Angular Training

Services

Peter Bouda, hey@peterbouda.eu

ng-lisbon.com

Why Routing

- Routing maps URLs to specific functionality
- In a single-page application routing is not strictly necessary
- Routing in Angular serves 3 purposes:
  - Save the state of the app, for bookmarks, links, ...
  - You can connect the routes to modules, then create bundles for those parts
  - You can restrict the access to certain URLs, e.g. based on user roles

ng-lisbon.com

Define Routes

```
const routes: Routes = [
  { path: 'component-one', component: ComponentOne },
  { path: 'component-two', component: ComponentTwo }
];
```

ng-lisbon.com

Define Routes: Parameters

path	URL that is displayed in the browser
component	The component that will be displayed at the URL
redirectTo	Redirect to another route
pathMatch	How is the path compared, `full` vs. `prefix` (default)
children	An array of child routes
loadChildren	Child routes will be loaded from another module

RouterModule

```
import { RouterModule, Routes } from '@angular/router';
const routes: Routes = [
        { path: 'component-one', component: ComponentOne },
        { path: 'component-two', component: ComponentTwo }
];
export const routing = RouterModule.forRoot(routes);
```

Then in app.module:

```
...
import { routing } from './app.routes';

@NgModule({
        imports: [
                BrowserModule,
                routing
        ],
        ...
```

Redirect and Default Route

```
{ path: "", redirectTo: "/home", , pathMatch: "full" },
{ path: "/home", component: HomeComponent },
{ path: "/login", component: LoginComponent },
{ path: "**", redirectTo: "/home" }
```

ng-lisbon.com

Links to Routes

```
<a routerLink="/home">Component One</a>

<a [routerLink]="['/product', 'new']">Component One</a>
```

Navigate programmatically:

```
this.router.navigate(['/product', 'new']);
```

Attention: You have to inject the "Router" service!

Display the Router Components

```
import { Component } from '@angular/core';
@Component({
  selector: 'app',
  template: `
    <nav>
      <a routerLink="/home">Startseite</a>
      <a routerLink="/products">Produkte</a>
    </nav>
    <router-outlet></router-outlet>
  `
})
export class AppComponent {}
```

Parameters in Routes

- For example the product ID in a product URL: `/products/1`
- What you need to do:
  - Add the parameter to the route definition
  - Create a link with the parameter
  - Query parameter value in the component and
    e.g. load the product from the API

ng-lisbon.com

Route Parameter Definition

```
export const routes: Routes = [
  { path: '', redirectTo: 'products', pathMatch: 'full' },
  { path: 'products', component: ProductList },
  { path: 'products/new', component: ProductEdit }
  { path: 'products/:id', component: ProductDetails }
];
```

Attention: The order matters!

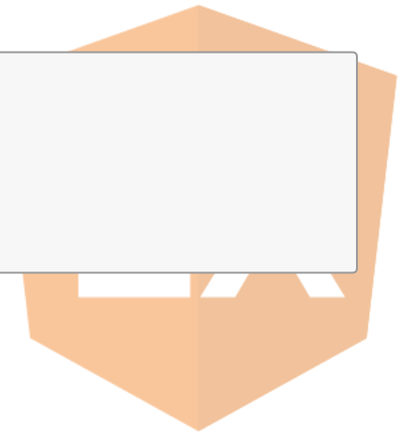
ng-lisbon.com

Create Links

```
<a *ngFor="let product of products" [routerLink]="['/products', product.id]">
    {{ product.name }}
</a>
```

Or programmatically:

```
goToProductDetails(id) {
  this.router.navigate(['/products', id]);
}
```



Read Parameters

```
import { Component, OnInit, OnDestroy } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
@Component({
  selector: 'app-product-details',
  template: `
    <div>Detail for product: {{id}}</div>
  `,
})
export class ProductDetailsComponent implements OnInit, OnDestroy {
  id: number;
  private subscription: any;
  constructor(private route: ActivatedRoute) {}
  ngOnInit() {
    this.subscription = this.route.params.subscribe(params => {
      this.id = +params['id'];
      // Load details
    });
  }
  ngOnDestroy() {
    this.subscription.unsubscribe();
  }
}
```

Why Child Routes?

- A site has different areas, like product pages, with lists, details, shop, ...
- Those areas share other parts, like the submenu, feature boxes, ...
- The areas that contain changing content can be implemented with child routes (one route for display of a product, anther route for editing, ...)
- Those functionalities can be organized in modules and the child routing enables the bundling that comes with Angular

Child Routes

```
export const routes: Routes = [
  { path: '', redirectTo: 'products', pathMatch: 'full' },
  { path: 'products', component: ProductList },
  { path: 'products/:id', component: ProductDetails,
    children: [
      { path: '', redirectTo: 'spec', pathMatch: 'full' },
      { path: 'spec', component: ProductSpec },
      { path: 'edit', component: ProductEdit }
    ]
  }
];
```

Attention: Add `<router-outlet>` to e.g. product-detail.component.html!