Author – Kuldeep Saxena Mail to – kuldeep.saxena@paxcel.net

Paxcel Technology Pvt Ltd.

MX Executor

What is it? –

This is designed to provide monitor and management capability to Java runnable tasks/Threads. This wraps normal java executor to provide various features listed below -

Features –

It provides many features which can be controlled by any jmx client or directly by components using it –

- *Execute a task (Java Runnable) as with old executors*
- *Force/Normal Pause/resume a registered task at runtime*
- *Force/Normal stop/restart a registered task at runtime*
- *Schedule a registered task at runtime, remove schedule, activate/in-activate schedule at runtime*
- *Provide Thread (Task) state (INITIALIZED, STARTED, WAITING, PAUSED, FORCE_PAUSED, STOPPED, FORCE_STOPPED, RESUMED, RUNNING, REMOVED, UNKNOWN) by providing Notification API*
- *Execution Count (how many times) task has been executed since start*
- *Wait Interval configuration for task at runtime (Time to wait for next execute of your task)*
- *Time spend in last execution of task (number of milliseconds spend by your task on a cycle of execution)*
- *CPU utilization report for individual thread.*
- *Overall monitoring/control for all tasks executed using this executor, one click pause/stop/start of all threads within the executor*

How to use –

Sample Hello World –

See sample under package net.paxcel.labs.mxexecutableservice.test

```
final MXExecutor executor = MXExecutors.newCachedScheduledServiceExecutor("Scheduled Service Test Executor");

                executor.addTaskStateListener(new MXExecutorListener() {

                        @Override
                        public void onStateChanged(String serviceId, MXExecutableTaskState newState,
    int executionCount) {
                                System.out.println("New State for Service " + serviceId + " = "
    + newState +" Count " + executionCount);
                        }
                });
```

```
                try {
                        executor.execute("Service 1", new HelloWorldExecutableTask(), -1, 20000);
                } catch (MXTaskException e1) {
                        e1.printStackTrace();
                }
                try{
                        Thread.sleep (5000);
                }catch (Exception e){

        }
```

Call is similar to call you does for java executors Executors.newXXXX (); - difference in call to create MX Executor is parameter passed to it, It is used as domain name in JMX to merge your tasks (runnable) in same groups.
Now you have Executor instance – next to run your runnable tasks, in normal executor you need to do executor.execute (Runnable);

You can use same call as well as provide a name of thread using below call -

```
executor.execute("Service 1", new HelloWorldExecutableTask());
```

Here first parameter is a unique name for thread within current executor. Second parameter is Runnable instance – which is your runnable instance.

You can add listener to this executor to listen for state (as mentioned above – task states) event.

Listener registration call is also very simple. Following is a sample –

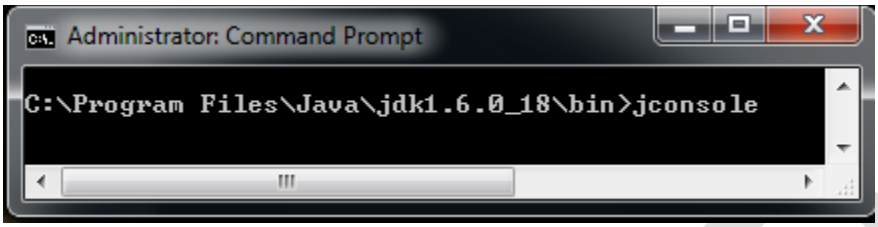
```
executor.addScheduledServiceListener(new MXExecutorListener() {

                @Override
                public void onStateChanged(String serviceId, MXExecutableTaskState newState,
int executionCount) {


                                        System.out.println("New State for Service " + serviceId + " = "
+ newState +" Count " + executionCount);
                }
        });
```

