DESARROLLO DE APLICACIONES CON ANGULAR

Angular

ANGULAR

- Peticiones HTTP
  - Clase @angular/common/http/HttpClient:
    - Permite realizar peticiones HTTP.
    - Disponible como clase inyectable.
    - Dispone de todo tipo de métodos para hacer peticiones (PUT, GET, DELETE, POST,…).

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

ANGULAR

- Peticiones HTTP
  - Clase @angular/common/http/HttpHeaders:
    - Representa las opciones de configuración del encabezado de una petición HTTP.
    - Se indica mediante una propiedad headers de tipo HttpHeaders.
    - Se agregan a las peticiones.
    - Es inmutable.

```
const httpOptions = {
      headers : new HttpHeaders({
        'Content-Type':'application/json;
charset=UTF-8'
      })
}
```

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

ANGULAR

- Peticiones HTTP
  - Objeto @angular/common/http/HttpParams:
    - Representa los parámetros incluidos en una petición o respuesta HTTP.
    - Es inmutable.

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

ANGULAR

- Peticiones HTTP
  - Objeto @angular/common/http/HttpErrorResponse:
    - Representa un error HTTP.

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

ANGULAR

- Peticiones HTTP
  - Importaciones necesarias:
  - En app.module.ts:
    - `import {HttpClientModule} from '@angular/common/http';`
  - En el servicio:
    - `import { HttpClient, HttpErrorResponse, HttpHeaders, HttpParams } from '@angular/common/http';`
    - `import { Observable, throwError } from 'rxjs';`
    - `import { catchError, retry } from 'rxjs/operators';`
    - Se inyecta la referencia:
      - `constructor(private clienteHttp: HttpClient) { }`

ANGULAR

- Peticiones HTTP-Ejemplo de configuración
  - Entorno:
    - const httpHeaders = {
      headers: new HttpHeaders({
        'Content-Type': 'application/json',
      })
    };
    - static URL: string = "https://vehiculos-historicos.herokuapp.com";
    - static path = "/vehiculos";
    - constructor(private http: HttpClient) { }

ANGULAR

- Peticiones HTTP-POST

```
addVehiculo(vehiculo: Vehiculo): void {
    const body = {};
    const parametros = new HttpParams()
      .append('nombre', vehiculo.nombre)
      .append('descripcion', vehiculo.descripcion)
      .append('imagen', vehiculo.imagen);
    this.http.post<any>(HttpClientService.URL + HttpClientService.path, body, { params: parametros }).subscribe(
      {
        next(retorno:any) {
          //retorno es de tipo object
          console.log("Next:" + retorno.code);
          console.log("Next:" + retorno.message);
        },
        error(error:HttpErrorResponse) {
          //error es de tipo HttpErrorResponse
          console.log("Error:" + error.name);
          console.log("Error:" + error.message);
          console.log("Error:" + error.error);
          console.log("Error:" + error.ok);
        },
        complete() {
          console.log("Complete");
        }
      });
  }
```

Parámetros

ANGULAR

- Peticiones HTTP-GET
  - HttpClient.get()→Obtención de información del servidor.
    - Método asíncrono.
    - Envía una petición HTTP y devuelve un Observable.
    - Recibe dos argumentos:
      - URL (endpoint)
      - Objeto con la configuración de la petición.

ANGULAR

- Peticiones HTTP-GET
  - HttpClient.get()→Obtención de información del servidor.
    - En una de sus formas, el método get se puede especificar indicando la clase del objeto que recibe.
    - this.http.get<ClaseResultado>(this.configUrl);

ANGULAR

- Peticiones HTTP-GET
  - HttpClient.get()→En su forma más genérica no es necesario especificar el tipo de resultado, aunque es recomendable si se conoce.

```
this.clienteHttp.get<any>("http://localhost:4200/assets/test.json");
```

vs

```
this.clienteHttp.get<Persona>("http://localhost:4200/assets/test.json");
```

