If your Vue.js 2 applications grows bigger and consists of multiple views routing becomes a core part of your project. Let's explore how you can use the Vue core package vue-router to implement routing in your web application.

Setting Up Our Vue.js 2 Project

We're using Vue CLI to set up our Vue.js project. If you haven't installed Vue CLI on your system yet you need to complete the installation first by using the following command:

```
$ npm install -g vue-cli
```

Having executed the installation successfully a new command `vue` becomes available. You can use this command the initiate a new Vue project in the following way:

```
$ vue init webpack vue-router-01
```

You're being asked a few questions on the command line to complete the creation of the project. One question asks if you want to add vue-router to the project. Answer with Y (yes) to make sure the the Vue router package is added to the initial project setup.

Next you need to change into the newly created project folder:

```
$ cd vue-router-01
```

and complete the installation of dependencies by executing the following command:

```
$ npm install
```

Now the development web server can be started by entering:

```
$ npm run dev
```

Installing vue-router Package

If you're configuring your project with Vue CLI like shown before the *vue-router* package is added to your project automatically.

If you would like to add the *vue-router* package manually to your project later, you can do so by using the *npm* command in the following way:

```
$ npm install vue-router —save
```

Add vue-router To Our Application

If you've followed the recent steps and added vue-router to the initial configuration by using Vue CLI, routing is activated by default. The activation is taking place in file *src/router/index.js*:

```
import Vue from 'vue'
import Router from 'vue-router'
import Hello from '@/components/Hello'

Vue.use(Router)

export default new Router({
  routes: [
    {
      path: '/',
      name: 'Hello',
      component: Hello
    }
  ]
})
```

First `Router` is imported from the *vue-router* package via import statement. Next you need to call

`Vue.use(Router)` to make sure that Router is added as a middleware to our Vue project.

Setting Up Routes

A default route configuration is already included in file src/router/index.js. The route configuration is a JavaScript object consisting of three properties *path*, *name* and *component*. The configuration object is added to an array. The array needs to be assigned to the *routes* property of the object which is passed to the constructor of Router.

As you can see in the listing above the Router instance is created and exported. To activate the Router configuration the Router instance is imported in file src/main.js:

```
import router from './router'
```

The Router instance is then used to create Vue application instance, as you can see in the following:

```
new Vue({
  el: '#app',
  router,
  template: '<App/>',
  components: { App }
})
```

Standard Routes

Standard routes are easy to define. The default route configuration is a typical example of a standard route:

```
  {
    path: '/',
    name: 'Hello',
    component: Hello
  }
```

The route path is defined by using the path property. Setting the value of that property to '/' means that this is the default route of your application. If the application is running on localhost:8080 for example the default route is activated when the user accesses http://localhost:8080 in the browser.

Routes With Parameters

Let's add a second route configuration object to the array:

```
export default new Router({
  routes: [
    {
      path: '/',
      name: 'Hello',
      component: Hello
    },
    {
      path: '/firstroute/:name',
      name: 'FirstRoute',
      component: FirstRoute
    }
  ]
})
```

For the new route configuration the path value *'/firstroute/:name'* is used. The last part of the path string is defining a routing parameter: *name*. The routing parameter enables us to pass a value to the routes component via URL, e.g.:

http://localhost:8080/firstroute/bob

In this case the value of the *name* routing parameter is bob. The parameter can be accessed in the route component *FirstRoute*. Let's create a new file *src/components/FirstRoute.vue* and insert the code:

```
<template>
  <div>
    <h1>{{ msg }}</h1>
     Hello {{ $route.params.name }}
  </div>
</template>

<script>
 export default {
   name: 'firstroute',
   data () {
     return {
     msg: 'FirstRoute'
   }
 }
}
</script>
```

Embedding Routes Output

If a certain route is accessed the assigned Vue component is called and rendering it's HTML output based on the component's template code. The HTML output of the route components needs to be part of the browser output. The following placeholder element is

used to define the place in the HTML page where the route component output should be inserted:

```
<router-view></router-view>
```

In our Vue application this element is already placed in the template code of App component (in file src/App.vue), as you can see in the following:

```
<template>
  <div id="app">
    <img src="./assets/logo.png">
    <router-view></router-view>
  </div>
</template>
```

If you now try to access FirstRoute in the browser you should see the output of component *FirstRoute*.