Monitor and control using JMX Client -

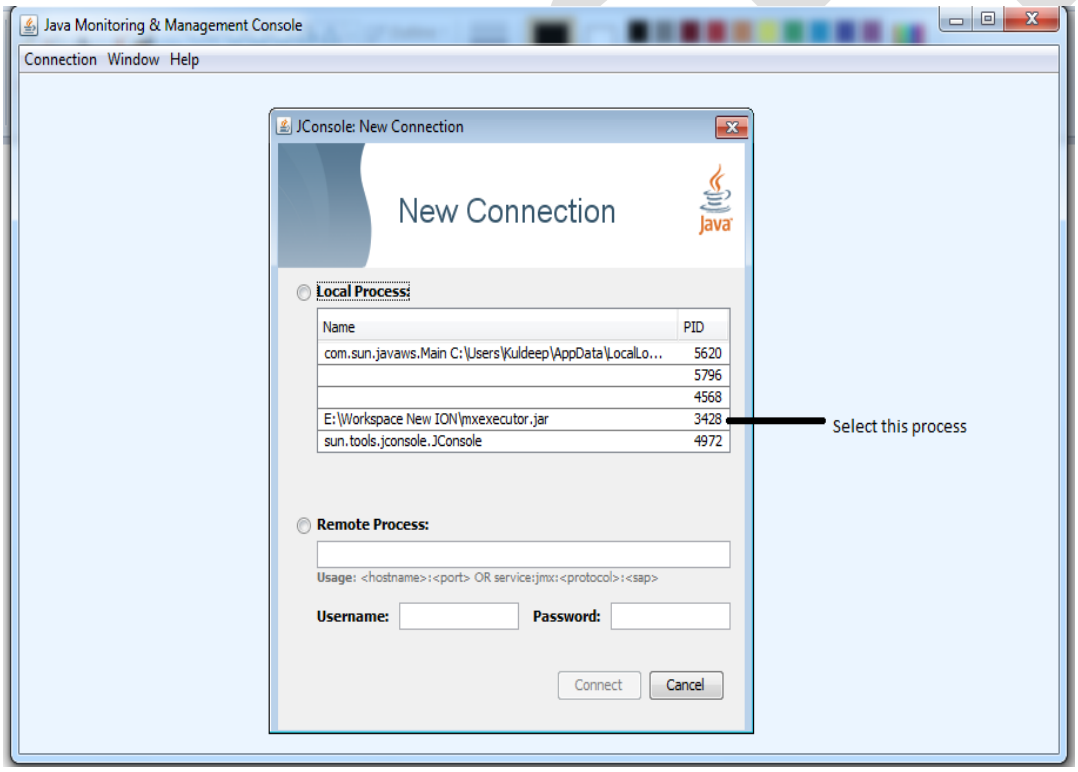
Next section show JConsole (JMX) client for sample code we written above – Yup just above line of code and when you run your program, you can easily start jconsole and in MBean section can monitor and manage executor. To

make things easier for simple test, just run (double click) mxexecutor.jar file in JVM, and then Start JConsole (It is executing sample Hello World Application attached with this jar) to see Mbean.

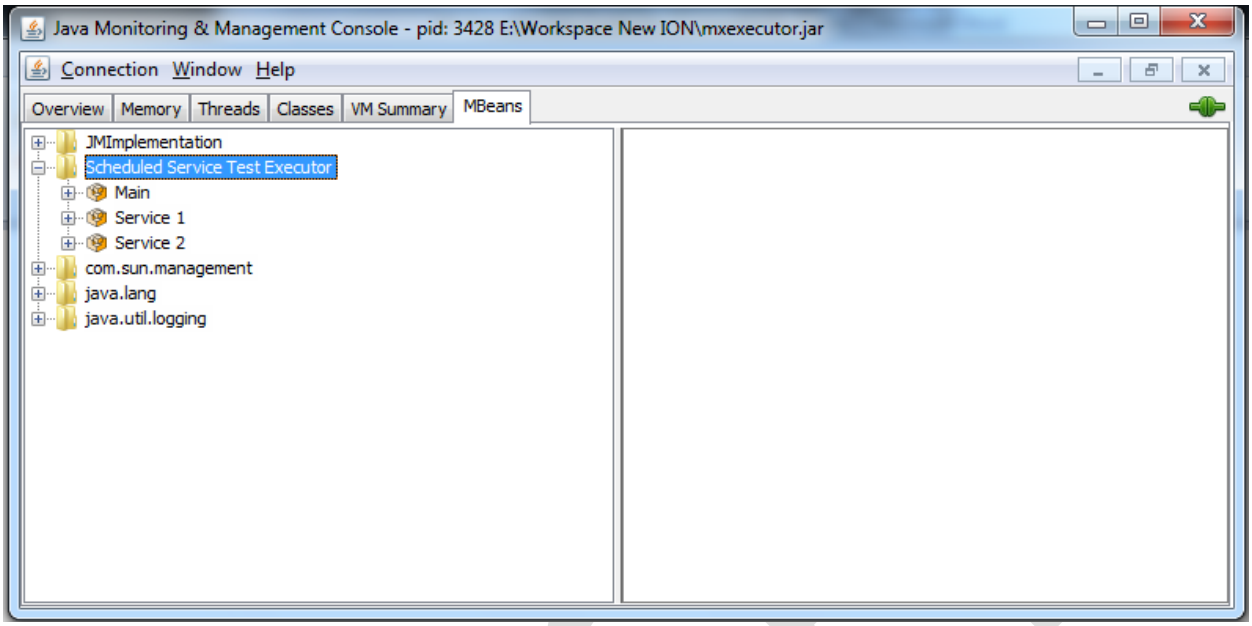
So as first step run that jar and then start jconsole provided with jdk.



