Setting Up Redux-ORM

Okay, so loading two fields was pretty easy. It's time to start actually working out what the application's data model will look like.

As described previously, the goal of this sample application is to build a miniature version of the Battletech MekHQ game tool. **We want to be able to track Battlemech Pilots assigned to a combat unit, which specific Battlemech each Pilot is assigned to, and ultimately organize Battlemechs and their Pilots into groups called "Lances" and "Companies"**.

We're going to use the Redux-ORM library to help define what our data types are and how they relate to each other. Those Model types can then be used to help us load data into our store, and query and update that data.

Before we can use Redux-ORM, we need to add the package:

```
yarn add redux-orm@0.9.4
```

That will download Redux-ORM and its other dependencies. Yarn will update the `package.json` and `yarn.lock` files, and should also add the package tarballs to the `./offline-mirror` folder we created earlier. We should commit all of those.

> **Commit 8b90d93: Add Redux-ORM package**

Next, we need to **create a Redux-ORM `ORM` instance, and a parent reducer for all of our entity data**. Since this code isn't specific to a single feature, we'll add it underneath the `app` folder.

Commit efacb84: Set up Redux-ORM and add an initial entities reducer

app/orm.js

```
import {ORM} from "redux-orm";

const orm = new ORM();

export default orm;
```

app/reducers/entitiesReducer.js

```
import {createReducer} from "common/utils/reducerUtils";

import orm from "app/orm"

const initialState = orm.getEmptyState()

export default createReducer(initialState, {
});
```

app/reducers/rootReducer.js

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

+import entitiesReducer from "./entitiesReducer";
import tabReducer from "features/tabs/tabsReducer";
import unitInfoReducer from "features/unitInfo/unitInfoReducer";

const rootReducer = combineReducers({
+   entities : entitiesReducer,
    unitInfo : unitInfoReducer,
    tabs : tabReducer,
});
```

If we look at our overall app state now using the Redux DevTools, we'll see `entities: {}`. That's because we haven't added any Model types to the ORM instance.

The first Model type we should add is for pilots. We'll keep it *really* simple - just a Pilot class, with no relations, and add that to our ORM instance. There's no specific rule for where Model classes should be defined, so for the moment we're going to define them inside the relevant feature folders.

> **Commit decdbea: Add initial Redux-ORM Pilot model**

features/pilots/Pilot.js

```
import {Model} from "redux-orm";

export default class Pilot extends Model {
    static modelName = "Pilot";
}
```

app/orm.js

```
import {ORM} from "redux-orm";

+import Pilot from "features/pilots/Pilot";

const orm = new ORM();

+orm.register(Pilot);

export default orm;
```

Now that we have the Pilot model class registered with the ORM instance, we can go back to the Redux DevTools and see that the generated initial state for our `entities` slice reducer now includes an empty "table" for the Pilot type: