

Angular

Enrutado y navegaci3n

CertiDevs

 ndice de contenidos

1. Enrutamiento	1
2. Configuraci�n de rutas	1
2.1. Navegaci�n y visualizaci�n de rutas	2
3. Separar enrutado en un m�dulo	2
4. Rutas con par�metros	3
5. ActivatedRoute	4
5.1. params	5
5.2. paramMap	5
5.3. snapshot	5
6. Rutas secundarias y anidadas	6
7. Ejemplo 1	7
8. Ejemplo 2	9

1. Enrutamiento

El enrutamiento en Angular permite navegar entre diferentes vistas y componentes dentro de una aplicación. Se logra utilizando el módulo `RouterModule` y un conjunto de rutas configuradas.

El enrutamiento en Angular permite navegar entre diferentes componentes y vistas en una aplicación de una sola página (SPA).

El enrutamiento se logra mediante la configuración de rutas o routes, la asignación de componentes a estas rutas y la utilización de un enrutador para gestionar la navegación y el estado de la aplicación.

2. Configuración de rutas

Las rutas en Angular se definen como una matriz de objetos que describen las rutas y sus componentes asociados.

Cada objeto de ruta tiene al menos dos propiedades: `path` y `component`.

¥ La propiedad `path` es una cadena que define la URL de la ruta.

¥ La propiedad `component` es una referencia al componente que se debe mostrar cuando se navega a la ruta.

Para configurar las rutas en una aplicación Angular, primero se debe importar el módulo `RouterModule` y el servicio `Routes` desde `@angular/router`.

A continuación, se debe llamar al método `forRoot()` del `RouterModule` con la matriz de rutas como argumento y agregar el `RouterModule` a la lista de importaciones del módulo principal de la aplicación.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'about', component: AboutComponent }
];

@NgModule({
  declarations: [AppComponent, HomeComponent, AboutComponent],
  imports: [BrowserModule, RouterModule.forRoot(routes)],
  bootstrap: [AppComponent]
})
```

```
export class AppModule { }
```

En este ejemplo, se configuran dos rutas: la ruta de inicio (representada por una cadena vacía) y la ruta "about". Cada ruta está asociada a un componente que se mostrará cuando se navegue a la ruta.

2.1. Navegación y visualización de rutas

Para navegar a una ruta en Angular, se puede utilizar la directiva `routerLink` en un enlace o elemento del DOM. La directiva `routerLink` acepta una expresión que define la ruta a la que se debe navegar.

```
<nav>
  <a routerLink="/">Inicio</a>
  <a routerLink="/about">Acerca de</a>
</nav>
```

Para mostrar el componente asociado a la ruta actual, se debe utilizar la directiva `router-outlet` en la plantilla de la aplicación.

```
<router-outlet></router-outlet>
```

El enrutador de Angular automáticamente actualizará el contenido del `router-outlet` con el componente asociado a la ruta actual.

3. Separar enrutado en un módulo

En una aplicación Angular, es una buena práctica separar la configuración del enrutado en un módulo dedicado llamado `AppRoutingModule`.

Esto ayuda a mantener el código más organizado y facilita la comprensión de la estructura de enrutado de la aplicación.

Configura las rutas y el enrutador en `app-routing.module.ts`.

Importa `RouterModule` y `Routes` de `@angular/router`, define tus rutas y exporta el `RouterModule` configurado:

```
// src/app/app-routing.module.ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

import { HomeComponent } from '../home/home.component';
import { AboutComponent } from '../about/about.component';

const routes: Routes = [
```

```

    { path: '', component: HomeComponent },
    { path: 'about', component: AboutComponent },
  ];

  @NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule],
  })
  export class AppRoutingModule {}

```

En este ejemplo, hemos definido dos rutas: una para el componente `HomeComponent` y otra para el componente `AboutComponent`.

Luego, importamos el `RouterModule` y lo configuramos con nuestras rutas utilizando el método `forRoot()`.

Importa el `AppRoutingModule` en tu `AppModule`:

```

// src/app/app.module.ts
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';

@NgModule({
  declarations: [
    AppComponent,
    HomeComponent,
    AboutComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}

```

4. Rutas con parámetros

En algunos casos, es posible que desee pasar datos a una ruta mediante parámetros.

Para definir una ruta con parámetros, se debe agregar un segmento de ruta con dos puntos (:) seguido del nombre del parámetro en la propiedad `path` de la ruta.

