Using Vue with ASP.NET Core By Example

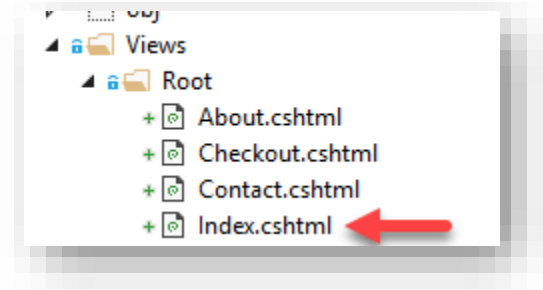
Lab 3 Instructions

Shawn Wildermuth, Instructor

1. Getting Data from the Server
    a. Open the **package.json** file.
    b. Add a new line for "**axios**" and use the latest version:

```
{
  "version": "1.0.0",
  "name": "asp.net",
  "private": true,
  "dependencies": {
    "jquery": "~3.3.1",
    "popper.js": "1.14.1",
    "bootstrap": "4.0.0",
    "bootswatch": "4.0.0",
    "@fortawesome/fontawesome-free-webfonts": "1.0.5",
    "vue": "2.5.16",
    "axios": "0.18.0"
  }
}
```

    c. Once it updates the packages, open the **Views/Root/Index.cshtml**:



    d. Add **axios** to the **scripts** section (before the **index.js**):

```
@section scripts {
  <script src="~/node_modules/vue/dist/vue.js"></script>
  <script src="~/node_modules/axios/dist/axios.js"></script>
  <script src="~/js/index.js"></script>
}
```

    e. Open **index.js**.
    f. Set the **product** collection with an empty array (erasing the demo data):

```
  data: {
    appName: "Product List",
    buyCount: 0,
    products: []
  },
```

g. Create a **mounted** section before methods and assign it a function:

```
...
mounted: function () {

},
methods: {
```

h. Inside the **mounted** function, call **axios.get** using **/api/products** as the URL.
i. Add a call to **then** and **catch** to handle the callback and error conditions:

```
mounted: function () {
  axios.get("/api/products")
    .then(function () {

    })
    .catch(function () {

    });
},
```

j. In the **then** function, add a new parameter called **res**.
k. Set the **this.products** with the **res.data.results** properties.
l. Add **bind(this)** to the end of the callback function:

```
    .then(function (res) {
      this.products = res.data.results;
    }.bind(this))
```

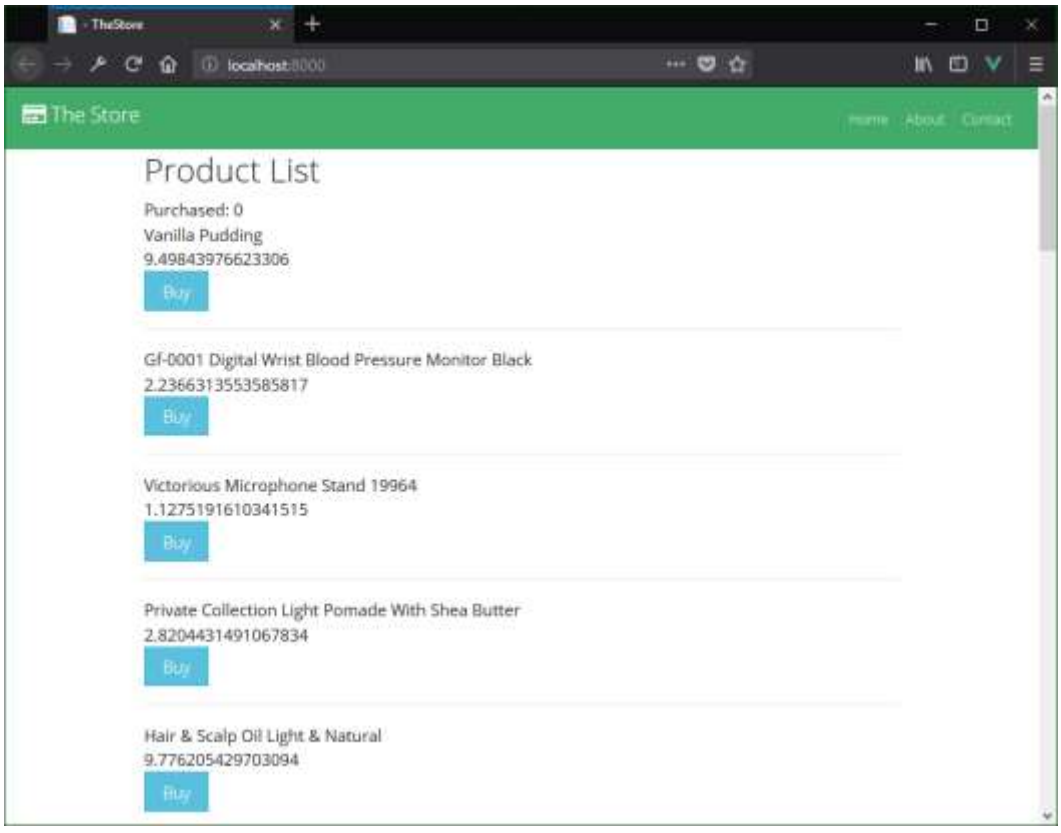
m. Create a new property in **data** called **error**, assign it with an empty string.

```
data: {
  appName: "Product List",
  buyCount: 0,
  products: [],
  error: ""
},
```

