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

Retry Example

Outline

The Retry Example demonstrates re-processing of data turned out to be failed while the data is processed. See the EgovRetrySampleFunctionalTests example for how to implement retry as per the predetermined configuration under ItemProcessing and ItemWriting. Retry is permitted until reaching the retry-limit to reduce likelihood of the job failure.

Description

Settings

Configuring Jobs

Check retrySample.xml, the Job configuration file for the Retry Example.

Refer to the following for chunk configuration in Job composition:

- retry-limit : The maximum retry count
- <retryable-exception-classes> : Range of the Exception thrown. Note that too frequent retry gives rise to roll-back that affect speed.

<include>: The exceptions to retry.
<exlclude>: The exceptions that won't actuate Retry among sub-exceptions.

```
<job id="retrySample" xmlns="http://www.springframework.org/schema/batch">
	<step id="step1">
		<tasklet>
			<chunk reader="itemGenerator" writer="itemWriter"
				commit-interval="1" retry-limit="3">
			<retryable-exception-classes>
				<include class="java.lang.Exception"/>
			</retryable-exception-classes>
			</chunk>
		</tasklet>
	</step>
</job>
```

Composition and Implementation of JunitTest

Composition of JunitTest

Work Junit Test that comprises retrySample configuration and relevant classes, where batch implementation is involved and the associated contents are viewed.

✔ See Junit Test Description for Batch Execution Environment for structure of the Class JunitTest.
✔ assertEquals(itemGenerator.getLimit()+2, itemProcessor.getCounter()) : A total of two retries are to be attempted. You need to make sure the entire process count should be set greater by two than what is read in reader.

```
@ContextConfiguration(locations = { "/egovframework/batch/simple-job-launcher-context.xml",
		"/egovframework/batch/job-runner-context.xml",
		"/egovframework/batch/jobs/retrySample.xml"})
public class EgovRetrySampleFunctionalTests {

	@Test
	public void testLaunchJob() throws Exception {
		jobLauncherTestUtils.launchJob();
```

```
                //items processed = items read + 2 exceptions
                assertEquals(itemGenerator.getLimit()+2, itemProcessor.getCounter());
        }
}
```

Implementation of JunitTest

See Implementation of JunitTest for more information.

References

- Retry

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

Repository Example

Outline

Refer to the following example for how to configure data processing (DB, File, etc.) when implementing the batch. See the example EgovRepositoryFunctionalTests for how to implement jdbcCursorIoJob and delimitedIoJob, where the resources comprise DBs and Files, respectively.

Description

Settings

Configuring Jobs

Check repositoryJob.xml, the Job configuration file for the Repository Example.

A Job is comprised of a pair of sub-jobs that uses either reader or writer depending on the type.

- jdbcCursorIoJob : A Job where resources comprise DBs
- delimitedIoJob : A Job where resources comprise Files

```
<!-- Register Jobs where Resources comprise DBs, in XML -->
<job id="jdbcCursorIoJob" xmlns="http://www.springframework.org/schema/batch">
  <step id="jdbcCursorIoStep1">
    <tasklet>
      <chunk reader="itemReaderDB" processor="itemProcessor" writer="itemWriterDB" commit-interval="2" />
    </tasklet>
  </step>
</job>

<!-- Register Files where Resources comprise Files, in XML or JobParameter -->
<job id="delimitedIoJob" xmlns="http://www.springframework.org/schema/batch">
  <step id="delimitedIostep1">
    <tasklet>
      <chunk reader="itemReaderFile" processor="itemProcessor" writer="itemWriterFile" commit-interval="2"/>
    </tasklet>
  </step>
</job>
```

Composition and Implementation of JunitTest

Composition of JunitTest

Work Junit Test that comprises repositoryJob configuration and relevant classes, where batch implementation is involved and the associated contents are viewed.

