Міністерство освіти і науки України
Запорізька державна інженерна академія

В.І. Попівщий

ПРОГРАМУВАННЯ ІНТЕРНЕТ

Навчально-методичний посібник

для студентів ЗДІА
спеціальності 121 "Інженерія програмного забезпечення"

Рекомендовано до видання
на засіданні кафедри ПЗАС
протокол № 16 від 03.04.2018 р.

Запоріжжя
2018

УДК 621.396.218
ББК 004.4
П 575

Укладач: ***В. І. Попівщий, доцент***

Відповідальний за випуск: ***зам.зав.кафедри ПЗАС***
доцент А.І.Безверхий

Рецензенти:

М.Ю. Пазюк – д.т.н., професор, завідувач кафедри автоматизації технологічних процесів Запорізької державної інженерної академії

Грищак В.З. – д.т.н., професор, завідувач кафедри прикладної математики Запорізького національного університету

Попівщий В.І.
П 575 Програмування Інтернет. Навчально-методичний посібник для студентів ЗДІА спеціальності 121 "Інженерія програмного забезпечення"- Запоріжжя, 2018. – 185 с.

Навчально-методичний посібник містить конспект лекцій з курсу, завдання до лабораторних робіт, екзаменаційні питання, список рекомендованої літератури.

ЗМІСТ

ПЕРЕДМОВА

Метою викладання дисципліни «Програмування Інтернет» є отримання студентами теоретичних знань і практичних навичок з програмування для мережі Інтернет на боці сервера, використання мов PHP, JavaScript, Python та сервера баз даних MySQL для розробки web-застосунків та web-інтерфейсів для баз даних.

Завданням навчальної дисципліни є огляд технологій Web-програмування, вивчення основних понять з розробки сайтів, детальне вивчення мови програмування PHP, технології Node.js, фреймворків Laravel, Express, Python Django, СКБД MySQL, набуття практичних навичок по створенню сайтів на основі використання сучасних мов програмування.

У результаті вивчення навчальної дисципліни студент повинен

знати:

- основи функціонування World Wide Web;
- основні технології створення Web-сайтів;
- протоколи обміну інформацією Web-серверів і клієнтських браузерів;
- основи мов програмування PHP, JavaScript, Python;
- основи конфігурування web-сервера Apache для роботи з PHP;
- основи об'єктно-орієнтованого програмування на PHP.

вміти:

- проводити інсталяцію та налаштування комплексу програм Apache + PHP + MySQL для ОС Windows;
- писати серверні додатки на мовах PHP, JavaScript, Python;
- проводити відладку програм;
- використовувати cookie та сесії;
- працювати з файлами та електронною поштою;
- використовувати бази даних MySQL, SQLite;
- застосовувати отримані знання для розробки динамічних web-сайтів.

Предметом даного курсу є технології розробки серверних веб-додатків на основі мов програмування PHP, JavaScript та Python. Розглядатимуться також деякі «супутні» технології, такі як Bootstrap, Angular та інші. Передбачається, що в попередніх курсах слухачі вже опанували HTML, CSS, JavaScript.

В наш час відбувається стрімкий розвиток веб-технологій як на боці клієнта, так і на боці сервера. Можна відзначити наступні сучасні тенденції в розвитку сайтів:

- мінімалістичний дизайн;
- застосування сучасних технологій HTML5 і CSS3;
- відмова від використання Flash на користь HTML5 і JavaScript;
- адаптивний дизайн;
- односторінкові сайти.

Ці тенденції треба враховувати при розробці веб-орієнтованого програмного забезпечення.

При написанні клієнтських сценаріїв традиційно використовується JavaScript та бібліотеки jQuery, Ember.js, Angular.js, Backbone.js, React.js та інші.

Розробка серверних сценаріїв включає в себе використання технологій програмування PHP, ASP.NET, JSP, Node.js, Python Django, Ruby on Rails для створювання скриптів, які генерують контент динамічно [1-35].

Важливо пам'ятати, що при розробці серверних сценаріїв, ви пишете програмне забезпечення, так само, як це робить C або Java програміст, з тією різницею, що ваше програмне забезпечення працює на веб-сервері і використовує цикл запиту-відповіді HTTP для інтерактивної взаємодії з клієнтами. Ця відмінність є істотною, оскільки позбавляє чинності багато класичних моделей програмного забезпечення, і вимагає іншого підходу для багатьох, здавалося б, простих принципів програмного забезпечення, таких як зберігання даних і управління пам'яттю.

Фундаментальна відмінність між клієнтськими і серверними скриптами є те, що в клієнтському сценарії код виконується в браузері клієнта, в той час як серверний скрипт виконується на веб-сервері.

Серверний код залишається прихованим від клієнта через те, що він обробляється на сервері. Клієнти ніколи не можуть побачити цей код, вони бачать тільки HTML вихід зі сценарію.

Розташування сценарію також впливає на те, до яких ресурсів він може отримати доступ. Серверні скрипти не можуть маніпулювати HTML або DOM сторінкою в браузері клієнта, як це можливо з клієнтських скриптів. І навпаки, серверний сценарій може отримати доступ до ресурсів веб-сервера, тоді як клієнт це не може. Розуміння того, де скрипти знаходяться і до чого вони можуть отримати доступ має важливе значення для якості написання веб-додатків.

Серверний скрипт може отримати доступ до будь-яких ресурсів, представлених йому сервером. Ці ресурси можна поділити на три категорії: ресурси для зберігання даних, веб-сервіси і додатки.

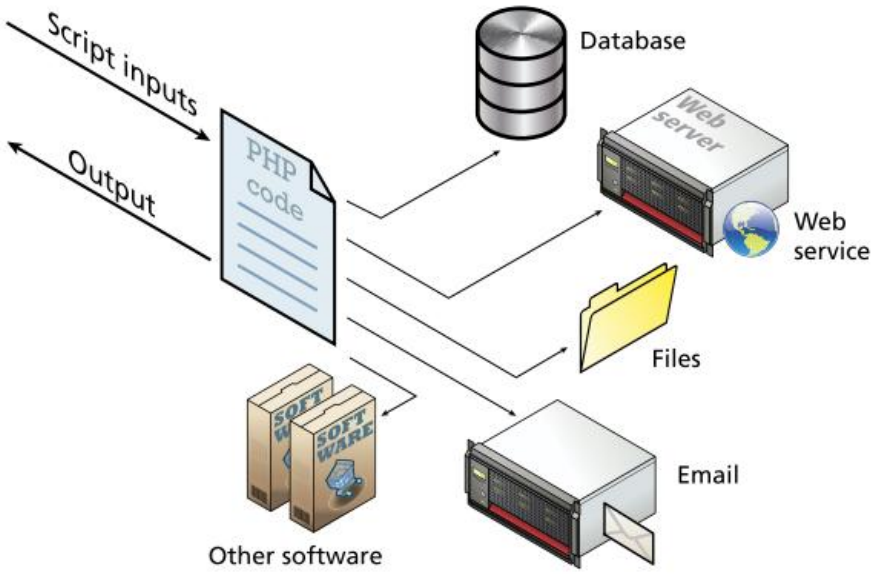


Рис. 1. Серверні скрипти мають доступ до багатьох ресурсів

Найбільш часто використовуваним ресурсом є **ресурс зберігання даних**, часто у вигляді з'єднання з системою управління базою даних. У той час як майже

кожен значний реальний сайт використовує певний тип бази даних, багато веб-сайтів також використовує файлову систему сервера; наприклад, в якості місця для зберігання користувальницьких завантажень.

Наступним видом ресурсів є **веб-сервіси**, часто пропоновані сторонніми постачальниками. Веб-сервіси використовують HTTP протокол щоб повернути дані в форматі XML або інші дані і часто використовуються, щоб розширити функціональність веб-сайту. Прикладом є служба геолокації, яка повертає назву міста і країни у відповідь на географічні координати.

Нарешті, ресурсом може бути **додаткове програмне забезпечення**, яке встановлене на сервері або до якого є доступ через мережеве з'єднання. Завдяки цьому, серверні додатки можуть відправляти і отримувати електронну пошту, мати доступ до сервісів аутентифікації користувачів і використовувати мережеві сховища даних. Ви можете підключити веб-додаток до звичайної телефонної мережі і посилати тексти або здійснювати дзвінки.

Є кілька різних серверних технологій для створення веб-додатків. Найбільш поширеними є:

- **ASP (Active Server Pages).** Це була перша серверна технологія Microsoft (також відома під назвою ASP Classic). Як PHP, ASP код (з використанням мови програмування VBScript) вбудовується в HTML; хоча він підтримує класи і деякі об'єктно-орієнтовані можливості, більшість розробників не використовували ці функції. Програмний код ASP інтерпретується під час виконання, отже, він може бути повільним у порівнянні з іншими технологіями.
- **ASP.NET**. Він замінив стару технологію ASP. ASP.NET є частиною Microsoft .NET Framework і може використовувати будь-яку мову програмування .NET (хоча найбільш часто використовуваною є мова C#). ASP.NET використовує явно об'єктно-орієнтований підхід, який зазвичай займає більше часу для вивчення ніж ASP або PHP, і часто використовується в великих корпоративних системах веб-додатків. Він також використовує спеціальні розмітки, так звані серверні елементи управління, які інкапсулюють загальні веб-функції, такі як списки, пов’язані з базою даних, перевірки форм, і реєстрації користувачів. Недавнє розширення цієї технології називається ASP.NET MVC і використовує шаблон проектування Model-View-Controller. Сторінки ASP.NET компілюються в проміжний формат файлу під назвою MSIL, що аналогічний байт-коду Java. ASP.NET потім використовує JIT (Just-In-Time) компілятор для компіляції MSIL в машинний виконуваний код таким чином його продуктивність може бути великою. Тим не менш, ASP.NET істотно обмежені серверів Windows.
- **JSP (Java Server Pages).** JSP використовує Java в якості мови програмування і, як ASP.NET, використовує явний об'єктно-орієнтований підхід і використовується у великих веб-системах підприємства, інтегрованих в середовище J2EE. Так як JSP використовує Java Runtime Engine, він також використовує JIT компілятор для швидкого часу виконання і є крос-платформеним. У той час як доля використання JSP в Інтернеті в цілому невелика, JSP має

значну частку ринку в інтрамережі, а також серед дуже великих і навантажених сайтів.

- **Node.js.** Це більш недавнє серверне середовище, яке використовує JavaScript на стороні сервера, що дозволяє розробникам вже знайомим з JavaScript використовувати тільки одну мову і для клієнтської сторони і для розробки на стороні сервера. На відміну від інших технологій розробки перерахованих тут, Node.js має також його власне програмне забезпечення веб-сервера, тим самим усуваючи необхідність використання Apache, IIS, або іншого програмного забезпечення веб-сервера [31-32].
- **Perl**. До появи та популяризації ASP, PHP, і JSP, Perl був основною мовою, яка використовувалася для розробки на стороні веб-сервера. Як мова, він виділяється широкими можливостями в маніпулюванні текстом. Це широко використовується в поєднанні із Common Gateway Interface (CGI), раннім стандартом API для зв'язку між додатками і програмним забезпеченням веб-сервера.
- **PHP**. Як і ASP, PHP є динамічно типізованою мовою, яка може бути вбудована безпосередньо в HTML, хоча тепер PHP підтримує найбільш поширені об'єктно-орієнтовані функції, такі як класи й спадкування. За замовчуванням, PHP сторінки компілюються в проміжне представлення, яке називають **opcodes**, і яке аналогічне байт-коду Java або MSIL для .NET Framework. Спочатку, PHP означав Personal Home Pages, хоча в даний час PHP трактується як рекурсивний акронім, що означає PHP: Hypertext Processor [1-29].
- **Python**. Це лаконічна, об'єктно-орієнтована мова програмування, що має безліч застосувань, у тому числі використовується для створення веб-додатків. Python також використовується в різноманітних фреймворках веб-розробки, таких як Django і Pyramid [33-35].
- **Ruby on Rails**. Це середовище розробки веб, яке використовує мову програмування Ruby. Як ASP.NET і JSP, Ruby on Rails використовує поширені програмні підходи до розробки, зокрема, шаблон MVC. Він об'єднує такі функції, як шаблони і движки, які спрямовані на зменшення кількості роботи, необхідної для створення нового сайту.

Всі ці технології об'єднує одне спільне: за допомогою програмної логіки вони генерують HTML і CSS, можливо і JavaScript на сервері і відправляють його назад до запитуючого браузера.

З цих серверних технологій, ASP.NET і PHP, здається, мають саму велику частку ринку. ASP.NET має тенденцію бути більш широко використовуваним для корпоративних додатків і в інтранет. Почасти через масивні бази користувачів WordPress, PHP є найбільш широко використовуваною технологією веб-розробки.

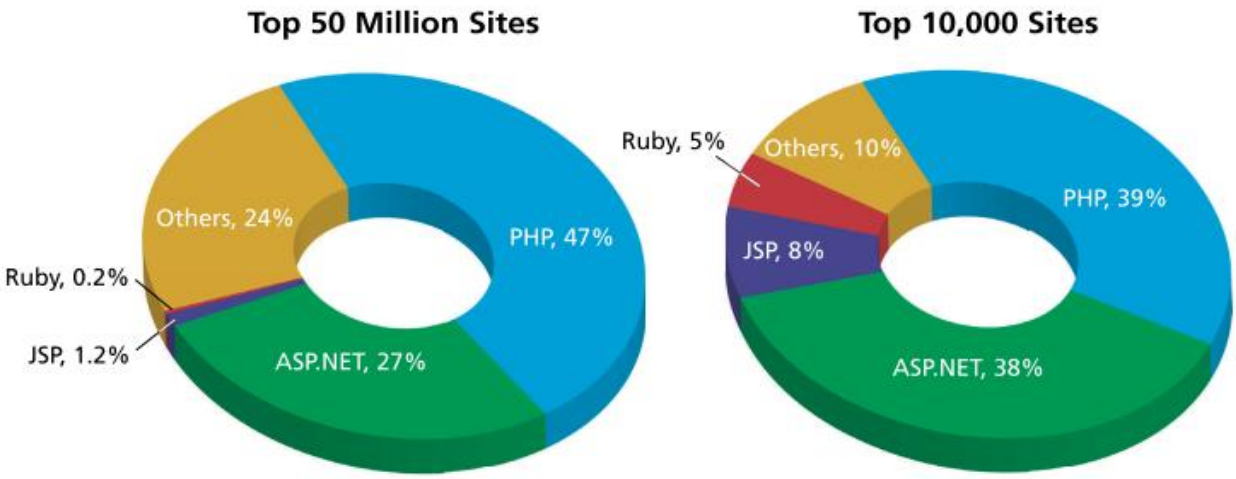


Рис. 2. Ринкова частка середовищ веб-розробки

В даному методичному посібнику наводяться конспект лекцій, завдання та методичні вказівки до виконання лабораторних робіт, методичні вказівки та завдання до самостійної роботи, екзаменаційні питання, список рекомендованої літератури.

1. КОНСПЕКТ ЛЕКЦІЙ

Лекції 1-3. Основи web-програмування

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

Інтернет та WEB

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

Від самого початку архітектура Інтернету мала наступний вигляд.

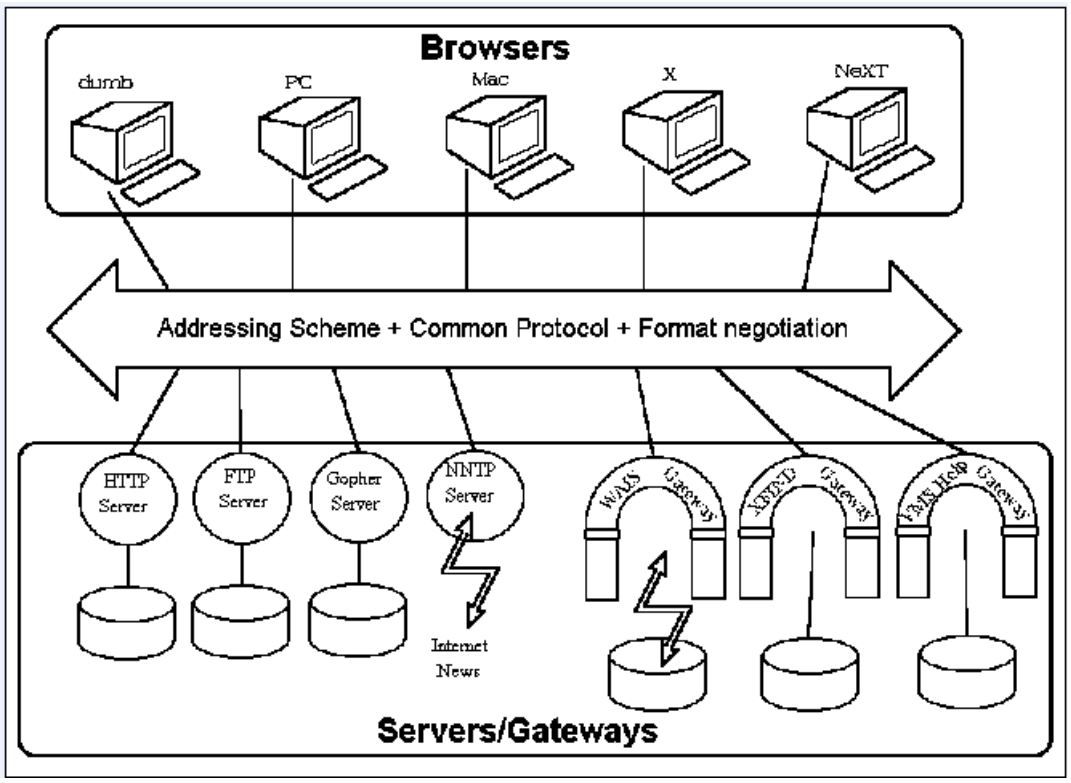


Рис. 3. Базова архітектура мережі Інтернет.

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

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

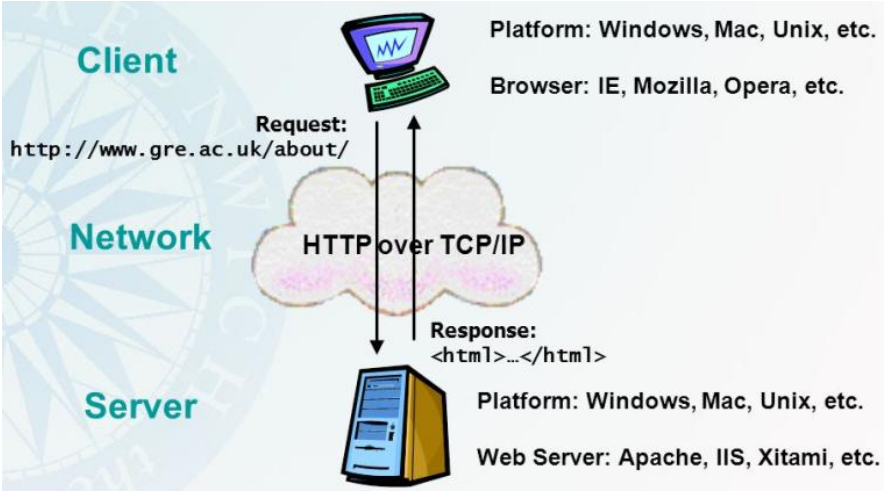


Рис. 4. Архітектура Web

Потік даних та взаємодія протоколів показані на Рис.5.

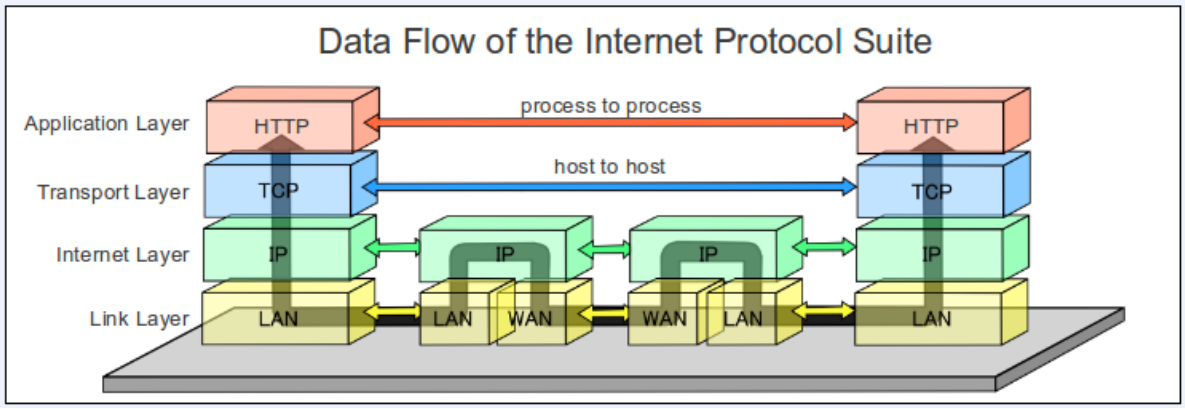


Рис.5. Потік даних та взаємодія протоколів для Web

Web-програмування

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

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

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

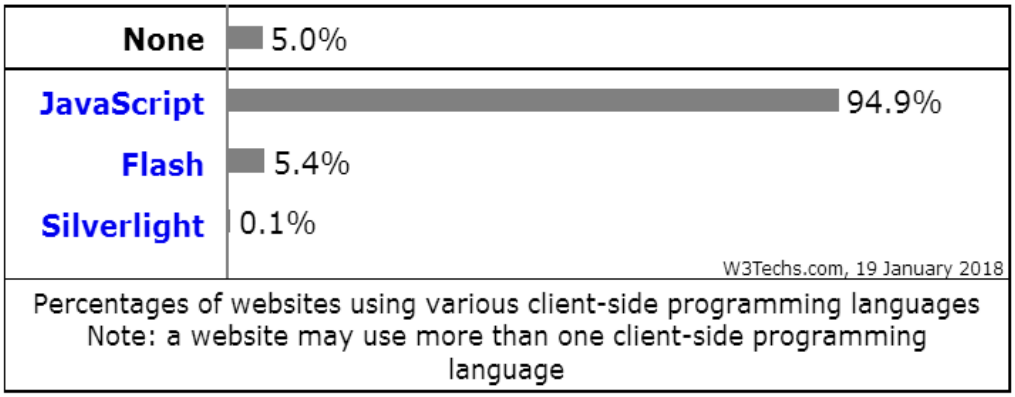


Рис.6. Використання мов програмування на стороні клієнта.

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

Таблиця 1. Використання JavaScript-бібліотек.

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

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

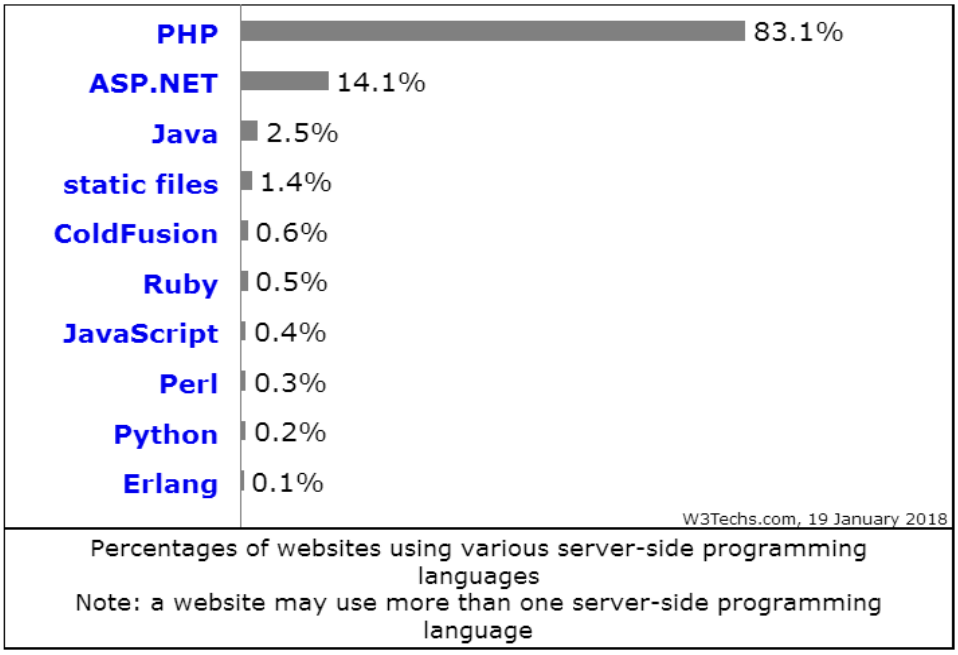


Рис. 7. Мови програмування в серверних web-технологіях

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

Налаштування робочого середовища розробки

Варіант 1. Установка вручну всіх програмних компонентів

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

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

Варіант 2. Попередньо упаковані установки

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

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

Варіант 3. Віртуальні сервери

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

Треба встановити наступне програмне забезпечення [8]:

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

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

Варіант 4. Онлайн середовища кодування

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

Редактори PHP та IDE

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

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

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

Фреймворки PHP

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

Найбільш популярними фреймворками PHP є такі:

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

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

CMS на базі PHP

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

Перерахуємо найбільш популярні CMS на базі PHP.

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

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

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

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

Робота за протоколом HTTP відбувається так:

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

Структура HTTP-повідомлення

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

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

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

HTTP-заголовок відділяється від HTTP-тіла порожнім рядком.

HTTP-запит (браузер – web-серверу)

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

Приклад (рядок запиту та заголовок запиту).

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

Відповідь:

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

(далі йде текст запитаної сторінки)

Рядок запиту (request line)

Має наступний формат:

Метод_запиту URL_ресурсу Версія_протоколу_HTTP

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

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

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

Заголовок HTTP-запиту

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

- Дата та загальна конфігураційна інформація

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

Тіло HTTP-запиту

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

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

HTTP-відповідь

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

Приклад.

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

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

Існують 5 класів відповідей:

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

Основні заголовки

- **Accept**. Інформує сервер про типи даних, які підтримуються браузером. Перечислення йде через кому. Використовується змінна оточення HTTP_ACCEPT.
- **Content-type**. Ідентифікує тип передаваємих даних. Іменування типів даних вказано в форматі стандарта MIME. Це тот самий формат передачі, який використовується методами GET і POST. Сервер ніяк не інтерпретує цей заголовок, а просто передає його сценарію через змінну оточення CONTENT_TYPE.
- **Content-length**. Цей заголовок містить довжину передаваємих даних в байтах при використанні метода передачі POST. Змінна оточення CONTENT_LENGTH.
- **Cookie**. В цьому заголовку зберігаються всі Cookies. Для установки Cookies застосовується заголовок Set-Cookie. Змінна оточення HTTP_COOKIE.

- **Location**. Отримавши цей заголовок разом з вказаним в ньому URL, браузер негайно переходить по вказаному URL.
- **Pragma**. Даний заголовок використовується для різних цілей, одна из яких – це заборона кешування документа.
- **Server**. Даний заголовок містить назву і версію програмного забезпечення сервера.
- **Referer**. Містить URL сторінки, звідки клієнт прийшов на нашу. Змінна оточення: HTTP_REFERER.
- **User-Agent**. Містить версію браузера. Змінна оточення: HTTP_USER_AGENT.

Стандарт MIME

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

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

Корисні програми для вивчення HTTP

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

Змінні оточення

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

- REMOTE_ADDR - IP-адрес хоста, що відправив запит.
- REMOTE_HOST - Ім'я хоста, с якого відправлено запит.
- REQUEST_METHOD - Метод, що був використаний при відправці запиту.

- QUERY_STRING - Інформація, що знаходиться в URL після знаку питання.
- SCRIPT_NAME - Віртуальний шлях до програми, яка повинна виконуватися.

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

echo getenv(“REMOTE_ADDR”);

Тема 3. Вступ до PHP.

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

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

PHP Is Not Part of Your Browser

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

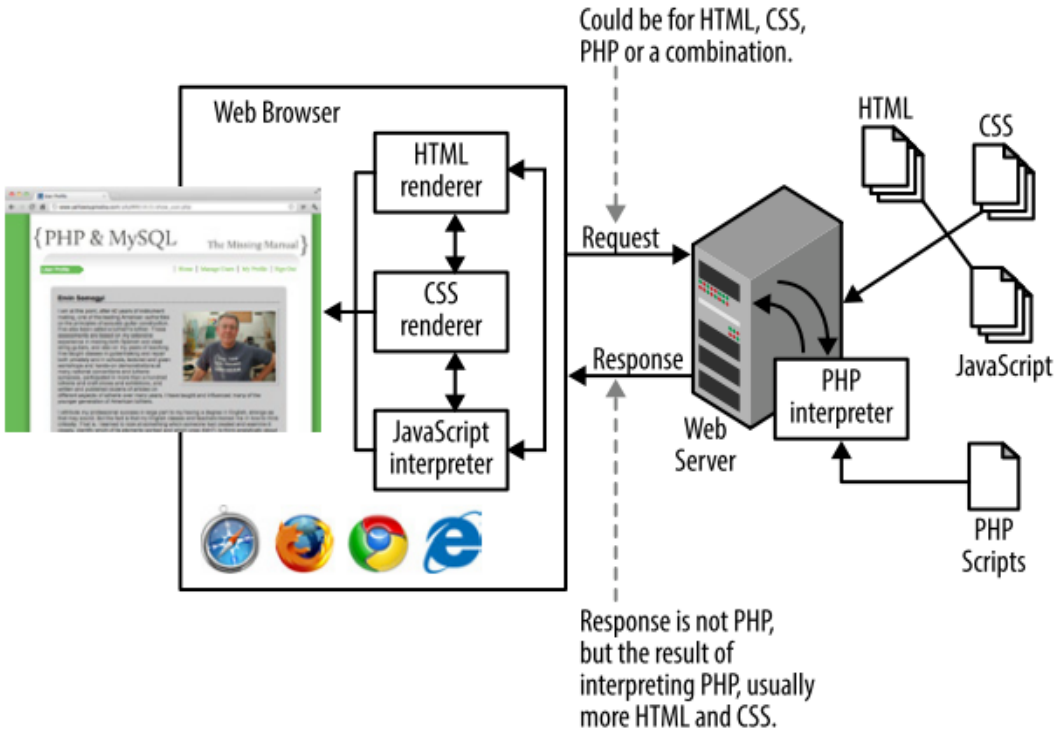


Рис. 8. Схема обробки запиту

Загальне визначення мови PHP

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

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

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

Приклад. Програма “Hello World”

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

Основні можливості мови

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

```
<?php
ECHO "Hello World"; // works
$variable = "Hello World";
echo $VARIABLE; // don't work
```

Приклад обробки форми

Файл formexample.php

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

Інша форма “Hello”-програми

Файл hello.php

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

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

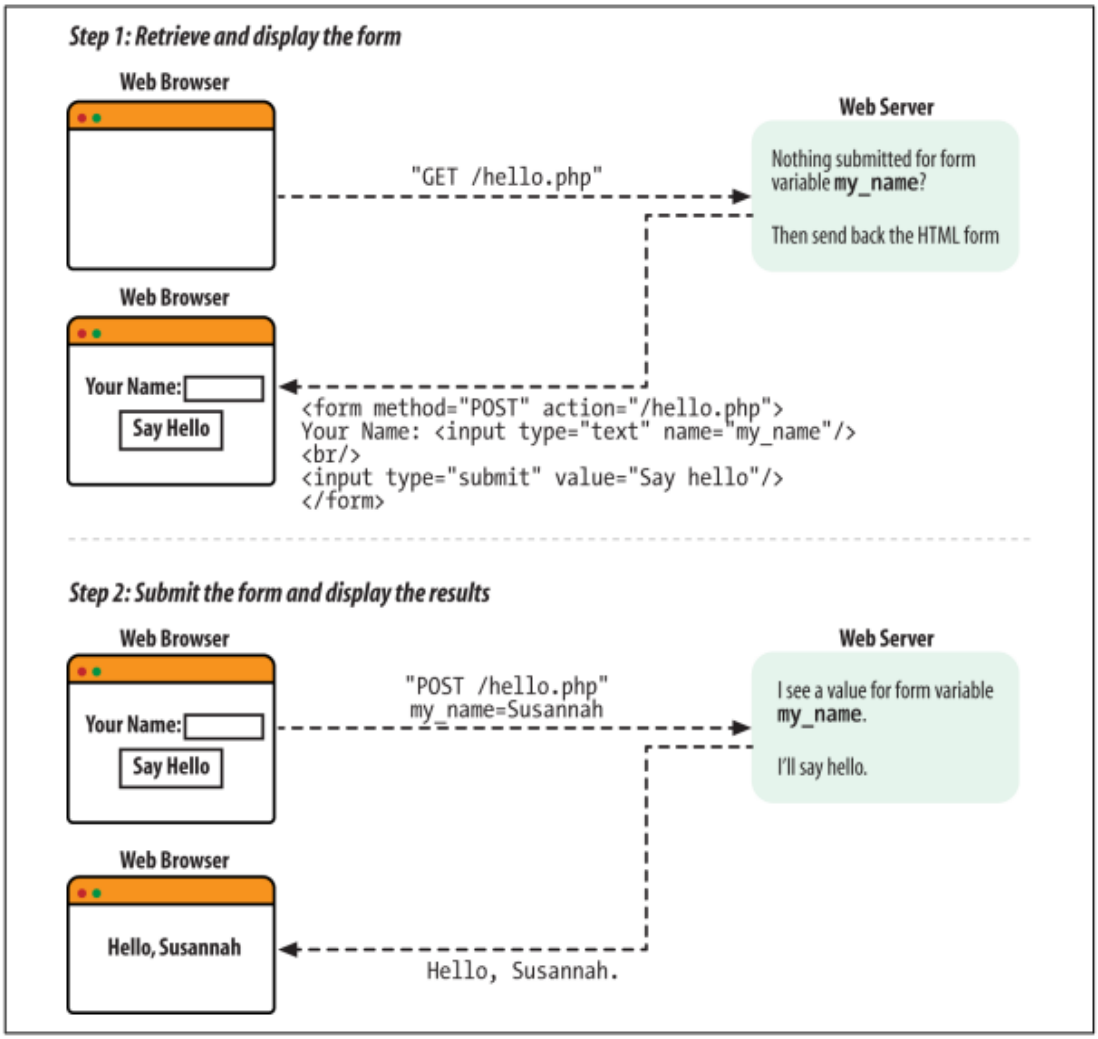


Рис. 9. Ілюстрація обробки форми

Відокремлення PHP коду від HTML розмітки

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

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

Echo-стиль <?= ?>

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

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

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

Приклад.

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

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

Закриваючий тег

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

Коментарі

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

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

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

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

Змінні в мові PHP

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

Локальні змінні

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

Приклад:

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

Сценарій виведе спершу 70, а потім 100.

Ключові слова global та static

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

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

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

Посилальні змінні. Жорсткі посилання. Операція &.

Приклад: $a=10; $b=&$a; $b=0;

Суперглобальні змінні

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

$_SESSION – Масив змінних, розміщених в сесіях PHP.
$_SERVER – Масив, що містить інформацію про сервер.

Типи даних

PHP підтримує вісім типів даних.Чотири скалярних типи:

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

Два змішані типи:

- array — масив;
- object — екземпляр класа.

Два спеціальних типи:

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

Приклад

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

Змінні змінних (Variable Variables). Це змінна, чиє імя містить іншу змінну.

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

Типи даних. Корисні функції

isset (ім'я_змінної) – повідомляє, чи існує змінна;.
unset(ім'я_змінної) – знищує (видаляє) вказану змінну;
empty(ім'я_змінної) – чи присвоєно змінній яке-небудь значення;
gettype(ім'я_змінної) – повертає тип вказаної змінної;
settype(ім'я_змінної, тип) - конвертує змінну в інший тип;
is_bool(ім'я_змінної) – перевіряє, чи є тип змінної логічним;
Функції is_numeric(), is_float(), is_int(), is_string(), is_object(), is_array() працюють за аналогією.

Константи

Визначаються за допомогою функції define(string_name, string_value).
Приклад

```
<?php
define('PI', 3.142);
echo PI;
```

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

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

Ключове слово const

Ви також можете використовувати ключове слово const для визначення констант.

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

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

Вбудовані константи

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

Рядки в PHP

- PHP-рядки являють собою серію байтів і не містять жодної інформації про те, як ці байти повинні бути переведені на символи.
- PHP зберігає довжину рядка разом із його вмістом і не спирається на закінчуючий символ, щоб позначити кінець рядка. Це допомагає зробити рядки безпечними, оскільки нульові символи в рядку не викликають плутанини.
- На 32-бітних системах рядок може займати до 2 Гб. Не існує конкретної межі щодо довжини рядка у 64-бітній системі PHP.
- Рядки можуть бути укладені в апострофи або в лапки. Це впливає на те, як рядки будуть оброблятися інтерпретатором.
- Рядки в апострофах залишаються майже незмінними.
  ```
  $win_path='C:\\Inetpub\\PHP\\';
  echo $win_path;     // Виведе   C:\Inetpub\PHP
  ```
- Рядки в лапках піддають попередній обробці.

- Деякі символьні послідовності, що починаються з ‘\’ замінюються спецсимволами
- Імена змінних (починаються з $) замінюються рядковими представленнями їх значень

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

Рядки Here-документ (heredoc)

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

Виклик зовнішньої програми

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

Приклад

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

(В PHP можна встановити безпечний режим з обмеженнями)

Символічні посилання

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

Приклад.

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

Фігурні дужки

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

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

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

```
echo "I can haz {$burger}s"; // I can haz Cheeseburgers
echo "I can haz { $burger }"; // I can haz { Cheeseburger }
```

Посилання на символи в рядках

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

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

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

Деякі функції для роботи з рядками

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

Проблема з підтримкою UTF-8

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

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

).

Приклад

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

Масиви в мові PHP

Масиви з числовими індексами. Індекси починаються з нуля.

```
<?php
    $zoo[0] = 'слон';
    $zoo[6] = 'крокодил';
    $zoo[4] = 'жираф';
    $zoo[] = 'осел';// Індекс дорівнює 7
    // або
    $zoo = array('лев', 'ведмідь', 'мавпа');
```

```
    echo count($zoo); // Количество элементов массива
$a = [1,2,3]; // PHP 5.4
?>
```

Функції створення масивів

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

Функція array_fill()

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

Результат:

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

Функція explode()

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

```
)
```

Випадкові елементи масиву

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

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

Приклад.

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

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

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

Асоціативні масиви

В асоціативних масивах використовується не числовий, а рядковий індекс.

```
<?php
     $pets['dog'] = 'Бульдог';
     $pets['cat'] = 'Шиншила';
     $pets['fish'] = 'Золота';
     // або
     $pets = array ('lizard' => 'Ігуана',
                          'spider' => 'Чорна вдова',
                          'parrot' => 'Ара');
     print_r ($pets);    // Друкування масиву
?>
```

Перегляд асоціативного масиву за допомогою foreach...as

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

```
?>
Результат:
copier: Copier & Multipurpose
inkjet: Inkjet Printer
laser: Laser Printer
photo: Photographic Paper
```

Перегляд асоціативного масиву за допомогою each та list

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

Багатовимірні масиви

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

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

Приклад.

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

```
                echo "$key=$value ";
            };
        echo "<br/>";
}
?>
```

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

Приклад. Створення багатовимірного асоціативного масиву.

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

Результат:

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

Сортування масивів

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

- **arsort** ($array [, $sort_flags]) – сортує масив $array в зворотному порядку зі збереженням відношення ключ-значення.

Приклад.

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

Сортування рядків відбувається за старшинством першої літери в алфавіті

Деякі функції для роботи з масивами

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

Суперглобальний масив $ _GET

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

Приклад 1.

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

Приклад 2.

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

Результат:

```
Array
(
```

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

Отримання даних форми

Приклад 1. Відправка даних

Файл login_form.html.

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

Приклад 1. Обробка форми

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

$ _POST - Якщо дані передані методом POST;

$ _GET - Якщо дані передані методом GET;

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

Файл handler.php

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

Як використовувати функцію filter_input

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

Синтаксис:

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

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

Приклад [21].

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

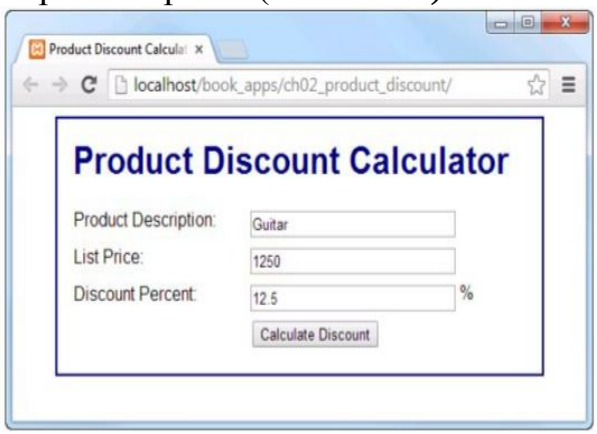


Рис.10. Форма

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

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

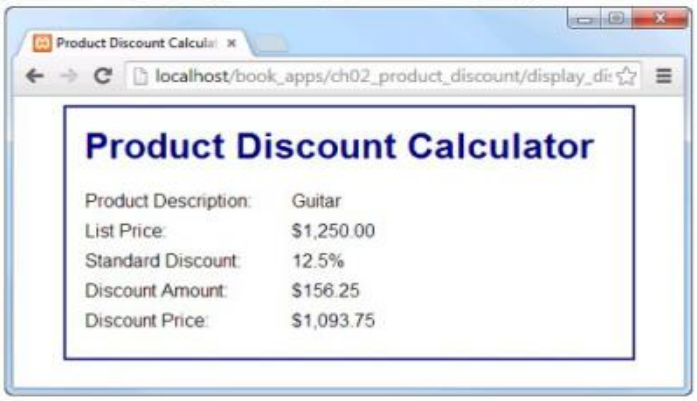


Рис.11. Результат обробки форми

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

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

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

Як отримати дані з radio button

Приклад.

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

Рис.12. Форма з radio button

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

А якщо нема значення за замовчуванням:

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

Як отримати дані з check box

Приклад.

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

Рис.13. Форма з check box

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

Як отримати дані з масиву check box

Приклад [21].

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

Рис.14. Форма з масивом check box

```
<?php
$toppings= filter_input(INPUT_POST,    ' top ' ,
FILTER_SANITIZE_SPECIAL_CHARS,   FILTER_REQUIRE_ARRAY) ;
?>
if  ($toppings !==  NULL)    {
$topl  =   $toppings [0];
$top2  =   $toppings [1];
$top3  =   $toppings [2];
}
```

Як отримати дані з drop-down list

Приклад [21].

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

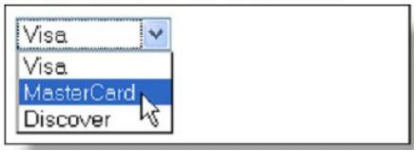


Рис. 15. Форма з випадаючим списком

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

Як отримати дані з list box

Приклад [21].

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

Рис.16. Форма 1 зі списком [21].

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

Рис.17. Форма 2 зі списком

```
<?php
$toppings  =  filter_input (INPUT_ POST,    ' top ' ,
FILTER_SANITIZE_SPECI AL_ CHARS,   FILTER_REQUIRE_ ARRAY );
if  ($toppings  !==  NULL)   {
foreach  ($toppings  as   $key=> $value)    {
echo  $key.‘ = '.$value.'<br>' ;          //  '0 = pep'   and
'1 = msh'
} }  else  {
echo  ' No   toppings  selected. ' ;
}
?>
```

Як отримати дані з text area

Приклад [21].

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

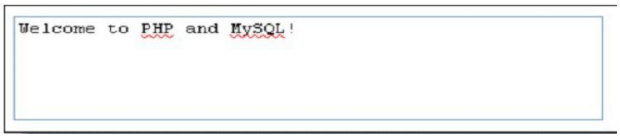


Рис.18. Форма з text area

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

Функції в PHP

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

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

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

Функція може мати нуль або більше параметрів.

Функція визначається ключовим словом *function*.

Декларування типів аргументів

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

Параметри за замовчуванням

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

Overloading Functions

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

Передача параметрів за посиланням

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

```
    $n1 = ucfirst(strtolower($n1));
    $n2 = ucfirst(strtolower($n2));
    $n3 = ucfirst(strtolower($n3));
}
?>
```

Результат: WILLIAM henry gatES
William Henry Gates

Повернення глобальних змінних

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

Результат: WILLIAM henry gatES
William Henry Gates

Змінне число параметрів

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

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

Return Type Declarations

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

Функції. Області видимості

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

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

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

Анонімні функції (Lambda) та замикання (Closure)

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

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

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

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

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

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

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

Основне призначення замикань – заміна глобальних змінних.

Генератори

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

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

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

Делегування генераторів

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

```
<?php
function generator() {
$a = [1,2,3];
yield from $a;
yield from range(4,6);
yield from sevenAteNine();
}
function sevenAteNine() {
for ($i=7; $i<10;$i++) {
yield $i;
}
}
$gen = generator();
foreach ($gen as $value) {
echo $value . PHP_EOL;
}
```

Date/Time Functions

Представлення часу в форматі *timestamp*.

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

Рядкове представлення дати.

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

Y – рік (чотири цифри)

F – назва місяця

j – номер дня в місяці

l – день тижня

. . .

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

Приклад.

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

Результат роботи програми показаний на Рис.19.

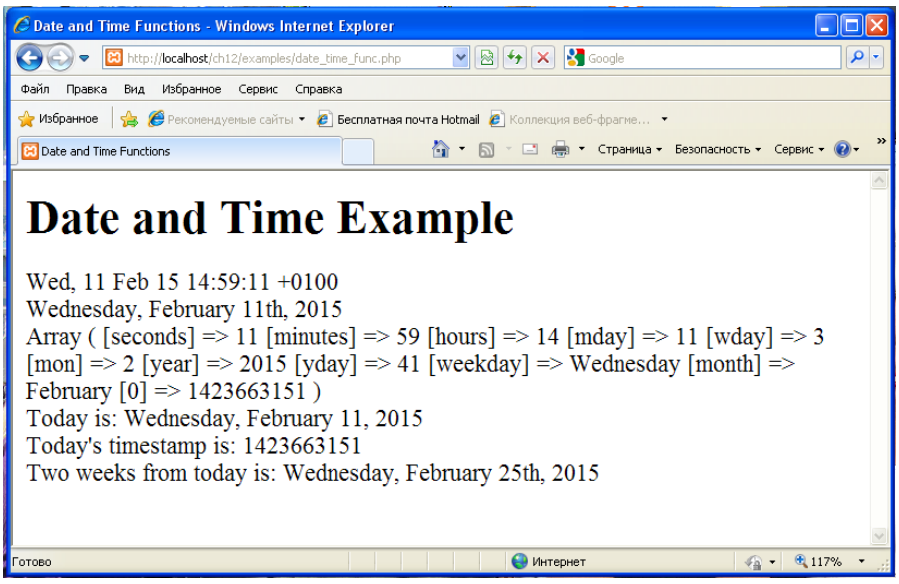


Рис.19. Результат роботи сценарію

Повторне використання коду (Code Reuse)

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

include - Підключає код із зазначеного файлу в поточний.

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

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

require_once - Працює аналогічно include_once, але з особливостями require.

Приклад.

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

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

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

Зв'язок PHP з MySQL

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

Визначення прав доступу до БД

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

Основні клієнтські команди MySQL

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

Клієнтська програма має інтерфейс командного рядка.

Щоб підключитися до сервера MySQL з використанням клієнта треба ввести команду

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

Після підключення до сервера треба вибрати БД

```
USE databasename;
```

Команда

```
SHOW TABLES;
```

дозволяє отримати список усіх таблиць.

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

```
SHOW COLUMNS FROM tablename;
```

Для перегляду значень таблиці:

```
SELECT * FROM tablename
```

Інтерфейс командного рядка

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

Результат показаний на Рис.20.

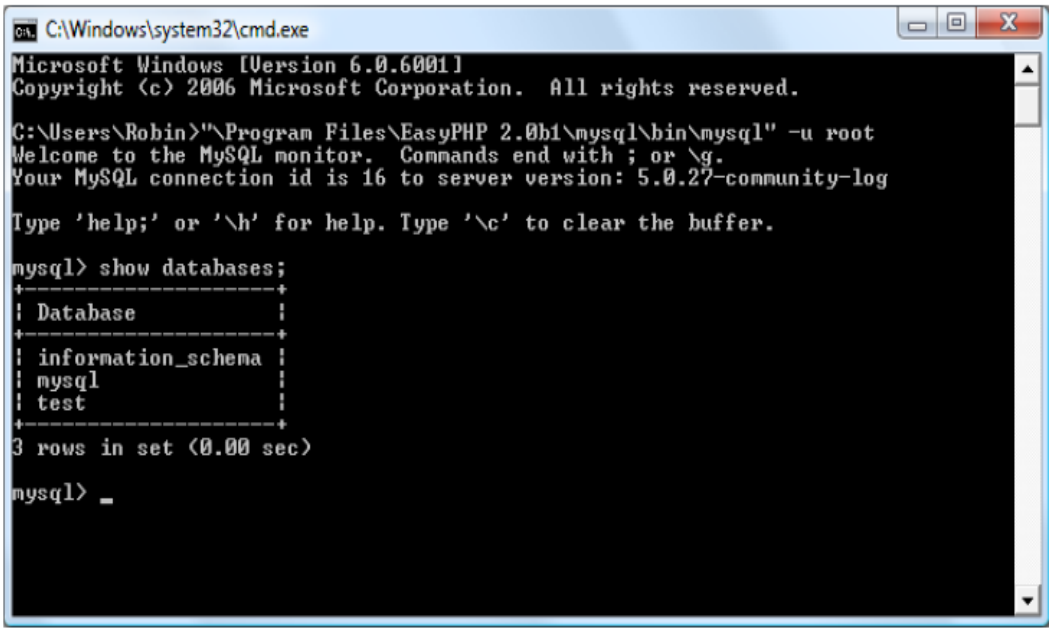


Рис.20. Інтерфейс командного рядка MySQL

Команди MySQL наведено на Рис.21.

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

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

Рис.21. Команди MySQL

Приклад. Створення бази даних.

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

Типи даних MySQL

Типи даних MySQL показані на Рис.22.

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

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

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

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

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

Data type
DATETIME
DATE
TIMESTAMP
TIME
YEAR

Рис.22. Типи даних MySQL

Графічний веб-інтерфейс для адміністрування бази даних MySQL

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

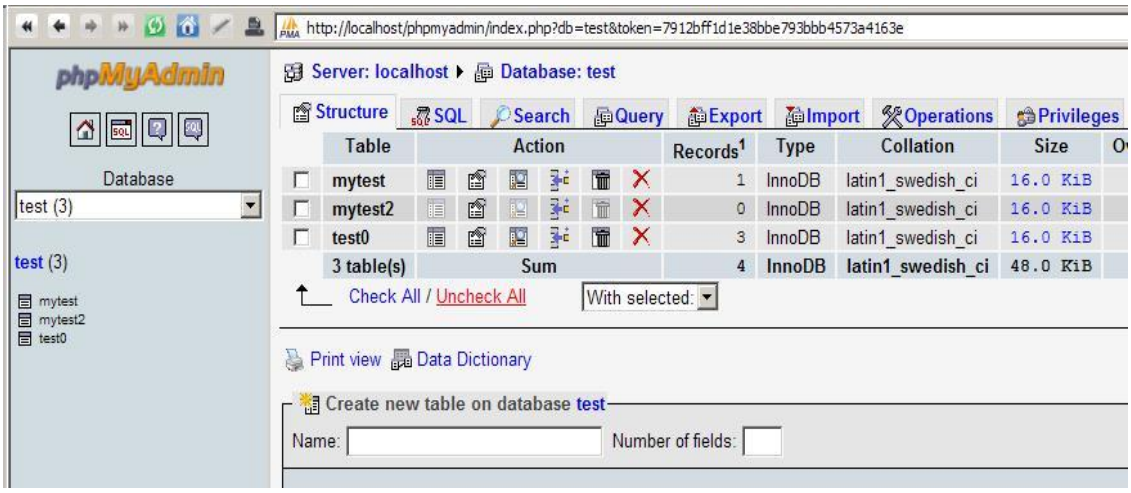


Рис.23. Вікно phpMyAdmin

Зв'язок PHP з MySQL

Існує три способи підключення до сервера MySQL з PHP:

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

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

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

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

Використання бібліотеки MySQLi

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

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

Якщо ви тестуєте сайт на локальному комп'ютері і у вас встановлений пакет XAMPP, то параметри повинні бути наступними

$server = 'localhost';
$username = 'root';
$password = '';

Основні функції бібліотеки MySQLi

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

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

```
<?php
if ($dbc = @mysqli_connect('localhost', 'root', '', 'myblog'))
{
$query = 'CREATE TABLE entries (
id INT UNSIGNED NOT NULL  AUTO_INCREMENT PRIMARY KEY,
```

```
title VARCHAR(100) NOT NULL,
entry TEXT NOT NULL,
date_entered DATETIME NOT NULL
) CHARACTER SET utf8';
if (@mysqli_query($query, $dbc))
{
     print '<p> The table has been created.</p>';
} else {
     print `. . . . . .'
}
```

Приклад: Вставка даних в таблицю БД [6]

Форма для додавання даних показана на Рис.24.

Рис.24. Форма вставки даних в таблицю БД

```
<form action="add_entry.php" method="post">
<p>Entry Title: <input type="text" name="title" size="40"
maxsize="100"></p>
<p>Entry Text: <textarea name="entry" cols="40"
rows="5"></textarea></p>
<input type="submit" name="submit" value="Post This Entry!">
</form>
.....
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
$problem = FALSE;
if (!empty($_POST['title']) && !empty($_POST['entry'])) {
 $title =  mysqli_real_escape_string($dbc,
trim(strip_tags($_POST['title'])));
 $entry = trim(strip_tags($_POST['entry'])); //можлива
ін'єкція! p.366
} else {
     print '<p style="color: red;">Please submit both a title
and an entry.</p>';
     $problem = TRUE;
}
if (!$problem) {
$dbc = mysqli_connect('localhost', `root', `', 'myblog');
. . . .    mysqli_set_charset($dbc, 'utf8');
```

```
     $query = "INSERT INTO entries (id, title, entry, date_entered)
VALUES (0, '$title', '$entry', NOW())";
     if (@mysqli_query($dbc, $query)) { . . . }
```

Приклад. Отримання даних з бази даних [6]

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

Результат показано на Рис.25.

Рис.25. Отримання даних з бази даних

Приклад. Видалення даних у базі даних [6]

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

Рис.26. Вибір Delete на сторінці view_entries.php

Після видалення треба вивести сторінку (Рис.27).

Рис.27. Сторінка після видалення

Реалізація має наступний вигляд [6]:

```
// Файл delete_entry.php
$dbc = mysqli_connect( . . . . );
if (isset($_GET['id']) && is_numeric($_GET['id']) ) {
$query = "SELECT title, entry FROM entries WHERE
id={$_GET['id']}";
if ($r = mysqli_query($dbc, $query)) {
$row = mysqli_fetch_array($r);
print '<form action="delete_entry.php" method="post">
<p>Are you sure you want to delete this entry?</p>
<p><h3>' . $row['title'] . '</h3>' .
$row['entry'] . '<br>
<input type="hidden" name="id" value="' . $_GET['id'] . '">
<input type="submit" name="submit" value="Delete this En-
try!"></p>
</form>';
} else {     // Друкує помилку  }
} elseif (isset($_POST['id']) && is_numeric($_POST['id'])) {
$query = "DELETE FROM entries WHERE id={$_POST['id']} LIMIT 1";
 $r = mysqli_query($dbc, $query);
 if (mysqli_affected_rows($dbc) == 1) {
print '<p>The blog entry has been deleted.</p>';
 } else {
     print '<p style="color: red;">Could not delete the blog
entry because:<br>' .
     mysqli_error($dbc) . '.</p><p>The query being run was: ' .
$query . '</p>';
}
. . . . . .
```

Приклад. Оновлення даних у базі даних [6]

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

Рис.28. Редагування даних

Після редагування треба вивести сторінку (Рис.29):

Рис.29. Сторінка після редагування

Реалізація має наступний вигляд [6]:

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

```
     print '<p style="color: red;">Please submit both a title and an
entry.</p>';
     $problem = TRUE;
     }
     if (!$problem) {
     $query = "UPDATE entries SET title='$title', entry='$entry'
WHERE id={$_POST['id']}";
      $r = mysqli_query($dbc, $query);
      if (mysqli_affected_rows($dbc) == 1) {
     print '<p>The blog entry has been updated.</p>';
     } else {    // Помилка }
     }
```

Використання бібліотеки MySQLi

2) Об’єктно-орієнтований стиль

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

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

Далі необхідно викликати методи об’єкта mysqli

Приклад [19]

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

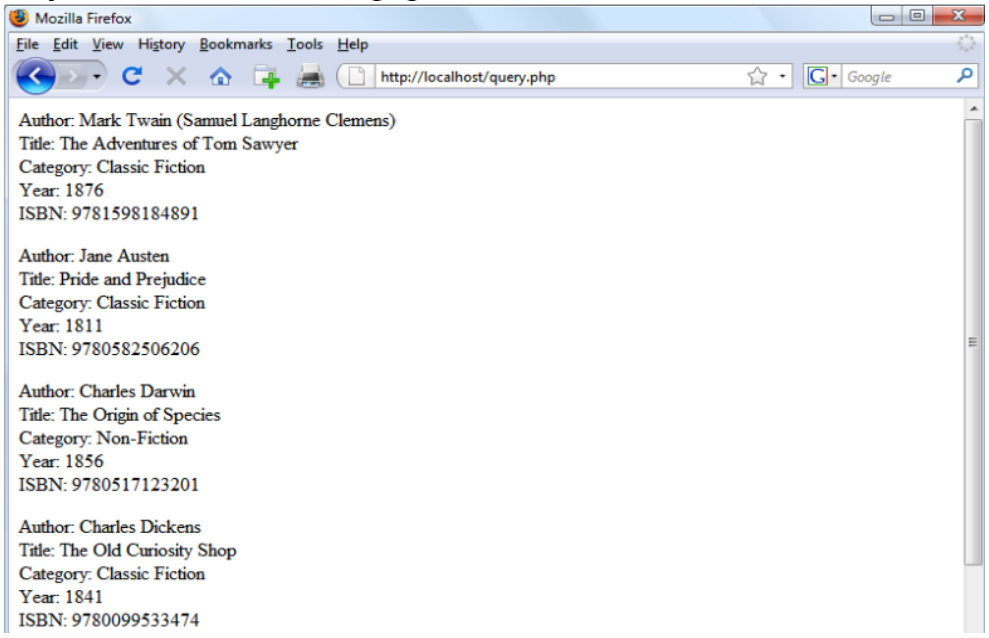


Рис.30. Виведення інформації з бази даних

Реалізація:

```
<?php // login.php
  $hn = 'localhost';
  $db = 'publications';
  $un = 'username';
  $pw = 'password';
?>
<?php //fetchrow.php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);
```

```
  $query  = "SELECT * FROM classics";
  $result = $conn->query($query);
  if (!$result) die($conn->error);
  $rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j){
          $result->data_seek($j);
          $row = $result->fetch_array(MYSQLI_ASSOC);
          echo 'Author: '   . $row['author']   . '<br>';
          echo 'Title: '    . $row['title']    . '<br>';
          echo 'Category: ' . $row['category'] . '<br>';
          echo 'Year: '     . $row['year']     . '<br>';
          echo 'ISBN: '     . $row['isbn']     . '<br><br>';
}
$result->close();
  $conn->close();
?>
```

Приклад [19]

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

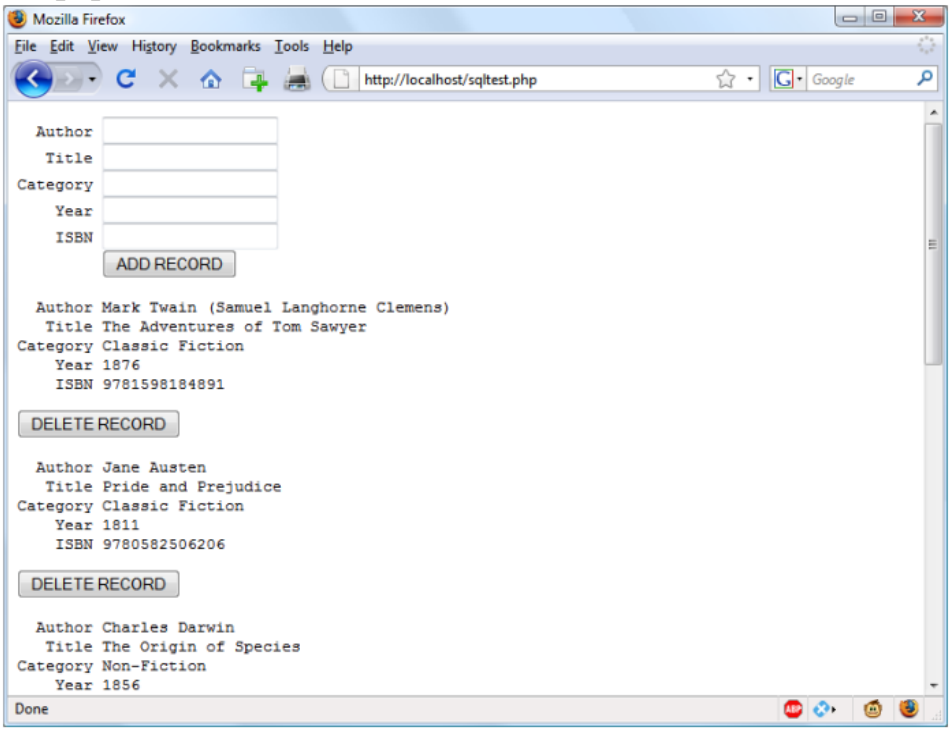


Рис.31. Додавання та видалення інформації

Реалізація:

```
<?php // sqltest.php
  require_once 'login.php';
  $conn = new mysqli($hn, $un, $pw, $db);
  if ($conn->connect_error) die($conn->connect_error);
  if (isset($_POST['delete']) && isset($_POST['isbn']))
  {
    $isbn   = get_post($conn, 'isbn');
    $query  = "DELETE FROM classics WHERE isbn='$isbn'";
    $result = $conn->query($query);
    if (!$result) echo "DELETE failed: $query<br>" .
```

```
        $conn->error . "<br><br>";
    }
  if (isset($_POST['author'])   &&
      isset($_POST['title'])    &&
      isset($_POST['category']) &&
      isset($_POST['year'])     &&
      isset($_POST['isbn']))
    {
      $author   = get_post($conn, 'author');
      $title    = get_post($conn, 'title');
      $category = get_post($conn, 'category');
      $year     = get_post($conn, 'year');
      $isbn     = get_post($conn, 'isbn');
      $query    = "INSERT INTO classics VALUES" .
        "('$author', '$title', '$category', '$year', '$isbn')";
      $result   = $conn->query($query);
      if (!$result) echo "INSERT failed: $query<br>" .
        $conn->error . "<br><br>";
    }
  echo <<<_END
    <form action="sqltest.php" method="post"><pre>
      Author <input type="text" name="author">
       Title <input type="text" name="title">
    Category <input type="text" name="category">
        Year <input type="text" name="year">
        ISBN <input type="text" name="isbn">
             <input type="submit" value="ADD RECORD">
    </pre></form>
  _END;
    $query  = "SELECT * FROM classics";
  $result = $conn->query($query);
    if  (!$result)  die  ("Database  access  failed:  "  .  $conn-
>error);
    $rows = $result->num_rows;
  for ($j = 0 ; $j < $rows ; ++$j)
    {
      $result->data_seek($j);
      $row = $result->fetch_array(MYSQLI_NUM);
      echo <<< _END
    <pre>
      Author $row[0]
       Title $row[1]
    Category $row[2]
        Year $row[3]
        ISBN $row[4]
    </pre>
    <form action="sqltest.php" method="post">
    <input type="hidden" name="delete" value="yes">
    <input type="hidden" name="isbn" value="$row[4]">
    <input type="submit" value="DELETE RECORD"></form>
  _END;
    }
  $result->close();
```

```
  $conn->close();
  function get_post($conn, $var)
  {
    return $conn->real_escape_string($_POST[$var]);
  }
?>
```

PHP Data Objects (PDO)

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

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

PDO входить до складу PHP з версії 5.1.

Зв'язок PHP з MySQL через PDO

З’єднання

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

PDO. Виконання запитів

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

Данні можна отримати за допомогою метода fetch()

```
$stmt = $pdo->query('SELECT name FROM users');
while ($row = $stmt->fetch())
{
    echo $row['name'] . "\n";
}
```

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

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

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

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

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

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

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

Інший варіант:

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

Приклад [26]

1) Створимо базу даних misc

2) Створимо таблицю users

// Файл misc.sql

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

3) Файл pdo.php

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

4) Файл first.php

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

Результат:

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

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

mysql> select * from users;

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

4) Файл user1.php

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

Результат показаний на Рис.32

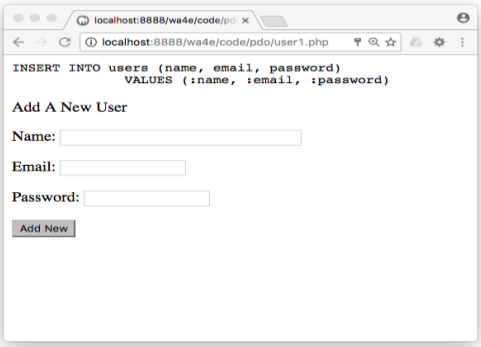


Рис.32. Додавання даних через PDO

Результат після додавання:

```
mysql> select * from users;
```

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

5) Файл user2.php

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

В результаті отримаємо наступну сторінку (Рис.33):

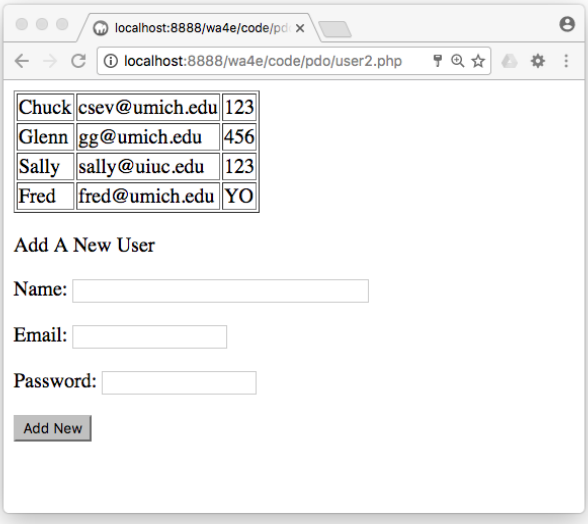


Рис.33. Сторінка після додавання даних

6)Файл user2del.php

Сторінка до видалення показана на Рис.34.

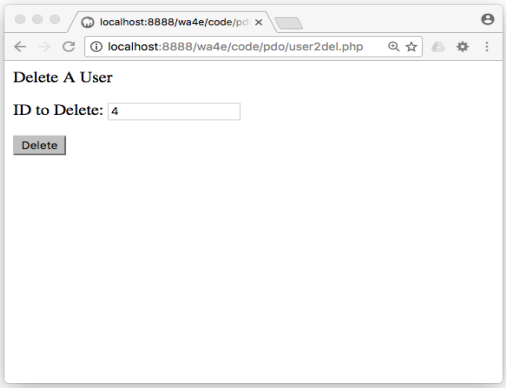


Рис.34. Сторінка до видалення

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

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

Сторінка після видалення показана на Рис.35.

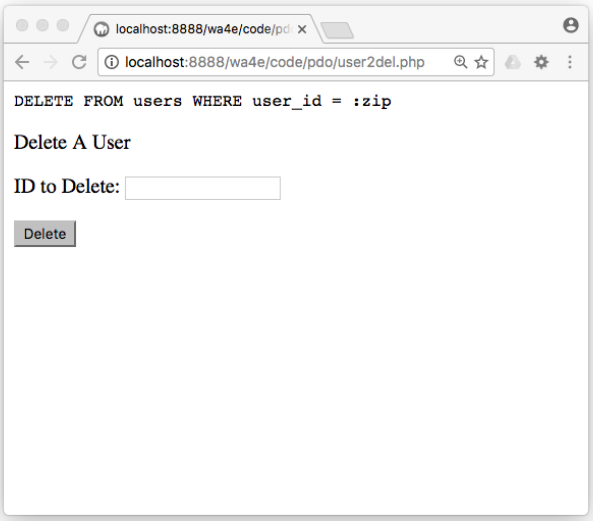


Рис.35. Сторінка після видалення

7)Файл user3.php

Форма з кнопкою видалення показана на Рис.36.

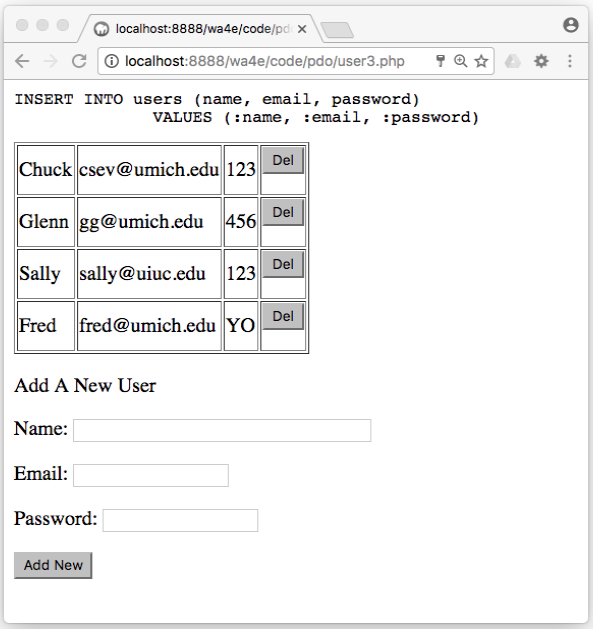


Рис.36. Форма з кнопкою видалення

```
<?php
require_once "pdo.php";
if ( isset($_POST['name']) && isset($_POST['email'])
      && isset($_POST['password'])) {
$sql = "INSERT INTO users (name, email, password)
VALUES (:name, :email, :password)";
echo("<pre>\n".$sql."\n</pre>\n");
$stmt = $pdo->prepare($sql);
$stmt->execute(array(
        ':name' => $_POST['name'],
        ':email' => $_POST['email'],
        ':password' => $_POST['password']));
}
if ( isset($_POST['delete']) && isset($_POST['user_id']) ) {
    $sql = "DELETE FROM users WHERE user_id = :zip";
    echo "<pre>\n$sql\n</pre>\n";
    $stmt = $pdo->prepare($sql);
    $stmt->execute(array(':zip' => $_POST['user_id']));
}
$stmt = $pdo->query("SELECT name, email, password, user_id FROM
users");
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
<html><head></head><body><table border="1">
<?php
foreach ( $rows as $row ) {
    echo "<tr><td>";
    echo($row['name']);
    echo("</td><td>");
    echo($row['email']);
    echo("</td><td>");
    echo($row['password']);
```

```
        echo("</td><td>");
        echo('<form method="post"><input type="hidden" ');
echo('name="user_id" value="'.$row['user_id'].'">'."\n");
echo('<input type="submit" value="Del" name="delete">');
echo("\n</form>\n");
        echo("</td></tr>\n");}?>
    </table>
    <p>Add A New User</p>
    <form method="post">
    <p>Name:<input type="text" name="name" size="40"></p>
    <p>Email:<input type="text" name="email"></p>
    <p>Password:<input type="password" name="password"></p>
    <p><input type="submit" value="Add New"/></p>
    </form>
    </body>
```

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

Cookies

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

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

Життєвий цикл cookie виглядає так:

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

Життєвий цикл cookies показаний на Рис.37.

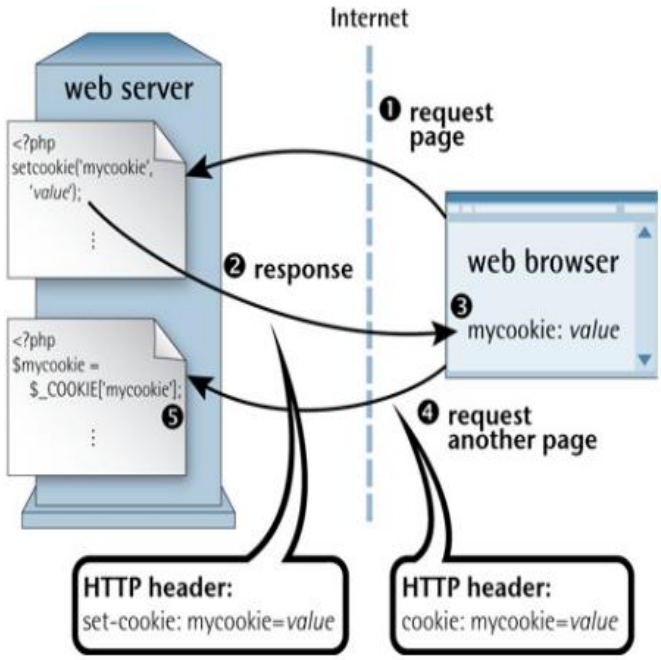


Рис.37. Життєвий цикл cookies

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

Установка cookie

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

Тільки перший параметр є обов'язковим.
Якщо термін не вказано, cookie-файл буде дійсним тільки протягом сеансу.

Доступ до cookie

Отримати значення cookie можна двома способами.

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

Приклад

```
<?php   // Встановлюємо cookie
setcookie ("TestCookie", "value");//До кінця сеансу
setcookie ("TestCookie", "value", time()+3600);//На 1 годину
setcookie ("TestCookieArray[1]", "value1");  //Массив Cookie
setcookie ("TestCookieArray[2]", "value2");
?>
```

Cookie стане доступною тільки після перезавантаження сторінки.

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

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

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

Створення БД сайту та таблиці користувачів.

Файл sql.sql

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

Головний файл

```
<?php # Script 3.4 - index.php
$page_title = 'Welcome to this Site!';
include ('./includes/header.html');
?>
<h1 id="mainhead">Big Header</h1>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<h2>Subheader</h2>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
<p>This is where you'll put the main page content.</p>
```

```
<?php
include ('./includes/footer.html');
?>
```

Головна сторінка сайту показана на Рис.38.

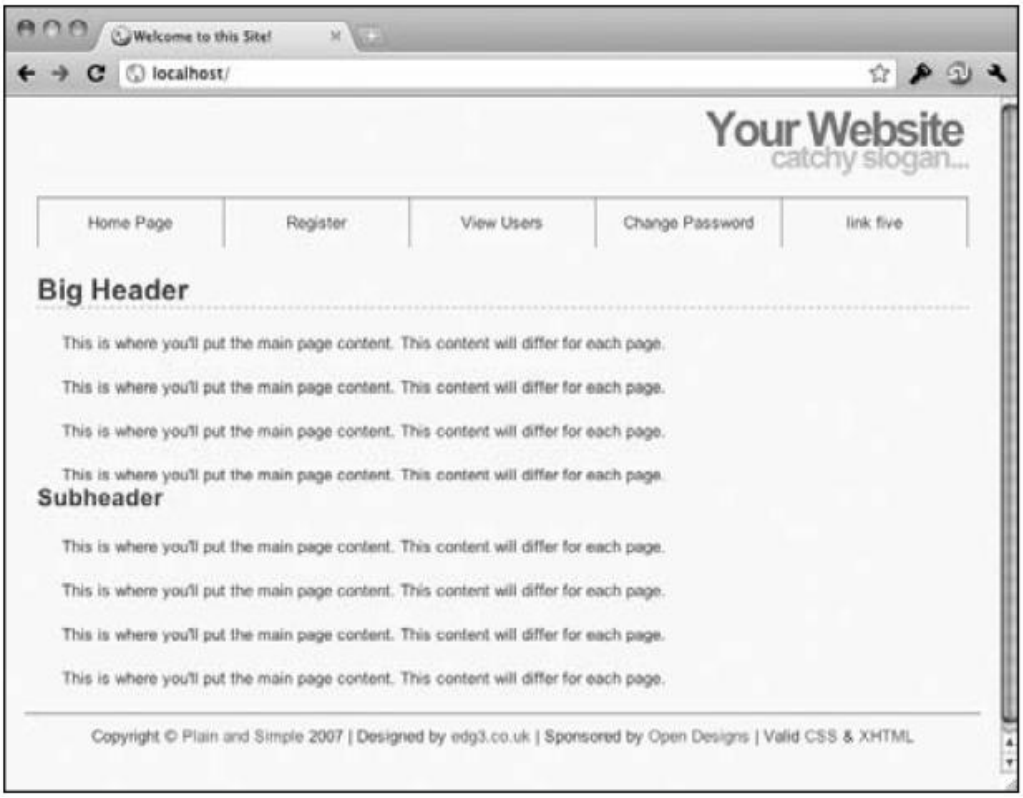


Рис.38. Головна сторінка сайту

Файл header.html

```
<html >
<head><title> <?php echo $page_title; ?> </title>
     <link rel="stylesheet" href="includes/style.css"
type="text/css‟ media="screen"/>
<meta http-equiv="content-type" content="text/html;
charset=utf-8"/>
</head>
<body>
<div id="header"><h1>Your Website</h1><h2>catchy
slogan...</h2></div>
<div id="navigation">
<ul>
     <li><a href="index.php">Home Page</a></li>
     <li><a href="register.php">Register</a></li>
     <li><a href="view_users.php">View Users</a></li>
<li><a href="password.php">Change Password</a></li>
     <li> <?php  // Create a login/logout link:
if ( (isset($_COOKIE['user_id'])) &&
(basename($_SERVER['PHP_SELF']) != 'logout.php') ){
echo '<a href="logout.php">Logout</a>';
} else {  echo '<a href="login.php">Login</a>';}
?></li></ul>
</div>
<div id="content">
```

```
 <!-- Start of the page-specific content. -->
```
Стилі для навігації
```
#navigation {
background:#fafafa;
border-right:1px solid #999;
margin:0 auto;
width:750px;
height:40px;
list-style:none;
}
#navigation li {
border-left:1px solid #999;
float:left;
width:149px;
list-style:none;
}
#navigation a {
color:#555;
display:block;
line-height:40px;
text-align:center;
}
#navigation a:hover {
background:#e3e3e3;
color:#555;
}
#navigation .active {
background:#e3e3e3;
color:#777;
}
```
Форма реєстрації показана на Рис.39

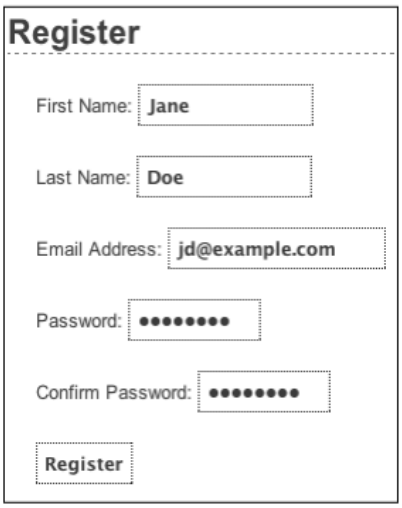


Рис.39. Форма реєстрації

Фрагменти файлу register.php
```
<?php
$page_title = 'Register';
include ('includes/header.html');
require ('mysqli_connect.php');
$errors = array(); // Initialize an error array.
     // Check for a first name:
     if (empty($_POST['first_name'])) {
          $errors[] = 'You  forgot  to  enter  your  first  name.';
```

```
          } else {
               $fn = mysqli_real_escape_string($dbc,
trim($_POST['first_name']));
          }
     .  .  .
     if (empty($errors)) {    // If everything's OK.
     $q = "INSERT  INTO  users (first_name, last_name, email, pass,
registration_date) VALUES ('$fn', '$ln', '$e', SHA1('$p'), NOW() )";
               $r = @mysqli_query ($dbc, $q); // Run the query.
               if ($r) {   // If it ran OK.
                        echo  '<h1>Thank you!</h1><p>You are now
registered. </p>
                          <p><br /></p>';
               } else { // If it did not run OK.
               echo '<h1>System Error</h1>
               <p class="error">You could not be registered due to a
system error.</p>';
               echo '<p>' . mysqli_error($dbc) . '<br /><br />Query: ' .
$q . '</p>
               }   // End of if ($r) IF.
               mysqli_close($dbc);    // Close the database connection.
               include ('includes/footer.html');
               exit();
     . . .
```

Файл mysqli_connect.php

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

Фрагменти файлу login_page.inc.php

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

```
</form>
<?php include ('includes/footer.html'); ?>
```

Фрагменти файлу login.php

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

Фрагменти файлу login_functions.inc.php

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

Робота з сесіями

- Сесії (session) в PHP призначені для зберігання на сервері даних при переходах користувача між різними сторінками веб-сайту.
- Сесією (сеансом) називають період часу, який користувач проводе на сайті.
- При відкритті сесії користувачу присвоюється його унікальний ідентифікатор сеансу (SID), який зберігається на боці клієнта у вигляді cookie.

- Якщо підтримка cookie відключена, то SID автоматично додається до URL на даному сайті.
- У той же час web-сервер зберігає цей SID на своєму боці та записує сесійні дані в файли на своєму жорсткому диску.

Ілюстрація механізму сесій показана на Рис.40.

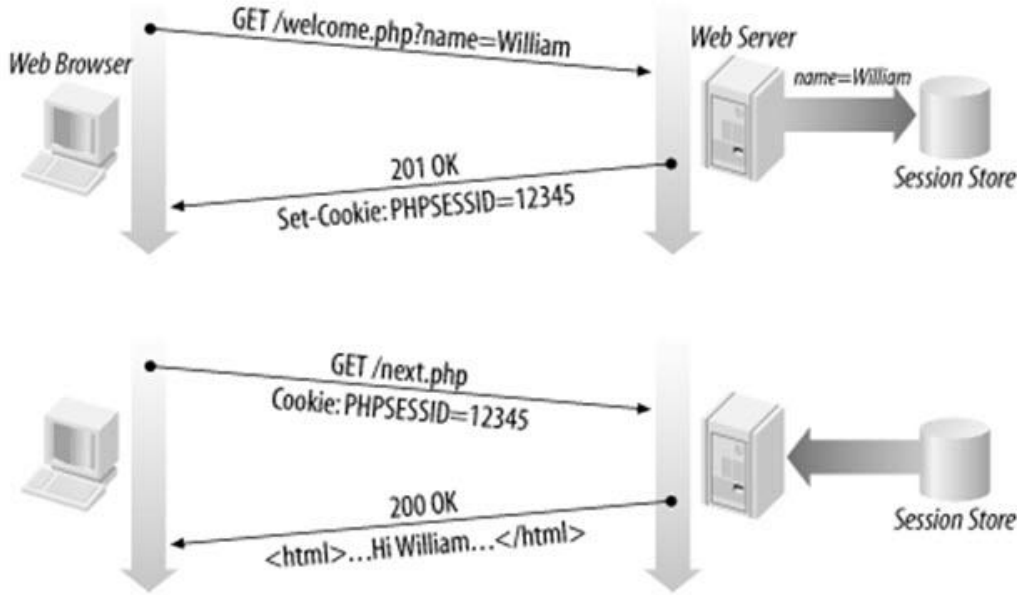


Рис.40. Ілюстрація механізму сесій

Відслідковування сеанса в PHP

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

Приклад.

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

```
    }
?>
```

Робота з файлами

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

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

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

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

Відкриття файлів

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

Ви повинні передати два параметри fopen ():

• Назва файлу у вашій файловій системі.

• Режим файлу, з яким ви хочете його відкрити.

Режими файлу

r — відкрити файл лише для читання;

r+ — відкрити файл для читання і запису;

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

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

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

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

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

Читання файлів

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

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

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

- fgetcsv() – прочитати рядок із файлу з заданим вказівником та проаналізувати поля для полів CSV.

- file_get_contents() – взяти файл з заданим іменем та прочитайте його вміст в рядок.
- readfile() – прочитати файл з заданим іменем та записати вміст у вихідний буфер.
- file() – прочитати цілий файл у масив.

Запис у файли

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

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

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

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

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

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

Ось простий приклад використання деяких з цих функцій:

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

Функції файлової системи

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

Директорії

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

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

Інформація про файл

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

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

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

Ось приклад з керівництва PHP:

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

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

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

Завантаження файлів на сервер

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

```
//Файл index.html
<html><head><title> Завантаження файлів на сервер </title>
</head>
<body>
<h2><b> Форма для завантаження файлів </b></h2>
<form action= "upload.php" method="post"
                          enctype="multipart / form-data">
<input type= "file" name="filename"><br>
<input type=" submit" value="Завантажити"><Ьr>
</ form>
< /body>
</html>
```

Будемо мати наступну сторінку (Рис.41).

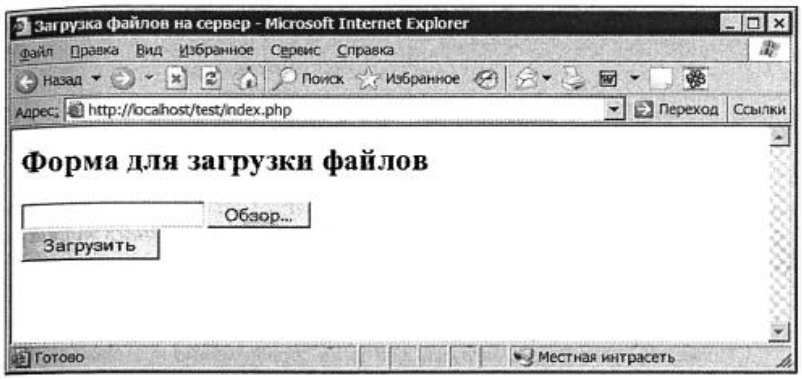


Рис.41. Форма завантаження файлів на сервер

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

Елемент введення цієї форми повинен мати тип file.

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

Характеристики завантаженого файлу доступні через двовимірний суперглобальний масив $_FILES:

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

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

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

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

Код скрипта обробки форми (upload.php)

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

Регулярні вирази (regex) в PHP

Regex представляють собою шаблони для пошуку в рядках.

Є два класи regex:

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

Мова PHP дозволяє використовувати Perl-сумісні regex.

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

Обмежувачі

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

Метасимволи в регулярних виразах

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

Загальні типи символів

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

Таблиця 2. Загальні типи символів

Символ	Тип символів
\d	Будь-яка десяткова цифра
\h	Будь-який горизонтальний символ пробілу
\s	Будь-який символ пробілу
\v	Будь-який вертикальний символ пробілу
\w	Будь-який символ, який утворює "слово"
\D	Будь-який символ, який не є десятковою цифрою
\H	Символ, який не є горизонтальним пробілом
\S	Будь-який символ, який не є пробілом

\V	Символ, який не є символом вертикального пробілу
\W	Будь-який символ "не слова"

Межі (Boundaries)

Символ "слово" - це будь-яка літера, цифра або символ підкреслення.
Межа слова - це позиція в рядку, де починається або закінчується слово.
Символи межі наведені в Таблиці 3.

Таблиця 3. Символи межі

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

Класи символів

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

Збіг більше ніж один раз

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

Виділення груп

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

```
<?php
$subject = "I can haz Cheeseburgers";
```

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

Буде виведено рядок "I can haz Cheeseburger"

Жадібність і лінивість (Greed and Laziness)

За замовчуванням, відповідність є "жадібною“

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

Виведе: "html“

Квантифікатор * є “жадібним” і шукає саму довгу послідовність

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

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

Отримання всіх збігів

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

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

Найбільш вживаною функції є preg_match().

Синтаксис:

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

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

Наприклад:

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

Функція preg_replace()

Синтаксис:

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

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

Наприклад:

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

Лекції 7-9. ООП в PHP, огляд PHP-фреймворків, безпека.

Тема 7. Класи в PHP. Створення та клонування об'єктів. Успадкування. Перевизначення методів. Інтерфейси. Обробка виняткових ситуацій. Трейти.

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

Синтаксис та обмеження для успадкування наведені в Таблиці 4.

Таблиця 4. 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

Приклад. 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;
         }
     }
     ?>
```

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 функція 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');
```

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

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

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

Метод hello() не визначений. Тому викликається метод __call().

Розглянемо ще один приклад:

```
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();
```

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";
    }
}
```

Розглянемо 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() { }
}
```

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
```

Константи класу

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

Звертання до констант класу

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

Статичні властивості та методи PHP 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 )
}
```

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();
     }
?>
```

Приклад 2.

```
<?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($argv[1]);
    }
if (!is_numeric($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

```
<?php
trait Singleton
```

```
{
     private static $instance;
     public static function getInstance() {
          if (!(self::$instance instanceof self)) {
               self::$instance = new self;
          }
          return self::$instance;
     }
}
class UsingTraitExample
{
     use Singleton;
}
$object = UsingTraitExample::getInstance();
var_dump($object instanceof UsingTraitExample); // true
```

Тема 8. Шаблон MVC. Огляд сучасних фреймворків. Використання Composer. Фреймворки Yii , Laravel, CodeIgniter, Zend Framework. Стандарти PSR.

Код PHP та HTML-шаблон сторінки

Є декілька підходів до розділення шаблону сторінки та коду сценарію [18].

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 (Дивись Рис.42).

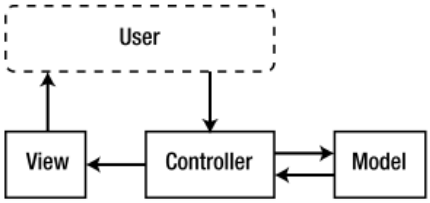


Рис.42 Шаблон Модель-Представлення-Контролер

Огляд сучасних PHP-фреймворків

Фреймворк (англ. Framework, каркас, платформа, структура) – інфраструктура програмних рішень, що полегшує розробку складних систем.

Найпопулярнішими PHP-фреймворками станом на 2018 рік вважаються Laravel, Symfony, Yii, CakePHP, Zend Framework, CodeIgniter.

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

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 .**

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

Composer - це пакетний менеджер рівня додатків для мови програмування PHP, який надає засоби з управління залежностями в PHP-додатку.

- 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.

Тема 9. Безпека створюваних web-застосунків. Файл .htaccess. MOD_REWRITE.

Безпека створюваних web-застосунків.

Безпека є головною проблемою для веб-програм. Навіть такі великі організації, як Організація Об'єднаних Націй, були зламані з використанням дуже простих недоліків безпеки [5].

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 інше.

Configuration

Бажано використовувати останні стабільні релізи PHP.

Переконайтеся, що ви зберегли свою операційну систему. Регулярно застосовуйте оновлення безпеки та стежте за новою інформацією про безпеку.

Errors and Warnings

Вам слід настроїти PHP, щоб приховати попередження та помилки під час їх виникнення. Помилки та попередження можуть дати хакеру ключ до внутрішніх функцій вашого коду, таких як назви каталогів та те, які бібліотеки ви використовуєте. Така інформація може допомогти йому використовувати вразливі місця у вашому стекі.

PHP as an Apache Module

Якщо PHP працює як модуль Apache, він буде запускатися з використанням того самого користувача, що і сервер Apache. Це означає, що він матиме ті самі дозволи та доступ, що і користувач Apache.

Краще налаштувати користувача для Apache, а не запустити його як "nobody". Користувач Apache повинен мати обмежений доступ до файлової системи, і не повинен перебувати в списку sudoers.

Ви повинні використовувати налаштування PHP open_basedir, щоб обмежити доступ до каталогів PHP. Ви можете протиставити його налаштуванню doc_root, що впливає на те, файли з яких каталогів PHP буде обслуговувати.

Session Security

Дві зони уваги, які вам слід знати, - це "викрадення сеансу" (session hijacking) та "фіксація сеансу" (“session fixation).

Викрадення сеансу

HTTP - це протокол без збереження стану, і веб-сервер, як очікується, одночасно обслуговуватиме кількох різних відвідувачів.

Сервер повинен мати можливість розрізняти клієнтів і робить це, призначаючи кожному клієнту ідентифікатор сеансу. Ідентифікатор сеансу можна отримати, викликаючи session_id(). Він створюється після виклику функції session_start().

Коли клієнт робить наступні запити на сервер, їм надають ідентифікатор сеансу, який дозволяє серверу пов'язувати запит із сеансом.

Клієнти можуть надавати сеанс як за допомогою файлів cookie, так і з параметром URL-адреси.

Файли cookie краще, але вони не завжди доступні. Якщо PHP не може використовувати файл cookie, він автоматично і прозоро використовуватиме URL-адресу, якщо ви не встановите параметр session.use_only_cookies у файлі php.ini.

Очевидним, що якщо ви зможете вказати на сервер чийсь ідентифікатор сеансу, то ви зможете маскуватися як цей користувач.

На Рис.43 показано сценарій, в якому злодійний користувач Боб здатний перехопити повідомлення Аліси на сервер. Bad Bob читає запит і витягує ідентифікатор сеансу (який міститься в заголовках файлів cookie для запиту HTTP). Тоді він може представити цей ідентифікатор сеансу на сервер, який тепер не може відрізнити його від Аліси.

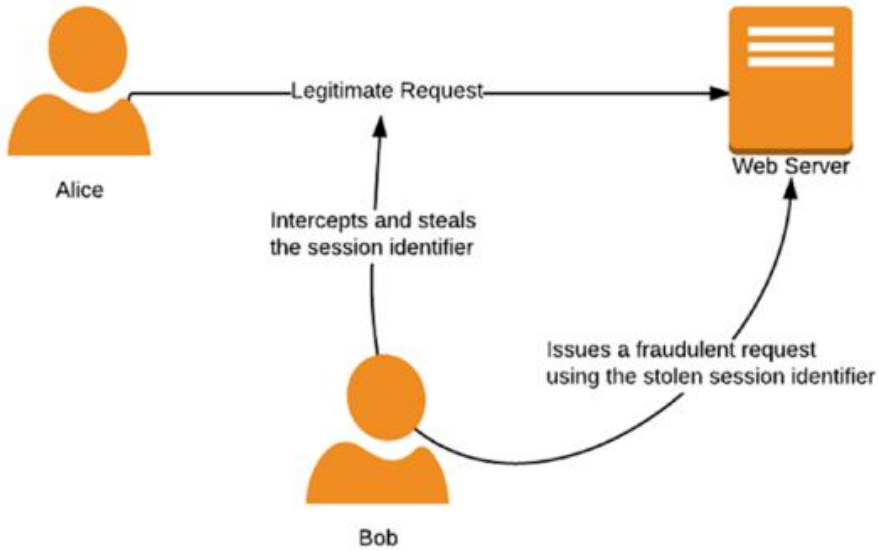


Рис.43. Викрадення сеансу

Отримання ідентифікатора сеансу іншого користувача може виконуватися кількома способами.

Якщо ідентифікатор сеансу відповідає передбачуваному шаблону, то зловмисник може спробувати визначити, що це буде для користувача. PHP використовує дуже випадковий спосіб генерування ідентифікатора сесії, тому вам не потрібно турбуватися про це.

Перевіряючи мережевий трафік між клієнтом і сервером, зловмисник може прочитати ідентифікатор сеансу. Ви можете встановити session.cookie_secure = On, щоб cookie-сесії були доступні лише через HTTPS, щоб пом'якшити це. HTTPS також буде шифрувати запитувану URL-адресу, і тому, якщо ідентифікатор сеансу буде переданий як параметр у запиті, він буде зашифрований.

Атаки проти клієнта, такі як атака XSS або троянець, запущені на комп'ютері, також можуть виявити ідентифікатор сеансу. Це може бути частково пом'якшене шляхом встановлення директиви session.cookie_httponly.

Session Fixation

Закріплення сесії використовує слабкість у веб-додатку. Деякі програми не генерують новий ідентифікатор сеансу для користувача під час автентифікації. Замість цього вони дозволяють використовувати існуючий ідентифікатор сеансу.

Напад відбувається, коли противник створює сеанс на веб-сервері. Він знає ідентифікатор сеансу для цього сеансу. Тоді він вводять в оману користувача, який використовує цей сеанс, і автентифікує себе. Після цього атакуючий може використовувати відомий ідентифікатор сеансу, і має права користувача автентифікації.

Існує декілька способів встановити ідентифікатор сеансу, і фактичний метод буде залежати від того, як програма приймає ідентифікатор.

Найпростіший спосіб зробити це було б передати ідентифікатор сесії в URL, як тут http://example.org/index.php?PHPSESSID=1234.

Найкращий спосіб зменшити ризик зафіксування сесії - це викликати функцію session_regenerate_id() кожного разу, коли змінюється рівень привілеїв, наприклад, після входу в систему.

Ви можете встановити session.use_strict_mode=On у вашому файлі конфігурації. Цей параметр змусить PHP використовувати тільки сайти, які ідентифікують сеанс. Він відхилить ідентифікатор сеансу, який буде надано користувачем. Це пом'якшить спроби маніпулювати файлом cookie.

Налаштування session.use_cookies=On та session.use_only_cookies=On перешкодить PHP приймати ідентифікатор сеансу з URL-адреси.

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).

Атака ефективна, оскільки клієнт вважає, що код походить з веб-сайту, якому він довіряє. Код може мати доступ до ідентифікаторів сеансу, файлів cookie, даних зберігання HTML та іншої інформації, пов'язаної з сайтом.

Існує декілька поширених типів атак XSS: stored, reflected та DOM.

У збереженій (stored) атаці XSS зловмисник може отримати вхід у місце збережігання на сервері. Прикладом можуть бути коментарі користувача, які ві-

дображаються на сайті і зберігаються в базі даних. Коли сайт виводить список коментарів користувачів іншому відвідувачеві, вони отримають шкідливий код.

У відображеній (reflected) атаці XSS зловмисник може скористатися вебсайтом, щоб вивести щось безпосередньо. Найпоширенішою формою цієї атаки є помилка заповнення форми, яка попередньо заповнює поля введення за допомогою раніше наданих полів або виводить помилкове значення поля. Надіславши відвідувача на створену URL-адресу, яка містить шкідливий код, як повідомлення про помилку (наприклад), зловмисник може змусити клієнта виконати його в контексті надійного сайту.

Атака DOM це така, яка цілком залежить від сторінки. Шкідливий код читається з елемента на сторінці, і викликає код, що виконується в самій сторінці.

Крім того, атаки XSS можна класифікувати як атаки на стороні сервера або на стороні клієнта. Атака на стороні сервера - це така, коли сам сервер видає шкідливий код. Клієнтська XSS виникає, коли ненадійні дані, надані користувачем, використовуються для оновлення DOM з небезпечним викликом JavaScript.

Пом'якшення нападів XSS

Найважливіше правило, яке слід дотримуватися - ніколи не надсилати неекрановані вихідні дані для клієнта. Завжди фільтруйте дані та видаляйте шкідливі теги, перш ніж надсилати їх клієнту.

Запам'ятайте цю мантру: “Filter input, escape output”.

Для цього є три корисні функції: htmlspecialchars(), htmlentities() і strip_tags().

Найбезпечнішим способом екранування виведення перед відображенням є використання filter_var ($ string, FILTER_SANITIZE_STRING).

CSRF-атака. (Cross Site Request Forgery)

Атака CSRF (“Підробка міжсайтових запитів”) експлуатує довіру, яку вебсайт має до клієнта. У цих атаках зловмисник нав'язує клієнту виконання команди на веб-сайті, який довіряє цьому клієнту.

Найбільш поширеною формою є відправлення запиту POST до форми введення.

Уявіть, що Аліса ввійшла на веб-сайт свого банку, який має форму, яка дозволяє їй переказати гроші на інший рахунок. Чак знає кінцеву точку цієї форми та ті поля вхідних даних, які вона має. Він якимось чином намагається змусити веб-браузер Аліси відправляти запит POST на цю форму, наказуючи банку переказати гроші на його рахунок. Банк довіряє веб-браузеру Alice, оскільки він має дійсний сеанс і виконує запит.

Існує багато способів, як Чак обманює веб-браузер Аліси, зокрема, використання фреймів і JavaScript.

Щоб пом'якшити ці запити, потрібно створити унікальний і дуже випадковий токен, який ви зберігаєте в сеансі Аліси. Коли ви видаєте форму, ви включаєте цей токен, так що, коли Аліса подає форму, вона також подає токен. Перш ніж обробляти форму, ви перевіряєте, чи поданий токен відповідає токену, що зберігається в її сеансі.

Чак не може дізнатися, який токен на сеансі Аліси, і тому не зможе його включити в його POST. Ваш код відхилить запит, яким він хоче обдурити Алісу, оскільки він не має дійсного токену.

Часто банки часто вимагають, щоб особа повторно автентифікувала під час виконання чутливої операції, і часто це вимагає двофакторної аутентифікації в рамках цього процесу.

SQL Injection

SQL-ін'єкція є найпоширенішою формою атаки в Інтернеті, і одна з найпростіших для захисту. SQL-ін'єкція відбувається, коли зловмисник може вставляти шкідливі команди в SQL-запит для виконання базою даних.

Багато установок баз даних дозволяють базі даних записувати файли на диск. Ця функція дозволяє хакерам створювати задні двері, використовуючи базу даних для запису скриптів PHP у каталозі, в якому веб-сервер обслуговує його.

Це означає, що ефект SQL ін'єкції не обмежується тим, що ваша база даних буде під загрозою, але може призвести до того, що зловмисник зможе виконувати довільний код у вашій базі даних.

Проблема з ін'єкцією SQL витікає з того факту, що в операторі SQL містяться дані та синтаксис. Забезпечуючи включення користувацьких даних в синтаксис функції, ми створюємо можливість того, що шкідливі дані можуть перешкоджати синтаксису.

Prepared Statements (підготовлені вирази)

Найефективнішим способом початку пом'якшення SQL-ін'єкції на мові PHP є використання виключно підготовлених виразів для взаємодії з вашою базою даних. Це допоможе виключити більшість атак на SQL-ін'єкцію, однак для цього не достатньо для надійності.

Підготовлені вирази настільки важливі, що драйвер PDO буде емулювати їх, якщо основний драйвер не підтримує їх.

Підготовлені вирази виконуються в три етапи:

1. Налаштуйте вираз з заповнювачами для даних.

2. Прив'яжіть фактичні дані до виразу.

3. Виконайте підготовлений вираз.

Можна пов'язати нові дані з виразом, який ви вже виконали, а потім запустити його знову за допомогою нового виразу. Двигун бази даних не повинен знову аналізувати SQL, що дає покращення продуктивності, крім переваг безпеки.

Цей код наводить приклад того, як підготувати, прив'язати та виконати вираз:

```
<?php
$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name = ?");
$stmt->bindParam(':name', $_GET['name'], PDO::PARAM_STR, 12);
$stmt->execute();
```

Зверніть увагу, що функція PDO::prepare() повертає об'єкт типу PDOStatement.

Ми використовуємо GET-змінну безпосередньо, тому нам не потрібно екранувати її, оскільки вона пов'язана як змінна з PDOStatement::bindParam() і не може змінити синтаксис SQL, який буде запускатися.

Інші драйвери баз даних на PHP також підтримують підготовлені вирази. Ось приклад з посібника для MySQL:

```
/* Prepared statement, stage 1: prepare */
if (!($stmt = $mysqli->prepare("INSERT INTO test(id) VALUES (?)"))) {
echo "Prepare failed: (" . $mysqli->errno . ") " . $mysqli->error;
}
/* Prepared statement, stage 2: bind and execute */
$id = 1;
if (!$stmt->bind_param("i", $id)) {
echo "Binding parameters failed: (" . $stmt->errno . ") " . $stmt-
>error;
}
if (!$stmt->execute()) {
echo "Execute failed: (" . $stmt->errno . ") " . $stmt->error;
}
```

Escaping (екранування)

Менш ефективним способом пом'якшення SQL-ін'єкції є уникнення спеціальних символів, перш ніж надсилати їх до бази даних. Цей спосіб більш схильний до помилок, ніж використання підготовлених виразів.

Якщо ви збираєтеся спробувати уникнути спеціальних символів, ви повинні використовувати певну функцію бази даних (наприклад, mysqli_real_escape_string()) або PDO::quote(), але не загальні функції, такі як addslashes().

Remote Code Injection

Впровадження віддаленого коду - це атака, коли зловмисник може отримати сервер для включення та виконання свого коду.

Functions That Evaluate Strings as Code

Деякі функції, такі як eval(), exec() та system(), чутливі до віддаленого використання коду. Якщо ви виконуєте змінну, яка включає користувацький ввід, вони зможуть вводити команди за допомогою escape символів.

Ви можете пом'якшити це, використовуючи escapeshellargs(), щоб виключити аргументи, передані до команди оболонки. Функція escapeshellcmd() сама екранує команди оболонки.

Функція assert() використовується для того, щоб переконатися, що певна умова є істинною і виконувати певні дії, якщо це не так. Це корисно для налагодження, але ви повинні вимкнути її для production. Ви можете використовувати функцію assert_options(), щоб налаштувати та вимикати assert().

Гра з include та require

Функції include() і require() дозволяють включати файли, вказані URL-адресою, якщо ввімкнено параметр PHP конфігурації allow_url_include.

Найпоширенішою подією є випадки, коли люди використовують змінну GET у URL-адресі, щоб визначити який-небудь динамічний вміст для включення. Це дуже аматорська помилка.

Наприклад, сайт може мати таку URL-адресу, як http://example.com/index.php?sidebar=welcome, а потім динамічно додавати файл welcome.php на бічну панель.

Зловмисник міг надати URL-адресу замість рядка "welcome" і мати свій власний код на сервері з тими ж рівнями привілеїв, що і користувач веб-сервера.

Щоб протистояти виникненню такої проблеми, ви можете вимкнути enable_url_fopen, використати basename() до вказаної вами змінної, щоб шляхи були включені лише з білого списку.

```
<?php
$page = $_GET['page'];
$allowedPages = array('adverts','contacts','information');
if ( in_array($page, $allowedPages) ) {
include basename($page . '.html');
}
```

Email Injection

Користувачам можливо ввести шістнадцяткові контрольні символи, які дозволяють їм змінювати текст повідомлення або список одержувачів.

Наприклад, якщо ваша форма дозволяє людині вводити свою електронну адресу як поле "від" для електронного листа, наступна рядок призведе до того, що додаткові одержувачі будуть включені як cc і сліпими одержувачами копії копії повідомлення:

sender@example.com%0ACc:target@email.com%0ABcc:anotherperson@emailexample.com,stranger@shouldhavefiltered.com

Зловмисник також може надати власне тіло і навіть змінити тип MIME відправленого повідомлення. Це означає, що ваша форма може бути використана спамерами для відправлення пошти.

Ви можете захиститись від цього декілька способів.

Переконайтеся, що ви правильно фільтруєте вхід, який ви використовуєте при надсиланні пошти. Функція filter_var() надає ряд прапорців, які ви можете використовувати, щоб переконатися, що ваші вхідні дані відповідають потрібному шаблону.

```
<?php
$from = $_POST["sender"];
$from = filter_var($from, FILTER_SANITIZE_EMAIL);
// send the email
```

Ви також можете встановити та використовувати Suhosin PHP розширення. Воно забезпечує директиву suhosin.mail.protect, яка буде захищати це.

Файл .HTACCESS (від англ. Hypertext access)

.HTACCESS – це файл додаткової конфігурації веб-сервера Apache.

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.

```
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)

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']

Приклад. Заборона завантаження файлів зображень з вашого сайту.

Треба 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.

Лекції 10-12. Мова JavaScript на боці клієнта і сервера та інші мови web-програмування

Тема 10. Технологія Node.js

Що таке Node.js ?

Node.js – платформа з відкритим кодом для написання серверної частини веб-застосунків на мові JavaScript.

Node.js характеризується такими властивостями:

- асинхронна однопотокова модель виконання запитів;
- неблокуюче введення/виведення;
- система модулів CommonJS (специфікація для модулів);
- движок JavaScript Google V8;

- для управління модулями використовується пакетний менеджер npm (node package manager).

Платформа Node.js призначена для створення додатків реального часу, що оброблюють великі обсяги даних (DIRTy-додатків).

Автор технології – Ryan Dahl (2009).

По суті не придумав нічого нового:

“node.js is a set of bindings to the V8 JavaScript VM” (Ryan Dahl)

- V8 – Google JavaScript Engine, що використовується в браузері Chrome.
- Краща технологія 2012 року по версії InfoWorld.
- Сайт nodejs.org

Ryan Dahl запропонував новий спосіб мислення про те, як треба будувати програмне забезпечення web-системи.

Головні ідеї:

- Виконання серверного коду в одному потоці.
- Асинхронне виконання операцій введення-виведення.
- Серверний JavaScript.

Це було і до Node.js, але саме Ryan Dahl все це об’єднав при розробці Node.js.

Неблоковане введення/виведення

Traditional I/O

```
var result = db.query(“select x from table_Y”);
doSomethingWith(result); // wait for result!
doSomethingWithOutResult(); // execution is bloked!
```

Non-bloking I/O

```
var result = db.query(“select x from table_Y”, function(result){
     doSomethingWith(result); // wait for result!
});
doSomethingWithOutResult (); // executes without any delay!
```

Платформа чи середовище?

- Node - це платформа, а не середовище розробки (framework) JavaScript-додатків.
- Поширеною помилкою є ототожнення Node з середовищами типу Rails або Django. Насправді Node є набагато більш низькорівневим інструментом.

Одне з популярних середовищ Node-розробки – це Express (http://expressjs.com/)

Ключові частини Node.js

Архітектура Node.js показана на Рис. 44.

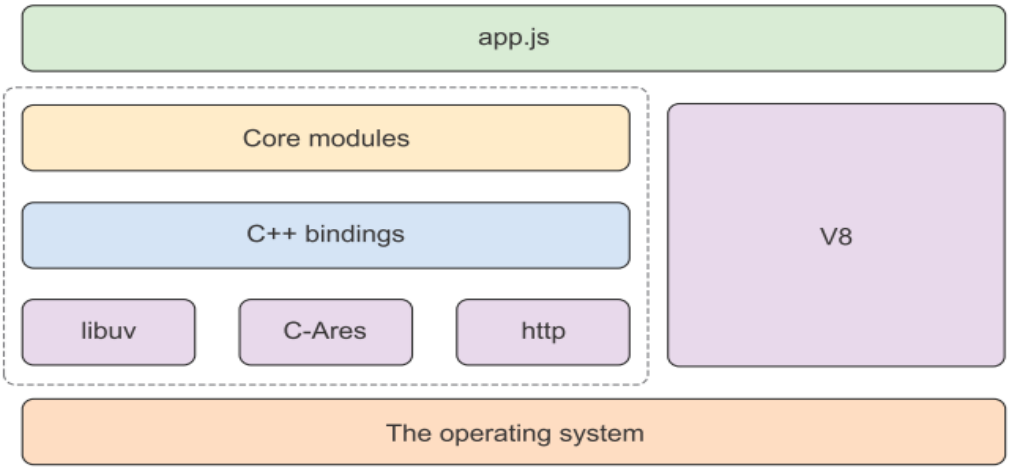


Рис.44. Ключові частини Node.js

libuv – binary library, which provides a fast run loop and non-blocking I/O for networking and the file system.

http – binary library for HTTP .

Трошки практики

Багато речей можна реалізувати на Node.js в декілька рядків коду. Реалізація HTTP-сервера:

```
var http = require ( 'http' );
http.CreateServer( function (req, res){
     res.writeHead( 200, { 'Content-Type': 'text/plain'});
     res.End('Hello World\n');
}).listen(process.env.PORT, process.env.IP);
console.log('Server running!');
```

Проблеми архітектур більшості web застосунків

Робота з базою даних:

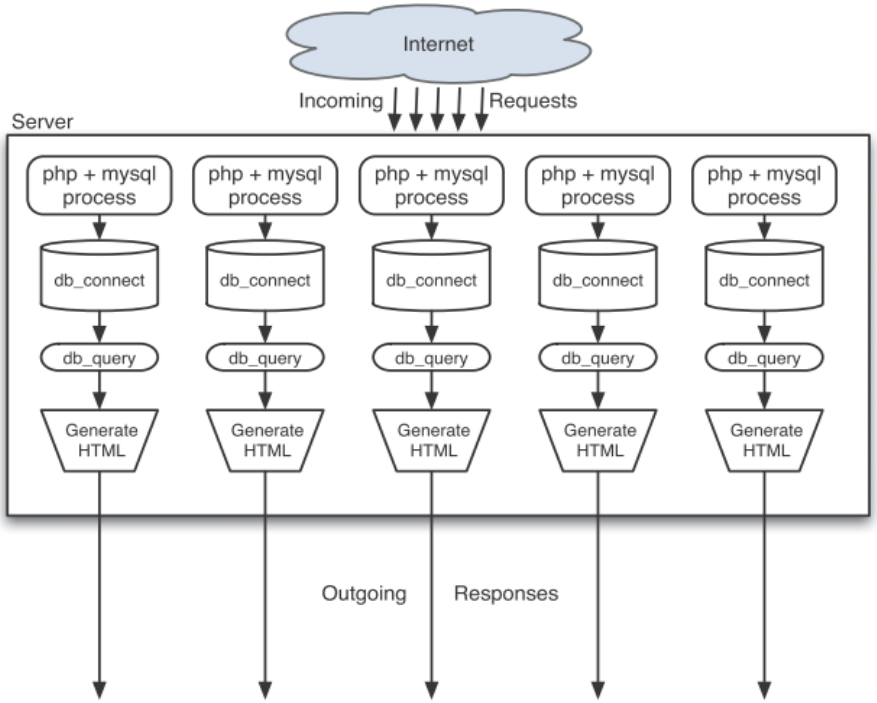
```
result = query('select * from T')
```

Що відбувається під час виконання цього коду?

Створюється окремий поток.

Серверний код чекає виконання операції.

Якщо запитів багато (Рис.45), то сервер може бути заблокованим.



.1 Traditional blocking IO web servers

Рис.45. Традиційний блокований Input/Output на сервері

Затрати на операції в сучасних комп'ютерах:
L1: 3 такти, L2: 14 тактів, RAM: 250 тактів,
Диск: 41 000 000, Мережа: 240 000 000 тактів.

Виділення потоку на обробку запиту клієнта
Перемикання між контекстами потоків затратне.
При масивній конкуренції ви не можете виділяти поток для кожного запиту. Таким чином всі запити, що надходять, будуть чекати.

Як працює Node.js в якості сервера?

Обробка запитів в Node.js ілюструют Рис.46 та Рис.47.

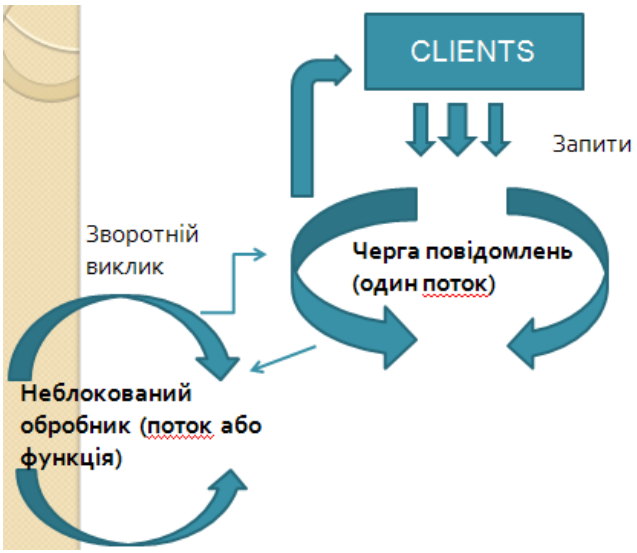


Рис.46. Обробка запитів в Node.js

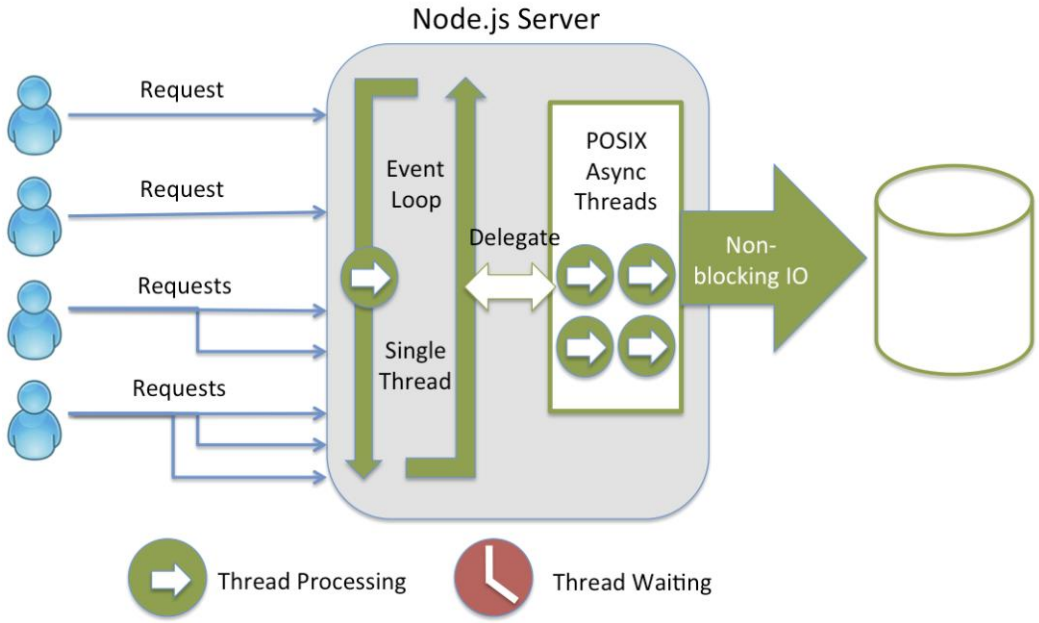


Рис.47. Обробка запитів на сервері Node.js

Порівняльна оцінка швидкодії Apache- та NGINX-серверів (http://mng.bz/eaZT сайт WebFaction) показана на Рис.48.

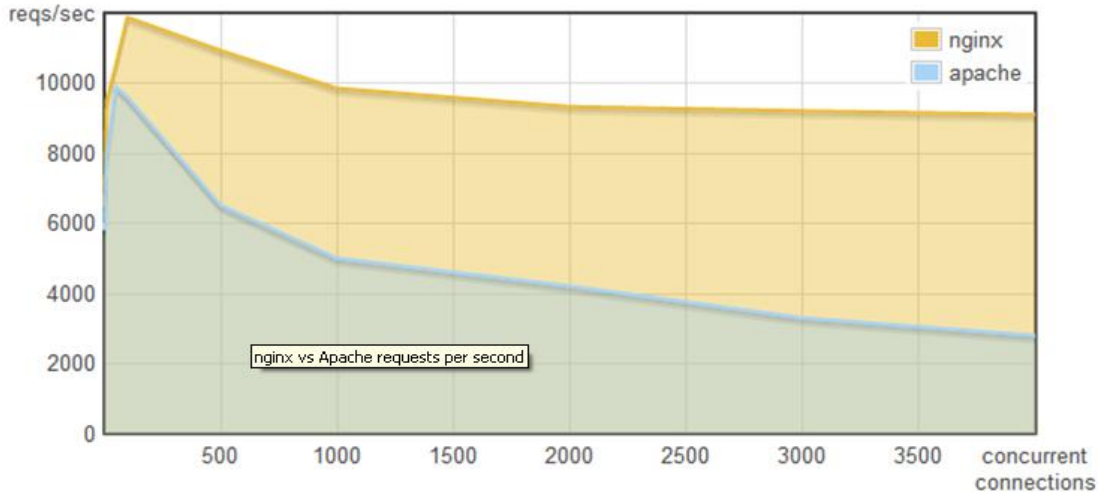


Рис.48. Швидкодії Apache- та NGINX-серверів

По осі X – кількість з'єднань, по осі Y – кількість запитів, що обробляються за секунду.

Перевага NGINX (як і Node.js) в тому, що замість багатопоточного підходу використовується асинхронність.

DIRTy-застосунки

Назва DIRTy походить від абревіатури DIRT (data-intensive real-time), що відноситься до застосунків реального часу.

Хорошим прикладом створеного в Node DIRTy-застосунку є Browserling (browserling.com). Ця програма дозволяє у вікні одного браузера відкривати і використовувати інші браузери.

Додатковий проект Testling (testling.com).

Що ж там всередині? Модулі.

Розробка в Node.js основана на модулях (модулі – це JavaScript-файли).

Саме модулі дозволяють застосунку спілкуватися з зовнішнім світом та структурувати ваш код.

Існують три джерела модулів:

- Вбудовані в Node (core modules).
- Ваші файли (власні модулі).
- Глобальний репозиторій.

Вбудовані модулі (core modules)

Забезпечують серверну інфраструктуру.

Модуль підключається так:

```
var fs = require ( 'fs' );
```

Приклади модулів:

fs (http://nodejs.org/api/fs.html) - file system module.

http (http://nodejs.org/api/http.html) – HTTP.

crypto (http://nodejs.org/api/crypto.html).

os - операційна система.

Глобальний репозиторій

Це екосистема Node Package Manager (NPM).

https://www.npmjs.org > 400 000 модулів.

Версії Node.js

Версії Node.js показані на Рис.49 (https://github.com/nodejs/Release).

Release	Status	Codename	Initial Release	Active LTS Start	Maintenance LTS Start	End-of-life
v0.10.x	**End-of-Life**	-	2013-03-11	-	2015-10-01	2016-10-31
v0.12.x	**End-of-Life**	-	2015-02-06	-	2016-04-01	2016-12-31
4.x	**Maintenance LTS**	Argon	2015-09-08	2015-10-01	2017-04-01	2018-04-30
5.x	**End-of-Life**		2015-10-29			2016-06-30
6.x	**Active LTS**	Boron	2016-04-26	2016-10-18	2018-04-30	April 2019
7.x	**End-of-Life**		2016-10-25			2017-06-30
8.x	**Active LTS**	Carbon	2017-05-30	2017-10-31	April 2019	December 2019[1]
9.x	**Current Release**		2017-10-01			June 2018
10.x	**Pending**	Pending	Apr 2018	October 2018	April 2020	April 2021

Рис.49. Версії Node.js

Огляд версій

Node.js v4.0.0 – Aug 2015. Движок v8 v4.5

Підтримка ES6 стандарту:

- Блочна область видимості
- Класи
- Типізовані масиви
- Генератори
- Promises
- Symbols
- Рядкові шаблони
- Колекції (Map, Set, і так далі)

Node.js v6.0.0 – Apr 2016. Движок v8 v5.0

- Зміни в API
- Поліпшили підтримку EcmaScript 6
- Конструктор об'єкта Buffer змінив свою поведінку
- Об'єкт EventEmitter отримав два нових методи prependListener і prependOnceListener
- Методи fs.realpath і fs.realpathSync тепер використовує оновлену логіку libuv
- Відмова від підтримки WinXP і Vista
- Поліпшили продуктивність, надійність і безпеку системи

Установка Node

http://nodejs.org/#download

На березень 2018:
Downloads: Latest LTS Version: 8.9.4 (includes npm 5.6.0).

Після завантаження натисніть INSTALL.

Після завершення установки ви отримаєте можливість запускати Node і npm з командного рядка Windows (режим REPL – Read-Eval-Print-Loop).

Відкрийте термінал (cmd.exe), наберіть "node" і натисніть Enter.

Робота в режимі терміналу показана на Рис.50.

Рис.50. Робота з Node.js в режимі терміналу

Багато прикладів для Node запускають через термінал.

Можна обійтися стандартним терміналом (cmd.exe).

Можна використати Far Manager 3.0.

Додаткові переваги можна отримати, використовуючи ConEmu (http://bit.ly/Con-Emu/) або Console2 (http://bit.ly/Console_2/).

Запуск скриптів

Приклад 1. Файл hello.js : console.log(“Hello, World!”);
Запускаємо: node hello.js

Приклад 2. HTTP-сервер. Файл server.js

```
var http = require("http");
http.createServer(function(request, response) {
   response.writeHead(200, {"Content-Type": "text/plain"});
   response.write("Hello World");
   response.end();
 }).listen(8888);
```

Запускаємо сервер: node server.js
Запускаємо браузер, http://localhost:8888/
Отримаємо : Hello World

Реалізація потоків даних

Платформа Node підходить для програмування потоків даних та організації їх передачі. Завдяки передачі даних фрагмент за фрагментом розробник отримує можливість обробляти дані по мірі їх накопичення, а не чекати, поки будуть передані всі дані, а потім виконувати які-небудь дії.

