Red Hat AMQ 7.7

AMQ JMS Pool ライブラリーの使用

AMQ Clients 2.7 向け

Last Updated: 2023-02-03

Red Hat AMQ 7.7 AMQ JMS Pool ライブラリーの使用

AMQ Clients 2.7 向け

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.

法律上の通知

概要

このガイドでは、ライブラリーのインストールと設定、実践的な例の実行、および他の AMQ コンポーネントでのクライアントの使用方法を説明します。

目次

第1章 概要

AMQ JMS Pool は、JMS 接続、セッション、およびメッセージプロデューサーのキャッシュを提供するライブラリーです。これにより、JMS API で定義された標準のライフサイクルを超えて接続リソースを再利用できます。

AMQ JMS Pool は、選択した JMS プロバイダーの **ConnectionFactory** をラップする標準の JMS **ConnectionFactory** インスタンスとして機能し、JMS プールの設定に基づいてそのプロバイダーからの **Connection** オブジェクトのライフタイムを管理します。プール **createConnection()** メソッドへの呼び出し間で 1 つ以上の接続を共有するように設定できます。

AMQ JMS Pool は AMQ Clients (複数の言語やプラットフォームをサポートするメッセージングライブラリースイート) に含まれています。クライアントの概要は、AMQ Clients の概要 を参照してください。本リリースに関する詳細は、AMQ Clients 2.7 リリースノート を参照してください。

AMQ JMS Pool は Pooled JMS メッセージングライブラリーに基づいています。

1.1. 主な特長

- JMS 1.1 および 2.0 との互換性
- 自動再接続
- 設定可能な接続およびセッションプールサイズ

1.2. サポートされる標準およびプロトコル

AMQ JMS Pool は、Java Message Service API のバージョン 2.0 をサポートします。

1.3. サポートされる構成

AMQ JMS プールは、以下に示す OS と言語のバージョンをサポートしています。詳細は、Red Hat AMQ 7 Supported Configurations を参照してください。

- 以下の JDK を使用する Red Hat Enterprise Linux 7 および 8:
  - OpenJDK 8 および 11
  - Oracle JDK 8
  - IBM JDK 8
- 以下の JDK を使用する Red Hat Enterprise Linux 6:
  - OpenJDK 8
  - Oracle JDK 8
- IBM AIX 7.1 と IBM JDK 8
- Oracle JDK 8 を搭載した Microsoft Windows 10 Pro
- Oracle JDK 8 を搭載した Microsoft Windows Server 2012 R2 および 2016
- Oracle JDK 8 を使用する Oracle Solaris 10 および 11

1.4. 本書の表記慣例

sudo コマンド
本書では、root 権限を必要とするすべてのコマンドに対して **sudo** が使用されています。すべての変更がシステム全体に影響する可能性があるため、**sudo** を使用する場合は注意が必要です。**sudo** の詳細は、sudo コマンドの使用を参照してください。

ファイルパス
本書では、すべてのファイルパスが Linux、UNIX、および同様のオペレーティングシステムで有効です (例: **/home/andrea**)。Microsoft Windows では、同等の Windows パスを使用する必要があります (例: **C:\Users\andrea**)。

変数テキスト
本書では、変数を含むコードブロックが紹介されていますが、これは、お客様の環境に固有の値に置き換える必要があります。可変テキストは矢印の中括弧で囲まれ、斜体の等幅フォントとしてスタイル設定されます。たとえば、以下のコマンドでは **<project-dir>** は実際の環境の値に置き換えます。

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

第2章 インストール

本章では、環境に AMQ JMS Pool をインストールする手順を説明します。

2.1. 前提条件

- AMQ リリースファイルおよびリポジトリーにアクセスするには、サブスクリプション が必要です。
- AMQ JMS Pool でプログラムを構築するには、Apache Maven をインストールする必要があります。
- AMQ JMS Pool を使用するには、Java をインストールする必要があります。

2.2. RED HAT MAVEN リポジトリーの使用

Red Hat Maven リポジトリーからクライアントライブラリーをダウンロードするように Maven 環境を設定します。

手順

1. Red Hat リポジトリーを Maven 設定または POM ファイルに追加します。設定ファイルの例は、「オンラインリポジトリーの使用」 を参照してください。

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

2. ライブラリーの依存関係を POM ファイルに追加します。

```
<dependency>
  <groupId>org.messaginghub</groupId>
  <artifactId>pooled-jms</artifactId>
  <version>1.1.0.redhat-00002</version>
</dependency>
```

