

## Installation

The SQLite3 extension is enabled by default as of PHP 5.3.0. It's possible to disable it by using **--without-sqlite3** at compile time.

Windows users must enable `php_sqlite3.dll` in order to use this extension. This DLL is included with Windows distributions of PHP as of PHP 5.3.0.

For detailed installation instructions, kindly check our PHP tutorial and its official website.

## PHP Interface APIs

Following are important PHP routines which can suffice your requirement to work with SQLite database from your PHP program. If you are looking for a more sophisticated application, then you can look into PHP official documentation.

S.N.	API & Description
1	<b>public void SQLite3::open</b> <i>filename, flags, encryption<sub>key</sub></i>  Opens an SQLite 3 Database. If the build includes encryption, then it will attempt to use the key.  If the <i>filename</i> is given as <b>':memory:'</b> , SQLite3::open will create an in-memory database in RAM that lasts only for the duration of the session.  If filename is actual device file name, SQLite3::open attempts to open the database file by using its value. If no file by that name exists then a new database file by that name gets created.  Optional flags used to determine how to open the SQLite database. By default, open uses <code>SQLITE3_OPEN_READWRITE   SQLITE3_OPEN_CREATE</code> .  2 <b>public bool SQLite3::exec</b> <i>string\$query</i>  This routine provides a quick, easy way to execute SQL commands provided by <code>sql</code> argument which can consist of more than one SQL command. This routine is used to execute a result-less query against a given database.  3 <b>public SQLite3Result SQLite3::query</b> <i>string\$query</i>  This routine executes an SQL query, returning an <b>SQLite3Result</b> object if the query returns results.  4 <b>public int SQLite3::lastErrorCode</b> <i>void</i>  This routine returns the numeric result code of the most recent failed SQLite request  5 <b>public string SQLite3::lastErrorMsg</b> <i>void</i>  This routine returns english text describing the most recent failed SQLite request.  6 <b>public int SQLite3::changes</b> <i>void</i>  This routine returns the number of database rows that were updated or inserted or deleted by the most recent SQL statement

## 7 **public bool SQLite3::close** void

This routine closes a database connection previously opened by a call to SQLite3::open.

## 8 **public string SQLite3::escapeString** string\$value

This routine returns a string that has been properly escaped for safe inclusion in an SQL statement.

# Connecting To Database

Following PHP code shows how to connect to an existing database. If database does not exist, then it will be created and finally a database object will be returned.

```
<?php
class MyDB extends SQLite3
{
    function __construct()
    {
        $this->open('test.db');
    }
}
$db = new MyDB();
if(!$db){
    echo $db->lastErrorMsg();
} else {
    echo "Opened database successfully\n";
}
?>
```

Now, let's run above program to create our database **test.db** in the current directory. You can change your path as per your requirement. If database is successfully created, then it will give the following message:

```
Open database successfully
```

# Create a Table

Following PHP program will be used to create a table in previously created database:

```
<?php
class MyDB extends SQLite3
{
    function __construct()
    {
        $this->open('test.db');
    }
}
$db = new MyDB();
if(!$db){
    echo $db->lastErrorMsg();
} else {
    echo "Opened database successfully\n";
}

$sql = <<<EOF
CREATE TABLE COMPANY
(ID INT PRIMARY KEY     NOT NULL,
NAME           TEXT      NOT NULL,
AGE            INT       NOT NULL,
ADDRESS        CHAR(50),
SALARY         REAL);
```

```

EOF;

$ret = $db->exec($sql);
if(!$ret){
    echo $db->lastErrorMsg();
} else {
    echo "Table created successfully\n";
}
$db->close();
?>

```

When above program is executed, it will create COMPANY table in your **test.db** and it will display the following messages:

```

Opened database successfully
Table created successfully

```

## INSERT Operation

Following PHP program shows how we can create records in our COMPANY table created in above example:

```

<?php
class MyDB extends SQLite3
{
    function __construct()
    {
        $this->open('test.db');
    }
}
$db = new MyDB();
if(!$db){
    echo $db->lastErrorMsg();
} else {
    echo "Opened database successfully\n";
}

$sql = <<<EOF
INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Paul', 32, 'California', 20000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Allen', 25, 'Texas', 15000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'Teddy', 23, 'Norway', 20000.00 );

INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 );
EOF;

$ret = $db->exec($sql);
if(!$ret){
    echo $db->lastErrorMsg();
} else {
    echo "Records created successfully\n";
}
$db->close();
?>

