МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО

МЕТОДИЧНІ ВКАЗІВКИ
ЩОДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«WEB-ТЕХНОЛОГІЇ»
ДЛЯ СТУДЕНТІВ ДЕННОЇ ФОРМИ НАВЧАННЯ
ЗІ СПЕЦІАЛЬНОСТІ 122 – «КОМП'ЮТЕРНІ НАУКИ»
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«КОМП'ЮТЕРНІ НАУКИ»
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»
(ЧАСТИНА 1)

КРЕМЕНЧУК 2020

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Web-технології» для студентів денної форми навчання зі спеціальності 122 – «Комп’ютерні науки» за освітньо-професійною програмою «Комп’ютерні науки» освітнього ступеня «Бакалавр» (частина 1)

Укладач асист. О. О. Скриль
Рецензент к. т. н., доц. І. С. Конох

Кафедра автоматизації та інформаційних систем

Затверджено методичною радою Кременчуцького національного університету імені Михайла Остроградського
Протокол № ___ від _____ 2020 року
Голова методичної ради _____ проф. В. В. Костін

ЗМІСТ

ВСТУП

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Web-технології» містять 5 лабораторних робіт. Виконання лабораторних робіт спрямоване на набуття практичних навичок розробки Web-сайтів і сервісів з використанням PHP.

У наданих методичних вказівках описана робота HTTP-протоколу. Розглянуто синтаксис мови програмування PHP та її використання для написання серверних додатків.

Викладення матеріалу в методичних вказівках побудовано від простого до складного. Використовується велика кількість прикладів, що сприяє кращому засвоєнню матеріалу. Окрім того, у методичних вказівках надані всі відомості, необхідні для виконання лабораторних робіт, що дають змогу зрозуміти принцип створення та роботи програми.

Освоєння Web-технологій є додатковим аспектом у вивченні необхідних технологій, яка забезпечує фахівця ще одним потужним інструментом для створення багатокомплексних систем автоматизації. Подібні системи є складним комплексом для обробки інформації, планування, диспетчеризації тощо.

Web-технології є надзвичайно потужним інструментарієм для розв’язання проблемних задач автоматизації, а також надають додаткові можливості для регулювання виробничих процесів. Засвоївши навички створення Web-базованих додатків фахівець має можливість створювати та працювати зі складними комплексними системами.

Майбутній фахівець, вивчаючи Web-технології і володіючи навичками програмування, буде в змозі самостійно провести розробку необхідного алгоритму і програмно його реалізувати, вбудувавши в Web-додаток.

Вивчення Web-технологій стимулює засвоєння розуміння роботи глобальної мережі, також фахівець засвоює різні підходи розробки та методології проектування Web-додатків.

Метою вивчення навчальної дисципліни «Web-технології» є вивчення основ роботи з протоколом HTTP для повного розуміння роботи Web-додатків, також розуміння роботи клієнтських додатків, оволодіння навикчок створення спеціалізованих користувацьких інтерфейсів.

Завданням навчальної дисципліни «Web-технології» є отримання студентами теоретичних основ і практичних навичок застосування Web-технологій для розв'язання задач автоматизації.

Завдяки виконанню лабораторних робіт майбутні спеціалісти практично засвоять технологію створення серверних Web-додатків з використанням мови програмування PHP.

У результаті вивчення навчальної дисципліни студент повинен

знати:

- загальні поняття Web-технологій;
- принципи роботи сайтів та сервісів Інтернет;
- основні стандарти обміну даними між додатками і сервісами Інтернет;
- мову гіпертекстової розмітки HTML та технологію CSS;
- можливості застосування клієнтських сценаріїв мовою JavaScript для створення інтерактивних сайтів;
- основи бібліотеки jQuery;
- основи мов розроблення сценаріїв PHP;
- інструменти розробки WEB-служб;

уміти:

- застосовувати framework's і CMS;
- проектувати та створювати сайти;
- розробляти мовою програмування JavaScript клієнтські сценарії для створення інтерактивних сайтів;
- застосовувати мову програмування PHP для створення динамічних сайтів.

1 ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТІВ З ЛАБОРАТОРНИХ РОБІТ

Звіт про виконання лабораторних робіт (далі – звіт) має бути написаний студентом власноруч, розбірливим почерком, чисто й охайно, однаковим чорнилом (синіми чи фіолетовими) чи пастою на аркушах білого паперу форматом А4 (210×297 мм) або набраний за допомогою комп'ютерної техніки шрифтом Times New Roman, розміром 14 пунктів на одному боці аркуша. На одній сторінці допускається не більш ніж три виправлення, зроблені охайно та розбірливо (допускається застосування коректора).

Зміст звіту

Звіт має містити:

– назву роботи;

– тему та мету роботи;

– завдання до самостійної роботи;

– тексти написаних програм;

– висновки з роботи.

Підготовка до виконання лабораторних робіт

Підготовка до кожної лабораторної роботи проводиться студентом самостійно. Студент повинен ознайомитися з теоретичними відомостями й порядком виконання роботи. За допомогою конспекту лекцій і додаткової літератури необхідно опрацювати певні теми й відповісти на контрольні питання, наведені в лабораторній роботі.

Під час виконання роботи необхідно виконати всі завдання, які наведені у відповідному пункті, та самостійно виконати завдання згідно зі своїм варіантом. Після завершення роботи необхідно оформити звіт, зміст якого визначаються відповідним пунктом. Для захисту роботи необхідно виконати додаткове завдання, яке визначає викладач.

Лабораторна робота захищається під час індивідуальної бесіди студента з викладачем.

2 ПЕРЕЛІК ЛАБОРАТОРНИХ РОБІТ

Лабораторна робота № 1

Тема. Базові синтаксичні конструкції мови PHP

Мета: отримати практичні навички алгоритмічного розв'язання задач мовою PHP.

Короткі теоретичні відомості

1. Що таке і як працює PHP?

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 Web-сервера. PHP є однією з найпоширеніших мов, що використовуються у сфері Web-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP підтримується переважною більшістю хостинг-провайдерів. PHP – проект відкритого програмного забезпечення.