```
var stream = fs.createReadStream('./resource.json')
stream.on('data', function (chunk) {
console.log(chunk)
})
stream.on('end', function () {
console.log('finished')
})
```

Подія data викликається після появи нового фрагмента даних, а подія end - після завантаження всіх фрагментів коду.

Записувані потоки даних

У Node також підтримуються записувані потоки даних, що дозволяють записувати фрагменти даних.

Один з подібних потоків - об'єкт відповіді (res), генерований у відповідь на запит до HTTP-сервера.

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
res.writeHead(200, {'Content-Type': 'image/png'});
     // Передача по каналах з зчитуваного потоку в записуваний
fs.createReadStream('./image.png').pipe(res);
}).listen(3000);
console.log('Server running at http://localhost:3000/');
```

Розробка власних модулів

Модуль logger.js:

```
exports.info = function ( msg ) {
     console.log( new Date() + ': ' + msg);
}
exports.error = function ( msg) {
     console.error( msg);
}
```

Головний файл app.js:

```
var  logger = require( './logger');
logger.info( 'Hello world!');
```

Запуск скрипта:

```
     node  app.js
```

Модуль Point.js

```
     function Point ( x, y) {
          this.x = x;
          this.y = y;
     }
     Point.prototype.print = function () {
          console.log( this.x + ', ' + this.y);
     }
     module.exports = Point;
```

Головний файл app.js

```
var Point = require( './Point');
var  pt = new Point( 50, 60);
pt.print();
```

Читання файлу в асинхронному режимі

```
var http = require('http'); // Файл server_readFile.js
var fs = require('fs');
http.createServer(function(req, res){
     fs.readFile('hello.js', 'utf8', function(err, data){
          res.writeHead(200, {'Content-Type': 'text/plain'});
          if(err)
     res.write('Could not find or open file for reading\n');
          else
               res.write(data);
          res.end();
     });
}).listen(8124,   function(){   console.log('bound   to   port
8124');});
console.log('Server running on 8124/');
```

Маршрутизація (routing)

```
var http = require('http');
http.createServer(function(req,res){
var path = req.url.replace(/\/?(?:\?.*)?$/, '').toLowerCase();
switch(path) {
     case '':
          res.writeHead(200, { 'Content-Type': 'text/plain' });
          res.end('Homepage');
          break;
```

```
        case '/about':
            res.writeHead(200, { 'Content-Type': 'text/plain' });
            res.end('About');
            break;
    default:
        res.writeHead(404, { 'Content-Type': 'text/plain' });
        res.end('Not Found');
        break;
    }
    }).listen(3000);
    console.log('Server started on localhost:3000; press Ctrl-C to ter-
minate....');
```

Спробуйте вказати URL:
http://localhost:3000
http://localhost:3000/about
http://localhost:3000/foo

Відображення статичних ресурсів

Нехай в теці public знаходяться файли home.html, about.html, notfound.html та в теці public/img файл logo.jpg. Розглянемо, як сервер Node може їх відобразити.

```
    // Файл server_StaticResources.js
    var http = require('http'),
    fs = require('fs');
    function serveStaticFile(res, path, contentType, responseCode)
{
        if(!responseCode) responseCode = 200;
        fs.readFile(__dirname + path, function(err,data) {
            if(err) {
                res.writeHead(500, { 'Content-Type': 'text/plain'
});
                res.end('500 - Internal Error');
            } else {
    res.writeHead(responseCode,
                    { 'Content-Type': contentType });
                res.end(data);
            }
        });
    }
    http.createServer(function(req,res){
            var path = req.url.replace(/\/?(?:\?.*)?$/, '')
                .toLowerCase();
        switch(path) {
            case '':
                serveStaticFile(res, '/public/home.html',
        'text/html');
                break;
    case '/about':
                serveStaticFile(res, '/public/about.html',
            'text/html');
                break;
```

```
                case '/img/logo.jpg':
                     serveStaticFile(res, '/public/img/logo.jpg',
                     'image/jpeg');
                     break;
                default:
                     serveStaticFile(res, '/public/404.html',
'text/html',                         404);
                     break;
          }
     }).listen(3000);
     console.log('Server started on localhost:3000; press Ctrl-C to
terminate....');
```

Використання менеджера пакетів NPN

Приклад. Відправка SMS через сервіс Twilio

```
npn  install  twilio
```

Файл app.js:

```
     var  twilio = require('twilio');
     var  client = twilio();
     client.sendMessage({
          to: '+37……..',
          from: '+37…..',
          body: 'Hello world'
     });
```

Запуск:

```
node  app.js
```

Примітка. Треба мати account на www.twilio.com

Файл package.json

- Є невід'ємною частиною NPM екосистеми і має особливе значення для NPM.
- Дуже важливо, щоб він був правильно налаштований, коли ви хочете поділитися своїм модулем зі світом, або якщо ви споживаєте модулі від інших людей.
- Щоб створити файл package.json в поточній папці, просто запустіть код: npm init
- Буде задано декілька питань (ім'я модуля, версія, автор, . . .)

Ви отримаєте файл package.json на зразок такого:

```
{
     "name": "ExpressCourseDemo",
     "version": "0.0.0",
     "description": "This demo app for the Express course",
     "author": "Carlos Souza  <carloshrosouza@gmail.com.com>",
     "dependencies": {
          "body-parser": "^1.8.1",
          "debug": "^0.8.1",
          "express": "^4.9.5",
          "lodash": "^2.4.1"
          },
```

```
    "scripts": {    "start": "node app.js"  }
}
```

Saving Dependencies

Коли ви встановлюєте якийсь модуль, тобто запускаєте npm install, у вас є додатковий прапорець командного рядка (--save), що говорить NPM записати в файл package.json інформацію про те, що ви встановили, як показано в лістингу:

```
npm install underscore --save
```

В файлі package.json з'явиться запис

```
"dependencies": {
    "underscore": "^1.6.0"
  }
```

Refresh the node_modules Folder

Усі додаткові модулі, які ви додаєте до вашого застосунку з допомогою npm, попадають в каталог node_modules.

Щоб оновити папку node_modules з вашого файлу package.json, ви можете запустити таку команду:

```
npm install
```

Висновок. Ще одна перевага використання package.json є те, що ви можете не зберігати папку node_modules, так як ви завжди можете отримати її копію з npmjs.org простою командою npm install

Створення нового Node-проекту

- Створити каталог проекту.
- Перейти в нього.
- Запустити npm init для створення файлу package.json (файл маніфесту проекта).
- За допомогою npm підключити до проекту необхідні модулі.
- Створити головний JavaScript файл app.js (та, можливо, інші файли).
- Виконати проект:
  node app.js

Вправа 1.

- Створити каталоги NPM1, NPM2 та зайти в NPM1
- npm install underscore
- Створити файл app.js

```
var arreg = require('underscore');
arreg.each(
     [1,2,3,4,5], function(num){
          console.log("Number : "+num);
     }
     );
```

- Запустити застосунок: node app.js.

- Зайти в каталог NPM2.
- npm init (для створення файлу package.json).
- На запит name та description ввести npm2, на інші запити натискати Enter.
- На останній запит (Is this ok?) відповісти yes.
- Переглянути файл package.json.
- npm install underscore –save
- Переглянути файл package.json, впевнившись, що в ньому з'явилось підключення бібліотеки underscore.
- Скопіюємо файл app.js з NPM1 в NPM2.
- Відкриємо в редакторі файл package.json та додамо в секцію "dependencies" рядок "express": "^4.0".
- Подамо команду npm install (в проект буде доданий фреймворк express.js).
- npm install body-parser –save
- Впевнюємося, що до проекту підключився модуль body-parser, призначений для розбору запиту POST.

Вправа 2

- Створіть каталог ROUTE_EXPRESS, а в ньому – каталог BASIC і війдіть в нього.
- npm init .
- На запит name ввести server, на запит entry point ввести server.js, на інші запити натискати Enter а в кінці – y.
- npm install express –save
- Створіть каталог public в каталозі BASIC.
- Створіть в каталозі BASIC файл server.js з наступним вмістом:

```
var express = require('express');
    var app = express();
    app.get(
               '/', function(req,res){
              res.sendFile(__dirname + '/public/index.html');
               }
         )
    app.listen(9000);
    console.log("Server initialised…");
```

- Створіть в каталозі public файл index.html з наступним вмістом:
  <h1> This page is testing EXPRESS</h1>
- Повертаємось в каталог BASIC та запускаємо сервер: node server.js
- В браузері набираємо адресу localhost:9000/ (повинна активізуватися сторінка index.html).
- Створюємо в каталозі public файл contacts.html з наступним вмістом:
  <h1> Contacts in Node.js</h1>
- Редагуємо файл server.js, додавши після app.get(. . .) крапку з комою (;) та додавши наступне:

```
    app.get(
                    '/contacts', function(req,res){
                   res.sendFile(__dirname +
'/public/contacts.html');
                    }
              )
```

- Запускаємо сервер: node server.js
- В браузері набираємо адресу localhost:9000/contacts (повинна активізуватися сторінка contacts.html).
- В каталозі ROUTE_EXPRESS створюємо каталог FULL та переходимо в нього.
- npm init
- На запит name ввести full, на інші запити натискати Enter а в кінці – y (буде сформовано файл package.json).
- npm install express --save
- npm install body-parser --save
- Створіть каталог public в каталозі FULL.
- Створюємо в каталозі public файл index.html з наступним вмістом:

```
<form name="form1" action="/login" method="post">
<label>User:</label><input type="text" name="user"
id="user">
<br>
 <label>Password:</label><input type="password"
name="password" id="password">
<br> ><input type="submit">
</form>
```

- Створіть в каталозі FULL файл server.js з наступним вмістом:

```
var express = require('express');
var bodyParser= require('body-parser');
var app = express();
app.use(bodyParser.urlencoded({extended : false}));
app.get(
            '/', function(req,res){
           res.sendFile(__dirname + '/public/index.html');
            }
     )
app.post(
            '/login', function(req,res){
           var user_name=req.body.user;
           var password=req.body.password;
           console.log("The user's name is:  "+user_name+"
And password is: "+password);
           res.end("ending");
            }
     )
app.listen(9000);
console.log("Server initialised…");
```

- Запускаємо сервер: node server.js (з каталогу FULL).

- В браузері набираємо адресу localhost:9000/ (повинна активізуватися сторінка index.html з формою).
- Вводимо логін, пароль і натискаємо Submit.

В браузері з'явиться "ending", а, повернувшись до консолі, бачимо інформацію про введений логін і пароль.

Автоматичний рестарт

При внесенні змін в ваш проект часто повторюються команди зупинки сервера та його повторного запуску. Це можна автоматизувати за допомогою пакета nodemon. Інсталюйте його глобально:

```
npm install -g nodemon
```

Застосовуйте nodemon для запуску вашого сервера:

```
nodemon server.js
```

Тема 11. Огляд фреймворків для Node. Основи Express.js

Фреймворки призначені для створення каркасу застосунку, прискорення розробки. Для Node існують декілька фреймворків (Express.js, Hapi.js, Koa.js, Meteor.js, Total.js, Compound.js, Geddy.js, Sails.js, Adonis.js), самим популярним з яких є Express.js.

Основи Express

Що таке Express.js?

- Express.js – це веб-фреймворк на базі модуля HTTP ядра Node.js (і компонентів фреймворку Connect). Express – був надбудовою над Connect.
- Компоненти Connect називають middleware.
- Вони є наріжним каменем філософії фреймворку, який є "конфігурація важливіше конвенції" (configuration over convention).
- Іншими словами, розробники можуть вільно вибрати ті бібліотеки , що ім потрібні для конкретного проекту ("нічого зайвого").
- Такий підхід забезпечує гнучкість і здатність тонко налаштовувати свої проекти (компонентний підхід замість монолітного).
- Express 4.0 (2014) вже не залежить від Connect.

Express.js бере на себе:

- Розбір тіла запиту HTTP.
- Розбір cookies.
- Управління сеансами.
- Організація маршрутизації запитів (зв'язки URL з методами HTTP).
- Визначення правильних заголовків відповіді, засноване на типах даних.

- Обробка помилок.
- Витяг параметрів URL.

Express.js :

- Використовує структури Node.js.
- Використовує шаблонізатори Jade/Pug, EJS , Mustache, Handlebars.
- Доступ до шаблону MVC
  - Model - - the Data
  - View - - the Template
  - Controller - - the JavaScript
- Створює (генерує) структуру каталогів проекту завдяки Scaffolding.

Інсталяція Express.js

Інсталяція останньої стабільної версії:

```
npm install express
```

Використовуйте @ для інсталяції специфічної версії:

```
npm install express@3.3.5
```

Формується каталог
node_modules / express

Інсталяція глобально: `npm install -g express`
Інсталяція в даний проект: `npm install --save express`
(якщо є готовий файл package.json)

"Hello World" на Express

Файл app.js:

```
var express = require("express");
var app = express();
app.get("/", function(request, response) {
        response.send("Hello, world!");
});
app.listen(3000, function() {
   console.log("Express app started on port 3000.");
});
```

Запускаємо з консолі: `node app.js`
Запускаємо браузер: http://localhost:3000/
Отримаємо: Hello, world!

Що таке cURL?

- Це утиліта командного рядка, що дозволяю взаємодіяти з різними видами серверів за різними протоколами з синтаксисом URL.
- cURL надає можливість гнучкого формування запиту із заданням таких параметрів, як cookie, user_agent, referrer і будь-яких інших заголовків.
- За допомогою cURL можна, наприклад, отримати вміст html-сторінки з командного рядка, не використовуючи для цього браузер:

- curl http://localhost:3000/
- Отримаємо (для попереднього прикладу): “Hello, world!”

- Сайт: http://curl.haxx.se/download.html

Відповідь у вигляді JSON

Функція send () конвертує Objects та Arrays в JSON.

```
Сервер (фрагмент):
app.get("/blocks", function(request, response) {
     var blocks = ['Fixed','Movable','Rotating'];
        response.send(blocks);
});
```

Запускаємо з консолі: `node app.js`

```
З іншої консолі:
$ curl –i http://localhost:3000/blocks
Отримаємо:
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8

["Fixed","Movable","Rotating"]
```

Ми тут використали прапорець `–i` щоб вивести заголовки відповіді.

Відповідь у вигляді HTML

Функція send () передає рядки у вигляді HTML.

```
Сервер (фрагмент):
app.get("/blocks", function(request, response) {
     var blocks = '<ul><li>Fixed</li><li>Movable</li></ul>';
        response.send(blocks);
});
```

Запускаємо з консолі: `node app.js`

```
З іншої консолі:
$ curl –i http://localhost:3000/blocks
Отримаємо:
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/html; charset=utf-8

<ul><li>Fixed</li><li>Movable</li></ul>
```

Перенаправлення на відносний шлях

```
Сервер (фрагмент):
app.get("/blocks", function(request, response) {
        response.redirect('/parts');
});
```

Запускаємо з консолі: `node app.js`

```
З іншої консолі:
$ curl –i http://localhost:3000/blocks
Отримаємо:
HTTP/1.1 302 Moved Temporarity
```

```
X-Powered-By: Express
Location: /parts
Content-Type: text/plain; charset=utf-8

Moved Temporarity. Redirecting to /parts
```

Приклад. Відсилання файлів методом sendFile().

Створимо два файли і директорію як показано на Рис.51.

Рис.51. Фрагмент файлової системи

//Файл index.html

```
<DOCTYPE html>
<html lang=”en”>
<head>
    <meta charset=”UTF-8”>
    <title>Building Blocks</title>
</head>
<body>
    <h1>Blocks</h1>
</body>
</html>
```

//Файл app.js

```
var express = require(‘express’);
var app = express();

app.get(‘/’, function(request, response){
    response.sendFile(__dirname + ‘/public/index.html’);
});

app.listen(3000);
```

Запускаємо з консолі: `node app.js`
Запускаємо браузер: `http://localhost:3000/`
Отримаємо результат, показаний на Рис.52.

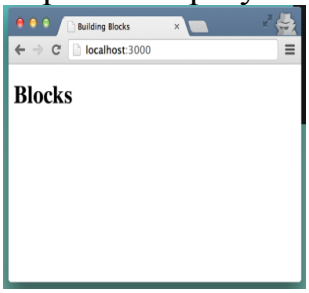


Рис.52. Результат запуску додатку

Middleware (програмне забезпечення проміжної ланки)

Зазвичай, терміном middleware позначали програмне забезпечення (на боці сервера), що об'єднує разом шар бізнес-логіки з сервером бази даних.

В нашому випадку для Node.js під терміном middleware слід розуміти колекцію модулів, які дозволяють інтегрувати в наш застосунок якусь функціональність на боці сервера або клієнта.

У випадку з Express до цієї категорії відноситься частина ланцюжка додатків, кожен з яких виконує певну функцію, пов'язану із запитом HTTP, - обробляє його чи обробляє запит для наступних проміжних додатків. Список програмного забезпечення проміжної ланки, що працює з Express, досить великий (http://expressjs.com/en/resources/middleware.html).

Стек обробки запитів в «чистому» Node.js та Express показаний на Рис.53 та Рис.54

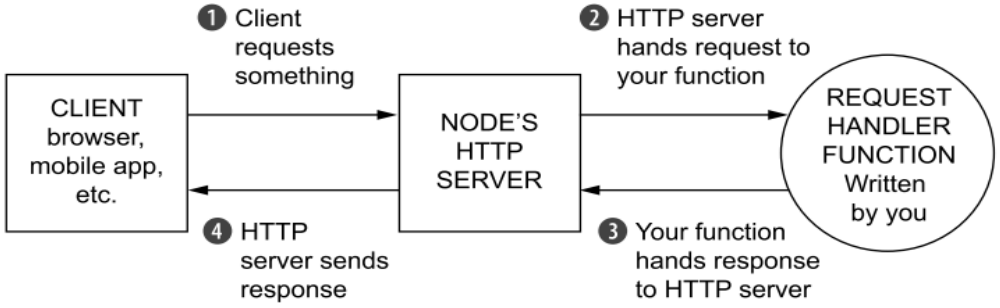


Рис.53. Потік обробки запитів в Node.js

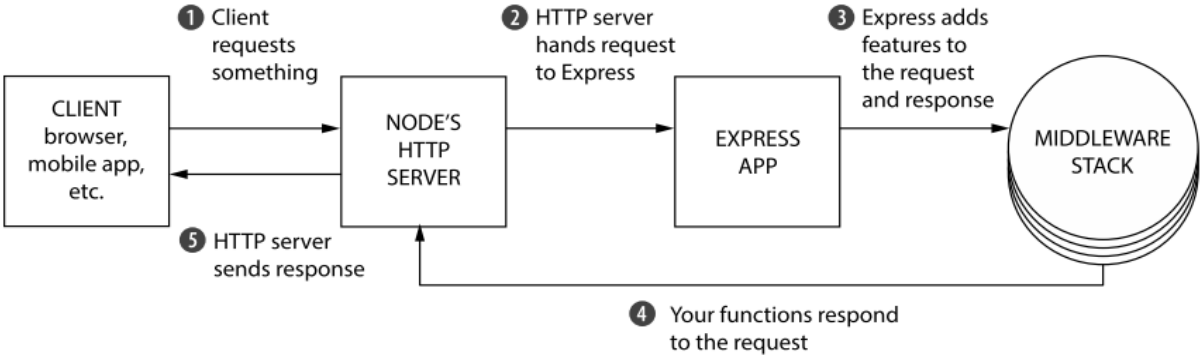


Рис.54. Потік обробки запитів в Express

Монтування middleware

Метод app.use додає middleware до стека застосунку.

Приклад. Монтування static middleware.

Для надання статичних файлів, наприклад, зображень, файлів CSS і JavaScript в Express використовується middleware express.static. Для того щоб почати безпосереднє надання файлів, необхідно передати ім'я каталогу, в якому знаходяться статичні ресурси, в middleware express.static.

Модифікуємо трохи наш попередній приклад.

//Файл app.js

```
var express = require('express');
var app = express();
```

```
app.use(express.static('public'));
app.listen(3000);
```

Тепер запустимо застосунок і побачимо, що результат буде точно таким, як на Рис.50.

Зверніть увагу, що static middleware обслуговую все, що знаходиться в папці public. Наприклад, якщо в папку public додати файл blocks.jpg і в файл index.html після 8-го рядка додати рядок <img src='blocks.jpg'> і перезапустити застосунок, то ми матимемо результат, показаний на Рис.55.

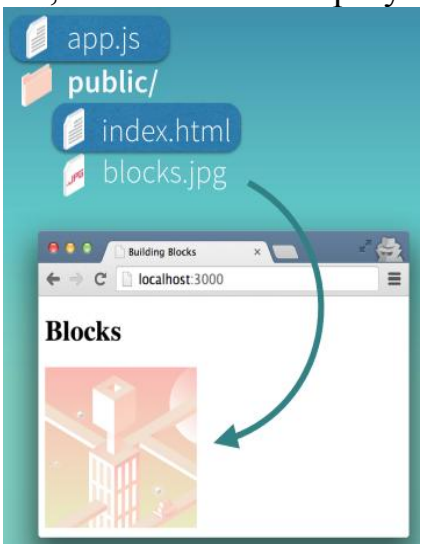


Рис.55. Модифікована сторінка прикладу

Маршрутизація в Express

Маршрутизація визначає, як додаток відповідає на клієнтський запит до конкретної адреси, і певного методу HTTP запиту.

Кожен маршрут може мати кілька обробників, які виконуються при співставленні маршруту.

Для визначення маршруту слід використати наступну структуру:

```
app.METHOD(path, callback)
```
, де

app – екземпляр Express;
METHOD – метод HTTP запиту (GET, POST і т.д.);
path – шлях на сервері;
callback – функція-обробник на указаний шлях.

Генератор структури застосунків Express

Можна використовувати інструмент express-generator, для швидкого створення каркасу застосунку.

Встановлюється express-generator наступною командою:

```
npm install express-generator -g
```

З параметром -h можна проглянути доступні опції:

```
express -h
```

В наступному прикладі створюється каркас застосунку Express з іменем myapp в поточній директорії:

```
express –view=pug myapp
```

Після чого треба встановити залежності:

```
cd myapp
npm install
```

На Windows запустіть застосунок такою командою:

```
set DEBUG=myapp:* & npm start
```

Згенерований застосунок має наступну структуру директорій (Рис.56):

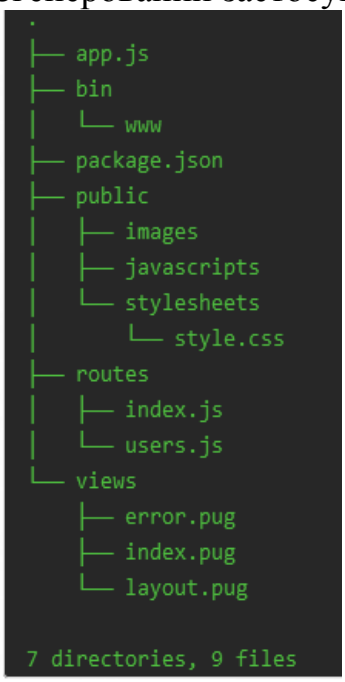


Рис.56. Згенерована структура каталогів застосунку

Модуль body-parser

Модуль, ймовірно, є найбільш важливим з усіх third-party middleware модулів. Він дозволяє розробникам обробляти вхідні дані, такі як вміст тіла та заголовки HTTP-запиту, транслюючі їх в сукупність об'єктів JavaScript/Node.js.

Інсталяція модуля:

```
npm install body-parser@1.8.1
```

Головні сервіси модуля body-parser:

- json() : Processes JSON data; e.g., {"name": "value", "name2": "value"};
- urlencoded() : Processes URL-encoded data; e.g., name=value&name2=value2;
- raw() : Returns body as a buffer type;
- text() : Returns body as string type.

Движки шаблонів (Template engines)

Це бібліотеки, які дозволяють нам використовувати різні мови шаблонів (EJS, Handlebars, Pug/Jade, Mustache) для представлення даних в Express.js.

Мови шаблонів мають спеціальні інструкції для движка, що показують як обробляти дані.

Процес об'єднання даних з шаблонами називається rendering (візуалізація)

Використання Jade/Pug

Jade перейменували в Pug через порушення торгового знаку. Систаксис трохи відрізняється.

Сайти: http://jade-lang.com/ та https://pugjs.org/

Інсталяція:
```
npm install jade --save
npm install pug --save
```

Монтувати движок шаблонів до застосунку необхідно в методі app.set, вказавши директорію та ім'я рушія, наприклад:

```
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
```

При обробці маршруту на сторінку index.pug можна передати параметри, наприклад:

```
res.render('index', {title: 'Express'});
```

Синтаксис Jade/Pug покажемо на прикладі.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Pug</title>
</head>
<body>
<h1>Pug Examples</h1>
<div class="container">
<p>Cool Pug example!</p>
</div>
</body>
</html>
```

Цей файл на Pug буде виглядати лаконічніше:

```
doctype html
html(lang='en')
  head
     title Pug
  body
     h1 Pug Examples
    .container
        p Cool Pug example!
```

Існує сайт (http://html2jade.org/) для конвертації html в jade.

Тема 12. Мова Python в web-програмуванні. Фреймворк Django.

Мова програмування Python

Python – інтерпретована, мультипарадигменна мова програмування високого рівня з динамічною типізацією, автоматичним управлінням пам'яттю і зручними високорівневими структурами даних. Розроблена в 1991 році Гвідо ван Россумом. Підтримує обробку винятків, паралельні обчислення. Підтримує декілька парадигм програмування, зокрема: об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану.

Імперативна - описує процес обчислення у вигляді інструкцій, що змінюють стан програми. Об'єктно-орієнтована - основною концепцією є поняття об'єкта, що ототожнюється з об'єктом предметної області. Функціональна - в ній процес обчислення трактується як обчислення значень функцій в математичному розумінні (спосіб вирішення задачі описується у вигляді залежності функцій одна від одної).

Динамічна типізація: немає попереднього оголошення типів – тип змінної виводиться в процесі виконання.

Застосування Python

- Інтерактивна консоль - потужний «калькулятор» (робота з числами, матрицями, файлами, зображеннями, статистичного аналізу та ін.).
- Мова програмування для невеликих скриптів (обробка зображень, створення резервних копій).
- Мова програмування для прототипування (швидке створення шаблону програми з UI, швидка перевірка роботи алгоритму).
- Мова програмування для повноцінних програм (Gajim, BitTorrent, Dropbox, EVE Online).
- Мова програмування для web-застосунків (фреймворки Django, Flask, TurboGears, web2py, Pylons, Zope, WebWare підтримують швидке створення повнофункціональних і якісних сайтів на основі Python).
- Вбудована в додатки мова програмування (вбудована Python-консоль в якій можна оперувати з об'єктами додатків мовою Python: 3D моделювання - Blender, Maya, обробка зображень - GIMP, робота з ГІС даними - ESRI ArcGIS, математичні пакети - Sage, IPython Notebook, офісний пакет OpenOffice).

Транслятор — установка

Завантажити транслятор для будь-якої платформи можна на сайті: http://www.python.org . Актуальна версія – Python 3.6.5 (квітень 2018).

Python IDE (Integrated Development Environment)

Найбільш популярні:
- IDLE - поставляється з компілятором, середовище розробки;

- Eclipse + PyDev (http://www.pydev.org/);
- PyCharm from JetBrains (https://www.jetbrains.com/pycharm/);
- Sublime Text (https://www.sublimetext.com/);
- Wingware's Wing IDE 101 (http://wingware.com/downloads/wingide-101).

Синтаксис Python

Відступи в коді

- Функції в Pyton не мають явних операторів begin/end або фігурних дужок для позначення початку й кінця коду функції.
- Єдиним розділювачем є двокрапка (:) та відступи в коді.
- Блоки коду описуються їх відступами.
- "Блок коду" – це функції, умовні оператори, тіла циклів і так далі.
- Відступ позначає початок блоку, а прибирання відступу - його закінчення.
- Явні дужки чи ключові слова не потрібні.

Стандарт PEP 8

PEP (python enhanced proposal - заявки на поліпшення мови python). Цей стандарт – посібник з написання коду на Python.

- Використовуйте 4 пробіли на кожен рівень відступу
- Python 3 забороняє змішування табуляції і пробілів в відступах
- Обмежте довжину рядка максимум 79 символами
- Відокремлюйте функції верхнього рівня і визначення класів двома порожніми рядками
- Кодування Python 3 повинно бути UTF-8 (ASCII в Python 2)
- Слід уникати використання пробілів в наступних ситуаціях:
    - Усередині круглих, квадратних і фігурних дужок animals(tiger[2], {eagle: 6})
    - Перед комою, крапкою з комою і двокрапкою if x == y:
    - Відразу перед відкриваючою дужкою, після якої слідує індекс - dict['key'] = list[index]
- Завжди оточуйте наступні бінарні оператори одним пропуском з кожного боку: оператори присвоювання (=, + =, - = і інші), оператори порівняння (==, <,>,! =, <>, <=,> =, In, not in, is, is not), логічні оператори (and, or, not)
- Не використовуйте пробіли навколо знаку =, якщо він використовується для позначення іменованого аргументу або повернуться до стандартних значень.

Приклад. Демонстрація операторів цикла for та while.

Файл for_while.py

```
lastName="Smith"
count=0
for letter in lastName:
    print(letter," ",count)
```

```
    count+=1
print("-------")
count=0
while (count<5):
    print(lastName[count]," ",count)
    count+=1
```

Введення даних

```
>> age = input(“Please, enter your age :  “)
Please, enter your age :  24
>> print( type(age))
<class ‘str’>
>> print (“You age is   “, age)
You age is  24
```

Файл sum_int.py

```
     n1 = int(input(“Enter n1 : “))
     n2 = int(input(“Enter n2 : “))
     sum = n1 + n2
     print(“n1 + n2 = “, sum)
```

Замість int можна використати float.

Рядки в Python

```
>>> S = ‘Spam’
>>> len(S)
4
>>> S[0]
‘S’
>>> S[-1]
‘m’
>>> S[-2]
‘a’
>>> S[1:3]    # Slice – “зріз”
‘pa’
>>> S + ‘xyz’    # Concatenation
‘Spamxyz’
>>> S              # S is unchanged
‘Spam’
>>> S * 8         # Repetition
‘SpamSpamSpamSpamSpamSpamSpamSpam’
>>> S = ‘Spam’
>>> S.find(‘pa’)
1
>>> S.replace('pa', 'XYZ')
'SXYZm'
>>> S
'Spam‘
>>> S = 'shrubbery'
>>> L = list(S)              # Expand to a list: [...]
>>> L
['s', 'h', 'r', 'u', 'b', 'b', 'e', 'r', 'y']
>>> L[1] = 'c'                # Change it in place
>>> ''.join(L)                # Join with empty delimiter
'scrubbery‘
>>> x = 3.4
>>> str(x)
```

```
'3.4'
>>> format(x, "0.5f")
'3.40000'
>>> smiles = "C(=N)(N)N.C(=O)(O)O"
>>> smiles.split(".")
['C(=N)(N)N', 'C(=O)(O)O']
if "Br" in "Brother":
     print ("contains brother")
email_address = "clin"
if "@" not in email_address:
     email_address += "@brandeis.edu"
```

Приклад: Чи є рядок паліндромом?

```
def isPalindrome(s):
    if len(s) <= 1:
        return True
    return s[0] == s[len(s) - 1] and isPalindrome(s[1 : len(s) - 1])

def main():
    line = input("Enter a string:  ")
    if isPalindrome(line):
        print("That was a palindrome !")
    else:
        print("That is not a palindrome !")
main()
```

Кортежи (tuple)

Кортеж в Python - це упорядкований набір об'єктів, в який можуть одночасно входити об'єкти різних типів. Кортеж задається перерахуванням його елементів у круглих дужках через кому, наприклад,

t = (12 , 'b' , 34.6 , 'derevo')

Елементам кортежу можна відразу зіставити які-небудь змінні:

t = (x , s1 , y , s2) =(12 , 'b' , 34.6 , 'derevo')

Основні операції з кортежами: len(t) , t1 + t2 , t ∗ n , t[i] , t[i : j :k] , min(t) , max(t).

>>> s= 'amamam '

>>> t = tuple(s)

t → (' a ' , 'm' , ' a ' , 'm' , ' a ' , 'm')

Списки (Lists)

Список в Python - це упорядкований набір об'єктів, в список можуть одночасно входити об'єкти різних типів.

lst = [12 , 'b' , 34.6 , 'derevo']

Основні операції зі списками: len(lst) , lst1 + lst2 , lst ∗ n , lst[i] , lst[i : j :k] , min(lst) , max(lst) , lst [i]=x , del lst [i] , lst [i : j]=x.

Основні методи списків:

Додавання елементу lst .append(x) .

Додавання кортежу або списку lst .extend(t).

lst .count(x) , lst .index(x) , lst .remove(x) , lst .pop(i) ,

lst .insert(i ,x) , lst.sort() , lst.reverse().

Набори (sets)

Набір використовується для зберігання невпорядкованого набору об'єктів. Щоб створити набір, використовуйте функцію set(), або фігурні дужки { }

```
>>> s = set( [3, 9, 2, 15])
>>> t = set ( “Hello”)
>>> t
{ ‘e’, ‘l’, ‘H’, ‘o’}
>>> {1, 2, 3, 4, 5}
```

Операції над наборами

```
a = t | s      # Union of t and s
b = t & s    # Intersection of t and s
c = t – s     # Set difference (items in t, but not in s)
d = t ^ s     # Symmetric difference (items in t or s, but not both)
t.add('x')            # Add a single item
s.update([10,37,42])   # Adds multiple items to s
t.remove('H')
```

Словники (Dictionaries)

Словники в Python - невпорядковані колекції довільних об'єктів з доступом по ключу. Їх іноді ще називають асоціативними масивами або хеш-таблицями.

```
>>> d = {}
>>> d
{}
>>> d = {'dict': 1, 'dictionary': 2}
>>> d
 {'dict': 1, 'dictionary': 2}
>>> d = dict(short='dict', long='dictionary')
>>> d
{'short': 'dict', 'long': 'dictionary'}
>>> d = dict([(1, 1), (2, 4)])
>>> d
{1: 1, 2: 4}
>>> D = { ‘a’: 1, ‘b’: 2, ‘c’: 3}
>>> D.keys()
        dict_keys([‘b’, ‘a’, ‘c’])
>>> D.values()
        dict_values([2, 1, 3])
>>> D.items()
        dict_items([(‘b’,2), (‘a’,1), (‘c’,3)])
>> D.copy()
{‘a’: 1, ‘b’: 2, ‘c’: 3}
D.clear()  - очистка словника
D.update(D2)
D.get(K)
D.pop(K)
```

Файли

```
f = open('data.dat')   #  f is a file object
for line in f:                #  Read each line as text
        print(line.strip())      #  Remove trailing newline character
f.close()                      #  Close the file
input_file = open(“in.txt")
output_file = open(“out.txt", "w")
for line in input_file:
                output_file.write(line)
```

Функції

Функцію можна визначити двома способами :

- за допомогою ключового слова def
- lambda-виразом

```
def  remainder (a, b):
        q = a // b       # // is truncating division.
        r = a - q*b
        return r
```

Виклик функції : result = remainder(37,15)

Змінні, визначені всередині функції, є локальними

```
count = 0
...
def foo():
        global count
        count += 1            # Changes the global variable count
```

Приклад

```
def multi_table(a):
        for i in range(1, 11):
                print('{0} x {1} = {2}'.format(a, i, a*i))
if __name__ == '__main__':
        a = input('Enter a number: ')
        multi_table(float(a))
```

Можна використовувати кортежі, щоб повернути декілька значень

```
def divide(a,b):
        q = a // b        # If a and b are integers, q is integer
        r = a - q*b
        return (q,r)
```

Виклик : q, r = divide(1456, 33)

Можна задавати значення аргументів за замовчуванням

```
def connect(hostname, port, timeout=300):
        # Function body
```

Виклик : connect('www.python.org', 80)

або так - connect(port=80,hostname="www.python.org")

Модулі

Модулі виконують дві важливих функції:

- Повторне використання коду.
- Управління адресним простором імен.

Python дозволяє помістити класи, функції або дані в окремий файл і використовувати їх в інших програмах. Такий файл називається модулем.

Підключити модуль можна за допомогою інструкції import

```
>>> import os
>>> os.getcwd()
'C:\\Python33‘
```

```
>>> import math
>>> math.e
2.718281828459045
```

Використання псевдонімів

```
>>> import math as m
>>> m.e
2.718281828459045
```

Інструкція from

from <Назва модуля> import <Атрибут 1> [as <Псевдонім 1>], [<Атрибут 2> [as <Псевдонім 2>] ...]

from <Назва модуля> import *

```
>>> from math import e, ceil as c
>>> e
2.718281828459045
>>> c(4.6)
5
```

Об’єкти та класи

Всі значення, використовувані програмами, є об’єкти.

Об’єкт містить деякі внутрішні дані та має методи, що дозволяють виконувати різні операції над цими даними.

Функція dir () виводить список всіх методів об'єкта.

```
>>> items = [37, 42]
>>> dir(items)
[‘__add__’, ‘__class__’, ‘__contains__’, ‘__delattr__’, ‘__delitem__’,
        ...
‘append’, ‘count’, ‘extend’, ‘index’, ‘insert’, ‘pop’,
‘remove’, ‘reverse’, ‘sort’]
>>>
```

Для визначення нових типів об'єктів використовується інструкція class

Приклад. Клас стека

```
class Stack(object):
   def __init__(self):    # Ініціалізація стека
      self.stack = [ ]
   def push(self,object):
      self.stack.append(object)
   def pop(self):
      return self.stack.pop()
   def length(self):
```

```
        return len(self.stack)
```

Клас Stack успадковує властивості і методи класу object.

self – аналог this (посилання на екземпляр класу).

Методи, імена яких починаються і закінчуються двома символами підкреслення, є спеціальними методами.

