# Vue.js

## 1. Instalasi Vue-cli

Pada pertemuan kali ini, kita akan membahas pembuatan tampilan front-end website menggunakan framework yang bernama Vue.js[[1]](#footnote-1).

Graphical user interface, website

Description automatically generated

Pertama-tama, install CLI untuk Vue dengan menggunakan npm.

npm install -g vue-cli

CLI ini akan membantu dan mempermudah kita melakukan setup Vue.js. Setelah selesai melakukan instalasi, jalankan perintah berikut ini untuk membuat project baru dengan nama demo\_example.

vue init webpack demo\_example

Pada saat menjalankan perintah di atas, Vue-cli akan menanyakan beberapa informasi sebagai berikut:

* Project name
* Project description
* Author
* Vue build: pilih runtime + compiler
* Vue-router yang akan dibahas lebih lanjut nantinya. Untuk saat ini, pilih Yes.
* ESLint yang merupakan tool untuk memastikan kode program yang anda buat mengikuti style tertentu dan menghindari error. Pilih Yes, kemudian pilih Standard.
* Setup test: pilih No.
* Dependency management: pilih NPM.

Setelah perintah init di atas dilakukan, jalankan perintah di bawah ini untuk melihat tampilan template Vue.js yang sudah dibuat dengan membuka localhost:8081 (sesuaikan port dengan informasi yang muncul pada layar terminal anda) pada browser.

cd demo\_example

npm run dev

Graphical user interface, text

Description automatically generated

Vue-cli akan membuat struktur direktori project anda secara otomatis. Kita akan melihat dari build/webpack.base.conf.js seperti di bawah ini,

module.exports = {

context: path.resolve(\_\_dirname, '../'),

entry: {

app: './src/main.js'

},

output: {

terlihat bahwa program anda dimulai dari file src/main.js. Isi dari file tersebut adalah seperti di bawah ini:

// The Vue build version to load with the `import` command

// (runtime-only or standalone) has been set in webpack.base.conf with an alias.

import Vue from 'vue'

import App from './App'

import router from './router'

Vue.config.productionTip = false

/\* eslint-disable no-new \*/

new Vue({

el: '#app',

router,

components: { App },

template: '<App/>'

})

Perhatikan bahwa template utama dari aplikasi ini dispesifikasikan pada App.vue yang terletak pada folder src ini pula. Jika App.vue dibuka, isinya adalah sebagai berikut:

<template>

<div id="app">

<img src="./assets/logo.png">

<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;

}

</style>

File di atas merupakan template utama dari aplikasi ini yang menampilkan logo Vue dan kemudian menampilkan tampilan sesuai dengan route yang ada.

Jika dilihat pada file src/router/index.js seperti di bawah ini, dapat diketahui bahwa jika kita membuka localhost:8081, maka komponen yang ditampilkan pada browser (dengan layout utama App.vue) adalah komponen HelloWorld yang berada pada direktori src/components/HelloWorld.vue.

import Vue from 'vue'

import Router from 'vue-router'

import HelloWorld from '@/components/HelloWorld'

Vue.use(Router)

export default new Router({

routes: [

{

path: '/',

name: 'HelloWorld',

component: HelloWorld

}

]

})

Bukalah file src/components/HelloWorld.vue dan hapus isinya. Kita akan mengganti isi file tersebut sebagai berikut:

<template>

<div>

<h1>Hello World</h1>

</div>

</template>

Kemudian, jika kita kembali ke browser dan membuka localhost:8081, maka akan didapat tampilan sebagai berikut:

A screenshot of a computer

Description automatically generated

sesuai dengan apa yang kita buat pada HelloWorld.vue.

