Removing redundant processes from the NJProcess table

April 18, 2017

10 York Mills Road, Suite 700
Toronto, ON, M2P 2G4
Tel: (416) 222-5611
Fax: (416) 222-8623

Contents

Removing redundant processes from the NJProcess table 1

Removing redundant processes from the NJProcess table

The NJProcess table stores the states of NexJ processes that occur periodically. The current state of a process also displays in the **Process** page in Admin Console.

In NexJ Studio, you can create a batch job that will periodically clean up old processes. The batch job must define which processes to delete and must invoke the `ETLDelete` ETL activity to ensure the NJProcess table and all related tables are also cleaned up. Consider the following when creating the batch job:

- Specify that the batch job delete only completed, cancelled, failed, or completed with errors processes. Running, paused, or waiting processes should not be deleted.
- Use a threshold to delete only processes that are older than a specific time period, for example processes that are older than 12 months.

The following sample Scheme code deletes processes that are older than 1 year and are completed, cancelled, failed, or completed with errors.

```
(define threshold (date-add-years (now) -1))

(define where
   `(and
      (< (@ createTime) ,threshold)
      (in?
         (@ status)
         ,(ProcessStatusEnum'get 'COMPLETED)
         ,(ProcessStatusEnum'get 'CANCELLED)
         ,(ProcessStatusEnum'get 'FAILED)
         ,(ProcessStatusEnum'get 'ERRORS)
      )
   )
)



(begin-privileged
   (SysETLActivity'invoke "ETLDelete" () () SysProcess SysProcess where '())
)
```

To create the batch job:

1. In NexJ Studio, create a new class.
2. In the **Overview** tab, define `SysBatchJob` as the base class of the new class.
3. Override the run event and add a main action that contains the Scheme code that you created earlier:
    a. Open the **Events** tab. On the top half of the screen, next to the **Events** list, click the **Override Base Events** button .

b. In the **Override Base Events** dialog, add the `run` event.

c. In the **Events** tab, in the **Actions** subtab, click the **Add** button and select **main**.

d. In the **Script** subtab, insert your Scheme code into the text field below the **Method** field.

4. Specify how often to run the batch job:

a. Open the **Attributes** tab. On the top half of the screen, next to the **Attributes** list, click the **Override Base Attributes** button.

b. In the **Override Base Attributes** dialog, add the `period` event.

c. In the **Attributes** tab, in the **Value** subtab, in the **Value** field, specify the number of minutes between each invocation of the batch job. For example, specify `10080` to create a weekly batch job.

5. Seed the batch job:

a. In the **Overview** tab, click the **Select** button in the **Aspects** field.

b. In the **Select Aspects** dialog, include the `PERSISTENCE_INITIALIZER` aspect.

c. Open the **Events** tab. On the top half of the screen, next to the **Events** list, click the **Override Base Events** button.

d. In the **Override Base Events** dialog, add the `initializePersistence` event.

e. In the **Events** list, select the `initializePersistence` event.

f. In the **Events** tab, in the **Actions** subtab, click the **Add** button and select **main**.

g. In the **Script** subtab, insert Scheme code that seeds the batch job in the **Condition** and **Method** fields. For example:
In **Condition**, enter `(null? (read-instance ProcessBatchJob '() '() #f))`
In **Method**, enter
```
ProcessBatchJob'new)
(pre-commit)
```

6. Save and close the new class.

The batch job will be seeded the next time the server starts if the server is set to reseed.