EZSQL 概况

Ezsql 是一个是你在 PHP 中能够更简便使用数据库(mySQL/Oracle8/9/ InterBase/FireBird/ PostgreSQL/MS-SQL/SQLite/SQLite c++).的小工具。你只需要将一个小小的 PHP 文件包含到你的脚本中，就可以替代你从手册中查到的 PHP 标准数据库函数，你将使用的是更加小巧和简单的 ezsql 函数。它可以自动缓存查询的结果并允许使用容易理解的函数去操作和提取其中的数据，而且不需要额外的服务器开销。它有着优秀的调试函数可以让你立刻看到你的 sql 代码的运行结果。大多数的 ezsql 函数可以返回对象或者关联数组或者数值数组。它可以大幅减少开发所需要的时间，并且在多数情况下可以精简你的代码并使程序运行的更加快速，你还可以非常简便的调试和优化你的查询。这是一个小类，并不会对你的网站造成很大的系统开销。

本文假定你是一个熟悉 PHP 并且对数据库和 SQL 语法有着基本认识的人，如果你是一个完全想新手，你可以通过阅读我们教程获得帮助。

简单的例子：

在这些例子中，我们不需要为 ezsql 输入额外的代码

例子 1

```
// Select multiple records from the database and print them out..
//取出复数的行并输出结果
$users = $db->get_results("SELECT name, email FROM users");
foreach ( $users as $user ){
    // Access data using object syntax/用对象的方式取得结果
    echo $user->name;
    echo $user->email;
}
```

Example 2

```
// Get one row from the database and print it out.取出一行结果并输出
$user = $db->get_row("SELECT name,email FROM users WHERE id = 2");
echo $user->name;
echo $user->email;
```

Example 3

```
// Get one variable from the database and print it out..
//从数据库取出一个变量并输出
$var = $db->get_var("SELECT count(*) FROM users");
echo $var;
```

Example 4

```
// Insert into the database 向数据库插入数据
$db->query("INSERT INTO users (id, name, email) VALUES (NULL,'justin','jv@foo.com')");
```

Example 5

```
// Update the database 更新数据
$db->query("UPDATE users SET name = 'Justin' WHERE id = 2)");
```

Example 6

```
// Display last query and all associated results
//输出之前的查询及其关联的结果
$db->debug();
```

Example 7

```
// Display the structure and contents of any result(s) .. or any variable
//输出所有取得的结果及变量的结构和内容
$results = $db->get_results("SELECT name, email FROM users");
$db->vardump($results);
```

Example 8

```
// Get 'one column' (based on column index) and print it out..
//取出一列（索引中存在的）并输出结果。
$names = $db->get_col("SELECT name,email FROM users",0)
foreach ( $names as $name ){
    echo $name;
}
```

Example 9

```
// Same as above ‘but quicker’
与上一个例子作用相同，但是更快
foreach ( $db->get_col("SELECT name,email FROM users",0) as $name ){
    echo $name;
}
```

Example 10

```
// Map out the full schema of any given database and print it out..
//显示任何被给出的数据库的结构并输出
$db->select("my_database");
foreach ( $db->get_col("SHOW TABLES",0) as $table_name ){
    $db->debug();
    $db->get_results("DESC $table_name");
}
$db->debug();
```

介绍

当进行数据库操作的时候，你可能会进行以下四种类型的基本操作。

1.执行一个查询进行插入或更新操作（不需要返回结果）

2.从数据库中获取一个变量

3.从数据库中取出一行

4.从数据库中得到一个结果集

EZsql 将以上四种操作封装到四个非常简单的函数中去：

bool $db->query(query)

返回布尔值：$db->query(query)

var $db->get_var(query)

返回变量 $db->get_var(query)

mixed $db->get_row(query)

输出布尔值和变量 $db->get_row(query)

mixed $db->get_results(query)

输出布尔值和变量 $db->get_results(query)

你在大多数情况下所需要的就是 ezsql 的这四个函数，当然，我们还有许多有用的函数可以使用，我们将在以后介绍他们。

注意：如果你需要使用 ezsql 的函数，你需要首先将它放到 global 中并初始化。

安装

在安装 ezsql 之前，你需要从网上下载 ez_sql.zip 到所需要使用这个类的服务器路径下。

在代码的开始添加如下内容

//包含 ezsql 的代码核心

include_once "ez_sql_core.php";

//包含对应的数据库组件（比如 mysql）

include_once "ez_sql_mysql.php";

//包含 ez_sql_mysql.php

//初始化数据库对象并创建连接

//同时需要输入以下变量 （数据库用户、密码、数据库名、主机名称（或 IP））

$db = new **ezSQL_mysql**('db_user','db_password','db_name','db_host');

