Quartz Scheduler Module

Overview

There are many activities within a healthcare information system which benefit from automation. For example: daily clinic reports might need to be generated the morning before a clinic opening, decision support rules might need to be run over a patient group once every week, or healthcare data coming in from outside sources might need to be analyzed every 30 mins. To give end implementations the flexibility to define these sorts of automated conveniences, the OpenMRS community has added scheduler functionality to the core software application. The OpenMRS scheduler (based on the JDK Timer) currently supports executing tasks on regular basis (i.e. a task will execute every X seconds, minutes, hours, days). However, there's no way to schedule a task to recur at a particular time every week. The data model supports this type of feature (start time and repeat interval), but we need to add some logic to the Scheduler to handle triggering these tasks to occur at exact times.

We want to support more complex scheduling (i.e. cron-like) in the very near future, where we can specify rules that will allow Tasks to run more/more frequently based on the day/time. For instance, an implementation might want to schedule a Task to run once every day at midnight, but also at 6pm on Saturday and Sunday. This type of scheduling would be difficult to support using the JDK Timer, so we would like to add an extension point to the scheduler to allow a developer (probably myself) to create a module based on the Quartz Scheduler. This will allow us to handle more advanced scheduling requirements.

Requirements

High-Level Specification

These are some of the high level specifications and user stories that could be included in this project. During the design phase, we will add even features and requirements. After creating the full backlog of features, the team will meet with users/stakeholders to prioritize all features in order to figure out which ones are going to be implemented in a first release.

- Discuss synchronization/threading issues that we've had to deal with in the current scheduler implementation.
- Decide whether to build as a module to extend current functionality or as a core component to replace the current functionality.
- Decide whether to include quartz tables in the openmrs schema or within a separate schema.
- Implement a Quartz version of the scheduler to extend (possibly replace) the current JDK Timer implementation.
- Implement advanced task scheduling (with Quartz cron triggers).
- Implement simple task scheduling (with Quartz simple triggers).
- Migrate existing scheduled tasks to use new simple task scheduling (if time permits).
- Allow users to create, retrieve, update, and delete tasks.
- Allow users to schedule tasks to be run on an interval (i.e. every 30 seconds) or at specific moments in time (Tue, March 24 at 10:42pm).
- Allow users to start and stop tasks.
- Allow users to schedule emails to be sent at a specific time.
- Create unit tests for each expected behavior of every method within the scheduler service and its domain objects.
- Redesign the Manage Scheduler user interface (if time permits).

