MDB カートリッジ　ユーザーガイド

第１．２版

２００６／０１／２４

本書は公開されている情報に基づいて(株)エクサとその協力会社が共同して新規に書き起こしたものであり、権利は(株)エクサが保有します。

本書の複製を内部ネットワーク等の媒体で 2 次配布する場合は本書が(株)エクサによって公開された文書であることを明記してください。

本書で使用する製品名はそれぞれ各社の商標、または登録商標です。本書の内容についてはできるかぎり正確であるよう努力しています。しかしながら、本書の内容に基づく結果については責任を負いかねますのでご了承ください。

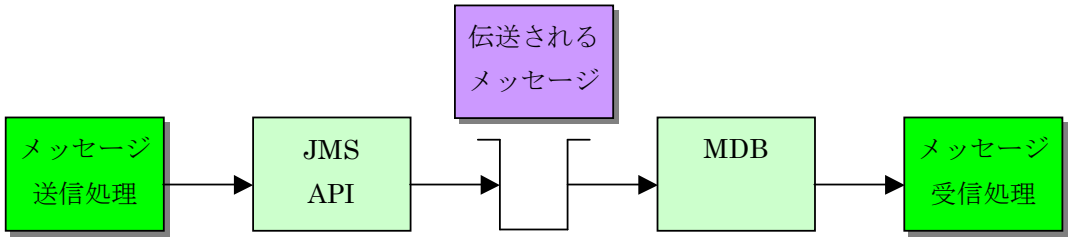
本書は無償で広く公開しておりますが、AndroMDA の利用方法や問題の解決などに関し、個別のお問い合わせには回答していません。

Java およびすべての Java 関連の商標は米国 Sun Microsystems, Inc.の登録商標または商標です。
MagicDraw UML は、米国 No Magic, Inc.の登録商標です。
JBoss は、米国 JBoss Inc.の登録商標です。
Object Management Group, OMG, Model Driven Architecture, Unified Modeling Language は Object Management Group, Inc. の登録商標です。
MDA、UML、OCL、XMI は Object Management Group, Inc. の商標です。
その他の会社名ならびに製品名は、各社の商標、もしくは登録商標です。

0　MDB カートリッジとは

MDB カートリッジは AndroMDA で用意されている既存カートリッジでは作成できない MDB（message-driven Enterprise JavaBeans）と JMS（Java Message Service）のメッセージ送信部分､つまり JMS によるメッセージ送信からそのメッセージを受信する MDB までを生成するための独自カートリッジである。ただし、JMS と MDB 両方を必ず生成する必要はなく、JMS 部分のみや MDB 部分のみの生成も可能となっている。MDB カートリッジは単独では使用できず、Spring カートリッジ、Hibernate カートリッジ、EJB カートリッジのいずれかと併用して使用する。
MDB カートリッジが生成するファイルは以下の４つである。

- メッセージオブジェクトクラス
  JMS と MDB 間で受け渡すメッセージを Java のクラスで表したもの(JavaBeans)
- メッセージ送信クラス
  JMS を利用してメッセージを送信するユーティリティ的なクラス
- メッセージ受信クラス
  MDB のインタフェースを継承した MDB の実装クラス
- EJB の設定ファイル
  メッセージを処理する MDB クラスなどの設定がされているファイル



この４種類のファイルは基本的に UML モデルより生成するのだが、EJB の設定ファイルに関しては、モデルのほかに外部ファイルを必要とする。詳細は利用方法の章で述べる。

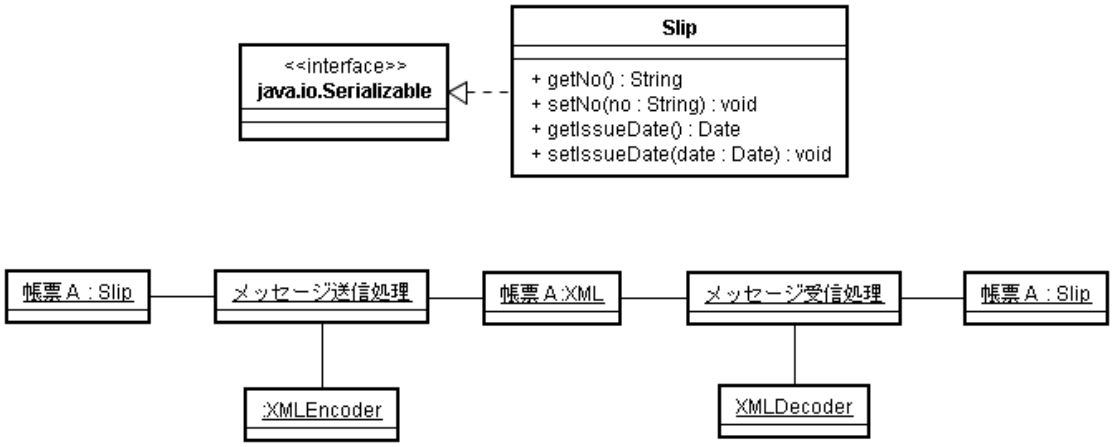
MDB カートリッジで使用するステレオタイプやタグ付き値は MDB プロファイルに記述されているため、MDB カートリッジ使用時にはこのプロファイルを組み込む必要がある。
本書では MDB カートリッジの導入方法と、利用方法について記す。
尚、本書で記述されている内容は、AndroMDA3.0 を前提として記述している。

０．１　メッセージオブジェクトクラス

メッセージをモデル上にクラスとして記述する。モデル上のクラスから、Java のクラス（JavaBeans）を生成する。JavaBeans を XML に変換したものをメッセージとして送受信する。

モデルの例（クラス図）を示す。

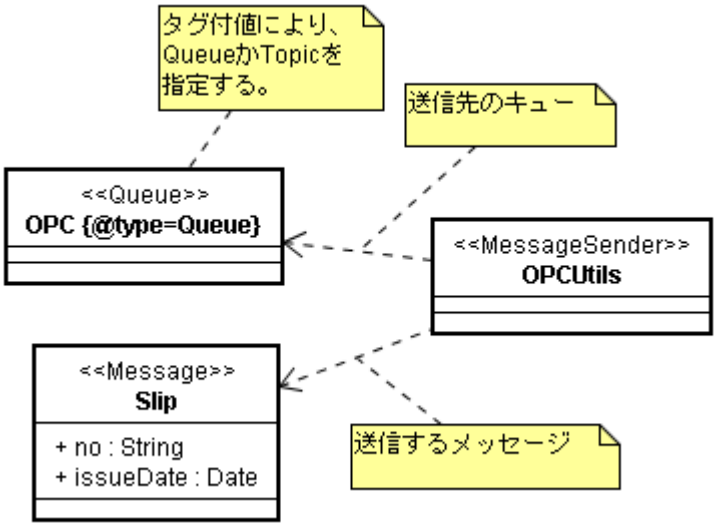
上図から生成される Java コードは下図（クラス図とインスタンス図）のようになる。



０．２　メッセージ送信クラス

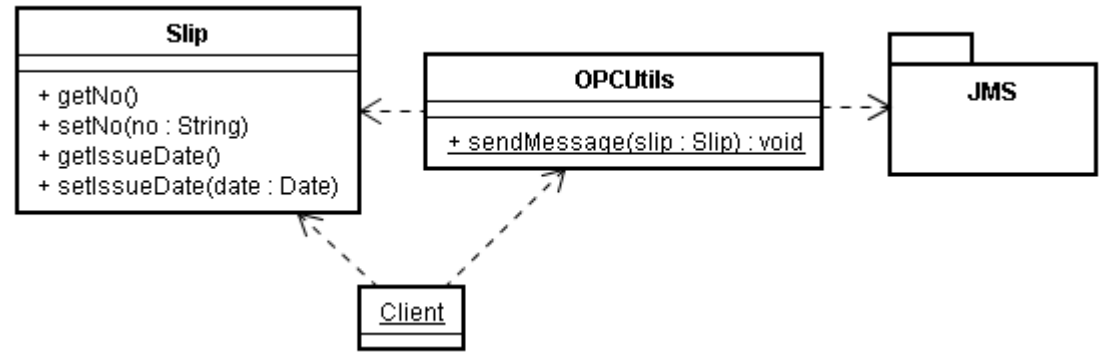
メッセージ送信クラスとは、JMS を利用してメッセージを送信するユーティリティ的なクラスである。

