



Cours Angular

4.2

| TD Hackers

| Routage – Service – Stockage des données

1] Service pour l'API

Dans le précédent TD, nous avons vu l'utilisation d'un service avec la création manuelle d'un objet `Hacker`. Nous allons voir maintenant le service spécifique à l'interrogation de l'API externe.

- **Mise en place du module HTTP**

L'accès pour notre application va se faire à l'aide du module Angular `HttpClientModule`. **Attention :** ce module subit souvent des modifications tel que décrit ici :

<https://angular.io/guide/deprecations#angularhttp>

Beaucoup de tutos sur Internet n'intègrent pas les nouveautés...

L'application porte sur la recherche de l'origine des hackers cherchant à pirater un serveur.

À partir d'une IP fournie par un outil comme **fail2ban** (http://www.fail2ban.org/wiki/index.php/Main_Page), l'application effectuera une recherche sur le pays d'origine via une API externe de localisation :

<https://freegeoip.app/json/>.

T1.1 Au niveau du fichier d'intégration du module `app.module.ts`, importez `HttpClientModule` :

```
...
import { HttpClientModule } from '@angular/common/http';
```

Et déclarez-le en l'ajoutant dans la section `imports` :

```
imports: [
  BrowserModule,
  AppRoutingModule,
  HttpClientModule
],
```

- **Création du service**

Voici d'abord une liste d'IPs correspondant réellement à des attaques sur mon serveur :

```
51.79.197.112
185.238.29.11
182.254.164.34
47.75.221.20
129.28.162.79
79.9.171.88
```

T1.2 Générez le nouveau service que l'on nommera `lookup` :

```
angular$ ng generate service services/lookup
```

T1.3 Remplacez l'intégralité du code `lookup.service.ts` par celui-ci :

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from 'src/environments/environment';

@Injectable({
  providedIn: 'root'
})

export class LookupService {

  constructor(private http: HttpClient) { }

  public sendGetRequest(IP: string){
    return this.http.get(environment.apiUrl + IP);
  }
}
```

Explications :

- On importe le module `Environment` pour définir un environnement local à l'application (ici pour mettre l'URL de l'API).
- On permet l'utilisation du module HTTP par l'injection dans le **constructeur** du service.
- la méthode `sendGetRequest()` se charge de la requête à partir d'une IP fournie.

T1.4 Il manque dans `environment.ts`, l'initialisation de l'URL de l'API :

```
export const environment = {
  production: false,
  apiUrl: 'https://freegeoip.app/json/'
};
```

Ce fichier, comme son nom l'indique sert à référencer au niveau global des variables d'environnement pour un projet Angular.

2] Saisie de l'IP pour une requête

Pour tester une requête, il va falloir créer un nouveau composant chargé de la saisie de l'IP.

- **Initialisations préalables**

L'application comportera dorénavant **trois pages** : **Home** (page d'accueil), **Liste** (l'affichage des serveurs attaquants) et **Saisie** permettant la recherche des renseignements d'un serveur à partir d'une IP.

T2.1 Créez le nouveau composant `saisie`.**T2.2** Intégrez la nouvelle page au routage et au menu.

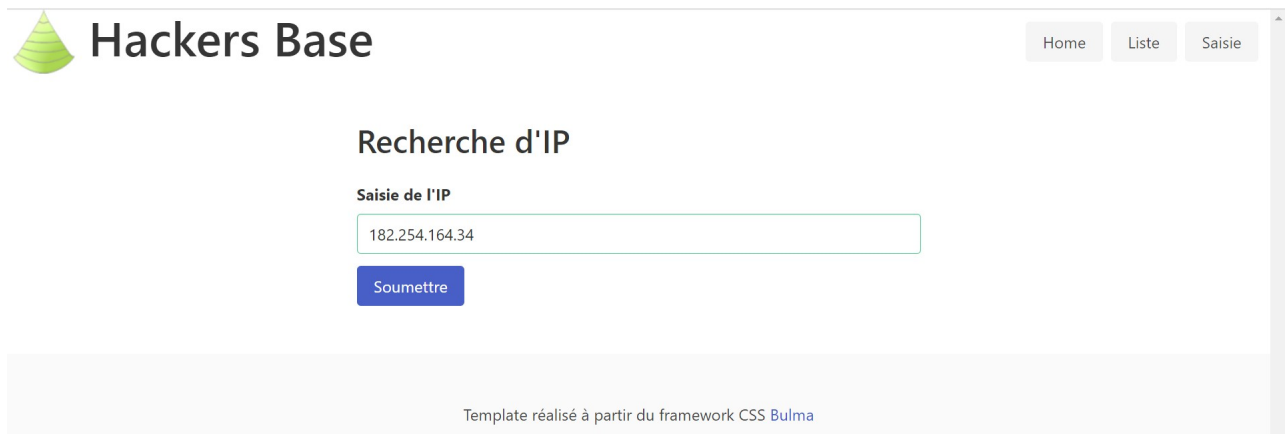
T2.3 Puisque nous activons les syntaxes de formulaire réactives pour Angular, il faut donc importer le module `ReactiveFormsModule` dans le fichier `app.module.ts`, faites-le...

Info : la façon de le faire, ainsi que la suite a été vue dans le **TD 2.2**.

- **Création du formulaire**

La page `Saisie` comportera un formulaire très simple, ne comportant qu'un champ de type texte et servant à l'entrée de l'adresse IP de l'attaquant.

T2.4 Complétez le code de `saisie.component.html` et de `saisie.component.ts` afin d'obtenir ceci :



The screenshot shows a web application titled "Hackers Base". It has a navigation bar with three buttons: "Home", "Liste", and "Saisie". The main content area is titled "Recherche d'IP". Below this title, there is a section labeled "Saisie de l'IP" containing a text input field with the value "182.254.164.34" and a blue button labeled "Soumettre". At the bottom of the page, there is a footer that reads "Template réalisé à partir du framework CSS Bulma".

A ce stade, il n'y a pas de validation autre que la non saisie de l'IP. Plus tard, en exercice facultatif vous vous exercerez à ajouter un autre validateur correspondant au bon formatage d'une IPv4...

- **Utilisation du service HTTP**

Le composant `Saisie`, outre la saisie de l'IP, devra faire appel au service `lookup` afin de récupérer les informations puis créer un objet `Hacker` et l'ajouter au tableau.

T2.5 Importez les deux services `HackerService` et `LookupService` dans le fichier `saisie.component.ts` ; importez aussi la classe `Hacker` car nous en avons besoin.

La suite va traiter de l'injection dans le constructeur des deux services.

T2.6 Changez la la ligne de code (toujours dans `saisie.component.ts`) du constructeur :

