***Цифры не врут…***

***Неизвестный мудрец***

###### Глава 6. Язык серверных скриптов PHP

**Введение**

Скриптовый язык, называемый PHP и предназначенный для создания серверных приложений сайтов, был разработан Р. Лердорфом в 1994г. Современная аббревиатура PHP происходит от Hypertext Preprocessor (Препроцессор гипертекста). Скрипты PHP обрабатывают на сервере данные из передаваемого браузером запроса клиента и возвращают клиенту результат в виде гипертекстового документа.

Использование PHP-скриптов позволяет добавить на сайт формы для обратной связи, счетчики посещений, гостевые книги, форумы, голосования, регистрацию и авторизацию, поиск информации, новости. На PHP можно генерировать динамические страницы, пересылать файлы cookies, содержащие индивидуальную информацию о клиенте. PHP обеспечивает поддержку различных баз данных. Широкое распро­странение получила связка PHP с СУБД MySQL. Вместе с PHP7 используются и нереляционные базы данных, например PostgreSQL и Redis. PHP умеет работать с почтовыми протоколами POP3 и SMTP, с протоколами TCP/IP, HTTP и другими, обеспечивает работу с файлами и сокетами.

Синтаксис PHP подобен синтаксису C++, хотя объектная модель ближе к модели Java. С использованием РНР созданы энциклопедия Wikipedia, социальные сети Facebook, "ВКонтакте", электронная площадка объявлений Avito.

В мире PHP остается одним из самых поnулярных и востребованных языков. В основе популярности РНР лежат следующие достоинства:

Изначальная ориентация на Wеб-разработку и создание Wеб-сайтов.

Кроссплатформенность - РНР перенесен на все основные операционные системы: Windows, Маc OS Х, Linux-cepвepы.

Бесплатность - PHP имеет открытый бесплатный исходный код. Не потребуется платить ни за сам язык, ни за большинство сопутствующих программ (редакторы, Wеб-серверы, базы данных).

Низкий порог входа - изучить РНР и создавать на нем готовые приложения проще, чем с использованием конкурирующих технологий (.NET, Python, Ruby, Go ).

Недостатками являются: отсутствие лидера, непоследовательный и несовместимый синтаксис. Сам РНР в настоящий момент стал полноценным объектно-ориентированным языком. Однако в нем полно старых процедурных артефактов, которые предлагается использовать. В РНР-разработку одновременно было вовлечено огромное количество программистов по всему миру. Все это породило большое число самых разных подходов, фреймворков и несовместимых друг с другом систем.

В 2015 году PHP подвергся значительному обновлению и дополнению, что привело к повышению производительности в версии PHP7 и более эффективному распределению памяти. Тесты, выполненные на реальных приложениях показывают, что PHP7 в среднем вдвое быстрее PHP5., а также использует на 50% меньше памяти вовремя обработки запросов. Ситуация с совместимостью исправляется при помощи РSR-стандартов.

Инструменты для разработки

Хорошим редактором для разработки кода PHP является редактор Adobe DreamWeaver с возможными расширениями (JQuery API или PHP Storm). Программисты Java могут использовать для разработки PHP-скриптов среду разработки Eclipse и плагины JSEclipse и Aptana Studio.

Чтобы выполнить код PHP, необходим веб-сервер **(**устаревающий Apache или nginx, или другой), препроцессор PHP и сервер БД MySql. Для работы с файлами полезной будет программа Total Comander, которая позволяет также разместить PHP-приложение на удаленном сервере. Для этого необходим хостинг сайта и DNS-адрес.

Отладка PHP-сайта возможна на локальном (localhost) компьютере. Для быстрой установки сервера Apache и БД MySql под Windows можно использовать пакет Open Server. В этот пакет входит еще и SMTP-сервис для работы с почтой. Из других установочных пакетов можно порекомендовать WAMP. Для MacOS можно использовать MAMP. Одним из лучших браузеров, используемых при разработке, является Firefox.

Что представляет собой РНР?

Язык РНР имеет ядро и расширения языка. Код, созданный на РНР, условно можно поделить на следующие типы:

- компоненты - библиотеки на РНР, которые собираются при помощи менеджера пакетов Composer *;*

- фреймворки - готовые сборки, из компонентов, при помощи которых

можно создавать сайты любой степени сложности. На следующие РНР-фреймворки стоит обратить внимание: Laravel, Zend, Yii.

- готовые приложения - готовые к использованию разработки на РНР. Это системы управления контентом (Wordpress, Drupal), форумы (phpBB),

- Wеб интерфейсы управления базами данных (phpMyAdmin, pgAdmin).

Кроме этого для запуска сайта потребуются дополнительное программное обеспечение и технологии:

- Сервер базы данных. Кроме MySql в PHP7 довольно популярными являются базы данных: PostgreSQL и Redis.

- Система контроля версий Git, служит для хранения истории разработки, резервного копирования, доставки кода на сервер, организации командной работы.

- Wеб-сервер- программа, которая обеспечивает взаимодействие клиента и

приложения посредством протокола НТТР. Для эксплуатации настоящего

сайта потребуется Wеб-сервер nginx или Apache.

**Встроенный веб-сервер**

В современных версиях можно начать изучение PHP в консольном режиме, используя встроеннный веб-сервер, без установки и конфигурирования полноценного веб-сервера. Для запуска встроенного сервера необходимо выполнить следующую команду с консоли в корневой папке веб-проекта:

> php -S localhost:8000

Опция–i выдаст конфигурацию PHP, подобно функции phpinfo().

> php -i

Напишем простую программу с именем hello.php.

<?php

if ($argc != 2) {

exit(1);

}

$name = $argv[2];

echo "Привет, $name\n";

PHP устанавливает две специальные переменные . [$argc](http://php.net/manual/ru/reserved.variables.argc.php) и [$argv](http://php.net/manual/ru/reserved.variables.argv.php) – аргументы командной строки, с которыми запущен скрипт. [$argc](http://php.net/manual/ru/reserved.variables.argc.php) содержит количество переданных аргументов, [$argv](http://php.net/manual/ru/reserved.variables.argv.php)— это массив строк, содержащий значение каждого аргумента. [$argv](http://php.net/manual/ru/reserved.variables.argv.php)[1] — всегда содержит название PHP скрипта, например hello.php.

Выражение exit() используется с ненулевым числом, если команда не удалась.

Для запуска сценария наберите в командной строке:

> php hello.php Мир

Результат:

*Привет, Мир*

Как PHP работает

Как и для JavaScript, PHP-код включается внутрь HTML-документа. Расширение файла, содержащего код, может быть PHP, HTM, HTML или PHTML. В конфигурационном файле веб-сервера Apache необходимо добавить директиву AddType обработки других расширений файлов. Препроцессор PHP может быть установлен на сервере в одном каталоге вместе с Apache или в другом месте, важно, чтобы в конфигурационном файле веб-сервера был правильно прописан к нему путь.

Перед началом работы с PHP-скриптами необходимо запустить Веб-сервер Apache. Обычно при этом запускается и сервер MySql. Затем надо загрузить браузер и в окне набрать адрес [www.localhost/name.php](http://www.localhost/name.php). Если сервер Apache не подключается, страница будет просто отсылаться пользователю в необработанном виде.

Код PHP может быть оформлен несколькими способами:

<?php //стиль xml, основной

//php инструкции

?>

Это основной вариант включения кода PHP в HTML-документ. Затем документ с включениями скрипта сохраняется в файле с расширением .php или .htm.

Пример:

<!–ppr0.php -->

<html>

<head> </head>

<?php //Основной вариант вставки PHP-кода

echo "Внутри PHP – при основном способе вставки кода PHP<BR><HR>";

phpinfo(); //Возвращает много информации о php

?>

<body> Вставка кода PHP внутрь документа Html<BR>

</body></html>

Вывод:

*внутри PHP – при основном способе вставки кода PHP;*

Функция phpinfo(), возвращающая большое количество информации, здесь выключена из-за экономии места. В выходном документе вставки PHP заменяются на строки, выводимые с помощью echo (string arg1, string argn). Для вывода могут использоваться также операторы print. В этих операторах скобки не обязательны в отличие от оператора форматированного вывода printf("format", arglist). PHP позволяет использовать и более сложные структуры для вывода большого текста документа:

<?php //ppr1.php

$exp = true;

if ($exp) {

?>

<strong>Это истина.</strong>

<?php

}

else {

?>

<strong>Это ложь.</strong>

<?php

}

printf("<br> %s", "Выход из PHP скрипта позволяет вывести большой текст");

?>

Вывод:

э*то истина.*

*Выход из PHP скрипта позволяет вывести большой текст.*

Когда PHP встречает закрывающие теги “?>”, он выводит в текст гипертекстового документа все, что находит до следующего открывающего тега. Выход из режима PHP более эффективен, чем отправка текста через echo или print.

В результате выполнения кода PHP создается новый гипертекстовый документ, готовый к отображению браузером. У этого документа PHP-вставки заменены на блоки, выведенные через echo или print.

Пример

### 6.1. Описание языка

**Типы данных**

PHP поддерживает 4 скалярных типа данных(литералы):

boolean (значения true и false),

integer (целые числа, которые могут быть записаны как десятичные, восьмеричные или шестнадцатеричные). Если разрядность составляет 64 бита, диапазон возможных значений -от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

float (длинные вещественные числа, соответствующие типу double языка Cи), значение которого составляет от ±2.23х10-308 до ±1.79х1О308

string(строки в одинарных или двойных апострофах). Может хранить строку, максимальный размер которой составляет 2 Гбайт

два составных типа: array и object.

array представляет массив. Это объединение нескольких переменных под одним именем. Доступ к индексированной переменной осуществляется при помощи индекса массива.

object - Это конструкция, объединяющая несколько полей - переменных и функции - методы их обработки

Имеется два специальных типа: resource и NULL. Дескрипторы resource применяются при работе с файлами, базами данных, динамическими изображениями и т. д

nul1 - Специальный тип, который сигнализирует о том, что переменная не была инициализирована;

callback - Некоторые функции РНР могут принимать в качестве аргументов другие callback - функции, которые называются функциями обратного вызова. Переменные данного типа содержат ссылки на такие функции

В текущей главе мы изучим лишь целые, вещественные числа, логический тип, строки.

Перед целыми восьмеричными значениями записывается нуль, перед шестнадцатеричными – 0x. Пример:

<?php //ppr2.php

$a = 10;

$b = 020;//Восьмеричное значение

$c = 0x41;//Шестнадцатеричное значение

$d = 3.62e+2;//Double

printf("%d %o %c %f",$a,$b,$c,$d);

?>

Вывод:

*10 20 A 362.000000*

В операторе printf() символы означают: %d – формат для десятичного целого, %o – для восьмеричного, %c – для вывода символа с указанным кодом, %f – для вывода вещественных значений.

Если необходимо проверить тип и значение выражения, можно использовать функцию var\_dump(). Если же для отладки необходимо представление типа, следует использовать gettype(). Чтобы проверить выражение на определенный тип, применяют функцию is\_*type()*.

Имена переменных в PHP начинаются символом $. Тип переменной определяется по значению. Если присвоить переменной $p строковое значение, $p становится строкой. Затем можно присвоить $p значение целого числа. При инициализации переменной присваивается значение константы или поименованной константы, объявляемой в операторе define() без знака $. Пример:

<?php //ppr3.php

define("d",5,true);//константа

$a = "Добро пожаловать ";

$b = "в PHP";

$c = 4;

$c++;

echo "$a$b$c.<br>"; //Получим – Добро пожаловать в PHP

print d;//Вывод константы 5

$p = NULL;

$a = "Маша";

$b = "Даша";

$c=$a+$b;//0 - Это не кантактенация

echo "<br>$c";

print var\_dump($p);

print gettype($a);

?>

Вывод:

*Добро пожаловать в PHP5.  
5  
0NULL string*

Интересно, что при сложении строк получаем нуль.

Комментарии в PHP используются, как и в C++: */\*многострочный комментарий \*/* или *// однострочный комментарий* или *#еще один однострочный комментарий.*

Строки в *PHP7* являются последовательностями символов, каждый символ 2 байта. Это означает, что PHP7 имеет встроенную поддержку Unicode, переход на которую реализован в версии PHP5.5. Строковый литерал можно представлять двумя способами: строки в одиночных кавычках и строки в двойных кавычках. Строка, заключенная в двойные кавычки, может содержать подстроку в одинарных кавычках и наоборот. В строке, заключенной в двойные кавычки, переменные будут заменены их значениями. Если строка заключена в одинарные кавычки, такая замена не производится, эта строка не разбирается в PHP.

<?php //ppr4.php

$name = 'Domine?';

$name\_1 = "Quo vadis, $name";

$name\_2 = ' Quo vadis, $name';

echo "$name\_1"; // Выведет – Quo vadis, Domine?

echo "$name\_2"; // Выведет – Quo vadis, $name

?>

Вывод:

*Quo vadis, Domine? Quo vadis, $name*

Для строк в двойных кавычках PHP поддерживает экранированные символы: \n – перенос строки; \r – возврат каретки; \t – горизонтальная табуляция; \\ – обратная косая черта; \$ – знак доллара; \\" – двойная кавычка; \код – код символа. Например:

$s="\x41 - это символ ‘a’ ";

$st=" можно использовать \" управляющий \" символ";

Еще один способ определения строк – использование heredoc-синтаксиса ("<<<"). После символа <<< указывается идентификатор, затем следует строка, а потом этот же идентификатор, закрывающий строку. Закрывающий идентификатор должен начинаться в первом столбце строки. Строка не содержит других символов за исключением точки с запятой (;).Первым символом перед закрывающим идентификатором должен быть символ новой строки. Например, \r. Пример:

<?php //ppr5.php

$name = "Max";

$str = <<<DEMO

Hello $name! <br/>

This is a demo message

with heredoc.

DEMO;

echo $str;

?>

Вывод:

*Hello Max!*

*This is a demo message with heredoc*.

Чтобы определить конец имени переменной в строке, можно заключить имя в фигурные скобки.

echo "He like some {$apple}s"; // Работает

Массивы и их инициализация

PHP поддерживает массивы с числовыми индексами и ассоциативные массивы, в качестве индексов в которых используются строки. Индексы массивов в квадратных скобках изменяются начиная с 0:

$massiv[0] = 'name';

$massiv[1] = 'phone';

Ассоциативные массивы (хэши) используют в качестве индексов строковые значения – ключи.

$new\_mas['name'] = 'Valera';

$new\_mas['email'] = 'rvs@bsu.by';

Массив может инициализироваться одним из двух способов: последовательным присвоением значений или посредством конструктора array().

$massiv[] = "третий"; //$massiv[2] = " третий "

$massiv[] = "четвертый";//$massiv[3] = " четвертый "

Можно задать сразу значения всех элементов массива:

$a = [1, 2, 3];

$b = ['one': 'orange', 'two': 'apple', 'three': 'lemon'];

Второй способ создания массива реализуется вызовом функции array():

$massiv = array('one','two','three');

Для ассоциативных массивов такой вызов будет иметь следующий вид:

$new\_massiv = array('name' => 'nobody', 'email' => 'mail@bsu.by');

Пример:

<?php //ppr6.php

$massiv = array('one','two','three');

$massiv[5] = "5";

$massiv[6] = "6";

$massiv[] = "7";

print\_r($massiv);

$new\_massiv = array('name' => 'nobody', 'email' => 'mail@bsu.by');

print\_r($new\_massiv);

print $new\_massiv['name']

?>

Результат:

*Array ( [0] => one [1] => two [2] => three [5] => 5 [6] => 6 [7] => 7 )*

*Array ( [name] => nobody [email] => mail@bsu.by ) nobody*

Функция print\_r() используется для вывода всего массива.

PHP содержит суперглобальные переменные-массивы, которые всегда доступны для всех блоков и используются для передачи данных от клиента к серверу. Вот эти суперглобальные переменные: $\_GET[], $\_POST[], $\_ENV[], $\_REQUEST[], $\_COOKIE[], $\_FILES[], $\_SERVER[], $\_SESSION[].

Массив $\_ENV содержит переменные среды окружения. Другие переменные будут описаны в дальнейшем. Следующий пример выполняет вывод значений элементов суперглобального массива $\_SERVER

<?php //ppr7.php

//print\_r($\_GET); //print\_r($\_POST); //print\_r($\_REQUEST);

//print\_r($\_COOKIE ); //print\_r($\_FILES);

print\_r($\_SERVER);

//print\_r($\_ENV); //print\_r($\_SESSION);

?>

Результаты:

*Array*

*( [COMSPEC] => C:\WINDOWS\system32\cmd.exe*

*[DOCUMENT\_ROOT] => z:/home/localhost/www*

*[HTTP\_ACCEPT] => image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/msword, application/vnd.ms-powerpoint, \*/\**

*[HTTP\_ACCEPT\_ENCODING] => gzip, deflate*

*[HTTP\_ACCEPT\_LANGUAGE] => ru*

*[HTTP\_CONNECTION] => Keep-Alive*

*[HTTP\_HOST] => localhost*

*[HTTP\_USER\_AGENT] => Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 2.0.50727)*

*[PATH] => \usr\local\ImageMagick;\usr\local\php5;*

*[REMOTE\_ADDR] => 127.0.0.1*

*…*

*[REQUEST\_METHOD] => GET*

Операции и выражения

Арифметические операции Таблица 6.1.

|  |  |
| --- | --- |
| **Пример** | **Результат** |
| $a + $b | Сумма $a и $b. |
| $a - $b | Разность $a и $b. |
| $a \* $b | Произведение $a и $b. |
| $a / $b | Частное от деления $a на $b. |
| $a % $b | Целочисленный остаток от деления $a на $b. |

Операция деления ("/") всегда возвращает число с плавающей точкой, даже если операнды являются целыми числами или строками.

Битовые операции

Битовые операции дают возможность устанавливать значение битов для целых чисел. Если операнды являются строками, битовая операция выполняется над символами строки.

Битовые операции Таблица 6.2.

| **Пример** | **Имя** | **Результат** |
| --- | --- | --- |
| $a & $b | And | Устанавливает биты, установленные и в $a, и в $b. |
| $a | $b | Or | Устанавливает биты, установленные в $a или в $b. |
| $a ^ $b | Xor | Устанавливает биты, установленные в $a или $b, но не в обоих. |
| ~ $a | Not | Устанавливает биты, которые в $a не установлены,. |
| $a << $b | Сдвиг влево | Сдвигает биты переменной $a на $b шагов влево. |
| $a >> $b | Сдвиг вправо | Сдвигает биты переменной $a на $b шагов вправо. |

Операции сравнения

Операции сравнения Таблица 6.3.

|  |  |  |
| --- | --- | --- |
| **Пример** | **Название** | **Результат** |
| $a == $b | равно | **TRUE**, если $a равно $b. |
| $a === $b | идентично | **TRUE**, если $a равно $b и они одного типа. |
| $a != $b | не равно | **TRUE**, если $a не равно $b. |
| $a <> $b | не равно | **TRUE**, если $a не равно $b. |
| $a !== $b | не идентично | **TRUE**, если $a не равно $b или они разных типов. |
| $a < $b | Меньше | **TRUE**, если $a строго меньше $b. |
| $a > $b | Больше | **TRUE**, если $a строго больше $b. |
| $a <= $b | Меньше-равно | **TRUE**, если $a меньше или равно $b. |
| $a >= $b | Больше-равно | **TRUE**, если $a больше или равно $b. |

Операция вопросительный знак: (ex1) ? (ex2) : (ex3). При этом значение выражения равно ex2, если ex1=TRUE, и ex3, если ex1= FALSE.

Логические операции

Логические операции Таблица 6.4.

|  |  |  |
| --- | --- | --- |
| **Пример** | **Имя** | **Результат** |
| $a and $b | And | **TRUE**, если и $a, и $b **TRUE**. |
| $a or $b | Or | **TRUE**, если $a или $b **TRUE**. |
| $a xor $b | Xor | **TRUE**, если $a или $b **TRUE**, но не оба. |
| ! $a | Not | **TRUE**, если $a не **TRUE**. |
| $a && $b | And | **TRUE**, если и $a, и $b **TRUE**. |
| $a || $b | Or | **TRUE**, если $a или $b **TRUE**. |

Операции and и or выполняются так же, как && и ||, однако имеют более высокий приоритет. В качестве false принимается 0, в качестве true не 0. Операции и функции возвращают в качестве true значение. Пример:

<?php //ppr8.php

$b = ('2' & '9');

echo "b=", $b, "<br>";// выдает '0'

$b=(2 && 9);

echo "b=", $b, "<br>";// выдает '1'

$a = (3 | 9);

echo "a=", $a, "<br>";// выдает 11

$c = $a and $b;

echo "c=", $c, "<br>";//выдает '11'

echo "c=", $c = $a && $b,"<br>";// выдает '1'

echo "c=", $c = 3||4, "<br>";//выдает '1'

echo "c=", $c = 3 or 4, "<br>";//1

?>

Строковые операции

Операция "+" используется только для сложения чисел, определяющих значениями строк, и не является операцией конкатенации (слияния строк). Операция конкатенации в PHP – это операция "."(точка).

<?php //ppr9.php

$one = '1';

$two = "2";

echo $one, $two;

print $one+$two; // выведет - 3

echo $one.$two; // выведет - 12

$one = (int)$one; $two = (float)$two;

print $one+$two;//3

$a = "Hello ";

$a = "World!"; // теперь $a содержит "Hello World!"

?>

Результат:

123123

Переменные $one и $two – цифровые строки, однако при выполнении арифметической операции сложения происходит неявное преобразование их значений к числовым и затем сложение. Явное преобразование: $one = (int)$one; $two= (float)$two;.

Рассмотрим особенности операторов сравнения строк. В PHP операнды сравниваются как строки только в том случае, если оба они – строки. В противном случае они сравниваются как числа. При этом любая строка, которую PHP не удается перевести в число, в том числе и пустая, будет восприниматься как 0. Поэтому для сравнения строк не рекомендуется использовать операторы сравнения == и ! =. Пример:

<?php //ppr10.php

$x = 0;

$y = 1;

if ($x == "") echo "<p>x – пустая строка</p>";

if ($y == "") echo "<p>y – пустая строка</p>";

// Выводит:

// x – пустая строка

?>

Данный скрипт сообщает нам, что $x – пустая строка, поскольку пустая строка трактуется как 0. Чтобы избежать преобразования типов, при сравнении строк пользуются оператором эквивалентности, который сравнивает величины и по значению, и по типу:

<?php //ppr11.php

$x = "string";

$y = "string";

$z = "Строка";

if ($x === $z) echo "<p>Строка X равна строке Z</p>";

if ($x === $y) echo "<p>Строка X равна строке Y</p>";

if ($x !== $z) echo "<p>Строка X НЕ равна строке Z</p>";

// Выводит:

// Строка X равна строке Y

// Строка X НЕ равна строке Z

?>

Операторы управления

PHP содержит все основные конструкции языка C++: условные операторы if…else (elseif), switch, четыре вида операторов цикла и др.

Оператор выбора *if* имеет следующий синтаксис:

*if* (boolexp) { */\*операторы, выполняемые при boolexp=true\*/*} *//1*

*else {* */\*операторы, выполняемые при boolexp=false \*/* } *//2*

Если выражение *boolexp* принимает значение true, то выполняется группа операторов 1, если false – группа операторов 2. Допустимо также использование конструкции *if {} else if {}.*

В следующем примере показана возможность отключения PHP, чтобы вывести без интерпретации часть html-документа после ?>.

<?php //ppr12.php

$boolexp = true;

//отключение php

if ($boolexp) : ?>

<img src = "img/A\_A.jpg" >

<?php endif; //включение php

$c = 5;

echo "c=", $c = $c or 4, "<br>";//1

?>

Оператор if() при этом надо закончить оператором endif (аналогично endwhile, endfor). В результате будет выведено изображение и с = 1.

Другой вариант состоит в использовании фигурных скобок вместо двоеточия:

<?php //ppr13.php

$bexp = true;

//отключение php

if ($bexp) { ?>

<img src = "img/A\_A.jpg" >

<?php } //включение php

$c = 5;

echo "c=", $c = $c or 4, "<br>";//1

?>

В PHP существует четыре вида циклов, первые три из них аналогичны соответствующим циклам в С++:

Цикл с предусловием:

*while (*boolexp) { */\**операторы, выполняемые при boolexp=true *\*/* }

*Цикл с постусловием:*

*do* { */\**операторы, выполняемые при boolexp=true *\*/* }

*while*(boolexp);

*for*(exp1; boolexp; exp3){ */\**операторы*\*/* } *//* цикл с параметрами

Здесь по традиции exp1 – начальное выражение, *boolexp* – условие выполнения цикла, exp3 –изменение параметра цикла. Циклы выполняются, пока булевское выражение *boolexp* равно true.

Eще один цикл, упрощающий доступ к массивам:

*foreach*($array as[$key => ] $value){ */\**операторы*\*/* }

При проходе каждого элемента массива в переменную $key помещается индекс данного элемента, а в переменную value – значение элемента. Индекс $key может отсутствовать. Пример:

<?php //ppr14.php

$massiv = array('one','two','three');

foreach($massiv as $value)echo $value;//результат: onetwothree

?>

Как и в случае операторов if и swi tch, оператор for поддерживает синтаксис без фигурных скобок с использованием ключевых слов for и endfor. Пример

<?php

for ($i = О; $i < 5; $i++) :

echo "$i<br />";

endfor;

Аналогично C++ используется оператор выбора варианта switch:

switch(ex) {

caseex1:{*/\*операторы\*/*}

break;

caseexN:{*/\*операторы\*/*}

break;

default: {*/\*операторы\*/*}

}

При совпадении значения ключа ex с одним из значений, указанным в case, выполняется соответствующий вариант, пока не встретится оператор break. Значения exp1, …, expN могут быть константами целого, вещественного или стрингового типа. Оператор break прерывает цикл, в котором находится. Для прерывания одной итерации цикла используется оператор continue. После этих операторов может стоять константа, указывающая сколько циклов прерывается.

<html>

<head></head>

<body>

<p>В прямоугольной матрице переставить столбцы в порядке возрастания суммы их элементов</p>

<?php //ppr15.php

$matrix = array();

$n = 3;

$m = 5;

for($i = 0;$i<$n;$i++)

for($j = 0;$j<$m;$j++){

$matrix[$i][$j] = rand(0,9);

}

for($i = 0;$i<$n;$i++){

for($j = 0;$j<$m;$j++){

echo($matrix[$i][$j]);

echo(" ");

}

echo("<br>");

}

$sum = array();

for($j = 0;$j<$m;$j++){

$sum[$j] = 0;

for($i = 0;$i<$n;$i++){

$sum[$j]+= $matrix[$i][$j];

}

}

echo("<br>");

for($i = 0;$i<$m;$i++){

echo($sum[$i]);

echo(" ");

}

for($i = 0;$i<$m–1;$i++)

for($j = 0;$j<$m–1-$i;$j++){

if($sum[$j]>$sum[$j+1]){

$temp = $sum[$j];

$sum[$j] = $sum[$j+1];

$sum[$j+1] = $temp;

for($k = 0;$k<$n;$k++){

$temp = $matrix[$k][$j];

$matrix[$k][$j] = $matrix[$k][$j+1];

$matrix[$k][$j+1] = $temp;

}

}

}

echo("<br>");

echo("new matrix:");

echo("<br>");

for($i = 0;$i<$n;$i++){

for($j = 0;$j<$m;$j++){

echo($matrix[$i][$j]);

echo(" ");

}

echo("<br>");

}

?>

</body> </html>

### 6.2. Функции

*Определение и вызов функций*

Функции объявляются при помощи ключевого слова function, после которого следует имя функции, параметры функции в круглых скобках и в фигурных скобках записываются операторы, составляющие тело функции:

function fname ($arg\_1, $arg\_2, ..., $arg\_n)

{

echo "Пример \n";

return $val; //возвращаемое значение

}

Необязательный оператор return возвращает значение любого типа, в том числе список и объект. В определении функции не обязательно указывать тип возвращаемого значения.

1. Пример простой функции. Файл function.php

<?php

function getSum()

{

$sum = 10 + 5;

return $sum;

}

echo getSum(); // 15

Функция не принимает ни одного аргумента, а вычисляет сумму и возвращает результат. После этого она вызывается в теле оператора echo.

В РНР функция может вызываться до ее объявления.

