

---

## SCALING THE ENTERPRISE WITH JMS

### Integrating J2EE Applications with WebSphere MQ

#### WebSphere MQ

##### ▶▶ Synchronous

What if A & B are on  
two different

- networks,
- protocols,
- time zones

Synchronous messaging (time-dependent) is a communications technique  
Synchronous Example: telephone call

##### ▶▶ Asynchronous

A & B transparent to

- networks,
- protocols,
- time zones

Asynchronous messaging (time-independent) communications technique  
Asynchronous Example: e-mail

#### MQI – Message Queue Interface

- MQSeries Application programming interface

#### MQSeries Objects

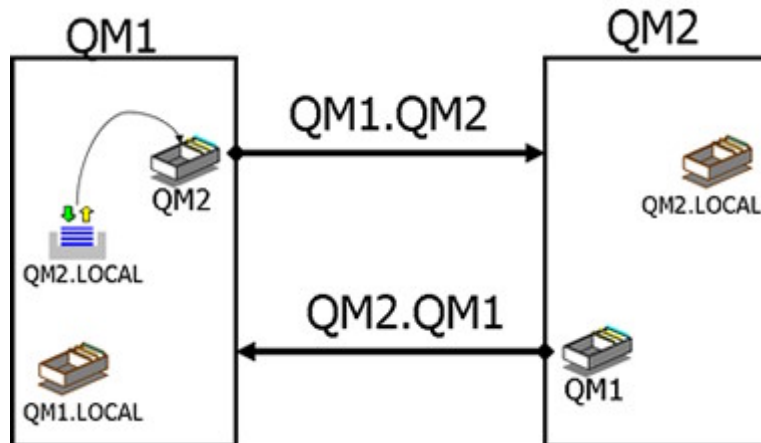
- An MQSeries Object is a recoverable resource managed by MQSeries
- Queue Manager
  - Entity that manages, administers all queuing and messaging functionality
  - Multiple Queue Managers per machine on all platforms except VSE/ESA
  - Queue Managers can belong to any number of Clusters
- MQI (Message Queue Interface)
  - MQCONN
  - MQDISC
  - MQOPEN
  - MQCLOSE
  - MQGET
  - MQPUT
  - MQPUT1
  - MQCONNX

## SCALING THE ENTERPRISE WITH JMS

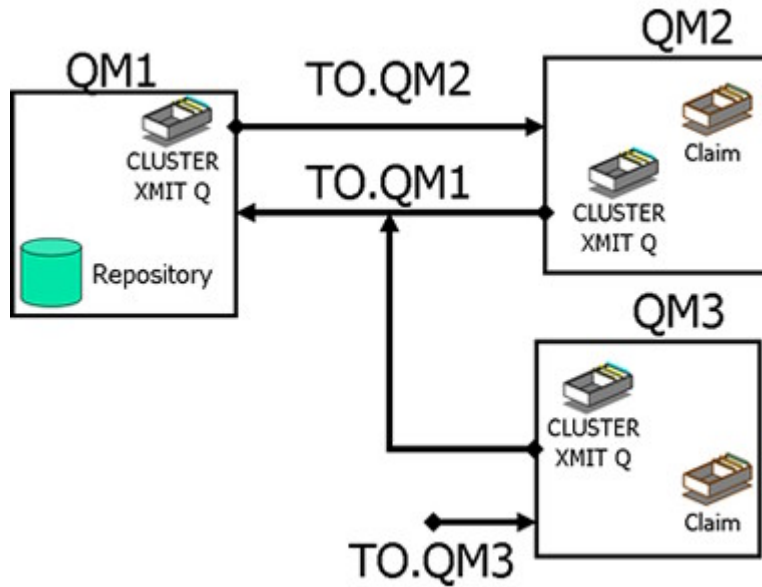
- MQBEGIN
- MQCMIT
- MQBACK
- MQINQ
- MQSET
- Procedural calls are replaced with JAVA equivalents
  - MQQueueManager
  - MQQueue
  - MQMessage
  - MQPutMessageOptions
  - MQGetMessageOptions

```
MQQueueManager qM = new MQQueueManager ("QM1");  
MQQueue q1 = qM.accessQueue("Q1", MQC.MQOO_OUTPUT,  
"QM1",NULL, NULL);  
qM.disconnect();
```

### WebSphere MQ Distributed Queuing



**WebSphere MQ Clusters**



**WebSphere MQ and Java**

- ▶▶ Initial Java bindings provided to satisfy the Java programming model
- ▶▶ Implemented as just another language binding
- ▶▶ No comprehensive support for the J2EE specifications
- ▶▶ J2EE has evolved and the container specifications have solidified
- ▶▶ IBM more committed to Java and WebSphere platform

**WAS and Messaging**

**Java Messaging Service**

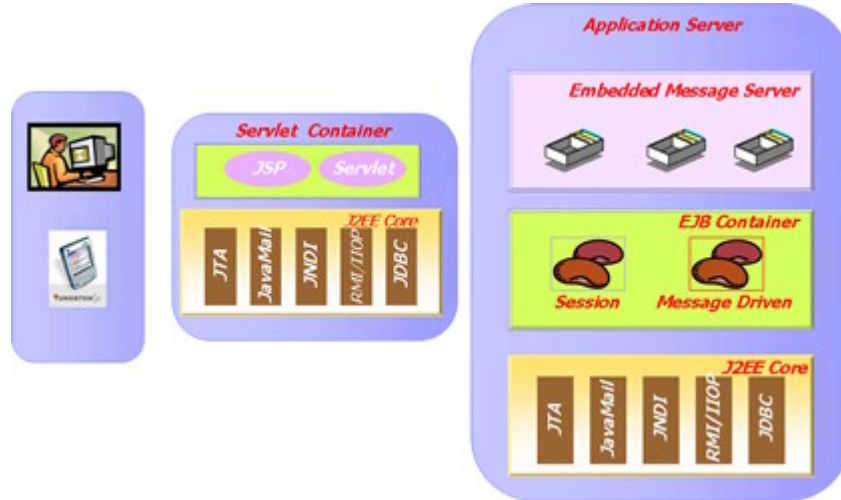
- ▶▶ Part of J2EE Specifications
- ▶▶ Introduces messaging to Java applications
- ▶▶ Unifies messaging paradigms
  - Point-to-Point
  - Publish-Subscribe
- ▶▶ Provides a consistent framework for messaging for Java applications, regardless of vendor

**JMS in WAS**

- ▶▶ WAS Evolution and JMS
  - First JMS support from WMQ using Support Pac MA88 and MA0C for Publish-Subscribe
  - WAS 3.5 and WAS 4.0 used this facility

- ▶ WAS and WMQ aligning closer
  - WAS V5 Supports J2EE 1.3 and EJB 2.0
  - JMS Provider included in WAS V5
  - Further enhancements expected

### Embedded Messaging Server



- ▶ WAS 5 includes Embedded Messaging Server
- ▶ WebSphere MQ Code base
- ▶ Limited Functionality (WAS Specific) Queue Manager
- ▶ Simplified Administration using WAS Administrative Console
- ▶ No need to use SupportPacs such as MA88 and MA0C
- ▶ Can only be for WAS Applications

### Advantages of WAS Embedded Messaging Server

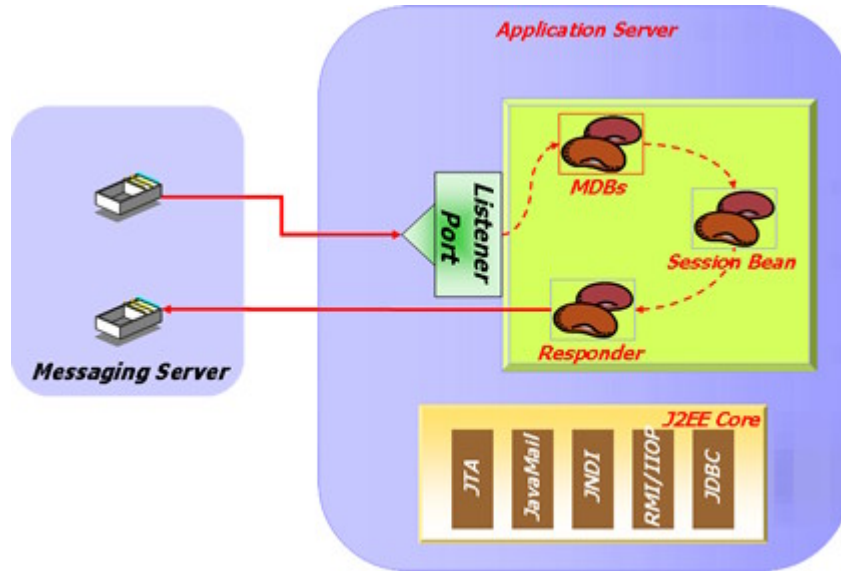
- ▶ Useful for small and non-production applications
- ▶ WAS only messaging
- ▶ No support for JMS to MQ messaging (no server channels)
- ▶ Embedded Messaging Client provides support for within the cell use
- ▶ Do not use MQ Explorer and runmqsc to administer
- ▶ Cannot co-locate WMQ and Embedded Messaging Server

### Message Driven Beans

- ▶ Part of J2EE 1.3 Specifications
- ▶ Provides a clean container managed facility for implementing message consumers
- ▶ Application programmer abstracted from infrastructure details

## SCALING THE ENTERPRISE WITH JMS

- ▶ Programmer implements the onMessage() method – Container invokes the onMessage() method
- ▶ All business logic encapsulated in the onMessage() method



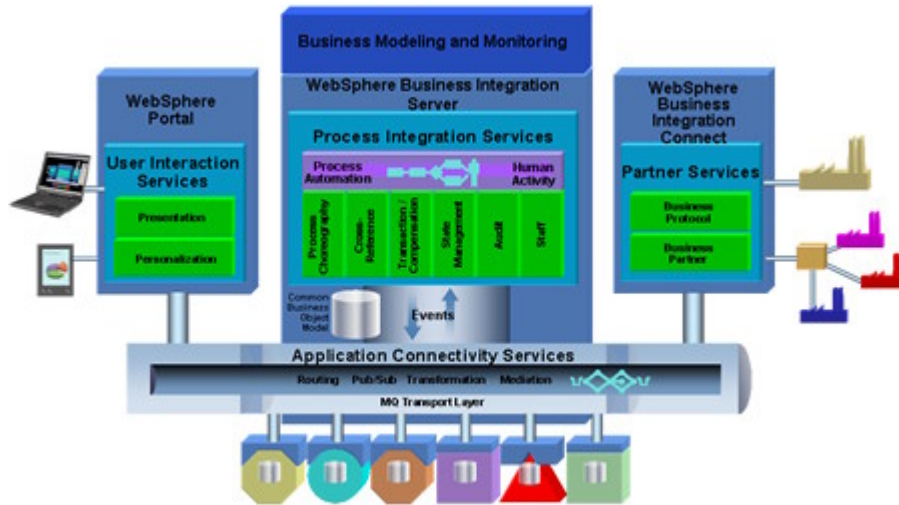
- ▶ Transactional Support
  - Bean Managed Vs. Container Managed
- ▶ Durable Vs. Non-Durable
- ▶ Separate Message processing and business logic
- ▶ Issues around fail-over & scalability
- ▶ Message Retry
- ▶ Handling Poison Messages

### WAS and WMQ

- ▶ Better Integration in WMQ 5.3 and WAS 5.x
- ▶ WMQ Clustering is not panacea
- ▶ Architect WMQ and WAS topology together
  - Failover
    - Both WMQ Failure and WAS Failure
  - Load-Balancing
- ▶ JMS and non-JMS application design considerations
- ▶ MDBs address only Consumers, no framework for Producers
- ▶ Message Conversion
  - JMS to MQ
  - MQ to JMS

The Future

WebSphere Business Integration Reference Architecture



This is the reference architecture for the WBI family. Let’s start at the bottom of the chart and work our way up the stack. Integration solutions typically begin with the fundamental need to inter-connect multiple applications. These may include prepackaged applications, like SAP, PeopleSoft, or Siebel, and often times include applications were built years ago with no requirement or design considerations for inter-connecting and interacting with other applications or components in a distributed system. There are 3 basic layers within the architecture:

**Application Connectivity**

Application connectivity is all about providing integration middleware that allows information to flow between the applications in a way that abstracts the details of the information flow from the applications themselves. This layer provides Connectivity Management – the ability to attach applications to a transport isolating the details of the connection from the internals of the application – Connectivity Management is provided through adapters Information Delivery Management – the ability to determine the appropriate destination for the information flow and ensure that it is in the form required by the destination – centralizing the logic so that it is not repeated in each of the inter-connected applications

**Process Integration**

Process Integration is all about providing integration middleware that controls the flow of (i.e. manages) execution of function across various heterogeneous, connected applications in a way that abstracts the details of the flow of activity from the applications themselves. This layer provides functions that Catch and handle application events that need to be propagated to other applications Control the flow of actions required among the interconnected applications Allow the interaction of people and applications. Provide control and state management required for long-running processes

**Modeling and Monitoring**

Modeling and monitoring is all about being able to graphically create runtime assets that implement a business process and about viewing, collecting, analyzing, and using data from the runtime system to upgrade the processes, effectively providing the necessary tools for continuous process improvement. This layer provides the

functions required for ...

- Efficient implementation of business processes
- Analyzing and assessing process efficiency and effectiveness
- Continuous business improvements through process management and change

When appropriate, these layers can be extended to end users through the functions and capabilities of our WebSphere Portal technologies. The end users may be employees or may access the system from outside the enterprise, for example, may be customers, partners, or suppliers. And when it is necessary to integrate 3rd parties systems, the layers can be extended again, using the capabilities of the WebSphere Business Connect technologies.

Let's now begin to drill down in each of these layers and describe the key components used to deliver these functions ...

### **WebSphere Business Integration Benefits**

- Patented architecture and open standards platform
  - The only architecture based on a Common Business Object Model to separate cross-application business logic from data representation
    - Reduces number of integration points
    - Enables greater re-use than any of our competition
- Pre-built, industry-focused content
  - Business Process Modules (Collaborations)
  - Adapters for specific technology platforms and packaged applications
  - Re-usable objects (ASBO's, GBO, Maps, Collaborations)
  - WebSphere Business Integration for industry solutions
- Integrated development environment
  - Process Designer, Map Designer, Adapter Development Toolkit
  - Can separate connectivity, mapping and process design to parallel stream development and reduce time
  - Reduces upfront development costs, faster time to benefit
- Significantly reduced maintenance and upgrade costs
  - Fewer integration points and higher reuse means less to maintain
  - Central repository of integration points, monitoring tools and improved failure logic reduces uncontrolled errors and associated systems administration costs
  - Isolation layer reduces time/cost/risk for future enhancements and upgrades