Displaying a List of Pilots

Now that we have pilots added to the store, we can update our `<Pilots>` and `<PilotsList>` components to display them. Since we already had `<Pilots>` acting as a container component, all we really need to do is connect it to the store, return an array of plain JS pilot objects as a prop, and switch from using the sample entry we had stored in its state to the array from props.

To do this, we're going to import the singleton Redux-ORM `ORM` instance we created. In our `mapState` function, we'll create a `Session` instance from the current set of "tables" in our state, and use the `Pilot` class we defined to query those tables. We'll retrieve a list of the actual JS objects, and return those as a prop.

Commit 1f17b50: Connect the Pilots component to render a list of pilots from the store

features/pilots/Pilots.jsx

```
// Omit imports not relevant to this commit

const mapState = (state) => {
    // Create a Redux-ORM Session from our "entities" slice, which
    // contains the "tables" for each model type
    const session =  orm.session(state.entities);

    // Retrieve the model class that we need.  Each Session
    // specifically "binds" model classes to itself, so that
    // updates to model instances are applied to that session.
    // These "bound classes" are available as fields in the sesssion.
    const {Pilot} = session;

    // Query the session for all Pilot instances.
    // The QuerySet that is returned from all() can be used to
    // retrieve instances of the Pilot class, or retrieve the
    // plain JS objects that are actually in the store.
```

```
    // The toRefArray() method will give us an array of the
    // plain JS objects for each item in the QuerySet.

    const pilots = Pilot.all().toRefArray();

    // Now that we have an array of all pilot objects, return it as a prop
    return {pilots};
}


export class Pilots extends Component {
    render() {
        const {pilots = []} = this.props;

        // Use the first pilot as the "current" one for display, if availa
ble.
        const currentPilot = pilots[0] || {};

        // Omit rendering code, which didn't change
    }
}

export default connect(mapState)(Pilots);
```

And with that, we finally have something useful and interesting displayed on screen!