✔ See Junit Test Description for Batch Execution Environment for structure of the Class JunitTest.
✔ assertEquals(BatchStatus.COMPLETED, jobExecution.getStatus()) : Make sure you check the batch execution result is COMPLETED.
✔ jobLauncher.run(jobRegistry.getJob("jdbcCursorIoJob"), getUniqueJobParameters("jdbcCursorIoJob")) : Executes the job entitled jdbcCursorIoJob
✔ getUniqueJobParameters hands off the input and output resources required for JobParameter batches

```
@ContextConfiguration(locations = {
                "/egovframework/batch/simple-job-launcher-context.xml",
```

```
                "/egovframework/batch/jobs/repositoryJob.xml" })
public class EgovRepositoryFunctionalTests {

        @Test
        public void testUpdateCredit() throws Exception {
                // 1. Implementation of Jobs where resources comprise DBs
                JobExecution jobExecution =
jobLauncher.run(jobRegistry.getJob("jdbcCursorIoJob"),getUniqueJobParameters("jdbcCursorIoJob"));
                assertEquals(BatchStatus.COMPLETED, jobExecution.getStatus());

                // 2. Implementation of Jobs where resources comprise Files
                jobExecution =
jobLauncher.run(jobRegistry.getJob("delimitedIoJob"),getUniqueJobParameters("delimitedIoJob"));
                assertEquals(BatchStatus.COMPLETED, jobExecution.getStatus());
        }

        /**
         * Method to configure JobParameter
         */
        protected JobParameters getUniqueJobParameters(String jobName) {

                JobParametersBuilder builder = new JobParametersBuilder();
                // Job Parameters Registrable
                builder.addString("inputFile",  "/egovframework/data/input/delimited.csv");
                builder.addParameter("timestamp", new JobParameter(new Date().getTime()));

                JobParameters jobParameters = builder.toJobParameters();
                return jobParameters;
        }
}
```

Implementation of JunitTest

See Implementation of JunitTest for more information.

Verify Result

- You can check out the data processing result where the Repository Type is DB, in the Table CUSTOMER.

Status | Result1

	ID	VERSION	NAME	CREDIT
1	1	0	customer1	100005
2	2	0	customer2	100005
3	3	0	customer3	100005
4	4	0	customer4	100005

- delimitedOutput.csv 파일을 보면 Repository 타입이 File 인 데이터 처리 결과를 확인할 수 있다.

	A	B	C	D	E
1	customer1	15			
2	customer2	25			
3	customer3	35			
4	customer4	45			
5	customer5	55			
6	customer6	65			
7					

References

- ItemReader
- ItemWriter

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

Work Group Example

Outline

When implementing batches, you need to categorizes a series of jobs into the proper unit as they become necessary.

Description

Settings

Configuring Jobs

In the Workgroup Example, you may use delegatingJob used in other examples.

Refer to the following for chunk configuration in Job composition:

```
<job id="delegateJob" xmlns="http://www.springframework.org/schema/batch">
	<step id="delegateStep1">
		<tasklet>
			<chunk reader="reader" writer="writer" commit-interval="3"/>
		</tasklet>
	</step>
</job>
```

Configuring Launcher

Check out group-job-launcher-context, the launcher configuration file for work group example.

