Тема 1. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ ЯЗЫК СЦЕНАРИЕВ PHP

Язык программирования PHP используется для создания скриптов, работающих на стороне сервера. Он поддерживается почти на всех известных платформах, почти во всех операционных системах, на самых разных серверах и по праву считается самым популярным языком, используемым для Web-программирования.

Первая версия языка появилась в 1995 году. Сегодня распространена версия PHP 4.0 и уже вышли в свет бета-версии PHP 5.0. В аббревиатуре PHP используется рекурсивным акронимом: Hypertext Preprocessor, что значит "PHP: препроцессор гипертекста ".

Возможности языка охватывают почти все области интернет-сервисов. С помощью PHP можно создавать изображения, PDF-файлы, флэш-ролики, в него включена поддержка большого числа современных баз данных, встроены функции для работы с текстовыми данными любых форматов, включая XML, и функции для работы с файловой системой. PHP поддерживает взаимодействие с различными сервисами посредством соответствующих протоколов, таких как протокол управления доступом к директориям LDAP, протокол работы с сетевым оборудованием SNMP, протоколы передачи сообщений IMAP, NNTPи POP3, протокол передачи гипертекста HTTP и т.д. Для создания приложений электронной коммерции существует ряд полезных функций, таких как функции осуществления электронных платежей.

PHP очень прост в изучении. Достаточно ознакомиться лишь с основными правилами синтаксиса и принципами его работы, и можно начинать писать собственные программы, причем браться за такие задачи, решение которых на другом языке требовало бы серьезной подготовки.

В качестве практической основы курса будут рассмотрены задачи, решаемые с помощью технологии клиент- сервер, и PHP соответственно будет использоваться для создания скриптов, обрабатываемых сервером. Поэтому необходимо на рабочем компьютере установить web-сервер и интерпретатор PHP. В качестве web-сервера используется Apache, как наиболее популярный среди web-разработчиков. Для просмотра результатов работы программ понадобится web-браузер, например Internet Explorer.

Так как установка серверного окружения специальная задача, то для тех, кто не желает вникать в устройство PHP и требуемых для его работы компонентов, существуют готовые дистрибутивы, содержащие все наиболее распространенные расширения PHP. Один из самых известных – дистрибутив Денвер (http://dklab.ru/chicken/web/). Инструкции по его установке можно прочитать на сайте разработчиков.

Пособие построено в форме самоучителя. Предполагается, что студены, изучая его, будут тестировать рассматриваемые примеры.

1.1. Начальные сведения

Как и HTML-документы, PHP-программы состоят из простого текста, поэтому для их написания можно использовать любой текстовый редактор, например Блокнот, если вы работаете в Windows.

Разберем простейшую PHP-программу:

Пример 1.1.1. Первая PHP-программа

```
<?php
  print "Люди чаще капитулируют, чем терпят крушение. ";
?>
```

После набора текста программы, файл необходимо сохранить, задав ему некоторое осмысленное имя и правильное расширение. Например: myprog.php. Далее необходимо скопировать файл на сервер. После этого к нему можно обратиться с помощью браузера. На рис. 1.1.1 показан результат работы программы на экране браузера.

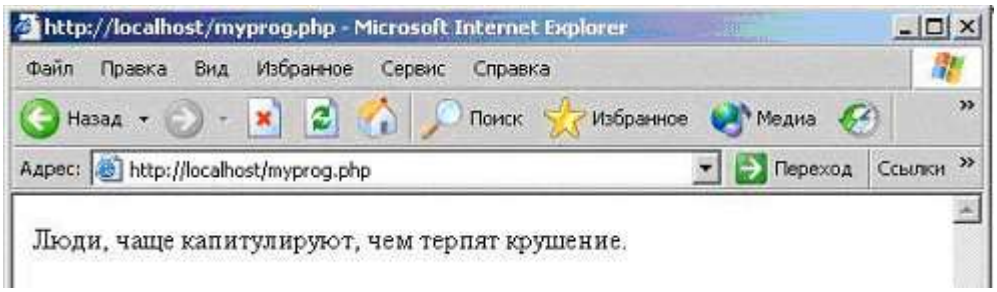


Рис. 1.1.1 Результат работы программы

Рассмотрим текст программы внимательнее.

При написании PHP-программы необходимо сообщить интерпретатору, как отличить команды, которые он должен обрабатывать, от простого HTML-текста. В противном случае команды будут приняты за HTML-текст и переданы браузеру. Для этого код PHP заключается в специальные тэги <?php и ?>. После тэга начала сценария (<?php) в программе следует функция print, которая предназначена для вывода данных в окно браузера.

Функция – это команда, действие, которой зависит от переданных ей данных. Данные, передаваемые функции, называются ее аргументом и помещаются в скобках после имени функции. В нашем примере функции print() передается строка, каждая строка должна быть заключена в кавычки двойные или одинарные. Функция print единственная функция, где скобки, в которых должен быть заключен аргумент, можно опускать. Заканчивается единственная строка нашей программы точкой с запятой, это сделано для того, чтобы сообщить интерпретатору об окончании инструкций. Последнее, что есть в нашей программе это закрывающий тэг(?>).

Примечание. Вместо функции print можно использовать функцию echo, которую чаще называют оператором или конструкцией. В наших примерах

использование print и echo абсолютно идентично и является делом вкуса каждого. В 4-ой версии PHP можно использовать укороченную версию оператора вывода, например:<? ="Всем привет!"?>. Объясняют это тем, что вывод текста – отнюдь не основная задача программирования.

Взаимодействие PHP и HTML

Рассмотренный нами пример 1.1.1 состоит только из PHP-кода. Однако возможности PHP позволяют создать скрипт, в котором помимо команд PHP будет присутствовать HTML-текст. Рассмотрим пример 1.1.2, позволяющий убедиться, что в создании такого симбиоза нет ничего сложного.

Пример 1.1.2. Скрипт, содержащий PHP-команды и HTML-текст

```
<html>
  <head>
     <title> Пример 1.1.2 </title>
  </head>
<body>
  <h1>
    <?php
          print "Люди чаще капитулируют, чем терпят крушение. ";
    ?>
  </h1>
</body>
</html>
```

Для того чтобы создать смешанный документ достаточно просто добавить HTML-текст перед открывающим (<?php) или после закрывающего (?>) тэгов PHP. Интерпретатор PHP игнорирует все, что находится вне тэгов языка. Результатом работы данной программы является отображение нашего предложения (Люди чаще капитулируют, чем терпят крушение), выделенного как заголовок первого уровня. Просмотрим появившуюся в браузере информацию, как исходный HTML-текст (см. рис. 1.1.2).

Он выглядит как обычный HTML-документ. Интерпретатор PHP преобразует PHP-команды в обычный HTML-текст, который и передается браузеру. Количество блоков PHP команд, которые можно включать в HTML-документ, ничем не ограничено.

Следует учесть, что все, что вы определите в первом PHP блоке, будет доступно программе в последующих блоках.

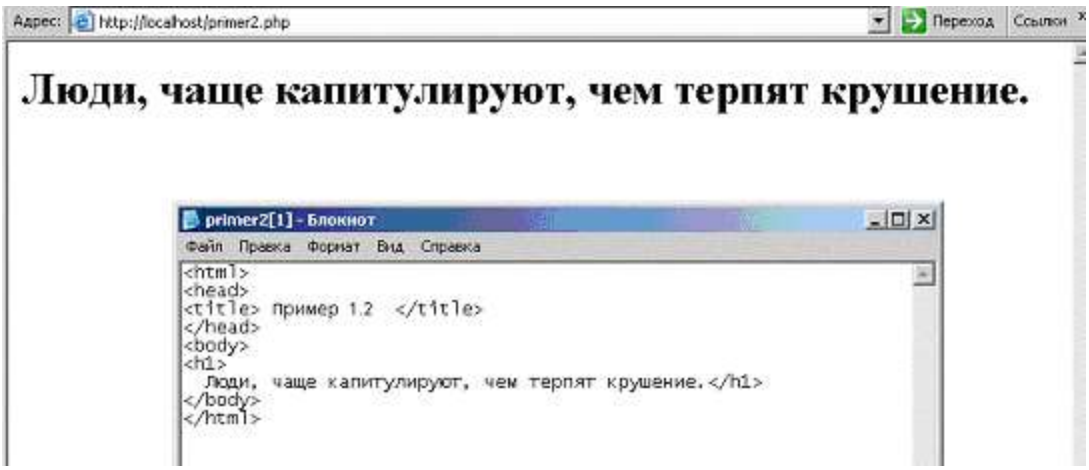


Рис. 1.1.2 Вывод программы из примера 1.2 в виде исходного текста

Задание для самопроверки

Вывести на экран браузера три различных предложения так, чтобы первое было выделено курсивом, второе имело шрифт и размер отличный от шрифта и размера первого предложения, а третье имело уникальный, по отношению к первым двум предложениям, цвет. Использовать для вывода предложений функцию print или echo.

1.2. Элементы языка PHP

Переменные

Немного изменим пример 1.1.1.

Пример 1.2.1 Знакомство с переменными

```
<?php
    $text = "Люди чаще капитулируют, чем терпят крушение. ";
print $text;
?>
```

Визуально код программы стал больше на одну строчку, а также изменился аргумент функции print. Если набрать эту программу и сравнить результат ее работы с результатом работы примера 1.1.1, то вы не увидите никакой разницы. Проанализируем, как это работает.

За открывающим тэгом языка PHP следует строчка, в которой мы создаем переменную $text и записываем с помощью оператора присваивания (знак равенства) в нее наше предложение.

Переменная – это средство для хранения данных определенного типа. Все данные, с которыми работает программа, хранятся в виде переменных. Значение переменной можно изменять в любой момент. Имя переменной должно обязательно начинаться со знака доллара $. Имя переменной может

состоять только из символов латинского алфавита, цифр и знака подчеркивания и не должно начинаться с цифры. После строчки, в которой мы создаем переменную и присваиваем ей первоначальное значение, идет функция print. В качестве аргумента у данной функции присутствует переменная $text. Функция print просто выводит на экран содержимое переменной $text. Последней строчкой идет закрывающий тэг PHP.

Заметим, что в текстовые переменные можно включать тэги для их оформления. Тэги можно использовать и непосредственно в функции print, если аргумент заключить в кавычки.

Чтобы закрепить знания, посмотрим на пример 1.2.2.

Пример 1.2.2 Знакомство с переменными

```
<?php
    $text = "Люди чаще капитулируют, чем терпят крушение. ";
$a =  2345;
$bcd_dfg = "ш";
$_6578 = "<b> терпят </b>";

print $text;
print $a;
print "<h1> $bcd_dfg <h1>";
print $_6578;
?>
```

В этом примере создано уже четыре переменные ($text, $a, $bcd_dfg, $_6578). Для простоты разделим весь ряд значений, которые можно занести в переменные, на два типа текстовые (берутся в кавычки) и числовые (в кавычки не берутся, представляют собой различные числа). В нашем примере числовым значением обладает только переменная $a. Над числовыми переменными, как правило, выполняются арифметические операции.

Константы

Использование переменных прекрасный способ для хранения и обработки данных. Их значения можно изменять по мере надобности в любое время. Впрочем, может сложиться такая ситуация, что вам понадобится иметь такое значение, которое вы категорически не захотели бы менять. В этом случае вам надо создать константу.

Рассмотрим пример, в котором используют не переменную, а константу.

Пример 1.2.3 Создание константы

```
<?php
define("NAME", "Masha");
print NAME;
```

?>

Результат работы программы – появление на экране браузера слова Masha.

Сначала с помощью функции define, создается константа. У данной функции первым аргументом идет имя константы, а вторым аргументом значение этой константы. Заметим, что имена констант принято обозначать прописными буками.

Операторы и выражения

Оператором называется символ или последовательность символов, с помощью которых из одного или нескольких значений получается новое значение.

Операндом называется значение или переменная, к которому применяется оператор. Обычно у оператора бывает несколько операндов.

Получим новое значение с помощью оператора умножения и двух операндов: 4 * 10. В данном случае операнды - 4 и 10, а * - оператор. В результате мы получаем новое значение – число 40.

Выражением называется комбинация операндов и операторов, производящая некоторое значение. Однако не обязательно для образования выражения использовать операторы. Выражением в PHP является все, что имеет некоторое значение. Например: $text (переменная), функция print() – все это выражения.

Арифметические операторы

Переменные, имеющие числовые значения, идеально подходят для арифметических операций. Рассмотрим пример 1.2.4.

Пример 1.2.4 Арифметические операторы.

```
<?php
$a = 5; $b = 6;
$summa = 0; $summa = $a + $b;
print $summa;
?>
```

В этом примере имаются три переменные ($a, $b, $summa). При создании переменных мы присвоили им значения (5, 6 и 0 – соответственно). Затем следует строчка $summa = $a + $b. С точки зрения логики, очевидно, что мы хотим получить сумму $a и $b и записать получившееся значение в переменную $summa, при этом предыдущее значение $summa стирается. Для получения нового значения переменной $summa используется оператор присваивания (знак равенства) и арифметический оператор сложения (знак плюса). Над переменными PHP можно проводить различные математические операции (например, сложение, вычитание, умножение, деление). При этом арифметическими операторами являются знаки: +, -, *.

Например: $summa = $a + $b*($b + $a – 6)/$a*$b. Порядок вычисления будет аналогичен порядку вычисления в обычном математическом выраже-

нии. Проверьте себя – результат должен быть равен 41. А теперь отредактируйте пример 1.2.4 и убедитесь в этом.

Оператор конкатенации

Над текстовыми переменными тоже можно выполнять различные действия. Рассмотрим небольшой пример.

Пример 1.2.5 Оператор конкатенации.

```
<?php
$text = "люди чаще капитулируют, чем терпят крушение. ";
print "Я думаю,  ".$text;
?>
```

Здесь мы видим незнакомый момент только в аргументе функции print(). С помощью оператора конкатенации (обыкновенная точка после кавычек) мы присоединяем к небольшому кусочку текста: "Я думаю, " значение переменной $text. Наконец, функция print выводит все в одной строчке: "Я думаю, люди чаще капитулируют, чем терпят крушение".

Этот оператор объединяет две строки, присоединяя правую строку к левой. Независимо от типа своих операндов, оператор конкатенации всегда обрабатывает их как строки.

Например: "new "."home" после работы оператора будет: new home

Операторы сравнения

Операторы сравнения предназначены для того, чтобы сравнивать значения своих операндов. Они возвращают true (истина, в PHP это любое ненулевое значение), если сравнение успешно, и false (ложь, в PHP это пустая строчка) – в противном случае.

Пример 1.2.6. Оператор сравнения

```
<?php
$a = 1000;
print ($a = = 1000);
?>
```

В данном случае переменной $a присваивается значение 1000. Далее мы просим функцию print вывести на экран браузера результат работы условного оператора сравнения (знак двойного равенства), который спрашивает у нашей переменной, равно ли 1000 ее значение или нет. Так как значение переменной равно 1000, то на экране браузера появляется 1 (истина). Если бы присвоили $a любое значение отличное от 1000, то при сравнении получили бы false (ложь), на экране браузера не отобразилось бы ничего. Можно проводить сравнения двух операндов не только на равенство, но и на то кто них больше или меньше или на их неравенство. В качестве операндов могут фигурировать как переменные, так и целые выражения. Основные операторы приведены в табл. 1.2.1.

Таблица 1.2.1

Операторы сравнения

Оператор	Название	Условие выполнения	Пример
= =	Равенство	Левый операнд равен правому	$x = = 34
!=	Неравенство	Левый операнд не равен правому	$x != 34
>	Больше чем	Левый операнд больше правого	$x > 34
>=	Больше или равно	Левый операнд больше или равен правому	$x >= 34
<	Меньше чем	Левый операнд меньше правого	$x < 34
<=	Меньше или равно	Левый операнд меньше или равен правому	$x <= 34

Задание для самопроверки

1. Создайте десять переменных имеющих различные текстовые значения и выведите их на экран браузера одной строкой.

2. Создайте две числовые переменные. Примените к ним различные операторы сравнения. Используйте функцию print или echo для вывода результатов сравнения.

1.3.Управление потоком

Условные инструкции

Программы, с которыми мы до сих пор имели дело, работали очень просто. Одни и те же инструкции выполнялись в одном и том же порядке при каждом выполнении программы. Большинство программ изменяют свое поведение в зависимости от изменяющихся условий, и для этого им приходится вычислять значения некоторых выражений. Способность реагировать на определенные условия и принимать решения делает Web-проекты по-настоящему динамичными. PHP осуществляет это с помощью инструкции if.

Инструкция if

Пример 1.3.1 Инструкция if

```
<?php
$flat = "большая";
if ($flat == "большая")
   {
   print "У меня большая квартира";
   }
?>
```

В этой небольшой программе создана единственная переменная $flat, которой присвоено текстовое значение (большая). Далее следующей строчкой начинается блок инструкций if, который продолжается вплоть до закрывающего тэга PHP. Давайте рассмотрим этот блок более подробно. Для этого изобразим его структурно.

```
if (выражение)
```

```
{
// этот фрагмент выполняется, если выражение истинно
}
```

Идет сам if, в скобках содержится условное выражение ($flat = = "большая"), мы проверяем значение переменной $flat. Если условный оператор вернет нам true, то инструкция if заставит выполниться целый блок идущий сразу за ней и обрамленный фигурными скобками. Иначе говоря, if можно для удобства восприятия мысленно заменить словом если. Получается, если истина, то происходит то, что стоит в фигурных скобках, если ложь, то ничего не происходит.

Относительно нашего примера можно сказать, что с помощью оператора сравнения мы сравниваем значение переменной $flat со строкой "большая". Они совпадают (истина), поэтому блок программы, следующий за инструкцией if, выполняется. Результат работы программы вывод предложения – у меня большая квартира.

При написании инструкции if, может понадобиться указать альтернативный блок инструкций, который следует выполнить в том случае, если условие не выполняется. Для этого после блока if нужно поместить блок else.

Примечание. Выше в примере программы использован однострочный комментарий в стиле языка C++. PHP поддерживает еще два вида комментариев, в стиле Unix и языка Си.

Итак,

```
// Это однострочный комментарий в стиле C++
#  Это тоже однострочный комментарий в стиле Unix
/* А это многострочный
    комментарий */
```

Блок else инструкции if

Изменим пример 1.3.1, добавив в него блок else.

Пример 1.3.2 Инструкция if с блоком else

```
<?php
$flat = "маленькая";
if ($flat == "большая")
{
print "У меня большая квартира";
}
else
{
print "У меня квартира $flat";
}
?>
```

Результатом работы программы является вывод на экран браузера предложения – У меня квартира маленькая.

В переменной $flat записана строка "маленькая" – она не совпадает со строкой "большая", поэтому условие не выполняется. Это означает, что первый блок инструкций пропускается и выполняется блок инструкций, следующий за словом else. Использование блока else с инструкцией if позволяет программам принимать довольно сложные решения, однако пока мы ограничены бинарным выбором (да - нет).

Структурно это все выглядит так:

```
if (выражение)
{
// этот фрагмент выполняется, если выражение истинно
}
else
{
// этот фрагмент выполняется, если выражение ложно
}
```

Блок elseif инструкции if

С помощью конструкции if-elseif-else можно проверить несколько условий перед тем, как выполнить фрагмент программы.

Модифицируем пример 1.3.2, добавив в него блок elseif.

Пример 1.3.3 Инструкция if с блоком elseif и с блоком else

```
<?php
$flat = "маленькая";
if ($flat == "большая")
{
print "У меня большая квартира";
}
elseif ($flat == "средняя")
{
print "У меня средняя квартира";
}
else
{
print "У меня квартира $flat";
}
?>
```

Эту ситуацию можно кратко обрисовать так: сначала спрашивают большая квартира или нет, если ДА, то выполняется первый блок, если НЕТ, то тогда начинают проверять второе выражение(спрашивают, может квартира

средняя), если ДА, то выполняется второй блок, идущий за elseif. В том случае, когда ни одно из проверяемых выражений не является истинным, начинает работать блок else.

В общем виде работу оператора elseif можно представить так:

```
if (выражение_1)
{// этот фрагмент выполняется, если выражение_1 истинно}
elseif (выражение_2)
{// этот фрагмент выполняется, если выражение_1 ложно, а выражение_2 истинно }
else
{// этот фрагмент выполняется во всех остальных случаях }
```

Если первое выражение не истинно, то первый блок игнорируется. В блоке elseif вычисляется второе выражение, и если оно истинно, то выполняется соответствующий блок программы, иначе выполняется блок инструкций else. Блоков elseif может быть сколько угодно, а блок else может отсутствовать, если в нем нет необходимости.

Циклы

Иногда нам требуется несколько раз повторить тот или иной фрагмент программы. Для многократного повторения фрагмента программы служат так называемые циклы. Цикл – это группа команд, многократно выполняемых компьютером, пока некоторое условие продолжения цикла остается истинным. Существуют два типа циклов:

Цикл, управляемый счетчиком.

Цикл, управляемый контрольным значением.

Первый случай, характеризуется использованием специальной переменной, называемой счетчиком. Счетчик определяет, сколько раз должен выполниться цикл. Поэтому можно определить количество повторений до начала выполнения цикла. Счетчик увеличивается (обычно на 1) всякий раз, когда выполняется тело цикла.

Многократно исполняемые строчки программы, количество повторений которых зависит либо от контрольного значения, либо от счетчика, называются телом цикла. Когда значение счетчика показывает, что было выполнено соответствующее число повторений, цикл завершается и программа продолжает свою работу со строчки, следующей сразу за циклом.

Второй случай, характеризуется тем, что число повторений заранее не известно. Тело цикла выполняется до тех пор, пока контрольное выражение не станет ложным (например, пока одна переменная больше другой).

Цикл for

Цикл for является циклом, управляемым счетчиком. Рассмотрим и разберем небольшой пример.

Пример 1.3.4 Цикл for

```
<?php
for ($x = 1; $x <= 10; $x = $x +1)
{           print $x;
    }
?>
```

В первой строчке программы сразу за ключевым словом for следуют скобки. В этих скобках содержатся три выражения, разделенные точкой с запятой. В первом выражении счетчику цикла присваивается некоторое начальное значение, во втором – проверяется условие цикла, а в третьем происходит увеличение или уменьшение счетчика (подготовка следующего цикла).

В нашем случае в первом выражении мы инициализируем счетчик цикла (создаем переменную $x, и присваиваем ей некоторое начальное значение), во втором пишем тестовое выражение (цикл должен выполняться, пока $x <= 10), в третьем происходит увеличение счетчика на единицу. Далее идет само тело цикла, заключенное в фигурные скобки. Тело цикла будет выполняться, пока значение счетчика не станет равным 11. Условие выполнения цикла станет ложным и цикл закончится. Результатом работы программы будет вывод: 12345678910.

Зачем мы увеличиваем счетчик? Это сделано для того, чтобы наш цикл не стал бесконечным. Если мы не будем этого делать, то значение $x всегда будет равно единице, а значит, выражение $x <= 10 всегда будет истинно, а это, в свою очередь, приведет к бесконечности цикла for. Тело цикла будет выполняться бесконечно.

Структурно наш цикл выглядит так:

```
for (инициализация; условие выполнения цикла; увеличение)
{
// тело цикла
}
```

Цикл while

Основное отличие цикла while от цикла for состоит в том, что цикл while чаще всего применяется, когда количество циклов заранее не известно и он управляется контрольным значением.

Пример 1.3.5 Цикл while

```
<?php
$y = 10;
```

```
$x = 1;
while($x!=$y)
{
print "$x<br>";
print "$y<hr>";
$y=$y-2;
$x=$x+1;
}
?>
```

Сначала мы инициализируем (создаем и присваиваем начальное значение) переменные $x и $y. Далее идет сам цикл while. Он работает следующим образом, пока контрольное выражение в круглых скобках истинно (в нашем случае пока $x не равно $y), тело цикла выполняется. В нашем случае, пока выражение истинно происходит вывод на экран текущего значения переменных, увеличение значения $x на 1 ($x =$x + 1) и уменьшение значения $y на 2 ($y=$y-2).

Результатом работы программы будет вывод чисел: 1 10 2 8 3 6. Но, чтобы отделить числа, выводимые на каждом цикле, здесь использованы тэги разрыва строки и горизонтальной линии (
 и <hr>). Как только значения переменных $x и $y становятся равными (это произойдет, когда обе переменные будут равны 4), цикл прекращает свою работу.

Цикл while по своей структуре весьма напоминает условную конструкцию if.

```
while(контрольное выражение)
{
// тело цикла
}
```

Задание для самопроверки

1. Создайте программу, позволяющую вывести на экран браузера наибольшую числовую переменную. Таких переменных должно быть, по крайней мере, пять.

2. Напишите программу, которая выводила бы все нечетные числа от 1 до 100.

1.4.Функции

Функции – это самый главный компонент хорошо организованной программы. Они облегчают чтение текста программы и позволяют многократно использовать отдельные ее части.

Функция – это блок инструкций, который программа может вызвать. При вызове данный блок инструкций выполняется. Функции можно передать некоторые значения, и она будет их обрабатывать. Закончив работу, функция может вернуть полученное значение в точку ее вызова.

Функции бывают двух видов – встроенные, т.е. получаемые вместе с самим PHP, и те, которые вы пишете самостоятельно – пользовательские. Вспомним, что пример 1.1 состоял только из единственного вызова функции.

Пример 1.4.1. Первая PHP-программа

```
<?php
        print "Люди чаще капитулируют, чем терпят крушение. ";
?>
```

В данном примере мы вызвали функцию print(), передав ей текстовую строку. Вызов функции состоит из имени функции, в данном случае print, и следующей за ним пары скобок (в случае с функцией print эти скобки можно опускать). Если вы хотите передать функции некоторую информацию, то должны поместить ее между скобками. Вся информация, которая находится между скобками функции, называется аргументом функции. Иначе говоря, аргумент функции – это значение, передаваемое функции при вызове.

Создание пользовательской функции

Попробуем самостоятельно создать некоторую функцию. Разберем небольшой пример.

Пример 1.4.1 Определение функции

```
<?php
function bigprint()
{
    print "<h1> Люди чаще капитулируют, чем терпят крушение.</h1>";
}
bigprint();
?>
```

Для создания (определения) функции используется ключевое слово function. Имя функции указывается после ключевого слова function, а после него следует пара скобок. Если вы хотите, чтобы ваша функция принимала несколько аргументов, то должны поместить в скобках несколько имен переменных, разделив их запятыми. Потом, в теле функции, эти переменные получат те значения, которые вы укажете при вызове функции. Даже если ваша функция не использует никаких аргументов (как в нашем примере), вы все равно должны после ее имени поставить пару скобок.

