MERGEMAP : L'OPÉRATEUR QUI BOOSTE TON ANGULAR

Ton appli ralentit à cause de multiples requêtes HTTP ? MergeMap va changer la donne !

Elodie TURLIER

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

01

COMPRENDRE MERGEMAP

L'opérateur ***mergeMap*** (également connu sous le nom de ***flatMap***) transforme chaque émission d'un Observable en un autre Observable interne. Il fusionne ensuite tous ces flux en un seul.

En termes simples, mergeMap permet de créer et de gérer plusieurs flux d'Observables concurrents.

✓ Exécution parallèle de plusieurs flux
✓ Optimisation maximale des performances

Idéal pour :

- Requêtes HTTP simultanées
- Traitement parallèle de données

Elodie TURLIER
Angular, IA, agilité & storytelling

02

COMMENT ÇA FONCTIONNE

Le comportement peut être visualisé comme un système où plusieurs processus sont lancés en parallèle, chacun travaillant indépendamment des autres, et dont les résultats sont fusionnés dans un flux unique à la sortie.

03

LA SYNTAXE À CONNAÎTRE

```
mergeMap(
  project: function: Observable,
  resultSelector: function: any,
  concurrent: number
): Observable
```

- **project** : Une fonction qui prend la valeur émise par l'Observable source et retourne une Observable interne
- **resultSelector** (optionnel) : Une fonction pour transformer le résultat
- **concurrent** (optionnel) : Un nombre qui limite la quantité d'Observables internes actives simultanément

Elodie TURLIER
Angular, IA, agilité & storytelling

04

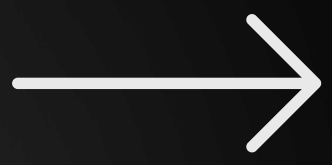
EXEMPLE PRATIQUE

```ts
import { of } from 'rxjs';
import { mergeMap } from 'rxjs/operators';

const srcObservable = of(1, 2, 3, 4);

srcObservable.pipe(
  mergeMap(val => of(`Résultat ${val}`))
).subscribe(val => console.log(val));
```

Sortie :

```
Résultat 1
Résultat 2
Résultat 3
Résultat 4
```

Elodie TURLIER
Angular, IA, agilité & storytelling



05

MERGEMAP VS SWITCHMAP VS CONCATMAP VS EXHAUSTMAP

Opérateur	Comportement	Cas d'usage
`switchMap`	Annule la requête précédente	Recherche dynamique
`concatMap`	Exécute une requête après l'autre (séquentiel)	Transactions ordonnées
`exhaustMap`	Ignore les nouvelles requêtes pendant une en cours	Actions répétées

Elodie TURLIER
Angular, IA, agilité & storytelling

06

CAS CONCRET: TRAITEMENT DE REQUÊTES HTTP PARALLÈLES

Par exemple, lorsque vous devez récupérer des données pour chaque élément d'une liste :

```
this.userIds$.pipe(
  mergeMap(id => this.http.get(`/api/users/${id}`))
).subscribe(userData => {
  // Traiter les données utilisateur à mesure qu'elles arrivent
});
```

Cette approche est particulièrement efficace lorsque l'ordre de réception des réponses n'est pas important et que vous souhaitez maximiser les performances en exécutant toutes les requêtes simultanément

Elodie TURLIER
Angular, IA, agilité & storytelling

07

CAS CONCRET: TRANSFORMATION DE DONNÉES API

MergeMap est également utile pour transformer les données reçues d'une API en effectuant des requêtes secondaires :

```
this.http.get('/api/users').pipe(
  mergeMap((users: User[]) => from(users)),
  mergeMap(user => this.http.get(`/api/addresses/${user.id}`))
).subscribe(address => {
  // Traiter chaque adresse individuellement
});
```

Dans cet exemple, la première requête obtient une liste d'utilisateurs, qui est ensuite transformée en une série de requêtes d'adresse individuelles, toutes traitées en parallèle

Elodie TURLIER
Angular, IA, agilité & storytelling

08

AVANTAGES

- **Exécution concurrente** : MergeMap permet l'exécution simultanée de plusieurs Observables, maximisant ainsi les performances pour les opérations parallèles.

- **Préservation des opérations** : Contrairement à switchMap, mergeMap ne cancelle pas les opérations en cours, ce qui est crucial pour les opérations d'écriture ou les processus qui doivent être complétés.

- **Flexibilité** : Le paramètre concurrent permet de contrôler précisément le degré de parallélisme, offrant un équilibre entre performances et utilisation des ressources.

INCONVÉNIENTS

- **Risques de fuites mémoire** : En maintenant plusieurs souscriptions actives simultanément, mergeMap peut potentiellement créer des fuites de mémoire si les Observables internes ont une longue durée de vie.

- **Ordre non garanti** : Les valeurs peuvent être émises dans un ordre différent de celui des valeurs sources, ce qui peut compliquer le raisonnement sur le code et causer des bugs subtils.

- **Complexité accrue** : La gestion de multiples flux concurrents peut rendre le code plus difficile à comprendre et à débogue.

Elodie TURLIER
Angular, IA, agilité & storytelling

10

OPTIMISE TON UTILISATION

- Contrôle précisément la **concurrence** pour ne pas surcharger tes ressources :

```
from(largeArrayOfIds).pipe(
  mergeMap(id => this.http.get(`/api/resource/${id}`), 5)
  // Limite à 5 requêtes simultanées
).subscribe();
```

- Protège ton flux avec une **gestion robuste des erreurs** :

```
items$.pipe(
  mergeMap(item => this.http.post('/api/items', item).pipe(
    catchError(error => {
      console.error(`Erreur lors du traitement de l'élément ${item.id}`, error);
      return of(null); // Continue le flux malgré l'erreur
    })
  ))
).subscribe();
```

Elodie TURLIER
Angular, IA, agilité & storytelling

11

EN RÉSUMÉ ET PASSE À L'ACTION !

Utilise ***mergeMap*** pour gérer efficacement des flux multiples simultanément dans tes applis Angular.

✓ Requêtes HTTP parallèles
✓ Optimisation maximale des performances
✓ Traitements non annulables

Pour approfondir:
La doc officiel Rxjs | merge map
https://rxjs.dev/api/operators/mergeMap

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