# Appendix A: Glossary

## A.1. Spring Batch Glossary

*Batch*

An accumulation of business transactions over time.

*Batch Application Style*

Term used to designate batch as an application style in its own right, similar to online, Web, or SOA. It has standard elements of input, validation, transformation of information to business model, business processing, and output. In addition, it requires monitoring at a macro level.

*Batch Processing*

The handling of a batch of many business transactions that have accumulated over a period of time (such as an hour, a day, a week, a month, or a year). It is the application of a process or set of processes to many data entities or objects in a repetitive and predictable fashion with either no manual element or a separate manual element for error processing.

*Batch Window*

The time frame within which a batch job must complete. This can be constrained by other systems coming online, other dependent jobs needing to execute, or other factors specific to the batch environment.

*Step*

The main batch task or unit of work. It initializes the business logic and controls the transaction environment, based on commit interval setting and other factors.

*Tasklet*

A component created by an application developer to process the business logic for a Step.

*Batch Job Type*

Job types describe application of jobs for particular types of processing. Common areas are interface processing (typically flat files), forms processing (either for online PDF generation or print formats), and report processing.

*Driving Query*

A driving query identifies the set of work for a job to do. The job then breaks that work into individual units of work. For instance, a driving query might be to identify all financial transactions that have a status of "pending transmission" and send them to a partner system. The driving query returns a set of record IDs to process. Each record ID then becomes a unit of work. A driving query may involve a join (if the criteria for selection falls across two or more tables) or it may work with a single table.

*Item*

An item represents the smallest amount of complete data for processing. In the simplest terms, this might be a line in a file, a row in a database table, or a particular element in an XML file.

*Logicial Unit of Work (LUW)*

A batch job iterates through a driving query (or other input source, such as a file) to perform the set of work that the job must accomplish. Each iteration of work performed is a unit of work.

*Commit Interval*

A set of LUWs processed within a single transaction.

*Partitioning*

Splitting a job into multiple threads where each thread is responsible for a subset of the overall data to be processed. The threads of execution may be within the same JVM or they may span JVMs in a clustered environment that supports workload balancing.

*Staging Table*

A table that holds temporary data while it is being processed.

*Restartable*

A job that can be executed again and assumes the same identity as when run initially. In other words, it is has the same job instance ID.

*Rerunnable*

A job that is restartable and manages its own state in terms of the previous run's record processing. An example of a rerunnable step is one based on a driving query. If the driving query can be formed so that it limits the processed rows when the job is restarted, then it is re-runnable. This is managed by the application logic. Often, a condition is added to the `where` statement to limit the rows returned by the driving query with logic resembling "and processedFlag!= true".

*Repeat*

One of the most basic units of batch processing, it defines by repeatability calling a portion of code until it is finished and while there is no error. Typically, a batch process would be repeatable as long as there is input.

*Retry*

Simplifies the execution of operations with retry semantics most frequently associated with handling transactional output exceptions. Retry is slightly different from repeat, rather than continually calling a block of code, retry is stateful and continually calls the same block of code with the same input, until it either succeeds or some type of retry limit has been exceeded. It is only generally useful when a subsequent invocation of the operation might succeed because something in the environment has improved.

*Recover*

Recover operations handle an exception in such a way that a repeat process is able to continue.

*Skip*

Skip is a recovery strategy often used on file input sources as the strategy for ignoring bad input records that failed validation.