

SPRING Scheduler



UP ASI
Bureau E204

Plan du Cours

- **Introduction**
- **@Scheduled Annotation**
- **Fixed delay vs Fixed rate**
- **Cron expression**

Introduction

- La planification(scheduling) consiste à exécuter les tâches pendant une période de temps spécifique.
- Spring Boot Scheduling est une fonctionnalité pratique qui nous permet de planifier des tâches dans nos applications Spring Boot.
- Par exemple, si vous voulez que votre application exécute une tâche après un intervalle fixe ou en fonction d'un calendrier.
- **Le scheduler fait partie du module Core du framework Spring (Pas de dépendances à ajouter dans le pom.xml).**

@Scheduled Annotation

- Spring Boot utilise l'annotation **@Scheduled** pour la planification des tâches.
- Il faut respecter certaines règles lors de l'utilisation de cette annotation :
 1. Les méthodes doivent être sans paramètre.
 2. Le type de retour de la méthode doit être void.
- Pour activer la planification (scheduling), il faut ajouter l'annotation **@EnableScheduling** à la classe main.

@EnableScheduling

@SpringBootApplication


```
public class TpStockProjectApplication {  
    public static void main(String[] args) {  
        SpringApplication.run(TpStockProjectApplication.class, args);  
    }  
}
```

@Scheduled Annotation

- Spring Boot permet de créer facilement une tâche de planification (scheduling task).
- Il suffit d'annoter la méthode avec l'annotation **@Scheduled**.
- L'annotation **@Scheduled** définit la planification (par exemple, quand la méthode sera exécutée, etc.).
- Nous pouvons passer certains paramètres à l'annotation pour personnaliser le comportement.

Fixed Rate

- Pour planifier un déclenchement de méthode à une date fixe, nous pouvons utiliser le paramètre **fixedRate** dans l'annotation **@Scheduled**.
- Fixed Rate : permet à Spring d'exécuter la tâche à des intervalles périodiques, même si la dernière exécution est en cours.



exécuter la méthode
toutes les 3 secondes

```
@Scheduled(fixedRate = 3000)
public void fixedRateMethod() {
    log.info(" Schedule tasks using fixedRate");
}
```

Fixed Rate

[illegible]

```

2023-03-13 01:09:08.199 INFO 27328 --- [ restartedMain] c.e.demo.AddressServiceApplication : Starting AddressServiceApplication using Java 1.8.0_60 on DESKTOP-MSHJESB with PID 27328 ()
2023-03-13 01:09:08.202 INFO 27328 --- [ restartedMain] c.e.demo.AddressServiceApplication : No active profile set, falling back to 1 default profile: "default"
2023-03-13 01:09:08.256 INFO 27328 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable
2023-03-13 01:09:08.256 INFO 27328 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property to '0'
2023-03-13 01:09:09.208 INFO 27328 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-03-13 01:09:09.222 INFO 27328 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-03-13 01:09:09.222 INFO 27328 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.71]
2023-03-13 01:09:09.294 INFO 27328 --- [ restartedMain] o.a.c.c.f.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-03-13 01:09:09.294 INFO 27328 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1037 ms
2023-03-13 01:09:09.643 INFO 27328 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-03-13 01:09:09.672 INFO 27328 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
schedule tasks using fixedRate - 01:09:09.681
2023-03-13 01:09:09.683 INFO 27328 --- [ restartedMain] c.e.demo.AddressServiceApplication : Started AddressServiceApplication in 1.802 seconds (JVM running for 2.555)
schedule tasks using fixedRate - 01:09:12.692
schedule tasks using fixedRate - 01:09:15.690
schedule tasks using fixedRate - 01:09:18.694
schedule tasks using fixedRate - 01:09:21.692
schedule tasks using fixedRate - 01:09:24.691
schedule tasks using fixedRate - 01:09:27.690

```

Fixed Delay

- Pour fixer un délai fixe entre la dernière exécution et le début de l'exécution suivante, nous pouvons utiliser le paramètre **fixedDelay**.
- Ce paramètre compte le délai **après l'exécution de la dernière invocation**.

```
@Scheduled(fixedDelay = 2000)
public void fixedDelayMethod() {
    log.info("Schedule tasks using fixedDelay");
}
```

Les tâches sont déclenchées avec un **retard de 2 secondes**.



