Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

Кафедра информационных систем
и программной инженерии

Х.М. Салех

Основы разработки ВЕБ-приложений

Методические указания к лабораторным работам

Владимир 2019

PHP

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.

Преимущества PHP

Главным фактором языка PHP является практичность. PHP должен предоставить программисту средства для быстрого и эффективного решения поставленных задач. Практический характер PHP обусловлен пятью важными характеристиками:

- традиционностью;
- простотой;
- эффективностью;
- безопасностью;
- гибкостью.

Существует еще одна «характеристика», которая делает PHP особенно привлекательным: он распространяется бесплатно! Причем, с открытыми исходными кодами (Open Source).

Традиционность

Язык PHP будет казаться знакомым программистам, работающим в разных областях. Многие конструкции языка позаимствованы из Си, Perl.

Код PHP очень похож на тот, который встречается в типичных программах на C или Pascal. Это заметно снижает начальные усилия при изучении PHP. PHP — язык, сочетающий достоинства Perl и Си и специально нацеленный на работу в Интернете, язык с универсальным (правда, за некоторыми оговорками) и ясным синтаксисом.

И хотя PHP является довольно молодым языком, он обрел такую популярность среди web-программистов, что на данный момент является чуть ли не самым популярным языком для создания web-приложений (скриптов).

Простота

Сценарий PHP может состоять из 10 000 строк или из одной строки — все зависит от специфики вашей задачи. Вам не придется подгружать библиотеки, указывать специальные параметры компиляции или что-нибудь в этом роде. Механизм PHP просто начинает выполнять код после первой экранирующей последовательности (<?) и продолжает выполнение до того момента, когда он встретит парную экранирующую последовательность (?>). Если код имеет правильный синтаксис, он исполняется в точности так, как указал программист.

PHP — язык, который может быть встроен непосредственно в html -код страниц, которые, в свою очередь будут корректно обрабатываться PHP -интерпретатором. Мы можем использовать PHP для написания CGI-сценариев и избавиться от множества неудобных операторов вывода текста. Мы можем привлекать PHP для формирования HTML-документов, избавившись от множества вызовов внешних сценариев.

Большое разнообразие функций PHP избавят вас от написания многострочных пользовательских функций на C или Pascal .

Эффективность

Эффективность является исключительно важным фактором при программировании для многопользовательских сред, к числу которых относится и web .

Очень важное преимущество PHP заключается в его **«движке»**. «Движок» PHP не является ни компилятором, ни интерпретатором. Он является транслирующим интерпретатором. Такое устройство «движка» PHP позволяет обрабатывать сценарии с достаточно высокой скоростью.

По некоторым оценкам, большинство PHP-сценариев (особенно не очень больших размеров) обрабатываются быстрее аналогичных им программ, написанных на Perl. Однако, чтобы не делали разработчики PHP, откомпилированные исполняемые файлы будут работать значительно быстрее – в десятки, а иногда и в сотни раз. Но производительность PHP вполне достаточна для создания вполне серьезных web-приложений. Подробно об устройстве и характеристиках «движка» PHP можно ознакомиться **здесь**.

Безопасность

PHP предоставляет в распоряжение разработчиков и администраторов гибкие и эффективные средства безопасности, которые условно делятся на две категории: средства системного уровня и средства уровня приложения.

1. Средства безопасности системного уровня

В PHP реализованы механизмы безопасности, находящиеся под управлением администраторов; при правильной настройке PHP это обеспечивает максимальную свободу действий и безопасность. PHP может работать в так называемом безопасном режиме (safe mode), который ограничивает возможности применения PHP пользователями по ряду важных показателей. Например, можно ограничить максимальное время выполнения и использование памяти (неконтролируемый расход памяти отрицательно влияет на быстродействие сервера). По аналогии с cgi-bin администратор также может устанавливать ограничения на каталоги, в которых пользователь может просматривать и исполнять сценарии PHP, а также использовать сценарии PHP для просмотра конфиденциальной информации на сервере (например, файла passwd).

