Setting Up The Middleware · redux-observable

Setting Up The Middleware

Now that we know what Epics are, we need to provide them to the redux-observable middleware so they can start listening for actions.

Root Epic

Just like redux requiring a single root Reducer, redux-observable also requires you to have a single root Epic. As we learned previously, we can use `combineEpics()` to accomplish this.

One common pattern is to import all your Epics into a single file, which then exports the root Epic, along with your root Reducer.

redux/modules/root.js

```
import { combineEpics } from 'redux-observable';
import ping, { pingEpic } from './ping';
import users, { fetchUserEpic } from './users';

export const rootEpic = combineEpics(
  pingEpic,
  fetchUserEpic
);

export const rootReducer = combineReducers({
  ping,
  users
});
```

Configuring The Store

Now you'll want to create an instance of the redux-observable middleware, passing along our newly created root Epic.

```
import { createEpicMiddleware } from 'redux-observable';
import { rootEpic } from './modules/root';
```

```
const epicMiddleware = createEpicMiddleware(rootEpic);
```

When you put that together with your existing Store configuration, it will look something like this:

redux/configureStore.js

```
import { createStore, applyMiddleware } from 'redux';
import { createEpicMiddleware } from 'redux-observable';
import { rootEpic, rootReducer } from './modules/root';

const epicMiddleware = createEpicMiddleware(rootEpic);

export default function configureStore() {
  const store = createStore(
    rootReducer,
    applyMiddleware(epicMiddleware)
  );


  return store;
}
```

Redux DevTools

If you're using the Redux DevTools Extension, you'll need to invoke `window.devToolsExtension.updateStore(store)`, otherwise your Epics will not receive any actions you dispatch using the DevTools UI.

```
const epicMiddleware = createEpicMiddleware(pingEpic);

const store = createStore(pingReducer,
  compose(
    applyMiddleware(epicMiddleware),
    window.devToolsExtension ? window.devToolsExtension() : f => f
  )
);

if (window.devToolsExtension) {
  window.devToolsExtension.updateStore(store);
}
```

This is required to get around a limitation of Redux, but is planned to be fixed soon.