注意：在大多数的系统中，localhost 这个主机名称是最直接有效的名字，如果你不确定你所需要设置的参数，请咨询服务提供商或查询提供商的文档。

如果你是第一次在机器上安装 ezsql，在你配置好一个可用的 mysql 账号之前。请留空用户名和密码。

运行 ezSQL 演示

当你已经安装好 ezsql 之后，你可以运行 ez_demo.php 来看它是如何工作的，输入此路径以运行该文件：

http://yourserver.com/install_path/mysql/demo.php

http://你的域名/安装路径/mysql/demo.php

如果你是在本地调试该程序，则输入此路径

http://127.0.0.1/install_path/mysql/demo.php

http://127.0.0.1/安装路径/mysql/demo.php

这个演示文件能为我们做什么呢？他可以通过 ezsql 的内置函数将你在文件开始所选定的数据库结构输出出来，你可能会被他简短的代码所震惊，我将这段代码放在这里，你可以通过阅读它认识 ezsql 的简洁和效率

```
<?php
//包含 ezsql 的数据核心
include_once "ez_sql_core.php";
```

```
//包含对应数据库的操作文件
include_once "ez_sql_mysql.php";
//初始数据库对象并创建连接
$db = new ezSQL_mysql('db_user','db_password','db_name','db_host');
$my_tables = $db->get_results("SHOW TABLES",ARRAY_N);
$db->debug();
foreach ( $my_tables as $table ){
    $db->get_results("DESC $table[0]");
    $db->debug();
}
?>
```

对 EZSQL 演示文件的解释

<?php

这是执行 PHP 语言的开始标记

include_once “ez_sql.php”;

这就是如何在 PHP 文件中包含 ezsql 的方法，一般都在代码的开始，然后之后才能使用 ezsql 的函数。

$my_tables = $db->get_results(“SHOW TABLES”,ARRAY_N);

get_results()是 ezsql 的一个从数据库中输出一个集合的函数，输出的集合是一个数组，在这里，调用了标准的 mysql 函数“show table”然后将结果集存储在一个新的数组里面，当使用 $db->get_results(),的时候，它可以将取得的结果存储到一个多维数组中，第一维的数组中存储的是索引，每一个索引指向一个对象，关联数组或数值数组并将所有值存储在一行中。

例如，使用开关变量 ARRAY_A 可以输出一个这样的数组

```
$users = $db->get_results(“SELECT id,name FROM users”,ARRAY_A);
$users[0] = array (“id” => “1”, “name” => “Amy”);
$users[1] = array (“id” => “2”, “name” => “Tyson”);
```

If you wanted a numerical array use the switch ARRAY_N.

如果你需要一个数值数组，你可以使用开关变量 ARRAY_N.

```
$users = $db->get_results(“SELECT id,name FROM users”,ARRAY_N);
$users[0] = array (0 => “1”, 1 => “Amy”);
$users[1] = array (0 => “2”, 1 => “Tyson”);
```

If you wanted an object (which is the default option) you don’t need a switch..

如果你需要输出一个对象（默认输出），你可以不使用开关变量。

```
$users = $db->get_results(“SELECT id,name FROM users”);
$users[0]->id = “1”;
$users[0]->name = “Amy”;
$users[1]->id = “2”;
$users[1]->name = “Tyson”;
```

从数据库中返回数值数组的对象对于 PHP 的处理非常简单，例如循环输出一个对象结果只需要这么做：

```
$users = $db->get_results(“SELECT id,name FROM users”);
foreach( $users as $user ){
    echo $user->id;
    echo $user->name;
}
```

如果你确定有输出结果你可以跳过一步：

```
foreach( $db->get_results(“SELECT id,name FROM users”) as $user ){
    echo $user->id;
    echo $user->name;
}
```

如果你不能确定有没有返回结果你可以这么写：

```
If ( $users= $db->get_results(“SELECT id,name FROM users”) )
foreach( $users as $user ){
    echo $user->id;
    echo $user->name;
}else{
    echo “No results”;
    $db->debug();
}
```

这个函数可以输出最近的一个查询并包含一个已经排好版的表格包含有该查询所产生的所有结果和列的信息。

```
foreach ( $my_tables as $table)
```

这是 PHP 中使用的一个非常简单的遍历数组的方法，在这个过程中，数组 $my_tables 将通过运行 ezsql 的命令$db->get_results(“SHOW TABLES”,ARRAY_N)创建，其中，开关变量 ARRAY_N 将使得该返回结果为一个数值数组。

输出的结果将是这样的：

```
$my_tables[0] = array (0 => “users”);
$my_tables[1] = array (0 => “products”);
$my_tables[2] = array (0 => “guestbook”);
```

这个 foreach 的循环将遍历所有在$my_tables[n]中的元素并将他们转换到数值数组中，结构如下：

```
array(0 => “value”, 1 => “value”, etc.);
```

