УВАЖАЕМЫЕ СТУДЕНТЫ! Законспектируйте в своей рабочей тетради по дисциплине приведенную лекцию (объемом 4-5 страницы), ответьте *письменно* на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: igor-gricenko-95@mail.ru **в течении ТРЕХ дней.**

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)132-63-42

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция 18.
Тема: Взаимодействие PHP и MySQL
Цель: Ознакомится с основными аспектами взаимодействия PHP и MySQL

В дистрибутив PHP входит расширение, содержащее встроенные функции для работы с базой данных *MySQL*. В этой лекции мы познакомимся с некоторыми основными функциями для работы с *MySQL*, которые потребуются для решения задач построения web-интерфейсов с целью отображения и наполнения базы данных. Возникает вопрос, зачем строить такие интерфейсы? Для того чтобы вносить информацию в базу данных и просматривать ее содержимое могли люди, не знакомые с языком запросов SQL. При работе с web-интерфейсом для добавления информации в базу данных человеку нужно просто ввести эти данные в *html-форму* и отправить их на сервер, а наш скрипт сделает все остальное. А для просмотра содержимого таблиц достаточно просто щелкнуть по ссылке и зайти на нужную страницу.

Для наглядности будем строить эти интерфейсы для таблицы Artifacts, в которой содержится информация об экспонатах виртуального музея информатики. В предыдущей лекции мы уже приводили структуру этой коллекции, а также ее связи с коллекциями описания персон (Persons) и изображений (Images). Напомним, что каждый экспонат в коллекции Artifacts описывается с помощью следующих характеристик:

- название (title);
- автор (author);
- описание (description);
- альтернативное название (alternative);
- изображение (photo).

Название и альтернативное название являются строками менее чем 255 символов длиной (т.е. имеют тип VARCHAR(255)), описание - текстовое поле (имеет тип TEXT), а в полях "автор" и "изображение" содержатся идентификаторы автора из коллекции Persons и изображения экспоната из коллекции Images соответственно.

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 seperator 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:

Листинг 11.0.1. Форма для ввода данных в таблицу Artifacts (html, txt)

```
<?
$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 базе
$n = mysql_num_fields($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 " <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> ". $name_f ."</b></font></td>";
switch ($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>";
?>
```

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'].
Листинг 11.0.2. insert.php (html, txt)

```
<?
$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 базе
$n = mysql_num_fields($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>";
```

?>

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, 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 браузер 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 север 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 настоящие имена 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
```

Пример 17.1. Запрос с дублирующимися именами колонок (html, txt)

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:

Листинг 17.1.1. Отображение данных из Artifacts в виде таблицы в браузере (html, txt)

```
<?
/ * 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 " <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 $k => $val) { // перебираем все
                // имена полей
        $value = mysql_result($q,$i,$val); // получаем
                // значение поля
        echo "<td><font size=2> $value</font></td>";
                // выводим значение поля
        }
    echo "</tr>";
    }
echo "</table>";
```

Сделаем то же самое с помощью *mysql_fetch_array()*:

Листинг 17.1.2. Отображение данных из Artifacts в виде таблицы в браузере. Вариант 2 (html, txt)

```
<?
/* ... начало  то же, что и в предыдущем примере */
// отображаем значения полей
for($i=0;$i<$n; $i++){
// получаем значение поля в виде ассоциативного массива
     while($row = mysql_fetch_array($q, MYSQL_ASSOC)) {
        echo "<tr>";
        foreach  ($names as $k => $val){
            echo "<td><font size=2> $row[$val]</font></td>";
                // выводим значение поля
        }
    echo "</tr>";
    }
}
echo "</table>";
?>
```