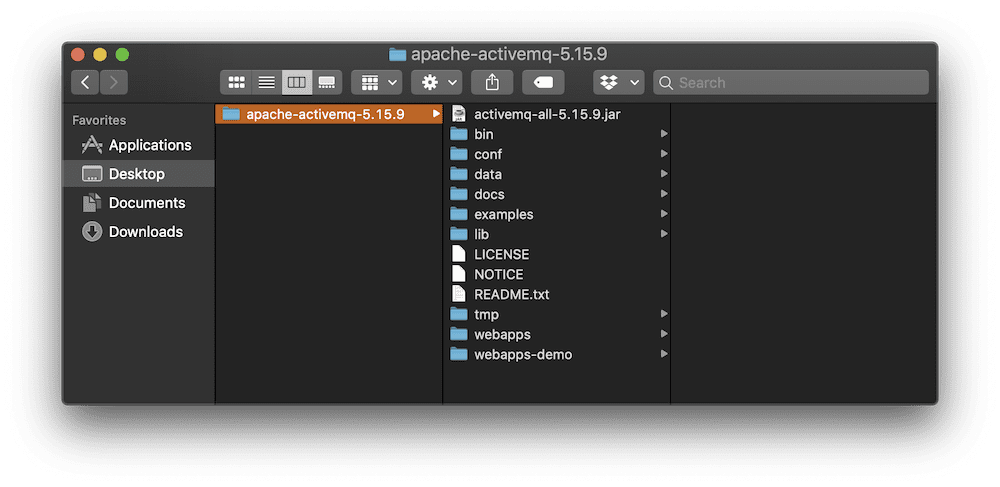
Message-Driven Beans with ActiveMQ and TomEE

The example contains three MDBs and one stand-alone JMS client. The purpose of the system is to extract image URLs and hyperlinks from a web page and process them independently. Ok, let’s get started!

**1. Download ActiveMQ 5 “Classic”**

* Go to the Apache ActiveMQ [web page](https://activemq.apache.org/components/classic/download/) and download ActiveMQ 5 for your operating system.
* Copy the archive to your Desktop and Unzip it.

You should now have a Desktop/apache-activemq-5.a.b directory (where a.b is minor and point version e.g. Desktop/apache-activemq-5.15.9)



* Open a console window and navigate ActiveMQ directory.

$ cd Desktop/apache-activemq-5.a.b/

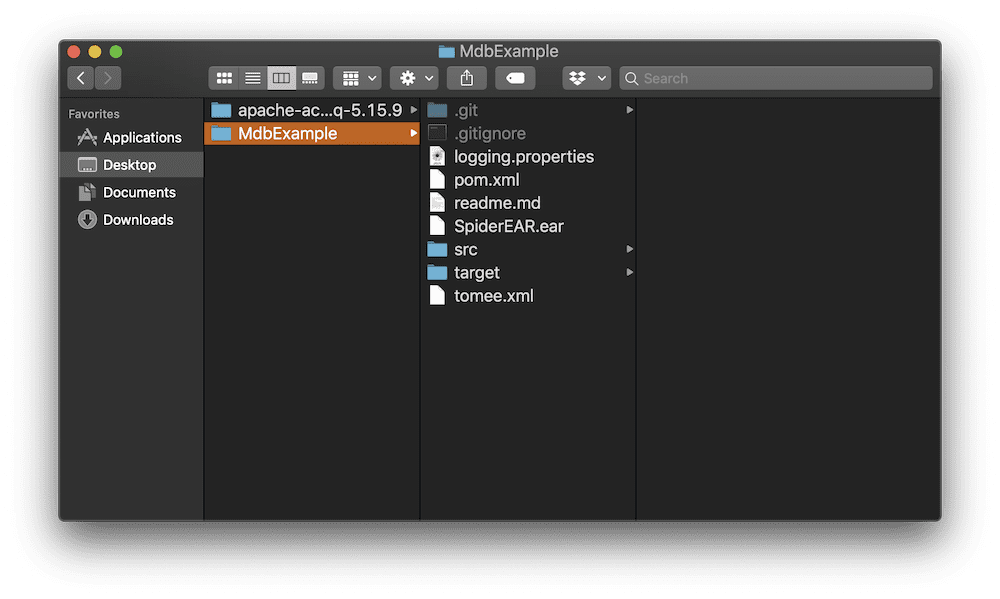
* Start the ActiveMQ broker.

$ bin/activemq console

**Step 2: Download the MDB Example file and Install**

* [Download](https://github.com/tomitribe/MdbExample/archive/master.zip) the example zip.
* Extract the example zip and copy it to your Desktop.

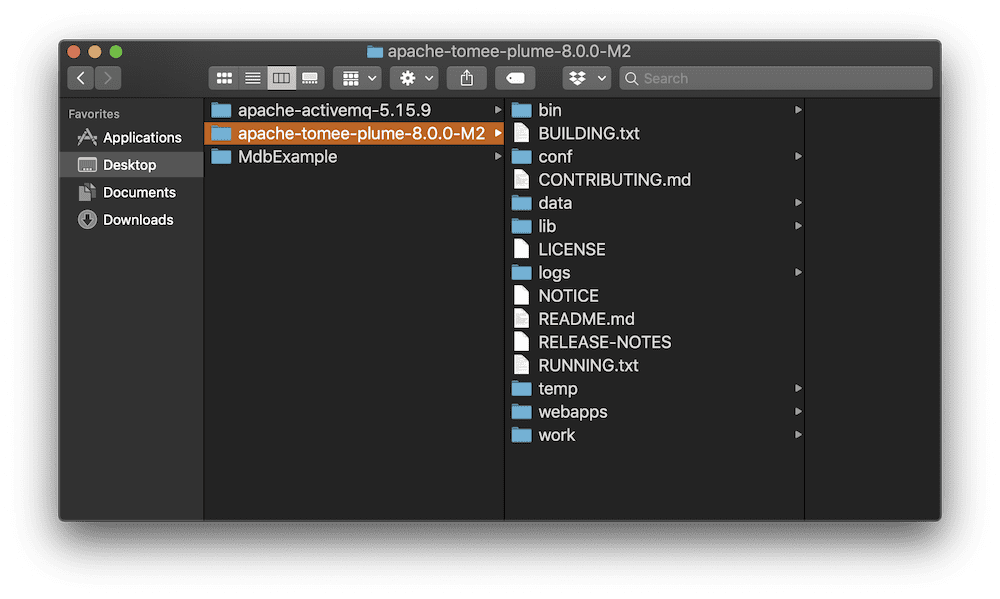
After you’ve downloaded and expanded the MdbExample on your Desktop it should look something like this.



**Step 3: Download, configure, and start TomEE Plume**

* [Download](http://tomee.apache.org/download-ng.html) the latest version of the TomEE plume application server.
* Extract the TomEE Archive and copy it to your Desktop.

You should now have a directory with a name like apache-tomee-plume-<whatever version> on your desktop.



* Copy configuration files into TomEE Directory.

In the MdbExample directory that you downloaded, you will find two configuration files: tomee.xml and logging.properties. These need to be copied over to the apache-tomee-plume-<whatever version>/conf directory. When you copy these files you will be overwriting the original files, which is what you want.

* Open a new console windows and navigate to the MdbExample directory

$ cd Desktop/MdbExample

* Copy the tomee.xml file to the apache-tomee-plume-<whatever version>/conf directory. Be sure to change the <whatever version> to the proper version as was done below.

$ cp tomee.xml ../apache-tomee-plume-8.0.0-M2/conf

The tomee.xml file tells the TomEE application server how to connect to the ActiveMQ broker and the names of JMS topic and queue that will be used with the MDB container system.

* In the same console window enter in the following command

$ cp logging.properties ../apache-tomee-plume-8.0.0-M2/conf

The logging.properties file includes changes that suppress most of the output form TomEE when running in the console window. This will make it easier to see the output from the MDBs you develop.

* Copy the SpiderEAR.ear file from the MdbExample directory.

$ cp SpiderEAR.ear ../apache-tomee-plume-8.0.0-M2/webapps

The SpiderEAR.ear is a Java EE Enterprise Archive (JAR) that contains the three MDBs that you will be deploying into the TomEE application server. The source code for these MDBs can be found in the MdbExample/src/main/java directory on your Desktop or in the MdbExample GitHub [repository](https://github.com/tomitribe/MdbExample/tree/master/src/main/java/example).

* Open a third console windows and navigate to the TomEE directory.

$ cd Desktop/apache-tomee-plume-8.0.0-M2

* Start TomEE by executing the Catalina run command.

$ bin/catalina.sh run

After the TomEE application server starts there will be a lot of output on the screen and then it will pause. After the pause TomEE will detect the SpiderEAR.ear archive and deploy it resulting in some more output (see example below) saying that the SpiderMDB, ImageMDB, and LinkMDB MDBs have been deployed.

24-May-2019 08:37:42.369 INFO [ContainerBackgroundProcessor[StandardEngine[Catalina]]] sun.reflect.DelegatingMethodAccessorImpl.invoke Deploying web application directory [/Users/tomitribe/Desktop/apache-tomee-plume-8.0.0-M2/webapps/SpiderEAR]

24-May-2019 08:37:42.408 INFO [ContainerBackgroundProcessor[StandardEngine[Catalina]]] org.apache.jasper.servlet.TldScanner.scanJars At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scanning can improve startup time and JSP compilation time.

24-May-2019 08:37:42.414 INFO [ContainerBackgroundProcessor[StandardEngine[Catalina]]] sun.reflect.DelegatingMethodAccessorImpl.invoke Deployment of web application directory [/Users/tomitribe/Desktop/apache-tomee-plume-8.0.0-M2/webapps/SpiderEAR] has finished in [45] ms

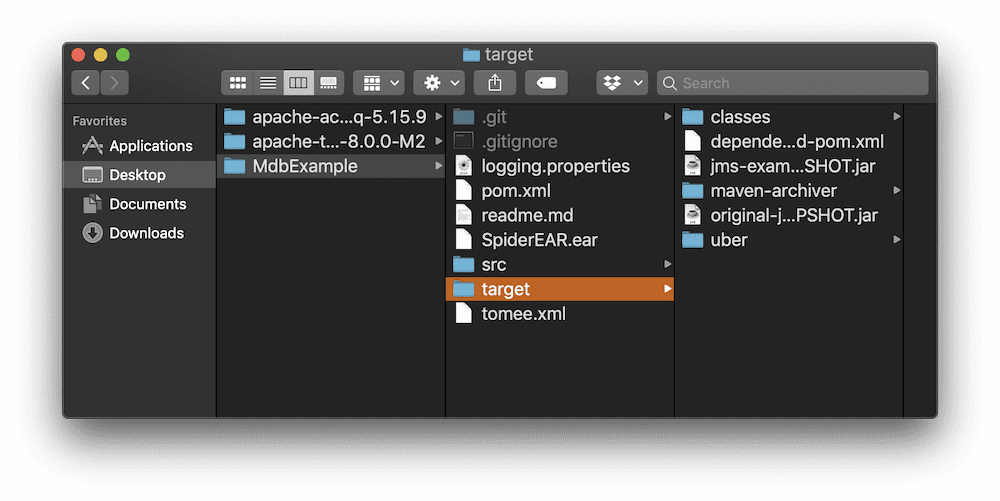
**Step 4: Start the Producer and send some Web URLs**

The Desktop/MdbExample directory contains the JMS Client application that you will use to initiate the program. It will send a message to the MDBs in the TomEE container.

* Switch to the console windows in the Desktop/MdbExample directory and run the Maven install command

$ mvn install

When the mvn install command is finished running you should be able to see a new target directory in the MdbExample directory like that shown below.

