MDB Hello World Using JEE CDI On WildFly With ActiveMQ RAR Deployed as a Module

# Introduction

This is the 4th 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 [Part 2](http://palashray.com/mdb-hello-world-with-wildfly-and-activemq-with-rar-deployment/), we demonstrated how to integrate a MDB with an external [Apache ActiveMQ](https://activemq.apache.org/) broker running in a separate process. We used the ActiveMQ RAR deployed independently to achieve integration between Wildfly and ActiveMQ.

In [Part 3](http://palashray.com/mdb-hello-world-with-wildfly-and-activemq-rar-deployed-as-a-module/), we deployed the Active MQ RAR as a Module in Wildfly and communicated with an Active MQ Broker running in a separate process.

In this increment, we are going to show how to achieve the same thing as [Part 3](http://palashray.com/mdb-hello-world-with-wildfly-and-activemq-rar-deployed-as-a-module/), but using a more bare bones approach. Instead of using Spring for our dependency injection, we would be using the CDI, and instead of a Spring Controller, we are going to use a HttpServlet.

## What is CDI?

The CDI or the [Context and Dependency Injection](https://docs.oracle.com/javaee/6/tutorial/doc/giwhl.html), is a dependency injection framework provided by JEE Containers. It is a part of the JEE Specification. With CDI, we could *inject* any JNDI Resource or any Container Managed Beans, pretty much like we do in the Spring Framework. It also provides various lifecycle hooks like PostConstruct and PreDestroy.

# Implementation Details

We will use the sources from the previous series as a starting point. The previous source can be found here:

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

We would rename the project to *mdb-activemq-module-demo-plain*.

## The pom.xml

We would remove all Spring related dependencies. The only addition would be *javax.annotation-api* dependency for the CDI annotations.

|  |
| --- |
| <**dependency**>  <**groupId**>javax.annotation</**groupId**>  <**artifactId**>javax.annotation-api</**artifactId**>  <**version**>1.3.2</**version**>  <**scope**>provided</**scope**>  </**dependency**> |

This is how the pom.xml would look like:

<https://github.com/paawak/blog/blob/master/code/mdb-demo/wildfly/external-activemq/module-deployment/mdb-activemq-module-demo-plain/pom.xml>

## The Servlet

The *HelloWorldMDBServletClient* extends the *HttpServlet*. It uses CDI for injecting the JMS *ConnectionFactory* and *Queue* as shown below:

|  |
| --- |
| @WebServlet("/rest/author") public class HelloWorldMDBServletClient extends HttpServlet {   private static final long serialVersionUID = 1L;   private static final Logger LOGGER = LoggerFactory.getLogger(HelloWorldMDBServletClient.class);   @Resource(mappedName = "java:/ActiveMQConnectionFactory")  private ConnectionFactory connectionFactory;   @Resource(mappedName = "java:/queue/HELLOWORLDMDBQueue")  private Queue queue;   @Override  protected void doPost(HttpServletRequest req, HttpServletResponse resp)  throws ServletException, IOException {   AuthorRequest authorRequest = new AuthorRequest();  authorRequest.setAuthorId(Long.valueOf(req.getParameter("authorId")));  authorRequest.setAuthorFirstName(req.getParameter("authorFirstName"));  authorRequest.setAuthorLastName(req.getParameter("authorLastName"));  authorRequest.setGenreShortName(req.getParameter("genreShortName"));  authorRequest.setGenreName(req.getParameter("genreName"));   LOGGER.debug(" Received authorRequest: {}", authorRequest);   resp.setContentType("text/html");  PrintWriter out = resp.getWriter();  out.write(  "<h1>Quickstart: Example demonstrates the use of <strong>JMS 2.0</strong> and <strong>EJB 3.2 Message-Driven Bean</strong> in JBoss EAP.</h1>");  try {   Connection connection = connectionFactory.createConnection();  Session session = connection.createSession(true, Session.AUTO\_ACKNOWLEDGE);  MessageProducer producer = session.createProducer(queue);   out.write("<p>Sending messages to <em>" + queue + "</em></p>");  out.write("<h2>The following message will be sent to the destination:</h2>");  String text = new ObjectMapper().writeValueAsString(authorRequest);  TextMessage message = session.createTextMessage(text);  producer.send(message);  out.write("Message: " + text + "</br>");  session.close();  connection.close();  out.write(  "<p><i>Go to your JBoss EAP server console or server log to see the result of messages processing.</i></p>");  } catch (JMSException e) {  throw new RuntimeException(e);  } finally {  if (out != null) {  out.close();  }  }  }  } |

Note the use of the *@WebServlet* annotation for path mapping.

## The MDB

The MDB remains the same from the [previous series](https://github.com/paawak/blog/blob/master/code/mdb-demo/wildfly/external-activemq/module-deployment/mdb-activemq-module-demo-spring/src/main/java/com/swayam/demo/mdb/spring/listener/AuthorRequestListenerBean.java).

