Create Simple PHP 8 CRUD REST API with MySQL & PHP PDO

Last updated on by Digamber

This is a step by step PHP 8 & MySQL REST API tutorial, In this tutorial i am going to share with you how to create a PHP 8 CRUD (Create, Read, Update, Delete) RESTful API with MySQL database.

If you aspire to hold a fundamental understanding of PHP frameworks, then you must check out our previous tutorial where we described How to Create a PHP Laravel 6 CRUD Web App with MySQL.

This tutorial is going to cover how to build simple CRUD REST API in PHP and MySQL, Test PHP RESTful API with Postmen and Setup PHP development environment on your local development system from scratch.

Table of Contents

What is API (Application Programming Interface)?

API means Application Programming Interface, and It is a set of routines, protocols, and tools for creating software applications. An API interface makes communication possible between various software components.

In software development, API is a URL that handles the data for the web application through HTTP Requests GET, POST, UPDATE & DELETE, and manages the CRUD operations.

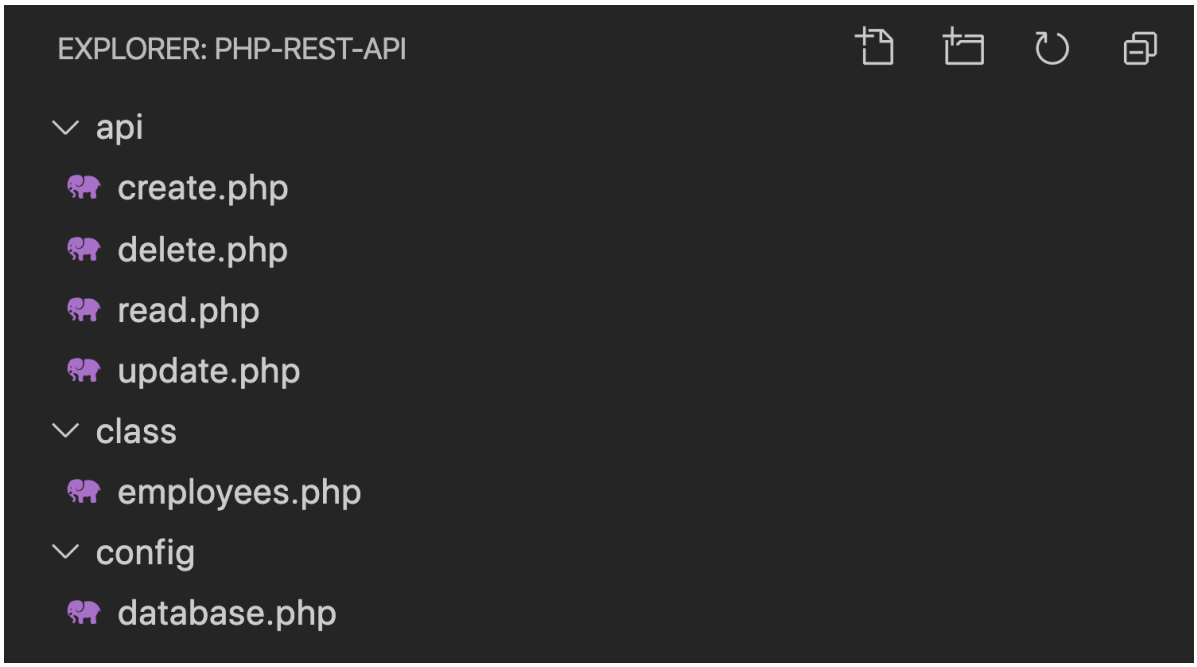
What is REST API?

> Representational state transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, called RESTful Web services, provide interoperability between computer systems on the Internet. RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. Other kinds of Web services, such as

SOAP Web services, expose their own arbitrary sets of operations.
source: wikipedia

PHP 8 API Project File Structure

This is going to be the file structure of our PHP 8 REST API project, we created **api**, **class** and **config** folders to store the API and MySQL database related configurations files.



You can run your PHP project through MAMP or WAMP however we are using the command line tool to start the PHP project.

Navigate to the directory where you have located your PHP project.

```
cd php-project-name
```

Bash

Once you are in the project folder then run the following command to start the PHP app.

```
php -S 127.0.0.1:8080
```

Bash

Following output will be displayed on your terminal screen.

```
PHP 7.3.11 Development Server started at Thu May  7 13:21:35 2020
Listening on http://127.0.0.1:8080
Document root is /Users/digamber/Desktop/php-rest-api
```

Bash

Check your project on the following url: http://127.0.0.1:8080

MySQL Database Configuration

Before we get started we need to create a `phpapidb` database, well this can be done through two methods either you can use the **PhpMyAdmin**, or you can do it by accessing the MySQL root user via the terminal or command-line tool to create the Database.

