



РАЗРАБОТКА WEB- ПРИЛОЖЕНИЙ НА PHP&MYSQL

УРОК 1

Введение. Основы синтаксиса

ВВЕДЕНИЕ.....	3
WEB-ПРОГРАММИРОВАНИЕ.....	3
Что такое web-программирование?.....	3
Задачи web-программирования и их решение.....	4
Немного терминов.....	5
АРХИТЕКТУРА «КЛИЕНТ-СЕРВЕР». ВЫПОЛНЕНИЕ СЕРВЕРНЫХ СЦЕНАРИЕВ. CGI.....	6
Схема взаимодействия сервера и клиента.....	6
Как работают скрипты на сервере?.....	7
ОБЗОР СУЩЕСТВУЮЩИХ СЕРВЕРНЫХ WEB-ТЕХНОЛОГИЙ.....	10
ASP.NET.....	11
ColdFusion.....	11
Ruby on Rails.....	12
JSP.....	13
КРАТКАЯ ИСТОРИЯ PHP. ОБЗОР ВЕРСИЙ И ВОЗМОЖНОСТЕЙ.....	14
Что такое PHP и область применения.....	14
История PHP и обзор ранних версий [4].....	14
PHP 3.0 [4].....	15
PHP 4.0 [4].....	16
PHP 5.0 [4].....	17
ПЛАТФОРМА PHP & MYSQL. ДОСТОИНСТВА И НЕДОСТАТКИ.....	18
Базы данных для web-программирования.....	18
Достоинства и недостатки.....	18

<u>ОБЗОР ИНСТРУМЕНТАРИЯ.....</u>	<u>20</u>
<u>УСТАНОВКА ИНСТРУМЕНТАРИЯ.....</u>	<u>24</u>
<u>Установка Apache.....</u>	<u>24</u>
<u>Установка PHP.....</u>	<u>25</u>
<u>Конфигурирование.....</u>	<u>26</u>
<u>Установка MySQL и дополнительное конфигурирование PHP.....</u>	<u>28</u>
<u>Упрощенная установка платформы WAMP.....</u>	<u>32</u>
<u>Распространенные редакторы кода.....</u>	<u>34</u>
<u>ОСНОВЫ СИНТАКСИСА.....</u>	<u>36</u>
<u>Включение PHP-кода внутрь страницы.....</u>	<u>36</u>
<u>Запись выражений. Комментарии.....</u>	<u>37</u>
<u>Переменные.....</u>	<u>39</u>
<u>Типы данных.....</u>	<u>40</u>
<u>Преобразование типов данных.....</u>	<u>44</u>
<u>Константы.....</u>	<u>46</u>
<u>ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА.....</u>	<u>48</u>
<u>Операторы, операции и выражения.....</u>	<u>48</u>
<u>Условные конструкции.....</u>	<u>51</u>
<u>Циклические конструкции.....</u>	<u>54</u>
<u>ДОМАШНЕЕ ЗАДАНИЕ.....</u>	<u>57</u>
<u>ИТОГИ.....</u>	<u>57</u>

ВВЕДЕНИЕ

В предыдущих курсах вы познакомились с технологиями, которые позволяют определять внешний вид и содержимое web-страницы, а также управлять ими (**XHTML**, **CSS**, **JavaScript**). Кроме того, вы уже знаете, что существует серьезная разница между внешним видом – **представлением** и содержимым – **контентом** с точки зрения подходов создания и управления.

Наверняка, изучая эти технологии, вы задумывались над тем, что они достаточно трудоёмки и громоздки в плане написания кода и для того чтобы создать мало-мальски серьёзный проект необходимо написать порой тысячи страниц. А это – непосильная задача не только для одного человека, но и для целой команды людей.

Поэтому, конечно же, существуют инструменты, которые помогают автоматизировать процесс создания и эксплуатации сайта. В их основе лежит web-программирование, которое за последние 5-8 лет уже полноценно оформилось в отдельную отрасль.

WEB-ПРОГРАММИРОВАНИЕ

Что такое WEB-ПРОГРАММИРОВАНИЕ?

Зародившись в научной среде, Интернет изначально был ориентирован на обмен статичной текстовой информацией. Очень скоро, перехлестнувшись за рамки академических институтов, Интернет шагнул к простым пользователям, потребности которых требовали динамических возможностей от web-страниц что, соответственно, повлекло изменение требований к знаниям и умениям разработчиков.

Сегодня web-разработка – это целая индустрия, включающая в себя несколько профессиональных направлений, каждое из которых решает задачу по удовлетворению тех или иных требований пользователей.

Эстетические и имиджевые требования удовлетворяются силами Web-дизайнеров. К удовлетворению требований к удобству и функ-

циональности интерфейса сайта весьма причастны **HTML**-кодеры или **HTML**-верстальщики. Потребность в интерактивных и мультимедийных материалах утоляют **Flash**-разработчики и специалисты по **JavaScript**. А где же место Web-программирования?

Web-программист – самая на первый взгляд незаметная из этих профессий, поскольку чаще всего пользователь сайта не соприкасается с результатами его труда. Но, тем не менее, именно web-программист создает ту среду, где потом разворачивается деятельность web-дизайнеров, **HTML**-кодеров и **Flash**-разработчиков.

ЗАДАЧИ WEB-ПРОГРАММИРОВАНИЯ И ИХ РЕШЕНИЕ.

Сформулируем задачи, которые стоят перед web-программистом:

- 1 Создание динамически изменяемых элементов страниц, реагирующих на поведение пользователей.**
- 2 Добавление «интеллекта» на страницу: «узнавание» пользователя, возможность настройки интерфейса.**
- 3 Обеспечение возможности автоматической генерации («на лету») web-страниц.**
- 4 Сбор информации от пользователя, хранение этой информации и управление ею.**
- 5 Реализация работы какого-либо сервиса, доступ к которому будет осуществлен со страниц сайта.**

Решение этих важных задач осуществляется двумя путями:

- 1 Клиентское программирование** – создание небольших программ (скриптов), выполняющихся в среде браузера. Вы с этим направлением уже знакомы, благодаря **JavaScript**. С его помощью можно решить первую и вторую задачи (частично третью), но вот самых серьезных задач при помощи только клиентских скриптов решить нельзя.
- 2 Серверное программирование** – разработка скриптов, которые выполняются на сервере и способны генерировать **HTML**-код, работать с файловой системой, базами данных.

Немного терминов

Именно изучением серверного программирования мы и займемся, но для этого нужно определить несколько терминов.

Узел – компьютер, подключенный к сети, у которого есть уникальный адрес. В сети Интернет под адресом будем понимать **IP-адрес**. При этом, совершенно не важно что собой представляет узел в сети. С этой точки зрения Интернет можно рассматривать, как множество узлов, каждый из которых потенциально может связаться с любым другим. Не стоит смешивать понятия **узел** и конкретный компьютер. Один компьютер может одновременно обслуживать несколько узлов, т.е. иметь несколько **IP-адресов**. В общем, главное – это адрес.

Сервер – это узел, который содержит данные, используемые другими узлами. Другие узлы связываются с ним для получения данных. Поэтому понятие сервер двояко. С одной стороны – это компьютер (**hardware**), подключенный к сети и доступный другим её узлам. С другой – это программа (**software**), которая управляет доступом к ресурсам компьютера-сервера. В нашем случае будем называть эту программу **web-сервер**. Подробнее о **web-серверах** речь пойдет в [главе 6](#).

Клиент – это узел, обращающийся к серверу для получения доступа к его ресурсам. Под клиентом, как и сервером, также можно одновременно понимать как компьютер (**hardware**), так и программу (**software**). Любой сервер, обращаясь к другому серверу, будет выступать в качестве клиента. Т.е. граница между клиентом и сервером зыбка.

В клиент-серверном взаимодействии в рамках web-программирования, под клиентом как программой мы будем понимать браузер.

Хост – самое сложное понятие. Происходит оно от английского глагола to host – селить, размещать, проводить. Т.е. хост – это место для размещения данных на узле. Всякий узел автоматически становится хостом, но не наоборот, потому, что хост может быть виртуальным и, таким образом, один узел может содержать несколько хостов.

АРХИТЕКТУРА «КЛИЕНТ-СЕРВЕР». ВЫПОЛНЕНИЕ СЕРВЕРНЫХ СЦЕНАРИЕВ. CGI.

СХЕМА ВЗАИМОДЕЙСТВИЯ СЕРВЕРА И КЛИЕНТА

Занимаясь программированием ранее, например, на **Visual C++**, **Delphi**, или даже **Visual Basic**, вы должны были привыкнуть к такой схеме работы программы: происходит событие – выполняется функция (подпрограмма) – выводится результат, и все это выполняется на одном компьютере. В web-программировании все обстоит по-другому.

Давайте рассмотрим подробнее, что происходит, когда мы вводим в адресной строке браузера (клиента) **URL (Universal Resource Locator)**, или в просторечии – адрес). Будут осуществлены такие действия:

- 1 Браузер просит операционную систему найти в сети узел по имени, указанному в **URL** и открыть соединение с ним.
- 2 Браузер открывает соединение с web-сервером по протоколу **http (hyper text transfer protocol)**.
- 3 Браузер отправляет web-серверу запрос на получение страницы, запрошенной в **URL**.
- 4 Web-сервер формирует ответ (чаще всего – **HTML**-код, поскольку общение идет по протоколу **http**, который и создан был специально для передачи гипертекста, т.е. **HTML**) браузеру и закрывает соединение;
- 5 Браузер получает **HTML**-код и занимается рендерингом страницы;

Обратите внимание на четвертый пункт. Еще до того, как вы увидели на экране запрошенную страницу, соединение с сервером закрыто, и он о вас забыл. И когда вы введете другой (даже тот же самый) адрес (**URL**), или щелкните по ссылке, или нажмете на кнопку **HTML**-формы – эта схема повторится снова.

Такую схему работы называют "клиент-сервер". Клиент в данном случае – браузер.

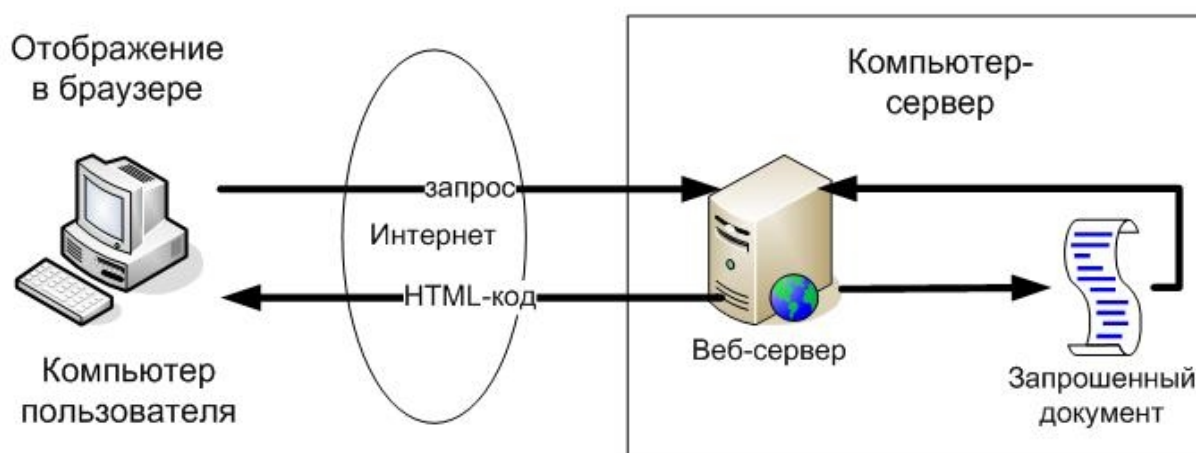
Итак, соединение с web-сервером длится от подтверждения соединения web-сервером до отправки им же последнего символа запрошенного файла. Браузеры во время соединения отображают некий индикатор:



- Mozilla Firefox



- Microsoft Internet Explorer



Схематически это выглядит так, как показано на рисунке.

КАК РАБОТАЮТ СКРИПТЫ НА СЕРВЕРЕ?

Вроде бы не сложно. Но на самом деле немножко сложнее. ☺.

Итак, например, когда мы набираем в нашем браузере:

`http://www.tratatasite.com/folder/subfolder/docfile.ext`

Мы ожидаем, что сейчас получим какой-либо документ (чаще всего, конечно же, мы рассчитываем на «родной» для сайтов формат – **HTML**, но это не обязательно). Иными словами, мы рассчитываем, что на хосте в каталоге `/folder/subfolder/` расположен файл `docfile.ext`, который нам сейчас и доставят по сети.

