

J-Spring 2007

Enterprise Integration Patterns in Action

Jos Dirksen



Agenda

- 🚀 **Introduction**
- 🚀 The what and why of EIP
- 🚀 EIP in Action
- 🚀 Summary
- 🚀 Questions



Who am I?

- 🚀 Jos Dirksen
- 🚀 Software Architect
 - ☛ Work for Atos Origin
 - ☛ BAS Emerging Technologies
- 🚀 Focus on:
 - ☛ Open Source
 - ☛ Integration
- 🚀 Working on book:
 - ☛ Open Source ESBs in action
 - ☛ Together with Tijs Rademakers





What are we going to do?

- ✈ Short introduction in EIP
- ✈ No theoretical walk through of 65+ patterns
- ✈ What you get is about 8 patterns
 - ☛ Based on a case study
 - ☛ We're going to implement the patterns live

Request application

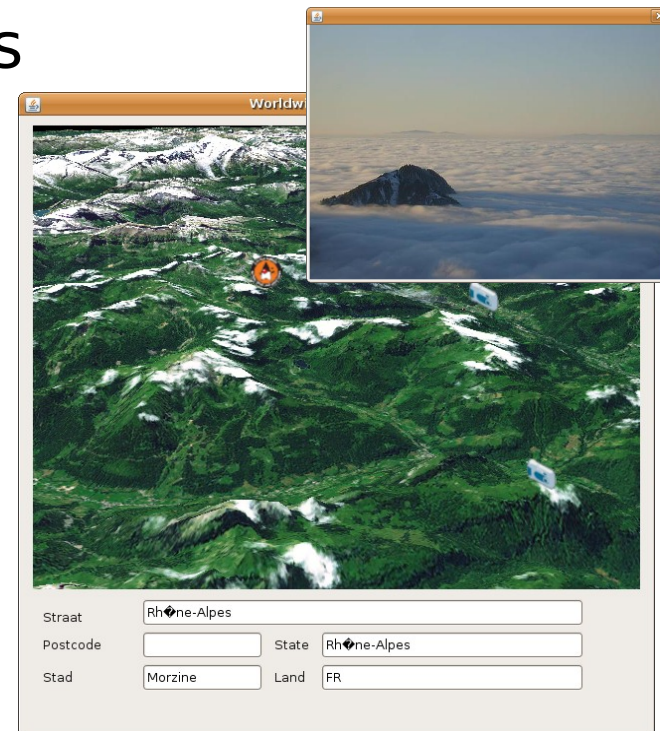
Location:

Message:

Images:

Errors:

EI Patterns





Agenda

- 🚀 Introduction
- 🚀 **The what and why of EIP**
- 🚀 EIP in Action
- 🚀 Summary
- 🚀 Questions



Why enterprise integration?

- ✈ There is usually more than one application
 - ☛ Custom build applications
 - ☛ Mergers
 - ☛ Legacy systems
 - ☛ SAP etc.
- ✈ Customers do not think about system boundaries
 - ☛ One request can span many applications
- ✈ Applications need to be integrated to:
 - ☛ Support common business processes
 - ☛ Support data sharing between applications
 - ☛ Promote reuse of information and services



Challenges of EI

- ✈ Vital for the business
 - ☪ Politics
 - ☪ It doesn't end with deployment
- ✈ Wide diversity:
 - ☪ Different OS: Windows, HP-UX, OS/400, Linux etc.
 - ☪ Different protocols: http, ftp, mail, file, legacy, custom
 - ☪ Different languages: java, .net, cobol
- ✈ Limited amount of control
 - ☪ Need to integrate with proprietary systems
 - ☪ Limited amount of control over the entities



How can patterns help?






“Anyone who claims that integration is easy must be incredibly smart, incredibly ignorant or have a financial interest in making you believe that integration is easy”

Hohpe, Woolf

- 🚀 Standard proven solutions
 - 🍪 Learned by “trial and error”
- 🚀 Fill the gap between vision and implementation
- 🚀 No “shrink-wrapped” solutions
- 🚀 Common dictionary



Agenda

-  Introduction
-  The what and why of EIP
-  **EIP in Action**
-  Summary
-  Questions

