Hands-on Lab - React Redux

Estimated Time Needed: 40 mins

In this lab, you will be building an increment counter using Redux.

Prerequisites

- Familiarity with linux commands.
- You must have completed the labs in the previous sections.

  All the commands and the shortcut buttons in this lab, in the linux based lab environment that is provided. You may need to do it with appropriate variation if you are not using the lab environment and are doing it on Windows.

Objective:

After completing this lab, you will be able to use state management to increment the counter using Redux. Redux library has all that it requires for store management while react-redux binds react and redux libraries together. It centralizing an application's state and logic, thereby making the state accessible outside of the component.

The store management with redux has 3 main components:

Actions - are blocks of information that send data from your application to your store. Actions must have a type property that indicates the type of action being performed.

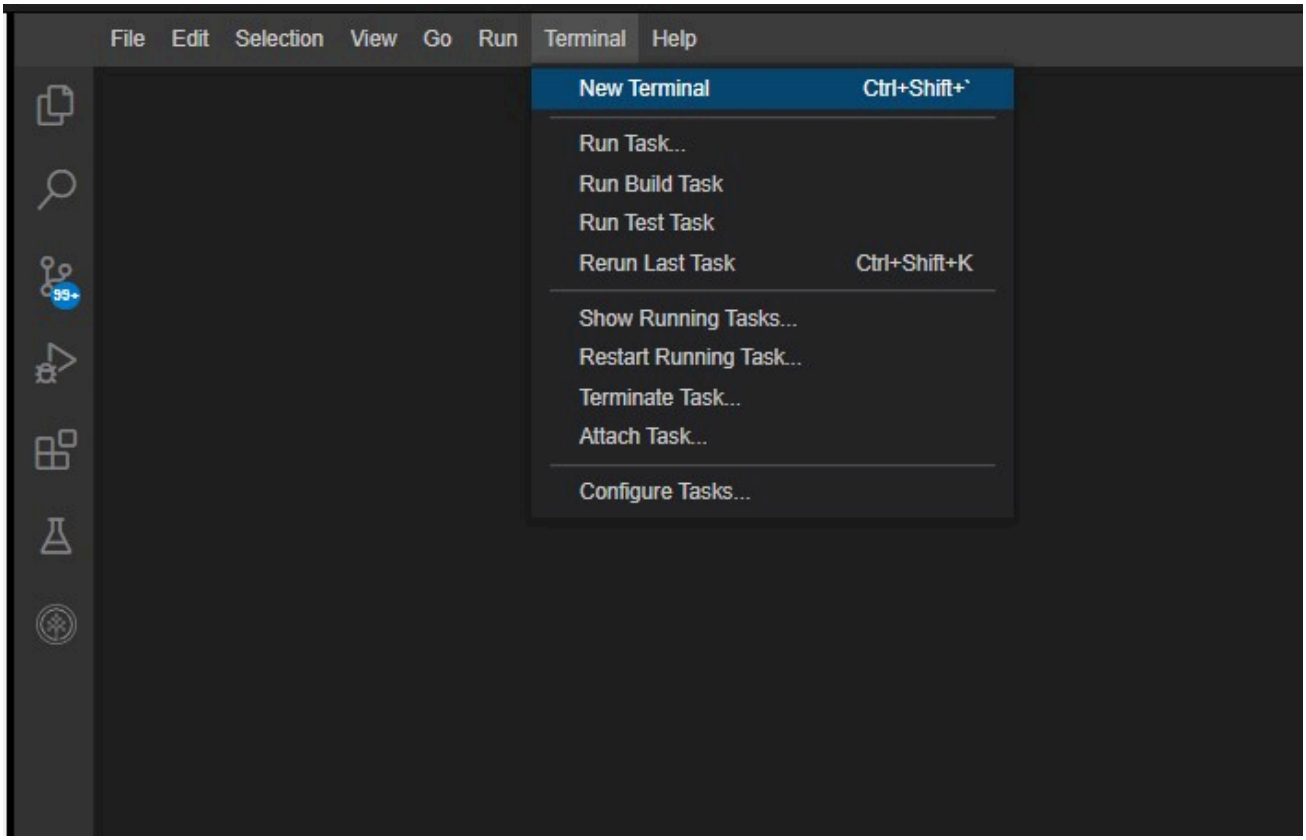
Reducers -Reducers specify how the application's state changes in response to actions sent to the store.

Store -The Store is the object that brings the action and reducer together. The store has the following responsibilities: Holds application state; Allows access to state; Allows state to be updated via dispatch(action);

Set-up:Clone the repository

1. Go to the git repository https://github.com/ibm-developer-skills-network/uqwxd-react_labs.git that contains the starter code needed for this lab and clone the repository.

2. In the lab environment, open a terminal window by choosing Terminal > New Terminal from the menu.



3. Change to your project folder, if you are not in the project folder already.

```
1
cd /home/project
```

Copied! Executed!

4. Clone the Git repository, if it doesn't already exist.

```
1
git clone https://github.com/ibm-developer-skills-network/uqwxd-react_labs.git
```

Copied! Executed!

5. Change to the directory uqwxd-react_todolist to start working on the lab.

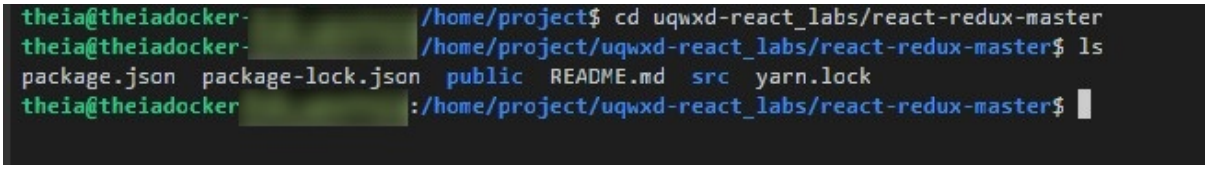
```
1
cd uqwxd-react_labs/react-redux-master
```

Copied! Executed!

6. List the contents of this directory to see the artifacts for this lab.

```
1
ls
```

Copied! Executed!



Creating the increment counter application using Redux in React

In this application that we are building, to learn the use of redux with react, we will have one *MainPanel* component which contains two internal Components, **MyButton** and **DivPanel**.

`MyButton` is a button component. The application maintains a `counter` which keeps track of the number of times the button is clicked. The value of this counter will be displayed in the `DivPanel`. The content of DivPanel will be automatically refreshed everytime the counter value changes. These are effectively two different components.

1. Install the libraries required for your application using the below command. Once installed, verify if the required packages are installed in the package.json file.

```
1
```

```
npm install --save -s redux react-redux react react-dom react-scripts react-service-worker web-vitals
```

Copied!

2. The only action you are going to perform is incrementing of the counter. Edit **src/action/index.js** and paste the following code in it. Click the button below to open **index.js**.

Open **index.js** in IDE

```
1
2
3
4
5
6
7
8
```

```
const increment = (val) => {
    return {
        type : 'INCREMENT',
        inc : val
    }
}

export default increment;
```

Copied!

val is the value you want to increase the counter by everytime the button is clicked. Now that you have your action which defines what is to be done, you will create the reducers which will define how it is done.

3. Edit **src/reducers/index.js** and paste the following code in it. Click the button below to open **index.js**.

Open **index.js** in IDE

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
import {combineReducers} from 'redux'

```

```
const counter = (state=0,action)=>{
    if(action.type === 'INCREMENT') {
        //This will increase the value of counter by the value passed to the increment method
        return state+action.inc;
    }
    //Returns the current value of the counter
    return state;
}

const myReducers = combineReducers({counter});

export default myReducers;
```

Copied!

Now you have your action and reducers. What is left to be created is the store. Before you create the store you will create the components.

4. Edit **src/components/MyButton.js** and paste the following code in it. Click the button below to open **MyButton.js**.

Open **MyButton.js** in IDE

```
1
2
3
4
5
6
7
8
9
10
11
12
```

```
import React from 'react'
import { useDispatch} from 'react-redux';
import increment from '../actions'

const MyButton = ()=>{
    let dispatch = useDispatch();
    return (
        <button onClick={()=>dispatch(increment(1))}>Increase counter</button>
    );
}

export default MyButton;
```

Copied!

`useDispatch` dispatches the event to the store and finds out what action is to be taken and uses the appropriate reducer to do the same.

5. Edit **src/components/DivPanel.js** and paste the following code in it. Click the button below to open **DivPanel.js**.

Open **DivPanel.js** in IDE

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
import React from 'react'
import { useSelector } from 'react-redux';

const DivPanel = () =>{
    let counterVal = useSelector(state => state.counter)
    return (
        <div>
            The present value of counter is {counterVal}
        </div>
    );
}

export default DivPanel;
```

Copied!

`useSelector` is used to select the state from the store whose value you want to access.

6. Edit **src/components/MainPanel.js** and paste the following code in it. Click the button below to open **MainPanel.js**.

Open **MainPanel.js** in IDE

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
import React from 'react'
import MyButton from './MyButton'
import DivPanel from './DivPanel';

const MainPanel = ()=>{
    return (
        <div>
            This is main panel <MyButton></MyButton>
            <DivPanel></DivPanel>
        </div>
    );
}
export default MainPanel;
```

Copied!

7. You have all the panels created. Now let's render the MainPanel through **App.js**. App.js contains the code give below. Click the button below to open **App.js**.

Open **App.js** in IDE

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
import React from 'react';
```

```
import MainPanel from './components/MainPanel';

function App() {
  return (
      <div>
        <MainPanel/>
      </div>
    );
}

export default App;

```

Copied!

8. Now for the final set up of the react application. You need to create and set up the store, where you can manage all the states (in this application, the counter) you want. This is done in the main index.js in the src folder. Click the button below to open **index.js**.

Open **index.js** in IDE

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import {Provider} from 'react-redux'
import myReducers from './reducers'
import {legacy_createStore as createStore} from 'redux';;

//Create the store
const myStore = createStore(myReducers);

//This will console log the current state everytime the state changes
myStore.subscribe(()=>console.log(myStore.getState()));

//Enveloping the App inside the Provider, ensures that the states in the store are available
//throughout the application
ReactDOM.render(<Provider store={myStore}><App/></Provider>, document.getElementById('root'));
```

Copied!

9. In the terminal, ensure you are in the **react-redux** directory and run the following command to start the server and run the application.

```
1
```

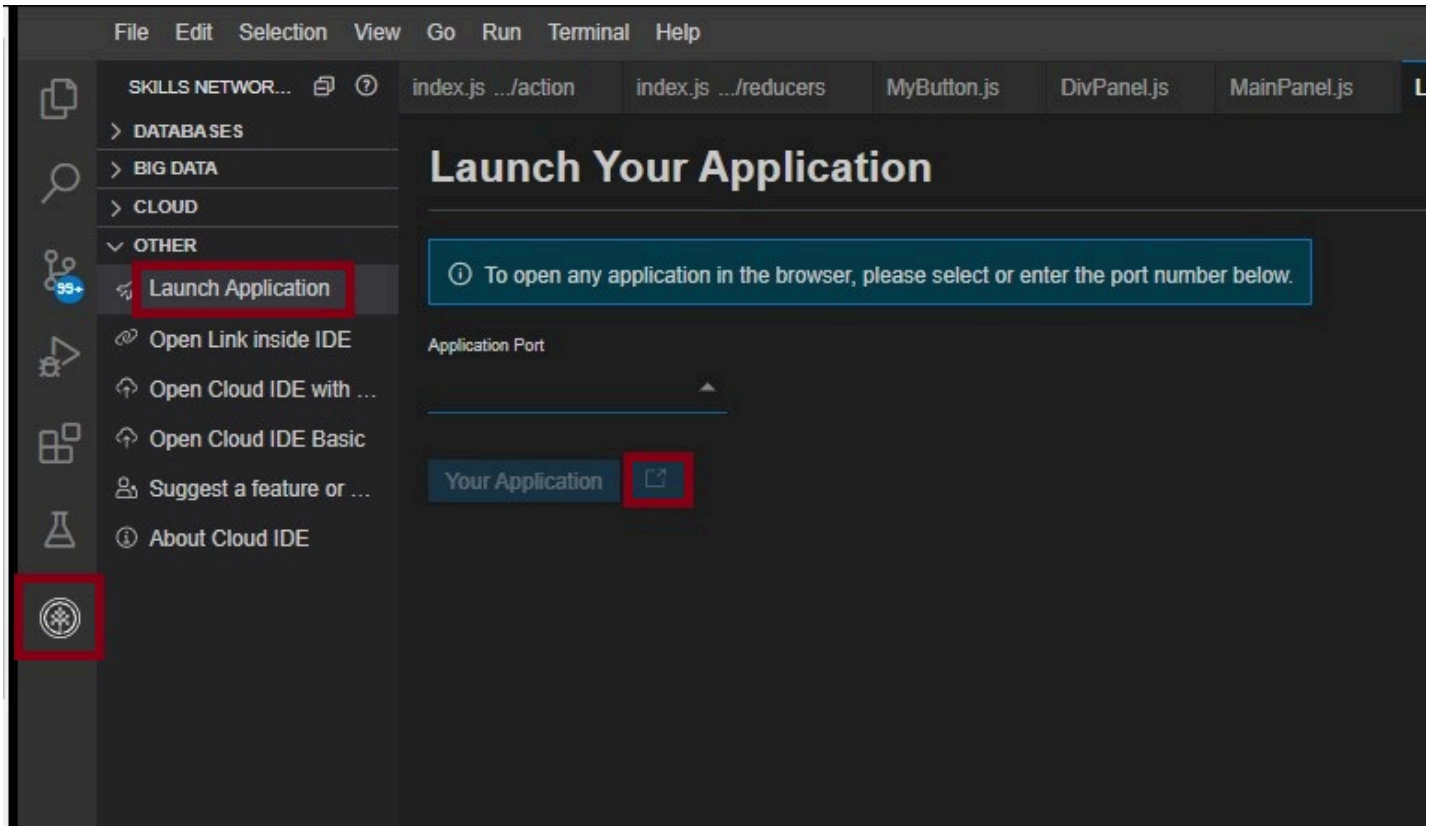
```
npm start
```

Copied! Executed!

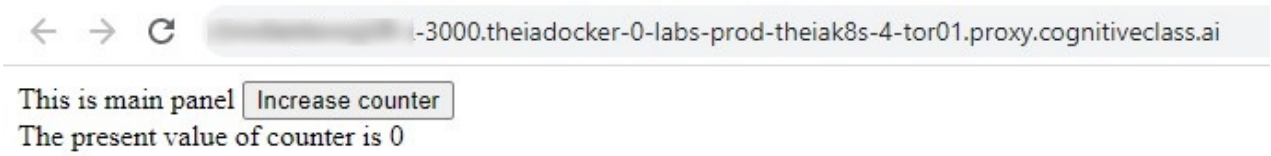
You will see this output indicating that the server is running.

10. To verify that the server is running, click the button below to open the React application on your browser or click on the Skills Network button on the left to open the Skills Network Toolbox. Then click **Other**. Choose Launch Application and enter the port number 3000 on which the server is running and click the launch icon.

Redux Application



The increment counter application using Redux will appear on the browser as seen in the image below.Check the application by incrementing the counter.



11. To stop the server, go to the terminal in the lab environment and press `ctrl + c` to stop the server.

Congratulations! You have completed the lab for creating increment counter Application using Redux in React.

Summary

In this lab, you have used Redux in React to build an increment counter application that allows you to increase the counter each time the increase counter button is clicked.

Author(s)

Sapthashree K S