MDB Hello World With WildFly And ActiveMQ With RAR Deployment

# Introduction

This is the 2nd increment of the [MDB Series](http://palashray.com/java/java/mdb/). In [Part 1](http://palashray.com/mdb-hello-world-with-wildfly-and-embedded-artemis-mq/), we spoke about what is an MDB? We then, took a simple design problem: a servlet posts user data to a Message Queue. The Message Broker then sends a notification to a MDB. We used the Artemis MQ, which comes embedded in Wildfly, as our Message Broker.

In this increment, we are going to use [Apache ActiveMQ](https://activemq.apache.org/), running as a separate service, as our Message Broker. We are going to use the RAR archive, or specifically, the activemq-rar archive, to communicate between Wildfly and ActiveMQ.

## What is a RAR?

In simple terms, a RAR Archive or a [Resource Adapter Archive](https://docs.oracle.com/javaee/6/tutorial/doc/gipgl.html), is a JEE Component that helps a JEE Server to communicate with an external Resource. The RAR implements the JEE Connector architecture specs. It can be either deployed within a EAR file, or independently. In our example, we are going to deploy the *activemq.rar* independently.

# Implementation Details

The structure will be very similar to what we had in the [1st increment of the series](http://palashray.com/mdb-hello-world-with-wildfly-and-embedded-artemis-mq/), with subtle differences as highlighted below.

## The pom.xml

The pom is a slight improvement, with minor version upgrades for the below dependencies:

|  |
| --- |
| <**dependency**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>javax.servlet-api</**artifactId**>  <**version**>3.1.0</**version**>  <**scope**>provided</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>javax.jms</**groupId**>  <**artifactId**>javax.jms-api</**artifactId**>  <**version**>2.0.1</**version**>  <**scope**>provided</**scope**>  </**dependency**>  <**dependency**>  <**groupId**>javax.ejb</**groupId**>  <**artifactId**>javax.ejb-api</**artifactId**>  <**version**>3.2.2</**version**>  <**scope**>provided</**scope**>  </**dependency**> |

We have removed the ***org.wildfly:wildfly-jms-client-bom*** dependency, as it is no longer needed.

Apart from the above changes, all other elements like the war-plugin etc., remains the same.

## Defining the JMS Queue

The ***WEB-INF/embedded-artemis-jms.xml*** file is no longer needed, instead, the *Queue* is defined in the *standalone.xml* configuration, as described in a later section.

### mdb.properties

There are slight changes to the JNDI names of the Queue and the ConnectionFactory:

|  |
| --- |
| ACTIVEMQ\_QUEUE\_LOOKUP=java:/queue/HELLOWORLDMDBQueue ACTIVEMQ\_JMS\_CONNECTION\_FACTORY=java:/ActiveMQConnectionFactory WILDFLY\_USER=user WILDFLY\_PASSWORD=user123 |

## MdbConfig

This remains the same, except for the change in the property file keys.

## AuthorRestController

This remains the same, without any changes.

## Configuring the MDB to Receive JMS Messages

The *AuthorRequestListenerBean* remains the same structurally, with a few minor changes in the *@MessageDriven* annotation as shown below:

|  |
| --- |
| @MessageDriven(name = "HelloWorldQueueMDB",  activationConfig = {  @ActivationConfigProperty(propertyName = "destinationType",  propertyValue = "javax.jms.Queue"),  @ActivationConfigProperty(propertyName = "destination",  propertyValue = "HELLOWORLDMDBQueue"),  @ActivationConfigProperty(propertyName = "acknowledgeMode",  propertyValue = "Auto-acknowledge") }) public class AuthorRequestListenerBean implements MessageListener { |

## The standalone.xml

As before, we would need to modify the default *standalone.xml* that comes bundled with WildFly. We would be taking Wildfly 18 for our reference. Copy the *$WILDFLY\_HOME/standalone/configuration/standalone.xml* and rename it to *standalone-with-external-activemq-rar-deployment.xml*. You can see the differences here:

<https://github.com/paawak/blog/commit/2013c222f8d847e7d41c8910c92ff68f67eef4a8?diff=unified>

The differences are highlighted below.

### Adding the ActiveMQ Messaging Extension

Under the *extensions* section, we would need to add the below extension:

|  |
| --- |
| <**extension** module="org.wildfly.extension.messaging-activemq"/> |

### Defining the ActiveMQ Messaging SubSystem

Like before, we would need to define the new subsystem for ActiveMQ Messaging:

|  |
| --- |
| <**subsystem** xmlns="urn:jboss:domain:messaging-activemq:3.0">  <**server** name="default">  <**security-setting** name="#">  <**role** name="guest" send="true" consume="true" create-non-durable-queue="true" delete-non-durable-queue="true"/>  </**security-setting**>  <**address-setting** name="#" dead-letter-address="jms.queue.DLQ" expiry-address="jms.queue.ExpiryQueue" max-size-bytes="10485760" page-size-bytes="2097152" message-counter-history-day-limit="10"/>  <**http-connector** name="http-connector" socket-binding="http" endpoint="http-acceptor"/>  <**http-connector** name="http-connector-throughput" socket-binding="http" endpoint="http-acceptor-throughput">  <**param** name="batch-delay" value="50"/>  </**http-connector**>  <**in-vm-connector** name="in-vm" server-id="0">  <**param** name="buffer-pooling" value="false"/>  </**in-vm-connector**>  <**http-acceptor** name="http-acceptor" http-listener="default"/>  <**http-acceptor** name="http-acceptor-throughput" http-listener="default">  <**param** name="batch-delay" value="50"/>  <**param** name="direct-deliver" value="false"/>  </**http-acceptor**>  <**in-vm-acceptor** name="in-vm" server-id="0">  <**param** name="buffer-pooling" value="false"/>  </**in-vm-acceptor**>  <**jms-queue** name="ExpiryQueue" entries="java:/jms/queue/ExpiryQueue"/>  <**jms-queue** name="DLQ" entries="java:/jms/queue/DLQ"/>  <**connection-factory** name="InVmConnectionFactory" entries="java:/ConnectionFactory" connectors="in-vm"/>  <**connection-factory** name="RemoteConnectionFactory" entries="java:jboss/exported/jms/RemoteConnectionFactory" connectors="http-connector"/>  <**pooled-connection-factory** name="activemq-ra" entries="java:/JmsXA java:jboss/DefaultJMSConnectionFactory" connectors="in-vm" transaction="xa"/>  </**server**>  </**subsystem**> |