```
s = Stack()
s.push("Dave")
s.push(42)
s.push([3,4,5])
x = s.pop()       # x отримає значення [3,4,5]
y = s.pop()       # y отримає значення 42
del s             # Знищує об'єкт s
```

Всі методи класу Stack можуть застосовуватися тільки до екземплярів даного класу (тобто до створених об'єктів).

Статичні методи

```
class EventHandler(object):
   @staticmethod
   def dispatcherThread():
      while (1):
         # Чекання запиту
...
EventHandler.dispatcherThread() # Виклик метода
```

В даному випадку @staticmethod оголошує наступний за ним метод статичним.

@staticmethod - це приклад використання декоратора

Приклад класу

```
class Car:
            def  __init__( self, make, model, year)
                  self.make = make
                  self.model = model
                  self.year = year
                  print("Instance object of the class is created")
            def  displayParameters(self) :
                  print("Make  : ", self.make)
                  print("Model  : ", self.model)
                  print("Year  : ", self.year)
                  print()

      car1 = Car("A", "B", 2015)
      car1.displayParameters()
```

Class variable

```
class Car:
            totalNumber = 0
            def  __init__( self, make, model, year)
```

```
                    Car.totalNumber +=1
                    self.make = make
                    self.model = model
                    self.year = year
                    print(“Total number of car is ”, Car.totalNumber)
               def  displayParameters(self) :
                    print(“Make  : “, self.make)
                    print(“Model  : “, self.model)
                    print(“Year  : “, self.year)
                    print()
          car1 = Car(“A”, “B”, 2015)
          car2 = Car(“C”, “D”, 2014)
```

Деструктор

```
class Car:
               totalNumber = 0
               def  __init__( self, make, model, year) :
                    Car.totalNumber +=1
                    self.make = make
                    self.model = model
                    self.year = year
               def  __del__ (self) :
                    Car.totalNumber -=1
               def  displayParameters(self) :
                    print(“Make  : “, self.make)
                    print(“Model  : “, self.model)
                    print(“Year  : “, self.year)
          car1 = Car(“A”, “B”, 2015)
          car2 = Car(“C”, “D”, 2014)
          del car2
```

Приватні поля класу та властивості

Ми можемо приховати атрибути класу (зробити їх приватними), якщо перед іменем атрибуту поставимо два підкреслення.

```
     class Gadget:          # Файл  Gadget.py
          """A class used for modelling Gadgets in a web shop."""
          __weight = 100
          __operating_system = None
          __battery_capacity = 2000
          __screen_size = 1
          def __init__(self, weight, operating_system,
battery_capacity, screen_size):
              self.__weight = weight
              self.__operating_system = operating_system
              self.__battery_capacity = battery_capacity
     self.__screen_size = screen_size
          def get_weight(self):
               return self.__weight
          def set_weight(self, weight):
```

```
          self.__weight = weight
weight = property(get_weight, set_weight)
     @property
     def operating_system(self):
          return self.__operating_system
     @operating_system.setter
     def operating_system(self, new_os):
          self.__operating_system = new_os
```

Застосування

```
>>> from Gadget import Gadget
>>> my_iphone = Gadget(240,'iOS',1980,4)
>>> my_iphone.weight
240
>>> my_iphone.weight = 255
>>> my_iphone.weight
255
>>> my_iphone.operating_system
'iOS'
>>> my_iphone.operating_system = 'iOS 8.1'
>>> my_iphone.operating_system
'iOS 8.1'
```

Одиночне успадкування в Python

В Python успадкування здійснюється за допомогою наступного синтаксису: class MySubClass(MyBaseClass).

```
class Animal:
    __name = None
    __age = 0
    __is_hungry = False
    __nr_of_legs = 0
    def __init__(self, name, age, is_hungry, nr_of_legs):
        self.name = name
        self.age = age
        self.is_hungry = is_hungry
        self.nr_of_legs = nr_of_legs
    def eat(self, food):
        print("{} is eating {}.".format(self.name, food))
    @property
    def name(self):
        return self.__name
    @name.setter
    def name(self,new_name):
        self.__name = new_name
    @property
    def age(self):
        return self.__age
    @age.setter
    def age(self,new_age):
        self.__age = new_age
    @property
    def is_hungry(self):
        return self.__is_hungry
    @is_hungry.setter
    def is_hungry(self,new_is_hungry):
```

```
            self.__is_hungry = new_is_hungry
        @property
        def nr_of_legs(self):
            return self.__nr_of_legs
        @nr_of_legs.setter
        def nr_of_legs(self,new_nr_of_legs):
            self.__nr_of_legs = new_nr_of_legs
    class Snake(Animal):
        __temperature = 28
        def __init__(self, name, temperature):
            super().__init__(name, 2, True, 0)
            self.temperature = temperature
        def eat(self, food):
            if food == "meat":
                super().eat(food)
            else:
                raise ValueError
        @property
        def temperature(self):
            return self.__temperature
        @temperature.setter
        def temperature(self,new_temperature):
            if new_temperature < 10 or new_temperature > 40:
                raise ValueError
            self.__temperature = new_temperature
```

Multiple Inheritance in Python

```
    class Phone(object):
        def __init__(self):
            print("Phone constructor invoked.")
        def call_number(self, phone_number):
            print("Calling number {}".format(phone_number))
    class Computer(object):
        def __init__(self):
            print("Computer constructor invoked.")
        def install_software(self, software):
            print("Computer installing the {}
software".format(software))
    class SmartPhone(Phone,Computer):
        def __init__(self):
            Phone.__init__(self)
            Computer.__init__(self)
    . . . . . .
    >>> my_iphone = SmartPhone()
    Phone constructor invoked.
    Computer constructor invoked.
    >>> my_iphone.call_number("123456789")
    Calling number 123456789
```

Обробка винятків

```
    try:
        Value1 = int(input("Type the first number: "))
        Value2 = int(input("Type the second number: "))
        Output = Value1 / Value2
```

```
except  ValueError:
    print("You must type a whole number!")
except  KeyboardInterrupt:
    print("You pressed Ctrl+C!")
except  ZeroDivisionError:
    print("Attempted to divide by zero!")
except  ArithmeticError:
    print("An undefined math error occurred.")
else:
    print(Output)
TryAgain = True
while TryAgain:
    try:
        Value = int(input("Type a whole number. "))
    except  ValueError:
        print("You must type a whole number!“)
        try:
            DoOver = input("Try again (y/n)? ")
        except:
            print("OK, see you next time!")
            TryAgain = False
        else:
            if (str.upper(DoOver) == "N"):
                TryAgain = False
    except KeyboardInterrupt:
        print("You pressed Ctrl+C!")
        print("See you next time!")
        TryAgain = False
    else:
        print(Value)
        TryAgain = False
```

Генерація винятків

Приклад 1

```
try:
    raise ValueError
except ValueError:
    print("ValueError Exception!")
```

Приклад 2

```
try:
    Ex = ValueError()
    Ex.strerror = "Value must be within 1 and 10."
    raise Ex
except ValueError as e:
    print("ValueError Exception!", e.strerror)
```

Пакети

Пакети в мові Python забезпечують спосіб розподілу набора модулів по каталогам файлової системи та використання точкової нотації для доступу до модулів підпакетів.

Управління пакетами

Пакетами Python можуть бути інструменти, бібліотеки, фреймворки, застосунки. Існують десятки тисяч доступних пакетів, які можна використовувати для створення власних проектів.

Інструменти управління пакетами

Найбільш часто використовувані менеджери пакетів Python – це

- pip
- easy_install.

Дані інструменти допомагають виконати наступні завдання: скачування, установка, видалення пакетів; зборка пакетів; управління пакетами Python і багато іншого.

Основи web-фреймворку Django

Django (Джанго) – високорівневий відкритий Python-фреймворк для розробки веб-систем. Названий на честь джазмена Джанго Рейнхардта.

Архітектура Django подібна MVC. Рівневу архітектуру Django часто називають «Модель-Шаблон-Подання» (MTV).

Сайт https://www.djangoproject.com/

Має власний ORM ((Object-relational mapping)) для роботи з базами даних, вбудований інтерфейс адміністратора, диспетчер URL на основі регулярних виразів, шаблонізатор, бібліотеку для роботи з формами, інтернаціоналізацію.

Останній реліз - Django 2.0.3 (на квітень 2018). Розробники: Russell Keith-Magee, Adrian Holovaty, Simon Willison, Jacob Kaplan-Moss, Wilson Miner – перший реліз в 2003 році.

Веб-фреймворк Django використовується в таких великих і відомих сайтах, як Instagram, Bitbucket, Disqus, Mozilla Firefox, NASA, The Washington Times, Pinterest та ін. Також Django використовується в якості веб-компонента в різних проектах, таких як Graphite - система побудови графіків та моніторингу, FreeNAS - вільна реалізація системи зберігання та обміну файлами та ін. DjangoSites (https://djangosites.org/) - це сайт, який відстежує веб-сайти, створені за допомогою Django. Станом на березень 2018 р. налічується понад 5280 сайтів.

Як порівняти Django з іншими веб-фреймворками?

Станом на серпень 2017 року (https://devopedia.org/django), Django був шостим за популярністю веб-фреймворком (після ASP.NET, AngularJS, Ruby on Rails, ASP.NET MVC, React). Якщо розглядати тільки основані на Python фреймворки, то Django є на верхній позиції.

Серед основаних на Python фреймворків є наступні:

- Non-full-stack - Flask, Bottle, CherryPy, Pyramid.
- Full-stack - Django, Tornado, TurboGears, web2py.

Основні принципи Django: Django прагне бути “пітонічним”, DRY (Don’t Repeat Yourself), слабка залежність та гнучкість, швидка розробка.

Архітектура Django має наступний вигляд (Рис.57)

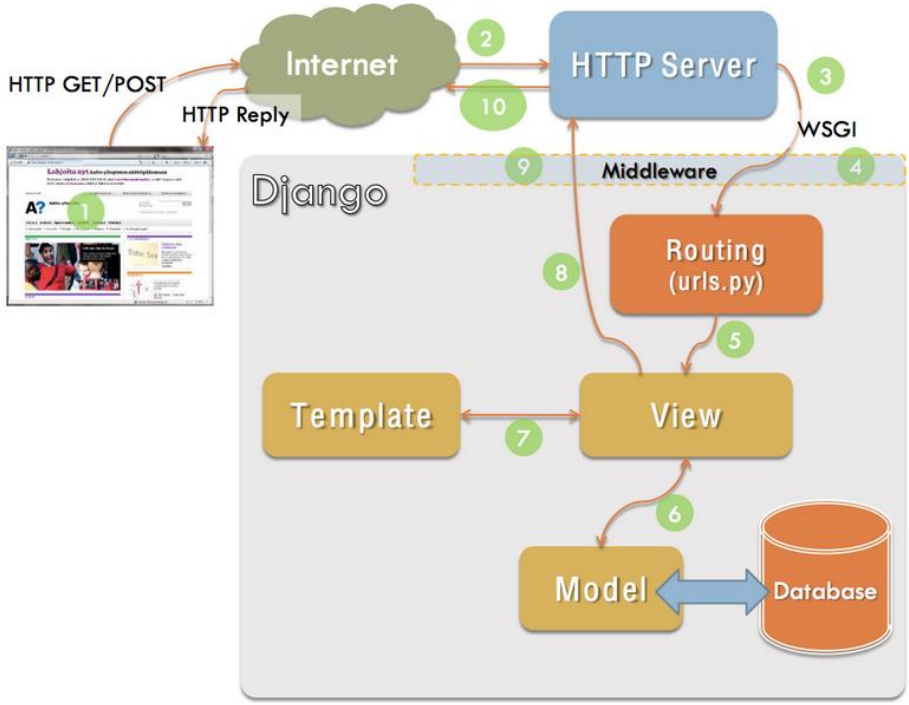


Рис.57. Архітектура Django.

Маємо наступні компоненти архітектури MTV:

Model – рівень доступу до даних. Це стосується доступу, перевірки та взаємовідносин даних. Моделі - це класи Python, які посереднюють між Django ORM та таблицями бази даних.

Template – презентаційний шар. Відповідає за те, як дані відображаються клієнту.

View – шар бізнес-логіки. Це міст між моделями та шаблонами. Перегляд (view) відкриває дані моделі та перенаправляє їх до шаблону для презентації. На відміну від визначення перегляду в традиційній архітектурі MVC, перегляд Django описує, які дані ви бачите, а не як ви бачите. Це стосується обробки, а не презентації.

Проект в Django

Проект в Django являє собою набір конфігураційних файлів і застосунків Django. Проект може містити множину застосунків Django.

Застосунок Django – набір файлів, що містять опис моделей та представлень, які є частиною одного пакета Python і являють собою повний застосунок. Застосунок може міститися у декількох проектах.

Створення та запуск Django проекта

Почати новий проект:

```
django-admin.py startproject example
```

Запустити локальний сервер:

```
python manage.py runserver
```

Мінімальна структура проекта Django

example/ - каталог проекта

example/ - пакет проекта

__init__.py
settings.py - налаштування проекта
urls.py - конфігурація URL адрес
wsgi.py - точка входу для wsgi
manage.py - командна утіліта Django

WSGI (Web Server Gateway Interface) - стандарт взаємодії між Python-програмою, що виконується на стороні сервера, і самим веб-сервером, наприклад Apache.

Middleware

Крім додатків і серверів, стандарт WSGI дає визначення middleware-компонентів (компонентів проміжного шару), що надають інтерфейси як додатку, так і серверу. Тобто для сервера middleware є додатком, а для додатка - сервером. Це дозволяє складати «ланцюжки» WSGI-сумісних middleware-компонентів.

Middleware можуть брати на себе такі функції (але не обмежуються цим): обробка сесій, аутентифікація / авторизація, управління URL (маршрутизація запитів), балансування навантаження, пост-обробка вихідних даних (наприклад, перевірка на валідність), зміна заголовків.

Створення Django застосунків

```
python manage.py startup blog
```

Після створення нового Django застосунку необхідно зареєструвати його в списку застосунків, що належать проекту.

Файл settings.py:

```
INSTALLED_APPS = (
    . . . . . .
    ‘example’,
    ‘blog’
)
```

Мінімальна структура Django застосунку

blog/ - пакет за стосунку;
__init__.py
admin.py - налаштування панелі адміна;
migrations/ - пакет, що містить міграції даних;
__init__.py
forms.py - опис форм введення інформації;
models.py - опис таблиць БД;
tests.py - unit tests;
views.py - бізнес-логіка за стосунку.

Функція подання (view)

Являє собою Python функцію, яка породжує вміст сторінки.

```
from django.http import HttpResponse
def hello(request):
```

return HttpResponse(“Hello, world!”)

Об’єкти запиту і відповіді (HttpRequest, HttpResponse)

Клас HttpRequest являє собою HTTP-запит, отриманий з сторінки клієнта, і містить:

- Інформацію про URL (path і т.д.).
- HTTP-заголовки запиту.
- Інформацію про відправлені дані (GET, POST та FILES).
- Інформацію про сесії та cookies (session та COOKIES).
- Інші змінні сервера.

Клас HttpResponse слугує для конструювання відповіді на запит.

```
HttpResponse(content = “”, mimetype = None, status = 200, content_type = DEFAULT_CONTENT_TYPE)
```

Але якщо необхідно додавати вміст поступово, можна використовувати об’єкт response як об’єкт файлу:

```
response = HttpResponse()
response.write(“<p>Here’s my Web Page</p>”)
response.write(“<p>Here’s another paragrath</p>”)
```

Підкласи HttpResponse

Модуль django.http містить декілька підкласів HttpResponse, які представляють різні види HTTP-відповідей: HttpResponseRedirect (302), HttpResponseNotFound (404), HttpResponseForbidden (403), HttpResponseServerError (500) та інші.

Конфігурування URLs

Побудовано на принципі слабкої зв’язності.

Кожна конфігурація повинна зберігатися у вигляді файлу з програмним кодом Python, зазвичай з іменем urls.py.

Такі файли повинні визначати об’єкт urlpatters, який отримується в результаті виклику функції django.conf.patterns.

Шлях до головного конфігуратора визначається в settings.py змінною ROOT_URLCONF.

Приклад.

```
from django.conf.urls import patterns, include, url
     from example.views import hello

     urlpatterns = patterns(“,
          #Examples
          (r’^hello/’, hello)
          (r’^$’, ‘example.views.home’, name=‘home’)
          (r’^example/’, include(‘example.blogs.urls’))
     )
```

Система шаблонів Django

Шаблон в Django являє собою рядок тексту в спеціальному форматі, призначений для відділення подання документу (view) від його даних.

Мова шаблонів в Django не має ніякого відношення до Python, і створена для веб-технологів та дизайнерів інтерфейсів.

Ідеологічні основи шаблонів Django

- Бізнес-логіку треба відділити від логіки подання.
- Систаксис не треба прив'язувати до HTML/XML.
- Передбачається, що дизайнери впевнено володіють HTML.
- Не передбачається, що дизайнери вміють програмувати на Python.
- Не ставилось завдання винайти нову мову програмування.

Конфігурування та завантаження Django шаблонів

Django пропонує зручний та потужний API для завантаження шаблонів з файлової системи.

- Спершу в settings.py треба задати:
  TEMPLATE_DIRS = (
  "/шлях до/папки що/містить шаблони/"
)
- Потім можна використовувати:
  from django.shortcuts import render_to_response
  def hello(request):
  return render_to_response('hello.html')

Модель даних в Django

Логічний опис структури даних, що зберігаються в БД, надається мовою Python. Всі моделі являють собою клас, що спадкує django.db.models.Model

```
#app/models.py  (загальний синтаксис):
from django.db import models
class CustomModel(models.Model):
    name = models.CharField(max_length=30)
    body = models.TextField()
    email = models.EmailField(blank=True)
    number = models.IntegerField()
```

Міграції в Django

Їх використовують для переносу змін в моделях (додавання/видалення поля або моделі і т.д.) на структуру бази даних.

Міграції – це дуже потужний механізм, який треба дуже обережно застосовувати в реальних умовах.

```
manage.py makemigrations [<app_label>]
```

2. ЛАБОРАТОРНІ РОБОТИ

2.1 Лабораторна робота 1. Робота з формами в PHP.

Завдання: Створити форму реєстрації на сервері з перевіркою введених даних на коректність.

Приклад виконання аналогічного завдання [8].

Початкова форма показана на Рис.58.

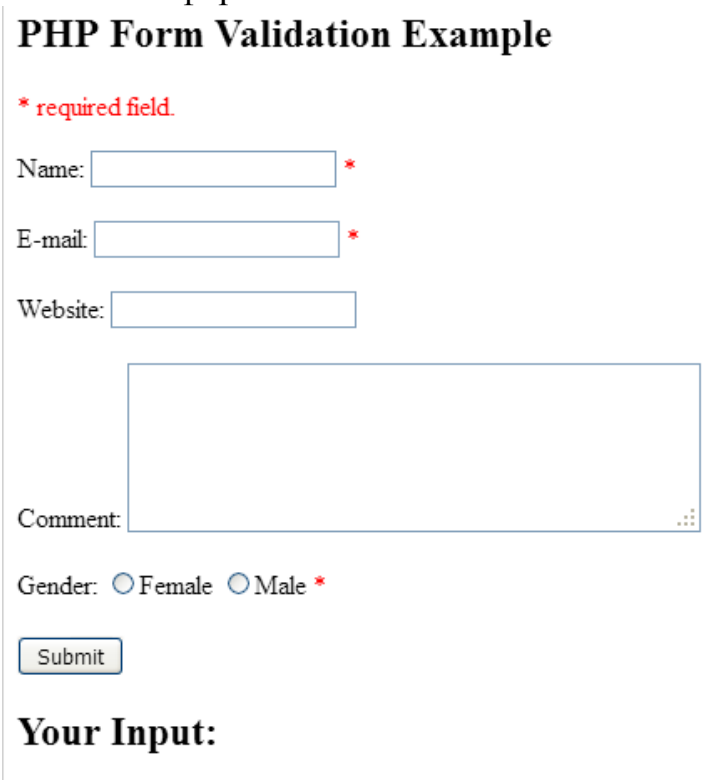


Рис.58. Початкова форма

Обробку форми виконує наступний скрипт:

```
<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
```

```
    // check if name only contains letters and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
      $nameErr = "Only letters and white space allowed";
    }
  }
  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }
  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid
    if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }
  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }
  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field.</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  <span class="error">* <?php echo $nameErr;?></span>
  <br><br>
  E-mail: <input type="text" name="email">
  <span class="error">* <?php echo $emailErr;?></span>
```

```
  <br><br>
  Website: <input type="text" name="website">
  <span class="error"><?php echo $websiteErr;?></span>
  <br><br>
  Comment: <textarea name="comment" rows="5" cols="40"></textarea>
  <br><br>
  Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <span class="error">* <?php echo $genderErr;?></span>
  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>

</body>
</html>
```

2.2 Лабораторна робота 2. Використання MySQL за допомогою PHP.

Завдання: Модифікувати лабораторну роботу 1, додавши збереження даних реєстрації (профіля користувача) в базі даних MySQL. Необхідно забезпечити можливість перегляду й редагування профілів.

Методичні вказівки.

Треба запустити phpMyadmin, створити базу даних users з полями, що відповідають полям реєстраційної форми. Далі при написанні скрипта для обробки форми вибрати спосіб спілкування з MySQL (бібліотеки mysqli або PDO) і використати матеріали лекції 5.

2.3 Лабораторна робота 3. Простий сайт з авторизацією.

Завдання: Використовуючи результати лабораторних робіт 1 і 2, створити сайт, що складається з п'яти сторінок:

1) Головна сторінка, що містить форму входу в систему.
2) Сторінка перегляду або зміни персональних даних.
3) Сторінка реєстрації.
4) Гостьова книга.

5) Сторінка "Про сайт".

У профілі користувача повинне бути присутнім поле "Фотографія", через яке користувач завантажує на сервер свою аватарку, яка відображається біля повідомлень даного користувача в гостьовій книзі.

Ідентифікаційну інформацію про користувача потрібно зберігати в cookie.

Пароль у базі даних зберігається в хешованому вигляді.

Гостьова книга повинна бути доступна тільки аутентифікованим користувачам.

Сайт повинен бути організований із простим поділом дизайну й змісту (верхній блок, основний зміст, нижній блок).

Всі часто використовувані функції повинні бути винесені в окремий файл із розширенням inc, який сторінки будуть підключати за допомогою команди require_once.

2.4 Лабораторна робота 4. Поліпшений сайт.

Завдання: До сайту, розробленому в лабораторній роботі 3 необхідно додати: 1) шаблон дизайну сайту; 2) роль адміністратора в профілі користувача; 3) модерування гостьової книги адміністратором сайту (видалення окремих повідомлень, видалення й блокування нових повідомлень порушника, фільтрація по IP адресі).

2.5 Лабораторна робота 5. Розробка CMS для адміністрування сайту.

Завдання: Додати до сайту систему керування вмістом, доступну тільки користувачам, що мають роль адміністратора.

CMS повинна забезпечити:

1. Можливість вибору шаблону дизайну сайту.
2. Редагування динамічної зміни тексту сторінок (в TextArea або вбудованому HTML редакторі, наприклад, CKEditor), які зберігаються в базі даних.
3. Систему статей (або новин) з їхнім редагуванням в адмінській частині та зберіганням у базі даних.
4. Адміністрування користувачів (видалення, зміна пароля).
5. Перегляд і оновлення статистики відвідувань сайту.

2.6 Лабораторна робота 6. Розробка сайту в середовищі WordPress.

Завдання: Розробити локально в середовищі WordPress умовний сайт курсів навчання з наданням послуг Навчання, Консультації, Розробка. В якості теми застосувати тему Sydney (http://athemes.com/theme/sydney/).

Хід роботи.

Передумови

Комп’ютер, з’єднаний з Інтернет.

XAMPP в якості локального сервера.

PHP версії 5.2.4 або більше.

MySQL версії 5.0 або більше.

Пакет WordPress (https://wordpress.org/ або https://uk.wordpress.org/).

1) Установка WordPress:

STEP1 Встановлюємо локальний сервер (XAMPP).

STEP2 Стартуємо сервер.

STEP3 Створюємо директорію xampp/htdocs/myblog і копіюємо в неї вміст архіву пакета WordPress (файл wordpress-4.6.1-uk.zip).

STEP4 Відкриваємо phpMyAdmin і створюємо базу даних my_first_blog (with utf8_general_ci).

STEP5 Відкриваємо WordPress в браузері (http://localhost/myblog) і робимо відповідні на лаштунки (Рис.59)

Рис.59. П’ятий крок установки WordPress

STEP6 Run and Install.

STEP7 Задаємо параметри облікового запису адміністратора (Рис.60) localhost/myblog/wp-login.php).

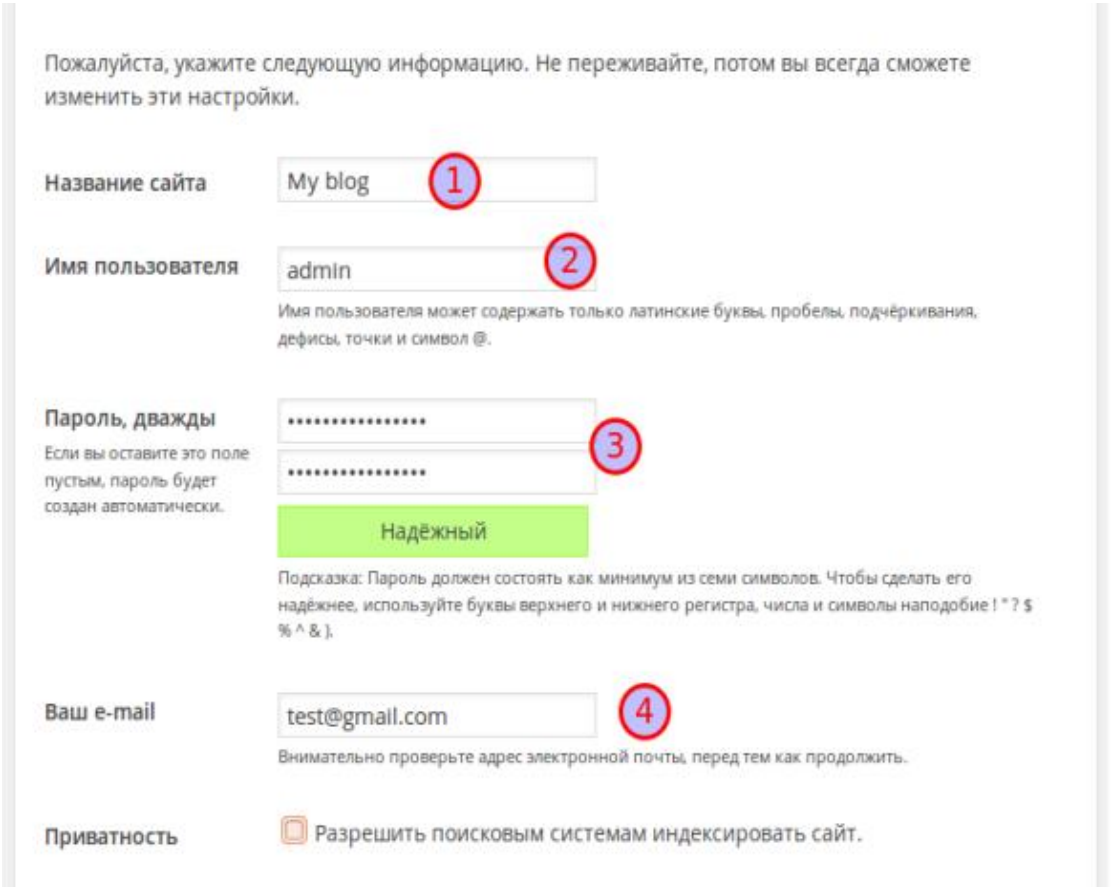


Рис.60. Задання параметрів облікового запису адміністратора

STEP8 Після того, як WP встановлено, для входу в адмін-панель використовуйте адресу http://localhost/myblog/wp-admin (вводячи логін і пароль, задані вами в ході установки WP)

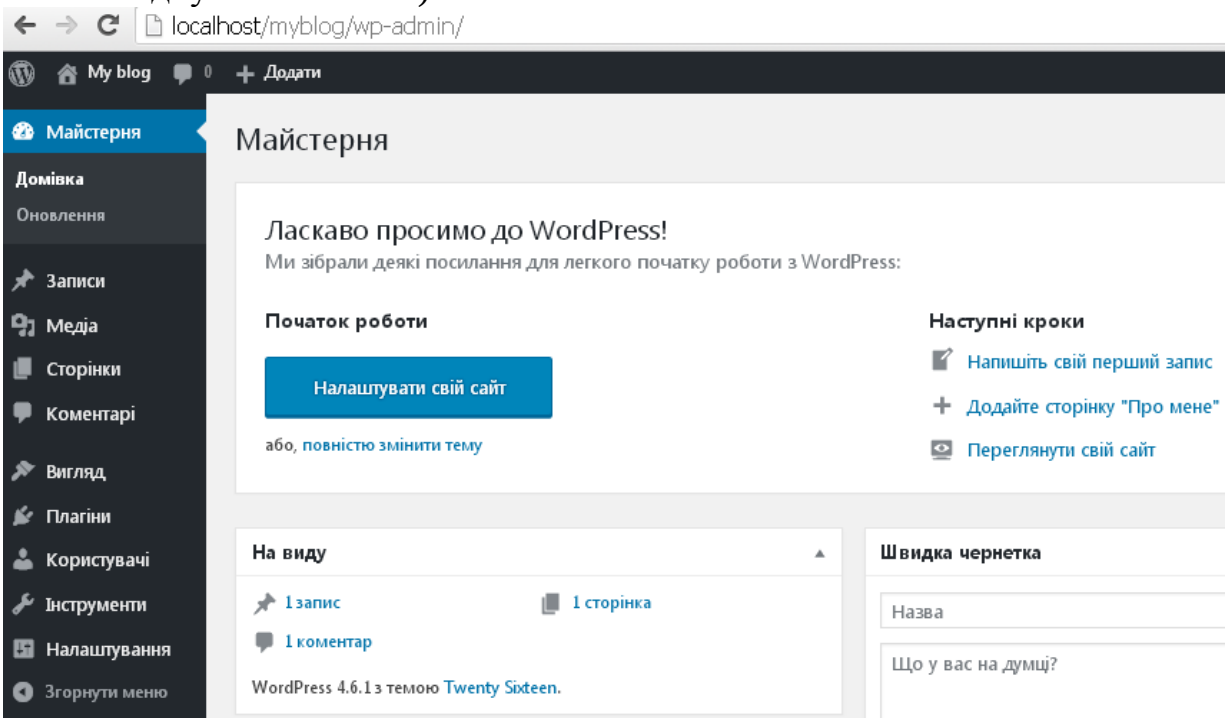


Рис.61. Панель адміністратора

Фронтальна частина буде доступна за адресою http://localhost/myblog/

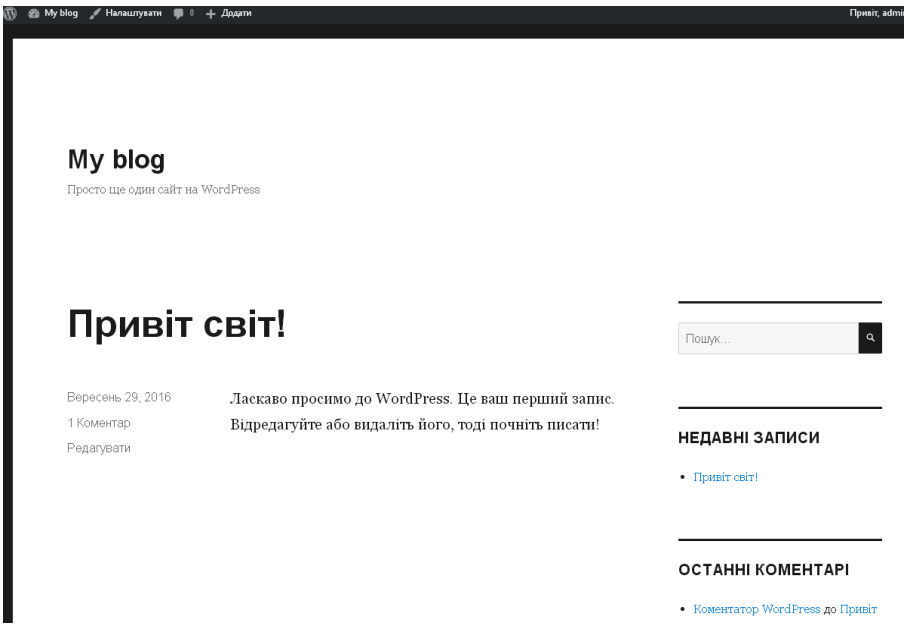


Рис.62. Фронтальна частина сайту

2) Налаштування WordPress після установки.

STEP1 Налагоджуємо людино-зрозумілі посилання (“человеко-понятные урлы - ЧПУ”). Вони використовуються для більш зрозумілого читання URL’ів, як людьми, так і пошуковими машинами. Отже, переходимо в Адмін-панель (*Майстерня)/Налаштування/Постійні посилання* і вмикаємо “*Назва запису*”. Далі встановлюємо пагін **Cyr-To-Lat**. Даний плагін прописує в URL заголовок статті латинськими літерами.

Отже, переходимо в *Адмін-панель/Плагіни/Додати*/, в пошуковому рядку вводимо назву плагіну, завантажуємо і встановлюємо його.

Зауваження. При роботі в корпоративній мережі кафедри ПЗАС ЗДІА треба налаштувати WordPress на роботу з проксі. Для цього необхідно додати в файл wp-config.php два рядки:

define(WP_PROXY_HOST, '192.168.1.1');

define(WP_PROXY_PORT, 3128);

STEP2 Додаємо плагін **Loco Translate**. Це найкращий плагін, призначеним для локалізації (перекладу) тем WordPress, він вбудовується безпосередньо в панель управління системою WordPress, не вимагаючи при цьому використання зовнішніх редакторів, на зразок Poedit.

3) Створення структури сайту. Нехай наш сайт буде складатися з наступних розділів: Головна, Мій блог, Фотогалерея, Про мене, Контакти. Розділи Про мене та Контакти будуть статичними.

STEP1 Створення/редагування статичних сторінок WP. Перейдіть на адмін-панель і виберіть в меню зліва Сторінки/Додати (буде відкрита сторінка редактора). Введіть заголовок сторінки (“Про мене”), основний текст та, якщо потрібно, зображення. Для збереження змін натисніть Оприлюднити. Далі натисніть Переглянути сторінку.

STEP2 Блог. Категорії та Записи. Блог складається з Постів = статей, які можна згрупувати по Категоріях. В меню зліва на адмін-панелі виберемо

Записи/Категорії. Створимо дві категорії: Блог і Фотогалерея. Щоб створити записи Блога перейдіть по посиланню “Записи/Додати” в меню зліва. Для кожного запису треба вибрати Формат (Стандартний, Примітка, Зображення, Відео, Цитата, Посилання, Галерея, Статус, Аудіо, Чат) і Категорію. Додайте декілька записів, вибираючи для Блога формат Стандартний, а для Фотогалереї – формат Зображення.

STEP3 Створення і управління меню. Коли всі необхідні сторінки та категорії створено, ми можемо створити меню – навігаційний список розділів нашого сайту. Для цього перейдемо по посиланню “Вигляд/Меню” в лівій панелі. Для створення нового меню слід дати назву (наприклад, “Головне меню”) і натиснути “Створити меню”. Після чого ви зможете додавати в ваше меню посилання на відповідні Розділи. Список можливих варіантів буде на лівій панелі. Після додавання потрібних пунктів меню, потрібно задати для меню область розташування. В нашому випадку ми повинні розмістити в меню 5 пунктів згідно завданню: Головна, Мій блог, Фотогалерея, Про мене, Контакти.

4) Підключення теми Sydney. Як підключити тему? Через меню Майстерня (Адмін-панель, Консоль)/Вигляд/Теми. За замовчуванням, діючою темою є тема Twenty Sixteen. Якщо стандартні теми вас не влаштовують, ви можете завантажити та встановити сторонню тему, натиснувши Додати нову. Тему треба попередньо завантажити і помістити в каталог …\xampp\htdocs\myblog\wp-content\themes. Після активації теми її треба налаштувати, щоб вміст вашого сайту відображався належним чином. Для цього перевірте ваші меню: можливо потрібно задати нові області призначення. Після зміни теми також потрібно налаштувати віджети.

STEP1 Скачуємо тему Sydney (http://athemes.com/theme/sydney/) і поміщаємо її в каталог myblog\wp-content\themes.

STEP2 Підключаємо тему Sydney. Під час активації теми потрібно додатково встановити плагіни Sydney Toolbox (або Types в старих версіях) та Page Builder.

STEP3 Далі потрібно на основі Sydney створити дочірню тему (Child Theme). Це дасть змогу редагувати тему, пристосувати її до ваших потреб. Для цього потрібно інсталювати плагін child theme (Плагіни/Додати, в рядку пошуку: “child theme”, в списку знайдених плагінів вибрати Child Theme Configurator, встановити і активувати його). Далі вибираємо Інструменти/Child Themes. В пункті 2 виберемо тему Sydney.

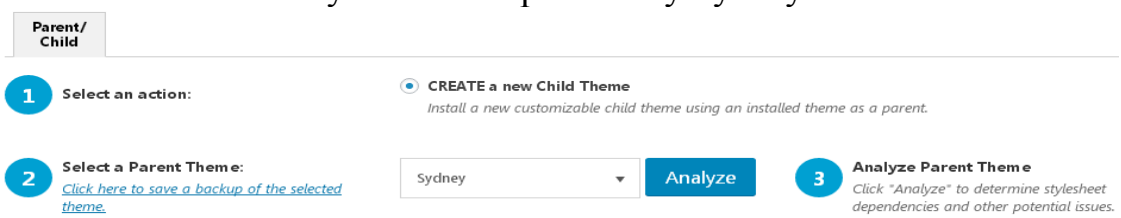


Рис.63. Підключення теми Sydney

Після натискання Analyze з'являються ще пункти 4 – 9. Погоджуємося з налаштуваннями за замовчуванням, в пункті 7 вводимо Theme Website - http://localhost/myblog/theme/sydney. Ставимо "галочку" в пункті 8 і натискаємо Create New Child Theme. Далі переходимо в Вигляд/Теми і бачимо в списку тему Sydney Child. Активуємо тему Sydney Child. Тепер її можна модифікувати.

STEP4 Зміна зображень та тексту в Sydney. Попередньо потрібно підготувати 2 зображення (1920x1275) та одне зображення (1280x549) для заміни титульних зображень в Sydney. Нові зображення повинні відповідати тематиці вашого сайту. Відвідуємо наш сайт (http://localhost/myblog/). Натиснемо "Налаштувати". Натиснемо "Область заголовка", далі "Тип заголовка". Вибираємо Тип заголовка сайта "Без заголовка (тільки меню)". Натиснемо "Зберегти і оприлюднити" та "<". Натиснемо "Слайдер заголовка" потім "Змінити зображення". Натиснемо "Завантаження файлів", вибираємо файл і завантажуємо, далі натиснемо "Обрати зображення". Задамо заголовок слайду "Ласкаво просимо до нашого блогу". Для другого слайду теж змінюємо зображення. Задамо заголовок слайду "Готові почати навчання?". Натиснемо "Зберегти і оприлюднити" та "<". Натиснемо "Зображення в заголовку". Далі можемо змінити поточний заголовок, а можемо сховати зображення, щоб звільнити місце для основного вмісту сторінок. Натиснемо "<" і ще раз "<". Далі можна відредагувати "Назву сайту, ключову фразу, логотип", "Шрифти", "Кольори", "Фонове зображення" і т.д. Далі заходимо в "Параметри блогу", в "Розміщенні блогу" замість "Класично" вибираємо "Сітка". В розділі "Довжина уривка" замість "55" вибираємо "20". Натиснемо "Зберегти і оприлюднити" та "<". Далі заходимо в "Підвал", в Області віджетів підвалу вибираємо "Один" замість "Три". Заходимо в "Кольори", "Фон меню" – "Обрати колір", набираємо "#c9c9c9" (це сірий фон), "Текст Body" – "Обрати колір", вибираємо чорний (#000000). "Колір бічної колонки (sidebar)" – "Обрати колір", чорний. "Зберегти і оприлюднити" та "<".

STEP5 Створення логотипу сайта. Заходимо на https://logomakr.com/ . Натискаємо "Next" і знову "Next" і далі "DONE". Натискаємо "Browse All Logos". Можемо організувати пошук логотипів за темою (people, book, laptop, . . .). Обравши логотип, задаємо колір, напис, орієнтацію, розмір, і зберігаємо логотип. Далі переходимо на наш сайт, натискаємо "Налаштувати", "Назва сайту, ключова фраза, логотип", в розділі "Завантажити свій логотип" натискаємо "Виберіть зображення", "Завантаження файлів", "Обрати файли", обираємо файл нашого логотипу і натискаємо "Обрати зображення", а далі "Зберегти і оприлюднити".

STEP6 Редагування теми Sydney. Далі треба діяти згідно документації (http://athemes.com/documentation/sydney/). Створення статичної головної сторінки. Додати/Сторінку, даємо ім'я, наприклад, "Моя головна сторінка", Шаблон: Головна сторінка (вибрати із списка), Оприлюднити. Додати/Сторінку, "My blog page", Оприлюднити. Налаштування сайту. Відвідати

сайт (http://localhost/myblog/). Натиснути “Налаштувати”. Головна сторінка відображає: “Статична сторінка”. Статична головна сторінка/ Головна сторінка/ Моя головна сторінка (вибрати із списка), Сторінка записів/ My blog page (вибрати із списка). Натиснути “Збережено”.

STEP7 Редагування теми Sydney. Створення послуг Навчання, Консультації, Розробка. Після встановлення плагіну Sydney Toolbox ви можете створювати Послуги, Співробітники, Клієнти та Відгуки (Services, Employees, Testimonials and Clients – доступні в Майстерні) – Дивись Демо (http://athemes.com/documentation/sydney/). Давайте створимо на головній сторінці щось на кшалт такого:

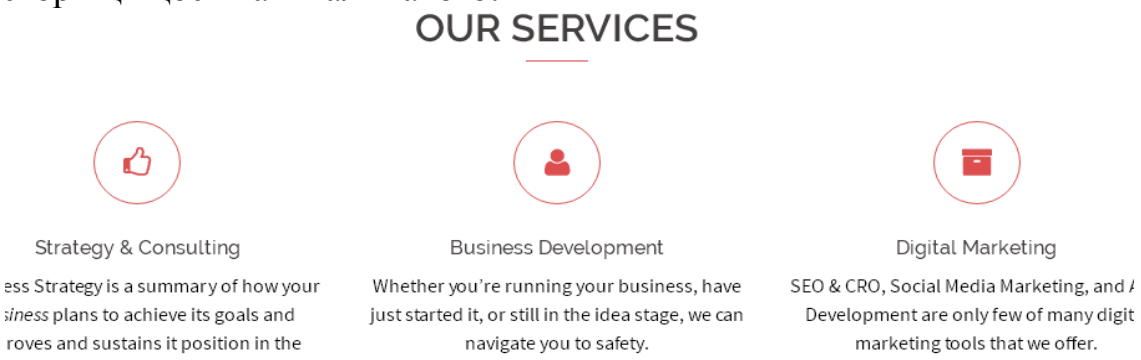


Рис.64. Створення послуг

Додавання послуг. Майстерня/ Services/Add New Service. Даємо назву, наприклад, “Розробка”. В вікні редактора вводимо “Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.”. В розділі “Service Icon” натискаємо “here” і підбираємо, наприклад, “fa-github”. В розділі “Service link” вводимо http://localhost/myblog/development (цю сторінку треба потім додати). Далі можна натиснути “Додати нову категорію” і дати їй ім’я, наприклад, “Розробка”. Далі натиснути “Оприлюднити”. Add New Service. Даємо назву, наприклад, “Навчання”. В вікні редактора вводимо “Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.”. В розділі “Service Icon” – “fa-laptop”. В розділі “Service link” вводимо http://localhost/myblog/education. Приєднуємо до категорії “Розробка”. Далі натискаємо “Оприлюднити”. Add New Service. Даємо назву, наприклад, “Консультації”. В вікні редактора вводимо “Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.”. В розділі “Service Icon” – “fa-university”. В розділі “Service link” вводимо http://localhost/myblog/consult. Приєднуємо до категорії “Розробка”. Далі натискаємо “Оприлюднити”.

STEP8 Sydney. Редагування головної сторінки. В Майстерні виберемо “Сторінки”, “Всі сторінки”, виберемо для редагування “Моя головна сторінка”. Натиснемо “Page Builder”. Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Тиснемо “Add Widget”, “Віджети теми Sydney”, “Sydney FP: Services Type A”, наводимо курсор миші на область Sydney FP: Services Type A і тиснемо “Edit”. В полі “Заголовок” набираємо “Наші послуги”, в

полі “Введіть слаг вашої категорії” наберіть “Розробка” і натисніть кнопку “Done”, а далі натисніть “Оновити” і “Переглянути сторінку” – ми побачимо на головній сторінці секцію “НАШІ ПОСЛУГИ” (Консультації, Навчання, Розробка). Зменшимо верхній і нижній відступ секції “НАШІ ПОСЛУГИ”. Для цього тиснемо “Редагувати сторінку” (повертаємось на Головну сторінку). Перед зоною Sydney FP: Services Type A наводимо курсор миші на зображення гаєчного ключа і тиснемо “Edit Row”, далі тиснемо “Theme”, в полі “Верхній і нижній відступ” задаємо 50. Далі тиснемо “Done” потім “Оновити” і “Переглянути сторінку”.

STEP9 Редагування головної сторінки. Додамо на головну сторінку ще одну секцію, наприклад, “Чим ми займаємося” (What We Do). Для цього тиснемо “Редагувати сторінку” (повертаємось на Головну сторінку). Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Клікаємо на новій секції (щоб зробити активною), тиснемо “Add Widget”, виберемо внизу списку віджет “Текст”. В секції “Текст” тиснемо “Edit”, в полі “Назва”вводимо “Чим ми займаємося” і тиснемо “Done”. В Майстерні наводимо курсор на Записи/Додати, тиснемо праву кнопку миші і виберемо “Open link on the new tab”. Поле заголовку залишаємо пустим, а в зоні вмісту вводимо, наприклад, “Ми даємо можливість отримати професію програміста” та “Приєднуйтесь!”. Виділяємо ці два рядки, на панелі інструментів в випадаючому списку “Параграф” виберемо “Заголовок 3” і Вирівнювання по центру. Далі в редакторі переходимо в режим Тексту, копіюємо в буфер html-код (<h3 style="text-align: center;">Ми даємо можливість отримати професію програміста.</h3><h3 style="text-align: center;">Приєднуйтесь!</h3>). Повертаємось на Головну сторінку, в секції “Текст” тиснемо “Edit”, вставляємо текст з буфера і тиснемо “Done”. Далі будемо редагувати секцію “Текст”. Перед зоною “Текст” наводимо курсор миші на зображення гаєчного ключа і тиснемо “Edit Row”, далі тиснемо “Theme”, в полі “Верхній і нижній відступ” задаємо 60. В полі “Колір” тиснемо “Обрати колір” і виберемо білий колір (@ffffff). В полі “Фонове зображення” тиснемо “Select Image”, “Завантаження файлів”, “Обрати файли” і т.д. В полі “Розміщення розділу” виберемо “На всю ширину” і далі тиснемо “Done” і “Оновити”. Переглянемо сторінку.

STEP10 Додавання на Головну сторінку ще двох секцій типу “Call To Actions” (з кнопками). Переходимо до режиму редагування Головної сторінки. Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Нова секція з’явиться між двома попередніми. Наводимо курсор миші на значок поряд з гаєчним ключем і перетягуємо секцію вниз (після секції “Текст”). Наводимо на гаєчний ключ і тиснемо “Edit Row”, далі “Theme”, в полі “Верхній і нижній відступ” задаємо 50. В полі “Фонове зображення” тиснемо “Select Image”, “Завантаження файлів”, “Обрати файли” і т.д. В полі “Розміщення розділу” виберемо “На всю ширину” і далі тиснемо “Done”. Тиснемо “Add Widget”, виберемо “Layout Builder”. Далі тиснемо “Edit”, потім тиснемо “Add Row”, виберемо 1 (стовбець), тиснемо “Insert”. Далі тиснемо “Add Row”, виберемо 2 (стовбця), тиснемо “Insert”. Робимо активним верхній ря-

док і тиснемо “Add Widget”, виберемо “Текст”. В рядку “Текст” тиснемо “Edit”, в полі “Назва”вводимо “УЗНАЙТЕ БІЛЬШЕ ПРО НАШІ КУРСИ І ПОСЛУГИ” і тиснемо “Done”. В рядку “Текст” наводимо на гаєчний ключ і тиснемо “Edit Row”, далі “Theme”, в полі “Верхній і нижній відступ” задаємо 0, в полі “Колір” обитаємо червоний (#dd3333) і тиснемо “Done”. В правому стовбці тиснемо “Edit”, в полі “Введіть свій заклик до дії” вводимо, наприклад, “Звертайтесь до нас за консультацією”, в полі “Посилання кнопки” вводимо http://localhost/myblog/contact-us/, в полі “Заголовок кнопки” вводимо “НАШІ КОНТАКТИ”. Далі тиснемо “Attributes”, в полі “CSS Styles” вводимо background-color: rgba(0,0,0,0.3); Тиснемо “Done”. В рядку з двох стовбців наводимо на гаєчний ключ і тиснемо “Edit Row”, далі “Theme”, в полі “Верхній і нижній відступ” задаємо 0, в полі “Колір” обитаємо червоний (#dd3333) і тиснемо “Done”. Тиснемо “Done”, потім тиснемо “Оновити”. Тепер можна переглянути наш оновлений сайт.

STEP11 Створення відео секції на головній сторінці. Переходимо до режиму редагування Головної сторінки. Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Нова секція з’явиться між попередніми. Наводимо курсор миші на значок поряд з гаєчним ключем і перетягуємо секцію вниз (після секції “Layout Builder”). Тиснемо на секцію, щоб зробити її активною. Тиснемо “Add Widget”, виберемо “Віджети теми Sydney”, “Sydney Video”. Наводимо на гаєчний ключ і тиснемо “Edit Row”, далі “Theme”, в полі “Верхній і нижній відступ” задаємо 30, в списку “Розміщення розділу” виберемо “На всю ширину” і тиснемо “Done”. Далі переходимо до редагування нашого віджету. Тиснемо “Edit”. В полі “Вставте посилання відео” вставимо посилання, наприклад, https://www.youtube.com/watch?v=wej0v-7deoQ. Тиснемо “Layout”, в полі “Padding” вводимо 20. Тиснемо “Done”. Тиснемо “Оновити”. Далі можна переглянути оновлену головну сторінку (з відео).

STEP12 Створення секції для відгуків. Переходимо до режиму редагування Головної сторінки. Тиснемо “Add Row”, виберемо 1 (стовбець) і тиснемо “Insert”. Нова секція з’явиться між попередніми. Наводимо курсор миші на значок поряд з гаєчним ключем і перетягуємо секцію вниз (після секції “Sydney Video”). Тиснемо на секцію, щоб зробити її активною. Тиснемо “Add Widget”, виберемо “Віджети теми Sydney”, “Sydney FP: Call to action”. Перед редагуванням віджету відредагуємо рядок. Тиснемо “Edit Row” далі “Theme”, в полі “Верхній і нижній відступ” задаємо 30, в полі “Фонове зображення” тиснемо “Select Image”, “Завантаження файлів”, “Обрати файли” і т.д. В списку “Розміщення розділу” виберемо “На всю ширину” і тиснемо “Done”. Тепер переходимо до редагування віджету “Sydney FP: Call to action”. Тиснемо “Edit”. В полі “Введіть свій заклик до дії” вводимо, наприклад, “НАШІ ВИПУСКНИКИ ГОТОВІ ПОДІЛИТИСЯ ВРАЖЕННЯМИ”, в полі “Посилання кнопки” вводимо http://localhost/myblog/success-stories, в полі “Заголовок кнопки” вводимо “ВІДГУКИ”. Ставимо галочку в пункті “Відображати кнопку врівень із текстом?”. Тиснемо “Layout”, в полі

“Padding” вводимо 0. Тиснемо “Done”. Тиснемо “Оновити”. Далі можна переглянути оновлену головну сторінку.

2.7 Лабораторна робота 7. Розробка сайту за допомогою фреймворку Laravel.

Завдання: Розробити в середовищі Laravel простий сайт, що працює з машинами (клас Car з полями make, model, produced_on). Передбачити перегляд, редагування, додавання та видалення інформації.

Теоретичні відомості, методичні вказівки та хід роботи.

Передумови
Комп’ютер, з’єднаний з Інтернет.
XAMPP 5.6.28 в якості локального сервера.
Laravel 5.2.
Пакетний менеджер Composer.

1) Інсталяція Laravel 5.2 на XAMMP

КРОК1. Встановлення локального сервера (XAMPP 5.6.28)
КРОК2. Встановлення пакетного менеджера Composer (завантажити з сайту https://getcomposer.org/Composer-Setup.exe). В ході інсталяції треба вказати місцезнаходження файлу php.exe. Якщо інсталювати на Обчислювальних Центрах ЗДІА, то вказати адресу proxy-сервера (http://192.168.0.222:3128).
КРОК3. Відкрийте термінал Windows (cmd.exe) перейдіть в каталог xampp/htdocs і подайте команду: `composer create-project laravel/laravel mysite "5.2.*"`. Після закінчення буде створено каталог `xammp/htdocs/mysite` з наступним вмістом (Рис.65):

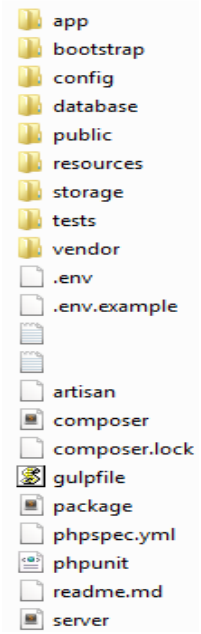


Рис.65. Структура каталогів застосунку
КРОК4. Активізуйте Apache і MySql з панелі управління XAMPP.

КРОК5. Наберіть в браузері: `localhost/mysite/public` (повинен з'явитись екран Laravel 5).

Наступні кроки 6,7,8,9 бажано виконати після налаштування бази даних застосунку (дивись далі).

КРОК6. Для подальшої зручності в роботі бажано сконфігурувати віртуальний хост в XAMMP. Для цього треба відкрити в редакторі (наприклад Notepad++) файл `C:\xampp\apache\conf\extra\httpd-vhosts.conf` (якщо у вас XAMMP на диску C:\) і додати наступні рядки в кінець файлу:

```
# VirtualHost for LARAVEL.DEV
   <VirtualHost  *:80>
            ServerAdmin webmaster@dummy-host2.laravel.dev
        DocumentRoot "C:/xampp/htdocs/mysite/public"
        ServerName  laravel.dev
        ErrorLog  "logs/laravel.dev-error.log"
        CustomLog  "logs/laravel.dev-access.log" common
   </VirtualHost>
```

Після цього наш Apache буде прослуховувати `laravel.dev` з'єднання.

КРОК7. Залишилось внести деякі зміни в системний файл `C:\Windows\System32\drivers\etc\hosts`. Додайте в файл hosts рядок:

```
127.0.0.1  laravel.dev
```

КРОК8. (Тільки на Обчислювальних Центрах ЗДІА). Включіть адресу `laravel.dev` в виключення проксі-сервера в налаштуваннях вашого браузера.

КРОК9. Наберіть в браузері: `laravel.dev` (повинен з'явитись екран Laravel 5).

2) Налаштування фреймворку.

Відкрийте файл .env і відредагуйте деякі рядки:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=mylaravel
DB_USERNAME=root
DB_PASSWORD=
```

Увійдіть в phpMyadmin (http://localhost/phpmyadmin) і створіть базу даних mylaravel (з utf8_general_ci).

3) Побудова простого застосунку. Для демонстрації можливостей Laravel побудуємо застосунок, що працює з машинами.

Модель Car

Laravel має вбудований в інтерфейс командного рядка ремісник (Artisan), який надає вам багато корисних команд для побудови додатка.

КРОК1. Відкрийте термінал Windows (cmd.exe) перейдіть в каталог застосунку (xampp/htdocs/mysite) і подайте команду для генерації нової моделі Car:

```
php artisan make:model Car --migration
```

Всі моделі зберігаються в каталозі app, так що попередня команда згенерує файл app/Car.php з наступним кодом:

```
<?php
```

```
namespace App;
use Illuminate\Database\Eloquent\Model;
class Car extends Model
{
     //
}
```

Завдяки вбудованій функціональності моделі Laravel, створивши порожній клас моделі, Laravel буде вважати, що ця модель пов'язана з таблицею бази даних з ім'ям cars.

Завдяки прапорцю --migration при створенні моделі, Laravel також генерує міграційний файл бази даних для створення таблиці бази даних (https://laravel.com/docs/5.2/migrations). Файл міграції database/migrations/[мітка часу] _create_cars_table.php містить наступний код:

```
. . . . . .
public function up() {
Schema::create('cars', function (Blueprint $table) {
     $table->increments('id');
     $table->timestamps();
}); }
. . . . . .
```

КРОК2. Додайте три додаткових стовпців, які будуть зберігати характеристики автомобілів (виробника, модель та дату виробництва):

```
$table->string('make');
$table->string('model');
$table->date('produced_on');
```

КРОК3. Подайте Artisan-команду для запуску міграції, що створить таблицю cars:

```
php artisan migrate
```

КРОК4. Додайте вручну декілька записів в таблицю автомобілів (cars) бази даних нашого застосунку (mylaravel). Для цього наберіть в браузері

localhost/phpmyadmin і т.д.

Контролер

В Laravel, тип об'єкта - такий, як Car, в даному випадку - називають в якості ресурсу (resource).

КРОК5. Створіть контролер ресурсів (resource controller) - контролер для обробки всіх запитів, пов'язаних з ресурсом Car - за допомогою іншої простої команди Artisan:

```
php artisan make:controller CarController
```

Це створить файл app/Http/Controllers/CarController.php контролера з відповідним кодом.

Маршрути (Routes)

На попередньому кроці Laravel автоматично згенерував контролер з усіма типовими CRUD операціями. Тепер нам просто потрібно визначити маршрути, щоб зв'язати URL-адреси з усіма цими діями контролера. Знову ж таки, оскільки це такий загальний сценарій, ви можете визначити єдиний маршрут (resource route), який створює маршрути для всіх дій контролера ресурсів.

КРОК6. У файлі конфігурації маршрутів – app/Http/routes.php —додайте наступне визначення маршруту для ресурсу Car:

```
Route::resource('cars', 'CarController');
```

Це єдине визначення маршруту буде визначати всі маршрути, пов'язані з нашим ресурсом Car (Рис.66):

Request Type	Path	Action	Route Name
GET	/cars	index	cars.index
GET	/cars/create	create	cars.create
POST	/cars	store	cars.store
GET	/cars/{car}	show	cars.show
GET	/cars/{car}/edit	edit	cars.edit
PUT/PATCH	/cars/{car}	update	cars.update
DELETE	/cars/{car}	destroy	cars.destroy

Рис.66. Маршрути застосунку

Тепер, наприклад, давайте здійснимо реалізацію сторінки Show Car (машини з конкретним id).

Метод дії show контролера

Згідно з попередньою таблицею, сторінка Show Car буде доступною за адресою: http://app.url/cars/{car} (в нашому випадку app.url – це laravel.dev). В цьому випадку {car} буде, насправді, ідентифікатором id об'єкта автомобіля в базі даних.

Так, наприклад, URL, щоб переглянути автомобіль з ідентифікатором 1 буде http://app.url/cars/1.

З метою реалізації сторінки Show Car, в show дії контролера, нам необхідно:

- Використати модель Car, щоб отримати відповідний об'єкт автомобіля з бази даних.
- Завантажити вид (view) для сторінки Show Car, і передати йому об'єкт Car, витягнутий з бази даних.

КРОК7. По-перше, для того, щоб отримати доступ до моделі Car в контролері, нам потрібно додати новий вираз вище класу контролера (файл app/Http/Controllers/CarController.php):

```
use App\Car;
```

КРОК8. Потім ми можемо завершити метод дії show наступним кодом:

```
public function show($id)
{
    $car = Car::find($id);
    return view('cars.show', array('car' => $car));
}
```

Вид (View)

Файли видів Laravel всі зберігаються в папці resources/views. І вони можуть бути організовані в підпапках в цьому каталозі. У попередньому кроці, ми послались на вид з назвою cars.show. Це говорить Laravel шукати файл виду, розташо-

ваний в підпапці cars (всередині головного каталогу видів resources/views) під назвою show.blade.php.

Файли видів Laravel використовують шаблонний движок Blade, і, отже, називаються *.blade.php

КРОК9. Створіть каталог cars всередині каталогу resources/views

КРОК10. Створіть файл resources/views/cars/show.blade.php з наступним кодом:

```
<!DOCTYPE html>
<html>
<head> <title>Car {{ $car->id }}</title> </head>
<body>
<h1>Car {{ $car->id }}</h1>
    <ul>
        <li>Make: {{ $car->make }}</li>
        <li>Model: {{ $car->model }}</li>
        <li>Produced on: {{ $car->produced_on }}</li>
    </ul>
</body>
</html>
```

Метод дії index контролера

В цьому методі дії необхідно вивести всі автомобілі.

КРОК11. В файл app/Http/Controllers/CarController.php додайте метод дії index з наступним вмістом:

```
public function index()
 {
    $cars = Car::orderBy('id', 'desc')->paginate();
    return view('cars.index', compact('cars'));
 }
```

Вид для перегляду всіх автомобілів.

КРОК12. Створіть файл resources/views/cars/index.blade.php з наступним кодом:

```
@foreach($cars as $car)
<car>
    <h1>Car {{ $car->id }}</h1>
   <ul>
     <li>Make: {{ $car->make }}</li>
     <li>Model: {{ $car->model }}</li>
     <li>Produced on: {{ $car->produced_on }}</li>
   </ul>
</car>
<hr>
@endforeach
```

Виконайте кроки 6 і 7 по конфігуруванню віртуального хоста laravel.dev (слайди 7, 8, 9 даної презентації).

Наберіть в браузері: http://laravel.dev/cars/. Повинен з'явитись список всіх машин, що внесені в базу даних.

Наберіть в браузері: `http://laravel.dev/cars/1`

(повинна з'явитись інформація про першу машину)

КРОК13. Подальше вдосконалення застосунку.

Вивчіть документацію, потрібну для додавання в застосунок можливості додавання нових машин, редагування та видалення інформації.

2.8Лабораторна робота 8. Розробка сайту за технологією Node+Express.

Завдання: В середовищі фреймворку Express розробити застосунок “PC Repair” з наступним інтерфейсом (Рис.67):

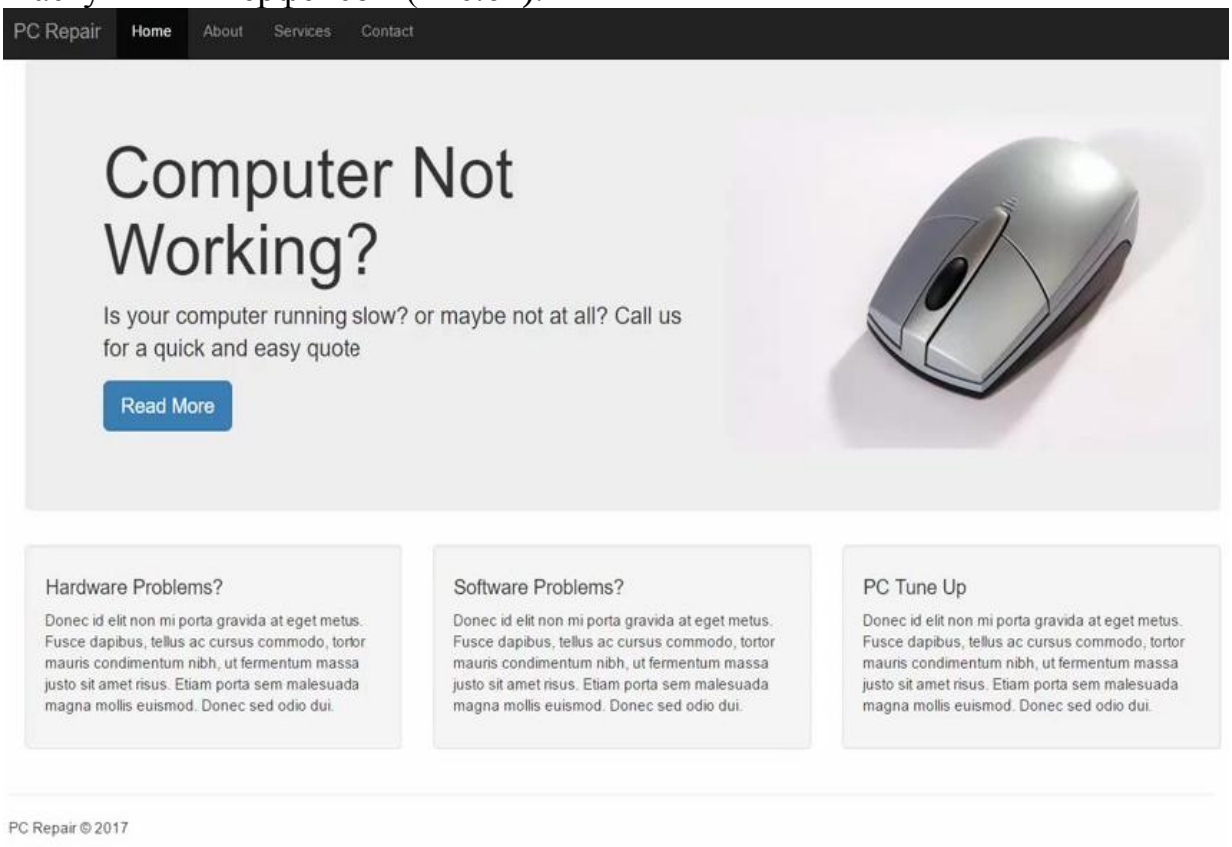


Рис.67. Інтерфейс застосунку

Хід роботи

1. Інсталювати Node.js (https://nodejs.org/)
2. Відкрити термінал (cmd.exe), перейти в каталог проекту.
3. Інсталювати Express
    - npm install express -g
4. Інсталювати express-generator
    - npm install express-generator -g
5. Згенерувати систему директорій проекту
    - express pcrepair --pug
6. Перейти в каталог проекту: cd pcrepair
7. Створити файл package.json
    - npm install

8. Завантажити директорію проекту (pcrepair) в редактор Sublime Text або Visual Studio Code та відкрити файл package.json, звернути увагу на рядок “start”: “node ./bin/www”
9. Запустити проект
    - npm start
    - В вікні браузера набрати localhost:3000
10. В редакторі відкрити файл app.js та уважно його вивчити.
11. Переглянути файли каталогу routes (index.js, users.js). Набрати в браузері localhost:3000/users і проаналізувати відповідь.
12. Переглянути файли каталогу views (error.pug, index.pug, layout.pug).
13. Інсталювати nodemon для автоматичного рестарту:
    - npm install -g nodemon
14. Запустити nodemon: nodemon
15. Підготуємо шаблон титульної сторінки за допомогою Bootstrap. Заходимо на http://getbootstrap.com/, тиснемо Download і на наступній сторінці ще раз Download. Видобуваємо файли архіву до допоміжної папки.
16. В папку public/stylesheets нашого проекту копіюємо файл bootstrap.css.
17. В файл views/layouts.pug після 5-го рядка додаємо рядок “link(rel=‘stylesheet’, href=‘/stylesheets/bootstrap.css’)
18. Оновлюємо сторінку в браузері.
19. На сайті Bootstrap вибираємо в випадаючому списку v3.3.7, потім внизу пункт Examples, тиснемо на Starter template, переглядаємо вихідний код сторінки. Копіюємо в секції body тег навігації ‘<nav . . . </nav>’
20. Заходимо на сайт Jade Converter (http://html2jade.org/), вставляємо зліва скопійований фрагмент, в правій панелі отримуємо результат.
21. Копіюємо конвертований фрагмент навігації і вставляємо в секцію body файлу layout.pug, як показано на наступному слайді.

Рис.68. Файл layout.pug

22. Оновлюємо сторінку в браузері щоб побачити навігаційне меню.

23. Відредагуємо файл layout.pug. Змінемо “Project Name” на “PC Repair”, 20-й рядок на “a(href=‘/’) Home”, 22-й рядок на “a(href=‘/about’) About”. Після 22-го рядка вставимо рядок “li” та рядок “a(href=‘/services’) Services”. Підкорегуємо бувший 24-й рядок: “a(href=‘/contact’) Contact”. 19-й рядок представимо у вигляді: “li(class=title==‘Home’ ? ‘active’ : undefined)”.
24. Корегуємо файл layout.pug.

```
ul.nav.navbar-nav
  li(class=title=='Home' ? 'active' : undefined)
    a(href='/') Home
  li(class=title=='About' ? 'active' : undefined)
    a(href='/about') About
  li(class=title=='Services' ? 'active' : undefined)
    a(href='/services') Services
  li(class=title=='Contact' ? 'active' : undefined)
    a(href='/contact') Contact
```

Рис.69. Корегування файлу layout.pug

Перевіримо зміни в браузері.

25. Попрацюємо над головною сторінкою. На сайті http://html2jade.org в лівій колонці наберемо текст, подібний наступному

```
<div class="row">
  <div class="col-md-4">
    <h2>Hardware problems?</h2>
    <p>Donec id elit non mi porta gravida at eget metus </p>
  </div>
  <div class="col-md-4">
    <h2>Software problems?</h2>
    <p>Donec id elit non mi porta gravida at eget metus </p>
  </div>
  <div class="col-md-4">
    <h2>PC Tune Up</h2>
    <p>Donec id elit non mi porta gravida at eget metus </p>
  </div>
</div>
```

Рис. 70. Розмітка для файлу index.pug

26. Копіюємо праву колонку і вставляємо в файл index.pug

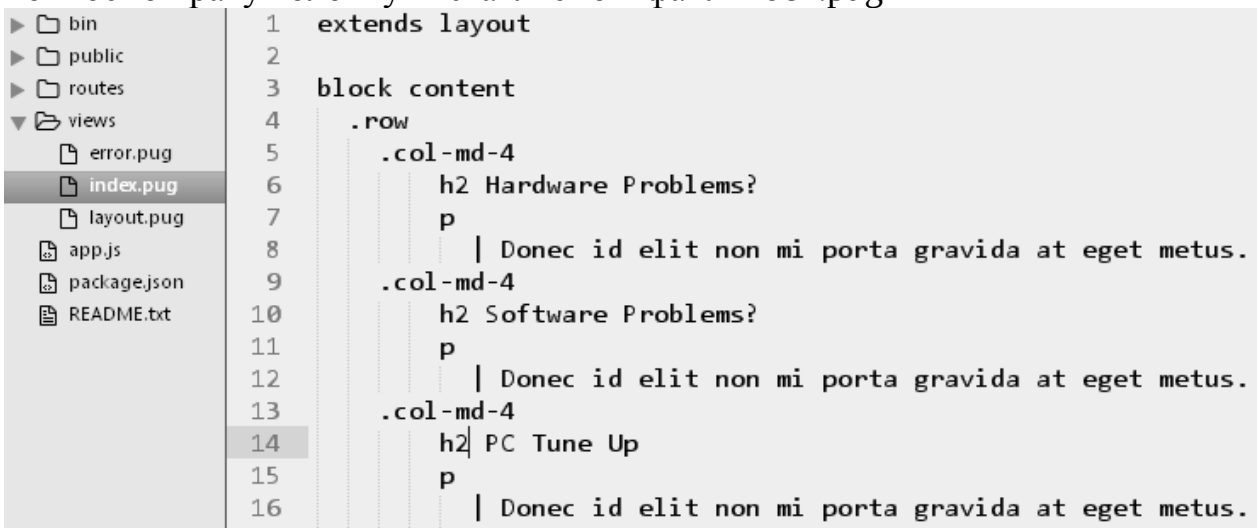


Рис.71. Файл index.pug

27. Оновлюємо браузер і бачимо на головній сторінці три колонки.
28. Трохи підкорегуємо головну сторінку (файл index.pug).

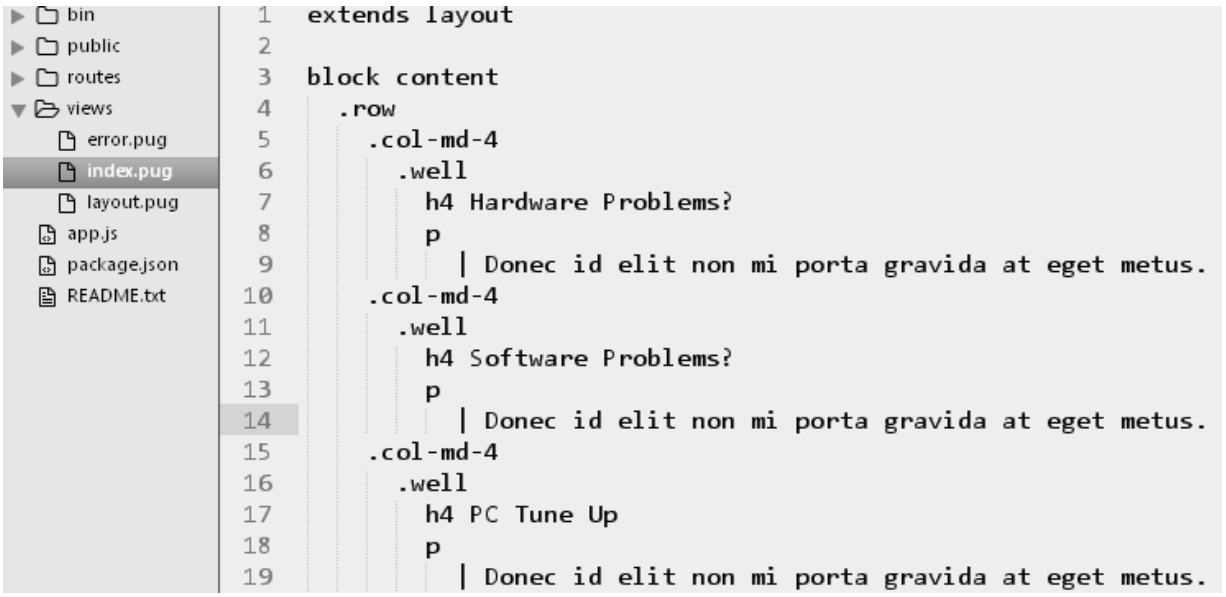


Рис.72. Корегування файлу index.pug

29. Оновлюємо браузер.
30. Для покращення вигляду вставляємо в файл layout.pug перед останнім рядком рядок ".container", оновлюємо браузер щоб побачити результат.
31. Вставимо на головну (index.pug) сторінку джамботрон.

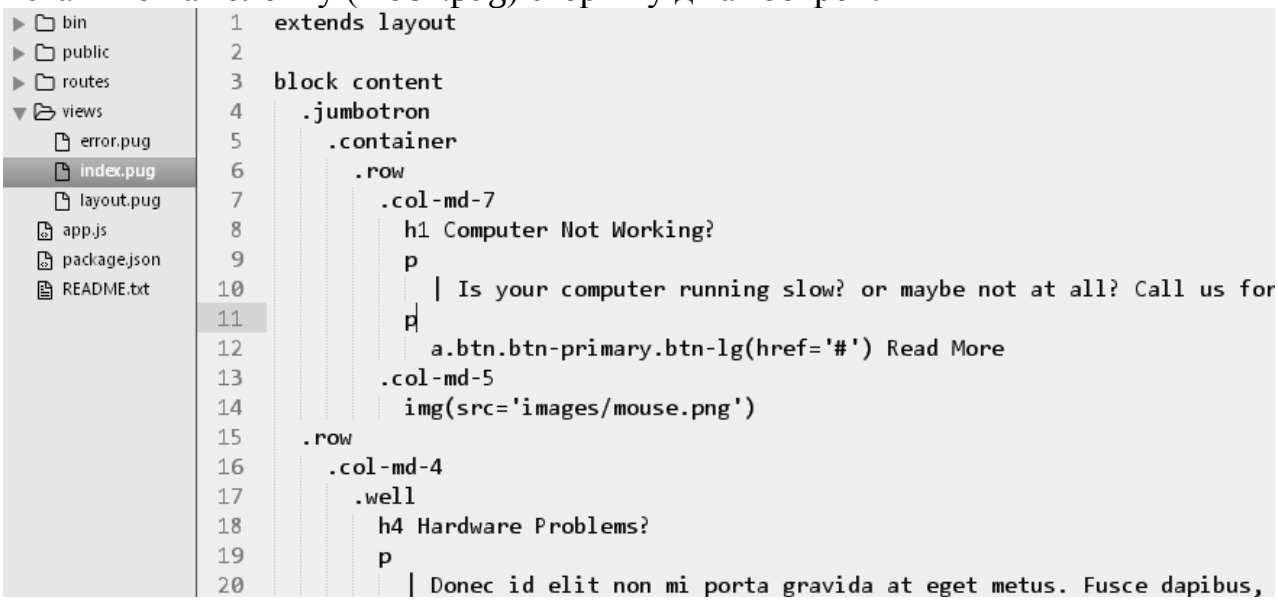


Рис.73. Вставка джамботрону

32. Всталяємо mouse.png в каталог public/images, оновлюємо браузер щоб побачити результат.
33. Додамо до шаблону сайту (layout.pug) footer і оновимо браузер.

```
            span.sr-only Toggle navigation
            span.icon-bar
            span.icon-bar
            span.icon-bar
          a.navbar-brand(href='#') PC Repair
        #navbar.collapse.navbar-collapse
          ul.nav.navbar-nav
            li(class=title=='Home' ? 'active' : undefined)
              a(href='/') Home
            li(class=title=='About' ? 'active' : undefined)
              a(href='/about') About
            li(class=title=='Services' ? 'active' : undefined)
              a(href='/services') Services
            li(class=title=='Contact' ? 'active' : undefined)
              a(href='/contact') Contact
    .container
      block content

    hr
    footer
      p PC Repair &copy; 2017
```

Рис.74. Додавання footer

34. Попрацюємо над пунктом About. В директорії routes створюємо файл about.js.

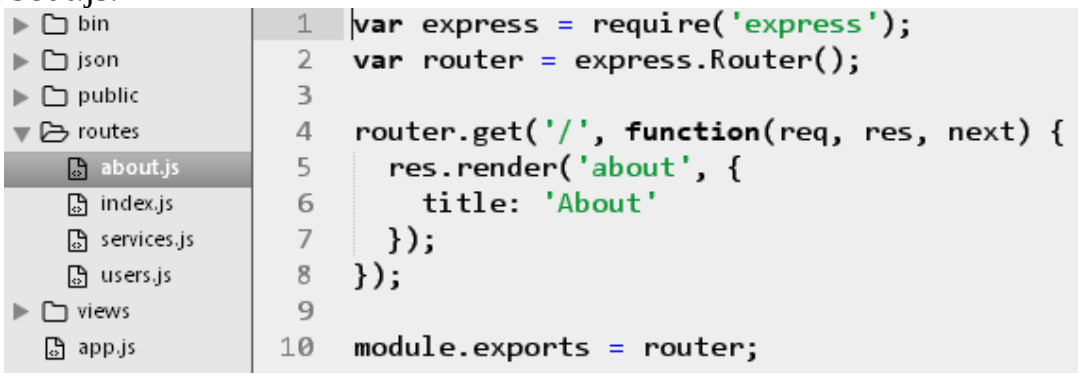


Рис.75. Файл about.js

35. Редагуємо файл app.js:
    - 9-й рядок: var about = require('./routes/about');
    - 26-й рядок: app.use('/about', about);

36. В директорії views створюємо файл about.pug і оновлюємо браузер. Пункт About працює.

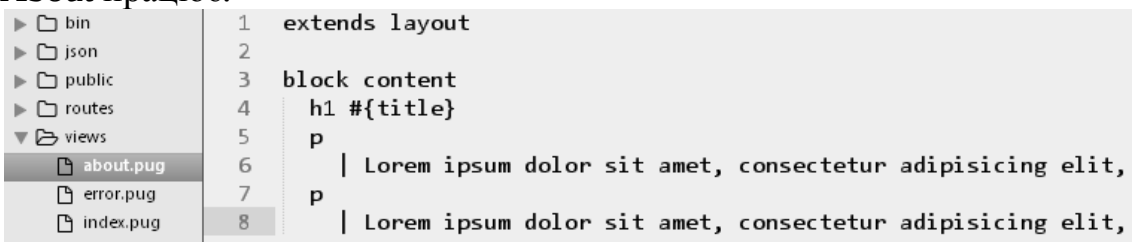


Рис.76. Файл about.pug

37. Попрацюємо над сторінкою Services. В директорії routes створюємо файл services.js і копіюємо в нього вміст файлу about.js.