Наша программа просто выводит на экран браузера строку, обрамленную тэгами заголовков <h1>. В программе определена функция bigprint(), не использующая аргументов. Поэтому скобки были оставлены пустыми. После строчки, в которой мы задаем имя и список принимаемых аргументов нашей функции – function bigprint(), идет тело функции, обрамленное фигурными

скобками. В теле функции происходит вызов функции print, которой в качестве аргумента передается строчка, заключенная между тегами <h1>.

Последней строчкой нашей программы является непосредственный вызов функции bigprint().

Рассмотрим пример определения функции, которая принимает аргументы и делает с ними нечто осмысленное.

Пример 1.4.2 Определение функции с аргументами

```
<?php
function bigprint($text)
{
    print "<h1> $text</h1>";
}
bigprint("Люди чаще капитулируют, чем терпят крушение. ");
bigprint("У счастливых цифр много нулей. ");
?>
```

Теперь наша функция bigprint должна получать аргумент – строку, поэтому необходимо было при определении функции поместить в скобки переменную $text. Значение, переданное функции при вызове, будет записано в эту переменную. В теле функции мы выводим переменную $text, обрамленную тэгом заголовка <h1>. Так как мы вызывали функцию bigprint два раза, то результатом работы данной программы будут две разные строчки, оформленные как заголовки первого уровня.

Область видимости переменных

Переменная, созданная внутри некоторой функции, становится локальной по отношению к данной функции. Это означает, что она недоступна ни для других функции, ни для любого фрагмента программы вне функции, в которой она создана.

Приведем пример.

Пример 1.4.3 Область видимости переменных

```
<?php
function tester()
{
    $text = "Люди чаще капитулируют, чем терпят крушение. ";
}
print $text;
?>
```

Результатом работы данной программы будет пустая строчка. Мы не увидим строчки, «Люди чаще капитулируют, чем терпят крушение». Это значение локальной (созданной внутри функции) переменной $text, а значит, не может быть доступно вне тела нашей функции tester(). Если мы захотим вывести значение каких-то переменных созданных внутри функции на экран браузера, нам необходимо поместить функцию print внутри тела функции, которой принадлежат данные переменные.

Пример 1.4.4. Область видимости переменных 2

```
<?php
function tester()
{
    $text = "Люди чаще капитулируют, чем терпят крушение. ";
print $text;
}
tester();
?>
```

В этом случае все будет работать предельно корректно, значение переменной $text будет выведено на экран браузера. Но такая функция не сможет выводить разные значения, все переменные определены внутри ее тела.

Другой способ сделать внутреннюю переменную видимой в теле программ мы это использовать инструкцию return. С ее помощью можно "вернуть любые объекты". Вот пример функции, возвращающей квадрат своего аргумента.

Пример 1.4.5

```
function mysqrt($a)
{      return $a*$a;}
print mysqrt(4);
```

В результате вызов функции в операторе вывода даст ответ 16.

Задание для самопроверки

Напишите функцию, имеющую в качестве аргументов две числовые переменные. Данная функция должна выводить наибольшую из переменных на экран браузера. Сравните не меньше трех пар чисел.

1.5. Массивы

У переменной есть один существенный недостаток, в ней одновременно можно хранить только одно значение. Однако существует особая разновидность переменных, позволяющая обойти это ограничение. Массив дает возможность хранить сколько угодно значений под одним и тем же именем.

Для чего нужны массивы? В первую очередь, для гибкости - вы можете сохранить два значения, а можете и две сотни, не заводя новых переменных. Кроме того, массив дает возможность эффективно обрабатывать эти переменные. Можно просмотреть их все в цикле, выбрать любую нужную или от-

сортировать в числовом или алфавитном порядке, можно даже указать свой собственный порядок сортировки.

Доступ к любому элементу массива осуществляется по его индексу. Индекс (ключ) служит для однозначной идентификации элемента внутри массива. Каждый элемент массива имеет свой уникальный индекс. Чаще всего индексом является некоторое целое число. Следует учесть, что первому элементу массива присваивается нулевой индекс, второй элемент массива имеет индекс равный 1, третий – 2 и т.д. Для того чтобы начать работать с каким-то элементом массива, достаточно просто знать его индекс.

Создание массива

Элементам массива могут быть присвоены значения двумя способами – непосредственно или с помощью функции array(). Рассмотрим оба способа.

Определение массива с помощью функции array()

Пример 1.5.1 Определение массива с помощью функции array()

```
<?php
$students = array("Ira","Vadim","Alex","Ann");
print $students[0];
?>
```

Данный пример состоит всего из двух строчек. В первой строке мы создаем массив с помощью функции array(), в скобках которой содержатся элементы нашего массива. Имя массива задается перед знаком равенства ($students), правила создания имени массива схожи с правилами создания имени переменной. Во второй строчке мы просим функцию print() вывести элемент массива с индексом 0 (это означает, что мы просим вывести первый элемент массива).

Индекс элемента массива, к которому происходит обращение, указывается в квадратных скобках после имени массива. Таким способом можно обращаться к элементу массива, как для получения его значения, так и для присвоения ему значения. В нашем примере, Ira имеет индекс 0, Vadim –1, Alex – 2, Ann – 3. Очевидно, запросив вызов на экран значения массива имеющего индекс 0, мы увидим на экране браузера имя Ira.

Создание элементов массива с помощью идентификатора

Существует возможность создать новый массив или добавить элемент к тому, который уже есть, с помощью идентификатора (имени) массива. Для этого достаточно указать имя массива и пару пустых квадратных скобок. Давайте создадим массив $students таким способом.

Пример 1.5.2 Создание новых элементов с помощью идентификатора

```
<?php
$students[] = "Ira";
$students[] = "Vadim";
```

```
$students[] = "Alex";
$students[] = "Ann";
print $students[0];
?>
```

Обратите внимание на то, что мы не указываем номер элемента в квадратных скобках. PHP автоматически вычисляет его, освобождая нас от необходимости помнить о том, какой следующий элемент свободен.

Результат работы программы из примера 5.2 будет полностью аналогичен результату работы программы из примера 5.1.

После того как массив создан, можно добавлять к нему новые элементы. В примере 1.5.3 мы создаем массив с помощью функции array() и добавляем к нему новые элементы.

Пример 1.5.3 Добавление новых элементов

```
<?php
$students = array("Ira","Vadim","Alex","Ann");
$students[] = "Max";
print $students[4];
?>
```

Во второй строчке программы мы добавили в массив новый элемент, имеющий индекс 4 и значение Max. После этого, обратившись к нему по индексу, мы вывели его на печать.

Ассоциированные массивы

К любому элементу массива можно обратиться по индексу. Раньше в качестве индексов мы использовали целые числа. Это довольно удобно, но имеется один существенный недостаток этого метода. Представьте себе массив, элементы которого хранят телефоны ваших друзей. Допустим, вы хотите посмотреть телефон Макса. Для этого вам надо вспомнить, под каким индексом хранится этот телефон у вас в массиве. Когда телефонов всего 5 – 6 это еще реально, но когда их 50. В этом случае нам очень сильно могут помочь ассоциированные массивы. К каждому элементу такого массива можно обратиться по имени - ключу. Вы просто просите показать вам элемент массива под именем Макс и сразу получаете интересующий вас телефон. Удобно и просто.

Ассоциированный массив – это массив, к элементу которого можно обратиться по имени (ключу). Ассоциированный массив можно создать непосредственно или с помощью функции array().

Создание ассоциированного массива с помощью функции array()

Пример 1.5.4 Создание ассоциированного массива с помощью функции array()

```
<?php
$tel = array(
```

```
    "Max Koshelev" => "580-46-82",
    "Ann Reish" => "589-90-34",
    "Pashsa Golikov" => "480-57-58"
  );
print $tel["Max Koshelev"];
?>
```

Для того чтобы создать ассоциированный массив с помощью функции array(), нужно задать как имя (Max Koshelev, Ann Reish, Pashsa Golikov), так и значение (580-46-82, 589-90-34, 480-57-58) для каждого элемента. В нашем примере мы создаем массив $tel из трех элементов. К любому элементу массива можно обратиться по имени (ключу). Ключи – это строки, которые необходимо брать в кавычки. Ключ может состоять из нескольких слов (как у нас, фамилия и имя). Результат работы программы – появление на экране браузера телефона Кошелева Максима - 580-46-82.

Непосредственное создание ассоциированного массива

Создать новый массив или добавить к существующему пару – ключ/значение можно, просто присвоив значение элементу массива, указав этот элемент по имени ключа.

Пример 5.5 Непосредственное задание ассоциированного массива

```
<?php
$tel["Max Koshelev"] = "580-46-82";
$tel["Ann Reish"] = "589-90-34";
$tel["Pashsa Golikov"] = "480-57-58";
print $tel["Max Koshelev"];
?>
```

В данном примере мы опять создаем массив $tel.

Работа с массивами

Получение размера массива

К любому элементу массива можно обратиться по его номеру. Однако механизм работы с массивами настолько гибок, что иногда вы можете не знать, сколько элементов содержится в массиве. В этом случае на помощь приходит функция count(). Эта функция сообщает нам количество элементов массива.

Пример 1.5.6 Получение размера массива

```
<?php
$student["name"] = "Ann";
```

```
$student["surname"] = "Petrova";
$student["age"] = 20;
print count($student);
?>
```

В данном примере мы сначала создаем массив $student, а затем просим вывести на экран браузера результат работы функции count(). В качестве аргумента функции используем массив $student, количество элементов которого необходимо подсчитать. Результатом работы программы является появление на экране браузера числа 3.

Просмотр массива с помощью цикла foreach

Очень часто требуется вывести на экран браузера все элементы массива. Для выполнения этой операции лучше всего использовать инструкцию foreach.

Пример 1.5.7 Просмотр массива с помощью инструкции foreach

```
<?php
$students = array("Ira","Vadim","Alex","Anna);
foreach($students as $temp)
    {
print "$temp<br>";
}
?>
```

В данном примере мы сначала создаем массив $students, состоящий из четырех элементов. Затем мы используем инструкцию foreach. В данном случае $students – это имя массива, который нужно просмотреть, а $temp – переменная, где будет временно храниться значение каждого элемента. Инструкция foreach работает следующим образом: сначала значение каждого элемента массива временно помещается в переменную $temp, а потом выводится на печать с помощью функции print, содержащейся в теле (между фигурных скобок) инструкции foreach. Элементы массива перебираются последовательно, один за другим. Результатом работы данной программы будет вывод на экран браузера следующего:

Ira

Vadim

Alex

Anna

Просмотр в цикле ассоциированного массива

Для того чтобы просмотреть в цикле ассоциированный массив, нужно написать инструкцию foreach несколько по-другому. Дело в том, что нам

придется временно сохранять не только значение каждого элемента, но и его имя - ключ.

Пример 1.5.8 Просмотр ассоциированного массива

```
<?php
$tel = array(
      "Max Koshelev" => "580-46-82",
      "Ann Reish"=> "589-90-34",
      "Pashsa Golikov" => "480-57-58"
            );
foreach ($tel as $key=>$temp)
{
       print "$key - $temp<br>";
}
?>
```

Здесь $tel – это имя массива, $key – переменная (ключ), в которой сохраняется имя каждого элемента массива, а $temp – переменная, где временно сохраняются значения каждого элемента.

Вывод этой программы выглядит следующим образом:

Max Koshelev - 580-46-82

Ann Reish - 589-90-34

Pashsa Golikov - 480-57-58

Сортировка простого массива с помощью функции sort()

Сортировка данных (т.е. расположение данных в некотором специальном порядке, например, по возрастанию или убыванию) является одним из наиболее важных применений компьютера.

Функция sort() принимает всего один аргумент – массив – и сортирует его в алфавитном порядке, если хотя бы один, из числа его элементов является строкой, и в числовом порядке, если все его элементы - числа. Эта функция преобразует переданный массив.

Пример 1.5.9 Сортировка простого массива с помощью функции sort()

```
<?php
$massiv = array("c", "a", "b");
sort($massiv);
foreach($massiv as $temp)
{
print "$temp<br>";
}
?>
```

Результатом работы программы будет вывод на экран элементов отсортированного массива $massiv.

a

b

c

Функция sort() изменяет положение элементов внутри нашего массива. Если до начала сортировки при запросе элемента массива с нулевым индексом вы получили бы значение c, то после применения функции sort() результатом на аналогичный запрос будет значение a. Сортировка массива в обратном порядке выполняется функцией rsort().

Сортировка ассоциированного массива производится аналогично, только используются функции asort() (сортировка по значениям в порядке возрастания) и ksort() (сортировка по ключам). Для сортировки в обратном порядке используются функции arsort() и krsort() соответственно.

Задание для самопроверки

1. Создайте массив, состоящий из десяти элементов, причем значение каждого последующего элемента должно быть больше предыдущего в три раза, используйте при создании данного массива цикл. Осуществите вывод данного массива на экран браузера. Выведите элементы массива в обратном порядке

2. Создайте ассоциированный массив ИМЯ => РОСТ (например, Маша => 182), состоящий не менее чем из 6 элементов. Обеспечить вывод исходного массива, отсортированного по алфавиту и отсортированного в порядке убывания роста.

1.6.Работа с формами

Мы рассмотрели достаточное количество примеров, но всем им не хватало связи с реальностью. Пора приступать к работе. В Web-пространстве для передачи данных от посетителя Web-страницы на сервер используются HTML-формы. В PHP предусмотрены многочисленные средства для работы с формами.

Для работы с данными из формы в современных версиях PHP используются специальные переменные окружения, которые представляют собой ассоциативные массивы. Сейчас нас интересуют две такие переменные - $_POST и $_GET. Они используются при соответствующих методах передачи данных через форму (параметр method в теге form). Ключом в массивах является имя поля в форме, а значением – значение поля. Например, если в форме был выбран метод GET и было поле name для занесения имени пользователя, то файлу, указанному в параметре action тега form, будет передан массив с одним элементом. Доступен он будет следующим образом:

```
echo $_GET['name'];
```

А при методе POST:

```
echo $_ POST['name'];
```

Так чем же отличаются методы POST и GET? Об этом мы поговорим чуть позже.

Программа обработки данных, введенных пользователем

Для того чтобы материал был понятнее, до некоторого времени будем разделять HTML-текст и текст PHP-программы.
Пример 1.6.1 HTML-форма для ввода пароля доступа (файл primer6_1.html)

```
<html>
<head>
<title>  Form </title>
 </head>
 <body>
<form action="primer6_2.php" method="GET">
Введите пароль: <input type="password" name = "pas" size="10" Maxlength =
"10">
<input type="submit" value="Проверка пароля" name = "go" >
</form>
 </body>
</html>
```

Мы создали форму, в которой есть модифицированное текстовое поле (текстовое поле типа password позволяет скрыть ввод пароля от посторонних наблюдателей) с именем "pas" и кнопка передачи данных "submit"с именем "go". Сделаем некоторые пояснения.

Рассмотрим дескриптор <form>. Дескриптор или тэг– это простой элемент разметки, который всегда имеет такой вид: <дескриптор>. Контейнером называется пара дескрипторов HTML в форме: <дескриптор> </дескриптор>. Элемент <дескриптор> служит для включения, а </дескриптор> для выключения одного и того же объекта.

Дескриптор <form> быть в начале каждой формы. При создании этого дескриптора указывается имя файла, находящегося на сервере, который будет обрабатывать, получаемую из формы информацию, и способ пересылки информации, задаваемый атрибутами ACTION и METHOD.

ACTION – указывает на файл primer6_1.php, который должен обрабатывать данные полученные от формы. Поскольку мы указали здесь только имя файла, а путь к нему опустили, то этот файл должен находиться в том же каталоге на сервере, что и HTML–документы.

Атрибут METHOD – указывает, как следует отсылать информацию ее обработчику. Этот атрибут может принимать значение POST – при этом данные из формы передаются отдельно от URL обработчика, или GET – в этом случае информация из формы для передачи на сервер записывается в конец

URL обработчика. Если сказать проще, разница между ними заключатся в том, будете вы видеть имена и значения предаваемых переменных в адресной строке вверху окна браузера или нет.

Дескриптор <Input> необходим для получения информации от пользователя. Параметр Type устанавливает нужный тип поля ввода, Name – указывает имя поля, Size –размер поля ввода в символах, Maxlength – устанавливает максимальное количество символов, вводимых в поле ввода.

Submit – кнопка, с помощью которой можно отправить на сервер данные, введенные в форму. Используя это значение с атрибутом Value, на кнопке можно установить любой текст.

На приведенном ниже рисунке можно посмотреть, какая форма у нас в итоге получилась.

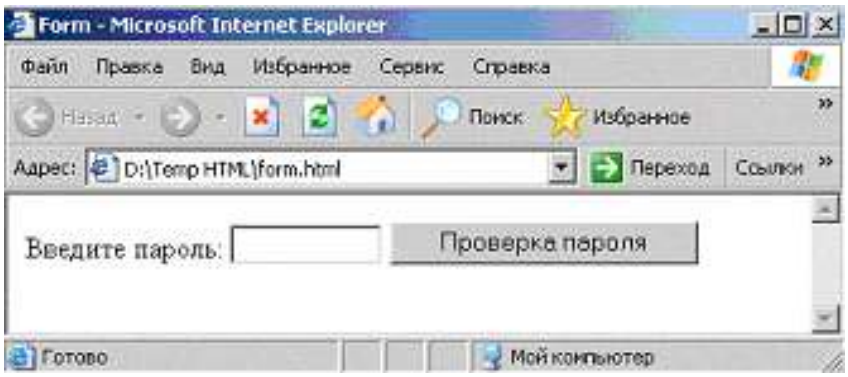


Рис 1.6.1 HTML-форма для ввода пароля

Далее расположен файл с PHP-программой. Все данные из нашей формы будут поступать в этот файл, который будет их обрабатывать.

Пример 1.6.2 Обработчик HTML-формы для ввода пароля доступа (файл primer6_2.php)

```
<?php
if ($_GET['pas'] == "17er0482")
{
print "Добро пожаловать!";
}
else
{
print "Незарегистрированный пользователь!";
}
?>
```

Текст обработчика формы должен находиться в файле с именем primer6_2.php, и этот файл будет запускаться сервером тогда, когда пользователь нажмет кнопку передачи на форме. В программе мы обращаемся к единственной переменной $_GET['pas'], значение которой и просим ввести пользователя. Именно в эту переменную попадает значение пароля, который вводит пользователь. В самой PHP-программе мы сравниваем это значение с правильным значением (17er0482) и в зависимости от результатов сравнения принимаем решение о статусе пользователя.

Обработка элементов с многозначным выбором

В предыдущем примере у нас одному элементу соответствовало одно значение. Для того, чтобы обрабатывать информацию от элементов, которые позволяют выбирать сразу несколько значений из списка, необходимо использовать массивы. Рассмотрим небольшой пример с использованием тэга select.

Пример 1.6.3 HTML-форма с тэгом select (primer6_3.html)

```
 <html>
<head>
<title>Form with select  </title>
</head>
<body>
<form action="primer6_4.php" method="GET">
Ваш НИК:  <input type="text" name="nick" size = "8" maxlength = "8">
<br>
Немного о себе:
<br>
<textarea name="info" rows = "5" cols = "30">
</textarea>
<br>
Увлечения:
<br>
<select name = "interests[]" multiple>
<option> Компьютеры
<option> Спорт
<option> Активный отдых
<option> Книги
</select>
<br>
<input type="submit" value="Послать анкету" name = "go">
```

```
</form>
</body>
</html>
```

Теперь в программе, предназначенной для обработки этой формы, значения, выбранные пользователем в элементе "interests[]", будут доступны как элементы массива $_GET['interests']. Вся полученная информация передается файлу primer6_3.php .На рисунке 1.6.2 вы увидите, как будет выглядеть наша форма.

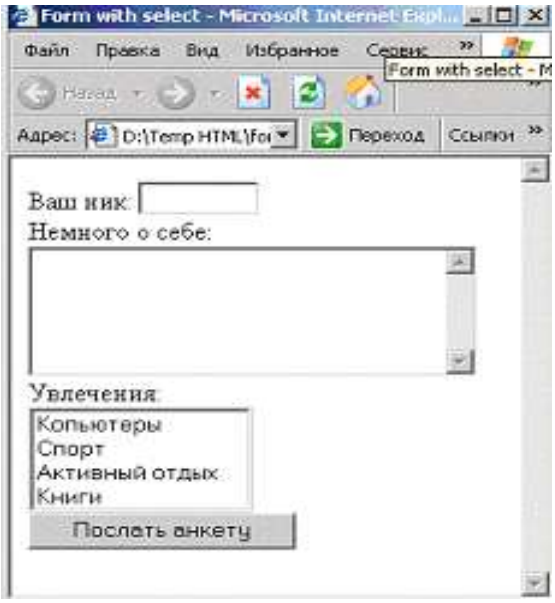


Рис 1.6.2 Форма-анкета

Пример 1.6.4 Обработка результатов анкетирования (файл primer6_4.php)

```
<?php
print "<h1>".$_GET['nick']."</h1><br>";
print "<b>Немного о себе<br>";
print $_GET['info']."<br>";
print "Мои увлечения</b><br>";
foreach($_GET['interests'] as $temp)
{
print "<i>".$temp."</i><br>";
}
?>
```

В нашей PHP-программе мы выводим на экран браузера значения переменных, полученных от формы ($_GET['nick'], $_GET['info'] и массив $_GET['interests').

Расположение HTML-текста и PHP-программы на одной странице

При некоторых обстоятельствах вам может понадобиться расположить PHP-программу, обрабатывающую данные на той же странице что и форму, передающую такие данные. Для того чтобы четко понимать зачем это делается, нужно знать различия между PHP и HTML.

PHP – это язык программирования, используемый на стороне сервера, конструкции которого вставляются HTML-текст. В отличие от обычного HTML-текста Web-страницы, программа PHP не передается браузеру, а обрабатывается интерпретатором PHP. Фрагменты HTML-текста остаются при этом без изменений, а операторы PHP выполняются и результат их обработки вставляется в HTML-текст, после чего они все вместе передаются браузеру клиента. Таким образом, для создания серьезного Web-проекта требуются два человека: дизайнер, обладающий тонким дизайнерским чутьем (работа заключается в размещении некоторого ресурса на Web-странице и в создании единого оформительского стиля), и программист, который пишет блоки программы, позволяющие оживить проект, сделать его интерактивным.

Основным недостатком HTML является то, что Web-страницы созданные с мощью HTML статичны, они мертвы и выводят всегда одну и туже информацию, хотя и в очень удобном виде. PHP позволяет убрать эту статичность.

Попробуем написать программу, в которой некоторая форма выводится несколько раз. Рассмотрим небольшой пример программы-игры, в которой игрок должен угадать слово.

Пример 1.6.5 Охота на слово (файл primer6_6.php)

```
<?php
$real_word = "bike";
if ($_GET['word'] = = "")
{print "Введите слово";}
elseif ($_GET['word'] = = $real_word)
{print "Вы победили";}
else
{print "Пробуйте еще"; }
?>
<html>
<head>
<title> Охота на слово</title>
</head>
<body>
<form method = "GET">
Я думаю, что загаданное слово: <input type="text" name="word">
<input type="submit" value="Ответ готов" name="go">
</form>
</body>
</html>
```

С начала у нас идет блок PHP команд, в котором мы через переменную $real_word задаем слово, которое необходимо угадать. У нас также инициализирована переменная $word, в ней хранится значение, которое вводит пользователь. Мы постоянно проверяем, идентичны ли значения этих двух переменных, в случае успеха поздравляем пользователя с победой, если нет, просим попробовать еще раз, а если пользователь забыл ввести слово, просим его выполнить эту несложную операцию. В HTML-блоке мы просто создаем форму в которой пользователь пробует угадать наше слово. Эта форма будет выводиться постоянно до прекращения работы программы, данные, вводимые в нее, будут при этом постоянно обновляться.

Обратите внимание на то, что у нас отсутствует атрибут ACTION дескриптора FORM. Это связано с тем, что у нас обработчик формы и сама форма располагаются в одном файле.

Перенаправление пользователя

Для простоты объяснения, под пользователем будем понимать человека, зашедшего на вашу Web-станицу, а под перенаправлением обыкновенную загрузку другого документа в браузере, без участия в этом процессе самого пользователя.

Программа из предыдущего примера имеет один существенный недостаток: форма полностью выводится на экран, независимо от того, угадал пользователь слово или нет. Из-за того, что страница в основном сделана в статичном варианте, избежать этого довольно трудно. Однако в случае успеха мы можем перенаправить пользователя на другую страницу, где он увидит текст поздравления.

Когда программа на сервере общается с клиентом, она должна сначала послать некоторые заголовки, содержащие информацию о том документе, который за ними следует. Вы можете послать свой собственный заголовок с помощью функции header();

Послав браузеру заголовок "Location", вы перенаправите пользователя на новую страницу.

Немного модифицируем пример 1.6.5.

Пример 1.6.6 Перенаправление пользователя с помощью заголовка "Location"

```
<?php
$real_word = "bike";
if ($_GET['word'] = = "")
{
print "Введите слово";
}
elseif ($_GET['word']= = $real_word)
{
header("Location: winner.html");
}
```

```
else
{
print "Пробуйте еще";
}
?>
<html>
<head>
<title> Охота на слово</title>
</head>
<body>
<form method = "GET">
Я думаю, что загаданное слово: <input type="text" name="word">
</form>
</body>
</html>
```

Изменения довольно легко заметить, в случае, когда пользователь угадывает слово, программа перенаправляет его на страницу winner.html (не забудьте, что данная страница должна существовать). Поскольку мы указали после имени заголовка (Location:) только имя файла, а путь к нему опустили, то этот файл должен находиться в том же каталоге на сервере, что и HTML–документы.

Задание для самопроверки

Напишите "программу-калькулятор", которая должна передать два числа, операцию, выполняемую над ними, и выдать результат на экран.

1.7.Работа с файлами

Включение файлов в документ

Хранение данных в переменных и массивах является временным; все эти данные теряются при завершении работы программы. Для постоянного хранения больших объемов данных используются файлы. Использование файлов очень удобно. Допустим, вы хотите узнать, подбирал ли кто-нибудь пароль к вашему ресурсу. Для этого достаточно создать файл, в который будут записываться все попытки неудачного ввода пароля пользователями, и проанализировать его (см. пример 1.7.8).

Для того, чтобы включить файл PHP-документ достаточно воспользоваться директивой include(). После этого текст PHP-программы во включаемом файле будет исполняться точно так же, как если бы он был записан во включающем файле непосредственно. Это удобно, когда вы хотите использовать одну и ту же программу в различных документах.

Функция include() требует одного аргумента – пути к файлу, который необходимо включить в документ.

