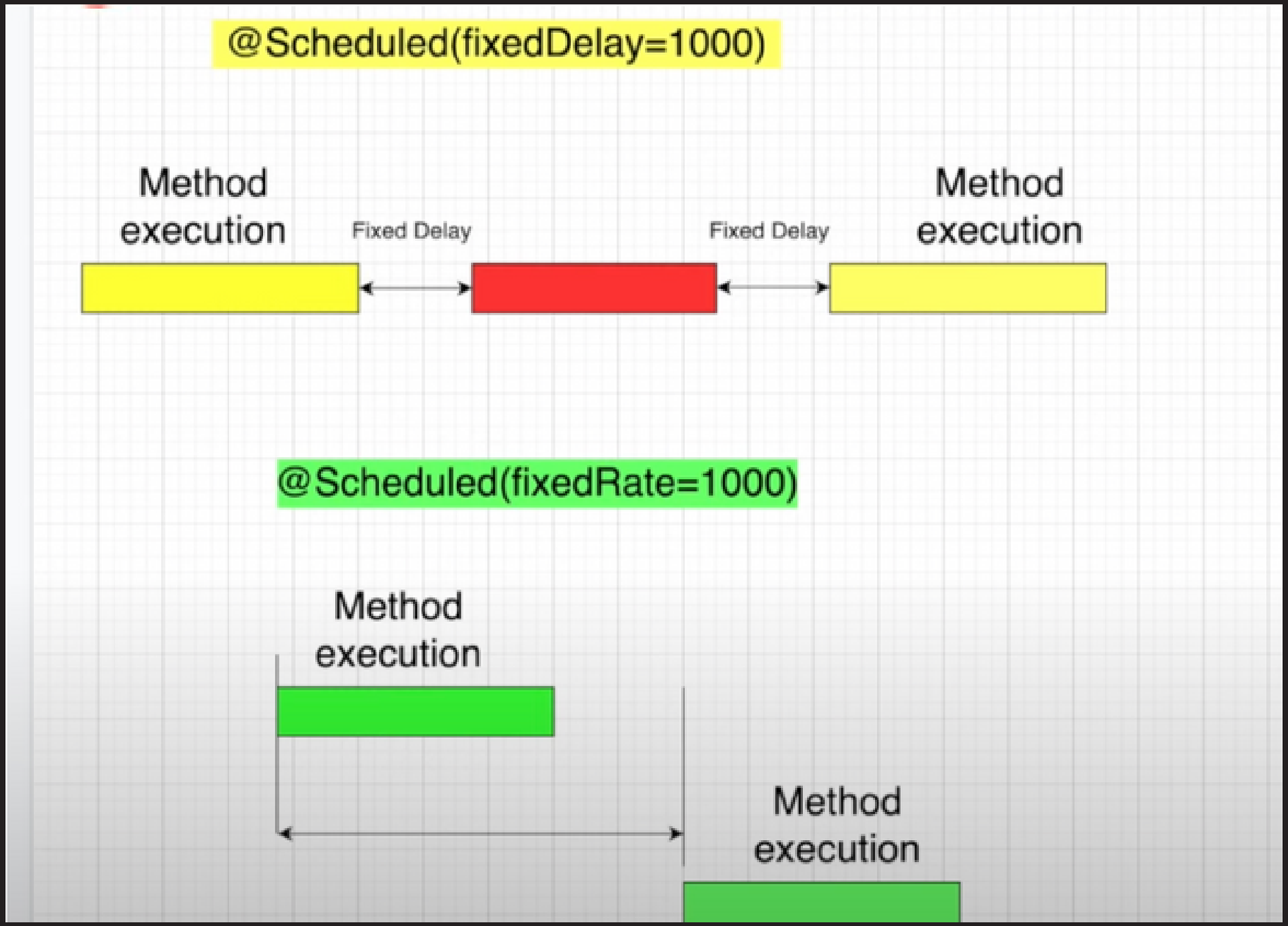
Synchronous & Asynchronous Task Schedulers

```java
@Scheduled(fixedDelay = 1000)
public void scheduler() throws InterruptedException {
    LocalDateTime current = LocalDateTime.now();
    DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
    String formattedDateTime = current.format(format);
    log.info("Scheduler time " + formattedDateTime);
}
```

```
c.e.S.SpringSchedulerApplication         : Started SpringSchedulerApplication in 1.365 seconds (process running for 3.053)
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:41
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:42
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:43
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:44
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:45
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:46
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:47
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:48
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:49
com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:18:50
```

```java
    @Scheduled(fixedRate = 1000)
    public void scheduler1() throws InterruptedException {
        LocalDateTime current = LocalDateTime.now();
        DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
        String formattedDateTime = current.format(format);
        log.info("Scheduler time " + formattedDateTime);
        Thread.sleep( millis: 1000);
//        won't wait for the sleep time
    }
```

```
1:45:50.122+05:30  INFO 2147 --- [           main] c.e.S.SpringSchedulerApplication         : Started SpringSchedulerApplication in 1.491
1:45:51.121+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:51
1:45:52.123+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:52
1:45:53.128+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:53
1:45:54.132+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:54
1:45:55.133+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:55
1:45:56.136+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:56
1:45:57.140+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:57
1:45:58.142+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:58
1:45:59.146+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:45:59
1:46:00.152+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:46:00
1:46:01.157+05:30  INFO 2147 --- [   scheduling-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 01:46:01
```


