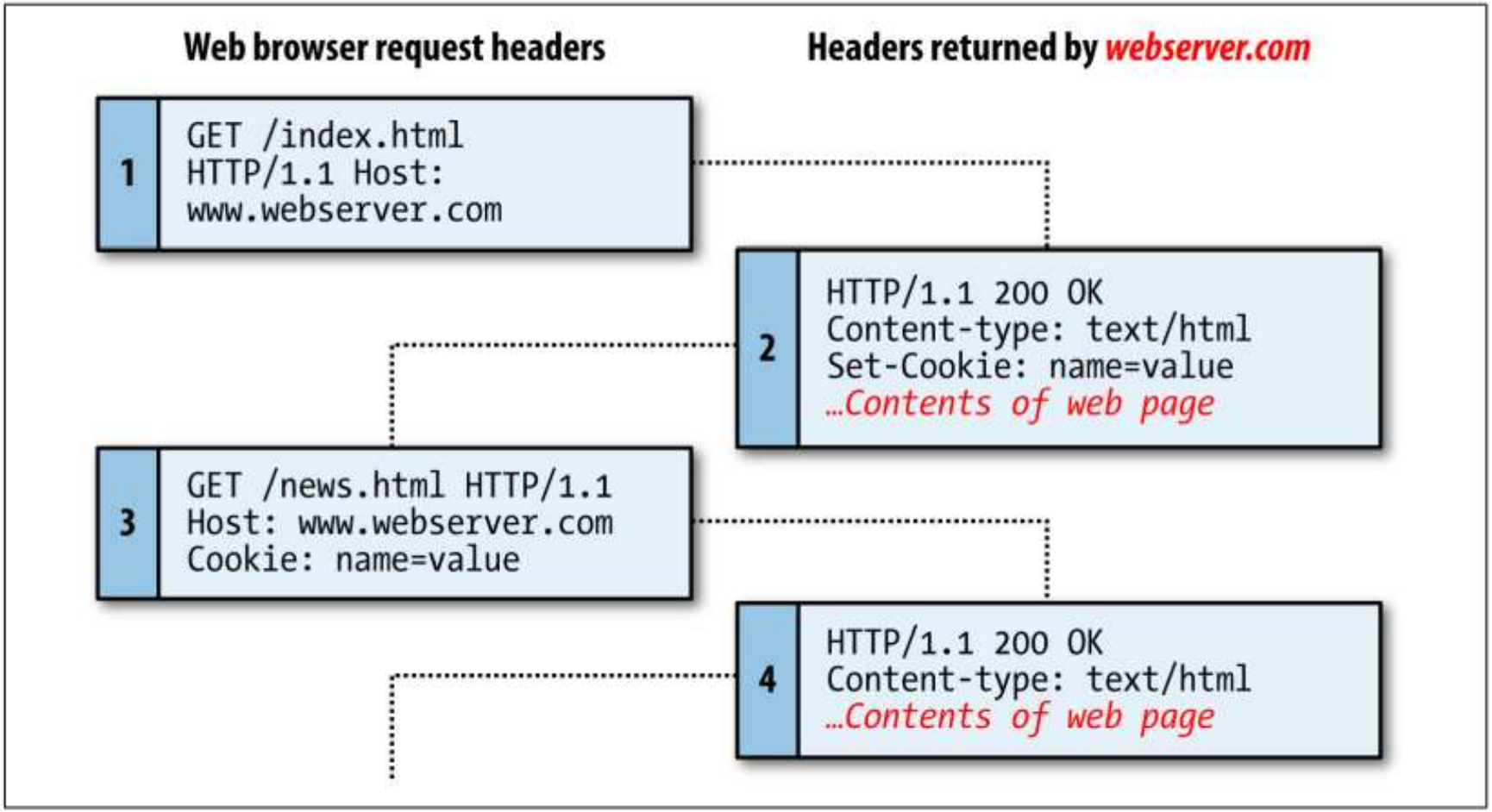


Figure 12-1. A browser/server request/response dialog with cookies

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 на протязі сеансу.

Доступ до cookie

Отримати значення cookie можна двома способами.

1. Усі cookies доступні через змінну оточення $_COOKIE, як $_COOKIE[‘ім'я_cookie’]
2. 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]");
?>
```

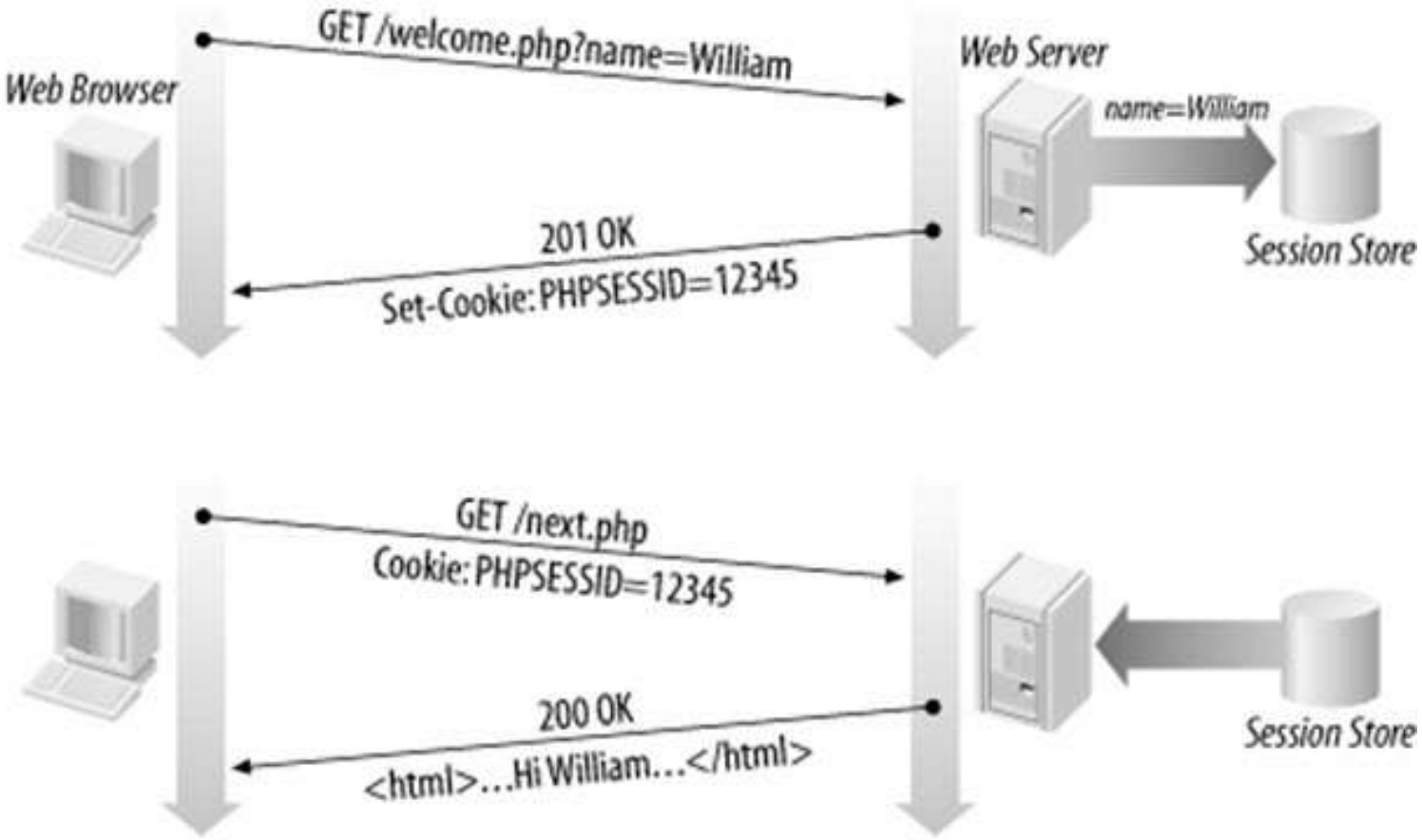
Робота з сесіями

- 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.

Відслідковування сеанса в PHP

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

Робота з сесіями



Робота з сесіями

```
<?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
<?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";
  }
?>
```

Файли. Режими відкриття файлів.

int fopen(string filename, string mode [, …])

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.

Файли. Читання та запис.

```
fread(int f, int length)
```

```
Приклад. $f = fopen("testfile.txt", "r+");
$str = fread($f, 10);//Читаємо перші 10 символів
$str = fread($f, 12); //Читаємо наступні 12
```

```
fgets(int f[, int length]); //Прочитати рядок
fgetc(int f); //Прочитати символ
fwrite(int f, string ws [, int length]) //Пишемо
fputs // Те ж, що і fwrite
```

```
Пряма робота з даними в файлі
//Можно читати файл не відкриваючи його
readfile(string filename);
//І навіть отримати вміст файлу у вигляді масиву
array file(string filename [, int
  use_include_path])
```

Маніпуляції всередині файла

```
//кінець файла
bool feof($f)
//Установка курсора
int fseek(int f, int offset [, int whence])
offset – кількість символів, на які треба
  перемістится. whence:
SEEK_SET – рух починається з початку файла;
SEEK_CUR – рух починається від поточної позиції;
SEEK_END – рух починається від кінця файла.
//Читаємо останні 10 знаків
fseek($f, -10, SEEK_END);
$s = fread($f, 10);
//Взнаємо поточну позицію
$pos = ftell($f);
rewind($f); //сброс курсора
```

Файли. Корисні функції.

```
file_exists("test.txt"); //Чи існує файл?
is_file("test.txt"); //Це файл чи директорія?
is_dir("/tmp");
//Перевірка статуса
is_readable("test.txt");
is_writable("test.txt");
is_executable("test.txt");
filesize("test.txt"); //Взнаємо розмір файла
$atime = fileatime("test.txt");
date("d M Y", $atime); //Дата останнього звертання
$mtime = filemtime("test.txt");
date("d M Y", $mtime); //Дата зміни файла
$ctime = filectime("test.txt");
date("d M Y", $ctime); //Дата створення файла(Windows)
copy(string source, string destination);//Копіювання файла
rename(str oldname, str newname);
  //Перейменування файла
unlink(string filename);
  //Видалення файла
```

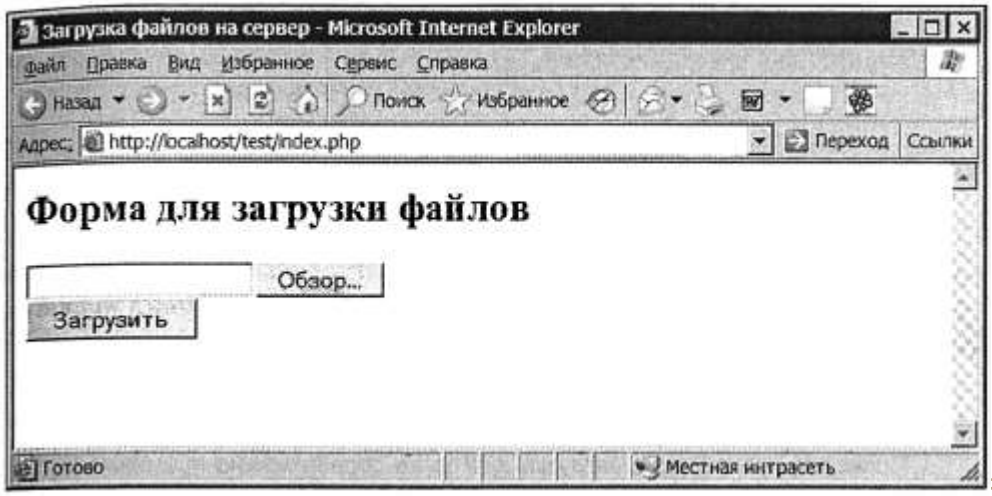
Директорії. Корисні функції.

```
//Поточна директорія
getcwd()
//Зміна директорії
chdir(string dirname)
//Створення директорії
mkdir(string dirname[, int mode])//!0777
//Видалення директорії
rmdir(string dirname)//тільки пуста!
//Відкриваємо
$dirhandle = opendir(string dirname)
//Читаємо
readdir($dirhandle)
//Закриваємо
closedir($dirhandle)
```

Завантаження файлів на сервер

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

```
<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>
```



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:

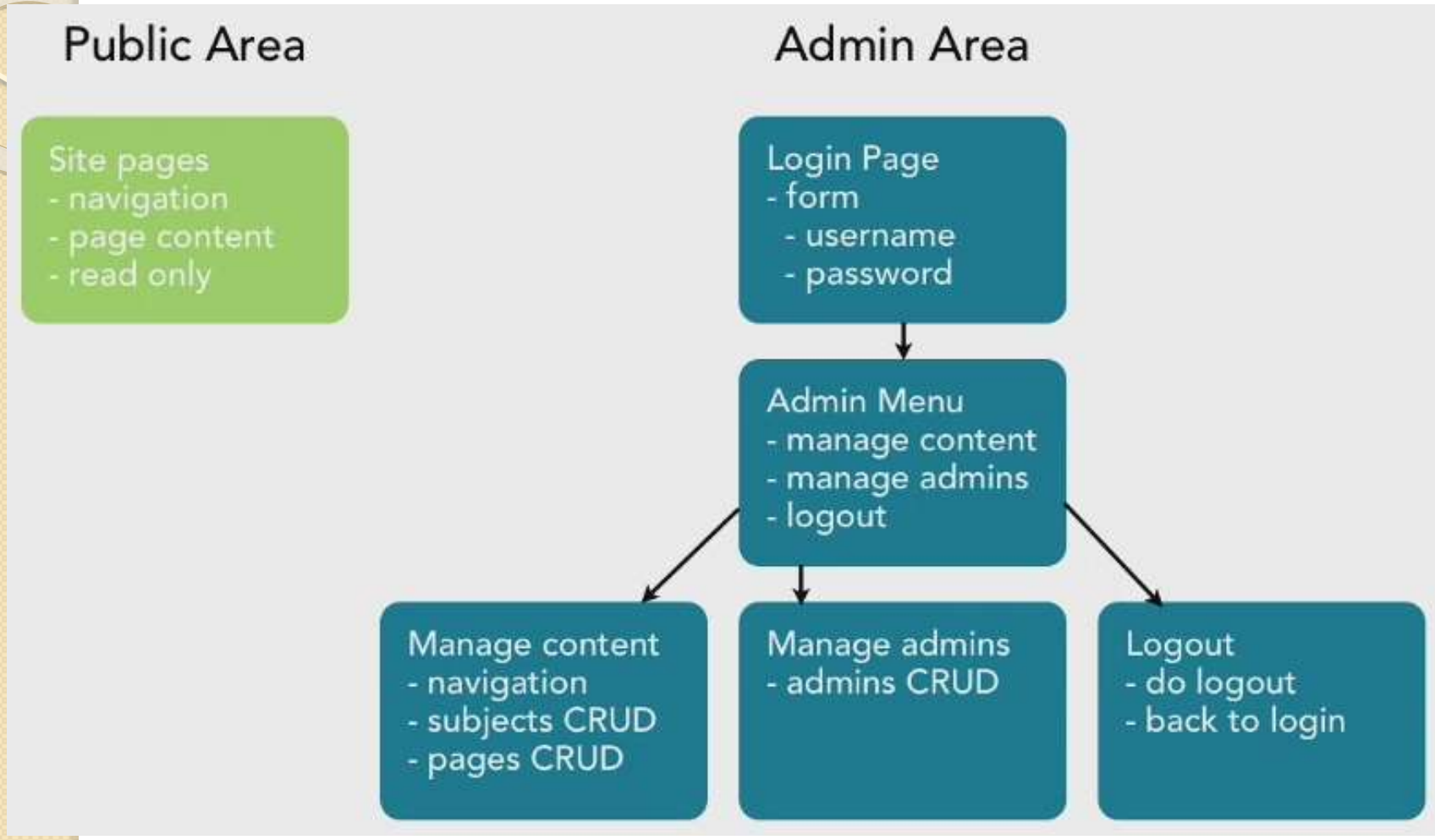
- $_FILES['filename']['name'] – ім'я файлу на клієнтській машині
- $_FILES['filename']['size'] – розмір завантаженого файлу в байтах
- $_FILES['filename']['type'] – MIME-тип файлу
- $_FILES['filename']['tmp_name'] – ім'я тимчасового файлу, в якому зберігається завантажений файл

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>
```

