Тема 4.3
«Функции PHP по работе с базами данных»

Вступление

PHP поддерживает большое количество самых разнообразных СУБД, но поскольку MySQL является, де-факто, стандартом для использования с PHP, именно функции по работе с этой СУБД мы сейчас и рассмотрим.

С идентификатором или без него?

Исторически так сложилось, что функции по работе с MySQL могут использовать идентификатор соединения, указанный явно или не указанный вообще. Во втором случае используется «соединение по умолчанию».

По мере развития PHP использование «идентификатора по умолчанию» стало считаться плохой практикой. Он до сих пор работает, но использовать его не рекомендуется. И мы – не будем.

Итак, рассмотрим настройки PHP, связанные с MySQL, а также функции PHP по работе с MySQL и их особенности.

Настройки (php.ini)

mysql.allow_persistent (On/Off) – разрешать ли постоянные соединения с MySQL;

mysql.max_persistent (целое число) – максимальное количество постоянных соединений на один процесс;

mysql.max_links (целое число) – максимальное количество соединений с MySQL на один процесс, включая постоянные соединения;

mysql.default_port (строка) – TCP-порт, используемый для соединения с СУБД по умолчанию (если не был указан другой при вызове mysql_connect/mysql_pconnect). Если эта директива опущена, порт будет взят из переменной среды MYSQL_TCP_PORT, значения mysql-tcp в /etc/services или константы MYSQL_PORT, указанной при компиляции, в указанном порядке. Win32 использует только константу MYSQL_PORT.

Настройки (php.ini)

mysql.default_socket (строка) – тип сокета, используемого для соединения с локальной СУБД;

mysql.default_host (строка) – адрес сервера, используемый для соединения с СУБД, если не указан другой при вызове mysql_connect/mysql_pconnect;

mysql.default_user (строка) – имя пользователя, используемое для соединения с базой данных, если не указано другое при вызове mysql_connect/mysql_pconnect;

mysql.default_password (строка) – пароль, используемый для соединения с базой данных, если не указан другой при вызове mysql_connect/mysql_pconnect;

mysql.connect_timeout (целое число) – время ожидания ответа от СУБД до разрыва соединения в секундах.

Соединение с СУБД

resource mysql_connect ([string server [, string username [, string password [, bool new_link [, int client_flags]]]]]) – возвращает указатель на соединение с MySQL в случае успешного выполнения, или FALSE при неудаче.

Следующие значения по умолчанию установлены для отсутствующих параметров: server = 'localhost:3306', username = ‘имя пользователя владельца процесса сервера’ и password = пустой пароль.

Параметр server может также включать номер порта, к примеру "hostname:1234" или путь к сокету, к примеру ":/path/to/socket" для локального сервера.

При указании параметру server значения "localhost" или "localhost:port" клиентская библиотека MySQL будет пытаться соединиться с локальным сокетом. Если вы всё же хотите использовать TCP/IP, используйте адрес "127.0.0.1" вместо "localhost".

Соединение с СУБД

Если второй вызов функции произошёл с теми же аргументами mysql_connect(), новое соединение не будет установлено. Вместо этого функция вернёт ссылку на уже установленное соединение. Параметр new_link, равный TRUE, может заставить функцию mysql_connect() открыть ещё одно соединение, даже если соединение с аналогичными параметрами уже открыто.

Параметр client_flags должен быть комбинацией из следующих констант:

- MYSQL_CLIENT_COMPRESS – включает компрессию данных, передаваемых между MySQL и PHP;
- MYSQL_CLIENT_IGNORE_SPACE – позволяет вставлять пробелы после имён функций;
- MYSQL_CLIENT_INTERACTIVE – ждать interactive_timeout секунд вместо wait_timeout до закрытия соединения (в настройках MySQL).

Соединение с сервером будет закрыто при завершении исполнения скрипта, если до этого оно не будет закрыто с помощью функции mysql_close().

