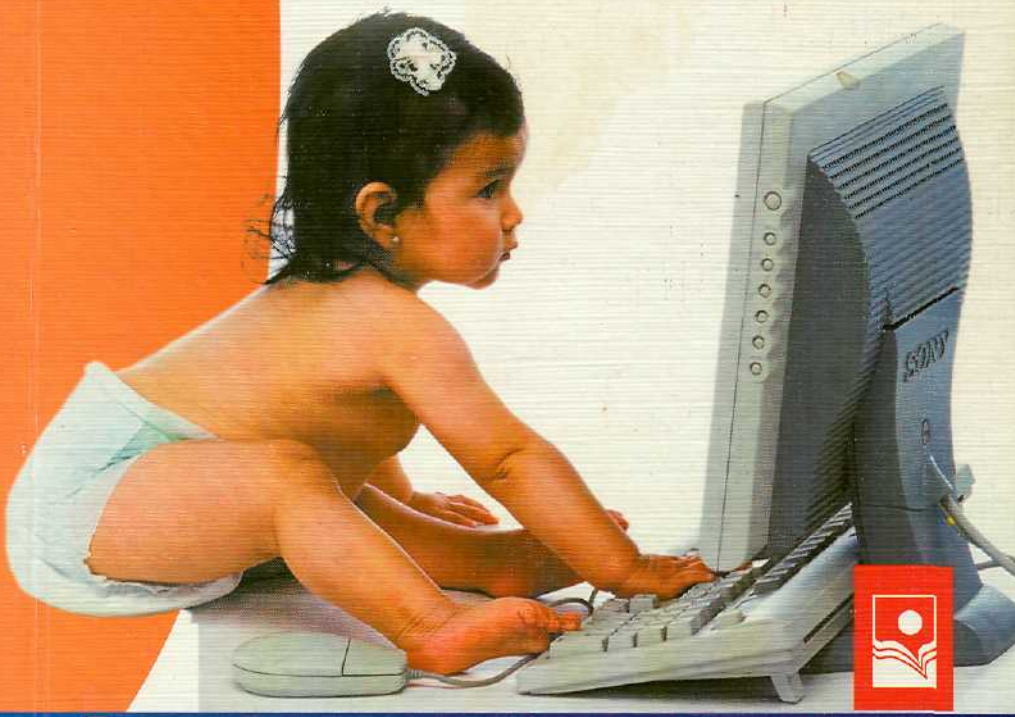
А. Кухарчик

PHP



обучение на примерах

А. Кухарчик

PHP:

обучение на примерах

МИНСК ООО «НОВОЕ ЗНАНИЕ» 2004

УДК 004.92
ББК 32.973.26-018.2
К96

Кухарчик А.

К96 **PHP:** обучение на примерах/ А. Кухарчик. — Мн.: Новое знание, 2004. — 237 с.

ISBN 985-475-050-7.

Просто и доступно изложены основы PHP — популярного языка написания скриптов для Web-страниц. Выполняя несложные примеры, читатель сможет освоить азы программирования на PHP, создать динамическую страницу, счетчик посещения сайта, систему голосования, адресную книгу, интернет-магазин. Описаны типичные ошибки использования сценариев, а также пути их исправления. Книга содержит фрагменты кода, который можно применять при создании собственных проектов.

Предназначена прежде всего для начинающих, но может быть полезна и опытным программистам, использующим **PHP.**

УДК 004.92
ББК 32.973.26-018.2

ISBN 985-475-050-7

Оглавление

Часть I
Что такое интернет

Интернет уже давно прочно вошел в нашу жизнь. Прошли те времена, когда многие пользователи персональных компьютеров в целях экономии дискового пространства удаляли программу Internet Explorer из всеми нами любимой операционной системы, так как просто не нуждались в ней. Теперь даже те, у кого нет доступа к «Паутине», стараются не трогать эту программу, ведь форматы интернета давно уже перешли в другие сферы нашей компьютерной жизни. Очень часто в формате HTML (см. приложение 1) выпускаются различные электронные книги (например, энциклопедии), журналы и т.д. И все это многообразие в обязательном порядке требует наличия на компьютере браузера — программы для просмотра HTML-документов. Яркий представитель этого класса — Internet Explorer, благодаря стараниям корпорации Microsoft давно уже вышедший в лидеры программ такого рода.

А уж если в компьютере поселился модем (что по нынешним временам далеко не редкость, цены на модемы сильно снизились), избежать программы-браузера просто невозможно. Ведь хочется же хоть одним глазком посмотреть, а что же там, в большом мире. И когда знакомство с интернетом состоялось, отказаться от преимуществ Сети не так-то просто.

Через некоторое время бестолкового (или толкового) путешествия по интернету начинаешь осознавать, что надо как-то реализовать себя и обозначиться на бескрайних просторах Глобальной сети. Своя страница в интернете — это даже престижно, и пусть там мало полезного, а счетчик фиксирует только ваши не частые

посещения, сделанное может стать началом серьезного проекта и изменить всю вашу жизнь.

Путешествуя по просторам интернета, вы, наверное, не раз обращали внимание на адреса страниц, которые порой достигают внешне очень больших размеров и просто приводят в недоумение наличием странных символов. И конечно, все имели дело с формами, которые требовалось заполнить и получить что-то взамен (услуги, вещи и т.д.). Или видели постоянно изменяющиеся страницы, на которых сначала одно, а через минуту — уже другое.

Но немногие задумываются над тем, как все это работает. Предлагаю вам окунуться в мир программирования. Этот мир во многом ни на что не похож, но у него есть свои законы и правила, свои плюсы и минусы. После первого шага, сделанного в этом мире, идти будет уже гораздо легче.

Сегодня практически не осталось человека, который не слышал бы об интернете. Еще с десяток лет назад о нем знали только специалисты узкого профиля, а пользователей можно было пересчитать буквально по пальцам. Но время идет вперед, и то, что еще вчера казалось дорогим и ненужным удовольствием, сегодня не только реальность, но и насущная необходимость.

Итак, что такое интернет? Это слово пришло к нам из английского языка и является составляющим от слов «inter» и «net» (между и сеть). Понятно, что это слово должно обозначать какую-то связь между сетями, в которые объединены компьютеры в различных местах. Так оно и есть на самом деле. Интернет — это свободная международная сеть компьютеров, объединяющая в себе все страны и все континенты нашей планеты. Впечатляет? Да, действительно, работая в интернете, вы с легкостью перемещаетесь между компьютерами (виртуально, конечно), которые могут быть расположены на огромных расстояниях друг от друга.

Физически интернет представляет собой сеть из волоконно-оптических или медных проводов, соединяющих удаленные друг от друга компьютеры или сети компьютеров, на которых размещена различная информация. Вместо проводов могут применяться

другие современные технологии: радио, лазерная или инфракрасная связь, GPRS и т.д. Впрочем, для нас это пока дело недалекого будущего. Компьютеры, обслуживающие интернет, тоже несколько необычные. Это не те «персоналки», которые мы привыкли видеть на работе или дома, а специальные серверы, которые умеют одновременно выполнять сразу несколько важных функций, обеспечивая бесперебойную работу интернета круглосуточно. Для них пишут специальные программы и операционные системы, производят надежное оборудование с повышенными требованиями, и управляют всем этим специалисты высокого уровня.

Точного дня рождения интернета никто не знает. Известно только, что первые разработки производились в 1969 году в США. Тогда эти исследования были строго засекречены и предназначались в первую очередь для военных целей. Отпечаток тех далеких дней до сих пор лежит на структуре интернета — он работает практически независимо от конкретных серверов и магистральных линий. Повреждение одного или нескольких каналов передачи данных почти не сказывается на качестве связи, так как Сеть сама перестраивается и направляет информационные потоки по целым и наименее загруженным линиям. Основная идея интернета —- отсутствие централизованного управления. Именно этот факт и привел к взрывному развитию Сети.

Сегодня интернет — это глобальная информационная сеть, доступная практически каждому желающему. По исследованиям на момент написания книги, почти 10 % населения нашей планеты (около 580 млн человек) имеют доступ к интернету. Если развитие Сети пойдет такими же темпами, как в последнее десятилетие, то к 2005 году будет преодолен рубеж в 1 млрд пользователей.

Самая «интернетизированная» страна сегодня — это, как ни странно, Исландия. Доступ к Сети там имеет почти 70 % населения страны. США занимают в этом списке только пятое место, уступив также Швеции, Дании и Гонконгу. Очень важную роль играет развитие телекоммуникационных услуг в стране и, конечно, уровень благосостояния населения. Ведь пользоваться услу-

гами Сети можно не только при помощи настольных, но и карманных компьютеров, мобильных телефонов и т.д.

Наиболее распространенный и самый старый сервис, предоставляемый интернетом, — это электронная почта. Практически мгновенная доставка сообщения привлекает все больше и больше поклонников своей простотой и доступностью. Кроме текста, есть возможность передать любой файл, прикрепляя его к письму. Но сервисом, принесшим интернету его нынешнюю известность, по праву считают WWW (World Wide Web, Всемирная паутина). Она была создана в 1992 году Тимом Бернерсом-Ли из Европейского центра ядерных исследований (CERN), расположенного в Женеве (Швейцария). К октябрю 1993 года WWW насчитывала около 200 действующих Web-серверов, а уже к июню 1995 года — свыше 6,5 млн. В настоящее время существует более 36 млн Web-серверов, разбросанных по всему миру.

Всемирная паутина представляет собой систему связанных друг с другом страниц, использующих так называемый гипертекст. Связь между страницами и связь одного места на странице с другим осуществляют гиперссылки. При помощи простого щелчка мышью по гиперссылке, которая может быть не только в виде текста, но и графического изображения, можно перейти в любое место Сети. Специальный язык разметки гипертекста называется HTML (HyperText Markup Language). Он позволяет разместить на странице текст, картинки, оформить простой текст (выделить курсивом, полужирным, изменить размер, тип или цвет шрифта и т.д.), а также сделать еще много других интересных вещей. Подробнее про HTML читайте в приложении 1.

Каждый документ в интернете имеет свой уникальный URL (Uniform Resource Locator, унифицированный указатель ресурсов). Он является ключом к поиску информации, находящейся в Сети. Введя URL в адресную строку браузера, можно получить полную информацию, содержащуюся на удаленном компьютере, которому присвоен такой адрес.

Рассмотрим URL на примере:

http://www.wnk.biz/some_directory/index.html

Адрес читается слева направо. Первая его часть — http:// — указывает тип сервера, которым вы хотите воспользоваться. В частности http:// указывает на протокол HTTP (см. ниже).

Вторая часть адреса — www.wnk.biz — доменное имя сервера. Домены — это зоны, на которые делится интернет и которые в свою очередь подразделяются по типу: домен com — обозначает коммерческие организации, edu — учебные и научные, gov — правительственные, mil — военные, net — сетевые, org — другие организации. Кроме того, существуют домены, указывающие на страну, в которой расположен данный сервер. Например, by — Беларусь, ru — Россия, ua — Украина, pl — Польша и т.д. Каждой стране присвоен свой уникальный домен. Это все домены первого уровня. Они подчиняются специальной международной организации, созданной для контроля и наблюдения за интернетом. Именно там недавно было принято решение об открытии новых доменов, например: info — информационные ресурсы, biz — бизнес-ресурсы и т.д.

Третья часть — some_directory — указывает путь на сервере (каталог some_directory) к файлу, который вы собираетесь открыть.

Последним следует имя самого файла — index — и его расширение — html.

На самом деле, все адреса в Сети, как, впрочем, и любая информация в компьютере, представлены в виде набора цифр. Каждый сервер имеет свой уникальный адрес, например 192.128.45.194. Это так называемый IP-адрес компьютера, и уже с ним ассоциируются нормальные имена сайтов, которых на одном сервере с одним IP-адресом может быть очень много.

Сайт — это набор различных интернет-документов, объединенных под одним адресом. Сайты бывают самых различных тематик: поисковые, каталоги, информационные, личные странички, игровые и т.д. Очень многие сайты рекламируют друг друга и, таким образом, поднимают свою посещаемость. А высокая посещаемость — это залог привлекательности сайта для рекламодателя, а значит, возможность заработать.

Вот так устроен интернет. Это только малая часть его возможностей, но зато наиболее часто используемая.

Зачем нужен интернет? Интернет может служить универсальным средством для многих слоев населения планеты. Именно поэтому он и приобрел такую огромную популярность. Во-первых, это огромное количество информации. Настолько огромное, что существуют специальные сайты — каталоги и поисковики, которые регулярно автоматически просматривают доступные участки Сети и педантично регистрируют содержимое документов в своих базах данных. Во-вторых, это универсальное средство общения. Всевозможные чаты, электронная почта, телеконференции — чего только не придумано для того, чтобы упростить процесс контакта друг с другом. Интернет вдруг сделал весь мир очень маленьким и легкодоступным. Не представляет никаких трудностей одновременно общаться с друзьями из совершенно разных стран и континентов. И все это в реальном времени. В-третьих, интернет представляет собой четвертое средство массовой информации (после газет, радио и телевидения). Пользоваться интернетом в этом качестве гораздо удобнее и эффективнее, чем всем остальным, вместе взятым. Кроме того, устойчивость сети к катаклизмам часто играет просто неоценимую роль в развитии событий. Самый свежий пример — пожар на Останкинской телебашне в Москве в 2000 году. Тогда прекратили работу теле- и радиосистемы, но интернет продолжал исправно функционировать, и люди могли продолжать получать информацию о развитии событий. Наконец — бизнес. В интернете можно не только получить информацию, но и, например, купить что-то в интернет-магазине или поучаствовать в интернет-аукционе. А если вы хороший специалист, вам не составит труда найти высокооплачиваемую работу.

Не сомневаюсь, каждый найдет для себя что-то в интернете, настолько он многообразен и универсален.

Как пользоваться интернетом у себя дома? Чтобы ходить в интернет, нужен компьютер. Достаточно слабого, подойдет даже старенький 486, но учитывайте то, что современные страницы уже давно не ограничиваются только текстом. Они полны графи-

ки и активного содержимого, что определяет некоторые требования к ресурсам компьютера. И конечно, современные браузеры — это объемные и многофункциональные программы, которые также требовательны к системам, на которых они работают. Но в любом случае слабенького Pentuim вполне достаточно, а новые компьютеры даже более чем избыточны.

Кроме компьютера нужен модем. Это специальное устройство, преобразующее цифровую информацию (понятную компьютеру) в модулированный аналоговый сигнал (который может быть передан по телефонным линиям), и наоборот, воспринимающее такой сигнал и осуществляющее обратное преобразование. Само собой, наличие телефонной линии обязательно. Модемы бывают самые разные: недорогие и очень дорогие, внутренние и внешние, с дополнительными возможностями и без них. Для дома будет вполне достаточно недорогого внутреннего модема. Он устанавливается в системный блок компьютера, не требует отдельного питания, прост в настройке и не занимает места на рабочем столе. Стоимость такого устройства крайне мала, по сравнению со стоимостью компьютера, и приобретать его надо. Компьютер без модема сегодня — это то же самое, что телевизор без антенны.

Если модем есть, нужно настроить соединение с провайдером. Провайдер — это организация, предоставляющая услуги по подключению к компьютерным сетям, в частности к интернету. Существует такая услуга, как беспарольный доступ к интернету. Суть ее в том, что не надо заключать договор с провайдером, нет абонентской платы, а счет приходит вместе со счетом за телефон и включает в себя поминутную оплату времени, проведенного в интернете за прошедший месяц. Такой подход очень удобен, так как позволяет подключиться с любого компьютера без заключения договора и работать тогда, когда это необходимо.

Приглашаю и вас приобщиться к интернету и внести в его развитие свой посильный вклад. Это совсем не так сложно, как кажется на первый взгляд.

Часть II
Готовимся к созданию Web-страниц

Самое главное в этом деле, как и в любом другом, — решиться. Далеко не так просто, как кажется на первый взгляд, принять решение и начать делать все с нуля, на собственном опыте узнавая все «прелести» сайтостроительства. А тем более такого сложного, как, например, интернет-магазин.

Честно говоря, я вам даже немного завидую. Вы сейчас держите в руках эту книгу и, может быть, даже думаете, читать ее или нет, нужно мне это программирование или просто воспользоваться готовыми решениями? У меня, к сожалению, такого выбора просто не было. Я вынужден был заняться программированием просто потому, что так было надо для меня и для моей работы. Хотя, если посмотреть с другой стороны, это было даже к лучшему, иначе я просто не смог бы получить тот огромный (это мне так кажется) опыт разработки не только собственного интернет-магазина, чата, форума и т.д., но и многих других сервисов.

Можно долго взвешивать все за и против выбора готового решения, но я приведу довод, который перевешивает все остальные. В собственноручно сделанном проекте вы сможете быть хозяином и не зависеть от каких бы то ни было разработчиков. Хочу, тут баннер поставлю, хочу, здесь форму для подписки на новости размещу. И это уже не говоря о том, что так выйдет не только намного дешевле, но и перспективней! В процессе создания своих программ (пусть сначала небольших) закладывается база для следующих, еще более серьезных проектов.

Есть, правда, и минусы, как и в любом серьезном деле. Самый большой из них — время. Время, время и еще раз время. Это

часы, дни и ночи, убитые перед монитором в бесчисленных экспериментах. Так что подумайте еще раз перед тем, как мы начнем. Тем более что я не всегда буду давать практически готовые решения, и вам просто придется многое додумывать и изобретать. Я стараюсь показать путь, а идти по нему придется уже самостоятельно. Иначе было бы неинтересно, правда?

Итак — вперед...

Железо, железо и еще раз железо...

Как ни жаль, но без компьютера, желательно своего, «домашнего», нам просто не обойтись. Почему именно домашнего? Объясняю. Когда я прихожу на работу в офис, то просто не могу написать ни строчки кода. Все ходят, что-то спрашивают, куда-то зовут, всем все надо именно в такой ответственный момент!

Совсем другое дело — дома. Не спеша встал, позавтракал, дети уже давно в садике (школе, университете, еще не вернулись с дискотеки и т.д.), сел себе спокойненько за компьютер — и работа пошла, ни на что не отвлекаясь! К этому всему неплохо бы иметь хорошую жену, чтобы приготовила завтрак, отправила детей в школу и отвечала на телефонные звонки. Но тут я не советчик, ищите выход сами.

Теперь, собственно, «железо», т.е. аппаратное обеспечение компьютера. Не ждите, что я буду уговаривать вас использовать для работы самую современную технику. Да, это приятно, когда все работает и работает быстро. Но, взглянув правде в глаза, надо признать, что вполне достаточно правильно настроенного Pentium-I Windows 98. Что такое «правильно настроенного»? Поясню, на своем примере. У меня довольно мощный по сегодняшним меркам компьютер, но он пока один, и кроме меня его используют жена и не по годам смышленый старший сын. Да и младшая дочь уже ручонки протягивает. И мне надо как-то защищаться от этого безобразия, так как нет-нет да и поломают мои доброжелательные домочадцы что-то в системе.

Вот и приходится или восстанавливать все заново, что мне просто лень делать, так как это долго, или ограничивать их права. Именно это я и делаю, и вам советую. Правда, в таком случае Pentium-I не обойдешься. Ему не вынести установленной у меня Windows XP с файловой системой NTFS. Я делаю так: себе назначаю права администратора, сыну и жене — пользователя (эту умную мысль я прочитал где-то уже после того, как на собственном опыте постиг истину). Вот уже больше года все в порядке. Если не считать плановых замен винчестера, когда я все равно предпочитаю переустанавливать систему, то больше ее трогать просто нет причин.

Но это в идеале. Если так «разогнаться» нет возможности, устанавливаем Windows 98 на компьютер, на котором она будет работать. Неплохо бы иметь побольше памяти, чтобы быстрее все работало, и винчестер немаленький, и привод CD-RW для резервирования своих трудов, и модем получше, и монитор побольше. Неплохие запросы, да? Собственно, без всего этого можно обойтись. Любое современное оборудование вполне подходит.

Очень полезно иметь какое-то эффективное средство для резервирования и переноса данных. У меня эти задачи разделены между приводом CD-RW и flash-накопителем. Удобно записывать большие объемы на CD, а маленькие (например, свою работу синхронизировать дома и в офисе) — при помощи USB flash-накопителя. Только мой вам совет: не забывайте правильно завершать работу с flash-накопителем.

Домашняя эргономика — безопасность прежде всего!

Занимаясь компьютерами уже больше десяти лет (начиная с 1988 года, если не считать программируемые калькуляторы, на которых не один раз совершил посадку на Луну и другие планеты Солнечной системы, кто-то еще помнит, что это значит?), я постоянно задаюсь вопросом, как максимально защитить себя от их пагубного влияния на здоровье? Ведь не секрет, что если ра-

ботать на компьютере достаточно долго и ежедневно, это влияние почувствуется очень скоро и достаточно сильно.

Самый ответственный момент во всей эргономике, на мой взгляд, — это глаза. Их нужно беречь больше всего, так как они страдают в первую очередь. Еще в бытность самодельного «Спектрума» (популярный компьютер 1980-х годов) приходилось садиться подальше от телевизора и делать периодические перерывы в работе, прогуливаясь на свежем воздухе. Монитор в то время казался верхом совершенства эргономики, да и слова-то такого никто не знал. Уже давно бытовые телевизоры не применяются в качестве мониторов, но защита человеческого глаза от излучения электронно-лучевой трубки ушла не так уж и далеко. Конечно, я знаю о защитных покрытиях на кинескопах, о соответствии строгим требованиям мировых стандартов, о защитных фильтрах. Но на самом деле все несколько сложнее. Например, вы не задумывались, куда должно уходить вредное излучение? Ведь оно никуда не исчезает, просто специальные защитные системы отводят его... на заземление. Плохо или хорошо они это делают, нам никогда не узнать, пока нет этого самого заземления. Так что и говорить о соответствии монитора мировым стандартам и эффективности защитного экранного фильтра можно только в том случае, если заземление подключено, причем правильно. А у вас оно есть? У меня, например, нет.

И у вас, скорее всего, ответ будет отрицательный. Это неудивительно: кому нужна забота о человеке? Вот и строили, и строят до сих пор дома без третьего провода в розетке, а значит, и без возможности заземления.

Если не позаботиться о себе самому, никто о тебе не позаботится. Возможность заземления все же есть, просто для этого придется приложить некоторые усилия, знания и способности.

Самый простой (но неверный) способ — подключить компьютер к батарее центрального отопления. Действительно, трубы и батареи в любом случае хорошо заземлены, но все равно риск получить 220 В на корпусе компьютера остается, так как никто не даст гарантии, что не произойдет случайное замыкание на батарею бытового прибора. Да и если на корпусе вашего компьютера

появится напряжение, его может почувствовать сосед, который на свою беду решит принять душ. Так что палка эта о двух концах, и оба могут больно ударить.

Если у вас есть встроенный ТВ-тюнер, вам повезло. Оплетка телевизионного кабеля заземлена, и кое-какую защиту вы получите, но не стоит на нее особо рассчитывать. Оплетка имеет сопротивление более 70 Ом, а в любом учебнике электротехники сказано, что заземление должно подключаться специальными кабелями с сопротивлением не более 4 Ом. В качестве кабеля для заземления хорошо (хоть и не идеально) подойдет медный витой жгут. Он гибкий, легко прокладывается по помещению, его можно купить на рынке. А вот куда его подключить — другой вопрос, причем самый важный в этом деле.

Со стороны компьютера все более-менее ясно. Как правило, на корпусе (если он достаточно хороший) есть специальный разъем для подключения заземления. Если нет — прикрутите (только очень крепко) жгут к любой металлической части корпуса компьютера. Предварительно, конечно, хорошо зачистите жгут и корпус. Этим вы обеспечите надежное заземление всех устройств — и принтера, и модема, и монитора. А вот с другой стороны придется подключаться к распределительному щитку. Если это подъезд жилого дома, нужно будет вызывать электрика, так как щитки, как правило, закрыты на замок. Если щиток открыт, подавите в себе желание прикрутить жгут к нулевому проводу (большой винт с гайкой, к которому сходятся много неизолированных проводов). Он, конечно, заземлен, но за такое самоуправство можно получить штраф. А правильно — просверлить отверстие в металлической части распределительного щитка и с помощью винта крепко прикрутить жгут заземления к нему, не забыв все хорошенько зачистить. Вот такое решение и будет безопасным.

Если вы живете в частном доме, заземление можно сделать самому. Достаточно закопать неглубоко во влажную землю лист металла, и подключить жгут к нему. Можно для этих целей воспользоваться железными конструкциями, погруженными в землю. Только не перепутайте их с громоотводом, а то такое «заземление» сыграет с вами злую шутку во время очередной грозы.

Впрочем, не огорчайтесь, если установить заземление нет никакой возможности. Если вас мучают головные боли от работы за компьютером и болят до рези глаза, вам стоит попробовать специальные очки. В них используются фирменные линзы, имеющие патент, гарантии и соответствующие сертификаты. Линзы снимают часть (будем реалистами) вредного излучения от мониторов, телевизоров, яркого солнца и т.д. На практике это действительно приносит пользу (проверено). Уж не знаю, что там эти линзы снимают на самом деле, но голова и глаза при их применении уже не болят. Найти линзы (и заказать очки) можно в специализированных коммерческих отделах, торгующих оптикой. По словам продавцов, линзы пользуются популярностью и могут быть подобраны индивидуально под любое зрение. Только один совет: если уж покупаете, покупайте в легкой (и соответственно дорогой) оправе. А то голова будет болеть уже не от монитора, а от очков :-) Сами линзы не дорогие — за пару около двух евро, так что попробовать стоит.

Кроме внешних средств защиты от излучения мониторов, пристальное внимание стоит обратить на сам аппарат. Если он старый, не соответствует требованиям мировых стандартов, то ни о какой эффективной защите не может быть и речи. Советую вам сменить такой монитор. Если вы на это решились, обратите внимание на дорогие и фирменные модели. На мой взгляд, — лучше хороший 17-дюймовый монитор, чем плохой или посредственный 19-дюймовый. Исходите из собственных финансовых возможностей, но знайте одно: выбранный монитор должен обеспечивать работу при вашем любимом разрешении с частотой на порядок выше рекомендованной. Рекомендованной считается частота 85 Гц, но оптимальный вариант — 100 Гц. Таким образом, если вы предпочитаете работать с разрешением экрана 1024 x 768, ваш монитор должен обеспечивать работу при 100 Гц и более. Но не менее 100 Гц! С другой стороны, в случае более 100 Гц монитор будет работать в режиме, близком к критическому, и ничего хорошего от такого аппарата ждать не приходится — либо сведение, либо четкость, либо яркость страдают. А чаще все вместе. Да и нежелательно эксплуатировать монитор при предельно допустимых частотах. А в характеристиках указываются именно эти, предельные параметры.

Самым лучшим решением будут мониторы на основе жидких кристаллов. Если не говорить об играх, профессиональной графике, дизайне, и цене — это идеальный выбор. Правда, цены уже начали снижаться, но пока еще находятся на достаточно высоком уровне. Мониторы на ЖКИ не имеют вредных для глаз излучений, не подвержены мерцанию, их видимая площадь экрана больше мониторов аналогичной диагонали на лучевой трубке. Множество преимуществ говорит само за себя. Однако прежде чем покупать монитор, обязательно попросите подключить его к компьютеру и показать, как он работает. Я видел модели, на которые без слез смотреть больно в прямом и переносном смысле: то изображение дрожит, то резкость, яркость и контрастность «хромают», то угол зрения предельно узкий. Очень многое зависит от матрицы, от качества кабеля и, конечно, от сигнала, который выдает видеокарта. Идеально — подключение по цифровому каналу, без преобразования в аналоговый и обратно, но такие мониторы пока редкость. Будем надеяться, что положение исправится.

С точки зрения эргономики важно также рассмотреть устройства ввода, т.е. клавиатуру и мышь. Клавиатуру желательно иметь с подставкой, «ломаную» и т.д., кому что придется по душе. Я, например, использую самодельный вариант — обычную дешевую клавиатуру с небольшим дополнением, которое несложно изготовить самостоятельно. Достаточно сшить (хорошо бы поручить это жене, сестре, матери...) мешочек из мягкой ткани в полтора раза длиннее клавиатуры и по ширине примерно равный ей, набить его плотным материалом (можно поролоном) и зашить. Толщину можно подобрать экспериментально, в пределах 5—10 мм. Руки будут лежать на этом мягком приспособлении перед клавиатурой, и значительно меньше уставать как при печати, так и при использовании мыши. Мышь в таком случае тоже можно любую, только желательно с колесом прокрутки, которое очень сильно снижает ее дневной пробег и позволяет меньше напрягаться держащей ее руке. Мой выбор — оптические «грызуны». Но не их дешевые аналоги, с которыми мучаться при работе приходится еще больше, чем с грязной шариковой мышкой. В сторону радиомышек даже не смотрите — не пода-

рок. Самое главное их преимущество — отсутствие провода, но недостатков очень много. Самый большой — наличие батареек или аккумуляторов (в первом случае их надо периодически менять, во втором — заряжать, и то, и другое — не сахар).

Еще один важный момент — воздух. Действительно, качество воздуха в помещении очень сильно влияет на самочувствие человека. При работе компьютера (как и вообще любого электрического прибора) воздух не только нагревается, но и обогащается положительными ионами и насыщается озоном, поэтому возьмите себе за правило работать только при открытой форточке. Идеальный вариант — приобрести ионизатор воздуха. Если вы считаете это несколько дорогим решением, можно воспользоваться бытовым вентилятором. Если поставить его около форточки и направить на рабочее место, он неплохо справится с удалением вредного для здоровья воздуха. Очень хорошо в этом случае подходят офисные вентиляторы на высокой ножке. Они имеют многопозиционный переключатель скорости вращения лопастей и вращающийся вентилятор в защитном кожухе.

И, конечно же, нельзя забывать о правильной осанке, перерывах в работе и горячем питании (именно горячем, а не бутербродном). Все это вносит свою лепту в качество работы и в хорошее самочувствие. А это в наше время ценится выше всего.

Программное обеспечение для Web-мастера

Один из основных инструментов Web-мастера — программа-браузер. Они, как и любое программное обеспечение, бывают разные. Я использую Internet Explorer. Это браузер, который по умолчанию встроен в самую популярную среди пользователей операционную систему. Несомненно, такая интеграция сыграла свою роль в том факте, что на сегодняшний день доля этого браузера составляет более 90 %. И это при том, что есть такие прекрасные программы-браузеры, как, например, Netscape Navigator или Opera.

Конечно, каждый выбирает для себя то, что считает лучшим.

Есть и обратная сторона медали. Очень часто разработкой проекта занимается не команда специалистов, в которой каждый отвечает за свою часть работы, а один человек. Недостатки такого подхода очевидны, однако ничего не поделать. И тут уж приходится быть мастером на все руки — заниматься и дизайном, и программированием, и многим другим. А значит, вам пригодятся на компьютере все популярные нынче в мире программы-браузеры. Ведь очень часто хорошо выглядевшая страница в одном браузере совершенно не смотрится в другом. Идеала добиться сложно, но какой-то компромисс всегда можно найти. Впрочем, это на ваш выбор. Если вы используете в работе Internet Explorer, то большинство посетителей увидят ваше творчество так, как вы и задумывали. Оптимально, если вы на странице будете подписывать, что она оптимизирована для определенной версии определенного браузера.

Нам также понадобятся текстовый редактор и FTP-менеджер. Вы можете использовать любые, только желательно, чтобы текстовый редактор обеспечивал нумерацию строк и элементарное выделение цветом синтаксических конструкций. Я рекомендую CuteHTML из комплекта CuteFTP (рис. 1), так как он обеспечивает все вышеперечисленные требования, прост в обслуживании, не требует инсталляции (точнее — интегрируется в систему при первом запуске).

CuteFTP можно использовать и в качестве FTP-менеджера, но я применяю обычный Windows Commander, так как он еще и позволяет в привычной оболочке работать с файлами (рис. 2).

Устанавливаем все эти программы.

Дальше надо выделить место для вашего творчества. Это может быть любая папка в любом месте любого диска, но лучше, если это будет в корневом каталоге или даже вообще на отдельном диске. Почему лучше? Потому, что там должен храниться не только сайт, но и сервер, который будет эмулировать работу настоящего сервера в интернете. А со временем сайт может разрастись или появятся разные версии сайта. Так что место на диске понадобится.

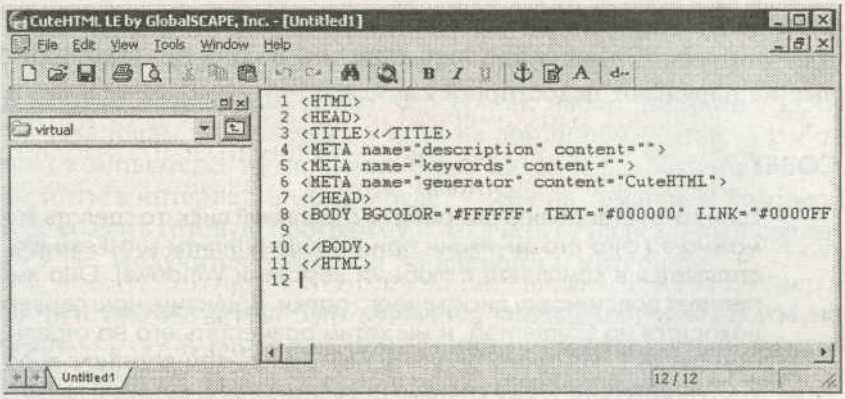


Рис. 1. Окно программы CuteHTML

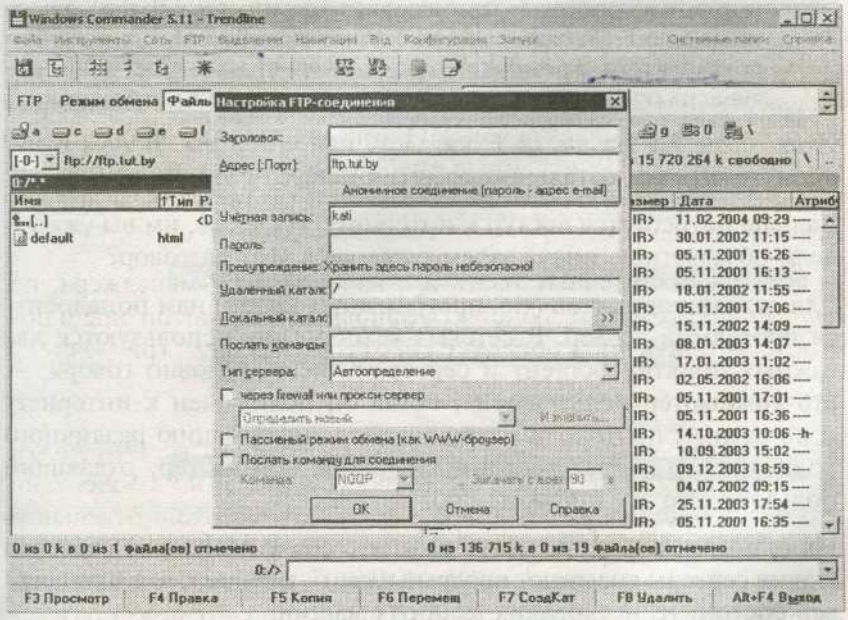


Рис. 2. Использование Windows Commander в качестве FTP-менеджера

Можно на компьютере для работы выделить отдельный диск.. Для этого существует много способов, в том числе и таких, которые не нарушают целостности системы.

СОВЕТ

Если хочется выделить для работы отдельный диск, то сделать это можно за счет его эмуляции при помощи утилиты subst.exe (поставляется в комплекте с любыми версиями Windows}. Она выполняет подстановку диска вместо папки. Допустим, наш сервер находится на C:\internef\, и мы хотим разместить его на отдельном диске E, которого реально нет. Выполняем команду: subst E: C:\internet. Теперь в системе появился диск E, который является полной копией папки C:\internet\, т.е. там находится то же самое, только теперь это располагается в корневом каталоге диска E. И, конечно, еще на диске C в папке internet, оттуда информация никуда не делась. Запускаем наш сервер из диска E, и он нормально работает. При условии, конечно, что сервер настроен на диск E. По окончании работы можно отключить диск командой subst E: /d. В результате диск E исчезнет, но информация, физически расположенная в C:\internet\, останется нетронутой.

Итак, создайте на диске E каталог, например usr. В нем будут происходить все остальные события нашего повествования.

Дальше потребуется доступ к интернету. Надеюсь, им вы уже успели обзавестись, иначе зачем тогда весь этот разговор?

Для отладки скриптов (т.е. программных кодов) нам понадобится программа-сервер. В сетевых технологиях используются два важных понятия: клиент и сервер. Клиент, условно говоря, — это компьютер пользователя, когда он подключен к интернету и по разным протоколам запрашивает информацию различного рода. Сервер — это удаленный другой компьютер, отдающий пользователю эту информацию.

На одном компьютере могут работать сразу несколько серверов (такие серверы называют виртуальными), а бывает, что один сервер состоит из нескольких мощных машин.

Вам, скорее всего, придется работать с виртуальными серверами. Они наиболее распространены, несмотря на небольшое

снижение производительности и ограниченные возможности настройки.

В данный момент нам нужен сервер как набор программ. Если мы установим такой сервер у себя на домашнем (читай — рабочем) компьютере, то не нужно будет каждый раз закачивать все скрипты в интернет, а можно будет у себя на домашнем компьютере создать готовую программу и закачать ее на реальный сервер по FTP. Получится экономия на времени в интернете.

Самым распространенным сервером является Apache. Идем на сайт http://apache.org (рис. 3) и там открываем страницу для скачивания[1].

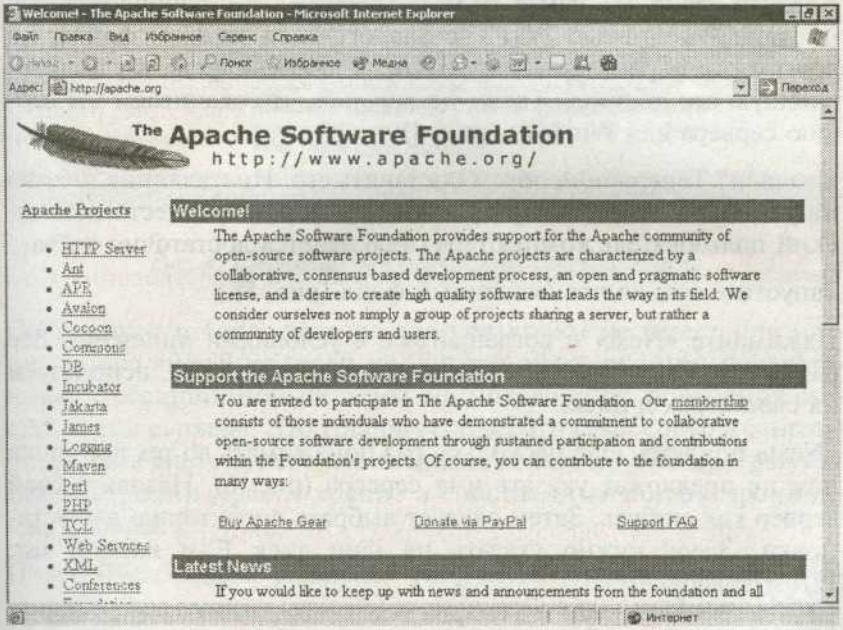


Рис. 3. Сайт http://apache.org

[1] На момент написания книги это страница http://httpd.apache.org/download.cgi. *(Примеч. ред.)*

Лучше всего, если вы скачаете и установите у себя на компьютере ту версию сервера, с которой работает ваш хостинг-провайдер[1]. Этим вы застрахуете себя от возможных неприятностей в виде некорректной работы программы. Есть тут и «подводный камень». Дело в том, что у вашего хостинг-провайдера, скорее всего, в качестве операционной системы будет не Windows, а одна из версий Unix или даже FreeBSD. Но прелесть устанавливаемого сервера Apache в том, что он не просто будет работать с любой операционной системой, он будет работать одинаково. Это для нас немаловажно, так как ставить себе такие экзотические операционные системы не каждый согласится. Windows как-то привычней.

Каждый файл на сайте http://apache.org сопровождается цифровой подписью PGP3[2] с аналогичным именем файла, но другим расширением. Не спутайте собственно файл и удостоверяющую его подпись. Также не забудьте, что скачиваем мы версию сервера для Windows (рис. 4).

Скачали? Теперь попробуем поставить его. Инсталляция похожа на установку обычной программы, с которой среднестатистический пользователь компьютера сталкивается достаточно часто.

Запустите только что скачанный файл (рис. 5).

Нажимайте «Next» и соглашайтесь с условиями лицензии. Все равно там, как обычно, — никакой ответственности, используем на свой страх и риск.

«Next» придется еще несколько раз понажимать до тех пор, пока нам не предложат указать имя сервера (рис. 6). Назовите свой сервер как-нибудь. Затем следует выбрать директорию для установки. Здесь нужно указать на наш диск E и каталог usr: E:/usr/apache/.

---

[1] Хостинг-провайдер — организация, предоставляющая место на сервере для размещения вашего сайта. *{Примеч. ред.)*

[2] PGP (Pretty Good Privacy, почти полная приватность) — ряд программных продуктов, позволяющих зашифровать и расшифровать файлы и электронные сообщения при их отправке и получении, а также добавлять цифровую подпись к файлам и любой другой информации, предоставляемой в цифровом виде. *{Примеч. ред.)*

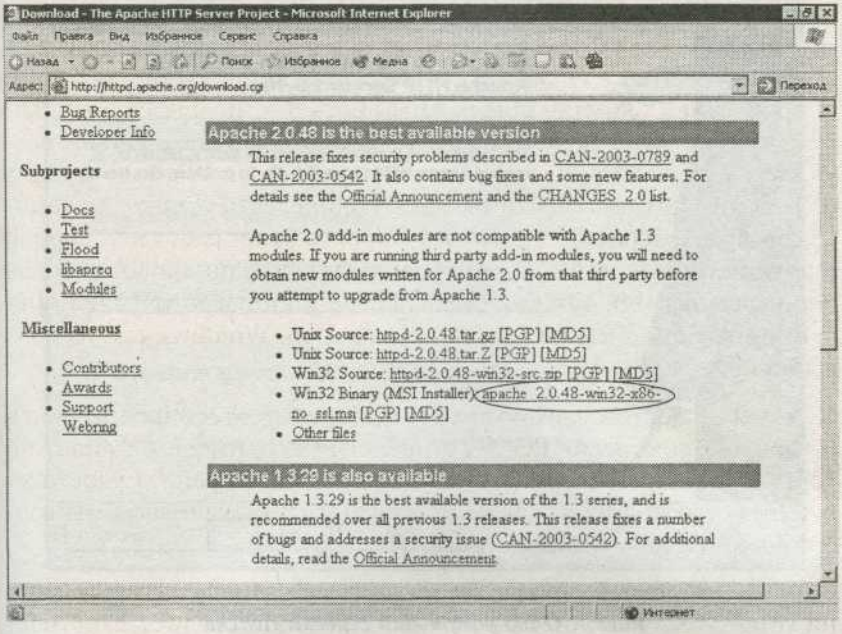


Рис. 4. Страница для скачивания сервера Apache

Собственно говоря, каталог для установки не имеет принципиального значения, если не считать того, что при переустановке операционной системы не потребуется повторной инсталляции сервера. Это говорит о том, что сам сервер не интегрируется в систему, что очень любят делать другие программы. Все настройки Apache хранит в собственных конфигурационных файлах.

Опять «Next», далее выбираем типичную конфигурацию для установки (это пункт «Typical») и опять уже знакомый нам «Next», даже два раза.

В процессе этого сервер установится в выбранную нами директорию в типичной конфигурации. После завершения установки нас поздравят с этим эпохальным событием и предложат нажать кнопку «Finish», что тут же надо сделать.

