Red Hat AMQ 2021.Q3

使用 AMQ JMS Pool 库

用于 AMQ 客户端 2.10

Last Updated: 2023-02-20

Red Hat AMQ 2021.Q3 使用 AMQ JMS Pool 库

用于 AMQ 客户端 2.10

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本指南描述了如何安装和配置库，运行实践示例，并将您的客户端与其他 AMQ 组件一起使用。

目录

使开源包含更多

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始： master、slave、blacklist 和 whitelist。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看 CTO Chris Wright 信息。

第 1 章 概述

AMQ JMS Pool 是一个库，提供 JMS 连接、会话和消息制作者缓存。它允许重复利用 JMS API 定义的标准生命周期之外的连接资源。

AMQ JMS 池作为标准的 JMS **ConnectionFactory** 实例运行，它打包了您所选 JMS 提供程序的 **ConnectionFactory**，并根据 JMS 池的配置管理该提供程序的 **Connection** 对象的生命周期。它可以配置为在调用者间共享到池 **createConnection()** 方法的一个或多个连接。

AMQ JMS Pool 是 AMQ 客户端的一部分，这是一套支持多种语言和平台的消息传递库。有关客户端的概述，请参阅 AMQ 客户端概述。有关此发行版本的详情，请参考 AMQ Clients 2.10 发行注记。

AMQ JMS 池基于池 JMS 消息传递库。

1.1. 主要特性

- JMS 1.1 和 2.0 兼容
- 自动重新连接
- 可配置连接和会话池大小

1.2. 支持的标准和协议

AMQ JMS Pool 支持 Java 消息服务 API 版本 2.0。

1.3. 支持的配置

如需当前与 AMQ JMS Pool 支持的配置相关的信息，请参阅红帽客户门户网站中的 Red Hat AMQ 7 支持的配置。

1.4. 文档惯例

sudo 命令
在本文档中，**sudo** 用于任何需要 root 权限的命令。使用 **sudo** 时要小心，因为任何更改都可能会影响整个系统。有关 **sudo** 的详情请参考 使用 sudo 命令。

文件路径
在这个文档中，所有文件路径都对 Linux、UNIX 和类似操作系统有效（例如 **/home/andrea**）。在 Microsoft Windows 中，您必须使用等效的 Windows 路径（例如 **C:\Users\andrea**）。

变量文本
本文档包含代码块，它们需要使用特定于环境的值替换。变量文本括在箭头大括号内，样式为圆形单空间。例如，在以下命令中，将 ***<project-dir>*** 替换为您的环境的值：

```
$ cd <project-dir>
```

第 2 章 安装

本章指导您完成在您的环境中安装 AMQ JMS 池的步骤。

2.1. 先决条件

- 您必须有 订阅 才能访问 AMQ 发行文件和存储库。
- 要使用 AMQ JMS 池构建程序，您必须安装 Apache Maven。
- 要使用 AMQ JMS 池，您必须安装 Java。

2.2. 使用 RED HAT MAVEN 存储库

配置您的 Maven 环境，以从红帽 Maven 存储库下载客户端库。

流程

1. 将红帽存储库添加到您的 Maven 设置或 POM 文件。如需示例配置文件，请参阅 第 B.1 节 “使用在线存储库”。

```
<repository>
  <id>red-hat-ga</id>
  <url>https://maven.repository.redhat.com/ga</url>
</repository>
```

2. 将库依赖关系添加到您的 POM 文件。

```
<dependency>
  <groupId>org.messaginghub</groupId>
  <artifactId>pooled-jms</artifactId>
  <version>2.0.0.redhat-00001</version>
</dependency>
```

该客户端现在在 Maven 项目中可用。

2.3. 安装本地 MAVEN 存储库

作为在线存储库的替代选择，可以将 AMQ JMS 池作为基于文件的 Maven 存储库安装到您的本地文件系统中。

流程

1. 使用您的订阅 下载 **AMQ 客户端 2.10.0 JMS Pool Maven 存储库**zip 文件。
2. 将文件内容提取到您选择的目录中。
   在 Linux 或 UNIX 中，使用 **unzip** 命令提取文件内容。

```
$ unzip amq-clients-2.10.0-jms-pool-maven-repository.zip
```