User Stories

1. Must Have

- Allow user to create a new task.
- Allow user to view the details of a particular schedule.
- Allow user to delete a particular schedule.
- Allow user to start a particular schedule.
- Allow user to stop a particular schedule from running.
- Allow user to execute scheduling combinations.
  - Allow user to schedule a task to run only once on 6/23/2009 11:52PM (Tested)
  - Allow user to schedule a task which would run every 5 seconds on 6/23/2009 11:52PM infinitely (Tested)
  - Allow user to schedule a task which would run every 5 seconds on 6/23/2009 11:52PM for only 10 times. (Tested)
  - Allow user to schedule a task which would run every 5 seconds on 6/23/2009 11:52PM until 6/24/2009 12:45AM. (Tested)
  - Allow user to schedule a task which would run every 7 Minutes on 6/24/2009 11:53PM for only 9 times. (Tested)
  - Allow user to schedule a task which would run every 3 Minutes on 6/24/2009 11:53PM until 6/24/2009 12:46AM. (Tested)
  - Allow user to schedule a task which would run every 1 Hours on 6/24/2009 11:53PM infinitely (Tested)
  - Allow user to schedule a task which would run every 1 Hours on 6/23/2009 11:51PM for only 9 times. (Tested)
  - Allow user to schedule a task which would run every 1 Hours on 6/24/2009 11:53PM until 6/25/2009 12:54AM. (Tested)
  - Allow user to schedule a task which would run every 2 Days on 6/24/2009 11:53PM infinitely (Tested)
  - Allow user to schedule a task which would run every 2 Days on 6/24/2009 11:51PM for only 9 times. (Tested)
  - Allow user to schedule a task which would run every 1 Days on 6/24/2009 11:53PM until 6/27/2009 12:54AM. (Tested)
  - Allow user to schedule a task which would run Every Weekday Infinitely (Tested)
  - Allow user to schedule a task which would run Every Weekday But end on 6/27/2009 12:54AM. (Tested)
  - Allow user to schedule a task which would execute on a selected day of the week or several days of a week and would run for infinite number of times. (Tested)
  - Allow user to schedule a task which would execute on a selected day of the week or several days of a week and would run until a given end date. (Tested)
  - Allow user to schedule a task which would execute once every 2 months and repeat only 2 times. (Tested)
  - Allow user to schedule a task which would execute once every 2 months for infinite number of times. (Tested)
  - Allow user to schedule a task which would execute once every 2 months and it to end on a given end date/time. (Tested)
• Allow user to schedule a task which would run on the first Sunday of every month infinitely. (Tested)
• Allow user to schedule a task which would run on the last Monday of every month infinitely. (Tested)
• Allow user to schedule a task which would run on the second Wednesday of every month and end on a given date and time. (Tested)
• Allow user to schedule a task which would run on the third Tuesday of every month and end on a given date and time. (Tested)
• Allow user to schedule a task which would run on 21st of January every year infinitely. (Tested)
• Allow user to schedule a task which would run on 21st of January every year and end on a given date/time. (Tested)
• Allow user to schedule a task which would run on the first Sunday of every March infinitely. (Tested)
• Allow user to schedule a task which would run on the last Monday of every September infinitely. (Tested)
• Allow user to schedule a task which would run on the second Wednesday of every June and end on a given date and time. (Tested)
• Allow user to schedule a task which would run on the third Tuesday of every December and end on a given date and time. (Tested)
• Allow user to schedule a task to run once every 3 minutes (or every 30 seconds or every 10 hours or every 1 days).
• Allow user to schedule a task to run every 2 weeks on Mondays.
• Allow user to schedule a task to run on the 1st day of every month.
• Allow user to schedule a task to run on the 2nd Monday of the month.
• Allow user to schedule a task to run once every day at 6pm.
• Allow user to run a single task (class) on several different schedules.
• Allow user to execute a particular task immediately.
• Allow user to view the status of a particular schedule (Started, Stopped, Finished, Failed).
• Allow user to view the next occurrence time of which the scheduler will start executing.
• Allow user to view the date of which a particular schedule is being stopped running.
• Allow user to view the status of a particular schedule.
• Allow user to view the description of the status of a particular schedule.
• Allow user to set the priority of the schedule.
• Allow user to set the recurrence option for the schedule (Run Once, Seconds, Minutes, Hours, Daily, Weekly, Monthly, Yearly, Cron). 
• Allow user to restart the scheduler.
• Allow user to Pause all schedules in the scheduler.
• Allow user to resume all paused schedules.
• Allow user to view a particular tasks, number of remaining executions.
• Allow user to view a particular schedules list of execution dates

2. Should Have

• Allow user to search for a particular schedule by its description
• Allow user to sort the schedule manager by its columns.

3. Could Have

• Allow user to identify schedules by its priority with its unique row banding color. Allow user to click on the class name and view a full description of it.
• Allow user to view number of jobs in the Scheduler.
• Allow user to view number of jobs by status (started, finished, failed, stopped).

4. Won't Have

• Allow user to skip the next occurrence of a task from being executed.
• Allow user to view the reason for a task failure.

5. Unprioritized

• Allow user to view log file from the scheduler manager
• Allow user to view scheduler specific log messages from the Manage Scheduler page
• Allow user to view task specific log messages from the
• Allow user to subscribe and receive email notifications on task lifecycle events (start, each execution, end, error)
• Allow user to specify "properties" that the task can use during it's execution
• Allow user to select a Task subclass from a drop down (should include all Task subclasses from core and modules)
• Allow user to edit task definitions (which task to run).
• Allow user to update the schedule for an existing Task.
• Allow user to duplicate an existing task definition (task + schedule)
• Allow user to choose class from a drop down list.