Пример 1.7.1 Включение файлов в документ (файл primer7_1.php)

```
<?php
    include("primer7_2.php");
?>
```

Пример 7.2 Содержимое включаемого файла primer7_2.php

```
<?php
print "<h1> HELLO, WORLD!</h1>";
?>
```

Функция include() в примере 1.7.1 вставляет в документ содержание другого документа, который приведен в примере 1.7.2. В результате мы получаем вывод на экран строки Hello,World. Не забывайте, если файл находится не в том же каталоге на сервере, что и HTML-документы, необходимо указать к нему полный путь.

Исследование файлов

Проверка существования файла

Для того чтобы проверить существует ли нужный вам файл, применяется функция file_exists(). Эта функция принимает строку, содержащую полный или относительный путь к файлу. Если файл найден, то функция возвращает true, иначе – false.

Пример 1.7.3 Проверка существования файла (primer7_3.php)

```
<?php
if (file_exists("primer7_2.php"))
{
    print "primer7_2.php – это файл";
}
?>
```

Программа работает очень просто, если файл существует, мы сообщаем об это через функцию print пользователю.

Определение размера файла

Функция filesize("имя файла") определяет размер файла в байтах.

Пример 1.7.4 Определение размера файла (файл primer7_4.php)

```
<?php
```

```
print filesize("primer7_2.php");
?>
```

Результатом работы программы будет вывод на экран браузера размера запрашиваемого файла.

Создание и удаление файлов

В PHP можно легко создать файл с помощью функции touch(имя_файла). Получив строку с именем файла, эта функция создает пустой файл с заданным именем. Если же такой файл уже существует, то функция не меняет его содержания, но изменяет дату модификации.

Существующий файл можно удалить с помощью функции unlink(имя_файла), которая в качестве аргумента получает имя файла.

Пример 1.7.5 Создание (файл primer7_51.php)

```
<?php
touch("primer.txt");
print "Файл primer.txt создан";
?>
```

Удаление файла (primer7_52.php)

```
<?php
unlink("primer.txt");
print "Файл primer.txt удален";
?>
```

В этих двух примерах файл сначала создается, а затем удаляется. Обо всех своих действиях программа сообщает пользователю через функцию print().

После запуска каждой программы интересно пронаблюдать ситуацию в папке, содержащей эти скрипты. Файл primer.txt должен сначала появиться, а потом исчезнуть.

Иногда вам может понадобиться создать файл, поработать с ним, а затем удалить его. Такие файлы называют временными и их довольно часто используют в различных программах.

Открытие файла для чтения, записи или добавления

Для того чтобы с файлом можно было работать, его нужно открыть. Для этого существует функция fopen(). Данной функции передаются два аргумента – строка с именем файла и путем к нему, а также строка, описывающая

режим открытия файла. Самые простые режимы – это чтение, запись и добавление в конец файла.

Таблица 1.7.1.

Режимы открытия файла

Режим	Описание
r	Только чтение. Указатель текущей позиции устанавливается в начало файла
r+	Чтение и запись. Указатель текущей позиции устанавливается в начало файла
w	Только запись. Указатель текущей позиции устанавливается в начало файла, а все содержимое файла уничтожается. Если файл не существует, функция пытается создать его
w+	Чтение и запись. Указатель текущей позиции устанавливается в начало файла, а все содержимое файла уничтожается. Если файл не существует, функция пытается создать его
a	Только запись. Указатель текущей позиции устанавливается в конец файла. Если файл не существует, функция пытается создать его
a+	Чтение и запись. Указатель текущей позиции устанавливается в конец файла. Если файл не существует, функция пытается создать его

Строки, описывающие режим открытия файла, выглядят соответственно как "r", "w" и "a". Функция fopen() возвращает целое число, если файл удалось открыть, и значение false, если по каким-то причинам этого сделать не удалось. Целое число (указатель на файл), возвращаемое функцией, необходимо присвоить некоторой переменной, называемой манипулятором. Позднее, если вы захотите выполнить над файлом некоторые действия, вам будет достаточно указать только имя этого манипулятора, а не полный или относительный путь к файлу.

Таким образом, открытие файла для чтения будет выглядеть так:

```
$f=fopen("file.txt", "r");
```

Для записи в файл возможны два режима("a" и "w"). Если вы откроете файл в режиме записи ("w"), то все информация в файле будет уничтожена, и новые данные запишутся в начало файла. При этом если файл отсутствовал, то он создается. Если вы откроете файл в режиме добавления ("a"), то все новые данные будут добавлены в конец файла.

После завершения работы файл всегда следует закрывать функцией fclose().

Функция fclose($f) закрывает файл с заданным манипулятором $f. При успешном закрытии возвращается TRUE, при неудаче — FALSE.

Пример 1.7.6. Работа с указателем, возвращаемым функцией fopen()
(файл primer7_6.php)

```
<?php
```

```
$p = fopen("primer.txt", "r");
$fs=filesize("primer.txt");
$text=fread($p, $fs);
print " размер файла=".$fs." байт <br>";
print "его содержимое - ".$text;
fclose($p);
?>
```

Для проверки работы этой программы сначала создайте в рабочем каталоге текстовый файл primer.txt и запишите туда произвольный текст.

В данной программе мы открываем его для чтения с помощью функции fopen(). Получаем от этой функции указатель на файл и сохраняем его в переменной $p. Для вывода на экран браузера размеров файла мы пользуемся уже не его именем, а полученным от функции fopen() указателем $p. Здесь мы записали его в переменную $fs. А затем прочитываем его содержимое во вспомогательную переменную $text. В функции чтения нужно указать, сколько символов будет прочитано. У нас это определило функция filesize(), но его можно указать и с запасом.

Если все прошло гладко и были выполнены все необходимые действия с файлом, то по окончании работы его следует закрыть. Для этого существует функция fclose(), которой нужно передать указатель на файл.

Построчное чтение из файла с помощью fgets()

Открыв файл для чтения, вам может понадобиться прочитать из него несколько строк. Для этого достаточно воспользоваться функцией fgets(). Данная функция имеет два аргумента – указатель на открытый файл, из которого мы собираемся читать, и максимальное количество символов, которое можно прочесть из файла. Функция fgets() будет читать данные из файла до тех пор, пока не дойдет до конца строки (не встретит символ конца строки) или конца файла.

Пример 1.7.7 Построчное чтение из файла (файл primer7_7.php)

```
<?php
$p = fopen("guess.txt", "r") or die ("Could'n open file!");
while(!feof($p) )
{
$line = fgets($p,10);
print $line."<br>";
}
fclose($p);?>
```

Рассмотрим это пример детально. В первой строчке файл открывается для чтения guess.php (просто он был создан ранее и там содержится достаточное количество строк для чтения), полученный указатель сохраняется в переменной $p. Однако в этой же строчке используется ранее незнакомый момент – or die("Could'n open file!"). Функция die() просто выводит свой аргумент на экран браузера и прекращает выполнение программы.

Таким образом, всю сточку целиком можно понять так, если открыть файл не удастся (если функция fopen() вернет не указатель на файл, а значение false), на экран будет выведена строка Could'n open file! и работа программы закончится. Если все пройдет гладко и файл будет открыт без проблем, функция die() будет просто проигнорирована.

Во второй строчке вы можете наблюдать цикл while. Цикл будет работать до тех пор, пока не будет достигнут конец файла.

Для определения конца файла используется функция feof(). Данная функция возвращает true при достижении конца файла и false – в противном случае. Напомним, что символ ! означает инверсию. В качестве аргумента мы передаем этой функции переменную $p, содержащую указатель на наш файл.

Таким образом, всю строчку можно понять следующим образом: до тех пор пока функция feof() возвращает значение ложь (иначе говоря, до тех пор пока мы не дошли до конца файла), цикл while будет выполняться.

В теле цикла выполняются следующие действия: мы читаем с помощью функции fgets() строчку или 10 байт, если конец строки не был достигнут, мы присваиваем прочитанную строку переменной $line и выводим ее на экран, добавляя тэг
 для удобства чтения. Ограничение длины строки (у нас 10 символов) можно не указывать.

Еще раз напомним, перечисленные в теле цикла действия выполняются до тех пор, пока не будет достигнут конец файла.

Запись в файл с помощью функции fwrite()

У функции fwrite() всего два аргумента – указатель на файл и строка. Функция просто записывает эту строку в файл.

Для иллюстрации работы функции fwrite() немного модифицируем пример 6.1 из предыдущей темы. В этом примере мы анализировали правильность введенного пароля. Теперь мы будем записывать время всех неудачных попыток в текстовый файл (new.doc).

Пример 1.7.8 HTML-форма для ввода пароля доступа (form2.html)

```
 <form action="primer7_9.php" method="GET">
Введите пароль: <input Type="password" Name = "pas" Size="10"
Maxlength="10">
<input type="submit" value="Проверка пароля">
</form>
```

Никаких изменений в самой форме ввода, кроме указания файла - обработчика в атрибуте action, сделано не будет, они не нужны.

Пример 1.7.9 Обработчик HTML-формы для ввода пароля доступа (файл primer7_9.php)

```
<?php
if($_GET['pas']=="17er0482")
{
$string="  Password OK ---->>>\r\n";
$fp = fopen("new.doc", "a") ;
fwrite( $fp, $string);
fclose( $fp );
}
else
{
$d= date("j F Y H.i ");
$string=$d."  Password error ---->>>".$_GET['pas']."\r\n";
$fp = fopen("new.doc", "a") ;
fwrite( $fp, $string);
fclose( $fp );
header("Location: form2.html");
}
?>
```

Программа проверяет правильность введенного пароля. Если пароль правильный, файл new.doc открывается в режиме добавления и в него с помощью функции fwrite() записывается строка Password OK. Служебные символы "\r\n" создают перевод строки в текстовом документе.

Если пароль неправильный, в тот же файл записывается текущая дата и время (подробнее о них в следующей теме), строка Password error ---->>> и само значение введенного пароля (\r это символ возврата курсора, а \n – это символ новой строки). С помощью функции header() с параметром Location пользователю отправляется исходная форма для ввода, чтобы он попробовал ввести пароль еще раз.

Задание для самопроверки

Создайте форму для сборки анкетных данных студента. Все данные, полученные из формы должны сохраняться в некотором файле.

1.8.Работа с датой и временем

Практически не один серьезный проект не обходится без работы с датой и временем. Встроенная в PHP функция date() предоставляет вам всю необходимую информацию о текущей дате и времени. Мы рассмотрим вызов

данной функции в упрощенном варианте, когда функции date() передается только один параметр, строка, позволяющая функции понять, как именно следует выводить текущую дату/время.

Пример 1.8.1 Использование функции date() для получения текущей даты и текущего времени.

```
<?php
print date("j F Y H:i");
?>
```

Результатом работы данной программы будет вывод даты в следующем представлении: 13 April 2004 20:25. Вид (формат), в котором представлена дата, полностью зависит от переданной в качестве аргумента строчки "j F Y H.i". Разберем эту строку.

j, F, Y, H, i – это специальные символы, получая которые функция date() понимает, в каком формате необходимо представить дату.

j – номер дня в месяце без предваряющего нуля (в нашем случае происходит вывод числа 13).

F – полное английское название месяца (April).

Y – год, представленный с помощью 4 цифр(2004).

H – час, представленный в 24-часовом формате(20), i – минуты, от «00» до «59»(25).

Для более красивого вывода между H и i поставлено двоеточие (поэтому и получилось 20:25). Функция date() универсальна, она выводит только ту информацию, которая нас интересует. Допустим, нас интересует только сегодняшнее число, функция легко предоставит вам эту информацию, если вы просто передадите ей в качестве аргумента строку с символом j: print date("j");

Ниже представлена таблица символов, которые позволяют изменять формат даты.

Таблица 1.8.1

Символы, позволяющие изменять формат даты

Символ	Описание
U	Количество секунд, прошедших с полночи 1 января 1970 года (с момента основания UNIX)
Y	Год(4 цифры)
y	Год(2 цифры)
z	Номер дня от начала года(от 0 до 365)
F	Полное английское название месяца(April)
m	Номер месяца(две цифры: от 01 до 12)
n	Номер месяца без предваряющего нуля: от 1 до12
M	Трехсимвольная английская аббревиатура месяца(Apr)
d	Номер дня в месяце(2 цифры: от 01 до 31)
j	Номер дня в месяце без предваряющего нуля
l	День недели по-английски (например, Tuesday)

D	Трехбуквенная английская аббревиатура дня недели(Tue)
A	До или после полудня: «PM» или «AM»
H	Часы от «00» до «23»
h	Часы от «00» до «12»
i	Минуты от «00» до «59»
s	Секунды от «00» до «59»

Задание для самопроверки

Напишите форму с одной единственной кнопкой, при нажатии на которую мы бы узнавали текущий месяц, день недели и год.

1.9.Форматирование данных при выводе на экран

Работа с функцией printf()

Функции printf() передается в качестве аргумента строка, которая называется строкой управления форматом, и аргументы, которые выводятся в соответствии с этими форматами. В строке управления (она записывается в кавычках) располагается поясняющий текст, а в местах, где должны размещаться выводимые данные вместо них указывается формат, в котором они должны выводиться. Сами же данные располагаются далее как дополнительные аргументы и разделяются запятыми.

Рассмотрим пример, который выводит два числа с помощью функции printf():

Пример 1.9.1 Вывод двух чисел в десятичном представлении

```
<?php
printf("Первое число: %d<br>Второе число: %b", 5555,8934);
?>
```

Результат работы программы будет следующим:
Первое число:5555
Второе число:10001011100110

В строку управления форматом вставлен специальный код, называющийся определителем преобразования,. Определитель преобразования начинается с символа процента и определят то, как функция printf() будет обрабатывать соответствующий дополнительный аргумент. В строку управления форматом можно включить любое количество определителей преобразования. При этом нужно передать функции printf() такое же количество аргументов.

В нашем примере используются два определителя преобразования. Первый из определителей соответствует первому дополнительному аргументу функции, числу 5555, а второй относится к аргументу 8934. Символ d, сле-

дующий за знаком процента означает, что первый аргумент надо представить в виде десятичного числа, а символ b, сигнализирует о том, что число 8964 необходимо представить в двоичной форме. Символы, следующие за определителем преобразования (после знака процента), принято называть определителями типа.

Таблица 1.9.1

Наиболее распространенные определители типа

Определитель	Описание
d	Выводит аргумент как десятичное число
b	Выводит аргумент как двоичное число
f	Выводит аргумент как число с плавающей запятой
o	Выводит аргумент как восьмеричное число
X	Выводит аргумент как шестнадцатеричное число в верхнем регистре
s	Выводит аргумент как строку
nf	Выводит число с n знаками после запятой

Задание точности

Порой приходится округлять полученные данные при выводе их на экран. В частности, это очень удобно при работе с валютами. Определитель точности должен находиться прямо перед определителем типа. Определитель точности состоит из точки, за которой следует количество знаков после запятой, которое необходимо отображать. Этот определитель действует только на данные, выводящиеся с определителем типа f.

Пример 1.9.2 Задание точности

```
<?php
$a = 5/3;
printf("%.2f",$a);
?>
```

Результат: 1.67. Так как мы попросили вывести только два знака после запятой, это и произошло.

Задание для самопроверки

Создайте форму для перевода десятичного числа в двоичное.

1.10.Работа со строками

World Wide Web является в первую очередь текстовой средой. И независимо от того, насколько WWW богата остальными компонентами, за всем этим находится HTML. Поэтому не случайно то, что в PHP4 существует большое количество функций, с помощью которых можно форматировать строки, а также исследовать и манипулировать ими.

Выделение необходимого элемента строки

Допустим, нам необходимо выделить из строки какой-нибудь отдельный элемент. Для этого достаточно указать его порядковый номер в строке. Нумерация элементов в строке подчиняется правилам нумерации элементов массива. Первый элемент строки имеет нулевой порядковый номер, второй - первый и т.д.

Пример 1.10.1 Выделение необходимого элемента строки

```
<?php
$str = "vegas";
print $str[0];
?>
```

Результат работы данной программы вывод первого элемента строки символа v. Строку можно представить как массив символов. Таким образом, вы можете обращаться к отдельному элементу строки как к элементу массива.

Определение длины строки

Для определения длины строки используется функция strlen(). В качестве аргумента она принимает строку, длину которой необходимо выяснить. Функция передает нам целочисленное значение, равное количеству символов в переданной строке.

Пример 1.10.2. Определение длины строки

```
<?php
$str = "vegas";
print strlen($str);
?>
```

Результатом работы данной программы является вывод числа 5. Очень часто данная функция используется при проверке длины пароля на различных формах регистрации.

Нахождение подстроки в строке

Может сложиться такая ситуация, что вам надо знать присутствует ли некоторая подстрока (часть строки) в строке. Это можно сделать с помощью функции strstr($st1, $st2), которая выделяет из строки подстроку, если она встречается.

Функции strstr() передаются два аргумента: исходная строка $st1 и искомая подстрока $st2. Функция возвращает false, если такой подстроки в строке

нет. В противном случае функция возвращает часть исходной строки ($st1), которая начинается с искомой подстроки ($st2).

Например, strstr("Вася","а") вернет значение "ася", а strstr("Вася","Боб") вернет false.

Рассмотрим пример, использования этой функции, когда вы запрещаете использование некоторых символов при вводе пользовательского пароля при регистрации.

Пример 1.10.3 Запрещение ввода в регистрационную форму символов @ . , !

```
<?php
if ($_GET['pas'] != "")
 {
 if ((strstr($_GET['pas'],"@")) or (strstr($_GET['pas'],".")) or
(strstr($_GET['pas'],",")) or (strstr($_GET['pas'],"!")))
   {
   print "Вы ввели недопустимый символ!<br>";
   }
 else
   print "Все в порядке!<br>";
  }
else
{
print "Введите пароль!<br>";
}
?>

<html>
<head>
<title>
        Проверка ввода
</title>
</head>
<body>
<form method="GET">
Введите пароль: <input type="password" name="pas">
<input type="submit"  value="go">
</form>
</body>
</html>
```

Сначала в программе создается элементарная форма с единственной кнопкой. Форма будет отправлять данные PHP-программе, которая находится в том же файле, что и сама форма. От формы PHP-программа должна получить пароль, введенный пользователем, а если выразиться точнее, значение переменной $_GET['pas']. В PHP-программе сначала проверяется, введен ли пароль ($_GET['pas']!= ""), если нет, выводится сообщение (Введите пароль). Если значение переменной $_GET['pas'] не является пустой строкой (пароль введен), то с помощью инструкции if и функции strstr() проверяется, не ввел ли пользователь запрещенные символы (@ . , !).

Расщепление строки

Другой пример, когда нужно разбить строку, содержащую перечисление каких-то объектов на отдельные элементы и организовать из них массив. Это делает функция explode($st1,$st2). Первый параметр $st1 – строка (обычно символ), которая является разделителем, а $st2– строка подлежащая расщеплению. Рассмотрим пример, в котором в переменой $price записаны цены товаров, разделенные символом вертикальная черта (|).

Пример 1.10.4. Разбиение строки на элементы массива

```
<?
$price="100|150|300|580|220";
$priceArr=explode("|",$price);
print count($priceArr);// выведет 5
print "<hr>";
print $priceArr[0];// выведет 100
print $priceArr[1];// выведет 150
print $priceArr[4];// выведет 220
?>
```

Здесь в результате работы функции explode формируется массив $priceArr, состоящий из 5 элементов. Напоминаем, что их подсчитывает функция count(), а нумерация элементов массива начинается от нуля, поэтому последний элемент будет иметь индекс четыре. Если разделитель $st1 не содержится в исследуемой строке, то функция explode вернет массив, состоящий только из одного элемента – это исходная строка $st2. Если же разделитель – пустая строка "", то функция explode вернет значение FALSE.

Заметим, что данная функция будет использована в примере работы с корзиной товаров интернет-магазина.

Изменение регистра

В PHP есть несколько функций позволяющих изменить регистр строки, это может пригодиться для использования в косметических целях. Для преобразования строки в верхний регистр используется функция strtoupper(). Этой функции передается исходная строка, а результатом ее работы является строка, у которой все символы являются символами верхнего регистра.

Пример 1.10.5. Использование функции strtoupper()

```
<?php
$text ="hello";
$new_text =strtoupper($text);
print $new_text;
?>
```

Результат работы программы – HELLO. Для преобразования строки в символы нижнего регистра используется функция strtolower(), этой функции также передается только строка.

Но наиболее полезной функцией для изменения регистра без сомнения можно назвать функцию ucwords(). Эта функция преобразовывает первую букву каждого слова передаваемой строки в соответствующую букву верхнего регистра. В качестве аргумента принимает строку, в которой необходимо провести преобразование.

Пример 1.10.6 Использование функции ucwords()

```
<?php
$name = "voronin vadim";
$new_name = ucwords($name);
print $new_name;
?>
```

Программа выведет на экран браузера строку (Voronin Vadim); Данная функция используется чаще всего при регистрации пользователей.

Задание для самопроверки

Модифицируйте пример 1.10.3. Выводите на печать не только предупреждающее сообщение, но и сам недопустимый символ, который был введен.

1.11.Связь с базами данных на примере сервера MySQL

Одной из самых приятных возможностей языка PHP является легкость, с которой программист на нем может обращаться с базами данных. База данных – это совокупность связанных двухмерных таблиц, содержащих определенную информацию. Программное обеспечение, которое управляет базой данных, называется системой управления базой данных (СУБД). В данном уроке мы будем говорить о связи PHP и СУБД сервера MySQL.

База данных в MySQL, образно говоря, – это обыкновенная папка (каталог). Таблица физически представляется файлами с расширениями frm, myd, myi. Логически таблица является совокупностью записей. Запись – это набор полей, содержащих связную информацию. Имя базы данных должно быть

уникально в пределах системы, имя таблицы – в пределах базы данных, а имена полей – в пределах таблицы.

Данная тема довольно важна и перед ее изучением поговорим о взаимодействии между браузером, Web-сервером, вашей программой, написанной на PHP, и сервером баз данных MySQL. Необходимо четко понимать, что принято называть сервером, а что клиентом. Для этого нужно разбираться в архитектуре "клиент-сервер".

Архитектура «Клиент-сервер»

Сервер – это программа, представляющая какие-то услуги другим программам. Примеры серверов: Web-сервер Apache, серверы баз данных – MySQL, ORACLE, сетевые файловые системы и др.

Клиент – это программа, использующая услугу, представляемую программой сервера. Примеры клиентов: браузер (например, Internet Explorer), почтовый клиент (например, Outlook), клиент ICQ. Часто люди клиентом или сервером просто называют компьютер, на котором работает какая-то из этих программ. В сущности, клиент и сервер – это роли, исполняемые программами. Клиенты и серверы физически могут находиться на одном компьютере. Одна и та же программа может быть и клиентом, и сервером одновременно, это только роли. Если привести пример из реальной жизни, то столовая или кафе являются серверами, они предоставляют услуги клиентам, населению, но в то же время столовая или кафе могут быть клиентами других структур или организаций, например, снабженческих.

Сервер и клиент в сети между собой «разговаривает» на «языке» (в широком смысле слов), понятном обеим сторонам. Этот «язык» называют протоколом.

Приведем примеры некоторых протоколов:

FTP (File Transfer Protocol)

HTTP (Hyper Text Transfer Protocol)

SMTP (Simple Mail Transfer Protocol)

NNTP (Network News Transfer Protocol)

На «языке» протокола клиент отправляет серверу команду, которую тот понимает, выполняет и отправляет клиенту результат выполнения.

В нашем случае происходит взаимодействие между браузером, сервером Apache и сервером баз данных MySQL.

Как же происходит общение PHP программы с базой данных MySQL?

С таблицами баз данных можно выполнять традиционные действия прямо из программы PHP. Сначала к базе данных нужно подключиться, потом можно создавать, изменять и удалять таблицы, наполнять их данными, изменить и удалять данные, а так же осуществлять поиск по таблицам.

Подключение к серверу базы данных

Чтобы начать работу со своей базой данных, необходимо сначала подключиться к серверу баз данных. Для этого можно воспользоваться функцией

mysql_connect(). Данная функция имеет три аргумента: имя компьютера, имя пользователя и его пароль. Функция возвращает нам некоторое целое число (указатель на соединение), вся дальнейшая работа будет осуществляться через этот указатель.

Пример 1.11.1 Подключение к серверу базы данных (файл primer11_1.php)

```
<?php
$p = mysql_connect("имя хоста", "имя пользователя","пароль") or die("Не могу подключиться к серверу базы данных");
?>
```

При выполнение данной работы имя хоста, имя пользователя, пароль, а так же имя базы данных нужно узнать у преподавателя. Обычно этим занимается хостинг-компания.

В данном примере использована знакомая вам функция die(). Если нам не удастся подключиться, mysql_connect() вернет не целое число, а false, тогда функция die() закончит работу программы и выведет предупреждающее сообщение. Если программа из этого примера работает правильно, вы должны увидеть пустую страницу.

Выбор базы данных

После того, как соединение с сервером было установлено, вам нужно выбрать базу данных, с которой вы хотите начать работу. Для этого необходимо воспользоваться функцией mysql_select_db(), этой функции необходимо передать два параметра: имя базы данных и указатель на соединение. Функция возвращает значение истина, если база данных существует и доступ к ней возможен, и значение ложь, если сделать этого нельзя.

Пример 1.11.2 Открытие подключения и выбор (файл primer11_2.php)

```
<?php
print "Соединяюсь…";
$p = mysql_connect("имя хоста", "имя пользователя","пароль")  or die("Не могу подключиться к серверу базы данных");
print "Успешно";
print "Открываю базу данных new…";
mysql_select_db("имя базы данных") or die("No base!!! ");
print "База открыта";
?>
```

Функция mysql_select_db("имя базы данных") открывает уже существующую базу new. Если вы попытаетесь открыть несуществующую базу, программа, из данного примера, выведет строчку No base!!! и закончит свою работу.

Добавление данных в таблицу

Теперь самое время создавать таблицы нашей базы данных. Создадим в базе таблицу с именем telephones. В этой таблице будем хранить фамилии, адреса электронной почты и телефоны своих знакомых.

При создании таблицы необходимо определить наименования полей, их тип и некоторые характеристики. Сделаем небольшое отступление и перечислим основные типы данных.

Типы данных

Типы целочисленных данных	
tinyint	от -128 до 127
smallint	от -32 768 до 32767
mediumint	от -8 388 608 до 8 388 607
int	от -2 127 483 648 до 2 127 483 647
bigint	от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807
Типы вещественных чисел	
float	число с плавающей точкой небольшой точности
double	число с плавающей точкой двойной точности
Строковые типы данных	
tinytext	до 255 символов
text	до 65 535 символов
mediumtext	до 16 777 215 символов
longtext	до 4 294 967 295 символов

Другие типы данных.

Существуют так же типы данных, предназначенные для хранения даты и времени, изображений и др. Для хранения четко заданного количество символов используется специальный тип данных VARCHAR, где в скобках указывается это количество, например, VARCHAR(20).

