Букварь по PHP и MySQL

Александр Качанов
(kachanov@ogs.gomel.by)

Вадим Ткаченко
(vvtk@stealthcomp.com)
(http://web.stealthcomp.com)

Андрей Головин
(mine@convex.ru)
(http://exper.ural.ru)

Версия 1.1 (от 16.05.2000)

Содержание

О данном букваре

Этот документ создан на основе компиляции нескольких документов, авторами которых являются перечисленные на титульной старнице люди, а также на основе перевода с английского страниц-учебников, которые в обилии можно встретить на многочисленных Web-узлах в Интернете, посвященных программированию на PHP и связи его с MySQL.

Данный труд не ставит целью заменить собой мануалы по PHP и MySQL. Дело в том, что большинство материалов, в том числе и руководства к данным программным продуктам пока распространяются на английском языке. Качественные пособия на русском языке встречаются редко.

Данный букварь просто поможет вам сделать первые шаги в изучении PHP и способах взаимодействия его с БД MySQL. Буду рад, если в этой книге вы найдете ответы на свои основные вопросы.

Труд еще не закончен и будет пополнятся и улучшаться.

Источники информации

Основные источники – это руководство по PHP (www.php.net) и MySQL (www.mysql.com).

Главными вашими настольными книгами должны стать:

- PHP3 Manual

- http://www.php.net/manual/ - HTML-версия в Интернете с аннотациями-комментариями пользователей по каждому разделу руководства
- http://www.php.net/distributions/manual.zip - Просто руководство только запакованное для выкачки и локального просмотра
- http://www.php.net/distributions/manual_a-l.pdf , http://www.php.net/distributions/manual_m-x.pdf - PDF-версии руководства (разбит на 2 части)
- http://www.php.net/docs.php3 - остальные варианты руководства
- http://www.webclub.ru/materials/php/ - перевод на русский руководства по PHP2 (устарело, но вполне годится)

- MySQL manual
  - http://www.mysql.com/doc.html - полная россыпь документации по MySQL, а также инструкции. Все на английском языке.

Также много материалов (опять-таки на англйиском) можно найти на http://www.webreview.com и http://www.devshed.com.

Некоторые разделы букваря являются переводами из этих источников.

Что такое PHP?

PHP – это язык серверных скриптов (server scripting language), встраиваемый в HTML, который интерпретируется и выполняется на сервере.

PHP является препроцессором HTML. Т.е. его работа построена по следующей схеме:

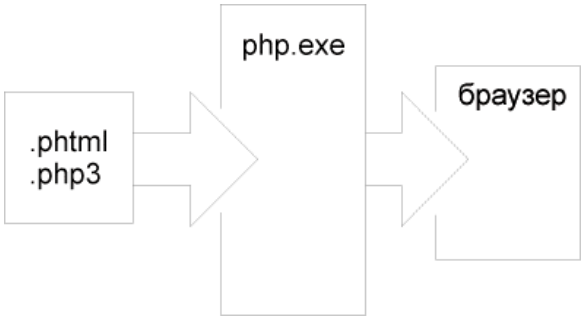

Рис. 1

До того, как сервер "отдаст" файл броузеру, его просматривает препроцессор-интерпретатор. Для того, тобы это происходило, файлы, которые подвергаются обработке препроцессором, должны иметь определенное расширение (обычно это .phtml или .php3, но эти значения можно поменять) и содержать (хотя это не обязательное требование) код для препроцессора. Перед отправкой страницы PHP-код проигрывается на сервере и брозеру выдается результат в виде опять таки HTML-страницы, которая может сильно отличаться от той, что хранится на сервере. Обычные же страницы, имеющие расширение .html/.htm Web-сервер будет отправлять броузеру без какой-либо обработки.

Основное отличие от CGI-скриптов, написанных на других языках, типа Perl или C – это то, что в CGI-программах вы сами пишете выводимый HTML-код, а, используя PHP – вы встраиваете свою программу-скрипт в готовую HTML-страницу, используя открывающий и закрывающий теги (в примере <?php и ?>).

PHP называется языком *серверных скриптов* в отличие от JavaScript/Jscript/VBScript, которые являются языками *клиентских скриптов*. Это значит, что PHP-скрипт выполняется на сервере, а клиенту передается результат его работы, тогда как в JavaScript-код полностью передается на клиентскую машину и только там выполняется броузером.

Любители MS Internet Information Server найдут, что PHP очень похож на Active Server Pages (ASP), а энтузиасты Java скажут, что PHP похож на Java Server Pages (JSP). Еще некоторыми аналогами PHP являются языки ColdFusion (www.allaire.com) и embPerl. Все эти языки позволяют размещать код, выполняемый на Web-сервере, внутри HTML-страниц. При желании и умении программировать, вы могли бы и сами написать такой препроцессор, который бы позволял вставлять в Web-страницы код, скажем на языке Фокал или Алгол.

Проще всего работу PHP показать на примере. Так выглядит web-старница с элементами php:

```
<html>
<head>
<title>Пример</title>
</head>
<body>
<?php echo "Привет, я PHP-программа!"; ?>
</body>
</html>
```

Рис. 2

После выполнения этого скрипта мы получим страничку, в которой будет написано

Привет, я PHP-программа!

Рис. 3

Открыв исходный текст данной страницы мы увидим следующее.

```
<html>
<head>
<title>Example</title>
</head>
<body>
Привет, я PHP-программа!
</body>
</html>
```

Рис. 4

Как видите, в результирующей странице нет и следа PHP-кода. Казалось бы, весьма просто и бесполезно, но PHP позволяет делать и более сложные и фантастические вещи, о некоторых из них будет рассказо ниже.

Возможности PHP

В нескольких словах – на PHP можно сделать все, что можно сделать с помощью CGI–программ. Например: обрабатывать данные из форм, генерировать динамические страницы, получать и посылать куки (cookies).

Кроме этого в PHP включена поддержка многих баз данных (databases), что делает написание Web-приложений с использованием БД до невозможности простым.

Вот неполный перечень поддерживаемых БД:

Adabas D	InterBase	Solid
dBase	mSQL	Sybase
Empress	MySQL	Velocis
FilePro	Oracle	Unix dbm
Informix	PostgreSQL	ODBC

Вдобавок ко всему PHP понимает протоколы IMAP, SNMP, NNTP, POP3 и даже HTTP, а также имеет возможность работать с сокетами (sockets) и общаться по другим протоколам.

Краткая история PHP

Началом PHP можно считать осень 1994 года, когда Rasmus Lerdorf решил расширить возможности своей домашней страницы и написать небольшой движок для выполнения простейших задач. Такой движок был

готов к началу 1995 года и назывался Personal Home Page Tools (отсюда и сокращение PHP). Умел он не очень много – понимал простейший язык и всего несколько макросов.

К середине 1995 года появилась вторая версия, которая называлась PHP/FI Version 2. Приставка FI – присоединилась из другого пакета Rasmusa, который умел обрабатывать формы (Form Interpritator). PHP/FI компилировался внутрь Apache и использовал стандартный API Apache. PHP скрипты оказались быстрее аналогичных CGI – скриптов, так как серверу не было необходимости порождать новый процесс. Язык PHP по возможностям приблизился к Perl, самому популярному языку для написания CGI-программ. Была добавлена поддержка множества известных баз данных (например, MySQL и Oracle). Интерфейс к GD – библиотеке, позволял генерировать gif-картинки на лету. С этого момента началось широкое распространение PHP/FI.

В конце 1997 Zeev Suraski и Andi Gutmans решили переписать внутренний движок, с целью исправить ошибки интерпретатора и повысить скорость выполнения скриптов. Через полгода, 6 июня 1998 года вышла новая версия, которая была названа PHP 3.

К лету 1999 года PHP 3 был включен в несколько коммерческих продуктов. По данным NetCraft на ноябрь 1999 PHP использовался в более чем 1 млн. доменах.

На сегодняшний день (май 2000) готовится к выпуску новая версия PHP 4, в которой внутренний движок будет снова переписан (он имеет название Zend - www.zend.com). Предполагается, что производительность новой версии будет в десятки раз выше, чем у существующей. Уже вышли в свет предварительные тестовые версии RC1 и RC2 этого продукта. В скором времени, наверное, выйдет уже и официальная версия.

Почему нужно выбирать PHP?

Разработчикам Web-приложений нет необходимости говорить, что web-страницы - это не только текст и картинки. Достойный внимания сайт должен поддерживать некоторый уровень интерактивности с пользователем: поиск информации, продажа продуктов, конференции и т.п. До недавних пор все это традиционно реализовалось CGI-скриптами, написанными на Perl. Но оказалось, что CGI-скрипты очень плохо масштабируемы. Каждый новый вызов CGI-скрипта, требует от ядра порождения нового процесса, а это занимает процессорное время и тратит оперативную память. PHP предлагает другой вариант – он работает как часть Web-сервера, и этим самым похож на ASP от Microsoft или ColdFusion от Allaire.