これで、Maven プロジェクトでクライアントを使用できるようになります。

2.3. ローカル MAVEN リポジトリーのインストール

オンラインリポジトリーの代わりに、AMQ JMS Pool をファイルベースの Maven リポジトリーとしてローカルファイルシステムにインストールできます。

手順

1. サブスクリプションを使用して、**AMQ Clients 2.7.0 JMS Pool Maven repository**の .zip ファイルをダウンロードします。
2. 選択したディレクトリーにファイルの内容を抽出します。
   Linux または UNIX では、**unzip** コマンドを使用してファイルの内容を抽出します。

```
$ unzip amq-clients-2.7.0-jms-pool-maven-repository.zip
```

Windows では、.zip ファイルを右クリックして、**Extract All** を選択します。

3. 抽出されたインストールディレクトリー内の **maven-repository** ディレクトリーにあるリポジトリーを使用するように Maven を設定します。詳細は、「ローカルリポジトリーの使用」 を参照してください。

2.4. サンプルのインストール

手順

1. サブスクリプションを使用して、**AMQ クライアント 2.7.0 JMS プール**.zip ファイルをダウンロードします。

2. 選択したディレクトリーにファイルの内容を抽出します。
Linux または UNIX では、**unzip** コマンドを使用してファイルの内容を抽出します。

```
$ unzip amq-clients-2.7.0-jms-pool.zip
```

Windows では、.zip ファイルを右クリックして、**Extract All** を選択します。

.zip ファイルの内容を抽出すると、**amq-clients-2.7.0-jms-pool** という名前のディレクトリーが作成されます。これはインストールの最上位ディレクトリーであり、本書では **<install-dir>** と呼びます。

第3章 スタートガイド

本章では、環境を設定して簡単なメッセージングプログラムを実行する手順を説明します。

3.1. 前提条件

- 例を作成するには、Red Hat リポジトリー または ローカルリポジトリー を使用するように Maven を設定する必要があります。
- サンプルをインストール する必要があります。
- **localhost** での接続をリッスンするメッセージブローカーが必要です。匿名アクセスを有効にする必要があります。詳細は、ブローカーの開始を参照してください。
- **queue** という名前のキューが必要です。詳細は、キューの作成 を参照してください。

3.2. 実行中の HELLO WORLD

Hello World の例では、文字列 Hello World の各文字に対して **createConnection()** を呼び出し、一度に 1 つずつ転送します。AMQ JMS Pool が使用されているため、各呼び出しは同じ基礎となる JMS **Connection** オブジェクトを再利用します。

手順

1. **<install-dir>/examples** ディレクトリーで次のコマンドを実行して、Maven を使用してサンプルをビルドします。

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

dependency:copy-dependencies を追加すると、依存関係が **target/dependency** ディレクトリーにコピーされます。

2. **java** コマンドを使用して例を実行します。
Linux または UNIX の場合:

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

Windows の場合:

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

Linux で実行すると、以下のような出力になります。

```
$ 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
```

サンプルのソースコードは **<install-dir>/examples/src/main/java** ディレクトリーにあります。JNDI およびロギング設定は、**<install-dir>/examples/src/main/resources** ディレクトリーにあります。

第4章 設定

AMQ JMS Pool **ConnectionFactory** 実装は、プールの動作と管理する JMS リソースを制御する複数の設定オプションを公開します。

設定オプションは、**JmsPoolConnectionFactory** オブジェクトに **設定** されたメソッドとして公開されます。たとえば、maxConnections オプションは **setMaxConnections(int)** メソッドを使用して設定されます。

4.1. 接続オプション

これらのオプションは、JMS プールがプール内の接続を作成し、管理する方法に影響します。

プールされた **ConnectionFactory** は、接続の作成に使用されるユーザーとパスワードの組み合わせごとに接続のプールと、ユーザー名やパスワードのない個別のプールを作成します。接続をより詳細にわたりプールに分割する必要がある場合は、個別のプールインスタンスを明示的に作成する必要があります。

maxConnections
: 単一プールの接続の最大数。デフォルトでは 1 回です。

connectionIdleTimeout
: 現在貸し出されていない接続をプールから削除できるようになるまでのミリ秒単位の時間。デフォルトは 30 秒です。値 0 は、タイムアウトを無効にします。

connectionCheckInterval
: 期限切れの接続を定期的にチェックする間隔 (ミリ秒単位)。デフォルトは 0 で、チェックが無効になっていることを意味します。

useProviderJMSContext
: 有効になっている場合は、基盤となる JMS プロバイダーの **JMSContext** クラスを使用します。これはデフォルトでは無効にされます。
通常の操作では、プールは、プロバイダー実装を使用する代わりに、独自の汎用 **JMSContext** 実装を使用してプールからの接続をラップします。一般的な実装には、プロバイダーの実装にはない制限がある場合があります。ただし、有効にすると、**JMSContextAPI** からの接続はプールによって管理されません。

