Spring Batch Framework

Spring batch 2.1

본 문서는 전자정부프레임워크 실행환경 (배치처리) 교재를 바탕으로, Spring batch 2.1 기반의 배치 처리 어플리케이션 구현을 위한 참고용으로 작성되었습니다.

대덕 인재 개발원 : 최희연

❑ Batch Job 이란?

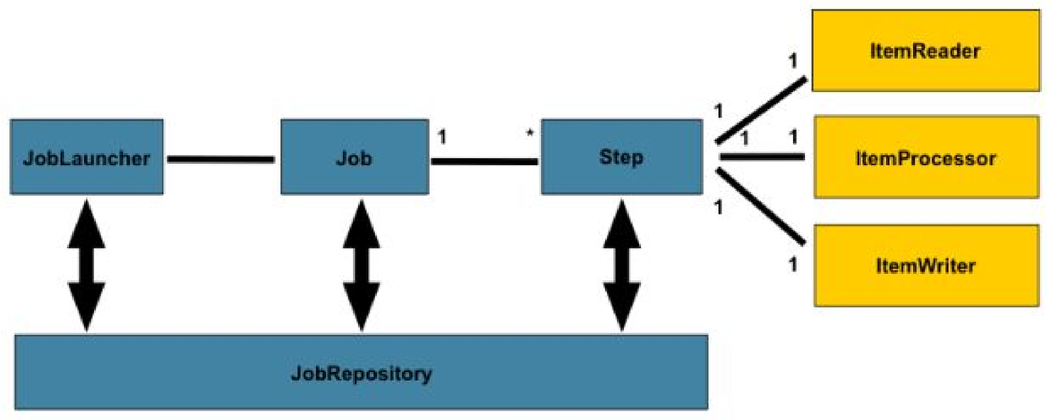
- 사용자의 인터랙션 없이 컴퓨터에 의해 일련의 프로그램 집합이 처리되는 것을 의미.
- Ex) 프린트 출력, 로그 분석, 주기적 처리가 필요한 회계 결산 이나 급여 작업등.

특징	설명
관리자에 의한 실행	- 상용 대형 컴퓨터나 서버에서는 일반적으로 시스템 사용자에 의해 실행
스케쥴링 사용	- 특정 시간에 일정 주기에 따라 실행
백그라운드 실행	- 주로 백그라운드 실행 - 배치 작업보다 우선순위가 높고 전면에서 실행되는 사용자 인터랙티브 프로그램이 요청을 기다리는 시간에 실행.

❑ Spring Batch 기능

- Batch Monitoring 기능 제공
- Commit Interval 지원
- Retry, Restart, Skip 기능 지원
- Commit/Rollback/Retry Count 정보 제공
- Quartz, Command Line, Web 등을 통한 실행 지원