Однако на самом деле ситуация несколько иная. По двум причинам:

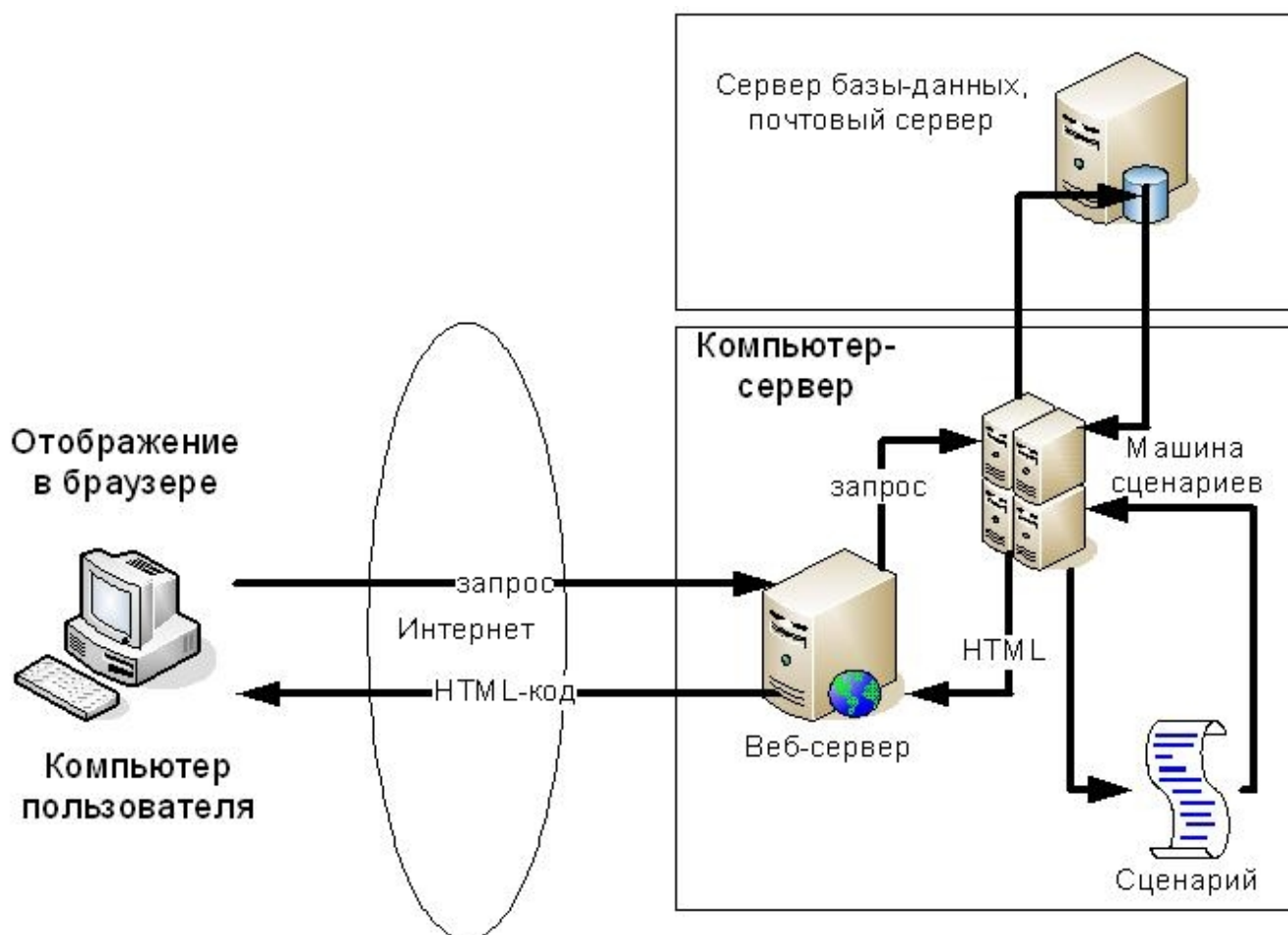
- 1 Путь `/folder/subfolder/`, равно как и файл `docfile.ext` на хосте может вообще не существовать. Т.е. эта часть URL может быть условностью. Администратор сервера может задать псевдоним (**alias**) для любого ресурса на своём сервере.

2 Файл `docfile.ext` может быть вовсе не текстовым документом, а программой, которая в ответ на наш запрос тут же запустится и выполнится, возвратив результаты своей работы, хотя бы в том же **HTML**-формате (или, разумеется, в любом другом, – например, это может быть изображение). Возникает вопрос: «кому возвратит?». Получателем результата будет web-сервер. Ведь это именно он запустил эту программу. А затем web-сервер уже и передаст этот результат запросившему его браузеру по сети.

Последний пункт в общих чертах объясняет суть такого понятия как **CGI – Common Gateway Interface (Общий шлюзовой интерфейс)**.

Как раз **CGI** обеспечивает все то, что выглядит так прозрачно для пользователя. Традиционно программы, работающие в соответствии с соглашениями **CGI**, называют **сценариями (скриптами)**.

Схема взаимодействия клиента и сервера теперь будет выглядеть так:



Это, без сомнения, весьма впечатляет. Ведь, используя **CGI**, мы можем генерировать на сервере, по команде пользователя (находящегося за тысячи километров, например), **HTML** (и не только) любой сложности и отправлять ему.

Например, пусть нам нужно, чтобы в каком-то документе представлялись текущая дата и время. Разумеется, проставить их в документе мы заранее не сможем, поскольку они будут меняться с течением времени. Зато мы можем написать программу, которая вычислит дату, вставит ее в документ в нужное место и затем отправит затребовавшему документ пользователю.

Т.е. сформулируем, что же такое **CGI**:

Это правила взаимодействия внешней программы с web-сервером, суть которых такова: на сервере имеются программы (сценарии), которые будут запускаться по запросу от другой программы – web-сервера, получать данные от него и генерировать результирующий HTML-код, который web-серверу же и возвращается, после чего программа освобождает память и передает управление web-серверу обратно.

ОБЗОР СУЩЕСТВУЮЩИХ СЕРВЕРНЫХ WEB-ТЕХНОЛОГИЙ



На каких же языках пишутся **CGI** сценарии? На любых! На заре web-программирования **CGI** сценарии писались на **C/C++**. Но, очень быстро всем стало ясно, что это не совсем удобно, поскольку язык **C++** не создавался как язык **CGI**-скриптов.

На помощь пришел, написанный в 1987 году для обработки текстов, язык **Perl**. Этот язык стал широко использоваться для создания **CGI**-скриптов, но, очень скоро, с ростом популярности и распространенности web-программирования, появилась потребность в еще более удобных инструментах. На сегодня можно сказать, что максимальную популярность среди разработчиков завоевали:

- 1 **PHP**.
- 2 **ASP.NET** от компании **Microsoft**.
- 3 **Cold Fusion** от компании **Macromedia** (которую в 2005 году поглотил **Adobe Systems**).
- 4 **Ruby on Rails**
- 5 **Java Server Pages** от фирмы **Sun**.

Дальнейшая наша работа и будет связана с детальным рассмотрением возможностей и применения языка программирования **PHP**.

А тем временем рассмотрим вкратце возможности альтернативных технологий серверного web-программирования.

ASP.NET



Это технология создания web-приложений и web-сервисов от компании **Microsoft**. Она является составной частью платформы **Microsoft.NET** и развитием более старой технологии **Microsoft ASP**. На данный момент последней версией этой технологии является **ASP.NET 4.0**.

ASP.NET внешне во многом сохраняет схожесть с более старой технологией **ASP**, что позволяет разработчикам относительно легко перейти на **ASP.NET**. В то же время внутреннее устройство **ASP.NET** существенно отличается от **ASP**, поскольку она основана на платформе **.NET** и, следовательно, использует все новые возможности, предоставляемые этой платформой. [11]

Главным достоинством данной технологии является факт компиляции сценариев, что, безусловно, повышает быстродействие, а, значит, раскрывает большие возможности для разработчика.

Недостатки – это сложность изучения, а также проприетарность (небесплатность) технологии и всего необходимого программного обеспечения.

ColdFusion



ColdFusion объединяет в себе две вещи: сервер приложений и язык называемый **CFML** – **ColdFusion Markup Language**, который основан на тегах, как и **HTML**. Этот язык применяется для генерации **HTML** кода на сервере. Язык **ColdFusion** также может использоваться как надстройка к другим серверам приложений, **J2EE** серверам приложений. В любом случае, язык **CFML** используется для быстрого и легкого создания мощных приложений, которые могут быть запущены как на **ColdFusion** сервере приложений, так и на любом другом. В частности **ColdFusion** может использоваться как под управлением **Apache**, так и под управлением **IIS**.

Язык был создан **JJ Allaire** в 1995 году и до 2005 года поддерживался **Macromedia**, но на данный момент продуктом владеет **Adobe**.

Достоинство технологии состоит в том, что она довольно проста, благодаря тому, что основана на тегах, но, **ColdFusion** – это также проприетарное программное обеспечение.

RUBY ON RAILS

Объектно-ориентированный программный каркас для создания web-приложений на языке программирования **Ruby**. **Ruby on Rails** предоставляет архитектурный образец **Model-View-Controller** (модель-представление-контроллер) для web-приложений, а также обеспечивает их интеграцию с web-сервером и сервером базы данных. Предоставляет однородную среду для разработки динамических **AJAX**-интерфейсов, с обработкой запросов и выдачи данных в контроллерах, отражения предметной области в базе данных.



Ruby on Rails является открытым программным обеспечением и распространяется под лицензией **MIT**. Был создан Дэвидом Хейнмеером Ханссоном на основе его работы над средством управления проектами **Basecamp** и выпущен в июле 2004 года. Последней версией является 2.3.8.

Ruby on Rails может работать как с **Apache** так и другими web-серверами. Для разработки и отладки часто используется встроенный в **Ruby** web-сервер. Поддерживаются СУБД **MySQL**, **Firebird**, **PostgreSQL**, **IBM DB2**, **Oracle** и **Microsoft SQL Server**. Также поддерживается встраиваемая база данных **SQLite**.

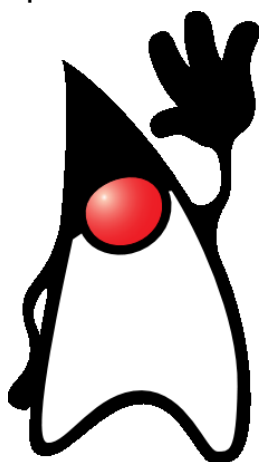
Для **Windows** существует дистрибутив Instant Rails с настроенной и готовой к работе сразу после установки рабочей средой для разработки **Rails**-приложений, которая включает в себя сервер **Apache** и СУБД **MySQL**. Для платформ **Windows**, **Linux**, **MacOSX** имеется комплексный установщик **BitNami RubyStack**, включающий в себя все необходимое для разработки в среде **Rails**, включая **Ruby**, **RubyGems**, **Ruby on Rails**, **MySQL**, **Apache**, **Mongrel** и **Subversion**.

Для разработки Ajax-приложений RoR поставляется с интегрированными **JavaScript**-библиотеками **Prototype** и **Script.aculo.us**, но также можно использовать и **jQuery**.^[2]

Отметить можно весьма динамичное развитие технологии и числа разработчиков.

Скачать дистрибутив можно по адресу: <http://www.rubyonrails.ru/>

JSP



Java Server Pages — технология, позволяющая web-разработчикам динамически генерировать **HTML**, **XML** и другие web-страницы. Не является составной частью единой технологии создания бизнес-приложений **Java EE**, т.к. может использоваться отдельно, а технология **Java EE** может использоваться без использования **JSP**. Технология позволяет внедрять **Java**-код, а также **EL** (**expression language**) в статичное содержимое страницы. Также могут использоваться библиотеки **JSP** тегов для внедрения их в **JSP**-страницы. Страницы компилируются **JSP**-компилятором в сервлеты, представляющие собой **Java**-классы, которые выполняются на сервере. Сервлеты также могут быть написаны разработчиком, не используя **JSP**-страницы. Эти технологии могут дополнять друг друга.

JSP — одна из высокопроизводительных технологий, так как весь код страницы транслируется в **java**-код сервлета с помощью компилятора **JSP** страниц **Jasper**, и затем компилируется в байт-код виртуальной машины **java** (**JVM**). Сервлет-контейнеры (**Tomcat**), способные исполнять **JSP** страницы, написаны на платформонезависимом языке **Java**, который может работать под различными операционными системами и платформами.

КРАТКАЯ ИСТОРИЯ PHP. ОБЗОР ВЕРСИЙ И ВОЗМОЖНОСТЕЙ.

Что такое PHP и область применения.

PHP – интерпретируемый язык программирования, созданный для генерирования **HTML**-страниц на web-сервере и работы с базами данных. В настоящее время поддерживается подавляющим большинством хостингов. Входит в **LAMP** – «стандартный» набор для создания web-сайтов (**Linux**, **Apache**, **MySQL**, **PHP**).

В области программирования для Сети, **PHP** – один из популярнейших скриптовых языков (наряду с **JSP**, **Perl** и языками, используемыми в **ASP.NET**) благодаря своей простоте, скорости выполнения, богатой функциональности и распространению исходных кодов на основе лицензии **PHP**.



PHP отличается наличием ядра и подключаемых модулей, «расширений»: для работы с базами данных, сокетами, динамической графикой, криптографическими библиотеками, документами формата **PDF** и т. п. Любой желающий может разработать своё собственное расширение и подключить его. Существуют сотни расширений, однако в стандартную поставку входит лишь несколько десятков хорошо зарекомендовавших себя. ^[3]

Можно констатировать, что, на сегодня, **PHP** – самый популярный и распространенный язык web-программирования. Абсолютное большинство web-проектов написаны именно на нём.

История PHP и обзор ранних версий [4]

Истоки **PHP** лежат в старом продукте, имевшем название **PHP/FI**. Последний был создан Расмусом Лерддорфом (**Rasmus Lerdorf**) в 1994 году и представлял собой набор **Perl**-скриптов для ведения статистики посещений его резюме. Этот набор скриптов был назван '**Personal Homepages Tools**' (Инструменты для персональных домашних страниц). Очень скоро потребовалась большая функциональность и Расмус пишет новую, намного более обширную вер-

сию на **C**, работающую с базами данных и позволяющую пользователям разрабатывать простейшие web-приложения. Расмус решил выложить исходный код **PHP/FI** на всеобщее обозрение, исправление ошибок и дополнение. Такая практика, зарождавшись тогда, весьма развита и сегодня.

PHP/FI (**Personal Home Page** / **Forms Interpreter** – Персональная Домашняя страница / Интерпретатор Форм) включал в себя базовую функциональность сегодняшнего **PHP**. Он имел переменные в стиле **Perl**, автоматическую интерпретацию web-форм и возможность встраиваться в **HTML**-код. Собственно синтаксис языка имел много общего с **Perl**, хотя и был намного проще и ограниченнее.

