# **CRUD Data Operation with Mysql**

Handles data from client to server database

## **Overview**

Since we have already a model created in Laravel application. We can also do create/insert, delete, view and update data from our phpMyAdmin/Mysql database with our application.

## **Create/Insert**

We can add new set of data into our table in the mysql database. Just follow the steps.

### Step 1 – Create new Route

You must created new route to bring the user there and create new todo task. From our **web.php** in route folder. Add new route there.

Route::get('new-todos', [TodosController::class, 'create']);

### Step 2 – Add new function

In your controller add new function same in the one you added in your route in my case its “**create**”.

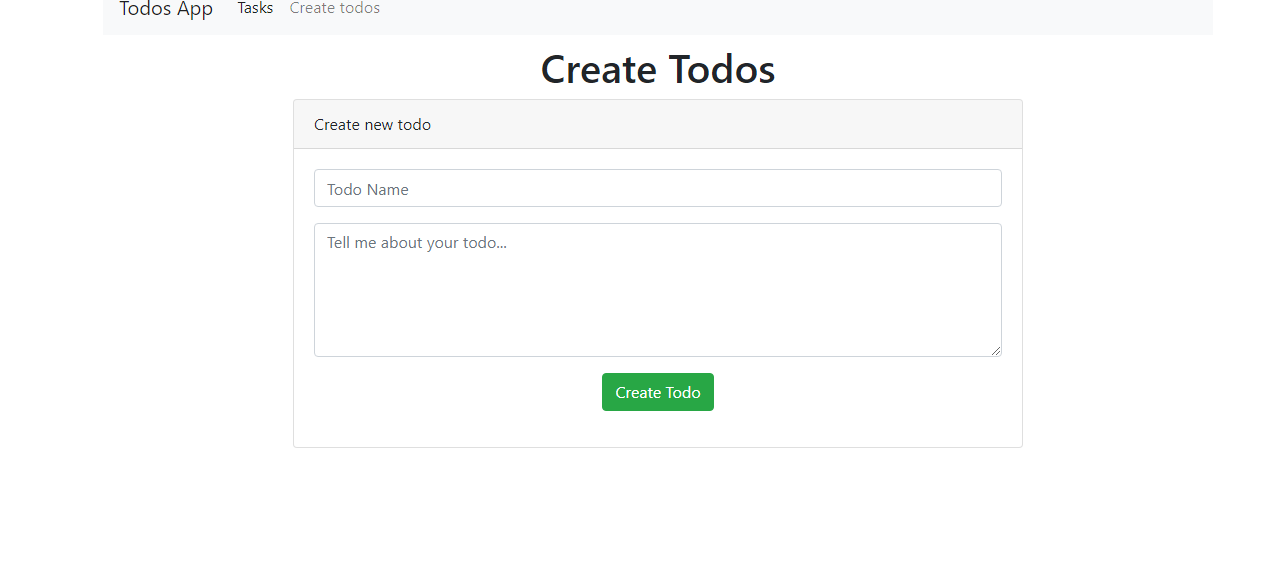
public function create(){

return view('todos.create');

}

### Step 3 – Add new view (Form interface)

Since I return view at my **create()** function in my Controller. You must also create new view in your own. In my case I create new file at my view folder and I put there my HTML Form Interface with Blade/Laravel Templating.



**Note:** Your form must have @csrf template string. This @csrf will generate \_token to validate our request from Laravel upon submission**.** It also prevent from 412 Http Error Request

<form class="todos-form create" action="/save-todos" method="POST">

    @csrf <!--- Same this one  -->

    <div class="form-group">

        <input class="form-control" type="text" name="name" id="name" placeholder="Todo Name">

    </div>

    <div class="form-group">

        <textarea class="form-control" name="description" id="description" cols="5" rows="5" placeholder="Tell me about your todo..."></textarea>