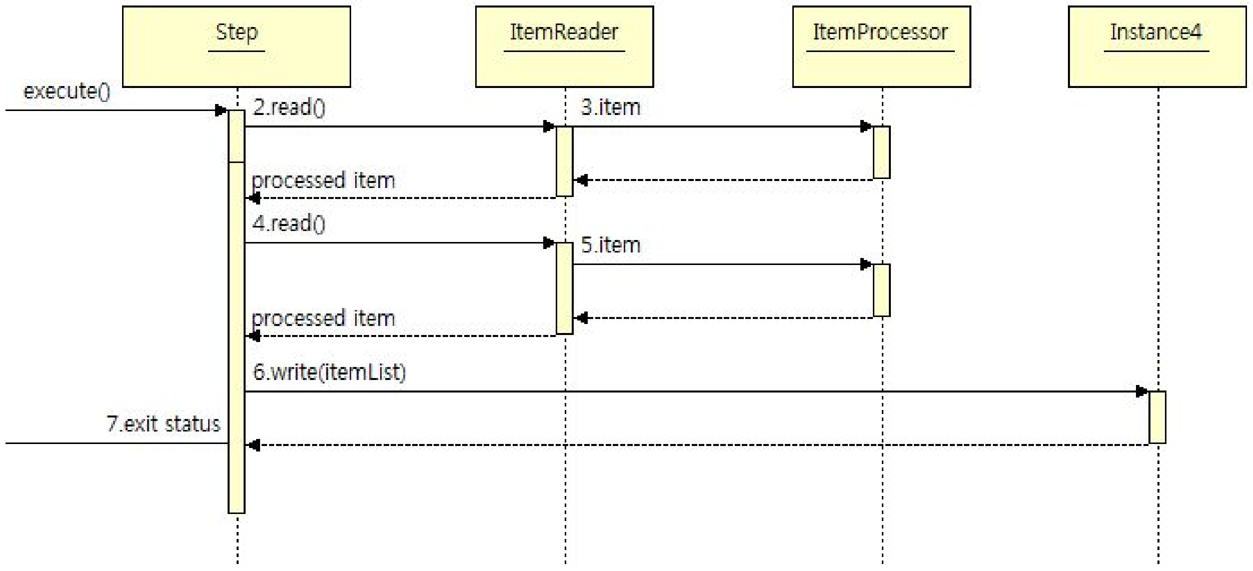
❑ Spring Batch 기본 구성



❑ Spring Batch 구성요소

- JobLauncher : Job 과 JobParameters 를 받아 batch job 을 실행함.
  - JobExecution jobLauncher.run (job, jobParameters)
- Job : 처리할 작업을 의미함, 논리적인 Job 실행의 개념
- JobParamerter : Job 실행에 사용되는 파라미터 집합으로 실행 중 Job 의 식별 혹인 Job 에 의해 참조되는 데이터로 사용.
- JobInstance : 논리적인 Job 실행(JobInstance=Job+JobParameter)
- jobExecution : JobInstance 에 대해 실행을 위한 한번의 시도. 시작시간, 종료시간, status(start,complete,fail) 등의 속성을 가짐.
- JobRepository : Job 실행 정보 저장소. jobInstance, jobExecution, stepExecution, 및 각 context를 저장함.

– Step : Batch job 을 구성하는 독립적인 하나의 단계로 batch 처리를 제어하는데 필요한 모든 정보를 포함.
  - Chunk-oriented step : 조회(ItemReader)-처리(ItemProcessor)-기록(ItemWriter)의 사이클로 구성되는 step.
    - Chunk : 하나의 트랜잭션 안에서 처리되는 아이템들의 묶음.
    - Chunk size : commit-interval 에 의해 결정됨.
  - Task-oriented step : 데이터에 대한 처리 위주의 step.



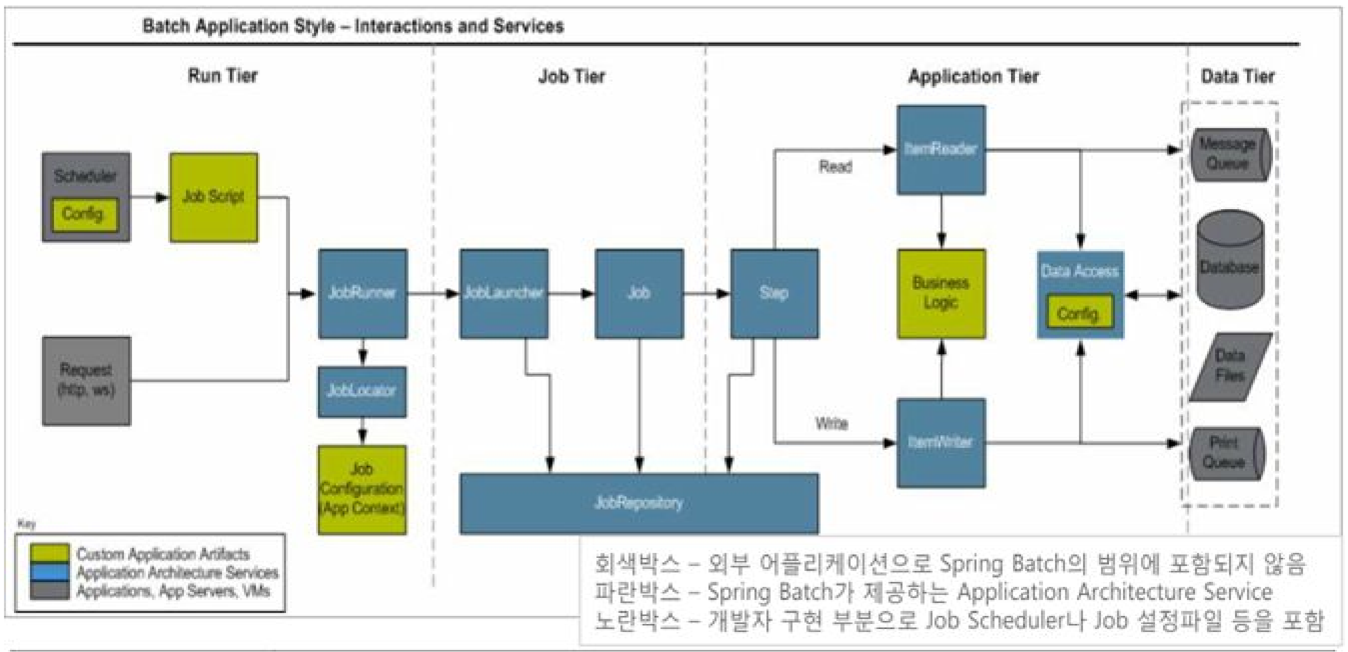
– Item : 처리할 데이터의 가장 작은 구성 요소. Ex) DB의 레코드(엔터티), xml 의 element 등.

– ItemReader : Step (Chunk-orientied) 안에서 파일이나 DB 등으로부터 item 을 읽어들임. 마지막 item 까지 read 한 후 read() 메소드에서 null을 리턴함.

– ItemWriter : Step 안에서 파일 또는 DB 에 item 을 기록함.

– ItemProcessor : itemReader 에서 읽어들인 item에 대한 처리 로직.

❑ Spring Batch Architecture



회색박스 – 외부 어플리케이션으로 Spring Batch의 범위에 포함되지 않음
파란박스 – Spring Batch가 제공하는 Application Architecture Service
노란박스 – 개발자 구현 부분으로 Job Scheduler나 Job 설정파일 등을 포함

❑ Hello batch world 만들기

– Read from ‘datas/member.txt‘ -> processing -> write to database
– Read from database -> processing -> write to output/memberoutput.xml and memberoutput.txt
– Delete datas/member.txt

1. Batch job 설계
   - Job 을 구성하는 step 에 대한 설계
     - Chunked step 을 구성하는 reader, processor, writer 에 대한 설계
2. Batch 어플리케이션 실행 환경 설정.
   - JobRepository 가 meta-data 를 유지할 데이터베이스 영역 생성.