2. Средства безопасности уровня приложения

В стандартный набор функций PHP входит ряд надежных механизмов шифрования. PHP также совместим с многими приложениями независимых фирм, что позволяет легко интегрировать его с защищенными технологиями электронной коммерции (e-commerce). Другое преимущество заключается в том, что исходный текст сценариев PHP нельзя просмотреть в браузере, поскольку сценарий компилируется до его отправки по запросу пользователя. Реализация PHP на стороне сервера предотвращает похищение нетривиальных сценариев пользователями, знаний которых хватает хотя бы для выполнения команды View Source.

Подробно о безопасности PHP можно ознакомиться **здесь**

Гибкость

Поскольку PHP является встраиваемым (embedded) языком, он отличается исключительной гибкостью по отношению к потребностям разработчика. Хотя PHP обычно рекомендуется использовать в сочетании с HTML, он с таким же успехом интегрируется и в JavaScript, WML, XML и другие языки.

Кроме того, хорошо структурированные приложения PHP легко расширяются по мере необходимости (впрочем, это относится ко всем основным языкам программирования).

Нет проблем и с зависимостью от браузеров, поскольку перед отправкой клиенту сценарии PHP полностью компилируются на стороне сервера. В сущности, сценарии PHP могут передаваться любым устройствам с браузерами, включая сотовые телефоны, электронные записные книжки, пейджеры и портативные компьютеры, не говоря уже о традиционных ПК. Программисты, занимающиеся вспомогательными утилитами, могут запускать PHP в режиме командной строки.

Поскольку PHP не содержит кода, ориентированного на конкретный web-сервер, пользователи не ограничиваются определенными серверами (возможно, незнакомыми для них). Apache, Microsoft IIS, Netscape Enterprise Server, Stronghold и Zeus — PHP работает на всех перечисленных серверах. Поскольку эти серверы работают на разных платформах, PHP в целом является платформенно-независимым языком и существует на таких платформах, как UNIX, Solaris, FreeBSD и Windows 95/98/NT/2000/XP/2003.

Наконец, средства PHP позволяют программисту работать с внешними компонентами, такими как Enterprise Java Beans или COM-объекты Win32. Благодаря этим новым возможностям PHP занимает достойное место среди современных технологий и обеспечивает масштабирование проектов до необходимых пределов.

Бесплатное распространение

Стратегия Open Source, и распространение исходных текстов программ в массах, оказало несомненно благотворное влияние на многие проекты, в первую очередь — Linux, хотя и успех проекта Apache сильно подкрепил позиции сторонников Open Source. Сказанное относится и к истории создания PHP, поскольку поддержка пользователей со всего мира оказалась очень важным фактором в развитии проекта PHP.

Принятие стратегии Open Source и бесплатное распространение исходных текстов PHP оказало неоценимую услугу пользователям. Вдобавок, отзывчивое сообщество пользователей PHP является своего рода «коллективной службой поддержки», и в популярных электронных конференциях можно найти ответы даже на самые сложные вопросы.

