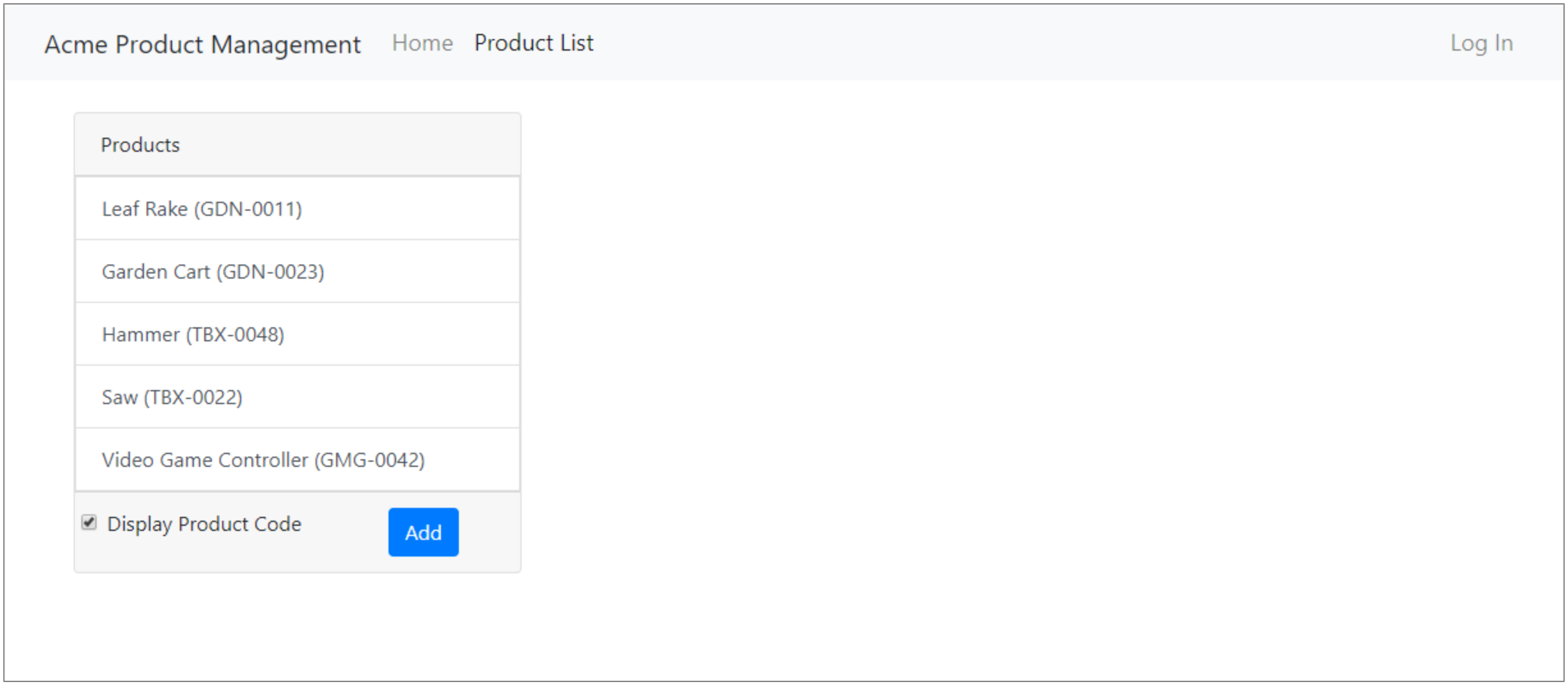
Performing Update Operations

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata blogs.msmvps.com/deborahk/


I CAN'T
DO IT


Acme Product Management
Home
Product List
Log In
Products
Leaf Rake (GDN-0011)
Garden Cart (GDN-0023)
Hammer (TBX-0048)
Saw (TBX-0022)
Video Game Controller (GMG-0042)
Display Product Code
Add

Acme Product Management Home Product List Log In

Products

- Leaf Rake (GDN-0011)
- Garden Cart (GDN-0023)
- Hammer (TBX-0048)
- Saw (TBX-0022)
- Video Game Controller (GMG-0042)

Display Product Code Add

Add Product

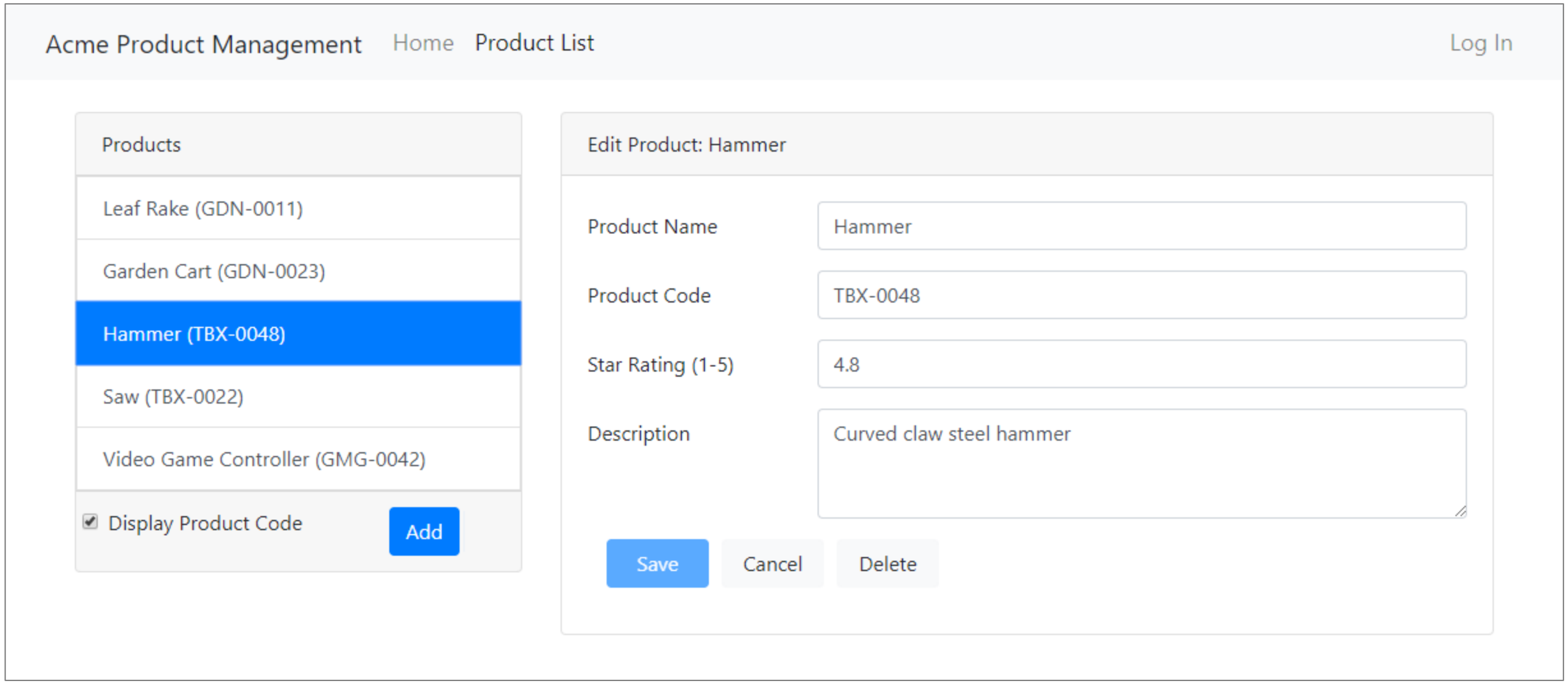
Product Name: Name (required)

