*Лекция 11: Взаимодействие PHP и MySQL*

### **Построение интерфейса для добавления информации**

Итак, у нас есть какая-то *таблица* в базе данных. Чтобы построить *интерфейс* для добавления информации в эту таблицу, нужно ее структуру (т.е. набор ее полей) отобразить в *html-форму*.

Разобьем эту задачу на следующие подзадачи:

* *установка соединения с БД* ;
* *выбор рабочей БД* ;
* *получение списка полей таблицы* ;
* отображение полей в *html-форму*.

После этого данные, введенные в форму, нужно записать в базу данных. Рассмотрим все эти задачи по порядку.

#### **Установка соединения**

Итак, первое, что нужно сделать, - это установить соединение с базой данных. Воспользуемся функцией *mysql\_connect*.

**Синтаксис*****mysql\_connect***

ресурс mysql\_connect ( [строка server

[, строка username [, строка password

[, логическое new\_link

[, целое client\_flags]]]]])

Данная функция устанавливает соединение с сервером *MySQL* и возвращает указатель на это соединение или FALSE в случае неудачи. Для отсутствующих параметров устанавливаются следующие значения по умолчанию:

server = 'localhost:3306'

username = имя пользователя владельца

процесса сервера

password = пустой пароль

Если функция вызывается дважды с одними и теми же параметрами, то новое соединение не устанавливается, а возвращается ссылка на старое соединение. Чтобы этого избежать, используют параметр new\_link, который заставляет в любом случае открыть еще одно соединение.

Параметр client\_flags - это комбинация следующих констант: *MYSQL\_CLIENT\_COMPRESS* (использовать протокол сжатия), *MYSQL\_CLIENT\_IGNORE\_SPACE* (позволяет вставлять пробелы после имен функций), *MYSQL\_CLIENT\_INTERACTIVE* (ждать interactive\_timeout секунд - вместо wait\_timeout - до закрытия соединения).

Параметр new\_link появился в PHP 4.2.0, а параметр client\_flags - в PHP 4.3.0.

Соединение с сервером закрывается при завершении исполнения скрипта, если оно до этого не было закрыто с помощью функции *mysql\_close()*.

Итак, устанавливаем соединение с базой данных на локальном сервере для пользователя nina с паролем "123":

<?

$conn = mysql\_connect(

"localhost", "nina","123")

or die("Невозможно установить

соединение: ". mysql\_error());

echo "Соединение установлено";

mysql\_close($conn);

?>

Действие *mysql\_connect* равносильно команде

shell>mysql -u nina -p123

#### **Выбор базы данных**

После *установки соединения* нужно выбрать базу данных, с которой будем работать. Наши данные хранятся в базе данных book. В *MySQL**выбор базы данных* осуществляется с помощью команды use:

mysql>use book;

В PHP для этого существует функция *mysql\_select\_db*.

**Синтаксис*****mysql\_select\_db*:**

логическое mysql\_select\_db (

строка database\_name

[, ресурс link\_identifier])

Эта функция возвращает TRUE в случае успешного *выбора базы данных* и FALSE - в противном случае.

Сделаем базу данных book рабочей:

<?

$conn = mysql\_connect(

"localhost","nina","123")

or die("Невозможно установить

соединение: ". mysql\_error());

echo "Соединение установлено";

mysql\_select\_db("book");

?>

#### **Получение списка полей таблицы**

Теперь можно заняться собственно решением задачи. Как получить список полей таблицы? Очень просто. В PHP и на этот случай есть своя команда - mysql\_list\_fields.

Синтаксис mysql\_list\_fields

ресурс mysql\_list\_fields (

строка database\_name,

строка table\_name

[, ресурс link\_identifier])

Эта функция возвращает список полей в таблице table\_name в базе данных database\_name. Получается, что выбирать базу данных нам было необязательно, но это пригодится позже. Как можно заметить, результат работы этой функции - переменная типа ресурс. То есть это не совсем то, что мы хотели получить. Это ссылка, которую можно использовать для получения информации о полях таблицы, включая их названия, типы и флаги.

Функция *mysql\_field\_name* возвращает имя поля, полученного в результате выполнения запроса. Функция *mysql\_field\_len* возвращает длину поля. Функция *mysql\_field\_type* возвращает тип поля, а функция *mysql\_field\_flags* возвращает список флагов поля, записанных через пробел. Типы поля могут быть int, real, string, *blob* и т.д. Флаги могут быть not\_null, primary\_key, unique\_key, *blob*, auto\_increment и т.д.