В 1997 выходит **PHP/FI 2.0**, вторая версия тоже написана на **C**. Она обозначила группу пользователей: несколько тысяч людей по всему миру, с примерно 50000 доменами, что составляло около 1% всего числа доменов Интернета. Несмотря на то, что разработкой занималось уже несколько людей, **PHP/FI 2.0** все еще оставался крупным проектом одного человека.

Официально **PHP/FI 2.0** вышел только в ноябре 1997 года, после проведения большей части своей жизни в β -версиях. Вскоре после выхода его заменили α -версии **PHP 3.0**.

PHP 3.0 [4]

PHP 3.0 была первой версией, напоминающей **PHP**, каким мы знаем его сегодня. В 1997 году Энди Гутманс (**Andi Gutmans**) и Зив Сураски (**Zeev Suraski**) переписали код с начала: разработчики сочли **PHP/FI 2.0** не пригодным для разработки приложения электронной коммерции, над которым они работали для проекта израильского технологического института, расположенного в Хайфе. Для совместной работы над **PHP 3.0** с помощью базы разработчиков **PHP/FI 2.0** Энди, Расмус и Зив решили объединиться и объявить **PHP 3.0** официальным преемником **PHP/FI**, разработка же **PHP/FI** была практически полностью прекращена.

Одной из сильнейших сторон **PHP 3.0** была возможность расширения ядра. В последствии интерфейс написания расширений привлек к **PHP** множество сторонних разработчиков, работающих над своими модулями, что дало **PHP** возможность работать с огромным

количество баз данных, протоколов, поддерживать большое число **API**. Фактически, это и был главный ключ к успеху, но стоит добавить, что немаловажным шагом оказалась разработка нового, намного более мощного и полного синтаксиса с поддержкой ООП.

Абсолютно новый язык программирования получил новое имя. Разработчики отказались от дополнения о персональном использовании, которое имелось в аббревиатуре **PHP/FI**. Язык был назван просто '**PHP**' – аббревиатура, содержащая рекурсивный акроним: '**PHP: Hypertext Preprocessor**' (**PHP**: Препроцессор Гипертекста).

К концу 1998, **PHP** использовался десятками тысяч пользователей. Сотни тысяч web-сайтов сообщали о том, что они работают с использованием языка. В то время **PHP 3.0** был установлен приблизительно на 10% серверах Интернета. **PHP 3.0** был официально выпущен в июне 1998 года после 9 месяцев публичного тестирования.

К зиме 1998 года, практически сразу после официального выхода **PHP 3.0**, Энди Гутманс и Зив Сураски начали переработку ядра **PHP**. В задачи входило увеличение производительности сложных приложений и улучшение модульности кода **PHP**. Расширения дали **PHP 3.0** возможность успешно работать с набором баз данных и поддерживать большое количество различных **API** и протоколов, но **PHP 3.0** не имел качественной поддержки модулей и приложения работали не эффективно.

PHP 4.0 [4]



The PHP Company

Новый движок, названный '**Zend Engine**' (от имен создателей: **Zeev** и **Andi**), успешно справлялся с поставленными задачами и впервые был представлен в середине 1999 года. **PHP 4.0**,

основанный на этом движке и принесящий с собой набор дополнительных функций, официально вышел в мае 2000 года, почти через два года после выхода своего предшественника **PHP 3.0**. В дополнение к улучшению производительности, **PHP 4.0** имел еще несколько ключевых нововведений, таких как поддержка сессий, буферизация вывода, более безопасные способы обработки вводимой пользователем информации и несколько новых языковых конструкций.

Прекращение выпуска обновлений **PHP 4** было запланировано на конец 2007 года. Однако вплоть до 8 августа 2008 года выпускались критические обновления безопасности. С 9 августа 2008 года всякая поддержка версии **PHP 4.x** была прекращена.

PHP 5.0 [4]

Пятая версия **PHP** была выпущена разработчиками 13 июля 2004 года. Изменения включают обновление ядра **Zend (Zend Engine 2)**, что существенно увеличило эффективность интерпретатора. Введена поддержка языка разметки **XML**. Полностью переработаны функции ООП, которые стали во многом схожи с моделью, используемой в Java. Нововведения, однако, были сделаны с расчётом сохранить наибольшую совместимость с кодом на предыдущих версиях языка.

Шестая версия **PHP** находится в стадии разработки с октября 2006 года. В ней уже сделано множество нововведений. Также много внимания уделено поддержке Юникода.

ПЛАТФОРМА PHP & MYSQL. ДОСТОИНСТВА И НЕДОСТАТКИ.

Базы данных для WEB-ПРОГРАММИРОВАНИЯ

Здесь много говорилось о работе **PHP** с базами данных. Почему это так важно? На сегодня, именно реляционные базы данных – это самый распространенный инструмент для длительного и упорядоченного хранения информации, что, по сути, и является основой-основ Интернет как информационной среды. Т.е. возможность работы с базами данных для языков web-программирования – это не «дополнительная возможность», а насущная необходимость, без которой разработка современных web-сайтов немыслима.



Действительно **PHP** способен взаимодействовать с большим количеством реляционных баз данных, но главенствующее положение среди них занимает СУБД **MySQL** – продукт скандинавской компании **MySQLAB**.

Достоинства и недостатки

Одним из главных достоинств связки **PHP & MySQL** является их **свободное распространение**. Действительно интерпретатор **PHP** распространяется бесплатно и его можно свободно скачать с сайта php.net и установить на любой машине. СУБД **MySQL** (mysql.com) тоже является бесплатной для использования в связке с **PHP**. Такого рода программное обеспечение называют **open-source software**.



Сильной стороной является **простота языка PHP** (по моему мнению, он гораздо проще, чем **Java Script** ☺. Скоро вы сами в этом, я надеюсь, убедитесь) и, соответственно, легкость во взаимодействии с **MySQL**.

Не забываем о требованиях к аппаратным и программным средствам, необходимым для работы **PHP & MySQL**. Эти требования весьма низки. Практически любая машина, на которой может быть установлена Windows 2000 и выше или операционная система се-

мейства Unix может эффективно обслуживать web-разработку на базе связки **PHP & MySQL**. Как следствие, язык **PHP** в связке с **MySQL** поддерживает **достаточно высокое быстродействие** и отличную надежность и стабильность.

Что до достоинств, то они в целом, очевидны. А какие же недостатки? Как ни странно, один из недостатков запросто может восприниматься некоторыми программистами как достоинство! Это **отсутствие средств графической визуальной разработки** по принципу **WYSIWYG**, как например в **Visual Studio** или **Microsoft Access**. Т.е. для эффективной работы с **PHP & MySQL** практически весь код нужно писать вручную. Зато взамен мы получаем оптимизацию кода, управляемость, надежность и «красоту».

Следствием из описанной проблемы есть **трудоёмкость при разработке** сложных и серьёзных проектов, требующих использования средств разделения шаблонов результирующего **HTML**-кода и управляющей логики. Но к счастью, эта проблема довольно успешно решается общими усилиями разработчиков **PHP** по всему миру.

ОБЗОР ИНСТРУМЕНТАРИЯ.

Что же из программных средств необходимо для создания, отладки и тестирования сценариев на **PHP**?

Вернёмся к схеме работы сервера и клиента по обмену данными и разберем устройство сервера.

Платформа, на которой функционирует сервер, Интернет может быть описана 4-мя компонентами:

- 1 **Операционная система**. Изначально **PHP** разрабатывался для работы в среде **UNIX**, но (хорошая новость!) сейчас **PHP** успешно интегрируется в среду **Windows**.

Таким образом, для того чтобы изучать **PHP**, нам нет необходимости устанавливать **UNIX**-систему, к примеру, дома. Мы прекрасно сможем обойтись **Windows XP** или выше. Т.е. упомянутая выше платформа **LAMP** превращается в **WAMP**. ☺

Но хочу отметить, что это справедливо для учебных и отладочных целей. Реальный хостинг на **PHP** реализуется как раз таки на базе **UNIX**-систем.

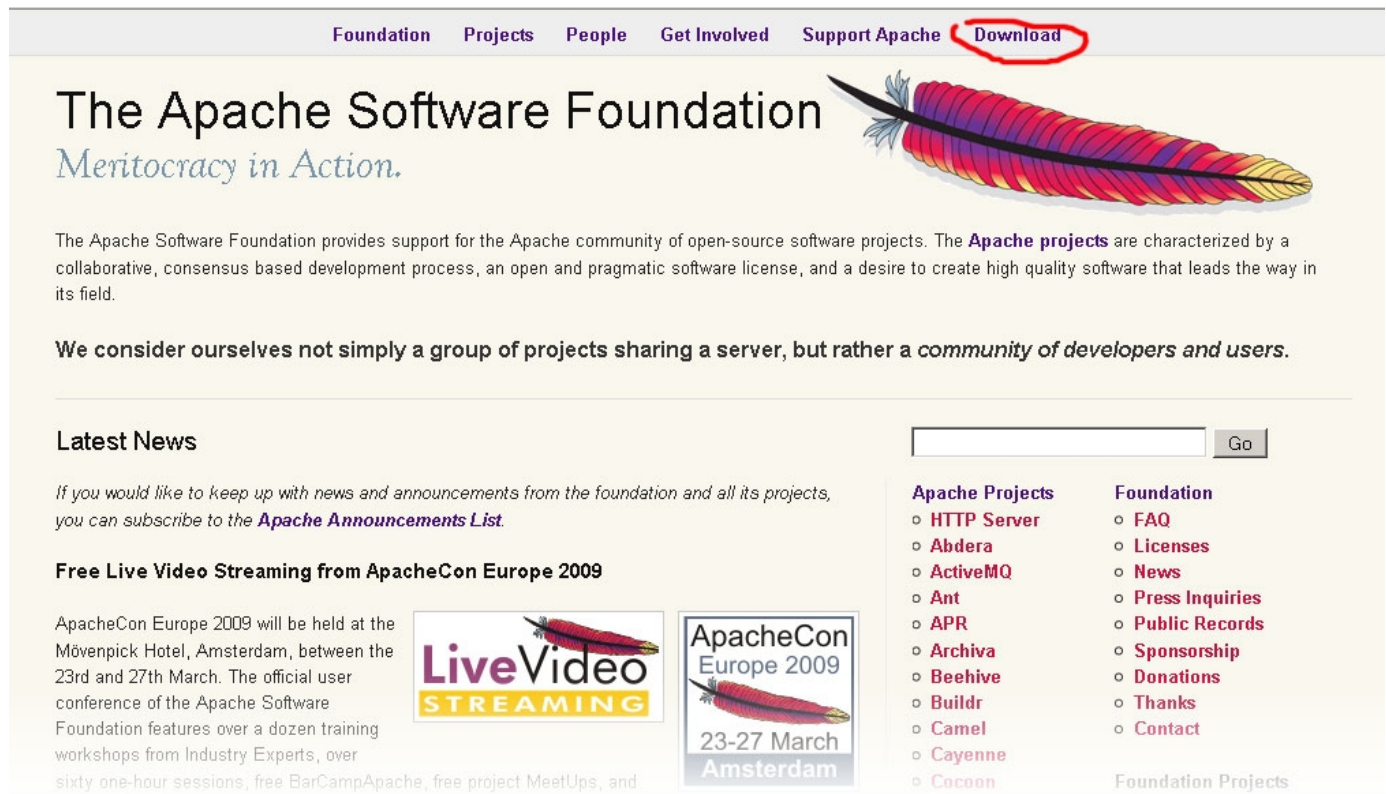
Установку, настройку и администрирование операционной системы мы здесь обсуждать не будем. Это – тема отдельного (и надо сказать весьма объёмного) курса.

- 2 **Web-сервер**. Чуть раньше ([в гл. 1](#)) давалось определение web-сервера. Какие же они бывают?

На сегодня двумя наиболее распространёнными web-серверами, вместе занимающими около 85% рынка, являются:

- **Apache HTTP Server** – бесплатно распространяющийся (**open source**) web-сервер, разрабатываемый с 1995 года силами **Apache Software Foundation**. Наиболее часто используется в **Unix**-подобных ОС, но, тем не менее, без особых проблем работает под управлением **Windows**. Основной упор сделан на гибкость и надёжность в эксплуатации.
- **Internet Information Services (IIS)** от компании **Microsoft**, распространяется с ОС семейства **Windows NT** с 1995 года. Работает только под управлением **Windows**.

PHP в настоящее время может работать как в связке с **Apache**, так и в связке с **IIS**. Поскольку исторически **PHP** все таки ориентирован на **Apache**, то и мы тоже будем его использовать, тем более то он бесплатен. ☺. Его легко можно скачать с сервера apache.org.



The screenshot shows the Apache Software Foundation website. At the top, there is a navigation bar with links: Foundation, Projects, People, Get Involved, Support Apache, and Download (which is circled in red). The main heading is "The Apache Software Foundation" with the tagline "Meritocracy in Action." Below this, there is a paragraph about the foundation's support for open-source software projects. A large, colorful feather graphic is on the right. Further down, there is a "Latest News" section with a link to the Apache Announcements List. Below that, there is a section for "Free Live Video Streaming from ApacheCon Europe 2009" with a date of 23-27 March in Amsterdam. On the right side, there are two columns of links: "Apache Projects" (including HTTP Server, Abdera, ActiveMQ, Ant, APR, Archiva, Beehive, Buildr, Camel, Cayenne, Cocoon) and "Foundation" (including FAQ, Licenses, News, Press Inquiries, Public Records, Sponsorship, Donations, Thanks, Contact). There is also a "Foundation Projects" link at the bottom right.

