

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе №1 по дисциплине

ТЕХНОЛОГИИ ИНТЕРНЕТ-ПРОГРАММИРОВАНИЯ

на тему

«Основы протокола *HTTP*. Установка веб-сервера. Основы *PHP*»

Выполнил ст. гр. 820601
Проверил преп. каф. ИТАС

А.Р. Шведов
А.Л. Гончаревич

Минск 2022

СОДЕРЖАНИЕ

Введение	3
1 Постановка задачи	4
2 Теоретическая часть	5
3 Ход работы	7
Заключение.....	12

ВВЕДЕНИЕ

PHP — язык программирования, который наиболее распространён в сфере веб-разработки. Язык *PHP* работает на удаленном сервере, поэтому он и называется серверный язык программирования.

Любой скрипт *PHP* состоит из последовательности операторов. Оператор может быть присваиванием, вызовом функции, циклом, условным выражением или пустым выражением. Операторы обычно заканчиваются точкой с запятой. Также операторы могут быть объединены в группу заключением группы операторов в фигурные скобки. Группа операторов также является оператором.

Интернет — это множество компьютеров по всему миру, соединённых между собой проводами в единую сеть. Все компьютеры делятся на две большие группы: клиенты и сервера. Клиенты инициируют запросы на сервера, а те, в свою очередь, их принимают, обрабатывают и отправляют клиенту ответ.

PHP позволяет решить множество задач связанных с клиент-серверной архитектурой, например:

1 С помощью *HTML* можно только создать форму. А обработать то, что ввёл пользователь, может лишь *PHP*.

2 Если Вы делаете блог на чистом *HTML*, то на каждую статью требуется создавать новый файл. Добавлять и редактировать записи придётся вручную. *PHP* позволяет обойтись с помощью одного файла, а статьи хранить в базе данных. Благодаря этому, можно сделать админку, из которой можно будет добавлять и редактировать контент.

3 *PHP* позволяет реализовать механизм авторизации на сайте.

1 ПОСТАНОВКА ЗАДАЧИ

Изучить семантику, синтаксис и возможности языка *PHP*. Изучение базового синтаксиса в языке *PHP*, работу с переменными, разными типами данных, операциями сравнения и логическими операциями. Ознакомление с основами серверных технологий.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

HTTP — это протокол, позволяющий получать различные ресурсы, например *HTML*-документы. Протокол *HTTP* лежит в основе обмена данными в Интернете. *HTTP* является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем, обычно веб-браузером (*web-browser*). Полученный итоговый документ может состоять из различных поддокументов, являющихся частью итогового документа: например, из отдельно полученного текста, описания структуры документа, изображений, видео-файлов, скриптов и многого другого.

Клиенты и серверы взаимодействуют, обмениваясь одиночными сообщениями (а не потоком данных). Сообщения, отправленные клиентом, обычно веб-браузером, называются запросами, а сообщения, отправленные сервером, называются ответами.

PHP — это широко используемый язык сценариев общего назначения с открытым исходным кодом.

Говоря проще, *PHP* это язык программирования, специально разработанный для написания *web*-приложений (сценариев), исполняющихся на *Web*-сервере. Аббревиатура *PHP* означает «*Hypertext Preprocessor* (Препроцессор Гипертекста)». Синтаксис языка берет начало из *C*, *Java* и *Perl*. *PHP* достаточно прост для изучения. Преимуществом *PHP* является предоставление *web*-разработчикам возможности быстрого создания динамически генерируемых *web*-страниц.

Значительным отличием *PHP* от какого-либо кода, выполняющегося на стороне клиента, например, *JavaScript*, является то, что *PHP*-скрипты выполняются на стороне сервера. Вы даже можете сконфигурировать свой сервер таким образом, чтобы *HTML*-файлы обрабатывались процессором *PHP*, так что клиенты даже не смогут узнать, получают ли они обычный *HTML*-файл или результат выполнения скрипта.

Приведем схему работу сервера *PHP*:

Apache передаёт *PHP*-интерпретатору файл на выполнение;

PHP видит, что клиент запросил пятнадцатую статью. А все записи, которые есть в блоге, хранятся на сервере в базе данных. Поэтому *PHP*-интерпретатор делает запрос к *MySQL*; *MySQL* возвращает текст статьи;

PHP-интерпретатор подставляет ответ базы данных в определённый *html*-шаблон и завершает свою работу. На этом обработка запроса окончена и в дело снова вступает *Apache*.

