Laravel AngularJS Admin Panel with RESTful API Demo

In this tutorial, we are going to create a simple CREATE, READ, UPDATE, and DELETE CRUD application. We will use Laravel 5 for the backend and AngularJS for the front end. In accordance with the laws of beauty, we will use twitter bootstrap to add beauty to our simple application.

Step 1: Create new Laravel 5 Application

Open the command prompt or terminal and browser to the root of the web server. On windows assuming you have XAMPP installed to drive C, run the following command

cd C:\xampp\htdocs

The above command browsers to the root directory Run the following command to create a new Laravel project using composer.

composer create-project laravel/laravel EmployeeAdmin

HERE,

- the above code creates a new project in htdocs named angulara

Step 2: Database migrations

We first need to set the database configuration for our application

Open *.env* file in the project root

Set the database configurations as shown below

```
DB_HOST=localhost
DB_DATABASE=angulara
DB_USERNAME=root
DB_PASSWORD=melody
```

Save the changes

Note: use the database name, username and password that match the ones you have on your machine.

We will now use the artisan command to create a migration file that will create a table for employee records.

Run the following artisan command to install the migration table in our database.

php artisan migrate:install

You will get the following message

Migration table created successfully

Run the following artisan command to create a migration file

php artisan make:migration create_employees_table

You will get the following message

Created migration: 2016_06_22_034550_create_employees_table

Let's now modify the newly created migration file

Open /database/migrations/2016_06_22_034550_create_employees_table

Modify the contents to the following

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateEmployeesTable extends Migration {

    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up() {
        Schema::create('employees', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name')->unique();
            $table->string('email')->unique();
            $table->string('contact_number');
            $table->string('position');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down() {
        Schema::drop('employees');
    }
```

```
}
```

Save the changes.

Run the following artisan command to run the migration

php artisan migrate

You will get the following messages

Migrated: 2014_10_12_000000_create_users_table

Migrated: 2014_10_12_100000_create_password_resets_table

*Migrated: 2016_06_22_034550_create_employees_table*Check your database in MySQL. You should have an employees table created.

Step 3: Simple REST API

Let's now create a controller for our REST API.

Run the following artisan command

php artisan make:controller Employees

You will get the following message

Controller created successfully.

Let's now create an Eloquent ORM model for our REST API

php artisan make:model Employee

You will get the following message

Model created successfully.

Let's now add a fillable array to our model Open Employee.php controller in /app/Employee.php

Modify the code to the following

```
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Employee extends Model
{
   protected $fillable = array('id', 'name', 'email','contact_number','position');
}
```

Let's now modify the controller code

Open Employees.php in /app/Http/Controllers/Employees.php

Update the code to the following

```
<?php

namespace App\Http\Controllers;

use App\Employee;
use Illuminate\Http\Request;
use App\Http\Requests;
use App\Http\Controllers\Controller;

class Employees extends Controller {

    /**
     * Display a listing of the resource.
     *
     * @return Response
     */
    public function index($id = null) {
        if ($id == null) {
            return Employee::orderBy('id', 'asc')->get();
        } else {
            return $this->show($id);
        }
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  Request  $request
     * @return Response
     */
    public function store(Request $request) {
        $employee = new Employee;

        $employee->name = $request->input('name');
        $employee->email = $request->input('email');
        $employee->contact_number = $request->input('contact_number');
        $employee->position = $request->input('position');
        $employee->save();
```

```
    return 'Employee record successfully created with id ' . $employee->id;
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return Response
 */
public function show($id) {
    return Employee::find($id);
}

/**
 * Update the specified resource in storage.
 *
 * @param  Request  $request
 * @param  int  $id
 * @return Response
 */
public function update(Request $request, $id) {
    $employee = Employee::find($id);

    $employee->name = $request->input('name');
    $employee->email = $request->input('email');
    $employee->contact_number = $request->input('contact_number');
    $employee->position = $request->input('position');
    $employee->save();

    return "Sucess updating user #" . $employee->id;
}

/**
 * Remove the specified resource from storage.
 *
 * @param  int  $id
 * @return Response
 */
```