Листинг 3. Файл function\_call.php

<?php

echo getSum () ;

function getSum()

{

$sum = 10 + 5;

return $sum; }

<?php //ppr16.php – найти все натуральные, не превосходящие m

//и содержащие хотя бы одну девятку в десятичном представлении

$m=rand(10,150); //случайное значение m

print "m = $m<br>";

for($i = 1;$i<$m;$i++)

{$k = f($i);

if($k! = 0)print "$k<br>";

}

function f($n)

{$l = $n;

do

{

if(($l%10)==9)return $n;}

while(($l/ = 10)! = 0);

return 0;

}

?>

Пример. Вычисление площади треугольника по формуле Герона.

<html>

<head><title>Square</title></head>

<body>

Стороны треугольника:<br>

<?php //ppr17.php

$a = rand(1,100);

$b = rand(1,100);

$c = rand(1,100);

print "$a<br>$b<br>$c<br>";

Proverka($a,$b,$c);

function Proverka($a,$b,$c)

{if(($a+$b <= $c)||($a+$c <= $b)||($b+$c <= $a))

echo "Треугольник не существует!!!!<br>Перезагрузите страницу!<br>";

else{

echo "OK! Треугольник с указанными сторонами существует!!!!<br>";

$s = Geron($a,$b,$c);

echo "Его площадь равна<br> ";

print "$s";

}

}

function Geron($a,$b,$c){

$p = ($a+$b+$c)/2;

return sqrt($p\*($p–$a)\*($p–$b)\*($p–$c));

}

?>

</body> </html>

*Параметры и аргументы функции*

Передадим складываемые числа в качестве аргументов (листинг 6).

Листинг 6. Параметры функции. Файл params.php

<?php

function getSum($left, $right)

{

$sum = $left + $right;

return $sum;

}

echo getSum(10, 5); // 15

Переменные $left и $right, которые задаются в круглых скобках при объявлении, называются параметрами функции. Значения и переменные, передаваемые функции при вызове, называются аргументами функции. В качестве аргументов могут выступать выражения и даже другие функции.

*Типы параметров и возвращаемого значения*

При объявлении функции в PHP7 допускается указывать типы параметров и тип возвращаемого значения. Последний указывается через двоеточие непосредственно перед телом функции (листинг 8).

Листинг 8. Тиnы параметров и возвращаемого значения. Файл types.php

<?php

function getSum(int $fst, int $snd):int

{

return $fst + $snd;

}

echo getSum(2, 2); // 4

echo getSum(2 . 5, 2 . 5); // 4

*Передача параметров по значению и ссылке*

В PHP, как и в С++, аргументы передаются в функцию по значению. При этом создаются копии аргументов, а сами аргументы не изменяются. Если необходимо дать функции возможность модифицировать аргументы, передавать их надо по ссылке &:

function ff(&$strike)

{

$strike .= 'and extra.';//добавление значения к $strike

}

$strike = 'This is a string, ';

ff($strike);

echo $strike;// выводит 'This is a string, and extra.'

При передаче ссылки в функцию передается не значение переменной $strike, а ее адрес. После этого изменение переменной $strike в функции влечет изменение аргумента $strike в вызывающей функцию программе.

Листинг9. Передача арrумента no значению. Файл params\_by\_val.php

<?php

function getSum($var) // аргумент передается по значению

{

$var = $var + 5;

return $var;

}

$newvar = 20;

echo getSum($newvar); // 25

echo "<br />$newvar"; // 20

Если аргумент передается по ссылке, при любом изменении значения параметра происходит изменение переменной-аргумента (листинг 10).

Листинг 10. Передача аргумента по ссылке. Файл paгams\_by\_ref.php

<?php

function ge t Sum (&$ var) //аргуме н т переда е т с я по ссыпке

{

$va r= $var + 5;

return $var·;

$newva r = 20;

}

e cho getSum ($newvar); // выводит 25

echo "<br />$newvar"; // выводит 25

Операция ссылки & в объявлении функции и в присвоении возвращаемого значения позволяет возвратить ссылку:

function &reference()

{ return &ref;

}

$newref =&reference();

Объекты и массивы передаются в функцию по ссылке, поэтому применительно к таким параметрам можно не применять символ амперсанда &.

*Необязательные параметры*

Функция может определить параметры по умолчанию (значение аргументу присваивается в случае, если аргумент не был передан в функцию)

function f($а = "0")

{

return $а;

}

echo f(); // 0.

echo f("1");// 1.

При вызове такой функции без аргумента в качестве аргумента будет взято значение по умолчанию. Значение по умолчанию обязано быть константным выражением, но не переменной. Эти значения должны находиться справа от любых значений не по умолчанию.

Листинг 11. Необязательные параметры. Файл paгams\_default.php

<?php

function getSum ($ left = 10, $right 5){

$sum = $left + $right;

return $sum;

}

echo getSum () ;

echo getSum (5 ) ;

// ВЫВОДИТ 15

*//* выводит 10

echo getSum(5, 0); // выводит 5

*Переменное количество параметров*

Для того чтобы создать функцию, которая принимает переменное количество аргументов, перед последним параметром следует указать многоточие. Внутри функции такой параметр рассматривается как массив, содержащий все дополнительные параметры (листинг 13).

Листинг 13. Переменное число параметров. Файл varargs.php

<?php

function echoList( ... $items)

{

foreach ($items as $v) {

echo "$v<br />\n"; // выводим элемент

1

*//* Отображаем строки одну под другой

echoList('PHP', 'Python', 'Ruby', 'JavaScript');

Оператор … может использоваться не только перед аргументами функций, но и при вызове с массивом. Это позволяет осуществить "развертывание" массива.

Пусть имеется функция tooManyArgs () с большим количеством параметров. Можно поместить значения параметров в массив $args и передать его функции, предварив оператором ... , который развернет элементы массива в соответствующие параметры (листинг l 4 ).

Листинг 11.14. Использование оператора . .. . Файл toomanyargs.php

<?php

function tooManyArgs($fst, $snd, $thd, $fth)

{

echo "Первый параметр: $fst<br />";

echo "Второй параметр: $snd<br />";

echo "Третий параметр: $thd<br />";

echo "Четвертый параметр: $fth<br />";

}

*//* Отображаем строки одну под другой

$items = ['РНР', 'Python', 'RuЬy', 'JavaScript'];

tooManyArgs( ... $items);

Несколько значений можно вернуть из функции с помощью списка и функции list().

function numbers()

{

return array (0, 1, 2);

}

list ($zero, $one, $two) = numbers();

### 6.2.1. Рекурсивные функции

<?php //ppr18.php

// Написать функцию вычисления x^n

function Evaluate($x,$n)

{

if ($n == 1) return $x;

else return Evaluate($x,$n–1)\*$x;

}

echo("x = 5; n = 3; res = ");

echo(Evaluate(5,3));

echo("<BR>");

echo("x = 0.2; n = 3; res = ");

echo(Evaluate(0.2,3));

?>

### 6.2.2. Область действия и время жизни переменных

*Глобальные переменные*

Переменные в функциях имеют локальную область видимости . Это означает, что даже если локальная (внутри функции) и внешняя (вне функции) переменные имеют одинаковые имена, то изменение локальной переменной никак не повлияет на внешнюю переменную (листинг 15).

Любая переменная, определенная внутри функции, ограничена локальной областью функции и уничтожается при выходе из функции.

Листинг 15. Файл vars\_scope.php

<?php

function getSurn ()

{

$var 5;

return $var;

}

$var = 10;

echo getSurn () ; //5 - локальная переменная

echo "<br />$var"; // 10 (внешняя переменная)

Если локальная переменная объявлена как global, то к ней возможен доступ из любой части программы (листинг .16).

Листинг 11 .1 б . Использование глобальной переменной . Файл global.php

<?php

function getSum()

{global $var;

$var 5; //изменяем глобальную переменную

return $var;

}

$var = 10;

echo "$var<br />"; //выводит 10

echo getSum(). '<br />'; //выводит 5(глобальная переменная\_изменена )

### echo "$var<br />"; //выводит 5

Переменные, объявленные вне функций, имеют глобальную область действия. Чтобы глобальная переменная, объявленная вне функции, была доступна внутри функции, ее необходимо внутри этой функции задекларировать как глобальную переменную со словом global.

<?php //ppr19.php

/\* глобальная область \*/

$a = 2;

$b = 3;

function Sum ()

{

global $a, $b; /\* декларируем переменные $a и $b как глобальные \*/

$c = $a + $b; /\* и теперь функция имеет к ним доступ \*/

echo $c;//5

global $myglobal;

$myglobal = 0;

}

echo ($myglobal) ;//неопределенная переменная $myglobal

Sum ();

echo $myglobal; //0

?>

*Статические переменные*

Время жизни локальной переменной – это время работы функции, область действия – локальная область функции. Для удлинения времени жизни локальной переменной ее объявляют статической переменной. Статическая переменная существует только в локальной области функции, но не теряет своего значения после выхода из функции.

function Count ()

{static $counter = 0; /\*не будет обнуляться при каждом обращении к функции \*/

echo $counter;

$counter++; }

Count(); //количество обращений к функции

Для того чтобы локальная переменная сохраняла свое предыдущее значение при новых вызовах функции, ее можно объявить статической при помощи ключевого слова static (листинг 18).

$counter устанавливается в ноль при первом вызове функции, и при последующих вызовах функция "помнит", каким было значение переменной при предыдущих вызовах.

Временем жизни статических и глобальных переменных является время выполнения сценария. То есть если пользователь перезагружает страницу, это приводит к новому выполнению сценария, переменная $counter инициализируется заново.

Cуперглобальные переменные $\_GET[], $\_POST[], $\_REQUEST[], $\_COOKIE[], \_FILES[], $\_SERVER[], $\_ENV[], $\_SESSION доступны постоянно и везде во время выполнения сценария.

Изменяемые (динамические) переменные. Иногда бывает удобно давать переменным изменяемые имена. Такие имена могут изменяться динамически. Например:

$a = "hi";

Изменяемая переменная берет некое значение и обрабатывает его как имя переменной. В приведенном выше примере значение hi может быть использовано как имя переменной, посредством применения двух записанных подряд знаков доллара, т.е.:

$$a = "PHP"; /\* фактически получается что $hi = "PHP" \*/

print "$a ${$a}"; /\* получаем на выходе: hi PHP \*/

print "$a $hi"; /\* или так, что одно и то-же \*/

### 6.2.3. Внешние библиотечные функции

В этом разделе описываются некоторые из часто используемых функций. Более обширный список доступен на веб-сайте PHP.net.

Функции для работы с массивами. Функции count(), sizeof() – используются для подсчета числа элементов в массиве; sort() – сортировка элементов; shuffle() – случайное перемешивание элементов; array\_slice($name, offset, length) – применяется для извлечения части массива, где $name имя массива; offset – начальная позиция; length указывает число вырезае­мых элементов. Функция array\_merge($name1, $name2) – используется для объединения или слияния массивов. Пример:

<?php //ppr20.php

$massiv = array( 'В пятницу вечером после работы

мама свои начинает заботы',

'папа залег на диван, как медведь',

'многое должен ребенок суметь');

foreach($massiv as $value) echo $value;//Вывод

sort($massiv);//Сортировка

foreach($massiv as $key=>$value)

echo ("<li>$key=>$value</li>");//

rsort($massiv);// Сортировка в обратном порядке

for($i = 0;$i<count($massiv);$i++){

echo("<br>$massiv[$i]");

}

shuffle($massiv); // Перемешивание

for($i = 0;$i<count($massiv);$i++){

echo("<br>$i–$massiv[$i]");

}

$massiv1 = array\_slice($massiv,2,3); //вырезать три члена, начиная со второго

// и присвоить их massiv1

print\_r($massiv1);

?>

*Вложенные функции*

Язык РНР позволяет объявлять функции внутри другой функции.

Вложенная функция не может использоваться до тех пор, пока не будет осуществлен вызов основной функции, который произведет объявление вложенной (листинг 22).

Листинг 11.22. Объявление вложенной функции. Файл nested.php

<?php //Объявление внешней и вложенной функций

function outte r()

{

function inner (){

return "Hello, world!";

}

}

*11* Вызываем функцию outter ( ) , чт обы объявить фун кцию inner ()

outter();

*11* Функция inner () не может быть вызвана до тех пор,

*11* пока не будет вы звана функция o utter( ) ;

ech o inner( ) ;

Динамическое имя функции

По аналогии с переменными, имя функции может быть динамическим и храниться в строковой переменной - передача такой переменной оператором круглых скобок (с параметрами, если они требуются) приводит к вызову функции. В листинге 23демонстрируется вызов функций hello () и bуе ( ) по случайному закону.

Листинг 23. Использование динамических имен функций. Файл dynamic.php

<?php

*//* О бъявление функций

function hello ( )

[

return 'Hello ! ';

function Ьуе ()

r e turn ' Вуе ! ' ;

*//* Случайный выбор функции

$var = rand(O, 1) ? 'hello'

*//* Вызов функции

$var();

'Ьуе';

*Анонимные функции*

Анонимные функции - это функции без имени. В листинге 24 приводится пример объявления такой функции.

Листинг 24. Анонимная функция. Файл anonim.php

<?php

$echoList function ( ... $str)

{

foreach ($str as $v) (

echo "$v<br />\n";

1;

*//* Вызов функции

$echoList ( 'РНР', 'Python', 'Ruby', 'JavaScript');

Анонимные функции допускается передавать в качестве аргументов другим функциям. Рассмотрим эту возможность на примере сортировки элементов массива. Массив строк или чисел можно отсортировать при помощи стандартной функции

sort () (листинг25).

Листинг 25. Анонимная функция. Файл sort.php

<?php

$arr = ['РНР', 'Python', 'RuЬy', 'JavaScript');

sort($arr);

echo '<pre>';

print\_r($arr); // JavaScript, РНР, Python, RuЬy

Однако если мы имеем дело с массивом объектов, корректно отсортировать его функцией sort () довольно сложно, т. к. разница между объектами может определяться по критериям, заложенным внутри объекта. Например, пусть в качестве элементов массива выступает набор точек класса Point, который содержит координаты точки двумерного пространства (листинг 26).

Листинг 26. класс точки Point. Файл point.php

<?php

class Point

puЫic $х;

puЬlic $у;

В качестве критерия для сортировки точек может быть выбрано расстояние от начала координат (О, О), которое вычисляется как корень квадратный из суммы квадратов значений по осям абсцисс и ординат: sqrt ($х \*\* 2 + $у \*\* 2). Для того чтобы отсортировать значения массива таких объектов, потребуется сравнить их расстояния. В этом случае удобнее воспользоваться функцией usort (), которая в качестве второго параметра принимает функцию

bооl usort(array &$.иrау, callaЫe $v&lue\_cC111p4re\_Eшlc)

В качестве такой функции может выступать анонимная функция, определенная непосредственно в списке аргументов (листинг 11.27). Функция должна принимать два сравниваемых значения и возвращать значение меньше нуля, О или значение больше нуля, если первый параметр окажется меньше, равным или больше второго .

Листинг 27. Сортировка массива точек. Файл sort\_anonim.php

<?php

require once ( 'point. php') ;

$fst = new Point;

$fst->x = 12;

$fst->y = 5;

$snd = new Point;

$snd->x = 1;

$snd->y = 1;

$thd = new Point;

$thd->x = 4;

$thd->y = 10;

$arr = [$fst, $snd, $thd];

usort($arr, function($a, $Ь) {

)) ;

$distance\_a = sqrt($a->x \*\* 2 + $а->у \*\* 2);

$distance\_b = sqrt($b->x \*\* 2 + $Ь->у \*\* 2);

return $distance а <=> $distance\_b;

echo '<pre>' ;

print\_r($arr);

Как видно из листинга, внутри анонимной функции для каждого из объектов вычисляется расстояние $distance\_a и $distance\_ь, которые затем сравниваются друг с другом при помощи оператора <=>. Результат выполнения скрипта выглядит следующим образом:

Array

[О] => Point Object

[х] => 1

[у] => 1

[1) => Point Object

[х] => 4

[у] => 10

[2] => Point Object

[х] => 12

[у] => 5

*Замыкания*

Замыкание - это функция, которая запоминает состояние окружения в момент своего создания . Даже если состояние затем изменяется, замыкание содержит первоначальное состояние. Замыкание в РНР применимо только к анонимным функциям. В отличие от других языков программирования, вроде JavaScript, замыкания не действуют автоматически. Для активизации необходимо использовать ключевое слово use, а за ним в скобках можно указать переменные, которые должны войти в замыкание (листинг l l .28).

Листинг 11.28. Замыкания. Файл closure.php

<?php

$message ='Работа не может быть nродолжена из-за оu.мбок:<Ьr /> ';

$check function(array $errors) use ($message)

);

if (isset($errors) && count($errors) > 0) {

echo $message;

foreach($errors as $error)

echo "$error<br />";

*158*

$check ( [] ) ;

*11* ...

$erorrs[] = 'Заполните имя пользователя';

$check($erorrs);

*11* ...

$message = 'Список требований'; //Уже не изменить

$erorrs = ['РНР', 'PostgreSQL', 'Redis'];

$check($erorrs);

В листинге 28 создается анонимная функция-замыкание, которая помещается в переменную $check, при помощи ключевого слова use замыкание захватывает переменную $message, которую использует в своей работе. Попытка изменить значение переменной позже не приводит к результату. Замыкание "помнит" состояние переменной в момент своего создания. Результатом выполнен}Щ скрипта будут следующие строки

Работа не может быть продолжена из-за ошибок:

Заполните имя пользователя

Работа не может быть продолжена из-за ошибок:

РНР

PostgreSQL

Redis

Основное назначение замыканий - замена глобальных переменных. В отличие от глобальных переменных, вы можете передать внутрь функции значение, но уже не сможете изменить переменную, переданную через механизм замыкания. Самое главное - никакие изменения глобальной переменной в других частях программы не смогут повлиять на значение, переданное через замыкание.

Функции для работы со строками. Приведем несколько распространенных функций: strlen($st) – возвращает длину строки; ltrim($st) – удаляет разделители типа пробела или табуляции в начале строки; rtrim($st) – удаляет разделители в конце строки; trim($st) – вырезает пробелы в начале и в конце строки. Пример:

<?php //ppr21.php

//Удаление лишних пробелов по-левому боку текста:<br />

$str = " –Текст с лишними пробелами по бокам.- “;

$str = ltrim($str); //удаление пробелов слева<br />

echo "$str", strlen($str)," <br />";

$str = rtrim($str); //Удаление пробелов справа<br />

echo "$str"," <br />";

$str = " Еще Текст с пробелами по бокам ". $str=trim($str);//Удаление пробелов <br />

echo "$strike";

?>

Вывод:

*-текст с лишними пробелами по бокам.- 89 ;*

*-текст с лишними пробелами по бокам.- ;*

*Еще Текст с лишними пробелами по бокам*

*-Текст с лишними пробелами по бокам.-*

Функция strpos($where, $what) находит в строке $where подстроку $what и возвращает ее позицию, иначе возвращает false. Функция substr ($st, $start [, $length]) – возвращает часть строки $st, начиная с позиции $start и длиной $length. Если $length не задана, то возвраща­ется вся подстрока до конца строки. Функция strstr($st, $ndl) –возвращает часть строки $st от первого вхождения $ndle до конца. Иначе возвращает FALSE. Для по­иска подстроки без учета регистра используется функция stristr().

<?php //ppr22.php

echo strpos("Hello","el"),"<br/>"; // Выводит 1

echo strstr("Hello","el"),"<br/>";

// Чтобы избежать проблем с определением типов используйте

//Операторы тождественных сравнений (===) (!==)

if (strpos("Norway","rwa") !== false)

echo "Строка rwa есть в Norway","<br/>";

//Пример использования substr()

echo $rest = substr("abcdef", 1),"<br/>"; //возвращает "bcdef"

echo $rest = substr("abcdef", 0, 4),"<br/>";//возвращает "abcd"

$strike = "Programmer";

echo substr($str,0,2),"<br/>"; // Выводит Pr

echo substr($str,-3,3),"<br/>"; // Выводит mer

//Пример strstr()

$email = 'user@example.com';

$domain = strstr($email, '@');

print "$domain, <br/>"; // печатает @example.com

?>

Вывод:

*1*

*Ello*

*Строка rwa есть в Norway*

*Bcdef*

*Abcd*

*Pr*

*Mer*

*@example.com,*

Функция strcmp(string $str1, string $str2) сравнивает две строки посимвольно и возвращает: 0, если строки совпадают; –1, если строка $str1 лексикографически меньше $str2; и 1, если $str1 больше $str2. Для функции strcasecmp($str1, $str2) регистр символов не учитывается.

Функция int strspn (string st1, string st2) возвращает длину первого сегмента строки st1, содержащего символы, присутствующие в st2. Следующий фрагмент показывает, как функция strspn( ) используется для проверки пароля. Здесь же сравниваются две одинаковые строки:

<? //ppr23.php

$string1 = "butter";

$string2 = "Butter";

if ((strcmp($string1, $string2)) == 0)

print "Strings are equivalent!";

else

print "Strings are not equivalent!";

if ((strcasecmp($string1, $string2)) == 0)

print "Строки совпадают с точностью до регистра!";

$password = "12345";

if (strspn($password, "1234567890") ==strlen($password))

print "Password cannot consist solely of numbers!";

?>

Вывод:

Strings are not equivalent!

Строки совпадают с точностью до регистра!

Password cannot consist solely of numbers!

str\_replace(string $from, string $to, string $st) – заменяет в строке $st все вхождения подстроки $from (с учетом регистра) на $to и возвращает результат. Исходная строка st меняется. В следующем примере символы перевода строки замещаются на их HTML-эквивалент – тэг <br>: $st=str\_replace("\n","<br>\n",$str). Функция string WordWrap(string $str, int $width = 75, string $break = "\n") разбивает блок текста $str на несколько строк, завершаемых символами $break так, чтобы на строке было не более $width букв. Возвращается получившаяся строка с символами перевода строки, заданными в $break.

Пример:

<?php //ppr24.php

$strike = "Это текст письма, которое нужно отправить адресату ";

// Разбиваем текст по 30 символов

$strike = WordWrap ($strike, 30, "<br>");

echo $strike;

?>

Вывод.

*Это текст письма, которое нужно отправить адресату*

Функция string stripcslashes (string strike) – возвращает строку с вырезанными обратными слэшами \n, \r; strip\_tags (string $str [, string $allow\_tags]) – удаляет из строки str все тэги и возвращает результат. В параметре $allow\_tags перечисляются вплотную друг к другу тэги, которые не следует удалять. Примеры:

<? //ppr25.php

$str = "<b>жирный текст</b>";

echo "$str,<br/>";

$str = strip\_tags($str);

echo "НЕ $str<br/>";

//Удаление всех тэгов, кроме <b> и <i>:

$str = "<h1>большой текст</h1> <b><i>жирный текст</i></b>";

echo "$str <br/>";

$str = strip\_tags($str,"<b><i>");

echo "$str,<br/>";

$strip = strip\_tags ($str); // Удаляет все теги из строки

echo "$strip<br/>";

?>

Следующие функции предназначены для быстрого URL-кодирования и декодирования. Функция UrlEncode(string $str) кодирует строку и возвращает результат. Например:

echo "<a href = /script.php?param = ".UrlEncode($UserData);

UrlDecode(string $st) производит URL-декодирование строки. Функция HtmlSpecialChars(string $str) обычно используется в комбинации с echo. Основное ее назначение – гарантировать, что в выводимой строке ни один участок не будет воспринят как тэг. Заменяет в строке некоторые символы (такие как амперсант, кавычки и знаки "больше" и "меньше") на их HTML-эквиваленты так, чтобы они выглядели на странице "самими собой". Применяется для формирования параметра value в различных элементах формы, или же при выводе сообщения в гостевой книге.

Функция strtoupper(string) – преобразует строку в верхний регистр; strtolower(string) – преобразует строку в нижний регистр; strrev(string) – возвращает строку string в обратном порядке.

Примеры:

<?php//ppr26.php

$string = "Hello World";

$another\_string = "Welcome to PHP";

echo strlen($string);

echo strtoupper($another\_string);

echo strrev($another\_string);

echo strpbrk($string, "W");

?>

Вывод:

*11*

*WELCOME TO PHP*

*PHP ot emocleW*

*World*

Функции форматных преобразований строк. Язык PHP поддерживает ряд форматных функций вывода. Функция sprintf ($format [, mixed args]) – возвращает строку, созданную из аргументов в соответствии со строкой $format. Последняя включает команды форматирования, предваренные символом %. Пример:

<? // ppr27.php Использование sprintf()

$num = 5;

$location = "tree";

$format = "There are %d monkeys in the %s";

printf($format,$num,$location); // выведет: "There are 5 //monkeys in the tree".

//sprintf(): целые числа с заполнением нулями

$year = 3;

$month = 10;

$day = 11;

$dat = sprintf(" %03d-%03d-%02d ", $year, $month, $day);

echo $dat;

//sprintf(): Форматирование валюты

$money1 = 68.75;

$money2 = 54.35;

$money = $money1 + $money2;

echo $money;// will output "123.1";

$formatted = sprintf(" %01.2f", $money);

echo $formatted; //выдаст "123.10"

?>

Вывод:

*There are 5 monkeys in the tree 003-010-11 123.1 123.10*

Функция printf($format [, mixed args, ...]) выполняет то же, что и sprintf(), только направляет результирующую строку в браузер пользователя. Функция sscanf (str, format [,var1]) – читает из строки str и интерпретирует ее в соответствии с форматом. Символ в строке формата совпадает с символом в строке ввода.

<? // ppr28.php Использование sscanf()

$serial = sscanf("SN/23501","SN/%s",&$ser);

// получение серийного номера и даты изготовления

echo $serial,"<br>";

$mandate = "January 01 2011";

list($month, $day, $year) = sscanf($mandate,"%s %d %d");

echo "Item $ser was manufactured on: $year-" .substr($month,0,3)."-$day\n";

// функция возвращает количество присвоенных значений.

?>

Вывод:

*1*

*Item 23501 was manufactured on: 2011-Jan-1*

Использование форматных функций ограничено из-за их низкой скорости: для разбора строк чаще привлекают регулярные выражения или функцию array explode (разделитель, $str[, int n]), которая делит строку на элементы и возвращает в виде массива. Количество элементов может ограничиваться параметром n. Функция string implode (разделитель, array фрагменты) – объединяет массив в строку. Пример:

<? // ppr29.php

$info = "Minsk | baseball | indians";

$user = explode("|", $info);

//$user[0] = "Minsk";$user[1] = "baseball";$user[2] = "Indians";

print\_r($user);

$cities=array("Colum","Youngstown","Cleveland","Cincinnati");

$city\_string = implode("l", $cities);

//$city\_string = "Colum1Youngstown1Cleveland1Cincinnati";

echo "<br>", $city\_string;

?>

Вывод:

*Array ( [0] => Minsk [1] => baseball [2] => indians )*

*ColumlYoungstownlClevelandlCincinnati*

Функция string strtok ($arg1, string $token) – выполняет лексемизацию строки, в результате которой строка arg1делится на слова (лексемы/tokens), где каждое слово отделено символом из $token. В примере строка "This is an example string" лексемизируeтся на отдельные слова с применением пробела как разделителя.

<? //ppr30.php

$string = "This is\tan example\nstring";

/\* Использовать символы tab и newline как лексемизирующие символы \*/

$tok = strtok($string," \n\t");

while ($tok) {

echo "Word=$tok<br>";

$tok = strtok(" \n\t");

}

?>

Вывод:

*Word = This*

*Word = is*

*Word = an*

*Word = example*

*Word = string*

В параметр token можно поместить несколько лексем.

void parse\_str (string $strike ) – разбирает strike так, как если бы она была строкой запроса, переданной через URL, и устанавливает переменные.

<? // ppr31.php Использование parse\_str()

$strike = "first = value &second[] = this+works &second[] = another";

parse\_str($strike);

echo $first; /\* печатает "value" \*/

echo $second[0]; /\* печатает "this works" \*/

echo $second[1]; /\* печатает "another" \*/

?>

Вывод:

value this works another

getHostByName(domain\_name) – преобразует переданное имя домена в IP адрес в формате nnn.nnn.nnn.nnn.

getHostByAddr(ip\_address) – преобразует данный IP-адрес в формате nnn.nnn.nnn.nnn в полное имя домена.

