AbstractDB 1.0.2 Manual

Contents

Package AbstractDB Procedural Elements

abstractdb.class.php

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

abstractdb_driver.class.php

AbstractDB Driver Base Class Definition

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

ABSTRACTDB_DRIVER_INCLUDED = true *[line 18]*

AbstractDB Driver included constant.

Flag that indicates that the driver base class file has been included.

- **Access** public

abstractdb_mysql.php

AbstractDB MySQL Driver

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

ABSTRACTDB_MYSQL_INCLUDED = true *[line 18]*

MySQL Driver included constant.

Flag that indicates that the MySQL driver has been included.

- **Access** public

affected_rows.php

GetAffectedRows Example

This example uses the GetAffectedRows method to get the number of rows affected by the last query.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

fetch_next_result.php

FetchNextResult Example

This example uses the FetchNextResult[Assoc, Object, Row] functions to retrieve query results. The method shown in this example uses FetchNextResultRow but the other functions would be used in the same way.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 18]*

Include the AbstractDB main class.

fetch_result.php

FetchResult Example

This example uses the FetchResult[Assoc, Object, Row] functions to retrieve the first query result. The method shown in this example uses FetchResultRow but the other functions would be used in the same way.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 18]*

Include the AbstractDB main class.

fetch_result_all.php

FetchResult All Example

This example uses the FetchResult[Assoc, Object, Row]All functions to retrieve the all query results. The method shown in this example uses FetchResultRowAll but the other functions would be used in the same way.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 18]*

Include the AbstractDB main class.

fetch_result_column.php

FetchResultColumn Example

This example uses the FetchResultColumn function to execute a query and retrieve the values in the first column of the result set.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 17]*

Include the AbstractDB main class.

fetch_result_field.php

FetchResultField Example

This example uses the FetchResultField function to execute a query and retrieve the value of the first field of the first row of the result set.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 17]*

Include the AbstractDB main class.

fetch_seek_result.php

FetchSeekResult Example

This example uses the FetchSeekResult[Assoc, Object, Row] functions to retrieve a query result. The method shown in this example uses FetchSeekResultRow but the other functions would be used in the same way.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 18]*

Include the AbstractDB main class.

field_count.php

GetFieldCount Example

This example uses the GetFieldCount method to get the number of fields in a result set.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

field_names.php

GetFieldNames Example

This example uses the GetFieldNames method to get the names of fields in a result set.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

insert_id.php

GetInsertID Example

This example uses the GetInsertID method to get the ID of the last inserted AUTO_INCREMENT record.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

instantiate_associative.php

Instantiation Example

This example uses an associative array of arguments to instantiate AbstractDB.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

instantiate_connectionstring.php

Instantiation Example

This example uses a ConnectionString argument to instantiate AbstractDB.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

query.php

Query Example

This example shows how to query a database using AbstractDB.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

query_column.php

QueryColumn Example

This example uses the QueryColumn function to query a database and retrieve the values in the first column of the result set.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 17]*

Include the AbstractDB main class.

query_field.php

QueryField Example

This example uses the QueryField function to query a database and retrieve the value of the first column of the first row of a result set.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 17]*

Include the AbstractDB main class.

query_result.php

Query Example

This example uses the Query[Assoc, Objects, Row] functions to query a database and retrieve the first result. The method shown in this example uses QueryRow but the other functions would be used in the same way.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 18]*

Include the AbstractDB main class.

query_result_all.php

Query All Example

This example uses the Query[Assoc, Object, Row]All functions to retrieve the all query results. The method shown in this example uses QueryRowAll but the other functions would be used in the same way.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 18]*

Include the AbstractDB main class.

replace.php

Replace Example

This example show how to use the Replace method.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

row_count.php

GetRowCount Example

This example uses the GetRowCount method to get the number of rows in a result set.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

set_error_handler.php

SetErrorHandler Example

This example show how to set an error handler for AbstractDB.

- **Package** AbstractDB
- **Filesource** Source Code for this file
- **License** GNU Lesser General Public License
- **Version** v 1.0.2
- **Copyright** (C) 2005 Pacific-Cybersoft. All Rights Reserved.
- **Author** Pacific-Cybersoft

function ExampleErrorHandler($source, $error) *[line 18]*
include **"../abstractdb.class.php"** *[line 16]*

Include the AbstractDB main class.

Package AbstractDB Classes

Class AbstractDB

[line 18]

AbstractDB Main Class

The main AbstractDB class used to interact with various DBMS packages via the use of driver classes.

- **Package** AbstractDB
- **Access** public

AbstractDB::$Support

array = [line 52]

List of supported features of the currently loaded driver.

- **Access** public

AbstractDB::$_arguments

array = [line 26]

List of connection arguments.

- **Access** private

AbstractDB::$_driver

object = [line 32]

Reference to the AbstractDB driver.

- **Access** private

AbstractDB::$_error

string = [line 38]

The last error message.

- **Access** private

AbstractDB::$_error_handler

string = [line 44]

Name of an error handling function passed in SetErrorHandler.

- **Access** private

Constructor function AbstractDB::AbstractDB($arguments) *[line 93]*

Instantiation example using an associative array of arguments.

```php
<?php
/**
* Instantiation Example
*
* This example uses an associative array of arguments to instantiate AbstractDB.
* @package AbstractDB
* @author Pacific-Cybersoft
* @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
* @version v 1.0.2
* @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
*/

/**
* Include the AbstractDB main class.
*/
include("../abstractdb.class.php"          );

/**
* @global AbstractDB An instance of the {@link AbstractDB} class.
*/
$db = new AbstractDB(array(
    "Type"          =>   "mysql"          ,
    "Username"          =>   "username"          ,
    "Password"          =>   "password"          ,
    "Hostname"          =>   "localhost"          ,
    "Database"          =>   "cybermatrix"          ,
    "Options"          =>array(
        "Persistent"          =>   true,
        "Port"          =>   3306)
    ));

// Always good practice to close AbstractDB at the end of each script.
$db->   Close();
?>
```

Instantiation example using a ConnectionString argument.

```php
<?php
/**
* Instantiation Example
*
* This example uses a ConnectionString argument to instantiate AbstractDB.
* @package AbstractDB
* @author Pacific-Cybersoft
* @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
* @version v 1.0.2
* @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
*/

/**
* Include the AbstractDB main class.
*/
include("../abstractdb.class.php"          );

/**
* @global AbstractDB An instance of the {@link AbstractDB} class.
*/
$db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost:3306/mydatabas
e?Options/Persistent=1"     ));

// Always good practice to close AbstractDB at the end of each script.
$db->   Close();
?>
```

Function Parameters:

- *array* **$arguments** Database connection and driver specific arguments.

  Arguments must be supplied as an associative array containing one of the following:

  ConnectionString:

A string composed of the connection arguments and optional or driver specific options in the form of
Type://**Username**:**Password**@**Hostname**:**Port**/**Database**?**Options**/option1=value1&**Options**/option2=value2

e.g. $args = array("ConnectionString" =>
"mysql://root:pass@localhost/MyDatabase?Options/Persistent=1");

or

Type: The type of database to connect to. AbstractDB is currently only distributed with a MySQL driver.

Username: The database username.

Password: The database password.

Hostname: The database hostname or IP Address.

Database: The name of the database.

Options: An associative array of optional or driver specific options. See individual driver documentation for a potential list of driver specific options. The value for Port should be specified in this list, otherwise the default port for the database type will be used.

e.g. $args = array("Type" => "mysql", "Username" =>
"root", "Password" => "pass", "Hostname" =>
"localhost", "Database" => "MyDatabase",
"Options" => array("Persistent" => 1));

Type and Database are required parameters and must be specified.

AbstractDB Constructor

Initilises an instance of the AbstractDB class.

- **Example**
- **Example**
- **Internal** Parses connection arguments and loads the driver class.
- **Access** public

function AbstractDB::ClearError() *[line 826]*

Clears the latest error message.

- **Access** private

bool function AbstractDB::Close() *[line 106]*

Closes the Database Connection

- **Access** public

bool function AbstractDB::EnsureRequiredArguments() *[line 838]*

Ensures Required Connection Arguments Exist

- **Internal** If there are missing arguments an error is set stating which arguments are missing.
- **Access** private

bool function AbstractDB::FetchNextResultAssoc(&$rs, &$assoc) *[line 121]*

FetchNextResult example.

```
1     <?php
2     /**
3     * FetchNextResult Example
4     *
5     * This example uses the FetchNextResult[Assoc, Object, Row] functions to retrieve query results.
6     * The method shown in this example uses FetchNextResultRow but the other functions would be used
7     * in the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
```

```
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename"          ))
24    {
25        // Loop through results
26        while($db->   FetchNextResultRow($rs, $row))
27        {
28            // Do something with the data in $row.
29            print_r($row);
30        }
31        // Free Result
32        $db->   FreeResult($rs);
33    }
34    else
35    {
36        die($db->   GetLastError());
37    }
38    // Always good practice to close AbstractDB at the end of each script.
39    $db->   Close();
40    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$assoc** A reference to an array that will contain the result row.

Fetches the Next Row as an Associative Array

- **Example**
- **Access** public

bool function AbstractDB::FetchNextResultObject(&$rs, &$object) *[line 143]*

FetchNextResult example.

```
1     <?php
2     /**
3     * FetchNextResult Example
4     *
5     * This example uses the FetchNextResult[Assoc, Object, Row] functions to retrieve query results.
6     * The method shown in this example uses FetchNextResultRow but the other functions would be used
7     * in the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
```

```php
    // Instatiate AbstractDB
    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
    // Execute a Query
    if($rs = $db->   Query("SELECT * FROM tablename"          ))
    {
        // Loop through results
        while($db->   FetchNextResultRow($rs, $row))
        {
            // Do something with the data in $row.
            print_r($row);
        }
        // Free Result
        $db->   FreeResult($rs);
    }
    else
    {
        die($db->   GetLastError());
    }
    // Always good practice to close AbstractDB at the end of each script.
    $db->   Close();
    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *object* **$object** A reference to an object that will contain the result row.

Fetches the Next Row as an Object

- **Example**
- **Access** public

bool function AbstractDB::FetchNextResultRow(&$rs, &$row) *[line 165]*

FetchNextResult example.

```php
    <?php
    /**
    * FetchNextResult Example
    *
    * This example uses the FetchNextResult[Assoc, Object, Row] functions to retrieve query results.
    * The method shown in this example uses FetchNextResultRow but the other functions would be used
    * in the same way.
    * @package AbstractDB
    * @author Pacific-Cybersoft
    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
    * @version v 1.0.2
    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
    */

    /**
    * Include the AbstractDB main class.
    */
    include("../abstractdb.class.php"          );

    // Instatiate AbstractDB
```

```
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename"          ))
24    {
25        // Loop through results
26        while($db->   FetchNextResultRow($rs, $row))
27        {
28            // Do something with the data in $row.
29            print_r($row);
30        }
31        // Free Result
32        $db->   FreeResult($rs);
33    }
34    else
35    {
36        die($db->   GetLastError());
37    }
38    // Always good practice to close AbstractDB at the end of each script.
39    $db->   Close();
40    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$row** A reference to an array that will contain the result row.

Fetches the Next Row

- **Example**
- **Access** public

bool function AbstractDB::FetchResultAssoc(&$rs, &$assoc) *[line 186]*

FetchResult example.

```
1     <?php
2     /**
3     * FetchResult Example
4     *
5     * This example uses the FetchResult[Assoc, Object, Row] functions to retrieve the first query
result.
6     * The method shown in this example uses FetchResultRow but the other functions would be used in the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
```

```
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename LIMIT 0,1"          ))
24    {
25        // Get the result
26        $db->   FetchResultRow($rs, $row);
27        // Do something with the data in $row.
28        print_r($row);
29    }
30    else
31    {
32        die($db->   GetLastError());
33    }
34    // Always good practice to close AbstractDB at the end of each script.
35    $db->   Close();
36    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$assoc** A reference to an array that will contain the result row.

Fetches a Row as an Associative Array

Fetches the first row as an associative array and then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::FetchResultAssocAll(&$rs, &$all) *[line 209]*

FetchResultAll example.

```
1     <?php
2     /**
3     * FetchResult All Example
4     *
5     * This example uses the FetchResult[Assoc, Object, Row]All functions to retrieve the all query
results.
6     * The method shown in this example uses FetchResultRowAll but the other functions would be used in
the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
```

```
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename"          ))
24    {
25        // Get the result
26        $db->   FetchResultRowAll($rs, $all);
27        // Do something with the data in $all.
28        foreach($all as $row)
29        {
30            print_r($row);
31        }
32    }
33    else
34    {
35        die($db->   GetLastError());
36    }
37    // Always good practice to close AbstractDB at the end of each script.
38    $db->   Close();
39    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$all** A reference to an array that will contain the result rows.

Fetches All Rows as Associative Arrays

Fetches all rows as associative arrays and then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::FetchResultColumn(&$rs, &$column) *[line 234]*

FetchResultColumn example.

```
1     <?php
2     /**
3     * FetchResultColumn Example
4     *
5     * This example uses the FetchResultColumn function to execute a query and retrieve the values in the
6     * first column of the result set.
7     * @package AbstractDB
8     * @author Pacific-Cybersoft
9     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
10    * @version v 1.0.2
11    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
12    */
13
14    /**
15    * Include the AbstractDB main class.
16    */
17    include("../abstractdb.class.php"          );
```

```
18
19    // Instatiate AbstractDB
20    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
21    // Execute a Query
22    if($rs = $db->   Query("SELECT * FROM tablename"          ))
23    {
24        // Get the result
25        $db->   FetchResultColumn($rs, $column);
26        // Do something with the data in $column.
27        print_r($column);
28    }
29    else
30    {
31        die($db->   GetLastError());
32    }
33    // Always good practice to close AbstractDB at the end of each script.
34    $db->   Close();
35    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$column** A reference to an array that will contain the result column rows.

Fetches a Result Column

Fetches all rows of the first column and then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::FetchResultField(&$rs, &$field) *[line 259]*

FetchResultField example.

```
1     <?php
2     /**
3     * FetchResultField Example
4     *
5     * This example uses the FetchResultField function to execute a query and retrieve the value of the
6     * first field of the first row of the result set.
7     * @package AbstractDB
8     * @author Pacific-Cybersoft
9     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
10    * @version v 1.0.2
11    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
12    */
13
14    /**
15    * Include the AbstractDB main class.
16    */
17    include("../abstractdb.class.php"          );
18
19    // Instatiate AbstractDB
```

```
20    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
21    // Execute a Query
22    if($rs = $db->   Query("SELECT * FROM tablename LIMIT 0,1"          ))
23    {
24        // Get the result
25        $db->   FetchResultField($rs, $field);
26        // Do something with the data in $field.
27        print_r($field);
28    }
29    else
30    {
31        die($db->   GetLastError());
32    }
33    // Always good practice to close AbstractDB at the end of each script.
34    $db->   Close();
35    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *mixed* **$field** A reference to a variable that will contain the field value.

Fetches a Result Field

Fetches the value in the first column of the first row and then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::FetchResultObject(&$rs, &$object) *[line 282]*

FetchResult example.

```
1     <?php
2     /**
3     * FetchResult Example
4     *
5     * This example uses the FetchResult[Assoc, Object, Row] functions to retrieve the first query
result.
6     * The method shown in this example uses FetchResultRow but the other functions would be used in the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
```

```
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename LIMIT 0,1"          ))
24    {
25        // Get the result
26        $db->   FetchResultRow($rs, $row);
27        // Do something with the data in $row.
28        print_r($row);
29    }
30    else
31    {
32        die($db->   GetLastError());
33    }
34    // Always good practice to close AbstractDB at the end of each script.
35    $db->   Close();
36    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *object* **$object** A reference to an object that will contain the result row.