Синтаксис PHP очень похож на синтаксис C или Perl. Люди, знакомые с программированием, очень быстро смогут начать писать программы на PHP. В этом языке нет строгой типизации данных и нет необходимости в действиях по выделению/освобождению памяти.

Программы, написанные на PHP, читаются достаточно легко. В отличие от Perl-программ PHP-код легко зрительно прочитать и понять.

В дополнение к своей бесплатности (хотя MySQL требует приобретения лицензии при использовании ее в коммерческих целях) связка PHP-MySQL является кросс-платформенной. Это значит, что вы можете, работая в Windows, разрабатывать приложения, предназначенные для работы под Unix. Кроме того, PHP может работать как внешний CGI-процесс, либо как обычный интерпретатор скриптов, либо как модуль, подключаемый к web-севреру Apache или IIS.

И наконец, так как данный продукт разрабатывается совместными усилиями, существует огромное количество документации и списков рассылки, к которым можно обратиться в случае возникновения каких-либо вопросов. Найденные ошибки исправляются достаточно быстро, ваши предложения и замечания всегда выслушают, рассмотрят, и если они окажутся ценными – реализуют в новой версии.

Недостатки PHP

1. Основным недостатком PHP 3, есть то, что по свой идеологии PHP изначально был ориентирован на написании небольших скриптов. Несмотря на то, что движок несколько раз переписывался, PHP 3 не пригоден для использования в сложных проектах – при обработке больших скриптов производительность системы резко падает (предчувствуя возмущение сторонников PHP 3, я скажу, что наличие такого недостатка подтверждает и сам разработчик Zeev Suraski). Однако этот недостаток будет ликвидирован в движке PHP 4, который, по словам того же разработчика, предназначен для работы в больших проектах.
2. PHP является интерпретируемым языком, и, вследствие этого, не может сравниться по скорости с компилируемым C. Однако при написании небольших программ, что, в общем-то, присуще проектам на PHP, когда весь проект состоит из многих небольших страниц с кодом, вступают в силу накладные расходы на загрузку в память и вызов CGI-программы, написанной на C.

3. Не такая большая база готовых модулей, как, например, CPAN у Perl. С этим ничего нельзя поделать – это дело времени. В PHP 4 разработчики предусмотрели специальный архив, аналогичный CPAN, и я думаю, очень скоро будет написано достаточное количество модулей для его наполнения.
4. Нет поддержки сессий (session), как, например, в ASP. В PHP 4 этот недостаток будет устранен.

Где взять PHP?

Главным сайтом, откуда распространяется PHP, является www.php3.net. На нем вы сможете найти версии PHP как для UNIX-платформы так и под платформу Win32.

Сайт разработчиков 4-й версии PHP под кодовым названием Zend расположен по адресу www.zend.com.

Как установить PHP?

Что такое MySQL?

MySQL – небольшой, компактный многопоточный сервер баз данных. MySQL характеризуется большой скоростью, устойчивостью и легкостью в использовании.

MySQL был разработан компанией TcX для внутренних нужд, которые заключались в быстрой обработке очень больших баз данных. Компания утверждает, что использует **MySQL** с 1996 года на сервере с более чем 40 БД, которые содержат 10,000 таблиц, из которых более чем 500 имеют более 7 миллионов строк.

MySQL является идеальным решением для малых и средних приложений. Исходные тексты сервера компилируются на множестве платформ. Наиболее полно возможности сервера проявляются на Unix-серверах, где есть поддержка многопоточности, что дает значительный прирост производительности. В варианте под Windows, MySQL может запускаться как сервис Windows NT или как обычный процесс на Windows 95/98.

На текущий момент MySQL все еще в стадии разработки, хотя версии 3.22 полностью работоспособны.

MySQL-сервер является бесплатным для некоммерческого использования. Иначе необходимо приобретение лицензии, стоимость которой составляет 190 EUR.

Возможности MySQL

MySQL поддерживает язык запросов SQL в стандарте ANSI 92, и кроме этого имеет множество расширений к этому стандарту, которых нет ни в одной другой СУБД.

Краткий перечень возможностей MySQL.

1. Поддерживается неограниченное количество пользователей, одновременно работающих с базой данных.
2. Количество строк в таблицах может достигать 50 млн.
3. Быстрое выполнение команд. Возможно MySQL самый быстрый сервер из существующих.
4. Простая и эффективная система безопасности.

MySQL действительно очень быстрый сервер, но для достижения этого разработчикам пришлось пожертвовать некоторыми требованиями к реляционным СУБД. В MySQL отсутствуют:

1. Поддержка вложенных запросов, типа SELECT * FROM table1 WHERE id IN (SELECT id FROM table2). Утверждается, что такая возможность будет в версии 3.23.
2. Не реализована поддержка транзакций. Взамен предлагается использовать LOCK/UNLOCK TABLE.
3. Нет поддержки внешних (foreign) ключей.
4. Нет поддержки триггеров и хранимых процедур.
5. Нет поддержки представлений (VIEW). В версии 3.23 планируется возможность создавать представления.

По словам создателей именно пункты 2-4 дали возможность достичь высокого быстродействия. Их реализация существенно снижает скорость сервера. Эти возможности не являются критичными при создании Web-приложений, что в сочетании с высоким быстродействием и малой ценой позволило серверу приобрести большую популярность.

Использование PHP

Ваша первая PHP-страница

Рад вам сообщить, что самое трудное у вас позади. Установка программы – это всегда сложный процесс, так как ни одна система не похожа на другую, и в каждом случае могут возникнуть особенные проблемы, с которыми никогда никто не сталкивался. Во всяком случае ваша база данных уже установлена и запущена, PHP-движок откомпилирован, установлен и связан с Web-сервером, который уже понимает, что ему нужно делать с документами, имеющими расширение .php3.

Окунемся же с головой в написание нашей первой PHP-страницы. Создайте текстовый файл под именем test.php3 и напишите в нем следующее:

```
<html>
<body>
<?php
$myvar = "Hello, World";
echo $myvar;
?>
</body>
</html>
```

Рис. 5

Теперь откройте броузер и наберите в нем URL созданной страницы, например: http://stage/test.php3. На экране в броузере вы должны увидеть следующее:

Hello, World

Рис. 6

Если вместо этого вы видите сообщения об ошибках, обратитесь в первым главам данной книги, а, если потребуется, то и к документации, и проверьте, все ли вы правильно сделали.

Открыв исходный текст страницы вы увидите в ней следующее:

```
<html>
<body>
Hello, World
</body>
</html>
```

Рис. 7

Это произошло потому, что PHP-движок на сервере просмотрел страницу, нашел в ней PHP-код, обработал его и выдал результат, который Web-сервером был отправлен в ваш броузер.

Поздравляем, вы написали первую в своей жизни PHP-страницу! Это уже не статическая HTML-страница с фиксированным текстом. Это уже настоящая программа, которая в зависимости от поданных в нее данных может выдавать различные результаты. А эта возможность целиком меняет всю философию публикации документов в Интернете, превращая их из статических в динамические, меняющиеся в зависимости от действий

пользователя. В нашей первой странице-программе эти возможности конечно не так уж и сильно видны, ведь единственное, что она делает – это выводит текст, который мы быстрее и проще написали бы руками. Тем не менее уже через несколько страниц вы поймете насколько гибкий и мощный инструмент находится у вас в руках. А пока, рассмотрим подробнее нашу первую PHP-страницу.

Первое, на что надо обратить внимание в вышеприведенном коде, это ограничители. Найдите строку, которая начинается с **<?php**. Для PHP-движка этот код означает начало блока команд, которые надо обработать и выполнить. Заканчивается блок ограничителем **?>**. Иными словами символы **<?php** и **?>** выполняют роль скобок. Все, что находится вне их, PHP-движок пропускает и отправляет в Web без всякой обработки, выполняя лишь только то, что находится внутри этих "скобок". Мощь PHP заключается в том, что PHP-код можно вставлять в любое – я подчеркиваю – в любое место HTML-страницы. Несколько позже мы рассмотрим некоторые из весьма интересных приемов, а сейчас давайте опустим подробности. Вместо скобок **<?php** … **?>** можно использовать и сокращенную нотацию **<?** … **?>**.

Некоторым программистам, которые работают также с ASP, удобнее писать скобки используя комбинацию **<%** … **%>**. PHP можно настроить на использование и таких скобок, если вам лень перестраиваться.

