The Art of Messaging

Designing Effective Messaging Systems

Mark Richards
Director and Sr. Architect, Collaborative Consulting, LLC
Author of Java Message Service 2nd Edition (O'Reilly)

topics

messaging use cases

the jms api and basic messaging

message design considerations

a brief look at restful jms messaging

messaging use cases

primary messaging features

asynchronous requests - never having to wait

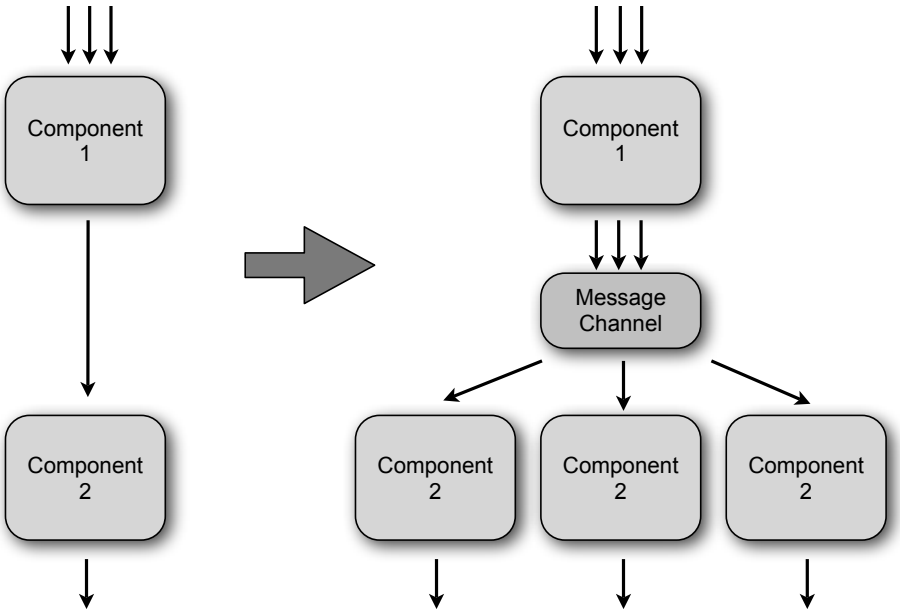
guaranteed delivery - knowing it will definitely get there

load balancing - doing multiple things at once

heterogeneous integration

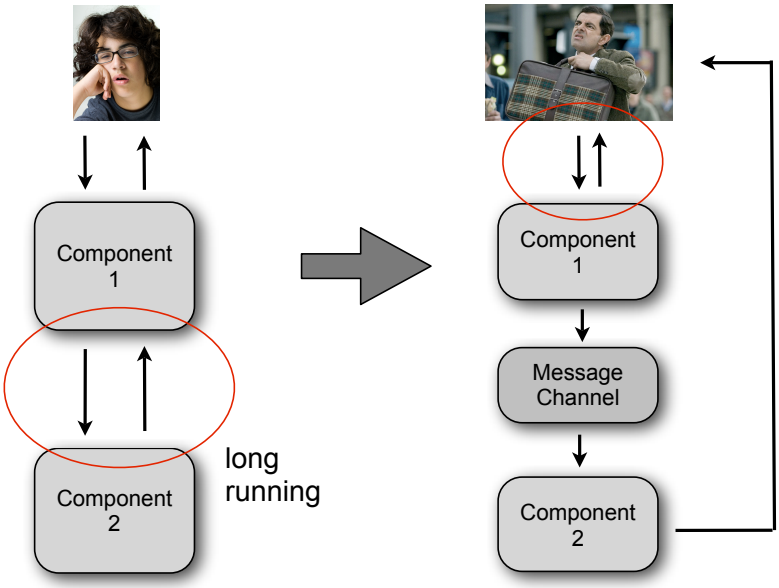
communicating with disparate components

reduced bottlenecks and increased scalability / throughput



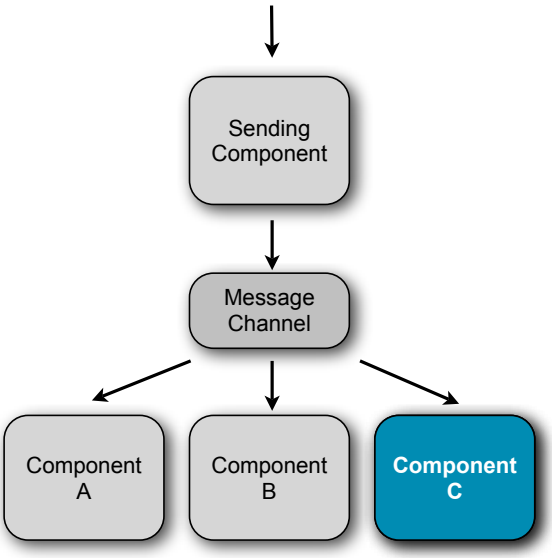
end user productivity

why make the user wait for long-running requests?