На сегодня используются **Apache** версий 1.3.x.x и 2.0.x.x.

Выбирайте тот файл пакета, который имеет имя подобное `apache_2.2.14-win32-x86-openssl-0.9.8k.msi` (где **xx**-текущая версия **Apache**).


3 Интерпретатор языка (машина сценариев). Это, как вы понимаете, то, ради чего мы здесь и собрались, поэтому ответ очевиден – **PHP** любой версии, начиная с 5.0. (04.03.2010 вышел **PHP** 5.3.2 (for **Windows**)).

Источником является, опять таки, сайт php.net, хотя существует множество других ресурсов, где можно скачать инсталляционный пакет **PHP**. Это, как правило, архив бинарных файлов **PHP5** в формате `*.zip`. Не нужно скачивать **PHP** в виде инсталляционных пакетов (типа `*.msi`).

Phpclub.ru:

PHP - рекурсивный акроним словосочетания «PHP: Hypertext Preprocessor»

[Создать сайт](#)



За последние 24 часа нас посетили 2424 программиста и 4 робота. Сейчас занимаются поиском восемь перцев ...

[Главная](#) [Скачать](#) [Документация](#) [Форум](#)

[phpinfo\(\)](#)

Сегодня:

- 10:58 — [Live Search](#) представил два новых Q&A сервиса
- 10:38 — [Nokia](#) запустит Ovi в России летом
- 10:29 — Директора и основатели [Google](#) получили за год по доллару
- 10:25 — Лазерное оружие становится все реальной
- 10:24 — [BFG](#) выпустила энергоэффективный БП мощностью 1200 ватт
- 10:24 — [Hitachi](#) выпустил скоростной жесткий диск нового поколения (1)
- 10:22 — [LG Lollipop](#): телефоны, меняющие внешний облик
- 10:19 — Поисковик [Google](#) поумнел
- 10:17 — Поддельные антивирусы приносят мошенникам до \$10 тысяч ежедневно
- 10:16 — [ВТБ](#) выдал первый транш ЗАО "Байкалвестком" в размере \$4.3 млн
- 10:10 — [Sun VDI 3](#) - несколько виртуальных сред на одном тонком клиенте
- 10:10 — Зарплата гендиректора и двух основателей [Google](#) за 2008 год составила \$3 (2)
- 10:09 — Как избежать отставок владельцев устройств

PHP — официальные новости:

- PHP 5.2.6 [final](#)
- Вышел последний [phpMyAdmin](#) 2.9.0
- Вышел последний [release candidate](#) перед 5.2.0
- Вышел [релиз](#) PHP 5.1.6
- Представлен новый [релиз](#) PHP, версии 4.4.4. и 5.1.5

Установить:

- [Подробная установка](#) связки Apache 2.0 + Php 5.1.4 + Mysql Server 4.1.16 под Win32 платформу

Документация:

- Руководство по [Apache 2](#)
- Руководство по [PHP](#)
- Руководство по [MySQL](#)

Скачать:

- PHP 5.2.9 Новый релиз! для Windows. Последняя версия!**
- PHP 5.2.1 для Windows.
- PHP 5.2.0 для Windows.

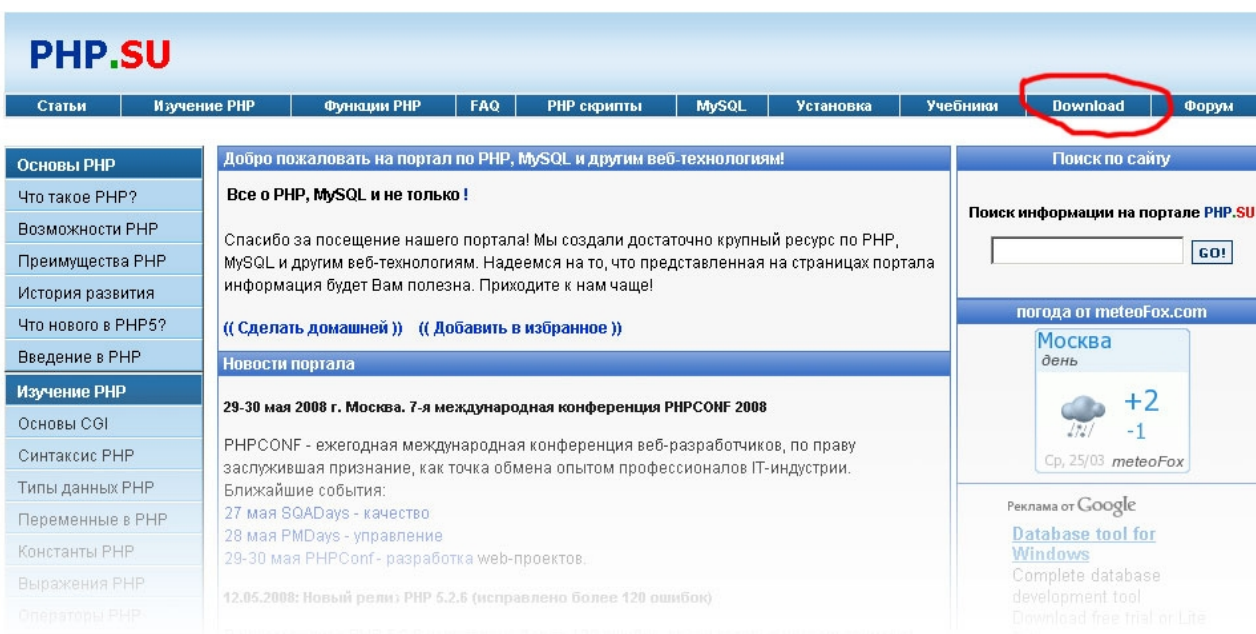
Реклама от Google

[Утилиты для MySQL](#)
Бесплатные мощные утилиты для администраторов и разработчиков
[www.devart.com](#)

[Хостинг от \\$0.99 / месяц](#)
Закажи хостинг PHP + MySQL и получи доменное имя бесплатно!
[www.abris.com.ua](#)

[Сайт](#)
Маркетингово
Правильные
Сайты Сайты

Php.su:



PHP.SU

Статьи | Изучение PHP | Функции PHP | FAQ | PHP скрипты | MySQL | Установка | Учебники | **Download** | Форум

Основы PHP

- Что такое PHP?
- Возможности PHP
- Преимущества PHP
- История развития
- Что нового в PHP5?
- Введение в PHP

Изучение PHP

- Основы CGI
- Синтаксис PHP
- Типы данных PHP
- Переменные в PHP
- Константы PHP
- Выражения PHP
- Операторы PHP

Добро пожаловать на портал по PHP, MySQL и другим веб-технологиям!

Все о PHP, MySQL и не только !

Спасибо за посещение нашего портала! Мы создали достаточно крупный ресурс по PHP, MySQL и другим веб-технологиям. Надеемся на то, что представленная на страницах портала информация будет Вам полезна. Приходите к нам чаще!

((Сделать домашней)) ((Добавить в избранное))

Новости портала

29-30 мая 2008 г. Москва. 7-я международная конференция PHPCONF 2008

PHPCONF - ежегодная международная конференция веб-разработчиков, по праву заслужившая признание, как точка обмена опытом профессионалов IT-индустрии.

Ближайшие события:

- 27 мая SQADays - качество
- 28 мая PMDays - управление
- 29-30 мая PHPConf - разработка web-проектов.

12.05.2008: Новый релиз PHP 5.2.6 (исправлено более 120 ошибок)

Поиск по сайту

Поиск информации на портале PHP.SU

погода от meteoFox.com

Москва
День
+2
-1
Ср, 25/03 meteoFox

Реклама от Google

[Database tool for Windows](#)
Complete database development tool
Download free trial or Life

А также и другие. Кроме того, на этих сайтах вы найдете сборники примеров, учебники, руководства и переводы документации, касающейся технологий работы с **PHP**.

4 **СУБД**. Полноценное использование реляционных баз данных для написания серверных скриптов будет рассматриваться в нашем курсе немного позже, но установить СУБД **MySQL** не ме-

шает уже сейчас. Ввиду свободного распространения данного ПО, его можно скачать либо с официального сервера либо с других сайтов, в частности и с тех, которые приведены выше.

УСТАНОВКА ИНСТРУМЕНТАРИЯ

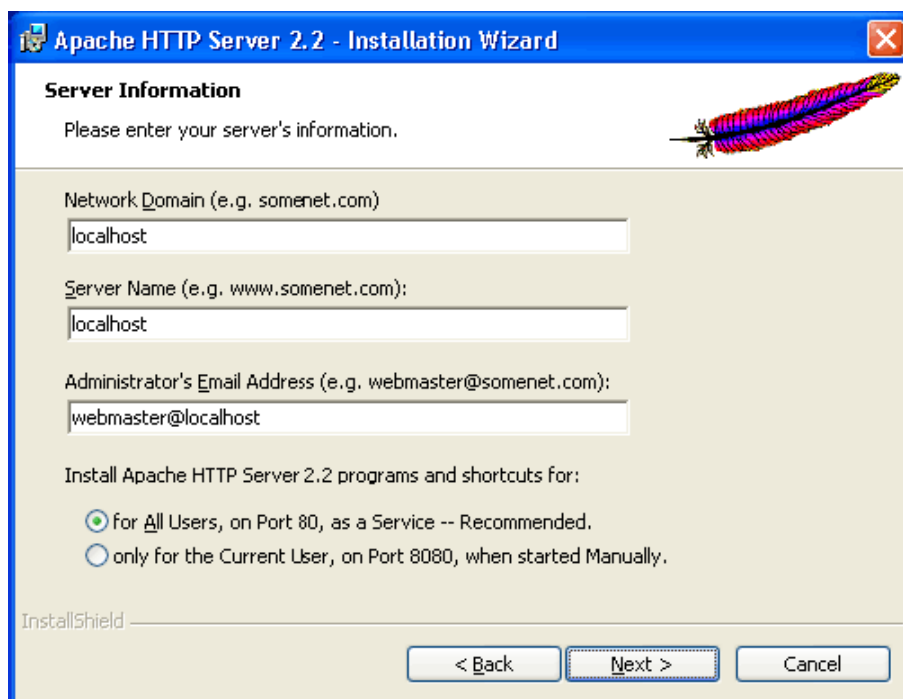
УСТАНОВКА Apache

Благодаря наличию бинарных дистрибутивов для **Apache** и **PHP**, скомпилированных под **Windows**, установить и конфигурировать **Apache 2.0.xx** для использования совместно с **PHP 5** в **Windows** не сложно.

Для начала, нужно убедиться в том, что порт 80 свободен. Он может быть занят какими-либо сервисами или приложениями, использующими Интернет для получения и передачи данных, т.е.: **ICQ**, **QIP**, **Skype** и другие, либо другой web-сервер, например, **Microsoft IIS**. Если такие приложения у Вас есть, то их нужно либо отключить либо удалить.

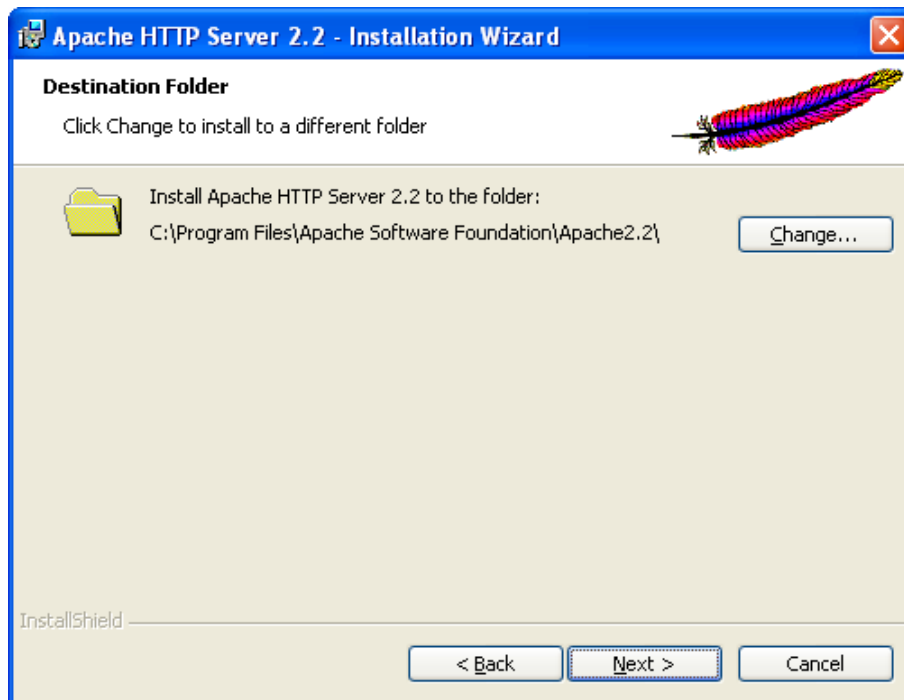
Кроме того, вы должны быть администратором на своей машине, т.е. обладать правами записи на системный диск и правами установки программного обеспечения.

Установка **Apache** проходит в стиле next-next. В начале установки нужно установить следующие параметры:



В качестве директории для установки выбираем директорию по

умолчанию:



Теперь распакуем архив *.zip с дистрибутивом **PHP 5** в папку `C:/php`, которую и создаём специально для этого.

УСТАНОВКА PHP

После установки **Apache**, возле часов появится значок в виде пера. Щелчком правой кнопки мыши на нём выбираем "Open Services". В открывшемся окне управления службами устанавливаем для службы **Apache** ручной тип запуска.

