Spring Boot ve Task Execution & Task Scheduling

harezmi bilişim çözümleri

Task Execution ve Task Scheduling

- Spring, kullanıcı etkileşimi olmadan arka planda ve periyodik çalışacak işler için **asenkron task execution** ve **task scheduling** kabiliyetleri sunmaktadır
- **TaskExecutor** ve **TaskScheduler** temel arayüzlerdir

Task Execution

- **TaskExecutor**, Java SE 5'de **thread pool kavramı**na karşılık gelmektedir
- TaskExecutor, **java.lang.Runnable** arayüzünü implement eden **task nesneleri**ni asenkron biçimde çalıştırır

```
taskExecutor.execute(new Runnable() {

    @Override
    public void run() {
        System.out.println("my task...");
    }

});
```

Task Scheduling

- Runnable task'ların hemen o anda değil de **gelecekte bir zamanda veya periyodik olarak** çalıştırılmaları da gerekebilir
- Spring bunun için de **TaskScheduler** arayüzünü sunmaktadır

```
taskScheduler.schedule(
    new Runnable() {

        @Override
        public void run() {
            System.out.println("my scheduled task");
        }
    },
    new CronTrigger("* 15 9-17 * * MON-FRI"));
```

@Scheduled & @Async Annotasyonları

- TaskExecutor veya TaskScheduler kullanmak **şart değil**dir
- Spring bean'ları içerisinde doğrudan **@Scheduled** ve **@Async** annotasyonlarına sahip metotlar da tanımlayabiliriz
- Bu metotlar da **asenkron ve zamanlanmış biçimde** çalışacaktır

@Scheduled & @Async Annotasyonları

```
@Scheduled(cron="*/5 * * * * MON-FRI")
public void doSomething() {
    //sadece hafta ici calisan bir task
}
```

```
@Scheduled(initialDelay=1000, fixedRate=5000)
public void doSomething() {
    //periyodik calisan bir task
}
```

Explicit biçimde çalıştırılmadıklarından input argüman alamazlar ve herhangi bir değer de dönemezler

Cron veya fixed rate trigger'lar tanımlanabilir

@Scheduled & @Async Annotasyonları

JAVA Eğitimleri

```
@Async
void doSomething(String s) {
    //asenkron calistirilacak bir task
}
```

Scheduled metotların aksine input argüman alabilirler, çünkü explicit biçimde çalıştırılacaklardır

```
@Async
Future<String> doSomething(int i) {
    //asenkron calistirilacak ve deger donen bir task
    return new AsyncResult<String>("test");
}
```

Asenkron tasklar değer de dönebilirler, ancak bu değer Future nesnesi üzerinden erişilebilir

```
@Async("myTaskExecutor")
void doSomething(String s) {
    //asenkron calistirilacak bir task
}
```

Default taskExecutor dışında başka bir TaskExecutor kullanılması da mümkündür

Spring Boot ve Task Execution

- Spring Boot'un **@Async**c anotasyonuna sahip metotları invoke edildiklerinde ayrı bir thread'de çalıştırması için uygulamada **@EnableAsync** anotasyonu **tanımlı olmalı**dır

```
@SpringBootApplication
@EnableAsync
public class PetClinicApplication {
    ...
}
```

Spring Boot ve Task Execution

- Spring Boot, default olarak **SimpleAsyncTaskExecutor** sınıfını kullanır
- Bu konfigürasyonu uygulamaya özel biçimde **değiştirmek mümkün**dür

```
@Configuration
public class AsyncConfig implements AsyncConfigurer {

    @Override
    public Executor getAsyncExecutor() {
        ThreadPoolTaskExecutor taskExecutor =
                    new ThreadPoolTaskExecutor();
        taskExecutor.initialize();
        return taskExecutor;
    }
}
```

Spring Boot ve Task Execution

- Asenkron tasklarda meydana gelebilecek hataları ele almak için de uygulamaya özel bir **AsyncUncaughtExceptionHandler** bean'i tanımlanabilir

```
@Configuration
public class AsyncConfig implements AsyncConfigurer {
    @Override
    public AsyncUncaughtExceptionHandler
                getAsyncUncaughtExceptionHandler() {
        return new AsyncUncaughtExceptionHandler() {
            @Override
            public void handleUncaughtException(Throwable ex, Method method,
Object... params) {
            System.out.println(">>>Handled uncaught exception :" +
ex.getMessage());
            }
        };
    }
}
```

Spring Boot ve Task Scheduling

- Spring Boot'un @**Scheduled** anotasyonu ile işaretlenmiş metotları çalıştırması için ise uygulamada @**EnableScheduling** anotasyonu **mevcut olmalı**dır

```
@SpringBootApplication
@EnableScheduling
public class PetClinicApplication {
    ...
}
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com