38. В кореневій директорії нашого проекту створюємо каталог json і в ньому створюємо файл services.json. Вміст файлу services.json:

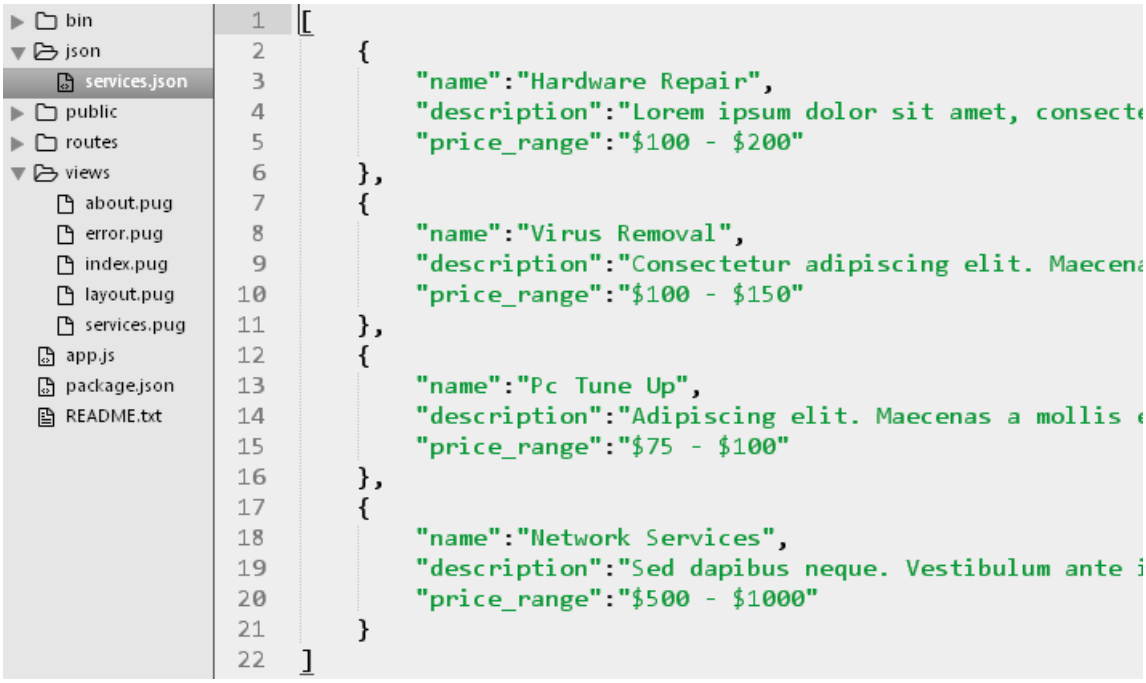


Рис.77. Файл services.json

39. Корегуємо файл services.js.

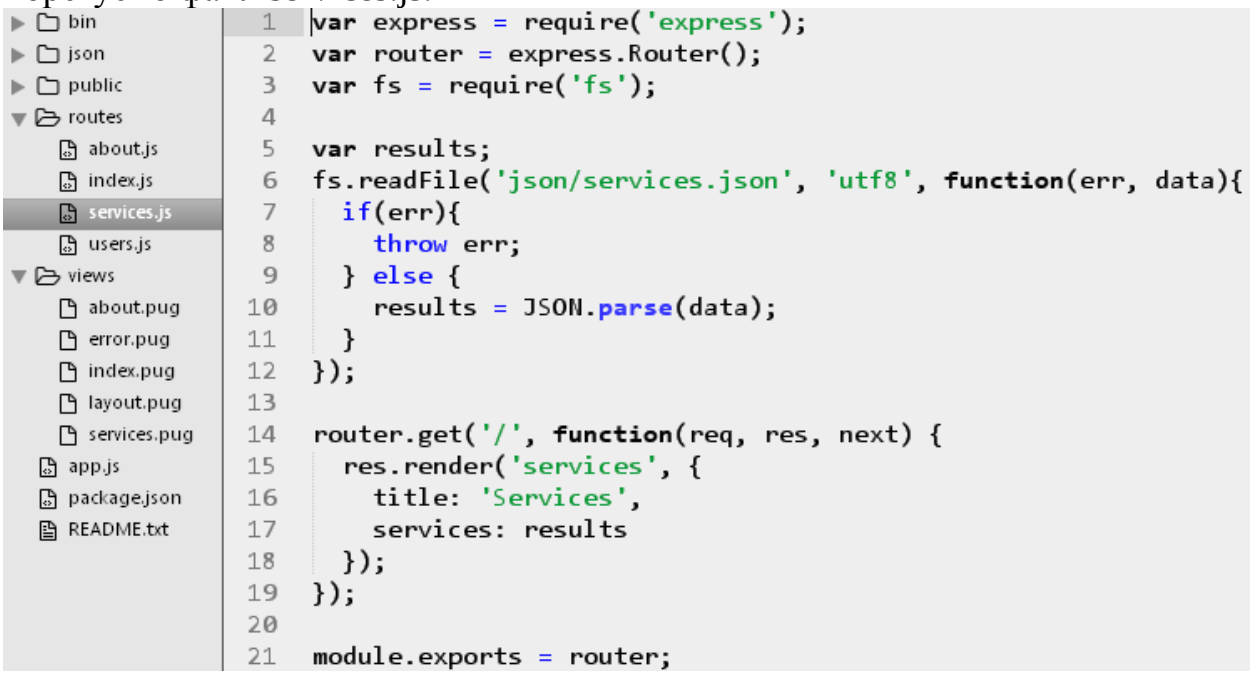


Рис.78. Файл services.js

40. Повертаємось до файлу app.js.
    - Після 9-го рядка вставимо рядок var services = re-quire(‘./routes/services’);
    - Після 27-го рядка вставимо рядок app.use(‘/services’, services);
41. В директорії views створюємо файл services.pug. В браузері вибираємо пункт меню Services щоб побачити результат.

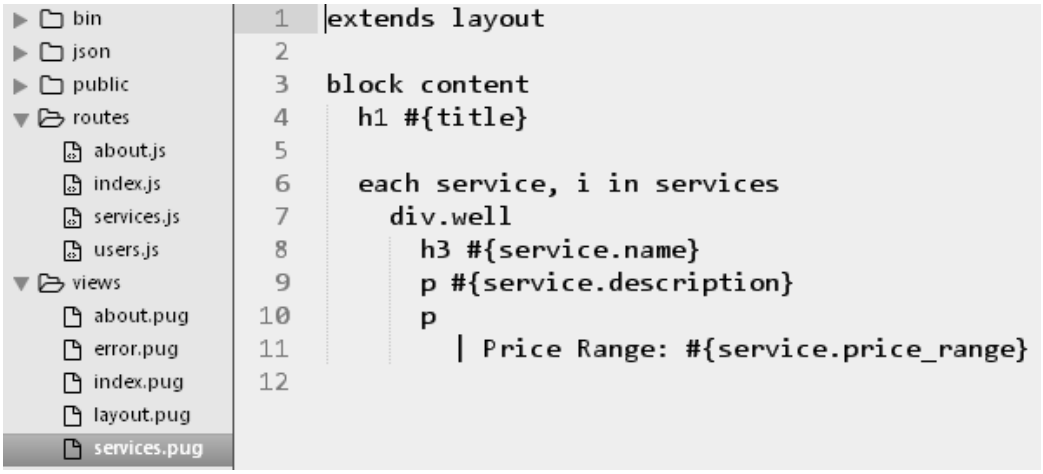


Рис.79. Файл services.pug

42. В директорії routes створюємо файл contact.js. Копіюємо вміст файлу index.js в contact.js. В 6-ому рядку змінюємо слово index на contact, а в 7-ому рядку слово Home на Contact.
43. Заходимо в файл app.js. Після 10-го рядка додаємо рядок var contact= require('./routes/contact'); Після 29-го рядка вставимо рядок app.use('/contact', contact);
44. В директорії views створюємо файл contact.pug.:

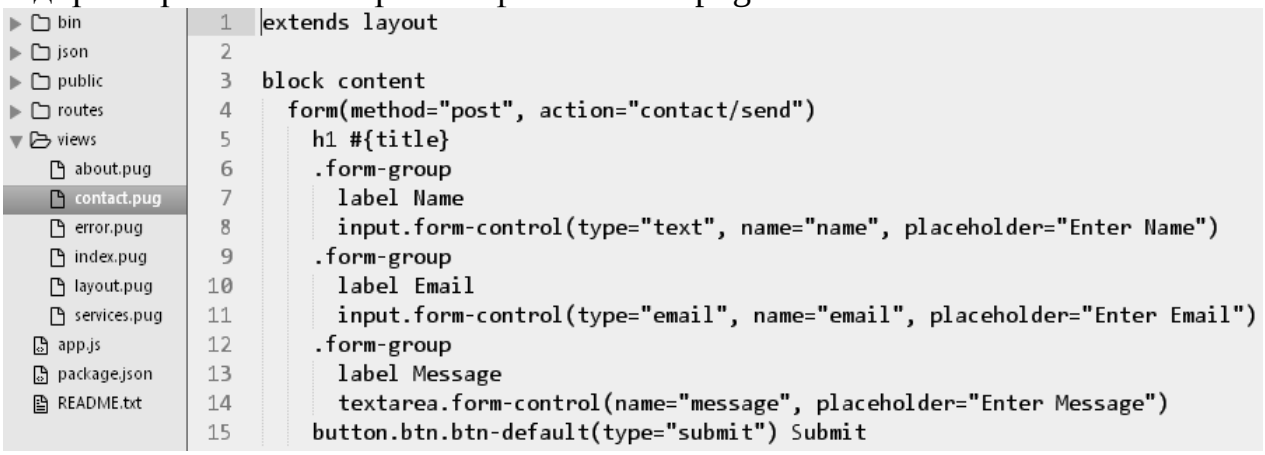


Рис.80. Вміст файлу contact.pug

45. В браузері вибираємо пункт меню Contact щоб побачити результат.
46. Для відправки повідомлень будемо використовувати Nodemailer (https://nodemailer.com/). Подаємо в консолі команду npm install nodemailer –save . Після цього подаємо команду nodemon
47. Повертаємось до файлу contact.js. Вивчемо зразок цього файлу, наведений нище, та документацію до Nodemailer. Внесемо корективи до цього файлу, налаштувавши вашу власну електронну адресу. Перевіримо в браузері роботу пункту меню Contact.

```
// File contact.js
var express = require('express');
var router = express.Router();
var nodemailer = require('nodemailer');
router.get('/', function(req, res, next) {
  res.render('contact', {
    title: 'Contact'
```

```
  });
});
// Send Email
// Send Email
router.post('/send', function(req, res, next){
    var transporter = nodemailer.createTransport({
      service: 'Gmail',
      auth: {
        user: 'YOUREMAIL',
        pass: 'YOURPASS'
      }
    });
    var mailOptions = {
      from: '"YOURNAME" <YOUREMAIL>',
      to: 'TOEMAIL',
      subject: 'Hello from PCRepair',
      text: 'You have a submission from... Name:
'+req.body.name+' Email: '+req.body.email+' Message:
'+req.body.message,
      html: '<p>You have a submission from...</p> <ul><li>Name:
'+req.body.name+'</li><li> Email: '+req.body.email+'</li><li>
Message: '+req.body.message+'</li></ul>'
    }
    transporter.sendMail(mailOptions, function(error, info){
      if(error){
        return console.log(error);
      }
      console.log('Message Sent: '+ info.response);
      res.redirect('/');
    });
});
module.exports = router;
```

2.9 Лабораторна робота 9. Розробка сайту за технологією Python+Django.

Завдання: За допомогою фреймворку Django розробити сайт простого блогу з наступними можливостями:

- Користувач-адміністратор може створювати, додавати, видаляти, редагувати записи (повідомлення) блогу.
- Звичайним користувач може переглядати список повідомлень блогу та детальну інформацію по кожному повідомленню.
- Передбачити розбиття на сторінки списку повідомлень блогу.
- На сторінці списку повідомлень розмістити посилання на RSS-канал, що відображає 5 останніх повідомлень блогу.

Теоретичні відомості, методичні вказівки та хід роботи.

Установка Django в Windows

1) Інсталяція Python (https://www.python.org/). На березень 2018 маємо реліз Python 3.6.4. Зверніть увагу на другу сторінку "Customize" майстера

установки, вам потрібно пролистати вниз і вибрати опцію "Add python.exe to the Path" (додати python.exe в системну змінну Path).

2) Створення віртуального середовища. Для роботи з Django необхідно спочатку створити віртуальне середовище для роботи. Віртуальне середовище являє собою підрозділ системи, в якому ви можете встановлювати пакети в ізоляції від решти пакетів Python. Відділення бібліотек одного проекту від інших проектів принесе користь при розгортанні вашого застосунку на сервері. Створіть для проекту новий каталог, наприклад з ім'ям D:\DjangoSites, перейдіть в цей каталог в термінальному режимі і створіть віртуальне середовище. Якщо ви працюєте в Python 3, то зможете створити віртуальне середовище наступною командою:
```
python -m venv djangosites_env
```
Команда запускає модуль venv і використовує його для створення віртуального середовища з ім'ям djangosites_env.

3) Активізація віртуального середовища. Після того, як віртуальне середовище буде створене, його необхідно активізувати наступною командою (для Windows):
```
djangosites_env\Scripts\activate
```
Команда запускає сценарій activate з каталогу djangosites_env\Scripts. Коли середовище активізується, його ім'я виводиться в круглих дужках. Тепер ви можете встановлювати пакети в середовищі і використовувати ті пакети, що були встановлені раніше. Пакети, встановлені в djangosites_env, будуть доступні тільки в той час, поки середовище залишається активним. Щоб завершити використання віртуального середовища, введіть команду deactivate.

4) Установка Django. Після того як ви створили своє віртуальне середовище і активізували його, встановіть Django:
```
pip install django==1.11.5
```
Так як ви працюєте в віртуальному середовищі, ця команда виглядає однаково в усіх системах. Версію Django можна не вказувати, тоді буде встановлена остання версія. (Якщо ви отримуєте помилку коли викликаєте pip на Windows, перевірте чи шлях до вашого проекту не містить пробілів, наголосів чи спеціальних символів).

5) Створення проекту mysite в Django. Не виходячи з активного віртуального середовища, введіть наступні команди для створення нового проекту (не забудьте про крапку . в кінці):
   - **`django-admin.py startproject mysite .`**
   - **`dir`** щоб побачити вміст каталогу D:\DjangoSites (там будуть каталоги djangosites_env і mysite та файл manage.py)
   - **`dir mysite`** щоб побачити вміст каталогу D:\DjangoSites\mysite (там будуть файли __init__.py, settings.py, urls.py, wsgi.py).

Файл manage.py – це коротка утиліта командного рядка, яка отримує команди і передає їх відповідній частині Django для виконання. Ми використовуємо ці

команди для управління такими завданнями, як робота з базами даних і запуск серверів.

В каталозі mysite знаходяться чотири файли, найважливішими з яких є файли settings.py, urls.py і wsgi.py. Файл settings.py визначає те, як Django взаємодіє з вашою системою і управляє вашим проектом. Ми змінимо деякі з існуючих налаштувань і додамо кілька нових налаштувань в ході розробки проекту. Файл urls.py повідомляє Django, які сторінки слід будувати у відповідь на запити браузера. Файл wsgi.py допомагає Django надавати створені файли (ім'я файлу є скороченням від «Web Server Gateway Interface»).

Оскільки ми використовуємо Python 3.5, який постачається з вбудованою базою SQLite, ми можемо використовувати цю базу даних у нашому проекті для розробки. Якщо ви плануєте розгортати вашу програму у виробничому середовищі, ви повинні використовувати розширену базу даних, таку як MySQL, Oracle або PostgreSQL. Згенерований файл settings.py містить базову конфігурацію для використання бази даних SQLite та список програм Django, які за замовчуванням працюють у вашому проекті. Нам потрібно створити таблиці в базі даних для початкового застосування.

6) Створення бази даних. Подамо команду
   `cd mysite`
   шоб перейти в каталог mysite. Щоб створити базу даних для проекту mysite, введіть наступну команду
   `python manage.py migrate`
   В терміналі ви побачите процес створення бази даних. Кожна зміна бази даних називається міграцією. Перше виконання команди migrate наказує Django перевірити, що база даних відповідає поточному стану проекту. Під час першого виконання цієї команди в новому проекті з використанням SQLite Django створює нову базу даних за нас. Django повідомляє про створення таблиць бази даних, необхідних для зберігання інформації, використовуваної в проекті, а потім перевіряє, що структура бази даних відповідає поточному коду. Django створює файл з ім'ям db.sqlite3.

7) Запуск Django Development Server.
   Переконаємося в тому, що проект був створений правильно. Введіть команду runserver:
   `python manage.py runserver`
   Django запускає сервер, щоб ви могли переглянути проект в своїй системі і перевірити, як він працює. Коли ви запитуєте сторінку, вводячи URL в браузері, сервер Django відповідає на запит; для цього він будує відповідну сторінку і відправляє сторінку браузеру.
   Тепер відкрийте браузер і введіть URL
   `http://localhost:8000/`
   - або http://127.0.0.1:8000/, якщо перша адреса не працює. Ви побачите щось схоже на Рис.81 - сторінку, яку створює Django, щоб повідомити вам, що все поки працює правильно.

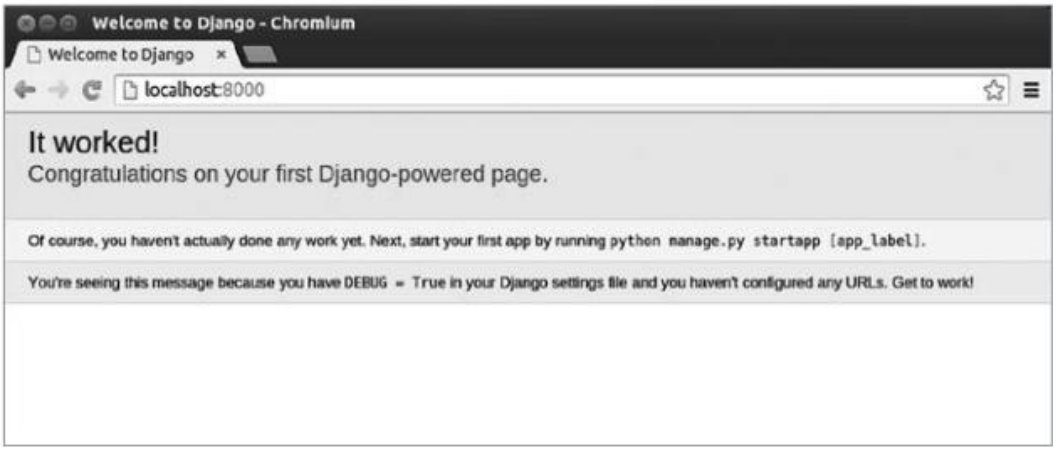


Рис.81. Перша сторінка Django

Ще не завершуйте роботу сервера (але, коли ви захочете перервати його, це можна зробити натисканням клавіш Ctrl + C).

8) Початок роботи над додатком blog. Створення моделей.

Проект Django є групою окремих додатків, спільна робота яких забезпечує роботу проекту в цілому. Поки ми створимо один додаток blog, який буде виконувати більшу частину роботи в нашому проекті.

До цього моменту команда runserver повинна продовжувати роботу в термінальному вікні, яке ви відкрили раніше. Відкрийте нове термінальне вікно (або вкладку) і перейдіть в каталог, що містить manage.py. Активізуйте віртуальне середовище і виконайте команду startapp:

```
python manage.py startapp blog
```

Команда startapp ім'я_додатку наказує Django створити інфраструктуру, необхідну для побудови додатку. Заглянувши зараз в каталог проекту, ви знайдете в ньому новий підкаталог з ім'ям blog. Відкрийте цей каталог, щоб побачити, які файли були створені Django. Найважливіші файли в цьому каталозі - **models.py**, **admin.py** і **views.py**.

Ми скористаємося файлом models.py для визначення даних, якими потрібно управляти в нашому додатку.

admin.py - Тут ви реєструєте моделі, щоб включити їх в сайт адміністрування Django.

views.py - логіка вашого додатку. Кожне представлення отримує http-запит, обробляє його та повертає відповідь.

Зараз ми створимо схему даних для нашого блогу. Відкриємо каталог проекту в редакторі Sublime Text і відкриємо файл models.py. Модель - це клас Python (підклас django.db.models.Models), в якому кожен атрибут представляє поле бази даних. Django створить таблицю для кожної моделі, яка визначена у файлі models.py. Коли ви створюєте модель, Django пропонує вам практичний API для легкого здійснення запитів до вашої бази даних.

Запишіть у файл blog/**models.py** наступний код:

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
```

```
# Our Post Model

class Post(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Draft'),
        ('published', 'Published'),
    )
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250,
unique_for_date='publish')
    author = models.ForeignKey(User, related_name='blog_posts')
    body = models.TextField()
    publish = models.DateTimeField(default=timezone.now)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES,
default='draft')

    class Meta:
        ordering = ('-publish',)

    def __str__(self):
        return self.title
```

Це наша основна модель для нашого блогу, давайте розглянемо поля, які ми щойно визначили для цієї моделі:

title - це поле для заголовка повідомлення. Це поле CharField, яке перетворюється на varchar у базі даних SQL.

slug - це поле, призначене для використання в URL-адресах. Slug має коротке позначення, містять лише літери, цифри, підкреслення чи дефіс. Ми будемо використовувати це поле slug для створення красивих, дружніх до SEO сторінок для наших публікацій в блозі. Ми додали параметри unique_for_date для цього поля, щоб ми могли створювати URL-адреси для публікації, використовуючи дату та slug публікації. Django дозволить запобігти декілька постів, що мають однакові slug та дату.

author - автор, це поле є зовнішнім ключем. Це поле визначає відносини багато до одного, ми повідомляємо Django, що кожен пост написаний користувачем, і користувач може написати кілька повідомлень. У цьому випадку ми покладаємось на модель користувача системи автентифікації Django.

body – це тіло повідомлення.

publish – це поле date time покаже, коли повідомлення було опубліковане. Ми використовуємо метод Django timezone.now як значення за замовчуванням.

created - це поле date time вказує, коли було створено повідомлення. Оскільки ми використовуємо auto_now_add, дата буде автоматично збережена при створенні об'єкта.

updated - цей поле date time показує, коли останній раз був оновлений певний пост. Оскільки ми тут використовуємо auto_now, дата буде автоматично оновлена при збереженні об'єкта.

status - це поле, яке показує статус публікації. Ми використовуємо параметри вибору (STATUS_CHOICES), тому значення цього поля можна встановити лише на один із зазначених варіантів.

Як ви бачите, Django поставляється з різними типами полів, які можуть бути використані для визначення ваших моделей. Ви можете знайти все про різні поля Django в https://docs.djangoproject.com/en/1.11/ref/models/fields/ .

Клас Meta містить метадані. Ми доручаємо Django сортувати результати за полями публікації в порядку зменшення за замовчуванням, коли ми запитуємо дані. Ми вказуємо спадний порядок, використовуючи знак мінус ('-publish',).

Django викликає метод __str __ () для виведення простого представлення моделі. Ми написали реалізацію __str __ (), яка повертає рядок, що зберігається в атрибуті title.

9) Оскільки ми маємо справу з датою, нам потрібно встановити модуль pytz. Цей модуль забезпечить визначення часового поясу для Python, і він потрібен, щоб SQLite працювала з датами. Отже заходимо в термінал і подаємо команду
pip install pytz

10) Активізація моделей. Щоб використовувати моделі, необхідно наказати Django включити додаток blog в спільний проект. Відкриємо файл settings.py, знайдемо секцію INSTALLED_APPS = () і додамо додаток blog. Ця секція буде виглядати так:
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
]
```
Тепер Django знає, що наш додаток blog активний для цього проекту і зможе самостійно переглядати його моделі. Тепер ми будемо створювати і застосовувати міграції. Давайте створимо в базі даних таблицю для нашої моделі Post.

11) Впевнившись, що в консолі активною є коренева директорія нашого проекту (тобто доступний файл manage.py), подаємо команду
python manage.py makemigrations blog
Django створив файл 0001_initial.py всередині каталогу migrations нашого додатку blog. Ви можете відкрити цей файл, щоб побачити, як виглядає міграція. Виконайте наступну команду, щоб перевірити код:
python manage.py sqlmigrate blog 0001
Для синхронізації нашої бази даних з моделлю Post виконайте таку команду:
python manage.py migrate
База даних тепер відображає поточний стан нашої моделі блогу.

12) Адміністративний сайт Django.

Django дозволяє легко працювати з моделями, визначеними для додатка, через адміністративний сайт. Цей сайт використовується адміністраторами сайту, а не рядовими користувачами. У цьому розділі ми створимо адміністративний сайт і використаємо його для додавання деяких тем через модель Post.

Спочатку ми створимо суперкористувача, а потім додамо модель Post на сайт адміністрації, далі ми налаштуємо те, як моделі відображаються. Щоб створити суперкористувача, запустіть таку команду:

```
python manage.py createsuperuser
```

Ваш термінал попросить вас ввести ім'я користувача, адресу електронної пошти та пароль. Ви можете ввести: admin, admin@example.com і залишити поля для пароля пустими.

Тепер ми розглянемо сайт адміністрування Django. Запустіть сервер розробника, запустивши таку команду:

```
python manage.py runserver
```

У своєму браузері перейдіть за цим посиланням:

http://localhost:8000/admin/

Ви повинні побачити екран входу адміністратора, як показано нижче (Рис.82):

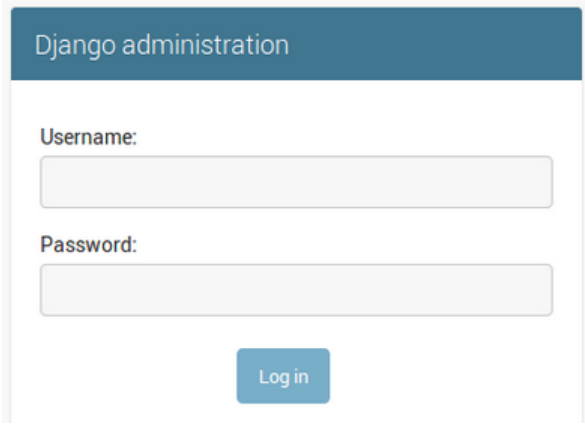


Рис.82. Форма входу адміністратора

Після введення даних ви побачите сторінку адміністратора, як показано тут:

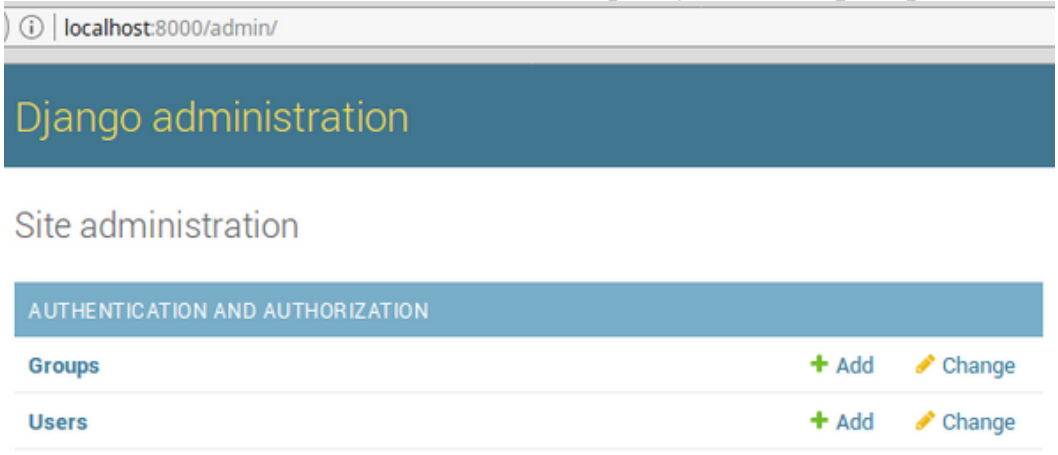


Рис. 83. Сторінка адміністратора

13) Реєстрація моделі на адміністративному сайті.

Django додає деякі моделі (наприклад, User і Group) на адміністративний сайт автоматично, але моделі, які ми створили, доведеться реєструвати вручну.

Щоб додати нашу модель Post на сайт адміністратора, давайте відкриємо файл admin.py і напишемо наступний код:

```
from django.contrib import admin
from .models import Post

admin.site.register(Post)
```

Тепер перезавантажте адміністративний сайт у своєму браузері, і ви побачите такий екранний знімок:

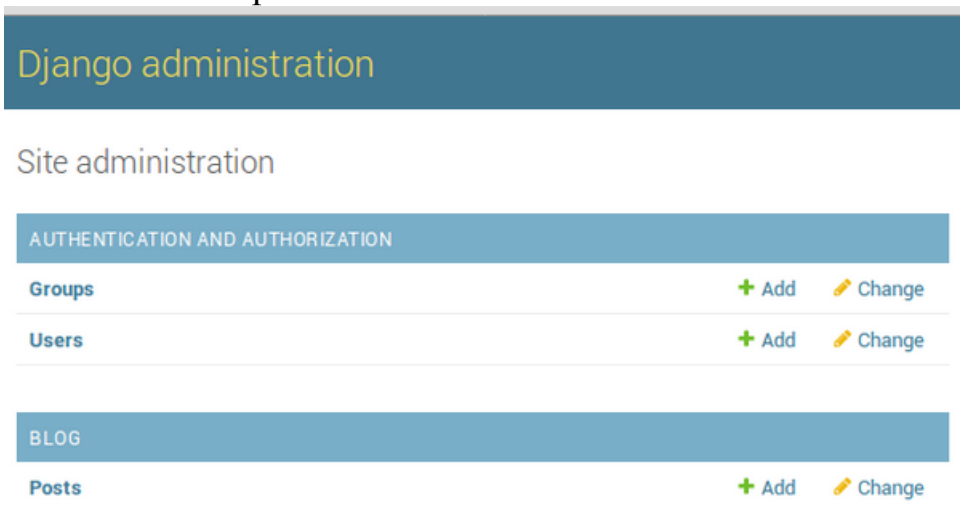


Рис.84. Сторінка адміністратора з моделлю Post

Тепер модель блогу доступна на нашому адміністративному сайті. Це лише невелика частка того, що може зробити адміністратор сайту Django. Коли ви реєструєте свою модель на сайті адміністратора Django, ви отримуєте зручний інтерфейс користувача для ваших моделей, що дозволяє вам просто переглядати, редагувати, створювати та видаляти об'єкти.

Натисніть посилання Add праворуч від Posts, щоб додати нову публікацію, ви побачите форму створення, яку Django згенерував динамічно на основі моделі Post. Ось екранний знімок:

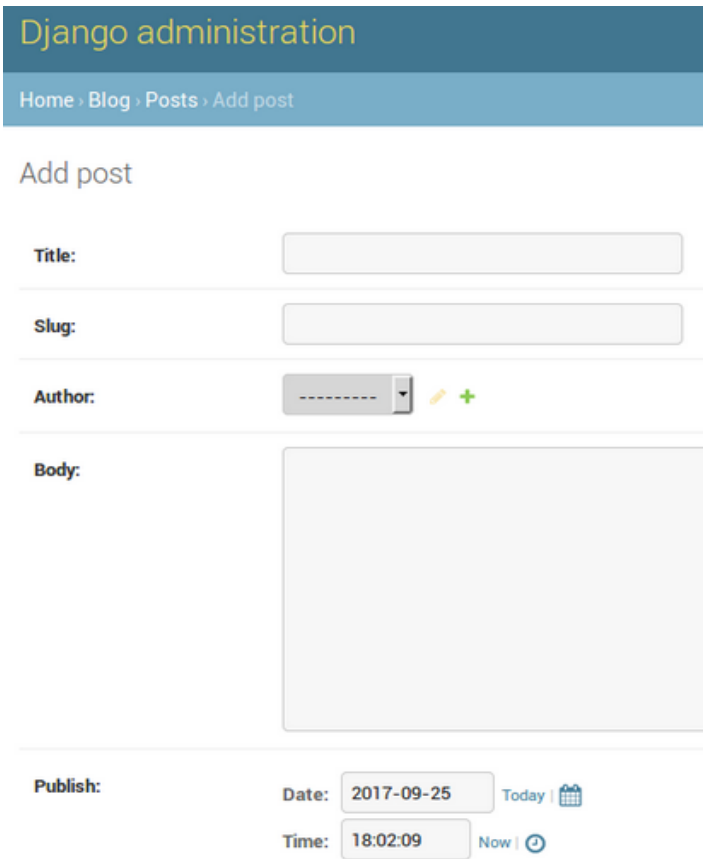


Рис.85. Форма введення данних, згенерована Django

Заповніть форму та натисніть кнопку збереження. Ви будете перенаправлені на сторінку Post з успішним повідомленням про створений вами пост. Django використовує різні формати віджетів для кожного типу полів, навіть складні поля, такі як поля часу дати, які відображаються за допомогою простого інтерфейсу, наприклад, вибору дати JavaScript.

Давайте налаштуємо адміністратора сайту для цього. Додайте наступний код у файл admin.py програми вашого блогу.

```
from django.contrib import admin
from .models import Post

class PostAdmin(admin.ModelAdmin):
    list_display = ('title','slug','author','status','created')
    list_filter = ('status','created','publish','author')
    search_fields = ('title','author')
    prepopulated_fields = {'slug': ('title',)}
    raw_id_fields = ('author',)
    date_hierarchy = 'publish'
    ordering = ['status', 'publish']

admin.site.register(Post, PostAdmin)
```

У клас PostAdmin ми можемо включити інформацію про те, як відображати моделі та взаємодіяти з ними на сайті адміністратора. Атрибут list_display дозволяє вам встановити поля вашої моделі, які ви хочете відобразити на сторінці вебсайту адміністратора. Щоб дізнатись більше про те, як налаштувати свій сайт адміністратора, відвідайте https://docs.djangoproject.com/en/1.11/ref/contrib/admin/ .

Тепер поверніться до свого браузера та перезавантажте сторінку списку публікацій. Тепер він буде виглядати як екранний знімок нижче:

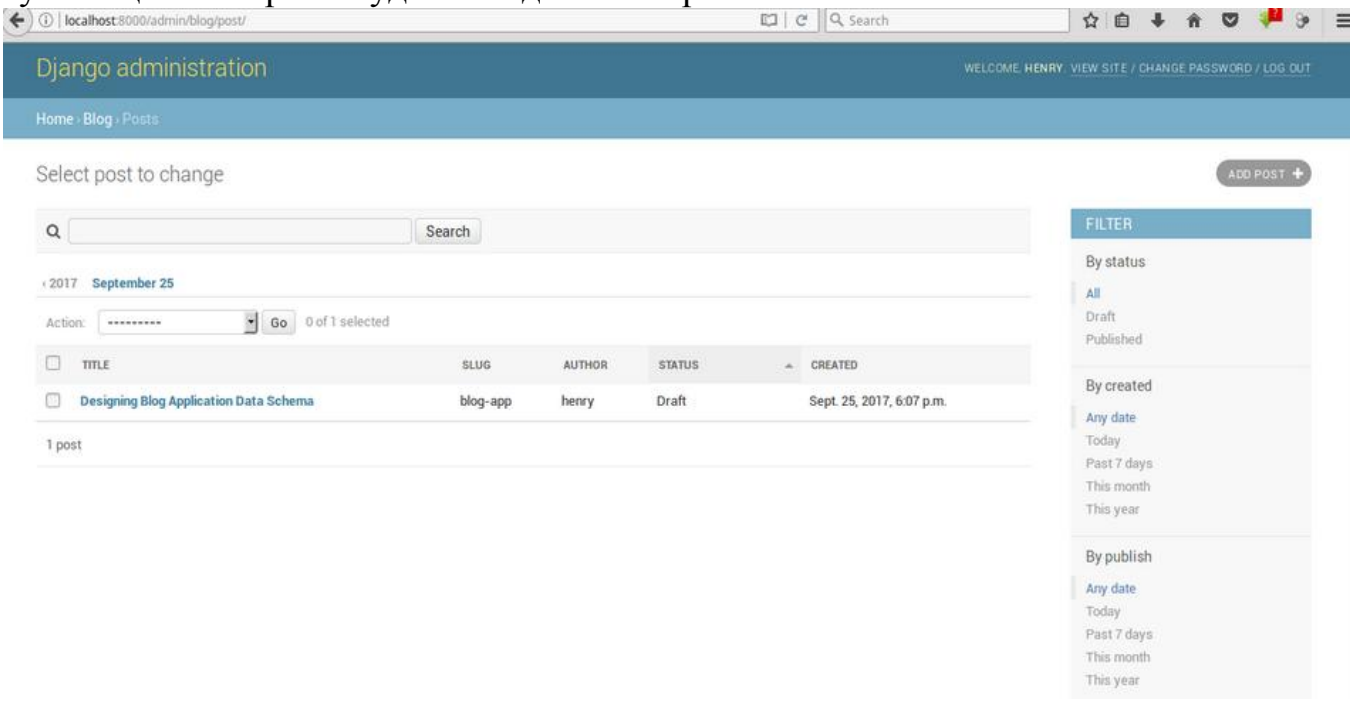


Рис.86. Сторінка списку повідомлень

Сторінка списку тепер включає праву бічну панель, яка дозволяє фільтрувати публікації на основі полів, включених до атрибута list_filter у рядку 6. Панель пошуку з'явилася на сторінці через атрибут search_fields на рядку 7. Трохи нижче поля пошуку присутня панель для навігації через ієрархію дат, це було визначено атрибутом date_hierarchy в рядку 10.

За допомогою декількох рядків коду ми налаштували спосіб відображення моделі нашого повідомлення на сайті адміністратора. Є багато способів налаштування та розширення сайту адміністрування Django. Красиво, ми зробили чудову роботу.

14) Робота з QuerySet & Managers для нашого блогу. Інтерактивна оболонка Django.

Введені дані можна проаналізувати на програмному рівні в інтерактивному термінальному сеансі. Це інтерактивне середовище, зване оболонкою (shell) Django, прекрасно підходить для тестування і діагностики проекту.

У цьому розділі ми спочатку вивчимо, як створювати, оновлювати, завантажувати та видаляти об'єкти в базі даних. Потім ми побачимо, як ми можемо створити модельних менеджерів. Тепер, коли у вас є повністю функціональний сайт адміністрування для керування вмістом вашого блогу, пора навчитися отримувати інформацію з бази даних та взаємодіяти з нею.

Django поставляється з потужною API абстракції баз даних, які дозволяють легко створювати, отримувати, оновлювати та видаляти об'єкти. Mapper Relationship Object Django (ORM) сумісний з такими базами даних, як MySql, SQLite, PostgreSQL та Oracle. Пам'ятайте, що ви можете визначити базу даних свого проекту, редагуючи налаштування бази даних у файлі settings.py вашого проекту. Django може працювати з декількома базами одночасно, і навіть ви мо-

жете запрограмувати маршрутизатори баз даних, які обробляють дані будь-яким способом. Створивши свої моделі даних, Django надає вам безкоштовну API для взаємодії з ними. Тепер давайте створювати об'єкти.

Щоб створити свій перший об'єкт, активуйте віртуальне середовище та перейдіть до базової директорії проекту, де знаходиться файл manage.py на вашому терміналі. Запустіть таку команду, щоб відкрити оболонку python:

```
python manage.py shell
```

Три знаки greater than (‘>>>’) показують, що наша інтерактивна консоль python готова до введення. По-перше, ми повинні імпортувати бібліотеку, введіть таку команду:

```
from blog.models import Post
```

і натисніть Enter.

Тепер щоб отримати користувача з бази даних за допомогою інтерактивної консолі python, введіть таку команду:

```
user = User.objects.get(username='admin')
```

і натисніть Enter.

Щоб створити публікацію, введіть таку команду:

```
post = Post.objects.create(title='Django Blog
App',slug='django-blog-app', body='learning how to create
blog',author=user)
```

Створений нами пост об'єкт знаходиться в пам'яті комп'ютера і не зберігається в базі даних. Щоб зберегти об'єкт повідомлення, нам потрібно скористатись методом save, отже запустіть таку команду:

```
post.save()
```

Отримання об'єкта на нашій інтерактивній консолі.

Mapper для зв'язку об'єктів (ORM) Django заснований на QuerySet. QuerySet - це сукупність об'єктів у вашій базі даних, які можуть мати кілька фільтрів для обмеження результатів. Ви вже знаєте, як отримати один об'єкт з бази даних за допомогою методу get. Кожна модель Django має щонайменше одного менеджера, а менеджер за замовчуванням називається **objects**.

Щоб отримати весь наш блог з бази даних, ми можемо використовувати all() метод, отже запустіть таку команду:

```
post_query = Post.objects.all()
```

Зауважте, що post_query = Post.objects.all() ще не виконано, Django QuerySet «ліниві», і вони оцінюються тільки тоді, коли вони змушені це зробити. Ця поведінка робить Django QuerySet дуже ефективним, отже щоб виконати наш запит запустіть таку команду:

```
Post.objects.all()
```

Тепер ми побачимо список всіх наших постів.

