Лабораторная работа № 1 "Основы протокола HTTP. Установка веб-сервера. Основы PHP"

1. Как работает Интернет

Прежде чем перейти к изучению языка PHP, необходимо иметь чёткое представление о том, как работает Интернет, понимать, что собой представляют сервера, и как наши компьютеры с ними взаимодействуют.
Для начала разберёмся с тем, что такое Интернет. Самое простое определение – 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.
Проще говоря, клиент – это компьютер, за которым мы сидим и в браузере просматриваем сайты. Сервер – это компьютер, на котором находится сайт. Клиент делает запрос на сервер, чтобы получить определённую веб-страницу какого-либо сайта.

Схема клиент-серверного взаимодействия

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

1. 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.

URL (англ. Uniform Resource Locator) – единообразный локатор (определитель местонахождения) ресурса. URL — это стандартизированный способ записи адреса ресурса в сети Интернет.
T.e. URL-адреса достаточно для того, чтобы мы смогли найти в сети сервер, на котором размещена интересующая нас веб-страница.

2. 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.

1) 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.

2) PHP-интерпретатор – программа, которая выполняет PHP-код. Результатом её работы обычно является HTML-код. Т.е., на данном этапе происходит преобразование php-скрипта в код, который умеет читать браузер.

В процессе своей работы PHP может использовать третью программу из связки – **MySQL**.

3) **MySQL** – свободная система управления базами данных. Данная программа отвечает за все операции с БД.

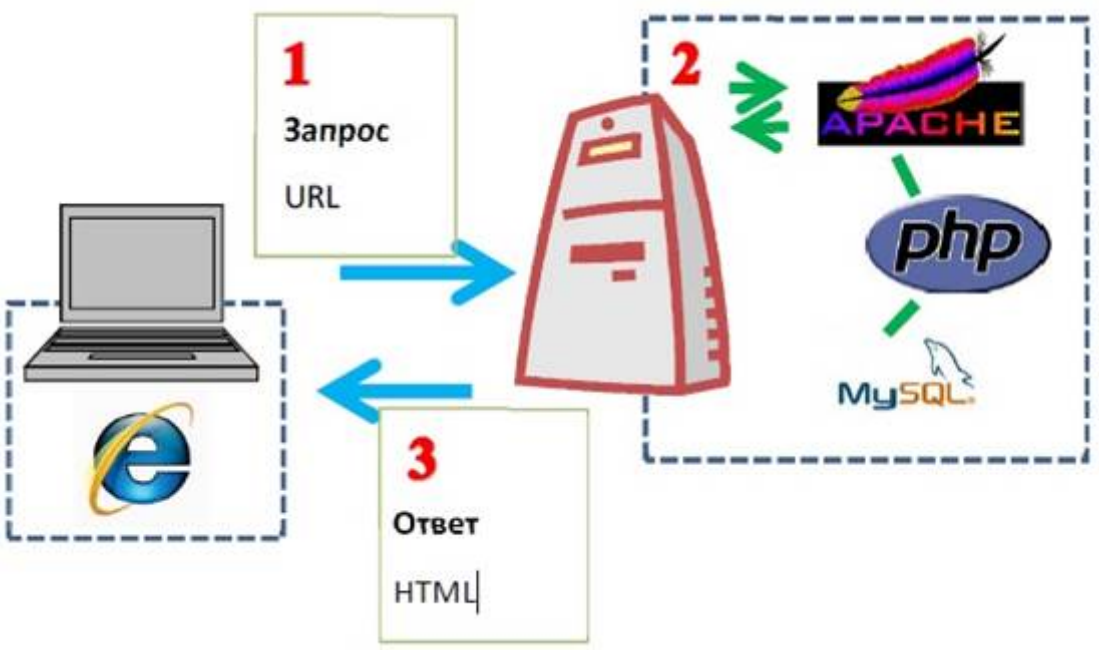
Попробуем представить себе ситуацию совместной работы PHP и MySQL. Допустим, клиент открывает в браузере какой-то блог и читает статью №15.

