МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Основи WEB програмування

Методичні вказівки для виконання лабораторних робіт

з дисципліни «Програмування -1 »

для студентів спеціальності „ Автоматизація та комп'ютерно-інтегровані технології ”

Рекомендовано Вченою радою інженерно-хімічного факультету

Київ
НТУУ «КПІ»
2016

Основи WEB програмування: Метод. вказівки до виконання лабораторних робіт з дисципліни «Програмування -1 » для студентів спеціальності „ Автоматизація та комп'ютерно-інтегровані технології” / Уклад.: О.В. Ситніков”, 2016. – 40с.

Гриф надано Вченою радою ІХФ

(Протокол № від 2016р.)

Навчальне видання

ОСНОВИ WEB ПРОГРАМУВАННЯ
Методичні вказівки до виконання лабораторних робіт з дисципліни „ Програмування -1 ” для студентів спеціальності „ Автоматизація та комп'ютерно-інтегровані технології ”

Укладачі: Ситніков Олексій Володимирович

Відповідальний редактор А.І.Жученко, д-р техн.наук, проф.

Рецензент : А.Р. Степанюк, к.т.н., доц.

Авторська редакція

Зміст

Вступ

Історія розвитку Інтернет починається з 1986 року, коли Національний науковий фонд США (*NSF*) почав створення мережі *NSFNET*, в якій використовувались високошвидкісні телефонні канали, що з`єднували 6 суперкомп`ютерів у різних куточках країни на основі протоколу *TCP*/*IP* та інших технологій. Мережа *NSFNET* стала основною магістральною (*backbone*) структурою для Інтернет, офіційною датою виникнення якого є 1990 рік. В даний час основу Інтернет складають високошвидкісні магістральні мережі. Незалежні автономні мережі підключаються до магістральної мережі через крапки мережного доступу *NAP* (*Network Access Point*)

PHP (*Personal Home Pages*) - серверна мова сценаріїв, яка вбудовується безпосередньо в *HTML*-код, була розроблена у 1994 році Расмусом Лердорфом. Проте в 1997 році інтерпретатор був переписаний іншими програмістами - з'явилась мова *PHP*3 з ширшими можливостями, яка завоювала досить високу популярність. Крім цього, абревіатура *PHP* стала офіційно розшифровуватись як *PHP Hypertext Preprocessor* (препроцесор гіпертексту *PHP*). Використання *PHP* доцільне при створенні часто обновлюваних або громіздких у написанні програм, швидкість виконання для яких не є критичним параметром. PHP скрипт працює досить швидко, але не так швидко як заздалегідь скомпільована програма.

В *PHP* існує безліч інтерфейсів для роботи з БД: вбудовані бібліотеки для роботи з *MySQL*, Oracle, dbm, та інші, через стандарт відкритого інтерфейсу зв'язку з базами даних ODBC можна підключатися до всіх баз даних, до яких існує драйвер. Ефективність є дуже важливим чинником при програмуванні для середовищ розрахованих на безліч користувачів, до яких належить і *WEB*. Важливою перевагою *PHP* є те, що ця мова належить до інтерпретованих та дозволяє обробляти сценарії з достатньо високою швидкістю. продуктивність *PHP* цілком достатня для створення серйозних *WEB*-додатків.

MySQL - компактний багатопотоковий сервер баз даних. В даний час реляційні системи керування базами даних (СУБД) є важливим інструментом в багатьох областях, починаючи з традиційних: бізнес, наукове дослідження, освіта і закінчуючи розробкою пошукових серверів в *Internet*. Проте, не зважаючи на важливість наявності хорошої бази даних для ведення інформаційних ресурсів і доступу до них, багато організацій не застосовують їх в своїй роботі.

MySQL - реляційна СУБД типу клієнт-сервер, створена в Скандинавії. СУБД *MySQL* включає *SQL*-сервер і програми-клієнти, що здійснюють доступ до серверу, засоби адміністрування і програмний інтерфейс для програмування своїх особистих програм. СУБД *MySQL* продовжує стрімко розвиватися.

Apache HTTP-сервер - є кросплатформним ПО, підтримує операційні системи *Linux, MacOS, Windows*. Основними перевагами *Apache* вважаються надійність і гнучкість конфігурації. Він дозволяє підключати зовнішні модулі для надання даних, використовувати СУБД для аутентифікації користувачів, модифікувати повідомлення про помилки.

Ядро *Apache* включає в себе основні функціональні можливості, такі як обробка конфігураційних файлів, протокол *HTTP* і система завантаження модулів, повністю написано на мові програмування *C*.

Система конфігурації Apache заснована на текстових конфігураційних файлах. Має три умовних рівня конфігурації: Конфігурація сервера, Конфігурація віртуального хоста, Конфігурація рівня директорії.

Механізм віртуальних хостів. *Apache* має вбудований механізм віртуальних хостів. Він дозволяє повноцінно обслуговувати на одному IP-адресу безліч сайтів (доменних імен), відображаючи для кожного з них власне вміст. Для кожного віртуального хоста можна вказати власні настройки ядра і модулів, обмежити доступ до всього сайту або окремих файлів. Приклад встановлення наведено в додатку 1.

Лабораторна робота №1

Основи роботи з *PHP*- дизайнером

Мета роботи: Навчитись працювати з *PHP*-дизайнером. Поняття змінних в *PHP*-документах.

Теоретичні відомості

Вікно редактора *PHPDesigner* 8 виглядає наступним чином.

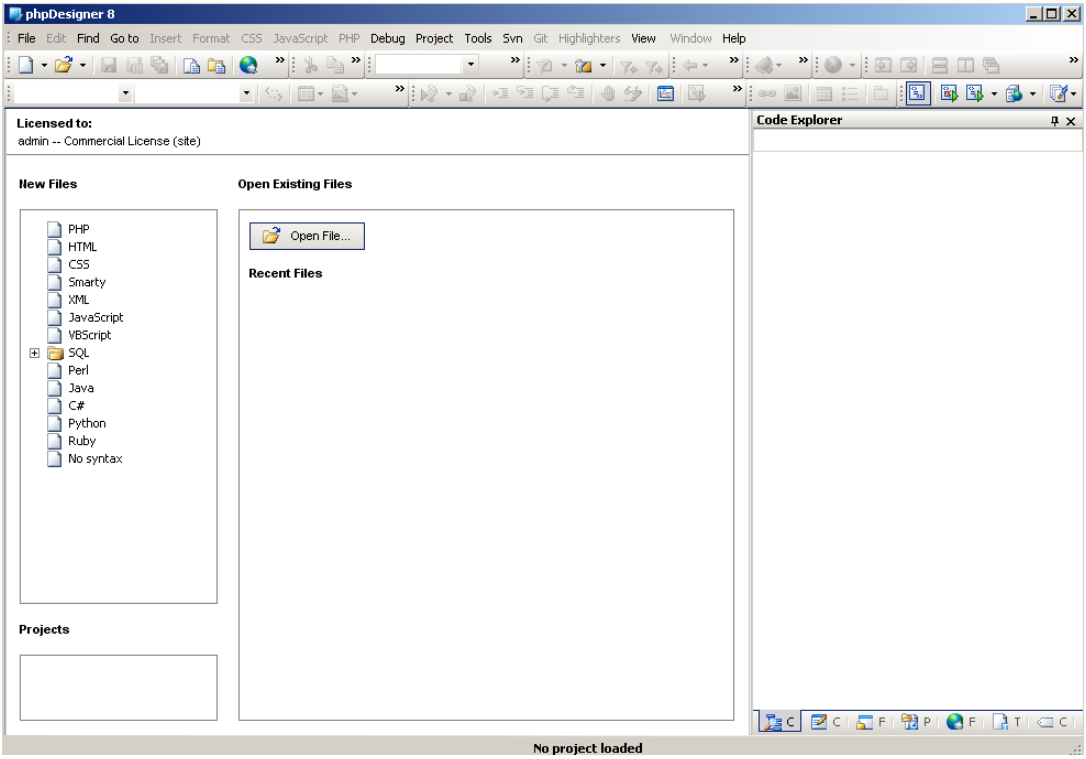


Рис.1 *PHPDesigner* 8

Пункт меню *file*, *edit* містить стандартні для *windows*-програм складові. По центру присутня кнопка, що пропонує відкрити файл, в правій частині присутній навігатор по коду (відкритий коли відкритий код).