Возможен еще один из вариантов скобок показан ниже:

```
<SCRIPT LANGUAGE="PHP">
инструкции
<SCRIPT>
```

Рис. 8

Еще одна деталь, на которую вы обратите внимание, - это точка с запятой в конце каждой строки кода. Это так называемые "разделители", которые служат для отделения одного набора команд от другого. Вообще-то весь PHP-код можно писать в одной строке, разделяя команды точкой с запятой. Но читать такой код будет неудобно, поэтому в наших примерах после каждой точки с запятой мы ставили перевод строки, а также еще один перевод строки, чтобы яснее выделить группы команд. Не забывайте про точку с запятой в конце строки, это наиболее частая ошибка у начинающих программистов.

Наконец, вы заметили, что перед словом myvar стоит символ $ (доллар). Этот символ сообщает PHP, что перед ним переменная. Мы присвоили (используя символ "=") строку "Hello, World" переменной $myvar. Переменные помимо строк могут содержать числа и массивы. В любом случае любая переменная всегда обозначается символом $.

Истинная сила языка PHP содержится в его функциях. В теории, функция – это блок команд, который выполняет какую-то операцию. Если скомпилировать PHP со всеми имеющимися для него дополнениями, вы получите доступ к более чем 700 функциям. Так что, свои собственные функции вам придется писать разве что в исключительном случае.

Каждая функция имеет свое название и синтаксис. В первом нашем примере была использована функция **echo**, которая, как вы догадались, выводит строку, заключенную в кавычки, или переменную, которая идет следом.

Давайте еще раз внимательно рассмотрим исходную PHP-страницу. В принципе, она ни чем не отличается от обычной HTML-страницы. Только вместо расширения .html (или .htm) мы ей присвоили расширение .php3. Для Web-сервера это расширение послужило сигналом, что данную страницу перед отправкой надо пропустить через PHP-движок. Строки <html><body> …. </body></html> будут проигнорированы PHP-движком. Он обратит внимание только на то, что написано внутри скобок **<?php** … **?>**. В результате мы получим то, что изображено на рис. 2. В принципе, всю HTML-страницу мы могли бы сгенерировать с помощью PHP-команд, например:

```
<?php
echo "<html>";
echo "<body>";
$myvar = "Hello World";
echo $myvar;
echo "</body>";
echo "</html>";
```

```
?>
```

Рис. 9

Результат был бы тот же. Но с точки зрения программирования вообще, этот код - некачественный. Ведь PHP-движку придется обработать уже шесть строчек кода вместо прежних двух. Зачем утруждать PHP-движок выводом тегов <html>, <body>, </body>, </html>, если они и так выводятся в странице? При написании кода никогда не забывайте о производительности. Всегда старайтесь улучшить или изменить код так, чтобы у PHP-движка уходило как можно меньше времени на его обработку. Некоторые советы по оптимизации кода мы приведем несколько позднее.

Вся подноготная PHP

Одной из очень полезных функций языка PHP является **phpinfo()**. Она выводит на экран всю информацию о PHP-движке, установленном на сервере, причем в удобной форме таблицы. Создайте страницу со следующим кодом и вызовите ее из броузера:

```
<html>
<body>
<?php
phpinfo();
?>
</body>
</html>
```

Рис. 10

В связи с этим примером хочется вспомнить знаменитую фразу из старого фильма: "Всего одна строка, а как много она нам говорит". Перед нами страница, наполненная различными интересными и полезными сведениями не только о самом PHP-движке, но и о Web-сервере, его расширениях и возможностях. Найдете строки, начинающиеся на MySQL. Они содержат информацию о MySQL, работающем на сервере. Если эти строки отсутствуют, значит по каким-то причинам PHP не поддерживает работу с MySQL. Обратитесь в первым главам данной книги, а, если потребуется, то и к документации, и проверьте, все ли вы правильно сделали.

Вывод текста в HTML-страницу

Самый простейший способ общения с пользователем через Web-страницу, это послать ему в странице какой-нибудь текст. Это можно сделать двумя способами: с помощью функции **print** или **echo**:

```
<?php
print "Hello, world.";
?>
```

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

Рис. 11

Print это функция, которая отправляет броузеру текст. Между словом "print" и символом ";" мы помещаем строку, которая обозначается кавычками. Все, что находится внутри кавычек, будет отправлено броузеру. Зная это, мы теперь можем отправить текст "Hello, World" несколькими способами. Создайте файл print.php3 со следующим текстом:

```
<html>
<body>
<?php
print "Здесь используется функция print.";
print "<p>";
echo "А здесь использована функция echo.",
" ",
"P.S. Можно добавить вторую текстовую строку.",
" ",
"Текстовые строки разделяются запятыми.";
print "<p>";
printf ("Здесь используется функция printf.");
print "<p>";
printf ("Функция printf в основном используется для форматированного вывода чисел и строк.");
print "<p>";
printf ("Не забывайте о скобках, когда пользуетесь функцией printf.");
?>
</html>
</body>
```

Рис. 12

В результате выполнения кода в броузере мы получим следующий результат:

Здесь используется функция print.

А здесь использована функция echo. P.S. Можно добавить вторую текстовую строку. Текстовые строки разделяются запятыми.

Здесь используется функция printf.

Функция printf в основном используется для форматированного вывода чисел и строк.

Не забывайте о скобках, когда пользуетесь функцией printf.

Рис. 13

Функция **print** – самый простейший способ отправки текста в броузер.

Функция **echo** работает так же, как и **print**, однако позволяет добавлять к первой текстовой строке, другие строки, разделяя их запятыми.

Функция **printf** отображает числа в определенном формате, например, выводит дробное число с определенным количеством нулей после запятой, поэтому в функции **printf** использование скобок обязательно.

При работе со скобками пользуйтесь следующими тремя правилами:

- **echo** со никогда не используется со скобками
- **printf** всегда используется со скобками

- **print** используется и так и так

Работа с формами

В этом примере показано, как в PHP легко обрабатывать данные, полученные от HTML–форм. Для понимания этой главы от вас требуются крепкие знания языка HTML и принципа работы HTML-форм и а также понимания разницы двух методов передачи данных в них (GET И POST).

Чаще всего серверные скрипты используются для обработки результатов заполнения форм. Например, в гостевой книге посетитель вводит данные в форму, которая затем обрабатывается на сервере. Отвечая на какой-либо опрос пользователь, аналогично, устанавливает значение определенных полей формы.

Напомню, какие тэги и атрибуты должна содержать форма:

```
<FORM NAME="имя_формы"
       ACTION="путь_к_обработчику"
       METHOD="метод_передачи_переменных">
поля ввода...
</FORM>
```

Цветом выделены те элементы, которые пригодятся нам в этом опыте. Прежде всего разберемся, что такое "обработчик". Это скрипт на сервере, в который будут переданы значения полей ввода.

Каждое поле ввода имеет атрибут NAME, которое будет передано в обработчик вместе со своим значением. Существует два метода передачи данных: GET и POST. Их отличие состоит в том, что при использовании метода GET значения полей присоединяются к URL, указанному в атрибуте ACTION. Происходит это таким образом:

http://site.domain/action.php3**?имя=значение&**...**имя=значение**

Пары "имя=значение" создаются для каждого элемента ввода, для которого указано имя атрибутом NAME.

В случае использования метода POST значения полей передаются в заголовке запроса к серверу. Формат передачи при этом методе нам, в общем-то, не интересен. Просто примем к сведению, что значения передаются "незаметно" для обычного пользователя.

При исполнении скрипта на языке PHP создаются переменные с именами, соответствующими именам полей и содержащие соответствующие значения.

Предположим, что мы создали форму следующего вида:

```
<FORM ACTION="mult.php3" METHOD="GET">
<INPUT TYPE="text" NAME="first" SIZE="4" MAXLENGTH="4">
<INPUT TYPE="text" NAME="second" SIZE="4" MAXLENGTH="4">
<INPUT TYPE="Submit" VALUE="Умножить">
</FORM>
```

Скрипт, содержащийся в файле mult.php3 может выглядеть следующим образом:

```
<?php
   Header("Content-type: text/html");
   echo "$first умножить на $second получится ", $first*$second;
?>
```

Как видим, все довольно просто.

Необходимо напомнить, что существует специальный тип поля HIDDEN. Это поле, которое не выводится на экран, но, если ему присвоено имя атрибутом NAME, значение его передается в форму. Это бывает полезно, например, когда один обработчик может производить не одно, а несколько действий. С помощью такого поля мы можем задать тип действия, которое мы хотим произвести с данными формы.

