

Міністерство освіти і науки України
Державний заклад «Південноукраїнський національний
педагогічний університет імені К.Д. Ушинського»
Фізико-математичний факультет

Шувалова Ольга Ігорівна

WEB-ПРОГРАМУВАННЯ. ПОБУДОВА WEB-ОРІЄНТОВАНОЇ
ІНФОРМАЦІЙНОЇ СИСТЕМИ

методичні рекомендації

Одеса 2019

УДК: 378.013+004.9

Друкується за рішенням Вченої ради Державного закладу
«Південноукраїнський національний педагогічний університет
імені К.Д. Ушинського» (протокол №11 від 27 червня 2019 року).

Шувалова О.І.

Web-програмування. Побудова Web-орієнтованої інформаційної системи: Методичні рекомендації до лабораторних робіт з теми «Бази даних в проектуванні інформаційних систем». - Одеса: Університет Ушинського, 2019. - 55с.

Рецензенти:

Максимов Максим Віталійович, доктор технічних наук, професор, завідувач кафедри Комп'ютерних технологій автоматизації Одеського національного політехнічного університету.

Брескіна Лада Валентинівна, кандидат педагогічних наук, доцент кафедри Прикладної математики та інформатики Південноукраїнського національного педагогічного університету імені К. Д. Ушинського.

Методичні рекомендації «**Web-програмування. Побудова Web-орієнтованої інформаційної системи**» – це навчальне видання для лабораторно-практичної частини другого змістового модулю «Бази даних в проектуванні інформаційних систем» навчальної дисципліни ВС 3.6 «Бази даних» для студентів 4 курсу спеціальності 014 Середня освіта (Фізика) або 014 Середня освіта (Математика) з другою спеціальністю 014 Середня освіта (Інформатика). Поданий у методичних рекомендаціях матеріал охоплює питання, що розкриває методи побудови клієнтської та адміністративної частини Web-орієнтованої інформаційної системи. Базовий акцент зроблено на опрацювання SQL запитів та формування інтерфейсів введення та виведення даних SQL запитів. Web-орієнтована інформаційна система реалізується на основі MVC (Model-View-Controller) архітектури з використанням мови серверного програмування PHP, шаблонизатору Twig та мови SQL запитів при роботі з СУБД MySQL.

Реалізація навчальних прикладів до лабораторно-практичних робіт прив'язується до реальної реалізації у інформаційних системах, що побудовано на основі CMS Joomla.

Ключові слова: Web-програмування, MVC архітектура, шаблонизатор Twig, мова PHP, методика інформатики, інформаційна система, бази даних.

Зміст

Вступ.....	4
1. Лабораторна робота №1. MVC-архітектура (Model-View-Controller) Web-орієнтованої системи. Розгортання системи “MVC-старт”.....	5
2. Лабораторна робота №2. Клас PDO в мові PHP. Налаштування роботи з базою даних. Представлення результатів SELECT запиту у форматі таблиці інформаційної системи.....	15
3. Лабораторна робота №3. Опрацювання більш складних SELECT запитів до бази даних. Представлення результатів SELECT запиту у форматі таблиці інформаційної системи.....	22
4. Лабораторна робота №4. Параметричні SELECT запити до бази даних. Налаштування пошуку даних в базі даних. Представлення результатів SELECT запиту у форматі таблиці інформаційної системи.....	25
5. Лабораторна робота №5. Параметричні SELECT запити до бази даних. Структурування результатів у форматі блогу інформаційної системи.....	30
6. Лабораторна робота №6. Побудова інтерфейсу адміністрування Web-орієнтованої інформаційної системи. Менеджер адміністрування таблиці.....	36
7. Лабораторна робота №7. Адміністративний розділ. Команда INSERT. Менеджер додавання запису.....	42
8. Лабораторна робота №8. Адміністративний розділ. Команда UPDATE. Менеджер редагування запису.....	46
9. Лабораторна робота №9. Адміністративний розділ. Команда DELETE. Менеджер видалення запису.....	51

Вступ

Навчальна дисципліна ВС 3.6 «Бази даних» для студентів 4 курсу спеціальності 014 Середня освіта (Фізика), 014 Середня освіта (Математика) з другою спеціальністю 014 Середня освіта (Інформатика) за навчальним планом підготовки фахівця освітнього рівня «бакалавр» охоплює 5 кредитів загальною кількістю 150 годин. Аудиторні заняття - 74 години (лекції — 14 годин, лабораторно-практичні роботи — 60 години), 76 годин відведено на самостійну роботу студентів. Зміст навчальної дисципліни розділено за програмою на два змістових модуля, перший з яких охоплює загальну теорію реляційних баз даних та мову SQL запитів, а другий присвячено питанням Web-програмування, побудові Web-орієнтованих інформаційних систем.

Методичні рекомендації «Web-програмування. Проектування та програмна реалізації Web-орієнтованої інформаційної системи» розроблено відповідно до програми лабораторно-практичних робіт другого змістового модуля навчальної дисципліни ВС 3.6 «Бази даних». В методичних рекомендаціях представлено 9 лабораторно-практичних робіт, що за програмою навчальної дисципліни розраховано на 30 години аудиторної роботи, одне підсумкове заняття (2 години) по звітуванню результатів проектної роботи та 40 годин самостійної роботи студентів.

Зміст лабораторно-практичних робіт спирається на базові знання студентів, що отримано в межах навчальної дисципліни «Інформатика» та «Комп'ютерні мережі» на першому, другому та третьому курсах навчання.

В цих методичних рекомендаціях надається можливість спроектувати індивідуальні бази дані до предметної галузі “Школа”, “Університет” з наданням орієнтовних рекомендацій щодо структури таблиць і їх зв'язків. Пропонується налаштувати авторський дизайн майбутньої інформаційної системи. У лабораторних роботах 2, 3, 4, 5 приділено увагу до правил структурування та відображення результатів SELECT запитів у форматі таблиці, блогу та матеріалу (статті). У лабораторних роботах 6, 7, 8, 9 виконано проектування та програмну реалізацію адміністративного розділу інформаційної системи, що забезпечує роботу з командами INSERT, UPDATE, DELETE. У всіх лабораторних роботах проводиться паралель з програмною реалізацією відповідних тематичних розділів в професійній реалізації Web-орієнтованої інформаційної системи засобом CMS Joomla.

З методичної точки зору вагоме значення має набуття практичних навичок проектування та опрацювання баз даних. Системне бачення взаємозв'язків між тематичними розділами науки “Інформатика” через вивчення Web-програмування в подальшому планується використовувати у курсі “Методика навчання інформатики старшої школи” у магістратурі за спеціальністю 014 Середня освіта (Інформатика) в відповідності до навчального плану.

Лабораторна робота №1

Мова php. Налаштування інтерфейсу виведення результатів. MVC-архітектура (Model-View-Controller) Web-орієнтованої системи.

Розгортання системи “MVC-старт”

Мета роботи: Придбання практичних навичок створення інтерфейсу виведення даних в Web-орієнтованій системі з MVC архітектурою. Базова структура файлів частини контроллера (Controller) та частини представлення (View).

Теоретичний матеріал

Мова php - це мова серверного програмування, що означає:

- 1) запуск програми здійснюється під керуванням Web-серверу;
- 2) результати виконання програми виводяться в гіпертекстовий інтерфейс;
- 3) для запуску програми php файли необхідно розмістити на хостингу.

Інструкція щодо користування хостингом:

- 1) найкраще працювати з реальним хостингом. Безкоштовний хостинг на даний момент працює на [zzz.com.ua](https://www.zzz.com.ua) Інструкція щодо правил роботи. (<https://www.zzz.com.ua/ru/pomoshch>);
- 2) альтернативою стає робота на віртуальному Web сервері Denwer (<http://www.denwer.ru/>). Інструкція щодо правил роботи (<http://forum.dklab.ru/denwer/>).

Базові підходи до виведення результатів роботи php програми:

- 1) пряме виведення (через використання функції echo в php файлі);
- 2) виведення в шаблон сторінки. В цьому випадку ми формуємо в php файлі 1-масив змінних, що будемо виводити. Формуємо 2-файл-шаблон, в який будемо виводити.

Розгляд прикладу.

Для пояснення сутності різних підходів до виведення даних розглянемо простий приклад.

На основі наданих значень довжин сторін прямокутника знайти його периметр та площу і вивести на екран початкові дані і результати.

Рішення даної задачі в мові програмування php принципово не відрізняється від рішення даної задачі в будь-якій мові програмування. Звичайний лінійний алгоритм.

```
<?php
```

```
...
```

```
$a=7;
```

```
$b=8;
```

```
$S=$a*$b;
```

```
$P=2*($a+$b);
```

```
...
```

```
?>
```

Далі постає питання, куди виводити результат. Виявляється, що для виведення

результату треба:

- 1) сформувати гіпертекстову сторінку з якимось (може навіть найпростішим дизайном);
- 2) в даній сторінці визначити місце для виведення результатів рішення задачі;
- 3) вивести отримані результати в визначене місце на гіпертекстовій сторінці з використанням HTML тегів для форматування даних, що виводяться.

Для остаточного рішення нашої задачі-прикладу візьмемо базову гіпертекстову сторінку наступної структури (Рис. 1)

Перший спосіб виведення результатів - це пряме виведення. Результуючий файл представлено на рисунку (Рис. 2)

В чому постає проблема при даному варіанті реалізації виведення результатів:

1. при форматуванні нам приходится об'єднувати теги HTML форматування з php кодом, а значить програмна частина перетинається з дизайном сторінки;
2. виведення даних можливе саме у вказане місце, якщо виникає необхідність щось змінювати в іншій частині гіпертекстового документу, то засобів немає.

Інший підхід - це розмежування частини контроллера та дизайну на основі MVC архітектури. Побачити як виглядає програмна реалізація даного приклада можна в моделі на рисунку (рис. 3)

Переглядаючи дві схеми (рис.2) та (рис.3) ми можемо бачити те, що в першому варіанті виведення даних шаблон гіпертекстової сторінки поділяється на 2 частини, а в моделі MVC в шаблоні виділяється варіативний іменований блок. Такі іменовані блоки можуть бути в довільному місці гіпертекстового документа і їх кількість не обмежена.

Подальше на практиці ми будемо користуватися моделлю MVC для виведення результатів роботи php скриптів. В файлі-шаблоні ми будемо користуватися синтаксисом бібліотеки Twig.

Модель MVC. Шаблонізатор Twig.

Стандартна схема архітектури MVC «Модель-Вид-Контролер» Model View Controller зображена на наступному рисунку (рис. 4).

Розберемо по пунктах дану схему.

У моделі MVC, як випливає з назви, є три основних компоненти: Модель, Дизайн, і Контролер.

Дизайн (вид) відповідає за відображення інформації, що надходить із системи або в систему.

Модель є «суттю» системи і відповідає за дані системи, її інформаційну частину (як правило це база даних).

Контролер є сполучною ланкою між «дизайном» і «моделлю» системи, за допомогою якого і існує можливість провести поділ між ними. Контролер отримує дані від користувача і передає їх в «модель». Крім того, він отримує повідомлення від моделі, і передає їх в «дизайн».

Стосовно до інтернет-додатків здається, що частини контролер і дизайн

об'єднані, тому що за відображення і одночасно за введення інформації відповідає браузер.

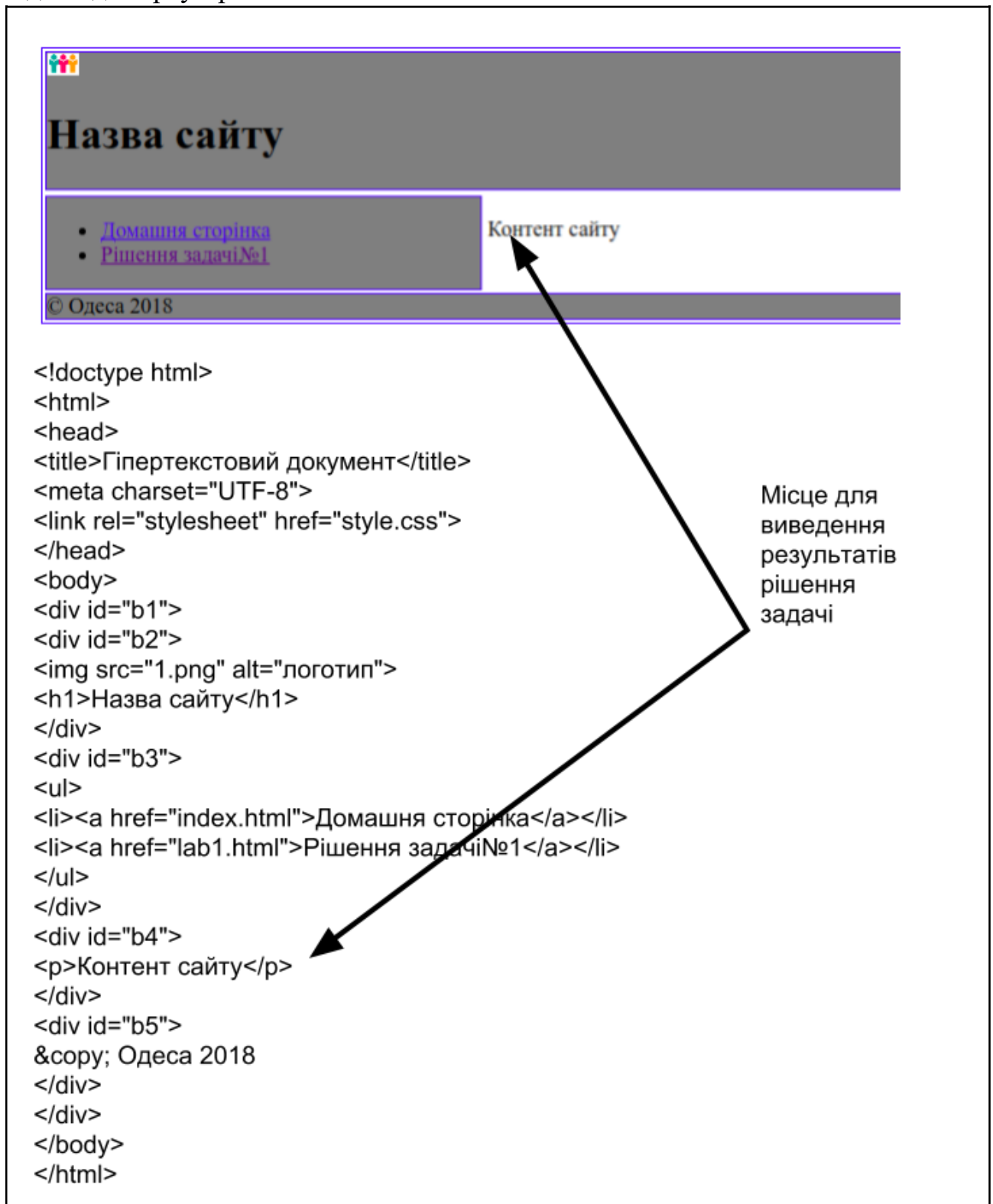


Рис. 1. Визначення місця для виведення результатів рішення php задачі

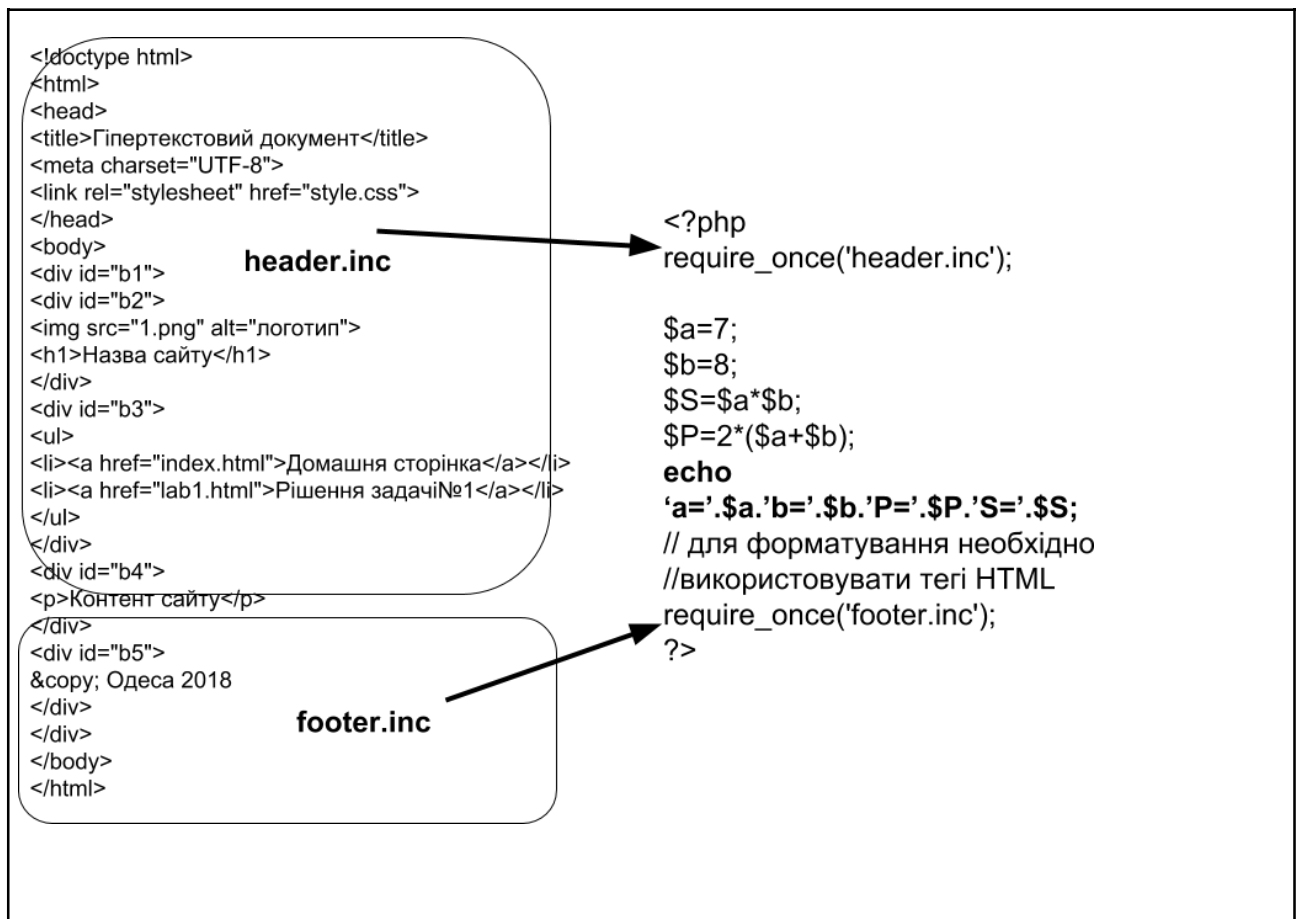


Рис. 2. Виведення результатів в визначене місце в шаблон гіпертекстової сторінки

Роз'єднати ці дві частини дозволяє сучасна технологія використання шаблонизатора.

Шаблонизатор дозволяють домогтися відділення прикладної логіки і даних від логіки представлення. Це дуже зручно в ситуаціях, коли програміст і верстальник шаблону - різні люди. Шаблонизатор сприяє виконанню концепції поділу.

Довгий час для розробки сайтів на РНР використовував шаблонизатор Smarty. Досить зручний шаблонизатор, так би мовити де-факто для РНР. Паралельно розвивалися інші шаблонизатор. Наведемо порівняльну таблицю шаблонизаторів за швидкістю (Таблиця 1) [2].

Twig шаблонизатор для РНР є одним з найбільш зручних і швидкодіючих.

Розгортаємо базові файли та папки системи “MVC-старт”.

