Spring Framework
Module 9 – Task Execution and Scheduling

Vyacheslav Yakovenko
Last update: March, 2012

Содержание

- Spring task and scheduling API;
- Quartz;

Spring :: Планировщики задач

Назначение – выполнить задачу в определенный момент времени или через определенный промежуток времени.

- Существуют три основных API
  - Java's Timer - запускается через определенные промежутки времени ;
  - API Spring Framework
    - пакеты:
      - org.springframework.scheduling
      - org.springframework.core.task
  - Quartz scheduler:
    - http://www.quartz-scheduler.org/
    - также может быть запущен через определенные промежутки времени ;
    - может быть запущен в конкретный момент ;

Spring :: Task :: TaskExecutor

- В Spring Framework абстракция позволяющая организовать выполнение задач, базируется на интерфейсе **`TaskExecutor`**.
- **`TaskExecutor`**, предоставляет всего один метод `void execute(Runnable task)`, позволяющий передать «задачу», имплементирующую интерфейс `Runnable`.
- **`TaskExecutor`** — предоставляет функционал схожий с `java.util.concurrent.Executor`, единственная цель которого, абстрагироваться от Java 1.4

Spring :: Task :: TaskExecutor

Реализации `TaskExecutor`:

- `SimpleAsyncTaskExecutor` — не использует потоки повторно, каждый раз создавая новый поток;
- `SyncTaskExecutor` — вызовы происходят в текущем потоке, не поддерживает асинхронное выполнение задач;
- `ConcurrentTaskExecutor` — обертка на `java.util.concurrent.Executor` ;
- `SimpleThreadPoolTaskExecutor` — наследник Quartz's `SimpleThreadPool` ;
- `ThreadPoolTaskExecutor`;
- `TimerTaskExecutor`;
- `WorkManagerTaskExecutor` — использует CommonJ WorkManager ;

Spring :: Task :: TaskScheduler

Spring 3, дополнительно, вводит новый интерфейс TaskScheduler:

```
public interface TaskScheduler {
    ScheduledFuture schedule(Runnable task, Trigger trigger);
    ScheduledFuture schedule(Runnable task, Date startTime);
    ScheduledFuture scheduleAtFixedRate(Runnable task, Date
startTime, long period);
    ScheduledFuture scheduleAtFixedRate(Runnable task, long
period);
    ScheduledFuture scheduleWithFixedDelay(Runnable task, Date
startTime, long delay);
    ScheduledFuture scheduleWithFixedDelay(Runnable task, long
delay);
}
```

Spring :: Task :: TaskScheduler

LUXOFT TRAINING

- Основное достоинство этого интерфейса заключается в том, что он позволяет не привязвываться к конкртеной реализации планировщика.
- Это становится особенно важно, когда задача должна выполняться под управлением сервера приложений, где приложение не может самостоятельно порождать потоки.
- Для этих целей Spring предоставляет `TimerManagerTaskScheduler`, делегирующий выполнение задач в CommonJ TimerManager, получаемый обычно из JNDI.

Spring :: Task :: Trigger

- Интерфейс, также введенный в Spring 3.
- Основной идеей использования этого интерфейса, является то, что выполнение задачи должно базироваться на основе предыдущего выполнения, т.е. учитывая контекст.

```
public interface Trigger {
    Date nextExecutionTime(TriggerContext
  triggerContext);
}
public interface TriggerContext {
    Date lastScheduledExecutionTime();
    Date lastActualExecutionTime();
    Date lastCompletionTime();
```

Spring :: Task :: Trigger

Пример:

```
scheduler.schedule(task,
  new CronTrigger("* 15 9-17 * * MON-FRI"));
```

- Выполняется:
  - Каджые 15 минут;
  - С 9 до 17 ;
  - С Пн по Пт ;

Spring :: Task :: Namespace

- Spirng 3 вводит дополнительный namespace — task, позволяющий проинициализировать соответствующие бины в контексте приложения:

```
<task:scheduler id="scheduler" pool-size="10"/>
<task:executor id="executor" pool-size="10"/>
```

- А также включить автообнаружение компонент, проаннотированных с помощью `@Scheduled`:

```
<task:annotation-driven … />
```

Spring :: Task :: Пример

Описание задач в контексте приложения:

```
<task:scheduled-tasks scheduler="myScheduler">
    <task:scheduled ref="someObject"
method="someMethod" fixed-rate="5000"/>
    <task:scheduled ref="anotherObject"
method="anotherMethod" cron="*/5 * * * * MON-FRI"/>
</task:scheduled-tasks>

<task:scheduler id="myScheduler" pool-size="10"/>
```