```
    public function destroy(Request $request, $id) {
        $employee = Employee::find($id);

        $employee->delete();

        return "Employee record successfully deleted #" . $request->input('id');
    }
}
```

We now need to define the routes for our REST API

Open /app/Http/routes.php

Modify the code to the following

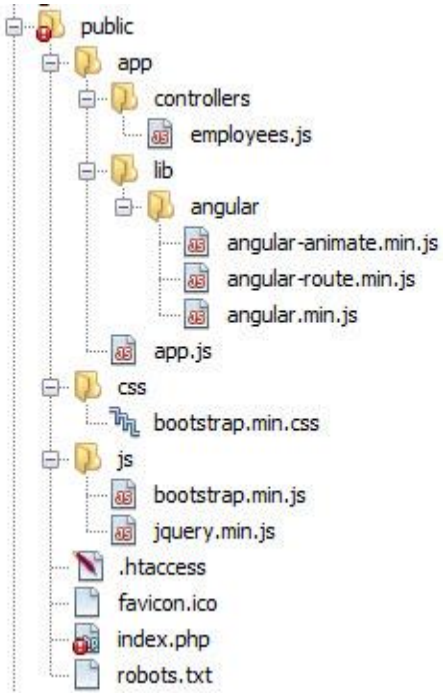
```
<?php

/*
|--------------------------------------------------------------------------
| Application Routes
|--------------------------------------------------------------------------
|
| Here is where you can register all of the routes for an application.
| It's a breeze. Simply tell Laravel the URIs it should respond to
| and give it the controller to call when that URI is requested.
|
*/

Route::get('/', function () {
    return view('index');
});

Route::get('/api/v1/employees/{id?}', 'Employees@index');
Route::post('/api/v1/employees', 'Employees@store');
Route::post('/api/v1/employees/{id}', 'Employees@update');
Route::delete('/api/v1/employees/{id}', 'Employees@destroy');
```

That was it for our REST API. Let’s now create the frontend using AngularJS

Step 4: AngularJS front-end

AngularJS application structure

Our application will have the following structure



HERE,

- app – contains all AngularJS related JavaScript files
- app/controllers – contains all AngularJS controllers
- app/lib – this directory will contain all AngularJS core files. You can also load AngularJS from a CDN network.
- css – contains all CSS files
- js – contains all regular JavaScript files for our UI.

Create the directories as shown in the above image

AngularJS app.js

This file will be used to define our application

Create a new file /public/app/app.js

Add the following code to it

```
var app = angular.module('employeeRecords', [])
        .constant('API_URL', http://yourserver/api/v1/');
```

HERE,

- `var app = angular.module('employeeRecords', [])` creates an AngularJS module and assigns the object to the variable app. All AngularJS files will be reference the variable app

- `.constant('API_URL', 'http://localhost/angulara/public/api/v1/');` defines a constant variable with the API URL.

AngularJS controllers employees.js

This is the file that will be responsible for interacting with our API

Create a new file /public/app/controllers/employees.js

Add the following code to it

```
app.controller('employeesController', function($scope, $http, API_URL) {
    //retrieve employees listing from API
    $http.get(API_URL + "employees")
            .success(function(response) {
                $scope.employees = response;
            });

    //show modal form
    $scope.toggle = function(modalstate, id) {
        $scope.modalstate = modalstate;

        switch (modalstate) {
            case 'add':
                $scope.form_title = "Add New Employee";
                break;
            case 'edit':
                $scope.form_title = "Employee Detail";
                $scope.id = id;
                $http.get(API_URL + 'employees/' + id)
                        .success(function(response) {
                            console.log(response);
                            $scope.employee = response;
                        });
                break;
            default:
                break;
        }
        console.log(id);
        $('#myModal').modal('show');
    }

    //save new record / update existing record
    $scope.save = function(modalstate, id) {
        var url = API_URL + "employees";