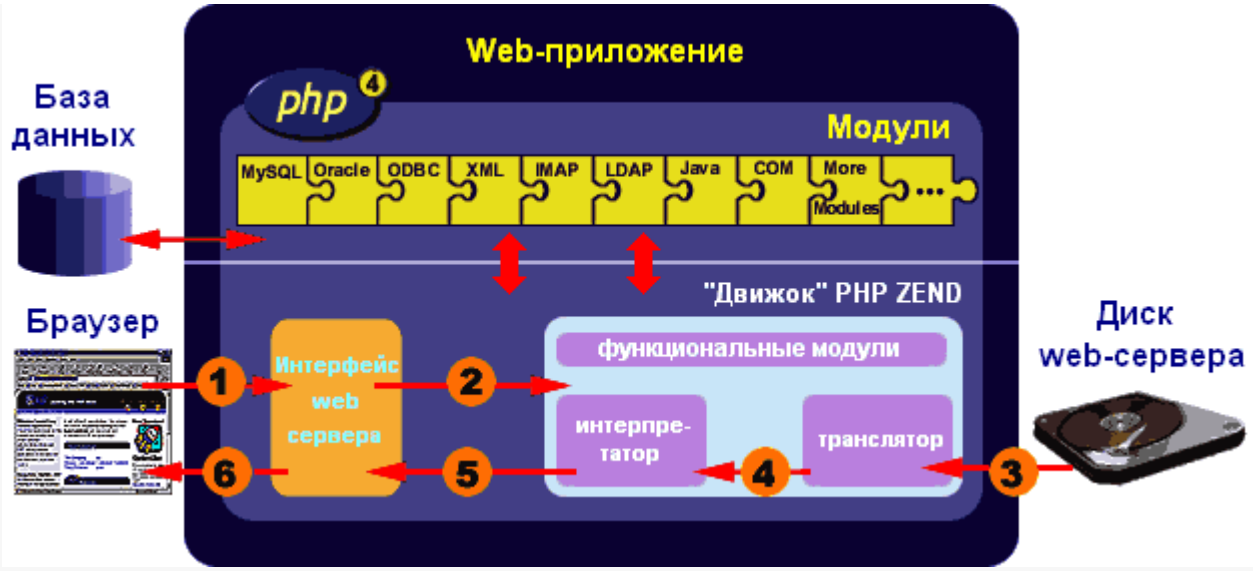
Программа, переводящая код, написанный на одном языке программирования, на другой называется транслятором. Компилятор – это тоже транслятор. Он переводит код, написанный на языке высокого уровня, в машинный код. В результате процесса компиляции создается двоичный исполняемый файл, который уже можно запускать без компилятора.

Интерпретатор – это совершенно другая категория. Интерпретатор не переводит код, а исполняет его. Интерпретатор анализирует код программы и исполняет каждую его строку. Каждый раз при исполнении такого кода, необходимо пользоваться интерпретатором.

По производительности интерпретаторы значительно уступают компиляторам, поскольку двоичный код выполняется намного быстрее. Зато интерпретаторы позволяют полностью контролировать программу во время ее исполнения.

Что касается PHP, то он не является ни компилятором, ни интерпретатором. PHP представляет собой нечто среднее, между компилятором и интерпретатором. Попробуем в этом разобраться и рассмотрим, как PHP обрабатывает код.

Рассмотрим рисунок:



Мы видим, что PHP составлен из двух почти независимых блоков — транслятора и интерпретатора. Зачем же понадобилось так делать? Конечно, из соображений быстродействия.

На вход PHP подается сценарий. Он переводит его (транслирует) , проверяя синтаксис, в специальный байт-код (внутреннее представление). Затем PHP выполняет байт-код (а не код самой программы), при этом он не создает исполняемый файл.

Байт-код значительно компактнее обыкновенного кода программы, поэтому его легче (и быстрее) интерпретировать (выполнять). Посудите сами: синтаксический разбор осуществляется всего один раз на этапе трансляции, а исполняется уже "полуфабрикат" - байт-код, который гораздо более удобен для этих целей. Поэтому, PHP больше является интерпретатором, нежели компилятором.

Установка Apache 2.0.xx и PHP5 в Windows

Установка Apache Apache 2.0.xx под Windows и конфигурирование его для использования совместно с PHP 5 достаточно простая задача. Вам не потребуется компилировать исходники Apache и PHP, так как всегда доступны бинарные дистрибутивы для Apache и PHP.

Перед установкой убедитесь, что логин, под которым Вы работаете, входит в группу Administrators. Убедитесь, что 80-й TCP-порт не занят каким-либо другим сервисом или приложением, например, Microsoft IIS. Если 80-й порт занят, измените порт в настройках IIS-а (или другого сервиса) на другой, или же отключите или деинсталлируйте данное приложение.

