

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчёт  
по лабораторной работе №1  
«Основы протокола *HTTP*.  
Установка веб-сервера.  
Основы *RНР*»  
по дисциплине «Технологии интернет-программирования»

Выполнил:

студент гр. 820601

Шведов А. Р

Проверил:

А. Л. Гончаревич

Минск 2022

# СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	1
ВВЕДЕНИЕ.....	3
1 ПОСТАНОВКА ЗАДАЧИ.....	4
2 Теоретическая часть.....	5
2.1 Схема клиент-серверного взаимодействия .....	5
2.2 Основы <i>RНР</i> .....	6
3 ХОД РАБОТЫ .....	8
ЗАКЛЮЧЕНИЕ .....	13

## ВВЕДЕНИЕ

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

Любой скрипт *RНР* 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.

*RНР* 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 *RНР*.

2 Если Вы делаете блог на чистом *HTML*, то на каждую статью требуется создавать новый файл. Добавлять и редактировать записи придётся вручную. *RНР* 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 *RНР* 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

### 2.1 Схема клиент-серверного взаимодействия

Сейчас мы по шагам рассмотрим, каким образом происходит общение клиента и сервера:

Всё начинается с того, что на своем компьютере в браузере мы набираем адрес какого-нибудь сайта. В этот момент происходит запрос на сервер. То, что написано в адресной строке, обычно называется *URL*.

Итак, сервер принял запрос. Для того, чтобы его обработать, на сервере должен быть установлен ряд специальных программ. Они могут иметь разные вариации, но принцип работы является одинаковым. Мы с Вами рассмотрим самую популярную связку *Apache + PHP + MySQL*.

*Apache* – самая важная часть, без которой вообще ничего не будет работать. Данная программа также называется веб-сервером, и именно она умеет принимать и анализировать запросы от клиентов. По сути, задача *Apache* состоит в том, чтобы понять, какую веб-страницу у него попросили, и в соответствии с этим сформировать ответ.

Для того, чтобы лучше понять дальнейшие действия, происходящие на сервере, необходимо подумать, а какой вообще формат данных клиент хочет получить в ответе? На это есть однозначный ответ – поскольку запрос создавался пользователем в браузере, то сервер должен вернуть *HTML*-код, так как браузер понимает именно этот формат данных.

*HTML* (англ. *HyperText Markup Language* — «язык разметки гипертекста») — стандартный язык разметки документов во Всемирной паутине. Язык *HTML* интерпретируется браузерами и отображается в виде документа в удобной для человека форме.

В связи с этим, дальнейшие действия сервера зависят от запроса клиента. Например, если *URL* выглядел следующим образом *http://site.ru/index.html*, то, скорее всего, *Apache* сразу может сформировать ответ клиенту, так как на сервере есть файл *index.html* с готовым *HTML*-кодом. Интереснее выглядит ситуация, если клиент обратился по адресу *http://site.ru/index.php*, так как это означает, что у нас нет файла с готовым *HTML*-кодом. На сервере есть файл *index.php*, в котором написан *php*-скрипт, который нет смысла возвращать клиенту в браузер, так как браузер не понимает данного языка. Поэтому, в таком случае в действие вступает следующая программа из связки – *PHP*-интерпретатор.

В итоге схема работы сервера строится из следующих действий:

