Java Message Service (JMS)

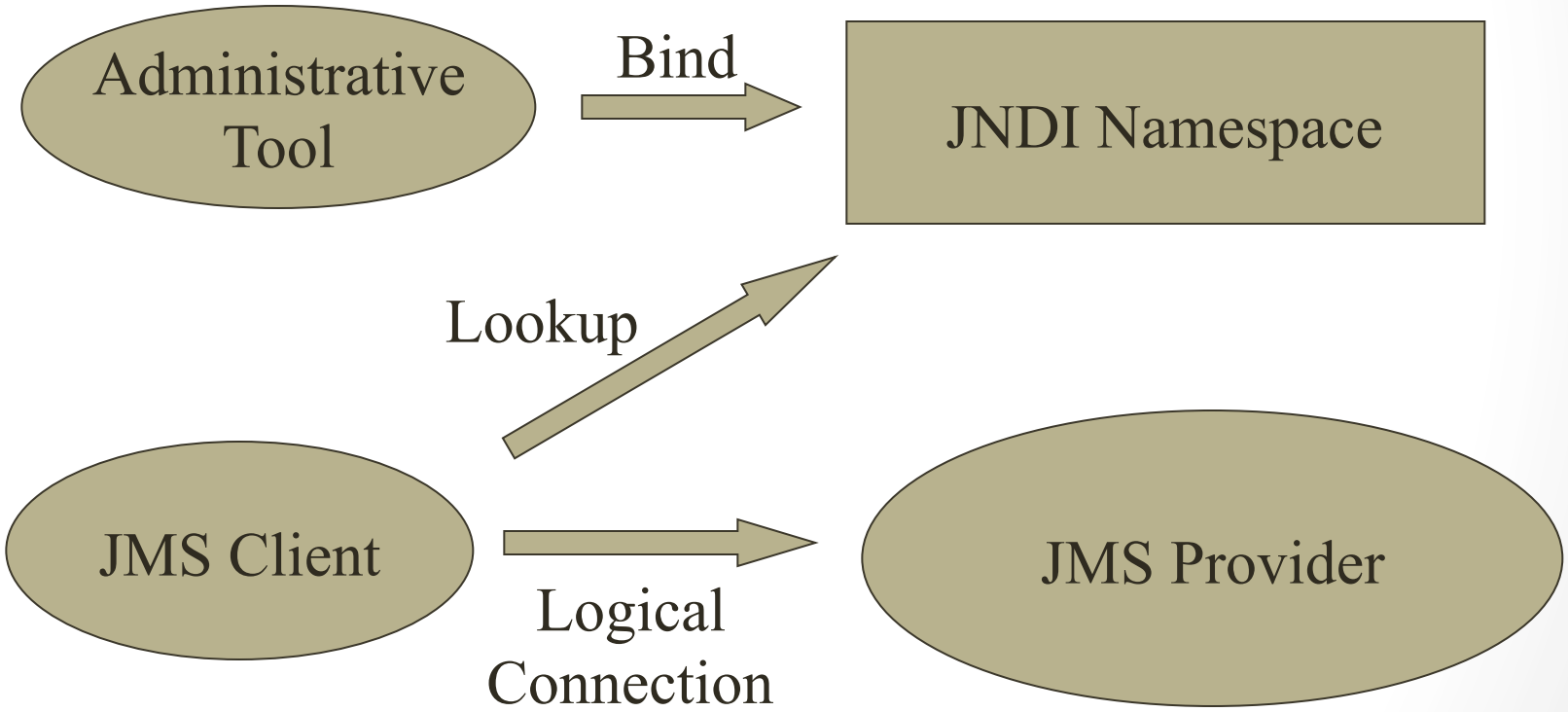
What is JMS?

- A common way for Java programs to
  - **create, send, receive**
  - and **read** distributed enterprise **messages**
- *loosely coupled* **communication**
- *Asynchronous* messaging
- ***Reliable*** delivery
  - A message is guaranteed to be delivered once and only once.

