top
КОМПЬЮТЕРНАЯ АКАДЕМИЯ

Создание web–приложений, исполняемых на стороне сервера при помощи языка программирования PHP и технологии AJAX

Урок №1

Содержание

Структура веб-приложения

Мы приступаем к изучению языка PHP, предназначенного для создания веб-приложений. Какое место занимает PHP на рынке веб-разработки? По данным исследования, проводимого компанией W3Techs, на октябрь 2016 года PHP используется более чем 82% всех сайтов. Результаты этого исследования обновляются и приводятся по адресу:

https://w3techs.com/technologies/overview/programming_language/all

Кроме того, на PHP написаны такие сайты, как Facebook, Wikipedia, Twitter, Vk и многие другие. Думаю, это достаточная мотивация для того, чтобы изучить этот язык программирования.

Вы должны сразу запомнить, что любое веб-приложение является клиент-серверным. Это значит, что оно состоит из двух частей – серверной и клиентской. Серверная часть, как правило, располагается на удаленном компьютере и управляется специальной программой, веб-сервером. Самый распространенный веб-сервер – Apache. С ним мы и будем работать. PHP является серверным языком. Значит, PHP код располагается и выполняется в серверной части веб-приложения.

Клиентская часть веб-приложения – это браузер, установленный на компьютере клиента. Работа веб-приложения сводится к диалогу между клиентским браузером и веб-сервером.

Давайте введем несколько важных терминов. В чем заключается работа с сайтом или веб-приложением? Обычно пользователь хочет получить от удаленного ресурса (сайта или веб-приложения) какую-либо информацию. Для этого он выполняет в браузере определенные действия, а браузер создает и отправляет на сервер запрос или Request. Это и есть первый термин, который надо запомнить. Request – это просьба браузера к серверу прислать некоторые данные. По своей структуре Request – это набор строк в текстовом формате. Формат запроса определен в специальном протоколе под названием HTTP (Hyper Text Transfer Protocol), о котором мы поговорим позже.

У пользователя есть три способа создать запрос (Request) к серверу.

1. Ввести адрес требуемого ресурса в адресную строку браузера и нажать Enter.
2. Кликнуть по какой-либо ссылке на странице.
3. Нажать кнопку типа submit в какой-либо форме.

Например, при вводе в адресную строку браузера google.com.ua, браузер сформирует запрос такого вида:

```
GET / HTTP/1.1
Host: www.google.com.ua
User-Agent: Mozilla/5.0(Windows; U; Windows NT 5.1; ru; rv:1.9.2)
Gecko/20100115 Firefox/3.6 GTB7.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ru,en-us;q=0.7,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: windows-1251,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
```

```
Cookie: PREF=ID=2578ccacb1ecf2aa:U=3a0a80ae418769c4:TM=12645
79766:LM=1284111039:DV=sTIUytipGDoC:S=N0XC7wB0v7c6CCnH;
```

А если пользователь заполнит форму входа на какой-нибудь странице, браузер сформирует Request, подобный такому:

```
POST http://www.site.ua/news.html HTTP/1.0
Host: www.site.ua
User-Agent: Mozilla/3.0
Accept: text/html
Accept: image/gif
Referer: http://www.site.ua/index.html
Cookie: income=1
Content-Type: application/x-www-form-urlencoded
Content-Length: 31
\r\n
login=John%20Smith&password=123
```

Первая строка в запросе содержит имя команды, которую надо выполнить серверу, URL затребованного клиентом ресурса и версию HTTP. Кроме стартовой строки, запрос содержит еще набор строк, расшифровывающих его содержание. Такие строки называются заголовками и имеют формат Header:Value, где Header – это предопределенные в протоколе HTTP значения, например: Host, User-Agent, Connection, If-Modified-Since и др. HTTP также отвечает за доставку запроса от браузера к серверу.

Теперь посмотрим, что происходит, когда запрос добирается до веб-сервера. Получив запрос, сервер готовит и отправляет обратно клиенту специальный текстовый объект, называемый ответом (Response). Затем

сервер сразу же разрывает соединение с клиентом. Ответ, как и запрос, состоит из набора текстовых строк. Первая строка содержит версию протокола HTTP и код возврата. Затем, как и в запросе, идет группа строк заголовков. А уже после пустой строки, в теле ответа, пересылается контент, запрошенный клиентом. Вот пример типичного Response:

```
HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Sat, 8 Oct 2016 04:36:25 GMT
Server: LiteSpeed
Connection: close
X-Powered-By: W3 Total Cache/0.8
Pragma: public
Expires: Sat, 8 Oct 2016 05:36:25 GMT
Etag: “pub1259380237;gz”
Cache-Control: max-age=3600, public
Content-Type: text/html; charset=UTF-8
Last-Modified: Sat, 8 Oct 2016 03:50:37 GMT
X-Pingback: http://net.tutsplus.com/xmlrpc.php
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie, User-Agent
\r\n
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Strict//EN”
        “http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>
<head>
<meta http-equiv=”Content-Type”content=”text/html;charset=utf-8”/>
<title>Top 20+ MySQL Best Practices – Nettuts+</title>
<!-- ... rest of the html ... -->
```

Не пугайтесь обилия строк-заголовков в Request и Response. Вам не надо учить их напамять. Хотя с неко-

торыми их них мы познакомимся поближе в ходе наших занятий. Данные фрагменты просто показывают вам внешний вид Request и Response. Схематично взаимодействие браузера и веб-сервера выглядит так:

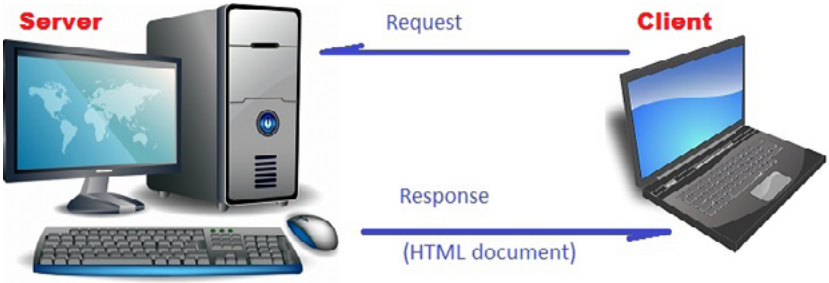

Рис. 1. Взаимодействие браузера и веб-сервера

Сейчас вы должны запомнить, что работа веб-приложения сводится к взаимодействию двух программ: браузера и веб-сервера. Веб-сервер получает Request от браузеров клиентов, обрабатывает их и отправляет обратно клиентам Response. Получив Response, браузер отображает у себя в клиентской области присланный сервером контент, расположенный после пустой строки в теле Response, и пользователь видит полученную страницу. Кроме того, запомните, что именно в протоколе HTTP описаны структуры Request и Response.

Для работы с PHP нам понадобится какая-либо программа веб-сервер, которая будет обрабатывать наши запросы. Для учебных целей наш веб-сервер будет локальным, т.е. будет располагаться на том же компьютере, с которого мы будем создавать Request.

Мы уже отметили выше, что самым распространенным веб-сервером является Apache. Поэтому будем ис-

пользовать Apache и текстовый редактор Sublime Text. Текстовый редактор можно использовать и другой, но Sublime Text является хорошим выбором с точки зрения простоты использования, небольшого размера и мощного набора функций. Уроки будут проводиться в предположении, что вы пользуетесь редактором Sublime Text. Что касается веб-сервера, то этот вопрос мы закроем в двух последующих разделах.

Установка и настройка Apache

В предыдущем разделе мы отметили, что самым распространенным веб-сервером является Apache. Существует мнение, что установка и настройка Apache – очень сложный процесс. Сейчас мы с вами выполним установку и настройку Apache, и вы сможете со знанием дела составить собственное мнение об этом процессе, сначала сделаем одну оговорку. Для разработки сайтов не обязательно использовать полноценную установку этого веб-сервера. Существует целый ряд альтернатив, и в ходе наших занятий мы будем использовать одну из них. Не смотря на это, мы все равно выполним установку и настройку Apache – в познавательных целях.

Веб-сервер Apache распространяется совершенно бесплатно, поэтому смело идем на страницу разработчиков и выбираем для скачивания подходящий установщик. Адрес указанной страницы:

http://archive.apache.org/dist/httpd/binaries/win32/.

Я работаю под управлением Windows, поэтому и установщик выбираю именно для Windows. Чуть позже вы узнаете, в чем отличия между версиями 2.2 и 2.4, поэтому сейчас я просто продемонстрирую установку веб-сервера Apache версии 2.2.25.