Allow user to display the schedule list according to a time period of his choice (Day, Week, Month, Year). *

• Allow user to receive an email alert before a particular schedule starts for the first time.
• Allow user to receive emails when a particular schedule failed executing.
• Allow user to view the number of times a particular schedule should execute.
• Allow user to view the number of times which a particular schedule has executed.
• Allow user to view the scheduler Summary

Project Plan

Iteration 1 (Sunday, May 31, 2009)

• Created new basic module for quartz scheduler
• Committed quartzscheduler module to the svn repository
• Made module deployable in OpenMRS 1.5
• Integrated code and configuration that has already been written into the new module
• Allow user to schedule a simple hello world task to write a message to the log on some interval
Iteration 2 (Sunday June 7, 2009)
- Got to know more on how quartz work and its post-schedule (what options does it have after scheduling a job) features.
- Allow users to enter the task name of the class.
- Allow users to enter the description of the schedule.
- Allow users to set the priority of the schedule.
- Allow users to set the recurrence option for the schedule (a) Run Once (b) Seconds (c) Minutes (d) Hours (e) Daily (f) Weekly (g) Monthly (h) Yearly (i) Cron

Iteration 3 (Sunday June 14, 2009)
- Allow users to view class name of the class being scheduled.
- Allow users to click on the class name and view a full description of it.
- Allow users to view the owner of a particular schedule.
- Allow users to view the date of which a particular schedule is created.
- Allow users to view the date of which a particular schedule is be stopped running.
- Allow users to view the status of a particular schedule.
- Allow users to view the priority of a particular schedule.
- Allow users to view the status of a particular schedule.
  - Started
  - Stopped
  - Finished
  - Failed
- Allow users to view the next occurrence time of which the scheduler will start executing.
- User stories (view Task page)
  - Allow user to view the status of a particular schedule (Started, Stopped, Finished, Failed).
  - Allow user to view the next occurrence time of which the scheduler will start executing.
  - Allow user to view the date of which a particular schedule is being stopped running.
  - Allow user to view the status of a particular schedule.
  - Allow user to view the description of the status of a particular schedule.
- Added more strict validations to the scheduler Form page

Iteration 4 (due Sunday June 21, 2009)
completed implementation of the following pages:
- Scheduler Manager Page
- View Task Page
- Create Task Page
- QA test on the Create Task page (Bugs Found):
  - RunOnce
  - Seconds
    - when selected the repeat option it always executes one more time than what i put.
  - Minutes
    - Doesn't execute on the given end time, the schedule finished before it.
    - when selected the repeat option it always executes one more time than what i put.
  - Hours
    - when selected the repeat option it always executes one more time than what i put.
- Allow users to execute a particular task immediately.
- Allow users to skip the next occurrence of a task from being executed.
- Allow users to delete a particular schedule.
- Finalize feature list of the 3 pages with Justin.
- Finish the View Task page completely.
- Try to integrate openMRS Jobs, replacing the hello world dummy job.
- Deploy quartz scheduler module to labs

Iteration 5 (due Sunday June 28, 2009)
- Scheduler Types QA

Iteration 6 (due Sunday July 5, 2009)
- Fixed scheduling bugs in the TabsForm page.
- Allow user to restart the scheduler.
- Allow user to Pause all schedules in the scheduler.
- Allow user to resume all paused schedules.
- Allow user to view a particular tasks, number of remaining executions.
- Allow user to click on the class name and view a full description of it (scheduler manager page).
- Allow user to view a particular schedules list of execution dates
- Allow user to view number of jobs in the Scheduler.
- Allow user to view number of jobs by status (started, finished, failed, stopped).

Iteration 7 (due Sunday July 12, 2009)
- Followed up with some R&D on how to edit quartz JobDataMap on the fly.
- Fixed remaining bugs.

