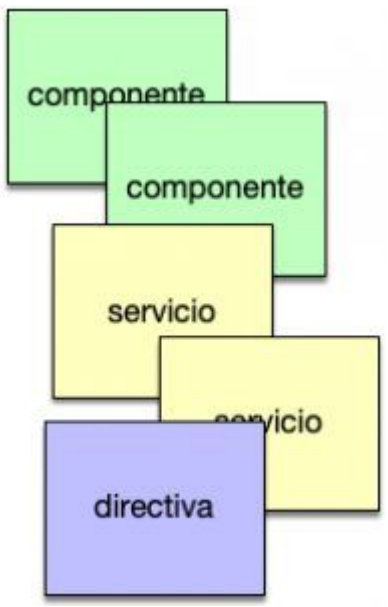
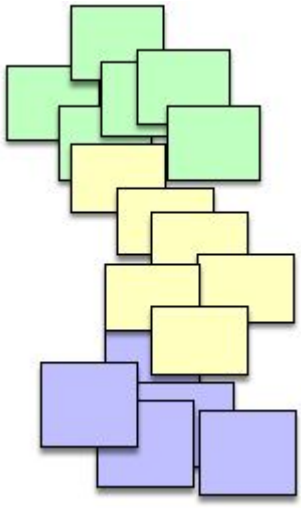
Tabla de Contenidos

- Angular Lazy Loading Modules
- Angular Router y Lazy Modules
- Lazy Modules y PreLoad

El concepto de Angular Lazy Loading Modules es un concepto importante a nivel de programación en el mundo de las arquitecturas SPA . Cuando nosotros construimos una aplicación en Angular esta aplicación esta compuesta por un montón de componentes , servicios directivas etc.

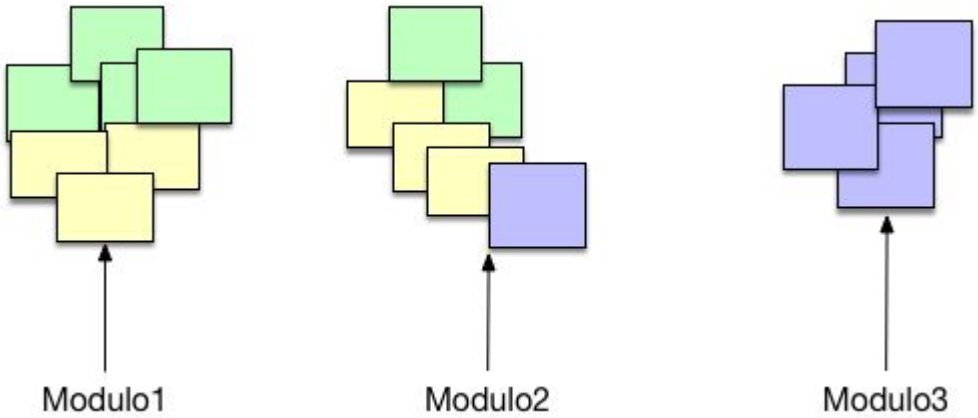


Esto no es muy problemático para una aplicación de tamaño pequeño o mediano pero según la aplicación crece se convierte en algo importante ya que hay que cargar en el navegador mucho código de JavaScript de golpe y los tiempos de carga se pueden ver afectados.



Angular Lazy Loading Modules

Para solventar este problema el primer enfoque es organizar nuestros componentes y servicios en módulos de tal forma que estos se puedan cargar de forma vaga en un futuro y no en el momento de arranque de la aplicación.



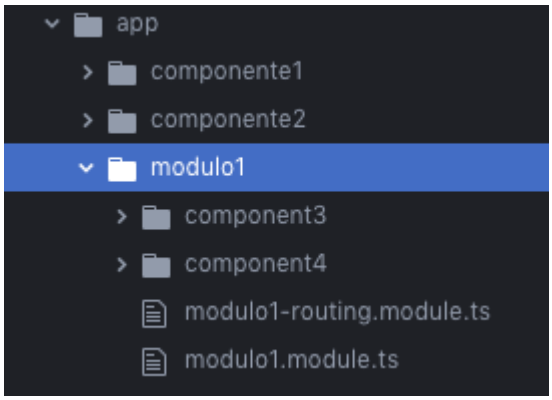
Angular Router y Lazy Modules

Para configurar la carga vaga de módulos tendremos que generar un nuevo módulo en la aplicación y en él ubicar componentes , servicios o lo que necesitemos.

```
ng generate module Modulo1 --routing
```