Fetches a Row as an Object

Fetches the first row as an object and then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::FetchResultObjectAll(&$rs, &$all) *[line 305]*

FetchResultAll example.

```
1     <?php
2     /**
3     * FetchResult All Example
4     *
5     * This example uses the FetchResult[Assoc, Object, Row]All functions to retrieve the all query
results.
6     * The method shown in this example uses FetchResultRowAll but the other functions would be used in
the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
```

```
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename"          ))
24    {
25        // Get the result
26        $db->   FetchResultRowAll($rs, $all);
27        // Do something with the data in $all.
28        foreach($all as $row)
29        {
30            print_r($row);
31        }
32    }
33    else
34    {
35        die($db->   GetLastError());
36    }
37    // Always good practice to close AbstractDB at the end of each script.
38    $db->   Close();
39    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$all** A reference to an array that will contain the result rows.

Fetches All Rows as Objects

Fetches all rows as objects and then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::FetchResultRow(&$rs, &$row) *[line 330]*

FetchResult example.

```
1     <?php
2     /**
3     * FetchResult Example
4     *
5     * This example uses the FetchResult[Assoc, Object, Row] functions to retrieve the first query
result.
6     * The method shown in this example uses FetchResultRow but the other functions would be used in the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
```

```
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename LIMIT 0,1"          ))
24    {
25        // Get the result
26        $db->   FetchResultRow($rs, $row);
27        // Do something with the data in $row.
28        print_r($row);
29    }
30    else
31    {
32        die($db->   GetLastError());
33    }
34    // Always good practice to close AbstractDB at the end of each script.
35    $db->   Close();
36    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$row** A reference to an array that will contain the result row.

Fetches a Row

Fetches the first row and then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::FetchResultRowAll(&$rs, &$all) *[line 353]*

FetchResultAll example.

```
1     <?php
2     /**
3     * FetchResult All Example
4     *
5     * This example uses the FetchResult[Assoc, Object, Row]All functions to retrieve the all query
results.
6     * The method shown in this example uses FetchResultRowAll but the other functions would be used in
the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
```

```
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename"          ))
24    {
25        // Get the result
26        $db->   FetchResultRowAll($rs, $all);
27        // Do something with the data in $all.
28        foreach($all as $row)
29        {
30            print_r($row);
31        }
32    }
33    else
34    {
35        die($db->   GetLastError());
36    }
37    // Always good practice to close AbstractDB at the end of each script.
38    $db->   Close();
39    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$all** A reference to an array that will contain the result rows.

Fetches All Rows

Fetches all rows and then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::FetchSeekResultAssoc(&$rs, $row_num, &$assoc) *[line 379]*

FetchSeekResult example.

```
1     <?php
2     /**
3     * FetchSeekResult Example
4     *
5     * This example uses the FetchSeekResult[Assoc, Object, Row] functions to retrieve a query result.
6     * The method shown in this example uses FetchSeekResultRow but the other functions would be used in
7     * the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
```

```
14
15      /**
16      * Include the AbstractDB main class.
17      */
18      include("../abstractdb.class.php"          );
19
20      // Instatiate AbstractDB
21      $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22      // Execute a Query
23      if($rs = $db->   Query("SELECT * FROM tablename"          ))
24      {
25          // Get the result
26          if($db->   FetchSeekResultRow($rs, 3, $row))
27              // Do something with the data in $row.
28              print_r($row);
29          }
30
31          // Free Result
32          $db->   FreeResult($rs);
33      }
34      else
35      {
36          die($db->   GetLastError());
37      }
38      // Always good practice to close AbstractDB at the end of each script.
39      $db->   Close();
40      ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *int* **$row_num** The position in the result set of the row to return.
- *array* **$assoc** A reference to an array that will contain the result row.

Fetches a Row as an Associative Array

Fetches the row at the specified position in the result set as an associative array.

- **Example**
- **Access** public

bool function AbstractDB::FetchSeekResultObject(&$rs, $row_num, &$object) *[line 407]*

FetchSeekResult example.

```
1       <?php
2       /**
3       * FetchSeekResult Example
4       *
5       * This example uses the FetchSeekResult[Assoc, Object, Row] functions to retrieve a query result.
6       * The method shown in this example uses FetchSeekResultRow but the other functions would be used in
7       * the same way.
```

```
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename"          ))
24    {
25        // Get the result
26        if($db->   FetchSeekResultRow($rs, 3, $row))
27            // Do something with the data in $row.
28            print_r($row);
29        }
30
31        // Free Result
32        $db->   FreeResult($rs);
33    }
34    else
35    {
36        die($db->   GetLastError());
37    }
38    // Always good practice to close AbstractDB at the end of each script.
39    $db->   Close();
40    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *int* **$row_num** The position in the result set of the row to return.
- *object* **$object** A reference to an object that will contain the result row.

Fetches a Row as an Object

Fetches the row at the specified position in the result set as an object.

- **Example**
- **Access** public

bool function AbstractDB::FetchSeekResultRow(&$rs, $row_num, &$row) *[line 435]*

FetchSeekResult example.

```
1     <?php
```

```php
    /**
    * FetchSeekResult Example
    *
    * This example uses the FetchSeekResult[Assoc, Object, Row] functions to retrieve a query result.
    * The method shown in this example uses FetchSeekResultRow but the other functions would be used in
    * the same way.
    * @package AbstractDB
    * @author Pacific-Cybersoft
    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
    * @version v 1.0.2
    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
    */

    /**
    * Include the AbstractDB main class.
    */
    include("../abstractdb.class.php"          );

    // Instatiate AbstractDB
    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
    // Execute a Query
    if($rs = $db->   Query("SELECT * FROM tablename"          ))
    {
        // Get the result
        if($db->   FetchSeekResultRow($rs, 3, $row))
            // Do something with the data in $row.
            print_r($row);
        }

        // Free Result
        $db->   FreeResult($rs);
    }
    else
    {
        die($db->   GetLastError());
    }
    // Always good practice to close AbstractDB at the end of each script.
    $db->   Close();
    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *int* **$row_num** The position in the result set of the row to return.
- *array* **$row** A reference to an array that will contain the result row.

Fetches a Row

Fetches the row at the specified position in the result set.

- **Example**
- **Access** public

bool function AbstractDB::FreeResult(&$rs) *[line 460]*

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.

Frees a Result Resource

Frees the resources associated with the given result handle returned by executing a query.

- **Access** public

int function AbstractDB::GetAffectedRows() *[line 477]*

GetAffectedRows example.

```
1     <?php
2     /**
3     * GetAffectedRows Example
4     *
5     * This example uses the GetAffectedRows method to get the number of rows affected by the last query.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20    // Execute a Query
21    if($rs = $db->   Query("SELECT * FROM tablename"          ))
22    {
23        // Get the number of affected rows
24        $affected = $db->   GetAffectedRows();
25        // Free Result
26        $db->   FreeResult($rs);
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

Gets the Number of Affected Rows

- **Example**
- **Access** public

string function AbstractDB::GetDatabase() *[line 493]*

Gets the Name of the Current Database

- **Access** public

function AbstractDB::GetDriverError() *[line 882]*

Gets the Latest Error from the Driver

- **Internal** Sets an error based on the driver error.
- **Access** private

int function AbstractDB::GetFieldCount(&$rs) *[line 507]*

GetFieldCount example.

```
1     <?php
2     /**
3     * GetFieldCount Example
4     *
5     * This example uses the GetFieldCount method to get the number of fields in a result set.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20    // Execute a Query
21    if($rs = $db->   Query("SELECT * FROM tablename"          ))
22    {
23        // Get the number of affected rows
24        $fieldcount = $db->   GetFieldCount($rs);
25        // Free Result
```

```
26          $db->   FreeResult($rs);
27      }
28      else
29      {
30          die($db->   GetLastError());
31      }
32      // Always good practice to close AbstractDB at the end of each script.
33      $db->   Close();
34      ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.

Gets the Number of Fields

Gets the number of fields returned by the given result handle.

- **Example**
- **Access** public

bool function AbstractDB::GetFieldNames(&$rs, &$fields) *[line 526]*

GetFieldNames example.

```
1       <?php
2       /**
3       * GetFieldNames Example
4       *
5       * This example uses the GetFieldNames method to get the names of fields in a result set.
6       * @package AbstractDB
7       * @author Pacific-Cybersoft
8       * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9       * @version v 1.0.2
10      * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11      */
12
13      /**
14      * Include the AbstractDB main class.
15      */
16      include("../abstractdb.class.php"          );
17
18      // Instatiate AbstractDB
19      $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20      // Execute a Query
21      if($rs = $db->   Query("SELECT * FROM tablename"          ))
22      {
23          // Get the field names
24          $db->   GetFieldNames($rs, $fields);
25          // Do something with the field names
26          foreach($fields as $field)
27          {
28              echo($field)
29          }
30          // Free Result
31          $db->   FreeResult($rs);
```

```
32    }
33    else
34    {
35        die($db->   GetLastError());
36    }
37    // Always good practice to close AbstractDB at the end of each script.
38    $db->   Close();
39    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$fields** A reference to an array that will contain the field names.

Gets the Field Names of a Query

- **Example**
- **Access** public

mixed function AbstractDB::GetInsertID() *[line 544]*

GetInsertID example.

```
1     <?php
2     /**
3     * GetInsertID Example
4     *
5     * This example uses the GetInsertID method to get the ID of the last inserted AUTO_INCREMENT record.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20    // Execute a Query
21    if($db->   Query("INSERT tablename (field1, field2) VALUES ('field1', 'field2')"          ))
22    {
23        // Get the insert ID
24        $inserID = $db->   GetInsertID();
25    }
26    else
27    {
28        die($db->   GetLastError());
29    }
30    // Always good practice to close AbstractDB at the end of each script.
31    $db->   Close();
32    ?>
```

Gets the Last Insert ID

- **Example**
- **Access** public

string function AbstractDB::GetLastError() *[line 560]*

Gets the Last Error.

- **Access** public

int function AbstractDB::GetRowCount(&$rs) *[line 574]*

GetRowCount example.

```
1     <?php
2     /**
3     * GetRowCount Example
4     *
5     * This example uses the GetRowCount method to get the number of rows in a result set.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20    // Execute a Query
21    if($rs = $db->   Query("SELECT * FROM tablename"          ))
22    {
23        // Get the number of rows
24        $rowcount = $db->   GetRowCount($rs);
25        // Free Result
26        $db->   FreeResult($rs);
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.

Gets the Number of Rows

Gets the number of rows returned by the last query of the given result handle.

- **Example**
- **Access** public

bool function AbstractDB::IsResource($resource, $callee) *[line 909]*

Function Parameters:

- *resource* **$resource** A variable to ensure is a resource.
- *string* **$callee** The name of the method calling this function. Used in the setting of the error message.

Checks That a Parameter is a Resource

Checks that a given parameter is a resource type variable.

- **Access** private

bool function AbstractDB::LoadDriver() *[line 925]*

Loads an AbstractDB Database Driver

Loads the AbstractDB driver for the type of database being connected to.

- **Internal** If the driver could not be loaded an error is set explaining the reason for the failure.

- **Access** private

bool function AbstractDB::ParseArguments($arguments) *[line 964]*

Function Parameters:

- *array* **$arguments** A list of connection and driver specific arguments.

Parses Connection and Driver Specific Arguments

- **Access** private

bool function AbstractDB::ParseConnectionArguments($connectionString) *[line 1002]*

Function Parameters:

- *string* **$connectionString** A connection string in the format **Type**://**User**:**Pass**@**Host**:**Port**/**Database**?**Options**/option1=value1&**Options**/option2=value2

Parses ConnectionString Argument

Parses the ConnectionString argument passed in via the constructor.

- **TODO** There may be a use for the fragment part of the parsed connection string.
- **Access** private

resource function AbstractDB::Query($sql) *[line 596]*

Query example.

```
1     <?php
2     /**
3     * Query Example
```

```
4     *
5     * This example shows how to query a database using AbstractDB.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20    // Execute a Query
21    if($rs = $db->   Query("SELECT * FROM tablename"          ))
22    {
23        // Use $rs to retrieve result rows.
24        $db->   FetchNextResultRow($rs, $row);
25        // Free Result
26        $db->   FreeResult($rs);
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.

Executes an SQL Statement

Executes the given SQL statement.

- **Example**
- **Internal** If the query did not execute successfully an error is set explaining as best as possible the reason for the failure.
- **Access** public

bool function AbstractDB::QueryAssoc($sql, &$assoc) *[line 614]*

Query example.

```
1     <?php
2     /**
3     * Query Example
4     *
5     * This example uses the Query[Assoc, Objects, Row] functions to query a database and retrieve the
```

```
6     * first result. The method shown in this example uses QueryRow but the other functions would
7     * be used in the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($db->   QueryRow("SELECT * FROM tablename LIMIT 0,1"          , $row))
24    {
25        // Do something with the data in $row
26        $field0 = $row[0];
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.
- *array* **$assoc** A reference to an array that will contain the result row.

Executes an SQL Statement

Executes the given SQL statement and retrieves the first result row as an associative array, then frees the result set.

- **Example**

bool function AbstractDB::QueryAssocAll($sql, &$all) *[line 633]*

Query All example.

```
1     <?php
2     /**
3     * Query All Example
4     *
5     * This example uses the Query[Assoc, Object, Row]All functions to retrieve the all query results.
6     * The method shown in this example uses QueryRowAll but the other functions would be used in the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
```

```
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   QueryRowAll("SELECT * FROM tablename"          , $all))
24    {
25        // Do something with the data in $all.
26        foreach($all as $row)
27        {
28            print_r($row);
29        }
30    }
31    else
32    {
33        die($db->   GetLastError());
34    }
35    // Always good practice to close AbstractDB at the end of each script.
36    $db->   Close();
37    ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.
- *array* **$all** A reference to an array that will contain the result rows.

Executes an SQL Statement

Executes the given SQL statement and retrieves all result rows as associative arrays, then frees the result set.

- **Example**

bool function AbstractDB::QueryColumn($sql, &$column) *[line 653]*

QueryColumn example.

```
1     <?php
2     /**
3     * QueryColumn Example
4     *
5     * This example uses the QueryColumn function to query a database and retrieve the values in the
6     * first column of the result set.
7     * @package AbstractDB
8     * @author Pacific-Cybersoft
9     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
10    * @version v 1.0.2
11    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
12    */
```

```
13
14      /**
15      * Include the AbstractDB main class.
16      */
17      include("../abstractdb.class.php"          );
18
19      // Instatiate AbstractDB
20      $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
21      // Execute a Query
22      if($db->   QueryColumn("SELECT * FROM tablename"          , $column))
23      {
24          // Do something with the data in $column
25          foreach($column as $key=>   $value)
26              echo("     Value at index $k is $v\r\n"     );
27      }
28      else
29      {
30          die($db->   GetLastError());
31      }
32      // Always good practice to close AbstractDB at the end of each script.
33      $db->   Close();
34      ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.
- *array* **$column** A reference to an array that will contain the result column rows.

Executes an SQL Statement

Executes the given SQL statement and retrieves all rows of the first result column, then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::QueryField($sql, &$field) *[line 673]*

QueryField example.

```
1       <?php
2       /**
3       * QueryField Example
4       *
5       * This example uses the QueryField function to query a database and retrieve the value of the
6       * first column of the first row of a result set.
7       * @package AbstractDB
8       * @author Pacific-Cybersoft
9       * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
10      * @version v 1.0.2
11      * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
12      */
13
14      /**
```

```
15    * Include the AbstractDB main class.
16    */
17    include("../abstractdb.class.php"          );
18
19    // Instatiate AbstractDB
20    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
21    // Execute a Query
22    if($db->   QueryField("SELECT * FROM tablename LIMIT 0,1"          , $field))
23    {
24        // Do something with $field
25        echo($field);
26    }
27    else
28    {
29        die($db->   GetLastError());
30    }
31    // Always good practice to close AbstractDB at the end of each script.
32    $db->   Close();
33    ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.
- *mixed* **$field** A reference to a variable that will contain the field value.

Executes an SQL Statement

Executes the given SQL statement and retrieves the value in the first column of the first row, then frees the result set.

- **Example**
- **Access** public

bool function AbstractDB::QueryObject($sql, &$object) *[line 692]*

Query example.