```
const routes: Routes = [
  // ...
  { path: 'user/:id', component: UserComponent }
];
```

Para acceder al valor del parámetro en el componente asociado a la ruta, se debe inyectar el servicio `ActivatedRoute` y utilizar la propiedad `params` para obtener un `Observable` que emite los parámetros de la ruta.

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html'
})
export class UserComponent implements OnInit {
  userId: string;

  constructor(private route: ActivatedRoute) {}

  ngOnInit() {
    this.route.params.subscribe(params => {
      this.userId = params['id'];
    });
  }
}
```

En este ejemplo, el componente `UserComponent` se suscribe al `Observable` de parámetros de ruta y actualiza la propiedad `userId` cuando cambian los parámetros.

La propiedad `userId` puede ser utilizada en la plantilla o en la lógica del componente para mostrar o cargar datos específicos del usuario.

Para navegar a una ruta con parámetros utilizando la directiva `routerLink`, se puede pasar un array con la ruta y los valores de los parámetros.

```
<a [routerLink]="['/user', '123']">Ver perfil del usuario 123</a>
```

5. ActivatedRoute

El servicio `ActivatedRoute` proporciona información sobre la ruta, sus parámetros y datos adicionales asociados con la ruta, también relaciones jerárquicas con otras rutas.

`ActivatedRoute` es una clase proporcionada por el módulo `@angular/router` que contiene información sobre la ruta activa asociada con el componente cargado por el enrutador.

Existen varias formas de leer parámetros de la ruta utilizando `ActivatedRoute`:

5.1. params

params: `params` es un `Observable` que emite un objeto de parámetros cada vez que cambian los parámetros de la ruta. Puedes suscribirte a este `Observable` para leer los parámetros y realizar acciones cuando cambien los parámetros.

```
// En tu componente
constructor(private route: ActivatedRoute) {}

ngOnInit(): void {
  this.route.params.subscribe((params) => {
    console.log(params);
  });
}
```

5.2. paramMap

paramMap: `paramMap` es un `Observable` que emite un `ParamMap` cada vez que cambian los parámetros de la ruta. `ParamMap` es un objeto que representa un conjunto inmutable de parámetros y ofrece métodos para trabajar con los parámetros. A diferencia de `params`, `paramMap` facilita la manipulación de múltiples valores de parámetros y la lectura de valores de parámetros de manera segura.

```
// En tu componente
constructor(private route: ActivatedRoute) {}

ngOnInit(): void {
  this.route.paramMap.subscribe((paramMap) => {
    console.log(paramMap.get('id'));
  });
}
```

5.3. snapshot

snapshot: A diferencia de `params` y `paramMap`, que son `Observables`, `snapshot` es una instantánea estática de la información de la ruta activa en un momento específico. Puedes utilizar `snapshot` para leer los parámetros de la ruta sin suscribirte a un `Observable`, pero no recibirás actualizaciones cuando cambien los parámetros.

```
// En tu componente
constructor(private route: ActivatedRoute) {}

ngOnInit(): void {
```

```
É console.log(this.route.snapshot.params);
É console.log(this.route.snapshot.paramMap.get('id'));
}
```

El uso de `params` y `paramMap` es más recomendable en la mayoría de los casos, ya que te permite reaccionar a cambios en los parámetros de la ruta. Sin embargo, `snapshot` puede ser útil en situaciones donde solo necesitas leer los parámetros de la ruta una vez y no esperas que cambien durante la vida útil del componente.

En resumen, `ActivatedRoute` es una clase fundamental en el enrutamiento de Angular que proporciona información sobre la ruta activa. Puedes utilizar `params`, `paramMap` o `snapshot` para leer y gestionar los parámetros de la ruta según tus necesidades y la dinámica de tu aplicación.

