NOTIFICATION SENDING AS CROSS CUTTING CONCERN IN ENTERPRISE APPLICATIONS

B. A. I. Sampath[1], C.J. Basnayakege[2]

[1]*Department of Electrical and Computer Engineering, Open University of Sri Lanka*
[2]*Department of Electrical and Computer Engineering, Open University of Sri Lanka*

INTRODUCTION

Technology has made vast changes in society. People use very high tech devices in their day to day life. Personal computers, mobile phones are some of them. Most software were built to use as standalone applications in earlier days. Now the situation has changed and everything functions as services which can be shared among many people. Software development focuses on building applications which are more centralized and reusable. Design architecture of software projects have become module based and different types of design concerns are used to achieve it. Modules are standalone components which can be operated independently. This approach has made many reusable components which can be reused with any other development project.

Application developed for the business domain or any other domain has mainly focused on increasing the productivity of the operations. In the same way the software development company also followed different types of methodologies, technologies and standards to increase the productivity of the software development projects. There are some aspect in the application development that are considered as the cross cutting concern which means the software concern that is external and orthogonal to the problem that a software component is designed to address. Transaction handling, security, logging, error handling, synchronization, memory allocation are some cross cutting concern in software development. We have considered that notification sending process in the application as the cross cutting concern. It is a different aspect of programming which should not interfere with the business logic. The developer should not worry about the notification sending process in the business logic. This requirement is satisfied with this project.

Most applications have a huge user base and the application interacts with users in different ways. Most business domains have a primary requirement to send various types of notifications to their users based on their activities. Identifying the various types of notifications which are common to many applications and finding the medium by which it is sent to the user and finally combining all these into one software component is the main objective of this project. There are different types of medium available today to communicate with others. Email (Electronic-mail), SMS (Short Message Service), MMS (Multimedia Message Service) and Fax are some of them. Email and SMS are the most common medium which is integrated to many applications to send messages to the end users.

Promotions, payments, offers, reminders and activities are some of the notification types commonly sent in many applications. If any application needs to send a notification mentioned previously, then the application has to facilitate such media integrations. Integration of Email or SMS sending can be written as a separate package and reuse it in the other classes or write it in a class which a particular operation processing is some of the way to achieve it. This approach is the result of a more specific solution which can be used within

[1]*Correspondences should be addressed to B. A. I. Sampath, Department of Electrical and Computer Engineering, Open University of Sri Lanka*

that project only because application logics are written to satisfy the scope of the business domain. If the developer wants to use it in another separate software project there could be many issues and modification to that particular coding to adapt it. Developing a more generic software module and reusing it in any software development project saves time in the development.

METHODOLOGY

We were concerned with a set of attributes before starting this project and searched whether there were any alternatives which have already been developed to cater for all these attributes. Following are those attributes: support more than one notification medium, support concurrency, support scheduling, support bulk sending, support multimedia content (images and templates) configurable with any service providers (Email and SMS service providers), deliver under open source license.

We could find out some software components which support each of this attributes separately. But none of the software components support all of these attributes at once. For example there are a number of software components which support sending email integration into the application. But none of them support concurrency, scheduling, bulk sending itself. All these attributes depend on the way that the developer will do the coding. If there are any requirements to schedule Email sending, then the developer has to think of a way to design the correct approach, develop it and test it. It may take considerable man days even a team to work on that. We focus on addressing this in our project development. We reused some of the existing software components which supported above attributes rather than developing it from scratch.

Notification API is integrated with two types of service providers. One such provider is Simple Mail Transfer Protocol (SMTP) server and the other one is SMS gateway. The developer can add configuration details of SMTP server in property file according to the client requirements. Kannel has a rich set of features which are very useful in the development process of sending SMS. The main advantage of it is that the developer can set the Kannel configuration to any mobile service providers. Currently Notification API supports one SMSC.

Notification jobs are stored in H2 memory database. Quartz scheduler runs in the same time interval and checks whether that new notification has arrived to the database. If the notification already came, then the scheduler will push the notification to the queue. Then the separate thread will handle the sending notification. This underlying mechanism was implemented with the design pattern called work stealing queue. This is the solution of IBM for the multi-threading application to handle the load of tasks and keep the application performance status higher. The notification API supports to send the Velocity email template message with embedded inline images.

Notification API is distributed as a Java Archive (jar) file. It is supported for Java based project development. Figure 1 depicts the component organization of the Notification API's notification engine. It communicated with some third party API in the run time.

