HOME START HERE BASIC ADVANCED FUNCTIONS INTERFACES TIPS TRYIT

Calling MySQL Stored Procedures from Node.js

?

Summary: in this tutorial, you will learn how to call a stored procedure in MySQL from a Node.js application.

The steps for calling a stored procedure are similar to the steps for executing a query as follows:

1. Connect to the MySQL database server.
2. Call the stored procedure by executing the `CALL spName` statement. The `spName` is the name of the stored procedure.
3. Close the database connection.

Calling a MySQL stored procedure example

For the demonstration, we create a new stored procedure `filterTodo` to query rows from the `todos` table based on the value of the `completed` field.

```
DELIMITER $$

CREATE PROCEDURE `filterTodo`(IN done BOOLEAN)
BEGIN
    SELECT * FROM todos WHERE completed = done;
```

the done argument is true, it returns all completed todos, otherwise, it returns the non completed todos.

As you know, to call a stored procedure in MySQL, you use the `CALL` statement. For example, to call the `filterTodo` stored procedure, you execute the following statement:

```
CALL filterTodo(true);
```

The statement returns the following result set:

```
+----+---------------------------+-----------+
| id | title                     | completed |
+----+---------------------------+-----------+
|  4 | It should work perfectly  |         1 |
+----+---------------------------+-----------+
1 row in set (0.00 sec)

Query OK, 0 rows affected (0.01 sec)
```

The following storedproc.js program calls the `filterTodo` stored procedure and returns the result set:

```
let mysql = require('mysql');
let config = require('./config.js');

let connection = mysql.createConnection(config);

let sql = `CALL filterTodo(?)`;

connection.query(sql, true, (error, results, fields) => {
  if (error) {
    return console.error(error.message);
  }
  console.log(results[0]);
});

connection.end();
```

Notice that the program uses the `config.js` module that stores the database's information:

```
module.exports = config;
```

In the `CALL` statement, we used a placeholder (?) to pass data to the stored procedure.

When we called the `query()` method on the `connection` object, we pass the value of the `done` argument as the second argument of the `query()` method.

Let’s run the storedproc.js program.

```
> node storedproc.js
[ RowDataPacket { id: 4, title: 'It should work perfectly', completed: 1 } ]
```

The program displayed 1 row as expected.

In this tutorial, you have learned how to call a stored procedure in MySQL from a Node.js program.

Was this tutorial helpful? 👍 Yes 👎 No

« Previous Tutorial:
Deleting Data in MySQL from Node.js

MYSQL GETTING STARTED

- What Is MySQL?
- Install MySQL Database Server
- Connect to MySQL Server
- Download MySQL Sample Database
- Load Sample Database

MYSQL NODE.JS TUTORIAL

- MySQL Node.js: Connecting to M
Database

MySQL Node.js: Deleting Data

MySQL Node.js: Calling Stored Procedures

MYSQL PROGRAMMING INTERFACES

PHP MySQL Tutorial

Node.js MySQL Tutorial

Python MySQL Tutorial

Perl MySQL Tutorial

MySQL JDBC Tutorial

OTHERS

MySQL Cheat Sheet

MySQL Resources

RECENT MYSQL TUTORIALS

MySQL Stored Object Access Control

Install MySQL on Ubuntu

Install MySQL CentOS

Getting Started with MySQL

Connect to MySQL Server

How To Lock User Accounts in MySQL

How To Unlock User Accounts in MySQL Server

Restart MySQL Server

MySQL database. We regularly publish useful MySQL tutorials to help web developers and database administrators learn MySQL faster and more effectively.

All MySQL tutorials are practical and easy-to-follow, with SQL script and screenshots available. More About Us

SITE LINKS

About Us

Contact Us

Request a Tutorial

Privacy Policy