Cello How-To Guide

Scheduler

techcello

Contents

1 Time based Scheduler

- CelloSaaS includes a feature rich job scheduler (Quartz Scheduler).
- You can schedule jobs to run at a designated date and time (such as every weeknight at 11:00pm), or upon the occurrence of a designated event (such as when inventory drops below a certain level).
- The Scheduler can also use to execute workflow process, which is named groups of steps that work together to accomplish a task.
- Steps in the workflow can be a program, sub chain or an event, and you specify rules that determine when each step runs and what the dependencies between steps are.
- In this tutorial you learn how to use the Scheduler to simplify the scheduling of complex tasks required for Multi-Tenant SaaS application developers.
- Steps need to be followed to get started with the Scheduler

    1) Write Jobs for your business application
    2) Register Time scheduler in Quartz
    3) Triggers
    4) Installation

1.1 Create Jobs

Implement the Job that inherits from `Quartz.IJob interface and write your business logic inside Execute method.`

Sample Code:

```
public class HelloJob : Quartz.IJob
{
       public void Execute(Quartz.JobExecutionContext context)
       {
              FluentConfiguration.Configure();

              //Get job inputs
              JobDataMap data = context.JobDetail.JobDataMap;

              // Your business logic logic
           }
        }
```

```
}

internal static class FluentConfiguration
{
 internal static void Configure()
 {
       if (CelloSaaS.Configuration.CelloConfigurator.IsConfigured)
       {
              return;
       }

       CelloSaaS.Configuration.CelloConfigurator.Reset();

       // Register the modules
       CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.Notification.Notifi
       cationModuleConfigurator>();
       CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.WorkFlow.WorkflowMo
       duleConfigurator>();
       CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.DataBackup.DataBack
       upModuleConfigurator>();
       CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.Integration.Integra
       tionModuleConfigurator>();
       CelloSaaS.Configuration.CelloConfigurator.RegisterModule<CelloSaaS.Configuration.DBCel
       loModuleConfigurator>();

       // Register the entities
       CelloSaaS.Configuration.CelloConfigurator.RegisterEntity<CelloSaaS.WorkFlow.WorkflowEn
       tityConfigurator>();
       CelloSaaS.Configuration.CelloConfigurator.RegisterEntity<CelloSaaS.Configuration.DBCel
       loEntityConfigurator>();

       // Register Application related configuration
       CelloSaaS.Configuration.CelloConfigurator.RegisterModule<AppModuleConfigurator>();
       CelloSaaS.Configuration.CelloConfigurator.RegisterEntity<AppEntityConfigurator>();

       CelloSaaS.Configuration.CelloConfigurator.Configure();
  }
}
```

1.2 Register Job in Quartz

1) Create a new Job instance for already created Job and add the job in database.

```
// Create a new Job
Job timeJob = new Job
{
       Name = "Time Job",
       CreatedBy = UserIdentity.UserId,
       JobType = "TestJobs.SampleJob, TestJobs.dll",
       Status = true
};

SchedulerService schedulerService = new SchedulerService();
// Add to database
string jobId = schedulerService.AddJob(timeJob);
```

2) Create time scheduler with newly created job. Add the Time scheduler in Quartz.

```
// create a new time schedule
TimeSchedule timeSchedule = new TimeSchedule
{
        JobId = jobId,
       JobReferenceName = "Test Job RefName",
       ScheduleId = "85E3577B-831D-4A8D-B034-60EBB161EF6B",
       StartDateTime = DateTime.UtcNow.AddSeconds(10),// start after 10 seconds
       EndDateTime = null,
       CreatedBy = UserIdentity.UserId,
       Status = true
};

// Create job parameters
JobParameter jobParameter = new JobParameter();
jobParameter.Key = "FileName";
jobParameter.Value = "SHAJ";

// Add Job Parameter to list
IList<JobParameter> jobParameters = new List<JobParameter>();
jobParameters.Add(jobParameter);

TimeSchedulerService timeSchedulerService = new TimeSchedulerService();
// Now add the time schedule and job parameters to the database
timeSchedulerService.AddTimeScheduleToQuartz(timeSchedule, jobParameters,
Quartz.SimpleTrigger.RepeatIndefinitely, TimeSpan.FromDays(1));
```

1.3 Triggers

Triggers can be created with nearly any combination of the following directives:

- at a certain time of day (to the millisecond)
- on certain days of the week
- on certain days of the month
- on certain days of the year
- not on certain days listed within a registered Calendar (such as business holidays)
- repeated a specific number of times
- repeated until a specific time/date
- repeated indefinitely
- repeated with a delay interval

Jobs are given names by their creator and can also be organized into named groups. Triggers may also be given names and placed into groups, in order to easily organize them within the scheduler. Jobs can be added to the scheduler once, but registered with multiple Triggers.

Here's a quick snippet of code, that instantiates and starts a scheduler, and schedules a job for execution:

```
// construct a scheduler factory
ISchedulerFactory schedFact = new StdSchedulerFactory();

// get a scheduler
IScheduler sched = schedFact.GetScheduler();
sched.Start();

// construct job info
JobDetail jobDetail = new JobDetail("SampleJob", null, typeof(HelloJob));

// fire every hour
Trigger trigger = TriggerUtils.MakeHourlyTrigger();

// start on the next even hour
trigger.StartTimeUtc = TriggerUtils.GetEvenHourDate(DateTime.UtcNow);
trigger.Name = "myTrigger";
sched.ScheduleJob(jobDetail, trigger);
```

For further details, please refer http://quartznet.sourceforge.net/features.html

1.4 Installation

1) Click install.bat file from the cellosaas_v4.3.2_developer_package\CelloSaaS Service Installers\Quartz folder to install Quartz.