Создаём в корне диска `C:`, т.е. диска, куда мы установили сервер, каталог `/apache` – в нём будет находиться всё что нужно будет для функционирования сервера и все скрипты. В каталоге `/apache` нужно создать еще одну пустую папку – `localhost`, в которой, в свою очередь, – папку `www`. Собственно она и нужна будет нам для скриптов

Конечно же, такая структура каталогов не обязательна. В можете самостоятельно выстраивать каталоги. Главное – правильная конфигурация. Нижеприведенные примеры построены, исходя из предложенной структуры папок.

Теперь приступим к конфигурированию. Для этого, в директории установки **Apache** и поддиректории `conf` найдем файл `httpd.-`

`conf` и откроем его в блокноте.

Конфигурирование

Для загрузки **PHP** интерпретатора, в конец блока загрузки модулей необходимо добавить строку:

```
LoadModule php5_module "C:/php/php5apache2_2.dll"
```

Эта строка подключит интерпретатор **PHP** к **Apache**-серверу в качестве модуля, что в значительной мере повышает быстродействие. Сокращая время связи между ними, поскольку теперь они будут работать в рамках одного процесса.

Кроме того, нам понадобится модуль `mod_rewrite`. Для его загрузки раскомментируем (т.е. уберём #) строку:

```
LoadModule rewrite_module modules/mod_rewrite.so
```

Определение каталога, содержащего конфигурационный файл **PHP** (о нём ниже) осуществляем путём добавления ниже следующей строки:

```
PHPIniDir "C:/php"
```

Кроме того, выполним следующие действия:

- Раскомментируем строку: `ServerName localhost:80`

- Вместо строки:

```
DocumentRoot      "C:/Program      Files/Apache      Software  
Foundation/Apache2.2/htdocs"
```

впишем:

```
DocumentRoot "C:/apache"
```

- Блок:

```
<Directory />  
    Options FollowSymLinks  
    AllowOverride None  
    Order deny,allow  
    Deny from all  
</Directory>
```

заменим на нижеследующий:

```
<Directory />  
    Options Includes Indexes FollowSymLinks  
    AllowOverride All  
    Allow from all  
</Directory>
```

- А вот первоначальный блок управления нужно либо удалить, либо закомментировать. Он выглядит так:

```
<Directory "C:/Program Files/Apache Software  
Foundation/Apache2.2/htdocs">  
    Options Indexes FollowSymLinks  
    AllowOverride None  
    Order allow,deny  
    Allow from all  
</Directory>
```

- Блок:

```
<IfModule dir_module>  
    DirectoryIndex index.html  
</IfModule>
```

Заменяем на:

```
<IfModule dir_module>  
    DirectoryIndex index.html index.htm index.shtml index.php  
</IfModule>
```

- Строку: `ErrorLog "logs/error.log"` заменим на нижеследующую (в этом случае просматривать глобальный файл ошибок сервера будет удобнее): `ErrorLog "C:/apache/error.log"`
- Строку: `CustomLog "logs/access.log" common` заменим на: `CustomLog "C:/apache/access.log" common`
- Для работы **SSI** (включения на стороне сервера) следующие строки, находящийся в блоке `<IfModule mime_module>`, необходимо найти и раскомментировать:

```
AddType text/html .shtml  
AddOutputFilter INCLUDES .shtml
```

- Ниже, в тот же блок, добавим `<IfModule mime_module>`, две строки:

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

- Раскомментируем строки:

```
Include conf/extra/httpd-mpm.conf
```



```
Include conf/extra/httpd-autoindex.conf
Include conf/extra/httpd-vhosts.conf
Include conf/extra/httpd-manual.conf
Include conf/extra/httpd-default.conf
```

Теперь в файле `httpd-vhosts.conf` из директории `conf\extra` проведем изменение существующих блоков примеров виртуальных хостов. Вместо них нужно вписать следующее:

```
<VirtualHost *:80>
    DocumentRoot "C:/apache/localhost/www"
    ServerName localhost
    ErrorLog "C:/apache/localhost/error.log"
    CustomLog "C:/apache/localhost/access.log" common
</VirtualHost>
```

Теперь самое время осуществить пробный запуск. Запустим сервис **Apache**, воспользовавшись меню, всплывающим над значком



пера в системном трее: . Если сервер стартовал, то значит, при конфигурировании фатальных ошибок для настроек сервера допущено не было.

Тестирование работоспособности **PHP** проведем путем запуска простенького скрипта, который нужно сохранить в папку `www` под именем `index.php`:

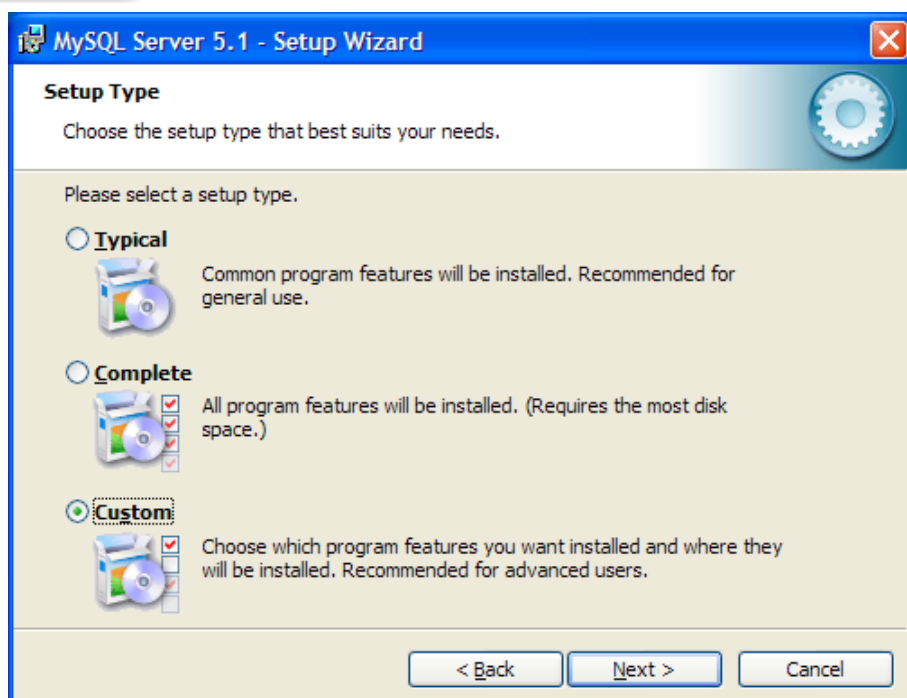
ПРИМЕР 7.3.1

```
<?php
    print "PHP version: ".phpversion();
?>
```

Запускаем браузер, набрав в адресной строке `localhost`. В результате мы должны увидеть номер версии интерпретатора.

УСТАНОВКА MySQL и ДОПОЛНИТЕЛЬНОЕ КОНФИГУРИРОВАНИЕ PHP

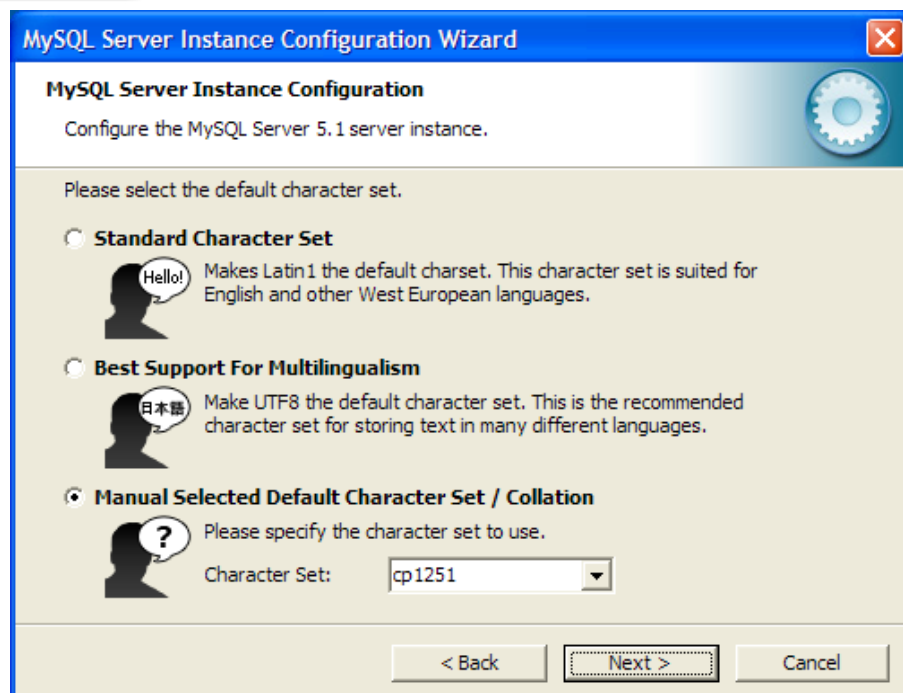
Итак, теперь нам нужно установить **MySQL**. Скачанный дистрибутив оснащен инсталлятором и, таким образом, при установке нам нужно следовать его указаниям, нажимая на кнопку Next:



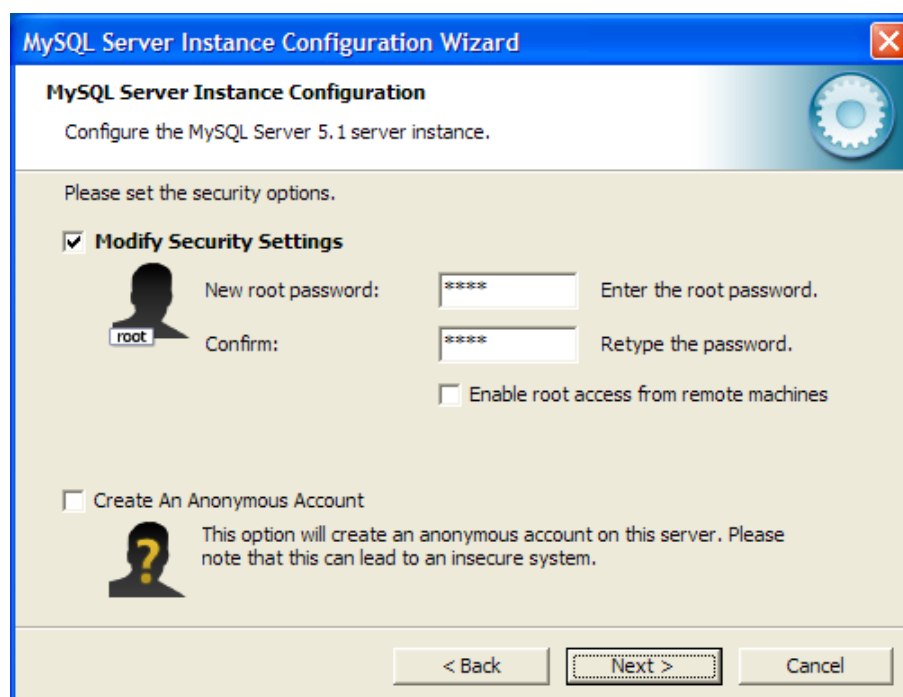
Сразу после установки инсталлятор предложит приступить к конфигурированию:



Большая часть настроек конфигурации вполне может быть сохранена исходном вариант. Изменить нужно будет кодировку по умолчанию:



, а также установить пароль администратора, т.е. пользователя с логином `root`:



Вот, собственно, и все. Теперь нужно заняться внесением изменений в файл настроек **PHP** с целью того, чтобы интерпретатор мог связываться с сервером баз данных **MySQL**.

Файл настроек **PHP** называется `php.ini` и поставляется в дистрибутиве **PHP** он в двух вариантах: `php.ini-development` и

`php.ini-production`. Как следует из названий, один из файлов оптимизирует интерпретатор для разработки, другой – для использования в качестве рабочего модуля. Файл `php.ini-development` переименовываем в `php.ini`. Его правкой мы и займёмся.

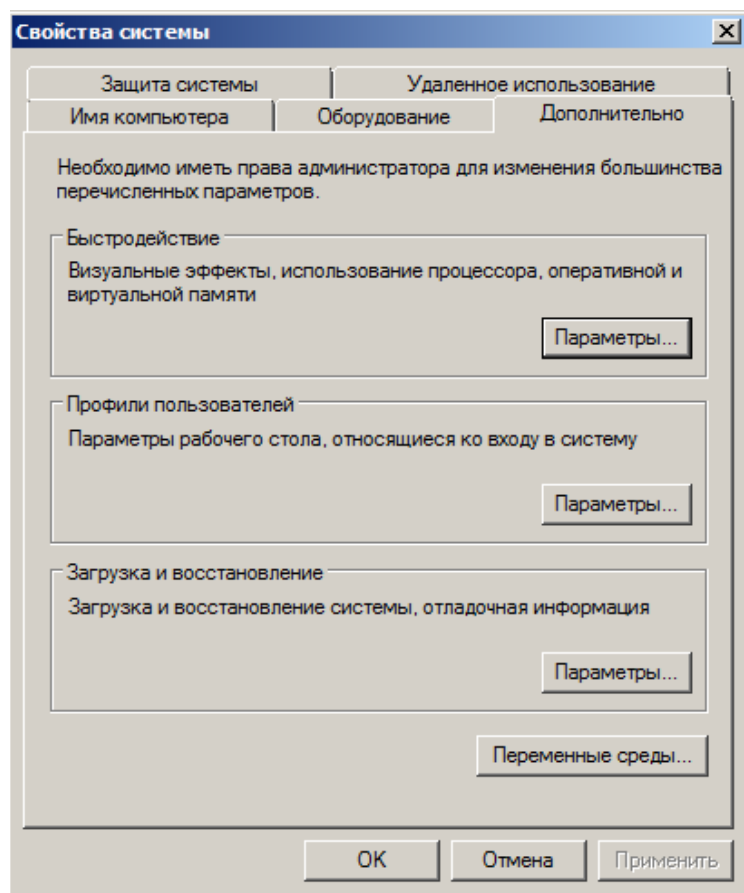
Нужно найти примерно следующую строку:

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

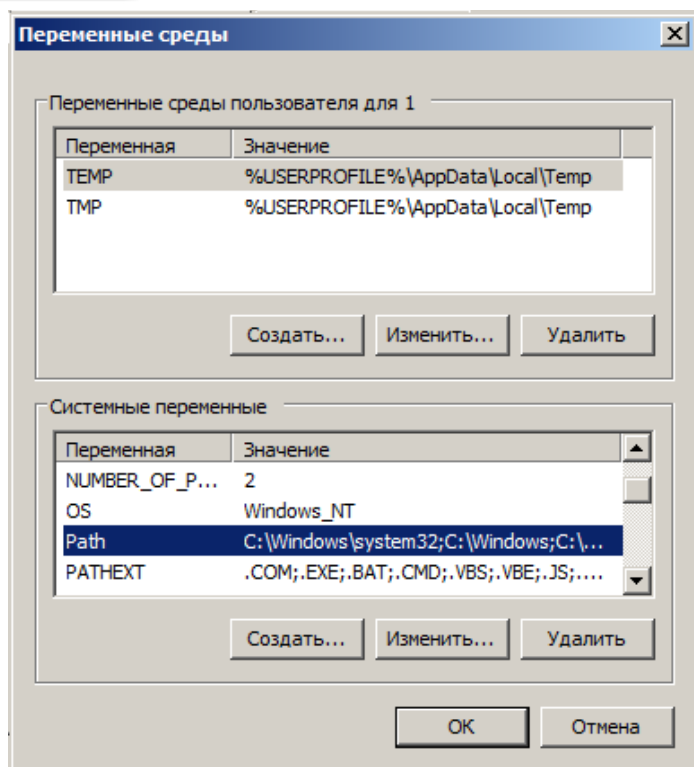
Поскольку перед ней стоит `;`, то эта директива закомментирована. Раскомментируем её и, тем самым, включим поддержку **MySQL** в **PHP5**.

