# ShedLock with Spring

### 

**CREATE** **TABLE** shedlock (

name **VARCHAR**(64),

lock\_until **TIMESTAMP**(3) **NULL**,

locked\_at **TIMESTAMP**(3) **NULL**,

locked\_by **VARCHAR**(255),

**PRIMARY** KEY (name)

)

@Component

**class** **BaeldungTaskScheduler** {

@Scheduled(cron = "0 0/15 \* \* \* ?")

@SchedulerLock(name = "TaskScheduler\_scheduledTask",

lockAtLeastFor = "PT5M", lockAtMostFor = "PT14M")

**public** **void** **scheduledTask**() {

// ...

}

}

ShedLock używa bazy danych lub innego centralnego mechanizmu przechowywania do zarządzania blokadami. Gdy zadanie cronowe próbuje się uruchomić, ShedLock sprawdza, czy istnieje aktywna blokada dla tego zadania. Jeśli nie, tworzy blokadę i pozwala na uruchomienie zadania. Inne instancje tego samego zadania sprawdzają blokadę i nie uruchamiają się, jeśli blokada już istnieje.

### **Jak działa blokowanie z ShedLock?**

* **lockAtMostFor**: Maksymalny czas, przez jaki blokada jest utrzymywana. Nawet jeśli zadanie zakończy się wcześniej, blokada zostanie zwolniona dopiero po upływie tego czasu.
* **lockAtLeastFor**: Minimalny czas, przez jaki blokada jest utrzymywana. Nawet jeśli zadanie zakończy się wcześniej, blokada zostanie utrzymana przez ten czas.

# Quartz with clustering

### 

### **spring.datasource.url=jdbc:postgresql://localhost:5432/yourdbname spring.datasource.username=yourusername spring.datasource.password=yourpassword spring.jpa.hibernate.ddl-auto=update # Quartz properties spring.quartz.job-store-type=jdbc**

### 

<https://www.quartz-scheduler.org/documentation/quartz-1.8.6/configuration/ConfigJDBCJobStoreClustering.html>

https://dev.to/hronom/spring-boot-quartz-scheduler-in-cluster-mode-35ej

### **Pesimistic BLOCKING JPA**

### 

### **Pesymistyczne blokowanie w JPA**

W JPA można używać dwóch rodzajów pesymistycznego blokowania:

1. **Pesymistyczne blokowanie odczytu (PESSIMISTIC\_READ)**:
   * Blokuje rekord w sposób, który uniemożliwia innym transakcjom jego modyfikację, ale pozwala na jego odczyt.
   * Używa się go, gdy chcemy zapewnić, że dane nie będą zmienione podczas ich przetwarzania przez bieżącą transakcję.
2. **Pesymistyczne blokowanie zapisu (PESSIMISTIC\_WRITE)**:
   * Blokuje rekord w sposób, który uniemożliwia zarówno jego modyfikację, jak i odczyt przez inne transakcje.
   * Jest bardziej restrykcyjny i używa się go, gdy chcemy zapewnić pełną kontrolę nad danymi podczas ich przetwarzania.

### **Implementacja pesymistycznego blokowania w JPA z PostgreSQL**

Aby użyć pesymistycznego blokowania w JPA, można skorzystać z adnotacji @Lock w połączeniu z metodami EntityManager lub poprzez zapytania JPQL.

#### **Przykład użycia EntityManager**

java

import javax.persistence.EntityManager;

import javax.persistence.LockModeType;

import javax.persistence.PersistenceContext;

import javax.transaction.Transactional;

@Transactional

public class MyService {

@PersistenceContext

private EntityManager entityManager;

public MyEntity findAndLockEntity(Long id) {

// Pessimistic read lock

return entityManager.find(MyEntity.class, id, LockModeType.PESSIMISTIC\_READ);

}

public void updateEntity(Long id, String newValue) {

// Pessimistic write lock

MyEntity entity = entityManager.find(MyEntity.class, id, LockModeType.PESSIMISTIC\_WRITE);

entity.setValue(newValue);

entityManager.persist(entity);

}

}

import javax.persistence.EntityManager;

import javax.persistence.LockModeType;

import javax.persistence.PersistenceContext;

import javax.transaction.Transactional;

@Transactional

public class MyService {

@PersistenceContext

private EntityManager entityManager;

public void updateEntity(Long id, String newValue) {

MyEntity entity = entityManager.find(MyEntity.class, id, LockModeType.PESSIMISTIC\_WRITE);

entity.setValue(newValue);

entityManager.persist(entity);

}

}

Transakcja T1 wykonuje find na rekordzie MyEntity z identyfikatorem id przy użyciu LockModeType.PESSIMISTIC\_WRITE.

Jeśli inna transakcja T2 również spróbuje zablokować ten sam rekord z PESSIMISTIC\_WRITE, będzie musiała czekać, aż T1 zakończy (commit lub rollback).