Лабораторная № 10

11. Связь с базами данных на примере MySQL

Одной из самых приятных особенностей языка PHP является легкость, с которой программист на нем может общаться с базами данных. В этой главе в основном будет говориться о СУБД MySQL, но и с остальными базами данных PHP работает аналогично и с такой же легкостью. Из-за чего уделяется такое внимание именно MySQL? Потому что эта система управления базами данных соответствует духу PHP, она распространяется бесплатно и обладает достаточной мощностью для того, чтобы использовать ее для решения реальных задач. Кроме того, существуют версии MySQL для самых разных платформ. О работе с сервером MySQL можно прочесть в файле **mysql-php.mht** (в папке **info**)

Подключение к серверу базы данных

Перед тем как начинать работать со своей базой данных, вам нужно подключиться к серверу MySQL. Для этого в языке PHP есть функция `mysql_connect()`. Данная функция имеет три аргумента, в которых указывается имя компьютера, имя пользователя и пароль. Если вы опустите эти необязательные аргументы, то функция будет считать, что требуется компьютер `localhost`, а имя пользователя и его пароль не требуются, т.е. не установлены в таблице `mysqluser`. Конечно, такое подключение не имеет большого смысла — его можно использовать только для проверки сервера, поэтому в дальнейшем в наших примерах мы будем указывать и имя пользователя (для данных лабораторных это **stud**), и его пароль (**stud**). Функция `mysql_connect()` возвращает идентификатор подключения, если все прошло успешно. Этот идентификатор можно сохранить в переменной и в дальнейшем пользоваться им для работы с сервером MySQL.

В следующем примере приводится фрагмент, выполняющий подключение к серверу баз данных:

```
$mysql_user = "stud";
$mysql_password = "stud";

$conn = mysql_connect("localhost",
$mysql_user, $mysql_password);

if (!$conn ) die("Нет соединения с MySQL");
```

Если вы используете PHP в сочетании с сервером Apache, то для подключения к базе данных можете воспользоваться функцией `mysql_pconnect()`. С точки зрения программиста обе эти функции работают одинаково, однако незначительная разница все-таки есть. Если вы воспользуетесь функцией

mysql_pconnect(), то соединение с сервером не исчезает после завершения работы программы или вызова функции mysql_close().

Выбор базы данных

После того как соединение с сервером MySQL установлено, вам нужно выбрать базу данных, с которой собираетесь работать. Для этого существует функция mysql_select_db(). Этой функции нужно передать имя базы данных и второй необязательный аргумент — идентификатор подключения к серверу. Если этот второй аргумент опустить, то по умолчанию будет использован идентификатор последнего полученного подключения. Функция возвращает true, если указанная база данных существует и доступ к ней возможен. В следующем примере мы выбираем базу данных с именем **study**.

```
$database = "study";
mysql_select_db($database)
or die ("Нельзя открыть $database");
```

Обработка ошибок

До сих пор мы проверяли значения, возвращаемые функциями MySQL, и вызывали функцию die() для прекращения выполнения программы. Однако можно вывести на экран браузера более подробное сообщение об ошибке, которое может пригодиться при отладке программы. При любом неудачном завершении операции MySQL устанавливает номер ошибки и строку с ее описанием. Номер ошибки можно получить используя функцию mysql_errno(), а строку с описанием — с помощью функции mysql_error(), которая возвращает строку с описанием ошибки, если такая произойдет.

Если программа будет пытаться открыть несуществующую базу данных (например, stadi), то в этом случае функция die() выведет примерно такое сообщение:

```
Нельзя открыть stadi: Access denied for user:
'stud@localhost' to database 'stadi'
```

Добавление данных в таблицу

Получив доступ к базе данных, мы можем добавлять информацию в ее таблицы. Рассмотрим это на примере. Представьте себе, что нам нужно создать узел, на котором пользователи сети могут покупать для себя доменные имена.

Мы создаем в базе данных study таблицу с именем domains_**NNN** (где **NNN** — *первые 3-4 латинских буквы вашей фамилии, напр.* domains_kov).

Таблица содержит 4 поля:

- поле с именем `id`, имеющее целый тип (`INT`), которое всегда должно быть заполнено (`NOT NULL`) и значение которого будет автоматически увеличиваться при добавлении новых записей (`AUTO_INCREMENT`), это поле является первичным ключом (`PRIMARY KEY`);
- поле `domain`, куда может быть записана строка переменной длины (не более 40 символов) — типа `TEXT;`
- поле `myname`, содержащее фамилию пользователя;
- поле `mail`, содержащее адрес пользователя — типа `TEXT` без указания ограничения по количеству символов.

Для создания этой таблицы была использована такая команда SQL:

```
create table domains_NNN
       (id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY(id),
        domain TEXT(40),
        myname TEXT(40),
        mail TEXT
       );
```

Для добавления данных в эту таблицу нам нужно сконструировать и выполнить запрос SQL. Для этого в языке PHP есть функция `mysql_query()`. Этой функции нужно передать строку с запросом и необязательный идентификатор подключения. Если данный идентификатор опущен, то будет использоваться последнее полученное подключение. Функция возвращает положительное число, в случае успешного выполнения запроса, и `false`, если в запросе содержится ошибка или если вы не имеет права на выполнение такого запроса.

```
$query = "create table domains_NNN
         (id INT NOT NULL AUTO_INCREMENT,
          PRIMARY KEY(id),
          domain TEXT(40),
          myname TEXT(40),
          mail TEXT
         )";
// здесь NNN - начало вашей фамилии

$result = mysql_query($query)
          or die ("<p>Ошибка: ".mysql_error());
```

Имейте в виду, что успешное выполнение запроса не обязательно подразумевает изменение данных в таблице. В листинге 11.1 приведена программа, которая подключается к серверу и выбирает базу данных, а

функция mysql_query() выполняет оператор INSERT, вставляющий данные в таблицу domains_NNN базы данных study.

Листинг 11.1. Подключение к базе данных и добавление записи в таблицу

```
<html> <head>
<title> Листинг 11-1. Добавление записи в таблицу
</title> </head> <body>
<?php
$user = "stud"; $pass = "stud";
$db = "study";
$table = "domains_NNN";
// здесь и далее NNN – начало вашей фамилии!
$conn = mysql_connect("localhost", $user, $pass);
if (!$conn ) die("Нет соединения с MySQL");
mysql_select_db($db, $conn )
     or die ("Нельзя открыть $db: ".mysql_error());
$query = "create table $table
          (id INT NOT NULL AUTO_INCREMENT,
           PRIMARY KEY(id),
           domain TEXT(40),
           myname TEXT(40),
           mail TEXT
          )";
$result = mysql_query($query)
           or die ("<p>Ошибка: ".mysql_error());
$query = "INSERT INTO $table (domain, myname, mail)
   VALUES('123abc.com', 'abc', 'abc@mail.ru')";
mysql_query($query, $conn)
     or die ("Нельзя добавить данные в таблицу
              $table: " .mysql_error());
print "<p>Данные в таблицу $table добавлены";
mysql_close($conn);
?>
</body> </html>
```

Заметьте, что при работе с одной и той же таблицей удобнее ее имя присвоить какой-нибудь переменной (в данном случае это $table). Если вдруг имя

таблицы в базе данных изменится, вам достаточно будет внести исправление всего в одну переменную, а не выискивать по всему скрипту, где еще используется данная таблица.

Обратите внимание и на то, что мы не указываем значение поля `id`. Это поле изменяется автоматически.

