

Redux thunk est un middleware qui se positionne entre les actions et les reducers, pour permettre de gérer les cas asynchrones du type requête axios.

PROBLÈMES


SORTIE

CONSOLE DE DÉBOGAGE

TERMINAL

```
MBP-de-Matt:React_demographie matt$ npm install redux-thunk
```

## Flux asynchrone

 Modifier sur GitHub

Dernière mise à jour il y a 3 mois

Sans [middleware](#), le magasin Redux prend uniquement en charge le [flux de données synchrone](#). C'est ce que vous obtenez par défaut avec `createStore()`.

Vous pouvez améliorer avec `applyMiddleware()`. Ce n'est pas obligatoire, mais cela vous permet d'[exprimer des actions asynchrones de manière pratique](#).

Le middleware asynchrone tel que [redux-thunk](#) ou [redux-promise](#) enveloppe la méthode du magasin et vous permet d'envoyer autre chose que des actions, par exemple, des fonctions ou des promesses. Tout middleware que vous utilisez peut alors interpréter tout ce que vous envoyez, et à son tour, peut transmettre des actions au middleware suivant de la chaîne. Par exemple, un intergiciel Promise peut intercepter des promesses et envoyer une paire d'actions de début / de fin de manière asynchrone en réponse à chaque promesse. `dispatch()`.

Lorsque le dernier intergiciel de la chaîne distribue une action, il doit s'agir d'un objet ordinaire. C'est à ce moment que le [flux de données Redux synchrone](#) a lieu.

Consultez le [code source complet pour l'exemple asynchrone](#).

On ajoute l'import thunk et on le passe en argument de la fonction applyMiddleware()

```
JS index.js ●
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import registerServiceWorker from './registerServiceWorker';
4
5  //----Import de composant
6  import App from './components/App';
7
8  //----Import de reducers
9  import reducers from './reducers';
10
11 //----Import Redux
12 import { Provider } from 'react-redux';
13 import { createStore, applyMiddleware } from 'redux';
14
15 //----Import thunk
16 import thunk from 'redux-thunk';
17
18
19
20 const createStoreWithMiddleWare = applyMiddleware(thunk)(createStore);
21
22 ReactDOM.render(
23   <Provider store={createStoreWithMiddleWare(reducers, window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__())}>
24     <App />
25   </Provider>,
26   document.querySelector('.container'));
27   registerServiceWorker();
28
```

On ne peut pas faire nos requête comme cela car redux ne gère pas les actions asynchrones donc il n'attendra pas le retour de la requête axios ce qui nous retournera une erreur

```
JS index.js •
1 import axios from "axios";
2
3 export const GET_COUNTRIES = 'GET_COUNTRIES';
4 export const ERROR_GET_COUNTRIES = 'ERROR_GET_COUNTRIES';
5
6
7 export function getCountries(){
8
9     axios("http://api.population.io:80/1.0/countries").then(function(response){
10         return ({type:GET_COUNTRIES, payload:response.data.countries});
11     }).catch(function(error){
12         return ({type:ERROR_GET_COUNTRIES, error:error.response.data.detail});
13     })
14
15 }
```

KO

Grâce à thunk on on retourne une fonction qui prend en paramètre un dispatch qui se chargera de le transmettre au reducer concerner.

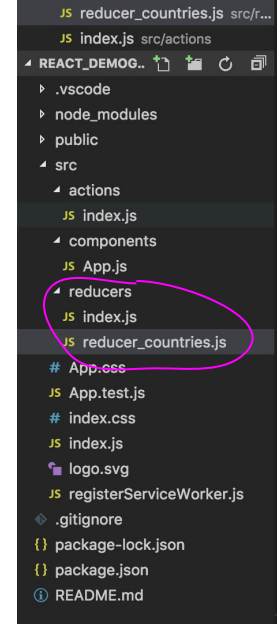
Le fait de retourner une fonction cela force redux a attendre la fin du déroulement de la fonction, le temps que la requête se fait et qu'elle nous fasse son retour.

```
JS index.js x
1 import axios from "axios";
2
3 export const GET_COUNTRIES = 'GET_COUNTRIES';
4 export const ERROR_GET_COUNTRIES = 'ERROR_GET_COUNTRIES';
5
6
7 export function getCountries(){
8     return function (dispatch){
9         axios("http://api.population.io:80/1.0/countries").then(function(response){
10             dispatch ({type:GET_COUNTRIES, payload:response.data.countries});
11         }).catch(function(error){
12             dispatch ({type:ERROR_GET_COUNTRIES, error:error.response.data.detail});
13         })
14     }
15 }
```