Пример: определяем в какой стране живет посетитель:

<? //ppr32.php

$host = gethostbyaddr($\_SERVER['REMOTE\_ADDR']);

$countrys = array( ru => Россия, ua => Украина, by => Беларусь );

$array = array\_reverse(explode(".",$host));

if(!empty($countrys[$array[0]]))

echo "Ваша страна: ".$countrys[$array[0]];

else echo "Откуда Вы пришли – я не знаю :(";

$ip = gethostbyname('localhost');

echo "<br>",$ip;

?>

Вывод:

*Откуда Вы пришли – я не знаю :(*

*127.0.0.1*

Преобразование строк и файлов к формату HTML. Функция nl2br(string $str) заменяет все символы новой строки (\n) эквивалентными конструкциями HTML <br>. Функция string htmlentities (string $str ) преобразует символы в эквивалентные конструкции HTML. Функция string htmlspecialchars (string $strike ) преобразует следующие символы: & – в &amp; " " – в &quot; < – в &lt; > – в &gt. В частности, эта функция позволяет предотвратить ввод пользователями опасных символов в форумах. Следующий пример демонстрирует удаление потенциально опасных символов функцией htmlspeclalchars( ):

<?php //ppr33.php

$text= " Мишка косолапый

По лесу идет ";

print $text;

// Преобразовать символы новой строки "\n" в <br>

$htinl = nl2br($text);

print $htinl;

$input = "The cookbook, entitled Cafe Francaise' costs < $42.25.";

echo "input:", $input;

$conv = htmlentities($input);

// $conv = "The cookbook, entitled 'Caf&egrave;

// Frac&ccediliaise' costs &lt; 42.25.";

echo "<br> conv:",$conv;

$strike = "A 'quote' is <b>bold</b>";

echo "<br>strike:", $strike,"<br>";

// Outputs: A 'quote' is &lt;b&gt;bold&lt;/b&gt;

echo htmlentities($strike),"<br>";

// Outputs: A &#039;quote&#039; is &lt;b&gt;bold&lt;/b&gt;

echo htmlentities($strike, ENT\_QUOTES);

//Функция htmlspecialchars( ) преобразует следующие символы:

//& преобразуется в &amp; " " в &quot; < преобразуется в &lt; > в &gt;.

// удаление потенциально опасных символов функцией htmlspeclalchars( ):

$input = "I just can't get of PHP & those fabulous cooking recipes!";

echo "<br>input:", $input;

$conv = htmlspecialchars($input);

// $conv\_input = "I just can't &lt;&lt;enough&gt;&gt; of PHP &amp those fabulous cooking recipes!"

echo "<br> conv:",$conv;

?>

Если функция htmlspecialchars( ) используется в сочетании с nl2br( ), то последнюю следует вызывать после htmlspecialchars( ). В противном случае конструкции <br>, сгенерированные при вызове nl2br( ), преобразуются в видимые символы.

Преобразование HTML в простой текст

Функция string strip\_tags (string строка [, string разрешенные\_тeги]), удаляет из строки все теги HTML и РНР. Используется для удаления опасных символов из сообщений на форумах или в гостевых книгах.

Функция get\_meta\_tags (string имя\_файла/URL [, int включение\_пути]) предназначена для поиска в файле HTML тегов МЕТА. Функция ищет в заголовке документа теги, начинающиеся словом МЕТА, и сохраняет имена тегов и их содержимое в ассоциативном массиве.

Пример:

<html><head>

<title>ppr34.PHP Recipes</title>

<meta name = "keywords" content = " PHP, code, recipes, chef, programming, web">

<meta name = "description" content = "PHP Recipes provides savvy readers HP programming and gourmet cuisine!">

<meta name = "author" content = "WJ ">

</head><body>

*// Переменная $meta\_tags содержит массив со следующей информацией:*

*// $meta\_tags["keywords"] = "PHP. food. code, recipes, chef, programming. Web":*

*// $meta\_tags["description"] = "PHP Recipes provides savvy readers in* PHP

*//programming and gourmet cuisine"; $meta\_tags["author"] = "WJ ";*

Преобразование строки к верхнему и нижнему регистру в РНР вы­полняют функции: strtolower(); strtoupper( ); ucfirst( ); ucwords( ). Функция string strtolower(string $str) преобразует все алфавитные символы строки к нижнему регистру.

**Хэш-функции.** Рассмотрим использование хэш-функций. Функция md5(string $str) возвращает хэш-код строки $str, основанный на алгоритме "MD5 Message-Digest Algorithm". Хэш-код – это строка, практически уникальная для каждой из строк $str. Вероятность того, что две разные строки, дадут одинаковый хэш-код, близка к нулю. Если длина строки $st может достигать нескольких тысяч символов, то ее MD5-код занимает максимум 32 символа. Хэш-код и алгоритм MD5 используются, например, для проверки паролей на истинность. В файле паролей можно хранить не сами пароли, а их хэш-коды. При попытке пользователя войти в систему вычисляют хэш-код введенного пароля и сравнивают его с записанным в файле.

Пример использования алгоритма хеширования MD5:

<?php //ppr35.php

$pass\_a = "MySecret";

$pass\_b = "MySecret";

//Выводим хеш-код строки MySecret($pass\_a) – исходный пароль

echo "<b>Хеш-код исходного пароля '$pass\_a':</b><b style=\"color:green\">".md5($pass\_a)."</b><br>";

//Выводим хеш-код строки MySecret($pass\_b) – верифицируемый пароль

echo "<b>Хеш-код верифицируемого пароля '$pass\_b':</b><b style=\"color:green\">".md5($pass\_b)."</b><br>";

// Сравниваем хеш-коды MD5 исходного и верифицируемого пароля

echo "<h3>Проверяем истинность введенного пароля:</h3>";

if(md5($pass\_a)===md5($pass\_b))

echo "<h3 style=\"color:green\">Пароль верный! (Хеш-коды совпадают)</h3>";

else echo "<h3 style=\"color:red\">Пароль неверный! (Хеш-коды не совпадают)</h3>"

// выводит: Пароль верный! (Хеш-коды совпадают)

// Попробуйте изменить значение строки $pass\_b :

?>

Функция crc32($st) вычисляет 32-битную контрольную сумму строки $st и возвращает 32-битное целое число. Эта функция работает быстрее md5(), но выдает менее надежные хэш-коды.

Установка локальных настроек. Локалью называют совокупность таких настроек, как формат даты и времени, язык, кодировка. Для установки локали используется функция SetLocale($category, $locale).

Регулярные выражения. Базовым средством для поиска данных по шаблону являются регулярные выражения – последовательность симво­лов, применяемых для поиска искомого текста. Простейшее регулярное выражение совпадает с одним или несколькими символами строки в наклонных скобках, например, /Jav/. Квадратные скобки [ ] означают “любой символ из перечисленных в скобках”, например [123abc], и могут содержать интервалы. Конструкция [a-zA-Z] совпадает с любым символом, входящим в интервалы (a-z и A-Z), конструкция [^a-zA-Z] совпадает с любым символом, не входящим в указанные интервалы. Символ точка означает <любой символ>. Например, выражение р.р совпадает с символом р, за которым следует произвольный символ, после чего опять следует символ р. Специальные символы представляет собой алфавитный символ с префиксом \ – признаком особой интерпретации следующего символа: \d – десятичная цифра, \D – любой символ кроме десятичной цифры; \w – алфавитно-цифровой символ,\W – не алфавитно-цифровой;\s – пробельный символ, \S – не пробельный символ; часто используются символы \n ,\r , \t ,\\. Символ \b совпадает с границами слов: /sa\b/, символ \В не совпадает с границей слова: /sa\B/.

Символы +, \* и {...}, обозначающие количество повторений отдельного символа или конструкции, заключенной в квадратные скобки, называются квантификаторами. Рассмотрим несколько примеров.

Если записать регулярное выражение /stud/ как /stud/i, поиск осуществляется без учета регистра. Выражение /^stud/i означает, что stud должно находиться в начале слова (student), а /stud$/ – в конце (restud). Регулярное выражение /^$/i соответствует пустой строке. Шаблон /stu+/ совпадает с последовательностью stu, за которой могут следовать дополнительные символы u. Аналогично шаблон /st{2,4}/ совпадает с символом s, за которым следуют от 2 до 4 экземпляров символа t. Выражение /[\d]+/ используется для поиска цифровой подстроки. Шаблон /<([\w]+)>/ совпадает с конструкциями, заключенными в угловые скобки, например тегами HTML.

Подстроки в регулярных выражениях можно группировать при помощи круглых скобок: /домен – (by|ru|uk|com)/ соответствует строке домен – by или другой. Круглые скобки ( ) означают логическую группу выражений, которая может повторяться (+).

В шаблонах может использоваться оператор | (или), который проверяет совпадение одной из нескольких альтернатив.

В РНР существует пять функций поиска по шаблону с использова­нием Perl-совместимых регулярных выражений: preg\_match(); preg\_match\_all( ); preg\_replace( ); preg\_split( ); preg\_grep( ).

preg\_match(pattern, $str, [regs]) – ищет в строке $str соответствия с регулярным выражением pattern , и сохраняет их в массиве regs (если указано). Пример:

<?php //ppr36.php

$sub = "abcdef";

$pattern = '/^def/'; //ищем def в начале строки

preg\_match($pattern, $sub, $matches);

print\_r($matches);

echo"<br/>";

$pattern = '/def$/';// ищем def в конце строки

preg\_match($pattern, $sub, $matches);

print\_r($matches);

?>

Вывод:

*Array()   
Array ( [0] => def )*

Функция preg\_replace (mixed шаблон, mixed замена, mixed строка)

может использовать регулярные выражения в параметрах шаблон и замена. Параметры шаблон и замена могут представлять собой масивы. Функция preg\_replace( ) перебирает элементы обоих массивов и выполняет замену по мере их нахождения.

<?php //ppr36.php

$strike = "Mersedes is а good car ever created!";

// Выполнить поиск слова без учета регистра :

if (preg\_match("/mers/i", $strike, $match)! = 0)

{

print\_r($match);

}

$strike = "регулярное выражение"; // просто строка

$preg = preg\_replace("/р.+е/i", "<i>[вырезано]</i>", $strike);

echo "<br/>".$preg;

echo"<br/>";

$string = 'April 15, 2010';

$pattern = '/(\w+) (\d+), (\d+)/i';

$replacement = '${1}1,$3';

echo preg\_replace($pattern, $replacement, $string);

?>

Результат:

*Array([0]=>Mers)   
[вырезано]  
April1,2010*

Функция int preg\_match\_all (string шаблон, string строка, array совпадения) находит все совпадения шаблона в заданной строке. Следующий пример показывает, как найти текст, заключенный между тегами HTML:

<?php//ppr37.php

preg\_match\_all("/\(? (\d{3})? \)? (?(1) [\-\s] ) \d{3}-\d{4}/x", "Call 555-1212 or 1-800-555-1212", $phones);

print\_r ($phones);

echo "<br/>";

$html = "<b>bold text</b><a href = howdy.html>click me</a>";

preg\_match\_all("/(<([\w]+)[^>]\*>)(.\*?)(<\/\\2>)/", $html, $matches, PREG\_SET\_ORDER);

foreach ($matches as $val) {

echo "matched: " . $val[0] . "\n";

echo "part 1: " . $val[1] . "\n";

echo "part 2: " . $val[2] . "\n";

echo "part 3: " . $val[3] . "\n";

echo "part 4: " . $val[4] . "\n\n";

}

echo "<br/>";

$userinfo = "Name: <b>Tusik Valery</b> <br> Title: <b>PHP Тeacher</b>";

preg\_match\_all ("/<b>(.\*)<\/b>/U", $userinfo, $pat\_array);

print\_r ($pat\_array);

print "<br/>".$pat\_array[0][0]." <br> ".$pat\_array[0][1]."\n";

?>

Результат:

*Array ( [0] => Array ( [0] => 555-1212 [1] => 800-555-1212 ) [1] => Array ( [0] => [1] => 800 ) )   
matched: bold text part 1: part 2: b part 3: bold text part 4:*

*matched:* [*click me*](http://localhost/my/howdy.html) *part 1:* [*part 2: apart3:click me part4:*](http://localhost/my/howdy.html)  *Array ( [0] => Array ( [0] => Tusik Valery [1] => PHP Тeacher ) [1] => Array ( [0] => TusikValery[1] => PHPТeacher))   
TusikValery   
PHP Тeacher*

Функция array preg\_split (string шаблон, string $strike) разбивает строку $strike в массив посредством регулярного выражения. Пример:

<?php//ppr38

$inf = "+Tusik+++VS+++++@gmail.com+++++bsu.by";

$fields = preg\_split("/[\+]{1,}/", $inf);

print\_r ($fields);

$i = 0;

while ($i < sizeof($fields)){

print $fields[$i]. "<br/>";

echo $i++."<br/>";

}

?>

Результат:

Array([0] => [1] => *Tusik* [2] => VS[3] => @gmail.com[4] => bsu.by)

0

*Tusik*

1

VS

2

@gmail.com

3

bsu.by

4

Функция array preg\_grep (string $pattern, array $mas) перебирает все элементы заданного массива и возвращает в виде массива все элементы, в которых совпадает заданное регулярное выражение. Пример использования функции для поиска в массиве слов, начинающихся на р:

<?php//ppr39

$foods = array ("pasta", "steak", "fish", "potatoes");

// Поиск элементов, начинающихся с символа "р".

// за которым следует один или несколько символов

$pfoods = preg\_grep("/p(\w+)/", $foods);

print\_r( $pfoods);

echo "<br/>";

$i = 0;

while ($i < sizeof($foods)) {

print $pfoods[$i]. "<br>";

$i++;}

?>

Результат:

Array([0] => pasta[3] => potatoes)

pasta

Potatoes

Регулярные выражения – мощный инструмент для поиска таких данных, как адреса электронной почты или поиск URL. Но в то же время регулярные выражения работают медленно. Всегда следует искать более "легкую" замену регулярным выражениям, поскольку скорость выполнения скрипта в таких случаях резко возрастает.

Функции даты и времени. PHP включает следующие функции даты и времени: date(format) – возвращает текущее время сервера; checkdate(month, day, year) – проверяет заданную дату; time() – возвращает текущее время сервера, измеренное в секундах начиная с 1 января 1970 г. Пример:

<?php//ppr40

echo"<span style = 'font:10pt arial'>Today is date('lFjY') </span>";

echo "<br/>";

echo "<span style = 'font:10pt arial'>The current time is: date('g:i:s a')</span>";

?>

В приведенном примере время выводится с помощью параметров формата времени g, i, s, и a. Двоеточия и пробелы также вставляют для разделения часов, минут, секунд и признаков am/pm.

Математические функции

* abs() – модуль числа, sqrt() – квадратный корень;
* acos() – arccos, asin() – arcsin, atan() – arctan;
* cos() – cos, sin() – sin, tan() – тaнгенс, exp() – txp;
* floor() – округляет дробь в меньшую сторону, ceil() – в большую сторону; round() – округляет число типа float;
* fmod() – возвращает дробный остаток от деления;
* log10() – десятичный логарифм, log() – натуральный логарифм;
* pi() – возвращает число Пи;
* pow() – степенное выражение;
* rand() – генерирует случайное число, srand() – изменяет начальное число генератора псевдослучайных чисел.

### 6.3. Объектно-ориентированное программирование

### 6.3.1.Основные понятия

Объектно-ориентированное программирование (ООП) – методология программирования, основанная на представлении программ в виде совокупности объектов, каждый из которых является экземпляром конкретного класса. Объект – обладающий именем набор данных (полей объекта), физически находящихся в памяти компьютера, и методов, имеющих доступ к ним и выполняющих операции над ними. Имя объекта используется для доступа к полям данных и методам, составляющим объект. В классе дается обобщенное описание некоторого набора реально существующих объектов. Объектно-ориентированное программирование основано на принципах: *абстрагирования данных; инкапсуляции; наследования; полиморфизма; позднего связывания.*

*Инкапсуляция (encapsulation)* – принцип, объединяющий данные и код, манипулирующий этими данными, а также защищающий данные от прямого внешнего доступа. Доступ к данным класса возможен только посредством методов этого же класса.

*Наследование (inheritance)* – это процесс, посредством которого объект может приобретать свойства объекта-предка и добавлять к ним свойства и методы, характерные только для него. Наследование бывает линейное – класс имеет один и только один суперкласс (предок) и множественное. В PHP не используется множественное наследование – класс не может иметь несколько предков.

*Полиморфизм (polymorphism)* – механизм, использующий одно и то же имя метода для решения похожих, но несколько отличающихся задач. Целью полиморфизма, применительно к ООП, является использование одного имени для задания общих для объектов класса действий.

Механизм *позднего связывания* при выполнении программы определяет принадлежность объекта конкретному классу и осуществляет вызов метода, относящегося к классу, объект которого был использован.

### 6.3.2. Классы и Объекты

Модель объектов PHP похожа на модель объектов Java. Класс представляет объединение переменных и методов – функций, работающих с этими переменными. Переменные класса определяются со спецификатором доступа private, public или protected. Методы определяются в классе с ключевым словом function и спецификатором доступа. Пример:

<?php//ppr41

class Mycart

{ private $items = 0; //переменные класса

public function add ($n) // Добавить $n артикулов

{

$this -> items + = $n;

}

public function remove ( $n) // Изъять $n артикулов

{

if ($this -> items > $n) {

$this->items - = $n;

return true; }

else { return false; }

}

public function show()

{

echo '$items =', $this -> items ;

}

}//конец класса

$instance = new Mycart(); //ссылка на объект

$instance->add(5);

$instance->remove(2);

$instance->show();

echo $instance->items ;//ошибка, нет доступа к items

?>

Вывод:

*$items=3*

*Fatal error:*

Здесь $instance представляет ссылку на объект класса, который создается операцией new. Каждый объект получает числовой идентификатор (handler), который используется при обращении к объекту. Чтобы иметь возможность доступа к переменным и функциям внутри класса, можно использовать псевдопеременную $this, которая является ссылкой на текущий объект.

Спецификаторы доступа public, protected и private, позволяют указывать степень доступа к свойствам и методам класса. К общедоступным (public) свойствам и методам можно получить доступ без ограничений. Защищенные (protected) элементы класса доступны внутри класса, в котором они объявлены, и в наследуемых от него классах. Частные (private) элементы доступны только методам в классе, в котором они объявлены.

Если не указывать ни один из спецификаторов, то по умолчанию элемент будет иметь уровень доступа public.

Конструкторы и деструкторы

Для инициализации переменных класса используется функция, которая вызывается автоматически при создании объекта класса с помощью операции new. Такая функция называется *конструктором*. В PHP4 имя этой функции совпадает с именем класса, однако в PHP5 конструктором класса является метод с именем \_\_construct(). При уничтожении объекта вызывается специальный метод \_\_destruct() – деструктор класса.

<?php //ppr42

class MyClass {

private $property;

function \_construct() {

echo "Запущен конструктор";

}

function \_\_destruct() {

echo "Запущен деструктор";

}

function MyClass($n = 1) {

echo "Запущен другой конструктор";

$this->property = $n;

}

}

$obj = new MyClass(); // Выводит "Запущен конструктор"

unset($obj); // Выводит "Запущен деструктор"

$obj1 = new MyClass(10);

?>

Вывод:

*Запущен конструктор. Запущен деструктор. Запущен конструктор. Запущен деструктор.*

Для совместимости с предыдущей версией, если при создании объекта в классе не найдется конструктора \_\_construct(), то PHP5 пытается выполнить метод, имя которого совпадает с именем класса(как в PHP4).

В приведенном выше примере конструктор не перегружается, а вызывается первый конструктор.

В случае наследования классов конструктор базового класса при создании порожденного класса неявно не вызывается. Если необходимо

вызвать конструктор или деструктор базового класса из порожденного класса, это можно сделать явно, через указатель parent.

<?php //ppr43

class MyClass {

function \_\_construct() {

echo "конструктор базового класса";

}

function \_\_destruct() {

echo " деструктор базового класса";

}

}

class DerivedClass extends MyClass {

function \_\_construct() {

echo "конструктор порожденного класса";

parent::\_\_construct();

}

function \_\_destruct() {

echo " деструктор порожденного класса";

parent::\_\_destruct();

}

}

$obj = new DerivedClass();

unset($obj);

?>

Вывод:

*конструктор порожденного класса, конструктор базового класса,*

*деструктор порожденного класса, деструктор базового класса*

Рассмотрим еще один пример вывода даты/времени:

<?php

class Dateclass {

var $month = array( "Январ", "Феврал", "Март", "Апрел", "Ма", "Июн", "Июл","Август","Сентябр", "Октябр", "Декабр", "Январ");

var $day = array ( "Воскресенье", "Понедельник", "Вторник", "Среда", "Четверг", "Пятница", "Суббота");

var $dnum; var $mnum; // Переменные класса

var $daym; var $year;

function Dateclass() { //Конструктор: инициализация переменных

$this->dnum = date("w");

$this->mnum = date("n");

$this->daym = date("d");

$this->year = date("Y");

}

function show(){// метод

$dnum = $this->dnum;

$mnum = $this->mnum;

$daym = $this->daym;

$year = $this->year;

$textday = $this->day[$dnum];

$monthm = $this->month[$mnum-1];

if ($mnum == 3||$mnum == 8)

{ $k = "а"; }

else {$k = "я";}

echo "Сегодня: $textday, $daym $monthm$k $year г.";

}

}

$obj = new Dateclass(); // Создание объекта

$obj -> show(); // Вызов метода из объекта

?>

Результат:

*Сегодня: Воскресенье, 05 Апреля 2013 г.*

В PHP объекты представляют объектные ссылки, передача объекта в качестве параметра функции происходит по ссылке, а не по значению. Если необходимо провести копирование объекта, то используется клонирование объекта со всеми методами, свойствами и значениями:

<?php //ppr44

class MyClass{

public $property = 1;

}

$obj1 = new MyClass;

$obj2 = $obj1; // ссылка на тот же объект

echo $obj1->property; // Выводит 1

$obj2->property = 2;

echo $obj1->property; // Выводит 2

$obj3 = clone $obj1;//создание нового объекта

echo $obj3->property; // Выводит 2

$obj3->property = 3;

echo $obj1->property; // Выводит 2

?>

Для копирования объекта используется ключевое слово clone, вызывающее метод \_\_clone() к которому нельзя обратиться непосредственно. Метод \_\_clone() можно перегрузить в классе, его перегрузка позволит изменить значения свойств копируемого объекта:

<?php //ppr45

class MyClass{

var $property;

function \_\_clone() {

$this->property = 2;

}

}

$obj1 = new MyClass;

$obj1->property = 1;

$obj2 = clone $obj1;

echo $obj1->property; // Выводит 1

echo $obj2->property; // Выводит 2

?>

Метод \_\_clone() не может принимать никакие аргументы, однако позволяет обратиться к исходному объекту через указатель $this и получаемому в результате копирования объекту через указатель $that.

### 6.3.3. Наследование классов и интерфейсов

Класс может быть наследником только одного суперкласса и множества интерфейсов. Пример

<?php

class MyClass extend BaseSuperClass implements I1,I2,I3{

//Реализация класса

}

?>

Класс MyClass наследует свойства одного базового класса BaseSu­perClass и интерфейсов I1,I2,I3. В вершине иерархии наследования размещают абстрактные базовые классы, содержащие абстрактные методы. Абстрактные методы не имеют реализации.

<?php //ppr46

abstract class MySuperClass {

abstract public function abstrFunc();//абстрактный метод

}

class MyClass extends MySuperClass {

public function abstrFunc() {

echo “Be-Be-Be”;

}

}

$obj = new MyClass;

$obj->abstrFunc(); // Выводит Be-Be-Be

?>

Невозможно создать объект абстрактного класса, можно только определить производный от него класс, реализующий методы, и создать объект от производного класса. Стоит отметить, что абстрактные классы также могут содержать и не абстрактные методы. Класс имеет только одного предка, но зато может быть создан на основе любого числа интерфейсов.

Интерфейсами (interface) являются чисто абстрактные классы, содержащие только абстрактные методы и вообще не имеющие никаких свойств. При этом методы объявляются без указания каких-либо спецификаторов, в том числе и abstract и не реализованы.

<?php //ppr47

interface Int1 {

function func1();

}

interface Int2 {

function func2();

}

class MyClass implements Int1, Int2 {

public function func1() {

echo 1;

}

public function func2() {

echo 2;

}

}

$obj = new MyClass;

$obj->func1(); // Выводит 1

$obj->func2(); // Выводит 2

?>

Таким образом, хотя множественное наследование классов и не поддерживается, можно создавать классы на основе нескольких интерфейсов.

В PHP можно определять методы класса и классы как финальные (final). Метод, при определении которого используется ключевое слово final, не может быть переопределен в производных классах.

<?php

class MyClass {

final public function func() {

// Код метода

}

}

class MyClass1 extends MyClass {

// Следующий код вызывает ошибку

// переопределения финального метода

// базового класса MyClass

public function func() {

// Код метода

}

}

?>

Кроме этого, если final используется при определении самого класса, то порождение от него других классов становится невозможным.

<?php //ppr48

final class MyClass {

// Код описания класса

}

// Следующий код вызывает ошибку

// порождения от финального класса

class MyClass1 extends MyClass {

// Код описания класса

}

?>

Если класс определен как final, то и все методы данного класса автоматически становятся финальными, так что, определять их явно как final уже нет необходимости. В отличие от Java, определять свойства класса как финальные константы в PHP недопустимо. В PHP для этого введен новый элемент класса – константа класса.

<?php

class MyClass {

const CONSTANT = "константа класса";

}

echo MyClass::CONSTANT; // Выводит "константа класса"

?>

В PHP возможно объявление статических свойств класса.

<?php

class MyClass {

static $static = 1;

}

echo MyClass::$static; // Выводит 1

?>

В PHP используются статические свойства и методы классов. Статические свойства едины для всех объектов класса. Изменение такого свойства в одном из методов любого объекта приводит к его изменению для всех остальных объектов данного класса. Статические методы классов как и статические свойства, принадлежат всему классу в целом. Это позволяет использовать такой метод без создания объекта класса.

<?php

class MyClass {

static function statFunc() {

echo "статический метод";

}

}

MyClass::statFunc(); // Выводит "статический метод"

?>

В статическом методе невозможно использовать указатель на текущий объект $this, так как при вызове такого метода неизвестно, в контексте какого объекта он вызывается, такого объекта может и не существовать.

Ключевое слово instanceof позволяет определять, является ли объект экземпляром определенного класса или же экземпляром производного класса.

<?php //ppr49

class MyClass { }

$obj1 = new MyClass();

if ($obj1 instanceof MyClass) {

echo "\$obj1 – объект класса MyClass";

}

class MyClass1 extends MyClass { }

$obj2 = new MyClass1();

if ($obj2 instanceof MyClass) {

echo "\$obj2 – объект класса, производного от MyClass";

}

interface Int { }

class MyClass2 implements Int { }

$obj3 = new MyClass2();

if ($obj3 instanceof Int) {

echo "\$obj3 – объект класса, созданного на основе интерфейса Int";

}

?>

С помощью instanceof можно также определить, является ли объект экземпляром класса, созданного на основе определенного интерфейса.

В PHP введена возможность разыменования (dereferencing) объектов, которые возвращаются функциями.

<?php //ppr50

class MyClass1 {

public function showClassName() {

echo "объект класса MyClass1";

}

}

class MyClass2 {

public function showClassName() {

echo "объект класса MyClass2";

}

}

function deref($obj) {

switch ($obj) {

case "MyClass1":

return new MyClass1();

case "MyClass2":

return new MyClass2();

}

}

deref("MyClass1")->showClassName(); // Выводит "объект

// класса MyClass1"

deref("MyClass2")->showClassName(); // Выводит "объект

// класса MyClass2"

?>

Данный механизм позволяет вызывать методы объектов, имена классов которых возвращаются пользовательскими функциями.

В PHP имеется возможность производить уточнения типов классов (class type hints), которые передаются методам в качестве параметров.

<?php //ppr51

interface Int1 {

function func1(Int1 $int1);

}

interface Int2 {

function func2(Int2 $int2);

}

class MyClass implements Int1, Int2 {

public function func1(Int1 $int1) {

// Код метода

}

public function func2(Int2 $int2) {

// Код метода

}

}

$obj1 = new MyClass;

$obj2 = new MyClass;

$obj1->func1($obj2);

$obj1->func2($obj2);

?>

При этом уточнение типов классов производится только на этапе исполнения программы.

Магические методы

Функции [\_\_construct](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.decon.html#language.oop5.decon.constructor), [\_\_destruct](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.decon.html#language.oop5.decon.destructor), [\_\_call](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.overloading.html#language.oop5.overloading.methods), [\_\_callStatic](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.overloading.html#language.oop5.overloading.methods), [\_\_get](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.overloading.html#language.oop5.overloading.members), [\_\_set](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.overloading.html#language.oop5.overloading.members), [\_\_isset](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.overloading.html#language.oop5.overloading.members), [\_\_unset](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.overloading.html#language.oop5.overloading.members), [\_\_sleep](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.magic.html#language.oop5.magic.sleep), [\_\_wakeup](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.magic.html#language.oop5.magic.sleep), [\_\_toString](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.magic.html#language.oop5.magic.tostring), [\_\_invoke](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.magic.html#language.oop5.magic.invoke), [\_\_set\_state](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.magic.html#language.oop5.magic.set-state), [\_\_clone](file:///C:\Documents%20and%20Settings\Admin\Documents%20and%20Settings\Admin\Application%20Data\Admin\Application%20Data\Microsoft\php\php_manual_en\html\language.oop5.cloning.html), имена которых начинаются с символа " \_ ", имеют особую функциональность в PHP-классах. Методы доступа к свойствам объектов \_\_get($name) и \_\_set($name,$value) позволяют проводить динамическое назначение значений свойствам объектов. При вызове несуществующего метода объекта автоматически вызывается специальный метод \_\_call().

<?php

class MyClass {

function \_\_call($name, $params) {

echo "Вызван метод $name с параметром $params[0]";

}

}

$obj = new MyClass;

echo $obj->method(1); // "Вызван метод method с параметром 1"

?>

В качестве параметров \_\_call() принимает имя вызываемого метода и передаваемые этому методу параметры. Псевдо-константа \_\_METHOD\_\_ возвращает имя класса и вызываемый метод.

Метод \_\_toString() позволяет выполнить перегрузку преобразования объекта в строку.

Обработка ошибок

В версии PHP5 введена современная схема обработки *исключений*. Конструкция **try/catch/throw** позволит весь код обработки ошибок локализовать в одном месте сценария.

Пример:

<?php //ppr52

try {

$fp = @fopen("file.txt", "w");

if (!$fp) throw new Exception("Невозможно открыть файл!");

// Запись данных в файл

fclose($fp);   
 }

catch (Exception $exception) {

echo "Ошибка в строке ", $exception->getLine();

echo $exception->getMessage(); // Выводит "Невозможно

// открыть файл"

}

?>

В конструкции можно использовать *несколько блоков catch*. Также возможно создание собственных классов исключений, производных от встроенного класса Exception.

Оператор @ служит в PHP для отключения вывода сообщений об ошибках на текущей строке кода. Практика использования @ не является наилучшей, в нагруженных приложениях использовать @ невозможно из-за ощутимого торможения. Ошибки нужно ловить с помощью Exceptions или исправлять.

В PHP 7 появилось много новых классов исключений, призванных обрабаты­вать типы ошибок, с которыми можно столкнуться. Для обеспечения совместимости между версиями, добавлен новый интерфейс **Throwable**, он может быть реализован как исключениями уровня ядра, так и пользовательскими исключениями.

Анонимные классы

Анонимные классы — двоюродные братья анонимных функций, Анонимные классы легко создаются и используются так же, как и обычные объекты. Вот пример из документации.  
  
До PHP 7:

class MyLogger {

public function log($msg) {

print\_r($msg . "\n");

}

}

$pusher->setLogger( new MyLogger() );

Использование анонимного класса:

$pusher->setLogger(new class {

public function log($msg) {

print\_r($msg . "\n");

}

});

Приведем несколько обновлений:

В PHP 7 расширили ранее существовавшее объявление параметров в методах (классах, интерфейсах и массивах) путем добавления четырех скалярных типов — целого (int), с плавающей запятой (float), логического (bool) и строкового (string) в качестве возможного типа параметра.  
  
Кроме того, опционально мы можем указать тип результата, возвращаемого функцией или методом. Поддерживаются типы bool, int, float, string, array, callable, имя класса или ин­терфейса и parent (для методов класса).

class Calculator

{

*// объявляем, что параметры имеют целый тип integer*

public function addTwoInts(int $x, int $y): int {

*// явно объявляем, что метод возвращает целое*

return $x + $y;

}

}

### Ошибки PHP и журнал error\_reporting

Библиотека PHP может генерировать ряд ошибок, не относящихся к уровню исключений, – ошибок OOP.

В PHP обрабатываются три уровня ошибок: информационные ошибки, ошибки с возможностью действия (предупреждения) и неустранимые (фатальные) ошибки. Информационные ошибки: E\_NOTICE-компилятор столкнулся с логической проблемой, которая может привести к ошибке выполнения. Аналогично: E\_USER\_NOTICE – генерируемое пользователем уведомление.

Предупреждения означают, что, возможно, придется завершить работу с выдачей сообщения из-за того, что нельзя открыть файл и т. д. Ошибка E\_WARNING – предупреждение, генерируемое во время выполнения программы и связанное с возможностью выполнения функции или блока. Аналогично: E\_USER\_WARNING – генерируемое пользователем предупреждение. E\_COMPILE\_WARNING – предупреждения о времени компиляции.

Неустранимые (фатальные) ошибки возникают тогда, когда выполнение или загрузка программы невозможны. E\_ERROR(1) – фатальные ошибки времени выполнения; E\_PARSE(4) – ошибки разбора времени компиляции; E\_COMPILE\_ERROR(64) – фатальные ошибки времени компиляции; E\_CORE\_ERROR(16) – фатальные ошибки при начальном старте PHP; E\_ALL – все вышеуказанное.

Уровень сообщений об ошибках устанавливается как значение директивы error\_reporting. В PHP установлено значение по умолчанию error\_reporting = E\_ALL & ~E\_NOTICE, что означает отображение всех ошибок и предупреждений, за исключением логических ошибок, которые имеют уровень E\_NOTICE. Начальное значение error\_reporting может быть изменено в php.ini-файле директивой error\_reporting, в Apache httpd.conf-файле директивой php\_error\_reporting и, наконец, оно может быть установлено на этапе прогона скрипта функцией error\_reporting().

Параметр error\_reporting в php.ini файле устанавливает уровень строгости ошибок E\_ERRORLEVEL, которые будут обрабатываться внутренним обработчиком ошибок. Параметр display\_error устанавливает вывод сообщений об ошибках на экран; параметр log\_errors – регистрацию ошибок в журнальном файле, указанном в параметре error\_log.

Cуществует большое количество скриптов не справляющихся с пользовательскими ошибками. Любой скрипт может «свалиться» при наступлении критичных условий. Всегда нужно: проверять результаты вызова функций; проверять результаты системных вызовов. При вызове функции, результаты которой подвергаются дальнейшей обработке, следует убедиться, что возвращаемые данные находятся в интервале допустимых значений. При обращении к внешним файлам или процессам всегда следует проверять, все ли работает корректно. Например, при вызове функции sql\_connect() следует убедиться, что подключение к БД действительно имело место.

$conn = @sql\_connect ($host, $user, $pass);

if (!$conn) {

die (sprintf ("Ошибка [%d]: %s", sql\_errno (), sql\_error ()));

}

?>

При разработке следует установить уровень сообщений об ошибках error\_reporting в файле php.ini на E\_ALL, т. е. на высший уровень. При другой конфигурации многие ошибки типа «неверное регулярное выражение», «недопустимое значение» могут ускользнуть. Предположим, например, что error\_reporting выставлен не на максимум, а, на E\_ERROR.

Скрипт выполняет функцию do\_math, но не сообщает об ошибке «деление на ноль», которая, однако, имела место (при $i=$j=0 вывода результата просто не было).

<?php

error\_reporting(E\_ERROR);

mt\_srand ((double)microtime() \* 1000000);

function do\_math ($a, $b) {

return (($a - $b) \* 2) / mt\_rand();

}

for ($i = 5, $j = –5; $i > –5; $i– –, $j++){

print $j / do\_math ($i, $j) . "\n";

}

?>

PHP выдает сообщения об ошибках непосредственно в браузер и не позволяет подавить их. Однако в PHP имеется возможность перехвата таких сообщений с помощью функции set\_error\_handler(). В следующем примере set\_error\_handler() назначает обработчиком по умолчанию функцию error\_handler(). В случае возникновения ошибки вызывается error\_handler(), и встроенная функция error\_log() регистрирует сбой в файле error\_file. Если происходит ошибка класса E\_ERROR, работа скрипта прекращается и выводится сообщение об ошибке.

<?php

// Индивидуальный обработчик ошибок, определен функцией

// set\_error\_handler()

function error\_handler ($type, $message, $file = \_\_FILE\_\_, $line = \_\_LINE\_\_) {

error\_log("$message, $file, $line", 3, 'error\_file');

if ($type & E\_ERROR) {

print 'Произошла ошибка, зарегистирована.';

exit;

}

}

set\_error\_handler('error\_handler');

?>

**Конструкции включений.** Cкрипт PHP можно собрать из нескольких отдельных файлов, html-страниц и php-скриптов, используя конструкции require и include. Конструкция require позволяет включать файлы в сценарий PHP до исполнения сценария, конструкция include позволяет включать файлы в код PHP-скрипта и во время выполнения сценария. При запуске программы интерпретатор заменит require или include на содержимое файла, например:

<?php // script.php

require "header.htm";

include "footer.htm";

require\_once "header.htm";

include\_once "footer.htm";

?>

Поясним разницу между require и include. При выполнении включения PHP анализирует строки сценария, пока не доходит до конструкции include. Дойдя до include, PHP переключается на указанный в include файл, что может снизить быстродействие. С require таких проблем нет, поскольку файлы включаются до выполнения сценария.

В сценариях бывает сложно контролировать включение в скрипт одного и того же файла несколько раз. Используя конструкции однократного включения require\_once и include\_once, можно быть в этом уверенным.

### 6.4. Файлы

Файлы могут быть текстовыми, содержащими символьные строки и бинарными, представляющими последовательность байт. При работе с файлами осуществляются три действия: открытие файла для чтения или записи; чтение или запись; закрытие файла.

Функция открытия файла:

fopen ($filename, $mode [, use\_include\_path]).

Функция открывает файл и возвращает его дескриптор или false. Режим открытия mode может принимать следующие значения:

Бинарный файл ”b”, текстовой “t” или значение по умолчанию.

"r" – файл открывается только для чтения. "r+" – открывается на чте­ние и на запись. Текущий указатель файла устанавливается в его начало;

"w" – файл открывается для перезаписи. Указатель файла устанавливается в его начало. Все старое содержимое файла теряется, счетчик длины файла устанавливается равным 0. Если файл с указанным именем не существует, функция пытается его создать;

"w+" – файл открывается для чтения и записи. Указатель файла устанавливается в его начало. Все старое содержимое файла теряется;

"a" – файл открывается на добавление (запись). Указатель файла по­мещается в конец файла. Если файла с указанным именем не существует, функция пытается его создать. "a+" – файл открывается для чтения и записи. Вот несколько примеров открытия файлов:

$fp = fopen ("/home/a/bases.dat", "br");

$fp = fopen ("/home\_dir/client/count.txt", "w+");

$fp = fopen ("http://www.yahoo.com/pp.txt", "r");

В случае успеха функция fopen() возвращает ссылку на открытый файл, а в случае ошибки – булевское значение false. В РНР имеется возможность удаленного открытия файлов путем указания URL-адреса в качестве параметра $filename функции fopen (). Осуществлять запись в таких файлах, нельзя, но можно их читать.

URL, содержащие недопустимые символы, например пробельные символы в имени файла, необходимо кодировать перед их использова­нием. Функция urlencode () принимает параметр URL, подлежащий кодированию, и возвращает закодированный URL.

Пример:

<?php //ppr53

/\* Открыть файл для чтения \*/

$fr = fopen("myfile.txt", 'r');

/\* Открыть бинарный файл для чтения/добавления \*/

$fr = fopen("myfile.dat", 'ba+');

/\* Открыть файл для чтения/записи (искать файл в пути,

заданном директивой include\_path)\*/

$fr = fopen("code.php", 'w+', true);

/\* Открыть файл index.php, для чтения по протоколу HTTP \*/

$fr = fopen("http://www.php.net/index.php", 'r');

/\* Открыть файл index.php, для чтения по протоколу FTP \*/

$fr = fopen("ftp://ftp.php.net/index.php", 'r');

/\*Закодировать URL, затем открыть для чтения по протоколу HTTP \*/

$url = "http://www.php.net/this is my invalid URL.php";

$url = urlencode($url);

$fr = fopen($url, 'r');

?>

Файл закрывается с помощью функции bool fclose ($fp).

Для работы с текстовыми файлами используются функции fgets (), которая извлекает строку, и fputs (), которая записывает строку в файл. Функция считывания fgets( $fp, $length) возвращает строку дли­ной до length-1 байт, и завершается после загрузки length-1 символов либо после обнару­жения символа конца строки, либо при обнаружении признака конца файла. Пример построчного вывода на экран содержи­мого файла:

<? //ppr55

$fp = fopen ("pr415.php", "r");

while (!feof ($fp)) {

$stroka = fgets ($fp, 80);

echo $stroka."<BR>";

}

fclose($fp);

?>

Загрузка строки с пропуском HTML-тегов выполняется функцией fgetss ($fp, $length [, string allowable\_tags]). При вводе из считанной строки удаляются все HTML-теги, которые в ней присутствуют. Есть возможность использовать необязательный третий аргумент для указания разрешенных тегов, которые удалены не будут.

Альтернативой fgets () является функция fscanf (). Синтаксис:

fscanf ($filename, $format , $var\_one [, $var\_two ...]]])

где $filename – входной поток; $format задает шаблон для чтения; $var\_one, $var\_two – переменные списка ввода, передаваемые по ссылке. В случае успешного завершения ввода fscanf () возвращает количество разобранных элементов, а случае ошибки возвращает false. Допустимые символы форматирования функции fscanf (): %b – двоичное число; %с – одиночный символ; %d – десятичное число со знаком; %u – десятичное число без знака; %f – число с плавающей запятой; %о – восьмеричное число; %s – строка; %x – шестнадцатеричное число.

Функция fputs () служит для записи строки (или любых других данных) в указанный поток и имеет следующий синтаксис: fputs($filename, $data [, int $length]), где $filename выходной поток; $data содержит записываемые данные; необязательный параметр $length задает размер фактически записываемых данных.

Чтение и запись бинарных файлов

*Чтение данных из файла string fread (int $fp, int length).* Функция осуществляет чтение до length байт из файла, адресуемого указателем fp. Чтение прекращается, если будет считано указанное количество байт, или будет достигнут конец файла.

*Запись данных в файл: int fwrite (int $fp, string str [, int length]).*

С помощью функции осуществляют запись содержимого строки string в файл, адресуемый указателем fp. Если задан аргумент length, операция записи прекращается после вывода указанного количества символов либо при достижении конца строки.

<? //ppr56

$open = fopen("my\_file.txt","w+");

//Очищаем файл и добавляем в него строку,

//если файл не существует, то он создаётся:

fwrite($open,"строка\r\n");

fclose($open);

//Добавляем новую строку в конец файла:

$open = fopen("my\_file.txt","a");

fwrite($open,"новая строка\r\n");

fclose($open);

?>

В созданном файле my\_file.txt будут записаны две строки:

строка

новая строка

Существуют другие функции ввода, например загрузка всего файла:

array file ($filename [, int use\_include\_path]).

Функция file() записывает файл в мас­сив. При этом каждый элемент массива представляет собой одну строку файла. Символ новой строки является последним символом каждой строки.

<? //ppr57

//Создаем массив $array, где каждый индекс будет равняться номеру строки в файле:

$array = file("ppr56.php");

print\_r($array);

$count = count($array); // количество строк в файле

echo $count;

?>

Будет выведено:

*Array ( [0] => //Создаем массив $array, где каждый индекс будет равняться но­меру строки в файле: [2] => $array=file("php416.php"); [3] => print\_r($array); [4] => //Подсчитываем количество строк в файле: [5] => $count=count($array); [6] => echo $count; [7] => ?> ) 8*

<?php //ppr58.php – вывод строк из файла

$fi = "ppr58.php"; // имя файла

$array = file($fi);

//Считываем из файла первые 3 строки:

echo "<hr>";

$n = 3; //количество считываемых строк

for($i = 0;$i<$n;$i++)

{print "$i : $array[$i],'\n'"; }

for($i = (count($array)–$n–1);$i<count($array)–1; $i++)

{ echo "$i : $file[$i],'\n'"; }

?>

Будет выведено три первых и три последних строки:

0 : <?php //php418.php – вывод строк из файла ,

' '1 : $fi = "pr418.php"; // имя файла ,

' '2 : $array = file($fi); ,

' '10 : { echo "$i : $array[$i],'\n'"; } ,

' '11 : ,

' '12 : ?> ,' '

Объединяем два файла в один массив $new\_array:

<? //ppr59

$files = array(

"php56.php", # первый файл

"php57.php" # второй файл

);

for($i = 0;$i<count($files);$i++)

{

$array[] = file($files[$i]);

}

while(list(,$result) = each($array))

{

for($i = 0;$i<count($result);$i++)

{ $new\_array[] = $result[$i]; }

}

print\_r($new\_array); #выводим массив

?>

В следующем примере массив устанавливается в файл. Затем с помощью функции fwrite() вы­полняется замена строки в файле на указанную:

<?PHP //ppr60

$line = "1"; # строка, которую нужно изменить

$replace = "du du du"; # на что нужно заменить

$filarr = file("my\_file.txt");

$f = fopen("my\_file.txt","w");

for($i = 0;$i<count($filarr);$i++)

{

if(($i+1)! = $line){fwrite($f,$file[$i]);}

else{fwrite($f,$replace."\r\n");}

}

fclose($f);

?>

Поиск слова в файле. Ищем в файле file.txt слово «привет»:

<?PHP//ppr61

$word = "привет"; # искомое

$f = "file.txt"; # имя файла, в котором будем искать слово

$open = fopen($file,"r");

while(!feof($open)) $search. = fgets($open,1024);

fclose($open);

if(eregi(strtolower($word),strtolower($search)))

{ echo "Слово ".$word." – найдено"; }

else

{ echo "Слово ".$word." – не найдено"; }

?>

Прямой доступ к указанной позиции бинарного файла (номеру байта) осуществляется с помощью функции fseek($file, $offset [, $reference]);

$offset указывает число байт смещения от начала файла ($reference =SEEK\_SET или по умолчанию), от текущей позиции файла ($reference=SEEK\_CUR), от конца файла ($reference= SEEK\_END). Функция fseek () может использоваться только для файлов, расположен­ных в локальной файловой системе, и не работает с файлами, которые открываются удаленно по протоколу FTP. Функция fseek () возвращает нуль в случае успешного завершения и –1, если ошибка. Отсчет смещений указателя в fseek() начинается с нуля.

Копирование одного файла в другой можно выполнить с помо­щью функции copy(filefrom,fileto);. Функция unlink(filename) удаляет файл. Функция rename(oldname, newname) переименовывает файл.

### Работа с каталогами в РНР

РНР предоставляет набор функций для работы с каталогами: opendir() и closedir. Функция opendir () имеет следующий синтаксис: $dir\_reference =opendir($dir\_path);

Функция opendir() выведет сообщение об ошибке, если указанный каталог не существует или дескриптор каталога, в случае успеха. Функция closedir() принимает единственный параметр - дескриптор каталога из вызова opendir().

После открытия каталога его элементы можно прочитать с помощью функции readdir($dir\_reference), где $dir\_reference – значение, ко­торое возвращает функция opendir(). При успешном завершении функ­ция возвращает строку, содержащую имя одного из файлов каталога. Каждый последующий вызов функции readdir () возвращает очередной файл, пока весь список файлов не будет исчерпан. Если файлов в ката­логе больше нет или произошла ошибка, readdir() вернет значение false. Выведем содержимое папки c:/windows

<?php //ppr62

$dir = "c:/windows"; # папка, которую нужно прочитать

if($OpenDir = opendir($dir))

{

while(($file = readdir($OpenDir)) !== false)

if($file ! = "." && $file ! = "..")

echo $file."<br>";

}

else echo "нет прав";

?>

Для создания каталога можно использовать функцию mkdir(name), для удаления – rmdir(), для изменения – chdir(). Пример чтения каталога:

<?php//ppr63

$dr = @opendir(' /tmp/') ;

if(!$dr) {

echo "Ошибка при открытии каталога /tmp/!<BR>"; exit;

while (($files) = readdir($dr)) !== false);

print\_r(Sfiles);

?>

Здесь символ "@" используется для подавления вывода сообщения об ошибке.

Если необходимо повторно просмотреть содержимое каталога, РНР предоставляет функцию, которая позволяет «перемотать» оглавление каталога в исходное состояние. Эта функция, называемая rewinddir (), имеет следующий синтаксис: rewinddir($dir\_reference)

Полезным является метод получения списка файлов, удовлетворяющих определенному шаблону. Этой цели служит функция glob($filemask [, flags]). Здесь $filemask – это строка, содержащая шаблон поиска (напри­мер, \*.txt). Значения flags: GLOB\_MARK – добавлять слэш к именам, которые являются каталогами; GLOB\_ONLYDIR – вернуть только каталоги, совпадающие с шаблоном. Использование функции glob ():

<?php//ppr64

$directories = glob("/tmp/\*", GL0B\_ONLYDIR);

$complete = glob("/tmp/\*");

$files = array\_diff($directories, $complete);

echo "Каталоги в /tmp/<BR>";

foreach($directories as $val) ( echo "$val<BR>\n";)

echo "<BR><&aunbi в /tmp/<BR>";

foreach($files as $val) ( echo "$val<BR>\n";)

?>

### 6.5. Передача данных по протоколу HTTP

Передача гипертекстовых данных по протоколу HTTP основывается на запросах/ответах. Запрос создается браузером клиента. Когда пользователь набирает в браузере адрес: http://bsu.by:80/my.php, вот что при этом происходит:

Браузер разбирает URL-адрес и определяет действие:

1. Использовать протокол HTTP.

2. Извлечь запрошенный ресурс с сервера на bsu.by, порт 80.

3. Использовать приложение /my.php.

На основе данной информации создается запрос HTTP в следующей форме: метод запроса, URI, версия протокола, сообщение, содержащее управляющую информацию запроса, информацию о клиенте и тело сообщения. Запрос(Request) выглядит так:

GET /my.php HTTP/1.1

Accept: image/gif, image/png, image/jpeg, \*/\*

Accept-Language: en-us

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 8.0;

Windows XP; .NET CLR 1.1.4322)

Host:bsu.by

Connection: Keep-Alive

Вот что эти строки значат:

GET – метод HTTP: "Дай информацию для скрипта в /mу.php и вышли ее, используя протокол HTTP 1.1".

Accept – "понимаю графическую информацию в форматах".

Accept-Language – "язык – английский, американский".

Accept-Encoding – "понимаю типы сжатия gzip и deflate".

User-Agent – "тип браузера – IE 8, выполняющийся под управлением Windows XP".

Host – "доставь информацию для скрипта /my.php компьютера на хосте bsu.by".

Connection: Keep-Alive – "держать подключение HTTP открытым, пока браузер не закроет его". Это повышает производительность, иначе веб-страница будет иметь множество под­ключений.

В конце запроса помещается его тело.

Сервер отвечает на запрос сообщением, содержащим строку статуса (включая версию протокола и код статуса – успех или ошибка), за которой следует MIME-подобное сообщение, включающее информацию о сервере, метаинформацию о содержании ответа и само тело ответа. Ответ (response) сервера посылается клиенту (браузеру) в виде:

НТТР/1.1 200 0К

Date: Моn, 08 Dec 2012 16:46:40 GMT

Server: Apache/1.3.27 (Unix) mod\_throttle/3.1.2 PHP/5.3.2

X-Powered-By: PHP/4.3.2

X-Accelerated-By: PHPA/1.3.3r2

Connection: close

Content-Type: text/html; charset = utf-8

<html lang = "en-US" xml:lang = "en-US" «nlns = "http://www.w3.org/1999/xhtml">

<head>

<script>

HTTP-ответ состоит из двух частей. Вначале идет порция сведений о самой запрошенной информации – заголовок ответа (response header). Затем следует пустая строка и далее – сама запрошенная информация. Эта вторая часть называется телом (body). Вот что означают эти части:

HTTP/1.1 – сообщает клиенту, что информация будет отправлена по протоколу HTTP версии 1.1. Код 200 состояния HTTP означает: «Все хорошо, документ найден и будет отправлен».

Date – сообщает клиенту дату, установленную на сервере, с которого поступает информация. Время GMT, т. е. по Гринвичу.

Server – тип сервера, предоставляющего информацию.

Connection – сообщает клиенту, что соединение будет закрыто после того, как сервер завершит отправку информации.

Content-Type – сообщает клиенту, какой тип содержимого будет отправлен. В дополнение также указывается набор символов.

### Клиентские методы HTTP. Метод клиента определяет запрос, отправленный от веб-клиента HTTP-серверу. Существуют три основных типа запросов: по методам GET, POST, HEAD. По методу GET контент запроса посылается через поля формы или через строку URL, по методу POST – через поля формы. Запросы можно создать и отправить с помощью средств JavaScript. Если необходимо получить информацию о запрошенном URL, но не информацию самого URL, то используют запрос HEAD.

Простой способ просмотреть HTTP-запросы и ответы состоит в установке плагина для браузера Firefox, который называется HttpFox и доступен для свободного скачивания (рис. 6.2.). Пример:

## 

*Рис. 6.2.Http-запрос и ответ*

### 6.6. Обработка HTML-форм

Одно из главнейших достоинств PHP – реализация возможности работы с формами HTML для отправки информации на сервер и получения информации с сервера. Форма состоит из поименованных элементов, заключенных между HTML-тэгами <form...> и </form>. В качестве элементов формы могут выступать поля ввода текста, кнопки, выпадающие меню, переключатели, checkbox, а также картинки. Каждый элемент формы может пересылаться на сервер, после чего становится доступен серверным программам на PHP.

Важным свойством формы является то, что в ее заголовке в открывающем тэге <form action=”my.php”> можно указать адрес файла, содержащего код PHP. При загрузке кода в него передадутся значения всех элементов формы. Для поля ввода текста это введенный текст, для переключателя или checkbox – это true при отмеченном и false при неотмеченном выборе, для рисунка – координаты указателя мыши относительно верхнего левого угла изображения.

Переменные, передаваемые через форму, помещаются в глобальные ассоциативные массивы $\_POST\_ или $\_GET. Например, cодержимое поля ввода текста

*<input type = text name = ’myname’ size = 30>*

окажется в элементе $\_GET['myname']. Различие между двумя методами состоит в том, что при передаче данных методом GET эти данные отображаются в адресной строке браузера, а при использовании метода POST – нет. Переменные $\_POST и $\_GET доступны и во всех функциях, расположенных в программе РНР, так как являются глобальными. Часто используемые суперглобальные переменные PHP: $\_GET[], $\_POST[], $\_REQUEST[], $\_COOKIE[], $\_FILES[], $\_SERVER[], $\_ENV[], $\_SESSION[].

### 6.6.1. Передача переменных из формы в скрипт

Каждое поле ввода имеет имя (атрибут name) и значение передаваемые в обработчик. Обработчик – это PHP-скрипт на сервере. Пример:

<form action = "action.php" method = "POST">

Ваше имя: <input type = "text" name = "name" />

Ваш возраст: <input type = "text" name = "age" />

Пароль:<input type = "text" name = "password" />

<input type = "submit">

</form>

Когда пользователь заполнит форму и нажмет кнопку submit, будет вызван обработчик события *action.php*. В этом файле может быть:

<html>

<body>Здравствуйте,

<?php //ppr65

echo $\_POST["name"];

?>.

Вам <?php echo $\_POST["age"]; ?> лет.

Пароль:<?php echo $\_POST["password"]; ?>

</body></html>

Вывод:

*Здравствуйте, Егор. Вам 20 лет. Пароль:\*\*\*\*\*\**

Переменные $\_POST["name"] и $\_POST["age"] автоматически установлены в значения "name" и "age" из формы средствами PHP. Если бы мы использовали метод GET, то информация нашей формы была бы в суперглобальной переменной $\_GET ["name" ] и $\_GET["age"]. Можно использовать переменную $\_REQUEST, если источник данных не имеет значения. Эта переменная содержит смесь данных $\_GET, $\_POST, $\_COOKIE и $\_FILE.

