Working with Amazon Simple Queue Service using java

Posted by: [Niraj Singh](https://www.javacodegeeks.com/author/niraj-singh/" \o ") in [Enterprise Java](https://www.javacodegeeks.com/category/java/enterprise-java/) June 25th, 2013

Amazon Simple Queue Service or SQS is a highly scalable hosted messaging queue provided by Amazon Webservice stack. Amazon SQS can be used to completely decouple operations of different components within the system which otherwise exchange data to  perform independent tasks. Amazon SQS also helps us in saving the data which would be lost in case the application is down or if one of the component becomes unavailable.

Amazon SQS Features (Copied directly from amazon website)

1. **Redundant infrastructure**—Guarantees delivery of your messages at least once, highly concurrent access to messages, and high availability for sending and retrieving messages
2. **Multiple writers and readers**—Multiple parts of your system can send or receive messages at the same time. SQS locks the message during processing, keeping other parts of your system from processing the message simultaneously.
3. **Configurable settings per queue**—All of your queues don’t have to be exactly alike. For example, one queue can be optimized for messages that require a longer processing time than others.
4. **Variable message size**—Your messages can be up to 65536 bytes (64 KiB) in size. For even larger messages, you can store the contents of the message using the Amazon Simple Storage Service (Amazon S3) or Amazon SimpleDB and use Amazon SQS to hold a pointer to the Amazon S3 or Amazon SDB object. Alternately, you can split the larger message into smaller ones.
5. **Access control**—You can control who can send messages to a queue, and who can receive messages from a queue
6. **Delay Queues**—A delay queue is one which the user sets a default delay on a queue such that delivery of all messages enqueued will be postponed for that duration of time. You can set the delay value when you create a queue with CreateQueue, and you can update the value with SetQueueAttributes. If you update the value, the new value affects only messages enqueued after the update.

With above knowledge in place let us try to use SQS for creating a simple photo processing service.

Problem Defination for the tutorial

We will be creating a simple photo-processing application with following components.

1. **Photo Uploader serivce** – This is a webservice which allows users to upload a photo to the system. Once the photo is uploaded they are stored in a temporary storage. To keep it simple we will assume that user has already uploaded the photo and stored it in a predefined location.
2. **AWSSimpleQueueServiceUtil** – This is a utility class which wraps a Amazon SQS client and performs basic CRUD operations on the SQS queue.
3. **PhotoProcessingManager** – Manages the entire show. It will invoke AWSSimpleQueueServiceUtil  to send/receive messages to SQS and invoke PhotoProcessor to process the photo and finally delete the message from the queue.  Mostly we should intend this class to act as Listener to the SQS but for simplicity we will just be using a Poll mechanism to pull the messages from SQS.
4. **PhotoProcessor** – Gets a photo message from SQS through PhotoProcessingManager and generates a thumbnail.

Before beginning it would be great if you go through video in the following link: [http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/Welcome.html](http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/Welcome.html.)

Steps for getting started

1. Create a amazon account. You will need a credit card for that
2. Log on to your console [console.aws.amazon.com](https://www.javacodegeeks.com/2013/06/console.aws.amazon.com).
3. On console dashboard search for SQS and click on it. It will take you to your SQS home.
4. Create a new SQS queue and name it PhotoQueue. Leave the rest of the setting as default. We can also create and delete a SQS queue dynamically but in this tutorial I have a pre-created queue which I will be using in my code.
5. Now we have a queue and now we will create a simple java project in our favorite java editor and see how we can leverage this queue.
6. Once you are done you need to download your security credentials. For this go to “My Account”/”security credentials”. What we are after is access credentials. You will see that there are 3 types of access credentials one of them is “Access Keys”. We need this to access and work on PhotoQueue we just created. We will create a new set of access keys and store the access key and secret key is a safe location.
7. Now download the  sdk for java from here. <http://aws.amazon.com/sdkforjava>. In lib folder of sdk copy the aws-java-sdk-1.3.33.jar to your project classpath.

Maven users can add following dependency in their POM

|  |  |
| --- | --- |
| 1 | <dependency> |
| 2 | <groupId>com.amazonaws</groupId> |

|  |  |
| --- | --- |
| 3 | <artifactId>aws-java-sdk</artifactId> |
| 4 | <version>1.3.33</version> |

|  |  |
| --- | --- |
| 5 | </dependency> |

Create a file called “AwsCredentials.properties” store this in your project. This file will contain following properties

|  |  |
| --- | --- |
| 1 | accessKey = |
| 2 | secretKey = |

The values of these properties are the access key you generated in the step 6.