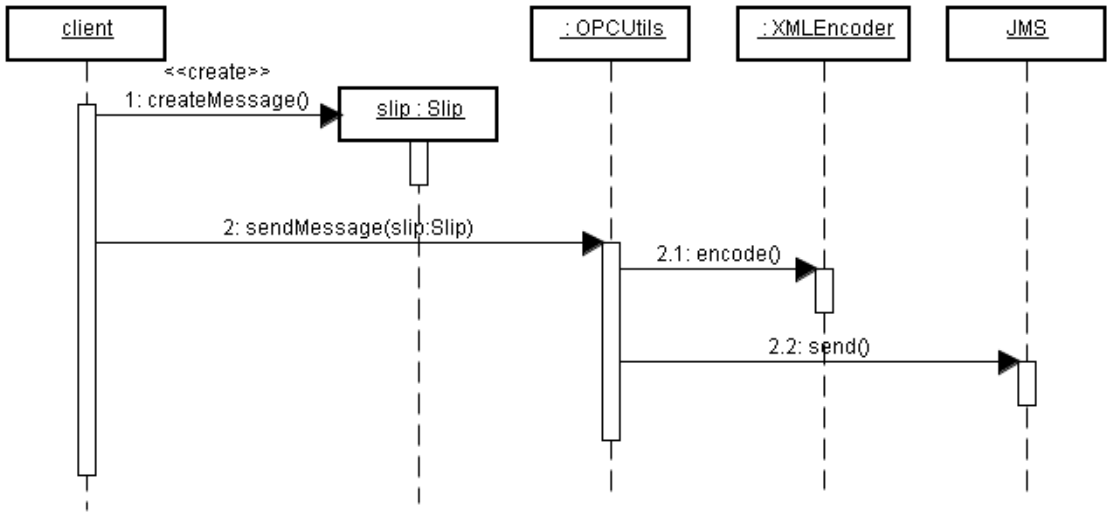
モデルの例（クラス図）を示す。



メッセージを送信するクラスは、ステレオタイプ<<MessageSender>>を付ける。また、メッセージの送信先となるキューをクラスとして表し、ステレオタイプ<<Queue>>を付ける。

上図のモデルから生成される Java コードは下図（クラス図とシーケンス図）のようになる。

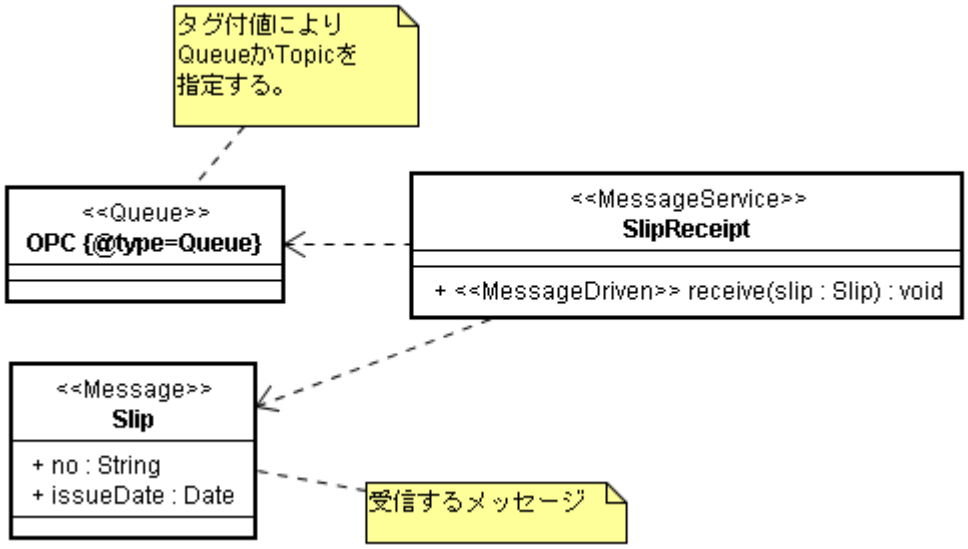




MessageSender クラス（上図では、OPCUtils クラス）は、メッセージクラス（上図では、Slip クラス）を送信するためのメソッド（上図では、sendMessage メソッド）を持つ（static メソッドである）。