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.

3. Финальный этап. Apache отправляет готовый ответ клиенту.

(соотносите все шаги, которые мы рассмотрели, с цифрами на схеме ниже)



Таким образом выглядит общая схема работы Интернета. Обратите внимание на то, что когда мы приводили примеры URL, то в начале адреса постоянно фигурировала аббревиатура HTTP.

2. Протокол HTTP

HTTP (сокр. от англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов). На самом деле, общение клиента и сервера осуществляется с помощью HTTP-пакетов, т.е., и запрос, и ответ упаковываются в некую стандартную форму.
Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном ниже порядке:

1. **Стартовая строка** — определяет тип сообщения. В ней указываются общие сведения. Например, для HTTP-запроса будет указан метод, адрес документа на сервере и версия протокола. А в ответе будет указана версия протокола, код состояния и пояснение. Указание версии необходимо для того, чтобы корректно прочитать сообщение, а, например, код состояния показывает браузеру, удалось найти данную страничку или же произошла какая-нибудь ошибка.

2. **Заголовки** — характеризуют тело сообщения, параметры передачи и прочие сведения. Каждый их них представляет собой разделённую двоеточием пару параметр-значение. Заголовки определяют для браузера различные полезные настройки. Вспомните, наверняка, хотя бы раз, когда Вы заходили на какой-нибудь сайт, вместо русских букв видели там странные абракадабры. Это означало проблему с кодировками, т.е. сервер прислал нам ответ в одном формате, а браузер отображает его в другом. Чтобы не происходило такой ошибки, сервер может отправить специальный заголовок, с указанием кодировки, например:

Content-Type: text/plain; charset=cp1251

3. **Тело сообщения** — это, непосредственно, данные сообщения, т.е. сформированный html- код.

На этом мы закончим рассмотрение HTTP-сообщений. Нет никакого смысла углубляться
В их изучение, поскольку всё это автоматически формируют браузер и сервер, а не PHP-программист. На данном этапе достаточно иметь лишь общее представление о структуре HTTP-пакетов, так как иногда это позволяет избежать неприятных и головоломных ошибок в PHP.

3. Зачем нужен PHP

Пожалуй, это самый важный вопрос для начинающего программиста. Дело в том, что в ближайшие три урока мы с Вами будем изучать основы PHP, т.е. темы, которые необходимы для дальнейшего изучения языка, но совершенно не раскрывающие того, зачем он нужен и какую даёт выгоду. Вопрос это очень непростой, и пока что мы рекомендуем Вам просто поверить в то, что PHP – это здорово! Когда Вы начинали писать HTML-код и CSS, результат приходил незамедлительно: дописали пару строк – в браузере всё стало красиво отображаться. У PHP иное назначение, и результат его работы в первых уроках будет выглядеть как чёрные буквы на белом фоне. Однако без PHP невозможно решить огромное количество задач. Вот лишь некоторые из них:

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.
4. PHP умеет работать с базой данных.
5. PHP умеет работать с файлами на сервере.
6. PHP в зависимости от неких условий может генерировать различный HTML-код. Например, в зависимости от времени суток или времени года ставить на сайт различные фоновые изображения.
7. И многое другое.

Вообще, зачем нужен PHP и какую выгоду он даёт, Вам станет ясно только после 4-ого занятия. А пока что запаситесь терпением, так как основы PHP - не самая интересная тема, однако, без неё невозможно научиться программировать на данном языке.

4. Установка веб-сервера на локальный компьютер

Мы рассмотрели архитектуру клиент-серверного взаимодействия и обнаружили, что PHP выполняется на сервере. Соответственно, если Вы сейчас захотите написать свой первый PHP-скрипт, то не сможете его запустить, так как для этого нужен определённый набор программ.

