LESSON 18

SPRING BOOT: SCHEDULED JOBS

SPRING BOOT: SCHEDULED JOBS

- @Scheduled Annotation
- Fixed delay or Fixed rate
- Cron expressions
- Cron expression macros
- Running Tasks in Parallel
- TaskScheduler

@SCHEDULED ANNOTATION

@Scheduled аннотация используется для планирования задач. Информация о триггере должна быть предоставлена вместе с этой аннотацией. Он принимает один атрибут из cron, fixedDelay или fixedRate для указания расписания выполнения в разных форматах.

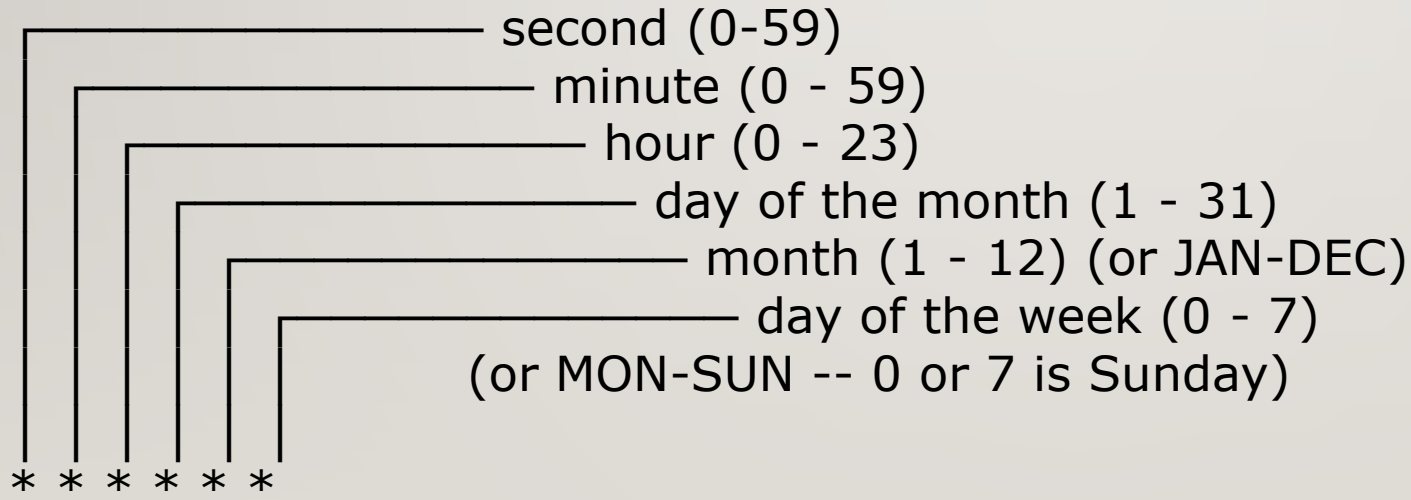
FIXED DELAY OR FIXED RATE

Мы используем fixedDelay атрибут, чтобы настроить выполнение задания после фиксированной задержки, он означает фиксированный интервал между концом предыдущего задания и началом нового задания. **Новое задание всегда будет ждать завершения предыдущего задания.** Его следует использовать в ситуациях, когда вызовы методов должны происходить последовательно.

Мы используем атрибут fixedRate, чтобы указать интервал для выполнения задания через фиксированный интервал времени. Его следует использовать в ситуациях, когда вызовы методов независимы. **Время выполнения метода не учитывается при решении, когда начинать следующее задание.**

CRON EXPRESSIONS

Выражение cron представляет собой строку из шести-семи полей, разделенных пробелом, для представления триггеров на секунду, минуту, час, день месяца, месяц, день недели и, необязательно, год. Однако выражение cron в Spring Scheduler состоит из шести полей, как показано ниже:



CRON EXPRESSIONS

Например, выражение cron: 0 15 10 * * *запускается в 10:15 каждый день (каждую 0-ю секунду, 15-ю минуту, 10-й час, каждый день). *указывает, что выражение cron соответствует всем значениям поля. Например, *в поле минут означает каждую минуту.

Такие выражения, как 0 0 * * * *, трудно читать. Чтобы улучшить читаемость, Spring поддерживает макросы для представления часто используемых последовательностей.

Пример:

@Scheduled(cron = "@hourly")

CRON EXPRESSIONS MACROS

Spring предоставляет следующие макросы:

@hourly,

@yearly,

@monthly,

@weekly, а также

@daily

RUNNING TASKS IN PARALLEL

По умолчанию Spring использует для запуска задач локальный однопоточный планировщик . В результате, даже если у нас есть несколько методов @Scheduled , каждый из них должен ждать, пока поток завершит выполнение предыдущей задачи.

Если наши задачи действительно независимы, их удобнее запускать параллельно. Для этого нам нужно предоставить TaskScheduler , который лучше соответствует нашим потребностям.