Кроме определения типа данных при создании таблицы можно указать дополнительные инструкции – модификаторы и ключи. Они определяют "поведение" данных и те операции, которые можно или запрещено выполнять в соответствующих столбцах таблицы. Например,

NOT NULL – обязательно непустое поле,

PRIMARY KEY(имя_поля) – ключевое поле,

AUTO_INCREMENT – значение поля автоматически увеличивается на 1 (счетчик), используется только для ключевых и целочисленных полей,

INDEX или KEY – индексированные поля (первичный ключ автоматически индексирован),

UNIQUE – указывает на не повторяемость значений данных в поле.

Итак, вернемся к нашей таблице. Она будет содержать 4 поля следующих типов:

Имя поля	Тип поля и уточняющие характеристики – модификаторы и ключи	Назначение поля
id	INT AUTO_INCREMENT PRIMARY KEY	Счетчик записей, значения которого автоматически увеличиваются на 1, первичный ключ.
surname	VARCHAR(20)	Поле, которое будет содержать фамилии. Предположим, что длина фамилии не будет превышать 20 символов
email	VARCHAR(20)	Поле для записи телефонов
tel	VARCHAR(20)	Поле для адресов электронной почты

Таблицу telephones создадим с помощью инструкции SQL:

```
create table telephones(id INT AUTO_INCREMENT PRIMARY KEY,
surname VARCHAR(20),
email VARCHAR(20),
tel VARCHAR(20));
```

Осталось только выполнить этот SQL запрос. Для этого используется функция mysql_querry(). Этой функции необходимо передать два параметра: строку с запросом и указатель на соединение. Функция возвращает положительное число (указатель запроса), в случае успешного выполнения запроса, и false в случае неудачи. Рассмотрим пример, в котором создаю описанную выше таблицу.

Пример 1.11.3 Создание таблицы telephones (файл primer11_3.php)

```
<?php
$p = mysql_connect("имя хоста", "имя пользователя","пароль") or die("Не могу подключиться к серверу базы данных");
mysql_select_db("имя базы данных", $p) or die("Не могу открыть");
$query = "create table telephones(id INT AUTO_INCREMENT PRIMARY KEY, surname VARCHAR(20), email VARCHAR(20), tel VARCHAR (20))";
mysql_query($query, $p) or die("Не удалось выполнить запрос");
print "Таблица создана";
?>
```

Сформированный SQL запрос в виде строки записывается в переменную $query. Далее передаем эту строку, как первый аргумент функции mysql_query(). Вторым параметром идет указатель на соединение, полученный от функции mysql_connect(). Таблица создана.

Примечание. Нетрудно заметить, что запрос CREATE TABLE можно поместить непосредственно в функцию mysql_query в качестве первого па-

раметра. Мы же занесли его в отдельную переменную $query для упрощения отладки первых программ.

Теперь заполним таблицу данными. В этом нет ничего сложного, достаточно корректно составить SQL запрос.

Пример 1.11.4 Добавление данных в таблицу (файл primer11_4.php)

```
<?php
$p = mysql_connect("имя хоста", "имя пользователя","пароль");
mysql_select_db("имя базы данных",$p) or die ("Не могу открыть");
$query = "INSERT INTO telephones(surname ,email ,tel)
values('Kolbin', 'user2000@mail.ru', '580-46-82')" ;
mysql_query($query, $p) or die("Не удалось выполнить запрос");
print "Данные добавлены";
?>
```

Обратите внимание на то, что мы не указываем значение поля id. Это поле изменяется автоматически. В нашем примере мы добавили запись, содержащую информацию о человеке по фамилии Колбин. При каждом выполнении программы из примера 11.4 в таблицу будет добавляться новая запись, содержащая одни и те же данные, но с новым идентификатором id (он автоматически увеличивается на 1).

Заметим, что в предыдущем примере использовались как одинарные, так и двойные кавычки. Используйте их на свое усмотрение, только если Вы открыли двойную кавычку, то и закрыть Вы должны двойной. Внутри одних кавычек нужно использовать другие.

В примере 1.11.5 приведена программа, которая добавляет в таблицу данные, введенные пользователем.

Пример 1.11.5 Форма для ввода информации (файл add_user.html)

```
<html>
<body>
<form action="add.php" method="POST">
Surname     <input type="text" name="surname">
Email:       <input type="text" name="email">
Tel:          <input type="text" name="tel">
<input type="submit" value="ADD">
</form>
<body>
<html>
```

В этом файле все довольно просто: создаем три текстовых поля для ввода информации и кнопку для отсылки этой информации файлу add.php, который будет ее заносить в базу данных.

Пример 1.11.6 Обработчик формы (файл add.php)

```
<?php
function add_data($surname, $email, $tel)
{
$p=mysql_connect("имя хоста", "имя пользователя","пароль")  or die("Не могу подключиться к серверу базы данных");
mysql_select_db("имя базы данныз",$p) or die("NO BASE!");
$query = "INSERT INTO telephones(surname , email , tel)
values('$surname','$email','$tel')";
mysql_query($query,$p) or die("Could't execute");
}
if (($_POST['surname'] !="") and ($_POST['email'] !="") and ($_POST['tel'] !="""))
{
add_data($_POST['surname'], $_POST['email'], $_POST['tel']);
print "Added data: Surname: $_POST['surname'] Email:  $_POST['email']  TEl: $_POST['tel']  <br>";
}
    else
{
print "Check data<br>";
}
?>
<html>
<body>
<a href="add.html">Add another</a>
</body>
</html>
```

Функция add_data() использована для добавления данных в базу. Внутри функции сначала мы подключаемся к нашей базе new. Если вы помните, в этой базе уже есть одна таблица (telephones). Функции add_data() необходимо передать в качестве аргументов три параметра: фамилию ($surname), электронный адрес ($email) и телефон ($tel). Эти параметры PHP программа получит от нашей формы сразу после того, как будет нажата кнопка add.

Сразу после создания функции идет инструкция if, в задачу которой входит следующее. Если переменная $surname и (and) переменная $email и переменная $tel не являются пустыми (!="") строками (иначе говоря, если вы заполнили все три поля для ввода информации), передать их функции add_data() в качестве аргументов и вывести информационное сообщение, повествующее о том, какие данные добавлены в базу. Если хотя бы одна из

этих переменных содержит пустую строку (например, кто-то забыл ввести фамилию или телефон), выводится предупреждающая строчка, данные в базу не добавляются.

В конце программы имеется HTML-блок, который позволяет вернуться назад на страницу add_user.html с помощью ссылки. Эта ссылка необходима для случая, когда кто-то захочет ввести еще данные.

Доступ к информации, содержащейся в базе данных

Пора научиться читать информацию из нашей базы данных. Для этого необходимо сделать SQL-запрос типа SELECT. В качестве примера напишем небольшую программу, в которой выводятся все записи из базы данных. Для этого в запросе SELECT стоит символ *

Пример 1.11.7. Вывод всех записей из Базы данных (файл primer11_7.php)

```
<?php
$p = mysql_connect("имя хоста", "имя пользователя","пароль");
mysql_select_db("имя базы данных") or die ("NO BASE!");
$query = "SELECT *  FROM telephones" ;
$result = mysql_query($query) or die ("Can't execute");

while($mas_info = mysql_fetch_row($result))
{
foreach($mas_info as $temp)
    {
     print "$temp – ";
     }
print "<br>";

}
```

После выполнения запроса в переменную $result заносится весь массив данных из базы. Теперь из нужно "разобрать" по записям и вывести все поля каждой записи. Один их приемов – использовать вложенные циклы. Внешний цикл while , управляет записями. Каждая запись записывается в ассоциированный массив $mas_info. А внутренний цикл foreach выводит каждое поле этого массива, разделяя их символом (–). Чтобы разделить записи на экране внешний цикл содержит оператор перевода строки print "
";

В условном выражении цикла while мы присваиваем переменной $mas_info массив значений, получаемый от функции mysql_fetch_row(). Цикл будет выполняться до тех пор, пока не закончатся записи. Переменной $mas_info ничего не будет присвоено, условие примет значение false и цикл закончит работу.

А теперь рассмотрим пример имитирующий работу поисковой системы. Попробуем искать в базе записи, удовлетворяющие запрошенной фамилии.
Пример 1.11.8 Поиск записи по фамилии (файл primer11_8.php)

```
<?php
$p = mysql_connect("имя хоста", "имя пользователя","пароль");
mysql_select_db("имя базы данных") or die ("NO BASE!");
$s=$_POST['surname'];
$query = "SELECT surname , tel , email
    FROM telephones
    WHERE surname ='$s' " ;

$result = mysql_query($query) or die ("Can't execute");

while($mas_info = mysql_fetch_row($result))
{
foreach($mas_info as $temp)
     {
     print "$temp – ";
     }
print "<br>";

}
?>

<html>
<head>
<title>     Find tel by name </title>
</head>
<body>
    <form method ="POST">
Surname:  <input type = "text" name = "surname">
              <input type = "submit" value = "FIND">
     </form>
</body>
</html>
```

Сначала мы подключаемся к нашей базе new. Затем формируем текст SQL-запроса, который должен помочь нам по фамилии получить другую информацию. В тексте запроса фигурирует значение переменной $surname, зна-

чение которой будем получать от пользователя через форму ввода. Сам текст мы сохраняем в переменной $query, которую передаем как аргумент функции mysql_query(). Данная функция возвращает нам положительное число, если текст запроса написан без ошибок, которое мы будем называть указатель запроса. Это число поможет нам с помощью различных функций получить информацию относительно запроса, на который оно указывает.

"Разбор" записей выполняется так же, как и в предыдущем примере.

В HTML-блоке программы мы формируем форму для поиска записей по фамилии. Данная форма имеет единственное поле, предназначенное для ввода, интересующей нас фамилии, и кнопка отправки запроса (submit).

Задание для самопроверки

Создайте базу данных для хранения анкетных данных сотрудников какого-нибудь предприятия.

Напишите одну программу, которая позволит пользователям заполнять эту базу данных, и вторую, которая будет выводить данные о всех сотрудниках. Предусмотрите возможность поиска сотрудников по району проживания.

ЧАсть 2. СИСТЕМЫ УПРАВЛЕНИЯ КОНТЕНТОМ

2.1. Понятие контента и способы управления контентом

Важнейшими составляющими бизнеса являются веб-сайты, а инструментальные средства для создания и развертывания веб-сайтов становятся все более гибкими и простыми в использовании.

На ранних этапах развития Интернета, разработка сайта сводилась к созданию файловой структуры из html-страничек и размещению в них помимо непосредственно данных различных дополнительных элементов, таких как меню навигации или ссылки, присущие без исключения всем страницам, но которые необходимо было вносить в каждую из них вручную. Такие сайты являлись статическими.

Бурный рост популярности электронного бизнеса привел к экспоненциальному увеличению как объемов информации, так и числа посетителей сайтов. Увеличились трудозатраты на поддержание сайта на сервере. Создатель сайта был вынужден тратить время не только на непосредственное размещение статьи или публикации, но и на внесение сопутствующей информации, например, ссылок на эту статьи, создания меню навигации и элементов, постоянно присутствующих в определенных позициях на сайте (например, сведений об авторе и др.).

Выходом из данной ситуации стало создание нового класса программ, которые выполняли рутинные операции, не связанные с непосредственным созданием содержимого. Такие системы называют CMS «Content Management System». На русский язык данная аббревиатура переводится как «Система управления содержимым», либо «Система управления контентом», часто для простоты их называют «движком сайта».

Контентом называется всё информационное наполнение интернет сайта, газеты, журнала. В переводе с английского языка слово контент, означает «содержание». К нему относятся текст, фотографии, картинки, видео и аудиофайлы - все, что пользователь видит при просмотре страницы. Так же контентом принято называть ту часть информационной составляющей сетевого ресурса (или сайта), которую пользователь может использовать по своему усмотрению, в частности загрузить на собственный компьютер и сохранить для личного использования.

Системы управления контентом позволили отделить дизайн сайта от его содержания путем создания шаблонов страниц, которые формировались "на лету" добавляя содержимое из баз данных или файловых хранилищ.

Таким образом, Системой управления контентом принято называть информационную систему или компьютерную программу, используемую для обеспечения и организации совместного процесса создания, редактирования и управления контентом (то есть содержимым) электронного предприятия.

Главной целью такой системы является возможность собирать в единое целое все разнотипные источники знаний и информации, доступные как внутри организации, так и за ее пределами.

Другая задача — обеспечить взаимодействие сотрудников, рабочих групп и проектов с созданными ими базами знаний, информацией и данными так, чтобы их легко можно было найти, извлечь и повторно использовать привычным для пользователя образом.

Таким образом, системы управления контентом должны обеспечивать:

-легкий ввод содержательной информации,

-контроль автоматизированных бизнес-процессов,

-пользовательские услуги, например, возврат в предыдущее состояние,

-динамическое направление затребованных данных целевым группам пользователей.

2.2. Классификация систем управления контентом

В общем случае системы управления содержимым делятся на:

•систему управления содержанием масштаба предприятия (ECMS — Enterprise Content Management System - системы управления содержанием предприятий),

•систему управления веб-содержимым (WCMS – Web Content Management System).

В силу того, что ECMS имеют глубокую внутреннюю классификацию по предметным областям (HRM, DMS, CRM, ERP и т. д.), термин CMS заместил собой WCMS, превратившись в синоним системы управления сайтами. Подобные CMS позволяют управлять текстовым и графическим наполнением веб-сайта, предоставляя пользователю интерфейс для работы с содержимым сайта, удобные инструменты хранения и публикации информации, автоматизируя процессы размещения информации в базах данных и её выдачи в HTML страницах.

Существует множество готовых систем управления содержимым сайта, в том числе и бесплатных. Их можно разделить на три типа по способу работы (рис. 2.1):

Рис. 2.1. Классификация CMS по способу работы

Генерация страниц по запросу. Системы такого типа работают на основе связки:

«Модуль редактирования → База данных → Модуль представления».

Модуль представления генерирует страницу с содержанием при запросе на него, на основе информации из базы данных. Информация в базе данных изменяется с помощью модуля редактирования. Страницы заново создаются сервером при каждом запросе, что в свою очередь создаёт дополнительную нагрузку на системные ресурсы.

Нагрузка может быть многократно снижена при использовании средств кэширования, которые имеются в современных веб-серверах. Смысл кэширования в следующем: для первого пользователя, пришедшего в магазин, страница действительно собирается, а для всех последующих посетителей она просто высылается в виде статик-контента, поскольку она уже была сохранена в кэш-памяти и в любой момент готова к отсылке. Данный способ снижает нагрузку на сервер в десятки раз. Кэш может обновляться как автоматически, по истечении некоторого срока времени или при внесении изменений в определённые разделы сайта, так и вручную по команде администратора.

Генерация страниц при редактировании. Системы этого типа — суть программы для редактирования страниц, которые при внесении изменений в содержание сайта создают набор статических страниц. При таком способе в жертву приносится интерактивность между посетителем и содержимым сайта.

Смешанный тип. Сочетает в себе преимущества первых двух. Может быть реализован путём кэширования — модуль представления генерирует страницу один раз, в дальнейшем она в несколько раз быстрее подгружается из кэша. Другой подход — сохранение определённых информационных блоков на этапе редактирования сайта и сборка страницы из этих блоков при запросе соответствующей страницы пользователем.

Разные CMS позволяют проектировать сайты различной сложности. Использование подобных систем позволяет разработчику не реализовывать заново стандартный функционал, а воспользоваться готовым решением, тем самым значительно сократить расходы на разработку. Именно фактом сокращения трудозатрат и объясняется достаточно широкое распространение CMS систем.

Преимущества использование CMS:

- абстрагирование от оформления сайта — сотрудник работает только над содержимым сайта;
- автоматизация задач по управлению сайтом;
- возможность создания различных по правам доступа частей сайта;
- данные хранятся не в виде файлов, а в реляционных СУБД, что значительно упрощает и ускоряет работу.

Все системы CMS можно разделить на следующие группы:

коммерческие «коробочные», «бесплатные» и собственной разработки (рис.2.2). Один и тот же «движок» может иметь как платную, так и бесплатную версию.

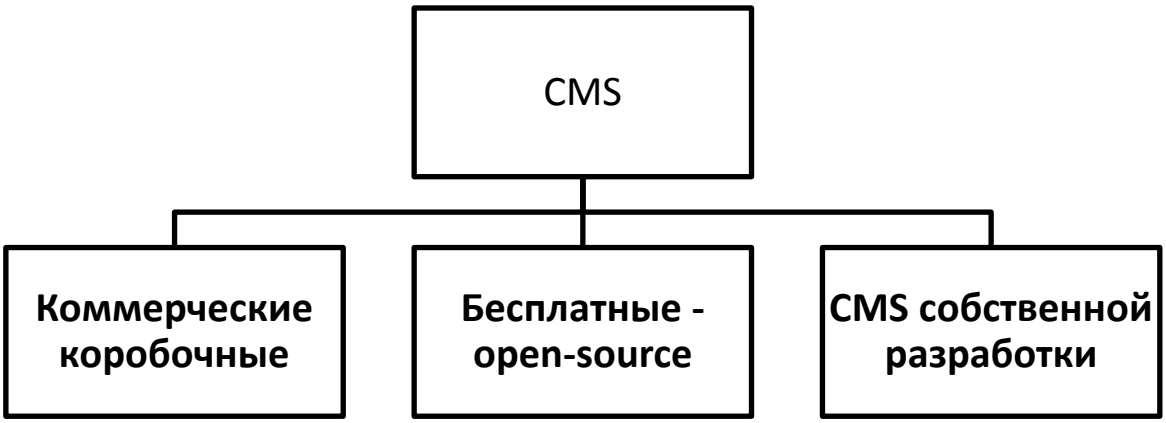


Рис.2.2

Коммерческие коробочные CMS

Коммерческие коробочные системы управления сайтами — продукты, созданные коммерческими организациями с целью извлечения прибыли от продажи лицензии и/или технической поддержки. Эти программные продукты, отчуждаемы от разработчика, позволяют самостоятельно разработать с их помощью сайт. Коммерческие системы управления контентом создают серьёзные компании — разработчики программного обеспечения, учитывая в разработанной CMS потребности типовых проектов, большого количества пользователей, и возможность дальнейшего развития. Примеры участников рынка: 1С-Битрикс, ABO.CMS, Amiro.CMS, Atilekt.CMS, diafan.CMS, DJEM, HostCMS, NetCat, S.Builder, SiteEdit, Twilight CMS, UMI.CMS.

Преимущества коммерческих CMS:

- официальная техническая поддержка;
- широкий выбор функционала, как со стороны клиентской части, так и с административной;
- отличная документация – руководство пользователя, разработчика и т.д. и т.п. Для некоторых систем проводится обучение клиентов использованию системы.

Недостатки коммерческих CMS:

- сложная система настроек;
- изменения программного кода запрещено;
- изменение дизайна возможно частично, возможна коррекция цвета или же добавить шаблоны;

Бесплатные CMS

Бесплатные «движки» имеют открытый код (open-source) – и распространяются с так называемой открытой лицензией. Это основное их

отличие от «коробочных». Открытый код таких CMS позволяет расширить возможности для модификации и решения нестандартных задач.

Бесплатные CMS — это программное обеспечение, отвечающее следующим условиям: программу можно свободно использовать с любой целью; доступность исходного текста программы; можно свободно распространять копии программы; программу можно свободно улучшать и публиковать свою улучшенную версию.

Основное достоинство бесплатных движков – это их доступность. Они весьма функциональны, имеют множество модулей, плагинов и шаблонов.

Главный их недостаток вытекает из их преимущества. Так как скрипт движка может скачать любой пользователь интернет, то и любой желающий может попробовать найти уязвимости в нем для взлома сайта, построенного на этой CMS, или заражения его вирусами.

Примеры участников рейтинга: WordPress, Joomla, uCoz, Drupal, CMSMade Simple, Danneo, DataLife Engine, MODx, TextPattern, TYPO3.

Преимущества бесплатных CMS:

- большой выбор систем - очень большое количество компаний, а особенно фрилансеров, занимающихся их разработкой;
- полный доступ к коду и базе данных, много бесплатных дополнений и модулей;
- не требовательны к хостингу;

Недостатки бесплатных CMS:

- отсутствие официальной технической поддержки;
- техническую поддержку, преимущественно осуществляют фрилансеры, нет гарантии их высокой квалификации;
- как и многие популярные проекты с открытым исходным кодом, базовые версии CMS систем очень ненадёжны, на практике их функционал необходимо расширять дополнительными модулями, код которых тоже открыт и может быть доступен злоумышленнику;
- для настройки системы пользователь должен обладать достаточной технической квалификацией — знание html, основ программирования и работы с БД.

CMS собственной разработки

Системы управления собственной разработки есть практически у каждой веб-студии, однако в последнее время их популярность снижается, так как код таких CMS закрытый.

Данные продукты отвечают полностью задачам, которые призван решать конкретный сайт, но их повторное использование затруднено. Использование собственных разработок систем управления сайтом связано с риском того, что клиент фактически будет в дальнейшем привязан к создателям.

2.3. Структура CMS

Все системы управления контентом (CMS) представляют собой некоторое программное средство, устанавливаемое на Web-сервере и предназначаемое для создания и обслуживания динамических сайтов.

Все они в том или ином объеме предполагают отделение контента от дизайна, поддержку бизнес-процессов и минимизацию программистских усилий при разработке сайтов. Обобщенная структура CMS представлена на рис. 2.3.

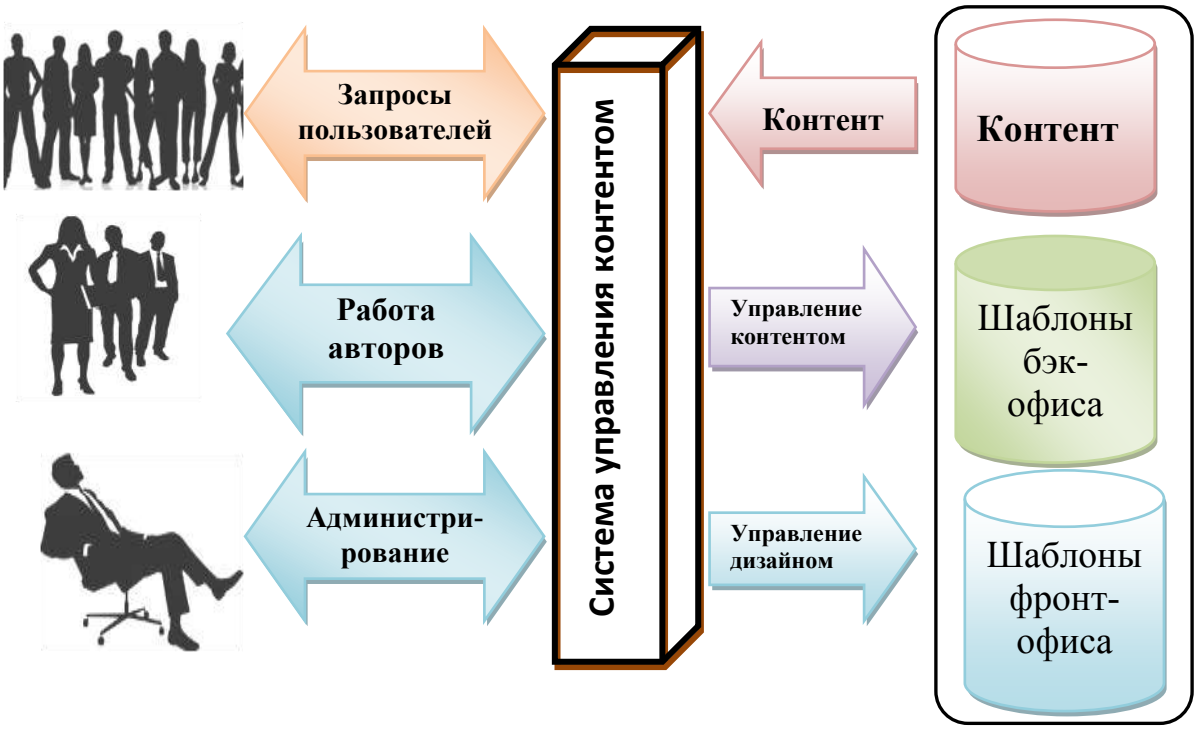


Рис. 2.3. Обобщенная структура CMS

В системе CMS присутствует два хранилища. В первом (обычно реляционная СУБД) хранятся все данные, которые публикуются на сайте. Во втором (это может быть и файловая система) хранятся элементы представления – шаблоны фронт-офиса и бэк-офиса, графические изображения и т.д.

Существует минимум два специализированных рабочих места.

Первое рабочее место – для разработчиков сайта и администраторов. С его помощью они задают структуру сайта, структуру контента, определяют внешний вид сайта, настраивают шаблоны представления информации. Этот инструментарий обычно не полностью автоматизирован. Для настройки сайта разработчики частично работают через средства CMS, часть информации размещается напрямую.

Второе рабочее место – для владельцев сайта (авторов). Оно позволяет сотрудникам компании самостоятельно размещать информацию на сайте, без участия разработчиков. Менеджеры заказчика работают только через специализированное рабочее место.

CMS предоставляет возможность оперативного обновления информации сотрудником – информацию публикует сотрудник, владеющий информацией, без дополнительных посредников в виде технических специалистов. Результатом является снижение стоимости поддержки – обновление информации производится самостоятельно, нет необходимости оплачивать труд собственного или внешнего web-мастера.

Традиционная конфигурация хранилища в CMS представлена на рис. 2.4.

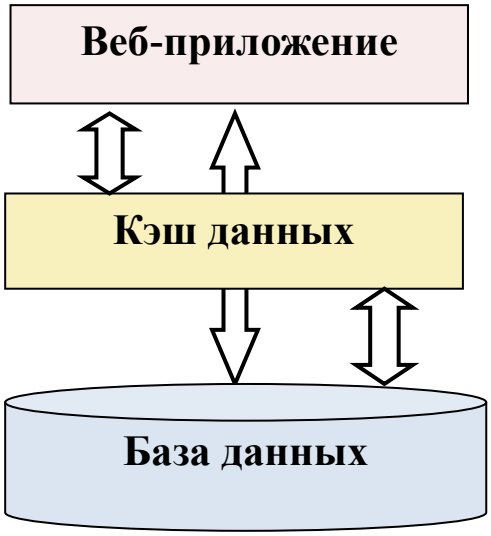


Рис. 2.4. Конфигурация хранилища в CMS

Работа виртуального предприятия под нагрузкой – одно из слабых мест в электронном бизнесе. В отличие от статичной страницы, когда пользователю просто передается уже готовая Web-страница за десятки миллисекунд, то в динамическом сайте страница сначала собирается системой, а затем уже отправляется пользователю примерно за 1,5 сек.