В лівій частині пропонується створити новий файл вказаного типу, для кожного новоствореного документа буде створено шаблон з вмістом основних тегів.

В *PHP*-дизайнері існує можливість попереднього перегляду результату створення сторінки. Дана дія відбувається натисканням *Run.*

Для перегляду попереднього результату, перед натисканням *Run*, необхідно зберегти файл наступним чином:

назва.тип

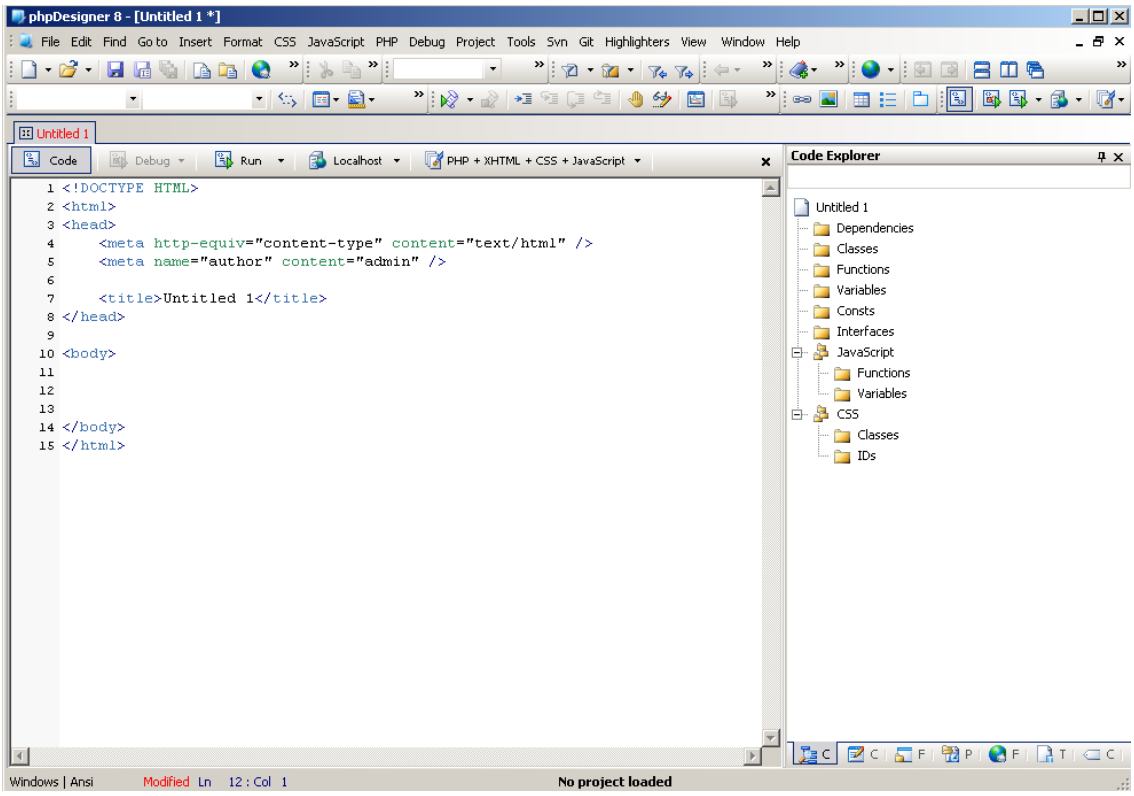


Рис. 2 Шаблон для *html*-документа.

Приклад виклику веб-сторінки в браузері за допомогою локального серверу:

localhost/шлях до документа/назва.*php*

Якщо не вказувати шлях, то документ повинен бути збережений за адресою *C*:/*AppServ*/*www*/, тоді виклик буде наступний:

localhost /назва.*php*

Розміщення *PHP*-коду на *html*-сторінці відбувається з використанням наступної конструкції:

<?php

PHP-код

?>

<?*php* – відкриває *PHP*-код, а ?> – відповідно закриває. Інформація, що розміщена між <?*php* … ?> буде сприйматися браузером, як *PHP*-код.

Розглянемо ряд команд *PHP*-коду:

echo “…” – дозволяє виводи інформацію, що знаходиться між подвійними лапками, на сторінку, приклад

echo “Привіт !”;

print (“…”) – виводить інформацію на екран, що знаходиться в дужках

Вивід тексту можна здійснювати використовуючи *echo* двічі.

```
<?php
echo “Привіт !”;
echo “Як справи ?”;
?>
```

Однак виникне проблема пов’язана з тим, що не відбудеться перехід на новій рядок після першого *echo* і вся інформація наведена у лапках буде виведена в одному рядку наступним чином:

Привіт !Як справи ?

Для того щоб здійснити перехід на наступний рядок після виводу першого текстового рядка необхідно в одному *html*-документі двічі відкрити *PHP*-код. Розглянемо на прикладі:

```
 <?php
echo “Привіт !”;
?>
<br>
<?php
echo “Як справи ?”;
?>
```

В *PHP*-коді після кожного рядка ставиться ; .

Змінні в *PHP*-документах

$ ім’я змінної = значення;

Приклад:

```
$a = 5;
$x= 2.56;
$c = “abc”;
$f = true;
```

Відповідно *a* – типу *integer*, *x* – типу *float*, *c* – типу *string* та *f* – типу *boolean*.

Вивести значення змінної можна за допомогою *echo* або *print*.

```
echo  $c;                 результат –  abc
echo  “abc $a”;           результат –  abc 5
echo  “abc ”.$c;          результат –  abc abc
print  $c;                результат –  abc
print  (“abc ”.$a);       результат –  abc 5
```

Якщо змінній надається двічі значення, то попереднє буде затиратись:

```
$a = 5;
$a = 7;
echo  $a;                 результат –  7
```

Порядок виконання роботи

1. Відкрити *PHP*-дизайнер.
2. Із запропонованого списку вибрати *html*-файл.
3. Між тегами *body* записати наступний код:

```
<?php
echo  “Привіт !”;
?>
```

4. Зберегти даний файл у своїй папці з ім’ям lesson_1_1, а як тип вибрати *PHP*.
5. Натиснути *Run*. Передивившись результат повернутись до коду натиснувши *Code*.
6. Внести зміни

```
<?php
echo  “Привіт !”;
?>
<br>
<?php
echo  “Як справи ?”;
```

```
?>
```

7. Зберегти з ім’ям lesson_1_2, передивитись результат та повернутись до коду
8. Внести зміни

```
<?php
$a = 5;
$x= 2.56;
$c = “abc”;
$f = true;
echo  $a;
?>
<br>
<?php
echo  $x ;
?>
<br>
<?php
echo  $c ;
?>
<br>
<?php
echo  $f ;
?>
```

9. Зберегти з ім’ям lesson_1_3, передивитись результат та повернутись до коду
10. Внести зміни до коду, використовуючи змінні та їх значення з п.8, що б результат виводився наступним чином :

a=5+2.56=7.56

зберегти з ім’ям lesson_1_4

11. Внести зміни до коду, що б результат виводився наступним чином :

f=1

f=0

зберегти з ім’ям lesson_1_5.