При использовании метода GET все данные упаковываются в адресную строку следующим образом: вначале имена переменных и их значения преобразуются в вид, безопасный для передачи в строке URL. Все символы, кроме алфавитно-цифровых и знака подчер­кивания "\_"(кирилица), заменяются знаком процента "%" и двумя шестнадцатерич­ными цифрами кода. Пробелы заменяются на "%20". После этого они преобразуются в форму name = value и собираются в строку, отделяясь друг от друга знаком & (амперсанд). Происходит это так:

http://site.domain/action.php?имя = значение&имя = значение

Пары "имя = значение" создаются для каждого элемента ввода, для которого указано имя атрибутом NAME. В таком виде данные и попадают к скрипту, который должен проделать обратную операцию и извлечь из строки названия переменных и их значения. Строка запроса может содержать не более 254 символов. Это одно из ограничений метода GET, которое не позволяет передавать объемные данные. В отличие от других серверных языков программирования, PHP делает всю работу по расшифровке строки запроса, переданной методом GET.

В настоящее время передача данных происходит через суперглобальные массивы $\_GET и $\_POST. Пример:

<html> <!—ppr66.php -->

<head> </head>

<body>

<?php //

$a = $\_GET['a'];

$b = $\_GET['b'];

$с = $\_GET['с'];

$c++;

echo "a = $a,b = $b"; //

print "<br>c = $c";//

echo "<br><a href = 'ppr66.php?a = 1&b = 2&c = 3'*>новая передача</a>"*

?>

</body>

</html>

Результат:

После запуска:http://localhost/My/pr421.php?a =10&b =20&c =30

a = 10,b = 20

c =31

[новая передача](http://localhost/MyPrimers/pr421.htm?a=1&b=2&c=3)

Если нажмем на гиперссылку [новая передача](http://localhost/MyPrimers/pr421.htm?a=1&b=2&c=3), получим

a =1,b =2

c =4  
[новая передача](http://localhost/MyPrimers/pr421.php?a=1&b=2&c=3)

6.6.2. Передача данных по методу POST

Какой из методов POST или GET лучше? В случае динамических сайтов, когда показываемая страничка определяется не только адресом, но и переменными – как в интернет-магазине, лучше использовать метод GET. Для передачи информации, вводимой в форме, в большинстве случаев лучше подходит метод POST.

Предположим, что мы создали форму следующего вида:

<form action = "multi.php" method = "POST">

<input type = "text" NAME = "first" SIZE = "4" MAXLENGTH = "4">

< input type = "text" NAME= "second" SIZE= "4" MAXLENGTH = "4">

< input type = "Submit" VALUE = "Умножить">

</form>

Скрипт, содержащийся в файле multi.php, может выглядеть следующим образом:

<?php

$first = $\_POST['first']; $second = $\_POST['second'];

echo "$first умножить на $second получится ", $first\*$second;

?>

Вывод:

*5 умножить на 6 получится 30.*

Кроме простого задания переменной в форме, PHP позволяет использовать и массивы. PHP воспринимает массивы в контексте переменных формы. Пример:

<form action = "array.php" method = "post">

Имя: <input type = "text" name = "user[name]"><br>

E-mail: <input type = "text" name = "user[email]"><br>

Хобби: <br>

<select multiple name = "hobbi[]">

<option value = "книги">Книги

<option value = "компьютер">Компьютер

<option value = "музыка">Музыка

<option value = "спорт">Спорт

</select>

<input type = "submit" value = "Отправить">

</form>

<?php //array.php – обработчик формы

/\* выводим выбранные в форме переменные \*/

print\_r($\_POST);

?>

Вывод:

### 6.6.3. Ввод данных из различных форм

Рассмотрим, передачу значений элементов из различных форм.

Многострочное поле ввода текста *textarea* позволяет отправлять сразу несколько строк. По умолчанию создается пустое поле шириной в 20 символов и состоящее из двух строк. В атрибутах можно ука­зать ширину поля(cols) и число строк(rows). Атрибут readonly запрещает редактировать, удалять и изменять текст, предназначенный только для чтения.

<form action = "action.php" name = "myform" method = "post">

<textarea name = "msg" cols = "20" rows = ”10” ></textarea>

<input name = "Submit" type = submit value = "Отправить">

</form>

<?php // обработчик формы action.php:

$msg = $\_POST['msg'];

echo $msg;

?>

Флажoк c*heckbox* предлагает ряд вариантов. Группа флажков состоит из элементов <input>, имею­щих одинаковые атрибуты name и type = checkbox. Чтобы элемент был отмечен по умолчанию, необходимо пометить его как checked. Если элемент выбран, то сценарию поступит строка имя = значение, в противном случае в обработчик формы не придет ничего. Переменная $\_POST вернет пустое значение:

<form action = " ppr67.php" method = post>

<input name = "mycolor" type = "checkbox" value = "blue">Синий

<input name = "mycolor" type = "checkbox" value = "black">Черный

<input name = "mycolor" type = "checkbox" value = "white">Белый

<input name = "Submit" type = submit value = "Выбрать">

</form>

<?php //ppr67.php

if (!empty($\_POST['mycolor'])) echo $\_POST['mycolor'];

// Если выбран хоть 1 элемент

else echo "Выберите значение";

?>

Переключатели *radio* предлагают пользователю ряд вариантов, но разрешает выбрать только один из них.

<input name="Имя переключателя" type=" radio" value="Значение">

Атрибут name задает имя переключателя, type задает тип radio, а атрибут value за­дает значение. Если пользователь выберет переключатель, то сценарию будет передана строка имя=значение. Можно указать параметр checked, который указывает на то, что переключатель будет отмечен по умолчанию при загрузке страницы. Переключатели можно объединять в группы, для этого они должны иметь одно и тоже имя.

Если пользователь не выбрал кнопку в наборе радио кнопок, то данный элемент будет отсутствовать в массиве.

<?php//ppr68.php

while(list($key,$val) = each($rdi))

echo "ключ - $key, значение - $val<br>\n";

?>

Выпадающий список *select* представляет собой выпадающий или раскрытый спи­сок, при этом одновременно могут быть выбраны одна или несколько строк. Если в теге <option> указан параметр selected, то строка будет из­начально выбранной. Параметр size задает, сколько строк будет занимать список. Если size равен 1, то список будет выпадающим. Если указан ат­рибут multiple, то разрешено выбирать несколько элементов из списка.

<select name = "Имя списка" size = “Размер” multiple>

<option value = ”Значение”>Отображаемый текст в списке</option>

</select>

При передаче данных выпадающего списка сценарию передается строка имя = значение, а при раскрытом списке передается строка имя = значение1&имя = значение2&имя = значениеN.

Кнопка отправки формы *submit* cлужит для отправки формы сценарию.

<input type = "submit" name = "Имя кнопки" value = "Текст кнопки">

После нажатия на кнопку сценарию передается строка имя = текст кнопки. Кнопка SUBMIT тоже может передавать значение в обработчик. Значение устанав­ливается из атрибута VALUE.

Кнопка для загрузки файлов *browse* cлужит для загрузки файлов на сервер. Начинающий тэг <form> содержит необходимый атрибут encrypt. Атрибут encrypt принимает значение multipart/form-data, который изве­щает сервер о том, что вместе с информацией посылается и файл. При создании текстового поля также необходимо указать тип файла – “file”. Задача: Загрузка файла на сервер

<form enctype = "multipart/form-data" action = "upload.php" method = "post">

Загрузить файл: <input name = "myfile" type = "file">

<input type = "submit" value = "Отправить">

</form>

В данной форме присутствует элемент browse, который открывает диалоговое окно выбора файла для загрузки на сервер. При нажатии на кнопку «отправить», файл передается сценарию-обработчику. Перед написанием обработчика необходимо определиться, в какой каталог будет копироваться файл:

<?php// upload.php

if(isset($\_FILES["myfile"])) // Если файл существует

{

$catalog = "../image/"; // Наш каталог

if (is\_dir($catalog)) // Если такой каталог есть

{

$myfile = $\_FILES["myfile"]["tmp\_name"]; // Времменый файл

$myfile\_name = $\_FILES["myfile"]["name"]; // Имя файла

if(!copy($myfile, $catalog)) echo 'Ошибка при копировании файла '.$myfile\_name // Если не удалось скопировать файл

}

else mkdir('../image/');

// Если такого каталога нет, то мы его создадим

}

?>

Элемент Type =image. В обработчик передаются координаты мыши относительно рисунка: *имя*.x = *значение*&*имя*.y = *значение*. Вместо стандартной кнопки можно использовать изображение с помощью тега:

<input type = "image" src = "image.gif" name = "sub" /> .

Когда пользователь щелкнет на изображении, форма будет передана на сервер с двумя дополнительными переменными – sub\_x и sub\_y. Они содержат координаты нажатия пользователя на изображение.

Пример:

<html><head>

<title>Simpleform.html </title>

</head>

<body>

<form action = "form.php" method = "GET">

Имя:<input type = "text" name = "fio"><p>

<input type = "image" src = "image.gif" name = "sub" />

</form>

</body>

</html>

<?php //form.php

$fio = $\_GET['fio'];

echo "Hello, $fio<br>";

print\_r ($\_GET);

?>

*Hello,Alex  
Array ( [fio] => valera [sub\_x] => 83 [sub\_y] => 64 )*

На поле для ввода пароля type = password символы, набираемые в форме, не будут отображаться на экране. Пример:

<input type = "password" name = "txtName" size = "10" maxlength = "5">

Скрытое текстовое поле *hidden* позволяет передавать сценарию служебную информацию, не отображая еу на странице.

<input name = "email" type = "hidden" value = "spam@nospam.ru">

### 6.7. PHP и сервер баз данных MySQL

MySQL – это многопоточный быстрый сервер баз данных, разрабо­танный компанией TcX. MySQL поддерживает стандарт языка запросов SQL и имеет множество расширений к стан­дарту. MySql-сервер загружается автоматически вместе с загрузкой веб-сервера и может работать сразу с несколькими базами данных:

MySql – сервер: БД1; БД2; БД3; БД4.

Под БД понимается совокупность взаимосвязанных таблиц, объеди­ненных общим управлением. Перед началом работы с MySql надо: спро­ектировать БД; спроектировать структуру таблиц БД; создать таблицы БД; создать БД.

Каждая таблица состоит из записей. В MySql используются сле­дующие типы полей записей:

целые числа: INT(-2147483648..2147483647), TINYINT(-128..127), SMALLINT, MEDIUMINT, BIGINT;

вещественные числа: DOUBLE, REAL(синоним DOUBLE), FLOAT, DECIMAL (числовая строка), NUMERIC;

строки: VARCHAR (1..255), TINYTEXT (1..255), TEXT(1..65535); MEDIUMTEXT, LONGTEXT;

блочно-бинарные данные: BLOB(до 65535 символов), TINYBLOB(до 255 символов), MEDIUMBLOB, LONGBLOB: при по­иске, в отличие от TEXT, учитывается нижний и верхний регистр симво­лов;

дата и время: DATE(гггг-мм-дд), TIME(чч:мм:сс), DATETIME(гггг-мм-дд чч:мм:сс ), TIMESTAMP.

К полям могут присоединяться модификаторы: primary key, auto\_increment, default.

Создание БД и таблиц может быть выполнено вручную, например c помощью приложения phpmyadmin, постовляемого вместе с PHP и MySql. Другой способ создание БД и таблиц состоит в использовании ко­манд языка SQL, являющегося частью MySql.

*Команды SQL для создания баз данных и таблиц*

CREATE DATABASE db\_name – создать базу данных;

CREATE TABLEtable\_name(Имя\_поля1 тип1, Имя\_поля2 тип2) – соз­дать таблицу;

DROP DATABASE db\_name – удалить базу данных;

DROP TABLE table\_name **–** удалить таблицу;

ALTER DATABASE db\_name – модифицировать базу данных;

ALTER TABLEtable\_name **–** модифицировать таблицу;

INSERT INTO table\_name(Имя\_поля1 тип1, Имя\_поля2 тип2…) val­ues(‘val1’,’val2’,…) – вставка записи со значениями полей val1, val2, … ;

DELETE FROM table\_name WHERE-выражение – удаление записи, для которой выполнено выражение;

UPDATE table\_nameSET(Имя\_поля1 ‘val1’, ‘Имя\_поля2 val2’…) Where-выражение – обновить таблицу.

Порядок работы с SQL-запросами следующий: cначала создается строка $sql, содержащая SQL-запрос. Затем эта строка передается для выполнения функции mysql\_query($sql). Пример:

mysql\_connect(); //соединяемся с сервером

$sql = "create database ‘mydb’"; // создаем базу данных mydb:

mysql\_query($sql);//выполнение директивы SQL

//Cоздаем таблицу:

$sql = "CREATE TABLE userstable

(name VARCHAR(25), email VARCHAR(25), choice VARCHAR(15)) "

mysql\_query($sql); );//выполнение директивы SQL

Сначала выполняется соединение с сервером MySQL функцией:

int mysql\_connect([$host],[$username],[pass­word]);

Здесь host – имя хоста (по умолчанию localhost); username – имя; password – пароль. Функция возвращает параметр соедине­ния id или 0, если соединение не прошло. После этого саздается база данных и таблица.

*Выборка данных из БД.* Выборка данных из существующей базы начинается с создания соединения с сервером MySQL:

mysql\_connect(); //или mysql\_pconnect().

Затем выбирается база данных для работы:

mysql\_select\_db($db\_name[, int link\_identifier]);.

Здесь link\_identifier – необязательный параметр, содержащий id-соединение, возвращаемое из mysql\_connect(). Если параметр отсутствует, выбирается последнее открытое соединение. Функция возвращает true или false.

Выполнить запрос к базе данных:

mysql\_query(string $query[, int $link\_identifier]);

Здесь $query – строка, содержащая SQL-запрос. Функция возвращает ID-результат или 0, если ошибка.

Закрыть соединение с MySQL:

mysql\_close(int link\_identifier).

Функция возвращает значение true или false. Пример:

<html><head>

<title>Запрос информации</title>

<body>

Введите Ваши данные <p>

<table width = 400><tr><td align = right>

<form action = "ppr70.php" method = "post">

Ваше имя:<BR>

<input type = "text" name = "name" size = "20" maxlength = "30"><P>

Ваш email:<BR>

<input type = "text" name = "email" size = "20" maxlength = "30"><P>

<input type = "submit" value = "Отправить запрос!">

</form>

</td></tr></table>

</body></html>

<?php // ppr70.php – скрипт создает БД и таблицу

$dbName = 'mydb';

/\* создать соединение с сервером\*/

$link = mysql\_connect( "localhost" , "root" ) or exit("Could not connect");;

/\* уничтожить старую БД \*/

/\*$sql = sprintf("DROP DATABASE %s ",$dbName);

if (mysql\_query($sql, $link)) {

echo "Database drop successfully\n";

} else { echo 'Error drop database: ' . mysql\_error() . "\n"; } \*/

/\* создать новую БД, если БД не существует\*/

$sql = "create database $dbName";

if (mysql\_query($sql, $link)) {//Выполнение директивы SQL

echo "Database created successfully\n";

} else { echo 'Error creating database: ' . mysql\_error() . "\n"; }

//Cоздаем таблицу: mydb.clients

$tablename = 'clients' ;

$sql=sprintf("CREATE TABLE %s.%s (

`name` VARCHAR( 25 ) NOT NULL , `email` VARCHAR( 25 ) NOT NULL ) ENGINE = MYISAM ",$dbName,$tablename);

if (mysql\_query($sql, $link)) {

echo "Table created successfully\n";

} else { echo 'Error creating Table: ' . mysql\_error() . "\n"; }

$name = $\_POST['name'];

$email = $\_POST['email'];

// вставка записи со значениями полей val1, val2,

$sql=sprintf("INSERT INTO %s.%s values('$name', '$email')", $dbName, $tablename);

if (mysql\_query($sql, $link)) {

echo "Table insert successfully\n";

} else { echo 'Error insert Table: ' . mysql\_error() . "\n"; }

mysql\_close($link); /\* Закрыть соединение \*/

$link = mysql\_connect( "localhost","root" )or die("Не могу создать соединение ");

//Выбрать БД

$result = mysql\_select\_db("$dbName",$link) or die("Не могу выбрать базу ");

//Создать SQL-запрос

$sql = sprintf("Select \* from %s", $tablename);

$result = mysql\_query($sql,$link);

if (!$result) { die('Invalid query: ' . mysql\_error()); }

$myrow = mysql\_fetch\_array($result); //установка первой записи

echo"<br>", $myrow['name'];//вывод поля “name” первой записи

echo $myrow['email'];

mysql\_close($link);/\* Закрыть соединение \*/

?>

Язык SQL был специально раз­работан для за­просов и получения данных из таблиц БД. Класси­ческий пример – использо­вание условия WHERE в синтаксисе SQL.

<?php

$statement = "SELECT name, phone FROM samp\_table";

$statement .= " WHERE id='5'";

Данный код более эффективен, чем поиск по БД с использованием PHP. Для сортировки результатов рекомендуется применять синтаксис SQL ORDER BY, а не PHP-функцию ksort().

<?php

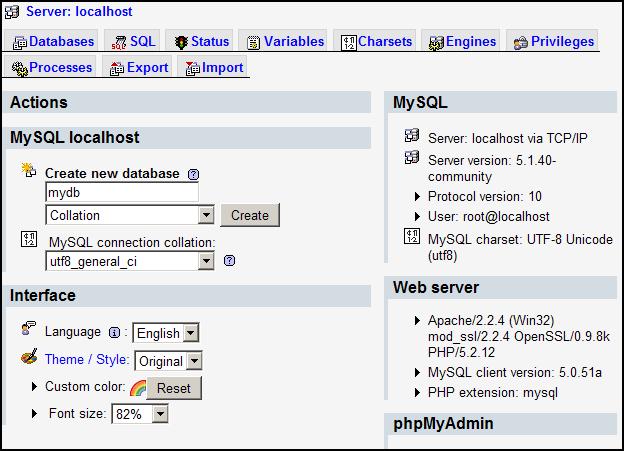
$statement = "SELECT name, email, phone FROM some\_table ";

$statement .= "WHERE name IS LIKE '%baggins' ORDER BY name";

# 6.7.1. Утилита Phpmyadmin

Программное средство phpmyadmin представляет удобные инструменты для создания баз данных и таблиц. Вызов утилиты зависит от конфигурации веб-сервера, например: [localhost/tools/phpmyadmin](http://www.localhost/phpmyadmin).

С помощью phpmyadmin часто вручную создают базу данных и устанавливают права доступа. Затем с помощью этой же утилиты можно создать структуру таблиц и сами таблицы (рис. 6.1.).



*Рис. 6.1.Создание базы и таблицы с помощью phpMyAdmin*

Выборка данных из таблиц БД

Выборка данных из таблиц базы производится SQL-командой SELECT в соответствии с заданным условием Where.

$result = mysql\_query("SELECT …WHERE…");

Для обработки данных, на которые указывает дескриптор result, имеется ряд функций. Функция mysql\_num\_rows(int result) возвращает количество строк в результате запроса. Функция mysql\_result(result, i, column) возвращает значение поля в столбце column и в строке i. Функция mysql\_fetch\_row($result)) возвращает массив, соответствующий очередной записи таблицы. Функция mysql\_fetch\_array(result) возвращает результаты запроса в виде ассоциированного массива. Пример:

<?php//ppr71

$db = mysql\_connect("localhost", "root");

mysql\_select\_db("mydb",$db);

$result = mysql\_query("SELECT \* FROM employees",$db);

$str = mysql\_result($result,0,"first");

printf("First Name: %s<br>\n",$str);

$str = mysql\_result($result,0,"last");

printf("Last Name: %s<br>\n",$str);

$str = mysql\_result($result,0,"address");

printf("Address: %s<br>\n",$str);

$str = mysql\_result($result,0,"position");

printf("Position: %s<br>\n",$str);

mysql\_close($db);

?>

Для исполнения SQL запроса служит функция mysql\_query(). Результаты выполнения запроса помещается в переменную $result. С помощью функции mysql\_result() извлекаем из переменной $result первую, вторую и т.д. записи. Перебирая записи, выберем все, что хранятся в БД. Рассмотрим более универсальные функции выборки данных mysql\_fetch\_row() и mysql\_fetch\_array().

<html> <body>

<?php//ppr72

$db = mysql\_connect("localhost", "root");

mysql\_select\_db("mydb",$db);

$result = mysql\_query("SELECT \* FROM employees",$db);

echo "<table border = 1>\n";

echo "<tr><td>Name</td><td>Position</tr>\n";

while ($myrow = mysql\_fetch\_row($result))

{

printf("<tr><td>%s %s</td><td>%s</td></tr>\n", $myrow[1], $myrow[2], $myrow[3]);

}

echo "</table>\n";

?>

</body> </html>

Цикл while() выполняется, пока в массиве $result остается запись для выборки, которую необходимо извлечь с помощью функции mysql\_fetch\_row(). Функция выбирает из массива строку, записываемую в $myrow и автоматически переходит на следующую строку. При достижении конца массива mysql\_fetch\_row() вернет значение false. Элементы массива $myrow нумеруются с 0.

Рассмотрим использование функция mysql\_fetch\_array(), с помощью которой можно обращаться к каждому полю массива по имени. Например, если раньше для получения имени нам приходилось писать $myrow[1], то теперь мы можем писать $myrow["first"] ("first" – название столбца в массиве). Пример:

<?// ppr73.php – скрипт читает из БД

$dbName = "mydb";

/\* создать соединение \*/

$link = mysql\_connect("localhost", "Alex", "12345")

or exit("Could not connect");

mysql\_select\_db("$dbName",$link) or die("Не могу выбрать базу данных ");

$sqlq = "Select \* from students"; //создать SQL-запрос

$result = mysql\_query($sqlq,$link) or die ("Invalid query");

$myrow = mysql\_fetch\_array($result); //Установка первой записи

echo $myrow["name"];//Вывод поля “name” первой записи

do//цикл просмотра всех записей

{

echo “сотрудник-”, $myrow[‘id’];

echo $myrow["name"];

}

while($myrow=mysql\_fetch\_array($result));

mysql\_close();/\* Закрыть соединение \*/

?>

Существует несколько способов передачи параметров запроса от клиента к серверу: первый – использовать метод GET или POST для передачи данных из формы; второй – набрать параметры прямо в адресной строке браузера при использовании метода GET; третий – вставить параметры в обычную ссылку при использовании метода GET:

<a href = "http://my\_machine/mypage.php3?id = 1">

При передаче данных от клиента к серверу часто используется переменная $PHP\_SELF, в которой хранится имя и URL текущей страницы - ссылка на сам PHP-скрипт по URL. Вместо $PHP\_SELF в настоящее время используется переменная $\_SERVER['PHP\_SELF']. Пример:  
echo "<form action=\"".$\_SERVER['PHP\_SELF']."\" method=\"get\">";  
В данном случае мы хотим через ссылку вызвать страницу из нее самой. Затем с помощью if-else будем переводить стрелки с одного кода на другой, гоняя страницу по кругу. К имени самой страницы в ходе работы добавлена информация: переменные и их значения. При втором проходе страницы программа отреагирует на эти пары name = value. В данном случае проверим, есть ли переменная $id, и в зависимости от результата выполним тот или иной код. Вот как это будет выглядеть:

<html><body>

<?php//ppr74

$db = mysql\_connect("localhost", "root");

mysql\_select\_db("mydb",$db);

if ($id)

{

$result = mysql\_query("SELECT \* FROM employees WHERE id = $id",$db);

$myrow = mysql\_fetch\_array($result);

printf("First name: %s\n<br>", $myrow["first"]);

printf("Last name: %s\n<br>", $myrow["last"]);

printf("Address: %s\n<br>", $myrow["address"]);

printf("Position: %s\n<br>", $myrow["position"]);

}

else

{

// show employee list

$result = mysql\_query("SELECT \* FROM employees",$db);

if ($myrow = mysql\_fetch\_array($result))

{

// display list if there are records to display

do

{

printf("<a href = \"%s?id=%s\">%s %s</a><br>\n", $\_SERVER['PHP\_SELF'], $myrow["id"], $myrow["first"], $myrow["last"]);

}

while ($myrow = mysql\_fetch\_array($result));

}

else

{ echo "Sorry, no records were found!"; }

}

?>

</body></html>

**Сохранение данных в базе данных.** Введем информацию из формы и сохраним в базе данных. Для удобства название каждого элемента формы совпадает с названием поля в базе данных. Помимо этого мы присвоили имя кнопке submit, чтобы затем проверить, есть ли значение переменной $submit. Таким образом можно узнать, вызыва­ется страница в первый или во второй раз. Все пере­менные, передаваемые странице, будут выводиться на экран и в базу данных с помощью переменной $\_POST. Теперь возьмем переданную через форму информацию и внесем ее в базу данных.

<html><body>

<?php //ppr75

if (isset($\_POST['submit'] )){

// process form

while (list($name, $value) = each($\_POST)) {

echo "$name = $value<br>\n";

}

$db = mysql\_connect("localhost", "root");

mysql\_select\_db("mydb",$db);

$sql = "INSERT INTO employees (first,last,address,position) VALUES

('$first','$last','$address','$position')";

$result = mysql\_query($sql);

echo "Thank you! Information entered.\n";

} else{

// display form

?>

<form method = "post" action = "<?php echo $\_POST['$PHP\_SELF']?>">

First name:<input type = "Text" name = "first "><br>

Last name:<input type = "Text" name = "last "><br>

Address :<input type = "Text" name = "address "><br>

Position :<input type = "Text" name = "position"><br>

<input type = "Submit" name = "submit" value = "Enter information">

</form>

<?php

} // end if

?>

</body></html>

Мы внесли данные в базу. Теперь научимся менять записи, которые уже находятся в таблице. Редактирование данных сочетает в себе два кода: извлечение данных из базы с выводом их на экран и внесение данных через форму обратно в базу.

<html><body>

<?php//ppr76

$db = mysql\_connect("localhost", "root");

mysql\_select\_db("mydb",$db);

if (isset($\_POST['submit'] )){

// here if no ID then adding else we're editing

if (isset($\_POST['id'] )){

$sql = "UPDATE employees SET first = '$first',last = '$last',

address = '$address',position = '$position' WHERE id=$id";

} else {

$sql = "INSERT INTO employees (first,last,address,position) VALUES

('$first','$last','$address','$position')";

}

// run SQL against the DB

$result = mysql\_query($sql);

echo "Record updated/edited!<p>";

} elseif ($delete) {

// delete a record

$sql = "DELETE FROM employees WHERE id = $id";

$result = mysql\_query($sql);

echo "$sql Record deleted!<p>";

} else {

// this part happens if we don't press submit

if (isset($\_POST['id'] )){

// print the list if there is not editing

$result = mysql\_query("SELECT \* FROM employees",$db);

while ($myrow = mysql\_fetch\_array($result)) {

printf("<a href = \"%s?id = %s\">%s %s</a> \n", $\_POST["$PHP\_SELF"], $myrow["id"], $myrow["first"], $myrow["last"]);

printf("<a href = \"%s?id = %s&delete = yes\">(DELETE)</a><br>", $\_POST["$PHP\_SELF"], $myrow["id"]);

}

}

?>

<P>

<a href = "<?php echo $\_POST["$PHP\_SELF"]?>">ADD A RECORD</a>

<P>

<form method = "post" action = "<?php echo $\_POST["$PHP\_SELF"]?>">

<?php

if (isset($\_POST['id'] ))

{

// editing so select a record

$sql = "SELECT \* FROM employees WHERE id = $\_POST['id'] ";

$result = mysql\_query($sql);

$myrow = mysql\_fetch\_array($result);

$id = $myrow["id"];

$first = $myrow["first"];

$last = $myrow["last"];

$address = $myrow["address"];

$position = $myrow["position"];

// print the id for editing

?>

<input type = hidden name = "id" value = "<?php echo $id ?>">

<?php

}

?>

First name:<input type = "Text" name = "first" value = "<?php echo $first ?>"><br>

Last name:<input type = "Text" name = "last" value = "<?php echo $last ?>"><br>

Address:<input type = "Text" name = "address" value = "<?php echo $address ?>"><br>

Position:<input type = "Text" name = "position" value = "<?php echo $position ?>"><br>

<input type = "Submit" name = "submit" value = "Enter information">

</form>

<?php

}

?>

</body></html>

Программа делится на три части. Первое if()-выражение проверяет, была ли нажата кнопка Submit, и если была, проводится ли проверка, есть ли в поданных данных переменная $id. Если ее нет, значит происходит добавление новой записи. В противном случае редактируем уже существующую запись. Далее проверяем, определена ли переменная $delete. Если да, удаляем запись. Наконец, переходим к действию, которое будет выполняться по умолчанию: выводим просто список служащих и форму. Здесь мы опять проверяем существование переменной $id. Если она существует, мы просим базу данных выдать сведения о выбранном служащем. В противном случае выводим пустую форму. Так можно передавать информацию от одной страницы к другой через URL с помощью ссылок и через формы.