En nuestro caso vamos a dividir la aplicación en dos partes , la primera contendrá el

componente1 y componente2 que se cargan de forma automática cuando la aplicación arranca y la otra parte será el componente3 y el componente4 que se cargarán de forma lazy o vaga y se encuentran ubicados en el Módulo1.




```
import { NgModule } from '@angular/core';
import { Routes, RouterModule, Router } from '@angular/router';
import { Componente1Component } from
'./componente1/componente1.component';
import { Componente2Component } from
'./componente2/componente2.component';
import { FormsModule } from '@angular/forms';

const routes: Routes = [
  {path: 'pagina1', component: Componente1Component },
  {path: 'pagina2', component: Componente2Component },
  {path:'modulo1', loadChildren:
'./modulo1/modulo1.module#Modulo1Module'}
];

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

```
})
export class AppRoutingModule { }
```

Cómo se puede observar existen dos rutas iniciales “pagina1” y “pagina2” que se cargan de forma natural. Sin embargo las rutas que caen dentro de la url “/modulo1” , tienen una configuración especial ya que se cargan apoyándonos en loadChildren que carga el módulo de forma vaga. ¿Cómo podemos ver esto en ejecución de una forma rápida? . En este caso es relativamente sencillo ya que nos vale con generar una serie de links en el componente principal que nos permita navegar a las diferentes urls unas estarán precargadas y otras se cargarán en un futuro.

```
<!--The content below is only a placeholder and can be replaced.-->
<nav>
  <p>
    <a routerLink="/pagina1">pagina1</a>
  </p>
  <p>
    <a routerLink="/pagina2">pagina2</a>
  </p>