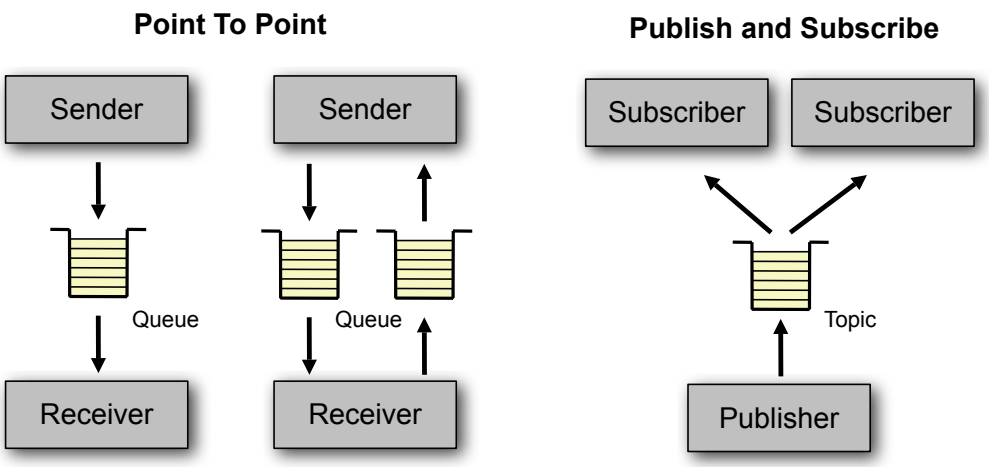
architecture agility

agility through abstraction and decoupling

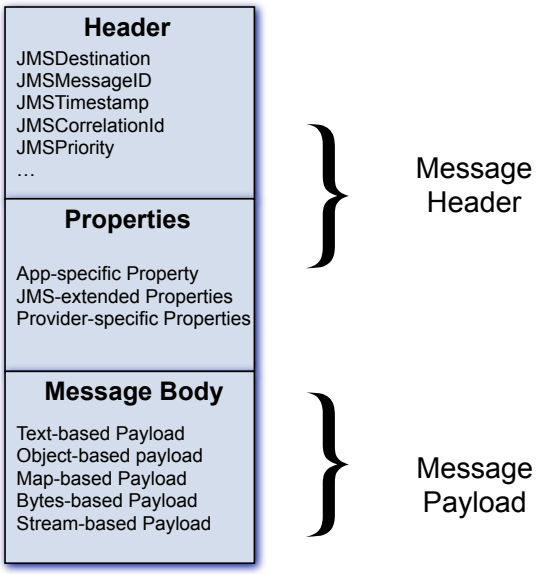


the jms api and basic messaging

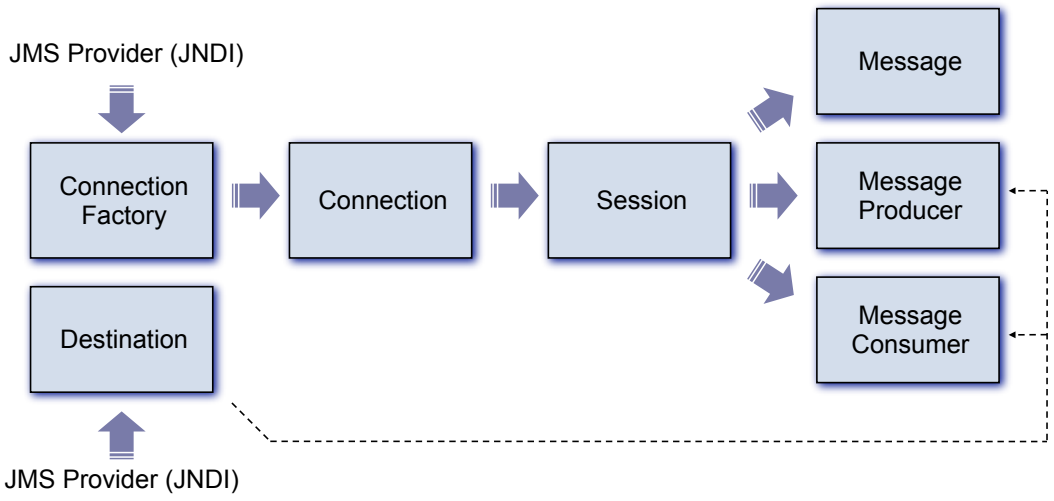
messaging models



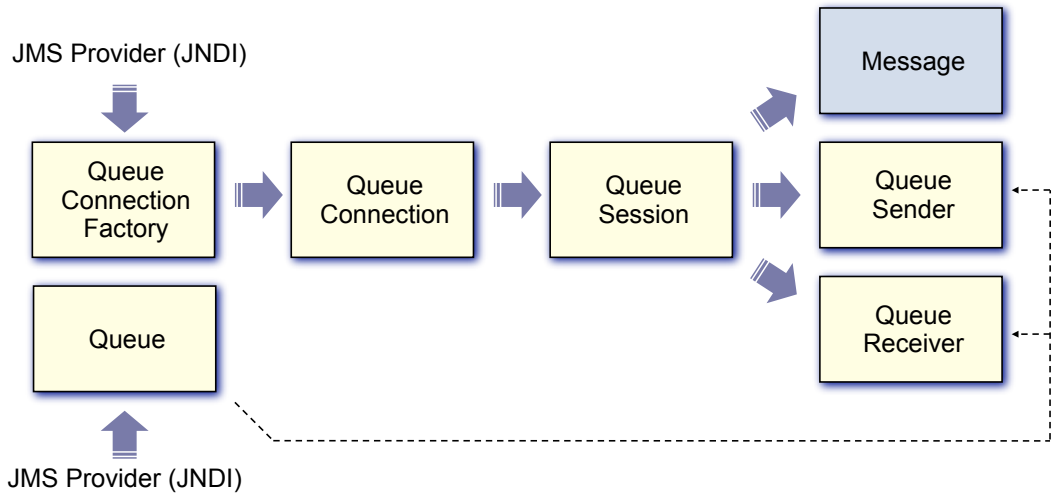
jms message structure



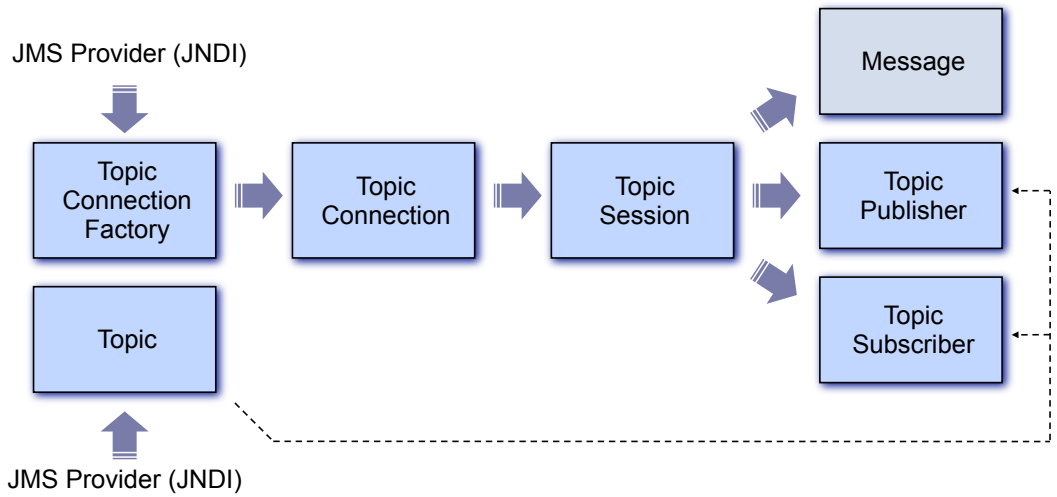
jms api - generic interfaces



jms api - queue-based interfaces



jms api - topic-based interfaces



messaging demo

messaging in java and groovy

messaging design considerations

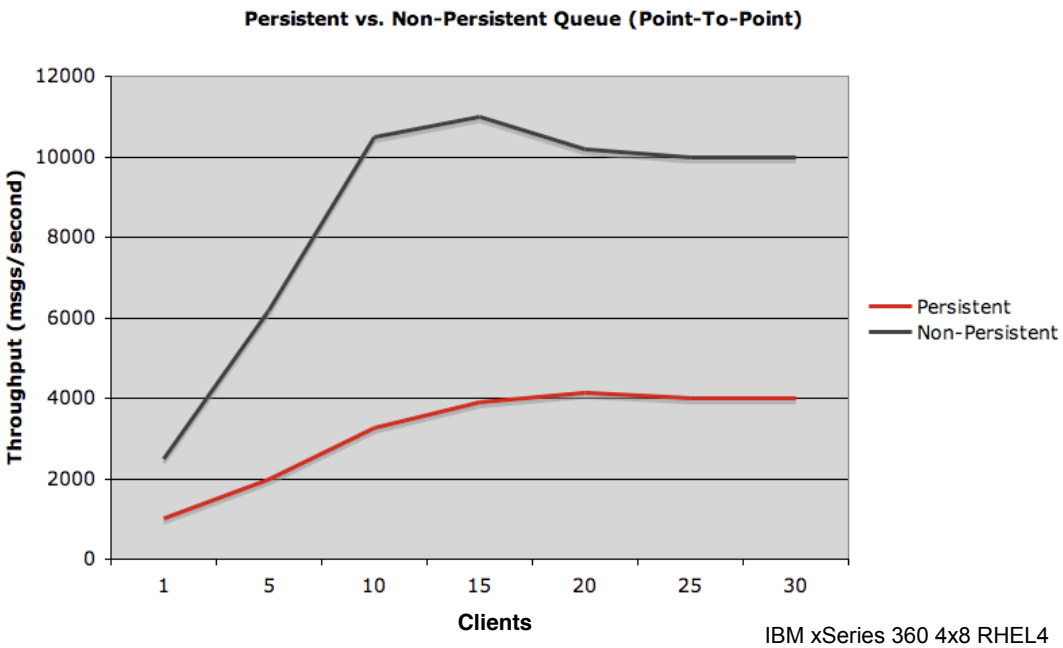
message persistence

