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

FixedLength Examples

Outline

This Section describes how the fixed length flat file data are processed. In Spring, you can use both FlatFileItemReader and FlatFileItemWriter to do so.

Description

Settings

Configuring Jobs

Check out fixedLengthIoJob.xml, the job configuration file for the FixedLength 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 FixedLengthTokenizer 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="resource" value="#{jobParameters[inputFile]}" />
	<property name="lineMapper">
		<bean class="org.springframework.batch.item.file.mapping.DefaultLineMapper">
			<property name="lineTokenizer">
				<bean
class="org.springframework.batch.item.file.transform.FixedLengthTokenizer">
					<property name="names" value="name,credit" />
					<property name="columns" value="1-9,10-11" />
				</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>
</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 FormatterLineAggregator.

```
<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%-2.0f" />
                        </bean>
                </property>
</bean>
```

Composition and Implementation of JunitTest

Composition of JunitTest

Implement the FixedLength 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/fixedLengthIoJob.xml")
public class EgovFixedLengthFunctionalTests extends EgovAbstractIoSampleTests {

        ...

        @Override
        protected JobParameters getUniqueJobParameters() {
                return new JobParametersBuilder(super.getUniqueJobParameters()).addString("inputFile",
                                "/egovframework/data/input/fixedLength.txt").addString("outputFile",
                                "file:./target/test-outputs/fixedLengthOutput.txt").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);
```

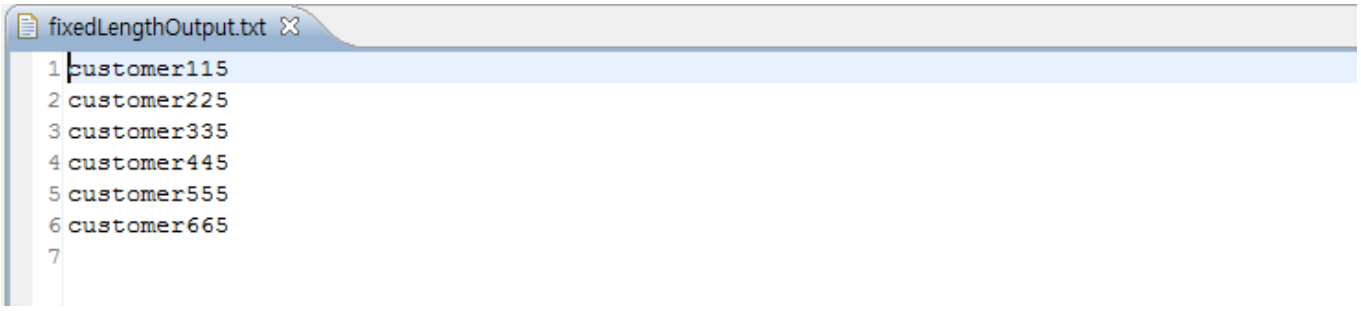
```
            close(reader);

            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 fixedlengthJob. You may also check the relevant data are modified when the Job is implemented.



References

- FlatFileItemReader
- FlatFile ItemWriter