A JMS Application

- JMS Clients
  - Java programs that **send/receive** messages
- Messages
- Administered Objects
  - preconfigured JMS objects created by an admin for the use of clients
  - ConnectionFactory, Destination (queue or topic)
- JMS Provider
  - messaging system that implements JMS and **administrative** functionality

JMS Administration

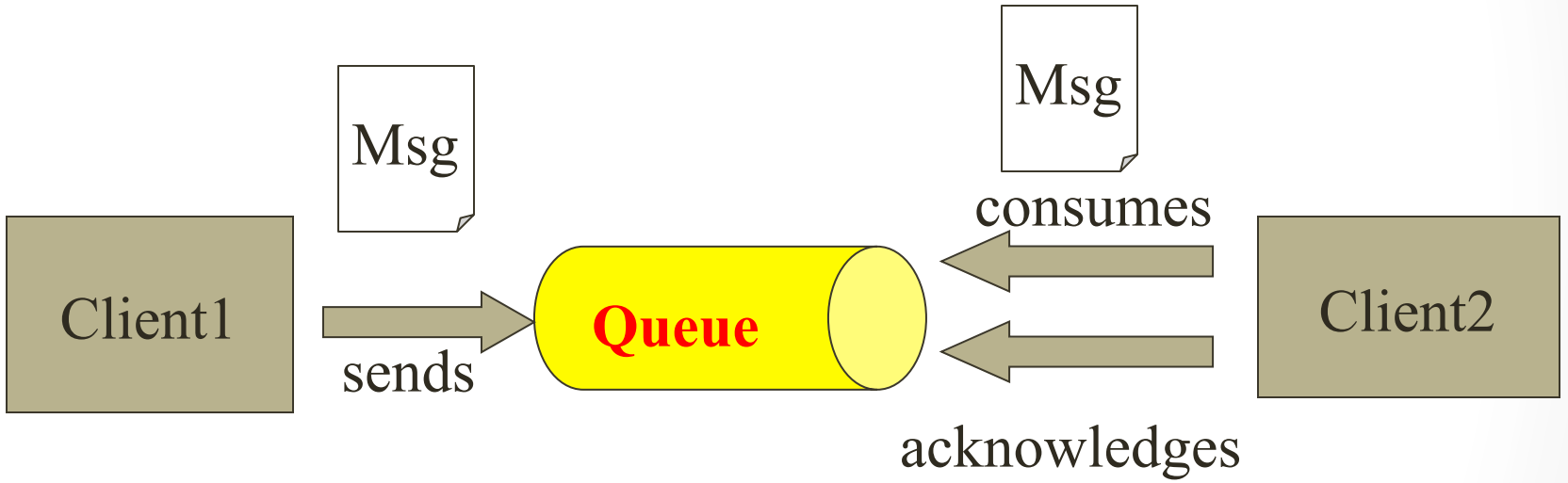


Java Naming & Directory Interface (JNDI)

