

Navigation and Routing Additional Techniques

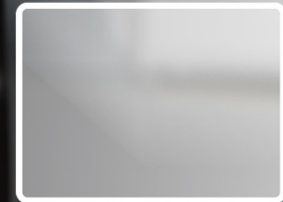
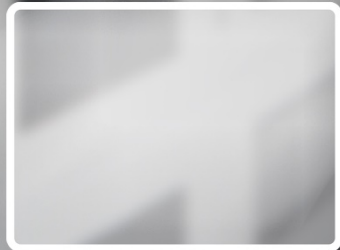


Deborah Kurata

CONSULTANT | SPEAKER | AUTHOR | MVP | GDE

@deborahkurata | blogs.msmvps.com/deborahk/





Module Overview



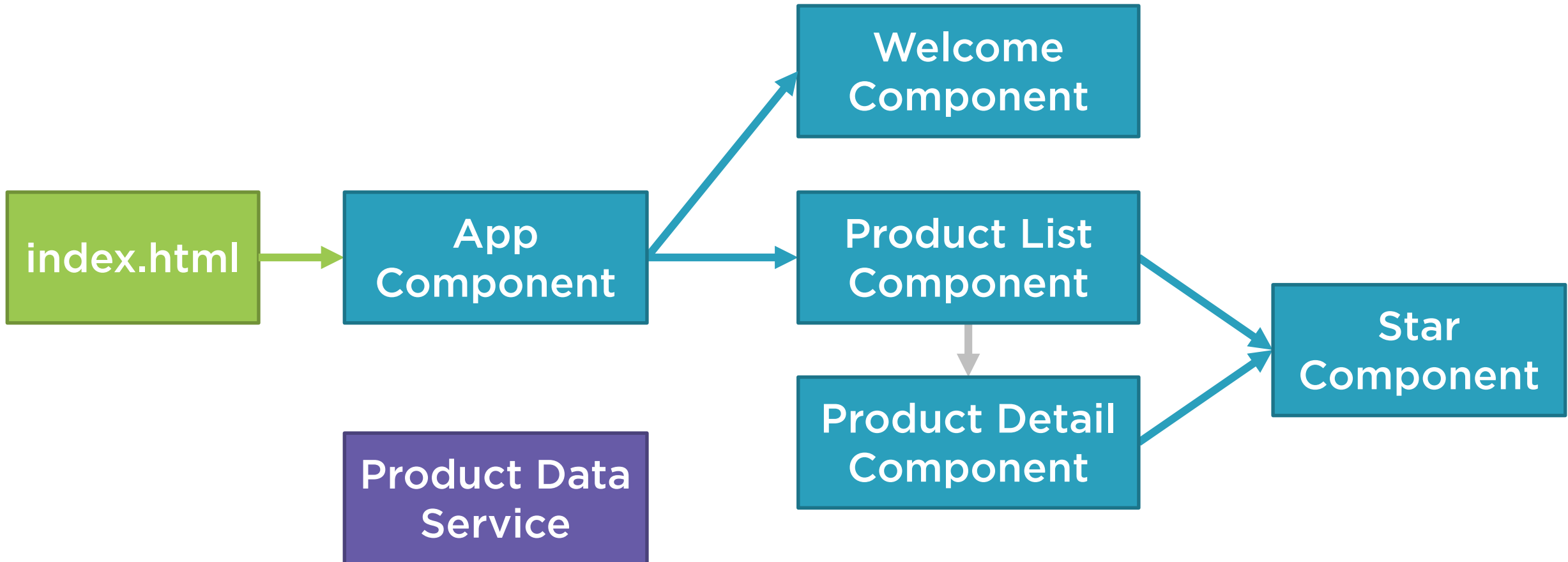
Passing Parameters to a Route

Activating a Route with Code

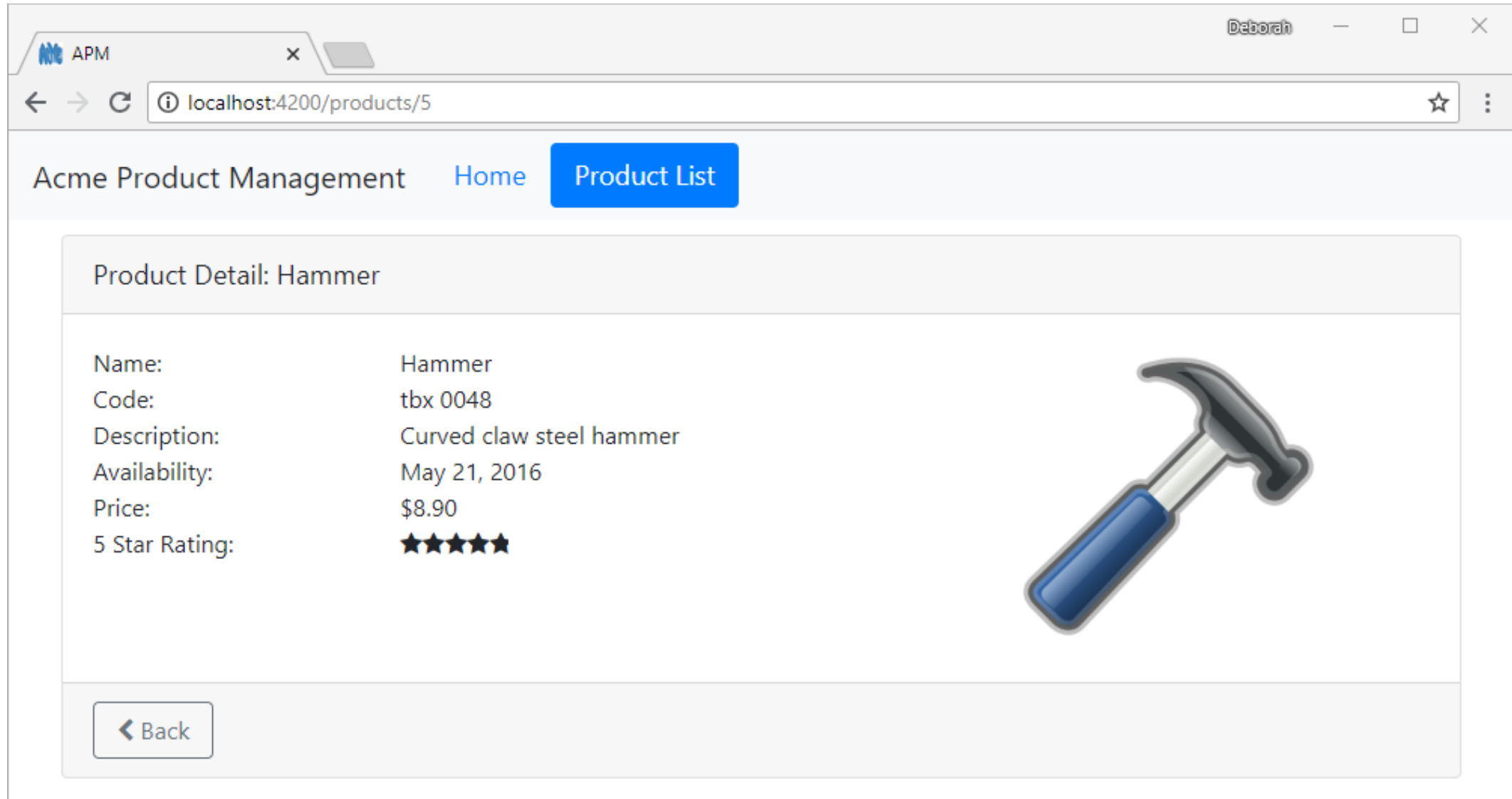
Protecting Routes with Guards



Application Architecture



Passing Parameters to a Route



Passing Parameters to a Route

app.module.ts

```
@NgModule({
  imports: [
    ...,
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id', component: ProductDetailComponent },
      { path: 'welcome', component: WelcomeComponent },
      { path: '', redirectTo: 'welcome', pathMatch: 'full' },
      { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
    ])
  ],
  declarations: [...],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```



Passing Parameters to a Route

product-list.component.html

```
<td>
  <a [routerLink]="[' /products', product.productId]">
    {{product.productName}}
  </a>
</td>
```

app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```



Reading Parameters from a Route

product-detail.component.ts

