Laravel

Practical 6

James Ooi

Apr 2018

Contents

1 Managing Assets

In this practical, we will implement client-side scripting (JavaScript) in our Travelife Club app. Prior to writing JavaScript, we will need to set up our JavaScript scaffolding and manage our JavaScript assets which will be included in our app.

We will be looking at two popular JavaScript libraries in our practical classes: **React** and **jQuery**.

1.1 Configure React Scaffolding

To use **React** JavaScript library in our app, configure the preset to React scaffolding:

```
php artisan preset react
```

When completed, you must run the **npm install** and **npm run dev** NPM commands again as instructed.

If you are using a mixture of **React** and **jQuery** libraries in your app, leave the preset configured to React scaffolding.

However, if you are using **jQuery** only, you may remove the React scaffolding with this command:

```
php artisan preset none
```

1.2 Laravel Mix

In your **webpack.mix.js** file, you will notice that it has been configured to use React if you have run the abovementioned **preset** command.

Your **webpack.mix.js** file should contain the following code:

```
1 mix.react('resources/assets/js/app.js', 'public/js')
2    .sass('resources/assets/sass/app.scss', 'public/css');
```

In this case, the source of your React scripts are specified in **resources/assets/js/app.js** while your SASS styles are specified in **resources/assets/sass/app.scss**.

Running the **npm run dev** or **npm run production** command will compile your scripts and styles to the directory specified in the second parameter. In this case, your script file will be compiled into **public/js/app.js** while your style file will be compiled into **public/css/app.js**.

You may wish to change the default source and destination script and style file as you wish, depending on your application requirements.

However, for the time being, we shall leave the defaults as they are.

2 Getting Started with React

Open the **resources/assets/js/app.js** file. Your file should look like this:

```
/**
 * First we will load all of this project's JavaScript dependencies which
 * includes React and other helpers. It's a great starting point while
 * building robust, powerful web applications using React + Laravel.
 */

require('./bootstrap');

/**
 * Next, we will create a fresh React component instance and attach it to
 * the page. Then, you may begin adding components to this application
 * or customize the JavaScript scaffolding to fit your unique needs.
 */

require('./components/Example');
```

You will notice that your **app.js** file specifies two **require** statements to require the **bootstrap.js** and **components/Example.js** files. When using **require**, you do not have to specify the file extension as it expects all your JavaScript files to have the **.js** extension.

Take a look at the **bootstrap.js** file. This file bootstraps (do NOT confuse this with Bootstrap CSS) any JavaScript dependencies that your app may require, such as the **jQuery** library. You may modify this file to specify any dependencies that you may need.

The **components/Example.js** file is an example of a React **component** that is automatically generated. You may take a look at the file to see how you can create a simple React component.

As we will be building our own React components, comment or remove the **require('./components/Example');** statement in your **app.js** file.

2.1 Building a React Component

Here, we will build a React component to display attributes of our **Division** model. We will modify the existing code in our app to fetch data from the server and dynamically render the component with the data on the client side.

2.1.1 The API Routes

To begin, we shall define the following API route in the **routes/api.php** file:

```
Route::get('/division/{division}', 'DivisionController@apiShow')
    ->name('division.api-show');
```

Accessing this route will require the prefix **api** in our URL using default settings. Accessing the above route is therefore via the URL **/api/division/2** to access the route with the parameter value equals to **2**.

As we are using the **web** middleware for authentication, change the API authentication to use the **web** middleware instead of **api** middleware in the **app/Providers/RouteServiceProvider.php** file as follows:

```
/**
 * Define the "api" routes for the application.
 *
 * These routes are typically stateless.
 *
 * @return void
 */
protected function mapApiRoutes()
{
    Route::prefix('api')
         ->middleware('web')
         ->namespace($this->namespace)
         ->group(base_path('routes/api.php'));
}
```

2.1.2 The Controller Methods

Define the **apiShow** method in **DivisionController** as follows:

```
/**
 * Returns resource in JSON
 *
 * @param  int  $id
 *
 * @return \Illuminate\Http\Response
 */
public function apiShow($id)
{
    $division = Division::find($id);

    if($division) {
        return $division;
    }
    else {
        return response()->json(null);
    }
}
```

Go to the URL **http://localhost:8088/api/division/{division}** and ensure that you get the corresponding JSON data for the division specified by the parameter **{division}**. Take note

that you must have logged in to your app before trying to do so as we have implemented authentication.