4.2. セッションオプション

これらのオプションは、プールされた接続から作成されるセッションの動作に影響します。

maxSessionsPerConnection
: 各接続の最大セッション数。デフォルトは 500 です。負の値は制限を削除します。
制限を超えた場合、**createSession()** は、設定に応じて、例外をブロックまたは出力します。

blockIfSessionPoolIsFull
: 有効にすると、セッションがプールで使用可能になるまで **createSession()** ブロックを呼び出します。これは、デフォルトで有効になっています。
無効にすると、**createSession()** の呼び出しは、使用可能なセッションがないと **IllegalStateException** を出力します。

blockIfSessionPoolIsFullTimeout

createSession() へのブロックされた呼び出しが **IllegalStateException** を出力するまでのミリ秒単位の時間。デフォルトは -1 です。これは、呼び出しが永久にブロックされることを意味します。

useAnonymousProducers

有効になっている場合は、**createProducer()** へのすべての呼び出しに単一の匿名 JMS **MessageProducer** を使用します。これは、デフォルトで有効になっています。
まれに、この動作が望ましくない場合があります。無効にすると、**createProducer()** を呼び出すたびに、新しい **MessageProducer** インスタンスが生成されます。

第5章 例

本章では、サンプルプログラムで AMQ JMS Pool を使用する方法について説明します。

5.1. 前提条件

- 例を作成するには、Red Hat リポジトリー または ローカルリポジトリー を使用するように Maven を設定する必要があります。
- サンプルを実行するには、システムに 実行中で、設定されたブローカー が必要です。

5.2. 接続の確立

この例では、新しい接続プールを作成し、接続ファクトリーにバインドし、プールを使用して新しい接続を作成します。

例: 接続の承認 - **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. プールの設定

以下の例は、接続およびセッション設定オプションを設定する方法を示しています。

例: プールの設定 - **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. サンプルの実行

サンプルプログラムをコンパイルして実行するには、次の手順を使用します。

手順

1. 新しいプロジェクトディレクトリーを作成します。これは、以降の手順では **<project-dir>** と呼ばれます。

2. Java リストのサンプルを以下の場所にコピーします。

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

3. テキストエディターを使用して、新しい **<project-dir>/pom.xml** ファイルを作成します。次の XML を追加します。

```
<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>1.1.0.redhat-00002</version>
  </dependency>
  <dependency>
   <groupId>org.apache.qpid</groupId>
   <artifactId>qpid-jms-client</artifactId>
   <version>${qpid-jms-version}</version>
  </dependency>
 </dependencies>
</project>
```

${qpid-jms-version} を、希望の Qpid JMS バージョンに置き換えます。

4. プロジェクトディレクトリーに移動し、**mvn** コマンドを使用してプログラムをコンパイルします。

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

dependency:copy-dependencies を追加すると、依存関係が **target/dependency** ディレクトリーにコピーされます。

5. **java** コマンドを使用してプログラムを実行します。
Linux または UNIX の場合:

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

Windows の場合:

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

これらのサンプルコマンドは **Connect** の例を実行します。別の例を実行するには、**Connect** を任意のサンプルのクラス名に置き換えます。

Linux で **Connect** の例を実行すると、以下の出力が表示されます。

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

付録A サブスクリプションの使用

AMQ は、ソフトウェアサブスクリプションから提供されます。サブスクリプションを管理するには、Red Hat カスタマーポータルでアカウントにアクセスします。

A.1. アカウントへのアクセス

手順

1. access.redhat.com に移動します。
2. アカウントがない場合は、作成します。
3. アカウントにログインします。

A.2. サブスクリプションのアクティベート

手順

1. access.redhat.com に移動します。
2. **サブスクリプション** に移動します。
3. **Activate a subscription** に移動し、16 桁のアクティベーション番号を入力します。

A.3. リリースファイルのダウンロード

.zip、.tar.gz およびその他のリリースファイルにアクセスするには、カスタマーポータルを使用してダウンロードする関連ファイルを検索します。RPM パッケージまたは Red Hat Maven リポジトリーを使用している場合は、この手順は必要ありません。

手順

1. ブラウザーを開き、access.redhat.com/downloads で Red Hat カスタマーポータルの **Product Downloads** ページにログインします。
2. **JBOSS INTEGRATION AND AUTOMATION** カテゴリーの **Red Hat AMQ** エントリーを見つけます。
3. 必要な AMQ 製品を選択します。**Software Downloads** ページが開きます。
4. コンポーネントの **Download** リンクをクリックします。