```
1     <?php
2     /**
3     * Query Example
4     *
5     * This example uses the Query[Assoc, Objects, Row] functions to query a database and retrieve the
6     * first result. The method shown in this example uses QueryRow but the other functions would
7     * be used in the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
```

```
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($db->   QueryRow("SELECT * FROM tablename LIMIT 0,1"          , $row))
24    {
25        // Do something with the data in $row
26        $field0 = $row[0];
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.
- *object* **$object** A reference to an object that will contain the result row.

Executes an SQL Statement

Executes the given SQL statement and retrieves the first result row as an object, then frees the result set.

- **Example**

bool function AbstractDB::QueryObjectAll($sql, &$all) *[line 711]*

Query All example.

```
1     <?php
2     /**
3     * Query All Example
4     *
5     * This example uses the Query[Assoc, Object, Row]All functions to retrieve the all query results.
6     * The method shown in this example uses QueryRowAll but the other functions would be used in the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
```

```
t;));
22    // Execute a Query
23    if($rs = $db->   QueryRowAll("SELECT * FROM tablename"          , $all))
24    {
25        // Do something with the data in $all.
26        foreach($all as $row)
27        {
28            print_r($row);
29        }
30    }
31    else
32    {
33        die($db->   GetLastError());
34    }
35    // Always good practice to close AbstractDB at the end of each script.
36    $db->   Close();
37    ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.
- *array* **$all** A reference to an array that will contain the result rows.

Executes an SQL Statement

Executes the given SQL statement and retrieves all result rows as objects, then frees the result set.

- **Example**

bool function AbstractDB::QueryRow($sql, &$row) *[line 730]*

Query example.

```
1     <?php
2     /**
3     * Query Example
4     *
5     * This example uses the Query[Assoc, Objects, Row] functions to query a database and retrieve the
6     * first result. The method shown in this example uses QueryRow but the other functions would
7     * be used in the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
```

```
23    if($db->   QueryRow("SELECT * FROM tablename LIMIT 0,1"          , $row))
24    {
25        // Do something with the data in $row
26        $field0 = $row[0];
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.
- *array* **$row** A reference to an array that will contain the result row.

Executes an SQL Statement

Executes the given SQL statement and retrieves the first result row, then frees the result set.

- **Example**

bool function AbstractDB::QueryRowAll($sql, &$all) *[line 748]*

Query All example.

```
1     <?php
2     /**
3     * Query All Example
4     *
5     * This example uses the Query[Assoc, Object, Row]All functions to retrieve the all query results.
6     * The method shown in this example uses QueryRowAll but the other functions would be used in the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   QueryRowAll("SELECT * FROM tablename"          , $all))
24    {
25        // Do something with the data in $all.
26        foreach($all as $row)
27        {
```

```
28            print_r($row);
29        }
30    }
31    else
32    {
33        die($db->   GetLastError());
34    }
35    // Always good practice to close AbstractDB at the end of each script.
36    $db->   Close();
37    ?>
```

Function Parameters:

- *string* **$sql** The SQL statement to be executed.
- *array* **$all** A reference to an array that will contain the result rows.

Executes an SQL Statement

Executes the given SQL statement and retrieves all result rows, then frees the result set.

- **Example**

bool function AbstractDB::Replace($table, $fields) *[line 776]*

Replace example.

```
1     <?php
2     /**
3     * Replace Example
4     *
5     * This example show how to use the Replace method.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20
21    $fields = array(
22        "PrimaryKeyField"          =>array(   "Type"          =>   "numeric"          ,
"Key"          =>   true, "Value"          =>   1, "Null"          =>   false),
23        "Field2"          =>array(   "Type"          =>   "text"          , "Key"          =>   false,
"Value"          =>   "replaced value"          , "Null"          =>   false),
24        "Field3"          =>array(   "Type"          =>   "text"          , "Key"          =>   false,
"Value"          =>   ""          , "Null"          =>   true));
25
26    // Execute a REPLACE Query
27    if(!$db->   Replace("tablename"          , $fields))
```

```
28        die($db->   GetLastError());
29    // Always good practice to close AbstractDB at the end of each script.
30    $db->   Close();
31    ?>
```

Function Parameters:

- *string* **$table** The name of the table to execute the replace query on.
- *array* **$fields** An associative array of field definitions. Keys should be the field names and values should be an associative array containing the following keys:

  Key => bool indicating that this field is the primary key or part of a unique index. Key values must not be NULL.

  Type => either "text", "numeric", "bool".

  Value => the value of the field.

  Null => bool indicating if the value of the field should be set to NULL.

  e.g. $fields = array("Field1" => array("Key" => true,
  "Type" => "numeric", "Value" => 123, "Null"
  => false));

Executes an SQL Replace Query

- **Example**
- **Access** public

string function AbstractDB::SetDatabase($dbName) *[line 790]*

Function Parameters:

- *string* **$dbName** The name of the new database to perform queries on.

Sets the Current Database

- **Access** public

function AbstractDB::SetError($scope, $message) *[line 1065]*

Function Parameters:

- *string* **$scope** The scope of the error, generally the function in which it occured.
- *string* **$message** The actual error message.

Sets the Latest Error Message.

Sets the latest error message and calls an error handling function if one was set.

- **Access** private

bool function AbstractDB::SetErrorHandler($functionName) *[line 807]*

SetErrorHandler example.

```
1     <?php
2     /**
3     * SetErrorHandler Example
4     *
5     * This example show how to set an error handler for AbstractDB.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    function ExampleErrorHandler($source, $error)
19    {
20        // Do something with the info.
21        // $source will be a copy of the AbstractDB object.
22        print_r($source);
23        // $error will be an associative array containing error scope and message.
24        echo($error["Scope"          ] . " - "           . $error["Message"          ]);
25    }
26
27    // Instatiate AbstractDB
28    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
29    // Set the error handler
30    $db->   SetErrorHandler("ExampleErrorHandler"          ) or die($db->   GetLastError());
31    // Now if we execute a bad query or whenever an error occurs, ExampleErrorHandler will be called.
```

```
32    $db->   Query("SELECT NonExistentColumn from BadTableName"          );
33    // Always good practice to close AbstractDB at the end of each script.
34    $db->   Close();
35    ?>
```

Function Parameters:

- *string* **$functionName** The name of a function to be called when an error occurs.

Sets an Error Handling Function.

- **Example**
- **Access** public

Class AbstractDB_Driver

[line 27]

AbstractDB Driver Base Class

Driver base class from which all other AbstractDB drivers extend.

- **Package** AbstractDB
- **Abstract Element**

AbstractDB_Driver::$Support

array = [line 55]

List of supported features of the currently loaded driver.

- **Access** public

AbstractDB_Driver::$_arguments

array = [line 35]

List of connection and driver specific arguments.

- **Access** private

AbstractDB_Driver::$_connection

resource = [line 41]

Database connection handle.

- **Access** private

AbstractDB_Driver::$_error

string = [line 47]

The last error message.

- **Access** private

Constructor function AbstractDB_Driver::AbstractDB_Driver($arguments) *[line 66]*

Function Parameters:

- *array* **$arguments** A list of connection and driver specific arguments. See AbstractDB for details concerning connection arguments.

AbstractDB Driver Constructor

Initilises an instance of the AbstractDB Driver base class.

- **Internal** Stores the arguments parameter as a local field.
- **Access** public

int function AbstractDB_Driver::AffectedRows() *[line 83]*

Gets the Number of Affected Rows

Gets the number of rows affected by the last query.

- **Internal** This method must be overriden in extended classes with a call to parent if this feature is supported.

  If this feature is not supported an error should be set explaining this.

- **Access** public

function AbstractDB_Driver::ClearError() *[line 331]*

Clears the latest error message.

- **Access** private

bool function AbstractDB_Driver::Close() *[line 97]*

Closes the Database Connection

- **Internal** This method must be overriden in extended classes with a call to parent if this feature is supported.

  If this feature is not supported this method should return a default value of true.

- **Access** public

bool function AbstractDB_Driver::Connect() *[line 350]*

Opens a Database Connection

Attempts to connect to the database using the parameters given in the constructor.

- **Internal** This method must be overriden in extended classes or it will cause the script to exit.

  The _connection field should be set to hold the resource link.

  If the connection fails an error should be set that explains as best as possible the reason for the connection failure.

- **Access** private

bool function AbstractDB_Driver::DataSeek(&$rs, $row_num) *[line 113]*

Function Parameters:

- *resource* **$rs** A reference to a result handle returned by executing a query.
- *int* **$row_num** The 0 based index of the row that the result pointer should move to.

Move a Result Pointer to the Specified Row

- **Internal** This method must be overriden in extended classes with a call to parent if this feature is supported.

If this feature is not supported an error should be set explaining this.

- **Access** public

array function AbstractDB_Driver::FetchAssoc(&$rs) *[line 128]*

Function Parameters:

- *resource* **&$rs** A reference to a resource handle returned by executing a query.

Fetches a Result Row as an Associative Array

- **Internal** This method must be overriden in extended classes with a call to parent if this feature is supported.

  If this feature is not supported an error should be set explaining this.

- **Access** public

mixed function AbstractDB_Driver::FetchField(&$rs) *[line 143]*

Function Parameters:

- *resource* **&$rs** A reference to a resource handle returned by executing a query.

Fetches the First Field Value

Fetches the value from the first field of a result row.

- **Internal** This method must be overriden in extended classes with a call to parent.
- **Access** public

object Returns function AbstractDB_Driver::FetchObject(&$rs) *[line 158]*

Function Parameters:

- *resource* **&$rs** A reference to a resource handle returned by executing a query.

Fetches a Result Row as an Object

- **Internal** This method must be overriden in extended classes with a call to parent if this feature is supported.

  If this feature is not supported an error should be set explaining this.

- **Access** public

array function AbstractDB_Driver::FetchRow(&$rs) *[line 171]*

Function Parameters:

- *resource* **&$rs** A reference to a resource handle returned by executing a query.

Fetches a Result Row

- **Internal** This method must be overriden in extended classes with a call to parent.
- **Access** public

int function AbstractDB_Driver::FieldCount(&$rs) *[line 186]*

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.

Gets the Number of Fields

Gets the number of fields returned by given result handle.

- **Internal** This method must be overriden in extended classes with a call to parent.
- **Access** public

bool function AbstractDB_Driver::FieldNames(&$rs, &$fields) *[line 200]*

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.
- *array* **$fields** A reference to an array that will contain the field names.

Gets the Field Names of a Query

- **Internal** This method must be overriden in extended classes with a call to parent.
- **Access** public

bool function AbstractDB_Driver::FreeResult(&$rs) *[line 217]*

Function Parameters:

- *resource* **&$rs** A reference to a resource handle returned by executing a query.

Frees a Result Resource

Frees the resources associated with the given result handle.

- **Internal** This method must be overriden in extended classes with a call to parent if this feature is supported.

  If this feature is not supported the method should return a default value of true.

- **Access** public

string function AbstractDB_Driver::GetLastError() *[line 228]*

Gets the Last Error.

- **Access** public

mixed function AbstractDB_Driver::InsertID() *[line 244]*

Gets the Last Inserted AUTO_INCREMENT ID

Gets the ID of the last AUTO_INCREMENT record inserted into the databse.

- **Internal** This method must be overriden in extended classes with a call to parent if this feature is supported.

  If this feature is not supported an error should be set explaining this.

- **Access** public

resource function AbstractDB_Driver::Query($sql) *[line 264]*

Function Parameters:

- *string* **$sql** The SQL statement to execute on the database.

Executes an SQL Statement.

Executes an SQL statement passed in as a parameter.

- **Internal** This method must be overriden in extended classes or it will cause the script to exit.

  If the query fails an error should be set that explains as best as possible the reason for the query failure.

- **Access** public

resource function AbstractDB_Driver::Replace($table, $fields) *[line 290]*

Function Parameters:

- *string* **$table** The name of the table to execute the replace query on.
- *array* **$fields** An associative array of field definitions. Keys should be the field names and values should be an associative array containing the following keys:

  Key => bool indicating that this field is the primary key or part of a unique index. Key values must not be NULL.

  Type => either "text", "numeric", "bool".

  Value => the value of the field.

  Null => bool indicating if the value of the field should be set to NULL.

  e.g. $fields = array("Field1" => array("Key" => true,
  "Type" => "numeric", "Value" => 123, "Null"
  => false));

Executes an SQL Replace Query

- **Internal** This method must be overriden in extended classes with a call to parent if this feature is supported.

  If this feature is not supported an error should be set explaining this.

- **Access** public

int function AbstractDB_Driver::RowCount(&$rs) *[line 305]*

Function Parameters:

- *resource* **$rs** A reference to a resource handle returned by executing a query.

Gets the Number of Rows

Gets the number of rows returned by given result handle.

- **Internal** This method must be overriden in extended classes with a call to parent.
- **Access** public

mixed function AbstractDB_Driver::SetDatabase($dbName) *[line 320]*

Function Parameters:

- *string* **$dbName** The name of the database to set active.

Sets the Current Active Database

- **Internal** This method must be overriden in extended classes with a call to parent.

  If this operation fails an error should be set explaining as best as possible the reason for the failure.

- **Access** public

function AbstractDB_Driver::SetError($scope, $message) *[line 361]*

Function Parameters:

- *string* **$scope** The scope of the error, generally the function in which it occured.
- *string* **$message** The actual error message.

Sets the Error Message

- **Access** private

Class AbstractDB_MySQL

[line 25]

AbstractDB MySQL Driver Class

- **Package** AbstractDB
- **Access** public

Constructor function AbstractDB_MySQL::AbstractDB_MySQL($arguments) *[line 34]*

AbstractDB MySQL Driver Constructor

Initilises an instance of the AbstractDB MySQL Driver class.