В следующей таблице перечислены все возможные элементы ввода, которые используются в формах.

Тип	Описание	Вид
TEXT	Поле ввода текста	
SELECT	Выбор из списка.	
RADIO	Радио-кнопка. Используется для выбора одного из предложенных вариантов.	
CHECKBOX	Кнопка-флажок. Используется для выбора варианта.	
SUBMIT	Кнопка, которая инициирует вызов обработчика формы.	
IMAGE	Изображение. Используется как кнопка типа SUBMIT	
<TEXTAREA>	Область ввода текста.	

Теперь рассмотрим, как значения и состояния этих элементов передаются в обработчик.

TEXT - здесь все просто. Введенное значение передается в виде: *имя=значение* (для удобства будем предполагать, что метод передачи значений установлен в GET). В обработчике значение можно получить из переменной $*имя*.

SELECT - значение берется из атрибута VALUE выбранного элемента <OPTION>. Например для <SELECT> такого вида:

```
<SELECT NAME="mySelect">
   <OPTION VALUE="test1">test1</OPTION>
   <OPTION VALUE="test2">test2</OPTION>
   <OPTION VALUE="test3">test3</OPTION>
</SELECT>
```

Строка будет содержать mySelect=test1, в случае выбора первого элемента списка. Переменная в скрипте будет выглядеть так: $mySelect.

Элемент <SELECT> может иметь атрибут MULTIPLE, что позволяет выбирать несколько значений из списка. В этом случае к имени элемента <SELECT> необходимо добавить пару квадратных скобок: *имя*[]. Строка будет выглядеть так: *имя[]=значение&имя[]=значение*..., а в скрипте доступ к выбранным значениям можно осуществить, как к элементам массива $*имя*.

В случае, если не заданы атрибуты VALUE, то передаваться будет то, что содержится между тэгами <OPTION> и </OPTION>.

RADIO - Здесь значение будет браться из атрибута VALUE, строка выглядит аналогично элементу типа TEXT. Доступ из скрипта, тоже аналогичен. Если вы забыли установить это значение, то будет передано значение on

CHECKBOX - если флажок установлен, то передается значение on, если флажок не установлен, то переменная не передается вообще. Таким образом, установку флажка в скрипте можно проверить, сравнив значение переменной $*имя* с "on". Переменная и строка выглядят аналогично элементу типа TEXT.

SUBMIT - кнопка SUBMIT, как ни странно, тоже может передавать значение в обработчик. Я не могу себе представить зачем это нужно, но тем не менее. Значение устанавливается из атрибута VALUE. Все остальное аналогично полю типа TEXT.

IMAGE - Самый интересный элемент. В обработчик передаются два значения: *имя*.x и *имя*.y, которые представляют собой координату указателя мыши относительно верхнего левого угла изображения. Строка выглядит следующим образом: *имя*.x=*значение*&*имя*.y=*значение*. В скрипте устанавливаются переменные $*имя*_x и $*имя*_y.

<TEXTAREA> - абсолютно аналогично элементу типа TEXT.

При пересылки строковых значений они перекодируются специальным образом. Все символы, кроме алфавитно-цифровых и знака подчеркивания "_" заменяются знаком процента "%" и двумя

шестнадцатеричными цифрами кода. Пробелы заменяются на знак "+". При установке переменных в скрипте производится обратное декодирование.

PHP предоставляет еще одну интересную особенность. Мы можем каждому элементу присвоить имя переменной массива. Например:

```
<FORM NAME="testForm" ACTION="test.php3">
name: <INPUT TYPE="text" NAME="personal[name]"><BR>
e-mail: <INPUT TYPE="text" NAME="personal[email]"><BR>
<INPUT TYPE="SUBMIT">
</FORM>
```

В этом случае мы сможем получить доступ к значениям, обращаясь к элементам ассоциативного массива: $personal["name"] и $personal["email"].

Кроме того, если включена директива PHP <?php_track_vars?>, то, при передаче значений, будут заполнены массивы $HTTP_GET_VARS и $HTTP_POST_VARS, для соответствующих методов передачи переменных в обработчик

Вызов формы самой на себя.

Работа с MySQL (вывод данных из базы данных)

Для понимания этой главы от вас требуются крепкие знания языка SQL и принципов работы баз данных. Для начала создаем базу данных и таблицу. Входим в командную строку MySQL, и выполняем команды:

```
mysql > CREATE DATABASE mydb;
mysql> CREATE TABLE employees
       ( id tinyint(4) DEFAULT '0' NOT NULL AUTO_INCREMENT,
       first varchar(20), last varchar(20), address varchar(255),
       position varchar(50), PRIMARY KEY (id), UNIQUE id (id));
       INSERT INTO employees VALUES
       (1,'Bob','Smith','128 Here St, Cityname','Marketing Manager');
       INSERT INTO employees VALUES
       (2,'John','Roberts','45 There St ,Townville','Telephonist');
       INSERT INTO employees VALUES
       (3,'Brad','Johnson','1/34 Nowhere Blvd, Snowston','Doorman');
```

Рис. 14

В результате у нас будет создана база данных mydb. В ней будет создана таблица employees (работники). И в эту таблицу будут вставлены три записи с данными о работниках. Для экспериментов с PHP и MySQL этого вполне достаточно.

Давайте выведем эти данные из базы данных в HTML-страницу. Для общения с MySQL из PHP понадобятся следующие функции.

int mysql_connect(string hostname, string username, string password);

Создать соединение с MySQL.

Параметры:

Hostname – имя хоста, на котором находится база данных.

Username – имя пользователя.

Password – пароль пользователя.

Функция возвращает параметр типа int, который больше 0, если соединение прошло успешно, и равен 0 в противном случае.

int mysql_select_db(string database_name, int link_identifier);

Выбрать базу данных для работы.

Параметры:

Database_name – имя базы данных.

link_identifier – ID соединения, которое получено в функции mysql_connect. (параметр необязательный, если он не указывается, то используется ID от последнего вызова mysql_connect)

Функция возвращает значение true или false

int mysql_query(string query, int link_identifier);

Функция выполняет запрос к базе данных.

Параметры:

query – строка, содержащая запрос

link_identifier – см. предыдущую функцию.

Функция возвращает ID результата или 0, если произошла ошибка.

int mysql_result(int result, int i, column);

Функция возвращает значение поля в столбце column и в строке i.

int mysql_close(int link_identifier);

Функция закрывает соединение с MySQL.

Параметры:

link_identifier – см. выше.

Функция возвращает значение true или false

Создайте файл с расширением .php3 и наберите в нем следующий текст:

```
<html>
<body>
<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
$result = mysql_query("SELECT * FROM employees",$db);
printf("First Name: %s<br>\n", mysql_result($result,0,"first"));
printf("Last Name: %s<br>\n", mysql_result($result,0,"last"));
printf("Address: %s<br>\n", mysql_result($result,0,"address"));
printf("Position: %s<br>\n", mysql_result($result,0,"position"));
mysql_close($db);
?>
</body>
```

```
</html>
```

Рис. 15

Теперь рассмотрим построчно, что происходит в этой программе. Функция **mysql_connect()** открывает связь с сервером баз данных MySQL. В качестве параметров мы указываем ей имя узла (*localhost*), на котором находится база данных, имя пользователя (root), под которым мы будем с ней работать, и пароль (*в данном случае он пустой и потому не указывается*).

Имя узла localhost означает, что сервер MySQL находится на той же машине, что и сам Web-сервер с PHP-движком. В принципе ничто не мешает вам (имея права) обратиться к серверу MySQL, который находится на соседней машине или вообще на другом конце земного шара. Такие эксперименты авторы данной книги уже проводили и весьма успешно. О способах построения приложений с использованием нескольких машин мы поговорим немного позже.

В результате выполнения этой функции получаем некое значение, которое присваиваем переменной $db. Эта переменная называется идентификатором соединения (см. выше описание синтаксиса).

Соединившись с сервером выбираем базу данных, с которой будем работать (ведь на одном и том же сервере могут "крутиться" несколько баз данных). Это делается с помощью функции **mysql_select_db()**. В качестве параметров мы передаем название нужной нам базы данных и идентификатор соединения, полученный нами при выполнении предыдущей команды.

В результате выполнения функции **mysql_select_db()** мы получаем значение *true* или *false*. Если соединение с базой данных произошло успешно – *true*, если нет – *false*. Для того, чтобы наша программа-страница работала более интеллектуально, мы можем при желании проанализировать возвращаемое значение и если оно будет *false*, вывести хорошее информативное сообщение об ошибке. Как это делается, мы рассмотрим в других, более сложных примерах.