Находим на указанной странице ссылку на файл с именем httpd-2.2.25-win32-x86-no_ssl.msi и скачиваем его. Установщик имеет размер чуть более 5 МБ. После завершения скачивания запускаем этот файл от име-

ни администратора. У вас откроется мастер установки Apache:

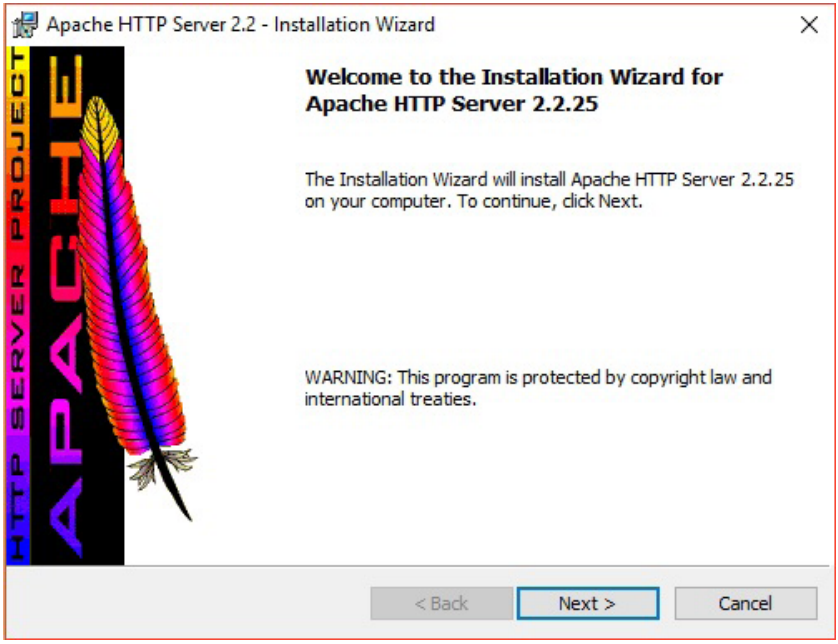

Рис. 2. Начало установки Apache

Особенно вмешиваться в процесс установки не надо, поэтому жмем кнопку Next. Соглашаемся с правилами пользования программой и идем дальше.

Занести некоторые данные надо будет, когда вы увидите окно мастера установки с заголовком Server Information. Для наших целей в этом окне надо будет указать в качестве Network Domain и Server Name значение localhost. Это говорит о том, что мы планируем использовать данную установку Apache как локальный сервер. В третьем поле Administrator's Email Address занесите какой-либо адрес. На этот адрес будут прихо-

дить сообщения при возникновении проблем с работой веб-сервера.

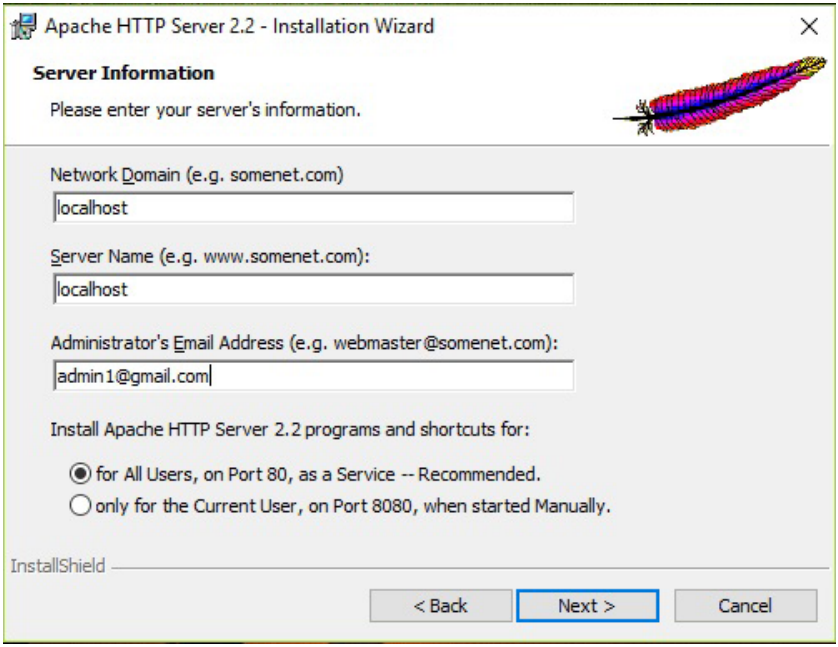

Рис. 3. Указание имен домена и сервера

Снова жмем кнопку Next до самого завершения установки. На одном из этапов установки мастер предложит указать папку установки. Здесь желательно оставить путь по умолчанию. У меня это C:\Program Files (x86)\Apache Software Foundation\Apache2.2\.

Веб-сервер Apache по умолчанию устанавливается как служба. Давайте проверим, появилась ли на нашем компьютере служба Apache.

Для этого нажимаем комбинацию клавиш Win+R и заносим в появившееся окно строку services.msc – это имя оснастки служб. Чтобы выполнить эту команду, вы

должны быть администратором на своем компьютере. Права администратора нам будут нужны еще не раз, поэтому вы должны позаботиться о том, чтобы получить их. При успешном выполнении указанной команды, у вас должно открыться окно, подобное приведенному на рисунке:

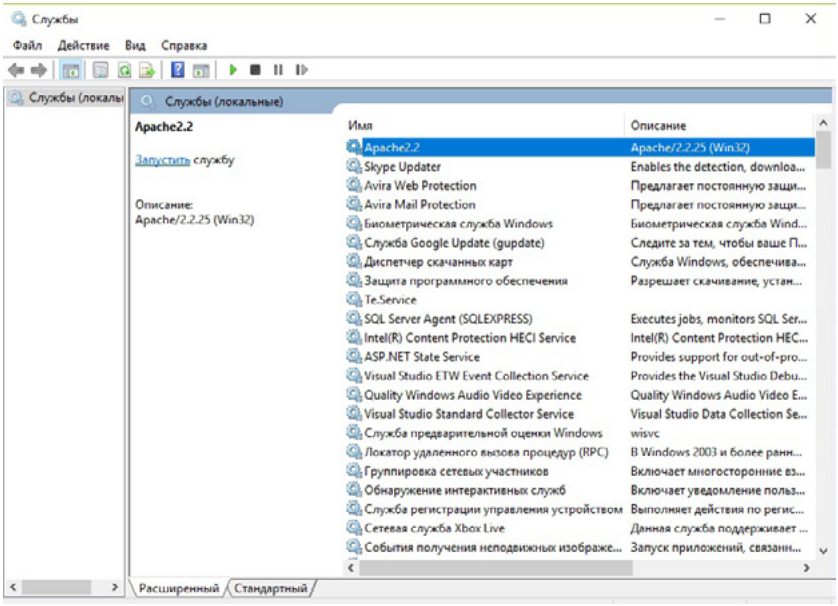

Рис. 4. Оснастка для управления службами

Выделенная служба с именем Apache2.2 и есть установленный нами веб-сервер. Как видите, в установке ничего сложного нет. Но одной установки недостаточно для работы. Теперь надо выполнить настройку Apache.

Рассмотрим процесс настройки. Надо признать, что настройка Apache выглядит старомодно и неуклюже. Она сводится к редактированию достаточно больших текстовых файлов, в которых разбросаны опции на-

стройки. Запомните, что символ хеш «#» в этих файлах является символом комментария. Если перед какой-либо строкой настроек стоит «#», значит эта строка закомментирована. Поэтому чаще всего надо будет убирать или добавлять этот символ в разных строках.

Файл конфигурации располагается по пути «C:\Program Files (x86)\Apache Software Foundation\Apache2\conf\httpd.conf». Откройте в текстовом блокноте этот файл (httpd.conf). Обратите внимание, в ходе редактирования этого файла мы будем ссылаться на некоторые папки, которых у нас пока еще нет. Это такие папки как c:\php и c:\apache. Мы создадим их после изменения файла настроек.

1. Найдите строку, начинающуюся с ServerRoot (у меня это строка 35), и приведите ее к виду (если вы установили Apache в другую папку, то укажите свой путь): ServerRoot "C:/Program Files (x86)/Apache Software Foundation/Apache2.2".

   Директива ServerRoot указывает на каталог, в котором установлен веб-сервер Apache.

2. Найдите строку
   LoadModule rewrite_module modules/mod_rewrite.so
   и раскомментируйте ее (в моем файле это строка 153).

   Данный модуль предназначен для синтаксического анализа и динамического преобразования URL.

3. После всех строк, начинающихся с LoadModule (или #LoadModule), добавьте такую строку:

   LoadModule php5_module «C:/php/php5apache2_2.dll» (в моем файле эта строка стала 176).

4. В строке PHPIniDir (у меня это строка 219) добавьте путь, чтобы получилась такая строка:

   PHPIniDir «C:/php».

   Директива PHPIniDir указывает на каталог, в котором установлен PHP.

5. Найдите строку #ServerName www.example.com:80 (у меня это строка 218) и замените ее на такую строку: ServerName localhost:80 (обратите внимание на удаленный символ комментария!).

6. Найдите строку DocumentRoot «c:/Apache2/htdocs» (у меня это строка 245) и замените ее на такую строку: DocumentRoot «C:/apache»

   Директива DocumentRoot указывает на каталог, в котором будут размещаться сайты.

7. Найдите начало блока <Directory /> (у меня это строка 227) и замените все содержимое блока на такое:

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

8. Найдите начало блока <IfModule dir_module> (у меня это строка 281) и замените все его содержимое на такое:

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

9. Найдите строку, начинающуюся с ErrorLog (у меня это строка 300), и замените ее на такую строку: ErrorLog «C:/apache/error.log»
10. Найдите строку, начинающуюся с CustomLog (у меня это строка 331), и замените ее на такую строку: CustomLog «C:/apache/access.log» common
11. Найдите начало модуля <IfModule mime_module> (у меня это строка 390) и раскомментируйте в нем такие строки:

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

12. Добавьте в этот же модуль <IfModule mime_module> еще такие строки:

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

13. Найдите и раскомментируйте еще такие строки (у меня они начинаются с номера строки 488):

```
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
```

Сохраните этот файл и закройте его.
Ура! Закончили! Но только с одним файлом☺

Теперь откройте в блокноте файл «C:\Apache2\conf\extra\httpd-vhosts.conf». В нем тоже надо произвести замену. Если вы планируете размещать на одном ком-

пьютере более одного сайта, вам надо создать так называемые виртуальные хосты, которые описываеются в указанном файле httpd-vhosts.conf.