We have written a detailed article on Installing and Setting up MySQL in Mac's Terminal app.

We are accessing the root user via command-line to create the database.

```
mysql -u root -p
```

Bash

Create a new MySQL database.

```
CREATE DATABASE phpapidb;
```

Bash

Check the database.

```
SHOW DATABASES;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| phpapidb           |
| sys                |
+--------------------+
```

Bash

Use the database.

```
use phpapidb;
```

Bash

Once you are done with the database creation, then you should run the SQL script to create the table.

```
CREATE TABLE IF NOT EXISTS `Employee` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(256) NOT NULL,
  `email` varchar(50),
  `age` int(11) NOT NULL,
  `designation` varchar(255) NOT NULL,
  `created` datetime NOT NULL,
  PRIMARY KEY (`id`)
)ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=19;
```

Bash

Make sure what table you have created.

```
SHOW tables;
```

```
+--------------------+
| Tables_in_phpapidb |
+--------------------+
| Employee           |
+--------------------+
```

Bash

Populate data into the `Employee` table.

```
INSERT INTO `Employee` (`id`, `name`, `email`, `age`, `designation`, `created`)
VALUES
(1, 'John Doe', 'johndoe@gmail.com', 32, 'Data Scientist', '2012-06-01
02:12:30'),
(2, 'David Costa', 'sam.mraz1996@yahoo.com', 29, 'Apparel Patternmaker', '2013-
03-03 01:20:10'),
(3, 'Todd Martell', 'liliane_hirt@gmail.com', 36, 'Accountant', '2014-09-20
03:10:25'),
(4, 'Adela Marion', 'michael2004@yahoo.com', 42, 'Shipping Manager', '2015-04-11
04:11:12'),
(5, 'Matthew Popp', 'krystel_wol7@gmail.com', 48, 'Chief Sustainability
Officer', '2016-01-04 05:20:30'),
(6, 'Alan Wallin', 'neva_gutman10@hotmail.com', 37, 'Chemical Technician',
'2017-01-10 06:40:10'),
(7, 'Joyce Hinze', 'davonte.maye@yahoo.com', 44, 'Transportation Planner',
'2017-05-02 02:20:30'),
(8, 'Donna Andrews', 'joesph.quitz@yahoo.com', 49, 'Wind Energy Engineer',
'2018-01-04 05:15:35'),
(9, 'Andrew Best', 'jeramie_roh@hotmail.com', 51, 'Geneticist', '2019-01-02
02:20:30'),
(10, 'Joel Ogle', 'summer_shanah@hotmail.com', 45, 'Space Sciences Teacher',
'2020-02-01 06:22:50');
```

Bash

If you are using PHP with MySQL 8.0+ you might get this error: **The server requested authentication method unknown to the client**

MySQL 8 uses **auth_socket** as a default plugin, so your applications require you to log in to your database using a password. You have to log in as root to mysql by using the following SQL script.

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'password';
```

Bash

1. You need to replace `'password'` with your root password.
2. If your app is not logged in to your database with the root user, then replace the `'root'` user in the above command with the user that your application is using.

Make Database Connection

The following code holds the MySQL database details such as db name, user name and password. It makes the database connection with PHP using the PDO.

> The PHP Data Objects (PDO) extension specifies a lightweight, consistent interface for accessing databases in PHP. Every database driver that implements the PDO interface can expose database-specific features as regular extension functions. Note that you cannot perform any database functions using the PDO extension by itself; you must use

a database-specific PDO driver to access a database server.
source – php.net

Create the `config` folder inside the PHP API project, also create the **database.php** file and place the following code.

```
<?php
    class Database {
        private $host = "127.0.0.1";
        private $database_name = "phpapidb";
        private $username = "root";
        private $password = "xxxxxxxx";

        public $conn;

        public function getConnection(){
            $this->conn = null;
            try{
                $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" .
$this->database_name, $this->username, $this->password);
                $this->conn->exec("set names utf8");
            }catch(PDOException $exception){
                echo "Database could not be connected: " . $exception-
>getMessage();
            }
            return $this->conn;
        }
    }
?>
```

PHP

Create PHP Class

Now, we will create the PHP Class by the name of Employee. The PHP Classes are used to execute Object-Oriented Programming in PHP. PHP Classes encapsulate the values of the table for the Database. We will define the values that are stored in the variable form in the SQL table.

The SQL table properties are associated with the Connection class that is added via the constructor class.

Each object class contains the CRUD method that is being operated via the PHP functions to perform the CREATE, READ, UPDATE & DELETE operations for the defined table rows.

Create **class/employees.php** file and define the CRUD methods inside the Employee class.