从而通过遍历$my_tables 我们可以取得所有的值：

```
foreach ($my_tables as $table)
echo $table[0];
```

如果我们生成一个关联数组将是这个样子：

```
$users = $db->get_results(“SELECT id,name FROM users”,ARRAY_A);
foreach ( $users as $user ){
    echo $user[‘id’];
    echo $user[‘name’];
}
```

但是如果我们没有返回值的话，遍历将返回一个警告，所以，安全起见，我们应当这么写：

```
if ( $users = $db->get_results(“SELECT id,name FROM users”,ARRAY_A))
foreach ( $users as $user ){
    echo $user[‘id’];
    echo $user[‘name’];
}else{
    echo “No Users”:
}
```

当没有任何结果的时候，将返回 false；

```
$db->get_results(“DESC $table[0]”);
```

在这个查询中我们嵌套了一个遍历循环，注意我们将前一个函数用到了新的函数中，从习惯上来说，你可能需要使用不同的数据库资源标示，但是在 ezsql 中，我们可以使这个过程变的简单，以嵌套在查询中。

你可能会对我为什么使用数值数组来输出结果而不是通过对象或关联数组呢？

原因是，我并不能确定所取得结果的第一列的名字，而我能够确定的是通过数值数组的话，我可以将结果以元素[0]将第一列输出出来

参考：sql 命令 show tables 可以依靠数据库中使用的名字命名第一个列。如果你的数据库命名为 users，这个列将被命名为 Tables_in_users，你的数据库命名为 customers，这个列将被命名为 Tables_in_ customers 等。

$db->debug();

这个函数将输出上一个查询的所有结果，在这里将输出上面的查询结果。

$db->get_results("DESC $table[0]");

你可能会注意到，**get_results 这个函数并没有分配值，（比如一个变量）这是因为如果你不分配 ezsql 函数的查询结果的话，这个结果将被缓存并可以被任何的 ezsql 的函数所调用，比如$db->debug()可以显示出这个被存储的结果，然后使用空值访问任何一个 ezsql 的查询的话将可以访问到之前所存储的查询结果，下面是一个更直接的结果**

amy, amy@foo.com

tyson, tyson@foo.com

//结果将被缓存

$users = $db->get_results("SELECT name,email FROM users");

/这里将从上一个结果中获取变量。

echo $db->get_var(null,1,1);

//这里将把空查询传递给 get_var 然后取出之前的查询结果。

Output: tyson@foo.com

//关闭循环

?>

ezsql 函数

$db->get_results -- get multiple row result set from the database (or previously cached results)
从数据库（或之前缓存的结果）中取出复数的行

$db->get_row -- get one row from the database (or previously cached results)
从数据库（或之前缓存的结果）中取出一行

$db->get_col -- get one column from query (or previously cached results) based on column offset 从数据库（或之前缓存的结果）中取出预先设定的一列

$db->get_var -- get one variable, from one row, from the database (or previously cached results) 从数据库（或之前缓存的结果以及一行）一个变量

$db->query -- 对数据库进行一次查询并缓存任何返回结果

$db->debug --输出之前的查询的结果并返回其结果

$db->vardump -- 返回任何指定变量的结构和内容

$db->select -- 选定一个新数据库供接下来的使用

$db->get_col_info -- 获得一列的信息（例如：名字和类型）

$db->hide_errors --关闭 ezsql 的错误输出

$db->show_errors -- 打开 ezsql 的错误输出

$db->escape -- 格式化任何非标准 PHP 查询

$db = new db -- 初始化数据库对象

ezsql 的变量

$db->num_rows – 输出之前对数据库进行查询的影响行数。

$db->insert_id -- 从之前的 insert 查询中获取所生成的自增 ID

$db->rows_affected -- 获取之前所生效的 INSERT，UPDATE,以及 DELETE 所影响的行数。

$db->num_queries -- 对当前代码所生效的查询进行追踪

$db->debug_all – 如果将这个变量设置为 true，它将输出代码中所有的查询和结果。

$db->cache_dir – Path to mySQL caching dir.

设置 mysql 结果的缓存路径

$db->cache_queries – Boolean flag (see mysql/disk_cache_example.php)

布尔型标示，（详见 mysql/disk_cache_example.php）

$db->cache_inserts – Boolean flag (see mysql/disk_cache_example.php)

布尔型标示，（详见 mysql/disk_cache_example.php）

$db->use_disk_cache – Boolean flag (see mysql/disk_cache_example.php)

布尔型标示，（详见 mysql/disk_cache_example.php）

$db->cache_timeout – Number in hours (see mysql/disk_cache_example.php)

以小时为单位计算 （详见 mysql/disk_cache_example.php）

命令详解

$db = new db

$db = new db -- Initiate new db object. Connect to a database server. Select a database.