OK

On implémente le reducer et on paramètre la connection du reducer avec le store(state) de redux.

```
JS index.js .../reducers JS reducer_countries.js x JS index.js .../actions
1 import {GET_COUNTRIES,ERROR_GET_COUNTRIES} from '../actions/index';
2
3 export default function(state=null,action){
4   switch(action.type){
5     case GET_COUNTRIES :
6       return action.payload
7     case ERROR_GET_COUNTRIES :
8       return action.errors
9     default:
10      return state
11   }
12 }
13 |
```



```
JS index.js .../reducers x JS reducer_countries.js JS index.js .../actions
1 import { combineReducers } from 'redux';
2 import reducer_Countries from './reducer_countries';
3
4 const rootReducer = combineReducers({
5   countries : reducer_Countries
6 });
7
8 export default rootReducer;
9 |
```

On veut connecter notre composant Search\_bar au store pour avoir accès au state des pays.

import {connect} pour connecter react a redux.

import {bindActionCreators} pour connecter les actions aux reducers.

import {getCountries} qui est notre action"fonction".

```
JS Search_bar.js x JS index.js
1 import React, { Component } from 'react';
2 import {connect} from "react-redux";
3 import {bindActionCreators} from "redux";
4 import {getCountries} from "../actions/index";
5
6 class Search_Bar extends Component {
7   render () {
8     return (
9       <div>
10
11       </div>
12     )
13   }
14 }
15
16 export default connect(mapStateToProps, mapDispatchToProps)(Search_Bar);
```

On finit la connection de notre composant...

**componentWillMount()** nous permet de réaliser des opérations avant que notre composant soit rendu

**mapStateToProps(state)** nous permet de donner au composant accès aux props du store Redux.

**mapDispatchToProps(dispatch)** permet de donner au composant accès au dispatch d'une action que l'on a définies dans les actions et importer dans notre composant.

```
JS App.js x JS Search_bar.js x
1 import React, { Component } from 'react';
2 import {connect} from "react-redux";
3 import {bindActionCreators} from "redux";
4 import {_getCountries} from "../actions/index";
5
6 class Search_Bar extends Component {
7
8   componentWillMount(){
9     this.props._getCountries();
10  }
11
12  render () {
13    return (
14      <div>
15
16      </div>
17    )
18  }
19 }
20
21 const mapStateToProps = (state) =>{
22   return{
23     countries : state.countries
24   }
25 }
26
27 const mapDispatchToProps = (dispatch) =>{
28   return bindActionCreators({_getCountries},dispatch);
29 }
30
31 export default connect(mapStateToProps, mapDispatchToProps)(Search_Bar);
```

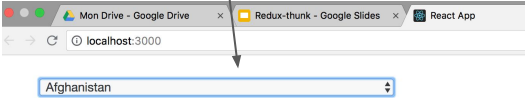
Note importante:

Dans un composant connecté au store, on peut grâce au destructeur ES6 récupérer une props du store dont on a besoin.

Et ensuite effectuer le traitement que l'on souhaite dessus.

Rappel on a accès au props grâce la fonctions mapStateToProps(state).

Notre search\_bar



```
JS Search_bar.js • JS App.js
1  import React, { Component } from 'react';
2  import {connect} from "react-redux";
3  import {bindActionCreators} from "redux";
4  import {getCountries} from "../actions/index";
5
6  //----Import Style
7  import '../style/search_bar.css';
8
9  class Search_Bar extends Component {
10
11    componentWillMount(){
12      this.props._getCountries();
13    }
14
15    //NOTE: fonction qui retourne un composant, qui est appelle dans le render.
16    renderSelectBox(){
17      const {countries} = this.props;
18      if (countries){
19        return (
20          <select className="col-lg-10 input-group">
21            {
22              countries.map((country) => {
23                return <option key={country} value={country}>{country}</option>
24              })
25            }
26          </select>
27        )
28      }else{
29        return <div>No country found</div>
30      }
31    }
32
33    render () {
34      return (
35        <div className="search_bar">
36          {this.renderSelectBox()}
37        </div>
38      )
39    }
40  }
41
42  const mapStateToProps = (state) =>{
43    return{
44      countries : state.countries
45    }
46  }
47
```