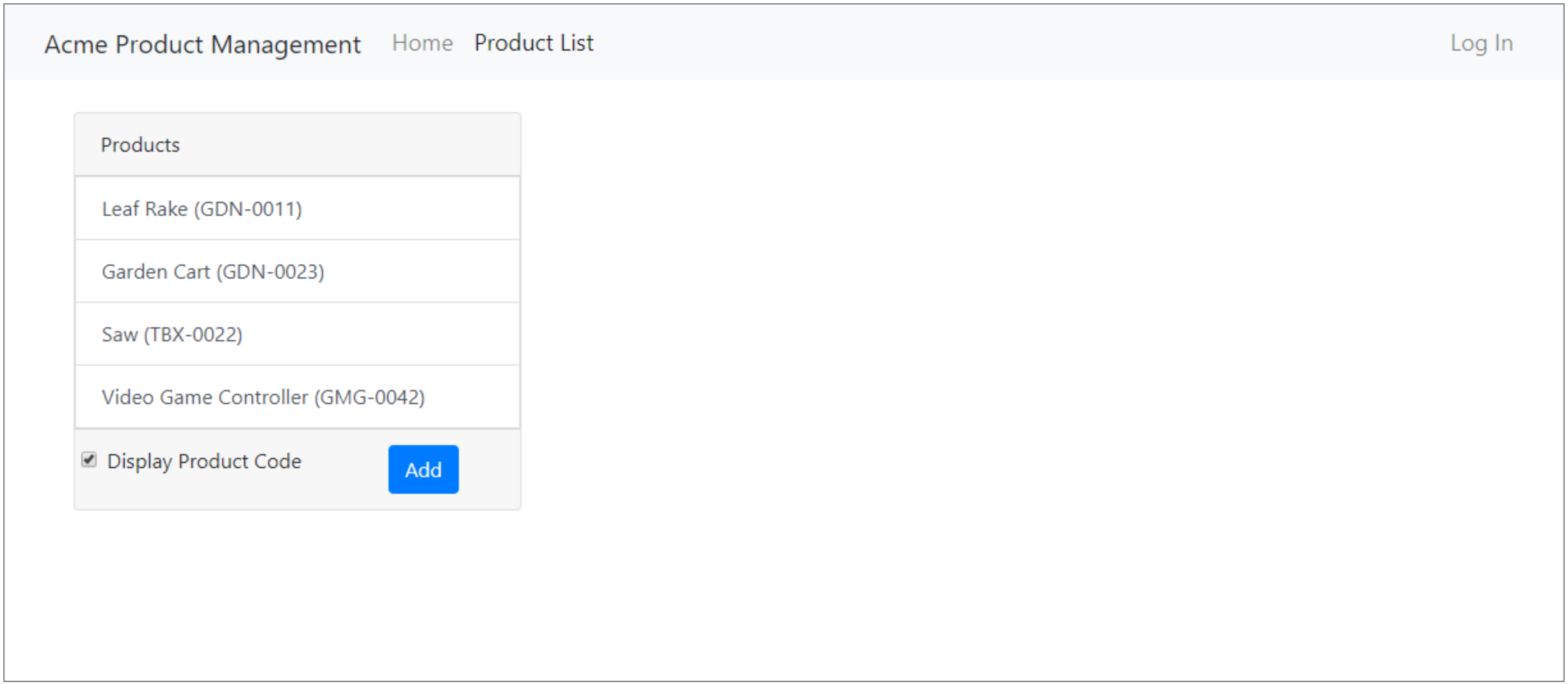
Product Code: Code (required)

