News System

# Overview

This code contains 4 project. I build it using Intelij. And was using intelij to run it also. I did not use maven to run the system. If you want to use maven you need to update the project to support framework module, by adding a module to the POM.

# Use

First run the news system, and you will see a welcome message.

To start sending messages open NewsServerApplication.java file on Intelij and right click on the fun and chose run. This will send messages to the News system and will show you an async behavior on processing these messages. On the Intelij Console you will start seeing green notification messages like these for only user 4 .

----------------------------Notification---------------------------------!

User Id: 4

Data: A[

----------------------------End Of Notification---------------------------------!

These message are printed from the device that got notified about them.

## Login logout

To simulate the login and logout behavior, you need to run UserManagementApplication. The same was you did for NewsServer Application. This will simulate user behaviors for user 1 , 2, and 3. As soon as the system will start running you will start seeing notification messages on the Intelij Console screen like:

----------------------------Notification---------------------------------!

User Id: 4

Data: A[

----------------------------End Of Notification---------------------------------!

----------------------------Notification---------------------------------!

User Id: 1

Data: A[

----------------------------End Of Notification---------------------------------!

## Async News receiving

NewsServerApplication sends requests to the news system every 100ms. After running with 100ms stop the three system and change the 100ms in the NewsServerApplication.java file under run method to 10ms then re-run the three system as you did before. You will notice the system is running and able to handle the requests. This is due to the @Transactional(isolation=Isolation.READ\_UNCOMMITTED) define in NewsProcessorTask file. If you removed @Transactional(isolation=Isolation.READ\_UNCOMMITTED) and re-run the system you will start seeing exceptions.

# Systems

## News

### Overview

This system process coming news from News Servers and send notification to user devices. It implements a TestDevice which prints green messages to Console just to simulate this behavior. The system also exposed a set of API so it can be used by News Servers and UserManagement applications. The system is very easy to expand by separating its services to deferent servers.

This system exposes the following APIS

/news

/Notifications

/Users

/Users/{id}.

### Code

The system has two main sub systems(RecievingService, and notification) that can be extracted to other servers in case of scalability need. Both of them provide rest interfaces to interact with them.

The RecievingService system receives news from the NewsServer, and send them to a bool threaded task which will process the request by creating needed data into the database and then send notification for any logged in users by using the notification system.

The Notification system will asynchronously send notification to the required device. It is currently implements only a testing notification device. After the device receives a message it notifies the News system that the message had been received so the News system will be update that news status. The system can be extended to support any needed device later on.

### Database data

To have a good idea on the news data I got the following from schema.sql

**INSERT** **INTO** Category(id,name)**VALUES**(1, 'Sports');

**INSERT** **INTO** Category(id,name)**VALUES**(2, 'News');

**INSERT** **INTO** Category(id,name)**VALUES**(3, 'General');

**INSERT** **INTO** UserStatus(id,name)**VALUES**(0, 'Not Active'); --user is not active

**INSERT** **INTO** UserStatus(id,name)**VALUES**(1, 'Active'); -- user is active but may be logged out

**INSERT** **INTO** UserStatus(id,name)**VALUES**(2, 'Loggedin'); --user is active and logged in

**INSERT** **INTO** Users(id,name,status)**VALUES**(1, 'User1', 1); --active user and has category 1 and 2

**INSERT** **INTO** Users(id,name,status)**VALUES**(2, 'User2', 1); --active user and has category 3 and 2

**INSERT** **INTO** Users(id,name,status)**VALUES**(3, 'User3', 1); --active user and has category 3

**INSERT** **INTO** Users(id,name,status)**VALUES**(4, 'User4', 2); --logedin user and has category 1, 2 and 3

**INSERT** **INTO** Users(id,name)**VALUES**(5, 'User5'); --inactive user and has category 1, 2 and 3.

**INSERT** **INTO** Users(id,name)**VALUES**(6, 'User6'); --inactive user and has category 1, 2 and 3.

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(1, 1);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(1, 2);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(2, 2);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(2, 3);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(3, 3);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(4, 1);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(4, 2);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(4, 3);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(5, 1);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(5, 2);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(5, 3);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(6, 1);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(6, 2);

**INSERT** **INTO** UserCategories(userId,categoryId)**VALUES**(6, 3);

**INSERT** **INTO** UserNewsDeliveryStatus(id,name)**VALUES**(1, 'created');

**INSERT** **INTO** UserNewsDeliveryStatus(id,name)**VALUES**(2, 'send to notify');

**INSERT** **INTO** UserNewsDeliveryStatus(id,name)**VALUES**(3, 'notifying');

**INSERT** **INTO** UserNewsDeliveryStatus(id,name)**VALUES**(4, 'delivered');

## News Server

### Overview

This system sends news continuously every 100ms to the News system. The system randomly chose news category and send it. The news data starts with aa, ab, ac, …, aaa,aab, aac and so on.

### Code

This system will create a news by randomly select a news category and automatically create a news message. The news message will grow with time, and it has data of asci characters starting from ‘a’ and ending with ‘Z’. The system will post the created news to News system rest method /news. After sending the news it sleeps for N ms then continue the process.

## User Management

### Overview

This system simulated use logging in and out to the system. The purpose is to allow you to see that the user will start receiving news as soon as the user log in.

### Code

This system defines three users with ids 1,2 and 3. The system continuously sends login request waits for 4 seconds and send a logout request, then do the same for send user then for the third user, and start again from the first user.

## Notification web page

There is page 1.html. but unfortunately I did not continue its implementation, but usually this is how it can be done.

# Systems to implement

## Background News

We need a background news service to send news that had not been send to user or devices because they were inactive, not logged in, or for some reason or another the system could not deliver the message, for example because of downtime maintenance.

## UI

We need to provide UI for user management, and user news, administration and others.