М.О. Слабінога

БЕКЕНД-РОЗРОБКА МОВОЮ PHP

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра комп'ютерних систем і мереж

М. О. Слабінога

БЕКЕНД-РОЗРОБКА МОВОЮ PHP

ЛАБОРАТОРНИЙ ПРАКТИКУМ

Івано-Франківськ

2022

УДК 004.75

С 47

Рецензент:

Я.І. Заячук кандидат технічних наук, доцент кафедри комп’ютерних систем і мереж Івано-Франківського національного технічного університету нафти і газу

Рекомендовано методичною радою університету
(протокол № від р.)

Слабінога М.О.
С 47. Бекенд-розробка мовою PHP: лабораторний практикум / М.О. Слабінога – Івано-Франківськ: ІФНТУНГ, 2022. – 64 с.

МВ 02070855- -2022

Лабораторний практикум з дисципліни “Бекенд-розробка мовою PHP” розроблений відповідно до робочої програми навчальної дисципліни та робочого навчального плану.

Призначено для підготовки бакалаврів за спеціальністю 123 - “Комп’ютерна інженерія”. Лабораторний практикум може бути використаний студентами очної та заочної форм навчання.

УДК 004.75
С 47

Рецензент:

Заячук Я.І. кандидат технічних наук, доцент кафедри комп’ютерних систем і мереж Івано-Франківського національного технічного університету нафти і газу

Рекомендовано методичною радою університету
(протокол № від)

Слабінога М.О.
С 47. Бекенд-розробка мовою PHP / М.О. Слабінога – Івано-Франківськ: ІФНТУНГ, 2022. – 64 с.
МВ 02070855- -2022

Лабораторний практикум з дисципліни “Бекенд-розробка мовою PHP” розроблений відповідно до робочої програми навчальної дисципліни та робочого навчального плану.

Призначено для підготовки бакалаврів за спеціальністю 123 - “Комп’ютерна інженерія”. Лабораторний практикум може бути використаний студентами очної та заочної форм навчання.

УДК 004.75

МВ 02070855- -2022

Завідувач кафедри комп’ютерних систем і мереж — С.І. Мельничук

Узгоджено:

Член експертно-рецензійної комісії університету — В.В. Бандура

Нормоконтролер — Г. Я. Томашівська

Провідний бібліотекар НТБ — Г. М. Мацюк

ЗМІСТ

ЗАГАЛЬНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Мета вивчення дисципліни – набуття фахівцями компетенцій в розробці серверного програмного забезпечення мовою PHP

У результаті вивчення дисципліни студент повинен демонструвати такі **результати навчання** через знання, уміння та навички:

- Вміти здійснювати пошук інформації в різних джерелах для розв'язання задач комп'ютерної інженерії.
- Знати новітні технології в галузі комп'ютерної інженерії.
- Якісно виконувати роботу та досягати поставленої мети з дотриманням вимог професійної етики.

Вивчення навчальної дисципліни передбачає формування та розвиток у студентів компетентностей, передбачених відповідним стандартом вищої освіти України:

загальні:

Z3. Здатність застосовувати знання в практичних ситуаціях

Z7. Вміння виявляти, ставити та вирішувати проблеми

фахові:

P2. Здатність використовувати сучасні методи і мови програмування для розроблення алгоритмічного та програмного забезпечення.

P3. Здатність створювати системне та прикладне програмне забезпечення комп'ютерних систем і мереж.

ЛАБОРАТОРНА РОБОТА № 1

Вступ до мови програмування PHP

Мета роботи: Навчитися створювати файли з програмним кодом мовою PHP та ознайомитися з її функціями

Теоретичні відомості

PHP (англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні вебсервера. PHP є однією з найпоширеніших мов, що використовуються у сфері веброзробок (разом із Java, .NET, JavaScript, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP — проєкт відкритого програмного забезпечення.

PHP інтерпретується вебсервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, тому що браузер отримує готовий html-код. Це є перевагою з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

PHP — мова, у код якої можна вбудовувати безпосередньо html-код сторінок, які, у свою чергу, коректно оброблюватимуться PHP-інтерпретатором. Обробник PHP просто починає виконувати код після відкриваючого тегу (<?php) і продовжує виконання до того моменту, поки не зустріне закриваючий тег.

Велика різноманітність функцій PHP дає можливість уникати написання багаторядкових функцій, призначених для користувача, як це відбувається в C або Pascal.

Наявність інтерфейсів до багатьох баз даних

у PHP вбудовані бібліотеки для роботи з MySQL, PostgreSQL, SQLite, mSQL, Oracle, dbm, Hyperware, Informix, InterBase, Sybase.

завдяки стандарту відкритого інтерфейсу зв'язку з базами даних (англ. Open Database Connectivity Standard, ODBC) можна підключатися до всіх баз даних, до яких існує драйвер.

Нетрадиційність

Мова PHP здаватиметься знайомою програмістам, що працюють в різних областях. Багато конструкцій мови запозичені з C, Perl. Код PHP дуже схожий на той, який зустрічається в типових програмах мовами C або Pascal. Це помітно знижує початкові зусилля при вивченні PHP. PHP — мова, що поєднує переваги Perl та C і спеціально спрямована на роботу в Інтернеті, мова з універсальним і зрозумілим синтаксисом. І хоча PHP є досить молодою мовою, вона здобула таку популярність серед web-програмістів, що в наш час є найпопулярнішою мовою для створення вебзастосунків (скриптів).

Наявність сирцевого коду та безкоштовність

Стратегія Open Source, і розповсюдження початкових текстів програм в масах, безсумнівно справили сприятливий вплив на багато проєктів, в першу чергу — Linux хоч і успіх проєкту Apache сильно підкріпив позиції прихильників Open Source. Сказане відноситься і до історії створення PHP, оскільки підтримка користувачів зі всього світу виявилася дуже важливим чинником в розвитку проєкту PHP. Ухвалення стратегії Open Source і

безкоштовне розповсюдження початкових текстів PHP надало неоціненну послугу користувачам. Окрім цього, користувачі PHP в усьому світі є свого роду колективною службою підтримки, і в популярних електронних конференціях можна знайти відповіді, навіть на найскладніші питання.

Ефективність

Ефективність є дуже важливим чинником у програмуванні для середовищ розрахованих на багато користувачів, до яких належить і web. Важливою перевагою PHP є те, що ця мова належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю. За деякими оцінками, більшість PHP-сценаріїв (особливо не дуже великих розмірів) обробляються швидше за аналогічні їм програми, написані на Perl. Проте хоч би що робили розробники PHP, виконавчі файли, отримані за допомогою компіляції, працюватимуть значно швидше — в десятки, а іноді і в сотні разів. Але продуктивність PHP достатня для створення цілком серйозних вебзастосунків.

Всі сценарії оформляються у вигляді блоків коду. Ці блоки можуть бути поміщені в HTML-код, але відділені від нього відповідними обмежувачами. Код PHP в HTML повинен знаходитись між початковим тегом <?php та кінцевим ?> (або між <script language="php"> та </script>) Бажаним варіантом виділення PHP коду є варіант <?php ?>, оскільки саме такі початковий та кінцевий теги дозволять використовувати PHP код у документах, які відповідають правилам XML. Також можна використовувати скорочений запис <? ?> (інколи потрібно активізувати даний стиль у файлі налаштувань інтерпретатора php.ini: змінна short_open_tag повинна мати значення On) і записом у стилі ASP: <% %> (в php.ini змінна asp_tags повинна мати значення On). Проте, стиль ASP не рекомендується і очікується, що він буде відсутній у PHP 6[джерело?].

Найпростіша програма Hello world на PHP виглядає так:

```
<?php
   echo 'Hello, world!';
 ?>
```

PHP виконує код, що знаходиться в середині обмежувачів на кшталт <?php ?>. Все, що знаходиться поза обмежувачами, виводиться без змін. Таким чином виконується вставка PHP-коду в HTML-код. Наприклад, код html-сторінки з попереднім прикладом виглядатиме так:

```
<html>
  <head>
    <title>Тестуємо PHP</title>
  </head>
  <body>
    <?php echo 'Hello, world!'; ?>
  </body>
</html>
```

Завдання на лабораторну роботу

1. Створіть php-файл з кодом, який виводить у браузері "Hello World!".

2. Перевірте написаний код в онлайн трансляторі мови PHP (наприклад, тут) https://www.w3schools.com/php/php_compiler.asp

3. Відредагуйте файл php, щоб код виводив ваше ім'я замість "Світ" у фразі "Привіт Світ".

4. Перевірте написаний код.

5. Зареєструйте обліковий запис на Bitbucket (bitbucket.org) або Github (github.com).

6. Створіть проект та репозиторій для своїх лабораторних робіт.

7. Завантажте свій php-файл у репозиторій.

Контрольні запитання

1. Якими символами починається та завершується PHP код у файлі?
2. Які правила найменування змінних у PHP?
3. Які типи даних використовуються в PHP?