@Scheduled(fixedDelay=1000)
Method execution
Fixed Delay
Fixed Delay
Method execution
@Scheduled(fixedRate=1000)
Method execution
Method execution

```java
@Scheduled(fixedRate = 1000, initialDelay = 1000)
public void scheduler2() throws InterruptedException {
    LocalDateTime current = LocalDateTime.now();
    DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
    String formattedDateTime = current.format(format);
    log.info("Scheduler time " + formattedDateTime);
    Thread.sleep( millis: 1000);
}
```

for larger rates
executed after every 2 hours PT02H (PT02S)

```java
//    for larger rates
//    executed after every 2 hours PT02H ( PT02S )
   @Scheduled(fixedRateString = "PT02H", initialDelay = 1000)
   public void scheduler3() throws InterruptedException {
       LocalDateTime current = LocalDateTime.now();
       DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
       String formattedDateTime = current.format(format);
       log.info("Scheduler time " + formattedDateTime);
       Thread.sleep( millis: 1000);
   }
```

For parallel execution we need to use @Async Annotation

```java
//    For parallel execution we need to use @Async and @EnableAsync Annotations
    @Async
    @Scheduled(fixedRateString = "PT02S", initialDelay = 1000)
    public void scheduler4() throws InterruptedException {
        LocalDateTime current = LocalDateTime.now();
        DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
        String formattedDateTime = current.format(format);
        log.info("Scheduler time " + formattedDateTime);
        Thread.sleep( millis: 1000);
    }
```

```
2023-01-04T02:04:46.533+05:30  INFO 2345 --- [         task-3] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:04:46
2023-01-04T02:04:48.539+05:30  INFO 2345 --- [         task-4] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:04:48
2023-01-04T02:04:50.538+05:30  INFO 2345 --- [         task-5] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:04:50
2023-01-04T02:04:52.537+05:30  INFO 2345 --- [         task-6] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:04:52
2023-01-04T02:04:54.537+05:30  INFO 2345 --- [         task-7] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:04:54
2023-01-04T02:04:56.538+05:30  INFO 2345 --- [         task-8] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:04:56
2023-01-04T02:04:58.537+05:30  INFO 2345 --- [         task-1] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:04:58
2023-01-04T02:05:00.537+05:30  INFO 2345 --- [         task-2] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:05:00
2023-01-04T02:05:02.537+05:30  INFO 2345 --- [         task-3] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:05:02
2023-01-04T02:05:04.537+05:30  INFO 2345 --- [         task-4] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:05:04
2023-01-04T02:05:06.537+05:30  INFO 2345 --- [         task-5] com.example.SpringScheduler.Scheduler    : Scheduler time 04-01-2023 02:05:06
```

we can add the schedulers via cron expression

```
public abstract String cron()
```

A cron-like expression, extending the usual UN*X definition to include triggers on the second, minute, hour, day of month, month, and day of week.

For example, `"0 * * * * MON-FRI"` means once per minute on weekdays (at the top of the minute - the 0th second).

The fields read from left to right are interpreted as follows.

- second
- minute
- hour
- day of month
- month
- day of week

The special value `"-"` indicates a disabled cron trigger, primarily meant for externally specified values resolved by a `${...}` placeholder.