Естественно, что при каждом выполнении программы из листинга 11.1 в таблицу будет добавляться новая запись, содержащая одни и те же данные. В листинге 11.2 приведена программа, которая добавляет в таблицу данные, введенные пользователем.

Листинг 11.2. Добавление в базу данных информации, введенной пользователем

```
<html> <head>
<title> Листинг 11-2. Добавление в базу данных
        информации, введенной пользователем
</title> </head> <body>
<?php
function Add_to_database($domain, $myname, $mail,
                         &$dberror)
   {  #1
    $user = "stud"; $pass = "stud";
    $db = "study";  $table = "domains_NNN";
    $conn = mysql_connect("localhost", $user, $pass);
    if (!$conn )
      { $dberror = "Нет соединения с MySQL сервером";
        return false; }
    if (! mysql_select_db($db, $conn))
      { $dberror = mysql_error();
        return false;
      }
$query = "INSERT INTO $table (domain, myname, mail)
          VALUES('$domain', '$name', '$mail')";
    if (! mysql_query($query, $conn))
      { $dberror = mysql_error();
        return false;
      }
    return true;
   }  #1
```

```
function Write_form()
   {  #2
    print "<form action='ls11-2.php'
            method='POST'>\n";
    print "<p>Введите имя домена: \n";
    print "<input type='text' name='domain-f'> ";
    print "<p>Введите ваш e-mail: \n";
    print "<input type='text' name='mail-f'> ";
    print "<p>Введите вашу фамилию: \n";
    print "<input type='text' name='myname-f'> ";
    print "<p><input type='submit' value='Записать! '>\n
           </form>\n";
   }  #2
    // Ввод данных в таблицу
$domain=$_POST['domain-f'];
$myname=$_POST['myname-f'];
$mail=$_POST['mail-f'];
if (isset($domain) && isset($myname) && isset($mail) )
   {  #3
    // Обязательно проверять, что вводит пользователь!
    $dberror = "";
    $ret = Add_to_database($domain, $myname, $mail,
                           $dberror);
    if (!$ret)
          {print "Ошибка: $dberror<br>";}
    else  print "Спасибо";
   }  #3
else Write_form();
?> </body> </html>
```

В программе из листинга 11.2 мы для простоты и краткости опустили один очень важный момент — нами не проверялись данные, введенные пользователем. А так поступать, вообще-то, не следует. Нельзя доверять пользователю в вопросах достоверности данных. Все данные, введенные им в форму, нужно тщательно проверять.

С помощью функции `isset()` проверяется существование переменных `$domain`, `$myname` и `$mail`. Если эти переменные определены, т.е. им присвоены значения элементов формы `'domain-f'`, `'myname-f'`,

'mail-f', переданные методом POST, то мы вызываем функцию Add_to_database().

Функция Add_to_database() имеет 4 аргумента: переменные $domain, $myname, $mail, которые переданы пользователем, и строку $dberror. В эту строку будет записано сообщение об ошибке, если такая произойдет. Поэтому мы передаем эту переменную по ссылке. Все изменения, произошедшие с данной переменной внутри функции, повлияют на саму переменную, а не на ее копию.

Нами создается подключение к серверу MySQL, и если оно не происходит, то прекращаем выполнение программы, вернув значение false. Мы выбираем базу данных, содержащую таблицу domains и строим SQL-запрос, добавляющий переданные пользователем данные в таблицу. Этот запрос нами передается функции mysql_query(). Если хотя бы в одной из функций mysql_select_db() или в mysql_query() произойдет ошибка, то строка с описанием этой ошибки будет записана в переменную $dberror и функция вернет значение false. Если все в порядке, функция возвращает true.

В самой программе у нас есть возможность проверить значение, возвращенное функцией Add_to_database(). Если функция вернула значение true, то мы можем быть уверены в том, что данные добавлены. Если нет, то это означает, что произошла ошибка. В таком случае выводим сообщение об ошибке на экран браузера. Мы знаем, что описание этой ошибки содержится в переменной $dberror, поэтому включаем ее в текст сообщения.

Если в самом первом операторе if мы обнаруживаем, что одна из переменных — $domain, $myname или $mail — не определена, то это означает, что данные не переданы пользователем. В таком случае вызываем функцию Write_form(), которая выводит форму на экран пользователя.

Относительно функции mysql_query() нужно помнить следующее: ее аргумент, т.е. строка запроса к базе данных, **не должна** содержать символа «;». Фактически это означает, что аргументом может быть только один SQL-запрос.

Следовательно, если требуется последовательно выполнить несколько SQL-запросов, необходимо столько же раз использовать mysql_query(), как в следующем примере:

```
$query = "INSERT INTO domains_NNN (domain, myname, mail)
          values('aaa.com', 'aaa', 'aaa@mail.ru')";
$result = mysql_query($query);
$query = "INSERT INTO domains_NNN (domain, myname, mail)
          values('bbb.com', 'bbb', 'bbb@mail.ru')";
$result = mysql_query($query);
```

Получение значения автоматически изменяемого поля

В предыдущих примерах мы добавляли записи в таблицу, не беспокоясь о значении поля id, которое автоматически увеличивалось при каждом создании новой записи. Если нам понадобится значение этого поля для некоторой записи, мы всегда можем его получить с помощью SQL-запроса. Однако как поступить, если это значение нам нужно немедленно? В языке PHP есть функция mysql_insert_id(), возвращающая значение ключа последней добавленной записи. Этой функции нужно передать идентификатор подключения, а если его опустить, то будет использовано последнее подключение.

Таким образом, если нам необходимо сообщить пользователю номер, присвоенный его заказу, мы можем для этого воспользоваться функцией mysql_insert_id() непосредственно после добавления новых данных в таблицу.

```
$query = "INSERT INTO domains_NNN (domain, myname, mail)
          values('$domain', '$myname, '$mail')";
mysql_query($query, $conn);
$id = mysql_insert_id();
print "Спасибо. Ваш номер - $id.
Пожалуйста используйте его в других запросах.";
```

Доступ к информации

Мы уже умеем добавлять данные в базу данных, но необходимо еще и уметь их оттуда читать. Вы, наверное, догадываетесь, что для этого нужно сделать SQL-запрос типа SELECT, но как воспользоваться результатами этого запроса, т.е. возвращенными записями? При успешном выполнении запроса функция mysql_query() возвращает идентификатор результата запроса, и данный идентификатор мы можем передавать другим функциям для обработки результата.

Число записей, найденных в запросе

Узнать число записей, возвращенных в результате выполнения запроса SELECT, можно с помощью функции mysql_num_rows(). Этой функции нужно передать идентификатор запроса, а возвращает она число записей, содержащихся в этом результате.

Результат запроса

После выполнения запроса SELECT и получения его идентификатора вы можете в цикле просмотреть все записи, найденные в результате запроса. PHP

создает для вас внутренний указатель, в котором записана позиция в наборе записей результата. Этот указатель автоматически перемещается на следующую позицию после обращения к текущей записи.

С помощью функции `mysql_fetch_row()` можно для каждой записи получить массив, состоящий из ее полей. Этой функции нужно передать идентификатор запроса, а вернет она массив. По достижении конца запроса функция `mysql_fetch_row()` вернет значение `false`.

