http://www.egovframe.go.kr/wiki/doku.php?id=egovframework:rte2:brte:batch_core:item_writer

ItemWriter

Outline

ItemWriter is an interface for a chunk of items, regardless of type of object.

Description

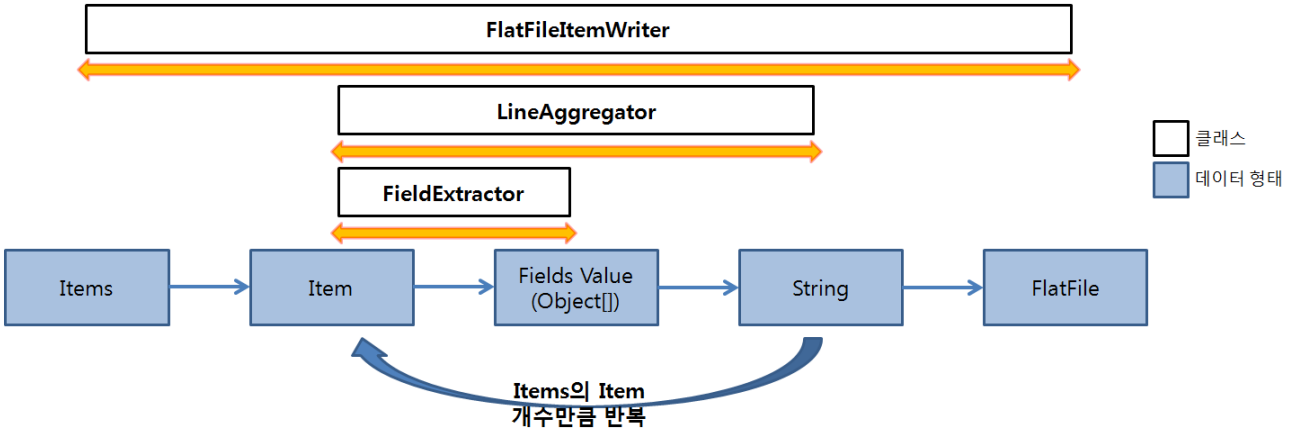
ItemWriter works similarly to ItemReader, but for a polar opposite function.
See the following for how ItemWriter interface works:

```
public interface ItemWriter<T> {

    void write(List<? extends T> items) throws Exception;

}
```

The method write() is required for ItemWriter and attempts writing while the object is handed off to the factor.

FlatFile ItemWriter

FlatFileItemWriter is dependent on Resource and LineAggregator which defines FlatFileItemWriter as either delimiter or fixed-length line.



Category	Data	Description
LineAggregator	Item → String	A series of processes for the items through ItemReader and ItemProcessor to be converted into the String of Line 1 (inclusive of FieldExtractor) - **DelimitedLineAggregator** : Conversion of item and field into the String of Line 1 with the delimiter between them. - **FormatterLineAggregator** : Conversion of item and field into the String of Line 1 based upon the customized format.
FieldExtractor	Item → Fields	Conversion of fields out of an item into Object[].

See the following examples for how FlatFileItemWriter, segmented by either delimiter or fixed length, is dependent on FlatFileItemWriter, LineMapper, LineTokenizer and FieldSetMapper:

Aggregate Method	Settings

Delimiter

```
<bean id="itemWriter" class="org.springframework.batch.item.file.FlatFileItemWriter" scope="step">
    <property name="resource" value="#{jobParameters[outputFile]}" />
    <property name="lineAggregator">
        <bean class="org.springframework.batch.item.file.transform.DelimitedLineAggregator">
            <property name="delimiter" value=","/>
            <property name="fieldExtractor">
                <bean
class="org.springframework.batch.item.file.transform.BeanWrapperFieldExtractor">
                    <property name="names" value="name,credit"/>
                </bean>
            </property>
        </bean>
    </property>
</bean>
```

Fixed Length

```
<bean id="itemWriter" class="org.springframework.batch.item.file.FlatFileItemWriter" scope="step">
    <property name="resource" value="#{jobParameters[outputFile]}" />
    <property name="lineAggregator">
        <bean class="org.springframework.batch.item.file.transform.FormatterLineAggregator">
            <property name="format" value="%-9s%-2s" />
            <property name="fieldExtractor">
                <bean
class="org.springframework.batch.item.file.transform.BeanWrapperFieldExtractor">
                    <property name="names" value="name,credit"/>
                </bean>
            </property>
        </bean>
    </property>
</bean>
```

You need to make sure you configure BeanWrapperFieldExtractor as follows:

Items Configured	Description
names	Represents fields of class VO

✔ Particular attention is needed for configuration to fit the concerned LineAggregator.