## The standalone.xml

The standalone.xml is identical to the [previous series](https://github.com/paawak/blog/blob/master/code/mdb-demo/wildfly/external-activemq/module-deployment/mdb-activemq-module-demo-spring/src/main/wildfly/standalone-with-activemq-module-deployment-spring.xml), just renamed to *standalone-with-activemq-module-deployment-plain.xml*.

# Source Code

The complete source can be found here:

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

# Running the Demo

For preparing the ActiveMQ RAR as a module, adding users and running ActiveMQ locally, refer to the [previous series](http://palashray.com/mdb-hello-world-with-wildfly-and-activemq-rar-deployed-as-a-module/). Identical steps are to be followed.

# Starting WildFly

Now we can start Wildfly with our custom configuration as below:

WILDFLY\_HOME/bin/standalone.sh -c standalone-with-activemq-module-deployment-plain.xml

After Wildfly starts successfully, you can access the Author page with the below URL:

[http://localhost:8080/mdb-activemq-module-demo-plain/author.jsp](#)

# Problems Faced

## @JMSDestinationDefinition not supported by Wildfly 18

This code was running on Wildfly 13. At that time, the *HelloWorldMDBServletClient* servlet looked like this:

|  |
| --- |
| @JMSDestinationDefinitions(value = { @JMSDestinationDefinition(name = HelloWorldMDBServletClient.QUEUE\_JNDI\_NAME, interfaceName = "javax.jms.Queue", destinationName = "HELLOWORLDMDBQueue") }) @WebServlet("/rest/author") public class HelloWorldMDBServletClient extends HttpServlet { ... |

The source can be found here:

<https://github.com/paawak/blog/blob/c6d644c045257eb12b208f5a5036553cd9ba167c/code/mdb-demo/wildfly/external-activemq/module-deployment/mdb-activemq-module-demo-plain/src/main/java/com/swayam/demo/mdb/plain/web/servlet/HelloWorldMDBServletClient.java>

Note the use of the [@JMSDestinationDefinition](https://docs.oracle.com/javaee/7/api/javax/jms/JMSDestinationDefinition.html) for specifying the Queue. In the [standalone.xml](https://github.com/paawak/blog/blob/c6d644c045257eb12b208f5a5036553cd9ba167c/code/mdb-demo/wildfly/external-activemq/module-deployment/mdb-activemq-module-demo-plain/src/main/wildfly/standalone.xml), under the *resource-adapter* we did not have the *admin-objects* section with the Queue details, as it was specified in the Servlet above.

All was well in Wildfly 13. However, as soon as I deployed this in Wildfly 18, I started getting a huge stack trace and the deployment failed.