참고) http://docs.spring.io/spring-batch/2.1.x/reference/html/metaDataSchema.html

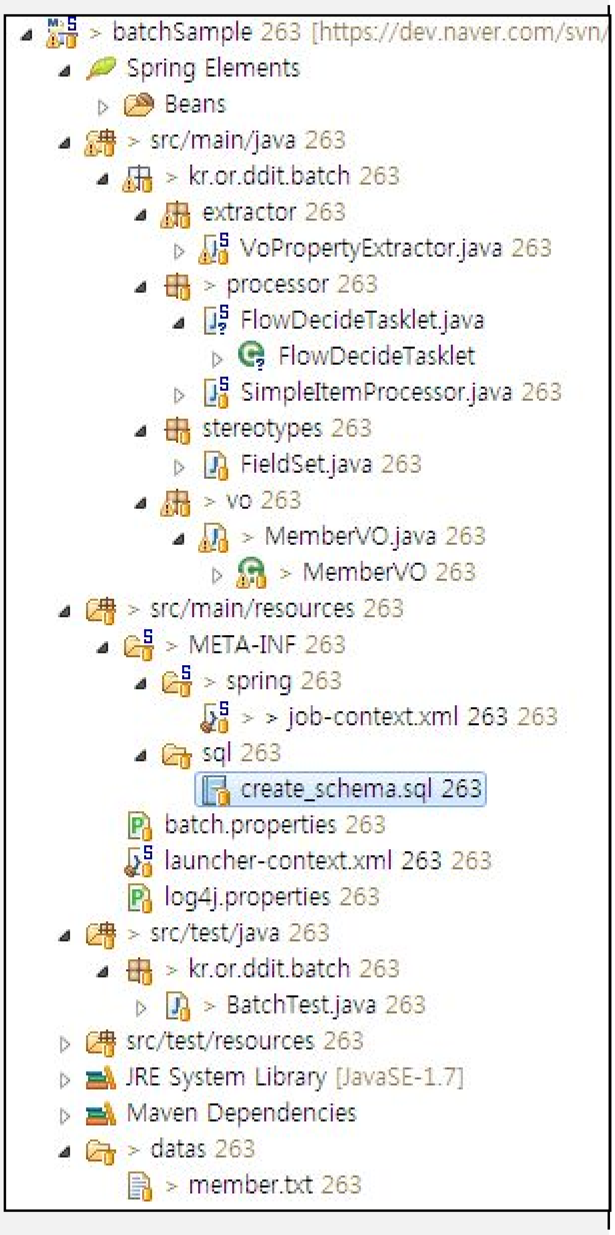
```
CREATE SEQUENCE BATCH_STEP_EXECUTION_SEQ MAXVALUE 9223372036854775807 NOCYCLE;
CREATE SEQUENCE BATCH_JOB_EXECUTION_SEQ MAXVALUE 9223372036854775807 NOCYCLE;
CREATE SEQUENCE BATCH_JOB_SEQ MAXVALUE 9223372036854775807 NOCYCLE;

CREATE TABLE BATCH_JOB_INSTANCE  (
        JOB_INSTANCE_ID NUMBER  NOT NULL PRIMARY KEY ,
        VERSION NUMBER ,
        JOB_NAME VARCHAR2(100) NOT NULL,
        JOB_KEY VARCHAR2(32) NOT NULL,
        constraint JOB_INST_UN unique (JOB_NAME, JOB_KEY)
) ;

CREATE TABLE BATCH_JOB_EXECUTION  (
        JOB_EXECUTION_ID NUMBER  NOT NULL PRIMARY KEY ,
        VERSION NUMBER  ,
        JOB_INSTANCE_ID NUMBER NOT NULL,
        CREATE_TIME DATE NOT NULL,
        START_TIME DATE DEFAULT NULL ,
        END_TIME DATE DEFAULT NULL ,
        STATUS VARCHAR2(10) ,
        EXIT_CODE VARCHAR2(100) ,
        EXIT_MESSAGE VARCHAR2(2500) ,
        LAST_UPDATED DATE,
        constraint JOB_INST_EXEC_FK foreign key (JOB_INSTANCE_ID)
        references BATCH_JOB_INSTANCE(JOB_INSTANCE_ID)
) ;

CREATE TABLE BATCH_JOB_PARAMS  (
        JOB_INSTANCE_ID NUMBER NOT NULL ,
        TYPE_CD VARCHAR2(6) NOT NULL ,
        KEY_NAME VARCHAR2(100) NOT NULL ,
        STRING_VAL VARCHAR2(250) ,
        DATE_VAL DATE DEFAULT NULL ,
        LONG_VAL NUMBER ,
        DOUBLE_VAL DOUBLE PRECISION ,
        constraint JOB_INST_PARAMS_FK foreign key (JOB_INSTANCE_ID)
        references BATCH_JOB_INSTANCE(JOB_INSTANCE_ID)
) ;
```

```
CREATE TABLE BATCH_STEP_EXECUTION  (
            STEP_EXECUTION_ID NUMBER  NOT NULL PRIMARY KEY ,
            VERSION NUMBER NOT NULL,
            STEP_NAME VARCHAR2(100) NOT NULL,
            JOB_EXECUTION_ID NUMBER NOT NULL,
            START_TIME DATE NOT NULL ,
            END_TIME DATE DEFAULT NULL ,
            STATUS VARCHAR2(10) ,
            COMMIT_COUNT NUMBER ,
            READ_COUNT NUMBER ,
            FILTER_COUNT NUMBER ,
            WRITE_COUNT NUMBER ,
            READ_SKIP_COUNT NUMBER ,
            WRITE_SKIP_COUNT NUMBER ,
            PROCESS_SKIP_COUNT NUMBER ,
            ROLLBACK_COUNT NUMBER ,
            EXIT_CODE VARCHAR2(100) ,
            EXIT_MESSAGE VARCHAR2(2500) ,
            LAST_UPDATED DATE,
            constraint JOB_EXEC_STEP_FK foreign key (JOB_EXECUTION_ID)
            references BATCH_JOB_EXECUTION(JOB_EXECUTION_ID)
) ;

CREATE TABLE BATCH_STEP_EXECUTION_CONTEXT  (
            STEP_EXECUTION_ID NUMBER NOT NULL PRIMARY KEY,
            SHORT_CONTEXT VARCHAR2(2500) NOT NULL,
            SERIALIZED_CONTEXT CLOB ,
            constraint STEP_EXEC_CTX_FK foreign key (STEP_EXECUTION_ID)
            references BATCH_STEP_EXECUTION(STEP_EXECUTION_ID)
) ;

CREATE TABLE BATCH_JOB_EXECUTION_CONTEXT  (
            JOB_EXECUTION_ID NUMBER NOT NULL PRIMARY KEY,
            SHORT_CONTEXT VARCHAR2(2500) NOT NULL,
            SERIALIZED_CONTEXT CLOB ,
            constraint JOB_EXEC_CTX_FK foreign key (JOB_EXECUTION_ID)
            references BATCH_JOB_EXECUTION(JOB_EXECUTION_ID)
) ;
```

