Eric Ferreira Follow
Developer
Nov 4 · 5 min read

Angular 6 - Criando uma aplicação em lazyload com splash de carregamento

Vamos criar aqui uma aplicação que roda em modo "lazy loading", exibindo um "splash "de carregamento do conteúdo em que está sendo baixado no momento.

Provavelmente você ja deve ter desenvolvido uma put#@* aplicação, colocado ela no ar e depois os seus clientes começaram a reclamar que o sistema não abre ou que demora minutos para ser aberto, e acabou sendo extremamente frustante, não é mesmo?

"Como assim o sistema não ta abrindo no browser do cliente!?" 😠

E isso com certeza ocorreu porque você não desenvolveu seu sistema aplicando o conceito de "lazy loading", que é exatamente pedir para o browser carregar os scripts no momento em que é preciso ser carregado. É comum frameworks SPAs gerarem em seu build um script com tudo junto, a não ser que você peça para que ele não faça isso, mas para isso você precisa dizer quais e quando o conteúdo deve ser carregado, e é isto que vamos aprender a fazer aqui.

Mão na massa 👊

01 - Criando a aplicação em Angular 6

Para começar, vamos criar uma aplicação com rotas (pois as rotas vão ser nossas aliadas para que tudo aconteça):

```
ng new app-lazyloading --routing
cd app-lazyloading
```

02 - Criando os módulos

No Angular, é possível dividir sua aplicação em módulos, o que acaba deixando seu código mais organizado. Mas o que mais importa para nós agora é saber que são estes módulos que iremos usar para deixar nossa aplicação mais leve.

Neste nosso exemplo vamos criar abaixo 3 módulos, supondo que estejamos fazendo uma espécie de site, onde cada página represente

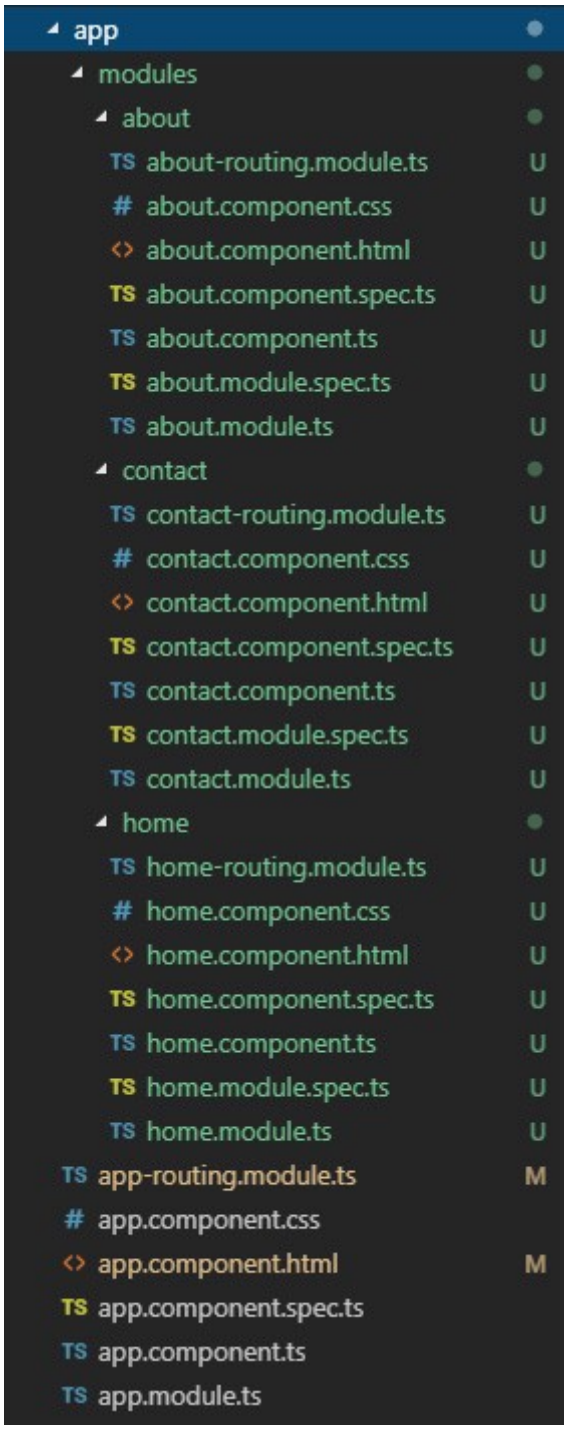
um conteúdo totalmente diferente. (Repare que estamos colocando dentro de uma pasta “modules” e também estamos criando com rotas):

Home: `ng g m modules/home --routing`
About: `ng g m modules/about --routing`
Contact: `ng g m modules/contact --routing`

Como só criamos os módulos, temos também que criaremos os componentes para cada um:

Home: `ng g c modules/home`
About: `ng g c modules/about`
Contact: `ng g c modules/contact`

Após criado os módulos e seus componentes, você deverá ter algo bem parecido com isso:



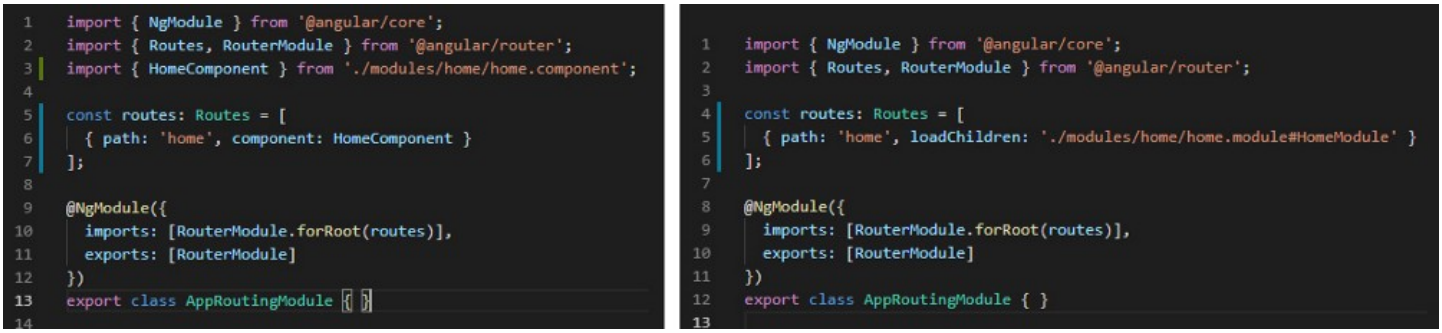
03 - Alterando o módulo de rotas principal

Agora iremos alterar o módulo de rotas principal que o Angular criou para nós no momento que criamos o projeto. Abra o arquivo **app-routing.module.ts**. Você verá que neste módulo não há nada implementado. Como estamos criando uma espécie de site com 3 páginas, precisamos definir as rotas para cada uma delas e qual conteúdo elas deverão exibir.

Existem duas formas de pedirmos uma rota para exibir um conteúdo, pode ser via:

Componente: é quando se coloca diretamente o componente que queremos que seja renderizado, no entanto esse componente fará parte do seu módulo atual, mas se você usar muito deste recurso, poderá ocorrer do seu módulo ficar muito pesado e trará lentidão ao ser baixado pelo browser.

Módulo: esta forma é quando se coloca o caminho de um módulo, onde ele não faz parte do módulo atual em que está. Desta forma você diz para o sistema de rotas do módulo atual que, quando a tal rota for acessada o módulo "X" será carregado, assim o mesmo será baixado e em seguida seu componente principal será renderizado.



```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './modules/home/home.component';

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

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

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
  { path: 'home', loadChildren: './modules/home/home.module#HomeModule' }
];

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

Lado esquerdo por "**Componente**" e do lado direito por "**Módulo**"

Como estamos preocupados com a lentidão em que o sistema poderá ter, iremos escolher a segunda opção, por isso criamos módulos separados para cada finalidade. Então, nosso arquivo com o módulo de rotas ficará assim:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [
  { path: 'home', loadChildren: './modules/home/home.modul
  { path: 'about', loadChildren: './modules/about/about.mo
  { path: 'contact', loadChildren: './modules/contact/cont
];

@NgModule({
```

Se formos testar nossa aplicação ainda não irá funcionar, pois o módulo de rotas de cada módulo que criamos precisa ter uma rota default, que

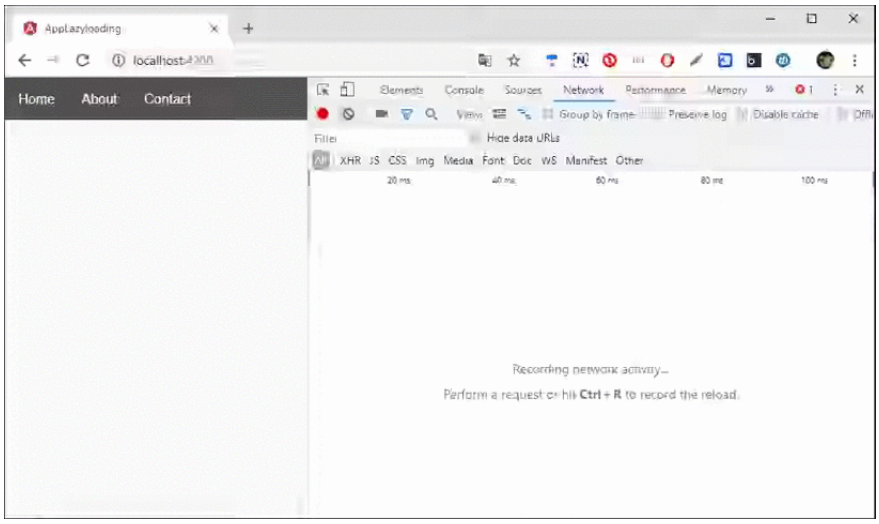
neste caso será aqueles componentes únicos que criamos para cada módulo.