Смысл кэширования в следующем: для первого пользователя, пришедшего в магазин, страница действительно собирается, а вот для всех последующих посетителей она просто высылается в виде статического контента, поскольку она уже была сохранена в кэш-памяти и в любой момент готова к отсылке. Данный способ снижает нагрузку на сервер в десятки раз.

Существует и более сложное кэширование – так называемое "*горячее кэширование*", когда после запроса администратора все страницы сайта заранее собираются и в таком виде хранятся в ожидании запроса.

Определим некоторые методы кэширования модулей CMS:

•Кэширования всех страниц.

Идея метода заключается в том, чтобы по множеству входных параметров и их диапазонам воссоздать множество всех HTML-страниц создаваемых модулем и сохранить. Далее при обращении пользователя к модулю CMS передавать ему сохраненную HTML-страницу. Если хотя бы один параметр модуля имеет бесконечное количество допустимых значений, то метод неприменим. Однако это ограничение можно обойти, путем выделения подмножества параметров имеющих конечное количество

допустимых значений и кэширования результатов работы модуля на этих параметрах.

•Кэширование при обращении.

При обращении пользователя к модулю сначала сохранить результат работы модуля в HTML-страницу, а затем передать пользователю сохраненную страницу. При повторном обращении с тем же диапазоном входных параметров передавать пользователю сохраненную HTML-страницу.

Для крупных корпоративных и информационных порталов, работающих с большой нагрузкой, используются распределенные базы данных – веб-кластеры и облачные хранилища.

Облачное хранилище данных — модель онлайн-хранилища, в котором данные хранятся на многочисленных распределённых в сети серверах, предоставляемых в пользование клиентам, в основном, третьей стороной. В отличие от модели хранения данных на собственных выделенных серверах, приобретаемых или арендуемых специально для подобных целей, количество или какая-либо внутренняя структура серверов клиенту, в общем случае, не видна. Данные хранятся и обрабатываются в так называемом облаке, которое представляет собой, с точки зрения клиента, один большой виртуальный сервер.

Физически же такие серверы могут располагаться удалённо друг от друга географически, вплоть до расположения на разных континентах. Такие технологии успешно используются крупнейшими интернет- магазинами, например, Amazon.com, компанией Google и др.

Модуль Веб-кластер - это комбинация технологических решений, которые позволяют распределить один сайт на несколько серверов, решая тем самым несколько задач: обеспечение высокой доступности сайта; его масштабирование в условиях возрастающей нагрузки; балансирование нагрузки, трафика, данных между несколькими серверам. Система позволяет снимать резервные копии со специально выделенных узлов кластера, не влияя на работу сайта.

«Географический веб-кластер» повышает отказоустойчивость проекта и обеспечивает независимость от дата-центра (специализированного помещения или здания для размещения серверного и сетевого оборудования). В различных дата-центрах объединяются несколько групп веб-кластеров, находящихся в разных городах или странах. В случае отказа одного дата-центра, в работу мгновенно включается другой, без необходимости восстановления «бэкапа» (резервного копирования). На рис. 2.5 представлена обобщенная схема реализации таких технологий, используемая в CMS 1С-Битрикс.

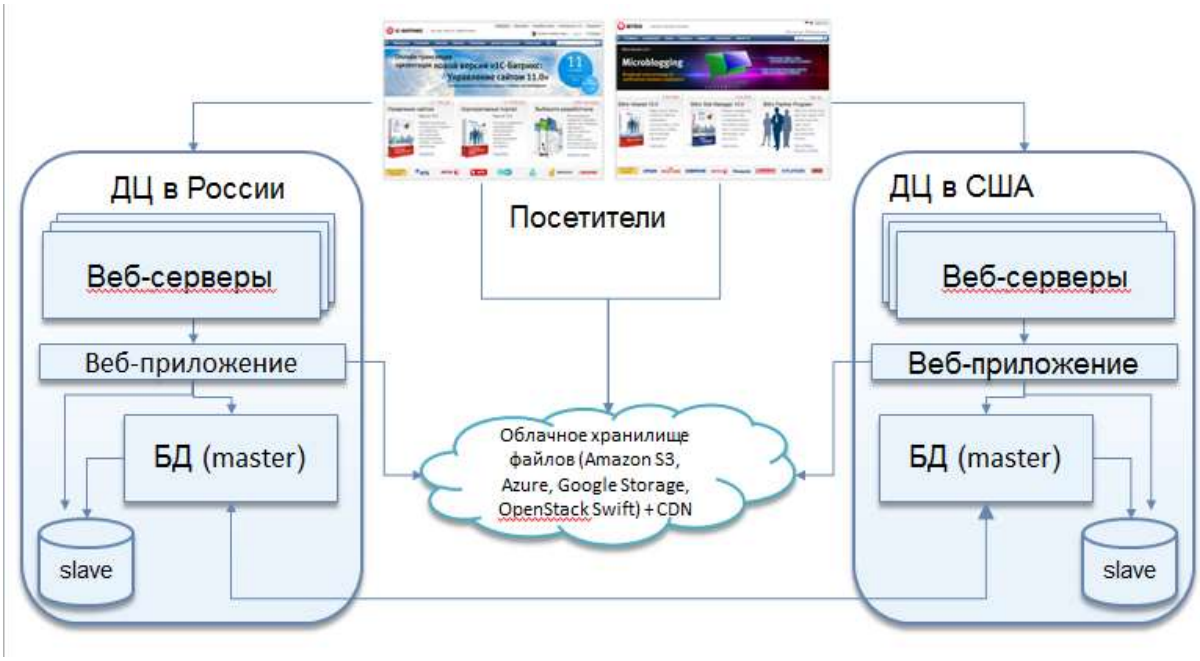


Рис. 2.5. CMS с распределенной базой данных

В рамках CMS функционально реализованы многие сервисы – поиск, форумы, голосования и опросы – то есть коммуникации и обратная связь. Решаются вопросы рекламы, web-аналитики и безопасности. Для коммерческих проектов через CMS происходит интеграция с бизнес-процессами, подключение к системам оплаты и бухгалтерским службам. То есть наиболее востребованная функциональность уже реализована в CMS и может быть сразу использована.

На рис. 2.6 представлены сервисы, предлагаемые CMS 1С-Битрикс

Рис. 2.6. Сервисы, предлагаемые CMS 1С-Битрикс

2.4. Модули CMS

При создании сайта на основе CMS полностью или частично используются готовые модули, которые уже прошли неоднократное тестирование. CMS разделяют данные и их представление, следовательно, внешний вид сайта меняется с намного меньшими затратами.

Именно подключаемые модули CMS, обеспечивают сайту те функции и сервисы, которые позволяют наиболее эффективно взаимодействовать web-ресурсу с посетителями сайта. В подобных системах контент разделен на отдельные модули по типам содержимого. Структура данных зависит от модуля, и вся работа с контентом сосредоточена внутри модуля. Модули независимы и полностью отвечают за работу с документами данного типа.

На рис. 7 представлена типовая страница сайта в CMS Amiro. Шаблон страницы разделен на пять частей. В середине – основное содержание страницы, а в правую и левую колонки, а также в верхнюю и нижние части встраиваются типовые модули.

Сам шаблон страницы задается с помощью фиксированного набора характеристик — типы документов строго фиксированы. Расширять функциональность можно за счет добавления нового модуля, замены или редактирования существующего.

Рис. 2.7. Шаблон страницы CMS Amiro

Модули автономны и независимы друг от друга. Нет никакой системы связей между документами разных модулей и между документами одного и того же модуля.

Стандартный набор типов модулей таков:

- ссылки,
- статьи,
- файлы,
- новости,
- разделы,
- форум.

Несмотря на очевидную ограниченность модульной модели данных, системы на такой основе наиболее популярны благодаря своей простоте.

У модульных CMS-систем есть один общий недостаток — строго фиксированная в пределах модуля структура содержимого. Однако для расширения их функциональности можно воспользоваться внешними модулями, которых в сети Интернет немало. Очевидное преимущество этих систем — возможность получения почти полностью готового к использованию портала за короткое время.

Отдельный набор модулей используется для администрирования сайта, исследования статистики его посещений, поисковой оптимизации, ведения бизнеса и др.

Перечень модулей достаточно объемен:

коммуникатор (компрессия, кеширование); SEO-оптимизатор (оптимизация работы с поисковым продвижением); пользователи; новости; статьи; блог (сетевой дневник); теги; стикеры; комментарии (для новостей, статей, каталога и др); подписка, рассылка; опросы; вопрос-ответ; обратная связь; форум; гостевая; вакансии; файловый архив; фотоальбомы; доска объявлений; каталог продукции; каталог-портфолио; каталог - база данных; поиск по сайту; управление рекламой; RSS экспорт (для новостей, статей, блога, форума и др); защита от спама (в форуме, обсуждениях, формах); сервисные модули (автоматическое обновление системы, резервное копирование данных); Plug-ins (добавление скриптов сторонних разработчиков) и др.

Остановимся подробнее на наиболее важных.

•Менеджер сайта (рис. 2.8) – один из ключевых модулей системы Amiro.CMS, позволяющий управлять всей структурой и навигацией сайта, макетами, дизайном, созданием статических и динамических страницами, создавать страницы с модулями, перемещать и удалять страницы, задавать ключевые слова и ссылки, осуществлять групповые операции с картой сайта и многое другое.

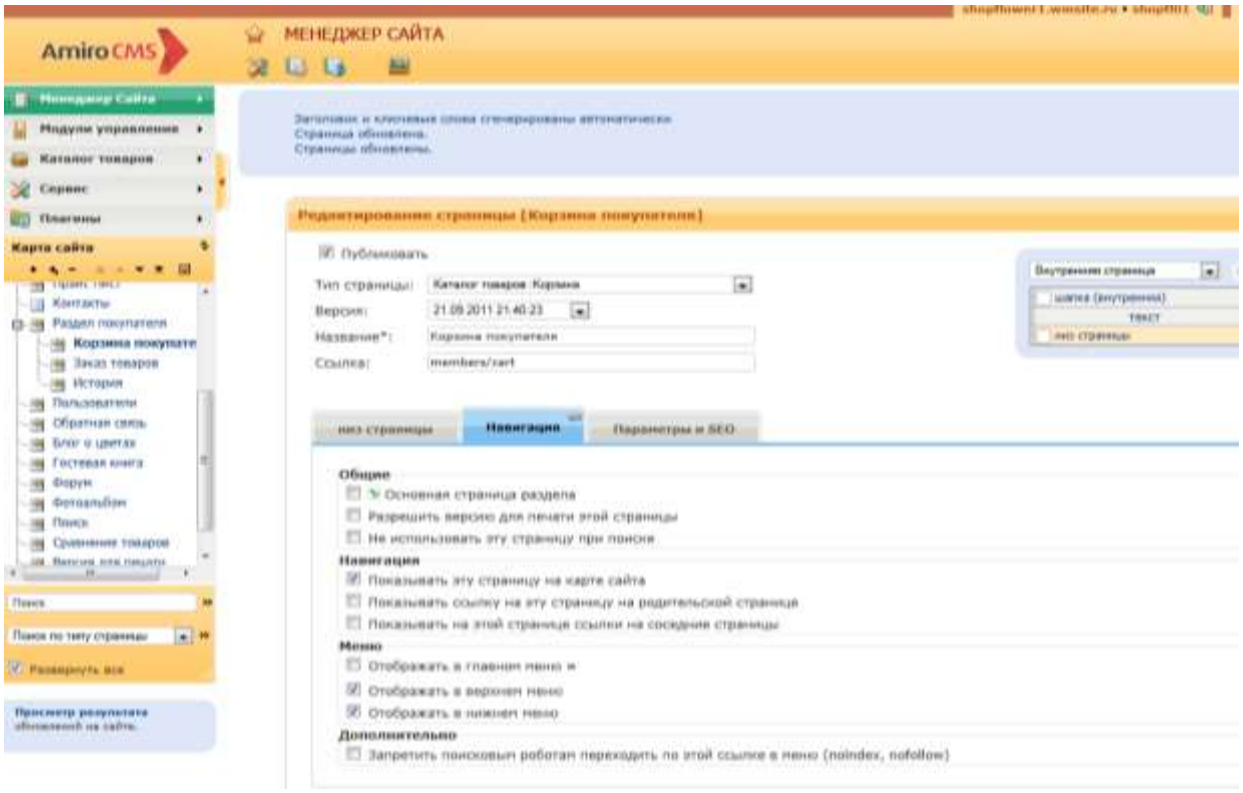


Рис. 2.8. Менеджер сайта Amiro.CMS

Amiro.CMS имеет удобный визуализированный процесс управления всем сайтом. Структурно менеджер состоит из:

-Панели отображения карты сайта,

-Панели управления страницей/макетом,

-Визуального редактора,

-Панели дополнительных свойств страницы.

Остановимся подробнее на основных модулях систеи управления контентом.

•Модуль <Опросы> позволяет администратору узнавать мнение посетителей по различным вопросам, а также получать дополнительную информацию о посетителях. Опросы могут выводиться как отдельным списком, так и в виде специального блока в боковых столбцах других страниц сайта.

•Модуль <Рейтинги> позволяет посетителям сайта оценивать опубликованные в нем контент: новости, товары, услуги.

•Модуль <Скидки> с модулем <Купоны> позволяет управлять скидками на товары интернет-магазина и персональными скидками покупателей.

•Модуль <Купоны> позволяет создавать купоны и категории купонов для использования в модуле скидок.

•Модуль <Обмен данными> позволяет выполнять импорт и экспорт данных интернет магазина и других модулей в различных форматах: экспорт данных в XML для Яндекс.маркет, Рамблер, обмен данными с Microsoft Excel (csv), интеграция с 1С.

•Модуль <Каталог - база знаний> предназначен для управления структурированными данными, опубликованными на сайте: справочниками, энциклопедиями, словарями, пр.

•Модуль <Реклама> служит для установки рекламных мест на странице сайта, удобного управления рекламными материалами, их выводом на сайт, подсчета статистики и управления балансом пользователей.

•Модуль <Статистика> служит для учета посещаемости интернет ресурса. В разделе «Статистика» можно получить информацию о поисковых запросах, посещаемых страницах и информацию о функционировании роботов, предназначенных для индексирования сайта в поисковых системах (рис.2.9).

Рис. 2.9. Статистика активности сайта.

2.5. Сравнительный анализ CMS

Анализ рынка коммерческих CMS

Компанией Itrack проводится регулярный мониторинг систем управления контентом, установленных на сайтах. На основе полученных данных был составлен рейтинг систем управления контентом.

За II полугодие 2013 г. было опрошено 4 747 414 доменов зоны ru, прирост доменов по сравнению с первым полугодием 2013 года составил более 494 000 (11,6 %).

67,9% (+1,2%) опрошенных доменов ответили в течение 30 секунд, а CMS обнаружена на 20,8% (+2,2%) доменов. Доля платных тиражных CMS составляет примерно 13,1% (-0,3%) от общей доли обнаруженных CMS, а доля узкоспециализированных — 3,5%.

При составлении этого рейтинга была использована новая технология сканирования сайтов. Благодаря этому значительно увеличилось количество найденных установок CMS.

В рейтинге принимают участие следующие CMS:

Платные тиражные CMS:

1С-Битрикс, ABO.CMS, Amiro.CMS, diafan.CMS, HostCMS, ImageCMS, Jimdo, NetCat, S.Builder, Simpla, SiteEdit, UMI.CMS.

Бесплатные CMS:

CMS Made Simple, Danneo, DataLife Engine, Drupal, InstantCMS, Joomla, LiveStreet, MaxSite CMS, MODx, Textpattern, TYPO3, uCoz, WordPress.

Узкоспециализированные CMS:

WebAsyst Shop-Script, VamShop, ShopCMS, InSales, PHPShop, PHPShop Free, OsCommerce, Melbis Shop, AdVantShop.NET.

В группу узкоспециализированных CMS включены системы, использующиеся исключительно для создания узкой категории сайтов: например, интернет-магазинов.

На рис. 2.10 приведены данные по наиболее популярным системам среди всех категорий.

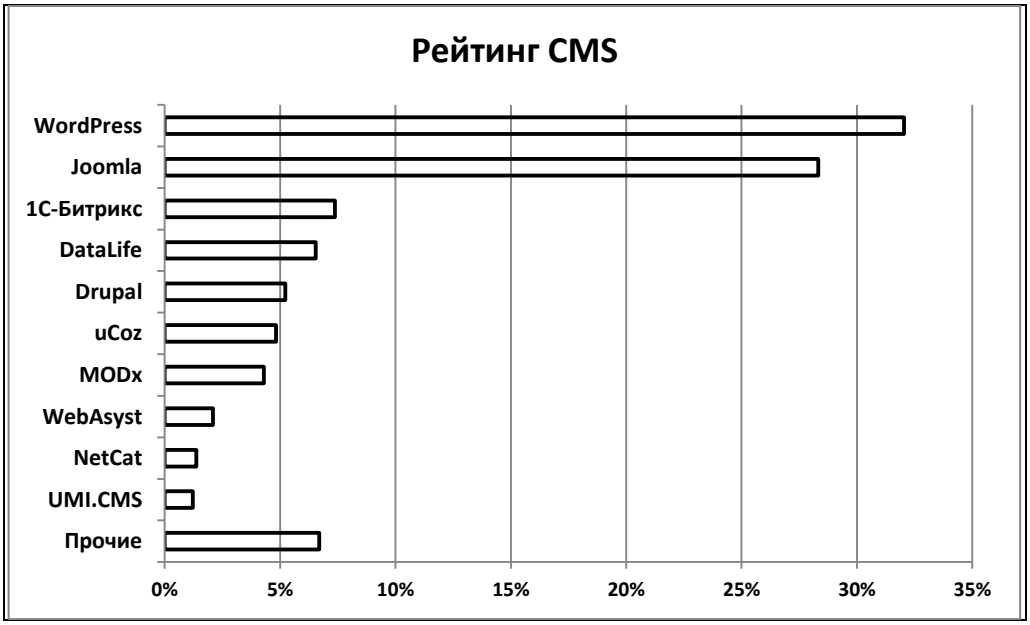


Рис. 2.10. Общий рейтинг CMS

В первых строчках рейтинга фигурируют свободно распространяемые WordPress и Joomla, а также система управления 1С-Битрикс, DataLife Engine, распространяемая как на коммерческой, так и на бесплатной основе. Далее следуют Drupal, uCoz и другие продукты, доля которых колеблется в районе 5 процентов. Популярность системы WordPress очевидна – она проста в установке, управлении и использовании, а Joomla привлекает российских интернет-пользователей огромной базой дополнительных компонентов для данной платформы и внушительному количеству тем для оформления сайтов.

Для получения более точной информации о долях рынка общий рейтинг Систем Управления Контентом, можно поделить на рейтинг платных тиражных CMS (рис. 2.11, 2.12 и 2.13) и рейтинг бесплатных CMS (рис. 2.14, 2.15, 2.16).

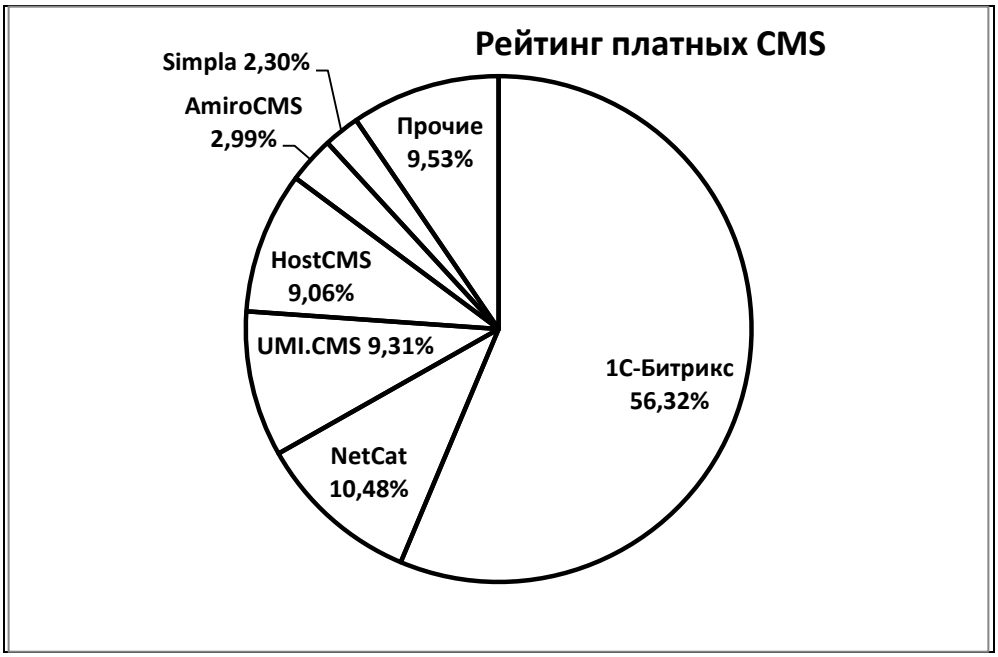
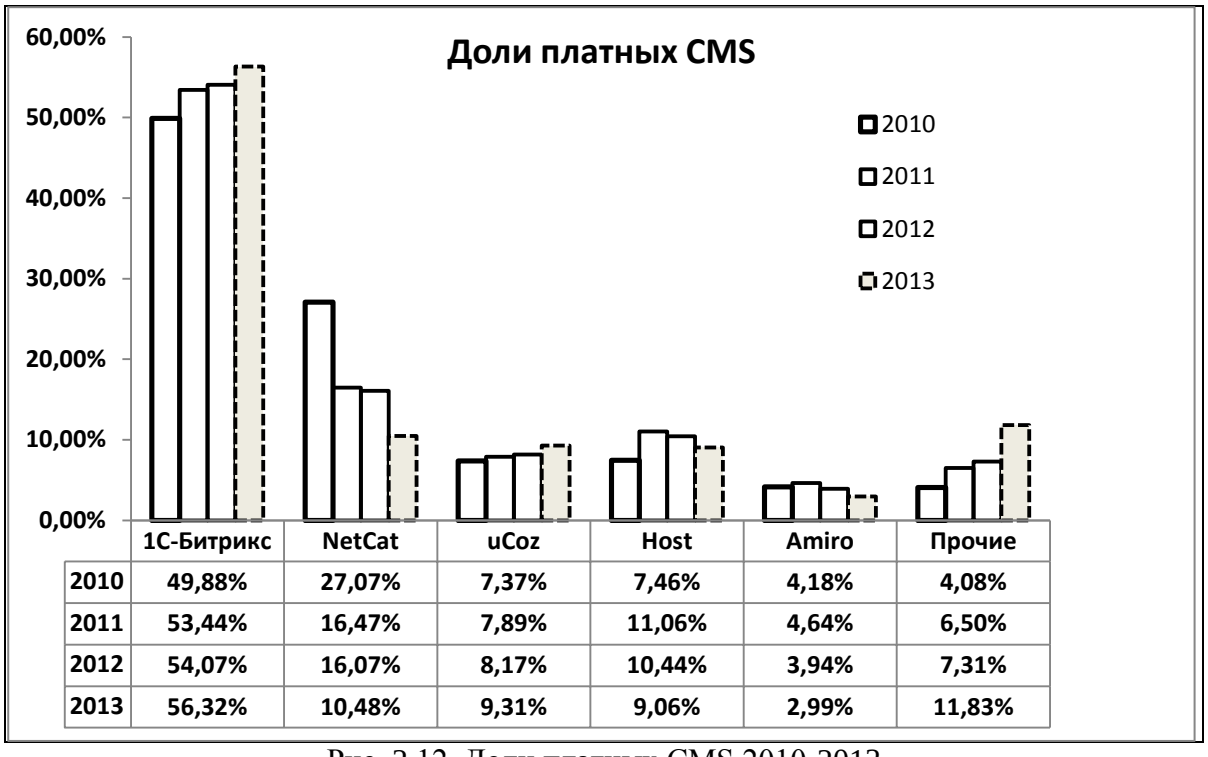
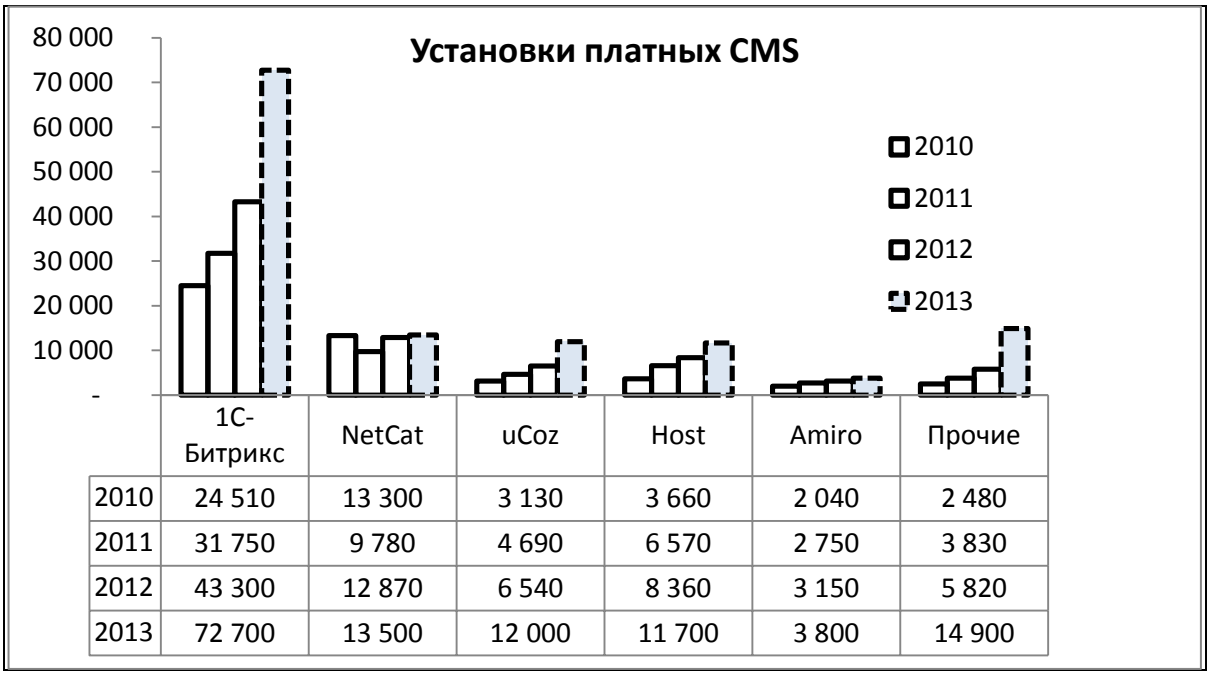


Рис. 2.11. Рейтинг платных тиражных CMS



	1С-Битрикс	NetCat	uCoz	Host	Amiro	Прочие
2010	49,88%	27,07%	7,37%	7,46%	4,18%	4,08%
2011	53,44%	16,47%	7,89%	11,06%	4,64%	6,50%
2012	54,07%	16,07%	8,17%	10,44%	3,94%	7,31%
2013	56,32%	10,48%	9,31%	9,06%	2,99%	11,83%

Рис. 2.12. Доли платных CMS 2010-2013



	1С-Битрикс	NetCat	uCoz	Host	Amiro	Прочие
2010	24 510	13 300	3 130	3 660	2 040	2 480
2011	31 750	9 780	4 690	6 570	2 750	3 830
2012	43 300	12 870	6 540	8 360	3 150	5 820
2013	72 700	13 500	12 000	11 700	3 800	14 900

Рис. 2.13. Количество установок платных CMS 2010-2013 г.

