Atif Follow
Oct 22, 2019 · 2 min read · Listen

User Session in React JS

Session implementation in react application is quite different then other application developed in technologies like .net or java.

There are different recommendation for session management in react apps. First of all we will talk about these options and then will go through to one I have used.

Cookies or Local-storage with Closure as a Wrapper

Use a closure wrapper to maintain user information. Once user is logged in we can store that information in cookies/local-storage and can be retrieve in app components.

Here we have a wrapper to hold user info .

```
var UserProfile = (function() {
  var full_name = "";

  var getName = function() {
    return full_name;    // Or pull this from cookie/localStorage
  };

  var setName = function(name) {
    full_name = name;
    // Also set this in cookie/localStorage
  };

  return {
    getName: getName,
    setName: setName
  }

})();

export default UserProfile;
```

```
import UserProfile from './UserProfile';

UserProfile.setName("Some Guy");
```

Data can be retrieved in any component from wrapper.

```
import UserProfile from './UserProfile';

UserProfile.getName();
```

Redux React Session API

Node Package Manager (npm) has a very useful API ***redux-react-session*** to maintain session is react application using redux store. This API provide method like ***initSessionService, refreshFromLocalStorage, checkAuth*** and some advance features like ***Immutable JS.***

API has its own ***sessionReducer*** and ***sessionService.*** Following is the way to add reducer in **combineReducers**.

```
import { combineReducers } from 'redux';

import { sessionReducer } from 'redux-react-session';

const reducers = {

// ... your other reducers here ...

session: sessionReducer

};

const reducer = combineReducers(reducers);
```

We can initiate our session using API service while creating the redux store.

```
import { createStore } from 'redux';

import { sessionService } from 'redux-react-session';
```


To store session we can use INDEXEDDB, WEBSQL, LOCALSTORAGE or COOKIES like…

```
const options = { refreshOnCheckAuth: true, redirectPath: '/home',
driver: 'COOKIES'};

sessionService.initSessionService(store, options)
```

Once we have initiate session promise will be resolved or rejected and session flag *checked* will be updated to *true*.

If session is active or expired *authenticated* flag is set to *true* or *false*. We can retrieve this flag from session to set our routing.

```
const { bool } = PropTypes;

Routes.propTypes = {

sessionInfo: bool.isRequired,

checked: bool.isRequired

};

const mapStateToProps = state => {

return{sessionInfo: state.session.authenticated,  checked:
state.session.checked,}

};

export default connect(mapStateToProps)(Routes);
```

I have implemented customized routing.

```
const Routes = (sessionInfo,checked) => (sessionInfo.checked && <div>
<Switch>

<Route exact path="/" render={() => <Redirect to="/projects" />} />

<DeverseRoute exact path="/login" component={Login} authenticated=
{sessionInfo} />
```


```
</Switch></div>);

//Diverse Route is used for login Route

const DeverseRoute = ({ component: ContentComponent,authenticated,...rest
}) => (
<Route

{...rest}

render={props => (authenticated.sessionInfo ? (

<Redirect to={{pathname: '/projects',state: { from: props.location }}}/>)
: (

<ContentComponent {...props} />))

}/>);
DeverseRoute.propTypes = {component:
PropTypes.func,layout:PropTypes.func,};

DeverseRoute.defaultProps = {component: EmptyComponent,layout:NoLayout};

//LayoutRoute is used to switch main components

const LayoutRoute = ({ component: ContentComponent, layout:
Layout,authenticated ,...rest }) => (<Route{...rest}render={props =>
(authenticated.sessionInfo ? (
<Layout><ContentComponent {...props} /></Layout>
) : (
<Redirect to={{pathname: '/login',state: { from: props.location}}}/>
))}/>);
LayoutRoute.propTypes = {component: PropTypes.func,layout:
PropTypes.func,};

LayoutRoute.defaultProps ={component:EmptyComponent,layout:NoLayout};
```

Thank you !