Iteration 8 (due Sunday July 19, 2009)
- Changed DumbJob to TaskWrapper
- view_tasks.jsp renamed to taskList.jsp
- tabForms.jsp renamed to taskForm.jsp
• Added the drop down list. other changes will follow after a UAT
• Schedule List on the View Task page should be "Next 10 execution times"
• Added Light Green and red. Green shows started tasks. Red shows paused/stopped tasks
• The "Run Now" and "Delete" buttons should be on opposite sides. In addition, the "delete" button should be smaller (the "trash" icon) or should be a checkbox, with a Delete Selected button under the table (or in the table footer).
• Added server time to the create task page.
• Removed priority from the Create Task and leave that as a detail for the Edit Task page.
• The Scheduler Manage page should also show the current time so users can compare the current time against all of the tasks that are in the system.
• Added the scheduler logger page with tailing functionality.
• Added the base to send notification mail
• Added the edit task functionality.

Iteration 9 (due Sunday July 26, 2009)
• Added Notification sending features
• Added properties functionality
• minor UI changes

Iteration 10 (due Sunday Aug 2, 2009)
• added the notification Title after the UI change
• UI changes (new functionality should stop at this stage)

Iteration 11 (due Sunday Aug 9, 2009)
• UI
  • Renamed Scheduler Manager to Task Manager
  • Renamed Scheduler Summary to Scheduler Manager
  • Renamed Scheduler Managerh4. =
    • Added some spacing between the scheduler Manager buttons
    • Added tooltips to Restart/Pause/Resume buttons saying what it'll do.
  • Scheduler Log h4. =
    • Changed "Tail Every" to be "Refresh every" with option (Never, 10 seconds, 30 seconds, 60 seconds)
    • Changed "Get Log Now!" to "Refresh Now"
    • added Divs around the logging contents so the user sees the whole page and only the logging content is scrollable.

**h4. Task Form **h2.
• Made the dialog box a little wider and taller. Maybe 750 x 550.
  • For the "Often" radio button, add "Every" above the options (like this) Every ( ) ___ seconds ( ) ______ minutes ( ) ______ hours ( ) ___ days ( ) weekday (Monday, Tuesday, Wednesday, Thursday, Friday)
  • Changed the "Start Date" / "Run Time" to be a single line (like "End at") Start at 01 00 00 AM archive:08-04-2009cal
  • Changed "Repeat infinite times" to "Repeat forever"

**Functionality**
• Added the Edit QuartzTask method
• mail fix (receive only once)
• Surrounded the pause/start scheduler with a box
• Number of lines displayed message in the scheduler log
• Fixed the Email URL

Iteration 12 (due Sunday Aug 16, 2009)
• Tooltips made more consistent.
• resuming a task would work only if the scheduler is in running state
• Added sorting by title to the scheduler manager datagrid.
• close button was made consistent in the create task page.
• made the module user guide.
• blog and project page update.

Mockups
Application

Student Checklist

- The candidate will complete the following tasks (at a minimum):
  - write a proposal that includes a realistic project plan.
  - complete the OpenMRS Application Template.
  - submit their application for review to the Google Summer of Code site.
- The candidate possesses:
  - some/all of the skills required for this project (Java, Hibernate, Spring, Quartz).
  - a good understanding of the project – ask good questions, write a proposal that speaks to me.
  - an understanding of the existing system and its limitations.
  - the skills to write a project plan that is realistic and well planned.
  - the ability to learn quickly.
  - enthusiasm for the project.
  - a willingness to learn.
  - a willingness to contribute before/after Google Summer of Code.
  - a willingness to maintain the project (source code, documentation) after the Google Summer of Code.
  - a willingness to communicate with their mentor and other OpenMRS developers.
• ... a willingness to communicate with users / implementers (the broader OpenMRS community).
• ... a good sense of humor.
• The candidate likes:
  • ... to take long walks on the beach.
  • ... to talk ... or not talk ... for hours.
  • ... snow peas.

FAQs

If you are a student with a question, please add your question to the list below and email Justin Miranda. I will respond as soon as I can.