Финальный этап. *Apache* отправляет готовый ответ клиенту. Схема клиент-серверного взаимодействия представлена на рисунке 2.1

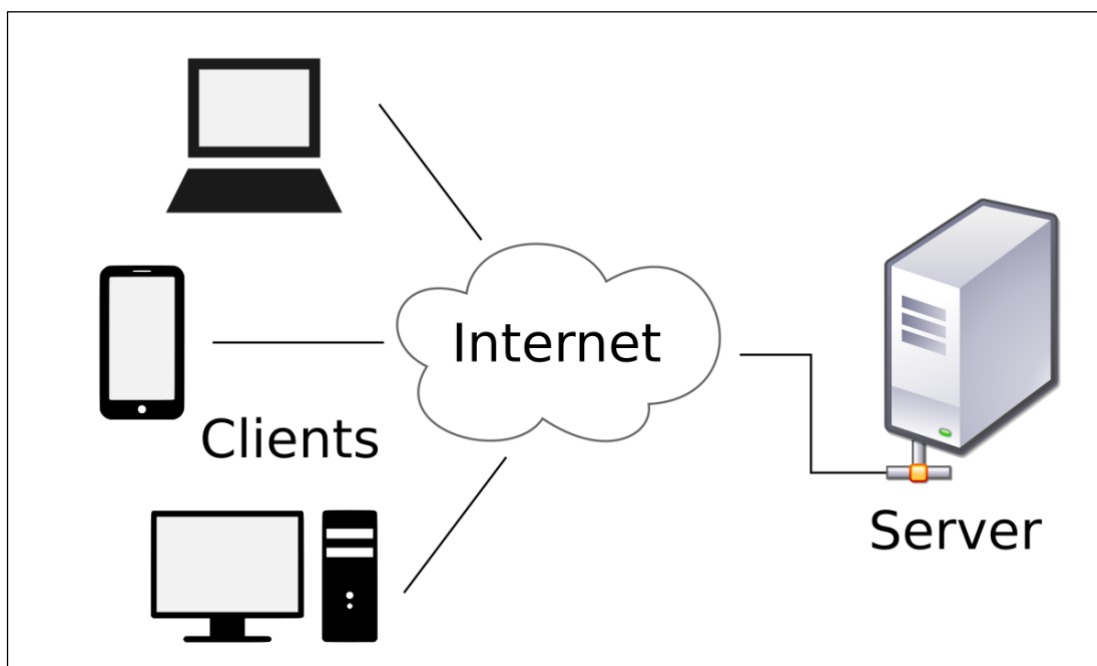


Рисунок 2.1 — Клиент-серверная архитектура

3 ХОД РАБОТЫ

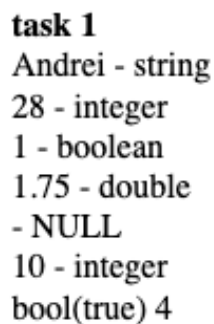
Рассмотрим практическую реализацию операторов, приведенных в цели данной работы. Работа выполняется с помощью редактора кода — *VSCode*.

С помощью оператора *echo* выведите на страницу переменные различных типов данных. На рисунке 3.1 приведен результат работы скрипта. Фрагмент кода:

```
echo "<b> task 0 </b> <br>";
echo 'Hello World';
echo "<br>" . "<br>";
echo "<b> task 1 </b> <br>";

// Declare variables
$name = "Andrei"; // string
$age = 28; // integer
$isMale = true; // bool
$height = 1.75; // float
$children = null; // null
define('MARK', 10); // const

// Print the variables and their types.
echo $name . ' - ' . gettype($name) . '<br>';
echo $age . ' - ' . gettype($age) . '<br>';
echo $isMale . ' - ' . gettype($isMale) . '<br>';
echo $height . ' - ' . gettype($height) . '<br>';
echo $children . ' - ' . gettype($children) . '<br>';
echo MARK . ' - ' . gettype(MARK) . '<br>';
```



```
task 1
Andrei - string
28 - integer
1 - boolean
1.75 - double
- NULL
10 - integer
bool(true) 4
```

Рисунок 3.1 — Вывод переменных различных типов данных

Повторим вывод, заключив переменные в двойные и одинарные кавычки. Т.к. *PHP* обрабатывает переменные только в строках с двойными кавычками, можно видеть, что вывод в одинарных кавычках показал те же символы, которые были в файле *.php*.

Так же стоит заметить, что используемая функция *echo* не вывела переменные со значением *NULL* и типом *bool*, потому что *NULL* выводится как пустой символ, а *echo bool* лишь возвращает «1» как код результата. На рисунке 3.2 приведён результат вывода для заданий 2 и 3. Фрагмент кода:

```
echo "<b> task 2 </b> <br>";  
echo "$name" . '<br>';  
echo "$age" . '<br>';  
echo "$isMale" . '<br>';  
echo "$height" . '<br>';  
echo "$children" . '<br>';  
echo "MARK" . '<br>';
```

```
echo "<b> task 3 </b> <br>";  
echo '$name' . '<br>';  
echo '$age' . '<br>';  
echo '$isMale' . '<br>';  
echo '$height' . '<br>';  
echo '$children' . '<br>';  
echo '$MARK' . '<br>';
```

task 2 Andrei 28 1 1.75 MARK task 3 \$name \$age \$isMale \$height \$children \$MARK
--