by default, all messages are marked as PERSISTENT

persistence is required to support guaranteed delivery

but, there are trade-offs to consider....

message persistence



IBM xSeries 360 4x8 RHEL4
WMQ6.0 2K MSG

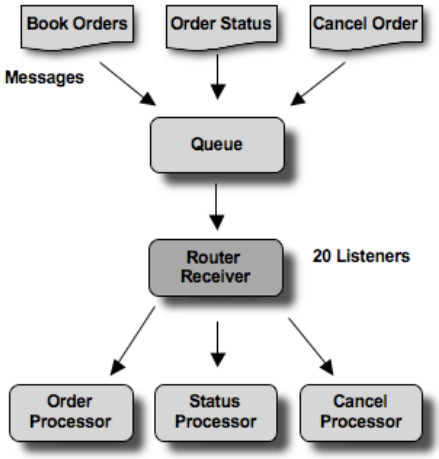
message persistence

set through the message **DeliveryMode**

```
//can be set for all messages
QueueSender sender = session.createSender(request);
sender.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

//can be set for a single message
TextMessage msg = session.createTextMessage();
msg.setText(xml.toString());
sender.send(msg, DeliveryMode.NON_PERSISTENT, 4, 30000);
```

single queue design approach



single queue design approach

```
public void onMessage(Message message) {
   try {
      String xml = ((TextMessage)message).getText();
      int type = message.getIntProperty("type");

      if (type == NEW_BOOK_ORDER) {
         orderProcessor.placeOrder(xml);
      } else if (type == ORDER_STATUS) {
         statusProcessor.checkOrderStatus(xml);
      } else if (type == CANCEL_ORDER) {
         cancelProcessor.cancelOrder(xml);
      } else {
         throw new Exception("Invalid Order Type: " + type);
      }
   } catch (Exception up) {
   ...
}
```

