

基于 J2EE 的电子商务开发模型及其实现

学习如何使用 Publish 和 Subscribe MDB、IBM Rational Application Developer、IBM WebSphere Enterprise Service Bus、Application Server Unit Test Engine，以及 IBM WebSphere Network Deployment 进行设计和开发按需应变的 J2EE 应用程序。这篇详细的指南包括了如何创建 MDB 并向其他 MDB 发布信息，并详细说明了如何从 IDE 直接将应用程序部署到应用程序服务器网络基础结构中。

介绍

在这样一个不断成长的 Java™ 世界中，期望与挑战日复一日的不断增加。每个人都希望站在不断变化的前沿，但是技术文章常常不能为我们提供开发者所需的细节内容。因此，本指南为您详尽展示了如何开发、部署与运行按需应变的 Java 2™ Platform Enterprise Edition™ (J2EE™) 应用程序。在这个过程中，您还会学到如何完成以下相关任务：

- 使用发布订阅 (PubSub) 消息
- 创建消息驱动 Bean (MDB) 向其他 MDB 发布信息
- 使用 IBM WebSphere® Enterprise Service Bus
- 在 IDE 中开发 Java™ Message Service (JMS) 代码
- 把来自您的 IDE 中的 IBM WebSphere Application Server 升级到 WebSphere Application Server network
- 直接从 IDE 中将应用程序部署到 WebSphere Application Server network
- 仅用 IDE (而不需要消息软件的许可证) 开发、部署与测试 JMS 应用

场景和设计概述

我们已设计了一种基于实际生活的场景，这个场景易于理解并且易于关联到技术。它使用按需应变的方式在不同应用程序服务器之间交换信息。虽然有很多方法来编写软件，但是为了交换信息的目的，我们选择了发送消息的方式来加以实现，因为它本身就拥有同步与异步的优势。

在这个实践场景中，分布在不同区域的一家公司需要完成用户实际货物订单的要求。如果某个地点不能完成一个订单，则需要其他地方代替它完成。由于安全原因，不同地点的员工不能查看其他地点的数据库。每一地点的应用程序运行于本地的应用服务器，且仅对本地职员开放。

本场景只包括三个地点，但是您可以把这种设计应用于任何多的地点中。这种设计的美妙之处在于所有地点都可以通过不同项目名称使用相同代码库。

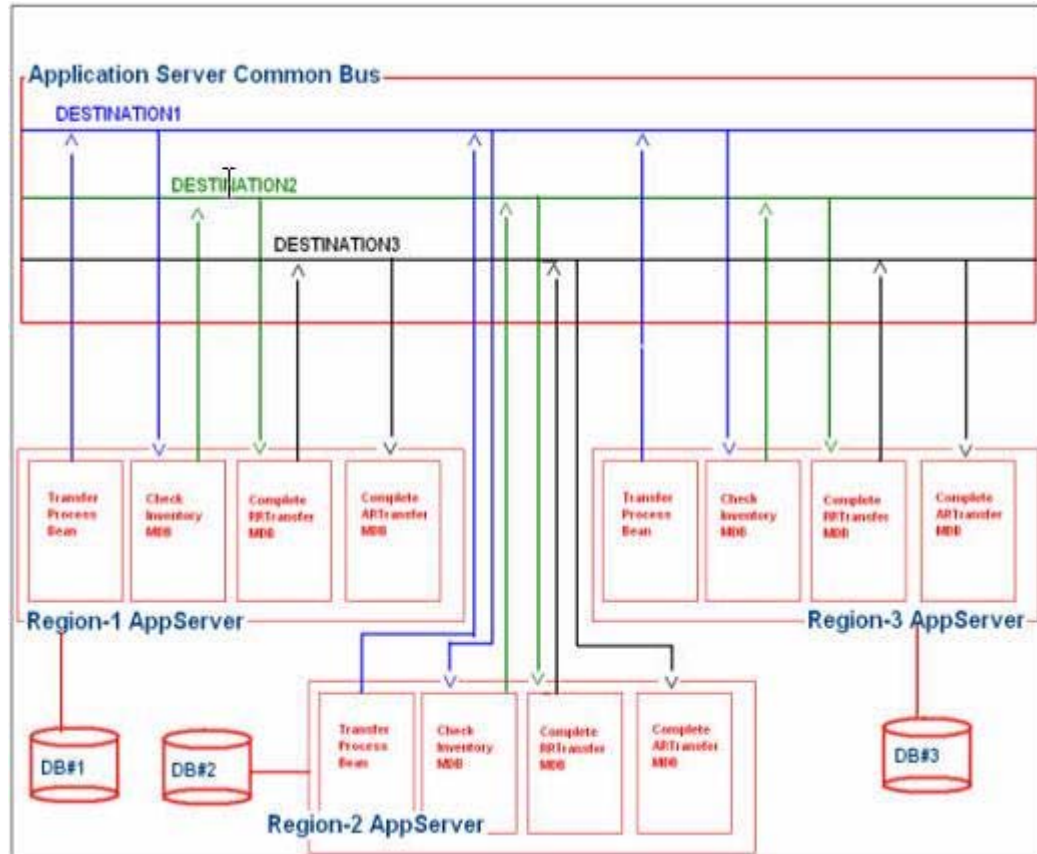
设计细节

在下面的例子中，公司的三个地点通过 WebSphere Application Server Network Deployment network 的通用总线架构共享数据。图 1 使用不同颜色标记沿着信息路由的三个目的地，箭头表示信息流的方向。例如，来自 Region 1 的 Transfer Process bean 发送数据至 DESTINATION1，它在所有地点发送对于 Check Inventory MDB 的请求信息。

图 1 描述的设计有如下假设：

- 每一地点拥有同一代码库的不同实例。
- 每一地点拥有自己的数据库。
- 每一地点拥有唯一的 Region ID，因此如果存货清单满足请求消息来自于相同的地点，则 MDB 不会处理此信息。

图 1. 消息流图



如果 Region 1 想将一个订单传送给另一地点来执行，它需要发送 Java Message Service 消息至通用总线目的地，此目的地基于 WebSphere Enterprise Service Bus 技术，并运行于应用服务网络。所有监测通用总线目的地的 MDB 会采用 PubSub 技术报警或进行触发。在响应链中每个被触发的 MDB 会触发下一个 MDB。消息使用标准语言，如 XML（您可以使用任何形式的语言作为标准，它取决于您的需要）。

我们为每个功能分别创建了一个 **topic**，一个 **destination**，以及相应的 MDB：

- 一个被称为 **TransferProcessBean** 的会话 bean，发送消息至 DESTINATION1
- 以下为所触发的 3 种 PubSub MDB：
 1. **CheckInventoryMDB** 监测或 *监听* DESTINATION1 并发送消息至 DESTINATION2
 2. **CompleteRRTransferMDB** (RR 表示 Requesting Region) 监听 DESTINATION2 并发送消息至 DESTINATION3

3. CompleteARTransferMDB (AR 表示 Accepting Region) 监听 DESTINATION3

在本例中, TransferProcessBean 消息使用如下格式 (您的消息格式与内容可以不同, 这取决于您的业务):

表 1. 消息格式举例

```
<message>
  <fromRegion> 01</ fromRegion>
  <toRegion> ALL</toRegion>
  <subject> TRANSFER-REQUEST</subject>
  ...add more tags as per your business needs
</message>
```

当然, 您可以根据您的业务需求增加更多不同的标签。

一旦 **CheckInventoryMDB** 消息被发送, 那么每个监听 DESTINATION1 的地点都会被触发响应。如果一个地点已有足够多的存货来满足用户订单, 它会自动保留必需的条目, 并发送一个其有足够存货满足此订单的响应。这些响应被发送至 DESTINATION2, 并且 <toRegion> 标签告知每一个地点的接受者, 此消息是否为它们的地点所想要的。消息看起来基本上类似如下:

表 2. 消息格式举例

```
<message>
  <fromRegion>02</ fromRegion>
  <toRegion>01</toRegion>
  <subject>TRANSFER-RESPONSE</subject>
  <details>Fill in your info/data whether you can accept or not</detail>
</message>
```

一旦前一个消息在 DESTINATION2 中被发送, 那么 **CompleteRRTransferMDB** 指令就会在每个地点中被触发。只有处于请求地点的 MDB 处理这个消息, 因为它是基于 Region 标签值的。如果有多个地点可以满足此订单, 那么请求地点的 MDB 只选择一个, 并且相应地发送一个消息至 DESTINATION3。

紧接着, 监听 DESTINATION3 的 **CompleteARTransferMDB** 被触发。它在存货存在的新地点中, 根据其订单创建逻辑完成订单的创建。现在, 订单传输完成了。

此设计的重要优势

在此设计中存在一些很关键的考虑事项:

- 单一应用程序可以通过在不同地点的最小改变进行开发与部署。
- 当您增加了一个新地点，它就成为应用服务网络中的一部分，并且可以与其他地点以零代价交换消息。
- 无论其他地点发生什么，每个地点都会独立工作。
- 每个地点不需要知道网络中存在多少个其他地点。
- 每个地点不需要知道是否其他地点正在运行。
- 如果一个地点部署了 IBM WebSphere® MQ，即使它不在线，消息也会在它再次上线时传递给它。

实践训练

现在您已经理解了基本设计，您可以开始关注如何在 IDE 和应用服务器上使用它了。为了证明的需要，本例子使用 IBM Rational Application Developer® 作为 IDE，IBM WebSphere® Application Server Version 6.1 Unit Test Environment 作为应用程序服务器。在您成功测试之后，您要使用 IBM WebSphere® Application Server Network Deployment V6.1 作为应用程序服务网络。两者之间的区别是 WebSphere Application Server **Unit Test Environment** 只能运行一个应用程序服务器，而 WebSphere Application Server **Network Deployment** 可运行支持多地点的多个应用程序服务器。您应能够在其他 IDE 和支持相似技术的应用程序服务器中实现相同的结果。

WebSphere Application Server V6.1 拥有基本消息功能。当您完成测试准备部署到生产环境时，一般的您会使用 JMS 产品作为生产环境，如 WebSphere MQ。

创建会话 bean 和 MDB

开始时创建一个会话 bean 和三个 MDB。

为创建会话 bean:

1. 打开 Rational Application Developer。
2. 创建一个 J2EE 1.4 应用程序的 2.1 Enterprise JavaBeans™ (EJB) 项目。
3. 在 EJB 项目中，创建一个被称为 TransferProcessBean 的会话 bean。您可以用来进行此项工作的方法之一是 `publishMessage1`。它被编码以发送主题消息。您可以在看到 `writeYourLogicInThisMethod1` 的地方增加您的逻辑。

表 3. 对会话 bean 的编码

```
public void publishMessage1()
{
    TopicConnectionFactory topicConnectionFactory = null;
    TopicConnection topicConnection = null;
    Topic topic = null;
    TopicPublisher topicPublisher = null;
    TopicSession topicSession = null;
    String connectionFactoryName = "java:comp/env/TCF1IdeRef";
```

```

String topicName = "java:comp/env/Topic1IdeRef";

//Add your try, catch blocks
Context ctx = new InitialContext();
topicConnectionFactory = (TopicConnectionFactory) ctx
    .lookup(connectionFactoryName);

topicConnection =
topicConnectionFactory.createTopicConnection();
boolean transacted = false;
topicSession = topicConnection.createTopicSession(transacted,
    TopicSession.AUTO_ACKNOWLEDGE);
topic = (Topic) ctx.lookup(topicName);
topicPublisher = topicSession.createPublisher(topic);
Message myOutgoingMessage1 = null;

//Add your logic in writeYourLogicInThisMethod1
String myOutgoingTmpMessage1 = writeYourLogicInThisMethod1();
myOutgoingMessage1 =
topicSession.createTextMessage(myOutgoingTmpMessage1.trim());

//Publish the Topic
topicPublisher.publish(myOutgoingMessage1);
topicPublisher.close();
topicSession.close();
topicConnection.close();
}

```

接着，创建三个 MDB：

- CheckInventoryMDB
- CompleteRRTransferMDB
- CompleteARTransferMDB

当您创建每个 MDB 时，选择：

- JMS type: **javax.jms.MessageListener**
- Transaction type: **Container**
- Activation configuration of destinationType: **javax.jms.Topic**
- Activation configuration of acknowledgeMode: **Auto-acknowledge**

当触发 MDB 时，被激活的方法是 **onMessage**。在 **writeYourLogicInThisMethod2** 中根据您的需要为每个 MDB 增加您的逻辑。以下是 **onMessage** 方法的内容：

Listing 4. The onMessage method

```
public void onMessage(javax.jms.Message msg)
{
    String myIncomingMessage1;
    //Add your try, catch blocks.
    //Similar to Session Bean,
    make sure to declare JMS related stuffs and Close them once done

    // Read the incoming Message
    myIncomingMessage1 = ((TextMessage) msg).getText();

    //Add your logic in writeYourLogicInThisMethod2
    String myOutgoingMessage2 =
writeYourLogicInThisMethod2(myIncomingMessage1);

    // Publish the Message. Code  publishMessage2 similar to Session
Bean
    publishMessage2(myOutgoingMessage2);
}
```

部署 JNDI 引用

Java Naming 和 Directory Interface™ (JNDI) 属于 Java™ 平台的一部分。它给基于 Java 技术的应用程序提供一个多重命名与目录服务的统一接口。您需要一些 JNDI 来声明 MDB 监听器，这些是现在您需要做的。

按照如下步骤编辑您的 EJB DD (配置描述符)：

1. 在 **Bean** 页签下，选择 **CheckInventoryMDB**。
2. 在 **WebSphere Bindings** 页签下，选择 **JCA Adapter**。
3. 对于 **ActivationSpec JNDI**，输入 `jms/AS1`。
4. 对于 **Destination JNDI name**，输入 `jms/Topic1`。
5. 对于 **CompleteRRTransferMDB**，输入 `jms/AS2` 和 `jms/Topic2`。
6. 对于 **CompleteARTransferMDB**，输入 `jms/AS3` 和 `jms/Topic3`。

这意味着当一个消息在 `jms/Topic1` 中发送时，`CheckInventoryMDB` 会自动运行。相同的情况发生在 `jms/Topic2` 与 `CompleteRRTransferMDB`、`jms/Topic3` 与 `CompleteARTransferMDB` 中。

声明消息目的地

现在您准备为会话 bean 和 MDB 指定消息目的地。

对于会话 bean:

1. 在 EJB DD 中, 进入 **References** 标签, 选择会话 bean。
2. 按照如下步骤增加一个新的资源引用 (Resource Reference) :
 - o **Name:** TCF1IdeRef
 - o **Type:** javax.jms.TopicConnectionFactory
 - o **Auth:** Application
 - o **Share scope:** Shareable
3. 还要按照以下步骤为主题消息增加一个新的资源参数:
 - o **Name:** Topic1IdeRef
 - o **Type:** javax.jms.Topic
 - o **Auth:** Application
 - o **Share scope:** Shareable
4. 然后, 选择 **Topic1IdeRef**。
5. 在 **WebSphere Bindings** 下, 通过键入 `.jms/Topic1` 更新 JNDI。
6. 类似的, 为 **TCF1IdeRef** 键入 `.jms/TCF01`。

对于 **MDB**, 按照以下步骤替换或修改 **TCF1IdeRef** 和 **Topic1IdeRef** :

- 对于 **CheckInventoryMDB**: **TCF2IdeRef** 和 **Topic2IdeRef**
- 对于 **CompleteRRTransferMDB**: **TCF3IdeRef** 和 **Topic3IdeRef**

注意: 记住 **CompleteARTransferMDB** 不发送任何消息; 它只是简单的监听 **DESTINATION3**。因此, 您不需要为那个 **MDB** 增加引用。

开发前端

现在您已经建立起应用核心, 您需要有一种机制引发能够触发其他 **MDB** 的会话 bean。我们有很多方式触发会话 bean。例如, 通过 Java 类调用会话 bean, 诸如 `servlet` 或 `Faces JSP`。假设您创建了包含命令按钮的 `Faces JSP`。编辑 `Faces JSP`, 然后把会话 bean 拖到命令按钮上, 这样点击按钮就可以触发会话 bean。

定义应用服务器的 **JMS** 参数

在 **WebSphere Application Server Unit Test Engine** 的消息引擎中您需要 **JMS** 引用。**Rational Application Developer** 是一份免费拷贝。下面是您必须配置的内容:

- Service integration bus
- Destinations
- Bus members
- Connection factories
- Activation specs

注意: 需要更多资料, 请访问 [WebSphere Application Server Version 6 Information Center](#)

启动您的 WebSphere Application Server Unit Test Environment, 并进入 Admin Console。
按照如下步骤键入 JMS 引用:

1. **第一步: 创建 WebSphere Enterprise Service Bus 通用总线**
 1. 在 **Service Integration** 下, 进入 **Buses**
 2. 创建以 **CommonBus1** 命名的新总线。
2. **第二步: 创建总线目的地**
 1. 在 **Buses** 下, 进入 **CommonBus1/Destinations/New**
 2. 设置 **Select destination type** 为 **Topic space**。
 3. 设置 **Identifier** 为 **DESTINATION1**。
 4. 选择 **Bus member** 作为您的服务器。
 5. **重复第二步两次**, 一次为 **DESTINATION2** 一次为 **DESTINATION3**。
3. **第三步: 创建 Bus members**
 1. 在 **Service Integration** 下, 进入 **Buses**。
 2. 通过选择您的结点和服务器为 **CommonBus1** 增加一个新总线成员。(“例如:
” **Node=pvtndserverNode01, Server=server1**)
4. **第四步: 创建 Topic Connection Factories、 Topics 和 Activation Specs:****注意:**
您将重复三遍第四步: 3 TCFs、3 Topics、3 Activation Specs。第二和第三遍的值在方括号 [] 内。例如 **TCF1WASref [TCF2WASref, TCF3WASref]**。
 1. 在 **Resources > JMS Providers > Default messaging** 内。
 2. 选择您的结点和服务器。
5. **4.1 步: 创建 Topic Connection Factory。**
 1. **JNDI name:** **jms/TCF01 [jms/TCF02 , jms/TCF03]**
 2. **Bus name:** **CommonBus1**
 3. **Client identifier value:** (example) **client1 [client2, client3]**
 4. **Durability Subscription Home:** 键入
<NodeName>. <serverName>-<busName> (例如:
pvtndserverNode01.server1-CommonBus1)
 5. **Non-persistent message reliability:** **Assured persistent**
6. **4.2 步: 为 Topic1WASref [Topic2WASref, Topic3WASref] 创建主题。**
 1. **JNDI name:** **jms/Topic1 [jms/Topic2, jms/Topic3]**
 2. **Topic Name:** **Topic1IdeRef**, 它应与 **TopicName [Topic2IdeRef, Topic3IdeRef]** 的值相同
 3. **Bus name:** **CommonBus1**
 4. **Topic space:** **DESTINATION1 [DESTINATION2, DESTINATION3]**
7. **4.3 步: 创建 activation spec。** 例如: **AS1WASref [AS1WASref, AS3WASref]**。
 1. **JNDI name:** **jms/AS1 [jms/AS3 , jms/AS3]**
 2. **Destination type:** **Topic**
 3. **Destination JNDI name** (与 JMS Topic 的 JNDI 名称相匹配): **jms/Topic1 [jms/Topic2, jms/Topic3]**
 4. **Bus name:** **CommonBus1**
 5. **Subscription durability:** **Durable**
 6. **Subscription name:** 值的例子: **sn1 [sn2 , sn3]**