1. For photo processing I am using [**imgscalr**](http://www.thebuzzmedia.com/software/imgscalr-java-image-scaling-library/). It is a lightweight and awesome photo processing library in java for doing simple tasks like resize, rotate, crop etc. You can download the jar from <http://www.thebuzzmedia.com/software/imgscalr-java-image-scaling-library/#download>. Maven users can add the following to their dependency list.

|  |  |
| --- | --- |
| 1 | <dependency> |
| 2 | <groupId>org.imgscalr</groupId> |

|  |  |
| --- | --- |
| 3 | <artifactId>imgscalr-lib</artifactId> |
| 4 | <version>4.2</version> |

|  |  |
| --- | --- |
| 5 | <type>jar</type> |
| 6 | <scope>compile</scope> |

|  |  |
| --- | --- |
| 7 | </dependency> |

Now we are ready to rock and roll and get our hands dirty with some code.

AWSSimpleQueueServiceUtil.java

|  |  |
| --- | --- |
| 001 | package com.aranin.adconnect.util.aws; |
| 002 |  |

|  |  |
| --- | --- |
| 003 | import com.amazonaws.auth.BasicAWSCredentials; |
| 004 | import com.amazonaws.services.sqs.AmazonSQS; |

|  |  |
| --- | --- |
| 005 | import com.amazonaws.services.sqs.AmazonSQSClient; |
| 006 | import com.amazonaws.services.sqs.model.\*; |

|  |  |
| --- | --- |
| 007 |  |
| 008 | import java.io.FileInputStream; |

|  |  |
| --- | --- |
| 009 | import java.util.List; |
| 010 | import java.util.Properties; |

|  |  |
| --- | --- |
| 011 |  |
| 012 | /\*\* |

|  |  |
| --- | --- |
| 013 | \* Created by IntelliJ IDEA. |
| 014 | \* User: Niraj Singh |

|  |  |
| --- | --- |
| 015 | \* Date: 3/19/13 |
| 016 | \* Time: 10:44 AM |

|  |  |
| --- | --- |
| 017 | \* To change this template use File | Settings | File Templates. |
| 018 | \*/ |

|  |  |
| --- | --- |
| 019 | public class AWSSimpleQueueServiceUtil { |
| 020 | private BasicAWSCredentials credentials; |

|  |  |
| --- | --- |
| 021 | private AmazonSQS sqs; |
| 022 | private String simpleQueue = "PhotoQueue"; |

|  |  |
| --- | --- |
| 023 | private static volatile  AWSSimpleQueueServiceUtil awssqsUtil = newAWSSimpleQueueServiceUtil(); |
| 024 |  |

|  |  |  |
| --- | --- | --- |
| 025 | | /\*\* |
| 026 | \* instantiates a AmazonSQSClient <http://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/services/sqs/AmazonSQSClient.html> | | |

|  |  |
| --- | --- |
| 027 | \* Currently using  BasicAWSCredentials to pass on the credentials. |
| 028 | \* For SQS you need to set your regions endpoint for sqs. |

|  |  |
| --- | --- |
| 029 | \*/ |
| 030 | private   AWSSimpleQueueServiceUtil(){ |

|  |  |
| --- | --- |
| 031 | try{ |
| 032 | Properties properties = new Properties(); |

|  |  |  |  |
| --- | --- | --- | --- |
| 033 | properties.load(newFileInputStream("D:/samayik/adkonnection/src/main/resources/AwsCredentials.properties")); | | |
| 034 | | this.credentials = new   BasicAWSCredentials(properties.getProperty("accessKey"), |

|  |  |  |  |
| --- | --- | --- | --- |
| 035 | properties.getProperty("secretKey")); | | |
| 036 | | this.simpleQueue = "PhotoQueue"; |

|  |  |
| --- | --- |
| 037 |  |
| 038 | this.sqs = new AmazonSQSClient(this.credentials); |

|  |  |
| --- | --- |
| 039 | /\*\* |
| 040 | \* My queue is in singapore region which has following endpoint for sqs |

|  |  |
| --- | --- |
| 041 | \* [https://sqs.ap-southeast-1.amazonaws.com](https://sqs.ap-southeast-1.amazonaws.com/) |
| 042 | \* you can find your endpoints here |