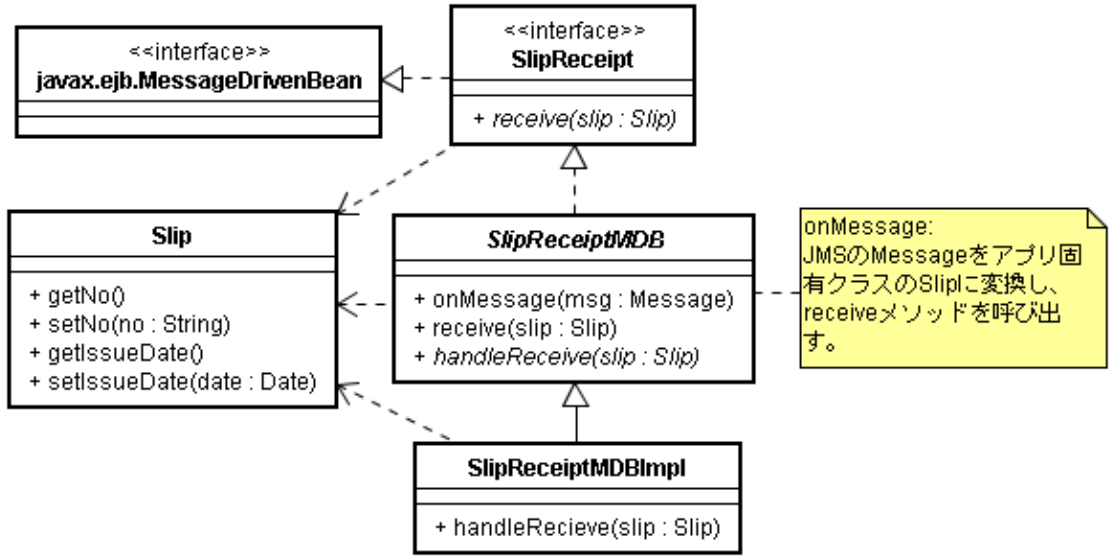
０．３　メッセージ受信クラス

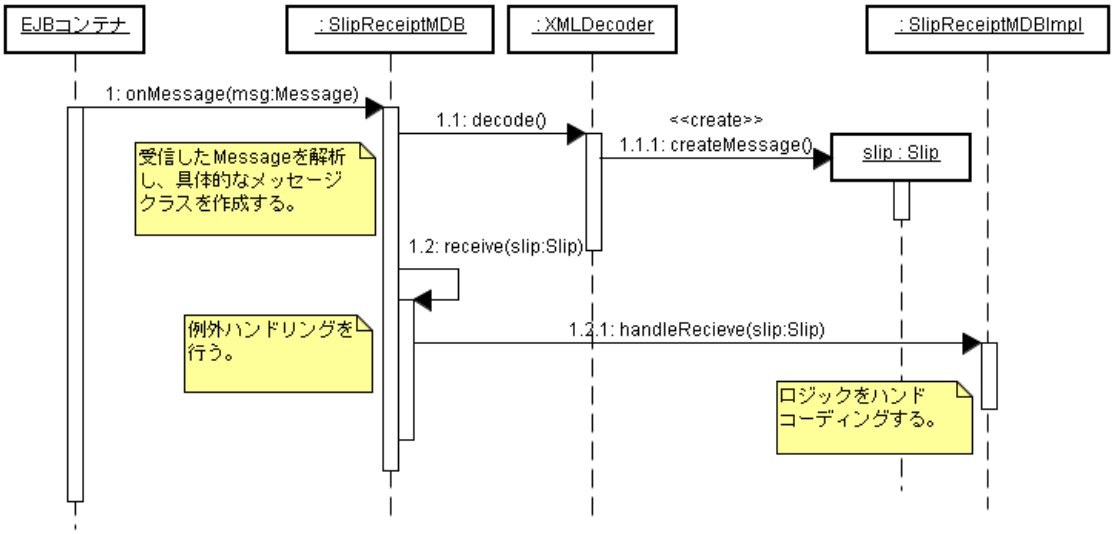
メッセージ受信クラスとは、MDB のインタフェースを継承した MDB の実装クラスである。モデルは下図のようになる。



メッセージ受信処理を行うクラスにステレオタイプ<<MessageService>>を付け、メッセージ受信処理を行うメソッドにステレオタイプ<<MessageDriven>>を付ける。また、メッセージの受信元となるキューをクラスとして表し、ステレオタイプ<<Queue>>を付ける。

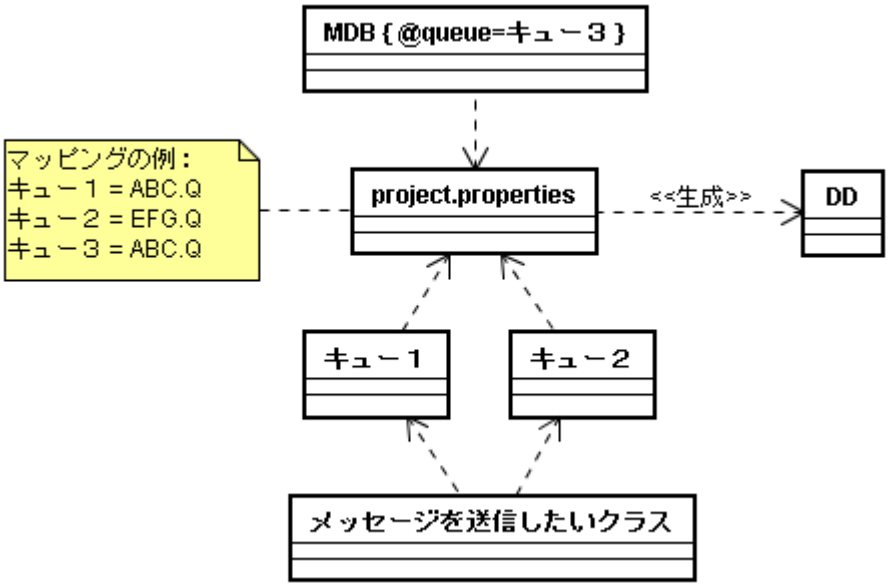
上図のモデルから生成される Java コードは下図（クラス図とシーケンス図）のようになる。





０．４　EJB の設定ファイル

EJB の設定ファイルとは、メッセージを処理する MDB クラスなどの設定がされているファイルである。



モデル中に記述するキュー名は論理的な名称を使用し、論理的な名称と実際のキュー名との関連は外部ファイル（AndroMDA の設定ファイル「project.properties」）にて指定する。EJB の DD（Deployment Descriptor）は、project.properies をもとに自動生成する。

1　MDB カートリッジ導入方法

MDB カートリッジは AndroMDA が用意しているカートリッジではないため、AndroMDA のプロジェクト生成をしても MDB カートリッジの情報は組み込まれていない。本章では MDB カートリッジを導入する方法を記す。

尚、１．１については、MDB カートリッジに変更がない限り 1 度実施すればよい。１．２については、MDB カートリッジを利用する新規プロジェクトを作成する際に設定する必要がある。

１．１　MDB カートリッジライブラリの配置

MDB カートリッジを利用するためには、ビルド時に MDB カートリッジライブラリが参照できるようにしておく必要がある。そのため、MDB カートリッジのライブラリをリポジトリに配置する。本書では用意されている jar を直接配置する方法と、MDB カートリッジのプロジェクトからビルドして配置する方法を記す。

尚、MDB カートリッジライブラリに変更が加えられた場合は再度、ライブラリの配置を行う必要がある。

１．１．１　　Jar ファイルの配置

用意されている Jar ファイル「**andromda-mdb-cartridge-3.0.jar**」を指定の場所へコピーする。コピー先は、Maven のリポジトリフォルダの「**andromda¥jars**」以下にコピーする。

特に変更していなければリポジトリフォルダは「**C:¥Documents and Settings¥ログインユーザ名¥.maven¥repository**」となる。

１．１．２　　MDB カートリッジのビルド

MDB カートリッジのプロジェクトがある場合は、そのプロジェクトをビルドすることによってリポジトリに配置される。尚、MDB カートリッジのプロジェクトフォルダ名は「**andromda-mdb**」である。

１．２ プロジェクト毎の設定

ここで記述する内容は、MDB カートリッジを利用するプロジェクト毎に最低 1 度は設定する必要がある。設定する項目というのは、MDB プロファイルの適用、MDB カートリッジ情報の設定、その他の設定がある。

１．２．１　　MDB プロファイルの適用

MDB カートリッジでは独自のステレオタイプやタグ付き値を使用している。モデルでステレオタイプやタグ付き値を使用できるように MDB のプロファイルを適用する必要がある。適用するには、プロファイルを指定の場所へ配置し、モデルで読み込む。

まず、用意されている MDB プロファイルの「**mdb-profile.xml**」を「**プロジェクトルート¥mda¥src¥uml**」へコピーする。この場所にプロファイルを置いていない場合、プロジェクトのビルド時にエラーとなる。

※本書での「プロジェクトルート」とは AndroMDA プロジェクトのルートディレクトリのことを指す(図１－１．では MDBCartridgeTest がプロジェクトルートである)。