## 2. Membuat Tampilan Login Menggunakan Buefy

Pada bagian ini, kita akan membuat tampilan login dengan menggunakan bantuan pustaka bernama Buefy.

npm install –save buefy

Untuk menggunakan buefy pada aplikasi anda, tambahkan potongan kode berikut ini pada src/main.js.

import Buefy from 'buefy'

import 'buefy/dist/buefy.css'

Vue.use(Buefy)

Kemudian, bukalah file src/router/index.js dan tambahkan route untuk login sebagai berikut:

import Vue from 'vue'

import Router from 'vue-router'

import HelloWorld from '@/components/HelloWorld'

import Login from '@/components/Login'

Vue.use(Router)

export default new Router({

routes: [

{

path: '/',

name: 'HelloWorld',

component: HelloWorld

},

{

path: '/login',

name: 'Login',

component: Login

}

]

})

Kemudian, buatlah komponen login pada direktori src/components/Login.vue. Misalkan untuk menguji apakah berhasil, tambahkan kode sederhana seperti di bawah ini pada Login.vue.

<template>

<div>

<h1>Login</h1>

</div>

</template>

Lihatlah hasilnya melalui browser localhost:8081/#/login.

A screenshot of a computer

Description automatically generated

Lanjutkan kode program untuk login seperti di bawah ini:

<template>

<section class="section">

<div class="container is-max-desktop">

<h1 class="title">Login</h1>

<hr>

<b-field label="Username">

<b-input maxlength="30"></b-input>

</b-field>

<b-field label="Password">

<b-input type="password" maxlength="30"></b-input>

</b-field>

<b-button class="is-primary" expanded>Login</b-button>

</div>

</section>

</template>

## 3. Membuat Tampilan CRUD untuk User

Selanjutnya kita akan membuat contoh tampilan lainnya, misalnya untuk CRUD (Create, Read, Update, Delete) users.

Pertama-tama, kita akan menghapus logo Vue pada src/App.vue.

<template>

<div id="app">

<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;

}

</style>

Kemudian tambahkan file-file berikut untuk tampilan User:

* src/components/User.vue
* src/components/UserAdd.vue
* src/components/UserDetail.vue

Tambahkan juga router untuk membuka halaman User tersebut pada src/router/index.js.

import Vue from 'vue'

import Router from 'vue-router'

import HelloWorld from '@/components/HelloWorld'

import Login from '@/components/Login'

import User from '@/components/User'

import UserAdd from '@/components/UserAdd'

import UserDetail from '@/components/UserDetail'

Vue.use(Router)

export default new Router({

routes: [

{

path: '/',

name: 'HelloWorld',

component: HelloWorld

},

{

path: '/login',

name: 'Login',

component: Login

},

{

path: '/user',

name: 'User',

component: User

},

{

path: '/user-add',

name: 'UserAdd',

component: UserAdd

},

{

path: '/user-detail',

name: 'UserDetail',

component: UserDetail

}

]

})

Kemudian ketikkan kode program di bawah ini pada src/components/User.vue.

<template>

<section class="section">

<div class="container is-max-desktop">

<h1 class="title">Users</h1>

<hr>

<div class="buttons">

<b-button class="is-primary" tag="router-link" to="/user-add">Add New User</b-button>

<b-button class="is-secondary" tag="router-link" to="/user-detail">Edit Dummy User</b-button>

<b-button class="is-danger" @click="trashModal">Delete Dummy User</b-button>

</div>

<b-table :data="users" :columns="columns" />

</div>

</section>

</template>

<script>

export default {

data () {

return {

users: [

{

'no': 1,

'username': 'johndoe',

'email': 'john.doe@gmail.com',

'fullname': 'John Doe'

},

{

'no': 2,

'username': 'janedoe',

'email': 'jane.doe@gmail.com',

'fullname': 'Jane Doe'

},

{

'no': 3,

'username': 'jansonh',

'email': 'jansonh@fti.untar.ac.id',

'fullname': 'Janson Hendryli'

}

],

columns: [

{

field: 'no',

label: 'No',

width: '40',

numeric: true,

centered: true

},

{

field: 'username',

label: 'Username',

centered: true

},

{

field: 'email',

label: 'Email Address',

centered: true

},

{

field: 'fullname',

label: 'Full Name',

centered: true

}

]

}

},

methods: {

trashModal () {

this.$buefy.dialog.confirm({

title: 'Delete Confirmation',

message: 'Are you sure want to delete <b>Dummy User</b>?',

confirmText: 'Yes',

cancelText: 'Cancel',

type: 'is-danger',

onConfirm: () => { }

})

}

}

}

</script>

Graphical user interface

Description automatically generated

Tambahkan potongan kode di bawah ini pada src/components/UserAdd.vue.

