Урок 1

Введение в PHP

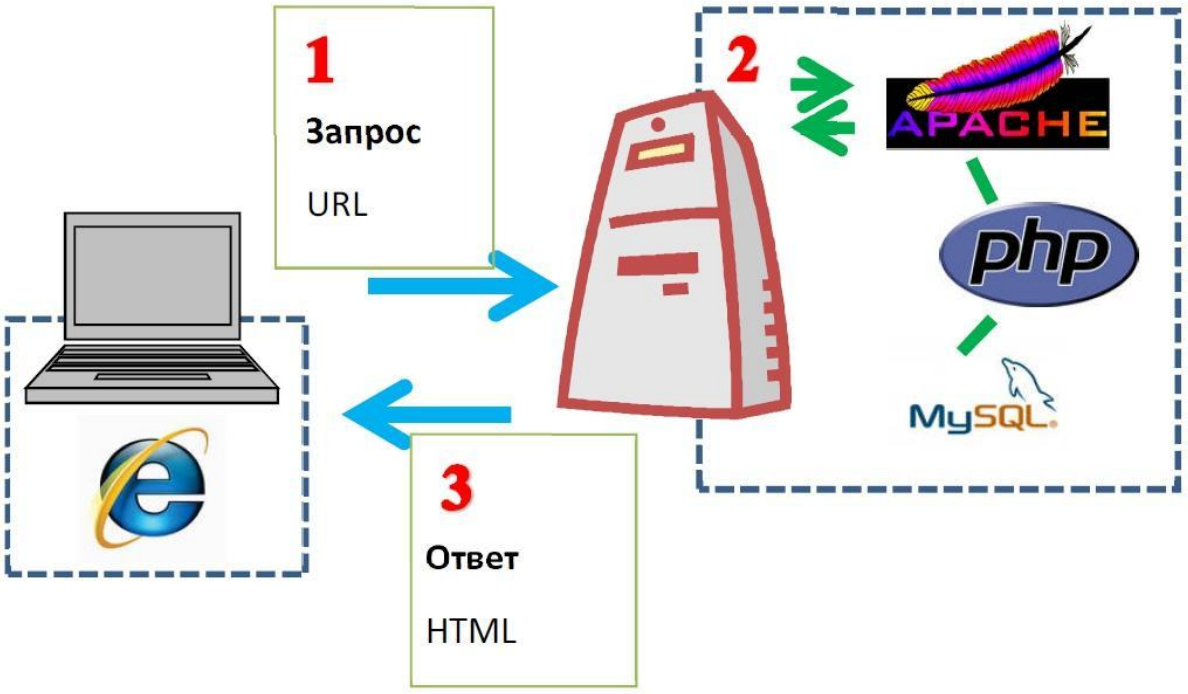
Знакомство с языком

Принципы работы динамических сайтов

На курсе «Основы HTML/CSS» вы уже ознакомились с тем, как создаются простые статические страницы на сайтах и как обратиться к ним по URL, как стилизовать их и создать по ним навигацию.

Задача этого курса - научить делать интерактивные сайты, умеющие реагировать на действия пользователя, обрабатывать введённые данные, помогать работать с сайтом и многое другое. В итоге мы получим несложный, но функционирующий динамический сайт, который будет использовать все полученные в процессе курса знания.

Как мы помним, практически все ресурсы сети Internet доступны посредством URL. Если с HTML всё достаточно просто, то в случае с PHP ситуация гораздо интереснее. Для работы с динамическими сайтами на сервере должен быть установлен перечень специальных программ. Их набор может иметь разные вариации, но принцип работы один:



1. Необходим принимающий входящее соединение веб-сервер – без этой части ничего не заработает. Он умеет принимать и анализировать запросы клиентов. Задача веб-сервера в том, чтобы понять, какую веб-страницу у него попросили, и в соответствии с этим сформировать ответ.
2. PHP-интерпретатор – программа, выполняющая PHP-код. Результатом её работы обычно является HTML-код (но, зачастую, не только он). На данном этапе происходит преобразование PHP-скрипта в знакомый нам статический HTML-код, который умеет читать браузер. В процессе своей работы PHP может использовать третью программу из связки – MySQL.
3. MySQL – свободная система управления базами данных. Зачем она нужна, если у нас уже есть PHP? База данных – это нечто похожее на виртуальный шкаф, данные в котором