Синтаксис у всех этих команд одинаков:

строка mysql\_field\_name (

ресурс result, целое field\_offset)

строка mysql\_field\_type (

ресурс result, целое field\_offset)

строка mysql\_field\_flags (

ресурс result, целое field\_offset)

строка mysql\_field\_len (

ресурс result, целое field\_offset)

Здесь result - это идентификатор результата запроса (например, запроса, отправленного функциями mysql\_list\_fields или *mysql\_query* (о ней будет рассказано позднее)), а field\_offset - порядковый номер поля в результате.

Вообще говоря, то, что возвращают функции типа mysql\_list\_fields или *mysql\_query*, представляет собой таблицу, а точнее, указатель на нее. Чтобы получить из этой таблицы конкретные значения, нужно задействовать специальные функции, которые построчно читают эту таблицу. К таким функциям и относятся *mysql\_field\_name* и т.п. Чтобы перебрать все строки в таблице результата выполнения запроса, нужно знать число строк в этой таблице. Команда *mysql\_num\_rows* (ресурс result ) возвращает число строк во множестве результатов result.

А теперь попробуем получить список полей таблицы *Artifacts* (коллекция экспонатов).

<?

$conn = mysql\_connect(

"localhost","nina","123")

or die("Невозможно установить

соединение: ". mysql\_error());

echo "Соединение установлено";

mysql\_select\_db("book");

$list\_f = mysql\_list\_fields (

"book","Artifacts",$conn);

$n = mysql\_num\_fields($list\_f);

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

$type = mysql\_field\_type($list\_f, $i);

$name\_f = mysql\_field\_name($list\_f,$i);

$len = mysql\_field\_len($list\_f, $i);

$flags\_str = mysql\_field\_flags (

$list\_f, $i);

echo "<br>Имя поля: ". $name\_f;

echo "<br>Тип поля: ". $type;

echo "<br>Длина поля: ". $len;

echo "<br>Строка флагов поля: ".

$flags\_str . "<hr>";

}

?>

В результате должно получиться примерно вот что (если в таблице всего два поля, конечно):

Имя поля: id

Тип поля: int

Длина поля: 11

Строка флагов поля:

not\_null primary\_key auto\_increment

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Имя поля: title

Тип поля: string

Длина поля: 255

Строка флагов поля:

#### **Отображение списка полей в html-форму**

Теперь немножко подкорректируем предыдущий пример. Будем не просто выводить информацию о поле, а отображать его в подходящий элемент *html-формы*. Так, элементы типа *BLOB* переведем в textarea (заметим, что поле description, которое мы создавали с типом TEXT, отображается как имеющее тип *BLOB* ), числа и строки отобразим в текстовые строки ввода <input type=text>, а элемент, имеющий метку автоинкремента, вообще не будем отображать, поскольку его значение устанавливается автоматически.

Все это решается довольно просто, за исключением выделения из списка флагов флага auto\_increment. Для этого нужно воспользоваться функцией *explode*.

Синтаксис *explode*:

массив explode( строка separator,

строка string [, int limit])

Эта функция разбивает строку string на части с помощью разделителя separator и возвращает массив полученных строк.

В нашем случае в качестве разделителя нужно взять пробел " ", а в качестве исходной строки для разбиения - строку флагов поля.

Итак, создадим форму для ввода данных в таблицу *Artifacts*:

<?

$conn=mysql\_connect("localhost","nina","123");

// устанавливаем соединение

$database = "book";

$table\_name = "Artifacts";

mysql\_select\_db($database); // выбираем базу данных для

// работы

$list\_f = mysql\_list\_fields($database,$table\_name);

// получаем список полей в таблице

$n = mysql\_num\_fields($list\_f); // число строк в результате

// предыдущего запроса (т.е. сколько всего

// полей в таблице Artifacts)

echo "<form method=post action=insert.php>";

// создаем форму для ввода данных

echo "&nbsp;<TABLE BORDER=0 CELLSPACING=0 width=50% ><tr>

<TD BGCOLOR='#005533' align=center><font color='#FFFFFF'>

<b> Add new row in $table\_name</b></font></td></tr><tr><td></td></tr></TABLE>";

echo "<table border=0 CELLSPACING=1 cellpadding=0 width=50% >";