### Adding the MDB Resource Adapter

As before, under the subsystem ejb3 section, we would need to add the details of the *Resource Adapter* to be used to communicate with the ActiveMQ Messaging system.

<**mdb**>  
 <**resource-adapter-ref** resource-adapter-name="activemq-rar-5.15.4.rar"/>  
 <**bean-instance-pool-ref** pool-name="mdb-strict-max-pool"/>  
 </**mdb**>

### Describing the ActiveMQ Resource Adapter

This is a new section that we would need to add under the *resource-adapters* to describe the ActiveMQ RAR. Note that we would need to specify the ActiveMQ broker details, as well as the Queue JNDI lookup details here.

|  |
| --- |
| <**subsystem** xmlns="urn:jboss:domain:resource-adapters:5.0">  <**resource-adapters**>   <**resource-adapter** id="activemq-rar-5.15.4.rar">   <**archive**>activemq-rar-5.15.4.rar</**archive**>  <**transaction-support**>XATransaction</**transaction-support**>   <**config-property** name="UseInboundSession">false</**config-property**>   <**config-property** name="Password">admin</**config-property**>   <**config-property** name="UserName">admin</**config-property**>   <**config-property** name="ServerUrl">tcp://localhost:61616?jms.rmIdFromConnectionId=true</**config-property**>   <**connection-definitions**>   <**connection-definition** class-name="org.apache.activemq.ra.ActiveMQManagedConnectionFactory"  jndi-name="java:/ActiveMQConnectionFactory" enabled="true" pool-name="ActiveMQConnectionFactory">   <**xa-pool**>   <**min-pool-size**>1</**min-pool-size**>   <**max-pool-size**>20</**max-pool-size**>   <**prefill**>false</**prefill**>   <**is-same-rm-override**>false</**is-same-rm-override**>   </**xa-pool**>   </**connection-definition**>   </**connection-definitions**>   <**admin-objects**>   <**admin-object** class-name="org.apache.activemq.command.ActiveMQQueue"  jndi-name="java:/queue/HELLOWORLDMDBQueue"  use-java-context="true" pool-name="HELLOWORLDMDBQueue">   <**config-property** name="PhysicalName">HELLOWORLDMDBQueue</**config-property**>   </**admin-object**>   </**admin-objects**>   </**resource-adapter**>   </**resource-adapters**>   </**subsystem**> |

This is how the completed *standalone-with-external-activemq-rar-deployment.xml* looks like:

<https://github.com/paawak/blog/blob/master/code/mdb-demo/wildfly/external-activemq/rar-archive-deployment/mdb-activemq-rar-demo-spring/src/main/wildfly/standalone-with-external-activemq-rar-deployment.xml>

# Source Code

The complete source can be found here:

<https://github.com/paawak/blog/tree/master/code/mdb-demo/wildfly/external-activemq/rar-archive-deployment/mdb-activemq-rar-demo-spring>

# Running the Demo

## Building the war

To build the war file, we do:

|  |
| --- |
| mvn clean package |

## Deploying the RAR

The ActiveMQ RAR is no longer included in the [ActiveMQ Distribution](https://activemq.apache.org/components/classic/download/), but, can be downloaded from here: <https://repo1.maven.org/maven2/org/apache/activemq/activemq-rar/>

Just copy the RAR file into the *$WILDFLY\_HOME/standalone/deployments* directory.

## Creating a Guest User in WildFly

Follow the steps as described in the [previous increment](http://palashray.com/mdb-hello-world-with-wildfly-and-embedded-artemis-mq/).

## Deploying the war

Copy the *mdb-activemq-rar-demo-spring.war* into the directory *$WILDFLY\_HOME/standalone/deployments*.

## Starting the ActiveMQ Locally

Download and unpack the [ActiveMQ distribution](https://activemq.apache.org/components/classic/download/) for your platform. Then start it with

|  |
| --- |
| bin/activemq start |

## Starting WildFly with our custom configuration

Copy the *src/main/wildfly/standalone-with-external-activemq-rar-deployment.xml* into the *$WILDFLY\_HOME/standalone/configuration/* directory. Then start WildFly with the below command:

|  |
| --- |
| $WILDFLY\_HOME/bin/standalone.sh -c standalone-with-external-activemq-rar-deployment.xml |

After WildFly starts successfully, you can access the Author page here:

[http://localhost:8080/mdb-activemq-rar-demo-spring/author.html](#)

As before, you should be able to post a Message through the Author Page, and the same would appear on the Console from the MDB as a Json Message.

|  |
| --- |
| 23:32:55,072 INFO [com.swayam.demo.mdb.rar.spring.listener.AuthorRequestListenerBean] (default-threads - 2) Text message received: {"authorId":1,"authorFirstName":"aaaa","authorLastName":"bbbb","genreShortName":"cccc","genreName":"dddd"} |