находятся в систематизированном и структурированном виде. Вы можете доставать из шкафа только те данные, которые вам необходимы в текущий момент времени.
База данных управляется специализированным ПО. В нашем случае это MySQL. Вы обмениваетесь информацией с сервером БД на согласованном языке (SQL). Сервер БД вместе с веб-сервером образует тандем, читающий и записывающий данные и информацию, предоставляя их посетителям ресурса.
БД возвращает данные PHP-скрипту, который расставляет их по нужным местам страницы или пакета данных, формируя окончательный ответ.

4. Ответ возвращается веб-серверу, который и отдаёт клиенту страницу или пакет данных.

Какие бывают наборы ПО для обеспечения работы всей схемы?

1. LAMP – сокращение, обозначающее набор серверного ПО, широко используемый в сети Internet. Назван по первым буквам названий его компонентов:
   - **L**inux – операционная система Linux;
   - **A**pache – веб-сервер;
   - **M**ariaDB / MySQL – СУБД;
   - **P**HP – язык программирования, используемый для создания веб-приложений (помимо PHP могут подразумеваться другие языки, такие как Perl и Python).
2. XAMPP – кроссплатформенная сборка веб-сервера, X (любая из четырёх операционных систем), Apache, MySQL, PHP, Perl.
3. Очень часто встречается сборка, в которой есть и Apache и NGINX. При этом запрос изначально поступает к NGINX, сервер NGINX отдаёт только картинки и текст (статический контент), а Apache перехватывает запросы, которые необходимо обрабатывать на стороне PHP. Это вызвано тем, что Apache – более ресурсоёмкая система, поэтому для сокращения ресурсных расходов обработку статики отдают в NGINX.
4. LEMP (LNMP) – Nginx вместо Apache (Nginx читается Engine-X). Наиболее быстрая и удобная связка, в которой также участвует сервер PHP-FPM, обрабатывающий запросы к PHP-скриптам. NGINX, в свою очередь, обрабатывает статический контент.

Принципы работы PHP

Язык программирования PHP создали в 1995 году, планируя применять его именно для разработки веб-страниц. PHP получил очень широкое распространение за счёт своей простоты и открытого исходного кода интерпретатора – благодаря этому доработкой языка могли заниматься и энтузиасты, и коммерческие компании.

PHP – серверный интерпретируемый язык программирования. Это означает, что вся логика и все данные будут работать на сервере. Иными словами, мы должны их отправить туда для того, чтобы случилась некая магия.

Все файлы сайта сохранены на серверах, однако не все они на них интерпретируются. HTML и CSS, как и файлы изображений, сервер пересылает браузеру непосредственно, вне зависимости от содержания. PHP-файлы отличаются тем, что они содержат код, который интерпретируется на сервере. Клиенту (браузеру) отправляется не сам PHP-код, а результат его исполнения, который часто является чистым HTML и CSS. Исходный код программ на PHP исполняется непосредственно на веб-сервере. В отличие от JavaScript, пользователь не может посмотреть исходный код программы и узнать, что в ней происходит. Это очень хорошо с точки зрения безопасности, т.к. злоумышленникам будет гораздо сложнее адаптироваться под поведение логики сайта.

При использовании только HTML каждая страница сайта – это отдельный новый файл. PHP позволяет создать весь сайт, используя как минимум 1 файл (почему «как минимум» мы узнаем в течение курса).

Данные содержимого страниц можно хранить в БД, благодаря чему можно сделать удобный интерфейс управления этими данными.

Важно помнить, что PHP хранит только то состояние, которое создано в текущий момент времени. В отличие от классических компилируемых языков программирования, работающих всё время (от запуска до остановки программы), PHP скрипт начинает работать с момента обращения к нему веб-сервера и заканчивает (забывает всё, что знал) тогда, когда отдаёт сгенерированный пакет данных обратно веб-серверу.

Подготовка рабочей среды

Для выполнения команд языка нам необходимо подготовить окружение. Как мы уже выяснили, нам понадобятся следующие компоненты:

- веб-сервер;
- интерпретатор PHP;
- база данных MySQL.

