Cours 4 : JSON et plus

Mardi 24 Novembre 2020

Antoine Flotte (aflotte@excilys.com)

Excilys
Développeurs de passion

SOMMAIRE

package.json

Qu’est ce ?

Dans votre projet vous avez un fichier package.json.

Npm est un gestionnaire de paquets donc on peut ajouter des librairies qu’il va charger pour nous avec :

npm install

Et le package.json est l’équivalent de votre pom, il contient la liste de ces librairies

Si vous avez besoin de borner une version :

"@angular/cli": "~9.0.7", = version < 9.1.0 donc mineur fixe

"codelyzer": "^5.1.2", = version < 6.0.0 donc majeur fixe

"@angular/cli": "9.0.7" version fixe

Il y a aussi un fichier package-lock.json dans votre projet

Il liste les versions exacte que vous utilisez, avec les versions des dépendances de vos dépendances.

Cela permet de s’assurer d’avoir un comportement fixe.

Il faut donc le garder sur votre git.

Dans votre projet vous avez un dossier node_modules qui va contenir toutes les dépendances de votre projet, elle sont chargé en fonction de votre package.json quand vous faite :

npm install

Pour ajouter une dépendance neccessaire :

npm install dependanceName

Pour qu'elle s'ajoute à votre package.json :

npm install dependanceName --save

i18n

Faire :

npm install @ngx-translate/core --save

Ajouter dans les imports de app.module.ts : TranslateModule.forRoot() et

import {TranslateModule} from '@ngx-translate/core';

Dans le dossier src/asset créer un dossier i18n qui va contenir en.json et fr.json

Loader

Vous avez besoin d'un loader :

npm install @ngx-translate/http-loader --save

Puis dans votre app.module.ts ajouter au début :

```
import {HttpClient} from '@angular/common/http';
import {TranslateHttpLoader} from '@ngx-translate/http-loader';

export function createTranslateLoader(http: HttpClient) {
 return new TranslateHttpLoader(http, './assets/i18n/', '.json');
}
```

Puis remplacer dans imports dans app.module.ts le :

```
TranslateModule.forRoot()
```

par :

```
   TranslateModule.forRoot({
      loader: {
          provide: TranslateLoader,
          useFactory: (createTranslateLoader),
          deps: [HttpClient]
      }
  })
```

Dans votre html vous pouvez appeler des pipes :

{{ 'val' | function }} va appeler la fonction avec ‘val’ en parametttre avant de l’afficher

Translate string

```
<div>{{ 'HELLO' | translate }}</div>
```

pour traduire une string suivant le langage actuel

lang.json

en.json

```
{
    "HELLO": "hello"
}
```

fr.json

```
{
    "HELLO": "salut"
}
```

Changer langue

Pour changer la langue utiliser :

```
this.translate.use("fr");
```

La documentation officiel :

https://github.com/ngx-translate/core

De la même façon vous pouvez formater les dates avec:

{{ birthday | date:"MM/dd/yy" }}

fichier de configuration

Ajouter un fichier dans le dossier environnements (dans le dossier src) :

configuration.ts

qui contient :

```
const baseUrl = 'http://localhost:8080';
export const environment = {
 production: false,
 filmUrl: baseUrl + '/film'
};
```

Et là où vous en avez besoin :

```
import { environment } from 'src/environments/environment';
```

Puis :

```
base_url = `${environment.filmUrl}`;
```

Service

ng generate service filmWS

Puis dans le fichier .ts qui a été créé vous mettez vos fonctions d'appel au back qui retourne un Observable

Et dans vos composants vous appelez le service en question en le mettant dans le constructor() (comme pour le http) et vous pouvez maintenant appeler ses fonctions avec this.serviceName.function()