Исследования показывают больше 50% рынка находится у 1С-Битрикс, которая с каждым годом укрепляет свои позиции на рынке. Следующие за лидером NetCat, UMI.CMS и Host.CMS близки по своим показателям. Популярность двух последних заметно возрастает.

Обзор коммерческих систем управления контентом

Рассмотрим лидеров рынка коммерческих систем управления контентом.

Логотип NetCat

NetCat

Система рассчитана на использование для следующих видов сайтов: корпоративные представительства, интернет-сервера портального типа, библиотеки данных, файл-архивы, интернет-издания, СМИ, электронные магазины и прочее, в т.ч. сложные интерактивные веб-системы.

Система администрирования в NetCat разделена на две части: интерфейс пользователя и интерфейс разработчика. Для использования системы не требуется знание интернет-технологий, языков программирования и разметки. Интерфейс системы прост и интуитивно понятен для пользователя, имеющего опыт работы на компьютере.

Стандартные возможности системы (создание рубрикатора, адаптация дизайна, наполнение содержанием, администрирование), необходимые для большинства сайтов, могут легко дополняться нестандартными решениями для электронной коммерции, каталогами различного типа, системами статистики, системами управления рекламой.

Кроме стандартной конфигурации NetCat, возможна гибкая адаптация системы под нужды заказчика.

К заметным недостатка данной системы можно отнести то, что программный код смешивается с html- разметкой, отсутствие системы контроля версий и требовательность системы - даже для отображения простейшей страницы требуется довольно большое объем SQL- запросов.

Далее следует активно развивающаяся система 1С-Битрикс.

Логотип 1С- Битрикс

1С-Битрикс

Профессиональная система для создания и управления интернет-проектами: корпоративными сайтами, интернет-магазинами, информационными порталами, интернет –сообществами, социальными сетями и другими сайтами.

Главное преимущество системы – это тесное взаимодействие сайта с программой «1с Предприятие». Хорошо разработанный процесс покупки и продажи товаров. Позволяет на базе одной системы создавать, как и внешний сайт, так и внутренний сайты. На внешнем сайте могут находиться магазин и информация для клиентов, на внутреннем – сервисы и информация для сотрудников компании.

Для разработки своих компонентов есть специальный механизм, который позволяет брать за основу существующие компоненты и их расширять. Еще одна особенность: кроме панели администратора, система позволяет пользователям управлять содержимым сайта прямо с самого сайта, не заходя в панель администратора. Постоянная техническая поддержка. Использование только одной технологи PHP. Обилие документации для разработчиков и для пользователей на русском языке.

Эту систему лучше использовать, если планируется завязать внешний и внутренний сайт в единую базу и если нужно будет использовать «1с-предприятие» непосредственно для электронной торговли. Довольно удобная система для разработчиков и пользователей. Покрывает большинство потребностей компании и не требует значительного времени на разработку.

Следующей системой управления контентом следует HostCMS.

Логотип HostCMS

HostCMS

Удобная современная система управления сайтами. Для ежедневной работы с CMS не понадобится дополнительно обучать сотрудников – корректировка новостей, пресс-релизов и содержания сайта производится с использованием интуитивно-понятного интерфейса

К числу достоинств данной системы можно отнести систему сжатия (компрессии), резервное копирование, постоянный доступ ко всем обновлениям системы.

В отличие от конкурентов, HostCMS предлагает возможность создания на разных доменах (в том числе и на поддоменах) различных сайтов, управляемых одним экземпляром системы управления. Это удобно при разработке нескольких сайтов для одного клиента.

Обновленный интерфейс системы позволяет в процессе ввода данных проверять корректность и в случае ошибки сообщает о необходимости корректировки или запрещает вставку данных. Использование системы кэширования позволяет значительно сократить нагрузку на сервер. Как следствие, увеличивается скорость загрузки страниц и уменьшается трафик пользователя. Есть возможность простого проведения SEO-оптимизации для поисковых систем, можно указать мета-теги для каждой отдельной страницы сайта.

К недостаткам можно отнести плохо развитое сообщество данной системы.

Последней из списка коммерческих систем управления контентом следует UMI.CMS

Логотип UMI.CMS

UMI.CMS

Коммерческая система управления сайтом (CMS), написанная на языке программирования PHP. Создаётся с 2004 года командой российских разработчиков «Юмисофт». В массовую продажу поступила в 2007 году. Существует в бесплатной и коммерческой версиях.

Обе версии специально разработаны для небольших коммерческих сайтов малого и среднего бизнеса. Единственное отличие бесплатной версии – ограничение числа страниц сайта десятью. Продукт позволяет добавлять новые разделы и страницы, редактировать структуру в наглядной форме "дерева", редактировать страницы и вести историю их изменений с возможностью «отката» к предыдущим версиям, публиковать новости сайта.

Система имеет удобный интерфейс, который одинаково удобен в освоении и использовании как для разработчиков сайтов, владельцев сайтов и их пользователей. Система строится на основе сложения модулей, каждый из которых добавляет дополнительную функциональность в систему.

Отличительной чертой системы так же является функция "Edit-in-Place" ("редактирование на месте" – без входа в административный интерфейс). Технология Edit-In-Place позволяет управлять сайтом на самом сайте, аналогично редактированию документа в Word, без использования административного интерфейса CMS.

В UMI.CMS осуществлена полная интеграция с торговыми конфигурациями платформы «1С: Предприятие», обеспечивающая импорт-экспорт данных в двустороннем порядке. Так же в UMI.CMS интегрированы

сервисы Яндекса. Данная система идеально подходит для создания сайтов с несложным функционалом. К сожалению, при выполнении трудных задач данная система уступает по функциональным возможностям в сравнении с аналогами.

Анализ рынка бесплатных систем управления контентом

Данные мониторинга систем управления контентом, установленных на сайтах, проводимые Компанией Itrack, представлены на рис. 2.14, 2.15 и 2.16.

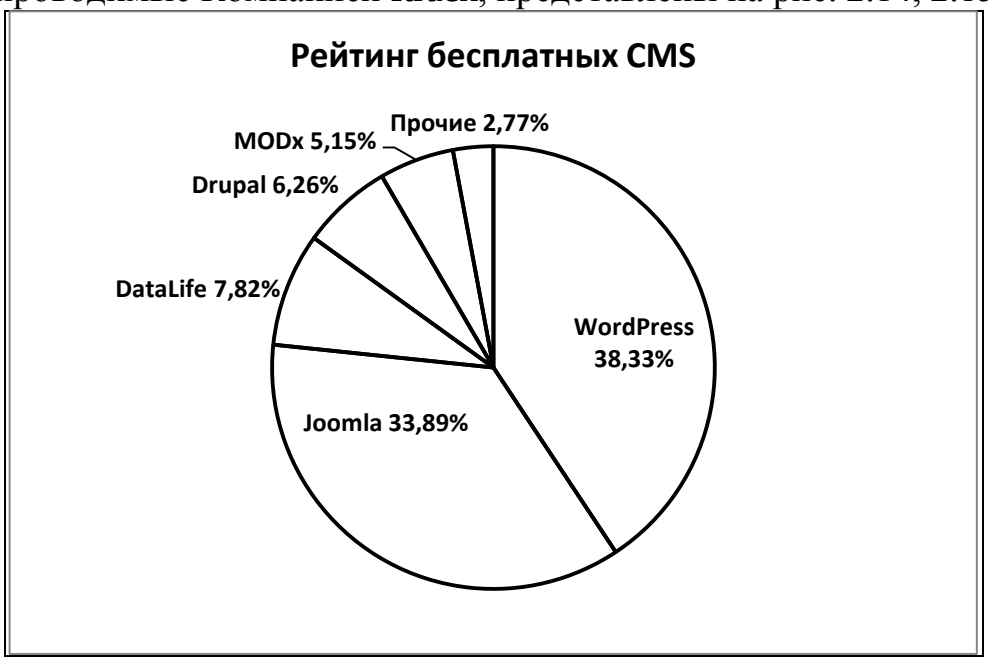
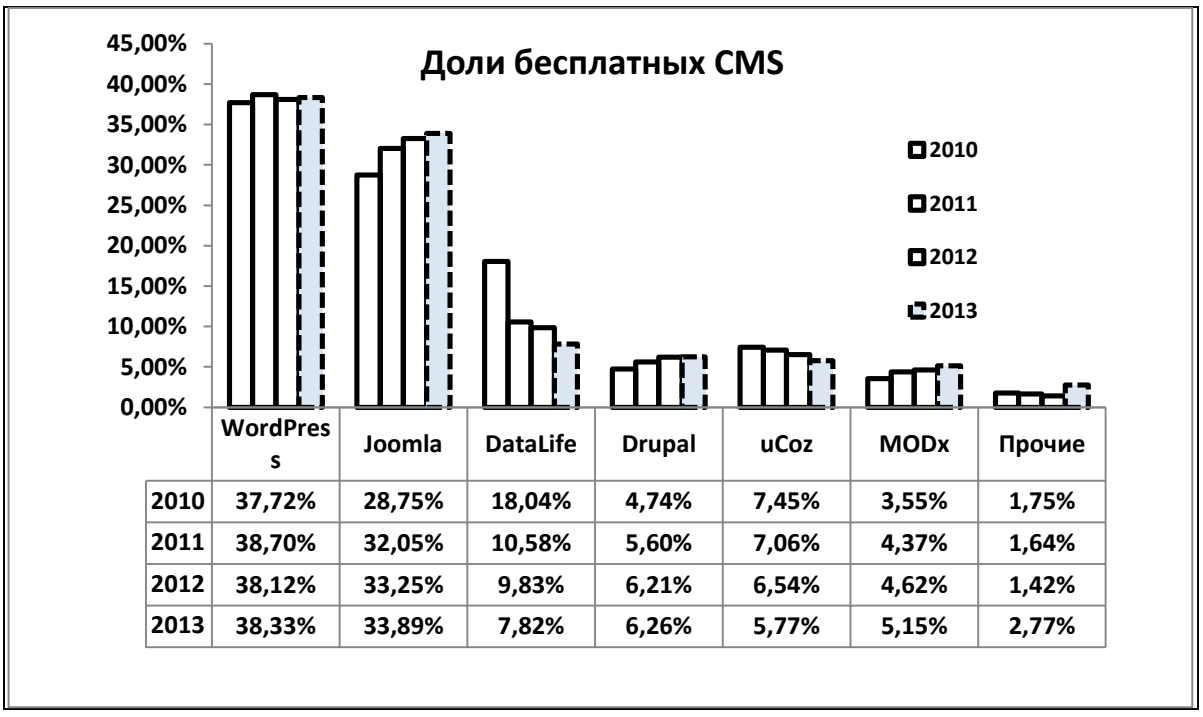
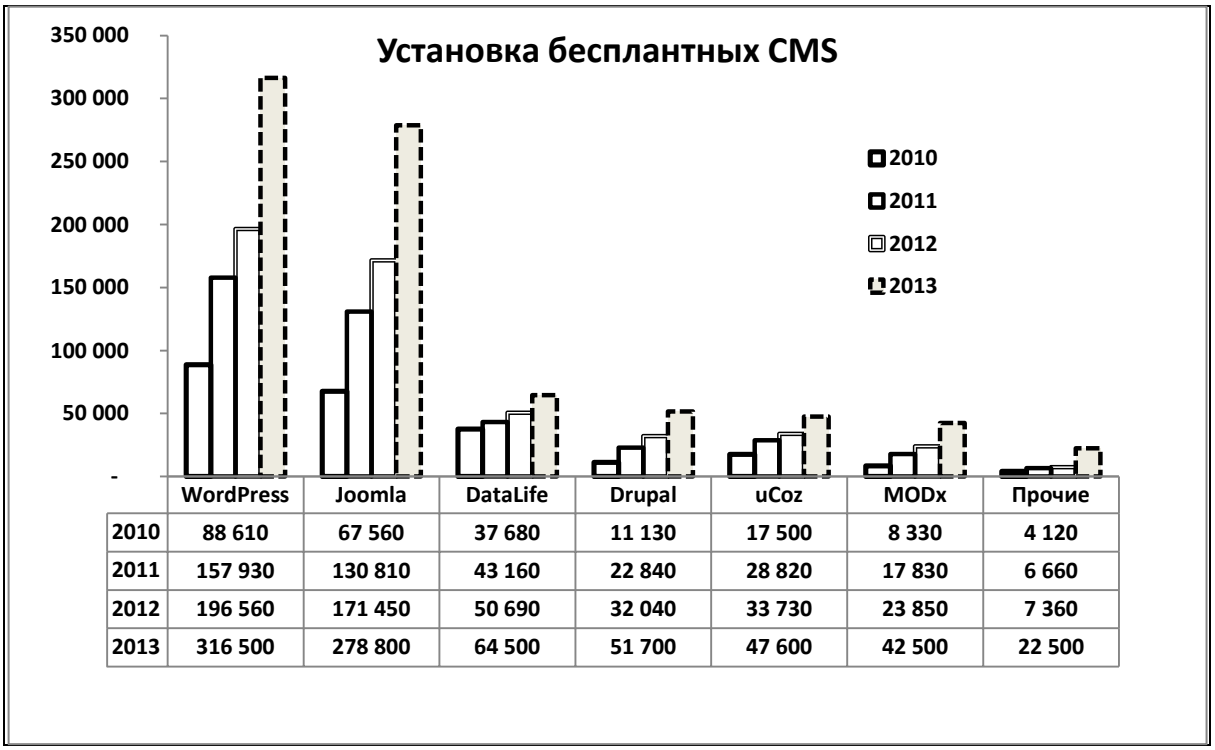


Рис. 2.14. Рейтинг бесплатных CMS



	WordPress	Joomla	DataLife	Drupal	uCoz	MODx	Прочие
2010	37,72%	28,75%	18,04%	4,74%	7,45%	3,55%	1,75%
2011	38,70%	32,05%	10,58%	5,60%	7,06%	4,37%	1,64%
2012	38,12%	33,25%	9,83%	6,21%	6,54%	4,62%	1,42%
2013	38,33%	33,89%	7,82%	6,26%	5,77%	5,15%	2,77%

Рис. 2.15. Доли бесплатных CMS 2010-2013



	WordPress	Joomla	DataLife	Drupal	uCoz	MODx	Прочие
2010	88 610	67 560	37 680	11 130	17 500	8 330	4 120
2011	157 930	130 810	43 160	22 840	28 820	17 830	6 660
2012	196 560	171 450	50 690	32 040	33 730	23 850	7 360
2013	316 500	278 800	64 500	51 700	47 600	42 500	22 500

Рис. 2.16. Количество установок бесплатных CMS 2010-2013

Рынок бесплатных систем управления контеном также успешно развивается. Львиную долю рынка составляют Joomla и WordPress – более 70% и практически поровну делят ее между собой. И на третьей позиции находится DataLife Engine – около 8%.

Обзор бесплатных CMS

Первым из бесплатных систем управления контентом следует WordPress.

Логотип Wordpress

WordPress

Самая распространенная на данный момент система управления сайтом. До недавнего времени позиционировалась, как максимально простой блоговый движок. С выходом 3-ей версии, возможности CMS расширились и ее сфера применения от блогов до достаточно сложных новостных ресурсов и интернет-магазинов.

Встроенная система «тем» и «плагинов» вместе с удачной архитектурой позволяет конструировать практически любые проекты - форумы, интернет-магазины, галереи и др.

Администрирование сайта на WordPress очень простое. Система имеет огромное количество учебных материалов и, самое главное, можно найти практически любую документацию по WordPress на родном языке.

У данной системы существует ряд недостатков.

Система имеет ряд уязвимых мест, которые снижают ее безопасность. Но при контроле за выходом новых версий продукта и своевременном

обновление в большинстве случаев безопасность гарантируется. Кроме того большое количество установленных плагинов нагружают систему, что ведет к уменьшению скорости загрузки и при большом трафике, могут привести к проблемам с доступностью сайта. Частично проблемы решаются кешированием страниц

Следующей по популярности следует система управления контентом Joomla.

Логотип Joomla!

Joomla

Очень мощная и гибкая система управления сайтом, имеет уникальный настраиваемый шаблон, в котором в принципе можно решить задачу любой сложности. Огромное количество учебных материалов по Joomla на русском языке, что значительное упрощает знакомство с данной системой и помогает при решении задач сложного уровня.

К сожалению Joomla является одной из самых уязвимых систем. Это связанно в основном не с ядром системы, которое своевременно обновляется, устраняя проблемы с безопасностью, а с самим принципом открытого кода системы. Посторонние разработчики сами выпускают большое количество расширений, использование некоторых из них приводит к уязвимости сайта.

Имеются и проблемы с производительностью для сайтов с большой посещаемостью.

Еще одним недостатком является не очень хорошая индексируемость сайтов выполненных на Joomla. То есть в плане поисковой оптимизации система не очень хорошо настроена.

Joomla имеет достаточно сложную настройку сайта на начальном периоде, что создаст определенные сложности для новичков.

Следующей в рейтинге следует система управления контентом DataLife Engine

Логотип DataLIfe Engine

DataLife Engine

Система предлагает многопользовательский новостной движок, обладающий большими функциональными возможностями. Движок предназначен, в первую очередь, для создания новостных блогов и сайтов с большим информационным контентом.

Однако он имеет большое количество настроек, которые позволяют использовать его практически для любых целей. Движок может быть интегрирован практически в любой существующий дизайн и не имеет никаких ограничений по созданию шаблонов для него.

Еще одной ключевой особенностью DataLife Engine - является низкая нагрузка на системные ресурсы. Даже при очень большой аудитории сайта нагрузка не сервер будет минимальной и посетитель не будете испытывать каких-либо проблем с отображением информации.

Движок оптимизирован под поисковые системы, что приведет на сайт дополнительных клиентов. Использование продвинутой технологии AJAX позволяет сэкономить трафик владельцев и посетителей, а также снижает нагрузку на сервер.

К недостаткам можно отнести то, что данный движок имеет небезопасный алгоритм работы с параметрами запросов.

Последней из списка бесплатных систем управления контентом идет uCoz

Логотип uCoz.

uCoz

Современная бесплатная система управления сайтом, которая работает по принципам Web 2.0 и позволяет создавать сложнейшие проекты с необычайной простотой и легкостью.

Удобная панель управления позволяет быстро добавлять/ редактировать/ удалять информацию на сайте, а гибкая система различных настроек и управления доступом – придать любой дизайн и наделить необходимой функциональностью.

Модули системы могут быть оптимизированы и настроены под конкретный проект, что позволяет создавать более сложные сайты: простой сайт-визитку, крупное интернет-представительство компании, информационный портал и т.п.

Систему uCoz не нужно скачивать и устанавливать, достаточно просто зарегистрироваться, и система полностью в Вашем распоряжении.

Особенностью системы является специальный язык шаблонов, позволяющий выполнять проверки различных условий в момент генерации страницы. Также конструктор шаблонов, позволяющий быстро вносить изменения в дизайн всего сайта или вносить коррективы. Система строится на системе модулей, что позволяет выполнять несложные задачи. Так же присутствует интеграция с популярными вэб-сервизами такими как: Яндес-маркет, WebMoney, DepositFiles, Youtube, PayPal.

К недостаткам системы можно отнести то, что ее код является закрытым, что ограничивает возможности разработчиков. И, к сожалению, данная система в последнее время очень популярная среди «серых сайтов».

Безопасность сайтов, построенных на основе CMS

По результатам исследования сайты, использующие, бесплатные CMS в среднем в четыре раза чаще подвергаются заражениями и попадают в черные списки, чем сайты на коммерческих CMS.

Для исследования были выбраны случайным образом 30 000 действующих сайтов в домене RU, созданные на различных CMS. Выборка сайтов с определением типов CMS была предоставлена компанией iTrack

В рамках исследования проводился анализ следующих характеристик сайтов.

- Версия CMS и веб-сервера.
- Наличие вирусов на сайте.
- Наличие сайта в черных списках Google и Yandex.
- Наличие сайта в черных списках Phishtank, DNSRBL, UCE PROTECT и других.
- Корреляция между версией CMS и наличием проблем на сайте.

На рис. 2.17 показан процент зараженных сайтов, реализованных на бесплатных CMS.

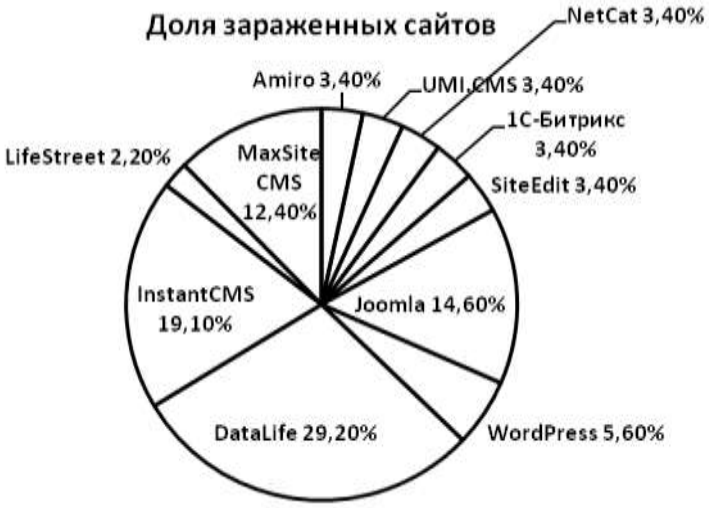


Рис. 2.17. Зараженные сайты

Как видно из рисунка, у бесплатных CMS ситуация с безопасностью сильно различается. Так, например, в черные списки попадает каждый двадцатый сайт на Datalife Engine. В то же время среди сайтов на CMS TYPO3 не было обнаружено ни одного зараженного сайта. Разработчики этой компании уделяют значительное внимание повышению безопасности своей CMS.

Среди наиболее распространенных в Рунете коммерческих CMS компания iTrack не выделяет явного лидера – все они имели примерно одинаковое количество сайтов с проблемами.

Наблюдается заметная корреляция между использованием наиболее свежих версий CMS и наличием проблем на сайтах, как среди коммерческих, так и среди бесплатных CMS. Своевременно обновляемые версии CMS снижают риск появления проблем в среднем в два раза. Показатели обновления версий CMS приведены на рис. 2.18.

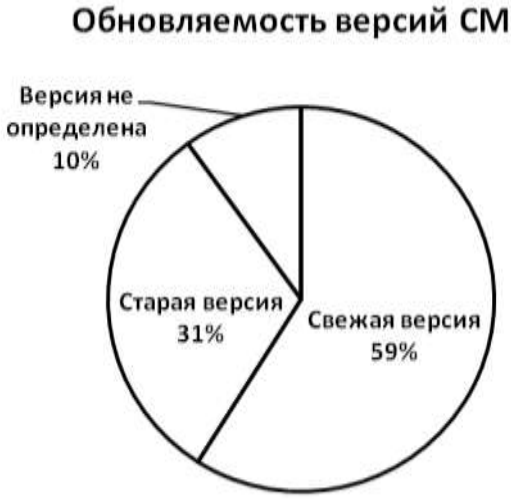


Рис. 2.18 Показатели обновляемости версий CMS.

Наиболее явно польза от перехода на свежие версии заметна на примере Joomla и WordPress.

Самую последнюю версию Joomla использовали только 3% сайтов, а самую последнюю версию WordPress использовали 15% сайтов. При этом доля проблемных сайтов на Joomla в три раза выше, чем на WordPress. Очевидно, что своевременное обновление версии CMS повышает безопасность сайта.

Одной из гипотез авторов исследования является то, что сайты на коммерческих CMS чаще разрабатываются силами профессиональных разработчиков, которые своевременно следят за обновлением версий, что приводит к большей доле зараженных проектов на бесплатных решениях.

Распределение проблемных сайтов по типам веб-серверов в целом совпадает с рейтингом популярности веб-серверов, в связи с чем можно говорить, что версия того или иного веб-сервера не является решающим фактором безопасности сайта.

Яндекс заносит сайты в черный список в два раза чаще, чем Google. При этом пересечение списков Google и Yandex составляет всего 10% от общего числа занесенных в черные списки сайтов.

Вывод: веб-мастерам, seo-оптимизатором и другим специалистам, которые отвечают за продвижение сайтов, следует обязательно отслеживать наличие сайта в черных списках обоих поисковиков.

Одной из ключевых проблем на рынке является низкая осведомленность владельцев сайтов о возникающих угрозах.

После завершения исследования была проведена попытка связаться с владельцами около 130 сайтов, которые были заражены вирусами или находились в черных списках поисковых систем. Более половины владельцев, чьи сайты активно используются для продвижения товара или услуги, не знали о наличии проблем с их сайтами. При этом треть из них продолжала тратить средства на обслуживание или продвижение сайта.

В дополнение к первой части исследования компания проверила все 30000 коммерческих сайтов на наличие в черных списках по признаку рассылки спама с IP-адреса, на котором размещен сайт.

Почти 15% всех проверенных сайтов находятся в одном из списков UCE PROTECT, то есть непосредственно участвуют в рассылке спама или находятся по соседству с сайтами, которые рассылают спам. При этом 70% сайтов находятся в одной подсети, а 28% — на одном IP адресе с сайтами – источниками спама.

2.6. Критерии выбора CMS

Приведем краткий список наиболее часто используемых критериев для выбора CMS.

- Соответствие CMS требованиям проекта.
- Стоимость CMS.
- Стоимость эксплуатации.
- Время разработки.
- Удобство пользования.
- Наличие технической поддержки (для платных).

Эти критерии могут помочь выбрать систему, но это далеко не полный их список, и сделать окончательный правильный выбор, руководствуясь только этим нельзя.

Требования, соответствие которым будет являться несомненным преимуществом системы, следующие.

-Управление правами доступа.

-Поиск содержимого.

-Аутентификация перед просмотром какого-либо содержимого.

-Простой курс обучения по работе с системой управления содержимым .

-Простой интерфейс администрирования системы управления содержимым.

-Соответствие ежемесячно возможным затратам на хостинг.

-Прозрачное ценообразование на внедрение, дополнения, дизайн.