This will bring up JConsole main screen –



On Mbean Tab you will see MBean for your executor (Scheduled Service Test Executor), with 3 child nodes -

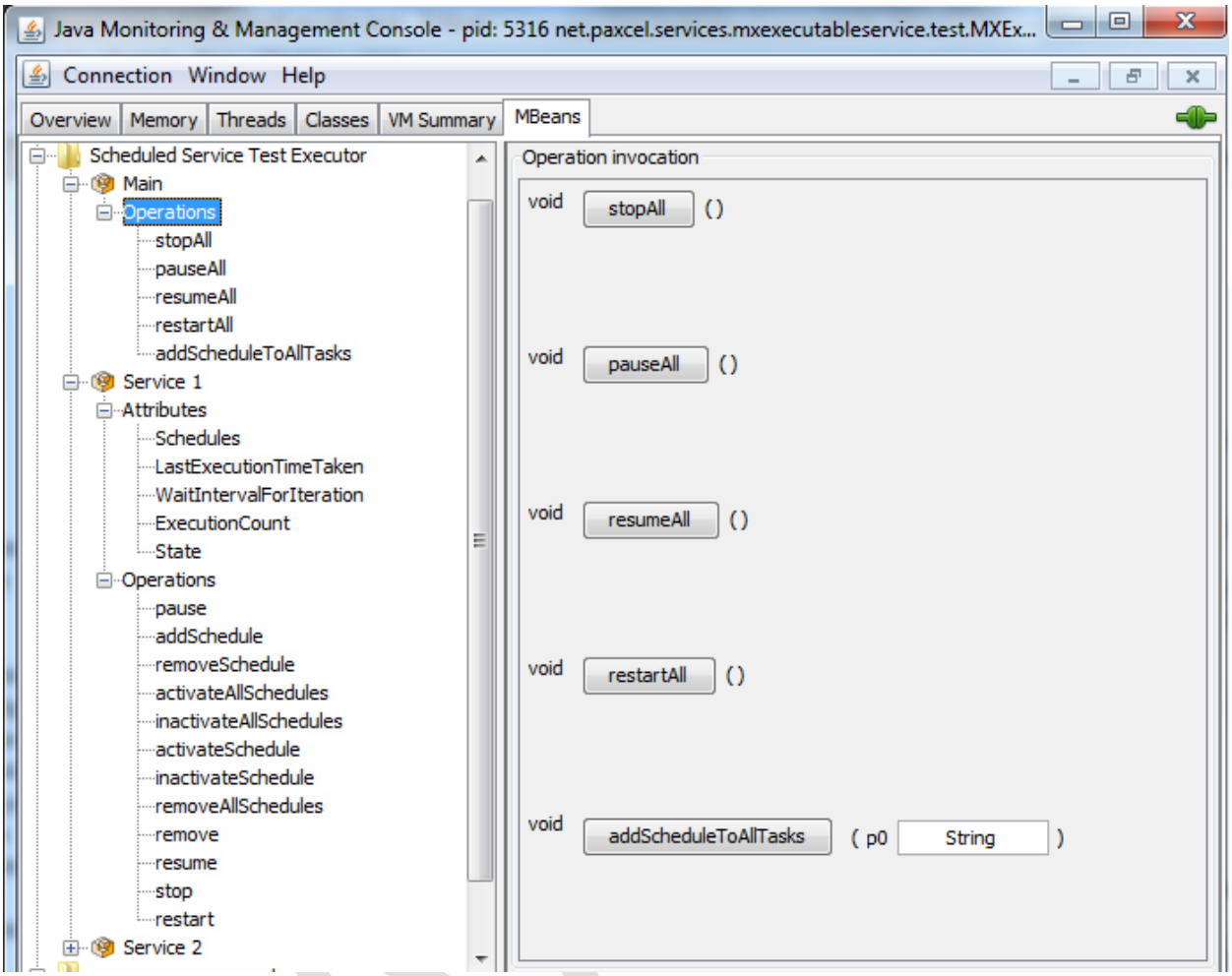


Main – is MBean to control all Tasks (Runnable) of this executor.