初始化一个数据库对象，链接数据库服务器，选择数据库

说明

$db = new db(string username, string password, string database name, string database host)

$db = new db(字符串用户名, 字符串密码, 字符串数据库名称, 字符串主机名)

这个函数完成 3 件事：1.初始化一个数据库对象 2.链接数据库服务器 3.选择数据库。你可以再次初始化这个函数为新的数据库对象，这意味着你可以同时进行两个并发的数据库连接，另外，你也可以在同时连接两个完全不同的数据服务器。

Note: For the sake of efficiency it is recommended that you only run one instance of the db object and use $db->select to switch between different databases on the same server connection.

建议：为了提高效率，建议初始化一个数据库对象然后使用$db->select 对一个服务器的两个数据库分别进行操作。

例子

```
// Initiate new database object..初始化数据库对象
$db2 = new db(”user_name”, ”user_password”, ”database_name”, “database_host”);
// Perform some kind of query..使用查询
$other_db_tables = $db2->get_results(“SHOW TABLES”);
// 你可以对将要连接的数据库进行查询
$existing_connection_tables = $db->get_results(“SHOW TABLES”);
// 输出之前的所有查询结果
$db->debug();
$db2->debug();
```

$db->select

$db->select -- select a new database to work with

选择一个将使用的数据库

说明

bool $db->select(string database name)

布尔型 $db->select(字符串数据库名称)

$db->select() 使用当前的数据库连接设置，选择一个将使用的数据库

例子

//从用户的其他数据库中取出一个用户的名字（需要初始化数据库）

$user_name = $db->get_var("SELECT name FROM users WHERE id = 22") ;

//选择 stats 这个数据库（注意，切换数据库了）

$db->select("stats");

//从用户的其他数据库中取出一个用户的名字

$total_hours = $db->get_var("SELECT sum(time_logged_in) FROM user_stats WHERE user = '$user_name'") ;

//重新返回当前数据库并继续当前的工作

$db->select("users");

$db->query

$db->query -- send a query to the database (and if any results, cache them)

对数据库进行一次查询并缓存任何返回结果

说明

bool $db->query(string query)

布尔型 **$db->query(字符串 查询）**

$db->query()向当前选择的数据库执行一个查询，要注意的是，你可以通过这个命令执行任何查询命令，任何的返回结果都将被缓存并可以在你执行一个空查询之前所调用，如果有返回结果，该函数的值为 true，若无结果，返回 false。

例子 1

向数据库中加入新用户

$db->query("INSERT INTO users (id,name) VALUES (1,'Amy')") ;

例子 2

// Update user into the database..更新数据库中的用户

$db->query("UPDATE users SET name = 'Tyson' WHERE id = 1") ;

例子三

// Query to get full user list..查询整个用户列表

$db->query("SELECT name,email FROM users") ;

// Get the second row from the cached results by using a null query..

使用空查询取出之前缓存的结果的第二行

$user_details = $db->get_row(null, OBJECT,1);

// Display the contents and structure of the variable $user_details..

显示出变量$user_details 的结构和内容

$db->vardump($user_details);

$db->get_var

$db->get_var -- get one variable, from one row, from the database (or previously cached results)

从数据库的一行（或之前缓存的结果）中取得一个变量

说明

var $db->get_var(string query / null [,int column offset[, int row offset])

返回变量 $db->get_var(字符串查询/空值)

$db->get_var() gets one single variable from the database or previously cached results. This function is very useful for evaluating query results within logic statements such as if or switch. If the query generates more than one row the first row will always be used by default. If the query generates more than one column the leftmost column will always be used by default. Even so, the full results set will be available within the array $db->last_results should you wish to use them.

$db->get_var()从数据库的一行（或之前缓存的结果）中取得一个变量，这个函数配 合 switch 和 if 来判断查询结果是非常有效的，如果查询返回了复数的行，那么将默认返回第一行，如果返回了复数的列，那么将默认返回最左边的列，另外，该查询的结果将可以被 $db->last_results 以数组的方式取得

例子 1

// Get total number of users from the database..

//取出数据库中的用户总数

$num_users = $db->get_var(“SELECT count(*) FROM users”) ;

例子 2

// Get a users email from the second row of results (note: col 1, row 1 [starts at 0])..

//从第二行中取出一个用户的 email 地址（说明，第一行，第一列，默认起始值为 0）

$user_email = $db->get_var(“SELECT name, email FROM users”,1,1) ;

// Get the full second row from the cached results (row = 1 [starts at 0])..

//取出之前缓存的整个第二行

$user = $db->get_row(null,OBJECT,1);

// Both are the same value..结果一样

echo $user_email;

echo $user->email;

例子 3

// Find out how many users there are called Amy..

