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

# Delimited Examples

## Outline

This Section describes how the delimiters are used to process the flat file data. In Spring, you can use both FlatFileItemReader and FlatFileItemWriter to do so.

## Description

### Settings

#### Configuring Jobs

**Check out delimitedIoJob.xml, the job configuration file for the Delimited example.**

The following configurations are available in FlatFileItemReader:

- resource : Processable files
- lineMapper : A mapper of lines for file that comprises lineTokenizer and fieldSetMapper
  - lineTokenizer : A tokenizer of lines to split the lines to the fixed position via DelimitedLineTokenizer to establish a fieldSet Object
  - fieldSetMapper : FieldSet to be mapped into the object.

```
<bean id="itemReader" class="org.springframework.batch.item.file.FlatFileItemReader" scope="step">
	<property name="lineMapper">
		<bean class="org.springframework.batch.item.file.mapping.DefaultLineMapper">
			<property name="lineTokenizer">
				<bean
class="org.springframework.batch.item.file.transform.DelimitedLineTokenizer">
					<property name="delimiter" value=","/>
					<property name="names" value="name,credit" />
				</bean>
			</property>
			<property name="fieldSetMapper">
				<bean
class="org.springframework.batch.item.file.mapping.BeanWrapperFieldSetMapper">
					<property name="targetType"
value="egovframework.brte.sample.common.domain.trade.CustomerCredit" />
				</bean>
			</property>
		</bean>
	</property>
	<property name="resource" value="#{jobParameters[inputFile]}" />
</bean>
```

Refer to the following for how FlatFileItemWriter is configured:

- resource : Result file
- lineAggregator : Converts the object into the string to be written in the file. Also converts the object into FieldSet in FieldSetCreator and String as per the pre-defined format in DelimitedLineAggregator.

```
<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>
```

## Composition and Implementation of JunitTest

### Composition of JunitTest

#### Implement the Delimited example and verify the result of batch by comprising @Test as follows:

✔ See Junit Test Description for Batch Execution for more information.
✔ The parameter information required for querying is to be sent out to the JobParameter in getUniqueJobParameters.
✔ Pre-batch data and post-batch data are to be compared to each other in EgovAbstractIoSampleTests.
✔ assertEquals(BatchStatus.COMPLETED, jobExecution.getStatus()): Check out the batch implementation is COMPLETED.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "/egovframework/batch/jobs/delimitedIoJob.xml")
public class EgovDelimitedFunctionalTests extends EgovAbstractIoSampleTests {

        ...

        @Override
        protected JobParameters getUniqueJobParameters() {
                return new JobParametersBuilder(super.getUniqueJobParameters()).addString("inputFile",
                                "/egovframework/data/input/delimited.csv").addString("outputFile",
                                "file:./target/test-outputs/delimitedOutput.csv").toJobParameters();
        }

}
@ContextConfiguration(locations = { "/egovframework/batch/simple-job-launcher-context.xml",
"/egovframework/batch/job-runner-context.xml"})
@TestExecutionListeners( { DependencyInjectionTestExecutionListener.class,
StepScopeTestExecutionListener.class })
public abstract class EgovAbstractIoSampleTests {

        // JobLauncherTestUtils to test the batches.
        @Autowired
        @Qualifier("jobLauncherTestUtils")
        private JobLauncherTestUtils jobLauncherTestUtils;

        // Reader for batches
        @Autowired
        private ItemReader<CustomerCredit> reader;

        /**
         * Batch Testing
         */
        @Test
        public void testUpdateCredit() throws Exception {

                open(reader);
                List<CustomerCredit> inputs = getCredits(reader);
```

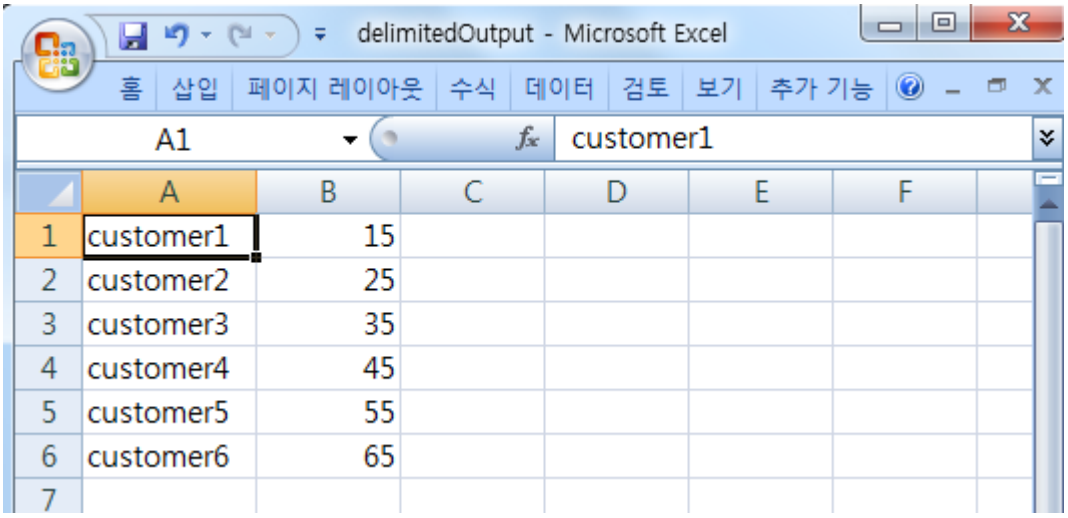
```
                close(reader);

                JobExecution jobExecution = jobLauncherTestUtils.launchJob(getUniqueJobParameters());
                assertEquals(BatchStatus.COMPLETED, jobExecution.getStatus());

                pointReaderToOutput(reader);
                open(reader);
                List<CustomerCredit> outputs = getCredits(reader);
                close(reader);

                assertEquals(inputs.size(), outputs.size());
                int itemCount = inputs.size();
                assertTrue(itemCount > 0);

                for (int i = 0; i < itemCount; i++) {

        assertEquals(inputs.get(i).getCredit().add(CustomerCreditIncreaseProcessor.FIXED_AMOUNT).intValue(),
                                                    outputs.get(i).getCredit().intValue());
                }

        }

    ...
}
```

**Implementation of JunitTest**

See Implementation of JunitTest for more information.

### Verification of Result

You can check out the following files are generated as a result of the foregoing test. You may also check the relevant data are modified when the Job is implemented.




## References

- FlatFileItemReader
- FlatFile ItemWriter