-Наличие сети партнеров (возможность расторгнуть договор без необходимости заново создавать сайт).

-Возможность дорабатывать систему неспециалистом.

-Поддержка нескольких языков.

Так же нужно оценивать следующие технические параметры.

-Скорость работы и нагрузка на сервер, алгоритм предоставления кэшированных страниц.

-Наличие механизмов резервного копирования.

-Постоянные обновления безопасности.

-Наличие модулей поисковой оптимизации.

Усложняет выбор системы управления сайтом тот факт, что сегодня нет жестких факторов ценообразования – стоимость CMS нередко зависит от прихоти разработчика. Клиент может получить продукт сомнительного качества по завышенной цене.

Портал CMS Magazine провел опрос для исследования факторов, влияющих на выбор CMS. Посетители сайта расставили приоритеты следующим образом (табл. 2.1):

Таблица 2.1.

Факторы выбора	**Приоритеты**
Удобство эксплуатации	45%
Стоимость CMS	29%
Известность продукта	9%
Другие мотивы	7%
Совет веб-студии	6%
Рекомендация коллег	4%

Для большинства клиентов цена является если не главным, то одним из значимых факторов при покупке. Остановимся подробнее на рассмотрении этого аспекта при выборе программы.

Стоимость проекта

Оценивая стоимость проекта, следует учитывать как начальные инвестиции, так и стоимость его обслуживания.

Вот вопросы, на которые нужно ответить при выборе CMS.

1.Что из себя представляет CMS: «коробочный» продукт, индивидуальную разработку студии или команды специалистов, или open source систему?

2.Сколько стоят услуги подрядчиков по индивидуальной доработке CMS, внедрению новых модулей, разработке дизайна?

3.Каковы минимальные и максимальные требования к специалисту, которому предстоит обслуживать систему? Сколько стоят его услуги, какова его зарплата при работе в штате?

4.На каких серверах может размещаться CMS? Требования к хостингу.

Расходы, связанные с реализацией проекта внедрения CMS, можно разбить на несколько категорий - по видам расходов и их периодичности.

В табл. 2.2. перечислим основные расходы для коммерческих коробочных систем и возьмем для сравнения самые популярные CMS этого сегмента.

Таблица 2.2.

Расходы, связанные с внедрением и эксплуатацией CMS

Статьи затрат	**Начальные инвестиции**	**Эксплуатационные расходы**
CMS и дополнительное программное обеспечение	Стоимость лицензий на CMS и системное программное обеспечение, а также услуг по их настройке (доработке).	Стоимость обновлений и обращений в службу поддержки.

Оборудование	Стоимость ПК, серверов, ИБП, коммуникационного, офисного оборудования и т. д., а также затраты на их модернизацию, установку и настройку.	Стоимость обращений в службу поддержки, ремонта оборудования, приобретения расходных материалов и т. д., а также арендные/ лизинговые платежи.
Персонал	Стоимость обучения персонала.	Расходы на оплату работы IT-персонала, обслуживающего CMS. Увеличение зарплаты сотрудникам в связи с ростом их квалификации.
Организационные расходы	Стоимость работ по предпроектному анализу и управлению проектом. Стоимость изменений в организационной структуре, вызванных использованием CMS. Расходы, связанные с потерями в организационной эффективности на период внедрения CMS.	Аренда каналов широкополосного доступа в Интернет.

Примерные цены программных продуктов коммерческих фирм приведены в табл. 2.3.

Таблица 2.3.

CMS	Стоимость различных видов CMS				
	Визитка	Корпоративный сайт	Интернет-каталог (витрина)	Магазин	Портал максимальной сложности
1С-Битрикс	5 000	13 000	25 000	50 000	300 000
UMI.CMS	4 000	9 000	18 000	40 000	-
NET.CAT	6 000	15 000	21 000	21 000	40 000
ABO.CMS	9 000	18 000	24 000	30 000	-
Twilight	4 000	12 000	-	27 000	-
Amiro.CMS	4 000	12 000	16 000	27 000	-
HostCMS	6 000	-	-	30 000	30 000

Как видно цены на продукты одного вида не сильно отличаются. Выделяется 1С-Битрикс, но многофункциональность и интегрированность с бизнес-процессами предприятия, делают эту систему предпочтительной для масштабных проектов.

Однако большую часть затрат составляют услуги по дизайну и продвижению сайта, а стоимость CMS в годовой стоимости проекта составляет 20-30%. Исключение составляет ABO.CMS. Разработка ее дизайна выполняется быстрее всего, но при этом сама система значительно дороже аналогов. В стоимость дизайна входят так же раскрутка и продвижение.

Затраты на эксплуатацию

После покупки системы, создания сайта, и покупки хостинга, возникают проблемы обновления системы, обновления контента, и использования технической поддержки.

Как правило, техническая поддержка бесплатна только первый год, и чтобы продолжать ее получать в дальнейшем придется заплатить, правда эти суммы ни один издатель не раскрывает.

Система постоянно обновляется, появляются новые модули, совершенствуется код, устраняются «дыры» в безопасности. Как правило, после первого года бесплатными остаются только обновления безопасности, а все остальное становится платным. Стоимость обновлений составляет 20-30% от первоначальной стоимости системы.

CMS системы тем и хороши, что обновлять содержимое может любой неспециалист после небольшого знакомства с системой. Если же возникают проблемы, то все серьезные издатели готовы обучить сотрудников на курсах.

Разные системы написаны на разных языках. Как правило, это связка PHP+MySQL, но бывают и исключения. Если система основана на продуктах Microsoft (например, на .NET) то стоимость хостинга будет отличаться от обычного Unix-хостинга в 2-3 раза.

Методы администрирования и статистического анализа деятельности предприятия на базе CMS

Одна из основных задач программного обеспечения администрирования сайта заключается в возможности редактирования информации на готовом веб-сайте. В инструментарии администрирования также могут реализовываться и дополнительные сервисы, направленные на получение статистики и ее анализ, возможность работать с хостинг-аккаунтом, настройкой прав пользователей веб-сайта и администраторов.

Зона администрирования часто реализована как отдельный сайт со своей навигацией, дизайном и наполнением, что значительно упрощает работу с управляемыми информационными ресурсами.

Для реализации администрирования создается база данных, и подготавливается шаблон для автоматического разворачивания базы на других интернет ресурсах.

Основными таблицами базы данных являются следующие.

- Администраторы – таблица с учетными записями пользователей панели администрирования.
- Модули – в этой таблице хранится список модулей со всеми параметрами каждого модуля.
- Доступ – таблица доступа к модулям и конкретным возможностям модулей.
- Новости, блоки и т. д.

Для каждого модуля реализовано несколько таблиц обеспечивающих полную автономность модуля.

Для защиты от несанкционированного входа применяется аутентификация с помощью технологии сессий.

Все данные хранятся на сервере и возможность несанкционированного входа снижается. Смена ключа сессии происходит через определенные интервалы времени, что делает невозможным длительное использование системы при перехвате ключа.

Инструментарий администрирования выглядит в виде набора специализированных модулей. Каждый модуль выполняет определенную задачу и состоит из нескольких страниц.

Для добавления (удаления) модуля достаточно добавить (удалить) несколько файлов в общую директорию и добавить (удалить) одну запись в базу данных. Реализована возможность автоматического подключения новых модулей.

Безопасность

Как показали исследования компании iTrack многие владельцы сайтов, построенных на CMS, не задумываются над этим вопросом и часто не подозревают о зараженности своих электронных предприятий. Поэтому важно представлять механизмы угрозы безопасности сайтов и способы борьбы с ними.

Вне зависимости от того, что толкает хакеров на деструктивные действия, их арсенал вполне традиционен. Одним из самых популярных методов взлома считаются SQL-инъекции. Проблема SQL-инъекций весьма реальна и довольно серьезна. В любой системе есть код и данные. Эволюция информационных систем разделила два этих понятия, и сегодня мы видим, что данные находятся в отдельных хранилищах, называемых базами данных. Что касается веб-систем, то чаще всего наибольшую ценность представляют собой данные, хранимые в них. SQL-инъекции нацелены как раз на то, чтобы атаковать веб-системы и получить доступ к системе управления базами данных (СУБД).

С помощью SQL-инъекций злоумышленник может копировать все данные, хранимые в СУБД, получать неавторизованный доступ к системе, выводить систему из строя, уничтожать данные СУБД и, в некоторых случаях, получать полный контроль над сервером.

Лучшей защитой от SQL-инъекций является аккуратная проверка тех частей кода, которые обрабатывают данные поступающего на сервер HTTP запроса и подготавливают их для включения в качестве параметров SQL-запроса, отсылаемого далее СУБД.

Понимая, что на разработчика не всегда можно положиться, создатели современных языков программирования (например, самый популярный в сети - PHP) изобретают встроенные защиты от SQL инъекций. Так, например, если включить в конфигурационном файле PHP режим интерпретатора magic_quotes, использование SQL-инъекции становится более сложной задачей и значительно сужает область уязвимого кода.

Классические методы защиты от SQL-инъекций используются на стадии разработки продукта. Первое, что должны сделать разработчики – обезопасить данные пользователя системы, то есть проверять ВСЕ данные, приходящие от пользователя системы, а также данные, передающиеся различными способами с клиентской системы – cookies, скрытые параметры, POST запросы и т.п.

Есть типовые варианты борьбы с инъекциями вредоносного кода методом отсечения (экранирования) кавычек из запроса - стандартная функция mysql_escape_string и ей подобные. Но, это не всё, так как система будет работать в разных условиях, на разных платформах, и использование типовой функции на некоторых из них будет просто невозможно, а соответственно приведет к уязвимости всей системы в целом.

Более грамотный подход, предполагает разработку «оберток», когда параметры запроса обрабатываются специальными методами. Существует немало моментов, которые необходимо предусмотреть при проектировании и разработке «оберточных» функций системы.

Вторым этапом является правильная и безопасная интеграция системы на платформу и БД. На этом этапе необходимо отключить функции, которые могут помочь атакующему правильно сконфигурировать периметр сети, системы IDS и многие другие необходимые мероприятия.

На заключительном этапе необходимо сделать независимый аудит безопасности сконфигурированной системы, это позволит выявить и своевременно устранить недочеты безопасности, возникшие в ходе интеграции.

Еще одно решение этой проблемы: перед передачей внешних данных в запрос экранировать недопустимые символы.

Кроме SQL-инъекций есть и другие методы популярных атак на уязвимые места веб-сайтов – PHP-including, PHP- инъекции, XSS и другие.

Все они имеют разные уровни опасности и разделены на две группы:

-уязвимости с кодом, исполняемым на стороне клиента, и

-уязвимости с кодом, исполняемым на стороне сервера.

С точки зрения владельцев сайтов, наибольшую опасность представляют уязвимости находящиеся на стороне сервера, поскольку проникновение злоумышленника на сервер может означать полное уничтожение сайта или критичную потерю конфиденциальных данных.

Несмотря на то, что инструментарий хакеров давно известен, постоянно появляются все новые способы взлома, не выходящие за рамки вышеописанных слабых мест. И порой разработчики CMS проигрывают в этой гонке, не успевая выпускать очередные заплатки для своих систем.

Возможности CMS для поисковой оптимизации сайтов

В последнее время разработчики CMS все чаще стали говорить о возможностях своих продуктов в области SEO-оптимизации.

На возможности своей CMS в области поисковой оптимизации обращают внимание разработчики почти всех платных систем. Как правило, системы здесь имеют специальный модуль, помогающий создавать оптимизированные тексты под поисковые системы. В CMS также имеется собственная система обмена ссылками, механизм задания внутренних и внешних ссылок и инструменты для контроля за местами сайта в поисковой выдаче по ключевым запросам. В бесплатных CMS также можно найти SEO модуль.

При оптимизации сайтов важно учитывать требования оптимизаторов. При перечислении необходимых SEO-возможностей CMS чаще всего называют следующие характеристики:

- «правильные» заголовки, выделяемые тегами;
- отсутствие дублирующих ссылок, ведущих на одну и ту же страницу;
- «правильный» html-код;
- возможность присваивать каждой странице собственные мета-теги;

Модуль SEO-оптимизации должен предоставлять возможность гибкого управления важными для SEO тегами, title, мета-теги и пр. Также важным является анализ статистики посещений и поведения на сайте, анализ основных показателей в поисковых машинах, автоматизацию покупки-продажи ссылок, отслеживание изменений позиций сайта в поисковиках.

Сегодня на рынке есть много решений, которые поддерживают эти возможности независимо от CMS, установленной на сайте. Зачастую, владельцы сайтов мало руководствуются SEO возможностями той или иной CMS, а в первую очередь, обращают внимание на её простоту и функциональность, и лишь поверхностно смотря на её SEO функции, которые обычно сводятся к возможности редактировать общий для всех страниц титул, прописывать ключевые слова и статистике посещений.

Однако частого этого не достаточно, да и не все функции на самом деле полезны на практике. Есть множество факторов, которые усложняют выбор системы, особенно, если сайт представляет собой сложный проект, например, интернет-магазин или сайт компании с каталогом товаров и услуг, для которых важно, чтобы на их сайт приходили потенциальные покупатели. Наличие SEO функций должно рассматриваться на этапе анализа функциональности системы наряду с остальными функциями.

Из общих требований к SEO для CMS можно назвать обеспечение постоянной индексации сайта (встречаются системы, для которых это проблема) и высокую скорость генерации страниц.

Всегда возникает желание сравнить возможности коммерческих и бесплатных решений, которых на сегодняшний день предостаточно. Однозначной зависимости SEO-инструментов и стоимости системы нет, скорее, есть зависимость от «национальности» CMS. Те продукты, которые создаются в России для отечественных пользователей, должны быть более

дружелюбными к отечественным поисковым системам, биржам ссылок, интеграции с офлайновыми инструментами оптимизаторов.

Но чаще всего SEO-функции в бесплатных системах либо полностью отсутствуют, либо находятся на примитивном уровне («редактирование титулов» и «мета-тегов»).

В платных CMS, как правило, предоставляется полный комплекс услуг по продвижению сайта.

Часть 3. WEB-КОНТЕНТ ТИПОВЫХ ЭЛЕКТРОННЫХ ПРЕДПРИЯТИЙ

3.1. Типовая контент-модель электронного предприятия B2C

Типичным представителем сегмента B2C является Интернет-магазин. Это электронная торговая точка, предоставляющая возможность в он-лайновом режиме и в рамках имеющегося ассортимента осуществлять покупку нужных товаров. Как и обычный магазин, электронный магазин имеет торговый зал – электронную витрину, а также всю необходимую бизнес-инфраструктуру для управления процессом электронной торговли через Интернет.

Электронный магазин должен обеспечить решение следующих задач:

- публикация каталога товаров;
- формирование заявки клиента на приобретение товара;
- формирование заявки на получение товара со склада;
- сопровождение процесса доставки товара клиенту;
- получение денег за товар.

В традиционной схеме электронной покупки участвуют: покупатель (формирующий заказ), электронный магазин, расчетный центр (через который ведутся электронные платежные расчеты), организация, обеспечивающая доставку товаров клиенту со склада.

Потенциальный покупатель при помощи браузера осуществляет доступ к информационному серверу магазина. На сервере размещена электронная витрина, где представлены каталог товаров с возможностью поиска по запросу и необходимые средства для ввода регистрационной информации, заполнения бланка заказа, возможно, проведение платежей и оформления доставки.

Регистрация покупателя обычно производится при оформлении заказа. Выбрав товар, клиент заполняет бланк, в котором указывается, как будет осуществлена оплата и доставка.

После составления заказа и регистрации вся собранная информация о покупателе поступает из электронной витрины в торговую систему магазина. В торговой системе проверяется наличие товара на складе, инициируется запрос к платежной системе. При отсутствии товара на складе направляется запрос поставщику, а покупателю сообщается время реализации заказа.

Из выше сказанного, в самом общем случае структура электронного магазина состоит из следующих элементов:

Титульная страница

Это первая страница сайта и «лицо» электронного магазина, куда автоматически попадает пользователь при наборе URL-адреса в браузере. Ее главная задача как можно быстрее справиться с наплывом посетителей, быстро направив покупателя в нужный раздел.

Этому способствует целый набор средств навигации для быстрого перемещения по каталогу товаров. К элементам навигации относятся: меню, карта сайта, выделенные гиперссылки, «облако тегов».

Система регистрации

Функция данной системы провести регистрацию посетителей, если они впервые посещают сайт, провести аутентификацию повторных посетителей, посредством ввода своего логического имени (идентификатора) и пароля, полученных при предыдущем посещении. В целях появления повторных клиентов возможна реализация для них дополнительных услуг (рассылки, скидки и т.п.). Процесс регистрации может инициироваться системой до или после выбора товаров. Ввод регистрационных данных в самом начале имеет больше преимуществ, так как позволяет легко отслеживать предпочтения конкретного посетителя и, учитывать их при повторном посещении, тем самым, повышая качество обслуживания (например, выводить сразу ту часть каталога, которая ранее заинтересовала посетителя). Однако регистрация после выбора товара позволяет покупателю сохранить анонимность и экономит время.

Технологии идентификации посетителей:

- базовая идентификация (регистрация на сайте);
- файлы-cookie (хранятся на web-сервере);
- доменное имя;
- IP-адрес;
- персональные адреса (у каждого пользователя – свой адрес);
- строгая аутентификация (цифровые сертификаты, смарт-карты).

Каталог и поисковая система

Включает в себя многоуровневый поиск по различным критериям (вид товара, наименование, страна-производитель, технические характеристики, цена и др.). Специалисты IBM утверждают – оптимально, чтобы пользователь находил любую информацию на сайте в три нажатия мышкой. Иначе у многих не хватит терпения или они «заблудятся» (правило «8 секунд» – если покупатель не нашел то, что искал в течение восьми секунд, то он перейдет на другой сайт). Около 77% клиентов пользуются при покупке функцией поиска, а 43% считают ее самой важной.

Получив первое положительное впечатление от пребывания в магазине, покупатель начинает «прохаживаться вдоль его прилавков». Для этого

большинство предприятий предлагают один сценарий – навигацию по каталогу товаров.

Интернет-каталоги – это разветвленные системы, они разбиты на разделы (типы товаров), подразделы и т. д. Двигаясь по одной ветви каталога, покупатель выбирает нужный ему товар, а затем переходит к другой его ветви, к другому типу товаров. При этом ему часто приходится возвращаться к началу каталога, поскольку он не видит его целиком. Это очень неудобно. Дело в том, что в обычном магазине, когда вы стоите у полок с одним товаром, вы одновременно видите, где расположены другие товары. Дружественный интерфейс магазина позволяет видеть общий план всего каталога, а не только перемещаться по нему вверх или вниз. В этом случае покупатель может легко перейти совершенно к другой ветви каталога.

Для больших каталогов очень важны возможности индексации товаров и их поиска. Анализ статистики покупок в действующих электронных магазинах показал, что 50% заказов в них было сделано сразу же после поиска по каталогу. Столь большой процент покупок после осуществления поиска вызван на самом деле не тем, что покупатель точно знает свои потребности, а тем, что подавляющее большинство магазинов (и в первую очередь российских) представляет собой именно web-каталог с возможностью поиска. Покупатель с удовольствием «прогулялся» бы по интернет-магазину (полистал бы книжки, послушал диски и т. д.), но ему просто не предоставляют этой возможности, буквально насильно загоняя в поисковую систему.

Основную задачу, которую должны ставить и решать создатели ЭП, прорабатывая структуру каталога, можно сформулировать следующим образом: сделать максимально дружественную систему навигации с широкой информационной поддержкой. Это необходимо для закрепления и поддержания того положительного впечатления о вашем предприятии, которое возникает у покупателя, когда он входит в него.

По статистике только 46% электронных магазинов имеют логично построенный каталог товаров с интуитивно понятной навигацией, а 26% покупателей не совершает покупок из-за сложности навигации на сайте.

Описание товара

В обычном магазине покупатель или берет товар в руки и знакомится с ним, или начинает задавать различные вопросы о товаре продавцам. В электронном магазине такое невозможно, но покупателю также хочется посмотреть на технические параметры товара, почитать документацию, убедиться в наличии (или в отсутствии) в нем той или иной функции. Все это необходимо предложить ему в виде различного рода описаний товаров и советов по их выбору. Подобного рода дополнительная информация может храниться вне каталога, но она обязательно должна присутствовать. По своему объему информация такого рода часто может превосходить каталог,

поэтому лучше всего размещать ее в отдельном структурированном пространстве.

При создании раздела с описанием товаров/услуг рекомендуется пользоваться принципом интерактивной обработки контента. Сущность принципа заключается в особой организации доступа пользователей к информации, размещаемой на сайте. В соответствии с ним пользователи просматривают не всю информацию о товаре или услуге, а только ту ее часть, которая действительно необходима им для работы и принятия решений. Доступ к информации, размещенной на сайте, осуществляется по следующей схеме (рис. 3.1).

Уровень идентификации. Ограничивается просмотром только наименований и стоимостных характеристик товаров (обычно эта информация пользуются клиентами для поиска и идентификации требуемого товара в каталоге, а также постоянными клиентами для заказа хорошо знакомых товаров).

Уровень ознакомления. Если предыдущей информации клиенту не достаточно для осуществления выбора, то он может подробнее ознакомиться с основными характеристиками товара, его изображением и системой скидок, действующей при покупке товара (обычно этой информацией пользуются клиенты, желающие оформить заказ на товары приобретаемые вновь).

Уровень изучения. Если пользователь желает подробнее ознакомиться с особенностями товара, то он может выборочно загрузить на свой компьютер файлы с подробными описаниями, просмотреть рисунки, видеоизображения и другие мультимедиа данные, сопровождающие товар.

Таким образом, для некоторых категорий товаров (книги, компакт-диски и т.п.) вполне достаточно лишь подробного текстового описания, тогда как для более дорогих товаров (бытовая техника, электронная аппаратура и т.п.) необходимо наличие видео, аудио и другой мультимедийной информации. В то же самое время обилие мультимедийной информации раздражает пользователей с низкой технической оснащенностью, которые, не желая ждать долгой загрузки картинок или видео изображений, покидают сайт.

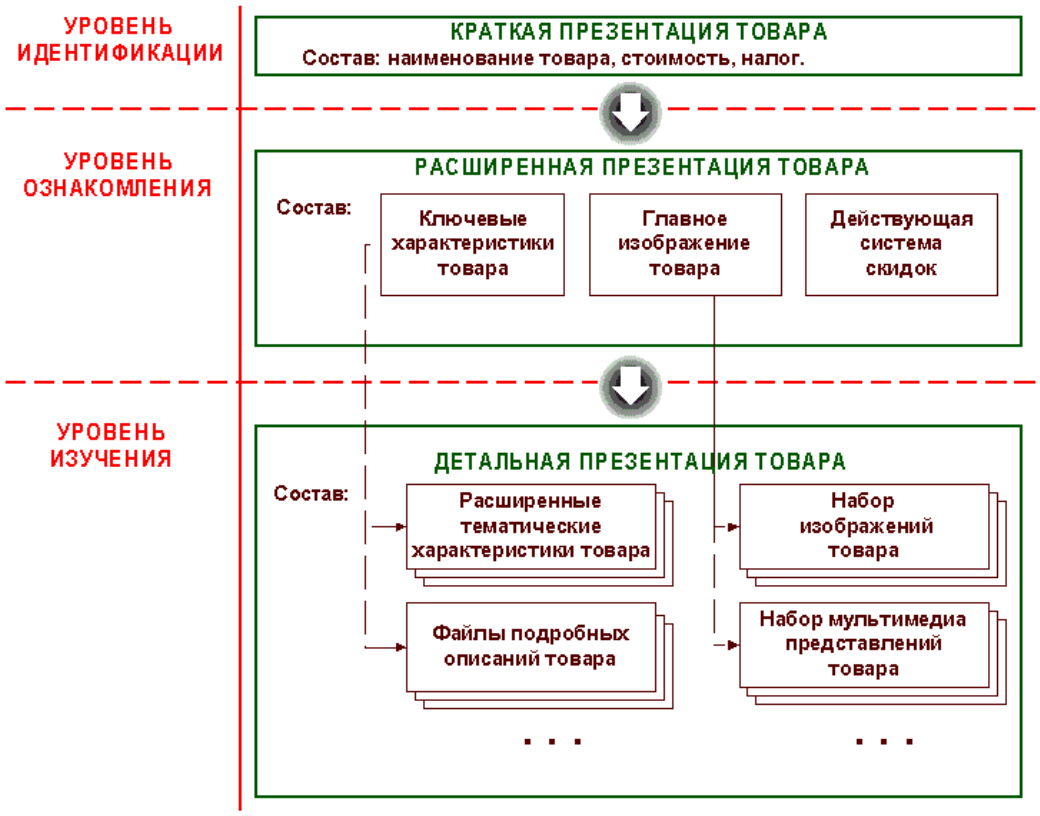


Рис. 3.1. Уровни интерактивной обработки контента

По статистике 56,8% электронных магазинов публикуют сопроводительную информацию по товарам и 43,2% ограничиваются лишь наименованием и ценой, и только 32,4% предлагают дополнительные материалы (ссылки на производителей, рецензии, статьи и т.п.). Около 25% потенциальных покупателей не совершают покупки именно из-за недостатка информации о товаре. Согласно опросу 44% американских Интернет-покупателей считают, что наличие подробного крупного изображения товара увеличивает вероятность совершения покупки.

Формирование корзины

Во всех электронных магазинах выбор товаров происходит по традиционной форме - через складывание товара в виртуальную торговую тележку или корзину (shopping cart). Понравившиеся товары покупатель складывает в корзину, которую он может видеть в любой момент, он так же может добавлять в нее новые товары или изменять количество уже находящихся в корзине товаров.

В корзине отражаются выбранные покупателем товары, стоимость покупки, дополнительные сервисные услуги (необходимость в упаковке товара и т.п.). Предоставляется возможность изменить содержимое корзины (добавить новый товар или удалить один из выбранных, отложить выбранный товар на определенный срок и т.д.).

Если покупатель ранее уже совершал покупки в этом магазине и зарегистрировался, то желательно, чтобы он видел свои предыдущие корзины, даже те, которые не оплачивал, а просто формировал ранее.

Система оплаты

На данном этапе посетитель выбирает одну из предложенных схем оплаты. В случае использования электронных платежных систем данный раздел должен быть связан с процессинговым центром платежной системы.

Система доставки

Здесь потребитель выбирает один из предложенных видов доставки. В случае торговли электронным контентом (программное обеспечение, отчеты аналитических исследований, электронная литература и т.п.) непосредственно по сети Интернет.

Подтверждение заказа

На этом этапе клиент подтверждает свое желание совершить покупку в соответствии с ранее сделанным выбором. Посетитель, прошедший этот этап, считается покупателем.

