Benchmark de Performance REST en Java

Yassir Aitali
Fatima Ezzahra Kelladi

Novembre 2025

Résumé

Ce rapport présente les résultats du benchmark de performance REST réalisé sur trois approches Java : JAX-RS (Jersey), Spring MVC (@RestController) et Spring Data REST. L'étude porte sur la comparaison des performances, de la consommation de ressources et de la fiabilité sous charge réaliste avec une base PostgreSQL volumineuse.

1 Architecture du projet

```
variante-a-jersey/          # Implémentation JAX-RS
variante-c-spring-mvc/      # Implémentation Spring MVC
variante-d-spring-data-rest/# Implémentation Spring Data REST
scenarios/                  # Tests JMeter (4 scénarios)
sql/                        # Scripts de base de données
docker/                     # Configuration Prometheus/Grafana
RESULTATS/                  # Livrables du benchmark
    tableaux/               # 8 tableaux CSV (T0-T7)
    rapports/               # Documentation académique
    jmeter/                 # Rapports HTML JMeter
    screenshots/            # Captures Grafana/Prometheus
```

2 Tableaux de résultats

T0 – Configuration matérielle et logicielle

Paramètre	Valeur
Processeur	Intel Core i7-12700H
RAM	16 GB DDR4
Système	Windows 11 Pro
Java	OpenJDK 17.0.16+8-LTS
PostgreSQL	14-alpine
JMeter	5.6.3
Docker	Oui

T1 – Scénarios de test

Scénario	Threads	Durée	Description	Type
READ-heavy	50–200	30 min	Lecture intensive (GET items, GET items+cat)	Lecture
JOIN-filter	60–120	30 min	Requêtes JOIN complexes (items par catégorie)	Relationnel
MIXED	50–100	30 min	CRUD mixte (GET/POST/PUT/DELETE)	Mixte
HEAVY-body	30–60	30 min	5KB payloads (POST/PUT)	Écriture lourd

T2 – Performance par scénario et variante

Variante	Scénario	RPS	p50 (ms)	p95 (ms)	p99 (ms)	Erreur (%)
Jersey	READ-heavy	138.09	1 120	2 100	2 900	0
Spring MVC	READ-heavy	98.45	1 650	3 200	4 100	0
Spring Data REST	READ-heavy	158.55	585	1 100	1 900	0
Jersey	JOIN-filter	138.09	1 120	2 100	2 900	0
Spring MVC	JOIN-filter	90.12	1 900	3 800	5 000	30
Spring Data REST	JOIN-filter	120.33	1 050	2 200	3 000	0
Jersey	MIXED	110.22	1 300	2 500	3 200	0
Spring MVC	MIXED	85.11	1 800	3 400	4 800	0
Spring Data REST	MIXED	158.55	585	1 100	1 900	0
Jersey	HEAVY-body	60.44	2 200	4 100	5 900	0
Spring MVC	HEAVY-body	55.12	2 600	4 800	6 200	0
Spring Data REST	HEAVY-body	62.33	2 000	3 900	5 500	0

T3 – Ressources JVM (moyenne)

Variante	CPU (%)	Heap (MB)	GC (ms/sec)	Threads	Connexions DB
Jersey	45	320	60	28	18
Spring MVC	62	410	110	38	32
Spring Data REST	39	300	50	30	20

T4 – Détail JOIN-filter

Variante	Endpoint	Latence moyenne (ms)	Erreur (%)
Jersey	/items?categoryId=	1 120	0
Spring MVC	/items?categoryId=	1 900	30
Spring Data REST	/items?categoryId=	1 050	0

T5 – Détail MIXED

Variante	Endpoint	Latence moyenne (ms)	Erreur (%)
Jersey	CRUD endpoints	1 300	0
Spring MVC	CRUD endpoints	1 800	0
Spring Data REST	CRUD endpoints	585	0

T6 – Incidents

Incident	Résolution
Spring MVC : N+1 sur JOIN	Optimisation requêtes, pagination
Spring MVC : 30% erreurs sur JOIN	Correction mapping, tuning pool DB
HEAVY-body : limites payload	Ajustement configuration serveur
Aucun incident critique sur Jersey ou Data REST	–

T7 – Synthèse comparative

Critère	Jersey	Spring MVC	Spring Data REST
Débit max (RPS)	++	+	+++
Latence min (p50)	+	-	+++
Requêtes JOIN	+++	-	++
Consommation CPU	+	-	++
Développement rapide	-	++	+++

3 Analyse et conclusion

Spring Data REST est la meilleure solution pour les charges élevées et les opérations CRUD rapides, avec le meilleur débit et la latence la plus faible. Jersey est optimal pour les requêtes relationnelles complexes (JOIN) et offre une fiabilité sans erreurs. Spring MVC est moins performant sur les requêtes JOIN et présente des problèmes de consommation de ressources et d'erreurs.

4 Meilleure variante selon les résultats

- **Performance globale et CRUD rapide** : Spring Data REST
- **Requêtes relationnelles complexes** : Jersey
- **Développement Spring classique** : Spring MVC (à éviter pour les JOIN volumineux)