```
<?php
    class Employee{

        // Connection
        private $conn;

        // Table
        private $db_table = "Employee";

        // Columns
        public $id;
        public $name;
        public $email;
        public $age;
```

```
        public $designation;
        public $created;

        // Db connection
        public function __construct($db){
            $this->conn = $db;
        }

        // GET ALL
        public function getEmployees(){
            $sqlQuery = "SELECT id, name, email, age, designation, created FROM
" . $this->db_table . "";
            $stmt = $this->conn->prepare($sqlQuery);
            $stmt->execute();
            return $stmt;
        }

        // CREATE
        public function createEmployee(){
            $sqlQuery = "INSERT INTO
                        ". $this->db_table ."
                    SET
                        name = :name,
                        email = :email,
                        age = :age,
                        designation = :designation,
                        created = :created";

            $stmt = $this->conn->prepare($sqlQuery);

            // sanitize
            $this->name=htmlspecialchars(strip_tags($this->name));
            $this->email=htmlspecialchars(strip_tags($this->email));
            $this->age=htmlspecialchars(strip_tags($this->age));
            $this->designation=htmlspecialchars(strip_tags($this->designation));
            $this->created=htmlspecialchars(strip_tags($this->created));

            // bind data
            $stmt->bindParam(":name", $this->name);
            $stmt->bindParam(":email", $this->email);
            $stmt->bindParam(":age", $this->age);
            $stmt->bindParam(":designation", $this->designation);
            $stmt->bindParam(":created", $this->created);

            if($stmt->execute()){
               return true;
            }
            return false;
        }

        // READ single
        public function getSingleEmployee(){
            $sqlQuery = "SELECT
                        id,
                        name,
                        email,
                        age,
                        designation,
                        created
                      FROM
                        ". $this->db_table ."
                    WHERE
                       id = ?
                    LIMIT 0,1";
```

```
    $stmt = $this->conn->prepare($sqlQuery);

    $stmt->bindParam(1, $this->id);

    $stmt->execute();

    $dataRow = $stmt->fetch(PDO::FETCH_ASSOC);

    $this->name = $dataRow['name'];
    $this->email = $dataRow['email'];
    $this->age = $dataRow['age'];
    $this->designation = $dataRow['designation'];
    $this->created = $dataRow['created'];
}

// UPDATE
public function updateEmployee(){
    $sqlQuery = "UPDATE
                ". $this->db_table ."
            SET
                name = :name,
                email = :email,
                age = :age,
                designation = :designation,
                created = :created
            WHERE
                id = :id";

    $stmt = $this->conn->prepare($sqlQuery);

    $this->name=htmlspecialchars(strip_tags($this->name));
    $this->email=htmlspecialchars(strip_tags($this->email));
    $this->age=htmlspecialchars(strip_tags($this->age));
    $this->designation=htmlspecialchars(strip_tags($this->designation));
    $this->created=htmlspecialchars(strip_tags($this->created));
    $this->id=htmlspecialchars(strip_tags($this->id));

    // bind data
    $stmt->bindParam(":name", $this->name);
    $stmt->bindParam(":email", $this->email);
    $stmt->bindParam(":age", $this->age);
    $stmt->bindParam(":designation", $this->designation);
    $stmt->bindParam(":created", $this->created);
    $stmt->bindParam(":id", $this->id);

    if($stmt->execute()){
       return true;
    }
    return false;
}

// DELETE
function deleteEmployee(){
    $sqlQuery = "DELETE FROM " . $this->db_table . " WHERE id = ?";
    $stmt = $this->conn->prepare($sqlQuery);

    $this->id=htmlspecialchars(strip_tags($this->id));