Теперь нам придется добавить каталог **PHP** в PATH операционной системы. Для этого нужно внести изменения в системную переменную. Добраться до неё можно следующим способом:

Контекстное меню на кнопке «Мой компьютер», где выберем «Свойства». Далее вкладка «Дополнительно»:



Теперь нажимаем кнопку «Переменные среды...»:



Двойной щелчок на строке "Path". В поле "Значение переменной" нужно к тому что там уже существует, добавить путь к каталогу с установленным **PHP**, т.е.:

`C:\php`

Обратите внимание на то, что пути разделяет точка с запятой. Обязательно нужно установить в переменных среды путь к **PHP** перед уже имеющимся там путем к каталогу `bin` установочной директории **MySQL**. Чтобы внесенные изменения вступили в силу нужна будет перезагрузка ОС.

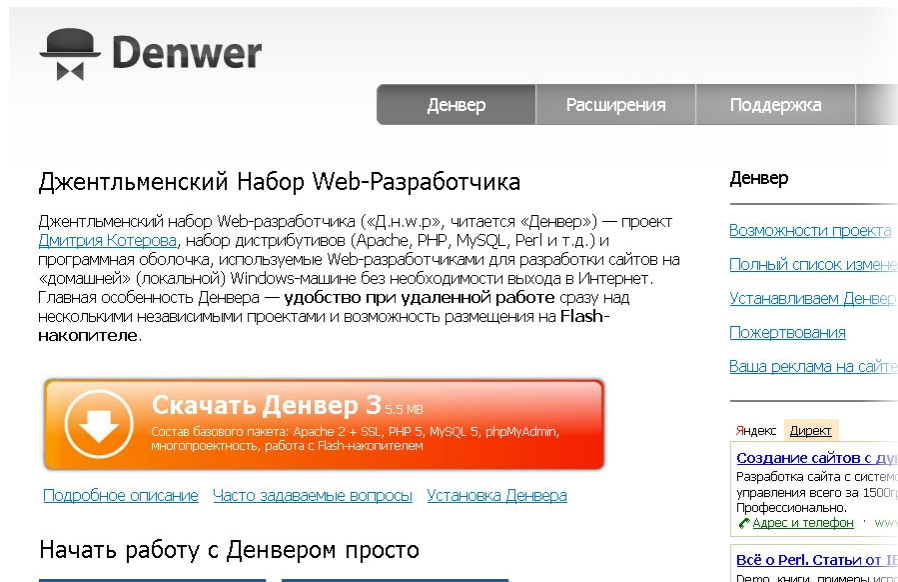
Вот теперь совсем всё. ☺

А сегодня я бы хотел упомянуть еще вот о чем.

УПРОЩЕННАЯ УСТАНОВКА ПЛАТФОРМЫ **WAMP**.

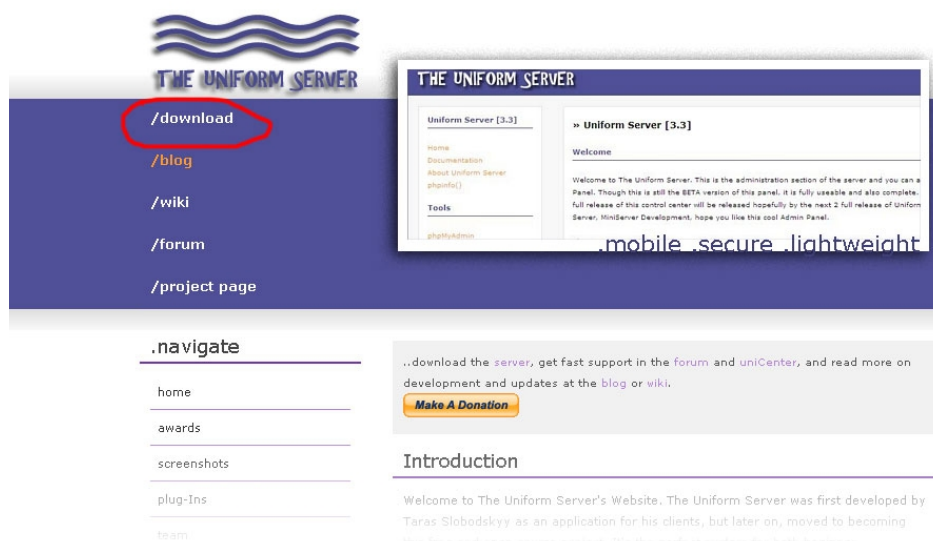
Задачу по установке и настройке связки **WAMP** можно упростить, если воспользоваться готовыми решениями. Одним из наиболее популярных и удобных пакетов, который автоматически создаст готовую полнофункциональную платформу для web-разработки, является пакет **denwer** от Дмитрия Котерова. Его можно скачать и на сайте denwer.ru. Установка пакета весьма проста и обильно документирована на русском языке. Но, установив данный пакет, вы не

сможете самостоятельно разобраться со всеми тонкостями установки, настройки и функционирования сервера, а это совсем не лишние знания для web-программиста на **PHP** в профессиональной жизни. Хотя, в любом случае, выбирать вам. ☺.



The screenshot shows the Denwer website interface. At the top, there's a navigation bar with links: [Денвер](#), [Расширения](#), [Поддержка](#). Below this, the main heading is "Джентльменский Набор Web-Разработчика". A large orange button says "Скачать Денвер 3 5.5 MB". To the right, there's a sidebar with links like "Возможности проекта", "Полный список изменений", "Устанавливаем Денвер", "Пожертвования", and "Ваша реклама на сайте". At the bottom, there's a section titled "Начать работу с Денвером просто".

Помимо denwer'а существует несколько подобных пакетов. Например **UniformServer**:



The screenshot shows the UniformServer website. The top navigation bar has links: [/download](#), [/blog](#), [/wiki](#), [/forum](#), and [/project page](#). The [/download](#) link is circled in red. Below this, there's a section titled "Introduction" with a "Make A Donation" button. The main content area shows a welcome message and a list of links: [Home](#), [Documentation](#), [About Uniform Server](#), [phpinfo\(\)](#), [Tools](#), and [phpMyAdmin](#).

Установка данного пакета заключается в том, что содержимое архива просто нужно распаковать в любую папку (даже на **Flash**-носитель!). После чего запускать и останавливать при помощи соответствующих утилит, которые будут в корневой папке сервера.

РАСПРОСТРАНЕННЫЕ РЕДАКТОРЫ КОДА.

Язык программирования **PHP** является, как уже говорилось, встраиваемым непосредственно в код **HTML**, а значит редактирование кода может осуществляться практически везде, где можно редактировать **HTML**.

Тем не менее, существуют специально разработанные редакторы, ориентированные на работу именно с **PHP**.

1 **PHP Expert Editor**

Удобная в работе интегрированная среда разработки под **Windows** с поддержкой **UTF-8**. Этот редактор понимает **Perl**, **Python**, **JavaScript**, **HTML** (само собой) и др. Имеется встроенный **HTTP**-сервер и отладчик для выполнения, проверки и отладки скриптов.

Скачать можно по адресу: <http://www.phpexperteditor.com>

Реализованы:

- проверка синтаксиса **PHP** и быстрая вставка функций **PHP** с подсказкой параметров, а также справка по **PHP** с поиском по ключевому слову;
- проводники (обозреватели) кода, проектов, файлов и библиотек с избранным;
- внутренний браузер и клиент **FTP** с поддержкой **SFTP**;
- настраиваемые шаблоны кода;
- настраиваемое выделение кода и выделение скобок, свертывание блока кода;

2 **Zend Studio**.

Среда разработки (**Integrated Development Environment**) от разработчиков самого **PHP**. Реализованы:

- Продвинутое форматирование, подсветка синтаксиса, автоподстановка, быстрое комментирование и сворачивание блоков;
- шаблоны кода, поиск и замена;
- проводники файлов, кода, функций и т.д.;
- руководство **PHP**;
- генерация объектного кода;
- поддержка **JavaScript**, **XHTML** на уровне схем **PHP/HTML**, позволяющих перетаскивать **HTML** внутри скрипта;

- интеграция со всеми проектами **Zend Technologies**.
- Отличительной особенностью является возможность удаленной отладки и профайлинга. Для удаленной отладки требуется установить **Zend Studio Server** (в простонародии **Zend Debugger**), который представляет собой серверный модуль.

Скачать можно по адресу:

<http://www.zend.com/en/products/studio/downloads>

3 Dreamweaver

Многоцелевой редактор для web-разработки. Разработан и поддерживался компанией **Macromedia** в версиях до 8-й (2005 год). Следующие версии (начиная с **Dreamweaver CS3** (2007)) выпускает **Adobe**. Богатый инструментарий, открытость приложения для всевозможных настроек, удобный интерфейс и другие особенности сделали **Dreamweaver** одним из наиболее популярных **HTML**-редакторов в мире. «Понимает» также и множество языков web-разработки: **JavaScript**, **CSS**, **PHP** и т.д., что позволяет использовать **Dreamweaver** для web-разработки на стыке с web-дизайном.

ОСНОВЫ СИНТАКСИСА.

ВКЛЮЧЕНИЕ PHP-КОДА ВНУТРЬ СТРАНИЦЫ

Код на языке **PHP** записывается в виде текстовых файлов с расширением `.php`, хотя это, по большому счету, и не обязательно. Расширение в данном случае необходимо лишь как ориентир для сервера **Apache**. Увидев `php`-расширение файла, он передаст этот файл на обработку интерпретатору. См. предыдущую главу, в которой мы устанавливали ПО. Там мы ассоциировали web-сервер и интерпретатор на расширение `.php`.

Содержимое файлов с расширением `.php` может включать как непосредственно код языка **PHP**, так и обыкновенную **XHTML**-разметку, а также всё, что может встраиваться в неё, т.е. определение стилей **CSS**, код скриптов на **JavaScript** и т.д. Интерпретатор при обработке файла аккуратно выполнит директивы «своего» языка – **PHP**, и проигнорирует, оставив «как есть», всё остальное. В результате получится итоговая **XHTML**-страница. Для того, чтобы интерпретатор мог отличить код от постороннего (для него) текста, используются специальные теги, называемые **последовательностями символов перехода в PHP** (**escaping to PHP**). Вариантов таких символов несколько:

- **Полные (стандартные теги).**

```
<?php  
?>
```

Полные теги используются наиболее часто. Во-первых, эти теги доступны по умолчанию без каких-либо дополнительных настроек. Во-вторых, они делают код страницы более наглядным, так как указывают на технологию создания. В-третьих, они соответствуют стандартам W3C.

- **Короткие теги**

```
<?  
?>
```

Удобны своей краткостью записи. Одним из её преимуществ яв-

ляется возможность быстрого вывода значения в тело страницы:

ПРИМЕР 8.1.1

```
<p>My name is <?= $name ?>
```

Конечно это справедливо лишь в том случае, если переменная `$name` была определена ранее в коде. (О переменных и их определении речь пойдёт ниже).

Однако поддержка коротких тегов может быть выключена в конфигурационном файле `php.ini` при помощи директивы `short_open_tag`.

- **ASP-теги**

```
<%
```

```
%>
```

Сегодня используются крайне редко и остались «в наследство» от более старых версий. За их поддержку отвечает директива `asp_tags` в файле `php.ini`.

- **Script-теги**

```
<script language="php">
```

```
</script>
```

Это совсем уж ископаемое, которое вряд ли кто-то будет использовать.

ЗАПИСЬ ВЫРАЖЕНИЙ. КОММЕНТАРИИ.

Итак, напишем первый скрипт на **PHP**. По традиции, это будет приветствие миру:

ПРИМЕР 8.2.1

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

Как видно, в языке **PHP**, так же как и в многих других языках, все выражения разделяются точкой с запятой «;». Как вариант, завершаться выражение может `?>`.