Returns: an expression that can be parsed to a cron schedule

See Also: `org.springframework.scheduling.support.CronExpression.parse(String)`

Gradle: org.springframework:spring-context:6.0.3 (spring-context-6.0.3.jar)

```java
//    we can add the schedulers via cron expression
//    will be executed after every 2 seconds
   @Scheduled(cron = "*/2 * * * * *")
   public void scheduler5() throws InterruptedException {
       LocalDateTime current = LocalDateTime.now();
       DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
       String formattedDateTime = current.format(format);
       log.info("Scheduler time " + formattedDateTime);
       Thread.sleep( millis: 1000);
   }
```

```java
//    0 */2 * * * *
//    0 0 */2 * * *
//    0 */2 * * * TUE

   @Scheduled(cron = "${cron.expression.value}")
   public void scheduler6() throws InterruptedException {
       LocalDateTime current = LocalDateTime.now();
       DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
       String formattedDateTime = current.format(format);
       log.info("Scheduler time " + formattedDateTime);
       Thread.sleep( millis: 1000);
   }
```

```
cron.expression.value = 0 */2 * * * *
```

We can run tasks parallel via @Async annotation and Task Scheduler

```
1 usage
@SpringBootApplication
@EnableScheduling
public class SpringSchedulerApplication {

    public static void main(String[] args) { SpringApplication.run(SpringSchedulerApplication.c


    @Bean
    public TaskScheduler taskScheduler(){
        ThreadPoolTaskScheduler threadPoolTaskScheduler = new ThreadPoolTaskScheduler();
//      threadPoolTaskScheduler.setPoolSize(5);
        threadPoolTaskScheduler.setThreadNamePrefix("ThreadPoolTaskScheduler");
        return threadPoolTaskScheduler;
    }

}
```

```
cron.expression.value = 0 */2 * * * *
spring.task.scheduling.pool.size=5
```

```
    public int noOfThreads1(){
        taskScheduler.setPoolSize(3);
        return 3;
    }

    public int noOfThreads2(){
        taskScheduler.setPoolSize(5);
        return 5;
    }
```

```java
@PostConstruct()
public void schedule7() throws InterruptedException{
    taskScheduler.scheduleWithFixedDelay(
            new RunnableTask("Specific time, 3 Seconds from now"),
            delay: 3000
    );
    taskScheduler.scheduleWithFixedDelay(
            new RunnableTask("Fixed 1 second Delay"), delay: 1000);
    taskScheduler.scheduleAtFixedRate(
            new RunnableTask("Fixed Rate of 2 seconds") , period: 2000);
    taskScheduler.scheduleAtFixedRate(new RunnableTask(
            "Fixed Rate of 2 seconds"), period: 3000);
    CronTrigger cronTrigger
            = new CronTrigger( expression: "10 * * * * ?");
    taskScheduler.schedule(new RunnableTask("Cron Trigger"), cronTrigger);
    PeriodicTrigger periodicTrigger
            = new PeriodicTrigger( period: 2000, TimeUnit.MICROSECONDS);
    taskScheduler.schedule(
            new RunnableTask("Periodic Trigger"), periodicTrigger);
    periodicTrigger.setFixedRate(true);
    periodicTrigger.setInitialDelay(1000);
}
```

SpringSchedulerApplication

Console | Actuator

```
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
Thu Jan 05 00:15:35 IST 2023 Runnable Task with Periodic Trigger on thread ThreadPoolTaskScheduler1
```

```java
@Configuration
@ComponentScan(
        basePackages="com.example.SpringScheduler",
        basePackageClasses={SpringSchedulerApplication.class})
public class ThreadPoolTaskSchedulerConfig {

    @Bean
    public ThreadPoolTaskScheduler threadPoolTaskScheduler(){
        ThreadPoolTaskScheduler threadPoolTaskScheduler
                = new ThreadPoolTaskScheduler();
        threadPoolTaskScheduler.setPoolSize(5);
        threadPoolTaskScheduler.setThreadNamePrefix(
                "ThreadPoolTaskScheduler");
        return threadPoolTaskScheduler;
    }
}
```

```java
5 usages
@SpringBootApplication
@EnableScheduling
public class SpringSchedulerApplication extends SpringBootServletInitializer {


    public static void main(String[] args) {
        ConfigurableApplicationContext context = SpringApplication.run(SpringSchedulerApplication.class, args);
        context.getBean(SpringSchedulerApplication.class).onInitialLoad();
    }

    2 usages
    public void onInitialLoad() {
        System.out.println("onInitialLoad SpringSchedulerApplication");
    }


    @Bean
    public ThreadPoolTaskScheduler taskScheduler(){
        ThreadPoolTaskScheduler threadPoolTaskScheduler = new ThreadPoolTaskScheduler();
//      threadPoolTaskScheduler.setPoolSize(5);
        threadPoolTaskScheduler.setThreadNamePrefix("ThreadPoolTaskScheduler");
        return threadPoolTaskScheduler;
    }

}
```