MODULES FONCTIONNELS

ULT – DÉPARTEMENT GÉNIE INFORMATIQUE

Module: Projet Framework et Développement

Public cible: 5° Génie Logiciel

Houcem Hedhly

houssem.hedhli@ult-tunisie.com

2021/2022

PLAN

I. Principe de base
II. Avantages
III. Implémentation
IV. Lazy Loading

I. PRINCIPE DE BASE

- Application Angular = au moins le module principal `AppModule`.
- Mauvaise pratique: coder toute une application dans le module principal (ça dépend de la taille de l'application).
- Bonne pratique: diviser le code dans des modules séparés (généralement par fonctionnalité).
- Créer un module fonctionnel (Feature Module) pour chaque partie de l'application.
- Les modules fonctionnels peuvent collaborer entre eux, et avec le `AppModule`.

II. AVANTAGES

- Améliore la productivité et la collaboration.
- Optimiser la taille de de `AppModule`.
- Clarté du code.
- Localiser le code plus rapidement
- Lazy Loading

III. IMPLÉMENTATION

- Générer un nouveau module à l'aide de la CLI:

```
$ ng generate module module-name --route=path --module=app
```

ou bien

```
$ ng g m module-name --route=path -m=app
```

- `--route=path` : permet de définir le chemin du module pour le lazy loading.
- `--module=app` : permet de préciser le module qui déclare le module fonctionnel.
- Si on souhaite générer un module de routing séparé on rajoute le flag `--routing`

III. IMPLÉMENTATION

- Pour ajouter des components à ce module on utilise la commande

```
$ ng generate component module-name/component-name
```

ou bien

```
$ ng generate component component-name --module=module-name
```

- Le nouveau component sera ajouté dans le module qui l'a déclaré.
- le module sera à son tour ajouté dans le module qui l'a déclaré.

IV. LAZY LOADING

- Par défaut, les `NgModules` sont chargés en mode Eager Loading.
- Tous les modules sont chargés dès le démarrage de l'application.
- Il faut considerer de différer le chargement de certains modules si la taille de l'application est important.
- Le chargement en différé (Lazy Loading) aide à diminuer la taille des packages de démarrage de l'application → réduire le temps de démarrage.

IV. LAZY LOADING

- Utiliser `loadChildren` au lieu de `component` dans le route:

```
const routes: Routes = [
  {
    path: 'items',
    loadChildren: () => import('./items/items.module')
                          .then(m => m.ItemsModule) },
];
```

IV. LAZY LOADING

- Le module de routing dans le `AppModule` utilise **`forRoot()`**.
- Indiquer que c'est le module de routing principal dans l'application.
- Ne peut pas être utilisé qu'une seule fois dans une application.
- Les modules de routing dans les modules fonctionnels utilisent **`forChild()`**.
- Indiquer que c'est un module de routing supplémentaire à l'intérieur d'un module fonctionnel.
- Peut être utilisé autant qu'il y a des modules fonctionnels dans l'application.