|  |  |  |
| --- | --- | --- |
| 043 | \* <http://docs.aws.amazon.com/general/latest/gr/rande.html> | |
| 044 | \* |

|  |  |
| --- | --- |
| 045 | \* Overrides the default endpoint for this client ("sqs.us-east-1.amazonaws.com") |
| 046 | \*/ |

|  |  |
| --- | --- |
| 047 | this.sqs.setEndpoint("[https://sqs.ap-southeast-1.amazonaws.com](https://sqs.ap-southeast-1.amazonaws.com/)"); |
| 048 | /\*\* |

|  |  |
| --- | --- |
| 049 | You can use this in your web app where    AwsCredentials.properties is stored in web-inf/classes |
| 050 | \*/ |

|  |  |
| --- | --- |
| 051 | //AmazonSQS sqs = new AmazonSQSClient(new ClasspathPropertiesFileCredentialsProvider()); |
| 052 |  |

|  |  |
| --- | --- |
| 053 | }catch(Exception e){ |
| 054 | System.out.println("exception while creating awss3client : " + e); |

|  |  |
| --- | --- |
| 055 | } |
| 056 | } |

|  |  |
| --- | --- |
| 057 |  |
| 058 | public static AWSSimpleQueueServiceUtil getInstance(){ |

|  |  |
| --- | --- |
| 059 | return awssqsUtil; |
| 060 | } |

|  |  |
| --- | --- |
| 061 |  |
| 062 | public AmazonSQS getAWSSQSClient(){ |

|  |  |
| --- | --- |
| 063 | return awssqsUtil.sqs; |
| 064 | } |

|  |  |
| --- | --- |
| 065 |  |
| 066 | public String getQueueName(){ |

|  |  |
| --- | --- |
| 067 | return awssqsUtil.simpleQueue; |
| 068 | } |

|  |  |
| --- | --- |
| 069 |  |
| 070 | /\*\* |

|  |  |
| --- | --- |
| 071 | \* Creates a queue in your region and returns the url of the queue |
| 072 | \* @param queueName |

|  |  |
| --- | --- |
| 073 | \* @return |
| 074 | \*/ |

|  |  |
| --- | --- |
| 075 | public String createQueue(String queueName){ |
| 076 | CreateQueueRequest createQueueRequest = new CreateQueueRequest(queueName); |

|  |  |
| --- | --- |
| 077 | String queueUrl = this.sqs.createQueue(createQueueRequest).getQueueUrl(); |
| 078 | return queueUrl; |

|  |  |
| --- | --- |
| 079 | } |
| 080 |  |

|  |  |
| --- | --- |
| 081 | /\*\* |
| 082 | \* returns the queueurl for for sqs queue if you pass in a name |

|  |  |
| --- | --- |
| 083 | \* @param queueName |
| 084 | \* @return |

|  |  |
| --- | --- |
| 085 | \*/ |
| 086 | public String getQueueUrl(String queueName){ |

|  |  |
| --- | --- |
| 087 | GetQueueUrlRequest getQueueUrlRequest = new GetQueueUrlRequest(queueName); |
| 088 | return this.sqs.getQueueUrl(getQueueUrlRequest).getQueueUrl(); | |

|  |  |
| --- | --- |
| 089 | } |
| 090 |  |

|  |  |
| --- | --- |
| 091 | /\*\* |
| 092 | \* lists all your queue. |

|  |  |
| --- | --- |
| 093 | \* @return |
| 094 | \*/ |

|  |  |
| --- | --- |
| 095 | public ListQueuesResult listQueues(){ |
| 096 | return this.sqs.listQueues(); |

|  |  |
| --- | --- |
| 097 | } |
| 098 |  |

|  |  |
| --- | --- |
| 099 | /\*\* |
| 100 | \* send a single message to your sqs queue |

|  |  |
| --- | --- |
| 101 | \* @param queueUrl |
| 102 | \* @param message |

|  |  |
| --- | --- |
| 103 | \*/ |
| 104 | public void sendMessageToQueue(String queueUrl, String message){ |

|  |  |
| --- | --- |
| 105 | SendMessageResult messageResult =  this.sqs.sendMessage(newSendMessageRequest(queueUrl, message)); |
| 106 | System.out.println(messageResult.toString()); |