ЛАБОРАТОРНА РОБОТА № 2

Встановлення та налаштування локального сервера. Основні файли конфігурації сервера

Мета роботи: Навчитися встановлювати пакет, що включає в себе веб-сервер, інтерпретатор мови та сервер баз даних, а також ознайомитися з основними файлами конфігурацій.

Теоретичні відомості

XAMPP — безкоштовна багатоплатформова збірка вебсервера з відкритим початковим кодом, що містить HTTP-сервер Apache, базу даних MariaDB, MySQL й інтерпретатори скриптів для мов програмування PHP та Perl, а також додаткові бібліотеки, що дозволяють запустити повноцінний вебсервер.

XAMPP - це акронім:

- 'X' (будь-яка з чотирьох операційних систем)
- 'A'pache
- 'M'ySQL
- 'P'HP

- 'P'erl.

Повний пакет містить:

- Web-сервер Apache з підтримкою SSL
- СКБД MySQL
- Утиліту phpMyAdmin.
- PHP
- FTP-сервер FileZilla
- Perl
- Сервлет контейнер Apache Tomcat
- POP3/SMTP сервер

Для windows надається панель для управлінням встановленими на сервері засобами XAMPP Control Panel.

Для встановлення XAMPP необхідно завантажити всього один файл формату zip, tar або exe, а компоненти програми не вимагають настройки. Програма регулярно оновлюється, для включення до складу новітніх версій Apache / MySQL / PHP та Perl. Також XAMPP йде з безліччю інших модулів, включаючи OpenSSL та phpMyAdmin.

Для користувача інтерфейс програми настільки простий, що її називають «збіркою для ледачих» («lazy man's WAMP / LAMP installation»).

Установка XAMPP займає менше часу, ніж установка кожного компонента окремо. Цей web-сервер поширюється в повній, стандартній і мінімальній (відомої як XAMPP Lite) версіях. Всі додаткові модулі також доступні для скачування.

З додаткових можливостей можна відзначити, що сама компанія випускає пакети оновлення, які випускаються у вигляді zip, 7-zip, tar або exe, які дозволяють оновити всі компоненти з однієї версії збірки xampp на новішу.

Завдання на лабораторну роботу

1. Встановіть локальний веб-сервер за допомогою XAMPP, LAMP або MAMP (залежно від вашої операційної системи)
2. Дослідіть файли конфігурації сервера apache.conf та php.ini. Ознайомтесь із змінними в ньому.
3. Відредагуйте файл php.ini, щоб дозволити завантаження до 10*(# порядковий в списку) мегабайт.

Контрольні запитання

1. Які програмні продукти входять до пакету XAMPP?
2. Які файли налаштувань існують у веб-сервера Apache?
3. Які файли відповідають за налаштування інтерпретатора PHP та СКБД MySQL?

ЛАБОРАТОРНА РОБОТА № 3

Робота з основними алгоритмічними структурами

Мета роботи: Навчитися використовувати умови та цикли для вирішення алгоритмічних задач мовою PHP

Теоретичні відомості

Як правило, вирази в php програмі виконуються в порядку їх написання у вихідному коді. Для зміни цього порядку потрібно використовувати умовні конструкції. Така необхідність виникає перед програмістом php практично в будь-якій програмі, коли при виконанні певної умови повинен виконуватися один блок скрипта, а при виконанні іншої умови – інший. Таким чином, PHP сценарії – дуже гнучкі і можуть залежати від безлічі умов.

Основою роботи всіх конструкцій є перевірка умов на істинність або хибність. Залежно від результату такої перевірки інтерпретується той чи інший блок php скрипта. Давайте розглянемо наступний приклад:

$color=«Червоний»

Якщо змінна $color буде мати значення «Червоний», то вираз – істинний, в іншому ж випадку він – помилковий. Результатом такого виразу можуть бути TRUE або FALSE відповідно. У мові PHP є 2 основні умовні конструкції: if і switch. Хоча у них і однакові завдання, кожна використовується зазвичай в окремих випадках.

Конструкція if

У мові PHP є кілька способів запису конструкції if. У самому простому if перевіряє хибність або істинність певної умови і залежно від результату перевірки виконує або не виконує групу виразів, розміщених у фігурних дужках.

У даному випадку конструкція if буде мати наступний формат запису:

```
<?php
if(якась умова){
Блок виразів
}
?>
```

Код в фігурних дужках буде виконуватися, якщо умова істинна. В іншому випадку блок виразів буде просто пропущений.

Розглянемо приклад:

```
<?php
$b=5;
if($b<10){
echo $b . «менша десяти»;
}
?>
```

У цьому прикладі змінній $b присвоюється значення 5. Потім виконується порівняння 5<10. Як ми бачимо, воно - істинне, тому код у фігурних дужках буде виконуватися. У результаті на екран буде виведена фраза 5 менше десяти. Наведений вище спосіб запису може бути трохи ускладнений за допомогою команди else. Синтаксис розширеного формату наступний:

```
<?php
if(якась умова){
Блок виразів
}
else{
Інший  блок виразів
}
?>
```

Якщо умови в if – істинні , тоді буде виконуватися перший блок виразів, якщо помилкові – другий блок.

На цьому можливості if не закінчуються. У скрипт можна додавати будь-яку кількість додаткових перевірок. Для цього використовується команда elseif. Тоді спосіб запису повної форми if може бути, наприклад, таким:

```
<?php
if(якась умова){
Блок виразів 1
}
elseif{
Альтернативний блок виразів 2
}
else{
Альтернативний блок виразів 3
}
?>
```

В теорії кількість блоків elseif не обмежується, проте не потрібно захоплюватися їх створенням.

Цикл в PHP – це керуюча конструкція, яка покликана виконувати блок коду кілька разів. Це означає, що вам не потрібно багато разів копіювати і вставляти код в файл, достатньо тільки використовувати правильну інструкцію циклу.

Тепер давайте подивимося, які є оператори циклу в PHP:

цикл while

цикл do ... while

цикл for

цикл foreach

1. PHP цикл while

Ймовірно, це найпростіший цикл в PHP. Його синтаксис досить простий:

while (умова) { // блок коду }

Блок коду буде виконуватися до тих пір, поки умова буде істинною. Реалізація нашого першого прикладу з циклом while буде виглядати так:

```
<?php
$i = 1;
while ($i <= 7) {
  echo " $i ";
  $i++;
}
?>
```

Результат буде точно таким же, як і в попередніх прикладах. Однак, якщо ви встановите для змінної ітератора $i значення 10 ($i = 10), тоді нічого не буде виводитися, оскільки умова не буде виконуватися. А якщо ж ви забудете додати ітератор автоінкременту (збільшення на одиницю після кожної ітерації), це призведе до нескінченного виконання циклу, оскільки умова ніколи не зміниться, і вона завжди буде істинною. Будьте уважні в таких випадках.

2. PHP цикл do while

Цей варіант циклу дуже схожий на попередній цикл while, але він має одну важливу відмінність. З циклом do while блок коду буде виконаний хоча б один раз. Це пов’язано з тим, що в разі циклу do while PHP перевіряє умову тільки після першої ітерації. Це добре видно з синтаксису:

```
<?php
do {
```

```
  //блок коду
} while (умова)
?>
```

А окрім цього даний цикл не сильно відрізняється від циклу while. Нижче приклад того, як використовувати цикл do while:

```
<?php
$i = 1;
do {
  echo " $i ";
  $i++;
} while ($i <= 7)
?>
```

3. PHP цикл for

Цикл for – це найскладніший цикл в PHP, але він також і найбільш часто використовується. У разі використання циклу for ви виконуєте ініціалізацію змінної циклу, робите перевірку умови, а потім оновлюєте змінну циклу і все в одному рядку. Синтаксис циклу for наступний:

```
<?php
for ( вираз1; умова; вираз2 ) { // блок коду }
?>
```

Цей синтаксис вимагає трохи більше пояснення.

вираз1 містить частину ініціалізації циклу. Тут ви можете встановити змінну циклу, наприклад, так $i = 1.

умова – ця частина містить перевірку певної умови. Ви можете написати тут умову, і вона буде перевірятися перед кожною ітерацією. Це означає, що може трапитися так, що блок коду взагалі не буде виконаний.

У коді вираз2 вказується інформація для оновлення змінної циклу.

Приклад використання циклу for виглядає так:

```
<?php
for($i = 1; $i <= 7; $i++) {
  echo " $i ";
}
?>
```

Кожен елемент може бути порожнім або містити кілька виразів, які розділяються комами. Використовуючи цей підхід, наведений вище код можна зробити ще менше. Однак підтримувати такий код буде трохи складніше:

```
<?php
for( $i = 1, $a = 0; $i <= 7; print " $i ", $i++);
?>
```

4. PHP цикл foreach

Остання структура циклу в PHP – це foreach. Це спеціальний цикл, оскільки його можна використовувати тільки для масивів. Призначення циклу foreach – виконати ітерацію по кожному елементу масиву. Якщо ви спробуєте використовувати його зі звичайною змінною, ви отримаєте помилку.