    $stmt->bindParam(1, $this->id);

    if($stmt->execute()){
        return true;
    }
```

```
            return false;
        }

    }
?>
```

PHP

The Employee class manages the CRUD operation

- `__construct()` — Makes the database connection ready.
- `getEmployees()` — Get all records.
- `getSingleEmployee()` — Get single records.
- `createEmployee()` — Create record.
- `updateEmployee()` — Update record.
- `deleteEmployee()` — Fetch single record.

Fetch MySQL Table Records using PHP REST API Endpoint

The following code retrieves all the records from MySQL table. So create **read.php** file in the `api` folder and place the following code.

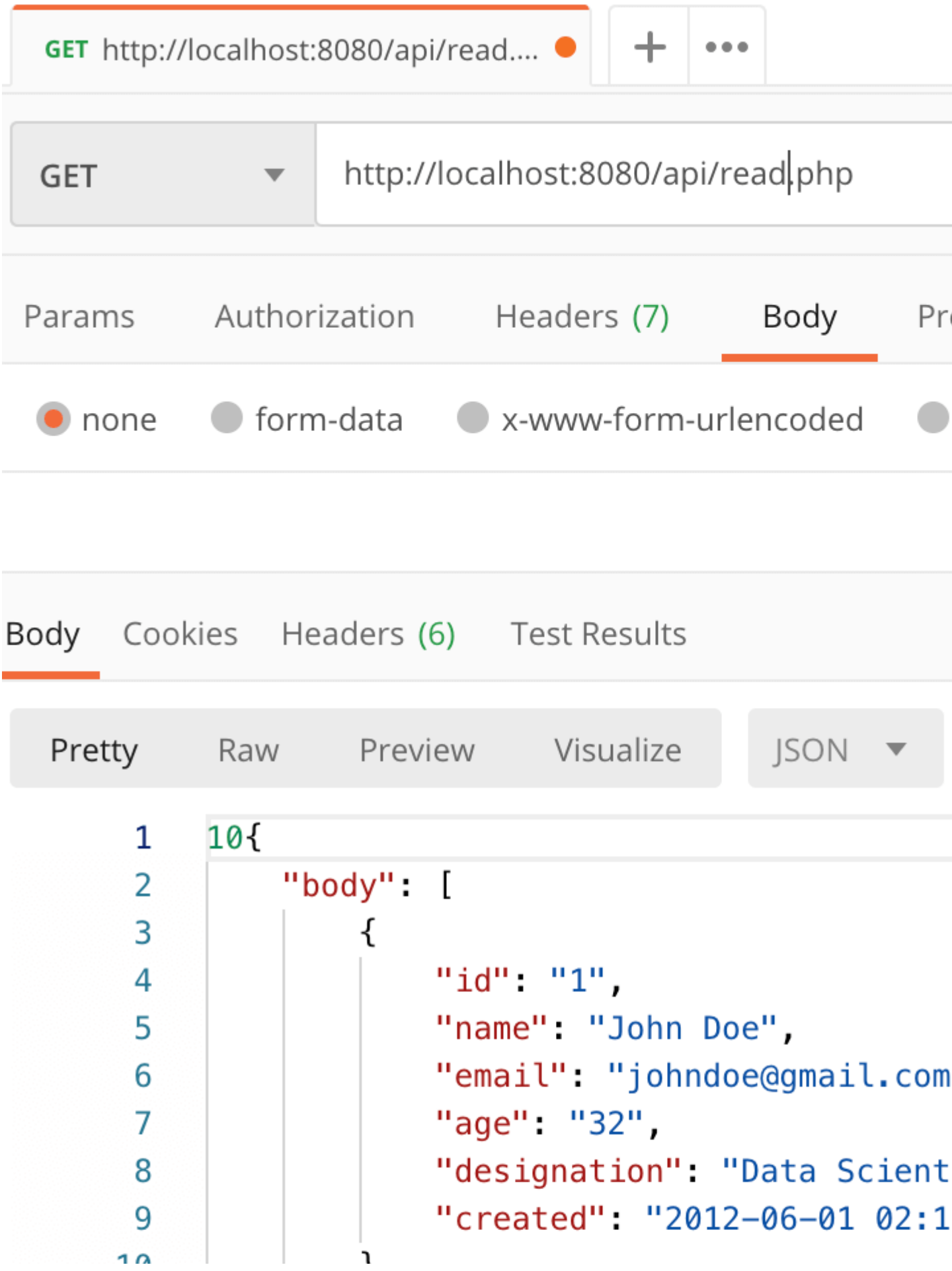
```
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");

    include_once '../config/database.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $items = new Employee($db);

    $stmt = $items->getEmployees();
    $itemCount = $stmt->rowCount();


    echo json_encode($itemCount);

    if($itemCount > 0){

        $employeeArr = array();
        $employeeArr["body"] = array();
        $employeeArr["itemCount"] = $itemCount;

        while ($row = $stmt->fetch(PDO::FETCH_ASSOC)){
            extract($row);
            $e = array(
                "id" => $id,
                "name" => $name,
                "email" => $email,
                "age" => $age,
                "designation" => $designation,
                "created" => $created
            );

            array_push($employeeArr["body"], $e);
        }
        echo json_encode($employeeArr);
```