Service 1 – Is our Runnable Task (we provided name = Service and Id = 1)

Service 2 – Is our second Runnable Task (we provided name = Service and Id = 2)

Expand main and any one of service node –



In above figure Operations within Main (MBean) are shown at right pane –

stopAll - Stops all tasks (Threads) running in this executor

pauseAll – Pause all tasks (Threads) running in this executor

resumeAll – Resumes all tasks (paused) in this executor

addScheduleToAllTasks – Add schedules to all tasks running in this executor. – Schedule value format is as below –

(HH?MM?SS?HH?MM?SS?DD,DD...

Where HH is 24 hr format hr (<=24)

MM is minutes (<= 60)

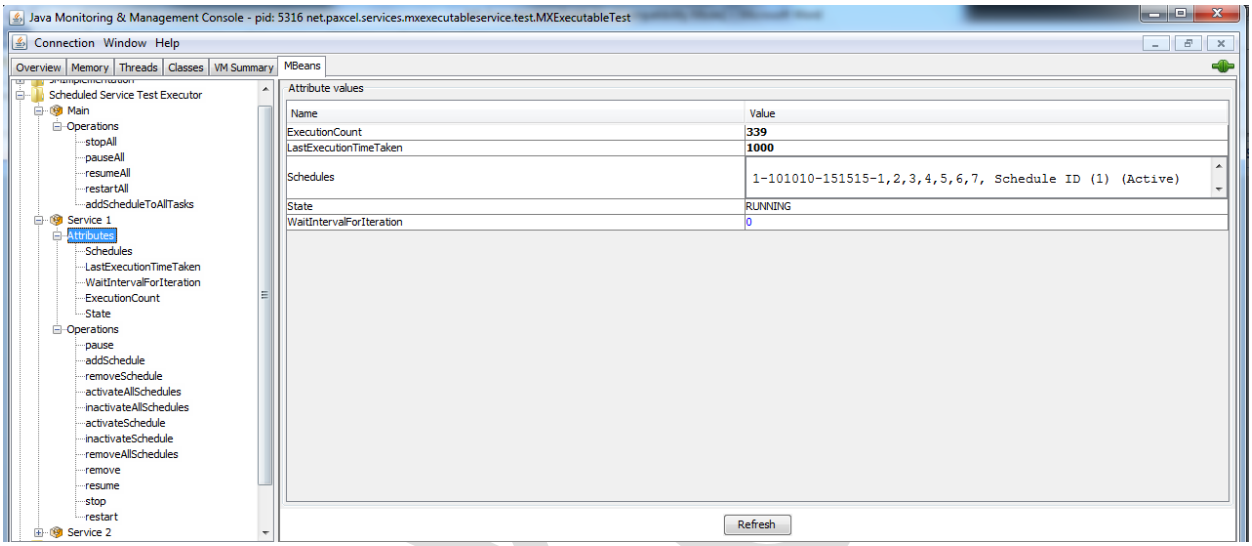
SS is seconds (<=60)

DD is day (between sunday to saturday) - 1 = Sunday (same as java {@link Calendar} and so on, you can provide multiple values separated by (,)

** ? is any character*

** First there combincation HH?MM?SS is for start and next HH?MM?SS for end time*

Now go to Service node and select Attributes –



It has 5 attributes shown to you

Execution Count – (339 in above image) – is number of times this task already ran

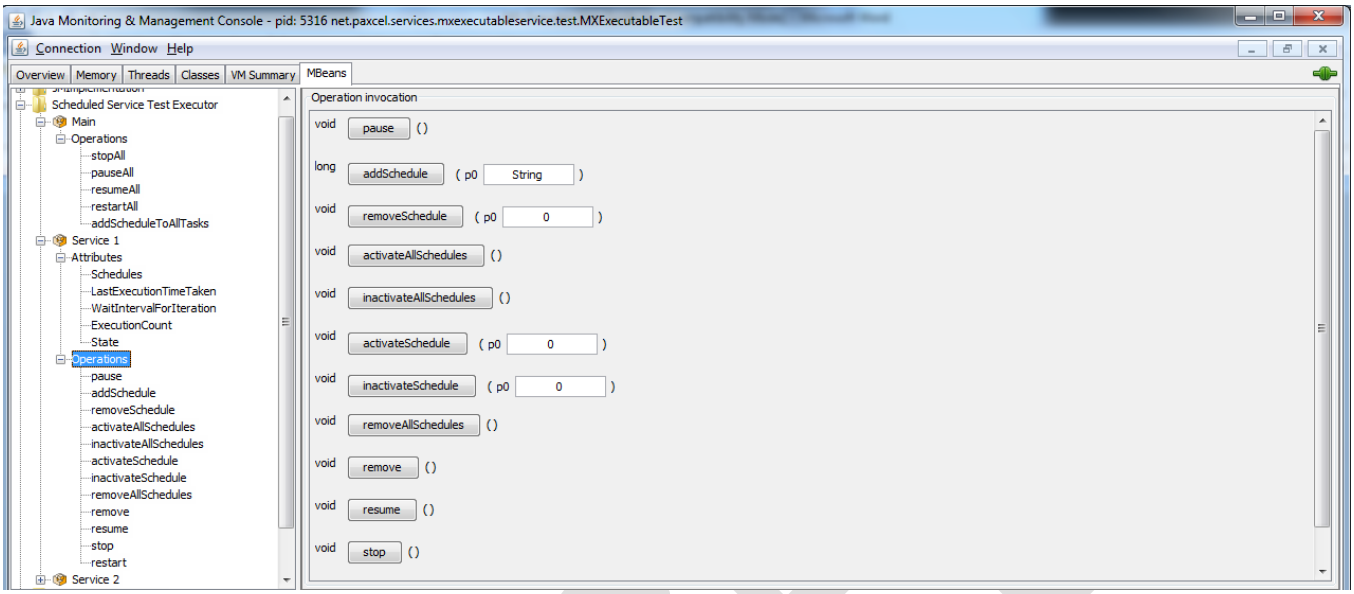
LastExecutionTimeTaken – (1000 in above image) – is number of milliseconds your task taken in recent execution

Schedules – (1-101010-151515-1,2,3,4,5,6,7, Schedule ID (1) (Active) in above figure) is schedule value (Task Id – HHMMSS-HHMMSS-DD,DD,DD,DD,DD,DD,DD, Schedule Id (ID) (Active – schedule can be activated and inactivated.)

State – (RUNNING in above figure) – is task state

WaitIntervalForIteration – Wait for thread for subsequent execution (0 in above figure)

Now Explore Operations Section by selecting it –



It shows operations you can perform on your task to control it at runtime-

1) *pause – pause the task*
2) *addSchedule - add a schedule to the task (10 10 10 15 15 15 1,2,3,4,5,6,7 was our input which you see in schedules section above)*
3) *removeSchedule – removes schedule (you need to provide id which was returned to you at time of schedule addition, 1 was in our case)*
4) *activateAllSchedules – activate all inactivated schedules*
5) *inactivateAllSchedules – inactivate all active schedules*
6) *activateSchedule – activate given schedule*
7) *inactivateSchedule – inactivate given schedule*
8) *removeAllSchedules – removes all schedules*
9) *remove – stops and removes task*
10) *resume – resumes task if was paused earlier*
11) *stop – stops task*

MXBean Properties –

You can access MXBean using code API as well, MXBean Name details is below –

Main Executor MXBean Name - "<ExeutorName>:type=Main" (No 2 executor with same name can be created in same JVM)

Individual Thread MXBean Name – "<ExecutorName>:type=<ThreadName>" (No 2 thread with same name can exists in an executor) – If you used default version execute(Runnable) – Then name is auto assigned – which will be Thread-<auto-counter>.

Please refer to API Java docs for further details on using Application.

*Also it is worth to mention one point – It is highly recommended to **NOT** to use infinite while loops in your Runnable code. Above sample call (execute) does that for you.*

Contact Kuldeep Saxena on kuldeep.saxena@paxcel.net *or* saxena_knp@yahoo.com