On veut que notre search bar nous montre par défaut la France comme pays selectionner



Dans App notre composant container principal, on définit une constante avec la valeur que l'on souhaite, on la passe en propriété du composant `<Search_Bar/>`

```
JS App.js      JS Search_bar.js
1  //----Import composant react
2  import React, { Component } from 'react';
3
4  //----Import composant
5  import Search_Bar from '../container/Search_bar';
6
7
8  const DEFAULT_COUNTRY = "France";
9  class App extends Component {
10   render() {
11     return (
12       <div>
13         <Search_Bar defaultCountry={DEFAULT_COUNTRY}/>
14       </div>
15     );
16   }
17 }
18
19 export default App;
20
```



Dans le constructeur de notre Search\_bar on définit le state selectedCountry avec la valeur du props que l'on a passé en propriété du composant <Search\_Bar> dans App.js.

On utilise l'eventHandler OnChange(oEvt) pour pouvoir changer la valeur de la searchBar.

On Implémente donc la méthode search(oEvt).

Et l'eventHandler onChange fournis l'événement qui permet de valoriser le state a chaque nouveau pays choisie.

```
JS App.js
JS Search_bar.js
4 import {getCountries} from '../actions/index';
5
6 //----Import Style
7 import '../style/search_bar.css';
8
9 class Search_Bar extends Component {
10
11   constructor(props){
12     super(props)
13     this.state = {selectedCountry:this.props.defaultCountry}
14   }
15
16   componentWillMount(){
17     this.props._getCountries();
18   }
19
20   //NOTE: fonction qui retourne un composant, qui est appelle dans le render.
21   renderSelectBox(){
22     const {countries} = this.props;
23     if (countries){
24       return (
25         <select value={this.state.selectedCountry} onChange={(e) => this.search(e)} className="col-lg-10 input-group">
26           {
27             countries.map((country) => {
28               return <option key={country} value={country}>{country}</option>
29             })
30           }
31         </select>
32       )
33     }else{
34       return <div>No country found</div>
35     }
36   }
37
38   search(e){
39     this.setState({selectedCountry:e.target.value})
40     //TODO: faire fonction qui lance la rechche pour obtenir un tableau de state
41   }
42
43   render () {
44     return (
45       <div className="search_bar">
46         {this.renderSelectBox()}
47       </div>
48     )
49   }
50 }
```

A partir de là on a juste sortie les actions dans un fichier à part pour que soit plus propre.  
Donc pour utiliser on fera logiquement ACTION.GET\_COUNTRIES...etc

JS index.js

JS action.js ×

```
1  const ACTION = {  
2      GET_COUNTRIES : 'GET_COUNTRIES',  
3      ERROR_GET_COUNTRIES : 'ERROR_GET_COUNTRIES',  
4      GET_MORTALITY : 'GET_MORTALITY'  
5  }  
6  
7  
8  export default ACTION;
```

Maintenant on veut récupérer des données pour faire connaître le taux de mortalité chez les hommes et les femmes de 25 ans.

On crée donc une nouvelle action qui consiste à aller chercher de la data.

On imbrique une requête axios dans la première et on la dispatch.