```
    }

    else{
        http_response_code(404);
        echo json_encode(
            array("message" => "No record found.")
        );
    }
?>
```

PHP

Let us test the PHP API Endpoint using Postman, open the Postman and use the following URL and click on the Send button to check the output.

Method	Endpoint
GET	http://localhost:8080/api/read.php


GET http://localhost:8080/api/read....
GET
http://localhost:8080/api/read.php
Params
Authorization
Headers (7)
Body
none
form-data
x-www-form-urlencoded
Body
Cookies
Headers (6)
Test Results
Pretty
Raw
Preview
Visualize
JSON
10{
"body": [
{
"id": "1",
"name": "John Doe",
"email": "johndoe@gmail.com
"age": "32",
"designation": "Data Scient
"created": "2012-06-01 02:1

Get Single Row from MySQL Database via PHP API

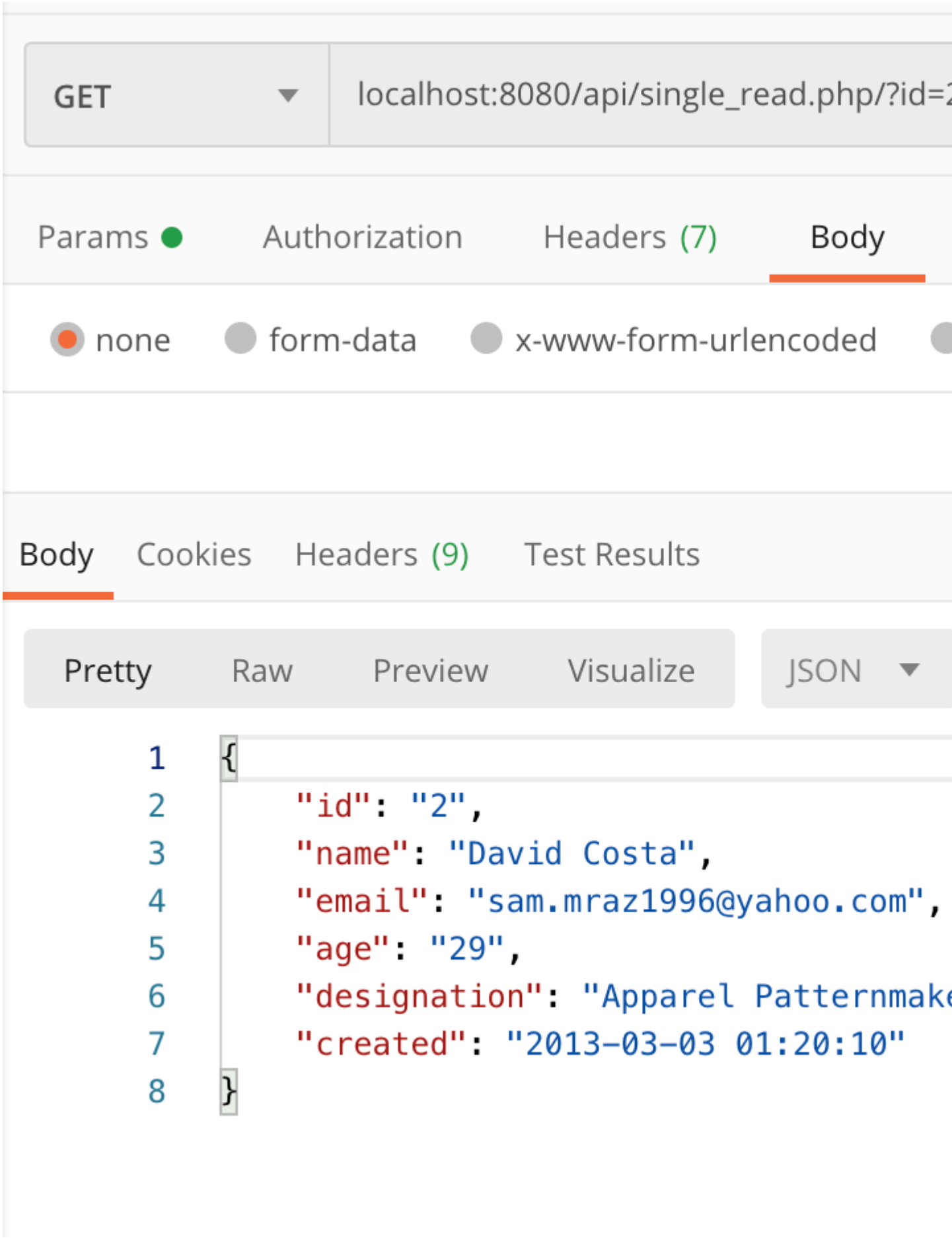
The following code fetch single column from MySQL database's table. Create **single_read.php** file in the `api` folder and place the following code.

The SELECT statement is being used here to get the table's column, in the following code, we are selecting the Employee values from the MySQL table.