И наконец мы обращаемся к базе данных с запросом, написанным на языке SQL. Для этого служит функция **mysql_query()**. В качестве первого параметра мы передаем текст запроса, а в качестве второго – "скармливаем" идентификатор, полученный от выполнения функции **mysql_connect()**.

Результаты выполнения функции **mysql_query()** – записи, удовлетворяющие нашему запросу - помещаем в переменную $result.

И наконец, с помощью функции **mysql_result()** извлекаем из результатов выполнения нашего запроса (т.е. переменной $result), первый ряд-запись (который имеет порядковый номер 0), и значение определенного поля (по его имени). Перебирая друг за другом записи от 0 до 2, мы выберем все записи, что хранятся в нашей маленькой базе данных.

Привести иллюстрацию исполнения данного кода

В данном коде в полной мере используется функция **printf()**, с которой мы знакомились в предыдущей главе. Для тех, кто когда-либо работал с языком Perl или C, строки с функцией printf покажутся весьма знакомыми. Если говорить коротко, то в каждой приведенной выше строке комбинация символов "%s" обозначает, что вместо нее должна быть поставлена переменная, идущая во второй части выражения printf. Причем эта переменная должна быть переведена в тип "строковая переменная". Более подробное описание синтаксиса функции **printf()** читайте в руководстве по языку.

Вот так легко можно работать с базой данных в PHP.

В следующей главе мы познакомимся с более интеллектуальными, чем **mysql_result(),** функциями выборки данных из БД **mysql_fetch_row()** и **mysql_fetch_array()**, В дальнейшем мы рекомендуем пользоваться именно ими, как более быстрыми и удобными, чем **mysql_result()**.

Работа с MySQL (вывод данных из базы данных). Часть II

Давайте теперь попробуем вывести все записи, хранящиеся в нашей базе данных. Обратимся к базе данных со следующим кодом:

```
<html>
<body>
<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
```

```
$result = mysql_query("SELECT * FROM employees", $db);
echo "<table border=1>\n";
echo "<tr><td>Name</td><td>Position</tr>\n";
while ($myrow = mysql_fetch_row($result))
       {
       printf("<tr><td>%s %s</td><td>%s</td></tr>\n", $myrow[1], $myrow[2], $myrow[3]);
       }
echo "</table>\n";
?>
</body>
</html>
```

Рис. 16

Вы вероятно заметили, что в данном коде вы ввели несколько новых функций и конструкций. Наиболее очевидной из них является цикл while(). Цикл говорит, что до тех пор, пока в переменной $result остается запись для выборки, ее необходимо извлечь с помощью функции **mysql_fetch_row** и присвоить переменной $myrow. А после этого выполнить код, что расположен внутри фигурных скобок "{}". Приглядитесь к коду внимательнее и разберитесь в этой конструкции.

Для понимания этого кода разберем понятие "массив". Выполнение функции mysql_query дает в результате массив, который хранится в переменной $result. Если представить этот массив схематически, то он будет выглядеть так:

	0 id	1 first	2 last	3 address	4 position	Порядковый номер элемента массива
$result =	1	Bob	Smith	128 Here St, Cityname	Marketing Manager	0
	2	John	Roberts	45 There St ,Townville	Telephonist	1
	3	Brad	Johnson	1/34 Nowhere Blvd, Snowston	Doorman	2

Переменная $result является массивом. Причем не простым массивом, а двумерным. В нем содержатся три строки с номерами от 0 до 2. каждая из которых содержит 5 столбцов от 0 до 4. Для того, чтобы вывести на странице все записи, нам надо пройти от 0-й строчки массива до 2-й. Лучше всего это делать в цикле с помощью функции **mysql_fetch_row** (которая в переводе буквально означает – "выбрать ряд").

Функции **mysql_fetch_row** в качестве параметра подается массив $result. Функция выбирает из него строку, которую мы можем записать в переменную $myrow и автоматически переходит на следующую строку. Вызвав снова **mysql_fetch_row**, мы выберем следующую строку из массива, и так далее до тех пор, пока не достигнем конца массива. В этом случае **mysql_fetch_row** вернет значение *false*, которое послужит нам сигналом, что все записи выбраны и можно завершить цикл.

Теперь наша задача как-то вывести в теле цикла полученную запись. Выбранный ряд у нас хранится в переменной $myrow. Она также, как и $result, является массивом, только одномерным. Схематически это выглядит так:

	0 id	1 first	2 last	3 address	4 position	Порядковый номер элемента в массиве $results
$myrow =	1	Bob	Smith	128 Here St, Cityname	Marketing Manager	0

А вот как будет выглядеть содержимое переменной $myrow при втором прохождении цикла:

	0 id	1 first	2 last	3 address	4 position	Порядковый номер элемента в массиве

						$results
$myrow =	2	John	Roberts	45 There St ,Townville	Telephonist	1

И наконец, при третьем:

	0 id	1 first	2 last	3 address	4 position	Порядковый номер элемента в массиве $results
$myrow =	3	Brad	Johnson	1/34 Nowhere Blvd, Snowston	Doorman	2

К каждому столбцу в массиве $myrow мы можем обратиться по его порядковому номеру, который заключается в квадратные скобки. Например, в первом цикле, $myrow[1] равно "Bob", во втором $myrow[4] равно " Telephonist".

На первый взгляд процедура извлечения данных весьма сложна, но она вполне логически понятна и объяснима. Главное не забывайте, что элементы массивов нумеруются от 0, а не от 1, так как здесь мы имеем дело с компьютерной, а не человеческой логикой.

Далее, вывод переменных в HTML с помощью функции **printf()** – дело техники, уже знакомой нам по предыдущему примеру.

Наш код грешит одним недостатком: если в базе данных не будут найдены записи, удовлетворяющие нашему запросу, мы не получим никакого сообщения об этом. Неплохо было бы, чтобы программа выдавала какое-нибудь сообщение. Сделаем ее более дружественной.

Работа с MySQL (вывод данных из базы данных). Часть III

Взгляните на следующий код:

```
<html>
<body>
<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
$result = mysql_query("SELECT * FROM employees",$db);
if ($myrow = mysql_fetch_array($result))
{
        echo "<table border=1>\n";
        echo "<tr><td>Name</td><td>Position</td></tr>\n";
        do
        {
        printf("<tr><td>%s %s</td><td>%s</td></tr>\n", $myrow["first"], $myrow["last"],
        $myrow["address"]);
        }
        while ($myrow = mysql_fetch_array($result));
        echo "</table>\n";
}
else
{
        echo "Sorry, no records were found!";
}
?>
</body>
```

```
</html>
```

Рис. 17

В данном коде мы опять ввели некоторые новые понятия, но они достаточно просты. Во-первых, вместо функции **mysql_fetch_row()** мы использовали функцию **mysql_fetch_array()**. Она работает точно так же, как и **mysql_fetch_row()** за одним замечательным исключением: с помощью этой функции мы можем обращаться к каждому полю массива не по номеру, а по имени. Например, если раньше для получения имени нам приходилось писать $myrow[1] (1 – второй столбец массива), то теперь мы можем писать $myrow["first"] ("first" – название столбца в базе данных и в массиве). Второй вариант естественно гораздо информативнее и удобнее.

Кроме этого, в коде использован цикл do/while и условная конструкция if-else.

Выражение if-else говорит, что если мы можем присвоить значение $myrow, то надо начать выборку, в противном случае мы понимаем, что записей нет, переходим к блоку else и выводим соответствующее сообщение. Чтобы проверить, как работает эта часть кода, замените SQL-выражение на "SELECT * FROM employees WHERE id=6" или на какое-нибудь другое, которое не даст результата.

Цикл do/while – это всего лишь вариант цикла while(), который мы использовали в предыдущем примере. Мы обратились за помощью к циклу do/while по одной простой причине. В конструкции if мы уже сделали выборку первого ряда и присвоили его переменной $myrow. Если бы мы сейчас воспользовались прежней конструкцией (т.е. while ($myrow = mysql_fetch_row($result)), мы бы затерли значения первой выбранной записи, заменив ее значениями второй записи. В случае же с циклом do/while мы проверяем условие после того, как код цикла выполнится по крайней мере один раз. Таким образом, ни одна запись не ускользнет из наших рук.

А сейчас давайте сделаем код в цикле и if-else конструкцию еще более красивым. Уверен, результат вам понравится.

Создаем ссылки на лету

Сейчас мы поработаем со параметрами запроса. Как вы уже наверняка знаете, существует три способа передачи параметров запроса. Первый, использовать метод GET в форме (с ним мы разобрались в главе "Работа с формами"). Второй – набрать параметры прямо в адресной строке броузера. И третий, это вставить параметры в обычную ссылку на странице. То есть сделать ссылку примерно такого вида

```
<a href="http://my_machine/mypage.php3?id=1">
```

Рис. 18

Сейчас мы научимся создавать такие ссылки на лету.

Для начала, давайте обратимся к базе данных и выведем список персонала. Взгляните на следующий код. Многое в нем вам будет знакомо.

```
<html>
<body>
<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
$result = mysql_query("SELECT * FROM employees",$db);
if ($myrow = mysql_fetch_array($result))
{
        do
        {
        printf("<a href=\"%s?id=%s\">%s %s</a><br>\n", $PHP_SELF,  $myrow["id"],$myrow["first"],
$myrow["last"]);
        }
        while ($myrow = mysql_fetch_array($result));
}
```

```
else
{
        echo "Sorry, no records were found!";
}
?>
</body>
</html>
```

Рис. 19

Все вам должно быть знакомо, кроме функции **printf()**. Поэтому, давайте рассмотрим ее поближе. Во-первых, обратите внимание на обратные косые черты. Они говорят PHP-движку, что необходимо вывести символ, следующий за чертой, а не рассматривать его, как служебный символ или как часть кода. В данном случае это касается кавычки, которая нам нужна в тексте ссылки, но для PHP является символом окончания текстовой строки.

Далее, в коде используется интересная переменная $PHP_SELF. В этой переменной всегда хранится имя и URL текущей страницы. В данном случае эта переменная важна для нас потому, что мы хотим через ссылку вызвать страницу из нее самой. То есть вместо того, чтобы делать две страницы, содержащие разные коды для разных действий, мы все действия запихнули в одну страницу. С помощью условий if-else мы будем переводить стрелки с одного кода на другой, гоняя одну и ту же страницу по кругу. Это конечно увеличит размер страницы и время, необходимое на ее обработку, но в некоторых случая, такой трюк очень удобен.

Переменная $PHP_SELF гарантирует нам, что наша страница будет работать даже в том случае, если мы перенесем ее в другой каталог или даже на другую машину.

Как мы уже сказали, ссылки, сгенерированные в цикле ссылаются на ту же самую страницу, только к имени самой страницы на лету добавлена некоторая информация: переменные и их значения. Переменные, которые передаются в строке-ссылке, автоматически создаются PHP-движком, и к ним можно обращаться так, как если бы вы их создавали в коде сами. При втором проходе страницы наша программа отреагирует на эти пары name=value и направит ход исполнения на другие рельсы. В данном случае мы проверим, есть ли переменная $id, и в зависимости от результата выполним тот или иной код. Вот как это будет выглядеть:

```
<html>
<body>
<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
// display individual record
if ($id)
{
        $result = mysql_query("SELECT * FROM employees WHERE id=$id",$db);
        $myrow = mysql_fetch_array($result);
        printf("First name: %s\n<br>", $myrow["first"]);
        printf("Last name: %s\n<br>", $myrow["last"]);
        printf("Address: %s\n<br>", $myrow["address"]);
        printf("Position: %s\n<br>", $myrow["position"]);
}
else
{
        // show employee list
        $result = mysql_query("SELECT * FROM employees",$db);
        if ($myrow = mysql_fetch_array($result))
        {
```

```
        // display list if there are records to display
                do
                {
                printf("<a href=\"%s?id=%s\">%s %s</a><br>\n", $PHP_SELF, $myrow["id"],
                $myrow["first"], $myrow["last"]);
                }
                while ($myrow = mysql_fetch_array($result));
        }
        else
        {
        // no records to display
        echo "Sorry, no records were found!";
        }
}
?>
</body>
</html>
```

Рис. 20

Код усложнился, поэтому мы добавили в него некоторые комментарии, чтобы он стал яснее. Для однострочных комментариев можно использовать символы //. Если комментарий нужно уместить на нескольких строчках, используйте скобки /* … */.

Итак, магия закончилась, вы наконец создали действительно полезную PHP-страницу, работающую с MySQL. Теперь давайте научимся добавлять данные с помощью форм.

Работа с MySQL (сохранение данных в базе данных)

Мы научились извлекать данные из базы и выводить их на странице. Теперь давай попробуем осуществить обратное действие. С PHP это не составит большого труда. Создадим простую форму:

```
<html>
<body>
<form method="post" action="<?php echo $PHP_SELF?>">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Address:<input type="Text" name="address"><br>
Position:<input type="Text" name="position"><br>
<input type="Submit" name="submit" value="Enter information">
</form>
</body>
</html>
```

Рис. 21

Обратите внимание, мы опять используем переменную $PHP_SELF. Как мы уже сказали, PHP-код можно как угодно перемешивать с обычным HTML. Также обратите внимание, что название каждого элемента формы совпадает с названием поля в базе данных. Вообще-то, это не обязательно, но весьма удобно, чтобы в дальнейшем не запутаться в том, какая переменная какому полю в базе данных соответствует.

Помимо этого мы присвоили имя кнопке Submit. Это сделано для того, чтобы в коде затем проверить, есть ли переменная $submit. Таким образом, когда страница будет вызываться, мы будем узнавать, вызывается ли она в первый или во второй раз.

Следует еще раз отметить, что вовсе не обязательно писать код так, чтобы страница снова и снова вызывала саму себя. Программу можно разделить на две, три и более страниц, если угодно. Но при таком подходе, вся программа находится в одном файле, а это бывает весьма удобно.

Итак, давайте добавим код, который будет проверять, введены ли в форму данные. Пока это будет лишь простая проверка, при которой все переменные, передаваемые странице, будут выводиться на экран с помощью переменной $HTTP_POST_VARS. Эта переменная удобна в случае отладки. Если вы хотите вывести на экран вообще все переменные, используемые в странице, вызовите переменную $GLOBALS.

```
<html>
<body>
<?php
if ($submit) {
// process form
while (list($name, $value) = each($HTTP_POST_VARS)) {
echo "$name = $value<br>\n";
}
} else{
// display form
?>
<form method="post" action="<?php echo $PHP_SELF?>">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Address:<input type="Text" name="address"><br>
Position:<input type="Text" name="position"><br>
<input type="Submit" name="submit" value="Enter information">
</form>
<?php
} // end if
?>
</body>
</html>
```

Рис. 22

Ну что ж, выглядит неплохо. Теперь давайте возьмем подданную через форму информацию и внесем ее в базу данных.

```
<html>
<body>
<?php
if ($submit) {
	// process form
	$db = mysql_connect("localhost", "root");
	mysql_select_db("mydb",$db);
	$sql = "INSERT INTO employees (first,last,address,position) VALUES
('$first','$last','$address','$position')";
	$result = mysql_query($sql);
	echo "Thank you! Information entered.\n";
} else{
	// display form
```

```
?>
<form method="post" action="<?php echo $PHP_SELF?>">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Address:<input type="Text" name="address"><br>
Position:<input type="Text" name="position"><br>
<input type="Submit" name="submit" value="Enter information">
</form>
<?php
}
// end if
?>
</body>
</html>
```

Рис. 23

Мы внесли данные в базу. Тем не менее наш код далек от идеального. Что случится, если при заполнении формы кто-то оставит пустые поля или введет текст в поле, в которое надо ввести число? Что произойдет, если в поданных данных будет ошибка? Не беспокойтесь. Сейчас мы все исправим.

На протяжении всего учебника мы записывали SQL-выражение в переменную ($sql), прежде чем передать запрос в базу данных через функцию **mysql_query()**. Это делается на случай отладки. Если что-то пойдет не так, мы всегда сможем вывести интересующее нас SQL-выражение на экран и проверить, нет ли в нем ошибок.