LineAggregator	Items Configured	Description	Configuration
DelimitedLineAggregator	delimiter	Delimiter being the threshold of conversion of the fields of an item into the String of Line 1.	, (comma)
FormatterLineAggregator	format	Type and fixed length of fields for conversion of the fields of an item into the String of Line 1.	%-9s%-2s

XML ItemWriter

StaxEventItemWriter

Note that XML reading is symmetrical to XML writing. Requiring Resource, marshaller and rootTagName StaxEventItemWriter involves transfer of Java object to marshaller for filtering of the events StartDocument and EndDocument for each fragment, by way of OXM tools, and using of customized event writer for resources.

See the following for how StaxEventItemWriter can be configured using XStreamMarshaller:

```
<bean id="itemWriter" class="org.springframework.batch.item.xml.StaxEventItemWriter">
    <property name="resource" ref="outputResource" />
    <property name="marshaller" ref="customerCreditMarshaller" />
    <property name="rootTagName" value="customers" />
```

```
    <property name="overwriteOutput" value="true" />
</bean>
```

See the following for how customerCreditMarshaller shall be configured for reference of dependency by marshaller:

```
<bean id="customerCreditMarshaller" class="org.springframework.oxm.xstream.XStreamMarshaller">
    <property name="aliases">
        <util:map id="aliases">
            <entry key="customer" value="egovframework.brte.sample.domain.trade.CustomerCredit" />
            <entry key="credit" value="java.math.BigDecimal" />
            <entry key="name" value="java.lang.String" />
        </util:map>
    </property>
</bean>
```

Database ItemWriter

With transaction doing it all that are required to ItemWriter of flat file and XML where it comes into play for database to trace and remove items written at certain moments, database does not require item-writing as the transaction comprises the equivalent function. It is thus advised for user to implement ItemWriter interface to generate DAO or use one of those customized ItemWriters written from th perspective of general processing.

JdbcBatchItemWriter

Refer to the following examples for how XML configuration is made, using JDBC's JdbcBatchItemWriter.
Like JdbcCursorItemReader, you can designate the dataSource for DB connection and configure the queries to be executed for the attribute SQL:

```
<bean id="itemWriter" class="org.springframework.batch.item.database.JdbcBatchItemWriter">
        <property name="assertUpdates" value="true" />
        <property name="itemSqlParameterSourceProvider">
                <bean
class="org.springframework.batch.item.database.BeanPropertyItemSqlParameterSourceProvider" />
        </property>
        <property name="sql" value="UPDATE CUSTOMER set credit = :credit where id = :id" />
        <property name="dataSource" ref="dataSource" />
</bean>
```

IbatisBatchItemWriter

Refer to the following XML configuration for IbatisBatchItemWriter:

```
<bean id="itemWriter"
        class="org.springframework.batch.item.database.IbatisBatchItemWriter">
        <property name="statementId" value="updateCredit" />
        <property name="sqlMapClient" ref="sqlMapClient" />
</bean>
```

See the following for how sqlMapClient is referred to. All you need to do is to configure the directory of query files prepped by iBatis in the attribute configLocation:

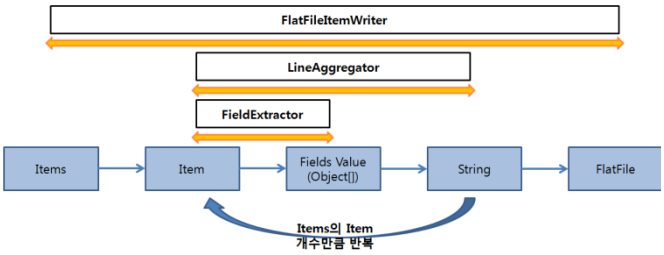
```
<bean id="sqlMapClient" class="org.springframework.orm.ibatis.SqlMapClientFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="configLocation" value="ibatis-config.xml" />
</bean>
```

eGovFlatFileItemWriter, specific to eGovFramework

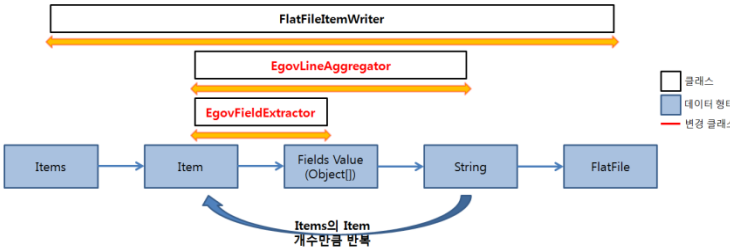
For improved performance in file-based configuration in Spring, eGovFramework offers eGovFlatFileItemWriter that has no LineMapper, the largest contributor to latency.

