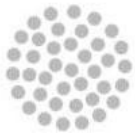
ESCUELA:
TECNOLOGÍA

SPRING BATCH

11 de Octubre de 2018



indra

ÍNDICE

- NOMBRE APELLIDO PROFESOR
  José Mª Díaz Charcán

- VER PERFIL COMPLETO:

  linkedin.com/company/icono-training-consulting

- CONTACTO

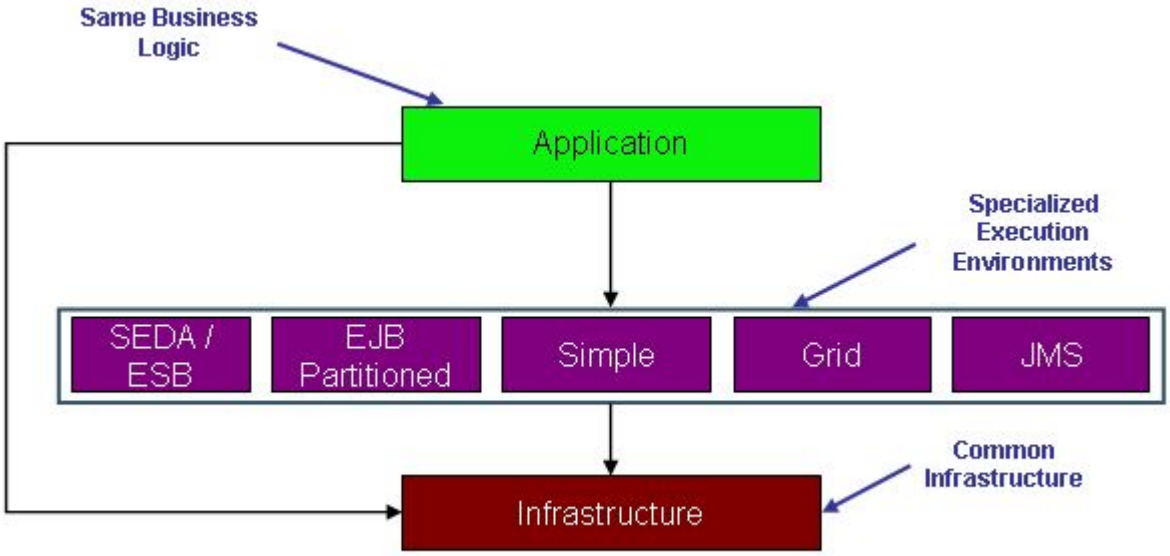
  training@iconotc.com

Características

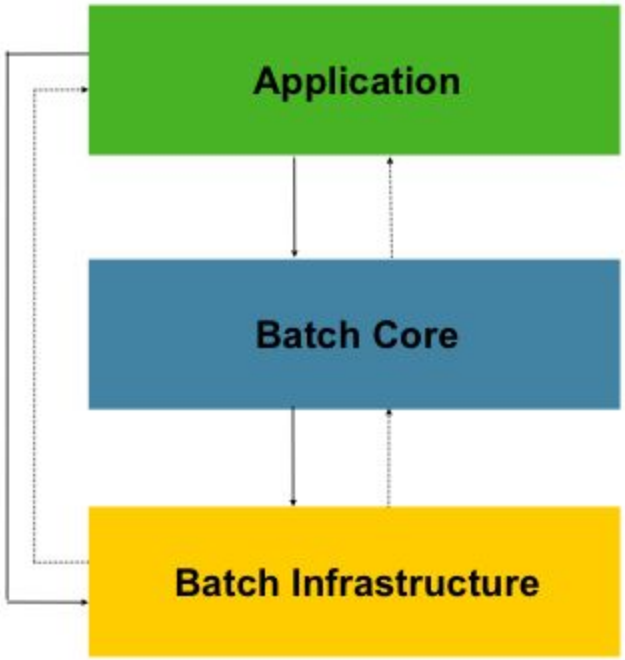
Spring Batch es un framework para el procesamiento por lotes (o ejecuciones "batch") que proporciona:

- un marco general para la creación de programas batch.
- almacenamiento de la información de ejecuciones.
- utilización de conceptos conocidos para el procesamiento batch (Job, Step, JobInstance, JobExecution, JobRepository, etc).
- utilidades para realizar acciones comunes en procesamientos batch (lectura/escritura de archivos, acceso a base de datos, etc).