Spring :: Task :: Пример

Описание задач с использованием аннотации:

`@Scheduled`:

```
@Scheduled(fixedDelay=5000)
public void doSomething() {
    // something that should execute periodically
}

@Scheduled(fixedDelay=5000)
public void doSomething() {
    // something that should execute periodically
}
```

** - методы должны быть обявлены в `@Service` - компоненте*

Spring :: Task :: Пример

Описание задач с использованием аннотации `@Scheduled`:

```
@Scheduled(cron="*/5 * * * * MON-FRI")
public void doSomething() {
    // something that should execute on weekdays only
}
```

Spring :: Task :: Quartz

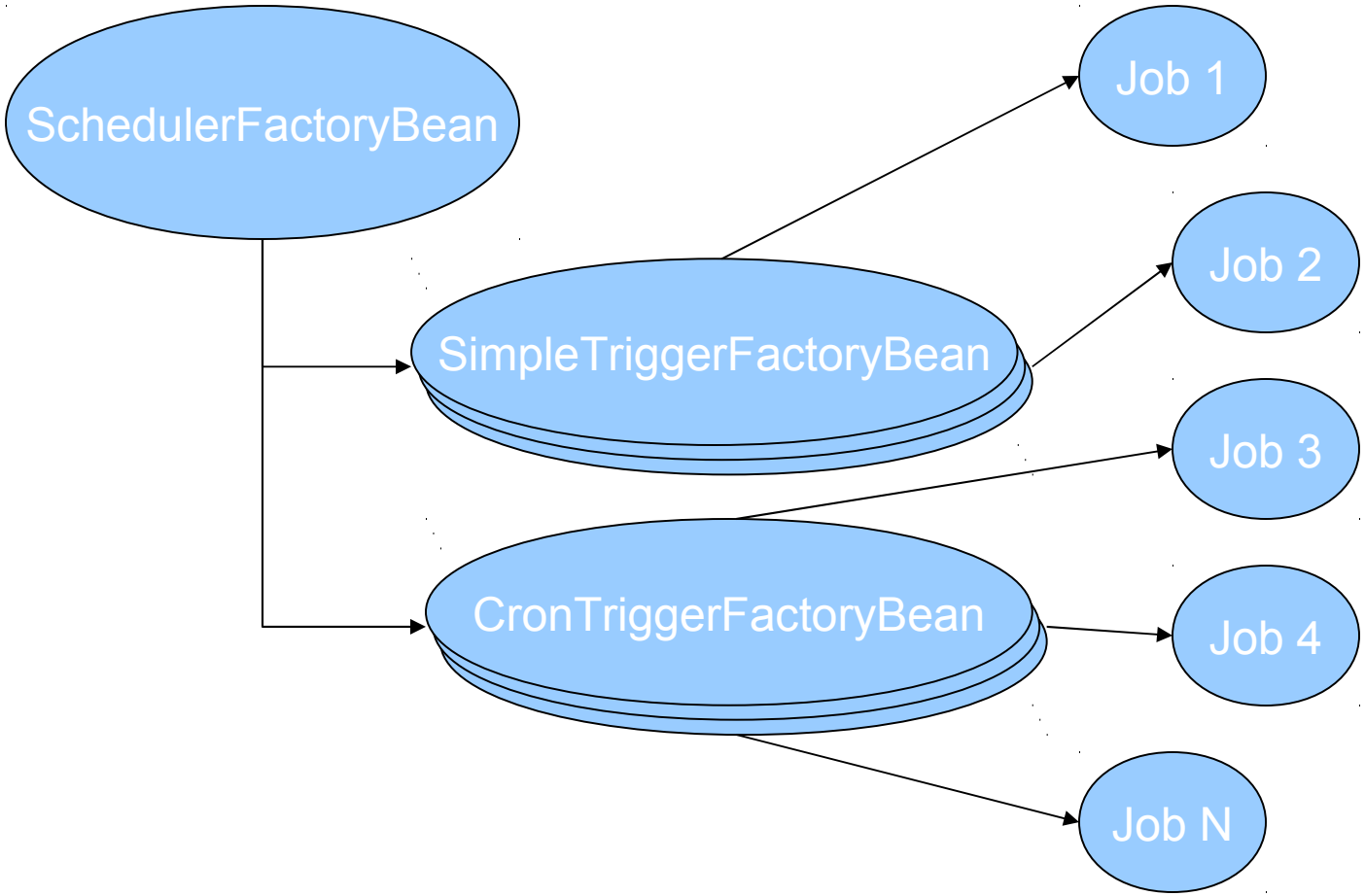
Одной из распространненых внешних библиотек для работы с задачами является Quartz, в которой для описания задачи используется интерфейс `JobDetail`