### Stack Trace

|  |
| --- |
| 07:45:40,398 ERROR [org.jboss.msc.service.fail] (MSC service thread 1-3) MSC000001: Failed to start service jboss.deployment.unit."mdb-activemq-module-demo-plain.war".INSTALL: org.jboss.msc.service.StartException in service jboss.deployment.unit."mdb-activemq-module-demo-plain.war".INSTALL: WFLYSRV0153: Failed to process phase INSTALL of deployment "mdb-activemq-module-demo-plain.war"  at org.jboss.as.server@10.0.3.Final//org.jboss.as.server.deployment.DeploymentUnitPhaseService.start(DeploymentUnitPhaseService.java:183)  at org.jboss.msc@1.4.11.Final//org.jboss.msc.service.ServiceControllerImpl$StartTask.startService(ServiceControllerImpl.java:1739)  at org.jboss.msc@1.4.11.Final//org.jboss.msc.service.ServiceControllerImpl$StartTask.execute(ServiceControllerImpl.java:1701)  at org.jboss.msc@1.4.11.Final//org.jboss.msc.service.ServiceControllerImpl$ControllerTask.run(ServiceControllerImpl.java:1559)  at org.jboss.threads@2.3.3.Final//org.jboss.threads.ContextClassLoaderSavingRunnable.run(ContextClassLoaderSavingRunnable.java:35)  at org.jboss.threads@2.3.3.Final//org.jboss.threads.EnhancedQueueExecutor.safeRun(EnhancedQueueExecutor.java:1982)  at org.jboss.threads@2.3.3.Final//org.jboss.threads.EnhancedQueueExecutor$ThreadBody.doRunTask(EnhancedQueueExecutor.java:1486)  at org.jboss.threads@2.3.3.Final//org.jboss.threads.EnhancedQueueExecutor$ThreadBody.run(EnhancedQueueExecutor.java:1377)  at java.base/java.lang.Thread.run(Thread.java:834) Caused by: java.lang.NullPointerException  at org.jboss.as.naming@18.0.1.Final//org.jboss.as.naming.deployment.ContextNames.bindInfoFor(ContextNames.java:351)  at org.wildfly.extension.messaging-activemq//org.wildfly.extension.messaging.activemq.deployment.JMSConnectionFactoryDefinitionInjectionSource.getDefaulResourceAdapter(JMSConnectionFactoryDefinitionInjectionSource.java:416)  at org.wildfly.extension.messaging-activemq//org.wildfly.extension.messaging.activemq.deployment.JMSDestinationDefinitionInjectionSource.getResourceValue(JMSDestinationDefinitionInjectionSource.java:137)  at org.jboss.as.ee@18.0.1.Final//org.jboss.as.ee.component.deployers.ModuleJndiBindingProcessor.addJndiBinding(ModuleJndiBindingProcessor.java:289)  at org.jboss.as.ee@18.0.1.Final//org.jboss.as.ee.component.deployers.ModuleJndiBindingProcessor$1.handle(ModuleJndiBindingProcessor.java:240)  at org.jboss.as.ee@18.0.1.Final//org.jboss.as.ee.component.ClassDescriptionTraversal.run(ClassDescriptionTraversal.java:54)  at org.jboss.as.ee@18.0.1.Final//org.jboss.as.ee.component.deployers.ModuleJndiBindingProcessor.processClassConfigurations(ModuleJndiBindingProcessor.java:244)  at org.jboss.as.ee@18.0.1.Final//org.jboss.as.ee.component.deployers.ModuleJndiBindingProcessor.deploy(ModuleJndiBindingProcessor.java:158)  at org.jboss.as.server@10.0.3.Final//org.jboss.as.server.deployment.DeploymentUnitPhaseService.start(DeploymentUnitPhaseService.java:176)  ... 8 more  07:45:40,418 INFO [org.infinispan.factories.GlobalComponentRegistry] (MSC service thread 1-1) ISPN000128: Infinispan version: Infinispan 'Infinity Minus ONE +2' 9.4.16.Final 07:45:40,708 INFO [org.jboss.as.clustering.infinispan] (ServerService Thread Pool -- 80) WFLYCLINF0002: Started client-mappings cache from ejb container 07:45:40,761 ERROR [org.jboss.as.controller.management-operation] (Controller Boot Thread) WFLYCTL0013: Operation ("deploy") failed - address: ([("deployment" => "mdb-activemq-module-demo-plain.war")]) - failure description: {  "WFLYCTL0080: Failed services" => {"jboss.deployment.unit.\"mdb-activemq-module-demo-plain.war\".INSTALL" => "WFLYSRV0153: Failed to process phase INSTALL of deployment \"mdb-activemq-module-demo-plain.war\"  Caused by: java.lang.NullPointerException"},  "WFLYCTL0412: Required services that are not installed:" => [  "jboss.deployment.unit.\"mdb-activemq-module-demo-plain.war\".WeldStartService",  "jboss.deployment.unit.\"mdb-activemq-module-demo-plain.war\".beanmanager"  ],  "WFLYCTL0180: Services with missing/unavailable dependencies" => [  "jboss.deployment.unit.\"mdb-activemq-module-demo-plain.war\".batch.artifact.factory is missing [jboss.deployment.unit.\"mdb-activemq-module-demo-plain.war\".beanmanager]",  "jboss.deployment.unit.\"mdb-activemq-module-demo-plain.war\".weld.weldClassIntrospector is missing [jboss.deployment.unit.\"mdb-activemq-module-demo-plain.war\".beanmanager, jboss.deployment.unit.\"mdb-activemq-module-demo-plain.war\".WeldStartService]"  ] } 07:45:40,839 INFO [org.jboss.as.server] (ServerService Thread Pool -- 45) WFLYSRV0010: Deployed "mdb-activemq-module-demo-plain.war" (runtime-name : "mdb-activemq-module-demo-plain.war") 07:45:40,854 INFO [org.jboss.as.controller] (Controller Boot Thread) WFLYCTL0183: Service status report WFLYCTL0184: New missing/unsatisfied dependencies:  service jboss.deployment.unit."mdb-activemq-module-demo-plain.war".WeldStartService (missing) dependents: [service jboss.deployment.unit."mdb-activemq-module-demo-plain.war".weld.weldClassIntrospector]  service jboss.deployment.unit."mdb-activemq-module-demo-plain.war".beanmanager (missing) dependents: [service jboss.deployment.unit."mdb-activemq-module-demo-plain.war".weld.weldClassIntrospector, service jboss.deployment.unit."mdb-activemq-module-demo-plain.war".batch.artifact.factory] WFLYCTL0186: Services which failed to start: service jboss.deployment.unit."mdb-activemq-module-demo-plain.war".INSTALL: WFLYSRV0153: Failed to process phase INSTALL of deployment "mdb-activemq-module-demo-plain.war" WFLYCTL0448: 2 additional services are down due to their dependencies being missing or failed |

### Solution

After quite some effort, I figured out the solution. I removed the *@JMSDestinationDefinitions* annotation from the *HelloWorldMDBServletClient*, and instead defined the Queue details in the *standalone.xml*, within the *admin-objects* section, under the *resource-adapter*. With this change, now, all is well, and i was able to deploy this on Wildfly 18.