12.Внести зміни до коду, що б результат виводився наступним чином :

abc 5

abc

abc

зберегти з ім’ям lesson_1_6.

Вміст звіту: Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, коди сторінок створених відповідно до завдання 10-12.

Контрольні запитання

1. Призначення та складові *PHP*-дизайнера?
2. Структура *PHP*-кода ?
3. Що робить *echo*, *print*?
4. Засоби створення змінних у *PHP*.

Лабораторна робота №2

Конструкція *if-else*. Цикли. *Switch*

Мета роботи: Навчитись використовувати конструкцію *if-else*. Основу роботу з циклами. Робота з конструкцією *switch*.

Теоретичні відомості

Коли необхідно виконати дію при виконанні конкретної умови використовують конструкцію *if-else*. Виглядає наступним чином:

```
if  ( умова )
{
дії, якщо умова виконана;
}
else
{
дії, якщо умова не виконана;
}
```

До операцій порівняння відносять:

== – дорівнює, != – не дорівнює,

< / > – менше / більше,

<= / >= – менше або дорівнює / більше або дорівнює

Якщо необхідна перевірка по двох або більше умов :

&& – всі умови повинні виконуватись

| | – хоча б одна з умов повинна виконуватись

В *PHP* використовують два цикли – *while* та *for*.

```
while ( умова )
{
дії;
}
або
do
{
```

дії;

}

while (умова)

for (створюємо змінні; умова; дія)

{

дії;

}

Операції інкременту/декременту

$a++ – збільшення *a* на 1, повернення *a*

++$a– повернення *a*, збільшення *a* на 1

$a-- – зменшення *a* на 1, повернення *a*

--$a – повернення *a*, зменшення *a* на 1

$a+=x – дія відповідна до $a=$a+x

$a-=x – дія відповідна до $a=$a-x

Перевіряти конструкцію можна також за допомогою конструкції *switch*

switch (вираз або змінна) {

case значення 1:

дії;

break;

case значення 2:

дії;

break;

…

default:

дії якщо не співпало з жодним значенням;

}

break – оператор приривання дії групи операторів при відповідному *case*

Порядок виконання роботи

1. Відкрити *PHP*-дизайнер.
2. Із запропонованого списку вибрати *html*-файл.

3. Між тегами *body* записати наступний код:

```
<?php
$a=5;
if  ($a>5)
{
$b=sqr($a);
echo  $b;
}
else
{
$b=sqrt($a);
echo  $b;
}
?>
```

4. Зберегти даний файл у своїй папці з ім’ям lesson_2_1, а як тип вибрати *PHP*.
5. Натиснути *Run*. Передивившись результат повернутись до коду натиснувши *Code*.
6. Внести зміни

```
<?php
$a=5; $b=10;
if  ($a>=5)  && ($b!=11)
{
$x=$a+$b;
echo  $x;
}
?>
```

7. Зберегти з ім’ям lesson_2_2, передивитись результат та повернутись до коду
8. Внести зміни (розрахунок $y=\sqrt{a+0,5}$, при a є [5,13), крок по a = 2)

```
<?php
$a=5;
while ($a<13)
{
$y=sqrt($a+0.5);
echo  $y;
$a+=2;
}
?>
```

9. Зберегти з ім’ям lesson_2_3, передивитись результат та повернутись до коду
10. Внести зміни щоб кожний результат виводився з нового рядка
11. Зберегти з ім’ям lesson_2_4, передивитись результат та повернутись до коду
12. Внести зміни (розрахунок $y=\sqrt{a+0,5}$, при a є [1,10], крок по a = 0.5)

```
<?php
for ($a=1; $a<=10; $a+=0.5 )
{
$y=sqrt($a+0.5);
echo  $y;
}
?>
```

13. Зберегти з ім’ям lesson_2_5, передивитись результат та повернутись до коду
14. Внести зміни щоб кожний результат виводився з нового рядка, а кроків розрахунку було в тричі менше
15. Зберегти з ім’ям lesson_2_6, передивитись результат та повернутись до коду
16. Внести зміни

```
<?php
```

```
$a=2; $b=4; $c=5; $x=7;
switch ($a) {
        case 1:
          echo $b;
        break;
        case 2:
         echo $c;
        break;
        default:
         echo $x;
}
?>
```

17. Зберегти з ім’ям lesson_2_7, передивитись результат та повернутись до коду
18. Внести зміни: є 5 довільних значень змінної *a*, якщо заздалегідь надане значення змінної дорівнює одному з 5, то вивести його квадрат, коли варіантів не знайдено, то вивести – «варіанти відсутні».
19. Зберегти з ім’ям lesson_2_8, передивитись результат та повернутись до коду

Вміст звіту: Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, коди сторінок створених відповідно до завдання 10, 14, 18.

Контрольні запитання

1. Використання конструкції *if-else*.
2. Принцип роботи циклу *for*.
3. Принцип роботи циклу *while.*
4. Складові конструкції *switch.*

Лабораторна робота №3

Масиви. Цикл *foreach*.

Мета роботи: Навчитись обробляти масиви, використовувати цикл *foreach*.

Теоретичні відомості

Масиви у *PHP* призначені для зберігання інформації довільного вмісту. До комірок масиву можна вносити числову і текстову інформацію.

$array1[3]=5;

*array*1 – назва массиву, [3] – номер комірки, 5 – інформація, що передається комірці.

$array1[]=”ru”;

$array1[]=”en”;

$array1[]=”de”;

Коли не вказано номер комірки заповнення починається з комірки під номером 0.

Також масив можна заповнити рядком :

$mas1= array(“a”, “b”, “c”); або $mas1= array(1=> “a”, 2=> “b”, 3=> “c”);

При необхідності можна змінти нумерацію комірок, наприклад на букви, при цьому заповнювати необхідно в стопчик

$mas1= array(

“a” => “ru”

“b” =>“en”

“c”=> “de”

);

Вивід значення комірок можливо деякими способами:

print_r($array1) – виведеться весь масив: Array ([0] => ru [1] => en [2] => de)

print_r($array1[2]) або

echo $array1[2] – виведеться комірка масиву 2: de. Внаслідок того, що тільки одна комірка виводиться , то не вказуються, що це масив.

echo $mas1[“b”] – виведеться комірка масиву *b*: en.

