1. src\index.js

ساختنstate موجود در redux(store)

const rootReducer = combineReducers({

    ctr: counterReducer, *// ctr = {counter} =>> state.ctr.counter*

    res: resultReducer *// res= {results} =>> state.res.results*

*//this state is the state in store(in redux),the state of redux*

});

src\store\reducers\result.js+src\store\reducers\counter.js

const store = createStore(rootReducer, composeEnhancers(applyMiddleware(logger)));

.2پاس دادنstoreبه کامپوننت های react

*//The store is in redux passed as props to components*

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

3.src\App.js

class App extends Component {

  render() {

    return (

      <div className="App">

       <Counter />

      </div>

    );

  }

}

4.src\containers\Counter\Counter.js

4-1.

*//this state is the state in store(in redux),the state of redux*

const mapStateToProps = *state* => {

    return {

        ctr: *state*.ctr.counter,

        storedResults: *state*.res.results

    }

};

4-2.

    render() {

        return (

            <div>

                <CounterOutput value={**this**.props.ctr} />

                <CounterControl label="Increment" clicked={**this**.props.onIncrementCounter} />

                <CounterControl label="Decrement" clicked={**this**.props.onDecrementCounter} />

                <CounterControl label="Add 10" clicked={**this**.props.onAddCounter} />

                <CounterControl label="Subtract 15" clicked={**this**.props.onSubtractCounter} />

                <hr />

                <button onClick={() => **this**.props.onStoreResult(**this**.props.ctr)}>Store Result</button>

                <ul>

                    {**this**.props.storedResults.map(*strResult* => (

                        <li key={*strResult*.id} onClick={() => **this**.props.onDeleteResult(*strResult*.id)}>{*strResult*.value}</li>

                    ))}

                </ul>

            </div>

        );

    }

5-1.

پس از کلیک روی دکمه ها ی Increment، Decrement، Add 10، Subtract 15،actionها ارسال میشوند به store(reducers)

const mapDispatchToProps = *dispatch* => {

    return {

*//dispatch({type: actionTypes.INCREMENT}) => dispatch({type: INCREMENT} => action.type == INCREMENT*

        onIncrementCounter: () => dispatch(actionCreators.increment()),

        onDecrementCounter: () => dispatch(actionCreators.decrement()),

        onAddCounter: () => dispatch(actionCreators.add(10)),

        onSubtractCounter: () => dispatch(actionCreators.subtract(15)),

    }

};

5-1-1. src\store\actions\index.js

export {

    add,

    subtract,

    increment,

    decrement

} from './counter';

export {

    storeResult,

    deleteResult

} from './result';

5-1-2.

ساخت actionها توسط *ActionCreator*

src\store\actions\counter.js

import \* as actionTypes from './actionTypes';

*//increment  is ActionCreator*

export const increment = () => {

    return {

        type: actionTypes.INCREMENT

    };

};

*//decrement  is ActionCreator*

export const decrement = () => {

    return {

        type: actionTypes.DECREMENT

    };

};

*//add  is ActionCreator*

export const add = ( *value* ) => {

    return {

        type: actionTypes.ADD,

        val: *value*

    };

};

*//subtract  is ActionCreator*

export const subtract = ( *value* ) => {

    return {

        type: actionTypes.SUBTRACT,

        val: *value*

    };

};

5-1-3.reducer in store of redux

src\store\reducers\counter.js

const reducer = ( *state* = initialState, *action* ) => {

    switch ( *action*.type ) {

        case actionTypes.INCREMENT:

            return updateObject(*state*, {counter: *state*.counter + 1});

        case actionTypes.DECREMENT:

            return updateObject(*state*, {counter: *state*.counter - 1});

        case actionTypes.ADD:

            return updateObject(*state*, {counter: *state*.counter + *action*.val});

        case actionTypes.SUBTRACT:

            return updateObject(*state*, {counter: *state*.counter - *action*.val});

    }

    return *state*;

};

5-1-4. src\store\utility.js

export const updateObject = (*oldObject*, *updatedValues*) => {

    return {

        ...*oldObject*,

        ...*updatedValues*

    }

};

5-1-5. src\containers\Counter\Counter.js

*//this state is the state in store(in redux),the state of redux*

const mapStateToProps = *state* => {

    return {

        ctr: *state*.ctr.counter,          *//props.ctr*

        storedResults: *state*.res.results *//props.storedResults*

    }

};

export default connect(mapStateToProps, mapDispatchToProps)(Counter);

5-1-6. src\containers\Counter\Counter.js

<CounterOutput value={**this**.props.ctr} />

5-2.

پس از کلیک روی دکمه ها ی onDeleteResult ، Store Result ،actionها ارسال میشوند به store(reducers)

5-2-1. src\store\actions\index.js

export {

    add,

    subtract,

    increment,

    decrement

} from './counter';

export {

    storeResult,

    deleteResult

} from './result';

5-2-2.

ساخت actionها توسط *ActionCreator*

src\store\actions\result.js

import \* as actionTypes from './actionTypes';

*//storeResult  is ActionCreator*

export const storeResult = (*res*) => {

    return {

        type: actionTypes.STORE\_RESULT,

        result: *res*

    };

}

*// redux-thunk*

*//saveResult  is ActionCreator*

*/\* export const saveResult = ( res ) => {*

*return {*

*type: actionTypes.STORE\_RESULT,*

*result: res*

*};*

*}*

*//redux-thunk*

*//storeResult is thunk function which returns a function(ActionCreator) instead of  action*

*export const storeResult = ( res ) => {*

*return (dispatch, getState) => {*

*setTimeout( () => {*

*const oldCounter = getState().ctr.counter;*

*console.log(oldCounter);*

*dispatch(saveResult(res));*

*}, 2000 );*

*}*

*}; \*/*

*//deleteResult  is ActionCreator*

export const deleteResult = ( *resElId* ) => {

    return {

        type: actionTypes.DELETE\_RESULT,

        resultElId: *resElId*

    };

};

5-2-3.reducer in store of redux

const reducer = (*state* = initialState, *action*) => {

    switch (*action*.type) {

        case actionTypes.STORE\_RESULT:

            return updateObject(*state*, {

                results: *state*.results.concat({

                    id: new Date(),

*// value: action.result \* 2*

                    value: *action*.result

                })

            });

        case actionTypes.DELETE\_RESULT:

            const updatedArray = *state*.results.filter(*result* => *result*.id !== *action*.resultElId);

            return updateObject(*state*, {

                result: updatedArray

            });

    }

    return *state*;

};

5-2-4. src\store\utility.js

export const updateObject = (*oldObject*, *updatedValues*) => {

    return {

        ...*oldObject*,

        ...*updatedValues*

    }

};

5-2-5. src\containers\Counter\Counter.js

*//this state is the state in store(in redux),the state of redux*

const mapStateToProps = *state* => {

    return {

        ctr: *state*.ctr.counter,          *//props.ctr*

        storedResults: *state*.res.results *//props.storedResults*

    }

};

export default connect(mapStateToProps, mapDispatchToProps)(Counter);

5-2-6. src\containers\Counter\Counter.js

 <ul>

{

**this**.props.storedResults.map(*strResult* => (

<li key={*strResult*.id} onClick={() => **this**.props.onDeleteResult(*strResult*.id)}>{*strResult*.value}</li>

))

}

</ul>