la fonction prend en paramètre un country que l'on fournira par la suite a l'appelle de l'action.

```
JS index.js x
1  import axios from 'axios';
2  import {ACTION} from './action';
3
4  const API_END_POINT = "http://api.population.io:80/1.0/";
5  const DEFAULT_PARAM = "25/today";
6
7  export const GET_COUNTRIES = 'GET_COUNTRIES';
8  export const ERROR_GET_COUNTRIES = 'ERROR_GET_COUNTRIES';
9  export const GET_MORTALITY = 'GET_MORTALITY';
10
11  //NOTE: Recupere les pays de l'API
12  export function _getCountries(){
13    return function (dispatch){
14      axios(`${API_END_POINT}countries`).then(function(response){
15        dispatch ({type:ACTION.GET_COUNTRIES, payload:response.data.countries});
16      }).catch(function(error){
17        dispatch ({type:ACTION.ERROR_GET_COUNTRIES, errors:error.response.data.detail});
18      })
19    }
20  }
21
22  //NOTE: Recupere le taux de mortalité dans l'API
23  export function _getMortality(country){
24    return function (dispatch){
25      return axios(`${API_END_POINT}mortality-distribution/${country}/male/${DEFAULT_PARAM}`).then((responseMale) => {
26        axios(`${API_END_POINT}mortality-distribution/${country}/female/${DEFAULT_PARAM}`).then((responseFemale) => {
27          console.log(responseMale, "responseMale");
28          console.log(responseFemale, "responseFemale");
29          dispatch(
30            {
31              //NOTE: voir ACTION.GET_MORATLITY
32              type : ACTION.GET_MORTALITY,
33              payload : {
34                country : country,
35                male : responseMale.data.mortality_distribution,
36                female : responseFemale.data.mortality_distribution
37              }
38            }
39          ).catch(function(error){
40            console.log(error, "ERROR");
41          });
42        });
43      });
44    }
45  }
```

Dans Search\_bar on importe la nouvelle action `_getMortality` et l'ajoute bien a `mapDispatchToProps` pour pouvoir utiliser notre action dans notre composant

```
JS index.js JS Search_bar.js x
1 import React, { Component } from 'react';
2 import {connect} from "react-redux";
3 import {bindActionCreators} from "redux";
4 import {_getCountries,_getMortality} from "../actions/index";
5
6 //----Import Style
7 import '../style/search_bar.css';
8
9 class Search_Bar extends Component {
10
11   constructor(props){
12     super(props)
13     this.state = {selectedCountry:this.props.defaultCountry}
14   }
15
16   componentWillMount(){
17     this.props._getCountries();
```

```
  }
18
19   const mapStateToProps = (state) =>{
20     return{
21       countries : state.countries
22     }
23   }
24
25   const mapDispatchToProps = (dispatch) =>{
26     return bindActionCreators({_getCountries, _getMortality},dispatch);
27   }
28
29   export default connect(mapStateToProps, mapDispatchToProps)(Search_Bar);
```

On crée le reducer\_mortality et on le paramètre dans reducers/index.js

```
JS reducer_mortality.js • JS index.js
1  import {ACTION} from '../actions/action';
2
3  export default function (state = [], action){
4    switch(action.type){
5      case ACTION.GET_MORTALITY:
6        return [
7          ...state,{ |
8            country : action.payload.country,
9            male: action.payload.male,
10           female : action.payload.female
11         ]
12       }
13     default:
14       return state;
15   }
16 }
```

```
JS reducer_mortality.js • JS index.js ✕
1  import { combineReducers } from 'redux';
2  import reducer_countries from './reducer_countries';
3  import reducer_mortality from './reducer_mortality';
4
5  const rootReducer = combineReducers({
6    countries : reducer_countries,
7    mortality : reducer_mortality
8  });
9
10 export default rootReducer;
11
```

## Création du composant Flag

Notre composant prend des paramètres

```
JS App.js JS mortality_list.js JS mortality_list_item.js JS flag.js x
1 import React from 'react'
2
3 const URL_BASE = "http://www.sciencekids.co.nz/images/pictures/flags680/"
4
5 const Flag = ({country,className}) => {
6   return (
7     <div>
8       <img className={className} src={` ${URL_BASE}${country}.jpg`}/>
9     </div>
10   )
11 }
12
13 export default Flag
```

En appelant notre composant on lui passe des arguments en "dur"

```
JS App.js JS mortality_list.js JS mortality_list_item.js x JS flag.js
1 import React from 'react'
2
3 //---- Import style
4 import '../style/mortality_List_Item.css';
5
6 //----Import composant
7 import Flag from './flag';
8
9
10 const Mortality_List_Item = () => {
11   return (
12     <div>
13       <Flag country={"France"} className="flag_medium"/>
14     </div>
15   )
16 }
17
18 export default Mortality_List_Item
```