Оператор *foreach* призначений для перебору елементів масиву

foreach (масив as змінна):

дії;

endforeach;

або

foreach (масив as змінна)

{

дії;

}

Як приклад:

foreach ($mas1 as $m)

{

echo $m;

}

Перебирається масив, та кожне значення комірки заноситься до змінної *m* та виводиться на екран

Порядок виконання роботи

1. Відкрити *PHP*-дизайнер.
2. Із запропонованого списку вибрати *html*-файл.
3. Між тегами *body* записати наступний код:

```
<?php
$array1[]=”ru”;
$array1[]=”en”;
$array1[]=”de”;
$ array2= array(1=> “ru”, 2=> “en”, 3=> “de”);
$ array3= array(
                “a” => “ru”
                “b” =>“en”
                “c”=> “de”
                 );
?>
```

4. Зберегти даний файл у своїй папці з ім’ям lesson_3_1, а як тип вибрати *PHP*.
5. Натиснути *Run*. Передивившись результат повернутись до коду натиснувши *Code*.
6. Внести зміни до коду, щоб в ньому виводився вміст комірок
7. Зберегти даний файл у своїй папці з ім’ям lesson_3_2, а як тип вибрати *PHP*.
8. Натиснути *Run*. Передивившись результат повернутись до коду натиснувши *Code*.
9. Внести зміни

```
<?php
$array1= array(1=> “ru”, 2=> “en”, 3=> “de”);
foreach ($array1 as $x)
{
echo  “мова - $x <br>”;
}
?>
```

10. Зберегти даний файл у своїй папці з ім’ям lesson_3_3, а як тип вибрати *PHP*.
11. Натиснути *Run*. Передивившись результат повернутись до коду натиснувши *Code*.
12. Внести зміни, задавши масив з 5 елементів(числа), на екран виводити тільки більші за 10 (виконувати за допомогою *foreach*)
13. Зберегти даний файл у своїй папці з ім’ям lesson_3_4, а як тип вибрати *PHP*.
14. Натиснути *Run*. Продивитись результат.

Вміст звіту: Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, коди сторінок створених відповідно до завдання п.6, 12.

Контрольні запитання

1. Як у *PHP* заповнюється масив ?
2. Як заповнити масив в циклі?
3. Якій в *PHP* передбачений вміст комірок?
4. Принцип роботи циклу *foreach*.

Лабораторна робота №4

Функції

Мета роботи: Навчитись використовувати, створювати функції у *PHP*.

Теоретичні відомості

В *PHP* так як і інших мовах програмування присутні функції, поділяються на вбудовані та визначені користувачем.

Конструкція функції визначеної користувачем:

```
function назва(глобальні зміні)
{
дії;
}
```

Глобальні зміні можуть бути відсутні

Прклад:

```
function func1()
{
echo “Hello !!!”;
}
```

Щоб викликати функцію необхідно прописати її ім’я:

func1(); – результатом роботи буди вивід тексту *Hello* !!!

Функція з глобальними параметрами:

```
$a=”name”;
function func2($a)
{
echo “Hello, “.$a.”!!!”;
}
```

func2($a); – результат роботи *Hello, name* !!!

Також в *PHP* значенням змінних можна надавати значення функції:

```
function func3($a)
{
$b=$a*$a;
```

```
return $b;
}
$x=func3(10);
echo $x;   –  результат роботи                100
```

або

```
$c=10
$x=func3($c);
echo $x;   –  результат роботи                100
```

return $b; – повернення значення змінної *b*

Порядок виконання роботи

1. Відкрити *PHP*-дизайнер.
2. Із запропонованого списку вибрати *html*-файл.
3. Між тегами *body* записати наступний код:

```
<?php
function func1()
{
echo “Hello !!!”;
}
func1();
?>
```

4. Зберегти даний файл у своїй папці з ім’ям lesson_4_1, а як тип вибрати *PHP*.
5. Натиснути *Run*. Передивившись результат повернутись до коду натиснувши *Code*.
6. Внести зміни до коду

```
$a=”name”;
function func2($a)
{
echo “Hello,  “.$a.”!!!”;
}
```

func2($a);

7. Зберегти даний файл у своїй папці з ім’ям lesson_4_2, а як тип вибрати *PHP*.
8. Натиснути *Run.* Передивившись результат повернутись до коду натиснувши *Code*.
9. Внести зміни

```
function func3($a)
{
$b=$a*$a;
return $b;
}
$x=func3(10);
echo $x;
```

10. Зберегти даний файл у своїй папці з ім’ям lesson_4_3, а як тип вибрати *PHP*.
11. Натиснути *Run.* Передивившись результат повернутись до коду натиснувши *Code*.
12. Внести зміни, щоб функції передавалась змінна, задана наперед. В функції повинна бути перевірка – коли значення змінної більше 25, то повернути значення кореня зі змінної, а коли менше або дорівнює – куб змінної.
13. Зберегти даний файл у своїй папці з ім’ям lesson_4_4, а як тип вибрати *PHP*.
14. Натиснути *Run.* Передивившись результат повернутись до коду натиснувши *Code*.
15. Внести зміни, щоб функція заповнювала комірки масиву квадратами номеру комірки, в циклі заповнити масив з 10 елементів, вивести заповнений масив на екран в стовпчик з відображенням номеру комірки. (В циклі використовувати створену функцію)

16. Зберегти даний файл у своїй папці з м.’ям lesson_4_5, а як тип вибрати *PHP*.
17. Натиснути *Run.* Передивившись результат повернутись до коду натиснувши *Code*.

Вміст звіту: Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, коди сторінок створених відповідно до завдання п. 12, 15.

Контрольні запитання

1. Яка структура функції в *PHP* ?
2. Що таке глобальны змінні, як їх використовувати в функції ?
3. Призначення *return*.
4. Як визвати створену функцію ?

Лабораторна робота №5

Глобальні масиви

Мета роботи: Навчитись обробляти інформацію масивів *POST*, *GET*, *SERVER*.

Теоретичні відомості

Масив *POST* призначений для зберігання даних отриманих з форми. В атрибуті *method* тегу *form* необхідно вказати метод обробки інформації.

<form method=”POST” action =”name.php”>

Атрибут *action* призначений для вказівки на обробник, до якого звертається форма. Після виконання обробником дій по роботі з даними формами, він повертає новий *html*-документ. В якості обробника може виступати адреса електронної пошти, тоді необхідно після лапок вказати *mailto*: . При відправці форми буде запущена поштова програма.

Як приклад використання масиву *POST* може бути створення двох полів для логіну та паролю

<input type=”text” name=”login”>

<input type=”password” name=”pswd”>

<input type=”submit” name=”scan” value=”Войти” >

type=”*submit*” – кнопка, що відправляє інформацію

Далі в *PHP* коді необхідно прописати

echo $login=$_POST[‘login’];

echo $ pswd =$_POST[‘pswd’];

Масив *GET* призначений для зберігання даних отриманих з рядка адреси. В атрибуті *method* тегу *form* необхідно вказати метод обробки інформації

<form method=”GET” action =”name.php”>

Як приклад використання масиву *GET* може бути створення двох полів для логіну та паролю

<input type=”text” name=”login”>

<input type=”password” name=”pswd”>

<input type=”submit” name=”scan” value=”Войти” >

Масив *SERVER* зберігає данні, що отримані з сервера, вивести вміст масиву:

print_r ($_SERVER)

Порядок виконання роботи

1. Відкрити *PHP*-дизайнер.
2. Із запропонованого списку вибрати *html*-файл.
3. Між тегами *body* записати наступний код:

```
<form method="POST" action=" lesson_5_1.php">
Логін   <input type="text" name="login"><br>
Пароль <input type="password" name="pswd"><br>
<input type="submit" name="scan" value="Войти" >
</form>
<?php
        echo $login=$_POST[‘login’];
        echo $ pswd =$_POST[‘pswd’];
?>
```