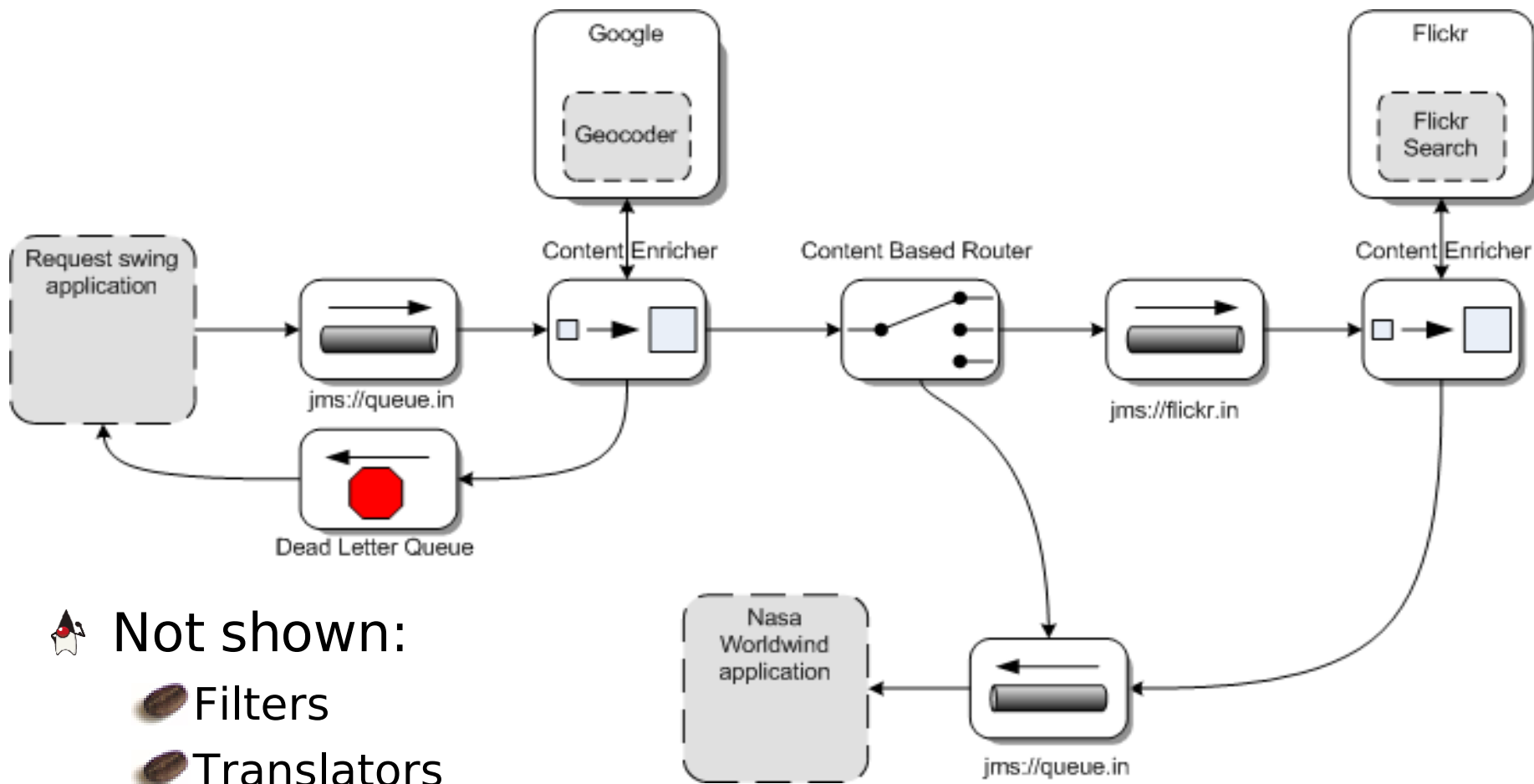


Case study

- ✈ Swing app to make “location” requests
- ✈ Swing app to show the results of these requests
 - ☛ Shows location on NASA Worldwind map
 - ☛ Shows the message send by the client
 - ☛ Shows images shot in the area
- ✈ Integration middleware
 - ☛ Calls Geocoder API to translate location to coordinates
 - ☛ Calls flickr API to search for images shot in the area
 - ☛ And of course glues everything together...
 - ☛ Implement with Mule for this example



EIP Case







🚀 Not shown:

- ☕ Filters
- ☕ Translators






Case part 1

Message channel pattern:

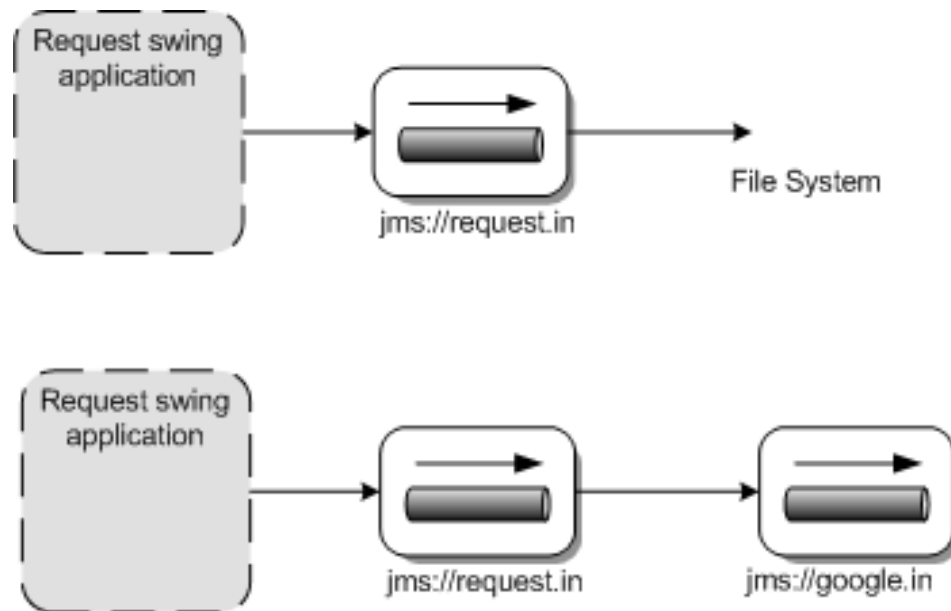
-  Asynchronous, reliable communication
-  Can function when sender/receiver are not present
-  Stores and holds message until receiver is available
-  For example: JMS, WS-RM, file system, mail

Canonical data model pattern:

-  Applications use different formats
-  Translator pattern can solve this partially
-  Defines a standard format for the messages



Case part 1





Case part 2

✈️ Translator pattern:

- ☘ Translate from one data format to the other
- ☘ Translate from application specific to canonical format
- ☘ For example: XSLT, XML Mappers

✈️ Content-enricher pattern:

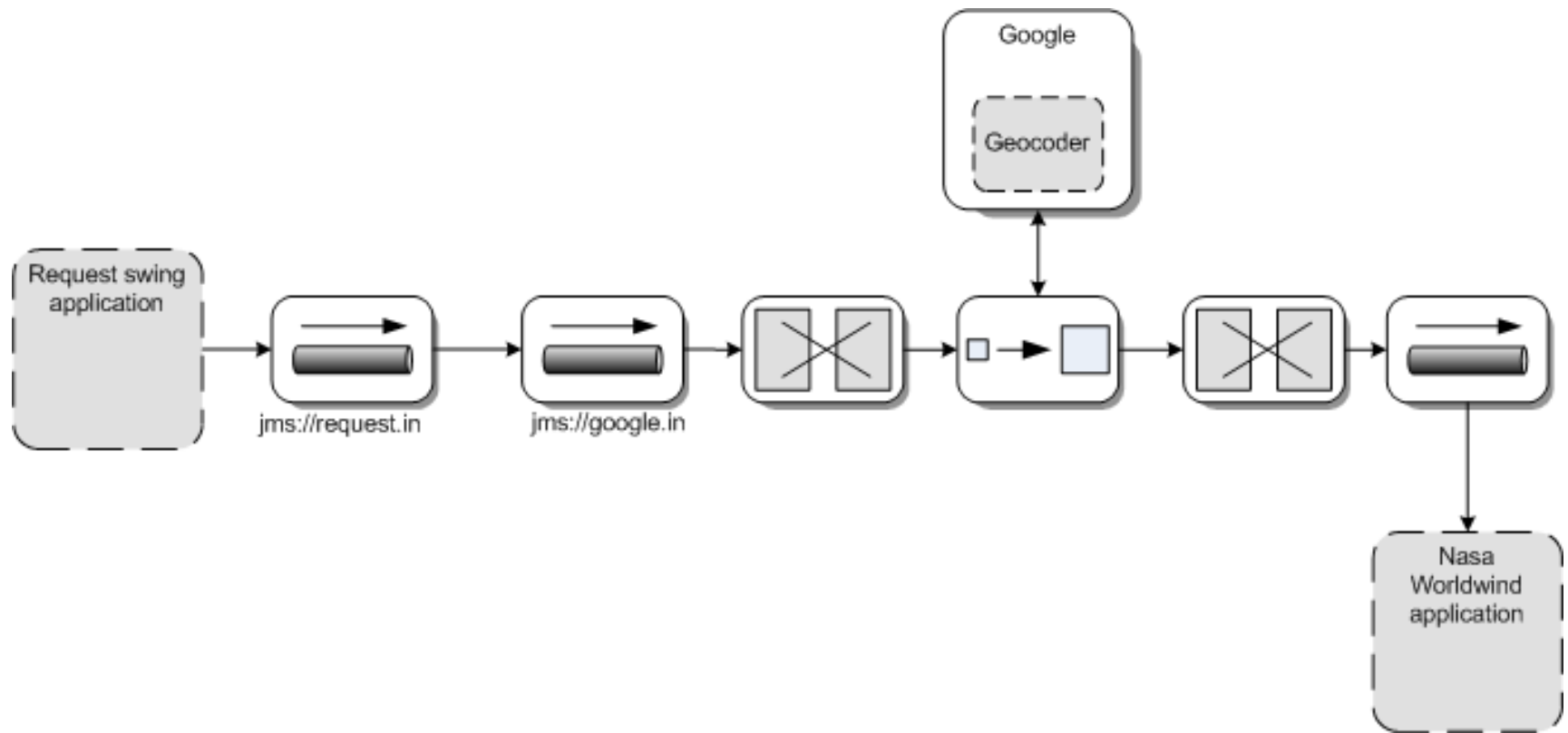
- ☘ Message originator doesn't have all the information
- ☘ Content-enricher adds extra data to the message

✈️ Content-enricher versus translator:

- ☘ Translator doesn't add extra data
- ☘ Content-enricher leaves the structure alone



Case part 2








Case part 3

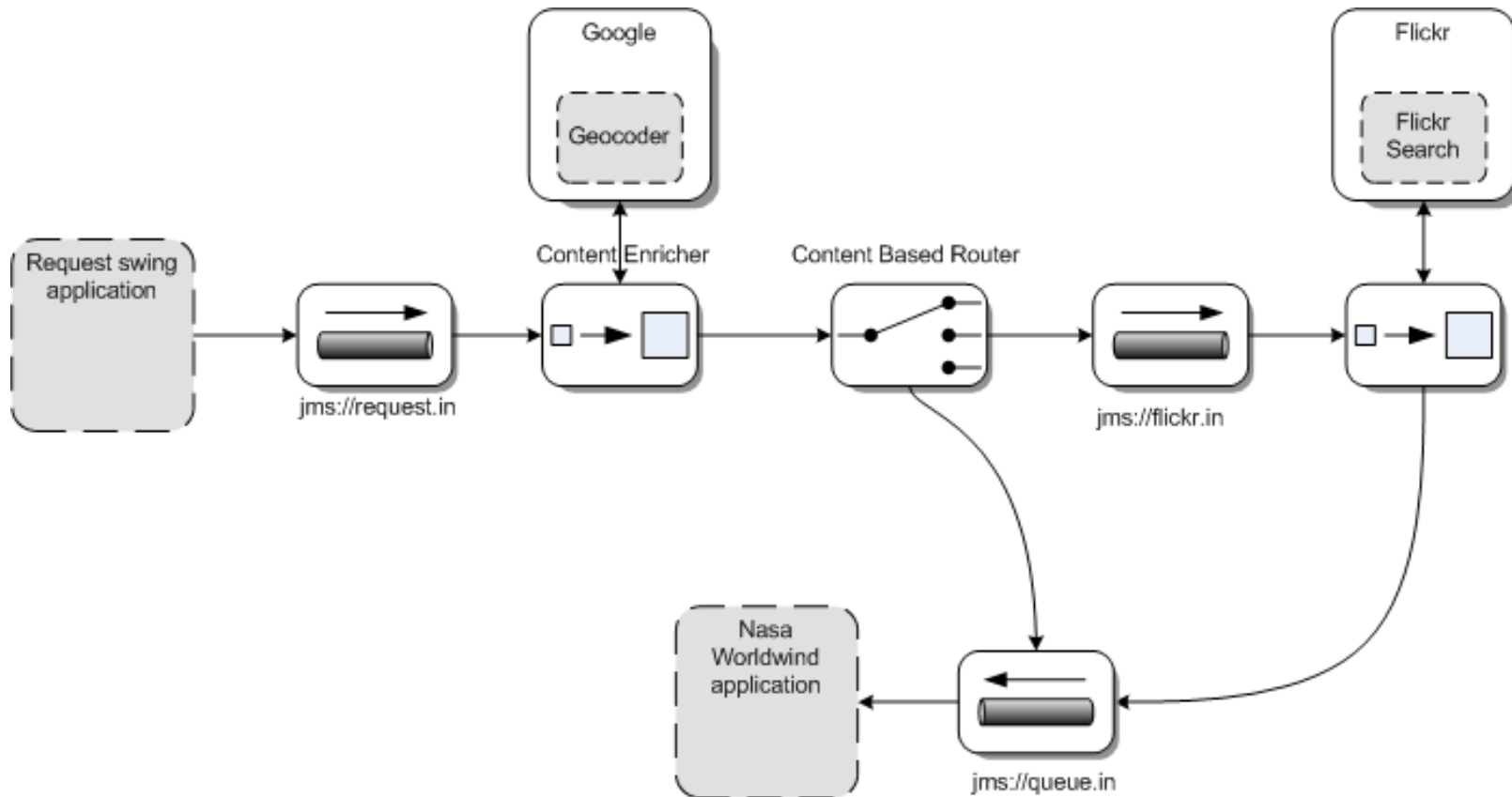
 Content-based routing pattern: 