*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

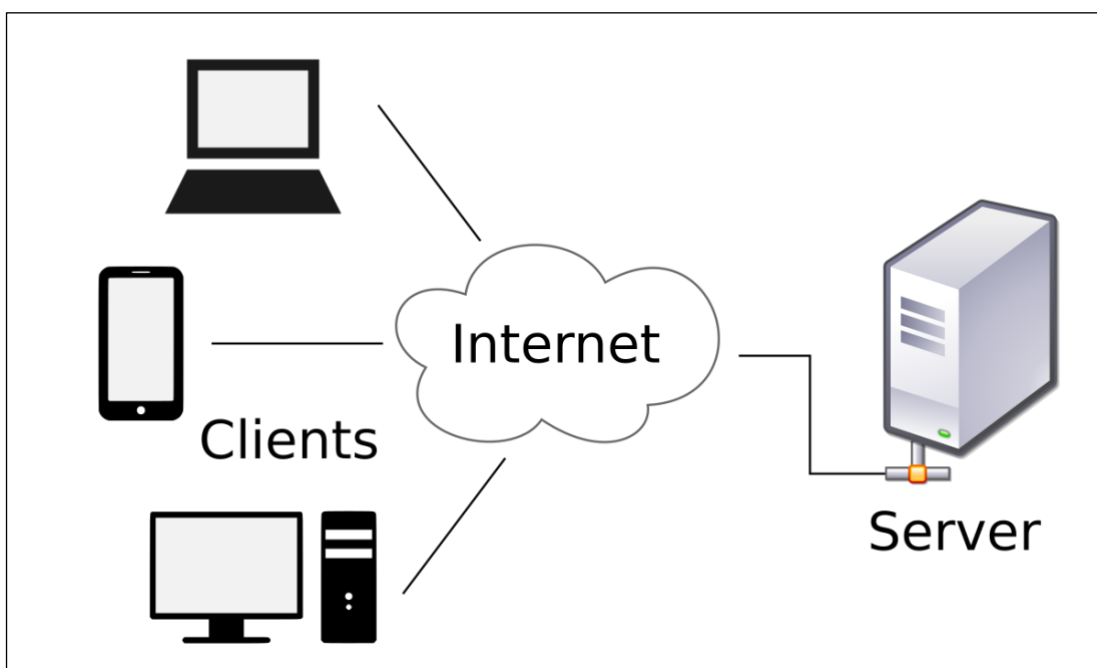


Рисунок 3.2 — Клиент-серверная архитектура

## 2.2 Основы *PHP*

*PHP* (рекурсивный акроним словосочетания *PHP: Hypertext Preprocessor*) — это распространённый язык программирования общего назначения с открытым исходным кодом. *PHP* специально сконструирован для веб-разработок и его код может внедряться непосредственно в *HTML*.

Переменная в *PHP* — это своеобразный контейнер, который может содержать определенную информацию. Для того, чтобы создать такой "контейнер", нам нужно его назвать и указать, что в нем должно "лежать".

Делается это с помощью знака "\$", который означает, что мы имеем дело с переменной. В *RHP* существует 4 простых типа переменных (целые числа, числа с плавающей точкой, строки, булевские значения) и 2 сложных типа (объекты и массивы).

*RHP* позволяет выполнять множество базовых операций над переменными:

- Арифметические операции;
- Операции сравнения;
- Логические операции.

Оператор в *RHP* — это некоторое законченное предложение на языке программирования. Любое выражение, после которого поставлен символ @;», воспринимается интерпретатором, как отдельный оператор. Таким образом, оператор состоит из одного или целого набора выражений, соединенных операциями.

### 3 ХОД РАБОТЫ

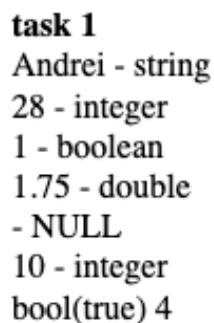
Рассмотрим практическую реализацию операторов, приведенных в цели данной работы. Работа выполняется с помощью редактора кода — *VSCode*.

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>';
```



```
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

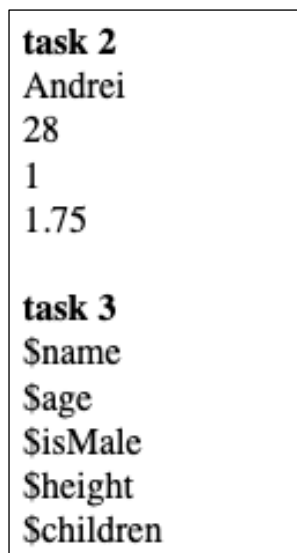


Повторим вывод, заключив переменные в двойные и одинарные кавычки. Т. к. *PHP* обрабатывает переменные только в строках с двойными кавычками, можно видеть, что вывод в одинарных кавычках показал те же символы, которые были в файле *.php*.

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



```
task 2  
Andrei  
28  
1  
1.75  
  
task 3  
$name  
$age  
$isMale  
$height  
$children
```

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:

```
echo "<b> task 4 </b> <br>";
echo "Выведите на экран любое четверостишие. Для каждой новой
строки используйте отдельный оператор echo." . '<br>';
echo nl2br("Каждая строчка должна быть отдельной строковой
переменной. \r\n Также необходимо использовать переводы строки.") . '<br>';
echo "После четверостишия поставьте инициалы автора и сделайте их
подчёркнутыми." . '<br>';
echo "<u>". 'Someone A.P.' . "</u>";
```

<p><b>task 4</b> Выведите на экран любое четверостишие. Для каждой новой строки используйте отдельный оператор <i>echo</i>. Каждая строчка должна быть отдельной строковой переменной. Также необходимо использовать переводы строки. После четверостишия поставьте инициалы автора и сделайте их подчёркнутыми. <u>Someone A.P.</u></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.
<u> Someone A.P. </u>");
```

<p><b>task 5</b> Выведите на экран любое четверостишие. Для каждой новой строки используйте отдельный оператор <i>echo</i>. Каждая строчка должна быть отдельной строковой переменной. Также необходимо использовать переводы строки. После четверостишия поставьте инициалы автора и сделайте их подчёркнутыми. <u>Someone A.P.</u></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 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>';
```

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

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 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 архитектуры.