

Spring Quartz Entegrasyonu



Quartz Nedir?

- Quartz açık kaynak kodlu ve oldukça popüler bir **job scheduling kütüphanesidir**
- Herhangi bir Java uygulamasına **kolaylıkla entegre edilebilir**
- Spring, Quartz'ın kullanımını **kolaylaştırmak** için bazı **yardımcı sınıflar** sunmaktadır

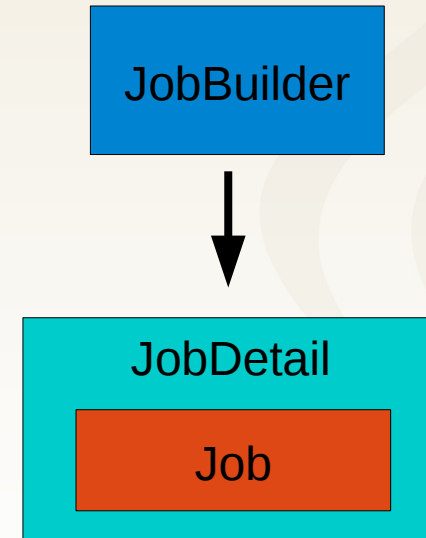
Neden Quartz?

- Aslında Java kendi **built-in timer mekanizmasına** sahiptir
- Basit ihtiyaçlar için Java'daki **java.util.Timer** ve **java.util.TimerTask** sınıfları kullanılabilir
- Ancak persistence mekanizması yoktur, zamanlama imkanları kısıtlıdır, thread-pool kabiliyeti yoktur, taskların yönetimi ile ilgili bir özelliğe de sahip değildir

Quartz Yapı Taşları: Job & JobDetail

- **Job** arayüzünü implement eden sınıf içerisinde asıl iş yapılır
- Job instance'larını tanımlamak için de **JobDetail** kullanılır

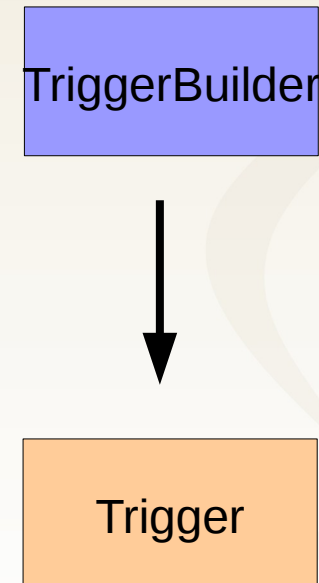
JobDetail instance'ları
JobBuilder ile oluşturulur