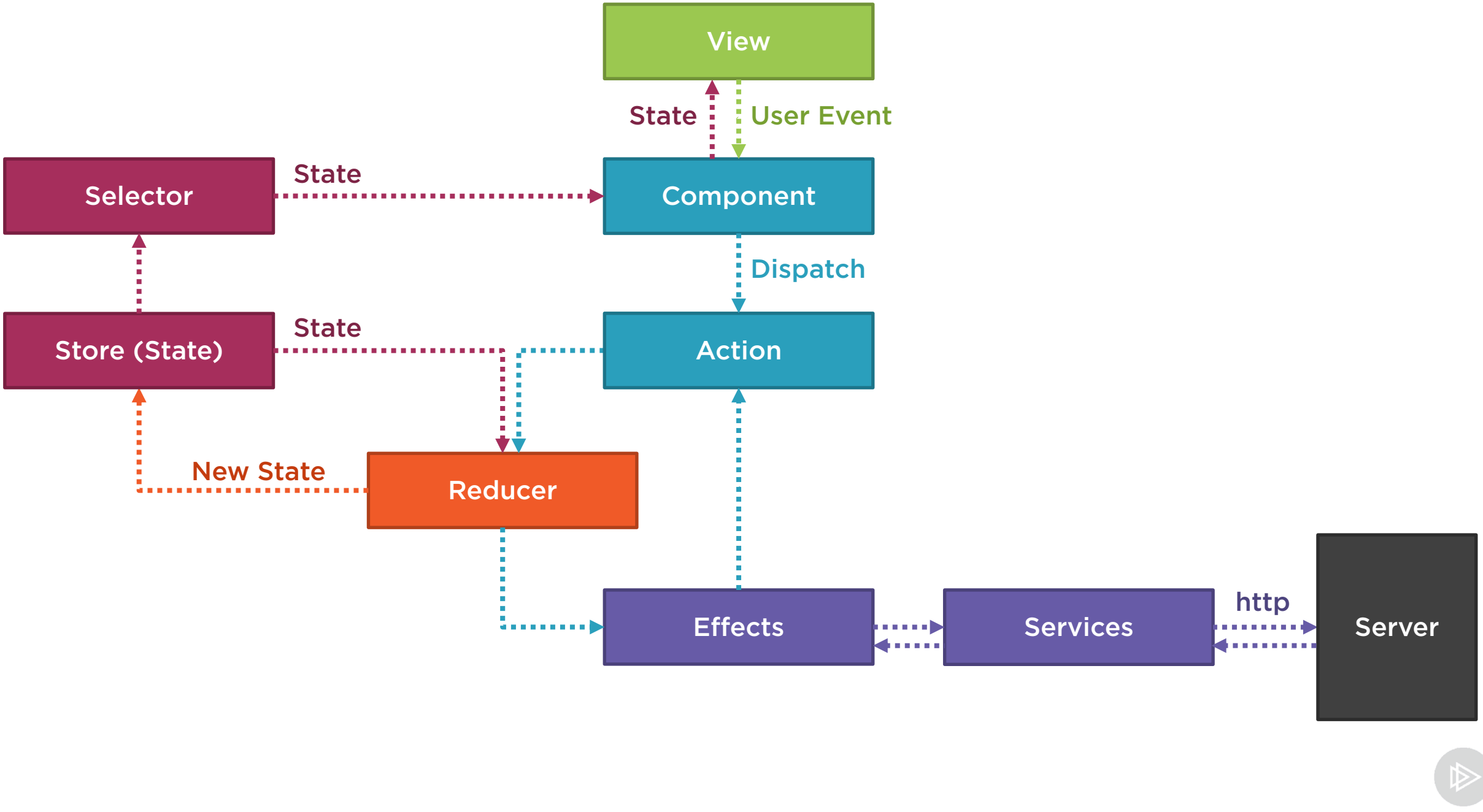
Star Rating (1-5): 0

Description: Description

Save Cancel Delete


Acme Product Management
Home
Product List
Log In
Products
Leaf Rake (GDN-0011)
Garden Cart (GDN-0023)
Hammer (TBX-0048)
Saw (TBX-0022)
Video Game Controller (GMG-0042)
Display Product Code
Add
Edit Product: Hammer
Product Name
Hammer
Product Code
TBX-0048
Star Rating (1-5)
4.8
Description
Curved claw steel hammer
Save
Cancel
Delete

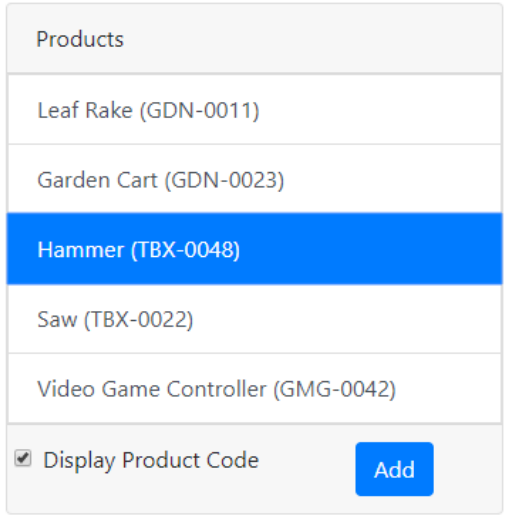

Acme Product Management
Home
Product List
Log In
Products
Leaf Rake (GDN-0011)
Garden Cart (GDN-0023)
Saw (TBX-0022)
Video Game Controller (GMG-0042)
Display Product Code
Add


View
State
User Event
Selector
State
Component
Dispatch
Store (State)
State
Action
New State
Reducer
Effects
Services
http
Server

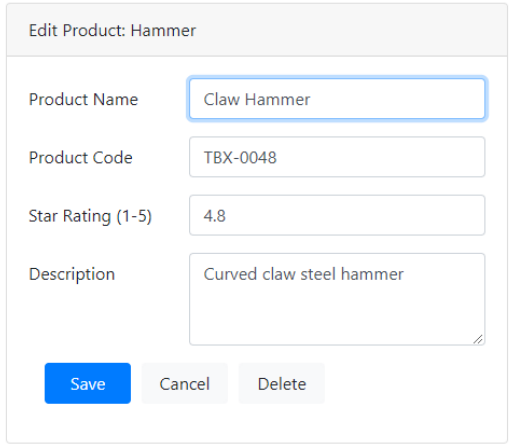
Module Overview

- Identify the state and actions
- Strongly type the state and build selectors
- Strongly type the actions with action creators
- Dispatch an action
- Build the effect to process the action
- Process the success and fail actions