Building a Content Management System (CMS)

(Lynda - PHP with MySQL Essential Training)



Building a Content Management System (2)

Building a Content Management System (3)

Building the CMS database

subjects	pages	admins
id	id	id
menu_name	subject_id	username
position	menu_name	hashed_password
visible	position	
	visible	
	content	

```
mysql> CREATE TABLE admins (
    ->   id INT(11) NOT NULL AUTO_INCREMENT,
    ->   username VARCHAR(50) NOT NULL,
    ->   hashed_password VARCHAR(60) NOT NULL,
    ->   PRIMARY KEY (id)
    -> );
```

Система шаблонів Smarty

- **Smarty (англ. розумний) – це компілюючий обробник шаблонів до PHP, один з інструментів, що дозволяють відділити прикладну логіку і дані від представлення в дусі концепції MVC.**
- **Мова шаблонів Smarty розширює HTML smarty-тегами, які вбудовуються в документ**.
- Ці теги можуть представляти собою PHP-змінні (наприклад {$variable}), функції (наприклад {insert file=header.tpl}) або базові конструкції (наприклад {if …} … {else} … {/if}).
- Де скачати: http://www.smarty.net/download(Остання версія Smarty 3.1.30 , Aug 7th, 2016)
- Документація: http://www.smarty.net/documentation (російською мовою - тільки по Smarty 2)

Smarty

- Після розархівування ми бачимо директорію libs з необхідними бібліотеками. Найбільш цікавий файл Smarty.class.php, який містить клас Smarty. Він підтримує основну функціональність.
- Директорію libs треба скопіювати в директорію свого сайту (наприклад localhost) і переіменувати в smarty3
- Підключити файл Smarty.class.php
- Створити 4 директорії:

  templates – для зберігання шаблонів

  templates_c – для скомпільованих шаблонів

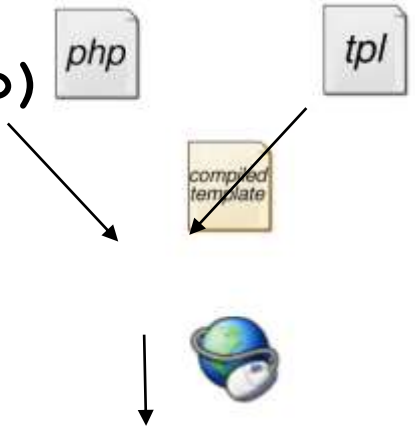
  configs – для конфігураційних параметрів

  cache – для зберігання кешованих файлів

Smarty

Приклад. Файл index.php

```
<?php
require_once('smarty3/Smarty.class.php)
$smarty = Smarty();
$smarty->template_dir = 'templates';
$smarty->compile_dir = 'templates_c';
$smarty->config_dir = 'configs';
$smarty->cache_dir = 'cache';
// Отримуємо дані з БД (у вигляді асоціативного
  масиву)
$news = $DB->query("SELECT * FROM news);
// Передаємо масив в шаблонізатор
$smarty->assign('news', $news);
// Виводимо на екран шаблон з директорії
  templates
$smarty->display(news.tpl);
?>
```



Smarty

Фрагменти файлу news.tpl

```
<html>
. . .
{foreeach from=$news item=item}
<h1>Заголовок: {$item.title}</h1>
<b>Текст новини:</b>
<div class="news_content">
{$item.content}
</div>
{/foreach}
. . .
```

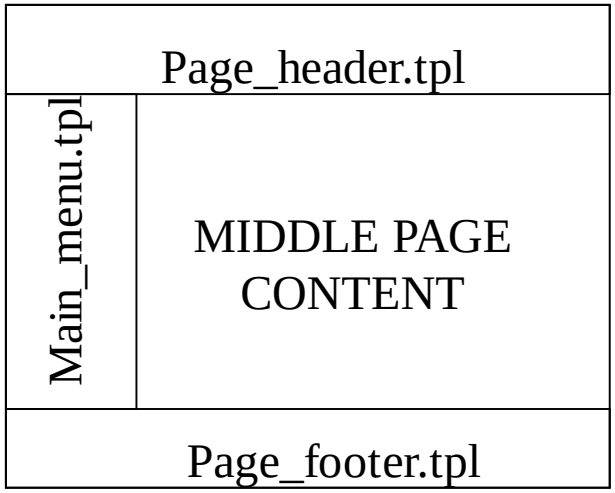
Клас Smarty відповідальний за:

- Читання шаблону
- Підстановку змінних в шаблонні файли
- Показ файлів шаблонів (*.tpl)

Корисне посилання: PHP Шаблонизатор Smarty
https://www.youtube.com/watch?v=LuH916Is87A

Smarty

- Сторінка поділяється на декілька шаблонів
- Формується головна сторінка main_page.tpl



Корисні посилання:

Робин Никсон Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript , Питер, 2011 (стр. 286 Работа с шаблонами в системе Smarty)

5 популярных PHP-шаблонизаторов (8 мая 2017)
https://quasi-art.ru/library/it/5-top-php-template-engines
(Blade, Mustache, Smarty, Twig, Volt)

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.

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.

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.

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 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

Жадібність і лінивість

- 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>"
         }
}
```

Регулярні вирази в PHP. Метасимволи

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

Регулярні вирази, сумісні з мовою Perl

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

Найбільш вживані функції:

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';
   }
?>
```

Регулярні вирази, сумісні з мовою Perl

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";
// /$1 посилання на результат, який був знайдений першими
    круглими дужками
print preg_replace($pattern, $replacement, $str);
?>
```

Авторизація користувачів в PHP

39.11 Система реєстрації (стор. 871) – розібрати самостійно

39.12 Система авторизації за допомогою сесій (стор. 880) – розібрати самостійно

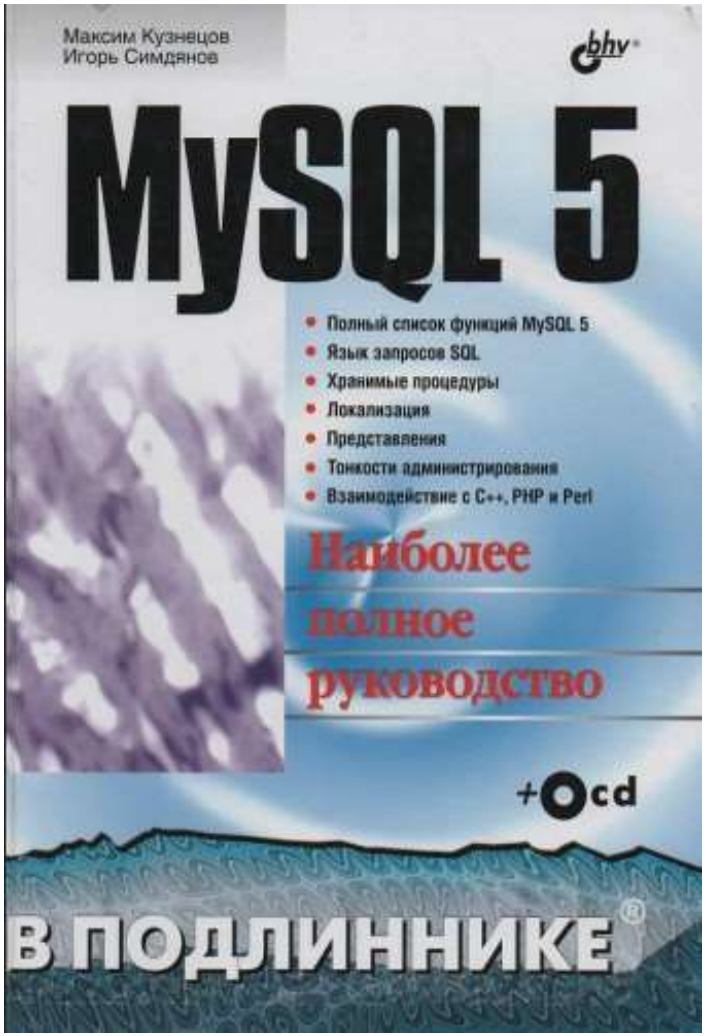
39.13 Базова HTTP-авторизація – розглянемо

39.15 Посторінкова навігація

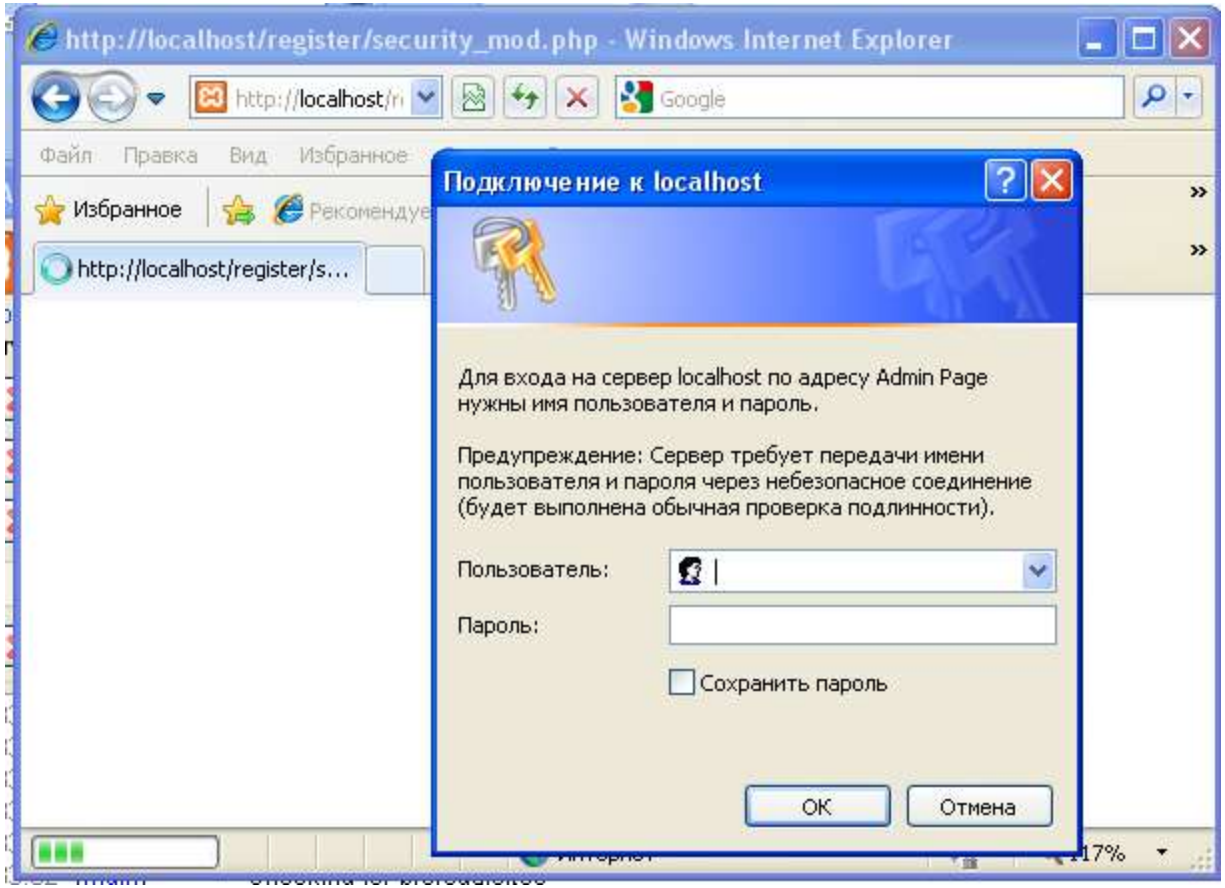
39.16 Алфавітна навігація

39.21 Зберігання зображень в БД

Вивчайте коди з диску до книги.



Базова HTTP-авторизація



Базова HTTP-авторизація

Для реалізації даного виду авторизації необхідно надіслати браузеру клієнта HTTP-заголовки

```
www-Authenticate: Basic realm="Admin Page"
HTTP/1.0  401 Unauthorized
```

які виведуть форму для введення імені користувача та пароля. Ім'я користувача буде поміщено браузером в змінну $_SERVER['PHP_AUTH_USER'], а пароль – в $_SERVER['PHP_AUTH_PW'].

Зауваження: Ці змінні доступні тільки у випадку, коли PHP встановлений у якості модуля, а не CGI-застосунку.

Давайте створимо файл **security_mod.php**, включення якого за допомогою директиви **include** буде давати захист сторінки паролем.

Файл security_mod.php

```
<?php
  require_once("config.php");
  // Якщо користувач не авторизувався - авторизуємось
  if(!isset($_SERVER['PHP_AUTH_USER']))
  {
    Header("WWW-Authenticate: Basic realm=\"Admin Page\"");
    Header("HTTP/1.0 401 Unauthorized");
    exit();
  }
  else
  {
      $_SERVER['PHP_AUTH_USER'] =
  mysql_real_escape_string($_SERVER['PHP_AUTH_USER']);
      $_SERVER['PHP_AUTH_PW'] =
  mysql_real_escape_string($_SERVER['PHP_AUTH_PW']);
    $query = "SELECT pass FROM userlist
              WHERE name='".$_SERVER['PHP_AUTH_USER']."'";
    $lst = @mysql_query($query);
```

Файл security_mod.php (закінчення)

```
// Якщо помилка в SQL-запиті - видаємо вікно
    if(!$lst)
    {
      Header("WWW-Authenticate: Basic realm=\"Admin Page\"");
      Header("HTTP/1.0 401 Unauthorized");
      exit();
    }
    // Якщо такого користувача немає - видаємо вікно
    if(mysql_num_rows($lst) == 0)
    {
      Header("WWW-Authenticate: Basic realm=\"Admin Page\"");
      Header("HTTP/1.0 401 Unauthorized");
      exit();
    }
    // Якщо усі перевірки пройдено, порівнюємо хеші паролей
    $pass = @mysql_fetch_array($lst);
    if(md5($_SERVER['PHP_AUTH_PW']) != $pass['pass'])
    {
      Header("WWW-Authenticate: Basic realm=\"Admin Page\"");
      Header("HTTP/1.0 401 Unauthorized");
      exit();
    }
  }
?>
```

Об’єктно-орієнтований PHP

- Об'єктно-орієнтований код працює повільніше, ніж процедурний код, але полегшує моделювання та керування складними структурами даних.
- PHP підтримує об'єктно-орієнтоване програмування з версії 3.0, і з тих пір ця об'єктна модель була значно розширена і реформована.

Декларування класів та створення об'єктів