Мы уже знаем, как вставлять данные в базу. Теперь давайте научимся менять записи, которые уже находятся в таблице. Редактирование данных сочетает в себе два кода. Которые мы уже проходили: извлечение данных из базы с выводом их на экран, и внесение данных через форму обратно в базу. Тем не менее программа правки данных немного отличается тем, что мы в форме должны вывести некую конкретную запись. Для начала давайте воспользуемся кодом из предыдущей главы, для вывода списка служащих на экран. Однако теперь информацию о служащих мы будет отображать в форме. Код страницы будет выглядеть так:

```
<html>
<body>
<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
if ($id) {
// query the DB
$sql = "SELECT * FROM employees WHERE id=$id";
$result = mysql_query($sql);
$myrow = mysql_fetch_array($result);
?>
<form method="post" action="<?php echo $PHP_SELF?>">
<input type=hidden name="id" value="<?php echo $myrow["id"] ?>">
First name:<input type="Text" name="first" value="<?php echo $myrow["first"] ?>"><br>
Last name:<input type="Text" name="last" value="<?php echo $myrow["last"] ?>"><br>
Address:<input type="Text" name="address" value="<?php echo $myrow["address"] ?>"><br>
Position:<input type="Text" name="position" value="<?php echo $myrow["position"] ?>"><br>
<input type="Submit" name="submit" value="Enter information">
</form>
<?php
```

```
} else {
// display list of employees
$result = mysql_query("SELECT * FROM employees",$db);
while ($myrow = mysql_fetch_array($result)) {
printf("<a href=\"%s?id=%s\">%s %s</a><br>\n", $PHP_SELF, $myrow["id"], $myrow["first"],
$myrow["last"]);
}
}
?>
</body>
</html>
```

Рис. 24

В этой странице мы просто вывели в каждое поле формы соответствующее значение из базы данных, что было достаточно несложно. Теперь мы усложним программу. Добавим к ней возможность внесения отредактированных данных назад в базу. Опять таки мы прибегаем к помощи кнопки Submit, которой присваиваем имя, чтобы при втором проходе страницы проверить, какую часть кода нам надо выполнять. Также мы здесь используем слегка измененное SQL-выражение.

```
<html>
<body>
<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
if ($id) {
if ($submit) {
$sql = "UPDATE employees SET first='$first',last='$last',address='$address',position='$position'
WHERE id=$id";
$result = mysql_query($sql);
echo "Thank you! Information updated.\n";
} else {
// query the DB
$sql = "SELECT * FROM employees WHERE id=$id";
$result = mysql_query($sql);
$myrow = mysql_fetch_array($result);
?>
<form method="post" action="<?php echo $PHP_SELF?>">
<input type=hidden name="id" value="<?php echo $myrow["id"] ?>">
First name:<input type="Text" name="first" value="<?php echo $myrow["first"] ?>"><br>
Last name:<input type="Text" name="last" value="<?php echo $myrow["last"] ?>"><br>
Address:<input type="Text" name="address" value="<?php echo $myrow["address"] ?>"><br>
Position:<input type="Text" name="position" value="<?php echo $myrow["position"] ?>"><br>
<input type="Submit" name="submit" value="Enter information">
</form>
<?php
}
} else {
// display list of employees
$result = mysql_query("SELECT * FROM employees",$db);
```

```
while ($myrow = mysql_fetch_array($result)) {
printf("<a href=\"%s?id=%s\">%s %s</a><br>\n", $PHP_SELF, $myrow["id"], $myrow["first"],
$myrow["last"]);
}
}
?>
</body>
</html>
```

Рис. 25

Вот так. Нам удалось вместить все, что мы знаем и умеем в один код. Здесь вы можете увидеть, как мы используем выражение if() внутри другого выражения if() для последовательно проверки нескольких условий.

Теперь пришло время свести все вместе и создать одну супер-крутую PHP-страницу.

```
<html>
<body>
<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
if ($submit) {
// here if no ID then adding else we're editing
if ($id) {
$sql = "UPDATE employees SET first='$first',last='$last',address='$address',position='$position'
WHERE id=$id";
} else {
$sql = "INSERT INTO employees (first,last,address,position) VALUES
('$first','$last','$address','$position')";
}
// run SQL against the DB
$result = mysql_query($sql);
echo "Record updated/edited!<p>";
} elseif ($delete) {
// delete a record
$sql = "DELETE FROM employees WHERE id=$id";
$result = mysql_query($sql);
echo "$sql Record deleted!<p>";
} else {
// this part happens if we don't press submit
if (!$id) {
// print the list if there is not editing
$result = mysql_query("SELECT * FROM employees",$db);
while ($myrow = mysql_fetch_array($result)) {
printf("<a href=\"%s?id=%s\">%s %s</a> \n", $PHP_SELF, $myrow["id"], $myrow["first"],
$myrow["last"]);
printf("<a href=\"%s?id=%s&delete=yes\">(DELETE)</a><br>", $PHP_SELF, $myrow["id"]);
}
}
?>
<P>
```

```
<a href="<?php echo $PHP_SELF?>">ADD A RECORD</a>
<P>
<form method="post" action="<?php echo $PHP_SELF?>">
<?php
if ($id)
{
        // editing so select a record
        $sql = "SELECT * FROM employees WHERE id=$id";
        $result = mysql_query($sql);
        $myrow = mysql_fetch_array($result);
        $id = $myrow["id"];
        $first = $myrow["first"];
        $last = $myrow["last"];
        $address = $myrow["address"];
        $position = $myrow["position"];
        // print the id for editing
?>
<input type=hidden name="id" value="<?php echo $id ?>">
<?php
}
?>
First name:<input type="Text" name="first" value="<?php echo $first ?>"><br>
Last name:<input type="Text" name="last" value="<?php echo $last ?>"><br>
Address:<input type="Text" name="address" value="<?php echo $address ?>"><br>
Position:<input type="Text" name="position" value="<?php echo $position ?>"><br>
<input type="Submit" name="submit" value="Enter information">
</form>
<?php
}
?>
</body>
</html>
```

Рис. 26

На первый взгляд код выглядит сложным, однако это не так. Программа делится на три части. Первое if() выражение проверяет, была ли нажата кнопка Submit, и если была, проводится проверка, есть ли в поданных данных переменная $id. Если ее нет, значит происходит добавление новой записи. В противном случае мы редактируем уже существующую запись.

Далее мы проверяем, определена ли переменная $delete. Если да, мы удаляем запись. Обратите внимание, что в первом выражении if() мы проверяем переменную, которая была подана с помощью метода POST, а в данном if() выражении мы проверяем переменную, которая является частью данных отправленных с помощью метода GET.

Наконец, мы переходим к действию, которое будет выполняться по умолчанию: то есть выводим просто список служащих и форму. Здесь мы опять проверяем существование переменной $id. Если она существует, мы просим базу данных выдать сведения о выбранном служащем. В противном случае выводим пустую форму.

Все, чему мы научились, мы поместили в один большой код. Мы использовали циклы while() и выражения if(), а также целую гамму основных команд языка SQL - SELECT, INSERT, UPDATE, и DELETE

Наконец, мы рассмотрели, как можно передавать информацию от одной страницы к другой через URL с помощью ссылок и через формы.

Маленькие советы

Начинающий: echo "$var";

Профи: echo $var;

Начинающий: print("Test");

Профи: print "Test";

Начинающий: echo"PHP";

Профи: ?>PHP<?

Начинающий: $a[0]=1; $a[1]=2; $a[2]=3;

Профи: $a = array(1,2,3);

Начинающий: if($a>1) { $b=2; } else { $b=3; }

Профи: $b = ($a>1) ? 2:3;

Начинающий: $result=mysql_query(...);

Профи: $result=mysql_query(...) or die (mysql_error());

Варианты построения сети

О том, как использовать несколько машин для PHP и MySQL, чтобы распределить нагрузку.

Эта глава предназначена тем, кто размещаете свое приложение не на чужом сервере, а на своем и пользуется своей собственной линией выхода в Интернет. Во-первых, разрешите вас поздравить с таким богатством, так как вы – одни из немногих, кто себе может позволить такую роскошь. Скорей всего у вас есть внутренняя локальная сеть, которая через прокси смотрит в Интернет. На прокси-сервере расположен и Web-сервер, с помощью которого вы выставили в Интернет Web-узел своей компании или организации. Понятно, что установка дополнительных программ увеличит нагрузку на сервер, который мы уверены и без того загружен.

Мы предлагаем решение, которое позволит распределить нагрузку по двум машинам и значит увеличить скорость работы PHP-приложений, взаимодействующих с базой данных.