PHP працює з Web-сервером, який являє собою програмне забезпечення, яке доставляє Web-сторінки в інтернеті.

Якщо PHP встановлений, Web-сервер налаштований так, щоб очікувати, певні розширення файлів, які містять вставки мови PHP. Часто розширення .php або .phtml, але будь-яке розширення можна використовувати. Коли Web-сервер отримує запит на файл з налаштованим розширенням, він надсилає HTML-структуру так, як є, але оператори PHP обробляються програмним забезпеченням PHP, перш ніж вони надсилаються запитувачу.

Коли PHP-вирази оброблені, Web-сервер надсилає Web-браузеру лише вихідний файл або будь-яке друковане на екрані повідомлення. Висловлювання мови PHP, які не створюють жодного виводу на екран, не входять до виходу, надісланого браузеру, тому користувач не може переглядати код PHP.

PHP не інтегрований з усіма Web-серверами, але працює з багатьма популярними Web-серверами. PHP добре працює з Web-сервером Apache. PHP також працює з Microsoft Internet Information Services (IIS) та іншими.

2. Оголошення змінних

Змінна – це засіб для зберігання значення, такого як текстовий рядок «Hello World!» або чисельне значення 4. У PHP ви визначаєте змінну так:

```
$variable_name = Value;
```

Як ви бачите, будь-яка змінна починається зі знаком долара. Неможна про це забувати, інакше нічого працювати не буде.

Окрім того, імена чутливі до регістру, тому $a і $A - це дві різні змінні.

```
$text = "Hello world";
$counter = 3;
$price = 4.4;
```

3. Масиви

Насправді масив у PHP – це упорядковане відображення, яке встановлює відповідність між значенням і ключем. Цей тип оптимізований у декількох напрямках, тому ви можете використовувати його як власне масив, список (вектор), хеш-таблицю (що є реалізацією карти), словник, колекцію, стек, чергу і, можливо, щось ще. Оскільки масиву може бути інший масив PHP, можна також створювати дерева і багатовимірні масиви.

3.1 Створення масиву за допомогою array()

```
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);
$arr = array(1,2,3);
echo gettype($arr) . "\n";
echo $arr[0];
```

3.2 Створення масиву за допомогою []

```
$arr = [];
echo gettype($arr); //array
$arr = [1,2,3];
echo gettype($arr) . "\n";
```

```
echo $arr[0];
$arr = [1,"qwe",2.2];
echo gettype($arr) . "\n";
echo $arr[1];
```

4. Робота з print, printf, echo

print – виводить рядок

int print (string $arg)

print насправді не є справжньою функцією (це конструкція мови), тому укладати аргументи в дужки необов’язково. Завжди повертає 1.

```
print("Hello World!!! \n");
print "Hello World!!! \n";
$res = print "Hello World!!! \n";
//you can use res. Why? I dont know :(
print $res;
```

printf – Виводить відформатований рядок.

int printf (string $format [, mixed $args [, mixed $...]])

Повертає довжину рядка, що виводиться.

```
$number = 9;
$str = "Beijing";
printf("There are %u million bicycles in %s.",$number,$str);
```

echo – виводить однин або більше рядків. Символ кінця рядка не додається.

void echo (string $arg1 [, string $...])

Насправді echo – це не функція (це мовна конструкція), тому укладати аргументи в дужки необов’язково. echo (на відміну від інших мовних конструкцій) не поводиться як функція, тому не завжди може бути використана в контексті функції. До того ж, якщо ви хочете передати більше одного аргументу в echo, ці аргументи не можна укладати в дужки.

```
echo "Hello world \n";
echo "Some text ", "Some other text \n";
```

```
echo("echo function? \n");
//echo("", ""); error
```

echo має також коротку форму, що являє собою знак рівності, наступний безпосередньо за відкритим тегом.

```
<!-- Cannot be inside of this &lt;?php -->
<?= ‘qwe’ ?>
```

Єдина відмінність від print полягає у тому, що echo приймає список аргументів і нічого не повертає.

5. Умовні оператори

Конструкція elseif є поєднання if і else. Аналогічно else, вона розширює оператор if для виконання різних виразів, якщо початкового оператора if еквівалентна FALSE. Однак, на відміну від else, виконання альтернативного вираження відбудеться тільки тоді, коли умова оператора elseif буде рівною TRUE.

```
if ($a > $b) {
    echo "a больше, чем b";
} elseif ($a == $b) {
    echo "a равен b";
} else {
    echo "a меньше, чем b";
}
```

6. Цикли

Цикл з передумовою while

```
while (condition) {
  code to be executed;
}
$i = 5;
while(--$i > 0){
    echo $i;
}
```

Цикл з післяумовою do...while

```
do {
  code to be executed;
}
while (condition);
```

```
$i = 1;
do{
   echo $i;
} while(--$i);
```

Цикл з лічильником for

```
for (initialization; condition; increment){
  code to be executed;
}
```

```
for($i = 0; $i < 5; $i++){
   echo $i;
}
```

7. Функції

Що таке функції:

– функція є блоком операторів, які можна використовувати повторно в програмі.

– функція не виконуватиметься відразу із завантаженням сторінки.

– функція виконуватиметься за допомогою виклику функції.

Синтаксис оголошення функції:

```
function functionName() {
   code to be executed;
}
```

Інформація може передаватися функціям через аргументи. Аргумент подібний до змінної. Аргументи задаються після імені функції, у дужках. Ви можете задати стільки аргументів, скільки хочете, просто відділіть їх комою.

```
function sayHello($helloText){
```

```
    echo $helloText;
}
```

Завдання до самостійної роботи

Виконайте завдання згідно з варіантом.

1. Розробити програму розв’язання системи лінійних алгебраїчних рівнянь методом Гаусса.
2. Розробити програму розв’язання системи лінійних алгебраїчних рівнянь вз тридіагональною матрицею коефіцієнтів методом прогонки.
3. Розробити програму для побудови послідовності Фібоначі.
4. Розробити програму для сортування елементів у послідовності чисел.
5. Розробити програму для розв’язання квадратних рівнянь.