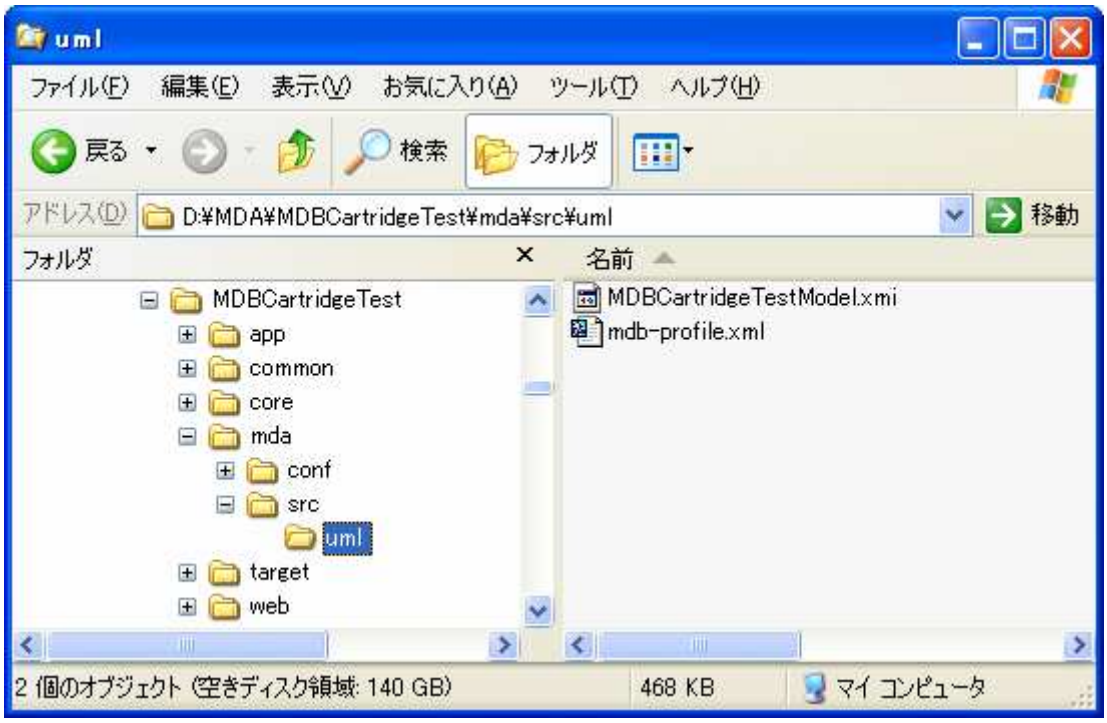


図１－１．MDB プロファイルの配置場所

次にモデル上で MDB プロファイルを使用できるように設定する。まず、MagicDraw でモデルを開く。

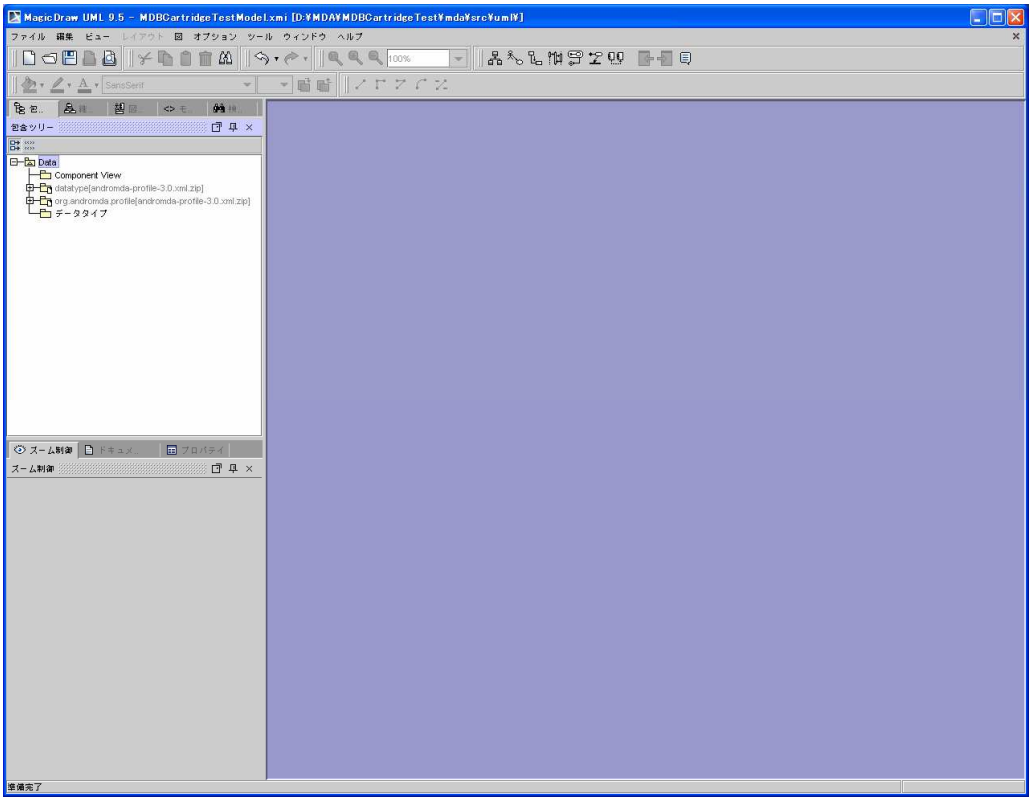


図１－２．モデルを開いた直後の様子

図１－３．のように「ファイル」メニューの「プロファイル／モジュールを使用」を選択する。

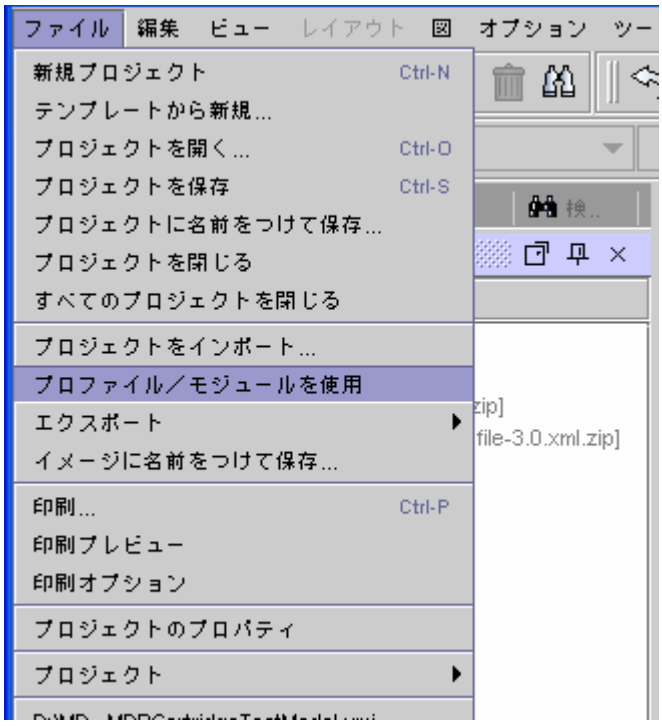


図１－３．「プロファイル／モジュールを使用」の選択

図１－４．のように表示されたダイアログの「...」ボタンを押下する。

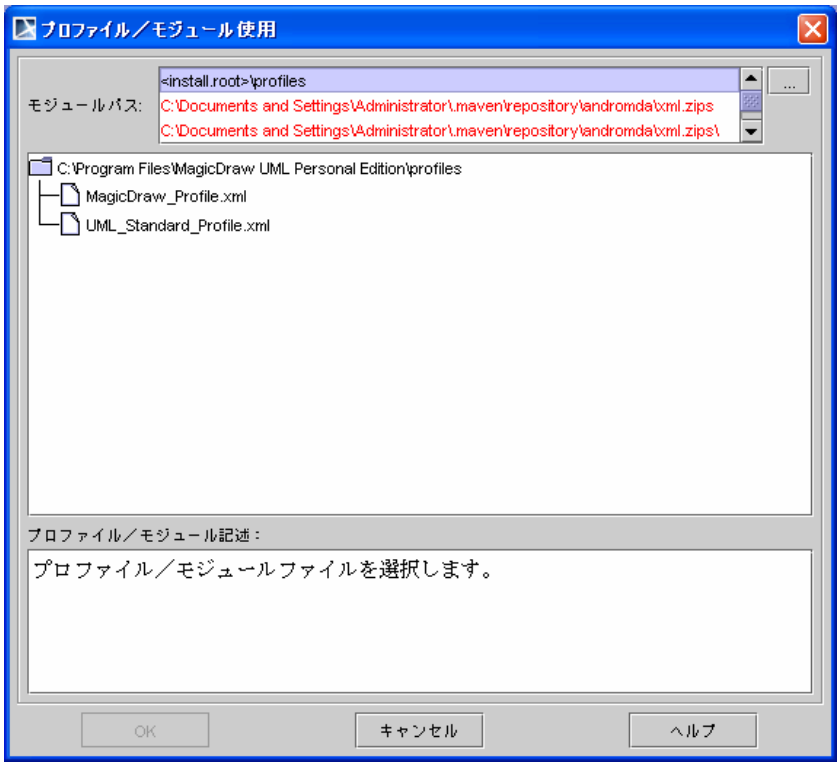


図１－４．プロファイル／モジュール使用ダイアログ

表示されたダイアログの「追加」ボタンを押下し、プロファイルが存在するフォルダを選択する。ここでは、先ほどプロファイルをコピーした「**プロジェクトルート¥mda¥src¥uml**」を選択する。すると図１－５．のような「パス変数を使用しますか？」というダイアログが表示されるので「**<project.dir>**」を選択して「選択を使用」ボタンを押下する。尚、「パス変数を使用しますか？」のダイアログは出ない場合があるが、問題なく選択したフォルダはモジュールパスに追加される。