Соединение с СУБД

Пример:

```
$lnk = mysql_connect("localhost", “username", "password")
        or die("Could not connect: " . mysql_error());
```

Это «сокращённая запись» проверки успешности выполнения операции функциями, возвращающими FALSE в случае неудачи. Классическая развёрнутая запись будет такой:

```
if (FALSE===($lnk=mysql_connect(“localhost”,”username”, “password”))) die ("Could not connect: " . mysql_error());
```

Соединение с СУБД

resource mysql_pconnect ([string server [, string username [, string password [, int client_flags]]]]) – возвращает указатель на постоянное соединение с MySQL или FALSE, в случае ошибки.

Функция mysql_pconnect() устанавливает **постоянное соединение** с сервером MySQL, а в остальном схожа (в параметрах) с mysql_connect() за исключением **двух отличий:**

1) При соединении, функция пытается найти уже открытый (постоянный) указатель на тот же сервер с тем же пользователем и паролем. Если он найден, возвращён функцией будет именно он, вместо открытия нового соединения.

2) Соединение с SQL-сервером не будет закрыто, когда работа скрипта закончится. Вместо этого, оно останется рабочим для будущего использования (mysql_close() также не закрывает постоянные соединения).

Внимание! Соединения такого типа работают только, если PHP установлен как модуль Apache, а НЕ как внешнее cgi-приложение.

Соединение с СУБД

Постоянные соединения

Постоянные соединения представляют собой связи с СУБД, которые не закрываются при завершении скрипта.

При получении запроса на постоянное соединение PHP проверяет, имеется ли идентичное постоянное соединение (которое было открыто при предыдущих обращениях) и, если такое найдено, использует его. В случае, если идентичного соединения нет, PHP создаёт новое.

Под «идентичным соединением» подразумевается соединение, открытое на тот же хост с таким же именем пользователя и паролем.

Постоянные соединения полезны в том случае, если при открытии большого количества SQL-соединений возникает ощутимая нагрузка на сервер. То, насколько велика эта нагрузка, зависит от многих факторов. Например, от того, какая именно база данных используется, находится ли она на том же компьютере что и ваш веб-сервер, насколько загружена машина, на которой установлен SQL-сервер, и так далее.

Соединение с СУБД

Преимущества постоянных соединений:

- экономят время (иногда – существенное) на повторной установке соединения;
- позволяют избежать превышения «лимита установленных соединений в единицу времени», который иногда бывает довольно строгим у некоторых хостеров.

Недостатки постоянных соединений:

- в случае, если скрипт использует сложную логику работы с БД (включая блокировку таблиц и транзакции, если их поддерживает СУБД) и завершится с ошибкой до разблокировки таблиц или завершения транзакции (в некоторых СУБД)) – дальнейшая работа с СУБД может потребовать перезапуска СУБД.
- повышает риск исчерпать «лимит одновременно установленных соединений», который у некоторых хостеров также достаточно жёсткий.

Соединение с СУБД

Выводы по постоянным и временным соединениям:

- временные соединения оказываются более эффективными, когда для генерации страницы используется «долговыполняющийся скрипт» (соединение с СУБД должно быть открыто достаточно долго) и сайт одновременно может просматривать большое количество посетителей;
- постоянные соединения хороши для случаев, когда к СУБД происходит множество кратковременных подключений, но количество одновременно открытых соединений будет невелико;
- если же нам приходится выполнять параллельно большое количество кратковременных операций с БД, наилучшим решением является оптимизация логики работы PHP-скрипта для кэширования и повторного использования полученных от СУБД данных или вообще кэширования всей готовой сгенерированной страницы.

Выбор БД

После того, как мы установили соединение с СУБД, следует указать, с какой базой данных мы будем работать.

bool mysql_select_db (string database_name [, resource link_identifier]) – возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Функция mysql_select_db() выбирает для работы указанную базу данных на сервере, на который ссылается переданный указатель. Если параметр указателя опущен, используется последнее открытое соединение.