```
<?php
class ExampleClass
{
// class code
}
```

```
<?php
$exampleObject = new
    ExampleClass();
$anotherObject = new
    ExampleClass;
```

Синтаксис та обмеження для успадкування

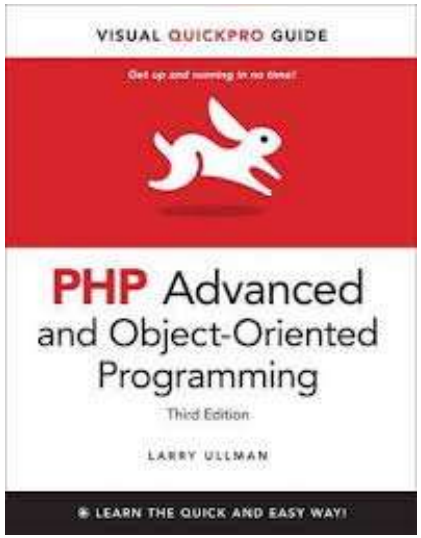
Concept	Syntax	Limitation
Inherit from a class	class A extends A_Parent	Class may have only one parent
Interface inheritance	Interface A extends B, C	Interface can inherit multiple interfaces
Inherit from an abstract class	Interface A extends B, C	Interface can inherit multiple interfaces
Implement interface	class A implements A_ Interface	Class can implement multiple interfaces
Trait	class Foo { use A_trait; }	Class can use multiple traits

ООП в PHP

Оголошення класу товару (файл commodity.php)


```
<?php
  class Commodity {
      public $name; // Назва
      public $category; // Категорія
      public $price; // Ціна
      public $availability; // Наявність
   function __construct($name, $category, $price
= null, $availability = False) {
       echo 'Запущено конструктор...<br/>';
       $this->name = $name;
       $this->category = $category;
       $this->price = $price;
       $this->vailability = $availability;
   }
```

```
function __destruct() {
        echo 'Запущено деструктор...<br/>';
    }
function getPrice() {
        return (is_null($this->price) ? 'N/A' :
  $this->price);
    }
function setPrice($new_price) {
        $this->price = $new_price;
    }
}
?>
```



ООП в PHP

Створення об'єктів

```
<?php
require_once 'commodity.php';
$obj = new Commodity('Розробка на PHP 5',
  'book');
echo $obj->name.'-'.$obj->getPrice().'<br/>';
$obj->setPrice(230.0);
echo $obj->name.'-'.$obj->getPrice().'<br/>';
?>
```

Клонування об’єктів

```
<?php
require_once 'commodity.php';
$obj1= new Commodity(‘Розробка на PHP
  5‘,’book’);
$obj1->price = 150.0;
$obj2 = $obj1; // Копіювання вказівників
$obj2->price = 230.0;
echo $obj1->price.'<br/>'; // 230.0
echo $obj2->price.'<br/>'; // 230.0
?>
```

Для клонування: **`$obj2 = clone $obj1;`**

Конструктор копіювання можна перевантажити, додавши в клас свою функцію __clone(). (Ця ф-я без аргументів, але може звертатися до вхідного об'єкта через **$this** та до копії через **$that**)

Простори імен

- Починаючи з PHP 5.3, крім класів доступний ще один спосіб організації проекту - простір імен.
- Він дозволяє організувати код у вигляді віртуальної ієрархії, що нагадує файлову систему.
- Як файли з однаковими іменами ізольовані, перебуваючи в різних каталогах, так класи, функції і константи PHP можуть бути ізольовані по різних просторах імен.
- Це дозволяє уникати конфліктів зі сторонніми бібліотеками, а також полегшує пошук і завантаження файлів.

Повне визначення імен в просторі імен

- Якщо ви працюєте в просторі імен, то інтерпретатор припускає, що імена відносяться до поточного простору імен.

```
<?php
namespace MyApp\Helpers;
class Formatters
{
public static function asCurrency($val) {
// statement
}
}
```

- Якщо ми хочемо використати цей клас з іншого простору імен, нам потрібно надати повний шлях до класу, як у цьому прикладі:

```
<?php
namespace MyApp\Lib;
echo MyApp\Helpers\Formatters::asCurrency(10);
```

Крім того, ви можете використовувати оператор **use** для імпорту простору імен, щоб вам не довелося постійно використовувати довгий формат:

```
<?php
namespace MyApp\Lib;
use MyApp\Helpers\Formatters;
echo Formatters::asCurrency(10);
```

Автозавантаження класів

- Класи повинні бути визначені перед їх використанням, але ви можете використовувати автозавантаження для завантаження класів, коли це необхідно.
- Стандарт кодування PSR-4 визначає, де PHP шукатиме клас.
- Автозавантаження на PHP виконується функцією **spl_autoload_register ().**

```
<?php
function my_autoloader($class) {
include 'classes/' . $class . '.class.php';
}

spl_autoload_register('my_autoloader');
```

ООП в PHP

Успадкування

```
<?php
require_once 'commodity.php';
class Book extends Commodity {
    public $authors;
    function __construct($name, $authors, $category,
                         $price = null,
                         $availability = False) {
        parent::__construct($name, $category, price,
                            $availability)
        $this->authors = $authors;
    }
    function getAuthors() {
        return $this->authors;
    }
}
?>
```

Статичні члени класу

```php
<?php
  class Visitor
  {
    private static $visitors = 0;
    function __construct()
    {
        self::$visitors++;
    }
   static function getVisitors()
   {
        return self::$visitors;
    }
  }
```

```php
$visits = new Visitor();
   echo
Visitor::getVisitors()."<br />";

 $visits2 = new Visitor();
    echo
   Visitor::getVisitors()."<br />";
 ?>
```

The results are as follows:

1

2

Перевизначення методів

```
<?php
  $object = new Son;
  $object->test();
  $object->test2();
  class Dad
  {
    function test()
    {
      echo " [Class Dad] I am
    your Father<br> ";
    }
  }
```

```
class Son extends Dad
 {
    function test()
    {
      echo " [Class Son] I am
    Luke<br> ";
    }
    function test2()
    {
      parent::test();
    }
 }
?>
```

Magic (__*) Methods. Методи доступу __get() і __set()

Методи __get() і __set() викликаються при звертанні до недоступних властивостей класу. Наприклад, якщо ми б вирішили зберігати всі властивості товару в масиві, то ці методи нам би знадобились.

```
<?php
class Commodity {
    private $properties;
    function __set($name, $value) {
       echo "Задання нової властивості $name = $value";
        $this->properties[$name]=$value;
    }
    function __get($name) {
        echo "Читання значення властивості", $name;
        return $this->properties[$name];
    }
}
?>
... $book = new Book(...);
$book->weight = 100;...$weight=$book->weight;
```

ООП в PHP

Фінальні методи. Метод, при визначенні якого використано ключове слово **`final`**, не можна перевантажувати в класах-спадкоємцях. Якщо **`final`** використано при визначенні самого класу, то породження від нього інших класів стає неможливим.

Методи **`__toString()`** та **`__call()`**

Метод **`__toString()`** викликається інтерпретатором PHP, якщо йому необхідно перетворити об'єкт в рядок (тобто при кожному виклику функцій echo() та print()). Перевантаживши його, ви можете вказати, як об'єкт повинен виглядати в рядковому вигляді.

Метод **`__call()`** викликається, якщо викликаний метод в класі не визначено.

Інші спеціальні методи (magic methods)

`__sleep()`, **`__wakeup()`** – потрібні при пакуванні/розпакуванні

`__autoload()` – описує, як підключаються файли з визначенням класів

ООП в PHP

```
class Car{
 public function __call($name, $arguments){
        echo "hello world!";
    }
}
$car = new Car();
$car->hello();
```

Метод hello() не визначений.

Тому викликається метод __call()

Корисні посилання: Відеокурс **PHP with MySQL Beyond the Basics, intermediate level, 2009**

http://www.lynda.com/PHP-tutorials/php-with-OOP-beyond-the-basics/653-2.html

ООП в PHP

```
class Car
{
    protected $_color;
    protected $_model;
    public function __call($name, $arguments)
    {
        $first = isset($arguments[0]) ? $arguments[0] :
  null;
        switch ($name)
        {
              case "getColor":
                 return $this->_color;
              case "setColor":
                  $this->_color = $first;
                  return $this;
              case "getModel":
                  return $this->_model;
              case "setModel":
                  $this->_model = $first;
                  return $this;
         }
    }
}
$car = new Car();
$car->setColor("blue")->setModel("b-class");
echo $car->getModel();
```

ООП в PHP

Абстрактні методи та класи

```
<?php
abstract class Commodity {
    ...
    abstract function printProperties();
}
?>
```

Інтерфейси

```
interface Int1 {
    function funct1();
    function funct2();
    . . .
}
class MyClass implements Int1, Int2 {
    public function funct1() {
        echo "Виклик метода 1";
    }
    public function funct2() {
        echo "Виклик метода 2";
    }
}
```

Приклад

```
class Person
{
  private $firstname;
  private $lastname;
  public function getFullName()
  {
    $fullname = $this->firstname." ".$this->lastname;
    return $fullname;
  }
  public function setFullName($aFirstname, $aLastname)
  {
    $this->firstname = $aFirstname;
    $this->lastname = $aLastname;
  }
}
```

```
interface JobCodes
{
    const PAYROLL = "01";
    const MANAGER = "02";
    const RETAIL  = "03";
}

interface StandardFunctions
{
    public function getJobTitle($aJobCode);
    public function showFullName();
}
```

```
class Employee extends Person
    implements JobCodes, StandardFunctions
{
    private $employeeId;
    private $jobcode;
    public function __construct($aFirstname, $aLastname,
                                $aEmpId, $aJobcode)
    {
        $this->setFullName($aFirstname, $aLastname);
        $this->employeeId = $aEmpId;
        $this->jobcode = $aJobcode;
    }
    function getJobTitle($aJobCode) {  }
    function showFullName() { }
}
```

ООП в PHP. Вбудовані інтерфейси.

```
interface Iterator {
  public function rewind();// Returns the iterator the
   beginning
  public function next();   // Get to the next member
  public function key();// Get the key of the current
   object.
  public function current();//Get the value of the
   current object
  public function valid();// Is the current index valid?
}
```

Будь-який клас, який реалізує інтерфейс Iterator, може використовуватись в циклах, до них буде застосовано ітератор.

Приклад

```
<?php
class iter implements iterator {
    private $items;
    private $index = 0;
    function __construct(array $items) {
        $this->items = $items;
    }
    function rewind() {
        $this->index = 0;
    }
    function current() {
        return ($this->items[$this-
  >index]);
    }
    function key() {
        return ($this->index);
    }
```

```
function next() {
        $this->index++;
        if (isset($this->items[$this->index])) {
            return ($this->items[$this->index]);
        } else {
            return (NULL);
        }
    }
function valid() {
        return (isset($this->items[$this-
  >index]));
    }
}
$x = new iter(range('A', 'D'));
foreach ($x as $key => $val) {
    print "key=$key\t value=$val\n";
}
```

Результат

```
key=0   value=A
key=1   value=B
key=2   value=C
key=3   value=D
```

ООП в PHP

Константи класу

```
class MyClass {
    const CONSTANT = 'value';
}
```

Звертання до констант класу

```
echo MyClass::CONSTANT; // Виводе "value“
```

Статичні властивості та методи - нагадують C++

Порівняння об'єктів

При використанні операції порівняння (==) об'єкти рівні, якщо вони є екземпляри одного класу, мають однаковий набір атрибутів та однакові значення цих атрибутів.

При використанні операції перевірки ідентичності(===) об'єкти вважаються ідентичними, якщо вони посилаються на один і той же екземпляр одного класу.

Анонімні класи

- PHP 7 ввів анонімні класи, які дозволяють вам визначити клас на льоту і створити з нього об'єкт. Ось простий приклад використання анонімного класу:

```
<?php
$object = new class('argument') {
public function __construct(string $message) {
echo $message;
}
};
```

Рефлексія

- В PHP функції API рефлексії дозволяють вам перевіряти елементи PHP під час виконання та отримувати інформацію про них.
- API Reflection був введений в PHP 5.0, а за замовчуванням був включений в PHP 5.3.
- Одне з найпоширеніших місць, де використовується відображення, - це unit-тестування.
- Один з прикладів того, де полегшує відображення, це тестування приватної властивості в класі. Ви можете використовувати рефлексію, щоб зробити доступною приватну властивість.

```
<?php
$reflectionObject = new ReflectionClass('Exception');
print_r($reflectionObject->getMethods());
```

Обробка виняткових ситуацій (виключень)

PHP 5 додає парадигму обробки виключень, вводячи структуру try/throw/catch. Вам залишається тільки створити клас, який успадковує клас винятків Exception

```
class Exception {
/* Properties */
protected string $message ;
protected int $code ;
protected string $file ;
protected int $line ;
/* Methods */
public __construct ([ string $message = "" [, int $code = 0
                                  [, Exception $previous = NULL ]]] )
final public string getMessage ( void )
final public Exception getPrevious ( void )
final public int getCode ( void )
final public string getFile ( void )
final public int getLine ( void )
final public array getTrace ( void )
final public string getTraceAsString ( void )
public string __toString ( void )
final private void __clone ( void )
}
```

Обробка виняткових ситуацій

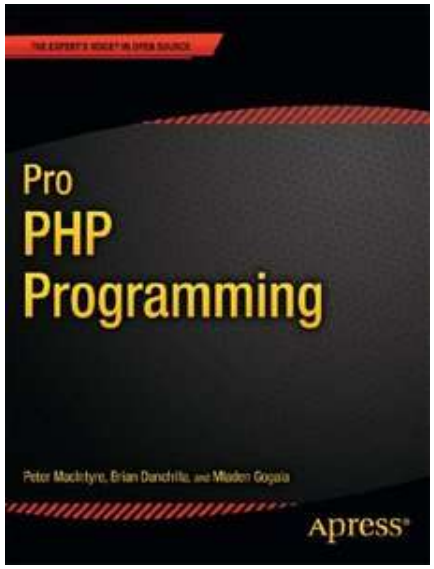
Приклад 1.