7. **Client identifier:** 与您键入的 TCF/client identifier 名称一致。例如:
client1 [client2, client3]
8. **Durability Subscription Home:** 键入 node name 、 server name 、 bus name。例如: pvtndserverNode01.server1-CommonBus1
8. **第五步: 重启 WebSphere 服务器。**
9. **第六步: 为 WebSphere Application Development Unit Test Environment 部署应用程序, 并运行 Faces JSP。**
 1. **为了进行配置, 在 Servers 视图中右击 WebSphere Application Server Unit Test Engine, 然后选择 Add/Remove Projects 子菜单。**
 2. 增加您的项目。
 3. 选择 Finish。
 4. **为运行应用程序, 右击 Faces JSP 并且选择 Run on Server 子菜单。**
 5. 选择您的 WebSphere Application Server Unit Test Engine 配置。
 6. 选择 Finish。

现在您的应用程序已经可以运行, 它将根据编码逻辑运转。这样就完成了 WebSphere Application Server Unit Test Engine 级别的测试。

将 WebSphere Application Server Unit Test Engine 升级为 WebSphere Application Developer Network Deployment network

下一步需要从 WebSphere Application Server Unit Test Engine 迁移到 WebSphere Application Server Network Deployment 级别, 在那里您可以实现多地点。

注意: 为了实现上面的内容, 您需要在本地或远程服务器安装 WebSphere Application Server Network Deployment。

1. 为了将 WebSphere Application Server Unit Test Engine 升级为 WebSphere Application Server Network Deployment network, 进入 WebSphere Application Server Unit Test Engine 目录的命令行模式并运行这个命令:

```
<RADinstall-root>\runtimes\base_v6\profiles\<profileName>\bin\addNode
<WAS-ND hostname>-includeapps
```

2. 标记 includeapps 会自动地把所有结点级的应用程序与 JMS 引用移动到 ND。它没有覆盖任何元级消息。因此, 您需要在 WebSphere Application Server Network Deployment 管理员控制台上完成“定义应用服务器 JMS 引用”的 1 至 3 步内容。
3. 在发出 addNode 命令后, 您会获得成功升级到下一级的确认消息。
4. 这之后, 您需要启动另一个称为 NodeAgent 的 Java Virtual Machine (JVM) 进程, 以在 WebSphere Application Server Unit Test Environment 中进行工作。为启动 NodeAgent, 您必须运行如下命令:

```
<RADinstall-root>\runtimes\base_v6\profiles\<profileName>\bin\startNode
```

在完成这个练习之后, 您可以运行 removeNode 命令重载为先前状态的测试环境。

如何在 Rational Application Developer 中创建 WebSphere Application Server Network Deployment 服务器

现在您已经将 WebSphere Application Server Unit Test Environment 升级为 WebSphere Application Server Network Deployment 级别，您需要从 Rational Application Developer 直接部署应用程序。

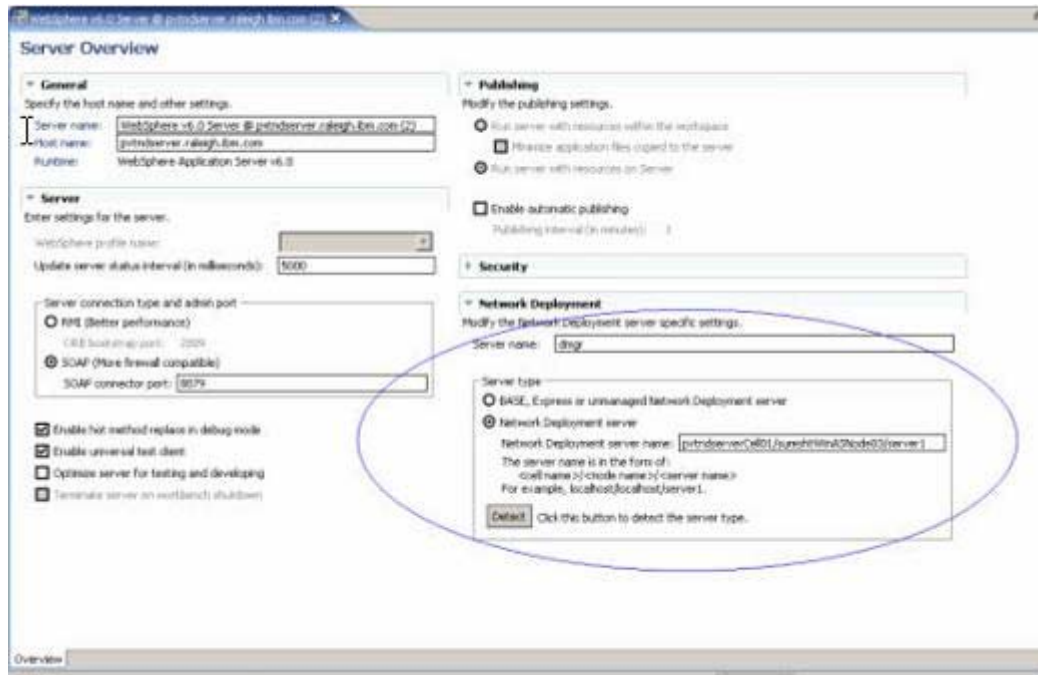
1. 进入 WebSphere Application Server Network Deployment 管理员控制台。一般来说，您可以在目录中通过 `<hostname> 9060/ibm/console` 登陆。与您的管理员确认 WebSphere Application Server 的端口号是 9060。
2. 通过 `ND-AdminConsole > System Administration > Deployment Manager > Name` 标记 the Deployment Manager 名称。有代表性的是 `dmgr`。
3. 通过 `System Administration > Deployment Manager > Ports` 标记 `SOAP_CONNECTOR_ADDRESS` 端口号。

回到 Rational Application Developer 会话。

- 在 **Server** 视图中，以指向 WebSphere Application Server Network Deployment 机器的主机名创建新的 WebSphere Application Server V6.1 服务器。
- 在 **SOAP 连结端口号**，确认输入先前提到的值。
- 对于 **Server name**，按照先前的方法键入 Deployment Manager 名称。
- 对于 **Server type**，选择 **Network Dep.**
- 对于 **Network Dep. server name**，输入您的 **cell name**、**node name**、**server name**。
例如：`pvtndserverCell101/sureshtWinASNode03/server1`。

在创建应用程序服务器之后，您可以双击 **Servers** 视图中的 entry，会看到配置情况。它与 [图 2](#) 中的屏幕截图相似。

图 2. 应用程序服务器配置实例



在 WebSphere Application Server Network Deployment 中进行发布与运行

现在您能够从 Rational Application Developer 到 WebSphere Application Server Network Deployment 发布应用程序了，这与早先 WebSphere Application Server Unit Test Environment 中的做法相似。您还应该能够运行 Faces JSP。最后，确保您看到 MDB 被触发了，并且接着他们有序地触发后面的 MDB。

这将会帮助返回到 [图 1](#)，并回顾一下数据流：

1. TransferProcessBean 会话 bean 发送消息至 DESTINATION1
2. CheckInventoryMDB 监听 DESTINATION1 并发送到 DESTINATION2
3. CompleteRRTransferMDB 监听 DESTINATION2 并发送到 DESTINATION3
4. CompleteARTransferMDB 监听 DESTINATION3

您的应用程序已经就绪。

如果您不再需要安装，您可以发出 `removeNode` 命令从 WebSphere Network Deployment 网络中释放 WebSphere Application Server Unit Test Environment。有必要的，可以在 WebSphere Application Server Version 6 Information Center 中查找更多 `removeNode` 命令的消息。

这样就完成了您的练习。

总结

现在您已经使用 WebSphere Enterprise Service Bus、PubSub MDB 以及 Unit Test Environment 与 Network Deployment 版本的 IBM WebSphere Application Server 开发和测试了一个按需应变的 J2EE 应用程序。您的下一步可以转到生产环节，在那里您可能有一个 JMS 产品，例如 WebSphere MQ（发送消息和队列等待），这使得异步通信成为现实。Rational Application Developer 把项目工件从一个版本迁移到下一个版本，这样，您从应用程序设计、开发和部署中所花费的时间和工作量的节省中使您和您的团队在未来受益。