   在 Windows 上，右键单击 .zip 文件并选择" **提取所有"。**

3. 配置 Maven，以使用提取的安装目录中 **maven-repository** 目录中的存储库。如需更多信息，请参阅 第 B.2 节 “使用本地存储库”。

2.4. 安装示例

流程

1. 使用 **git clone** 命令将源存储库克隆到名为 **pooled-jms** 的本地目录中：

```
$ git clone https://github.com/messaginghub/pooled-jms.git pooled-jms
```

2. 进入 **pooled-jms** 目录，并使用 **git checkout** 命令切换到 **2.0.0** 分支：

```
$ cd pooled-jms
$ git checkout 2.0.0
```

在本文档中生成的本地目录被称为 ***<source-dir>***。

第 3 章 入门

本章将引导您完成设置环境并运行简单消息传递程序的步骤。

3.1. 先决条件

- 若要构建示例，必须将 Maven 配置为 使用红帽存储库 或 本地存储库。
- 您必须 安装示例。
- 您必须有一个在 **localhost** 中侦听连接的消息代理。它必须启用匿名访问。如需更多信息，请参阅 启动代理。
- 您必须有一个名为 **queue** 的队列。如需更多信息，请参阅 创建队列。

3.2. 运行 HELLO WORLD

Hello World 示例为字符串 "Hello World" 的每个字符调用 **createConnection()**，一次传送一个。因为 AMQ JMS 池正在使用中，因此每个调用重复利用相同的底层 JMS **Connection** 对象。

流程

1. 通过在 ***<source-dir>*/pooled-jms-examples** 目录中运行以下命令来使用 Maven 构建示例。

```
$ mvn clean package dependency:copy-dependencies -DincludeScope=runtime -DskipTests
```

添加 **dependency:copy-dependencies** 会导致依赖项复制到 **target/dependency** 目录中。

2. 使用 **java** 命令来运行示例。
在 Linux 或 UNIX 中：

```
$ java -cp "target/classes:target/dependency/*" org.messaginghub.jms.example.HelloWorld
```

在 Windows 中：

```
> java -cp "target\classes;target\dependency\*" org.messaginghub.jms.example.HelloWorld
```

在 Linux 上运行它会产生以下输出：

```
$ java -cp "target/classes/:target/dependency/*" org.messaginghub.jms.example.HelloWorld
2018-05-17 11:04:23,393 [main           ] - INFO  JmsPoolConnectionFactory       - Provided ConnectionFactory is JMS 2.0+ capable.
2018-05-17 11:04:23,715 [localhost:5672]] - INFO  SaslMechanismFinder            - Best match for SASL auth was: SASL-ANONYMOUS
2018-05-17 11:04:23,739 [localhost:5672]] - INFO  JmsConnection                  - Connection ID:104dfd29-d18d-4bf5-aab9-a53660f58633:1 connected to remote Broker: amqp://localhost:5672
Hello World
```

这个示例的源代码位于 ***<source-dir>*/pooled-jms-examples/src/main/java** 目录中。JNDI 和日志记录配置位于 ***<source-dir>*/pooled-jms-examples/src/main/resources** 目录中。

第 4 章 CONFIGURATION

AMQ JMS Pool **ConnectionFactory** 实施公开了多个配置选项，它们控制池的行为及其管理的 JMS 资源。

配置选项作为 **JmsPoolConnectionFactory** 对象中的 **set** 方法公开。例如： *maxConnections* 选项使用 **setMaxConnections(int)** 方法设置。

4.1. 连接选项

这些选项影响 JMS 池在池中创建和管理连接的方式。

池的 **ConnectionFactory** 为每个用于创建连接的用户和密码组合创建一个连接池，并为没有用户名或密码的用户创建单独的池。如果需要更加精细的连接划分到池中，您必须明确创建不同的池实例。

maxConnections
: 单个池的最大连接数。默认值为 1。

connectionIdleTimeout
: 当前没有贷款的连接前的时间（毫秒为单位）可以被从池中驱除。默认值为 30 秒。0 代表禁用超时。

connectionCheckInterval
: 定期检查已过期连接的时间（毫秒为单位）。默认值为 0，表示禁用检查。

useProviderJMSContext
: 如果启用，使用底层 JMS 供应商的 **JMSContext** 类。它默认是禁用的。
在正常操作中，池使用自己的通用 **JMSContext** 实现来包装来自池中的连接，而不使用提供程序实施。般实施可能会限制供应商实施。但是，启用后，来自 **JMSContext** API 的连接不会由池管理。