Розглядаючи підсумкову структуру файлів та папок системи “MVC-старт” (Рис.5) визначимо їх структурні компоненти:

1. index.php це приклад файлу системи, що відповідає за рішення задачі мовою php, формування масиву результатів, що мають виводитись в гіпертекстовому інтерфейсі та визначає файл-шаблон, в якому буде виведено дані (Рис.6)

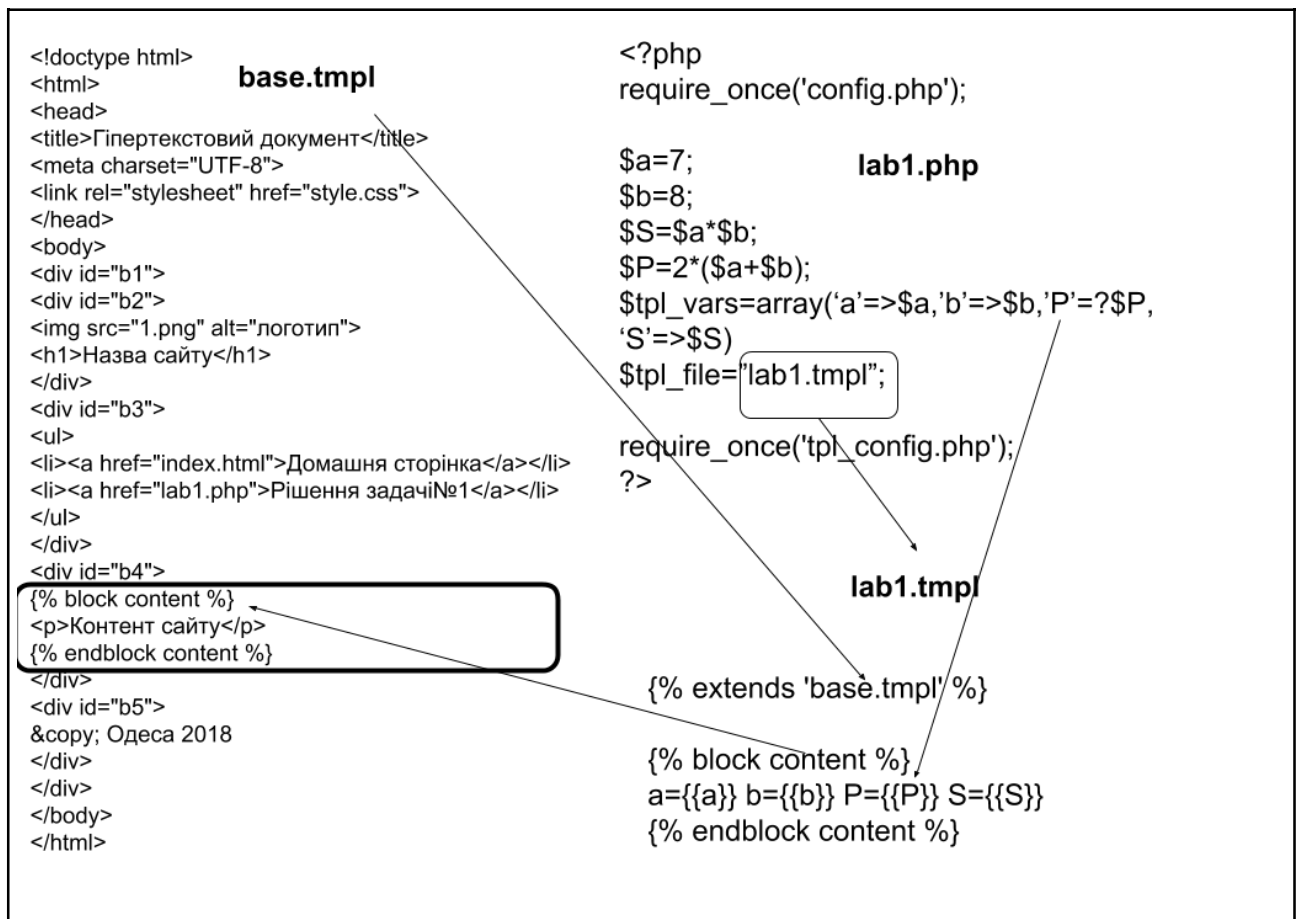


Рис. 3. Виведення результатів в іменованний варіативний блок

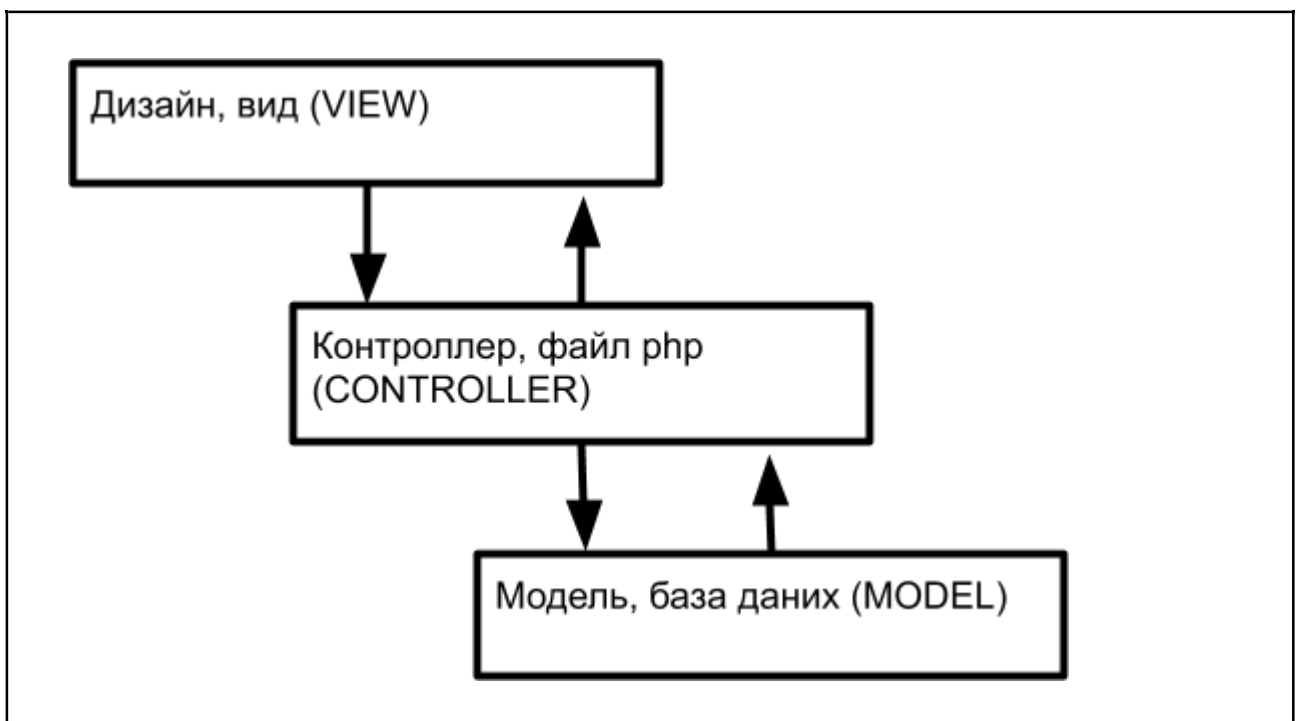


Рис. 4. Схема архітектури MVC

Таблиця 1. Порівняння шаблонізаторів за швидкістю

Бібліотека	Час (сек)	Пам'ять (Кб)
Мова PHP	2.4	114
Twig	3	383
PHPTAL	3.8	598
Dwoo	6.9	1 645
Smarty 2	12.9	610
Smarty 3	14.9	799
Calypso	34.3	614
eZ Templates	53	2 783

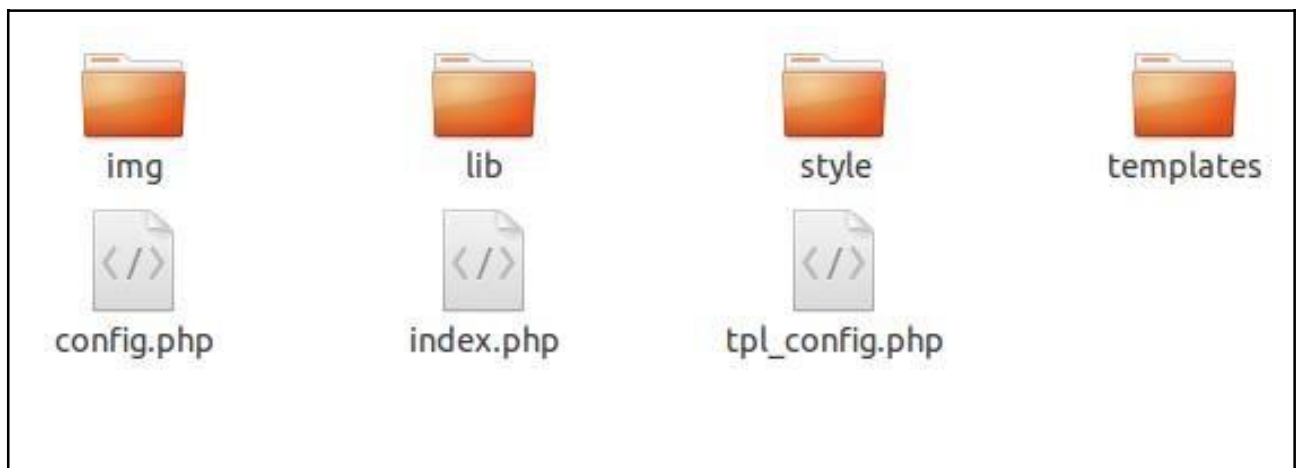


Рис. 5. Файли та папки системи “MVC-старт”



Рис 6. Зміст файлу index.php

```
// Підключення конфігураційного файлу
require_once ('config.php');
/* Нижче визначаємо масив змінних доступних в шаблоні. У першому прикладі
даний масив порожній*/
$tpl_vars = array ();
/* Нижче визначаємо ім'я файлу-шаблону, який знаходиться в папці templates */
$tpl_file = 'index.tpl';
// Запускаємо шаблонізатор і виводимо відповідну сторінку
require_once ('tpl_config.php');
```

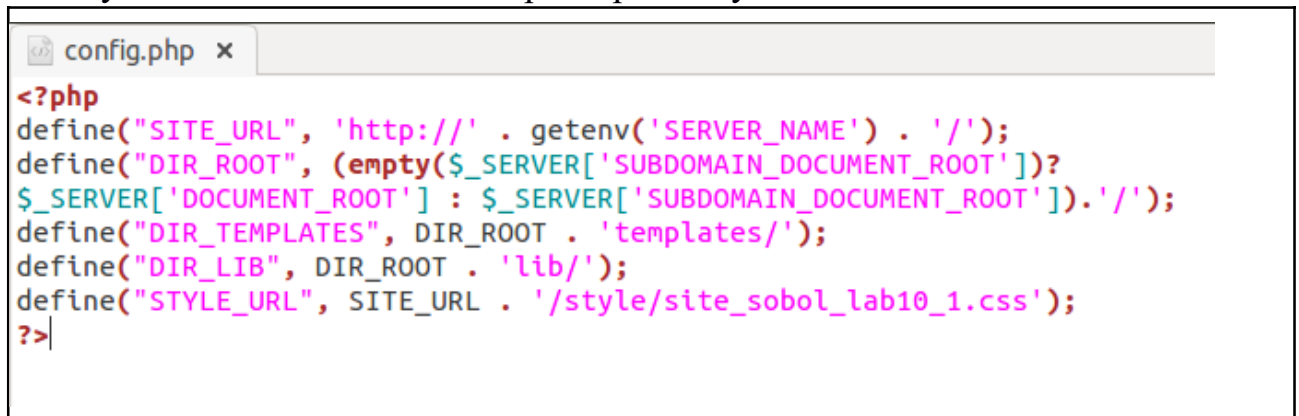
В папці templates повинен бути створений файл index.tpl

2. **config.php** - визначення констант, які пов'язані зі шляхами і адресами сайту, при перенесенні сайту на новий хостинг (Рис.7)

Першим кроком на шляху побудови професійного сайту є створення файлу config.php, в котрому ми налаштовуємо автоматичне визначення URL адреси сайту, що дозволяє без змін в налаштуваннях переносити сайт на новий хостинг. Всі константи, що визначаються в конфігураційному файлі інтуїтивно зрозумілі за своїми назвами. Для їх визначення використано стандартні функції php.

getenv('SERVER_NAME') повертає ім'я хоста з URL адреси сайту.

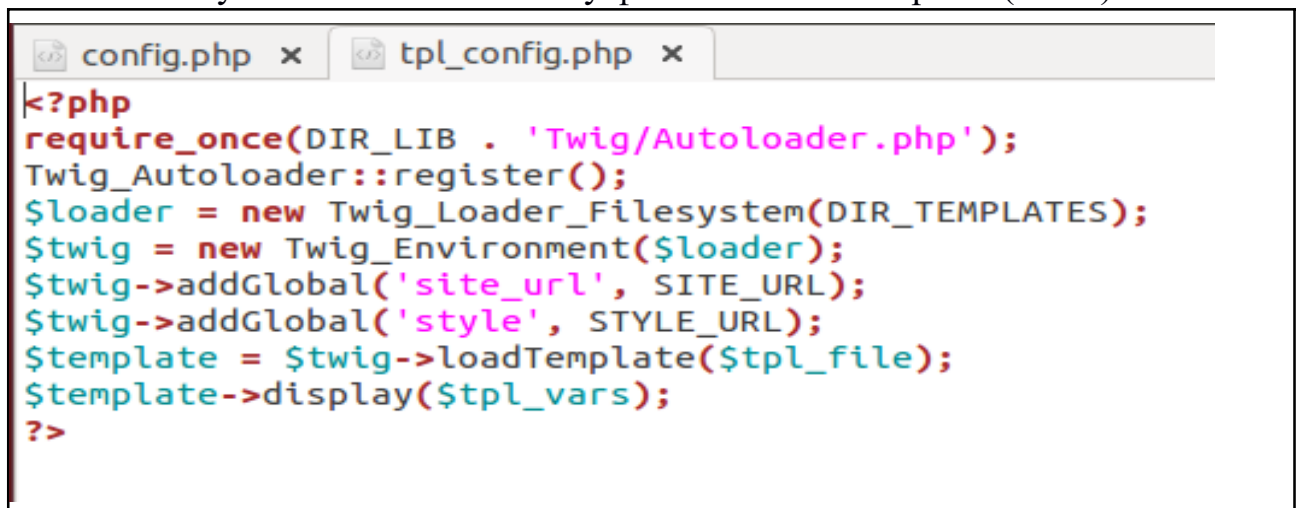
\$_SERVER['SUBDOMAIN_DOCUMENT_ROOT'] з масиву змінних оточення визначаємо повний шлях на сервер до кореневого каталогу сайту (в залежності від налаштувань Web серверу це може бути \$_SERVER['DOCUMENT_ROOT']). У випадку розташування сайту у вкладених папках на хостингу, необхідно дописувати шлях до основної деректорії сайту.



```
<?php
define("SITE_URL", 'http://' . getenv('SERVER_NAME') . '/');
define("DIR_ROOT", (empty($_SERVER['SUBDOMAIN_DOCUMENT_ROOT'])?
$_SERVER['DOCUMENT_ROOT'] : $_SERVER['SUBDOMAIN_DOCUMENT_ROOT']).'/');
define("DIR_TEMPLATES", DIR_ROOT . 'templates/');
define("DIR_LIB", DIR_ROOT . 'lib/');
define("STYLE_URL", SITE_URL . '/style/site_sobol_lab10_1.css');
?>
```

Рис. 7. Стартова структура файлу config.php

3. **tpl_config.php** - визначає базові команди по підключенню бібліотеки "Twig", які повинні бути виконані в кожному файлі частині контролю (Рис.8)



```
<?php
require_once(DIR_LIB . 'Twig/Autoloader.php');
Twig_Autoloader::register();
$loader = new Twig_Loader_Filesystem(DIR_TEMPLATES);
$twig = new Twig_Environment($loader);
$twig->addGlobal('site_url', SITE_URL);
$twig->addGlobal('style', STYLE_URL);
$template = $twig->loadTemplate($tpl_file);
$template->display($tpl_vars);
?>
```

Рис.8. Файл tpl_config.php

Нижче наведено пояснення до команд у файлі tpl_config.php

// Підключаємо необхідні бібліотечні файли

```

require_once(DIR_LIB . 'Twig/Autoloader.php');
//Запускаємо Twig
Twig_Autoloader::register();
//Визначаємо путь до папки з шаблонами
$loader = new Twig_Loader_Filesystem(DIR_TEMPLATES);
/*Визначаємо шлях до папки cache (тимчасові файли), яка має бути відкрита на
запис. Саме в цій папці формуються php файли з шаблонів при першому
запуску.*/
$twig = new Twig_Environment($loader);
/*Два наступних рядка роблять доступними відповідні глобальні константи
конфігураційного файлу у кожному шаблоні*/
$twig->addGlobal('site_url', SITE_URL);
$twig->addGlobal('style', STYLE_URL);
/*Далі визначаємо ім'я змінної, що зберігає шлях до файла-шаблону відносно
папки templates. Змінна $tpl_file має визначатися в кожному php файлі системи
*/
$template = $twig->loadTemplate($tpl_file);
/*Далі виводимо на екран результати, що передано через масив $tpl_vars. Масив
$tpl_vars має бути визначений в кожному php файлі.*/
$template->display($tpl_vars);

```

4 - **lib** - це папка, що містить файли бібліотеки "Twig".

На сайті [Twig](http://twig.symfony.com) (twig.symfony.com) — вказано, що для роботи необхідним є **PHP 5.2.4** і вище.

Далі необхідно встановити бібліотеку Twig, останньою на момент публікації даних матеріалів є версія 1.10.3 (<http://twig.sensiolabs.org/>)[3]. З архіву нам необхідна папка **Twig**, що знаходиться в папці **lib**. Її ми копіюємо в свою папку **lib**. Отримуємо на хостингу:

lib/-

|_ Twig

Необхідно розташувати в корні власного сайту папку **lib** і можна починати роботу з шаблонизатором **Twig**. Папка **lib** має знаходитись в кореневому каталозі сайту.

5. **templates** - це папка, що містить базовий шаблон і шаблони всіх сторінок системи. На старті тут має бути файл **base.tpl** та шаблон **index.tpl** сторінки **index.php** (Рис. 9).

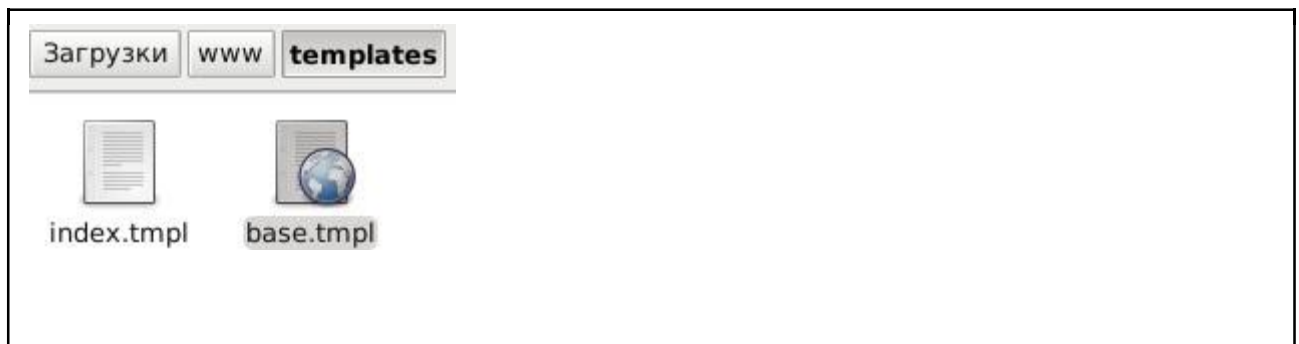


Рис. 9. Стартова структура папки з шаблонами

На рисунку (рис. 10) наведено правила взаємодії файлів частини дизайну і правила підключення дизайну до файлу частини контроллера (index.php)

7. Папки **img** і **style** без зміни переносяться зі статичного варіанту сайту. Для стартової системи MVC-старт папка **img** пуста, а в папці **style** знаходиться файл **style.css**, що визначає стильове оформлення базового шаблону **base.tmpl**.

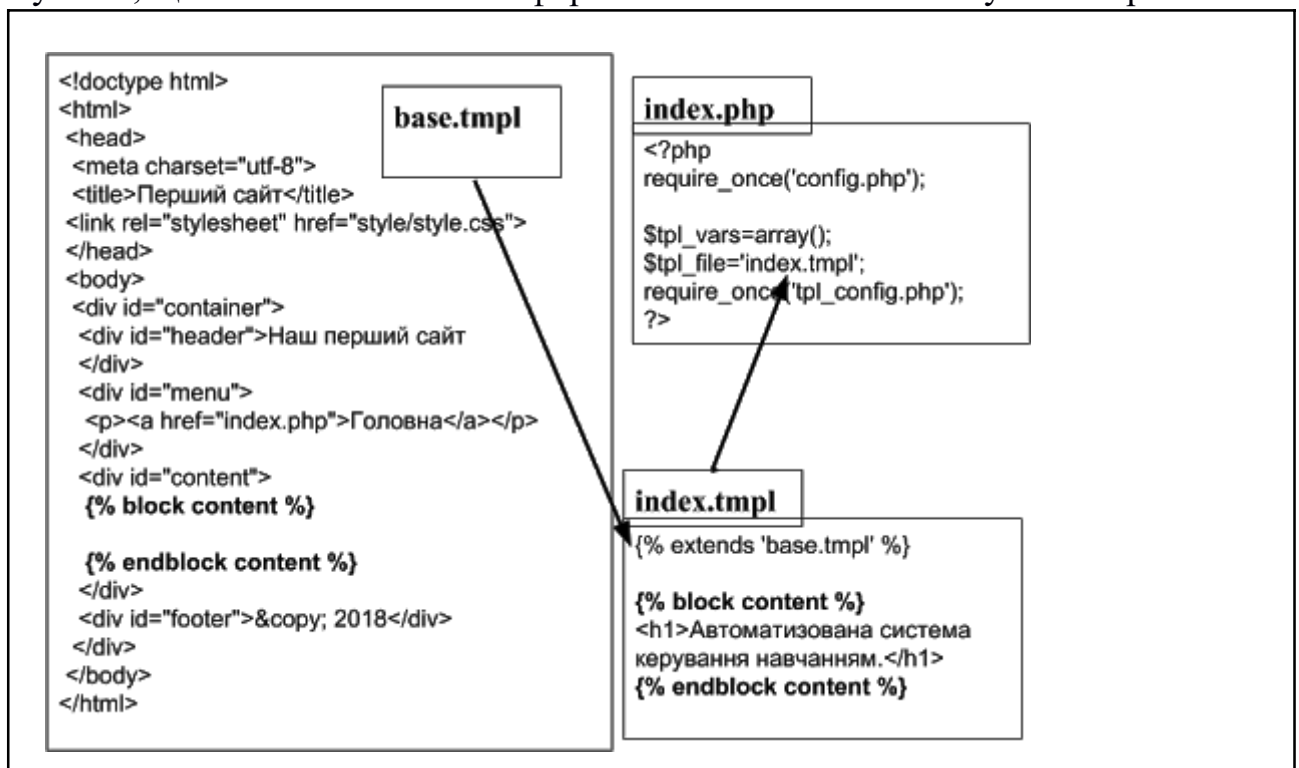


Рис. 10. Правила взаємодії файлів частини дизайну і правила підключення дизайну до файлу частини контроллера

Хід роботи:

1. Зареєструвати власний хостинг (реальний або віртуальний, безкоштовний або платний). Вивчити правила роботи з даним хостингом.
2. Завантажити стартові файли та папки системи “MVC-старт” на хостинг. Посилання для завантаження архіву системи “MVC-старт”: <http://pdpu.edu.ua/doc/mvc/mvc.zip>
3. Створити гіпертекстовий файл **base.html**, що буде покладено в дизайн індивідуальної інформаційної системи. Вивчаємо правила підключення нового

шаблону до системи і налаштування варіативного блоку “content” (рис. 10). Шаблон **base.tmpl**—це базовий шаблон. На його основі будуть розроблятися шаблони до всіх php файлів сайту. Для переведення сайту до роботи в моделі MVC файл base.tmpl розробляється на основі файла base.html з додаванням ряду змін:

3.1. Підключення файла зі стилями в файлі base.html `<link rel="stylesheet" href="style/style.css" >` перетворюється до вигляду `<link rel="stylesheet" href="{{style}}" >`. Зверніть увагу `{{style}}` - це виведення глобальної змінної, яка визначена в файлі `tpl_config.php` (`$twig->addGlobal('style', STYLE_URL);`) (Рис. 7) `STYLE_URL` – це константа, що була визначена в файлі `config.php` (`define("STYLE_URL", SITE_URL . '/style/style.css')`) (Рис. 6);

3.2. В блоці “content” `<div id="content"> </div>` необхідно додати варіативний блок шаблонизатора Twig: `<div id="content"> {% block content %} {% endblock content %}</div>` content – це ім'я блоку і воно може бути довільним часто його назва збігається з ім'ям зовнішнього блоку `div`. Кількість таких динамічних блоків в файлах-шаблонах може бути довільною. Кожен такий блок може бути перевизначений в новому файлі-шаблоні, побудованому на основі шаблону `base.tpl`.