ANGULAR

- Peticiones HTTP-GET
  - Ejemplo 1/3. Estructura de datos (interface).

```
export interface Persona {
    nombre:string,
    edad:number,
    empleado:boolean
}
```

ANGULAR

- Peticiones HTTP-GET
  - Ejemplo 2/3. Llamada HTTP.

```
getPersona():any {
   return this.clienteHttp.get<Persona>("http://localhost:4200/assets/test.json");
 }
```

ANGULAR

- Peticiones HTTP-GET
  - Ejemplo 3/3. Suscripción al Observable.

```
this.getPersona()
    .subscribe((data: Persona) => {
        console.log(data);
    });
```

ANGULAR

- Peticiones HTTP-GET

```
readVehiculos():void{
       this.http.get<any>(HttpClientService.URL + HttpClientService.path).subscribe(
       {
         next(retorno:any) {
           //retorno es de tipo object
           console.log(retorno);
         },
         error(error:HttpErrorResponse) {
           //error es de tipo HttpErrorResponse
           console.log("Error:" + error.name);
           console.log("Error:" + error.message);
           console.log("Error:" + error.error);
           console.log("Error:" + error.ok);
         },
         complete() {
           console.log("Complete");
         }
       });
 }
```

ANGULAR

- Peticiones HTTP-GET con parámetros

```
readVehiculo(id:number):void{
   const endPath = `/findById?id=${id}`;
   this.http.get<any>(HttpClientService.URL + HttpClientService.path + endPath).subscribe(
     {
       next(retorno:any) {
         //retorno es de tipo object
         console.log(retorno);
       },
       error(error:HttpErrorResponse) {
         //error es de tipo HttpErrorResponse
         console.log("Error:" + error.name);
         console.log("Error:" + error.message);
         console.log("Error:" + error.error);
         console.log("Error:" + error.ok);
       },
       complete() {
         console.log("Complete");
       }
     });
```

ANGULAR

- Peticiones HTTP-PUT

```
updateVehiculo(vehiculo: Vehiculo): void {
   const body = {};
   const parametros = new HttpParams()
     .append('id',vehiculo.id)
     .append('nombre', vehiculo.nombre)
     .append('descripcion', vehiculo.descripcion)
     .append('imagen', vehiculo.imagen);
   this.http.put<any>(HttpClientService.URL + HttpClientService.path, body, { params: parametros
}).subscribe(
      {
        next(retorno: any) {
          //retorno es de tipo object
          console.log("Next:" + retorno.code);
          console.log("Next:" + retorno.message);
        },
        error(error: HttpErrorResponse) {
          //error es de tipo HttpErrorResponse
          console.log("Error:" + error.name);
          console.log("Error:" + error.message);
          console.log("Error:" + error.error);
          console.log("Error:" + error.ok);
        },
        complete() {
          console.log("Complete");
        }
      });
 }
```

ANGULAR

- Peticiones HTTP-DELETE

```
 deleteVehiculo(id: number): void {
const endPath = `/?id=${id}`;
    this.http.delete<any>(HttpClientService.URL + HttpClientService.path + endPath).subscribe(
      {
        next(retorno: any) {
          //retorno es de tipo object
          console.log("Next:" + retorno.code);
          console.log("Next:" + retorno.message);
        },
        error(error: HttpErrorResponse) {
          //error es de tipo HttpErrorResponse
          console.log("Error:" + error.name);
          console.log("Error:" + error.message);
          console.log("Error:" + error.error);
          console.log("Error:" + error.ok);
        },
        complete() {
          console.log("Complete");
        }
      });
  }
```