Если нет ни одного открытого соединения, функция попытается соединиться с сервером аналогично функции mysql_connect(), вызванной без параметров.

Каждый последующий вызов функции mysql_query() будет работать с выбранной базой данных.

Выбор БД

Пример:

```
$lnk = mysql_connect('localhost', ‘username', 'password')
        or die ('Not connected : ' . mysql_error());
mysql_select_db(‘dbname', $lnk)
or die ('Can\'t use dbname : ' . mysql_error());
```

А если возникли ошибки?

string mysql_error ([resource link_identifier]) – возвращает строку, содержащую текст ошибки выполнения последней функции MySQL, или «» (пустая строка) если операция выполнена успешно.

Если в функцию не передан параметр ссылки на соединение, будет использовано последнее открытое соединение.

Данная функция возвращает код ошибки только последней выполненной функции. Поэтому проверяйте результат выполнения одной операции до выполнения следующей.

Пример:

```
mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno($lnk) . ": " . mysql_error($lnk);
// 1146: Table ‘dbname.nonexistenttable' doesn't exist
```

А если возникли ошибки?

В предыдущем примере мы также использовали функцию mysql_errno().

int mysql_errno ([resource link_identifier]) – возвращает код ошибки последней функции работы с MySQL, или 0 если операция выполнена успешно. Коды ошибок следует читать в руководстве по MySQL.

Эта функция также возвращает код ошибки только последней выполненной функции.

Выполнение запроса

resource mysql_query (string query [, resource link_identifier]) – посылает запрос активной базе данных сервера, на который ссылается переданный указатель.

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 **буфферизируется (т.е., фактически, данные оказываются в пространстве памяти PHP)**.

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 в случае успешного выполнения запроса и FALSE в случае ошибки.

Выполнение запроса

Результат работы этой функции, не равный FALSE, говорит ТОЛЬКО о том, что запрос был выполнен успешно. Он не говорит о количестве затронутых или возвращённых рядов.

Вполне возможна ситуация, когда успешный запрос не затронет ни одного ряда.

Пример использования mysql_query():

```
$result = mysql_query("SELECT my_col FROM my_tbl");
```

Функция mysql_query() также считает ошибочной ситуацией и вернёт FALSE, если у пользователя, от имени которого установлено соединение, не хватает прав на работу с указанной в запросе таблицей.

Функция

resource mysql_db_query (string database, string query [, resource link_identifier])

выполняющая запрос к указанной базе данных не рекомендована к использованию, начиная с PHP 4.0.6.

Выполнение запроса

resource mysql_unbuffered_query (string query [, resource link_identifier]) – выполняет запрос, результаты которого **НЕ будут буферизироваться**.

Это позволяет сохранить достаточно большое количество памяти для SQL-запросов, возвращающих большое количество данных.

Кроме того, вы можете начать работу с полученными данными сразу после того, как первый ряд был получен: вам не приходится ждать до конца выполнения SQL-запроса.

Замечание: использование mysql_unbuffered_query() имеет недостатки:

- вы не можете использовать функции mysql_num_rows() и mysql_data_seek() с результатом запроса, возвращённым этой функцией;
- вы должны будете обработать все ряды запроса до отправки нового запроса.

Обработка результатов запроса

int mysql_affected_rows ([resource link_identifier]) mysql_affected_rows() – возвращает количество рядов, затронутых последним INSERT, UPDATE, DELETE запросом к серверу, на который ссылается указатель link_identifier.

Если ресурс не указан, функция использует последнее, успешное соединение, выполненное с помощью функции mysql_connect().

Если последний запрос был DELETE без указания WHERE и, соответственно, таблица была очищена, функция вернёт 0.

Эта функция не работает с запросами SELECT – только с запросами, модифицирующими таблицу.

Если последний запрос был неудачным, функция вернёт -1.

