Managing Collections with Entity Adapter

Duncan Hunter

Lead Frontend Developer

@dunchunter | duncanhunter.com.au

Module Introduction

Why use entity adapter?

Add @ngrx/entity

Use entity adapter:

- Interface and collection methods

- Selectors

- Update types in actions

Entity State adapter are for managing record collections.

Why Entity Adapter

Reduce boilerplate

Performant CRUD operations

Type-safe selectors

What is an Entity?

```
export interface Product {
 id: number;
 name: string;
 price: number;
}
```

What is an Entity?

```
export interface Product {
 id: number;
 name: string;
 price: number;
}

const products: Product[] = [
 { id: 1, price: 3, name: ‘Hammer’ },
 { id: 2, price: 2, name: ‘Nails’ },
 { id: 3, price: 4, name: ‘Spanner’ }
];
```

Products State

```
export interface ProductsState {
 showProductCode: boolean;
 loading: boolean;
 errorMessage: string;
 products: Product[];
}
```

Current Code - Selectors

```
export const selectProductById = createSelector(
 selectProducts,
 selectRouteParams,
 (products, { id }) => products.find((product) => product.id === parseInt(id))
);
```

Current Code - Reducer

```
on(ProductsAPIActions.productUpdatedSuccess, (state, { product }) => ({
  ...state,
  loading: false,
  products: state.products.map((existingProduct) =>
  existingProduct.id === product.id ? product : existingProduct
 ),
})),
```

Products Dictionary vs Array

```ts
export interface ProductsState {
 showProductCode: boolean;
 loading: boolean;
 errorMessage: string;
 products: Product[];
}
```

```ts
export interface ProductsState {
 showProductCode: boolean;
 loading: boolean;
 errorMessage: string;
 ids: string[] | number[];
 entities: Dictionary<Product>;
}
```

Products Dictionary vs Array

```ts
export interface ProductsState {
 showProductCode: boolean;
 loading: boolean;
 errorMessage: string;
 products: Product[];
}
```

```ts
export interface ProductsState {
 showProductCode: boolean;
 loading: boolean;
 errorMessage: string;
 ids: string[] | number[];
 entities: Dictionary<Product>;
}
```

Entity Interfaces - EntityState<T>

```
export interface ProductsState extends EntityState<Product> {
 showProductCode: boolean;
 loading: boolean;
 errorMessage: string;
}
```

Entity Interfaces - EntityState<T>

```
export interface ProductsState extends EntityState<Product> {
 showProductCode: boolean;
 loading: boolean;
 errorMessage: string;
 // ids: string[] | number[];
 // entities: Dictionary<Product>;
}
```

Entity Adapter – createEntityAdapter

```
import { createEntityAdapter, EntityAdapter } from ‘@ngrx/entity’;

export const adapter: EntityAdapter<Product> = createEntityAdapter<Product>({});
```

Entity Adapter – createEntityAdapter

```
import { createEntityAdapter, EntityAdapter } from ‘@ngrx/entity’;

export const adapter: EntityAdapter<Product> = createEntityAdapter<Product>({
 selectId,
 sortByName
});
```

Entity Adapter – createEntityAdapter

```
import { createEntityAdapter, EntityAdapter } from ‘@ngrx/entity’;

export const adapter: EntityAdapter<Product> = createEntityAdapter<Product>({
 selectId,
 sortByName
});

export function selectId(product: Product): string {
 return product.productId;
};
```

Entity Adapter – createEntityAdapter

```
import { createEntityAdapter, EntityAdapter } from ‘@ngrx/entity’;

export const adapter: EntityAdapter<Product> = createEntityAdapter<Product>({
 selectId,
 sortByName
});

export function selectId(product: Product): string {
 return product.productId;
};

export function sortByName(a: Product, b: Product): number {
 return a.name.localeCompare(b.name);
};
```

Entity Adapter – getInitalState

```
const intitialState: ProductsState = adapter.getInitialState({
 showProductCode: true,
 loading: false,
 errorMessage: ''
});
```

Entity Adapter – getInitalState

```
const intitialState: ProductsState = adapter.getInitialState({
 showProductCode: true,
 loading: false,
 errorMessage: '',
 // ids: [],
 // entities: {}
});
```