Чтение отдельных полей

Когда вы, выполнив запрос `SELECT`, получили идентификатор результата от функции `mysql_query()`, то можете определить количество возвращенных полей с помощью функции `mysql_num_fields()`. Этой функции нужно передать идентификатор результата, а возвратит она целое число, равное количеству найденных полей.

```
$result = mysql_query("SELECT * from domains_NNN");
$num_fields = mysql_num_fields($result);
```

Каждое поле в этом наборе имеет свой номер, начиная с нуля, и вы можете, указав идентификатор результата и номер поля, определить все его свойства, включая имя, тип, максимальную длину и флаги.

Для выяснения имени поля передайте идентификатор результата и номер поля функции `mysql_field_name()`.

```
$result = mysql_query("SELECT * from domains_NNN");
$num_fields = mysql_num_fields($result);
for ($x=0; $x<$num_fields; $x++)
     mysql_field_name($result, $x)."<br>\n";
```

Для того чтобы узнать максимальную длину поля, передайте идентификатор результата и номер поля функции `mysql_field_len()`.

```
$result = mysql_query("SELECT * from domains_NNN");
$num_fields = mysql_num_fields($result);
for ($x=0; $x<$num_fields; $x++)
     mysql_field_len($result, $x)."<br>\n";
```

Для того чтобы узнать флаги, связанные с этим полем, передайте идентификатор результата и номер поля функции `mysql_field_flags()`.

```
$result = mysql_query("SELECT * from domains_NNN");
$num_fields = mysql_num_fields($result);
for ($x=0; $x<$num_fields; $x++)
     mysql_field_flags($result, $x)."<br>\n";
```

Точно так же можно выяснить тип поля.

```
$result = mysql_query("SELECT * from domains_NNN");
$num_fields = mysql_num_fields($result);
for ($x=0; $x<$num_fields; $x++)
     mysql_field_type($result, $x)."<br>\n";
```

В листинге 11.3 приведен пример выполнения запроса `SELECT`, который запрашивает все строки таблицы `domains_NNN`, затем определяет размер этой таблицы с помощью функции `mysql_num_rows()` и выводит на экран всю таблицу `domains_NNN`.

Листинг 11.3. Вывод всех записей таблицы

```
<html> <head>
<title> Листинг 11-3. Вывод всех записей таблицы
</title> </head> <body>
<?php
$user = "stud";
$pass = "stud";
$db = "study";
$table = "domains_NNN";
$conn = mysql_connect("localhost", $user, $pass);
if (! $conn ) die("Нет соединения с MySQL");
mysql_select_db($db, $conn)
     or die ("Нельзя открыть $db: ".mysql_error());
$result = mysql_query("SELECT * FROM $table");
$num_rows = mysql_num_rows($result);
// количество записей в запросе
print "<P>В таблице $table содержится $num_rows строк";
$num_fields = mysql_num_fields($result);
// количество столбцов в запросе
print "<p><table border=1>\n";
print "<tr>\n";
for ($x=0; $x < $num_fields; $x++)
     {
      $name = mysql_field_name($result, $x);
      print "\t<th>$name</th>\n";
      // печатаем имя $x-того столбца
     }
      print "</tr>\n";
```

```
while ($a_row = mysql_fetch_row($result))
    {      // печатаем содержимое столбцов
     print "<tr>\n";
     foreach ($a_row as $field)  // $a_row – массив
     print "\t<td>$field</td>\n";
     print "</tr>\n";
    }
print "</table>\n";
mysql_close($conn);
?>
</body> </html>
```

После подключения к серверу базы данных и выбора базы мы с помощью функции `mysql_query()` посылаем запрос на сервер базы данных. Возвращенный результат запроса сохраняем в переменной `$result`. Потом нами будет использоваться эта переменная для обращения к результату запроса: в переменной `$num_rows` мы сохраняем количество записей в запросе, а в переменной `$num_fields` — количество полей (столбцов) в запросе. Затем, в цикле, в переменную `$name` заносим название `$x`–того поля таблицы и выводим это название в виде ячеек-заголовков `<th>`.

В условном выражении цикла `while` мы присваиваем переменной `$a_row` значение, возвращенное функцией `mysql_fetch_row()`. Результатом операции присваивания является значение ее правого операнда, поэтому данное условное выражение имеет значение `true`, если функция `mysql_fetch_row()` вернет положительное число. Внутри цикла `while` мы просматриваем массив, записанный в переменной `$a_row`, и выводим каждый элемент этого массива на экран, обрамляя его тегами ячейки таблицы.

Кроме того, к полям записи можно обратиться по имени: функция `mysql_fetch_array()` возвращает ассоциативный массив, в котором в качестве ключа используются имена полей. В следующем примере используется эта функция.

```
print "<table border=1>\n";
while ($a_row = mysql_fetch_array($result))
print "<tr>\n";
print "<td>$a_row[mail]</td>
       <td>$a_row[domain]</td>\n";
print "</tr>\n";
print "</table>\n";
```

Изменение данных

Данные в таблице можно изменять с помощью функции `mysql_query()` в сочетании с оператором `UPDATE`. Как и ранее, успешное выполнение оператора `UPDATE` не обязательно означает, что данные были фактически изменены. Для того чтобы узнать количество измененных данных в таблице, вам придется вызвать функцию `mysql_affected_rows()`. Этой функции нужно передать идентификатор подключения. Как и раньше, если данный идентификатор опустить, то будет использовано последнее подключение. Этой функцией можно пользоваться в сочетании с любым запросом, в результате которого данные могли быть изменены (`UPDATE`, `INSERT`, `REPLACE` или `DELETE`, но не `SELECT`!).

В листинге 11.4 приведен пример программы, позволяющей администратору базы данных изменять любые данные в поле `domain` таблицы `domains_NNN`.

Листинг 11.4. Использование функции `mysql_query()` для изменения данных в таблице

```
<html> <head>
<title> Листинг 11-4. Использование функции mysql_query()
        для изменения данных в таблице
</title> </head> <body>
<?php
$user = "stud"; $pass = "stud";
$db = "study";
$table = "domains_NNN";
$conn = mysql_connect("localhost", $user, $pass);
if (! $conn ) die("Нет соединения с MySQL");
mysql_select_db($db, $conn)
or die ("Нельзя открыть $db");
if (isset($domain) && isset($id))
   {$query = "UPDATE $table
              SET domain='$domain'
              WHERE id='$id' ";
    $result = mysql_query($query);
    if (! $result) die
       ("Нельзя обновить: ".mysql_error());
    print"<p>Таблица $table обновлена: "
   .mysql_affected_rows()." строк изменено";
   }
?>
```

```
<p><form action="ls11-4.php" method='POST'>
<select name="id">
<?
$result = mysql_query("SELECT domain, id
                       FROM $table");
while($a_row = mysql_fetch_array($result))
     {
      print "<OPTION VALUE=$a_row[id]";
      if (isset($id) && $id == $a_row[id])
          print " SELECTED";
      print "> $a_row[domain]\n";
     }
mysql_close($conn);
?>
</select>
  введите новое значение:
<input type="text" name="domain">
<p><input type="submit" value="Заменить">
</form>
</body> </html>
```

Мы открываем соединение с сервером и выбираем базу данных как обычно. Потом проверяем существование переменных `$domain` и `$id`. Если они существуют, то мы строим SQL-запрос, в котором изменяем значение поля `domain` для тех записей, у которых значение поля `id` совпадает со значением переменной `$id`. Мы не получим сообщения об ошибке, если попытаемся воспользоваться несуществующим значением переменной `$id` или если значение поля `domain` и так совпадает со значением переменной. Просто в таком случае функция `mysql_affected_rows()` вернет нам 0. Полученное значение (в данном примере это должна быть 1) мы выводим на экран браузера.