Контрольні питання

1. Що таке PHP? Як PHP працює?
2. Назвіть базові типи PHP?
3. Як влаштовані масиви в PHP?
4. Назвіть відмінність між echo, print, printf.
5. Поясніть тип object у PHP.

Література: [1–8].

Лабораторна робота № 2

Тема. Робота з HTTP-протоколом

Мета: отримати практичні навички роботи з HTTP-протоколом мовою програмування PHP.

Короткі теоретичні відомості

1. Обробка параметрів, переданих за допомогою HTTP-методів

1.1 HTTP GET

GET використовується для запиту даних із зазначеного ресурсу.

Зауважте, що рядок запиту (пари імен / значень) надсилається в URL-адресі запиту GET:

/test/demo_form.php?name1=value1&name2=value2

Що потрібно знати використовуючи GET:

– запити GET можна кешувати;

– запити GET залишаються в історії браузера;

– запити GET можна додавати до закладки;

– запити GET ніколи не слід використовувати для роботи з конфіденційними даними;

– запити GET мають обмеження довжини;

– запити GET використовують лише для запиту даних (не зміни).

Для роботи із GET – параметрами використовується змінна $_GET.

$_GET – Ассоціативний массив зміних, переданих PHP-скриптові через параметри URL (відомі також як рядок запиту). Зауважте, що масив не тільки заповняється для GET-запитів, а також для всіх запитів, які оформленні у вигляді рядка запиту.

```
<?php
echo ‘Привет, ‘ . htmlspecialchars($_GET["name"]) . ‘!’;
?>
```

Мається на увазі, що користувач ввів в браузері адресу /?name=Иван

Зверніть увагу на функцію htmlspecialchars(), ця функція перетворює html теги на відповідні структури, наприклад:

- ‘&’ (амперсанд) перетворюється в ‘&&’
- ‘"‘ (подвійна) перетворюється в ‘&"‘ when ENT_NOQUOTES is not set.
- ‘‘‘ (одинарна) перетворюється в ‘&’’ тільки в режимі ENT_QUOTES.
- ‘<‘ (знак "менше ніж") перетворюється в ‘&<‘
- ‘>‘ (знак "більше ніж") перетворюється в ‘&>‘

Якщо не обробляти передані користувачем дані, можна опинитися в доволі небезпечній ситуації. Нехай у нас є файл form.php, який містить форму, яка за допомогою методу GET відправляє параметр name на скрипт form_process.php. form_process.php, у свою чергу виводить даний параметр.

1. 2 HTTP POST

POST використовується для передавання даних на сервер для створення /

оновлення ресурсу.

Дані, надіслані на сервер з POST, зберігаються в тілі HTTP-запиту:

POST /test/demo_form.php HTTP/1.1

Host: w3schools.com

name1=value1&name2=value2

Деякі інші примітки щодо запитів POST:

– запити POST ніколи не кешуються;

– запити POST не залишаються в історії Web-переглядача;

– запити POST не можна додавати до закладки;

– запити POST не мають обмежень щодо довжини даних.

Для роботи з POST парметрами в PHP використовується змінна $_POST

$_POST – Асоціативний масив даних, переданих скрипту через HTTP методом POST р використаннzv application/x-www-form-urlencoded або multipart /form-data в заголовку Content-Type запиту HTTP.

<?php

echo ‘Привет ‘ . htmlspecialchars($_POST["name"]) . ‘!’;

?>

2 Робота з HTTP-заголовками

Для відправлення HTTP-заголовка в PHP використовується функція header:

header (string $header [, bool $replace = TRUE [, int $http_response_code]]) : void

Функцію header () можна викликати тільки якщо клієнту ще не передавалися дані. Тобто вона має бути першою у відповіді, перед її викликом не повинно бути ніяких HTML-тегів, порожніх рядків та. ін.. Досить часто виникає помилка, коли під час читання коду файловими функціями, на кшталт include або require, у цьому коді трапляються прогалини або порожні рядки, які виводяться до виклику header (). Ті ж проблеми можуть виникати у разі використання PHP / HTML в одному файлі.

<html>

```
<?php
/* Цей приклад призведе до помилки */
header(‘Location: http://www.example.com/’);
exit;
?>
```

Приклади встановлення заголовків:

```
<?php
header("HTTP/1.0 404 Not Found");
?>
```

3. Робота із COOKIE

Cookie (кукі) - це невеликі набори даних (не більше 4 кбайт), за допомогою яких Web-сайт може зберегти на комп’ютері користувача будь-яку інформацію. За допомогою куки можна відстежувати активність користувача на сайті: авторизований користувач на сайті чи ні, відстежувати історію його візитів і т. д.

Для збереження Cookie на комп’ютері користувача використовується функція setcookie (). Вона має таке визначення:

```
bool setcookie(string $name, string $value, int $expire,
 string $path, string $domain, bool $secure, bool $httponly);
```

– **name**: ім’я cookie, яке використовуватиметься для доступу до його значення;

– **value**: значення або вміст cookie - будь-який алфавітно-цифровий текст не більше 4 кбайт;

– **expire** (необов’язковий параметр): термін дії, після якого cookie знищуються. Якщо даний параметр не встановлений або дорівнює 0, то знищення cookie відбувається після закриття браузера;

– **Path** (необов’язковий параметр): шлях до каталогу на сервері, для якого будуть доступні cookie. Якщо задати ‘/’, cookie будуть доступні для всього сайту. Якщо задати, наприклад, ‘/ mydir /’, cookie будуть доступні тільки з каталогу / mydir / ‘і всіх його підкаталогів. Заводський параметр –це поточний

каталог, в якому встановлюються cookie.

– **Domain** (необов'язковий параметр): задає домен, для якого будуть доступні cookie. Якщо це домен другого рівня, наприклад, localhost.com, то cookie доступні для всього сайту localhost.com, у тому числі і для його піддоменів типу blog.localhost.com.

