ListApp1 Details

Thursday, August 13, 2020 9:32 AM

Introduction

ListApp1 is an Angular Web application that is the basis for exploring NgRx in conjunction with this NgRx tutorial.

This simple application maintains lists and in this implementation provides an editable list of favorite ice cream flavors.

It consists of two pages; one being the home page, which contains a list of available lists, and another page which is the list of favorite ice cream flavors.

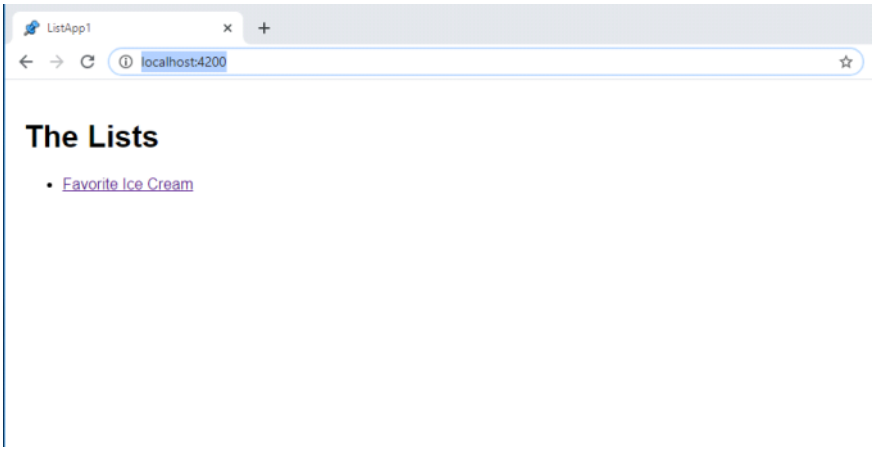


Figure 1 - Home Page showing the available lists.

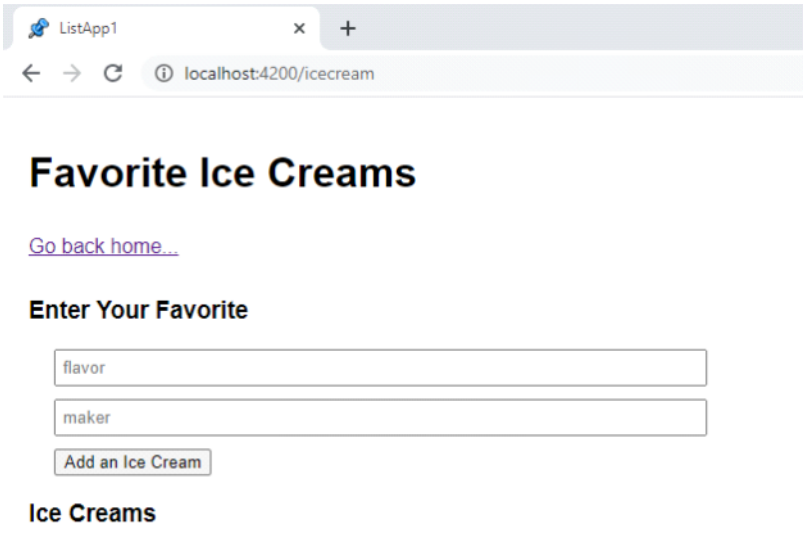


Figure 2 - Favorite Ice Creams list, unpopulated.

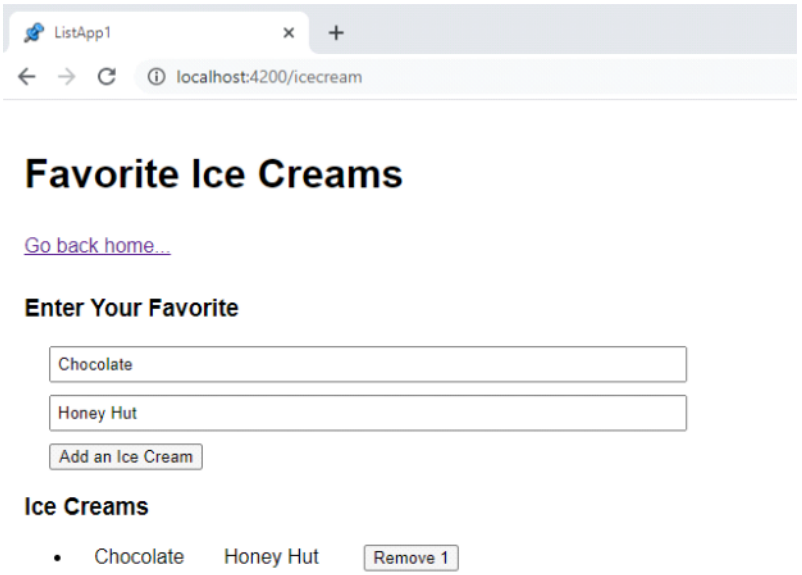


Figure 3 - Favorite Ice Creams list, populated.

Ice Cream List Operation

To add an ice cream flavor, enter the flavor and maker you prefer. Click the "Add an Ice Cream" button. The flavor and maker should now appear in the list below the edit controls (Figure 3).

Clicking the "Remove 1" button, which is next to the list entry, will delete the item from the list.

Ice Cream List Design

From an Angular perspective, the Favorite Ice Creams page is composed of three Angular components

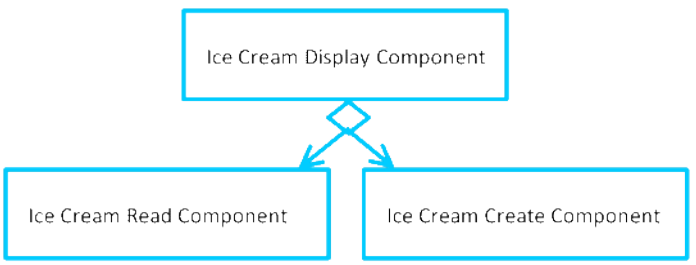
- `ice-cream-create`
- `ice-create-read`
- `ice-cream-display`

`ice-cream-create` is responsible for the entry of the ice cream flavor and maker.

`ice-cream-read` is responsible for displaying the list of ice creams. Each entry in the list includes

- The name of the flavor
- The maker of the flavor
- An button control that removes the item from the list

`ice-cream-display` is the parent container for the `ice-cream-create` and `ice-cream-read` components.



ice-cream-create Component

The `ice-cream-create` Angular component consists of the following HTML and TypeScript.

ice-cream-create.component.html

```
<div>
    <h3>Enter Your Favorite</h3>
    <div>
        <input type="text" placeholder="flavor" #flavor>
        <input type="text" placeholder="maker" #maker>
        <br>
        <button (click)="addIceCream(flavor.value, maker.value)">Add an Ice Cream</button>
    </div>
</div>
```

ice-cream-create.component.ts

```
import { Component, OnInit, Output, EventEmitter } from '@angular/core';
import { IceCream } from './../models/ice-cream';

@Component({
  selector: 'app-ice-cream-create',
  templateUrl: './ice-cream-create.component.html',
  styleUrls: ['./ice-cream-create.component.scss']
})
export class IceCreamCreateComponent implements OnInit {
  @Output() newFlavor = new EventEmitter<IceCream>();

  constructor() { }

  ngOnInit() { }

  addIceCream(theFlavor: string, theMaker: string) {
    const iceCream: IceCream = {flavor: theFlavor, maker: theMaker};
    this.newFlavor.emit(iceCream);
  }
}
```

This component defines an output event named `newFlavor`. Upon the user clicking the "Add an Ice Cream" button, the `newFlavor` event is emitted with the new ice cream flavor that is to be added to the displayed list.

ice-cream-read Component

The `ice-cream-read` Angular component consists of the following HTML and TypeScript.

ice-cream-read.component.html

```
<div *ngIf="iceCreams">
    <h3>Ice Creams</h3>
    <ul>
        <li *ngFor="let iceCream of iceCreams | async; let i = index">
            <span>{{ iceCream.flavor }}</span>
            <span>{{ iceCream.maker }}</span>
            <button (click)="deleteIceCream(i)">Remove {{i+1}}</button>
        </li>
    </ul>
</div>
```

ice-cream-read.component.ts

```
import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';
import { IceCream } from './../models/ice-cream';
import { Observable } from 'rxjs/Observable';

@Component({
```

```
  selector: 'app-ice-cream-read',
  templateUrl: './ice-cream-read.component.html',
  styleUrls: ['./ice-cream-read.component.scss']
})
export class IceCreamReadComponent implements OnInit {
  @Input() iceCreams: Observable<IceCream[]>;
  @Output() deleted = new EventEmitter<number>();

  constructor() { }

  ngOnInit() { }

  deleteIceCream(index: number) {
    this.deleted.emit(index);
  }
}
```

This component accepts as input an observable array of `IceCream`, which is defined as

ice-cream.ts

```
export interface IceCream {
    flavor: string;
    maker: string;
}
```

For output, it emits an event, `deleteIceCream`, which indicates the index of the list item being removed from the list.

ice-cream-display Component

The `ice-cream-read` Angular component is the parent container for the read and create components. This component instantiates the array of `IceCream` and the associated `Observable`. It listens to the events provided by the two child components via `onNewFlavor` and `onDeleteFlavor`. It consists of the following HTML and TypeScript.

ice-cream-display.component.html

```
<h1>Favorite Ice Creams</h1>
<div class="link-home">
    <a [routerLink]="['/home']">Go back home...</a>
</div>
<app-ice-cream-create (newFlavor)="onNewFlavor($event)"></app-ice-cream-create>
<app-ice-cream-read [iceCreams]="iceCreamObservable" (deleted)="onDeleteFlavor($event)"></app-ice-cream-read>
```

ice-cream-display.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { Subject } from 'rxjs/Subject';
import { IceCream } from './../models/ice-cream';

@Component({
  selector: 'app-ice-cream-display',
  templateUrl: './ice-cream-display.component.html',
  styleUrls: ['./ice-cream-display.component.scss']
})
export class IceCreamDisplayComponent implements OnInit {
  public iceCreamSubject: Subject<IceCream[]> = new Subject<IceCream[]>();
  public iceCreamObservable: Observable<IceCream[]> = this.iceCreamSubject.asObservable();
  private iceCreams: IceCream[] = [];

  constructor() { }

  ngOnInit() { }
```

```
  onNewFlavor(newFlavor: IceCream) {
    this.iceCreams.push(newFlavor);
    this.iceCreamSubject.next(this.iceCreams);
  }

  onDeleteFlavor(index: number) {
    this.iceCreams = this.iceCreams.filter((iceCream, id, array) => (index !== id));
    this.iceCreamSubject.next(this.iceCreams);
  }
}
```