Синтаксис циклу foreach виглядає так:

```
<?php
foreach (array as $value) { // блок коду }
?>
```

Він означає, що при кожній ітерації фактичне значення масиву буде копіюватися в змінну $value, і ви можете використовувати це значення в блоці коду. Отже, розглянемо приклад коду для даного циклу:

```
<?php
// масив даних
$firstList = array(1, 2, 3, 4, 5, 6, 7);
// перебираємо кожен елемент масиву циклом foreach
foreach ($firstList as $value) {
  echo " $value ";
}
?>
```

Існує альтернативний синтаксис циклу foreach для обробки асоціативних масивів. Ви можете використовувати його, якщо хочете знати не тільки фактичне значення елемента, але й ключ. Синтаксис циклу foreach для асоціативних масивів:

```
<?php
foreach (array as $key => $value) {
  // блок коду
}
?>
```

В цьому випадку ви можете використовувати обидві інформації (ключ і значення) в своєму блоці коду наступним чином:

```
<?php
// масив даних
$firstList = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e'
=> 5, 'f' => 6, 'g' => 7);
// перебираємо кожен елемент масиву циклом foreach
foreach ($firstList as $key => $value) {
  echo " $key-$value ";
}
?>
```

Завдання на лабораторну роботу

1. Згідно Таблиці 1 в Додатках та вашого варіанту в колонці 1, обчисліть значення функції 2 в залежності від параметру 3, що приймають значення, розташовані в колонці 4, при значенні параметру a з колонки 5 та b з колонки 6.

2. Згідно Таблиці 2 в Додатках та вашого варіанту з колонки 1, виведіть значення функції 2, при початкових значеннях a зі стовпця 3 та b зі стовпця 4, проітерувавши параметр 5 у значеннях від 6 до 7 з кроком 8, або певну кількість разів 9.

Контрольні запитання

1. Опишіть структуру умов в мові PHP
2. Яка різниця між циклами while..do та do..while?
3. Як працює команда break всередині циклу?

ЛАБОРАТОРНА РОБОТА № 4

Функції. Робота з функціями

Мета роботи: Навчитися оголошувати та викликати функції в PHP

Теоретичні відомості

Функція – це блок коду, який написавши один раз, потім можна багаторазово використовувати в своєму проекті. Замість того, щоб постійно копіювати один і той же код, вам досить буде тільки використовувати ім’я функції. Все, що вам потрібно зробити, це написати код один раз у визначенні функції, а потім викликати функцію всюди, де вам це потрібно.

У PHP є маса вбудованих (готових) функцій, але ви можете створювати свої власні функції. Головне – знати, як це правильно робити, і в цій статті ми розглянемо найважливіші основи. Іноді, правда, не варто винаходити велосипед, а достатнім буде використовувати вже готові функції в PHP.

Оголошення функції – це іншими словами її створення. Коли ви хочете створити нову функцію, вам потрібно використовувати ключове слово function. Воно скаже PHP, що ви хочете створити функцію. Після цього слова йде назва (ім’я) функції, після якої в фігурних дужках пишеться сам код функції.

При оголошенні функції вам потрібно використовувати правильне ім’я для неї. Це схоже на визначення нової змінної. Допустиме ім’я функції може починатися з літери або нижнього підкреслення. Однак назва функції не може починатися з цифри (хоча вони можуть з’являтися в будь-якому іншому місці імені функції) або спеціального символу. Імена функцій не чутливі до регістру.

Ім’я повинно однозначно ідентифікувати функцію, тобто, мати описову назву, а не бути набором символів. Не можна використовувати в якості назви функції одне з існуючих імен функцій PHP.

Після імені функції йдуть дужки (), в яких вказуються аргументи. Це та інформація, з якою надалі буде працювати функція.

В якості аргументів (або параметрів) функції можуть виступати певні значення, змінні, інші функції і т.д. Ваша функція може мати стільки параметрів, скільки ви хочете або скільки потрібно.

Важливо знати про параметри функції:

Параметри повинні відповідати правилам іменування змінних. Іншими словами, використовуйте знак долара $, якщо це не функція, і тільки дозволені символи.

Якщо в списку параметрів є більше однієї змінної, ви повинні розділити їх комами.

Ви можете використовувати ці аргументи в будь-якому місці коду функції, але вони застосовуються тільки всередині блоку коду функції. Ви не можете отримати доступ до переданих змінним параметрів в іншому місці програмного коду.

Після імені функції і круглих дужок з параметрами (або без них) слідують фігурні дужки {}. А всередині цих дужок знаходиться безпосередньо сам код функції. Відкриваюча фігурна дужка { вказує, що це початок коду функції. Закриваюча фігурна дужка } вказує на кінець коду функції. Будь-які змінні, які ви визначаєте всередині коду функції, застосовуються тільки до коду функції. Ви не можете використовувати змінні функції в коді PHP поза визначенням функції.

Базовий формат для визначення функції виглядає наступним чином:

```
function ім’яФункції(параметри функції) {
 код функції
 return значення;
}
```

Приклад невеликої функції, яка повертає простий текст:

```
<?php
 function helloWorld() {
 echo "Привіт, світ! Це моя перша PHP функція!";
}
?>
```

Щоб викликати цю функцію, використовується наступна конструкція:

```
<?php
 helloWorld(); // результат: Привіт, світ! Це моя перша PHP функція!
?>
```

Завдання на лабораторну роботу

1. Оформіть дві задачі вашого варіанту з лабораторної роботи №3 як функції, що залежать від аргументів a або a та b.

Контрольні запитання

1. Порядок оголошення функції
2. Порядок виклику функції
3. Правила найменування функцій

ЛАБОРАТОРНА РОБОТА № 5

Об’єктно-орієнтоване програмування в PHP

Мета роботи: набути навиків реалізації об’єктно-орієнтованого підходу мовою PHP.

Теоретичні відомості

Об’єктно-орієнтоване програмування – це стиль програмування, який покликаний змусити вас думати про програмування як про реальний світ. В ООП ви працюєте з об’єктами, а не тільки з даними. Коли ви хочете створити новий об’єкт, ви робите це за допомогою класів.

Класи – це фундаментальна частина ООП. Думайте про клас, як про якийсь проект чи прототип. Він описує, яким буде об’єкт і що він може робити, містить доступні властивості і методи. Простим прикладом класу може бути будівля. Клас будівлі визначає особливості однотипної будівлі і те, як вона повинна функціонувати.

Використовуючи клас будівлі, ви можете створити нову бетонну будівлю, і вона буде новим об’єктом. В ООП цей об’єкт називається екземпляром класу. Класи і екземпляри можна порівняти з породами собак: собака – це клас, а кожна порода є екземпляром цього класу.

Давайте подивимося на класи. У PHP клас може містити змінні-члени. Ці змінні називаються властивостями. Вони використовуються для визначення характеристик об’єкта. Коли ви хочете визначити поведінку, ви використовуєте методи. Методи – це функції, по-іншому. Функція, оголошена всередині об’єкта, є методом. Інша назва, але це працює однаково.

Кожен клас повинен починатися з ключового слова class. Після цього йде ім’я класу (назва). Допустиме ім’я класу має починатися з літери або нижнього підкреслення, після чого може йти будь-яка кількість букв, цифр

або підкреслень. Після назви класу йдуть фігурні дужки {}, всередині яких містяться властивості і методи, доступні в цьому класі.

Приклад класу:

```
<?php
class nameOfClass {
  // тут властивості і методи класу
}
```

Приклад класу Dog:

```
<?php
class Dog {
 public $poroda; // властивості класу
 public $vik;
 public $mastb;
 public $name;
 // нижче метод (функція) класу
 public function getInfo() {
   echo "Порода собаки на ім'я $this->name – це $this->poroda. Вік собаки – $this->vik, а масть – $this->mastb.";
 }
}
```

Примітка: ключове слово public перед усіма властивостями і методом getInfo() є специфікатором видимості. Це говорить про те, що властивість або метод можуть бути доступні з будь-якого місця в коді. Інші специфікатори видимості: private і protected. Про них ми поговоримо трохи пізніше.

Коли ви створюєте новий об’єкт класу або його примірник, це називається «створенням екземпляру». Щоб створити екземпляр об’єкта класу, ви повинні використовувати ключове слово new. Якщо ви хочете отримати доступ до властивостей і методів об’єкта або примірника, ви повинні використовувати конструкцію стрілки ->. Якщо ви хочете привласнити значення властивості, використовуйте оператор присвоювання =, як і в випадку зі змінною.

Давайте продемонструємо це на прикладі з собакою. Спочатку ми створили новий клас під назвою Dog. Цей клас має декілька властивостей, таких як «vik», «poroda», «mastb», «name» і один метод «getInfo». Далі ми створюємо новий екземпляр цього класу під назвою $DogVivcharka. Не забувайте про правила іменування змінних.