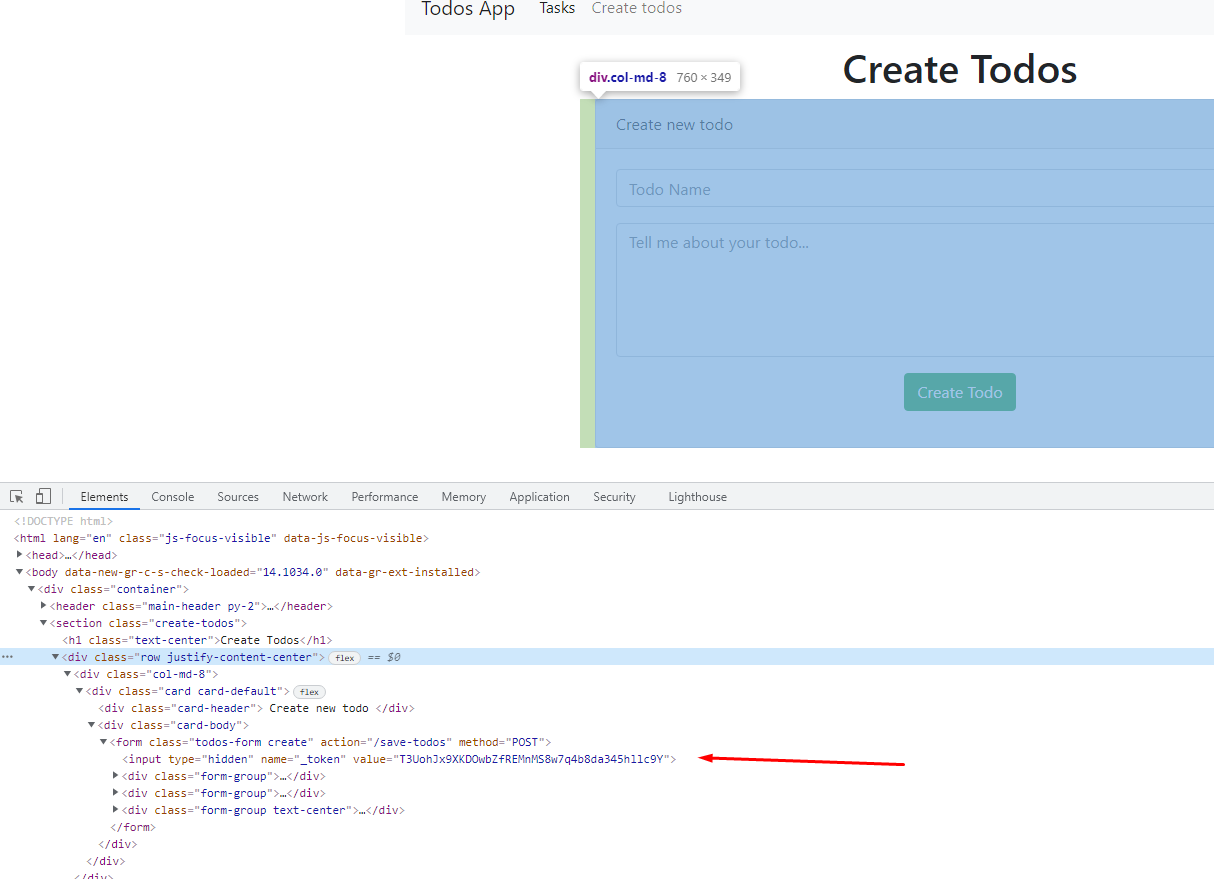
    </div>

    <div class="form-group text-center">

        <button class="btn btn-success">Create Todo</button>

    </div>

</form>

****

### Step 4 – Fetch Data

To fetch/get the Data upon form submission. The HTTP Request POST handles the data. Now we need to create new route in our **web.php**

Route::post('save-todos', [TodosController::class, 'saveTodo']);

This route::post will be called when form action in our create form runs. This only runs when form submissions happens.

<form class="todos-form create" action="/save-todos" method="POST">

Form action= “**/save-todos**” and route::post url(‘**save-todos**’ … must be matched. Since we call saveTodo in your route:post to **TodosController**. You must proceed to add it in controller

public function saveTodo(){

        //code here

}

### Step 5 – Save the Data

Since we get the request. We just need to call the data in POST request and pass it to variable or loop to get.

$data = request()->all();

dd($data);

Then pass it to our model to save it to the database using the **save()** function of our model. Afterwards redirect it to the **/todo** page

public function saveTodo(){

        //code here

        $data = request()->all();

        dd($data);

        $todo = new Todo();

        $todo->name = $data['name'];

        $todo->description = $data['description'];

        $todo->completed = false;

        $todo->save();

        return redirect('/todos');

 }

## **Update**

To update the data table from your database. You need to follow the steps below.

### Step 1 – Create new route

You must created new view route to bring the user there and edit the todo task. From our **web.php** in route folder. Add new route there.