```
<?php
  try{
  @$fp = fopen("file.txt","w");
  if(!$fp)throw new Exception("Неможливо відкрити файл");
  // Запис даних
  . . .
  fclose($fp);
  }
  catch(Exception $ext){
      echo "Помилка в рядку ", $ext->getLine();
      echo $ext->getMessage();
  }
?>
```

Обробка виняткових ситуацій

Приклад 2. (Pro PHP Programming, 2011, p.22)

```
<?php
class NonNumericException extends Exception {
    private $value;
    private $msg = "Error: the value %s is not
  numeric!\n";
    function __construct($value) {
        $this->value = $value;
    }
    public function info() {
        printf($this->msg, $this->value);
    }
}
try {
    $a = "my string";
    if (!is_numeric($argv[1])) {
        throw new NonNumericException($argv[1]);
    }
```



Обробка виняткових ситуацій

```
if (!is_numeric($argv[2])) {
        throw new NonNumericException($argv[2]);
    }
    if ($argv[2] == 0) {
        throw new Exception("Illegal division by
  zero.\n");
    }
    printf("Result: %f\n", $argv[1] / $argv[2]);
}
catch(NonNumericException $exc) {
    $exc->info();
    exit(-1);
}
catch(Exception $exc) {
    print "Exception:\n";
    $code = $exc->getCode();
    if (!empty($code)) {
        printf("Erorr code:%d\n", $code);
    }
    print $exc->getMessage() . "\n";
    exit(-1);
}
print "Variable a=$a\n";
?>
```

Результат

./script3.1.php 4 2
Result: 2.000000
Variable a=my string
./script3.1.php 4 A
Error: the value A is not numeric!
./script3.1.php 4 0
Exception:
Illegal division by zero.

Standard PHP Library (SPL)

- Стандартна бібліотека PHP - це сукупність класів та інтерфейсів, які є рецептами для вирішення загальних проблем програмування. Вона доступна і скомпільована в PHP з версії 5.0.0.
- Класи діляться на категорії.
- Основні категорії: Data Structures, Iterators, Exceptions, File Handling, ArrayObject, SplObserver and SplSubject

Трейти (Traits)

- Трейти були введені в PHP 5.4.0 і призначені для полегшення деяких обмежень одиночного успадкування.
- Трейт містить сукупність методів і властивостей, подібно до класу, але не може створити екземпляр самостійно.
- Замість цього трейт входить до класу, і клас може потім використовувати його методи та властивості, як якщо б вони були оголошені в самому класі.
- Ми використовуємо ключове слово trait, щоб оголосити трейт; Щоб включити його в клас, ми використовуємо ключове слово use. Клас може використовувати кілька трейтів.

Оголошення та використання трейтів (1)

```
<?php
trait Singleton
{
    private static $instance;
    public static function getInstance() {
            if (!(self::$instance instanceof self)) {
                    self::$instance = new self;
            }
            return self::$instance;
    }
}
```

Оголошення та використання трейтів (2)

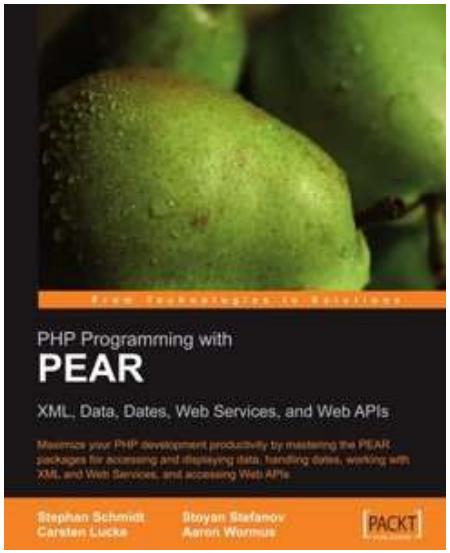
```
class UsingTraitExample
{
    use Singleton;
}
$object = UsingTraitExample::getInstance();
var_dump($object instanceof UsingTraitExample); // true
```

Бібліотека PEAR

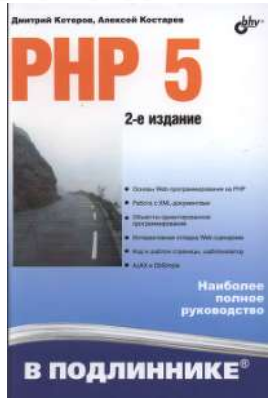
PEAR (англ. *PHP Extension and Application Repository* — укр. *Репозиторій Розширень і Додатків PHP*) — це бібліотека класів PHP з відкритим вихідним кодом, велике сховище програмного забезпечення яке включає багато складових частин і в цілому призначене для розширення області застосування і підвищення надійності мови PHP.

Приклади пакетів PEAR

- Auth. Аутентифікація користувача;
- Benchmark. Калібрування продуктивності;
- DB. Забезпечення зв'язку з базою даних;
- Calendar. Календарні об'єкти та функції;
- Log. Ведення журналів.;
- Mail. Взаємодія з засобами протоколів POP, IMAP і SMTP



Використання перенаправлень (redirect) в PHP



Причини: (Котеров Д. PHP 5. В подлиннике, 2008)

- Зміна доменного імені сайту
- Перенаправлення на сторінку реєстрації
- З http://www.site.com на http://site.com
- При додаванні в кінці посилання слеша (з http://.../news на http://.../news/)

Зовнішній редирект

Він представляє собою переадресацію, ініційовану браузером на прохання скрипта. Для цього браузеру треба надіслати тег <meta> і завершити роботу:

1-й спосіб.

```
<?php
  echo '<meta http-equiv="Refresh"
  content="0; URL=/some/other/script.html">';
  exit();
  ?>
```

через 0 секунд

Redirect в PHP

2-й спосіб. Надіслати спец. заголовок Location

```
Header("Location:
  http://{$_SERVER['SERVER_NAME']}/other/script.html");
exit();
```

Внутрішній редирект. (Він працює тільки в CGI-версії PHP)

Він обробляється на сервері, а не в браузері.

Для здійснення внутрішнього редиректу треба вказати у заголовку Location не абсолютний URL (з префіксом http:// та іменем хоста), а абсолютний URI (тобто URL без імені хоста).

З цього випливає, що внутрішній редирект може виконуватись в межах одного сайту.

Недолік внутрішнього редиректу:

Браузер про нього не знає, і може невірно відобразити сторінку

Redirect в PHP

Приклад. (Внутрішній редирект)

```
<?php
  Header("Status: 200 OK);
  // Отримуємо URI-директорію поточного скрипта
  $dir = dirname($_SERVER['SCRIPT_NAME']);
  Header("Location: $dir/result.php);
  exit();
?>
```

Самопереадресація (на ту ж сторінку) - стор. 961 (988)

Вона інколи потрібна через особливості протоколу HTTP (дивись Котерова).

Приклад – проста гостьова книга.

Код PHP та HTML-шаблон сторінки

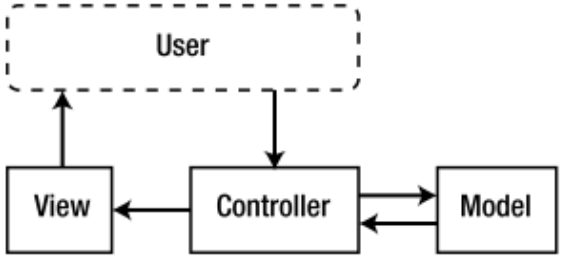
Є декілька підходів до розділення шаблону сторінки та коду сценарію. (Дивись Котеров Д. глава 46, стор. 977 (1004))

1-й варіант. "Вкраплення" HTML в код (це взагалі без відділення – самий гірший варіант)

2-й варіант. Вставка коду в шаблон

```
<html> <body><h2>Останні новини:</h2>
<? $f=fopen("../news.txt) ?>
<? for($i=...){ ?>
  <li><?=$i?>-та новина:<?=fgets($f,1024)?>
<?}?>
</body></html>
```

Це трохи краще.



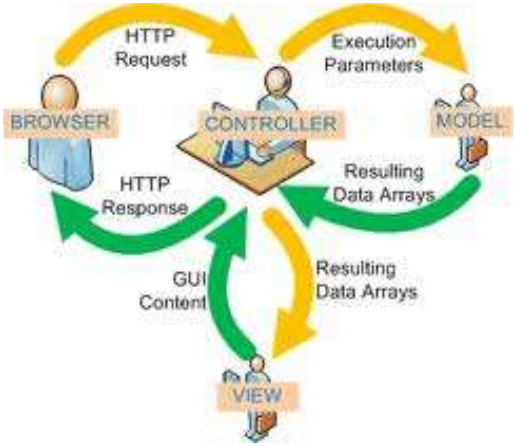
3-й варіант. Model-View-Controller

Модель. Представляє предметну область системи, її вміст. Включає БД системи та код, що з нею працює.

Представлення. Це шаблон, що визначає зовнішній вигляд.

Контролер. Код бізнес-логіки, приймає дані від користувача, пов'язує Модель та Шаблон.

Шаблон MVC



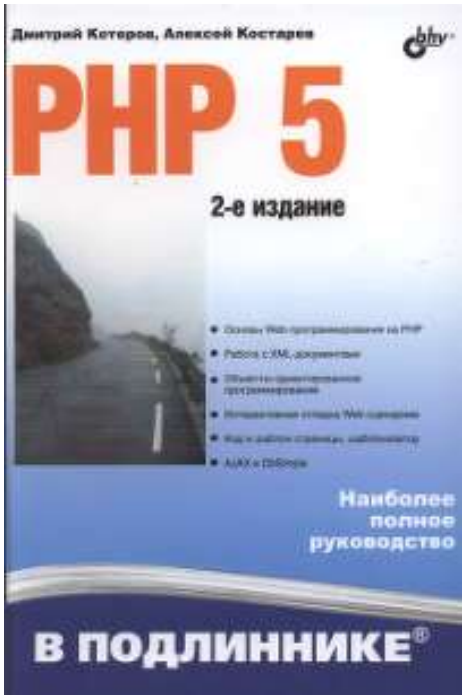
Приклад. "Гостьова книга" (в стилі MVC)

Котеров Д., стор.981(1008)

- Виділяється окремий каталог на сервері.
- View (шаблон сторінки) – view.html
- Controller – controller.php
- Model – model.php. Модель (ядро) гостьової книги. Містить функції LoadBook($fn) та SaveBook($fn,$Book)

Говорять, що заголовки (прототипи) ф-й ядра складають програмний інтерфейс (API) ядра, а самі тіла ф-й – реалізацію цього інтерфейсу. Модель при цьому визначає вміст системи.

Зауваження. Реалізацію можна замінити, наприклад зберігати гостьову книгу в БД.



Корисне посилання: Гостьова книга

http://ruseller.com/lessons.php?rub=37&id=1337

Popular MVC Frameworks

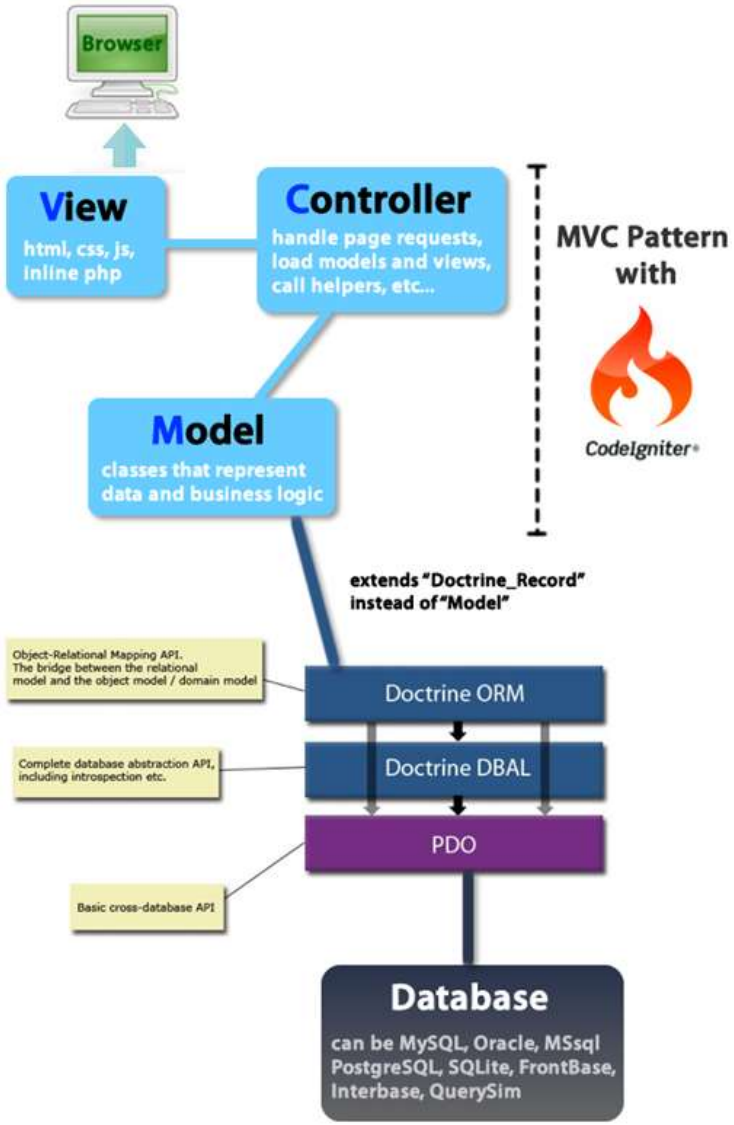
- CodeIgniter

- Zend Framework

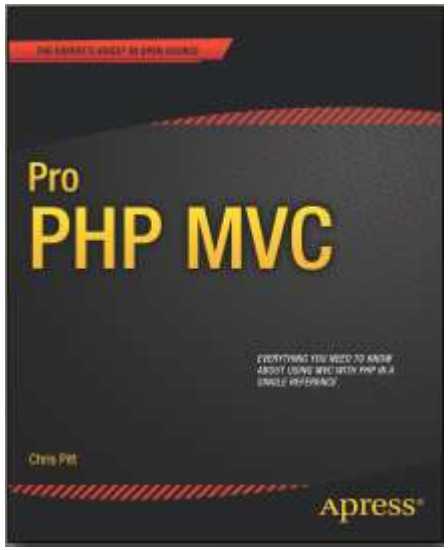
- CakePHP

- Kohana

- Yii



Creating Our Own MVC Framework



SQLQuery
(from base class)
_dbHandle
_result
connect(address, account, pwd, name)
disconnect()
seletAll()
select(id)
query(query, sigleResult = 0)

Template
(from base class)
variables
_controller
_action
set(name, value)
render()
__construst(controller, action)

#_template

Controller
(from base class)
_controller
_action
_model
__construct(mode, controller, action)
set(name, value)
__destruct()

Model
(from base class)
_model
_table
__construct()
__destruct()

Controller.__construct
$this->$model =& new $model;

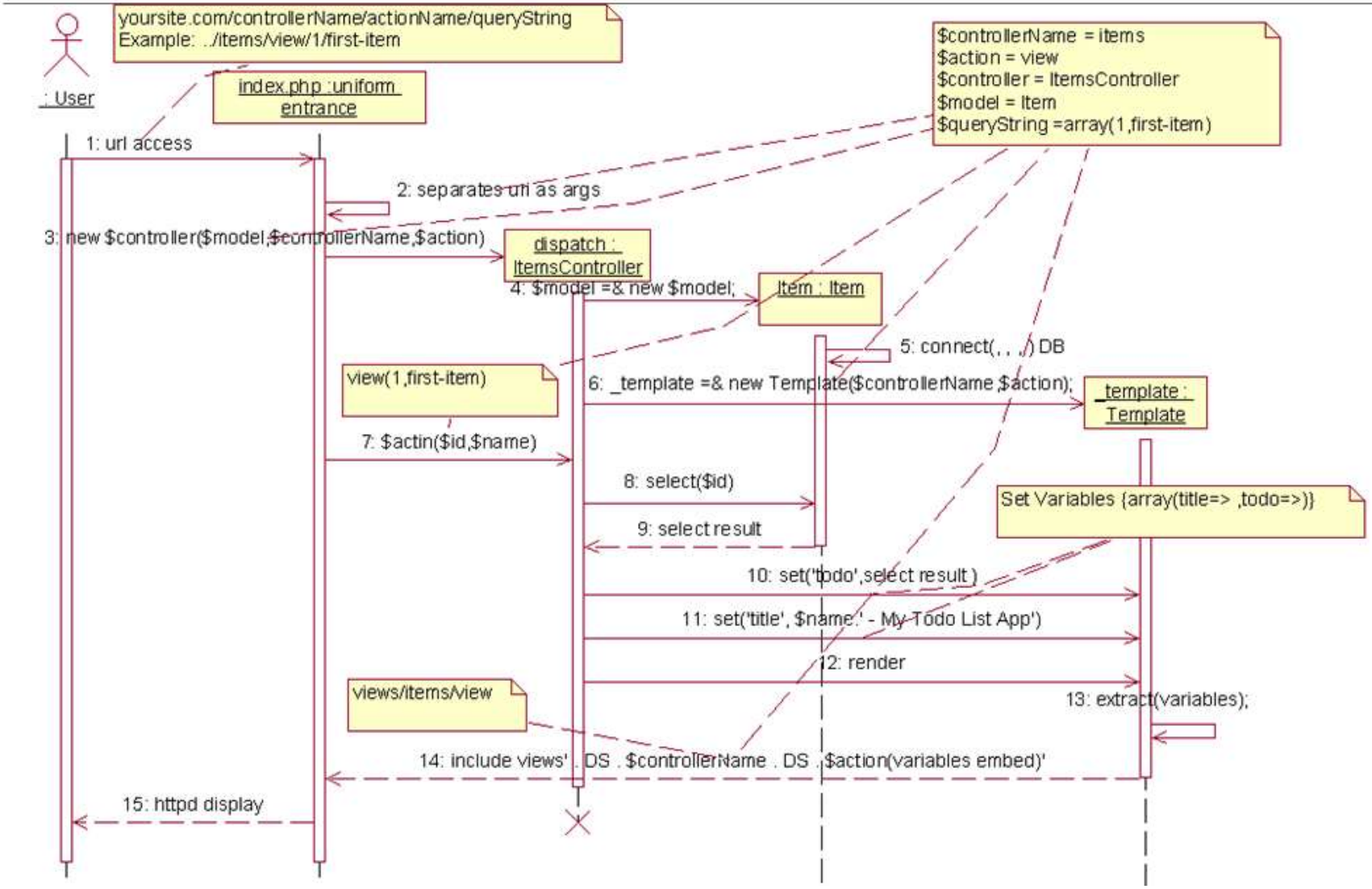
ItemsController
(from Items)
view(id, name)
viewall()
add()
delete(id)

operations:$action($queryString)

#$model(Item)

Item
(from Items)

Creating Our Own MVC Framework



Безпека створюваних web-застосувань

Семенов А.Н., Симдянов И.В.

Безопасное программирование на PHP

http://www.php.su/articles/?cat=security&page=012

Безпека створюваних web-застосувань

(Кузнецов М.В., Симдянов И.В. PHP. Практика создания Web-сайтов, 2009)

Безпека в середовищі Інтернет включає в себе багато різних аспектів, таких як

- **система захисту Web-сервера**;
- **безпеку баз даних**;
- **перевірка даних, що вводяться користувачами**;
- **прийоми і методи шифрування** та ін.

Перевірка обов'язковості введення поля: функції empty(), isset(), trim()

Перевірка числових значень: intval(), регулярні вирази для дійсних чисел “/^[\d]*[\.,]?[\d]*$/“

Перевірка правильності заповнення e-mail

Публікація зображень та файлів.

Треба заборонити завантаження на сервер довільних файлів, аналізуючи розширення файлів

.....

