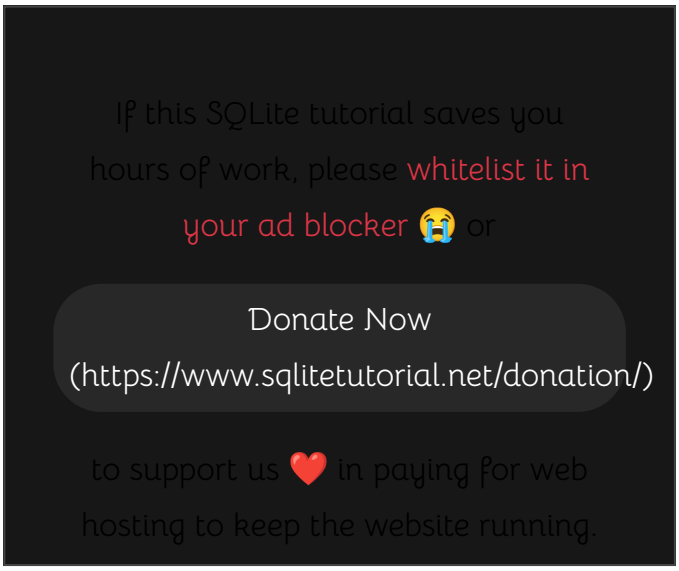
SQLite PHP: Creating Tables



Summary: in this tutorial, we will show you how to create new tables in the SQLite database using PHP PDO.

We will create two new tables in the phpsqlite database that we created in the previous tutorial (https://www.sqlitetutorial.net/sqlite-php/connect/) . The following illustrates the SQL script that creates the `projects` and `tasks` tables.

```
CREATE TABLE IF NOT EXISTS projects (
    project_id   INTEGER PRIMARY KEY,
    project_name TEXT    NOT NULL
);

CREATE TABLE IF NOT EXISTS tasks (
    task_id        INTEGER PRIMARY KEY,
    task_name      TEXT    NOT NULL,
    completed      INTEGER NOT NULL,
    start_date     TEXT,
    completed_date TEXT,
    project_id     INTEGER NOT NULL,
    FOREIGN KEY (
        project_id
    )
```

```
    REFERENCES projects (project_id) ON UPDATE CASCADE
                                     ON DELETE CASCADE
);
```

To create a new table in an SQLite database using PDO, you use the following steps:

1. First, connect to the SQLite database (https://www.sqlitetutorial.net/sqlite-php/connect/) by creating an instance of the PDO class.
2. Second, execute the CREATE TABLE (https://www.sqlitetutorial.net/sqlite-create-table/) statement by calling the `exec()` method of the PDO object.

We will reuse the `SQLiteConnection` class that we developed in the previous tutorial. The following `SQLiteCreateTable` class demonstrates how to create new tables in the `phpsqlite` database.

```
<?php

namespace App;

/**
 * SQLite Create Table Demo
 */
class SQLiteCreateTable {

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

    /**
     * connect to the SQLite database
     */
    public function __construct($pdo) {
        $this->pdo = $pdo;
    }
```

```
    /**
     * create tables
     */
    public function createTables() {
        $commands = ['CREATE TABLE IF NOT EXISTS projects (
                        project_id   INTEGER PRIMARY KEY,
                        project_name TEXT NOT NULL
                      )',
            'CREATE TABLE IF NOT EXISTS tasks (
                    task_id INTEGER PRIMARY KEY,
                    task_name  VARCHAR (255) NOT NULL,
                    completed  INTEGER NOT NULL,
                    start_date TEXT,
                    completed_date TEXT,
                    project_id VARCHAR (255),
                    FOREIGN KEY (project_id)
                    REFERENCES projects(project_id) ON UPDATE CASCADE
                                                    ON DELETE CASCADE)'];
        // execute the sql commands to create new tables
        foreach ($commands as $command) {
            $this->pdo->exec($command);
        }
    }

    /**
     * get the table list in the database
     */
    public function getTableList() {

        $stmt = $this->pdo->query("SELECT name
                                   FROM sqlite_master
                                   WHERE type = 'table'
                                   ORDER BY name");
        $tables = [];
        while ($row = $stmt->fetch(\PDO::FETCH_ASSOC)) {
            $tables[] = $row['name'];
        }
```

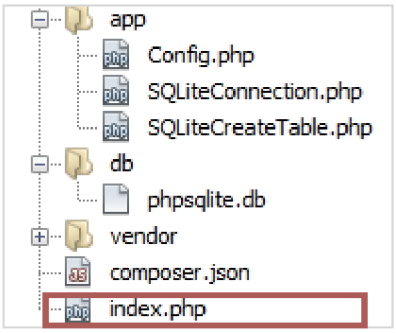
```
        return $tables;
    }

}
```

How it works.

The `createTables()` method is used to create tables in the `phpsqlite` database. First, we have an array that stores the `CREATE TABLE` statements. Then we loop through the array and execute each `CREATE TABLE` statement one by one using the `exec()` method of the PDO object.

The `getTableList()` method selects all the tables in an SQLite database by querying table name in the `sqlite_master` table. The predicate in the WHERE (https://www.sqlitetutorial.net/sqlite-where/) clause ensures that the query returns only the tables, not the views. You will learn how to query data in using PDO in the subsequent tutorial.

Now it's time to use the classes that we have developed.



In the index.php file, you use the following code:

```
<?php
require 'vendor/autoload.php';

use App\SQLiteConnection as SQLiteConnection;
use App\SQLiteCreateTable as SQLiteCreateTable;

$sqlite = new SQLiteCreateTable((new SQLiteConnection())->connect());
// create new tables
$sqlite->createTables();
```

```
// get the table list
$tables = $sqlite->getTableList();
?>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scal
        <meta name="description" content="">
        <meta name="author" content="sqlitetutorial.net">
        <title>PHP SQLite CREATE TABLE Demo</title>
        <link href="http://v4-alpha.getbootstrap.com/dist/css/bootstrap

    </head>
    <body>
        <div class="container">
            <div class="page-header">
                <h1>PHP SQLite CREATE TABLE Demo</h1>
            </div>

            <table class="table table-bordered">
                <thead>
                    <tr>
                        <th>Tables</th>
                    </tr>
                </thead>
                <tbody>
                    <?php foreach ($tables as $table) : ?>
                        <tr>

                            <td><?php echo $table ?></td>
                        </tr>
                    <?php endforeach; ?>
                </tbody>
            </table>
        </div>
```

```
    </body>
</html>
```

First, we create a new instance of the SQLiteCreateTable class and pass the PDO object which is created by using the SQLiteConnection class.

Second, we call the createTables to create the new tables and the getTableList method to query the newly created tables.

Third, in the HTML code, we display the table list.

The following illustrates the result of the index.php script:

In this tutorial, we have shown you how to create new tables by executing the `CREATE TABLE` statement using PHP PDO.