Для того, чтобы глубже разобраться с правилами синтаксиса, рассмотрим более сложный пример:

ПРИМЕР 8.2.2

```
<html><body>
<h1>Здравствуй!</h1>
<?
// Вычисляем текущую дату в формате "день.месяц год"
$dat=date("d.m y");
// Вычисляем текущее время
$tm=date("h:i:s");
// Выводим их
echo "Текущая дата: $dat года<br />\n";
echo "Текущее время: $tm<br />\n";
?>
// Выводим цифры
</body></html>
```

Как видим, в данном примере присутствуют **HTML**-теги. Как это может быть? Дело в том, что результатом выполнения любого **PHP** кода может быть текст, в том числе и гипертекстовая разметка. Таким образом, на выходе, результат **PHP**-сценария свободно смешивается с изначальным текстом (разметкой), как бы встраиваясь в него. Такая совокупность называется **результатирующим HTML-кодом**.

Сам код сценария располагается между открывающим тэгом **<?** и закрывающим **?>**. Если же в программе нужно что-то вывести, то нужно использовать оператор **echo** (это не функция, а конструкция языка: ведь, в конце концов, если это функция, то где же скобки?). Мы подробно рассмотрим ее работу в дальнейшем.

Итак, PHP ориентирован на встраивание кода в HTML-разметку и любой текст, расположенный за пределами <? и ?>, выводится в браузер непосредственно, т. е. воспринимается, как вызов оператора echo.

Последняя аналогия очень важна. В дальнейшем будет ясно почему.

Нетрудно догадаться, что часть строки после **//** является комментарием и на программу никак не влияет.

Это – однострочный комментарий, т.е. он распространяется либо до конца строки, либо до конца **PHP**-блока.

Комментарии еще бывают и многострочными:

```
/*  
это комментарий  
...и еще одна строка  
*/
```

Нужно следить за тем, чтобы не возникало вложенных комментариев:

ПРИМЕР 8.2.3

```
<?php  
/*  
echo "It's works"; /* Этот комментарий был добавлен ранее,  
а затем оказался внутри другого комментария */  
*/  
?>
```

Обратите внимание на предпоследнюю строку примера. Закрывающая метка комментария, выделенная зелёным, не будет восприниматься как комментарий.

ПЕРЕМЕННЫЕ.

Давайте посмотрим, что происходит дальше. Вот строка:

```
$dat=date("d.m y");
```

Делает она следующее: переменной с именем `$dat` присваивается значение, которое вернула функция `date()`. Имена переменных в **PHP** всегда начинаются со знака `$` и состоят только из букв, цифр и знака подчеркивания. Назначая имена переменных, как и прочих идентификаторов, нужно соблюдать следующие правила:

- имя переменной не может начинаться с цифры;
- имя переменной может состоять из кириллических букв как полностью, так и частично (для PHP 6);
- имя переменной чувствительно к регистру!;
- длина имени переменной не ограничивается;
- имя переменной не должно совпадать с ключевыми словами языка.

В **PHP**, во-первых, нет необходимости явно описывать переменные (они инициализируются интерпретатором в момент первого использования), а во-вторых, нигде не указывается их тип (про типы

мы поговорим чуть позже). Интерпретатор сам определяет тип по хранящимся в переменной данным. Это означает, что **PHP** – язык с **динамической типизацией** данных.

А насчет функции `date()` ... Можно заметить, что у нее задается один параметр, который определяет формат результата. Например, в нашем случае это будет строка вида "27.04 09". На следующей строке мы опять видим комментарии, а дальше — еще один оператор, похожий на ранее описанный. Он присваивает переменной `$tm` текущее время в формате "часы:минуты:секунды", опять же при помощи вызова `date()`. Все возможности этой полезной функции будут подробно описаны в одном из следующих уроков.

Далее следуют операторы `echo`, выводящие текстовые строки и нашу дату и время.

Вот какой результирующий код получился в результате работы нашего сценария:

```
<html><body>
<h1>Здравствуйтесь !</h1>
Текущая дата: 27.04 09 года<br />
Текущее время: 10:55:16<br />
</body></html>
```

Как видите, выходные данные сценария комбинировались с текстом, расположенным вне скобок `<? и ?>`. В этом-то и заключена основная сила **PHP**: в легком встраивании кода в тело документа.

Это встраивание 15 лет назад не в последнюю очередь повлияло на то, что **PHP** достиг такой популярности в мире. В наши дни это качество языка практически не используется. Более того, в больших проектах встраивания стараются всячески избегать, стремясь, использованием шаблонизаторов, как можно четче разделить код: логика – отдельно, разметка – отдельно. В будущем мы рассмотрим методы работы с шаблонизаторами, а пока, в учебных целях, встраиванием пренебрегать не станем.

Типы данных

Практически все языки программирования можно разделить на языки со статической типизацией и динамической типизацией. **PHP**, как мы уже выяснили, это язык с динамической типизацией (так же как в прочем и уже знакомый нам **JavaScript** 😊), т.е. тип данных

определяется интерпретатором. Тем не менее, типы данных в языке имеются и вполне четко определены. Их 8:

1 Integer.

Целые числа, которые являются последовательностью из одной или нескольких цифр без дробной части. Числа могут быть положительными или отрицательными, а также записанными в десятичной, восьмеричной (начинаются с 0) или в шестнадцатеричной (начинаются с 0x) формах (регистр букв значения не имеет). Диапазон целых чисел может варьироваться в зависимости от платформы.

2 float (double)

Вещественные числа в отличие от целых чисел имеют дробную часть. Дробная часть всегда отделяется точкой и ни в коем случае запятой!

3 Boolean

Это простейший тип. Он выражает истинность значения – это может быть либо `TRUE`, либо `FALSE`. (Оба слова регистронезависимы). Булев тип был введен в **PHP4**.

4 String

Строка – это набор символов. В **PHP** символ это то же самое, что и байт, это значит, что возможно ровно 256 различных символов. Это справедливо для **PHP** версий 5 и старше, которые не имеют встроенной поддержки **Unicode**. Версия **PHP 6** эту поддержку имеет. Ограничений на размер строк, налагаемых **PHP**, нет.

Строка может быть определена при помощи одинарных кавычек (символ `'`) и при помощи двойных кавычек (`"`). Чтобы использовать одинарную кавычку внутри строки, как и во многих других языках, ее необходимо предварить символом обратной косой черты (`\`), т.е. экранировать ее.

ПРИМЕР 8.4.1

```
<?php
```



```
echo 'просто некая строка';
echo 'Также можно вставлять в строку символ
новой строки вот так';
echo 'Однажды Арнольд сказал: "I\'ll be back"';
// Выведет: Однажды Арнольд сказал: "I'll be back"
echo 'Вы удалили C:\\*..*?';
// Выведет: Вы удалили C:\\*..*?
?>
```

В двойных кавычках **PHP** распознает большее количество управляющих последовательностей для специальных символов:

- `\n` новая строка
- `\r` возврат каретки
- `\t` горизонтальная табуляция
- `\\` обратная косая черта
- `\$` знак доллара
- `\"` двойная кавычка

Но самым важным свойством строк в двойных кавычках является обработка переменных. Если строка определяется в двойных кавычках, переменные внутри нее обрабатываются, а если в одинарных, то не обрабатываются.

Если интерпретатор встречает знак доллара (\$), он захватывает так много символов, сколько возможно, чтобы сформировать правильное имя переменной. Если вы хотите точно определить конец имени, заключайте имя переменной в фигурные скобки.

ПРИМЕР 8.4.2

```
<?php
$beer='Heineken';
echo "$beer's taste is great"; //работает, "'" это неверный
символ для имени переменной
echo "He drank some $beers"; // не работает, 's' это верный
символ для имени переменной
echo "He drank some ${beer}s"; // работает
echo "He drank some {$beer}s"; // работает
?>
```

Другой способ определения строк – это использование **heredoc**-синтаксиса ("<<<"). После <<< необходимо указать

идентификатор, затем идет строка, а потом этот же идентификатор, закрывающий вставку. **Heredoc**-текст ведет себя так же, как и строка в двойных кавычках, при этом их не имея. Это означает, что вам нет необходимости экранировать кавычки в **heredoc**, но вы по-прежнему можете использовать вышеперечисленные управляющие последовательности. Переменные обрабатываются, но с применением сложных переменных внутри **heredoc** нужно быть также внимательным, как и при работе со строками.

ПРИМЕР 8.4.3

```
<?php
$str = <<<EOD
Пример строки,
охватывающей несколько строчек,
с использованием heredoc-синтаксиса.
EOD;
```

Символы в строках можно использовать и модифицировать, определив их смещение, относительно начала строки, начиная с нуля, в фигурных скобках после строки.

ПРИМЕР 8.4.4

```
<?php
// Получение первого символа строки
$str = 'Это тест.';
$first = $str{0};
// Получение третьего символа строки
$third = $str{2};
?>
```

5 array.

Массивы, как и в других языках программирования – это набор нескольких элементов. В PHP массивы бывают нескольких видов – индексируемые (списки), ассоциативные и смешанные.

Подробно рассматривать массивы мы будем в одном из следующих уроков.

6 Object

Концепция объектного типа данных является неотъемлемой частью **ООП** (объектно-ориентированного программирования). Этот раздел подробно будет нами рассматриваться в течении даже нескольких уроков. ☺

7 Resource.

Ресурс – это специальная переменная, содержащая ссылку на внешний ресурс. Ресурсы создаются и используются специальными функциями. Например, функциями для работы с базами данных.

8 null

Специальное значение `NULL` говорит о том, что эта переменная не имеет значения. `NULL` – это единственно возможное значение типа `NULL`. Переменная считается `NULL` если ей была присвоена константа `NULL` или еще не было присвоено никакого значения.

ПРЕОБРАЗОВАНИЕ ТИПОВ ДАННЫХ.

Поскольку **PHP** – это язык с динамической типизацией, то в программе, как правило, нигде нет явного указания на то, какого типа данные хранятся в той или иной переменной. Часто это и не требуется, а вот когда возникает необходимость использовать переменную в определенном контексте, не зная заранее какого типа данные в ней будут, то могут возникнуть разного рода эффекты, которые могут сослужить добрую службу, а могут и навредить, став причиной логических ошибок в скриптах. Таким образом, за преобразованием типов необходимо внимательно следить.

Для проверки типа переменной есть функция `gettype()`, возвращающая тип, который **PHP** назначил переменной:

ПРИМЕР 8.5.1

```
<?php
$var = "5";
$var1 = 5;
echo(gettype($var)); //string
echo "<br>";
echo(gettype($var1)); //integer
?>
```

Кроме этого, существуют еще несколько стандартных функций, которые занимаются определением типа переменных и часто включаются в условные операторы. Вот они.

- `is_integer($a)` – возвращает `true`, если `$a` – целое число.
- `is_double($a)` – возвращает `true`, если `$a` – действительное число.
- `is_string($a)` – возвращает `true`, если `$a` является строкой.

- `is_array($a)` – возвращает `true`, если `$a` является массивом.
- `is_object($a)` – возвращает `true`, если `$a` объявлена как объект.
- `is_boolean($a)` – возвращает `true`, если `$a` определена как логическая переменная.

Существует несколько правил преобразования:

- Если строка начинается с допустимого числового значения, то данная строка при выполнении над ней числовой операции будет преобразована в целое число, Если же строка не начинается с числа, то она будет преобразована в 0.

ПРИМЕР 8.5.2

```
<?php
    $str = "72Sgh_8";
    $res = 5 + $str;
    echo $res; //выведет 77
    echo $str; //выведет 72Sgh_8, т.к. значение самой пере-
менной не изменится при выполнении данной операции
?>
```

- Строка переводится в число с плавающей точкой только в том случае, если число занимает всю строку. Если же в строке будут встречены какие-то буквенные символы, то строка преобразуется в целое число.

ПРИМЕР 8.5.3

```
<?php
    $str = "2.3";
    $res = 5 + $str;
    echo $res; //выведет 5.8
?>
```

- При преобразовании в логический тип, следующие значения рассматриваются как `FALSE`:
 - ✓ Сам булев `FALSE`;
 - ✓ целое `0` (ноль);
 - ✓ число с плавающей точкой `0.0` (ноль);
 - ✓ пустая строка и строка `"0"`;
 - ✓ массив с нулевыми элементами;
 - ✓ объект с нулевыми переменными-свойствами;
 - ✓ специальный тип `NULL` (включая неустановленные переменные);

Все остальные значения рассматриваются как `TRUE`

Кроме того, существует способ явного преобразования типов. Для этого существуют операторы преобразования типа (**casting**), которые представляют собой нужный тип, записываемый в скобках перед переменной:

- `(int)` или `(integer)`
- `(real)`, `(double)` или `(float)`
- `(bool)` или `(boolean)`
- `(string)`
- `(array)`
- `(object)`

ПРИМЕР 8.5.4

```
<?php
    $a = 99.2;
    $a = (int)$a;           //=99
    $b = (double)$a;       //=99.0
    $c = (string)$a;       //= "99"
?>
```

Существует также функция `settype()`, которая явно устанавливает тип:

ПРИМЕР 8.5.5

```
<?php
    $var = "5";
    echo (gettype($var));
    settype($var, integer);
    echo "<br>";
    echo (gettype($var));
?>
```

Константы

Помимо переменных, в **PHP** существуют и константы – именованные величины, неизменные в процессе выполнения программы. Константы определяются при помощи функции `define()`:

ПРИМЕР 8.6.1

```
define(SALUTATION, "Hello, world!");
```

Различия между константами и переменными:

- У констант нет приставки в виде знака доллара (\$);
- Константы можно определить только с помощью функции `define()`, а не присваиванием значения;
- Константы могут быть определены и доступны в любом месте без учета области видимости;
- Константы не могут быть определены или аннулированы после первоначального объявления;
- Константы могут иметь только скалярные значения.