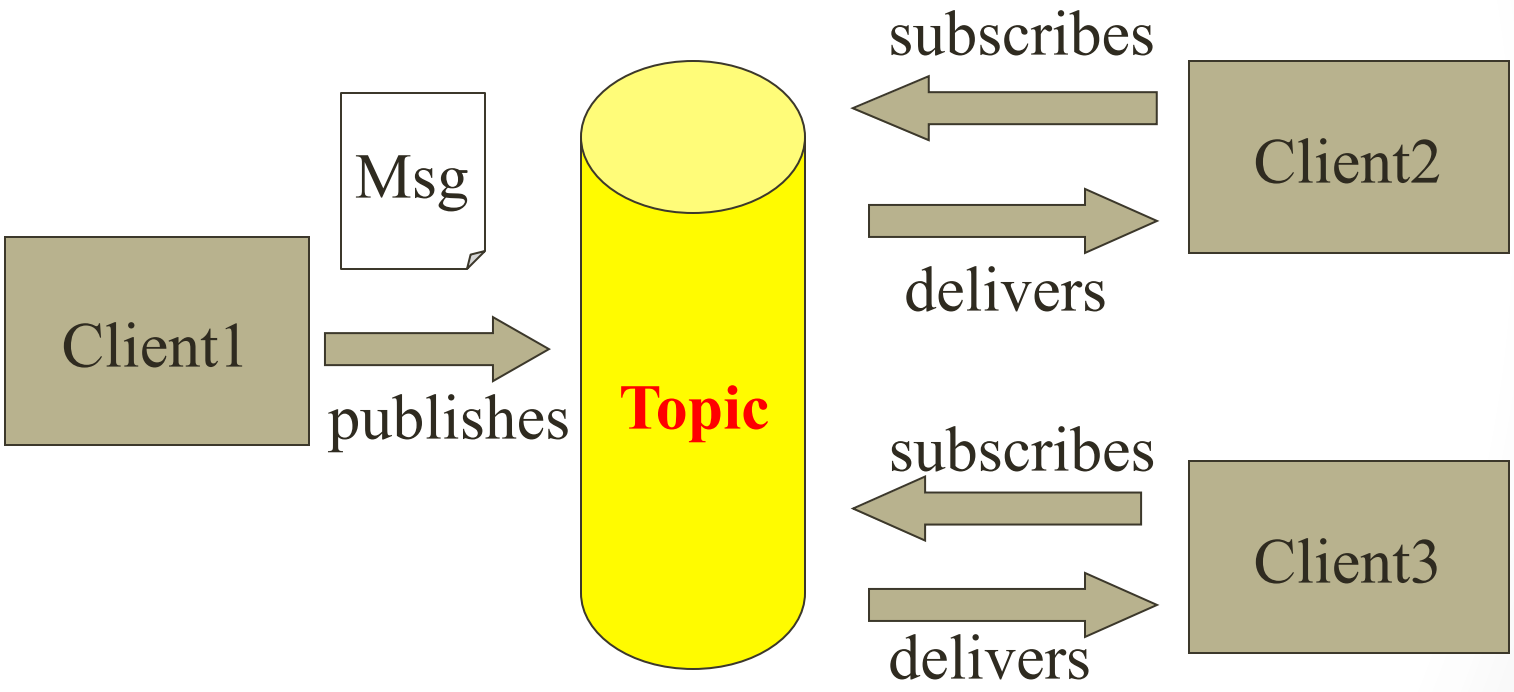
JMS Messaging Domains

- Point-to-Point (PTP)
  - built around the concept of message queues
  - each message has only one consumer
- Publish-Subscribe systems
  - uses a “topic” to send and receive messages
  - each message has multiple consumers

