How To Build A Simple Single Page Application Using Vue 2 (Part 2)

Ogundipe Samuel Ayo
Software Developer

Dec 21, 2017
Also Discussed in SPAs
14 COMMENTS
15,210 VIEWS

Top Courses

Part2 of this tutorial dealt with the basics of building a single page application, with page linking automatically done by Vue. In this tutorial you will learn how to:

- pass parameters alongside the routes, and
- use route guards to protect routes and block off non-authenticated users from visiting the site.

Starting off from the end of the last tutorial, head over to the GitHub repository to clone and continue with the application.

What will you build?

You are going to build on the existing project done in Part 1, add navigation guards and a route to pass parameters.

Table of Contents

1. What will you build?
2. Passing parameters to routes
3. Using route guards
4. Conclusion

Here is a pictorial view of what we will be building.

Passing parameters to routes

In Vue, parameters can be passed to a route by simply putting the /:id attribute before the parameter name (:id). Whatever content the parameter holds will be referred to as the "id", which can be accessed using $route.params.id in the Home component, which was declared as its component. Let's begin.

Creating the parameter route

Create a new component in the src/components folder called Parameter.vue , and copy the following contents into it:

```
( path: '/name/:id', component: Home )
```

From the example, it can be said that the route will give the result of /name/3 , or any integer in place of :id. Whatever content the parameter holds will be referred to as the "id", which can be accessed using $route.params.id in the Home component, which was declared as its component. Let's begin.

```
<!-- src/components/Parameter.vue
export default {
  name: 'Parameter',
  data () {
    return {
      msgParam: ''
    }
  },
  created() {
    this.msgParam = this.$route.params.id;
  }
}
</script>
```

In the above piece of code, a param component was defined, which is the template that holds a text box and a button.

Now, go to the main.js file, import the Parameter component, as well as create a path that handles the route.

> Related Course: Build an Online Shop with Vue

Just above the Router.Router block of code, add the following import:

```
import Parameter from './components/Parameter'
```

After doing the above and importing the param component, replace the routes block with this:

```
routes: [
  { path: '/', component: Home },
  { path: '/parameter/:id', component: Parameter },
]
```

Notice that in the above piece of code, the only difference that occurred was the addition of the param route. However, there is more step to go so that the route can be shown with the about and home links.

Use the router link tag in our App.vue file, and add the following line of code, just after the "about" link:

```
<!-- src/components/App.vue -->
<router-link v-bind:to="'/parameter/'+msgParam">Go to /parameter/:id</router-link>
```

Take a look at the homepage, it should look like this:

Notice that the param link is just after the "about" link. Click on the param link, and you should see the following:

Open up the developer's tool in the console. If you click on the button, you will likely encounter this error:

```
[vue-router] Route with name 'parameter/:id' does not exist
```

This is because, in the code above, the method attached to the button triggers this:

```
this.$route.push({ name: 'Parameter/:id', params:{ id: this.msgParam }});
```

which has not defined any route called parameter/:id that accepts parameters.

The next step is to create the component that handles the route. Create a new file in src/components called parameter/id.vue , and paste the following contents into it.

```
<!-- src/components/parameter/id.vue -->
  The parameter value that was passed to id is {{ $route.params.id }}

  export default {
  name: 'parameter/id'
  }
</script>
```

This creates a component that holds a span element, which also prints out the parameter that was passed to it. For this to work, add this new component to the routes.

Just above the Router.Router block of code, add the following import:

```
import parameter/id from './components/parameter/id'
```

After doing the above, and importing the parameter/id component, replace the routes block with this:

```
routes: [
  { path: '/', component: Home },
  { path: '/parameter/:id', component: parameter/id },
]
```

Notice that the last route has two new adjustments, which was not visible in other routes. The first adjustment is the /:id , which is the placeholder for the routes you are passing in. The second adjustment is the name object in the route object.

To understand the name property in the object, know that just like any other router, the Vue router contains the concept of naming routes. Instead of remembering the full URL of the new route, the shorter way of calling routes is by naming the routes. Also, take a look back at the $route.push method in the param component, you will see that the route object is being pushed, which identifies the route by name.