Question 0: How are you doing?

Very well. Thank you for asking.

Question 1: I was going through your blog post for the quartz project and had some issues for the following about the following user stories ("Allow a user to schedule emails to be sent at a specific time" and "Allow a user to schedule reports to be generated and sent to stakeholders at a specific time"). Don't you think this is a bit out of scope for the scheduler? since Generating reports and sending emails different tasks, should the scheduler "do" those things rather than "executing" them? I noted that there were dummy tasks inserted in the demo app to send emails and generate reports. is that the solution you expect from the quartz scheduler (to run predefined tasks)?

Actually, these are very well within the scope of the project. We want to implement a scheduling system that can actually be used for use cases that we need. These two stories provide very basic use cases that we need to support. With that said, you won't need to write the functionality to generate a report or send an email. That's already done for you. You just need to integrate the scheduler with the components that provide those services (i.e. ReportService and MessageService).

Question 2: Is there a definite place where you store the Task classes? Or are you planning to do so in the future?

The class that implements Task is being stored in the "task_definition" table as a string that represents the class. The actual Class will be in a module or core and is available through the classpath.

Question 3 Is the project vacant?

Yes. You can submit applications/proposals until Friday, April 3, 2009.

Question 4 Can you review my proposal?

I will try my best to review and give feedback for students who request it. However, it's been really busy this week, so please use the checklist on our application template as well as the checklist above for suggestions on what I'd personally like to see from an application/proposal.

Question 5: I'm interested in your project, how should I proceed?

Ask questions about the project and then submit an application on the Google Summer of Code website.

Question 6: As I understood you have added an extension point named Quartz to the Scheduler of OpenMRS. So where to add the Service to get the Quartz functions? Is it within the Quartz package or outside. (Of course the implementation of Quartz will be in the Quartz package).

This is one of the design questions we need to answer at the beginning of this project: whether to replace or extend the current implementation of the scheduler. The student will lead the conversation with developers and stakeholders in order to come to an understanding of the preferred approach. Another factor in this decision will be whether or not the existing tasks can be migrated to the new scheduler framework before the end of the project. If not, then we might be better off extending the current scheduler implementation.

Question 7: Is it really needed to add the Simple Trigger as well? The functionality of the Simple Trigger is very much similar to the Scheduler already implemented in OpenMRS (Therefore in case we need a Simple schedule we can use the existing OpenMRS service). Depends on whether we're planning to replace the existing functionality or extend it. My bias would be to replace the current system, but we're going to have to discuss that during the design phase.

Question 8: To create a Cron Trigger we need to use the Cron Expressions. So if a developer using the services of QuartzSchedulerService should input the relevant Cron Expression. Is this appropriate?

Most likely. The API will need to be extended to support Cron expression. We'll discuss API changes once the project begins. From a UI perspective, we can create a basic UI that allows a user to enter a cron expression (as a first pass). However, we should also plan to build a more user-friendly user interface. The first pass UI could then be hidden from the user behind an "Advanced..." link (we'll keep this UI around for administrators that like the cron).
Question 9: What is the Data Store that should be used? In order to maintain the state we have to use a JDBC Database. In that case there are about 12 tables that need to be created within OpenMRS Database. How to handle this? Is it ok if this much number of tables to be added into the OpenMRS Database for scheduling? This is another design question that the student will need to answer. We'll either add it to the OpenMRS schema OR create a new quartz specific schema. In the former case, we'll probably prepend the tables with a prefix that sets it apart from the core tables.

Resources

Project Documents

- Project Discussion (has not been created yet)
- Release Backlog (has not been created yet)
- Iteration Backlog (has not been created yet)

OpenMRS Documents

- Scheduler API
- Quartz Scheduler Developer’s Guide
- Quartz Scheduler User Guide
- Technical Discussion about existing Scheduler

Blogs

- Justin Miranda
- Student Blog

External

- For more information on Quartz, please see the Quartz Project home page.
- more resources to follow

<body></html>