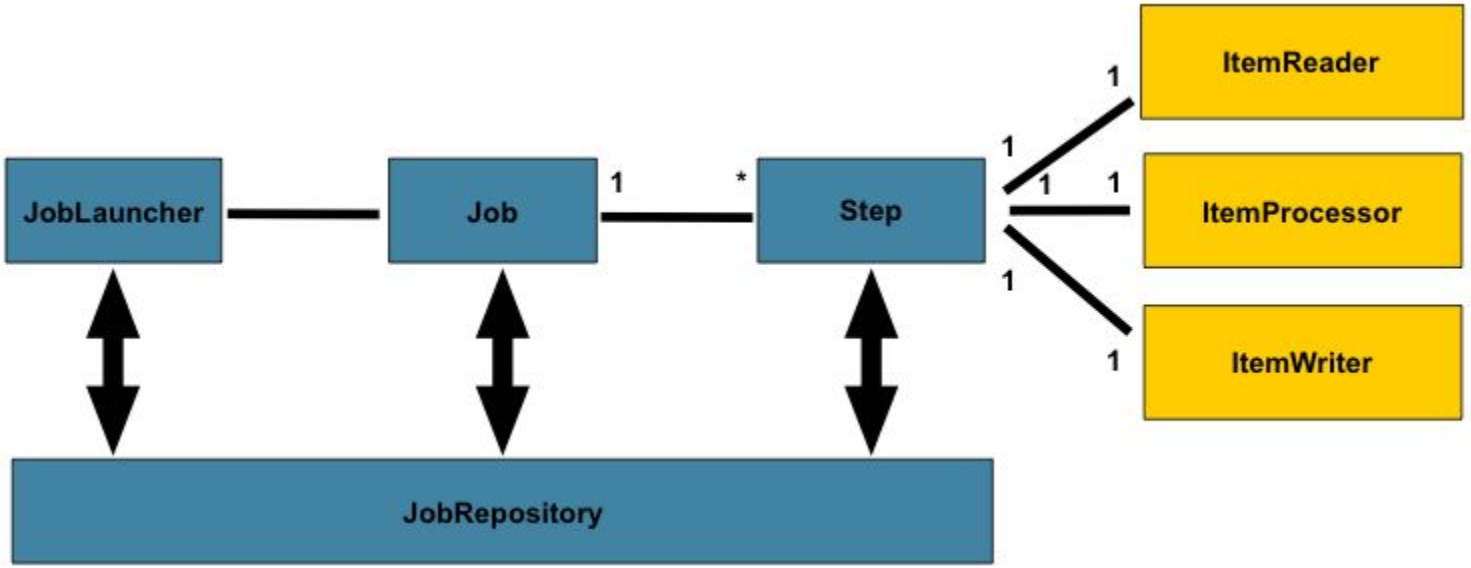
Arquitectura I



Arquitectura II



Arquitectura III



Conceptos básicos I

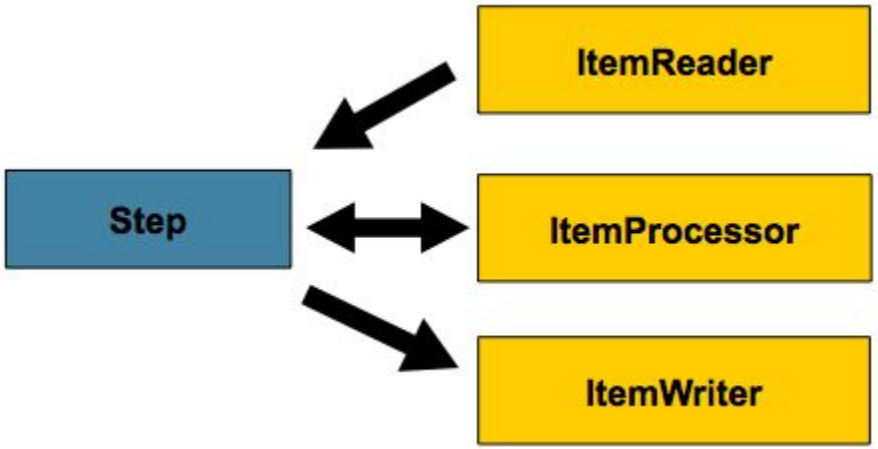
- **JobRepository**: es un contenedor donde se almacenan las ejecuciones de los Job, usualmente en tablas propias de Spring Batch desplegadas en una base de datos.
- **JobLauncher**: representa un mecanismo para lanzar un Job junto con un conjunto de JobParameters.
- **Job**: representa un trabajo batch a ejecutar. Está compuesto de uno o más Step (pasos), los cuales se ejecutan secuencialmente por defecto.

Conceptos básicos II

- **Step**: es un objeto de dominio que encapsula una ejecución de una fase de un Job. Un step puede leer un archivo, escribir en una base de datos, realizar una transformación, etc.
- **ItemReader**: mecanismo de obtención de datos de entrada en un Step.
- **ItemWriter**: mecanismo de generación de datos de salida, uno o muchos, en un Step.
- **ItemProcessor**: mecanismo para el procesamiento de datos según determinadas reglas de negocio. Transformaciones.