Для того, чтобы превратить наш домашний локальный компьютер в полноценный веб-сервер, нам требуется установка трех программ:

1. Веб-сервера Apache
2. PHP-интерпретатора
3. Системы управления базами данных MySQL

Однако, вручную их установить и настроить достаточно непросто, поэтому лучше воспользоваться специальным решением, которое автоматизирует процесс установки веб-сервера на локальный компьютер.
Denwer - локальный сервер (Apache, PHP, MySQL, Perl и т.д.) и программная оболочка, используемые веб-разработчиками для разработки сайтов на «домашней» (локальной) Windows-машине без необходимости выхода в Интернет.
Для его установки выполните следующие шаги:

1. Перейдите на сайт http://www.denwer.ru
2. Скачайте последнюю версию дистрибутива.
3. Запустите инсталлятор и следуйте инструкциям.

5. Основы PHP

5.1 Базовый синтаксис PHP

Для начала напишем простейший PHP-скрипт.

```
<?php

echo 'Hello!';

?>
```

Теперь давайте подробнее рассмотрим, из чего же он состоит.
Все PHP-сценарии пишутся в виде блоков кода. Эти блоки при необходимости могут быть встроены в HTML и обычно определяются с помощью строки <?php в начале и ?> — в конце. Все, что вне этих идентификаторов блока, интерпретатор PHP игнорирует и передает обратно веб-серверу для отображения на стороне клиента.
Например:

```
<html>

<head><title>Первый PHP-сценарий</title></head> <body>


Пример работы php:<br/>

<?php

echo 'Hello!';

?>

</body>

</html>
```

Всё, что находится не внутри конструкции <?php ?> будет воспринято как HTML-код.

Здесь Вы также познакомились с первым оператором PHP — echo. Он представляет базовый метод PHP для отправки содержимого клиенту, т.е. выводит данные на экран.
Как и во всех Си-подобных языках, в PHP каждое предложение завершается точкой с запятой.
В PHP все, что находится между /* и */, трактуется как комментарий, используемый для пояснений в теле сценария, и игнорируется интерпретатором. Для однострочных комментариев применяются символы //, которые комментируют весь текст до конца строки. Советуем сразу начать вырабатывать привычку комментировать свой код, так как в дальнейшем это поможет избежать сложностей при его чтении.

5.2 Переменные

Переменная – это область памяти, адрес которой можно использовать для осуществления доступа к данным. Данные, находящиеся в переменной, называются значением этой переменной.
В PHP имена переменных всегда начинаются с символа $ и содержат произвольную комбинацию символов, при условии, что первый символ после $ будет буквой или знаком подчеркивания. В число допустимых символов входят заглавные и прописные латинские буквы, цифры и символ подчеркивания («_»). Теоретически переменную в PHP можно назвать и по-русски, но мы настоятельно не рекомендуем этого делать.
Измените PHP код следующий образом:

```
<?php

$x = 5;

echo $x;

?>
```

Сейчас мы объявили переменную x, в которую записали значение 5. Прочитать значение переменной мы можем, просто написав её имя.
По ходу выполнения скрипта мы также можем менять значение данной переменной. Для этого следует воспользоваться оператором присваивания, справа от которого мы напишем новое значение:

```
<?php

$x = 5;

echo $x;

$x = 10;

echo $x;

?>
```

5.3 Константы

Константы представляют собой контейнеры для данных, как и переменные, но после присваивания константе значения его уже нельзя изменить.
Константы создаются в PHP с помощью функции define():

```
define('PI', '3.14');
```

Принято записывать имена констант буквами верхнего регистра, хотя возможно выбирать любые имена, отвечающие правилам именования переменных.

