[CGA-JAVA-WFDA] Web Front-end Development Angular 2.1

[Bài đọc] Module trong Angular

Module là gì?

Ngay khi bắt đầu khởi tạo một dự án Angular, đã có ngay một module mặc định đó là AppModule và các thành phần của nó bao gồm

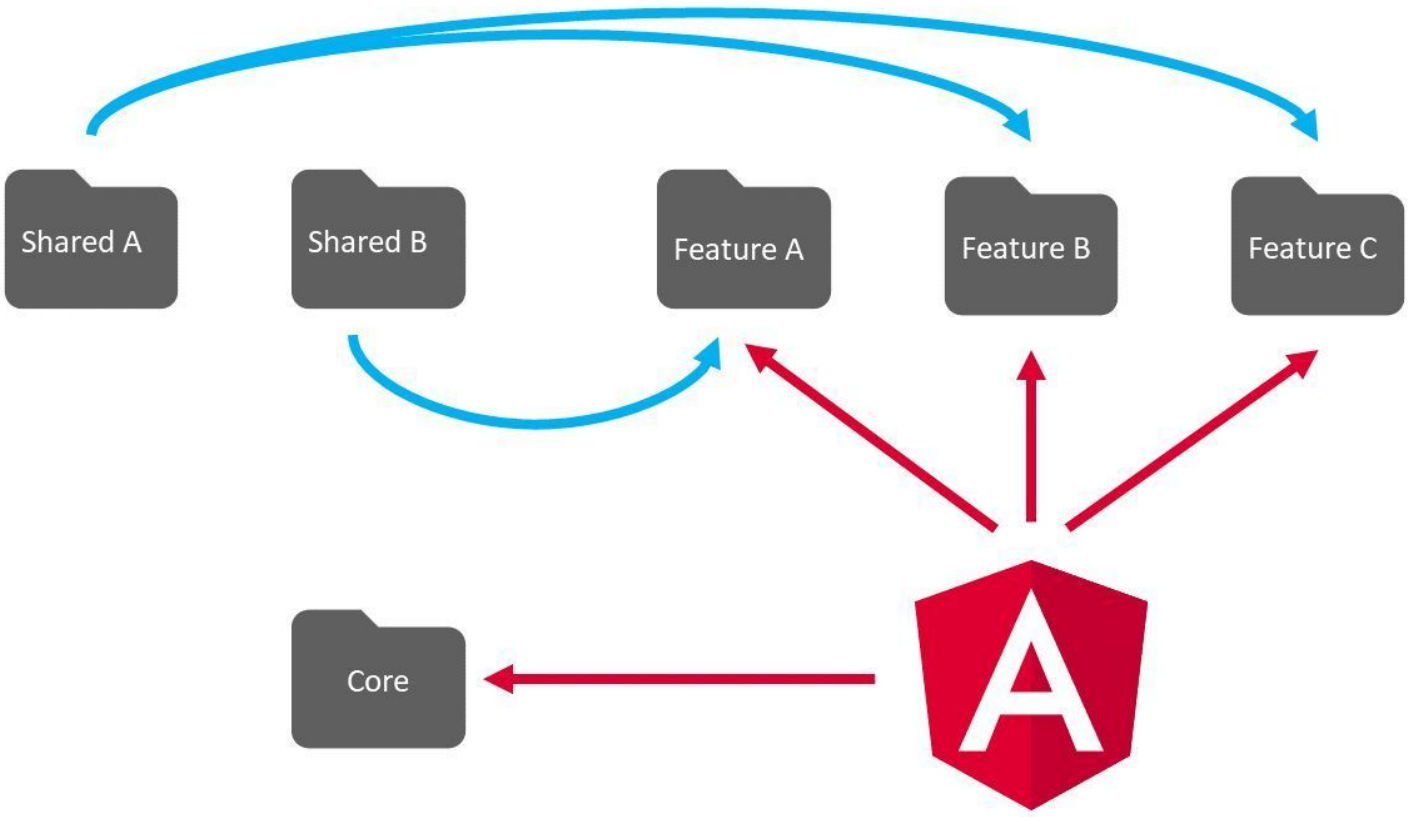
- declarations: Dùng để khai báo những thành phần chúng ta sẽ dùng ở trên template
- providers: Dùng để khai báo các service dùng trong toàn bộ các module của con
- imports: Nó là một mảng các module cần thiết để được sử dụng trong ứng dụng.
- bootstrap: Định nghĩa component gốc của module