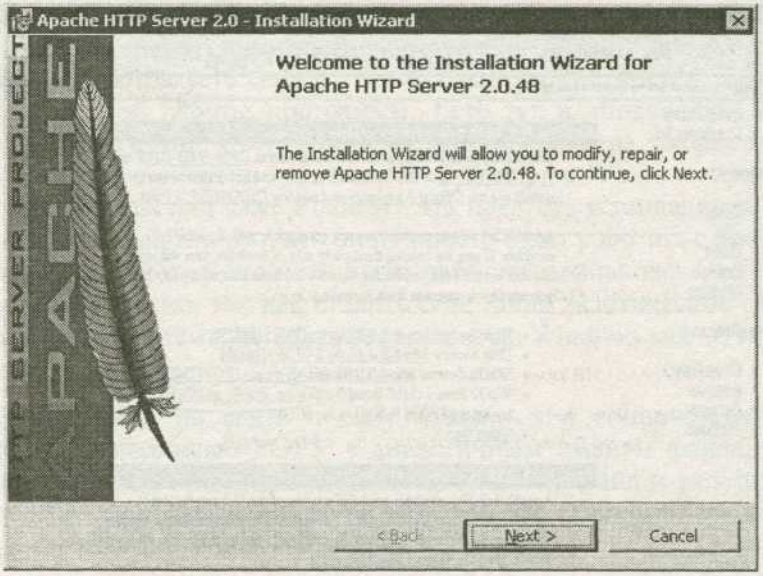


Рис. 5. Окно установки сервера Apache

Все, сервер установлен, можно его запускать. Это делается разными способами, и самый простой — зайти в каталог, в который производили установку, и запустить файл apache.exe.

Запустится окно (рис. 7). Его закрывать не надо — это и есть работающий Apache-сервер. Можно сделать для работы несколько ярлыков. Так, для запуска Apache используется «apache.exe -k start», для перезапуска — «apache.exe -k restart», а для остановки — «apache.exe -k shutdown». В последних версиях установщика Apache эти ярлыки автоматически создаются в меню «Пуск».

Другой способ запуска сервера заключается в использовании сторонних программ (я бы даже сказал — программулек), и в этом случае запуск становится на редкость приятным процессом. Я имею в виду программу-диспетчер Apache Manager для Windows. Она представляет собой красный квадратик, который «поселяется» на системной панели около часов и мирно ждет

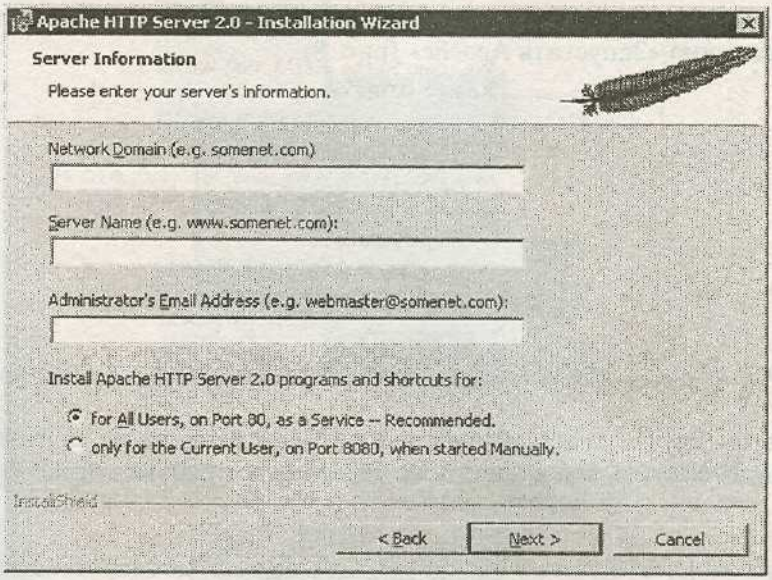


Рис. 6. Промежуточный этап установки сервера Apache

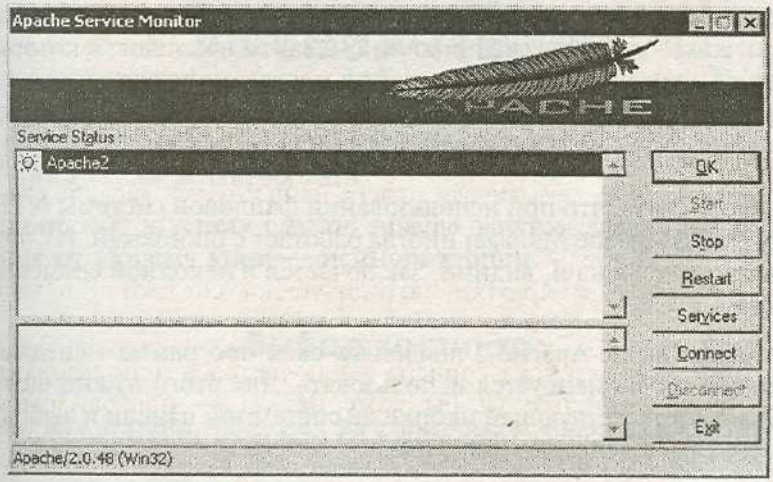


Рис. 7. Окно работающего Apache-сервера

того времени, когда по нему щелкнут и выберут пункт контекстного меню «Запустить Apache» (рис. 8).

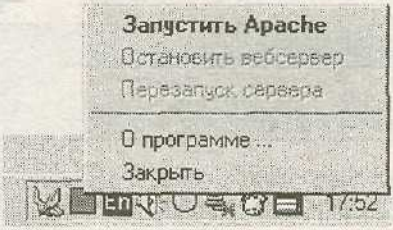


Рис. 8. Запуск Apache-сервера при помощи программы Apache Manager

В этом случае красный цвет квадратика сменится на зеленый, и будет светить все время, пока сервер будет работать (рис. 9).

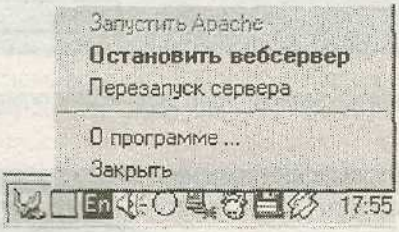


Рис. 9. Остановка Apache-сервера при помощи программы Apache Manager

Надо отметить, что при использовании файловой системы NTFS программа Apache Manager иногда работает с ошибками. Их причина пока не ясна и, видимо, заключается в неполной совместимости.

В версиях выше Apache 2 появилась своя программа-менеджер, которую и рекомендуется использовать. Для этого нужно щелкнуть по соответствующей иконке на системной панели и выбрать необходимое действие (рис. 10).

Работает? Если вы видите что-то типа «Running all Apache services» (или квадратик программы-диспетчера загорелся зеленым), то — да.

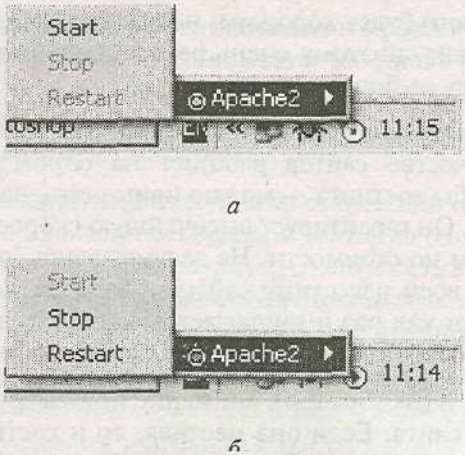


Рис. 10. Запуск (а) и остановка (б) сервера Apache при помощи собственной программы-диспетчера

Перед использованием выждите минутку — на начальном этапе Apache может прекратить свою работу из-за разного рода ошибок. Если не повезло, и окно само закрывается или в нем появляется еще какой-то текст, то, значит, где-то ошибка, попробуйте проделать процедуру запуска заново. Программа обычно подсказывает, в какой строке конфигурационного файла ошибка. Чаще всего неправильно указаны пути к файлам. Их надо исправить. Конфигурационные файлы находятся в папке e:/usr/apache/conf/. Самый важный из них — httpd.conf. Его нам, возможно, придется немного позже редактировать.

Первый шаг к успеху сделан: Apache запущен. Прежде чем перейти ко второму этапу — немного теории.

Выбор хостинга

Рано или поздно, но перед разработчиком практически любого интернет-ресурса встает вопрос: какой хостинг использовать? Что это такое? Хост — место, где будет работать ваш сайт. Хостинг — комплекс услуг по предоставлению такого места. На-

сколько это место будет хорошим, настолько будет пользоваться успехом ваш сайт, поэтому очень рекомендую внимательно подойти к вопросу выбора хостинга.

Мы рассмотрим так называемый виртуальный хостинг, т.е. когда большое количество сайтов работает на одном сервере. Есть и другие способы хостинга — можно применять, например, выделенный сервер. Он гарантирует высочайшую скорость работы, однако несравним по стоимости. Не задумывались, почему так быстро работают всем известные сайты, а ваш так долго грузится? Одна из причин как раз и заключается в разных типах хостинга. Если можете позволить себе выделенный сервер, я вам завидую.

Первое, на что надо обращать внимание при выборе хостинга, — это аудитория сайта. Если она местная, то и хостинг лучше подыскивать географически близкий, чтобы сигналы не делали дальний путь из «прекрасного далека» в «суровую действительность».

Однако есть одна тонкость: большинство фирм, предоставляющих услуги хостинга, на самом деле не имеют собственного оборудования для работы, а скупают такой хостинг, образно говоря, оптом, а потом предоставляют в розницу. Цены на оптовую закупку хостинга значительно ниже розничных, и это позволяет таким фирмам не просто зарабатывать, но и быть конкурентоспособными, обеспечивая хороший сервис и техническую поддержку. Фирма по услугам хостинга может располагаться в соседнем с вами кабинете, а сам сервер будет находиться за океаном, поэтому в первую очередь уточните, где именно физически будет работать сайт.

Знание еще одного параметра — скорости подключения к основным магистралям интернета — необязательно. В любом случае каждый уважающий себя дата-центр (место для серверов, подключенных непосредственно к магистралям интернета) старается увеличить такой канал всеми доступными ему способами. На, этом можно не особо заострять внимание, если только вам не требуются специфические условия.

Дальше мы должны оценить необходимое для работы сайта место. Когда вы создадите свой проект и оцените его размеры, то

не торопитесь делать выводы и заказывать ровно столько. Помимо непосредственно сайта, в место на диске сервера входят и объем занимаемых баз данных, и самое главное — почта. А приходилось ли вам получать мегабайты спама ? Мне — да, и мои ящики занимают на сервере больше места, чем сам сайт.

Еще есть такое понятие, как журналы статистики сайта — log-файлы. Они услужливо ведутся сервером и иногда очень полезны для анализа посещаемости, но отрицательно сказываются на занимаемом месте. Иногда log-файлы превышают все разумные пределы, и тогда приходится удалять их, как правило, по FTP.

Таким образом, не отказывайтесь от большого места. Я считаю разумным 100 Мб, этого будет достаточно даже с учетом будущего развития.

И, наконец, самое главное для нас, программистов. Поддержка всех технологий программирования на стороне сервера должна присутствовать в обязательном порядке. Впрочем, я думаю, давно невозможно найти хостинг без такой поддержки, так как установка, например, PHP ничего не стоит хостинг-провайдеру. Кроме того, надо поинтересоваться полным списком предоставляемых услуг и сервисов и уточнить, есть ли в его числе нужные именно вашему сайту.

Правда, этот момент совершенно не касается так называемых бесплатных хостингов. У них даже не ищите какого-то подобия программных технологий на стороне сервера. Их не будет. К тому же бесплатны они чисто номинально, вы будете рассчитываться баннерными показами, так что подумайте, надо ли вам такое удовольствие. Я вам решительно не советую пользоваться бесплатным хостингом, хотя и сам когда-то начинал именно с такого варианта. Впрочем, в качестве тестовой площадки и для хранения файлов большого размера иногда сгодятся, но не более того.

Стоит также обращать внимание на версии программного обеспечения у хостинг-провайдера. Добросовестный хостинг-провайдер всегда следит за выходящими обновлениями и устанавли-

[1] Спам — ненужная информация, обычно рекламного характера, приходящая на почтовый ящик пользователя без его желания. *(Примеч. ред.)*

вает их у себя, причем делает это не бездумно, а только если тестовый образец не вызывает сбои в работе.

Несомненно, в списке услуг просто необходимо ежедневное резервирование данных. Как вариант может быть ручное управление резервным копированием файлов, хотя практика показывает, что пользоваться восстановлением системы приходится не так уж и часто.

Дальше можно поинтересоваться возможностями персональной настройки сайта, т.е. наличием на хостинге персональной панели управления. Стандартные функции — управление почтовыми ящиками, статистика, FTP-аккаунты и т.д. Приятно, если глаз радуют всевозможные «примочки» — авось пригодятся. Часто предоставляются услуги по инсталляции уже готовых чатов, форумов, систем статистики и баннерообмена. Это, конечно, не для нас, программистов, однако приятно.

Также очень важно присутствие хорошей, компетентной и, главное, не очень медленной службы поддержки. Проблемы бывают разными, и обратиться за помощью часто больше просто некуда. Мнение о службе поддержки может появиться уже после предварительного общения с ней перед заказом хостинга, но не прогадайте — пока вы не заплатили деньги, вас будут обхаживать и лелеять, а потом бросят на произвол судьбы. Универсального рецепта нет, можно посоветовать только поискать в Сети информацию и отзывы о выбранном вами хостинг-провайдере, почитать подробно форум поддержки, который, конечно, обязательно должен присутствовать у хостинг-провайдера. Просто будьте внимательны, но не перегибайте палку. Не отбрасывайте в сторону тех, о которых говорят плохо. Вполне возможно, это делают конкуренты, чтобы выжить кого-то со своей территории бизнеса. Попробуйте найти других клиентов данного хостинг-провайдера и спросите их мнения. Не думаю, что кто-то откажется, — скажут и хорошее и плохое.

Поинтересуйтесь методами оплаты за предоставленные услуги хостинга и найдите приемлемый для себя. Как правило, это какой-либо из вариантов удаленной оплаты — либо через банк, что долго и дорого, либо при помощи электронных платежей

(например, через популярную систему Webmoney), при использовании которых часто делают скидки, и к тому же оплату можно произвести в течение нескольких минут, не отрываясь от компьютера.

Кстати, хочу сказать несколько слов о системе Webmoney. Часто приходится слышать, что системы электронных платежей не являются надежным средством для того, чтобы рассчитываться за товары или услуги, потому что могут обанкротиться или украсть все деньги. Для таких приведу всего два убедительных аргумента: первый — за каждую транзакцию (перевод виртуальных денег) с вашего счета снимаются небольшие проценты. Но в масштабе всех пользователей это огромная сумма, и гораздо выгоднее продолжать работу, а не обмануть всех, И второе — в принципе невозможно украсть сразу все деньги из платежных систем. Они до тех пор являются такими, пока их товарные знаки (виртуальные деньги) можно обменять на наличные. А если все деньги присвоить, соответственно нигде никто их не примет к обмену, так что это просто невозможно и бессмысленно.

Из собственного опыта могу сказать: система Webmoney работает наиболее эффективно и безукоризненно. Рекомендую всем освоить данный несложный тип платежной системы.

Итак, подведем итог изложенным мыслям. Самое главное — не торопиться и не предъявлять слишком завышенных требований. Помните: идеального хостинга не существует, у каждого есть какой-то недостаток или даже не один. Нужно просто стараться, чтобы эти недостатки причиняли как можно меньше неудобства. И конечно, достоинств должно быть гораздо больше, чем недостатков.

Часть III
Знакомьтесь — PHP

История создания PHP

Язык PHP появился осенью 1994 года. Его создатель — Расмус Лердорф (Rasmus Lerdorf) (рис. 11) — использовал язык в своих целях, чтобы иметь представление о тех людях, которые посещают его сайт и знакомятся с его резюме.

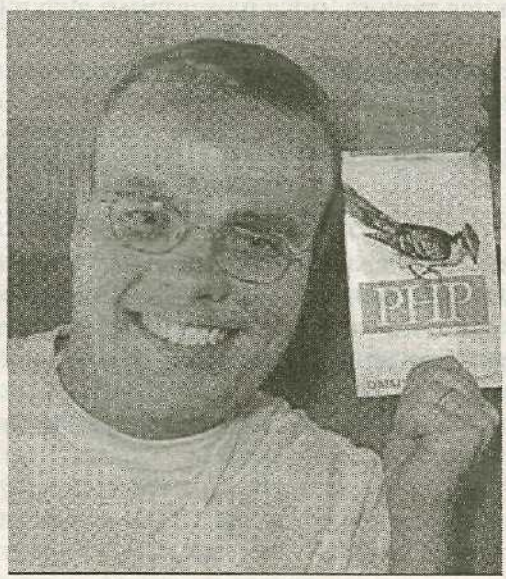


Рис. 11. Создатель PHP — Расмус Лердорф

По словам разработчика языка, PHP был написан буквально за день в промежутках между деловыми встречами. Сначала это

была просто невзрачная CGI-оболочка[1], написанная на языке Perl, которая служила исключительно для специфических целей. Такое приложение (его называют шлюзом, или CGI-программой) запускается сервером в реальном режиме времени. Сервер передает запросы пользователя CGI-программе, обрабатывающей их, и возвращает результат работы программы на экран пользователя. Таким образом, посетитель получает динамическую информацию, которая может изменяться в результате влияния различных факторов. Сам шлюз может быть написан на различных языках программирования — C/C++, Fortran, Perl, TCL, Unix Schell, Visual Basic, Apple Script и др. Создатель PHP для разработки шлюза сначала выбрал язык Perl, как наиболее простой и доступный.

В результате дальнейшей эксплуатации выяснилось, что CGI-оболочка обладает малой производительностью (медленно работает), и Расмус Лердорф вынужден был переписать все заново, но уже на языке C, что позволило увеличить скорость работы PHP. Пользователи сервера, на котором располагался сайт с первой версией PHP, заинтересовались этим языком. Лердорф не предполагал, что кто-то другой будет пользоваться этим языком, но PHP довольно быстро перерос в самостоятельный проект, и в начале 1995 года вышла первая известная версия продукта, называвшаяся Personal Home Page Tools (средства для персональной домашней страницы). На тот момент PHP обладал более чем скромными возможностями. Он имел простейший анализатор кода, который понимал несколько специальных команд, а также разные утилиты для сайта, необходимые для разработки гостевой книги, счетчика посещений, чата, системы статистики и т.п. К середине 1995 года язык был основательно переработан, а также появилась возможность обработки форм и были добавлены функции работы с базами данных. В таком виде вышла вторая версия продукта. Затем была более дополненная третья и, наконец, современная нам четвертая[2] версия PHP.

Сегодня PHP — это мощный кроссплатформенный набор средств, который располагается на сервере и предназначен для обработки

[1] CGI (Common Gateway Interface, общий шлюзовой интерфейс) — стандарт, предназначенный для создания серверных приложений. *(Примеч. ред.)*

[2] К моменту выхода книги была готова версия PHP 5.0 beta 4. *(Примеч. ред.)*

специального кода, встраиваемого в HTML-страницу. Благодаря этому появилась возможность легко создавать динамические сайты. Файлы, созданные таким образом, хранятся и обрабатываются на сервере. Когда посетитель запрашивает документ с PHP-кодом, скрипт обрабатывается не браузером посетителя, как, например, при использовании JavaScript, а сервером (точнее, сервер передает управление специальной программе, обрабатывающей PHP-код). Посетителю передаются уже только результаты работы. Точно так же работает CGI-программа, написанная на C или Perl.

Но в отличие от CGI, PHP-код можно встраивать в любое место HTML-страницы, что является основным преимуществом PHP по отношению к CGI. Кроме того, PHP очень прост для изучения и не требует каких-либо специфических знаний. Например, мне вполне хватило опыта, приобретенного лет десять назад на уроках информатики в школе, на которых мы изучали язык Basic на очень модных и дорогих тогда Yamaha.

Несмотря на столь радужную характеристику, есть у PHP и недостатки. Стоит отметить довольно медленную, по сравнению с CGI-программами, работу больших (именно больших, так как маленькие скрипты не вызывают существенной нагрузки) PHP-скриптов. PHP — интерпретируемый язык[1], что непременно ведет к ухудшению производительности в случае очень больших и сложных программ, но для выполнения несложных манипуляций на сайте PHP — лучший выбор. К тому же последние версии PHP практически лишены этих недостатков. Полностью переписанный и оптимизированный код сделал свое дело, и если ваш хостинг-провайдер позволяет использовать PHP версии более 4.1 — будьте спокойны. Этому очень способствовал тот факт, что частично PHP стал компилируемым языком. Но только частично, в нем очень гармонично уживаются и интерпретатор, и ком-

[1] Интерпретатор — транслятор (программа или устройство, которое переводит программу с одного языка программирования на другой), анализирующий команды или операторы исходной программы и немедленно выполняющий их. Таким образом, интерпретатор одновременно и транслирует, и выполняет заданную программу в отличие от компилятора, который только транслирует всю программу без ее выполнения. *(Примеч. ред.)*

пилятор. Недаром к середине 2000 года PHP использовался более чем на 2,5 млн сайтов.

Установка PHP

Прежде всего надо скачать PHP. Идем на официальный сайт http://www.php.net и ищем там раздел «downloads» (рис. 12).

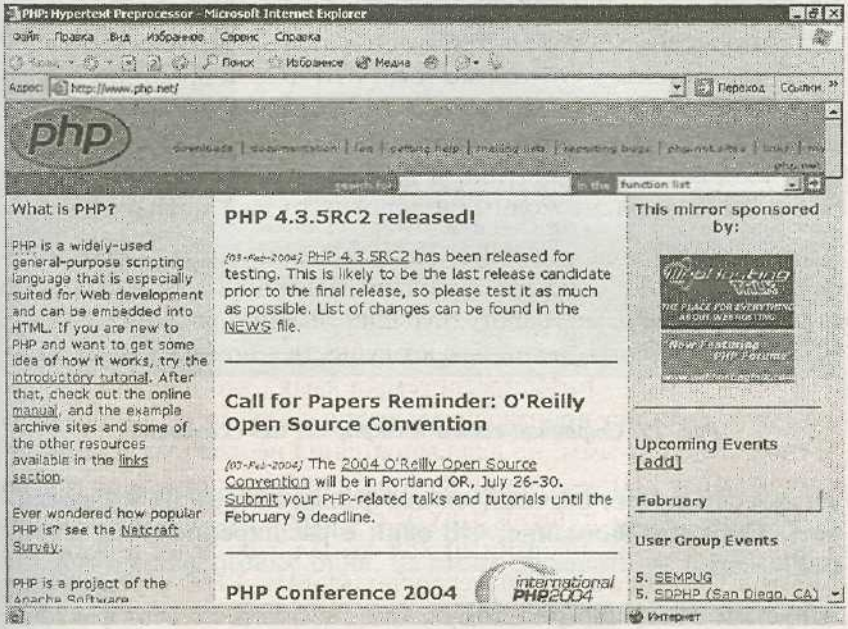


Рис. 12. Главная страница сайта www.php.net

После того как откроется страница, скачиваем два файла из раздела Windows Binaries (рис. 13).

Решайте, какую версию скачать в зависимости от того, какая установлена у вашего хостинг-провайдера. Лучше всего, конечно,

скачивать более свежую версию, так как в ней наверняка исправлены старые ошибки.

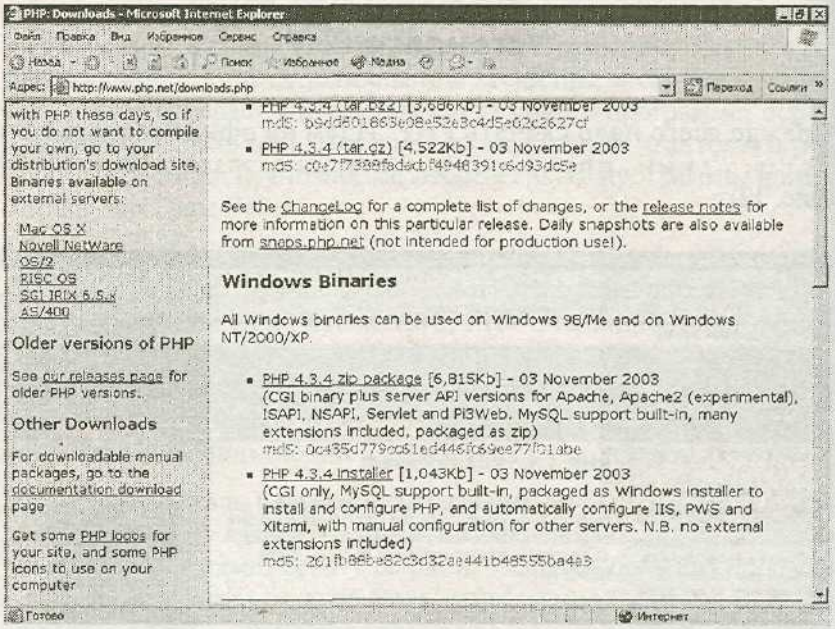


Рис. 13. Страница сайта www.php.net для скачивания

Итак, если вы уже скачали эти два файла, начнем их устанавливать. Обратите внимание, что один с расширением exe, а второй — zip.

Запустите exe-файл (рис. 14).

По традиции нажимайте «Next», соглашайтесь с условиями лицензии и выбирайте тип установки Standard. Далее необходимо выбрать директорию. Как вы помните, у нас есть специальная папка для работы — usr. Указываем PHP путь e:\usr\php\ и устанавливаем его туда. Придется еще ввести адрес SMTP-сервера и свой адрес электронной почты. Введите туда что-нибудь. Вероятнее всего, при работе на домашнем компьютере вам это не по-

надобится. Все равно отправлять почту вы будете уже в интернете, а там свои настройки.

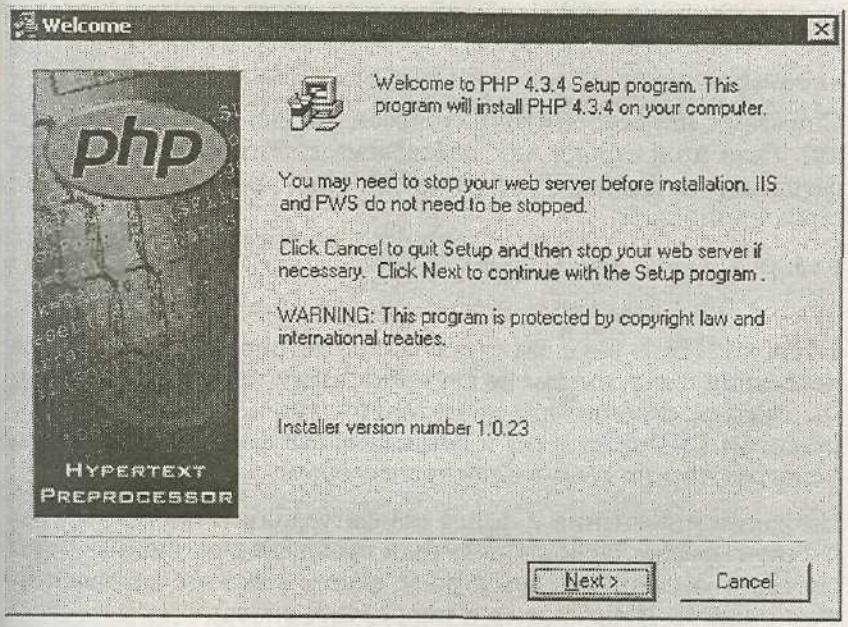


Рис. 14. Окно установки PHP

Еще надо выбрать сервер, с которым будет работать PHP. Как вы помните, это Apache. Если в процессе установки появятся какие-нибудь диалоговые окна, не стоит обращать на них особого внимания.

Теперь осталось установить дополнительные модули и научить сервер Apache распознавать PHP-код.

Настройка PHP для Apache

Откройте конфигурационный файл Apache httpd.conf из папки E:/usr/apache/conf при помощи любого текстового редактора, например «Блокнота» или Word.

Видите, в нем много текста, перед которым стоит символ «#». Это — комментарии. Как правило, они на английском, но есть варианты русского Apache.

Ищем следующую строку:

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

Снимаем с нее комментарий (убираем символ «#» в начале строки). Если этой строки нет, добавляем ее. Это позволит серверу распознавать файлы с расширением php. Сразу же после этой строки допишите:

```
ScriptAlias /_php/ "путь к PHP/"
Action application/x-httpd-php "/_php/php.exe"
```

«Путь к PHP» — в нашем случае это e:/usr/php/. Не забудьте про последний слеш, он там не случайно. Этим нехитрым способом мы создаем синоним для директории с установленным препроцессором PHP (php.exe) и связываем все файлы с PHP-кодом непосредственно с самим препроцессором.

Сохраните изменения в файле конфигурации и попробуйте запустить Apache. Если вы получили сообщение об ошибке, просмотрите указанную в нем строку, скорее всего это синтаксическая ошибка. Исправьте ее и попробуйте снова.

Будем считать этот этап пройденным.

Виртуальные хосты в Apache

Как я уже говорил, серверы в Сети устроены таким образом, что на одном компьютере могут быть десятки и сотни виртуальных серверов. Нам придется организовать что-то подобное у себя на компьютере, чтобы иметь возможность разрабатывать дома не один сайт, а сколько угодно. Мы научимся делать два, остальные вы сможете добавить аналогично.

Опять открываем знакомый нам файл конфигурации httpd.conf и добавляем в конец файла строки:

```
<VirtualHost 127.0.0.1>
ServerAdmin admin@servername.com
```

```
ServerName www.servername.com

DirectoryIndex index.php3 index.phtml index.php
index.htm index. html index.shtml index.shtm

DocumentRoot "e:/usr/public_html/host/virtual"

ScriptAlias /cgi-bin/ "e:/usr/public_html/host/virtual/
cgi-bin/"

ErrorLog e: /usr/public_html/host/virtual/logs/error. log

CustomLog e:/usr/public_html/host/virtual/logs/access.log
common

</VirtualHost>

VirtualHost 127.0.0.2>

ServerAdmin admin@my-site.com

ServerName www.my-site.com

DirectoryIndex index.php3 index.phtml index.php index.htm
index.html index.shtml index.shtm

DocumentRoot "e:/usr/public_html/host/start"

ScriptAlias /cgi-bin/ "e :/usr/public_html/host/virtual/
cgi-bin/"

ErrorLog e:/usr/public_html/host/virtual/logs/error.log

CustomLog e:/usr/public_html/host/virtual/logs/access.log
common

</VirtualHost>
```

И таким же образом дальше, по мере необходимости. Обратите внимание на различия. В строке с VirtualHost указываются разные IP-адреса, по которым будет осуществляться доступ к выбранным сайтам. Имя сервера и почтовый ящик администратора — разные. В строчке с DocumentRoot надо указать путь к директории, в которой будут храниться файлы конкретного сайта.

В каталоге virtual нужно создать еще один каталог — LOG для log-файлов Apache. Сами файлы создавать вручную не надо, они будут созданы Apache автоматически.

СОВЕТ

> Если вы не очень нуждаетесь в log-файлах, можно периодически их удалять. Это позволит серверу работать немного быстрее, да и место сэкономит. Сделать это можно только тогда, когда сервер остановлен. Log-файлы можно сохранять в разные папки для каждого сайта отдельно, но если они не очень нужны, можно задать в одну папку, чтобы легче было удалять.

Как обычно, сохраните файл конфигурации и попробуйте запустить Apache. Если ошибка — ищите ее в указанной строке.

Тестирование PHP

Убедимся, что PHP-скрипты работают. Для этого создадим в директории d:/usr/public_html/host/virtual файл test.php со следующим содержанием:

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

Теперь наберите в браузере: `http://127.0.0.1/test.php`. Должна отобразиться страница с разнообразной информацией о PHP, которая генерируется функцией `phpinfo()` (рис. 15).

Все, PHP работает.

Установка и настройка дополнительных модулей

Обратите внимание на второй файл, который мы скачали. Он имеет расширение zip и представляет собой обычный архив. Распакуйте его. Найдите в нем директорию extensions и скопируйте ее полностью к себе в каталог с установленным PHP.

Это — дополнительные модули, которые нужны для работы с изображениями, службами улучшенной криптографии и т.д.

А если и не нужны сразу, то некоторые готовые разработки могут их использовать, так что пусть будут.

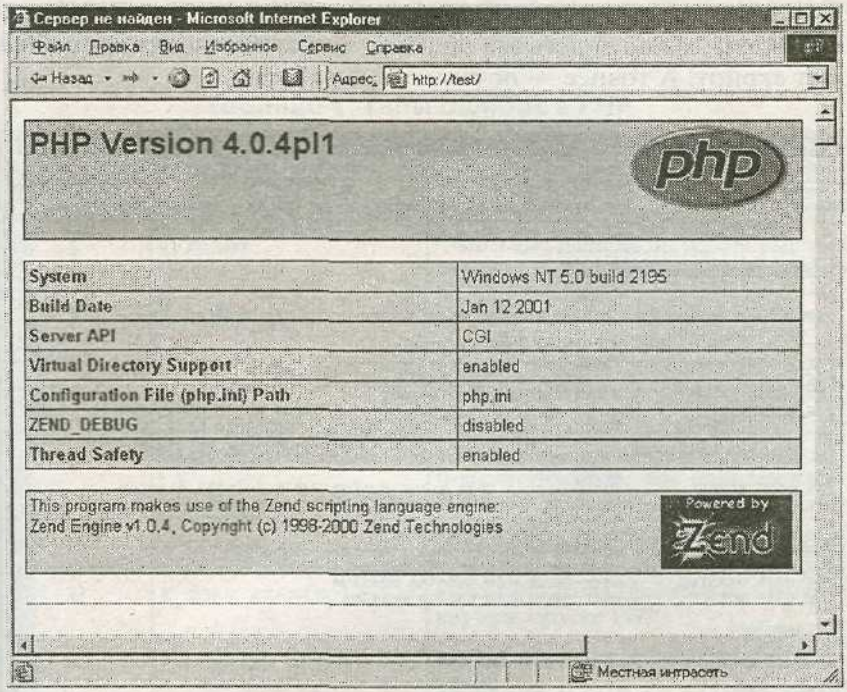


Рис. 15. Страница информации о PHP, сгенерированная функцией phpinfo()

Теперь надо подключить новые модули к PHP. Делается это во втором важном конфигурационном файле — php.ini. Готового его варианта нет, придется создать самостоятельно. Надо отметить, что и без этого файла PHP сможет работать, но у нас не будет возможности настраивать нужные параметры.

Итак, в дистрибутиве ищем файл php.ini-recommended и переименовываем его в php.ini. Редактировать можно, как обычно, в любом текстовом редакторе, так как он, как и конфигурационный файл Apache, представляет собой простой текст.

Открыли? Ищите в нем раздел Error handling and logging (рис. 16).

Этот раздел отвечает за настройку ошибок, выводимых на экран. Нам очень важно видеть эти ошибки, чтобы понимать, как работает скрипт. А точнее — почему он не работает.

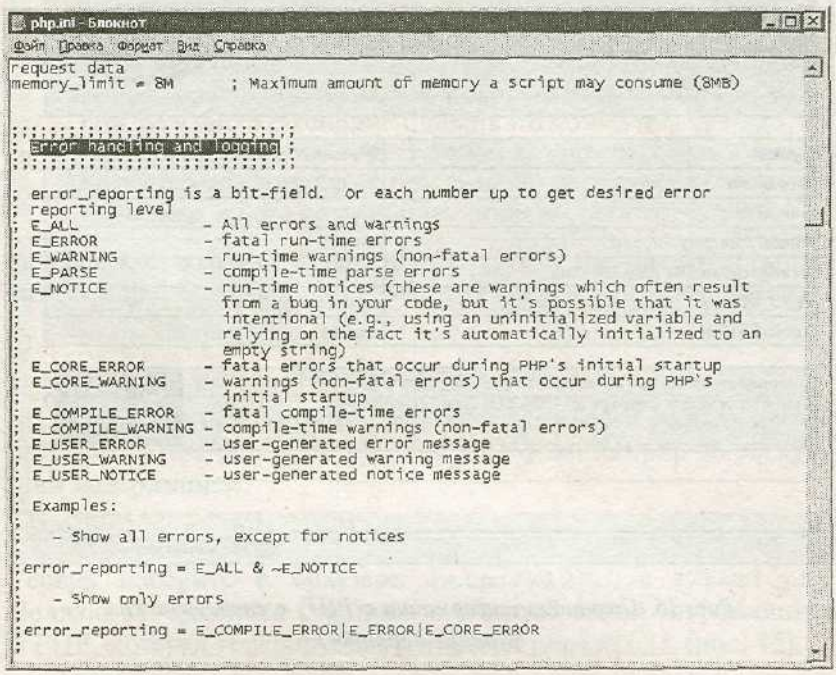


Рис. 16. Раздел Error handling and logging в файле php.ini

Здесь чуть ниже будет переменная `error_reporting = E_ALL`. Конечно, можно оставить ее такой, какая она есть, но тогда приготовьтесь получать сообщения об ошибках по поводу и без повода. Например, если происходит попытка использования неинициализированной переменной (см. ниже). Это не страшно, так как в этом случае переменная автоматически инициализируется и приравнивается или к нулю, или к пустой строке, в зависимости от типа, но сообщение об ошибке все равно появится на экране. Чтобы избежать таких ситуаций, надо прописать тут: `error_reporting = E_WARNING`. В табл. 1 приведен полный

список доступных значений: вы можете выбрать более подходящие для вашего сайта.

Таблица 1.Типы ошибок в PHP

Значение	Константа	Описание
1	E_ERROR	Фатальные ошибки на стадии выполнения
2	E_WARNING	Предупреждения на стадии выполнения (нефатальные ошибки)
4	E_PARSE	Ошибки анализа на стадии компиляции
8	E_NOTICE	Уведомления на стадии выполнения (менее серьезные, чем предупреждения)
16	E_CORE_ERROR	Фатальные ошибки при запуске PHP
32	E_CORE_WARNING	Предупреждения (нефатальные ошибки) при запуске PHP
64	E_COMPILE_ERROR	Фатальные ошибки на стадии компиляции
128	E_COMPILE_WARNING	Предупреждения на стадии компиляции (нефатальные ошибки)
256	E_USER_ERROR	Сообщение об ошибке, генерируемое пользователем
512	E_USER_WARNING	Предупреждение, генерируемое пользователем
1024	E_USER_NOTICE	Уведомление, генерируемое пользователем
2047	E_ALL	Все вышеуказанное

Еще чуть дальше нужно найти параметр `display_errors = Off` и установить его в On, включив, таким образом, выдачу сообщений об ошибках именно на экран, а не в log-файлы. Там просто неудобно искать эти ошибки.

В разделе Data Handling переменную register_globals = Off установим в On (рис. 17). Так мы сделали доступными для считывания переменные окружения[1] непосредственно по их имени, а не через специальный массив переменных окружения.

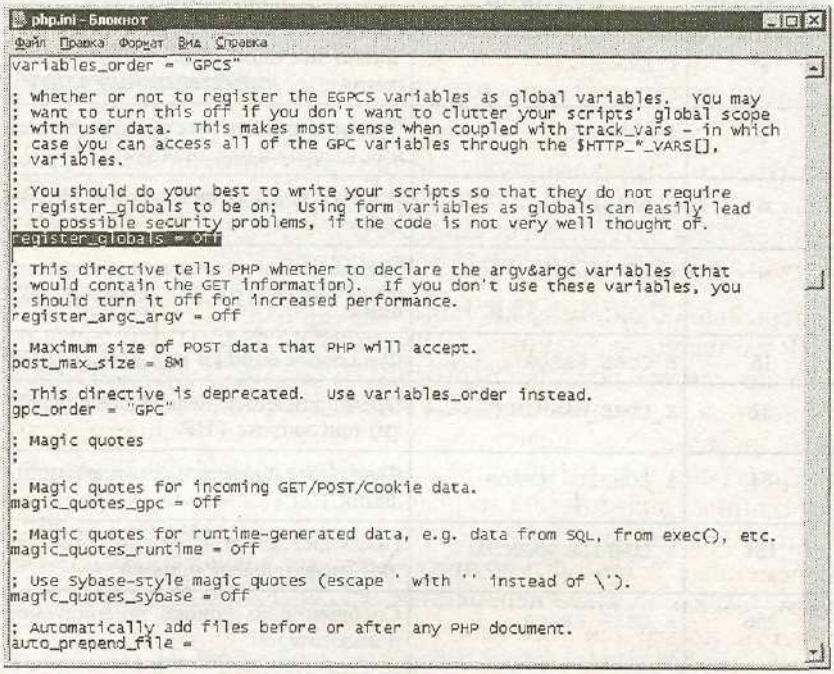


Рис. 17. Настройка переменных окружения в файле php.ini

Теперь нужно указать путь к папке, в которой находятся наши модули расширения. Он задается в переменной extension_dir. Задайте его правильно, в соответствии с тем, где именно находится у вас папка с модулями. Если этот путь указан неправиль-

[1] Переменные окружения — это параметры операционной системы, содержащие такие сведения, как имя диска, путь или имя файла. Например, переменная окружения TEMP задает папку, в которой хранятся временные файлы программ. *(Примеч. ред.)*

но, при загрузке ненайденного модуля PHP будет выдавать ошибку.

ВНИМАНИЕ

Полученный таким образом файл php.ini надо скопировать в директорию, в которой установлена Windows (как правило, одноименная директория в корневом каталоге диска C). Именно там PHP будет искать этот файл. В последних версиях PHP его можно сохранить в директорию с установленным сервером Apache.

Учтите, что загрузка и использование дополнительных модулей может в значительной степени снижать производительность интерпретатора PHP. Лучше всего использовать в работе только те модули, которые нужны при работе сайта.

Теперь еще об одном важном моменте. Во всех руководствах по PHP написано, что данные, полученные из формы или переданные по ссылке вот так: name.phtml?a=1&b=2, автоматически становятся переменными PHP $a и $b. На самом деле это может быть не совсем так. Дело в том, что в целях безопасности, начиная с версии 4.1, PHP настраивается по умолчанию так, чтобы переданные значения не назначались переменным, потому что таким образом легко уничтожить уже существующие в скрипте переменные и взломать скрипт. Хотя это является сомнительным, так как нужные переменные всегда можно отследить и защитить, но все-таки это факт.

За назначение переменных отвечает параметр register_globals в php.ini. Если register_globals = On, то все полученные скриптом данные будут назначены соответствующим переменным. Если register_globals = Off, то получить значение переменной можно, обратившись к массиву, соответствующему способу передачи данных в скрипт.

На этом установку можно считать завершенной. Осталось поговорить о некоторых удобствах.

Переносимость и совместимость

Один раз настроив свой сервер, хочется знать, как избежать данной процедуры каждый раз при переустановке системы. Такая возможность есть.

Если ваш сервер, PHP и сайты находятся в одном каталоге, перед переустановкой системы надо сделать резервную копию этого каталога. Этим вы застрахуете себя от возможных проблем.

После того как система переустановлена, нужно восстановить только два файла — php.ini, чтобы восстановились все настройки PHP (если они не делались, забудьте об этом файле, PHP будет работать с установками по умолчанию), и hosts, который находится по адресу E:\WINDOWS\system32\drivers\etc (рис. 18). Это служебный файл Windows, который нужен для сопоставления имен. Его местоположение меняется в зависимости от версии Windows, но вы можете воспользоваться поиском, чтобы найти этот файл.