Child Routes

It's also possible to define child routes. Let's take a look at the following extended router configuration:

```
export default new Router({
  routes: [
```

```
    {
      path: '/',
      name: 'Hello',
      component: Hello
    },
    {
      path: '/firstroute/:name',
      name: 'FirstRoute',
      component: FirstRoute,
      children: [
        {
          path: 'child',
          component: FirstRouteChild
        }
      ]
    }
  ]
})
```

The property *children* has been added to the configuration object of *FirstRoute*. The property gets assigned an array with is containing another routes configuration object for route *FirstRouteChild*. Let's implement *FirstRouteChild* as well. Create a new file *FirstRouteChild.vue* in folder *src/components* and insert the following code:

```
<template>
  <div>
    <h1>{{ msg }}</h1>
  </div>
</template>

<script>
export default {
  name: 'firstroutechild',
  data () {
```

```
    return {
      msg: 'FirstRouteChild'
    }
  }
}
</script>
```

Linking To Routes

The user is able to access a route by entering the corresponding URL in the browser directly. If you would like to add links to route URLs you can make use of the <router-link> element to generate a HTML link automatically:

```
<router-link to="/route-one/route-one-child">Link to route one, child one</router-link>
```

The *to* attribute contains the path to the route. You can also use the a colon before the to attribute to be able to assign a dynamic expression:

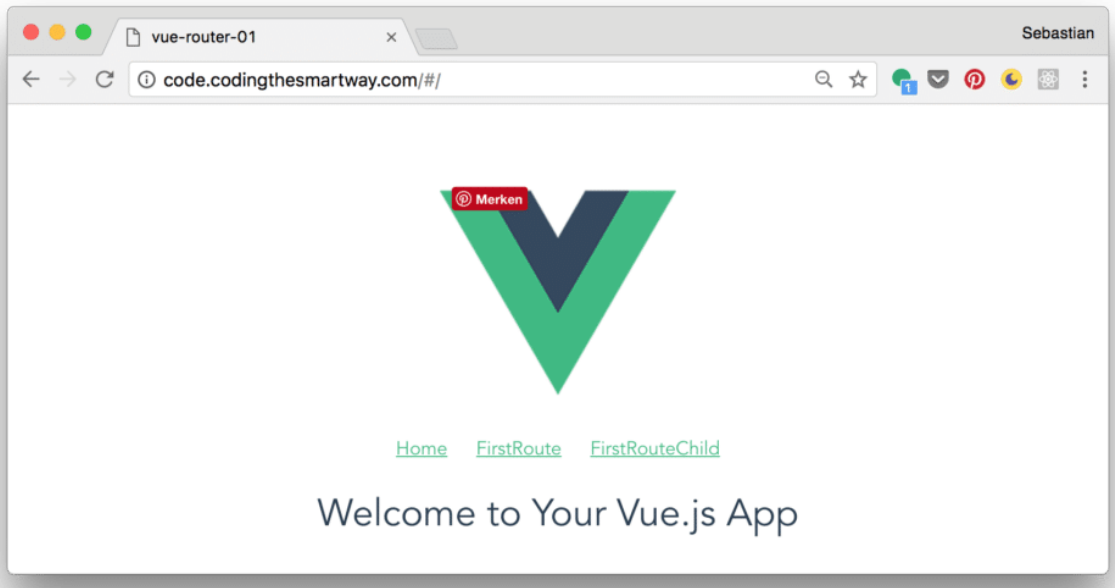
```
<router-link :to="myRouteTargetString">Link to dynamic route</router-link>
```

To link to the three routes which are available in our application we're changing the implementation of App component in file *App.vue*:

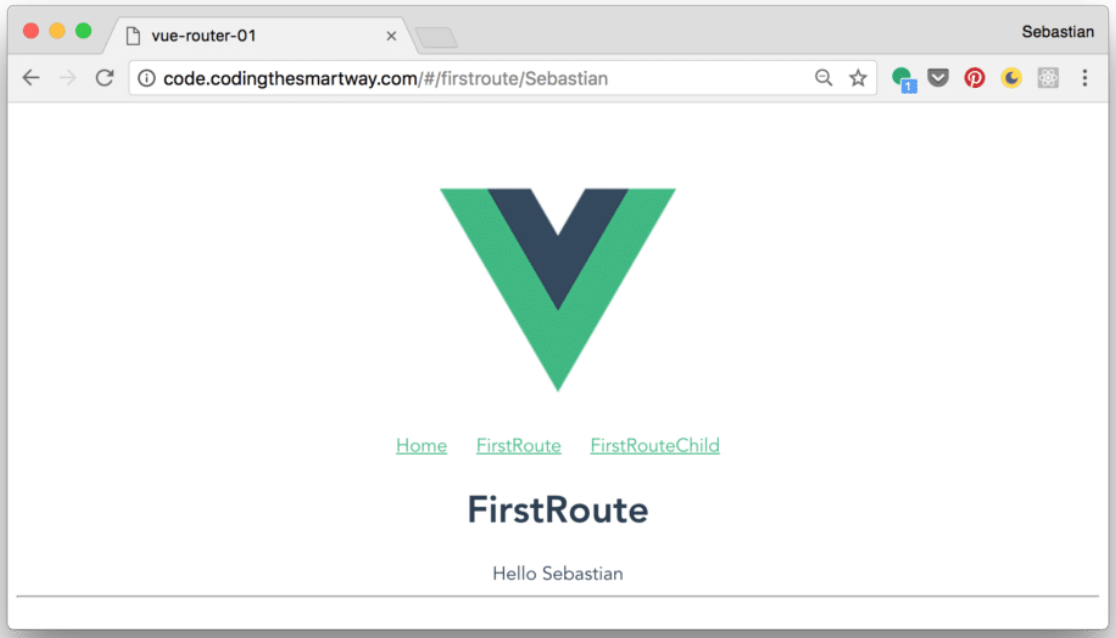
```
<template>
  <div id="app">
    <img src="./assets/logo.png">
    <div>
      <ul>
        <li><router-link to="/">Home</router-link></li>
        <li><router-link to="/firstroute/Sebastian">FirstRoute</router-link></li>
        <li><router-link
to="/firstroute/Sebastian/child">FirstRouteChild</router-link></li>
      </ul>
    </div>
    <router-view></router-view>
  </div>
</template>

<script>
export default {
  name: 'app'
}
</script>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}

ul {
  list-style-type: none;
  padding: 0;
}

li {
  display: inline-block;
```

```
  margin: 0 10px;
}

a {
  color: #42b983;
}
</style>
```

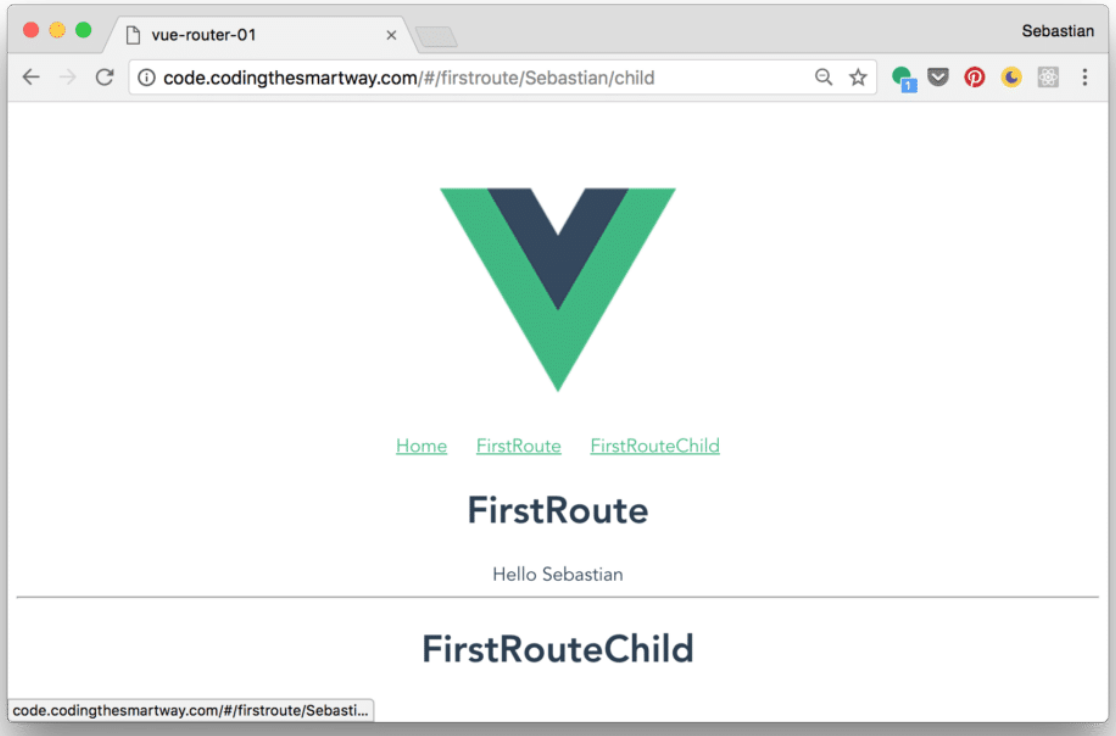
Now the final result should look like the following:



By clicking on the link *FirstRoute* the view changes to include the output of *FirstRoute*component without reloading the page:



A click on link *FirstRouteChild* changes the view to also contain the output of component *FirstRouteChild* in addition to the output of *FirstRoute* component:


vue-router-01
Sebastian
code.codingthesmartway.com/#/firstroute/Sebastian/child
Home
FirstRoute
FirstRouteChild
FirstRoute
Hello Sebastian
FirstRouteChild
code.codingthesmartway.com/#/firstroute/Sebasti...