        //append employee id to the URL if the form is in edit mode
        if (modalstate === 'edit'){
```

```
            url += "/" + id;
        }

        $http({
            method: 'POST',
            url: url,
            data: $.param($scope.employee),
            headers: {'Content-Type': 'application/x-www-form-urlencoded'}
        }).success(function(response) {
            console.log(response);
            location.reload();
        }).error(function(response) {
            console.log(response);
            alert('This is embarassing. An error has occured. Please check the log
for details');
        });
    }

    //delete record
    $scope.confirmDelete = function(id) {
        var isConfirmDelete = confirm('Are you sure you want this record?');
        if (isConfirmDelete) {
            $http({
                method: 'DELETE',
                url: API_URL + 'employees/' + id
            }).
                    success(function(data) {
                        console.log(data);
                        location.reload();
                    }).
                    error(function(data) {
                        console.log(data);
                        alert('Unable to delete');
                    });
        } else {
            return false;
        }
    }
});
```

HERE,

```
app.controller('employeesController', function($scope, $http, APIURL){…}
```

defines a controller employeesController in the app variable that we created in /app/app.js. We have injected $scope, $http, and a contant APIURL as dependencies

```
$http.get(APIURL + "employees").success(function(response) {$scope.employees =
response;});
```

uses Angular $http to call the API. APIURL + "employees" is passed as a parameter to $http. If the call is successful, the response to passed to .success anonymous function. The anonymous function assigns the response to $scope.employees variable. The $scope.employees variable will be available in our view.

- `$scope.toggle = function(modalstate, id) {…}` displays the modal form

- `$scope.save = function(modalstate, id){…}` saves a new record / updates an existing record
- `$scope.confirmDelete = function(id){…}` deletes an existing record

Step 5: Displaying data from the REST API using AngularJS

We will now create a view that displays the data from the REST API. Both blade template and AngularJS use double curly braces to display data. In order to avoid conflicts between the two, we will not save the view as a blade template. It will be a regular view.

Create a new file in /resources/views/index.php

Add the following code

```
<!DOCTYPE html>
<html lang="en-US" ng-app="employeeRecords">
    <head>
        <title>Laravel 5 AngularJS CRUD Example</title>

        <!-- Load Bootstrap CSS -->
        <link href="<?= asset('css/bootstrap.min.css') ?>" rel="stylesheet">
    </head>
    <body>
        <h2>Employees Database</h2>
        <div  ng-controller="employeesController">

            <!-- Table-to-load-the-data Part -->
            <table class="table">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Email</th>
                        <th>Contact No</th>
                        <th>Position</th>
                        <th><button id="btn-add" class="btn btn-primary btn-xs" ng-
click="toggle('add', 0)">Add New Employee</button></th>
                    </tr>
                </thead>
                <tbody>
                    <tr ng-repeat="employee in employees">
                        <td>{{  employee.id }}</td>
                        <td>{{ employee.name }}</td>
                        <td>{{ employee.email }}</td>
                        <td>{{ employee.contact_number }}</td>
```

```
                <td>{{ employee.position }}</td>
                <td>
                    <button class="btn btn-default btn-xs btn-detail" ng-
click="toggle('edit', employee.id)">Edit</button>
                    <button class="btn btn-danger btn-xs btn-delete" ng-
click="confirmDelete(employee.id)">Delete</button>
                </td>
            </tr>
        </tbody>
    </table>
    <!-- End of Table-to-load-the-data Part -->
    <!-- Modal (Pop up when detail button clicked) -->
    <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel" aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">×</span></button>
                    <h4 class="modal-title" id="myModalLabel">{{form_title}}</h4>
                </div>
                <div class="modal-body">
                    <form name="frmEmployees" class="form-horizontal" novalidate="">


                        <div class="form-group error">
                            <label for="inputEmail3" class="col-sm-3 control-label">Name</label>
                            <div class="col-sm-9">
                                <input type="text" class="form-control has-
error" id="name" name="name" placeholder="Fullname" value="{{name}}"
                                ng-model="employee.name" ng-required="true">
                                <span class="help-inline"
                                ng-
show="frmEmployees.name.$invalid && frmEmployees.name.$touched">Name field is required</spa
n>
                            </div>
                        </div>


