UTILISES-TU BIEN SWITCHMAP EN ANGULAR ?

Découvre comment optimiser tes applications en maîtrisant cet opérateur Rxjs.

Elodie TURLIER

Angular, IA, agilité & storytelling : construire des apps prêtes à survivre à l’apocalypse digitale | Du besoin à la livraison

01

SWITCHMAP, C'EST QUOI ?

SwitchMap est un opérateur de transformation qui fait partie de la famille des "**opérateurs d'aplatissement**" (flattening operators) de RxJS. Ces opérateurs sont essentiels pour gérer les flux d'observables imbriqués, un scénario courant dans les applications Angular.

switchMap* = combinaison de *map* + *switchAll

✓ Transforme une valeur en Observable (comme map)
✓ N'abonne que le dernier Observable actif (comme switchAll)
✓ Annule automatiquement les anciens abonnements

Idéal pour interfaces réactives, comme recherche ou navigation dynamique

Elodie TURLIER
Angular, IA, agilité & storytelling

02

COMMENT ÇA FONCTIONNE ?

1. Observable source émet une valeur
2. switchMap déclenche une fonction → Observable interne
3. Se désabonne des précédents
4. S’abonne au plus récent uniquement

Il ne s'intéresse qu'à l'observable le plus récent, abandonnant les précédents.

Elodie TURLIER
Angular, IA, agilité & storytelling

03

ANALOGIE SIMPLE

- Tu envoies des messages texte à plusieurs amis en même temps.
- Tu changes d’avis, tu parles à un autre ami.
- Le message au premier n’est jamais envoyé.

Utiliser ***switchMap*** revient à **abandonner complètement la conversation avec un ami précédent dès que vous commencez à écrire à un nouvel ami, sans jamais lui envoyer de message**.

C’est ça switchMap :

- Toujours le dernier Observable
- Annule les conversations précédentes

En revanche, ***mergeMap*** serait comme envoyer des messages à plusieurs amis en parallèle sans arrêter aucune conversation

Elodie TURLIER
Angular, IA, agilité & storytelling

04

SYNTAXE

```
switchMap(
  project: (outerValue, outerIndex) => Observable,
  resultSelector?: (outerValue, innerValue, outerIndex, innerIndex) => any
): Observable
```

- **project** (obligatoire) : une fonction qui prend chaque valeur émise par l'observable source et retourne un nouvel observable (appelé "observable interne").
- **resultSelector** (optionnel, désormais déprécié) : une fonction qui permet de combiner la valeur de l'observable source, la valeur de l'observable interne, et leurs indices respectifs, pour produire la valeur finale émise. Cette fonction est rarement utilisée dans les versions récentes de RxJS.

Elodie TURLIER
Angular, IA, agilité & storytelling

05

COMPARAISON RAPIDE AVEC LES AUTRES OPÉRATEURS

Opérateur	Description	Quand l'utiliser ?
`switchMap`	N'écoute que le dernier	Recherche, navigation, input utilisateur
`mergeMap`	Écoute tous les Observables en parallèle	Upload multiple, traitement batch
`concatMap`	Enchaîne les Observables un par un	Sauvegardes, envois séquentiels
`exhaustMap`	Ignore les nouvelles valeurs pendant exécution	Auth, validation unique

Elodie TURLIER
Angular, IA, agilité & storytelling

CAS CONCRET: 06
RECHERCHE HTTP AVEC ANNULATION

Lorsqu'un utilisateur effectue des actions qui déclenchent des requêtes successives (comme taper dans un champ de recherche), switchMap permet de n'envoyer que la requête correspondant à la dernière action et d'annuler les précédentes.

```
const searchTerms$ = fromEvent(this.searchInput.nativeElement, 'input')
  .pipe(
    map(event => (event.target as HTMLInputElement).value),
    debounceTime(300),
    distinctUntilChanged(),
    switchMap(term => this.searchService.search(term))
  );

searchTerms$.subscribe(results => this.searchResults = results);
```

- La requête précédente est annulée et seule la dernière est traitée.
- Moins de charge serveur, plus de cohérence UI