4. Зберегти даний файл у своїй папці з ім’ям lesson_5_1, а як тип вибрати *PHP*.
5. Натиснути *Run.* Ввести в текстові поля довільні логін та пароль, та натиснути на кнопку з написом «Войти». Повернутись до коду натиснувши *Code*.
6. Внести зміни до коду

```
<form method="GET" action=" lesson_5_2.php">
<H1>Введіть группу</H1>
<input type="text" name="id"><br>
<input type="submit" name="scan" value="Войти" >
</form>
<?php
?>
```

7. Зберегти даний файл у своїй папці з ім’ям lesson_5_2, а як тип вибрати *PHP*.
8. Натиснути *Run.* Ввести в текстове поле *LA* та натиснути на кнопку з написом «Войти». Переглянути запис в рядку адреси. Повернутись до коду натиснувши *Code*.
9. Внести зміни до коду

```
</form>
<?php
If ($_GET[‘id’]==LA)
{
        echo “Автоматизація”;
        }
        Else
        {
        echo “Інша спеціальність”;
}
?>
```

10. Зберегти даний файл у своїй папці з ім’ям lesson_5_3, а як тип вибрати *PHP*.
11. Натиснути *Run.* Ввести в текстове поле *LA* та натиснути на кнопку з написом «Войти». Повернутись до коду натиснувши *Code*.
12. Внести зміни до коду, щоб заповнити декілька раз текстове поле коду групи переглянути вміст масиву *GET.*
13. Зберегти даний файл у своїй папці з ім’ям lesson_5_4, а як тип вибрати *PHP*.
14. Натиснути *Run.* Переконатись в правильності виконаного завдання. Повернутись до коду натиснувши *Code*.
15. Внести зміни до коду, щоб заповнити декілька раз текстове поле логіну можна було переглянути вміст масиву *POST.*

16. Зберегти даний файл у своїй папці з ім’ям lesson_5_5, а як тип вибрати *PHP*.
17. Натиснути *Run.* Переконатись в правильності виконаного завдання. Повернутись до коду натиснувши *Code*.

Вміст звіту: Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, коди сторінок створених відповідно до завдання 12, 15.

Контрольні запитання

1. Призначення масиву *POST* ?
2. Призначення масиву *GET* ?
3. Призначення масиву *SERVER* ?

Лабораторна робота №6

Константи. *Cookie*. Сесії. Коментарі

Мета роботи: Навчитись створювати константи, робота з сесіями, обробка *cookie.*

Теоретичні відомості

Досить часто стоїть задача розміщення коментаря для фрагменту коду (рядка, операції) веб-документу, щоб було зрозуміло який блок дій відбувається на даному етапі. При виконанні скрипта, коментар не виводиться.

Коментарі бувають одно- та багаторядкові.

Однорядковий коментар:

// Виводимо значення змінної *a*

echo $a;

Багаторядковий коментар:

/* Виводимо значення змінної *a*,

що відображає суму *x* та *y* */

echo $a;

Для уникнення невизнання символів кирилиці необхідно в тегу <*head*> прописати:

<meta charset="utf-8">

Константи призначені для зберігання значень, що в них внесено та без їх зміни.

define (“a”, “10”);

a – ім’я константи, 10 - значення

Вивід значення константи на екран:

echo a; – без символу $.

Значення константи може бути використано в подальшому як число так і текст

$b=a

echo $b+10; – результат 20

$b=a

echo $b."10"; – результат 10Hello

Не обов'язковим є третій параметр *define*("*a*", "10", *true*/*false*) – вказує на регістр *true* – регістр враховується, а *false* – регістр не враховується.

Cookie зберігають інформацію за визначений період

Створення:

setcookie("ім'я ", "значення")

Вивести на екран виконується за допомогою масиву *COOKIE*[], в якості номера комірки вказується ім'я *cookie*.

echo $_COOKIE['ім'я'];

Існує можливість зберігати *cookie* певний час:

setcookie("ім'я ", "значення", time()+час);

час визначається в секундах

Сесії (*session*) – дозволяє зберігати велику інформацію, при закритому браузері знищується. Містить декілька функцій:

session_start(); – старт сесії,

session_destroy(); – видалення сесії,

unset ($_session['ім'я']); – видалення елементу сесії.

Порядок виконання роботи

1. Відкрити *PHP*-дизайнер.
2. Із запропонованого списку вибрати *html*-файл.
3. Між тегами *body* записати наступний код:

```
<form method="POST" action=" lesson_6_1.php">
Логін   <input type="text" name="login"><br>
Пароль <input type="password" name="pswd"><br>
<input type="submit" name="scan" value="Войти" >
</form>
<?php
    /* Виводимо значення константи login,
    що відображає  логін  */
```

```
define (“login”, $_POST[‘login’];);
echo login;
?>
```

4. Зберегти даний файл у своїй папці з ім’ям lesson_6_1, а як тип вибрати *PHP*.
5. Натиснути *Run.* Ввести в текстові поля довільні логін та пароль, та натиснути на кнопку з написом «Войти». Повернутись до коду натиснувши *Code*.
6. Внести зміни до коду, щоб константа відображала пароль
7. Зберегти даний файл у своїй папці з ім’ям lesson_6_2, а як тип вибрати *PHP*.
8. Натиснути *Run.* Ввести в текстові поля довільні логін та пароль, та натиснути на кнопку з написом «Войти». Повернутись до коду натиснувши *Code*.
9. Внести зміни до коду, в саму початку документа (над <!DOCTYPE HTML>) прописати

```
<?php
setcookie("name","123");
?>
```

Між тегами *body* записати наступний код:

```
<?php
        echo $_COOKIE['name'];
?>
```

10. Зберегти даний файл у своїй папці з ім’ям lesson_6_3, а як тип вибрати *PHP*.
11. Натиснути *Run.* Продивитись результат. Повернутись до коду натиснувши *Code*.
12. Внести зміни до коду, щоб *cookie*, яка зберігається 1 годину, заносилися до константи та вивести константу на екран з урахуванням регістру

13. Зберегти даний файл у своїй папці з ім’ям lesson_6_4, а як тип вибрати *PHP*.
14. Натиснути *Run*. Продивитись результат. Повернутись до коду натиснувши *Code*.
15. Внести зміни до коду

```
<?php
        if (isset($_POST['scan']))
        {
                $log=$_POST['login'];
        }
        if (!isset($log))
        {
        ?>
<form method="POST" action="2.php">
Логін   <input type="text" name="login"><br>
Пароль <input type="password" name="pswd"><br>
          <input type="submit" name="scan" value="Войти" >
</form>
<?php
        }
        else
        {
        echo "Логин - ".$log;
        }
?>
```

16. Зберегти даний файл у своїй папці з ім’ям lesson_6_5, а як тип вибрати *PHP*.
17. Натиснути *Run*. Ввести довільний логін та пароль та натиснути «Войти». Повернутись до коду натиснувши *Code*.