2.1.3 Specifying the JavaScript Asset

In your **layouts/app.blade.php** view file, make sure the following line is present just before the closing **body** tag:

```
<script src="{{asset('js/app.js')}}"></script>
```

2.1.4 The View Template

Modify the **divisions/show.blade.php** file as follows:

```
@extends('layouts.app')

@section('content')
    <!-- Bootstrap Boilerplate... -->
    <div id="division-show" class="panel-body"></div>
    <script>
        var __props = {
            url: "{{ route('division.api-show', $division->id) }}",
        };
    </script>
@endsection
```

2.1.5 Some Changes to the Model

If you have noticed the JSON output, the value of the **state** attribute for each **Division** model is as retrieved from the **divisions** table, i.e. the 2-digit code that we defined for each state.

However, we would like to display the full state name. To do so, define the following *mutator* method in our **Division** model class:

```
/**
 * Get the descriptive state name.
 *
 * @return string
 */
public function getStateNameAttribute()
{
    return Common::$states[$this->state];
}
```

The method **getStateNameAttribute()** defined above will allow us to access the attribute using the property **state_name**.

We now define the **appends** property in the same model class to specify **state_name** as the attribute that will be appended to the model instance:

```
protected $appends = [
    'state_name',
];
```

Try accessing **http://localhost:8088/api/division/{division}** again and ensure that the **state_name** attribute is now present in the JSON output.

2.1.6 The React Component

Create a file named **Division.js** in **resources/assets/js/components** as follows:

```
import React, { Component } from 'react';
import ReactDOM from 'react-dom';

export default class Division extends Component {
    /**
     * Constructor
     *
     */
    constructor(props) {
        super(props);

        //Initialize the state in the constructor
        this.state = {
            fetched: false,
            division: null,
        }
    }

    /**
     * This is a lifecycle method that gets called after
     * the component is rendered.
     * Here, we fetch the JSON data of albums from the API.
     */
    componentDidMount() {
        var url = this.props.url;

        fetch(url, {
            headers: {Accept: 'application/json'},
            credentials: 'same-origin',
        })
        .then(response => {
            if (response.ok) return response.json();
            else throw Error([response.status, response.statusText].join(' '));
        })
        .then(division => {
```

```
36              this.setState({ fetched: true });
37              this.setState({ division });
38          })
39          .catch(error => alert(error));
40      }
41
42      renderHeadings() {
43          return (
44              <thead>
45                  <tr>
46                      <th>Attribute</th>
47                      <th>Value</th>
48                  </tr>
49              </thead>
50          )
51      }
52
53      renderDivision() {
54          return (
55              <table className="table table-striped">
56                  { this.renderHeadings() }
57                  <tbody>
58                      <tr>
59                          <td>Code</td>
60                          <td>{ this.state.division.code }</td>
61                      </tr>
62                      <tr>
63                          <td>Name</td>
64                          <td>{ this.state.division.name }</td>
65                      </tr>
66                      <tr>
67                          <td>Address</td>
68                          <td style={ {
69                              whiteSpace: 'pre-line',
70                          } }>{ this.state.division.address }</td>
71                      </tr>
72                      <tr>
73                          <td>Postcode</td>
74                          <td>{ this.state.division.postcode }</td>
75                      </tr>
76                      <tr>
77                          <td>City</td>
78                          <td>{ this.state.division.city }</td>
79                      </tr>
80                      <tr>
81                          <td>State</td>
82                          <td>{ this.state.division.state_name }</td>
83                      </tr>
84                      <tr>
85                          <td>Created</td>
```

```
                        <td>{ this.state.division.created_at }</td>
                    </tr>
                </tbody>
            </table>
        )
    }

    renderLoader() {
        return (
            <div>
                Loading division...
            </div>
        );
    }

    render() {
        if(this.state.fetched && this.state.division) {
            return this.renderDivision();
        }
        else {
            return this.renderLoader();
        }
    }
}

(() => {
    var element = document.getElementById('division-show');
    if(__props && element) {
        ReactDOM.render(<Division {...__props} />, element);
    }
})();
```

2.1.7 Requiring Our React Component

Now add the following **require** statement to our **app.js** file:

```
require('./components/Division');
```

2.1.8 Test It Out

Go to **http://localhost:8088/division/{division}** and you should see our **Division** React component rendered on the screen.

2.2 Build Another React Component

Let's build another React component named **Divisions.js** to generate a table dynamically to display a listing of divisions. Test it out after completing it.