                        <div class="form-group">
                            <label for="inputEmail3" class="col-sm-3 control-label">Email</label>
                            <div class="col-sm-9">
                                <input type="email" class="form-
control" id="email" name="email" placeholder="Email Address" value="{{email}}"
```

```
                        ng-model="employee.email" ng-required="true">
                        <span class="help-inline"
                        ng-
show="frmEmployees.email.$invalid && frmEmployees.email.$touched">Valid Email field is required<
/span>
                    </div>
                </div>


                <div class="form-group">
                    <label for="inputEmail3" class="col-sm-3 control-
label">Contact Number</label>
                    <div class="col-sm-9">
                        <input type="text" class="form-
control" id="contact_number" name="contact_number" placeholder="Contact Number" va
lue="{{contact_number}}"
                                        ng-model="employee.contact_number" ng-
required="true">
                                    <span class="help-inline"
                                        ng-
show="frmEmployees.contact_number.$invalid && frmEmployees.contact_number.$touched
">Contact number field is required</span>
                                    </div>
                                </div>


                                <div class="form-group">
                                    <label for="inputEmail3" class="col-sm-
3 control-label">Position</label>
                                    <div class="col-sm-9">
                                        <input type="text" class="form-
control" id="position" name="position" placeholder="Position" value="{{position}}"


                                        ng-model="employee.position" ng-
required="true">
                                    <span class="help-inline"
                                        ng-
show="frmEmployees.position.$invalid && frmEmployees.position.$touched">Position f
ield is required</span>
                                    </div>
                                </div>


                            </form>
                        </div>
                        <div class="modal-footer">
```

```
                            <button type="button" class="btn btn-primary" id="btn-save" ng-click="save(modalstate, id)" ng-disabled="frmEmployees.$invalid">Save changes</button>
                        </div>
                    </div>
                </div>
            </div>
        </div>

        <!-- Load Javascript Libraries (AngularJS, JQuery, Bootstrap) -->
        <script src="<?= asset('app/lib/angular/angular.min.js') ?>"></script>
        <script src="<?= asset('js/jquery.min.js') ?>"></script>
        <script src="<?= asset('js/bootstrap.min.js') ?>"></script>

