WEB-ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

А.О. Лісняк, С.В. Чопоров

Зміст

Лабораторна робота №1. Налагодження роботи веб-сервера.

Мета: отримати практичні навички налагодження роботи веб-сервера на доступній програмній платформі.

Теоретичні відомості

Всі сайти відкриваються браузерами як HTML-документи. Більшість сучасних сайтів мають динамічні елементи, тобто фрагменти контенту, які змінюються в часі та безпосередньо залежать від запитів відвідувачів. Для того, щоб HTML-сторінки стала динамічними (могли залежати від поведінки людини або зовнішніх подій) веб-серверу необхідно мати спеціалізоване програмне забезпечення для виконання інтерпретованих скриптів або скомпільованих програм. Схема такої взаємодії представлена на рис.1

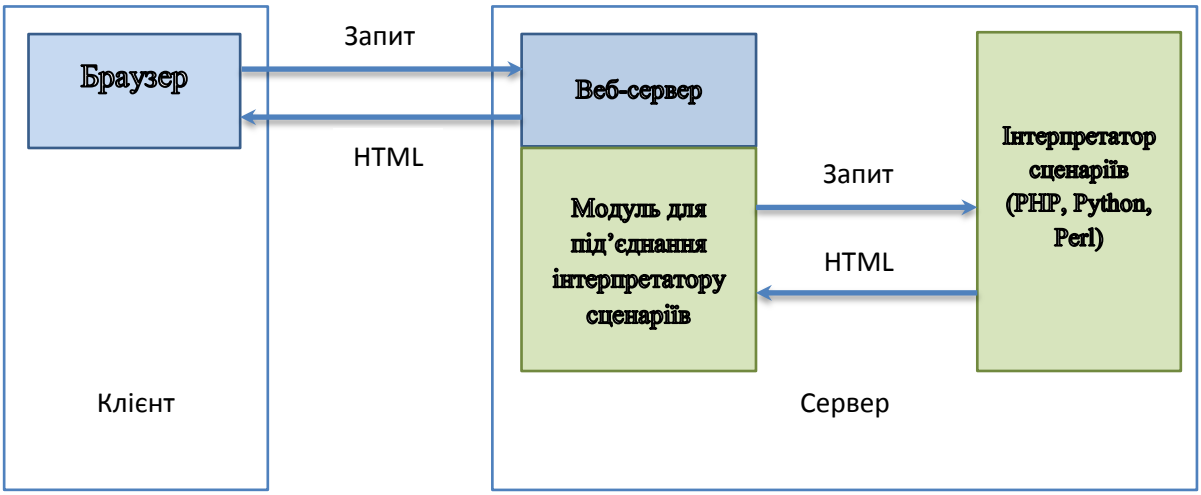


Рисунок 1. – Схема обробки запиту веб-сервером.

Для роботи серверу використовують перелік сучасних Веб-серверів: Apache, Nginx, Microsoft-IIS, LiteSpeed, Google Servers, Tomcat та інші. Найбільш розповсюдженими з яких є Apache та Nginx [1].

До найбільш поширених мов програмування для роботи на стороні серверу відносять: PHP, ASP.NET, Java, ColdFusion, Perl, Ruby, Python, JavaScript та інші. Серед них PHP займає 80% [2].

Завдання

1. Обрати одну з версій операційної системи Linux (Ubuntu, Suse, CentOS, Mint) та встановити її у якості віртуальної машини.

2. Використовуючи команди операційної системи налагодити роботу веб-серверу Apache, інтерпретатору сценаріїв PHP та СУБД MySql останніх версій.

3. У кореневій директорії веб-сервера створити php-скрип (index.php), що виводить інформацією наступного змісту: ПІБ, номер групи, назва дисципліни у лівій колонці та інформацію про поточну версію інтерпретатора php-скриптів і програмному забезпеченні встановленого на сервері (функції phpinfo).

4. Встановити не пустий пароль доступу до облікового запису «root» СУБД MySql, та створити нового користувача «user» для доступу клієнтів.

5. Встановити веб-панель керування СУБД MySql – PhpMyAdmin та протестувати можливості доступу до неї користувачів «root» та «user».

6. Створити віртуальний хост з назвою «name.labs», де name – прізвище англійською мовою та розмістити на ньому HTML-сторінку з короткою інформацією про себе.

7. Протестувати доступ до localhosh та name.labs на локальному та зовнішньому[1] хостах.

8. Написати звіт до якого включити команди ОС та дії які виконувались при налагодженні роботи веб-сервера.

Контрольні питання

1. Що таке веб-сервер?
2. Чи може веб-сервер Apache виконувати сценарії PHP?
3. Що таке віртуальний хост і навіщо він потрібен?
4. Який номер порту має веб-сервер за замовченням?

[1] На віддалено хосту необхідно внести інформацію про домен name.labs до файлу hosts

Лабораторна робота №2. Основи мови програмування PHP.

Мета: отримати практичні навички використання типів даних та основних керуючих конструкцій мови програмування PHP.

Теоретичні відомості

PHP – это широко используемый язык сценариев общего назначения с открытым исходным кодом. Аббревиатура PHP означает «Hypertext Preprocessor» (Препроцессор Гипертекста). Важным преимуществом языка PHP перед такими языками, как языков Perl и C заключается в возможности создания HTML документов с внедренными командами PHP.

PHP нечто средние между компилятором и интерпретаторам и обрабатывает сценарии по следующему принципу:

- на вход PHP подается сценарий;
- он переводится (транслируется в специальный байт-код (внутреннее представление);
- выполняется байт-код (а не код самой программы!), не создавая при этом исполняемого файла.

Байт-код значительно компактнее обыкновенного кода программы, поэтому он интерпретируется (выполняется) быстрее.

Переменные PHP определяются по следующим правилам:

- имя переменной должно начинаться со знака доллара $;
- имя переменной не должно содержать никаких других символов, кроме символов латинского алфавита, цифр и знака подчеркивания;
- имена переменных в PHP чувствительны к рецистру;
- объявлять переменную можно в любом месте программы, но до места первого использования.

Переменные в PHP могут содержать данные любого типа. Исключение составляют константы, которые могут содержать только число или строку. Выбор типа осуществляется самим интерпретатором. Однако иногда PHP может ошибиться и сопоставить переменной неправильный тип. В таких случаях нужно задавать тип явно.

Таблица 2.1 – Типы переменных в PHP

Тип переменной	Описание
Integer	Целое число. Диапазон значений: от -2 147 483 648 до -2 147 483 647 (32 бита).

double (или float)	Вещественное число очень большой
String	Строка.
Array	Массив.
Object	Объект.
Bool	Логический тип данных. Переменные этого типа могут принимать только одно из двух значений: true (истина) или false

Язык PHP предоставляет много средств для определения типа переменных:

– *is_int($x)* или is_integer ($x) – возвращает true, если переданная переменная — целое число;

– *is_double($x)* или is_float ($x) – возвращает true, если переданная переменная – вещественное число;

– *is_string ($x)* – возвращает true, если переданная переменная строка;

– *is_array ($x)* – возвращает true, если переданная переменная массив;

– *is_object ($x)* – возвращает true, если переменная является объектом;

– *is_bool ($x)* – возвращает true, если переменная объявлена, как логическая;

– *gettype($x)* – возвращает строку, описывающую тип переменной (integer, double, string, object, array, bool или unknown type, если невозможно определить тип).

Если PHP неправильно определил тип переменной, можно указать его явно. Для этого используется функция *settype ($x, $type),* где *$type* — это одна из строк, возвращаемых функцией *gettype ().*

Основные операторы языка PHP и их приоритет представлены в таблице 2.2.

Таблица 2.2 – Приоритет операций.

Приоритет	Оператор
13	(постфикс)++ (постфикс) —
12	++(префикс) --(префикс)
11	*/%
10	+ -
9	« »
8	«= »=
7	== !=
6	&
5	«
4	!
3	&&
2	II
1	= += .= *=/=%= »=«==&="= | =

Основные управляющие конструкции языка PHP приоритет представлены в таблице 2.3.

Таблица 2.3 – Операторы.

Название	Оператор
Условный оператор	*if (условие) { ... }* *if (условие) { ... } else { ... }*
Цикл с предусловием	*while (условие) { ... }*
Цикл с постусловием	*do {...} while (условие)*
Цикл со счетчиком	for*($i;$i<10;$i++){...}* *foreach($items as $val){...}* *foreach($items as $key=>$val){...}*
Оператор switch	*switch ($n) {* *case "1": {...}* *break;* *...* *default: {...}* *}*

Проверка существования переменной — очень удобная возможность языка PHP. Благодаря ей можно проверить, передан ли сценарию определенный параметр. Для проверки используют функцию *isset ($name).*

Чтобы не занимать память можно удалить ненужные нам переменные. Это делается с помощью функции *unset($name).*

Вывод данных и значений переменных осуществляется с использованием оператора *echo* и функции *print.*

Пример. Вывести таблицу умножения числа 2.

```
<?php
$n = 2;
echo "Таблица умножения числа $n <br>";
for($i = 0; $i<=9; $i++){
     echo  $n." x ".$i." = ".$n*$i."<br>";
}
?>
```

Завдання

1. Використовуючи середовище розробки веб-додатків[2] створити проект PHP у який додати:

- папку для зберігання css файлів та файл styles.css;

[2] NetBeans IDE, Exclipse IDE або іншу.

– скрипт index.php з шаблоном документу у форматы HTML5.

2. Використовуючи мову програмування PHP у тілі документа організувати відображення таблиці множення для чисел від 1 до 9.

3. Кожен запис таблиці множення розташувати у окремій комірці таблиці та мати класову помітку про парний та не парний номер комірки; кожен наступний рядок таблиці відокремити пустим рядком (рис. 2.1).

4. Описати у файлі styles.css стилі для придання таблиці необхідного вигляду та під’єднати його до скрипта index.php.

5. Розмістити файли проекту у папці «Lab1» кореневої директорій веб-серверу.

6. Протестувати результати роботи та оформити звіт.

Таблиця множення			
1	1 x 0 = 0	2 x 0 = 0	3 x 0 = 0
2	1 x 1 = 1	…	…
3	…		
4			
5			
6			
7			
8			
9			
10			
11	…	…	…

Контрольні питання

1. Які правила визначення змінних у мові програмування PHP?

2. Яким чином можливо примусово встановити тип даних змінної прогрими?

3. Які особливості оголошення та використання масивів?

4. Який масив називають асоціативним?

5. Чи можливо знищувати змінні та перевіряти їх існування?

Лабораторна робота №3. Передача та обробка параметрів форми.

Мета: навчитися створювати HTML-форми, передаватися параметри методами GET та POST, обробляти отримані параметри на стороні сервера.

Теоретичні відомості

В класичному програмуванні, зазвичай, програма отримує від користувача набор вхідних даних або параметрів в залежності від яких алгоритм програми формує певний результат. Робота сценаріїв на стороні веб-сервера теж доволі часто залежить від параметрів, що передає користувач. Правила передачі вхідних параметрів та отримання результату к даному випадку регламентовані протоколом HTTP.

Обмін повідомленнями між клієнтом та сервером відбувається за схемою «запит-відповідь» при цьому передається певна кількість службових даних (наприклад кодування веб-сторінки й/або запитати версію сторінки в потрібних мові/кодуванні і таке інше.) та, при необхідності, дані користувача.

Відправка параметрів відбувається двома різними методами.

Метод GET – передача параметрів виконуються безпосередньо у рядку URI (наприклад: http://www.example.net/resource?param1=value1&param2= value2). Згідно зі стандартом HTTP, запити типу GET вважаються *ідемпотентними* — багатократне повторення одного і того ж запиту GET повинне приводити до однакових результатів (за умови, що сам ресурс не змінився за час між запитами). Це дозволяє кешувати відповіді на запити GET.

Метод POST – передає призначені для користувача дані включеними в тіло запиту. На відміну від методу GET, метод POST не відображається у рядку запиту браузера та не вважається ідемпотентним, тобто багатократне повторення одних і тих же запитів POST може повертати різні результати.

Веб-форма (форма) — елемент веб-сторінки, який дає користувачам можливість вводити інформацію і відправляти її на сервер для подальшої обробки.

Форма задається за допомогою тегу form, усередині якого розташовуються елементи керування. Крім загальних для HTML атрибутів , в form можуть бути присутніми наступні:

– *action* – дія, обов'язковий атрибут , що містить URI обробника форми;

– *method* – метод відправки форми, атрибут, який приймає значення GET (за замовчуванням) або POST;

– *enctype* – тип кодування для вмісту;

– *accept* – список MIME - типів для завантаження файлів;

– *name* – ім'я форми;

– *onsubmit* – обробник події «форма відправлена»;

– *onreset* – обробник події «форма очищена»;
– *accept - charset* список підтримуваних наборів символів.

У HTML визначені наступні елементи керування:
– *input* наступних типів:
- *submit*
- *image*
- *reset,*
- *button*
- *checkbox*
- *radio*
- *password*
- *hidden*
- *file*

– *button;*
– *textarea;*
– *select* з елементами *optgroup* і *option* всередині;
– *label.*

У HTML5 визначено деякі додаткові елементи управляння але не всі вони є кросбраузерними.

У мові програмування PHP не треба піклуватися про отримання, розкодування і будь-яку іншу обробку даних прийнятих від користувача. PHP автоматично заповнює ряд вбудованих масивів, які ще називають суперглобальними. Їх імена та призначення наступні:

$_SERVER – серверні змінні;
$*_ENV* – змінні середовища, в якій працює PHP;
$*_COOKIE* – змінні, що передаються за допомогою cookies;
$*_GET* – параметри, що передані за допомогою методу GET;
$*_POST* – параметри, що переданих методом POST;
$*_FILES* – дані про файли, що передані за допомогою методу POST;
$*_REQUEST* – містить усередині себе масиви $*_GET*, $*_POST* та $*_COOKIE*;
$*_SESSION* – зберігає параметрів сесії.

Таким чином для обробки даних переданих користувачем зазвичай використовують масиви $*_GET*, $*_POST* та $*_REQUEST*.

Для виконання перегляду значень змінних будь яких типів зручно використовувати допоміжні функції PHP *print_r()* та *var_dump().*

Наприклад, якщо маємо запит «http://name.labs/test.php?a=10&b=hello», то для перегляду прийнятих параметрів у скрипті test.php необхідно написати:

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

```
print_r($_GET);
echo “</pre>”;
?>
```

Завдання

1. Розробити скрипт *form.php*, що відображає HTML-форму профілю користувача, що задовольняє наступним вимогам:
   - форма повинна містити всі основні елементи керування;
   - обов'язково повинне буті питання, що дає можливість вибрати один варіант відповіді з декількох запропонованих;
   - дані на сервер повинні буті передані одним з методів – GET або POST.
2. Оформити макет форми (рис. 3.1) використовуючи CSS framework – Bootstrap.

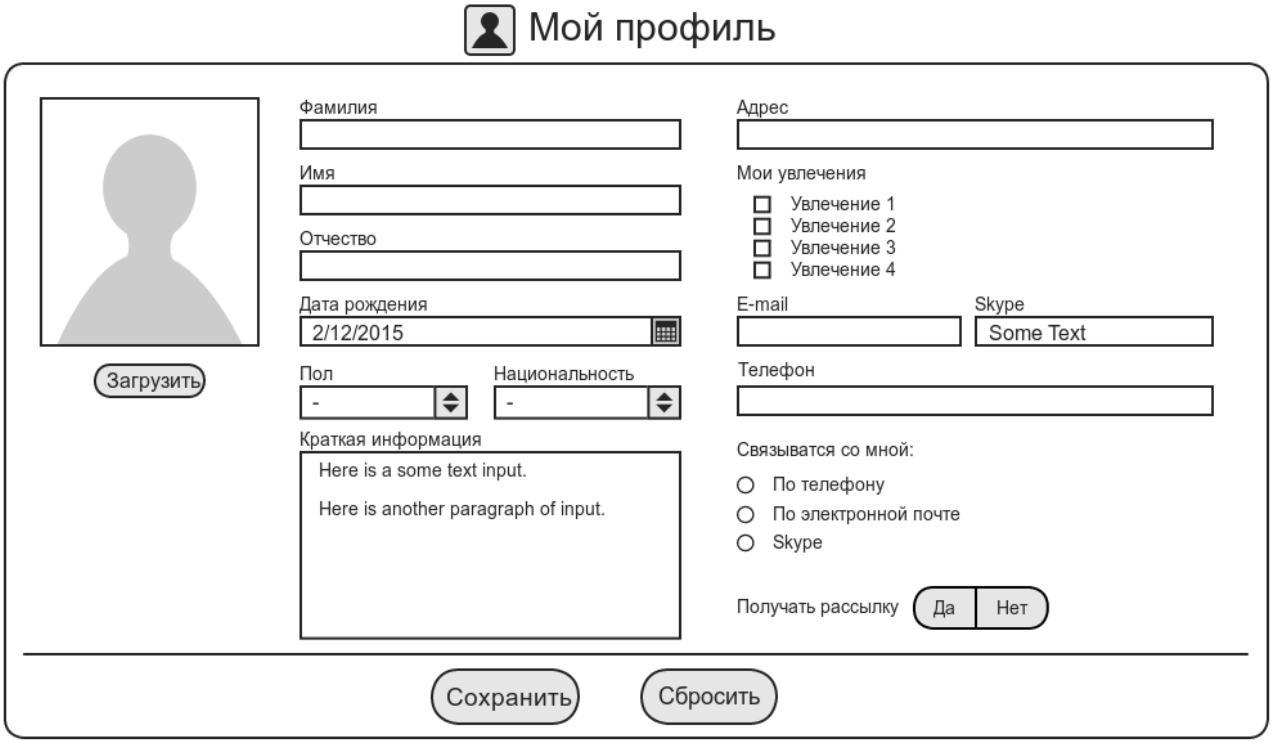


Рисунок 3.1 – Макет форми.

3. Передбачити можливість відображення форми на пристроях з різною роздільною здатністю.
4. При відображенні на пристроях шириною менших 720px поля типу textarea суховати.
5. Всі поля форми повинні мати перевірку на не пусте значення та відповідність формату вводу даних.
6. Поля другої колонки перевірити на відповідність формату використовуючи засоби HTML5 та JavaScript.

7. Поля третьої колонки перевірити на відповідність формату використовуючи PHP та у разі помилки відобразити відповідне повідомлення (використати відповідний компонент Bootstrap) не втративши даних, що вже заповнив користувач.

8. При завантаженні фото використати JavaScript об’єкт FileReader та показати фото профілю до відправки на сервер.

9. Відправити форму скрипту form.php та зберегти у файлі (самостійно обрати формат).

10. Передбачити неможливість повторного відправлення форми при оновленні сторінки (виконати перенаправлення сторінки).

11. При відкритті форми відобразити збережені у файлі дані.

12. Отриману веб-сторінки та пов'язані з нею дані (папка з картинками, файл стилів) скопіювати в папку «lab3» та розмістити її в кореневій директорій веб-сервера.

13. Протестувати роботу сторінок на віддаленому сервері.

14. Оформити звіт про виконання лабораторної роботи.

Додатково: Виконати обрізку та масштабування завантаженого фото перед збереженням.

Контрольні питання

1. Який html тег відповідає формі?
2. Які функції php використовуються при роботі з файлами?
3. Які типи передачі параметрів серверу існують?
4. Яким чином php – скрипт отримує доступ до переданих серверу параметрів?

Лабораторна робота №4 Динамічні веб – сторінки.

Мета: опанувати навички ділення HTML-сторінок на окремі блоки та генерація динамічних HTML сторінок з використанням мови програмування PHP.

Теоретичні відомості

Веб-сторінки можуть складатись із статичних текстових файлів, що зберігаються у файловій системі веб-сервера (статичні веб-сторінки), або веб-сервер може створювати сторінки за запитом браузера (динамічні веб-сторінки).

Динамічна веб-сторінка (англ. dynamic web page) — веб-сторінка, вміст якої може змінюватись. Частини динамічних сторінок зберігаються як окремі файли (статичні частини, php-скрипти) на диску веб-сервера. За запитом клієнта сторінка збирається з набору цих файлів (рис. 4.1) – технологія SSI (Server-Side Includes - включення на стороні сервера).

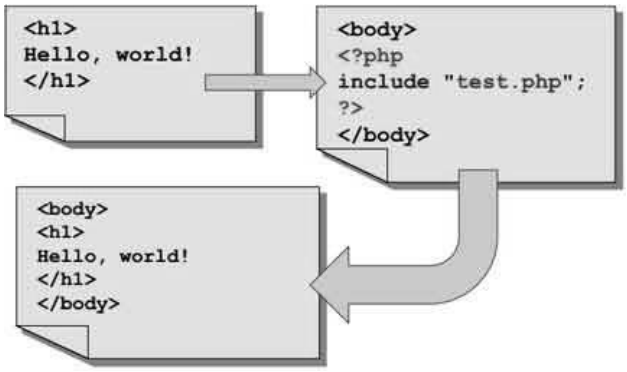


Рисунок 4.1 – Принцип роботи SSI

В PHP існують ряд функції емуляції SSI: include, include_once, require, require_once.