Пример:

```
mysql_query("DELETE FROM mytable WHERE id < 10");
echo "Records deleted:“. mysql_affected_rows();
```

Обработка результатов запроса

Более детальную информацию (удобно для отладки) о результатах выполнения запроса можно получить с помощью функции

string mysql_info ([resource link_identifier]) – возвращает детальную информацию о последнем запросе к серверу, на который ссылается переданный функции указатель link_identifier.

Если параметр link_identifier не указан, используется последнее открытое соединение.

Эта функция возвращает строку для всех описанных ниже запросов. Для всего остального она возвращает FALSE.

Формат строки зависит от вида запроса.

Пример (виды запросов и ответов):

INSERT INTO ... SELECT ...
String format: Records: 23 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
String format: Records: 37 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
String format: Records: 42 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
String format: Records: 60 Duplicates: 0 Warnings: 0
UPDATE
String format: Rows matched: 65 Changed: 65 Warnings: 0

Извлечение данных

array mysql_fetch_array (resource result [, int result_type]) – возвращает массив (индексация по умолчанию двойная – числа и названия полей) с одним рядом результата запроса, или FALSE, если рядов больше нет.

Второй опциональный аргумент result_type в функции mysql_fetch_array() – константа и может принимать следующие значения:

MYSQL_ASSOC – возвратить только ассоциативный массив;

MYSQL_NUM – возвратить только индексный массив;

MYSQL_BOTH – возвратить массив с двойной индексацией.

Имена полей, возвращаемые этой функцией, регистро-зависимы.

Пример:

```
$result = mysql_query("SELECT id, name FROM mytable");
while ($row = mysql_fetch_array($result, MYSQL_BOTH))
 { prin_r($row) }
mysql_free_result($result);
```

Извлечение данных

Функции

array mysql_fetch_row (resource result)

и

array mysql_fetch_assoc (resource result)

делают то же самое, что и mysql_fetch_array(), вызванная с параметром result_type, равным MYSQL_NUM и MYSQL_ASSOC соответственно.

Функция

object mysql_fetch_object (resource result) – возвращает объект со свойствами, соответствующими колонкам в обработанном ряду, или FALSE, если рядов больше нет.

Эта функция работает аналогично mysql_fetch_array(), с единственным отличием – возвращает объект вместо массива.

Извлечение данных

mixed mysql_result (resource result, int row [, mixed field]) mysql_result() – возвращает значение одной ячейки результата запроса.

Аргументы: row – ряд получившейся в результате запроса таблицы, field – имя или номер (не рекомендуется) колонки.

Работая с большими результатами запросов, следует использовать одну из функций, обрабатывающих сразу целый ряд результата (mysql_fetch_row(), mysql_fetch_array(), mysql_fetch_assoc() и mysql_fetch_object()). Так как эти функции возвращают значение нескольких ячеек сразу, они НАМНОГО быстрее mysql_result(). Однако, если нам нужно выполнять много мелких операций с разными рядами, эта функция оказывается полезной.

Вызовы функции mysql_result() не должны смешиваться с другими функциями, работающими с результатом запроса.

Пример:

```
$result = mysql_query("SELECT name, phone FROM employee")
echo mysql_result($result,2,” phone”);
```

Извлечение данных

bool mysql_data_seek (resource result_identifier, int row_number) – возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Перемещает внутренний указатель в результате запроса к ряду с указанным номером. Следующий вызов mysql_fetch_row() (или аналогичных функций) вернёт именно его.

Параметр row_number должен быть значением от 0 до mysql_num_rows - 1. Функцию mysql_num_rows() скоро рассмотрим.

Замечание: функция mysql_data_seek() может быть использована только с mysql_query(), но не с mysql_unbuffered_query().