Тепер давайте дізнаємося, як ми можемо використовувати метод фільтрації. Розглянемо приклад, де ми фільтруємо публікації, створені автором admin. Щоб виконати це, треба запускати таку команду:

```
Post.objects.filter (author__username = 'admin')
```

І, нарешті, розглянемо приклад, як фільтрувати за допомогою декількох полів. Наприклад, ми хочемо отримати всі пости, написаі admin в 2017 році. Це команда, яку ми буемо використовувати:

```
Post.objects.filter (publish__year = 2017, author__username =
'admin')
```

Ми створювали запити за допомогою методу пошуку полів із використанням двох підкреслень, наприклад author__username. Наступне, що нам потрібно навчитися, - метод виключення, ви можете виключити певний результат із вашого запиту, використовуючи метод exclude менеджера. Наприклад, ми можемо отримати весь пост, опублікований в 2017 році, і титул якого не починається словом Django. Ми можемо використовувати цю команду:

```
Post.objects.filter(publish__year=2017)\
.exclude(title__startswith='Django')
```

Тепер розглянемо метод order_by. Наприклад, ви можете отримати всі об'єкти, упорядковані за їхнім заголовком. Для цього виконайте таку команду:

```
Post.objects.order_by('title')
```

Видалення об'єктів

Якщо ви хочете видалити об'єкт, ви можете зробити це з екземпляра об'єкта, передаючи ідентифікатор, використовуючи цю команду:

```
post_del = Post.objects.get(id=1)
```

а потім викликати метод видалення за допомогою такої команди:

```
post_del.delete()
```

Менеджери моделей

Як ми вже згадували раніше, objects є менеджером за умовчанням кожної моделі, який використовується для доступу до всіх об'єктів у базі даних, але ми також можемо визначити спеціальні менеджери для наших моделей. Ми збираємося створити спеціальний менеджер, щоб отримати всі публікації з опублікованим статусом. Існує два способи додавання менеджерів до ваших моделей, додавання додаткових менеджерських методів або зміна початкового менеджера QuerySet.

Відредагуйте файл models.py нашої програми для блогу та переконайтеся, що він містить такий код.

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User

# Custom Manager
class PublishedManager(models.Manager):
    def get_queryset(self):
        return su-
per(PublishedManager,self).get_queryset().filter(status='published')

# Our Post Model
class Post(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Draft'),
        ('published', 'Published'),
    )
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250, unique_for_date='publish')
    author = models.ForeignKey(User, related_name='blog_posts')
    body = models.TextField()
```

```
    publish = models.DateTimeField(default=timezone.now)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES, de-
fault='draft')

    # The default manager
    objects = models.Manager()

    # Custom made manager
    published = PublishedManager()

    class Meta:
        ordering = ('-publish',)

    def __str__(self):
        return self.title
```

В рядках 6 - 8 ми визначили нашого індивідуального менеджера моделі нашої публікації. На рядку 7 ми визначили метод get_queryset, який повертає queryset для виконання. Після визначення нашого індивідуального менеджера ми повинні використовувати його в нашій моделі Post для виконання запитів, для цього ми називаємо цього менеджера в рядку 29. Щоб дізнатися більше про менеджерів відвідайте https://docs.djangoproject.com/en/1.11/topics/db/managers/ .

15) Створення списку публікацій та детальних переглядів для нашого блогу.

Перегляд (view) Django - це лише функція Python, яка отримує веб-запит і повертає веб-відповідь. Всередині перегляду йде вся логіка, необхідна для повернення бажаної відповіді. Спочатку ми створимо представлення для нашого блогу, потім ми визначимо шаблон URL для кожного представлення даних і, нарешті, ми створимо шаблон HTML, щоб відтворити данні, згенеровані views.

Кожне представлення (view) рендерить шаблон, передавши в нього змінну, і повертає HTTP-відповідь з рендерованим виходом. Почнемо зі створення функції post_list_view, яка буде містити список публікацій, та функції post_detail_view для детального представлення. Відредагуйте файл views.py програми блогу та переконайтеся, що у вас є такий код:

```
from django.shortcuts import render, get_object_or_404
from .models import Post

def post_list_view(request):
    posts = Post.published.all()
    return render(request, 'blog/post/list.html', {'posts': posts})

def post_detail_view(request, year, month, day, post):
    post = get_object_or_404(Post, slug=post, status='published',
publish__year=year, publish__month=month, publish__day=day)
    return render(request, 'blog/post/detail.html', {'post': post})
```

Ви тільки що створили свої перші Django views. На рядку 4, post_list_view - цей вигляд приймає об'єкт запиту як єдиний параметр, завжди пам'ятайте, що цей параметр потрібен усім переглядам. На рядку 5, posts = Post.published.all () - ми

отримуємо всі публікації, використовуючи менеджер публікацій, який ми створили раніше. У рядку 6 ми використовуємо функцію render, надану Django, щоб відобразити список публікацій у даний шаблон Ця функція приймає об'єкт запиту, шлях шаблону та змінні для рендеринга.

На рядку 8, post_detail_view - це наша друга функція, це представлення, яке покаже окреме повідомлення. Цей перегляд приймає запит, рік, місяць, день та post параметри, щоб отримати опублікований допис з даним значенням slug та датою. Пам'ятайте, коли ми створили модель Post, ми додали унікальний параметр дати в поле slug, таким чином, ми гарантуємо, що буде лише одна публікація з унікальним slug за певну дату.

Для кожного перегляду в Django потрібен шаблон url для його відображення в браузері. Шаблон url у Django складається з регулярних виразів Python, представлення та назви, які дозволяють назвати його будь-де у вашому проекті. Django проходить через кожний шаблон URL-адреси і зупиняється на першому, який відповідає запитуваній URL-адресі. Тоді Django імпортує вигляд, що відповідає шаблону url, і виконує його, передаючи екземпляр класу HTTP-запита і аргументів.

У каталозі додатка blog створіть файл **urls.py** та переконайтеся, що він містить такі рядки коду:

```
from django.conf.urls import url
from . import views

urlpatterns = [
    url(r'^$', views.post_list_view, name='post_list_view'),
  url(r'^(?P<year>\d{4})/(?P<month>\d{2})/(?P<day>\d{2})/(?P<post>[-
\w]+)/$', views.post_detail_view, name='post_detail_view'),
]
```

На рядку 5 - це URL-адреса, яка не приймає жодних аргументів і відображається до post_list_view. У рядку 6 - це шаблон url, який приймає чотири аргументи, а саме рік - вимагає чотирьох цифр, місяць - вимагає двох цифр, день - вимагає двох цифр і post - який може складатися з слів і дефісів. Цей URL посилається на post_detail_view.

Створення файлу urls.py для кожного додатку в Django - це найкращий спосіб повторно використовувати ваш додаток іншими проектами Django. Тепер ми маємо включити шаблони url нашого блогу в основні шаблони URL-адрес проектів. Відредагуйте файл urls.py, який знаходиться в директорії mysite, і переконайтеся, що він містить такий код:

```
"""mysite URL Configuration

The `urlpatterns` list routes URLs to views. For more information please
see:
    https://docs.djangoproject.com/en/1.11/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  url(r'^$', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  url(r'^$', Home.as_view(), name='home')
```

```
Including another URLconf
    1. Import the include() function: from django.conf.urls import url,
include
    2. Add a URL to urlpatterns:  url(r'^blog/', include('blog.urls'))
"""
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'blog/', include('blog.urls', namespace='blog', app_name='blog')),
]
```

У рядку 21 ми повідомляємо Django, що він включає в себе шаблони URL-адрес, визначені у файлі urls.py блогу, ми даємо цим URL-адресам простір імен blog, що полегшує посилання на цю групу URL-адрес.

Канонічні URL-адреси для моделей

Ми можемо використовувати post_detail_view, який ми визначили раніше, для створення канонічної URL-адреси для об'єкта post. Згідно конвенції Django слід додати до моделі метод get_absolute_url, який повертає канонічну URL-адресу об'єктів. Відкрийте файл models.py нашого блогу та переконайтеся, що він містить такий код:

```
from django.db import models
from django.utils import timezone
from django.contrib.auth.models import User
from django.core.urlresolvers import reverse

# Custom Manager

class PublishedManager(models.Manager):
    def get_queryset(self):
        return
super(PublishedManager,self).get_queryset().filter(status='published')


# Our Post Model

class Post(models.Model):
    STATUS_CHOICES = (
        ('draft', 'Draft'),
        ('published', 'Published'),
    )
    title = models.CharField(max_length=250)
    slug = models.SlugField(max_length=250, unique_for_date='publish')
    author = models.ForeignKey(User, related_name='blog_posts')
    body = models.TextField()
    publish = models.DateTimeField(default=timezone.now)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES,
default='draft')

    # The default manager
    objects = models.Manager()
```

```
    # Custom made manager
    published = PublishedManager()

    class Meta:
        ordering = ('-publish',)

    def __str__(self):
        return self.title

    def get_absolute_url(self):
        return reverse('blog:post_detail_view', args=[self.publish.year,
self.publish.strftime('%m'),self.publish.strftime('%d'),self.slug])
```

У рядку 4 ми імпортуємо бібліотеку для нашого канонічного методу reverse url. На лініях 41 та 42 ми використовуємо метод reverse, який дозволяє створювати URL-адреси за їхніми іменами та опційними параметрами. Зауважте, що ми використовуємо функцію strftime у self.publish.strftime ('% m'), self.publish.strftime ('% d'), щоб побудувати URL-адресу, використовуючи місяць та день з передуючими нулями. Ми будемо використовувати метод get_absolute_url у наших шаблонах.

16) Створення шаблонів (templates) для нашого блогу.

У цьому розділі ми будемо додавати шаблони, щоб відображати наші публікації блогу зручним способом. Ми вже створили перегляди (views) та шаблони URL-адрес (url patterns) для нашого додатку blog, тепер пора додавати шаблони (templates). У нашому каталозі blog ми збираємося створити дві папки, названі **templates** та **static**. Всередині каталогу templates створіть іншу папку та назвіть її **blog**. Всередині каталогу blog створіть інший каталог і назвіть його **post**. Ще в каталозі blog створіть файл HTML і назовіть його **master.html.** Всередині папки post створіть два HTML-файли, **detail.html** та **list.html** Всередині каталогу static створіть іншу папку з назвою **css**. Завантажте bootstrap і розмістіть файл **bootstrap.min.css** в папці css, яку ви щойно створили.

Тепер ваш каталог має виглядати так:

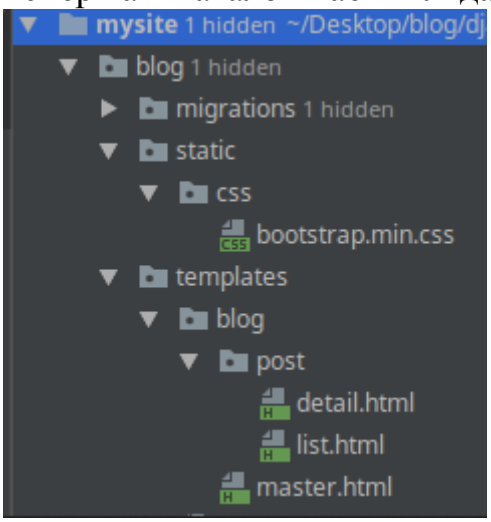


Рис.87. Каталог проекту

Файл master.html включатиме основну HTML-структуру веб-сайту, ми розділимо файл master.html на дві основні області, основну область вмісту та бічну

панель. Файли list.html і detail.html успадкують від master.html, щоб відобразити список публікацій блогу та окреме повідомлення відповідно. Django має потужну мову шаблонів, яка дозволяє вказати, яким чином відображаються дані, вона базується на тегах шаблонів (template tags), які виглядають як змінні шаблону тегів (tag template variable) та фільтри (filters). Щоб дізнатись більше про Django template language відвідайте https://docs.djangoproject.com/en/1.11/topics/templates/ .

Відредагуйте файл **master.html** та переконатися, що він має такий код:

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <link rel="stylesheet" href="{% static "css/bootstrap.min.css" %}">
</head>
<body>
  <div class="container">
      <div class="row">
          <div class="col-md-8">
              {% block content %}

              {% endblock %}
          </div>
          <div class="col-md-4">
              <h3>Muvalab Blog</h3>
              <p>This is our sidebar</p>
          </div>
      </div>
  </div>
</body>
</html>
```

Тепер давайте поглянемо на код. На рядку 1, де ми маємо {% load staticfiles%}, ми повідомляємо Django про завантаження статичних файлів тегів шаблонів файлів, які надаються програмою django.contrib.staticfiles. За допомогою цього тегу шаблонів ви можете включити статичні файли. В рядку 7 ми завантажуємо bootstrap.min.css, використовуючи тег статичного фільтра. У рядку 5 ми маємо теги блоків {% block title%} (% endblock%}, блочні теги повідомляють Django, що ми хочемо визначити блок у цій області. Шаблони, які успадковуються з шаблону master.html, можуть заповнювати цю область блоку вмістом. У цьому шаблоні ми визначили два блоки під назвою title та content.

Відредагуйте файл **list.html** та переконатися, що він має такий код:

```
{% extends "blog/master.html" %}

{% block title %}Our Blog{% endblock %}

{% block content %}
   <h2 class="page-header">Recents Posts</h2>
    {% for post in posts %}
       <h3><a href="{{ post.get_absolute_url }}">{{ post.title }}</a></h3>
        <p class="small">Published {{ post.publish }} by {{ post.author
}}</p>
        <p>{{ post.body|truncatewords:25|linebreaks }}</p>
```

```
    {% endfor %}

{% endblock %}
```

У рядку 1 ми використовуємо тег extends, щоб успадкувати master.html. У рядку 3 ми заповнюємо блок title нашим заголовком. З рядків від 5 до 13 ми визначили нашу область вмісту (block content), в рядку від 7 до 11 ми проводимо ітерацію по всім повідомленням, і ми показуємо назву повідомлення, дату, тіло повідомлення та автора. У рядку 8 ми включаємо посилання на публікацію, використовуючи get_absolute_url (канонічний URL).

Давайте подивимось, що ми маємо на цей момент. Запустіть свій сервер розробки і завжди пам'ятайте, що треба активувати віртуальне середовище. Щоб запустити сервер, виконайте таку команду:

```
python manage.py runserver
```

Відкрийте наступне посилання у своєму веб-переглядачі:

```
http://localhost:8000/blog/
```

і ви повинні побачити такий екран:

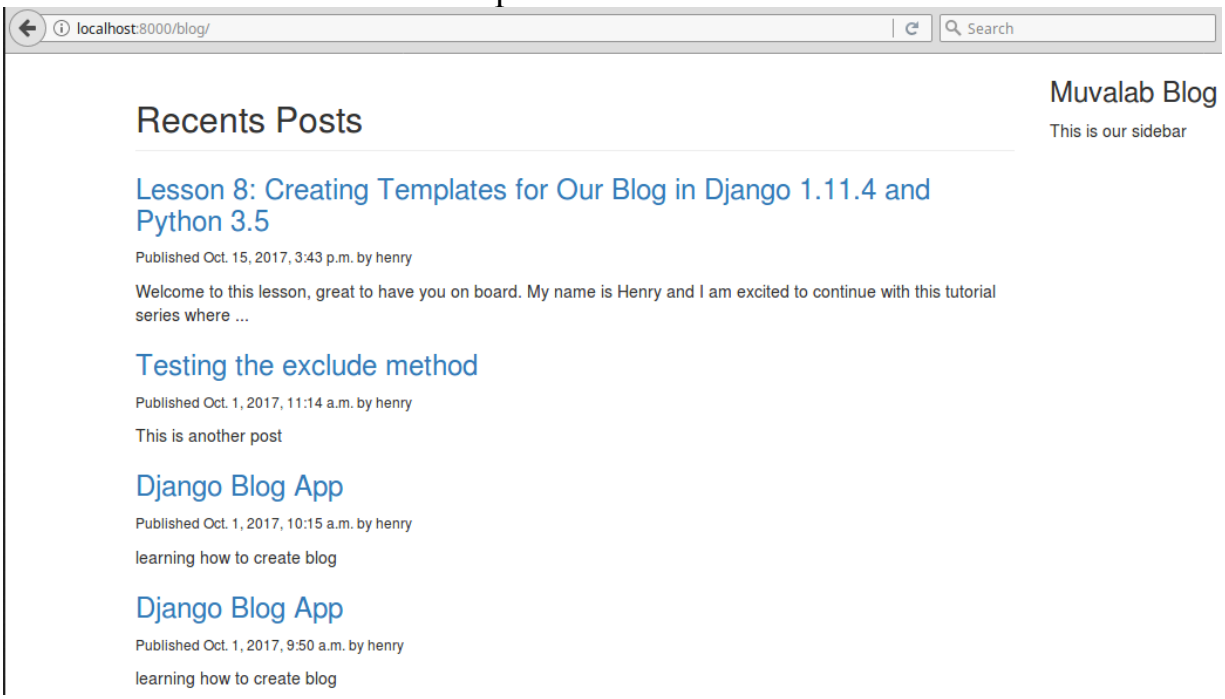


Рис.88. Список повідомлень блогу

Примітка. Вам потрібно мати публікацію, статус якої є published, щоб ви могли переглянути список публікацій.

Тепер відредагуйте файл **detail.html** та переконайтеся, що він має такий код:

```
{% extends "blog/master.html" %}

{% block title %}{{ post.title }}{% endblock %}

{% block content %}
   <h3>{{ post.title }}</h3>
    <p class="small">Published {{ post.publish }} by {{ post.author }}</p>
    <p>{{ post.body|linebreaks }}</p>
{% endblock %}
```

Тепер поверніться до свого веб-переглядача та натисніть на одне з заголовків повідомлень, щоб побачити повний допис. Ви повинні побачити щось на зразок цього:

Рис.89. Сторінка окремого повідомлення

Тепер подивіться на URL-адресу свого браузера, ми створили дружню URL-адресу для нашого блогу.

17) Створення розбивки на сторінки (Pagination) для нашого блогу.

Коли кількість публікацій у блозі буде зростати, ви скоро зрозумієте, що вам потрібно розділити список публікацій на кілька сторінок. Django має вбудований клас pagination, який дозволяє розробникам легко керувати розбивкою на сторінки. Тепер відредагуйте файл views.py і переконайтеся, що він має такий код:

```
from django.shortcuts import render, get_object_or_404
from django.core.paginator import Paginator, EmptyPage,
PageNotAnInteger
from .models import Post

def post_list_view(request):
    list_objects = Post.published.all()
    paginator = Paginator(list_objects, 3)
    page = request.GET.get('page')
    try:
        posts = paginator.page(page)
    except PageNotAnInteger:
        posts = paginator.page(1)
    except EmptyPage:
        posts = paginator.page(paginator.num_pages)
    return render(request, 'blog/post/list.html', {'posts': posts})

def post_detail_view(request, year, month, day, post):
    post = get_object_or_404(Post, slug=post, status='published',
publish__year=year, publish__month=month, publish__day=day)
    return render(request, 'blog/post/detail.html', {'post': post})
```

На рядку 2 ми імпортуємо Django paginated classes. Ми змінили post_list_view, розглянемо як працює pagination у Django:

Рядок 7 – створюється об’єкт класу Paginator з заданням кількості об'єктів, які ви хочете відобразити на кожній сторінці.

Рядок 8 - Ми отримуємо параметр GET сторінки, який вказує поточний номер сторінки.

Рядок 10 - Ми отримуємо об'єкт для потрібної сторінки page, викликаючи метод page об'єкта paginator.

Рядок 12 - Якщо параметр сторінки не є цілим числом, ми отримуємо першу сторінку результатів, якщо цей параметр є більшим, ніж остання сторінка результатів, ми отримуємо останню сторінку.

Рядок 14 - ми передаємо в paginator кількість сторінок та отримуємо об'єкти до шаблону.

Тепер нам потрібно створити шаблон, щоб відобразити paginator таким чином, щоб він міг бути включений у будь-який шаблон, який використовує pagination. У папці **template** додатку blog створіть новий файл та назвіть його **pagination.html**. Переконайтеся, що pagination.html має такий код:

```
<div class="container">
    <div class="row">
    <span class="pagination">
        {% if page.has_previous %}
            <a href="?page={{ page.previous_page_number
}}">previous</a>
        {% endif %}

      <span class="active">
          page {{ page.number }} of {{ page.paginator.num_pages }}
      </span>
    {% if page.has_next %}
        <a href="?page={{ page.next_page_number }}">next</a>
    {% endif %}
    </span>
</div>
</div>
```

Шаблон сторінок (pagination template) очікує об'єкта page, щоб відобразити попереднє посилання, наступне посилання, поточну сторінку та загальну сторінку. Тепер давайте відкриємо наш файл шаблону **list.html**, розташованого в blog/post/ директорії, і переконайтеся, що він має такий код:

```
{% extends "blog/master.html" %}

{% block title %}Our Blog{% endblock %}

{% block content %}
   <h2 class="page-header">Recents Posts</h2>
    {% for post in posts %}
       <h3><a href="{{ post.get_absolute_url }}">{{ post.title
}}</a></h3>
        <p class="small" style="color: #777777">Published {{
post.publish }} by {{ post.author }}</p>
        <p>{{ post.body|truncatewords:25|linebreaks }}</p>
```

```
    {% endfor %}

    {% include "pagination.html" with page=posts %}
{% endblock %}
```

У рядку 13 ми просто включили нашу сторінку pagination.html у нижню частину блоку вмісту. Оскільки об'єкт page ми передаємо до шаблону list.html, ми у рядку 13 задаємо page = posts. Подібний підхід ви можете використовувати для повторного використання шаблону сторінок у ваших проектах Django, а також для розбивки на сторінки різних моделей.

Переконайтеся, що ви додали більше 5 повідомлень для тестування своїх сторінок. Після завершення додавання повідомлень розгорніть сервер розробки та відкрийте це посилання у своєму веб-переглядачі:

http://localhost:8000/blog/

Ось можливий екранний знімок браузера:

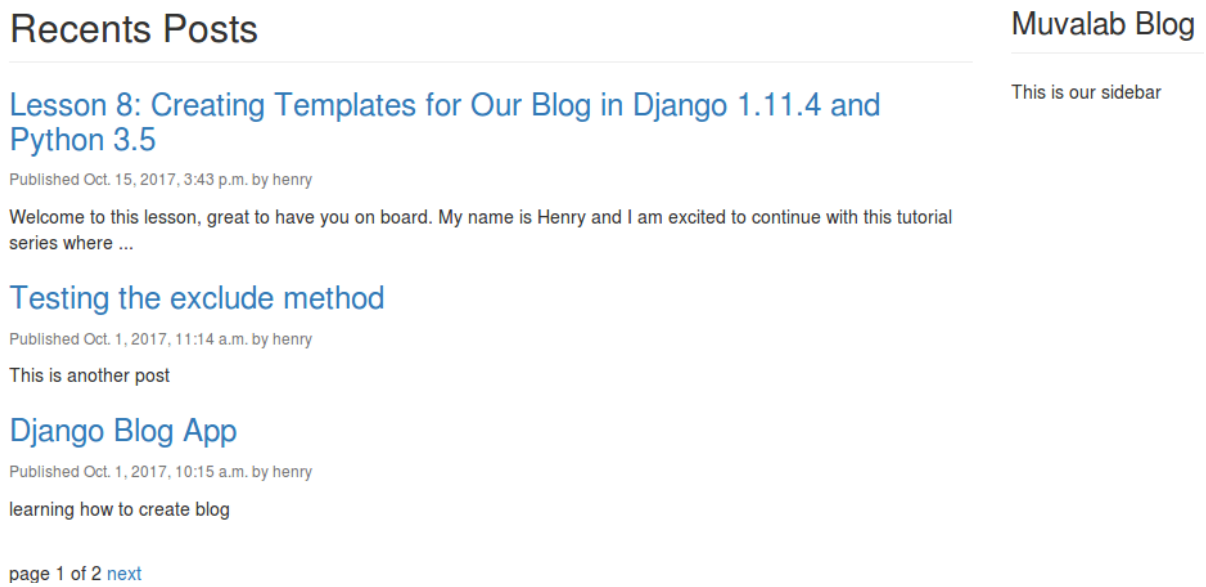


Рис.90. Список повідомлень блогу з розбиттям на сторінки

Ви побачите розбивку на сторінки (pagination) в нижній частині сторінки списку публікацій, і ви матєте змогу переходити по сторінках.

18) Створення RSS-каналів для нашого блогу.

Веб-фреймворк Django має вбудований syndication feed framework, який ви можете використовувати для динамічного створення каналів RSS або atom feeds. У нашому каталозі blog створіть новий файл з назвою **feeds.py** і переконайтеся, що в ньому є такий код:

```
from django.contrib.syndication.views import Feed
from django.template.defaultfilters import truncatewords
from .models import Post

class PostsFeed(Feed):
    title = 'Henry Blog Feeds'
    link = '/blog/'
    description = 'Our latest Posts!'

    def items(self):
```

```
        return Post.published.all()[:5]

    def item_title(self, item):
        return item.title

    def item_description(self, item):
        return truncatewords(item.body, 20)
```

Давайте зрозуміємо вищезазначений код. У рядку 1 ми імпортуємо бібліотеку синдикації, яку нам надає Django. У рядку 2 ми імпортуємо фільтр шаблонів, який допоможе нам підсумувати тіло нашого блогу. У рядку 3 ми імпортуємо нашу модель Post з файлу models.py.

У рядках 5 - 8 ми створюємо клас під назвою PostsFeed, і ми передаємо потік синдикації в цей клас, таким чином, у нас є PostFeed (Feed), який стає підкласом класу Feed базової синдикації Django. Атрибути title, link та description відповідають назві, посиланням та опису RSS відповідно.

У рядках 10 - 11 ми створили метод items, який витягує об'єкти з нашої моделі Post, які будуть включені в канал, в цьому випадку ми вказуємо 5 останніх публікацій. У рядках 13 - 14 ми створили метод item_title, який повертає назву отриманого об'єкта. У рядках 16 - 17 ми створили метод item_description, який повертає тіло повідомлення, і ми вирізаємо це тіло до перших 20 слів.

Тепер нам потрібно змінити файл **urls.py** нашої блогової програми, переконайтеся, що файл має такий код:

```
from django.conf.urls import url
from . import views
from .feeds import PostsFeed

urlpatterns = [
    url(r'^$', views.post_list_view, name='post_list_view'),

url(r'^(?P<year>\d{4})/(?P<month>\d{2})/(?P<day>\d{2})/(?P<post>[-
\w]+)/$', views.post_detail_view, name='post_detail_view'),
    url(r'^feed/$', PostsFeed(), name='post_feed')
]
```

У рядку 3 ми імпортуємо клас PostsFeed, який ми щойно створили в нашому файлі feeds.py. У рядку 8 ми інсталюємо PostsFeed в новому шаблоні URL-адреси. Давайте перевіримо наш код, повертаємо ваш сервер розробки та переходимо до цього URL:

http://localhost:8000/blog/feed/

і ви побачите наступний вивід у своєму веб-переглядачі.

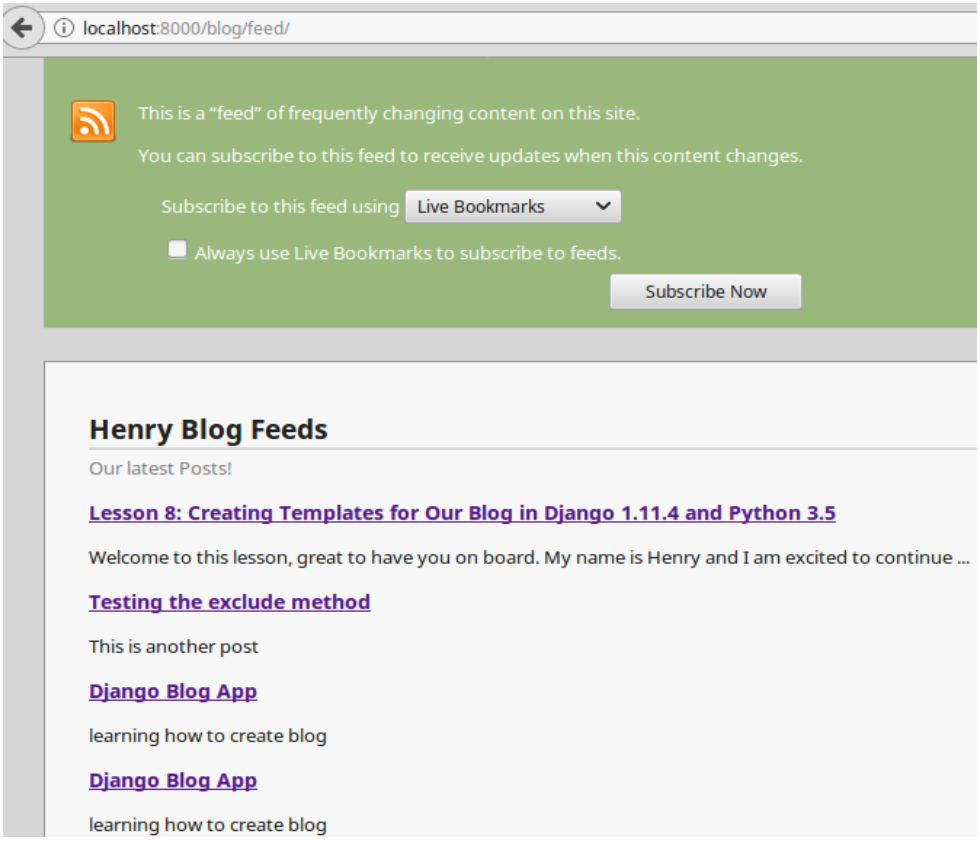


Рис.91. Сторінка RSS-каналів для блогу

Останній крок - створити посилання на передплату RSS на бічній панелі блогу. У нашому шаблоні відкрийте **master.html** і переконайтеся, що він має такий код:

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %}{% endblock %}</title>
    <link rel="stylesheet" href="{% static "css/bootstrap.min.css"
%}">
</head>
<body>
  <div class="container">
      <div class="row">
          <div class="col-md-8">
              {% block content %}

              {% endblock %}
          </div>
          <div class="col-md-4">
              <h3 class="page-header">Muvalab Blog</h3>
              <p><a href="{% url "blog:post_feed" %}"
target="_blank">Subscribe to RSS Feed</a></p>
          </div>
  </div>
</body>
</html>
```

У рядку 19 ми додали цей код: `<p><a href="{% url "blog:post_feed" %}" target="_blank">Subscribe to RSS Feed</a></p>`, які додають посилання для підписки на канал RSS на бічній панелі блогу. Тепер відкрийте наступне посилання у своєму веб-переглядачі:

http://localhost:8000/blog/

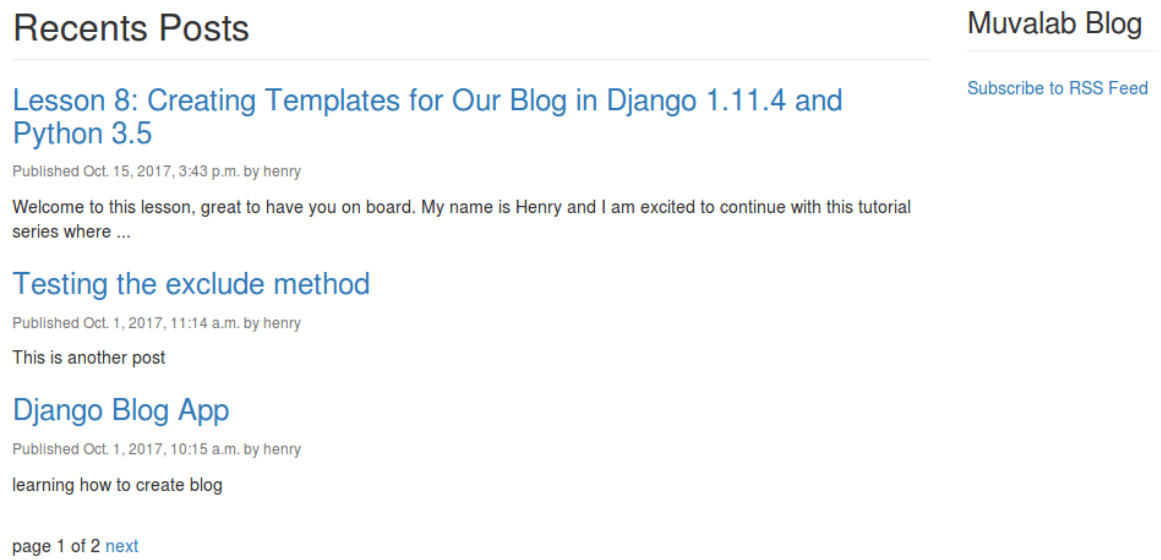


Рис.92. Сторінка з посиланням підписки на RSS-канали

Наведене вище зображення показує, що наше посиланя підписки було успішно додане.

3 ЕКЗАМЕНАЦІЙНІ ПИТАННЯ

1. Варіанти налаштування робочого середовища розробки серверних додатків на мові PHP.
2. Протокол HTTP. Структура запиту та відповіді. Основні заголовки. Стандарт MIME.
3. Загальна характеристика мови PHP. Основи синтаксису PHP: імена змінних, функцій, констант. Теги включення PHP в HTML.
4. Синтаксис PHP: типи змінних та даних, Область видимості змінних. Глобальні і локальні змінні.
5. Синтаксис PHP: цикли, умовні оператори. Варіанти синтаксису.
6. Рядки в PHP. Основні операції з рядками в PHP.
7. Масиви в PHP. Масиви з числовою індексацією. Асоціативні масиви. Функції для роботи з масивами.
8. Багатовимірні масиви в PHP.
9. Суперглобальні масиви в PHP.
10. Користувацькі функції в PHP (оголошення і виклик функції, вкладені функції, аргументи, рекурсивні функції, області видимості).
11. Анонімні функції (Lambda) та замикання (Closure) в PHP. Генератори.

12. Приймання даних з форми. Масив $_REQUEST. Приклад простої форми та програми, яка обробляє отримані дані.
13. Використання регулярних виразів в PHP.
14. Механізм сесій в PHP. Місце зберігання даних сесії. Послідовність дій сервера та клієнта під час роботи з сесією.
15. Сесії в PHP: масив $_SESSION, створення сесії, збереження даних в сесії, використання даних сесії.
16. Введення в MySQL. Доступ до бази даних з командного рядка.
17. Бази даних в PHP на прикладі MySQL. Бібліотека MySQLi. З'єднання з базою, виконання запитів SQL, додавання, отримання , редагування, видалення даних.
18. Бібліотека PDO (з'єднання з базою даних, виконання запитів).
19. Об'єктно-орієнтований підхід в PHP. Створення класів та екземплярів класу, властивості та методи, конструктори, деструктори, клонування, успадкування.
20. Обробка виключень в PHP.
21. Абстрактні класи та методи в PHP.
22. Інтерфейси в PHP. Трейти в PHP.
23. Стандарти PSR. Пакетний менеджер Composer.
24. Робота з MVC-фреймворками (на прикладі Laravel).
25. Шаблон проектування MVC (модель, вид, контролер, типова послідовність роботи MVC-додатку, огляд фреймворків PHP).
26. PHP. Завантаження файлів на сервер.
27. Обробка XML в PHP: підхід з використанням SimpleXML та DOM, SAX.
28. PHP 7. Нові особливості PHP 7 у порівнянні з попередніми версіями.
29. Безпека створюваних web-застосунків. Session Security. XSS-ін'єкція. CSRF-атака. SQL-ін'єкція.
30. Файл .HTACCESS. Модуль MOD_REWRITE.
31. Що таке NodeJS? Як працює NodeJS? Архітектура NodeJS.
32. Створення HTTP-сервера на NodeJS.
33. Модулі NodeJS.
34. NPM – менеджер пакетів для NodeJS.
35. Маршрутизація (routing) в NodeJS.
36. Як завантажити HTML-сторінку в NodeJS (відображення статичних ресурсів)?
37. Роль файла package.json в NodeJS.
38. Що таке middleware в NodeJS?
39. Основи фреймворку Express.js для NodeJS.
40. Middleware (програмне забезпечення проміжної ланки) для Express. Маршрутизація в Express. Генератор структури застосунків Express.

41. Движки шаблонів (Template engines) для Express.
42. Типи даних Python. Кортежи (Tuple). Списки (Lists). Набори (Sets). Словники (Dictionaries).
43. Об’єкти та класи в Python.
44. Одиночне успадкування в Python.
45. Пакети в Python. Менеджери пакетів Python.
46. Огляд web-фреймворку Django. Архітектура Django.
47. Інсталяція Django. Створення та налаштування найпростішого “It worked!”-проекту.
48. Створення Django додатку (application) за допомогою утиліти manage.py. Структура каркасу додатків Django. Реєстрація додатка.
49. Об’єктно-реляційна проекція баз даних засобами Django.
50. Моделі в Django. Типи даних полів моделей. Підключення та налаштування інтерфейсу адміністратора.
51. Представлення (views) в Django. Налаштування URL-адрес. Шаблони.

ЛІТЕРАТУРА

1. Котеров Д.В., Симдянов И.В. PHP 7. В подлиннике. – СПб.:БХВ, 2016. – 1088 с.
2. Дэвид Скляр. Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов. – М.-СПб.-Киев,: Диалектика, 2017 – 464 с.
3. Джош Локхарт. Современный PHP. Новые возможности и передовой опыт. - М.: ДМК Пресс, 2016. - 304 с.
4. Скляр Д., Трахтенберг А. PHP. Рецепты программирования. 3-е изд. — СПб.: Питер, 2015. — 784 с.
5. Andrew Beak PHP 7 Zend Certification Study Guide. – Apress, 2017. – 294 p.
6. Larry Ullman. PHP for the Web, Fifth Edition. Visual QuickStart Guide. - Peachpit Press, 2016 – 474 p.
7. Larry Ullman. PHP and MySQL for Dynamic Web Sites, Fifth Edition. - Peachpit Press, 2018 – 1187 p.
8. Tom Butler, Kevin Yank. PHP & MySQL: Novice to Ninja, 6th Edition. - SitePoint Pty. Ltd., 2017
9. Antonio Lopez. Learning PHP 7. - Packt Publishing, 2016. – 393 p.
10. Branko Ajzele. Modular Programming with PHP 7. - Packt Publishing, 2016. – 313 p.
11. Phil Sturgeon, Josh Lockhart. PHP. The Right Way. - Lean Publishing, 2014
12. Karol Król. WordPress Complete, Sixth Edition. - Packt Publishing, 2017
13. Sanjib Sinha. Beginning Laravel. – Apress, 2017. – 189 p.
14. Mizanur Rahman. PHP 7 Data Structures and Algorithms. - Packt Publishing, 2017

15. Mikael Olsson. PHP 7 Quick Scripting Reference. - Apress, 2016. – 136 p.
16. Jose Palala, Martin Helmich. PHP 7 Programming Blueprints. - Packt Publishing, 2016. – 304 p.
17. Люк Веллинг, Лора Томсон. Разработка веб-приложений с помощью PHP и MySQL, Четвертое издание. - М.-СПб.-Киев,: Вильямс, 2010 – 848 с.
18. Котеров Д.В., Костарев А.Ф. PHP5. – СПб.: БХВ, 2005. – 1120 с.
19. Робин Никсон Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 3-е изд. — СПб.: Питер, 2015. — 688 с.
20. Larry Ullman. PHP and MySQL for Dynamic Web Sites, Fourth Edition. — Peachpit Press, 2012. — 677 p.
21. Joel Murach, Ray Harris. Murash's PHP and MySQL, 2nd Edition. – Mike Murach & Associates, Inc., 2014. – 872 p.
22. Коггшолл, Дж. PHP5. Полное руководство. - М.:"Вильямс", 2006. — 752 с.
23. Аргерих Л., Ванкиу Ч., Коггшолл Дж. Профессиональное PHP программирование. - М.: Символ-Плюс, 2006. — 1048 с.
24. Конверс Т., Парк Дж., Морган Кл. PHP5 и MySQL. Библия пользователя. - М.: Вильямс, 2006. — 1216 с.
25. Гутманс Э., Баккен Ст., Ретанс Д. PHP5. Профессиональное программирование.- М.: Символ-Плюс, 2006. — 704 с.
26. Web Applications for Everybody. – Електронний ресурс. Режим доступу: https://www.wa4e.com/ .
27. Серверні web-технології в світі. – Електронний ресурс. Режим доступу: http://w3techs.com/technologies/overview/programming_language/all.
28. Веселов А.І., Новікова О.С. Web програмування. Методичні вказівки до виконання лабораторних робіт. ЗДІА. – 2008. – 65 с.
29. Попівщий В. І., Веселов А. І., Новікова О. С. Web-програмування. Навчально-методичний посібник, ЗДІА, 2009, 158 с.
30. Романчик В.С. Веб-программирование. — Минск: БГУ, 2013. — 407 с.
31. Шэлли Пауэрс. Изучаем Node. Переходим на сторону сервера. 2-е изд., доп. и перераб. — СПб.: Питер, 2017. — 304 с.
32. Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript. — СПб.: Питер, 2017. — 336 с.
33. Мэтиз Эрик. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. - СПб.: Питер, 2017. – 496 с.
34. Leif Azzopardi, David Maxwell. How to Tango with Django 1.9. A beginners guide to Python/Django, - Lean Publishing, 2016
35. Дронов В.А. Django: практика создания Web-сайтов на Python. - СПб.: БХВ, 2016. – 528 с.