Після цього ми призначаємо конкретні значення всіх властивостей, які доступні в класі Dog, і новому примірнику $DogVivcharka. І, нарешті, ми будемо викликати метод getInfo() в $DogVivcharka, щоб отримати інформацію про цей новий екземпляр.

```
<?php
class Dog {
 public $poroda;
 public $vik;
 public $mastb;
 public $name;
 public function getInfo() {
     echo "Порода собаки на ім'я $this->name – це $this->poroda. Вік собаки – $this->vik, а масть – $this->mastb.";
 }
```

```
    }
    // створення нового екземпляра класу Dog
    $DogVivcharka = new Dog();
    // присвоєння нових значень властивостям об'єкта або примірника $DogVivcharka
    $DogVivcharka->poroda = "вівчарка";
    $DogVivcharka->vik = "3 роки";
    $DogVivcharka->mastb = "біла";
    $DogVivcharka->name = "Рекс";
    // викликаємо метод getInfo()
    $DogVivcharka->getInfo();
    // Результат: Порода собаки на ім'я Рекс – це вівчарка. Вік собаки – 3 роки, а масть – біла.
    ?>
```

Ви, мабуть, помітили, що ми використовували $this в методі getInfo() для доступу до всіх властивостей. $this є псевдозмінною. Це посилання на об’єкт або клас, з яким ви працюєте в даний момент. Ви можете «перекласти» $this в «цього об’єкта або класу». Іншими словами, $this->name – це все одно, що сказати «властивість name цього об’єкта або класу».

Класи PHP можуть успадковувати властивості і методи іншого класу. Коли один клас успадковує властивості або методи іншого, він називається підкласом. Клас, від якого успадковується підклас, називається батьківським класом. Коли ви хочете, щоб один клас успадковував властивості або методи іншого, вам потрібно використовувати ключове слово extends.

Давайте розглянемо простий приклад, щоб продемонструвати, як працює успадкування. Спочатку ми створимо клас Lyudina. Цей клас буде містити ряд відкритих властивостей і один метод. Потім ми будемо

використовувати extends для створення двох підкласів під назвою Cholovik і Zhinka.

```
<?php
// батьківський клас
class Lyudina {
 public $name;
 public $age;
 public $isHuman = true;
 public function skazhiHello() {
  echo "Привіт, мене звати $this->name.";
 }
}
// підклас Cholovik, який успадковує батьківський клас Lyudina
// успадкування властивостей $name, $age, $isHuman та skazhiHello
// додавання властивості $pol
class Cholovik extends Lyudina {
  public $pol = "чоловічий";
}
// підклас Zhinka, який успадковує батьківський клас Lyudina
// успадкування властивостей $name, $age, $isHuman і skazhiHello
// додавання властивості $pol
class Zhinka extends Lyudina {
  public $pol = "жіночий";
}
// новий екземпляр підкласу Cholovik
$ivan = new Cholovik();
```

```
// призначення нових значень доступним властивостями
$ivan->name = "Іван";
$ivan->age = "35";
// виклик методу skazhiHello()
$ivan->skazhiHello(); // результат: Привіт, мене звати Іван
// новий екземпляр підкласу Zhinka
$julya = new Zhinka();
// призначення нових значень доступним властивостями
$julya->name = "Юлія";
$julya->age = "25";
// виклик методу skazhiHello()
$julya->skazhiHello(); // результат: Привіт, мене звати Юлія
?>
```

Як ви вже знаєте, PHP включає в себе щось, що називається специфікатором видимості або типом видимості. Цей специфікатор визначає, як і звідки можна отримати доступ до конкретної властивості і методу. До цих пір ми використовували відкритий (public). Це ключове слово вказує, що властивість або метод, з яким ви працюєте, доступні з будь-якого місця. Крім public, є два інших специфікатори: protected і private. Специфікатор protected робить властивості і методи доступними тільки всередині самого класу шляхом успадкування, а також для батьківських класів. Специфікатор private робить властивості і методи доступними тільки для класу, який їх визначає. Пам’ятайте, що в PHP кожна властивість класу повинна мати тип видимості.

Для методів це працює трохи не так. Ви можете оголосити метод без будь-якого явного типу видимості. Коли ви зробите це, PHP автоматично оголосить цей метод як відкритий (public).

Завдання на лабораторну роботу

На основі об’єктно-орієнтованого підходу, розробити програму, базові та похідні класи, та опрацювання наборів даних (з відображенням на екран) згідно варіанту, поданого в таблиці 3 додатків.

Підготувати набори вхідних даних, що включають значення аргументів та символьне позначення операції чи типу для тестування роботи програми, зокрема для: value - масив чисел і позначень системи їх представлення, shape - масив фігур і позначень операцій розрахунку, operation - масив матриць і позначень виконуваних операцій.

Передбачити спадкування від базового класу по класу на кожен елемент завдання, зокрема для базового класу value похідними є класи одиниць виміру (наприклад: метр, ярд тощо), shape похідними є класи фігур (наприклад: ромб, коло тощо), operation похідними є класи операцій (наприклад: перестановка рядків, обнулення тощо). Для роботи з класами використати абстрактні функції.

Контрольні запитання

1. Оголошення класів та конструктори в PHP.
2. Наслідування в PHP
3. Модифікатори доступу в PHP

ЛАБОРАТОРНА РОБОТА № 6

Робота з файлами в PHP

Мета роботи: навчитися використовувати файли для вводу та виводу інформації та результатів обчислень у мові PHP.

Теоретичні відомості

Мова програмування PHP має ряд функцій, які ви можете використовувати для створення, читання, завантаження і редагування файлів. Коли вам потрібно відкрити файл, використовуйте функцію fopen() з ім'ям файлу в якості параметра. Цю функцію також можна використовувати і для створення нового файлу. Якщо вказаного в параметрі файлу не існує, тоді PHP створить його.

У функції відкриття файлу fopen() є 8 режимів, і для використання одного з них, ви передаєте його код в якості другого параметра функції. Цими режимами є r, w, a, x, r+, w+, a+ та x+. Нижче наведені короткі описи для кожного із цих режимів.

Якщо ви хочете записати дані в файл, тоді використовуйте функцію fwrite(). Ця функція має два параметри. Перший параметр – це файл для запису (його назва). Другий параметр – це вміст (наприклад, рядок), яке ви хочете записати в цей файл. Коли вам потрібно закрити файл, використовуєте функцію fclose(). Ця функція також повертає TRUE, якщо файл успішно закритий, або FALSE, якщо закриття не вдалося. Рекомендується завжди закривати всі відкриті файли після того, як ви закінчите з ними працювати.

```
<?php
// a: Відкриває файл тільки для запису. Це додасть новий контент до існуючих даних.
// a+: Відкриває файл для читання або запису, а також створює новий файл, якщо файл не існує.
// r: Відкриває файл тільки для читання.
// r+: Відкриває файл для читання або запису.
// x: Створює новий файл тільки для запису.
// x+: Створює новий файл для читання або запису.
// w: Відкриває файл тільки для запису. Це зітре вміст файлу. Якщо файл не існує, він буде створений.
```

```
// w+: Відкриває файл для читання або запису. Це зітре вміст файлу. Якщо файл не існує, він буде створений.
?>
```

Деякі практичні приклади операцій з файлами в PHP:

```
<?php
// Відкрити існуючий файл в режимі запису
// Пам'ятайте, що "w" стирає існуючий вміст файлу.
$exampleFile = fopen("some-file.txt", "w");
// Або створити новий файл в режимі запису, використовуючи ім'я файлу, якого ще не існує.
$newFile = fopen("some-new-file.txt", "w");
// Запис до першого (існуючого) файлу
fwrite($exampleFile, "Цей текст замінить будь-який існуючий вміст файлу.");
// Записати в другий (новий) файл, використовуючи змінну
$someText = "Це текст для другого файлу";
fwrite($newFile, $someText);
// Закрити перший (існуючий) файл
fclose($exampleFile);
// Закрити другий (новий) файл
fclose($newFile);
?>
```

Зверніть увагу на режим, який ви використовуєте. Пам’ятайте, що режим w видалить весь існуючий вміст файлу. Якщо ви хочете додати контент в файл, вам потрібно відкрити файл в режимі додавання (a).

```
<?php
// Відкрити існуючий файл в режимі додавання
$file = fopen($exampleFile, 'a');
// Додати контент до вже існуючих даних
fwrite($file, "Деякий новий додатковий текст");
// Закрити файл
```

```
fclose($file);
?>
```

Ви можете просто прочитати файл, не відкриваючи його. В цьому випадку ви можете використовувати функцію file(). Ця функція прочитає весь файл у вигляді масиву. Кожен елемент в масиві відповідає одному рядку у файлі.

```
<?php
// Читання з файлу і збереження вмісту у вигляді масиву в змінній
$fileContent = file('some-file.txt');
// Проходження циклом по масиву $fileContent і вивід кожного рядка
foreach ($fileContent as $oneLine) {
  echo $oneLine ."", "";
}
?>
```

Завдання на лабораторну роботу

На основі попередньої лабораторної роботи, додайте можливості роботи з файлами, а саме:

1) зчитуйте набори даних із файлу input.txt

2) записуйте результати роботи програми в файл output.txt

Контрольні запитання

1. Режими відкриття файлів.
2. Функції для зчитування файлів.