3. Batch 어플리케이션 구현 환경 설정

maven dependencies 설정

```
<properties>
        <spring.framework.version>3.0.6.RELEASE</spring.framework.version>
        <spring.batch.version>2.1.7.RELEASE</spring.batch.version>
</properties>
<dependencies>
        <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.7</version>
        <scope>test</scope>
        </dependency>
        <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>${spring.framework.version}</version>
        <scope>test</scope>
        </dependency>
        <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${spring.framework.version}</version>
        </dependency>
        <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.framework.version}</version>
        </dependency>
        <dependency>
        <groupId>cglib</groupId>
        <artifactId>cglib-nodep</artifactId>
        <version>2.2</version>
        </dependency>
        <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-aop</artifactId>
        <version>${spring.framework.version}</version>
        </dependency>
        <dependency>
        <groupId>org.springframework.batch</groupId>
        <artifactId>spring-batch-core</artifactId>
        <version>${spring.batch.version}</version>
        </dependency>
        <dependency>
        <groupId>org.springframework.batch</groupId>
        <artifactId>spring-batch-infrastructure</artifactId>
        <version>${spring.batch.version}</version>
        </dependency>
        <dependency>
        <groupId>commons-dbcp</groupId>
        <artifactId>commons-dbcp</artifactId>
        <version>1.2.2</version>
        </dependency>
```



```
batchSample 263 [https://dev.naver.com/svn/
  Spring Elements
    Beans
  src/main/java 263
    kr.or.ddit.batch 263
      extractor 263
        VoPropertyExtractor.java 263
      processor 263
        FlowDecideTasklet.java
          FlowDecideTasklet
        SimpleItemProcessor.java 263
      stereotypes 263
        FieldSet.java 263
      vo 263
        MemberVO.java 263
          MemberVO 263
  src/main/resources 263
    META-INF 263
      spring 263
        job-context.xml 263 263
      sql 263
        create_schema.sql 263
    batch.properties 263
    launcher-context.xml 263 263
    log4j.properties 263
  src/test/java 263
    kr.or.ddit.batch 263
      BatchTest.java 263
  src/test/resources 263
  JRE System Library [JavaSE-1.7]
  Maven Dependencies
  datas 263
    member.txt 263
```

```
            <dependency>
            <groupId>commons-io</groupId>
            <artifactId>commons-io</artifactId>
            <version>1.4</version>
            </dependency>
            <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.6.8</version>
            </dependency>
            <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.6.8</version>
            </dependency>
            <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.14</version>
            </dependency>
            <dependency>
            <groupId>com.oracle</groupId>
            <artifactId>ojdbc14</artifactId>
            <version>10.2.0.1.0</version>
            <type>pom.lastUpdated</type>
            </dependency>
            <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
            <version>${spring.framework.version}</version>
            </dependency>
            <dependency>
            <groupId>org.quartz-scheduler</groupId>
            <artifactId>quartz</artifactId>
            <version>2.2.1</version>
            </dependency>
            <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-oxm</artifactId>
                        <version>${spring.framework.version}</version>
            </dependency>
</dependencies>
```