4. Підготувати 5 нових сторінок в системі, що мають імена `lab1-1.php` – `lab1-5.php`. Дані сторінки структуруються по аналогії з файлом `index.php` (Рис.6). Змінити ім'я файлу шаблону на `lab1-1.tmpl` – `lab1-5.tmpl` Для кожної сторінки налаштувати індивідуальний шаблон, на якому інформацію про себе в блоці `content` структуровано з використанням різних об'єктів HTML.

5. Деталізація кожного шаблону:

`lab1-1.tmpl` використати об'єкти заголовков (`h1-h6`) та абзац (`p`)

`lab1-2.tmpl` використати об'єкт список (`ul, ol`) та рисунок (`img`)

`lab1-3.tmpl` використати об'єкт таблиця (`table`)

`lab1-4.tmpl` використати об'єкти заголовков (`h1-h6`), рисунок (`img`) та & послідовність

`lab1-5.tmpl` використати об'єкти заголовков (`h1-h6`) та таблиця (`table`)

6. В блоці `menu` базового шаблону власної інформаційної системи додати посилання на сторінки, що було створено.

Література

1. Влад Мержевич. URL: <http://htmlbook.ru/> (дата звернення 23.11.2016)
2. Fabien Potencier. Шаблонизаторы в PHP. URL: <https://habrahabr.ru/post/75901/> (дата звернення 23.11.2016)
3. Twig: The flexible, fast, and secure template engine for PHP. URL: <http://twig.sensiolabs.org/> (дата звернення 23.11.2016)
4. Введение в MVC для интернет-разработок. URL: <http://bourabai.kz/dbt/mvc.htm> (дата звернення 23.11.2016)

Лабораторна робота №2

Об'єктно-орієнтовані основи мови php. Клас PDO. Налаштування роботи з базою даних

Мета: Опанування методів класа PDO для підключення до бази даних та опрацювання простих запитів до бази даних. Придбання практичних навичок формування шаблону «Таблиця» виведення результатів SELECT запиту.

Теоретичний матеріал

Клас PDO для підключення до бази даних та опрацювання запитів до бази даних. PDO - це клас для роботи з СУБД. У PDO спосіб з'єднання DSN.

Приклад 1. Налаштування з'єднання з базою даних

```
define("DB_HOST", "mysql.zzz.com.ua"); //визначаємо хост серверу БД
define("DB_USERNAME", "sasha123123"); //ім'я користувача бази даних
define("DB_PASSWORD", "Sasha123"); //пароль користувача бази даних
define("DB_NAME", "dimitrashko97"); //імя бази даних

$dbh = new
PDO('mysql:host='.DB_HOST.';dbname='.DB_NAME,DB_USERNAME,DB_PAS
SWORD); //створюємо об'єкт класа PDO
$dbh->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION); //налаштування опрацювання помилок
$dbh->exec("set names utf8"); //налаштування кодування об'єкта
```

Обробка винятків

Часто рекомендується при роботі з класом PDO користуватись конструкцією try-catch для оброблення винятків PDOException. Це можна опускати, PHP коректно виводить повідомлення про помилки, нам для роботи цього достатньо.

Виконання запитів.

Для виконання запитів можна користуватися двома методами.

1. Якщо в запит не передаються ніякі змінні, то можна скористатися функцією query(). Вона виконає запит і поверне спеціальний об'єкт - PDO statement. Отримані дані будуть переведені у формат двовимірного асоціативного масиву методом fetchAll(PDO::FETCH_ASSOC).

Приклад 2. Якщо в SQL запит не передаються ніякі змінні

```
$stmt = $dbh-> query ( 'SELECT * FROM users');
$result = $stmt->fetchAll(PDO::FETCH_ASSOC); //отримаємо масив
//одновимірних масивів з ключами-іменами атрибутів таблиці users
```

2. Якщо в запит передаються змінні необхідно підготувати запит, а другою дією підставити підготовлені змінні. Отримані дані будуть переведені у формат двовимірного асоціативного масиву методом fetchAll(PDO::FETCH_ASSOC).

Приклад 3. Якщо в SQL запит передаються змінні

```
$stmt = $dbh-> prepare ( 'SELECT * FROM users WHERE id=:id and name=:name1');  
$stmt->execute(['id' => $id, 'name1' => $name1]); // визначаємо :id та :name1  
// та виконуємо SQL запит  
$result = $stmt->fetchAll(PDO::FETCH_ASSOC); // отримаємо масив  
// одновимірних масивів з ключами-іменами атрибутів таблиці users, що  
// відповідає умові SQL запиту
```

Хід роботи:

1. Розглянемо приклад роботи з СУБД MySQL у середовищі PHP MyAdmin. На хостингу в панелі керування необхідно створити базу даних, до неї користувача, що має ім'я та пароль і зафіксувати шлях до хоста баз даних (Рис.1).



Рис.1. Заведення бази даних і користувача (Приклад для хостингу zzz.com.ua)
Дана інформація має бути відображена в відповідних константах конфігураційного файлу, що наведено у прикладі

Приклад 4. Налаштування констант конфігураційного файлу config.php

```
define("DB_HOST", "mysql.zzz.com.ua");  
define("DB_USERNAME", "sha123123");  
define("DB_PASSWORD", "Sa6667896");  
define("DB_NAME", "dimit");
```

2. Створити таблицю бази даних та наповнити її щонайменше 5 записами.

Для цього ми скористаємось SQL запитом CREATE та INSERT

Приклад 5. Створюємо таблицю users з наповненням

```
CREATE TABLE `news` (
```

```

`id` int(20) UNSIGNED NOT NULL,
`name` varchar(255) NOT NULL,
`description` text NOT NULL,
`data` date NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

INSERT INTO `news` (`id`, `name`, `description`, `data`) VALUES
(1, 'Новина №1', 'Детальна інформація №1', '2018-09-22'),
(2, 'Новина №2', 'Детальна інформація №2', '2018-09-11'),
(3, 'Новина №3', 'Детальна інформація №3', '2012-11-19'),
(4, 'Новина №4', 'Детальна інформація №4', '2012-11-13'),
(5, 'Новина №5', 'Детальна інформація №5', '2012-11-01');

```

Наведений у прикладі SQL код запитів запускаємо в середовищі PHP MyAdmin у розділі SQL (Рис. 2)

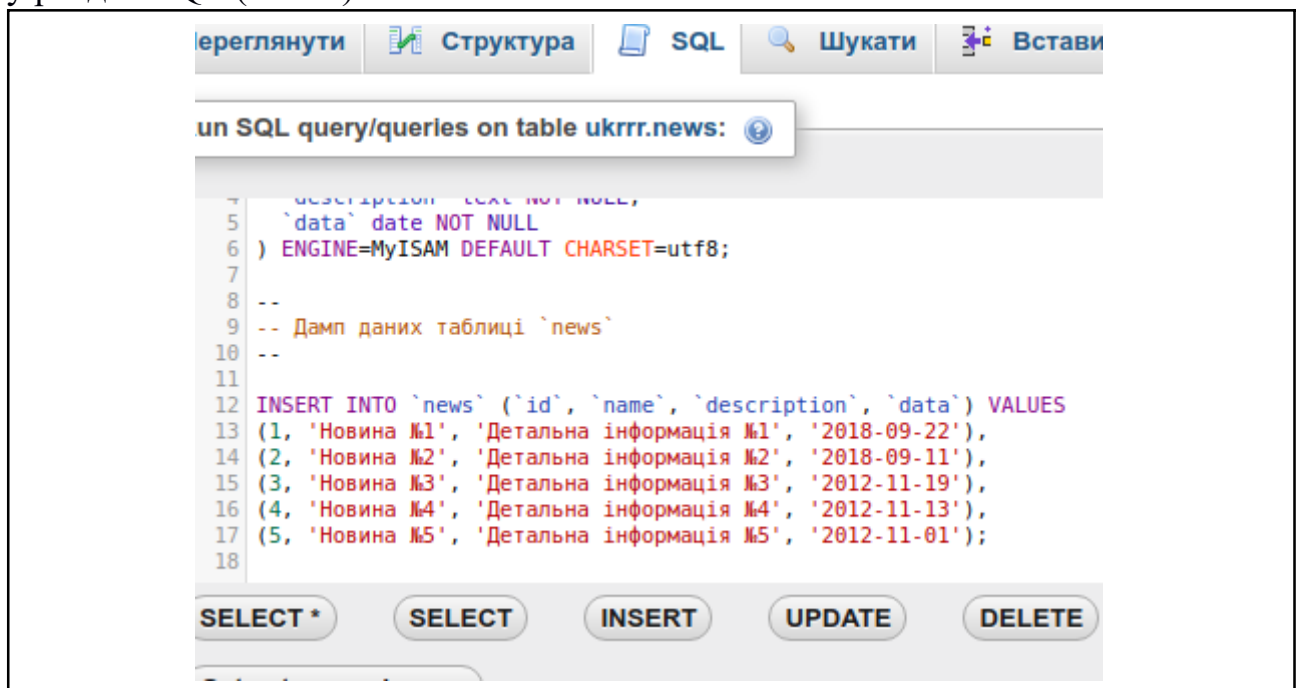


Рис.2. Виконання команд CREATE та INSERT в середовищі PHP MyAdmin
Після виконаних дій База даних з таблицею налаштована. Нам можна приступати до формування сторінок інформаційної системи на основі даних з таблиці бази даних.

3. Налаштувати конфігураційний файл для роботи з базою даних.

Першим етапом для організації роботи з базою даних в інформаційній системі стає налаштування конфігураційного файлу. Тут ми визначаємо константи, що забезпечують підключення до бази даних, а також функцію, в котрій формується об'єкт класу PDO, що опрацьовує роботу з базою даних. Єдиною різницею Прикладаб від Приклада1 стає оформлення з'єднання з базою даних в окрему функцію. Це необхідно тому, що конфігураційний файл підключається до кожного файлу частини контролеру, але не завжди ми потребуємо з'єднання

з базою даних.

Приклад 6

```
define("DB_HOST", "mysql.zzz.com.ua");
define("DB_USERNAME", "sha123123");
define("DB_PASSWORD", "Sa6667896");
define("DB_NAME", "dimit");
function db_connect(){
    $dbh = new
PDO('mysql:host='.DB_HOST.';dbname='.DB_NAME,DB_USERNAME,
DB_PASSWORD);
    $dbh->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $dbh->exec("set names utf8");
    return $dbh;
}
```

4. Запустити у власній інформаційній системі приклад 7, що наведено нижче

Після налаштування конфігураційного файлу в наведений спосіб розглянемо приклад рішення задачі на вибірку всіх записів з таблиці news бази даних. Файл частини контролера lab10.php, в якому виконується виклик функції для підключення бази даних, опрацювання запиту SQL, формування асоціативного масиву-результату та передача результату, що отримано в шаблон.

Приклад 7(lab10.php)

```
<?php
require_once('config.php');
$dbh = db_connect();//підключення до БД
$stmt = $dbh-> query('SELECT * FROM news');// Виконується SQL запит
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);// Результат в масив
$tpl_vars = array('result' => $result);
$tpl_file = 'lab10.tpl';
require_once('tpl_config.php');
?>
```

Відповідно сформуємо шаблон lab10.tpl для виведення результатів у вигляді таблиці. Тут використано два вкладених цикли. Зовнішній - по елементах масиву result, а внутрішній обробляє один кортеж - результат.

Приклад 7 (lab10.tpl)

```
{% extends 'base.tpl' %}
{% block content %}
<table border="5">
{% for user_value in result %} //result - переданий з контроллера результат
<tr>
```

```
{% for key, value in user_value %} //user_value - це черговий рядок таблиці
<td>
{{value}} //виводимо значення чергового атрибута в рядку таблиці news
</td>
{% endfor %} //кінець внутрішнього циклу
</tr>
{% endfor %} // кінець зовнішнього циклу
</table>
{% endblock content %}
```

У результаті отримаємо дані з таблиці news у табличному форматі (рис. 3)

Наш перший сайт

[Головна](#)
[Лабораторна робота №2](#)
[Лабораторна робота №3](#)
[Лабораторна робота №6](#)
[Лабораторна робота №8](#)
[Адмін розділ](#)

1	Новина №1	Детальна інформація №1	2018-09-22
2	Новина №2	Детальна інформація №2	2018-09-11
3	Новина №3	Детальна інформація №3	2012-11-19
4	Новина №4	Детальна інформація №4	2012-11-13
5	Новина №5	Детальна інформація №5	2012-11-01

© 2016

Рис. 3. Результат вибірки з таблиці у табличному шаблоні

Важливим є той факт, що для отримання результату любого SQL запиту на вибірку інформації достатньо в частині контролера поміняти SELECT запит.

5. Хід виконання індивідуальн завдання

- 1- За своїм порядковим номером у групі вибрати варіант індивідуального завдання
- 2- Побудувати таблиці бази даних предметної галузі за своїм варіантом, встановити відповідні зв'язки з урахуванням посильної цілісності та цілісності сутностей
- 3- Наповнити базу даних даними з відповідної предметної галузі. Щонайменше по 5 записів в кожену таблицю. При наповненні враховуйте посильну цілісність!
- 4- Створити 3 сторінки сайту, що з використанням одного шаблону представляють результат на вибірку всіх даних з кожної таблиці бази даних у табличному форматі.
- 5- В головне меню додати посилання, що відкривають 4 сторінки системи з представленими в табличному форматі даними таблиць (1-дані з таблиці-

прикладу+ Зсторінки – дані з таблиць індивідуальної бази даних).

6. Варіанти індивідуальних завдань

1. Побудувати базу даних на основі якої можна зберігати дані про успішність кожного учня школи в кожному класі на протязі всіх років навчання.

учень(код, повне_ім'я, дата_народження, рік_перший)

предмет(код, назва, клас)

успішність(код, код_учень, код_предмет, семестр, оцінка)

2. Побудувати базу даних на основі якої можна зберігати дані про позакласні події, що проведено кожним учителем на протязі всіх років навчання.

учитель(код, повне_ім'я, дата_народження, рік_перший, спеціальність)

види_заходів(код, назва, опис)

проведення_заходів(код, код_учителя, код_заходу, дата, місце, учасники, результати)

3. Побудувати базу даних на основі якої можна зберігати дані про позакласні гуртки, що відвідував кожен учень в кожному класі на протязі всіх років навчання.

учень(код, повне_ім'я, дата_народження, рік_перший)

гуртки(код, назва, керівник)

відвідування(код, код_учень, код_гуртка, навчальний_рік)

4. Побудувати базу даних на основі якої можна зберігати дані про хвороби і термін перебування на лікарняному кожного учня школи в кожному класі на протязі всіх років навчання.

учень(код, повне_ім'я, дата_народження, рік_перший)

хвороби(код, назва, катигорія)

реєстрація_хвороб(код, код_учень, код_хвороби, дата1, дата2)

5. Побудувати базу даних на основі якої можна зберігати дані про участь в олімпіадах та конкурсах кожного учня школи в кожному класі на протязі всіх років навчання.

учень(код, повне_ім'я, дата_народження, рік_перший)

конкурси(код, назва, місце, дата)

участь_конкурс(код, код_учень, код_конкурси, дата, результат)

6. Побудувати базу даних на основі якої можна зберігати дані про суспільні обов'язки кожного учня школи в кожному класі на протязі всіх років навчання.

учень(код, повне_ім'я, дата_народження, рік_перший)

обов'язки(код, назва, клас)

виконання_обов'язків(код, код_учень, код_обов'язки, рік, семестр, оцінка)

7. Побудувати базу даних на основі якої можна зберігати дані про підвищення кваліфікації вчителів на протязі всіх років роботи.

учитель(код, повне_ім'я, дата_народження, рік_перший, спеціальність)

програма_підвищення(код, назва, місце, дата1, дата2)

підвищення_кваліфікації(код, код_учителя, код_програма, дата1, дата2, результати)

8. Побудувати базу даних на основі якої можна зберігати дані про проходження педагогічної практики студентів.

студент(код, повне_ім'я, дата_народження, рік_перший, спеціальність)

школи(код, назва, місце, керівник, фах)

практика(код, код_студента, код_школи, дата1, дата2, результати)

9. Побудувати базу даних на основі якої можна зберігати дані про участь студентів у позакласних подіях університету.

студент(код, повне_ім'я, дата_народження, рік_перший, спеціальність)

події(код, назва, місце, керівник, дата)

участь(код, код_студента, код_події, результати)

10. Побудувати базу даних на основі якої можна зберігати дані про навчальні плани для підготовки фахівців з різних спеціальностей.

предмети(код, назва, всього_годин, аудиторні, лекційні, самостійні)

спеціальності(код, назва, факультет)

плани(код, код_предмета, код_спеціальності, семестр)

11. Побудувати базу даних на основі якої можна зберігати дані про розподіл навчального навантаження між кафедрами.

плани(код, код_предмета, код_спеціальності, семестр)

кафедра(код, назва, факультет)

навантаження(код, код_плани, код_кафедри, навчальний_рік)

12. Побудувати базу даних на основі якої можна зберігати дані про розподіл навчального навантаження між викладачами кафедри.

навантаження(код, код_плани, код_кафедри, навчальний_рік)

викладач(код, повне_ім'я, дата_народження, рік_перший, спеціальність)

навантаження_викладач(код, код_навантаження, код_викладача, навчальний_рік, кількість годин)

Література

1. Web-технологии. Основы языка PHP. URL:<https://htmlweb.ru/php/php2.php> (дата звернення 22.11.2016)
2. Довідник з MySQL. URL:<http://www.mysql.ru/docs/man/SELECT.html> (дата звернення 22.05.2019)
3. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч.посібник. – Електронне видання, 2018. – 118 с. URL:<https://dspace.uzhnu.edu.ua/jspui/handle/lib/19776> (дата звернення 22.05.2016)
4. Титенко С. В. Структурные основы онтологически-ориентированной системы управления информационно-учебным Web-контентом / С. В. Титенко // Управляющие системы и машины. - 2012. - № 2. - С. 35-42. - Режим доступа: http://nbuv.gov.ua/UJRN/USM_2012_2_6

Лабораторна робота №3

Опрацювання більш складних запитів до бази даних. Повторення мови SQL.

Мета: Придбання практичних навичок використання SELECT запиту при побудові сторінок інформаційної системи.

Теоритичний матеріал

Загальний вигляд SELECT запиту (основи)

SELECT

[DISTINCT | DISTINCTROW | ALL]

select_expression

[FROM table_references

[WHERE where_definition]

[GROUP BY {unsigned_integer | col_name | formula} [ASC | DESC], ...]

[HAVING where_definition]

[ORDER BY {unsigned_integer | col_name | formula} [ASC | DESC], ...]

[LIMIT [offset,] rows]

Формування запитів до двох таблиць з використанням конструкції JOIN.

В даній лабораторній роботі в першу чергу необхідно згадати базові правила щодо формування запиту до 2-х таблиць. Представимо, що в таблиці, що має назву А первинний ключ - це атрибут, що має назву PK. Таблиця, що має назву В посилається на таблицю А за допомогою зовнішнього ключа на ім'я FK. За таких стартових умов надаються приклади вибірки перетину, лівого та правого об'єднання відношень. Відповідні SQL запити представлені на рисунку (рис.1).

Формування обмеження результуючих кортежей SELECT запиту за допомогою конструкції WHERE.