5.4 Типы данных

- PHP существует 4 простых типа переменных (целые числа, числа с плавающей точкой, строчки, булевские значения) и 2 сложных типа (объекты и массивы). В этом уроке мы будем работать только с простыми типами. Переменные разных типов задаются следующим образом:

```
$x = 50;          // Целое число
$x = 50.5;        // Число с плавающей точкой
$x = 'Hello!';    // Строка
$x = true;        // Булевский (логический) тип
```

Первые два типа должны быть интуитивно понятны из жизни, а вот два других мы рассмотрим подробнее.
Сначала строки. Строка – это набор любых символов. В PHP существует два способа задания строк в двойных, либо одинарных кавычках.
Разница между ними заключается в том, что если внутри строки первого типа написать имя переменной, то интерпретатор PHP подставит значение этой переменной при результирующем выводе. В случае же со вторым типом строк (одинарные кавычки) подстановки не происходит. Для лучшего понимания напишите следующий пример и посмотрите результат.

```
<?php

$int = 100;

$one = "Значение переменной равно $int<br/>";

$two = 'Значение переменной равно $int<br/>';

echo $one;

echo $two;

?>
```

Если этот сценарий исполнить, его вывод будет таким:

```
Значение переменной равно 100

Значение переменной равно $int
```

- теперь рассмотрим булевский тип. Переменная данного типа может иметь всего два значения: истина или ложь. Например:

```
$a = true;          // Истина

$a = false;         // Ложь
```

- точки зрения типизации переменных PHP относится к свободно-типизированным языкам. Это значит, что изначально нет необходимости задавать определенный тип переменной. Вместо этого, когда ей присваивается значение, PHP трактует ее соответствующим образом в зависимости от значения и контекста, в котором она применяется.

5.5 Преобразование типов

Преобразование типов используется для того, чтобы работать со значением переменной одного типа так, как если бы она имела другой тип. Эта возможность пригодится нам, когда на 4-ом занятии мы научимся получать данные от пользователя. Оператор преобразования типа – имя типа, заключенное в круглые скобки:
(string) – строка
(int) – целое число
(double) – число с плавающей точкой
(bool) – булевская переменная (true/false)

Пример использования:

```
<?php

$a = '734.789';

echo (int) $a ;

?>
```

Этот код выведет «734», потому что переменная $a будет восприниматься как целое число (с типом int), и дробная часть будет отброшена.

5.6 Операторы и операции

Операции над строками

Наиболее важной операцией над строками является операция их объединения (конкатенации). В PHP она записывается с помощью символа точки («.»). Пример:

```
<?php

$name = 'Дмитрий';

echo 'Hello, ' . $name . '!';

?>
```

Ещё пример:

```
$x = 'Hello, ';

$y = 'World';

$z = $a . $b;

echo $c ;
```

По аналогии с арифметическими операциями есть сокращенный оператор конкатенации .=

```
$x .= $y;
```

эквивалентна

```
$x = $x . $y;
```

- языке PHP существует огромное множество функций для работы со строками, но так как обсуждение функций у нас ещё впереди, то дальнейшие средства обработки строк оставим на потом.

Арифметические операции

Основные операторы, предназначенные в PHP для выполнения математических действий, ничем не отличаются от тех, которые мы используем в жизни:

```
<?
php
$x =            7 + 2;/* сложение */
$x = $x - 5;          /* вычитание */
$x = $x / 2;          /* деление */
$x = $x * $x;         /* умножение         */
                                            деления
$x = $x % 3;          /* остаток от          */
?
>
```

Если оба аргумента – целые числа, то результат также будет целым числом. В то же время, если хотя бы один из операндов – число с плавающей точкой, то и результат будет величиной с плавающей точкой, даже если получается целое число. Пример: 0.5 + 1.5 = 2.0 , а не 2.
Для этих пяти операторов также существуют сокращенные версии для случая, когда одним из операндов выступает переменная, которой присваивается значение. Например, чтобы не писать $a = $a + $b, можно сокращенно записать $a += $b. А для того чтобы удвоить значение переменной $a, достаточно написать $a *= 2 (тоже самое, что $a = $a * 2).

Для увеличения или уменьшения значения переменной на единицу есть еще два сокращенных оператора: инкремента (++) и декремента (--). Подробнее мы рассмотрим их на 3-ем занятии, при изучении циклов.