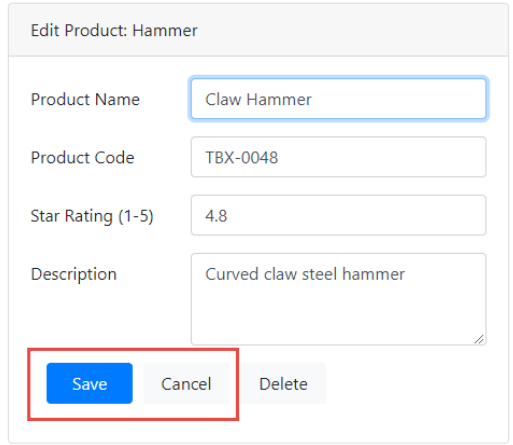
Goal: Update a Product



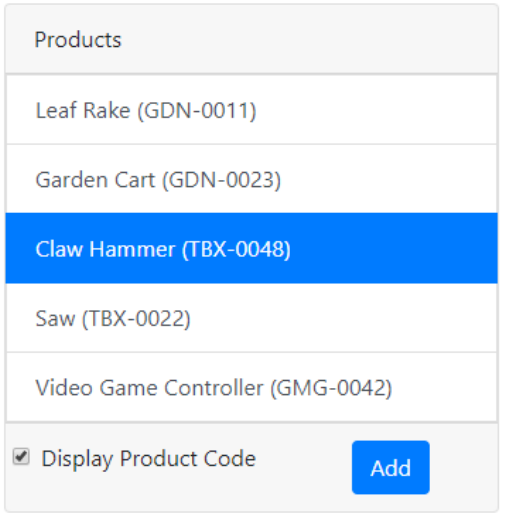
Select a product



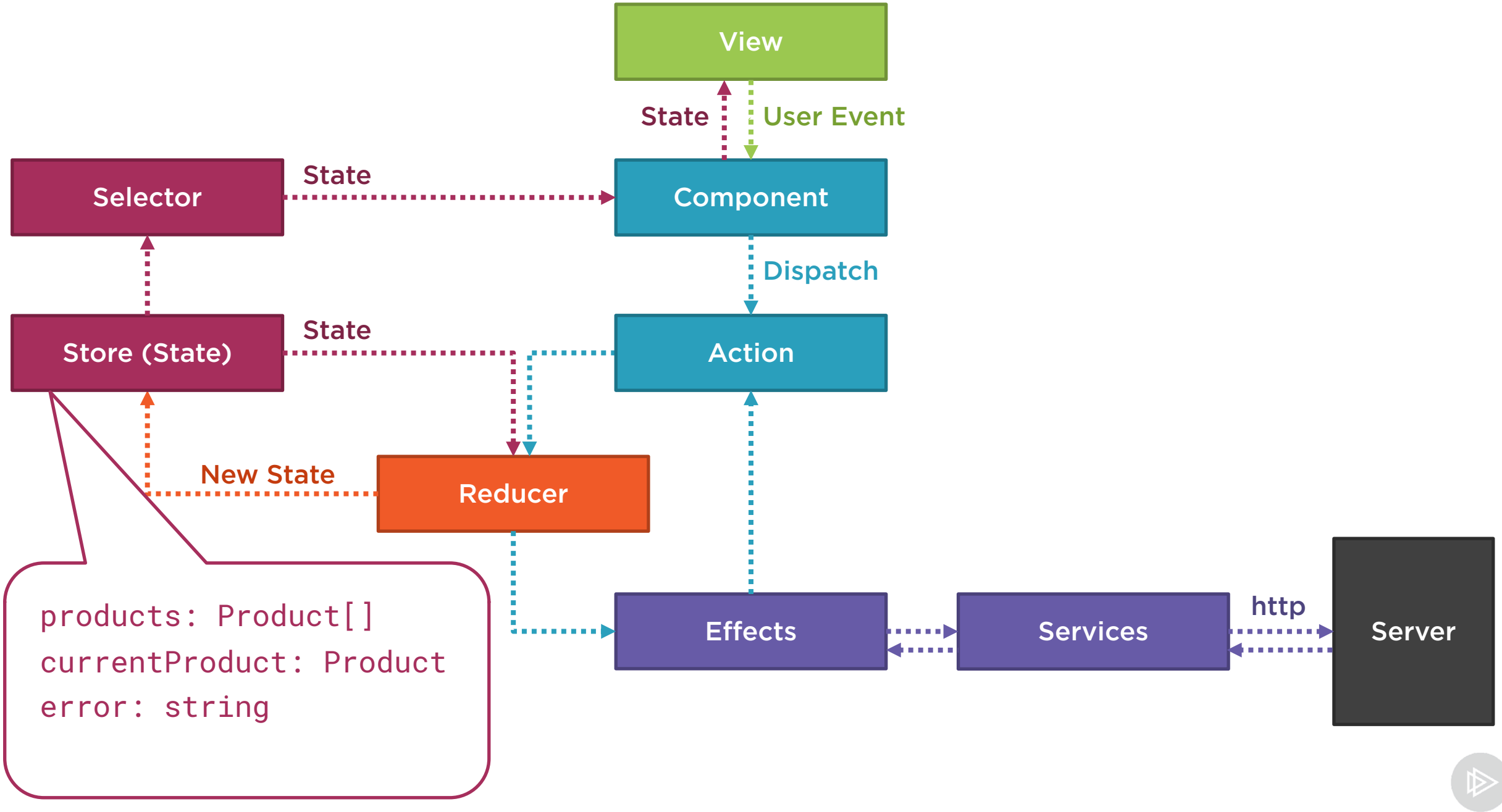
Edit its properties

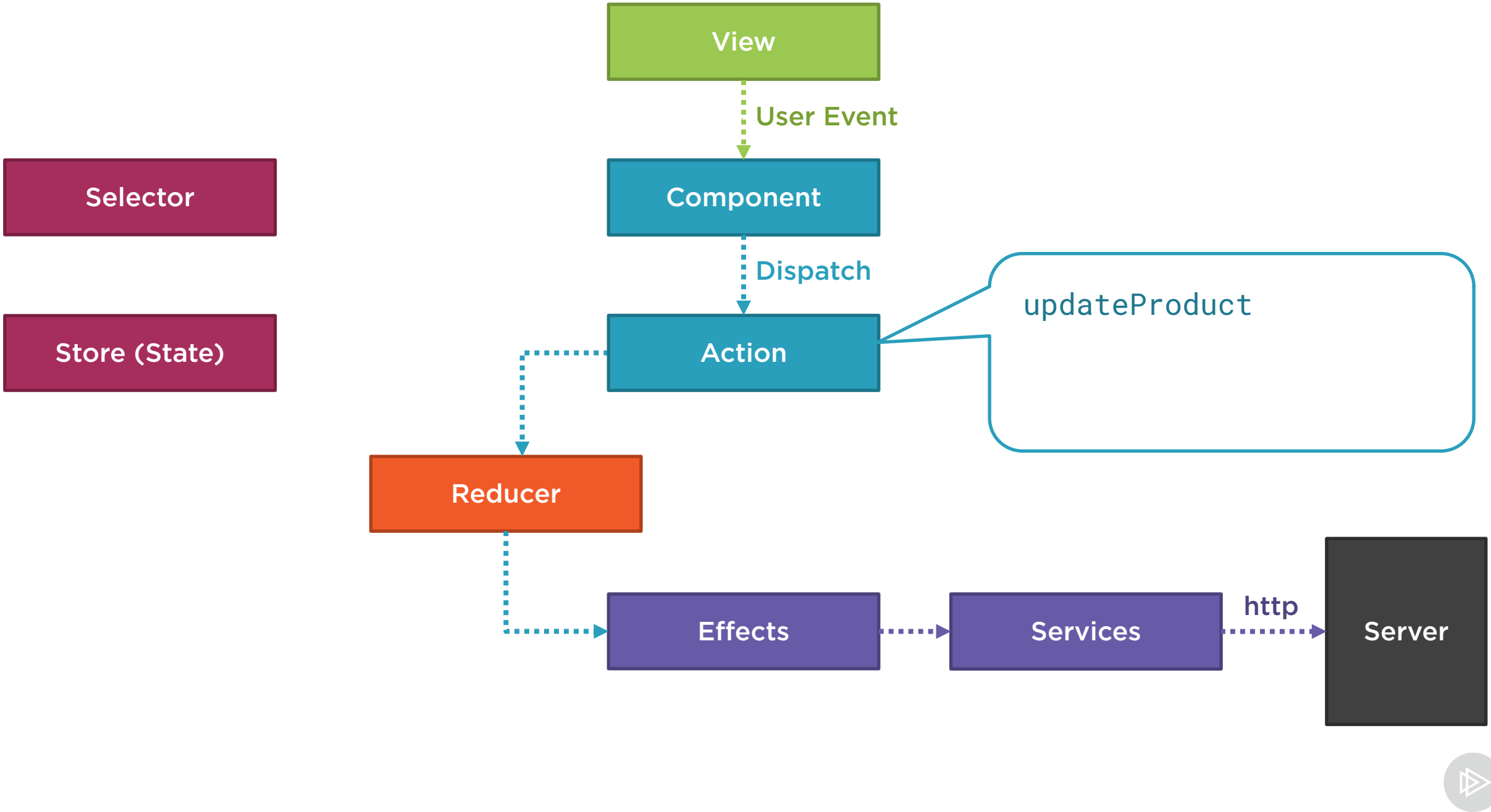


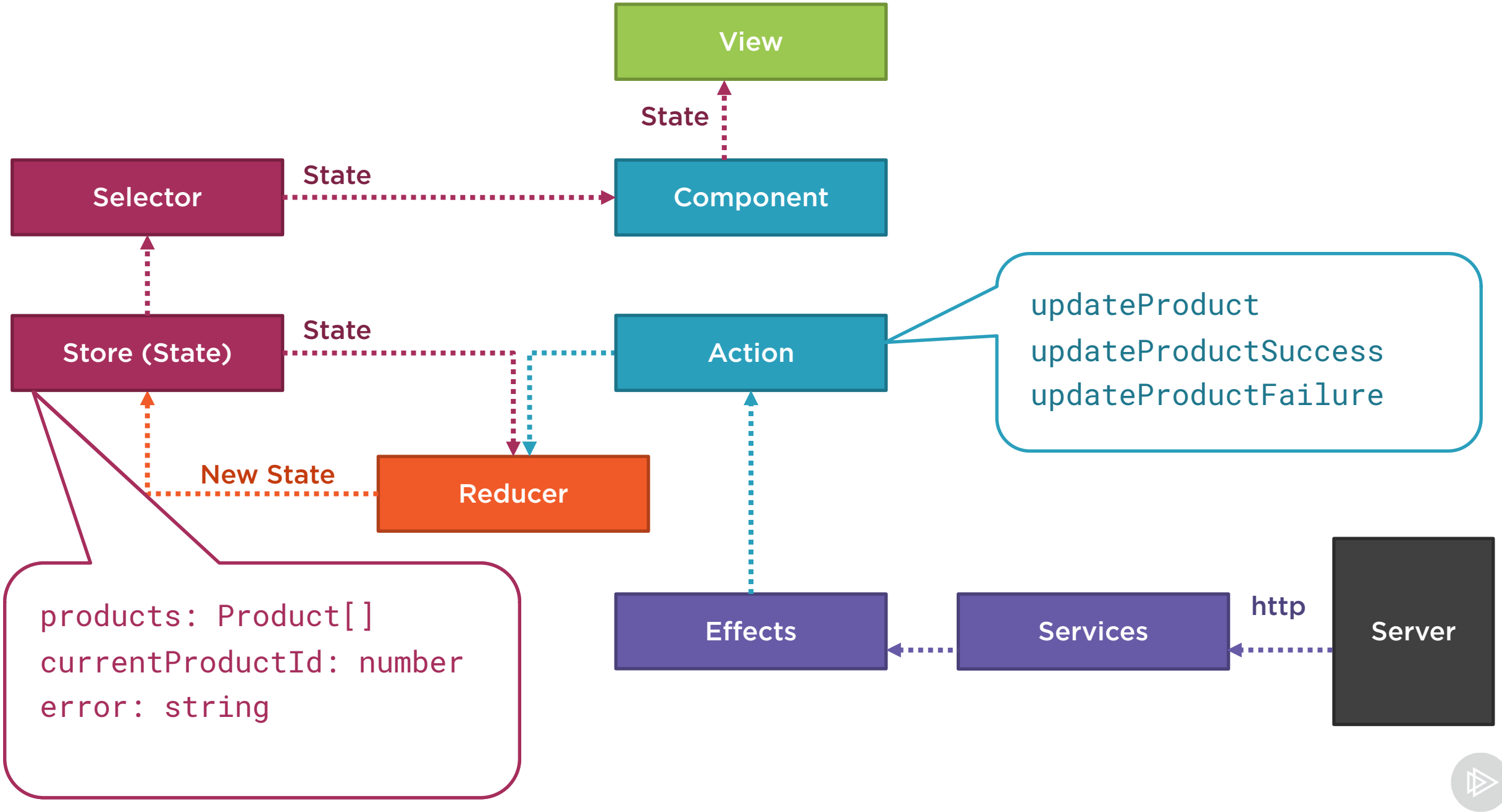
Save or cancel



Display the updated product


View
State
User Event
Selector
State
Component
Dispatch
Store (State)
State
Action
New State
Reducer
products: Product[]
currentProduct: Product
error: string
Effects
Services
http
Server


View
User Event
Component
Dispatch
Selector
Store (State)
Action
updateProduct
Reducer
Effects
Services
http
Server


View
State
Selector
State
Component
Store (State)
State
Action
updateProduct
updateProductSuccess
updateProductFailure
New State
Reducer
products: Product[]
currentProductId: number
error: string
Effects
Services
http
Server

Defining the (Strongly Typed) State

Define an interface

```
export interface ProductState {
 showProductCode: boolean;
 currentProductId: number;
 products: Product[];
}
```

Set initial values

```
const initialState: ProductState = {
 showProductCode: true,
 currentProductId: null,
 products: []
};
```

Build selectors

```
export const getProductId = createSelector(
 getProductFeatureState,
 state => state.currentProductId
);
```

Demo

Defining the (strongly typed) state

Defining the (Strongly Typed) Actions

Define action for the operation

```
export const updateProduct = createAction(
  '[Product] Update Product',
  props<{ product: Product }>()
);
```

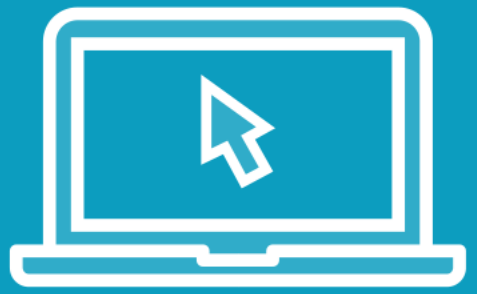
Define action for success

```
export const updateProductSuccess = createAction(
  '[Product] Update Product Success',
  props<{ product: Product }>()
);
```