Нами выводится форма, с помощью которой администратор может вносить изменения. Обратите внимание на то, что мы опять используем функцию `mysql_query()` для того, чтобы получить значение полей `id` и `domain` и использовать их в HTML-тексте для формирования списка. Администратор будет выбирать в этом списке данные, подлежащие изменению. Если администратор уже передал форму и выбранное значение `id` совпадает текущим полем, мы выводим в элемент `OPTION` строку `SELECTED`. Это делается для того, чтобы новое значение, выбранное администратором, было выделено в форме.

Важное замечание. Если аргумент функции `mysql_query()` является переменной, значение которой было передано из другой программы с помощью метода `POST`, и в этой переменной содержится SQL-запрос, в котором есть апострофы, например

```
$query = "SELECT * FROM domains where id='2' ";
```

то эти апострофы будут экранированы обратной косой чертой (\). Для того, чтобы можно было использовать значение переменной `$query` в функции `mysql_query()`, необходимо применить функцию `stripslashes()`, которая убирает все управляющие символы (в том числе и обратную косую черту перед апострофами) из указанной строки.

В листинге 11.5 приведен пример программы, использующей функцию `stripslashes()`.

Листинг 11.5. Использование функции `stripslashes()`

```
<html> <head>
<title>Листинг 11-5. Использование функции stripslashes()
</title> </head> <body>
<?php
$user = "stud"; $pass = "stud";
$db = "study";
$table = "domains_NNN";
$conn = mysql_connect("localhost", $user, $pass);
if (! $conn ) die("Нет соединения с MySQL");
mysql_select_db($db, $conn)
or die ("Нельзя открыть $db");
$query = "SELECT * FROM $table WHERE id=\'2\' ";
settype($query, string);
$plain_query = stripslashes($query);
$result = mysql_query($plain_query);
$a_row = mysql_fetch_array($result);
print "<p>Ваше имя: $a_row[myname]";
mysql_close($conn);
?>
</body> </html>
```

Перед использованием функции `stripslashes()` желательно явно задать тип «строка» для переменной, которая будет ее аргументом, что делается с помощью функции `settype()`.

Получение информации о базе данных

До сих пор мы рассматривали функции, предназначенные для сохранения и чтения данных. Однако в PHP есть несколько функций для того, чтобы вы могли получить дополнительную информацию о базах данных, доступных в данном подключении.

Список баз данных

Список всех баз данных, доступных для данного подключения, можно получить с помощью функции `mysql_list_dbs()`. Этой функции нужно передать идентификатор подключения, а вернет она другой идентификатор, с помощью которого можно получить список всех доступных баз данных. Для этого вам необходимо вызвать функцию `mysql_tablename()` для каждой найденной базы данных. Эта функция имеет два аргумента: идентификатор результата и индекс базы данных. Функция возвращает имя указанной базы данных. Базы данных индексируются, начиная с нуля. Для того чтобы узнать, сколько баз данных найдено функцией `mysql_list_dbs()`, нужно вызвать функцию `mysql_num_rows()`, передав ей идентификатор результата, возвращенный функцией `mysql_list_dbs()`.

Имейте в виду, что идентификатор результата, возвращенный функцией `mysql_list_dbs()`, можно использовать аналогично тому, как это мы делали с идентификатором результата, возвращенным функцией `mysql_query()`. В частности, его можно передавать функции `mysql_fetch_row()`, которая вернет ассоциативный массив с именами баз данных.

Список таблиц в базе данных

С помощью функции `mysql_list_tables()` есть возможность получить список всех таблиц, входящих в указанную базу данных. Этой функции нужно передать имя базы данных и необязательный аргумент — идентификатор подключения. Если указанная база данных существует и вы имеет соответствующие права, то функция вернет вам идентификатор результата, который можно обработать функцией `mysql_fetch_row()` или аналогичной ей.

В следующем примере показано, как с помощью функции `mysql_list_tables()` можно получить список всех таблиц базы данных.

```
$result = mysql_list_tables("sample", $conn);
```

```
while ($tab_rows = mysql_fetch_row($result))
       print "$tab_rows[0]<br>\n";
```

Структура базы данных

В листинге 11.6 приведена программа, в которой с помощью всех рассмотренных ранее средств мы получаем информацию о базе данных **sample**, доступной нам через существующее подключение.

Листинг 11.6. Вывод структуры базы данных

```
<html> <head>
<title> Листинг 11-6. Вывод структуры базы данных
</title> </head> <body>
<?php
$user = "stud"; $pass = "stud";
$db = "sample";
$conn = mysql_connect("localhost", $user, $pass);
if (! $conn ) die("Нет соединения с MySQL");
    $tab_res = mysql_list_tables($db, $conn);
    print "<dl><dd>\n";
    while ($tab_rows = mysql_fetch_row($tab_res))
      {  #1
       print "<p><b>$tab_rows[0]</b>\n";
//$tab_rows[0] т.к. работаем только с одной БД
       $query_res = mysql_query(
                    "SELECT * from $tab_rows[0]");
       $num_fields = mysql_num_fields($query_res);
       print "<dl><dd>\n";

       for ($x=0; $x<$num_fields; $x++)
        {  #2
         print "<i>";
         print mysql_field_type($query_res, $x);
            // тип поля
         print "</i> <i>";
         print mysql_field_len($query_res, $x);
            // max-ая длина поля
         print "</i> <b>";
         print mysql_field_name($query_res, $x);
            // имя поля
         print "</b> <i>";
         print mysql_field_flags($query_res, $x);
            // флаги поля (not null и т.п.)
```

```
          print "</i><br>\n";
        }  #2
          print "</dl>\n";
      }  #1
    print "</dl>\n";
mysql_close($conn);
?>
</body> </html>
```

Мы, как обычно, подключаемся к MySQL-серверу и вызываем функцию `mysql_list_tables()`, в которой указываем имя конкретной базы данных возвращает нам идентификатор результата.

Функция `mysql_list_tables()` возвращает идентификатор результата, который мы передаем функции `mysql_fetch_row()` и начинаем цикл, перебирая имена таблиц. Если бы мы просматривали структуру нескольких баз данных, то массив таблиц `$tab_rows` содержал бы значения, соответствующие всем базам данных. В данном же случае, для одной базы массив `$tab_rows` имеет лишь одно значение: `$tab_rows[0]`.

Нами выводится имя таблицы и составляется SQL-запрос `SELECT`, в котором запрашиваются имена полей таблицы. Запрос мы передаем функции `mysql_query()` и получаем еще один идентификатор результата.

Этот идентификатор мы передаем функции `mysql_num_fields()` и получаем количество полей в найденном наборе.

Нами выполняется цикл `for`, в котором перебираются все поля. В переменной `$x` содержится номер текущего поля. Мы вызываем функции, рассмотренные ранее, которые выводят информацию о каждом поле. Эта информация выводится на браузер.

При выполнении программы мы должны получить полный список всех таблиц и характеристики полей, доступных в данном подключении. Ниже показан пример такого вывода.

cust

int 4 **cnum** *not_null primary_key*
string 10 **cname** *not_null*
string 10 **city** *not_null*
int 6 **rating** *not_null*
int 5 **snum**

Примечание. Если вы работаете с MySQL на своем компьютере, то сперва создайте базу данных **`sample`**, а в ней — таблицы, как в файле **`sample.sql`**

Пароль на страницу

Организация входа на страницу только после ввода логина и пароля возможна разными способами. Один из них — ***HTTP-аутентификация в PHP*** (файл **`HTTP-autentif.mht`** в папке **info**).