JobDetail nesnesi, Job nesnesini
wrap eder

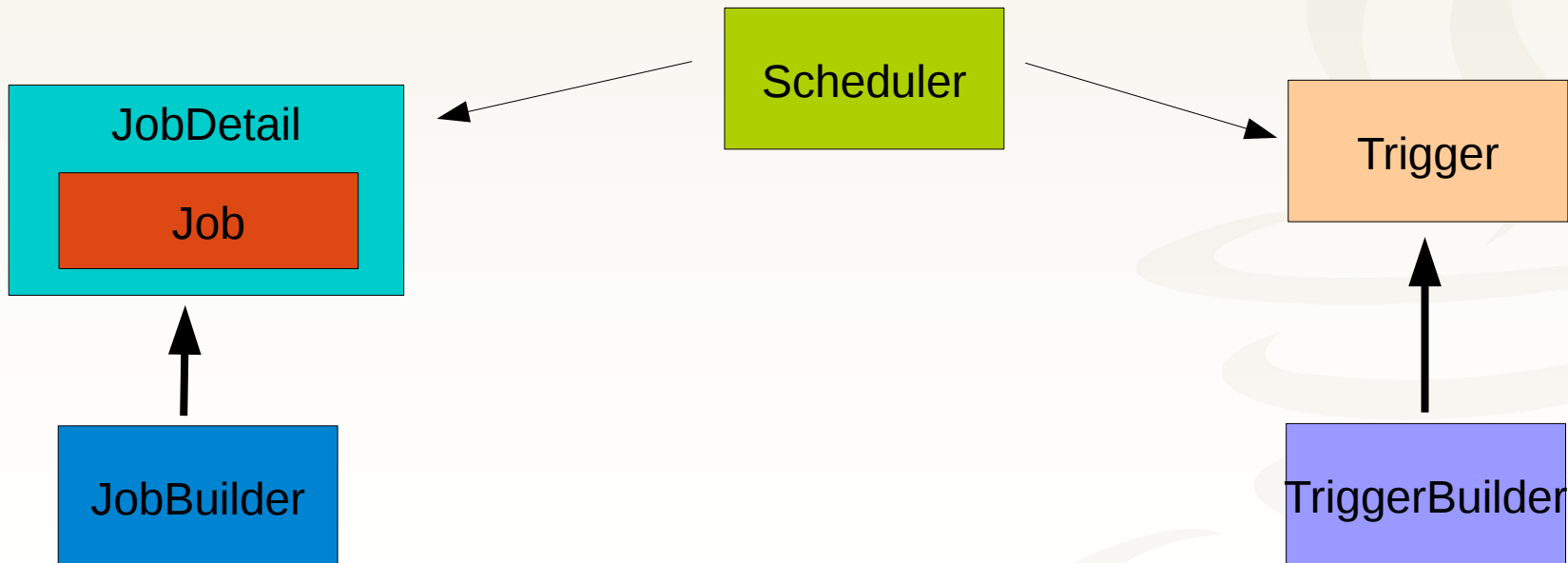
Quartz Yapı Taşları: Trigger

- Job'ların ne zaman çalıştırılacağı bilgisi **Trigger** nesnesi ile tanımlanır
- Trigger nesnelerini oluşturmak için de **TriggerBuilder** kullanılır



Quartz Yapı Taşları: Scheduler

- Task'ların çalıştırma işini ise Scheduler gerçekleştirir
- JobDetail ve Trigger nesnelerini parametre olarak alır



Tek Başına Quartz Örneği: Job & JobDetail

```
public class HelloJob implements Job {  
    public void execute(JobExecutionContext context) throws JobExecutionException {  
        System.err.println("HelloJob is executing now...");  
    }  
}
```

```
JobDetail jobDetail = JobBuilder  
    .newJob(HelloJob.class)  
    .withIdentity("myJob", "group1")  
    .build();
```

} Job instance veya job definition'dır. "**Job class**" belirtilerek oluşturulur

Tek Başına Quartz Örneği: Trigger

```
Trigger trigger = TriggerBuilder.newTrigger()  
    .withIdentity("myTrigger", "group1")  
    .startNow()  
    .withSchedule(simpleSchedule()  
        .withIntervalInSeconds(40)  
        .repeatForever())  
    .build();
```

Job instance'ının invoke
edileceği schedule
oluşturulur

Tek Başına Quartz Örneği: Scheduler

```
try {  
    Scheduler scheduler = StdSchedulerFactory.getDefaultScheduler();  
  
    scheduler.start();  
  
    scheduler.scheduleJob(jobDetail, trigger);  
  
    scheduler.shutdown();  
} catch (SchedulerException ex) {  
    //handle ex...  
}
```

Job instance ve trigger
scheduler'a parametre verilerek
Task başlatılmış olur

Quartz JobDataMap ile Job'a Input Parametere Geçmek

```
public class MsgJob implements Job {  
  
    public void execute(JobExecutionContext context) throws JobExecutionException {  
  
        JobDataMap dataMap = context.getJobDetail().getJobDataMap();  
  
        String msg = dataMap.getString("msg");  
        boolean state = dataMap.getBoolean("state");  
  
        System.out.println("MsgJob says: " + msg + ", state: " + state);  
  
    }  
}
```

```
JobDetail job1 =  
JobBuilder.newJob(MsgJob.class)  
    .withIdentity("myJob1", "group1")  
    .usingJobData("msg", "Hello!")  
    .usingJobData("state", true)  
    .build();
```

```
JobDetail job2 =  
JobBuilder.newJob(MsgJob.class)  
    .withIdentity("myJob2", "group1")  
    .usingJobData("msg", "Goodbye!")  
    .usingJobData("state", false)  
    .build();
```

JobDataMap ile Job nesnesine bir takım parametreler geçmek mümkündür. Bu sayede aynı job class'ı farklı veri ile çalışacak job instance'ları şeklinde Tanımlanabilir.

