JARED GRAY (/)

(http://github.com/jd-gray) (http://stackoverflow.com/users/4735914/jdgray?tab=profile) (https://www.linkedin.com/in/jareddgray)

Sunday, July 23

Build Rails forms using Vuejs and Webpacker

About a month ago I started using Vuejs with a work project and I enjoyed every minute of it. I want to spread a bit of knowledge on how easy it is to build a form using Vuejs with Rails. Overall, this is a very high level of using Vuejs in a Rails application. I do hope that it helps out others get started and make adjustments along the way.

First lets create our Rails application. This project will use Rails 5.1.2.

```
rails new vue-js-rails --database=postgresql --webpack=vue
```

Notice the `webpack` tag. This is where we will use webpacker (https://github.com/rails/webpacker) with Vuejs.

Next lets scaffold together a `Todo` model, controller, and views:

```
rails g scaffold todo title content
```

This is a bit overkill as you will see we are only going to be using the `index` and `create` actions.

Then create and migrate the database:

```
rails db:create db:migrate
```

Afterwards lets get the Rails app started up and confirm we are good to go!

```
rails s
http://localhost:3000/
```

You should see the "Yay! You're on Rails!" welcome page. Lets now exit our local server and start construction of the create form.

First lets set our root path to our `todos/index.html.erb`:

```
# routes.rb
root to: 'todos#index'
```

Delete everything in `todos/index.html.erb` to start with a fresh canvas.

We will need to install vue-resource (https://github.com/pagekit/vue-resource) as we will be using web requests to create our `Todo`'s. I am using `yarn`, but the documentation uses a few ways to install.

```
yarn add vue-resource
```

I am going to skip `verify_authenticity_token` for now since the javascript form does not include Rails csrf protection token. I invite you to come up with a solution to get around this. Here is a Stackoverflow post (https://stackoverflow.com/questions/8503447/rails-how-to-add-csrf-protection-to-forms-created-in-javascript) that should help.

```
# todos_controller.rb
skip_before_action :verify_authenticity_token
```

Now onto creating out Vuejs form. Create `todo_form.js` in `javascript/packs/`. This will be necessary so we can use `javascript_pack_tag` in our `index.html.erb` view.

```
# todo_form.js
import Vue from 'vue'
import App from './app.vue'

document.addEventListener('DOMContentLoaded', () => {
  document.body.appendChild(document.createElement('todo_form'));
  const app = new Vue(App).$mount('todo_form');
});
```

We can now edit and change our `javascript/packs/app.vue` page to accomodate our Vuejs form. This is were we will create the input's and submit functionality. This design uses a `template`, `script`, and `style` guideline. We will create all of our view elements, javascript, and styling on this page.

```
# app.vue

<template>
  <div id="app">
    <label for="title">Title:</label>
    <input v-model="title" placeholder="Title">

    <label for="content">Content:</label>
    <input v-model="content" placeholder="Content">

    <button v-on:click="submit">Submit</button>
  </div>
</template>

<script>
  import Vue from 'vue'
  import VueResource from 'vue-resource'
  import App from './app.vue'

  Vue.use(VueResource);

  export default {
    data: {
      title: '',
      content: ''
    },
    methods: {
      submit: function () {
        this.$http.post('/todos', { title: this.title, content: this.content });
        this.title = '';
        this.content = '';
      }
    }
  }
</script>

<style scoped>
</style>
```

This can look a bit intimidating, but it really shouldn't be. Within our `<template>` tags we are constructing the inputs and submit button. The v-model (https://vuejs.org/v2/guide/forms.html) attribute two-way binds the input's value with our variables defined in our `data` object. The `v-on:click` links our submission with the `submit` function in our `methods` object. This is where we will use `vue-resource` as the `$http.post` publishes our parameters to our `create` method within the `TodosController`.

Now that we have our form created, lets include it in our view:

```
# todos/index.html.erb
<%= javascript_pack_tag 'todo_form' %>
```

Lets start up the Rails server and webpack development server, Make sure each of these are done in a separate tab.

```
rails s
./bin/webpack-dev-server
```

You should now see the form and be able to create a `Todo`! But we are not displaying the `Todo` in the view. Let's exit our local server's and create the functionality to view our `Todo`'s without refreshing the page.

I will be using the `gon` gem to access our Ruby variables in our Javascript files.

```
# Gemfile
gem 'gon'
```

Install by running `bundle install`.

Finally include `gon` in our `application.html.erb` file:

```
# application.html.erb
<%= Gon::Base.render_data %>
```

Yay! `gon` setup is complete!

Lets pass our `Todo`'s to our javascript file:

```
# todos_controller.rb
def index
...
  gon.todos = @todos
end
```

We can now access our `Todo`'s by using `gon`. Notice that I added `todos` in the `data` object. I also added a callback for our response. Once a `Todo` is created, it will be added to the `todos` array and can then be added to our view.

```
# app.vue
    data: {
      title: '',
      content: '',
      todos: gon.todos
    },
    methods: {
      submit: function () {
        this.$http.post('/todos', { title: this.title, content: this.content }).then(response => {
          this.todos.unshift(response.body);
        });
        this.title = '';
        this.content = '';
      }
    }
```

Now we need to write some html and use `v-for` to loop through our `todos` :

```
# app.vue
<div id="app">
...
    <ol>
      <li v-for="todo in todos">
        <b>Title:</b> {{ todo.title }}
        <b>Content:</b> {{ todo.content }}
      </li>
    </ol>
</div>
```

Start up both servers and give it a test run!

```
rails s
./bin/webpack-dev-server
```

Now that we have our Vuejs form up and running, I invite you to challenge yourselves and do more. This was an introductory to getting a Vuejs form up and running, but it could be improved. Think about using `ActionCable` to publish the `Todo` 's to the view. Also write some tests. I purposely left those out ;).

If you got lost or something isn't working, check out the repo: https://github.com/jd-gray/vue-js-rails (https://github.com/jd-gray/vue-js-rails)