Можно собрать комплект самостоятельно, а можно скачать готовую сборку. Есть много популярных сборок:

- Open Server;
- XAMPP;
- Denwer.

Наиболее гибкая и удобная версия для Windows – Open Server. На Linux-семейство отлично подойдёт XAMPP. Если же захочется собрать всё с нуля собственными руками, то специально для этого была подготовлена большая статья, находящаяся здесь.

Кроме того, для разработки необходимо создать некий тестовый сайт, к которому можно обратиться через браузер. Как правило, готовые сборки предоставляют весь необходимый для этого функционал.

Помимо этого нам понадобится среда разработки. Программы на PHP можно писать в любом текстовом редакторе, даже в том, что по умолчанию входит в стандартный комплект вашей ОС. Однако, это неудобно. Существуют различные редакторы с подсветкой синтаксиса:

- Notepad++;
- Atom;
- Visual Studio Code.

Воспользоваться можно и профессиональными IDE: PhpStorm или Netbeans. Однако на первых порах рекомендуется минимально использовать автоматизированные подсказки от IDE. Задача начинающего разработчика – понять структуру языка и отыскивать ошибки самостоятельно, «взглядом». Ведь где-то IDE может и не оказаться, и тогда проблемы с разработкой будут просто огромные. Поэтому остановимся на самой простой опции : Notepad++ или Atom.

Для работы с БД нам потребуется GUI, чтобы упростить рутинные операции. Можно использовать самый простой клиент – HeidiSQL или среду MySQL-разработчика – MySQL Workbench.

Hello, World!

Настало время написать самую первую программу на PHP. Для этого нужно зайти в корневую директорию нашего тестового сайта и создать там файл с названием index.php. Сразу же отметим, что файлы мы создаём в кодировке UTF-8 without BOM, что соответствует большинству регламентируемых стандартов кода, в том числе и стандартов кода на PHP.

Напишем в файле следующие строки:

```
<?php
echo "Hello, World!";
?>
```

Для начала просто наберём в адресной строке название нашего тестового сайта (к примеру, http://mysite.com), чтобы убедиться, что всё отработало как следует. На экране браузера вы увидите примерно следующее:

Любому PHP сценарию для начала работы требуется дескриптор <?php, который даёт понять интерпретатору, что дальше идёт PHP-код. Очень похоже на HTML-тэги! Можно использовать сокращённый синтаксис <? , но, согласно современным стандартам кода, этого делать не рекомендуется.

Далее мы видим команду echo, единственное назначение которой – вывод на экран того, что передали ей в строчке дальше. Для нас это строка, заключённая в кавычки – «Hello, World!».

Каждая строка команды заканчивается символом «;», отделяющим смысловой блок команды от других команд и блоков. В нашем примере была только одна инструкция, и точку с запятой можно опустить. Но хорошим стилем в PHP считается всегда её ставить. Это указывает на то, что команда завершена.

Завершение программы обозначается с помощью «?>», но это необязательно.

Базовые конструкции языка

Переменные и константы

Программа, которую мы написали чуть ранее, по сути ничем не отличается от статической HTML страницы. Пользователь никак не может повлиять на получаемое содержимое. Поэтому двигаться нужно по пути усложнения кода. Добавим в наш скрипт переменные:

```php
<?php
$name = "GeekBrains user";
echo "Hello, $name!";
?>
```

Только что добавилась переменная. С точки зрения PHP, мы выделили область памяти, адрес которой можно использовать для осуществления доступа к данным. Данные, находящиеся в переменной, называются значением этой переменной. У любой переменной есть определённый числовой адрес, определяющий, в какой именно области памяти лежит значение той или иной переменной шестнадцатеричное число. Машина среди таких переменных ориентируется без проблем, однако для человека это достаточно трудно. В языках программирования переменным можно задавать синоним — например, $name. В данном случае в этой переменной записано имя. Любая переменная в PHP обозначается знаком «$». Переменная в PHP должна состоять из латинских символов (хотя язык допускает имя переменной на других языках) и цифр, не должна содержать спецсимволов, кроме «_», и не может начинаться с числа.