Рисунок 3.2 — Вывод переменных с двойными и одинарными кавычками

В задании 4 необходимо вывести на экран любое четверостишие. Для каждой новой строки отдельный оператор *echo*. Заметим, что функция *nl2br()* является вспомогательной и всего лишь добавляет перенос строки. На рисунке 3.3 отображён вывод. Фрагмент кода:

```
$a = "Выведите на экран любое четверостишие. Для каждой новой  
строки используйте отдельный оператор echo.\n";  
$b = "Каждая строчка должна быть отдельной строковой  
переменной.\n Также необходимо использовать переводы строки.\n";  
$c = "После четверостишия поставьте инициалы автора и сделайте их  
подчёркнутыми.\n";  
$d = "Someone A.P.";   
echo nl2br($a . $b . $c . $d);
```

task 4

Выведите на экран любое четверостишие. Для каждой новой строки используйте отдельный оператор *echo*. Каждая строчка должна быть отдельной строковой переменной. Также необходимо использовать переводы строки. После четверостишия поставьте инициалы автора и сделайте их подчёркнутыми.
Someone A.P.

Рисунок 3.3 — Вывод четверостишия с подписью автора

В задании 5 необходимо выполнить задание 4 одним оператором *echo*. На рисунке 3.4 отображён вывод. Фрагмент кода:

```
echo nl2br("Выведите на экран любое четверостишие. Для каждой  
новой строки используйте отдельный оператор echo.  
Каждая строчка должна быть отдельной строковой переменной.  
Также необходимо использовать переводы строки.  
После четверостишия поставьте инициалы автора и сделайте их  
подчёркнутыми.  
<i> Someone A.P. </i>");
```

task 5

Выведите на экран любое четверостишие. Для каждой новой строки используйте отдельный оператор *echo*. Каждая строчка должна быть отдельной строковой переменной. Также необходимо использовать переводы строки. После четверостишия поставьте инициалы автора и сделайте их подчёркнутыми.
Someone A.P.

Рисунок 3.4 — Вывод четверостишия с подписью автора

В задании 6 необходимо попробовать в выражении использовать разные типы данных, например, сложить число «10» и строку «20 приветов». В таком случае браузер покажет ошибку из-за складывания переменных различных типов, но из-за того, что *PHP* является слабо типизированным языком, переменные динамически преобразованы. Таким образом интерпретатор видит число «20» в строке и складывает его с «10», то есть вывод равен «30».

На рисунке 3.5 отображён вывод результатов. Фрагмент кода:

```
echo "<b> task 6 </b> <br>";
echo 10 + "20 приветов" . '<br>';
echo 5 + NULL . '<br>';
echo $age + $height . '<br>';
```

<p>task 6</p> <p>Warning: A non-numeric value encountered in /Users/andreishvedau/OneDrive - Admiral Markets AS/University/ТИ-нерПч.2/Lab1/hello.php on line 76</p> <p>30</p> <p>5</p> <p>29.75</p>

Рисунок 3.5 — Вывод сложения разных типов данных

Оператор *XOR* «исключающее или» возвращает значение *true*, если один и только один из операндов имеет значение *true*. Если оба операнда имеют значение *true*, оператор вернет значение *false*. Так как приоритет оператора *XOR* такой же как и у операторов *AND* и *OR* (ниже чем у оператора присваивания), и он используется в выражении с присваиванием, то сначала он вычисляет и возвращает значение левого операнда. Именно поэтому во фрагменте кода использованы кавычки.

На рисунке 3.6 приведем результат выполнения скрипта, который выводит значения операций со всеми возможными вариантами операндов (4 варианта) согласно заданию 7. Фрагмент кода:

```
$a = (false xor false);
$b = (true xor true);
$c = (false xor true);
$d = (true xor false);
var_dump($a,$b,$c,$d);
```

task 7
bool(false) bool(false) bool(true) bool(true)

Рисунок 3.6 — Вывод значений оператора *XOR*

Напишем скрипт по замене значений двух переменных местами. Этого можно добиться двумя способами, которые показаны во фрагменте кода. Первый способ — использовать приведение типов и функции *list()*, *array()*. Второй — математический. Результат выполнения программы приведен на рисунке 3.7. Фрагмент кода:

```
echo "<b> task 8 </b> <br>";
$x = 10;
$y = 15;
echo "$x, $y" . '<br>';

// var 1
list($x,$y) = array($y,$x);
echo "$x, $y" . '<br>';

// var 2
$x = $x + $y; $y = $x - $y; $x = $x - $y;
echo "$x, $y" . '<br>';
```

task 8
10, 15
15, 10
10, 15

Рисунок 3.7 — Результат выполнения программы

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы мною были изучены базовый синтаксиса языка *PHP*. Были рассмотрены работа с переменными, разными типами данных, операциями сравнения и логическими операциями. Так же я ознакомился с основами серверных технологий и клиент-серверной архитекту