On passe ensuite notre dumb composant <Mortality\_List\_Item> a <Mortality\_list> et pour finir on passe notre <Mortality\_list> a App.js.

```
JS App.js JS mortality_list.js x JS mortality_list_item.js JS flag.js
1 import React, { Component } from 'react'
2 import Mortality_List_Item from '../components/mortality_list_item';
3
4
5 class Mortality_list extends Component {
6   render () {
7     return (
8       <div>
9         <Mortality_List_Item/>
10      </div>
11    )
12  }
13 }
14
15 export default Mortality_list
```

```
JS App.js x JS mortality_list.js JS mortality_list_item.js JS flag.js
1 //---Import composant react
2 import React, { Component } from 'react';
3
4 //---Import composant
5 import Search_Bar from '../container/Search_bar';
6 import Mortality_list from '../container/mortality_list';
7
8 const DEFAULT_COUNTRY = "France";
9 class App extends Component {
10   render() {
11     return (
12       <div>
13         <Search_Bar defaultCountry={DEFAULT_COUNTRY}/>
14         <Mortality_list/>
15       </div>
16     );
17   }
18 }
19
20 export default App;
21
```

```

JS App.js x JS mortality_list.js JS mortality_list_item.js JS index.js
1 //---Import composant react
2 import React, { Component } from 'react';
3
4 //---Import composant
5 import Search_Bar from '../container/Search_bar';
6 import Mortality_list from '../container/mortality_list';
7
8 const DEFAULT_COUNTRY = "France";
9 class App extends Component {
10   render() {
11     return (
12       <div>
13         <Search_Bar defaultCountry={DEFAULT_COUNTRY}/>
14         <Mortality_list defaultCountry={DEFAULT_COUNTRY}/>
15       </div>
16     );
17   }
18 }
19
20 export default App;
21

```

```

JS App.js JS mortality_list.js x JS mortality_list_item.js JS index.js
1 import React, { Component } from 'react'
2 import Mortality_List_Item from '../components/mortality_list_item';
3 import {connect} from 'react-redux';
4 import {bindActionCreators} from "redux";
5 import {getMortality} from '../actions/index';
6
7
8 class Mortality_list extends Component {
9
10   componentWillMount(){
11     this.props._getMortality(this.props.defaultCountry)
12   }
13
14   renderMortalities(){
15     const {mortalities} = this.props;
16     return mortalities.map((data)=>{
17       return <Mortality_List_Item key={data.country} mortality={data}/>
18     })
19

```

```

JS App.js JS mortality_list.js JS mortality_list_item.js x JS index.js
1 import React from 'react'
2 //--- Import style
3 import '../style/mortality_List_Item.css';
4
5 //--- Import composant
6 import Flag from './flag';
7
8 //--- Import ReactChartkick
9 import ReactChartkick, { ColumnChart } from 'react-chartkick';
10 import Chart from 'chart.js';
11 ReactChartkick.addAdapter(Chart)
12
13 const xTitle = "Age";
14 const yTitle = "% Mortalité";
15
16 const Mortality_List_Item = (props) => {
17   const formattedDataMale = formatMortalityData(props.mortality.male);
18   const formattedDataFemale = formatMortalityData(props.mortality.female);
19
20   return (
21     <tr>
22       <td><Flag country={props.mortality.country} className="flag_medium"/></td>
23       <td className="col-md-6"><ColumnChart xtitle=(xTitle) ytitle=(yTitle) data={formattedDataMale} max={35}/></td>
24       <td className="col-md-6"><ColumnChart xtitle=(xTitle) ytitle=(yTitle) data={formattedDataFemale} max={35}/></td>
25     </tr>
26   )
27 }
28
29 function formatMortalityData(mortality){
30   const filteredData = mortality.filter((data) => {
31     if(data.age >=101){
32       return false;
33     }else{
34       return data;
35     }
36   });
37
38   const array = filteredData.map((data) => {
39     if(data.age <=101){
40       return [Number(data.age.toFixed(0)), Number(data.mortality_percent).toFixed(0)]
41     }
42   });
43   return array;
44 }
45
46 export default Mortality_List_Item;

```