# Study Spring Batch Support

The Goal of this study is to gather the information about Spring Batch support in JEE analyzer, the entry and exit points of a spring batch application and the links between spring batch objects

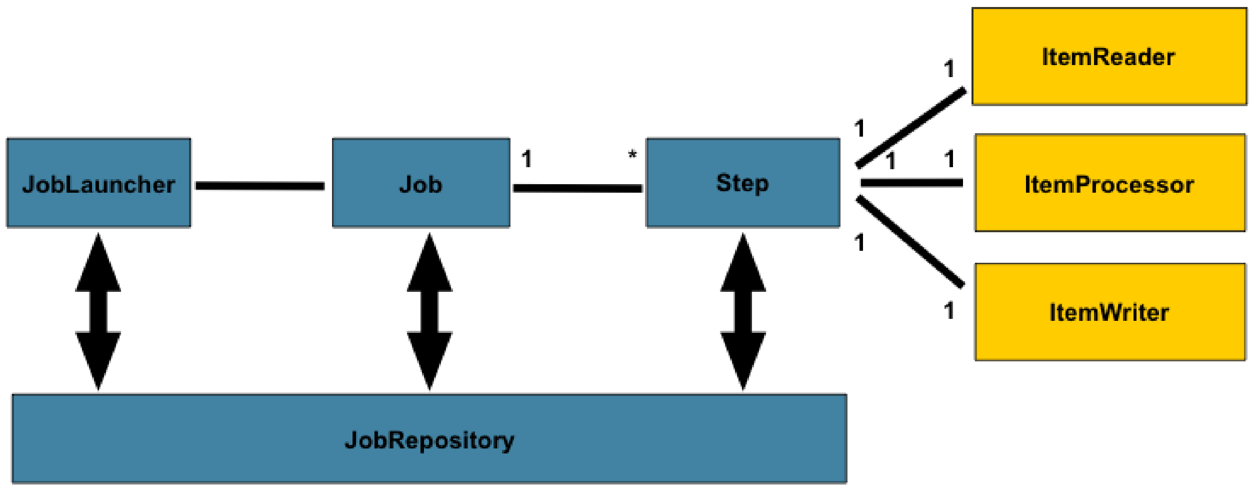
## Introduction

Spring Batch is an open source framework for batch processing – execution of a series of jobs. Spring Batch provides classes and APIs to read/write resources, transaction management, job processing statistics, job restart and partitioning techniques to process high-volume of data.

A batch application can be divided in three main parts:

- Reading the data (from a database, file system, etc.
- Processing the data (filtering, grouping, calculating, validating…)
- Writing the data (to a database, reporting, distributing…)

A typical Spring Batch Application flows:-



## Item Reader

Readers are abstractions responsible of the data retrieval. They provide batch processing applications with the needed input data

There are different implementation of Reader :

- FlatFileItemReader
- HibernateCursorItemReader
- ItemReaderAdapter
- JdbcCursorItemReader
- JdbcPagingItemReader

## Item Writer

Writers are abstractions responsible of writing the data to the desired output database or system.