```
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Methods: POST");
    header("Access-Control-Max-Age: 3600");
    header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers, Authorization, X-Requested-With");

    include_once '../config/database.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $item = new Employee($db);

    $item->id = isset($_GET['id']) ? $_GET['id'] : die();

    $item->getSingleEmployee();

    if($item->name != null){
        // create array
        $emp_arr = array(
            "id" =>  $item->id,
            "name" => $item->name,
            "email" => $item->email,
            "age" => $item->age,
            "designation" => $item->designation,
            "created" => $item->created
        );

        http_response_code(200);
        echo json_encode($emp_arr);
    }

    else{
        http_response_code(404);
        echo json_encode("Employee not found.");
    }
?>
```

PHP

Method	Endpoint
GET	localhost:8080/api/single_read.php/?id=2


GET
localhost:8080/api/single_read.php/?id=
Params
Authorization
Headers (7)
Body
none
form-data
x-www-form-urlencoded
Body
Cookies
Headers (9)
Test Results
Pretty
Raw
Preview
Visualize
JSON
{
"id": "2",
"name": "David Costa",
"email": "sam.mraz1996@yahoo.com",
"age": "29",
"designation": "Apparel Patternmake
"created": "2013-03-03 01:20:10"
}

Insert or Add Single Record in MySQL Table

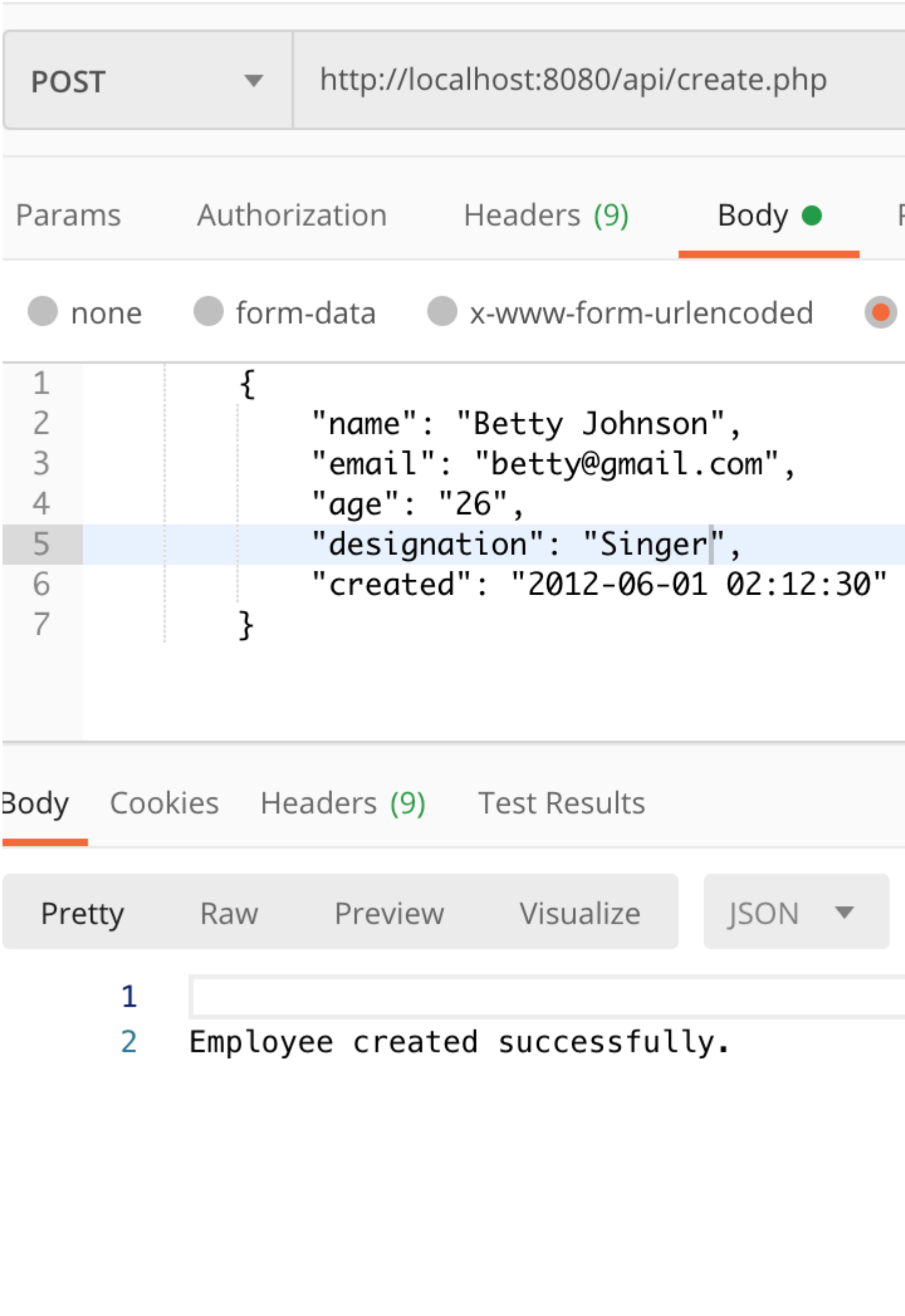
In this step we will create PHP REST API Endpoints to insert or add a single record in MySQL table.

Create **create.php** file in the `api` folder and add the following code.