Route::get('todos/{todo}/edit', [TodosController::class, 'editTodo']);

### Step 2 – Add new function

In your controller add new function same in the one you added in your route in my case its “**editTodo**”. Since we pass dynamic argument 'todos/**{todo}/**edit'. We need to create a function that having 1 parameter.

public function editTodo($todoId){

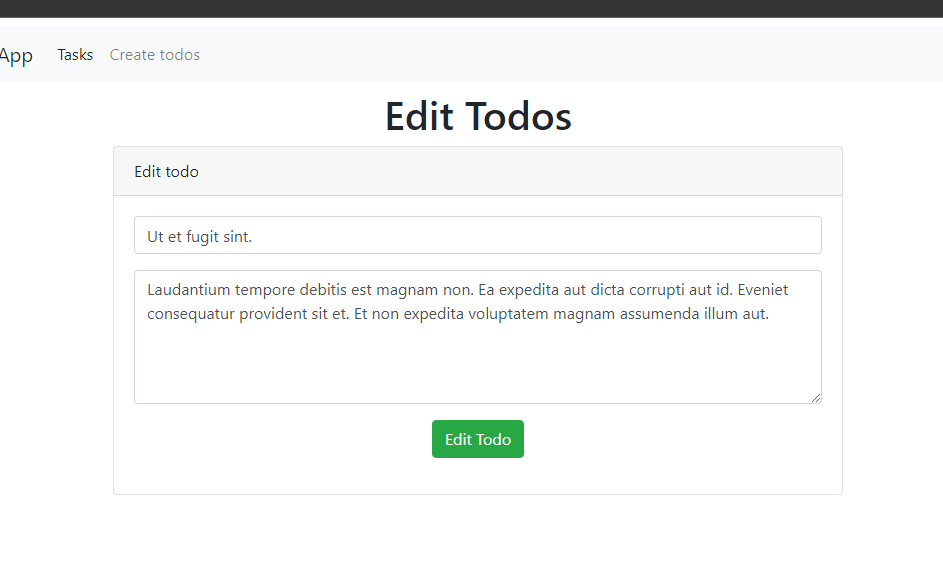
     $todo = Todo::find($todoId);

     return view('todos.edit')->with('todo', $todo);

}

### Step 3 – Add new view (Form interface)

Since I return view at my **editTodo()** function in my Controller. You must also create new view in your own. In my case I create new file at my view folder and I put there my HTML Form Interface with Blade/Laravel Templating.



**Note:** Your form must have @csrf template string. This @csrf will generate \_token to validate our request from Laravel upon submission**.** It also prevent from 412 Http Error Request

<form class="todos-form edit" action="/todos/{{ $todo->id }}/update-todos" method="POST">

       @csrf

      <div class="form-group">

           <input class="form-control" type="text" name="name" id="name" placeholder="Todo Name" value="{{ $todo->name }}">

       </div>

       <div class="form-group">

            <textarea class="form-control" name="description" id="description" cols="5" rows="5" placeholder="Tell me about your todo...">{{ $todo->description }}</textarea>

        </div>

        <div class="form-group text-center">

            <button class="btn btn-success">Edit Todo</button>

       </div>

</form>

**Important:**

1. action="/todos/**{{ $todo->id }}**/update-todos"
2. Populate the input fields from todo from database value="{{ $todo->name }}"

### Step 4 – Fetch Data

To fetch/get the Data upon form submission. The HTTP Request POST handles the data. Now we need to create new route in our **web.php**

Route::post('todos/{todo}/update-todos', [TodosController::class, 'updateTodo']);

This route::post will be called when form action in our edif form runs. This only runs when form submissions happens.

<form class="todos-form edit" action="/todos/{{ $todo->id }}/update-todos" method="POST">

Form action= “/todos/{{ $todo->id }}/update-todos” and route::post url(‘**todos/{todo}/update-todos’** … must be matched. Since we call updateTodo in your route:post to **TodosController**. You must proceed to add it in controller

public function updateTodo($todoId){

        //code here

### Step 5 – Update the data

Since we get the request. We just need to call the data in POST request and pass it to variable or loop to get. Also we need to find the data with specific Id that we pass in our dynamic url “/todos/{{ $todo->id }}/update-todos”

public function updateTodo($todoId){

 $data = request()->all(); //get the request upon form submission

 $todo = Todo::find($todoId); //find data from table with unique id

Then pass it to our model to save it to the database using the **save()** function of our model. Afterwards redirect it to the **/todo** page

public function updateTodo($todoId){

        $this->validate(request(),

            [

                'name' => 'required|min:6',

                'description' => 'required'

            ]

        );

        $data = request()->all();

        $todo = Todo::find($todoId);

        $todo->name = $data['name'];

        $todo->description = $data['description'];

        $todo->save();

        return redirect('/todos');

 }