```
// Витягаємо з імені файла розширення
$ ext  =  strtolower ( strrchr ( $_FILES [ 'filename' ][ 'name ' ] ,"." ) ) ;

// Дозволяємо завантажувати файли тільки певного формату
$extentions  =  array ( ".jpg", ".gif" ) ;

// Перевіряємо, чи входить розширення в список
if( in_array ( $ ext ,   $extentions ) ) {  . . .
```

Можна також перевіряти тип файла через елемент суперглобального масиву $_FILES[‘filename’][‘type’]

SQL-ін'єкція

Впровадження (“внедрение”) SQL-коду (SQL injection) – один з найпоширеніших способів злому сайтів та програм, які працюють з БД.

Саму ідею впровадження розглянемо на прикладі.

Нехай ми маємо таблицю користувачів userslist, що має 5 полів: id_user, name, pass, email, url. Виводиться ім'я користувача та e-mail.

Ось фрагмент скрипта

```
$query = "SELECT * FROM userslist  WHERE  id_user =
    $_GET[id_user]";
```

Тут GET-параметр id_user підставляється в SQL-запит без всякої перевірки. Це дозволяє зловмиснику здійснити SQL injection.

SQL-ін'єкція (продовження)

Замість числа в SQL-запит можна впровадити довільний рядок. Наприклад, можна сформувати такий запит:

```
SELECT * FROM userslist WHERE id_user = 1
UNION
SELECT * FROM userslist WHERE id_user = 2
```

- Будуть знайдені записи як для першого, так і для другого запитів, але виведеться тільки для першого.
- Щоб вивести результати другого запиту замість 1 можна підставити -1. Тоді перший запит не дасть результатів.
- Інколи використовують конструкцію ORDER BY, яка сортує результати запитів
- Поки-що це нешкідливо.

SQL-ін'єкція (продовження)

SQL-ін‘єкції дозволяють витягувати довільні поля форми, у тому числі і пароль.

Для цього зловмисник розшифровує символ * в другому запиті

```
SELECT * FROM userslist WHERE id_user = 1
UNION
SELECT id_user, name, pass, email, url FROM userslist WHERE
   id_user = 2 ORDER BY id_user DESC
```

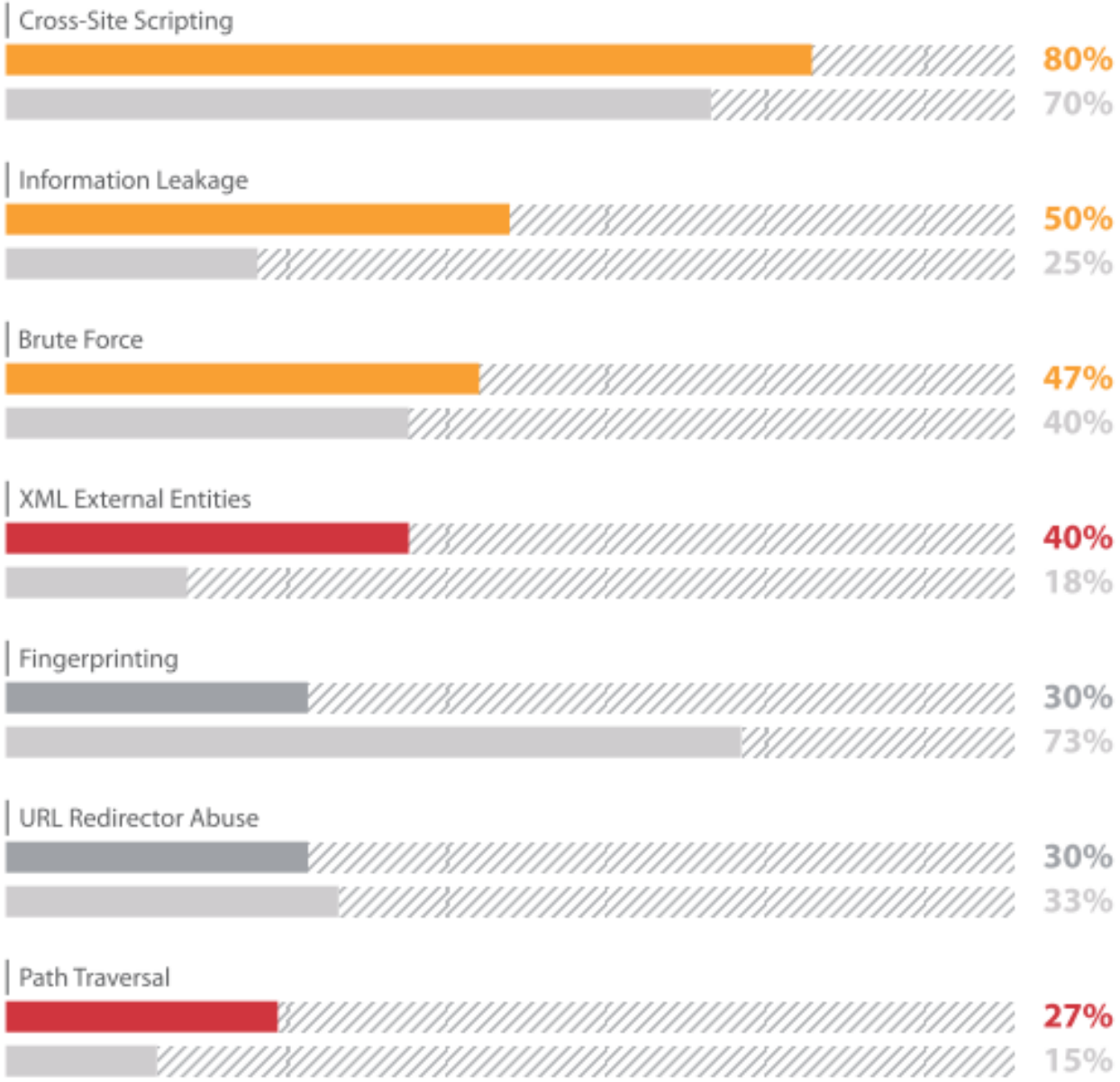
Стовпці name та pass текстові, тому їх можна поміняти місцями.

Імена стовпців формуються по першому запиту, тому буде виведено поле pass

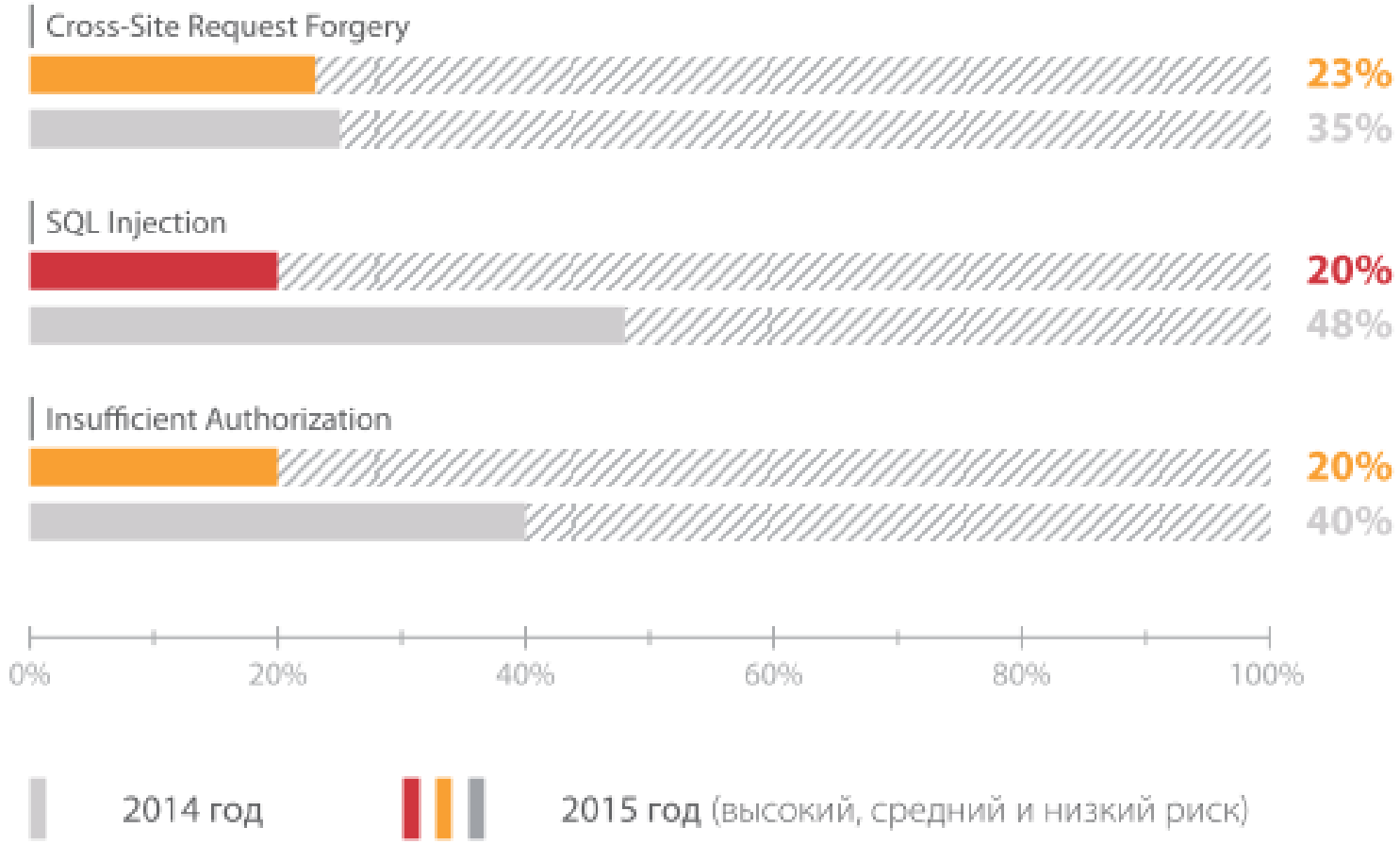
SQL-ін'єкція (закінчення)

- SQL-ін‘єкція може здійснюватися не тільки по числовому, а і по текстовому полю (дивись книгу)
- Щоб цьому запобігти треба екранувати лапки функцією mysql_real_escape_string()
- Слід перевіряти ВСІ змінні одержувані від користувача - GET, POST, COOKIE. У всі з них без зусиль можна вбудувати злобливий код.
- Особливу увагу слід приділяти паролям, тому що в них не прийнято вводити обмеження на символи.

Вразливості web-застосунків (1)



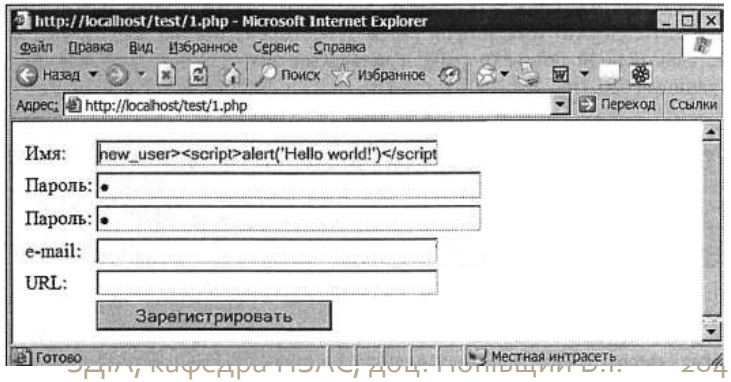
Вразливості web-застосунків (2)



- Уязвимости web-приложений https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/Web-Vulnerability-2016-rus.pdf

XSS-ін'єкція (Міжсайтовий скриптинг)

- XSS-ін'єкція – це атака, яка дозволяє зловмиснику вставляти в HTML-код сайту вставки шкідливого HTML-коду, використовуючи скрипти JavaScript.
- Специфіка подібних атак полягає в тому, що для атаки на сервер в якості засобу атаки використовується авторизований на цьому сервері клієнт
- XSS розшифровується як Cross-Site Scripting (CSS вже використано, тому XSS)
- Розглянемо фрагмент системи реєстрації



XSS-ін'єкція (продовження)

Ось вразливий рядок при виведенні списку