3. Процедура порядкового читання файлів.

ЛАБОРАТОРНА РОБОТА № 7

Робота з базами даних в PHP

Мета роботи: Навчитися додавати, видаляти та отримувати дані з бази даних MySQL засобами PHP.

Теоретичні відомості

Як ви знаєте, за допомогою PHP ви можете створювати динамічний контент сайту. Динамічний контент не обов'язково вимагає бази даних, однак в більшості випадків вона потрібна. Всі системи управління контентом (CMS), блоги або прості обробники форми використовують базу даних. У світі PHP найбільш часто використовувана база даних – MySQL. Тому далі ми зосередимося на тому, як використовувати PHP і MySQL для створення динамічного контенту.

Переконайтеся, що у вас встановлена і правильно налаштована система, що підтримує як PHP, так і MySQL.

Щоб використовувати будь-яку базу даних в PHP (або іншими мовами програмування), вам спочатку потрібно з'єднатися з сервером бази даних. Підключення до бази даних MySQL в PHP вимагає наявності певної інформації. Вам необхідно отримати наступні дані для встановлення з'єднання:

Ім'я хоста (hostname)

Ім'я користувача бази даних (username)

Ім'я бази даних (database)

Пароль (password)

Для підключення до бази даних спочатку необхідно підключитися до сервера бази даних MySQL, а на наступному кроці вибрати необхідну базу даних. Звичайно, якщо з'єднання з сервером не вдалося, то немає сенсу продовжувати сценарій, тому ми обриваємо підключення. Давайте подивимося, як це виглядає в PHP:

```
<?php
$db_handle = mysqli_connect("localhost", "username", "password");
if($db_handle === false){
        die("ПОМИЛКА: Неможливо підключитися. " . mysqli_connect_error());
}
echo "Підключення до сервера пройшло успішно! <br>";
mysqli_select_db($db_handle, "test") or die(mysqli_error());
echo "База даних була обрана!";
mysqli_close($db_handle);
?>
```

Як ви можете бачити, функція mysqli_connect() встановлює з'єднання з сервером MySQL, і в якості наступного кроку ми можемо обрати базу даних за допомогою функції mysqli_select_db().

ВАЖЛИВО! Зверніть увагу, що ми використовуємо mysqli_connect() замість mysql_connect(), оскільки останнє розширення застаріло починаючи з версії PHP 5.5.0 і було видалено в PHP 7.0.0.

Тепер прийшов час зробити щось більш цікаве. Наприклад, давайте спробуємо витягти дані з нашої таблиці. Для цього спочатку потрібно

відкрити з’єднання і вибрати відповідну базу даних, як і раніше. Наступним кроком є відправка SQL-запиту – в даному випадку оператора SELECT – в базу даних. Тепер спробуйте отримати всі записи з MySQL бази. Приклад SQL-запиту для цього:

SELECT * FROM users;

Нам потрібно надіслати цю команду на сервер і зберегти відповідь. Ми можемо зробити це, використовуючи функцію mysqli_query() наступним чином:

```
<?php
$db_handle = mysqli_connect("localhost", "username", "password");
if($db_handle === false){
        die("ПОМИЛКА: Неможливо підключитися. " . mysqli_connect_error());
}
mysqli_select_db($db_handle, "test") or die(mysqli_error());
$result = mysqli_query($db_handle, "SELECT * FROM users");
echo $result;
?>
```

Цей запит робочий, але як відобразити дані зі змінної $result? Якщо ви спробуєте вивести її за допомогою echo, ви отримаєте щось на зразок цього:

PHP Recoverable fatal error: Object of class mysqli_result could not be converted to string

Це не те, що нам потрібно. Для правильного відображення обраних даних вам потрібно зробити трохи більше. У PHP є функції, які ви можете використовувати для отримання даних з результуючого набору бази даних MySQL. Ось ці функції:

mysqli_fetch_assoc() – витягує результуючий ряд у вигляді асоціативного масиву

mysqli_fetch_row() – отримує рядки результуючої таблиці у вигляді масиву

mysqli_fetch_array() – витягує рядок результату як асоціативний масив, звичайний масив або обидва

Всі вони перетворять один запис результату в масив, і пізніше ви можете використовувати цей масив на свій розсуд. Для отримання масиву необхідно викликати одну з вищезгаданих функцій. В даному прикладі ми використовуємо асоціативну версію.

```
<?php
$row = mysqli_fetch_assoc($result);
echo "ID: " . $row['id'] . ", Ім’я:" . $row['name'] . ", Вік:" . $row['age'] . ", Місто:" . $row['city'] . "<br>";
?>
```

Однак функція повертає тільки один запис, а в наборі результатів повинно бути 5 записів. Щоб обробити всі записи, нам потрібно створити цикл. Функції вибірки повертають масив, якщо в наборі результатів є запис, і повертають false, якщо записів більше немає. Для вибірки ми будемо використовувати простий цикл while:

```
<?php
$result = mysqli_query($db_handle, "SELECT * FROM users");
while($row = mysqli_fetch_assoc($result)){
```

```
        echo "ID: " . $row['id'] . ", Ім’я:" . $row['name'] . ", Вік:" . $row['age'] . ", Місто:" . $row['city'] . "<br>";
    }
    ?>
```

Вставка даних в базу даних за допомогою PHP

У цьому розділі ми створимо код для вставки нових записів в нашу базу даних MySQL. Це досить легко. Нам також потрібно створити з’єднання з базою даних, і ми знову будемо використовувати функцію mysqli_query(). Однак в цьому випадку ми будемо використовувати її з оператором вставки в SQL. Тепер давайте спробуємо додати нового користувача в нашу таблицю. Даний SQL-запит буде виглядати наступним чином:

INSERT INTO users (name,age,city) VALUES ('Danilo',44,'Dnipro');

Тепер збережіть цей SQL-запит у змінній і передайте його в якості параметра mysqli_query наступним чином:

```
    <?php
    $sql = "INSERT INTO users (name,age,city) VALUES ('Danilo',44,'Dnipro')";
    $result = mysqli_query($db_handle, $sql);
    ?>
```

Оновлення даних майже таке ж, як і вставка. Вам потрібно лише змінити оператор в SQL-запиті і використовувати його, як і раніше:

```
    <?php
    $sql = "UPDATE users SET age=45 WHERE name='Danilo'";
    $result = mysqli_query($db_handle, $sql);
    ?>
```

Як ви, напевно, знаєте, тут приблизно все однакове. Тільки потрібно замінити оператор в SQL-запиті:

```
    <?php
```

```
$sql = "DELETE FROM users WHERE name='Danilo'";
$result = mysqli_query($db_handle, $sql);
?>
```

Завдання на лабораторну роботу

За основу використайте код лабораторної роботи 5.

1. Створіть базу даних.

2. Створіть дві таблиці зі зв'язком між ними - одна буде відповідати за види операцій, які підтримує система, інша за вхідні і вихідні дані.

3. Створіть в PHP форму для додавання нової операції

4. Створіть в PHP форму для відображення даних операцій

5. Створіть в PHP кнопку для видалення даних операцій

Контрольні запитання

1. Підключення до бази даних засобами PHP
2. Виконання базових запитів до бази даних.
3. Отримання даних з бази.

ЛАБОРАТОРНА РОБОТА № 8

Робота з JSON та CURL в PHP

Мета роботи: Навчитися виконувати зовнішні запити з допомогою CURL та працювати з форматом JSON.

Теоретичні відомості

Бібліотека PHP `cURL` є гарним засобом для завантаження даних з іншого сервера. На основі запиту він формує HTTP-запит, який надсилає на цільовий сервер, а після завантаження містить API для (відносно) простої обробки даних.

На відміну від рідної функції file_get_contents (через яку ми також можемо робити HTTP-запити), вона пропонує набагато кращі можливості налаштування і завантажує сторінки/файли як справжній браузер.

cURL працює, надсилаючи запит на веб-сайт, і цей процес включає в себе наступні чотири частини:

1. Ініціалізація.

```
$handle = curl_init();
```

2. Налаштування параметрів. Є багато варіантів, наприклад, параметр, який визначає URL-адресу.

```
curl_setopt($handle, CURLOPT_URL, $url);
```

3. Виконання за допомогою curl_exec() .

```
$data = curl_exec($handle);
```

4. Відпустіть маркер cURL.

```
curl_close($handle);
```

Друга частина є найцікавішою, тому що вона дозволяє нам визначити, як працює cURL дуже точно, використовуючи безліч опцій, які він пропонує.

Завдання на лабораторну роботу

За основу використайте код лабораторної роботи 7.

1. Створіть файл php, який буде повертати дані з таблиці у форматі JSON.

2. З допомогою "чистого" Javascript або jQuery реалізуйте читання з цього файлу запитом get та сформуйте таблицю.

3. З допомогою PHP виконайте запит CURL до створеного в пункті 1 файлу, та на основі відповіді, поверніть відформатовану таблицю.