-  Route the message based on content
-  Check for a certain property

 Message filter pattern: 

-  Make sure only interesting messages are received
-  Avoid overhead
-  Can be build into the broker (e.g. JMS selectors)

Case part 3





Case part 4

🚀 Wire-tap pattern:

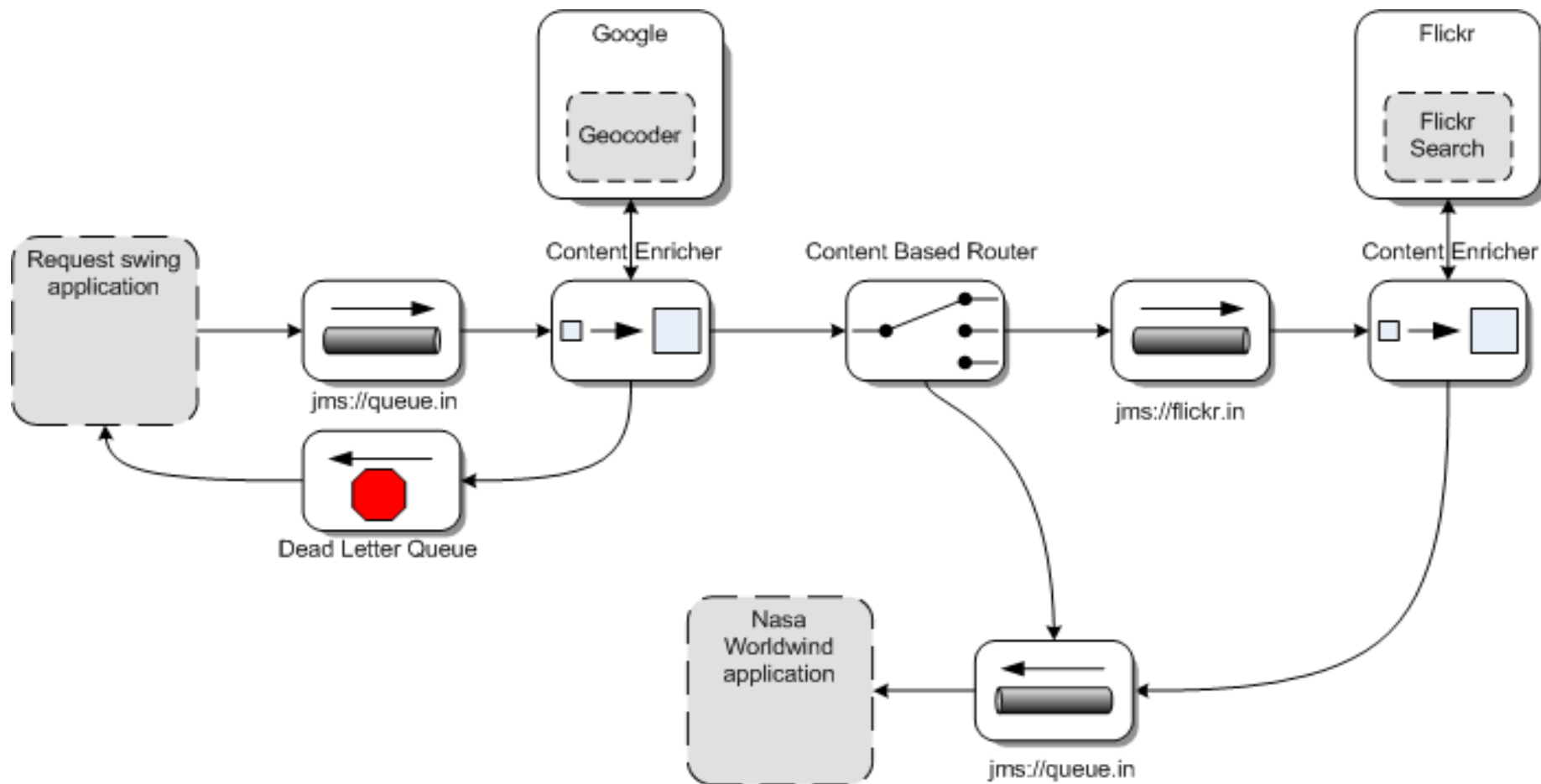
- ☘ Listen in on the messages that are sent
- ☘ Used for monitoring purposes
- ☘ Can be used for tracking and tracing