Извлечение данных

Пример:

```
$query = "SELECT last_name, first_name FROM friends";
$result = mysql_query($query);
// получение рядов в обратном порядке
for ($i = mysql_num_rows($result) - 1; $i >= 0; $i--)
{
  if (!mysql_data_seek($result, $i))
   {
    echo "Cannot seek to row $i: " . mysql_error();
    continue;
   }
    if(!($row = mysql_fetch_object($result)))           continue;
    echo "$row->last_name $row->first_name<br />\n";
}
mysql_free_result($result);
```

Извлечение данных

object mysql_fetch_field (resource result [, int field_offset]) – возвращает объект, содержащий информацию о колонке.

Может использоваться для получения информации о колонках конкретного запроса. Если смещение field_offset не указано, функция возвращает информацию о первой колонке, которая ещё не была обработана предыдущим вызовом mysql_fetch_field().

Свойства объекта:

- name – название колонки;
- table – название таблицы, которой принадлежит колонка;
- max_length – максимальная длинна содержимого колонки;
- not_null – 1, если колонка не может быть равна NULL;
- primary_key – 1, если колонка – первичный ключ;
- unique_key – 1, если колона – уникальный индекс;
- multiple_key – 1, если колонка – не уникальный индекс;
- numeric – 1, если колонка численная;
- blob – 1, если колонка – BLOB;
- type – тип колонки (название типа в MySQL);
- unsigned – 1, если колонка строго положительная;
- zerofill – 1, если колонка заполняется нулями;

Извлечение данных

int mysql_field_seek (resource result [, int field_offset]) – перемещает указатель к колонке с переданным индексом.

Если вызов функции не содержит смещения, будет возвращено текущее смещение.

Используется совместно с только что рассмотренной функцией mysql_fetch_field().

array mysql_fetch_lengths (resource result) – возвращает массив длин для каждого поля, содержащегося в последнем ряду, обработанном функциями mysql_fetch_row(), mysql_fetch_assoc(), mysql_fetch_array(), и mysql_fetch_object().

Извлечение данных

string mysql_field_flags (resource result, int field_offset) mysql_field_flags() – возвращает флаги указанной колонки.

Каждый флаг возвращается как отдельное слово отделённое от предыдущего пробелом. Полученное значение можно разбить в массив, используя функцию explode().

Возвращаются следующие флаги (если ваша версия MySQL поддерживает работу с ними): «not_null», «primary_key», «unique_key», «multiple_key», «blob», «unsigned», «zerofill», «binary», «enum», «auto_increment», «timestamp».

Извлечение данных

int mysql_field_len (resource result, int field_offset) mysql_field_len() – возвращает длину указанной колонки.

string mysql_field_name (resource result, int field_index) – возвращает название колонки с указанным номером field_index. Колонки нумеруются с нуля.

string mysql_field_type (resource result, int field_offset) – возвращает тип колонки с указанным номером field_offset. Колонки нумеруются с нуля.

string mysql_field_table (resource result, int field_offset) – возвращает название таблицы, которой принадлежит указанное field_offset поле, если запрос выполнялся к нескольким таблицам. Аналог mysql_tablename().

Извлечение данных

int mysql_num_rows (resource result) – возвращает количество рядов результата запроса.

Эта функция работает только с запросами SELECT. Чтобы получить количество рядов, обработанных функциями INSERT, UPDATE, DELETE, используйте функцию mysql_affected_rows().

Пример:

```
$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);
```

При использовании mysql_unbuffered_query() функция mysql_num_rows() не вернёт корректного значения до тех пор, пока все ряды не будут получены.

int mysql_num_fields (resource result) – возвращает количество полей результата запроса result. В остальном аналогична mysql_num_rows().

Извлечение данных

int mysql_insert_id ([resource link_identifier]) – возвращает ID, сгенерированный колонкой с AUTO_INCREMENT последним запросом INSERT к серверу, на который ссылается переданный функции указатель link_identifier.

Если параметр link_identifier не указан, используется последнее открытое соединение.

Возвращает 0, если последний запрос не работал с AUTO_INCREMENT полями. Если вам надо сохранить значение, убедитесь, что mysql_insert_id() вызывается сразу после запроса INSERT.

