

COS30043

Interface Design and Development



Lecture 9 – Single Page Application

2021 – Semester 1



1

Contents

- Using Router for Menu in SPA
- Using Vuetify tabs in SPA
- Using Pagination in SPA



2

USING ROUTER FOR MENU IN SPA

3

Single-page Application (SPA)

- a web application that is contained in a single web page to provide user experience similar to desktop applications
- all code, HTML, CSS and JavaScript, is retrieved on initial page load, and appropriate resources are dynamically loaded as necessary based on response to user actions
- the entire page does not reload nor control transfer to another page
- interaction involves dynamic communication with the web server in the background



4

SPA - Route

- **Creating VueRouter**

```
const router = new VueRouter({  
  routes: [  
    {  
      // login route  
      path: '/login',  
      component: Login,  
      name: "login"  
    },
```

5 - Interface Design and Development, © Swinburne



5

SPA - Route (Continued)

```
    // logout route  
    {  
      path: '/logout',  
      name: "logout"  
    },  
    // dashboard  
    {  
      path: '/dashboard',  
      component: Dashboard,  
      name: 'dashboard'  
    } ] })
```

6 - Interface Design and Development, © Swinburne



6

SPA – Menu

```
Vue.component('app-nav', {  
  
  template: `  
    <div>  
      <v-spacer></v-spacer>  
      <v-btn>  
        <router-link to="/login"  
          v-on:click.native="logout()" replace>  
          Logout<v-icon>mdi-logout</v-icon>  
        </router-link>  
      </v-btn>  
    </div>  
  `
```

7 - Interface Design and Development, © Swinburne



7

SPA – Menu (Continued)

```
,  
  
  methods: {  
    logout() {  
      this.$root.logout();  
    }  
  }  
});
```

8 - Interface Design and Development, © Swinburne



8

USING VUETIFY TABS IN SPA

9

Vuetify Tab Example

Data

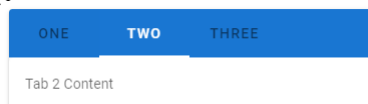
```
data () { return { tab: null,
  items: [
    { tab: 'One', content: 'Tab 1 Content' },
    { tab: 'Two', content: 'Tab 2 Content' }
  ]
  ...
}
```

Template

```
<v-card>
  <v-tabs v-model="tab" background-color="primary" >
    <v-tab v-for="item in items" :key="item.tab" >
      {{ item.tab }}
    </v-tab>
  </v-tabs>

  <v-tabs-items v-model="tab">
    <v-tab-item v-for="item in items2" :key="item.tab" >
      <v-card >
        <v-card-text>{{ item.content }}</v-card-text>
      </v-card>
    </v-tab-item>
  </v-tabs-items>
</v-card>
```

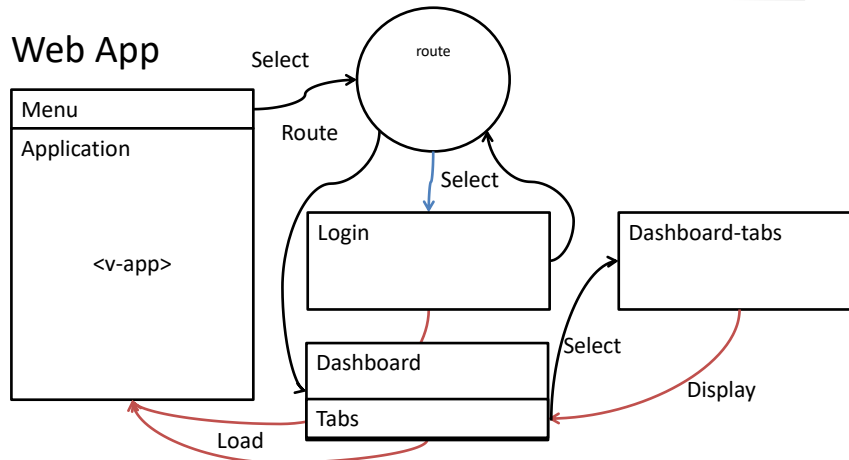
Using a shared v-model, the v-tabs-items will sync with the currently selected v-tab.



