Spring Boot Caching – Cheat Sheet

Java Fleet · Spring Boot Aufbau · Tag 6 · von Elyndra Valen

Die 4 wichtigsten Annotationen

Annotation	Wirkung
`@Cacheable("name")`	Ergebnis wird gecacht. Methode läuft nur beim ersten Aufruf pro Parameter-Kombination.
`@CacheEvict("name")`	Löscht Einträge aus dem Cache, z.B. nach Update/Delete. allEntries=true leert alles.
`@CachePut("name")`	Methode läuft IMMER, Ergebnis wird zusätzlich im Cache aktualisiert.
`@Caching(...)`	Kombiniert mehrere Cache-Annotationen auf einer Methode.

Setup in 2 Schritten

```
1. Dependency (pom.xml):
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-cache</artifactId>
</dependency>

2. Aktivieren (Application- oder Config-Klasse):
@EnableCaching
```

Caffeine für Production (statt Simple Cache)

```
@Bean
public CacheManager cacheManager() {
    CaffeineCacheManager cm = new CaffeineCacheManager("calculations");
    cm.setCaffeine(Caffeine.newBuilder()
        .maximumSize(100)                      // Limit! Nie unbegrenzt
        .expireAfterWrite(10, TimeUnit.MINUTES)  // TTL
        .recordStats());                        // Monitoring aktivieren
    return cm;
}
```

	Simple Cache	Caffeine
Größen-Limit	■ Nein	■ Ja (maximumSize)
TTL / Ablauf	■ Nein	■ Ja (expireAfterWrite)
Monitoring	■ Nein	■ Ja (recordStats)
Einsatz	Lernen / Prototyp	Production

■■ Der Cache-Key-Kollisions-Bug

Zwei Methoden mit demselben Cache-Namen und denselben Parametern kollidieren! Spring's Standard-Key nutzt NUR die Parameter, nicht den Methodennamen:

```
@Cacheable(value = "calculations",
key = "#root.methodName + '_' + #a + '_' + #b")
public double add(double a, double b) { ... }
```

Merksatz: Teilen sich mehrere Methoden einen Cache-Namen, MUSS der Key den Methodennamen enthalten – sonst gibt es falsche Ergebnisse!

Best Practices

• Immer Limits setzen: .maximumSize(...) – niemals unbegrenzt

• Sprechende Cache-Namen: "user-profiles" statt "cache1"
• recordStats() in Production immer aktivieren
• Nach Datenänderung: @CacheEvict oder @CachePut nutzen
• Cachen: Produktkataloge, Stammdaten, Konfigurationen
• NICHT cachen: Echtzeit-Daten, Sessions, sensible Daten (Passwörter, Kreditkarten!)
• Nur public-Methoden cachen – @Cacheable an private Methoden wirkt nicht (Proxy!)
• Self-Invocation vermeiden: gecachte Methoden immer von außen (Controller) aufrufen

Troubleshooting – Kurzreferenz

Symptom	Ursache	Fix
Cache wirkt nie	@EnableCaching fehlt	Config- oder Application-Klasse prüfen
Cache wirkt nie	Methode ist private	Nur public-Methoden cachen
Cache wirkt nie	Self-Invocation	Methode über Controller/Bean von außen aufrufen
OutOfMemoryError	Simple Cache ohne Limit	Zu Caffeine wechseln + maximumSize setzen
Falsches Ergebnis	Key-Kollision	#root.methodName in den Key aufnehmen
@CacheEvict wirkt nicht	Cache-Name stimmt nicht überein	Namen in @Cacheable/@CacheEvict abgleichen

Caffeine vs. Redis (Bonus)

	Caffeine	Redis
Läuft als	embedded (in der JVM)	externer Server
Installation	nur Maven-Dependency	OS-Installation / Docker nötig
Geschwindigkeit	~0.001ms	~1–5ms (Netzwerk)
Überlebt Restart	Nein	Ja
Geteilt zw. Instanzen	Nein	Ja