Другие способы, использующие куки или сессии, правда, для PHP 4 (т.е. без обращения к суперглобальным массивам), приведены здесь:

Пароль на страницу. Части 1–3 (файл **`parole_page-1.mht`** в папке **info**)

Пароль на страницу. Части 4–5 (файл **`parole_page-2.mht`** в папке **info**)

Дополнительную информацию можно, разумеется, найти и в интернете.

12. Работа с файлами

Включение файлов в документ

В каждый PHP-документ можно включить файл с помощью команды `include()`. После этого текст PHP-программы во включаемом файле будет исполняться точно так же, как если бы он был записан во включающем файле непосредственно. Этим удобно пользоваться для одной и той же программы во многих документах.

Предположим, вы написали очень ценную программу, которой можно и нужно пользоваться во многих документах. У вас есть возможность непосредственно вставить текст этой программы в каждый документ, где она необходима. Но в таком случае при обнаружении ошибки в программе или просто в случае ее модификации вам придется вносить изменения в каждом документе, использующем данную программу. Избежать этого позволяет оператор `include()`. Вы можете записать свою программу в один документ, а потом включать этот документ в другие. Функция `include()` требует одного аргумента — пути к файлу, который нужно включить в документ. В листинге 12.1 приведен пример того, как в простую PHP-программу включается содержимое файла, причем содержимое включаемого файла обрабатывается как PHP-программа.

Листинг 12.1. Использование функции `include()`

```
<html> <head>
<title>Листинг 12-1. Использование функции include()
</title> <body>
<?php
  include ("ls12-2.php");
?> </body> </html>
```

Листинг 12.2. Включаемый файл с PHP-программой

```
<?php
print " А меня вставили (;o)<br>";
print "Да еще и считать заставили... 4 + 4 = ".(4+4);
?>
```

Функция `include()` в листинге 12.1 вставляет в документ содержание другого документа, который приведен в листинге 12.2. При выполнении документа из листинга 12.1 строки, содержащиеся во включаемом документе, выводятся на экран. Причем, если вы хотите, чтобы содержимое этого файла обрабатывалось как PHP-программа, вы должны обрамлять его начальным и завершающим тегами PHP.

Включаемые файлы в PHP могут возвращать значения так же, как это делают функции. Использование оператора `return` прерывает выполнение включаемого файла, как это происходит с функцией. В листингах 12.3 и 12.4 приведен пример использования функции `include()`, возвращающей значение.

Листинг 12.3. Использование функции `include()`, возвращающей значение

```
<html> <head>
<title> Листинг 12-3. Использование функции include(),
        возвращающей значение </title> </head> <body>
<?php
$addResult = include("ls12-4.php");
print "Вставленный файл возвращает $addResult";
?>
</body> </html>
```

Листинг 12.4. Включаемый файл, возвращающий значение

```
<?php
$retval = ( 4 + 4 );
return $retval;
?>
```

Команду `include()` можно использовать внутри условного оператора. В таком случае включаемый файл будет выполняться только при выполнении условия.

Например, в следующем фрагменте команда `include()` не будет выполнена.

```
$test = false;
if ($test)
   {
    include("a_file.txt");  // не будет включен
   }
```

Если команду `include()` использовать внутри цикла, то она будет выполняться при каждой итерации. В листинге 12.5 приведен пример того, как команда `include()` используется внутри цикла `for`. Обратите внимание на то, что при каждой итерации включается другой файл.

Листинг 12.5. Использование `include()` внутри цикла

```
<html> <head>
<title> Листинг 12-5. Использование include()
        внутри цикла </title> </head> <body>
<?php
for ($x = 1;  $x<=3;  $x++)
    {
     $incfile = "incfile$x".".txt";
     print "Включаем файл $incfile<br>";
     include("$incfile");
     print "\n<p>";
    }
?>
</body> </html>
```

При выполнении программы из листинга 12.5 в текст страницы будут включены три разных файла: `"incfile1.txt"`, `"incfile2.txt"` и `"incfile3.txt"`. Если предположить, что в каждом файле содержится только объявление его названия, то вывод этой программы будет выглядеть так:

Включаем файл incfile1.txt
incfile1

Включаем файл incfile2.txt
incfile2

Включаем файл incfile3.txt
incfile3

В языке PHP существует команда `require()`, которая действует аналогично команде `include()`. Эту команду тоже можно использовать в циклах, однако возвращать значения она не может.

Исследование файлов

Перед тем как начинать работать с файлом или каталогом, неплохо бы узнать о нем побольше. В языке PHP есть много функций для получения самой разнообразной информации о вашей файловой системе. В этом разделе мы вкратце рассмотрим некоторые из них.

Проверка существования файла

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

```
if (file_exists("test.txt"))
    print "test.txt найден";
```

Файл или каталог?

Иногда нужно убедиться, что исследуемый объект действительно является файлом, а не каталогом. Для этого существует функция `is_file()`. Эта функция требует указания пути к файлу и возвращает булево значение:

```
if (is_file("test.txt"))
    print "test.txt - это файл";
```

Иногда нужно убедиться, что исследуемый объект является каталогом. Для этого существует функция `is_dir()`. Эта функция требует указания пути к каталогу и тоже возвращает булево значение:

```
if (is_dir("/tmp"))
    print "/tmp - это каталог";
```

Проверка статуса файла

После того как файл найден и вы убедились в том, что это именно то, что вам нужно, можете узнать, каков статус этого файла, т.е., другими словами, что с ним можно делать — читать его, или писать в него, или выполнять его. Для этого в PHP есть несколько функций.

Функция `is_readable()` сообщает вам о том, можете ли вы читать указанный файл, В системе UNIX бывают случаи, что вы видите файл, но не имеете возможности читать его содержимое. Функция `is_readable()`

требует строку, содержащую имя и путь к файлу, и возвращает булево значение:

```
if (is_readable("test.txt"))
    print "test.txt можно читать";
```

Функция `is_writable()` сообщает вам о том, существует ли у вас возможность писать в указанный файл. Функция `is_writable()` требует строку, содержащую имя и путь к файлу, и возвращает булево значение:

```
if (is_writable("test.txt"))
    print "в test.txt можно писать";
```

Функция `is_executable()` сообщает вам о том, можете ли вы исполнять указанный файл. Это определяется на основании расширения имени файла или на основании ваших прав доступа к нему в данной операционной системе. Функция `is_executable()` требует строку, содержащую имя и путь к файлу, и возвращает булево значение:

```
if (is_executable("test.txt"))
    print "test.txt можно выполнять";
```

Определение размера файла

Функция `filesize()` определяет размер файла в байтах. Она возвращает значение `false`, если определить размер файла не удается.

```
print "Размер файла test.txt - ";
print filesize("test.txt");
```

Информация о дате и времени

Иногда бывает нужно узнать время создания файла или время его последнего изменения либо чтения. Для этого есть несколько функций.