Để tạo module sử dụng câu lệnh ng generate module ten_module (hoặc ng g m tên_module)

Feature Module: Gom các component hoặc service có liên quan đến nhau hoặc cùng nằm trong một feature nào đó thành một nhóm

Có những loại module nào?

- modules of pages
- modules of global services
- modules of reusable components



Việc tách ra các module có một tầm ảnh hưởng rất lớn đến việc phát triển một dự án angular nếu phân chia nó hợp lý và khoa học, một dự án sẽ phát triển dễ dàng, dễ bảo trì, dễ tiếp cận. Khi khởi tạo một dự án hãy xem xét kĩ, và nên tạo ra một rule thống nhất trong quá trình phát triển để khi mỗi người tạo một module hay thêm những component sẽ không làm phá vỡ các quy tắc mọi người đang làm.

2. Scope của các thành phần trong Module

Chúng ta sẽ bắt đầu nhầm lẫn khi scope của component/directive/pipe sẽ không giống với service.

Scope của những component (được khai báo trong thuộc tính declarations) chỉ có thể sử dụng được trong nội bộ module đó.

Scope của những service (được khai báo trong thuộc tính providers) có thể sử dụng trong toàn bộ dự án. Điều đó có nghĩa là đối với feature module (không lazy load) các service chỉ cần được khai báo ở bất kỳ module con nào, khi import vào module chính sẽ public single instance trên toàn bộ module con và module chính.

Ví dụ, có 1 accountService, và 2 module userModule và orderModule được inject vào AppModule. Chỉ cần khai báo đăng ký providers cho accountService ở bất kỳ feature module nào (hoặc có thể khai báo vào appModule) đều có thể sử dụng service đó ở bất kì module nào.

Vậy khi nào nên import module nào?

- Nếu module đó được import để sử dụng các component, chúng ta sẽ phải import vào các module nào chúng ta muốn sử dụng vì scope của component chỉ có scope locally.
- Nếu module đó được import để sử dụng các service, chúng ta chỉ nên import nó 1 lần trong module chính. Nếu không, chúng ta có thể sẽ gặp phải lỗi không tìm thấy component vì chúng ta chưa import module đó vào

gặp phải lỗi không tìm thấy component vì chúng ta chưa import module đó vào.

3.Lazy load module và chia module kèm routing

Bình thường với những Angular App ít chức năng, ít module và route thì chúng ta có thể import tất các Module vào AppModule và thêm route vào ngay AppRoutingModule. Tuy nhiên với những dự án lớn việc load tất cả các module cùng lúc khi tải trang sẽ có thể gây chậm trễ làm giảm trải nghiệm người dùng, hơn nữa việc viết tất cả các route vào AppRoutingModule cũng sẽ gây khó trong quá trình phát triển file quá dài cũng gây ức chế trong quá trình phát triển.

Ví dụ một app về giao dục có 3 role là admin, teacher và student. Với mỗi route này chia nó thành mỗi feature module, trong mỗi feature module đó sẽ có một file route nằm trong module đó, để định nghĩa các route liên quan đến các màn hình trong role đó

Ở phần AdminRoute sẽ làm như sau:

```
const routes: Routes = [
  {
    path: 'users',
    component: UsersListComponent,
  },
  {
    path: 'users/:id',
    component: UserDetailComponent
  },
  {
    path: '',
    component: AdminHomeComponent,
  }
];

@NgModule({
  declarations: [],
  imports: [
    CommonModule,
    RouterModule.forChild(routes)
  ],
  exports: [RouterModule]
})
export class AdminRouting { }
```