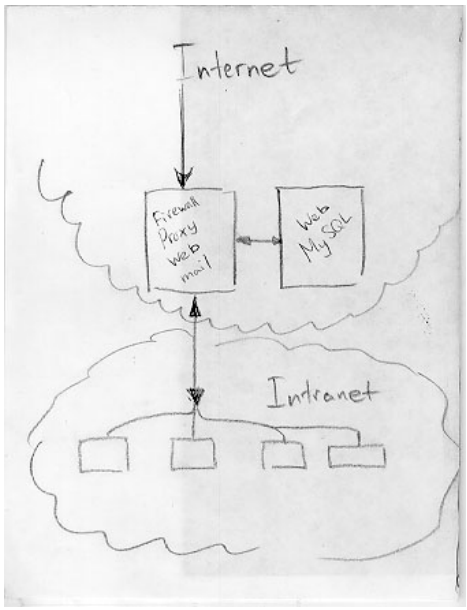

Рис. 27

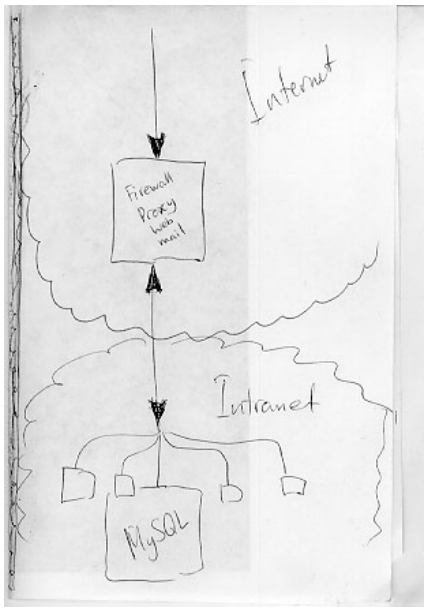

Рис. 28

Первый вариант изображен на Рис. 27. В Интернет выносятся две машины. На одной устанавливается Web-сервер и PHP, а на другой сервер MySQL. При обращении к базе данных с помощью функции **mysql_connect()** вы указываете не как обычно *localhost*, а IP-адрес соседней машины.

Например:

IP-адрес первой машины 226.117.45.2

IP-адрес второй машины 226.117.45.7

Тогда в PHP-странице, что работает на первой машине будет строка

```
mysql_connect("226.117.45.7",$db);
```

Недостаток этой схемы в том, что для второй машины требуется дополнительный IP-адрес. С другой стороны к вашей базе данных могут обратиться из других PHP-страниц, которые хранятся на других машинах в Интернете.

Вторая схема, изображенная на Рис. 28, еще более изощренная. В локальной сети выбирается машина, на которую устанавливается сервер MySQL. При обращении к базе данных с помощью функции mysql_connect() вы опять-таки указываете IP-адрес второй машины.

Например:

IP-адрес первой машины: 226.117.45.2 – по такому адресу она видна из Интернета,

192.168.0.5 – по такому адресу она видна внутри сети

IP-адрес второй машины 192.168.0.104 - по такому адресу она видна внутри сети, а из Интернета не видна

Тогда в PHP-странице, что работает на первой внешней машине будет строка:

```
mysql_connect("192.168.0.104",$db);
```

Пример создания законченного приложения

Создадим простой HTML файл-форму.

```
<HTML>
<HEAD>
<TITLE>Запрос информации</TITLE>
<BODY>
<CENTER>
Хотите больше знать о наших товарах?
<P>
<TABLE WIDTH = 400><TR><TD align = right>
<FORM ACTION="email.php3" METHOD="POST">
Ваше имя:<BR>
<INPUT TYPE="text" NAME="name" SIZE="20" MAXLENGTH="30">
<P>
Ваш email:<BR>
<INPUT TYPE="text" NAME="email" SIZE="20" MAXLENGTH="30">
<P>
Меня интересуют:
<SELECT NAME="preference">
<OPTION value = "Яблоки">Яблоки
<OPTION value = "Апельсины">Апельсины
</SELECT>
<P>
<INPUT TYPE="submit" VALUE="Отправить запрос!">
</FORM>
</TD></TR></TABLE></CENTER>
</BODY>
</HTML>
```

Рис. 29

Назовем этот файл request.html. В нем мы указали, что данные формы будут обрабатываться файлом email.php3. Приведем его содержание:

```
<?
/* Этот скрипт получает переменные из request.html */
PRINT "<CENTER>";
PRINT "Привет, $name.";
PRINT "<BR><BR>";
PRINT "Спасибо за ваш интерес.<BR><BR>";
PRINT "Вас интересуют $preference. Информацию о них мы пошлем вам на email: $email.";
PRINT "</CENTER>";
?>
```

Рис. 30

Теперь, если пользователь вызовет request.html и наберет в форме имя “Вася”, email: vasya@pupkin.com и скажет, что его интересуют “Яблоки”, а после этого нажмет "Отправить запрос!", то на экране появится следующий текст:

Привет, Вася

Спасибо за ваш интерес.

Вас интересуют Яблоки. Информацию о них мы пошлем вам на email: vasya@pupkin.com

Рис. 31

Давайте усложним пример. Мы должны сдержать обещание и выслать email.

Для этого в PHP есть функция MAIL.

void mail(string to, string subject, string message, string add_headers);

to – email адрес получателя.

subject – тема письма.

message – собственно текст сообщения.

add_headers – другие параметры заголовка письма (необязательный параметр).

Допишем в конец файла email.php3 следующий код:

```
<?
mail($email, "Запрос на информацию", "$name\n
Спасибо за ваш интерес!\n
Вас интересуют $preference\n
Мы их распространяем бесплатно. Обратитесь в ближайший филиал нашей компании и получите ящик этого
продукта.\n
");
mail("administration@me.com",
"Был запрос на информацию.",
"$name интересовали $preference\n
email-адрес: $email. \n");
?>
```

Рис. 32

Вот теперь пользователь будет получать письмо с более подробной информацией о наших товарах. Также письмо получит и администратор сайта.

Когда интересующихся нашими товарами станет очень много, мы захотим их как-то упорядочить и хранить информацию о них в базе данных.

Теперь наш файл email.php3 будет иметь следующий вид:

```
<?
/* Этот скрипт получает переменные из request.html */

/* Некоторые переменные */

$hostname = "localhost";
$username = "myusername";
$password = "mypassword";
$dbName = "products";

/* Таблица MySQL, в которой хранятся данные */
$userstable = "clients";

/* email администратора */
$adminaddress = "administration@me.com";

/* создать соединение */
$db = mysql_connect($hostname,$username,$password) or die("Не могу создать соединение ");

mysql_select_db("$dbName",$db) or die("Не могу выбрать базу данных ");

print "<CENTER>";
print "Привет, $name.";
print "<BR><BR>";
print "Спасибо за ваш интерес.<BR><BR>";
print "Вас интересуют $preference. Информацию о них мы пошлем вам на email: $email.";
print "</CENTER>";

/* Отправляем email */
mail($email, "Запрос на информацию", "$namen\n
Спасибо за ваш интерес!\n
Вас интересуют $preference\n
Мы их распространяем бесплатно. Обратитесь в ближайший филиал нашей компании и получите ящик этого
продукта.\n
");

mail("administration@me.com",
"Был запрос на информацию.",
"$name интересовали $preference\n
email-адрес: $email. \n");

/* Вставить информацию о клиенте в таблицу */
$query = "INSERT INTO $userstable VALUES('$name','$email', '$preference')";

$result = mysql_query($query);

print "Информация о вас занесена в базу данных.";

/* Закрыть соединение */
```

```
mysql_close();
?>
```

Рис. 33

Теперь кроме письменных уведомлений, информация о клиенте и его интересах будет заносится в таблицу MySQL.

После занесения данных, нас иногда будет интересовать вопрос так кого же из наших клиентов интересует товар “Яблоки” (не путать с Apple Macintosh, по поводу Apple Macintosh см. www.stealthcomp.com).

Напишем скрипт apple.php3

```
<?/* Скрипт показывает клиентов, которые яблоки любят больше, чем апельсины */
$hostname = "localhost";
$username = "myusername";
$password = "mypassword";
$dbName = "products";

/* Таблица MySQL, в которой хранятся данные */
$userstable = "clients";

/* создать соединение */
mysql_connect($hostname,$username,$password) or die("Не могу создать соединение ");

@mysql_select_db("$dbName") or die("Не могу выбрать базу данных ");

/* Выбрать всех клиентов - яблочников */
$query = "SELECT * FROM $userstable WHERE choice = 'Яблоки'";

$result = mysql_query($query);

/* Как много нашлось таких */
$number = mysql_numrows($result);

/* Напечатать всех в красивом виде*/
$i = 0;

if ($number == 0)

{
       print "<CENTER><P>Любителей яблок нет</CENTER>";
}

elseif ($number > 0)

{
       print "<CENTER><P>Количество любителей яблок: $number<BR><BR>";
       while ($i < $number)

       {
              $name = mysql_result($result,$i,"name");
              $email = mysql_result($result,$i,"email");
              print "Клиент $name любит Яблоки.<BR>";
              print "Его Email: $email.";
              print "<BR><BR>";
              $i++;
       }
print "</CENTER>";
}
?>
```

Рис. 34

Здесь мы использовали новую функцию:

> **int mysql_num_rows(int result);**
>
> **Параметры:**
>
> result – содержит ID результата запроса.
>
> *Функция возвращает количество строк в результате запроса.*

Вот и все, коммерческий продукт практически готов.

Все вопросы и замечания по данной работе присылайте на E-mail: kachanov@ogs.gomel.by.