С помощью функции `fileatime()` можно узнать, когда к файлу последний раз осуществлялся доступ. Функция требует указания имени файла и возвращает дату последнего доступа к нему. Под доступом к файлу понимается чтение или запись в него. Время возвращается в системе эпохи UNIX, т.е. представляет собой количество секунд, прошедших после 1 января 1970 г. В последующих примерах мы преобразуем это число в понятный для человека формат с помощью функции `date()`. Подробнее о работе с датами и об их преобразовании будет рассказано в 14-й главе «Работа с датами».

```
$atime = fileatime("test.txt");
print "Последний раз доступ к test.txt был ";
print date("D d M Y g:i A",  $atime);
// Вывод: Mon 10 Feb 2003 1:30 PM
```

Дату последней модификации файла можно узнать с помощью функции `filemtime()`. Функция требует указания имени файла и возвращает дату последнего изменения содержимого файла.

```
$mtime = filemtime("test.txt");
print "Последний раз test.txt был модифицирован ";
print date("D d M Y g:i A",  $mtime);
// Вывод: Mon 10 Feb 2003 1:30 PM
```

Так же можно узнать дату последнего изменения файла с помощью функции `filectime()`. В системе UNIX дата изменения устанавливается тогда, когда изменяется содержимое файла, или права доступа к нему, или его владелец. В других системах эта функция возвращает дату создания файла.

```
$ctime = filectime("test.txt");
print "Последний раз test.txt был изменен ";
print date("D d M Y g:i A", $ctime);
// Вывод: Mon 10 Feb 2003 1:30 PM
```

Внимание! Все нижеследующие листинги на сервере не функционируют т.к. для этого нужны соответствующие права доступа на каталог. В данном случае их не дали из-за опасности хакерских атак.

Создание и удаление файлов

Если нужный вам файл еще не существует, то его можно создать с помощью функции `touch()`. Получив строку с именем файла, эта функция создает пустой файл с заданным именем. Если такой файл уже существует, то функция не изменяет его содержания, но изменяет дату модификации.

```
touch("myfile.txt");
```

Существующий файл можно удалить с помощью функции `unlink()`. Эта функция тоже получает имя файла.

```
unlink("myfile.txt");
```

Все функции для создания, удаления или модификации файлов требуют, чтобы были правильно установлены права доступа.

Открытие файла для чтения, записи или добавления

Чтобы с файлом можно было работать, его нужно открыть для чтения, или записи, или же для того и другого. Для этого существует функция `fopen()`. Данной функции передаются два аргумента — строка с именем файла и путем к нему, а также строка, описывающая режим открытия файла. Самые обычные

режимы доступа к файлу — это чтение, запись и добавление в конец файла. Соответствующие строки выглядят как `"r"`, `"w"` и `"a"`. Функция `fopen()` возвращает целое число, которое позже используется для доступа к открытому файлу. Это число называется указателем на файл, и его можно присвоить переменной. Для того чтобы открыть файл для чтения, нужно написать такую конструкцию:

```
$fp = fopen("test.txt", "r");
```

Соответственно, для записи следует открывать файл следующим образом:

```
$fp = fopen("test.txt", "w");
```

Для добавления в конец файла функция открытия выглядит так:

```
$fp = fopen("test.txt", "a");
```

Функция `fopen()` возвращает `false`, если открыть файл по какой-то причине не удалось. Поэтому перед тем как начинать работать с файлом, следует проверить значение возвращенного функцией указателя. Сделать это можно таким образом:

```
if ($fp = fopen("test.txt", "w"))
   {// работа с файлом
   }
```

Если открыть файл не удалось, есть возможность прервать выполнение программы:

```
$fp = fopen("test.txt", "w")
 or die ("Не удается открыть файл");
```

Если функция `fopen()` вернет `true`, остальная часть выражения не будет обрабатываться, и функция `die()`, которая просто выводит свой аргумент на экран браузера заканчивает выполнение программы, никогда не будет вызвана. Если `fopen()` вернет `false`, то будет вызвана функция `die()`.

Если все прошло гладко и вы выполнили все необходимые действия с файлом, то по окончании работы его следует закрыть. Для этого существует функция `fclose()`, которой нужно передать указатель на файл, полученный ранее от функции `fopen()`:

```
fclose($fp);
```

Чтение из файла

В языке PHP есть несколько функций для чтения данных из файла.

Чтение строк с помощью `fgets()`

Открыв файл для чтения, вы можете захотеть прочитать из него несколько строк. Для чтения строки из файла существует функция `fgets()`, и ей нужно передать указатель на открытый файл. Кроме того, у этой функции есть второй аргумент, указывающий максимальное число символов, которое можно прочесть из файла до того, как встретится конец строки или файла. Функция `fgets()` будет читать данные из файла до тех пор, пока не встретит символ конца строки или конца файла.

```
$line = fgets($fp, 1024); // $fp - это указатель на файл
```

С помощью описанной функции можно читать строки, но вам нужен способ определить достижение конца файла. Для этого существует функция `feof()`. Данная функция возвращает `true` при достижении конца файла и `false` — в противном случае. Этой функции тоже нужно передавать указатель на файл.

```
feof($fp); // $fp - это указатель на файл
```

Теперь вы уже знаете достаточно для того, чтобы написать программу, читающую данные из файла строка за строкой. Пример такой программы приведен в листинге 12.6.

Листинг 12.6. Открытие файла и чтение из него строк

```
<html> <head>
<title>Листинг 12-6. Открытие файла и чтение из него
       строк </title> </head> <body>
<?php
$filename = "test.txt";
$fp = fopen($filename,"r")
      or die("Нельзя открыть $filename");
while (! feof($fp))
     {
      $line = fgets($fp, 1024);
      print "$line<br>";
     }
?>
</body> </html>
```

Мы открываем файл, указывая его имя в аргументе функции `fopen()`, и используем функцию `die()` для того, чтобы быть уверенными, что в случае невозможности открыть файл выполнение программы будет остановлено. Чаще

всего это происходит в результате того, что файл просто не существует, но в системе UNIX это может произойти и по той причине, что права доступа к файлу не позволяют вашей программе читать из него. Чтение файла происходит в цикле `while`, в проверочном выражении которого мы используем функцию `feof()` для проверки того, что конец файла еще не достигнут. Другими словами, цикл выполняется до тех пор, пока не будет достигнут конец файла. Внутри цикла нами с помощью функции `fgets()` читается строка или 1024 байта, если конец строки не достигнут. Мы присваиваем прочитанную строку переменной `$line` и выводим ее на экран браузера, добавляя тег `<br>` для удобства чтения.

Чтение произвольного количества символов с помощью функции `fread()`

Можно читать данные из файла не по строкам, а кусками произвольного размера. Для этого существует функция `fread()`. Эта функция имеет два аргумента — указатель на файл и количество символов, которое нужно прочесть. Возвращает данная функция количество символов, которое ей фактически удалось прочесть. Это число может не совпасть с затребованным при достижении конца файла или в других случаях.

```
$var = fread($fp, 16);
```

В листинге 12.7 приведена измененная программа, читающая данные из файла не по строкам, а порциями по 16 байт.

Листинг 12.7. Чтение файла функцией `fread()`

```
<html> <head>
<title> Листинг 12-7. Чтение файла функцией
        fread()</title> </head> <body>
<?php
$filename = "test.txt";
$fp = fopen($filename,"r")
      or die("Нельзя открыть $filename");
while (! feof($fp))
    {
     $var = fread($fp,16);
     print "$var<br>";
     }
?>
</body> </html>
```

Функция `fread()` позволяет вам указать, сколько символов нужно прочесть из файла, но она не позволяет выбирать, с какого места начинать чтение. Для этого существует функция `fseek()`. Данная функция имеет два аргумента — указатель на файл и количество символов, на которое нужно отступить от начала файла.