Also, eGovFramework offer **EgovFieldExtractor and EgovFixedLineAggregator**, the lighter versions of BeanWrapperFieldExtractor and FormatterLineAggregator.

Structure of Spring's FlatFileItemWriter



Structure of eGovFramework's eGovFlatFileItemWriter



Improvements	Description
EgovFieldExtractor	Offers FieldExtractor that features improved field extraction performance compared to BeanWrapperFieldExtractor.
EgovFixedLineAggregator	Contrary to the Spring's FormatterLineAggregator that converts strings into various formats using a Java method format() that involves latency when the length of string is assigned, the LineAggregator focuses on assignment of the length of string to get rid of latency (when needed to designate format, you can also take the best advantage of EgovFixedLineAggregator to get rid of latency by applying formats directly from VO).

✔ Concurrent use of Spring's FormatterLineAggregator, DelimitedLineAggregator and EgovFieldExtractor is available. It is thus advised to replace BeanWrapperFieldExtractor into EgovFieldExtractor for high performance writing.

Refer to the following for how FlatFileItemWriter configurations made based on BeanWrapperFieldExtractor / FormatterLineAggregator(or DelimitedLineAggregator) and EgovFieldExtractor / EgovFixedLineAggregator(or DelimitedLineAggregator) can be compared:

Fixed Length Configuration

Category	Settings
Spring's FlatFileItemWriter	<bean id="itemWriter" class="org.springframework.batch.item.file.FlatFileItemWriter" scope="step"> <property name="resource" value="#{jobParameters[outputFile]}" /> <property name="lineAggregator"> <bean class="org.springframework.batch.item.file.transform.FormatterLineAggregator"> <property name="fieldExtractor"> <bean class="org.springframework.batch.item.file.transform.BeanWrapperFieldExtractor"> <property name="names" value="name,credit" /> </bean> </property> <property name="format" value="%-9s%-2s" /> </bean> </property> </bean>
eGovFramework's eGovFlatFileItemWriter	<bean id="itemWriter" class="org.springframework.batch.item.file.FlatFileItemWriter" scope="step"> <property name="resource" value="#{jobParameters[outputFile]}" /> <property name="lineAggregator"> <bean class="egovframework.brte.core.item.file.transform.EgovFixedLengthLineAggregator"> <property name="fieldExtractor">

```
                <bean
class="egovframework.brte.core.item.file.transform.EgovFieldExtractor">
                    <property name="names" value="name,credit" />
                </bean>
            </property>
            <property name="fieldRanges" value="9,2" />
        </bean>
    </property>
</bean>
```

How to configure EgovFlatFileItemWriter	Description
fieldRanges	Range (fixed length) of fields for conversion of the fields of an item into the String of Line 1.
names	Represents fields of class VO
padding	Pattern of padding

How to configure Delimiter types

Category	Settings

Spring's FlatFileItemWriter	`<bean id="itemWriter" class="org.springframework.batch.item.file.FlatFileItemWriter" scope="step">`
	`  <property name="resource" value="#{jobParameters[outputFile]}" />`
	`  <property name="lineAggregator">`
	`  <bean class="org.springframework.batch.item.file.transform.DelimitedLineAggregator">`
	`  <property name="fieldExtractor">`
	`  <bean class="org.springframework.batch.item.file.transform.BeanWrapperFieldExtractor">`
	`  <property name="names" value="name,credit"/>`
	`  </bean>`
	`  </property>`
	`  <property name="delimiter" value=","/>`
	`  </bean>`
	`  </property>`
	`</bean>`

eGovFramework's eGovFlatFileItemWriter	`<bean id="itemWriter" class="org.springframework.batch.item.file.FlatFileItemWriter" scope="step">`
	`  <property name="resource" value="#{jobParameters[outputFile]}" />`
	`  <property name="lineAggregator">`
	`  <bean class="org.springframework.batch.item.file.transform.DelimitedLineAggregator">`
	`  <property name="fieldExtractor">`
	`  <bean class="egovframework.brte.core.item.file.transform.EgovFieldExtractor">`
	`  <property name="names" value="name,credit"/>`
	`  </bean>`
	`  </property>`
	`  <property name="delimiter" value=","/>`
	`  </bean>`
	`  </property>`
	`</bean>`

How to configure EgovFlatFileItemWriter	Description
delimiter	Delimiter being the threshold of conversion of the fields of an item into the String of Line 1.
names	Represents fields of class VO

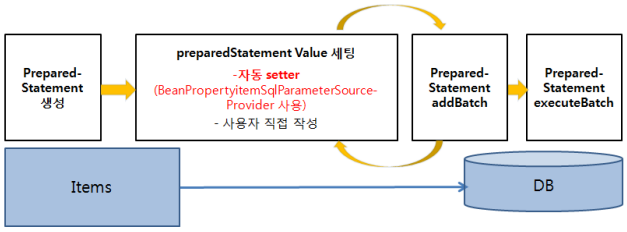
Note the configurations may vary by LineAggregator you use.