Установка Apache 2.0.xx в Windows

Для получения дистрибутива Apache 2.0.xx, обратитесь на страницу загрузок Apache нашего портала (Download / Apache), либо скачайте бинарный инсталляционный пакет (*.msi) с зеркала официального сайта проекта Apache HTTPD здесь. Выбирайте тот файл пакета, который имеет имя подобное apache_2.0.xx-win32-x86-no_ssl.msi (где xx-текущая версия Apache).

Сразу после получения Apache, установите его, процесс установки тривиален и не имеет каких-либо сложных особенностей. Обратите внимание на путь установки Apache (Destination Folder), отнеситесь к выбору директории для установки Вашего Apache с вниманием. Однако мы будем рассматривать конфигурирование Apache и PHP на примере установки в директорию по умолчанию:

```
C:/Program FIles/Apache Group/
```

После того, как установка завершена, можете приступать к установке PHP5.

Установка PHP5 и связка его с Apache 2.0.xx

Скачайте архив бинарных файлов PHP5 (архивы *.zip) последней стабильной версии, обратившись либо к нашей странице загрузок PHP (Download / PHP), либо к странице загрузок на официальном сайте PHP (php.net). Не скачивайте PHP в виде инсталляционных пакетов (типа *.msi), в данной ситуации они Вам не помогут.

Скачали PHP5 в виде *.zip архива? Тогда продолжим. Откройте корневой каталог Вашего Apache, по умолчанию это должен быть:

```
C:/Program Files/Apache Group/Apache2/
```

Создайте в корневом каталоге Apache подкаталог PHP. Например:

```
C:/Program Files/Apache Group/Apache2/PHP/
```

Разархивируйте файлы из архива PHP5 *.zip в созданный подкаталог PHP. Причем обратите внимание на три очень важных файла: php-cgi.exe, php5ts.dll и php5apache2.dll , они должны будут находиться именно в созданном Вами каталоге /PHP. Если перечисленные файлы присутствуют, все в порядке, можно приступать к конфигурированию.

Теперь остановите Ваш Apache командой (воспользуйтесь меню Пуск -> Выполнить):

```
NET STOP Apache2
```

Если команда NET STOP Apache2 не сработает, используйте команду NET STOP Apache

Вот теперь откройте конфигурационный файл Apache, который имеет имя httpd.conf. По умолчанию он расположен в подкаталоге conf:

```
C:/Program Files/Apache Group/Apache2/conf/
```

Откройте файл конфигурации Apache (httpd.conf) в "блокноте" и добавьте в него следующие строки (добавляйте эту директиву после большого списка таких директив как LoadModule):

```
LoadModule php5_module "C:/Program Files/Apache Group/Apache2/PHP/php5apache2.dll"
```

Далее, найдите в конфигурационном файле Apache следующую директиву:

```
DirectoryIndex index.html
```

И измените ее на:

```
DirectoryIndex index.php index.phtml index.php5 index.html index.htm index.shtml
```

Далее, Вам необходимо сопоставить расширения .php, добавив в httpd.conf следующую директиву (добавляйте ее в район директив AddType):

```
AddType application/x-httpd-php .php
```

Теперь создайте каталог, в котором будут храниться Ваши документы и скрипты (.php и .html файлы). Допустим, это будет каталог:

```
C:/www/
```

Теперь найдите директиву DocumentRoot и измените ее значение на следующее:

```
DocumentRoot "C:/www"
```

Таким образом, в каталоге C:/www/ будут размещаться Ваши .html страницы и .php скрипты.

Создайте подкаталоги:

```
C:/www/cgi/
C:/www/cgi-bin/
```

В них будут храниться скрипты .cgi (включая бинарные). Далее установите директивы ScriptAlias следующим образом:

```
ScriptAlias /cgi/ "C:/www/cgi/"
ScriptAlias /cgi-bin/ "C:/www/cgi-bin/"
```

Далее найдите следующую секцию директив:

```
<Directory "C:/Program Files/Apache Group/Apache2/htdocs">
```

И замените ее на следующую:

```
<Directory "C:/www">
Options Indexes Includes
AllowOverride All
Allow from all
</Directory>
```

Таким образом Вы задали ряд настроек по умолчанию для каталога C:/www и всех его подкаталогов. А именно: автоматическую генерацию списка содержимого каталога при просмотре его в браузере (если это не нужно, установите просто Options Includes), поддержку SSI, а также возможность использовать файлы .htaccess для индивидуальной настройки каталогов.

