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

Batch Runner Practices

Outline

This Section describes how to use EgovBatchRunner in execution of batches. You need to run, stop and re-run the asynchronous jobs.

Description

Settings

Configuring EgovBatchRunner

See Batch Runner for how to configure EgovBatchRunner.

Configuring Launcher

Check batchRunnerTest.xml, the job configuration file for Batch Runner examples.

The example assumes asynchronous job and designates the Class SimpleAsyncTaskExecutor to TaskExecutor of JobLauncher to do so.

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

Configuring Jobs

Check batchRunnerTest.xml, the job configuration file for Batch Runner examples.

✔ When re-run, a job uses JobParametersIncrementer to use the previous JobParameter as is.
✔ simpleJob is inherited for use of JobParametersIncrementer in batchRunnerTestJob. Restartable in simpleJob is configured ‘true’.
✔ Step inherited simpleStep as well.

```
<bean id="simpleJob" class="org.springframework.batch.core.job.SimpleJob" abstract="true">
	<property name="jobRepository" ref="jobRepository" />
	<property name="restartable" value="true" />
</bean>

<bean id="simpleStep" class="org.springframework.batch.core.step.item.FaultTolerantStepFactoryBean"
abstract="true">
	<property name="transactionManager" ref="transactionManager" />
	<property name="jobRepository" ref="jobRepository" />
</bean>

<bean id="batchRunnerTestJob" parent="simpleJob">
	<property name="jobParametersIncrementer">
		<bean class="org.springframework.batch.core.launch.support.RunIdIncrementer" />
	</property>
	<property name="steps">
		<bean id="infiniteStep" parent="simpleStep">
```

```
                <property name="itemReader">
                        <bean id="reader"
class="egovframework.brte.sample.common.domain.trade.GeneratingTradeItemReader">
                                <property name="limit" value="1000000" />
                        </bean>
                </property>
                <property name="itemWriter">
                        <bean id="writer"
class="egovframework.brte.sample.example.support.EgovDummyItemWriter" />
                </property>
            </bean>
        </property>
</bean>
```

Composition and Implementation of JunitTest

Composition of JunitTest

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

See the following for how batch is implemented:

1. The title of job and JobParameter is transferred to the Method start () of EgovBatchRunner and earns executionId.
2. Use executionId to check for the identity between the title of Job and JobParameter.
3. Use stopAndCheckStatus() to suspend the jobs underway.
4. The executionId for the job initiated in the Method restart() for EgovBatchRunner to reryb batchRunnerTestJob.
5. Use stopAndCheckStatus() again to suspend the jobs underway.

```
@Test
public void testStartStopResumeJob() throws Exception {

        String jobName = "batchRunnerTestJob";
        String jobParameters = egovBatchRunner.createUniqueJobParameters();

        long executionId = egovBatchRunner.start(jobName, jobParameters);

        assertEquals(jobName, egovBatchRunner.getJobInstance(executionId).getJobName());
        assertEquals(jobParameters.toString(),
egovBatchRunner.getJobOperator().getParameters(executionId));
        stopAndCheckStatus(executionId);

        long resumedExecutionId = egovBatchRunner.restart(executionId);
        assertEquals(jobParameters.toString(),
egovBatchRunner.getJobOperator().getParameters(resumedExecutionId));
        stopAndCheckStatus(resumedExecutionId);

}
```

Use stopAndCheckStatus() to check the execution status of job and suspend the jobs underway.

1. Use Thread.sleep() for one-second hiatus to secure the asynchronous execution of jobs.
2. Check the execution information, use the Method stop() of EgovBatchRunner and suspend the jobs.
3. Check the executionId of the jobs underway and stand by for 0.1 sec.
4. Print out the summary of the concerned Job and check if BatchStatus is Stopped.

```
private void stopAndCheckStatus(long executionId) throws Exception {
        String jobName = egovBatchRunner.getJobInstance(executionId).getJobName();

        // wait to the job to get up and running
        Thread.sleep(1000);
```

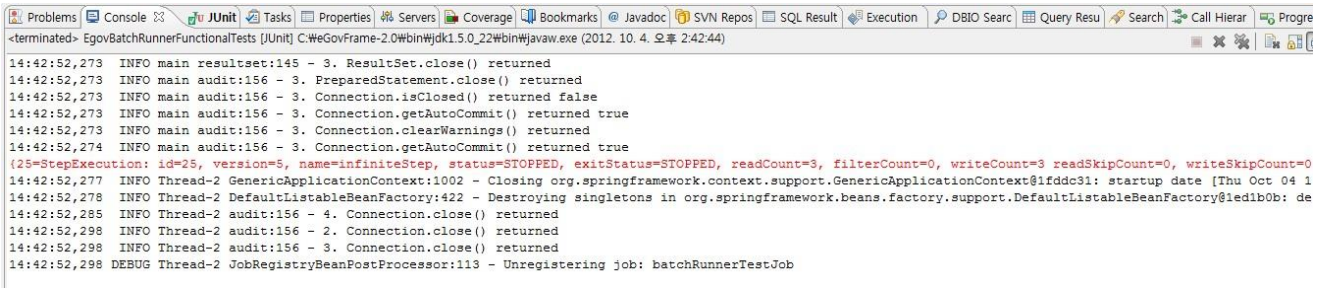
```
                Set<Long> runningExecutions =
egovBatchRunner.getJobOperator().getRunningExecutions(jobName);
                assertTrue("Wrong executions: " + runningExecutions + " expected: " + executionId,
runningExecutions
                        .contains(executionId));
                assertTrue("Wrong summary: " + egovBatchRunner.getJobOperator().getSummary(executionId),

        egovBatchRunner.getJobOperator().getSummary(executionId).contains(BatchStatus.STARTED.toString()));

                egovBatchRunner.getJobOperator().stop(executionId);

                int count = 0;
                while
(egovBatchRunner.getJobOperator().getRunningExecutions(jobName).contains(executionId) && count <= 10)
                        {  logger.info("Checking for running JobExecution: count=" + count);
                        Thread.sleep(100);
                        count++;
                }

                runningExecutions = egovBatchRunner.getJobOperator().getRunningExecutions(jobName);
                assertFalse("Wrong executions: " + runningExecutions + " expected: " + executionId,
runningExecutions
                        .contains(executionId));
                assertTrue("Wrong summary: " + egovBatchRunner.getJobOperator().getSummary(executionId),

        egovBatchRunner.getJobOperator().getSummary(executionId).contains(BatchStatus.STOPPED.toString()));

                // there is just a single step in the test job
                Map<Long, String> summaries =
egovBatchRunner.getJobOperator().getStepExecutionSummaries(executionId);
                System.err.println(summaries);
                assertTrue(summaries.values().toString().contains(BatchStatus.STOPPED.toString()));
        }
```

Implementation of JunitTest

See Implementation of JunitTest for more information.

Verify Result

Check if BatchStatus is Stopped in Job Summary of Console.



References

- Batch Runner