LineAggregator	Items Configured	Description	Configuration
DelimitedLineAggregator	delimiter	Thresholds of field	, (comma)
FormatterLineAggregator	format	Type and threshold range of field	%-9s%-2s
EgovFixedLengthLineAggregator	fieldRanges	Threshold range of field	9,2

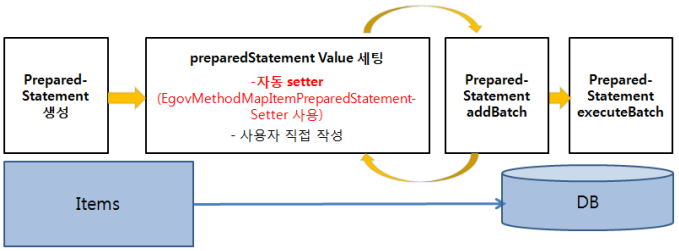
eGovDBItemWriter, specific to eGovFramework

Contrary to Spring's JdbcBatchItemWriter sets PreparedStatement based upon the customized configuration of query parameters for XML configuration, eGovDBItemWriter gets rid of latency when massive data are processed, realized in eGovFramework by way of EgovJdbcBatchItemWriter.

Structure of Spring's JdbcBatchItemWriter



Structure of eGovFramework's EgovJdbcBatchItemWriter



For automatic setting of PreparedStatement, JdbcBatchItemWriter and EgovJdbcBatchItemWriter use the classes BeanPropertyItemSqlParameterSourceProvider and EgovMethodMapItemPreparedStatementSetter, respectively. See the following for how these are configured:

Category	Settings
JdbcBatchItemWriter	<bean id="itemWriter" class="org.springframework.batch.item.database.JdbcBatchItemWriter"> <property name="assertUpdates" value="true" /> <property name="itemSqlParameterSourceProvider">\<bean class="org.springframework.batch.item.database.BeanPropertyItemSqlParameterSourceProvider" /> </property> <property name="sql" value="UPDATE CUSTOMER set credit = :credit where id = :id" /> <property name="dataSource" ref="dataSource" /> </bean>
EgovJdbcBatchItemWriter	<bean id="itemWriter" class="egovframework.brte.core.item.database.EgovJdbcBatchItemWriter"> <property name="assertUpdates" value="true" /> <property name="itemPreparedStatementSetter">\<bean class="egovframework.brte.core.item.database.support.EgovMethodMapItemPreparedStatementSetter" /> </property> <property name="sql" value="UPDATE CUSTOMER set credit =? where id =?"/> <property name="params" value="credit,id"/> <property name="dataSource" ref="dataSource" /> </bean>

✔ In EgovMethodMapItemPreparedStatementSetter, you can configure the parameter values in the form of values in ‘params’.

See the following for more details on configurations:

writer	Settings	Description
JdbcBatchItemWriter	Customized class	Customized setValues for configuration in PreparedStatement
	BeanPropertyItemSqlParameterSourceProvider	Configures data for PreparedStatement using parameters in SQL.
EgovJdbcBatchItemWriter	Customized class	Customized setValues for configuration in PreparedStatement
	EgovMethodMapItemPreparedStatementSetter	Configures data for PreparedStatement using parameters in SQL and 'params'.

EgovJdbcBatchItemWriter also configures the customized class in th form of PreparedStatementSetter. Refer to the following for how the class EmployeeItemPreparedStatementSetter inherited EgovItemPreparedStatementSetter for customized preparation of statements:

```
<bean id="itemWriter" class="org.springframework.batch.item.database.EgovJdbcBatchItemWriter">
        <property name="itemPreparedStatementSetter">
                <bean
class="egovframework.brte.sample.example.support.EmployeeItemPreparedStatementSetter" />
        </property>
        <property    name="sql“    value="update into UIP_EMPLOYEE (num, name, sex) values (?, ?, ?)" />
        <property name="dataSource" ref="dataSource" />
</bean>
```

References

- http://static.springsource.org/spring-batch/reference/html/readersAndWriters.html