Service Monitor Application

service-monitor is an java web based Service Monitoring application designed to monitor the status of multiple services

This application has the following features:

- A service is defined as a host/port combination. To check if a service is up, the monitor will establish a TCP connection to the host on the specified port.
- If a connection is established, the service is up. If the connection is refused, the service is not up.
- The monitor will allow callers to register interest in a service, and a polling frequency. The callers will be notified when the service goes up and down.
- The monitor detects multiple callers registering interest in the same service, and will not poll any service more frequently than once a second.
- The monitor allows callers to register a planned service outage. The caller will specify a start and end time for which no notifications for that service will be delivered.
- The monitor will allow callers to define a grace time that applies to all services being monitored. If a service is not responding, the monitor will wait for the grace time to expire before notifying any clients. If the service goes back on line during this grace time, no notification will be sent.

Tools and Technologies

Service Monitor Application uses the following tools and technologies:

Technology	Version
JDK	1.8
Spring Boot	2.1.3.RELEASE
H2 Database	1.4.197
Log4j	2.11.2
Junit	4.12
Quartz Scheduler	2.3.1
JQuery	1.12.1
Bootstrap	3.3.4
Maven	4.0.0
Git	2.20.1
Github	N/A
Eclipse	Oxygen

Deliverables

The deliverables for the application can be found at the below location [SYSTEM PATH TO PROJECT FOLDER]\service-monitor\deliverables

Deliverable	Summary

Class Diagram.PNG	Class diagram of the application
Project Structure.PNG	Diagram depicting the project structure
Code Coverage Report.PNG	Contains the latest junit code coverage report of the project
Service Monitor Application Screen Shots.docx	Contains the application screenshots
Technical Design Document.pdf	Technical Design Document for the application
Requirement Specification.pdf	Requirement Specification for “Service Monitor Application”
service-monitor-javadoc.jar	Javadoc for the project
service-monitor-0.0.1-SNAPSHOT.jar	Executable “Service Monitor Application”

Assumptions

- The Service Monitor will be designed as a web application
- The data in In-Memory DB is sufficient and can be promoted to a persistent storage in future.
- Importance is given to the Java code and not look and feel of the application so using simple Jquery and Bootstap for the application due to time constraints of creating a seperate Angular project
- We are free to use scheduling tools like Quartz and Spring Scheduler and any other open sources tools for the project

Setup

- Ensure Jdk 1.8 or above is installed on your system. If not, you can download the latest version from the below link.
  Please follow the instructions in the link for the setup
  JDK Installation
- unzip the application(service-monitor.zip) to the folder of your choice.
- After you unzip the application, inside the [PATH TO APPLICATION]/service-monitor/deliverables folder you will find application executable with the name “service-monitor-0.0.1-SNAPSHOT.jar”
- Open a terminal and navigate to the “[PATH TO APPLICATION]\ActivityScheduler-master\deliverables” folder

```
$ cd [PATH TO APPLICATION FOLDER]\deliverables
```

- run the following command to execute the jar in the folder

```
$ java -jar service-monitor-0.0.1-SNAPSHOT.jar
```

- Open browser of your choice and nagivate to the url http://localhost:8080/
- Alternatively, You can just right click on the “service-monitor-0.0.1-SNAPSHOT.jar” , click on ‘Open With…’ and select ‘Java™ Platform SE binary’, Wait for sometime and then navigate to the url http://localhost:8080

Usage

Refer to : Service Monitor Application Screen Shots.docx for the details.