Контрольні запитання

1. Функції для роботи з JSON в PHP.
2. Кроки для CURL запиту.
3. Доступні опції CURL запитів.

ЛАБОРАТОРНА РОБОТА № 9

Робота з сеансами (сесіями) в PHP

Мета роботи: Навчитися застосовувати механізми автентифікації та авторизації з допомогою сеансів.

Теоретичні відомості

Сесії і cookies призначені для зберігання відомостей про користувачів при переходах між кількома сторінками. При використанні сесій дані зберігаються у тимчасових файлах на сервері. Файли з cookies зберігаються на комп'ютері користувача, і за запитом відсилаються броузером серверу.

Використання сесій і cookies дуже зручно і виправдано в таких додатках як Інтернет-магазини, форуми, дошки оголошень, коли, по-перше, необхідно зберігати інформацію про користувачів протягом кількох станиць, а, по-друге, своєчасно надавати користувачеві нову інформацію.

Протокол HTTP є протоколом "без збереження стану". Це означає, що даний протокол не має вбудованого способу збереження стану між двома транзакціями. Коли користувач відкриває спочатку одну сторінку сайту, а потім переходить на іншу сторінку цього ж сайту, то грунтуючись тільки на

засоби, що надаються протоколом HTTP неможливо встановити, що обидва запиту відносяться до одного користувача. Необхідний метод, за допомогою якого було б відстежувати інформацію про користувача протягом одного сеансу зв'язку з Web-сайтів. Одним з таких методів є управління сеансами за допомогою призначених для цього функцій. Для нас важливо те, що сеанс по суті, являє собою групу змінних, які, на відміну від звичайних змінних, зберігаються і після завершення виконання PHP-сценарію.

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:

session_start

синтаксис:

session_start ();

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

Після ініціалізації сесії з'являється можливість зберігати інформацію в су-перглобальном масиві $ _SESSION. Нехай є файл index.php у якому в масив $ _SESSION зберігається змінна і масив.

```
<? php
   / / Ініціюємо сесію
   session_start ();
   / / Розміщуємо значення в сесію
   $ _SESSION ['Name'] = "value";
```

```
    / / Розміщуємо 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 суперглобального масиву $ _SESSION. У наступному лістингу наводиться вміст сторінки other.php, де витягуються дані, раніше поміщені на сторінці index.php.

Після завершення роботи з сесією спочатку потрібно разрегістріровать всі змінні сесії, а потім викликати функцію unset ():
синтаксис:

```
unset ($ _SESSION ["username"]);
```

Завдання на лабораторну роботу

За основу використайте код та базу даних лабораторної роботи №8.

1. Створіть в базі даних таблицю users з трьома полями - id, login та password.

2. Додайте в таблицю запис, для запису паролю використайте формат хешу md5.

3. Створіть файл login.php, куди додайте форму входу та обробник даної форми. Обробник приймає логін та пароль, і, якщо вони правильні, записує ім'я користувача в масив сесії, перенаправляючи його на основну сторінку, реалізовану в лабораторній роботі №7. Якщо логін або пароль неправильний,

сторінка логіну обновляється з відповідним повідомленням про неправильний логін або пароль.

4. Відредагуйте основний файл вашої програми у лабораторній роботі №7 так, щоб якщо масив сесії не містить імені користувача, з даного файлу відбувався редірект на форму входу.

Контрольні запитання

1. Механізм автентифікації
2. Хешування паролів
3. Збереження даних у псевдомасиві $_SESSION

ЛАБОРАТОРНА РОБОТА №10

Робота з SMTP

Мета роботи: Навчитися звертатися до поштових серверів з допомогою SMTP.

Теоретичні відомості

Для надсилання листів через SMTP до PHP будемо використовувати бібліотеку PHPMailer. Для цього її необхідно завантажити:
github.com/PHPMailer/PHPMailer
Розпакувати архів у папку з розширеннями (додатками).
Або встановити за допомогою composer:

composer require phpmailer / phpmailer

І приступаємо до використання.

Мінімальний код для надсилання листів за допомогою SMTP

Використання бібліотеки досить просте, наприклад відправимо лист з яндекс пошти через SMTP:

```
<? php

require 'PHPMailer/PHPMailerAutoload.php' ;

$mail = новий PHPMailer ;

$mail -> isSMTP ();
$mail -> Host = 'smtp.yandex.ru' ;
$mail -> SMTPAuth = true ;
$mail -> Username = 'robot@devreadwrite.com' ; //Логін
$mail -> Password = '*******' ; //Пароль
$mail -> SMTPSecure = 'ssl' ;
$ mail -> Port = 465 ;

$mail -> setFrom ( 'robot@devreadwrite.com' , 'Robot' );
$ mail -> isHTML ( true );

$mail -> Subject = 'Тема листа' ;
$mail -> Body = '<b>HTML</b> версія листа' ;
$mail -> AltBody = 'Текстова версія листа, без HTML тегів (для клієнтів, що не підтримують HTML)' ;

//Відправка повідомлення
if (! $mail -> send ()) {
    echo 'Помилка при надсиланні. Помилка: ' . $mail -> ErrorInfo ;
} else {
    echo 'Повідомлення успішно надіслано' ;
}
```

Цього мінімального коду вистачить для 90% завдань надсилання листів.

Завдання на лабораторну роботу

1. Створіть тестовий аккаунт на ukr.net або іншій пошті (небажано використовувати свій "живий аккаунт", оскільки є ризик що ви забудете видалити свій пароль зі звіту або поділитеся ним з одногрупниками!

2. Створіть файл feedback php, що містить форму зворотнього зв'язку. Додайте туди поля.

3. Підключіть пакет PHPMailer та напишіть обробник форми зворотнього зв'язку з відправкою мейлу через SMTP, використовуючи авторизаційні дані.

4. Обмежте доступ до файлу feedback.php незалогіненим користувачам по прикладу попередньої лабораторної роботи

Контрольні запитання

1. Використання PHPMailer.
2. Параметри підключення до SMTP сервера.
3. Особливості заголовків листів.

ЛАБОРАТОРНА РОБОТА № 11

Робота з моделлю MVC

Мета роботи: Навчитися формувати проекти PHP застосовуючи модель MVC

Теоретичні відомості

MVC - це програмна структура, яка зазвичай використовується в галузі як основа для створення ефективних веб-додатків. Це архітектурна схема, яка складається з трьох компонентів Model, View та Controller, що ефективно відокремлює Business Logic від користувальницького інтерфейсу програми.

MVC складається з трьох компонентів

Модель

Вид

Контролер

Модель: Простими словами, Модель містить дані про програму. Тут вказується вся інформація, яка має бути важливою для відображення або відображення, її вимоги щодо доступу та інших перевірок.

Перегляд: Перегляд відображає дані в компоненті Модель. Будь-яка відповідь від користувача також розпізнається та надсилається до компонента Controller.

Контролер: Контролер відповідає за надання даних, присутніх у Моделі, компоненту «Вид» та інтерпретацію відповідей користувачів, які розпізнаються компонентом «Перегляд».

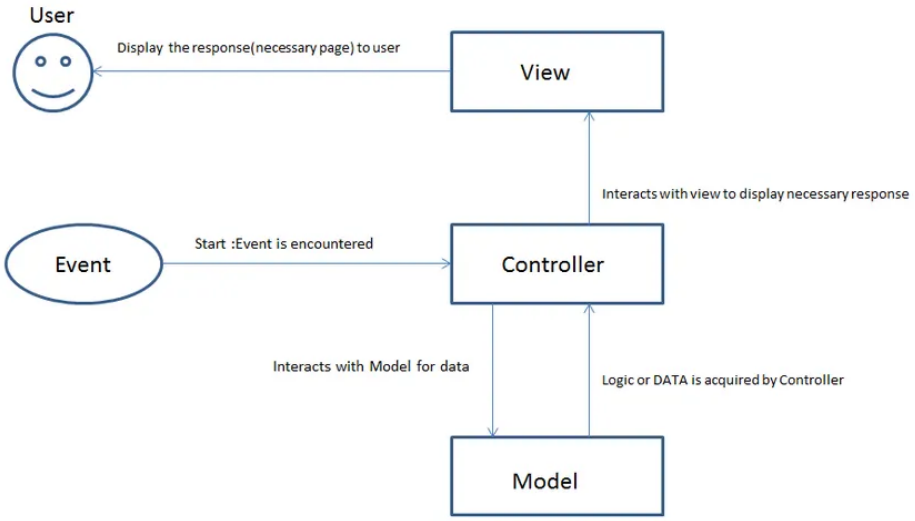

Рисунок 1. Модель MVC.

Завдання на лабораторну роботу

1. Створіть новий проект. Частково ви зможете скористатися кодом із попередніх лабораторних.

2. Створіть файл конфігурації, куди внесіть параметри підключення до бази даних.

3. Створіть пустий файл index.php та три каталоги - views, controllers та models.

4. Реалізуйте дві моделі - перша буде містити структуру класів із попередніх лабораторних. Друга - відповідати за підключення до бази даних, отримання, додавання та видалення даних з бази.

5. Винесіть форму додавання даних у окремий файл виду. Аналогічно винесіть в окремий файл табличку відображення даних. При бажанні, додайте вид, що реалізовуватиме меню навігації.

6. Додайте контроллер, який, в залежності від даних про операцію, отриманих із адресного рядка, буде відображати дані з бази, відображати форму додавання даних, додавати дані, відправлені з форми, видаляти дані по запиту, використовуючи вищезазначені моделі та види.

7. Викличте розроблений контролер із файлу index.php.

Контрольні запитання

1. Структура MVC
2. Взаємодія файлів в проекті PHP
3. Розділення функціоналу та вигляду додатку.

ЛАБОРАТОРНА РОБОТА № 12

Робота з PHPUnit

Мета роботи: Навчитися створювати базові тести в PHPUnit

Теоретичні відомості

Наразі PHPUnit є найпопулярнішим фреймворком модульного тестування PHP. Окрім таких функцій, як знущання над об'єктами, він забезпечує аналіз покриття коду, журналювання та безліч інших потужних функцій.

Завантажити: PHPUnit поширюється у файлі PHAR (PHP Archive), завантажте його тут.

Додайте його до системного $PATH: після завантаження файлу PHAR переконайтеся, що він виконуваний, і додайте його до вашої системної $PATH. Щоб ви могли запускати його будь-де.

Припустимо, що ви працюєте на машині OSX. Цей процес можна виконати у вашому терміналі за допомогою наведених нижче команд:

```
wget https://phar.phpunit.de/phpunit.phar
chmod +x phpunit.phar
sudo mv phpunit.phar /usr/local/bin/phpunit
```

Якщо ви зробили все правильно, ви зможете побачити версію PHPUnit, ввівши команду нижче у своєму терміналі.

```
phpunit --version
```

Час створити свій перший модульний тест! Перед цим нам потрібен клас для перевірки. Давайте створимо дуже простий клас під назвою Calculator . І написати для нього контрольну роботу.

Створіть файл із назвою «Calculator.php» і скопіюйте наведений нижче код у файл. Цей клас Calculator має лише функцію Add . :

```
?

<?php
class Calculator
{

    public function add($a, $b)
    {
        return $a + $b;
    }

}
```

Створіть тестовий файл "CalculatorTest.php" і скопіюйте наведений нижче код у файл. Ми детально пояснимо кожну функцію.

```
<?php
require 'Calculator.php';

class CalculatorTests extends PHPUnit_Framework_TestCase
{
    private $calculator;

    protected function setUp()
    {
        $this->calculator = new Calculator();
    }

    protected function tearDown()
    {
        $this->calculator = NULL;
```

```
    }

    public function testAdd()
    {
        $result = $this->calculator->add(1, 2);
        $this->assertEquals(3, $result);
    }