- CompositeItemWriter
- FlatFileItemWriter
- ItemWriterAdapter

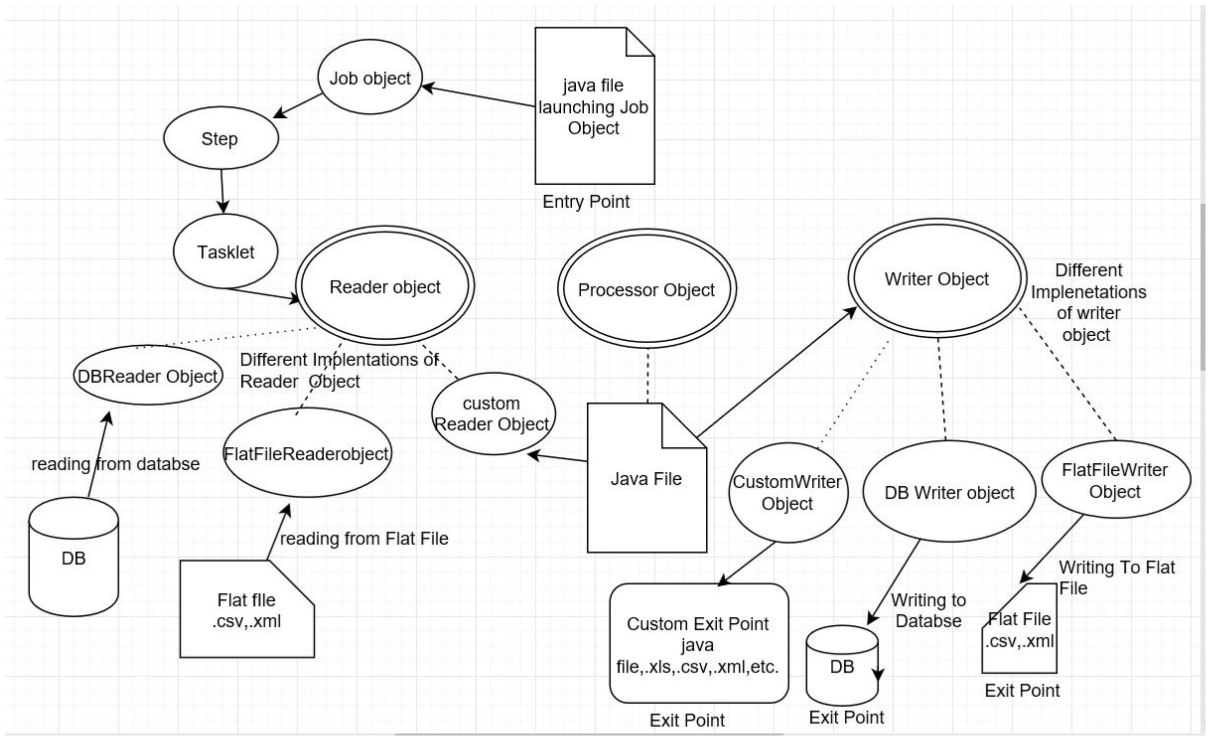
## Item Processors

Processors are in charge of modifying the data records converting it from the input format to the output desired one

## Spring Batch Objects To be identified in JEE Analyzer

This Section includes the requirements gathered from consultants for the type of spring batch objects to be identified.

This is the flow chart with entry and exit points



### Entry Point

The Entry point for a spring Batch application in TCC will be from Java File which will read the beans(Job,Job launcher) configured in xml file start the spring batch application.

### Exit Point

There can be different Exit points for Spring batch application to be identified in TCC depending on the implementation of writer object so if its a FlatFileWriter ,the exit point will be flat file(xml,csv etc),

If the implementation is DbWriter then the Exit point will be of Database, and if its a custom implementation then Exit point can be anything depending on the implementation

provided in CustomWriter,it can be xml,csv,java file etc.

## Sample Spring Batch Application

This is a sample spring batch application that is reading data from csv file and writing to database.

A Job is configured in a series of steps

Job->Step->tasklet

A step is configured with reader and processor and a writer,processor is optional.

**Job.XML file Configuration**

**Job.xml**

```
<import resource="../config/context.xml" />
<import resource="../config/database.xml" />

<batch:job id="helloWorldJob">    //helloWorldJob Job consisting of
steps,steps having tasklet
   <batch:step id="step1">
  <batch:tasklet>
   <batch:chunk reader="cvsFileItemReader" writer="xmlItemWriter"  //reader
writer and processor dependency
                              processor="itemProcessor"
commit-interval="10">
   </batch:chunk>
  </batch:tasklet>
   </batch:step>
 </batch:job>
```

These beans cvsFileItemReader,xmlItemWriter and itemProcessor are configured in xml file

Then JobRepsoitroy bean is configured

**context.xml**

```
<bean id="jobLauncher"
  class="org.springframework.batch.core.launch.support.SimpleJobLauncher">
//joblauncher having dependency of JobRepository
  <property name="jobRepository" ref="jobRepository" />
</bean>
<bean id="jobRepository"

class="org.springframework.batch.core.repository.support.JobRepositoryFact
oryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="transactionManager" ref="transactionManager" />
  <property name="databaseType" value="mysql" />
</bean>
<bean id="transactionManager"

class="org.springframework.batch.support.transaction.ResourcelessTransacti
onManager" />
```

This JobRepository bean is dependent on datasource so  this property will be configured in a separate xml,which will be having configuration of database connection

A reader can have different implementations:-

- A FlatFileReader that is reading from flat files xml,csv file and writing to database or writing to a flat file or wirting to java file
- A DbReader that is reading data from Databse and writing to flat file or database or a custom writer.

- A customReader that is reading from custom files and writing it to database or custom writer or flat file.

A Writer can have different implementations:-

- A FlatFilewriter that is writing to flat files xml,csv file.
- A DbReader that is  writing to flat file or database or a custom writer.
- A customReader that is  writing it to database or custom writer or flat file.

**This Job is called in the following way**

```
public static void main(String [] args){

String[] springConfig  =
  {
   "spring/batch/jobs/job.xml"   //Load the Job.xml file,this Job.xml has
dependency on JobLauncher
           // which will load the Repository
  };

 ApplicationContext context =
   new ClassPathXmlApplicationContext(springConfig);

 JobLauncher jobLauncher = (JobLauncher) context.getBean("jobLauncher");
//get the JobLauncher object
 Job job = (Job) context.getBean("helloWorldJob");   //get the Job Object

try {
  JobExecution execution = jobLauncher.run(job, new
JobParameters());//start the job by calling the run method.
  System.out.println("Exit Status : " + execution.getStatus());

 } catch (Exception e) {
  e.printStackTrace();
 }
}
```