```

When above program is executed, it will create given records in COMPANY table and will display following two lines:

```

Opened database successfully
Records created successfully

```

## SELECT Operation

Following PHP program shows how we can fetch and display records from our COMPANY table created in above example:

```
<?php
class MyDB extends SQLite3
{
    function __construct()
    {
        $this->open('test.db');
    }
}
$db = new MyDB();
if(!$db){
    echo $db->lastErrorMsg();
} else {
    echo "Opened database successfully\n";
}

$sql =<<<EOF
SELECT * from COMPANY;
EOF;

$ret = $db->query($sql);
while($row = $ret->fetchArray(SQLITE3_ASSOC) ){
    echo "ID = ". $row['ID'] . "\n";
    echo "NAME = ". $row['NAME'] . "\n";
    echo "ADDRESS = ". $row['ADDRESS'] . "\n";
    echo "SALARY = ".$row['SALARY'] . "\n\n";
}
echo "Operation done successfully\n";
$db->close();
?>
```

When above program is executed, it will produce the following result:

```
Opened database successfully
ID = 1
NAME = Paul
ADDRESS = California
SALARY = 20000

ID = 2
NAME = Allen
ADDRESS = Texas
SALARY = 15000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY = 20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY = 65000

Operation done successfully
```

## UPDATE Operation

Following PHP code shows how we can use UPDATE statement to update any record and then fetch and display updated records from our COMPANY table:

```
<?php
class MyDB extends SQLite3
```

```

{
    function __construct()
    {
        $this->open('test.db');
    }
}
$db = new MyDB();
if(!$db){
    echo $db->lastErrorMsg();
} else {
    echo "Opened database successfully\n";
}
$sql =<<<EOF
    UPDATE COMPANY set SALARY = 25000.00 where ID=1;
EOF;
$ret = $db->exec($sql);
if(!$ret){
    echo $db->lastErrorMsg();
} else {
    echo $db->changes(), " Record updated successfully\n";
}

$sql =<<<EOF
    SELECT * from COMPANY;
EOF;
$ret = $db->query($sql);
while($row = $ret->fetchArray(SQLITE3_ASSOC) ){
    echo "ID = ". $row['ID'] . "\n";
    echo "NAME = ". $row['NAME'] . "\n";
    echo "ADDRESS = ". $row['ADDRESS'] . "\n";
    echo "SALARY = ". $row['SALARY'] . "\n\n";
}
echo "Operation done successfully\n";
$db->close();
?>

```

When above program is executed, it will produce the following result:

```

Opened database successfully
1 Record updated successfully
ID = 1
NAME = Paul
ADDRESS = California
SALARY = 25000

ID = 2
NAME = Allen
ADDRESS = Texas
SALARY = 15000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY = 20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY = 65000

Operation done successfully

```

## DELETE Operation

Following PHP code shows how we can use DELETE statement to delete any record and then fetch and display remaining records from our COMPANY table:

```
<?php
```

```

class MyDB extends SQLite3
{
    function __construct()
    {
        $this->open('test.db');
    }
}
$db = new MyDB();
if(!$db){
    echo $db->lastErrorMsg();
} else {
    echo "Opened database successfully\n";
}
$sql =<<<EOF
DELETE from COMPANY where ID=2;
EOF;
$ret = $db->exec($sql);
if(!$ret){
    echo $db->lastErrorMsg();
} else {
    echo $db->changes(), " Record deleted successfully\n";
}

$sql =<<<EOF
SELECT * from COMPANY;
EOF;
$ret = $db->query($sql);
while($row = $ret->fetchArray(SQLITE3_ASSOC) ){
    echo "ID = ". $row['ID'] . "\n";
    echo "NAME = ". $row['NAME'] . "\n";
    echo "ADDRESS = ". $row['ADDRESS'] . "\n";
    echo "SALARY = ". $row['SALARY'] . "\n\n";
}
echo "Operation done successfully\n";
$db->close();
?>

```

When above program is executed, it will produce the following result:

```

Opened database successfully
1 Record deleted successfully
ID = 1
NAME = Paul
ADDRESS = California
SALARY = 25000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY = 20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY = 65000

Operation done successfully

```

Loading [MathJax]/jax/output/HTML-CSS/jax.js