У простих умовах значення одного виразу (зазвичай значення відповідного атрибуту кожного кортежа) порівнюється із значенням іншого виразу (зазвичай конкретне значення). Прості умови відбору, після застосування до деякого атрибуту повертають для кожного кортежа значення TRUE, FALSE або NULL. За допомогою правил логіки ці прості умови можна об'єднувати в більш складні, використовуючи при цьому логічні операції AND, OR, NOT.

Операції для побудови простих запитів:

- Операції порівняння для числових даних: = , < > , < , <= , > , >=.
- Операції порівняння для даних типу дата: BETWEEN, NOT BETWEEN
- Операції порівняння для строкового типу: LIKE шаблон або ім'я стовпця NOT LIKE шаблон. Для побудови шаблонів використовуємо: % співпадає з будь-якою послідовністю з нуля чи більше символів; _ (символ підкреслення) співпадає з будь-яким окремим символом; ESCAPE символ пропуску.
- Операції порівняння значення атрибуту з NULL: ім'я стовпця IS NULL або ім'я стовпця IS NOT NULL.

- Перевірка приналежності до множини значень: вираз, що перевіряється IN (список констант відокремлених комами) або вираз, що перевіряється NOT IN (список констант відокремлених комами)

Для впорядкування результатів запиту використовується конструкція ORDER BY. ORDER BY ім'я_атрибуту ASC/DESC. При впорядкуванні можна обирати зростаючий (ASC) або спадний (DESC) порядок. За замовчуванням дані сортуються по зростанню. При наявності сортування по 2-м та більше атрибутам в першу чергу все відсортується за даними по першому атрибуту, а для однакових даних цього атрибуту буде виконано сортування по другому атрибуту і так далі.

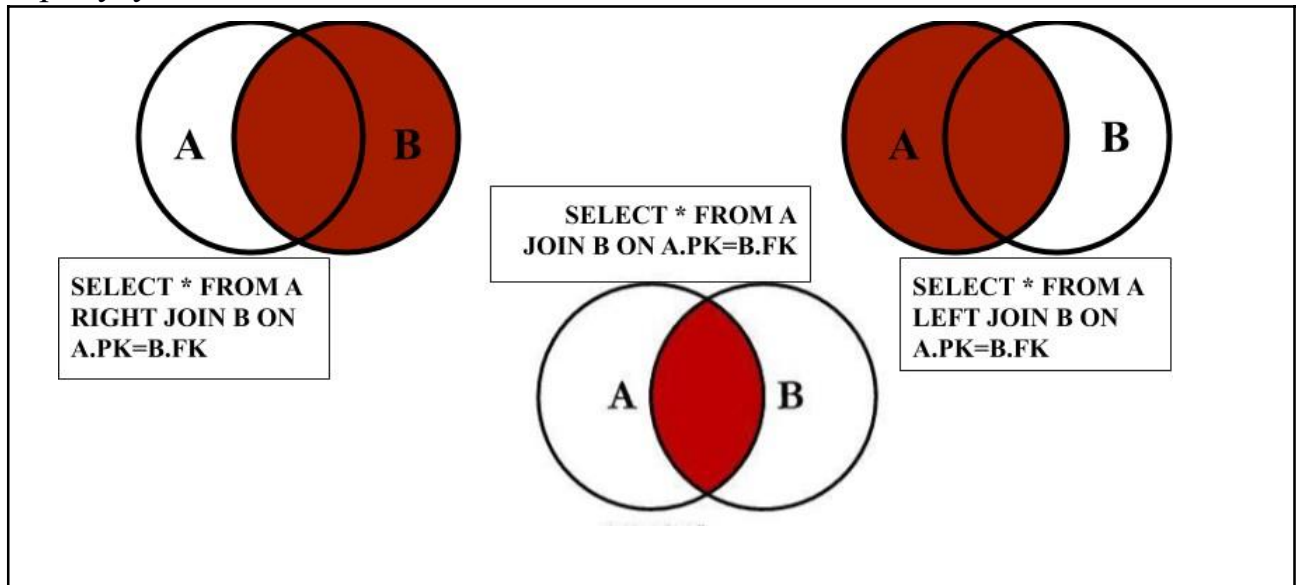


Рис. 1. Базові правила об'єднання відношень

Формування запитів на основі агрегатних функцій.

Для проведення підсумків по інформації, яка міститься в базі даних, застосовуються агрегатні функції. Агрегатна функція приймає в якості аргументу будь-який атрибут (стовпець даних), а повертає одне значення. Базові агрегатні функції: `sum(стовпець/вираз)` обчислює суму всіх значень, `avg(стовпець/вираз)` обчислює середнє всіх значень, `min(стовпець/вираз)` знаходить найменше серед всіх значень, `max(стовпець/вираз)` знаходить найбільше серед всіх значень, `count(стовпець)` підраховує кількість значень, що містяться в стовпці `count(*)` підраховує кількість рядків в таблиці - результату запиту. Приклад1

```
SELECT COUNT(*) FROM A//повертає число - кількість кортежей в A
```

Запити з групуванням (блок GROUP BY) Запит, який включає в себе блок GROUP BY, називається запитом з групуванням, оскільки він об'єднує рядки початкових таблиць в групи. Групування має бути виконано по всім атрибутам результату SELECT запиту за умови наявності одної або більше агрегатних функцій. Приклад2

```
SELECT country, COUNT(*) FROM A GROUP BY (country)//повертає таблицю
```

- назва country та кількість кортежей, з такою назвою country

Умови відбору груп (HAVING). Так само як WHERE використовується для відбору окремих рядків. Блок HAVING необхідно використовувати для формування умов, в котрих фігурують результати обчислень агрегатних функцій. Приклад3

```
SELECT country, COUNT(*) FROM A  
GROUP BY (country)  
HAVING COUNT(*)>5;
```

повертає таблицю - назви country, для яких кількість кортежей, з такою назвою country перевищує 5

Хід роботи:

1. До бази даних індивідуального завдання другої лабораторної роботи формуємо запити на вибірку інформації наступного характеру:

1.1. (Три запити). З включенням в запит всіх трьох таблиць сформувати три SQL запити SELECT, що реалізують використання відповідно конструкцій Left Join, Join, Right Join. Сформувати три запити, що надають різні результати (для отримання різних результатів виконати відповідні зміни в заповненні бази даних)

1.2. (Один запит). На основі любого одного запиту з попереднього підпункту (підпункт №1) сформувати 1 новий запит шляхом додавання обмеження Where, Order by. Сформулювати для даного запиту текстове пояснення, що відповідає на питання: "Що саме вибрано з бази даних предметної галузі?"

1.3. (Три запити). На основі любого одного запиту з підпункту №1 сформувати 3 нових запит шляхом використання 1- агрегатної функції, 2- агрегатної функції та конструкції Group by, 3-агрегатної функції та конструкції Having. Сформулювати для даних запитів текстове пояснення, що відповідає на питання: "Що саме вибрано з бази даних предметної галузі?"

2. На основі розроблених SELECT запитів **побудувати 7 сторінок інформаційної системи** з використанням шаблону табличного виведення інформації в відповідності до попередньої лабораторної роботи

3. В головне меню додати посилання, що відкривають 7 сторінок системи.

Література

1. Довідник з MySQL. URL:<http://www.mysql.ru/docs/man/SELECT.html> (дата звернення 22.05.2019)
2. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. – Електронне видання, 2018. – 118 с. URL:<https://dspace.uzhnu.edu.ua/jspui/handle/lib/19776> (дата звернення 22.05.2016)

Лабораторна робота №4

Параметричні запити до бази даних. Налаштування пошуку даних в базі даних.

Мета: Формування практичного досвіду використання методів класу PDO для опрацювання параметричних запитів. Аналіз станів обробки параметричних запитів.

Теоретичний матеріал

Методи класу PDO для опрацювання SQL запитів з параметром. Якщо в запит передаються змінні необхідно підготувати запит, а другою дією підставити підготовлені змінні. Отримані дані будуть переведені у формат двовимірною асоціативного масиву методом `fetchAll(PDO::FETCH_ASSOC)`.

Приклад 3. Якщо в SQL запит передаються змінні

```
$stmt = $dbh-> prepare ( 'SELECT * FROM users WHERE id=:id and name=:name1');
$stmt->execute(['id' => $id, 'name1' => $name1]); //визначаємо :id та :name1
//та виконуємо SQL запит
$result = $stmt->fetchAll(PDO::FETCH_ASSOC); //отримаємо масив
//одновимірних масивів з ключами-іменами атрибутів таблиці users, що
//відповідає умові SQL запиту
```

Значенням параметру SQL запиту може бути значення довільна змінної програми частини контроллера.

Аналіз станів роботи пошукової системи Google

Класичним варіантом формування параметричного запиту до бази даних є пошук даних, що відповідають якомусь критерію. Проводячи паралель з реальними Web-орієнтованими системами розглянемо пошукову систему Google. Дана система на стартовій сторінці пропонує форму для введення критерію пошуку по мережі Інтернет. Насправді роботи Google індексують всі ресурси мережі та додають їх в єдину базу даних, а далі виконується стандартний SELECT запит до таблиць бази даних, в котрі внесено проіндексовані Web-ресурси мережі Інтернет. Сторінка пошукової системи Google відкривається з двох станів: Стан1 - це форма для введення критерію пошуку; Стан2 - це результати Select запиту за критерієм, що було введено (рис.1).

Ми можемо бачити, що критерій пошуку, що введено у форму Стану1 передається на оброблення методом `get` через приєднання змінних критерію пошуку до URL адреси (рис.2). Треба зазначити, що Стан1 і Стан2 роботи пошукової системи можна обробляти в одному файлі частини контроллера або в різних. В межах нашого курсу ми обробляємо дані стани в одному файлі частини контроллера.

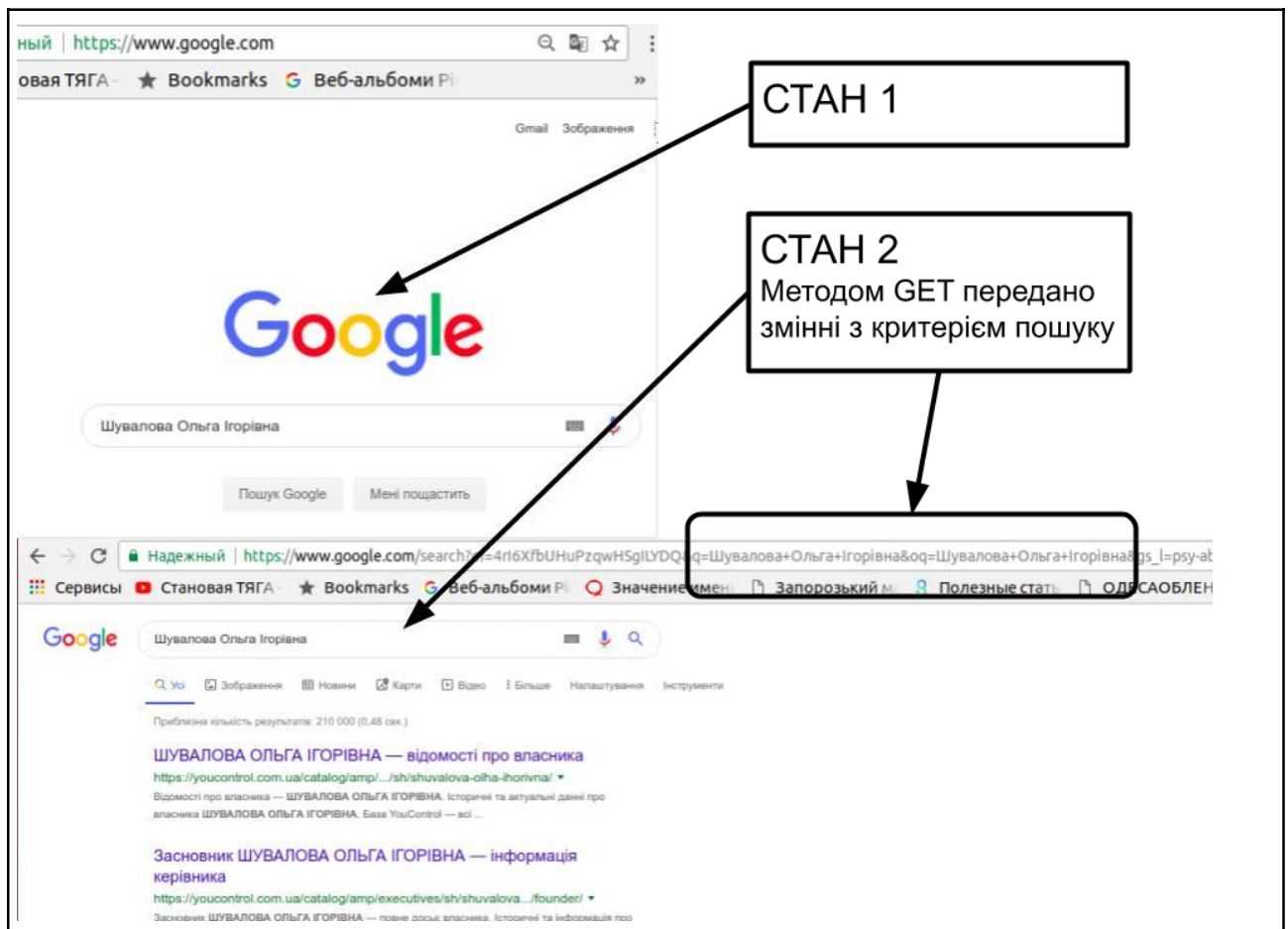


Рис.1. Стани роботи пошукової системи Google

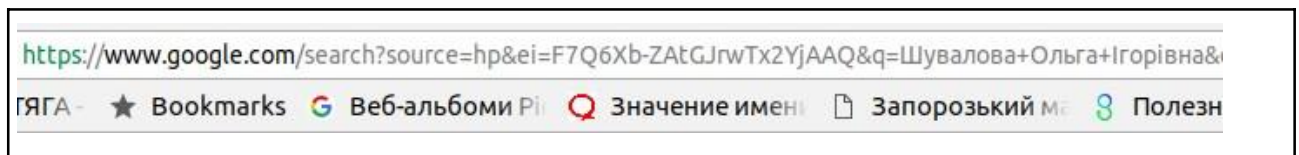


Рис. 2. Адресна строка Google пошуку при передачі даних з форму на оброблення

Отримання зовнішніх даних - це передача даних методами post та get.

`$_GET` - асоціативний масив змінних, що передаються скрипту через параметри URL (відомі також як рядок запиту).

Приклад 1

URL/?t=1&b=2

В файлі, що відкривається з URL (наприклад lab8.php) доступні

`$t=$_GET['t'];`//має значення 1

`$b=$_GET['b'];`//має значення 2

Якщо форма гіпертекстового документа пересилає дані методом get, то програма-обробник їх отримує через параметри URL адреси, що зчитуються в масиві `$_GET`

```
<form method="get" action="lab8.php">
```

```
<input type="text" name="t" value="1">
<input type="text" name="b" value="2">
</form>
```

Тоді в файлі lab8.php доступні
\$t=\$_GET['t'];//має значення 1
\$b=\$_GET['b'];//має значення 2

\$ _POST - асоціативний масив змінних, що передаються скрипту стандартним вхідним потоком.

Приклад 2. Якщо форма гіпертекстового документа пересилає дані методом post, то програма-обробник їх отримує стандартним вхідним потоком, з якого зчитуються дані в масиві \$ _POST

```
<form method="post" action="lab8.php">
<input type="text" name="t" value="1">
<input type="text" name="b" value="2">
</form>
```

Тоді в файлі lab8.php доступні
\$t=\$_POST['t'];//має значення 1
\$b=\$_POST['b'];//має значення 2

Хід роботи:

1. Реалізувати в індивідуальній інформаційній системі приклад рішення задачі.

Для прикладу рішемо задачу пошуку запису в таблиці news нашої бази даних, що було створено в Лабораторній роботі №2, за введеним номером новини.

Нам необхідно створити файл lab12.php, що працює у двох станах:

Стан1 - пропонуємо форму для введення ідентифікаційного номеру новини;

Стан2 - відображаємо результат опрацювання критерію вибору з таблиці news.

На рисунку зображено результуючий вигляд станів рішення задачі-прикладу на пошук інформації в інформаційній системі (рис.3)

Розглянемо детально хід рішення даної задачі.

При формуванні частини контролеру, а саме файлу lab12.php, нами вибрано за критерій Стану2 визначеність зовнішньої змінної id, що передається з форми методом get. Перший запуск файлу lab12.php працює в режимі Стан1. В даному стані ми передаємо в шаблон пусте значення result (результату вибірки з бази даних) (рис.4).

Після заповнення форми і натиснення кнопки Submit в файл lab12.php приходять дані з асоціативного масиву \$_GET, а саме значення \$_GET['id'] та \$_GET['sub'], в форматі приєднаного до URL адреси потоку (рис.3). За таких умов файл частини контролеру працює в режимі Стан2. Загальний вигляд файлів частини контролеру та шаблон(частина представлення) наведено на рисунку (рис.4)

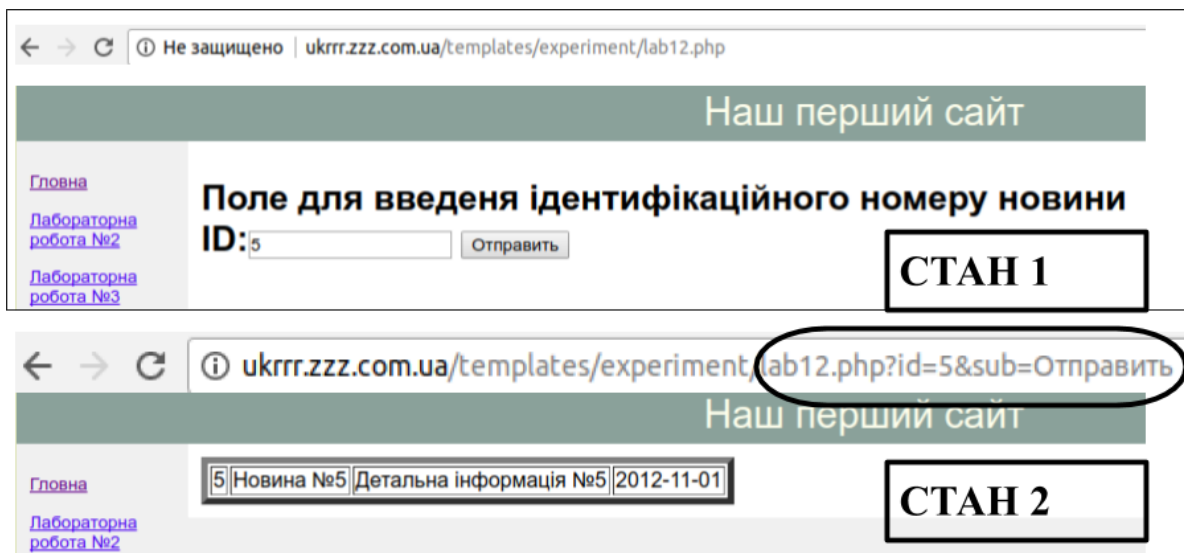


Рис. 3. Результуючий вигляд пошукової системи з урахуванням станів

Розглянемо детально програмний код, що виконується саме в Стані2:

```
$id_news=(int)$_GET['id']; //значення, що прийшло з форми в $id_news
$dbh = db_connect();//підключаємось до БД
$stmt = $dbh->prepare ( 'SELECT * FROM news WHERE id=
:id_news');//підготовляємо SQL запит
$stmt->execute(['id_news' => $id_news]);//підставляємо значення замість
параметру та виконуємо запит
$result = $stmt->fetchAll(PDO::FETCH_ASSOC); //формуємо асоціативний
масив-результат
$tpl_vars = array(
'x'=>$_SERVER['PHP_SELF'],
'result' => $result);
//формуємо масив змінних, що передаємо в шаблон
```

2. В головному меню індивідуальної інформаційної системи **створити посилання, що відкриває реалізований у вашій системі приклад.**

3. Завдання для самостійної роботи.

За аналогією з прикладом організувати розширений пошук по таблицях бази даних, що було розроблено в межах Лабораторної роботи № 2.

Розширений пошук передбачає введення щонайменше три параметра в форму пошуку, один з параметрів обов'язковий. З обов'язковим параметром побудувати умову в SELECT запиті з операцією like.

Один запит на пошук має включати всі 3 таблиці бази даних.

Результати пошуку представити в табличному форматі.