        <!-- AngularJS Application Scripts -->
        <script src="<?= asset('app/app.js') ?>"></script>
        <script src="<?= asset('app/controllers/employees.js') ?>"></script>
    </body>
</html>
```

HERE,

- `<html lang="en-US" ng-app="employeeRecords"> ng-app="employeeRecords"` attached our AngularJS application to the html tag. This will give it control over all elements in html tag.
- `<div ng-controller="employeesController">` links the div to the employeesController. This will make available all of the functions under employeesController to this div.
- `<tr ng-repeat="employee in employees">` used the AngularJS directive ng-repeat to loop through the results of the collection variable employees. ng-repeat is similar to the foreachloop.

Load the following URL in your web browser

http://localhost/EmployeeAdmin/

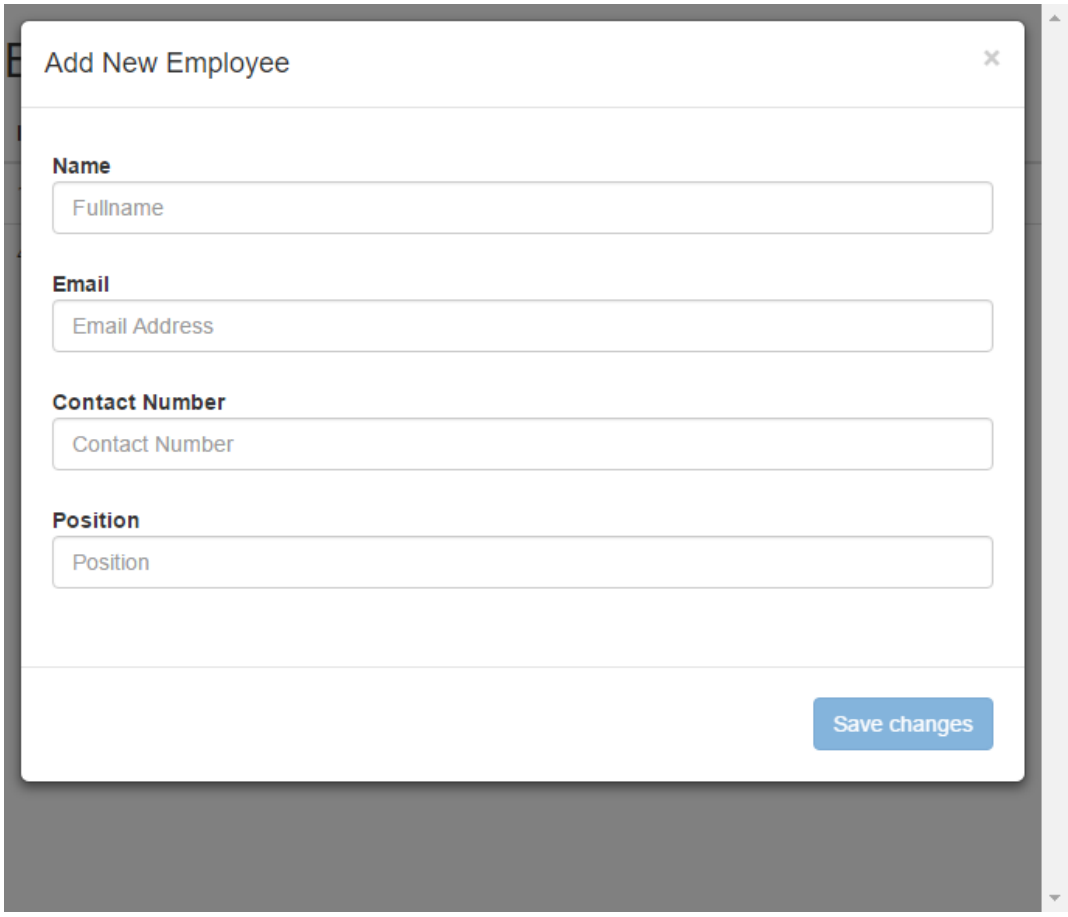
You will get the following

Employees Database

ID	Name	Email	Contact No	Position	Add New Employee
1	yongjiang zhang	tinnnng@gmail.com	123456789	Tutor	Edit Delete
4	dasdasda	dasdasd@gmai.com	fafd	dadasd	Edit Delete

Click on Add New Employee button

You will get the following modal form



Add more employees, edit existing record and even delete some

Use the comments section below if you get any errors. Our team will respond to you.

Step 6: AngularJS form validation

AngularJS simplifies the process of validating forms.

Locate the code for the form and have a look at it

HERE,

- `<form name="frmEmployees" class="form-horizontal" novalidate="">` defines a form frmEmployees and add the novalidate attribute to stop HTML5 from validating our form
- `<input type… ng-model="employee.name" ng-required="true">` ng-model is used for data binding. For example, anything entered in name text box is made available to employee.name variable. When AngularJS changes the value of employee.name, it is made available to the textbox too.ng-required= "true" validates our form and checks if a value has been supplied. If no value is supplied, a glass of $invalid is added to our form
- `<span class="help-inline" ng-show="frmEmployees.name.$invalid && frmEmployees.name.$touched">`Name field is required ng-show only displays this element if the name text box has an invalid class
- `ng-disabled="frmEmployees.$invalid"` disables the submit button if the form has an invalid class. If the user enters all required details in the correct format, the submit button is enabled.

Summary

AngularJS is a powerful client-side MVC framework that simplifies developing frontend parts of a web applications. Subscribe to our free newsletter and we will let you know when we publish tutorial series on AngularJS.