```
import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute) {
  console.log(this.route.snapshot.paramMap.get('id'));
}
```

app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```



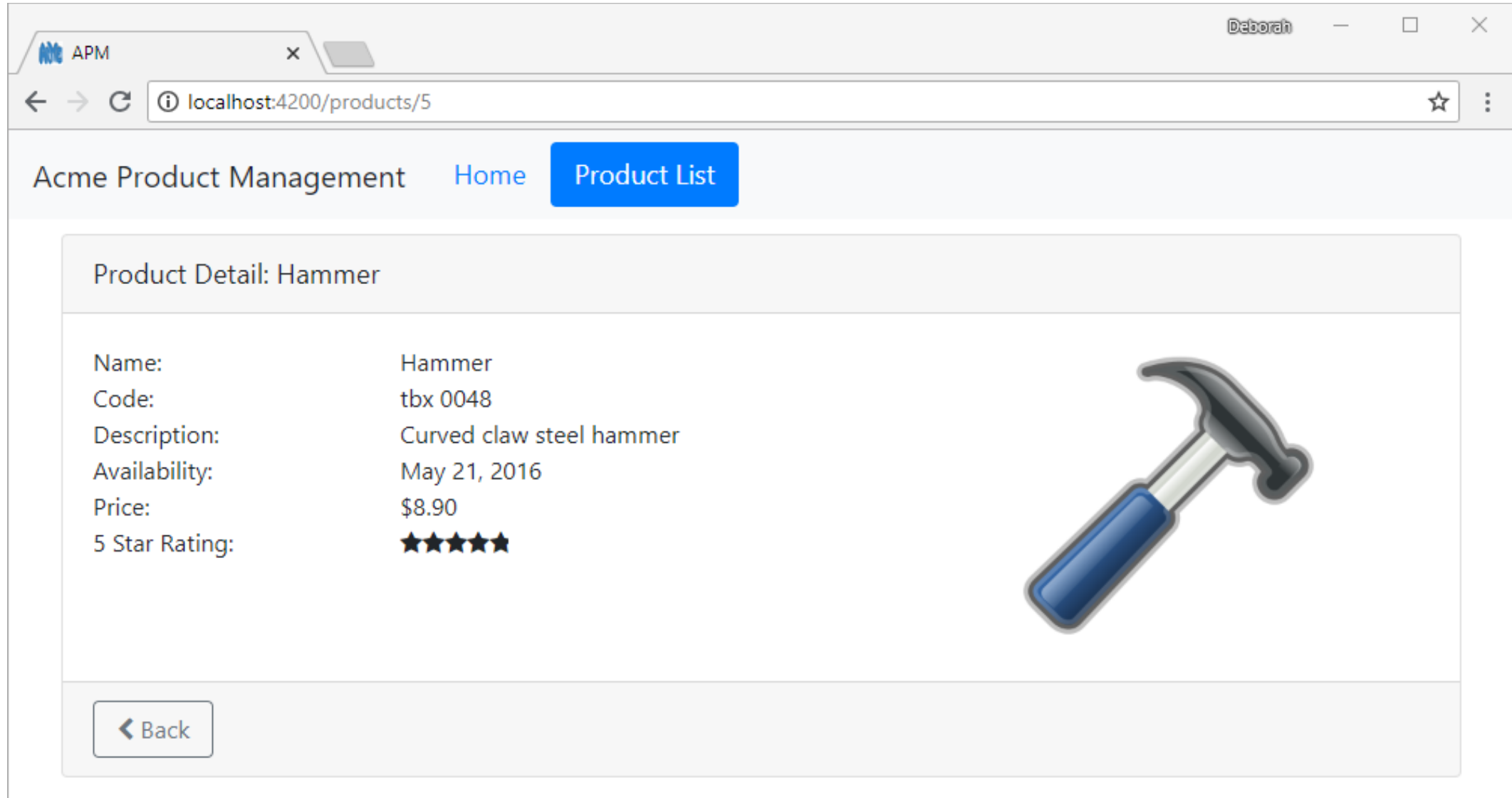
Demo



Passing Parameters to a Route



Activating a Route with Code



Activating a Route with Code

product-detail.component.ts

```
import { Router } from '@angular/router';  
...  
constructor(private router: Router) { }  
  
onBack(): void {  
    this.router.navigate(['/products']);  
}
```



Protecting Routes with Guards



CanActivate

- Guard navigation to a route

CanDeactivate

- Guard navigation from a route

Resolve

- Pre-fetch data before activating a route

CanLoad

- Prevent asynchronous routing



Building a Guard

product-detail.guard.ts

```
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';

@Injectable({
  providedIn: 'root'
})
export class ProductDetailGuard implements CanActivate {

  canActivate(): boolean {
    ...
  }
}
```



Using a Guard

app.module.ts

```
@NgModule({
  imports: [
    ...,
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id',
        canActivate: [ ProductDetailGuard ],
        component: ProductDetailComponent },
      ...])
  ],
  declarations: [...],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```



Demo



Protecting Routes with Guards



Routing Checklist: Passing Parameters

app.module.ts

```
{ path: 'products/:id', component: ProductDetailComponent }
```

product-list.component.html

```
<a [routerLink]="[' /products', product.productId]">
  {{product.productName}}
</a>
```

product-detail.component.ts

```
import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute) {
  console.log(this.route.snapshot paramMap.get('id'));
}
```



Routing Checklist: Activate a Route with Code



Use the Router service

- Import the service
- Define it as a dependency

Create a method that calls the navigate method of the Router service

- Pass in the link parameters array

Add a user interface element

- Use event binding to bind to the created method

Routing Checklist: Protecting Routes with Guards



Build a guard service

- Implement the guard type (CanActivate)
- Create the method (canActivate())

Register the guard service provider

- Use the providedIn property

Add the guard to the desired route



Summary



Passing Parameters to a Route

Activating a Route with Code

Protecting Routes with Guards



Learning More



"Angular Routing"

- Required, optional, & query parameters
- Prefetching with route resolvers
- Child and secondary router outlets
- Router guards
- Lazy loading