- **Internal** Calls the AbstractDB_Driver constructor passing the given parameters and sets the values of the Support list.

function AbstractDB_MySQL::AffectedRows() *[line 46]*
function AbstractDB_MySQL::Close() *[line 52]*
function AbstractDB_MySQL::Connect() *[line 243]*
function AbstractDB_MySQL::DataSeek(&$rs, $row_num) *[line 75]*
function AbstractDB_MySQL::FetchAssoc(&$rs) *[line 81]*
function AbstractDB_MySQL::FetchField(&$rs) *[line 87]*
function AbstractDB_MySQL::FetchObject(&$rs) *[line 96]*
function AbstractDB_MySQL::FetchRow(&$rs) *[line 102]*
function AbstractDB_MySQL::FieldCount(&$rs) *[line 108]*
function AbstractDB_MySQL::FieldNames(&$rs, &$fields) *[line 114]*
function AbstractDB_MySQL::FreeResult(&$rs) *[line 129]*
function AbstractDB_MySQL::InsertID() *[line 144]*
function AbstractDB_MySQL::Query($sql) *[line 154]*
function AbstractDB_MySQL::Replace($table, $fields) *[line 166]*
function AbstractDB_MySQL::RowCount(&$rs) *[line 222]*
function AbstractDB_MySQL::SetDatabase($dbName) *[line 228]*

Appendices

Appendix A - Class Trees

Package AbstractDB

AbstractDB

- AbstractDB

AbstractDB_Driver

- AbstractDB_Driver
  - AbstractDB_MySQL

Appendix B - README/CHANGELOG/INSTALL

CHANGELOG

version 1.0.2 (Released TBA)

- Removed the result handle parameter from GetAffectedRows. This parameter was never required.
- Private method IsResource now sets an error if the $resource parameter is not a valid resource.
- Added a method to main class and drivers to get the values in the first column of a result set; FetchResultColumn()
- Added a method to main class and drivers to query and get the values in the first column of a result set; QueryColumn()
- Added a method to main class and drivers to get the value in the first column of the first row of a result set; FetchResultField()
- Added a method to main class and drivers to query and get the value in the first column of the first row of a result set; QueryField()
- Changed license from GNU General Public License to GNU Lesser General Public License. This is due to the fact that AbstractDB is planned to be included in future commercial applications developed by Pacific-Cybersoft.

version 1.0.1 (Released 14/03/2005)

- Added the ability to connect to database servers without a Hostname, Username or Password. These are no longer required arguments.

version 1.0.0 (Released 13/03/2005)

- Fixed a bug in the connection argument parser that was incorrectly parsing the Username and Hostname arguments.
- Better handling of driver results and return values for all methods that fetch result sets.
- Added a private method to the main class that checks a given variable is a resource. Used in all methods that accept a resource as a parameter.
- Removed the $rs parameter from the GetInsertID method. Not needed.
- Added usage examples.

version 0.1.1 Alpha (Released 08/03/2005)

- Added a method to main class and drivers to get the current database name; GetDatabase()
- Added a method to main class and drivers to get the field count; GetFieldCount()
- Added a method to main class and drivers to get the names of field returned by a query; GetFieldNames()
- Added a method to main class and drivers to get the number of rows returned by a query; GetRowCount()
- Added a method to main class and drivers to execute REPLACE queries; Replace()
- Added a method to main class and drivers to set the name of the current active database; SetDatabase()
- Several documentation error ammendments

version 0.1.0 Alpha (Released 06/03/2005)

- initial pre version 1 alpha release with MySQL Driver

INSTALL

AbstractDB Installation Instructions
---

1) Upload all of the files extracted from the distribution file onto your web
server. Ensure that the drivers sub directory is in the same directory as
the abstractdb.class.php and abstractdb_driver.class.php files.

2) Include the abstractdb.class.php file wherever you need to use AbstractDB
functionality.

include("path/to/abstractdb/abstractdb.class.php");

AbstractDB will handle locating and loading the required driver classes.

LICENSE

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it. You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use,
not price. Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get

it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to

encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of

a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may

distribute the object code for the work under the terms of Section 6.
Any executables containing that work also fall under Section 6,
whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or
link a "work that uses the Library" with the Library to produce a
work containing portions of the Library, and distribute that work
under terms of your choice, provided that the terms permit
modification of the work for the customer's own use and reverse
engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the
Library is used in it and that the Library and its use are covered by
this License. You must supply a copy of this License. If the work
during execution displays copyright notices, you must include the
copyright notice for the Library among them, as well as a reference
directing the user to the copy of this License. Also, you must do one
of these things:

a) Accompany the work with the complete corresponding
machine-readable source code for the Library including whatever
changes were used in the work (which must be distributed under
Sections 1 and 2 above); and, if the work is an executable linked
with the Library, with the complete machine-readable "work that
uses the Library", as object code and/or source code, so that the
user can modify the Library and then relink to produce a modified
executable containing the modified Library. (It is understood
that the user who changes the contents of definitions files in the
Library will not necessarily be able to recompile the application
to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the
Library. A suitable mechanism is one that (1) uses at run time a
copy of the library already present on the user's computer system,
rather than copying library functions into the executable, and (2)
will operate properly with a modified version of the library, if
the user installs one, as long as the modified version is
interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at
least three years, to give the same user the materials
specified in Subsection 6a, above, for a charge no more
than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy
from a designated place, offer equivalent access to copy the above
specified materials from the same place.

e) Verify that the user has already received a copy of these
materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the
Library" must include any data and utility programs needed for
reproducing the executable from it. However, as a special exception,
the materials to be distributed need not include anything that is
normally distributed (in either source or binary form) with the major
components (compiler, kernel, and so on) of the operating system on
which the executable runs, unless that component itself accompanies
the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to

refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE

LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the
library `Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!

README

AbstractDB - PHP Database Abstraction Class

Developer: Pacific-Cybersoft
Web: http://www.pacific-cybersoft.com

License: GNU LGPL - see LICENSE and further details below.
Version: 1.0.2

AbstractDB is a PHP database abstraction layer, objected oriented and
extensible via driver implementations which extend from a common base class.

This package encapsulates all of the basic functions required by most
developers without the complexity and/or utter simplicity of many other
abstraction packages.

Incorporating AbstractDB into your applications is as easy as including a
single file as such:

include("path/to/abstractdb/abstractdb.class.php");

AbstractDB will handle locating and loading the required driver classes itself.

For installation instructions see the INSTALL file.
For license details see the LICENSE file.
For development history see the CHANGELOG file.

AbstractDB is distributed with a manual generated with phpDocumentor
(http://www.phpdoc.org) which is available in HTML, PDF and CHM (Windows Help)
formats. You can find the manuals in the /docs directory.

AbstractDB is an Open Source project. We are always on the lookout for
developers willing to contribute small amounts of time to the development and
enhancement of this package, especially in the area of driver development. If
you would be interested in contributing please feel free to contact us at
spambox at pacific-cybersoft dot com.

Please also report bugs and/or feature requests to the above mentioned email
address.

details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Known Issues:

- Examples in CHM (Windows Help) documentation are not working.