```
echo  "<a  href=$_SERVER [PHP_SELF]?name=$data [0] > ".
htmlspecialchars ( $data [0] )."</a><br> " ;
```

- Тут відсутнє екранування спецсимволів HTML (щоб браузер гарантовано не прийняв рядок за тег HTML, потрібно заекранувати п'ять символів: ‘ "& <>)
- Тепер якщо в формі ввести new_user><script>alert ('Hello world!')</script, то це дозволить виконати скрипт JavaScript (це поки-що нешкідливо)
- Можна перенаправляти користувачів на інший сайт new_user><script>location.href=.http://www.crackhost.ru';</script
- Можна викрадати cookie та паролі в них

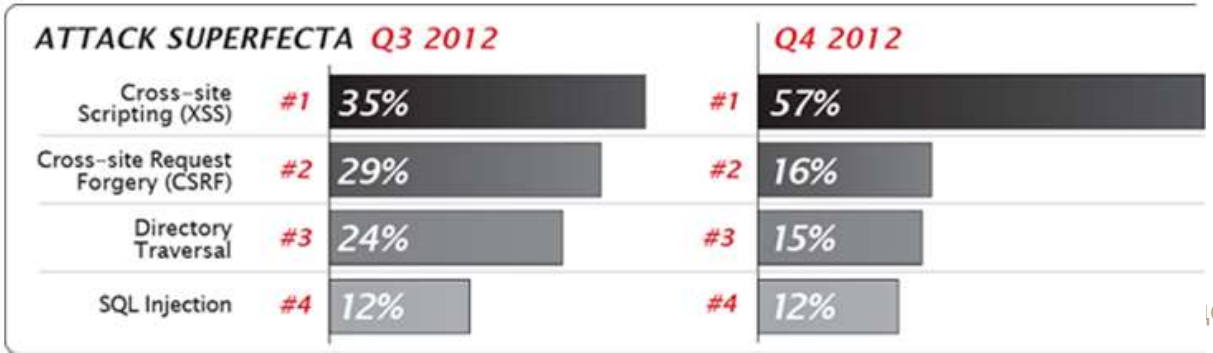
XSS-ін'єкція

- **Подводные камни в интернет рекламе или чем опасен XSS** *Статья для журнала Хакер Спец №75 (февраль 2007)*

http://websecurity.com.ua/articles/the_danger_of_xss/

- **Количество XSS-атак растет угрожающими темпами**

http://www.securitylab.ru/news/436914.php



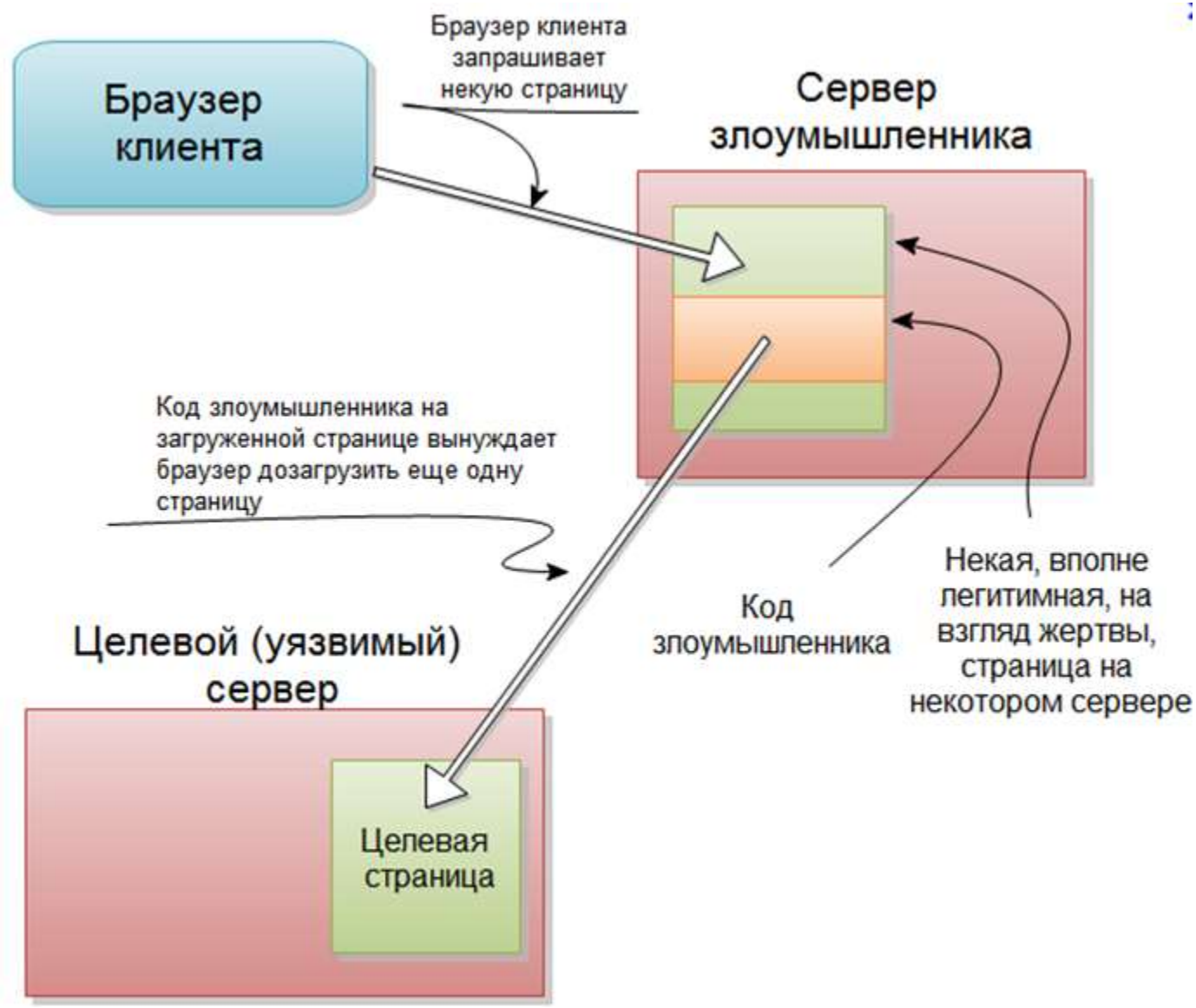
CSRF-атака. (Cross Site Request Forgery)

“Підробка міжсайтових запитів”

Це вид атаки на відвідувачів веб-сайтів, який використовує недоліки протоколу HTTP

- Якщо жертва заходить на сайт, створений зловмисником, від її особи таємно відправляється запит на інший сервер (наприклад, на сервер платіжної системи), що здійснює якусь шкідливу операцію (переказ грошей).
- Найбільш простий засіб захисту: веб-сайти повинні вимагати підтвердження дій користувача
- Більш поширений метод захисту – секретний ключ (до всіх важливих форм додається приховане поле)

CSRF-атака


Браузер клиента
Браузер клиента запрашивает некую страницу
Сервер злоумышленника
Код злоумышленника на загруженной странице вынуждает браузер дозагрузить еще одну страницу
Код злоумышленника
Некая, вполне легитимная, на взгляд жертвы, страница на некотором сервере
Целевой (уязвимый) сервер
Целевая страница

HTTP-injection

Хакер може дописати до URL JavaScript-код, який через змінну PHP_SELF або REQUEST_URI програміст сам вставить в свою сторінку

Ініціалізація змінних

Якщо включена директива PHP register_globals, користувач може створити в скрипті будь-яку змінну з будь-яким вмістом.

Захист: примусова ініціалізація змінних.

Безпека MySQL

- Не виконувати процес MySQL від імені root
- Створювати окремих користувачів для кожної БД

.HTACCESS (від англ. Hypertext access)

.HTACCESS – це файл додаткової конфігурації веб-сервера Apache (див. www.htaccess.net.ru)

- Дозволяє задавати велику кількість **додаткових параметрів і дозволів для роботи веб-сервера в окремих каталогах**
- **Файл .htaccess можна розмістити в довільному каталозі.** Директиви цього файлу діють на всі файли поточного та вкладених каталогів

Використання

- Авторизація, аутентифікація
- Зміна URL-адрес (довгих, складних на короткі)
- Заборона/Відкриття доступу для певних IP-адрес

MOD_REWRITE. Директиви складного перенаправлення

- **MOD_REWRITE – це модуль в складі Apache. Цей модуль дозволяє «переписувати» URL сторінки «на льоту».**
- За спеціальними правилами запит на URL з клієнтської програми буде розбитий на частини, проаналізований та перероблений в інший URL перед виконанням запиту.
- Зазвичай це використовують для конвертування динамічного URL з параметрами у статичний з іменем файла (створення "дружніх" адрес).
- Наприклад, запит до сайту новин:

www.новини.in.ua/search.php?день=12&місяць=квітень&рік=2006 перетвориться на:

www.новини.in.ua/search-12-april-2006.html

Як використовувати MOD_REWRITE

Створити директиви і розмістити їх в файлі .htaccess

- Директива RewriteEngine - вмикає / вимикає механізм mod_rewrite
- Директива RewriteRule - описує правило зміни адреси URL
- Директива RewriteCond - визначає умову, при якій відбувається перетворення.
- RewriteCond - визначає умову для якого-небудь правила.
- Перед директивою RewriteRule знаходиться нуль або декілька директив RewriteCond

Приклад. Перенаправляємо запити на сторінку dummy.html на сайт Google, використовуючи перенаправлення 301

```
RewriteEngine on
RewriteRule ^dummy\.html$ http://www.google.com/ [R=301]
```

Як працює RewriteRule

Узагальнений синтаксис директиви має вигляд:

```
RewriteRule Pattern Substitution [Optional Flags]
```

Pattern - регулярний вираз шаблону. Якщо URL відповідає шаблону, то правило виконується

Substitution - новий URL, який буде використовуватися замість адреси, що відповідає шаблону

[Optional Flags] - один або кілька прапорів, які визначають поведінку правила

Приклад. Запобігаємо використанню посилань на зображення з вашого сайту

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER}
   !^http://(www\.)?example\.com/.*$ [NC]
RewriteRule .+\.(gif|jpg|png)$ - [F]
```

Набір правил у файлі .htaccess говорить:

- якщо змінна HTTP_REFERER містить значення,
- і воно не починається на http://example.com/ або http://www.example.com/,
- і запитуваний URL містить ім'я файлу зображення, то треба відмовити запитом з помилкою "403 Forbidden“

(NC – чутливість до регістру символів)

MOD_REWRITE

Приклад. Направлення всіх запитів одному скрипту для обробки. Для цього потрібно додати у файл .htaccess наступний вміст:

```
RewriteEngine   on
RewriteCond   %{REQUEST_FILENAME}  !-f
RewriteCond   %{REQUEST_FILENAME}  !-d
RewriteRule   ^(.*)$  index.php  [L, QSA]
```

Скрипт index.php буде брати URL з $_SERVER['REQUEST_URI']

MOD_REWRITE

(З статтею “20+ правил .htaccess”)

- **Заборона завантаження файлів зображень з вашого сайту** (записати в кінці файлу .htaccess)

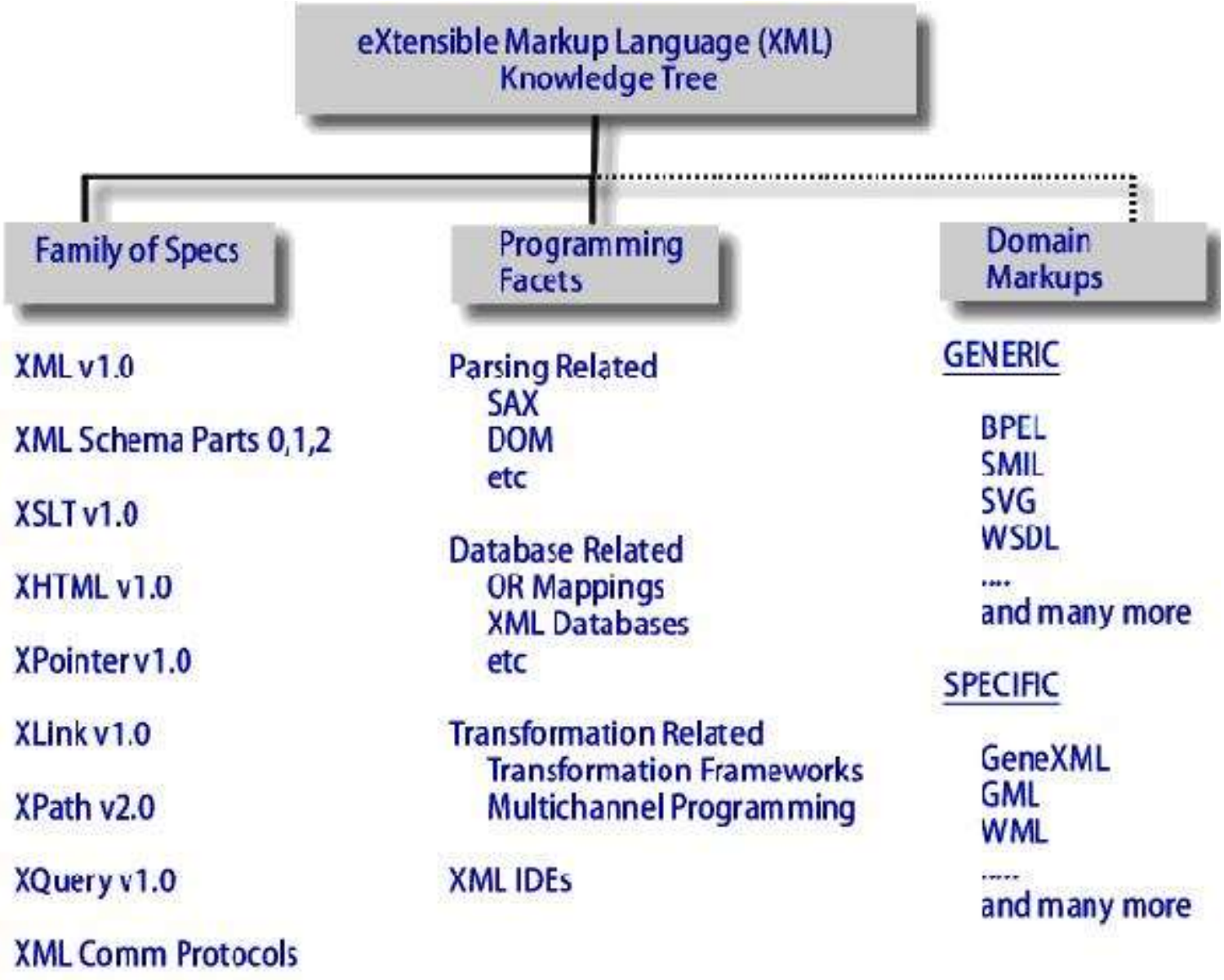
```
Options +FollowSymlinks
RewriteEngine On
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://(www.)?vash_site.com/ [nc]
RewriteRule .*.(gif|jpg|png)$ http://vash_site.com/img/stop_stealing.gif[nc]
```

Де stop_stealing.gif – зображення, яке буде з'являтися при спробі завантажити gif | jpg | png

XML та JSON

- Web-сервер відповідає на запити клієнта та відправляє запитані дані.
- Це може бути простий текст, але зазвичай використовують інший формат: XML або JSON
- Мова XML розроблялась для того, щоб забезпечити передачу даних між різними застосунками та навіть платформами.
- JSON надає можливість передавати дані в більш компактній формі, ніж XML.
- До того ж JSON легше обробляється JavaScript-сценаріями

