Red Hat AMQ Clients 2.11

AMQ Spring Boot Starter の使用

AMQ Clients 2.11 向け

Last Updated: 2023-06-28

Red Hat AMQ Clients 2.11 AMQ Spring Boot Starter の使用

AMQ Clients 2.11 向け

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

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

第1章 概要

AMQ Spring Boot Starter は、AMQ メッセージングを使用する Spring ベースのアプリケーションを作成するためのアダプターです。スタンドアロンの Spring アプリケーションを構築できる Spring Boot スターターモジュールを提供します。スターターは、AMQ JMS クライアントを使用して、AMQP 1.0 プロトコルを使用して通信します。

AMQ Spring Boot Starter は、複数の言語と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のスイートである AMQ Clients の一部です。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 2.11 リリースノート を参照してください。

AMQ Spring Boot Starter は、AMQP 1.0 JMS Spring Boot プロジェクトに基づいています。

1.1. 主な特長

- メッセージングが組み込まれたスタンドアロンの Spring アプリケーションをすばやく構築
- JMS リソースの自動設定
- JMS 接続およびセッションの設定可能なプーリング

1.2. サポートされる標準およびプロトコル

- Spring Boot API のバージョン 2.2
- Java Message Service API のバージョン 2.0
- Advanced Message Queueing Protocol (AMQP) のバージョン 1.0

1.3. サポートされる構成

AMQ Spring Boot Starter でサポートされている設定に関する最新情報は、Red Hat Customer Portal で Red Hat AMQ でサポートされる設定 を参照してください。

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 インストールシステム

この章では、AMQ Spring Boot Starter をご使用の環境に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 Spring Boot Starter を使用してプログラムをビルドするには、Apache Maven をインストールする必要があります。
- AMQ Spring Boot Starter 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.amqphub.spring</groupId>
  <artifactId>amqp-10-jms-spring-boot-starter</artifactId>
  <version>2.5.6.redhat-00001</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 Spring Boot Starter 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 2.11.0 Spring Boot Starter Maven リポジトリー** 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-2.11.0-spring-boot-starter-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。

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。
- **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。
- **examples** 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、ブローカーへの接続を作成し、グリーティングを含むメッセージを **examples** キューに送信して、受信しなおします。成功すると、受信したメッセージをコンソールに出力します。

例: **Hello World!** の送受信 - **HelloWorld.java**

```
package net.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.jms.annotation.EnableJms;
import org.springframework.jms.annotation.JmsListener;
import org.springframework.jms.core.JmsTemplate;

@EnableJms
@SpringBootApplication
public class HelloWorld implements CommandLineRunner {
    @Autowired
    private JmsTemplate jmsTemplate;

    public static void main(String[] args) {
        SpringApplication.run(HelloWorld.class, args);
    }

    @Override
    public void run(String... strings) throws Exception {
        sendMessage("Hello World!");
    }

    public void sendMessage(String text) {
        System.out.println(String.format("Sending '%s'", text));
        this.jmsTemplate.convertAndSend("example", text);
    }

    @JmsListener(destination = "example")
    public void receiveMessage(String text) {
```

```
        System.out.println(String.format("Received '%s'", text));
    }
}
```

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. サンプルリストを次の場所にコピーします。

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

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.0</version>
  </parent>

  <dependencies>
    <dependency>
      <groupId>org.amqphub.spring</groupId>
      <artifactId>amqp-10-jms-spring-boot-starter</artifactId>
      <version>2.5.6.redhat-00001</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>
```

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
```

5. **java** コマンドを使用してプログラムを実行します。

```
$ java -jar target/example-1.0.0-SNAPSHOT.jar
```

Hello World の例を実行すると、次のコンソール出力が生成されます。

```
$ java -jar target/example-1.0.0-SNAPSHOT.jar


  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.5.RELEASE)

[...]
2018-11-05 16:21:09.849  INFO 14805 --- [main] net.example.HelloWorld: Started
HelloWorld in 1.074 seconds (JVM running for 1.38)
Sending 'Hello World!'
Received 'Hello World!'
```

3.3. 追加例

その他のサンプルプログラムは、AMQP 1.0 JMS Spring Boot プロジェクト および Spring プロジェクト から入手できます。

第4章 設定

次のオプションを **application-properties** ファイルで使用して、SpringBoot アプリケーションを設定できます。

4.1. 接続オプション

これらのオプションは、AMQ Spring Boot Starter がリモート AMQP ピアへの接続を確立する方法を決定します。スターターは AMQ JMS を使用してネットワーク経由で通信します。詳細は、AMQ JMS Client の使用 を参照してください。

amqphub.amqp10jms.remoteUrl

AMQ JMS クライアントが新しい接続を確立するのに使用する接続 URI。

接続 **URI** 形式

```
amqp[s]://host:port[?option=value[&option2=value...]]
```

詳細は、AMQ JMS クライアントの使用の 接続 URI を参照してください。

amqphub.amqp10jms.username

接続の認証に使用されるユーザー名。

amqphub.amqp10jms.password

接続の認証に使用されるパスワード。

amqphub.amqp10jms.clientId

接続に適用されるクライアント ID。

amqphub.amqp10jms.receiveLocalOnly

有効になっている場合、timeout 引数を指定して **receive** する呼び出しは、コンシューマーのローカルメッセージバッファーのみをチェックします。それ以外の場合は、リモートピアもチェックされ、使用可能なメッセージがないことを確認します。これはデフォルトでは無効にされます。

amqphub.amqp10jms.receiveNoWaitLocalOnly

有効になっている場合、**receiveNoWait** の呼び出しは、コンシューマーのローカルメッセージバッファーのみをチェックします。それ以外の場合は、リモートピアもチェックされ、使用可能なメッセージがないことを確認します。これはデフォルトでは無効にされます。

4.2. プーリングオプション

これらのオプションは、AMQ Spring Boot Starter が JMS 接続およびセッションをキャッシュする方法を決定します。スターターは、プーリングに AMQ JMS Pool を使用します。詳細は、AMQ JMS Pool Library の使用 を参照してください。

amqphub.amqp10jms.pool.enabled

プーリングを有効にするかどうかを制御します。これはデフォルトでは無効にされます。

amqphub.amqp10jms.pool.maxConnections

単一プールの接続の最大数。デフォルトでは 1 回です。

amqphub.amqp10jms.pool.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。

amqphub.amqp10jms.pool.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。

amqphub.amqp10jms.pool.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。

amqphub.amqp10jms.pool.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。

amqphub.amqp10jms.pool.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。

amqphub.amqp10jms.pool.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。

amqphub.amqp10jms.pool.explicitProducerCacheSize

匿名プロデューサーを使用しない場合、JMS **Session** は、明示的な宛先を持つ特定の数の **MessageProducer** オブジェクトをキャッシュするように設定できます。キャッシュされたプロデューサーと一致しない新しいプロデューサーが作成されると、キャッシュ内の最も古いエントリーが削除されます。

amqphub.amqp10jms.pool.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。

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. パッケージ用システムの登録

この製品の 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。

システムを登録する方法は、以下のリソースを参照してください。

- Red Hat Enterprise Linux 7 - システム登録およびサブスクリプション管理
- Red Hat Enterprise Linux 8 - システム登録およびサブスクリプション管理

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 Spring Boot Starter の例では、**example** という名前のキューを持つ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 **examples** という名前のキューを作成します。

```
$ <broker-instance-dir>/bin/artemis queue create --name example --address example --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: 2022-11-12 20:18:09 +1000