Elodie TURLIER
Angular, IA, agilité & storytelling

07

CAS CONCRET: ROUTING ET NAVIGATION

switchMap est également très utile pour gérer les paramètres de route dans Angular. Lorsqu'un utilisateur navigue d'une route à une autre, on veut souvent récupérer des données basées sur les paramètres de la nouvelle route

```
ngOnInit() {
  this.route.paramMap.pipe(
    switchMap(params => {
      const employeeId = params.get('id');
      return this.employeeService.getEmployee(employeeId);
    })
  ).subscribe(employee => this.employee = employee);
}
```

- Gère les changements de route rapides
- Évite l'affichage de données périmées

Elodie TURLIER
Angular, IA, agilité & storytelling

CAS CONCRET: UTILISATION AVEC NGRX EFFECTS

08

Dans l'écosystème ***NgRx***, ***switchMap*** peut être utilisé dans les effets pour transformer des actions en d'autres actions basées sur les résultats d'opérations asynchrones.

```
@Effect()
loadCustomers$ = this.actions$.pipe(
  ofType(CustomerActionTypes.LOAD_CUSTOMERS),
  switchMap(() =>
    this.customerService.getCustomers().pipe(
      map(customers => new LoadCustomersSuccess(customers)),
      catchError(error => of(new LoadCustomersFailure(error)))
    )
  )
);
```

Dans cet exemple, lorsqu'une action LOAD_CUSTOMERS est dispatched, l'effet utilise switchMap pour appeler le service getCustomers() et transformer le résultat en une nouvelle action (soit success soit failure)

Elodie TURLIER
Angular, IA, agilité & storytelling

09

QUAND UTILISER SWITCHMAP

- **Recherches en temps réel:** Idéal pour les barres de recherche où seuls les résultats de la dernière requête sont pertinents.
- **Navigation entre routes:** Lorsque vous naviguez rapidement entre les pages et ne voulez traiter que les données de la dernière page visitée.
- **Annulation de requêtes en cours:** Parfait pour les scénarios où les requêtes précédentes deviennent obsolètes dès qu'une nouvelle est lancée.
- **Prévention des fuites de mémoire:** Dans les situations impliquant des observables à longue durée de vie où l'on pourrait oublier de se désabonner.

Elodie TURLIER
Angular, IA, agilité & storytelling

10

QUAND ÉVITER SWITCHMAP

- **Opérations d'écriture:** Évitez switchMap pour les opérations qui doivent absolument être complétées, comme les écritures dans une base de données. Dans ce cas, concatMap est plus approprié.
- **Traitements parallèles:** Si vous avez besoin que plusieurs opérations s'exécutent en parallèle, mergeMap est un meilleur choix.
- **Respect de l'ordre des émissions:** Si l'ordre d'exécution est important, concatMap garantit que les observables sont traités dans l'ordre d'émission.

Elodie TURLIER
Angular, IA, agilité & storytelling

COMBINAISONS UTILES

- ***debounceTime*** + ***distinctUntilChanged*** + ***switchMap***

Exemple : Validation d'un nom d'utilisateur pendant la saisie

- ***map*** + ***switchMap***

Exemple : Combiner des observables pour un formulaire de connexion (Transformer + requête dépendante)