XML

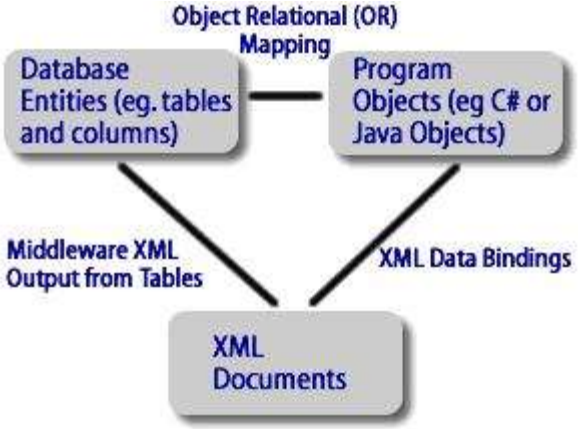

eXtensible Markup Language (XML)
Knowledge Tree
Family of Specs
Programming Facets
Domain Markups
XML v1.0
XML Schema Parts 0,1,2
XSLT v1.0
XHTML v1.0
XPointer v1.0
XLink v1.0
XPath v2.0
XQuery v1.0
XML Comm Protocols
Parsing Related
SAX
DOM
etc
Database Related
OR Mappings
XML Databases
etc
Transformation Related
Transformation Frameworks
Multichannel Programming
XML IDEs
GENERIC
BPEL
SMIL
SVG
WSDL
....
and many more
SPECIFIC
GeneXML
GML
WML
.....
and many more

XML

Domain Markup Languages:

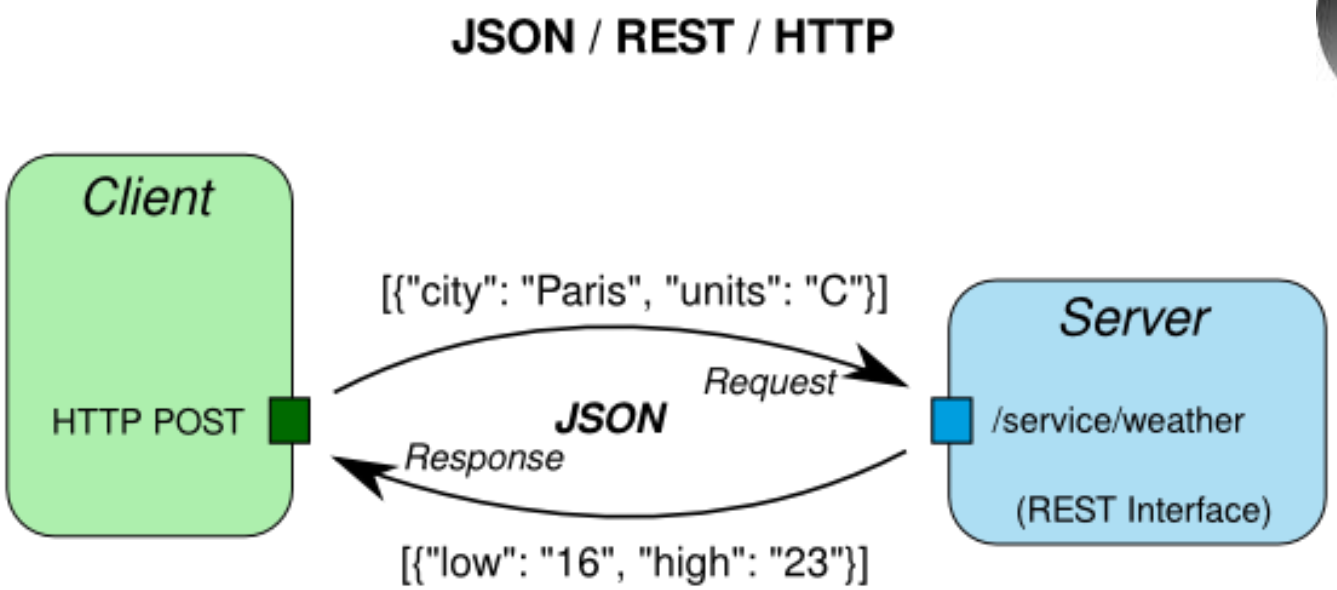
- Simple Vector Graphics (SVG)
- Web Services Definition Language (WSDL)
- Business Process Execution Language (BPEL)
- Wireless Markup Language (WML)

Database Related – relations in object oriented XML applications



XML та JSON

JSON
JavaScript Object Notation



XML та JSON

Мова XML (eXtensible Markup Language – розширювана мова розмітки)

- Схожа на HTML (теж є теги), але в XML теги не визначені наперед.

Синтаксис XML.

Програми, що читають XML-документи, називаються парсерами. Сучасні браузери мають такі парсери.

Правила:

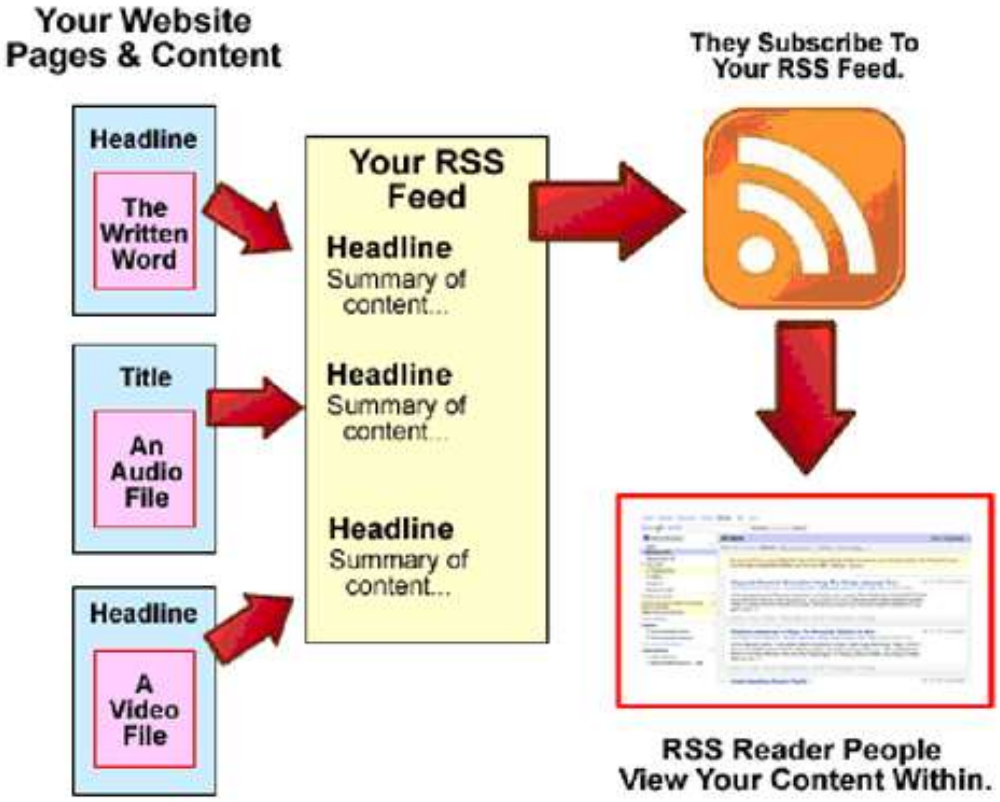
1. Всі теги повинні бути парними
2. Теги не повинні перекривати один одного
3. Тільки один кореневий елемент
4. Всі елементи можуть мати дочірні елементи
5. Регістр символів в назві тегів має значення
6. На початку документу треба вказати XML-декларацію <?xml version='1.0' encoding='UTF-8' ?>
7. Атрибути

Приклад

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
   <book>
        <author>Л.Аткинсон</author>
        <title>PHP5. Библиотека профессионала</title>
        <pubyear>2005</pubyear>
        <price>448</price>
   </book>
   <book>
        <author>Д.Коггзолл</author>
        <title>PHP5. Полное руководство</title>
        <pubyear>2006</pubyear>
        <price>336</price>
   </book>
```

RSS – новинні канали (feed)

RSS (Real Simple Syndication)— це сімейство XML-форматів, що використовується для публікації та постачання інформації, що часто змінюється, наприклад нових записів в блозі, заголовків новин, анонсів статей, зображень.



JSON

JavaScript Object Notation – “запис об'єктів в форматі JavaScript”

JSON оперує такими поняттями:

- Об'єкт – неупорядкований набір пар “ключ/значення”. Об'єкт починається з відкриваючої фігурної скобки ({) та закінчується “}”. Кожне ім'я супроводжується двокрапкою (:), пари “ключ/значення” розділені комами
- Масив – упорядкована колекція значень. Починається з ‘[‘, закінчується ‘]’. Значення розділені комами. Значення може бути рядком в лапках, числом, true, false, null, об'єктом або масивом. Ці структури можуть бути вкладеними.

JSON

- Рядок – колекція нуля або більше символів Unicode, укладена в лапки. Використовується зворотній слеш для екранування.
- Число представляється так, як в мові C або Java, крім того, що використовується тільки десяткова система числення.
- Пробіли можуть міститися між довільними лексемами

В PHP, починаючи з версії 5.2, існують функції, що перетворюють об'єкти PHP в об'єкти JSON і назад.

json_decode()

json_encode()

PHP 5 та XML

Засоби PHP 5 для роботи з XML:

- DOM (Document Object Model) – читання, модифікація та створення нових XML-документів
- SAX (Simple API for XML) – отримання інформації з XML-документу
- XSLT (Extensible Stylesheet Language Transformations) – перетворення XML-документів в інші формати
- SimpleXML – читання та модифікація XML-документу

Технологія DOM дозволяє працювати з деревом XML-документа.

Робота з DOM виконується в ОО-стилі. Для початку треба створити новий об'єкт DOMDocument.

PHP 5 та XML

Об'єкти для роботи з DOM

Об'єкт	Опис
DOMattr	Працює з атрибутами тегів
DOMCharacterData	Провадить операції із значеннями елементів
DOMDocument	Основний об'єкт. Відповідає за роботу з документом в цілому
DOMDocumentType	У кожного DOMDocument є свій тип документа, значенням якого є або null, або об'єкт DOMDocumentType
DomElement	Працює з елементами (тегами)
DOMEntity	Інтерфейс, що представляє відому сутність
DOMNode	Працює з елементами (вузлами)

PHP 5 та XML

Приклад.

```
<?php
$xml_data = “<plane><mark>IL-86</mark>
<owner>Aeroflot</owner></plane>;
$doc = domDocument::loadXML($xml_data);
$plane = $doc->getElementsByTagName(“plane”);
foreach($plane as $element){
  $cap = $doc->createElement(“capacity”,”360”);
  $element->appendChild($cap);
}
echo $doc->saveXML();
?>
```

SAX

Існує два основних стандартних методи створення XML-парсерів: SAX (Simple API for XML Parsing), і DOM (Document Object Model). Ці методи відрізняються підходом до створення парсера.

Метод SAX заснований на подієвій моделі. У цій моделі документ XML аналізується як поток даних.

У процесі аналізу виникають події:

- відкриття тега,
- вміст тега,
- закриття тега.

Створюючи власні обробники даних подій, користувач створює власний XML парсер для своєї схеми XML.

Бібліотека SimpleXML

Для роботи з багатьма XML-документами нема необхідності занурюватися в DOM.

В PHP 5 існує і більш простий, але доволі потужний інструмент – SimpleXML.

Основна ідея полягає в тому, що кожне XML-дерево можна конвертувати прямо в PHP-об'єкт.

Приклад. Файл tours.xml

```
<tours>
<tour id="1">
  <destination>Crimea</destination>
  <period>7</period>
  <price>600</price>
</tour>
<tour id="2">
<destination>Croatia</destination>
  <period>7</period>
  <price>800</price>
</tour>
</tours>
```

Бібліотека SimpleXML

Приклад (закінчення). Виведення на екран даних