– **Secure** (необов'язковий параметр): указує на те, що значення cookie має передаватися по протоколу HTTPS. Якщо задано true, cookie від клієнта буде передано на сервер, тільки якщо встановлено захищене з'єднання. За замовчуванням дорівнює false.

```
$key = 'message';
$value = 'Hello World';
setcookie($key, $value);
setcookie('temp','temp', time()+3600); // срок дії 1 година
```

Щоб отримати cookie, можна використовувати глобальний асоціативний масив $_COOKIE:

```
if (isset($_COOKIE['message'])) echo $_COOKIE['message'];
```

Для видалення cookie досить в якості терміну дії вказати будь-який час в минулому:

```
setcookie ("message", "", time() - 3600);
```

Завдання до самостійної роботи

Розробіть php-додаток, який реалізує механізм захисту сторінки. Функціонал захищеної сторінки реалізуйте згідно з варіантом.

Варіанти:

1. Форма додавання двох чисел.
2. Форма перетворення тексту на повністю капсовий.
3. Форма видалення пробілів з тексту.
4. Форма перевірки парності числа.
5. Форма перевірки на ділення двох чисел без остачі.
6. Форма конвертації десяткового числа в бінарне.
7. Форма конвертації бінарного числа в двійкове.

8. Форма обрізання тексту до певної (заданої користувачем) довжини.

9. Форма пошуку підрядка в рядку.

10. Твій варіант 10! Тобі пощастило, можеш вибрати будь-який варіант із 1–9.

Контрольні питання

1. Як у PHP можна встановити HTTP заголовки?

2. Як у PHP можна працювати з HTTP Cokkie?

3. Як у PHP можна працювати з HTTP Session?

Література: [1–8].

Лабораторна робота № 3

Тема. Використання mysqli і PDO для роботи з базою даних

Мета: отримати практичні навички роботи з базою даних, використовуючи mysqli і PDO.

Короткі теоретичні відомості

1. Mysqli

Розширення mysqli, або як його ще називають поліпшене (improved) розширення MySQL, було розроблено, щоб надатидати можливість програмістам у повній мірі скористатися функціоналом MySQL-сервера версій 4.1.3 і вище. Розширення mysqli включається у поставку PHP версій 5 і вище.

Для ввімкнення розширення потрібно(модифікувати php.ini):

1. Установити директорію розширень

; Directory in which the loadable extensions (modules) reside.

; http://php.net/extension-dir

; extension_dir = "./"

; On windows:

extension_dir = "C:/php/ext"

2. Розкоментувати потрібне розширення

extension=php_mysqli.dll

1.1 З’єднання з базою даних

Для відкриття з’єднання з базою даних потрібно в конструктор mysqli передати параметри підключення:

```
$mysqli = new mysqli("localhost", "user", "password", "database");
```

1.2 Виконання запитів

За виконання запитів відповідають функції mysqli_query(), mysqli_real_query() і mysqli_multi_query(). Найчастіше застосовується функція mysqli_query(), оскільки вона виконує відразу два завдання: виконує запит і буферизует на клієнті результат цього запиту (якщо він є). Виклик mysqli_query() ідентичний послідовному виклику функцій mysqli_real_query () і mysqli_store_result().

1.3 DDL

```
<?php
$mysqli = new mysqli("localhost", "root", "", "test");
$mysqli->query("DROP TABLE IF EXISTS test");
$mysqli->query("CREATE TABLE test(id INT, name text)");
```

1.4 Виконання CRUD

CREATE

```
$mysqli->query("INSERT INTO test(id,name) VALUES (1,’alex’), (2, ‘john’);");
```

READ

```
$res = $mysqli->query("SELECT * FROM test;");
while ($row = $res->fetch_assoc()) {
   echo " id = " . $row[‘id’] . " name = ". $row[‘name’] . "\n";
}
$res = $mysqli->query("SELECT * FROM test;");
while ($row = $res->fetch_assoc()) {
   echo " id = " . $row[‘id’] . " name = ". $row[‘name’] . "<br>";
}
$mysqli->query("UPDATE test set name = ‘alex new’ where id = 1;");
$res = $mysqli->query("SELECT * FROM test;");
```

```
while ($row = $res->fetch_assoc()) {
   echo " id = " . $row[‘id’] . " name = ". $row[‘name’] . "<br>";
}
```

DELETE

```
$mysqli->query("DELETE FROM test where id = 1;");
```

2. Транзакції

Алгоритм, під час роботи з транзакціями:

- почніть транзакцію за допомогою команди mysqli_begin_transaction.
- виконайте одну або більше команд SQL, наприклад SELECT, INSERT, UPDATE або DELETE.
- перевірте, чи немає помилки і чи все відповідає вашим вимогам.
- якщо є якісь помилки, то виконайте команду ROLLBACK, інакше виконайте команду COMMIT.

Синтаксис:

mysqli_begin_transaction (mysqli $link [, int $flags = 0 [, string $name]]) : bool

Коректні прапори:

• MYSQLI_TRANS_START_READ_ONLY: Стартувати транзакцію як "START TRANSACTION READ ONLY". Потрібно MySQL 5.6 або вище.

• MYSQLI_TRANS_START_READ_WRITE: Стартувати транзакцію як "START TRANSACTION READ WRITE". Потрібно MySQL 5.6 або вище.

• MYSQLI_TRANS_START_WITH_CONSISTENT_SNAPSHOT: Стартувати транзакцію як "START TRANSACTION WITH CONSISTENT SNAPSHOT".

Після використання транзакцій:

```
<?php
$mysqli = new mysqli("127.0.0.1", "my_user", "my_password", "sakila");
if ($mysqli->connect_errno) {
   printf("Connect failed: %s\n", $mysqli->connect_error);
   exit();
```

```
}
$mysqli->begin_transaction(MYSQLI_TRANS_START_READ_ONLY);
$mysqli->query("SELECT first_name, last_name FROM actor");
$mysqli->commit();
$mysqli->close();
?>
```

3. PDO

PDO (PHP Data Objects) – розширення PHP, яке реалізує взаємодію з базами даних за допомогою об’єктів. Профіт у тому, що відсутня прив’язка до конкретної системи управління базами даних.