// для каждого поля получаем его имя, тип, длину и флаги

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

$type = mysql\_field\_type($list\_f, $i);

$name\_f = mysql\_field\_name ($list\_f,$i);

$len = mysql\_field\_len($list\_f, $i);

$flags\_str = mysql\_field\_flags ($list\_f, $i);

// из строки флагов делаем массив,

// где каждый элемент массива - флаг поля

$flags = explode(" ", $flags\_str);

foreach ($flags as $f){

if ($f == 'auto\_increment') $key = $name\_f;

// запоминаем имя автоинкремента

}

/\* для каждого поля, не являющегося автоинкрементом, в

зависимости от его типа выводим подходящий элемент формы \*/

if ($key <> $name\_f){

echo "<tr><td align=right bgcolor='#C2E3B6'><font size=2>

<b>&nbsp;". $name\_f ."</b></font></td>";

switch ($type){

case "string":

$w = $len/5;

echo "<td><input type=text name=\"$name\_f\"

size = $w ></td>";

break;

case "int":

$w = $len/4;

echo "<td><input type=text name=\"$name\_f\"

size = $w ></td>";

break;

case "blob":

echo "<td><textarea rows=6 cols=60 name=\"$name\_f\"></textarea></td>";

break;

}

}

echo "</tr>";

}

echo "</table>";

echo "<input type=submit name='add' value='Add'>";

echo "</form>";

?>

Листинг 11.0.1. Форма для ввода данных в таблицу Artifacts

#### **Запись данных в базу данных**

Итак, форма создана. Теперь нужно сделать самое главное - отправить данные из этой формы в нашу базу данных. Как вы уже знаете, для того чтобы записать данные в таблицу, используется команда INSERT языка SQL. Например:

mysql> INSERT INTO Artifacts

SET title='Петров';

Возникает вопрос, как можно воспользоваться такой командой (или любой другой командой SQL) в PHP скрипте. Для этого существует функция *mysql\_query()*.

Синтаксис *mysql\_query*

ресурс mysql\_query ( строка query

[, ресурс link\_identifier])

*mysql\_query()* посылает SQL-запрос *активной базе данных* *MySQL* сервера, который определяется с помощью указателя link\_identifier (это ссылка на какое-то соединение с сервером *MySQL* ). Если параметр link\_identifier опущен, используется последнее открытое соединение. Если открытые соединения отсутствуют, функция пытается соединиться с СУБД, аналогично функции *mysql\_connect()* без параметров. Результат запроса буферизируется.

**Замечание**: *строка запроса* НЕ должна заканчиваться точкой с запятой.

Только для запросов SELECT, SHOW, *EXPLAIN*, *DESCRIBE*, *mysql\_query()* возвращает указатель на результат запроса, или FALSE, если запрос не был выполнен. В остальных случаях *mysql\_query()* возвращает TRUE, если запрос выполнен успешно, и FALSE - в случае ошибки. Значение, не равное FALSE, говорит о том, что запрос был выполнен успешно. Оно не говорит о количестве затронутых или возвращенных рядов. Вполне возможна ситуация, когда успешный запрос не затронет ни одного ряда. *mysql\_query()*также считается ошибочным и вернет FALSE, если у пользователя недостаточно прав для работы с указанной в запросе таблицей.

Итак, теперь мы знаем, как отправить запрос на вставку строк в базу данных. Заметим, что в предыдущем примере элементы формы мы назвали именами полей таблицы. Поэтому они будут доступны в скрипте insert.php, обрабатывающем данные формы, как переменные вида $*\_POST*['имя\_поля'].

<?

$conn=mysql\_connect("localhost","nina","123");// устанавливаем

// соединение

$database = "book";

$table\_name = "Artifacts";

mysql\_select\_db($database); // выбираем базу данных

$list\_f = mysql\_list\_fields($database,$table\_name);

// получаем список полей в таблице

$n = mysql\_num\_fields($list\_f); // число строк в результате

// предыдущего запроса

// составим один запрос сразу для всех полей таблицы

$sql = "INSERT INTO $table\_name SET "; // начинаем создавать

// запрос, перебираем все поля таблицы

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

$name\_f = mysql\_field\_name ($list\_f,$i); // вычисляем имя поля

$value = $\_POST[$name\_f]; // вычисляем значение поля

$j = $i + 1;