### Работа с MySQL занесение и получение данных из базы данных

В следующем примере показано как в PHP легко обрабатывать данные с HTML-форм.

<html><head>

<title>Запрос информации</title>

<body>

<center> Хотите знать о наших товарах? <p>

<table width = 400><tr><td align = right>

<form action = "ppr77.php" metod = "post">

Ваше имя:<BR>

<input type = "text" name = "name" size = "20" maxlength = "30">

<P>

Ваш email:<BR>

<input type = "text" name = "email" size = "20" maxlength = "30">

<p>

Меня интересуют:

<select name = "preference">

<option value = "Компьютеры">Компьютеры

<option value = "Автомобили">Автомобили

</select>

<p>

<input type = "submit" value = "Отправить запрос!">

</form>

</td></tr></table></center>

</body>

</html>

Назовем этот файл php28.htm. В нем мы указали, что данные формы будут обрабатываться файлом ppr77.php.

<?// ppr77.php скрипт получает переменные из php428.htm

$hostname = "localhost";

$username = "myusername";

$password = "mypassword";

$dbName = "my\_db";

/\* Таблица MySQL, в которой хранятся данные \*/

$userstable = "clients";

/\* создать соединение \*/

mysql\_connect() or die("Не могу создать соединение ");

mysql\_select\_db("$dbName") or die("Не могу выбрать базу данных ");

print "<CENTER>";

print "Привет, $name.";

print "<BR><BR>";

print "Спасибо за ваш интерес.<BR><BR>";

print "Вас интересуют $preference. Информацию о них мы пошлем вам на email: $email.";

print "</CENTER>";

/\* Отправляем email \*/

mail($email, "Запрос на информацию", "$namen\n

Спасибо за ваш интерес!\n

Вас интересуют $preference\n. Обратитесь в ближайший филиал нашей компании\n");

/\* Вставить информацию о клиенте в таблицу \*/

$query = "INSERT INTO $userstable VALUES('$name','$email', '$preference')";

$result = MYSQL\_QUERY($query);

print "Информация о вас занесена в базу данных.";

/\* Закрыть соединение \*/

mysql\_close();

?>

Вывод:

Привет, valera.

Спасибо за ваш интерес.

Вас интересуют Компьютеры. Информацию о них мы пошлем вам на email: Bender@bsu.by.

Информация о вас занесена в базу данных.

Вот так легко можно работать с базой данных в PHP. Теперь, кроме письменных уведомлений, информация о клиенте и его интересах будет заноситься в таблицу MySQL.

?>

### 6.8. Работа с E-mail

E-mail посылается с сервера веб через службу протокола пересылки почты SMTP. В PHP для отправки e-mail используется функция

mail( $to,$subj, $msg[, $headers[, ad\_par]]);

Функция mail() отправляет сообщение $msg получателю $to. Разде­лив запятой адреса, можно задать несколько получателей, например $to = 'user@ex.com, anouser@ex.com'. Переменная $subj указывает тему, headers содержит необязательные заголовки, в число которых входит адрес отправителя. С помощью mail() можно высылать сообщения с присоединенными файлами. Функция возвращает TRUE, если почта была успешно принята для доставки, и FALSE – в ином случае. Пример:

<?php//ppr78

$to = 'address@domain.com';

$subject = 'HelloMail’;

$message = 'hello';

$headers = 'From: master@example.com' . "\r\n" .

'Reply-To: master@example.com' . "\r\n" .

'X-Mailer: PHP/' . phpversion();

if(mail($to, $subject, $message, $headers))

{ print("Yes");}  
else{ print("No"); };

?>

Функция mail() не отправляет сообщение, а обрашается к почтовому агенту, реализованному в функции sendmail(). В системе Windows при использовании localhost sendmail() представляет псевдопочтового агента, который помещает почту в файл.

Приведенная ниже форма HTML поволяет пользователю вводить имя и адрес. Когда нажимается кнопка "Registration", данные формы передаются приложению email.php как переменные $\_POST[]:

<html><head>

<title>ppr79.htm</title>

</head>

<body>

<FORM NAME = "testForm" ACTION = "ppr80.php" METHOD = "POST">

name: <INPUT TYPE = "text" NAME = "fname"><BR>

e-mail: <INPUT TYPE = "text" NAME = "email"><BR>

<INPUT TYPE = "SUBMIT" value = "Registration">

</form>

</body></html>

<?php // ppr80l.php

$to = $\_POST['email'];

$subject = "Подтверждение регистрации";

$msg = "Дорогой: " . $\_POST['fname'] . " " . ",\n\n";

$msg .= "Вы успешно зарегистрировались.";

$headers = "From: Registration Site <myaddr@mydomain.com>";

$headers .= "Reply-To: registration@mydomain.com";

if(mail($to, $subject, $msg, $headers))

{

echo "$msg". "Message sent! ,". $\_POST['fname'];

}

else { echo "Message not sent!"; }

?>

Дорогой: Alex , Вы успешно зарегистрировались.

Message sent! , Alex

### 6.9. Cookies-наборы

РНР не имеет средств для запоминания предыдущих запросов клиента к серверу. Если необходимо создать веб-сайт, запоминающий посетителей и использующий их данные для создания персонального содержимого, решение мо­жет быть в использовании cookies. Cookies – это механизм хранения данных о предыдущем по­сещении сайта на компьютере клиента. При первом посещении сайта сервер передает браузеру клиента для сохранения в файле идентифицирующий клиента cookie-набор. При следующем посещении данного сайта, браузер вместе с запросом перешлет этот cookie-файл обратно серверу. Cookie хранятся в системном каталоге Cookies в текстовых файлах формата user­name@host.txt.

Когда серверное приложение намеревается установить cookie-набор, происходит следующая последовательность событий: Сервер посылает клиенту в HTTP-ответе заголовок Set-cookie, содержащий данные о клиенте, которые он хочет сохранить.

НТТР/1.0 200 ОК

[Здесь опять идут заголовки]

Set-Cookie: user\_id=265; domain=bsu.by ;

Expires=Mon, 20-Sep-2010 16:54:56 GMT;Path=/

Клиент сохраняет эту информацию в cookie-наборе, существующем как маленький текстовый файл, содержащий значения для данного сайта. При последующих запросах этого сайта браузер вспомнит: «есть cookie-набор для этого сайта» и отправит эту информацию в виде:

GET /index.php HTTP/1.0

[Злесь идут заголовки]

Cookie: user\_id=265

Синтаксис HTTP-заголовка Set-Cookie:

Set-Cookie: NAME = VAL; expires = DATE; path = PATH; domain = DNAME; secure

Параметр NAME-имя cookie-набора, VAL – URL-кодированное значение. Параметр DATE задает дату и время, когда cookie потеряет актуальность. Параметр secure указывается, если cookie посылается через защищенное HTTP-соединение. Параметры PATH и DNAME представляют путь и домен, для которых предназначен cookie-набор. В качестве значения параметра DNAME используется не просто my.org, а .my.org (с ведущей точкой). Это еще одна гарантия от злоупотребления cookie. Существует несколько доменов верхнего уровня, для которых в имени домена достаточно двух уровней: .СОМ, .EDU, .NET, .ORG, .GOV, .MIL, .INFO. Все другие домены требуют минимум трех точек в DNAME.

Значение cookie можно изменить, послав дополнительный заголовок Set-Cookie с новым значением. Для удаления достаточно изменить текущее значение времени так, чтобы набор потерял актуальность в прошлом.

Set-Cookie: my = myvl; expires = Tuesday, 03-Dec-2012 13:01:59 GMT; path = /; domain = .co.org;

Здесь был установлен cookie-набор my со значением myvl для всех файлов в домене co.org. Этот cookie-набор установлен с граничной датой актуальности декабря 2012 года (03-Dec-2009) и временем 13:01:59 GMT.

Посмотрим другие возможности реализации cookie-наборов, например использование HTML-дескрипторов. Дескриптор <МЕТА> можно использовать для имитации HTTP-заголовков внутри HTML-страницы, применяя атрибуты HTTP-EQUIV и CONTENT.

<head>

<title><!--HTML-код --> </title>

<МЕТА HTTP-EQUIV="Set-Cookie" CONTENT="mycookie=myvalue;

path=/; domain=.bsu.by">

</head>

<!– остаток HTML-документа //-->

Вторым методом установки cookie-набора в PHP-коде является создание заголовка Set-Cookie с помощью PHP-функции header().

<?php

header("Set-Cookie:mycookie = myvl; path = /;domain = .cg.org");

?>

В связи с тем, что функция используется для отправки HTTP-заголовков, она должна быть выполнена до пересылки содержимого с помощью операторов echo или print.

Третий и наиболее известный метод установки cookie-наборов – использование PHP-функции setcookie ():

setcookie($name [,$value [, $expire [, $path [, $domain (, $secure]]]]]);

Поскольку и для заголовков header() перед фунцией setcookie () не должно быть ника­кого вывода, даже пустая строка перед открывающим дескриптором <?php > вызовет ошибку. Рассмотрим пример:

<?php //cooke0.php

$userid = $\_COOKIE['userid'];

setcookie($userid, 12);

print\_r($\_COOKIE);

?>

Вывод:

Array ( [First] => [vc] => 24 )

<?php //cooke1.php

$username = $\_POST['userid'];

if(isset($\_COOKIE[$username]))

{ $userid = $\_COOKIE[$username];

echo "id1 = ",$userid,"<br>"; }

else {

$userid = $username;

setcookie($userid, "value", mktime(0,0,0,1,22,2011),"/");

$userid = $\_COOKIE[$username];

echo "id2 = ",$userid,"<br>"; }

//echo "id = ",$userid,"<br>";

print\_r($\_COOKIE);

?>

Вывод:

id1 = value  
Array ( [First] => [vc] => 24 [value] => [Mycooke] => value )

Следует помнить, что cookie-наборы должны предшествовать началу документа, пересылаемого по HTTP. Если нельзя установить cookie-набор перед отправкой части документа, можно использовать функцию буферизации ob\_start(), которая позволит устанавливать cookie-набор после вывода содержимого документа.

<?php

ob\_start();

echo "Hello\n";//вывод в буфер

setcookie("cookiename", "cookiedata");

ob\_end\_flush();//Вывод буфера

?>

Для удаления cookie-набора можно задать параметр $value=NULL.

<?php

setcookie("mycookie", NULL, NULL, "/", ".bsu.by");

?>

Для доступа к значению cookie-набора, полученного от браузера, РНР использует суперглобальный массив $\_COOKIE, который хранит значения cookie-наборов. Каждый ключ (имя переменной cookie-набора) в этом массиве представляет отдельный cookie-набор. Обращение к функции setcookie() было быть помещено до любого вывода, включая теги и символы-разделители. Если setcookie() выполняется успешно, то возвращается TRUE, однако это еще не означает, что пользователь принял cookie. Проверка поддержки cookie-наборов из РНР рассматривается в следующем примере:

<?php

if(!isset(\_GET[‘testcookie'])) {

setcookie("testcookie", "test\_value");

header("Location: $\_SERVER["PHP\_SELF"])?testcookie = l");

exit; }

else {

if(isset($\_COOKIE['testcookie'])) {

setcookie("testcookie");

echo "Прием cookie-наборов включен"; }

else {

echo "cookie-наборы не поддерживаются!";

}

}

?>

В первом операторе if делается попытка создать cookie-набор и затем перенаправить браузер на другую страницу с помощью дополнительного параметра GET. После повторного выполнения сценария с дополнительным параметром проверяется существование cookie-набора. Если клиент не поддерживает cookie-наборы, тестовый cookie-набор уничтожается вызовом функции setcookie () без параметров.

### 6.10. Сессии (Сеансы)

Сессии TCP состоят из трех этапов: установки соединения, передачи данных, завершения соединения. Говорят, что сеансы используют кон­цепцию, аналогичную с парковкой автомобилей, поскольку сеансу на­значается идентификатор сеанса («квитанция»), который затем присутст­вует в каждом запросе. При старте сессии назначается идентификатор SID-номер вроде "db17e8ab5892a486f60f208487152348". Идентификатор сессии сохраняется в cookies клиента PHPSESSID, кроме этого на сер­вере создается временный файл для хранения переменных сессии /php5/tmp/sess\_ db17e8ab5892a486f60f208487152348. Информацию о сессиях можно хранить и в базе данных.

Если требуется, чтобы внутренние страницы сайта "запоминали" имя пользователя и пароль, которые он вводит на главной странице, эта информация должна сохраняется в переменных сеанса в указанном временном файле на сервере. Переменные сессий хранятся в глобальном массиве PHP $\_SESSION.

Пример:

<?

// файл 1.php

session\_start();

$\_SESSION['asd'] = 'aaaaaa';

$\_SESSION['name']['qwe'] = 'тест';

?>

<a href = "2.php">тест</a>

<?

// файл 2.php

session\_start();

print $\_SESSION['asd'];

print $\_SESSION['name']['qwe'];

?>

В первом файле в сессии будут помещаться переменные, а во втором эти переменные будут получаться из сессии. Для начала работы с сессиями, надо запустить их командой session\_start();. После этого появляется доступ к глобальному массиву $\_SESSION. Если номер сессии нельзя передать в cookie браузера (куки отключены), то SID будет передаваться в ссылках URL, к которым будет добавляться. Например, test.ru/index.php?PHPSESSID=db17e8ab5892a486f60f208487152348.

В РНР существует несколько способов создания сеанса. Первый метод заключается в прямом указании РНР начать сеанс, используя функцию session\_start(), которая не принимает параметров и не возвращает значения. Когда эта функция вызывается, все переменные, ассоциированные с данным сеансом, реконструируются или создаются заново. Второй подход заключается в вызове функции sessionreadonly() вместо session\_start (). При этом все сеансовые переменные создаются заново, любые изменения переменных по окончании выполнения сценария не сохраняются.

Система PHP имеет для поддержки сеансов суперглобальный массив $\_SESSION[], который содержит зарегистрированные в данный момент переменные сеанса.

Сеансы существуют, пока не выполняется одно из событий: закры­вается окно браузера; завершается заданное на сервере максимальное время сеанса; используется функция PHP session\_destroy(). Эта функция не принимает параметры и уничтожает любые cookie-наборы и данные, ассоциируемые с активным сеансом.

##### Отмена регистрации сеансовых переменных

Сеансовые переменные можно удалить, уничтожив весь сеанс. Для удаления отдельных переменных можно использовать оператор unset(), удаляющий элемент из суперглобального массива $\_SESSION, или функцию session\_unregister(name). В PHP имеется также функция header("Location: http://www.domain.com"), которая используется для перенаправления браузера на страницу, определенную параметром Location во время осуществления сеансов.

Для доступа к конфиденциальному сайту переменная сеанса может создаваться только после успешного ввода пользователем своего имени и пароля. В этом случае переменная сеанса содержит значение, которое передается со страницы на страницу с указанием, что пользователь имеет право доступа к ресурсам, связанным с сайтом. Когда пользователь выходит из сессии, переменная сеанса должна возвращаться в исходное со­стояние. Это выполняется с помощью функции session\_destroy().

Пример:

<?php // login.php

if ($\_POST['submit'] == "Login")

{

// Сценарий проверки имени пользователя и пароля должен //кодироваться здесь, если аутентификация прошла успешно

session\_start();

$\_SESSION['access'] = "yes";

header(Location:access.php);

}

if ($\_POST['submit'] == "Log Out")

{

//Если пользователь решает выйти

session\_destroy();

}

?>

<?php // access.php

// если пользователь обращается к этой странице,

// проверьте, что он был аутентифицирован в login.php

if ($\_SESSION['access'] != "yes")

{

header(Location:login.php);

}

?>

На странице Login.php создается сеанс, чтобы гарантировать, что пользователь был аутентифицирован. После ввода имени и пароля, выполняют щелчок на кнопке "Login". Сценарий PHP проверяет, будут ли пароль и имя пользователя допустимы. Затем переменной сеанса присваивается значение «yes» и доступ предоставляется. Пользователь перенаправляется на страницу access.php. Сценарий на access.php проверяет, существует ли сеанс. Если сеанс не существует, применяется функция header(), чтобы перенаправить пользователя назад на страницу login.php.

Страница login.php содержит второй сценарий, который используется для уничтожения переменной сеанса, вызывая функцию session\_destroy() щелчком на кнопке "Log Out".

Приложения веб, которые используют данные сеанса, могут быть доступны одновременно множеству пользователей. Поскольку id является уникальным для каждого пользователя, его можно применять для идентификации пользователей, не создавая при этом индивидуальных имен пользователей и паролей. Значение сеанса id хранится в подкаталоге PHP /sessiondata и может быть извлечено с помощью функции session\_id(). В РНР сеансы работают с переменными внутри глобальной области. Это означает, что для регистрации переменной в функции, эта переменная должна быть объявлена как гло­бальная с помощью оператора global.

При работе с сеансами часто необходимо определить, были ли заре­гистрированы сеансовые переменные. Это можно сделать с помощью оператора isset, позволяющего проверить существование соответствую­щего ключа в массиве $\_SESSION.

Поскольку иногда формат, предлагаемый $ID, может быть неприем­лемым при передаче информации о сеансе через HTML-форму, в РНР предусмотрены функции session\_name() и session\_id(), которые возвращают, соответственно, имя сеанса и его идентификатор. Независимо от применяемого метода, идентификатор сеанса должен использоваться каждый раз, когда URL ссылается на внутренний ресурс. Например, при применении гиперссылки прекрасно работает константа $ID:

<А HREF = "checkout.php?<?php echo $ID; ?> ">Перейти к окончатель­ному расчету</А>

С другой стороны, при работе с HTML-формами, идентификатор се­анса передается с помощью скрытых элементов формы<FORM ACTION = "order.php" METHOD = GET>

<INPUT TYPE = "hidden" NAME = "<?php echo session\_name(); ?>"

VALUE = "<?php echo session\_id(); ?>">

<!– Остальной HTML-код формы //—> </FORM>

При передаче идентификатора сеанса важно понимать, что это нужно делать, только если URL относится к локальному веб-серверу. Для предотвращения негативных последствий, связанных с безопасностью, идентификатор сеанса никогда не следует передавать во внешних URL.

### 6.11. Сериализация объектов

Сериализация – это процесс, посредством которого массив или объект класса преобра­зуется в символьную строку для дальнейшей передачи или хранения. Сериализация вы­полняется следующим образом: serialize($obj);.

Пример:

<?php //pr30

$a = array("key" =>"testing", 1 => 10, 2 => "mystring");

echo $b = serialize($a) ,"<br>";

print\_r (unserialize($b));

?>

Вывод :

a:3:{s:3:"key";s:7:"testing";i:1;i:10;i:2;s:8:"mystring";}

Array ( [key] => testing [1] => 10 [2] => mystring )

Строка сериализации не может непосредственно использоваться для передачи с по­мощью HTTP-протокола или для сохранения в базе данных, так как в ней могут содержаться недо­пустимые символы. Если информацию нужно сохранить в базе данных, то следует применить функцию addslashes(). При работе с HTTP-протоколом нужно использовать функцию urlencode(). После сериализации и кодирования строка может быть передана в базу данных как скрытый элемент HTML-формы или записана в файл для дальнейшего использования.

Для восстановления переменной из ее сериализованного представления в РНР преду­смотрена функция unserialize ():

unserialize($obj\_string [, $callback\_function])

где $obj\_string – строка сериализации для восстановления переменной; $callback\_function – имя необязательной функции обратного вызова для использования, если unserialize () реконструирует объект, который не был определен. При успешном за­вершении функция unserialize() возвращает реконструированный объект, или же false, если РНР не может реконструировать сериализованные данные.

##### Сериализация объектов в сессиях

Функция serialize() возвращает строку, содержащую представление объекта в виде по­тока байтов. При сериализации объекта PHP сохраняет в строке имя класса и значения свойств объекта. Методы объекта и данные о родительских классах или имплементируе­мых интерфейсах не сохраняются. Чтобы иметь возможность десериализации объекта, должен быть определен класс этого объекта, т. е. если у вас имеется объект $a класса A в файле page1.php и вы его сериализуете, то получите строку, которая ссылается на класс A и содержит все значения переменных, содержащихся в $a. Если нужно выполнить десериализацию в файле page2.php, воссоздав объект $a класса A, определение класса A должно быть в page2.php. Это можно сделать, например, сохранив определение класса A в include-файле и подключив этот файл и в page1.php, и в page2.php.

//classa.php:

<?php

class A

{

private $one = 1;

public function show\_one()

{

echo $this->one;

}

}

?>

<?php

//pr31.php:

include 'classa.php';

$a = new A;

$s = serialize($a);

echo 's=',$s;

// сохраняем $s там, где pr32.php может его найти.

$fp = fopen("store", "w");

fputs($fp, $s);

fclose($fp);

?>

Вывод:

s=O:1:"A":1:{s:6:"Aone";i:1;}

<?php

//pr32.php:

// Это необходимо для правильной работы десериализации.

include ("classa.php");

$s = implode("", @file("store"));

$a = unserialize($s);

// Теперь используем функцию show\_one() объекта $a.

$a -> show\_one();

?>

Выводы:

1. Если вы используете сессии и session\_register() для регистрации объектов, эти объ­екты сериализуются автоматически в конце каждой PHP-страницы и десериализуются ав­томатически на каждой последующей странице. Это означает, что эти объекты могут поя­виться на любой странице, после того как стали частью вашей сессии. Рекомендуется включать определение классов всех таких зарегистрированных объектов во все страницы, даже если вы и не используете во всех них эти классы. Если объект десериализуется в отсут­ствие определения класса, он потеряет ассоциацию с этим классом и станет практически бесполезен.

Итак, если в вышеприведенном примере $a стало частью сессии через запуск ses­sion\_register("a"), вы должны включить/include файл classa.inc во все ваши страницы, а не только в page1.php и page2.php.

Подготовка данных к сохранению.

Данные сессии, хранятся в массиве $\_SESSION.

Данные хранятся на стороне сервера и об их наличии и составе пользователь может только догадываться. В программе эти данные могут состоять из любых типов данных, доступных программисту: строки, числа, массивы и объекты, но не ресурсы (об этом ниже). При записи данные подвергаются процедуре сериализации (serialization) – преобра­зования структур любой сложности (не содержащих ресурсов) в строку, из которой в последствии можно восстановить прежнюю структуру. Для сериализации используется функция serialize(). Для обратного процесса – десериализации – unserialize(). При старте сессии эти PHP, получив от обработчика сессии строку, посредством функции unserialize() восста­навливает данные и помещает их в массив $\_SESSION. При завершении программы про­исходит обратный процесс: массив $\_SESSION автоматически сериализуется и строка пе­редается обработчику сессии для записи, т. е. обработчик сессии работает только со стро­кой и не зависит от сложности структуры хранимых данных.

Рассмотрим особенности работы сериализации:

2. Объекты и их классы.

Особенность сериализации объекта заключается в том, что сохраняется только состояние объекта – его данные, а для «оживления» обязательно заранее должен быть объявлен класс.

Выходов из этого положения два: заранее загружать объявления необходимых клас­сов, либо воспользоваться специальным обработчиком, который сможет попытаться под­грузить требуемый класс. Назначается обработчик в ini.php (unserialize\_callback\_func'), но может быть изменен во время работы программы.

Выглядит этот обработчик так:

Код:

<?php

ini\_set("unserialize\_callback\_func", "my\_unserialize\_callback\_func");

function my\_unserialize\_callback\_func($classname)

{

// Предпологаем, что файлы, содержащие классы,

// находятся в поддиректории classes и имеют имя

// соответствующее имени класса.

$filename = "classes/$classname.php";

if (!is\_file($filename))

die("Не удалось подгрузить класс '$classname'");

include\_once($filename);

}

?>

Для того чтобы можно было управлять процессом сериализации/ десериализации в классах PHP предусмотрены специальные методы: \_\_sleep() и \_\_wakeup(). Метод \_\_sleep вызывается перед сериализацией чтоб можно было убрать ненужные связи и должен вернуть массив со списком свойств, которые нужно сериализовать. Метод \_\_wakeup() нужен для выполнения процедур инициализации объекта после де­сериализации. \_\_wakeup вызывается после unserialize, чтоб можно было необходимые связи вернуть.

Пример:

class A implements Serializable {

private $x='y';

function \_\_construct()

{ echo \_\_METHOD\_\_ . " called\n"; }

function \_\_destruct()

{ echo \_\_METHOD\_\_ . " called\n"; }

function serialize()

{ return $this->x; }

function unserialize($s)

{ $this->x=$s; }

}

class B {

private $x='y';

function \_\_construct()

{ echo \_\_METHOD\_\_ . " called\n"; }

function \_\_destruct()

{ echo \_\_METHOD\_\_ . " called\n"; }

function \_\_sleep()

{ return array('x'); }

}

echo "Serializable (the new way)\n";

$x1 = new A;

$s = serialize($x1);

var\_dump($s);

$y1 = unserialize($s);

var\_dump($y1);

echo "\_\_sleep/\_\_wakeup (the old way)\n";

$x2 = new B;

$s = serialize($x2);

var\_dump($s);

$y2 = unserialize($s);

var\_dump($y2);

### 6.12. Сетевые соединения и сокеты

Понятие «сокет» означает «гнездо», «разъем». Два гнезда можно связать соедине­нием и передавать между ними данные. Каждое гнездо принадлежит определенному хосту (host – хозяин, держатель), имеющему уникальный IP-адрес и порт ( [host]:[port] ). Например, 127.0.0.1:8888 означает, что сокет занимает порт 8888 на хосте 127.0.0.1(на данном компьютере). Чтобы не использовать труднозапоминаемый IP-адрес, для доступа к хостам используется система символьных имен DNS.

Сокеты – это абстракция, представляющая соединения двух распределенных приложений – кли­ента и сервера. Соединение все­гда устанавливают приложения-клиенты. В обязанность приложения-сервера входит про­слушивание порта и ожидание соединения. После соединения клиенты посылают или получают от сервера сообщения, а в конце ра­боты закрывают соединение. Сокеты представляют собой удобную технологию взаимо­действия между процессами в сети.

**Создание нового сокета**

Независимо от типа (клиентский или серверный) сокеты создаются с помощью функ­ции socket\_create ():

$socket = socket\_create($domain, $type, $protocol).

Здесь $domain представляет тип создаваемого сокета и может принимать значения: AF\_INET (протокол IPv4), AF\_INET6 (протокол IPv6), AF\_UNIX (локальное взаимодейст­вие). Второй параметр $type определяет тип взаимодействия через сокет. Значение $type = SOCK\_STREAM означает надежный двунаправленный поток, основанный на TCP. Значение $type = SOCK\_DGRAM означает ненадежный сокет, без подключения, передаю­щий данные по протоколу UDP. Используется, если важна скорость передачи. Значение $type = SOCK\_RAW – сокетное подключение, удобное для выполнения операций ICMP (Internet Control Message Protocol — протокол управляющих сообщений ), таких как trace и ping. Последний параметр $protocol может возвращаться функцией getprotobyname() или задаваться константой SOL\_TCP или SOL\_UDP для соединений TCP/UDP.

Функция socket\_create() возвращает ссылку на сокет, либо значение false в случае ошибки. Сокет может быть уничтожен функцией socket\_close($socket).

Информацию о последней ошибке в данном сокете можно получить с помощью функции $error\_code = socket\_last\_error($socket);. Чтобы транслировать сообщение в строковую, преду­смотрена дополнительная функция socket\_strerror($error\_code).

**Создание подключения клиентских сокетов**

Подключение сокета-клиента к серверу-сокету выполня­ется с помощью функции

socket\_connect($socket, $address [, $port]),

где $socket – сокет-клиент, участвующий в соединении; $address – IP-адрес сервера-сокета, к которому нужно подключиться; $port – порт сервера. Функция подклю­чает сокет-клиент к указанному серверу или возвращает зна­чение false.