single queue design approach

PLUS

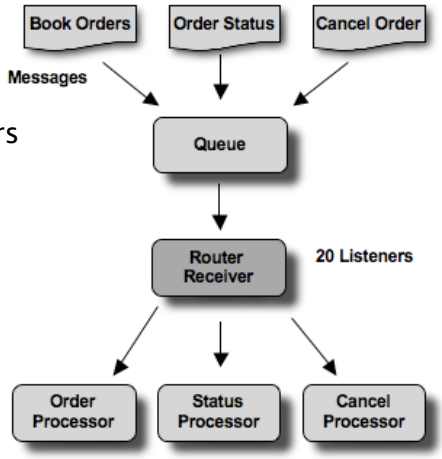
decoupled processors

ease of testing

ease of maintenance

design simplicity

design flexibility



MINUS

poor load balancing

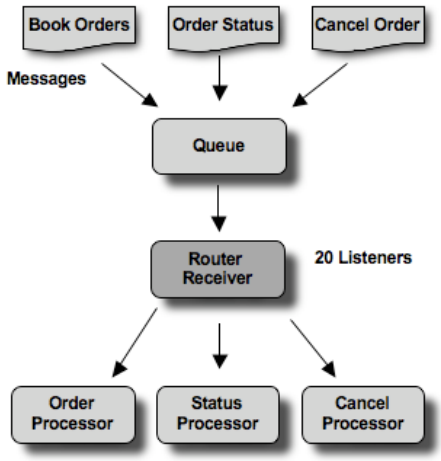
poor response time

poor concurrency

homogeneous processors

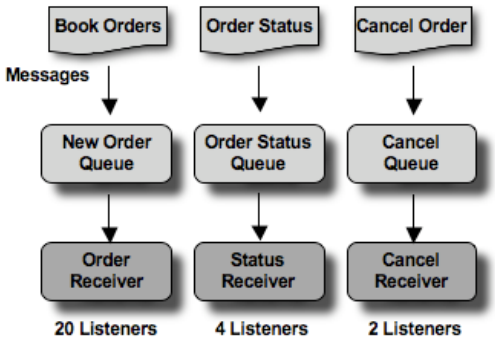
poor overall agility

single queue design approach

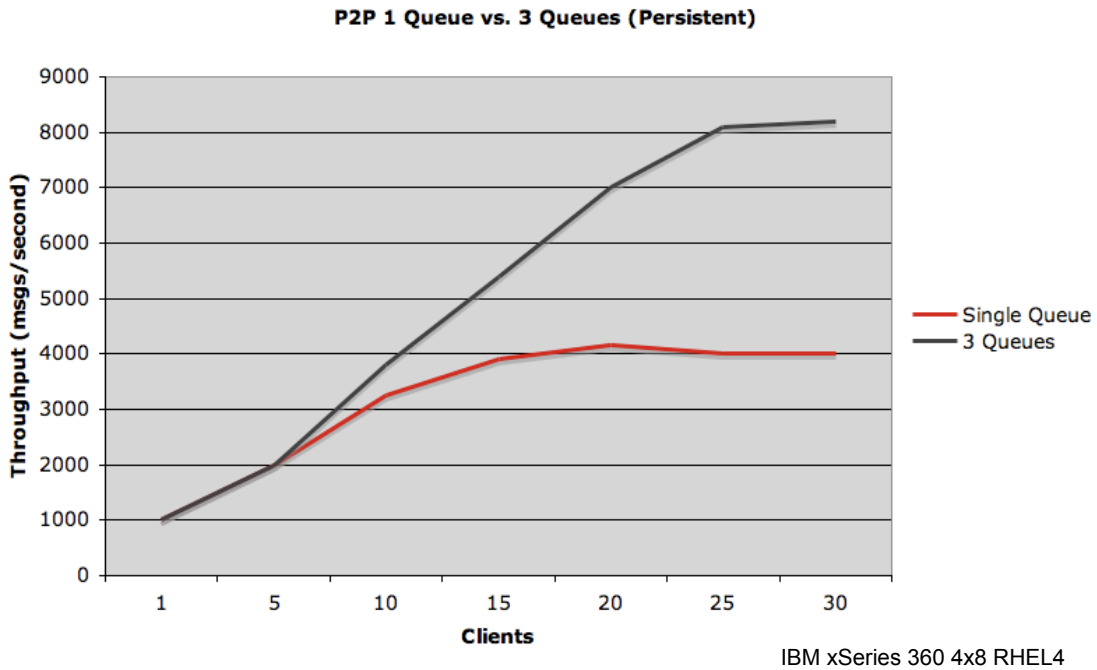


single queue design approach

a better design...