4.2. 会话选项

这些选项会影响从池式连接创建的会话的行为。

maxSessionsPerConnection
: 每个连接的最大会话数。默认值为 500。负值会删除任何限制。
如果超过限制，**createSession()** 会根据配置来阻断或抛出异常。

blockIfSessionPoolIsFull
: 如果启用，则阻止 **createSession()**，直到会话在池中可用。它会被默认启用。
如果没有可用的会话，则调用 **createSession()** 会抛出 **IllegalStateException**。

blockIfSessionPoolIsFullTimeout
: 被阻断调用 **createSession()** 前的时间以毫秒为单位抛出 **IllegalStateException**。默认值为 -1，即永远调用块。

useAnonymousProducers
: 如果启用，为所有对 **createProducer()** 的调用使用单个匿名 JMS **MessageProducer**。它会被默认启用。
在个别情况下，这种行为不可取。如果禁用，对 **createProducer()** 的每个调用都会产生新的 **MessageProducer** 实例。

第 5 章 示例

本章介绍如何通过示例程序使用 AMQ JMS 池。

有关更多示例，请参阅 池 JMS 示例。

5.1. 先决条件

- 若要构建示例，必须将 Maven 配置为 使用红帽存储库 或 本地存储库。
- 要运行这些示例，您的系统必须具有一个 正在运行的代理并配置了代理。

5.2. 建立连接

本例创建一个新连接池，将它绑定到连接工厂，并使用 池创建新连接。

示例：建立连接 - Connect.java

```
package net.example;

import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import org.apache.qpid.jms.JmsConnectionFactory;
import org.messaginghub.pooled.jms.JmsPoolConnectionFactory;

public class Connect {
    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("Usage: Connect <connection-uri>");
            System.exit(1);
        }

        String connUri = args[0];

        ConnectionFactory factory = new JmsConnectionFactory(connUri);
        JmsPoolConnectionFactory pool = new JmsPoolConnectionFactory();

        try {
            pool.setConnectionFactory(factory);

            Connection conn = pool.createConnection();

            conn.start();

            try {
                System.out.println("CONNECT: Connected to '" + connUri + "'");
            } finally {
                conn.close();
            }
        } finally {
            pool.stop();
        }
    }
}
```

5.3. 配置池

本例演示了设置连接和会话配置选项。

示例：配置池 - ConnectWithConfiguration.java

```
package net.example;

import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import org.apache.qpid.jms.JmsConnectionFactory;
import org.messaginghub.pooled.jms.JmsPoolConnectionFactory;

public class ConnectWithConfiguration {
    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("Usage: ConnectWithConfiguration <connection-uri>");
            System.exit(1);
        }

        String connUri = args[0];

        ConnectionFactory factory = new JmsConnectionFactory(connUri);
        JmsPoolConnectionFactory pool = new JmsPoolConnectionFactory();

        try {
            pool.setConnectionFactory(factory);

            // Set the max connections per user to a higher value
            pool.setMaxConnections(5);

            // Create a MessageProducer for each createProducer() call
            pool.setUseAnonymousProducers(false);

            Connection conn = pool.createConnection();

            conn.start();

            try {
                System.out.println("CONNECT: Connected to '" + connUri + "'");
            } finally {
                conn.close();
            }
        } finally {
            pool.stop();
        }
    }
}
```

5.4. 运行示例

要编译和运行示例程序，请使用以下步骤：

流程

1. 创建新项目目录。这在以下步骤中被称为 **<project-dir>**。

2. 将 Java 列表示例复制到以下位置：

```
<project-dir>/src/main/java/net/example/Connect.java
<project-dir>/src/main/java/net/example/ConnectWithConfiguration.java
```