}
```

Рядок 2: включити файл класу Calculator.php . Це клас, з яким ми будемо тестувати, тож обов’язково включите його.

Рядок 8: setUp() викликається перед кожним тестом. Майте на увазі, що він запускається перед кожним тестом, тобто якщо у вас є інша тестова функція. Він також запустить setUp() перед ним.

Рядок 13: подібно до setUp() , tearDown() викликається після завершення кожного тесту.

Рядок 18: testAdd() — тестова функція для функції add . PHPUnit розпізнає всі функції з префіксом test як тестову функцію та запустить її автоматично. Ця функція насправді дуже проста, ми спочатку викликаємо функцію Calculator.add , щоб обчислити значення 1 плюс 2. Потім ми перевіряємо, чи повертає вона правильне значення, використовуючи функцію PHPUnit assertEquals .

Остання частина роботи — запустити PHPUnit і переконатися, що він пройшов усі тести. Перейдіть до папки, де ви створили тестовий файл, і виконайте наведені нижче команди з вашого терміналу:

phpunit CalculatorTest.php

Ви повинні побачити успішне повідомлення

Завдання на лабораторну роботу

1. Створіть декілька тест-кейсів для програми, виконаної вами лабораторній роботі №5.

Контрольні запитання

1. Види тестування
2. Встановлення PHPUnit
3. Написання тестів для PHPUnit.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Уроки програмування PHP [Електронний ресурс] https://sebweo.com/category/uroki-programuvannya/
2. PHP Tutorial [Електронний ресурс] https://www.w3schools.com/php/
3. PHP CURL Tutorial [Електронний ресурс] https://phpenthusiast.com/blog/five-php-curl-examples
4. Сесії в PHP [Електронний ресурс] Режим доступу: https://wiki.cuspu.edu.ua/index.php/%D0%A1%D0%B5%D1%81%D1%96%D1%97_%D0%B2_PHP
5. PHP Unit - Starting Tutorial. [Електронний ресурс] https://startutorial.com/view/phpunit-beginner-part-1-get-started

ДОДАТКИ

Таблиця 1 - Варіанти завдання 1 на лабораторну роботу 3

1	2	3	4	5	6
1	$y=\begin{cases} e^{\lvert x+a\rvert}\cdot\sin(x), & x = a \\ (x-a)^2\cos(x)^2, & a < x < a^2 \end{cases}$	x	1.62 1.41	1.62	-1.25
2	$z=\begin{cases} \ln^2(y+1), & y < a \\ \sqrt{y}+a^{b-1.5}, & a \le y \le b \end{cases}$	y	2.86 3.98	3.463	4.215
3	$c=\begin{cases} g^{1.4}+b\sin^2(a/2) & b \le g < a \\ x(g^{1.4}+b) & g < b \end{cases}$	g	4.751 1.17	6.435	1.87
4	$d=\begin{cases} a\sin(b\alpha+\pi/10) & \alpha = b \\ b\cos^2(\alpha^2) & b < \alpha \le a \end{cases}$	α	1.321 2.65	3.65	1.321
5	$e=\begin{cases} arctg\left(\frac{3}{2x+1}+b\right) & x \ge 2a \\ b\arcsin\left(\frac{a+b}{\ln(x)}\right) & b \le x < a \end{cases}$	x	11.68 3.86	4.63	0.68
6	$f=\begin{cases} \frac{l+b}{2+\sin^2(a)} & a \le l < 2a \\ \sqrt{a+1}\cdot\sin(a+0.1) & l \ge 4a \end{cases}$	l	2.38 7.6	1.361	-

7	$h=\begin{cases}\lg^2(a+b)\cdot b & x \le b \\ \ln(b^2+a/b) & a \ge x > b\end{cases}$	x	4.26 7.29	8.3	6.8
8	$g=\begin{cases}2\cos^2(h+a)^2 & h < a \\ \ln(b+3\sin(h^2)) & a < h < b+a\end{cases}$	h	0.21 1.65	0.261	1.658
9	$o=\begin{cases}\ln^2(a+q) & q > a+b \\ \frac{b^2}{2}+a\sin^3(q) & b < q \le a\end{cases}$	q	9.477 4.35	4.652	3.825
10	$p=\begin{cases}e^{\frac{x}{2}+\sqrt{x}} & a < x \\ \ln^2(b)+tg\left(x+\frac{\pi}{10}\right) & b > x\end{cases}$	x	3.2 6.856	1.5	10.638
11	$r=\begin{cases}\sqrt[3]{\ln(o)}+o & a > o \ge b \\ o+a\sin(o^2+\pi/12) & o > a\end{cases}$	o	1.651 0.581	12.83	0.863
12	$s=\begin{cases}\frac{a^2-p}{b\log_2(b+p)} & 2.5 \le p < a \\ \frac{3.51\sqrt{p}}{a-1} & 4 < p \le b\end{cases}$	p	3.271 4.956	3.651	5.82
13	$t=\begin{cases}\frac{a^2+\sqrt[3]{r}}{\ln^2(r+b)} & r \ge a \\ \frac{\sqrt{ae^r}}{r} & a/2 > r \ge a/3\end{cases}$	r	8.269 2.892	6.347	21.4

14	$i = \begin{cases} \dfrac{\sqrt{b} + \cos^2(a^2 s)}{\ln\lvert s - 5\rvert} & 1.5 > s \geq b - a \\ ab + tg(s/3 + \pi/12) & s > b \end{cases}$	s	1.326 3.906	0.854	3.625
15	$k = \begin{cases} \sqrt[3]{b \sin(t^2 + \pi/45)} & t < a \\ \sqrt{\dfrac{a}{3}\cos\left(t + \dfrac{\pi}{12}\right)} & t > b \end{cases}$	t	0.365	1.265	3.126
16	$l = \begin{cases} ay + y\sin^2(y + b) & \dfrac{\lvert b\rvert}{2} > y > a \\ \dfrac{y^2 a}{2.5 + y} + \dfrac{by}{a} & y \geq \lvert b\rvert \end{cases}$	y	0.375 0.506	0.265	-0.883
17	$m = \begin{cases} \sqrt{\lvert a - e^{l - 0.5}\rvert b} & l \leq a \\ a\ln(b) & (a + b)/2 < l \leq b \end{cases}$	l	2.685 7.21	4.735	8.216
18	$\alpha = \begin{cases} \beta^{a+1} + e^{\beta} & a < \beta \leq 1.6a \\ \dfrac{3(\beta^2 - a)}{0.1b} & b > \beta \geq 2a \end{cases}$	β	3.656 8.350	2.876	12.393
19	$\varphi = \begin{cases} \lg^2\left(a^{\lvert c - b\rvert}\right) & b \geq c > a \\ \ln(b + c^2 a) & c < 3 \end{cases}$	c	18.106 28.318	6.208	23.575

20	$y=\begin{cases}\sqrt[3]{\lvert x-a\rvert}\cdot b & x > ab \\ \dfrac{\lvert x-a\rvert^3}{2} & b/a > x \geq b\end{cases}$	x	1.627 0.291	1.326	0.967
21	$c=\begin{cases}a^2 tg(\varphi - b/2) & \varphi > a \\ btg\left(\dfrac{\varphi+b}{2}\right) & \varphi < a-b\end{cases}$	φ	1.36 2.314	2.128	0.366
22	$d=\begin{cases}e^{\sqrt{\alpha+b}}+b\sqrt{a} & \alpha > 3b \\ e^{\sqrt{\alpha+b}}+a\ln(b) & \alpha < a\end{cases}$		8.272 3.214	16.283	2.283
23	$f=\begin{cases}a^{y+1}+be^{y-1} & a > y \geq 1.5 \\ \left(a^{y+1}+be^{y-1}\right)/\sqrt{ay} & y > 2a\end{cases}$	y	2.225 7.093	2.703	12.385
24	$y=\begin{cases}b^3+\ln^2\lvert i+\pi\rvert & i < b \\ i^3+\ln^2\lvert i+\pi\rvert & i > a\end{cases}$	i	-5.5 -2.638	-2	-8

Таблиця 2 - Варіанти завдання 2 на лабораторну роботу 3

1	2	3	4	5	6	7	8	9
1	$t=\left(\pi - x^2\right)\cdot \sin\left(\sqrt{2.1b+x\ln(a)}\right)$	21.4	1.95	x	4.6	-	1.5	8

2	$x = \sqrt{1.5a - bq}\,\frac{tg(q^2 / 5\pi)}{e^q + b}$	121.2	3.8	q	1.4	1.85	-	9
3	$d = 2^{-\sqrt[p]{\left\|p + \sqrt[4]{\left\|p - a\right\|}\right\|}e^b}$	8.3	1.43	p	-	-4.74	0.15	10
4	$t = m^a + \sqrt[3]{m + \frac{b^3 m}{\log_2(e)}}$	1.6	2.09	m	-	16	1.5	7
5	$y = \frac{x arctg(x^2 - \sin(x))}{\sqrt{x - a}}$	0.83	-	x	1.15	-	0.35	11
6	$\beta = \frac{4\sin^2\left(\alpha + \frac{\pi}{4}\right)\sqrt{btg\frac{\alpha}{2}}}{\ln(\cos(\alpha) + a)}$	3.85	1.8	alpha	-	15^0	7^0	8
7	$z = \lg\left(e^{t-a} + b^a\right)\frac{b}{2}$	1.6	14.3	t	2.75	5.0	-	9
	$\alpha = \frac{\ln\left\|b\cos(\beta + 0.6)\right\|}{\sqrt[3]{bctg(\beta)}}$	-	6.215	beta	40^0	54^0	-	7
9	$m = \frac{1 + \cos^2(y + \pi / 12)}{y + a^y}$	2.43	-	y	1.62	-	0.15	8
10	$y = \frac{\cos(\pi\alpha^2)}{\sqrt{ab + \alpha^2}}$	1.85	2.63	alpha	-3.45	-	-1.6	11

11	$f = \sqrt[3]{b+d}\,ctg\left(\frac{\pi}{d+0.15}\right)$	-	3.85	d	-2.3	0.01	-	7
12	$k = \frac{\sin(ex^2)}{\sqrt{a+bx^3}}$	2.65	1.48	x	0.75	0.012	-	9
13	$z = \frac{arctg(a+y)^3}{\ln^2(a)} + \frac{\pi}{6a}$	3.2	-	y	-4.8	0.1	-	7
14	$x = \frac{\lvert b - z\rvert^3}{\lg(e^{z-0.3})} + tg(z)$	-	8.45	z	-	0.24	3.2	9
15	$s = \frac{\lg^3\lvert x+a\rvert}{\ln(x^2)+a/3}$	-16.3	-	x	-16.3	-	-9.3	9
16	$o = \frac{1+d+d^4/b}{\sqrt[3]{e^d+a}}$	-8.6	3.28	d	3.6	-	4.0	14
17	$y = \frac{\sqrt{t^3+b^2}}{\sin(arctg(z+b))}$	1.42	0.85	z	1.6	0.35	-	5
8	$c = \frac{f^{a+1}+f^{a-1}}{\lvert a+f\rvert}$	3.2	-	f	-4.8	0.1	-	7
19	$y = \frac{a\sin^2(\alpha+\pi/10)}{\ln(a) - \cos(\alpha/3)}$	8.9	-	alpha	35^0	-	50^0	6

20	$\sigma = \frac{\lg^2\lvert a - z\rvert}{\sqrt{e^{a+z}}} + \frac{z}{a}$	1.2	-	z	2.6	-	3.8	8
21	$\beta = \sqrt{x - \sin(x - 0.15)}$ $b / \sin^2(x)$	-	6.15	x	-	0.2	1.65	10
22	$f = \frac{x}{x + a}\sqrt{\lvert e^x - 1.6a\rvert}$	3.26	-	x	-	0.18	3.41	7
23	$t = ab(1 - \sin(g + 5)) +$ $+\sqrt{b /} \cos^2(g)$	1.85	6.21	g	22	3.1	-	9
24	$p = \frac{\ln(af) - 1}{3\sin^2(f) + 0.6}$	6.92	-	f	0.6	-	1.4	8
25	$x = e^{\frac{x^2 - a}{\sin(x)}}\sqrt{b\frac{\ln(x + 0.6)}{x}}$	4.55	7.53	x	4.8	0.25	-	5

Таблиця 3 - варіанти завдань на лабораторну роботу №5

№	Завдання	Базовий клас
1	Перерахунок довжини: сантиметр, фут, дюйм, ярд в метр	value
2	Перерахунок маси: грам, фунт, пуд в кілограм	value

3	Перерахунок температури: град.фаренгейта, кельвін в град.цельсія	value
4	Перерахунок тиску: паскаль, бар, мм.вод.ст. в атмосфера	value
5	Перерахунок об’єму: кварта, галон, барель в літр	value
6	Перерахунок швидкості: м/с, миль/год, ярд/с в км/год	value
7	Перерахунок потужності: кінська сила, дж/с в кіловат	value
8	Перерахунок витрати: галонів/милю, літрів/милю, галонів/км в літрів/км	value
9	Перерахунок вартості валюти: фунт, євро, долар в гривню	value
10	Перерахунок тепловтрат матеріалів: бетон, цегла, дерево в пінопласт	value
11	Розрахунок площі 2D фігур: прямокутник, трикутник, коло	shape
12	Розрахунок периметру 2D фігур: трикутник, п’ятикутник, еліпс	shape
13	Розрахунок кількості обертів 2D фігур на задану довжину: ромб, коло, еліпс	shape
14	Розрахунок довжин діагоналей 2D фігур: прямокутник, трапеція, ромб	shape
15	Розрахунок центру ваги (тяжіння) 2D фігур: трикутник, трапеція, півколо	shape

16	Перевірка трикутника на: рівносторонній, рівнобедрений, прямокутний	shape
17	Перевірка чотирикутника на: квадрат, прямокутник, трапеція	shape
18	Розрахунок площі 3D фігур: призма трикутна, конус, циліндр	shape
19	Розрахунок об’єму 3D фігур: циліндр, конус, куля	shape
20	Розрахунок периметру 3D фігур: тетраедр, піраміда, призма п’ятикутна	shape
21	Матриці: перестановка рядків, колонок, обнулення рядків, колонок	operation
22	Матриці: обнулення елементів над/під головню та побічною діагоналями	operation
23	Матриці: розрахунок суми, середнього елементів вказаного рядка/колонки	operation
24	Матриці: дзеркальне відображення по вертикалі/горизонталі/обох діагоналях	operation
25	Матриці: пошук min/max елементів над/під головню та побічною діагоналями	operation
26	Матриці: розрахунок суми, середнього, добутку усіх елементів	operation
27	Матриці: знаходження суми та кількості додатних та від’ємних елементів	operation

28	Матриці: визначити кількість нульових, одиничних, парних, непарних елементів	operation
29	Матриці: додавання, віднімання, транспонування, порівнювання	operation
30	Матриці: кількість нульових/не нульових елементів над/під головню діагоналлю	operation
31	Статистичні оцінки одномірного масиву: середнє, дисперсія, медіана	operation
32	Статистичні оцінки одномірного масиву: математичне сподівання, мода, СКВ	operation