Appendix C - Source Code

Package AbstractDB

File Source for abstractdb.class.php

Documentation for this file is available at abstractdb.class.php

```
1     <?php
2     /**
3     * @package AbstractDB
4     * @author Pacific-Cybersoft
5     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
6     * @version v 1.0.2
7     * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
8     */
9
10    /**
11    * AbstractDB Main Class
12    *
13    * The main AbstractDB class used to interact with various DBMS packages via the
14    * use of driver classes.
15    * @package AbstractDB
16    * @access public
17    */
18    class AbstractDB
19    {
20        /* PRIVATE FIELDS */
21        /**
22        * List of connection arguments.
23        * @access private
24        * @var array
25        */
26        var $_arguments;
27        /**
28        * Reference to the AbstractDB driver.
29        * @access private
30        * @var object
31        */
32        var $_driver;
33        /**
34        * The last error message.
35        * @access private
36        * @var string
37        */
38        var $_error;
39        /**
40        * Name of an error handling function passed in via {@link SetErrorHandler}.
41        * @access private
42        * @var string
43        */
44        var $_error_handler;
45
46        /* PUBLIC PROPERTIES */
47        /**
48        * List of supported features of the currently loaded driver.
49        * @access public
50        * @var array
51        */
52        var $Support;
53
54        /**
55        * AbstractDB Constructor
56        *
57        * Initilises an instance of the AbstractDB class.
58        * @access public
59        * @internal Parses connection arguments and loads the driver class.
60        * @param array $arguments Database connection and driver specific arguments.
61        *
62        *  Arguments must be supplied as an associative array containing one of the following:
63        *
64        *  ConnectionString:
65        *
66        *  A string composed of the connection arguments and optional or driver specific options in the
form
```

```
67          *  of
<b>Type</b>://<b>Username</b>:<b>Password</b>@<b>Hostname</b>&
gt;:<b>Port</b>/<b>Database</b>?<b>Options</b>/option1=value1&<
b>Options</b>/option2=value2
68          *
69          *  e.g. $args = array("ConnectionString" =>
"mysql://root:pass@localhost/MyDatabase?Options/Persistent=1");
70          *
71          *  or
72          *
73          *  Type: The type of database to connect to. AbstractDB is currently only distributed with a
MySQL driver.
74          *
75          *  Username: The database username.
76          *
77          *  Password: The database password.
78          *
79          *  Hostname: The database hostname or IP Address.
80          *
81          *  Database: The name of the database.
82          *
83          *  Options: An associative array of optional or driver specific options. See individual driver
84          *  documentation for a potential list of driver specific options. The value for Port should be
85          *  specified in this list, otherwise the default port for the database type will be used.
86          *
87          *  e.g. $args = array("Type" => "mysql", "Username" =>
"root", "Password" => "pass", "Hostname" =>
"localhost", "Database" => "MyDatabase", "Options" =>
array("Persistent" => 1));
88          *
89          *  Type and Database are required parameters and must be specified.
90          * @example instantiate_connectionstring.php Instantiation example using a ConnectionString
argument.
91          * @example instantiate_associative.php Instantiation example using an associative array of
arguments.
92          */
93          function AbstractDB($arguments)
94          {
95              $this->   ClearError();
96              if($this->   ParseArguments($arguments))
97                  $this->   LoadDriver();
98          }
99
100         /* PUBLIC FUNCTIONS */
101         /**
102         * Closes the Database Connection
103         * @access public
104         * @return bool Returns true if the database connection was successfully closed, otherwise false.
105         */
106         function Close()
107         {
108             $result = $this->   _driver->   Close();
109             $this->   GetDriverError();
110             return $result;
111         }
112
113         /**
114         * Fetches the Next Row as an Associative Array
115         * @access public
116         * @param resource $rs A reference to a resource handle returned by executing a query.
117         * @param array $assoc A reference to an array that will contain the result row.
118         * @return bool Returns true if the operation was successful, otherwise false.
119         * @example fetch_next_result.php FetchNextResult example.
120         */
121         function FetchNextResultAssoc(&    $rs, &    $assoc)
122         {
123             $this->   ClearError();
124             if($result = $this->   IsResource($rs, "FetchNextResultAssoc"          ))
125             {
126                 if(isset($this->   Support["FetchAssoc"          ]) &&         $this-
>   Support["FetchAssoc"          ])
127                     $result = (($assoc = $this->   _driver->   FetchAssoc($rs)) != false);
128                 else
129                     $reuslt = false;
130                 $this->   GetDriverError();
131             }
132             return $result;
133         }
134
135         /**
```

```
136         * Fetches the Next Row as an Object
137         * @access public
138         * @param resource $rs A reference to a resource handle returned by executing a query.
139         * @param object $object A reference to an object that will contain the result row.
140         * @return bool Returns true if the operation was successful, otherwise false.
141         * @example fetch_next_result.php FetchNextResult example.
142         */
143        function FetchNextResultObject(&    $rs, &    $object)
144        {
145            $this->   ClearError();
146            if($result = $this->   IsResource($rs, "FetchNextResultObject"          ))
147            {
148                if(isset($this->   Support["FetchObject"          ]) &&         $this-
>   Support["FetchObject"          ])
149                    $result = (($object = $this->   _driver->   FetchObject($rs)) != false);
150                else
151                    $reuslt = false;
152                $this->   GetDriverError();
153            }
154            return $result;
155        }
156
157        /**
158         * Fetches the Next Row
159         * @access public
160         * @param resource $rs A reference to a resource handle returned by executing a query.
161         * @param array $row A reference to an array that will contain the result row.
162         * @return bool Returns true if the operation was successful, otherwise false.
163         * @example fetch_next_result.php FetchNextResult example.
164         */
165        function FetchNextResultRow(&    $rs, &    $row)
166        {
167            $this->   ClearError();
168            if($result = $this->   IsResource($rs, "FetchNextResultRow"          ))
169            {
170                $result = (($row = $this->   _driver->   FetchRow($rs)) != false);
171                $this->   GetDriverError();
172            }
173            return $result;
174        }
175
176        /**
177         * Fetches a Row as an Associative Array
178         *
179         * Fetches the first row as an associative array and then frees the result set.
180         * @access public
181         * @param resource $rs A reference to a resource handle returned by executing a query.
182         * @param array $assoc A reference to an array that will contain the result row.
183         * @return bool Returns true if the operation was successful, otherwise false.
184         * @example fetch_result.php FetchResult example.
185         */
186        function FetchResultAssoc(&    $rs, &    $assoc)
187        {
188            $this->   ClearError();
189            if($result = $this->   IsResource($rs, "FetchResultAssoc"          ))
190            {
191                $result = $this->   FetchNextResultAssoc($rs, $assoc);
192                $this->   GetDriverError();
193                $this->   _driver->   FreeResult($rs);
194                $this->   GetDriverError();
195            }
196            return $result;
197        }
198
199        /**
200         * Fetches All Rows as Associative Arrays
201         *
202         * Fetches all rows as associative arrays and then frees the result set.
203         * @access public
204         * @param resource $rs A reference to a resource handle returned by executing a query.
205         * @param array $all A reference to an array that will contain the result rows.
206         * @return bool Returns true if the operation was successful, otherwise false.
207         * @example fetch_result_all.php FetchResultAll example.
208         */
209        function FetchResultAssocAll(&    $rs, &    $all)
210        {
211            $this->   ClearError();
212            if($result = $this->   IsResource($rs, "FetchResultAssocAll"          ))
213            {
214                while($result = $this->   FetchNextResultAssoc($rs, $all[])){}
```

```
215             array_pop($all);
216             $this->   GetDriverError();
217             $result = (strlen($this->   _error) == 0);
218             $this->   _driver->   FreeResult($rs);
219             $this->   GetDriverError();
220         }
221         return $result;
222     }
223 
224     /**
225     * Fetches a Result Column
226     *
227     * Fetches all rows of the first column and then frees the result set.
228     * @access public
229     * @param resource $rs A reference to a resource handle returned by executing a query.
230     * @param array $column A reference to an array that will contain the result column rows.
231     * @return bool Returns true if the operation was successful, otherwise false.
232     * @example fetch_result_column.php FetchResultColumn example.
233     */
234     function FetchResultColumn(&    $rs, &    $column)
235     {
236         $this->   ClearError();
237         if($result = $this->   IsResource($rs, "FetchResultColumn"          ))
238         {
239             while($column[] = $this->   _driver->   FetchField($rs)){}
240             array_pop($column);
241             $this->   GetDriverError();
242             $result = (strlen($this->   _error) == 0);
243             $this->   _driver->   FreeResult($rs);
244             $this->   GetDriverError();
245         }
246         return $result;
247     }
248 
249     /**
250     * Fetches a Result Field
251     *
252     * Fetches the value in the first column of the first row and then frees the result set.
253     * @access public
254     * @param resource $rs A reference to a resource handle returned by executing a query.
255     * @param mixed $field A reference to a variable that will contain the field value.
256     * @return bool Returns true if the operation was successful, otherwise false.
257     * @example fetch_result_field.php FetchResultField example.
258     */
259     function FetchResultField(&    $rs, &    $field)
260     {
261         $this->   ClearError();
262         if($result = $this->   IsResource($rs, "FetchResultField"          ))
263         {
264             $result = (($field = $this->   _driver->   FetchField($rs)) != false);
265             $this->   GetDriverError();
266             $this->   _driver->   FreeResult($rs);
267             $this->   GetDriverError();
268         }
269         return $result;
270     }
271 
272     /**
273     * Fetches a Row as an Object
274     *
275     * Fetches the first row as an object and then frees the result set.
276     * @access public
277     * @param resource $rs A reference to a resource handle returned by executing a query.
278     * @param object $object A reference to an object that will contain the result row.
279     * @return bool Returns true if the operation was successful, otherwise false.
280     * @example fetch_result.php FetchResult example.
281     */
282     function FetchResultObject(&    $rs, &    $object)
283     {
284         $this->   ClearError();
285         if($result = $this->   IsResource($rs, "FetchResultObject"          ))
286         {
287             $result = $this->   FetchNextResultObject($rs, $object);
288             $this->   GetDriverError();
289             $this->   _driver->   FreeResult($rs);
290             $this->   GetDriverError();
291         }
292         return $result;
293     }
294 
```

```
295         /**
296         * Fetches All Rows as Objects
297         *
298         * Fetches all rows as objects and then frees the result set.
299         * @access public
300         * @param resource $rs A reference to a resource handle returned by executing a query.
301         * @param array $all A reference to an array that will contain the result rows.
302         * @return bool Returns true if the operation was successful, otherwise false.
303         * @example fetch_result_all.php FetchResultAll example.
304         */
305         function FetchResultObjectAll(&    $rs, &    $all)
306         {
307             $this->   ClearError();
308             if($result = $this->   IsResource($rs, "FetchResultObjectAll"          ))
309             {
310                 while($result = $this->   FetchNextResultObject($rs, $all[])){}
311                 array_pop($all);
312                 $this->   GetDriverError();
313                 $result = (strlen($this->   _error) == 0);
314                 $this->   _driver->   FreeResult($rs);
315                 $this->   GetDriverError();
316             }
317             return $result;
318         }
319
320         /**
321         * Fetches a Row
322         *
323         * Fetches the first row and then frees the result set.
324         * @access public
325         * @param resource $rs A reference to a resource handle returned by executing a query.
326         * @param array $row A reference to an array that will contain the result row.
327         * @return bool Returns true if the operation was successful, otherwise false.
328         * @example fetch_result.php FetchResult example.
329         */
330         function FetchResultRow(&    $rs, &    $row)
331         {
332             $this->   ClearError();
333             if($result = $this->   IsResource($rs, "FetchResultRow"          ))
334             {
335                 $result = $this->   FetchNextResultRow($rs, $row);
336                 $this->   GetDriverError();
337                 $this->   _driver->   FreeResult($rs);
338                 $this->   GetDriverError();
339             }
340             return $result;
341         }
342
343         /**
344         * Fetches All Rows
345         *
346         * Fetches all rows and then frees the result set.
347         * @access public
348         * @param resource $rs A reference to a resource handle returned by executing a query.
349         * @param array $all A reference to an array that will contain the result rows.
350         * @return bool Returns true if the operation was successful, otherwise false.
351         * @example fetch_result_all.php FetchResultAll example.
352         */
353         function FetchResultRowAll(&    $rs, &    $all)
354         {
355             $this->   ClearError();
356             if($result = $this->   IsResource($rs, "FetchResultRowAll"          ))
357             {
358                 while($result = $this->   FetchNextResultRow($rs, $all[])){}
359                 array_pop($all);
360                 $this->   GetDriverError();
361                 $result = (strlen($this->   _error) == 0);
362                 $this->   _driver->   FreeResult($rs);
363                 $this->   GetDriverError();
364             }
365             return $result;
366         }
367
368         /**
369         * Fetches a Row as an Associative Array
370         *
371         * Fetches the row at the specified position in the result set as an associative array.
372         * @access public
373         * @param resource $rs A reference to a resource handle returned by executing a query.
374         * @param int $row_num The position in the result set of the row to return.
```

```
375         * @param array $assoc A reference to an array that will contain the result row.
376         * @return bool Returns true if the operation was successful, otherwise false.
377         * @example fetch_seek_result.php FetchSeekResult example.
378         */
379         function FetchSeekResultAssoc(&    $rs, $row_num, &    $assoc)
380         {
381             $this->   ClearError();
382             if($result = $this->   IsResource($rs, "FetchSeekResultAssoc"          ))
383             {
384                 if(isset($this->   Support["DataSeek"          ]) &&         $this-
>   Support["DataSeek"          ])
385                 {
386                     if($result = $this->   _driver->   DataSeek($rs, $row_num))
387                         $result = $this->   FetchNextResultAssoc($rs, $assoc);
388                 }
389                 else
390                     $reuslt = false;
391                 $this->   GetDriverError();
392             }
393             return $result;
394         }
395
396         /**
397         * Fetches a Row as an Object
398         *
399         * Fetches the row at the specified position in the result set as an object.
400         * @access public
401         * @param resource $rs A reference to a resource handle returned by executing a query.
402         * @param int $row_num The position in the result set of the row to return.
403         * @param object $object A reference to an object that will contain the result row.
404         * @return bool Returns true if the operation was successful, otherwise false.
405         * @example fetch_seek_result.php FetchSeekResult example.
406         */
407         function FetchSeekResultObject(&    $rs, $row_num, &    $object)
408         {
409             $this->   ClearError();
410             if($result = $this->   IsResource($rs, "FetchSeekResultObject"          ))
411             {
412                 if(isset($this->   Support["DataSeek"          ]) &&         $this-
>   Support["DataSeek"          ])
413                 {
414                     if($result = $this->   _driver->   DataSeek($rs, $row_num))
415                         $result = $this->   FetchNextResultObject($rs, $object);
416                 }
417                 else
418                     $reuslt = false;
419                 $this->   GetDriverError();
420             }
421             return $result;
422         }
423
424         /**
425         * Fetches a Row
426         *
427         * Fetches the row at the specified position in the result set.
428         * @access public
429         * @param resource $rs A reference to a resource handle returned by executing a query.
430         * @param int $row_num The position in the result set of the row to return.
431         * @param array $row A reference to an array that will contain the result row.
432         * @return bool Returns true if the operation was successful, otherwise false.
433         * @example fetch_seek_result.php FetchSeekResult example.
434         */
435         function FetchSeekResultRow(&    $rs, $row_num, &    $row)
436         {
437             $this->   ClearError();
438             if($result = $this->   IsResource($rs, "FetchSeekResultRow"          ))
439             {
440                 if(isset($this->   Support["DataSeek"          ]) &&         $this-
>   Support["DataSeek"          ])
441                 {
442                     if($result = $this->   _driver->   DataSeek($rs, $row_num))
443                         $result = $this->   FetchNextResultRow($rs, $row);
444                 }
445                 else
446                     $reuslt = false;
447                 $this->   GetDriverError();
448             }
449             return $result;
450         }
451
```

```
452         /**
453         * Frees a Result Resource
454         *
455         * Frees the resources associated with the given result handle returned by executing a query.
456         * @access public
457         * @param resource $rs A reference to a resource handle returned by executing a query.
458         * @return bool Returns true if the resource handle was successfully freed.
459         */
460         function FreeResult(&    $rs)
461         {
462             if($result = $this->   IsResource($rs, "FreeResult"          ))
463             {
464                 $result = $this->   _driver->   FreeResult($rs);
465                 $this->   GetDriverError();
466             }
467             return $result;
468         }
469
470         /**
471         * Gets the Number of Affected Rows
472         * @access public
473         * @return int Returns the number of rows affected by the last executed query, or false if the
474         *  driver does not support this feature.
475         * @example affected_rows.php GetAffectedRows example.
476         */
477         function GetAffectedRows()
478         {
479             $this->   ClearError();
480             if(isset($this->   Support["AffectedRows"          ]) &&         $this-
>   Support["AffectedRows"          ])
481                 $result = $this->   _driver->   AffectedRows();
482             else
483                 $result = false;
484             $this->   GetDriverError();
485             return $result;
486         }
487
488         /**
489         * Gets the Name of the Current Database
490         * @access public
491         * @return string The name of the current database.
492         */
493         function GetDatabase()
494         {
495             return $this->   _arguments["Database"          ];
496         }
497
498         /**
499         * Gets the Number of Fields
500         *
501         * Gets the number of fields returned by the given result handle.
502         * @access public
503         * @param resource $rs A reference to a resource handle returned by executing a query.
504         * @return int Returns the number of fields returned by the last executed query.
505         * @example field_count.php GetFieldCount example.
506         */
507         function GetFieldCount(&    $rs)
508         {
509             $this->   ClearError();
510             if($result = $this->   IsResource($rs, "GetFieldCount"          ))
511             {
512                 $result = $this->   _driver->   FieldCount($rs);
513                 $this->   GetDriverError();
514             }
515             return $result;
516         }
517
518         /**
519         * Gets the Field Names of a Query
520         * @access public
521         * @param resource $rs A reference to a resource handle returned by executing a query.
522         * @param array $fields A reference to an array that will contain the field names.
523         * @return bool Returns true if the operation was successful, otherwise false.
524         * @example field_names.php GetFieldNames example.
525         */
526         function GetFieldNames(&    $rs, &    $fields)
527         {
528             $this->   ClearError();
529             if($result = $this->   IsResource($rs, "GetFieldNames"          ))
530             {
```

```
531             $result = $this->   _driver->   FieldNames($rs, $fields);
532             $this->   GetDriverError();
533         }
534         return $result;
535     }
536
537     /**
538     * Gets the Last Insert ID
539     * @access public
540     * @return mixed Returns either the ID of the last inserted AUTO_INCREMENT record, or -1 if the
541     *  last query was not an insert, or false if the driver does not support this feature.
542     * @example insert_id.php GetInsertID example.
543     */
544     function GetInsertID()
