Module 3 Tutorial Lab

Restaurant Management System using React Redux part 2

In this tutorial lab, we will build upon the Restaurant Management System we build in Module 2's tutorial lab. We will build the front end using React components and tie the React components to Redux using React Redux.

The finished example should look like this: https://codesandbox.io/s/lr8nn59loz

Step 1. Setting up your project

In this step, we will set up the project structure so that you can keep your project organized. A completed example of this step can be found here : https://codesandbox.io/s/ojo05w15

1. Continue from where you left off in the Module 2 Tutorial. Or you can copy my finished Module 2 tutorial lab here and build from it: https://codesandbox.io/s/934rx9q00p
2. Add `react-redux` as a dependency.
3. Create a folder in `./src` called `components`.
4. Create a folder in `./src` called `containers`.
5. Inside the `components` folder, create a new file named `App.jsx`.
6. Copy the following inside `App.jsx`:

```
import React from "react";

const App = () => {
  return (
    <div>
      <h1>Hello World</h1>
    </div>
  );
};

export default App;
```

7. Copy the following inside `index.js`:

```
import React from "react";
import ReactDOM from "react-dom";
import { createStore } from "redux";
import { Provider } from "react-redux";
import App from "./components/App.jsx";
import reducer from "./reducers/reducer.js";
var store = createStore(reducer);

store.subscribe(() => {
  console.log(store.getState());
});

ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById("root")
);
```

In `index.js` we are doing the following:

- Importing all required components, reducers, and methods
- Creating a store using the reducer

- Subscribing an event listener to log the entire state whenever an action is dispatched
- Passing the store to the Provider components
- Wrapping the App component inside the Provider component
- Rendering the App

8. You should see Hello World rendered to the screen. You are now ready to begin the next step.

Step 2. Creating the Presentional components and the UI

In this step, we will build the presentational components and the UI of our application. A completed example of this step can be found here : https://codesandbox.io/s/74154xxzwx

If you look at the completed application, you can see that we can divide the UI into several react components:

TODO: IMAGE of React component breakdown

Here are the list of React components we will build:

- App - Contains the entire restaurant management application
- Overview - Displays the number of available tables and the total money earned from checked out tables
- Body - Contains the Details, Layout, and Menu components
- Details - Displays the current selected table and current table bill total. Contains OrderList component and ToggleButton component.
- OrderList - contains a list of the items ordered by the current selected table
- ToggleTable - Contains a button that is used to check in/ check out the current selected table
- Layout - Contains 16 TableButton components
- TableButton - Button that is used to select a table
- Menu - Contains 14 OrderButton components
- OrderButton- Button that is used to add items to a selected table that is checked in

Since this course is focused on React Redux and not React, these presentational components will be implemented for you. To complete this step, copy all the presentational components (including App.jsx) and container components from my example into your application: https://codesandbox.io/s/74154xxzwx

You may also fork the example.

In the example, you will also find that some boilerplate code for the containers have been filled out for you as well. However, they are incomplete and the container components are simply the same as the presentational components at the moment since the mapping functions aren't filled out. In this tutorial we will fill out the `mapStateToProps` and `mapDispatchToProps` methods that will describe how we want to connect out presentational components to Redux.

Step 3. Creating the Container components

In this step, we will be implementing the container components for the application. A completed example of this step can be found here : https://codesandbox.io/s/lr8nn59loz

1. Copy the following inside `OverviewContainer.js`:

```
import { connect } from "react-redux";
import Overview from "../components/Overview.jsx";

const mapStateToProps = state => {
  return {
    moneyEarned: state.moneyEarned,
    tableStatusData: state.tableStatusData
  };
};

const OverviewContainer = connect(mapStateToProps, null)(Overview);

export default OverviewContainer;
```

Here, we are passing in `state.moneyEarned` and `state.tableStatusData` as props to the Overview component. They will be addressed as `props.moneyEarned` and `props.tableStatusData` respectively.

2. Copy the following inside `DetailsContainer.js`:

```
import { connect } from "react-redux";
import Details from "../components/Details.jsx";

const mapStateToProps = state => {
  return {
    selectedTable: state.selectedTable,
    items: state.tableData[state.selectedTable]
  };
};

const DetailsContainer = connect(mapStateToProps, null)(Details);

export default DetailsContainer;
```

Here, we are passing in `state.selectedTable` and `state.tableData[state.selectedTable]` as props to the Details component. They will be addressed as `props.selectedTable` and `props.items` respectively.

3. Copy the following inside `ToggleTableContainer.js`:

```
import { connect } from "react-redux";
import ToggleTable from "../components/ToggleTable.jsx";
import toggleTable from "../actions/toggleTable.js";
import incrementMoneyEarned from "../actions/incrementMoneyEarned.js

const mapStateToProps = state => {
  return {
    selectedTable: state.selectedTable,
    tableStatus: state.tableStatusData[state.selectedTable],
    tableData: state.tableData
  };
};

const mapDispatchToProps = dispatch => {
  return {
    onToggle: (id, tableData) => {
      var total = 0;
      for (let i = 0; i < tableData[id].length; i++) {
        total += tableData[id][i].price;
      }
      dispatch(incrementMoneyEarned(total));
      dispatch(toggleTable(id));
    }
  };
};
const ToggleTableContainer = connect(mapStateToProps, mapDispatchToP
  ToggleTable
);

export default ToggleTableContainer;
```

Here, we are passing in `state.selectedTable`, `state.tableStatusData[state.selectedTable]` and `state.tableData` as props to the ToggleTable component. They will be addressed as `props.selectedTable`, `props.tableStatus` and `props.tableData` respectively.

We are also passing in an action dispatcher that will be addressed as `props.onToggle`. The action dispatcher calculates the total bill from the selected table and then dispatches an INCREMENT_MONEY_EARNED action with the total bill. Then a TOGGLE_TABLE action is

dispatched to check out the specified table.

4. Copy the following inside `OrderListContainer.js`:

```
import { connect } from "react-redux";
import OrderList from "../components/OrderList.jsx";
import deleteTableItem from "../actions/deleteTableItem.js";

const mapStateToProps = state => {
  return {
    selectedTable: state.selectedTable,
    items: state.tableData[state.selectedTable]
  };
};

const mapDispatchToProps = dispatch => {
  return {
    onDelete: (tableId, id) => {
      dispatch(deleteTableItem(tableId, id));
    }
  };
};
const OrderListContainer = connect(mapStateToProps, mapDispatchToPro
  OrderList
);

export default OrderListContainer;
```

Here, we are passing in `state.selectedTable` and `state.tableStatusData[state.selectedTable]` as props to the OrderList component. They will be addressed as `props.selectedTable`, and `props.tableStatus` respectively.

We are also passing in an action dispatcher that will be addressed as `props.onDelete`. The action dispatcher will dispatch a DELETE_TABLE_ITEM action with a specified table id to delete from and a specified index to delete.

5. Copy the following inside `TableButtonContainer.js`:

```
import { connect } from "react-redux";
import TableButton from "../components/TableButton.jsx";
import selectTable from "../actions/selectTable.js";

const mapStateToProps = state => {
  return {
    selectedTable: state.selectedTable,
    tableStatusData: state.tableStatusData
  };
};

const mapDispatchToProps = dispatch => {
  return {
    onSelect: id => {
      dispatch(selectTable(id));
    }
  };
};
const TableButtonContainer = connect(mapStateToProps, mapDispatchToP
  TableButton
);

export default TableButtonContainer;
```

Here, we are passing in `state.selectedTable` and `state.tableStatusData` as props to the TableButton Component. They will be addressed as `props.selectedTable`, and `props.tableStatusData` respectively.

We are also passing in an action dispatcher that will be addressed as `props.onSelect`. The action dispatcher will dispatch a SELECT_TABLE action with a specified table id.

6. Copy the following inside `OrderButtonContainer.js`:

```
import { connect } from "react-redux";
import OrderButton from "../components/OrderButton.jsx";
import addTableItem from "../actions/addTableItem.js";

const mapStateToProps = state => {
  return {
    selectedTable: state.selectedTable,
    tableStatusData: state.tableStatusData
  };
};

const mapDispatchToProps = dispatch => {
  return {
    onAdd: (name, price, tableId) => {
      dispatch(addTableItem(name, price, tableId));
    }
  };
};
const OrderButtonContainer = connect(mapStateToProps, mapDispatchToP
  OrderButton
);

export default OrderButtonContainer;
```

Here, we are passing in `state.selectedTable` and `state.tableStatusData` as props to the TableButton Component. They will be addressed as `props.selectedTable`, and `props.tableStatusData` respectively.

We are also passing in an action dispatcher that will be addressed as `props.onAdd`. The action dispatcher will dispatch an ADD_TABLE_ITEM action with a specified table id and an item object with a name and price to add.

7. You should now see the completed application. Test it out by selecting a table and adding a few items to it.

Again, the completed example can be seen here: https://codesandbox.io/s/lr8nn59loz