|  |  |
| --- | --- |
| 107 | } |
| 108 |  |

|  |  |
| --- | --- |
| 109 | /\*\* |
| 110 | \* gets messages from your queue |

|  |  |
| --- | --- |
| 111 | \* @param queueUrl |
| 112 | \* @return |

|  |  |
| --- | --- |
| 113 | \*/ |
| 114 | public List<Message> getMessagesFromQueue(String queueUrl){ |

|  |  |
| --- | --- |
| 115 | ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest(queueUrl); |
| 116 | List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages(); |

|  |  |
| --- | --- |
| 117 | return messages; |
| 118 | } |

|  |  |
| --- | --- |
| 119 |  |
| 120 | /\*\* |

|  |  |
| --- | --- |
| 121 | \* deletes a single message from your queue. |
| 122 | \* @param queueUrl |

|  |  |
| --- | --- |
| 123 | \* @param message |
| 124 | \*/ |

|  |  |
| --- | --- |
| 125 | public void deleteMessageFromQueue(String queueUrl, Message message){ |
| 126 | String messageRecieptHandle = message.getReceiptHandle(); |

|  |  |
| --- | --- |
| 127 | System.out.println("message deleted : " + message.getBody() + "." + message.getReceiptHandle()); |
| 128 | sqs.deleteMessage(new DeleteMessageRequest(queueUrl, messageRecieptHandle)); |

|  |  |
| --- | --- |
| 129 | } |
| 130 |  |

|  |  |
| --- | --- |
| 131 | public static void main(String[] args){ |
| 132 |  |

|  |  |
| --- | --- |
| 133 | } |
| 134 |  |

|  |  |
| --- | --- |
| 135 | } |

**PhotoProcessor.java**

|  |  |
| --- | --- |
| 01 | package com.aranin.adconnect.util.aws; |
| 02 |  |

|  |  |
| --- | --- |
| 03 | import org.imgscalr.Scalr; |
| 04 |  |

|  |  |
| --- | --- |
| 05 | import javax.imageio.ImageIO; |
| 06 | import java.awt.image.BufferedImage; |

|  |  |
| --- | --- |
| 07 | import java.io.File; |
| 08 |  |

|  |  |
| --- | --- |
| 09 | /\*\* |
| 10 | \* Created by IntelliJ IDEA. |

|  |  |
| --- | --- |
| 11 | \* User: Niraj Singh |
| 12 | \* Date: 3/19/13 |

|  |  |
| --- | --- |
| 13 | \* Time: 12:32 PM |
| 14 | \* To change this template use File | Settings | File Templates. |

|  |  |
| --- | --- |
| 15 | \*/ |
| 16 | public class PhotoProcessor { |

|  |  |
| --- | --- |
| 17 |  |
| 18 | public static void  generateImage(String imagePath, String origName, String targetName,int scalabity){ |

|  |  |
| --- | --- |
| 19 | String origImage =   null; |
| 20 | String targetImage = null; |

|  |  |
| --- | --- |
| 21 | File origFile = null; |
| 22 | BufferedImage buffImg = null; |

|  |  |
| --- | --- |
| 23 | File targetFile = null; |
| 24 | try{ |

|  |  |
| --- | --- |
| 25 | origImage =   imagePath + "/" + origName; |
| 26 | targetImage = imagePath + "/" + targetName; |

|  |  |
| --- | --- |
| 27 | origFile = new File(origImage); |
| 28 | buffImg = ImageIO.read(origFile); |

|  |  |
| --- | --- |
| 29 | buffImg = Scalr.resize(buffImg, Scalr.Method.SPEED, scalabity); |
| 30 | targetFile = new File(targetImage); |

|  |  |
| --- | --- |
| 31 | ImageIO.write(buffImg, "jpeg", targetFile); |
| 32 |  |

|  |  |
| --- | --- |
| 33 | }catch (Exception e){ |
| 34 | System.out.println("Exception in processing image : " + e); |

|  |  |
| --- | --- |
| 35 | }finally { |
| 36 | buffImg = null; |

|  |  |
| --- | --- |
| 37 |  |
| 38 | } |

|  |  |
| --- | --- |
| 39 | } |
| 40 | } |

PhotoFile.java

|  |  |
| --- | --- |
| 01 | package com.aranin.adconnect.util.aws; |
| 02 |  |

|  |  |
| --- | --- |
| 03 | /\*\* |
| 04 | \* Created by IntelliJ IDEA. |

|  |  |
| --- | --- |
| 05 | \* User: Niraj Singh |
| 06 | \* Date: 3/19/13 |

|  |  |
| --- | --- |
| 07 | \* Time: 12:29 PM |
| 08 | \* To change this template use File | Settings | File Templates. |