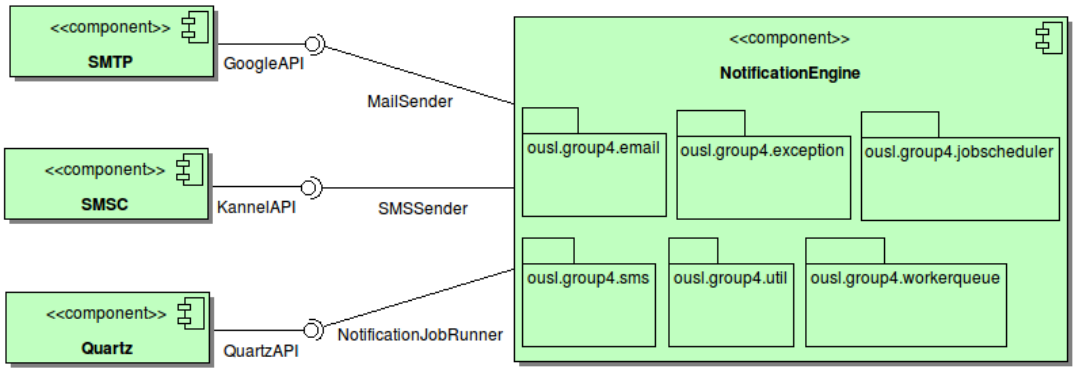

Figure 1 Component Diagram of Notification API

RESULTS AND DISCUSSION

Notification API consists of couple of methods where the developer wants to call in the coding. Rest of the operation is handled by the Notification API and it ensures that notification is sent to the recipient. There are two classes inside the library which are accessible to the outside developer after importing it to the project or after adding it as maven dependency. Notification API has two service classes (MailSender and SMSSender) which wrap all the internal services. Figure 2 and 3 depicts two methods which are more useful in notification sending in the application development.

```
/**
 * @param mailMap
 *            Must contains the below key names with relevant values
 *            <p>
 *            sender : the sender email address. cannot be null<br/>
 *            recipients : the receiver(s) email addresses and send type (String[][] array
 *            object). first array element: email address. second array element: pass one of
 *            constant in MailSendType. cannot be null. refer: MailSendType<br/>
 *            subject : the subject of email. cannot be null<br/>
 *            mailBody : the body of email. cannot be null<br/>
 *            attachments : Attachment(s) if necessary (Optional). must contain absolute path of
 *            file(s) (String[] array object).<br/>
 *            inlineImages : Inline image(s) if necessary (Optional). must contain absolute path
 *            of image(s) and content id (String[][] array object).
 *            </p>
 * @see ousl.group4.email.model.MailKeyBox
 * @throws ousl.group4.exception.NotificationAPIException
 */
void send(Map<String, Object> mailMap) throws NotificationAPIException;
```

Figure 2 Send method of MailSender.java

```
/**
 * @param smsMap Must contains the below key names with relevant values
 *            <p>
 *            sender : the sender mobile. cannot be null<br/>
 *            recipients : the receiver(s) mobile number (String[] array object). cannot be null.<br/>
 *            smsBody : the body of sms. cannot be null</p>
 * @see ousl.group4.sms.model.SmsKeyBox
 * @throws ousl.group4.exception.NotificationAPIException
 *
 */
void send(Map<String, Object> smsMap) throws NotificationAPIException;
```

Figure 3 Send method of SmsSender.java

Performance of the Notification API has increased by efficiently handling the concurrency. The main design concern here is to queue each and every notification sending inside the

library. Work stealing queue pattern is implemented in the Notification API and it ensures that a couple of threads are dedicated to execute notification sending job.

The following section specifies the nonfunctional requirements associated with the speed which the Notification API shall function.

Capacity concerning the minimum number of objects the Notification API can support. The Notification API shall support a minimum of 1 email to maximum of 500 emails at each method call. The Notification API shall support a minimum of 1 SMS to maximum of 100 SMS at each method call. The system shall support a minimum of 10,000 simultaneous interactions.

Latency concerning the maximum time that is permitted for the Notification API to execute specific tasks (i.e. send email or sms). This is dependent on the bandwidth of the network. Response time concerns the maximum time that is permitted for the Notification API to respond to requests: All system responses shall occur within 30 seconds. Throughput concerning how many executions of a given Notification API operation or use case path must the system is able execute in a unit of time: To Be Determined.

CONCLUSIONS/RECOMMENDATIONS

Notification API has been developed to customize with any changes and further any developer will be able to download the source and customize as it needs. It has already been implemented in two types of notifications medium which is email and SMS. Anyone is welcome to attach any enhancement with this API and used for their project developments.

NotificationJobRunner is responsible to notification scheduling and it is run as background process. It is highly recommended to define these classes as bean if the project uses spring framework. Figure 4 showed example bean configuration setting in the spring bean context file.

```
<!-- Notification API integration  -->
<bean name="notificationJobRunner" class="ousl.group4.jobscheduler.NotificationJobRunner"
     lazy-init="false" init-method="initSendNotifications"/>
<bean name="mailSender" class="ousl.group4.email.service.impl.MailSenderImpl"/>
<bean name="smsSender" class="ousl.group4.sms.service.impl.SmsSenderImpl"/>
```

Figure 4 Define Notification API service classes as spring beans

REFERENCES

Andreas Fink, Bruno Rodrigues, Stipe Tolj, Aarno Syvänen, Alexander Malysh, Lars Wirzenius, and Kalle Marjola, 2009. Kannel 1.4.3 User's Guide - Open Source WAP and SMS gateway. [E-book] Available at: www.kannel.org/userguide.shtml

Java.net - The Source for Java Technology Collaboration, 2011. A Method for Reducing Contention and Overhead in Worker Queues for Multithreaded Java Applications [online] Available at: http://today.java.net/article/2011/06/14/method-reducing-contention-and-overhead-worker-queues-multithreaded-java-applications

Quartz Scheduler - Enterprise Job Scheduler, 2011. Quartz Enterprise Job Scheduler Cookbook [online] Available at: http://quartz-scheduler.org/documentation/quartz-2.1.x/cookbook/

Developers.com, 2011. Java 5's BlockingQueue [online] Available at: http://www.developer.com/java/ent/article.php/3645111/Java-5s-BlockingQueue.htm

H2 database, 2012. Tutorial [online] Available at: http://www.h2database.com/html/tutorial.html

ACKNOWLEDGMENTS

I would like to give my gratitude to Dr. K. A. C. Udayakumar, Dr. L. S. K. Udugama and Dr. (Mrs.) K. G. H. U. W. Rathnayake who are major characters of the Bachelor of Software engineering program. Initial project proposal was accepted by them and their advice become more helpful in all the milestones we have passed successfully.

Next I would like to give my gratitude to group members H.M.P.P. Senevirathna, R.M.E.A. Rathnayaka, G.N.I. Garusinghe, W.A.D.C. Dinesh who gave me great support to successfully complete the project.