//找出有多少个名叫 amy 的用户

if ($n = $db->get_var(“SELECT count(*) FROM users WHERE name = ‘Amy’”)){

// If there are users then the if clause will evaluate to true. This is useful because

// we can extract a value from the DB and test it at the same time.

//如果有用户被返回的话，将输出一个 true 这在判断中是非常有利的，因为我们可以在这个时候对取出的值进行判断。

echo “There are $n users called Amy!”;

}else{

// If there are no users then the if will evaluate to false..

//如果没有用户返回，这里将判断为 false

echo “There are no users called Amy.”;

}

例子 4

// Match a password from a submitted from a form with a password stored in the DB

与数据库中存储的密码比对一个从 submit 事件中得到的密码，

if ($pwd_from_form == $db->get_var(“SELECT pwd FROM users WHERE name = ‘$name_from_form’”)){

// Once again we have extracted and evaluated a result at the same time..

//又一次我们在取出数据的同时进行了比较

echo “Congratulations you have logged in.”;

}else{

// If has evaluated to false..

//如果判断为 false

echo “Bad password or Bad user ID”;

}

$db->get_row

$db->get_row -- get one row from the database (or previously cached results)

从数据库（或之前缓存的结果）中取出一行

说明

object $db->get_ row(string query / null [, OBJECT / ARRAY_A / ARRAY_N [, int row offset]])

返回对象$db->get_ row(字符串查询/空值（OBJECT / ARRAY_A / ARRAY_N 三个可选参数）

$db->get_row() gets a single row from the database or cached results. If the query returns more than one row and no row offset is supplied the first row within the results set will be returned by default. Even so, the full results will be cached should you wish to use them with another ezSQL query.

$db->get_row()从数据库（或之前缓存的结果）中取出一行，如果查询得到的结果多于一行，将默认输出第一行，同时该查询的结果将被缓存以备其他 ezsql 查询使用。

例子 1

// Get a users name and email from the database and extract it into an object called user..

从数据库中获取用户名和邮件地址，并将它放入一个叫 USER 的对象。

$user = $db->get_row(“SELECT name,email FROM users WHERE id = 22”) ;

// Output the values..输出值

echo “$user->name has the email of $user->email”;

Output:输出值

Amy has the email of amy@foo.com

例子 2

// Get users name and date joined as associative array

从数据库中取出用户名和时间并放入关联数组

// (Note: we must specify the row offset index in order to use the third argument)

注意，我们需要设置行的索引值以使用该参数

$user = $db->get_row(“SELECT name, UNIX_TIMESTAMP(my_date_joined) as date_joined FROM users WHERE id = 22”,ARRAY_A) ;

// Note how the unix_timestamp command is used with as this will ensure that the resulting data will be easily

注意他可以非常简单的使用 uxix 时间戳确保结果

// accessible via the created object or associative array. In this case $user[‘date_joined’] (object would be $user->date_joined)

创建可访问的对象或关联数组，在这个例子中， $user[‘date_joined’]是对象 would be $user->date_joined。

echo $user[‘name’] . “ joined us on ” . date(“m/d/y”,$user[‘date_joined’]);

输出

Amy joined us on 05/02/01

例子三

// Get second row of cached results.

从缓存中取出第二行

$user = $db->get_row(null,OBJECT,1) ;

// Note: Row offset starts at 0

注意：默认值为 0

echo “$user->name joined us on ” . date(“m/d/y”,$user->date_joined);

输出

Tyson joined us on 05/02/02

结果 4

// Get one row as a numerical array..

以数值数组方式取值

$user = $db->get_row(“SELECT name,email,address FROM users WHERE id = 1”,ARRAY_N);

// Output the results as a table..

以表的方式输出结果

echo “<table>”;

for ($i=1; $i <= count($user); $i++){

echo “<tr><td>$i</td><td>$user[$I]</td></tr>”;

}

echo “</table>”;

输出:

1	amy
2	amy@foo.com
3	123 Foo Road

$db->get_results

$db->get_results – get multiple row result set from the database (or previously cached results)从数据库（或之前缓存的结果）中取出复数的行

说明

array $db->get_results(string query / null [, OBJECT / ARRAY_A / ARRAY_N])

返回数组（对象），可选参数 OBJECT / ARRAY_A / ARRAY_N

$db->get_row() gets multiple rows of results from the database based on query and returns them as a multi dimensional array. Each element of the array contains one row of results and can be specified to be either an object, associative array or numerical array. If no results are found then the function returns false enabling you to use the function within logic statements such as if.

$db->get_row()用查询从数据库中取出复数的行并转存为多维数组，数组中的每个元素包含一行结果，且可以以对象，数值数组，关联数组的方式调用，如果没有查询结果，该函数可以返回 false 供逻辑判断。

例子 1，以对象方式返回结果

Returning results as an object is the quickest way to get and display results. It is also useful that you are able to put $object->var syntax directly inside print statements without having to worry about causing php parsing errors.