4. Додати пункт меню, що відкриває результати виконання самостійної роботи

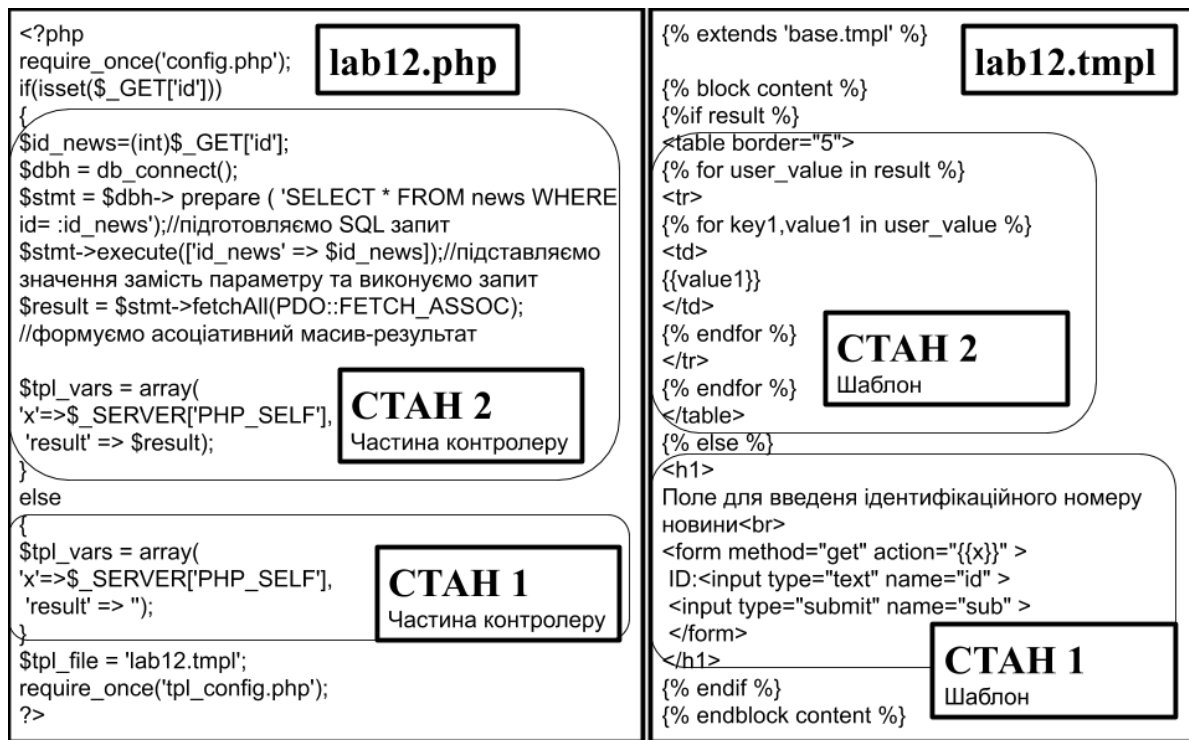


Рис. 4. Приклад рішення задачі на пошук у базі даних. Файли частини контролеру та шаблону (представлення).

Література

1. Fabien Potencier. Шаблонизаторы в PHP. URL:<https://habrahabr.ru/post/75901/> (дата звернення 23.11.2016)
2. Twig: The flexible, fast, and secure template engine for PHP. URL:<http://twig.sensiolabs.org/> (дата звернення 23.11.2016)
3. Введение в MVC для интернет-разработок. URL:<http://bourabai.kz/dbt/mvc.htm> (дата звернення 23.11.2016)
4. Web-технологии. Основы языка PHP. URL:<https://htmlweb.ru/php/php2.php> (дата звернення 22.11.2016)
5. Довідник з MySQL. URL:<http://www.mysql.ru/docs/man/SELECT.html> (дата звернення 22.05.2019)
6. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч.посібник. – Електронне видання, 2018. – 118 с. URL:<https://dspace.uzhnu.edu.ua/jspui/handle/lib/19776> (дата звернення 22.05.2016)

Лабораторна робота №5

Структурування результатів пошуку у форматі блогу

Мета: Формування практичних навичок роботи з параметричними SELECT запитам. Розроблення шаблону “Блог”.

Теоретичний матеріал

Аналіз сторінок типу «блог» на сайті, що розроблено з використанням CMS Joomla.

Табличний вигляд результатів вибору інформації з бази даних - це інтуїтивно зрозумілий формат. Переходячи до розуміння формату блога наведемо приклад з професійних сайтів. Так на сайтах, що розробляються на основі CMS Joomla формат блогу зазвичай передбачає виведення заголовку матеріалу, анонсу матеріалу та кнопки “ДЕТАЛЬНІШЕ...” (рис.1)



Рис. 1. Приклад відображення даних у форматі блогу з сайта Університету Ушинського

Сторінка типу «блог» надає список кортежей з кнопками «Детальніше». При натисненні на кнопку «Детальніше» ми переходимо до перегляду вибраної статті. Розглянемо цей момент з використанням термінології станів і отримаємо наступне:

Стану 1 - це “Блог”;

Стані 2 - відображення повного змісту обраної статті(рис. 2).

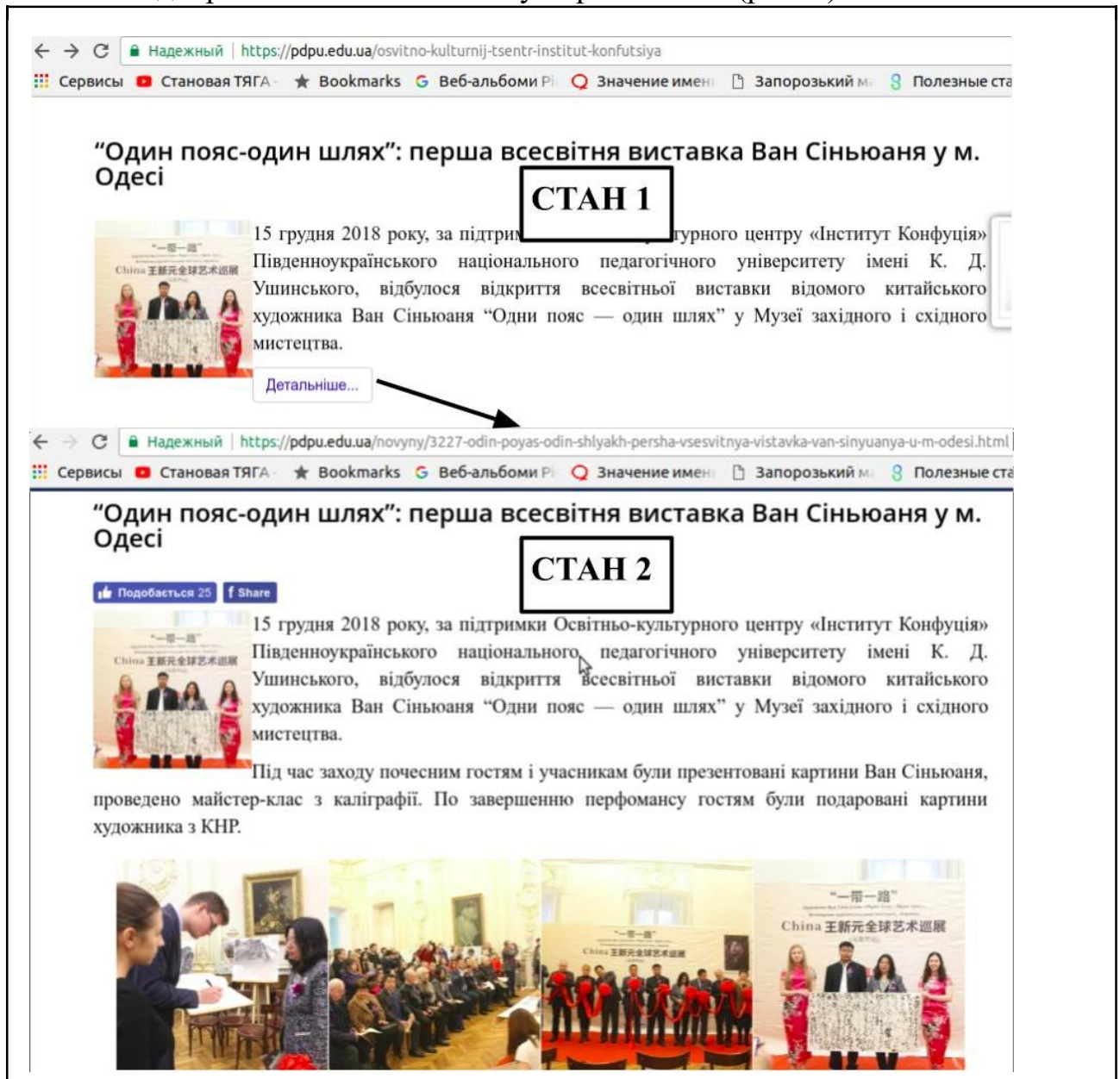


Рис. 2. Стани для результатів у вигляді Блогу (CMS Joomla). Приклад з сайту Університету Ушинського

Перехід зі Стану 1 до Стану 2 відбувається не через відправлення даних форми, а через перехід за гіперпосиланням, до якого прикріплено зовнішні змінні. В якості прикладу розглянемо, як можна передати значення змінної $pk=1$ і $x=25$. Нагадаємо, що передача даних в адресній строчці – це передача методом GET. Синтаксис передачі зовнішніх даних через URL адресу наступний:

URL/?pk=1&x=25

URL включає повний шлях до файлу

& ставиться між змінними, що передаються, у випадку однієї змінної & відсутній. Наш приклад:

<http://ukrrr.zzz.com.ua/templates/experiment/lab13.php/?pk=1>

З урахуванням перелічених фактів перейдемо до вивчення програмного коду реалізації сторінок типу “Блог” інформаційної системи.

Хід роботи:

1. Реалізувати у власній інформаційній системі приклад рішення задачі. В межах даної лабораторної роботи ми маємо можливість розглянути програмну реалізацію двох станів сторінок інформаційної системи типу “Блог” (рис. 3).

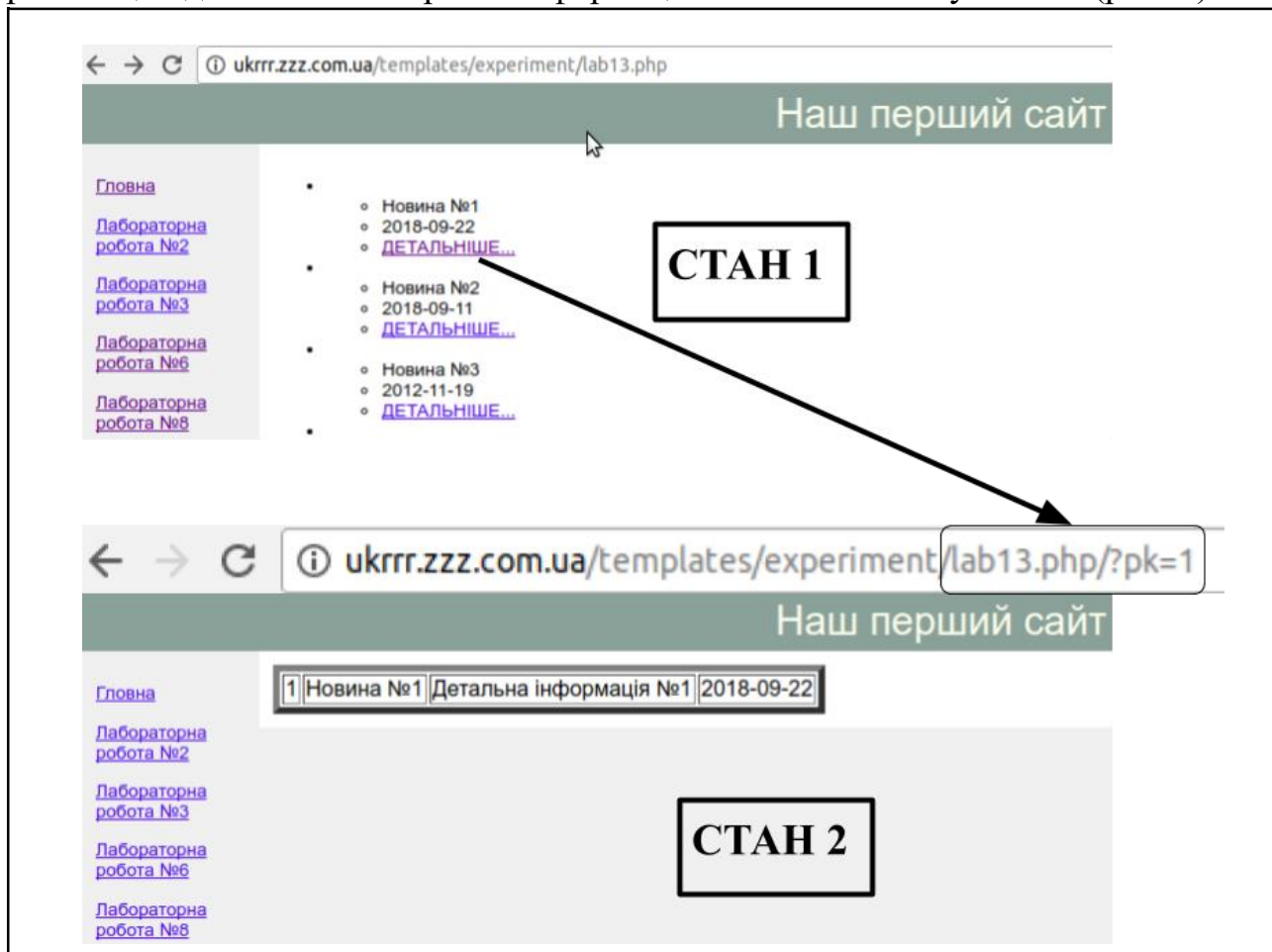


Рис. 3. Стани для відображення результатів Select запиту у форматі блога
На рисунку (рис.3) необхідно звернути увагу на ряд фактів:

У цьому прикладі перед нами постає завдання сформулювати файл частини контролеру lab13.php, що відкривається у двох станах:

Стан 1 - перегляд результатів SELECT запиту у форматі блогу з підключенням шаблону lab11.tpl;

Стан 2 - перегляд вибраного запису у табличному форматі з підключенням шаблону lab10.php (рис. 5).

<pre>{% extends 'base.tpl' %} {% block content %} {% for key,user_value in result %} {% for value1 in blog_atribut %} {{user_value[value1]}} {% endfor %} ДЕТАЛЬНІШЕ... {% endfor %} {% endblock content %}</pre> lab11.tpl СТАН 1	<pre>{% extends 'base.tpl' %} {% block content %} <table border="5"> {% for user_value in result %} <tr> {% for value1 in user_value %} <td> {{value1}} </td> {% endfor %} </tr> {% endfor %} </table> {% endblock content %}</pre> lab10.tpl СТАН 2
--	---

Рис. 5. Шаблони, що використано для опрацювання станів

В шаблон lab11.tpl (відображення результатів Стану 1) передаємо з частини контролеру масив \$tpl_vars з наступними змінними:

- масив з результатом запиту **result**
- масив з іменами атрибутів, дані по котрим будуть відображені для кожного кортежу **blog_atribut**
- ім'я атрибуту, що є однозначною ідентифікацією кожного кортежа результату вибірки **pk**

В шаблон lab10.tpl (відображення результатів Стану 2) передаємо масив з наступними змінними:

- 1 - масив з результатом запиту **result**

В даній лабораторній роботі ми отримали можливість скористатися вже існуючими шаблонами для опрацювання різних станів файлу частини контролеру підготуємо файл lab13.php (рис. 6).

3. Створити посилання в меню вашої інформаційної системи, що відкриває приклад.

4. Виконати завдання у індивідуальній базі даних інформаційної системи.

1. Допрацювати результати Лабораторну роботу №2 шляхом налаштування відображенні результатів виборки даних у форматі блогу
2. Підготувати шаблон типу “Матеріал”, який відображає повну інформацію одного кортежу бази даних.
3. Додаткова вимога 1 - представити дані не в табличному форматі (у вигляді розділених частин, в якості ознаки поділу можна поставити лінію) з

відображенням імен атрибутів для кожного розділу.

4. Додаткова вимога 2 - використати в шаблоні елементи стилізації засобами CSS (margin, padding, background, text)

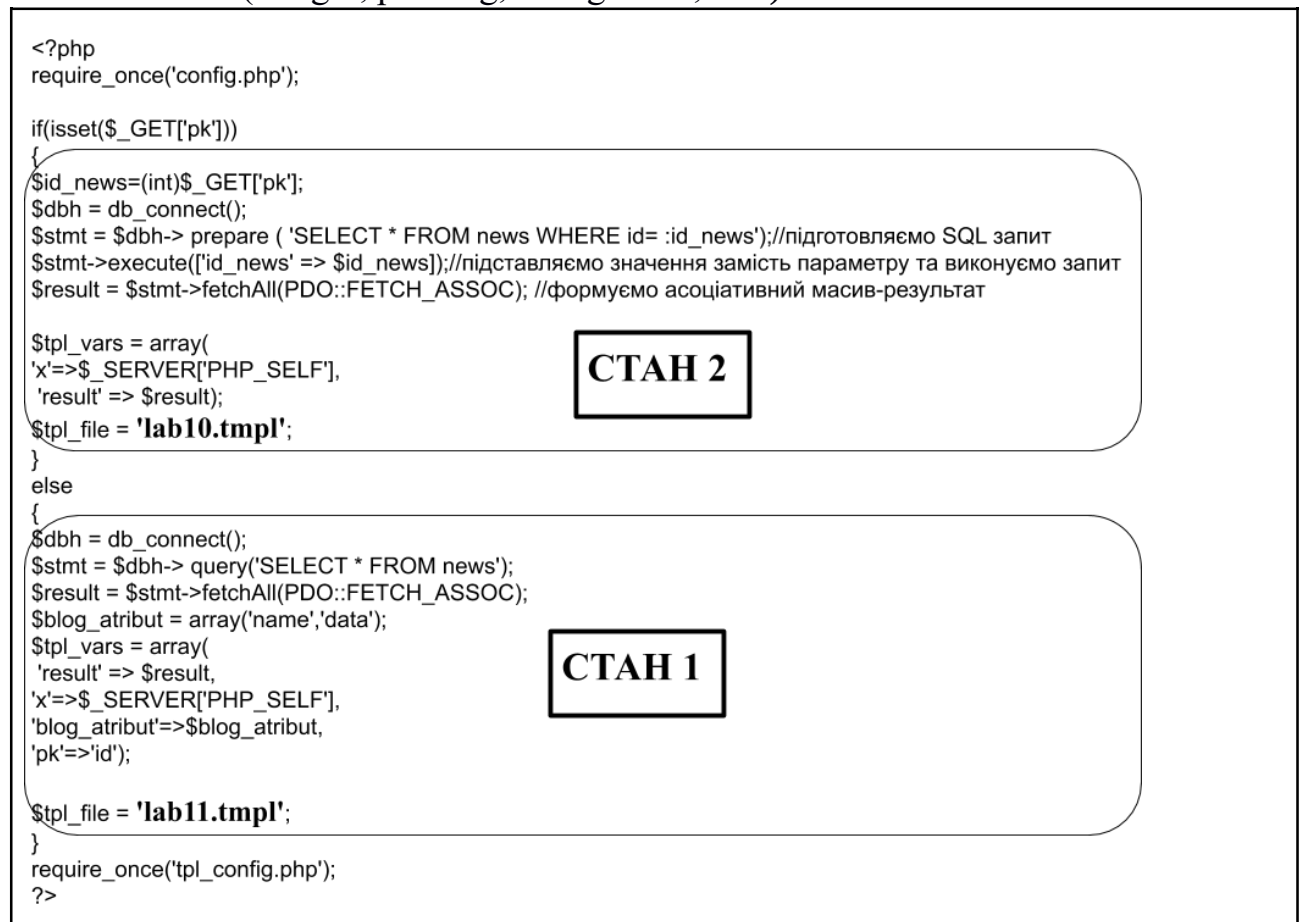


Рис. 6. Приклад реалізації станів файлу частини контролеру для формату “Блог”

Висновки

1. Ми побудували інформаційну систему, в якій розроблено шаблони для сторінок у форматі «Стаття», «Блог» та «Таблиця».
2. Зміст сторінок нашої системи залежить тільки від контенту бази даних та результатів SELECT запитів до бази даних.
3. Зміна дизайну в побудованій інформаційній системі відбувається шляхом зміни вигляду файла base.tmpl. Єдиною умовою стає збереження імен варіативних блоків.
4. Виконано чітке розмежування сторони дизайну і програмної частини. Для зміни вигляду шаблону типу «Стаття», «Блог» та «Таблиця» ми маємо один файл-шаблон, що можна формувати відповідно до вимог дизайну та використовувати для відображення будь якої кількості сторінок частини контролеру.
5. Сторінки частини контролеру запускаються у різних станах, для яких маємо можливість користуватися різними шаблонами

Література

1. Fabien Potencier. Шаблонизаторы в PHP. URL:<https://habrahabr.ru/post/75901/> (дата звернення 23.11.2016)
2. Twig: The flexible, fast, and secure template engine for PHP. URL:<http://twig.sensiolabs.org/> (дата звернення 23.11.2016)
3. Введение в MVC для интернет-разработок. URL:<http://bourabai.kz/dbt/mvc.htm> (дата звернення 23.11.2016)
4. Web-технологии. Основы языка PHP. URL:<https://htmlweb.ru/php/php2.php> (дата звернення 22.11.2016)
5. Довідник з MySQL. URL:<http://www.mysql.ru/docs/man/SELECT.html> (дата звернення 22.05.2019)
6. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. – Електронне видання, 2018. – 118 с. URL:<https://dspace.uzhnu.edu.ua/jspui/handle/lib/19776> (дата звернення 22.05.2016)

Лабораторна робота №6

Побудова інтерфейсу адміністрування Web-орієнтованої інформаційної системи

Мета: Формування практичних навичок розроблення нового шаблону адміністративного розділу. Налаштування меню адмін розділу.

Теоретичний матеріал

Проектування частини адміністрування Web-орієнтованої інформаційної системи.

Підходячи до питання проектування частини адміністрування інформаційної системи визначимо основну ціль роботи даного розділу, що полягає у маніпулюванні даними бази даних. Виходячи з даної цілі ми маємо наступні задачі:

1. В адміністративному розділі створити можливість вибору таблиці, що буде оброблятися.
2. Для вибраної таблиці створити посилання для додавання запису.
3. Для вибраної таблиці створити інтерфейс вибору записів для редагування та видалення.
4. Підготувати шаблон зовнішнього вигляду частини адміністрування.