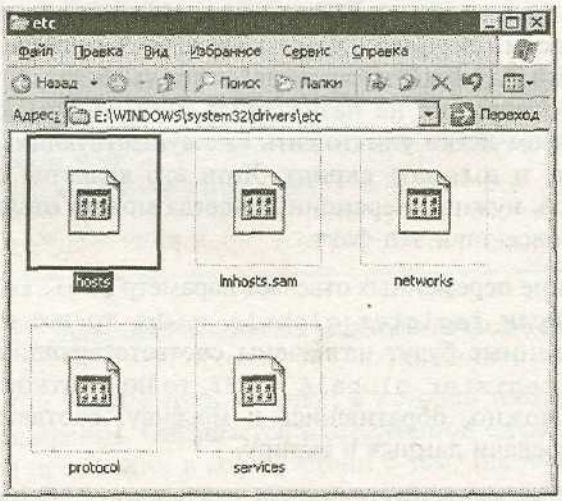


Рис. 18. Файл hosts

Вот пример содержимого файла hosts с моего компьютера:

```
127.0.0.1 localhost
127.0.0.6 204040
127.0.0.2 virtual
127.0.0.3 price
127.0.0.4 start
127.0.0.5 test
```

Как видите, он просто позволяет обращаться к сайту, используя имена вместо цифр. Смело редактируйте его в соответствии со своими потребностями. И, конечно, храните резервные копии.

Итак, если диск не меняется, достаточно восстановить эти два файла. Если же по каким-то причинам есть необходимость сменить диск, на котором будет работать сервер, надо открыть файл httpd.conf, расположенный в папке e:/usr/apache/conf. Откройте этот файл, например, в Word, при помощи поиска и замены произведите замену имени старого диска на новый (рис. 19), и после этого сохраните файл.

То же самое проделайте и с файлом php.ini, о нем мы говорили чуть выше.

Можно протестировать работу сервера в новых условиях и, если все в порядке, приступать к программированию, если нет — искать ошибку на основе информации, которую выдает окно DOS-сессии Apache, Как правило, одну за другой все ошибки устранить не сложно.

Теперь, когда все готово к работе, можно приступить непосредственно к изучению языка программирования PHP.

Синтаксис PHP

Как и у всякого языка программирования, у PHP есть свой синтаксис. Он очень похож на синтаксис языков C и Perl. Программисты, пишущие на этих языках, смогут освоить PHP буквально за несколько дней. Но даже если вы никогда не программировали,

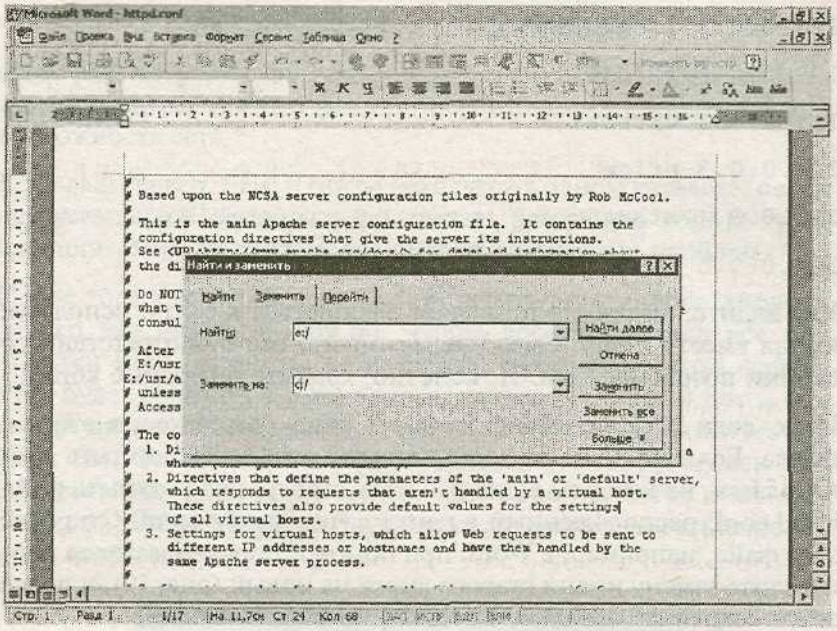


Рис. 19. Редактирование файла httpd.conf

PHP поддастся вам легко и обеспечит базу для перехода на языки более сложного уровня. Все команды и правила просты. Синтаксис включает в себя операторы, разделенные между собой точкой с запятой.

ВНИМАНИЕ

> Одна из основных ошибок начинающих программистов — отсутствие точки с запятой между операторами. Список других распространенных ошибок смотрите в приложении 2.

К счастью, ошибки в PHP по умолчанию выводятся на экран монитора (в отличие от CGI, где все ошибки записываются в log-файл, что, согласитесь, не очень удобно при анализе программы), и найти ошибку при определенной внимательности и опыте не составит большого труда. Тем более что умный интерпретатор подскажет номер строки, в котором произошла ошибка.

Чтобы сервер знал, что в файле есть PHP-код, расширение файла нужно сделать либо phtml, либо php3, либо php. Вообще, может быть назначено любое из этих расширений, но я рекомендую в целях совместимости всегда использовать phtml.

Каждый скрипт в PHP начинается с <?php и заканчивается ?>.

В любом месте скрипта PHP можно поместить комментарий, начинается он с /*, а заканчивается — */ Если комментарий небольшой и занимает только одну строку, можно поставить перед ним // и таким образом закомментировать любую строку до ее конца. Как обычно, пробелы, символы табуляции и перевод строки просто игнорируются и могут применяться для улучшения читабельности кода PHP.

СОВЕТ

В коде PHP удобно прятать комментарии к собственной программе и пояснения для себя. Если в случае с HTML комментарии вместе с кодом попадают к посетителю в браузер, то PHP свой код никак не отдает, а потому и комментарии вроде есть, а посетителю они не передаются.

Теперь напишем наш первый скрипт. Собственно, это не будет программой в полном смысле слова, но для тренировки подойдет. Наберите команду:

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

Сохраните с расширением phtml и запустите этот файл в браузере, обратившись к нему при помощи адреса 127.0.0.1/имя файла .phtml. Удивлены? Не ожидали так много разной информации?

Phpinfo — это функция для получения информации о возможностях и настройках PHP. Она позволяет получить данные даже на удаленном сервере, т.е. вам доступна эта информация без обращения к администратору сервера. Только для этого нужен доступ к серверу по протоколу FTP.

В приложении 3 приведены некоторые другие наиболее распространенные функции PHP.

Возможности PHP

Работа с базами данных

Использование PHP для создания Web-страницы, работающей с базой данных, невероятно просто.

На момент написания книги поддерживаются следующие базы данных: Oracle, Adabas D, Sybase, FilePro, mSQL, Velocis, MySQL, Informix, Solid, dBase, ODBC, Unix dbm, PostgreSQL. Несомненно список будет постоянно расширяться разработчиками PHP. Я не сомневаюсь, что если у вас есть на сервере база данных, то это одна из этих перечисленных. Впрочем, если нет, не беда, потому что PHP с легкостью позволяет сделать эмуляцию базы данных, делая запись непосредственно в файлы. Лишь бы места хватило.

HTTP-аутентификация средствами PHP

Аутентификация позволяет выполнить вход в зону, закрытую паролем, и доступна только при использовании модуля Apache. При использовании специальной функции вызывается диалоговое окно, позволяющее ввести логин и пароль, а в дальнейшем легко проверить введенные данные. Но на самом деле может оказаться проще и удобней самостоятельно организовать закрытую зону и предоставить для авторизованных посетителей к ней доступ.

Работа с изображениями

PHP может обрабатывать не только текст и HTML-код, но и работать *с* изображениями GIF, JPEG или PNG. Для этого вам надо скомпилировать PHP с библиотекой функций изображения — GD.

Поддержка закачки файлов

Позволяет реализовать закачку файлов на сервер пользователем. Пригодится, например, для обмена музыкой, рефератами, про-

граммами и т.д. Кроме того, PHP предоставляет полный контроль над загружаемым файлом, в том числе есть ограничения на размер, а также возможность управлять загруженным файлом.

Поддержка HTTP-cookie

Cookie — механизм для сохранения данных (небольших файлов до 4 Кб с информацией) в удаленном браузере и, таким образом, отслеживания или идентификации пользователей (см. приложение 4). Вы можете устанавливать файлы cookie, используя функцию `setcookie()`. Cookie — часть HTTP-заголовка, поэтому функция `setcookie {)` должна вызваться прежде, чем браузеру будет послана какая-нибудь информация для вывода, вплоть до пробела. Любой cookie, посланный вам от клиента, будет автоматически превращен в PHP-переменную.

Использование регулярных выражений

Регулярные выражения используются для сложного манипулирования строками в PHP. На мой взгляд, это одна из самых полезных возможностей PHP. На основе этих функций со строками можно сделать все, что угодно.

Обработка ошибок

Реализована как на глобальном, так и на локальном уровне. Вы можете вообще отключить вывод сообщений об ошибках на экран пользователя (полезно в готовом рабочем варианте, программы, закачанном на сервер) или обработать функцию таким образом, чтобы ошибка не была отображена, а сообщение об ошибке было отправлено по почте на любой адрес. Хотя, как правило, на это не обращают внимания, и порой в результате неправильных действий такая ошибка выводится в браузер. Программа может получать сведения об ошибочности действий и таким образом принимать решения о своей дальнейшей работе.

Управление электронными письмами

Оно сведено к одной функции. Есть адрес и текст, который нужно отправить на этот адрес. При помощи PHP письмо будет отправлено без проблем. В дальнейшем мы рассмотрим не один пример с поддержкой отправки писем.

На своем личном опыте я могу сказать, что сайт, построенный полностью на PHP, не испытывает каких-либо задержек в открытии и работе, даже несмотря на то, что интерпретатор PHP испытывает значительные нагрузки при размещении на сайте новостей, счетчика статистики, генерации баннеров нескольких типов и «склеивания» страницы из нескольких фрагментов. Это и многое другое будет нормально функционировать при условии элементарной внимательности и качественной настройки сервера.

Конечно, перечисленные возможности языка PHP далеко не полные, но по мере знакомства с ним вы научитесь создавать очень сложные вещи буквально одним движением руки (это я образно, конечно) и узнаете значительно больше.

Вывод на экран и переменные в PHP

PHP очень легко позволяет организовать вывод текста на экран. Рассмотрим пример скрипта:

```
<?php

echo "Привет, мир!";

?>
```

СОВЕТ

Не обязательно открывающий тег PHP писать именно так: <?php. Вполне допустим вариант <?, хотя это и не по стандарту. Но если уже давно поддерживаются оба варианта, почему бы второму тоже не стать стандартом? Выбирайте то, что вам больше нравится.

Этот скрипт может быть расположен в любом месте HTML-документа, и сам по себе он не несет ничего полезного, так как просто выводит на экран фразу «Привет, мир!». Но таким образом мы знакомимся с одной из наиболее распространенных функций PHP — echo, которая выводит информацию на экран пользователя.

Чтобы разобраться в работе скрипта, давайте познакомимся с переменными.

Переменная характеризуется именем, типом и значением. Имя может быть любым и включать в себя цифры, буквы английского алфавита и разрешенные символы (например, символ подчеркивания или тире). Все переменные в PHP обязательно должны начинаться со знака $, что позволяет интерпретатору безошибочно отличать переменные от функций. Давайте ясные и по возможности «говорящие» имена своим переменным, однако не слишком длинные, максимум из двух слов. Разделить два слова можно, поставив «_» (символ подчеркивания) или написав второе слово с заглавной буквы.

ВНИМАНИЕ

Названия функций в PHP не зависят от регистра (т.е. ECHO то же самое, что echo), а имена переменных — регистрозависимы (т.е. $os и $OS — разные переменные).

По типу переменные делятся на целочисленные, с плавающей запятой, строковые, объектные, булевы и массивы.

Заранее описывать тип переменной, как в языках Pascal или Visual Basic, не требуется. Разделение на типы в принципе условное, и все же каждая переменная автоматически стремится использовать правильный тип, соответствующий своему значению.

ВНИМАНИЕ

Если мы попытаемся использовать неинициализированную переменную в работе, это посчитается интерпретатором за ошибку. Но это не является ошибкой, так как переменная тут же будет

инициализирована и ее значение будет приравнено к нулю или пустой строке в зависимости от типа.

Однако это справедливо, только если уровень обработки ошибок не выставлен на максимальный контроль. Если же уровень ошибок максимален, обращение к несуществующей переменной вызовет сообщение об ошибке (программа при этом будет работать дальше). Чтобы подавить такое сообщение об ошибке, можно заранее описать все использующиеся в программе переменные, присвоить им значение «ноль» или «пустая строка». Второй вариант подавления ошибки: перед именем переменной поставить символ @, который подавляет сообщения об ошибках. Вот так: @$name; if (@$name) и т.д. Для постоянного подавления сообщений о таких ошибках нужно соответствующим образом настроить конфигурационный файл php.ini (см. **выше**).

Значение в соответствии с типом может быть практически любым. Например, $a = 5. Это говорит о том, что имя переменной — $a, тип — целочисленный, значение — 5. Еще примеры:

```
1 <?php
2 $name = 6;
3 $hl2 = 4.89;
4 $file_type = "path/index.phtml";
5 $os = "PHP для всех!";
6 ?>
```

Во второй строке нашего скрипта переменной $name присваивается значение 6, и эта переменная автоматически становится целочисленной.

В третьей строке кода переменной $hl2 присваивается значение 4.89, которое имеет тип числа с плавающей запятой. В четвертой и пятой строках кода переменным присваиваются значения со строковым типом. Все, что заключено в кавычки (включая цифры), будет интерпретировано как символьная строка.

Простейшие арифметические операции

Как и в других языках программирования, над переменными можно совершать любые арифметические действия в соответствии с обычной логикой. Достаточно указать переменную для результата, знак равенства и перечислить в порядке выполнения действий переменные или значения с необходимыми арифметическими знаками. Пример:

```
<?php
$a = 5;
$b = 3;
$c = 4;
$d = $a + $b - $c;
echo $d;
?>
```

Результат работы скрипта — вывод на экран цифры 4,

PHP поддерживает все математические функции, многоуровневые скобки, логические операции, операции увеличения или уменьшения на единицу и многое другое.

В PHP есть функции для увеличения или уменьшения переменной на единицу. Для этого нужно указать имя переменной и за ним без знака равенства подряд два плюса или минуса соответственно. Например, $a++; — переменная $a будет увеличена на единицу.

Нескольким переменным можно одновременно присвоить одно значение: $a = $b = 4; — обе переменные и $a и $b будут равны четырем. Вот еще несколько примеров нестандартных арифметических операций в PHP:

```
<?php
$b = $a = 5; /* присваиваем одинаковое значение переменным $a и $b */
```

```
$c = $a++; /* последующее увеличение, присваиваем $c начальное значение $a (5), а затем увеличиваем $a на единицу */

$e = $d = ++$b; /* предварительное увеличение, присваиваем переменным $d и $e значение $b, увеличенное на единицу, т.е. $d и $e равны 6 */

$f = 2 * ($d++); /* присваиваем переменной $f удвоенное значение переменной $d до его увеличения, т.е. 2 * 6 = 12, и затем увеличиваем $d на единицу */

$g - 2 * (++$e); /* присваиваем переменной $g удвоенное значение переменной $e после его увеличения, т.е. 2 * 7 = 14 */

$h = $g += 10; /* сначала увеличиваем значение $g на 10, что дает в результате 24, а затем присваиваем это значение переменной $h */

?>
```

Рассмотрим следующий работоспособный скрипт, который вы можете вставить к себе на страницу. Задача: вывести на экран через пробел определенное количество последовательных чисел. В нашем случае это 1, 2, 3, 4, 5, т.е. пять последовательных чисел, начиная с единицы. Код решения:

```
<?php
$i = 1; // присваиваем переменной $i значение 1

echo $i; // выводим переменную $i

$i++; // увеличиваем переменную на единицу

echo " ".$i; // выводим увеличенную на единицу переменную, не забыв о пробеле

// повторяем эти две операции необходимое количество раз

$i++;
echo " ".$i;
```

```
$i++;
echo " ".$i;

$i++;
echo " ".$i;

?>
```

Пример, конечно, не самый лучший[1], но основные понятия языка уяснить можно.

Увеличение переменной на единицу реализуется указанием двух плюсов после переменной — $i++, хотя ничего не мешает писать так — $i = $± + 1;. Выполнив эту инструкцию пять раз, соответственно увеличим переменную на пять, выводя между делом результат и не забывая о пробелах. На экран в результате работы скрипта будут выведены подряд (в строку) цифры «1 2 3 4 5».

Если бы стояла задача вывести указанные цифры не в строчку, а в столбец, то вместо пробелов в кавычках надо было бы поставить HTML-тег перевода строки
. Вот так — echo "
".$i;. В результате на экране появится столбик из пяти цифр.

Теперь рассмотрим алгоритм этого скрипта:

- начать скрипт;
- присвоить переменной $i значение 1;
- вывести переменную $i на экран;
- увеличить переменную $i на единицу;
- вывести пробел и переменную $i на экран;
- выполнять столько, сколько надо;
- закончить скрипт.

Как видите, ничего сложного нет. Просто делаем то, что нам надо.

А теперь посмотрите, что получится в HTML-коде: ни следа PHP! Это ли не мечта многих — иметь возможность скрыть свой HTML-код.

---

[1] Для решения данной задачи лучше было бы воспользоваться циклом (см. ниже), но на первых порах для знакомства я буду стараться все максимально упрощать.

Простейшие логические операции

Очень просто организуется сравнение: «если — то — иначе». Для этого в PHP применяется конструкция:

```
if ( ) { } else { }
```

Есть различные варианты синтаксиса этого оператора, но этот — основной и самый логичный из всех. Вот его расшифровка в переводе на русский (а точнее, на алгоритмический) язык:

(если) if (условие) (то) {выполняется то, что заключено в эти фигурные скобки} (иначе) else {выполняется то, что заключено в эти фигурные скобки}

После фигурных скобок ставить точку с запятой, как обычно между операторами, не обязательно. Но внутри фигурных скобок — разделение операторов между собой проводится только через точку с запятой, если только там не один оператор. Если же оператор один, то и круглые скобки могут не ставиться.

СОВЕТ

Я не рекомендую сильно увлекаться разными вариантами написания. Можно запутаться и не заметить элементарной ошибки. Лучше пользоваться стандартным, понятным и удобочитаемым способом, что позволит избежать глупых ошибок.

Допускается вложение нескольких операторов условия один в другой. В этом случае надо быть очень внимательным к количеству закрывающих фигурных скобок, так как при отсутствии даже одной из них интерпретатор выдаст ошибку.

ВНИМАНИЕ

Если вы где-то забыли закрыть скобки или поставили лишнюю скобку (например, в начале кода), интерпретатор выдаст сообщение о том, что ошибка произошла в последней строке кода. Бесполезно искать неточность в этой строке, ищите лучше там, где в последний раз вставляли условный оператор. Найти такую потерявшуюся скобку в большом скрипте бывает очень

сложно, для этого заранее заботьтесь об удобочитаемости скрипта. Это сэкономит не один час отладки в дальнейшем.

Рассмотрим несложный пример:

```
<?php
$a = 5;
$b = 9;
if ( $a == $b ) { echo $b - $a; } else { echo $b.$a; }
?>
```

При проверке истинности применяются два знака равенства для того, чтобы интерпретатор мог отличить сравнение от присваивания.

ВНИМАНИЕ

Программа не будет работать правильно, если вы забыли поставить двойной знак равенства при проверке истинности. И учтите, интерпретатор даже не подумает предупредить вас об этом! Для него это вполне обычная операция присваивания и одновременно проверки условия на истинность-ложность.

Результат работы вышеприведенного скрипта — 95, так как $a не равно $b, а команда echo $b.$a; (между переменными стоит точка, которая служит для объединения результатов в одну строку, а не знак арифметической операции) выводит подряд указанные переменные. Таким образом, точка служит для склеивания строк или переменных.

Проверка ложности обозначается символами !=, допустимы все остальные арифметические, логические символы и операторы (например — or, and, >, <= и т.д.).

Циклы

Циклы в программировании — это повторяющиеся несколько раз операции. Для реализации циклов в PHP используются операторы while, do...while, for и foreach. Начальное значение

указывается в начале цикла, а длительность его выполнения ограничивается каким-либо условием.

Примером цикла может служить копирование нескольких файлов. Алгоритм выполнения этого задания можно описать так:

- задать количество файлов;
- установить счетчик скопированных файлов в ноль;
- скопировать файл;
- проверить, не равно ли значение счетчика заданному количеству файлов;
- если нет — увеличить счетчик скопированных файлов и вернуться к началу цикла (опять скопировать файл);
- если да — закончить цикл.

Каждый проход цикла называется итерацией.

Теперь рассмотрим, как циклы реализуются в PHP:

```
<?php

$i = 0;

$n = 10;

while ($i <= $n):

echo $i."<br>\n";

$i++;

endwhile;

?>
```

Смысл скрипта очень прост. Присваиваем переменной $i значение, соответствующее началу цикла, а переменной $n — значение конца цикла. Далее открываем цикл оператором while (), и внутри его скобок описываем условие, при выполнении которого цикл будет продолжать свою работу. В нашем случае выполнение не прервется, пока $i <= $n. Как только это условие будет

нарушено, управление передастся следующей за циклом операции. Внутри цикла могут быть любые команды PHP (разделенные между собой как обычно — точкой с запятой).

Только нужно следить за тем, чтобы переменная $i, используемая в цикле, была увеличена (и совсем не обязательно на единицу), иначе цикл станет бесконечным, и интерпретатор будет выполнять его, пока не закроется сессия (окно браузера). Оператор endwhile означает конец цикла.

Скрипт, описанный здесь, выводит на экран браузера цифры от 0 до 10. Причем числа будут выведены в столбик, так как в строке функции вывода echo после переменной $i мы указали HTML-тег перевода строки
.

Для примера я привожу еще один, более быстрый вариант выполнения указанной выше задачи.

```
<?php
$i = 0;
while ($i <= 10)
{
echo $i++."<br>\n";
}
?>
```

Удивительно, но при выполнении этих двух примеров получается одинаковый результат, а скрипт практически поместился в одну строчку. Разница в стиле применения оператора цикла и в том, что переменная цикла выводится на экран одновременно с увеличением.

Рассмотрим еще один пример, основанный на применении конструкции PHP do...while. Это тоже цикл, и отличается от while тем, что значение логического выражения проверяется не до, а после окончания работы операторов, включенных в цикл. Таким образом, do...while гарантированно будет выполнен хотя бы один раз, что в случае с while совсем не обязательно (при ис-

пользовании while, если условие ложно, управление сразу будет передано дальше). Для циклов do...while существует только один вид синтаксиса:

```
<?php
$i = 0;

do

{

echo $i."<br>\n";

$i++;

}

while ($i <= 10);

?>
```

Еще один оператор цикла — for. Его синтаксис:

```
for (expr1; expr2; expr3) {последовательность операторов}
```

Первое выражение (expr1) является безусловным и выполняется в начале цикла. В начале каждой итерации выполняется expr2. Если оно истинно (равно true), то цикл продолжается и выполняется вложенный(е) оператор(ы). Если оно ложно (равно false), то цикл заканчивается. В конце каждой итерации выполняется expr3.

Каждое из этих выражений может быть пустым. Если expr2 пусто, то цикл продолжается бесконечно (PHP по умолчанию считает его истинным, как и в языке C). Это не так бесполезно, как кажется, так как зачастую требуется закончить выполнение цикла, используя оператор break в сочетании с логическим условием, вместо использования логического выражения в for. Если внутри любого цикла встречается оператор break, цикл безусловно прекращает выполнение итерации, и управление передается следующей за циклом команде. Например:

```
$a = 0;
while ($a < 5) {
 if ($arr[$a] == "stop")
{
break; /* Выполнение цикла прекращается, если
       в массиве $arr[] есть stop */
}
 $a++;
}
```

Если встречается оператор continue, то управление передается началу следующего ближайшего цикла. Например:

```
while (list($key,$value) = each($arr)) {
if ($key 2)
{
 continue;
}
```

Вот как можно реализовать вывод списка чисел на экран пользователя при помощи оператора for:

```
/* Пример 1 */
for ($i = 1; $i <= 10; $i++)
{ print $i; }
/* Пример 2 */
for ($i = 1;;$i++)
{ if {$i > 10) { break; } print $i; }
/* Пример 3 */
$i = 1; for (;;)
 { if ($i > 10) { break; } print $i; $i++; }
```

```
/* Пример 4 */
for ($i = 1; $i <= 10; print $i, $i++) ;
```

Вот так по-разному при помощи PHP можно реализовать сходные задачи.

Время и дата

В PHP наиболее часто при работе с форматами времени используется функция date. Ее синтаксис: $date = date ("параметр") ; Параметров может быть несколько, разделяются они между собой запятой. Допустимы следующие параметры:

a — может принимать значения "am"[1] или "pm";

A — "AM" или "PM";

d — день месяца, цифровой, две цифры (на первом месте при необходимости ноль), т.е. от 01 до 31;

D — день недели, текстовой, три буквы, например "Fri";

F — месяц, текстовой, длинный, например "January";

h — час, цифровой, 12-часовой формат, две цифры;

H — час, цифровой, 24-часовой формат, две цифры;

i — минуты, цифровой, две цифры, т.е, от "00" до "59";

j — день месяца, цифровой, без начальных нулей;

1 (строчная L) — день недели, текстовой, длинный, например "Friday";

L — указывает, високосный год или нет, т.е. "0" или "1";

m — месяц, цифровой, т.е. от "01" до "12";

M — месяц, текстовой, три буквы, например "Jan";

n — месяц, цифровой, одна цифра, т.е. от "1" до "12";

[1] От англ. AM (Ante Meridiem) — до полудня, PM (Post Meridiem) — после полудня. *(Примеч.ред.)*

o — разница со временем по Гринвичу, в часах, например "+0200";

s — секунды, цифровой, две цифры, т.е. от "О" до "59";

s — английский порядковый суффикс, текстовой, два символа, например "th"[1], "nd";

t — количество дней в данном месяце, т.е. от "28" до "31";

U — секунды с начала века Unix, т.е. с 1 января 1970 года;

Y — год, цифровой, четыре цифры;

w — день недели, цифровой, "О" означает воскресенье;

Y — год, четыре цифры, например "1999";

y — год, цифровой, две цифры, например "99";

z — день года, цифровой, например "299".

ВНИМАНИЕ

Некоторые параметры имеют различные значения при разном регистре, например d и D.

Теперь вы легко можете получить информацию о текущем времени и использовать ее на своем сайте. Один из самых распространенных вариантов — вывод текущего времени и даты. Конечно, все это выполняется и при помощи JavaScript, но видевшие эти скрипты поймут разницу (по крайней мере, в размере и скорости выполнения, не говоря уже о трафике от сервера к браузеру).

Время, на мой взгляд, выводить достаточно бесполезно, так как оно есть у каждого пользователя на системной панели Windows, а вот вывести число, день недели и месяц (да еще на русском языке) бывает полезно. Например, можно поприветствовать посетителей соответствующей фразой в зависимости от времени суток:

```
<?php
$h = date("H") ;
if ($h >= 5 && $h <= 11) echo "Доброе утро!";
```

[1] Имеются в виду суффиксы порядковых числительных в английском языке, например second (второй), seventh (седьмой). *(Примеч. ред.)*

```
if ($h >= 12 && $h <= 18) echo "Здравствуйте!";
if ($h >= 19 && $h <= 24) echo "Добрый вечер!";
if ($h >= 1 && $h <= 4) echo "Доброй ночи!";
?>
```

Цифры желаемого времени можно указать любые, в зависимости от личного понятия дня и ночи :-)

В этом скрипте сначала получаем текущее значение часов на сервере при помощи команды `date("H")`, затем проводим его анализ, и в зависимости от того, в каких пределах лежит полученное значение, выводим соответствующую фразу на экран посетителю.

Обратите внимание, что в данном примере используется именно время сервера, без учета часовых поясов и соответствующего смещения времени. Другими словами, это не обязательно будет время, правильное для посетителя, так как он может жить достаточно далеко от часового пояса сервера. Аналогичная проблема возникает в случае, если сервер расположен за рубежом, и, конечно, время на удаленном сервере точно не будет совпадать с необходимым для правильной работы сайта. В этом случае смещение необходимо учитывать и закладывать в программы заранее. Это можно сделать таким способом:

```
$time = date("H:i");
$time1 = date("H");
$time2 = date("i");
$time_s = 7;
$time1 = $time1 + $time_s;
if ($time1 >= 24) { $time1 = $time1 - 24; }
$time = "$time1:$time2";
```

В первой строке — просто получение текущего времени, используется в случае одинакового часового пояса с сервером, дальше — программа расчета правильного времени для разных часовых поясов. В переменной `$time_s` хранится собственно смещение.

Массивы

Массивы представляют собой ряд чисел или знаков, и имеют, как и все переменные, имя и значение. Но кроме этого они обладают индексом. Попробую пояснить. Допустим, вы стоите перед одноэтажным домом, в котором есть десять квартир. Дом — это определение, или имя, массива, а квартиры — это ячейки, которые в массиве доступны, как и любая переменная в PHP. Все массивы начинаются, как правило, с нуля, но это не принципиально, как нам удобно, так можем и начинать, хоть с 3423. Но правильнее, конечно, начать отсчет с нуля. Итак, первая квартира имеет имя $дом[квартира номер 0], вторая квартира — $дом[квартира номер 1], третья— $дом[квартира номер 2] и т.д. до девятой квартиры — $дом [квартира номер 8].

Обратите внимание: элементов массива 9, а последний индекс — 8. Здесь скрыта опасность запутаться, но если все четко себе представлять, такого не произойдет.

Таким образом, массивы в PHP выглядят так: $имя массива [индекс]. Имя может быть каким угодно, как и имя любой другой переменной. Индекс может быть либо числом, переменной, либо его может вообще не быть. В этом, нежелательном случае будет выбрана или записана ячейка массива, следующая за той, к которой было произведено последнее обращение в массиве. Например:

```
<?php
a[] = 1;
a[] = 67456;
a[] =0 "пример";
?>
```

В этом случае будет создан массив с именем $a и в его ячейки 0, 1 и 2 введены значения 1, 67456 и пример соответственно. Теперь достаточно дать команду echo $a [2] ; и на экран будет выведено слово «пример».

При таком задании массива может возникнуть проблема. Если массив с таким именем уже был определен раньше и в него были введены данные, то ввод данных продолжится с того индекса (точнее, именно с внутреннего указателя, а он может быть установлен и не на самый конец массива, хотя, как правило, это именно так), на котором ввод данных был прерван. Избежать этого можно, если явно указывать значение индекса ячеек — не $a[], а $a[0], $a[1], $a [2] и т.д. Или не путаться с именами массивов.

Рассмотренные нами массивы — одномерные. Бывают еще многомерные. Например, двумерные массивы можно тоже сравнить с домом, но не с одноэтажным, а, например, с пятиэтажным. В этом случае добавляется еще один индекс для учета смещения по этажам. Например:

```
$дом[первый этаж]  [квартира номер 0];

$дом[второй этаж]  [квартира номер 0];

...

$дом[пятый этаж]  [квартира номер 0];
```

С ячейками массива можно делать все то же, что и с любыми переменными PHP: применять арифметические, логические, операции сравнения, увеличения, уменьшения и т.д. А еще есть очень хорошая возможность, о которой нельзя не упомянуть.

PHP позволяет считать в любой массив целый файл, что дает большие перспективы и интересные возможности. Без такой функции не обходится ни один серьезный скрипт. Вот ее формат:

```
<php

$a = file ("имя файла");

?>
```

После выполнения этой команды в массиве $a будет находиться содержимое файла, имя или путь, к которому были указаны. Разделителем для разных элементов массива будет являться перевод

строки. Кстати, путь типа "http://...", как правило, не поддерживается. Он может быть только относительным. Это связано с безопасностью. Если на сервере (точнее, в настройках PHP) включен режим SafeMode, такие пути просто исключаются из запросов, и вы получите на экране сообщение об ошибке доступа. Хотя можете попробовать, вдруг у вас работает? А уж дальше что вы будете делать с полученной информацией — дело ваше.

Каждый массив имеет внутренний указатель, который определяет текущий элемент массива. В самом начале работы с массивом внутренний указатель находится на первом элементе. Функции end (), next (), prev () и reset () перемещают внутренний указатель массива. .

Функция end () устанавливает внутренний указатель массива на последнем элементе, next () передвигает внутренний указатель массива в сторону увеличения (т.е. вперед) и возвращает следующий элемент массива от текущей позиции внутреннего указателя массива или false, если элементов больше нет. Если массив содержит пустые элементы, то эта функция возвратит false и для этих элементов.

Функция prev () перемещает внутренний указатель массива в сторону уменьшения индекса (т.е. назад) и возвращает предыдущий элемент массива или false, если перед текущим нет элементов. Если массив содержит пустые элементы, то функция также возвратит false.

Функция reset <) устанавливает внутренний указатель массива на первом элементе.

Функция current {) возвращает элемент массива, на который в данный момент указывает внутренний указатель. Она не перемещает сам указатель. Если внутренний указатель находится в конце списка элементов, current () возвращает false. Если массив содержит пустые элементы (0 или пустую строку), то функция возвратит false для каждого из них.

Функция sort () сортирует массив по возрастанию, в том числе по латинскому и русскому алфавиту, так как русские буквы тоже имеют индекс, только несколько больший в отличие от латинских.

Функция rsort() сортирует массив в обратном порядке (по убыванию).

Конечно, это далеко не все функции, которые есть в PHP для работы с массивами. Но остальные —- достаточно специфичные. Рассмотренных нам вполне хватит для полноценной работы и для написания своих собственных программ.

Работа со строками

Строка — тип данных, значениями которого являются последовательности знаков (причем цифры могут выступать в роли строки наравне с другими символами). Строка также может состоять из одного символа или вообще быть пустой. Работа со строкой всегда начинается с ее ввода. Самое простое — присвоить переменной нужную строку. Вариантов как всегда много — или прочитать из файла, или выбрать из массива, или из формы, в которую пользователь введет информацию. Нужно четко представлять, что нам необходимо сделать дальше со строкой или набором строк. Несколько строк можно, например, объединить (еще говорят «склеить»). Для этого между строками достаточно поставить точку. Например:

```
<?php
$str = "Привет,";
$sto = "мир";
$qwe ="!";
$mir = $str. " ".$sto;
echo $mir.$qwe;
?>
```

В результате работы этого скрипта на экране появится надпись «Привет, мир», «собранная» из нескольких частей.

Если переменная не определена, она по умолчанию приравнивается к пустой строке. Однако это не число ноль, так как непосредст-

венно со строкой нельзя выполнить арифметическую операцию. Впрочем, если переменная все же действительно не определена заранее, выполнить операцию сложения, например, получится, так как переменные автоматически стремятся принять более правильный тип и станут целочисленными.

Если вы присваиваете переменной нужное значение, то оно, как правило, не нуждается в обработке, так как программист делает присвоение в коде программы, и конечно, в нужном виде. Но если строка считывается из файла или получается посредством формы, она (символьная строка) нуждается в обработке.

Самое первое, что стоит сделать, — удалить повторяющиеся пробелы. Для этого в PHP есть специальная функция: `chop(str);` Например:

```
$str = chop ($str);
```

В результате обработанное значение строки `$str` не будет содержать повторяющихся пробелов.

Если нужно убедиться в том, что строка не содержит пробелов в начале и в конце, применяется функция `trim(str);` Например:

```
$str = trim ($str);
```

Когда требуется удалить пробелы только в начале строки, нужно использовать функцию `ltrim()`.

Иногда бывает полезно проверить регистр символов. Функция `ucfirst()` делает первый символ в строке заглавным, независимо от того, каким он был до этого.

Есть и функция для перевода во всех словах в строке их первых букв в заглавные — `ucwords(str);` Кроме того, очень часто бывает необходимо сравнить строку с некоторым шаблоном. Частный случай — поиск в строке. Но нет никакой гарантии, что полученная строка введена пользователем или получена из файла в правильном виде. Другими словами, строка может содержать в середине слова или предложения чередующиеся строчные и прописные символы. Решение данной проблемы — в применении функций `strtolower(str);` (переводит строку в нижний

регистр) и strtoupper(str); (переводит строку в верхний регистр). Комбинирование всех этих функций приводит к корректному построению строки независимо от того, как она была введена или получена в начальном виде.

СОВЕТ

Функции обработки строк можно применять при проверке на соответствие логинов и паролей. Это не даст посетителям возможности использовать хитрые комбинации из маленьких и больших букв для создания похожих логинов.

Еще одна необходимая вещь при работе со строками — их обрезка. Она часто применяется при обработке форм для ввода данных. Представьте, что кто-нибудь из ваших «доброжелателей» введет в форму вашей гостевой книги текст этой книги. Как думаете, что получится? Чтобы избежать подобной ситуации, и нужно ограничить количество вводимых символов в любом поле формы. Сначала, конечно, нужно задать ограничение в HTML-коде самой формы:

```
<input maxlength="100" name="form">
```

Теперь форма с именем «form» ограничена в количестве вводимых символов числом 100. Но это еще далеко не все. Дело в том, что обойти такое ограничение очень просто, и нужно оно скорее для того, чтобы показать посетителю предел ограничения. Такие ограничения обходят следующим образом. Делают точно такую же страничку (пользователь может просмотреть ее код в любом браузере), изменяют число 100 на нужное, сохраняют ее у себя на диске с расширением HTML и запускают со своего компьютера. Серверу ведь все равно, откуда получать информацию (если не задать проверку в коде), он нормально воспримет такую подмену и обработает полученные данные. Чтобы запретить обрабатывать такие данные, нужно в обработчике проверять, откуда именно происходит ввод и каков размер полученных данных.

Для этого нужно воспользоваться PHP-функцией substring (string, start, length). Для нашего примера это будет выглядеть так:

```
$form = substr($form,0,99);
```

Этим вы просто отрезаете часть полученной строки, превышающую 100 символов (указывается цифра 99, так как счет символов начинается с нуля). Теперь все попытки ваших посетителей завалить вас информацией будут тщетны, так как ваш умный скрипт не пропустит больше определенного вами количества символов.

Собственно говоря, у функции `substr` совсем другое предназначение. Она возвращает часть строки `string`, определяемую параметрами `start` (начало) и `length` (длина). Если параметр `start` положительный, то возвращаемая строка будет начинаться с символа, который стоит на позиции `start` строки `string`. Например:

```
$form = substr("abcdef", 1); // вернет "bcdef"
$form = substr("abcdef", 1, 3); // вернет "bed"
```

Если параметр `start` отрицательный, то возвращаемая строка будет начинаться с символа, который стоит на позиции `start` от конца строки `string`. Например:

```
$rest = substr ("abcdef", -1); // вернет "f"
$rest = substr ("abcdef", -2); // вернет "ef"
$rest = substr("abcdef", -3, 1); // вернет "d"
```

Если параметр `length` указан и он положительный, то возвращаемая строка закончится через `length` символов от начала `start`. Это приведет к строке с отрицательной длиной (потому что начало будет за концом строки), поэтому возвращаемая строка будет содержать один символ от начала строки `start`. Если `length` указан и он отрицательный, то возвращаемая строка закончится за `length` от конца строки `string`. Это опять приведет к строке с отрицательной длиной, поэтому возвращаемая строка будет содержать один символ от начала строки `start`. Например:

```
$rest = substr("abcdef", -1, -1) ; // вернет "bede"
```

Вот такая полезная функция.

Кроме того, при обработке данных формы очень важно уметь вырезать из полученной, строки лишние или недопустимые символы. Для этого есть функция str_replace (needle, str, haystack) ;, при использовании которой все последовательности символов needle, введенные в строку haystack, заменяются на последовательность символов str Например:

```
$str = str_replace("", "\n", $str);

// вырезается символ ввода

$str = str_replace("red", "black", $str);

//в строке красный цвет будет заменен на черный
```

Вообще я очень люблю эту функцию и с ее помощью часто делаю многие полезные вещи. Не сомневаюсь, вы тоже сможете найти ей интересное применение.

Если требуются какие-то особые правила замены, то вам следует использовать функцию ereg_replace {), которую мы здесь не рассматриваем. Она хоть и более функциональна, но зато исполняется явно дольше и в большом цикле, например, приведет к ощутимым потерям времени.

Сессии

Сессии — очень эффективный механизм, появившийся в PHP 4.0. Упрощенно говоря, он позволяет передавать переменные от одного окна браузера к другому без их потери и без передачи методами POST или GET. При этом сессии используют в своей работе уникальные идентификаторы SID. Именно с их помощью можно определить, какой пользователь запустил сценарий. Идентификатор сессии хранится в cookie браузера. Фактически SID — это имя временного хранилища, т.е. временного файла, в котором PHP хранит информацию о сессии. Обычно эти файлы размещаются в каталоге tmp на диске сервера, но не на диске пользователя (в отличие от cookie).

Если cookie отключены, SID все равно передается, но уже при помощи методов GET ИЛИ POST, в зависимости от ситуации и более выгодного положения дел.

Имена переменных в сессиях задаются без знака доллара. Можно несколько раз регистрировать одну и ту же переменную в сессии, ее значение все равно будет сохранено.

Описание функций для работы с сессиями смотрите в приложении 3.

Часть IV

Программирование на PHP

Сравнение чисел

Задача: если числа равны — вывести сумму чисел, если числа не равны — выбрать большее и вывести его на экран. Решение:

```
1 <?php
2 $w = "4"; // первое число
3 $e = "6"; // второе число
4 if ( $w == $e )
5 {
6 echo $w + $e; // если первое и второе равны, выводим их сумму
7 exit;
8 }
9 if ( $w > $e ) { echo $w; } else { echo $e; }
10 // если нет — и одно больше другого — выводим числа
11 exit;
12 ?>
```

Опишем скрипт. Заданы два числа (строки 2—3). Сначала пытаемся проверить, равны числа или нет (строка 4). Обратите внимание на двойной знак равенства в операторе if. Дело в том, что если оставить тут один знак равенства, то переменной $w будет присвоено значение переменной $e. А в нашем случае надо проверить, равны ли переменные. Это и достигается двумя знаками

равенства. Другими словами, если вам надо провести сравнение, равны ли две переменные, не забудьте поставить два знака равенства. Поставите один — будете очень долго искать правды от вашего скрипта, а он упорно не будет работать.

Если числа равны, выводим их сумму, используя знак сложения между переменными и функцией echo (строка 6), а потом завершаем скрипт, если это надо командой exit. Команду завершения в этом месте можно было бы не ставить, в этом случае интерпретатор стал бы дальше обрабатывать код, и в конце его нашел бы ту же команду в строке 11, но это заняло бы определенное время. Старайтесь не заставлять интерпретатор производить ненужные действия, и он отплатит вам приличной скоростью работы.

Если же числа не равны, код вывода на экран суммы (обратите внимание, что он заключен в фигурные скобки) будет проигнорирован, и интерпретатор продолжит свою работу и станет проверять, какая из переменных больше (строка 9). Очевидно ведь, что если значения переменных не равны, то одно из них больше, а другое меньше.

Итак, если $w > $e, выводим на экран $w, так как она больше. А если условие не удовлетворяется, интерпретатор считает, что $w <= $e, и выполняется конструкция в скобках после else — { echo $e; }. Но так как на равенство мы уже проверяли переменные в начале скрипта (в строке 4), то эта конструкция выполнится, только если $w < $e. Вот тут нам и пригодился оператор exit в строке 7, так как если бы его не было, то при равенстве переменных на экран, кроме суммы переменных, была бы выдана переменная $e.

После обработки всех строк интерпретатор завершает свою работу, а браузеру передается результат в виде одного числа.

Еще один оператор, при помощи которого можно сравнить переменную или выражение с различными значениями и выполнить разные фрагменты кода в зависимости от того, чему будет равно значение выражения, -- это switch. Как всегда, лучше понять смысл на примере:

```
<?
function date_format($date)
// функция получает значение переменной $date
{
$year = substr($date, 0, 4);
// переменной $year присваиваем первые четыре символа
   //от $date
$month = substr($date, 4, 2};
// переменной $month — символы с пятого по шестой
$day = substr ($date, 6, 2};
switch ($month)
// конструкция switch($month) делает следующее:
// берется значение $month и проверяется на равенство
// условию case xx:
{
case 01: $month = "Января"; break;
case 02: $month = "Февраля"; break;
case 03: $month — "Марта"; break;
case 04: $month = "Апреля"; break;
case 05: $month = "Мая"; break;
case 06: $month = "Июня"; break;
case 07: $month = "Июля"; break;
case 08: $month = "Августа"; break;
case 09: $month = "Сентября"; break;
case 10: $month = "Октября"; break;
case 11: $month = "Ноября"; break; .
```

```
case 12: $month = "Декабря"; break;
}

$date = "$day $month $year г.";
echo $date;
}

$date = date("Ymd");
// вызываем функцию date() с необходимыми нам параметрами
date_format ($date) ;
// передаем полученное значение в функцию date_format ()
?>
```

Вот как можно применить на практике сравнение чисел.