Это число называют смещением.

```
fseek($fp,64);
```

Чтение отдельных символов с помощью функции `fgetc()`

Функция `fgetc()` работает почти так же, как функция `fread()`, разница заключается в том, что она читает каждый раз по одному символу. Поэтому ей не нужен второй аргумент. Вы просто передаете указатель на файл.

```
$char = fgetc($fp);
```

Запись в файл

Для записи в файл существуют два разных режима. Разница между ними заключается в способе вызова функции `fopen()`. Если вы откроете файл в простом режиме записи, т е. запишите функцию так:

```
fopen("test.txt", "w"),
```

то в существующем файле вся информация будет уничтожена и новые данные записаны в начало файла. Если такого файла не существует, то он будет создан. Если вы откроете файл в режиме добавления, т.е. запишите функцию так:

```
fopen("test.txt", "a"),
```

то все новые данные будут добавлены в конец файла.

Запись в файл с помощью `fwrite()`

У функции `fwrite()` есть два аргумента — указатель на файл и строка. Функция `fwrite()` просто записывает строку в файл.

В листинге 12.8 приведен пример использования этой функции.

Листинг 12.8. Запись в файл и добавление в его конец

```
<html> <head>
<title> Листинг 12-8. Запись в файл и добавление
        в его конец </title> </head> <body>
<?php
$filename = "test.txt";
 $fp = fopen($filename,"w")
      or die("Нельзя открыть $filename");
if (is_writable("test.txt")) { #1
print "Пишем в $filename<br>";
fwrite($fp, "Привет всем!\n");
fclose($fp);
$fp = fopen($filename,"a")
      or die("Нельзя открыть $filename");
print "Добавляем в конец $filename<br>";
fwrite($fp,"Еще дописали :-)\n");
} #1
fclose($fp);
?>
</body> </html>
```

Блокировка файла

Все, что мы говорили о записи в файл ранее, действует прекрасно, но только до тех пор, пока с вашей программой работает один-единственный пользователь. Но в реальной жизни нужно быть готовым к тому, что к вашей программе могут обратиться несколько пользователей, причем практически одновременно. Представьте себе, что произойдет, если несколько человек или программ, что в данном случае одно и то же, будут писать в один и тот же файл одновременно?! Информация в этом файле разрушится.

В PHP есть специальная функция, которая предназначена для предотвращения подобных ситуаций. Функция `flock()` блокирует файл, т.е. не позволяет другим пользователям читать этот файл или писать в него до тех пор, пока процесс, наложивший блокировку, не закончит работу с данным файлом. Эта функция имеет два аргумента — указатель на файл и целое число, указывающее режим блокировки. В таблице 12.1 приведены режимы блокировки, которые вы можете применить к файлу.

Таблица 12.1 — Режимы блокировки функции `flock()`

Номер режима	Тип блокировки	Описание
1	Частичная	Разрешает другим процессам читать файл, но запрещает запись в него
2	Полная	Запрещает другим процессам как чтение, так и запись в файл
3	Освобождение	Снимает блокировку с файла

Функцию `flock()` нужно вызывать сразу после открытия файла и потом вызывать ее повторно для освобождения.

```
$fp = fopen("test.txt","a");
flock($fp, 2);  // Полная блокировка
// Запись чего-то в файл
flock($fp, 3);  // Освобождение файла
fclose($fp);
```

Блокировка с помощью функции `flock()` не является абсолютной. С ней будут считаться только те программы, которые тоже пользуются этой функцией.

Работа с каталогами

Теперь, когда вы умеете проверять существование файла, можете читать его или писать в него, обратимся к работе с каталогами. В языке PHP есть несколько функций для работы с каталогами, с помощью которых вы можете создавать каталоги, удалять их или читать.

Создание каталога

Для создания каталога предназначена функция `mkdir()`. Она имеет два аргумента — строку, содержащую путь и имя нового каталога, и целое восьмеричное число, описывающее режим создания каталога. Этот аргумент оказывает влияние только в операционной системе UNIX. Восьмеричное число, описывающее режим создания, должно начинаться с нуля, а затем идут три цифры (каждая от 0 до 7), которые представляют собой права доступа (чтения — `r`; записи — `w`; исполнения программ — `x`) для владельца, группы и для всех остальных. Примеры задания прав доступа приведены в таблице 12.2.

Таблица 12.2 — Задание прав доступа к каталогу

	владелец	группа	остальные
Права доступа	r w x	r w x	r w x
Что разрешается (в виде двоичного числа: 1 – разрешено, 0 – запрещено)	1 1 1	1 1 1	1 1 1
Восьмеричное число	7	7	7
Двоичное число	1 1 1	1 0 1	1 0 0
Восьмеричное число	7	5	4

Функция `mkdir()` возвращает `true` в случае успеха и `false` — в противном случае. Неудача при создании каталога может быть вызвана в первую очередь тем, что ваша программа не имеет прав писать в тот каталог, в котором вы пытаетесь создать новый.

```
mkdir("testdir", 0777); // "разрешено всё всем"
chdir("testdir");       // переход в директорию testdir
```

Удаление каталога

Для удаления каталога из файловой системы предназначена функция `rmdir()`. Удалить каталог можно только в том случае, если ваша программа имеет на это право и если каталог пуст. У данной функции есть только один аргумент — имя и путь к удаляемому каталогу.

```
rmdir("testdir");
```

Открытие каталога для чтения

Перед тем как читать содержимое каталога, его нужно открыть и получить указатель на него с помощью функции `opendir()`. Эта функция имеет только один аргумент — имя и путь к каталогу. Она возвращает `true`, если каталог был успешно открыт, и `false` — в противном случае. Ошибка при открытии может быть вызвана тем, что каталог не существует или ваша программа не имеет права его читать.

```
$fp = opendir("testdir");
```

Чтение каталога

Точно так же, как вы читали строки из файла с помощью функции `fgets()`, у вас есть возможность читать в каталоге имена файлов и подкаталогов с помощью функции `readdir()`. Эта функция имеет один аргумент — имя каталога — и возвращает строку, содержащую имя найденного объекта, т.е. файла или подкаталога. По достижении конца файла она возвращает `false`. Имейте в виду, что функция `readdir()` возвращает имена объектов, но не пути к ним. В листинге 12.9 приведен пример программы, читающей содержимое каталога.

Листинг 12.9. Чтение каталога с помощью функции `readdir()`

```
<html> <head>
<title> Листинг 12-9. Чтение каталога с помощью
        функции readdir()</title> </head> <body>
<?php
$dirname = "testdir";
$dh = opendir($dirname);
while (gettype($file = readdir($dh)) != boolean)
      {
       if (is_dir("$dirname/$file")) print "(D)";
       print "$file<br>";
      }
closedir($dh);
?>
</body> </html>
```

Нами открывается каталог и в цикле читается его содержание. Мы вызываем функцию `readdir()` в проверочном условии цикла, присваивая возвращенное значение переменной `$file`. В теле цикла мы объединяем переменные `$dirname` и `$file`, получая полный путь, который потом проверяем. Если этот путь ведет к каталогу, то выводим букву `D` перед его именем. Потом печатаем переменную `$file`. В проверочном выражении цикла мы используем конструкцию, в которой приняты дополнительные меры безопасности.

Значение, возвращенное функцией `readdir()`, проверяется. До тех пор, пока строка, отличная от 0, превращается в значение `true`, все проходит гладко. Однако представьте себе, что в каталоге есть файлы с именами `"0"`, `"1"`, `"2"` и `"3"`. В таком случае имя файла `"0"` будет преобразовано в 0, и это вызовет завершение цикла. В листинге 12.9 такая ситуация исправлена тем, что мы проверяем каждое значение, возвращенное функцией `readdir()`.