Різниця між функціями include і require полягає в типі генеруються помилок підключення. Якщо підключений файл відсутній, то include видасть помилку рівня «Warning» (тобто код продовжить виконаються). А require – видасть помилку рівня «Fatal error» (тобто код завершує виконаються). Щоб помилки взагалі не виводяться, можна поставити перед ім'ям функції знак «собаки» @ - вона гасить всі помилки, які сталися при виконанні коду. Так робити не треба, тому як @ дуже повільний оператор (краще використовувати функцію для керування відображенням помилок – error_reporting).

Для запобігання помилок множинного включення файли підключають через include_once або require_once. Різниця між якими аналогічна попередньої парі функцій.

Поділ на файли найчастіше відповідає структурним елементам сторінок. Як правило, структура сторінки сайту має чітку організацію. В якості прикладу розглянемо структуру наведену на рис. 4.2 яка складається з:

1. шапка (header), до якої можуть входити логотип, назва, слоган, телефон, меню навігації;

2. основна частина (content) – меню навігації по тематичних розділах, основна інформація, зображення, банери;

3. нижня частина (footer) – копірайт, адреси, телефони, лічильники та банери, меню.

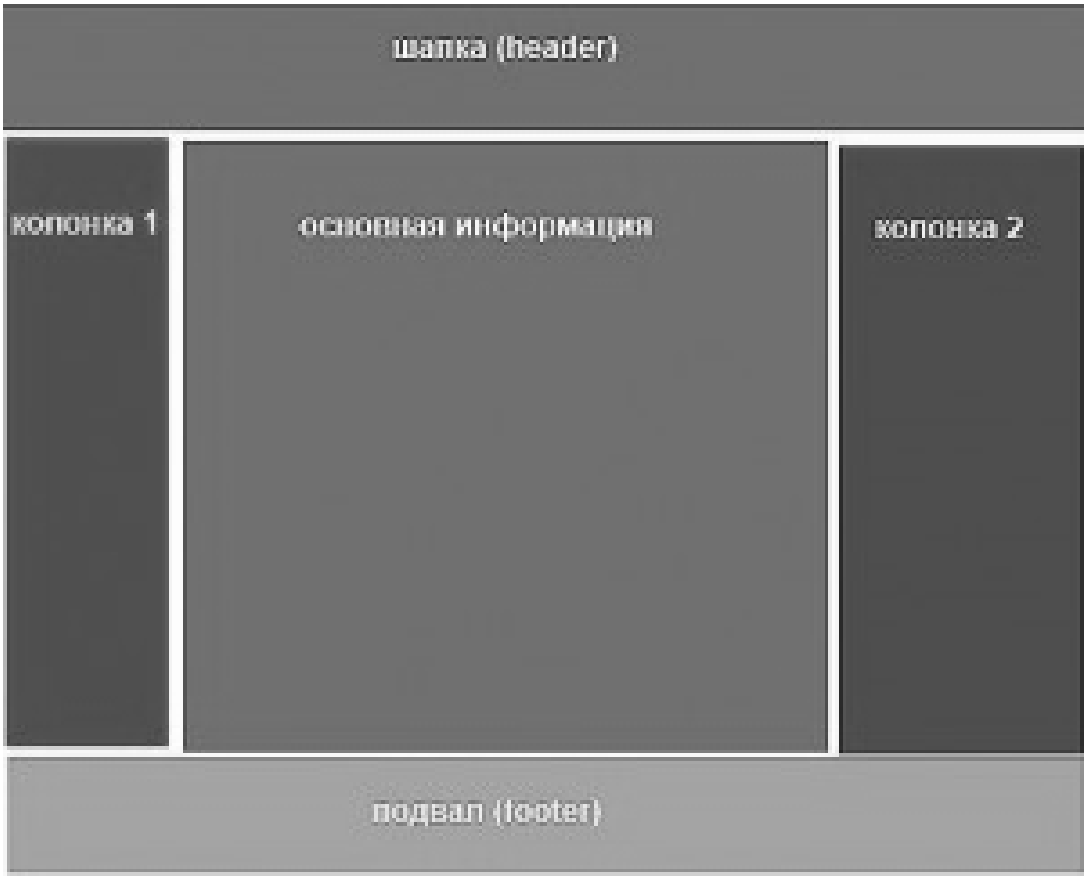


Рисунок 4.2 Структура оформлення сторінок сайту

Звичайно можливо різні варіанти та комбінації представленої структури.

Аналогічно діленню сторінок сайту на логічні елементи HTML-код сторінок може бути розбитий на відповідні блоки коду, збережений у вигляді php-скриптів і розкладені за призначенням в деревовидну структуру папок на веб-сервері. Приклад організації папок на сервері наведено на рис. 4.3.

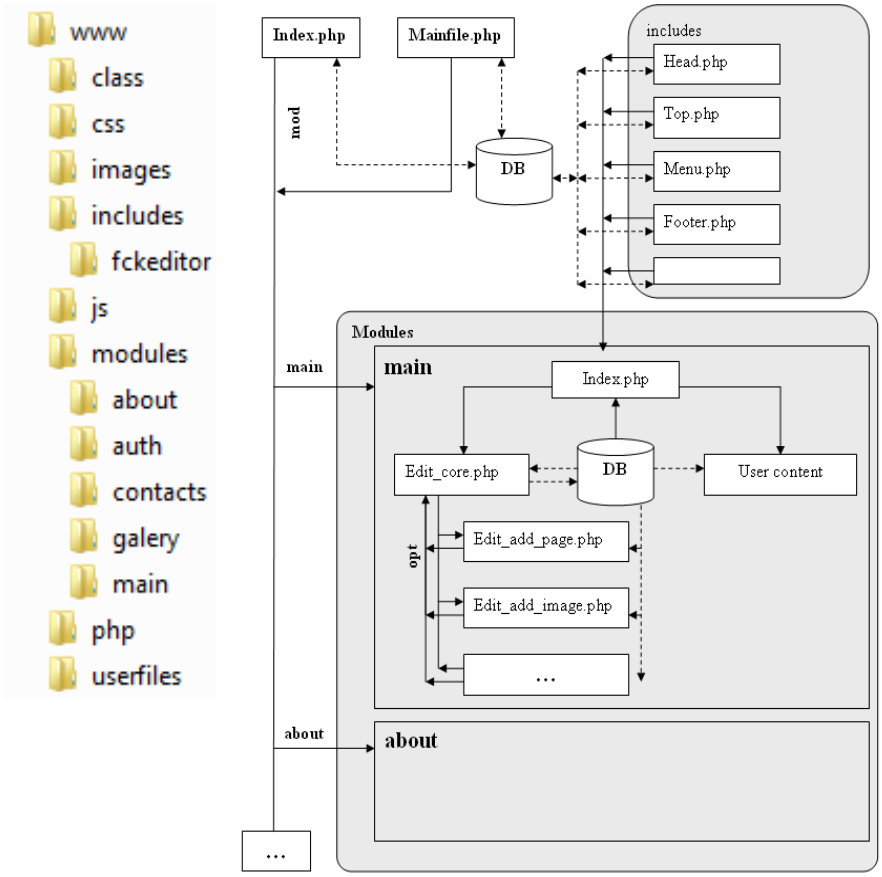


Рисунок 4.3 – Структура папок і модель «складання» сторінки сайту з окремих файлів (php-скриптів).