  <p>
      <a routerLink="/modulo1/pagina3">pagina3</a>
    </p>
    <p>
      <a routerLink="/modulo1/pagina4">pagina4</a>
    </p>
</nav>
<router-outlet></router-outlet>
```

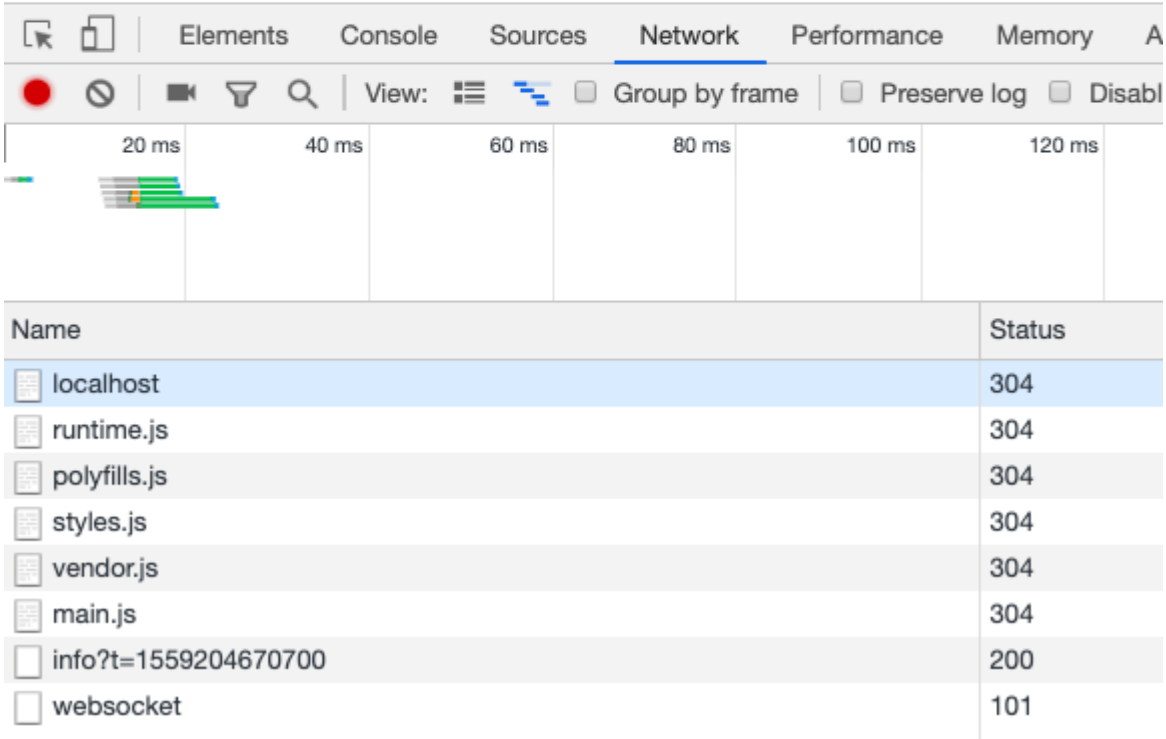
Vamos a comprobarlo, para ello cargamos la página inicial de angular utilizando ng-serve.

pagina1

pagina2

pagina3

pagina4



Como vemos se ha cargado el main.js , si echáramos un vistazo a su contenido podríamos ver que contiene el componente1 y el componente2 que pertenecen al app.module o módulo principal. No hay ninguna referencia al componente3 y al componente4 . Es normal estos componentes hemos solicitado que carguen de forma vaga cuando con el enrutador naveguemos a uno de ellos . Simplemente nos valdrá con pulsar en el link de componente3