図１－５．Path 変数使用確認ダイアログ

すると図１－６．のように選択した内容がダイアログ上に追加される。その後、「OK」ボタンを

押下する。

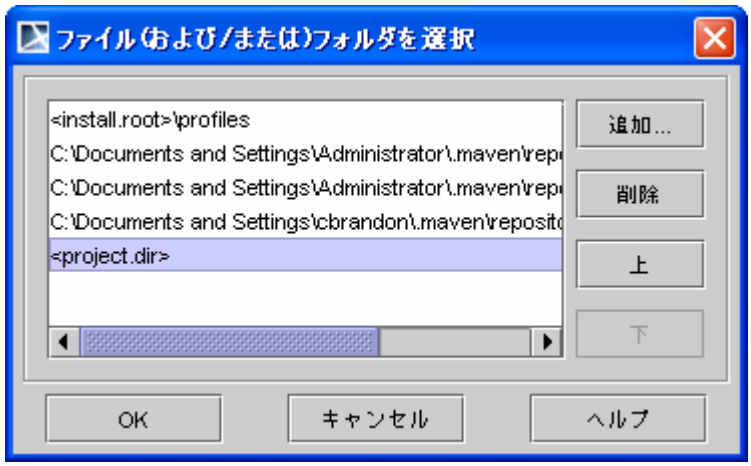


図１－６．フォルダの選択後の様子

追加したモジュールパス「**<project.dir>**」を選択すると、ファイル「**mdb-profile.xml**」が選択可能になるので選択し、「OK」ボタンを押下して決定する。

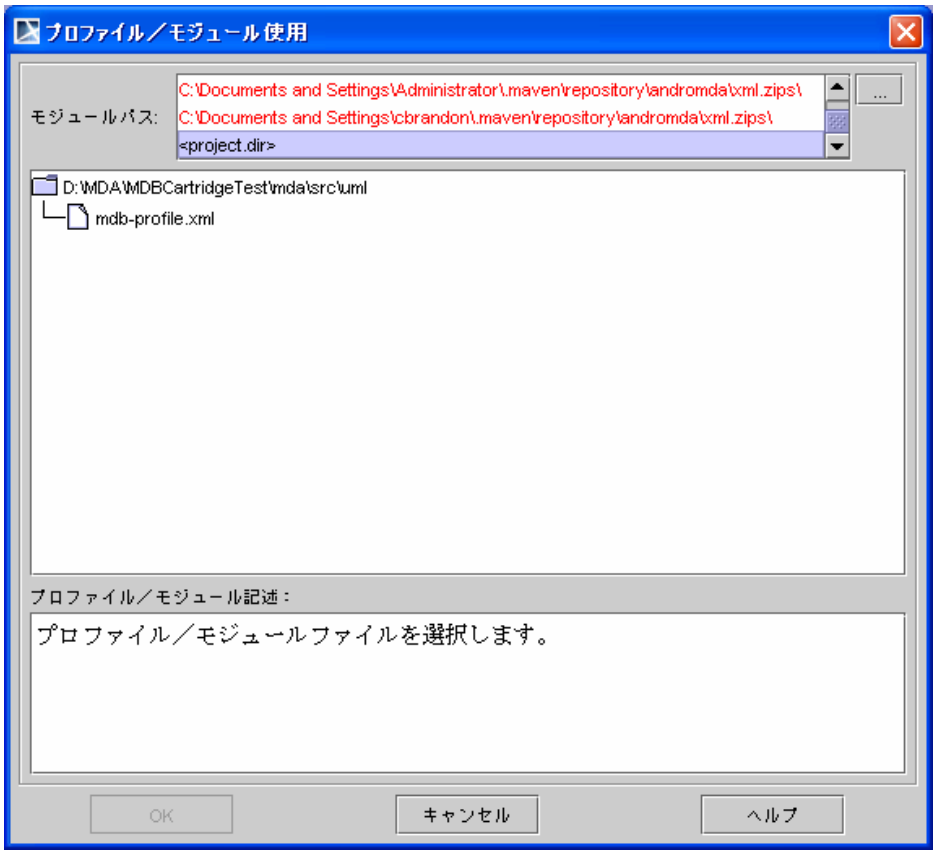


図１－７．プロファイル選択の様子

これで図１－８.のようにモデル上で MDB プロファイルのステレオタイプやタグ付き値を利用することができるようになる。

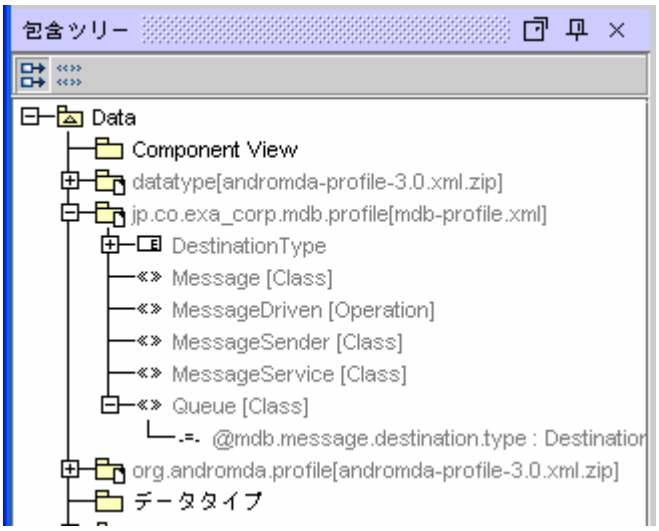


図１－８．プロジェクトモデルに MDB プロファイルを取り込んだ後の様子

１．２．２　　MDB カートリッジ情報の設定

AndroMDA プロジェクトを作成する際の質問に答えていくことによって既存のカートリッジの情報は設定されるが、MDB カートリッジは独自のため情報が設定されない。
そこでプロジェクトで MDB カートリッジを利用するということを設定しなければならない。
ここでは MDB カートリッジ情報の設定方法を記す。

プロジェクトに MDB カートリッジを認識させるには、「**プロジェクトルート¥mda**」 の下にある「**project.xml**」ファイルを編集するだけである。「**project.xml**」にはプロジェクトで使用するライブラリなどの設定が記述されている。その設定は「**<dependency>**」タグで囲って設定する。MDB カートリッジについても同様に設定する。