Spring 3.1, предоставляет для этих целей фабрику `JobDetailFactoryBean`, работающую корректно, как с 1-ой, так и 2-й версией Quartz:

```
<bean id="reportJob"
    class="org.springframework.scheduling.quartz.JobDetailFactoryBean">
    <property name="jobClass" value="lab.schedule.PrintingJob" />
</bean>
```

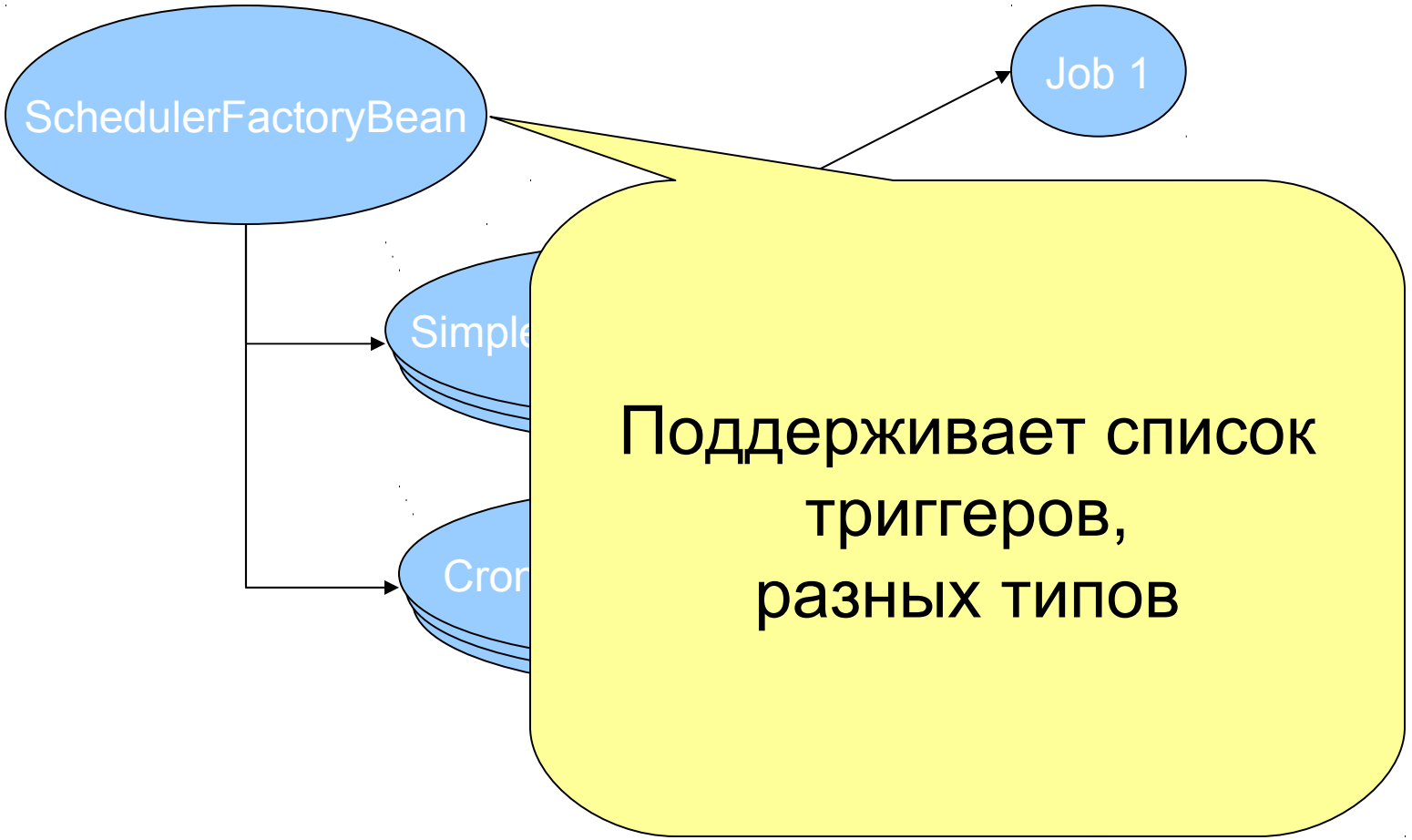
Spring :: Task :: Quartz

Схема взаимодействия бинов в общем случае:



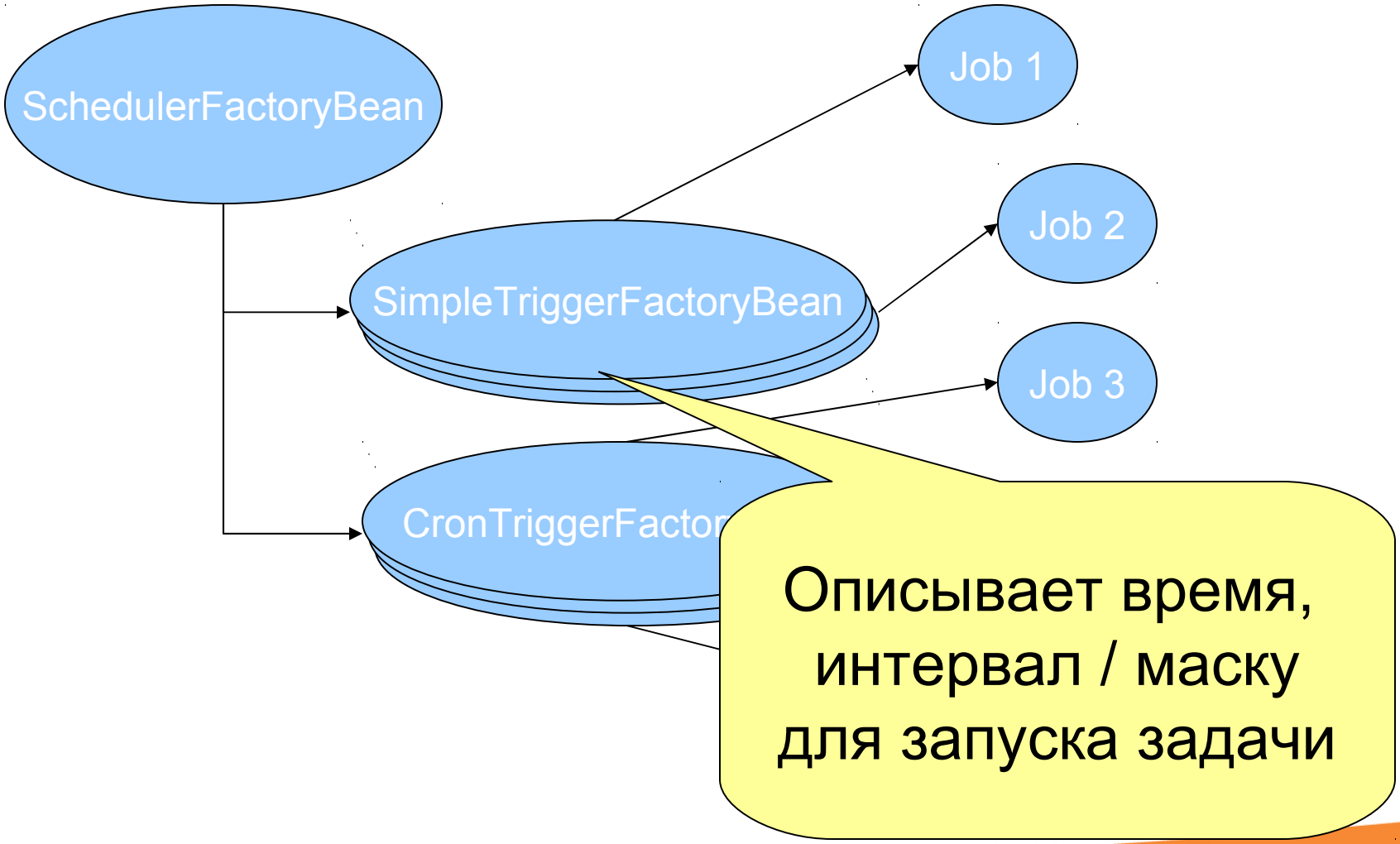
Spring :: Task :: Quartz

Схема взаимодействия бинов:



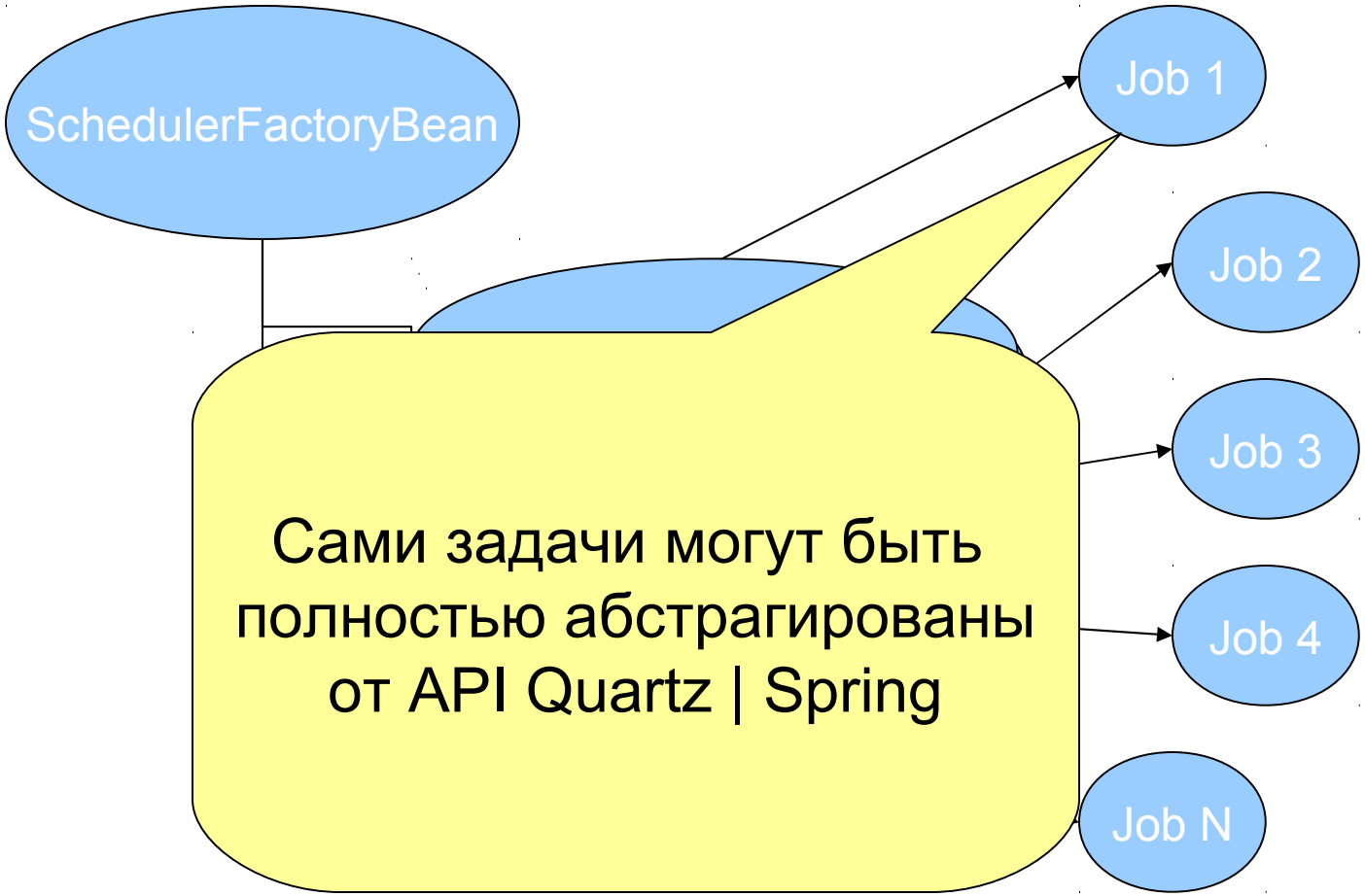
Spring :: Task :: Quartz

Схема взаимодействия бинов:



Spring :: Task :: Quartz

Схема взаимодействия бинов:



Spring :: Task :: Пример

LUXOFT TRAINING

Одной

```
<bean id="schedulerFactoryBean"
    class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
        <property name="triggers">
            <list>
                <ref bean="reportTrigger" />
            </list>
        </property>
</bean>

<bean id="reportTrigger"
class="org.springframework.scheduling.quartz.SimpleTriggerFactoryBean">
    <property name="jobDetail" ref="reportJob" />
    <property name="repeatInterval" value="1000" />
    <property name="startDelay" value="5000" />
</bean>
```

Упражнения

- №: 10 : Использование планирования задач.
  - 30 мин – самостоятельная работа;

LUXOFT TRAINING

Вопросы!?