6. Rutas secundarias y anidadas

El enrutamiento en Angular también admite rutas secundarias y anidadas, lo que permite crear aplicaciones más complejas y modulares.

Las rutas secundarias se definen utilizando la propiedad `children` en un objeto de ruta y se asocian a un componente hijo que se mostrará dentro del componente padre.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AppComponent } from './app.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { SettingsComponent } from './settings/settings.component';
import { ProfileComponent } from './profile/profile.component';

const routes: Routes = [
  É {
    É path: 'dashboard',
    É component: DashboardComponent,
    É children: [
      É { path: 'settings', component: SettingsComponent },
      É { path: 'profile', component: ProfileComponent }
    ]
  }
];

@NgModule({
  É declarations: [AppComponent, DashboardComponent, SettingsComponent,
  ProfileComponent],
  É imports: [BrowserModule, RouterModule.forRoot(routes)],
  É bootstrap: [AppComponent]
})
export class AppModule { }
```


En este ejemplo, se configura una ruta principal `dashboard` y dos rutas secundarias `settings` y `profile`.

Los componentes secundarios se mostrarán dentro del componente `DashboardComponent` cuando se navegue a las rutas secundarias.

Para mostrar el componente asociado a la ruta secundaria actual, se debe utilizar la directiva `router-outlet` en la plantilla del componente padre.

```
<!-- archivo dashboard.component.html -->
<div>
  <h1>Panel de control </h1>
  <router-outlet></router-outlet>
</div>
```

En resumen, el enrutamiento en Angular es una característica clave para crear aplicaciones de una sola página (SPA) y gestionar la navegación y el estado de la aplicación.

La configuración de rutas, la navegación y la visualización de componentes asociados a rutas, junto con el manejo de parámetros y rutas anidadas, son aspectos fundamentales para desarrollar aplicaciones Angular eficientes y escalables.

7. Ejemplo 1

Vamos a crear una aplicación simple con enrutamiento y navegación en Angular que tenga una página de inicio, una página de "Acerca de" y una página de "Contacto".

Comienza generando los componentes utilizando Angular CLI:

```
ng generate component home
ng generate component about
ng generate component contact
```

A continuación, configura las rutas para la aplicación en `app.module.ts`.

Importa `RouterModule` y `Routes` y define las rutas para cada componente:

```
// src/app/app.module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouterModule, Routes } from '@angular/router';

import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';
```

```
const routes: Routes = [
  { path: '', redirectTo: '/home', pathMatch: 'full' },
  { path: 'home', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: 'contact', component: ContactComponent },
];

@NgModule({
  declarations: [AppComponent, HomeComponent, AboutComponent, ContactComponent],
  imports: [BrowserModule, RouterModule.forRoot(routes)],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

Actualiza `app.component.html` para agregar un menú de navegación y un elemento `<router-outlet>`:

```
<!-- src/app/app.component.html -->
<nav>
  <a routerLink="/home">Inicio</a>
  <a routerLink="/about">Acerca de</a>
  <a routerLink="/contact">Contacto</a>
</nav>
<router-outlet></router-outlet>
```

A continuación, agrega contenido a cada componente:

```
<!-- src/app/home/home.component.html -->
<h1>Inicio</h1>
<p>Bienvenido a nuestra aplicación de ejemplo con enrutamiento y navegación en Angular. </p>
```

```
<!-- src/app/about/about.component.html -->
<h1>Acerca de</h1>
<p>Esta es la página "Acerca de" de nuestra aplicación de ejemplo. Aquí puedes incluir información sobre la aplicación y su propietario. </p>
```

```
<!-- src/app/contact/contact.component.html -->
<h1>Contacto</h1>
<p>Si tienes alguna pregunta o comentario, por favor contáctanos a través de nuestro formulario de contacto o correo electrónico. </p>
```

Con este ejemplo, hemos creado una aplicación simple con enrutamiento y navegación en Angular que permite al usuario navegar entre las páginas de "Inicio", "Acerca de" y "Contacto".

8. Ejemplo 2

En este ejemplo, vamos a crear una aplicaci3n de comercio electr3nico (ecommerce) simple que utiliza enrutamiento y navegaci3n en Angular para mostrar una lista de productos y permitir que el usuario navegue a una p3gina de detalles del producto.