После установки соединения данные могут быть отправлены че­рез сокет с помощью функци

socket\_write($socket, $buffer [, $length]).

Здесь $socket – это сокет для записи данных, переданных в пара­метре $buffer. Третий параметр $length может быть ука­зан, если не надо записывать весь буфер. Функция возвращает коли­чество запи­санных байт, либо значение false в случае ошибки.

Для чтения данных применяется функция socket\_read($socket,$length [, $type]), где $socket – сокет, из которого нужно прочитать максимум $length байт. Необя­зательный параметр $type принимает следующие зна­чения:

$type = PHP\_BINARY\_READ – интерпретировать данные как бинар­ные (по умолча­нию);

$type = PHP\_NORMAL\_READ – читать данные заданной длины либо пока не встретится символ новой строки (\г или \n). Пример:

<?php

$address = "127.0.0.1";

$port = 80;

$socket = socket\_create(AF\_INET,SOCK\_STREAM,SOL\_TCP);

socket\_connect($socket, $address, $port);

socket\_write($socket, "GET /index.php HTTP/1.1\n\n");

$result = "";

while($read = socket\_read($socket, 1024))

{ Sresult .= Sread;}

echo "Полученный результат: '$result'\n";

socket\_close($socket);

?>

**Создание серверных сокетов**

Первый шаг – создать со­кет с помощью функции socket\_create(). Второй шаг – привязать сокет к хасту (адресу и порту), с помощью функции socket\_bind($socket, $address [, $port]).

После привязки сервер-сокета к хосту, он должен быть настроен на прослу­шивание трафика с помощью функции socket\_listen($socket [, $backlog]). Необязательный параметр $backlog используется для создания очереди посредством указания максимально допустимого числа подключений клиентов. Функция возвращает значение true, указывающее на успешность настройки серверного сокета на прослушивание.

Последний шаг в создании серверного сокета – дать команду на прием входящих подключений функцией socket\_accept($socket); где $socket – сервер-сокет, включенный на прослушивание. Как только соединение будет установлено, функция вернет новый со­кетный ресурс, используемый для подключения. Если указанный в параметре $socket со­кет настроен как неблокирующий, функция socket\_accept() всегда будет возвращать false. Сокетный ресурс, возвращенный функцией socket\_accept (), не может быть повторно ис­пользован, поскольку он обслуживает только одно определенное текущее подключение. Сокет, переданный указанной функции в параметре $socket, может быть использован повторно. Пример.

<?рhр

/\* Запретить прерывание сценария по тайм-ауту \*/

set\_time\_limit(0);

$address = "127.0.0.1";

$port = 4545;

$socket = socket\_create(AF\_INET, SOCK\_STREAM, SOL\_TCP);

socket\_bind($socket, $address, $port);

socket listen($socket);

$connection = socket\_\_accept ($socket);

$result = trim(socket\_read($connection, 1024));

echo "Принят результат: '$result'\n";

socket\_close($connection);

socket\_shutdown($socket) ;

socket\_close($socket) ;

?>

**Одновременная работа с несколькими сокетами**

Сервер может работать одновременно с множеством клиентов. Чтобы сделать это, понадобится функция

socket\_select($read, $write, $$error, $sec [, $usec]),

где $read, $write и $error – переданные по ссылке массивы, которые должны содержать список всех сокетов, наблюдаемых на предмет чтения, записи и перехвата ошибок соответственно. Например, помещение активного сокета в массив, передаваемый в параметре $read, заставляет РНР проверять, есть ли в этом сокете данные для чтения. Последние два параметра – $sec и необязательный $usec – это значения тайм-аута, управляющие тем, как долго будет ожидать функция socket\_select (), прежде чем вернуть управление РНР. Функция socket\_select() возвращает целое число, указывающее общее количество измененных сокетов из переданного списка.

Сокеты, перечисленные в массиве $read, содержат данные, подлежащие чтению из них, либо входящие подключения к ним.

Сокеты, перечисленные в массиве $write, содержат данные, подлежащие записи в них.

Сокеты, перечисленные в массиве Serror, содержат ошибки, которые нужно обработать.

В случае ошибки socket\_select () возвращает значение false.

Для использования этой функции в реальном приложении первым делом должен быть создан сокет, представляющий сервер в целом. Этот главный сокет привязан к оп­ределенному адресу и порту и начнет прослушивание подключений; он будет также добавлен в массив $read, что обуславливает начало управляемого бесконечного цикла. Затем с помощью функции socket\_select () будет организован мониторинг главного сокета на пред­мет новых подключений. Когда появляется новое подключение, автоматически вызыва­ется функция socket\_accept (), что приводит к созданию нового серверного сокета, исполь­зуемого для взаимодействия с подключенным клиентом. Этот новый подключенный сокет затем подвергается мониторингу через тот же вызов socket\_select () (за счет добавления его к тому же массиву, куда уже добавлен наш главный сокет) и реализуется логика при­ложения, обеспечивающая функциональность сервера. Пример.

<?php

set\_time\_limit (0) ;

$NULL = NULL;

$address = "127.0.0.1";

$port = 4545;

$max\_clients = 10;

$client\_sockets = array();

$master = socket\_create(AF\_INET, SOCK\_STREAM, SOLJTCP);

$res = true;

$res s = @socket\_bind(Smaster, Saddress, Sport);

$res S = @socket\_listen(Smaster); if (! Sres) {

die("Невозможно привязать и прослушивать Saddress:Sport\л");

}

$abort = false;

$read = array(Smaster);

while(!Sabort) { $num\_changed = socket\_select(Sread, SNULL, SNULL, 0, 10);

/\* Изменилось что-нибудь? \*/

if(Snum\_changed) {

/\* Изменился ли главный сокет (новое подключение) \*/

if(in\_array(Smaster, Sread)) ( if(count($client\_sockets) < $max\_clients) { $client\_sockets[) - socket\_accept(Smaster);

echo "Принято подключение (" . count($client\_sockets) . " of $max\_clients)\n";

1

)

/\* Цикл по всем клиентам с проверкой изменений в каждом из них \*/ foreach($client\_sockets as $key => Sclient) ( /\* Новые данные в клиентском сокете? Прочи­тать и ответить \*/

if(in\_array($client, $read)) {

Sinput = socket\_read($client, 1024);

if(Sinput === false) (

socket\_shutdown($client);

unset($client\_sockets[$key]); ) else {

$input = trim($input);

if(!@socket\_write(Sclient, "Высказали: $input\n")) ( socket\_close($client); un­set($client\_sockets(Skey]);

}

}

if(Sinput == 'exit') ( socket\_shutdown{Smaster); Sabort = true;

}

}

}

}

$read = $client\_sockets;

$read[] = $master;

}

?>

**Функция fsockopen()**

Функция fsockopen() предназначена для соединения клиента с сервером. Она является универсальной и объединяет в себе функции, которые есть в PHP для работы с сокетами. Функция имеет следующий синтаксис:

int fsockopen (string target, int port[, int errno[, string errstr [, float timeout]]]).

Первый аргумент функции представляет собой URL хоста, с которым необхо­димо соединиться, второй – номер порта port, по которому произойдет соединение. Необязательный аргумент errno указывает на номер ошибки, а errstr содержит текст сообще­ния об ошибке. Аргумент timeout позволяет указать количество секунд, в течение кото­рого устанавливается соединение.

Рассмотрим простое приложение, позволяющее пользователю узнать, на кого и где зарегистрирован IP-адрес. Для этого мы с помощью функции fsockopen установим соединение с сервером whois.ripe.net, как это показано ниже:

<html>

<head>

<title>IP information</title>

</head>

<body>

<center>

<FORM action = ip.php method = post>

<br>

</i></b></font>

<br>

<INPUT type = text name = ip size = 35>

<input type = submit value = 'Введите IP-адрес'>

</form>

</center>

<?

if ($ip! = "")

{

$sock = fsockopen ("whois.ripe.net", 43, $errno, $errstr);

if (!$sock)

{

echo("$errno($errstr)");

return;

}

else

{

fputs ($sock, $ip."\r\n");

while (!feof($sock))

{

echo (str\_replace(":",":&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;",

fgets ($sock,128))."<br>");

}

}

fclose ($sock);

}

?>

</body></html>

### 6.13. Вывод графических данных с помощью РНР

Основной способ создания изображений основан на использовании библиотеки графических функций GD. В следующем примере информация о библиотеке функ­ций GD получена с помощью функции gd\_info() в виде ассоциированного массива.

<?php //imagepr0.php

print\_r(gd\_info());

?>

Результат:

Array

( [GD Version] => bundled (2.0.34 compatible)

[FreeType Support] => 1

[FreeType Linkage] => with freetype

[T1Lib Support] => 1

[GIF Read Support] => 1

[GIF Create Support] => 1

[JPG Support] => 1

[PNG Support] => 1

[WBMP Support] => 1

[XPM Support] =>

[XBM Support] => 1

[JIS-mapped Japanese Font Support] =>

)

Для создания изображений в GD в памяти компьютера создается холст для изображения. Затем на холст наносятся цвета. После этого выполняется рисование на холсте. Готовый рисунок либо сохраняется в файле, либо передается для отображения браузеру. Ненужный холст удаляется из памяти. Холсты для рисования создаются на основе палитры или в натуральном цвете. Для создания новых палитр используется функция imagecreate(), возвращающая ссылку на холст или false в случае неудачи:

$img =imagecreate($width, $height).

Параметры $width и $height определяют значения ширины и высоты холста в пикселях. После создания нового холста необходимо распределить цвет его палитры с помощью функции imagecolorallocate():

imagecolorallocate($img, $red, $green, $blue).

Параметр $img представляет cсылку на холст, на который распределяется цвет, а параметры $red, $green и $blue представляют шестнадцатеричные числа от 0x0 до 0xFF. Если цвет был распределен удачно, функция возвращает целочисленный индекс, соответствующий положению цвета в палитре; в противном случае она возвращает минус 1.

С помощью определенных функций выполняется рисунок на холсте. При создании нового изображения, первый цвет, распределяемый для палитры, автома­тически присваи­вается всему изображению как цвет фона.

Полученное изображение необходимо либо сохранить в файле, либо переслать для вывода в окне браузера. Если использовать при этом формат PNG, то вызыва­ется функ­ция imagepng (). Синтаксис этой функции выглядит следующим образом:

imagepng($img[, Sfilename]);

Аналогично используется функция imagejpeg($img, "imname.jpeg").

Параметр $img представляет ссылку на ресурс изображения, а необязательный параметр $filename – имя файла, в котором будет храниться изображение. Если имя файла не указано, функция выведет изображение непосредственно в окне браузера.

При­мер:

<?php //imagepr1.php

$width = 200; $height = 200;

$img = imagecreate($width, $height);//создание холста

$red = 0xff; $green = 0; $blue = 0;

imagecolorallocate($img, $red,$green,$blue);//палитра

header("Content-type: image/png");

imagepng($img);//вывод в окно браузера

?>

Для вывода изображений в других форматах могут использоваться функ­ции: image2wbmp() (формат WBMP); imagejpeg () (формат JPEG); imagegd() (формат GD); imagegd2 () (формат GD2). В следующем примере изображение квадрата поме­щается в файл image.jpeg

<?php //imagepr2.php

$width = 400; $height = 400;

$img = imagecreate($width, $height);

$red = 0; $green = 0xff; $blue = 0xff;

imagecolorallocate($img, $red,$green,$blue);

header("Content-type: image/jpeg");

imagejpeg($img, "image.jpeg"); //вывод в файл

?>

Вместо того чтобы создавать изображение на основе палитры, можно создать изображение в натуральном цвете с помощью функции imagecreatetruecolor ().

<?php//imagephp3.php

header ('Content-type: image/png');

//gif не поддерживает

$im = @imagecreatetruecolor(200, 20)

or die('Cannot Initialize new GD image stream');

$t\_color = imagecolorallocate($im, 233, 14, 91);

imagestring($im, 1, 5, 5, 'A Simple Text String', $t\_color);

imagepng($im);

imagedestroy($im);

?>

# Получение информации об изображении

Функции imagesx($img) и imagesy($img) возвра­щают значения (в пикселях) соответственно ширины и высоты изображения для данного ресурса. Еще одна похожая функция – это функция getimagesize(). Для изображений в натуральном цвете расширение GD предлагает функцию

imageistruecolor($img).

Эта функция возвращает булево значение true, если переданный ей ресурс изобра­жения $img имеет натуральный цвет; в противном случае она возвращает значение false.

Если необходимо узнать, может ли РНР работать с конкретным типом изображений, то лучше всего использовать функцию imagetypes ().

При выполнении эта функция возвращает битовое поле, сформированное из кон­стант: IMG\_GIF | IMG­\_JPG | IMG\_PNG | IMG\_WBMP с использованием логического "ИЛИ". Рассмотрим пример:

<?php

$supported = imagetypes();

if($supported & IMG\_JPG) {

echo "версия GD поддерживает изображения в формате JPEG."; }

else {

echo "версия GD не поддерживает изображения в формате JPEG.";

}

?>

# Функции рисования прямых и кривых линий

РНР поддерживает рисование линий, прямоугольников, эллипсов и многоугольников и построение заполненных форм. Можно выбирать способ рисования линии с помощью стилей и кистей. Рисование линий на холсте осуществляется с помощью функции

imageline($img,$start\_x, $start\_y, $end\_x, $end\_y, $color).

Параметр $img представляет ресурс изображения для рисования прямой линии, начи­ная с точки ($start\_x, $start\_y), до точки с координатами ($end\_x, $end\_y) с цветом, определяемым параметром $color. Для рисова­ния многоугольников и прямоугольников используются функции imagepolygon () и imagerectangle() соответственно:

imagerectangle($img,$topLx,$topLy,$bRx,$bRy,$color).

Размеры и расположение прямоугольника определяются точками верхнего левого угла ($topLx, $topLy) и нижнего правого угла ($bRx, $bRy).

Синтаксис функции imagepolygon() выглядит следующим образом:

imagepolygon($img, $points, $num\_points, $color).

Параметр $points – это массив, содержащий координаты (X; Y) каж­дой вер­шины, $num\_points – общее количество пар (X, Y) в массиве.

Пример:

<?php //imagepr6.php

// Create a 500 x 500 image

define("WIDTH", 500); define("HEIGHT", 500);

$canvas = imagecreatetruecolor(WIDTH, HEIGHT);

// Allocate colors

$pink = imagecolorallocate($canvas, 255, 105, 180);

$green = imagecolorallocate($canvas, 132, 135, 28);

$white = imagecolorallocate($canvas, 0xFF, 0xFF, 0xFF) ;

$black = imagecolorallocate($canvas, 0, 0, 0);

// Draw three rectangles each with its own color

imagerectangle($canvas, 50, 50, 150, 150, $pink);

imagerectangle($canvas, 45, 60, 120, 100, $white);

imagerectangle($canvas, 100, 120, 75, 160, $green);

$num = 5;

$mycolor = $white;

$color = 0xff;

//Рисование прямой

imageline($canvas, 0,0,500,500, $mycolor);

$points = array(20,20,50,HEIGHT/2,WIDTH/2,400,WIDTH-10,HEIGHT-10, WIDTH-10, 0);

//Рисование многоугольника

imagepolygon($canvas, $points, $num, $color);

// Output and free from memory

header('Content – Type: image/jpeg');

imagejpeg($canvas);

imagedestroy($canvas);

?>

GD поддерживает рисование дуги и эллипса посредством функций imagearc() и im­ageellipse().

imageellipse($img,$centerx,$centery,$width,$height,$color).

Параметры $centerx и $centery задают координаты центра эллипса. Форма эллипса оп­ределяется парамет­рами $width и $height. Окружность является частным случаем эллипса. Синтаксис функции рисования дуги:

imagearc($img, $center\_x, $center\_y,$width,$height, $start\_ang, $end\_ang,$color).

Функция imagearc() требует два дополнительных параметра, $start\_ang и $end\_ang, ко­торые представляют угол в градусах.

Пример:

<?php //imagepr7.php

define("WIDTH", 200);

define("HEIGHT", 100);

$img = imagecreate(WIDTH+100, HEIGHT+200);

$bg = imagecolorallocate($img, 0, 0xFF, 0);

$black = imagecolorallocate($img, 0, 0, 0);

$red = imagecolorallocate($img, 0xFF, 0, 0);

$center\_x = (int)WIDTH/2; $center\_y = (int)HEIGHT/2;

imageellipse($img, $center\_x, $center\_y, WIDTH, HEIGHT, $black);

imagearc($img, $center\_x, $center\_y, WIDTH-5, HEIGHT-5, 0, 360, $red);

header("Content-Type: image/png");

imagepng($img);

?>

# Заливка изображений

Расширение GD предлагает функции imagefilledrectangle () , imagefilledpolygon() и imagefilledellipse(), которые заливают всю геометриче­скую форму определенным цветом:

imagefilledarc($img, $centerx, $centery, $width, $height, $start\_ang, $end\_ang, $color, $style).

Дополнительный параметр $style представляет собой битовое поле, определяющее способ заливки. Пример построения секторной диаграммы:

<?php

// create image //imagepr10

define("WIDTH", 400); define("HEIGHT", 400);

$pie\_data = array (10, 5, 20, 40, 10, 15);

$image = imagecreatetruecolor(WIDTH, HEIGHT);

// allocate some solors

$white = imagecolorallocate($image, 0xFF, 0xFF, 0xFF);

$navy = imagecolorallocate($image, 0x00, 0x00, 0x80);

$cx = (int)WIDTH/2; $cy = (int)HEIGHT/2;

$a = WIDTH-20; $b = (int)$a/2;

$col = $white;

// make the 3D effect

imagefilledarc($image,$cx,($cx+20),$a,$b,0,360,$navy, IMG\_ARC\_PIE);

$langl = 0;

foreach($pie\_data as $perc) {

$arclen = (360 \* $perc) / 100;

imagefilledarc($image, $cx,$cy, $a,$b, $langl, ($langl+$arclen), $col ,IMG\_ARC\_PIE);

$langl + = $arclen; $col = $col-20000;

}

// flush image

header('Content-type: image/png');

imagepng($image);

imagedestroy($image);

?>

Функция imagefill () на любом сплошном участке заменяет один цвет другим выбран­ным цветом. Синтаксис функции imagefill () выглядит следующим образом:

imagefill($img, Sx, Sy, Scolor).

$х и $y определяют местоположе­ние заменяемого цвета, а параметр $color представ­ляет индекс цвета в палитре, который будет использован для замены. Альтернативная функция, которая обладает более широкими возможностями – это imagefilltoborder (). Ее синтаксис выглядит следующим образом:

imagefilltoborder($img,$х, Sy, Sborder, Scolor).

Эта функция выполнит заливку сплошного участка, определяемого не одним цветом, а цветом, устанавливаемым параметром $border.

# Удаление и изменение цветов палитры

Для удаления цветов из палитры можно воспользоваться функцией

imagecolordeallocate($img, $color).

Параметр $color задает цвет в палитре, который должен быть удален. После того как цвет удален из палитры, позиция используется для хранения другого цвета, назначаемого с помощью функции imagecolorallocate().

<?php //imagepr13

define("WIDTH", 300); define("HEIGHT", 300);

$img = imagecreate(WIDTH, HEIGHT);

$backg = $white = imagecolorallocate($img,0xff, 0xff,0xff);

$black = imagecolorallocate($img, 0, 0, 0);

$red = imagecolorallocate($img, 0xFF, 0, 0);

$blue = imagecolorallocate($img, 0, 0, 0xFF);

imagefilledrectangle($img, 0, 0, WIDTH-1, HEIGHT-1, $black);

imagefilledrectangle($img, 40,40, WIDTH-1, HEIGHT-1, $red);

imagefilledrectangle($img, 80, 80, WIDTH-1, HEIGHT-1, $blue);

imagecolordeallocate($img, $black);

$green = imagecolorallocate($img, 0, 0xff, 0);

// Get the color index for the background

$bg = imagecolorat($img, 0, 0);

// Set the backgrund to be blue

imagecolorset($img, $bg, 200,200, 255);

// Output the image to the browser

header("Content-Type: image/png");

imagepng($img);

imagedestroy($img);

?>

Расширение GD позволяет напрямую заменять определенный индекс в палитре дру­гим цветом с помощью функции imagecolorset($img, $color, $red, $green, $blue).

# Создание прозрачности в изображениях

В PHP не поддерживаются прозрачные изображения в формате GIF, но формат PNG позволяет использовать прозрачные цвета. Для этого цвет распределенный на палитру, необходимо объявить прозрачным с помощью функции

imagecolortransparent($img [, $color]).

Необязательный параметр $color задает цвет, который будет объявлен прозрачным и не будет отображаться. Это позволит видеть все, что находится под изображением (фон). В результате выполнения функция возвращает прозрачный цвет в палитре, а если параметр $color не был задан, то текущий прозрачный цвет. Для данного изображения можно определить только один прозрачный цвет. Поэтому при повторном вызове функции прозрачный цвет будет превращен в реальный цвет.

<?php //imagepr14

// Create a 150x30 image

$img = imagecreatetruecolor(150, 30);

$red = imagecolorallocate($img, 255, 0, 0);

$black = imagecolorallocate($img, 0, 0, 0);

// Make the background transparent

imagecolortransparent($img, $black);

// Draw a red rectangle

imagefilledrectangle($img, 4, 4, 150, 25, $red);

header("Content-Type: image/png");

imagepng($img);

// Save the image

imagepng($img, './imagecolortransparent.png');

imagedestroy($img);

?>

Кроме рассмотренной функции, возможно создание прозрачности с помощью альфа-смешивания цветов RGBA. Аббревиатура RGBA (Red, Green, Blue, Alpha) задает способ определения прозрачных цветов на основе тройки значений RGB. Цвета связаны посредством альфа-уровня, который лежит в диапазоне (от 0x0 до 0x7F) и соответствует степени прозрачности этого цвета при его помещении на холст.

Основой системы цветов RGBA является то, что когда цвет распределяется с помощью метода RGBA и помещается на холст, то цвет, который находится под закрашиваемой областью, не будет полностью удален. Наоборот, этот цвет будет комбинироваться со значением RGB помещаемого цвета для получения итогового цвета, отображаемого на холсте. Изображение должно содержать натуральные цвета, созданые с помощью РНР-функции imagecreatetruecolor (). Для создания цвета RGBA используется функция

imagecolorallocatealpha($img\_r, $red, $green, $blue, $alpha).

Параметр $alpha может принимать значение от 0 (непрозрачный цвет) до 127 (прозрачный цвет). Использование функции imagecolorallocatealpha ():

<?php //imagepr15

define("WIDTH", 300); define("HEIGHT", 300);

$img = imagecreatetruecolor(WIDTH, HEIGHT);

$white = imagecolorallocate($img, 0xFF, 0xFF, 0xFF);

$yellow = imagecolorallocate($img, 0xFF, 0xFF, 0);

$red = imagecolorallocate($img, 0xFF, 0, 0);

$blue\_t = imagecolorallocatealpha($img, 0, 0, 0xFF, 0x40);

imagefill ($img, 1, 1, Swhite) ;

imageline($img, 0,0, WIDTH-1, HEIGHT-1, $blue\_t);

imagefilledrectangle($img, (WIDTH/2)-50, (HEIGHT/2)-50,

(WIDTH/2)+50, (HEIGHT/2)+50, Syellow);

imagefilledrectangle($img, (WIDTH/2)-30, (HEIGHT/2)-30,

(WIDTH/2)+30, (HEIGHT/2)+30, $red);

imagefilledrectangle($img, 10, 10, WIDTH-11, HEIGHT-11, $blue\_t);

header("Content-Type: image/png");

imagepng($img);

?>

Три закрашенных прямоугольника были нарисованы поверх диагональной линии синего цвета. Как можно видеть, даже при том условии, что оба прямоугольника меньшего размера и линия были нарисованы раньше, нежели большой синий прямоугольник RGBА, видимыми остаются все три прямоугольника. Это объясняется тем, что во время рисования этого прямоугольника альфа-уровень составлял половину прозрачности (0x40 в шестнадцатеричной форме, 64 – в десятичной) и был смешан с цветами, поверх которых нарисован этот прямоугольник. Благодаря этому и получен визуальный эффект, при котором сквозь большой прямоугольник RGBA просматриваются остальные закрашенные формы, находящиеся за ним.

Помимо функции imagecolorallocatealpha(), расширение GD предлагает следующие функции для работы с цветами RGBA:

imagecolorallocatealpha(Simg\_r, Sred, Sgreen, Sblue, Salpha); imagecolorexactalpha($img\_r, Sred, Sgreen, Sblue, Salpha); imagecolorclosestalpha(Simg\_r, Sred, Sgreen, Sblue, Salpha); imagecolorresolvealpha($img\_r, Sred, Sgreen, Sblue, Salpha).

# Рисование с помощью кистей

Весь процесс рисования на холсте осуществляется с помощью цифровой кисти. Начиная с кисти, используемой по умолчанию, самым простым изменением является изменение толщины кисти. Для этих целей используется функция imagesetthickness():

imagesetthickness($img, $thickness).

Параметр $thickness определяет толщину кисти. Хотя функция imagesetthickness () позволяет рисовать на холсте формы произвольной толщины и цвета, с ее помощью нельзя нарисовать прерывистые линии. Для этих случаев расширение GD предлагает функцию imagesetstyle ():

imagesetstyle($img\_r, $style).

Параметр $style задает стиль в виде индексированных массивов, в которых в упорядоченном виде содержатся цвета, определяющие кисть в каждом пикселе. Например, если предположить, что существуют переменные Swhite и Sblack, которые устанавливают соответствующие цвета (белый и черный), то следующий массив может представить штриховую кисть:

Sdas = array($black, $black, $black, $white $white $white).

В этом примере массив Sdas – это простая кисть, которая состоит из шести пикселей: трех черных, за которыми следуют три белых. Впоследствии этот массив можно передать в качестве параметра Sstyle для функции imagesetstyle (), чтобы определить текущий стиль кисти. После того как стиль будет установлен с помощью этой функции, его можно использовать для рисования любой геометрической формы, поддерживаемой расширением GD, посредством константы IMGCOLORSTYLED, которая применяется вместо того, что обычно может представлять цвет при вызове функции imagesetstyle () выбирался стиль, после чего константа IMG\_COLOR\_STYLED, используемая вместо цвета, указывала на то, какой стиль необходимо выбрать в данный момент для рисования.

<?php

header("Content-type: image/jpeg");

$im = imagecreatetruecolor(100, 100);

$w = imagecolorallocate($im, 255, 255, 255);

$red = imagecolorallocate($im, 255, 0, 0);

/\* Draw a dashed line, 5 red pixels, 5 white pixels \*/

$style = array($red, $red, $red, $red, $red, $w, $w, $w, $w, $w);

imagesetstyle($im, $style);

imageline($im, 0, 0, 100, 100, IMG\_COLOR\_STYLED);

imagejpeg($im);

imagedestroy($im);

?>

Чтобы определить сложные кисти, ширина которых состав­ляет несколько пикселей, необходимо использовать иной способ, который заключается в создании другого изображения кисти с помощью функции image setbrush ():

imagesetbrush ($img\_r, $brush\_r).

Параметр $img\_r представляет ресурс изображения, а параметр $brush\_r задает иной ресурс изображения, содержащий требуемую кисть. Этот ресурс изображения кисти ничем не отличается от любого другого представленного ресурса изображения, и его можно создать с помощью целого ряда функций работы с изображениями GD. Как и в случае с функцией imagesetstyle (), функцию imagesetbrush () следует вызывать всякий раз при изменении кисти, а вместо цвета необходимо использовать специальную константу IMG\_COLOR\_BRUSHED всякий раз при вызове функции рисования GD, использующей кисть.

Кисти могут применяться и для заливки участков холста с помощью функции imagefilledpolygon() или imagefill(). Шаблон заливки называется мозаикой (tile) и определяется с помощью функции imagesettile($img, $tile\_r).

# Использование шрифтов и вывод строк

Расширение GD поддерживает три библиотеки шрифтов (TILib, FreeType и FreeType2), что дает возможность использовать в изображениях шрифты PostScript и TrueType. Кроме этого, расширение GD предлагает дополнительно пять внутренних шрифтов, использование которых сводится к работе с функцией

imagestring($img\_r, $font, $start\_x, $start\_y, $str, $color).