Удалите приведенные там примеры блоков виртуальных хостов и вместо них занесите:

```
NameVirtualHost *:80

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

Мы создали виртуальный хост с именем localhost и указали его корневую папку. Сохраните этот файл и закройте его. Теперь настройка завершена. Почти.

Собственно, веб-сервер Apache уже установлен и настроен. Нам осталось добавить к нему PHP. Зайдите на страницу загрузки PHP для Windows:

http://windows.php.net/downloads/releases/archives/

и загрузите файл с именем

php-5.3.5-src.zip

Вам сразу бросится в глаза обилие доступных версий PHP и тот факт, что мы выбираем далеко не новую версию.

Не удивляйтесь, что мы устанавливаем старую версию PHP. На это есть причина. Мы начнем знакомство с PHP в его классическом виде. Для этого нам и нужна выбранная версия. Но уже очень скоро мы перейдем к более новым версиям. Таким образом, вы будете знать об основных отличиях в этих версиях и будете пони-

мать в какой конфигурации приложения, какую версию PHP надо использовать.

Вообще говоря, устанавливать PHP не надо, его достаточно просто скопировать в требуемую папку. Поэтому распакуйте скопированный архив, переименуйте папку с php-5.3.5 на php и скопируйте эту папку на диск c:. Помните, мы в настройках указывали папку c:\php? Теперь эта папка у нас есть.

Затем на диске c: создайте такие вложенные папки c:\apache\localhost\www. Именно в папке www и надо создавать все свои сайты. После выполненных настроек надо перезапустить Apache. Вместе с установкой Apache на компьютер устанавливается специальная программа Apache Monitor. Ее можно найти в системном трее. С помощью этой программы можно останавливать, запускать и перезапускать веб-сервер Apache.

Проверим установленный Apache в деле. Создадим тестовый файл с именем index.php по пути c:\apache\localhost\www. Забегая вперед, занесем в этот файл такой текст:

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

Это вызов функции, которая выводит в браузер описание конфигурации установленного PHP. Этот файл лежит в папке установленного нами веб-сервера Apache. Давайте обратимся к веб-серверу с просьбой прислать нам результат обработки этого файла. Для этого надо запустить браузер и ввести в адресную строку такой адрес:

http://localhost/index.php

Мы уже отметили, что не будем работать с такой установкой Apache, а будем использовать некую альтернативу. Что это за альтернатива? Существует целый ряд так называемых сборок, которые включают в себя Apache (возможно, немного ограниченный в функционале), СУБД MySQL и ряд вспомогательных программных продуктов. Например, веб–клиент для работы с СУБД MySQL – программу PhpMyAdmin. Самыми популярными из сборок являются Denwer, XAMPP, OpenServer. Мы с вами будем работать с OpenServer. В следующем разделе рассмотрим установку этого продукта.

Поэтому установленные Apache и PHP нам не понадобятся, и их надо удалить.

Установка и настройка OpenServer

OpenServer – интегрированный пакет для веб-разработчика, который по умолчанию включает в себя веб-сервер Apache, СУБД MySQL, клиент PhpMyAdmin для работы с БД и ряд других полезных инструментов, которые нам понадобятся немного позже. Одним из преимуществ этого пакета является то, что он легко позволяет «налету» изменять версии используемых Apache и PHP. Кроме того, этот пакет допускает установку даже на флешку. В этом случае вы получаете portable рабочее место веб-разработчика.

Другими словами, если у вас будет флешка с установленным OpenServer, вы сможете работать над своими веб-приложениями на любом компьютере, куда вставите эту флешку. Вам не надо будет копировать проекты после занятий в аудитории, чтобы переслать их себе домой, для продолжения работы. При этом все же не забывайте делать архивные копии своих данных где-нибудь еще!

Бесплатно скачать установщик OpenServer можно, например, на этой странице:

http://open-server.soft112.com/

Запустите загруженный файл и распакуйте его на диск в какую-либо папку. После распаковки в указанной вами папке будет создана папка OpenServer с вложенными паками и двумя .exe файлами, как на рисунке:

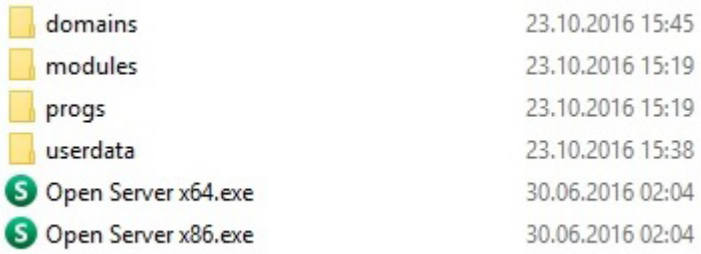

Рис. 5. Папка с установленным OpenServer

В зависимости от разрядности вашей операционной системы запустите либо Open Server x64.exe, либо Open Server x86.exe. В системном трее появится иконка в виде флага – это иконка OpenServer. Цвет флага говорит о статусе программы: зеленый – работает, красный – остановлена, желтый – в переходном состоянии. Выделите эту иконку и нажмите правую кнопку мыши. Вы увидите главное меню OpenServer.

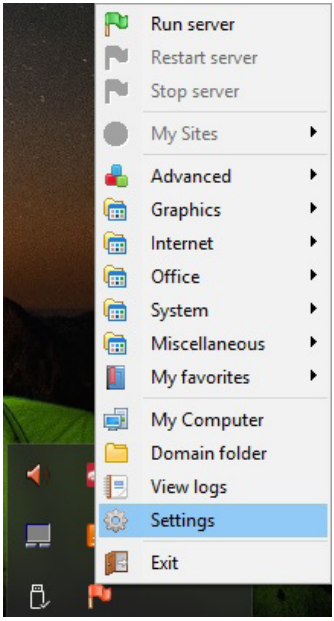

Рис. 6. Главное меню OpenServer

Активируйте пункт меню Settings. Раскроется окно настроек, в котором надо выбрать вкладку Modules. Эта вкладка показывает и позволяет изменять версии текущих модулей: самого PHP, сервера MySQL и других.

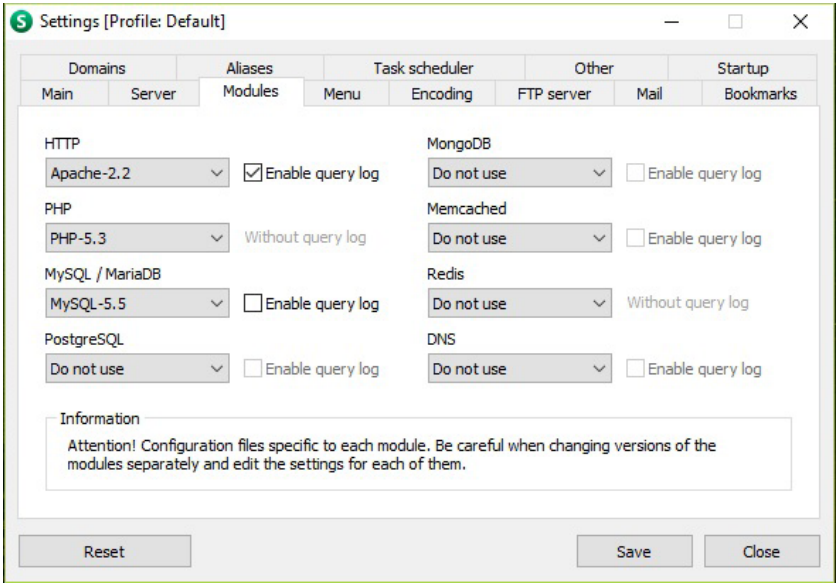

Рис. 7. Вкладка Modules

Когда у нас возникнет необходимость изменить версию PHP и Apache, мы воспользуемся этой вкладкой.

Очень полезной является вкладка Server, на которой отображены используемые программой порты, адреса и другие настройки. Вы должны помнить из курса сетевого программирования такое понятие, как сокет. Это атрибут, позволяющий приложению передавать свои данные в локальную сеть и получать присланные ему данные из локальной сети. По сути, сокет представляет собой комбинацию сетевого адреса компьютера и «но-

мера» конкретного приложения на этом компьютере, которому надо выходить в сеть. Такой номер приложения называется портом и представляет собой целое число в диапазоне 0-65535. Веб-приложение тоже является сетевым, но функционирует не в локальной сети, а в глобальной. Поэтому для выхода в сеть приложение должно иметь собственный номер порта. В вебе для этих целей зарезервирован порт 80 для браузера. У других компонент приложения – свои номера портов.

Если на вашем компьютере какой-либо порт занят и OpenServer не запускается, перейдите на эту вкладку, измените конфликтый номер порта, нажмите кнопки Save, а затем Close, и перезапустите OpenServer. Чаще всего, этих действий достаточно, чтобы решить проблему с запуском.

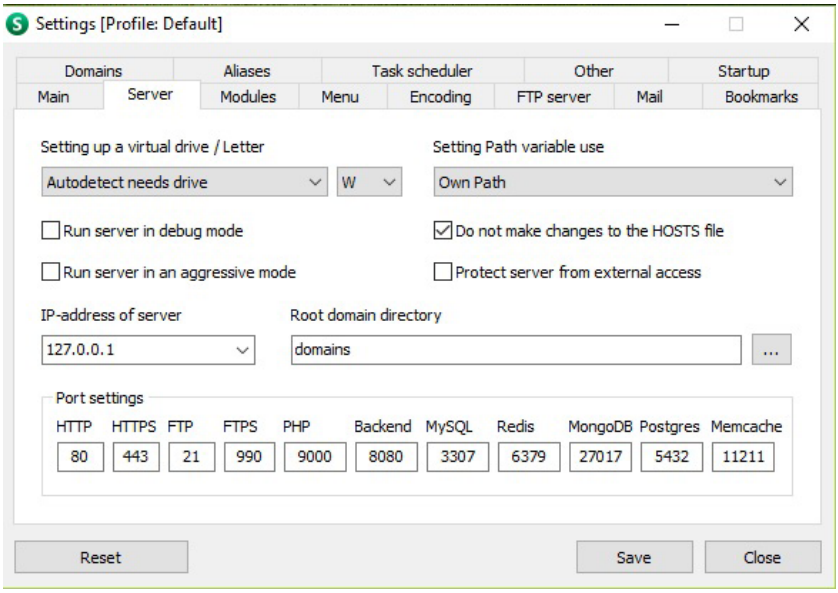

Рис. 8. Вкладка Server

Теперь откройте вкладку Domains и в поле «Auto root domain (in space)» выберите из выпадающего списка вторую опцию, как указано на рисунке:

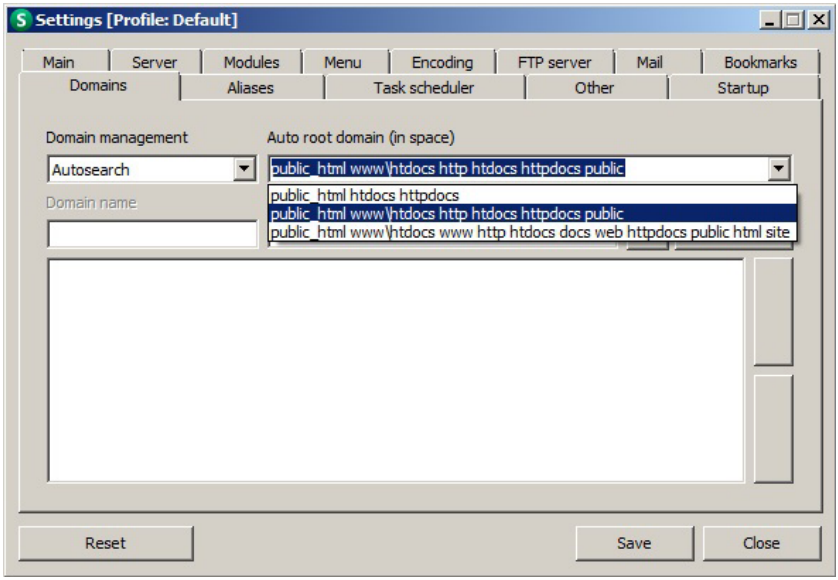

Рис. 9. Вкладка Domains

Запустите OpenServer и оставьте его в активном состоянии. С другими пунктами меню OpenServer мы будем знакомиться в ходе дальнейших занятий.

Иногда бывает так, что вы не являетесь локальным администратором на компьютере, где установлен OpenServer. В этом случае у вас нет доступа для изменения системного файла C:\Windows\System32\drivers\etc\hosts. Программе OpenServer требуется полный доступ к этому файлу. О том, есть у вас такой доступ или нет, вы узнаете при запуске OpenServer. Если полного доступа нет, OpenServer не запустится, сообщит о невозмож-

ности изменить файл hosts и попросит вас установить опцию, запрещающую вносить изменения в файл hosts.

Лучшим решением будет попытаться получить права доступа локального администратора. Если же это невозможно, выставьте на вкладке Settings флажок в поле «Do not make changes to the HOSTS file», как показано на рисунке 8. При этом работа с OpenServer будет нормальной и полноценной, но будет немного отличаться от ситуации, когда у вас есть полный доступ к файлу hosts.

Основы PHP

Давайте договоримся: если у вас нет проблем с доступом к файлу hosts, то все ваши файлы вы будете создавать в папке \OpenServer\domains рядом с папкой localhost. Проблемы с доступом могут быть в том случае, если вы не являетесь администратором на компьютере, где установлен OpenServer. Если же вы не можете изменять файл hosts, то выставьте опцию «Do not make changes to the HOSTS file», и создавайте свои файлы в папке \OpenServer\domains\localhost.

Создадим в требуемой папке (либо domains, либо localhost) папку Test, а в ней – файл test1.php. Откроем этот файл в Sublime Text и напишем в нем наш первый код:

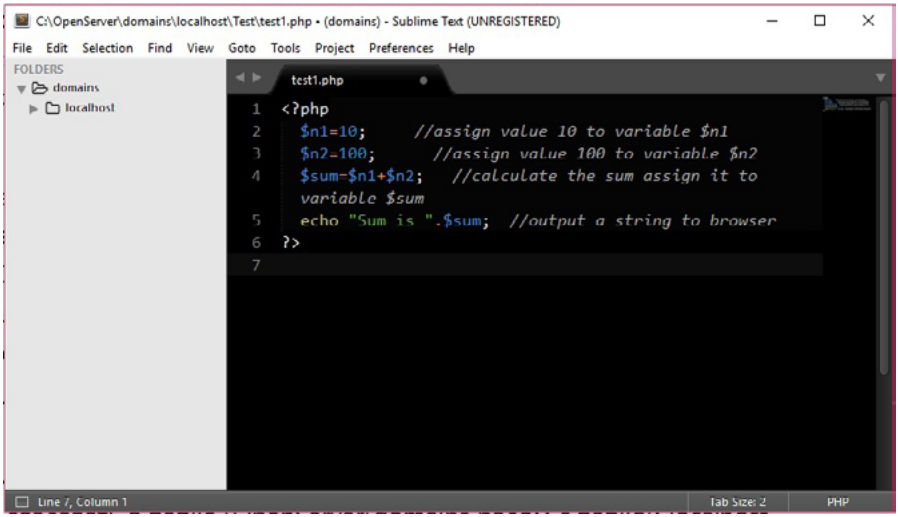

Рис. 10. Рабочее окно Sublime Text

PHP код должен заключаться в специальные теги. Стандартным вариантом считается использование тегов <?php и ?>. Однако, если в файле php.ini параметру short_open_tag присвоено значение 1, то можно использовать «короткие» теги <? и ?>. Относительно открывающих и закрывающих тегов запомните еще такой момент. Если у вас на странице расположен только PHP код (без HTML областей), то закрывающий тег желательно не ставить. Дело в том, что иногда за закрывающим тегом могут случайно добавляться невизуализируемые символы, типа пробелов или новых строк, которые в таком случае могут привести к нежелательному выводу на страницу.

Мы будем использовать теги <?php и ?>. Поэтому занесем в наш файл такое содержимое:

```
<?php
$n1=10;           //assign value 10 to variable $n1
$n2=100;          //assign value 100 to variable $n2
$sum=$n1+$n2;     //calculate the sum assign it to
variable $sum
echo "Sum is ".$sum;    //output a string to browser
?>
```

Этот маленький скрипт поможет нам ознакомиться с особенностями PHP и с базовыми принципами его использования. Мы объявили две переменные, вычислили их сумму, занесли результат в третью переменную и вывели в браузер конкатенацию некоей строки и вычисленной суммы.

Как увидеть результат выполнения этого кода? Запомните, для того чтобы выполнить PHP скрипт, недостаточно открыть файл в браузере! Поэтому забудьте,

пожалуйста, о перетаскивании файла с PHP кодом в браузер или о правой кнопке мыши и команде «Открыть с помощью», или о двойном клике по такому файлу. Чтобы PHP выполнялся, его надо передать веб-серверу. Хотя веб-сервер сам работать с PHP не умеет (он понимает только HTML), но получив Request с просьбой выполнить PHP код, он перешлет этот Request находящемуся рядом PHP, получит от PHP результат выполнения кода и перешлет клиенту результат выполнения такого файла в Response. Чтобы отправить наш файл веб-серверу, надо запустить браузер и в адресной строке указать такой путь к файлу:

http://localhost/Test/test1.php

Обратите внимание на значение localhost, оно говорит о том, что веб-сервер расположен на том же компьютере, откуда мы отправляем запрос. Обычно в этом месте адреса указывается имя домена. Если бы мы работали с версией Apache, установленной в нашей локальной сети, но на другом компьютере, например с адресом \\10.3.19.1, то нам надо было бы указать такой адрес:

http://10.3.19.1/Test/test1.php

Если вы написали PHP код без ошибок, то увидите в браузере такой результат:

```
Sum is 110
```

Однако запускать на выполнение PHP файлы можно и из меню OpenServer. Вызовите снова контекстное меню на иконке OpenServer и активируйте пункт меню My Sites. Если вы создали свою папку в папке domains, рядом с localhost, то увидите такую картину:

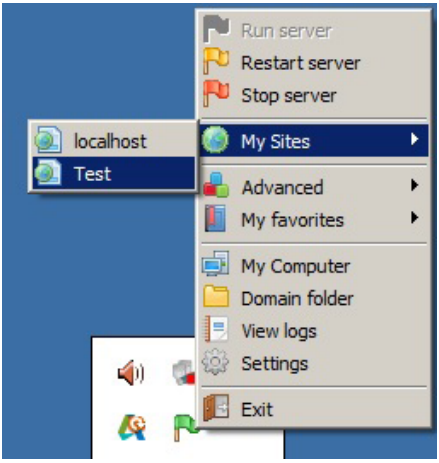

Рис. 11. Запуск сайта из меню OpenServer

Кликните по имени своей папки Test и увидите ее содержимое в браузере:

Рис. 12. Содержимое рабочей папки

Теперь остается только кликнуть по файлу test1.php, и он будет отправлен веб-серверу, а веб-сервер пришлет вам в браузер результат выполнения этого файла. Если

в папке будет файл с именем index.php, то он будет обработан автоматически, без необходимости кликать по нему. Запомните, что имя index – это имя стартового файла по умолчанию.

В нашем случае вы снова получите:

```
Sum is 110
```

Переменные

Переменные, как и в других языках программирования, являются контейнерами данных. В PHP нет специальной строки для объявления переменной. Переменная создается в тот момент, когда вы инициализируете ее каким-либо значением. Имя переменной обязательно должно начинаться со знака «$», после которого можно использовать большие и маленькие английские буквы, цифры и символ «_». Первым символом после «$» не может быть цифра. Имена переменных регистрозависимы. Переменным в PHP не надо указывать тип данных, они автоматически конвертируются к соответствующему типу при инициализации.

Переменные в PHP могут принимать значения таких типов:

- String – строки;
- Integer – целые числовые;
- Float – вещественные числовые;
- Boolean – логические;
- Array – массивы;
- Object – объекты классов.

- Resource – ссылка на внешние ресурсы, например, результаты выполнения запросов;
- NULL – это NULL.

Если переменной не присвоено какое-либо значение, то она равна NULL. С первыми четырьмя типами данных вы уже знакомы. О таких типах, как массивы, объекты и ресурсы мы поговорим подробно в следующих разделах уроков.

Инструмент echo

Это самый важный инструмент языка. Это не функция, а специальная конструкция PHP, которая принимает переданные ей значения, преобразовывает их в строковый вид и пересылает браузеру.

Если ей передать просто строку, она передаст ее в браузер.

Команды в коде	Вывод в браузере
`echo "Hello, world!";`	Hello, world!

Если этой конструкции передать имя какой-либо переменной, она попытается перевести значение переменной в строку и выведет его в браузер.

Команды в коде	Вывод в браузере
`$n1=10;`	
`echo $n1;`	10

Оператор «.» выполняет конкатенацию строк.

Команды в коде	Вывод в браузере
`$name="Ann";` `echo "Hello, ".$name."!";`	Hello, Ann!
`echo $n1;`	10

Строки в PHP можно заключать как в двойные кавычки «Ann», так и в одинарные кавычки ‘Ann’. Но PHP обрабатывает такие строки по разному. Строку в одинарных кавычках PHP не анализирует и передает в браузер без изменения. Строку в двойных кавычках PHP парсит и, если встречает там переменные, подставляет их значения.

Команды в коде	Вывод в браузере

`$n1=10;`
`$n2=100;`
`$sum=$n1+$n2;`
`echo '$n1+$n2='.$sum;`
	$n1+$n2=110

`$n1=10;`
`$n2=100;`
`$sum=$n1+$n2;`
`echo "$n1+$n2=".$sum;`
	10+100=110

Еще можно добавить, что echo можно использовать также и со скобками:

echo (“Hello, “.$name.”!”);

Конструкция echo не единственная в PHP умеет передавать данные для отображения в браузер. Есть еще очень похожая функция print(), выполняющая такое же действие. Есть функция print_r(), позволяющая выводить в браузер массивы. Есть функция var_dump(), умеющая выводить в браузер структуру любых объектов. С этими функциями мы познакомимся в ходе наших занятий.

Связь между PHP и HTML

Имейте в виду, что PHP, и, в частности, функция echo не знают и не понимают ни HTML, ни CSS. PHP умеет делать то, что делают другие языки программирования: хранить данные, выполнять вычисления и т.п. Кроме этого, PHP умеет пересылать в браузер строки (с помощью echo и других подобных функций). Если в этих строках случайно окажется HTML разметка, она будет отправлена в браузер, а уже браузер эту разметку обработает.

Поэтому вы должны заботиться о том, чтобы в передаваемых строках оказывалась необходимая и красивая HTML разметка. Давайте изменим последнюю строку в нашем файле test1.php на такую:

```
echo "<i>Sum is</i> <span style='color:red;'>
".$sum."</span>";
```

Выполните наш измененный файл, и в браузере вы увидите такой результат:

Sum is 110

Выполните в браузере команду «Просмотр кода страницы» и убедитесь, что в этом случае наша разметка попала в браузер. Вот таким образом мы и будем создавать сайты.

Инструкции PHP

PHP поддерживает базовый набор математических операций:

+ суммирование;

- вычитание;
* умножение;
/ деление;
% деление по модулю.

PHP поддерживает, как обычное присваивание =, так и «присваивания с…»:

+= присваивание со сложением;
-= присваивание с вычитанием;
*= присваивание с умножением;
/= присваивание с делением;
.= присваивание с конкатенацией для строк.

PHP поддерживает префиксный и постфиксный инкремент и декремент:

++$a;

$a++;

--$a,;

$a--.

PHP поддерживает логические операторы:
&&, ||, !, xor.

PHP поддерживает такие операции сравнения:

== равны;
!= не равны;
<> не равны;
=== тождественно равны (имеют одинаковые значения и тип);
!== тождественно не равны;

>= больше или равно;

> больше;

<= меньше или равно;

< меньше.

PHP поддерживает операции условных переходов. Простой if:

```
if(condition)
{
      actions;
}
```

Простой if-else:

```
if(condition)
{
      actions1;
}
else
{
      actions2;
}
```

if с несколькими условиями:

```
if(condition1)
{
      actions1;
}
elseif (condition2)
{
      actions2;
}
elseif (condition3)
{
      actions3;
```

```
}
else
{
      default actions;
}
```

Отметьте, что наряду с elseif также можно использовать конструкцию else if. Также в PHP можно пользоваться инструкцией switch. В инструкцию switch передается либо вычисляемое выражение, либо, чаще всего – переменная. В случае выражения, оно вычисляется и анализируется результат вычисления. В случае переменной – просто анализируется значение этой переменной:

```
switch ($role) {
    case «admin»:
        echo «You have full access!»;
        break;
    case «user”:
        echo “ You have full access to data but
cannot change table structure!”;
        break;
    case “operator”:
        echo “Your can only read data!”;
        break;
    default:
        echo “Your role is not recognized!”;
}
```

Массивы и циклы

В PHP очень важную роль играют массивы, которые могут быть индексными или ассоциативными. У массивов первого типа все элементы пронумерованы. Каждый элемент такого массива имеет свой индекс (или номер), и этот индекс является числом.

В ассоциативных массивах элементы тоже имеют индексы, но эти индексы являются строковыми. Поэтому говорят, что в ассоциативном массиве элементы не нумерованы, а именованы. Важно понимать, что массив в PHP – это набор пар ключ–значение. Ключ является ссылкой на конкретный элемент массива. У обычного массива ключ является числом, у ассоциативного – строкой.

Массивы можно объявлять с помощью инструкции array():

```
$ar1 = array();                 //empty array
$ar2 = array(10,20,30);         //array with 3
                                //elements:
                               //$ar2[0] has value 10,
                               //$ar2[1] has value 20
                               //and $ar2[2] has
                                 value 30

$ar3 = array("Argentina","Belgium","Canada");
                     //array with 3 elements:
                     //$ar3[0] has value "Argentina",
                     //$ar3[1] has value " Belgium "
                     //and $ar3[0] has value "Canada"

$ar4 = array("yellow"=>"banana", "red"=>"cherry",
                                  "green"=>"apple");
                                //associative array
```

Начиная с PHP версии 5.4 массивы можно объявлять с помощью [ ]:

```
$ar5 = [10,20,30];
$ar6 = ["yellow"=>"banana", "red"=>"cherry",
                               "green"=>"apple"];
```

Для работы с массивами в PHP существует много полезных функций. С помощью функции count() можно узнать количество элементов обычного или ассоциативного массива. С помощью функции print_r() можно вывести в браузер структуру и содержимое массива.

Существует ряд функций для сортировки массивов:

- sort() – сортирует индексный массив по возрастанию значений элементов;
- rsort() – сортирует индексный массив по убыванию значений элементов;
- asort() – сортирует ассоциативный массив по возрастанию значений элементов;
- ksort() – сортирует ассоциативный массив по возрастанию значений ключей;
- arsort() – сортирует ассоциативный массив по убыванию значений элементов;
- krsort() – сортирует ассоциативный массив по убыванию значений ключей.

Вставьте в наш файл test1.php такой код:

```
$ar4 = array("yellow"=>"banana", "red"=>"cherry",
                                 "green"=>"apple");
echo "Length of array is ".count($ar4)."</br>";
print_r($ar4);
```

и выполните этот файл.

Вы получите в браузере такой вывод:

```
Length of array is 3
Array ( [yellow] => banana [red] => cherry [green] => apple )
```

Обратите внимание на строку Array(…). Это вывод функции print_r(). Она выводит информацию о массиве именно в таком формате, и изменить этот формат нельзя.

Скоро вы узнаете, что в структуру языка встроены так называемые суперглобальные массивы, каждый из которых выполняет какую-либо очень важную для работы PHP функцию.

Циклы

В PHP существует четыре вида циклов:

while()
do-while()
for()
foreach()

Рассмотрим использование этих циклов в нашем файле, добавив туда такой код:

```
<?php
$ar4 = array("yellow"=>"banana", "red"=>"cherry",
                                 "green"=>"apple");
echo "Length of array is ".count($ar4)."</br>";
print_r($ar4);

$ar=array(11,5,89,117,56,200);
echo "</br>From while() loop:</br>";
$i=0;
while($i<count($ar))
{
      echo $ar[$i]." ";
      $i++;
}
echo "</br>";
echo "</br>From for() loop sorted array:</br>";
sort($ar);
for($i=0;$i<count($ar);$i++)
{
```

```
        echo $ar[$i]." ";
}
echo "</br>";
echo "</br>From foreach() loop assotiative array:</br>";
foreach($ar4 as $k => $v)
{
        echo "Key:".$k."  value:".$v."<br/>";
}
?>
```

Обратите внимание: перебирать элементы ассоциативного массива можно только в цикле foreach() с указанием конструкции $k => $v. При этом в переменную $k на каждой итерации будет заноситься значение ключа текущего для этой итерации элемента, а в переменную $v – значение элемента с ключом $k.

После выполнения файла test1.php вы увидите в браузере:

```
Length of array is 3
Array ( [yellow] => banana [red] => cherry [green] =>
                                                    apple )
From while() loop:
11 5 89 117 56 200

From for() loop sorted array:
5 11 56 89 117 200 a

From foreach() loop assotiative array:
Key:yellow value:banana
Key:red value:cherry
Key:green value:apple
```

И еще один момент. Когда вы вводите в адресную строку своего браузера путь к вызываемому файлу http://localhost/Test/test1.php, вы создаете Request к

веб-серверу. В этом Request вы просите веб-сервер прислать вам результат выполнения файла test1.php. У вас есть возможность увидеть сформированный Request. Добавьте в начало файла test1.php такой код:

```
$headers = apache_request_headers();
foreach ($headers as $header => $value) {
    echo "$header: $value <br />\n";
}
```

Функция apache_request_headers() возвращает строки–заголовки созданного Request в виде ассоциативного массива, а цикл foreach() выводит элементы этого массива в читабельном виде. Теперь вы увидели Request, созданный вашим браузером. В моем случае результат работы этого кода выглядит так:

```
Host: localhost
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 Windows  NT 10.0; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/
webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4
Cookie: _ym_uid=14643341631022603274; _ym_isad=2;
__lnkrntdmcvrd=-1
DNT: 1
```

Функции

Вы уже увидели, что в состав PHP входят встроенные функции – такие как count(), print_r() и другие. Мы

познакомимся еще и со многими другими. Кроме этого, как и других языках программирования, мы можем создавать свои собственные функции и затем использовать их в своем коде.

Создание функции в PHP происходит просто. Надо написать служебное слово function, затем имя создаваемой функции и в скобках – список параметров. Обратите внимание, что тип возвращаемого значения функции и даже сам факт наличия или отсутствия такого значения в объявлении функции никак не обозначается. Понять, есть у функции возвращаемое значение или нет, можно только по телу функции.

Создайте в нашей папке Test новый файл с именем functions.php, а в этом файле напишите такой код:

```
<?php
function AddNumbersColor($n1, $n2, $color)
{
      echo "Sum is: <span style='color:".$color.";'>".
($n1+$n2)."</span><br/>";
}
AddNumbersColor(110, 50000,"green");
      ?>
```

Затем активируйте этот файл, указав в браузере такой адрес:

http://localhost/Test/functions.php

В результате выполнения этого файла вы получите в браузере такой вывод:

Sum is: 50110

Рассмотрим написанный код. Мы создали функцию с именем AddNumbersColor(), принимающую два числа и название цвета. Эта функция не имеет никакого возвращаемого значения, так как в ней нет оператора return. Она просто выводит в браузер строку, содержащую фразу «Sum is: », и сумму переданных ей чисел. Значение суммы выводится в , а цвет текста этого элемента определяется значением третьего параметра. Функции в PHP могут иметь параметры со значениями по умолчанию. По законам стека, параметры со значениями по умолчанию должны располагаться в списке параметров последними. Измените нашу функцию и ее вызов таким образом:

```
function AddNumbersColor($n1, $n2, $color='#EE33FF')
{
      echo "Sum is: <span style='color:".$color.";'>".
($n1+$n2)."</span><br/>";
}
AddNumbersColor(110, 50000);
```

Теперь результат работы функции выглядит так:

Sum is: 50110

Что мы сделали? Добавили третьему параметру функции значение цвета по умолчанию. Обратите внимание, что PHP прекрасно понимает шестнадцатеричные описания цветов. При вызове функции мы не передали никакого значения для третьего параметра, и функция использовала значение цвета по умолчанию.

Давайте создадим функцию, которая что-то возвращает. Добавьте в файл такой код:

```
function AddNumbers($n1, $n2)
{
      return ($n1+$n2);
}
$total = AddNumbers(1,2)+AddNumbers(100,1000);
echo $total;
```

Как видите, функция AddNumbers() принимает два числа и возвращает их сумму. Никакого видимого вывода в браузер эта функция не производит. Вызов этой функции отличается от вызова предыдущей, так как теперь нам надо получать возвращаемое значение. Вы понимаете, что эту функцию можно использовать в выражениях, как в нашем примере. Мы суммируем две пары чисел, заносим сумму в переменную $total и затем выводим значение этой переменной.

При вызове нашего файла вы увидите в браузере значение 1103, но это не вывод из функции. Это результат вывода значения переменной $total после вычисления выражения.

Параметры в PHP функции передаются по значению. Давайте рассмотрим пример, подтверждающий это. Добавьте в наш файл еще такой код:

```
function IncNumberByTen($n1)
{
      $n1+=10;
      echo 'From function:'.$n1.'</br>';
}
$number=5;
IncNumberByTen($number);
echo 'From outside function:'.$number.'</br>';
```

Функция IncNumberByTen() принимает одно число и увеличивает его значение на 10. После этого выводит измененное значение в браузер. Возвращаемого значения у этой функции нет. Для демонстрации работы функции мы создали переменную $number со значением 5 и передали эту переменную в функцию. Функция увеличила полученное число на 10 и вывела измененное значение. Это показывает строка вывода:

```
From function:15
```

Когда же мы выводим значение переменной $number после вызова функции, мы видим не измененное исходное значение 5. Вы понимаете, почему это происходит. Потому, что функция, получив при вызове параметр, создала для себя его локальную копию и увеличила на 10 именно эту локальную копию, а не переменную $number. Поэтому, когда мы вывели значение исходной переменной после вызова функции, то увидели все то же значение 5.

```
From outside function:5
```

Такое поведение функции можно изменить, передавая ей значение параметра по ссылке. Добавьте в наш файл еще одну функцию:

```
function IncNumberByTenRef(&$n1)
{
      $n1+=10;
      echo 'From function:'.$n1.'</br>';
}

$number2=5;
IncNumberByTenRef($number2);
echo 'From outside function:'.$number2.'</br>';
```

Эта функция очень похожа на предыдущую. Единственное отличие состоит в наличии символа **&** перед параметром. Этот символ указывает на то, что параметр надо передавать по ссылке, а не создавать его локальную копию в функции. Вывод подтверждает, что после вызова этой функции изменилась исходная переменная $number2:

```
From function:15
From outside function:15
```

Вы должны запомнить еще такой факт: тело PHP функции создает отдельную область видимости, и поведение этой области видимости отличается от привычного вам. Добавим еще такой код в наш файл:

```
$title="Chapter 1";
function ColorTitle($color)
{
      echo "<h1 style='color:".$color.";'>".$tit
le."</h1>";
}
ColorTitle('#EE33FF');
```

Теперь немного измените предыдущий пример и вызовите функцию до ее определения:

```
ColorTitle('#EE33FF');
$title="Chapter 1";
function ColorTitle($color)
{
      echo "<h1 style='color:".$color.";'>".$tit
le."</h1>";
}
```

Активируйте этот файл. Возможно, вы будете удивлены, но вызов функции отработал правильно. Дело в том, что PHP сначала парсит файл с кодом, а потом выполняет. Поэтому объявление функции было обработано при первом проходе.

У нас есть некий заголовок в переменной $title, и мы написали функцию, позволяющую выводить такие заголовки в разных цветах. Но после вызова этой функции мы почему-то не увидели вообще никакого заголовка. В чем здесь ошибка?

Дело в том, что функция, встречая переменную $title, не понимает, что мы хотим работать с переменной $titlle, определенной вне функции. Наша функция создала себе свою собственную переменную $title, а поскольку значение этой созданной переменной NULL, то мы ничего и не увидели. Хотя, если бы вы не поленились и посмотрели код страницы, полученной в результате последнего вызова нашего файла, вы бы увидели там такую разметку:

```
<h1 style='color:#EE33FF;'></h1>
```

Наша функция выполнилась! Она честно сформировала вывод, но значение для созданного заголовка оказалось пустым, и мы ничего не увидели.

Как сделать так, чтобы функция поняла, что мы хотим использовать переменную, созданную вне ее области видимости? Для этого в PHP есть инструкция global. Добавьте в нашу функцию одну строку и снова вызовите ее:

```
function ColorTitle($color)
{
      global $title;
      echo "<h1 style='color:".$color.";'>".$tit
le."</h1>";
}
```

Теперь вы увидите красивый заголовок:

Chapter 1

include и require

Будет очень хорошо, если, глядя на код файла functions.php, вы испытываете чувство неудовлетворенности его внешним видом. Это говорит о вашем профессионализме. Даже если вы строго соблюдаете все отступы, код этого файла плохо читается, потому что в нем перемешаны определения функций и их вызовы. В PHP есть хорошие инструменты, чтобы исправить эту ситуацию.

Создайте в той же папке Test еще один файл с именем mycalls.php и вставьте в него вызовы всех написанных нами функций из файла functions.php. Созданный файл должен выглядеть таким образом:

```
AddNumbersColor(110, 50000);

$total = AddNumbers(1,2)+AddNumbers(100,1000);
echo $total.'</br>';


$number=5;
IncNumberByTen($number);
echo 'From outside function:'.$number.'</br>';

$number2=5;
IncNumberByTenRef($number2);
echo 'From outside function:'.$number2.'</br>';

$title="Chapter 1";
ColorTitle('#EE33FF');
```

Если вы вызовете сейчас фай mycalls.php, то получите сообщение об ошибке. Это понятно, поскольку определений вызываемых функций в этом файле нет. Измените код файла functions.php, оставив в нем только определения всех наших функций, и удалив все вызовы:

```
function AddNumbersColor($n1, $n2, $color='#EE33FF')
{
      echo "Sum is: <span style='color:".$color.";'>".
                              ($n1+$n2)."</span><br/>";
}

function AddNumbers($n1, $n2)
{
      return ($n1+$n2);
}
function IncNumberByTen($n1)
{
      $n1+=10;
      echo 'From function:'.$n1.'</br>';
}
```

```
function IncNumberByTenRef(&$n1)
{
      $n1+=10;
      echo 'From function:'.$n1.'</br>';
}

function ColorTitle($color)
{
      global $title;
      echo "<h1 style='color:".$color.";'>".$tite."
                                                </h1>";
}
```

Сейчас мы имеем два красивых файла. В одном содержатся определения наших функций, в другом – их вызовы. Код в обоих файлах красивый, читабельный, в нем нет ничего лишнего. Есть, правда, один маленький недостаток – этот код не работает. Но мы сейчас это исправим. Добавьте в самое начало файла mycalls.php такую строку:

```
include('functions.php');
```

и вызовите этот файл снова. В этот раз вы получите в браузере результаты работы всех наших функций:

Sum is: 50110
1103
From function:15
From outside function:5
From function:15
From outside function:15

Chapter 1

Если вы помните C++, то догадаетесь, что функция include() принимает путь к какому-либо файлу и включает код этого файла в место своего вызова. Эта функция включила в файл mycalls.php определения наших функций из файла functions.php. include() не единственная функция, позволяющая выполнять такие действия.

Давайте представим, что мы работаем над большим проектом, в составе которого много страниц кода. Эти страницы ссылаются одна другую с помощью функции include(). Что произойдет, если на какой-то странице окажется два include() на один и тот же файл? Давайте проверим. Вставьте в файл mycalls.php еще одну строку include('functions.php') и вызовите этот файл. Вы получите ошибку. Это понятно, так как теперь в mycalls.php возникает переопределение функций из functions.php. Вы должны согласиться с тем, что если такая систуация возникает случайно, то найти источник ошибки может быть очень трудно. Чтобы избавить разработчиков от таких проблем, в PHP добавили функцию include_once(). Она ведет себя точно так же, как функция include(), но если вы вставите в какой-либо файл хоть десять вызовов include_once() с одним и тем же файлом, никакой ошибки не произойдет. Просто девять последующих подключений будут проигнорированы.

В составе PHP есть еще пара функций, подобных include() – require() и require_once(). Они работают так же, как и include(), но при этом, если не найден требуемый для вставки файл, возвращают FATAL ERROR, а функции include() и include_once() в этом случае возвращают только WARNING. Поведение функций require() существенно отличалось от функций include() в PHP до

версии 4.2. В старых версиях PHP, функции require() не реагировали на условный вызов. Если вы вставляли вызов функции require() внутрь инструкции if, а при этом условие в этой инструкции if было ложным, функция require() все равно выполнялась. Функции include() и тогда и сейчас реагируют на условный вызов. Однако сегодня, функции require() тоже реагируют на условный вызов и отличаются от функций include() только возвращаемым значением, как указано выше.

Метод GET

Мы уже говорили о протоколе HTTP, , в котором определен формат Request и Response. Сейчас рассмотрим некоторые его особенности внимательнее. Этот протокол создавался, как протокол передачи гипертекста между узлами в глобальной сети. По природе – это текстовый протокол, позволяющий передавать гипертекстовые документы в формате HTML. Этот протокол предполагает, что существуют сервера, на которых располагается некая информация. И существуют клиенты, которым нужна информация, располагающаяся на серверах. Чтобы получить информацию от сервера, клиент должен сформировать запрос к серверу (Request) и в нем указать, какая информация ему нужна. В терминах HTTP, такая информация называется ресурсом. Каждый ресурс в глобальной сети имеет свой уникальный идентификатор, называемый URI (Unified Resource Identifier). Так вот, именно пересылкой таких ресурсов и занимается протокол HTTP. А при обработке ресурсов этот протокол использует URI. При передаче ресурса

HTTP позволяет кодировать пересылаемые данные. Поэтому, несмотря на свою текстовую природу, этот протокол позволяет передавать и бинарные данные.

В протоколе HTTP определено несколько способов передачи данных, называемых методами. В данном случае этот термин совпадает с общеупотребительным значением этого слова. Это не функция, определенная в классе, а именно способ, метод передачи данных. Сейчас мы познакомимся с одним из этих методов, по названию GET. Этот метод предназначен для пересылки клиенту данных по заданному клиентом URI. Обратите внимание, что метод GET только пересылает данные, но не может выполнять с ними никаких других действий, типа кодировки или чего-то другого. При этом клиент может передавать в адресной строке вместе с URI и дополнительные данные на сервер. Для передачи данных используется знак «?». Если в адресной строке присутствует этот знак, то все, что находится справа от него – это данные в формате пар «ключ=значение». Рассмотрим использование этого метода.

Передача данных на сервер методом GET

Вы уже много раз вводили значение в адресную строку браузера, и на основе введенного значения создавался Request к веб-серверу. Однако через адресную строку веб-серверу можно передавать и данные. Для этого в адресной строке используется символ «?». Запомните, этот символ в адресной строке означает передачу данных. Он является разделителем между адресом и данными. Все, что расположено слева от символа «?», это адрес, все что справа – это данные. Рассмотрим

пример. Создайте новый файл с именем testget.php и добавьте в него такой код:

```
$name=$_GET['name'];
$country=$_GET['country'];
$city=$_GET['city'];

echo 'I received such data:</br>';
echo 'Name is: '.$name.'<br/>';
echo 'Country is: '.$country.'<br/>';
echo 'City is: '.$city.'<br/>';
```

Очень скоро вы узнаете, что такое $_GET[]. Пока просто попробуйте выполнить этот файл, введя в адресную строку браузера такой адрес:

http://localhost/test/testget.php

Вы получите пустой ответ, выглядящий так:

```
I received such data:
Name is:
Country is:
City is:
```

Хотя, если вы используете для работы другую конфигурацию, не OpenServer, вы могли бы получить и сообщение об ошибке, говорящее о том, что элементы $_GET[‘name’] и другие не существуют. Для нас сейчас это не важно.

Давайте разберем этот код. Мы уже упоминали о существовании в PHP так называемых суперглобальных массивов. $_GET[] – один из них. Все суперглобальные (или просто глобальные) массивы являются ассоциативными. Их создавать не надо. Они входят в структуру

языка и создаются всегда, независимо от того, используем мы их или нет. Имена большинства из них начинаются с $_ , поэтому нам не стоит называть свои переменные таким образом, чтобы первым символом имени было подчеркивание «_». Имена всех глобальных массивов состоят из символов в верхнем регистре. Каждый глобальный массив выполняет определенные функции и используется в соответствующих ситуациях.

Массив $_GET[] используется при пересылке данных методом, который в протоколе HTTP называется get или GET. Имя метода регистронезависимо. Метод get выполняет пересылку данных через адресную строку. Упоминавшийся разделитель «?» – это атрибут метода get. Как это работает? Очень просто. Добавьте в адресную строку после адреса символ «?». После него укажите пересылаемые данные в формате пар ключ=значение. Если пар требуется несколько, разделяйте их символом «&». Также запомните, что в строке после символа «?» не может быть ни одного пробела до самого завершения строки.

Введите в браузер такую строку:

http://localhost/test/testget.php?name=Kelly&country=USA&city=Seattle

Теперь вы получите ответ, выглядящий так:

```
I received such data:
Name is: Kelly
Country is: USA
City is: Seattle
```

Как это работает? Если в адресной строке присут-

ствует «?», в работу включается массив $_GET[]. PHP читает пары ключ=значение, указанные в адресной строке после «?», и для каждой пары создает в массиве $_GET[] новый элемент. Индексом этого элемента является первое значение в паре, а значением этого элемента является второе значение в паре.

Таким образом, для пары name=Kelly в этом массиве создается элемент $_GET[‘name’], и в этот элемент заносится значение ‘Kelly’. Аналогично, в массиве $_GET[] создаются еще два элемента: $_GET[‘country’] и $_GET[‘city’] со своими значениями. Остальное вам понятно. Файл просто читает из массива $_GET[] соответствующие элементы и заносит их в переменные, из которых формируется вывод. Понятно, что эти переменные можно было не создавать, а выводить значения прямо из элементов $_GET[]:

```
echo 'Name is: '.$_GET['name'].'<br/>';
echo 'Country is: '. $_GET['country'].'<br/>';
echo 'City is: '. $_GET['city'].'<br/>';
```

Если вы вызовете этот файл с другими данными, вы получите другой вывод. Понятно, что ключи изменяться не должны. Вызов:

http://localhost/test/testget.php?name=Bob&country=Canada&city=Aurora

Вывод:

```
I received such data:
Name is: Bob
Country is: Canada
City is: Aurora
```

Что можно сказать о таком способе передачи данных на сервер? Он простой и быстрый. Однако объем пересылаемых данных очень ограничен. В зависимости от разных факторов и браузера, объем передаваемых данных составляет от приблизительно 150 до 300 символов. При этом данные, как вы видите, передаются в открытом виде и представляют собой строковые значения.

Вы познакомились с первым глобальным массивом PHP. Впереди еще много других. Но об этом мы поговорим в следующих уроках.

Домашнее задание

Перейдем к домашнему заданию. Вам надо выполнить следующую работу.

1. Написать PHP-скрипт, в котором создать текстовый массив с названиями цветов. Количество разных цветов должно быть больше четырех. Затем с помощью PHP отобразить на странице четыре div одинакового размера. Сделать так, чтобы при загрузке страницы все четыре div заливались случайными цветами, выбранными из массива. При этом, все четыре div должны заливаться разными цветами. Ни один цвет не должен повторяться.
2. Создать PHP функцию, принимающую один целочисленный параметр m, со значением в диапазоне 1-12. При вызове эта функция должна создавать в теге table календарь для месяца с номером m для текущего года. Таблица должна иметь такой вид:

   Пн Вт Ср Чт Пт Сб Вс

   Продумать оформление таблицы самостоятельно.