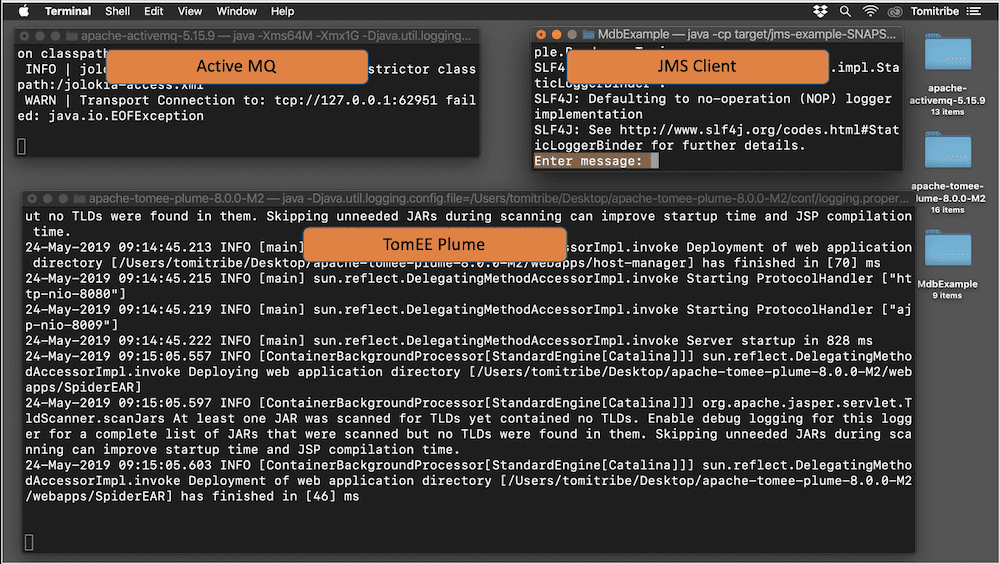


Assuming the target directory has been created you can now start the JMS client in the MdbExample directory.

* Start the JMS Client using the console window in the MdbExample directory.

$ java -cp target/jms-example-SNAPSHOT.jar example.Producer Topic

At this point your Desktop should look something like the image below (without the orange labels). There are three open console windows: One is running the ActiveMQ, another is running TomEE Plume and the third is running the JMS Client.



At the “Enter message:” prompt on the JMS client window enter any web site address and see the results. For this first run, let’s use the Tomitribe web site. Type http://www.tomitribe.com at the prompt as shown below and hit “Enter”

* Enter the http://www.tomitribe.com Web URL at the prompt.

$ java -cp target/jms-example-SNAPSHOT.jar example.Producer Topic

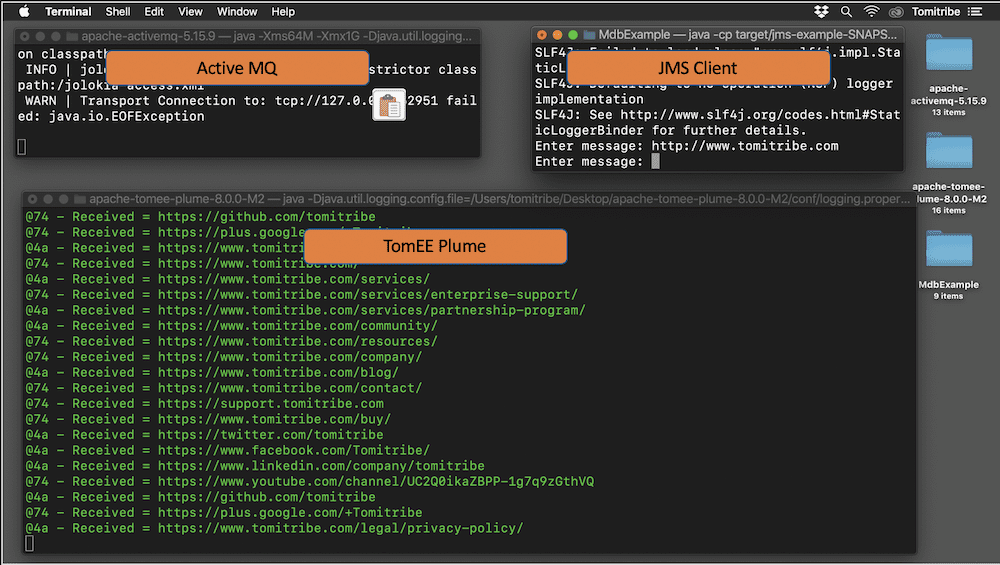
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