図１－９．に MDB カートリッジの設定内容のテンプレートを載せる。MDB カートリッジ導入時には、このテンプレートをコピー&ペーストして必要な部分を追加・変更すればよい。

```
<dependency>
  <groupId>andromda</groupId>
  <artifactId>andromda-mdb-cartridge</artifactId>
  <version>${andromda.version}</version>
  <properties>
    <!-- メッセージオブジェクトの出力先 -->
    <message-objects>${maven.andromda.common.generated.dir}</message-objects>
    <!-- 設定ファイルの出力先(ただし EJB カートリッジとの併用時は削除すること) -->
    <ejb-xml>${maven.andromda.core.generated.dir}</ejb-xml>
    <!-- メッセージ受信クラスの出力先 -->
    <service>${maven.andromda.core.generated.dir}</service>
    <!-- メッセージ受信実装クラスの出力先 -->
    <service-impl>${maven.andromda.core.manual.dir}</service-impl>
    <!-- メッセージ送信クラスの出力先 -->
    <message-util>${maven.andromda.core.generated.dir}</message-util>
    <!-- 論理キュー名と実キュー名のペアを設定しているファイルパス -->
    <propertyPath>${basedir}/project.properties</propertyPath>
    <!-- 永続化のカートリッジ種類 -->
    <persistenceType>spring</persistenceType>
    <!-- Spring Cartridge との併用時に使用される EJB 依存プログラムの生成先 -->
    <session-ejbs>${maven.andromda.core.generated.dir}</session-ejbs>
  </properties>
</dependency>
```

図１－９．MDB カートリッジ設定部分テンプレート

注意点として、MDB カートリッジの設定は必ず他のカートリッジの設定記述の後に記述すること。なぜならば、MDB カートリッジは併用しているカートリッジと競合するファイルが存在するが、MDB カートリッジは併用しているカートリッジが出力する内容と同一のものを自分の情報を付加して生成する。しかし、AndroMDA のビルドは project.xml の設定順に処理するため、併用しているカートリッジよりも先に設定しておくと併用しているカートリッジによって MDB カートリッジの生成ファイルが上書きされる。そのため、併用するカートリッジ設定の後に記述する。

設定に記述するプロパティの詳細については後述する。基本的には図１－９．の内容をそのまま使用して問題はない。ただし１つのプロパティは変更する可能性が高いためここで説明する。そのプロパティとは**<persistenceType>**である。このプロパティは永続化の部分をどのカートリッジが作成するのかを指定する。つまり、Spring、Hibernate、EJB カートリッジのいずれかになる。それぞれ指定する値は「**spring**」、「**hibernate**」、「**ejb**」である。尚、デフォルトは「**spring**」である。永続化の担当のカートリッジが MDB カートリッジと併用するカートリッジになる。

１．２．３　　その他（EJB カートリッジとの併用時の設定）

MDB カートリッジを EJB カートリッジと併用する際には Spring や Hibernate カートリッジと違い設定する必要な情報が増える。ここでは設定について記す。

まず、１．３で設定した project.xml の設定で**<persistenceType>**以外の変更場所がある。それは、**<ejb-xml>**である。このプロパティは ejb の設定ファイルの出力先なのだが、EJB カートリッジは、ejb の設定ファイル出力に Xdoclet を使用している（Xdoclet についてはここでは述べない)。そのため、このプロパティを設定する必要はないので削除またはコメントにする。しない場合は余分なファイルが出力されてしまう。よって EJB カートリッジとの併用時は図１－１０．のようにする。

```
<dependency>
  <groupId>andromda</groupId>
  <artifactId>andromda-mdb-cartridge</artifactId>
  <version>${andromda.version}</version>
  <properties>
    <!-- メッセージオブジェクトの出力先 -->
    <message-objects>${maven.andromda.common.generated.dir}</message-objects>
    <!-- 設定ファイルの出力先(ただし EJB カートリッジとの併用時は削除すること） -->
    <!--ejb-xml>${maven.andromda.core.generated.dir}</ejb-xml-->
    <!-- メッセージ受信クラスの出力先 -->
    <service>${maven.andromda.core.generated.dir}</service>
    <!-- メッセージ受信実装クラスの出力先 -->
    <service-impl>${maven.andromda.core.manual.dir}</service-impl>
    <!-- メッセージ送信クラスの出力先 -->
    <message-util>${maven.andromda.core.generated.dir}</message-util>
    <!-- 論理キュー名と実キュー名のペアを設定しているファイルパス -->
    <propertyPath>${basedir}/project.properties</propertyPath>
    <!-- 永続化のカートリッジ種類 -->
    <persistenceType>ejb</persistenceType>
    <!-- Spring Cartridge との併用時に使用される EJB 依存プログラムの生成先 -->
    <session-ejbs>${maven.andromda.core.generated.dir}</session-ejbs>
  </properties>
</dependency>
```

図１－１０．EJB カートリッジ併用時の設定内容

次に「**プロジェクトルート¥core**」にある「**maven.xml**」ファイルの編集を行う。編集内容は Xdoclet で MDB カートリッジが出力したファイルを Xdoclet 処理対象に含める設定を追加する。

編集箇所は２箇所である。

１箇所目：22 行目から 24 行目付近

```
<ant:fileset dir="${maven.build.src}">
    <ant:include name="**/*Bean.java" />
</ant:fileset>
```

ここに下記のように太字部分を追記する。

```
<ant:fileset dir="${maven.build.src}">
    <ant:include name="**/*Bean.java" />
    <ant:include name="**/*MDB.java" />
</ant:fileset>
```

２箇所目：47 行目から 50 行目付近

```
<ant:replace file="${ejb.descriptor.dir}/ejb-jar.xml"
                token="Bean&lt;/ejb-class&gt;"
                value="BeanImpl&lt;/ejb-class&gt;">
</ant:replace>
```

ここに下記のように太字部分を追記する。

```
<ant:replace file="${ejb.descriptor.dir}/ejb-jar.xml"
                token="Bean&lt;/ejb-class&gt;"
                value="BeanImpl&lt;/ejb-class&gt;">
</ant:replace>
<ant:replace file="${ejb.descriptor.dir}/ejb-jar.xml"
                token="MDB&lt;/ejb-class&gt;"
                value="MDBImpl&lt;/ejb-class&gt;">
</ant:replace>
```

以上が EJB カートリッジとの併用時に設定する必要な編集内容である。

２　MDB カートリッジの利用方法

２．１　ステレオタイプとタグ付き値

下記に MDB カートリッジで使用するステレオタイプとタグ付き値を記す。

表２－１．ステレオタイプ

名前	モデル	内容
Message	Class	メッセージオブジェクトを表す
MessageSender	Class	メッセージ送信クラスを表す
MessageService	Class	メッセージ受信クラスを表す
MessageDriven	Operation	メッセージ受信時に起動するメソッドを表す
Queue	Class	キューそのものを表す。

表２－２．タグ付き値

名前	ステレオタイプ	内容
@mdb.message.destination.type	Queue	送信タイプ(Queue/Topic)を設定

以降ステレオタイプ別の詳細を記述する。

２．２ <<Queue>>ステレオタイプ

<<Queue>>ステレオタイプは、論理的なキューを表すクラスである **(注意：<<Queue>>クラスは Java ソースコードを生成するわけではない。<<Queue>>クラスは JBOSS にキューを設定する DD ファイルを作成する等に使用する)**。<<Queue>>クラスは論理的なキューであり、<<Queue>>の完全修飾クラス名と外部ファイルにユーザが記述する実キュー名とのマッチングの情報をビルド時に利用する。外部ファイルは「**プロジェクトルート¥mda¥project.xml**」に「**<persistenceType>**」に指定する。尚、本書のテンプレートをそのまま使用した場合は、「**プロジェクトルート¥mda¥project.properties**」となる。以下の図２－１．を例にとって説明する。