Para o módulo **Home**, veja como fica:
- **home-routing.module.ts**

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home.component';

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

@NgModule({
```

Faça o mesmo para os outros módulos de rotas de **About** e **Contact**. Não se esqueça que cada um possui seu próprio componente.

Veja como ficou!



Perceba do lado direito que, para cada página (rota), o browser está baixando o conteúdo (script) do módulo que foi solicitado.

04 - Colocando um splash de "loading"

Mesmo quando recorremos ao "lazyload", é comum que algum módulo fique com um tamanho (KB) o suficiente para que o browser demore alguns segundinhos para baixa-lo. E mesmo que isso demore 1

segundo, devemos tratar isto informando algo na tela dizendo ao usuário que aguarde um momento.

O que precisamos fazer é identificar quando a transição de rota começa e quando ela termina, para que saibamos quando mostrar e quando esconder o aviso de “aguarde”.

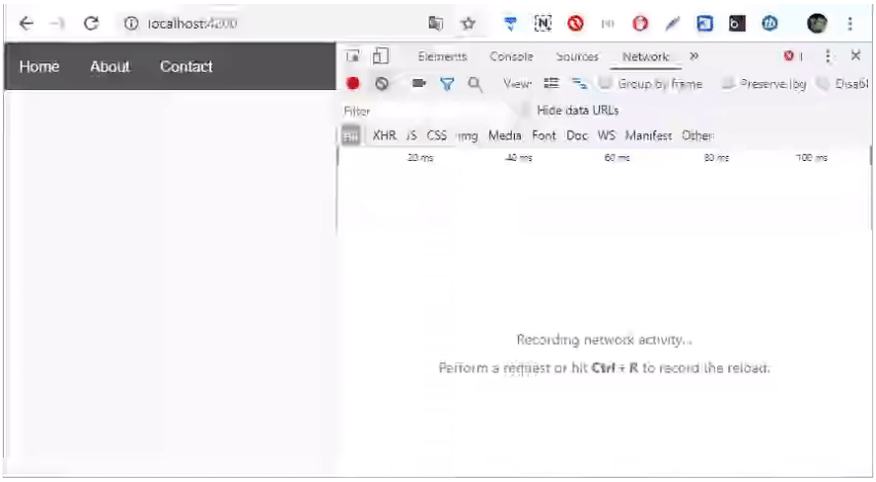
No componente principal do sistema, o **app.component**.ts, você precisará injetar no seu construtor o `private router: Router` e também criar uma propriedade `public routeLoading: boolean = false;` no mesmo componente, para que seja possível registrar quando começou e terminou de baixar o conteúdo da rota, conforme o código abaixo:

```
import { Component } from '@angular/core';
import { Router, NavigationStart, NavigationEnd, Navigatio

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  public routeLoading: boolean = false;

  constructor(private router: Router) {
    this.router.events.subscribe((event) => {
      if (event instanceof NavigationStart) {
        this.routeLoading = true;
      }

      if (event instanceof NavigationEnd ||
```

Assim como no código **TypeScript**, também teremos que modificar no **Html** do componente, para que a mensagem seja exibida para o usuário quando nossa propriedade “loadingRoute” for verdadeira.

```
<ul>
  <li>
    <a routerLink="/home">Home</a>
  </li>
  <li>
    <a routerLink="/about">About</a>
  </li>
  <li>
    <a routerLink="/contact">Contact</a>
  </li>
</ul>
```

👍 Agora sim, veja como ficou:



A cada vez que uma rota é chamada ele exibe o splash enquanto o conteúdo é baixado.

Observação: Por estarmos rodando a aplicação local e o pacote dos nossos módulos não tenham ficando grandes, o browser baixará muito rápido, portanto a aplicação foi manipulada a ponto de deixa-la mais lenta para poder ser visualizada a transição das rotas. Além disso, também foi modificada a folha de estilo padrão para que o objetivo do artigo fosse passada com mais clareza ao leitor.

Conclusão 📄

A intenção deste artigo foi mostrar, de forma simples, que quando trabalhamos com o conceito de modularização temos a oportunidade de carrega-los quando necessário. Além de deixar a aplicação mais leve e independente a tornamos mais responsável, já que fica a mesma fica totalmente “on demand”. Assim o browser só consumirá informações que realmente forem necessárias para aquela situação.

Espero ter ajudado, até a próxima! ✋