Правильно:

- $variable
- $myVariable
- $_variableWithDigits

Неправильно:

- $1stvariable
- $Переменная (сработает, но так нельзя!)
- $any%other/variablewithSymbols

Откройте ваш скрипт в браузере. Измените значение переменной и откройте снова. Добавьте новые переменные.

Константы

Константы мало чем отличаются от переменных, кроме одной важной детали: им нельзя присвоить другое значение. Константы используются, как некое подобие системных переменных, которые устанавливает разработчик, но обычный пользователь никак повлиять на них не может. Константы устанавливаются с помощью функции define(). Обычно они используются в файлах конфигурации.

```php
<?php
define('MY_CONST', 100);
echo MY_CONST;
?>
```

Обратите внимание, что обращение к константе осуществляется без знака «$». Правила написания названий констант такие же, как и для переменных, однако, согласно текущим стандартам, константы должны быть написаны заглавными буквами.

Типы данных

По сути, переменная – это коробка, в которую можно положить предмет определённого типа: ботинки, банку консервов или кота. У переменных также есть собственные специальные типы данных.

PHP поддерживает восемь простых типа данных:

- boolean (логический тип);
- integer (целые числа);
- double (дробные числа);
- string (строки);
- array (массивы);
- object (объекты);
- resource (ресурсы);
- NULL.

Рассмотрим все эти типы по порядку.

Переменные логического типа могут принимать два значения: true и false или, иначе говоря, истина и ложь. Чаще всего логические значения используются в условных конструкциях, о которых мы поговорим на следующем занятии.

Переменные типа integer представляют целое число со знаком в размере 32 бит (от -2 147 483 648 до 2 147 483 647). PHP обладает возможностью использовать также двоичные, восьмеричные и шестнадцатеричные числа по следующим шаблонам:

- шестнадцатеричные : 0[xX][0-9a-fA-F];
- восьмеричные : 0[0-7];
- двоичные : 0b[01].

В примере ниже мы выведем на экран число 42 с помощью переменной типа integer в различных системах счисления:

```
<?php
$int10 = 42;
$int2 = 0b101010;
$int8 = 052;
$int16 = 0x2A;
echo "Десятеричная система $int10 <br>";
echo "Двоичная система $int2 <br>";
echo "Восьмеричная система $int8 <br>";
echo "Шестнадцатеричная система $int16 <br>";
?>
```

Размер числа с плавающей точкой зависит от платформы. Максимально возможное значение, как правило, составляет ~1.8e308 с точностью около 14 десятичных цифр. Например:

```
<?php
$precise1 = 1.5;
$precise2 = 1.5e4;
$precise3 = 6E-8;
echo "$precise1 | $precise2 | $precise3";
?>
```

Для работы с текстом применяются **строки**, которые бывают двух типов: в двойных кавычках и одинарных. Тип кавычек определяет обработку строк интерпретатором. Как было видно из примера выше, записи переменных в двойных кавычках заменяются значениями, а переменные в одинарных кавычках остаются неизменными.

```
<?php
$a = 1;
echo "$a";
echo '$a';
?>
```

Массив определяет набор элементов, каждый из которых представляет пару ключ=>значение. О массивах мы поговорим позже в процессе изучения нашего курса.

Объект является одним из базовых понятий объектно-ориентированного программирования. О нём пойдёт речь в курсе PHP Level 2.

Тип NULL говорит о том, что переменная не определена. Использование этого значения применимо в случаях, когда нужно указать, что переменная не имеет значения. Например, если переменную необходимо определить, но не инициализировать. При попытке её использования интерпретатор выведет на экран Warning (диагностическое сообщение, предупреждение) о том, что переменная не установлена.

Ресурс – это специальный тип значения переменной, содержащий ссылку на внешний ресурс. В качестве такого ресурса могут использоваться, к примеру, файлы или подключения к базам данных. Ресурсы создаются и используются специальными функциями.

PHP – язык с динамической типизацией. Это означает, что любая переменная в процессе выполнения программы может изменить свой тип данных. В языках со строгой типизацией присвоить, например, переменной с типом boolean строчное или числовое значение не получится. В PHP тот же boolean можно легко превратить в string: true примет значение 1, а false – 0.

