



CRUD avec HttpClient

- ▶ **Introduction**
- ▶ **CRUD**
- ▶ **Problème de CORS**





Introduction



Introduction



- Pour appliquer les différentes méthodes : GET, POST, PUT et DELETE du service HttpClient de Angular, il faut:
- 1- Créer un service pour appeler ces différentes méthodes.
- 2- Injecter le service HttpClient dans le service créé sans oublier d'importer le module HttpClientModule.
- 3- Avoir les urls des différentes « api » de CRUD. Angular consomme les méthodes de CRUD créés dans la partie backend.



CRUD



Affichage - READ



Au niveau du service : Définition de méthode

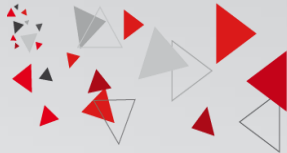
```
apiUrl : string = "http://.....";  
getData(): Observable<Type[]>{  
    return this.http.get<Type[]>(this.apiUrl);  
}
```

Au niveau du composant : consommation de la méthode du service

```
constructor(private _service : CRUDService) { }  
getDataFormService(){  
    this._service.getData().subscribe(res=>traitement);  
}
```



Ajout - Create



Au niveau du service : Définition de méthode

```
apiUrl : string = "http://.....";  
httpOptions = { headers: new HttpHeaders({  
    'Content-Type': 'application/json'})}  
  
addData(myObject:Type): Observable<Type>{  
    return this.http.post<Type>(this.apiUrl, myObject, this.httpOptions); }
```

Au niveau du composant : consommation de la méthode du service

```
constructor(private _service : CRUDService) { }  
addMyObject(obj:Type){  
    this._service.addData(obj).subscribe();  
}
```

► Modification - Update



Au niveau du service : Définition de méthode

```
apiUrl : string = "http://.....";  
httpOptions = { headers: new HttpHeaders({  
  'Content-Type': 'application/json'})}  
  
updateData(id:number,myObject:Type): Observable<Type>{  
  return this.http.put<Type>(this.apiUrl+'/'+ id, myObject, this.httpOptions); }
```

Au niveau du composant : consommation de la méthode du service

```
constructor(private _service : CRUDService) { }  
updateMyObject(id:number,obj:Type){  
  this._service.updateData(id,obj).subscribe();  
}
```


► Suppression - Delete



Au niveau du service : Définition de méthode

```
apiUrl : string = "http://.....";  
deleteData (myObject: Type | number): Observable<Type> {  
  const id = typeof myObject === 'number' ? myObject : myObject.id;  
  return this.http.delete<Type>(this.apiUrl+'/'+id);  
}
```

Au niveau du composant : consommation de la méthode du service

```
constructor(private _service : CRUDService) { }  
deleteMyObject(objToDelete){  
  this._service.deleteData(objToDelete).subscribe();  
}
```



Problème de CORS

Définition

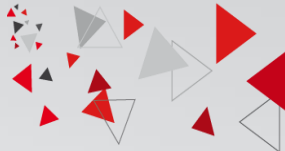


Le « *Cross-origin resource sharing* » (CORS) est un mécanisme qui consiste à ajouter des en-têtes HTTP afin de permettre à un agent utilisateur d'accéder à des ressources d'un serveur situé sur une autre origine que le site courant. Un agent utilisateur réalise une requête HTTP **multi-origine (*cross-origin*)** lorsqu'il demande une ressource provenant d'un domaine, d'un protocole ou d'un port différent de ceux utilisés pour la page courante

Définition



- Pour des raisons de sécurité, les requêtes HTTP multi-origine émises depuis les scripts sont restreintes.
- Ainsi XMLHttpRequest et l'API Fetch respecte la règle d'origine unique.
- Cela signifie qu'une application web qui utilise ces API peut uniquement émettre des requêtes vers la même origine que celle à partir de laquelle l'application a été chargée, sauf si des en-têtes CORS sont utilisés.



- Lorsque nous développons une application Angular qui a besoin d'un back-end pour conserver les données, le back-end est souvent servi sur un autre port de localhost.
- Par exemple, l'URL de l'application angular frontale est `http://localhost:4200`, tandis que l'URL du serveur principal est `http://localhost:3000`.
- Dans ce cas, si nous effectuons une requête HTTP de l'application frontale vers le serveur principal, il s'agit d'une requête interdomaine et nous devons effectuer un travail supplémentaire pour y parvenir

Solutions



Il y a deux solutions, nous pouvons utiliser

- CORS (Solution coté backend)
- un proxy côté serveur (Solution coté Angular)



Néthographie



- <https://developer.mozilla.org/fr/docs/Web/HTTP/CORS>
- <https://betterprogramming.pub/setup-a-proxy-for-api-calls-for-your-angular-cli-app-6566c02a8c4d>



► **Merci de votre attention**