Point-to-Point (PTP) Messaging



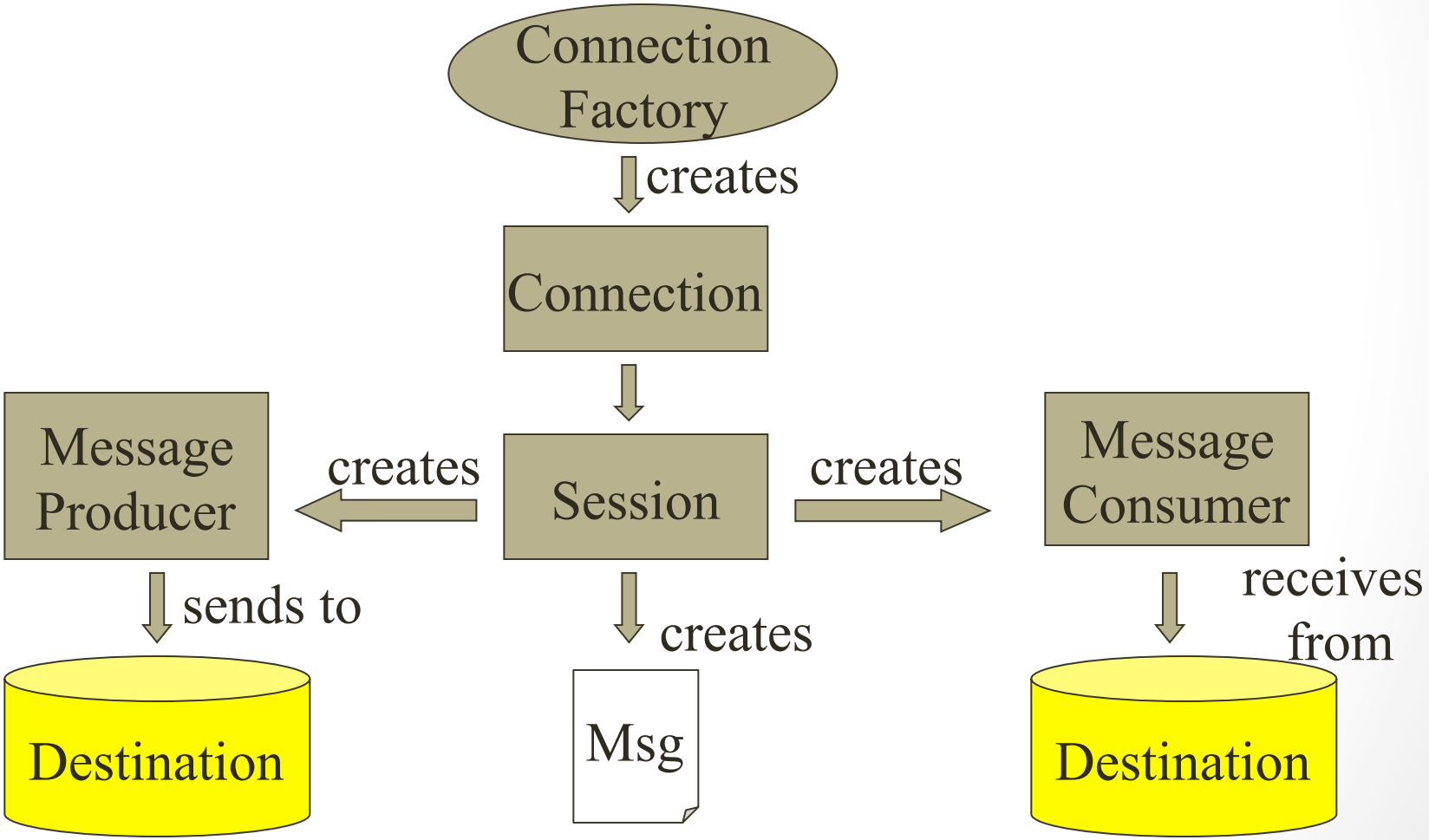
Publish/Subscribe Messaging



Message Consumptions

- **Synchronously**
  - A subscriber or a receiver explicitly fetches the message from the destination by calling the receive method.
  - The receive method can ***block*** until a message arrives or can time out if a message does not arrive within a specified time limit.
- **Asynchronously**
  - A client can register a *message listener* with a consumer.
  - Whenever a message arrives at the destination, the JMS provider delivers the message by calling the **listener's** onMessage() method.

JMS API Programming Model



JMS Client Example

- Setting up a connection and creating a session

```
InitialContext jndiContext = new InitialContext();
//look up for the connection factory
ConnectionFactory cf = jndiContext.lookup(connectionfactoryname);
//create a connection
Connection connection = cf.createConnection();
//create a session
Session session =
  connection.createSession(false,Session.AUTO_ACKNOWLEDGE);
//create a destination object
Destination queue = (Queue) jndiContext.lookup( “/jms/myQueue” );
//for PointToPoint
Destination topic = (Topic)jndiContext.lookup( “/jms/myTopic” );
//for publish-subscribe
```

Producer Sample

- Setup connection and create a session..
- Creating producer

```
MessageProducer producer =
        session.createProducer(dest1);
```

- Send a message

```
Message m = session.createTextMessage();
m.setText( “just another message” );
producer.send(m);
```

- Closing the connection

```
connection.close();
```

Consumer Sample (Synchronous)

- Setup connection and create a session..
- Creating consumer

```
MessageConsumer consumer =
        session.createConsumer(dest1);
```

- Start receiving messages

```
connection.start();
Message m = consumer.receive();
```

Consumer Sample (Asynchronous)

- Setup the connection, create a session…

- Create consumer
- Registering the listener

```
MessageListener listener=new MyListener();
consumer.setMessageListener(listener);
```

- `MyListener` should have onMessage()

```
public void onMessage(Message msg){
  //read the massage and do computation
}
```

