**Why I need redux in My React Application?**

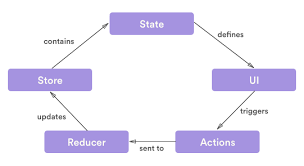
You can manage your whole application with states and props but managing this ever-changing state is hard.

*If a model can update another model, then a view can update a model, which updates another model, and this, in turn, might cause another view to update.*

At some point, you will no longer understand what happens in your app as you have lost control over the when, why, and how of its state.

This is the main motivation to learn redux in React Application.

Let’s first discuss some core concept of Redux, Then we will quick start to make Tasky App.



**Store:** One big object contain all application States.

**Actions:** User will perform action like calling webservice, press button or refresh the list.

**Reducers:** All action will call related reducers and Reducer will update the store.

**UI:** when we connect our UI components to store it will reflect the changes as reducer update store data.

**How to create React native app with Redux from scratch?**

**Create React app**

If you already setup the react-cli then open terminal and run

*create-react-app taskyapp  
cd taskyapp  
npm start*

**Install and configure Redux**

Redux is simple library and we need to use it with react, We will use package called *react-redux*

Install related node packages by *npm* or *yarn*.

*npm install – save redux react-redux react-router-dom redux-thunk redux-persist redux-logger lodash react-bootstrap*

*npm install redux*

**Packages info**

* **redux**: library for implementation of redux
* **react-redux**: combine redux with react
* **react-router-dom**: used for navigation in react application
* **redux-thunk**: redux directly do not support asynchronous coding, So we need to use this. We can also use redux-saga instead of it.
* **redux-persist**: It will make our store object persistent.
* **redux-logger**: it will create console log for every action user dispatch.
* **lodash**: javascript library for manipulating array.
* **React-bootstrap**: using bootstrap classes in React application.

Add bootstrap linking in *public/index.html* by adding following code.

|  |  |
| --- | --- |
|  | <link rel="stylesheet" |
|  | href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" |
|  | integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" |
|  | crossorigin="anonymous"> |
|  |  |
|  | <!-- Optional theme --> |
|  | <link rel="stylesheet" |
|  | href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" |
|  | integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" |
|  | crossorigin="anonymous"> |

**view rawindex.html** hosted with  by **GitHub**

**public/index.html**

And for other linking method go to **React-bootstrap**.

Make *src/actions*, *src/reducers*, *src/screens*, *src/component*, *src/store* folders.

*mkdir*

*actions*

*reducers*

*store*

*containers*

*components  
touch actions/index.js store/store.js reducers/index.js containers/main.js containers/tasky.js*

**Wire your redux – store, action, containers and reducers**

Make store/store.js file.and add the following code.

|  |  |
| --- | --- |
|  | import { createStore, applyMiddleware } from 'redux'; |
|  | import thunk from 'redux-thunk'; |
|  | import reducers from '../reducers/index'; |
|  | import { persistStore, persistReducer } from 'redux-persist' |
|  | import storage from 'redux-persist/lib/storage' |
|  | const persistConfig = { |
|  | key: 'root', |
|  | storage: storage, |
|  | } |
|  | const middlewares = [thunk]; |
|  | if (process.env.NODE\_ENV === `development`) { |
|  | const { logger } = require(`redux-logger`); |
|  |  |
|  | middlewares.push(logger); |
|  | } |
|  | const persistedReducer = persistReducer(persistConfig, reducers) |
|  | export default () => { |
|  | let store = createStore( |
|  | persistedReducer, |
|  | applyMiddleware(...middlewares) |
|  | ) |
|  | let persistor = persistStore(store) |
|  | return { store, persistor } |
|  | } |

**view rawstore.js** hosted with  by **GitHub**

**store.js**

* First we import *combineReducer* from reducer index which we are going to write later in this blog.
* In *persistConfig*, we are using local storage to save our store data So, Store data will not initialize every time.
* Then we will make array of middleware and adding *react-logger* middleware only in development purpose for generate logs of all actions trigger by user.
* Then *createStore* with persistent combine reducer and middleware and export it.

Make *reducer/index.js* file.and add the following code:

|  |  |
| --- | --- |
|  | import { combineReducers } from 'redux'; |
|  | const rootReducer = combineReducers({ |
|  | }); |
|  | export default rootReducer; |

**view rawindex.js** hosted with  by **GitHub**

**reducer/index.js**

* Import *combineReducers* and export it,We will edit this file later.

Now it’s time to make one simple component to wrap with redux provider.

So lets edit file *containers/Tasky.js* with below code:

|  |  |
| --- | --- |
|  | import React, { Component } from 'react'; |
|  | class Tasky extends Component { |
|  | constructor(){ |
|  | super(); |
|  | this.state={}} |
|  | render() { |
|  | return ( |
|  | <div className={'container-fluid text-center'}> |
|  | <p>Hii from Task Container</p> |
|  | </div> |
|  | ); |
|  | } |
|  | } |
|  | export default (Tasky); |

**view rawTasky.js** hosted with  by **GitHub**

Now edit *main.js* with below code.

|  |  |
| --- | --- |
|  | import React, { Component } from 'react'; |
|  | import { BrowserRouter as Router, Switch, Route, Link } from 'react-router-dom'; |
|  | import Tasky from './Tasky' |
|  |  |
|  | class Main extends Component { |
|  | render() { |
|  | return ( |
|  | <Router> |
|  | <div> |
|  | <nav className="navbar navbar-default"> |
|  | <div className="container-fluid"> |
|  | <div className="navbar-header"> |
|  | <Link className="navbar-brand" to={'/'}> |
|  | <p>{"ToDo App"}</p> |
|  | </Link> |
|  | </div> |
|  | <ul className="nav navbar-nav"> |
|  | <li className="active"> |
|  | <Link to={'/'}>Tasky</Link> |
|  | </li> |
|  | </ul> |
|  | </div> |
|  | </nav> |
|  | <Switch> |
|  | <Route exact path='/' component={Tasky} /> |
|  | </Switch> |
|  | </div> |
|  | </Router> |
|  | ); |
|  | } |
|  | } |
|  | export default Main; |

**view rawmain.js** hosted with  by **GitHub**

* In above code we are defining routes for all application with *react-router-dom*.
* In *<Router>* tag we are defining *<Switch>*.*<Switch>* is using for navigate only one route from the list.When we click on *<Link>* tag, its To parameter will decide which route to navigate.

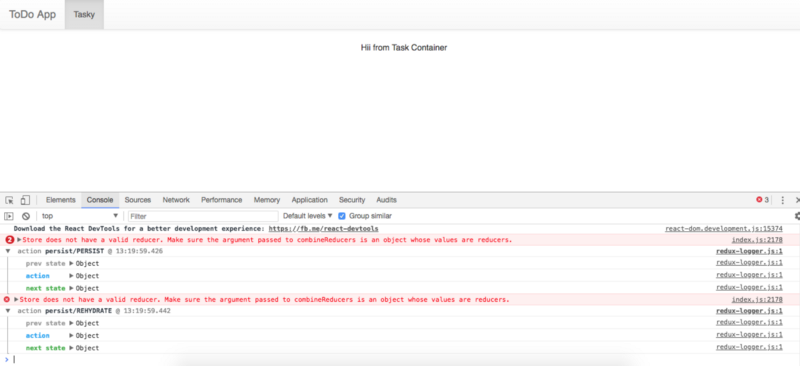
Now, we are going to bind our redux store with root component of application. Edit *app.js* with the following code.

|  |  |
| --- | --- |
|  | import React, { Component } from 'react'; |
|  | import './App.css'; |
|  | import Main from './containers/main' |
|  | import { Provider } from 'react-redux'; |
|  | import { PersistGate } from 'redux-persist/lib/integration/react' |
|  | import configureStore from './store/store' |
|  | let { store, persistor } = configureStore() |
|  | class App extends Component { |
|  | render() { |
|  | return(<Provider store={store}> |
|  | <PersistGate loading={null} persistor={persistor}> |
|  | <div className="App"> |
|  | <Main/> |
|  | </div> |
|  | </PersistGate> |
|  | </Provider> |
|  | ); |
|  | }} |
|  | export default App; |

**view rawapp.js** hosted with  by **GitHub**

**app.js**

If everything is working well you will get following output.



You will find some logs in console screens.

As we have initialise *redux-logger* package with our store middle-ware, It is showing logs for all actions which is dispatch. one error is showing because we have not initialise any reducer yet.

Now, we are going to make some basic UI (some pure react components) as our main focus in tutorial is redux don’t bother about components and UI design.

*cd src/components  
touch addTask.js filterTask.js listTask.js listComponent.js*

Above command will create related files and edit this files with below code.

First we change the add task code. Edit *addTask.js*

|  |  |
| --- | --- |
|  | import React, { Component } from 'react'; |
|  | class AddTask extends Component { |
|  | saveTask = () => { |
|  | let newTaskObj = { |
|  | title:document.getElementById('txtTitle').value, |
|  | description:document.getElementById('txtDesc').value, |
|  | priority:document.getElementById('selPriority').value, |
|  | id:new Date().valueOf() |
|  | } |
|  | this.props.saveTask(newTaskObj); |
|  | document.getElementById('txtTitle').value='' |
|  | document.getElementById('txtDesc').value='' |
|  | document.getElementById('selPriority').value='High' |
|  | } |
|  | render() { |
|  | return ( |
|  | <div className={'panel panel-default'} > |
|  | <div className="panel-heading">Add Task</div> |
|  | <div className="panel-body"> |
|  | <div className={"form-group"}> |
|  | <div className={"col-sm-3"}> |
|  | <input className={'form-control'} |
|  | id='txtTitle' |
|  | placeholder={'Title'} |
|  | type={'text'} /> |
|  | </div> |
|  | <div className={"col-sm-5"}> |
|  | <input className={'form-control'} |
|  | id='txtDesc' |
|  | placeholder={'Description'} |
|  | type={'text'} /> |
|  | </div> |
|  | <div className="col-sm-2"> |
|  | <select id={"selPriority"} className={'form-control'}> |
|  | <option value={'High'}>High</option> |
|  | <option value={'Medium'}>Medium</option> |
|  | <option value={'Low'}>Low</option> |
|  | </select> |
|  | </div> |
|  | <div className={"col-sm-2"}> |
|  | <button |
|  | className={'btn btn-info'} |
|  | onClick={()=>{this.saveTask()}}>Add Task</button> |
|  | </div> |
|  | </div> |
|  |  |
|  | </div> |
|  | </div> |
|  | ); |
|  | }} |
|  | export default AddTask; |

**view rawaddTask.js** hosted with  by **GitHub**

Edit *listTask.js*

|  |  |
| --- | --- |
|  | import React, { Component } from 'react'; |
|  | import TaskComponent from './listComponent' |
|  | class ListTask extends Component { |
|  | render() { |
|  | return ( |
|  | <div> |
|  | <div className={'panel panel-default'} > |
|  | <div className="panel-heading">Tasks</div> |
|  | <div className="panel-body"> |
|  | { |
|  | this.props.list.map((task,index)=> { |
|  | return <div key={index} className={'list-group-item col-sm-12'}> |
|  | <TaskComponent deleteTask={this.props.deleteTask} editTask={this.props.editTask} task={task}/> |
|  | </div> |
|  | }) |
|  | } |
|  | </div> |
|  | </div> |
|  | </div> |
|  | ); |
|  | }} |
|  | export default ListTask; |
|  |  |

**view raweditTask.js** hosted with  by **GitHub**

Edit *listComponent.js*

|  |  |
| --- | --- |
|  |  |
|  | import React, { Component } from 'react'; |
|  | import {Label} from 'react-bootstrap'; |
|  | import ModalComponent from './modalComponent'; |
|  |  |
|  |  |
|  | class ListComponent extends Component { |
|  | constructor(props){ |
|  | super(props); |
|  | this.state={ |
|  | show: false |
|  | } |
|  | } |
|  | handleHide = () => { |
|  | this.setState({ show: false }); |
|  | } |
|  | delete = (id) => { } |
|  | render() { |
|  | const {title,description,priority,id} = this.props.task; |
|  | return ( |
|  | <div> |
|  | <div className={'col-sm-6'}> |
|  | <h4>{title} {" "}<small>{description}</small></h4> |
|  | </div> |
|  | <div className={'col-sm-2'}> |
|  | <h4><Label bsStyle={(priority==='High')?"danger":(priority==='Medium')?"primary":"warning"}>{priority}</Label></h4> |
|  | </div> |
|  | <div className={'col-sm-1'} onClick={()=>{this.setState({ show: true })}}> |
|  | <h4><span className="glyphicon glyphicon-pencil"/></h4> |
|  | </div> |
|  | <div className={'col-sm-1'} onClick={()=>{this.delete(id)}}> |
|  | <h4><span className={'glyphicon glyphicon-trash'}/></h4> |
|  | </div> |
|  | <ModalComponent show={this.state.show} task={this.props.task} editTask={this.props.editTask} handleHide={this.handleHide} /> |
|  | </div> |
|  | ); |
|  | }} |
|  | export default ListComponent; |

**view rawlistComponent.js** hosted with  by **GitHub**

Edit *filterTask.js*

|  |  |
| --- | --- |
|  | import React, { Component } from 'react'; |
|  | class FilterScreen extends Component { |
|  | render() { |
|  | return ( |
|  |  |
|  | <div className={'panel panel-default'} > |
|  | <div className="panel-heading">Filters</div> |
|  | <div className="panel-body"> |
|  | <div className={'list-group-item'}> |
|  | <div className={'row'}> |
|  | <p> Priority</p> |
|  | <select id={'PriSelector'} onChange={()=>this.props.onPriorityFilterChange(document.getElementById('PriSelector').value)}> |
|  | <option value={'All'} defaultValue={(""===this.props.selectedPriority)}>All</option> |
|  | <option value={'High'} defaultValue={("High"===this.props.selectedPriority)}>High</option> |
|  | <option value={'Medium'} defaultValue={("Medium"===this.props.selectedPriority)}>Medium</option> |
|  | <option value={'Low'} defaultValue={("Low"===this.props.selectedPriority)}>Low</option> |
|  | </select> |
|  | </div> |
|  | </div> |
|  | <div className={'list-group-item'}> |
|  | <p>Tasks Search</p> |
|  | <input type="text" id={'txtSearch'} |
|  | className="form-control form-control-lg" |
|  | placeholder="Search" |
|  | value={this.props.searchText} |
|  | onChange={()=>{ |
|  | this.props.onSearchTextChange(document.getElementById('txtSearch').value) |
|  | }}/> |
|  | </div> |
|  | </div> |
|  | </div> |
|  |  |
|  | ); |
|  | } |
|  | } |
|  |  |
|  | export default FilterScreen; |

**view rawfilterTask.js** hosted with  by **GitHub**

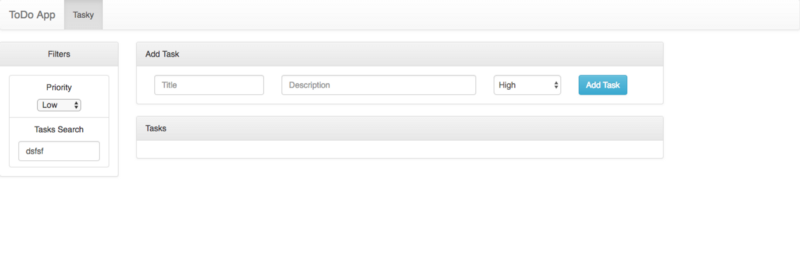
Now all we need is to call all component in our tasky component.

Edit *tasky.js* with the following code:

|  |  |
| --- | --- |
|  | import React, { Component } from 'react'; |
|  | import AddTask from '../components/addTask' |
|  | import FilterScreen from '../components/filterTask'; |
|  | import ListTask from '../components/listTask' |
|  | class Tasky extends Component { |
|  | constructor(){ |
|  | super(); |
|  | this.state={ |
|  | currentPage:1, |
|  | taskPerPage:5 |
|  | } |
|  | } |
|  | saveTask = (newTask) => {}; |
|  | deleteTask = (id) => { }; |
|  | editTask = (editedTask) => { } |
|  | onPriorityFilterChange = (value) => { }; |
|  | onSearchTextChange = (text) => { }; |
|  | render() { |
|  | const { currentPage, taskPerPage } = this.state; |
|  | let taskys = [] |
|  | const indexOfLastTodo = currentPage \* taskPerPage; |
|  | const indexOfFirstTodo = indexOfLastTodo - taskPerPage; |
|  | const currentTodos = taskys.slice(indexOfFirstTodo, indexOfLastTodo); |
|  | const pageNumbers = []; |
|  | for (let i = 1; i <= Math.ceil(taskys.length / taskPerPage); i++) { |
|  | pageNumbers.push(i); |
|  | } |
|  | return ( |
|  | <div className="row content"> |
|  | <div className="col-sm-2 sidenav"> |
|  | <FilterScreen onPriorityFilterChange={this.onPriorityFilterChange} |
|  | onSearchTextChange={this.onSearchTextChange} |
|  | /> |
|  | </div> |
|  | <div className={'col-sm-8 text-left'}> |
|  | <AddTask saveTask={this.saveTask}/> |
|  | <ListTask list={currentTodos} |
|  | deleteTask={this.deleteTask} |
|  | editTask={this.editTask} |
|  | /> |
|  | <ul className="pagination"> |
|  | { pageNumbers.map((i,index)=>{ |
|  | return <li key={index} onClick={()=>{this.setState({currentPage:i})}}> |
|  | <a>{i}</a> |
|  | </li> |
|  | })} |
|  | </ul> |
|  | </div> |
|  | </div> |
|  | ); |
|  | }} |
|  | export default (Tasky); |

**view rawtasky.js** hosted with  by **GitHub**

After replacing all code when you run application again you will get following output.



Up to this, it is a simple react application. Now we will store our task list in redux store object and add edit filter and delete our task with redux actions.

First we will make one reducer of task. For that we need to navigate to reducer directory and make file *taskReducer.js* and add the following code.

|  |  |
| --- | --- |
|  | const INITIAL\_STATE = { |
|  | task:[{ |
|  | title:"title",description:"description",priority:'High',id:"3678" |
|  | }], |
|  | }; |
|  | export default (state = INITIAL\_STATE, action) => { |
|  | switch (action.type) { |
|  | default: |
|  | return state; |
|  | }}; |

**view rawtaskreducer.js** hosted with  by **GitHub**

This is the basic code that every reducer contain. We can define our initial state of store object in reducer.

Now we need to add this *taskreducer* to *combineReducer* and use that in store. To implement this we will edit *reducer/index.js* file with following lines:

|  |  |
| --- | --- |
|  | import { combineReducers } from 'redux'; |
|  | import TaskReducer from './taskReducer'; //add this line |
|  |  |
|  | const rootReducer = combineReducers({ |
|  | task:TaskReducer //add taskreducer and name is task for future use. |
|  | }); |
|  |  |
|  | export default rootReducer; |

**view rawindex.js** hosted with  by **GitHub**

Now, we will connect *tasky.js* file to store.

For this connect function is used it is from *react-redux* package. It has two parameters

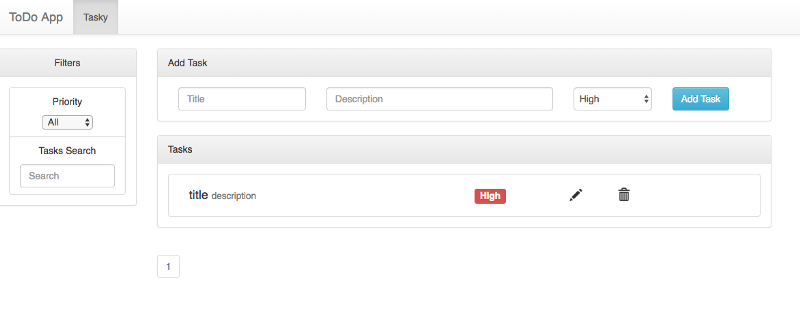
* *mapStateToProps*: In this we get store object value in our component
* *mapActionToProps*: In this all actions can be registered to edit store. All actions are defined in actions files.

Now, edit *tasky.js* with following lines:

|  |  |
| --- | --- |
|  | import { connect } from 'react-redux' |
|  | //edit render function |
|  | render() { |
|  | const { currentPage, taskPerPage } = this.state; |
|  | let taskys = this.props.tasks |
|  | ... |
|  |  |
|  | //edit last line with |
|  | const mapStateToProps = (state) => ({ |
|  | tasks:state.task.task, |
|  | }); |
|  | const mapActionToProps = ({}); |
|  | export default connect(mapStateToProps, mapActionToProps)(Tasky); |

**view rawtasky.js** hosted with  by **GitHub**

After connecting you will get output like this.



Now work on add Task functionality.

For this, first we need to add constants of action in index file and make one *taskAction.js* file in same folder.

Edit *actions/index.js*

*export const CREATE\_TASK = “CREATE\_TASK”;*

Edit *action/taskAction.js*

|  |  |
| --- | --- |
|  | import {CREATE\_TASK} from "./index"; |
|  | export const createTask = (request) => { |
|  | return { |
|  | type: CREATE\_TASK, |
|  | payload: request |
|  | }; |
|  | } |

**view rawtaskAction.js** hosted with  by **GitHub**

In taskAction we have created one function called *createTask* in that we are returning one object having type and payload.

So now in reducer we need to define this type to modified the store.

Now, we need to add one case for *createTask* in *taskReducer.js* file.

|  |  |
| --- | --- |
|  | ..... |
|  |  |
|  | case CREATE\_TASK: { |
|  | return { |
|  | ...state, |
|  | task : \_.concat(...state.task,action.payload) |
|  | };} |

**view rawtaskreducer.js** hosted with  by **GitHub**

Now need to call that action when Add task button is clicked,So edit Tasky.js with following code.

|  |  |
| --- | --- |
|  | import {createTask} from '../actions/taskAction' |
|  | ... |
|  | saveTask = (newTask) => { |
|  | this.props.createTask(newTask); |
|  | }; |
|  | ... |
|  | const mapActionToProps = ({ |
|  | createTask |
|  | }); |

**view rawtasky.js** hosted with  by **GitHub**