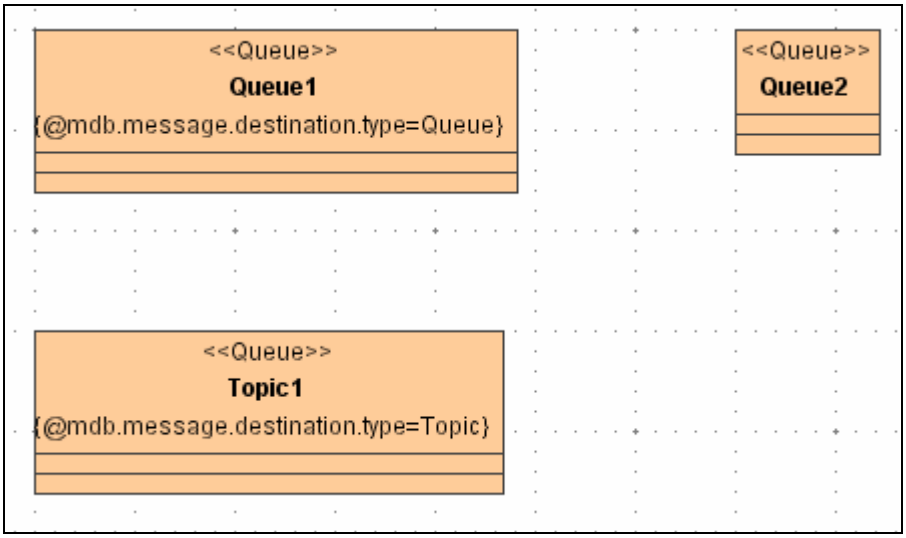


図２－１．ステレオタイプ<<Queue>>の記述例

- **Queue1** クラスは、タグ付き値@mdb.message.destination.type=Queue を設定している。Queue を設定するとこのクラスがキューを表すクラスになる。
- **Topic1** クラスは、タグ付き値@mdb.message.destination.type=Topic を設定している。Topic を設定するとこのクラスがトピックを表すクラスになる。
- **Queue2** クラスは、タグ付き値@mdb.message.destination.type を省略している。この場合はキューを表すクラスになる。
- **Queue1**、**Topic1**、 **Queue2** クラスの完全修飾クラス名が論理キュー名となる。

次に論理キュー名と実キュー名のマッピングを記述する外部ファイルの内容を説明する。ここでは、「**プロジェクトルート¥mda¥project.properties**」に図２－１．の論理キューのマッピングを記述する。

```
maven.multiproject.type=pom
～project.propertues の内容～

#論理キュー名と実キュー名のマッチング
mdb.property.jp.co.exa_corp.sample1.Queue1=system/Queue1
mdb.property.jp.co.exa_corp.sample1.Queue2=system/Queue2
mdb.property.jp.co.exa_crop.sample1.Topic1=system/Topic1
```

定義のフォーマットは「**mdb.property.完全修飾クラス名=実キュー名**」である。

「**mdb.property.**」の部分は MDB カートリッジがキューの定義情報を認識するための接頭語である。接頭語に続けてモデル図で定義したステレオタイプ<<Queue>>をつけたクラスの完全修飾クラス名を記述する。「=」をはさんで実キュー名を記述する。

２．３ <Message>>ステレオタイプ

<<Message>>ステレオタイプは、MDB カートリッジにおいてメッセージをやり取りするときに使用するメッセージそのものとなるクラスである。JavaBeans と同じ形式の Java クラスファイルを生成する。フィールドと関連は可視性を「**public**」にすること。

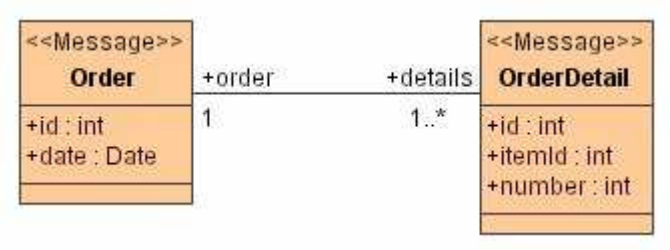


図２－２．ステレオタイプ<<Message>>の記述例

生成するないようには属性や関連の Setter や Getter も含まれる。

２．４　　<<MessageSender>>

<<MessageSender>>ステレオタイプは、<<Message>>を指定した<<Queue>>へ送るためのクラスである。フィールドやメソッド、タグ付き値などは設定しない。その代わり、<<Message>>クラスと<<Queue>>クラスへの依存が必要となる。図２－３．の MessageSend クラスには、Order クラスと Customer クラスを XML 形式でシリアライズし、Queue1（実キューqueue/system/Queue1）に送るメソッドを生成する（ここで Queue1 は図２－１．のキューと同一である）。

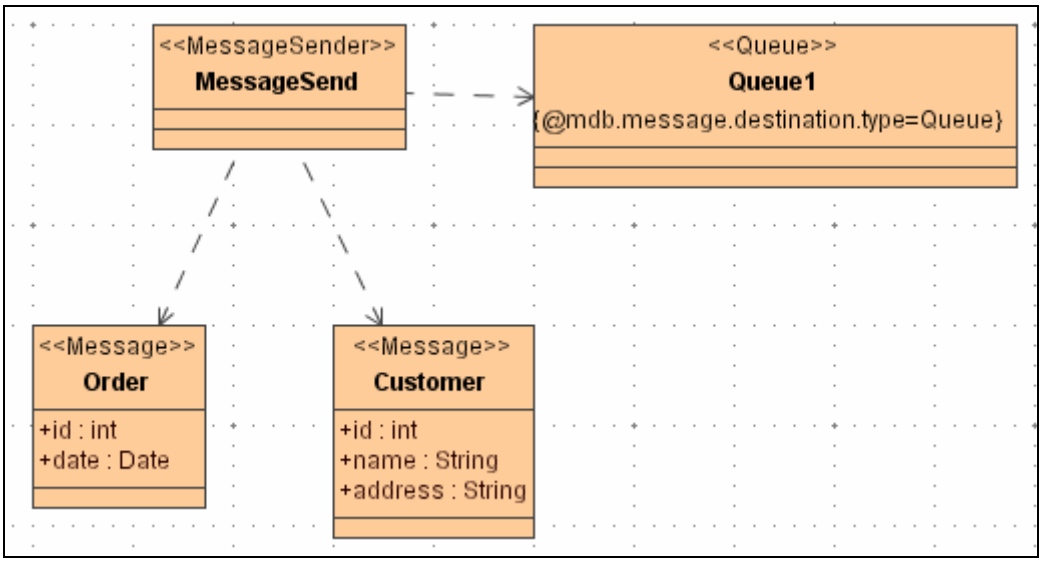


図２－３．ステレオタイプ<<MessageSender>>の記述例

MessageSend クラスから生成された Java コードは図２－４．のようなイメージとなる（実際に図２－４．のモデルを生成することではない）。

図２－４．<<MessageSender>>クラスの生成イメージ

- sendCustomer（Customer）メソッドは受け取った Customer を XML にシリアライズし、キューへ送信する
- sendOrder（Order）メソッドは受け取った Order を XML にシリアライズし、キューへ送信する

２．５　　<<MessageService>>

<<MessageService>>ステレオタイプは、指定した<<Queue>>から<<Message>>を受け取るためのクラスである。<<Queue>>クラスへの依存が必ず１つ必要になる。フィールドは設定しないが、ステレオタイプ**<<MessageDriven>>**がついたメソッドを最低１つは定義する必要がある。また、<<MessageDriven>>メソッドの引き数は必ず１つで<<Message>>型で無ければならない。さらに、そのクラスへの依存が必要となる。ただし、例外として<<MessageService>>クラスから依存する<<Message>>クラスが１つの場合にのみ、メソッドの引数を省略することができる。その場合は、ファイルに出力する際に MDB カートリッジが引数を付加して出力する。

