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

JobRepository

Outline

In JobRepository the in-batch processing metadata are reposited, logging the execution history of jobs, when the jobs are ended, how many times the jobs are executed and the result of execution.

Description

Being a persistent mechanism that reposits the information on jobs executed, JobRepository is used for the fundamental CRUD services for any type of information that features persistence, such as JobExecution and StepExecution. When a batch in Springs is executed initially, you need to reposit the information from StepExecution and JobExecution in JobRepository for it to be automatically renewed and persist.

Refer to the following for how JobRepository can be configured using batch's Namespace or the class JobRepositoryFactoryBean:

```
<job-repository id="jobRepository"
      data-source="dataSource"
      transaction-manager="transactionManager"
      isolation-level-for-create="SERIALIZABLE"
      table-prefix="BATCH_"
      max-varchar-length="1000"
/>
<bean id="jobRepository" class="org.springframework.batch.core.repository.support.JobRepositoryFactoryBean"
                        p:dataSource-ref="dataSource" p:transactionManager-ref="transactionManager" p:isolation-
level-for-create="ISOLATION_DEFAULT"
                       p:table-prefix="BATCH_" p:max-var-char-length="1000"/>
```

Transaction Configuration for the JobRepository

When using Napespace, the transactional advices are automatically generated around the repository to check for the proper state of metadata persisting to determine resumption of the failed batch processing. Note that the framework processing is not defined well when the repository does not transact. The fundamental isolation level is serializable at the highest isolation level and redefinableas well.

```
<job-repository id="jobRepository"
                        isolation-level-for-create="REPEATABLE_READ" />
```

When you do not use Namespace or Bean definition, you need to use AOP to manage repository transactions, as follows:

```
<aop:config>
      <aop:advisor
                pointcut="execution(* org.springframework.batch.core..*Repository+.*(..))"/>
      <advice-ref="txAdvice" />
</aop:config>

<tx:advice id="txAdvice" transaction-manager="transactionManager">
      <tx:attributes>
            <tx:method name="*" />
      </tx:attributes>
</tx:advice>
```

You can copy-and-paste the foregoing configuration for most of the time, provided you have secured Spring-aop and Spring-tx are available in the classpath or Namespace declared.

Changing the Table Prefix

In JobRepository you can change the prefixes of the metadata table. With the data tables commencing with the prefix BATCH_ (BATCH_JOB_EXECUTION and BATCH_STEP_EXECUTION) you need to change the table prefix to add the schema before the table prefix or a set of metadata tables within a schema.

```
<job-repository id="jobRepository"
                        table-prefix="SYSTEM.TEST_" />
```

With the foregoing configuration established, the title of metadata table now commences with SYSTEM.TEST_. BATCH_JOB_EXECUTION will be changed into SYSTEM.TEST_JOB_EXECUTION as well.

✔ Keep in mind you can only change the prefix of tablem not the title and column of table.

In-Memory Repository

In Spring you can define jobRepository in the form of in-memory, not database to avoid latency when committing the domain objects in database. To do so, you need to execute the job via in-memory repository as follows:

```
<bean id="jobRepository"
   class="org.springframework.batch.core.repository.support.MapJobRepositoryFactoryBean">
      <property name="transactionManager" ref="transactionManager"/>
</bean>
```

Note the in-memory is volatile, does not permit JVM instance to re-start and is unable to guarantee concurrent execution of multiple instances in a single parameter, making it not feasible for multi-threaded batch or partition. You'll thus need to use database for JobRepository just like that.

With repository supports rollback, you need to configure the transaction manager in a way that supports tests as well. ResourcelessTransactionManager thus comes in handy.

Non-standard Database Types in a Repository

When you intend to use a database that is not supported in Spring Batch, you need to use JobRepositoryFactoryBean instead of Namespace to configure the specific type of database that is deemed most closely associated with the concerned database.

- Spring Batch supports databases such as : DERBY, DB2, DB2ZOS, HSQL, SQLSERVER, MYSQL, ORACLE, POSTGRES, SYBASE and H2

```
<bean id="jobRepository" class="org.springframework.batch.core.repository.support.JobRepositoryFactoryBean">
      <property name="databaseType" value="db2"/>
      <property name="dataSource" ref="dataSource"/>
</bean>
```

Caution:

✔ Note that **altibase** and **tibero** are not supported. You'll thus need to set 'oracle' for databaseType in jobRepository to use either altibase or tibero.

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

References

- JobRepository :http://static.springsource.org/spring-batch/reference/html/configureJob.html#configuringJobRepository
- declarative_transaction_management
- Meta-Data Schema :http://static.springsource.org/spring-batch/reference/html/metaDataSchema.html