Вложение файлов в документ

Каждый, сделавший хоть одну страничку в Сети, сталкивался с проблемой изменения тех или иных данных на ней. Конечно, это не сложно, когда страница одна или их несколько. Но если вы сделали большой сайт, маленькое дополнение (например, строка в меню) в сотни файлов может превратиться в настоящий кошмар.

PHP быстро решает эту проблему, позволяя вкладывать одну страницу в другую. Достигается это с помощью операторов require () и include (). После этих операторов в круглых скобках должен стоять путь к вкладываемому файлу. Например: include ("text.phtml"). Различие между указанными операторами заключается в том, что require () подменяется содержимым указанного (файла и может быть использован только один раз, а include () вставляет и выполняет содержимое указанного

файла, что позволяет применить его несколько раз, например в цикле. В любом случае добавляемый код надо заключить в конструкцию <?php ... ?>.

Рассмотрим подробнее различия этих двух операторов. Я уже упоминал, что PHP частично является компилятором, так как при обработке преобразовывает код в свое внутреннее представление. Когда интерпретатор доходит до оператора include (), он останавливается и запускает работу программы с начала или с момента прежней остановки. И только после выполнения этого участка кода вызывает файл, указанный в include (). В случае с require () действия интерпретатора совсем иные. Он сразу обрабатывает указанный файл, не делая никаких остановок на выполнение предыдущего уже обработанного кода. Таким образом, скорость обработки require () значительно выше за счет отсутствия остановки на дополнительное вложение и обработку. Однако использовать в цикле или в условии require () уже не получится, в таких случаях нужен именно include ().

Вложения файлов могут происходить только внутри серверного пространства, доступного PHP. Другими словами, вы не можете использовать в имени файла http://, если только это не разрешено в настройках сервера.

Достаточно часто встречаются сайты, ссылки на которые включают в себя специальные символы — &, ?, %. Все это может быть признаком выполнения PHP-скрипта. Дело в том, что если ссылку написать так: адрес?имя=значение, то переменная с этим именем будет доступна под этим же именем в файле, на который указывает ссылка.

Например, ссылка http://name.com/index.phtml?lex=7 указывает на файл index.phtml. Но если на нее щелкнуть, файл будет запущен на сервере с инициализированной переменной PHP $1ex со значением 7. Этот прием позволяет легко передать программе нужные данные. Метод такой передачи называется GET. Еще есть, например, метод POST (см. приложение 5).

Если необходимо добавить несколько переменных, то они могут быть разделены знаком &. Теперь мы можем сделать сайт, кото-

рый будет доступен с помощью только одной страницы. А всю остальную информацию эта страница будет выводить на основании полученных по ссылке данных.

Выглядеть ссылка может следующим образом: `http://имя/index.phtml?link=1` Единица в конце ссылки и есть наш параметр, который будет подставляться в файле index.phtml. Например:

```
<html>
... начало файла ...
<?php
$url = "";
if (Slink == 1) { $url = "name1.phtml"; }
if ($link == 2} { $url = "name2.phtml"; }
if ($link == 3) { $url = "name3.phtml"; }
if ($link == 4) { $url = "name4.phtml"; }
if ($url == "") { $url = "error.phtml"; }
include ($url);
?>
... конец файла ...
</html>
```

Обратите внимание, написанный нами код в строке `if ($url == "") { $url = "error.phtml"; }` учитывает ситуацию, когда посетитель по разным причинам указал неправильный параметр. В этом случае выводится заранее заготовленная страница с сообщением об ошибке — error.phtml. Если же параметр соответствует какому-либо файлу сайта, он вкладывается в код файла index.phtml и исполняется.

Таким образом, начало и конец кода всегда остаются одинаковыми, а изменяется только середина. И какие-либо глобальные изменения уже не кажутся такими страшными, как раньше. Ведь

сделать их надо только в одном файле, а отразится это на всем сайте.

А вот то же самое, только более хитрым способом:

```
<html>
... начало файла ...
<?php
include ("name".$link. ".phtml");
?>
... конец файла ...
</html>
```

Результат работы абсолютно идентичен первому скрипту, за исключением того, что страница с сообщением об ошибке теперь стала называться просто name.phtml.

Есть и другой случай использования в адресе специальных символов. Его суть заключается в том, что у PHP есть доступ к так называемым переменным окружения сервера. Одна из этих переменных — запрашиваемый посетителем относительный[1] путь к странице на сайте. И этот путь становится нам доступен для использования.

В этом случае ссылки будут следующего вида: http://имя/index.phtml?patch/name.phtml. Вторая часть ссылки — patch/name.phtml — будет доступна, если мы считаем параметр $QUERY_STRING. Например:

```
$add = $QUERY_STRING
```

Изменим наш основной файл index.phtml так, чтобы все работало автоматически. А если запрашиваемый параметр не будет указан (правильно говоря — будет равен пустой строке), присвоим переменной $add имя файла, который должен быть открыт как

[1] Абсолютный путь — это полный адрес страницы в интернете, например http://www.my_site.com/page_1/info.html. Относительный путь учитывает только дополнения к основному адресу, в нашем случае — это page_1/info.html. *(Примеч.ред.)*

главная страница. Пусть это будет main.phtml. Тогда код будет выглядеть следующим образом:

```
<html>
... начало файла ...
<?php
$add = $QUERY_STRING;
if ($add == "") { $add = "main.phtml"; }
include ($url);
?>
... конец файла ...
</html>
```

СОВЕТ

Вот вам и пример оптимизации. Строку кода if ($add == "") { $add = "main.phtml"; } можно (и нужно) писать так: if (!$add) { $add = "main.phtml"; } Восклицательный знак перед именем переменной указывает на ее ложность. Другими словами, переменная равна пустой строке. Таким образом, смысл кода остается прежним, а написание и удобочитаемость значительно улучшаются.

Обратите внимание, что этот метод хоть и проще первого, но он открывает путь к получению информации о сервере, на котором расположен сайт с такой организацией структуры. Злоумышленник или просто любопытный человек при наличии знаний и стечении определенных обстоятельств сможет много узнать о вашем сервере, а это открывает прямой путь к «взлому». Еще один отрицательный момент такого метода заключается в потере гибкости управления и организации сайта. Вы сразу лишаете себя многих полезных возможностей, например, очень сложно становится передать в исполняемый файл какую-либо переменную из адресной строки, а иногда это нужно.

По этим причинам отнеситесь к этому методу, как к примеру, и старайтесь не использовать его в жизни.

Простейший счетчик посещений

Рассмотрим скрипт, который позволит организовать на любой из страниц вашего сайта счетчик посещений. Этот счетчик не будет полнофункциональным, так как имеет много недостатков, но как пример использования PHP вполне подойдет. В любом месте вашей страницы (но только там, где это нужно) вставьте следующий код:

```
1 <p>Посетителей страницы
2 <?php
3 $filename = "counter.dat";
4 $fp = @fopen($filename,"r"};
5 if ($fp)
6 {
7 $counter = fgets ($fp,10);
8 fclose($fp);
9 } else { $counter = 0; }
10 $counter++;
11 echo $counter;
12 $fp = @fopen($filename,"w");
13 if {$fp}
14 {
15 $counter = fputs{$fp,$counter};
16 fclose($fp);
17 }
18 ?></p>
```

В том же каталоге, в котором будет работать этот скрипт, необходимо создать файл counter.dat, закачать его на сервер и присвоить ему атрибуты, разрешающие запись.

Опишем скрипт построчно.

1. Выводим на экран надпись «Посетителей страницы» при помощи HTML-тега <p>.

2. Открываем скрипт.

3. Присваиваем переменной имя файла, в котором будет храниться количество посещений.

4. Открываем соединение с этим файлом, причем только на чтение.

5—6. Проверяем, успешно ли открылось соединение.

7. Если успешно, считываем из открытого файла первые 10 символов в переменную счетчика $counter.

8. Закрываем соединение.

9. Если соединение не открылось, присваиваем переменной счетчика ноль.

10. Увеличиваем переменную счетчика на единицу.

11. Выводим на экран переменную счетчика.

12. Открываем соединение на запись с очисткой всего содержимого файла.

13—15. Если успешно, то записываем новое значение переменной счетчика в файл.

16. Закрываем файл

Обработка форм

Посещая сайты, каждый не раз сталкивался с различными формами и полями для ввода. Как обрабатываются данные из формы? Ответ очевиден — при помощи PHP. Разберемся, как это работает.

Для начала давайте сделаем форму для отправки писем на любой адрес, который требуется ввести в форме:

```
<form method="POST" action="action.phtml">
<p><input type="text" name="email" size="20">
<br><input type="text" name="name" size="20">
<br><textarea rows="2" name="txt" cols="20"x/textarea>
<br><input type="submit" value="Отправить" name="Bl">
<input type="reset" value="Очистить" name="B2"</p></form>
```

Запустив этот HTML-код в браузере, вы увидите перед собой форму с тремя полями для ввода (рис. 20).

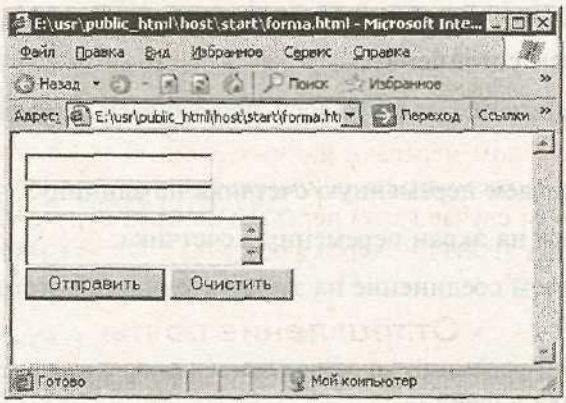


Рис. 20. Полученная форма для отправки писем

Обработка, которая начнется после нажатия кнопки «Отправить», будет передана файлу action.phtm. Имя первого поля — «email», и в него посетитель будет вводить адрес, по которому надо отправить письмо. Второе поле имеет имя «name», и оно предназначено для ввода имени того, кто хочет отправить письмо. Можно, конечно, обойтись и без этого, но для наглядности не помешает. Следующее поле — для ввода текста сообщения, и оно имеет имя «txt». Наша форма почти готова. Не хватает только эстетического оформления, но это уже дело десятое, можно сделать любые надписи или применить различные стили,

проявите свои творческие способности. А мы теперь перейдем к основной части — обработчику формы, который, как вы помните, находится у нас в файле action.phtm. Вот PHP-код для обработки этой формы:

```
<?php mail{$email, $name, $txt); ?>
```

Как видите, имена, которые мы присвоили полям для ввода в нашей форме, перешли точно таким же переменным в скрипт PHP, который указывается в форме как обработчик. Причем значением этих переменных PHP будет являться то, что посетитель вашей страницы введет в соответствующее поле. Адрес окажется в переменной $email, имя — в $name, а текст — в $txt.

Итак, любое имя формы становится переменной в обработчике, написанном на PHP. Метод, при помощи которого передаются переменные со своими значениями в интерпретатор, называется POST. Если помните, несколько ранее мы познакомились еще с одним методом передачи данных скрипту — GET. На данном примере вы можете видеть основное отличие двух похожих методов. В первом случае (GET) передача данных переходит по ссылке, во втором (POST) — через форму.

Отправление почты

Очень часто хочется знать, что происходит на сайте за время вашего отсутствия. Конечно, можно периодически проверять форум, гостевую книгу, другие сервисы сайта на предмет новых сообщений, но есть способ лучше. Почему бы не присылать себе сообщение на свой почтовый ящик, когда на сайте происходит то или иное событие, например подписка на рассылку, ошибка открытия файла, организация обратной связи, опрос мнения читателей о размещенной статье, ввод сообщения в форум, гостевую книгу и т.д. Способов применить эту возможность PHP много, разберем один из них с использованием функции mail. Ее синтаксис:

```
mail ( $email, "Введено сообщение", $str, "From: сообщение ") ;
```

Конечно, перед этим пользователи должны ввести соответствующие переменные либо вручную, либо с помощью формы (нужно следить за соответствием имен переменных в форме и в скрипте). При выполнении данной команды интерпретатор PHP пошлет письмо с текстом из переменной $str по адресу, указанному в переменной $email. Все остальное — служебная информация, которая может быть в некоторых случаях опущена за ненадобностью.

Для примера рассмотрим систему контроля над посетителями определенных страниц сайта, которая может понадобиться для анализа посещений определенных статей, страниц и т.д. Нам потребуется информация о посетителе, которая в PHP автоматически доступна через переменные окружения. Эту информацию мы будем отсылать себе на почтовый ящик:

```
<?php
$host = gethostbyaddr($REMOTE_ADDR);

$ip = getenv("REMOTE_ADDR");
$date = date("d M Y, H:i:s");

$str = ("
Дата — $date
Хост — $host
IP-адрес - $ip
————————————");
mail ( $email, "статистика", $str, "From:информация" );
?>
```

Значение хоста получается из переменной окружения, в которой хранится IP, — $REMOTE_ADDR. Далее получается сам IP из той же переменной и дата. Потом это задается в переменную для отправки на ящик, указанный в переменной $email.

Только не переусердствуйте — каждое посещение такой страницы вызовет отправку письма, и ящик может оказаться переполненным, особенно если поток посетителей будет большим.

Если вы разместите на одной из ваших страниц этот PHP-код, информация о посетителе (это дата входа, хост и IP-адрес) будет в вашем почтовом ящике.

Еще можно прислать себе сообщение при возникновении ошибки на сайте (например, открытия файла):

```
$filename = "data.txt";
$fp = @fopen($filename,"r");
if ( !$fp )
{
@mail( $email, "Ошибка!", "Ошибка открытия файла $filename!");
}
```

Как обычно, можно поставить знак @, и если возникнет ошибка при отправке почты, сообщение об этом не будет выведено на экран. А можно просто проверить, ушло письмо или нет:

```
if (@mail( $email, "Тест", $str))
{ ... что делать, если письмо ушло ... }
else
{ ... что делать, если произошла ошибка отправки (неправильные настройки сервера, не доступен ящик, сайт и т.д.) ... }
```

Если в переменной $email указать несколько адресов, разделенных пробелами, информация будет разослана по всем этим адресам, что открывает простой путь к организации собственной, ни от кого не зависящей системы почтовых рассылок.

Отправление письма в HTML-формате

Необходимость отправить письмо в HTML-формате возникает не так уж и редко, например, тогда, когда нужно организовать рассылку рекламных материалов компании. Это не обязательно может быть спам, а обыкновенная корпоративная рассылка. Конечно, в этом случае очень хотелось бы не просто помещать в такую рассылку текст, а оформлять его в соответствии со стандартами HTML.

Все, что нужно — это правильно сформировать заголовок, который отправляется вместе с письмом. Например:

```
<?
$header="Content-Type: text/html; charset=windows-1251\n";
mail("адрес, на который высылается письмо","Это может быть тема письма","

<h1>Сообщение в HTML-формате.</hl>
<p>Допустимы любые теги<br>
<hr>
<table><tr><td>Вот, например — </td>
<td>таблицы</td></tr></table>
<hr>

",$header);
?>
```

Как видите, переменная $header имеет заголовок с типом Content-Type: text/html; Это указание почтовой программе воспринимать полученное сообщение как HTML-письмо. Экспериментируя с кодом, вы сможете составлять такие письма и красиво оформлять их. Только не забудьте задать адрес, по которому будет отправлено письмо, и указать его тему.

А вот и еще один вариант, который позволяет выбрать кодировку отправляемого письма:

```
$to = "адрес, на который высылается письмо";
$subject = "тема письма";
$text = "текст письма";
$from = "от кого письмо";
$kodir = "кодировка письма";

// $kodir = "windows-1251";
// $kodir = "koi8-r";

$header = "From: $from\nReply-To: $from\n";
$header .= "MIME-Version: 1.0\n";

$header .= "Content-Type: text/plain; charset=$kodir\n";

$header .= "Content-Transfer-Encoding: 8bit\n\n";
$header .= "$text\n";
mail($to, $subject, $text, $header);
```

Надеюсь, вам уже все понятно без подробного рассмотрения примера. Видите закомментированные строки? В первых двух — варианты русской кодировки. Выбирайте ту, которая вам нужна. Последний знак комментария надо убрать, если письмо предполагается в HTML-формате. Только в этом случае не забудьте поставить комментарий на предыдущей строке.

Дата по-русски

Задача — вывести на экран пользователя дату на русском языке. Решение:

```
<?php
// определяем массив для месяцев
$q[] = "";
$q[] = "января";
$q[] = "февраля";
$q[] = "марта";
$q[] = "апреля";
$q[] = "мая";
$q[] = "июня";
$q[] = "июля";
$q[] = "августа";
$q[] = "сентября";
$q[] = "октября";
$q[] = "ноября";
$q[] = "декабря";

// определяем массив для дней недели
$e[0] = "воскресенье";
$e[1] = "понедельник";
$e[2] = "вторник";
$e[3] = "среда";
$e[4] = "четверг";
$e[5] = "пятница";
$e[6] = "суббота";

// считываем месяц
$m = date("m");
if ($m == "01") $m = 1;
if ($m == "02") $m = 2;
if ($m == "03") $m = 3;
if ($m == "04") $m = 4;
if ($m == "05") $m = 5;
if ($m == "06") $m = 6;
```

```
if ($m == "07") $m = 7;
if ($m == "08") $m = 8;
if ($m == "09") $m = 9;

// считываем день недели
$we = date("w");

// считываем число
$chislo = date("d");

// извлекаем день недели из ранее определенного массива $e
$den_nedeli = $e[$we];

// извлекаем значение месяца из ранее определенного
// массива $q
$mesyac = $q[$m];

echo "Сегодня ".$chislo." ".$mesyac", ".$den_nedeli;
?>
```

Этот скрипт не идеален, но работоспособен и хорошо подходит для того, чтобы учиться. Рассмотрим, как он работает. Сначала нужно определить два массива, в которых будут храниться соответственно русское название месяца и русское название дня недели. Месяц не может быть нулевым, поэтому нужно позаботиться о вводе элемента массива с нулевым индексом. Если индекс массива не указан, он принимается равным внутреннему указателю. Если массив пуст и еще не определен, внутренний указатель находится на первом элементе (в данном случае — на имеющем нулевой индекс).

Ввод нового элемента массива перемещает внутренний указатель на единицу вверх, и таким образом обеспечивается последующий ввод значения в ячейку массива, имеющую индекс, на единицу больший, чем предыдущий. В принципе, можно обеспечить ввод данных в массив разными способами. Но указанный здесь — самый простой и легкий для понимания. Мы просто присваиваем поочередно нужные нам данные элементам массива и таким образом заполняем его.

Точно так же и со вторым массивом, в котором индекс ячеек задается явно и внутренний указатель устанавливается на тот индекс, который задан. После ввода значения указатель также перемещается вверх на единицу. Разница между этими двумя методами в том, что при незаданных индексах заполнится ячейка массива, на которую указывает внутренний указатель. А он ведь может находиться и не в конце. А если индекс указан явно, внутренний указатель устанавливается на его значение, и запись происходит в явно указанную ячейку.

Когда массивы определены, нам нужно считать номер месяца. Если номер месяца меньше 10, он считывается с первым нулем (т.е. 02), и поэтому нам нужно отсечь ноль. Можно использовать разные алгоритмы и методы, но мы воспользуемся очень простым, но достаточно громоздким решением — сравним полученное решение с рядом заранее известных вариантов и изменим номер месяца на правильный без нуля. Но так делать не рекомендуется — есть более изящные методы. Например, проверить полученное значение на наличие первого ноля и, если результат проверки — истина (самый первый символ в строке — ноль), удалить первый символ.

Далее по ходу скрипта считываем день недели и число. С числом делать ничего не нужно, так как дата будет понятна всем, а вот день недели и месяц нужно обработать. Извлекаем из введенного нами ранее массива $e день недели. Номер дня недели указывает на ячейку массива, в котором хранится нужное русское имя, и таким образом мы в любом случае получим правильное значение. Изменяется номер дня недели — изменяется индекс ячейки, из которой считывается значение. Причем в отличие от массива с именем месяца, здесь нолю соответствует воскресенье, что мы и учли при вводе массива дней недели. Точно такую же операцию проводим и для месяца. Его номер указывает на ячейку массива, где хранится правильное имя месяца на русском языке.

В конце выводим результат на экран в произвольной форме.

Счетчик посещений с использованием базы данных

В данном примере мы будем не просто бездумно вести подсчет, а сформируем базу данных с возможностью просмотра статистики по каждой из существующих страниц.

Сначала разберемся более подробно с поставленной задачей. Есть несколько страниц сайта, каждая из которых имеет уникальный адрес — URL. Именно его мы и положим в основу нашей базы данных. Для этих целей отведем один файл с именем, например urlfile.txt.

Вторая составляющая статистики — количество посещений. Мы не будем пока касаться вопроса уникальности каждого посещения, а реализуем для начала их простой подсчет. Для статистики отведем еще один файл — counter.txt.

Этим обоим файлам — urlfile.txt и counter.txt — нужно установить соответствующие атрибуты, разрешающие запись в эти файлы, иначе скрипт выдаст ошибку.

Сам скрипт разместим в файле log.phtml, а вывод результатов — в файле index.phtml. Все четыре файла лучше сохранить в отдельную директорию (у нас она будет названа count), чтобы в дальнейшем не запутаться.

Когда все готово, можно начать программировать. Открываем свежесозданный файл log.phtml в текстовом редакторе и записываем в него следующие строки:

```
<?
if($QUERY_STRING != "")
{ $url = $PHP_SELF.'?'.$QUERY_STRING; }
else { $url = $PHP_SELF; }
```

Этим кодом мы считываем адрес, с которого вызывается скрипт и который задан в переменной окружения $PHP_SELF. НО нужно учитывать, что в адрес вполне могут быть включены параметры,

которые идут после вопросительного знака и которые можно прочитать уже из переменной окружения $QUERY_STRING. Итак, если переменная $QUERY_STRING у нас пуста, адрес считывается только из $PHP_SELF, если нет, адрес комбинируется из двух переменных окружения.

Дальше:

```
$add = $DOCUMENTROOT."/count/urlfile.txt";
$adds = $DOCUMENTROOT."/count/counter.txt";
$li = file($add);
$a = count($li);
```

Здесь мы формируем пути, по которым скрипт будет искать файлы для записи данных статистики.

$DOCUMENT_ROOT — еще одна переменная окружения, в ней хранится абсолютный путь к вашему сайту, благодаря наличию данной переменной обеспечивается стопроцентная работоспособность скрипта, из какой бы директории он ни был запущен. В конце этого кода считываем в массив с именем $li содержимое файла с базой данных адресов. Пока эта база пуста, и массив, соответственно, пуст.

Следующие действия:

```
$i = 0; $w = -1;
while ($i <= $a):
$tmp = trim(str_replace ("\n","", $li[$i]));
if ($tmp == $url) { $w = $i; $i = $a++; }
$i++;
endwhile;
```

Этот код отвечает за определение местоположения запрошенного адреса в массиве адресов. Массив пока пуст, поэтому запрошенный адрес не обнаруживается, и его необходимо туда записать. За это отвечает код:

```
if ($w == -1)
{
$fp = fopen($add, "a+");
if ($fp) { $fw = fwrite($fp, $url."\n"); fclose($fp);
}
$fp = fopen($adds, "a+");
if ($fp) { $fw = fwrite($fp, "0"."\n"); fclose ($fp); }

$w = $a++;
}
```

Файл адресов и файл количества посещений дополняются новыми данными, учитывается и перевод строки, так как он является разделителем при считывании файла в массив в дальнейшем. Если же запрошенный адрес уже есть в базе данных, этот код не выполняется, так как переменная $w равна номеру позиции запрошенного адреса в массиве адресов.

Таким образом, достигается равновесие: если адрес есть, то ничего не делаем, если нет — добавляем его туда, а переменная $w все равно указывает на номер позиции в массиве адресов.

Дальше:

```
$co = file($adds);
$co[$w] = trim(str_replace ("\n","", $co[$w]));
$co[$w]++; $count = $co[$w];
$co[$w] = $co[$w]."\n";
```

Данный код отвечает за загрузку базы данных количества посещений страниц в массив $co и за увеличение на единицу нужной позиции в этом массиве. Переменной $count передается значение текущего счетчика посещений данной страницы, его можно позже вывести на экран.

Посмотрите на вторую строку, нам уже встречалась эта функция. Дело в том, что когда PHP считывает данные из файла в массив,

разделителем считается перевод строки. Но это не значит, что он не попадает в массив. Кроме перевода строки, в массив могут попасть пробелы с начала и с конца строки, и от них, как и от перевода строки, нужно избавиться. Именно это и делает указанная строка кода.

Дальше:

```
$p = implode ("", $co);
$fp = fopen($adds, "w");

if ($fp) { $fw = fwrite($fp, $p) ; fclose ($fp); }
```

Этим кодом увеличиваем показание счетчика, дополняем его переводом строки для корректной записи и записываем весь массив $co в файл. Прежде чем это сделать, массив объединяется в одну переменную, и уже она легко записывается. Осталось завершить код тегом окончания кода PHP:

```
?>
```

Вот и все, что требовалось внести в самый большой и сложный файл log.phtml.

Дальше сделаем вывод результатов на экран, чтобы была возможность наглядно оценить посещаемость ваших страниц. За это, как вы помните, отвечает файл index.phtml Начнем теперь заполнять кодом его:

```
<? include ($DOCUMENT_ROOT."/count/log.phtml"); ?>
```

Так будет вызываться наш скрипт статистики. Подобная вставка вызовет либо пополнение базы данных адресов, либо увеличение соответствующей позиции счетчика на единицу. Результаты можно разместить в таблице:

```
<table align="center" border="1">

<?

$add = $DOCUMENT_ROOT."/count/urlfile.txt";

$adds = $DOCUMENT_ROOT."/count/counter.txt";

$li = file($add);
```

```
$co = file($adds);
$a = count($li);
```

Этот код нам уже знаком. Он практически ничем не отличается от такого же в файле log.phtml. Мы узнали адреса файлов базы данных и ввели информацию в массивы. Осталось ее только вывести, вставив нужные теги HTML:

```
$i = 0; $w = 0;
while ($i < $a):
echo "<tr><td>.
<a href=$li[$i]>$li[$i]</a>
</td><td>$co[$i]</td></tr>";
$i++;
endwhile;
?></table>
```

Теперь наши результаты будут выведены в таблице. Вот и вся работа!

В заключение хочу еще раз напомнить, что вызов скрипта статистики осуществляется следующей командой:

```
<? include ($DOCUMENT_ROOT. "/count/log.phtml"); ?>
```

Достаточно вставить этот код в любое место любого файла (но только с расширением для PHP), и он будет проиндексирован в базе данных, и в дальнейшем каждое посещение этой страницы будет учитываться.

Счетчик персональной посещаемости

Теперь попробуем сделать счетчик персонального учета посещаемости, который будет выводить посетителю информацию о том, сколько раз он заходил на вашу страничку.

Эту задачу мы реализуем при помощи технологии cookie (см, приложение 4).

Итак, попробуем реализовать персональный счетчик. В первых строках файла, еще до вывода любых тегов и сообщений, пишем:

```
<?
$y = mktime(0,0,0,1,1,2022);
if (isset ($name)) { setcookie("name "', "0", $y) ; }
else { $name++; setcookie("name ", $name, $y); }
?>
```

Смысл кода очень прост. Если cookie с именем $name установлен, его значение считывается и увеличивается на единицу. Если нет — в cookie записывается ноль. В дальнейшем он будет считан и учтен. В любом месте страницы теперь можно вывести результаты:

```
<p>Персональный счетчик - <? echo $name; ?></p>
```

Как видите, очень просто, весь код буквально поместился в одну строку.

Теперь вы легко сможете организовать с помощью технологии cookie много полезных приемов. Только не забывайте, что у некоторых пользователей cookie могут быть запрещены в настройках браузера. В таком случае нужно хотя бы вывести сообщение о необходимости разрешить прием cookie.

Подсчет переходов по ссылкам

Допустим, нужно узнать, сколько раз нажимают на ту или иную ссылку на вашем сайте. Например, иногда важно знать, сколько щелчков произвели посетители по баннеру. Сможете потом похвастаться перед рекламодателями.

Итак, приступим. Первым делом создадим себе две ссылки: http://virtual.brest.by/php/ и http://20404 0.com.

Вторая ссылка — это интересный сайт одной неплохой фирмы в моем городе. Кто очень интересуется, как извлекать информацию с файла Excel, может посмотреть, как это там сделано. Но это к слову. Теперь о деле.

Эти ссылки мы должны заменить одной своей. Допустим, у вас есть сайт http://home.name (нет такого сайта, не ищите, это я придумал). Сайт расположен на сервере, поддерживающем PHP.

В таком случае вы, например, можете заменить указанные выше ссылки (http://virtual.brest.by/php/ и http://204040.com) такими: http://home.name/reg.phtml?id=1 и http://home.name/reg.phtml?id=2.

Уловили суть? Вы направляете посетителя не сразу на выбранные им ресурсы, а сначала к себе на специальную страницу, на которой можно, проанализировав переменную $id, узнать, куда именно решил уйти ваш посетитель. Учесть этот переход в заранее созданном файле и потом без вопросов перенаправить этого непутевого посетителя на тот ресурс, который он выбрал. Обратите внимание, что ни в коем случае нельзя выводить ничего на экран, иначе переход не состоится, потому что функция перенаправления на другую страницу не сработает.

Приступим к реализации задуманного:

```
<?
$u[0] = "http://home.name";
$u[1] = "http://virtual.brest.by/php/";
$u[2] = "http://204040.com";
```

В этой части кода определяемся с адресами, на которые будут происходить перенаправления. Тестовые адреса у нас уже есть, вот их и будем использовать. Если произойдет ошибка с переменной $id, посетитель попадет на главную страницу вашего сайта — home.name. А можно и не на главную, достаточно задать адрес нужной страницы в первой строке.

Следующие действия:

```
$add = "log.txt";
$sl = file($add);
```

При помощи этого кода в массив считывается содержимое базы данных со статистикой переходов. Проверяем содержимое переменной $id на ошибку:

```
if ($id < 1 or $id >= count ($u) ) { $id = 0; }
```

Если ошибка присутствует, то присваиваем переменной ноль.

Дальше:

```
$url = $u[$id];
$temp = trim(str_replace {"\n","", $sl[$id]));
if (!$temp) { $temp = 0; }
$temp++;
$sl[$i] = $temp."\n";
```

Тут немного сложнее, но сейчас попробуем вместе разобраться. Нужный нам адрес задан в массиве $u, который мы описали в начале, поэтому считываем оттуда адрес, на который сделаем переход. Потом считываем в переменную $temp число, соответствующее количеству переходов по этому адресу. Увеличиваем полученное число на один и добавляем к нему символ перевода строки — для удобочитаемости результата.

Объединяем массив в одну переменную:

```
$sav = implode($sl, "");
```

и записываем в файл учета:

```
$fp = @fopen($add, "w+");
if ($fp) { $fw = fwrite($fp, $sav); fclose($fp); }

header ("Location: ".$url);
?>
```

Осталось только перейти по считанному адресу. Наш скрипт, кстати, умеет сам заполнять пустые поля, т.е. файл со статистикой не нужно заполнять нулями, он будет заполняться сам.

Вот и все. Теперь у вас есть возможность узнать, куда пропали ваши посетители, и убрать эту вредную ссылку. Шутка.

Сохраняем информацию о посещениях

Занимаясь разработкой даже небольшого интернет-проекта, в котором присутствуют динамические страницы, приходится сталкиваться с необходимостью хранить различные данные.

Это могут быть показания счетчика посещений, данные баннерной системы сайта или любая другая информация. Рано или поздно наступает момент, когда приходится задумываться об оптимизации хранения и учета таких данных. Конечно, лучше всего использовать базу данных, например MySQL, но часто это не доступно по разным причинам (допустим, ее может не быть у хостинг-провайдера).

Другой вариант — отвести каждому виду данных собственный файл. Это очень просто и доступно, и именно так поступают в тех случаях, когда надо быстро наладить и запустить в работу систему сайта. Однако со временем количество задач увеличивается, и однажды оказывается, что отведенные файлы «размножились» в больших количествах и сами начинают требовать учета.

Итак, профессиональные базы данных мы не рассматриваем. В таком случае остается всего несколько путей. Один — создание собственной базы на основе всего одного файла. Неплохое решение, но мы пойдем другим путем: будем хранить информацию непосредственно в файле, который исполняет программу.

Для примера рассмотрим систему, которая по очереди выводит баннеры разных сайтов. Это могут быть любые баннеры, и их количество, в принципе, не ограничено. У нас их будет три.

Следующий код вам надо разместить в самом начале файла:

```
<?
// 1
?>
```

Перед ним ничего не должно быть. Не беспокойтесь, это не помешает использовать в дальнейшем cookie, ведь этим кодом на экран не будет выводиться ничего. Этот код нельзя менять — все

переводы строк должны остаться на своих местах, т.е. код надо оставить в виде трех строк.

Дальше:

```
<?
$nomer_ban = 3;
$name_file = "путь к файлу и его имя";
$list = file($name_file);
?>
```

В переменной $nomer_ban будет храниться количество баннеров в системе баннерообмена. Их может быть любое количество, как я уже упоминал выше.

В переменной $file_name нужно указать путь к файлу, в который вносится этот код. А также следует помнить об атрибутах файла после закачки его, на сервер. Обязательно должна быть разрешена запись в файл. После того как служебные переменные определены, считаем наш файл в массив $list. При этом в него будет считан весь файл целиком, вместе с кодом, который сейчас исполняется.

Сам код счетчика нас в данном случае не интересует, нужно просто добраться до счетчика, который находится в этом же файле, проанализировать его, увеличить или сбросить (в зависимости от значения) и принять решение о выводе информации (у нас это, напомню, один из трех баннеров) на экран.

Следующая часть кода:

```
<?
$set = trim(str_replace ("\n","", $list[1]));
$set = trim(str_replace ("//","", $set));
$set++;
if ($set > $nomer_ban or $set < 1) { $set = 1; }
?>
```

В переменную $set считываем второй элемент из массива $list. Напомню, массивы начинают свою нумерацию с ноля, а значит, второй элемент массива будет обозначаться числом 1. Параллельно нужно удалить перевод строки и возможные пробелы в начале и в конце полученной строки.

Дальше нужно удалить оператор комментария (//) и пробел после него. Просто вырежем их, так как они нам не нужны.

В результате этих действий в переменной $set содержится число, соответствующее предыдущему показанию счетчика. Поступаем с ним так, как и положено с хорошим счетчиком, т.е. увеличиваем и проверяем, не вышел ли он за отведенные ему пределы. Если вышел, сбрасываем на единицу, если нет — работаем дальше.

Теперь надо, в зависимости от значения счетчика, вывести баннер. Сделать это можно, например, вот так:

```
<?
if ($set == 1) { ?> Тут код первого баннера <?
if ($set == 2) { ?> Тут код второго баннера <?
if ($set == 3) { ?> Тут код третьего баннера <?
?>
```

Это еще не все. После того как баннер выведен на экран, необходимо записать показания счетчика в отведенное для него место в файле с кодом, например:

```
<?
$list [1] = "// $set \n";
$str = implode("", $list);
$fp = fopen($name_file, "w");
if ($fp) { $fw = fwrite($fp, $str); fclose($fp); }
?>
```

Восстанавливаем второй элемент массива с новым значением счетчика. Переводим весь массив в одну переменную $str и за-

писываем полученную строку в файл. Запись идет в себя, но это никак не сказывается на работе программы.

Вот таким несложным способом легко добиться очень большой экономии времени. Имея готовое решение, больше не надо беспокоиться, куда заносить нужные данные. Теперь вы сможете хранить их непосредственно в коде своей программы.

И в конце несколько слов о том, как применить полученный код. Его надо полностью скопировать в отдельный файл, указать в начале количество баннеров и имя файла со скриптом, затем закачать на сервер, изменив атрибуты для разрешения записи. Вызывать этот файл нужно в том месте HTML-кода, в которое должен быть вставлен код баннера. Для этого можно использовать оператор PHP include ("имя файла"); Конечно, все расширения файлов должны быть phtml, php или php3 — в зависимости от настроек вашего сервера и ваших предпочтений.

Ах, баннеры, баннеры...

Не секрет, что в наше время интернет-страницы просто переполнены графикой. Без этого уже трудно представить себе какой-нибудь популярный проект. Одна часть этой графики — дизайн сайта, другая — баннеры (рекламная информация других сайтов в виде графической, часто даже анимированной, картинки). Посетителей это может раздражать, но все же хорошо сделанный баннер привлекает внимание к себе и, как следствие, к рекламируемому ресурсу. Мы поговорим не о самих баннерах, а о системах управления ими.

Если вам удалось сделать хороший, полезный и посещаемый интернет-проект, значит, вы неизбежно сталкивались с баннерами. Часто Web-мастеры предлагают обмен, иногда самому хочется разместить свой баннер на каком-либо ресурсе. В любом случае очень полезно иметь информацию о том, сколько раз ваш баннер был показан и показывается ли он вообще. Это избавит от необходимости раз в неделю (день, месяц, год и т.д.) проверять, работает ли ваш баннер на чужом сайте или давно удален.

Первое, что необходимо сделать, — определиться с кодом баннера. Код выглядит, как правило, так:

```
<a href="http://myhost.com"><IMG SRC="http://myhost.com/banner.gif" alt="Мой баннер" border="0"></a>
```

Конечно, это только шаблон, но с его помощью мы сможем построить то, что будет нужно. А нужно нам будет знать, сколько раз показывается тот или иной баннер на определенном сайте и, допустим, время последнего показа.

Для этого надо выделить один файл учета. Структура файла выглядит следующим образом: каждая строка соответствует одному из сайтов, на котором показывают ваши баннеры. Разделителем между строками является, как обычно, перевод строки. Разделителем в строке между информационными данными можно выбрать символ ^. Этот символ не встречается в адресной строке браузера и нам очень подходит.

Структура строки файла базы данных:

Адрес ^ время последнего посещения ^ количество посещений

Таким образом, получится три поля, которые нужно в дальнейшем будет менять в зависимости от того, откуда вызывается баннер. Чтобы это знать, надо ввести в код баннера специальный параметр — идентификатор сайта. Для каждого из сайтов, участвующих в обмене баннерами, идентификатор должен быть различным.

Чтобы обработать этот идентификатор, вам потребуется вызывать не сам баннер, а специальный скрипт PHP, который сначала обработает все данные, а затем выведет на экран нужный баннер. Вот что у нас получилось из кода баннера:

```
<a href="http://myhost.com">
<IMG SRC="http://myhost.com/banner.phtml?id=1"
alt="Мой баннер" border="0"></a>
```

Как видите, поменялось имя файла и добавился параметр id. Теперь вызывается не непосредственно сам баннер, а скрипт banner.phtml, который и ведет статистику показов и времени по-

сещения. В конце скрипта должен обязательно быть переход на баннер вот в таком виде:

```
header ("Location: banner.gif");
```

Теперь перейдем к вопросу о ведении статистики. Несложно сделать простой подсчет показов — достаточно считать в массив базу данных посещений, и затем увеличить на единицу ячейку массива с индексом, соответствующим переменной id. Это наш идентификатор.

Однако у нас более сложная задача — организовать возможность просмотра всей статистики в дальнейшем и запомнить время последнего показа баннера. Значит, необходимо записывать в базу и время, и адрес сайта, с которого вызывается баннер. Адрес мы будем запоминать только для того, чтобы в дальнейшем можно было узнать статистику показов, а в самой процедуре подсчета он не участвует.

Итак, наш скрипт прежде всего должен проверить на правильность полученную переменную $id. Если она не больше установленного значения и не меньше ноля, то все нормально, иначе присваиваем переменной $id ноль или выводим сообщение об ошибке (как кому как нравится). Значение переменной Sid с индексом ноль нужно специально зарезервировать для таких случаев, когда помещен неправильный код.

Если же все нормально, а так чаще всего и бывает, скрипт должен считать в память файл с базой данных:

```
$adds = "url.txt";

$txt = file($adds);
```

Вся информация оказывается в массиве с именем $txt. Идентификатор указывает на индекс элемента массива, содержащего данные сайта, с которого вызван баннер. Обратиться к этим данным очень просто:

```
$str = trim(str_replace ("\n","", $txt[$id]));
```

Одновременно мы удалили пробелы в начале и в конце строки и символ перевода строки.

Можно при необходимости проверить, есть ли такая учетная запись. Дальше надо разложить информацию на три составляющие — адрес, время последнего доступа и количество показов. Вот как это проще всего сделать:

```
list ($add, $time, $counter) = split ("^", $str);
```

Мы использовали наш разделитель — ^. Вся информация извлечена, теперь ее надо обновить и снова записать в файл. Для этого нужно просто увеличить переменную $counter на единицу, считать текущее время и присвоить его значение.

Если со счетчиком все ясно, то текущее время требует пояснения. Лучше и проще всего считывать его в формате Unix (см. приложение 6). Полученное значение будет равно количеству секунд, прошедшему с 1 января 1970 года. Это количество секунд очень легко преобразуется в дальнейшем в конкретные даты, а хранить его еще проще, так как это, по сути, просто большое число.

Получить его можно так:

```
$time = time(void);

$counter++;
```

Заодно увеличили показания счетчика. Теперь осталось только записать всю информацию снова в файл. Объединяем строку и заносим полученное значение в нужную ячейку массива (напомню, на нее указывает идентификатор в переменной $id):

```
$txt[$id] = $add."^".$time."^".$counter."\n";
```

И тут тоже не забываем о нашем разделителе и о переводе строки в конце для правильного считывания в дальнейшем. Перед тем как записать весь массив в файл с базой данных, необходимо сначала объединить массив в одну строку:

```
$str = implode("", $txt);

$fp = fopen($adds, "w");

if ($fp) { $fw = fwrite($fp, $str); fclose($fp); }
```

Вот и все, так как все поставленные задачи мы выполнили.

Я не привел весь код скрипта, а по порядку объяснял все действия как можно подробнее, чтобы вы не занимались бездумным копированием, а постарались самостоятельно развить предложенные мной идеи.

Счетчик посещений с выводом информации на экран

В качестве компенсации я предлагаю вам готовый скрипт вывода статистики на экран. Для этого достаточно вызвать в браузере файл stat.phtml.

Вот его код:

```
<?
$adds = "url.txt";
$txt = file($adds);

$i = 0;
while ($i <= count ($txt)):
$temp = trim(str_replace ("\n","", $txt [$i])) ;

list ($add, $time, $counter) = split ("^", $str);
echo $add." - ".$counter." : ".$time."<br>";

$i++;
endwhile;
?>
```

Как видите, еще проще. Единственная трудность — перевести формат времени в разумный вид, но для этого существует масса хороших и не очень алгоритмов. Их вы уже найдите сами — пусть это будет в качестве домашнего задания.

Счетчик сессий

Попробуем сделать счетчик сессий:

```
<?
session_name ("virtualbrest") ;
session_start ();
session_register("counter");
$counter = @$counter +1; ?>
<html><body>Нажмите кнопку "Обновить",
чтобы увеличить счетчик
<br>Счетчик: <?=$counter?>
</body></html>
```

В первой строке кода мы инициализируем имя сессии для SID, чтобы в дальнейшем можно было легко различать разные сессии. Если этого не сделать, PHP автоматически откроет сессию с именем PHPSESID. Функция session_start () инициализирует сессию, а функция session_register("counter") регистрирует в сессии переменную $counter.

