6-http

Comunicaciones http en Angular

---

1. El servicio HttpClient

2. Observables

3. Interceptores

---

1. El servicio HttpClient

Importación y declaración de servicios

Obtención de datos

Envío de datos

Actualización de datos

El módulo de comunicaciones

```
ng g m rates --routing true
ng g c rates/rates
```

`app-routing.module.ts`

```
{
  path: 'rates',
  loadChildren: './rates/rates.module#RatesModule'
},
```

`rates-routing.module.ts`

```
{
  path: '',
  component: RatesComponent
}
```

`header.component.html`

```html
<a routerLink="rates" class="button">
  <span> Rates</span>
</a>
```

1.1 Importación y declaración de servicios

Importación

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


@NgModule({
  declarations: [RatesComponent],
* imports: [HttpClientModule]
})
export class RatesModule { }
```

Dependencia

```
*import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-rates',
  templateUrl: './rates.component.html',
  styles: []
})
export class RatesComponent implements OnInit {
* constructor(private httpClient: HttpClient) {}

  ngOnInit() {}
}
```

1.2 Obtención de datos

```
export class RatesComponent implements OnInit {
  private urlapi = 'https://api.exchangeratesapi.io/latest';
  public currentEuroRates: any = null;

  constructor(private httpClient: HttpClient) {}

  ngOnInit() {
    this.getCurrentEuroRates();
  }

  private getCurrentEuroRates() {
    const currencies = 'USD,GBP,CHF,JPY';
    const url = `${this.urlapi}?symbols=${currencies}`;
    this.httpClient
*     .get(url)
      .subscribe(apiResult => (this.currentEuroRates = apiResult));
  }
}
```

Presentación en vista

```
<h2> Currency Rates. </h2>
<h3> From Euro to the world </h3>
<pre>{{ currentEuroRates | json }}</pre>
```

1.3 Envío de datos

```
export class RatesComponent implements OnInit {
  private myRatesApi = 'https://api-base.herokuapp.com/api/pub/rates';

  public postRates() {
    const rates = this.transformData();
    rates.forEach(rate =>
      this.httpClient
*       .post(this.myRatesApi, rate)
        .subscribe()
    );
  }

  private transformData() {
    const current = this.currentEuroRates.rates;
    return Object.keys(current).map(key => ({
      date: this.currentEuroRates.date,
      currency: key,
      euros: current[key]
    }));
    }
  }
```

Presentación en vista

```
<input value="Save Rates" type="button" (click)="postRates()" />
```

1.4 Actualización de datos

Refresco

```
export class RatesComponent implements OnInit {
 private myRatesApi = 'https://api-base.herokuapp.com/api/pub/rates';
 public myRates: any[] = null;


 public getMyRates() {
    this.httpClient
*     .get<any[]>(this.myRatesApi)
      .subscribe(apiResult => (this.myRates = apiResult));
  }
}
```

```
<input value="Refresh" type="button" (click)="getMyRates()" />
<pre>{{ myRates | json }}</pre>
```

Borrado

```
  public deleteMyRates() {
    this.httpClient
*     .delete(this.myRatesApi)
      .subscribe();
  }
```

```
<input value="Delete Rates" type="button" (click)="deleteMyRates()" />
```

---

2. Observables

Async

pipe

operators

```
ng g c rates/obserates
{
  path: 'observables',
  component: ObseratesComponent
}
<p><a [routerLink]="['observables']">Observables</a></p>
```

2.1 Async

Tuberías de Angular |

```
<h2> Currency Observable Rates. </h2>
<h3> From Euro to the $ world </h3>
<pre>{{ currentEuroRates$ | async | json }}</pre>
```

> Recibe un observable, se suscribe, y devuelve el dato cuando llegue.

En el controlador se exponen Observables

```
  private ratesApi = 'https://api.exchangeratesapi.io/latest';
* public currentEuroRates$: Observable<any> = null;

  constructor(private httpClient: HttpClient) {}

  ngOnInit() {
    this.getCurrentEuroRates();
  }

  private getCurrentEuroRates() {
    const currencies = 'USD,GBP,CHF,JPY';
    const url = `${this.ratesApi}?symbols=${currencies}`;
*   this.currentEuroRates$ = this.httpClient.get(url);
  }
```

> No es necesaria la suscripción en código

2.2 Pipe

Tuberías en RxJS .pipe()

```
public myRates$: Observable<any[]> = null;
private getCurrentEuroRates() {
  const url = `${this.ratesApi}?symbols=USD,GBP,CHF,JPY`;
  this.currentEuroRates$ = this.httpClient.get(url);
* this.myRates$ = this.currentEuroRates$.pipe(map(this.transformData));
}
private transformData(currentRates) {
  const current = currentRates.rates;
  return Object.keys(current).map(key => ({
    date: currentRates.date,
    currency: key,
    euros: current[key]
  }));
}
```

2.3 Operators

```
<pre>{{ myRates$ | async | json }}</pre>
```

El consumo sigue igual... pero...

--

```
private getCurrentEuroRates() {
const url = `${this.ratesApi}?symbols=USD,GBP,CHF,JPY`;
  this.currentEuroRates$ = this.httpClient.get(url)
*     .pipe(share());
  this.myRates$ = this.currentEuroRates$
      .pipe(
*       tap(d=>console.log(d)),
        map(this.transformData),
        tap(t=>console.log(t))
      );
}
```

---

3. Interceptores

La interfaz HttpInterceptor

Inversión del control vía token

Un auditor de llamadas

```
ng g s rates/AuditInterceptor
```

Hay que crear un servicio inyectable y hacerle cumplir una Interfaz

---

3.1 La interfaz HttpInterceptor

```
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest }
  from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class AuditInterceptorService implements HttpInterceptor {
  public intercept(
    req: HttpRequest<any>,
    next: HttpHandler )
    : Observable<HttpEvent<any>> {
    // throw new Error( 'Method not implemented.' );
    return next.handle(req);
  }

  constructor() { }
}
```

3.2 Inversión del control vía token

> 1. Quitamos el `providedIn: 'root'`

> 2. Tomamos el control de la inyección

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

```
```

El `HttpClient` en su constructor reclama `HTTP_INTERCEPTORS`, un array de múltiples dependencias

Le damos nuestro interceptor para que lo agregue a su array

3.3 Un auditor de llamadas

```
export class AuditInterceptorService implements HttpInterceptor {
  constructor() {}

  public intercept(req: HttpRequest<any>, next: HttpHandler){
    const started = Date.now();
    return next.handle(req).pipe(
      filter((event: HttpEvent<any>) => event instanceof HttpResponse),
      tap((resp: HttpResponse<any>) => this.auditEvent(resp, started))
    );
  }

  private auditEvent(resp: HttpResponse<any>, started: number) {
    const elapsedMs = Date.now() - started;
    const eventMessage = resp.statusText + ' on ' + resp.url;
    const message = eventMessage + ' in ' + elapsedMs + 'ms';
    console.log(message);
  }
}
```

> Next:

Vigilancia y seguridad en Angular

Uso de observables para monitorizar datos

Uso de interceptores para gestionar errores

Un notificador de problemas

> **Blog de apoyo:** Comunicaciones Http en Angular

> > By Alberto Basalo