A.4. パッケージ用のシステムの登録

RPM パッケージを Red Hat Enterprise Linux にインストールするには、システムが登録されている必要があります。ダウンロードしたリリースファイルを使用している場合は、この手順は必要ありません。

手順

1. access.redhat.com に移動します。
2. **Registration Assistant** に移動します。

3. ご使用の OS バージョンを選択し、次のページに進みます。
4. システムの端末に一覧表示されたコマンドを使用して、登録を完了します。

詳細は、How to Register and Subscribe a System to the Red Hat Customer Portal を参照してください。

付録B RED HAT MAVEN リポジトリーの追加

このセクションでは、Red Hat が提供する Maven リポジトリーをソフトウェアで使用する方法を説明します。

B.1. オンラインリポジトリーの使用

Red Hat は、Maven ベースのプロジェクトで使用する中央の Maven リポジトリーを維持しています。詳細は、リポジトリーのウェルカムページ を参照してください。

Red Hat リポジトリーを使用するように Maven を設定する方法は 2 つあります。

- Maven 設定にリポジトリーを追加する
- POM ファイルにリポジトリーを追加する

Maven 設定へのリポジトリーの追加

この設定方法は、POM ファイルがリポジトリー設定をオーバーライドせず、含まれているプロファイルが有効になっている限り、ユーザーが所有するすべての Maven プロジェクトに適用されます。

手順

1. Maven の **settings.xml** ファイルを見つけます。これは通常、ユーザーのホームディレクトリーの **.m2** ディレクトリー内にあります。ファイルが存在しない場合は、テキストエディターを使用して作成します。
   Linux または UNIX の場合:

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

   Windows の場合:

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

2. 次の例のように、Red Hat リポジトリーを含む新しいプロファイルを **settings.xml** ファイルの **profiles** 要素に追加します。

   例: **Red Hat** リポジトリーを含む **Maven settings.xml** ファイル

   ```
   <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>
```

Maven 設定の詳細は、Maven 設定リファレンス を参照してください。

POM ファイルへのリポジトリーの追加
プロジェクトで直接リポジトリーを設定するには、次の例のように、POM ファイルの **repositories** 要素に新しいエントリーを追加します。

例: **Red Hat** リポジトリーを含む **Maven pom.xml** ファイル

```
<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>
```

POM ファイル設定の詳細は、Maven POM リファレンス を参照してください。

B.2. ローカルリポジトリーの使用

Red Hat は、そのコンポーネントの一部にファイルベースの Maven リポジトリーを提供します。これらは、ローカルファイルシステムに抽出できるダウンロード可能なアーカイブとして提供されます。

ローカルに抽出されたリポジトリーを使用するように Maven を設定するには、Maven 設定または POM ファイルに次の XML を適用します。

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

${repository-url} は、抽出されたリポジトリーのローカルファイルシステムパスを含むファイル URL である必要があります。

表B.1 ローカル Maven リポジトリーの URL の例

オペレーティングシステム	ファイルシステムパス	URL
Linux または UNIX	**/home/alice/maven-repository**	**file:/home/alice/maven-repository**
Windows	**C:\repos\red-hat**	**file:C:\repos\red-hat**

付録C 例で AMQ ブローカーの使用

AMQ JMS Pool の例では、名前が **examples** というキューが含まれる実行中のメッセージブローカーが必要です。以下の手順に従って、ブローカーをインストールして起動し、キューを定義します。

C.1. ブローカーのインストール

Getting Started with AMQ Brokerの手順に従って、ブローカーをインストール して、ブローカーインスタンスを作成 します。匿名アクセスを有効にします。

以下の手順では、ブローカーインスタンスの場所を **<broker-instance-dir>** と呼びます。

C.2. ブローカーの起動

手順

1. **artemis run** コマンドを使用してブローカーを起動します。

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

2. 起動時にログに記録された重大なエラーがないか、コンソールの出力を確認してください。ブローカーでは、準備が整うと **Server is now live** とログが記録されます。

```
$ 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. キューの作成

新しいターミナルで、**artemis queue** コマンドを使用して **queue** という名前のキューを作成します。

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

プロンプトで質問に Yes または No で回答するように求められます。すべての質問に **N** (いいえ) と回答します。

キューが作成されると、ブローカーはサンプルプログラムで使用できるようになります。

C.4. ブローカーの停止

サンプルの実行が終了したら、**artemis stop** コマンドを使用してブローカーを停止します。

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

改訂日時：2023-01-28 12:23:43 +1000