6. Service HTTP

Introduction

La plus part des applications Web doivent pouvoir communiquer avec un backend au travers du protocol HTTP(S). Les applications étant monopage, les appels doivent être réalisés sans rechargement de page. Angular met à disposition un module `HTTPClientModule` permettant d'outiler la création de requêtes HTTP, et des libraires comme RxJs ou les Promise ES6 qui permettent de faire ces appels asynchrones.

Module Angular HTTPClientModule

Présentation

le module `HttpClientModule` met à disposition le service `HttpClient` permettant d'effectuer des requêtes HTTP. Ce service est basé sur l'objet natif Javascript `XMLHttpRequest` et offre donc les fonctionnalités disponibles au travers de son API. Le service met à disposition l'ensemble des méthodes :

- get
- post
- put
- delete
- patch
- head
- jsonp

L'object `HttpClient` est disponible par injection à partir du moment où `HttpClientModule` est référencé dans les `imports` du `module`. Il peut donc être disponible dans un service personnalisé par injection dans le constructeur.

Exemple :

```
import { HttpClient } from "@angular/common/http";

export class MyService {
  constructor(private http: HttpClient) {}
}
```

Documentation officielle

Header

Il est possible de personnaliser l'entête HTTP au travers de l'Objet `HttpHeaders`. Attention cet objet est immutable, lorsque vous effectuez une modification sur l'objet, l'objet lui même n'est pas modifié, mais il renvoie une nouvelle copie de l'objet intégrant la modification.

L'objet header devra être intégré à la propriété `options` lors de l'invocation de la méthode get, post, etc.

Exemple :

```
import { HttpHeaders } from "@angular/common/http";

const httpOptions = {
  headers: new HttpHeaders({
    "Content-Type": "application/json",
    Authorization: "my-auth-token",
  }),
};
```

https://angular.io/api/common/http/HttpHeaders

Paramètres

L'envoi de paramètres est réalisé de façon classique, soit directement dans l'URL pour la méthode get, soit dans le `body` pour les autres méthodes post, put, delete.

Exemple : ici nous envoyons dans le `body` la paramètre `name`.

```
this.http.post<any>("mon-api/mon-service", { name: "Toto" }, httpOptions);
```

Format de la réponse

Vous pouvez modifier le format de retour (si le service le permet) attendu par le client avec la propriété `responseType`. Celle-ci est à ajouter dans les options d'envoi.

Exemple :

```
const httpOptions = {
  headers: new HttpHeaders({
    "Content-Type": "application/json",
    Authorization: "my-auth-token",
  }),
  responseType: "text",
};
```

Par défaut le format est défini comme étant du `json`.

Envoi de la requête

L'envoi de la requête est réalisée au travers de l'objet `HttpClient`.

Exemple :

```
this.http.get<any>("mes-assets/mon-fichier.json", httpOptions);
this.http.post<any>("mon-api/mon-client/", { name: "Toto" }, httpOptions);
```

```
this.http.put<any>("mon-api/mon-client/", { id: 1, name: "Titi" }, httpOptions);
this.http.delete<any>("mon-api/mon-client/1", httpOptions);
```

Récupération asynchrone

Une fois l'envoi réalisé, la récupération du résultat est effectuée de manière asynchrone, il faut donc s'abonner (avec erreur ou succès) au retour du service. Pour cela on utilisera les promesses (Promise) ou les Observables de RxJs.

Exemple :

```
this.httpClient
  .get<any>("mes-assets/mon-fichier.json")
  .toPromise()
  .then((result: any) => {
    // result contient le flux JSON
  })
  .catch((error) => {
    // une erreur s'est produite
  });
```

Plus d'information sur les Promise.
Plus d'informations sur les Observables

Exercice

Présentation

Application Météo.

L'objectif de cet excercice est de charger la météo au travers du module Angular HTTP et des promesses.

Etapes d'injection du client HTTP

1. Importer le module Angular `HttpClientModule` .
2. Injecter le service `HttpClient` dans le service `meteo.service.ts`
3. Exploiter le service `HttpClient` en lieu et place de `fetch` , et utiliser la methode `toPromise()` pour résoudre la promesse, et récupérer le résultat.
4. Tester l'appel au service dans `meteo-view.component` , et souscrivez à la promesse pour récupérer les résultats. La modification doit être transparente pour le contrôleur.

Bravo, vous exploitez le gestionnaire de requête HTTP préconnisé pour Angular ! Nous allons regarder une alternative à la création de promesse, via RxJs.

Correction disponible ici