3. 使用文本编辑器创建新 **<project-dir>/pom.xml** 文件。在其中添加以下 XML:

```
<project>
  <modelVersion>4.0.0</modelVersion>

  <groupId>net.example</groupId>
  <artifactId>example</artifactId>
  <version>1.0.0-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>org.messaginghub</groupId>
      <artifactId>pooled-jms</artifactId>
      <version>2.0.0.redhat-00001</version>
    </dependency>
    <dependency>
      <groupId>org.apache.qpid</groupId>
      <artifactId>qpid-jms-client</artifactId>
      <version>${qpid-jms-version}</version>
    </dependency>
  </dependencies>
</project>
```

   将 **${qpid-jms-version}** 替换为您首选的 Qpid JMS 版本。

4. 更改到项目目录，并使用 **mvn** 命令编译该程序。

```
mvn clean package dependency:copy-dependencies -DincludeScope=runtime -DskipTests
```

   添加 **dependency:copy-dependencies** 会导致依赖项复制到 **target/dependency** 目录中。

5. 使用 **java** 命令来运行该程序。
   在 Linux 或 UNIX 中：

```
java -cp "target/classes:target/dependency/*" net.example.Connect amqp://localhost
```

   在 Windows 中：

```
java -cp "target\classes;target\dependency\*" net.example.Connect amqp://localhost
```

这些示例命令运行 **Connect** 示例。要运行另一个示例，将 **Connect** 替换为您所需示例的类名称。

在 Linux 上运行 **Connect** 示例会导致以下输出：

```
$ java -cp "target/classes:target/dependency/*" net.example.Connect amqp://localhost
CONNECT: Connected to 'amqp://localhost'
```

附录 A. 使用您的订阅

AMQ 通过软件订阅提供。要管理您的订阅，请访问红帽客户门户中的帐户。

A.1. 访问您的帐户

流程

1. 转至 access.redhat.com。
2. 如果您还没有帐户，请创建一个帐户。
3. 登录到您的帐户。

A.2. 激活订阅

流程

1. 转至 access.redhat.com。
2. 导航到 **My Subscriptions**。
3. 导航到 **激活订阅** 并输入您的 16 位激活号。

A.3. 下载发行文件

要访问 .zip、.tar.gz 和其他发布文件，请使用客户门户查找要下载的相关文件。如果您使用 RPM 软件包或 Red Hat Maven 存储库，则不需要这一步。

流程

1. 打开浏览器并登录红帽客户门户网站 **产品下载页面**，网址为 access.redhat.com/downloads。
2. 查找 **INTEGRATION 类别中 的红帽 AMQ** 条目。
3. 选择所需的 AMQ 产品。此时会打开 **Software Downloads** 页面。
4. 单击组件的 **Download** 链接。

A.4. 为系统注册软件包

要在 Red Hat Enterprise Linux 上安装此产品的 RPM 软件包，必须注册您的系统。如果您使用下载的发行文件，则不需要这一步。

流程

1. 转至 access.redhat.com。
2. 进入 **Registration Assistant**。
3. 选择您的操作系统版本，再继续到下一页。
4. 使用您的系统终端中列出的命令完成注册。

有关注册您的系统的更多信息，请参阅以下资源之一：

- Red Hat Enterprise Linux 7 - 注册系统并管理订阅
- Red Hat Enterprise Linux 8 - 注册系统并管理订阅

附录 B. 使用红帽 MAVEN 存储库

本节论述了如何在您的软件中使用红帽提供的 Maven 存储库。

B.1. 使用在线存储库

红帽维护一个中央 Maven 存储库，用于基于 Maven 的项目。如需更多信息，请参阅 存储库欢迎页面。

可以通过两种方式将 Maven 配置为使用 Red Hat 存储库：

- 将存储库添加到您的 Maven 设置中
- 将软件仓库添加到您的 POM 文件

将存储库添加到 Maven 设置中
这种配置方法适用于您的用户拥有的所有 Maven 项目，只要您的 POM 文件不覆盖存储库配置并启用包含的配置集。

流程

1. 找到 Maven **settings.xml** 文件。它通常位于用户主目录的 **.m2** 目录中。如果文件不存在，请使用文本编辑器创建该文件。
   在 Linux 或 UNIX 中：

