

Adding navigation

This guide builds on the first step of the Getting Started tutorial, [Get started with a basic Angular app](#).

At this stage of development, the online store application has a basic product catalog.

In the following sections, you'll add the following features to the application:

- Type a URL in the address bar to navigate to a corresponding product page.
- Click links on the page to navigate within your single-page application.
- Click the browser's back and forward buttons to navigate the browser history intuitively.

{@a define-routes}

Associate a URL path with a component

The application already uses the Angular `Router` to navigate to the `ProductListComponent`. This section shows you how to define a route to show individual product details.

1. Generate a new component for product details. In the terminal generate a new `product-details` component by running the following command:

```
ng generate component product-details
```

2. In `app.module.ts`, add a route for product details, with a `path` of `products/:productId` and `ProductDetailsComponent` for the `component`.
3. Open `product-list.component.html`.
4. Modify the product name anchor to include a `routerLink` with the `product.id` as a parameter.

The `RouterLink` directive helps you customize the anchor element. In this case, the route, or URL, contains one fixed segment, `/products`. The final segment is variable, inserting the `id` property of the current product. For example, the URL for a product with an `id` of 1 would be similar to

```
https://getting-started-myfork.stackblitz.io/products/1
```

5. Verify that the router works as intended by clicking the product name. The application should display the `ProductDetailsComponent`, which currently says "product-details works!"

Notice that the URL in the preview window changes. The final segment is `products/#` where `#` is the number of the route you clicked.



Product details view with updated URL

View product details

The `ProductDetailsComponent` handles the display of each product. The Angular Router displays components based on the browser's URL and [your defined routes](#).

In this section, you'll use the Angular Router to combine the `products` data and route information to display the specific details for each product.

1. In `product-details.component.ts`, import `ActivatedRoute` from `@angular/router`, and the `products` array from `../products`.

2. Define the `product` property.

3. Inject `ActivatedRoute` into the `constructor()` by adding `private route: ActivatedRoute` as an argument within the constructor's parentheses.

`ActivatedRoute` is specific to each component that the Angular Router loads. `ActivatedRoute` contains information about the route and the route's parameters.

By injecting `ActivatedRoute`, you are configuring the component to use a service. The [Managing Data](#) step covers services in more detail.

4. In the `ngOnInit()` method, extract the `productId` from the route parameters and find the corresponding product in the `products` array.

The route parameters correspond to the path variables you define in the route. To access the route parameters, we use `route.snapshot`, which is the `ActivatedRouteSnapshot` that contains information about the active route at that particular moment in time. The URL that matches the route provides the `productId`. Angular uses the `productId` to display the details for each unique product.

5. Update the `ProductDetailsComponent` template to display product details with an `*ngIf`. If a product exists, the `<div>` renders with a name, price, and description.

The line, `<h4>{{ product.price | currency }}</h4>`, uses the `currency` pipe to transform `product.price` from a number to a currency string. A pipe is a way you can transform data in your HTML template. For more information about Angular pipes, see [Pipes](#).

When users click on a name in the product list, the router navigates them to the distinct URL for the product, shows the `ProductDetailsComponent`, and displays the product details.



Product details page with updated URL and full details displayed

For more information about the Angular Router, see [Routing & Navigation](#).

What's next

You have configured your application so you can view product details, each with a distinct URL.

To continue exploring Angular:

- Continue to [Managing Data](#) to add a shopping cart feature, manage cart data, and retrieve external data for shipping prices.
- Skip ahead to [Deployment](#) to deploy your application to Firebase or move to local development.

@reviewed 2021-09-15