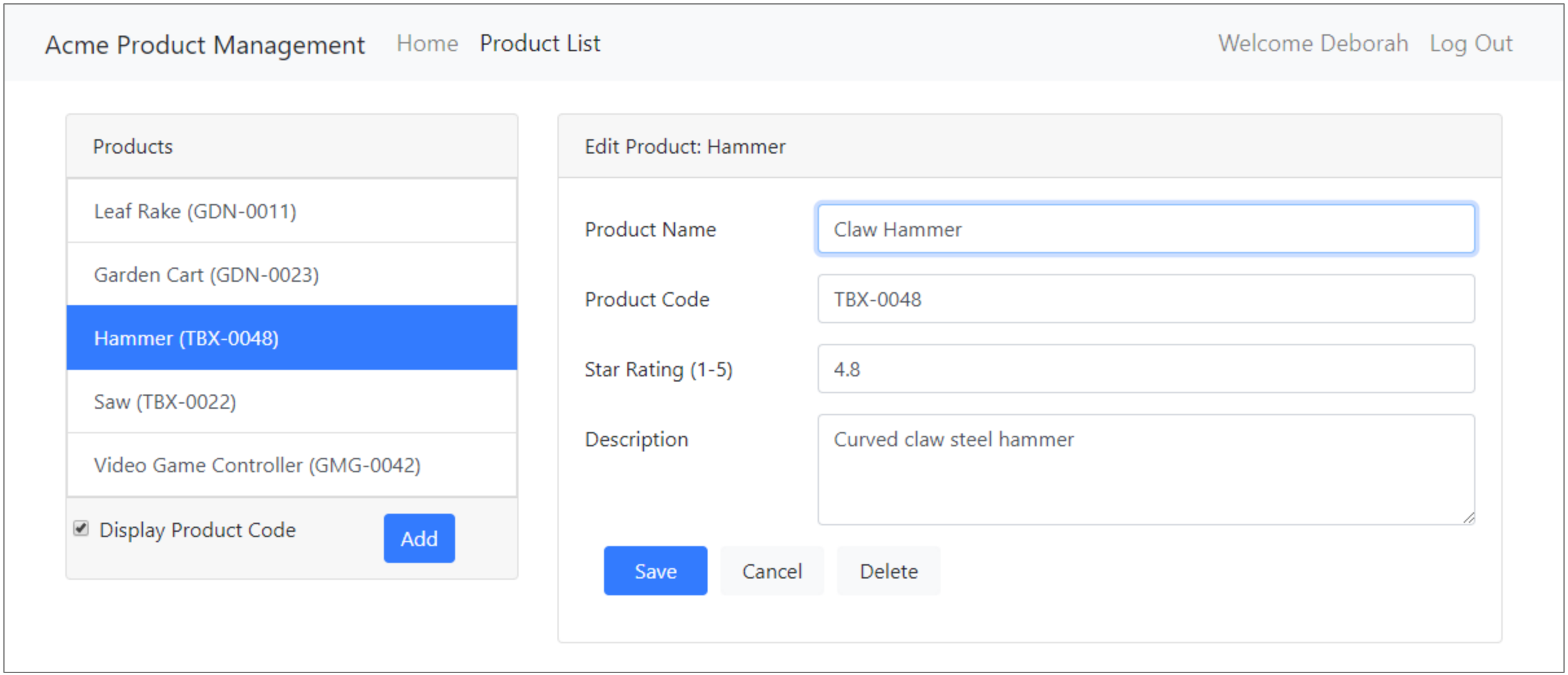
Define action for failure

```
export const updateProductFailure = createAction(
  '[Product] Update Product Fail',
  props<{ error: string }>()
);
```

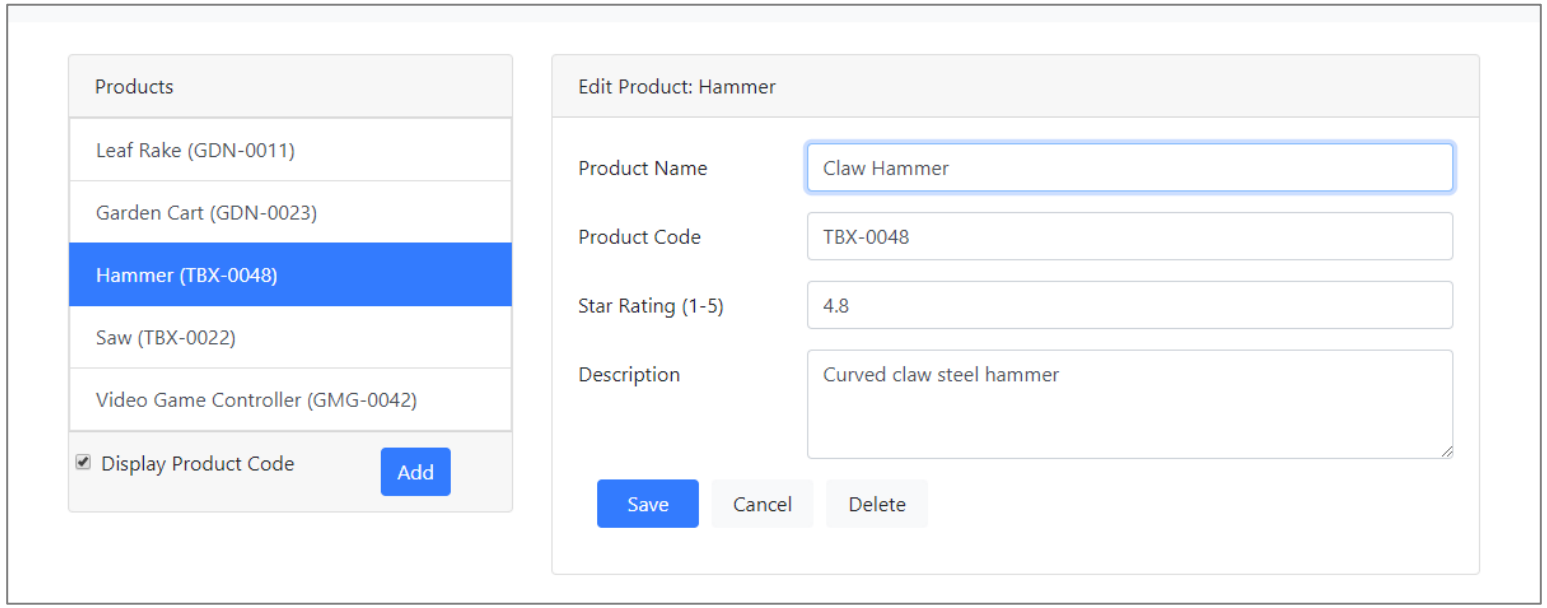
Demo



Defining the (strongly typed) actions


Acme Product Management
Home
Product List
Welcome Deborah
Log Out
Products
Leaf Rake (GDN-0011)
Garden Cart (GDN-0023)
Hammer (TBX-0048)
Saw (TBX-0022)
Video Game Controller (GMG-0042)
Display Product Code
Add
Edit Product: Hammer
Product Name
Claw Hammer
Product Code
TBX-0048
Star Rating (1-5)
4.8
Description
Curved claw steel hammer
Save
Cancel
Delete