Если запустить данный код и несколько раз обновить окно браузера, мы увидим, как счетчик увеличивается.

Создание динамического меню

Рассмотрим, как легко можно организовать выбор нескольких позиций для отправки формы. Такое часто бывает нужно для навигации по сайту, но конкретную задачу я не ставил, а только описал общие принципы.

В том же каталоге, где будет работать этот скрипт, нужно создать файл logo.txt и внести в него нужное количество строк. Именно на основании информации из него и будет построено меню.

```
1 <?
2 $lin = file ("logo.txt");
3 $a = count ($lin);
4 if ($a < 11)
5 {
6 $size = $a;
7 } else
8 {
9 $size = 10;
10 }
11 ?>
12 <P>
13 <select name="D1" size=<? echo $size; ?>
14 <? $i = 0;
15 while ($i < $a) :
16 echo "<option value=".$i.">".$lin[$i]."</option>";
17 $i++;
18 endwhile; ?>
19 </select><br>
20 <input type="submit" value="submit" name="B2">
```

Сначала все данные считываются из файла в массив (строка 2). Далее, после определения размера полученного массива (строка 3), определяем размер меню (строки 4—13). Если количество пунктов менее 11, оно становится размером (строки 4—6). Если больше — размер будет равен 10, а браузер добавит полосу прокрутки (строки 7—10). Если нужно, чтобы меню состояло всего из одного пункта и было выпадающим, нужно переменной $size присвоить значение 1.

Определившись с размером, в цикле выводим весь массив таким образом, чтобы его значения были пунктами меню (строки 15—18). Осталось только добавить кнопку для отправки данных полученной формы (строки 19—20).

Только не забудьте вставить этот код между тегами <form> и </form>, а затем определиться, куда именно будут отправлены данные и каким именно методом (см. приложение 5).

Подсчет количества обращений к пунктам меню

Рассмотрим, как организовать подсчет обращений к пунктам динамического меню, на примере подсчета скачанных файлов.

Сначала создадим форму:

```
<form method="POST" action="post.phtml">
<? $sll1 = file("url.txt"); ?>
<select name="ur" size="1" >
<option value="0"><? echo $sll1[0]; ?>: -1</option>
<option value="1"><? echo $sll1[1]; ?>: -2</option>
<option value="2"><? echo $sll1[2]; ?>: -3</option>
<option value="3"><? echo $sll1[3]; ?>: -4</option>
</select>
<input type="submit" value="смотреть" name="Bl" >
</form>
```

В этой форме всего четыре пункта (рис. 21), но вы можете задать больше или меньше в зависимости от того, сколько нужно.

Информация считывается из файла url.txt, соответственно его нужно создать и ввести туда четыре ноля через ввод (перевод строки) и, закачав на сервер, установить права доступа.

Дальше создаем файл для обработки нажатия — post.phtml:

```
if ($ur == "") { $ur = 0; }
$add = "url.txt";
$sl = file($add);
```

Рис. 21. Полученная форма

```
/* следующие четыре строки — адреса к нашим пунктам
меню */
$u[0] = "http://-1";
$u[1] = "http://-2";
$u[2] = "http://-3";
$u[3] = "http://-4";

$i = 0;
while ($i <= count ($u)):
$sl[$i] = trim(str_replace ("\n","", $sl[$i]));
if ($ur == $i)

{

$url = $u[$i];

$sl[$i]++;

}
$i++;
endwhile;

$sav = ""; $i = 0;
while ($i <= count ($u)):
```

```
$sav = $sav.$sl[$i]."\n";
$i++;
endwhile;

$sav = str_replace ("\n\n","\n", $sav);
$fp = @fopen ($add, "w+") ; if ($fp) { $fw = @fwrite{$fp,
$sav); @fclose($fp); }

header ("Location: ".$url);
?>
```

В этом коде все очень просто. Сначала считываем два массива: один — из файла, второй формируем в коде путем ввода адресов. Далее обрабатываем все при помощи цикла и, если индексы совпадают, увеличиваем показание нужного счетчика. Потом просто записываем весь массив снова в файл, но уже с новыми данными, и, наконец, перенаправляем посетителя на выбранный им пункт.

Блокируем доступ к файлу

Иногда возникает ситуация, при которой нельзя допустить, чтобы две операции выполнялись одновременно. Рассмотрим это на примере неоднократно описанного уже нами счетчика посещений.

Чтобы записать новое значение показания счетчика в файл, нужно его сначала обнулить, например, так:

```
$f = fopen("counter.txt","w+"); /* открыли файл на запись
и обнулили его */
```

Параметр w+ указывает, что перед записью в файл его необходимо обнулить (удалить все его содержимое). Это очень удобно, так как позволяет не заботиться о содержимом файла, в который предполагается сделать запись.

Однако как бы быстро после этого ни происходила запись нового значения, нет никакой гарантии, что в это время другой процесс PHP, вызванный действиями другого посетителя, в это время не считает пустой файл и не присвоит переменной счетчика значе-

ние ноля. После этого второй процесс запишет в файл единицу, так как будет считать, что предыдущее значение равно нолю.

Конечно, для возникновения такой ситуации необходимы специфические условия, но возможное в теории может случиться и на практике.

Решение этого вопроса очень простое. В PHP есть функция flock(descriptor, mode). Используется эта команда именно для блокировки доступа к определенному файлу (он должен быть открыт, например, командой fopen). Параметр mode может принимать такие значения:

- 1 — другие процессы могут отрыть этот файл только в режиме чтения;
- 2 — блокировать файл, т.е. другие процессы ничего не могут;
- 3 — разблокировать файл.

Для нашего примера нужно написать так:

```
flock($f,2); // заблокировали файл
```

И когда блокировка перестает быть нужна:

```
flock($f,3); // разблокировали файл
```

Если вы забудете снять блокировку, ничего страшного не случится, так как она снимется при закрытии файла. Файл же будет закрыт автоматически при закрытии процесса (когда посетитель закроет окно браузера).

Итак, если вы примените этот способ блокировки, процесс, инициированный другим посетителем сайта, будет ждать того момента, когда блокировка будет снята. И только тогда процесс пойдет дальше.

Стоит заметить, что в последних версиях PHP этот процесс автоматизирован, однако знать слабые стороны нужно.

СОВЕТ

Если у вас стоит ограничение на занимаемое сайтом место, следите за его объемом очень внимательно: если скрипту не

хватит места, чтобы сделать запись, он обнулит файл и ничего в него не запишет. А все ваши прежние данные будут просто утеряны. Часто в объем сайта входят log-файлы, почта и т.д. Это тоже надо учитывать в своей работе.

«Грабим»[1] странички

Не бойтесь, ничего криминального в этом нет. Речь идет о переносе информации со страницы в интернете на вашу страницу. Нам понадобится «подопытный кролик», на роль которого я предлагаю выбрать сайт http://subscribe.ru.

Там много полезной информации, и есть то, что нашим посетителям может действительно пригодиться. Я имею в виду список рассылок, недавно переведенных в категорию «серебряных».

Сделаем так, чтобы посетители нашего сайта смогли подписаться на понравившиеся им рассылки сразу на нашем сайте.

Сначала нам потребуется адрес сайта, с которого будем «грабить» информацию. Например: http://win.subscribe.ru/catalog/latest.

ВНИМАНИЕ

На момент написания книги все указанные тут адреса были работоспособными. Однако это не значит, что они останутся такими и в дальнейшем, потому что, как известно, интернет-адреса достаточно часто меняются. В этом случае материал может служить хорошим теоретическим пособием и все равно не теряет своего смысла.

По указанному адресу мы находим все переведенные в категорию «серебряных» рассылки. Причем список постоянно обновляется, оставаясь таким образом актуальным всегда, и в коде мы должны будем учесть эти изменения.

Сначала приведем весь код, затем поясним его.

---

[1] «Грабить» от англ. grab (хватать, захватывать). *(Примеч. ред.)*

```
<?php
// начало
$link = "http://win.subscribe.ru/catalog/latest";
$file = @fopen($link, "r");
if ($file) {
$rf = fread($file, 200000);
fclose($file);
} else
{
echo "<h3 align=center>Извините, запрошенная страница
временно не доступна!</h3>
<center>
<IMG src=http://virtual.bresttelecom.by/banner.jpg
width=468 height=60 border=0 alt=\"Виртуальный Брест\">
</A><br><br>";
}

// этап 1
$rf = trim (chop ($rf));
$s = strpos($rf, "<!-noindex-><FORM", 0);
$rf = substr($rf, $s);

// этап 2
$s = strpos($rf, "<!-/noindex-><table");
$rf = substr($rf, 0, $s);

// этап 3
$rf = str_replace ("/catalog/","http://win.subscribe.ru/
catalog/",
$rf);
$rf = str_replace ("/archive/","http://win.subscribe.ru/
archive/",
$rf);
$rf = str_replace ("ACTION=/member/
```

```
quick","ACTION=http://win.subscribe.ru/member/quick", $rf);
$rf = str_replace ("/img/money2.gif","http://
win.subscribe.ru/money2.gif", $rf);
$rf = str_replace ("/img/all4.gif","http://win.subscribe.ru/
af.gif", $rf);
$rf = str_replace ("/img/af.gif","http://win.subscribe.ru/
af.gif", $rf) ;

// этап 4
echo $rf;
?>
```

Теперь опишем код. В самом начале нам нужно скопировать содержимое страницы. Записываем ее адрес и открываем по нему соединение. Если соединение успешно установлено, можно считать весь файл (указываем 200 000 байтов для считывания, что явно больше размера открываемого файла). Если произошла ошибка открытия, предупреждаем об этом посетителя и выводим ему что угодно, например баннер.

Далее рассмотрим код по обозначенным этапам.

Этап 1 начинается со строки:

```
$rf = trim (chop ($rf));
```

Этой комбинацией мы значительно уменьшим объем обрабатываемых данных, так как уберем повторяющиеся пробелы, а также пробелы в конце и в начале файла.

Потом нам нужно определиться с местом, из которого будем выводить информацию:

```
$s = strpos($rf, "<!-noindex-><FORM", 0) ;
```

Эта команда позволяет найти номер позиции указанной последовательности символов в строке, в которую мы считали весь код файла. Результат помещается в переменную $s.

Следующей строкой удаляем все, что находится перед этой комбинацией (в том числе и баннеры):

```
$rf = substr($rf, $s);
```

Этап 2. Делаем почти то же самое, что и на первом этапе, но только для конца файла. Файл оказывается обрезан с начала и с конца так, как мы задаем. Обратите внимание, что в данном случае все оказалось очень просто, но иногда приходится применять другие методы для выделения кода, границы которого иногда трудно определить.

В результате этой обработки у нас уже есть почти все, что надо. Теперь можно было бы просто вывести информацию на экран нашему посетителю, но есть один нюанс, который нужно учитывать: ссылки не абсолютные, а относительные.

Однако относительных ссылок не так много, поэтому проблема решается просто.

Этап 3. Заменяем то, что есть, на то, что нам нужно. Например:

```
$rf = str_replace
("/catalog/","http://win.subscribe.ru/catalog/",    $rf);
```

Так мы заменяем во всей строке $rf относительные ссылки на абсолютные. Точно так же поступаем с остальными ссылками, которые встречаются в коде страницы. Это не очень оптимальное решение, но достаточно точное.

Этап 4. Выводим результат на экран посетителю. А этот результат — нужный нам код HTML-страницы, который и будет отображен браузером.

Если вы хотите интегрировать этот код в код своей страницы, вам скорее всего придется сделать еще одно — «подогнать» ширину таблиц. Ничего сложного в этом нет: находим, какой параметр отвечает за размер страницы, и заменяем его значение на пустую строку.

Перед запуском скрипта убедитесь в том, что все указанные адреса действительно существуют и содержат необходимую информацию в нужной форме. Иначе ваши посетители будут очень недовольны вашей работой.

Голосование на сайте

Голосование — это средство узнать мнение ваших посетителей по разным вопросам. Сами вопросы могут быть как полезными, так и бесполезными, но эту тему мы рассматривать не будем и для нашего примера возьмем простой вопрос: «Ваше мнение о сайте?» Варианты ответов предоставим следующие:

- Отлично!
- Нормально.
- Мне все равно.
- Это что-то страшное.

Можете сами продолжить список.

Для работы нам понадобятся три файла. В первом файле будем спрашивать посетителя о его мнении, во втором — хранить результаты и в третьем — выводить их и обрабатывать.

Острой необходимости сохранять все эти функции в разные файлы нет, но для простоты и удобства сделаем именно так. Первый файл будет иметь имя index.phtml, второй — golos.txt и третий — golos.phtml (вы, конечно, можете называть свои файлы, как хотите, только расширения должны быть такими, как я указал). Создайте эти три пустых файла.

Файл Index.phtml. Вы можете назвать этот и другие наши файлы так, как вам угодно, только не меняйте расширений. Дизайн и оформление предлагаю вам сделать самостоятельно, я не буду обращать на это внимание.

Прежде всего сделаем форму. Можно воспользоваться любым редактором HTML, они неплохо делают сами формы, но я привожу уже готовый и работоспособный код:

```
<form method="POST" action="golos.phtml">
<table border="1"><tr><td><table border="0">
<tr><td>Ваше мнение о сайте?</td></tr>
```

```
<tr><td><input type="radio" name="answer" value="1">
Отлично!</td></tr>
<tr><td><input type="radio" name="answer" value="2">
Нормально</td></tr>
<tr><td><input type="radio" name="answer" value="3">
Мне все равно</td></tr>
<tr><td><input type="radio" name="answer" value="4">
Это что-то страшное!</td></tr>

<tr><td><input type-"Submit" name="vote" value="От-
править"></td></tr>
<tr><td><input type="Submit" name="result" value="Смот-
реть результат"> </td></tr></tablex/td></tr></table></
form>
```

Здесь нет ни строчки кода PHP, но мы задали файлу расширение PHTML, чтобы в дальнейшем в этот файл можно было вставить PHP-код. Получилась такая форма (рис. 22).

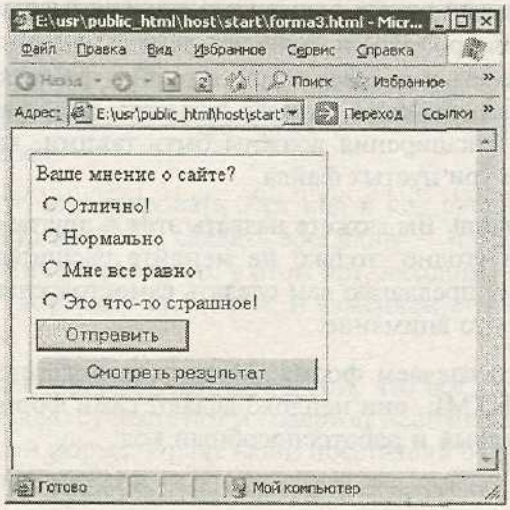


Рис. 22. Полученная форма для голосования

В HTML-коде все просто, мы предлагаем указать с помощью переключателя нужный вариант ответа и нажать на кнопку «Отправить».

Этот код можно легко вставить на любую страницу и внести в него нужные изменения. И самое главное — данные из формы должны передаваться PHP-скрипту, обрабатывающему результаты, поэтому имя файла, присваиваемое атрибуту `action`, должно соответствовать имени файла со скриптом.

Все имена форм, присвоенные в этом файле, станут соответствующими переменными в скрипте PHP, в который отправляются данные формы.

Файл golos.txt. В этом файле необходимо определиться, в каком формате мы будем хранить вводимую информацию. Проще всего организовать четыре строки (по количеству вариантов ответов), номера которых будут соответствовать номеру выбранного варианта ответа. Информацию из файла golos.txt можно считывать с помощью команды **PHP** ввода файла в массив — `file($array)`. Массив всегда начинается с нулевого индекса, поэтому первая строка нашего файла не будет использована и в нее можно ввести все, что угодно, например строку «Результаты голосования».

Далее введите еще четыре строки с нолями. Не забывайте нажимать ввод для перевода строки. Это — результаты (начальные) нашего голосования и, пока никто не проголосовал, они нулевые.

Файл golos.phtml. Мы дошли до самого главного и интересного. Сначала приведу весь код скрипта, а потом поясню его.

```
1 <?php
2 $file = "golos.txt";
3 $a = file($file) ;
4 $i = 1; $fi = count($a) ;
5 $n = 0;
6 while ($i <= $fi):
7 $a[$i] = trim(str_replace ("\n","", $a[$i]));
8 $n = $n + $a[$i];
9 $i++;
10 endwhile;
```

```
11 if ($answer != "") {
12 echo "<br>Спасибо, Ваше мнение учтено";
13 $a[$answer]++; $n++;

14 $rez = "Результаты голосования\n".$a[1].
"\n".$a[2]."\n".$a[3]. "\n".$a[4];
15 $fp = @fopen($file,"w");
16 if ($fp) { $counter = fputs{$fp,$rez); fclose($fp); }
17 else { echo "Произошла ошибка записи результатов!"; }

18 } else { echo "<br>Результаты голосования"; }
19 echo "<br>Отлично! - <b>".$a[1]."</b>";
20 echo "<br>Нормально - <b>".$a[2]."</b>";
21 echo "<br>Мне все равно - <b>".$a[3]."</b>";
22 echo "<br>Это что-то страшное! - <b>".$a[4]."</b>";
23 echo "<br><br>Всего проголосовало: ".$n;
24 ?>
```

Этот код не самый лучший и эффективный, зато логичный, простой и работоспособный. Вариантов реализации много, но мне не хотелось использовать что-то готовое, и поэтому я, не особо заботясь о дизайне (это вы и сами можете сделать), написал свой код, что заняло вместе с настройкой и отладкой около получаса.

Скрипт ориентирован на конкретно поставленную задачу, но вам ничего не стоит переделать его для своего количества вариантов ответов на вопрос. Теперь о том, как все работает.

Сначала мы задаем имя файла результатов (строка 2) и считываем результаты голосования в массив данных с именем $a (строка 3). Дальше идет цикл (строки 6—10), в котором мы обрабатываем полученный массив таким образом, чтобы он не содержал символов перевода строки и пробелов (строка 7). Удалять символы ввода и пробела необходимо для преобразования считанных данных из символьной строки в целое число. Это можно сделать разными методами, но в этом случае просто удаляются символы "\n" (что в PHP соответствует переводу строки) и обрезаются пробелы с начала и с конца строки функцией `trim()`. Результаты заносятся назад в массив, но уже в виде целочисленного значе-

ния, которое можно суммировать, увеличить на единицу, делить и т.д. Нас будет интересовать увеличение на единицу определенного элемента массива, индекс которого хранится в переменной $answer, которая пришла к нам по наследству из формы. Параллельно ведем подсчет количества проголосовавших, что несложно, так как это просто сумма значений нашего массива (строка 8).

После обработки полученного массива скрипт должен принять решение в соответствии с действиями посетителя — или показать результаты, или добавить голос в соответствующую позицию. Достигается это проверкой переменной $answer (строка 11), в которой сохраняется значение выбранного посетителем сайта варианта голосования. Если эта переменная пуста, значит была нажата кнопка показа результатов, и скрипт пропустит блок подсчета голоса (строки 14—16). Если переменная $answer не пуста, в ней содержится номер выбранного варианта голосования, а значит, мы можем просто увеличить на единицу значение нужной ячейки массива (строка 13). Кроме того, нужно увеличить значение количества проголосовавших, чтобы учесть голос только что проголосовавшего человека (строка 13).

Когда нужная ячейка массива увеличена, нужно записать результаты в файл. Для этого сначала открывается соединение с файлом (строка 15). Символ w указывает на необходимость очистки содержимого файла перед записью. Если соединение с файлом установлено, в файл записывается предварительно отформатированное значение переменной $rez (строка 16), а если не установлено — выводится сообщение об ошибке (строка 17).

Переменная $rez формируется следующим образом: значение всех ячеек массива (кроме самой первой — нулевой, которая не используется) объединяется таким образом, чтобы разделителем был символ перевода строки. Это позволит в дальнейшем корректно считать полученный таким образом файл.

Для объединения строк в PHP применяется точка. Обратите внимание, что наш массив из целочисленного перешел в разряд символьных.

И наконец, пришла пора вывести результаты на экран. Перевод строки в HTML осуществляется при помощи тега
.

Когда отформатированные результаты выведены, скрипт заканчивает свою работу. Этот блок можно организовать по-разному, но примененный мной метод самый простой. Лучше всего оформить результаты в таблице, тогда выведенная информация будет смотреться красивее. Обратите внимание: эта часть скрипта выполняется в любом случае, как и считывание данных из файла. Таким образом, достигается гарантированный вывод результатов на экран.

Вот и вся работа. Надеюсь, что вы теперь без труда сможете организовывать опросы своих посетителей на различные темы. Кроме того, голосование можно оснастить другими полезными функциями: например, подсчетом процентного соотношения голосов, графическим представлением результатов, запоминанием IP-адреса опрошенного, чтобы дважды не голосовали и т.д. Только в таких случаях вам придется проявить собственную фантазию и знание элементарной алгебры.

Гостевая книга

Когда сайт крепко становится на ноги и выходит из стадии младенчества, Web-мастер задается вопросом: как узнать, что именно хотят его посетители, что нравится и что не нравится им на сайте? Все это легко исследовать, установив так называемую «гостевую книгу», в которую посетители будут заносить свои впечатления и пожелания. Созданием гостевой книги сейчас и займемся.

Нам понадобятся два файла. В первом (назовем его guest.phtml) разместятся форма для ввода данных и сам скрипт гостевой книги, а во втором будут храниться результаты введенных данных в специальном формате. Не забывайте, что после закачки на сервер этих двух файлов на файл с результатами (guest.txt) нужно будет установить атрибуты, разрешающие запись в файл. Сделать это можно практически в любом из FTP-менеджеров, проверив атрибуты уже закачанного на сервер файла и установив в настройках все галочки.

Теперь перейдем к коду. Он должен быть добавлен в файл guest.phtml. Сначала сделаем форму для ввода данных:

```
<h2>Гостевая книга</h2>

<form action="guest.phtml" method="post">
Введите e-mail: <input type="text" name="email">

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

Сообщение: <br><textarea name="msg" rows="10"
cols="40"></textarea>

<p><input type="submit" value="0тправить"></p>
```

В этой форме три поля — адрес электронной почты (переменная email), имя посетителя (переменная name) и сообщение (переменная msg) (рис. 23).

Рис. 23. Форма гостевой книги

После того как посетитель введет данные и нажмет на кнопку «Отправить», все эти переменные будут доступны нашему скрипту, причем значения переменных будут соответствовать введенным данным. Теперь нужно все это обработать.

Определим имя файла, в который будем записывать данные, и максимальное количество сообщений, которое может быть выведено на экран:

```
<?

$files = "guest.txt";

$qq = 50;
```

Дальше:

```
if (!$email) { $email = "нет"; }

$msg = substr($msg,0,999);

$email = substr($email,0,39);

$name = substr($name,0,39);
```

Здесь введенные данные обработаны таким образом, чтобы переменная адреса не была пустой (то есть если туда ничего не ввели, она равна строке "нет"). При этом каждая переменная обрезается, чтобы ограничить количество вводимых символов. Это нужно для защиты от действий вредных посетителей, которые ради баловства начинают вводить в гостевую книгу массу информации, которая никому не нужна.

Проверим, не являются ли сообщения или имя пустыми строками:

```
if ($msg != "" && $name != "") {
```

Если сообщение или имя не указаны вообще, скрипт ничего никуда не запишет, а просто продолжит обработку дальше и выведет сообщения гостевой книги на экран (см. ниже). Но если и имя, и сообщение введены, скрипт, прежде чем вывести данные на экран, должен сделать запись отформатированных данных в файл для сообщений:

```
$time = Date("h:i:M:d");
$soo = "\n<b>$time $name (<a href=\"mailto: $email \">
$email </a>) </b><br> $msg<hr>";
$fp = fopen($files, "a+");
$fw = fwrite($fp, $soo);
fclose($fp); }
```

В этом коде сначала определяем и форматируем время ввода сообщения. Потом формируем строку для записи в файл. Она представляет собой последовательность нужных переменных, отформатированных тегами HTML. За счет этого нам потом будет очень легко просматривать архив сообщений и выводить на экран нужный промежуток (если количество сообщений превысит 100, вы это оцените — очень удобно указать ссылку и смотреть сообщения, например, с 50-го по 80-е). После того как строка подготовлена, она записывается в файл. Дальше — вывод результатов записи:

```
$lines = file ($files);
$a = count($lines);
$u = $a - $qq;
for($i = $a; $i >= $u ;$i-) { echo $lines[$i]; } /* вывод результатов на экран */
?>
```

Обратите внимание, что в первый раз, когда посетитель попадает на страницу с гостевой книгой, обработка этого кода идет сразу, так как переменные имени и сообщения пусты.

Этим кодом в массив считывается файл сообщений, и при помощи цикла выводится на экран его содержимое. Если количество сообщений превысило наше ограничение, они просто не показываются. Новые сообщения всегда отображаются вверху, около формы для ввода, так как вывод идет снизу вверх по индексу массива. Это очень удобно, но при желании может быть изменено.

Вот и все. В файл для сообщений ничего записывать не нужно — он будет заполняться по мере ввода данных. Архив сообщений всегда будет доступен, если вы поставите ссылку:

```
<ahref=guest.txt>архив</a>
```

Обратите внимание, что код не учитывает ввод посетителем HTML-тегов. Это уже несколько иная задача, но делается легко, если нужно.

Данный скрипт может использоваться также в любом месте, где нужно узнать мнение посетителей, например, о статье, новостях и т.д. Также это — простейший форум.

Чат

Технология создания

Чат — это текстовой диалог в реальном времени между пользователями интернета. Каждый участник разговора «высказывается» с помощью клавиатуры, получая ответы в специальном окне. Чаты весьма популярны и являются залогом хорошей посещаемости сайта.

Моя основная цель в данном параграфе — рассмотреть основные проблемы создания чата на PHP и подтолкнуть вас к собственным исследованиям в этой области. Это не так уж и сложно, как кажется на первый взгляд, и при определенных обстоятельствах есть возможность сделать свой собственный чат, пользуясь только PHP, даже без поддержки баз данных.

Очень многие Web-мастеры мечтают сделать для своих посетителей хороший чат. Как правило, Web-мастеры идут проторенным путем, ищут в интернете исходные коды чатов и размещают их у себя на сервере. Непременное условие — обязательное наличие на сервере поддержки одного из языков программирования. Как правило, чаты предпочитают делать, используя CGI (например, на языке Perl). Такие чаты получаются достаточно быстрыми и надежными.

Но как быть тому, кто по ряду причин не может себе позволить запускать скрипты CGI на своем сайте? или не может до конца освоить один из языков программирования, поддерживающий CGI? или, если даже и освоил один из таких языков, просто не знает, как подступиться к проблеме?

Можно, конечно, пойти по легкому пути. Например, открыть свой чат на каком-нибудь бесплатном сервере. Разработка такого чата сводится к настройке уже готового скрипта: выбору цвета, расположения фреймов (см. ниже) и другим мелким деталям.

Однако есть, как всегда, и отрицательные стороны такого решения. Никто просто так вам не даст пользоваться своим чатом — придется «расплачиваться» тем, что в вашем чате будут показываться чьи-то баннеры. И нет гарантии, что чат не закроют. Но это единственный способ сделать чат для тех, у кого хостинг-провайдер не обеспечивает поддержку CGI.

Если же ваш сервер работает с CGI или PHP, вы легко можете найти в интернете массу хороших и не очень скриптов чатов и разместить их у себя на сайте, тогда вы уже ни от кого не будете зависеть. Но есть недостатки и у этого способа. Поскольку исходные коды таких чатов общедоступны, у них почти всегда обнаруживается какая-нибудь уязвимость, позволяющая злоумышленникам «взламывать» ваш любимый чат.

Если вы все же готовы воспользоваться уже написанным кем-то кодом чата, будьте готовы к двум проблемам — предварительной настройке чата (это иногда непросто) и к возможности «взлома» чата недоброжелателями через стандартные ошибки, которыми часто просто изобилуют бесплатные скрипты.

Если вас это не устраивает, придется идти по трудному и сложному пути создания собственного чата, чем мы и попробуем заняться.

Сначала придется решать проблемы авторизации пользователей в чате. Это одна из самых больших проблем, и не только в чатах, но и во многих других случаях программирования.

Готовых решений масса, но у них есть очень большой недостаток: знание принципа шифрования неизбежно приводит к появ-

лению алгоритмов расшифровки. Значит, вам придется разрабатывать свой собственный алгоритм шифрования паролей пользователей с условием невозможности расшифровки за разумный промежуток времени (надеюсь, ФСБ или ФБР не будут «взламывать» ваш чат).

Кроме того, желательно сделать так, чтобы даже при краже файла с паролями их невозможно было расшифровать. В моей практике был случай, когда ошибка одного скрипта привела к появлению возможности доступа к исходным кодам на сайте и к краже паролей.

Выход из такой ситуации — не хранить незашифрованные пароли вообще.

Правда, у такого решения есть большой минус. Если человек забудет пароль, его нельзя будет восстановить, так как пароль нигде не хранится незашифрованным. В этом случае нужно удалить регистрационную запись пользователя и предоставить ему возможность повторно зарегистрироваться. В данном случае необходимо использовать почтовый ящик, который пользователь должен указать при регистрации. Если пользователь не указал ящик, то он сильно рискует, ведь установить точно, действительно ли он является владельцем регистрационной записи с потерянным паролем, уже нельзя. Если же ящик указан, и указан правильно, на него следует отправить запрос об удалении регистрационной записи, и при положительном ответе запись убирается путем удаления строки из файла с регистрационными данными. Кстати, сам адрес электронной почты тоже можно шифровать, но уже обратимым методом, в отличие от пароля. Хотя иногда это лишнее.

Возникает резонный вопрос: если пароли будут шифроваться до такого состояния, что их нельзя (как мы договорились, нельзя — это значит нельзя за разумный промежуток времени, а вообще-то можно все, только долго) даже расшифровать, как же эти пароли будут проверяться на соответствие с хранимыми в базе зашифрованными паролями?

Все очень просто и элегантно. Я опишу на примере Windows 2000, в которой реализован примерно следующий алгоритм. При регистрации пользователь вводит свой пароль, который обраба-

тывается специальной функцией необратимого шифрования. Результат заносится в базу паролей вместе с учетными записями (имя пользователя, описание и т.д.). Когда пользователь входит в систему в следующий раз, Windows опять шифрует введенный пароль своей стандартной функцией и сравнивает результат с тем, который хранится у нее в базе данных паролей. Если результат шифрования введенного пароля и хранящегося в базе являются идентичными строками, пользователь авторизован, если нет — ошибка ввода пароля.

Таким образом, достигается большая конфиденциальность, так как пароль хранится не в открытом виде, а в виде уникального идентификатора строки, соответствующей паролю. И даже если база данных с хранимыми паролями будет украдена, толку от этого похитителю будет мало, ведь открытых паролей там нет, только зашифрованные. А обратной расшифровке такие идентификаторы не поддаются, так уж устроена функция шифрования.

Есть еще одна проблема авторизации, которую не поможет решить даже наш подход к хранению паролей. Дело в том, что простые пароли очень легко восстановить методом перебора. Пишется несложная программа, которая перебирает различные варианты (комбинации символов пароля), шифрует их и сравнивает результат шифрования с хранящимся в украденной базе. Это называется методом прямого перебора паролей (для этого существуют даже специальные словари).

Таким образом, очень легко «взломать» нестойкий пароль. О стойкости пароля сказано уже очень много до меня, но вкратце смысл таков: ни в коем случае нельзя применять для пароля смысловое сочетание букв. Другими словами, пароль должен казаться полной бессмыслицей постороннему человеку. Такой пароль гораздо труднее подобрать методом прямого перебора и практически невозможно перебором слов по специальным словарям. Очень хорошо, если в пароле присутствуют знаки препинания, чередуются маленькие и большие буквы, используются цифры и т.д. Рекомендуйте это своим посетителям.

Кроме авторизации очень важным является пункт настройки чата. Как правило, применяются три варианта настройки: на-

стройка перед каждым входом в чат, настройка чата для посетителя один раз с дальнейшим использованием cookie и настройка чата в любой момент.

На мой взгляд, самым удобным является разумная комбинация всех трех вариантов. Если пользователь хочет настроить чат для себя — пусть настраивает, а если не хочет и ему нужен вход в чат одним щелчком, то для таких посетителей необходимо предусмотреть загрузку конфигурации чата по умолчанию.

Кроме загрузки по умолчанию, желательно предусмотреть гостевой вход, при котором не потребуется даже регистрация. Сделать это проще всего так: создать пользователя с именем Гость, выбрав для пароля, например, слово guest, и посетителю, входящему без ввода учетных данных (логина и пароля), загружать именно эту учетную запись.

Зарегистрированным пользователям можно предоставить возможность настраивать самые различные параметры — цвет фона, шрифт и его размер, фразу для входа, время обновления окна чата, фильтр нецензурной лексики, отключение графических иконок и т.д. Все это должно присутствовать в окне авторизации или настраиваться в самом чате.

Как правило, все параметры настройки, введенные ник и пароль, передаются методом POST (см. приложение 5) в загрузчик, где после соответствующей обработки в браузер загружаются несколько фреймов[1].

Сделать чат без фреймов теоретически можно, но пользователям будет очень неудобно, так что вам просто придется освоить фреймовую структуру.

Как правило, чат состоит из трех-четырех фреймов. Экран делится на несколько частей по горизонтали и вертикали. В горизонтально разделенных фреймах обычно выводят информацию о количестве посетителей в чате (чаще — просто их список) и, собственно, саму текстовую (сопровождаемую графическими

[1] Фреймы — подокна в окне сайта. Информация в одном фрейме размещается независимо от другого и может иметь свою автономную полосу прокрутки. *(Примеч. ред.)*

смайликами) информацию, которой обмениваются посетители. В вертикально разделенных фреймах можно обеспечить окно для ввода текста, выбор его вида, цвета и т.д. Там же выбираются иконки, если они предусмотрены. Кроме того, в вертикально разделенных фреймах можно выводить и другую полезную информацию — рекламу, ссылки, счетчики и т.д.

Перед тем как фреймовая структура будет сформирована и передана браузеру, нужно очень многое предусмотреть. Кроме авторизации необходимо вести учет входов (для показа тех, кто находится в данный момент в чате), проверять соответствие дат и предусмотреть файлы журналов разговоров, если это нужно.

Кроме того, надо проверять и корректировать все введенные пользователем данные. Не рассчитывайте, что переменные, переданные вашему загрузчику методом POST, не смогут принять непредусмотренное вами значение. Ведь можно воспользоваться еще и методом GET (СМ. приложение 5), а это значит, что вашему загрузчику могут такое передать..! По этой причине обрабатывать полученные переменные необходимо по всем правилам. Другими словами, всегда проверяйте, соответствует ли полученная переменная нужному диапазону, и если нет — лучше всего присвоить ей стандартное значение. Кстати, желательно контролировать метод передачи данных. Если вы применяете метод POST, проверьте переменную окружения $QUERY_STRING. ЕСЛИ она не пустая, значит, вашему загрузчику пытаются что-то передать еще и при помощи метода GET. ЭТИ ПОПЫТКИ нужно пресекать.

Кроме самого загрузчика фреймовой структуры, данные необходимо передать в другие файлы чата. Если этого не сделать, все настройки чата будут утеряны, а авторизация потеряет всякий смысл, так как любой желающий сохранит ваш загрузчик у себя на диске, подредактирует его, запустит и войдет в чат, минуя все входные процедуры. Как минимум, это дает возможность ему быть невидимым в чате, как максимум -- входить под любым ником, что, согласитесь, неприятно. Чтобы такого не происходило, вам придется немного усложнить один или несколько фреймов, точнее — программ, отвечающих за вывод информации во фреймы. Каждому соответствует свой собственный файл, и в каждом, соответственно, своя программа. Вот этим-то програм-

мам и нужно передать все данные о пользователе, причем информацию о посетителе передать так, чтобы нельзя было явно ее изменить.

Информацию о цвете и настройках можно передать как обычно, методом GET. Она, как правило, не влияет на существенные настройки чата и все равно доступна для изменения. Есть возможность передавать все данные с помощью cookie. Это самый легкий метод плох тем, что посетитель без поддержки cookie или с отключенной такой поддержкой в настройках браузера не сможет пользоваться вашим чатом. Изменить файл cookie на диске можно, но это уже несколько сложнее, и специалист, который сможет это сделать, вряд ли заинтересуется вашим чатом. У него и так работы хватает :-)

Можно зашифровать информацию, записываемую в cookie одним из методов шифрования, поддающихся обратной расшифровке. Однако, как я уже говорил, иногда это плохо, так как не совсем надежно. Остается только передать пароль и информацию об авторизованном пользователе тоже через метод GET.

Передавать методом GET учетные записи незашифрованными нельзя, да это и не требуется. Я говорил о специальном цифровом идентификаторе введенного пароля. На основе его, ника пользователя и любой другой произвольной переменной можно построить еще один цифровой идентификатор, который уже можно передавать открыто методом GET через загрузчик фреймов. Расшифровать такой идентификатор невозможно, так как неизвестны методы, которыми он строился.

Для усиления криптостойкости и вводится дополнительная строка. Можно скомбинировать сочетание идентификатора пароля, ника и дополнительной строки так, как угодно. Это даст возможность построить разные системы шифрования. В любом случае методика шифрования должна быть недоступной всем желающим.

Когда все данные переданы, фреймовая структура построена в загрузчике, нужно опять перепроверить настройки чата, служебную информацию и восстановить учетную запись вошедшего. Делается это тем же методом, что и в Windows 2000 (см. выше): перебором всех учетных записей, хранящихся в базе

данных. Напомню, нам нужно знать несколько вещей: переданный методом GET уникальный идентификатор пользователя, построенный на основе идентификатора пароля (сам пароль нигде не хранится, только его идентификатор), ника и дополнительной строки.

Первое у нас уже есть, так как оно передается из загрузчика методом GET, остальное считывается из базы данных о пользователях. Итак, шифруем все подряд учетные записи точно тем же методом, каким шифровался идентификатор пароля, ник и дополнительная строка. Полученный результат сравнивается с переданным идентификатором пользователя. Если они не совпадают, сверяем следующую учетную запись в базе, если совпадают — пользователь авторизован повторно. Если перебор окончен, а учетная запись не найдена, значит, введен неверный пароль.

Такой подход выглядит немного сложным, но зато гарантирует правильную авторизацию и защиту от входов под любым выбранным ником.

Конечно, и этот способ авторизации не абсолютно надежен, ведь все, что сделал один человек, второй может исследовать и переделать. Однако мне такая защита кажется достаточной.

Описанная методика не очень сложна, но накладывает некоторые ограничения на сервер. Если учетных записей достаточно много, нагрузка будет довольно большая. Можно пойти более легким путем — авторизовывать пользователя по идентификатору не каждый раз, а только один, в самом начале. Дальше можно передавать более простой идентификатор методом POST (например, номер учетной записи в базе данных). Однако учтите, что это дает «взломщику» еще одну возможность несанкционированного проникновения в ваш чат, что нежелательно.

Теперь поговорим о чате как о едином механизме. С теми фреймами, которые отвечают за вывод информации о посетителях, все более-менее понятно. А вот окно, в котором выводятся сообщения, стоит рассмотреть подробнее. Средства HTML позволяют специальными методами заставить окно браузера периодически обновлять свое содержимое.

Страница с сообщениями посетителей будет обновляться в соответствии или с установками по умолчанию, или с изменяемыми настройками. Собственно, это самая простая часть всего чата, так как тут нужно только считать файл, в который заносятся сообщения посетителей, отбросить лишнюю часть сообщений (чтобы остались только введенные последними) и вывести все на экран. Можно также применить фильтр нецензурных выражений и ограничение на длину сообщения или отдельного слова. Вот и все хитрости для этой части фреймов.

Следующая часть чата — окно для ввода сообщений. Именно здесь необходимо проверять авторизацию пользователя по цифровому идентификатору. Авторизация будет проводиться всякий раз, когда сообщение будет отправляться в чат после нажатия кнопки «Отправить».

И если пользователь не авторизован, ввод сообщения становится недоступен. Он блокируется с выводом на экран предупреждающего сообщения.

В данном фрейме можно также организовать выбор цвета и типа шрифта, вывод иконок и т.д.

Самое сложное — применить технологию, при помощи которой определяется, кто находится в чате в данный момент. Если отказаться от этого сервиса, все значительно упростится, и чат очень слабо будет загружать сервер. Но ведь это неинтересно, так что этот вариант мы пропустим и перейдем к следующему, самому сложному в отношении программирования фрейму. В нем можно увидеть, кто и сколько времени присутствует в чате. Для реализации данной возможности необходимо и в этом фрейме авторизовать посетителя по уникальному идентификатору, причем делать это с определенной периодичностью. Ведь нужно проверять, не появился ли в чате новый посетитель и не покинул ли чат кто-то из присутствующих. Чем чаще такие проверки, тем больше нагрузка на сервер. Можно ограничиться одной авторизацией, а дальше передавать простой идентификатор по методу GET. Этот идентификатор можно подделать, однако программа из предыдущего фрейма, которая реализует полноценную проверку данных, не позволит вывести сообщения от пользователя с поддельным идентификатором.

Как правило, обновления можно делать один раз в 15—20 секунд, а можно добавить специальную кнопочку или ссылку, нажав на которую пользователь обновит окно с никами присутствующих в чате.

Время обновления задается еще на этапе загрузчика фреймов. В специальный файл вносятся ник входящего, время входа и контрольное время. Фрейм, обновляясь с определенной периодичностью, вызывает запись в файл с данными о присутствующих в чате новой информации, но только той, что принадлежит определенному пользователю. Информация о других посетителях остается неизменной, обновляется только персональное контрольное время. Заодно проверяется контрольное время других посетителей. Если разность контрольного и текущего времени больше установленного (как правило, в два-три раза больше, чем время обновления фрейма), значит, или с браузером этого посетителя нет связи, или он просто закрыл окно чата и ушел. В любом случае можно считать посетителя ушедшим и запись о нем стирается из файла данных о присутствующих в чате. Таким образом, эти фреймы всех пользователей обновляют свое собственное контрольное время, чтобы не быть удаленным кем-то другим, и удаляют всех, чье контрольное время превысило установленный максимум.

Дальше необходимо вывести на экран список ников, которые остались в файле. Это и будет список пользователей, присутствующих в чате в режиме онлайн.

Вот и все технологии. Точнее, конечно, не все. За бортом остались такие вещи, как персональные комнаты, отправка сообщений на ICQ и многое другое. Однако это признаки профессиональных чатов. Если вы хотите себе именно такой, вам придется воспользоваться платными программами или даже целыми наборами программ. Если же хотите получить удовольствие от собственной разработки и вам достаточно простого чата на собственном сайте, вы можете воспользоваться моими наработками.

Еще раз повторю, я не претендую на полноту и законченность своих суждений. Моя цель — побудить вас провести собственное исследование в области построения чатов. В следующем параграфе я приведу готовый код.

Свой чат — это просто

Рассмотрим вариант кода чата, максимально адаптированный для новичка в программировании. Все, что вам придется делать, — это внимательно выполнить приведенные инструкции, и у вас появится свой чат, на основе которого можно легко построить более сложную систему.

Итак, наш чат должен быть максимально быстрый и удобный, его инсталляция должна занимать не более пяти минут, при условии, что все исходные коды есть и они правильно набраны. Базы данных использовать не будем, только собственные возможности PHP версии более чем 4. Эта версия уже давно стала стандартом в Сети, так что мы можем смело воспользоваться открывшимися возможностями.