Наданий інтерфейс підтримує, серед інших, такі популярні СУБД:

- MySQL;
- SQLite;
- PostgreSQL;
- Microsoft SQL Server.

3.1 Підключення до БД

Відомості для підключення до бази, представлені у вигляді рядка. Синтаксис опису відрізняється залежно від використовуваної СУБД. У прикладі працюємо з MySQL / MariaDB, тому вказуємо:

– тип драйвера;

– ім’я хоста, де розташована СУБД;

– порт (необов’язково, якщо використовується стандартний порт 3306);

– ім’я бази даних;

– кодування (необов’язково).

```
$dsn = "mysql:host=localhost;port=3306;dbname=solar_system;charset=utf8";
```

Тепер, коли рядок DSN готовий, створимо PDO-об’єкт. Конструктор на вході приймає такі параметри:

– рядок DSN;

– ім’я користувача, що має доступ до бази даних;

– пароль цього користувача;

– масив з додатковими параметрами (необов’язково).

```
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
  ];
$pdo = new PDO($dsn, ‘testuser’, ‘testpassword’, $options);
```

3.2 Виконання запитів

Для виконання запитів використовуються функції exec() і query().

exec() – запускає SQL-запит на виконання і повертає кількість рядків, задіяних для його виконання. Не повертає результат вибірки оператором SELECT. Якщо вам потрібно вибрати дані цим оператором одного разу під час виконання програми, скористайтеся методом PDO::query().

query() – виконує SQL-запит і повертає результуючий набір у вигляді об’єкта PDOStatement.

Приклад з використанням **query**:

```
<?php
$dsn = "mysql:host=localhost;port=3306;dbname=test;charset=utf8";
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
  ];
$pdo = new PDO($dsn, ‘root’, ‘‘, $options);
$res = $pdo->query("SELECT * FROM test;");
foreach($res as $row){
     echo ‘id = ‘ . $row[‘id’] . ‘ name = ‘ . $row[‘name’] . ‘amount = ‘ .
$row[‘amount’] . ‘<br>‘;
}
```

Приклад з використанням exec:

```
<?php
$dsn = "mysql:host=localhost;port=3306;dbname=test;charset=utf8";
```

```
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
   ];
$pdo = new PDO($dsn, 'root', '', $options);
$res = $pdo->query("SELECT * FROM test;");
foreach($res as $row){
     echo 'id = ' . $row['id'] . ' name = ' . $row['name'] . 'amount = ' . $row['amount'] . '<br>';
}
$rowCount = $pdo->exec("INSERT INTO test(id, name, amount) VALUES(3, 'Sarah', 44);");
echo 'Inserted. Affected rows ' . $rowCount . '<br>';
```

3.3 Іменовані параметри та підготовлені запити

Якщо ж у запит передається хоча б одна змінна, то цей запит в обов'язковому порядку повинен виконуватися тільки через підготовлені відображення. Що це таке? Це звичайний SQL - запит, у якому замість змінної ставиться спеціальний маркер - плейсхолдер. PDO підтримує позиційні плейсхолдери (?), для яких важливий порядок переданих змінних, і іменовані (:name), для яких порядок неважливий.

```
$params = array(':username' => 'test', ':email' => $mail, ':last_login' => time() – 3600);
$pdo->prepare('
   SELECT * FROM users
   WHERE username = :username
   AND email = :email
   AND last_login > :last_login');
$pdo->execute($params);
```

Щоб виконати такий запит, спочатку його потрібно підготувати за допомогою функції prepare(). Вона також повертає PDO statement.

```
<?php
$dsn = "mysql:host=localhost;port=3306;dbname=test;charset=utf8";
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
  ];
$pdo = new PDO($dsn, 'root', '', $options);
$res = $pdo->query("SELECT * FROM test;");
foreach($res as $row){
    echo 'id = ' . $row['id'] . ' name = ' . $row['name'] . 'amount = ' . $row['amount'] . '<br>';
}
$params = [
    'id' => 4,
    'name' => 'some name',
    'amount' => 123
];
$pdoStatement = $pdo->prepare("INSERT INTO test(id, name, amount) VALUES(:id, :name, :amount);");
$pdoStatement = $pdoStatement->execute($params);
$res = $pdo->query("SELECT * FROM test;");
foreach($res as $row){
    echo 'id = ' . $row['id'] . ' name = ' . $row['name'] . 'amount = ' . $row['amount'] . '<br>';
}
```

3.4 Відображення в об'єкти

PDO може відображати результати в об'єкти. Це стане в нагоді, якщо ви не хочете використовувати користувальницький рівень абстракції бази даних, але все одно хочете поведінку подібну ORM. Уявімо собі, що у нас є клас User з деякими властивостями, які відповідають іменам полів з бази даних.

```
class TestItem{
    public $id;
    public $name;
    public $amount;

    public function info(){
        echo ‘id = ‘ . $this->id . ‘ name = ‘ . $this->name . ‘ amount = ‘ . $this->amount;
    }
}
```

Без зіставлення об’єктів нам потрібно було б заповнити кожне значення поля (вручну або за допомогою конструктора), перш ніж ми зможемо правильно використовувати метод info ().

Це дозволяє визначати ці властивості до того, як об’єкт буде побудований навіть! Для isntance:

```
<?php
class TestItem{
    public $id;
    public $name;
    public $amount;
    public function info(){
        echo ‘id = ‘ . $this->id . ‘ name = ‘ . $this->name . ‘ amount = ‘ . $this->amount;
    }
}
$dsn = "mysql:host=localhost;port=3306;dbname=test;charset=utf8";
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
  ];
```

```
$pdo = new PDO($dsn, ‘root’, ‘‘, $options);
$query = "SELECT * FROM test";
$result = $pdo->query($query);
$result->setFetchMode(PDO::FETCH_CLASS, ‘TestItem’);
while ($test = $result->fetch()) {
   echo $test->info() . "<br>";
}
```

Завдання до самостійної роботи

До проекту https://github.com/endlesskwazar/php-examples, гілка – auth-hash додайте функціонал реєстрації.