Artefactos I

Procesamiento de datos en un paso

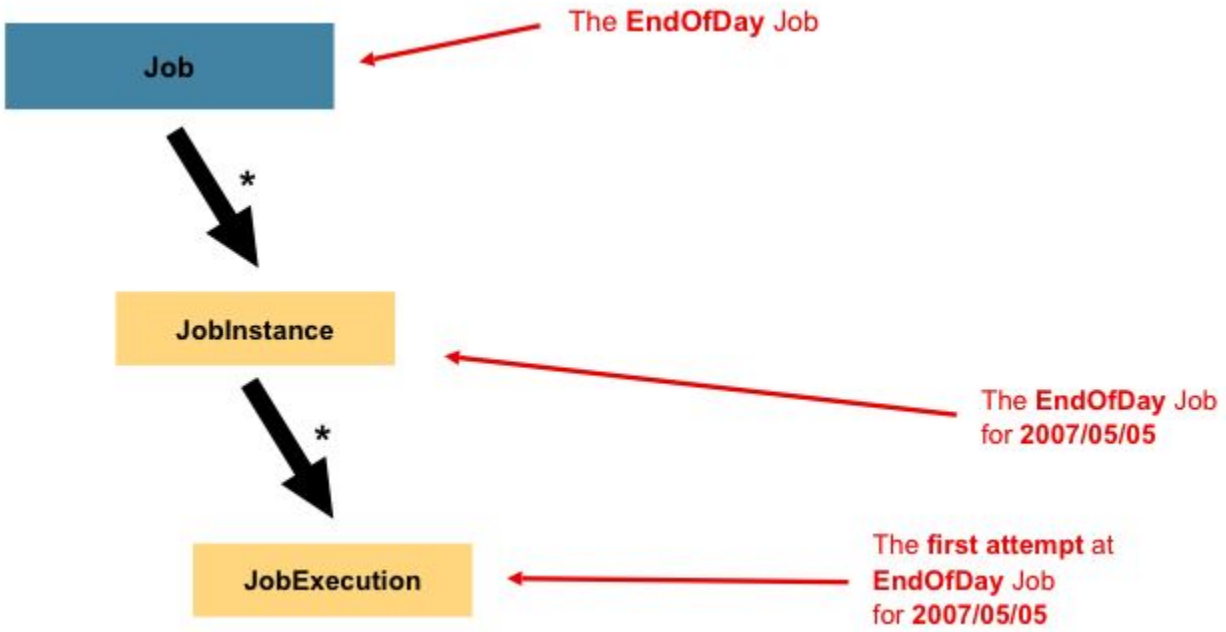


Conceptos básicos III

- **Espacio de nombres batch**: vocabulario xml especializado para declarar elementos propios de Spring Batch en una configuración.
- **JobInstance**: ejecución "lógica" de un Job.
- **JobParameters**: metadatos empleados para diferenciar una JobInstace de otra.
- **JobExecution**: el intento de ejecución de un Job. Contiene propiedades que Spring Batch hace