HttpClient and Observables

.NET

Angular provides a client HTTP API for Angular applications. The ***HttpClient*** *service class provides HTTP communication functionality for Angular Programs.*

HTTPS://ANGULAR.IO/TUTORIAL/TOH-PT4#WHY-SERVICES

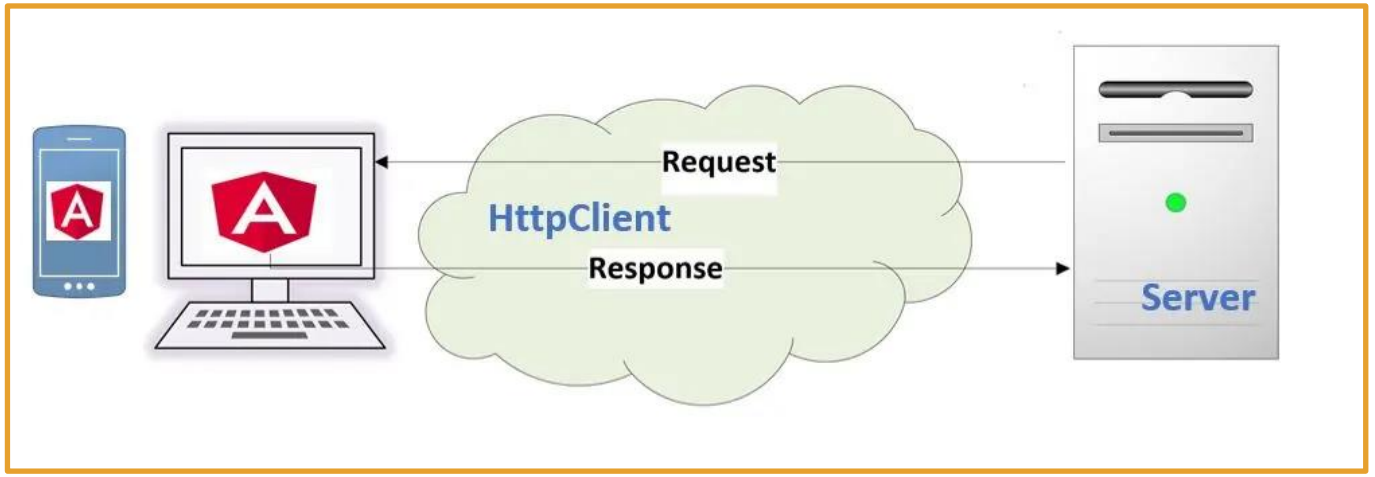
HttpClient - Overview

https://angular.io/guide/http
https://angular.io/start/start-data

Most front-end applications need to communicate with a server over HTTP in order to download or upload data and access other back-end services. Angular provides a client HTTP API for Angular applications, the HttpClient service class in @angular/common/http.

The HTTP client service offers the following major features.

- The ability to request typed response objects.
- Streamlined error handling.
- Testability features.
- Request and response interception.



HttpClient – Set-up

https://angular.io/guide/http#communicating-with-backend-services-using-http

1. Import the Angular ***HttpClientModule*** into app.module.ts.
    1. import { HttpClientModule } from '@angular/common/http';
2. Add HttpClientModule to the imports array of app.module.ts.
3. Import and inject HttpClient as a dependency of a service class.
    1. import { HttpClient } from '@angular/common/http';
    2. constructor(private http: HttpClient) { }
4. Import the following into the service class.
    1. import { Observable, throwError } from 'rxjs';
    2. import { catchError, retry } from 'rxjs/operators';

HttpClient.get<>

https://angular.io/guide/http#requesting-data-from-a-server

HttpClient.get<>() is asynchronous. It sends an HTTP GET request and returns an ***Observable*** that emits the requested data when the response is received. The return type varies based on the observe and responseType values passed to the call.

HttpClient.get<>() takes two arguments; the endpoint URL from which to fetch and an options object that you can use to configure the request.

```
options: {
    headers?: HttpHeaders | {[header: string]: string | string[]},
    observe?: 'body' | 'events' | 'response',
    params?: HttpParams|{[param: string]: string | number | boolean |
ReadonlyArray<string | number | boolean>},
    reportProgress?: boolean,
    responseType?: 'arraybuffer'|'blob'|'json'|'text',
    withCredentials?: boolean,
  }
```

```
configUrl = 'assets/config.json';

getConfig() {
  return this.http.get<Config>(this.configUrl);
}
```

HttpClient.post<>

https://angular.io/guide/http#making-a-post-request

The HttpClient.post<>() method is similar to get<>() in that it has a type parameter which you use to specify the data type you expect the server to return. The method takes 3 parameters.

- a resource URL
- body - The data to POST in the body of the request.
- options - An object containing method options which specify required headers, etc.

```
this.heroesService
  .addHero(newHero)
  .subscribe(hero => this.heroes.push(hero));
```

```
/** POST: add a new hero to the database */
addHero(hero: Hero): Observable<Hero> {
  return this.http.post<Hero>(this.heroesUrl, hero, httpOptions)
    .pipe(
      catchError(this.handleError('addHero', hero))
    );
}
```

HttpClient - Headers

https://angular.io/guide/http#adding-headers

Many servers require extra headers for save operations. For example, a server might require an authorization token or "Content-Type" header to explicitly declare the MIME type of the request body.

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

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

Instances of the ***HttpHeaders*** class are immutable. Use the .set() method to change existing headers.

```
httpOptions.headers =
  httpOptions.headers.set('Authorization', 'my-new-auth-token');
```

Obsevable vs Promise (and others)

https://rxjs-dev.firebaseapp.com/guide/observable
https://angular.io/guide/comparing-observables#observables-compared-to-other-techniques