Создаем несколько файлов:

index.phtml

tools.phtml

header.phtml

banner.phtml

main_window.phtml

msg.phtml

userlist.phtml

Теперь создаем директорию memolog и в ней два файла — log_chat.txt и kto_chat.txt. Этим файлам, когда вы закачаете их на сервер, надо обязательно установить атрибуты, разрешающие запись.

Когда все готово, приступим к программированию.

Файл index.phtml. Это файл начальной страницы, которая откроется, когда посетитель введет адрес чата в браузере. Через эту страницу обязательно должен пройти каждый, чтобы чат мог авторизовать посетителя и присвоить ему выбранный ник.

Открываем этот файл в текстовом редакторе и пишем код:

```
<form method="POST" action="header.phtml">
<b>Введите ник:</b><br>
<input type="text" name="person"
size="14" maxlength="10"><br>
<p><input type="submit" value="Войти в чат" name="Bl">
</form>
```

Это обычная форма (рис. 24), данные из которой передаются в файл header.phtml. Никакого графического дизайна код не содержит, чтобы вы могли интегрировать этот вход в чат в свой собственный дизайн.

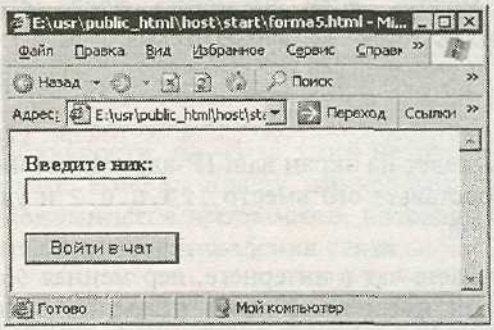


Рис. 24. Форма для начальной страницы чата

Атрибут name в теге <input> имеет значение person, и именно такую переменную ($person) получит наш загрузчик, которому форма передает данные. И именно в переменной $person окажется выбранный посетителем ник.

Файл **tools.phtml.** Служебный файл, в котором определяются нужные нам переменные и описываются пути к файлам. Кроме того, делаются стандартные для всех файлов чата функции, например, обращение к сессии с переменной chat_virtualbrest. Вот код этого файла:

```
<?
session_name ("chat_virtualbrest");
session_start();
```

Мы открываем сессию, после чего считываем переменную chat_virtualbrest, если она присутствует в системе.

```
if ( $ip == "127.0.0.2" ) { $server = 1; } else
 { $server = 0 ; }
```

Данный код позволяет определить, мы запускаем чат в интернете или на домашнем компьютере. Это бывает полезно знать, чтобы, например, не выводить на домашний компьютер баннеры или другую ненужную информацию. В качестве значения переменной Sip укажите свой локальный адрес вместо 127.0.0.2. Узнать его можно, если временно после строки $ip = getenv("REMOTE_ADDR"); добавить код:

```
echo $ip;
exit;
```

Программа выведет на экран ваш IP-адрес и остановит свое выполнение. Подставьте его вместо 127.0.0.2 и удалите ненужный больше код.

Если вы запускаете чат в интернете, переменная $server ложна, другими словами, имеет значение «0» (ноль). Если запуск чата произошел на домашнем компьютере, она равна «1» (единице). Теперь вы имеете возможность отказаться от какой-то части кода чата (с баннерами и другой ненужной информацией), выполнив условие:

if ($server) {то, что должно выполняться на домашнем компьютере} else {то, что должно выполняться в интернете}

Например:

```
if ( !$server ) { error_reporting(0); }
```

Этим мы отключили все сообщения об ошибках, если чат работает в интернете. Так поступать необязательно, но нашим посетителям будет приятно, что мы о них заботимся.

Следующей строкой считываем текущее время:

```
$time_nast = time(void);
```

Оно нам может понадобиться, и считывать его нужно каждый раз в этом служебном файле, который будет всегда выполняться.

```
$file_logchat = "memolog/log_chat.txt";
```

При помощи данной переменной присваиваем путь, по которому будет расположен log-файл чата. В него будет заноситься все, что пишут посетители.

Дальше:

```
$chat_number = 100;
```

```
$name_rob = "Robot";
```

Это служебные переменные. Первая ($chat_number) описывает время в секундах, за которое посетитель считается ушедшим, если с ним нет связи. Увеличение этого числа хотя и ведет к большей стабильности, но из-за этого не точно отражает состояние дел в чате. Имеется в виду, что человек уже ушел, а его ник все еще показывается программой, которая ожидает время его возвращения или восстановления связи.

Переменная $name_rob задает имя системного автоответчика, который будет иметь возможность вставить в общий разговор несколько слов, например кто пришел или ушел и т.д.

Задаем путь к файлу, в котором хранится список тех, кто в данный момент присутствует в чате:

```
$file_kto_in_chat = "memolog/kto_chat.txt";
```

Кроме времени в Unix-формате, нам понадобится и обычное время:

```
// $time = date("H:i");
```

Данная строка не зря закомментирована. Дело в том, что нет гарантии, что ваше локальное время совпадает с временем сервера в интернете. Если сервер расположен в том же часовом поясе,

что и вы, то раскомментируйте эту строку. Если же необходимо преобразовать время в ваш часовой пояс, то внесите в код следующие строки:

```
$time1 = date("H");

$time2 = date("i");

$time_s = 7

$time1 = $time1 + timer_s;

if ($time1 >= 24) { $time1 = $time1 - 24; }

$time = "$time1:$time2";

?>
```

В первой и во второй строках этого блока мы считываем в разные переменные показания часов и минут, потом в третьей строке вычитаем разницу локального времени и времени сервера, на котором работает чат (вы уж сами должны знать эту разницу, в нашем случае она равна семи). Определяем, не больше ли полученное число 24, и если это действительно так, то отнимаем 24. Осталось только объединить часы и минуты.

И не забудьте поставить закрывающий тег PHP:

```
?>
```

На этом все приготовления окончены, идем дальше.

Файл header.phtml. Очень важный и нужный файл. Его роль — сделать необходимые приготовления и проверить введенный ник, а затем передать управление непосредственно программе чата, сформировав при этом нужные фреймы.

Последовательно разберем его код.

В первой строке вызываем на исполнение служебный файл tools.phtml, описанный выше:

```
<?

include ("tools.phtml");
```

Во второй строке идет работа с переменными, нужными для авторизации. Работа ведется посредством сессий:

```
session_register("person", "pass");
```

Сессия запомнит эти переменные, и все, что в них сохранит программист, перейдет в другое окно.

Далее считываем в массив файл, в котором хранятся ники присутствующих:

```
$kto = file ($file_kto_in_chat);
```

Хранятся они там в определенном формате: сначала идет ник, потом в качестве разделителя символ «~» (собственно, не обязательно тильдой — просто она используется как наиболее уникальный и не встречающийся в переменных символ) и в конце строки — время последнего обновления связи с этим ником. Время хранится, как мы помним, в виде количества секунд, прошедших с 1 января 1970 года. Строки разделяются между собой при помощи символа перевода строки.

Начинаем проверять каждый ник, который записан в этом служебном файле:

```
$fi = count($kto); //количество записей в файле учета
$met = -1; /* метка присутствия ника в чате; -1 означает, что ник в чате отсутствует */
$i=0;
```

Открываем цикл проверки заходящего ника и в нем избавляемся от концевых пробелов и переводов строк в массиве с никами присутствующих в чате:

```
while ($i <= $fi);
$str = trim(str_replace ("\n","", $kto [$i]));
```

Далее раскодируем информацию об очередном нике:

```
if ($str)
{
list ($name_kto, $time_kto) = split ("~", $str);
```

Вся раскодировка заключается в раскладывании при помощи функции split по своим переменным имени ника ($name_kto) и времени последней связи с ним ($time_kto):

```
if { {$time_nast-$time_kto) > $chat_number and
$name_kto != $person)
```

А вот тут проверяем, если время связи с ником превышено, и если это входит не этот самый ник, нужно его удалить из списка тех, кто присутствует в чате. Делаем это:

```
{
$kto [$i] = "";
```

Для информации надо послать в чат сообщение, что этот ник ушел. Это даст возможность другим посетителям контролировать процесс входа и выхода других посетителей:

```
$sav = "$name_rob~$time~Ушел $name_kto~\n";
$fp = fopen($file_logchat, "a+");
if ($fp) { $fw = fwrite($fp, $sav); fclose($fp); }
}
```

Сначала формируем строку, которую отправим в окно чата. Ее мы составляем в соответствии с форматом, принятым нами для этого файла. Подряд идут несколько переменных, разделенных тильдой. В результате содержимое файла может выглядеть, как на рис. 25.

Потом открываем соединение с файлом чата, обеспечивая возможность записи туда информации (за это отвечает параметр a+ в функции fopen). Если соединение успешно открыто (оператор заключен в условие, а оно будет иметь значение истины, только если соединение успешно открыто), делаем собственно саму запись и закрываем соединение.

Если среди списка присутствующих в чате обнаружен входящий сейчас ник, присваиваем метке $met текущее значение переменной $1. Затем прекращаем обработку, присвоив $i значение 10 000 (предполагаем, что в файле учета меньше 10 000 записей), и устанавливаем метку $met, что ник уже есть в чате:

```
if ( $name_kto == $person ) { $met = $i; $i = 10000; }
```

Повторный ввод бывает, когда человек не по своему желанию покинул чат (чаще всего такое случается из-за потери или обрыва связи) и хочет снова войти, а его ник еще не удален из списка присутствующих..

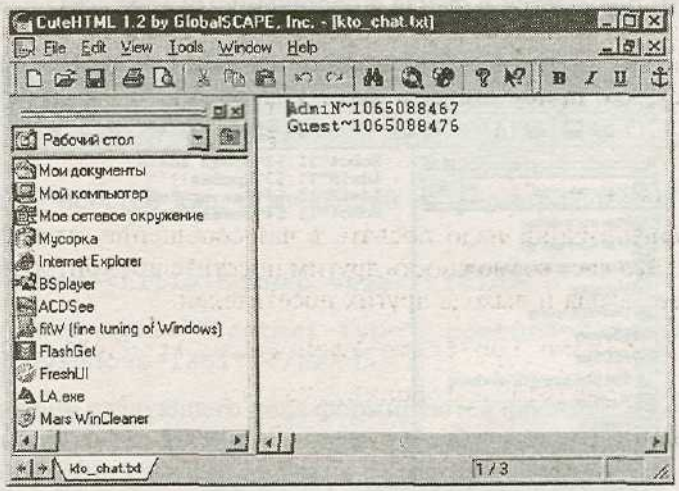


Рис. 25. Содержимое файла kto_chat.txt

Заканчиваем цикл обработки заходящего ника:

```
}
$i++;
endwhile;
```

Проверяем метку, чтобы выяснить, пришел новый ник или такой ник уже есть в чате.. Если она равна - 1 , значит, ник новый;

```
if {$met == -1) {
$sav = "$name_rob~$time~Вошел $person~\n";
$fp = fopen($file_logchat, "a+"};
if ($fp) { $fw = fwrite($fp, $sav); fclose($fp); }
$kto[] = "$person~$time_nast\n";
```

Не правда ли, знакомый нам блок? Робот сообщает в чат, что вошел такой-то ник (аналогично только что рассмотренному нами блоку с сообщением о выходе из чата). В последней строке мы добавляем новичка, только что вошедшего в чат, в массив файла с присутствующими никами.

Вот что получается в файле с сообщениями (рис. 26).

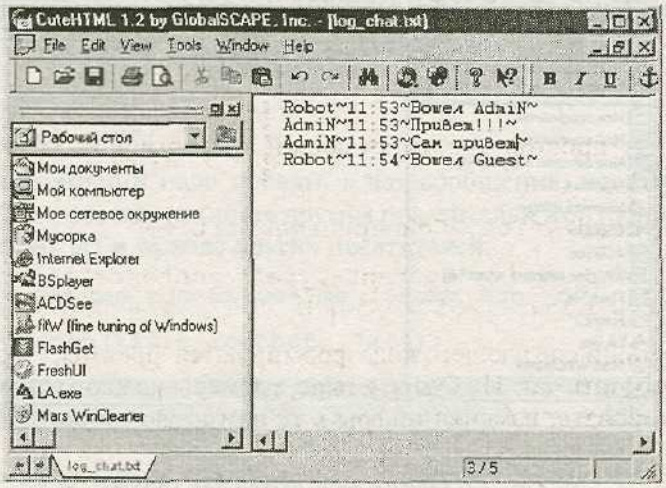


Рис. 26. Содержимое файла log_chat.txt

Иначе, если входящий ник уже есть в чате, просто обновляем его информацию в массиве, который в дальнейшем будет записан в файл со списком тех, кто присутствует в чате:

```
} else { $kto[$met] = "$person~$time_nast\n"; }
```

Обратите внимание, что запись идет в нужном нам формате: ник, тильда, текущее время, перевод строки. Это, как я уже говорил, формат хранения данных в этом файле.

Теперь преобразуем весь массив в одну переменную и вырежем двойные и тройные переводы строк, которые там могут случайно оказаться:

```
$sav = implode("\n", $kto);
$sav = str_replace ("\n\n","\n", $sav);
$sav = str_replace ("\n\n","\n", $sav);
```

Записываем эту строку с обновленными данными в файл, предварительно полностью обнулив его:

```
$fp = fopen($file_kto_in_chat, "w");
if ($fp) { $fw = fwrite($fp, $sav); fclose($fp); }
?>
```

Дальше — простой код, отвечающий за формирование HTML-страницы:

```
<html><headxtitle>Примерчата</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251"></head>
```

При помощи следующего кода формируются фреймы, в которых будет работать чат. Их будет четыре, размеры каждого вы можете отрегулировать, изменяя цифры в ту или иную сторону:

```
<frameset framespacing="0" rows="54,*,70">
<frame name="banner" scrolling="no" noresize target= "banner" src="banner.phtml">
<frameset cols="200,*">
<frame name="userlist" target="userlist" src="userlist.phtml" frameborder="0" scrolling="1">
<frame name="main_window" src="main_window.phtml" frameborder="0">
</frameset><frame name="msg" scrolling-"no" noresize target="msg" src="msg.phtml" frameborder="0">
</frameset>
```

Файл banner.phtml. В верхнем фрейме можно организовать вывод баннеров, сообщений от администрации сайта и т.д.

Например:

```
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

<style type=text/css>
A { text-decoration: none}
A { color: #FFFFFF}
A:hover {color: #EFEF4B; text-decoration:underline;}
A.mat {color: #FFFFFF}
A.mat:hover {color: #0066FF}
p { font: 8px Verdana }
</style></head>
<body topmargin=0 leftmargin=0 bgcolor=#546A8C>
```

Так мы подготовили заголовок и определили стили. Вы можете их поменять самостоятельно.

Теперь выводим в таблице по центру текст рекламы или список баннеров.

```
<table border="0" cellpadding="0" cellspacing="0"
width="90%" align="center"><tr><td>
<font color="white">
Текстрекламы</td></tr></table>
</body></html>
```

На этом с подготовкой окончено, переходим непосредственно к вводу, выводу сообщений и показу присутствующих в чате.

Файл main_window.phtml. Данная страница — особенная, она отличается от остальных тем, что периодически должна автоматически обновляться (у нас это будет происходить каждые пять секунд):

```
<? include {"tools.phtml"); ?>
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<meta http-equiv=Refresh content="5; URL=main_window.phtml">
```

Обновление нужно для того, чтобы посетители могли видеть новые сообщения, посланные в чат.

Уменьшение времени реагирования чата ведет к увеличению нагрузки на канал связи и на сервер, поэтому не стоит ставить этот параметр очень малым.

Теперь зададим стили (см. приложение 1), в соответствии с которыми будут отображаться сообщения:

```
<style type=text/css>
body {
scrollbar-face-color:#54 6A8C;
scrollbar-3dlight-color:#FFFFFF;
scrollbar-track-color: #FFFFFF;
scrollbar-arrow-color: #FFFFFF;
scrollbar-border-color:#000000 }
p { font: Verdana; font-size : 12; }
</style>
<body bgcolor=#E7E7E7>
```

Дальше нам надо прочитать файл с сообщениями и вывести на экран последние 30 сообщений:

```
<p>
<?
$user = file ($file_logchat);
$i = count($user) ;
```

```
$fi = $i - 30;
if ($fi < 0) { $fi = 0; }
while ($i >= $fi):
```

Считываем файл с сообщениями в массив и начинаем обрабатывать последние 30:

```
list ($name, $time_name, $msg, $komu) = split ("~", trim(str_replace ("\n","", $user [$i])));
```

Сообщения записаны в специальном компактном формате, который надо расшифровать, а заодно удалить символ перевода строки.

Дальше:

```
if ($name and $msg) {
if (!$komu or $komu == $person) {
if (!$komu) {
echo "<small>$time_name>
</small> <b>$name</b> $msg<br>";
 } else {
echo "<small>$time_name>
</small> <b>для $name от $komu:</b> $msg<br>";
 }
 }
 }
```

Вывод организован таким образом, что если кто-то захотел отправить сообщение только определенному нику, другие это сообщение не увидят. Вот такая простая реализация идеи привата.

```
$i-;
endwhile;
?><br></body></html>
```

Закрываем цикл и сам файл.

Файл msg.phtml. Этот файл — один из самых больших и сложных файлов чата в смысле программирования, но он практически не дает нагрузки на сервер, так как отрабатывает один раз, когда пользователь вводит свои данные в чат. Его основная функция — просто обеспечить ввод посылаемой строки в файл с сообщениями чата.

Как обычно, в начале вызываем служебный файл и описываем стили. Обращайте внимание, чтобы перед вызовом не было даже пробела, иначе будет ошибка работы с сессиями.

```
<? include ("tools.phtml"); ?>
<html><head><meta http-equiv="Content-Type" content="text/
html; charset=windows-1251">
<style type=text/css>
input.button {
border-style:solid;
border-width:1px;
border-width-color:#546A8C;
width:80px;
height:20px;
font-family: Verdana;
font-size: 10px;
color: #000000;
font-size: 10px;
background: #E7E7E7
}
p { font: 8px Verdana }
body { font: 12px Verdana }
</style>
```

Проверяем каждый присутствующий ник на истечение времени последней связи:

```
<?
$kto = file ($file_kto_in_chat);
$fi = count($kto); $met = -1; $i = 0;
 while ($i <= $fi):
 $str = trim(str_replace ("\n","", $kto [$i]));
if ($str) {
 list ($name_kto, $time_kto) = split ("~", $str);
 if ( $name_kto == $person ) { $met = $i; $i = 10000; }
 }
 $i++;
 endwhile; ?>
```

Если предел достигнут, чат удаляет такой ник. Такие проверки необходимо делать для того, чтобы своевременно удалять старые ники, владельцы которых уже давно ушли или связь с ними прервана.

Теперь сделаем форму для ввода текстового сообщения:

```
<base target="msg">
</head>
<body leftmargin="0" topmargin="8" bgcolor="#546A8C">
<? if ($met != -1) { ?>
<form method="POST">
    
<input maxlength="500" type="text" name="msg" value=""
size="20">
<input type="submit" value="сказать" class="button">
```

Получили такую форму (рис. 27).

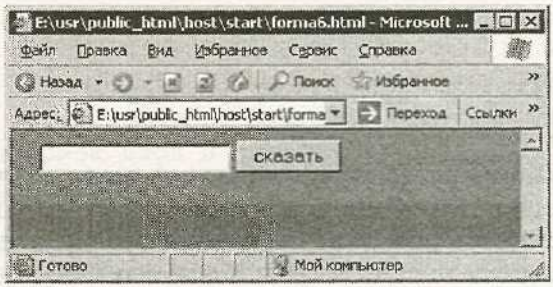


Рис. 27. Форма для ввода сообщения в чат

Мы задали ограничение на количество символов вводимого сообщения — 500. Такое ограничение «взломщики» могут легко обойти; как защититься от этого, читайте в главе «Работа со строками».

Если в чате больше двух собеседников, выводится меню для привата, которое позволит выбрать, кому именно будет послано сообщение в чате:

```
<? if ($fi > 2 ) { ?>
<select name=komu>
<option value="" selected>Кому</option>
<?

$i = 0; while ($1 < $fi) :
 $str = trim(str_replace ("\n","", $kto[$i]));
 if ($str != "") {
 list ($nik) = split ("~", $str);
 if ($nik != $person) {
echo "<option value=\"$nik\">$nik</option>";
 }
```

```
 }
$i++;
endwhile;
echo "</select>";
 }
```

Если поле «Кому» не трогать, сообщение смогут прочитать все присутствующие. Так можно реализовать простой приват. Однако алгоритм этого привата не оптимизирован, приватное сообщение приходится писать два раза в файл сообщений, а можно ввести еще одну позицию и проверять ее наличие.

Далее формируется строка в соответствии с установленными правилами: имя ника, время сообщения, само сообщение, кому сообщение:

```
 }
if ($met != -1) {
if {$msg and $person) {
 $sav = "$person~$time~$msg~$komu\n";
 $fp = fopen($file_logchat, "a+");
 if ($fp) { $fw = fwrite($fp, $sav); fclose($fp); }
```

Каждый параметр отделен от другого тильдой. В конце — обязательный перевод строки. Он служит разделителем и обеспечивает удобочитаемость базы сообщений.

Если выбрано сообщение для определенного ника (приватное сообщение), делаем еще одну аналогичную запись:

```
if ($komu) {
 $sav = "$komu~$time~$msg~$person\n";
 $fp = fopen($file_logchat, "a+");
 if ($fp) { $fw = fwrite($fp, $sav); fclose($fp); }
 }
```

Это нужно для того, чтобы сообщение смог увидеть не только тот, кому его послали, но и тот, кто послал.

Если посетитель не авторизован, появится сообщение об ошибке:

```
}
} else { echo "    Ошибка чтения данных — повторите вход в систему!"; }
?></form></body></html>
```

Все, с вводом данных покончено. Осталось вывести список тех, кто присутствует в чате в данный момент.

Напоминаю, код максимально упрощен, в нем отсутствуют многие необходимые вещи, например урезание вводимой строки в коде PHP. Попробуйте усовершенствовать его и добавить свои возможности.

Файл userlist.phtml. Данный файл наиболее сложный из всех в плане нагрузки на сервер. Он выполняется автоматически через определенный промежуток времени вне зависимости от действий пользователя. Нам придется проводить анализ файла со списком ников, присутствующих в чате, и редактировать его в соответствии с реальной ситуацией. Для начала, как всегда, подготовка:

```
<? include ("tools.phtml"}; ?>
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
```

Определяем промежуток, через который эта страница обновится:

```
<meta http-equiv=refresh content="15; url=userlist.phtml">
```

Цифра 15 — именно то значение, которое нам нужно. Это время в секундах, по истечении которого произойдет автообновление страницы. Скрипт выполнится очередной раз, внеся соответствующие записи в конфигурационные файлы чата, а другие посетители чата (точнее, их браузеры) уже будут знать, что есть такой человек, его оставляем в списке посетителей. Надо быть пре-

дельно осторожным, и подобрать это число экспериментальным путем, так как его уменьшение приводит к резкому увеличению загрузки сервера, а увеличение — к неадекватному отображению информации о посетителях.

Описываем стили для списка присутствующих:

```
<base target=userlist></head>
<style type=text/css>
body {
scrollbar-face-color:#546A8C;
scrollbar-3dlight-color:#FFFFFF;
scrollbar-track-color: #FFFFFF;
scrollbar-arrow-color: #FFFFFF;
scrollbar-border-color:#000000
 }
p { font: 10px Verdana }
td { font: 10px Verdana }
</style>
<body topmargin=0 leftmargin=0 bgcolor=#546A8C>
```

Для более удобного представления ников присутствующих в чате сделаем таблицу:

```
<form method="POST" target="_self">
<table border="0" width="100%" height="100%">
<tr><td width="8%"></td>
<td width="87%" bgcolor="#E7E7E7" valign="top">
<table border="0" width="100%"><tr>
<td width="100%" height="100%">
<big>В чате сейчас:</big><p>
```

Теперь код обработки списка ников:

```
<?
$kto = file ($file_kto_in_chat);
$fi = count($kto); $met = -1; $i = 0;
 while ($i <= $fi):
 $str = trim(str_replace ("\n","", $kto [$i]));
if <$str) {
 list ($name_kto, $time_kto) = split {"~", $str);
 if ( ($time_nast-$time_kto) > $chat_number)
 {
$kto [$i] = "";
$sav = "$name_rob~$time~Вошел $name_kto~\n";
$fp = fopen{$file_logchat, "a+") ;
if ($fp) { $fw = fwrite($fp, $sav); fclose($fp); }
 }
```

Опять выполнился блок обработки списка ников на наличие посетителей, уже давно покинувших чат. Все их учетные записи в процессе его работы будут уничтожены. Это основной блок проверки ников, и именно он дает такую нагрузку на сервер. Представьте, если эти скрипты будут работать одновременно у 20—30 человек. А если больше?

Дальше:

```
 if ( $name_kto == $person )
 {
$met = $i;
echo "<b>$name_kto</b><br>";
 } else {
echo "$name_kto<br>";
```

```
 }
 }
 $i++;
 endwhile;
```

Здесь выводятся оставшиеся ники в цикле перебора всех присутствующих.

Неавторизованным посетителям выводится сообщение об ошибке:

```
if ($met == -1)
{
echo "<p>Система остановлена – ошибка чтения данных!";

} else {
 $kto[$met] = "$person~$time_nast\n";
 }
```

Оно же будет показано, если время последней связи посетителя с чатом превысило установленный предел и ник был удален из списка присутствующих в чате.

Следующая часть кода:

```
$sav = implode("\n", $kto);
$sav = str_replace ("\n\n","\n", $sav);
$sav = str_replace ("\n\n","\n", $sav);
$fp = fopen($file_kto_in_chat, "w");
if ($fp) { $fw = fwrite($fp, $sav); fclose($fp); }
?>
```

В первой строке делаем слияние массива посетителей чата в одну строку, потом обрабатываем ее на наличие двух и трех идущих под-

ряд переводов строки, которые при обнаружении заменяем одинарным переводом строки. Потом открываем соединение с файлом и записываем туда объединенный массив из ников в чате.

И, на всякий случай, показываем ссылку, по которой можно обновить данные вручную, щелкнув мышкой:

```
<br><a href="userlist.phtml">Обновить</a>
</td></tr></table></td></tr>
</table></body></html>
```

Все, чат готов. Можете запускать и пользоваться!

Использование специального привата

Рассмотрим реализацию оригинального приватного общения в чате, при котором приватные сообщения выводятся в отдельное окошко, а не в общее окно с сообщениями всех пользователей.

На базе готового чата с возможностью регистрации и парольного входа создадим максимально защищенное пространство для переписки между двумя пользователями.

Ники в чате должны быть закреплены персонально, иначе теряется смысл привата.

Первое, что приходит на ум для решения поставленной задачи, — сделать несколько простых чатов и использовать их для приватов, разрешив вход по паролю. Это самый простой, но и самый неверный путь.

Если у вас всего несколько посетителей одновременно в чате, то такой подход еще может быть оправдан, но может случиться так, что все ваши приваты вдруг окажутся занятыми, а кто-то захочет еще. И увеличение количества готовых чатов —, не выход, так как есть способ лучше.

Не нужно множить чаты, достаточно сделать один, но работать он будет не с фиксированным файлом (в который заносятся сообщения, вводимые в чат), как происходит при работе обычного

чата, а с динамическим. Файлы будут создаваться по мере надобности и таким образом снимается ограничение на количество одновременно функционирующих приватов.

В первую очередь нужно подумать о том, что конкретно будет служить именем для файла сообщений в привате. Неплохой вариант — комбинация ника и пароля. Только не прямая, конечно, комбинация, а как-либо зашифрованная. Еще один вариант — генерация уникального имени в каталоге. Такую возможность предоставляют многие языки программирования (в том числе и PHP конечно же). Если нет такой возможности, можно генерировать случайное число и использовать его для имени файла. Однако все эти способы не дают абсолютной гарантии уникальности, а имя файла для привата обязательно должно быть уникальным.

Я предлагаю для генерации уникального имени использовать функцию времени, прошедшего с начала эпохи Unix (см. приложение 6).

Именно это количество секунд служит идеальным генератором случайных имен файлов для приватов в чате. Главное для привата — это уникальность имени. Сформировать такое имя при помощи времени Unix очень просто: достаточно считать показания секундомера Unix и объединить полученное значение со строкой, включающей в себя точку и расширение файла с сообщениями собеседника.

Первая строка этого файла уже занята под тег открытия кода и комментарий.

Эта же строка нам понадобится для авторизации. В нее нужно дописать ники двух пользователей, которым доступен именно этот файл с сообщениями. Каждый раз при авторизации нужно проверять, совпадают ли имена входящих в приват с записанными в этой строке, и на основании полученных данных пропускать посетителя в приват или нет. Рассмотрим эту технологию подробней.

Как приглашать в приват

Для приглашения в приват может использоваться или ссылка, или кнопка — это дело вкуса и возможностей. Нажав на такую ссылку, посетитель попадает на страницу, предлагающую выбрать, кого именно из присутствующих в чате в данный момент он желает пригласить в приват[1].

Если используется ссылка, данные авторизации человека, открывающего приват, нужно передать методом GET. А если кнопка — можно (скорее, даже нужно) использовать метод POST.

Можно при входе в приват еще раз попросить ввести пароль.

На этом этапе обязательно нужно проверить, есть ли такие пользователи среди зарегистрированных в чате, чтобы заблокировать возможность несанкционированного входа. Можно также предоставить входящим возможность немного настроить свой чат — выбрать цвет, скорость обновления и т.д.

Когда все, что нужно, выбрано (самое главное — кого пригласить в приватный чат), посетитель нажимает на кнопку входа, и запускается программа генерации привата.

Создание привата

Сначала, как и всегда, нужно проверить правильность пароля и ника. Если все нормально и такой пользователь зарегистрирован, переходим к тому, кого он пригласил. Этого посетителя тоже надо проверить, например, на присутствие в чате. Дальше необходимо сформировать данные для авторизации. Для этого считываем время Unix:

```
$time = time(void);
```

Определяем имя файла с сообщениями, используя любой путь к файлу:

```
$file = "любой путь".$time.".phtml";
```

[1] Конечно, чат должен поддерживать технологию отображения присутствующих, иначе придется предлагать выбор из всех зарегистрированных ников, что будет просто нехорошо с нашей стороны.

Формируем строку для записи в несуществующий пока файл, путь к которому мы только что задали:

```
$sav = "<? /* * $person * для * $kto * \n";
```

В переменных $person и $kto хранятся соответственно ники приглашающего и приглашенного в приват. Записываем строку $sav в файл специальной командой:

```
// пример записи переменной в файл
$file = "имя файла, в который записываем";
$sav = "то, что записываем";
$fp = fopen ($file, "w"); // открываем файл
if($fp)
{
$fw = fwrite ($fp, $sav); // если успешно, записываем в него
fclose($fp); // закрываем файл
}
else
{
/* тут можно вставить процедуру обработки ошибки записи
в файл */
}
```

Подобные команды устроены таким образом, что, если файл не существует, он создается. Однако создаваться он будет только в том случае, если у каталога, в котором он создается, установлены соответствующие права.

Как связаться с приглашенным

Для этого можно воспользоваться самым простым методом — вывести приглашение в основной чат. Оно должно быть оформлено в виде ссылки, чтобы приглашенный мог просто нажать на

нее, ввести в открывшейся странице пароль и сразу попасть в приват. Впрочем, иногда лучше вывести сообщение о приглашении в приват не всем сразу, а конкретному посетителю.

В ссылке приглашения должно передаваться имя файла привата, сгенерированное с помощью функции времени Unix. Передавать его в открытом виде нельзя и придется применить любой из обратимых методов шифрования.

Самый простой способ зашифровать — выполнить ряд обратимых арифметических действий. Ведь передавать нужно простое число, которое можно сложить с другим.

Вместе с зашифрованным именем файла надо передать идентификатор приглашенного, чтобы никто, кроме него, не мог попасть в приват. Я рекомендую воспользоваться тем методом, на основе которого реализована авторизация в чате. В принципе, можно передавать и сам ник, мы предусмотрели дополнительную защиту от чужого проникновения в приват, когда записали ники имеющих право войти в него в первую строку файла привата.

Приглашенному в приват тоже можно дать возможность настроить его для себя, потом проверить пароль и учетную запись и, если все нормально, разрешить вход в чат. При этом необходимо расшифровать имя файла привата. Его, как и ник входящего, надо передать в чат методом GET. ЭТО нужно для того, чтобы приват мог каждый раз отслеживать и идентифицировать эти данные.

Осталось упомянуть о том, что при работе привата надо каждый раз проверять соответствие ников вошедших с записанными в файле привата. Имя файла не должно появляться в ссылках ни на одном из этапов входа в приват. Пользуйтесь везде разными формулами для скрытия истинного имени файла привата. Старайтесь генерировать каждый раз новую формулу.

Приват следует сделать максимально простым и быстрым. Посетители используют приват не для каких-то дополнительных возможностей, а чтобы спокойно пообщаться в полной уверенности, что их не «подслушают».

Хорошим тоном считается наличие кнопки, очищающей разговоры в привате. Также желательно чтобы после выхода файл удалялся с сервера.

Проявите свою фантазию и попробуйте усовершенствовать приведенную схему.

Интернет-магазин

Технология создания

Сделать несложный интернет-магазин самому, как ни странно, очень просто. Несколько сложнее придумать, как привязать собственный магазин (здесь и далее подразумевается, конечно, интернет-магазин) к конкретным товарам. Например, есть набор канцтоваров в количестве 1000 наименований. Прайс с информацией о товарах обновляется практически ежедневно, и не просто обновляется, а кардинально. Причем изменяются не только цены, что еще полбеды, но и виды товаров, их описания, сроки гарантии и поставок и т.д. Попробуйте вносить все эти данные в свой магазин ежедневно, и вы сразу проникнитесь уважением к тем людям, которые делают такую работу. Именно обновление данных в магазине и занимает основную часть собственно самой программы. Это то, что необходимо продумать в первую очередь.

Прежде чем вы начнете делать свой магазин, необходимо досконально изучить прайс товаров и способы его обновления в оригинале. Допустим, есть фирма, в ассортименте которой компьютерные товары. Их достаточное количество, чтобы не заниматься обновлением вручную, а придумать систему для синхронизации данных прайса и магазина.

Пусть прайс у нас будет набран в программе Excel (как правило, это не так, многие фирмы используют в работе специальные программы, например «1С: Бухгалтерия», но это не важно, так как данные из таких пакетов всегда можно импортировать в Excel). Количество разделов в прайсе и товаров в них не фиксировано, а сам прайс представляет собой набор строк и колонок. В каждой

строке — товар или наименование раздела прайса (например, процессоры, память, мыши и т.д.). Количество колонок не имеет значения, обычно присутствуют наименование, описание и цена товара. Впрочем, могут быть и другие варианты.

Что и как можно использовать, имея такие данные? Конечно, существуют специальные модули для выборки данных из файла формата Excel (различные для разных языков программирования), но такие модули, как правило, не являются бесплатными, а значит, неприемлемы для нас. Кроме того, существует проблема настройки этих модулей. Иногда, чтобы это сделать, нужно обладать такой квалификацией, что проще самому все написать с нуля :-) Впрочем, эти модули и не нужны на самом деле. Что представляет собой наш прайс? Правильно, набор строк и колонок, причем строго организованных. А значит, из таких строк и колонок можно построить текстовый файл с точно такой же структурой. Только вместо визуального разделения строк и колонок (как в Excel) надо использовать какие-либо символы. Конечно, в этом случае внешний вид сильно изменится, так как длина строк всегда разная, но это и не важно, ведь структура останется прежней. Это будут все те же строки и колонки, только уже в текстовом формате, который очень легко прочитать и обработать при помощи практически любого языка программирования.

Excel умеет сохранять свои файлы в разных форматах. Нас будет интересовать сейчас формат «текстовый файл с разделителями». Тип разделителя не важен, в их качестве можно использовать даже запятые или пробелы. Мы же остановимся на табуляции.

Вот что у нас получилось: файл *.txt, в котором строки разделены переводом строки (это простой Enter), колонки разделены символами табуляции. Структура прайса осталась прежней, так как об этом позаботился мастер сохранения Excel.

Вот мы и нашли способ не заниматься рутинной работой по ежедневному обновлению прайса в нашем интернет-магазине, а максимально автоматизировать этот процесс. Перед тем как начинать работать с полученным файлом, можно подумать, как его немного сжать или обработать. Это бывает необходимо для

того, чтобы удалить ненужные колонки, повторяющиеся пробелы, временно отсутствующие в наличии товары и т.д.

Для этого надо считать файл в массив, обработать каждую строку на предмет наличия ненужной информации, записать полученный результат опять в массив на то же место и, наконец, весь массив записать в файл. Однако исходите из конкретной необходимости таких операций, вполне возможно достаточно просто удалить повторяющиеся пробелы.

Когда все готово, пора приступать к выборке данных из нашего текстового файла с прайсом. Сначала нужно считать файл в массив:

```
$file = "price.txt";
$tovar = file ($file);
```

Дальше надо обработать массив $tovar [] в цикле с выводом результатов работы на экран:

```
$i = 0; // счетчик начинается с нуля
while ($i < count ($tovar)) : // открыли цикл до последнего
                              // элемента в массиве
echo $tovar[$i]; // выводим строку на экран
echo "<br>"; // не забываем про перевод строк
$i++; // увеличиваем счетчик на единицу
endwhile; // заканчиваем цикл
```

Вот самый простой вариант вывода прайса. Он, правда, обладает существенным недостатком — весь прайс выводится за один раз, причем все колонки в одной строке. А это неприемлемо по эстетическим причинам.

Правильнее предоставить клиенту возможность выбора раздела прайса.

Сначала нужно отделить разделы прайса от информации о товаре. Это можно сделать, если обратить внимание на то, в какой из колонок находится название раздела. Как правило, в той же, что

и название товара. Но в строке с названием раздела нет информации о цене товара, а это может служить хорошим отличительным признаком для программы. Достаточно каждый раз проверять наличие в выбранной строке колонки цен. Если колонка пустая, значит, в строке — название раздела, и его можно вывести на экран, если нет — это товар, такую строку просто пропускаем, если надо выводить только названия разделов. И наоборот — для вывода только товаров. Программа просматривает в цикле строки файла. Если в колонке с ценой не пусто — это товар, и его надо вывести на экран, предварительно отформатировав информацию. Если же пусто — это начало следующего раздела прайса, и обработку надо прекратить, так как цель достигнута: весь выбранный посетителем раздел выведен на экран.

Нам надо продумать возможность просмотра конкретного раздела прайса. Можно, например, выводимые наименования разделов прайса сделать ссылками, в качестве параметра которых указать информацию о том, с какой строки начинается данный раздел. Немного модернизируем наш алгоритм вывода разделов прайса таким образом, чтобы после нажатия на ссылку-наименование раздела запускалась другая программа, которая начинала бы просмотр файла с прайсом не с начала, а с того места, которое передано по ссылке в качестве параметра, и уже выводила бы не разделы прайса, а цену, наименование, описание товаров и т.д.

Обязательно необходимо продумать удобную навигацию: ваш посетитель не должен заблудиться. На каждой странице должна быть ссылка на начало прайса и на список его разделов. Неплохо предоставить возможность перехода на следующий и на предыдущий разделы прайса прямо из списка товаров.

Теперь о том, как разделить информацию в выбранной строке по колонкам. Отведем каждой колонке свою переменную. Если колонок в прайсе пять, то и переменных надо зарезервировать пять. Чтобы не запутаться, возьмите похожие имена: `$n_1`, `$n_2`, `$n_3`, `$n_4`, `$n_5`. Теперь, если нужная строка находится в массиве `$tovar[$i]`, достаточно применить такую конструкцию:

```
list ($n_1, $n_2, $n_3, $n_4, $n_5) = split <"\t", $tovar[$i]) ;
```

И если колонки были разделены символом табуляции, они распределятся каждая в свою переменную. Если вы использовали другой разделитель колонок, вам надо указать его вместо "\t".

Теперь легко проверить наличие цены, а также отсортировать и отформатировать данные прайса нужным образом. Однако этого еще не достаточно для полноценного интернет-магазина, так как он подразумевает возможность заказа выбранных товаров.

Мы подошли к очень интересной теме — «виртуальная корзина». Если вы пробовали заказать товар на различных сайтах, то могли заметить, что работа корзины реализована по-разному. Что нужно обычному посетителю от интернет-магазина? Правильно, купить товар. А что ему для этого требуется? Удобная навигация, легкий и быстрый доступ к товарам, возможность изменить количество товара одного наименования в корзине, удалить товар из корзины, получить информацию о товаре (описание с фотографией и цена). Ограничимся этим необходимым минимумом, так как увеличение доступных функций отрицательно сказывается на быстродействии и существенно усложняет программу.

Итак, как можно сохранять информацию о товаре в корзине покупателя? Можно попросить покупателя зарегистрироваться в системе, но это может сократить количество посетителей магазина.

Подумайте: как относитесь к регистрации вы сами, часто ли хочется оставлять свои данные на сомнительных сайтах? Вот именно, очень не хочется. А значит, надо иметь очень веские причины, чтобы ввести регистрацию. Такими причинами могут быть: наличие постоянного круга покупателей, товар, пользующийся повышенным спросом, и т.д.

Мы же не будем останавливаться на регистрации, так как она в принципе ничего не меняет. Для сохранения информации о посетителе, в частности информации в корзине пользователя, будем использовать сессии PHP.

Нам нужно решить, что именно хранить в cookie. Можно — всю информацию о товаре, а можно только цифровой идентификатор и количество. Делайте, как вам проще.

Цифровой идентификатор занимает не очень много места в памяти, однако учтите, что с ним больше хлопот в смысле программирования. Придется писать код сопоставления идентификатора и выборки информации из файла прайса.

Обратите внимание еще на один момент. Если прайс очень часто меняется, информация в корзине покупателя может быстро устареть и не соответствовать действительности. Значит, стоит подумать о контроле даты. Дату, когда посетитель положил товар в корзину, тоже придется хранить в cookie, так как больше негде. Если эта дата сильно отличается от текущей, стоит предупредить покупателя о несоответствии, иначе могут произойти досадные казусы. Неприятно, когда заказ делается по цене недельной давности. Она ведь может не только упасть, но и увеличиться! Потом придется объясняться. Лучше позаботиться о таких вещах заранее. Хотя, если прайс более-менее постоянный, беспокоиться не стоит.

Цифровым идентификатором очень удобно выбрать номер позиции в прайсе. Этот номер — уникальный для каждого товара. По номеру позиции легко можно извлечь информацию о товаре из текстового файла с прайсом. Достаточно считать весь файл в массив:

```
$file = "price.txt";
$as = file ($file);
```

И обратиться к строке массива с нужным индексом:

```
$info = $as[$id];
```

$id — идентификатор, который нам нужен. Правда, применяя данный метод, нам не избежать контроля дат, но это не так уж и плохо.

Вместе с идентификатором товара надо запоминать и количество товара. По умолчанию при первом заказе вводится число 1. Дальше желательно предоставить возможность (в ограниченных

рамках, конечно) изменять это число. Для пересчета суммы понадобится кнопка. Если она будет в HTML-коде указана после формы, сработает и нажатие на клавишу Enter.

Кроме того, на странице работы с корзиной обязательно надо предусмотреть возможность окончательно оформить заказ или заказать что-то дополнительно. Это можно сделать и кнопками, и ссылками — что вам больше нравится.

Конечно, необходимо продумать, как именно выводить информацию о полной стоимости заказа. Не стоит забывать о скидках и подарках, если они, конечно, есть в вашем интернет-магазине. Хорошо впишутся сюда различные бонусы и информация о доставке. Если она бесплатная — напишите об этом. Лишним не будет.

Наконец, оформление формы заказа. Стандартные поля — имя, адрес, телефон, почтовый ящик и т.д. Чем меньше, тем лучше. Обязательно должно быть поле примечания — оставьте клиенту возможность высказать свои пожелания. Здесь же должна присутствовать вся информация о заказанном товаре, его стоимости и самая главная кнопка — «Заказать».