🚀 Dead Message Channel pattern:

- ☘ Handle messages that are incorrect
- ☘ Handle messages that can't be processed
- ☘ Channel can be monitored








EIP Case





Agenda

-  Introduction
-  The what and why of EIP
-  EIP in Action
-  **Summary**
-  Questions








Summary

- ✈ Enterprise integration still is hard
 - ☛ Patterns however make it easier..
- ✈ Patterns provide a visual and verbal language
- ✈ No vendor specific jargon
- ✈ Patterns provide best practices
 - ☛ But are no 'silver bullet'
- ✈ Currently available tools make EI easier:
 - ☛ Servicemix, Synapse, Mule
 - ☛ Websphere process server, Sonic ESB



Agenda

-  Introduction
-  The what and why of EIP
-  EIP in Action
-  Summary
-  Questions

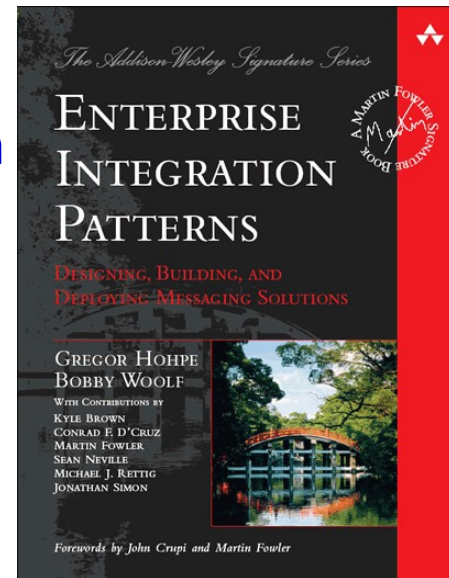


Questions

Any questions?

More info:

- Enterprise Integration Patterns, Hohpe and Woolf
- <http://www.enterpriseintegrationpatterns.com>
- <http://incubator.apache.org/servicemix>
- <http://activemq.apache.org/camel/>
- <http://mule.codehaus.org>



J-Spring 2007

Thank you



Types of enterprise integration

- ✈ 1970s: Batch Data Exchange
 - ☛ Export data into file; target application reads the data
- ✈ 1980s: Central Database
 - ☛ All applications access a common database
- ✈ 1990s: Remote Procedure Calls
 - ☛ One application directly calls another application
- ✈ Now: Messaging
 - ☛ Publish events to a queue / bus
 - ☛ Allow multiple subscribers to a message