Система поддержки клиентов

В процессе обслуживания клиента могут возникать различные нестандартные ситуации (задержка доставки, повреждения товара, брак, сбои в прохождении платежей и т.д.). Решением этих проблем занимается система поддержки клиентов. При нормальном исполнении заказа ее работа сводится к отображению на соответствующей web-странице, доступной клиенту после авторизации (разрешения на доступ), этапов исполнения заказа и его текущего состояния («отгружен», «оплачен», «ожидание поступления платежа» и т.п.).

Кроме того, необходимо предусмотреть средства онлайновой поддержки клиента. Для этого можно использовать интернет-пейджеры (например, ICQ), интернет-телефонию (Skype) или специальные программы веб-мессенджеры, выполняющие функцию онлайн-консультантов.

Дополнительные разделы

информация о компании;

раздел для партнеров;

раздел для общения с клиентами;

разделы конкурсов, розыгрышей, призов и рекламных акций.

Интернет-мерчандайзинг

Мерчандайзинг – современная технология разработки комплекса мер продвижения товаров и торговых марок в розничной торговле, используемая крупными предприятиями розничной торговли. Мерчандайзинг призван определять набор продаваемых в магазине товаров, способы выкладки товаров, снабжение их рекламными материалами, цены и т.п.

Те же задачи ставятся и перед интернет-магазином. Так же, как и в off-line-торговле, владелец может действовать исходя из некоторых уже заданных параметров (готового сайта или арендованной под магазин площади), а может начинать со свободного творчества и самостоятельно формировать задание по архитектуре Интернет-ресурса. При этом вмешательство специалистов по мерчандайзингу будет тем эффективнее, чем на более ранней стадии находится процесс разработки сайта.

В основном, архитектура хорошего Интернет-магазина напоминает архитектуру реальной торговой площади и состоит из: главной страницы (витрины), количества функциональных разделов (количество служебных помещений или торговых площадей), связей и путей взаимодействия разделов (пути и методы взаимодействия между служебными, складскими и торговыми помещениями), структуры back-office-систем хранения и управления информацией (складские, финансовые, маркетинговые подразделения) и т.д. Благодаря этой метафоре, специалисту по мерчандайзингу станет очевидно, что и приемы и методы работы с Интернет-ресурсом продавца сходны с методами работы с реальными торговыми площадями.

Главная страница

При оформлении витрины необходимо выполнять слудующе требования:

- *Информирование о скидках.* Информация о скидках является не менее действенной в Интернет, чем в реальном магазине. Кроме того, при помощи несложных специальных средств Вы можете заставить пользователя оставить краткую информацию о себе, и это не очень отяготит его. А взамен - дать ему именной дисконт, снабженный уникальным номером, что обычно бывает очень полезным средством найти индивидуальный подход к покупателю.

- *Индивидуальный дизайн.* Индивидуальный дизайн Интернет-магазина - вещь гораздо менее затратная, нежели оформление витрины реальной торговой точки, и уж точно на это стоит потратить некоторые деньги.

- *Актуальные шаблоны сайта.* Рекомендуется создать несколько вариантов дизайна сайта для наиболее крупных праздников.

- *Правило «first in - first out».* Покажите как можно ближе залежавшийся товар, сделайте его интересным, и Вы разгрузите склад. Если на самой посещаемой странице сайта появится информация о залежавшемся товаре (хотя бы в виде анонса или баннера), то Ваш шанс избавиться от излишков на складе повышается.

Все это в равной степени работает и в реальном, и в Интернет-магазине. Но если наличие карты отделов магазина или указатели маршрута на входе в реальный магазин полезная, но не всегда нужная и оправданная инновация, то навигационная панель в Интернет-магазине - вещь абсолютно необходимая. Навигация главной страницы может и должна отличаться от навигации других страниц. Кроме стандартного "меню сайта", можно добавить и различные дополнительные средства - баннерные или текстовые блоки с информацией о свежих поступлениях, товарах со сниженной ценой, хитах продаж, сопутствующих и сервисных услугах и т.д.

Маршрут

Маршрут движения пользователя по сайту, так же как и движение покупателя по магазину, обусловлен некоторым рядом причин: сила привычки, случайность выбора товара, наличие указателей и направляющих

и некоторыми другими. Выяснив, какие страницы и разделы сайта являются наиболее посещаемые, можно спланировать размещение на таких страницах своих рекламных материалов. Необходимо сделать так, чтобы путь к самым выигрышным и коммерчески привлекательным предложениям вел с любой страницы сайта. Сделать это можно, размещая на каждой странице баннер или текстовую ссылку, ведущую именно к таким разделам.

Точно так же, как и в реальном магазине, для привлечения внимания клиента Вы можете использовать эффекты контраста, цвета, движения и даже звука.

Цвет и контраст

В реальной торговле обычно говорят о цвете и свете, но в Интернет свет скорее стоит заменить контрастом. Выделение значимой информации из окружающей среды, конечно, обратит на себя внимание пользователя. Информация, которую особенно важно донести до пользователя, должна максимально отличаться от окружающей цветовой гаммой или оттенками.

Движение и звук

Движение и звук являются дополнительными средствами привлечения клиентов в наименее посещаемые места.

В Интернет звук используется пока очень редко. В большинстве же тех случаев, когда звук используется, он выполняет декоративные функции. Коммерческим целям звуковая информация служит обычно только на сайтах, занимающихся продажей мультимедийных товаров (дисков с музыкой или кино, мелодий для мобильных телефонов и т.п.), где короткие демо-ролики позволяют оценить предлагаемый товар. Тем не менее, с постоянным развитием и совершенствованием качества связи, ее удешевлением и ростом аудитории пользователей с хорошим каналом доступа в Интернет, можно ожидать и увеличение роли мультимедийной информации в качестве средств рекламы и мерчандайзинга.

Движение в виде анимированных баннеров и flash-роликов используется довольно давно и продуктивно. Именно анимированные рекламные модули - баннеры - считаются наиболее кликабельными, то есть привлекают большую аудиторию к рекламируемым ресурсам. Этим фактом можно пользоваться как для рекламы своих товаров и услуг на других Интернет-ресурсах, так и на собственном сайте для привлечения посетителя к наиболее интересным с коммерческой стороны разделам.

Склад и наличие товара

Информация об отсутствии товара на складе должна выясняться не во время обращения клиента, но еще на стадии выбора клиентом предложения. Современные системы автоматизации и интеграции складской программы и Интернет-сайта решат эту проблему.

Необходимо организовать эффективный обмен актуальной информацией о наличии товара на складе производителя или формировать

достаточный складской запас на площадях продавца, при наличии этих площадей, конечно.

Каталог

Один из основных плюсов Интернет-магазина в сравнении с реальной торговой точкой - относительная дешевизна "аренды" при возможностях почти бесконечно большого ассортимента. В связи с этим Интернет-магазины обычно стремятся расширить максимально свой ассортимент за счет привлечения новых марок. Часто так же бывает ситуация, когда ресурс, начинавший с довольно узкой специализации, постепенно расширяет ее, а с ней и ассортимент, естественно. В такой ситуации товару конкретного производителя очень просто потеряться.

В Интернет-магазине, конечно, нет полок, но, тем не менее, существует некая пространственная структура, а именно - рубрикация каталога, представляемая в виде иерархии рубрик и подрубрик. Логика построения этой иерархии может различаться, но в большинстве случаев сводится к двум основным типам: первый – когда отправной точкой, вершиной иерархии является распределение по типам товаров, здесь однотипные товары разных марок находятся в равноправных условиях; второй - когда распределение осуществляется по торговым маркам.

"*Расположению на уровне глаз*" в Интернет может соответствовать положение Вашего товара в первом же экране браузера, до того, как пользователь начнет пользоваться прокруткой.

Зоны импульсных покупок. В Интернет нет касс, зато, есть корзины и формы заказов. Интернет-магазин, при помощи специальных программных инструментов, может "напоминать" пользователям о сопутствующих товарах, исходя из самого характера заказа.

Ценники и спецификации

Интернет – очень оперативная среда. В реальном магазине потенциальный покупатель может сделать покупку не сегодня, а завтра, когда выяснит, что из всех ближайших конкурентов - в данной точке самая привлекательная цена или самый удобный сервис, но для этого ему потребуется обойти все магазины. Интернет пользователь получает эту информацию гораздо быстрее. Поэтому основные моменты способные повлиять на решение покупателя должны быть видны с первого взгляда.

На данном этапе самой важной информацией для пользователя будет:

- цена,
- способы оплаты,
- виды и сроки доставки.

Навигация

Вообще любые средства навигации по сути можно отнести к инструментам Интернет-мерчандайзинга. Среди основных инструментов можно выделить:

- Поисковая форма (поиск по сайту),

• Карта сайта,

• Внутренняя баннерная система - аналог рекламе на месте продажи в фирменном магазине (стикерам, флайерам и т.д.),

• Облако тегов – форма визуализации данных, представляет собой множество ключевых слов, начертанных разными размерами шрифта. Чем крупнее шрифт, тем чаще ключевое слово употребляется в контексте облака.

3.2. Типовая контент-модель корпоративного портала

Основные структурные элементы корпоративного портала:
- главная страница;
- система навигации по сайту;
- информация о компании;
- информация о продукции, товарах и услугах;
- прайс-лист;
- информационный раздел (общая информация);
- система регистрации и авторизации;
- раздел для партнеров;
- система online-заказа и его обработки;
- новости компании;
- раздел для общения с клиентами;
- разделы конкурсов, розыгрышей призов и рекламных акций.

Принципы построения некоторых из перечисленных структурных элементов (таких, как раздел для партнеров и система регистрации и авторизации) мало отличаются от принципов построения соответствующих разделов Интернет-магазина. На других остановимся подробнее.

Главная страница

Корпоративный сайт выполняет разнообразные функции. В отличие от Интернет-магазина, цели которого сгруппированы вокруг онлайн-продаж, корпоративный сайт решает более широкий круг задач. Одна из них – повышение узнаваемости и известности логотипа и элементов фирменного стиля компании, используемых не только в Интернете, но и на упаковке продукции, в печатных рекламных материалах, в традиционной рекламе и т. д. Поэтому оформление главной страницы сайта компании обычно осуществляется с использованием ее фирменного стиля и фирменных цветов. Часто на видном месте располагается слоган, используемый фирмой в рекламных кампаниях.

Поскольку корпоративный сайт поддерживает основной бизнес, на него "переезжает" информация о рекламных акциях, лотереях, розыгрышах призов, презентациях новых продуктов и других мероприятиях, проводимых компанией в рамках ее обычного бизнеса. Вся эта информация сжимается до кратких анонсов и заголовков новостей и размещается на главной странице сайта, чтобы посетитель сразу увидел объявления обо всех главных событиях и мероприятиях, проводимых компанией.

Информация о компании

Корпоративный сайт не является самодостаточной единицей. Его существование возможно только благодаря основной компании и только для нее. Поэтому раздел "Информация о компании" на корпоративном сайте фирмы обычно содержит весьма подробную информацию. Как правило, он состоит из следующих подразделов:

- Контактная информация.
- Как нас найти?
- История фирмы.
- Лицензии, патенты, разрешительные документы.
- Публичные финансовые документы.
- Информация о продукции, товарах и услугах.

Информационный раздел

Если главная страница сайта должна завладеть вниманием посетителя, то информационные разделы должны создать у него впечатление, что он нашел то, что искал, не зря потратил время, и что ему стоит сюда заходить почаще.

Структурировать информацию можно в соответствии с различными принципами:

по времени ее поступления (удобно для новостей и календаря предстоящих событий в какой-либо области);

по типу аудитории, для которой она предназначена или которой она может быть интересна;

по типу самой информации (новости, аналитика, и т. д.);

по степени важности ("срочные новости", "горящие путевки" и прочее).

Объем предлагаемой информации тоже играет важную роль. Необходимо помнить, что клиенты будут посещать сайт регулярно только тогда, когда информация на нем постоянно обновляется. Можно сделать сайт сколь угодно качественным и интересным, но если его контент будет статичным, пользователи будут посещать его всего несколько раз и навсегда уходить. Таким образом, компания может размещать на сайте только тот объем информации, на поддержание которого в актуальном состоянии у нее хватит сил и средств.

Весьма интересной и привлекательной для клиентов является возможность формирования контента непосредственно под нужды данного пользователя. К примеру, проект специализируется на аналитической информации по электронной коммерции и публикует на своем сайте материалы по этой теме. При этом конкретного клиента может интересовать далеко не вся информация, содержащаяся на сайте, а, скажем, только материалы о московских Интернет-магазинах, торгующих продуктами питания. В такой ситуации клиенту может быть предложено зарегистрироваться на сайте, и настроить интерфейс под свои нужды, указав какие материалы он хотел бы видеть. Для этого клиент должен выбрать

соответствующую категорию, ключевые слова и другие параметры. Теперь, в любое время, когда бы данный пользователь ни зашел на сайт, его вниманию в первую очередь будут предложены именно те материалы, которые его интересуют.

Безусловно, это не ограничивает возможности клиента по ознакомлению с материалами других разделов, но статьи, соответствующие его выбору, будут показываться ему в первую очередь. Естественно, клиент в любое время может изменить настройки интерфейса.

Иногда на корпоративных сайтах создаются разделы, содержащие информацию, не имеющую прямого отношения к компании-владельцу сайта и ее продукции. Это сведения об общем состоянии дел в отрасли, просто интересная информация, которая может быть полезна клиентам.

В информационном разделе можно привести интервью с соответствующими специалистами, привести рекомендации по выбору товаров и услуг и т. д. Делается это для того, чтобы привлечь посетителей на сайт компании и акцентировать их внимание на проблеме, которую решает продукция фирмы, не рекламируя ее прямо.

Новости компании

В динамично развивающейся компании все время происходит что-то новое. Захват новых рынков, привлечение новых и удержание старых клиентов – одни из основных задач фирмы. Инструментами для решения этих задач являются открытие филиалов, новых магазинов и торговых точек, изменение цен, проведение распродаж, предоставление скидок и др. Обо всех этих мероприятиях необходимо известить как можно более широкий круг потенциальных покупателей.

Чем чаще появляются свежие новости, тем лучше. Заходя на сайт, посетитель должен видеть, что сайтом занимаются, что он обновлялся совсем недавно, поскольку последняя новость датирована вчерашним или даже сегодняшним числом. Кроме того, он видит, что в компании постоянно происходит что-то положительное, идет бурная деятельность.

Это тоже благоприятствует формированию положительного имиджа. Хорошей новостью будет объявление о праздновании дня рождения фирмы, открытии филиала в таком-то городе, презентации нового продукта и т. д.

Раздел для общения с клиентами

Привлечение клиентов и просто посетителей к участию в жизни компании – удобный инструмент маркетинга. Когда значительное количество людей, уже купивших продукцию фирмы, удалось вовлечь в дискуссию, новых посетителей становится легче мотивировать к покупке, поскольку они могут убедиться, что очень многие люди уже совершили ее и довольны результатом.

Для организации общения компании с клиентами и клиентов друг с другом на сайте обычно создается форум или книга отзывов (специальный раздел, где любой посетитель может оставить свое сообщение или отзыв о

продукции, ответить на уже существующее мнение и поучаствовать в обсуждении). Администрация сайта обычно принимает участие в дискуссии, отвечает на адресованные ей прямые вопросы и комментирует высказывания клиентов.

Конкурсы, розыгрыши призов и рекламные акции

Обычно конкурсы и розыгрыши призов, организуемые компаниями в рамках рекламных акций, проводятся среди покупателей ее продукции и широко анонсируются посредством рекламы. Естественно, что сайт не может находиться в стороне от этих акций. На страницах сайта, посвященных розыгрышам призов, публикуется подробная информация об условиях участия, количество и перечень разыгрываемых призов, указываются места, где эти призы можно получить, отображаются результаты, публикуются данные о победителях, их фотографии и интервью с ними.

Помимо поддержки розыгрышей и лотерей, проводимых компанией среди своих покупателей, на сайте для всех посетителей можно организовывать конкурсы, прямо связанные с продукцией фирмы.

В случае, если речь идет не о корпоративном, а о тематическом портале, извлекающим прибыль из размещения рекламы, могут быть добавлены следующие разделы:

Система подготовки и публикации информации

Крупные компании имеют специальные службы, занимающиеся исключительно информационным наполнением сайта. Это могут быть штатные журналисты и аналитики или сторонние специалисты, работающие на договорных условиях. Для облегчения их работы, ускорения и упрощения публикации материалов на сайте, программистами разрабатываются специальные системы публикации, позволяющие сотрудникам самим публиковать информацию в соответствующих разделах сайта.

Источниками информации для формирования контента сайта могут служить периодические издания, собственные аналитические материалы, платные новостные ленты информационных агентств и др.

Информация для рекламодателей

Это очень важный раздел для всякого проекта, основанного на рекламной модели. Посещая его, рекламодатель должен получить исчерпывающую информацию о том, за что ему предлагают заплатить. Таким образом, данный раздел должен содержать информацию по следующим вопросам:

портрет аудитории сайта (распределение посетителей по странам и городам, средний возраст и доход, уровень образования, интересы посетителей);

посещаемость сайта. Рекламодатель должен знать размер аудитории, на контакт с которой он может рассчитывать.

рекламные носители и виды рекламы. У каждого сайта есть свои стандартные рекламные места, которые он может продать рекламодателю.

Эти места должны быть описаны в данном разделе. Кроме того, рекламодателю может быть предложено размещение рекламных сообщений в e-mail рассылке или спонсорство в каком-либо проводимом на сайте конкурсе или турнире;

возможности по таргетингу (фокусировке) рекламы на отдельные категории посетителей. Многие рекламодатели заинтересованы не во всей аудитории сайта контент-проекта, а только в ее части.

цены на размещение рекламы. Потенциальный рекламодатель должен сразу видеть, сколько ему придется заплатить. Прайс-лист должен быть подробным и содержать цены на размещение всех видов рекламы в соответствии с различными ценовыми моделями.

Информация для рекламных агентств

Рекламные агентства – профессиональная организация, - планирующая рекламные компании, производящая рекламу на заказ, а также размещающая рекламу.

Таким образом, рекламные агентства представляют перед рекламодателем интересы своих партнеров. Такими партнерами часто становятся контент-проекты. Они предоставляют рекламному агентству скидку с цен, указанных в прайс-листе. В результате рекламные агентства сами получают возможность делать скидки рекламодателю в пределах скидки, полученной от контент-портала.

Заключение

Внимательный взгляд на процессы, входящие в стоимостную цепочку электронного бизнеса, делает очевидным тот факт, что контент является критичным фактором ведения электронного бизнеса:

Контент сокращает время выхода на рынок. Закупки и продажи через Интернет сжимают временные рамки. Во многом это происходит благодаря многократному использованию контента, а также обмену контентом с целью коллективного использования содержащихся в нем знаний, дающего бизнес-менеджерам возможность гораздо быстрее понять, чем именно они обмениваются. Возможность использования существующей информации во множестве различных, специализированных форм сокращает время, необходимое для предоставления потребителям нужной им и максимально точной информации.

Контент управляет принятием решений о закупках. Персонализированный и целенаправленный контент позволяет менеджерам принимать более информированные решения, и делать это гораздо быстрее, т.к. необходимая для принятия решений информация находится в их распоряжении.

Контент повышает удовлетворенность потребителей. Благодаря наличию специализированной, актуальной и легкодоступной информации о продуктах и услугах, потребители имеют возможность решать собственные

проблемы (а также избегать нежелательных проблем) и получать удовлетворение от осознания того факта, что они сделали правильный выбор.

Сопровождение Web-сайта компании, строящей решение e-bussiness, более не является заботой дизайнера-одиночки или даже команды Web-разработчиков. Вся организация, участвующая в стоимостной цепочке "Снабжение-Производство-Сбыт", становится поставщиком контента и информации.

Web-сайт становится интегральной составляющей бизнес-процессов, и любой из участников бизнес-процессов оказывается интегрированным с Web-сайтом. Поэтому контент, используемый и создаваемый участниками бизнес-процессов, должен управляться специальными автоматизированными системами.

ГЛОССАРИЙ

CGI	*Common Gateway Interface* – Стандарт создания программных модулей. Обычно CGI-модули используются для создания программной части web-сайтов, например, для работы с базой данных или системой демонстрации рекламных баннеров.
Cookies	Специальные файлы-метки, направляемые сервером клиентскому браузеру при обращении к web-сайту. Сервер запрашивает браузер посетителя о наличии соответствующего файла cookie, и если таковой будет обнаружен, посещение считается повторным.
CMS	*Content Management System* – Система управления содержимым либо контентом, часто для простоты их называют «движком сайта», позволяющая автоматизировать управление сайтом: его дизайном, структурой, содержимым и применять дополнительные пользовательские функции, например, персонализация и т.д.
CRM-системы	*Customer Relationship Management* – Система управления взаимоотношениями с клиентами.
DMS	*Document management system* – Система управления документами.
ECMS	*Enterprise Content Management System* – Система управления содержимым уровня предприятия.
EDI	*Electronic Data Interchange.* Обмен электронными данными. 1. Передача деловой информации с компьютера на компьютер в стандартном формате. 2. Глобальная компьютерная сеть, отделенная от Интернет. Используется банками и другими финансовыми институтами для проведения платежей.

ERP	*Enterprise Resource Planning* – планирование ресурсов предприятия. ERP-система – конкретный программный пакет, реализующий стратегию ERP.
GUI - элементы	Graphical user interface – графический интерфейс пользователя, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.
HRM-системы	*Human Resource Management* – системы управления трудовыми ресурсами, автоматизирующие управления персоналом.
HTML	*Hyper Text Markup Language*. Язык разметки для создания страниц в Интернете, в которых объединены гиперссылки, текст, графика, звук и видео. HTML состоит из независящих от программного обеспечения команд, описывающих структуру web-страницы.
Master – Slave	Ведущий – ведомый. В сетях, работающих по принципу ведущий – ведомые, только ведущее устройство может инициировать передачу данных и определяет порядок доступа к сети.
SEO-оптимизация	*Search Engine Optimization* – поисковая оптимизация, комплекс мер для поднятия позиций сайта в результатах выдачи поисковых систем по определенным запросам пользователей с целью продвижения сайта.
SSL	*Secure Socket Layer* – протокол для аутентификации и криптозащиты каналов связи, работающих на базе протоколов TCP/IP. Согласно этому протоколу клиенты и сервера могут аутентифицировать друг друга, а затем обмениваться зашифрованными данными.
Web-сервер	Программное обеспечение, которое управляет данными на web-узле, контролирует доступ к этим данным и отвечает на запросы web-браузера.
Web-хостинг	Размещение web-страниц в сети Интернет на арендованном дисковом пространстве какого-либо сервера.
WWW	*World Wide Web* – всемирная информационная сеть, основанная на гипертекстовом представлении информации – набора страниц (сайтов) и ссылок на другие страницы (сайты) и выполняемые программные модули.

Авторизация	Разрешение на доступ к ресурсам или службам. Авторизация пластиковой карты – установление платежеспособности карты. В результате процесса авторизации эмитент карты дает разрешение на совершение транзакции. Этот процесс подтверждает, что ограничения по сумме платежа для данной карты не превышены и резервирует указанную сумму.
Аутентификация	Процедура проверки подлинности участвующих в сделке сторон и проверка соответствия их действий заключенному договору.
База данных	Информационная модель подсистем реального мира или другими словами – совокупность данных о конкретной предметной области. Назначение базы данных: сбор, хранение, поиск, извлечение и представления данных.
Баннер	Рекламный графический блок, связанный гиперссылкой с сайтом рекламодателя.
Бизнес-процесс	Совокупность связанных между собой операций, процедур, с помощью которых реализуется конкретная коммерческая цель деятельности компании в рамках определенной организационной структуры; при этом функции структурных подразделений и их отношения между собой заранее четко определены.
Брандмауэр	*Firewall* – аппаратное или программное средство, защищающее и контролирующее соединение одной сети с другими. Осуществляет контроль за доступом к локальной сети, анализируя содержимое поступающих извне информационных пакетов.
Веб - кластер	*Cluster* – скопление. Веб-кластер – это группа серверов, которые объединены высокоскоростными каналами связи и сгруппированы логически. Эти сервера способны обрабатывать идентичные запросы и используются как единый аппаратный ресурс. Веб-кластер предназначен для масштабных проектов с высокими требованиями по отказоустойчивости и надежности.
Верификация	Проверка адреса, куда поступает счет за купленный товар. Делается, чтобы бороться с обманом при покупках по почте, телефону, Интернет.
Выделенная линия	Постоянный канал связи, предоставленный провайдером.

Дата-центр	*Data center* – центр хранения и обработки данных. ЦОД/ЦХОД – это специализированное здание для размещения (хостинга) серверного и сетевого оборудования и подключения абонентов к каналам сети Интернет.
Интерактивный каталог	Специализированный ресурс Интернет, содержащий структурированную по тематическим разделам базу ссылок на другие web-сайты.
Интернет-провайдер	*ISP – Internet Service Provider.* Организация, предоставляющая доступ к Интернет, включая электронную почту и доступ к WWW.
Интранет	Корпоративная сеть, работающая на основе технологии WWW с использованием протоколов семейства TCP/IP.
Информационный портал	Большой виртуальный массив информации, включающий в себя множество различных тематических разделов меньшего размера либо некоторое количество самостоятельных проектов.
Клиент/сервер	Сетевая архитектура. Отдельные компьютеры, называемые серверами, специально выделяют для работы в качестве поставщиков услуг всем другим компьютерам, называемым клиентами, на которых пользователи выполняют свои задания. Серверы могут применяться для выполнения одной или нескольких функций, например, хранения данных, печати, связи, электронной почты и доступа в Web.
Контент	Любая информация, графическая или текстовая, размещенная на сервере.
Кэш	*Cache* – прятать. Промежуточный буфер с быстрым доступом. Кэширование веб-страниц может осуществляться с помощью CMS конкретного сайта для снижения нагрузки на сервер при большой посещаемости.
Логическая структура сайта	Набор тематических рубрик с распределенными по соответствующим разделам документами и спроектированными гиперсвязями между всеми страницами ресурса.
Облачное хранилище данных	*Cloud Storage* – модель онлайн-хранилища, в котором данные хранятся на многочисленных распределённых в сети серверах, предоставляемых в пользование клиентам, в основном, третьей стороной.

Плагин	*Plug in* – "подключать". Независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей. Плагины обычно выполняются в виде разделяемых библиотек.
Сервер приложений	Компьютер, с прикладным программным обеспечением для совместного использования всеми клиентами сети.
Система управления сайтом	Система, которая позволяет управлять информацией на сайте, модернизировать дизайн сайта и изменять структуру сайта.
Экстранет	Расширение сети Интранет до сетей внешних экономических партнеров.

Оглавление