以对象方式返回查询结果是最快的取得结果的方法，同时他也可以用 $object->var 这个方

法非常快速的取到结果而且不用担心 PHP 的解析错误。

```
// Extract results into the array $users  将结果放入数组$users
(and evaluate if there are any results at the same time)..
同时判断其中的任何结果
if ( $users = $db->get_results("SELECT name, email FROM users") ){
// Loop through the resulting array on the index $users[n]
通过索引$users[n]循环整个数组
foreach ( $users as $user ){
// Access data using column names as associative array keys
依靠列名取得数据并保存到关联数组的键值中
    echo "$user->name - $user->email<br>";
}
}else{
// If no users were found then if evaluates to false..
如果没有用户返回则输出 false
    echo "No users found.";
}
Output:
Amy - amy@hotmail.com
Tyson - tyson@hotmail.com
例子 2，以关联数组方式返回结果
以关联数组方式返回数据有助于你动态获取结果。
这里是例子
// Extract results into the array $dogs
将数据存入数组$dogs
(and evaluate if there are any results at the same time)..
同时可以进行判断
if ( $dogs = $db->get_results("SELECT breed, owner, name FROM dogs", ARRAY_A) ){
// Loop through the resulting array on the index $dogs[n]
以$dogs[n]为索引循环数组
foreach ( $dogs as $dog_detail ){
    // Loop through the resulting array
    循环赋值给数组
    foreach ( $dogs_detail as $key => $val ){
    // Access and format data using $key and $val pairs..
    以键名和键值的方式取得数据
        echo"<b>" . ucfirst($key) . "</b>: $val<br>";
    }
    输出 P
    echo "<p>";
}}else{
// If no users were found then if evaluates to false..
如果没有任何结果输出将得到 false
    echo "No dogs found.";
}
Output:
```

Breed: Boxer
Owner: Amy
Name: Tyson
Breed: Labrador
Owner: Lee
Name: Henry
Breed: Dachshund
Owner: Mary
Name: Jasmine

Example 3 – Return results as numerical array

以数值数组方式返回结果

Returning results as a numerical array is useful if you are using completely dynamic queries with varying column names but still need a way to get a handle on the results. Here is an example of this concept in use. Imagine that this script was responding to a form with $type being submitted as either 'fish' or 'dog'.

将结果以数值数组方式输出有利于使用动态方式查询结果并取出变量的列名（0,1,2,3…）但依然需要手段使我们取得结果，这里是一个例子讲告诉我们如何使用它，想象这是一个能够响应类别的函数，以 fish 或者 dog 的方式。

```
// Create an associative array for animal types..创建一个保存动物类别的关联数组
$animal = array ( "fish" => "num_fins", "dog" => "num_legs" );
// Create a dynamic query on the fly..
创建一个动态查询供使用
if ( $results = $db-> get_results ("SELECT $animal[$type] FROM $type",ARRAY_N)){
    foreach ( $results as $result ){
        echo "$result[0]<br>";
    }
}else{
    echo "No $animal\s!";
}
```

Output:

4
4
4

Note: The dynamic query would be look like one of the following...

动态查询其实是这个样子的

- SELECT num_fins FROM fish
- SELECT num_legs FROM dogs

It would be easy to see which it was by using $db->debug(); after the dynamic query call.

当动态的查询被执行之后，它将非常简单的被$db->debug()查看。

$db->debug

$db->debug – print last sql query and returned results (if any)

输出前一个 sql 查询并返回任何存在的结果

说明

$db->debug(void)

$db->debug() prints last sql query and its results (if any)

输出前一个 sql 查询并返回任何存在的结果

If you need to know what your last query was and what the returned results are here is how you do it.

如果你需要知道你上一个查询得到了什么，你可以这么做。

// Extract results into the array $users..将结果存入数组$users

$users = $db->get_results("SELECT name, email FROM users");

// See what just happened!查看它是如何执行的。

$db->debug();

$db->vardump

$db->vardump – print the contents and structure of any variable

输出变量的结构和内容

说明

$db->vardump(void)

$db->vardump() prints the contents and structure of any variable. It does not matter what the structure is be it an object, associative array or numerical array.

输出变量的结构和内容，可以无视所输出的变量的类型（对象，关联数组，或数值数组）

例子 1

如果你想知道这个变量的结构和值，你可以这么做

// Extract results into the array $users..

将结果放入数组$users.