$sql = $sql . $name\_f." = '$value'"; // дописываем в

// строку $sql пару имя=значение

if ($j <> $n) $sql = $sql . ", "; // если поле не

// последнее в списке, то ставим запятую

}

// перед тем как записывать что-то в базу,

// можно посмотреть, какой запрос получился

//echo $sql;

$result = mysql\_query($sql,$conn); // отправляем запрос

// выводим сообщение успешно ли выполнен запрос

if (!$result) echo " Can't add ($table\_name) ";

else echo "Success!<br>";

?>

Листинг 11.0.2. insert.php

Итак, задачу добавления данных с помощью web-интерфейса мы решили. Однако тут есть одна тонкость. При решении мы не учитывали тот факт, что значения некоторых полей ( author, photo ) должны браться из других таблиц ( Persons, Images ). Поскольку *MySQL* с внешними ключами не работает (уже работает - прим. эксперта), этот момент остается на совести разработчиков системы, т.е. на нашей совести. Нужно дописать программу таким образом, чтобы была возможность вводить в такие поля правильные значения. Но мы делать этого не будем, поскольку задача лекции состоит в том, чтобы познакомить читателя с элементами технологии, а не в том, чтобы создать работающую систему. Кроме того, имеющихся у читателя знаний вполне достаточно, чтобы решить эту проблему самостоятельно. Мы же обратимся к другой задаче - отображение данных, хранящихся в базе данных СУБД *MySQL*.

### **Отображение данных, хранящихся в MySQL**

Чтобы отобразить какие-то данные в браузере с помощью *PHP*, нужно сначала получить эти данные в виде переменных *PHP*. При работе с *MySQL* без посредника (такого, как *PHP*) *выборка* данных производится с помощью команды SELECT языка *SQL*:

mysql> SELECT \* FROM Artifacts;

В предыдущей главе мы говорили, что любой *запрос*, в том числе и на выборку, можно отправить на *сервер* с помощью функции *mysql\_query()* ; Там у нас стояла немного другая задача - получить данные из формы и отправить их с помощью запроса на вставку в базу данных. Результатом работы *mysql\_query()* там могло быть только одно из выражений, TRUE или FALSE. Теперь же требуется отправить *запрос* на выбор всех полей, а результат отобразить в браузере. И здесь результат - это целая *таблица* значений, а точнее, *указатель* на эту таблицу. Так что нужны какие-то аналоги функции *mysql\_field\_name()*, только чтобы они извлекали из результата запроса не имя, а *значение* поля. Таких функций в *PHP* несколько. Наиболее популярные - *mysql\_result()* и *mysql\_fetch\_array()*.

***Синтаксис******mysql\_result***

смешанное mysql\_result (ресурс result,

целое row [, смешанное field])

*mysql\_result()* возвращает *значение* одной ячейки результата запроса. *Аргумент* field может быть порядковым номером поля в результате, именем поля или именем поля с именем таблицы через точку tablename.*fieldname*. Если для имени поля в запросе применялся *алиас* ('select foo as bar from...'), используйте его вместо реального имени поля.

Работая с большими результатами запросов, следует задействовать одну из функций, обрабатывающих сразу *целый* ряд результата (например, *mysql\_fetch\_row()*, *mysql\_fetch\_array()* и т.д.). Так как эти функции возвращают *значение* нескольких ячеек сразу, они НАМНОГО быстрее *mysql\_result()*. Кроме того, нужно учесть, что указание численного смещения (номера поля) работает намного быстрее, чем указание колонки или колонки и таблицы через точку.

Вызовы функции *mysql\_result()* не должны смешиваться с другими функциями, работающими с результатом запроса.

***Синтаксис******mysql\_fetch\_array***

массив mysql\_fetch\_array ( ресурс result

[, целое result\_type])

Эта *функция* обрабатывает ряд результата запроса, возвращая *массив* (ассоциативный, численный или оба) с обработанным рядом результата запроса, или FALSE, если рядов больше нет.

***mysql\_fetch\_array()*** - это расширенная версия функции *mysql\_fetch\_row()*. Помимо хранения значений в массиве с численными индексами, функция возвращает значения в массиве с индексами по названию колонок.