Listener Example

```
public void onMessage(Message message) {
   TextMessage msg = null;
   try {
      if (message instanceof TextMessage) {
         msg = (TextMessage) message;
         System.out.println("Reading message:" + msg.getText());
      } else {
         System.out.println("Message of wrong type”);
      }
   } catch (Exception e) {
       System.out.println(“Exception: " + e.getMessage());
   }
}
```

JMS Messages

- Message Header
  - used for **identifying** and **routing** messages
  - contains vendor-specified values, but could also contain application-specific data
  - typically **name/value** pairs
- Message Properties (optional)
- Message Body(optional)
  - contains the data
  - five different message body types in the JMS specification

JMS Message Types

Message Type	Contains	Some Methods
TextMessage	String	getText,setText
MapMessage	set of name/value pairs	setString,setDouble,setLong,getDouble,getString
BytesMessage	stream of uninterpreted bytes	writeBytes,readBytes
StreamMessage	stream of primitive values	writeString,writeDouble,writeLong,readString
ObjectMessage	serialize object	setObject,getObject

More JMS Features

- Durable subscription
  - by default a subscriber gets only messages published on a topic while a subscriber is alive
  - durable subscription retains messages until a they are received by a subscriber or expire

More JMS Features

- Durable subscription
  - by default a subscriber gets only messages published on a topic while a subscriber is alive
  - durable subscription retains messages until a they are received by a subscriber or expire
- Request/Reply
  - by creating temporary queues and topics
    - Session.createTemporaryQueue()
  - producer=session.createProducer(msg.getJMSReplyTo());
    reply= session.createTextMessage("reply");
    reply.setJMSCorrelationID(msg.getJMSMessageID);
    producer.send(reply);

More JMS Features

- Transacted sessions
  - session=connection.createSession(true,0)
  - combination of queue and topic operation in one transaction is allowed
  - void onMessage(Message m) {
    try { Message m2=processOrder(m);
    publisher.publish(m2); session.commit();
    } catch(Exception e) { session.rollback(); }

More JMS Features

- Persistent/nonpersistent delivery
  - producer.setDeliveryMethod(DeliveryMode.NON_PERSISTENT);
  - producer.send(mesg, DeliveryMode.NON_PERSISTENT ,3,1000);
- Message selectors
  - SQL-like syntax for accessing header:

subscriber = session.createSubscriber(topic, "priority > 6 AND type = 'alert' ");

  - Point to point: selector determines single recipient
  - Pub-sub: acts as filter

JMS API in a Java EE Applications

- Java EE components can use the JMS API to send messages that can be consumed asynchronously by a specialized Enterprise Java Bean
  - **message-driven bean**