Приведение типов бывает автоматическим и явным. Примером автоматического преобразования типа является оператор сложения '+'. В случае, когда любой из операндов является числом с плавающей точкой, все операнды интерпретируются как числа с плавающей точкой, и результатом будет также число с плавающей точкой. В противном случае операнды будут интерпретироваться как целые числа, и результат также будет целочисленным. Стоит помнить о том, что меняется только результат. Сами операнды при использовании их после выражения свой тип не изменят.

Приведение типов можно провести и вручную. Для этого имя требуемого типа записывается в круглых скобках перед приводимой переменной:

```
<?php
$a = 10;
$b = (boolean) $b;
?>
```

Простейшие операции

Операции со строками

Можно объединять несколько строковых переменных в одну. Такая операция называется конкатенация.

```
<?php
$a = 'Hello,';
$b = 'world';
$c = $a . $b;
echo $c;
?>
```

Конкатенация в языке PHP выполняется с помощью символа «.». Кроме того, при выполнении данной операции все переменные других типов, если это возможно, будут приведены к строковому типу. Обратите внимание, что, согласно стандартам при конкатенации двух строк, справа и слева от точки должны быть пробелы.

Математические операции

Вполне резонно, что в PHP существуют стандартные математические операции.

```
<?php
$a = 4;
$b = 5;
echo $a + $b . '<br>';    // сложение
echo $a * $b . '<br>';    // умножение
echo $a - $b . '<br>';    // вычитание
echo $a / $b . '<br>';  // деление
echo $a % $b . '<br>'; // остаток от деления
echo $a ** $b . '<br>'; // возведение в степень
?>
```

Существуют и другие, менее очевидные: *=, /=, +=, = – эти операторы позволяют выполнить математическое действие и сразу же присвоить значение переменной.

```
<?php
$a = 4;
$b = 5;
$a += $b;
echo 'a = ' . $a;
$a = 0;
echo $a++;     // Постинкремент
echo ++$a;    // Преинкремент
echo $a--;     // Постдекремент
echo --$a;    // Предекремент
?>
```

Есть и операции сравнения.

```
<?php
$a = 4;
$b = 5;
var_dump($a == $b);  // Сравнение по значению
var_dump($a === $b); // Сравнение по значению и типу
var_dump($a > $b);    // Больше
var_dump($a < $b);    // Меньше
var_dump($a <> $b);  // Не равно
var_dump($a != $b);   // Не равно
var_dump($a !== $b); // Не равно без приведения типов
var_dump($a <= $b);  // Меньше или равно
var_dump($a >= $b);  // Больше или равно?>
```

Функция var_dump позволяет вывести тип переменной и её значение. Попробуйте выполнить код. Обратите внимание, что знак «=» не знак равенства, а знак присваивания.

Версии языка и их различия

В данный момент актуальными считаются версии PHP 5.6 и 7. Однако во многих компаниях используются и PHP 5.3+. Поэтому нужно быть готовыми к тому, что какие-то вещи будут несовместимы. О них будет сказано на каждом уроке в случае, если работа с тем или иным функционалом различается от версии к версии.

Практическое задание

1. Установить программное обеспечение: веб-сервер, базу данных, интерпретатор, текстовый редактор и проверить, что всё работает правильно.
2. Выполнить примеры из методички, разобраться, как это работает.
3. Объяснить, как работает данный код:

```
<?php
$a = 5;
$b = '05';
var_dump($a == $b);                         // Почему true?
var_dump((int)'012345');                    // Почему 12345?
var_dump((float)123.0 === (int)123.0); // Почему false?
var_dump((int)0 === (int)'hello, world'); // Почему true?
?>
```

1. Используя имеющийся HTML-шаблон, сделать так, чтобы главная страница генерировалась через PHP. Создать блок переменных в начале страницы. Сделать так, чтобы h1, title и текущий год генерировались в блоке контента из созданных переменных.
2. *Используя только две переменные, поменяйте их значение местами. Например, если a = 1, b = 2, надо, чтобы получилось: a = 2b = 1, . Дополнительные переменные использовать нельзя.

Дополнительные материалы

1. http://php.net/manual/ru/language.basicsyntax.comments.php
2. http://php.net/manual/ru/language.expressions.php
3. https://ru.wikipedia.org/wiki/HTTP

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Котеров Д.: PHP 5 в подлиннике.
2. Head First PHP and MySQL.