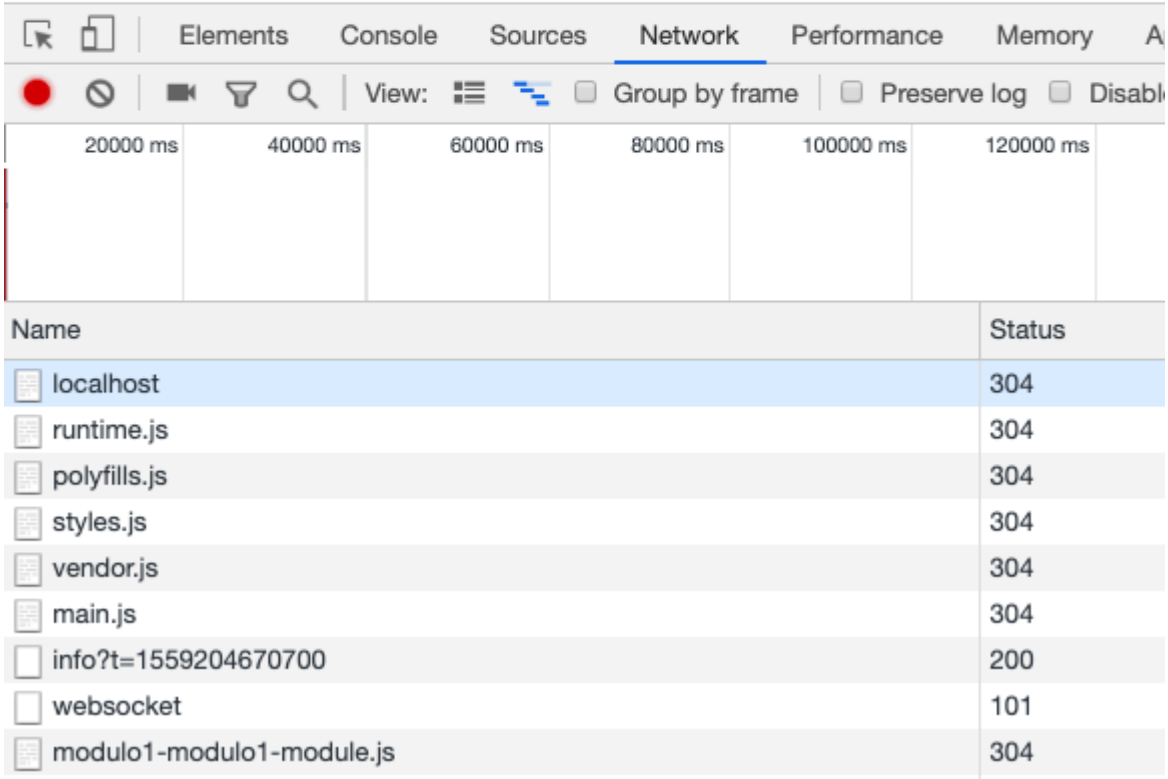
pagina1

pagina2

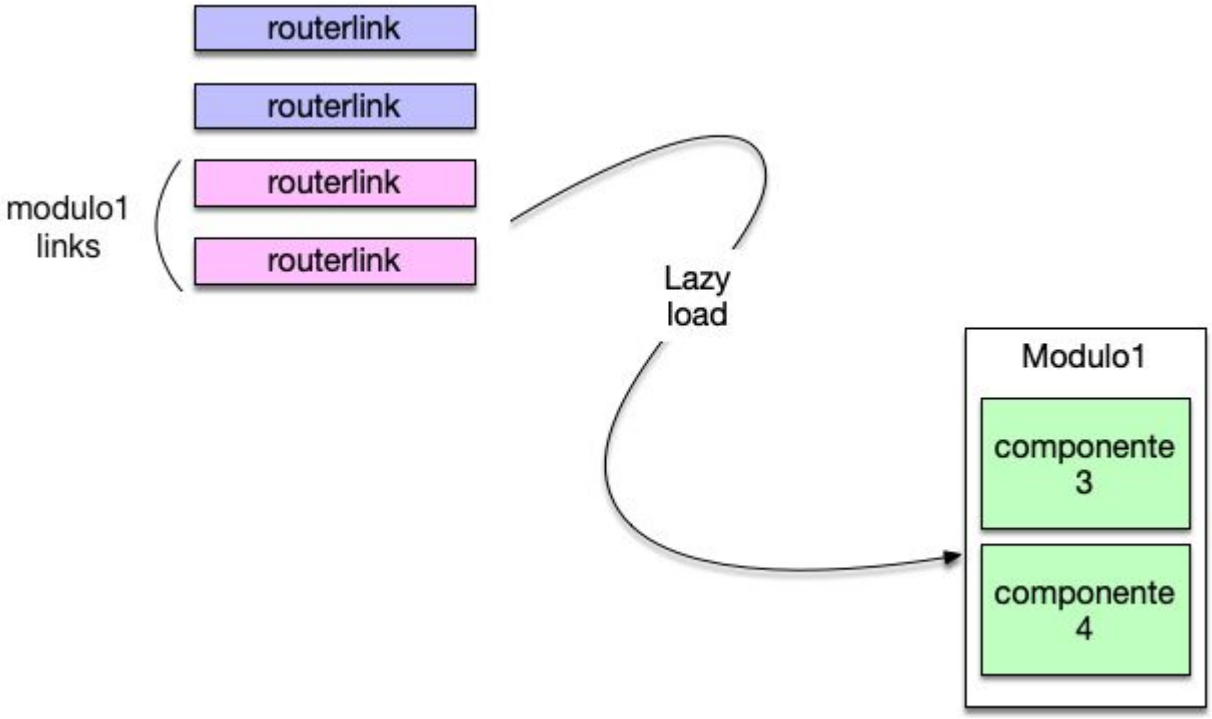
pagina3

pagina4

edad



En cuanto lo pulsemos veremos como un nuevo javascript es solicitado al servidor “modulo1-modulo1.module.js” este javascript se esta cargando de forma vaga al solicitar una navegación con el enrutador.



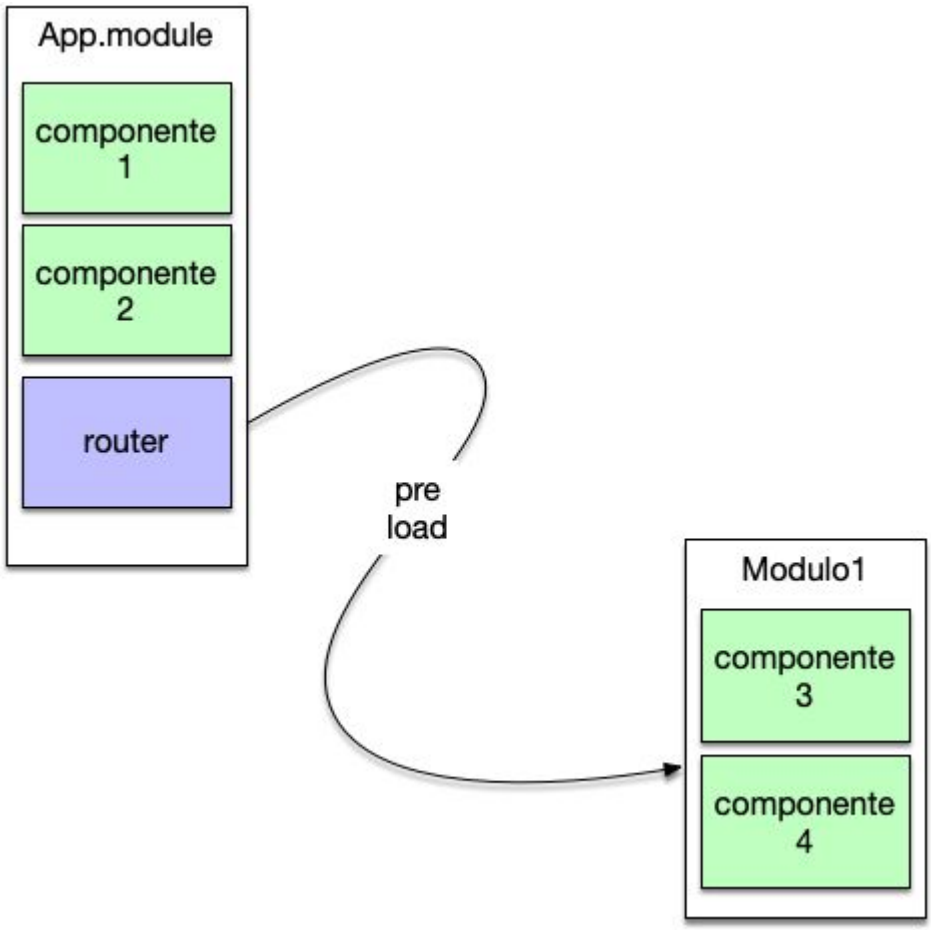
Lazy Modules y PreLoad

En un principio nos pueden parecer estas las dos opciones posibles dentro de Angular , pero no es así . Existe otra opción mas que nos permite realizar un prefeching o preload de esos módulos. ¿Esto para qué sirve? .Muy sencillo normalmente a todos nosotros nos gustaría cargar todo de golpe sin LazyLoad . Lamentablemente esta opción no es siempre posible , para ello existe la capacidad de LazyLoad . Esta se activa una vez que nosotros pulsamos en un link , a través de nuestro router. Si el módulo es muy muy grande , podemos tener un cierto retraso a la hora de ejecutarlo se tiene que cargar primero por completo. Angular nos da la opción de que en cuanto la aplicación se haya cargado únicamente con su módulo inicial empezar a solicitar módulos de forma asíncrona por detrás de forma totalmente transparente. Para ello hay que modificar la configuración del router y añadir la opción de preload.