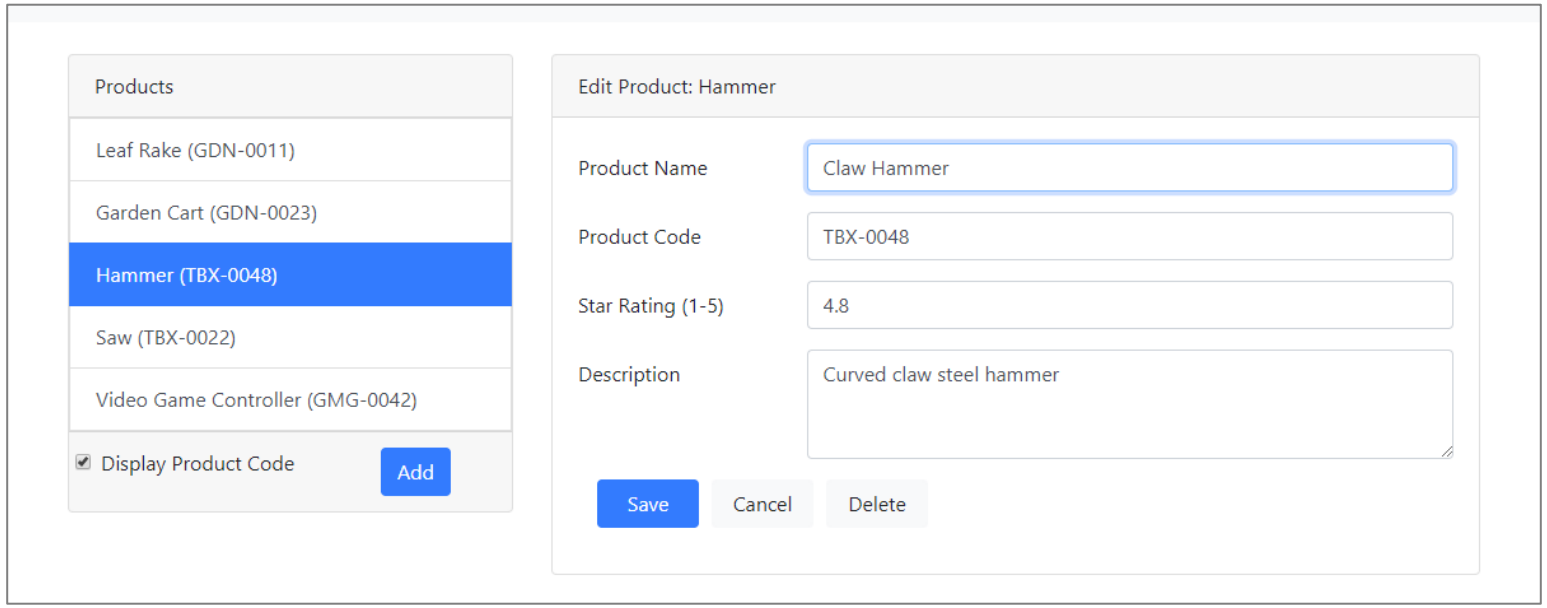
Template-driven Forms




```
<input class="form-control"
       id="productNameId"
       type="text"
       placeholder="Name (required)"
       required
       minlength="3"
       [(ngModel)]="product.productName"
       name="productName"
       #nameVar="ngModel" />
```

```
export class ProductEditComponent {
 pageTitle = 'Product Edit';
 errorMessage = '';

 product: Product;
 ...
}
```

Reactive Forms




```
this.productForm = this.fb.group({
  productName: ['', [Validators.required,
Validators.maxLength(50)]],
  productCode: ['', Validators.required],
  starRating: ['', NumberValidators.range(1, 5)],
  description: ''
});
```

Angular Reactive Forms

Dispatching an Action

Inject the store

```
import { Store } from '@ngrx/store';

constructor(private store: Store<State>) { }
```

Import the actions

```
import * as ProductActions from '../state/product.actions';
```

Call the dispatch method

```
this.store.dispatch(ProductActions.updateProduct({ product }));
```