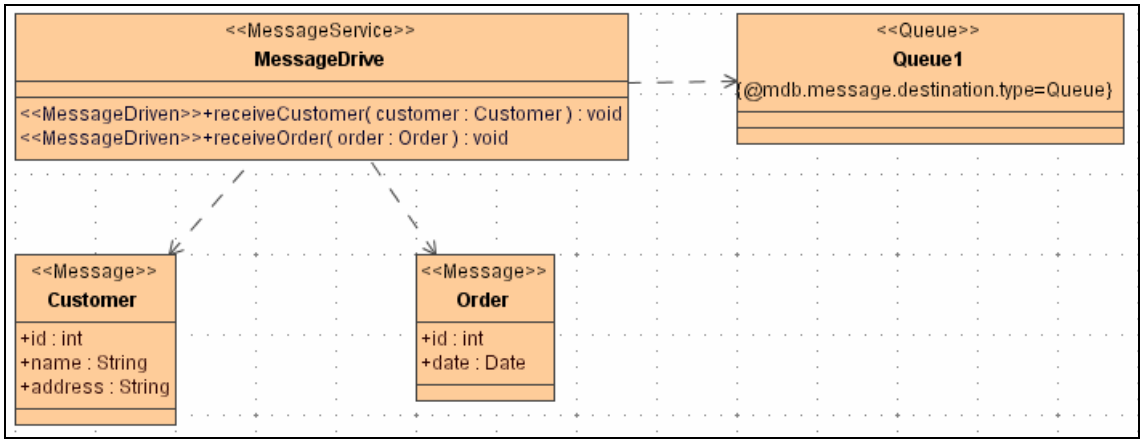


図２－５．ステレオタイプ<<MessageService>>、<<<MessageDriven>>の記述例

図２－５．の **MessageDrive クラス**からは１つのインタフェースと２つのクラスを生成する。生成した Java クラスのイメージを図２－６．に示す（実際に図２－６．のモデルを生成することはない）。

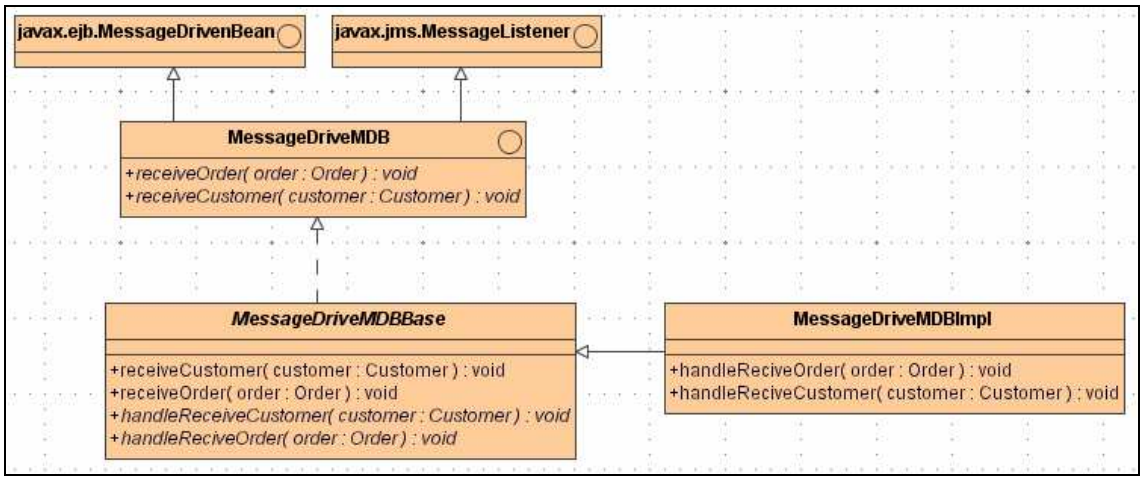


図２－６．<<MessageService>>、<<<MessageDriven>>の生成イメージ

- MessageDriveMDB（インタフェース）は、UML で定義したメソッドを定義する。また、javax.ejb.MessageDrivenBean ， javax.jms.MessageListener を継承する。
- MessageDriveMDBBase（抽象クラス）は、 MessageDriveMDB インタフェースを実装し

たクラスであり、onMessage(String)と UML で定義したメソッドと抽象メソッドを実装する。UML で定義したメソッドは、内部で抽象メソッド handleXXXX(xxxx)メソッドを呼びだす。例えば、receiveOrder(Order)は内部で抽象メソッド handleReceiveOrder(Order)を呼びだす。

また、onMessage(String)では、キューから受け取った XML メッセージをデシリアライズし、そのオブジェクトに応じて適切な<<MessageDriven>>メソッドを呼びだす。

- MessageDriveMDBImpl（クラス）は、MessageDriveMDBBase をスーパークラスとする。また、MessageDriveMDBBase の抽象メソッドをここで実装する**（注意：このクラスはビルドごとに上書きされない。1 度だけ生成する）**。プログラマは、このクラスに適切な実装コードを記述することとなる。

付録１

１．３で説明した project.xml に記述する MDB カートリッジのプロパティの内容を下記に記す。

- **message-object**
  必須：○
  デフォルト：なし
  内容：ステレオタイプ<<Message>>がついたクラスの出力先
  備考：なし
- **service**
  必須：○
  デフォルト：なし
  内容：ステレオタイプ<<MessageService>>がついたクラスのインタフェースと抽象クラスの出力先
  備考：なし
- **service-impl**
  必須：○
  デフォルト：なし
  内容：ステレオタイプ<<MessageService>>がついたクラスの実装クラスの出力先
  備考：なし
- **message-util**
  必須：○
  デフォルト：なし
  内容：ステレオタイプ<<MessageSender>>がついたクラスの出力先
  備考：なし
- **propertyPath**
  必須：○
  デフォルト：なし
  内容：論理キューと実キューのマッピングが記述されている外部ファイルパス
  備考：なし
- **ejb-xml**
  必須：×
  デフォルト：なし
  内容：ejb の設定ファイル(ejb-jar.xml や jboss.xml)の出力先
  備考：Spring、Hibernate カートリッジと併用する場合は必要、ただし、EJB カートリッジと併用する場合は不要

- **jmsFactory**
  必須：×
  デフォルト：ConnectionFactory
  内容：JMS Factory の指定
  備考：なし
- **persistenceType**
  必須：×
  デフォルト：spring
  内容：永続層で使用するカートリッジの種類
  備考：MDB カートリッジはこのプロパティで指定しているカートリッジと併用する
- **session-ejbs**
  必須：×
  デフォルト：ブランク
  内容：EJB を使用する場合のファイル出力先
  備考：spring カートリッジとの併用で使用する
- **securityRealm**
  必須：×
  デフォルト：ブランク
  内容：EJB のセキュリティー有無
  備考：spring、hibernate カートリッジとの併用で使用する
- **destinationType**
  必須：×
  デフォルト：Queue
  内容：送信タイプのデフォルト
  備考：なし
- **implementationOperationNamePattern**
  必須：×
  デフォルト：handle{0}
  内容：実装クラスのメソッド名パターン
  備考：なし
- **ejbViewType**
  必須：×
  デフォルト：local
  内容：EJB のタイプ
  備考：なし

- **serializable**

  必須：×

  デフォルト：true

  内容：メッセージクラスの永続化有無

  備考：なし

- **xmlEncoding**

  必須：×

  デフォルト：UTF-8

  内容：ejb 設定ファイルの XML エンコードタイプ

  備考：なし

- **parameterRequiredCheck**

  必須：×

  デフォルト：true

  内容：ステレオタイプ<<MessageService>>のついたクラスのメソッドパラメータの必須チェック

  備考：なし