Завдання

1. Обрати довільний шаблон сайту за визначеною з викладачем тематикою.
2. Провести аналіз структурних елементів шаблону, виділити його загальні та характерні блоки, наповнити блоки необхідним змістом.
3. У кореневій веб-директорії створити паки: css, js, images, modules (рис. 4.3).
4. Розділити сторінки сайту на структурні елементи виконуючи наступні кроки:

– скопіювати стилі шаблону до папки css;
– скопіювати скрипти до папку js;
– скопіювати всі необхідні зображення до папку images;
– вибрати головну сторінку сайту та перенести її до папки «modules/main» з ім'ям index.php (якщо необхідно, виправити відносні шляхи до файлів стилю, зображень і т.д.);

– відкрити файл modules / main / index.php і послідовно вирізати його структурні елементи (шапку та нижню частину, верхне меню і т.д.), зберігаючи їх в окремих файлах папки includes та підключаючи їх використовую функції include або require.

– вміст основної частину сторінки залишаємо без змін.

– якщо, тест роботи сторінки пройшов вдало, то копіюємо отриманий index.php в інші модулі і змінюємо їх вміст..

5. Змінити файл index.php (він є основною точкою входу) так, щоб він виконував пошук прийнятого за методом GET параметру «mod» (за замовчуванням mod=main) та виконував підключення файлу index.php з відповідної папки modules.

6. Змінити посилання верхнього меню для виклику сторінок через одну точку входу (наприклад: index.php?mod=main – посилання на головну сторінку).

7. У кореневій папці веб-сервера створити директорію «lab4» та скопіювати до неї отримані файли..

8. Протестувати результати виконаної роботи.

9. Оформити звіт про виконання лабораторної роботи.

Контрольні питання

1. Яка різниця між функціями include та require?
2. Коли може виникнути помилка множинного підключення файлів?
3. Який скрипт як правило використовують у якості основної точкою входу?
4. Чому доцільно використовувати одну точку входу?

Для нотаток:

Лабораторна робота №5. Система авторизації з використанням сесії.

Мета:навчитися використовувати механізм сесій PHP для системи авторизації.

Теоретичні відомості

Протокол HTTP є протоколом «без збереження стану». Це означає, що даний протокол не має вбудованого способу збереження стану між двома транзакціями.

Сесії та **куки** (**cookies**) призначені для зберігання відомостей про користувачів при переходах між сторінками сайту. При використанні сесій дані зберігаються у тимчасових файлах на сервері. Файли з куками зберігаються на комп'ютері користувача, і за запитом відсилаються броузером сервера.

При роботі з сесіями розрізняють наступні етапи:

відкриття сесії

реєстрація змінних сесії і їх використання

закриття сесії

Найпростіший спосіб відкриття сесії полягає у використанні функції session_start, яка викликається на початку PHP-сценарію:

```
bool session_start ( void )
```

Ця функція перевіряє, чи існує ідентифікатор сесії, і, якщо неіснує, то створює його. Якщо ідентифікатор поточної сесії вже існує, то завантажуються зареєстровані змінні сесії.

Після ініціалізації сесії з'являється можливість зберігати інформацію в глобальному масиві $_SESSION. Наприклад, є файл index.php у якому в масив $_SESSION зберігається змінна і масив.

```
<?php
  // Ініціюємо сесію
  session_start();
  // Записуємо значення в сесію
  $_SESSION['name'] = "value" ;
   //Записуємо масив у сесію
  $arr = array("first", "second", "third");
  $_SESSION ['arr'] = $arr;
  //Виводимо посилання на іншу сторінку
  echo "<a href="other.php">інша сторінка</a>" ;
?>
```

На сторінках, де відбувається виклик функції session_start (), значення даних змінних можна витягти з масиву $_SESSION. У наступному прикладі

коду приводиться вміст сторінки other.php, де беруться дані, раніше записані у сесію на сторінці index.php.

```
<?php
//Ініціюємо сесію
session_start();
//Виводимо вміст суперглобального масиву $_SESSION
print_r($_SESSION);
?>
```

Після завершення роботи з сесією спочатку потрібно видалити всі змінні сесії (session_unset), а потім викликати функцію session_destroy.

Cookies – це текстові дані, що зберігаються на стороні клієнта, і містять пари «ім'я – значення», з якими пов'язаний URL, за яким броузер визначає чи потрібно надсилати cookies на сервер.

Установка cookies проводиться за допомогою функції setcookie.

```
bool setcookie ( string name [, string value [, INT Expire [, String Path [, String Domain [, INT Secure ]]]]] )
```

де name – ім'я встановлюваного cookie;

value – значення, що зберігається в cookie з ім'ям name ;

expire – час в секундах з початку епохи-юнікс, по закінченні якого поточні cookie стають недійсними;

path – шлях, по якому доступний cookie;

domain – домен, з якого доступний cookie;

secure – директива, яка визначає, чи доступний cookie не по протоколу HTPPS. За замовчуванням ця директива має значення 0, що означає можливість доступу до cookie по звичайному протоколу HTTP.

Cookies треба обов'язково встановлювати перед відправкою в броузер будь-яких заголовків (будь-якої інформації що повинна відображатися на сторінці), оскільки самі cookies встановлюються у вигляді заголовків.

Значення, що зберігається в cookie можна отримати через суперглобальний масив $_COOKIE.

За замовчуванням cookies встановлюються на один сеанс роботи з броузером, однак можна задати для них більш тривалий термін існування. Це дуже зручна і корисна властивість, оскільки в цьому випадку користувачеві не потрібно надавати свої дані знову при кожному відвідуванні сайту.

Видалити cookie просто. Для цього треба викликати функцію setcookie і передати їй ім'я того cookie, який підлягає видаленню.

Завдання

1. Додати до сайту, створеного на попередніх лабораторних роботах, модуль авторизаціїз (auth), який буде викликатися за посиланням «Авторизація» з верхньої або нижньої частини будь якої сторінки сайту (тобто для авторизації не відводитися окремий пункт головного меню).

2. Створити у модулі скрипт index.php який приймає, окрім параметру mod – ім'я модулю, параметр opt – вказую на одну з можливих дій процесу авторизації (login або logout), та виводить форму авторизації[4]. Форма повинна мати наступну елементи вводу та атрибути:

– «Логін користувача» (ім'я параметру – username, type='text');
– «Пароль користувача» (ім'я параметру – username, type = 'password');
– кнопка «Авторизація»;
– приховані параметри mod="auth" та opt = "login";
– action = "index.php";
– method = "POST".