Please note: To schedule , the application currently uses CRON experssion which can be generated using below URL

CRON EXPRESSION GENERATOR

Technical Design

Project Structure

Refer to : Project Structure.PNG

Package Structure

Package	Summary
com.globalrelay.servicemonitor	contains application entrypoint classes
com.globalrelay.servicemonitor.configuration	contains application configuration classes
com.globalrelay.servicemonitor.constant	contains constants
com.globalrelay.servicemonitor.controller	contains the classes for rest endpoint implementations
com.globalrelay.servicemonitor.domain	contains all the domain classes
com.globalrelay.servicemonitor.exception	contains custom application exceptions
com.globalrelay.servicemonitor.facade	intefaces for facade

	classes
com.globalrelay.servicemonitor.facade.impl	implementation of facade classes
com.globalrelay.servicemonitor.job	intefaces for job classes
com.globalrelay.servicemonitor.job.impl	implementation of job classes
com.globalrelay.servicemonitor.repository	contains all the spring JPA repository classes
com.globalrelay.servicemonitor.service	intefaces for service classes
com.globalrelay.servicemonitor.service.impl	implementation for service classes
com.globalrelay.servicemonitor.strategy	intefaces for strategy classes
com.globalrelay.servicemonitor.strategy.impl	implementation for strategy classes
src/main/resources/static	contains the index.html file which is the applications UI
src/main/resources/static/css	contains application css files
src/main/resources/static/js	contains application script files

Domains

Class	Summary

AbstractDomainObject.java	An Abstract class for Auditing purpose. The fields created_at and updated_at will be autopopulated and increated by Spring JPA framework during create and update operations.
ServiceMonitorResponse.java	A domain class for ServiceMonitor Application responses. This class will be used to send both the success and failure messages.
ServiceMonitorStatus.java	A domain class to store the current status and execution details of a service monitoring job.
ServiceMonitorTask.java	A domain class to store the details of a task to be executed by the Scheduled job.

Class Design

Refer to : Class Diagram.PNG

Class	**Summary**
ServiceMonitorApplication.java	Spring boot application entrypoint class
ServiceMonitorController.java	REST Endpoints controller for Service Monitoring Application
EmailFacadeImpl.java	An implementation of EmailFacade interface that contains necessary logic to send an email to desired user with necessary subject line and text.
QuartzSchedulerFacadeImpl.java	An implementation of the interface ScheduleWriterFacade

	that contains necessary logic to create, start, stop and retrieve Jobs using an scheduler. Each Job should be capable of running a task
RouterJob.java	A generic Job that would be triggered by the Quartz Scheduler which will route the exectution to a specific job class mentioned in the task.
ServiceMonitorJob.java	An implementation of the AbstractJob. This job will be executed by the scheduler and contains the logic to trigger monitoring a service status
ServiceMonitorTaskRepository.java	An JPA repository interface to perform CRUD operations on the TASK table.
SchedulerServiceImpl.java	An implementation of {@linkplain SchedulerService} interface. This class implements logic to create, schedule, start, stop and search for tasks that needs to be scheduled using a Scheduler
ServiceMonitorServiceImpl.java	An implementation of {@linkplain ServiceMonitorService} interface. This class implements logic to retrieve task based on taskId, validate and perform monitoring of the service defined in the task

ServiceMonitorStrategyImpl.java	An implementation of {@linkplain ServiceMonitorStrategy}. This implementation uses Java TCP Socket API to check if the given service is up and running

Javadoc

Refer to : service-monitor-javadoc.jar

You can also find the javadoc generated under /target folder once the run the maven build on the application

Code Coverage

Below is the latest code coverage report generated on 14th April 2019

Refer to : Code Coverage Report.PNG

Future Enhancements

- Move the data towards a persistent storage.
- Implement the feature “If the grace time is less than the polling frequency, the monitor should schedule extra checks of the service.”
- Improve code coverage for Job and Strategy packages.
- Build the UI as seperate project in Angular for Maintainability
- Containerize the service-monitor project.

- Design state machines to efficiently handle changes for ACTIVE and INACTVIE statuses.
- Remove the CRON expression textbox and provide UI for user to choose the scheduling logic

Contact Information

Author : Ravikiran Butti,

Email Id : (ravikiran763@gmail.com)