Entegrasyonu – Job/JobDetail

- Öncelikle Spring Container içerisinde Job tanımları yapılmalıdır
- Bu iki farklı biçimde yapılabilir
 - **JobDetailBean** kullanarak
 - **MethodInvokingJobDetailFactoryBean** kullanarak

JobDetailFactoryBean Kullanımı

```
public class ExampleJob extends QuartzJobBean {  
    private int timeout;  
  
    public void setTimeout(int timeout) {  
        this.timeout = timeout;  
    }  
  
    protected void executeInternal(JobExecutionContext ctx)  
        throws JobExecutionException {  
        // perform task...  
    }  
}
```

Spring'e ait abstract bir sınıftır

```
<bean name="jobDetail"  
class="org.springframework.scheduling.quartz.JobDetailFactoryBean">
```

```
    <property name="jobClass" value="example.ExampleJob" />
```

```
    <property name="jobDataAsMap">
```

```
        <map>
```

```
            <entry key="timeout" value="5" />
```

```
        </map>
```

```
    </property>
```

```
</bean>
```

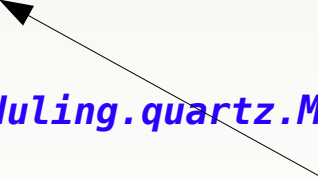
Job spesifik data map olarak tanımlanır, ancak Job sınıfı içerisine **setter metotlar** ile aktarılabilir

MethodInvokingJobDetailFactoryBean Kullanımı

```
public class ExampleBusinessObject {  
    // properties and collaborators  
  
    public void doIt() {  
        // do the actual work  
    }  
}
```

Sıradan bir POJO bean'inin
metodu scheduled bir task metodu
gibi çalıştırılabilir

```
<bean id="exampleBusinessObject" class="examples.ExampleBusinessObject"/>  
  
<bean id="jobDetail"  
class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">  
    <property name="targetObject" ref="exampleBusinessObject" />  
    <property name="targetMethod" value="doIt" />  
</bean>
```



Spring Quartz

Entegrasyonu – Trigger

- Daha sonra **SimpleTriggerFactoryBean** veya **CronTriggerFactoryBean** kullanarak Quartz Trigger nesneleri yaratılır

```
<bean id="simpleTrigger" class="org.springframework.scheduling.quartz.SimpleTriggerFactoryBean">
  <property name="jobDetail" ref="jobDetail" />
  <!-- 10 seconds -->
  <property name="startDelay" value="10000" />
  <!-- repeat every 50 seconds -->
  <property name="repeatInterval" value="50000" />
</bean>

<bean id="cronTrigger" class="org.springframework.scheduling.quartz.CronTriggerFactoryBean">
  <property name="jobDetail" ref="jobDetail" />
  <!-- run every morning at 6 AM -->
  <property name="cronExpression" value="0 0 6 * * ?" />
</bean>
```

Entegrasyonu – Scheduler

- Son adımda ise **SchedulerFactoryBean** ile daha önce tanımlanan trigger nesneleri eklenerek **Scheduler konfigürasyonu** tamamlanır

```
<bean class="org.springframework.scheduling.quartz.SchedulerFactoryBean">  
  <property name="triggers">  
    <list>  
      <ref bean="cronTrigger" />  
      <ref bean="simpleTrigger" />  
    </list>  
  </property>  
</bean>
```

İletişim

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

