



## ► 3. Tratamiento asíncrono de datos con RxJS


### Objetivos

- Dominar los constructores y operadores de RxJS
- Usar interceptores para tratar llamadas Ajax de forma centralizada
- Usar almacenes de información reactivos

### Referencia

#### RxJS. Transformation Operators in Examples (part 1)


Examples of buffer, bufferCount, bufferTime, bufferToggle, bufferWhen, concatMap, concatMapTo, exhaust, exhaustMap and expand RxJS Transformation operators (part 1). This article is a continuation of RxJS Basics in Examples. RxJS. Transformation Operators in

 <https://medium.com/weekly-webtips/rxjs-transformation-operators-in-examples-part-1-9c9fb7b4e705>



#### RxJS. Transformation Operators in Examples (part 2)


All you need to know about map, mapTo, mergeMap, mergeMapTo, scan, mergeScan, groupBy, pairwise, partition, pluck, switchMap, switchMapTo, window, windowCount, windowTime, windowToggle, and windowWhen RxJS Transformation operators in examples

 <https://medium.com/weekly-webtips/rxjs-transformation-operators-in-examples-part-2-9d0f09bdbc6d>



#### Introduction

is one of the hottest libraries in web development today. Offering a powerful, functional approach for dealing with events and with integration points into a growing number of frameworks, libraries, and utilities, the case for learning Rx has never been more appealing.


 <https://www.learnrxjs.io/>



Learn RxJS  
Powered by  GitBook

#### Top 10 ways to use Interceptors in Angular - Angular inDepth


Find out which superpowers you can start using today > Just like Batman develops his gadgets we can use interceptors to gain superpowers. There are many ways to use an interceptor, and I'm sure most of us have only scratched the surface. In this article, I will

 <https://indepth.dev/posts/1051/top-10-ways-to-use-interceptors-in-angular>



#### 7+ Ways to Leverage HttpInterceptors in Angular


According to angular.io: HTTP Interception is a major feature of @angular/common/http. With interception, you declare interceptors that inspect and transform HTTP requests from your application to the server. The same interceptors may also inspect and transform the server's

 <https://blog.bitsrc.io/7-ways-to-leverage-httpinterceptors-in-angular-59436611844d>



alexzuza/angular-refresh-token

Three ways to refresh token with Angular Http Interceptor - alexzuza/angular-refresh-token

 <https://github.com/alexzuza/angular-refresh-token>



### Handling HTTP Request using RxJs in Angular

Handling HTTP request in an Angular application is a crucial part. In this article we will go through below methods from RxJs to handle HTTP requests. map → applies the function supplied to each value of input observable and emits the resulting value as observable.

<https://abhishhek-ankush.medium.com/handling-http-request-using-rxjs-in-angular-25060a70c6d9>



### RxJS essentials that every Angular beginner needs to learn

If you don't understand at least the basics, the 101, the bread and butter of RxJS, you're leaving a lot of helpers behind. You're wasting your time and energy! Let me explain: Angular was grounded in RxJS paradigms and "lifestyle".

<https://gustavo-veloso.medium.com/rxjs-essentials-that-every-angular-beginner-needs-to-learn-23171f8fb525>



## 0 - Store\$

Implementación minimalista de un **Store Observable con RxJs**

### Una clase para ordenarlo todo

```
export interface Action {
  type: string;
  payload: any;
}

export class Store<T> {
  private _state$: BehaviorSubject<T>;
  private _actions$: BehaviorSubject<Action>;

  constructor(initialState: T) {
    this._state$ = new BehaviorSubject(this.getClone(initialState));
    const initialAction: Action = {
      type: 'INITIAL',
      payload: initialState,
    };
    this._actions$ = new BehaviorSubject(initialAction);
  }

  setState(mutation: Partial<T>) {
    const mutatedState = { ...this.getSnapshot(), ...mutation };
    this._state$.next(this.getClone(mutatedState));
  }

  dispatch(action: Action) {
    this.setState(action.payload);
    this._actions$.next(action);
  }
}
```

```

    reduce(action: Action, reducer: (currentStat: T, payload: any) => T) {
      const mutatedState = reducer(this.getSnapshot(), action.payload);
      this._state$.next(this.getClone(mutatedState));
      this._actions$.next(action);
    }

    getSnapshot() {
      return this.getClone(this._state$.value);
    }

    getState$() {
      return this._state$
        .asObservable()
        .pipe(map((state) => this.getClone(state)));
    }

    getActions$() {
      return this._actions$.asObservable();
    }

    private getClone(source: T): T {
      return { ...source };
    }
  }
}

```



Tienes un ejemplo más realista en :

<https://github.com/angularbuilders/angularbuilders/blob/main/libs/shared/global/src/lib/store.ts>