Существуют несколько predefined констант.

- `__FILE__` - хранит имя файла программы, которая выполняется в данный момент.
- `__LINE__` - содержит текущий номер строки, которую обрабатывает в текущий момент интерпретатор. Эта своеобразная "константа" каждый раз меняется по ходу исполнения программы.
- `PHP_VERSION` - версия интерпретатора PHP.
- `PHP_OS` - имя операционной системы, под которой работает PHP.
- `TRUE` или `true` - эта константа нам уже знакома и содержит значение "истина".
- `FALSE` или `false` - содержит значение "ложь".

В **PHP** существует также функция, которая проверяет, существует ли (была ли определена ранее) константа с указанным именем.

```
bool defined(string $name)
```

Возвращает `true`, если константа с именем `$name` была ранее определена.

ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА

ОПЕРАТОРЫ, ОПЕРАЦИИ И ВЫРАЖЕНИЯ

Если рассматривать выражения в качестве предложений, то подлежащим в нём будет **операнд** – некоторая величина, обрабатываемая в программе, а сказуемым – **оператор** – символическое обозначение некоторого действия, выполняемого с операндом.

ПРИМЕР 9.1.1

```
$sum=$a+$b;
```

Здесь `$sum`, `$a` и `$b` – это операнды, а `=` и `+` – это операторы.

Большинство операторов обрабатывают два операнда, поэтому называются они **бинарными**. Кроме того, как и в **C**, в **PHP** существуют **унарные** операторы и один **тернарный**.

Большинство операторов работают с операндами определенного типа. В случае несоответствия типов, интерпретатор попытается самостоятельно преобразовать типы, согласно тех правил, которые были рассмотрены ранее.

Рассмотрим чаще всего используемые операторы. Это:

- **арифметические операторы:**
 - `a+b` – сложение;
 - `a-b` – вычитание;
 - `a*b` – умножение;
 - `a/b` – деление;
 - `a%b` – остаток от деления `a` на `b`;

Операция деления `"/"` возвращает целое число (то есть, результат деления нацело), если оба выражения `a` и `b` – целого типа (или же строки, выглядящие как целые числа), иначе результат будет вещественным. Операция вычисления остатка от деления `"%"` работает только с целыми числами, так что применять ее к вещественным числам не стоит.

Также существуют, как и в **C**, унарные операторы инкремента и декремента, записываемые в двух формах:

`a++` или `a--` - постинкремент или постдекремент

`++a` или `--a` - прединкремент или преддекремент

Эти операторы являются сокращенной записью операции прибавления (или вычитания) к числу единицы. Они не могут использоваться ни с чем кроме переменной, т.о. выражение `45++` или `+"строка"` неверно.

Если оператор размещен справа, то интерпретатор сначала использует значение переменной, а потом изменит её («пост»-запись). При размещении слева – сначала изменение переменной, затем – использование («пред»-запись).

В качестве разновидности арифметического оператора, но работающего исключительно со строками, следует указать оператор `.` (точка). Это оператор **конкатенации**, т.е. слияния строк.

Например:

ПРИМЕР 9.1.2

```
<?php
$phrase1="Первое слово";
$phrase2="съела";
echo $phrase1.$phrase2."корова"; //Выведет: "Первое слово
съела корова"
?>
```

• операторы присваивания:

Реально существует только один оператор присваивания – `=`, но **PHP** поддерживает сокращенные формы записи: `+=`, `-=`, `/=`, `*=` и т.д. Все они присваивают левому операнду результат указанной операции между левым и правым операндами.

• операторы сравнения и операторы сравнения:

Операции сравнения позволяют сравнивать два значения между собой и, если условие выполнено, возвращают `true`, а если нет – `false`.

- `a==b` – истина, если `a` равно `b`;
- `a!=b` – истина, если `a` не равно `b`;
- `a<b` – истина, если `a` меньше `b`;
- `a>b` – аналогично больше;
- `a<=b` – истина, если `a` меньше либо равно `b`;

- `a >= b` – аналогично больше либо равно.
- `a & b` – истина, если истинны и `a` и `b`;
- `a || b` – истина, если истинно либо `a`, либо `b`.

Следует отметить, что в **PHP** сравнивать можно только скалярные (то есть строки и числа) переменные. Для массивов и объектов этого делать нельзя. При попытке их сравнения, они оба будут преобразованы в слово `array` или `object`. Оба эти слова, как Вы понимаете, совершенно равны, а, стало быть, какие бы массивы или объекты не сравнивать, результат всегда будет `true`.

Начиная с **PHP 4**, появился оператор эквивалентности – тройной знак равенства `===`. Он сравнивает не только значения выражений, но и их типы. Если сравнить `0` и `""` (пустую строку) при помощи `===`, то результат будет `true`, поскольку, если один из операндов логического оператора может трактоваться как число, то оба операнда трактуются как числа. При этом пустая строка превращается в `0`, который затем и сравнивается с нулем. Применение оператора эквивалентности решает эту проблему. Существует также и обратный оператор `!==` – не эквивалентно.

Помимо рассмотренных операторов, существуют также побитовые операторы, которые применяются не так уж часто и их описание вы сможете найти в [официальной документации](#).

Как и в других языках программирования, порядок выполнения операторов в **PHP** подчиняется определенным приоритетам.

Порядок выполнения операторов

Порядок	Оператор
слева направо	<code>++ --</code>
слева направо	<code>! - (int) (float) (string) (array) (object)</code>
слева направо	<code>* / %</code>
слева направо	<code>+ - .</code>
слева направо	<code>< <= > >=</code>
слева направо	<code>== != === !==</code>

Порядок	Оператор
слева направо	&&
слева направо	
слева направо	? :
справа налево	= += -= *= /= .= %=

Конечно, даже простые программы могут быть многовариантными, и здесь не обойтись без условных конструкций и циклов, которые объединяют выражения в еще большие конструкции – блоки.

УСЛОВНЫЕ КОНСТРУКЦИИ.

Обычно выражения в программе выполняются в том порядке, в котором они написаны в коде. Чтобы изменить этот порядок, необходимо использовать **условные конструкции**. Условных конструкций в **PHP** две: `if-else` и `switch`.

- **if-else**

существует несколько форм записи конструкции `if`. В самой простой форме происходит проверка какого-либо условия и, в зависимости от его истинности, выполнение блока, заключенного в фигурные скобки:

```
if (условие) {
    код;
}
```

В случае, если условие в скобках ложно, то блок в фигурных скобках будет просто пропущен.

ПРИМЕР 9.2.1

```
<?php
$a=10;
if ($a>5) {
    echo $a." больше пяти"; }
?> //Выведет 10 больше пяти
```

Эта форма записи может быть расширена при помощи команды

else:

```
if (условие) {  
    код;  
}  
else {  
    альтернативный код;  
}
```

Если условие в `if` будет истинным, то выполнится первый блок кода, в противном случае – второй блок кода.

В ветвления можно добавить неограниченное количество дополнительных проверок при помощи команды `elseif`:

```
if (условие) {  
    код;  
}  
elseif (альтернативное_условие) {  
    альтернативный код #1;  
}  
else {  
    альтернативный код #2;  
}
```

Количество блоков `elseif` может быть любым.

Конструкция `if-else` имеет еще один альтернативный синтаксис, хоть и применяется он значительно реже:

```
if (условие) :  
    код;  
elseif(альтернативное_условие) :  
    альтернативный код #1;  
else:  
    альтернативный код #2;  
endif;
```

Обратите внимание на расположение двоеточия ":"! Если его пропустить, будет сгенерировано сообщение об ошибке. Как обыч-

но, блоки `elseif` и `else` можно опускать.

- **switch**

Несколько последовательных конструкций `if-else`, для удобства, могут быть заменены конструкцией `switch-case`:

```
switch (выражение) {  
    case значение1: блок #1; [break;]  
    case значение2: блок #2; [break;]  
    case значение3: блок #3; [break;]  
    [default: блок_по_умолчанию; [break]]  
}
```

Делает она следующее: вычисляет значение выражения, а затем пытается найти строку `case` для соответствующего значения. Если такая строка обнаружена, выполняются команды, расположенные сразу после нее (причем на все последующие операторы `case` внимание не обращается, как будто их нет, а код после них остается без изменения). Если же найти такую строку не удалось, выполняются команды после `default`, если они заданы. Если же и `default` нет, то тогда блок пропускается весь и выполняется тот код в программе, который идет дальше.

ПРИМЕР 9.2.2

```
<?php  
switch($beer)  
{  
case 'tuborg';  
case 'carlsberg';  
case 'heineken': echo 'Good choice'; break;  
default: echo 'Please make a new selection...'; break;  
}  
?>
```

Обратите внимание на операторы `break` (которые условно заключены в квадратные скобки, чтобы подчеркнуть их необязательность), добавленные после каждой строки команд. Если бы не они, то при равенстве выражения значению №1 сработали бы не только команды из блок #1, но и все нижележащие.

Вот альтернативный синтаксис для конструкции `switch-case`:


```
switch (выражение):  
case значение1: блок #1; [break;]  
case значение2: блок #2; [break;]  
[default: блок_по_умолчанию; [break]]  
endswitch;
```

Конструкцию `switch`, при необходимости, всегда можно заменить структурой блоков `if-else`.

ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ

Циклы, это еще один способ нарушить стандартный порядок выполнения действий, а именно организовать их многократное повторение. Каждое повторение кода внутри цикла называется **итерацией**. Они будут осуществляться до тех пор, пока не будет выполнено условие выхода из цикла. **PHP** поддерживает три типа циклов.

- **Цикл с предусловием (while)**

Применяется в основном в тех случаях, когда количество итераций заранее не известно. Выполняться они будут пока истинно начальное условие:

```
while (условие) {  
код;  
}
```

или альтернативный синтаксис (применяется редко, но все равно встречается):

```
while (условие):  
код;  
endwhile;
```

Пример печати дней недели:

ПРИМЕР 9.3.1

```
<?php  
$currentDate=time(); //Текущая дата в секундах  
echo "До пятницы осталось:\n<ol>\n";  
while (date("l",$currentDate)!="Friday")  
{  
    echo "<li>". date("l",$currentDate). "</li>\n";  
}
```

```
$currentDate+=60*60*24; //«Перемотать» дату на сутки  
вперед  
}  
echo "</ol>\n";  
?>
```

- **Цикл с постусловием (do..while)**

Как ясно из названия такого цикла, отличается он, в принципе, только тем, что условие продолжения цикла проверяется не в начале итерации, а в конце. Это приводит к тому, что цикл с предусловием выполнится 0 и более раз, а цикл с постусловием – 1 и более раз.

```
do {  
}  
while (условие);
```

- **Цикл for**

От предыдущих конструкций этот цикл отличается тем, что условие выполнения цикла изменяется не в итерации, а в самой конструкции, т.е. мы имеем возможность запрограммировать цикл на вполне определенное количество итераций.

```
for (инициализация; условие; приращение) {  
код;  
}
```

Работает он следующим образом. Как только управление доходит до цикла, первым делом выполняются операторы, включенные в блок *инициализация* (слева направо). Эти команды перечисляются через запятую, например:

ПРИМЕР 9.3.2

```
for ($i=0, $j=10, $k="Test";)
```

Затем начинается итерация. Первым делом проверяется, выполняется ли *условие* (как в конструкции *while*). Если да, то все в порядке, и цикл продолжается. Иначе осуществляется выход из конструкции. Например:

ПРИМЕР 9.3.3

```
// прибавляем по одной точке  
for($i=0,$j=10,$k="Test"; $i<10;) $k.=".";
```

Предположим, что тело цикла проработало одну итерацию. После этого вступают в действие **приращение** (их формат тот же, что и у инициализации). Например:

ПРИМЕР 9.3.4

```
for($i=0,$j=10,$k="Points";$i<100;$j++, $i+=$j) $k=$k.".";
```

Хочется добавить, что приведенный пример (да и вообще любой цикл **for**) можно реализовать и через **while**, только это будет выглядеть не так изящно и лаконично. Например:

ПРИМЕР 9.3.5

```
$i=0; $j=0; $k="Points";  
while ($i<100) {  
    $k.=".";  
    $j++; $i+=$j;  
}
```

Как обычно, имеется и альтернативный синтаксис конструкции:

```
for (инициализация; условие; приращение):  
    код;  
endfor;
```

При использовании циклов может возникнуть ситуация, когда, вследствие логической ошибки, условие всегда истинно. В результате получаем так называемый вечный цикл. Интерпретатор **PHP** останавливает любой сценарий, если он занимает более 30 секунд процессорного времени. Этот таймаут можно изменить при помощи директивы **max_execution_time** конфигурационного файла **php.ini**.

В случае, если возникает необходимость нарушить выполнение цикла, то для этого существуют операторы **break** и **continue**. Команда **break** прерывает выполнение всего цикла, а **continue** прерывает текущую итерацию и переходит к следующей.

ДОМАШНЕЕ ЗАДАНИЕ.

Домашним заданием будет самостоятельная установка и настройка web-сервера на своей домашней машине.

ИТОГИ.

В этом уроке мы разобрались с сутью web-программирования, его задачами, а также выяснили, какие технологии сегодня актуальны в мире web-программирования.

Кроме того, решили очень важную задачу по установке и настройке программного обеспечения, которое будет нам необходимо в дальнейшем для работы, а также разобрали синтаксис языка и его основные конструкции, убедившись в том, что они во многом напоминают то, что уже знакомо из предыдущих курсов.

Далее мы рассмотрим базовые возможности языка.

До новой встречи!