✔ To use the Job Group Function you need to add the group entitled ‘groupName’ for the Bean related to the Class JobRegistryBeanPostProcessor.

```
<bean id="jobLauncher"	class="org.springframework.batch.core.launch.support.SimpleJobLauncher">
	<property name="jobRepository" ref="jobRepository" />
</bean>

<bean class="org.springframework.batch.core.configuration.support.JobRegistryBeanPostProcessor">
	<property name="jobRegistry" ref="jobRegistry"/>
	<property name="groupName" value="testJobGroup"/>
</bean>

<bean id="jobRepository" class="org.springframework.batch.core.repository.support.JobRepositoryFactoryBean"
	p:dataSource-ref="dataSource" p:transactionManager-ref="transactionManager" p:lobHandler-
ref="lobHandler"/>

<bean id="lobHandler" class="org.springframework.jdbc.support.lob.DefaultLobHandler"/>

<bean id="mapJobRepository"
class="org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean"
	lazy-init="true" autowire-candidate="false" />

<bean id="jobOperator" class="org.springframework.batch.core.launch.support.SimpleJobOperator"
	p:jobLauncher-ref="jobLauncher" p:jobExplorer-ref="jobExplorer"
	p:jobRepository-ref="jobRepository" p:jobRegistry-ref="jobRegistry" />

<bean id="jobExplorer" class="org.springframework.batch.core.explore.support.JobExplorerFactoryBean"
```

```
        p:dataSource-ref="dataSource" />

<bean id="jobRegistry"        class="org.springframework.batch.core.configuration.support.MapJobRegistry" />
```

Composition and Implementation of JunitTest

Composition of JunitTest

Work Junit Test using the delegatingJob and group-job-launcher-context configurations, where batch implementation is involved.

✔ See Junit Test Description for Batch Execution Environment for JunitTest class structure.
✔ assertEquals("[testJobGroup.delegateJob]",jobRegistry.getJobNames().toString()) : Make sure you configure the title of Job in the form of [title of Group, title of Job].

```
@ContextConfiguration(locations = { "/egovframework/batch/group-job-launcher-context.xml",
                                    "/egovframework/batch/jobs/delegatingJob.xml",
                                    "/egovframework/batch/job-runner-context.xml" })
public class EgovJobGroupDelegatingJobFunctionalTests {
        ...
        @Test
        public void testLaunchJob() throws Exception {
                jobLauncherTestUtils.launchJob();
                assertTrue(personService.getReturnedCount() > 0);
                assertEquals(personService.getReturnedCount(), personService.getReceivedCount());
                assertEquals("[testJobGroup.delegateJob]",jobRegistry.getJobNames().toString());
        }
}
```

Implementation of JunitTest

See Implementation of JunitTest for more information.

Verify Result

Check out the log in the console to see the groupName(testJobGroup) assigned in the launcher configuration are used as the title of categories.

```
SVN Repositories | Problems | Servers | Call Hierarchy | Type Hierarchy | SQL Results | Query Result | Synchronize | Progress | Search | Outline | JUnit | Console
<terminated> EgovJobGroupDelegatingJobFunctionalTests [JUnit] C:₩eGovFrame-2.0₩bin₩jdk1.5.0_22₩bin₩javaw.exe (2012. 9. 24. 오후 5:23:56)
17:24:01,727  INFO main audit:156 - 2. Connection.setAutoCommit(true) returned
17:24:01,727  INFO main audit:156 - 2. Connection.isReadOnly() returned false
17:24:01,727  INFO main audit:156 - 2. Connection.isClosed() returned false
17:24:01,728  INFO main audit:156 - 2. Connection.getAutoCommit() returned true
17:24:01,728  INFO main audit:156 - 2. Connection.clearWarnings() returned
17:24:01,729  INFO main audit:156 - 2. Connection.getAutoCommit() returned true
17:24:01,737  INFO main SimpleJobLauncher:121 - Job: [FlowJob: [name=testJobGroup.delegateJob]] completed with the following parameters: [{timestamp=1348475040996}]
17:24:01,742  INFO Thread-2 GenericApplicationContext:1002 - Closing org.springframework.context.support.GenericApplicationContext@9f5011: startup date [Mon Sep 24
17:24:01,742  INFO Thread-2 DefaultListableBeanFactory:422 - Destroying singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@1f4e571:
17:24:01,758  INFO Thread-2 audit:156 - 2. Connection.close() returned
17:24:01,758 DEBUG Thread-2 JobRegistryBeanPostProcessor:113 - Unregistering job: testJobGroup.delegateJob
```

References

- JobRegistry