```
<?php
$tours = simplexml_load_file("tours.xml);
foreach($tours->tour as $tour){
  echo "<p><b>Destination: ".
 $tour->destination."<br></b>";
  echo "Period (days): ".$tour->period."<br>";
  echo "Price (USD): ".$tour->price."</p>";
}
?>
```

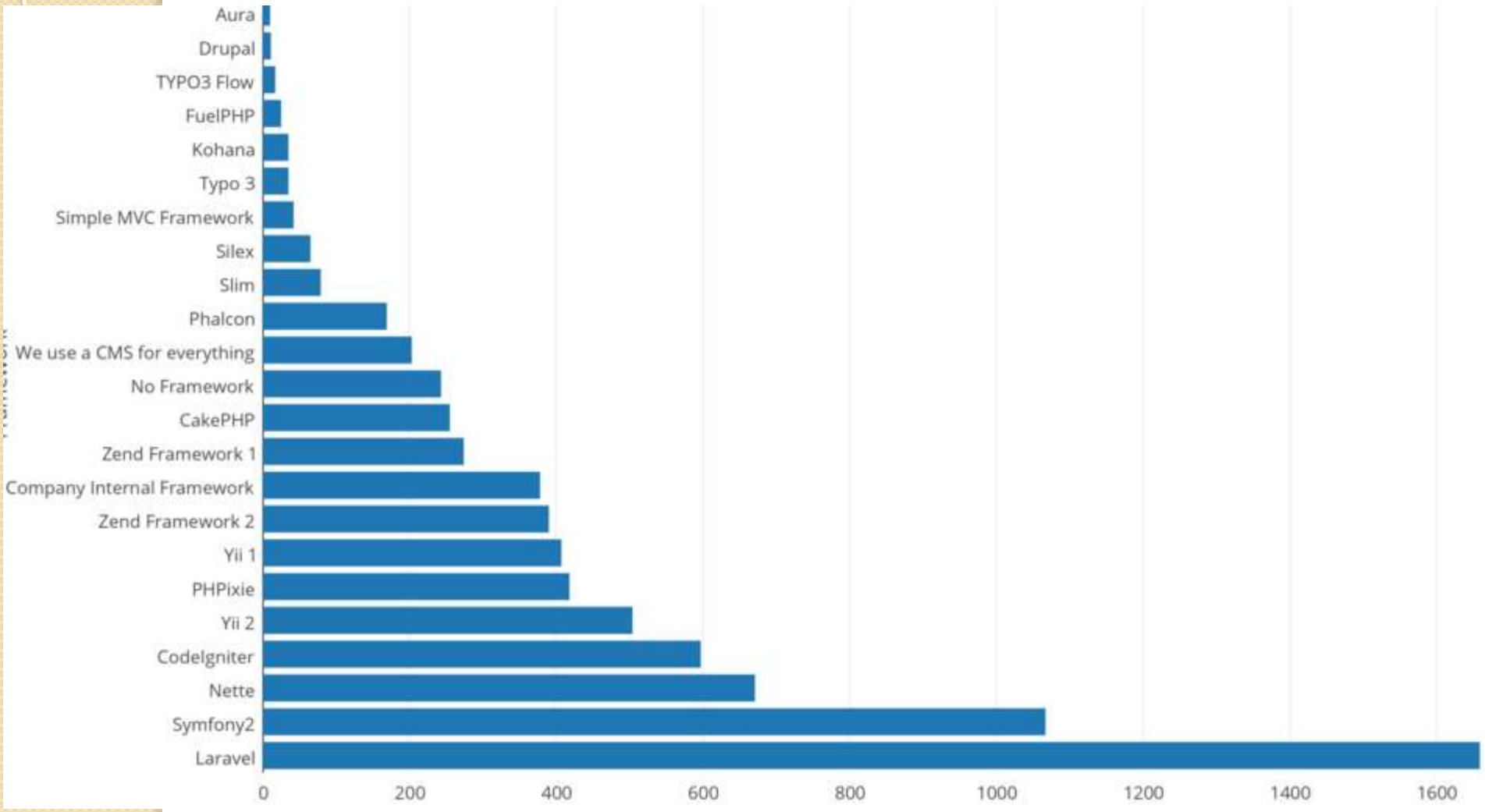
Огляд сучасних фреймворків

- Udemy - Learn Top Ten PHP FrameWorks By Building Projects (2016) - 4,23 Gb
- Udemy - Introduction to Modern Programming with PHP
- Lynda - MVC Frameworks for Building PHP Web Applications (2015)
- Modern PHP: New Features and Good Practices by Josh Lockhart, 2015
- Laravel Up and Running: A Framework for Building Modern PHP Apps, 2016
- Yii2 By Example by Fabrizio Caldarelli, 2015
- Сафронов М. Разработка веб-приложений в Yii 2, 2015

Lynda - MVC Frameworks for Building PHP Web Applications (2015)

- 01. Introduction
- 02. About PHP Frameworks
- 03. Zend
- 04. Symfony
- 05. CodeIgniter
- 06. CakePHP
- 07. Yii
- 08. Laravel
- 09. Conclusion

Популярність PHP фреймворків в 2015 році (https://habrahabr.ru/post/254277/)



Корисні посилання

- Symfony2 vs Yii2: какой фреймворк выбрать в 2016-м году? http://stfalcon.com/blog/post/symfony2-vs-yii
- LARAVEL VS SYMFONY — 5 ШАГОВ К ВЫБОРУ PHP ФРЕЙМВОРКА HTTP://DEVSIZE.RU/LARAVEL-VS-SYMFONY-5-STEPS-CHOOSING-FRAMEWORK
- Выбор фреймворка для PHP новичка 2015 год (codeigniter VS yii) http://weblone.ru/materials/24
- Что такого прекрасного в Laravel? https://laravel.ru/posts/177
- **Найкращі 10 PHP Фреймворків для розробників** http://code.in.ua/article/38-naykraschi-10-php-freymvorkiv-dlya-rozrobnikiv

Що надають фреймворки?

- Каркас застосунку на основі ООП та шаблону MVC
- Реалізують нові стандарти PSR
- Використовують Composer
- Вбудовані інструменти тестування
- Можливість швидкої розробки
- Забезпечують добре організований, зрозумілий код, який легко модифікувати
- Високу безпеку вашого сайту
- Сучасні веб-розробницькі практики

Стандарти PSR

PSR-0 - Autoloader Standard
PSR-1 - Basic Coding Standard
PSR-2 - Coding Style Guide
PSR-3 - Logger Interface
PSR-4 - Autoloader Standard

PHP Standards Recommendations

Стандарт PSR-1. Основний стандарт кодування

- Використовувати тільки теги <?php і <?=
- Тільки UTF-8 без BOM
- Класам НЕОБХІДНО давати імена в стилі **StudlyCaps**
- Константам класів НЕОБХІДНО давати імена у верхньому регістрі з символом підкреслення як роздільник: **MAIN_PRODUCT**
- Методам НЕОБХІДНО давати імена в стилі **camelCase**

Стандарти PSR

PSR-2. Стиль кодування

- Для оформлення відступів ПОВИННІ використовуватися чотири пробіли (але не знак табуляції)
- Довжина рядка < 120
- Після визначення простору імен (**namespace**) і після блоку імпорту просторів імен (**use**) ПОВИНЕН бути один порожній рядок
- Відкриваюча фігурна дужка у визначенні класу ПОВИННА розташовуватися на новому рядку
- Видимість НЕОБХІДНО оголошувати для всіх властивостей і методів (public, private …)
- В кінці файлу не закривати PHP **?>**
- . . . . . .

Composer

- Dependency Manager for PHP
- https://getcomposer.org/
- Аналог npm для Node.js NuGet для Visual Studio Bundler для Ruby
- З'явився 1 березня 2012 р.
- Composer працює через інтерфейс командного рядка і встановлює залежності (наприклад бібліотеки) для додатків
- Має офіційний репозиторій пакетів https://packagist.org/

Установка Composer під Windows та XAMPP

- Скачуємо сам Composer
  - https://getcomposer.org/Composer-Setup.exe
- Інсталлюємо
  - Під час інсталяції треба буде вказати шлях до php.exe (в xampp це як правило: C:\xampp\php\php.exe)
  - Потрібно включити php_openssl.dll (розкоментувати extension = php_openssl.dll). Йдемо в C:\xampp\php\php.ini і включаємо, що потрібно

Веб-служби

Веб-служба, *веб-сервіс* (англ. *web service*) — програмна система, що ідентифікується рядком URI, та публічні інтерфейси та прив'язки якої визначені та описані мовою XML.

Опис цієї програмної системи може бути знайдено іншими програмними системами, які можуть взаємодіяти з нею відповідно до цього опису з використанням повідомлень, що базуються на XML та передаються за допомогою інтернет-протоколів.

Стандарти, що використовуються веб-службами

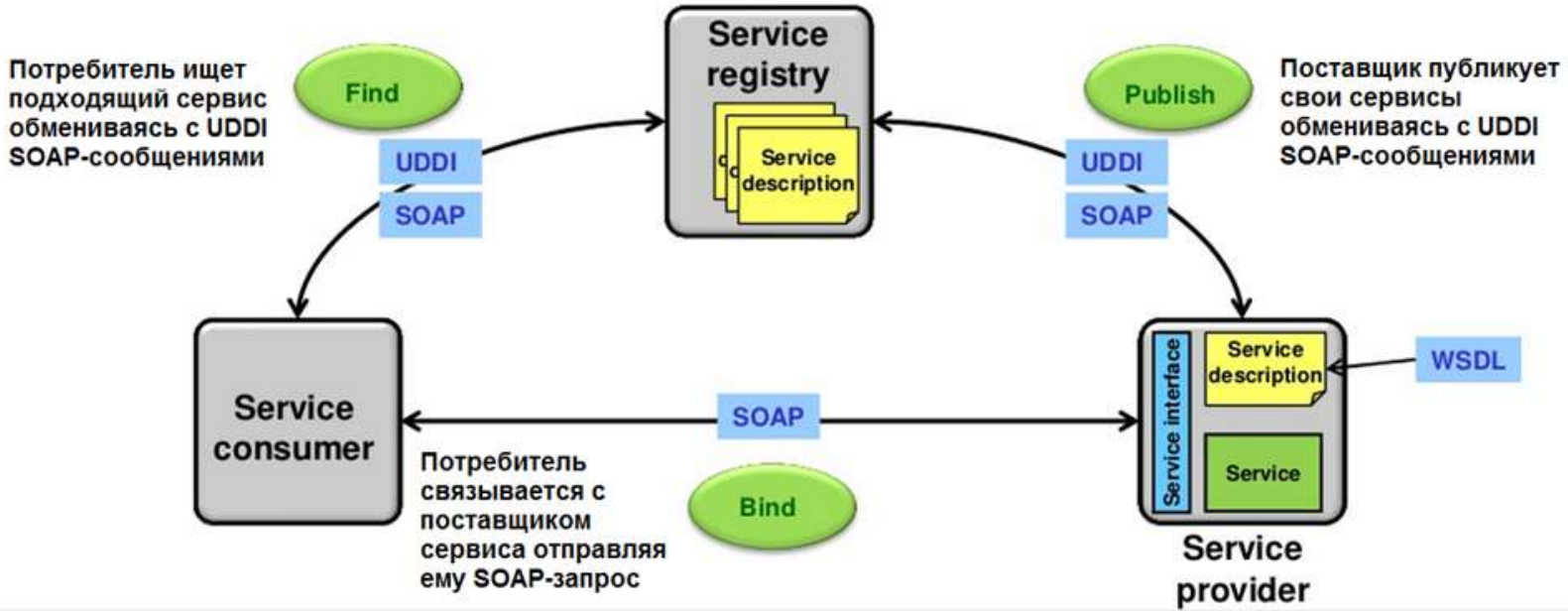
- XML: Розширювана мова розмітки, призначена для зберігання і передачі структурованих даних;
- SOAP: Протокол обміну повідомленнями на базі XML;
- WSDL: Мова опису зовнішніх інтерфейсів веб-служби на базі XML;
- UDDI: Універсальний інтерфейс розпізнавання опису та інтеграції (Universal Discovery, Description, and Integration).

Архітектура web-сервісу

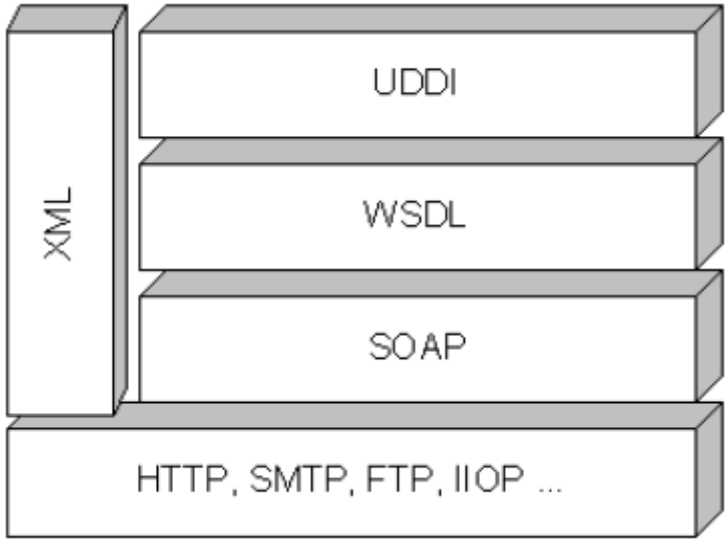
Архитектура вебсервиса

Связка SOAP+WSDL+UDDI определяет главную модель архитектуры Вебсервиса

SOAP:	Simple Object Access Protocol
WSDL:	Web Service Description Language
UDDI:	Universal Description and Discovery Protocol
Service consumer:	Потребитель сервиса
Service provider:	Сущность, которая реализует сервис (=сервер)
Service registry:	Центральное место, где перечислены доступные сервисы



Взаємозв'язок технологій



Публікація та пошук сервісів

Опис інтерфейсів сервісів

Обмін повідомленнями

Транспортна інфраструктура

Веб-сервіси

Веб-служби

Переваги веб-служб

- Веб-служби забезпечують взаємодію програмних систем незалежно від платформи
- Веб-служби базуються на відкритих стандартах та протоколах. Завдяки використанню XML досягається простота розробки та відлагодження веб-служб
- Використання інтернет-протокола HTTP забезпечує взаємодію програмних систем через міжмережевий екран

Web-сервіс – це програмне забезпечення, що надає платформено-незалежний доступ до своїх даних іншим програмним продуктам через Інтернет, з використанням XML та таких стандартів, як SOAP, WSDL та UDDI

Структура WSDL

WSDL-стандарт (***W**eb **S**ervices **D**escription **L**anguage*) має дві частини: інтерфейс сервісу та реалізацію сервісу.

Інтерфейс сервісу - це абстрактне визначення сервісу, на яке можуть посилатися різні реалізації сервісу. Реалізація сервісу описує, як конкретний сервіс реалізується даним постачальником.

<type> визначають типи XML-даних сервісу
<message> описує кожне SOAP-повідомлення (запит, відповідь)
<portType> визначає набір операцій (методів) сервісу
<binding> описує конкретний формат пересилання (SOAP, HTTP), MIME-тип повідомлення
<port> пов'язує кінцеву точку (мережеву адресу або URL) з елементом binding в інтерфейсі сервісу.
<service> місцезнаходження сервісу (містить набір елементів port)

Приклад WSDL-файлу

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Guid"
targetNamespace="http://www.hauser-wenz.de/Guid/"
xmlns:tns="http://www.hauser-wenz.de/Guid/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlnsгwsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="getGuidResponse">
      <part name="Result" type="xsd:string" />
   </message>
   <message name="getGuidRequest">
      <part name="prefix" type="xsd:string" />
   </message>
<portType name="GuidPortType">
      <operation name="getGuid" parameterOrder="prefix">
         <input message="tns:getGuidRequest"/>
         <output message="tns:getGuidResponse" />
      </operation>
</portType>
```

Приклад WSDL-файлу (закінчення)

```
<binding name="GuidEinding" type="tns:GuidPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getGuid">
      <soap:operation soapAction="urn:php5unleashed-guid#getGuid"/>
      <input>
        <soap:body use="encoded" namespace="urn:php5unleashed-guid"
          encodingstyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output>
        <soap:body use="encoded" namespace="urn:php5unleashed-guid"
          encodingstyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
</binding>
<service name="GuidService"
    <port name="GuidPort" binding="tns:GuidBinding">
      <soap:address location="http://localhost/php/guid-server.php"/>
    </port>
</service>
</definitions>
```

WSDL

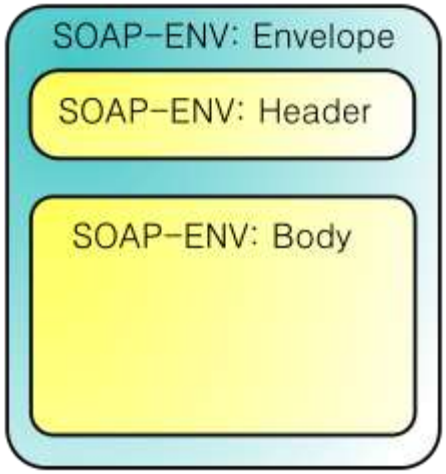
- Один з недоліків SOAP-розширення для PHP 5 пов'язаний з тим, що на відміну від інших реалізацій SOAP, воно не дозволяє створювати WSDL-опису автоматично.
- Автоматична генерація WSDL-опису є в PEAR реалізації SOAP

Протокол SOAP

- Simple Object Access Protocol
- SOAP визначає структуру повідомлень між постачальником веб-сервісу і споживачем

Повідомлення SOAP структурується так:

SOAP- конверт
SOAP-заголовок
Елемент заголовку 1
Елемент заголовку 2
…
Елемент заголовку N
Тіло SOAP
Елемент тіла 1
Елемент тіла 2
…
Елемент тіла N



Протокол SOAP

Приклад. Повідомлення з запитом GetLastTradePrice, який дозволяє клієнту послати запит про останні котируваннях певних акцій

POST/StockQuote HTTP/1.1
Host: *www.stockquoteserver.com*
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<**SOAP-ENV:Envelope** xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<**SOAP-ENV:Body**>
<m:GetLastTradePrice xmlns:m="Some-URI">
<symbol>DIS</symbol>
</m:GetLastTradePrice>
</**SOAP-ENV:Body**>
</**SOAP-ENV:Envelope**>

DIS – це параметр запиту (символ акції)
Заголовка тут нема.

Протокол SOAP

Приклад (закінчення). Відповідь на цей запит може виглядати наступним чином.

HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<**SOAP-ENV:Envelope** xmlns:
SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
<**SOAP-ENV:Body**>
<m:GetLastTradePriceResponse xmlns:m="Some-URI">
<Price>34,5</Price>
</m:GetLastTradePriceResponse>
</**SOAP-ENV:Body**>
</**SOAP-ENV:Envelope**>

UDDI

UDDI являє собою стандарт універсального опису, виявлення та інтеграції (Universal Description, Discovery and Integration) веб-сервісів і є важливою і невід'ємною частиною сервіс-орієнтованої архітектури (SOA)

UDDI – інструмент для розташування описів веб-сервісів (WSDL) для подальшого їх пошуку іншими організаціями та інтеграції в свої системи

Стандарт UDDI описує чотири елементи даних:

- businessEntity – Підприємство
- businessService – Послуга
- tModel – Тип послуги
- bindingTemplate – Зв'язуючий шаблон

UDDI - це "жовта книга" мережі, яка використовується, як провайдерами сервісів, так і споживачами.