### Redux Thunk

#### Objectives and Outcomes

Redux Thunk [middleware](https://github.com/reactjs/redux/blob/master/docs/advanced/Middleware.md) allows you to write action creators that return a function instead of an action. In this exercise you will see the use of redux thunk to return a function. At the end of this exercise you will be able to:

* Use Redux Thunk middleware to return a function instead of an action
* Use a logger to print a log of actions initiated on the Redux store.

#### Installing Redux Thunk and Logger

* Install Redux Thunk and Logger as shown below:

yarn add redux-thunk@2.2.0

yarn add redux-logger@3.0.6

* Then open configureStore.js and update it to use the Thunk and Logger as follows:

import {createStore, combineReducers, applyMiddleware } from 'redux';

. . .

import thunk from 'redux-thunk';

import logger from 'redux-logger';

. . .

combineReducers({

dishes: Dishes,

comments: Comments,

promotions: Promotions,

leaders: Leaders

}),

applyMiddleware(thunk, logger)

. . .

* Next, open ActionTypes.js and add new action types as follows:

. . .

export const DISHES\_LOADING = 'DISHES\_LOADING';

export const DISHES\_FAILED = 'DISHES\_FAILED';

export const ADD\_DISHES = 'ADD\_DISHES';

Then open ActionCreators.js and add new actions:

. . .

import { DISHES } from '../shared/dishes';

. . .

export const fetchDishes = () => (dispatch) => {

dispatch(dishesLoading(true));

setTimeout(() => {

dispatch(addDishes(DISHES));

}, 2000);

}

export const dishesLoading = () => ({

type: ActionTypes.DISHES\_LOADING

});

export const dishesFailed = (errmess) => ({

type: ActionTypes.DISHES\_FAILED,

payload: errmess

});

export const addDishes = (dishes) => ({

type: ActionTypes.ADD\_DISHES,

payload: dishes

});

* Next, open dishes.js and add the code to respond to actions as follows:

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

export const Dishes = (state = { isLoading: true,

errMess: null,

dishes:[]}, action) => {

switch (action.type) {

case ActionTypes.ADD\_DISHES:

return {...state, isLoading: false, errMess: null, dishes: action.payload};

case ActionTypes.DISHES\_LOADING:

return {...state, isLoading: true, errMess: null, dishes: []}

case ActionTypes.DISHES\_FAILED:

return {...state, isLoading: false, errMess: action.payload};

default:

return state;

}

};

* Add a new component named LoadingComponent.js to display a loading message as follows:

import React from 'react';

export const Loading = () => {

return(

<div className="col-12">

<span className="fa fa-spinner fa-pulse fa-3x fa-fw text-primary"></span>

<p>Loading . . .</p>

</div>

);

};

* Now we will update the remaining components to use the actions. First, open MainComponent.js and update it as follows:

. . .

import { addComment, fetchDishes } from '../redux/ActionCreators';

. . .

fetchDishes: () => { dispatch(fetchDishes())}

. . .

componentDidMount() {

this.props.fetchDishes();

}

. . .

const HomePage = () => {

return(

<Home

dish={this.props.dishes.dishes.filter((dish) => dish.featured)[0]}

dishesLoading={this.props.dishes.isLoading}

dishesErrMess={this.props.dishes.errMess}

promotion={this.props.promotions.filter((promo) => promo.featured)[0]}

leader={this.props.leaders.filter((leader) => leader.featured)[0]}

/>

);

}

const DishWithId = ({match}) => {

return(

<DishDetail dish={this.props.dishes.dishes.filter((dish) => dish.id === parseInt(match.params.dishId,10))[0]}

isLoading={this.props.dishes.isLoading}

errMess={this.props.dishes.errMess}

comments={this.props.comments.filter((comment) => comment.dishId === parseInt(match.params.dishId,10))}

addComment={this.props.addComment}

/>

);

};

. . .

* Open DishdetailComponent.js and update it as follows:

. . .

import { Loading } from './LoadingComponent';

. . .

if (props.isLoading) {

return(

<div className="container">

<div className="row">

<Loading />

</div>

</div>

);

}

else if (props.errMess) {

return(

<div className="container">

<div className="row">

<h4>{props.errMess}</h4>

</div>

</div>

);

}

else if (props.dish != null)

. . .

* Open HomeComponent.js and update it as follows:

. . .

import { Loading } from './LoadingComponent';

. . .

function RenderCard({item, isLoading, errMess}) {

if (isLoading) {

return(

<Loading />

);

}

else if (errMess) {

return(

<h4>{errMess}</h4>

);

}

else

return(

<Card>

<CardImg src={item.image} alt={item.name} />

<CardBody>

<CardTitle>{item.name}</CardTitle>

{item.designation ? <CardSubtitle>{item.designation}</CardSubtitle> : null }

<CardText>{item.description}</CardText>

</CardBody>

</Card>

);

}

. . .

<RenderCard item={props.dish} isLoading={props.dishesLoading} errMess={props.dishesErrMess} />

. . .

* Finally, update MenuComponent.js as follows:

. . .

import { Loading } from './LoadingComponent';

. . .

function RenderCard({item, isLoading, errMess}) {

if (isLoading) {

return(

<Loading />

);

}

else if (errMess) {

return(

<h4>{errMess}</h4>

);

}

else

return(

<Card>

<CardImg src={item.image} alt={item.name} />

<CardBody>

<CardTitle>{item.name}</CardTitle>

{item.designation ? <CardSubtitle>{item.designation}</CardSubtitle> : null }

<CardText>{item.description}</CardText>

</CardBody>

</Card>

);

}

. . .

<RenderCard item={props.dish} isLoading={props.dishesLoading} errMess={props.dishesErrMess} />

. . .

* Save all the changes and do a Git commit with the message "Redux Thunk".

#### Conclusions

In this exercise we saw the use of Redux Thunk and the Logger.