ANGULAR

- Peticiones HTTP-UPDATE con parámetros

```
updateVehiculo(vehiculo: Vehiculo): void {
   const body = {};
   const parametros = new HttpParams()
     .append('id',vehiculo.id)
     .append('nombre', vehiculo.nombre)
     .append('descripcion', vehiculo.descripcion)
     .append('imagen', vehiculo.imagen);
   this.http.put<any>(HttpClientService.URL + HttpClientService.path, body, { params: parametros
}).subscribe(
      {
        next(retorno: any) {
          //retorno es de tipo object
          console.log("Next:" + retorno.code);
          console.log("Next:" + retorno.message);
        },
        error(error: HttpErrorResponse) {
          //error es de tipo HttpErrorResponse
          console.log("Error:" + error.name);
          console.log("Error:" + error.message);
          console.log("Error:" + error.error);
          console.log("Error:" + error.ok);
        },
        complete() {
          console.log("Complete");
        }
      });
 }
```

ANGULAR

- Peticiones HTTP con HEADERS – Ejemplo

```
/**
 * https://rapidapi.com/thekevinconnor-lPsiMJNfyt0/api/free-epic-games/
 */

@Component({
  selector: 'app-http-test-headers',
  templateUrl: './http-test-headers.component.html',
  styleUrls: ['./http-test-headers.component.css']
})
export class HttpTestHeadersComponent implements OnInit {

  URL:string="https://free-epic-games.p.rapidapi.com/free";

  httpHeaders = {
    headers: new HttpHeaders({
      'X-RapidAPI-Key':'8fe6f42975msh4569bbbdea6402bp1c2cd3jsne0025680afe6',
      'X-RapidAPI-Host':'free-epic-games.p.rapidapi.com'
    })
  };
  constructor(private clienteHTTP:HttpClient) { }

  ngOnInit(): void {
    this.clienteHTTP.get<any>(this.URL, this.httpHeaders).subscribe(datos => {
      console.log(datos);
    });
  }
```

ANGULAR

- Peticiones HTTP
  - Las llamadas a get admiten parámetros.
    - clienteHttp.get<Persona>("http://localhost:4200/assets/test.json", {responseType:'json'});
  - Los parámetros se pueden proporcionar como un objeto:

```
const parametros = {
    responseType: 'json' as const,
};
return this.clienteHttp.get<Persona>("http://localhost:4200/assets/test.json", parametros);
```

Obligatorio

ANGULAR

- Peticiones HTTP
  - Gestión de la respuesta completa.
  - Se indica en los parámetros que se espera el objeto response.
  - La llamada se realiza de igual manera, cambia el resultado.

```
getPersona():any {
   const parametros = {
     responseType: 'json' as const,
     observe: 'response' as const,
};
return this.clienteHttp.get<Persona>("http://localhost:4200/assets/test.json", parametros);
```

ANGULAR

- Peticiones HTTP
  - Gestión de la respuesta completa.

Sólo datos

```
Angular is running in development mode. Call enabl
▶ {nombre: 'Fernando', edad: 50, empleado: true}
[webpack-dev-server] Live Reloading enabled.
```

Respuesta completa

```
Angular is running in development mode. Call enableProdMode() to enable production mode.
▼ HttpResponse {headers: HttpHeaders, status: 200, statusText: 'OK', url: 'http://localhost
  ▶ body: {nombre: 'Fernando', edad: 50, empleado: true}
  ▶ headers: HttpHeaders {normalizedNames: Map(0), lazyUpdate: null, lazyInit: ƒ}
    ok: true
    status: 200
    statusText: "OK"
    type: 4
    url: "http://localhost:4200/assets/test.json"
  ▶ [[Prototype]]: HttpResponseBase
[webpack-dev-server] Live Reloading enabled.
```

ANGULAR

- Peticiones HTTP
  - Gestión de la respuesta completa.

```
this.getPersona()
    .subscribe((data: any) => {
        console.log(data);
        console.log(data.ok);
        console.log(data.status);
        console.log(data.statusText);
        console.log(data.type)
    });
```

```
▼HttpResponse {headers: HttpHeaders, status: 200, statusText: 'OK', url: 'ht
  ▶body: {nombre: 'Fernando', edad: 50, empleado: true}
  ▶headers: HttpHeaders {normalizedNames: Map(0), lazyUpdate: null, lazyInit:
   ok: true
   status: 200
   statusText: "OK"
   type: 4
   url: "http://localhost:4200/assets/test.json"
  ▶[[Prototype]]: HttpResponseBase
true
200
OK
4
[webpack-dev-server] Live Reloading enabled.
```