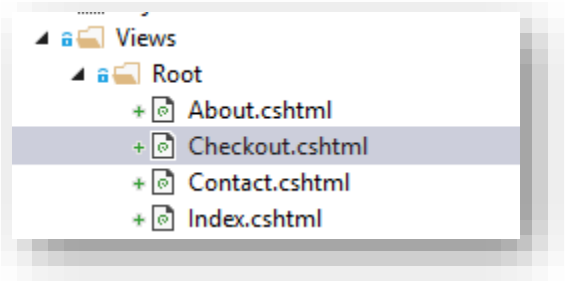
n. In the body of the **mounted** function, set the **error** to an empty string.
o. In the catch part of the call to **get**, and set the error and make sure and add bind(this) to the catch callback too:

```
mounted: function () {
  this.error = "";
  axios.get("/api/products")
    .then(function (res) {
      this.products = res.data.results;
    }.bind(this))
    .catch(function () {
      this.error = "Failed to load products";
    }.bind(this));
},
```

p. Run this and see the products be loaded in the API.



2. Binding to a Form
    a. Open the **Checkout.cshtml** file in the **Views/Root** directory:



    b. Create a **@section Scripts** at the top.
    c. Drag the **Vue.js** file into the **scripts** (like you did in the earlier labs).

```
@section scripts {
  <script src="~/node_modules/vue/dist/vue.js"></script>
}
```

    d. In the **div** add an **id** of **theForm**:

```
@section Scripts {
  <script src="~/node_modules/vue/dist/vue.js"></script>
}
<h2>Checkout</h2>
<div id="theForm">
  TODO
</div>
```

    e. Create a new file in the **wwwroot/js** called **checkout.js**.
    f. Create a new instance of **Vue**.
    g. Set the **el** to "**#theForm**":

```
new Vue({
  el: "#theForm"
});
```

    h. Back in the **Checkout.cshtml**, drag and drop the new **checkout.js** file into the **Scripts** section:

```
@section Scripts {
<script src="~/node_modules/vue/dist/vue.js"></script>
<script src="~/js/checkout.js"></script>
}
```

i. Create a **form** inside the **div**:

```
<div id="theForm">
  <form>
    <div class="form-group">
      <label>Name</label>
      <input class="form-control" />
    </div>
    <div class="form-group">
      <label>Email</label>
      <input class="form-control" />
    </div>
    <div class="form-group">
      <input type="submit" class="btn" value="Save" />
    </div>
  </form>
</div>
```

j. Back in the **checkout.js**, add a **data** section and create a **customer** property with **name** and **email**:

```
new Vue({
  el: "#theForm",
  data: {
    customer: {
      name: "",
      email: ""
    }
  }
});
```

k. Add a **methods** section and add an '**onSave'** function.

l. Call **alert** with **JSON.stringify** to show the state of the entire customer object:

```
new Vue({
  el: "#theForm",
  data: {
    customer: {}
  },
  methods: {
    onSave: function () {
      alert(JSON.stringify(this.customer));
    }
  }
});
```

m. Open the **checkout.cshtml** file.