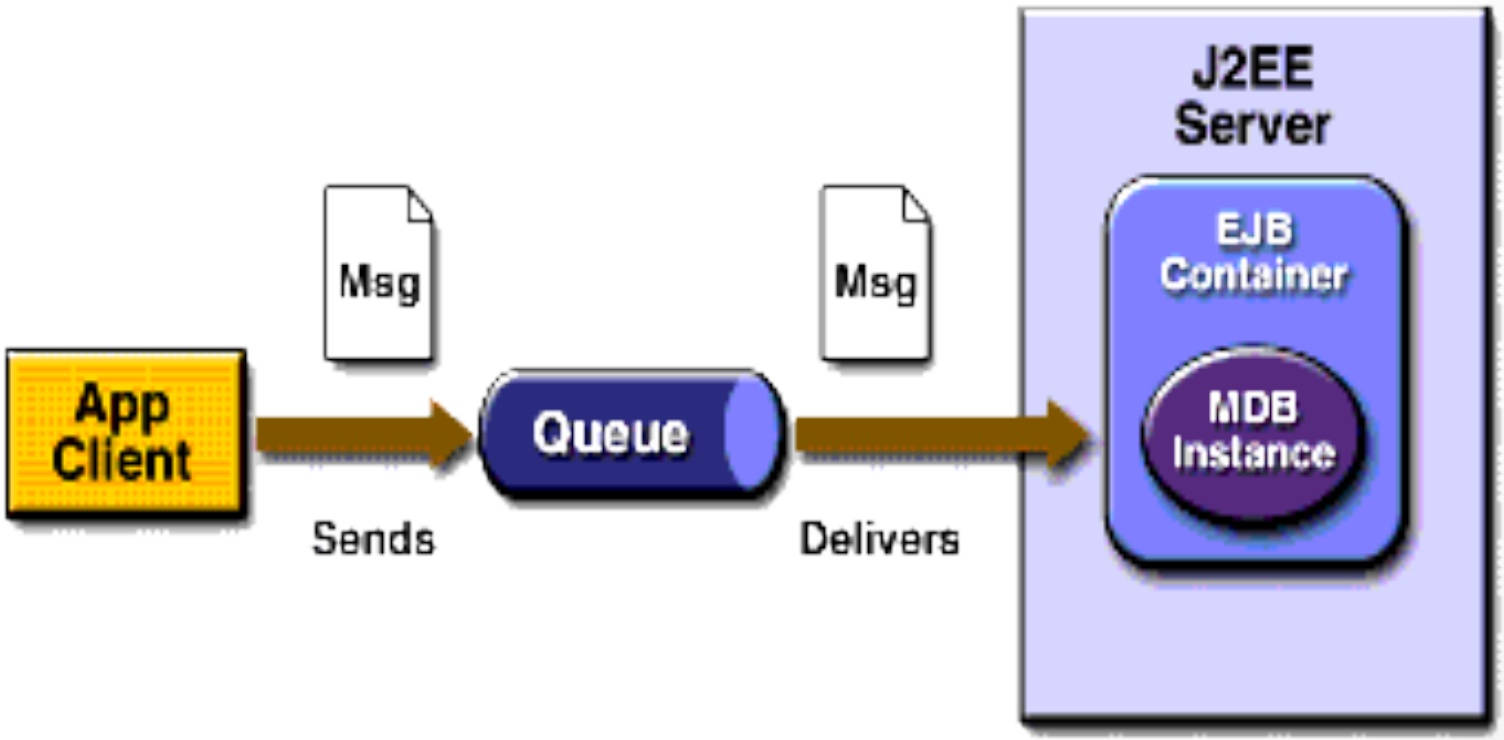
Enterprise Java Beans (EJB)

- EJB is a server-side component
  - encapsulates the **business logic** of an application
- EJB simplifies the development of large, distributed applications
  - EJB Container provides system-level services
    - e.g. **transaction management, authorization**
  - Control logic

Message–Driven Bean

- acts as a listener for the JMS,
  - processing messages **asynchronously**
- **specialized** adaptation of the JMS API used in the context of Java EE applications

JMS with EJB Example



MDB Example

```
public class MessageBean implements MessageDrivenBean,
                                        MessageListener{
  public void ejbCreate(){}
  public void ejbRemove(){}
  public void setMessageDrivenContext(MessageDrivenContext mdc){}

 pubic void onMessage(Message m){
     //do computation on the incoming message
    try {
        if (m instanceof TextMessage) {
              System.out.println("Mbean:message" + m.getText());
        }
     } catch (JMSException exp){ ...}
  }
}
```

JMS and JNDI

- JMS utilizes Java Naming & Directory Interface(JNDI).
- Advantages:
  - It **hides provider-specific details** from JMS clients.
  - It **abstracts** JMS **administrative information** into Java objects that are easily organized and administrated from a common management console.

SOAP and JMS

- Use JMS as a transportation layer for SOAP
- Example: Sun™ ONE Message Queue
  - enables to send JMS messages with **SOAP payload**
  - transportation of SOAP messages **reliably** and **publishing** SOAP messages to JMS subscribers

SOAP and JMS (using Sun™ ONE MQ)

Send a SOAP message

- Create a JMS session
- Create a SOAP message
- Transfer the SOAP message into JMS message

```
Message msg = MessageTransformer.SOAPMessageIntoJMSMessage
                                  (SOAPMessage, Session);
```

- Send the JMS message

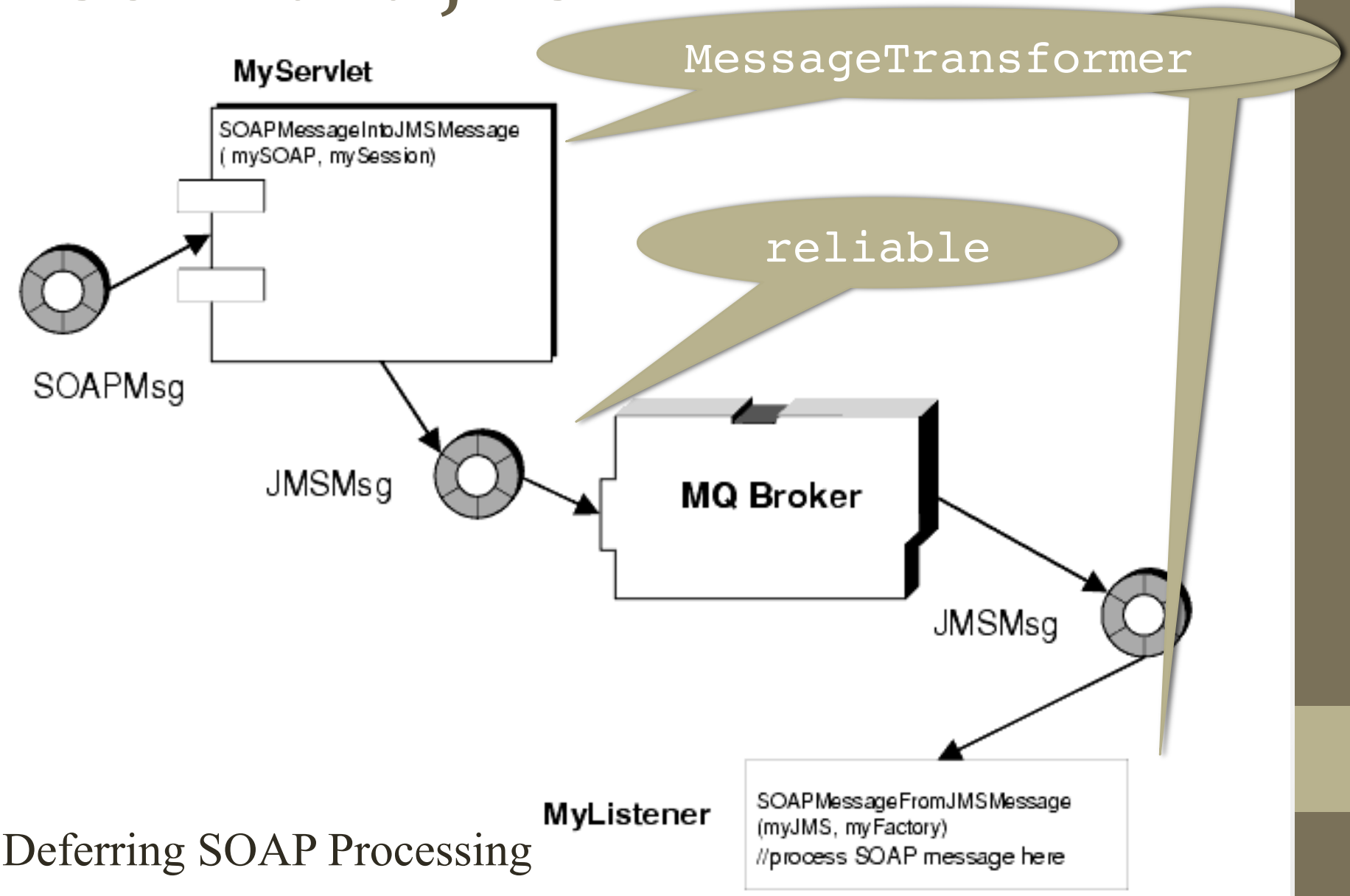
SOAP and JMS (using Sun™ ONE MQ)

