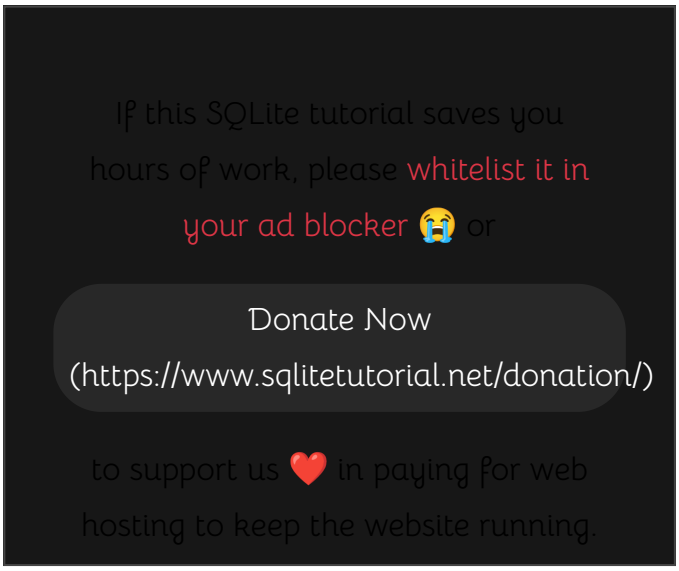
SQLite PHP: Update Data



Summary: in this tutorial, we will show you how to update data in the SQLite database using PHP PDO.

Steps for updating data in the SQLite database from PHP

The process of updating data is similar to the process of inserting data (https://www.sqlitetutorial.net/sqlite-php/insert/) . To update data in a table, you use these steps:

1. Connect to the SQLite database (https://www.sqlitetutorial.net/sqlite-php/connect/) by creating a new PDO object.
2. Prepare an UPDATE (https://www.sqlitetutorial.net/sqlite-update/) statement using the `prepare()` method of the PDO object. The `prepare()` method returns a `PDOStatement` object.
3. Bind values to the parameters of the `UPDATE` statement using the `bindValue()` method of the `PDOStatement` object.
4. Execute the `UPDATE` statement by calling the `execute()` method of the `PDOStatement` object. The `execute()` method returns true on success or false on failure.

SQLite PHP: update data example

We will use the `tasks` table that we created in the creating table tutorial (https://www.sqlitetutorial.net/sqlite-php/update/) for the demonstration.

See the following `SQLiteUpdate` class.

```
<?php

namespace App;

/**
 * PHP SQLite Update Demo
 */
class SQLiteUpdate {

    /**
     * PDO object
     * @var \PDO
     */
    private $pdo;

    /**
     * Initialize the object with a specified PDO object
     */
    public function __construct($pdo) {
        $this->pdo = $pdo;
    }

    /**
     * Mark a task specified by the task_id completed
     * @param type $taskId
     * @param type $completedDate
     * @return bool true if success and falase on failure
     */
    public function completeTask($taskId, $completedDate) {
        // SQL statement to update status of a task to completed
        $sql = "UPDATE tasks "
                . "SET completed = 1, "
```

```
                . "completed_date = :completed_date "
                . "WHERE task_id = :task_id";

        $stmt = $this->pdo->prepare($sql);

        // passing values to the parameters
        $stmt->bindValue(':task_id', $taskId);
        $stmt->bindValue(':completed_date', $completedDate);

        // execute the update statement
        return $stmt->execute();
    }
}
```

In the `completeTask()` method, we update the `completed` and `completed_date` columns of the `tasks` table using the `UPDATE` statement.

Suppose you want to mark the task with id 2 completed, you use the following code in the `index.php` file.

```
<?php

require 'vendor/autoload.php';

use App\SQLiteConnection;
use App\SQLiteUpdate;

$pdo = (new SQLiteConnection())->connect();
$sqlite = new SQLiteUpdate($pdo);
```

```
// mark task #2 as completed
$taskId = 2;
$result = $sqlite->completeTask($taskId, '2016-05-02');

if ($result)
    echo 'Task #$taskId has been completed';
else
    echo 'Whoops, something wrong happened.';
```

Execute the `index.php` script, we got the following message:

```
Task #2 has been completed
```

Let's verify the update using the following SELECT (https://www.sqlitetutorial.net/sqlite-select/) statement:

```
SELECT *
  FROM tasks
 WHERE task_id = 2;
```

The task with id 2 has been updated as shown in the screenshot above.

To get the number of rows affected by the `UPDATE` statement, you use `rowCount()` method of the `PDOStatement` object.

```
<?php
   $stmt->rowCount();
```

In this tutorial, we have shown you how to update data in the SQLite table using PHP PDO.