- ***takeUntil*** + ***switchMap***

Exemple : Gestion des intervalles avec pause et reprise

Elodie TURLIER
Angular, IA, agilité & storytelling

CHECKLIST MEMO

- ***switchMap*** : top pour interactivité utilisateur

- Préfère ***concatMap*** si ordre important ou complet

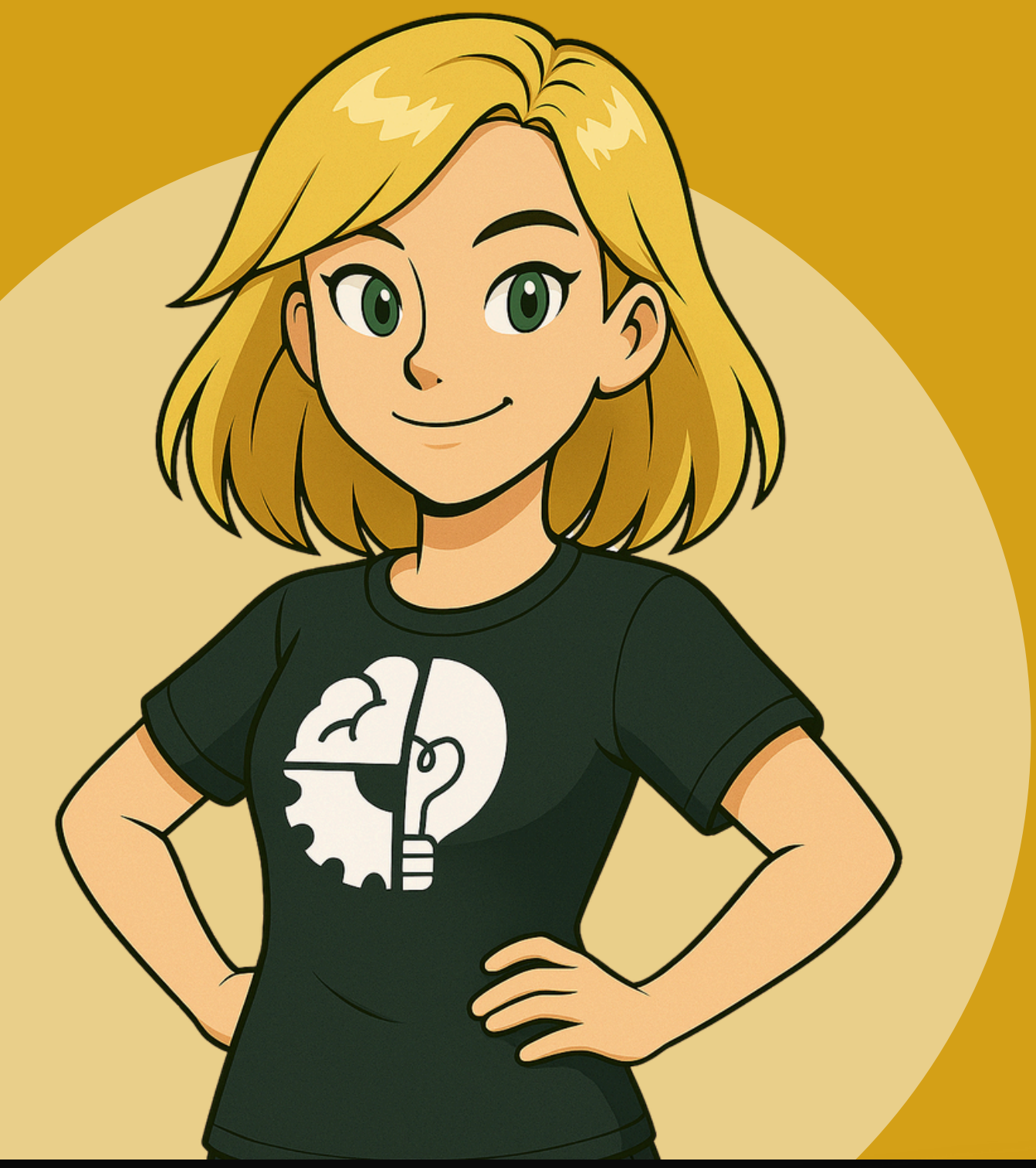
- Combine avec ***catchError*** pour robustesse

- **Attention** aux cas critiques où l’annulation est indésirable

Documentation officelle: Rxjs | switchMap
https://rxjs.dev/api/operators/switchMap

Elodie TURLIER
Angular, IA, agilité & storytelling

ABONNE TOI POUR PLUS DE CONTENU



Elodie TURLIER

Angular, IA, agilité & storytelling : construire des apps prêtes à survivre à l’apocalypse digitale | Du besoin à la livraison

Enregistre ou partage si ça t’a été utile