persistentes. Puede haber muchas ejecuciones, pero la JobInstance a la que pertenecen es la misma

Artefactos II

Modelo de ejecución de un "Job"

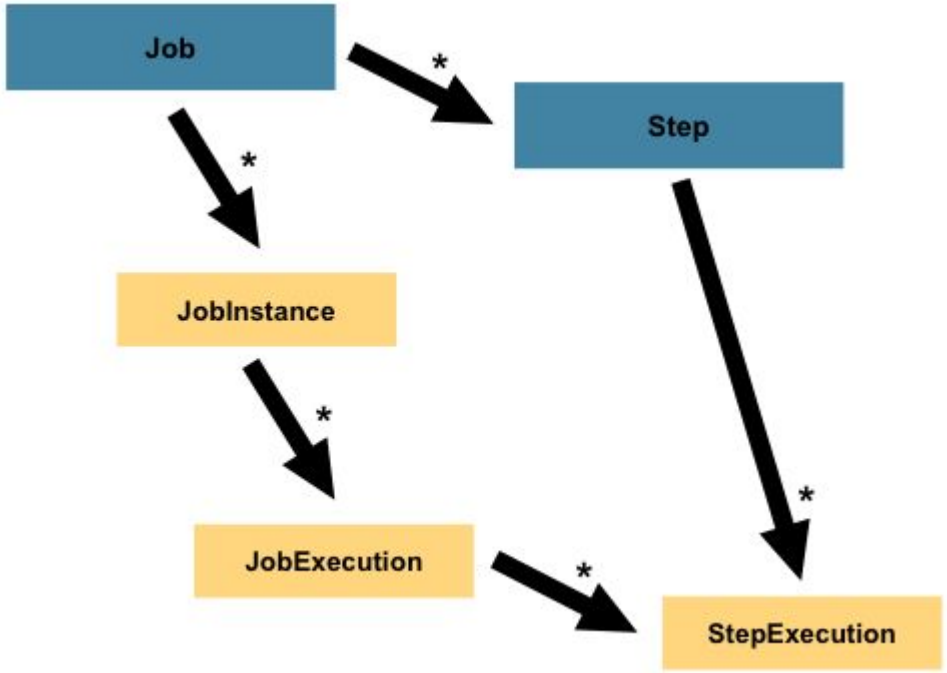


Conceptos básicos IV

- **StepExecution**: el intento de ejecución de un Step.
- **Execution Context**: colección de metadatos en formato clave-valor para hacer persistente información en el ámbito de una StepExecution y, por tanto, de una JobExecution. Uso típico: restablecimiento de un Job fallido.