Далее убедитесь, что у Вас есть директива AddHandler:

```
AddHandler .cgi .exe
```

Эта директива указывает Apache на то, что файлы с расширениями .cgi и .exe следует воспринимать как CGI-программы.

Установите параметры SSI:

```
Addtype text/html .shtml .shtm .stm
AddHandler server-parsed .shtml .shtm .stm
```

Благодаря этим директивам файлы с расширениями .shtml .shtm и .stm будут обрабатываться как SSI.

Теперь сохраните Ваш файл конфигурации httpd.conf.

А вот сейчас Вам нужно установить файл конфигурации PHP (php.ini) в корневой каталог Apache2. По умолчанию это:

```
C:/Program Files/Apache Group/Apache2/
```

Файл конфигурации php.ini Вы можете найти в архиве *.zip загруженного Вами PHP, а если он там отсутствует, что часто бывает, то скачать php.ini, причем с комментариями на русском языке, можете **здесь**.

После того, как Вы скопировали php.ini в корневой каталог Apache2, необходимо произвести конфигурирование PHP. Во-первых, создайте в каталоге PHP (директория по умолчанию C:/Program Files/Apache Group/Apache2/PHP) два новых подкаталога:

```
C:/Program Files/Apache Group/Apache2/PHP/PECL/
C:/Program Files/Apache Group/Apache2/PHP/sessions/
```

Созданные каталоги необходимы для библиотек расширений PECL и хранения файлов сессий соответственно.

Теперь откройте файл конфигурации php.ini и найдите в нем директиву, отвечающую за путь к расширениям:

```
extension_dir = "путь_к_расширениям_php"
```

Установите ее значение в соответствии с месторасположением каталога /PECL. То есть в нашем случае это:

```
extension_dir = "C:/Program Files/Apache Group/Apache2/PHP/PECL"
```

Теперь найдите директиву, отвечающую за путь к хранилищу файлов сессий:

```
session.save_path = путь_к_хранилищу_сессий
```

Установите значение этой директивы в соответствие с месторасположением каталога /sessions. В нашем случае по умолчанию это:

```
session.save_path = C:/Program Files/Apache Group/Apache2/PHP/sessions
```

С директориями все. Но есть еще один момент, касающийся только PHP5. Он заключается в том, что PHP5 не имеет встроенной поддержки MySQL (в отличие от PHP4). Это означает что Вам придется "вручную" установить поддержку MySQL.

Поддержка MySQL в PHP5 осуществляется с помощью внешнего расширения. Для включения поддержки MySQL в PHP5 выполните следующие операции. В *.zip архиве PHP найдите файл libmysql.dll и скопируйте его в корневой каталог Вашего Apache, то есть по умолчанию в:

```
C:/Program Files/Apache Group/Apache2/
```

Теперь отыщите в *.zip архиве PHP файл php_mysql.dll (в архиве он находится в подкаталоге ext, и не спутайте его сphp_mysqli.dll) и скопируйте его в каталог PECL, по умолчанию:

```
C:/Program Files/Apache Group/Apache2/PHP/PECL/
```

Теперь в файле php.ini найдите примерно следующую строку директивы:

```
;extension=php_mysql.dll
```

Если там такая строка отсутствует, впишите ее самостоятельно. Как видите, директива закомментирована (перед ней стоит **;**), раскомментируйте директиву, убрав знак точки с запятой перед ней (**;**):

```
extension=php_mysql.dll
```

Таким вот образом Вы включили поддержку MySQL в PHP5.

Теперь сохраните файл php.ini, приняв изменения.