Параметр $img\_r представляет ресурс изображения, $color – ресурс цвета, который будет использоваться при выводе текста. Параметр $font определяет выбор шрифта посредством целого числа (от 1 до 5), а параметры $start\_x и Sstart\_y задают координаты текста. Параметр $str устанавливает строку, которую необходимо отобразить. Использование функции imagestring ()

<?php //imagepr18

define("WIDTH", 300); define("HEIGHT", 100);

$img = imagecreate(WIDTH, HEIGHT);

$white = imagecolorallocate($img, 255,255,255);

$black = imagecolorallocate($img, 0,0,0);

imagerectangle($img, 0, 0, WIDTH-1, HEIGHT-1, $black);

$start\_x = 15; $start\_y = 15;

for($font\_num = 1; $font\_num <= 5; $font\_num++)

{ imagestring($img, $font\_num, $start\_x , $start\_y, "Font #$font\_num", $black);

$start\_y += 15;

}

header("Content-type: image/png") ;

imagepng($img) ;

?>

Расширение GD делает внутренние шрифты более универсальными, позволяя отображать текст иа холсте вертикально и горизонтально.

**Манипулирование и копирование изображений**

Рассмотрим копирование части или всего изображения в память компьютера для переноса на другой холст. Чаще всего применяется функция imagecopy

imagecopy($dest, $src, $dest\_x, $dest\_y, $src\_x, $src\_y, $src\_w, $src\_h).

Параметры $dest и $src представляют ресурсы изображений для конечного и исходного холстов. Параметры $dest\_x и $dest\_y – координаты конечного холста, куда будет скопирована часть исходного изображения, а параметры $src\_x, $src\_y, $src\_w и $src\_h определяют область копирования из исходного изображения.

Пример:

<?php //imagepr19

$src = imagecreatefromgif('molenr.gif');

$dest = imagecreatetruecolor(100, 100);

// Copy

imagecopy($dest, $src, 0, 0, 30, 30, 130,130);

// Output and free from memory

header('Content-Type: image/gif');

imagegif($dest);

imagedestroy($dest);

imagedestroy($src);

?>

**Что нового в технологиях PHP**

В 2012 г. вышла версия PHP 5.4.x. Добавлено множественное на­следование на PHP, которое имитируется с помощью трейтов Вместо опасного модификатора /e (eval) в регулярных выражениях рекомендуется использовать preg\_replace\_callback. Добавлен новый API для хэширования паролей. Новый API берет на себя генерацию надежных хэшей, скрывая от разработчика операции ручного указания salt и выбора алгоритма хэширования (по умолчанию используется Bcrypt). Новая функция hash\_pbkdf2 замедляет обычное хеширование, чтобы сделать перебор паролей достаточно трудоемким занятием.

Поддержка сопрограмм (coroutine) и генераторов (generator), предостав­ляющих простой и не требующий создания шаблонов способ реализации итераторов; Поддержка расширяющего возможности блоков исключений try/catch ключевого слова "finally". Заданный в секции finally код будет выполнен после всех остальных обработчиков исключений, даже если в catch блоке выполнится return, break, либо continue. Возможность использования empty() и isset() над результатом работы функции и произвольными выражениями. Прекращение поддержки Windows XP и Windows 2003. Прекращение поддержки устаревших функций.

В PHP добавлены и обновлены фреймворки. Например, создан фактически новый фреймворк Zend Framework 2. Yii Framework быстро развивается, проделана огромная работа и быстро растет число пользователей фреймворка. Появившись в начале года фреймворк Phalcon сразу привлек внимание тем, что разработан в виде расширения для PHP. Кроме высокой производительности, Phalcon подкупает документацией, наличием обучения и примерами. Composer – это замечательный инструмент управления на PHP, который пережил настоящий бум. Безусловен рост распространенности этого инструмента. По-прежнему популярны фреймворки-гиганты PHP – Symfony, Zend и Kohana.

Функции CSPRNG

Две новых функции для генерации крипографически безопасной строки и целых. Первая возвращает случайную строку длиной $len:

random\_bytes(int $len);

Вторая возвращает число в диапазоне $min… $max.

random\_int(int $min, int $max);

Синтаксис Escape-кода для Unicode

В отличие от многих других языков, до PHP версии 7, в PHP не было способа указать в строке escape-последовательность для Unicode символа. Теперь с помощью escape-после­довательности \u можно генерировать такие символы с помощью их кода из набора UTF-8. Это лучше, чем непосредственная вставка символов, лучше контролируются невидимые символы и символы, имеющие графическое отображение отличное от значения:

echo "\u{1F602}"; *// выводит смайлик*

Помните, что код, ранее работавший и использовавший пару символов \u, не будет кор­ректно работать в версии 7 из-за изменившегося поведения.

Старые SAPI и расширения

Были убраны старые и ненужные SAPI, такие как расширение mysql Другие расширения и SAPI были портированы на PHP 7. Убраны или более некорректны открывающие/закрывающие тэги в старом стиле 

<% ... %>, <%= ... %>, <script language="php"> ... </script>

### Лабораторная работа 7.

### PHP

### Для выполнения этой работы необходимо установить Веб-сервер, сервер баз данных MySQL, программные средства PHP. Установочным пакетом может быть пакет Denver или Wamp. Затем следует выполнить несколько примеров из текста книги, в том числе для работы с массивами.

### *Задание 1. Массивы и строки*

Выполните одно из перечисленных ниже упражнений.

1.1. В массиве строк проверьте начинается ли каждая строка символом “\*”. Строки, не начинающиеся символом “\* “ перенести в другой массив.

1.2. В массиве из n строк проверьте, содержит ли к-я строка символы /\*. Если не содержит, вставьте эти символы в начало, а \*/ в конец строки.

1.3. В массиве строк удалите все теги, заключенные в угловые скобки <>.

1.4. В массив случайным образом поместите строки, задающие «совет дня» или «цитату дня». Случайным образом выбрерите строку из массива с помощью функции shuffle(array arr); или arrayrand(array arr, int num);

1.5. Создайте многомерный массив: Факультет, Курс, Группа, Студенты. Вы­ведите список студентов каждой группы в алфавитном порядке.

1.6. Создайте многомерный массив: Факультет, Кафедра, Преподаватели. Вы­ведите список преподавателей каждой кафедры в алфавитном порядке.

1.7. В прямоугольной матрице переставьте столбцы в порядке возрастания суммы модулей элементов каждого столбца.

1.8. В прямоугольной матрице разделите все ее элементы на наибольшую сумму модулей элементов в строке.

1.9. Создайте ассоциативный массив {name,prof,adres}. Найдите количество различных элементов массива.

1.10. В прямоугольной матрице со значениями типа string выведите все одинако­вые элементы и их индексы.

1.11. В прямоугольной матрице с элементами типа string заменить нулями все элементы, которые находятся в столбцах или строках, где имеются заданные по образцу строки.

1.12. В прямоугольной матрице с элементами типа string найти отрезок массива наибольшей длины, в котором первый элемент равен последнему, второй – предпоследнему и т.д. Выведите отрезок и длину отрезка.

1.13. В прямоугольной матрице с элементами типа string найдите отрезок массива наибольшей длины, в котором элементы возрастают. Выведите отрезок и его длину.

1.14. Транспонируйте квадратную матрицу и умножте транспонированную матрицу на исходную.

1.15. Даны числа m , n и два упорядоченных массива:

a1≤a2≤...≤am и b1≤b2≤...≤bn. Создайте из элементов новый упорядоченный массив c1≤c2≤...≤cm+n.

1.16. Дана целочисленная матрица Аm,n, строки которой упорядочены по воз­растанию (ai,1≤ai,2≤...≤ai,n для i=1,…,m). Найдите и выведите числа, которые встречаются во всех строках.

1.17. Найдите сумму элементов аi,j матрицы Аm,n, для которых i-j=k.

1.18. Массив чисел Аm,n упорядочен по возрастанию в строках

(ai,1≤ai,2≤...≤ai,n для i=1,…,m) и столбцах (a1,j≤a2,j≤...≤am,j для j=1,...,n). Найдите элементы этого массива, равные заданному числу х и вывести их индексы. Количество действий в программе должно быть порядка m+n, а не m\*n.

1.19. Задана квадратная матрица А из значений типа string. Поверните ее на 90º против часовой стрелки.

1.20. Матрица K(m,m) состоит из нулей и единиц. Найдите в ней номера (ин­дексы)хотя бы одной строки или хотя бы одного столбца, не содержащих единицы, либо сообщите, что таковых нет.

1.21. В вещественном массиве размерности n наименьший элемент поместите на первое место, наименьший из оставшихся – на последнее место, следующий по величине – на второе место, следующий – на предпоследнее и так далее – до середины массива.

1.22. Латинским квадратом порядка n называется квадратная таблица разме­ром n × n, каждая строка и каждый столбец которой содержат все числа от 1 до n. Проверьте, является ли заданная целочисленная матрица латинским квадратом.

1.23. Матрицу K(m,m) заполните следующим образом: элементам, находящимся на периферии (по примеру матрицы), присвойте значение 1; периметру оставшейся подматрицы – значение 2 и так далее до заполнения всей матрицы.

1.24. В массиве строк проверьте начинается ли каждая строка символом \_. Строки не начинающиеся символом \_ исключить из массива.

1.25. В массиве из n строк проверьте, содержит ли *к*-я строка первые символы //. Если не содержит, вставьте эти символы в строку.

1.26. В массиве строк удалите все строки, являющимися HTML-тегами ссылок, которые начинаются <a href=” …”>

1.27. В массив случайным образом помещаются строки-пословицы. Случайным образом выберите *к* пословиц.

### *Задание 2. Функции*

Выполните одно из перечисленных ниже упражнений:

2.1. Среди *n* чисел найдите наибольшее и наименьшее простые числа.

2.2. Для заданного числа *n* постройте треугольник Паскаля.

2.3. Напишите функцию, возвращающую текст приветствия в соответствии с приведенной ниже схемой

<?php

if((date(“G”) >=5)AND(date(“G”) <= 11 ))echo “Good Morning!”;

if((date(“G”)>=12)ANDdate(“G”)<=18))echo “Good Afternoon!”;

if((date(“G”) >= 19)AND(date(“G”)<= 4))echo “Good Evening!”;

?>

2.4. Решите методом итераций уравнение вида . Очередное приближение корня находится по формуле ; начальное приближение  задается произвольно. Обратите внимание, что метод сходится, если только .

2.5. Вычислите определенный интеграл методом прямоугольников: , где *n* – количество отрезков разбиения; .

2.6. Найдите минимум заданной функции  на заданном отрезке  методом золотого сечения. Для этого на отрезке  находятся две точки:  и , где .

2.7. Напишите функцию вычисления .

2.8. Напишите функцию вычисления .

2.9. Напишите функцию *n*! Возможна ли рекурсия.

2.10. Напишите функцию вычисления НОД(*a, b*).

2.11. Напишите функцию вычисления НОК(*a, b*).

2.12. Напишите функцию решения квадратного уравнения .

2.13. Напишите функцию  – площадь треугольника.

2.14.. Из *n* чисел выведите наибольшее число, делящееся на 9.

2.15. Вычислите интеграл  методом трапеций.

2.16. Вычислите интеграл  методом Симпсона.

2.17. Решить уравнение методом Ньютона .

2.18. Значения функции  заданы таблично в массиве  при соответствующих значениях аргумента в упорядоченном массиве . Найдите значение функции в произвольной точке *x* по формуле Лагранжа:

.

2.19. Решите уравнение  методом деления отрезка пополам (методом дихотомии). На каждой итерации отрезок  делится пополам и выбирается та из половин, на концах которой функция  имеет значения разных знаков.

2.20. Напишите функцию вычисления чисел Фибоначчи: ; .

***Задание 3. Файлы и строки***

Файлы могут быть текстовыми, содержащими строки переменной длины, и би­нарными, представляющими последовательность байт.

При выполнении следующих пунктов задания используйте файлы, строки и объекты.

3.1. Текст записан одной длинной строкой. Признаком красной строки служит символ $. Переформатировать текст в 60-символьные строки, формируя абзацы.

3.2. Текст, сформированный построчно, выровняйте по правому краю так, чтобы каждая строка заканчивалась знаком препинания или одним пробелом. Выравнивание осуществить, вставляя дополнительные пробелы между словами (равномерно по всей строке).

3.3. Дан текст программы на каком-либо алгоритмическом языке и словарь зарезервированных слов этого языка (в английской транскрипции). Преобразуйте текст, записав все зарезервированные слова прописными буквами, а остальные кон­струкции (имена и так далее) – строчными. Русские буквы (имена, литералы) не за­менять.

3.4. Программа, записанная 80-байтовыми строками, в последних 8 байтах со­держит номер строки. Строки упорядочены по номерам, но не обязательно с шагом 1. Поступает изменение к программе в таком виде. Вставьте новые строки взамен имеющихся старых или между ними.

3.5. Текст записан 60-символьными строками, содержит следующие знаки кор­ректуры: $ – сделать красную строку; # удалить следующее слово; @ – удалить сле­дующую фразу. Сделайте указанную корректировку, переформатируя строки в пределах абзаца.

3.6. Часто встречающаяся ошибка начинающих наборщиков – дважды записанное слово. Обнаружьте и исправьте такие ошибки в тексте, записанном 80-символьными строками; переформатировать строки в пределах абзаца.

3.7. Стихотворный текст (в строке не более 80 символов) имеет четырехстрочную строфу. Запишите его «лесенкой» (по одному слову в строке), вставляя пустую строку после каждого четверостишия.

3.8. Поздравления. По заданному списку фамилий напечатайте каждому упомянутому в списке поздравление к определенному празднику. Чтобы избежать шаблона, перечень желаемых благ выбирать как случайное подмножество из заготовленного списка (например, здоровья, счастья, продвижения по службе, долголетия и т. д.). Можно сделать переменными и название праздника – для универсальности про­граммы.

3.9. Имеется список членов коллектива с указанием принадлежности каждого к различным общественным организациям (профком, ученый совет, общество книго­любов, федерация пентикса и т. д.). Напечатайте приглашение всем членам на очередное заседание указанной организации. Задается только вид организации, место и время сбора.

3.10. В заданном тексте найдите самое длинное слово и самую длинную фразу.

3.11. Обнаружено, что в тексте пропущены некоторые слова и словосочетания. Эти слова и словосочетания представлены отдельным списком в том порядке, в ка­ком должны быть вставлены. Места вставки отмечены в тексте символом $. Откор­ректируйте текст.

3.12. Текст не содержит собственных имен и сокращений, набран с использова­нием прописных и строчных русских букв. Проверьте, все ли фразы (и только они) начинаются с прописной буквы; при необходимости откорректируйте текст.

3.13. Текст, не содержащий собственных имен и сокращений, набран полностью прописными русскими буквами. Замените все прописные буквы, кроме букв, стоящих после точки, строчными буквами.

3.14. IP-адрес посетителя страницы находится как элемент суперглобального массива S – Server[΄Remoute\_ADDR΄]. Составьте список посетителей за текущий день (неделю, месяц).

3.15. Запрет посещений с определенных IP-адресов. В массиве хранятся IP-ад­реса, выход с которых на наш сайт нежелателен. Сравните IP-адрес посетителя с ад­ресами из списка и не пускайте нежелательного посетителя.

3.16. Составьте список серверов ($\_SERVER[΄SERVER\_NAME΄]) и список стра­ниц ($\_SERVER[΄PHP\_SELF΄]), которые мы посетили.

3.17. Составьте список страниц, с которых приходят к нам посетители ($\_SERVER[΄HTTP\_REFERER΄]).

3.18. Составьте список браузеров, операционных систем, поисковых роботов, которые посещают нашу страницу: ($\_SERVER[΄HTTP\_USER\_AGENT΄]), таким же образом определяются менеджеры-закачки типа ΄DownLoadMaster΄, ΄FlashGet΄ и другие.

3.19. Выясните языковые предпочтения посетителя, который браузер отправляет серверу в HTTP-заголовке и помещает в массив $\_SERVER[΄HTTP\_ACCEPT\_LANGUAGE΄].

3.20. В имеющемся словаре найдите пары слов (анаграммы), при прочтении каж­дого из которых в обратном направлении образуется другое слово пары, например, (ПОЛК, КЛОП); (БАР, РАБ).

3.21. IP-адрес посетителя страницы находится как элемент суперглобального массива S – Server[΄Remoute\_ADDR΄]. Составьте список посетителей за текущий день (неделю, месяц).

3.22. Если посетители приходят из одной организации proxy-server, то IP-адрес у них будет одинаковый. В некоторых случаях можно получить адреса машин в под­сети из среды окружения HTPP\_X\_FORWARDED\_FOR с помощью функции getenu (HTTP\_X\_FORWARDED\_FOR); составьте список посетителей на текущий день (не­делю, месяц).

3.23. Запрет посещений с определенных IP-адресов. В массиве хранятся IP-ад­реса, выход с которых на наш сайт нежелателен. Сравните IP-адрес посетителя с ад­ресами из списка и не пускайте нежелательного посетителя.

3.24. Выясните языковые предпочтения посетителя, который браузер отправляет серверу в HTTP-заголовке и помещает в массив $\_SERVER[΄HTTP\_ACCEPT\_LANGUAGE΄].

3.25. Синонимайзер (замените синонимами данный текст, вводимый из файла).

### *Задание 4. Работа с базами данных*

Выполните одно из перечисленных ниже упражнений

Создайте распределенную информационную систему. Из пользователей системы обязательно наличие *Администратора* и Зарегистрированного *Пользователя*.

В каждом из упражнений необходимо выполнить следующие действия:

* Организацию соединения с базой данных вынести в отдельный класс, метод которого возвращает соединение.
* Спроектируйте БД. Привести таблицы к одной из нормальных форм. Создать БД.
* Создайте класс для выполнения запросов на извлечение информации из БД.
* Создайте класс на добавление информации.
* Результаты выполнения запроса передайте клиенту.

4.1. ***Видеотека.*** В БД хранится информация о домашней видеотеке – фильмы, актеры, режиссеры.

Для фильмов необходимо хранить:

название; актеров; дату выхода; страну, в которой выпущен фильм.

Для актеров и режиссеров необходимо хранить:

ФИО; дату рождения; .

* Найдите все фильмы, вышедшие на экран в текущем и прошлом году.
* Выведите информацию об актерах, снимавшихся в заданном фильме.
* Выведите информацию об актерах, снимавшихся как минимум в двух фильмах.
* Выведите информацию об актерах, которые были режиссерами хотя бы одного из фильмов.
* Удалите все фильмы, дата выхода которых была более двухлет назад.

4.2. ***Расписание занятий.*** В БД хранится информация о преподавателях и проводимых ими занятиях.

Для предметов необходимо хранить:

название; время проведения (день недели); аудитории.

Для преподавателей необходимо хранить:

ФИО; предметы, которые он ведет; количество пар в неделю по каждому предмету; количество студентов, занимающихся на каждой паре.

* Выведите информацию о преподавателях, работающих в заданный день недели в заданной аудитории.
* Выведите информацию о преподавателях, которые не ведут заня­тия в заданный день недели.
* Выведите дни недели, в которых проводится наименьшее количество занятий.
* Вывести дни недели, в которых занято наименьшее количество аудиторий.
* Перенесите первые занятия заданных дней недели на последнее место.
  1. ***Письма.*** В БД хранится информация о письмах и отправителях.

Для людей необходимо хранить:

ФИО; - дата рождения.

Для писем необходимо хранить:

ФИО отправителя; ФИО получателя; тема; текст письма; дата отправки.

* Найдите пользователя, длина писем которого наименьшая.
* Выведите информацию о пользователях, а также о количестве полученных и отправленных ими письмах.
* Выведите информацию о пользователях, которые получили хотя бы одно сообщение с заданной темой.
* Выведите информацию о пользователях, которые не получали со­общение с заданной темой.
* Направьте письмо заданного человека с заданной темой всем людям.
  1. ***Сувениры.*** В БД хранится информация о сувенирах и их производителях. Для сувениров необходимо хранить: название; производителя; дату выпуска; цену. Для производителей необходимо хранить: название; страну.
* Выведите информацию о сувенирах заданного производителя.
* Выведите информацию о сувенирах, произведенных в заданной стране.
* Выведите информацию о производителях, чьи цены на сувениры меньше 1000.
* Выведите информацию о производителях заданного сувенира, про­изведенных в прошлом году.
* Удалите заданного производителя и его сувениры.
  1. ***Заказ.*** В БД хранится информация о заказах магазина и товарах в них.

Для заказа необходимо хранить:

номер заказа; товары в заказе; дату поступления.

Для товаров в заказе необходимо хранить:

товар; количество.

Для товара необходимо хранить:

название; описание; цену.

* Выведите полную информацию о заданном заказе.
* Выведите номера заказов, сумма которых не превышает 100 и количество различных товаров равно 1.
* Выведите номера заказов, содержащих товар с заданным заказом.
* Выведите номера заказов, не содержащих товар с заданным названием и поступившие в течение текущего дня.
* Сформируйте новый заказ, состоящий из товаров, заказанных в текущий день.
* Удалите все заказы, в которых присутствует заданное количество данного товара.
  1. ***Погода*.** В БД хранится информация о погоде в различных регионах.

Для погоды необходимо хранить:

регион; дату; температуру; осадки.

Для регионов необходимо хранить:

название; площадь; тип жителей.

Для типов жителей необходимо хранить:

название; язык общения.

* Выведите сведения о погоде в заданном регионе.
* Выведите даты, когда в регионе шел снег и температура была ниже – 10.
* Выведите информацию о погоде за прошедшую неделю в регионах, жители которых общаются на заданном языке.
* Выведите среднюю температуру за прошедшую неделю в регионах с площадью более 1000 км2.
  1. ***Города*.** В БД хранится информация о городах и их жителях.

Для городов необходимо хранить:

название; год создания; количество населения для каждого типа жителей.

Для типов жителей необходимо хранить:

город проживания;название;язык общения.

* + - Выведите информацию обо всех жителях заданного города, разговаривающих на заданном языке.
    - Выведите информацию обо всех городах, в которых проживают жи­дели выбранного типа.
    - Выведите информацию о городе с максимальным количеством населения и всех типах жителей в нем проживающих.
    - Выведите информацию самом древнем типе жителей.
  1. ***Словарь*.** В БД хранится англо-русский словарь, в котором для одного английского слова может быть указано несколько его значений и наоборот. Со стороны клиента вводятся последовательно английские (русские) слова. Для каждого из них выведите на консоль все русские (английские) значения слова.
  2. ***Словари*.** В двух различных базах данных хранятся два словаря: рус­ско-белорусский и белорусско-русский. Клиент вводит слово и выбирает язык. Выведите перевод этого слова.
  3. ***Стихотворения*.** В БД хранятся несколько стихотворений с указанием ав­тора и года создания. Для хранения стихотворений использовать объекты типа Blob. Клиент выбирает автора и критерий поиска.
     + В каком из стихотворений больше всего восклицательных предложений?
     + В каком из стихотворений меньше всего повествовательных предложений?
     + Есть ли среди стихотворений сонеты и сколько их?

### Задание 5. Создание WEB-объектов

Задания:

5.1. Загрузка файла на сервер. Просмотр каталога из клиентской формы. Выбор файла и загрузка на сервер

5.2. Система отправки писем Send Mail. Отправка e-mail. E-mail с вложенными файлами. Интерфейс:

Имя.

e-mail.

Отправить.

Обратная связь или «контакты».

5.3. Рекомендация другу. На указанный e-mail оправляется письмо со ссылкой и рекомендациями, подписанные вами.

5.4. Система рассылки SMS-сообщений. Поддержка мобильной связи.

5.5. Система рассылки ICQ-сообщений.

5.6. Получение информации о клиенте и сервере.

5.7. Регистрация на сайте. Заполнить регистрационную форму: Фамилия, Имя, Город, E-Mail, login.

Система должна сгенерировать уникальный password. После регистрации ото­слать password по E-Mail.

5.8. Авторизация посетителя на сайте (логин, пароль, смена пароля и т.д.).

5.9. Счетчик посещений Counter – текстовый и графический.

5.10. HTTP-аутентификация и приветствие зарегистрированного клиента.

Интерфейс: Не видели Вас две недели, Сэр. Рады видеть Вас снова на нашем сайте.

5.11. Удаленный web-counter. Необходимо реализовать службу наподобие RAMBLER:TOP100. Интерфейс: пользователь регистрируется и получает идентифи­катор. Таким образом он связывает свою страницу с сервером-носителем web-couter-а. После этого служба должна вести учет посетителей страницы пользователя.

Необходимо осуществить администраторские функции по включению и исклю­чению произвольного пользователя администратором.

5.12. Откуда заходят посетители или поисковые машины. Адрес, IP-адрес, госу­дарство.

5.13. Подсчет online посетителей на сайте.

5.14. Администрируемая Гостевая книга.

Имя:

e-mail:

Сайт:

Город:

Сообщение:

Отправить

Интерфейс: пользователь заходит на страницу гостевой книги, на которой он сможет ввести сообщение, а также просмотреть сообщения, оставленные другими пользователями. Создайте две кнопки «Ввести» и «Просмотреть».

Должны быть выполнены такие администраторские функции, как включение и исключение сообщения в гостевой книге, запрет и управление запретом на включе­ние сообщения, содержащие определяемые администатором выражения, запрет и управление запретом на включение сообщения с заданных администатором хостов.

Помимо указанных пунктов можно предусмотреть:

Ответ администратора, Шаблон вывода, Фильтр слов, Смайлы

5.15. Голосование.

Интерфейс:

Как жизнь?

Отлично/.Хорошо/.Средне/.Плохо/.Хуже плохого

Проголосовать

Для реализации голосования продумать детали:

структуру базы или файлов;

логику скрипта (вывод результатов, само голосование, администрирование, архив);

интерфейс пользователя;

анти-флуд (для защиты от накрутки);

администрирование – создается как отдельный скрипт, который защищается паролем и содержит формы управления голосованием. Голосование делается на оп­ределенный срок, потом оно неактивно. Оно может принудительно закрываться ад­мином.

5.16. Система оценки материалов, расположенных на сайте.

5.17. Тесты и викторины.

5.18. FTP-клиент на PHP. FTP. Список файлов каталога. Загрузка файла с FTP-сервера. Загрузка файла на FTP-сервер.

5.19. Система отслеживания адресов, по которым скачиваются копии текста из данного сайта.

5.20. Файловый менеджер:

удалять файлы с сайта/ копировать/ переименовать/ создавать новые папки/загрузка файлов на сайт

5.21. Электронная почта и список рассылки.

Список рассылки представляет собой систему регистрации удаленных пользо­вателей для получения ими электронных писем с информацией. Интерфейс: произвольный пользователь заходит на страницу списка рассылки, регистрирует там свой E-MAIL. Администратор заходит на страницу администрирования, набирает письмо и отправляет зарегистрированным пользователям. Необходимо реализовать простейшие администраторские функции: исключение пользователя из списка рассылки, включение в список (подписка и отписка), сохранение отправляемого сообщения (с возможным архивированием), операции связанные с авторизацией (смена пароля).

5.22. Доска объявлений.

5.23. Знакомства. Служба знакомств.

5.24. Чаты и общения.

5.25. Forum & FAQ.

Струтура базы для форума.

Таблица пользователя:

|ID\_user|User\_pass|User\_Name

Таблица форумов:

|ID\_forum|Forum\_name

Таблица тем

|ID\_theme|ID\_foruma|ID\_Autor|theme\_name|theme\_text

Посты:

|ID\_post|Post\_data|Post\_autor|Post\_text|

5.26. Навигация на PHP: различные меню, карта сайта.

5.27. Cookie. Установка cookie. Чтение cookie. Срок действия cookie. Удаление.

5.28. Сеансы. Сохранение данных сеансов. Создание счетчика посещений.

Сеансы без cookie. Удаление данных сеансов.

5.29. Система аутентификации пользователей с помощью сеансов и cookie.

5.30. Перевод с русского на транслит и обратно «text на translite».

5.31. Количество кликов по ссылкам на другие сайты.

5.32. Калькулятор.

5.33. Фотогалерея.

5.34. Календари и органайзеры.

5.35. Словарь-переводчик.

5.36. Система поиска для веб-сервера. Поиск производится только для документов, хранящихся на сервере-носителе, и должен поддерживаться поиск по сложным и регулярным выражениям.

5.37. Рубрикатор: система поиска по первой букве (A,B,C…).