Chú ý : `RouterModule.forChild(routes)`

```
@NgModule({
  declarations: [UsersListComponent, UserDetailComponent, AdminHomeComponent],
  imports: [
    CommonModule,
    AdminRouting
  ],
  providers: [
    AdminService
  ]
})
export class AdminModule { }
```

```
const routes: Routes = [
  {
    path: '',
    children: [
      {
        path: 'admin',
        loadChildren: () => import('./admin/admin.module').then(m => m.AdminModule)
      },
      {
        path: 'teacher',
        loadChildren: () => import('./teacher/teacher.module').then(m => m.TeacherModule)
      },
      {
        path: 'student',
        loadChildren: () => import('./student/student.module').then(m => m.StudentModule)
      }
    ]
  }
];

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

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

Như cách implement trên, ta có thể hiểu rằng với role admin khi ta tra cập các trang trong module đó thì path lúc nào cũng băt đầu bằng "admin" ví dụ 'admin/users', 'admin/users/1', tương tự với student và teacher

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

Với 3 FeatureModule kia chúng không cần phải thêm trực tiếp vào AppModule, với cách implement kia ta đã thực việc được LazyLoad Module, nghĩa là khi nào người dùng vào những đường dẫn của từng feature thì module đó mới được load, tức là Khi vào đường dẫn ban đầu ví dụ: localhost:4200, thì những Feature Module kia đều chưa được load, khi vào đường dẫn chẳng hạn localhost:4200/admin/users thì module admin mới được load

Vậy với cách implement này, có gì khác với cách import tất cả vào AppModule?

- Đối với component, sẽ không có thay đổi, chúng ta vẫn import module như ở trên

Đối với service, sẽ có chút khác biệt

- Chúng ta vẫn có thể thấy được những service được provide ở AppModule. Tuy nhiên những service chúng ta provide trong lazy load module sẽ có tính chất: (1) Nó chỉ available trong lazy load module trở xuống, các module bên ngoài sẽ không thấy và (2) Nếu lazy load module provide service giống với AppModule thì nó sẽ tạo ra instance mới.

4. Preload

Khi ứng dụng được load, tất cả các components của các lazy module đều chưa được load, mà chúng chỉ thực sự được load khi mà người dùng sử dụng đến chúng.

Như vậy là chúng ta đã Lazy Load thành công thì cải thiện được đáng kể tốc độ load lần đầu của ứng dụng. Tuy nhiên, có một điều vẫn chưa được tốt cho lắm, đó là các module chỉ được load khi mà người dùng sử dụng đến. Điều này có thể dẫn đến sự chậm trễ khi mà người dùng phải chờ đợi cho module đó được load, làm cho trải nghiệm người dùng xấu đi. Và preload được sử dụng để load ngầm những module đó

Đây là cách để preload tất cả các Module trong Angular app

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

Tuy nhiên, preload tất cả các lazy load module không phải lúc nào cũng là sự lựa chọn đúng đắn. Đặc biệt đối với các thiết bị di động hay những kết nối băng thông thấp. Chúng ta có thể sẽ phải tải những modules mà người dùng có thể rất ít khi chuyển hướng đến. Tìm ra sự cân bằng cả về hiệu năng và trải nghiệm người dùng là chìa khóa cho việc phát triển.

Ví dụ, trong ứng dụng email, chúng ta có 2 lazy load modules:

Settings module Email module Đây là một ứng dụng email client nên module Email sẽ được sử dụng rất rất thường xuyên. Tuy nhiên module Setings sẽ được người dùng sử dụng nhưng với tuần suất rất thấp. Do vậy mà việc preload module Email sẽ đem lại hiệu quả cao, trong khi với module Setings thì thấp.

Angular cung cấp một cách extend PreloadingStrategy để xác định một tùy chỉnh chiến lược Preload chỉ ra điều kiện cho việc preload các lazy load module. Chúng ta sẽ tạo một provider extend từ PreloadingStrategy để preload các modules mà có preload: true được xác định trong cấu hình route.

Last modified: Sunday, 14 March 2021, 4:51 PM

CHƯƠNG TRÌNH

- Career
- Premium
- Accelerator

TÀI NGUYÊN

- Blog
- Tạp chí Lập trình
- AgileBreakfast

Follow Us