3. Срипт index.php повинен реалізувати наступні дій в залежності від значення параметру opt:

– якщо такого параметру не існує і користувач вже зареєстрований вивести ім'я користувача, інакше вивести форму авторизації;
– login – перевіряти реєстраційні дані[5], що вводить користувач; якщо данні коректні вивести відповідне повідомлення та зареєструвати його ім'я у змінній сесії ($_SESSION["user"] = "admin"), а якщо данні не коректні вивести відповідне повідомлення та форму авторизації;
– logout – завершити сеанс роботи користувача; видалити всі дані сесії, завершити сесію та виконати перехід на головну сторінку сайту.

4. Якщо користувач зареєстрований (isset($_SESSION["user"]) == true) змінити посилання «Авторизація» на посилання «Імя користувача – Завершити сеанс» що виконує завершення сеансу роботи користувача

5. Отримані сторінки та пов'язані з ними дані (папка з картинками, файл стилів) скопіювати в кореневу папку веб-сервера.

6. Протестувати роботу сторінок на віддаленому сервері.

7. Оформити звіт про виконання лабораторної роботи.

Контрольні питання

1. Яка різниця між сесією ту cookie?
2. Як виконати перехід на іншу сторінку сайту у коді PHP?
3. Як встановити змінну сесії?
4. Як правильно завершити роботу сесії?

[3] Під «додаванням модулю» слід розуміти створення нової папки у папці modules з кореневої директорій сайту.
[4] Може бути реалізована у окремому скрипті
[5] Для простоти вважати що існує один користувач «admin» з паролем «111»

5. Де правильно викликати функції session_start()?

Лабораторна робота №6. Пакетний менеджер.

Мета: опанувати основи використання та створення пакетів на основі менеджера Composer.

Теоретичні відомості.

Composer – это пакетный менеджер уровня приложений для языка программирования PHP, который предоставляет средства по управлению зависимостями в PHP-приложении.

Composer работает через интерфейс командной строки и устанавливает зависимости (например библиотеки) для приложения. Он также позволяет пользователям устанавливать PHP-приложения, которые доступны на packagist.org, который является его основным репозиторием, где содержатся все доступные пакеты.

Таблиця 6.1 – Основні команди пакетного менеджера Composer

Команда	Опис	Приклад
init		
install		
update		

Приклад файлу composer.json

```
{
    "name": "munspel/web-lab",
    "description": "Exemple 1. Simple composer packege",
    "require": {
        "twig/twig": "^1.24"
    },
    "authors": [
        {
            "name": "Andrey Lisnyak",
            "email": "munspel@ukr.net"
        }
    ],
    "minimum-stability": "dev",

    "autoload":{
        "psr-4":{
            "WebLab\\":"core/"
        }
    }
}
```

Git – розподілена система керування версіями файлів та спільної роботи. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються

на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

Работа с Git в большинстве случаев означает работу в команде, поэтому файлы и директории, которые специфичны для данного проекта и для вашего рабочего окружения (например, логи, файлы IDE), необходимо исключить из фиксации. Для этого используется файл .gitignore .

Приклад виконання роботи можна переглянути виконавши наступні команди:

```
$ git clone https://github.com/lisnayk/web-lab.git
$ git checkout –f web-lab-01
$ composer install
```

Завдання

1. Для розробки проекту реалізувати на сервері наступну структуру каталогів:

```
webroot\
  xyzcore
     XYZCore.php
  web
     index.php
```

де xyz – перші літери прізвища, ім’я та по батькові англійського алфавіту.

2. Використовуючи команду **init** пакетного менеджера **composer** створити файл composer.json у кореневый директорії проекту, вказавши наступні парметри (не вказані залишити пустими):
   - Name (<vendor>/<name>): xyz/web-labs;
   - Description: Web-labs of XYZ;
   - Author: X Y <xyz@gmail.com>
   - До залежностей (dependencies) додати пакет twig/twig.
3. Додати до файлу composer.json параметри для генерації автозавантаження класів директорії xyzcore у простір імен XYZWebLab за стандартом psr-4 (дивись приклад у теоретичному матеріалі).
4. У файлі XYZCore.php описати клас XYZCore, який у конструкторі виводить коротку інформацію про виконавця роботи.
5. У файлі index.php підключити скрипт автозавантаження (__DIR__.'/../vendor/autoload.php') та створити екземпляр класу XYZCore.
6. Виконати команду **install** пакетного менеджера **composer** та протестувати роботу скрипта index.php
7. Створити комміт проекту у системи контролю версій git, виключивши з фіксації директорію vendor та файли composer.lock й додавши файл опису проекту README.md.

8. Предметом захисту роботи є робочій проект та вихідні файли у репозиторії git.

Контрольні питання

1. Яке призначення файлу composer.lock?
2. Яке призначення та синтаксис файлу .gitignore?
3. Приведіть основні характеристики сндрарту автозавантаженя psr-4.
4. Які переваги використання системи контроля версій?
5. Перерахуйте відомі Вам системи контролю версій.
6. Наведіть основні команди пакетного менеджера composer.
7. Яка команда пакетного менеджера composer використовується для включення пакету до проекту?

Лабораторна робота №7. Реалізація патерна Singelton.

Мета:навчитися використовувати можливості PHP для обробки графічних файлів та бібліотеку jQuery для налаштування відображення графічних файлів браузером.

Теоретичні відомості

Паттерн проектирования (design pattern) – это общее типовое решение некоторой проблемы, многократно повторяемое в процессе проектирования архитектуры программного продукта. Они показывают отношения и взаимодействия между классами, позволяют сделать систему гибкой и легко изменяемой. За счет их правильного использования повышается коэффициент использования готовых решений.

Singleton – один из самых простых для понимания паттерн. Основное назначение – гарантировать существование только одно экземпляра класса. Причиной обычно является следующее:

- требуется только один объект исходного класса;
- объект должен был доступен в любом месте приложения.

Singleton относиться к классу порождающих паттернов и в схеме MVC используется для порождения главного контроллера.