ANGULAR

- Peticiones HTTP
  - La gestión de los retornos de error de HTTP se realiza a través de la correcta gestión de Observable.

```
this.getPersona()
    .subscribe({
      next(data:any){
        console.log("NEXT:" + data.status);
        console.log("NEXT:" + data.body.nombre);
      },
      error(data:any){
        console.log("ERROR:" + data.status);
      },
      complete(){
        console.log(“COMPLETE");//Error 404 no provoca complete()
      }
    })
```

ANGULAR

- Peticiones HTTP-JSON

```
getPersonaJSON():any {
   const parametros = {
     responseType: 'json' as const,
     observe: 'response' as const,
   };
   return this.clienteHttp.get<Persona>("http://localhost:8080/test_get_json",parametros);
 }
```

ANGULAR

- Peticiones HTTP-XML

```
getPersonaXML() {
   const parametros = {
     responseType: 'text' as const,
     observe: 'response' as const,
   };
   return this.clienteHttp.get("http://localhost:8080/test_get_xml", parametros);
 }
```

ANGULAR

Peticiones HTTP-XML

Se necesita convertir el retorno en formato XML-string a objeto JS (JSON).
Librería propuesta: xml-js

https://www.npmjs.com/package/xml-js

Ejemplo de llamada y conversión:

```
let retorno: string = "";
    this.clienteHTTP.get(URL_DEL_WS, parametros).subscribe(
      retorno => {
          var conversor = require('xml-js');
          var retornoJS = conversor.xml2js(retorno.body, {compact: true, spaces: 4});
      }
    )
```

Objeto JS

ANGULAR

- Peticiones HTTP-MANEJO DE ERRORES

```
const self=this;
this.getPersonaJSON("http://localhost:8080/test_get_json")
   .subscribe({
     next(data:any){
       console.log(data);
     },
     error(data:any){
       self.handleError(data);
     },
     complete(){
       console.log("COMPLETE");
     }
   })
```

```
handleError(error: HttpErrorResponse) {
    if (error.status === 0) {
      //Error cliente
    } else {
      //Error servidor
      //error.status -> Código de retorno HTTP.
    }
  }
```

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

ANGULAR

- Peticiones HTTP-Reintentos

```
getPersonaJSON(url:string):any {
  const parametros = {
    responseType: 'json' as const,
    observe: 'response' as const,
  };
  return this.clienteHttp.get<Persona>(url,parametros).pipe(
     retry(2)
  );
}
```

ANGULAR

- HTTP-Interceptor
  - Permite interceptar y modificar las peticiones http para automatizar llamadas.
  - Módulo @angular/common/http.
  - Son inmutables.
  - Si hay múltiples, el orden de intercepción de entrada es el contrario del de salida.
  - Un interceptor implementa la interface HttpInterceptor, que dispone del método intercept.
  - Deben estar registrados para que se ejecuten.

ANGULAR

- HTTP-Interceptor
  - Importaciones (app.module.ts):

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

  - Creación:

```
ng generate interceptor nombre_interceptor
```

  - Registro (app.module.ts):

```
import { SimpleInterceptor } from './interceptors/simple.interceptor';
import { SecondInterceptor } from './interceptors/second.interceptor';

providers: [
    { provide: HTTP_INTERCEPTORS, useClass: SimpleInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
  ],
```

Este orden determina la secuencia de ejecución

ANGULAR

- HTTP-Interceptor

```
@Injectable()
export class SimpleInterceptor implements HttpInterceptor {

  constructor() {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler): Observable<HttpEvent<unknown>> {
    const headers = request.clone({
      url: request.url.replace('localhost','127.0.0.1'),
      headers: request.headers.set('clave', 'patata23'),
      params: request.params.set('nombre','Fernando')
    });
    return next.handle(headers);
  }
}
```