На этом конфигурирование Apache+PHP завершено. Пришло время испытать Ваш сервер. Создайте в каталогеC:/www/ файл index.php следующего содержания:

```
<?php
phpinfo();
?>
```

Теперь запустите Ваш Apache следующей командой (используйте также меню Пуск -> Выполнить):

```
NET START Apache2
```

Теперь откройте Ваш браузер и запросите свой сервер:

```
http://localhost/
```

Если Вы все сделали верно и Ваш веб-сервер Apache работает, Вы увидите информацию о PHP.

Синтаксис PHP

Общие понятия

Язык PHP специально предназначен для веб-программирования. PHP сочетает достоинства языков C и Perl и при этом весьма прост в изучении и обладает значительными преимуществами перед традиционными языками программирования.

Синтаксис PHP очень напоминает синтаксис языка C и во многом заимствован из таких языков как Java и Perl.

Программист C очень быстро освоит язык PHP и сможет использовать его с максимальной эффективностью.
В принципе, в PHP есть практически все операторы и функции, имеющиеся в стандартном GNU C (или их аналоги), например есть циклы (while, for), операторы выбора (if, switch), функции работы с файловой системой и процессами (fopen, *dir, stat, unlink, popen, exec), функции ввода-вывода (fgets,fputs,printf) и множество других...

Цель данного раздела - краткое ознакомление с основами синтаксиса языка PHP. Более подробную информацию по конкретным составляющим синтаксиса PHP вы найдете в соответствующих разделах.

PHP и HTML

Синтаксис любого языка программирования гораздо легче "почувствовать" на примерах, нежели используя какие-то диаграммы и схемы. Поэтому приведем пример простейшего скрипта на PHP:

```
<html>
     <head>
          <title>Пример</title>
     </head>
     <body>

          <?
          echo "Привет, я - скрипт PHP!";
          ?>

     </body>
</html>
```

Обратите внимание, что HTML-код корректно обрабатывается интерпретатором PHP.

Начало сценария вас может озадачить: разве это сценарий? Откуда HTML-тэги <html> и <body>? Вот тут-то и кроется главная особенность (кстати, чрезвычайно

удобная) языка PHP: PHP-скрипт может вообще не отличаться от обычного HTML-документа.

Идем дальше. Вы, наверное, догадались, что сам код сценария начинается после открывающего тэга <? и заканчивается закрывающим ?>. Итак, между этими двумя тэгами текст интерпретируется как программа, и в HTML-документ не попадает. Если же программе нужно что-то вывести, она должна воспользоваться оператором echo.

Итак, PHP устроен так, что любой текст, который расположен вне программных блоков, ограниченных <? и ?>, выводится в браузер непосредственно. В этом и заключается главная особенность PHP, в отличие от Perl и C, где вывод осуществляется с помощью стандартных операторов.

Разделение инструкций

Инструкции разделяются также как и в C или Perl - каждое выражение заканчивается точкой с запятой.

Закрывающий тег (?>) также подразумевает конец инструкции, поэтому два следующих фрагмента кода эквиваленты:

```
<?php
     echo "Это тест";
?>

<?php echo "Это тест" ?>
```

Комментарии в PHP скриптах

Написание практически любого скрипта не обходится без комментариев.

PHP поддерживает комметарии в стиле 'C', 'C++' и оболочки Unix. Например:

```
<?php
     echo "Это тест"; // Это однострочный комментарий в ст
иле c++
     /* Это многострочный комментарий
        еще одна строка комментария */
     echo "Это еще один тест";
     echo "Последний тест"; # Это комментарий в стиле обол
очки Unix
?>
```

Однострочные комментарии идут только до конца строки или текущего блока PHP-кода, в зависимости от того, что идет перед ними.

```
<h1>Это <?php # echo "простой";?> пример.</h1>
<p>Заголовок вверху выведет 'Это пример'.
```

Будьте внимательны, следите за отсутствием вложенных 'C'-комментариев, они могут появиться во время комментирования больших блоков:

```
<?php
  /*
     echo "Это тест"; /* Этот комментарий вызовет проблему
 */
  */
?>
```

Однострочные комментарии идут только до конца строки или текущего блока PHP-кода, в зависимости от того, что идет перед ними. Это означает, что HTML-код после // ?> БУДЕТ напечатан: ?> выводит из режима PHP и возвращает в режим HTML, но // не позволяет этого сделать.