```
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Methods: POST");
    header("Access-Control-Max-Age: 3600");
    header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers, Authorization, X-Requested-With");

    include_once '../config/database.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $item = new Employee($db);

    $data = json_decode(file_get_contents("php://input"));

    $item->name = $data->name;
    $item->email = $data->email;
    $item->age = $data->age;
    $item->designation = $data->designation;
    $item->created = date('Y-m-d H:i:s');

    if($item->createEmployee()){
        echo 'Employee created successfully.';
    } else{
        echo 'Employee could not be created.';
    }
?>
```

PHP

Method	Endpoint
POST	http://localhost:8080/api/create.php


POST
http://localhost:8080/api/create.php
Params
Authorization
Headers (9)
Body
none
form-data
x-www-form-urlencoded
{
"name": "Betty Johnson",
"email": "betty@gmail.com",
"age": "26",
"designation": "Singer",
"created": "2012-06-01 02:12:30"
}
Body
Cookies
Headers (9)
Test Results
Pretty
Raw
Preview
Visualize
JSON
Employee created successfully.

Edit/Update MySQL Table using PHP 8 API

This step explains to you how to Update or Edit the data for specific MySQL record. We can use the PHP 8 RESTful API to make the necessary update in the data that is stored in the MySQL database.

Create **update.php** file in the `api` folder and place the following code.

```
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Methods: POST");
    header("Access-Control-Max-Age: 3600");
    header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers, Authorization, X-Requested-With");

    include_once '../config/database.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $item = new Employee($db);

    $data = json_decode(file_get_contents("php://input"));

    $item->id = $data->id;

    // employee values
    $item->name = $data->name;
    $item->email = $data->email;
    $item->age = $data->age;
    $item->designation = $data->designation;
    $item->created = date('Y-m-d H:i:s');

    if($item->updateEmployee()){
        echo json_encode("Employee data updated.");
    } else{
        echo json_encode("Data could not be updated");
    }
?>
```

PHP

Method	Endpoint
POST	http://localhost:8080/api/update.php

POST http://localhost:8080/api/update.php

Params Authorization Headers (9) Body

none form-data x-www-form-urlencoded

```
{
    "id": 26,
    "name": "Samuel Johnson",
    "email": "sam@gmail.com",
    "age": "36",
    "designation": "Writer",
    "created": "2012-06-01 02:12:30"
}
```

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON

```
"Employee data updated."
```

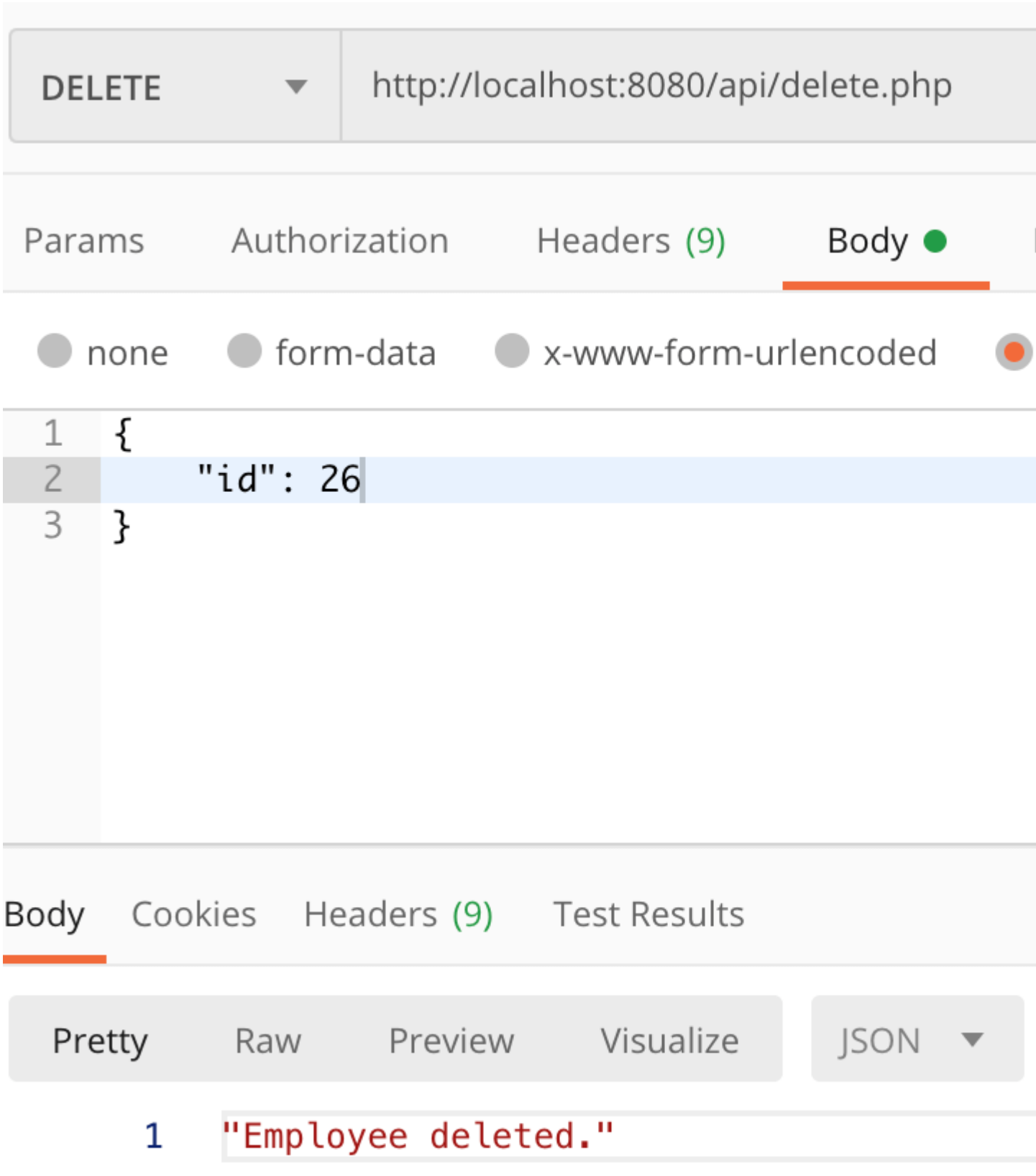
Remove/Delete Single Mysql Record using PHP API

Create **delete.php** file in `api` folder in this file we will write the login to delete or remove the single employee record from MySQL data table using PHP 8 RESTful API. We will make the API call using the `deletedEmployee()` method.

```
<?php
    header("Access-Control-Allow-Origin: *");
    header("Content-Type: application/json; charset=UTF-8");
    header("Access-Control-Allow-Methods: POST");
    header("Access-Control-Max-Age: 3600");
    header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-
Headers, Authorization, X-Requested-With");

    include_once '../config/database.php';
    include_once '../class/employees.php';

    $database = new Database();
    $db = $database->getConnection();

    $item = new Employee($db);

    $data = json_decode(file_get_contents("php://input"));

    $item->id = $data->id;

    if($item->deleteEmployee()){
        echo json_encode("Employee deleted.");
    } else{
        echo json_encode("Data could not be deleted");
    }
?>
```

PHP

Method	Endpoint
DELETE	localhost:8080/api/delete.php


DELETE
http://localhost:8080/api/delete.php
Params
Authorization
Headers (9)
Body
none
form-data
x-www-form-urlencoded
1 {
2 "id": 26
3 }
Body
Cookies
Headers (9)
Test Results
Pretty
Raw
Preview
Visualize
JSON
1 "Employee deleted."

Summary

So this was it, in this tutorial we have learned how to create simple CRUD RESTful API with PHP 8 & MySQL 8, we also explored about the useful PHP methods such as **htmlspecialchars()**, **bindParam()**, **execute()**, PHP PDO and **json_encode()**.

You can download the complete project files from GitHub.

https://www.positronx.io/create-simple-php-crud-rest-api-with-mysql-php-pdo/