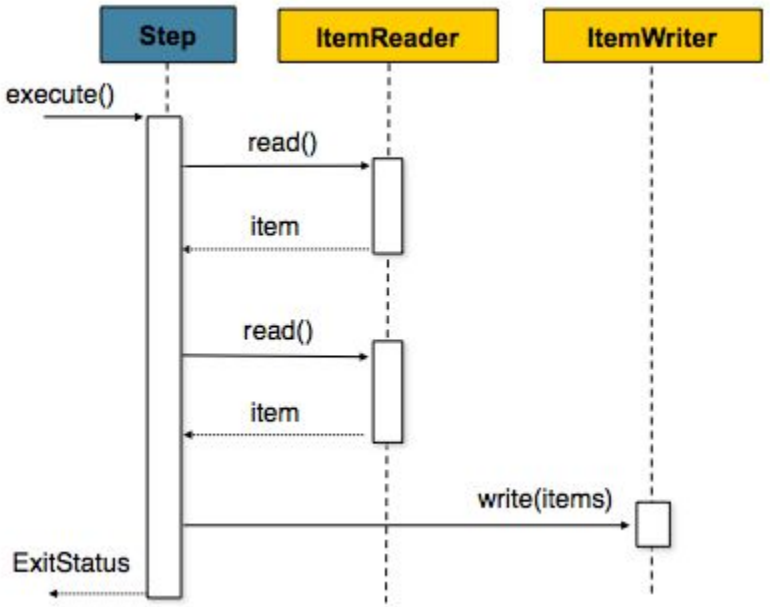
Artefactos III

Modelo de ejecución de un "step"



Artefactos IV

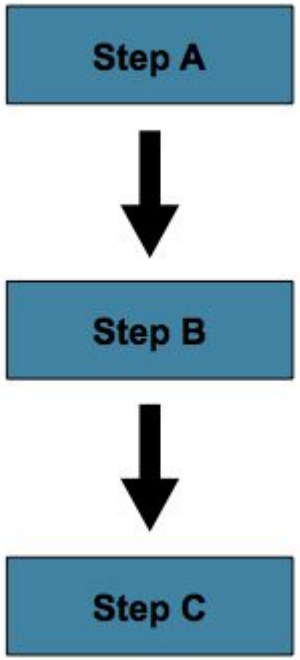
Ejecución de un paso



Artefactos V

Ejecución secuencial

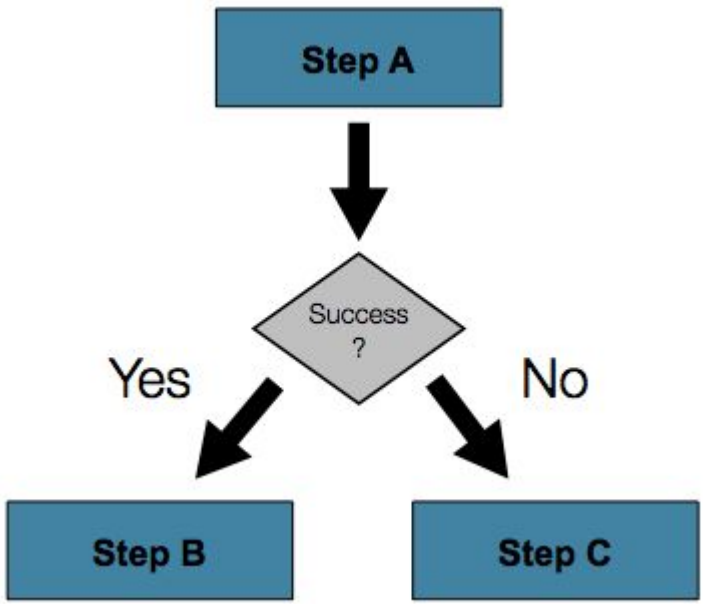
```
<job id="uno">
   <step id="A" next="B"/>
   <step id="B" next="C"/>
   <step id="C"/>
</job>
```



Artefactos VI

Ejecución condicional

```
<job id="job">
   <step id="stepA">
      <next on="FAILED" to="stepB" />
      <next on="*" to="stepC" />
   </step>
   <step id="stepB" next="stepC" />
   <step id="stepC" />
</job>
```