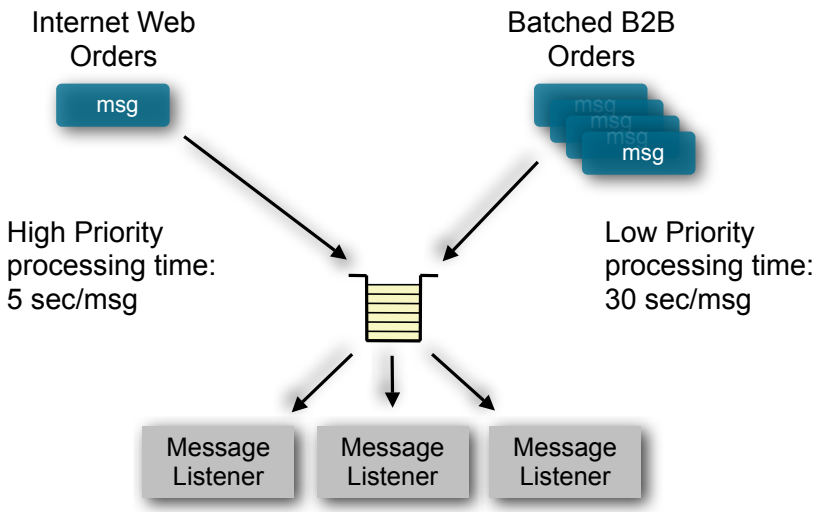


single queue design approach



IBM xSeries 360 4x8 RHEL4
WMQ6.0 2K MSG

using message priority



3 concurrent listener threads

using message priority

let's do the math....

T0: system is idle, queue is empty

T1: 20 batched orders come in from the b2b portal at low priority

T2: 3 batched orders get picked up by listeners

T3: 5 web orders come in from web portal

==> web order 1: ~35 second response time
==> web order 2: ~35 second response time
==> web order 3: ~35 second response time
==> web order 4: ~40 second response time
==> web order 5: ~40 second response time

Consider separate queues, even for the same type of message

a brief look at restful jms

java message service
vs.
web services

internal (inside the firewall): use JMS

external (outside the firewall): use Web Services

representational state transfer

architecture style where resources are accessed through verbs and nouns and represented as documents:

HTTP GET http://hostname/customer/1234

resources and representations	URI universal resource identifier	http methods GET, POST, PUT, DELETE
what is it?	where is it?	what action?

jms and rest

not a perfect match...

does a GET on a queue remove the next message from the queue or just browse the next message?

does a DELETE action on a queue pull the message off or delete the actual queue?

does a POST action on a queue send a message to a queue or create a new (or temporary) queue?

jms rest vendor options

websphere mq mappings

http://hostname/msg/queue/queue1

HTTP header	→	MQMD headers
text/plain or text/html	→	MQFMT_STRING (string msg)
other media types	→	MQFMT_NONE (binary msg)

HTTP GET	→	MQGET with browse
HTTP POST	→	MQPUT
HTTP DELETE	→	MQGET
HTTP PUT	→	no mapping

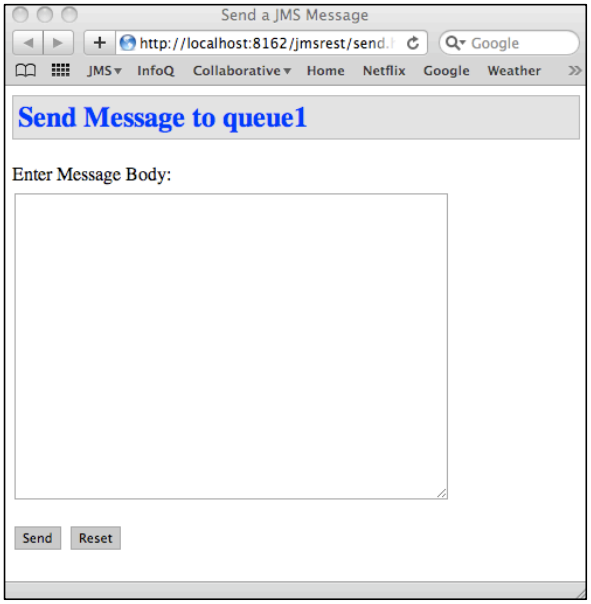
activemq mappings

http://hostname/approot/message/jms/queue1?type=queue

hidden post variables	→	message header properties
any content type	→	TextMessage

HTTP GET	→	receive() (for convenience)
HTTP POST	→	send()
HTTP DELETE	→	receive()
HTTP PUT	→	no mapping

Restful JMS Demo



summary and closing

hands-on unconference topics

publish and subscribe messaging

sending images and documents

sending large messages (message chunking)

transacted sessions

messaging acknowledgement modes

durable and non-durable subscribers

request/reply messaging (message correlation)

messaging pitfalls and how to avoid them

references

- Java Message Service, 2nd Edition (Mark Richards, O'Reilly, 2009)
  - http://oreilly.com/catalog/9780596522049/index.html
- JMS API
  - http://java.sun.com/products/jms/index.jsp
- ActiveMQ
  - http://activemq.apache.org
- Source Code
  - http://www.wmrichards.com/nfjs