4. Batch job meta-data 관리 및 job 실행을 담당하는 batch artifact bean 등록. (classpath:launcher-context.xml)

```
<context:property-placeholder location="classpath:batch.properties"/>
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
          p:driverClassName="${batch.jdbc.driverClassName}"
          p:url="${batch.jdbc.url}"
          p:username="${batch.jdbc.username}"
          p:password="${batch.jdbc.password}"
/>
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
                    p:dataSource-ref="dataSource"
/>
<batch:job-repository id="jobRepository"/>
<bean id="jobLauncher" class="org.springframework.batch.core.launch.support.SimpleJobLauncher"
                    p:jobRepository-ref="jobRepository"
/>
<import resource="classpath:/META-INF/spring/job-context.xml"/>
```

5. ItemProcessor 구현 및 Supplement artifact 구현

ïItemProcessor

```
package kr.or.ddit.batch.processor;
//import ….
public class SimpleItemProcessor implements ItemProcessor<MemberVO, MemberVO>{

                    @Override
                    public MemberVO process(MemberVO item) throws Exception {
                              System.out.println(item.getMem_name());
                              return item;
                    }

}
```

FieldSet 매핑을 위한 single value annotation

```
package  kr.or.ddit.batch.stereotypes;
// import…
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface FieldSet {
                    public String[] value();
}
```

흐름을 제어하기 위한 tasklet

```
package kr.or.ddit.batch.processor;
// import…
public class FlowDecideTasklet implements Tasklet{
		// step3 tasklet
		@Override
		public RepeatStatus execute(StepContribution contribution,
					ChunkContext chunkContext) throws Exception {
//				contribution.setExitStatus(ExitStatus.FAILED);
				contribution.setExitStatus(ExitStatus.COMPLETED);
				return RepeatStatus.FINISHED;
		}
		// step 3 실패시 실행될 task
		public void executeAtFail(){
				System.err.println("Step2 Failed! Job running Stop!");
		}
}
```

FlatFileItemWriter 에서 VO->item Line 시 사용할 extractor

```
package kr.or.ddit.batch.extractor;
// import..
/**
 * Single-value 어노테이션인 @FieldSet 을 통해 해당 VO의 프로퍼티 데이터를 배열로 추출.
*/
public class VoPropertyExtractor<T> implements FieldExtractor<T>{

		private String[] fields;
		@Override
		public Object[] extract(T item) {
				Class<? extends Object> clz = item.getClass();
				if(fields==null){
					FieldSet fieldSet = (FieldSet) clz.getAnnotation(FieldSet.class);
					fields = fieldSet.value();
				}
				Object[] returnValue = new Object[fields.length];
				for(int idx=0; idx<fields.length; idx++){
					PropertyDescriptor pd;
					try {
						pd = new PropertyDescriptor(fields[idx], clz);
						Method getter = pd.getReadMethod();
						Object tempObj = getter.invoke(item, null);
						returnValue[idx] = tempObj;
					} catch (Exception e) {
						throw new RuntimeException(e);
					}
				}
				return returnValue;
		}

}
```