## 1 - Store services

Por ejemplo un servicio para monitorizar la aplicación a distintos niveles

```

export interface TrackEntry {
  category: TrackerCategories;
  event: string;
  label?: string;
  value?: number;
}

export type TrackerCategories = 'ERROR' | 'BUSINESS' | 'SYSTEM';

@Injectable({
  providedIn: 'root',
})
export class TrackerStoreService {
  private store = new Store<TrackEntry>({
    category: 'SYSTEM',
    event: 'TRACKER_INIT',
    label: 'Angular.Builders',
  });

  constructor() {}

  trackEntry(entry: TrackEntry) {
    const action: Action = {
      type: 'TRACK_' + entry.category,
      payload: entry,
    };
  }
}

```

```

    this.store.dispatch(action);
  }

  selectByEvent$(event: string) {
    const byEvent = (state: TrackerEntry) => state.event === event;
    return this.store.getState$.pipe(filter(byEvent));
  }

  selectAnyErrors$() {
    const byErrorCategory = (state: TrackerEntry) => state.category === 'ERROR';
    return this.store.getState$.pipe(filter(byErrorCategory));
  }

  selectActions$() {
    return this.store.getActions$();
  }
}

```



Tienes un ejemplo más realista en:

<https://github.com/angularbuilders/angularbuilders/blob/main/libs/shared/global/src/lib/tracker.store.ts>



## Productores

Ejemplo más completo

```

@Injectable()
export class TrackerInterceptor implements HttpInterceptor {
  constructor(private tracker: TrackerStoreService) {}

  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    const startTimestamp = this.trackCallStart(request);
    return next.handle(request).pipe(
      tap((res) => {
        if (this.isCorsPreFlight(res)) return;
        this.trackCallEnd(startTimestamp, request);
      }),
      catchError((error: HttpResponse) => {
        this.trackCallError(error);
        return throwError(error);
      })
    );
  }

  private trackCallStart(request: HttpRequest<unknown>): number {
    const startTimestamp = new Date().getTime();
    this.tracker.trackEntry({
      category: 'SYSTEM',
      event: 'CALL_START',
      label: request.method + ' @ ' + request.url,
      value: startTimestamp,
    });
    return startTimestamp;
  }

  private trackCallEnd(startTimestamp: number, request: HttpRequest<unknown>) {
    const endTimestamp: number = new Date().getTime();
    const responseTime = endTimestamp - startTimestamp;
    this.tracker.trackEntry({

```

```

        category: 'SYSTEM',
        event: 'CALL_END',
        label: request.method + ' @ ' + request.url,
        value: responseTime,
    });
}

private trackCallError(error: HttpResponse) {
    const errorEntry: TrackerEntry = {
        category: 'ERROR',
        event: 'CODE_FAULT',
    };
    if (error.error instanceof ErrorEvent) {
        const codeError = error.error;
        errorEntry.label = codeError.message + ' @ ' + codeError.filename;
        errorEntry.value = codeError.lineno;
    } else {
        errorEntry.event = 'CALLER_FAULT';
        if (error.status >= 500) {
            errorEntry.event = 'SERVER_FAULT';
        } else if ([401, 403].includes(error.status)) {
            errorEntry.event = 'AUTH_FAULT';
        }
        if (error.url) errorEntry.label = error.url;
        errorEntry.value = error.status;
    }
    this.tracker.trackEntry(errorEntry);
}

private isCorsPreFlight(res: HttpEvent<any>) {
    return res.type === 0;
}
}

@Injectable()
export class ErrorHandlerService implements ErrorHandler {
    constructor(private tracker: TrackerStoreService) {}
    handleError(error: Error): void {
        if (!!error.name && error.name === 'HttpResponse') return;
        this.tracker.trackEntry({
            category: 'ERROR',
            event: 'CODE_FAULT',
            label: error.message + '@' + error.stack || 'unknown',
        });
    }
}

// En Core Module...

providers: [
    {
        provide: HTTP_INTERCEPTORS,
        useClass: TrackerInterceptor,
        multi: true,
    },
    {
        provide: ErrorHandler,
        useClass: ErrorHandlerService,
    },
],