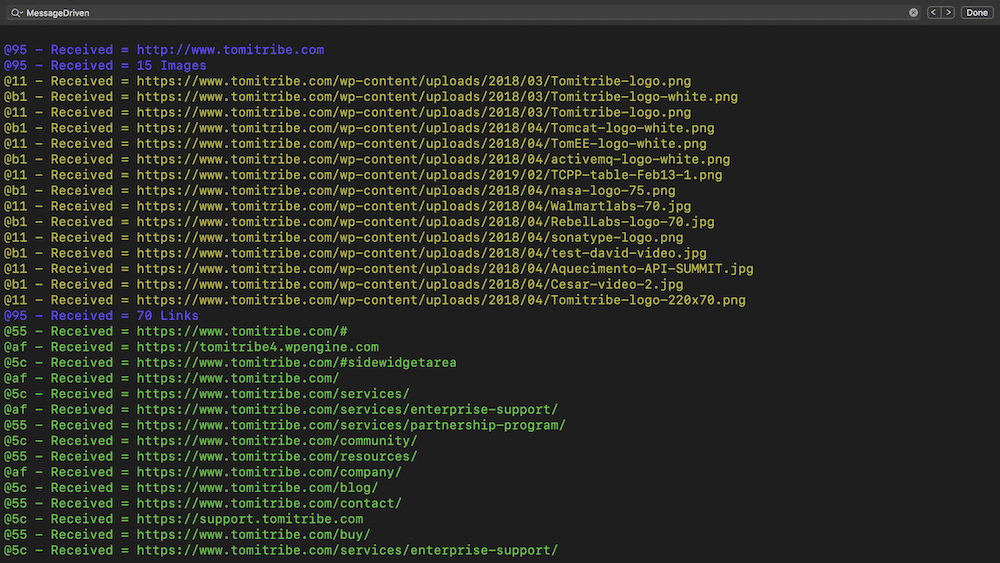
Enter message: http://www.tomitribe.com

After you enter the Tomitribe web site address the program will run and you will see lots of colored output in the TomEE Plume console windows like that shown below.



* Review the output.

Now take a closer look at the console windows in which TomEE is running. Scroll to start of the colored lines. It will look something like this:



The **blue** text is the output from the SpiderMDB. It shows that one of its instances received the URL from the JMS Client and then parsed the web page to determine that there are 15 images. After the blue lines is the output from the ImageMDB which prints every message it received in **orange**. One line of orange output for each image. The next blue line is the count of links and each **green** line represents a link that was sent in a text message from the SpiderMDB to the LinkMDB.

Every JMS message is handled by a single MDB instance, which is identified by the “@nn” tag at the start of each line. This shows that the work is divided up among many instances and that instances can and are reused – this enables MDBs to scale handling many messages and operations in a short period of time. Finally, the demo illustrates how MDB-to-MDB communication works providing enormous flexibility in a system architecture.

Congratulations you just ran the entire MdbExample and it should have taken you about five minutes.

**Step 5: Review the Source Code**

The source code for the JMS client and the three MDBs (SpiderMDB, ImageMDB, LinkMDB) are contained in the example file you downloaded and on [GitHub](https://github.com/tomitribe/MdbExample/tree/master/src/main/java/example). These source files contain a lot of comments which explain the code and how it works. Take the time to review that source code if want to see how MDBs process incoming JMS messages and send their own messages.