Current Code - Reducer

```
on(ProductsAPIActions.productUpdatedSuccess, (state, { product }) => ({
  ...state,
  loading: false,
  products: state.products.map((existingProduct) =>
  existingProduct.id === product.id ? product : existingProduct
 ),
})),
```

With EntityAdapter - Reducers

```
on(ProductsAPIActions.productUpdatedSuccess, (state, { update }) =>
 adapter.updateOne(update)
)
```

With EntityAdapter - Reducer

```
on(ProductsAPIActions.productUpdatedSuccess, (state, { update }) => ({
 adapter.updateOne(update, {
  ...state,
  loading: false,
}))
```

EntityAdapter - Collection Methods

Adapter Method	Description
addOne	Add one entity to the collection.
addMany	Add multiple entities to the collection.
setAll	Replace current collection with provided collection.
setOne	Add or Replace one entity in the collection.
setMany	Add or Replace multiple entities in the collection.
removeOne	Remove one entity from the collection.
removeMany	Remove multiple entities from the collection, by id or by predicate.
removeAll	Clear entity collection.
updateOne	Update one entity in the collection. Supports partial updates.
updateMany	Update multiple entities in the collection. Supports partial updates.
upsertOne	Add or Update one entity in the collection.
upsertMany	Add or Update multiple entities in the collection.
mapOne	Update one entity in the collection by defining a map function.
map	Update multiple entities in the collection by defining a map function.

Current Code - Selectors

```
export const selectProductById = createSelector(
 selectRouteParams,
 selectProducts,
 (products, { id }) => products.find((product) => product.id === parseInt(id))
);
```

EntityAdapter - Selectors

```
const {
 selectAll,
 selectEntities,
 selectIds,
 selectTotal
} = adapter.getSelectors();
```

EntityAdapter - Selectors

```
const {
 selectAll,
 selectEntities,
 selectIds,
 selectTotal
} = adapter.getSelectors();

export const selectAllProducts = selectAll;
export const selectProductEntities = selectEntities;
export const selectProductIds = selectIds;
export const selectProductTotal = selectTotal;
```

Current Code - Selectors

```
export const selectProductById = createSelector(
 selectRouteParams,
 selectProducts,
 (products, { id }) => products.find((product) => product.id === parseInt(id))
);
```

With EntityAdapter - Selectors

```
export const selectProductById = createSelector(
 selectRouteParams,
 selectProductEntities,
 ({ id }, entities) => entities[id])
);
```

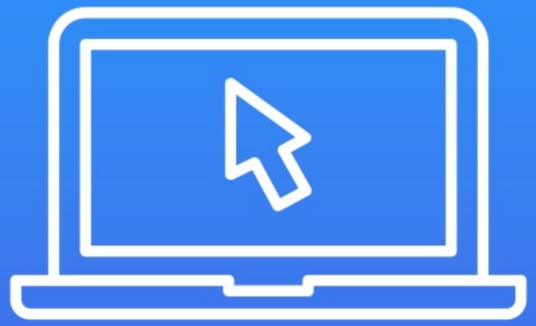
Current Code - Actions

```
export const ProductsAPIActions = createActionGroup({
 source: ‘Products API’,
 events: {
  ‘Product Updated Success’: props<{ product: Product }>(),
 },
});
```

With EntityAdapter – Action Updates

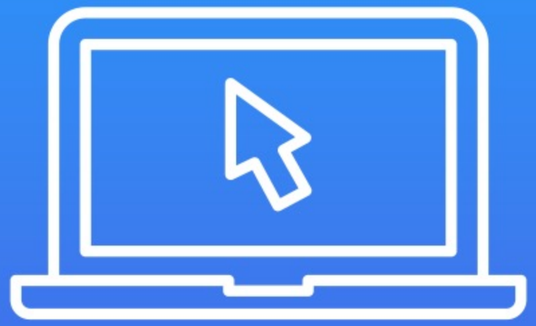
```
import { Update } from '@ngrx/entity';

export const ProductsAPIActions = createActionGroup({
 source: 'Products API',
 events: {
  'Product Updated Success': props<{ update: Update<Product> }>(),
 },
});
```

Demo



Add and use ngrx/entity in reducers

Demo



@ngrx/entity selectors