18. Внести зміни до коду, щоб з нового рядка виводився введений пароль
19. Зберегти даний файл у своїй папці з ім’ям lesson_6_6, а як тип вибрати *PHP*.
20. Натиснути *Run*. Продивитись результат. Повернутись до коду натиснувши *Code*.
21. Внести зміни до коду

```
<?php
session_start();
if (isset($_POST['scan']))
{
   $_SESSION['gr']=$_POST['gr'];
}
if (!isset($_SESSION['gr']))
{
?>
<form method="POST" action="2.php">
Група  <input type="text" name="gr"><br>
          <input type="submit" name="scan" value="Войти" >
</form>
<?php
}
else
{
echo "Група - ".$_SESSION['gr'];
}
?>
```

22. Зберегти даний файл у своїй папці з ім’ям lesson_6_7, а як тип вибрати *PHP*.

23. Натиснути *Run*. Ввести назву групи та натиснути «Войти». Повернутись до коду натиснувши *Code*.
24. Внести зміни до коду, щоб видаляти елемент сесії після її виконання.
25. Зберегти даний файл у своїй папці з ім’ям lesson_6_8, а як тип вибрати *PHP*.
26. Натиснути *Run*. Ввести назву групи та натиснути «Войти». Повернутись до коду натиснувши *Code*.

Вміст звіту: Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, коди сторінок створених відповідно до завдання 6, 12, 18, 24.

Контрольні запитання

1. Поняття константи.
2. Поняття *Cookie*.
3. Створення сесії – призначення, засоби.

Лабораторна робота №7

PhpMyAdmin

Мета роботи: Створення бази даних за допомогою *phpMyAdmin.*

Теоретичні відомості

Для зберігання великої кількості інформації на сайті необхідно створити базу даних. За допомогою *PHP* інформація буде виводитися на екран за відповідним запитом.

При підключенні до бази даних необхідно прописати *localhost/phpmyadmin/* на запит логін вказати *root*, а пароль 1234. Відкриється вікно приведена на рис.1.

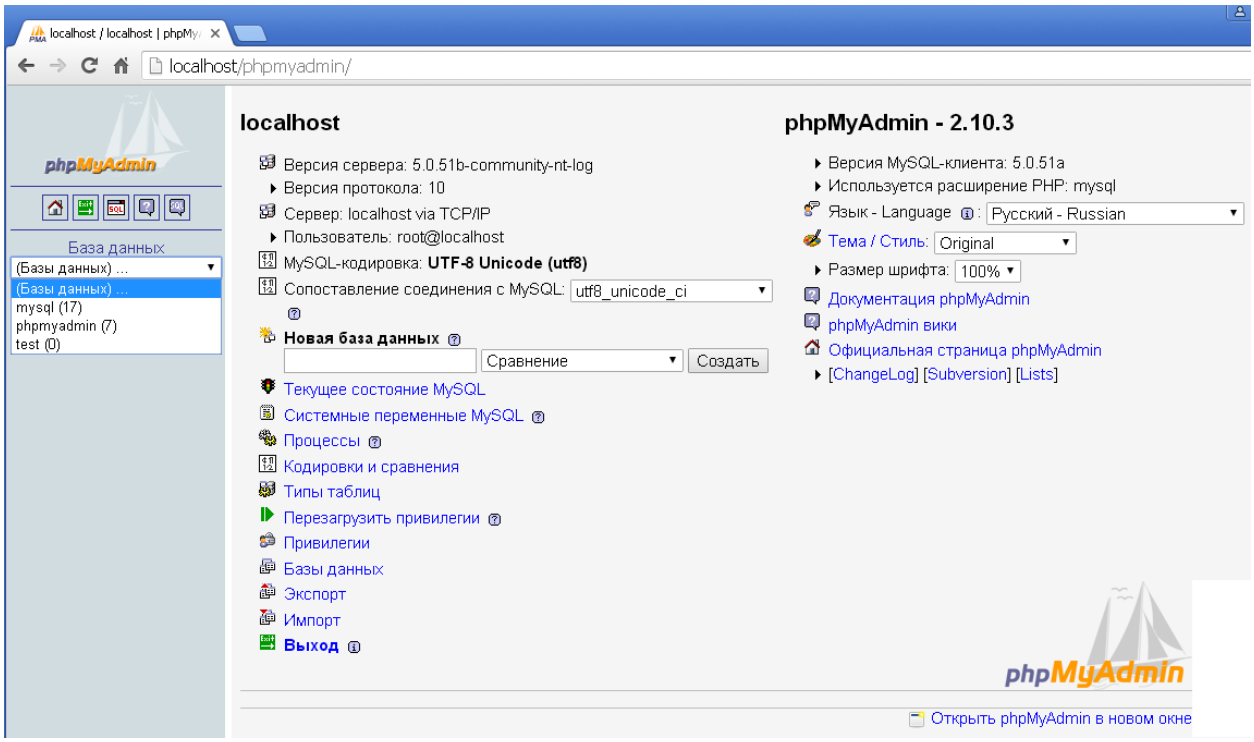


Рис. 3. Вікно *phpMyAdmin.*

Зліва приведений список існуючих баз даних. Необхідно перевірити щоб *MYSQL*-кодування було *UTF*-8/

При створені нової БД в полі «Новая база данных» прописують назву та натискають «создать». Новостворена база даних буде мати на початку 0 таблиць, для створення нової таблиці вказують «Имя», «Количество полей» та натискають «ОК»

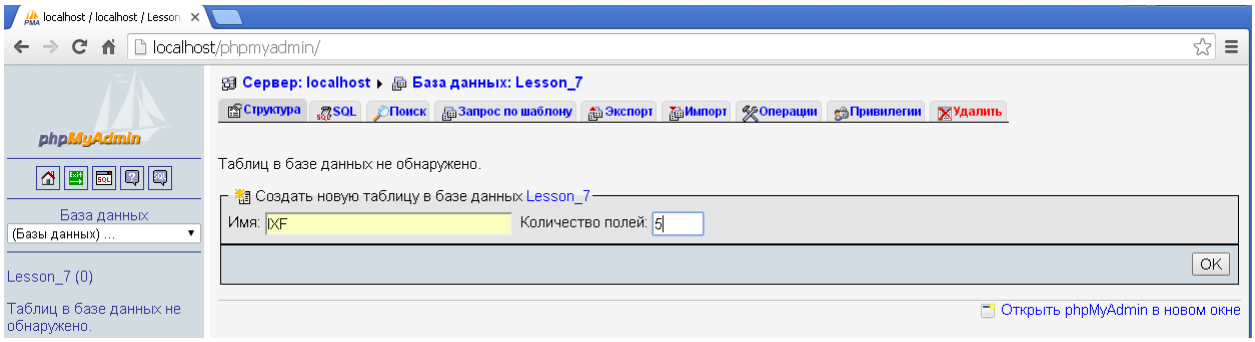


Рис. 4. Створення нової таблиці в базі даних.

В подальшому заповнюють поля в таблиці, перше поле називають *id*, як тип вказують *int*(*integer*).

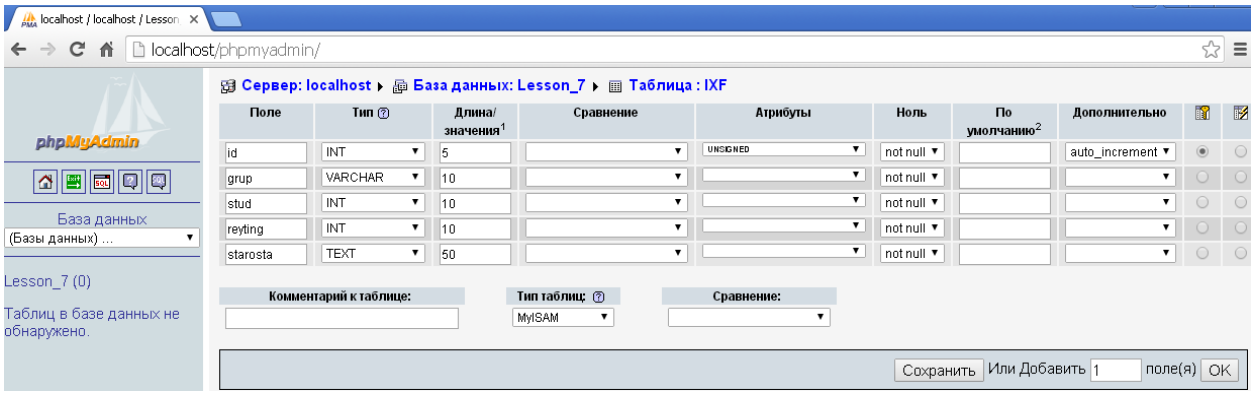


Рис. 5. Приклад заповнення полів.

В PHP міститься ряд функцій – роботою з базою даних:

mysql_connect(“localhost”, “root”, “1234”);– підключення до бази даних;

mysql_connect(“name”);– до якої бази підключитися конкретно;

mysql_close();– відключення від бази даних;

mysql_query(“SELECT * FROM ім’я ”);– запит для вибору, ім’я – вказати яка з таблиць.

Порядок виконання роботи

1. Відкрити браузер та прописати *localhost/phpmyadmin/*
2. Ввести логін *root* та пароль 1234.
3. Вибрати створити нову базу даних, як ім’я вибрати *lesson7.*
4. Створити нову таблицю з назвою групи, кількість полів – 5.
5. Перше поле заповнити згідно теоретичних відомостей, а чотири інші довільним чином
6. Відкрити *PHP*-дизайнер.

7. Із запропонованого списку вибрати *html*-файл.
8. Між тегами *body* записати наступний код:

```
<?php
Function connect()
{
   $conect = mysql_connect('localhost', 'root', '1234');
   return $conect;
}
if (connect())
{
  echo "Ви підключені";
}
?>
```

9. Зберегти даний файл у своїй папці з ім’ям lesson_7_1, а як тип вибрати *PHP*.
10. Натиснути *Run*. Передивитись результат. Повернутись до коду натиснувши *Code*.
11. Внести зміни до коду, щоб за запитом виводилось на екран вміст довільного поля
12. Зберегти даний файл у своїй папці з ім’ям lesson_7_2, а як тип вибрати *PHP*.
13. Натиснути *Run*. Продивитись результат. Повернутись до коду натиснувши *Code*.

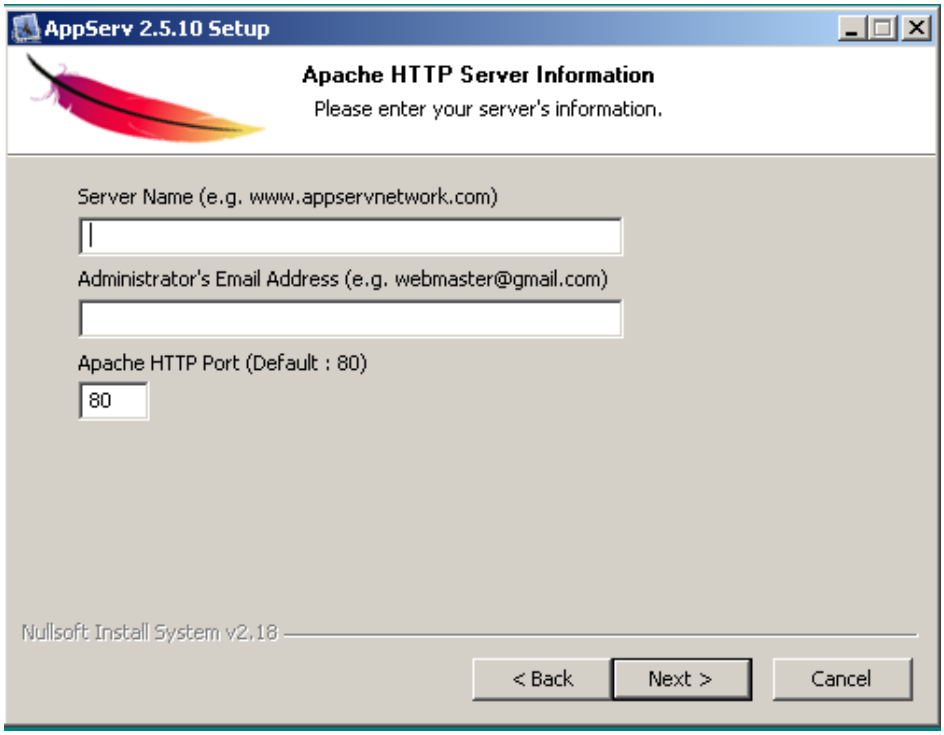
Вміст звіту: Звіт повинен містити назву роботи, мету, короткі теоретичні відомості, коди сторінок створених відповідно до завдання 11.

Контрольні запитання

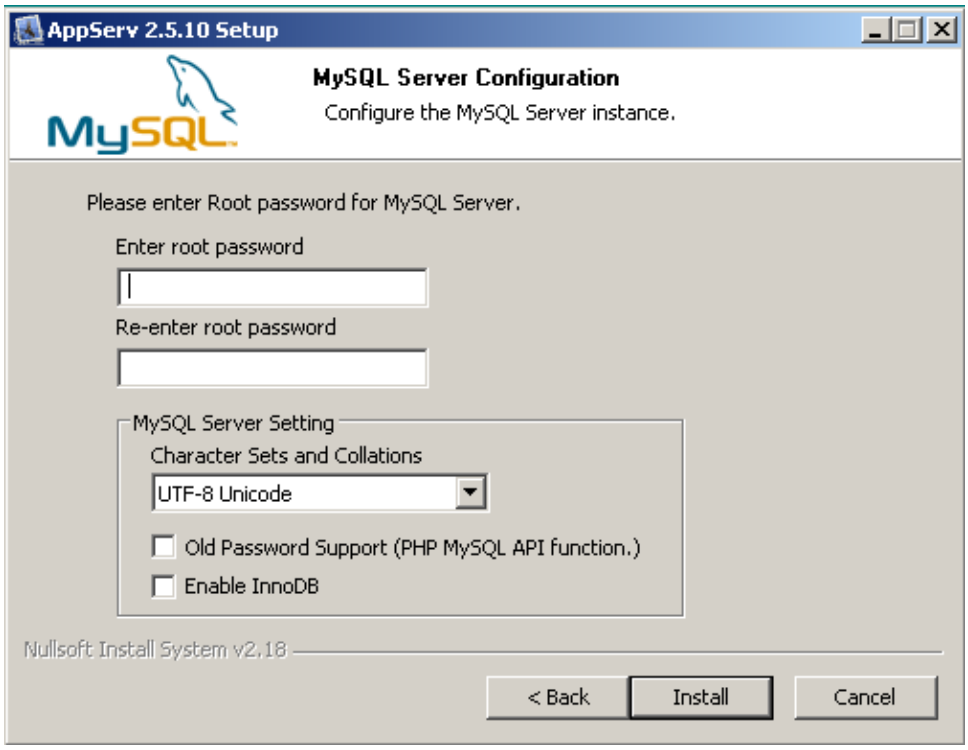
1. Як створити базу даних за допомогою *phpmyadmi* ?
2. Що таке «поле» ?
3. Що таке «запит»?