Формы и программы для передачи файлов на сервер

До сих пор мы рассматривали формы, передающие довольно простые данные. Однако современные браузеры, в том числе Netscape и Explorer, позволяют передавать файлы на сервер, и, разумеется, такую передачу поддерживает PHP. В этом разделе мы рассмотрим средства PHP для передачи файлов на сервер.

В первую очередь нам нужно создать соответствующую форму. HTML-форма, предназначенная для копирования файлов и имеющая соответствующее поле, должна содержать аргумент `ENCTYPE`:

```
ENCTYPE = "multipart/form-data"
```

Кроме того, PHP требует, чтобы в форме перед полем для копирования файлов находилось скрытое поле. Такое скрытое поле должно называться `max_file_size` и в нем должен быть записан максимальный размер файла (в байтах), который вы разрешаете передавать. Этот размер не должен превышать размер, установленный в поле `upload_max_filesize` в файле `php.ini`, который обычно равен 2 мегабайтам. Теперь можно подготовить само поле для передачи файла. Это обычный элемент `INPUT`, у которого в атрибуте `type` записано `"file"`. Имя ему вы можете выбрать по своему усмотрению. Все это продемонстрировано в листинге 12.10.

Листинг 12.10. Простая форма для передачи файла

```
<html> <head>
<title> Листинг 12-10. Простая форма для передачи файла
</title> </head> <body>
<form ENCTYPE="multipart/form-data"
      action="<?=$_SERVER['PHP_SELF'] ?>"
      method="POST">
<input type="hidden" name="MAX_FILE_SIZE" value="51200">
<input type="file" name="fupload"><br>
<input type="submit" value="Передать!">
</form>
</body> </html>
```

Обратите внимание на то, что опять эта форма вызывает страницу, в которой она записана. Так нужно сделать потому, что мы сейчас добавим PHP-программу, которая будет обрабатывать переданный файл. Мы сделали ограничение на размер файла в 50 килобайт (51200 байт) и назвали поле `"fupload"`. Как вы понимаете, этим именем нам скоро придется воспользоваться.

После того как файл передан на сервер, он получает уникальное имя и сохраняется в каталоге для временных файлов. На UNIX-системе это обычно каталог `/tmp`. Полный путь к файлу записывается в глобальную переменную с именем, совпадающим с именем поля для передачи этого файла (в данном случае это будет `$fupload`).

PHP сохраняет еще некоторую дополнительную информацию о переданном файле в глобальных переменных. Их имена произведены от имени поля для передачи файла, за которым следует символ подчеркивания, а потом строки `"name"`, `"size"` и `"type"`. В таблице 12.3 приведены описания этих переменных.

Таблица 12.3 — Глобальные переменные, описывающие переданный файл

Имя переменной	Описание	Пример
`$fupload`	Путь к временному файлу	`/tmp`
`$fupload_name`	Имя переданного файла	`test.gif`
`$fupload_size`	Размер переданного файла в байтах	`6835`
`$fupload_type`	Тип переданного файла в системе MIME	`image/gif`

Кроме того, в PHP есть встроенная переменная, в которой в виде массива записана информация о переданном файле. Если формой было передано несколько файлов, то в переменной `$HTTP_POST_FILES` будет содержаться массив, упорядоченный по именам полей формы. Каждый из элементов этого массива, в свою очередь, является ассоциативным массивом. Элементы таких массивов описаны в таблице 12.4.

Таблица 12.4 — Элементы массива, описывающие переданный файл

Имя переменной	Описание	Пример
`$HTTP_POST_FILES[fupload][name]`	Имя переданного файла	`test.gif`
`$HTTP_POST_FILES[fupload][size]`	Размер переданного файла в байтах	`6835`
`$HTTP_POST_FILES[fupload][type]`	Тип переданного файла в системе MIME	`image/gif`

Вооруженные всеми этими знаниями, мы можем написать несложную программу, выводящую информацию о переданном файле. Эта программа приведена в листинге 12.11. Если переданный файл является файлом рисунка в формате GIF, то программа даже выводит его на экран браузера.

Листинг 12.11. Программа обработки переданного файла

```
<html> <head>
<title> Листинг 12-11. Программа обработки переданного
        файла </title> </head>
<?php
$file_dir = "/tmp/uploads";
$file_url = "http://www.primer.ru/uploads";
if (isset($fupload))
   { #1
    print "<p>Путь:   $fupload\n";
    print "<p>Имя:    $fupload_name\n";
    print "<p>Размер: $fupload_size байт\n";
    print "<p>Тип:    $fupload_type<p>\n";
    if ($fupload_type == "image/gif")
       { #2
        copy($fupload, "$file_dir/$fupload_name")
        or die("Нельзя копировать");
        print "<img src=\"$file_url/$fupload_name\">\n";
       } #2
   } #1
?>
<body>
<form enctype="multipart/form-data"
      action="<?=$_SERVER['PHP_SELF'] ?>" method="POST">
<p>
<input type="hidden" name="MAX_FlLE_SIZE" value="51200">
<p>
<input type="file" name="fupload">
<p>
<input type="submit" value="Отправить файл">
</form>
</body> </html>
```

Сначала мы проверяем, определена ли переменная `$fupload`. Если она определена, то можем считать, что файл передан.

Никогда не надейтесь, во всяком случае всерьез не рассчитывайте на то, что ваша программа получает данные именно из той формы, из которой вы ожидаете, или что она получает данные в точности того типа, для которого вы ее спроектировали. В программе из листинга 12.11 мы сделали допущение на основе того, что имя переменной совпадает с именем поля из нашей формы. На самом деле ничто не может помешать недоброжелательному пользователю создать собственную форму и послать из нее данные нашей программе. В этой подделанной форме может быть поле с таким же именем, но с данными совсем другой природы. Не доверяйте данным, приходящим извне, даже если на первый взгляд кажется, что они пришли из вашей собственной формы.

Если файл передан, то путь к этому файлу на сервере записан в переменной `$fupload` и мы выводим этот путь на экран браузера. Кроме того, мы выводим имя файла, записанное в переменной `$fupload_name`, размер файла, который записан в переменной `$fupload_size`, и тип файла, записанный в переменной `$fupload_type`.

После этого проверяется значение переменной `$fupload_type`. Если оно совпадает с `"image/gif"`, мы считаем, что это GIF-файл. На самом деле было бы неплохо проверить тот случай, когда совпадения не происходит. Кроме того, мы считаем, что если файл имеет расширение GIF, то в нем обязательно содержится рисунок типа GIF, а это, строго говоря, не обязательно так, но пока для простоты опустим все эти проверки.

Решив, что имеем дело с рисунком типа GIF, мы копируем его из временного каталога в наш каталог. Функция `copy()` требует двух аргументов — путь к исходному файлу и новое его положение. Эта функция возвращает `true`, если файл скопирован успешно. Исходное размещение и имя файла записано у нас в переменной `$fupload`, а новое положение — в переменной `$file_dir`. Нам приходится объединять новое положение файла и его исходное имя для того, чтобы сформировать второй аргумент функции `copy()`. В результате файл в новом месте получает свое первоначальное имя. Имейте в виду, что в системе UNIX сервер обычно работает с минимальными правами, поэтому для того, чтобы программа могла скопировать файл и переименовать его, нужно проверить, имеет ли она на это право.

После того как файл скопирован, вам не обязательно удалять исходный. Это сделает за вас PHP.