10

Single-page Application (SPA)

- Web App



- One route and one target view

11 - Interface Design and Development, © Swinburne



11

Login form

- **Login form**

```
<v-form ref="form" v-model="valid"
  lazy-validation>
```

```
<v-text-field v-model="username"
  :counter="10":rules="usernameRules"
  label="Username" required >
</v-text-field>
```

12 - Interface Design and Development, © Swinburne



12

Log in form (Continued)

```
<v-text-field v-model="password"
  label="Password"  type="password"
  required >
</v-text-field>
<v-btn :disabled="!valid"
  color="success" class="mr-4"
  @click="validate">
  Login
</v-btn>
</v-form>
```

13 - Interface Design and Development, © Swinburne



13

SPA – Login component

```
// Creating Login component
const Login = {
  // defining variables to be used in the component
  data() {
    return {
      input: {
        username: "",
        password: ""
      },
      valid: false,
    }
  }
}
```

N.B. "... " indicates lines of code which are not shown.

14 - Interface Design and Development, © Swinburne



14

SPA – Login component (Continued)

```
methods: {
  login() {
    if (this.$refs.form.validate()) {
      ...

      const requestOptions = {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({
          username: this.input.username,
          password: this.input.password
        })
      };
    }
  }
}
```



15 - Interface Design and Development, © Swinburne

15

SPA – Login component (Continued)

```
// Login check
fetch("resources/api_user.php/", requestOptions)
.then( response =>{
  //turning the response into the usable data
  return response.json( );
})
.then( data =>{
  //This is the data you wanted to get from url
  if (data == null) { // didn't find this username password pair
    self.msg="username or password incorrect.";
  }
  else{
    this.$emit("authenticated", true);
    // $emit() function allows you to pass
    // custom events up the component tree.
    this.$router.replace({ name: "dashboard" });
  }
})
.catch(error => {
  self.msg = "Error: "+error;
});
...
N.B. "..." indicates lines of code which are not shown here
```



16

SPA – Login component (Continued)

```
template: `  
...  
<v-form ref="form" v-model="valid" lazy-validation>  
  <v-text-field v-model="input.username"  
    :counter="10" :rules="usernameRules"  
    label="Username" required>  
</v-text-field>  
  <v-text-field v-model="input.password"  
    label="Password" type="password"  
    :rules="passwordRules" required>  
</v-text-field>  
  <v-btn :disabled="!valid" color="success"  
    class="mr-4" @click="login()"> Login  
</v-btn>  
  <v-btn color="error" class="mr-4" @click="reset">  
    Reset  
</v-btn>  
</v-form>
```

17

USING PAGINATION IN SPA

SWIN
BUR
NE

SWINBURNE
UNIVERSITY OF
TECHNOLOGY

KNOW
ING

18

Pagination using Vue JS Paginate

- Link *vuejs-paginate.js* file in the **HTML**

```
<script src="js/vuejs-paginate.js"></script>
```

- Initialise *VuejsPaginate* component

```
Vue.component('paginate', VuejsPaginate);
```

N.B. "..." indicates lines of code which are not shown here



19 - Interface Design and Development, © Swinburne

19

Pagination in SPA

```
Vue.component('app-readmysql', {
  template: `
    ...
    <v-card-text>
      <ul class="list-group">
        <li class="list-group-item"
          v-for="person in getItem"
          :key="person.name" >
          <img v-bind:src='person.fpath'
            class="img-rounded"
            alt="smiley" height="30px" />
          {{person.name}} {{person.age}}
        </li>
      </ul>
    </v-card-text>
    ...
  `
})
```

N.B. "..." indicates lines of code which are not shown here



20 - Interface Design and Development, © Swinburne

20

Pagination in SPA (Continued)

```
...
<paginate
  :page-count="getPageCount"
  :page-range="5"
  :margin-pages= "1"
  :click-handler="clickCallback"
  :prev-text=" 'Prev' "
  :next-text="'Next'"
  :container-class="'pagination'"
  :page-class="'page-item'">
</paginate>
```

...