$users = $db->get_results("SELECT name, email FROM users");

// View the contents and structure of $users

输出数组$users.的结构和内容。

$db->vardump($users);

$db->get_col

$db->get_col – get one column from query (or previously cached results) based on column offset 按照列设置从查询中取出一列

说明

$db->get_col(string query / null [, int column offset])

$db->get_col(字符串查询/参数)

$db->get_col() extracts one column as one dimensional array based on a column offset. If no offset is supplied the offset will defualt to column 0.

I.E the first column. If a null query is supplied the previous query results are used.

$db->get_col()将设定的一列的值取出并放入一个一维数组中，如果没有参数，将返回第 1 列（数组下界从 0 开始，原文 column 0.）

注意，如果当前查询为空将提取前一个查询的第一列作为结果

例子 1

// Extract list of products and print them out at the same time..

将产品表存入数组并输出来，

```
foreach ( $db->get_col("SELECT product FROM product_list") as $product){
    echo $product;
}
```

使用之前缓存的结果

// Extract results into the array $users..

将结果存入数组$users

$users = $db->get_results(“SELECT * FROM users”);

// Work out how many columns have been selected..

计算其中的列数

$last_col_num = $db->num_cols - 1;

// Print the last column of the query using cached results..

从缓存结果中输出最后一列

```
foreach ( $db->get_col(null, $last_col_num) as $last_col ){
    echo $last_col;
}
```

$db->get_col_info

$db->get_col_info - get information about one or all columns such as column name or type

依据列名或类型取得一列或多列的信息

说明

$db->get_col_info(string info-type[, int column offset])

$db->get_col_info（字符串，信息类型/参数])

$db->get_col_info()依据列名或类型返回目标的一列或多列的信息，如果没有指定所要获取的信息类别，将输出默认的类别 name，如果没有没有列参数设定将默认输出所有列信息，如需要访问所有的目标信息，你可以使用这个调用缓存的方式$db->col_info。

可用的信息类别

mySQL

- name - column name 列名称
- table - name of the table the column belongs to 该列所在的表名称
- max_length - maximum length of the column 该列的最大长度
- not_null - 1 if the column cannot be NULL 判断该列不能为空
- primary_key - 1 if the column is a primary key 判断该列是否为主键
- unique_key - 1 if the column is a unique key 判断该列是否为唯一键（unique_key）
- multiple_key - 1 if the column is a non-unique key 判断该列是否为唯一键，输出结果与上一个相反
- numeric - 1 if the column is numeric 判断该列是否为数值
- blob - 1 if the column is a BLOB 判断该列是否为（binary large object)，二进制大对象（百科）
- type - the type of the column 列的类型
- unsigned - 1 if the column is unsigned 判断列是否为无符号类型（百科）
- zerofill - 1 if the column is zero-filled 判断列是否为 zerofill 类型（可能是得到之前的无符号结果（百度说的+_+））

MS-SQL / Oracle / Postgress

- name - column name 列名称
- type - the type of the column 列的类型
- length - size of column 该列的长度

SQLite

- name - column name 列名称

例子 1

// Extract results into the array $users..将结果存入数组$users

$users = $db->get_results(“SELECT id, name, email FROM users”);
// Output the name for each column type
按照列的类别输出列名
foreach ($db->get_col_info(“name”) as $name){
echo “$name
”;
}
Output:
id
name
Email
例子 2
// Extract results into the array $users..将结果存入数组$users
$users = $db->get_results(“SELECT id, name, email FROM users”);
// View all meta information for all columns..查看所有目标列的信息
$db->vardump($db->col_info);
$db->hide_errors

$db->hide_errors – turn ezSQL error output to browser off
关闭 **ezsql** 的错误输出
说明
$db->hide_errors(void)
$db->hide_errors() stops error output from being printed to the web client. If you would like to stop error output but still be able to trap errors for debugging or for your own error output function you can make use of the global error array $EZSQL_ERROR.
$db->hide_errors()停止向客户端输出 ezsql 的错误提示，如果你希望在关闭错误提示后依然能够得到调试的错误信息或你自己的错误提示函数，你可以使用全局数组 $EZSQL_ERROR 得到错误提示。
Note: If there were no errors then the global error array $EZSQL_ERROR will evaluate to false. If there were one or more errors then it will have the following structure. Errors are added to the array in order of being called.
注意：如果没有任何错误的话，全局数组$EZSQL_ERROR 将置为 false，如果有任何错误的话，将被以如下方式存入数组以备调用。
（以下为 ez 内部函数区，翻译了也用不了，所以不翻译了，反正大家都知道+_+）
$EZSQL_ERROR[0] = Array(
[query] => SOME BAD QUERY
[error_str] => You have an error in your SQL syntax near ‘SOME BAD QUERY' at line 1
)
$EZSQL_ERROR[1] = Array(
[query] => ANOTHER BAD QUERY
[error_str] => You have an error in your SQL syntax near ‘ANOTHER BAD QUERY' at line 1
)
$EZSQL_ERROR[2] = Array(
[query] => THIRD BAD QUERY
[error_str] => You have an error in your SQL syntax near ‘THIRD BAD QUERY' at line 1
)
例子

// Using a custom error function 使用一个常规的错误函数

$db->hide_errors();

// Make a silly query that will produce an error 使用一个错误的查询来制造一个错误信息。

$db->query("INSERT INTO my_table A BAD QUERY THAT GENERATES AN ERROR");

// And another one, for good measure

$db->query("ANOTHER BAD QUERY THAT GENERATES AN ERROR");

用另一个来做对照

// If the global error array exists at all then we know there was 1 or more ezSQL errors..

