# 1 Introducción

## 1.1 ¿Qué es un Batch? ¿Cuál es su utilidad?

El procesamiento por lotes (en adelante Batch) es un modo de procesamiento el cual implica la ejecución de un serie de tareas complejas sin intervención del usuario.

Muchas aplicaciones empresariales requieren ejecutar procesos sobre grandes cantidades de datos, estas aplicaciones hacen uso de Batch.

## 1.2 ¿Qué es Spring Batch?

Es un framework ligero que facilita el desarrollo de aplicaciones Batch mediante el uso de Spring. Spring Batch no es un framework de planificación (para ello está Quartz, Tivoli, ControlM...). Además provee de algunas funcionalidades más:

· Trazabilidad (Logging & Tracing)

· Transaccionalidad

· Tareas de procesado estadístico

## 1.3 Características de Spring Batch

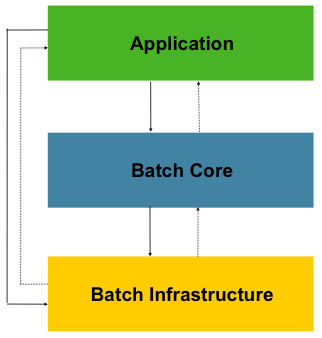
Las aplicaciones construidas con Spring Batch son:

* Flexibles: Simplemente cambiando un YML se puede alterar el orden de proceso de la aplicación.
* Mantenibles: En Spring Batch una Tarea ejecuta Pasos, estos Pasos pueden desacoplados, testeados y actualizados sin afectar a los otros Pasos. (en adelante, Tarea será Job y Paso será Step)
* Escalables: Empleando técnicas de particionado, Spring Batch permite crecer a la aplicación horizontalmente, permitiendo ejecutar los Steps de un Job en paralelo o paralelizar un hilo de ejecución.
* Robustas: En caso de fallo, se puede relanzar un Job justo donde se paró, desacoplando los Steps.
* Versátiles: Provee de un gran conjunto de lectores (Reader) y escribidores (Writer) de datos (XML, fichero plano, CSV, RMDB, ORM, NoSQL, etc). También permite lanzar los Jobs a través de múltiples canales (web, consola, programaciones, etc)
* Además:
  + Intento automático después de un fallo.
  + Traza de estado y estadísticas de duración durante y después de la ejecución del Batch.
  + Concurrencia de Jobs

# 2 Diseño de Spring Batch

## 2.1 Arquitectura de Spring Batch

El siguiente diagrama representa la arquitectura de Spring Batch a alto nivel. Se compone de tres capas: Application, Batch Core, Batch Infrastructure



* **Application**: Contiene todos Jobs y el código escrito por el desarrollador.
* **Batch Core**: contiene las clases que en runtime son necesarias para iniciar y controlar un Job. Incluye implementaciones para JobLauncher, Job y Step.
* **Batch Infr**: Contiene lectores, escritores y servicios comunes, que son utilizados tanto por el desarrollador de la aplicación, como por el propio framework.

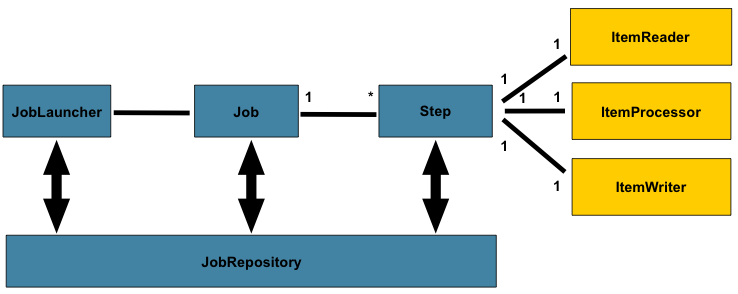
.

Tanto la Application como Batch Core están construidos sobre Batch Infrastructure común.

## 2.2 Componentes de Spring Batch

La siguiente imagen muestra los componentes de Spring Batch y como están conectados:

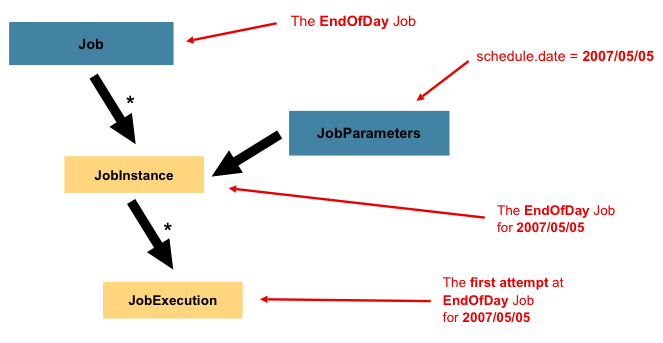
Job, es una versión simplificada de la arquitectura Batch de referencia:



El diagrama muestra los conceptos claves de Spring Batch. Un Job tiene muchos Steps, cada Step tiene un ItemReader, un ItemProcessor y un ItemWriter. Un Job necesita ser iniciado (mediante JobLaucher) y se necesita almacenar la información de ejecución (usando JobRepository).

### 2.2.1 Jobs

Job está en el nivel más alto de la jerarquía, como muestra el siguiente diagrama:



### 2.2.1.1 Job

Un Job es una entidad que encapsula un procesamiento por lotes completo. Además es un contenedor de instancias de Step. En él se combinan múltiples Steps que se enlazan mediante un flujo y permite una establecer una configuración global para los Steps.

Los Jobs se configuran a través de Java-config (o a través de XML), y permite establecer:

* El nombre del Job
* La definición y el orden de los Steps
* Si es reiniciable o no

Spring Batch provee de una implementación básica de un Job, la clase SimpleJob

|  |
| --- |
| @Bean  public Job **endOfDayJob**() {  return this.jobBuilderFactory.get("**EndOfDayJob**")  .start(readCsv())  .next(transformToXML())  .next(writeXML())  .end()  .build();  } |

### 2.2.1.2 JobInstance

Un JobInstance hace referencia al concepto de ejecución lógica de un Job, puede tener más de una ejecución (JobExecution) y se pueden identificar mediante JobParameters.

Por ejemplo, un Job que se tiene que ejecutar todos los días al final del día (“EndOfDayJob”). Existirá un JobInstance cada vez se ejecute, es decir, habrá un JobInstance para el día 01-Enero, otro para el 02-Enero, etc.

Un nuevo JobInstance significa 'comenzar desde el principio', y usar un JobInstance existente generalmente significa 'comenzar desde donde lo dejaste'.

|  |
| --- |
|  |