Exercise (Instructions): React-Redux-Form Revisited

Objectives and Outcomes

In this exercise we will explore the interaction between react-redux-form and the Redux store. We will see how to map the form into the store so that the state of the form will be persisted in the store. At the end of this exercise you will be able to:

- Use react-redux-form to interact with Redux store and store the state of the form in the store.

Updating the Feedback Form

- Add a new file named *forms.js* in the *redux* folder and add the following to it:

```
export const InitialFeedback = {
    firstname: '',
    lastname: '',
    telnum: '',
    email: '',
    agree: false,
    contactType: 'Tel.',
    message: ''
};
```

- Then, open *configureStore.js* and update it to add the form to the reducers:

```
        }),

. . .
```

- Next, open *MainComponent.js* and update it as follows:

```
. . .

import { actions } from 'react-redux-form';

. . .

  resetFeedbackForm: () => { dispatch(actions.reset('feedback'))}

. . .

              <Route exact path='/contactus' component={() => <Contact
                resetFeedbackForm={this.props.resetFeedbackForm} />} />
        console.log('Current State is: ' + JSON.stringify(values));
        alert('Current State is: ' + JSON.stringify(values));
        this.props.resetFeedbackForm();
        // event.preventDefault();
    }

. . .

                        <Form model="feedback" onSubmit={(values) => this
                          .handleSubmit(values)}>

                          . . .


                        </Form>

. . .
```

- Save all the changes and do a Git commit with the message "React Redux Forms Revisited".

Conclusions

In this exercise we have seen how to use react-redux-form together with Redux to persist form state.