Exercise (Instructions): Fetch from Server

Objectives and Outcomes

In this exercise you will incorporate Fetch into your React app and then use it to communicate with the REST API server. At the end of this exercise you will be able to:

- Incorporate Fetch into your React app
- Use Fetch to communicate with the REST API server

Fetch

- As a first step, let us install Fetch into our project as follows:

```
yarn add cross-fetch@2.1.0
```

- Now that we have installed Fetch, let us configure your application to connect to the server. First, create a file named *baseUrl.js* in the *shared* folder and add the following to it:

```
export const baseUrl = 'http://localhost:3001/';
```

- Make sure that the json-server is running and serving up the data as illustrated in the previous exercise
- Next, open *ActionTypes.js* and add the following:

```
. . .

import { baseUrl } from '../shared/baseUrl';

. . .

    return fetch(baseUrl + 'dishes')
    .then(response => response.json())
    .then(dishes => dispatch(addDishes(dishes)));

. . .


export const fetchComments = () => (dispatch) => {
    return fetch(baseUrl + 'comments')
    .then(response => response.json())
    .then(comments => dispatch(addComments(comments)));
};

export const commentsFailed = (errmess) => ({
    type: ActionTypes.COMMENTS_FAILED,
    payload: errmess
});

export const addComments = (comments) => ({
    type: ActionTypes.ADD_COMMENTS,
    payload: comments
});

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

    dispatch(promosLoading());

    return fetch(baseUrl + 'promotions')
    .then(response => response.json())
    .then(promos => dispatch(addPromos(promos)));
};

export const addPromos = (promos) => ({
    type: ActionTypes.ADD_PROMOS,
    payload: promos
});
```

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

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

export const Comments = (state = { errMess: null, comments:[]}, action) => {
  switch (action.type) {
    case ActionTypes.ADD_COMMENTS:
      return {...state, errMess: null, comments: action.payload};

    case ActionTypes.COMMENTS_FAILED:
      return {...state, errMess: action.payload};

    case ActionTypes.ADD_COMMENT:
        var comment = action.payload;
        comment.id = state.comments.length;
        comment.date = new Date().toISOString();
        return { ...state, comments: state.comments.concat(comment)};

    default:
      return state;
  }
};
```

- Similarly, open *promotions.js* and update it as follows:

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

        case ActionTypes.PROMOS_FAILED:
            return {...state, isLoading: false, errMess: action.payload};
```

```
        default:
          return state;
      }
};
```

- Now that the Redux actions are all updated, it's time to update the components.
- Open *MainComponent.js* and update it as follows:

```
. . .

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

. . .


const mapDispatchToProps = dispatch => ({
  addComment: (dishId, rating, author, comment) => dispatch(addComment(dishId,
    rating, author, comment)),
  fetchDishes: () => { dispatch(fetchDishes())},
  resetFeedbackForm: () => { dispatch(actions.reset('feedback'))},
  fetchComments: () => dispatch(fetchComments()),
  fetchPromos: () => dispatch(fetchPromos())
});

. . .

  componentDidMount() {
    this.props.fetchDishes();
    this.props.fetchComments();
              promoLoading={this.props.promotions.isLoading}
              promoErrMess={this.props.promotions.errMess}
              leader={this.props.leaders.filter((leader) => leader.featured)[0]}
          />

. . .

          <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.comments.filter((comment) => comment
              .dishId === parseInt(match.params.dishId,10))}
            commentsErrMess={this.props.comments.errMess}
            addComment={this.props.addComment}
          />

. . .
```

- Then, open *MenuComponent.js* and update it as follows:

```
. . .

import { baseUrl } from '../shared/baseUrl';

. . .

                    <CardImg width="100%" src={baseUrl + dish.image} alt={dish
                      .name} />

. . .
```

```
. . .

                    <RenderCard item={props.promotion} isLoading={props
                      .promoLoading} errMess={props.promoErrMess} />

. . .
```

- Then, open *DishdetailComponent.js* and update it as follows:

```
. . .

import { baseUrl } from '../shared/baseUrl';

. . .

                <CardImg top src={baseUrl + dish.image} alt={dish.name} />

. . .
```

- Save all the changes and do a Git commit with the message "Fetch from Server".

Conclusions

In this exercise you have learnt to install Fetch and use it communicate with the server.

Mark as completed