|  |  |
| --- | --- |
| 09 | \*/ |
| 10 | public class PhotoFile { |

|  |  |
| --- | --- |
| 11 | private String origName; |
| 12 | private String targetName; |

|  |  |
| --- | --- |
| 13 | public String imagePath; |
| 14 |  |

|  |  |
| --- | --- |
| 15 | public String getOrigName() { |
| 16 | return origName; |

|  |  |
| --- | --- |
| 17 | } |
| 18 |  |

|  |  |
| --- | --- |
| 19 | public void setOrigName(String origName) { |
| 20 | this.origName = origName; |

|  |  |
| --- | --- |
| 21 | } |
| 22 |  |

|  |  |
| --- | --- |
| 23 | public String getTargetName() { |
| 24 | return targetName; |

|  |  |
| --- | --- |
| 25 | } |
| 26 |  |

|  |  |
| --- | --- |
| 27 | public void setTargetName(String targetName) { |
| 28 | this.targetName = targetName; |

|  |  |
| --- | --- |
| 29 | } |
| 30 |  |

|  |  |
| --- | --- |
| 31 | public String getImagePath() { |
| 32 | return imagePath; |

|  |  |
| --- | --- |
| 33 | } |
| 34 |  |

|  |  |
| --- | --- |
| 35 | public void setImagePath(String imagePath) { |
| 36 | this.imagePath = imagePath; |

|  |  |
| --- | --- |
| 37 | } |
| 38 |  |

|  |  |
| --- | --- |
| 39 | public String toString(){ |
| 40 | return origName + "," +  targetName + "," + imagePath; |

|  |  |
| --- | --- |
| 41 | } |
| 42 | } |

SQSPhotoManager.java

[view source](https://www.javacodegeeks.com/2013/06/working-with-amazon-simple-queue-service-using-java.html#viewSource)[print](https://www.javacodegeeks.com/2013/06/working-with-amazon-simple-queue-service-using-java.html#printSource)[?](https://www.javacodegeeks.com/2013/06/working-with-amazon-simple-queue-service-using-java.html#about)

|  |  |
| --- | --- |
| 01 | package com.aranin.adconnect.util.aws; |
| 02 |  |

|  |  |
| --- | --- |
| 03 | import com.amazonaws.services.sqs.model.Message; |
| 04 |  |

|  |  |
| --- | --- |
| 05 | import java.util.List; |
| 06 | import java.util.StringTokenizer; |

|  |  |
| --- | --- |
| 07 |  |
| 08 | /\*\* |

|  |  |
| --- | --- |
| 09 | \* Created by IntelliJ IDEA. |
| 10 | \* User: Niraj Singh |

|  |  |
| --- | --- |
| 11 | \* Date: 3/20/13 |
| 12 | \* Time: 11:38 AM |

|  |  |
| --- | --- |
| 13 | \* To change this template use File | Settings | File Templates. |
| 14 | \*/ |

|  |  |
| --- | --- |
| 15 | public class SQSPhotoManager implements Runnable{ |
| 16 | private String queueUrl; |

|  |  |
| --- | --- |
| 17 | public static void main(String[] args){ |
| 18 | AWSSimpleQueueServiceUtil awssqsUtil =   AWSSimpleQueueServiceUtil.getInstance(); |

|  |  |
| --- | --- |
| 19 | /\*\* |
| 20 | \* 1. get the url for your photo queue |

|  |  |
| --- | --- |
| 21 | \*/ |
| 22 | String queueUrl  = awssqsUtil.getQueueUrl(awssqsUtil.getQueueName()); |

|  |  |
| --- | --- |
| 23 | System.out.println("queueUrl : " + queueUrl); |
| 24 |  |

|  |  |
| --- | --- |
| 25 | /\*\* |
| 26 | \* 2. Add a photo to the queue to be processed |

|  |  |
| --- | --- |
| 27 | \*/ |
| 28 |  |

|  |  |
| --- | --- |
| 29 | PhotoFile photo = new PhotoFile(); |
| 30 | photo.setImagePath("C:/Users/Public/Pictures/Sample Pictures"); |

|  |  |
| --- | --- |
| 31 | photo.setOrigName("Tree.jpg"); |
| 32 | photo.setTargetName("Tree\_thumb.jpg"); |

|  |  |
| --- | --- |
| 33 |  |
| 34 | /\*\* |

|  |  |
| --- | --- |
| 35 | \* 3. set the photofile in queue for processing |
| 36 | \*/ |

|  |  |
| --- | --- |
| 37 |  |
| 38 | awssqsUtil.sendMessageToQueue(queueUrl, photo.toString()); |

|  |  |
| --- | --- |
| 39 |  |
| 40 | /\*\* |

|  |  |
| --- | --- |
| 41 | \* get the messages from queue |
| 42 | \*/ |

|  |  |
| --- | --- |
| 43 |  |
| 44 | Thread managerthread = new Thread(new SQSPhotoManager(queueUrl),"T2"); |

|  |  |
| --- | --- |
| 45 | managerthread.start(); |
| 46 |  |

|  |  |
| --- | --- |
| 47 | } |
| 48 |  |

|  |  |
| --- | --- |
| 49 | public SQSPhotoManager(String queueUrl){ |
| 50 | this.queueUrl = queueUrl; |

|  |  |
| --- | --- |
| 51 | } |
| 52 |  |

|  |  |
| --- | --- |
| 53 | @Override |
| 54 | public void run() { |

|  |  |
| --- | --- |
| 55 | AWSSimpleQueueServiceUtil awssqsUtil =   AWSSimpleQueueServiceUtil.getInstance(); |
| 56 | boolean flag = true; |

|  |  |
| --- | --- |
| 57 | while(flag){ |
| 58 | List<Message> messages =  awssqsUtil.getMessagesFromQueue(this.queueUrl); |

|  |  |
| --- | --- |
| 59 | if(messages == null || messages.size() == 0){ |
| 60 | try { |

|  |  |
| --- | --- |
| 61 | Thread.sleep(1000); |
| 62 | } catch (InterruptedException e) { |

|  |  |
| --- | --- |
| 63 | e.printStackTrace();  //To change body of catch statement use File | Settings | File Templates. |
| 64 | } |

|  |  |
| --- | --- |
| 65 | }else{ |
| 66 | flag = false; |

|  |  |
| --- | --- |
| 67 | for (Message message : messages) { |
| 68 | String messagePhoto = message.getBody(); |

|  |  |
| --- | --- |
| 69 | System.out.println("photo to be processed : " + messagePhoto); |
| 70 | StringTokenizer photoTokenizer = new StringTokenizer(messagePhoto,","); |

|  |  |
| --- | --- |
| 71 | String source = null; |
| 72 | String target = null; |

|  |  |
| --- | --- |
| 73 | String path = null; |
| 74 |  |

|  |  |
| --- | --- |
| 75 | source = photoTokenizer.nextToken(); |
| 76 | target = photoTokenizer.nextToken(); |

|  |  |
| --- | --- |
| 77 | path = photoTokenizer.nextToken(); |
| 78 | System.out.println("source : " + source); |

|  |  |
| --- | --- |
| 79 | System.out.println("target : " + target); |
| 80 | System.out.println("path : " + path); |

|  |  |
| --- | --- |
| 81 | /\*\* |
| 82 | \* generate thumbmail within 150\*150 container |

|  |  |
| --- | --- |
| 83 | \*/ |
| 84 | PhotoProcessor.generateImage(path, source, target, 150); |

|  |  |
| --- | --- |
| 85 | } |
| 86 |  |

|  |  |
| --- | --- |
| 87 | /\*\* |
| 88 | \* finally delete the message |

|  |  |
| --- | --- |
| 89 | \*/ |
| 90 | for (Message message : messages) { |

|  |  |  |
| --- | --- | --- |
| 91 | awssqsUtil.deleteMessageFromQueue(this.queueUrl, message); | |
| 92 | } |

|  |  |
| --- | --- |
| 93 |  |
| 94 | } |

|  |  |
| --- | --- |
| 95 | } |
| 96 | } |

|  |  |
| --- | --- |
| 97 | } |

This will form the core for your PhotoProcessor application using SQS. This code has a glaring drawback. It polls on the SQS using a thread, it would be great if you can create a listener in your code which can subscribe to your queue and take necessary action when new message arrives. That is indeed the subject of my next post. Till then feel free to bombard me with questions, together we can find answer to those.