Аналіз прикладу адміністративного розділу в CMS Joomla.

Перед початком програмної реалізації частини адміністрування переглянемо приклад організації адміністрування таблиць бази даних у CMS Joomla (рис.1). На наведеному рисунку представлена стартова панель адміністративного розділу, що позначена цифрою 1. В цій панелі прямокутником виділено два посилання в лівому меню:

- перше - це посилання “Створити нову статтю”, що переводить до форми для створення нової статті з подальшою можливістю внести цей матеріал в відповідну таблицю бази даних через запуск команди INSERT;
- друге - це посилання “Статті”, що переводить до стартової сторінки менеджету керування даними відповідної таблиці. Цей менеджер на рисунку (рис.1) позначено цифрою 2. В менеджері дані таблиці представлені в табличному форматі з можливістю вибрати будь який кортеж для редагування (“Редагувати”) або видалення (“У кошик”), відповідно перше посилання запускає форму редагування вибраного запису з подальшим виконанням команди UPDATE цього кортежу таблиці бази даних, а друге - команду DELETE (будемо так рахувати з деякою долею припущень).

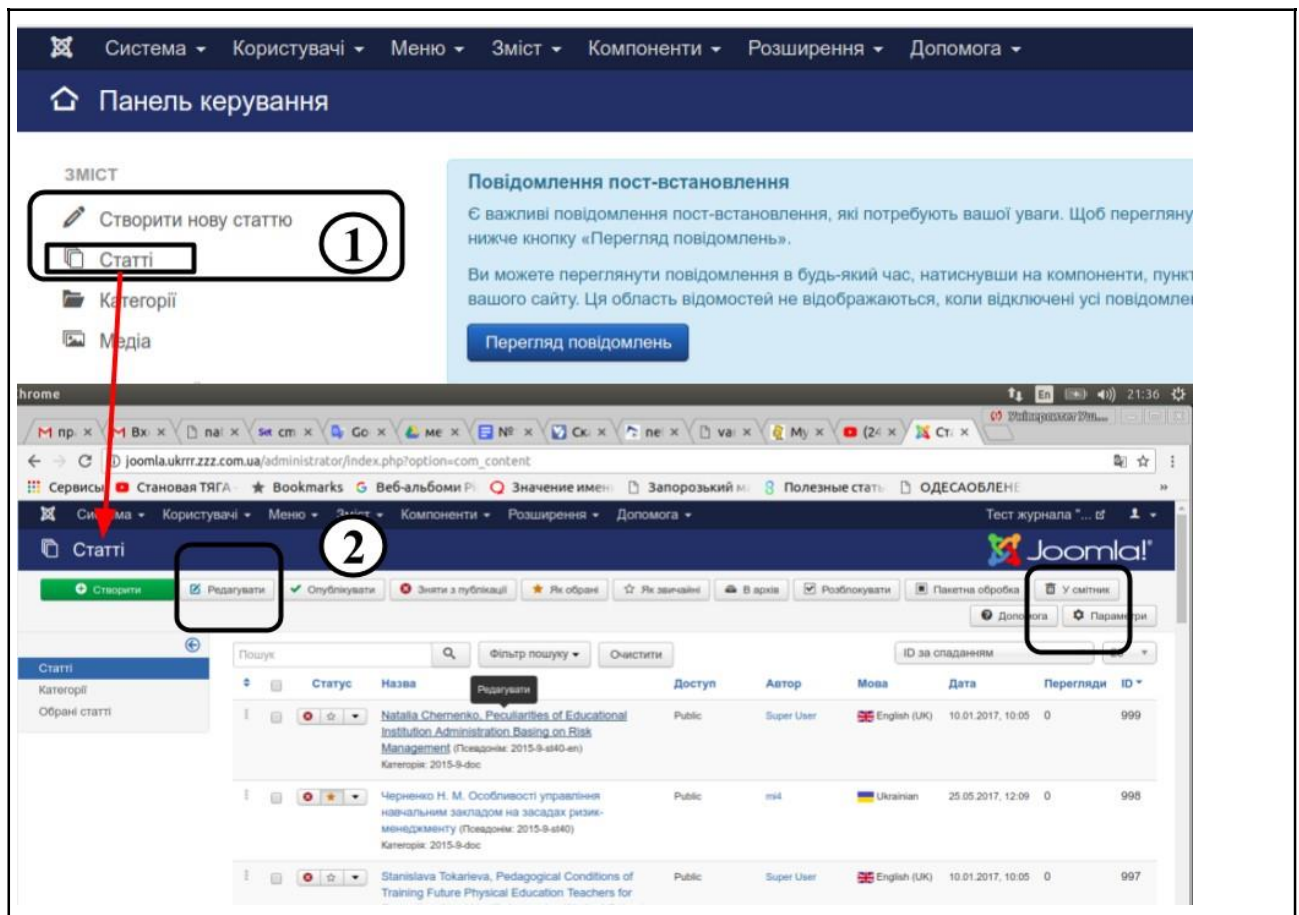


Рис. 1. Приклад організації адміністрування таблиці бази даних в CMS Joomla
Хід роботи:

1. Реалізувати у власній інформаційній системі приклад адміністрування таблиці news бази даних.

Задачею даної лабораторної роботи стає підготування інтерфейсу частини адміністрування інформаційної системи на прикладі опрацювання однієї таблиці news (Лабораторна робота №2). Згідно прикладу, що було розглянуто розіб'ємо задачу на підзадачі:

- 1 Підготувати базовий шаблон admin_base.tmpl (рис. 2) з варіативним блоком під контент сторінок адміністрування
- 2 Налаштувати в базовому шаблоні блок меню, що складається з наступних посилань (рис. 2):
 - 2.1 Адмін. панель - посилання на файл /administrator/index.php, що відкриває стартову сторінку панелі адміністрування
 - 2.2 Менеджер новин - посилання на файл /administrator/news.php, що відображає всі записи таблиці news в табличному форматі з можливістю вибору записів для редагування та видалення
 - 2.3 Додати новину - посилання на файл /administrator/add_news.php
 - 2.4 Повернутися на сайт - посилання на сторінку index.php основного сайта

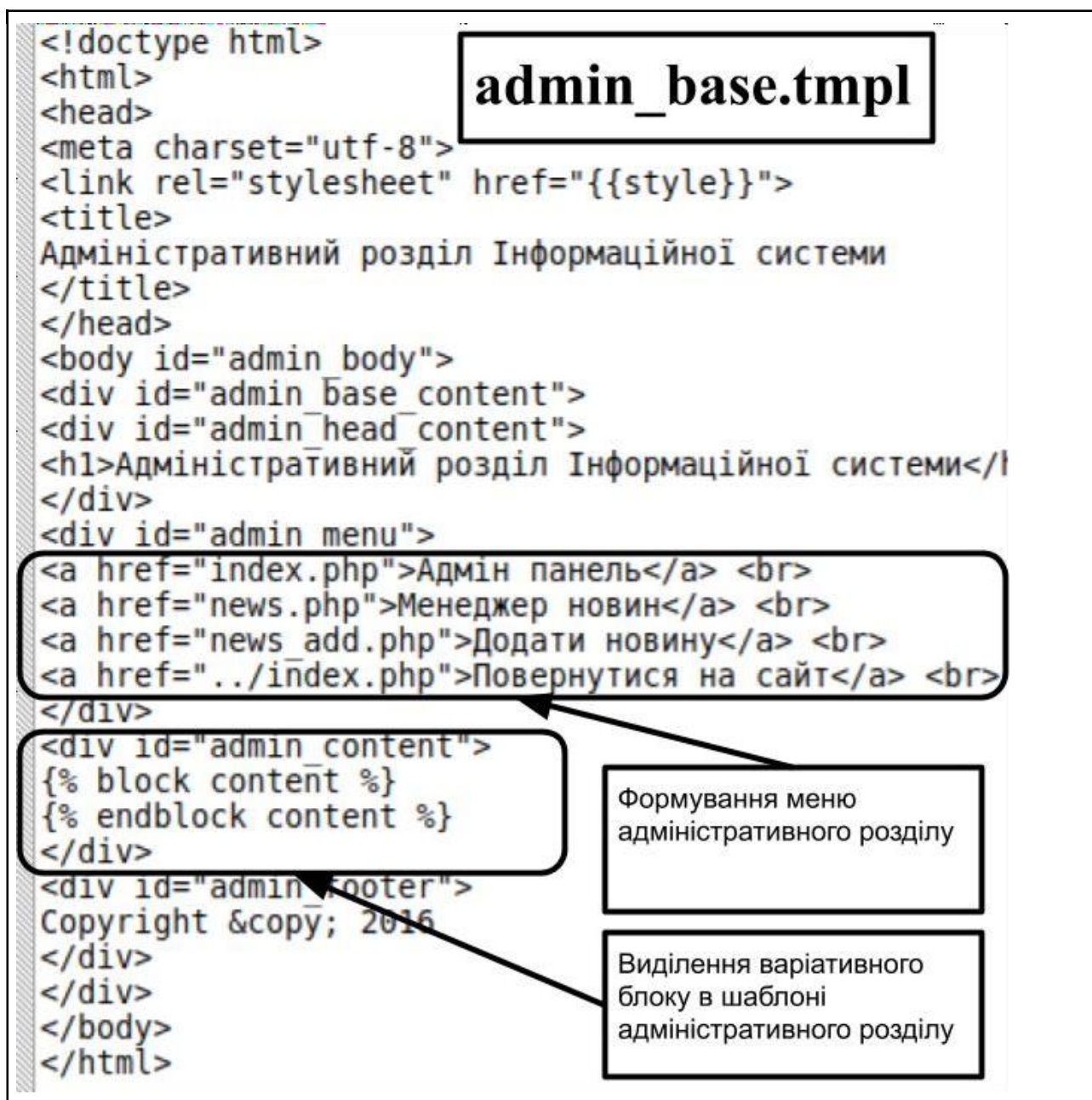


Рис. 2. Шаблон `admin_base.tpl` в папці `templates`

3 В кореневій директорії системи на хостингу створити папку `administrator`, в котрій будуть зберігатись всі файли частини контролеру адміністративного розділу (рис. 3).

4 На першому етапі створюємо стартовий файл адміністративного розділу, що відкриває базовий шаблон `administrator/index.php` (рис.3). В файлі `index.php`, що розташований в папці `administrator`, прописано шлях до конфігураційних файлів двома способами:

4.1 `../config.php` - піднімаємось в кореневу директорію і звертаємось до потрібного файла

4.2 `DIR_ROOT.`tpl_config.php`` - в цьому варіанті нами використано константу, що переводить в кореневий каталог інформаційної системи. Нагадаємо те, що ця константа визначається в файлі `config.php`.

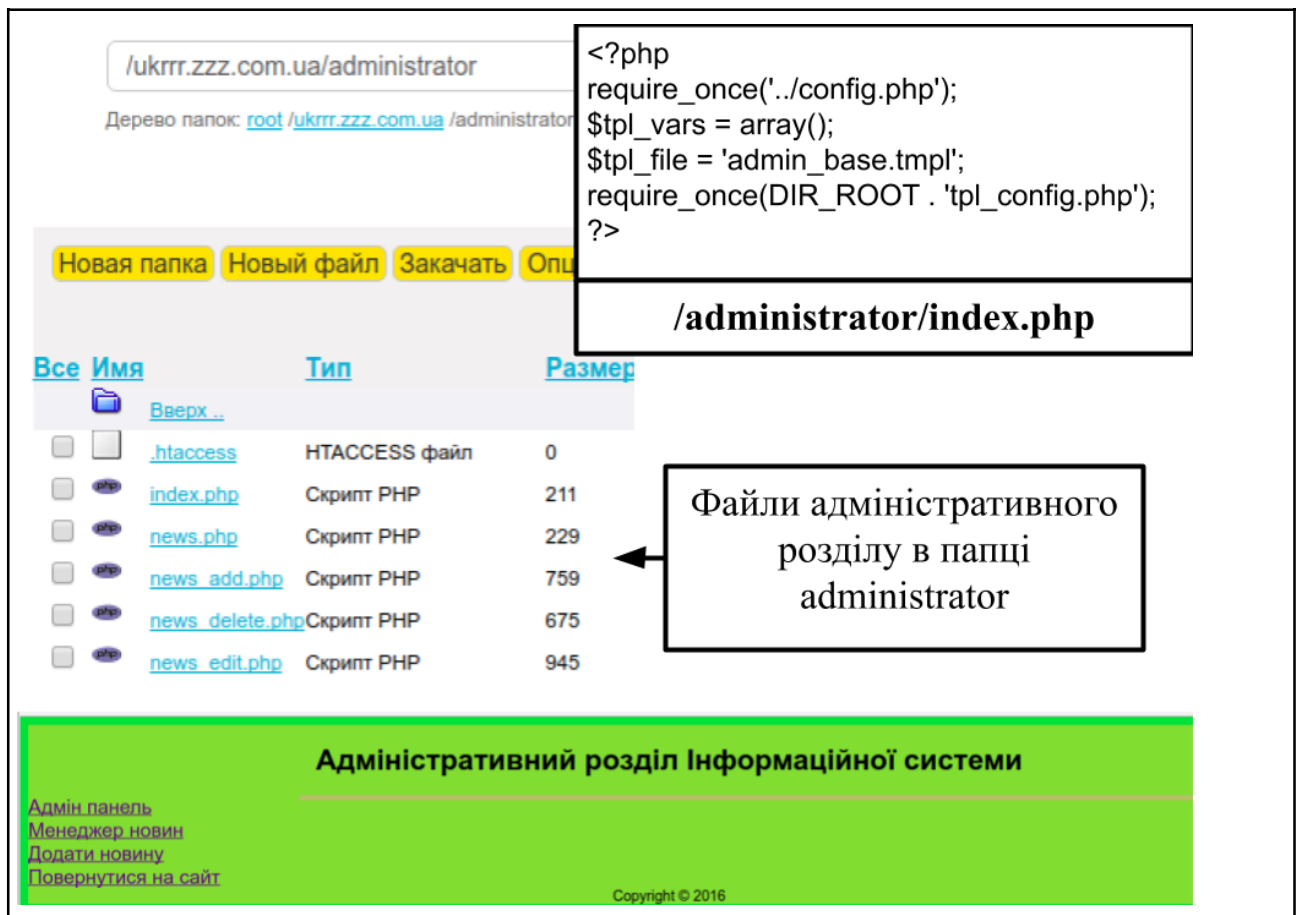


Рис. 3. Структурування Адмін панелі. Файли адміністративного розділу системи

5 Другий етап - це підготовка менеджера таблиці news. Менеджер таблиці будується шляхом вибірки всіх даних з таблиці news і представлення в табличному вигляді, що було розглянуто в Лабораторній роботі №2.

5.1 Файл частини контролеру аналогічний прикладу в лабораторній роботі №2 з одним нюансом, ми маємо передати в шаблон ім'я первинного ключа нашої таблиці. Це нам необхідно для подальшого налагодження роботи посилань для редагування та видалення вибраного запису, ім'я файлу news.php (рис.4) . В файлі news.php ми формуємо масив даних, що доступний в шаблоні, який складається з:

- result - результат вибірки всіх записів з таблиці, що представлено у вигляді асоціативного масиву
- pk - ім'я поля первинного ключа таблиці бази даних

5.2 Файл-шаблон admin_news.tpl (рис.4) представляє результати вибірки даних з таблиці news у табличному форматі з додатковими посиланнями для редагування та видалення кожного запису. Розглянемо більш детально код посилань "edit" та "delete":

```
<a href="news_edit.php?t={{user_value[pk]}}">edit</a>
<a href="news_delete.php?t={{user_value[pk]}}">delete</a>
```


Для запису з кодом 25 буде передано змінну t зі значенням 25:
news_edit.php?t=25

```
<?php
require_once('../config.php');
$dbh = db_connect();
$stmt = $dbh->query("SELECT * FROM news");
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);
$tpl_vars = array(
    'result' => $result,
    'pk'=>'id');
$tpl_file = 'admin_news.tmpl';
require_once(DIR_ROOT . 'tpl_config.php');
?>
```

```
{% extends 'admin_base.tmpl' %}

{% block content %}
<table border="5">
{% for user_value in result %}
<tr>
{% for value1 in user_value %}
<td>
{{value1}}
</td>
{% endfor %}
<td>
<a href="news_edit.php?t={{user_value[pk]}}">edit</a>
</td>
<td>
<a href="news_delete.php?t={{user_value[pk]}}">delete</a>
</td>
</tr>
{% endfor %}
</table>
{% endblock content %}
```

administrator/news.php

Адміністративний розділ Інформаційної

[Адмін панель](#)
[Менеджер новин](#)
[Додати новину](#)
[Повернутися на сайт](#)

1	Новина №1	Детальна інформація №1	2018-09-22	edit	delete
2	Новина №2	Детальна інформація №2	2018-09-11	edit	delete
3	Новина №3	Детальна інформація №3	2012-11-19	edit	delete
4	Новина №4	Детальна інформація №4	2012-11-13	edit	delete
5	Новина №5	Детальна інформація №5	2012-11-01	edit	delete

Copyright © 2016

Рис. 4. Менеджер таблиці news (Менеджер новин)

2 Завдання для реалізації в індивідуальній інформаційній системі.

Налаштувати адміністративний розділ для трьох таблиць бази даних, що було розроблено в межах індивідуального завдання Лабораторної роботи №2.

1. Підготувати шаблон адміністративного розділу з налаштованим меню для переходу до менеджерів кожної з трьох таблиць та до сторінок для додавання записів до кожної з трьох таблиць
2. Створити пусті файли частини контролеру, що будуть в подальшому розроблено в папці administrator
3. Налаштувати файл index.php частини адміністрування
4. Для кожної з таблиць бази даних налаштувати менеджер для редагування та видалення записів
5. Реалізувати побудову менеджера всіх трьох таблиць одним файлом частини контролеру і одним шаблоном.

Література

1. Fabien Potencier. Шаблонизаторы в PHP. URL:<https://habrahabr.ru/post/75901/> (дата звернення 23.11.2016)
2. Twig: The flexible, fast, and secure template engine for PHP. URL:<http://twig.sensiolabs.org/> (дата звернення 23.11.2016)
3. Введение в MVC для интернет-разработок. URL:<http://bourabai.kz/dbt/mvc.htm> (дата звернення 23.11.2016)
4. Web-технологии. Основы языка PHP. URL:<https://htmlweb.ru/php/php2.php> (дата звернення 22.11.2016)
5. Довідник з MySQL. URL:<http://www.mysql.ru/docs/man/SELECT.html> (дата звернення 22.05.2019)
6. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. – Електронне видання, 2018. – 118 с. URL:<https://dspace.uzhnu.edu.ua/jspui/handle/lib/19776> (дата звернення 22.05.2016)

Лабораторна робота №7

Адміністративний розділ. Команда INSERT

Мета: Формування практичних навичок розроблення менеджера додавання даних інформаційної системи.

Теоретичний матеріал.

Вивчення прикладу організації додавання даних в таблицю зі статтями в CMS Joomla

Розглянемо приклад організації роботи посилання “Створити нову статтю” 1- Панелі керування (рис. 1) в CMS Joomla. При переході за цим посиланням нам відкривається менеджер “Статті: нова стаття”, що пронумеровано цифрою 2 на рисунку (рис. 1). Цей менеджер - це форма для додавання нового запису в таблицю бази даних. Розглянемо кнопки, що визначають стани роботи даного менеджера:

“Зберегти” - при першому зверненні виконує команду INSERT і залишає відкритим запис, що було додано в відповідну таблицю в режимі редагування. При другому і подальших зверненнях ця кнопка запускає команду UPDATE для поточного запису таблиці;

“Зберегти і закрити” - виконує команду INSERT та переводить систему у менеджер для вибору записів для редагування та видалення;

“Зберегти і створити” - виконує команду INSERT і відкриває пусту форму для додавання нового запису;

“Скасувати” - закриває менеджер додавання записів і скасовує всі зміни, що було зроблено в полях форми. Система переводиться в режим менеджера редагування.