```
constructor(private hackerService:HackerService,private lookupService:LookupService) {}
```

La méthode liée à `ngSubmit` dans la balise `form` s'intitule `recherche()`. En voici le code intégral :

```
recherche() {  
  this.isSubmitted = true;  
  console.log('IP saisie', this.ipForm.value);  
  
  if (!this.ipForm.value.ip) {  
    this.badIP = true; return;  
  }  
}
```

```

    } else {
      this.lookupService.sendGetRequest(this.ipForm.value.ip).subscribe((data: Object) =>
      {
        let IP: string = "";
        let CN: string = "";
        let RN: string = "";
        let CI: string = "";

        // Tableau des clés et des valeurs de l'objet
        let keys = Object.keys(data);
        let values = Object.values(data);

        for (let i = 0; i < keys.length; i++) {
          if (keys[i] == 'ip') { IP = values[i]; }
          if (keys[i] == 'country_name') { CN = values[i]; }
          if (keys[i] == 'region_name') { RN = values[i]; }
          if (keys[i] == 'city') { CI = values[i]; }
        }

        let leHacker = new Hacker(IP, CN, RN, CI);
        this.hackerService.addHacker(leHacker);
      });
    }
  }
}

```

Explications :

- Cette méthode envoie au service l'IP saisie valide et ne récupère que les quatre éléments nécessaires pour remplir un objet `Hacker`.
- Elle crée ensuite un objet `Hacker` nommé `leHacker` pour ensuite utiliser la méthode d'ajout du service `hackerService` ; à ce stade, la variable globale du tableau des hackers est mise à jour.

Il faut maintenant voir si tout fonctionne et ceci se fera dans le composant `Liste`.

T2.4 Changez le code de `ngOnInit()` dans `liste.component.ts` pour ne laisser que ceci :

```

ngOnInit(): void {
  // Si le tableau est vide, on le dit
  if (this.nboflesHackers > 0) {
    console.log('contenu : ' + this.lesHackers[0].ip);
    console.log('contenu : ' + this.lesHackers[0].country_name);
    console.log('contenu : ' + this.lesHackers[0].region_name);
    console.log('contenu : ' + this.lesHackers[0].city);
  } else {
    console.log('Tableau Hackers vide !');
  }
}

```

Attention : certaines IP ne retournent pas toutes les informations demandées, c'est donc normal.

En allant après une saisie d'IP dans le menu `liste`, vous devriez obtenir comme informations dans la console quelque chose comme ceci :