Если нет регистрации, можно введенные данные запомнить (как вы понимаете, при помощи cookie) и при следующем заказе просто подставить. Клиент будет благодарен, поверьте.

После того как заказ оформлен и подтвержден, обязательно скажите спасибо клиенту за покупку и выведите еще раз информацию о заказе на экран. При этом не забудьте очистить корзину — она уже выполнила свою функцию. Вот и все. По крайней мере, для покупателя.

Теперь перейдем к внутренним процессам, происходящим при заказе. Для хранения информации обо всех заказах нужно создать служебный файл. В него будут заноситься все данные о заказах. Причем записывать уже надо не идентификаторы, а полную информацию — название, цену и т.д. (надеюсь, места у вас на хосте много). Так удобнее потом просматривать и обрабатывать. Для этого просто дописываем специальным образом сформированную строку в конец файла. Прочитать такой файл мож-

но и в текстовом редакторе, и при помощи специальной служебной программы.

Очень актуальной является своевременность обработки заказов. Я вам подскажу очень хороший метод, позволяющий практически мгновенно узнавать о заказе. И это без необходимости постоянно находиться в интернете и даже выходить в него раз в день. Нам понадобится простой пейджер. Практически все операторы предоставляют возможность отправить электронное сообщение на свой пейджер из интернета. Адрес для отправки выглядит приблизительно так: номер_пейджера@адрес_оператора.

Если на этот адрес отправить письмо, оно придет на ваш пейджер.

Неплохим вариантом может быть SMS на сотовый телефон, но надо уточнять у оператора, поддерживает ли его сеть такую возможность.

Как видите, на заказ можно отреагировать практически сразу, как только он поступит. Это повышает доверие клиентов к магазину и положительно сказывается на репутации.

Сервисы интернет-магазина

Создавая свой проект в Сети, и особенно это имеет отношение к интернет-магазину, не стоит отказываться от дополнительных возможностей, которые открываются благодаря использованию динамических сайтов. Посетителю всегда приятно ощущать постоянную работу и движение на сайте, а не созерцать «новости» и «новинки» месячной давности. Вы можете даже наработать материал немного вперед и потом по мере наступления сроков или просто по очереди выдавать его посетителям.

Новинки в виде графических баннеров

Одна из наиболее часто используемых в этой области технологий — новинки, оформленные в виде баннеров. Это можно сделать двумя способами.

Первый заключается в том, что изображения формируются на лету, при использовании какой-то готовой заготовки для фона и нескольких картинок для основной темы баннера. Кроме картинки, может присутствовать небольшой рекламный текст и цена. Поместить больше информации там, скорее всего, не получится. Зато ничто нам не мешает поставить на такой баннер ссылку, и в этом случае посетитель, если пожелает, без труда получит более полную информацию о товаре или услуге.

Второй путь — делать изображения уже готовыми, например, в графическом редакторе, и выводить их на экран по очереди, чередуя в зависимости от того, что посетитель просмотрел.

Эти два подхода не принципиально различны, можно использовать или тот, или другой в зависимости от поставленной задачи, а также от предпочтений и возможностей программиста. Первый подход более сложный в разработке, зато потом требует меньше времени на изготовление самих баннеров. Во втором случае наоборот — несложная программа, потом постоянная работа над новыми баннерами. Выбирайте, что вам удобнее.

В PHP для работы с изображениями предусмотрена специальная библиотека GD, которая должна быть подключена к интерпретатору. Подключение сводится к распаковке библиотеки GD в каталог для расширений PHP и включению ее поддержки в файле настроек php.ini, расположенном в каталоге Windows (см. гл. «Установка PHP»).

ВНИМАНИЕ

Обязательно выясните, обеспечивает ли ваш хостинг-провайдер поддержку этой библиотеки.

Итак, перед нами стоит задача вывода имеющихся изображений по очереди на экран посетителю. Допустим, что готовые баннеры лежат в каталоге banner.

Первое, что нам надо сделать, — просмотреть каталог, где будут размещаться баннеры для показа. Сделать это просто, так как PHP предоставляет массу возможностей по работе с файловой системой сервера. Вот пример:

```
$dir = opendir ("banner/");
while($f = readdir($dir))
  {
if (!stristr($f,"..") and $f != ".")
{ $tt[] = $f; }
 } }
```

В этом отрезке кода мы просматриваем выбранную директорию и все имена файлов, которые там имеются, заносим в массив $tt. Дальше можно работать уже не с директорией, так как это медленнее, а непосредственно с массивом, который будет содержать точную копию содержимого диска сервера (выбранной директории).

Решение, какой блок баннеров выводить, мы будем принимать на основании показаний счетчика. Счетчик можно хранить на компьютере клиента и показывать каждому посетителю следующий за просмотренным блок баннеров или вести общий для всех учет просмотренных блоков. В первом случае это проще, достаточно установить cookie для посетителя. Второй случай давайте рассмотрим подробнее:

```
$ttl = file ( "counter.php" );
$str = trim(str_replace ("\n","",$ttl[0]));
$str++;
if ($str >= count($tt) or $str < 0} { $str = 0; }
```

Считываем файл с показаниями счетчика в массив (или в строку), получаем из него первую строку, предварительно обрезав в ней пробелы и переводы строк. Потом увеличиваем показания счетчика и проверяем, не вышло ли его значение за установленный предел. В нашем случае допустимые значения лежат в пределах от ноля до размера массива, в котором хранятся имена файлов в директории с баннерами. И, если вышло, сбросить счетчик в ноль.

Показания счетчика необходимо снова записать в файл:

```
$fp = fopen("counter.php", "w");
if ($fp) { $fw = fwrite($fp, $str); fclose ($fp); }
```

Обратите внимание на имя файла, в котором хранится показание счетчика. Точнее, на его расширение. Оно, в принципе, может быть любым. Мы выбрали php, чтобы никто не мог получить к нему доступ. И кстати, не забудьте установить этому файлу атрибуты, разрешающие запись в него, иначе ничего работать не будет.

Начинаем работать над массивом имен баннеров:

```
$str1 = trim{str_replace (".","~", $tt[$str]));

@list ($id_price, $id_ras) = split("~", $str1);
```

Первая строка удаляет пробелы и переводит символы точки, являющейся разделителем между именем файла и его расширением, в тильду. Дальше раскладываем составляющую имени на собственно само имя и его расширение.

Если расширение очередного файла в каталоге подлежит обработке и выводу, то выводим баннер, если нет — ничего не делаем, пропускаем цикл:

```
if ($id_ras == "jpg" or $id_ras == "jpeg") {

echo "<img border=0 src=banner/$tt[$str]>"; } else {
```

В случае пропуска цикла требуется ввести еще одну переменную, что немного усложнит ситуацию:

```
@$siluro++; }
```

Дело в том, что некоторые баннеры, возможно, не понадобится выводить на экран, например, в связи с тем, что товара, который рекламирует баннер, нет в наличии. Учтем это в коде. Все, что нам надо, это еще несколько переменных. В самом начале такой части кода поместим следующие строки:

```
$temp_siluro = @$siluro;

if {@!$siluro} {
```

После блока чтения списка файлов надо поставить закрывающую скобку, так как этот блок должен выполняться только один раз — тогда, когда переменная `$siluro` равна нулю.

А в самом конце:

```
if (@$siluro != $temp_siluro and $siluro < 50) { include("mini.phtml"); }
```

Вы, наверное, уже догадались, что сам скрипт целиком должен располагаться в файле с именем mini.phtml, и вызываться в нужном месте командой include ("mini.phtml"); Получается интересная ситуация. Скрипт обрабатывается, и если по какой-то причине вдруг оказывается, что баннер не выведен на экран, то программа вызывает сама себя. Это позволяет избежать сбоев.

Текстовые баннеры

Иногда не требуются графические баннеры, достаточно текстовых, представляющих собой ссылку на статью, описание товара и т.д. В этом случае можно пользоваться более простым алгоритмом.

В файле chitatxt.php мы будем хранить то, что надо вывести на экран. Информация будет заноситься туда следующим образом: дата, адрес ссылки, текст для вывода на экран. Разделяться все параметры, как обычно, будут тильдой.

Итак, подготавливаем нужные нам переменные, считываем файл с текстовой базой в массив и определяем его длину:

```
$nomer = 1;
$file = "chitatxt.php";
$ii = 0;
$tt = file ( $file );
$qw = count($tt);
```

Открываем цикл, в котором и будем выводить наши сообщения:

```
while ($ii < 4):
```

Всего будет выведено четыре сообщения; если вам нужно другое количество, измените число в этом операторе. Если надо вывести только одно текстовое сообщение, можно не пользоваться циклом.

Ничего сложного, это мы уже проходили — раскладываем параметры строки на составляющие для последующей обработки:

```
$nomer++;
@$str = trim($tt[$nomer]);
list ($data, $add, $txt) = split ("~", $str);
```

Выводим ссылку в нужном нам виде:

```
echo "<b> $data</b> <a href=\"$add\"><br>$txt </a>";
```

Закрываем цикл и саму программу:

```
$ii++;
endwhile;
```

Все. Правда, мы не рассмотрели в этом примере сортировку по дате и вывод по счетчику, но это попробуйте сделать самостоятельно, в качестве домашнего задания.

Просмотр всех новинок

А как быть, если посетитель захочет посмотреть все приготовленные вами новинки? Он, конечно, может обновлять страницу столько раз, сколько потребуется до тех пор, пока не просмотрит все баннеры. Однако будет лучше, если мы создадим ссылку «Все новинки», указывающую на файл, в котором будет содержаться такой код:

```
<?
$dir = opendir("news/");
while ($f = readdir($dir))
 {
if (stristr($f,".html"))
{ $li[] = $f; }
 }
```

Знакомый для нас код, в нем мы просматриваем директорию, в которой хранятся новости и новинки, и помещаем все в массив.

```
$i = 0; while ($i < count($li)) :

include {"news/$li[$i]"); echo "<br><hr noshade size=1>";

$i++;

endwhile;

?>
```

Так выводим в цикле массив новинок, разделяя переводом строки и горизонтальной линией. Таким образом, наш магазин становится более удобным для посетителей.

Формирование комплексного заказа

Если в вашем интернет-магазине есть сводный прайс с несколькими позициями, можно сделать парочку интересных сервисов. Идеи, которые я сейчас опишу, можно использовать для любых товаров. Я же рассматриваю компьютеры, так как для них подобные идеи реализуются чаще всего.

Прайс должен быть выполнен в Excel и оформлен примерно следующим образом: в первой строке — наименование раздела, во второй — цена продукции, ее название и краткое описание. В третьей строке — опять наименование раздела, потом строка с товаром и т.д. Получится простой прайс, только, конечно, товаров в каждом разделе может быть сколько угодно (рис. 28).

Прайс из Excel нужно экспортировать в текстовый формат, например, с разделителями колонок — табуляциями (рис. 29).

После этого у вас получится соответствующий текстовый файл, колонки в котором будут разделены символами табуляции, а строки — символом абзаца (рис. 30).

Такой файл несложно обработать при помощи стандартных средств PHP.

29		Карманные брелоки-накопители	
30	21	32Mb Mini Easy Disk Removable USB flash memory	Брелок - накопитель, ультракомпактный
31	22	32MB Transcend JETFlash Drive USB	Мобильный накопитель данных в виде брелка!!!
32	32	64MB Seitec USB Bar - Removable flash memory	Мобильный накопитель данных в виде брелка!!!
33	36	64Mb PQI USB1.1 TravellingDisk	Носимый накопитель
34	42	128MB USB 2.0 CANYON USB flash memory	Брелок - накопитель
35	43	128MB Mobile Disk TwinMOS Removable USB flash memory	Брелок - накопитель
36	44	128MB Canyon USB 2.0 Flash Rubber Drives /для активных пользо	Резиновый материал корпуса, ударостойкий, пы
37	44	128MB Agenie A-Data USB 2.0!! / New!!! Faster!!	Брелок - накопитель, протокол USB2.0!
38	48	128MB USB 1.1 PQI TravellingDisk	Мобильный накопитель данных в виде брелка!!!
39	48	128MB USB 2.0 Mobile Disk TwinMOS Removable USB flash memory	Брелок - накопитель
40	65	256MB USB2.0 CANYON Flash Drive	Мобильный накопитель данных в виде брелка!!!
41	67	256MB Mobile Disk TwinMOS Removable USB flash memory	Накопитель данных в виде брелка!!!
42	68	256MB Canyon USB 2.0 Flash Rubber Drives /для активных пользов	Резиновый материал корпуса, ударостойкий, пы
43		Плееры CD, DVD, MP3 players	
44	53	Mustek MVR-100 - Цифровой диктофон + MP3 player	USB, Built-in LCD Display, MP3 Player, Microphone
45	82	Flash MP3 Player Creative Flash drive MuVo 64Mb	2 в 1- Брелок - накопитель + MP3 player, размеры

Рис. 28. Вид прайса в Excel

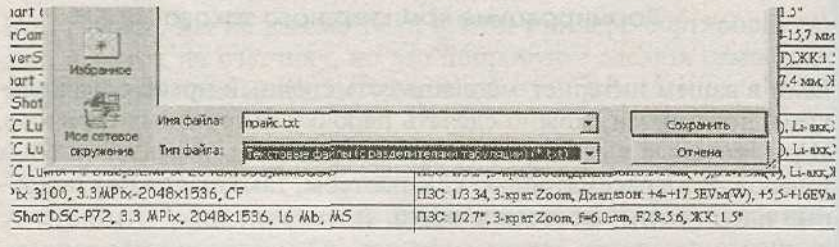


Рис. 29. Экспортирование прайса из Excel

```
27 498 CANON Digital IXUS V3, 3.2 Mpix-2048x1536, CF   "ПЗС 1/2.7"", 
28 612 PANASONIC Lumix LC40 Black, 4.1MPix-2240x1680, .MMC&SD  "ПЗС 1/
29     Карманные брелоки-накопители
30 21  32Mb  Mini Easy Disk Removable USB flash memory "Брелок - накоп
31 22  32MB  Transcend JETFlash Drive  USB Мобильный накопитель данных
32 32  64MB Seitec USB Bar - Removable flash memory    Мобильный накоп
33 36  64Mb PQI USB1.1 TravellingDisk  Носимый накопитель
34 42  128MB  USB 2.0  CANYON   USB flash memory  Брелок - накопитель
35 43  128MB  Mobile Disk TwinMOS Removable USB flash memory  Брелок -
36 44  128MB  Canyon  USB 2.0 Flash Rubber Drives /для активных пользов
37 44  128MB  Agenie A-Data USB 2.0!!  / New!!!/ Faster!!  "Брелок - на
38 48  128MB  USB 1.1  PQI  TravellingDisk Мобильный накопитель данных
39 48  128MB  USB 2.0  Mobile Disk TwinMOS Removable USB flash memory
40 65  256MB  USB2.0 CANYON Flash Drive    Мобильный накопитель данных
41 67  256MB  Mobile Disk TwinMOS Removable USB flash memory   Накопите
42 68  256MB  Canyon USB 2.0 Flash Rubber Drives /для активных пользова
43     Плееры CD, DVD, MP3 players
44 53  Mustek MVR-100 - Цифровой диктофон + MP3 player USB, Built-in LC
45 82  Flash MP3 Player Creative Flash drive MuVo 64Mb 2 в 1- Брелок -
```

Рис. 30. Вид экспортированного прайса

Итак, нам нужен список разделов, при щелчке на котором будет появляться меню с полным списком товаров из данного раздела. И главное — выбранный таким образом товар должен быть запомнен и потом подсчитан, а при необходимости добавлен в виртуальную корзину.

Начинаем работать над кодом:

Создаем форму, в которой будут происходить все дальнейшие действия:

```
<form method="POST"
name="pop" action="cena.phtml"
title="Подсчитать стоимость выбранных комплектующих">
```

При нажатии на кнопку SUBMIT форма направит данные, полученные от посетителя, обработчику с именем cena.phtml. Его мы рассмотрим в самом конце. А теперь — служебный файл read.phtml:

```
<?
$file=$DOCUMENT_ROOT."/$path/price.txt";
$as = file ( $file );
```

Загружаем в память (точнее — в массив с именем $as) полученный нами файл price.txt, который предварительно нужно закачать на сервер. Обратите внимание, на строку $file = $DOCUMENT_ROOT."/$path/price.txt"; Ее наличие приводит к тому, что где бы вы в дальнейшем ни запустили этот скрипт, он найдет файл с прайсом, и все благодаря переменной окружения $DOCUMENT_ROOTи,конечно,переменной$path,вкоторойзаранее нужно указать относительный путь к файлу с прайсом. Хранить этот файл непосредственно в корневом каталоге не рекомендую. Лучше заведите отдельную директорию, имя которой и укажите в переменной $path. Слеши ставить не надо, они уже заданы.

Следующий код позволит нам не выводить сообщения об ошибках, если скрипт работает в интернете, а не у вас на домашнем компьютере:

```
$ip=getenv("REMOTE_ADDR");
if ($ip != "127.0.0.1") { error_reporting(0); }
```

Конечно, IP-адрес должен быть реальным, т.е. тем, который используется у вас, иначе и на домашнем компьютере вы не получите ошибок, и разобраться, почему все не работает, будет не так просто.

Следующим кодом мы обрабатываем загруженный в массив файл таким образом, чтобы избавиться от ненужной информации и подготовить ее к дальнейшей удобной для нас обработке:

```
$i = 1;
while ($i < count($as)):

$str = trim(str_replace ("\n","", $as[$i]));
$str = str_replace ("\t","~", $str);
$str = str_replace ("\"","", $str);
list ($n_1, $n_2, $n_3) = split ("~", $str);
$n_1 = str_replace <",",".", $n_1);
$n_2 = str_replace (",",".", $n_2);

$as[$i] = "$n_1~$n_2~$n_3~$n_4~$n_5";

$as[$i] = str_replace (",","," , $as[$i]);
$as[$i] = str_replace (" "," ", $as[$i]);

$i++;
endwhile; ?>
```

Тут удаляются переводы строк, так как они нам больше не нужны, символы табуляции заменяются тильдой, удаляются лишние кавычки, если они есть, запятые перераспределяются для того,

чтобы можно было считать в PHP (запятые в PHP и в Excel — разные символы, и их надо преобразовать для PHP). Все, файл read.phtml закрываем и сделаем вторую подпрограмму, которая может не раз понадобиться нам в дальнейшей работе. Назовем файл с ней minor.phtml. Вот его содержимое:

```
<?
$str = str_replace ("\n","", $as[$q2]};
@list ($n_1, $n_2, $n_3) = split ("~", $str);
?>
```

Переменной $q2 присваивается искомое значение, а возвращаются разделенные на составляющие цена, название и описание товара (соответственно в переменных $n_1, $n_2 и $n_3). Дальше, возвращаемся к нашему первому файлу с формой.

Подготавливаем таблицу для того, чтобы наш список смотрелся красиво:

```
<? include ("read.phtml"); ?>
<table><tr><td>Цена</td>
<td>Название</td>
<td>Описание</tdx/tr>
<tr><td>
<?
$q2 = 0;
include ("minor.phtml");
echo $n_1;
?>: </td><td>
<select name="ur0" size="1">
<option value=all selected>
Позиция не выбрана</option>
<?
```

```
$ii =1; $i = 1; $s = count($as);
while ($i < $s) :
$q2 = $i; include ("minor.phtml");
if ($n_2 = "" and Si) {
```

Этот участок кода выполняется, если получены сведения, что программа обнаружила заголовок. Определяется это по пустой переменной с названием, так как название заголовка присваивается переменной с ценой. Это дает нам возможность выполнять разные действия в зависимости от ситуации.

Следующий код — для вывода заголовка:

```
echo "</select><br></td></tr><tr><td>$n_l:</td><td>
<select name=ur".$ii." size="l">
<option value=all selected>
Позиция не выбрана</option>";
$ii++; }
```

Для вывода цены товара, его наименования и описания:

```
else {
if ($n_l) {
echo "<option value=\"$i\">".$n_1.":".$n_3. "</option>";
} else { echo "</td><td></td></tr>"; }
}
```

Здесь все просто: формируются списки стандартными средствами HTML. Вы можете добавить к ним соответствующие стили, но не изменяйте PHP-код.

Окончание кода:

```
$i++;
endwhile;
?></select></td></tr></table>
```

```
<br><input type="hidden" name="nom" value="<? echo
$ii-1; ?>">
 <input TYPE="submit" VALUE="Подсчитать стои-
мость"
NAME="B3"></form>
```

Теперь осталось рассмотреть, что и как делает обработчик. Если вы помните, он у нас прописан в файле с именем cena.phtml.

```
<h3>Итоговая стоимость выбранных комплектующих</h3>
<? include ("read.phtml");
```

Уже знакомый нам файл read.phtml нашел применение и тут.

```
$i = 0;
$n_11 = 0;
while ($i <= $nom):
$nu = "ur".$i;
if ($$nu != "all") {
$q2 = $$nu; include ("minor.phtml");

$n_ll = $n_1+$n_ll;

$i++;
endwhile; ?>

<font color=red>".$n_ll."</font>
<br><a href=servis.phtml>
Повторить выбор</a>
```

В этом коде происходит простой подсчет цен, и есть только одна особенность, о которой стоит знать, — переменная в переменной.

Это значит, что если переменной $nu присвоить значение ur5, то значение $nu будет равно переменной $ur5. Например:

```
$ur5 = 10;

$nu = "ur5";

echo $nu;
```

Результат на экране — число 10.

Вот и все хитрости. Если вы не смогли понять, откуда взялась переменная $nom, могу подсказать: пришла из предыдущего файла с формой. В принципе, ее можно было найти и здесь, но проще сформировать именно там и передать сюда. Попробуйте модифицировать этот код так, как нужно именно вам, и разберитесь с работой форм, это избавит вас в дальнейшем от головной боли по поводу их работы.

Архив рассылок

Если на вашем сайте посетители могут подписаться на рассылку, совсем не лишним будет организовать архив рассылок. Конечно, это не нужно, например, для рассылки прайса фирмы, а вот для новостей или новинок будет очень полезным. Этим мы одновременно выполняем два действия: во-первых, позволяем посетителям просматривать неполученные ранее, просто пропущенные или удаленные рассылки без обращения к администратору. Во-вторых, гарантируем наращивание информации на сайте для поисковых систем. Они в последнее время стали интеллектуальными и индексируют только полезную информацию, которой является, например, хорошая рассылка новинок.

Самое главное, что нам понадобится, — это собственно сам архив рассылок. Сначала нужно выделить для этого отдельный каталог и определиться с форматом хранимых рассылок. Для каждого сеанса рассылки максимально может понадобиться хранить по три файла — обычный html с собственно рассылкой, zip для тех, кто

предпочитает скачивать информацию, и, наконец, текстовый файл с кратким содержанием или оглавлением рассылки (чтобы пользователь представлял себе, что и зачем он скачивает или смотрит). Причем файлы html и zip рекомендую хранить в разных подкаталогах. Это нужно для увеличения быстродействия всей системы в целом, так как осуществлять поиск будем по файлам, и лишние в этом каталоге нам ни к чему.

Итак, для хранения кода программы нам понадобится всего один файл. Назовите его так, как вам нравится, это не принципиально, и сделайте ссылку на него на своем сайте — «Архив рассылок». Теперь откройте сам файл в текстовом редакторе, и можно начинать программировать.

```
$katalog_archiv = "archiv/";
```

В этой переменной у нас хранится имя каталога, в котором находятся сами рассылки. Такие переменные лучше определять заранее, так как может понадобиться их изменить. Это будет легко сделать, поменяв значение переменной всего один раз, в начале.

Начало имени всех файлов у нас будет такое:

```
$name_temp = "Archiv_News!_";
```

Имена могут быть абсолютно любыми, так как отбрасываются в процессе работы, но с ними должны совпадать реальные имена в каталоге с архивом рассылок. Я, для удобства, принял для себя такой формат имен:

Начальное имя_число_месяц_год выпуска рассылки

Этого формата следует придерживаться для всех трех компонентов рассылки: файлов html, zip и txt. Последний также является файлом в формате html, но имеет расширение txt.

Открываем наш каталог с рассылками и ищем в нем файлы с расширением txt:

```
$dir_archiv = opendir{$katalog_archiv);

while($f = readdir($dir_archiv))

{
```

Вот почему я рекомендовал не хранить здесь остальные файлы, так как поиск усложняется, если файлов становится слишком много.

Сначала разделим имя проверяемого файла на составляющие, что понадобится нам в дальнейшем:

```
$f1 = trim(str_replace (".","~", $f) ) ;
@list ($id_name, $id_ras) = split ("~", $f1);
```

И вот, наконец, найден текстовый файл, можно начать вывод сообщения, так как это просто информация об очередной новинке, которую нужно показать на экране:

```
if ($id_ras == "txt")
// блок вывода сообщения об очередной дате новинки
{
```

Разделяем имя файла таким образом, чтобы выделить из него еще и дату рассылки. Нам не надо хранить ее отдельно, так как она присутствует в самом имени файла:

```
$f2 = trim(str_replace ($name_temp,"", $f));
$f2 = trim(str_replace (".txt","", $f2));

$f1= trim(str_replace ("_","~", $f2));
@list ($den, $m, $god) = split ("~", $f1);

$q[] = "";
$q[] = "января";
$q[] = "февраля";
$q[] = "марта";
$q[] = "апреля";
$q[] = "мая";
$q[] = "июня";
```

```
$q[] = "июля";
$q[] = "августа";
$q[] = "сентября";
$q[] = "октября";
$q[] = "ноября";
$q[] = "декабря";

if ($m == "01")  $m = 1;
if ($m == "02")  $m = 2;
if ($m == "03")  $m = 3;
if ($m == "04")  $m = 4;
if ($m == "05")  $m = 5;
if ($m == "06")  $m = 6;
if ($m == "07")  $m = 7;
if {$m == "08")  $m = 8;
if ($m == "09")  $m = 9;

$m = $q[$m];
```

Алгоритм немного грубый, но вы можете проявить свою фантазию и придумать что-то более простое и изящное.

Выводим на экран сообщение о рассылке и дате ее выхода:

```
echo "<p><b>Рассылка от $den $m 20$god года: </b><br>
Содержание:<br><br>";
```

Подготавливаем к рассылке текстовый файл:

```
include ("$katalog_archiv$name_temp$f2.txt");
```

Несмотря на расширение, HTML-код в нем будет выполнен. Осталось вывести ссылки для просмотра и скачивания нашей рассылки:

```
echo"<br><br><a href=$katalog_archiv$name_temp$f2.html>
Посмотреть</a> - <a href=$katalog_archiv$name_temp$f2.zip>
Скачать</a></p>";
```

И конечно, не забудьте завершить цикл поиска файлов:

```
} }
```

Вот и все, как видите, тоже совсем несложно. Обратите внимание, я оставил файлы html и zip в одном каталоге с файлами txt, хотя сам же намекал, что стоит их разнести для ускорения работы. Это я сделал сознательно для того, что бы вы сами продумали этот вопрос и потренировались на этом примере.

Простая аутентификация

Вопросы безопасности всегда волновали программистов и их работодателей. Первые бьются над извечным вопросом: как сделать, чтобы систему не «взломали». А вторые требуют от первых практически невозможного. В этой главе я дам несколько советов и попробую подсказать направление для ваших мыслей.

Совет первый: а нужно ли вам это?

Прежде чем начать, подумайте, стоит ли овчинка выделки. Нужно ли блокировать информацию, настолько ли она секретна и нужна ли кому-то еще, кроме вас?

Приведу пример. Одна фирма пожелала сделать отдельный прайс для дилеров с более низкими ценами. Разумеется, эта информация могла быть доступна только тем, кто является таким дилером, и никому более. Возникла парадоксальная ситуация. Пока информация была закрыта, ею пользовались только уже зарегистрированные дилеры. Как только информация стала доступна всем желающим, в фирму стали обращаться другие дилеры с предложениями о сотрудничестве. Оборот вырос. Сокрытие важной, казалось бы, информации пошло только во вред.

И таких примеров достаточно много. Не стоит все усложнять там, где можно обойтись более простыми решениями. И все же иногда этого не избежать.

Совет второй: забудьте все советы

Как ни странно, но это так: перед тем как разрабатывать свою систему безопасности, надо забыть все, что вы слышали раньше, и делать действительно свою систему. Копирование ни к чему хорошему не приведет, так как по скрытым особенностям всегда есть возможность подобрать алгоритм «взлома». Хорошая система — всегда индивидуальна, и ее алгоритм — информация сугубо секретная.

Но чтобы разобраться с принципами работы системы безопасности, рассмотрим следующий пример.

Пример системы безопасности

Давайте вернемся к той фирме, которая хотела спрятать прайс для дилеров, и попробуем подобным образом укрыть от чужого взгляда некоторую информацию. Для этого надо поместить файл в далекий-далекий каталог, запретить к нему доступ при помощи, например, файла .htacces и придумать уникальное имя для архива с информацией.

Далее, на входной странице размещаем следующую форму:

```
<h5>0птовый прайс</h5>
<form name="diler_vhod" method="post" action="diler.phtml">
<b>Пароль для входа:</b><br>
<input name="diler" type="password">
<input type="submit" value="Вход">
</form>
```

В эту форму (рис. 31) наши пользователи должны вводить пароль, и в зависимости от того, правильный он или нет, программа будет или разрешать команду на скачивание файла, или отказывать в доступе.

Рис. 31. Форма для ввода пароля

Сама программа имеет следующий вид:

```
<?
$add_price = "memo/price_diler.zip";
```

По этому адресу находится скрытый от всех файл. Его имя и имя каталога рекомендуется изменить так, чтобы нельзя было догадаться об их содержимом.

```
$pass = "3467899";
```

Так задаем пароль, по которому система разрешит вход.

Можно указать страницу, на которую попадет посетитель, если неправильно введет пароль. Я задал пустую строку, и по ней посетитель вообще никуда не попадет:

```
$no_file = "";
```

Следующий код проверяет введенные данные на соответствие правильному паролю:

```
if ($diler == $pass)
{ header ("Location: $add_price"); }
else { header ("Location: $no_file"); }
?>
```

Если пароль совпадает, даем команду на скачивание архива, если нет — открываем страницу с сообщением об ошибке. А еще лучше — совсем ничего не открывать, тогда тот, кто не знает пароля, будет думать, что система просто не работает в данный момент, и, в конце концов, ему просто надоест.

HTTP-аутентификация в PHP

HTTP-аутентификация в PHP возможна только при запуске PHP как Apache-модуля и, следовательно, недоступна в CGI-версии. В PHP-скрипте для Apache-модуля можно использовать функцию header () для отправки сообщения «Authentication Required» в клиентский браузер, что вызывает появление в нем окна для ввода имени пользователя и пароля.

После того как пользователь ввел логин и пароль, PHP-скрипт будет вызван снова с инициированными переменными окружения $PHP_AUTH_USER, $PHP_AUTH_PW и $PHP_AUTH_TYPE, в которых установлены имя пользователя, пароль и тип аутентификации, соответственно. Вот стандартный пример скрипта, который демонстрирует аутентификацию клиента:

```
<?php
 if  (!isset($_SERVER["PHP_AUTH_USER"]))  {
 header("WWW-Authenticate: Basic realm=\"My Realm\"");
 header("HTTP/1.0 401 Unauthorized");
 echo "Текст, отправляемый в том случае,
 если пользователь нажал кнопку Cancel\n";
 exit;
 } else {
 echo "<p>Привет {$_SERVER[`PHP_AUTH_USER`]}.</p>";
 echo "<p>Вы ввели пароль {$_SERVER[`$PHP_AUTH_PW ] } .</p>";
 }
?>
```

Таким образом, проверив указанные переменные окружения, легко принять решение о разрешении доступа, а уж затем вывести любую нужную информацию. Чтобы гарантировать совместимость со всеми браузерами, ключевое слово "Basic" должно быть записано с первой «В» в верхнем регистре, параметр функции realm должен заключаться в двойные кавычки (не одинарные), и ровно один пробел должен предшествовать числу "401" в строке "HTTP/1.0 401" заголовка.

Защита программы

Если вас не раз посещала мысль о возможности (или точнее, о невозможности) изменения ваших скриптов, знайте: есть способ заставить не работать программу, если в ней меняется хоть один байт. Правда, рассчитан этот способ в основном на тех, кто не очень разбирается в PHP, но ведь программисту обычно именно с такими заказчиками и приходится иметь дело.

Так вот в чем заключается мой метод. Создаем программу, настраиваем ее до той степени, когда вносить изменения уже не понадобится. В любом месте кода пишем какой-нибудь комментарий и обрамляем его большим количеством дефисов (не меньше ста). Далее в любом месте скрипта задается проверка на количество байтов в файле с программой. Сделать это легко, так как в PHP есть функция определения количества байтов в файле: filesize (name_file). В начале файла указываем его имя и количество выданных нам функцией байтов:

```
$ewq = "index.phtml";
$lo_call = 6507;
```

Дальше проверяем, соответствует ли запрошенное количество байтов тому, что введено в программу. Если нет — программа останавливается. Значение переменной $lo_call придется подбирать, но это не страшно — приблизительный размер мы определили, а регулировать его в любую сторону можно при помощи наших комментариев (помните, мы ставили дефисы?). Таким образом, добиваемся чтобы скрипт работал. Теперь при малейшем изменении он останавливается или выдает ошибку.

PHP в вопросах и ответах

Что такое PHP и что он может?

PHP — набор средств для выполнения кода на стороне сервера. Кроме того, **PHP** — язык программирования для написания такого кода. Появился в 1994 году как небольшая оболочка, исполняющаяся на стороне сервера. Отличается простотой изучения и полной интеграцией (в отличие от других приложений на стороне сервера) с кодом **HTML.** Это действительно так, ведь код **PHP** можно вставить в любое место странички, достаточно открыть его специальным тегом — <?php **и** закрыть ?>

Почему нельзя одновременно использовать PHP- и HTML-редакторы?

Можно, просто это неудобно. HTML-редакторы не понимают PHP-код, считая его обычным текстом. Для нормальной работы в редакторе **HTML** достаточно открывать код **PHP** так: <script language="php">, а закрывать, соответственно, так: </script>. Теперь все будет в порядке, PHP-код не будет вам мешать редактировать страницу.

Какие расширения могут быть у файла, содержащего PHP-код?

Можно сказать — любые. В настройках сервера, как правило, задаются в обязательном порядке php, php3, php4 и phtml. Я рекомендую использовать последнее, так как оно обеспечивает совместимости всех версий **PHP.** Часто спрашивают, как «подружить» **PHP** и SSI? Ответ напрашивается сам собой — записать в файле конфигурации **PHP** расширение shtml. И добавить туда все остальные расширения, для разнообразия. Конфигурационный файл находится на сервере, поэтому надо обращаться в службу технической поддержки вашего хостинг-провайдера.

Как проверить работоспособность скрипта на домашнем компьютере?

Это один из самых «больных» вопросов. Дело в том, что просто так нельзя увидеть работу скрипта PHP, его код должен быть обрабо-

тан сервером. А значит, сервер надо установить у себя на домашнем компьютере. В таком случае — это набор программ, благодаря которым становится возможным использовать домашний компьютер как реальный сервер в интернете. Это то же самое программное обеспечение, которое работает и на настоящем сервере, так что несовместимостей и ошибок вы впоследствии не получите. Скрипт будет работать у вас так, как и на удаленном компьютере. Впрочем, бывают исключения, но это касается очень тонких моментов настройки. Серверы бывают разные, но я рекомендую воспользоваться Apache, так как это самый распространенный, нетребовательный к ресурсам, относительно легкий в настройке сервер.

В двух словах процесс настройки сервера таков. Скачиваем и устанавливаем Apache, затем инсталлируем PHP и настраиваем (см. гл. «Установка PHP»). Сервер начнет работать. Получить доступ к нему можно, обратившись по адресу 127.0.0.1.

Как сделать так, чтобы не закачивать каждый раз файлы для проверки на сервер, т.е. можно ли проверять работоспособность PHP в офлайне?

Для простой проверки PHP-кода можно не устанавливать сервер. В этом случае делайте так: `php.exe -f index.php | more`. Только нужно сохранить все пути такими, какие они есть на самом деле.

Как сделать запись в файл? Все делаю, как положено, но там, где должна идти запись в файл, PHP выдает ошибку.

Скорее всего, вы не установили для файла атрибуты на запись. Сделать это очень просто. В вашем FTP-менеджере посмотрите свойства уже закачанного файла на сервере. Атрибуты его, скорее всего, будут установлены только для чтения. Вам надо изменить их на запись. Код атрибутов должен быть 666 или 777. После этого все будет работать, если в скрипте нет ошибки.

Как можно узнать содержимое файла с расширением php, который находится на каком-либо сервере?

В этом-то и заключается вся прелесть PHP (для администраторов сайтов, конечно), что никак нельзя. Нужно быть владельцем сер-

вера, иначе — практически невозможно. По крайней мере, я о таком не слышал.

Мне необходимо конвертировать одинарные кавычки в обратные слеши с одинарными кавычками. Как это можно сделать с помощью **регулярного выражения?**

Гораздо удобнее, чем регулярное выражение, использовать стандартную функцию addslashes (), которая обрамляет строку слешами. Она возвращает строку со слешами перед нужными символами (например, в запросах БД). Это одинарная и двойная кавычки, обратный слеш и нулевые значения. Директива magic_quotes_gpc в конфигурационном файле должна быть включена (установлена в положение ON).

Где можно найти полный список зарезервированных переменных и почему они не описаны в документации PHP?

Можно воспользоваться командой <?php phpinfo (); ?> , которая даст возможность просмотреть все виды информации о вашей установке **PHP,** включая список переменных окружения, а также специальных переменных, установленных вашим Web-сервером. Этот список действительно не может быть описан в документации PHP, поскольку он будет отличаться для каждого сервера.

Я пытаюсь получить доступ к одной из стандартных переменных CGI (таких как $DOCUMENT_ROOT или $HTTP_REFERER) в пользовательской функции, и не могу их найти. Что не так?

Переменные окружения являются обычными глобальными переменными, поэтому вы обязаны либо объявлять их как глобальные переменные в вашей функции (используя, например, "global$DOCUMENT_ROOT;") или применить массив глобальных переменных (т.е. "$GLOBALS ["DOCUMENT_ROOT"] ").

Как создать массивы в HTML-теге <form>?

Чтобы получить результат из формы, отправленный как массив **в** какой-либо PHP-скрипт, вы должны переименовать элементы тегов <input>, <select> или <textarea> таким образом:

```
<input name="Array[]">
<input name="Array[]">
<input name="Array[] ">
<input name="Array[]">
```

Обратите внимание на квадратные скобки после имен переменных, это то, что и делает их массивом. Вы можете группировать элементы в разные массивы, присвоив одно имя разным элементам:

```
<input name="Array[]">
<input name="Array[]">
<input name="OtherArray[]">
<input name="OtherArray[] ">
```

Это создаст два массива — Array и OtherArray, которые будут отправлены PHP-скрипту. Можно также присвоить вашим массивам специфические ключи:

```
<input name="AnotherArray[]">
<input name="AnotherArray[]">
<input name="AnotherArray[email]">
<input name="AnotherArray[phone]">
```

Массив AnotherArray теперь будет содержать ключи 0, 1, email и phone. Все, что касается обычных массивов, применимо и тут.

Что такое зарезервированные слова в PHP и где можно посмотреть их полный список?

Эти слова имеют специальные значения в PHP. Некоторые из них выглядят как функции, некоторые — как константы, но они на самом деле таковыми не являются: это конструкции языка. Вы не можете использовать какое-либо из этих слов как константу, имя класса или имя функции. Использование их в качестве имен переменных обычно не приводит к каким-либо последствиям, но лучше отнестись к ним внимательно, а еще лучше — просто не использовать. Вот их список:

```
and                    for
array()                foreach
as                     function
break                  global
case                   if
cfunction              include ()
class                  include_once()
const                  isset ()
continue               list ()
declare                new
default                old_function
die()                  or
do                     print()
echo()                 require ()
else                   require_once ()
elseif                 return ()
empty()                static
enddeclare             switch
endfor                 use
endforeach             var
endif                  while
endswitch              xor
endwhile               __CLASS__
eval                   __FILE__
exit ()                __FUNCTION__
extends                __LINE__
```

Как проверить наличие в какой-либо переменной, полученной из формы, числа или строки?

Перед передачей переменной преобразуйте ее, например, командой

```
$a = intval($a);
```

После этого на 100 % можно быть уверенным, что $a имеет целочисленное значение. Пример конвертации:

```
"aaaaaa" -> 0
"123aaa" -> 123
"123" -> 123
"123.55" -> 124
```

Если нужно получить дробное значение, используйте функцию doubleval (), если символьное — strval () и т.д.

Приложения

Приложение 1. Что такое HTML и CSS

HTML (HyperText Markup Language) — это язык разметки гипертекста. Что такое «гипертекст»? Это значит, например, то, что текст не простой, а оформлен с помощью специальных тегов. В таком случае любое место текста можно пометить специальной меткой и из другого места этого же документа легко перейти к поставленной метке. Таким образом, все документы могут быть объединены в единое целое, а переходы между документами организовываются очень просто.

У нас тут промелькнуло слово «теги» — предлагаю расшифровать и его. Теги — это специальные метки. Они указывают браузеру пользователя, что именно надо сделать с текстом, картинкой и т.д. Как правило, теги делятся на открывающие и закрывающие. Сделано это для того, чтобы браузер мог узнать зону действия тега.

В качестве примера рассмотрим простой случай — выравнивание текста по центру. Для этого в языке HTML есть тег . Это — открывающий тег. Есть у него и закрывающий тег — . Любой текст, заключенный между этими HTML-тегами будет выведен по центру экрана. Например:

 Этот текст будет выровнен по центру

Можно сформулировать несколько общих правил использования HTML:

- все теги заключаются в угловые скобки < >;
- закрывающий тег идентичен открывающему, только в закрывающем теге перед ключевым словом ставится косая черта (слеш);

- теги, нуждающиеся в закрытии, обязательно должны быть закрыты;
- теги, как правило, «говорящие», т.е. имеют близкое значению имя.

Запомните эти правила, они вам очень помогут в дальнейшем.

Надо сказать, что не все теги имеют закрывающих «двойников». Некоторые в них просто не нуждаются. К таким относятся, например, тег перевода строки
, тег-разделитель <hr>, метатеги (содержащие вспомогательную информацию) и т.д. Если тег не требует завершения, это специально оговаривается.

Вообще список HTML-тегов и их значений нужно смотреть в многочисленных специальных книгах по HTML.

В последнее время при создании сайтов все большую популярность приобретают каскадные таблицы стилей (CSS, Cascading Style Sheets). Их использование позволяет избавиться от многих дизайнерских проблем, часть из которых и вовсе не решалась стандартными средствами HTML.

Приложение 2. Самые частые ошибки

- Самая распространенная — не стоит завершающая оператор точка с запятой. Не ставится она только в конструкции if () {}else{} Основной признак — ошибка в строке, следующей за тем оператором, в котором нет точки с запятой. В окно браузера, если только это разрешено, посылается сообщение об ошибке, имя и путь к файлу, в котором произошла ошибка, и порядковый номер строки. Выглядит это следующим образом:

```
Parse error: parse error in c:\usr\local\public_html\
host:\virtual\index.phtml on line 2
```

- При сравнении на равенство двух переменных ставится два знака равенства. Исключений нет. Основной признак — скрипт работает, но не так, как от него ожидается. Найти такую

ошибку бывает порой очень трудно. Ведь сообщения не появляются, так как с точки зрения интерпретатора ошибки нет. Бороться можно только одним способом: внимательно писать сравнения и не забывать об этом правиле.

