Messaging

Mike Calvo

Citronella Software

Agenda

- Messaging Systems
- Java Messaging Service (JMS)
- Grails JMS Support

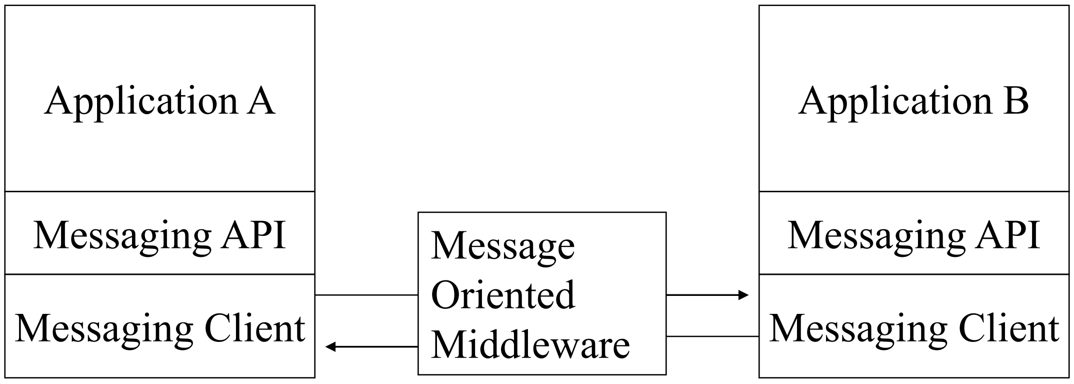
Messaging

- Asynchronous communication approach
- System components send messages to communicate
- Message sent to a channel
  - Not a component
- Messages are self contained packages of data and network routing information
- Message-Oriented-Middleware (MOM) provides message sending and delivery functionality

Messaging Predates RMI

- Messaging not dependent on TCP/IP protocol
- Messaging not dependent on Object-Oriented programming languages
  - COBOL
- Several messaging products have existed for many years
  - MQSeries (IBM)
  - MSMQ (Microsoft)
  - Rendevous (TIBCO)

High Level Messaging Architecture



Messaging client responsible for both sending and receiving messages

Messaging Concepts

- Channel
  - Virtual pipe that connects a sender to a receiver
- Message
  - Data transmitted on a channel
- Pipes and Filters
  - Chains of operations that can be performed on messages between sender and receiver

More Messaging Concepts

- Endpoint
  - Sender of data
  - Receiver of data
- Routing
  - Directing messages to the correct receiver

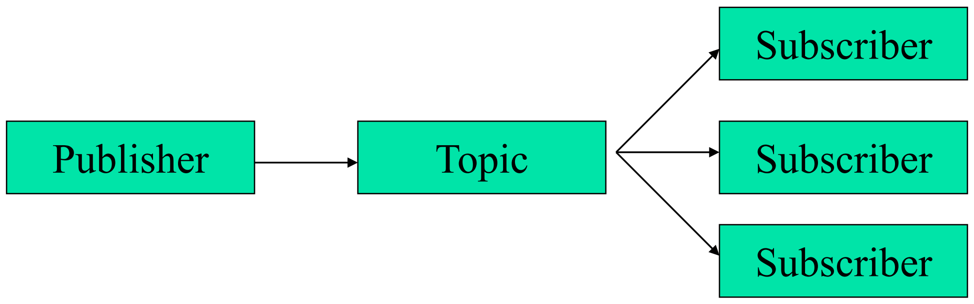
Messaging Channel Types

- Publish-and-subscribe
  - 1 to many
- Point-to-point
  - Queueing
  - 1 to 1

Publish-and-subscribe

- Subscription model
- Topic is a virtual channel
  - Conceptually like a discussion
  - Relates to something of interest
- Publisher is a message producer that sends a message to a topic
- Subscriber is a topic message consumer
  - Multiple subscribers can subscribe to a topic
  - Every subscriber receives a copy of the message
  - Subscription may be “durable”
- Typically a push model

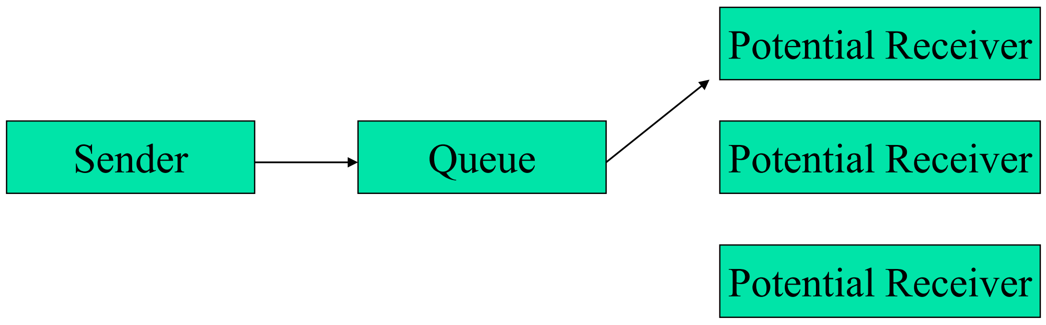
Publish-and-subscribe



Point-to-point

- Queue model
  - Messages are added to a queue for delivery
  - Messages are delivered when a consumer is available
- Queue can be implemented with
  - Pull model
  - Push model
- Queue may have multiple potential receivers
- Message producer adds a message to the queue
- Each message in the queue is guaranteed to only be consumed once
- Optional features: load balancing and queue browsing

Point-to-point



Selecting Delivery Model

- Strictly based on application requirements
- Publish-and-subscribe is used when many different types of components need to know when an event occurs
  - Example: Client wants to receive stock quotes
- Point-to-point is typically used when one type of resource can handle the event
  - Can represent a single resource
  - Can represent a pooled resource
  - Example: Client requests a buy order from a trader

Message

- Multiple parts
- Header
  - Metadata about the message
  - Used by messaging system
- Body
  - Payload
  - Typically ignored by messaging system

Classic Message Structure

- XML Message
  - Can be SOAP
  - Typed encoding
  - Header
  - Body
- Designed for cross-platform use

Pipes and Filters

- A Filter is a specific processing step on a message
- A Pipe is a channel that is passed into a filter and passed out of the filter allowing filters to be chained together
- Filters can be applied sequentially or in parallel

Uses of Pipes and Filters

- Encryption
- Audit logging
- Authorization
- Compression

Message Router

- A type of Filter
- Consumes a message from one channel and republishes it to another based on conditions
- Usually information in the message
  - Header information
  - Body information (Content-based)

Uses of Message Routers

- Decouple the sender of a message from it’s destination
- Load balancing
- Message Brokering
  - Move routing of messages out of application logic

Message Translation

- Message data may often take different formats
- Message Translators convert from the sending system’s data format to the receiving system’s format
- Can occur at many levels

Translation Levels

- Data Structures
  - Application layer representations
- Data Types
  - Strings become integers
- Data Representation
  - XML versus name value pairs versus fixed length data fields
- Transport
  - Communication protocols (JMS, HTTP, Sockets, etc)

Messaging Features

- Guaranteed delivery
  - Messages may need to be persisted in the event of server failure
  - Store-and-forward
- Security
  - Recipients of messages may need to be authorized to receive message

Messaging Benefits

- Queues and Topics allow large systems to be highly flexible and dynamic
  - Topics and queues dynamic
  - Producers, subscribers and consumers dynamic
  - Auditing and logging can be added dynamically
- Heterogeneous systems can communicate by specifying simple messaging information
  - Language and environment neutrality
- Much less tightly coupled approach than RMI

Tightly Coupled Synchronous Service Calls

- Large systems rely on interdependence of many systems
- With synchronous service calls
  - Failure of one component to communicate with another can prevent the entire system from functioning
  - Modification of service methods may require modification of other components to use it
  - Synchronous calls can delay processing needlessly
- Messaging provides a good alternative for communicating system to system

Messaging Challenges

- Complex Programming Model
- Sequence issues
- Synchronous scenarios
- Performance overhead
- Limited platform support
- Vendor lock-in

Commercial Messaging Systems

- Operating Systems
  - Windows MSMQ
- Application Servers
  - J2EE Application Servers
- EAI Suites
  - Enterprise Application Integration
- Cloud-based
  - Microsoft Azure

Java Message Service

- Vendor neutral Java API for accessing enterprise messaging systems
- Resource adapter
  - Similar to JDBC and JNDI
- Includes messaging client API for
  - Message sending (Producer)
  - Message receiving (Consumer)

JMS Concepts

- Connection Factory
- Connection
- Session
- Message Producer
- Message Consumer
- Destination
- Message

JMS Connection

- Represents a connection to the JMS Server
  - TopicConnection and QueueConnection
- Queue and topic versions
- Provides
  - Starting and stopping message traffic
  - Client authentication
  - Creating JMS Sessions
  - Connection metadata
- Obtained from a ConnectionFactory
  - Queue and topic connection factories are JNDI registered services

JMS Session

- Single-threaded context for producing and consuming messages
- Factory for creating
  - Messages
  - Producers
  - Consumers
- Queue and topic versions
- Specify Destination when creating Producer/Consumer
- Supports
  - transactional behavior
  - Acknowledgement modes

JMS Producer

- Used by client to send Messages to a Destination
- Two specific types
  - QueueSender
  - TopicPublisher
- Allows client to specify
  - Priority
  - Time to live
  - Delivery mode

JMS Consumer

- Used by client to receive Messages
  - Client registers a MessageListener with the Consumer
  - Messages are delivered to onMessage()
- Two types
  - QueueReceiver
  - TopicSubscriber
- Messages are
  - Pushed from JMS server
  - Received serially
- Supports
  - Message selector (message filter)

JMS Message

- Represents a message in the system
- Contains header and payload
- Several types of messages (payload-based):
  - Message (simple – no payload)
  - TextMessage
  - ObjectMessage
  - BytesMessage
  - StreamMessage
  - MapMessage
- Created by JMS Session

JMS Destination

- JMS administered object where messages can be delivered
- Types of Desinations
  - javax.jms.Topic
  - javax.jms.Queue
  - javax.jms.TemporaryTopic
  - javax.jms.TemproraryQueue
- Supports concurrent use
- Vendor-specific implementation
- Registered via JNDI to naming service

JMS Providers

- Most full JEE servers provide
  - JBoss, WebSphere, WebLogic
- ActiveMQ
  - Standalone Apache implementation

Grails and JMS

- Grails plugin
  - grails install-plugin jms
- Simplifies
  - Sending messages
  - Listening to messages
  - Creating destinations
- ActiveMQ plugin
  - grails install-plugin activemq

Grails JMS Plugin

- Provides access to Spring JMS support within Grails application
- jmsService can be injected into controllers and services
- Grails services can be exposed as JMS listeners

ActiveMQ Plugin

- Host ActiveMQ implementation of JMS from within Grails application
  - Typically not best practice for production but good for test and development

Common Asynchronous Scenario

- Sending email
- Grails support for email provided via plugins

Grails Email Plugins

- Mail Plugin
  - grails install-plugin mail
  - mailService available to controllers/services
- Greenmail Plugin
  - Good for testing email sending
  - grails install-plugin greenmail
  - Contributed by famous author