

⚡ Accélérez vos APIs Spring Boot

💬 Grâce à une mise en cache intelligente

Découvrez comment **@Cacheable**, **Caffeine** et **Redis** peuvent booster vos performances... sans complexité !



Le problème

Des appels redondants, trop coûteux

Une même requête revient 100 fois par minute ?

Et chaque appel relance un accès à la base ?


 Résultat : API lente, base saturée...



La solution

Mettre en cache les bons appels

Avec **@Cacheable**, Spring Boot vous permet de répondre en millisecondes, sans refaire tout le travail.

 Et c'est personnalisable avec Caffeine, Redis, etc.



@Cacheable en pratique

Un simple ajout d'annotation

```
@Cacheable("users")  
public User getUser(String id) {  
    return repository.findById(id);  
}
```

➡ Résultat: la première fois, on calcule. Ensuite, c'est instantané





Caffeine pour la vitesse

La meilleure option en mémoire locale

- ✓ ultra rapide (basé sur ***ConcurrentMap***)
- ✓ TTL et t aille configurables
- ✓ Parfait pour du cache temporaire
- ☞ Facile à intégrer dans ***application.yml***



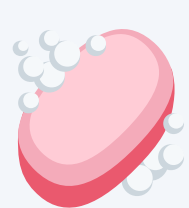
Redis pour le distribué

Le cache partagé entre instances

- ✓ Idéal en microservices ou cluster
- ✓ Support TTL, invalidation
- ✓ Externe mais ultra scalable

💡 Utilisez ***spring-data-redis*** pour l'intégrer proprement





À faire / À éviter

Bonnes pratiques essentielles

✓ Cache des appels déterministes

✓ TTL raisonnable

✓ Invalidation claire

✗ Ne jamais cacher un appel POST

✗ Ne pas surcharger avec trop de clés





Checklist Express

Avant d'activer le cache

☒ L'appel est lent ?

☒ Il est souvent répété ?

☒ Les données ne changent pas toutes les 5 secondes ?

☒ Alors c'est un bon candidat au cache ☒




Et toi ?

Quelle stratégie de cache utilises-tu ?

Redis ? Caffeine ?

Manuelle ou automatique ?

 Partagez vos retours ou
questions !