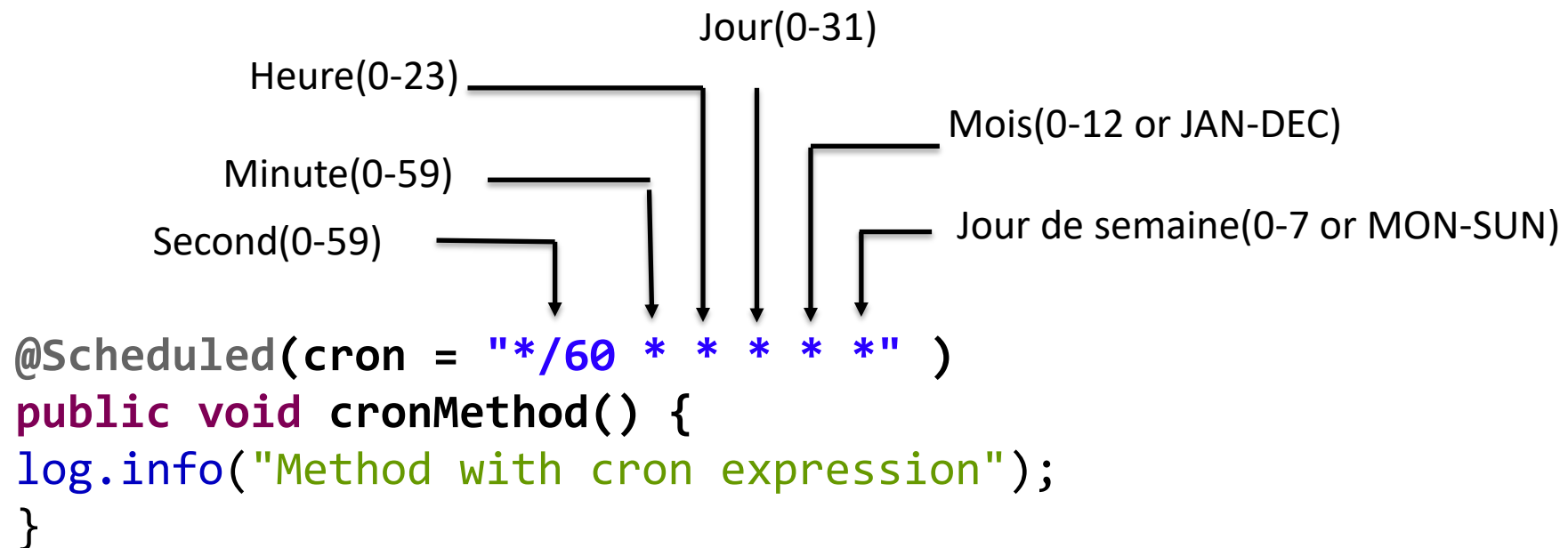
Fixed Delay

```
:: Spring Boot ::                (v2.7.8)

2023-03-13 01:05:37.326 INFO 19988 --- [ restartedMain] c.e.demo.AddressServiceApplication : Starting AddressServiceApplication using Java 1.8.0_60 on DESKTOP-MSHJESB wit
2023-03-13 01:05:37.326 INFO 19988 --- [ restartedMain] c.e.demo.AddressServiceApplication : No active profile set, falling back to 1 default profile: "default"
2023-03-13 01:05:37.548 INFO 19988 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-03-13 01:05:37.548 INFO 19988 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-03-13 01:05:37.548 INFO 19988 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.71]
2023-03-13 01:05:37.569 INFO 19988 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-03-13 01:05:37.570 INFO 19988 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 241 ms
2023-03-13 01:05:37.663 INFO 19988 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2023-03-13 01:05:37.670 INFO 19988 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-03-13 01:05:37.674 INFO 19988 --- [ restartedMain] c.e.demo.AddressServiceApplication : Started AddressServiceApplication in 0.37 seconds (JVM running for 100.407)
schedule tasks using fixedDelay - 01:05:37.674
2023-03-13 01:05:37.675 INFO 19988 --- [ restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
schedule tasks using fixedDelay - 01:05:39.680
schedule tasks using fixedDelay - 01:05:41.694
schedule tasks using fixedDelay - 01:05:43.705
schedule tasks using fixedDelay - 01:05:45.707
schedule tasks using fixedDelay - 01:05:47.718
schedule tasks using fixedDelay - 01:05:49.724
schedule tasks using fixedDelay - 01:05:51.729
schedule tasks using fixedDelay - 01:05:53.735
schedule tasks using fixedDelay - 01:05:55.742
schedule tasks using fixedDelay - 01:05:57.749
schedule tasks using fixedDelay - 01:05:59.754
```

Cron expression

- L'expression **Cron** est une façon **flexible** et **puissante** pour planifier les tâches.