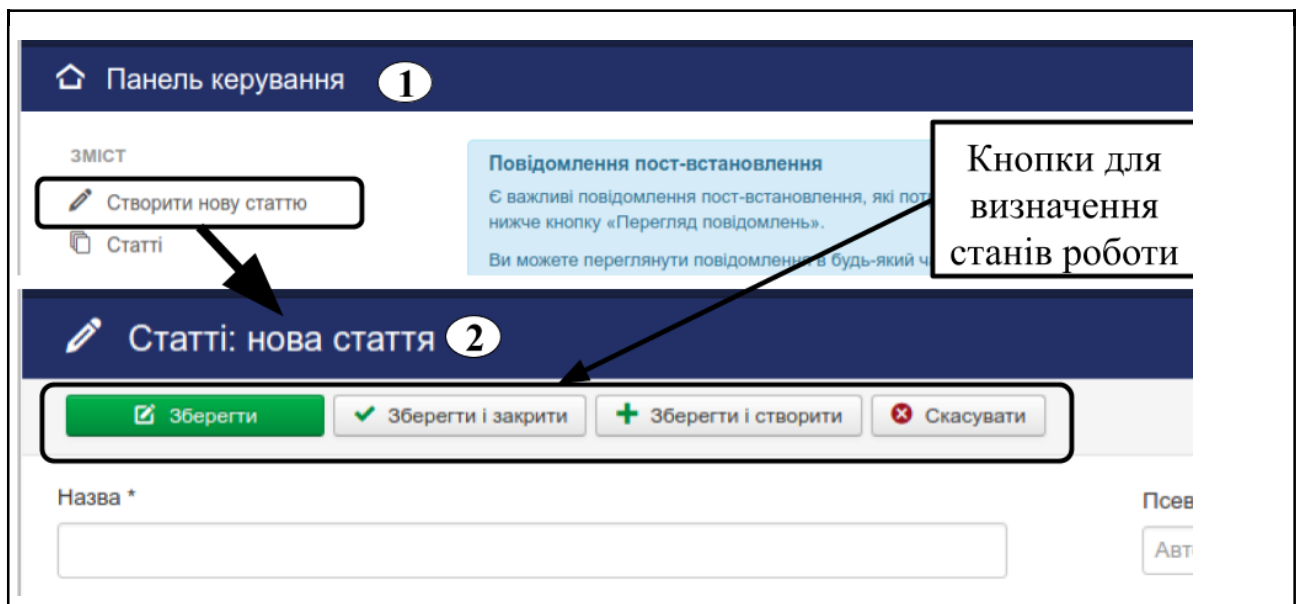


Рис. 1. Організація роботи по створенню нового запису таблиці бази даних
Хід роботи:

1. Реалізувати у індивідуальній інформаційній системі приклад формування менеджера додавання запису для таблиці news

Згідно з попередньою лабораторною роботою, нам необхідно в частині контролеру налаштувати роботу файлу /administrator/add_news.php

Для даного файлу опрацюємо чотири стани:

1-Перше відкриття файлу;

2-Стан після натиснення кнопки “Зберегти і закрити”;

3-Стан після натиснення кнопки “Зберегти і створити”;

4-Стан після натиснення кнопки “Скасувати”.

Проведемо аналіз кожного стану:

1- Цей стан відкриває пусту форму

2 - У другому стані виконується команда INSERT, що додає запис та відкривається менеджер редагування записів таблиці

3 - У третьому стані виконується команда INSERT, що додає запис та відкривається менеджер додавання в першому стані

4 - У четвертому стані без додаткових дій відкривається менеджер редагування записів таблиці

Критерієм настануплення кожного з станів є натиснення кнопки SUBMIT з відповідним значенням імені.

Для підвантаження потрібного файлу частини контролеру скористаємось командою: require_once('news.php') (рис. 2).

Підключення файлу-шаблону news_add.tpl відображає пусту форму для додавання даних таблиці news (рис. 4).

Зовні в системі форма виглядає, як звичайна форма, в котрій є три кнопки для опрацювання різних станів (рис. 3)

2. Виконати завдання для самостійної роботи.

Налаштувати Менеджер додавання записів у кожному з таблиць

2.1 Розробити для всіх таблиць бази даних шаблони форми-додавання даних.

2.2 Розробити файли частини контролеру, що опрацьовують 4 стани роботи Менеджера додавання даних для кожної з таблиць індивідуальної БД (рис.2)

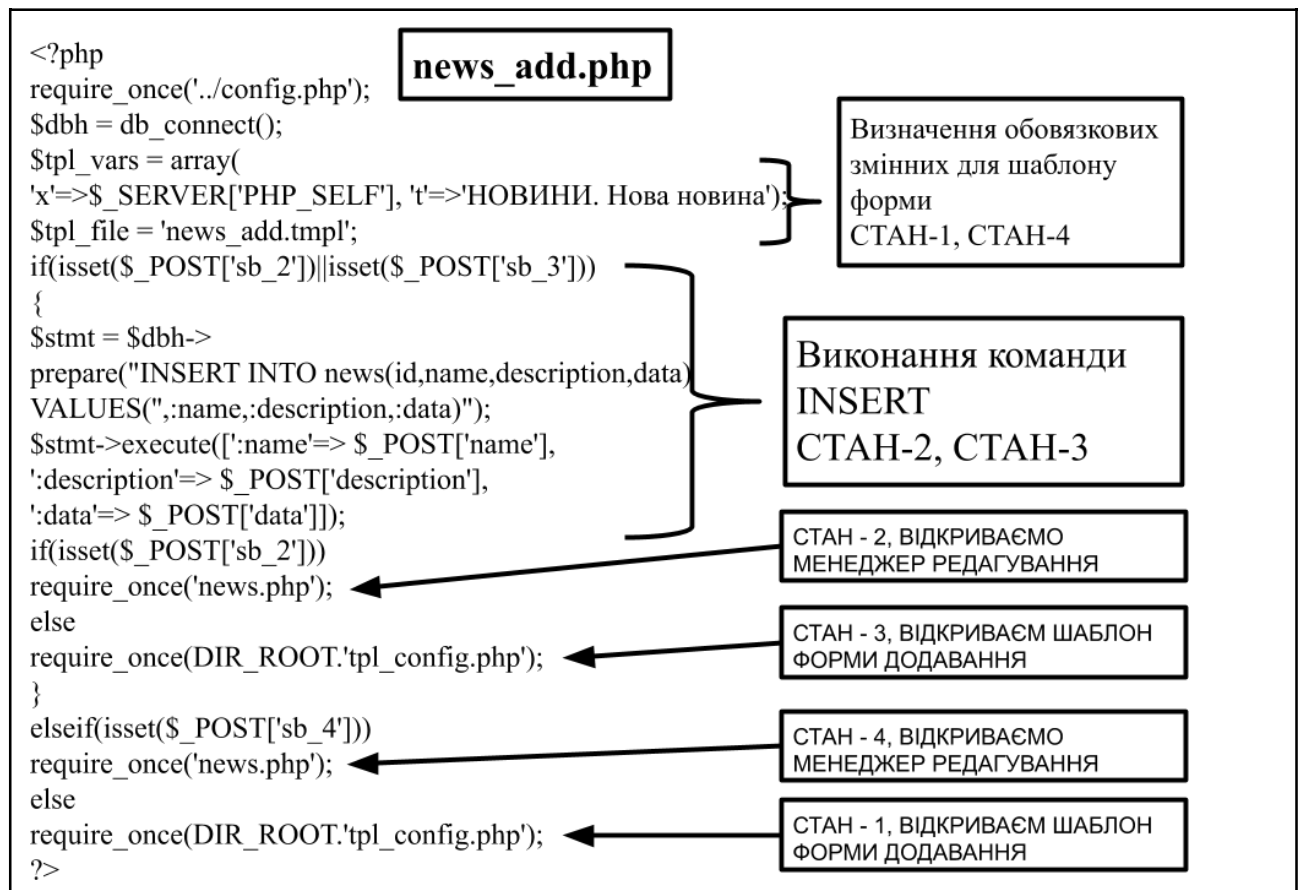


Рис. 2. Файл частини контролеру менеджера створення нової новини

Адміністративний розділ Інформації

[Адмін панель](#)
[Менеджер новин](#)
[Додати новину](#)
[Повернутися на сайт](#)

НОВИНИ. Нова новина

Назва новини:	
Детальна інформація:	
Дата публікації новини (уууу-мм-дд):	
Зберегти і закрити	Зберегти і створити
Скасувати	

Copyright © 2016

Рис. 3. Зовнішній вигляд прикладу Менеджера додавання новин

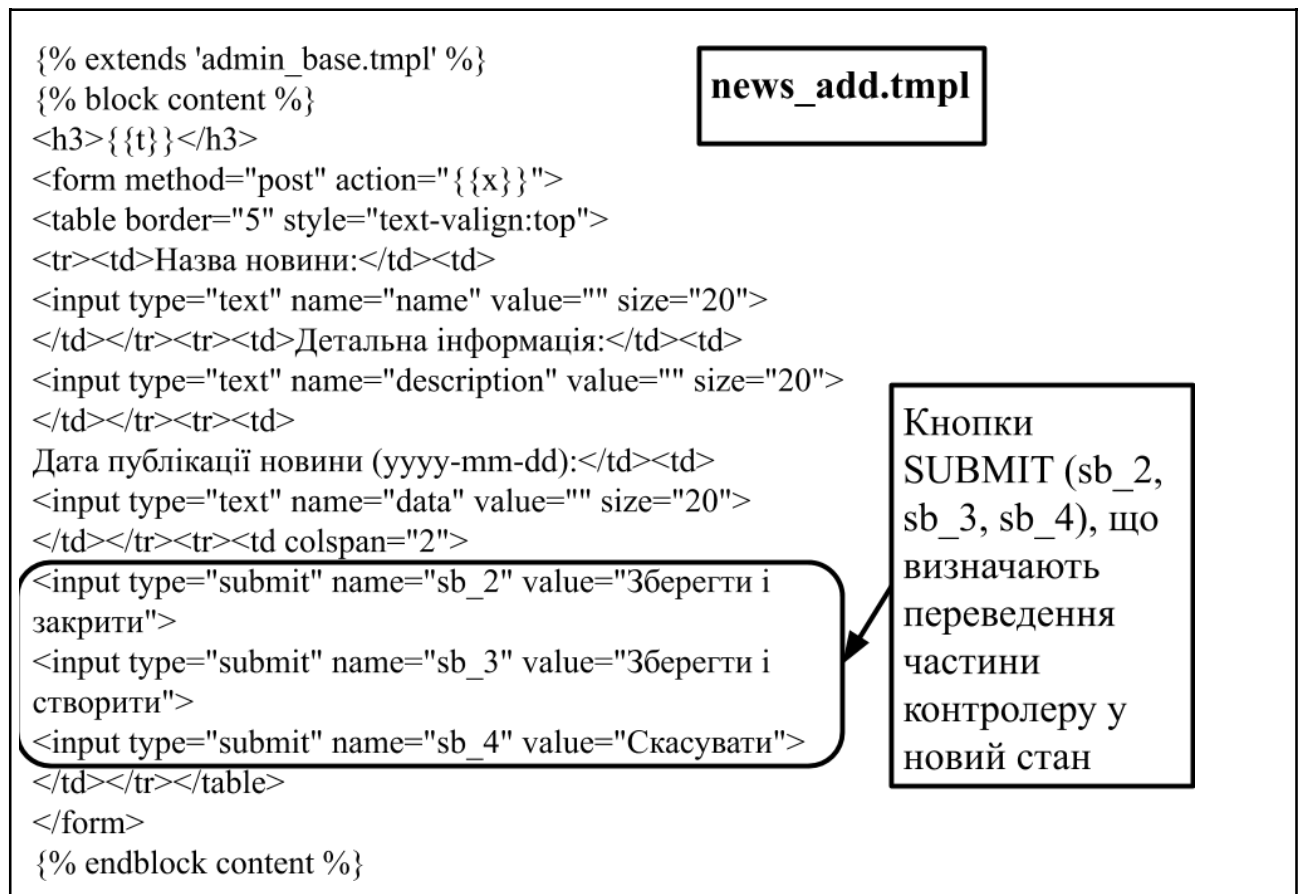


Рис. 4. Стани частини контролеру. Шаблон побудови форми додавання записів

Література

1. Fabien Potencier. Шаблонизаторы в PHP. URL: <https://habrahabr.ru/post/75901/> (дата звернення 23.11.2016)
2. Twig: The flexible, fast, and secure template engine for PHP. URL: <http://twig.sensiolabs.org/> (дата звернення 23.11.2016)
3. Введение в MVC для интернет-разработок. URL: <http://bourabai.kz/dbt/mvc.htm> (дата звернення 23.11.2016)
4. Web-технологии. Основы языка PHP. URL: <https://htmlweb.ru/php/php2.php> (дата звернення 22.11.2016)
5. Довідник з MySQL. URL: <http://www.mysql.ru/docs/man/SELECT.html> (дата звернення 22.05.2019)
6. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. – Електронне видання, 2018. – 118 с. URL: <https://dspace.uzhnu.edu.ua/jspui/handle/lib/19776> (дата звернення 22.05.2016)

Лабораторна робота №8

Адміністративний розділ. Команда UPDATE.

Мета: Формування практичних навичок розроблення менеджера редагування даних інформаційної системи.

Теоретичний матеріал

Вивчення прикладу організації редагування даних в CMS Joomla

Розглянемо приклад організації роботи менеджера редагування запису таблиці бази даних в CMS Joomla (рис. 1).

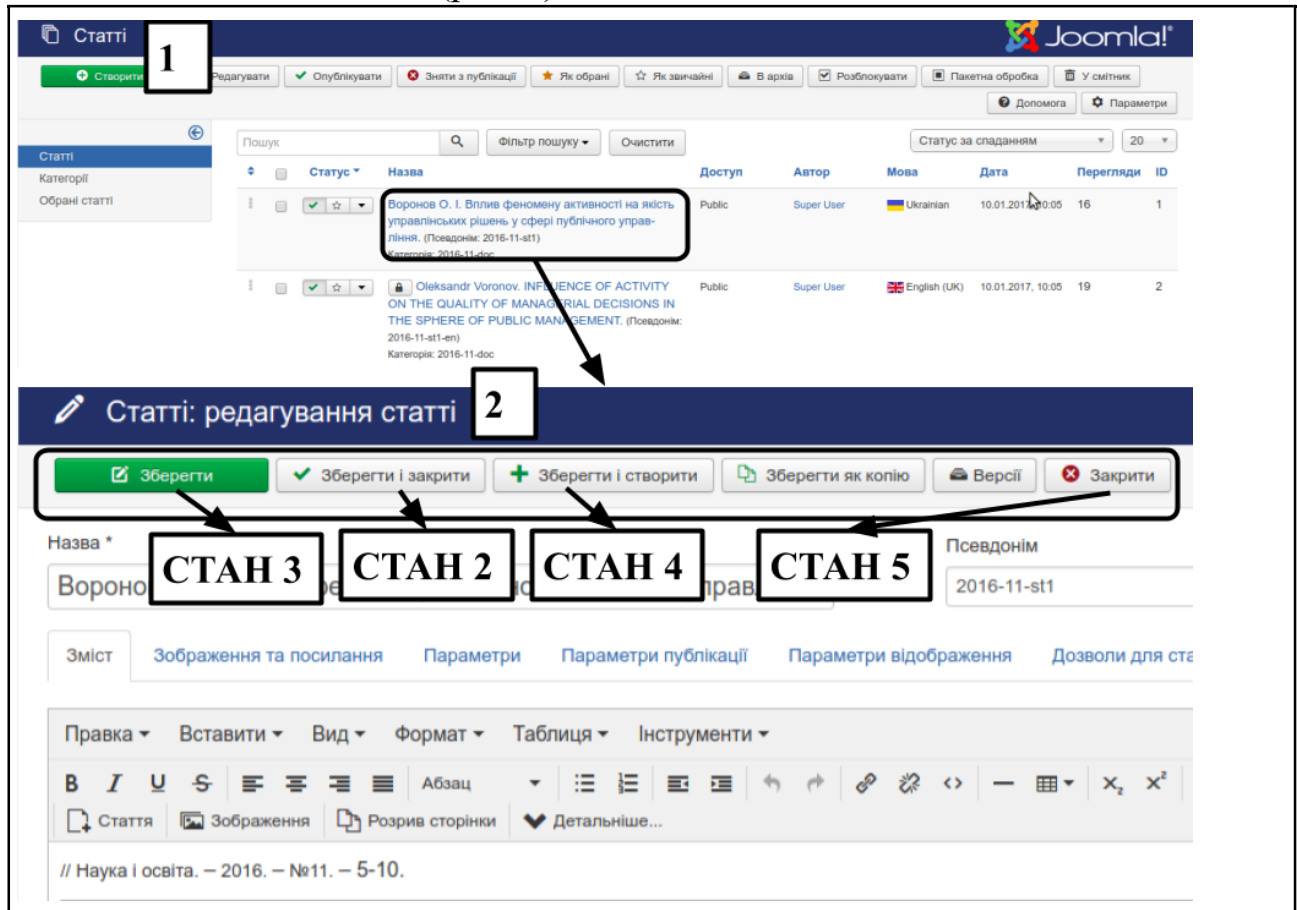


Рис. 1. Менеджер редагування запису в CMS Joomla

Згідно з рисунком в панелі адміністрування обраної таблиці, що позначено цифрою 1, ми натискаємо на назву необхідного запису і відкриваємо Менеджер редагування для даного запису, що позначено цифрою 2, (рис.1). Менеджер редагування відкриває форму, поля котрої заповнено значеннями відповідних атрибутів обраного запису. Процес редагування - це процес налаштування значень атрибутів через відповідні поля форми з подальшим виконанням команди UPDATE. Менеджер редагування працює в різних станах розглянемо деякі з них:

Стан1 - це відкриття обраного запису в режимі редагування

Стан2, Стан3, Стан4 - це виконання команди UPDATE з подальшим відкриттям:

Стан2 (“Зберегти і закрити”) - Менеджеру редагування таблиці;
Стан3 (“Зберегти”) - Менеджеру редагування цього запису;
Стан4 (“Зберегти і створити”) - Менеджеру додавання нового запису таблиці;
Стан5 (“Закрити”) - ігнорування змін, що внесено в форму і відкриття Менеджеру редагування таблиці.

Хід роботи:

1. Реалізувати в індивідуальній інформаційній системі приклад формування файлів Менеджеру редагування таблиці news (файл news_edit.php)

Для побудови файлу частини контролеру Менеджеру редагування обраного запису edit.php на першому етапі визначимось з зовнішніми змінними, що є критеріями відкриття даного файлу в різних станах:

Стан2 - визначино кнопку Submit на ім'я sb_st2

Стан3 - визначино кнопку Submit на ім'я sb_st3

Стан4 - визначино кнопку Submit на ім'я sb_st4

Стан5 - визначино кнопку Submit на ім'я sb_st5

Стан1 - не визначино жодної з кнопок Submit, але передано в файл методом GET значення змінної t (значення первинного ключа вибраного запису)

Стан0 - це спроба відкрити файл news_edit.php без передачі жодної з перелічених зовнішніх змінних. У цьому стані файл зразу перенаправляє систему у Менеджер редагування відповідної таблиці (рис. 2).

В файлі частини контролеру news_edit.php визначається функція fedit(\$dbh, \$id). Ця функція для переданого в якості фактичного параметру об'єкту, що ідентифікує з'єднання з базою даних \$dbh та значення первинного ключа запису, що потребує редагування формує:

1. Результат вибірку запису з таблиці бази даних \$result;
2. Ідентифікує масив змінних tpl_vars в функції fedit на рисунку (рис. 2), що передаються в шаблон (result - результат вибірки запису, x-шлях до поточного файлу, pk - ім'я атрибуту первинного ключа);
3. Ідентифікує ім'я шаблону (news_edit.tpl);
4. Підключає конфігураційний файл tpl_config.php для відображення Менеджеру редагування обраного запису.

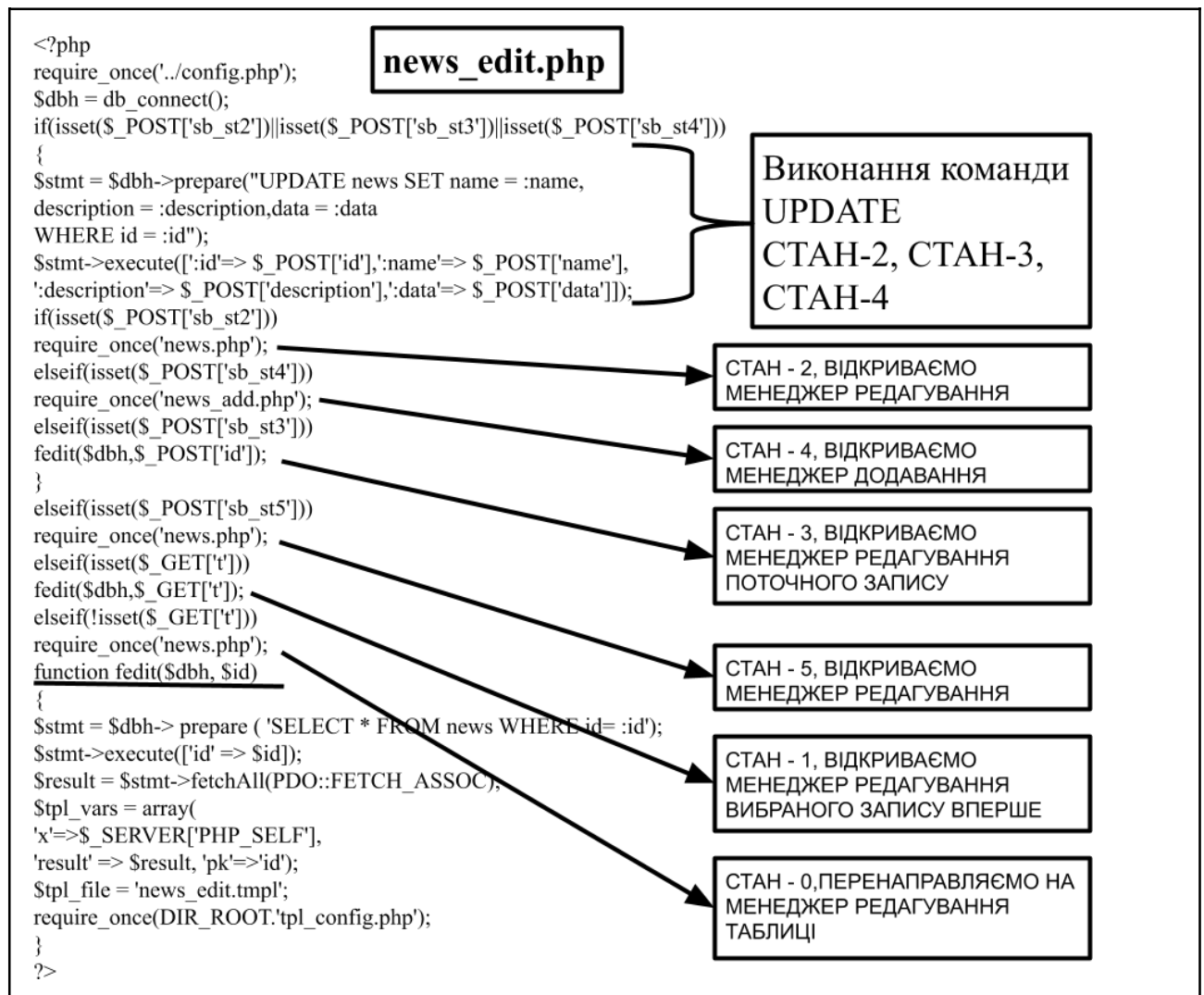


Рис. 2. Файл частину контролеру news_edit.php для формування Менеджеру редагування вибраного в Менеджері таблиці запису

2. Реалізувати в індивідуальній інформаційній системі приклад формування файлів Менеджеру редагування таблиці news (файл news_edit.tpl).