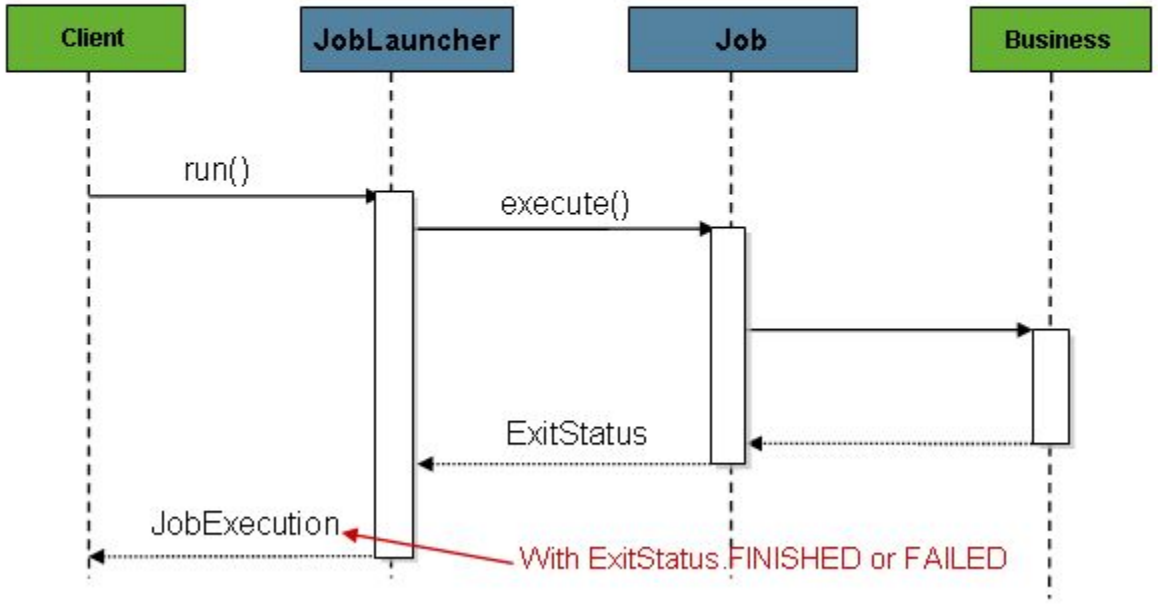


Configuración I

- Para configurar un trabajo: nombre, referencia a un repositorio y lista de pasos
  - Ver ejemplos SpringBatch0000 y SpringBatch0001
- Para configurar un paso: un ItemReader, un ItemWriter y, a menudo, un ItemProcessor
  - Ver ejemplos SpringBatch0002 y SpringBatch0003