Контрольні питання

1. Що таке MySQLI?
2. Як можна виконати SQL-запит за допомогою MySQLI?
3. Поясніть принцип роботи з транзакціями в MySQLI.
4. Що таке PDO? Назвіть відмінності PDO від MySQLI.
5. Що таке підготовлені запити в PDO?

Література: [1–8].

Лабораторна робота № 4

Тема роботи: Laravel

Мета роботи: ознайомитися з базовими можливостями Web-фреймворка Laravel.

Короткі теоретичні відомості

1. установлення Laravel

Laravel – безкоштовний Web-фреймворк з відкритим кодом, призначений для розробки з використанням архітектурної моделі MVC (англ. Model View Controller – модель – уявлення – контролер). Laravel випущений під ліцензією MIT.

Для того щоб почати розробляти за допомогою Laravel зручно використовувати composer. Тоді для створення проекту достатньо виконати

команду:

composer create-project --prefer-dist laravel/laravel blog

1.1 Структура проекту

app містить базовий код для вашої програми Laravel.

Http – каталог Http містить різні фільтри, запити та контролери.

Bootstrap – містить скрипти для роутинга

config – містить конфігураційні файли

database – містить міграції, сіди

public – загальнодоступний каталог допомагає у запуску проекту Laravel, а також містить інші сценарії (JavaScript і CSS), а також зображення, необхідні для вашого проекту.

Resources – каталог ресурсів містить всі файли Sass, мовні (локалізаційні) файли, шаблони (якщо такі є).

routes – каталогів містить всі файли визначення для маршрутизації, такі як console.php, api.php, channels.php і т.д.

storage – каталог зберігає файли сеансів, кеш, скомпільовані шаблони, а також різні файли, що генеруються фреймворком.

test – директорія, яка містить тести: unit, інтеграційні

1.2 Налаштування проекту

Часто необхідно мати різні значення для різних налаштувань в залежності від середовища, у якому виконується додаток. Наприклад, ви можете вирішити використовувати різні драйвери кешу на локальному та продакшн-сервері.

Для цього в Laravel використовується PHP-бібліотека DotEnv від Ванса Лукаса. У свіжій інсталяції Laravel в корені вашого застосування буде файл .env.example. Якщо ви встановили Laravel за допомогою Composer, цей файл буде автоматично перейменований в .env, інакше вам слід зробити це вручну.

Усі змінні в цьому файлі змінні будуть завантажені в супер-глобальну змінну PHP $_ENV, коли ваш додаток отримає запит. Але ви можете використовувати допоміжну функцію env() для отримання значень цих змінних у ваших конфігураційних файлах. Насправді, якщо ви поглянете в файли

налаштувань Laravel, то помітите кілька опцій, що вже використовують цю функцію:

‘Debug’ => env (‘APP_DEBUG’, false),

Другий аргумент цієї функції – значення за замовчуванням. Воно буде використано, якщо такої змінної середовища немає.

2. Artisan

Artisan – це інтерфейс командного рядка, який містить Laravel. Він надає ряд корисних команд, які допоможуть вам під час створення програми. Щоб переглянути список усіх доступних команд Artisan, можна скористатися командою list:

php artisan list

2.1 Запуск Laravel-застосунка

Для запуску Laravel-застосунка використовується команда:

php artisan serve

2.2 Реєстрація, авторизація, аутентифікація

Laravel має вбудований функціонал реєстрації та авторизації. Для включення цього модуля достатньо виконати команду:

php artisan make:auth

Ця команда створить усі необхідні файли для реєстрації та авторизації

Для того, щоб реєстрація і авторизація працювали потрібно застосувати міграції:

php artisan migrate

Щоб ця команда пройшла успішно, потрібно мати налаштоване з’єднання з базою даних.

3. Маршрутизація

Усі маршрути (routes) Laravel визначені у файлах маршрутів, які розташовані в каталозі routes. Ці файли автоматично завантажуються фреймворком. У файлі routes/web.php визначені маршрути для вашого web-інтерфейсу. Ці маршрути належать до групи посередників web, які забезпечують такі можливості, як стан сесії і CSRF-захист.

Маршрутизатор дозволяє реєструвати маршрути для будь-якого HTTP-запиту:

```
Route::get($uri, $callback);
Route::post($uri, $callback);
Route::put($uri, $callback);
Route::patch($uri, $callback);
Route::delete($uri, $callback);
Route::options($uri, $callback);
```

Іноді необхідно зареєструвати маршрут, який відповідає на HTTP-запити декількох типів. Це можна зробити за допомогою match(). Або ви можете зареєструвати маршрут, який відповідає на HTTP-запити всіх типів, за допомогою методу any ():

```
Route::match(['get', 'post'], '/', function () {
  //
});
Route::any('foo', function () {
  //
});
```

3.1 Маршрут як функція

У Laravel найпростіші маршрути беруть URI (шлях) і функцію-замикання, надаючи дуже простий і виразний метод визначення маршрутів:

```
Route::get('foo', function () {
  return 'Hello World';
});
```

3.2 Передавання параметра GET

Зрозуміло, іноді вам може знадобитися захопити сегменти URI у вашому маршруті. Наприклад, якщо вам необхідно захопити ID користувача з URL, це можна зробити, визначивши параметри маршруту:

```
Route::get('user/{id}', function ($id) {
  return 'User '.$id;
```

```
});
```

Ви можете визначити скільки завгодно параметрів:

```
Route::get('posts/{post}/comments/{comment}', function ($postId,
$commentId) {
  //
});
```

Іноді необхідно вказати параметр маршруту, але при цьому зробити його наявність необов'язковим. Це можна зробити, помістивши знак питання? після назви параметра. Не забудьте поставити значення за замовчуванням для відповідної змінної маршруту:

```
Route::get('user/{name?}', function ($name = null) {
  return $name;
});
Route::get('user/{name?}', function ($name = 'John') {
  return $name;
});
```

Ви можете обмежити формат параметрів вашого маршруту за допомогою методу where() на примірнику маршруту. Метод where() приймає назву параметра і регулярний вираз, що визначає обмеження для параметра:

```
Route::get('user/{name}', function ($name) {
  //
})->where('name', '[A-Za-z]+');
Route::get('user/{id}', function ($id) {
  //
})->where('id', '[0-9]+');
Route::get('user/{id}/{name}', function ($id, $name) {
  //
})->where(['id' => '[0-9]+', 'name' => '[a-z]+']);
```

4. Контролери

Замість того, щоб визначити всю логіку обробки запитів як функцію у

файлах маршруту, ви можете організувати цю поведінку за допомогою класів Controller. Контролери можуть групувати логіку обробки пов'язаних запитів в один клас. Контролери зберігаються в каталозі app/Http/Controllers.

Для генерації шаблону контролера можна використати команду:

php artisan make:controller [Namespace][Name]Controller

Для створення маршруту до контролера, точніше до його методу, використовується такий синтаксис:

Route::[httpMethod](‘[route]’, ‘[controller_name]@controller_method’);

Route::get(‘foo’, ‘FooController@index’);

5. Blade views

Blade - простий, але потужний шаблонизатор, що поставляється з Laravel. На відміну від інших популярних, шаблонізатор для PHP Blade не обмежує вас у використанні чистого PHP-коду у ваших вставках. Насправді всі Blade - представлення скомпільовані в чистий PHP-код і кеш, поки в них немає змін, а отже, Blade майже не навантажує вашу програму. Файли уявлень Blade використовують розширення .blade.php і зазвичай зберігаються в папці resources / views.

6. Валідація запитів

Laravel поставляється з простою, зручною системою валідації (перевірки вхідних даних на відповідність правилам) і отримання повідомлень про помилки - класом Validation.

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class FooController extends Controller
{
   public function index(){
      return "foo index";
   }
   public function edit(){
```

```
        return view(‘foo’);
    }
    public function store(Request $request){
        $request::validate(
            array(‘name’ => ‘required’),
        );
    }
}
```

7. Перевизначення Request

Для більш складних сценаріїв перевірки можна створити "запит форми". Запити форми – це спеціальні класи запитів, які містять логіку перевірки. Щоб створити клас запиту форми, скористайтеся командою make:request:

```
php artisan make:request FooRequest
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Http\Requests\FooRequest;
class FooController extends Controller
{
    public function index(){
        return "foo index";
    }
    public function edit(){
        return view(‘foo’);
    }
    public function store(FooRequest $request){
        // The incoming request is valid...
        // Retrieve the validated input data...
         $validated = $request->validated();
    }
}
```

Завдання до самостійної роботи

Створіть контролер, який приймає два запити: по GET – виводить форму, по POST - валідує і повертає користувачеві дані, які він увів. Поля форми згідно з варіатом.

Варіанти:

1. { name: required, min-length:3, age: number, min-value:18 }
2. { title: required, max-length: 10, price: required, unsigned, cannot allow 0 }
3. { name: required car-number: need contain 6 digit }
4. { appointment: required, patient_id: required, unsigned, doctor_id: required, unsigned }
5. { phoneModel: required, is4Gsupported: required }

Контрольні питання

1. Що таке Laravel?
2. Поясніть структуру директорій Laravel-проекту.
3. Як додати функціонал авторизації та реєстрації в Laravel?
4. Що таке Blade?
5. Як створити маршрут і додати до нього контролер?

Література: [1–8].

Лабораторна робота № 5

Тема. Робота з базою даних у Laravel

Мета: отримати практичні навички роботи з базою даних, використовуючи Eluqolent.

Короткі теоретичні відомості

1. Active Record

Active record (AR) – шаблон проектування додатків, описаний Мартіном Фаулером у книзі Patterns of Enterprise Application Architecture («Шаблони архітектури корпоративних додатків»). AR, є популярним способом доступу до даних реляційних баз даних в об’єктно-орієнтованому програмуванні.

Схема **Active Record** – це підхід до доступу до даних в базі даних. Таблиця бази даних або подання обгорнуті в класи. Отже, об'єктний екземпляр прив'язаний до єдиного рядка в таблиці. Після створення об'єкта новий рядок додаватиметься до таблиці до збереження. Будь-який завантажений об'єкт отримує свою інформацію від бази даних. Коли об'єкт оновлений, відповідний рядок у таблиці також буде оновлено. Клас обгортки реалізує методи засоби доступу або властивості для кожного стовпця в таблиці.

2. Eluqolent

Система об'єктно-реляційного відображення (ORM) Eloquent – красива і проста реалізація шаблону ActiveRecord у Laravel для роботи з базами даних. Кожна таблиця має відповідний клас-модель, який використовується для роботи з цією таблицею. Моделі дозволяють запитувати дані з таблиць, а також вставляти в них нові записи.

3. Міграції

Міграції схожі на систему контролю версій, але тільки для бази даних (БД). Вони дозволяють команді розробників легко змінювати схему БД і ділитися цими змінами. Міграції зазвичай пов'язані з будівником схем Laravel (Laravel's schema builder) для полегшення створення схеми БД.

Laravel Schema facade надає незалежну від БД підтримку створення та управління таблицями. Вона надає виразний API для всіх підтримуваних Laravel БД.

Для створення міграції використовуйте make: migration Artisan command:

php artisan make:migration create_users_table

Міграція буде поміщена в папку database/migrations. Кожне ім'я файлу міграції містить мітку часу, яка дозволяє Laravel визначати порядок застосування міграцій.

Опції --table і --create слугує для завдання імені таблиці та указання чи міграція створить нову таблицю. Інакше кажучи, ці опції слугують для попереднього заповнення генерування файлу міграції:

php artisan make:migration add_votes_to_users_table --table=users

php artisan make:migration create_users_table —create=users

Клас міграції містить 2 методи: up і down. up потрібен для додавання нових таблиць, стовпців або індексів у БД, тоді як down просто скасовує операції, виконані в методі up.

