**Resource Scheduler**

**Bhupinder Singh**

Contents

1. Introduction
2. Implementation
3. Java Packages
4. Resource Scheduler Java Classes
5. Execution

**Note:**

*I wanted to implement this solution more loosely coupled using JMS queue but due to lack of time I just implemented in spring. This is initial code. It still needs some tuning before deployed to production. Using JMS based loosely coupled solution, it makes this solution more reliable and we can maintain transactions mile stones. If some exception happens in Target Consumer we can rollback message back to queue and replay again.*

1. **Introduction**

Resource scheduler poll for different messages and send those messages to target consumer.

It listens to an Active MQ JMS queue. When there is any message arrives in the queue it forward that message to target consumer after applying priority algorithm.

Target Consumer

Apply Message Priority Algorithm

Get Message

Resource Scheduler Components

* 1. Message Producer

Message producer will put messages into Active MQ JMS queue (e.g. Test3). There can be multiple producers for the same queue but there will be only one consumer for the queue. Message will a serialized java object (MessageInfo).

* 1. Message Consumer

Message consumer will listen to JMS queue (Test3). Whenever there is new message arrives in queue message consumer take that message and send auto confirmation to the JMS broker. JMS consumer sends that messages to other java components for further processing.

* 1. Message Processor

Message processor process the messages send by message consumer. It prioritises the message based on Priority Algorithm and sends those messages to Target Message Consumer by Message Gateway.

* 1. Message Gateway

Message gateway send message to target consumer. It creates no of consumer threads based upon available resources.

* 1. Target Consumer

It will consume all the messages send by Message Gateway. After consuming the message it will intimate Message Processor that a resource has been free. So, Message Processor send next message to Message Gateway.

1. Implementation:

Resource scheduler application is implemented using Spring Framework. Spring provides powerful framework for enterprise applications. It uses JMS module to publish and consumer messages.

Message termination and cancelation use case scenario also handled in this code.

1. Java Packages
2. **org.jpmorgan.resource.msg** : This package contains all the interfaces and message class for resource scheduler.
3. **org.jpmorgan.resource.msg.mngt**: This package contains all the business logic classes which implements core message package interfaces.
4. **org.jpmorgan.resource.msg.exception**: This package contains custom exceptions used in scheduler implementation classes
5. **org.jpmorgan.resource.target**: This package contains target message consumer.
6. **org.test.jpmorgan.ressource.msg**: This package contains all the unit test classes.
7. Resource Scheduler Classes

MessageInfo:

This class is basic payload between consumer and producer. It contains following fields

MessageId

GroupId

Status

Details

Gateway:

This interface defines method to interact with target consumer

Send(MessageInfo)

MessageProcessor:

This interface defines method to process message received from message consumer.

GatewayImpl :

This class implements Gateway interface and send message to target consumer.

Send(MessageInfo)

MessageConsumer:

This class consumer messages send by message producer. This class implements JMS message listener interface. OnMessage () method invokes each time messages arrive in the queue.

MessagePrioroty**:**

This class implements message priority algorithms. It returns list of shorted messages to Message Processor class.

MessageProcessorImpl

This class implements message processing logic . It implements MesagePrcessor and Runnable interfaces. It implements following list of methods

Run()

updateResource()

getMessageList()

addMessage()

startConsumer()

MessageUtil :

This is utility class which define shared properities values.

ResourceSchedulerApp:

This is the main class which starts the resource scheduler application.

1. **Execution**

Pre-requisite:

* Spring Tool Suite should be installed
* Active MQ should be installed and started
* A test Queue by name Test3 should be created.
* Import resource scheduler project in Spring Tool Suite
* Resource Scheduler can be started by right clicking ResourceSchedulerApp class and selecting **run as application** option.
* To start JMS producer, right click on com.test. JMSMessageSender class. It will produce 10 JMS message with different group id (1 to 5 randomly).