Cron expression

- L'attribut 'zone' spécifiait le fuseau horaire. La valeur par défaut est vide, ce qui signifie utiliser le fuseau horaire du serveur

```
@Scheduled(cron="*/60 * * * * ", zone="Europe/Istanbul")
```

Exercise

1. `@Scheduled(cron = "15 * * * * *")`
2. `@Scheduled(cron = "*/15 * * * * *")`
3. `@Scheduled(cron = "0/15 * * * * *")`
4. `@Scheduled(cron = "0 0/30 11 * * *")`
5. `@Scheduled(cron = "0 0 8 ? 4 ?")` or `(cron = "0 0 8 * 4 *")`
6. `@Scheduled(cron = "0 0 9 14 2 SUN,TUE")`

Conclusion

field	Description
FixedRate	Nombre de millisecondes après l'heure de début de la dernière exécution de la méthode. => @retourne la période en millisecondes.
FixedDelay	Nombre de millisecondes après la dernière exécution de la méthode. => @retourne le délai en millisecondes.
cron	Écrire le cron et définir le calendrier. Vous pouvez également spécifier le fuseau horaire. => @retourne une expression qui peut être analysée dans un plan de cron

Travail à faire

Sur le projet (foyer), réaliser les services des slides suivants :

Travail à faire

Partie 6 : Spring Scheduler

Créer un service qui se déclenche toutes les minutes permettant d'afficher la liste des chambres de chaque foyer en respectant la signature et le format de l'affichage suivant :

void listeChambresParBloc()

```
INFO - tn.esprit.foyer.services.BlocServiceImpl - bloc : C ayant une capacité de : 300
INFO - tn.esprit.foyer.services.BlocServiceImpl - Liste des chambres du bloc C :
INFO - tn.esprit.foyer.services.BlocServiceImpl - chambre numéro 15 de type DOUBLE
INFO - tn.esprit.foyer.services.BlocServiceImpl - chambre numéro 23 de type SIMPLE
INFO - tn.esprit.foyer.services.BlocServiceImpl - chambre numéro 26 de type DOUBLE
INFO - tn.esprit.foyer.services.BlocServiceImpl - bloc : D ayant une capacité de : 120
INFO - tn.esprit.foyer.services.BlocServiceImpl - Liste des chambres du bloc D :
INFO - tn.esprit.foyer.services.BlocServiceImpl - chambre numéro 4 de type TRIPLE
```

Travail à faire

Partie 6 : Spring Scheduler

Créer un service qui se déclenche toutes les 5 minutes permettant d'afficher le pourcentage des chambres par type chambre en respectant la signature et le format de l'affichage suivant :

void pourcentageChambreParTypeChambre()

```
2023-11-11 19:01:55 - INFO - tn.esprit.foyer.services.ChambreServiceImpl - nbTotalsChambres : 4
```

```
2023-11-11 19:01:55 - INFO - tn.esprit.foyer.services.ChambreServiceImpl - le pourcentage des chambres pour le type SIMPLE est égale à 25.0
```

```
2023-11-11 19:01:55 - INFO - tn.esprit.foyer.services.ChambreServiceImpl - le pourcentage des chambres pour le type DOUBLE est égale à 50.0
```

```
2023-11-11 19:01:55 - INFO - tn.esprit.foyer.services.ChambreServiceImpl - le pourcentage des chambres pour le type TRIPLE est égale à 25.0
```


Travail à faire

Partie 6 : Spring Scheduler

Créer un service qui se déclenche toutes les 5 minutes permettant d'afficher le nombre de places encore disponible pour chaque chambre pour l'année en cours en respectant la signature et le format de l'affichage suivants :

void nbPlacesDisponiblesParChambreAnneeEnCours()

```
INFO - tn.esprit.foyer.services.ReservationServiceImpl - nb places restantes en 2023 pour la chambre 15 est égale à 0
INFO - tn.esprit.foyer.services.ReservationServiceImpl - nb places restantes en 2023 pour la chambre 23 est égale à 1
INFO - tn.esprit.foyer.services.ReservationServiceImpl - nb places restantes en 2023 pour la chambre 4 est égale à 0
INFO - tn.esprit.foyer.services.ReservationServiceImpl - nb places restantes en 2023 pour la chambre 26 est égale à 2
```

Travail à faire

Partie 5 : Services avancés

En vous inspirant de l'exemple de la méthode suivante :

HashMap<String,Float >calculNouveauMontantInscriptionDesEtudiants() ;

Créer une méthode schedulé qui se lance une seule fois par an (le 31 décembre de chaque année à 9h du matin) et qui calcule pour chaque étudiant le montant des taches de l'année en cours et le soustrait du montant de l'inscription (en sauvegardant le nouveau montant dans la base de données)

Rappel : le montant des taches est égale à la somme de tarif horaire* durée des taches réalisées pour l'année en cours

void updateNouveauMontantInscriptionDesEtudiants () ;

SPRING Scheduler

Si vous avez des questions, n'hésitez pas à nous contacter :

Département Informatique
UP Architectures des Systèmes d'Information
Bureau E204