2.2.1 The API Routes

```
Route::get('/division', 'DivisionController@apiIndex')
    ->name('division.api-index');
```

2.2.2 The Controller Methods

```
/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function index()
{
    return view('divisions.index');
}

/**
 * Returns resources in JSON
 *
 * @return \Illuminate\Http\Response
 */
public function apiIndex()
{
    $divisions = Division::orderBy('name', 'asc')->get();

    return $divisions;
}
```

2.2.3 The View Template

```
@extends('layouts.app')

@section('content')
    <!-- Bootstrap Boilerplate... -->
    <div id="division-index" class="panel-body"></div>
    <script>
        var __props = {
            url: "{{ route('division.api-index') }}",
        };
    </script>
@endsection
```

2.2.4 The React Component

```
1  import React, { Component } from 'react';
2  import ReactDOM from 'react-dom';
3
4  export default class Divisions extends Component {
5      /**
6       * Constructor
7       *
8       */
9      constructor(props) {
10         super(props);
11
12         //Initialize the state in the constructor
13         this.state = {
14             fetched: false,
15             divisions: null,
16         }
17     }
18
19     /**
20      * This is a lifecycle method that gets called after
21      * the component is rendered.
22      * Here, we fetch the JSON data of albums from the API.
23      */
24     componentDidMount() {
25         var url = this.props.url;
26
27         fetch(url, {
28             headers: {Accept: 'application/json'},
29             credentials: 'same-origin',
30         })
31         .then(response => {
32             if (response.ok) return response.json();
33             else throw Error([response.status, response.statusText].join(' '));
34         })
35         .then(divisions => {
36             this.setState({ fetched: true });
37             this.setState({ divisions });
38         })
39         .catch(error => alert(error));
40     }
41
42     renderHeadings() {
43         return (
44             <thead>
45                 <tr>
46                     <th>No.</th>
47                     <th>Code</th>
48                     <th>Name</th>
```

```
                    <th>City</th>
                    <th>State</th>
                    <th>Created</th>
                    <th>Actions</th>
                </tr>
            </thead>
        )
    }

    renderDivision() {
        return this.state.divisions.map((division, i) => {
            return (
                <tr key={ division.id }>
                    <td className="table-text">
                        <div>{ i + 1 }</div>
                    </td>
                    <td className="table-text">
                        <div>
                            <a href={ ['division', division.id].join('/') }>
                               { division.code }
                            </a>
                        </div>
                    </td>
                    <td className="table-text">
                        <div>{ division.name }</div>
                    </td>
                    <td className="table-text">
                        <div>{ division.city }</div>
                    </td>
                    <td className="table-text">
                        <div>{ division.state_name }</div>
                    </td>
                    <td className="table-text">
                        <div>{ division.created_at }</div>
                    </td>
                    <td className="table-text">
                        <div>
                            <a href={ ['division', division.id, 'edit'].join('/') }>
                               Edit
                            </a>
                        </div>
                    </td>
                </tr>
            );
        });
    }

    renderTable() {
        return (
            <table className="table table-striped">
```

```
99                  { this.renderHeadings() }
100                 <tbody>
101                     { this.renderDivision() }
102                 </tbody>
103             </table>
104         );
105     }
106
107     renderEmpty() {
108         return (
109             <div>
110                 No records found
111             </div>
112         );
113     }
114
115     renderLoader() {
116         return (
117             <div>
118                 Loading divisions...
119             </div>
120         );
121     }
122
123     render() {
124         if(this.state.fetched && this.state.divisions) {
125             if(this.state.divisions.length) {
126                 return this.renderTable();
127             }
128             else {
129                 return this.renderEmpty();
130             }
131         }
132         else {
133             return this.renderLoader();
134         }
135     }
136 }
137
138 (() => {
139     var element = document.getElementById('division-index');
140     if(__props && element) {
141         ReactDOM.render(<Divisions {...__props} />, element);
142     }
143 })();
```

2.2.5 Compiling your JavaScript

Now, compile your code using the following NPM command:

```
npm run dev
```

If you would like to have NPM watches your script and style files for changes and compiles them automatically instead, use the following NPM command:

```
npm run watch-poll
```

Note that once you have completed building your components and will like to publish your app into the production server, ensure that you compile your scripts and styles in production mode as this minifies the output JavaScript and CSS files to reduce file size:

```
npm run production
```

3 Getting Started with jQuery

In this section, we shall reimplement the similar display and listing of a single division and a list of divisions using **jQuery** instead of React.