545     {
546         $this->   ClearError();
547         if(isset($this->   Support["InsertID"          ]) &&         $this-
>   Support["InsertID"          ])
548             $result = $this->   _driver->   InsertID();
549         else
550             $result = false;
551         $this->   GetDriverError();
552         return $result;
553     }
554
555     /**
556     * Gets the Last Error.
557     * @access public
558     * @return string The last error message.
559     */
560     function GetLastError()
561     {
562         return $this->   _error;
563     }
564
565     /**
566     * Gets the Number of Rows
567     *
568     * Gets the number of rows returned by the last query of the given result handle.
569     * @access public
570     * @param resource $rs A reference to a resource handle returned by executing a query.
571     * @return int Returns the number of rows returned by the last executed query.
572     * @example row_count.php GetRowCount example.
573     */
574     function GetRowCount(&    $rs)
575     {
576         $this->   ClearError();
577         if($result = $this->   IsResource($rs, "GetRowCount"          ))
578         {
579             $result = $this->   _driver->   RowCount($rs);
580             $this->   GetDriverError();
581         }
582         return $result;
583     }
584
585     /**
586     * Executes an SQL Statement
587     *
588     * Executes the given SQL statement.
589     * @access public
590     * @internal If the query did not execute successfully an error is set explaining as
591     *  best as possible the reason for the failure.
592     * @param string $sql The SQL statement to be executed.
593     * @return resource The result handle for use in fetch result functions.
594     * @example query.php Query example.
595     */
596     function Query($sql)
597     {
598         $this->   ClearError();
599         $result = &    $this->   _driver->   Query($sql);
600         $this->   GetDriverError();
601         return $result;
602     }
603
604     /**
605     * Executes an SQL Statement
606     *
607     * Executes the given SQL statement and retrieves the first result row as an associative array,
then
608     * frees the result set.
```

```
609         * @param string $sql The SQL statement to be executed.
610         * @param array $assoc A reference to an array that will contain the result row.
611         * @return bool Returns true if the operation was successful, otherwise false.
612         * @example query_result.php Query example.
613         */
614        function QueryAssoc($sql, &    $assoc)
615        {
616            $this->   ClearError();
617            $result = false;
618            if($rs = &    $this->   Query($sql))
619                $result = $this->   FetchResultAssoc($rs, $assoc);
620            return $result;
621        }
622
623        /**
624        * Executes an SQL Statement
625        *
626        * Executes the given SQL statement and retrieves all result rows as associative arrays, then
627        * frees the result set.
628        * @param string $sql The SQL statement to be executed.
629        * @param array $all A reference to an array that will contain the result rows.
630        * @return bool Returns true if the operation was successful, otherwise false.
631        * @example query_result_all.php Query All example.
632        */
633        function QueryAssocAll($sql, &    $all)
634        {
635            $this->   ClearError();
636            $result = false;
637            if($rs = &    $this->   Query($sql))
638                $result = $this->   FetchResultAssocAll($rs, $all);
639            return $result;
640        }
641
642        /**
643        * Executes an SQL Statement
644        *
645        * Executes the given SQL statement and retrieves all rows of the first result column, then
646        * frees the result set.
647        * @access public
648        * @param string $sql The SQL statement to be executed.
649        * @param array $column A reference to an array that will contain the result column rows.
650        * @return bool Returns true if the operation was successful, otherwise false.
651        * @example query_column.php QueryColumn example.
652        */
653        function QueryColumn($sql, &    $column)
654        {
655            $this->   ClearError();
656            $result = false;
657            if($rs = &    $this->   Query($sql))
658                $result = $this->   FetchResultColumn($rs, $column);
659            return $result;
660        }
661
662        /**
663        * Executes an SQL Statement
664        *
665        * Executes the given SQL statement and retrieves the value in the first column of the first
666        * row, then frees the result set.
667        * @access public
668        * @param string $sql The SQL statement to be executed.
669        * @param mixed $field A reference to a variable that will contain the field value.
670        * @return bool Returns true if the operation was successful, otherwise false.
671        * @example query_field.php QueryField example.
672        */
673        function QueryField($sql, &    $field)
674        {
675            $this->   ClearError();
676            $result = false;
677            if($rs = &    $this->   Query($sql))
678                $result = $this->   FetchResultField($rs, $field);
679            return $result;
680        }
681
682        /**
683        * Executes an SQL Statement
684        *
685        * Executes the given SQL statement and retrieves the first result row as an object, then
686        * frees the result set.
687        * @param string $sql The SQL statement to be executed.
688        * @param object $object A reference to an object that will contain the result row.
```

```
689         * @return bool Returns true if the operation was successful, otherwise false.
690         * @example query_result.php Query example.
691         */
692        function QueryObject($sql, &    $object)
693        {
694            $this->   ClearError();
695            $result = false;
696            if($rs = &    $this->   Query($sql))
697                $result = $this->   FetchResultObject($rs, $object);
698            return $result;
699        }
700
701        /**
702         * Executes an SQL Statement
703         *
704         * Executes the given SQL statement and retrieves all result rows as objects, then
705         * frees the result set.
706         * @param string $sql The SQL statement to be executed.
707         * @param array $all A reference to an array that will contain the result rows.
708         * @return bool Returns true if the operation was successful, otherwise false.
709         * @example query_result_all.php Query All example.
710         */
711        function QueryObjectAll($sql, &    $all)
712        {
713            $this->   ClearError();
714            $result = false;
715            if($rs = &    $this->   Query($sql))
716                $result = $this->   FetchResultObjectAll($rs, $all);
717            return $result;
718        }
719
720        /**
721         * Executes an SQL Statement
722         *
723         * Executes the given SQL statement and retrieves the first result row, then
724         * frees the result set.
725         * @param string $sql The SQL statement to be executed.
726         * @param array $row A reference to an array that will contain the result row.
727         * @return bool Returns true if the operation was successful, otherwise false.
728         * @example query_result.php Query example.
729         */
730        function QueryRow($sql, &    $row)
731        {
732            $this->   ClearError();
733            $result = false;
734            if($rs = &    $this->   Query($sql))
735                $result = $this->   FetchResultRow($rs, $row);
736            return $result;
737        }
738
739        /**
740         * Executes an SQL Statement
741         *
742         * Executes the given SQL statement and retrieves all result rows, then frees the result set.
743         * @param string $sql The SQL statement to be executed.
744         * @param array $all A reference to an array that will contain the result rows.
745         * @return bool Returns true if the operation was successful, otherwise false.
746         * @example query_result_all.php Query All example.
747         */
748        function QueryRowAll($sql, &    $all)
749        {
750            $this->   ClearError();
751            $result = false;
752            if($rs = &    $this->   Query($sql))
753                $result = $this->   FetchResultRowAll($rs, $all);
754            return $result;
755        }
756
757        /**
758         * Executes an SQL Replace Query
759         * @access public
760         * @param string $table The name of the table to execute the replace query on.
761         * @param array $fields An associative array of field definitions. Keys should be the field
names and
762         *  values should be an associative array containing the following keys:
763         *
764         *  Key => bool indicating that this field is the primary key or part of a unique index. Key
values must not be NULL.
765         *
766         *  Type => either "text", "numeric", "bool".
```

```
767         *
768         *  Value => the value of the field.
769         *
770         *  Null => bool indicating if the value of the field should be set to NULL.
771         *
772         *  e.g. $fields = array("Field1" => array("Key" => true,
"Type" => "numeric", "Value" => 123, "Null" => false));
773         * @return bool Returns true if the operation was successful, otherwise false.
774         * @example replace.php Replace example.
775         */
776         function Replace($table, $fields)
777         {
778             $this->   ClearError();
779             $result = $this->   _driver->   Replace($table, $fields) ? true : false;
780             $this->   GetDriverError();
781             return $result;
782         }
783
784         /**
785         * Sets the Current Database
786         * @access public
787         * @param string $dbName The name of the new database to perform queries on.
788         * @return string The old database name that was set before calling this method, or false if an
error occurred.
789         */
790         function SetDatabase($dbName)
791         {
792             $this->   ClearError();
793             $result = $this->   _driver->   SetDatabase($dbName);
794             if($result !== false &&         $result != $this->   _arguments["Database"          ])
795                 $this->   _arguments["Database"          ] = $result;
796             $this->   GetDriverError();
797             return $result;
798         }
799
800         /**
801         * Sets an Error Handling Function.
802         * @access public
803         * @param string $functionName The name of a function to be called when an error occurs.
804         * @return bool Returns true if the error handling function was successfully set, otherwise
false.
805         * @example set_error_handler.php SetErrorHandler example.
806         */
807         function SetErrorHandler($functionName)
808         {
809             if(function_exists($functionName))
810             {
811                 $this->   _error_handler = $functionName;
812                 return true;
813             }
814             else
815             {
816                 $this->   SetError("SetErrorHandler"          , "     Could not set error handler,
function '$functionName' does not exist."     );
817                 return false;
818             }
819         }
820
821         /* PRIVATE FUNCTIONS */
822         /**
823         * Clears the latest error message.
824         * @access private
825         */
826         function ClearError()
827         {
828             $this->   _error = ""          ;
829         }
830
831         /**
832         * Ensures Required Connection Arguments Exist
833         * @access private
834         * @internal If there are missing arguments an error is set stating which
835         *  arguments are missing.
836         * @return bool Returns true if all required connection arguments have been set, otherwise false.
837         */
838         function EnsureRequiredArguments()
839         {
840             $result = true;
841             $error = ""          ;
842             if(isset($this->   _arguments))
```

```
843           {
844               if(!isset($this->   _arguments["Type"          ]))
845               {
846                   $result = false;
847                   $error .= ((strlen($error) >    0) ? ", "          : ""          ) .
"Type"          ;
848               }
849               /*if(!isset($this->_arguments["Hostname"]))
850               {
851                   $result = false;
852                   $error .= ((strlen($error) > 0) ? ", " : "") .
"Hostname";
853               }
854               if(!isset($this->_arguments["Username"]))
855               {
856                   $result = false;
857                   $error .= ((strlen($error) > 0) ? ", " : "") .
"Username";
858               }
859               if(!isset($this->_arguments["Password"]))
860               {
861                   $result = false;
862                   $error .= ((strlen($error) > 0) ? ", " : "") .
"Password";
863               }*/
864               if(!isset($this->   _arguments["Database"          ]))
865               {
866                   $result = false;
867                   $error .= ((strlen($error) >    0) ? ", "          : ""          ) .
"Database"          ;
868               }
869           }
870           else
871               $result = false;
872           if(!$result)
873               $this->   SetError("EnsureRequiredArguments"          , "The following required
arguments have not been set: "      . $error);
874           return $result;
875       }
876
877       /**
878       * Gets the Latest Error from the Driver
879       * @access private
880       * @internal Sets an error based on the driver error.
881       */
882       function GetDriverError()
883       {
884           $driverError = $this->   _driver->   GetLastError();
885           if(strlen($driverError) >    0)
886           {
887               $error = explode(": "          , $driverError);
888               switch(count($error))
889               {
890                   case 1:
891                       $this->   SetError("     {$this->   _arguments["Type"          ]}Driver"     ,
$error[0]);
892                       break;
893                   case 2:
894                       $this->   SetError($error[0], $error[1]);
895                       break;
896               }
897           }
898       }
899
900       /**
901       * Checks That a Parameter is a Resource
902       *
903       * Checks that a given parameter is a resource type variable.
904       * @access private
905       * @param resource $resource A variable to ensure is a resource.
906       * @param string $callee The name of the method calling this function. Used in the setting of
the error message.
907       * @return bool Returns true if the given parameter is a resource, otherwise false.
908       */
909       function IsResource($resource, $callee)
910       {
911           if(!($result = is_resource($resource)))
912               $this->   SetError($callee, "Resource specified is not a valid resource."          );
913           return $result;
914       }
```

```
915
916         /**
917          * Loads an AbstractDB Database Driver
918          *
919          * Loads the AbstractDB driver for the type of database being connected to.
920          * @access private
921          * @internal If the driver could not be loaded an error is set explaining the
922          *  reason for the failure.
923          * @return bool Returns true if the driver was successfully loaded, otherwise false.
924          */
925         function LoadDriver()
926         {
927             $dirname = dirname(__FILE__);
928             if(!defined("ABSTRACTDB_DRIVER_INCLUDED"          ))
929             {
930                 $driverfile = "     $dirname/abstractdb_driver.class.php"     ;
931                 if(!file_exists($driverfile))
932                 {
933                     $this->   SetError("LoadDriver"          , "     Could not load the AbstractDB
Driver base class. File $driverfile not found."     );
934                     return false;
935                 }
936                 include($driverfile);
937             }
938             if(!defined("ABSTRACTDB_"           . strtoupper($this->   _arguments["Type"          ]) .
"_INCLUDED"             ))
939             {
940                 $driverfile = "     $dirname/drivers/abstractdb_"      . strtolower($this-
>   _arguments["Type"          ]) . ".php"          ;
941                 if(!file_exists($driverfile))
942                 {
943                     $this->   SetError("LoadDriver"          , "     Driver file $driverfile could not
be found."     );
944                     return false;
945                 }
946                 include($driverfile);
947             }
948             $driver_class = "AbstractDB_"           . $this->   _arguments["Type"          ];
949             $this->   _driver = new $driver_class($this->   _arguments);
950             $this->   GetDriverError();
951             if(strlen($this->   _error) >    0)
952                 return false;
953             else
954                 $this->   Support = $this->   _driver->   Support;
955             return true;
956         }
957
958         /**
959          * Parses Connection and Driver Specific Arguments
960          * @access private
961          * @param array $arguments A list of connection and driver specific arguments.
962          * @return bool Returns true if arguments were successfully parsed, otherwise false.
963          */
964         function ParseArguments($arguments)
965         {
966             $result = false;
967             if(isset($arguments["ConnectionString"          ]))
968                 $this->   ParseConnectionArguments($arguments["ConnectionString"          ]);
969             else
970             {
971                 if(isset($arguments["Type"          ]))
972                     $this->   _arguments["Type"          ] = $arguments["Type"          ];
973                 if(isset($arguments["Username"          ]))
974                     $this->   _arguments["Username"          ] = $arguments["Username"          ];
975                 if(isset($arguments["Password"          ]))
976                     $this->   _arguments["Password"          ] = $arguments["Password"          ];
977                 if(isset($arguments["Hostname"          ]))
978                     $this->   _arguments["Hostname"          ] = $arguments["Hostname"          ];
979                 if(isset($arguments["Options"          ]))
980                 {
981                     if(isset($arguments["Options"          ]["Port"          ]))
982                         $this->   _arguments["Options"          ]["Port"          ] =
$arguments["Options"          ]["Port"          ];
983                     if(isset($arguments["Options"          ]["Persistent"          ]))
984                         $this->   _arguments["Options"          ]["Persistent"          ] =
$arguments["Options"          ]["Persistent"          ];
985                 }
986                 if(isset($arguments["Database"          ]))
987                     $this->   _arguments["Database"          ] = $arguments["Database"          ];
988             }
```

```
989             return $this->   EnsureRequiredArguments();
990         }
991
992         /**
993         * Parses ConnectionString Argument
994         *
995         * Parses the ConnectionString argument passed in via the constructor.
996         * @access private
997         * @param string $connectionString A connection string in the format
998         *
<b>Type</b>://<b>User</b>:<b>Pass</b>@<b>Host</b>:<b&gt
;Port</b>/<b>Database</b>?<b>Options</b>/option1=value1&<b>Options
</b>/option2=value2
999         * @return bool Returns true if the connection string was successfully parsed, otherwise false.
1000        * @todo There may be a use for the fragment part of the parsed connection string.
1001        */
1002        function ParseConnectionArguments($connectionString)
1003        {
1004            $result = true;
1005            $parsed = parse_url($connectionString);
1006            if(isset($parsed["scheme"          ]))
1007                $this->   _arguments["Type"          ] = urldecode($parsed["scheme"          ]);
1008            else
1009                $result = false;
1010            if(isset($parsed["host"          ]))
1011                $this->   _arguments["Hostname"          ] = urldecode($parsed["host"          ]);
1012            else
1013                $result = false;
1014            if(isset($parsed["port"          ]))
1015                $this->   _arguments["Options"          ]["Port"          ] =
urldecode($parsed["port"          ]);
1016            if(isset($parsed["user"          ]))
1017                $this->   _arguments["Username"          ] = urldecode($parsed["user"          ]);
1018            else
1019                $result = false;
1020            if(isset($parsed["pass"          ]))
1021                $this->   _arguments["Password"          ] = urldecode($parsed["pass"          ]);
1022            else
1023                $result = false;
1024            if(isset($parsed["path"          ]))
1025                $this->   _arguments["Database"          ] =
substr(urldecode($parsed["path"          ]), 1);
1026            else
1027                $result = false;
1028            if(isset($parsed["query"          ]))
1029            {
1030                $options=explode("&"              ,$parsed["query"          ]);
1031                for($option=0; $option <    count($options); $option++)
1032                {
1033                    if(gettype($equal = strpos($options[$option], "="          )) !=
"integer"          )
1034                    {
1035                        // This option does not have a value.
1036                        continue;
1037                    }
1038                    $argument = urldecode(substr($options[$option], 0, $equal));
1039                    $value = urldecode(substr($options[$option], $equal + 1));
1040                    if(gettype($slash = strpos($argument, "/"          )) == "integer"          )
1041                    {
1042                        if(substr($argument, 0, $slash) != "Options"          )
1043                        {
1044                            // Not a valid Options argument
1045                            continue;
1046                        }
1047                        $this->   _arguments["Options"          ][substr($argument, $slash + 1)] =
$value;
1048                    }
1049                    else
1050                        $this->   _arguments[$argument] = $value;
1051                }
1052            }
1053            if(isset($parsed["fragment"          ])){}
1054            return $result;
1055        }
1056
1057        /**
1058        * Sets the Latest Error Message.
1059        *
1060        * Sets the latest error message and calls an error handling function if one was set.
1061        * @access private
```

```
1062      * @param string $scope The scope of the error, generally the function in which it occured.
1063      * @param string $message The actual error message.
1064      */
1065     function SetError($scope, $message)
1066     {
1067         $this->   _error = "     $scope: $message"     ;
1068         if(strcmp($function = $this->   _error_handler, ""          ))
1069         {
1070             $error = array("Scope"           =>    $scope, "Message"           =>    $message);
1071             $function($this, $error);
1072         }
1073     }
1074
1075
1076 }
1077 ?>
```