Если ваша колонка AUTO_INCREMENT имеет тип BIGINT, значение, возвращаемое функцией mysql_insert_id(), будет искажено. Вместо него используйте функцию SQL LAST_INSERT_ID().

Пример:

```
mysql_query("INSERT INTO mytable (product) values ('kossu')");
echo “ID последней записи: ".mysql_insert_id();
```

Закрытие соединения

bool mysql_close ([resource link_identifier]) – закрывает соединение с базой данных MySQL, на которое указывает переданный указатель.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Если параметр link_identifier не указан, закрывается последнее открытое (текущее) соединение.

Использование mysql_close() не необходимо для непостоянных соединений. Они автоматически закрываются в конце скрипта.

Важно: mysql_close() не закрывает постоянные соединения, открытые функцией mysql_pconnect(). Постоянные соединения закрываются по таймауту или в случае той или иной ошибки. Проэмулировать такую «ошибку» (закрыв соединение со стороны MySQL) можно так (может не работать в Win32):

```
$lnk=mysql_pconnect(…);
$thread_id = mysql_thread_id($lnk);
mysql_querry("kill $thread_id“, $lnk);
```

Освобождение памяти

bool mysql_free_result (resource result) – освобождает память, занимаемую результатом, на который ссылается переданный функции указатель result.

Нуждается в вызове только в том случае, если вы всерьёз обеспокоены тем, сколько памяти используют ваши запросы к БД, возвращающие большое количество данных.

Вся память, используемая для хранения этих данных автоматически очистится в конце работы скрипта.

Возвращает TRUE в случае успешного завершения или FALSE в случае возникновения ошибки.

Информация о СУБД

Для написания гибких приложений, способных изменять своё поведение в зависимости от версии или состояния СУБД, следует использовать следующий набор функций:

resource mysql_list_processes ([resource link_identifier]) mysql_list_processes() – возвращает указатель на результат, содержащий в себе текущие, выполняемые сервером, процессы.

Пример:

```
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$result = mysql_list_processes($link);
while ($row = mysql_fetch_assoc($result))
echo $row["Id"].” - ”.$row["Host"].” - ”.$row["db"].” - ”.$row["Command"].” - ”.$row["Time"];
// 1 - localhost - test - Processlist 0 4 localhost mysql sleep - 5
```

int mysql_thread_id ([resource link_identifier]) – возвращает ID текущего соединения. Если соединение потеряно и вы пересоединились с помощью mysql_ping(), ID изменится.

Информация о СУБД

bool mysql_ping ([resource link_identifier]) – проверяет соединение с сервером. Если оно утеряно, автоматически предпринимается попытка пересоединения. Эта функция может быть использована в скриптах, работающих на протяжении долгого времени.

Возвращает TRUE, если соединение в рабочем состоянии и FALSE в противном случае.

string mysql_stat ([resource link_identifier]) – возвращает текущий статус сервера: время работы, количество потоков, запросов, открытых таблиц и количество запросов в секунду.

Для полного списка переменных статуса, используйте SQL-запрос “SHOW STATUS”.

Пример:

```
$link = mysql_connect('localhost', "mysql_user", "mysql_password");
$status = explode('  ', mysql_stat($link));
print_r($status);
```

Array ([0] => Uptime: 5380 [1] => Threads: 2 [2] => Questions: 1321299 [3] => Slow queries: 0 [4] => Opens: 26 [5] => Flush tables: 1 [6] => Open tables: 17 [7] => Queries per second avg: 245.595)

Информация о СУБД

string mysql_client_encoding ([resource link_identifier]) – возвращает название кодировки, с которой работает текущее соединение.

Пример:

```
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
echo mysql_client_encoding($link);
// latin1
```

string mysql_get_client_info (void) – возвращает строку, содержащую версию клиентской библиотеки.