Наступним кроком є знайомство з правилами побудови файлу сторони представлення news_edit.tpl. На основі результату вибірки обраного запису таблиці (result) та імені первинного ключа (pk) нам необхідно побудувати форму для редагування запису з кнопками SUBMIT, що описують перелічені вище стани роботи. Прямий найпростіший спосіб реалізації - це ручна побудова форми, в котрій явно прописано назви змінних, що співпадають з назвами атрибутів (Рис. 3). Для цікавих пропонується завдання на створення універсального шаблону Менеджеру підтвердження видалення обраного запису.

При побудові форми ми створюємо поле hidden для значення первинного ключа запису.

Значення решти полів форми відображено в вигляді однострокових полів введення даних - text

Загальний вигляд Менеджеру редагування запису представлено на рисунку (рис.4)

```
{% extends 'admin_base.tpl' %}
{% block content %}
<h3>{{t}}</h3>
<form method="post" action="{{x}}">
<table border="5" style="text-align: top">
{%for user_value in result%}
<tr><td>
Назва новини:
</td><td>
<input type="hidden" name="id" value="{{user_value[pk]}}">
<input type="text" name="name" value="{{user_value.name}}" size="20">
</td></tr><tr><td>
Детальна інформація:
</td><td>
<input type="text" name="description" value="{{user_value.description}}"
size="20">
</td></tr><tr><td>
Дата публікації новини (yyyy-mm-dd):
</td><td>
<input type="text" name="data" value="{{user_value.data}}" size="20">
</td></tr>
{%endfor%}
<tr><td colspan="2">
<input type="submit" name="sb_st3" value="Зберегти">
<input type="submit" name="sb_st2" value="Зберегти і закрити">
<input type="submit" name="sb_st4" value="Зберегти і створити">
<input type="submit" name="sb_st5" value="Закрити">
</td></tr>
</table>
</form>
{% endblock content %}
```

news_edit.tpl

Рис. 3. Шаблон news_edit.tpl Менеджеру редагування запису

3. Виконати завдання для самостійної роботи.

Налаштувати Менеджер редагування вибраного запису в кожній з таблиць бази даних

1. Розробити для всіх таблиць бази даних шаблони форми-редагування даних (рис. 3)
2. Розробити файл частини контролеру, що опрацьовує 6 станів роботи Менеджера редагування вибраного запису для кожної з таблиць індивідуальної БД (рис.2)
3. Проаналізувати, чи можна створити універсальний файл частини контролеру, що буде правильно працювати з будь-якою таблицею бази даних

4. Проаналізувати, чи можна створити шаблон форми, що буде правильно налаштовувати форму для будь-якої таблиці бази даних (Для цікавих пропонується завдання на створення універсального шаблону Менеджеру редагування обраного запису)

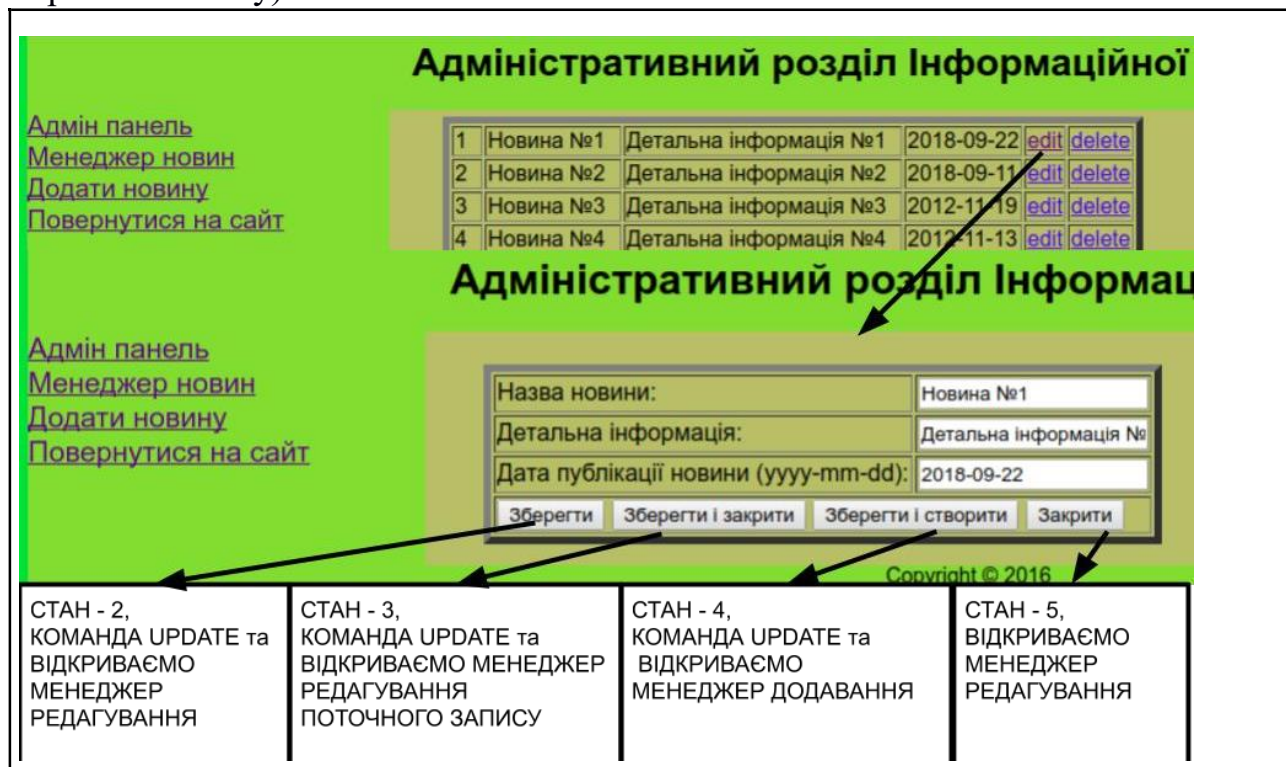


Рис. 4. Менеджер редагування запису таблиці news

Література

1. Fabien Potencier. Шаблонизаторы в PHP. URL:<https://habrahabr.ru/post/75901/> (дата звернення 23.11.2016)
2. Twig: The flexible, fast, and secure template engine for PHP. URL:<http://twig.sensiolabs.org/> (дата звернення 23.11.2016)
3. Введение в MVC для интернет-разработок. URL:<http://bourabai.kz/dbt/mvc.htm> (дата звернення 23.11.2016)
4. Web-технологии. Основы языка PHP. URL:<https://htmlweb.ru/php/php2.php> (дата звернення 22.11.2016)
5. Довідник з MySQL. URL:<http://www.mysql.ru/docs/man/SELECT.html> (дата звернення 22.05.2019)
6. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч. посібник. – Електронне видання, 2018. – 118 с. URL:<https://dspace.uzhnu.edu.ua/jspui/handle/lib/19776> (дата звернення 22.05.2016)

Лабораторна робота №9

Адміністративний розділ. Команда DELETE.

Мета: Формування практичних навичок розроблення менеджера видалення обраного запису таблиці інформаційної системи.

Теоретичний матеріал

Приклад реалізації видалення записів таблиці в CMS Joomla

Розглянемо приклад роботи менеджера редагування запису таблиці бази даних в CMS Joomla, в котрому виконуються дії по повному видаленню вибраного запису з таблиці бази даних (рис. 1).

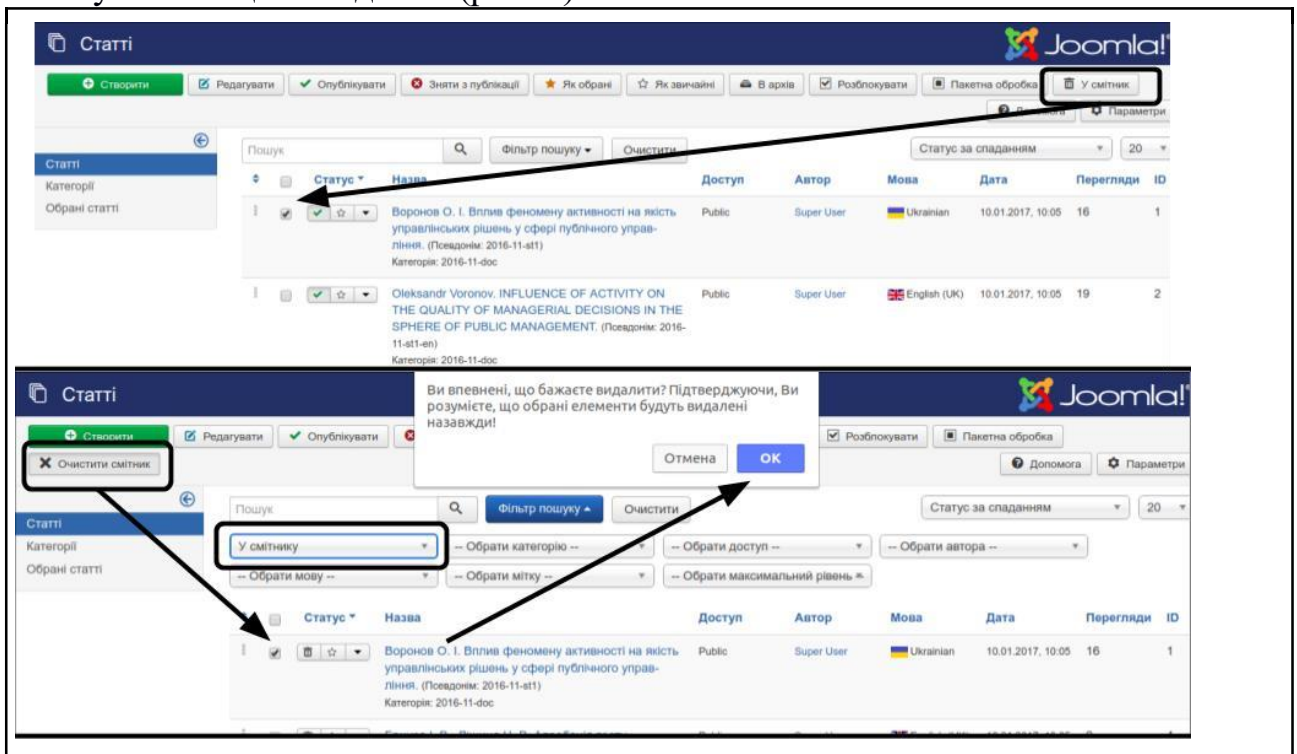


Рис. 1. Етапи видалення статті, що реалізуються в CMS Joomla
Видалення запису - це процес, що розбивається на два етапи.

- На першому для вибраного запису змінюється статус на “У смітнику”. Прийнято говорити: “Запис розташовано в смітнику”, або у “Кошику”. Насправді відбувається перший етап по видаленню запису, на якому запис потрапляє у список потенційно готових до видалення. Схематично перелічені дії представлено в верхній частині рисунку (Рис. 1).
- На другому етапі після вибору одного чи декількох записів, натискання кнопки “Очистити смітник” та підтвердження дій по видаленню вибраних записів виконується команда DELETE, що остаточно видалить запис з бази даних. Схематично перелічені дії представлено в нижній частині рисунку (Рис.1).
- Така багатоетапність процедури видалення запису практично

унеможливлнює видалення даних з таблиць бази даних через необачність.

Хід роботи:

1. Реалізувати у індивідуальній інформаційній системі приклад формування файлу частини контролеру news_delete.php для таблиці news

Наведений приклад роботи CMS Joomla продемонстрував багатоетапність процедури видалення. Для спрощення реалізації дій по видаленню запису таблиці news дії по видаленню виконаємо в два етапи (рис. 2):

- Перший етап видалення - це відкриття вибраного запису для перегляду
- Другий етап видалення - це виконання команди DELETE після натискання кнопки, що підтверджує видалення запису

Тепер спробуємо формалізувати стани файлу частини контролера Менеджеру видалення вибраного запису **news_delete.php (рис. 3):**

Стан0 - файл news_delete.php відкрито без передавання зовнішніх змінних. Повертаємось в Менеджер редагування таблиці news.php (рис. 3, Стан 0)

Стан1 - файл news_delete.php відкрито вперше з переданим методом GET значенням первинного ключа запису, що буде видалено (рис. 3, Стан 1). Відкриваємо запис в режимі перегляду з доступними кнопками переходу до інших станів. Вигляд даного стану в Інформаційній системі наведено на рисунку (рис. 2)

Стан2 - натиснуто кнопку “Підтвердити видалення” (sb_d2), виконується команда DELETE і завантажується Менеджер редагування таблиці news.php (рис. 3, Стан 2)

Стан3 - натиснуто кнопку “Відмінити” (sb_d3), завантажується Менеджер редагування таблиці news.php (рис. 3, Стан 3)

В файлі частини контролеру news_delete.php визначається функція f(\$dbh, \$id).

Ця функція для переданого в якості фактичного параметру об’єкту, що ідентифікує з’єднання з базою даних \$dbh та значення первинного ключа запису, що потребує редагування формує:

- 1- Результат вибірку запису з таблиці бази даних \$result;
- 2- Ідентифікує масив змінних tpl_vars в функції f(...) (рис. 2), що передаються в шаблон (result - результат вибірки запису, x-шлях до поточного файлу, pk - ім’я атрибуту первинного ключа);
- 3- ідентифікує ім’я шаблону (news_delete.tmpl)
- 4- підключає конфігураційний файл tpl_config.php для відображення

Менеджеру підтвердження видалення обраного запису.

В якості прикладу роботи Менеджера таблиці в режимі видалення запису та Менеджеру підтвердження видалення обраного запису розглянемо рисунок (рис. 2)



Рис. 2. Робота Менеджера видалення запису

2. Реалізувати у індивідуальній інформаційній системі приклад формування файлу частини дизайну news_delete.tmpl для таблиці news

Наступним кроком є знайомство з правилами побудови файлу сторони представлення news_delete.tmpl. На основі результату вибірки обраного запису таблиці (result) та імені первинного ключа (pk) нам необхідно побудувати відображення запису в табличному форматі та створити форму з hidden полем, що містить значення первинного ключа та з кнопками SUBMIT, що описують перелічені вище стани роботи. Прямий найпростіший спосіб реалізації - це ручна побудова таблиці та форми, в котрій явно прописано назви змінних, що співпадають з назвами атрибутів (Рис. 4). Для цікавих пропонується завдання на створення універсального шаблону Менеджеру підтвердження видалення обраного запису.

При побудові форми ми створюємо поле hidden для значення первинного ключа запису. Загальний вигляд файлу news_delete.tmpl представлено на рисунку (рис. 4).

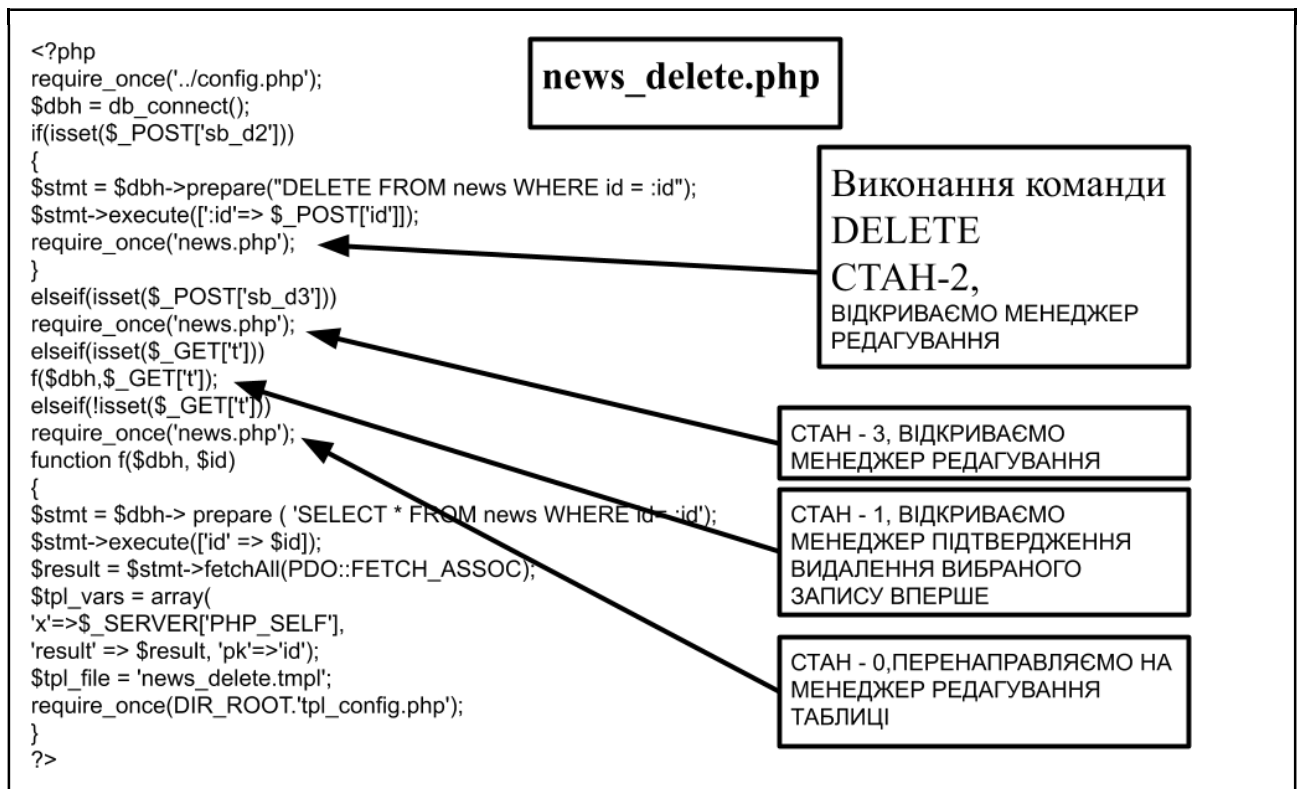


Рис. 3. Частина контролеру Менеджеру підтвердження видалення запису

3. Виконати завдання для самостійної роботи.

Налаштувати Менеджер видалення вибраного запису в кожній з таблиць індивідуальної бази даних

1. Розробити для всіх таблиць бази даних шаблони Менеджерів підтвердження видалення запису (рис. 4)
2. Розробити файл частини контролеру, що опрацьовує 4 стани роботи Менеджеру підтвердження видалення вибраного запису для кожної з таблиць індивідуальної БД (рис.2)
3. Проаналізувати, чи можна створити універсальний файл частини контролеру, що буде правильно працювати з будь-якою таблицею бази даних
4. Проаналізувати, чи можна створити універсальний шаблон форми та таблиці, що буде правильно налаштовувати Менеджер підтвердження видалення вибраного запису довільної таблиці бази даних

```

{% extends 'admin_base.tpl' %}
{% block content %}
<h3>{{t}}</h3>
<form method="post" action="{{x}}">
<table border="5" style="text-align:top">
{%for user_value in result%}
<tr><td>
Код новини:
</td><td>
{{user_value[pk]}}
<input type="hidden" name="id" value="{{user_value[pk]}}">
</td></tr>
<tr><td>
Назва новини:
</td><td>
{{user_value.name}}
</td></tr><tr><td>
Детальна інформація:
</td><td>
{{user_value.description}}
</td></tr><tr><td>
Дата публікації новини (yyyy-mm-dd):
</td><td>
{{user_value.data}}
</td></tr>
{%endfor%}
<tr><td colspan="2">
<input type="submit" name="sb_d2" value="Підтвердити видалення">
<input type="submit" name="sb_d3" value="Відмінити">
</td></tr>
</table>
</form>
{% endblock content %}

```

news_delete.tpl

Рис. 4. Частина дизайну (View) Менеджеру підтвердження видалення запису

Література

1. Fabien Potencier. Шаблонизаторы в PHP. URL:<https://habrahabr.ru/post/75901/> (дата звернення 23.11.2016)
2. Twig: The flexible, fast, and secure template engine for PHP. URL:<http://twig.sensiolabs.org/> (дата звернення 23.11.2016)
3. Введение в MVC для интернет-разработок. URL:<http://bourabai.kz/dbt/mvc.htm> (дата звернення 23.11.2016)
4. Web-технологии. Основы языка PHP. URL:<https://htmlweb.ru/php/php2.php> (дата звернення 22.11.2016)