File Source for abstractdb_driver.class.php

Documentation for this file is available at [abstractdb_driver.class.php]

```php
<?php
/**
* AbstractDB Driver Base Class Definition
*
* @package AbstractDB
* @author Pacific-Cybersoft
* @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
* @version v 1.0.2
* @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
*/

/**
* AbstractDB Driver included constant.
*
* Flag that indicates that the driver base class file has been included.
* @access public
*/
define("ABSTRACTDB_DRIVER_INCLUDED"          , true);

/**
* AbstractDB Driver Base Class
*
* Driver base class from which all other AbstractDB drivers extend.
* @package AbstractDB
* @abstract
*/
class AbstractDB_Driver
{
    /* PRIVATE FIELDS */
    /**
    * List of connection and driver specific arguments.
    * @access private
    * @var array
    */
    var $_arguments;
    /**
    * Database connection handle.
    * @access private
    * @var resource
    */
    var $_connection;
    /**
    * The last error message.
    * @access private
    * @var string
    */
    var $_error;

    /* PUBLIC PROPERTIES */
    /**
    * List of supported features of the currently loaded driver.
    * @access public
    * @var array
    */
    var $Support;

    /**
    * AbstractDB Driver Constructor
    *
    * Initilises an instance of the AbstractDB Driver base class.
    * @access public
    * @internal Stores the arguments parameter as a local field.
    * @param array $arguments A list of connection and driver specific arguments.
    *  See {@link AbstractDB} for details concerning connection arguments.
    */
    function AbstractDB_Driver($arguments)
    {
```

```
68              $this->   ClearError();
69              $this->   _arguments = $arguments;
70          }
71
72          /* PUBLIC FUNCTIONS */
73          /**
74          * Gets the Number of Affected Rows
75          *
76          * Gets the number of rows affected by the last query.
77          * @access public
78          * @internal This method must be overriden in extended classes with a call to parent if this
feature is supported.
79          *
80          *  If this feature is not supported an error should be set explaining this.
81          * @return int Returns the number of rows affected by the last executed query.
82          */
83          function AffectedRows()
84          {
85              $this->   ClearError();
86              return 0;
87          }
88
89          /**
90          * Closes the Database Connection
91          * @access public
92          * @internal This method must be overriden in extended classes with a call to parent if this
feature is supported.
93          *
94          *  If this feature is not supported this method should return a default value of true.
95          * @return bool Returns true if the database connection was successfully closed, otherwise false.
96          */
97          function Close()
98          {
99              $this->   ClearError();
100             return false;
101         }
102
103         /**
104         * Move a Result Pointer to the Specified Row
105         * @access public
106         * @internal This method must be overriden in extended classes with a call to parent if this
feature is supported.
107         *
108         *  If this feature is not supported an error should be set explaining this.
109         * @param resource $rs A reference to a result handle returned by executing a query.
110         * @param int $row_num The 0 based index of the row that the result pointer should move to.
111         * @return bool Returns true if the operation was successful, otherwise false.
112         */
113         function DataSeek(&    $rs, $row_num)
114         {
115             $this->   ClearError();
116             return false;
117         }
118
119         /**
120         * Fetches a Result Row as an Associative Array
121         * @access public
122         * @internal This method must be overriden in extended classes with a call to parent if this
feature is supported.
123         *
124         *  If this feature is not supported an error should be set explaining this.
125         * @param resource A reference to a resource handle returned by executing a query.
126         * @return array Returns an associative array if the operation was successful, otherwise false.
127         */
128         function FetchAssoc(&    $rs)
129         {
130             $this->   ClearError();
131             return false;
132         }
133
134         /**
135         * Fetches the First Field Value
136         *
137         * Fetches the value from the first field of a result row.
138         * @access public
139         * @internal This method must be overriden in extended classes with a call to parent.
140         * @param resource A reference to a resource handle returned by executing a query.
141         * @return mixed Returns the field value if the operation was successful, otherwise false.
142         */
143         function FetchField(&    $rs)
```

```
144         {
145             $this->   ClearError();
146             return false;
147         }
148 
149         /**
150         * Fetches a Result Row as an Object
151         * @access public
152         * @internal This method must be overriden in extended classes with a call to parent if this
feature is supported.
153         *
154         *  If this feature is not supported an error should be set explaining this.
155         * @param resource A reference to a resource handle returned by executing a query.
156         * @return object Returns an object if the operation was successful, otherwise false.
157         */
158         function FetchObject(&    $rs)
159         {
160             $this->   ClearError();
161             return false;
162         }
163 
164         /**
165         * Fetches a Result Row
166         * @access public
167         * @internal This method must be overriden in extended classes with a call to parent.
168         * @param resource A reference to a resource handle returned by executing a query.
169         * @return array Returns an array if the operation was successful, otherwise false.
170         */
171         function FetchRow(&    $rs)
172         {
173             $this->   ClearError();
174             return false;
175         }
176 
177         /**
178         * Gets the Number of Fields
179         *
180         * Gets the number of fields returned by given result handle.
181         * @access public
182         * @internal This method must be overriden in extended classes with a call to parent.
183         * @param resource $rs A reference to a resource handle returned by executing a query.
184         * @return int Returns the number of fields returned by the last executed query.
185         */
186         function FieldCount(&    $rs)
187         {
188             $this->   ClearError();
189             return 0;
190         }
191 
192         /**
193         * Gets the Field Names of a Query
194         * @access public
195         * @internal This method must be overriden in extended classes with a call to parent.
196         * @param resource $rs A reference to a resource handle returned by executing a query.
197         * @param array $fields A reference to an array that will contain the field names.
198         * @return bool Returns true if the operation was successful, otherwise false.
199         */
200         function FieldNames(&    $rs, &    $fields)
201         {
202             $this->   ClearError();
203             return false;
204         }
205 
206         /**
207         * Frees a Result Resource
208         *
209         * Frees the resources associated with the given result handle.
210         * @access public
211         * @internal This method must be overriden in extended classes with a call to parent if this
feature is supported.
212         *
213         *  If this feature is not supported the method should return a default value of true.
214         * @param resource A reference to a resource handle returned by executing a query.
215         * @return bool Returns true if the resource handle was successfully freed.
216         */
217         function FreeResult(&    $rs)
218         {
219             $this->   ClearError();
220             return true;
221         }
```

```
222
223         /**
224         * Gets the Last Error.
225         * @access public
226         * @return string The last error message.
227         */
228         function GetLastError()
229         {
230             return $this->   _error;
231         }
232
233         /**
234         * Gets the Last Inserted AUTO_INCREMENT ID
235         *
236         * Gets the ID of the last AUTO_INCREMENT record inserted into the databse.
237         * @access public
238         * @internal This method must be overriden in extended classes with a call to parent if this
feature is supported.
239         *
240         *  If this feature is not supported an error should be set explaining this.
241         * @return mixed Returns either the ID of the last inserted AUTO_INCREMENT record, or -1 if the
242         *  last query was not an insert.
243         */
244         function InsertID()
245         {
246             $this->   ClearError();
247             return -1;
248         }
249
250         /**
251         * Executes an SQL Statement.
252         *
253         * Executes an SQL statement passed in as a parameter.
254         * @access public
255         * @internal This method must be overriden in extended classes or it will cause
256         *  the script to exit.
257         *
258         *  If the query fails an error should be set that explains as best as possible the
259         *  reason for the query failure.
260         * @param string $sql The SQL statement to execute on the database.
261         * @return resource If the query was successful, the result handle of the query
262         *  used in result fetching functions, otherwise false.
263         */
264         function Query($sql)
265         {
266             die("Method AbstractDB_Driver::Query() must be redefined in extended classes without
calling parent."     );
267         }
268
269         /**
270         * Executes an SQL Replace Query
271         * @access public
272         * @internal This method must be overriden in extended classes with a call to parent if this
feature is supported.
273         *
274         *  If this feature is not supported an error should be set explaining this.
275         * @param string $table The name of the table to execute the replace query on.
276         * @param array $fields An associative array of field definitions. Keys should be the field
names and
277         *  values should be an associative array containing the following keys:
278         *
279         *  Key => bool indicating that this field is the primary key or part of a unique index. Key
values must not be NULL.
280         *
281         *  Type => either "text", "numeric", "bool".
282         *
283         *  Value => the value of the field.
284         *
285         *  Null => bool indicating if the value of the field should be set to NULL.
286         *
287         *  e.g. $fields = array("Field1" => array("Key" => true,
"Type" => "numeric", "Value" => 123, "Null" => false));
288         * @return resource If the replace query was successful, the result handle of the query,
otherwise false.
289         */
290         function Replace($table, $fields)
291         {
292             $this->   ClearError();
293             return false;
294         }
```

```
295
296         /**
297          * Gets the Number of Rows
298          *
299          * Gets the number of rows returned by given result handle.
300          * @access public
301          * @internal This method must be overriden in extended classes with a call to parent.
302          * @param resource $rs A reference to a resource handle returned by executing a query.
303          * @return int Returns the number of fields returned by the last executed query.
304          */
305         function RowCount(&    $rs)
306         {
307             $this->   ClearError();
308             return 0;
309         }
310
311         /**
312          * Sets the Current Active Database
313          * @access public
314          * @internal This method must be overriden in extended classes with a call to parent.
315          *
316          *  If this operation fails an error should be set explaining as best as possible the reason for
    the failure.
317          * @param string $dbName The name of the database to set active.
318          * @return mixed The name of the previously active database, or false if an error occured.
319          */
320         function SetDatabase($dbName)
321         {
322             $this->   ClearError();
323             return false;
324         }
325
326         /* PRIVATE FUNCTIONS */
327         /**
328          * Clears the latest error message.
329          * @access private
330          */
331         function ClearError()
332         {
333             $this->   _error = ""          ;
334         }
335
336         /**
337          * Opens a Database Connection
338          *
339          * Attempts to connect to the database using the parameters given in the constructor.
340          * @access private
341          * @internal This method must be overriden in extended classes or it will cause
342          *  the script to exit.
343          *
344          *  The _connection field should be set to hold the resource link.
345          *
346          *  If the connection fails an error should be set that explains as best as possible the
347          *  reason for the connection failure.
348          * @return bool Returns true if the connection was successfully made, otherwise false.
349          */
350         function Connect()
351         {
352             die("Method AbstractDB_Driver::Connect() must be redefined in extended classes without
    calling parent."     );
353         }
354
355         /**
356          * Sets the Error Message
357          * @access private
358          * @param string $scope The scope of the error, generally the function in which it occured.
359          * @param string $message The actual error message.
360          */
361         function SetError($scope, $message)
362         {
363             $this->   _error = "     $scope: $message"     ;
364         }
365     }
366     ?>
```

File Source for abstractdb_mysql.php

Documentation for this file is available at abstractdb_mysql.php

```php
<?php
/**
* AbstractDB MySQL Driver
*
* @package AbstractDB
* @author Pacific-Cybersoft
* @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
* @version v 1.0.2
* @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
*/

/**
* MySQL Driver included constant.
*
* Flag that indicates that the MySQL driver has been included.
* @access public
*/
define("ABSTRACTDB_MYSQL_INCLUDED"          , true);

/**
* AbstractDB MySQL Driver Class
* @access public
* @package AbstractDB
*/
class AbstractDB_MySQL extends AbstractDB_Driver
{
    /**
    * AbstractDB MySQL Driver Constructor
    *
    * Initilises an instance of the AbstractDB MySQL Driver class.
    * @internal Calls the {@link AbstractDB_Driver} constructor passing the given parameters and
sets
    *  the values of the Support list.
    */
    function AbstractDB_MySQL($arguments)
    {
        $this->   AbstractDB_Driver($arguments);
        $this->   Support["AffectedRows"          ] =
            $this->   Support["InsertID"          ] =
            $this->   Support["FetchObject"            ] =
            $this->   Support["FetchAssoc"            ] =
            $this->   Support["DataSeek"          ] = true;

    }

    /* PUBLIC FUNCTIONS */
    function AffectedRows()
    {
        parent::AffectedRows();
        return @mysql_affected_rows($this->   _connection);
    }

    function Close()
    {
        if($this->   _connection)
        {
            $result = @mysql_close($this->   _connection);
            if(!$result)
            {
                $error = mysql_error($this->   _connection);
                if(strlen($error) == 0 && isset(        $php_errormsg))
                    $error = $php_errormsg;
                else
                    $error = "The connection could not be closed, reason unknown."          ;
                $this->   SetError("Close"          , $error);
            }
            return $result;
```

```
67              }
68              else
69              {
70                  $this->   SetError("Close"          , "No connection to close."          );
71                  return false;
72              }
73          }
74
75          function DataSeek(&    $rs, $row_num)
76          {
77              parent::DataSeek($rs, $row_num);
78              return @mysql_data_seek($rs, $row_num);
79          }
80
81          function FetchAssoc(&    $rs)
82          {
83              parent::FetchAssoc($rs);
84              return @mysql_fetch_assoc($rs);
85          }
86
87          function FetchField(&    $rs)
88          {
89              parent::FetchField($rs);
90              $result = false;
91              if($row = $this->   FetchRow($rs))
92                  $result = $row[0];
93              return $result;
94          }
95
96          function FetchObject(&    $rs)
97          {
98              parent::FetchObject($rs);
99              return @mysql_fetch_object($rs);
100         }
101
102         function FetchRow(&    $rs)
103         {
104             parent::FetchRow($rs);
105             return @mysql_fetch_row($rs);
106         }
107
108         function FieldCount(&    $rs)
109         {
110             parent::FieldCount($rs);
111             return @mysql_num_fields($rs);
112         }
113
114         function FieldNames(&    $rs, &    $fields)
115         {
116             parent::FieldNames($rs, $fields);
117             for($index = 0; $index <    $this->   FieldCount($rs); $index++)
118             {
119                 $fields[] = @mysql_field_name($rs, $index);
120                 if($error = mysql_error($this->   _connection))
121                 {
122                     $this->   SetError("FieldNames"          , $error);
123                     return false;
124                 }
125             }
126             return true;
127         }
128
129         function FreeResult(&    $rs)
130         {
131             $result = @mysql_free_result($rs);
132             if(!$result)
133             {
134                 $error = mysql_error($this->   _connection);
135                 if(strlen($error) == 0 && isset(        $php_errormsg))
136                     $error = $php_errormsg;
137                 else
138                     $error = "Could not free resource handle, reason unknown."          ;
139                 $this->   SetError("FreeResult"          , $error);
140             }
141             return $result;
142         }
143
144         function InsertID()
145         {
146             parent::InsertID();
```

```
147             $result = @mysql_insert_id($this->   _connection);
148             if($result == 0)
149                 return -1;
150             else
151                 return $result;
152         }
153 
154         function Query($sql)
155         {
156             if(!$this->   Connect())
157                 return false;
158             if(!(@mysql_select_db($this->   _arguments["Database"          ], $this->   _connection)
&&          $rs = @mysql_query($sql, $this->   _connection)))
159             {
160                 $this->   SetError("Query"          , @mysql_error($this->   _connection));
161                 return false;
162             }
163             return $rs;
164         }
165 
166         function Replace($table, $fields)
167         {
168             parent::Replace($table, $fields);
169             for($keys = 0, $query = $values = ""          , reset($fields), $field = 0; $field <
count($fields); next($fields), $field++)
170             {
171                 $fieldname = key($fields);
172                 if($field >    0)
173                 {
174                     $query .= ", "          ;
175                     $values .= ", "          ;
176                 }
177                 $query .= $fieldname;
178                 if(isset($fields[$fieldname]["Null"          ]) &&
$fields[$fieldname]["Null"          ])
179                     $value = "NULL"          ;
180                 else
181                 {
182                     if(!isset($fields[$fieldname]["Value"          ]))
183                     {
184                         $this->   SetError("Replace"          , "     Value was not set for field
$fieldname."     );
185                         return false;
186                     }
187                     switch(isset($fields[$fieldname]["Type"          ]) ?
$fields[$fieldname]["Type"          ] : "text"          )
188                     {
189                         case "text"          :
190                             $value = "'"          . str_replace("'"          , "\'"          ,
$fields[$fieldname]["Value"          ]) . "'"          ;
191                             break;
192                         case "boolean"          :
193                             $value = ($fields[$fieldname]["Value"          ]) ? "1"          :
"0"          ;
194                             break;
195                         case "numeric"          :
196                             $value = strval($fields[$fieldname]["Value"          ]);
197                             break;
198                         default:
199                             $this->   SetError("Replace"          , "     Type was not specified for
field $fieldname."     );
200                             return false;
201                     }
202                 }
203                 $values .= $value;
204                 if(isset($fields[$fieldname]["Key"          ]) &&
$fields[$fieldname]["Key"          ])
205                 {
206                     if($value == "NULL"          )
207                     {
208                         $this->   SetError("Replace"          , "Key values can not be
NULL."     );
209                         return false;
210                     }
211                     $keys++;
212                 }
213             }
214             if($keys == 0)
215             {
216                 $this->   SetError("Replace"          , "No Key fields were specified."          );
```

```
217             return false;
218         }
219         return $this->   Query("     REPLACE INTO $table ($query) VALUES($values)"     );
220     }
221
222     function RowCount(&    $rs)
223     {
224         parent::RowCount($rs);
225         return @mysql_num_rows($rs);
226     }
227
228     function SetDatabase($dbName)
229     {
230         if(!$this->   Connect())
231             return false;
232         if(!($result = @mysql_select_db($dbName, $this->   _connection)))
233             $this->   SetError("SetDatabase"          , mysql_error($this->   _connection));
234         else
235         {
236             $result = $this->   _arguments["Database"          ];
237             $this->   _arguments["Database"          ] = $dbName;
238         }
239         return $result;
240     }
241
242     /* PRIVATE FUNCTIONS */
243     function Connect()
244     {
245         if($this->   _connection)
246             return true;
247         if(isset($this->   _arguments["Options"          ]["Persistent"          ]) &&
248             $this->   _arguments["Options"          ]["Persistent"          ] === true &&
249             function_exists("mysql_pconnect"          ))
250             $connect = "mysql_pconnect"          ;
251         else
252             $connect = "mysql_connect"          ;
253         $port = isset($this->   _arguments["Options"          ]["Port"          ]) ? ":"
. $this->   _arguments["Options"          ]["Port"          ] : ""          ;
254
255         $args = array();
256         if(isset($this->   _arguments["Hostname"          ]))
257         {
258             $server = $this->   _arguments["Hostname"          ] . $port;
259             $args[] = $server;
260             if(isset($this->   _arguments["Username"          ]))
261             {
262                 $args[] = $this->   _arguments["Username"          ];
263                 if(isset($this->   _arguments["Password"          ]))
264                     $args[] = $this->   _arguments["Password"          ];
265             }
266         }
267         switch(count($args))
268         {
269             case 1:
270                 $this->   _connection = @$connect($args[0]);
271                 break;
272             case 2:
273                 $this->   _connection = @$connect($args[0], $args[1]);
274                 break;
275             case 3:
276                 $this->   _connection = @$connect($args[0], $args[1], $args[2]);
277                 break;
278         }
279         if(!$this->   _connection)
280         {
281             $this->   SetError("Connect"          , isset($php_errormsg) ? $php_errormsg :
"Could not connect to MySQL server"          );
282             return false;
283         }
284         return true;
285     }
286 }
287 ?>
```

File Source for affected_rows.php

Documentation for this file is available at affected_rows.php

```php
<?php
/**
 * GetAffectedRows Example
 *
 * This example uses the GetAffectedRows method to get the number of rows affected by the last query.
 * @package AbstractDB
 * @author Pacific-Cybersoft
 * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
 * @version v 1.0.2
 * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
 */

/**
 * Include the AbstractDB main class.
 */
include("../abstractdb.class.php"          );

// Instatiate AbstractDB
$db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
// Execute a Query
if($rs = $db->   Query("SELECT * FROM tablename"          ))
{
    // Get the number of affected rows
    $affected = $db->   GetAffectedRows();
    // Free Result
    $db->   FreeResult($rs);
}
else
{
    die($db->   GetLastError());
}
// Always good practice to close AbstractDB at the end of each script.
$db->   Close();
?>
```