- В конструкции if() {}else{} отсутствует или стоит лишняя завершающая скобка. Исключений нет. Признак этой ошибки очень простой и позволяет обнаружить ее наличие очень просто. При запуске скрипта интерпретатор выдает ошибку в самой последней строке кода с завершающим тегом ?> Однако найти, где именно не стоит кавычка, бывает непросто. Придется пересмотреть весь код. Иногда меня выручала обычная интуиция — вместо того чтобы искать ошибку, исследуя весь код, я ставил скобку в предполагаемом месте и смотрел, как работает программа. Раза с пятого все получалось.

- Не забывайте про регистрозависимость переменных, например: $user не одно и то же, что и $User. Это совершенно разные переменные. Из-за этого могут возникать самые разные ошибки.

  Функции в PHP (как встроенные, так и определенные разработчиком) не подчиняются этому правилу: они регистронезависимы.

- Еще одна распространенная ошибка связана с заголовками.

  Как работает браузер с сервером? По специальному протоколу, который называется HTTP (HyperText Transport Protocol, протокол передачи данных в формате HTML, т.е. гипертекста). Когда вы вводите адрес в адресной строке браузера и нажимаете ввод, программа посылает HTTP-запрос серверу, и на этот запрос сервер всегда отвечает, если только он доступен. Первыми в ответе сервера всегда идут так называемые HTTP-заголовки и только потом может идти код страницы. Сервер устроен так, что заголовки посылаются автоматически перед любыми символами, даже перед пробелом. Это значит, что, если хоть один символ передан в браузер, заголовки уже находятся в браузере и вторично послать их никак нельзя. А PHP умеет работать с этими заголовками, используя их при работе, например, с сессиями, cookie и т.д.

Вот здесь возникает ошибка. Ее текст в окне браузера может выглядеть примерно так:

```
Warning: Cannot add header information - headers already sent by (output started at /site/name.phtml:3) on line 5.
```

Эта строка говорит о том, что произошла ошибка отправки заголовков. PHP сообщает, в каком скрипте и в какой его строке(output started at /site/name.phtml:3) произошел вывод информации, из-за которого автоматически послались заголовки. Зная номер строки, очень легко найти и исправить ошибку. Может быть, там HTML-теги, может быть, функция echo, а может, и просто пустая незамеченная строка или пробел перед первым тегом <?

Очень часто такую ошибку вызывает файл, который подключают при помощи функции include и в котором либо есть какой-то вывод, либо пустая строка после закрывающего PHP-тега. Обнаружить такую ошибку очень трудно. Для решения этой проблемы нужно функцию header () (или session_start (), или setcookie ()) поместить до вывода любого значения. Просто перенести повыше в скрипте.

Приложение 3. Некоторые функции PHP

Для работы со строками

- addslashes — выделяет строку обратной чертой. Ее синтаксис: string addslashes (string str); Возвращает строку с обратной чертой перед символами, которые должны быть выделены в запросах к базам данных и т.п. Это символы: ('), двойные кавычки ("), (\) и нулевые значения.
- chop — удаляет повторяющиеся пробелы. Ее синтаксис: string chop (string str); Сама строка задается на месте параметра str как в самой функции, так и при помощи переменной.
- convert_cyr_string — переводит строку из одной русской кодовой таблицы в другую. Синтаксис команды: string

`convert_cyr_string(string str, string from, string to);` Аргументы from и to являются одним символом, который определяет исходную и целевую кодовую таблицу. Поддерживаемые типы:

k — koi8-r;

w — windows-1251;

i — iso8859-5;

a — x-cp866;

d — x-cp866;

m — x-mac-Cyrillic

- echo — выводит на экран одну или более строк. Ее синтаксис: `echo {string arg1, string [argn] . . .);` Выводит все параметры. В действительности не является функцией (это языковая конструкция), поэтому не обязательно использовать круглые скобки. Например:

```
<?
echo "Привет, народ!"; /* выведет фразу "Привет,
народ" */?>
<?
echo "Позволяет выводить сложные предложения. Все
эти строки будут выведены.";
?>
```

Еще один пример:

```
<?
$a= 5;
$b= 3;
$c= 4;
$d= $a + $b - $c;
echo $d;
?>
```

В результате на экран будет выведено число 4.

```
<?
echo "$d";
?>
```

В результате этого кода будет выведено:

```
$d
```

- explode — разбивает строку на подстроки. Ее синтаксис: `array explode(string separator, string string [, int limit]);` Возвращает массив строк, содержащий элементы, разделенные строкой `separator`. Необязательный параметр `limit` задает количество элементов нового массива.

  Например:

```
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
5pieces = explode(" ", $pizza);
```

- get_meta_tags — извлекает все содержимое атрибутов метатегов из файла и возвращает в виде массива. Ее синтаксис: array get_meta_tags (string filename, int [use_include_path]) ; Открывает файл filename, обрабатывает его строка за строкой и извлекает метатеги. Например:

```
<meta name="author" content="name">
<meta name="tags" content="php3 documentation">
```

  Значение свойства name становится ключом, значение свойства `content` становится значением возвращаемого массива, поэтому вы можете легко использовать стандартные функции для его обработки или доступа к отдельным элементам. Специальные символы в значении свойства заменяются символом '_', остальные переводятся в нижний регистр. Установка параметра use_include_path в 1 приведет к тому, что PHP будет пытаться открыть файл по стандартному пути `include`.

- htmlentities — переводит все возможные символы в коды HTML. Ее синтаксис: string htmlentities (string string); Эта функция идентична htmlspecialchars (), кроме того что все символы, которые имеют соответствующий код HTML заменяются этим HTML-кодом. В настоящее время применяется кодовая таблица ISO-8859-1.

- htmlspecialchars — переводит специальные символы в коды HTML. Ее синтаксис: string htmlspecialchars (string string) ; Определенные символы имеют особое значение в HTML и должны быть заменены кодами HTML, если они таковые имеют. Эта функция возвращает строки с такими изменениями. Она полезна для отчистки полученного от пользователя текста от разметки HTML (доски сообщений, гостевые книги). В настоящее время осуществляются следующие замены:

  & (амперсанд) становится &

  " (двойные кавычки) становится "

  < (знак меньше) становится <

  > (знак больше) становится &gt.

  Следует отметить, что эта функция не заменяет ничего, кроме указанного выше. Для полной обработки обратите внимание на функцию htmlentities ().

- implode — объединяет массив элементов в строку. Ее синтаксис: `string implode(array pieces, string glue);` Возвращает строку, содержащую совокупность всех элементов массива, в том же порядке со строкой glue между каждым элементом.

  Например:

```
$array = array("lastname", "email", "phone");
$a = implode(",", $array);
echo $a;
?>
```

На экран будет выведено:

```
lastname,email,phone
```

- `join` — аналогична implode
- `ltrim` — удаляет пробелы в начале строки. Ее синтаксис: `string ltrim(string str);`
- `nl2br` — переводит символы перехода строки в HTML-тег разрыва строки. Ее синтаксис: `string nl2br(string string);` Возвращает `string` с
, вставляемыми перед каждой новой строкой.
- `parse_str` — разделяет строку на переменные. Ее синтаксис: `void parse_str(string str);` Разбивает строку `str`, как если бы она была URL-строкой запроса, и устанавливает переменные текущей среды.

Например:

```
<?php
$str = "first=value&arr[]=foo+bar&arr[]=baz";
parse_str($str) ;
echo $first; // value
echo $arr[0]; // foo bar
echo $arr[1]; // baz

parse_str{$str, $output) ;
echo $output[`first`]; // выведет value
echo $output['arr'][0]; // выведет foo bar
echo $output['arr'][1]; // выведет baz
?>
```

- `print` — выводит строку. Ее синтаксис: `print{string arg) ;` Аналогична функции echo.
- rawurldecode — возвращает строку, в которой последовательность символов % и двух шестнадцатеричных цифр заменяется соответствующим буквенным символом. Ее синтаксис: `string rawurldecode(string str);` Например, строка admin%20log%40name будет заменена admin log@name.

- `strlen` — возвращает длину строки. Ее синтаксис: `int strlen(string str);` Например:

```
<?php
$str = "abcdef";
echo strlen($str);
```

В результате на экран будет выведено «6».

- `str_replace` — заменяет все указанные строки на заданную последовательность символов. Ее синтаксис: `mixed str_replace(mixed search, mixed replace, mixed subject);` Заменяет указанные комбинации символов `search` в строке `subject` на комбинацию `replace` и возвращает полученную строку или массив.

Например:

```
<?
$string = "Я ходил в кино";
$strnew = str_replace("кино", "театр", $string);
```

В результате переменная `$strnew` будет иметь значение «`Я ходил в театр`».

- `strcmp` — двоичное сравнение строк (безопасное). Ее синтаксис: `int strcmp(string str1, string str2);` Возвращает <0, если `str1` меньше чем `str2`;. Возвращает > 0, если `str1` больше чем `str2`, и 0, если они равны. Обратите внимание, что это сравнение чувствительно к регистру.

- `stripslashes` — удаляет слеши из строки. Ее синтаксис: `string stripslashes(string str);` Возвращает строку `str` с вырезанными слешами. (\" заменяется на " и т.д.). Двойные \\ заменяются на \. Например:

```
<?php
$str = "Вы правнук О\"Генри?";
echo stripslashes($str);
?>
```

В результате на экран будет выведено:

Вы правнук О'Генри?

- strrpos — находит позицию последнего появления символа в строке. Ее синтаксис: int strrpos (string haystack, char needle) ; Возвращает номер позиции последнего появления символа needle в строке haystack. Обратите внимание, что needle в этом случае может быть только единственным символом. Если в качестве параметра needle указывается строка, то только первый символ будет использован. Если needle не найден, то возвращается false. Если параметр needle не является строкой, то он переводится в десятичное число и рассматривается как числовое значение символа.

- strrchr — находит последнее появление символа в строке. Ее синтаксис: string strrchr (string haystack, string needle) ; Эта функция возвращает позицию haystack, с которой начинается последнее появление needle и продолжается до конца haystack. Возвращает false, если needle не найден. Если параметр needle содержит более чем один символ, то используется первый символ. Если параметр needle не является строкой, то он переводится в целое число и рассматривается как числовое значение символа. Например:

```
// получение последней директории в $PATH
$dir = substr( strrchr( $PATH, ":" ), 1 );
/* получение всех символов до конца строки после
последней новой строки */
$text = "Line 1\nLine 2\nLine 3";
$last = substr ( strrchr( $text, 10 ), 1 );
```

- strrev — переворачивает строку. Ее синтаксис: string strrev(string string); Например:

```
<?php
echo strrev("Привет, мир! " ) ;
?>
```

В результате на экран будет выведено:

```
!рим, тевирП
```

- strstr — находит первое появление строки. Ее синтаксис: string strstr(string haystack, string needle); Возвращает все haystack с первого появления строки needle

и до конца. Если параметр `needle` не найден, то возвращает `false`. Если параметр `needle` не является строкой, то он переводится в целое число и рассматривается как числовое значение символа.

Например:

```
<?php
$address = "autc@yandex.ru";
$a = strstr($address, "@");
echo $a;
?>
```

В результате на экран будет выведено:

@yandex.ru

- `strtok` — разбивает строку на части. Ее синтаксис: `string strtok(string arg1, string arg2);` Например, строку «This is an example string» можно разбить на отдельные слова, используя пробел в качестве разделителя:

```
$string = "This is an example string";
$tok = strtok($string," ");
while($tok) { echo "Word=$tok<br>"; $tok = strtok(" ");}
```

В результате на экран будет выведено:

```
This

is

an

example

string
```

Обратите внимание, что только первый вызов функции `strtok` использует строковый аргумент. Для каждого последующего вызова функции `strtok` необходим только разделитель, так как это позволяет контролировать положение в текущей строке. Чтобы начать сначала или разбить новую строку, вам необходимо просто вызвать `strtok` с парамет-

ром строки опять для ее инициализации. Вы можете вставлять несколько разделителей в параметр разделителя. Строка будет разделяться при обнаружении любого из указанных символов. Также будьте внимательны к разделителям, равным 0. Это может вызвать ошибку в определенных выражениях.

- strtolower — переводит строку в нижний регистр. Ее синтаксис: string strtolower(string str); Возвращает строку string со всеми буквенными символами, переведенными в нижний регистр. Помните, что буквенные символы определяются установками сервера на локальном компьютере.

  Например:

```
<?php
$a = "У меня ЕсТь ВКУСНАЯ конфетка";
$a = strtolower($a);
echo $a;
?>
```

  В результате на экран будет выведено:

  у меня есть вкусная конфетка

- strtoupper — переводит строку в верхний регистр. Ее синтаксис: string strtoupper(string string); Возвращает строку string со всеми буквенными символами, переведенными в верхний регистр. Следует отметить, что буквенные символы определяются установками сервера на локальном компьютере.

Например:

```
<?php
$a = "У меня ЕсТь ВКУСНАЯ конфетка";
$a = strtoupper $a);
echo $a;
?>
```

В результате на экран будет выведено:

```
У МЕНЯ ЕСТЬ ВКУСНАЯ КОНФЕТКА
```

- strtr — переводит определенные символы. Ее синтаксис: `string strtr(string str, string from, string to);` Например: `$addr = strtr($addr, "дец", "aao");` Эта функция обрабатывает строку `str`, заменяя все появления каждого символа из строки `from` на соответствующие символы в строке `to`, и возвращает результат. Если строки `from` и `to` имеют различную длину, то дополнительные символы более длинной из строк игнорируются.

- substr — возвращает часть строки. Ее синтаксис: `string substr(string string, int start, int [length]);` Эта функция возвращает часть строки `string`, определяемую параметрами `start` (начало) и `length` (длина). Если параметр `start` положительный, то возвращаемая строка будет начинаться с символа, находящегося на позиции `start` строки `string`. Например:

```
$rest = substr("abcdef", 1); // вернет "bcdef"
$rest = substr("abcdef", 1, 3); // вернет "bed"
```

Если параметр `start` отрицательный, то возвращаемая строка будет начинаться с символа, находящегося на позиции `start` от конца строки `string`. Например:

```
$rest = substr("abcdef", -1); // вернет "f"
$rest = substr("abcdef", -2); // вернет "ef"
$rest = substr("abcdef", -3, 1); // вернет "d"
```

Если параметр `length` указан и он положительный, то возвращаемая строка закончится за `length` символов от начала `start`. Это приведет к строке с отрицательной длиной (потому что начало будет за концом строки), поэтому возвращаемая строка будет содержать один символ от начала строки `start`. Если `length` указан и он отрицательный, то возвращаемая строка закончится за `length` от конца строки `string`. Это приведет к строке с отрицательной длиной, поэтому возвращаемая строка будет содержать один символ от начала строки `start`. Например:

```
$rest = substr("abcdef", -1, -1); // вернет "bede"
```

- trim — удаляет пробелы с начала и с конца строки. Ее синтаксис: `string trim(string str);`
- ucfirst — переводит первый символ строки в верхний регистр. Ее синтаксис: `string ucfirst (string str) ;` Делает заглавным первый символ строки `str`, если этот символ буквенный. Следует напомнить, что буквенные символы определяются установками сервера на локальном компьютере.

  Например:

```
$a = "привет, мир!";
$a = ucfirst ($foo);
```

  В результате на экран будет выведено:

  Привет, мир!

- ucwords — переводит в верхний регистр первые символы каждого слова в строке. Ее синтаксис: `string ucwords (string str);` Делает заглавным первый символ каждого слова в строке `str`, если этот символ буквенный.

  Например:

```
$foo = "привет, мир!";
$foo = ucwords($foo);
```

  В результате на экран будет выведено:

  Привет, Мир!

Для работы с файлами

- basename — возвращает из полного пути имя файла. Ее синтаксис: `string basename (string path);` Получив строку, содержащую путь к файлу, данная функция возвратит основное имя файла. В Windows оба слеша — прямой (/) и обратный (\) — используются как разделители при задании пути. В других окружениях это только прямой слеш (/).

```
$path = "/home/httpd/html/index.php3";
$file = basename($path); /* $file устанавливается
в "index.php3"*/
```

- copy — копирует файл. Ее синтаксис: `int copy (string source, string dest) ;` Возвращает `true` при успешном завершении, в противном случае — `false`. Например:

```
if (!copy($file, $file.'.bak'))
{
print("при копировании файла $file произошла ошибка...<br>\n");
}
```

  В результате, если при копировании файла `$file` произойдет ошибка, то на экран будет выведено:

  при копировании файла $file произошла ошибка...

- dirname — возвращает путь к директории, в которой расположен файл. Ее синтаксис: `string dirname (string path) ;` Получив строку, содержащую путь к файлу, данная функция возвратит путь к директории, содержащей файл. В Windows оба слеша — прямой (/) и обратный (\) — используются как разделители при задании пути. В других окружениях это только прямой слеш (/).

  Например:

```
$path = "/etc/passwd";
$file = dirname($path) ; // $file расположен в "/etc"
```

- fclose — закрывает файловый указатель. Ее синтаксис: `int fclose (int fp);` Это показывает интерпретатору, что все действия над содержимым файла fp будут прекращены. Возвращает `true` при удачной операции и `false` при ошибке. Указатель должен быть действующим и указывать на файл, успешно открытый функциями fopen () или fsockopen ().

- feof — проверяет, находится ли указатель в конце файла. Ее синтаксис: `int feof(int fp);` Возвращает `true`, если указатель файла равен EOF (End Of File, конец файла) или при возникновении ошибки. В противном случае возвращает `false`. Указатель должен быть действующим и указывать на файл, успешно открытый функциями fopen (), popen () или fsockopen ().

- fgetc — получает символ из файла. Ее синтаксис: string fgetc(int fp); Возвращает строку, содержащую один символ, прочитанный по файловому указателю fp. При EOF возвращает false. Указатель должен быть действующим и указывать на файл, успешно открытый fopen (), popen (), или fsockopen().
- fgets — получает строку по указателю на файл. Ее синтаксис: string fgets (int fp, int length); Возвращает строку до length — читается по одному байту из файла, указанного в fp. Чтение заканчивается, если прочитано length символов, или до символов перевода строки и возврата каретки, или до EOF. Один байт прочитается в любом случае. При ошибке возвращает false. Указатель должен быть действующим и указывать на файл, успешно открытый функциями fopen (), popen () или fsockopen ().
- fgetss — получает строку в соответствии со значением файлового указателя и вырезает HTML-теги. Ее синтаксис: string fgetss(int fp, int length); Подобна fgets (), но отличается тем, что удаляет HTML- и PHP-теги из прочитанного текста.
- file — читает и записывает файл в массив. Ее синтаксис: array file {string filename); Каждая строка файла (вместе с символом перехода строки) будет соответствовать элементу массива.
- fileatime — показывает время последнего обращения к файлу. Ее синтаксис: int fileatime (string filename); Возвращает false в случае ошибки.
- filectime — выводит время последнего изменения файла в Unix. Ее синтаксис: int filectime (string filename); Возвращает false в случае ошибки. Работает только для ОС Unix.
- file_exists — проверяет существование искомого файла. Ее синтаксис: int file_exists(string filename); Возвращает true, если файл, определенный в filename, существует; иначе — false. Обратите внимание, что эта функция

неприменима для работы с удаленными файлами, поскольку файл должен быть доступен через файловую систему сервера.

Например:

```
<?php
$filename = "/path/to/my.txt";

if (file_exists($filename)) {
print "Файл $filename существует";
} else {
print "Файл $filename не существует";
}
?>
```

В результате, если указанный файл существует, на экран будет выведено:

Файл my.txt существует

Если же указанный файл не существует:

Файл my.txt не существует

- filemtime — время последнего изменения файла во всех операционных системах, кроме Unix. Синтаксис команды: int filemtime (string filename) ; Возвращает false в случае ошибки. Например: $filename = "my.txt"; определит время последнего изменения в файле my.txt.
- filesize —размер файла. Ее синтаксис: int filesize (string filename); Возвращает false в случае ошибки.
- filetype — показывает тип файла. Ее синтаксис: string filetype (string filename); Возвращает false. Возможные значения: fifo, char, dir, block, link, file, unknown.
- fopen — открывает файл или URL. Ее синтаксис: int fopen (string filename, string mode); Если filename начинается с "http://" (без учета регистра), открывается HTTP-соединение с указанным сервером и возвращает указатель файла на открытый файл. В указании директории нужно включать завершающие слеши. Если filename начинается

с "ftp://" (без учета регистра), открывается FTP-соединение с указанным сервером и указатель возвращается на искомый файл. Если сервер не поддерживает режим пассивного **FTP,** данная операция завершится ошибкой. Вы можете открывать файлы как для чтения, так и для записи через **FTP** (но не обе операции одновременно). Если filename начинается как-нибудь иначе, открывается файл вашей файловой системы, и указатель возвращается на открытый файл. Если при открытии файла происходит ошибка, функция возвращает false. Параметр mode может принимать следующие значения:

- r — открывает только для чтения, помещает указатель на начало файла;
- r+ — открывает для чтения и для записи, помещает указатель на начало файла;
- w — открывает только для записи, помещает указатель на начало файла и очищает все содержимое файла. Если файл не существует, создает новый файл;
- w+ — открывает для чтения и для записи, помещает указатель на начало файла и очищает все содержимое файла. Если файл не существует, создает новый файл;
- a — открывает только для записи, помещает указатель на конец файла. Если файл не существует, создает новый файл;
- a+ — открывает для чтения и для записи, помещает указатель на конец файла. Если файл не существует, создает новый файл.

Например:

```
$fp = fopen("/home/rasmus/file.txt", "r");
$fp = fopen("http://www.php.net/", "r");
$fp = fopen("ftp://user:password@example.com/", "w");
```

Если у вас возникают проблемы с чтением и записью в файл при использовании PHP как серверного модуля, помните, что применяемые вами файлы и директории должны быть дос-

тупными для серверных процессов. На платформе Windows будьте осторожны, избегая обратных слешей в путях, и используйте прямые слеши:

```
$fp = fopen("c:\\data\\info.txt", "r") ;
```

- fpassthru — выводит все данные из указателя файла. Ее синтаксис: `int fpassthru (int fp) ;` Читает до EOF и передает результат на стандартное устройство вывода. При возникновении ошибки возвращает false. Файловый указатель должен быть действующим и указывать на файл, успешно открытый функциями fopen (), popen () или fsockopen ().

- fputs — записывает в файл. Ее синтаксис: `int fputs (int fp, string str, int [length]);` Функция fputs () аналогична fwrite ().

- fread — производит бинарное чтение файла. Ее синтаксис: `string fread (int fp, int length);` Читает байты из файла, на который ссылается fp до length. Чтение заканчивается, когда прочитано length байтов или достигнут EOF.

- fseek — производит поиск в файле. Ее синтаксис: `int fseek(int fp, int offset);` Для файла fp смещает указатель в потоке байтов на offset байтов. Эквивалентно вызову в языке программирования C функции fseek (fp, offset, SEEK_SET). При удачном выполнении возвращает 0, в противном случае возвращает —1. Поиск после EOF не рассматривается как ошибка. Не используется для файловых указателей, возвращенных функциями fopen(), если путь к файлу начинается "http://" или "ftp://".

- fwrite — производит бинарную запись в файл. Синтаксис: `int fwrite(int fp, string string, int [length]);` Записывает содержимое string в файловый поток, указанный fp. Если аргумент length присутствует, запись останавливается после записи байта, находящегося на позиции length, или после записи всей строки string. Если есть аргумент length, то соответствующие конфигурационные опции magic_quotes_runtime игнорируются и никакие слеши из string не удаляются.

- is_dir — проверяет, является ли указанное имя названием директории. Ее синтаксис: bool is_dir (string filename); Возвращает true, если filename существует и является директория.
- is_executable — проверяет, относится ли файл к классу исполнимых. Ее синтаксис: bool is_executable (string filename) ; Возвращает true, если filename существует и является исполнимым файлом.
- is_file — проверяет, относится ли файл к классу обычных файлов. Ее синтаксис: bool is_file (string filename); Возвращает true, если filename существует и является обычным файлом.
- is_link — показывает, относится ли файл к символическим ссылкам. Ее синтаксис: bool is_link (string filename) ; Возвращает true, если filename существует и является символической ссылкой.
- is_readable — проверяет, относится ли файл к классу читаемых. Синтаксис команды: bool is_readable (string filename) ; Возвращает true, если filename существует и является доступным для чтения. Помните, что права доступа определяются именем, под которым пользователь вошел в систему.
- is_writeable — показывает, относится ли файл к классу записываемых. Ее синтаксис: bool is_readable (string filename); Возвращает true, если файл существует и доступен для записи. Помните, что права доступа определяются именем, под которым пользователь вошел в систему.
- link — создает жесткую ссылку. Ее синтаксис: int link{string target, string link);
- mkdir — создает директорию. Ее синтаксис: int mkdir (string pathname, int mode); Пытается создать директорию, указанную в pathname. Обратите внимание, что если вы захотите указать mode в восьмеричной системе, то число должно начинаться с 0. Например:

```
mkdir("/path/to/my/dir", 0700);
```

Возвращает true при успешном выполнении.

- pclose — закрывает процесс файлового указателя, запущенный функцией popen(). Ее синтаксис: int pclose(int fp); Это дает понять интерпретатору, что все действия над содержимым файла pf будут прекращены. Файловый указатель должен быть действующим и указывать на файл, успешно открытый вызовом popen().

- readfile — читает файл и записывает его на стандартное устройство вывода. Ее синтаксис: int readfile(string filename); Также возвращает количество прочитанных байтов. В случае возникновения ошибки возвращает false и если перед readfile не указана @, то выводится сообщение об ошибке. Если filename начинается с "http://" (без учета регистра), открывается HTTP-соединение с указанным сервером и текст ответа выводится на стандартное устройство вывода. В указании директории нужно включать завершающие слеши. Если filename начинается с "ftp://" (без учета регистра), открывается FTP-соединение с указанным сервером и файл ответа выводится на стандартное устройство вывода. Если сервер не поддерживает режим пассивного ftp, этот вызов завершится ошибкой. Если filename начинается как-нибудь иначе, будет открыт файл вашей файловой системы и его содержимое выведется на стандартное устройство вывода.

- readlink — показывает цель символической ссылки. Ее синтаксис: string readlink(string path); Работает аналогично функции readlink языка C и возвращает содержимое символической ссылки path или 0 в случае ошибки.

- rename — переименовывает файл. Ее синтаксис: int rename(string oldname, string newname); Пытается изменить имя файла, указанного в параметре oldname на newname. Возвращает true при успешном выполнении и false при сбое.

- rewind — позиционирует файловый указатель. Ее синтаксис: int rewind(int fp); Позиционирует файловый указатель для fp на начало потока файла. При возникновении ошибки

возвращает 0. Файловый указатель должен быть действующим и указывать на файл, успешно открытый функцией fopen ().

- rmdir — удаляет директорию. Ее синтаксис: int rmdir (string dirname) ; Пытается удалить директорию, путь к которой указан в параметре dirname. Директория должна быть пустой и ее атрибуты должны допускать это. При возникновении ошибки возвращает 0.
- stat — информация о файле. Ее синтаксис: array stat (string filename); Собирает статистику о файле filename и возвращает массив статистической информации, например, с такими элементами:
  - device (устройство);
  - number of link (номер ссылки);
  - user ID owner (ID пользователя или владельца);
  - group ID owner (ID группы владельца);
  - size in bytes (размер в байтах);
  - time of last access (время последнего доступа);
  - time of last modification (время последнего изменения);
  - number of blocks allocated (количество занятых блоков) и т.д.
- lstat — выводит информацию о файле или символической ссылке. Ее синтаксис: array lstat (string filename) ; Собирает информацию о файле или символической ссылке filename. Эта функция подобна stat (), но если параметр filename функции lstat — это символическая ссылка, то возвращает статус символической ссылки, а не статус файла, на который указывает данная ссылка. Возвращает массив статистической информации, с такими же элементами, как и функция stat.
- symlink — создает символическую ссылку. Ее синтаксис: int symlink(string target, string link) ; Создает символическую ссылку с целью, указанной в параметре target, и именем, заданным в link.

- tempnam — создает уникальное имя файла. Ее синтаксис: string tempnam(string dir, string prefix); Создает уникальное имя файла в указанной директории. Если директория не существует, tempnam () генерирует имя файла во временной директории системы. Возвращает новое временное имя файла или нулевую строку при ошибке.

- touch — устанавливает время модификации файла. Ее синтаксис: int touch(string filename, int time); Пытается установить время модификации файла filename в значение time. Если параметр time отсутствует, используется текущее время. Если файл не существует, то он создается. Возвращает true при успешном выполнении и false в обратном случае.

Для работы с cookie

- setcookie ("name", $data, $y) — записывает в cookie с именем name переменную $data. Переменная $y определяет, до какого времени будет храниться cookie на компьютере пользователя. Значение переменной $y должно быть определено заранее при помощи функции mktime.

- setcookie ("name") — удаляет cookie с именем name. Обратите внимание, что это та же функция, которая устанавливает cookie, однако для того чтобы удалить cookie, атрибуты $data и $y нужно не указывать.

Для работы с сессиями

- session_destroy — уничтожает все данные сессии. Ее синтаксис: bool session_destroy(void) ; Эта функция не удаляет глобальные переменные или cookie, ассоциированные с сессией. Возвращает true при успешном удалении данных, в противном случае — false.

- session_is_registered — проверяет, зарегистрирована ли переменная в текущей сессии. Синтаксис команды: bool session _is _registered(string name);

- session_regenerate_id — задает новый ID текущей сессии и сохраняет информацию о текущей сессии. Синтаксис: bool session_regenerate_id(void) ; Возвращает true в случае успешного завершения, false — в случае возникновения ошибки.
- session_register — регистрирует одну или несколько переменных в текущей сессии. Синтаксис команды: bool session_register (mixed name); Возвращает true, если все переменные успешно зарегистрированы в сессии.
- session_start — создает сессию или возобновляет текущую, основанную на SID, который вызван запросом при помощи cookie или методов POST и GET. Ее синтаксис: bool session_start (void); Функция всегда возвращает true. Если используются cookie, session_start должна быть вызвана перед тем, как любая переменная будет послана в браузер.
- session_write_close — записывает данные сессии и закрывает ее. Ее синтаксис: void session_write_close (void);
- session_unregister — отменяет регистрацию переменной с именем name в текущей сессии. Ее синтаксис: bool session_unregister (string name); Возвращает true при успешной отмене регистрации.
- session_unset — отменяет регистрацию всех переменных текущей сессии. Ее синтаксис: void session_unset (void)

Для работы с массивами

- array_sum — подсчитывает сумму значений массива. Ее синтаксис: mixed array_sum (array array); Возвращает фиксированное значение, т.е. результат ее деятельности не может быть массивом. Например:

```
$a = array (5,7,10,1) ;
echo array_sum($a);
```

В результате получится 23.

- count — подсчитывает количество элементов в массиве. Ее синтаксис: int count (mixed var [, int mode]) ; Возвращает единицу, если переменная не является массивом. Например:

```
$a[0] = "Hello";
$a[1] = 3;
$a[2] = 5;
$a[3] = "World";
$result = count($time);
echo $result
```

  В результате на экран будет выведено число 4.

- in_array — проверяет наличие указанной переменной в массиве. Ее синтаксис: bool in_array (mixed needle, array haystack[,bool strict]); Возвращает true, если указанное значение needle найдено в массиве с именем haystack, в противном случае — false. Если в третьем параметре bool strict установлен атрибут true, то функция проверяет тип переменной needle. Например:

```
<?
$new = array("Hello", "NTT", "World", "New");
if (in_array ("NTT", $new))
{
print "Значение NTT в массиве $new найдено \n";
}
?>
```

  Функция in_array вернет true, так как искомое значение существует в массиве, а на экране будет выведена строка

  "Значение NTT в массиве array найдено"

- end — устанавливает внутренний указатель массива на последнем элементе. Ее синтаксис: mixed end (array array).

- next — передвигает внутренний указатель массива в сторону увеличения (т.е. вперед) и возвращает следующий элемент массива от текущей позиции внутреннего указателя массива, или false, если элементов больше нет. Если массив содержит пустые элементы, то эта функция возвратит false и для этих элементов. Ее синтаксис: mixed next (array array).

- prev — перемещает внутренний указатель массива в сторону уменьшения индекса (т.е. назад) и возвращает предыдущий элемент массива или false, если перед текущим нет элементов. Если массив содержит пустые элементы, то функция также возвратит false. Ее синтаксис: mixed prev (array array).

- reset — устанавливает внутренний указатель массива на первом элементе. Ее синтаксис: mixed reset (array array);

- current — возвращает элемент массива, на который в данный момент указывает внутренний указатель. Ее синтаксис: mixed current (array array) ; Она не перемещает сам указатель. Если внутренний указатель находится в конце списка элементов, current() возвращает false. Если массив содержит пустые элементы (0 или пустую строку), то функция возвратит false для каждого из них.

- sort — сортирует массив по возрастанию, в том числе по латинскому алфавиту. Ее синтаксис: bool sort (array array)

Например:

```
<?php
$animals = array ("dog", "camel", "cat", "giraffe");
sort($animals);
reset($animals);
while (list($key, $val) = each($animals)) {
echo "fruits[" . $key . "] = " . $val . "\n";
}
?>
```

В результате на экран будет выведено:

```
animals [0] = camel
animals [1] = cat
```

```
animals [2] = dog
animals [3] = giraffe
```

- `rsort` — сортирует массив в обратном порядке (по убыванию). Ее синтаксис: `bool rsort ( array array) ;`

  Например:

```
<?php
$animals = array("dog", "camel", "cat", "giraffe");
rsort($animals);
reset($animals);
while (list($key, $val) = each($animals)) {
echo "$key = $val\n";
}
?>
```

  В результате на экран будет выведено:

```
0 = giraffe
1 = dog
2 = cat
3 = camel
```

Для работы с датой и временем

- `checkdate` — проверяет правильность даты/времени. Ее синтаксис: `bool checkdate (int month, int day, int year);` Возвращает `true`, если указанная дата правильная, иначе — `false`.
- `date` — возвращает время/дату сервера. Ее синтаксис: `$date = date ("параметр") ;` Параметров может быть несколько, разделяются они между собой запятой. Допустимы следующие параметры:
  - a — может принимать значения "am" или "pm";
  - A — "AM" или "PM";

[1] От англ. AM (Ante Meridiem) — до полудня, PM (Post Meridiem) — после полудня. *(Примеч. ред.)*

- d — день месяца, цифровой, две цифры (на первом месте ноль);
- D — день недели, текстовой, три буквы, например "Fri";
- F — месяц, текстовой, длинный, например "January";
- h — час, цифровой, 12-часовой формат, две цифры;
- H — час, цифровой, 24-часовой формат, две цифры;
- i — минуты, цифровой, две цифры;
- j — день месяца, цифровой, без начальных нулей;
- l (строчная L) — день недели, текстовой, длинный, например "Friday";
- m — месяц, цифровой;
- M — месяц, текстовой, три буквы, например "Jan";
- s — секунды, цифровой, две цифры;
- S — английский порядковый суффикс, текстовой, два символа, например "th", "nd"[1];
- U — секунды с начала века Unix, т.е. с 1 января 1970 года;
- Y — год, цифровой, четыре цифры;
- w — день недели, цифровой, "0" означает воскресенье;
- y — год, цифровой, две цифры;
- z — день года, цифровой, например "299".

Обратите внимание, что некоторые параметры имеют различные значения при разном регистре, например d и D.

- mktime — возвращает временную метку Unix, которая является целым числом, равным количеству секунд между точкой отсчета Unix (1 января 1970 года) и указанным временем. Ее синтаксис: `int mktime (int hour, int minute, int second, int month, int day, int year) ;` Например:

[1] Имеются в виду суффиксы порядковых числительных в английском языке, например second (второй), seventh (седьмой). (*Примеч.ред.)*

```
mktime(0,0,0,12,31,1997);
```

В результате получим количество секунд, прошедших с января 1970 года до 0:00 31 декабря 1997 года.

- time — возвращает текущее время, измеренное в секундах с эпохи Unix. Ее синтаксис: `int time (void) ;`

Приложение 4. Cookie

Cookie — небольшие файлы, которые сервер записывает на компьютер пользователя. Все браузеры поддерживают работу с ними, однако в этой технологии больше ограничений, чем возможностей. Например, нельзя записать в один cookie больше 4 Кб данных, нельзя обратиться к чужому cookie, только к созданному своим сервером, нельзя сохранить cookie куда-либо в другое место, кроме как в отведенное пользователем. Нельзя, нельзя, нельзя...

Конечно, все эти ограничения работают в том случае, если посетитель следит за безопасностью собственного компьютера и устанавливает самое современное программное обеспечение.

Очень большое количество сайтов использует в своей работе cookie, а часть из них просто не пустит пользователя к себе без включенной их поддержки в браузере. Есть специальные программы — брандмауэры, отсекающие cookie или отказывающие серверу в доступе, так что выбор все равно остается за посетителем. И все же среднестатистический пользователь не отключает cookie, а для опытных стоит написать предупреждение на сайте.

Чтобы установить cookie с помощью PHP, нужно использовать функцию setcookie. Ее синтаксис:

```
setcookie("имя", "значение");
```

Например, после выполнения

```
setcookie("name", "12345");
```

пока пользователь не закроет окно браузера, переменную $name со значением, равным числу 12 345, можно будет считать с помощью другого оператора:

```
isset ($name);
```

Причем эта переменная станет доступна на любой странице вашего сайта, пока посетитель не уйдет с этого сайта.

Обратите внимание, что в силу определенных свойств работа с cookie должна вестись до какого-либо вывода на экран (в браузер пользователя) любого текста (даже пробел нельзя ставить), в том числе и HTML-тегов. Например, так уже работать не будет:

```
<html>
... работаем с cookie...
```

Только так:

```
... работаем с cookie...
<html>
... любые другие теги и текст...
```

Запомните это ограничение и старайтесь подстраиваться под него, так как тут ничего поделать нельзя. Это не PHP виноват, так уж устроены эти самые cookie.

Можно проверить, установлена ли cookie, и на основании этого принять решение, что делать дальше. Например:

```
if (isset ($name)) { если установлена, то разрешаем вход на другие страницы сайта } else { если нет, предлагаем зарегистрироваться }
```

Надо помнить, что установленная таким образом переменная cookie «живет» только до тех пор, пока не закрыто окно браузера, ее породившее. Как только пользователь закрывает окно, установленная переменная уже не доступна. Это так называемая сессионная cookie.

Впрочем, совсем не сложно продлить срок жизни cookie до нужного времени. Делается это с помощью третьего параметра опе-

ратора setcookie, который указывает дату истечения срока действия cookie либо срок ее действия, если число меньше, чем дата установки:

```
$y = mktime(0,0,0,1,1,2005);

setcookie("name", "bret", $y);
```

Параметр $y указывает на количество секунд, прошедшее с 1 января 1970 года (см. приложение 6). Поэтому перед тем как задать, его нужно сформировать функцией mktime (). Дату нужно задать в такой последовательности: час, минута, секунда, месяц, день и год. На выходе получим нужное значение в секундах, прошедших с 1.01.1970. В примере cookie будет действительна (а значит, и поддастся считыванию) до 1 января 2005 года.

Переустановить или удалить cookie очень просто — достаточно либо указать новое значение, либо задать функцию setcookie с именем этой cookie и остальными пустыми параметрами: setcookie("name") ;

Приложение 5. Методы передачи данных POST и GET

Метод POST основан на передаче данных через форму, а значит, требует перехода либо по кнопке, либо по ссылке в тексте, либо по графическому рисунку, т.е. требуется реальное действие пользователя.

Метод передачи данных GET применяется тогда, когда надо передать информацию программе без активного действия. Вот пример ссылки, передающей данные программе, расположенной в файле index.phtml:

```
http://myhost.ru/index.phtml?temp=1&qwe=slovo
```

Как видите, передаются и цифры и символы, причем передать можно несколько переменных за один раз. В результате обработки такой ссылки программа в index.phtml получит две переменные — $temp и $qwe с соответствующими значениями.

Приложение 6. Время Unix

Давным-давно, когда компьютеры еще были очень большими, а программы очень маленькими, появилась необходимость отсчитывать время. У программистов прижился метод отсчета времени Unix. Начинает свой отсчет это время с 1 января 1970 года. Первая секунда этой даты считается первой, вторая — второй и т.д. На сегодня «набежало» уже достаточно большое число, но нам это не страшно, вручную считать не придется, есть специальные функции.

Итак, время Unix — количество секунд и миллисекунд, прошедших с 1 января 1970 года. В PHP оно определяется при помощи функции mktime. Ее синтаксис:

```
mktime(int hour, int minute, int second, int month,
int day, int year)
```

Например:

```
mktime(0,0,0,12,31,1997);
```

В результате получим количество секунд, прошедших с 1 января 1970 года до 0:00 31 декабря 1997 года.

Производственно-практическое издание

Кухарчик Андрей Леонидович

PHP: обучение на примерах

Ведущий редактор *А.В. Жвалевский*
Редактор *Е.С. Каляева*
Художник обложки *С.В. Ковалевский*
Компьютерная верстка *Д.М. Вербалович*
Корректор *К.А. Степанова*

Подписано в печать с готовых диапозитивов 02.04.2004.
Формат 60x84 $У_{16}$. Бумага газетная. Гарнитура Ньютон.
Печать офсетная. Усл. печ. л. 13,95. Уч.-изд. л. 10,86.
Тираж 3010 экз. Заказ № 2551

Общество с ограниченной ответственностью «Новое знание».
ЛВ № 310 от 01.07.2003. Минск, ул. Академическая, д. 28, к. 112.
Почтовый адрес: 220050, Минск, а/я 79.
Телефон/факс: (10-375-17) 211-50-38. E-mail: nk@wnk.biz

В Москве:
Москва, Колодезный пер., д. 2а.
Телефон (095) 234-58-53. E-mail: ru@wnk.biz

http://wnk.biz

Унитарное полиграфическое предприятие
«Витебская областная типография».
210015, Витебск, ул. Щербакова-Набережная, 4.

PHP: настольная книга программиста

А. Мазуркевич, Д. Еловой

2-еизд., испр.
480 с, с ил.

В удобной наглядной форме описаны все элементы PHP — популярного языка создания CGI-сценариев. Рассмотрены не только особенности синтаксиса языка, но и редактирование кода в программах EditPlus и UltraEdit, а также установка PHP и сервера Apache. Материал систематизирован таким образом, что читатель может использовать книгу и как учебник, и как справочник. Примеры, взятые из реальной практики Web-программирования, позволяют лучше усвоить теоретический материал.

Книга рассчитана на самую широкую аудиторию — не только на новичков, но и на опытных программистов.

ISBN985-475-076-0

ИЗДАТЕЛЬСТВО "НОВОЕ ЗНАНИЕ"

Flash MX 2004 и ActionScript 2.0: обучение на примерах

Д.А. Гурский, Ю.А. Гурский

446 *с., с ил.*

Первое практическое руководство по работе во Flash MX 2004 и созданию сценариев на языке ActionScript 2.0 состоит из десяти уроков, подобранных по возрастанию сложности. Понятные, наглядные примеры позволят быстро освоить программу и научиться создавать эффектные баннеры, несложную анимацию и онлайн-игры. Изложение сопровождается фрагментами кода на ActionScript 2.0 и всеми необходимыми иллюстрациями.

Книга адресована всем, кто начинает знакомиться с технологией Flash и нуждается в простом и удобном руководстве.

ISBN5-94735-035-1

Наши координаты:
в Москве: (095) 234-58-53, e-mail: ru@wnk.biz
в Минске: (10-375-17) 211-50-38, e-mail: nk@wnk.biz

http://wnk.biz

Хотите сделать свой сайт динамичным, привлекательным, функциональным?

Изучите книгу, которую держите в руках.
Вы научитесь создавать:

- **Счетчики различного типа**
- **Чаты**
- **Гостевые книги**
- **Службу рассылки**

Вы узнаете,
что PHP-программирование — это просто!

Что прочитать после этой книги?
А. Мазуркевич, Д. Еловой
"PHP: Настольная книга программиста"

ISBN 985-475-050-7
9 789854 750507