Операции сравнения

Ещё одну группу операторов составляют операции сравнения: «меньше» (<), «меньше или равно» (<=), «больше» (>) и «больше или равно» (>=). Все они сравнивают два заданных значения и возвращают логическое значение true или false. Например:
<?php

```
$a = (3 < 4);

echo $a ;

?>
```

В случае, если значение $a верно, скрипт выведет единицу, иначе – не выведет ничего. Это связано с особенностью представления булевых переменных в PHP.

Логические операции

Логические операторы проверяют булевы условия. Существует четыре главных булевых условия: И (and или &&), ИЛИ (or или ||) , НЕ (!) и исключающее ИЛИ (xor).
Операция **И** возвращает истину только в том случае, если истинны оба выражения:

$a	**$b**	**$a && $b**
false	false	false
true	false	false
false	true	false
true	true	true

Операция **ИЛИ** возвращает истину в том случае, если истинно хотя бы одно выражение:

$a	**$b**	**$a || $b**
false	false	false
true	false	true
false	true	true
true	true	true

Операция **НЕ** переворачивает значение выражения:

$a	**!$a**
false	true
true	false

- оператором xor Вам нужно будет разобраться в рамках выполнения домашнего задания, составив для него такую же таблицу.

Приоритетность операторов

Как и в математике, в php существует приоритет выполнения операторов. Однако, здесь сразу следует остановиться и подумать, а стоит ли его вообще запоминать?
Наверняка Вы знаете известную загадку «сколько будет два плюс два умножить на два», на которую большинство людей автоматически даёт ответ 8, не учитывая то, что умножение приоритетнее и должно выполняться в первую очередь. И это притом, что в математике приоритеты нужно было запомнить всего для 4 операций (+, -, *, /), а в PHP их будет в десятки раз больше.
Например, взгляните на такое выражение:

```
$c = $x * 5 <= $y * 8 - $a || $b;
```

Глаза сразу разбегаются. Понять, что в итоге получится, очень сложно даже человеку, хорошо знающему приоритеты выполнения операторов. Поэтому, наилучшим решением является постановка скобок, например:

```
$c = ($x * 5) <= (($y * 8) – ($a || $b));
```

Согласитесь, здесь, по крайней мере, с первого взгляда видно, что $c – это результат операции сравнения.

Контроль

- Что такое Интернет
- Как работает Интернет
- Что такое Клиент
- Что такое Сервер
- Какие три программы должны быть установлены на сервере, и за что каждая из них отвечает
- Обязательно ли во время формирования ответа отрабатывает php-интерпретатор
- Что такое HTTP
- Как Вы поняли, зачем нужен PHP
- Базовый синтаксис PHP

- Что такое комментарии и зачем они нужны
- Что такое переменные
- Чем константа отличается от переменной
- Что такое тип данных
- Какие типы данных Вы знаете
- Преобразование типов
- Что такое конкатенация
- Какой тип данных возвращают операции сравнения
- Логические операции
- Нужно ли запоминать приоритеты операторов в PHP

Задание

1. С помощью оператора echo выведите на страницу:

    - Целочисленную переменную
    - Переменную дробного типа
    - Переменную булевского типа
    - Строковую переменную
    - Константу

2. Повторите вывод, заключив переменные в двойные кавычки (“). Посмотрите, что получится. Объясните результат.

3. Повторите вывод, заключив переменные в одинарные кавычки (‘). Посмотрите, что получится. Объясните результат.

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.

5. Выполните эти же действия, с помощью **одного** оператора echo.

6. Попробуйте в выражении использовать разные типы данных, например, сложите число «10» и строку «20 приветов». Что получится? Объясните результат.

7. Дайте ответ на вопрос, как работает оператор xor? В каких случаях он возвращает значение true, в каких – false? Для этого напишите скрипт, который выводит значения операций со всеми возможными вариантами операндов (4 варианта). Чему равно $a xor $a для любых значений $a?

8. Дан фрагмент кода:

```
<?php
$x = 10;
$y = 15;
?>
```

Необходимо дописать несколько операций так, чтобы в итоге значения переменных поменялись местами. При этом, использовать другие переменные запрещается.