File Source for fetch_next_result.php

Documentation for this file is available at fetch_next_result.php

```
1     <?php
2     /**
3     * FetchNextResult Example
4     *
5     * This example uses the FetchNextResult[Assoc, Object, Row] functions to retrieve query results.
6     * The method shown in this example uses FetchNextResultRow but the other functions would be used
7     * in the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename"          ))
24    {
25        // Loop through results
26        while($db->   FetchNextResultRow($rs, $row))
27        {
28            // Do something with the data in $row.
29            print_r($row);
30        }
31        // Free Result
32        $db->   FreeResult($rs);
33    }
34    else
35    {
36        die($db->   GetLastError());
37    }
38    // Always good practice to close AbstractDB at the end of each script.
39    $db->   Close();
40    ?>
```

File Source for fetch_result.php

Documentation for this file is available at fetch_result.php

```
1     <?php
2     /**
3     * FetchResult Example
4     *
5     * This example uses the FetchResult[Assoc, Object, Row] functions to retrieve the first query
result.
6     * The method shown in this example uses FetchResultRow but the other functions would be used in the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename LIMIT 0,1"          ))
24    {
25        // Get the result
26        $db->   FetchResultRow($rs, $row);
27        // Do something with the data in $row.
28        print_r($row);
29    }
30    else
31    {
32        die($db->   GetLastError());
33    }
34    // Always good practice to close AbstractDB at the end of each script.
35    $db->   Close();
36    ?>
```

File Source for fetch_result_all.php

Documentation for this file is available at fetch_result_all.php

```php
<?php
/**
* FetchResult All Example
*
* This example uses the FetchResult[Assoc, Object, Row]All functions to retrieve the all query results.
* The method shown in this example uses FetchResultRowAll but the other functions would be used in the
* same way.
* @package AbstractDB
* @author Pacific-Cybersoft
* @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
* @version v 1.0.2
* @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
*/

/**
* Include the AbstractDB main class.
*/
include("../abstractdb.class.php"          );

// Instatiate AbstractDB
$db = new AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quot;));
// Execute a Query
if($rs = $db->   Query("SELECT * FROM tablename"          ))
{
    // Get the result
    $db->   FetchResultRowAll($rs, $all);
    // Do something with the data in $all.
    foreach($all as $row)
    {
        print_r($row);
    }
}
else
{
    die($db->   GetLastError());
}
// Always good practice to close AbstractDB at the end of each script.
$db->   Close();
?>
```

File Source for fetch_result_column.php

Documentation for this file is available at fetch_result_column.php

```
1     <?php
2     /**
3     * FetchResultColumn Example
4     *
5     * This example uses the FetchResultColumn function to execute a query and retrieve the values in the
6     * first column of the result set.
7     * @package AbstractDB
8     * @author Pacific-Cybersoft
9     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
10    * @version v 1.0.2
11    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
12    */
13
14    /**
15    * Include the AbstractDB main class.
16    */
17    include("../abstractdb.class.php"          );
18
19    // Instatiate AbstractDB
20    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
21    // Execute a Query
22    if($rs = $db->   Query("SELECT * FROM tablename"          ))
23    {
24        // Get the result
25        $db->   FetchResultColumn($rs, $column);
26        // Do something with the data in $column.
27        print_r($column);
28    }
29    else
30    {
31        die($db->   GetLastError());
32    }
33    // Always good practice to close AbstractDB at the end of each script.
34    $db->   Close();
35    ?>
```

File Source for fetch_result_field.php

Documentation for this file is available at fetch_result_field.php

```
1     <?php
2     /**
3     * FetchResultField Example
4     *
5     * This example uses the FetchResultField function to execute a query and retrieve the value of the
6     * first field of the first row of the result set.
7     * @package AbstractDB
8     * @author Pacific-Cybersoft
9     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
10    * @version v 1.0.2
11    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
12    */
13
14    /**
15    * Include the AbstractDB main class.
16    */
17    include("../abstractdb.class.php"          );
18
19    // Instatiate AbstractDB
20    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
21    // Execute a Query
22    if($rs = $db->   Query("SELECT * FROM tablename LIMIT 0,1"          ))
23    {
24        // Get the result
25        $db->   FetchResultField($rs, $field);
26        // Do something with the data in $field.
27        print_r($field);
28    }
29    else
30    {
31        die($db->   GetLastError());
32    }
33    // Always good practice to close AbstractDB at the end of each script.
34    $db->   Close();
35    ?>
```

File Source for fetch_seek_result.php

Documentation for this file is available at fetch_seek_result.php

```
1     <?php
2     /**
3     * FetchSeekResult Example
4     *
5     * This example uses the FetchSeekResult[Assoc, Object, Row] functions to retrieve a query result.
6     * The method shown in this example uses FetchSeekResultRow but the other functions would be used in
7     * the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   Query("SELECT * FROM tablename"          ))
24    {
25        // Get the result
26        if($db->   FetchSeekResultRow($rs, 3, $row))
27            // Do something with the data in $row.
28            print_r($row);
29        }
30
31        // Free Result
32        $db->   FreeResult($rs);
33    }
34    else
35    {
36        die($db->   GetLastError());
37    }
38    // Always good practice to close AbstractDB at the end of each script.
39    $db->   Close();
40    ?>
```

File Source for field_count.php

Documentation for this file is available at field_count.php

```php
<?php
/**
 * GetFieldCount Example
 *
 * This example uses the GetFieldCount method to get the number of fields in a result set.
 * @package AbstractDB
 * @author Pacific-Cybersoft
 * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
 * @version v 1.0.2
 * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
 */

/**
 * Include the AbstractDB main class.
 */
include("../abstractdb.class.php"          );

// Instatiate AbstractDB
$db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
// Execute a Query
if($rs = $db->   Query("SELECT * FROM tablename"          ))
{
    // Get the number of affected rows
    $fieldcount = $db->   GetFieldCount($rs);
    // Free Result
    $db->   FreeResult($rs);
}
else
{
    die($db->   GetLastError());
}
// Always good practice to close AbstractDB at the end of each script.
$db->   Close();
?>
```

File Source for field_names.php

Documentation for this file is available at field_names.php

```
1     <?php
2     /**
3     * GetFieldNames Example
4     *
5     * This example uses the GetFieldNames method to get the names of fields in a result set.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20    // Execute a Query
21    if($rs = $db->   Query("SELECT * FROM tablename"          ))
22    {
23        // Get the field names
24        $db->   GetFieldNames($rs, $fields);
25        // Do something with the field names
26        foreach($fields as $field)
27        {
28            echo($field)
29        }
30        // Free Result
31        $db->   FreeResult($rs);
32    }
33    else
34    {
35        die($db->   GetLastError());
36    }
37    // Always good practice to close AbstractDB at the end of each script.
38    $db->   Close();
39    ?>
```

File Source for insert_id.php

Documentation for this file is available at insert_id.php

```
1     <?php
2     /**
3     * GetInsertID Example
4     *
5     * This example uses the GetInsertID method to get the ID of the last inserted AUTO_INCREMENT record.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20    // Execute a Query
21    if($db->   Query("INSERT tablename (field1, field2) VALUES ('field1', 'field2')"          ))
22    {
23        // Get the insert ID
24        $inserID = $db->   GetInsertID();
25    }
26    else
27    {
28        die($db->   GetLastError());
29    }
30    // Always good practice to close AbstractDB at the end of each script.
31    $db->   Close();
32    ?>
```

File Source for instantiate_associative.php

Documentation for this file is available at instantiate_associative.php

```php
<?php
/**
* Instantiation Example
*
* This example uses an associative array of arguments to instantiate AbstractDB.
* @package AbstractDB
* @author Pacific-Cybersoft
* @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
* @version v 1.0.2
* @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
*/

/**
* Include the AbstractDB main class.
*/
include("../abstractdb.class.php"          );

/**
* @global AbstractDB An instance of the {@link AbstractDB} class.
*/
$db = new AbstractDB(array(
    "Type"          =>   "mysql"          ,
    "Username"          =>   "username"          ,
    "Password"          =>   "password"          ,
    "Hostname"          =>   "localhost"          ,
    "Database"          =>   "cybermatrix"          ,
    "Options"          =>array(
        "Persistent"          =>   true,
        "Port"          =>   3306)
    ));

// Always good practice to close AbstractDB at the end of each script.
$db->   Close();
?>
```

File Source for instantiate_connectionstring.php

Documentation for this file is available at instantiate_connectionstring.php

```
1     <?php
2     /**
3     * Instantiation Example
4     *
5     * This example uses a ConnectionString argument to instantiate AbstractDB.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    /**
19    * @global AbstractDB An instance of the {@link AbstractDB} class.
20    */
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost:3306/mydatabas
e?Options/Persistent=1"     ));
22
23    // Always good practice to close AbstractDB at the end of each script.
24    $db->   Close();
25    ?>
```

File Source for query.php

Documentation for this file is available at query.php

```php
<?php
/**
 * Query Example
 *
 * This example shows how to query a database using AbstractDB.
 * @package AbstractDB
 * @author Pacific-Cybersoft
 * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
 * @version v 1.0.2
 * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
 */

/**
 * Include the AbstractDB main class.
 */
include("../abstractdb.class.php"          );

// Instatiate AbstractDB
$db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
// Execute a Query
if($rs = $db->   Query("SELECT * FROM tablename"          ))
{
    // Use $rs to retrieve result rows.
    $db->   FetchNextResultRow($rs, $row);
    // Free Result
    $db->   FreeResult($rs);
}
else
{
    die($db->   GetLastError());
}
// Always good practice to close AbstractDB at the end of each script.
$db->   Close();
?>
```

File Source for query_column.php

Documentation for this file is available at query_column.php

```
1     <?php
2     /**
3     * QueryColumn Example
4     *
5     * This example uses the QueryColumn function to query a database and retrieve the values in the
6     * first column of the result set.
7     * @package AbstractDB
8     * @author Pacific-Cybersoft
9     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
10    * @version v 1.0.2
11    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
12    */
13
14    /**
15    * Include the AbstractDB main class.
16    */
17    include("../abstractdb.class.php"          );
18
19    // Instatiate AbstractDB
20    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
21    // Execute a Query
22    if($db->   QueryColumn("SELECT * FROM tablename"          , $column))
23    {
24        // Do something with the data in $column
25        foreach($column as $key=>   $value)
26            echo("     Value at index $k is $v\r\n"     );
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

File Source for query_field.php

Documentation for this file is available at query_field.php

```
1     <?php
2     /**
3     * QueryField Example
4     *
5     * This example uses the QueryField function to query a database and retrieve the value of the
6     * first column of the first row of a result set.
7     * @package AbstractDB
8     * @author Pacific-Cybersoft
9     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
10    * @version v 1.0.2
11    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
12    */
13
14    /**
15    * Include the AbstractDB main class.
16    */
17    include("../abstractdb.class.php"          );
18
19    // Instatiate AbstractDB
20    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
21    // Execute a Query
22    if($db->   QueryField("SELECT * FROM tablename LIMIT 0,1"          , $field))
23    {
24        // Do something with $field
25        echo($field);
26    }
27    else
28    {
29        die($db->   GetLastError());
30    }
31    // Always good practice to close AbstractDB at the end of each script.
32    $db->   Close();
33    ?>
```

File Source for query_result.php

Documentation for this file is available at query_result.php

```
1     <?php
2     /**
3     * Query Example
4     *
5     * This example uses the Query[Assoc, Objects, Row] functions to query a database and retrieve the
6     * first result. The method shown in this example uses QueryRow but the other functions would
7     * be used in the same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($db->   QueryRow("SELECT * FROM tablename LIMIT 0,1"          , $row))
24    {
25        // Do something with the data in $row
26        $field0 = $row[0];
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

File Source for query_result_all.php

Documentation for this file is available at query_result_all.php

```
1     <?php
2     /**
3     * Query All Example
4     *
5     * This example uses the Query[Assoc, Object, Row]All functions to retrieve the all query results.
6     * The method shown in this example uses QueryRowAll but the other functions would be used in the
7     * same way.
8     * @package AbstractDB
9     * @author Pacific-Cybersoft
10    * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
11    * @version v 1.0.2
12    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
13    */
14
15    /**
16    * Include the AbstractDB main class.
17    */
18    include("../abstractdb.class.php"          );
19
20    // Instatiate AbstractDB
21    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
22    // Execute a Query
23    if($rs = $db->   QueryRowAll("SELECT * FROM tablename"          , $all))
24    {
25        // Do something with the data in $all.
26        foreach($all as $row)
27        {
28            print_r($row);
29        }
30    }
31    else
32    {
33        die($db->   GetLastError());
34    }
35    // Always good practice to close AbstractDB at the end of each script.
36    $db->   Close();
37    ?>
```

File Source for replace.php

Documentation for this file is available at replace.php

```
1     <?php
2     /**
3     * Replace Example
4     *
5     * This example show how to use the Replace method.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20
21    $fields = array(
22        "PrimaryKeyField"          =>array(   "Type"          =>   "numeric"          ,
"Key"          =>   true, "Value"          =>   1, "Null"          =>   false),
23        "Field2"          =>array(   "Type"          =>   "text"          , "Key"          =>   false,
"Value"          =>   "replaced value"          , "Null"          =>   false),
24        "Field3"          =>array(   "Type"          =>   "text"          , "Key"          =>   false,
"Value"          =>   ""          , "Null"          =>   true));
25
26    // Execute a REPLACE Query
27    if(!$db->   Replace("tablename"          , $fields))
28        die($db->   GetLastError());
29    // Always good practice to close AbstractDB at the end of each script.
30    $db->   Close();
31    ?>
```

File Source for row_count.php

Documentation for this file is available at row_count.php

```
1     <?php
2     /**
3     * GetRowCount Example
4     *
5     * This example uses the GetRowCount method to get the number of rows in a result set.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    // Instatiate AbstractDB
19    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
20    // Execute a Query
21    if($rs = $db->   Query("SELECT * FROM tablename"          ))
22    {
23        // Get the number of rows
24        $rowcount = $db->   GetRowCount($rs);
25        // Free Result
26        $db->   FreeResult($rs);
27    }
28    else
29    {
30        die($db->   GetLastError());
31    }
32    // Always good practice to close AbstractDB at the end of each script.
33    $db->   Close();
34    ?>
```

File Source for set_error_handler.php

Documentation for this file is available at set_error_handler.php

```
1     <?php
2     /**
3     * SetErrorHandler Example
4     *
5     * This example show how to set an error handler for AbstractDB.
6     * @package AbstractDB
7     * @author Pacific-Cybersoft
8     * @copyright (C) 2005 Pacific-Cybersoft. All Rights Reserved.
9     * @version v 1.0.2
10    * @license http://www.gnu.org/copyleft/lesser.txt GNU Lesser General Public License
11    */
12
13    /**
14    * Include the AbstractDB main class.
15    */
16    include("../abstractdb.class.php"          );
17
18    function ExampleErrorHandler($source, $error)
19    {
20        // Do something with the info.
21        // $source will be a copy of the AbstractDB object.
22        print_r($source);
23        // $error will be an associative array containing error scope and message.
24        echo($error["Scope"          ] . " - "           . $error["Message"          ]);
25    }
26
27    // Instatiate AbstractDB
28    $db = new
AbstractDB(array("ConnectionString"          =>   "mysql://username:password@localhost/mydatabase&quo
t;));
29    // Set the error handler
30    $db->   SetErrorHandler("ExampleErrorHandler"          ) or die($db->   GetLastError());
31    // Now if we execute a bad query or whenever an error occurs, ExampleErrorHandler will be called.
32    $db->   Query("SELECT NonExistentColumn from BadTableName"          );
33    // Always good practice to close AbstractDB at the end of each script.
34    $db->   Close();
35    ?>
```

Appendix D - Todo List

In Package AbstractDB

In **AbstractDB::ParseConnectionArguments()**

- There may be a use for the fragment part of the parsed connection string.

Index

A

C

E

F

I

L

Q

R

S