Если несколько колонок в результате будут иметь одинаковые названия, будет возвращена последняя колонка. Чтобы получить *доступ*к первым, следует использовать численные *индексы массива* или *алиасы* в запросе. В случае *алиасов* именно их вы не сможете использовать в именах колонок, как, например, не сможете использовать "photo" в описанном ниже примере.

select Artifacts.photo as art\_image,

Persons.photo as pers\_image

from Artifacts, Persons

11.1. Запрос с дублирующимися именами колонок

Важно заметить, что *mysql\_fetch\_array()* работает НЕ медленнее, чем *mysql\_fetch\_row()*, и предоставляет более удобный *доступ* к данным.

Второй опциональный *аргумент* result\_type в функции *mysql\_fetch\_array()* является константой и может принимать следующие значения: MYSQL\_ASSOC, MYSQL\_NUM и MYSQL\_BOTH. Эта возможность добавлена в *PHP* 3.0.7. Значением по умолчанию является: MYSQL\_BOTH.

Используя MYSQL\_BOTH, получим *массив*, состоящий как из ассоциативных индексов, так и из численных. MYSQL\_ASSOC вернет только ассоциативные соответствия, а MYSQL\_NUM - только численные.

**Замечание**: имена полей, возвращаемые этой функцией, регистрозависимы.

Теперь отобразим данные из *Artifacts* в виде таблицы в браузере:

<?

/ \* сначала делаем то же, что и раньше: устанавливаем

соединение, выбираем базу и получаем список и число полей в таблице Artifacts \*/

$conn=mysql\_connect("localhost","nina","123");

$database = "book";

$table\_name = "Artifacts";

mysql\_select\_db($database);

$list\_f = mysql\_list\_fields($database,$table\_name);

$n1 = mysql\_num\_fields($list\_f);

// сохраним имена полей в массиве $names

for($j=0;$j<$n1; $j++){

$names[] = mysql\_field\_name ($list\_f,$j);

}

$sql = "SELECT \* FROM $table\_name"; // создаем SQL запрос

$q = mysql\_query($sql,$conn) or die(); // отправляем

// запрос на сервер

$n = mysql\_num\_rows($q); // получаем число строк результата

//рисуем HTML-таблицу

echo "&nbsp;<TABLE BORDER=0 CELLSPACING=0 width=90%

align=center><tr><TD BGCOLOR='#005533' align=center>

<font color='#FFFFFF'><b>$table\_name</b></font></td>

</tr></TABLE>";

echo "<table cellspacing=0 cellpadding=1 border=1

width=90% align=center>";

// отображаем названия полей

echo "<tr>";

foreach ($names as $val){

echo "<th ALIGN=CENTER BGCOLOR='#C2E3B6'>

<font size=2>$val</font></th>";

}

// отображаем значения полей

echo "</tr>";

for($i=0;$i<$n; $i++){ // перебираем все строки в

// результате запроса на выборку

echo "<tr>";

foreach ($names as $val) { // перебираем все

// имена полей

$value = mysql\_result($q,$i,$val); // получаем

// значение поля

echo "<td><font size=2>&nbsp;$value</font></td>";

// выводим значение поля

}

echo "</tr>";

}

echo "</table>";

Листинг 11.1.1. Отображение данных из Artifacts в виде таблицы в браузере

Сделаем то же самое с помощью *mysql\_fetch\_array()*:

<?

/\* ... начало то же, что и в предыдущем примере \*/

// отображаем значения полей

// получаем значение поля в виде ассоциативного массива

while($row = mysql\_fetch\_array($q, MYSQL\_ASSOC)) {

echo "<tr>";

foreach ($names as $val){

echo "<td><font size=2>&nbsp;$row[$val]</font></td>";

// выводим значение поля

}

echo "</tr>";

}

echo "</table>";

?>

Листинг 11.1.2. Отображение данных из Artifacts в виде таблицы в браузере. Вариант 2

### ***Заключение***

В этой лекции мы решили две задачи: добавление данных в базу данных и их *отображение* в браузере с помощью языка *PHP*. Для этого мы рассмотрели ряд функций, которые позволяют отправлять *SQL*-запросы к базе данных и обрабатывать полученные ответы. Используя приведенную здесь технологию, можно решить *целый* ряд похожих задач, таких как задачи изменения и удаления данных, задачи манипулирования таблицами *базы данных* (т.е. их создание, изменение и удаление) и т.п. Все это типовые задачи, возникающие при разработке систем управления данными, и умение их решать, как и умение работать с базами данных в целом, очень важно для web-программиста.