A lightweight, comprehensive batch framework designed to enable the development of robust batch applications vital for the daily operations of enterprise systems.

Features

- Chunk based processing
- Declarative I/O
- Start/Stop/Restart
- Retry/Skip

CIT Batch provides reusable functions that are essential in processing large volumes of records, including logging/tracing, job processing statistics, and job restart. Simple as well as complex, high-volume batch jobs can leverage the framework in a highly scalable manner to process significant volumes of information.

As shown in our batch processing example, a batch process is typically encapsulated by a Job consisting of multiple Steps. Each Step typically has a single ItemReader, ItemProcessor, and ItemWriter. A Job is executed by a JobLauncher, and metadata about configured and executed jobs is stored in a JobRepository.

Each Job may be associated with multiple JobInstances, each of which is defined uniquely by its particular JobParameters that are used to start a batch job. Each run of a JobInstance is referred to as a JobExecution. Each JobExecution typically tracks what happened during a run, such as current and exit statuses, start and end times, etc.

A Step is an independent, specific phase of a batch Job, such that every Job is composed of one or more Steps. Similar to a Job, a Step has an individual StepExecution that represents a single attempt to execute a Step. StepExecution stores the information about current and exit statuses, start and end times, and so on, as well as references to its corresponding Step and JobExecution instances.

An ExecutionContext is a set of key-value pairs containing information that is scoped to either StepExecution or JobExecution. CIT Batch persists the ExecutionContext, which helps in cases where you want to restart a batch run (e.g., when a fatal error has occurred, etc.). All that is needed is to put any object to be shared between steps into the context and the framework will take care of the rest. After restart, the values from the prior ExecutionContext are restored from the database and applied.

JobRepository is the mechanism in CIT Batch that makes all this persistence possible. It provides CRUD operations for JobLauncher, Job, and Step instantiations. Once a Job is launched, a JobExecution is obtained from the repository and, during the course of execution, StepExecution and JobExecution instances are persisted to the repository.