```
<?php
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;
class CreateFlightsTable extends Migration
{
   /**
    * Run the migrations.
    *
    * @return void
    */
   public function up()
   {
      Schema::create('flights', function (Blueprint $table) {
         $table->increments('id');
         $table->string('name');
         $table->string('airline');
         $table->timestamps();
      });
   }
   /**
    * Reverse the migrations.
    *
    * @return void
    */
   public function down()
   {
```

```
        Schema::drop('flights');
    }
}
```

4. Сіди

Laravel має простий метод заповнення бази даних тестовими даними, використовуючи класи-наповнювачі (seed classes). Ці класи зберігаються в database/seeds. Можна використовувати будь-яке ім'я для назви класу-наповнювача, але доцільно застосовувати імена, подібні UsersTableSeeder. За замовчуванням клас DatabaseSeeder уже створено в папці наповнювачів. У цьому класі ви можете використовувати метод call для запуску інших наповнювачів, що дозволяє вам контролювати порядок наповнення.

Для створення наповнювача можна використовувати команду make: seeder Artisan command. Команда створює наповнювач у папці database/seeds:

```
php artisan make:seeder UsersTableSeeder
```

За замовчуванням клас-наповнювач містить тільки один метод: run. Метод викликається, коли запускається команда db: seed Artisan command. У методі run ви можете вставляти дані в БД будь-яким зручним способом.

```
<?php
use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;
class DatabaseSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        DB::table('users')->insert([
```

```
            'name' => str_random(10),
            'email' => str_random(10).'@gmail.com',
            'password' => bcrypt('secret'),
        ]);
    }
}
```

У класі DatabaseSeeder можна використовувати метод call для запуску додаткових класів-наповнювачів. Метод call дозволить робити наповнення БД за допомогою безлічі файлів, не створюючи одиного великого класу-наповнювач. Просто передай методу бажаний клас-наповнювач для його запуску:

```
/**
 * Run the database seeds.
 *
 * @return void
 */
public function run()
{
    $this->call(UsersTableSeeder::class);
    $this->call(PostsTableSeeder::class);
    $this->call(CommentsTableSeeder::class);
}
```

5. Моделі

Насамперед створимо модель Eloquent. Моделі зазвичай розташовуються в папці app, але ви можете помістити їх у будь-яке місце, у якому працює автозавантажувач відповідно до вашого файлу composer.json. Усі моделі Eloquent успадковують клас Illuminate\Database\Eloquent\Model.

Найпростіше створити екземпляр моделі за допомогою Artisan-команди make: model:

```
php artisan make:model User
```

Якщо ви хочете створити міграцію БД для створенні моделі, використовуйте параметр --migration або -m:

```
php artisan make:model User --migration
php artisan make:model User -m
```

Зауважте, що ми не вказали, яку таблицю Eloquent повинен прив’язати до нашої моделі. Якщо це ім’я не вказано явно, то відповідно до прийнятого угодою буде використано ім’я класу в нижньому регістрі і в множині. У нашому випадку Eloquent припустить, що модель Flight зберігає свої дані в таблиці flights. Ви можете вказати довільну таблицю, визначивши властивість table в класі моделі:

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Flight extends Model
{
  protected $table = ‘my_flights’;
}
```

Завдання до самостійної роботи

До проекту https://github.com/endlesskwazar/php-examples, гілка — lara-db-ex. Доробіть CRUD-операції.

Контрольні питання

1. Поясніть патерн Active Record.
2. Як у Laravel створити модель.
3. Що таке міграції?
4. Що таке сіди?

Література: [1–8].

3 КРИТЕРІЇ ОЦІНЮВАННЯ ЗНАНЬ СТУДЕНТІВ

Перелік лабораторних робіт складається з п’яти робіт, які спрямовані на перевірку знань студентів з теорії та практики розв’язання прикладних задач.

Виконання лабораторних робіт і написання звітів є обов’язковим для всіх студентів. За п’ять робіт студент може отримати 30 балів.

За кожну лабораторну роботу виставляють:

– 6 балів, якщо студент виконав лабораторну роботу, написав звіт і захистив його, тобто правильно та вичерпно відповів на всі питання. Допускаються незначні неточності під час відповіді на одне з контрольних питань;

– 3 бали, якщо студент виконав лабораторну роботу, написав звіт, але не захистив його;

– у решті випадків виставляють 0 балів.

СПИСОК ЛІТЕРАТУРИ

1. Суэринг С. PHP и MySQL. Библия программиста. 2-е изд. Москва: Диалектика, 2010. 912 с.

2. Зандстра М. PHP: объекты, шаблоны и методики программирования. 2-е изд. Вильямс, 2011. 460 с.

3. Зандстра М. PHP: объекты, шаблоны и методики программирования. 3-е изд. Вильямс, 2012. 480 с.

4. Ченгаев Д. Технологии PHP и MySQL для создания сайтов. Москва: Диалектика, 2009. 496 с.

5. Бернс Д. JavaScript. Вильямс, 2001. 464 с.

6. Гудман Д. JavaScript. Библия пользователя. 5-е изд. Москва: Диалектика, 2006. 1184 с.

7. Антес Е. История интернета. Москва, С.-Петербург, Киев: Диалектика, 2001. 336 с.

8. Кроудер Д. Разработка Web-узлов для «чайников». Москва, С.-Петербург, Киев: Диалектика, 2001. 336 с.

Додаток А

Зразок оформлення титульної сторінки звіту

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ЗВІТ
ІЗ ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«WEB-ТЕХНОЛОГІЇ»

Виконав студент групи _____
ПІБ студента
Перевірив посада, ПІБ викладача

КРЕМЕНЧУК 20__

Методичні вказівки щодо виконання лабораторних робіт з навчальної дисципліни «Web-технології» для студентів денної форми навчання зі спеціальності 122 – «Комп’ютерні науки» за освітньо-професійною програмою «Комп’ютерні науки» освітнього ступеня «Бакалавр» (частина 1)

Укладач старш. викл. О. О. Скриль
Відповідальний за випуск старш. викл. Т. В. Горлова

Підп. до др. _____. Формат 60×84 1/16. Папір тип. Друк ризографія.
Ум. друк. арк. _____. Наклад _____ прим. Зам. № _____ Безкоштовно.

Редакційно-видавничий відділ
Кременчуцького національного університету
імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600