Primero, crea un servicio `ProductService` para gestionar los productos:

```
// src/app/product.service.ts
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root',
})
export class ProductService {
  private products = [
    { id: 1, name: 'Producto 1', description: 'Descripci3n del Producto 1', price: 100 },
    { id: 2, name: 'Producto 2', description: 'Descripci3n del Producto 2', price: 150 },
  ];

  getProducts() {
    return this.products;
  }

  getProductById(id: number) {
    return this.products.find(product => product.id === id);
  }
}
```

Luego, actualiza el archivo `app.module.ts` para configurar el enrutamiento:

```
// src/app/app.module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { RouterModule, Routes } from '@angular/router';

import { AppComponent } from './app.component';
import { ProductListComponent } from './product-list/product-list.component';
import { ProductDetailComponent } from './product-detail/product-detail.component';
import { ProductService } from './product.service';

const routes: Routes = [
  { path: '', redirectTo: '/products', pathMatch: 'full' },
  { path: 'products', component: ProductListComponent },
  { path: 'product/:id', component: ProductDetailComponent },
];
```

```

@NgModule({
  declarations: [AppComponent, ProductListComponent, ProductDetailComponent],
  imports: [BrowserModule, RouterModule.forRoot(routes)],
  providers: [ProductService],
  bootstrap: [AppComponent],
})
export class AppModule {}

```

En este archivo, hemos importado `RouterModule` y `Routes` y configurado las rutas para la aplicación.

La ruta `products` nos lleva al componente `ProductListComponent` y la ruta `product/:id` nos lleva al componente `ProductDetailComponent`.

Actualiza el componente `ProductListComponent` para utilizar el `ProductService` y navegar a la página de detalles del producto:

```

// src/app/product-list/product-list.component.ts
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { ProductService } from '../product.service';

@Component({
  selector: 'app-product-list',
  template: `
    <div>
      <h1>Lista de productos</h1>
      <ul>
        <li *ngFor="let product of products"
            (click)="viewProductDetails(product.id)">
          {{ product.name }}
        </li>
      </ul>
    </div>
  `,
})
export class ProductListComponent {
  products = [];

  constructor(private productService: ProductService, private router: Router) {
    this.products = productService.getProducts();
  }

  viewProductDetails(productId: number): void {
    this.router.navigate(['/product', productId]);
  }
}

```

Aquí, hemos importado `Router` y `ProductService`, y actualizado el componente para navegar a la

página de detalles del producto al hacer clic en un producto.

Actualiza el componente `ProductDetailComponent` para recibir el ID del producto de la ruta y utilizar el `ProductService` para cargar los detalles del producto:

```
// src/app/product-detail/product-detail.component.ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { ProductService } from '../product.service';

@Component({
  selector: 'app-product-detail',
  template: `
    <div *ngIf="product">
      <h2>Detalles del producto</h2>
      <p>Nombre: {{ product.name }}</p>
      <p>Descripción: {{ product.description }}</p>
      <p>Precio: {{ product.price }}</p>
    </div>`
})

export class ProductDetailComponent implements OnInit {
  product: any;

  constructor(private route: ActivatedRoute, private productService: ProductService) {}

  ngOnInit(): void {
    const productId = parseInt(this.route.snapshot.paramMap.get('id'), 10);
    this.product = this.productService.getProductById(productId);
  }
}
```

En `ProductDetailComponent`, hemos importado `ActivatedRoute` y `ProductService`, y actualizado el componente para cargar los detalles del producto utilizando el ID del producto obtenido de la ruta.

Finalmente, actualiza `app.component.html` para incluir el elemento `<router-outlet>`:

```
<router-outlet></router-outlet>
```

Con este ejemplo, hemos creado una aplicación de comercio electrónico simple que utiliza enrutamiento y navegación en Angular para mostrar una lista de productos y permitir que el usuario navegue a una página de detalles del producto.

El ID del producto se pasa en la URL y se utiliza para cargar los detalles del producto utilizando el `ProductService`.