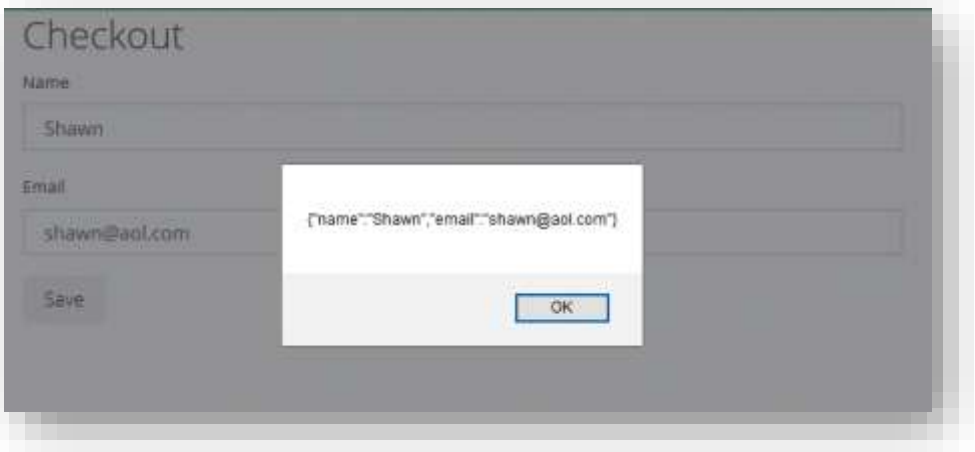
n. Add **v-model** and assign **customer.name** and **customer.email**:

```
<form>
  <div class="form-group">
    <label>Name</label>
    <input class="form-control" v-model="customer.name" />
  </div>
  <div class="form-group">
    <label>Email</label>
    <input class="form-control" v-model="customer.email" />
  </div>
  <div class="form-group">
    <input type="submit" class="btn" value="Save" />
  </div>
</form>
```

o. Add **v-on:submit.prevent** and call **onSave()** afterwards:

```
<form v-on:submit.prevent="onSave()">
```

p. Run the project and change the url to **http://localhost:8000/_/checkout** this to see that what you type in the form, shows up in the alert:



3. Adding Validation
    a. Open the **package.json** file.
    b. Add a new entry for '**vee-validate**' and the current version:

```
{
  "version": "1.0.0",
  "name": "asp.net",
  "private": true,
  "dependencies": {
    "jquery": "~3.3.1",
    "popper.js": "1.14.1",
    "bootstrap": "4.0.0",
    "bootswatch": "4.0.0",
    "@fortawesome/fontawesome-free-webfonts": "1.0.5",
    "vue": "2.5.16",
    "axios": "0.18.0",
    "vee-validate": "2.0.8"
  }
}
```

    c. Once installed, open the **checkout.js** file.
    d. Add **Vue.use(VeeValidate)** to the top of the file:

```
Vue.use(VeeValidate)

new Vue({
  el: "#theForm",
  data: {
    customer: {}
  },
  methods: {
    onSave: function () {
      alert(JSON.stringify(this.customer));
    }
  }
});
```

    e. Open the **checkout.cshtml** file.
    f. Add **vee-validate.js** to the script section:

```
@section Scripts {
  <script src="~/node_modules/vue/dist/vue.js"></script>
  <script src="~/node_modules/vee-validate/dist/vee-validate.js"></script>
  <script src="~/js/checkout.js"></script>
}
```

g. Add **name** and **v-validate** attributes to the first input:

```
<input class="form-control"
       v-model="customer.name"
       name="customerName"
       v-validate="{ required: true }" />
```

h. Add a **span** after the **input** and set its **class** to '**text-danger**'.

i. Inside the **span** use binding to show **errors.first("customerName")** to match the **name** of the **input**:

```
<input class="form-control"
       v-model="customer.name"
       name="customerName"
       v-validate="{ required: true }" />
<span class="text-danger">{{ errors.first("customerName") }}</span>
```

j. Run the page again.

k. Put your cursor in the name input, and tab out of it without typing anything.

l. You should see the error appear:

m. Repeat the process with the second **input** including a second validation for **email**:

```
<label>Email</label>
<input class="form-control"
       v-model="customer.email"
       name="email"
       v-validate="{ required: true, email: true }" />
<span class="text-danger">{{ errors.first("email") }}</span>
```

n. Run it to see both validations working: