Try it: Use forms for user input

Contents

Forms in Angular

Define the checkout form model

Create the checkout form

Next steps

At the end of Managing Data, the online store application has a product catalog and a shopping cart.

This section walks you through adding a form-based checkout feature to collect user information as part of checkout.

Forms in Angular

Forms in Angular build upon the standard HTML forms to help you create custom form controls and easy validation experiences. There are two parts to an Angular Reactive form: the objects that live in the component to store and manage the form, and the visualization of the form that lives in the template.

Define the checkout form model

First, set up the checkout form model. Defined in the component class, the form model is the source of truth for the status of the form.

- Open cart.component.ts.
- 2. Angular's FormBuilder service provides convenient methods for generating controls. As with the other services you've used, you need to import and inject the service before you can use it:
 - a. Import the FormBuilder service from the @angular/forms package.

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder } from '@angular/forms';
import { CartService } from '../cart.service';
```

The ReactiveFormsModule provides the FormBuilder service, which AppModule (in app.module.ts) already imports.

b. Inject the FormBuilder service.

```
export class CartComponent implements OnInit {
  items;

constructor(
  private cartService: CartService,
  private formBuilder: FormBuilder,
) {
  }

ngOnInit() {
  this.items = this.cartService.getItems();
}
}
```

3. Still in the Cart Component class, define the checkout Form property to store the form model.

```
src/app/cart/cart.component.ts

export class CartComponent implements OnInit {
   items;
   checkoutForm;
}
```

4. To gather the user's name and address, set the checkoutForm property with a form model containing name and address fields, using the FormBuilder group() method. Add this between the curly braces, {}, of the constructor.

```
src/app/cart/cart.component.ts

export class CartComponent implements OnInit {
   items;
   checkoutForm;

constructor(
   private cartService: CartService,
   private formBuilder: FormBuilder,
   ) {
    this.checkoutForm = this.formBuilder.group({
       name: '',
       address: ''
    });
   }
}
```

```
ngOnInit() {
   this.items = this.cartService.getItems();
}
```

- 5. For the checkout process, users need to submit their name and address. When they submit their order, the form should reset and the cart should clear.
 - a. In cart.component.ts, define an onSubmit() method to process the form. Use the CartService clearCart() method to empty the cart items and reset the form after its submission. In a real-world app, this method would also submit the data to an external server. The entire cart component class is as follows:

```
src/app/cart/cart.component.ts
import { Component, OnInit } from '@angular/core';
import { FormBuilder } from '@angular/forms';
import { CartService } from '../cart.service';
@Component({
  selector: 'app-cart',
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.css']
})
export class CartComponent implements OnInit {
  items;
  checkoutForm;
  constructor(
    private cartService: CartService,
    private formBuilder: FormBuilder,
     this.checkoutForm = this.formBuilder.group({
      name: '',
      address: ''
    });
  }
  ngOnInit() {
    this.items = this.cartService.getItems();
  }
  onSubmit(customerData) {
     // Process checkout data here
     this.items = this.cartService.clearCart();
```

```
this.checkoutForm.reset();

console.warn('Your order has been submitted', customerData);
}
```

Now that you've defined the form model in the component class, you need a checkout form to reflect the model in the view.

Create the checkout form

Use the following steps to add a checkout form at the bottom of the "Cart" view.

- Open cart.component.html.
- 2. At the bottom of the template, add an HTML form to capture user information.
- 3. Use a formGroup property binding to bind the checkoutForm to the form tag in the template. Also include a "Purchase" button to submit the form.

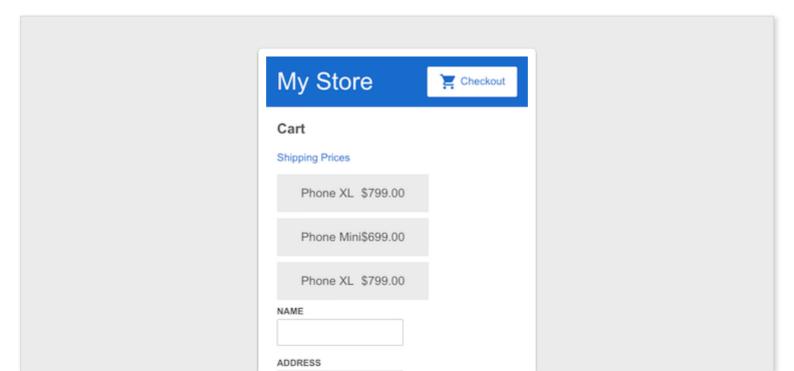
4. On the form tag, use an ngSubmit event binding to listen for the form submission and call the onSubmit() method with the checkoutForm value.

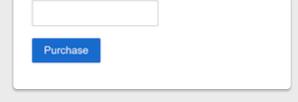
5. Add input fields for name and address. Use the formControlName attribute binding to bind the checkoutForm form controls for name and address to their input fields. The final complete component is as follows:

```
src/app/cart/cart.component.html
<h3>Cart</h3>
```

```
<a routerLink="/shipping">Shipping Prices</a>
<div class="cart-item" *ngFor="let item of items">
 <span>{{ item.name }} </span>
 <span>{{ item.price | currency }}</span>
</div>
<form [formGroup]="checkoutForm" (ngSubmit)="onSubmit(checkoutForm.value)">
 <div>
   <label for="name">
     Name
   </label>
   <input id="name" type="text" formControlName="name">
 </div>
 <div>
   <label for="address">
     Address
   </label>
    <input id="address" type="text" formControlName="address">
 </div>
 <button class="button" type="submit">Purchase/button>
</form>
```

After putting a few items in the cart, users can now review their items, enter their name and address, and submit their purchase:





To confirm submission, open the console where you should see an object containing the name and address you submitted.

Next steps

Congratulations! You have a complete online store application with a product catalog, a shopping cart, and a checkout function.

Continue to the "Deployment" section to move to local development, or deploy your app to Firebase or your own server.

RESOURCES	HELP	COMMUNITY	LANGUAGES
About	Stack Overflow	Events	简体中文版
Resource Listing	Gitter	Meetups	正體中文版
Press Kit	Report Issues	Twitter	日本語版
Blog	Code of Conduct	GitHub	한국어
Usage Analytics		Contribute	

Super-powered by Google ©2010-2020. Code licensed under an MIT-style License. Documentation licensed under CC BY 4.0.

Version 10.0.10-local+sha.b32126c335.