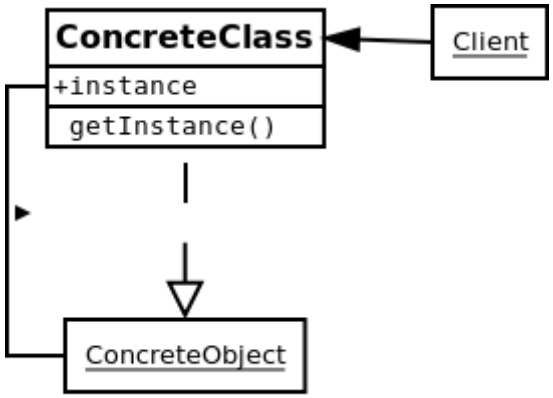


Рисунок 7.1 – Суть паттерна **Singleton.**

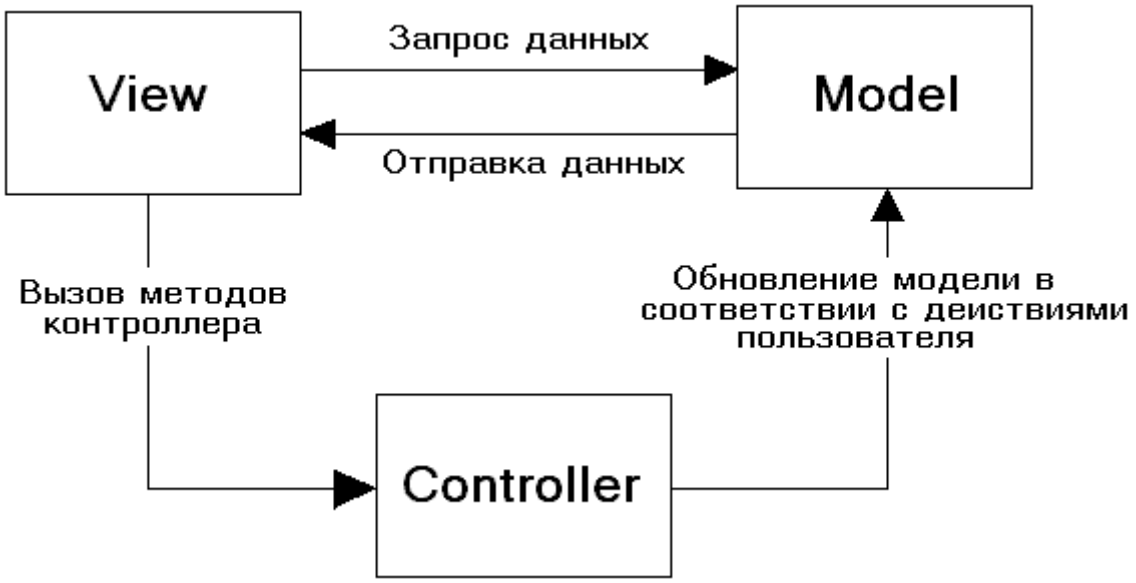
Реализация данного паттерна на языке программирования PHP следующая:

```
class SingletonTest {

    /**
     * Статическая переменная, в которой мы
     * будем хранить экземпляр класса
     *
```

```
     * @var SingletonTest
     */
    protected static $_instance;

    /**
     * Закрываем доступ к функции вне класса.
     * Паттерн Singleton не допускает вызов
     * этой функции вне класса
     *
     */
    private function __construct() {
        /**
         * При этом в функцию можно вписать
         * свой код инициализации. Также можно
         * использовать деструктор класса.
         * Эти функции работают по прежднему,
         * только не доступны вне класса
         */
    }

    /**
     * Закрываем доступ к функции вне класса.
     * Паттерн Singleton не допускает вызов
     * этой функции вне класса
     *
     */
    private function __clone() {

    }

    /**
     * Статическая функция, которая возвращает
     * экземпляр класса или создает новый при
     * необходимости
     *
     * @return SingletonTest
     */
    public static function getInstance() {
        // проверяем актуальность экземпляра
        if (null === self::$_instance) {
            // создаем новый экземпляр
            self::$_instance = new self();
        }
        // возвращаем созданный или существующий экземпляр
        return self::$_instance;
    }

}
```

Паттерн MVC описывает простой способ построения структуры приложения, целью которого является отделение бизнес-логики от пользовательского интерфейса. В результате, приложение легче масштабируется, тестируется, сопровождается и конечно же реализуется.

Рассмотрим концептуальную схему шаблона MVC:



В архитектуре MVC модель предоставляет данные и правила бизнес-логики, представление отвечает за пользовательский интерфейс, а контроллер обеспечивает взаимодействие между моделью и представлением.

Типичную последовательность работы MVC-приложения можно описать следующим образом:

1. При заходе пользователя на веб-ресурс, скрипт инициализации создает экземпляр приложения и запускает его на выполнение. При этом отображается вид, скажем главной страницы сайта.
2. Приложение получает запрос от пользователя и определяет запрошенные контроллер и действие. В случае главной страницы, выполняется действие по умолчанию (index).
3. Приложение создает экземпляр контроллера и запускает метод действия,
   в котором, к примеру, содержаться вызовы модели, считывающие информацию из базы данных.
4. После этого, действие формирует представление с данными, полученными из модели и выводит результат пользователю.

Контроллер — связующее звено, соединяющее модели, виды и другие компоненты в рабочее приложение. Контроллер отвечает за обработку запросов пользователя. Контроллер не должен содержать SQL-запросов. Их лучше держать в моделях. Контроллер не должен содержать HTML и другой разметки. Её стоит выносить в виды.

В хорошо спроектированном MVC-приложении контроллеры обычно очень тонкие и содержат только несколько десятков строк кода.

Логика контроллера довольно типична и большая ее часть выносится в базовые классы.

Приклад виконання роботи можна переглянути виконавши наступні команди:

```
$ git clone https://github.com/lisnayk/web-lab.git
$ git checkout -f web-lab-02
$ composer install
```

Завдання

1. Реалізувати клас XYZCore з попередньої лабораторної роботи за патроном проектування **Singleton.**
2. Реалізувати клас XYZController – базовий клас контролера.
3. У директорії web/Controllers створити нащадка XYZController SiteContrller з публічним методом doIndex().
4. У класі XYZCore реалізувати метод **run**, який у якості параметра приймає повне ім'я конфігураційного файлу, зберігає його вміст у приватному полі $config.
5. Реалізувати метод ксласу XYZCore (наприклад handleRequest), що обробляє параметр $_REQUEST['q'] (q – запит у форматы **controller/action),** створює экземпляр класу відповідного контролеру та викликає метод doAction.
6. Створити конфігураційний файл для визначення параметрів додатку у директорії core/config з наступними параметрами:
   - назва сайту;
   - контролер та дія за замовченням;
   - автор роботи.
7. У контролері за замовченням та методі за замовченням вивести таблицю множення з лабораторної роботи № 2.
8. Створити комміт проекту у системи контролю версій git
9. Предметом захисту роботи є робочій проект та вихідні файли у репозиторії git.

Контрольні питання

Лабораторна робота №8. Обробка та відображення графічних файлів.

Мета:навчитися використовувати можливості PHP для обробки графічних файлів та бібліотеку jQuery для налаштування відображення графічних файлів браузером.

Теоретичні відомості

Загрузка файлов методом POST ¶

Данная возможность позволяет загружать как текстовые, так и бинарные файлы. С помощью PHP-функций манипуляции файлами вы получаете полный контроль над тем что должно быть сделано после их загрузки.

Конфигурация загрузки файлов на сервере регламентируется набором параметров (директив): file_uploads, upload_max_filesize, upload_tmp_dir, post_max_size и max_input_time конфигурационного файла php.ini

Страница для загрузки файлов может быть реализована при помощи специальных элементов ввода формы:

```
<!-- Тип кодирования данных, enctype, ДОЛЖЕН БЫТЬ указан ИМЕННО так -->
<form enctype="multipart/form-data" action="__URL__" method="POST">
    <!-- Поле MAX_FILE_SIZE должно быть указано до поля загрузки файла -->
    <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
    <!-- Название элемента input определяет имя в массиве $_FILES -->
    Отправить этот файл: <input name="userfile" type="file" />
    <input type="submit" value="Send File" />
</form>
```

В приведенном выше примере *__URL__* необходимо заменить ссылкой на PHP-скрипт, обрабатывающий загрузку файла.

Скрытое поле *MAX_FILE_SIZE* (значение необходимо указывать в байтах) должно предшествовать полю для выбора файла, и его значение является максимально допустимым размером принимаемого файла в PHP.

Также следует убедиться, что в атрибутах формы вы указали *enctype="multipart/form-data"*, в противном случае загрузка файлов на сервер выполняться не будет.

Глобальный массив $_FILES содержит всю информацию о загруженных файлах. Его содержимое для формы выше приводится ниже. Обратите внимание, что здесь предполагается использование имени *userfile* для поля выбора файла, как и в приведенном выше примере. На самом деле имя поля может быть любым.

$_FILES['userfile']['name'] – оригинальное имя файла на компьютере клиента.

$_FILES['userfile']['type'] – Mime-тип файла, в случае, если браузер предоставил такую информацию. Пример: *"image/gif"*. Этот mime-тип не проверяется в PHP, так что не полагайтесь на его значение без проверки.

$_FILES['userfile']['size'] – размер в байтах принятого файла.

$_FILES['userfile']['tmp_name'] – временное имя, с которым принятый файл был сохранен на сервере.

$_FILES['userfile']['error'] – код ошибки, которая может возникнуть при загрузке файла.

По умолчанию принятые файлы сохраняются на сервере в стандартной временной папке до тех пор, пока не будет задана другая директория при помощи директивы upload_tmp_dir конфигурационного файла php.ini.

Проверка загружаемых на сервер файлов

Для получения более детальной информации вы можете ознакомиться с описанием функций is_uploaded_file() и move_uploaded_file(). Следующий пример принимает и обрабатывает загруженный при помощи формы файл.

```
<?php
// В PHP 4.1.0 и более ранних версиях следует использовать $HTTP_POST_FILE
S
// вместо $_FILES.

$uploaddir = '/var/www/uploads/';
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);

echo '<pre>';
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    echo "Файл корректен и был успешно загружен.\n";
} else {
    echo "Возможная атака с помощью файловой загрузки!\n";
}

echo 'Некоторая отладочная информация:';
print_r($_FILES);

print "</pre>";

?>
```

PHP-скрипт, принимающий загруженный файл, должен реализовывать логику, необходимую для определения дальнейших действий над принятым файлом. Например, вы можете проверить переменную $_FILES['userfile']['size'], чтобы отсечь слишком большие или слишком маленькие файлы. Также вы можете использовать переменную $_FILES['userfile']['type'] для исключения файлов, которые не удовлетворяют критерию касательно типа файла, однако, принимайте во внимание, что это поле полностью контролируется клиентом, используйте его только в качестве первой из серии проверок. Также вы можете использовать$_FILES['userfile']['error'] и коды ошибок при реализации вашей логики. Независимо от того, какую модель поведения вы выбрали, вы должны удалить файл из временной папки или переместить его в другую директорию.

В случае, если при отправке формы файл выбран не был, PHP установит переменную $_FILES['userfile']['size'] значением 0, а переменную $_FILES['userfile']['tmp_name'] - пустой строкой. none.

По окончанию работы скрипта, в случае, если принятый файл не был переименован или перемещен, он будет автоматически удален из временной папки.

Приклад виконання роботи можна переглянути виконавши наступні команди:

```
$ git clone https://github.com/lisnayk/web-lab.git
$ git checkout –f web-lab-03
$ composer install
```

Завдання

На базі прикладу виконати наступні завдання.

1. Реалізувати форму для завантаження зображень що включає в себе наступні елементи:
   - посилання на файл з зображенням;
   - підпис зображення;
   - випадаючий список з розмірами великого варіанту зображення;
   - випадаючий список з розміром маленького зображення;
   - випадаючий список з вказанням катерогії фотографій;
2. Для розробленої форми розробити відповідну структуру таблиць бази даних.
3. Розробити скрипт який виконує обробку форми, зберігає оброблені фото у окремих директоріях з унікальними іменами та створює (або оновлює) відповідний запис у базі даних.
4. Створити адміністративну сторінку яка має посилання на створення нового елементу, виводить усі збережені фото у табличному вигляді та відображає: мініатюру, заголовок, категорію, елементи керування (редагувати, видалити).
5. Розробити сторінку для відображення галереї зображень за категоріями.
6. Для перегляду повно розмірного фото використати будь-який існуючий JavaScript плагін (ColorBox, LightBox, або інший).

Контрольні питання

Індивідуальне завдання

Використовуючи напрацювання отримані при виконанні лабораторних робіт та приклади, що доступні у системі контролю версій (https://github.com/lisnayk/web-lab.git) розробити (доопрацювати) необхідний функціонал для публікації новин сайту. Необхідно задовольнити наступні вимоги:

1. Можливість зберігати та відображати фото анонсу, автора, короткий зміст, повний зміст, дату публікації, статус публікації, категорію публікації (Спорт, Політика, Наука, …), одну або декілька (максимум 10) фото з місця подій, відео yotoube при наявності.
2. Реалізувати форму для створення та редагування та видалення публікацій.
3. Реалізувати сторінку для відображення анонсів всіх новин та за категоріями.
4. Сторінку для перегляду обраної новини.

Реалізований проект розмістити на власній сторінці системи контролю версій.

Предметом захисту роботи є робочій проект та вихідні коди.

Лабораторна робота №9. Об’єктно-оріентовані моделі елементів суйту.

Лабораторна робота №10. Використання шаблонізаторів.

Лабораторна робота №10. Створення системи меню.

Лабораторна робота №11. Створення сторінок з використанням технології AJAX.

Мета: опанувати можливості технології AJAX на прикладі створення системи новин.

Кінцевий результат

Результати виконання переліку лабораторних робіт повинні привести до створення елементарної системи керування вмістом. Система повинна буди побудована на наступних принципах та засадах:

- мати едину точку входу (скрипт index.php);
- мати систему авторизації та реєстрації;
- реалізовувати роботу з декілька ми видами контенту: стаття, галерея зображень, новини.
- кожен з видів контенту повинен мати власну модель та наслідувати базову модель веб-сторінки.
- формування виводу усіх сторінок виконується на базі шаблонів;
- реалізовано систему створення багаторівневого меню.
- для роботи з новинами реалізувати можливості видалення матеріалів без перезавантаження сторінки.

У якості дизайну може бути використано довільний безкоштовний шаблон завантажений з мережі.

Література

1. Usage of web servers for websites – [http://w3techs.com/technologies/overview/web_server/all]
2. Usage of server-side programming languages for websites – [http://w3techs.com/technologies/overview/programming_language/all]