Додаток. Встановлення локального серверу *Apache*

1. Завантажити файл, дати згоду на ліцензію та вибрати директорію (C:\AppServ)
2. Вказати локальний хост (*localhost*) в першому рядку

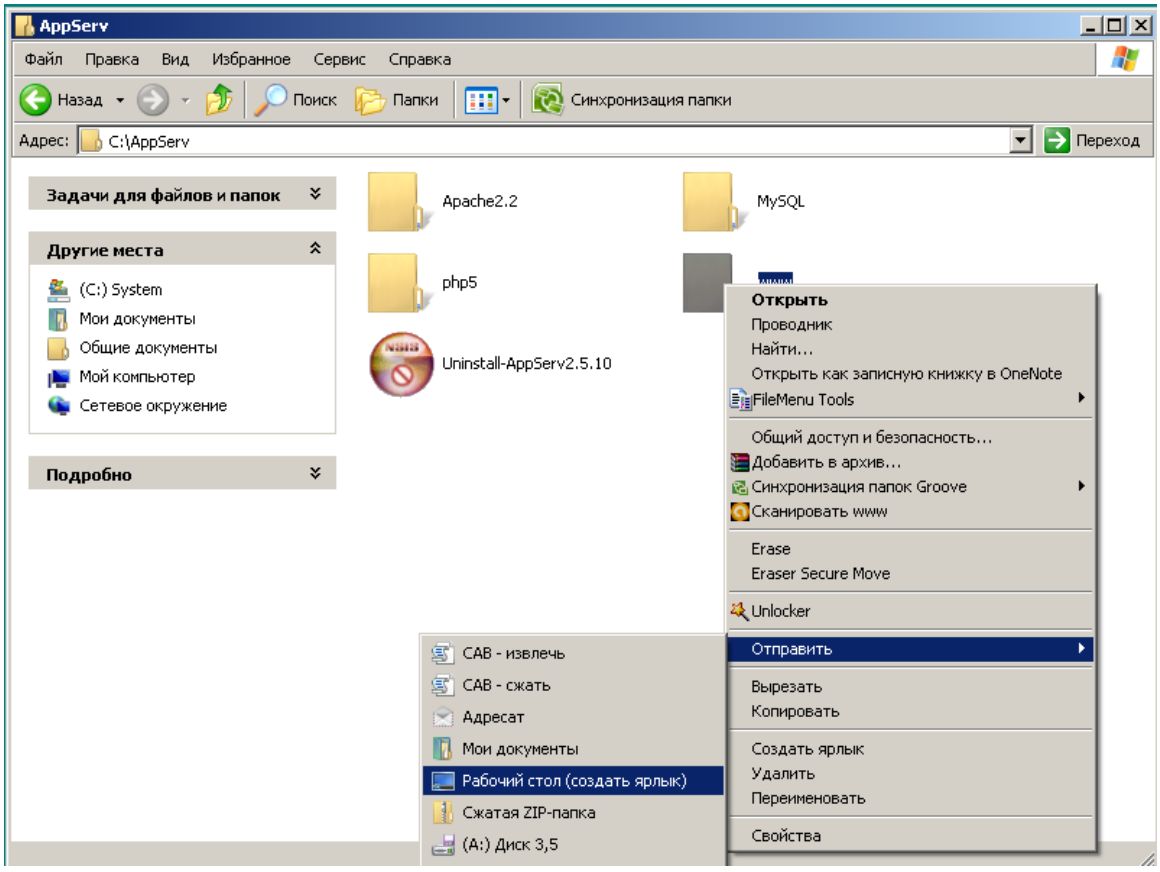


3. В другому рядку вказати поштову скриньку
4. Задати пароль (рекомендовано для навчання 1234)

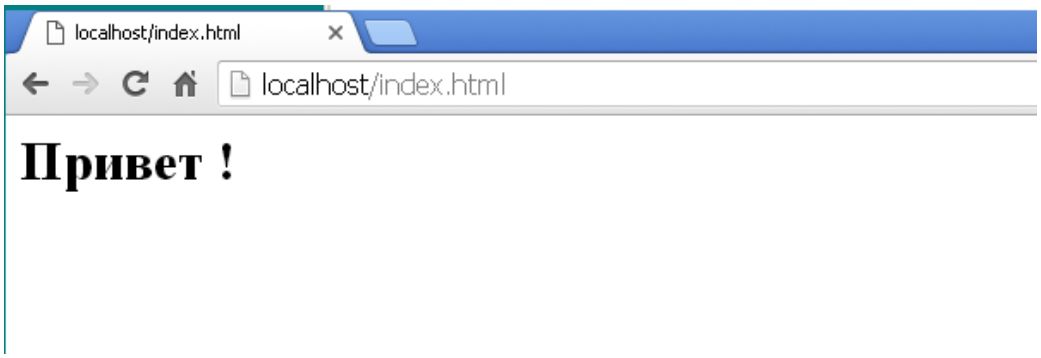


5. Вибрати *instal*

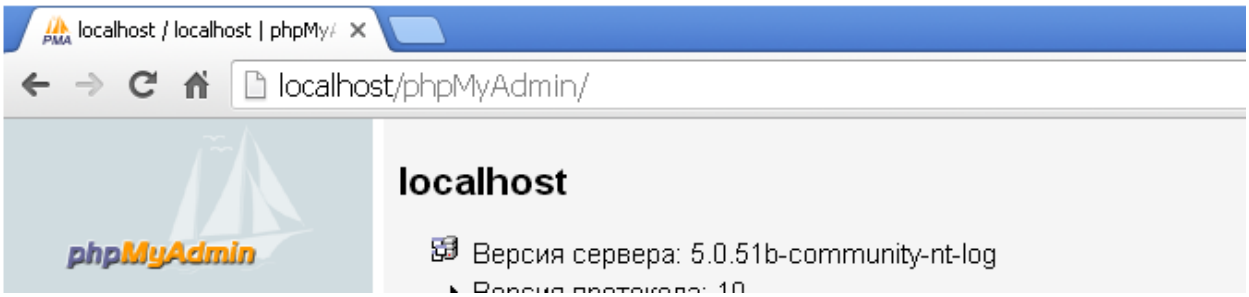
6. Для папки веб-проектів *WWW* створити на робочому столі ярлик



7. Для роботи з веб-документами (перегляду результату роботи) за допомогою локального сервера в браузері необхідно вказати *localhost /* та ім’я документа



8. Для входу до *phpMyAdmin* необхідно вказати localhost/phpMyAdmin/, ввести ім’я користувача (*root*) та пароль



Список рекомендованої літератури

1. Котеров Д.В. Самоучитель PHP 4. – СПб.: БХВ-Петербург, 2003. – 567с.

2. Спейнауєр С., Єкштейн Р. Справочник вебмастера. – Пер. с англ.. – СПб: Символ-Плюс, 2011. – 608с.

3. Харрис Э. PHP/MySQL для начинающих/ Пер. с англ. – М. КУДИЦ_ОБРАЗ, 2005 – 384с.