N.B. "..." indicates lines of code which are not shown here



21 - Interface Design and Development, © Swinburne

21

Pagination in SPA (Continued)

// Defining functions in *method* and *computed*

```
computed:{
  getItems: function() {
    let current = this.currentPage * this.perPage;
    let start = current - this.perPage;
    return this.persons.slice(start, current);
  },
  getPageCount: function() {
    return Math.ceil(this.persons.length /
this.perPage);
  }
},
methods:{
  clickCallback: function(pageNum){
    this.currentPage = Number(pageNum);
  }
},
}
```

22


Data Insert

```
...
<v-form>
  <v-text-field label="Name" v-model="name1" /></v-text-field>
  <v-text-field label="Age" v-model="age1" /></v-text-field>

  <v-radio-group label="Smiley Color" v-model="imgVar">
    <v-radio label="White" value="1"></v-radio>
    <v-radio label="Yellow" value="2"></v-radio>
  </v-radio-group>

  <v-btn depressed
    v-on:click="postData(name1,age1, imgVar)"
    color="primary">
    Add
  </v-btn>

</v-form>
...
N.B. "... indicates the lines of code which are not shown here
```




23 - Interface Design and Development, © Swinburne

23

Data Insert (Continued)

```
methods: {
  postData: function(nm, age, img) {
    //define url for api
    var postSQLApiURL = 'resources/apis.php/'

    var self = this;
    // POST request using fetch with error handling
    const requestOptions = {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify({
        name: nm,
        age: age,
        fpath: 'img/smiley' + img + '.png'
      })
    };
  };
};
```



24 - Interface Design and Development, © Swinburne

24

Data Insert (Continued)

```
fetch(postSQLApiURL, requestOptions)
.then( response =>{
    //turning the response into the usable data
    return response.json( );
})
.then( data =>{
    //This is the data you wanted to get from url
    self.msg = "Data Inserted Successfully." ;
})
.catch(error => {
    self.msg = 'There was an error!' + error;
});
```



25 - Interface Design and Development, © Swinburne

25

Data Update

```
<v-form name="myForm2" class="form-horizontal">

    <v-text-field label="Name" v-model="name2" />
    </v-text-field>

    <v-text-field label="Age" v-model="age2" />
    </v-text-field>

    <v-btn depressed v-on:click="putData(name2,age2)"
        color="primary">
        Update
    </v-btn>
```

```
</v-form>
```

...

N.B. "..." indicates the lines of code which are not shown here

26 - Interface Design and Development, © Swinburne



26

Data Update (Continued)

```
methods: {  
  putData: function(nm, age) {  
    //define url for api  
    var putSQLApiURL = 'resources/apis.php/name/'+nm;  
    var self = this;  
    // POST request using fetch with error handling  
    const requestOptions = {  
      method: 'PUT',  
      headers: {  
        'Content-Type': 'application/json'  
      },  
      body: JSON.stringify({  
        name: nm,  
        age: age  
      })  
    };  
  };
```

27 - Interface Design and Development, © Swinburne



27

Data Update (Continued)

```
fetch(putSQLApiURL, requestOptions)  
  .then( response =>{  
    //turning the response into the usable data  
    return response.json( );  
  })  
  .then( data =>{  
    //This is the data you wanted to get from url  
    self.msg="successful";  
  })  
  .catch(error => {  
    self.err=error  
  });
```

28 - Interface Design and Development, © Swinburne



28

Data Delete

```
...
<v-form>

<v-text-field label="Name" v-model="name3" />
</v-text-field>

<v-btn depressed v-on:click="delData(name3)"
        color="primary">
    Delete
</v-btn>

</v-form>
...
```

N.B."..." indicates lines of code which are not shown here

29 - Interface Design and Development, © Swinburne



29

Data Delete (Continued)

```
...
methods: {

    delData: function(nm, age) {
        var delSQLApiURL ='resources/apis.php/name/' + nm;
        var self = this;
        // DELETE request using fetch with error handling
        const requestOptions = {
            method: 'DELETE',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify({
                name: nm
            })
        };
    };
}
```

N.B."..." indicates lines of code which are not shown here

30

Data Delete (Continued)

```
fetch(delSQLApiURL, requestOptions)
  .then( response =>{
    //turning the response into the usable data
    return response.json( );
  })
  .then( data =>{
    //This is the data you wanted to get from url
    self.msg = "Data deleted Successfully"

  })
  .catch(error => {
    self.msg = 'There was an error!';
    self.statusText = error;
  });
```

31 - Interface Design and Development, © Swinburne