Now, click on the param link, and type in 52 in the text box, for example. Then click the go to route button, and you should see a page like this:

That is it! You have just passed in your first route to the component. Super easy, isn't it?

Using route guards

A route guard is a section that watches for changes in the routes and performs some actions, either before the route is transitioned into the next one, or after the route has been transitioned. Route guards can be used to protect routes against non-authenticated users from visiting the site. Vue router has global guards, per-route guards, as well as component guards.

There are two major global routes produced by Vue-router namely: beforeEach and afterEach guards. As the name implies, the "beforeEach" is performed before a new route, while the "afterEach" is performed after the "beforeEach" guard.

Per-routes guards can be added to the route object of the particular route to be guarded as a callback, while the "in-component" guards are defined in the components that handle those routes. For the purpose of this tutorial, stick to the "beforeEach" global route.

Open the main.js file, and immediately after the routes constant, add the following:

```
//main.js
//global route guards
router.beforeEach(to, from, next) {
  //check if the path user is going to is our parent path
  if(to.path == "/parameter/:id"){
    //check if the user is in a secure area
    if(localStorage.getItem("user")==null){
      //prompt user
      var name = prompt("please enter your username");
      //prompt for password
      var pass = prompt("please enter your password");
      //check if the user is a valid user, otherwise prompt again
      if (name == "samuel" && pass == "password"){
        //set the flag
        localStorage.setItem("user", name);
        //next to the route
        next()
      }else{
        //if the username and pass is wrong
        alert("Wrong username and password, you do not have permission to access that route")
        //redirect, do not move to the route
        return;
      }
    }
    else{
      next()
    }
  }
  else{
    next()
  }
}
);
```

The "router.beforeEach" function was called and it takes three arguments that are passed to it by the Vue router. The to parameter refers to the object of the path to be accessed, while the from parameter refers to the object of the previous path.

Check if the "to.path" is /param , which is the path for the route to be protected. If the path corresponds to the one to be protected, check the "localStorage" data holding the user details to see if it is defined. If the "localStorage" data is not defined, prompt for the username and password, and create the "localStorage" data. If it is defined, the function continues and the route is visible to the user.

Note: The next function must be called so that the router can process the user to the next page. If the function isn't called, a blank screen could come up. In this tutorial, a new string of username and password was used for the authentication details. However, a lot more can be done, such as requesting user authentication via a server.

Conclusion

You should now be able to pass parameters to routes, know what named routes are, and how to create them. Also, the types of route guards, as well as an implementation for the "beforeEach" route guard was explained.

Now, you should be able to create a single page application, register routes, pass in parameters to the routes, as well as guarding the routes. On these premises, you can begin to build awesome single page applications.

Stack Abuse
Learn JavaScript
Learn Node
Learn Vue.js
Latest Courses
LEARN JQUERY FOR Total Noobs
Beginner Friendly Step-by-Step Guide
Learn jQuery for Total Noobs Volume 2: Ajax & Practical Course
Learn jQuery for Total Noobs Volume 1: Jump Start Course
Learn jQuery for Total Noobs Volume 3: JavaScript Crash Course
Build Your First Angular Website
Getting Started with Scrum Bear: Sequence Diagram Basics
View Courses
Latest Posts
Sponsored
How to get to 1 million users in 1 year
The Total Guide to Free SSL Certificates for Local Sites
Build a Reddit Training App with React, GraphQL and User Authentication
Code Challenge #3: Handmade Functional Programming
Customize Payment Screens with Enhanced API, Sample Code and SDKs
Join Scotch's New Community Forum on Spectrum.chat
Show More
Hi, I'm Samuel!
I'm a Software Developer, Technical Author, Speaker, and Open Source Contributor. I'm passionate about JavaScript, Node.js, Vue.js, Angular, Ruby on Rails, Python, Flask, Go, Kubernetes, Docker, Linux, Cloud Native Technologies, Scaling Microservices, and Machine Learning.
Follow @Samuel_Ayo
Follow Samuel
Like
0
0
Related Courses
Build an Online Shop with Vue