VO

```
package kr.or.ddit.batch.vo;
// import…
@FieldSet({"mem_id", "mem_pass", "mem_name", "mem_regno1", "mem_regno2",
"mem_bir", "mem_zip", "mem_add1", "mem_add2", "mem_hometel",
"mem_comtel", "mem_hp", "mem_mail", "mem_job", "mem_like",
"mem_memorial", "mem_memorialday", "mem_mileage", "mem_delete"
})
@XmlRootElement(name="member")
@XmlAccessorType(XmlAccessType.PROPERTY)
public class MemberVO implements Serializable{
        private String mem_id;
        private String mem_pass;
        private String mem_name;
        private String mem_regno1;
        private String mem_regno2;
        private String mem_bir;
        private String mem_zip;
        private String mem_add1;
        private String mem_add2;
        private String mem_hometel;
        private String mem_comtel;
        private String mem_hp;
        private String mem_mail;
        private String mem_job;
        private String mem_like;
        private String mem_memorial;
        private String mem_memorialday;
        private Integer mem_mileage;
        private String mem_delete;
        //getter/setter
}
```

6. Batch job 기술.(batch 스키마 사용 : JSR-352 스펙의 Job Specification Language 기반)

(classpath:/META-INF/spring/job-context.xml) : step1 구성요소 등록

```
<!-- step1 artifact start -->
<bean id="fileReader" class="org.springframework.batch.item.file.FlatFileItemReader"
            p:resource="file:${user.dir}/datas/member.txt"
>
      <property name="lineMapper">
            <bean class="org.springframework.batch.item.file.mapping.DefaultLineMapper">
                  <property name="lineTokenizer">
                        <bean
                     class="org.springframework.batch.item.file.transform.DelimitedLineTokenizer"
                        p:delimiter="|"
                        p:names="#{T(org.springframework.core.annotation.AnnotationUtils).findAnnotatio
                     n(T(Class).forName('kr.or.ddit.batch.vo.MemberVO'),
                     T(Class).forName('kr.or.ddit.batch.stereotypes.FieldSet')).value()}"
                        />
                  </property>
                  <property name="fieldSetMapper">
                        <bean
                     class="org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper"
                        p:targetType="kr.or.ddit.batch.vo.MemberVO"
                        />
                  </property>
            </bean>
      </property>
</bean>
<bean id="itemProcessor" class="kr.or.ddit.batch.processor.SimpleItemProcessor" />
<bean id="jdbcWriter" class="org.springframework.batch.item.database.JdbcBatchItemWriter"
            p:dataSource-ref="dataSource"
>
      <property name="sql">
            <value>
                  INSERT INTO MEMBER
                   (
                    MEM_ID,    MEM_PASS,    MEM_NAME,    MEM_REGNO1,
                    MEM_REGNO2,    MEM_BIR,    MEM_ZIP,    MEM_ADD1,
                    MEM_ADD2,    MEM_HOMETEL,    MEM_COMTEL,    MEM_HP,
                    MEM_MAIL,    MEM_JOB,    MEM_LIKE,    MEM_MEMORIAL,
                    MEM_MEMORIALDAY,    MEM_MILEAGE,    MEM_DELETE
                   )
                   VALUES
                   (
                    :mem_id,    :mem_pass,    :mem_name,    :mem_regno1,
                    :mem_regno2,    TO_DATE(:mem_bir, 'YYYY-MM-DD'),    :mem_zip,    :mem_add1,
                    :mem_add2,    :mem_hometel,    :mem_comtel,    :mem_hp,
                    :mem_mail,    :mem_job,    :mem_like,    :mem_memorial,
                    TO_DATE(:mem_memorialday, 'YYYY-MM-DD'),    :mem_mileage,    :mem_delete
                   )
            </value>
      </property>
      <property name="itemSqlParameterSourceProvider">
            <bean
     class="org.springframework.batch.item.database.BeanPropertyItemSqlParameterSourceProvider" />
      </property>
</bean>
<!-- step1 artifact end -->
```

```
<!-- step2 artifact start -->
<bean id="flowDecideTasklet" class="kr.or.ddit.batch.processor.FlowDecideTasklet" />
<!-- step2 artifact end -->
<!-- step3 artifact start -->
<bean id="jdbcReader" class="org.springframework.batch.item.database.JdbcCursorItemReader"
        p:dataSource-ref="dataSource"
>
        <property name="sql">
                <value>
                        SELECT MEM_ID,  MEM_PASS,  MEM_NAME,  MEM_REGNO1,
                          MEM_REGNO2,  TO_CHAR(MEM_BIR, 'YYYY-MM-DD') AS MEM_BIR,  MEM_ZIP,
                        MEM_ADD1,
                          MEM_ADD2,  MEM_HOMETEL,  MEM_COMTEL,  MEM_HP,
                          MEM_MAIL,  MEM_JOB,  MEM_LIKE,  MEM_MEMORIAL,
                          TO_CHAR(MEM_MEMORIALDAY, 'YYYY-MM-DD') AS MEM_MEMORIALDAY,
                        MEM_MILEAGE,  MEM_DELETE
                        FROM MEMBER
                </value>
        </property>
        <property name="rowMapper">
                <bean class="org.springframework.jdbc.core.BeanPropertyRowMapper"
                        p:checkFullyPopulated="true" p:primitivesDefaultedForNullValue="true"
                >
                        <property name="mappedClass" value="kr.or.ddit.batch.vo.MemberVO" />
                </bean>
        </property>
</bean>
<bean id="jaxb2Marshaller" class="org.springframework.oxm.jaxb.Jaxb2Marshaller"
        p:classesToBeBound="kr.or.ddit.batch.vo.MemberVO"
/>
<bean id="xmlWriter" class="org.springframework.batch.item.xml.StaxEventItemWriter"
        p:resource="file:${user.dir}/output/memberoutput.xml"
        p:marshaller-ref="jaxb2Marshaller"
        p:rootTagName="members"
        p:transactional="true"
        p:saveState="true"
        p:overwriteOutput="true"
></bean>
<bean id="fileWriter" class="org.springframework.batch.item.file.FlatFileItemWriter"
        p:resource="file:${user.dir}/output/memberoutput.txt"
        p:appendAllowed="true"
        p:encoding="UTF-8"
        p:shouldDeleteIfEmpty="true"
        p:transactional="true"
>
        <property name="lineAggregator">
                <bean class="org.springframework.batch.item.file.transform.DelimitedLineAggregator"
                        p:delimiter="|"
                >
                        <property name="fieldExtractor">
                                <bean class="kr.or.ddit.batch.extractor.VoPropertyExtractor" />
                        </property>
                </bean>
        </property>
</bean>
<!-- step3 artifact end -->
```