如果全局错误数组存在，我们将得到这里存在一个或多个 ezsql 的错误。

if ($EZSQL_ERROR){

/ / View the errors

$db->vardump($EZSQL_ERROR);

}else{

echo "No Errors";

}

$db->show_errors

$db->show_errors – turn ezSQL error output to browser on

打开 ezsql 的错误输出

说明

$db->show_errors(void)

$db->show_errors() turns ezSQL error output to the browser on. If you have not used the function $db->hide_errors this function (show_errors) will have no effect.

打开 ezsql 的错误输出，在未使用函数$db->hide_errors 之前，此函数无效。

$db->escape

$db->escape – Format a string correctly in order to stop accidental mal formed queries under all PHP conditions.格式化任何非标准 PHP 查询

说明

$db->escape(string)

$db->escape() makes any string safe to use as a value in a query under all PHP conditions.

格式化任何非标准 PHP 查询使之适用于 PHP 环境

I.E. if magic quotes are turned on or off.

附加：无视 magic quotes（自动转义）

Note: Should not be used by itself to guard against SQL injection attacks. The purpose of this function is to stop accidental mal formed queries.

注意：不可用于防范 sql 注入攻击，该函数仅用于格式化非标准查询。

例子 1

// Escape and assign the value..**格式化值**

$title = $db->escape("Justin's and Amy's Home Page");

插入数据库

$db->query("INSERT INTO pages (title) VALUES ('$title')") ;

例子 2

// Assign the value..分配值

$title = "Justin's and Amy's Home Page";

// Insert in to the DB and escape at the same time..在插入数据库的时候格式化它

$db->query("INSERT INTO pages (title) VALUES ('". $db->escape($title)."')") ;

Ezsql 可以缓存你的查询使动态网站能运行的更快

If you want to cache EVERYTHING just do..如果你希望缓存所有结果，可以这么做

```
$db->use_disk_cache = true;
$db->cache_queries = true;
$db->cache_timeout = 24;
```

For full details and more specific options please see:

你可以查看以下文件来得到该参数的进一步信息

·mysql/disk_cache_example.php

·oracle8_9/disk_cache_example.php

SQLite示例文件

```
<?php
// Include ezSQL core
include_once "../shared/ez_sql_core.php";
// Include ezSQL database specific component
include_once "ez_sql_sqlite.php";
// Initialise database object and establish a connection 数据库初始化和建立连接
// at the same time - db_path / db_name
$db = new ezSQL_sqlite('./','sqlite_test.db');
// Create a table..创建表
$db->query("CREATE TABLE test_table ( MyColumnA INTEGER PRIMARY KEY, MyColumnB TEXT(32) );");
// Insert test data 插入测试数据
for($i=0;$i<3;++$i){
    $db->query('INSERT INTO test_table (MyColumnB) VALUES ('".md5(microtime())."');');
}
// Get list of tables from current database..
$my_tables = $db->get_results("SELECT * FROM sqlite_master WHERE sql NOTNULL;");
// Print out last query and results..
$db->debug();
// Loop through each row of results..
foreach ( $my_tables as $table ){
    // Get results of DESC table..
    $db->get_results("SELECT * FROM $table->name;");
    // Print out last query and results..
    $db->debug();
}
// Get rid of the table we created..
$db->query("DROP TABLE test_table;");
?>
```

MySQL 示例文件

```
<?php
    include_once "../shared/ez_sql_core.php";
    include_once "ez_sql_mysql.php";
    $db = new ezSQL_mysql('db_user','db_password','db_name','db_host'); //建立连接

    $current_time = $db->get_var("SELECT " . $db->sysdate());
    print "ezSQL demo for mySQL database run @ $current_time";
    // Print out last query and results..
    $db->debug();
    // Get list of tables from current database..
    $my_tables = $db->get_results("SHOW TABLES",ARRAY_N);
    // Print out last query and results..
    $db->debug();
    // Loop through each row of results..
    foreach ( $my_tables as $table ){
        // Get results of DESC table..
        $db->get_results("DESC $table[0]");
        // Print out last query and results..
        $db->debug();
    }
    ?>
```