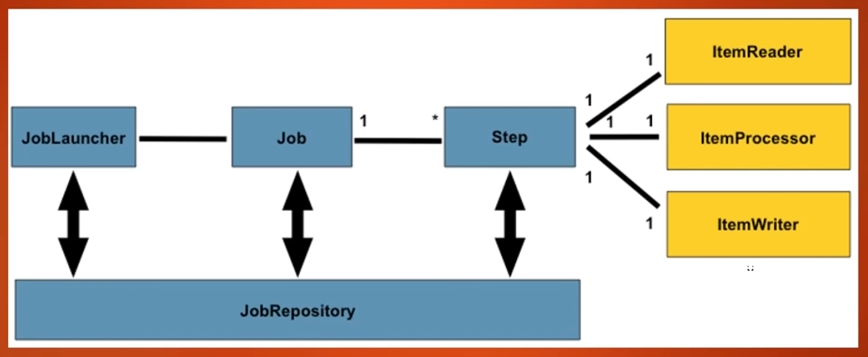
**Spring Batch:**



1. **Job:** It denotes the actual spring batch job which is created for doing a specific task/job. Each Job is configured with one or more steps.
2. **Step:** It represents piece of core put together to handle the particular job.
   1. **ItemReader:** It reads the input data and share each item one at a time to the processor/writer for further processing.
   2. **ItemProcessor:** It processes the incoming data from the itemReader and transforms to the form that itemWriter would understand.
   3. **ItemWriter:** Its writes the data of an item to the output one at a time.
3. **JobLuncher:** Lunches the job.
4. **JobRepository:** It provides the CRUD operations for jobLuncher and other batch core mechanisms to persistent their stats during course of execution.

Example: it will store the details like last success job, last failure job, job parameters etc…

**Spring AOP:**

Dependency: AOP (AOP + AspectJ)

1. AOP is the best approach for the cross cutting concerns.
2. Advice: Everything inside the method is called advice

|  |
| --- |
| @Before("execution(\* com.rky.aopdemo.business.\*.\*(..))")  public void before(JoinPoint joinPoint) {  //Inside this method everything is called ADVICE.  logger.info("Checking for the user access...");  logger.info("User accessed : {}", joinPoint);  } |

1. **@Before**: Executes before every methods which is mentioned in the **pointcut** expression

Ex: @Before(“execution(\* PACKAGE.\*.\*(..)”)

execution (\* PACKAGE.\*.\*(..))

**\***: Any return type

**PACKAGE**: package name

**\***: any class

**\***: any method

**(..)**: any number of arguments

1. **@After** :
2. **@AfterReturning**: After returning from the executed method
3. **@AfterThrowing** : After throw an error from the executed method
4. **@Around** :
5. The combination of **POINTCUT** and **ADVICE** is called **Aspect**.
6. Implementing the AOP around the methods is called **Weaving** and the framework is called the **weaver**.