– Job 구성 요소(step) 등록

```
<batch:job id="sampleJob1" restartable="false">
            <batch:step id="step1" next="step2">
                        <batch:tasklet>
                                    <batch:chunk commit-interval="5"
                                    reader="fileReader"
                                    processor="itemProcessor"
                                    writer="jdbcWriter"
                                    />
                        </batch:tasklet>
            </batch:step>

            <batch:step id="step2">
                        <batch:tasklet ref="flowDecideTasklet" />
                        <batch:next on="COMPLETED" to="step3"/>
                        <batch:next on="FAILED" to="step4"/>
            </batch:step>

            <batch:step id="step3">
                        <batch:tasklet>
                                    <batch:chunk
                                    commit-interval="5"
                                    reader="jdbcReader"
                                    >
                                                <batch:writer>
                                                            <bean
                                                            class="org.springframework.batch.item.support.CompositeItemWriter">
                                                                        <property name="delegates">
                                                                        <list>
                                                                        <ref bean="xmlWriter"/>
                                                                        <ref bean="fileWriter"/>
                                                                        </list>
                                                                        </property>
                                                            </bean>
                                                </batch:writer>
                                                </batch:chunk>
                        </batch:tasklet>
            </batch:step>

            <batch:step id="step4">
                        <batch:tasklet ref="flowDecideTasklet" method="executeAtFail" />
            </batch:step>
</batch:job>
```