Receive a SOAP message

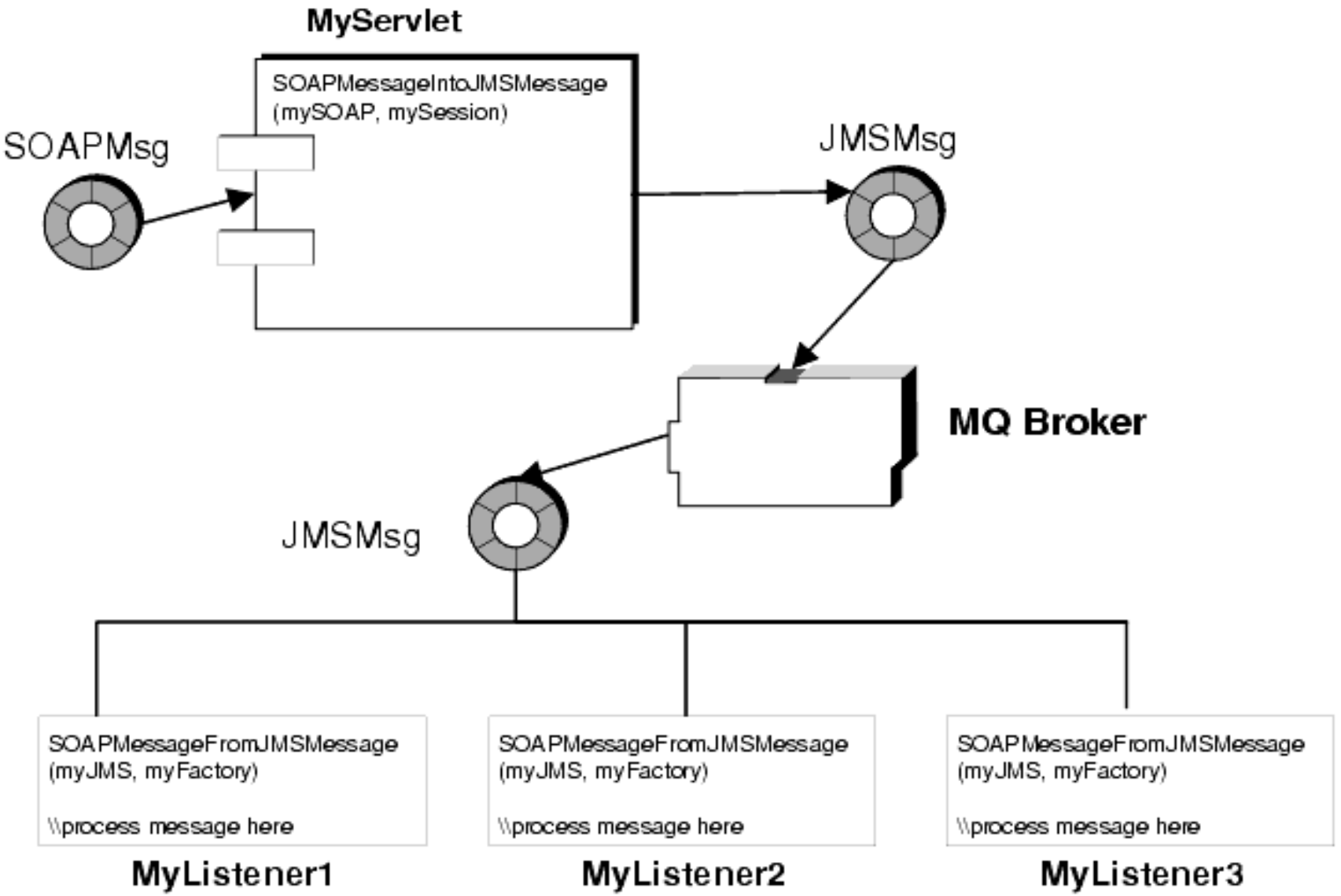
- Create a JMS session
- Receive the JMS message
- Transfer the JMS message into SOAP message

```
SOAPMessage msg = MessageTransformer.SOAPMessageFromJMSMessage
                                    (Message, MessageFactory);
```

SOAP and JMS



Deferring SOAP Processing



Publishing a SOAP message

JMS Providers

- SunONE Message Queue (SUN)
- MQ JMS (IBM)
- WebLogic JMS (BEA)
- JMSCourier (Codemesh)
  - merging C++ applications into a JMS environment