```

## Consumidores



Atención: Uso avanzado de **RxJS**. Constructor merge en `NavbarWidget` y auto close en `NotificationComponent`.

```
// NavbarWidget
<nav class="navbar has-shadow" role="navigation" aria-label="main navigation">
  <div class="navbar-brand">
    <a class="navbar-item" [routerLink]='["/"]">
      <strong class="ml-4 is-size-2">Angular Builders</strong> </a>
    </div>
    <div id="navbarItems" class="navbar-menu is-active">
      <ng-content select=".navbar-start">
      </ng-content>
      <div class="navbar-end mt-4 mr-4">
        <aside *ngIf="notification$ | async as notification"
          class="notification {{notification.class}}">
          {{ notification.message }}
        </aside>
        <a [routerLink]='["/resource-new"]' class="button ">Add new Resource</a>
      </div>
    </div>
  </nav>

export class NavbarWidget {
  notification$: Observable<Notification>;
  loading$: Observable<boolean>;

  constructor(tracker: TrackerStoreService) {
    // ToDo: create specific notifications by error event kind
    const error$ = tracker.selectAnyErrors$().pipe(
      map(() => ({
        class: 'is-danger',
        message:
          'There was an error!. Review your data and retry. If persists we will fix it ASAP!',
      })))
    );
    // ToDo: use another store for user notifications
    const success$ = tracker.selectByEvent$('NEW_RESOURCE').pipe(
      map((trackEntry) => ({
        class: 'is-success',
        message: trackEntry.label || 'Success',
      })))
    );
    this.notification$ = merge(error$, success$);

    const callStart$ = tracker
      .selectByEvent$('CALL_START')
      .pipe(map(() => true));
    const callEnd$ = tracker.selectByEvent$('CALL_END').pipe(map(() => false));
    this.loading$ = merge(callStart$, callEnd$);
  }
}

// NotificationComponent
<aside *ngIf="show$ | async" class="notification {{notification.class}} is-light mt-4">
  <button class="delete" (click)="onClose()"></button> M: {{ notification.message }}
</aside>

export class NotificationComponent implements OnInit {
  @Input() notification: Notification = { class: '', message: '' };

  show$ = new BehaviorSubject<boolean>(false);

  constructor() {}
  ngOnInit(): void {
    throw new Error('Method not implemented.');
```

```

}

ngOnChanges(changes: SimpleChanges): void {
  this.show$.next(true);
  timer(3000).subscribe(() => this.onClose());
}

onClose() {
  this.show$.next(false);
}
}

```

```

export class CoreModule {
  constructor(router: Router, tracker: TrackerStore) {
    router.events
      .pipe(filter((event) => event instanceof NavigationEnd))
      .subscribe({
        next: (routerEvent) =>
          tracker.trackEntry({
            category: 'BUSINESS',
            event: 'NAV',
            label: (routerEvent as NavigationEnd).urlAfterRedirects,
          }),
      });
    if (environment.production === false) {
      // ToDo: Use Redux DevTools
      tracker.selectActions$.subscribe((action) => console.table(action));
    }
  }
}

```

## 2 - Interceptores

Otros ejemplos de interceptores.

### Adaptador

```

# 📄 Adapter Interceptor
ng g interceptor adapter --project shared-data

@Injectable()
export class AdapterInterceptor implements HttpInterceptor {
  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    return next.handle(request).pipe(
      filter((event) => event instanceof HttpResponse),
      map((event) => event as HttpResponse<any>),
      map((response) => this.adaptResponse(response))
    );
  }

  adaptResponse(response: HttpResponse<any>) {
    const body = response.body;
    const adaptedBody = body['data'] || [];
    const adaptedResponse = response.clone({ body: adaptedBody });
    return adaptedResponse;
  }
}

```

## Retry

```
# ☞ Retry Interceptor
ng g interceptor retry --project shared-data
const RETRY_MAX = 3;
const DELAYED_RETRY_MS = 3000;

@Injectable()
export class RetryInterceptor implements HttpInterceptor {
  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    return next
      .handle(request)
      .pipe(retryWhen((error$) => this.tryRetry(error$)));
  }

  tryRetry(error$: Observable<HttpErrorResponse>) {
    return error$.pipe(
      concatMap((error, count) => {
        if (this.canRetry(error, count)) {
          return of(error);
        } else {
          return throwError(error);
        }
      }),
      delay(DELAYED_RETRY_MS)
    );
  }

  private canRetry(error: HttpErrorResponse, count: number) {
    return (error.status == 0 || error.status >= 500) && count < RETRY_MAX;
  }
}
```

## Seguridad

```
# ☞ Auth Interceptor
ng g interceptor auth --project shared-data

# 📖 depends on AuthService Store (see below)
@Injectable()
export class AuthInterceptor implements HttpInterceptor {
  constructor(private service: AuthService) {}
  intercept(
    request: HttpRequest<unknown>,
    next: HttpHandler
  ): Observable<HttpEvent<unknown>> {
    const authorization = 'Bearer ' + 'session token from AuthService';
    request = request.clone({
      setHeaders: {
        Authorization: authorization,
      },
    });
    return next.handle(request);
  }
}
```

## Registro de proveedores



```
@NgModule({
  imports: [CommonModule, HttpClientModule],
  exports: [HttpClientModule],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: RetryInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: AdapterInterceptor, multi: true },
  ],
})
export class CoreModule {}
```



**Alberto Basalo**