3.1 Configuring webpack.mix.js

Add the following statement to your **webpack.mix.js** file:

```
mix.js([
    'resources/assets/js/division.js',
], 'public/js/all.js');
```

In this case, we will be writing our JavaScript code in the file **resources/assets/js/division.js**. The script will be compiled to the output file **public/js/all.js**.

3.2 Specifying the JavaScript Asset

Comment out the existing asset that you defined in the **layouts/app.blade.php** file when you were using React earlier and add in the following:

```
{{--<script src="{{ asset('js/app.js') }}"></script>--}}
    <script src="{{ asset('js/all.js') }}"></script>
```

Now, we will be referring to **js/all.js** which will contain our JavaScript code which uses **jQuery** instead of the **js/app.js** which uses React.

3.3 Writing the JavaScript Code

Create a file **resources/assets/js/division.js** as follows:

```
(function(){
    window.$ = window.jQuery = require('jquery');

    function renderDivisions(wrapper)
    {
        $.ajax({
            url: __props.url,
            success: function(divisions) {
                if(divisions.length) {
                    var table = $('<table>')
                        .addClass('table')
                        .addClass('table-striped');

```

```
14                      var headings = $('<thead>')
15                          .append(
16                              $('<tr>')
17                                  .append($('<th>').text('No.'))
18                                  .append($('<th>').text('Code'))
19                                  .append($('<th>').text('Name'))
20                                  .append($('<th>').text('City'))
21                                  .append($('<th>').text('State'))
22                                  .append($('<th>').text('Created'))
23                                  .append($('<th>').text('Actions'))
24                          );
25
26                      var tbody = $('<tbody>');
27
28                      table.append(headings)
29                          .append(tbody);
30
31                      $.each(divisions, function(i, division) {
32                          var tr = $('<tr>')
33                              .append($('<td>').text(i + 1))
34                              .append($('<td>').text(division.code))
35                              .append($('<td>').text(division.name))
36                              .append($('<td>').text(division.city))
37                              .append($('<td>').text(division.state_name))
38                              .append($('<td>').text(division.created_at))
39                              .append($('<td>').text('Edit'));
40
41                          tbody.append(tr);
42                      });
43
44                      wrapper.append(table);
45                  }
46                  else {
47                      wrapper.append(
48                          $('<div>').text('No records found')
49                      );
50                  }
51              },
52          });
53      }
54
55      function renderDivision(wrapper)
56      {
57          $.ajax({
58              url: __props.url,
59              success: function(division) {
60                  var table = $('<table>')
61                      .addClass('table')
62                      .addClass('table-striped');
63
```

```
64                  var headings = $('<thead>')
65                      .append(
66                          $('<tr>')
67                              .append($('<th>').text('Attribute'))
68                              .append($('<th>').text('Value'))
69                      );
70
71                  var tbody = $('<tbody>');
72
73                  table.append(headings)
74                      .append(tbody);
75
76                  tbody
77                      .append(
78                          $('<tr>')
79                              .append($('<td>').text('Code'))
80                              .append($('<td>').text(division.code))
81                      )
82                      .append(
83                          $('<tr>')
84                              .append($('<td>').text('Name'))
85                              .append($('<td>').text(division.name))
86                      )
87                      .append(
88                          $('<tr>')
89                              .append($('<td>').text('Address'))
90                              .append($('<td>').text(division.address))
91                      )
92                      .append(
93                          $('<tr>')
94                              .append($('<td>').text('Postcode'))
95                              .append($('<td>').text(division.postcode))
96                      )
97                      .append(
98                          $('<tr>')
99                              .append($('<td>').text('City'))
100                             .append($('<td>').text(division.city))
101                     )
102                     .append(
103                         $('<tr>')
104                             .append($('<td>').text('State'))
105                             .append($('<td>').text(division.state_name))
106                     )
107                     .append(
108                         $('<tr>')
109                             .append($('<td>').text('Created'))
110                             .append($('<td>').text(division.created_at))
111                     );
112                 wrapper.append(table);
113             },
```

```
114         });
115     }
116
117     if($('#division-index')) {
118         renderDivisions($('#division-index'));
119     }
120
121     if($('#division-show')) {
122         renderDivision($('#division-show'));
123     }
124 })();
```

3.4 Compile and Test

Now, compile your code and test it out!

4 Exercises (Unguided)

Modify your application to use either **React** or **jQuery** JavaScript libraries for the following:

- The **index** and **show** methods of **MemberController**.
- The **index** and **show** methods of **GroupController**.