```
{path:'modulo1', loadChildren:
'./modulo1/modulo1.module#Modulo1Module',data: {preload:true}}
```

De esta forma una vez que el módulo de la aplicación este cargado se empezarán a solicitar

el resto de módulos de forma automática.



Eso sí hay que configurar también el tipo de Estrategia y activar la posibilidad de PreLoadAllModules a nivel del enrutado.

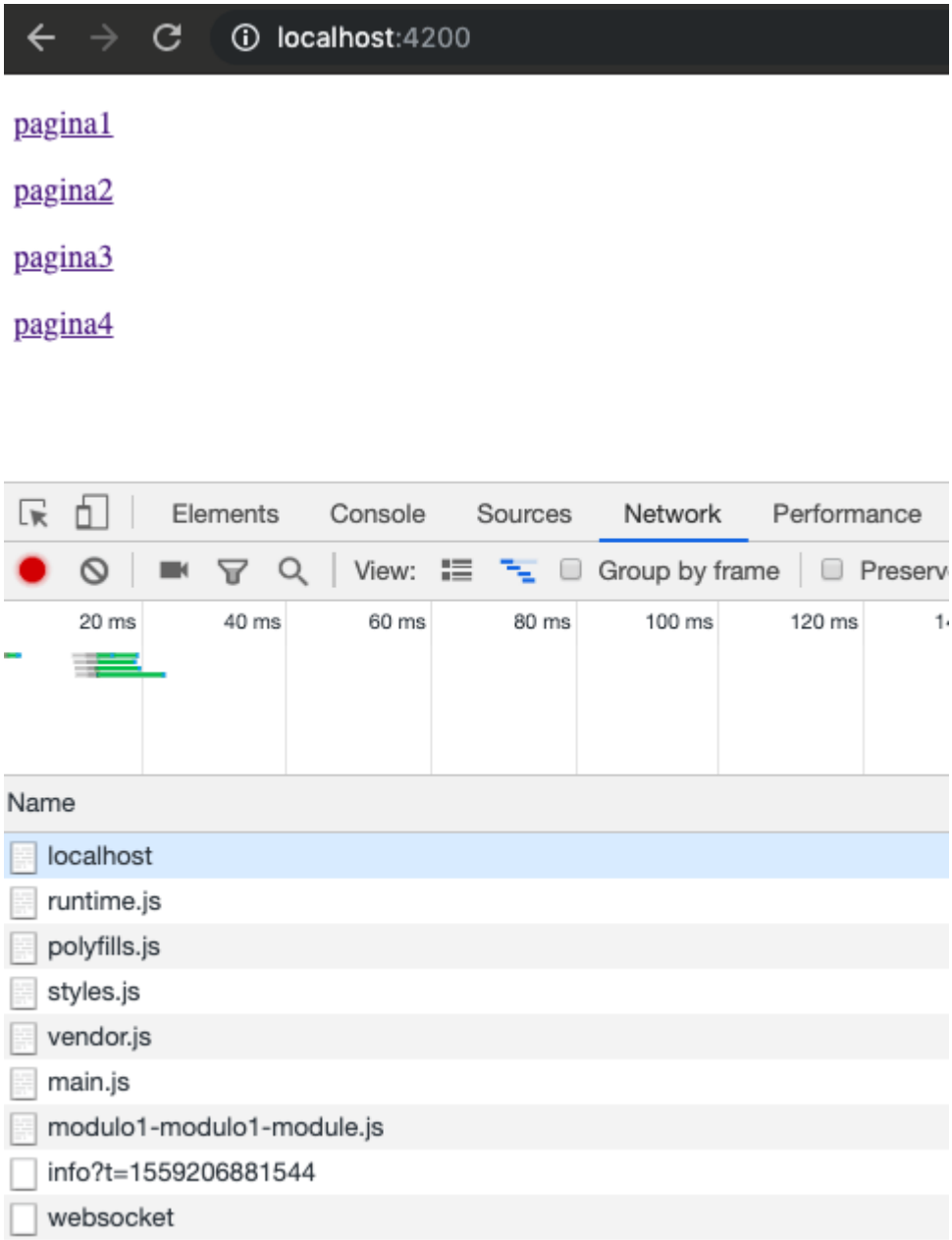
```
const routes: Routes = [
  {path: 'pagina1', component: Componente1Component },
  {path: 'pagina2', component: Componente2Component },
  {path:'modulo1', loadChildren:
'./modulo1/modulo1.module#Modulo1Module', data: {preload:true}}
];

@NgModule({
```

```
  imports: [RouterModule.forRoot(routes, {preloadingStrategy:
PreloadAllModules})],
  exports: [RouterModule],

})
export class AppRoutingModule { }
```

Con esto podremos ver que una vez cargada la página y sin navegar a ningún sitio los módulos se cargan de forma automática.



Acabamos de configurar Angular Lazy Loading Modules con una estrategia de Preload.

CURSO TYPESCRIPT
GRATIS

APUNTATE!!

Otros artículos relacionados

1. Angular custom Directive y sus tipos
2. Angular Services Singletons o no?
3. Angular Modules y el uso de servicios
4. AngularPreload