Переменные в PHP

Имена переменных обозначаются знаком $. То же самое "Привет, я - скрипт PHP! " можно получить следующим образом:

```
<?php
$message = "Привет, я - скрипт PHP!";
echo $message;
?>
```

Типы данных в PHP

PHP поддерживает восемь простых типов данных:

Четыре скалярных типа:

- boolean (двоичные данные)
- integer (целые числа)
- float (числа с плавающей точкой или 'double')
- string (строки)

Два смешанных типа:

- array (массивы)
- object (объекты)

И два специальных типа:

resource (ресурсы)
NULL ("пустые")

Существуют также несколько псевдотипов:

- mixed (смешанные)
- number (числа)
- callback (обратного вызова)

Скалярные типы данных

» Двоичные данные (boolean)

» Целые числа (Integer)

» Числа с плавающей точкой (Float)

» Строки (String)

Смешанные типы данных

» Массивы (Array)

» Объекты (Object)

Специальные типы данных

» Ресурсы (Resource)

» Пустой тип (NULL)

Псевдотипы данных

» Смешанный (Mixed)

» Числа (Number)

» Обратного вызова (Callback)

Дополнительно

» Манипуляции с типами данных

Выражения в PHP

Основными формами выражений являются константы и переменные. Например, если вы записываете "$a = 100", вы присваиваете '100' переменной $a:

```
$a = 100;
```

В приведенном примере $a - это переменная, = - это оператор присваивания, а 100 - это и есть выражения. Его значение 100.

Выражением может быть и переменная, если ей сопоставлено определенное значение:

```
$x = 7;
$y = $x;
```

В первой строке рассмотренного примера выражением является константа 7, а во второй строке - переменная $x, т.к. ранее ей было присвоено значение 7. $y = $x также является выражением.

Подробно о выражениях в PHP вы найдете

Операторы PHP

Оператором называется нечто, состоящее из одного или более значений (выражений, если говорить на жаргоне программирования), которое можно вычислить как новое значение (таким образом, вся конструкция может рассматриваться как выражение).

Примеры операторов PHP:

Операторы присвоения:

```
<?php

$a = ($b = 4) + 5; // результат: $a установлена значением
9, переменной $b присвоено 4.

?>
```

Комбинированные операторы:

```
<?php

$a = 3;
```

```
$a += 5; // устанавливает $a значением 8, аналогично запис
и: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // устанавливает $b строкой "Hello There!"
,  как и $b = $b . "There!";

?>
```

Строковые операторы:

```
<?php
$a = "Hello ";
$b = $a . "World!"; // $b содержит строку "Hello World!"

$a = "Hello ";
$a .= "World!";     // $a содержит строку "Hello World!"
?>
```

Существуют также логические операторы и операторы сравнения, однако их принято рассматривать в контексте управляющих конструкций языка.

Подробную информацию по операторам PHP вы найдете .

Управляющие конструкции языка PHP

Основными конструкциями языка PHP являются:

1. Условные операторы (if, else);
2. Циклы (while, do-while, for, foreach, break, continue);
3. Конструкции выбора (switch);
4. Конструкции объявления (declare);
5. Конструкции возврата значений (return);
6. Конструкции включений (require, include).

Примеры конструкций языка PHP:

```
<?php
if ($a > $b) echo "значение a больше, чем b";
?>
```

Приведенный пример наглядно показывает использование конструкции if совместно с оператором сравнения ($a > $b).

В следующем примере если переменная $a не равна нулю, будет выведена строка "значение а истинно (true), то есть показано взаимодействие условного оператора (конструкции) if с логическим оператором:

```php
<?php
if ($a) echo "значение а истинно (true) ";
?>
```

А вот пример цикла while:

```php
<?php
$x=0;
while ($x++<10) echo $x;
// Выводит 12345678910
?>
```