<template>

<section class="section">

<div class="container is-max-desktop">

<h1 class="title">Add New User</h1>

<hr>

<section>

<b-field label="Full Name">

<b-input />

</b-field>

<b-field label="Email">

<b-input type="email" maxlength="255" />

</b-field>

<b-field label="Username">

<b-input maxlength="30" />

</b-field>

<b-field label="Password">

<b-input type="password" maxlength="30" />

</b-field>

<b-button type="is-success" @click="saveUser" expanded>Save</b-button>

<br>

<router-link to="/user">Cancel</router-link>

</section>

</div>

</section>

</template>

<script>

export default {

methods: {

saveUser () {

this.$buefy.snackbar.open({

message: 'Success!',

type: 'is-success',

position: 'is-top'

})

this.$router.push('/user')

}

}

}

</script>

A screenshot of a computer

Description automatically generated

Pada blok <script>, kita dapat membuat berbagai methods yang dapat dipanggil misalnya pada saat tombol save ditekan. Pada contoh di atas, method saveUser dipanggil pada saat tombol save ditekan dan menampilkan snackbar dari Buefy, kemudian redirect ke halaman table User.

Pada UserDetail.vue, kita akan mengisi form user di atas dengan nilai-nilai yang didapat dari database. Akan tetapi, karena tampilan yang dibuat belum terhubung dengan sistem basis data, maka kita akan menggunakan data dummy terlebih dahulu.

Sebelum mengisi UserDetail.vue, ubahlah kode pada User.vue pada tombol Edit Dummy User sehingga menjadi sebagai berikut:

<b-button class="is-secondary" tag="router-link" :to="{ name: 'UserDetail', params: { id: 123 }}">Edit Dummy User</b-button>

Pada potongan kode di atas, pada saat tampilan berpindah ke UserDetail.vue, ada parameter id yang dilempar juga, yaitu nantinya merupakan ID dari user tersebut yang ada di database.

Nilai parameter ini kemudian dapat diakses dari UserDetail.vue dengan mengakses variabel this.$route.params.

Ketikkan kode UserDetail.vue di bawah ini.

<template>

<section class="section">

<div class="container is-max-desktop">

<h1 class="title">Edit User</h1>

<hr>

<section>

<b-field label="Full Name">

<b-input v-model="user.fullname" />

</b-field>

<b-field label="Email">

<b-input v-model="user.email" type="email" maxlength="255" />

</b-field>

<b-field label="Username">

<b-input v-model="user.username" maxlength="30" />

</b-field>

<b-field label="Password">

<b-input v-model="user.password" type="password" maxlength="30" />

</b-field>

<b-button type="is-success" @click="saveUser" expanded>Save</b-button>

<br>

<router-link to="/user">Cancel</router-link>

</section>

</div>

</section>

</template>

<script>

export default {

data () {

return {

user: {

fullname: '',

email: '',

username: '',

password: ''

}

}

},

mounted () {

if (this.$route.params.id === 123) {

this.user.fullname = 'Tester'

this.user.email = 'test@mail.net'

this.user.username = 'testtest'

}

},

methods: {

saveUser () {

this.$buefy.snackbar.open({

message: 'Success!',

type: 'is-success',

position: 'is-top'

})

this.$router.push('/user')

}

}

}

</script>

Kode yang berada di dalam mounted () { ... } akan dipanggil pada saat halaman UserDetail ini ter-load. Pada contoh ini kita hanya melihat jika params.id yang dilempar dari User.vue adalah 123, maka kita isi nilai awal dari user.

Dengan menghubungkan antara <b-input> dan variabel pada data() menggunakan properti v-model, maka nilai dari input field tersebut menyesuaikan nilai dari variabel user.

1. https://vuejs.org [↑](#footnote-ref-1)