2) Quartz will support both XML and SQL for register Jobs. CelloSaaS majorly supporting Database so comment the following section.
 # job initialization plugin handles our xml reading, without it defaults are used -->
#quartz.plugin.xml.type = Quartz.Plugin.Xml.JobInitializationPlugin, Quartz
#quartz.plugin.xml.fileNames = ~/quartz_jobs.xml

3) Uncomment the Database Jobs and change the connection string.
#Database Jobs
quartz.jobStore.type = Quartz.Impl.AdoJobStore.JobStoreTX, Quartz
quartz.jobStore.driverDelegateType = Quartz.Impl.AdoJobStore.StdAdoDelegate, Quartz
*quartz.jobStore.tablePrefix = QRTZ_*
quartz.jobStore.dataSource = celloScheduler
quartz.dataSource.celloScheduler.connectionString = Server=23.23.212.41;Database=ApplicationDB;Uid=sdev;Pwd=Password@123!;
quartz.dataSource.celloScheduler.provider = SqlServer-20
quartz.plugin.injectJobListner.type = CelloSaaS.TimeScheduler.QuartzSchedulerPlugin, CelloSaaS.TimeScheduler

4) Now you start the Quartz service using start.bat from the cellosaas_v4.3.2_developer_package\CelloSaaS Service Installers\Quartz folder.

1.5 *Sample Code*

Following sample code is used to trigger the particular job daily.

```
/// <summary>
 /// Sample code for Create job. Call this method only once in application.
 /// </summary>
 private void CreateJob()
 {
     string jobReference = "Sample Job";
     string emailJobName = "Quartz_Job_" + jobReference;

     SchedulerService schedulerService = new SchedulerService();

     Job reminderJob = new Job
     {
         Name = "Sample schedule Job",
         CreatedBy = UserIdentity.UserId,
         JobType = "TestApplication.SampleJob,TestApplication",
         Status = true
     };

     string jobId = schedulerService.AddJob(reminderJob);

     // create time schedule instance
     TimeSchedule timeSchedule = new TimeSchedule
     {
         JobId = reminderJob.JobId,
         JobReferenceName = jobReference,
         ScheduleId = "0E179534-E3A0-4C1D-9106-91F7937450C7",
         StartDateTime = DateTime.Today,
         EndDateTime = null,
         CreatedBy = UserIdentity.UserId,
         Status = true
     };

     Quartz.IJob iJobInstance = CreateJobInstance(reminderJob);

     Type jobInstanceType = iJobInstance.GetType();

     // Create Job Details
     JobDetail reminder = new JobDetail(
         reminderJob.Name,
        "SampleGroupName",
         jobInstanceType,
         false,
         true,
         true);

     Quartz.IJobListener listener = new CelloSaaS.TimeScheduler.QuartzJobListener();

     ISchedulerFactory schedulerFactory = new StdSchedulerFactory();
     IScheduler quartzScheduler = schedulerFactory.GetScheduler();

     quartzScheduler.AddJobListener(listener);

     reminder.AddJobListener(listener.Name);
```

```
            SimpleTrigger trigger = CreateSimpleTriggerForSampleJob(timeSchedule);

            quartzScheduler.ScheduleJob(reminder, trigger);

            ITimeSchedulerService timeSchedulerService =
ServiceLocator.Resolve<ITimeSchedulerService>();
            timeSchedulerService.CreateTimeSchedule(timeSchedule);
        }

        /// <summary>
        /// Craete Job instance
        /// </summary>
        /// <param name="job">Job Detail</param>
        /// <returns></returns>
        private Quartz.IJob CreateJobInstance(Job job)
        {
            //assembly name and type seperation
            if (!string.IsNullOrEmpty(job.JobType))
            {
                string[] assemblyAndType = job.JobType.Split(',');
                string typeName = assemblyAndType[0];
                string assemblyName = assemblyAndType[1];

                object jobObject = Activator.CreateInstance(assemblyName, typeName).Unwrap();

                Quartz.IJob iJobInstance = (Quartz.IJob)jobObject;

                return iJobInstance;
            }
            return null;
        }

        /// <summary>
        /// Create simple trigger for sample job.
        /// </summary>
        /// <param name="timeSchedule"></param>
        /// <returns></returns>
        private SimpleTrigger CreateSimpleTriggerForSampleJob(TimeSchedule timeSchedule)
        {
            string simpleTriggerName = "Quartz_Trigger" + timeSchedule.JobReferenceName;

            SimpleTrigger trigger = new SimpleTrigger(
                simpleTriggerName,
                "SampleGroupName", // Group Name
                timeSchedule.StartDateTime,
                timeSchedule.EndDateTime,
                SimpleTrigger.RepeatIndefinitely, // Run infinitely
                TimeSpan.FromDays(1) // Every day
                );

            return trigger;
        }
```

2 Contact Information

Any problem using this guide (or) using Cello Framework. Please feel free to contact us, we will be happy to assist you in getting started with Cello.

Email: support@techcello.com

Phone: +1(609)503-7163

Skype: techcello