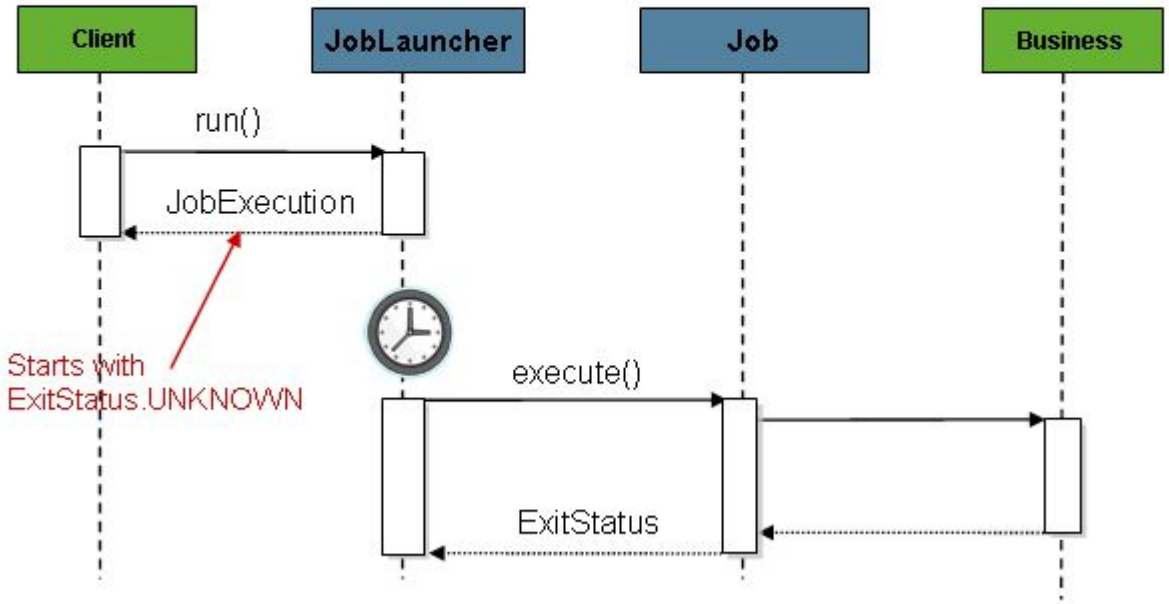
Lanzamiento de un Job I

Lanzamiento de un "job" desde un programa de escritorio o un "scheduler" especializado



Lanzamiento de un Job II

Lanzamiento de un "job" bajo HTTP, por ejemplo, empleando un "TaskExecutor"



Lanzamiento de un Job III

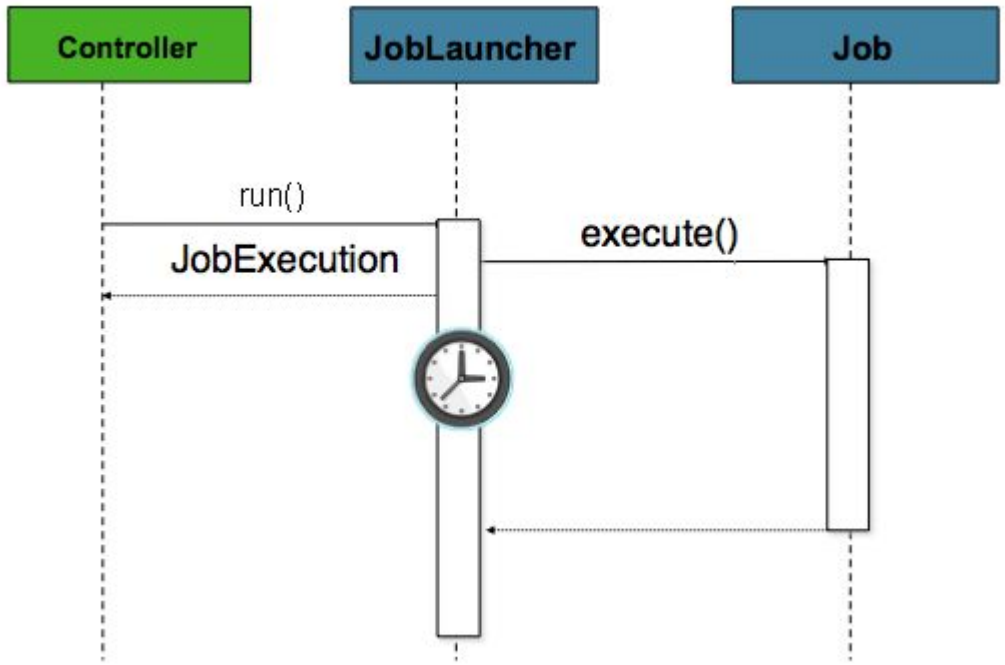
Ejecución desde un controlador:

```
@Controller
public class JobLauncherController {

                    @Autowired
        JobLauncher jobLauncher;

                    @Autowired
                Job job;


@RequestMapping("/jobLauncher.html")
        public void handle() throws Exception{
                jobLauncher.run(job, new JobParameters());
                                    }
}
```



Configuración II

- Procesamiento paralelo y escalabilidad
- Procesos cíclicos
- Reintento automático de operaciones fallidas