Пример:

```
echo mysql_get_client_info());
// 5.01.21
```

Информация о СУБД

string mysql_get_host_info ([resource link_identifier]) – возвращает строку, описывающую соединение, на которое ссылается переданный указатель link_identifier, включая имя хоста. Если параметр link_identifier опущен, будет использовано последнее соединение.

Пример:

```
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
echo mysql_get_host_info($link));
// Localhost via UNIX socket
```

int mysql_get_proto_info ([resource link_identifier]) – возвращает версию протокола, используемую соединением, на которое ссылается указатель link_identifier. Если параметр link_identifier опущен, используется последнее открытое соединение.

Пример:

```
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
echo mysql_get_proto_info());
// 10
```

Информация о СУБД

string mysql_get_server_info ([resource link_identifier]) – возвращает версию сервера, на соединение с которым ссылается переданный функции указатель link_identifier. Если параметр link_identifier опущен, будет использовано последнее открытое соединение.

```
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
echo mysql_get_server_info());
// 4.0.1-alpha
```

Передача данных

Синтаксис языка SQL достаточно неустойчив к попаданию в SQL-запросы внутри пользовательского текста символов, являющихся ключевыми для языка SQL.

Для того, чтобы защитить SQL-запрос от завершения неудачей в случае попадания в него таких символов, следует использовать функцию

string mysql_real_escape_string (string unescaped_string [, resource link_identifier]) – экранирует специальные символы в unescaped_string, принимая во внимание кодировку соединения, таким образом, что результат можно безопасно использовать в SQL-запросе в функции mysql_query().

Если вставляются бинарные данные, то к ним так же необходимо применять эту функцию. Она вызывает библиотечную функцию MySQL mysql_real_escape_string, которая добавляет обратные слеши к следующим символам:

\x00 \n \r \ ‘ " \x1a

Возвращает строку, в которой экранированы все необходимые символы, или FALSE в случае ошибки.

Передача данных

Пример (взлом с использованием SQL-injection):

```
$query = "SELECT * FROM users WHERE user='{$_POST['username']}' AND password='{$_POST['password']}'";
mysql_query($query);
// Мы не никак проверили переменную $_POST['password'],
// а она может содержать совсем не то, что мы
// ожидали. Например:
// $_POST['username'] = 'aidan';
// $_POST['password'] = "' OR ''='";
// посмотрим, какой запрос будет отправлен в MySQL:
echo $query;
// SELECT * FROM users WHERE name='aidan'
// AND password='' OR ''='
```

Передача данных

Примечание: функцию mysql_real_escape_string() можно использовать только после того, как установлено соединение с MySQL. В противном случае возникнет ошибка уровня E_WARNING, а функция возвратит FALSE. Если нужно обработать данные без установки соединения, можно использовать устаревшую аналогичную функцию

string mysql_escape_string (string unescaped_string)

которой не нужен идентификатор соединения.

Если в mysql_real_escape_string() link_identifier не указан, используется последнее открытое соединение.

Замечание: если в настройках PHP включена опция magic_quotes_gpc, сначала данные следует обработать функцией stripslashes().

Не использовать!

Следующие функции НЕ рекомендуется использовать, т.к. аналогичного эффекта можно добиться выполнением соответствующего SQL-запроса или набора функций.

mysql_create_db – создаёт базу данных MySQL (“create database dbname”);

mysql_db_name – возвращает название БД (выполнить запрос “show databases” и просмотреть его результат);

mysql_drop_db – уничтожает базу данных MySQL (“drop database dbname”);

mysql_list_dbs – возвращает список баз данных, доступных на сервере (“show databases”);

mysql_list_tables – возвращает список таблиц базы данных MySQL (“show tables from dbname”);

int mysql_change_user (string user, string password [, string database [, resource link_identifier]]) – изменяет пользователя для указанного текущего активного соединения, или соединения переданного опциональным параметром link_identifier. Вместо этого лучше закрыть соединение (mysql_close()) и открыть его заново (mysql_connect()).