Demo

Dispatching an action

Building the Effect

Build effect service and inject Actions

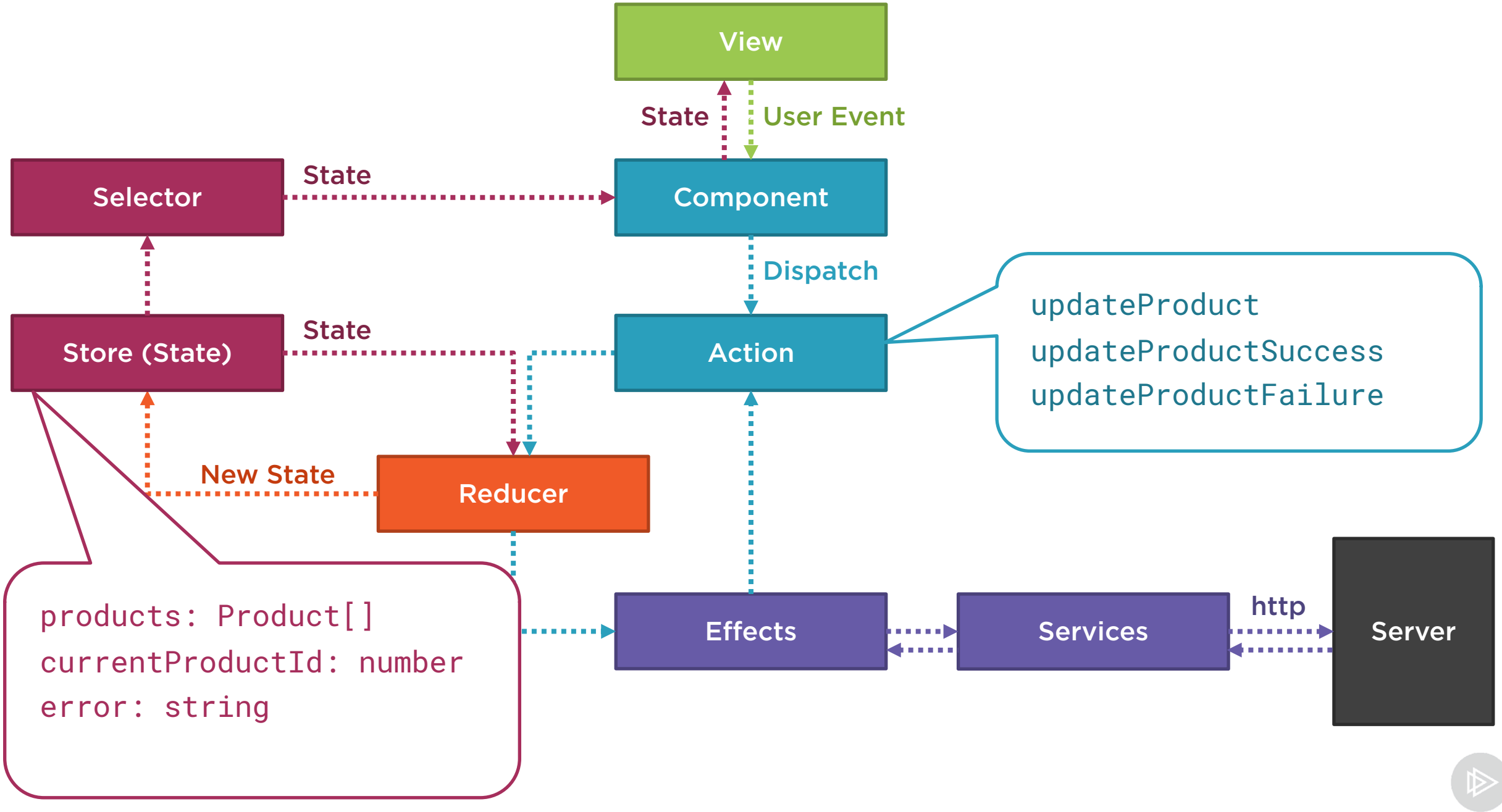
Define a property

Build the effect

```
@Injectable()
export class ProductEffects {

 constructor(private actions$: Actions) { }
}


updateProduct$ = createEffect(() => {

 return this.actions$.pipe(
  ofType(ProductActions.updateProduct),
  concatMap(action =>
   this.productService.updateProduct(action.product).pipe(
     map(product => ProductActions.updateProductSuccess({ product })),
     catchError(error => of(ProductActions.updateProductFailure({ error }))))
 ));
});
```

Demo

Building the effect


View
State
User Event
Selector
State
Component
Dispatch
Store (State)
State
Action
updateProduct
updateProductSuccess
updateProductFailure
New State
Reducer
products: Product[]
currentProductId: number
error: string
Effects
Services
http
Server

Processing the Success and Fail Actions

Process the success action

```
on(ProductActions.updateProductSuccess, (state, action) => {
  const updatedProducts = state.products.map(
    item => action.product.id === item.id ? action.product : item);
...
```

3. Claw Hammer

Original Array

1. Leaf Rake
2. Garden Cart
3. Hammer
4. Saw
5. Controller

New Array

3. Claw Hammer

Processing the Success and Fail Actions

Process the success action

Return the new state

Process the failure action and return new state

```
on(ProductActions.updateProductSuccess, (state, action) => {
  const updatedProducts = state.products.map(
    item => action.product.id === item.id ? action.product : item);


  return {
     ...state, products: updatedProducts,
     currentProductId: action.product.id, error: ''
  };
}),
on(ProductActions.updateProductFailure, (state, action)  => {
    return {
      ...state, error: action.error
    };
}),
```

Immutable vs. Mutable Array Methods

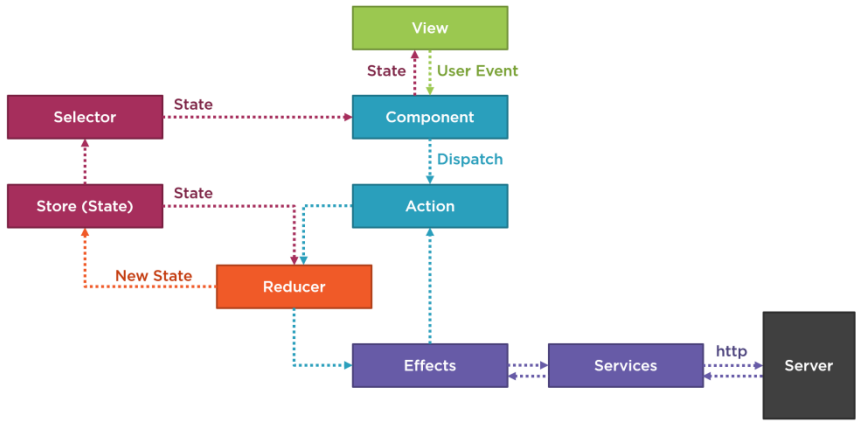
An immutable object or array cannot be modified after it is created.

`state.products.push(action.product)`	Mutable
`state.products.concat(action.product)`	Immutable
`[...state.products, action.product]`	Immutable
`state.products.shift()`	Mutable
`state.products.splice(0,2)`	Mutable
`state.products.filter(p => p.id !== action.product.id)`	Immutable
`state.products.map(p => p.id === action.product.id ? action.product : p)`	Immutable
`state.products.forEach(p => p.id === action.product.id ? action.product : p)`	Mutable

Demo

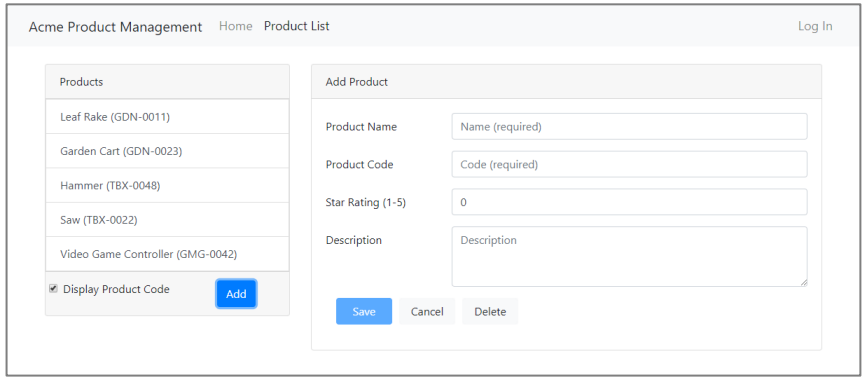
Processing the success and fail actions

Checklist:
Performing Operations with Side Effects



- Identify the state and actions
- Strongly type the state and build selectors
- Strongly type the actions with action creators
- Dispatch an action to kickoff the operation
- Build the effect to perform the operation and dispatch a success or fail action
- Process the success and fail actions in the reducer

Homework



- Identify the state and actions
- Define a state interface and selectors
- Build action creators
- Dispatch an action to kick off the operation
- Build the effect to process that action and dispatch the success and fail actions
- Process the success and fail actions in the reducer

https://github.com/DeborahK/Angular-NgRx-GettingStarted/tree/master/APM-Demo4