   ```
   /home/<username>/.m2/settings.xml
   ```

   在 Windows 中：

   ```
   C:\Users\<username>\.m2\settings.xml
   ```

2. 在 **settings.xml** 文件的 **profiles** 元素中添加包含红帽存储库的新配置集，如下例所示：

   示例：包含 Red Hat 软件仓库的 Maven settings.xml 文件

```
<settings>
  <profiles>
    <profile>
      <id>red-hat</id>
      <repositories>
        <repository>
          <id>red-hat-ga</id>
          <url>https://maven.repository.redhat.com/ga</url>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>red-hat-ga</id>
          <url>https://maven.repository.redhat.com/ga</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
```

```
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
  <activeProfiles>
    <activeProfile>red-hat</activeProfile>
  </activeProfiles>
</settings>
```

有关 Maven 配置的更多信息，请参阅 Maven 设置参考。

在您的 POM 文件中添加软件仓库
要在项目中直接配置存储库，请在 POM 文件的 **repositories** 元素中添加一个新的条目，如下例所示：

示例：包含 Red Hat 软件仓库的 Maven pom.xml 文件

```
<project>
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>example-app</artifactId>
  <version>1.0.0</version>

  <repositories>
    <repository>
      <id>red-hat-ga</id>
      <url>https://maven.repository.redhat.com/ga</url>
    </repository>
  </repositories>
</project>
```

有关 POM 文件配置的更多信息，请参阅 Maven POM 参考。

B.2. 使用本地存储库

红帽为其部分组件提供基于文件的 Maven 存储库。这些内容作为可下载存档提供，您可以提取到本地文件系统。

要将 Maven 配置为使用本地提取的存储库，请在 Maven 设置或 POM 文件中应用以下 XML：

```
<repository>
  <id>red-hat-local</id>
  <url>${repository-url}</url>
</repository>
```

${repository-url} 必须是包含提取仓库本地文件系统路径的文件 URL。

表 B.1. 本地 Maven 存储库的 URL 示例

操作系统	文件系统路径	URL
Linux 或 UNIX	**/home/alice/maven-repository**	**file:/home/alice/maven-repository**

操作系统	文件系统路径	URL
Windows	**C:\repos\red-hat**	**file:C:\repos\red-hat**

附录 C. 将 AMQ BROKER 与示例搭配使用

AMQ JMS Pool 示例需要一个正在运行的消息代理，其中包含名为 **queue** 的队列。使用以下步骤安装和启动代理并定义队列。

C.1. 安装代理

按照 AMQ Broker 入门 以 安装代理 并创建代理 实例 中的内容进行操作。启用匿名访问。

以下步骤将代理实例的位置称为 ***<broker-instance-dir>***。

C.2. 启动代理

流程

1. 使用 **artemis run** 命令启动代理。

```
$ <broker-instance-dir>/bin/artemis run
```

2. 检查控制台输出中是否有启动过程中记录的严重错误。代理日志记录 **Server is now live** （当它就绪时）。

```
$ example-broker/bin/artemis run

        __  __  ____    ____           _
       /\ |  \/  |/ __ \ |  _ \         | |
      /  \ | \  / | |  | | |_) |_ __ ___ | | _____ _ __
     / /\ \| |\/| | |  | |  _ <| '__/ _ \| |/ / _ \ '__|
    / ____ \| |  | | |__| | |_) | | | (_) |   <  __/ |
   /_/    \_\_|  |_|\____\_\____/|_|  \___/|_|\_\___|_|

 Red Hat AMQ <version>

2020-06-03 12:12:11,807 INFO  [org.apache.activemq.artemis.integration.bootstrap]
AMQ101000: Starting ActiveMQ Artemis Server
...
2020-06-03 12:12:12,336 INFO  [org.apache.activemq.artemis.core.server] AMQ221007:
Server is now live
...
```

C.3. 创建队列

在新终端中，使用 **artemis queue** 命令创建名为 **queue** 的队列。

```
$ <broker-instance-dir>/bin/artemis queue create --name queue --address queue --auto-create-address --anycast
```

系统将提示您回答一系列"是"或无问题。对所有设备回答 **N**。

队列创建后，代理就可与示例程序配合使用。

C.4. 停止代理

运行完示例后，使用 **artemis stop** 命令停止代理。

```
$ <broker-instance-dir>/bin/artemis stop
```

2021-08-31 15:46:19 +1000 修订