Patterns in Mule

Antoine Borg
Senior Consultant
Ricston Ltd
Build composite patterns

Learn how to build composite patterns using Mule’s existing Enterprise Integration Patterns
Who Am I?

- Senior Consultant with Ricston Ltd
- Frequently lecturing courses for developers, architects and sysadmins on Mule 1.x and 2.x across Europe
- Consulting and assisting our customers with Mule deployments across USA, Europe and Australia
- Co-author of “Mule 2: A Developer’s Guide”
- Mule & SOA blog at http://blog.ricston.com
Overall Presentation Goal

Learn how to build upon the integration patterns available in Mule
Question

- The Enterprise Integration Patterns already implemented in Mule are well-documented.
- How can we take advantage of these to build larger architectural patterns?
Enterprise Integration Patterns

- EIP by Gregor Hohpe & Bobby Woolfe:
  - [http://www.enterpriseintegrationpatterns.com](http://www.enterpriseintegrationpatterns.com)
  - Standard vocabulary for integration solutions
- Mule is designed and meant for integration:
  - Makes sense to refer to these patterns
- Patterns are implemented in several ways:
  - Directly available
  - Implicitly available
  - Through a combination of both
Direct Patterns in Mule

- Message Routers:
  - Selective Consumer
  - Aggregator
  - Resequencer
  - Exception-based Routing
  - Recipient List
  - Wiretap
- Message Endpoint
Using Direct Routing Patterns

<wire-tap-router>
  <payload-type-filter
      expectedType="com.ricston.message"/>
  <outbound-endpoint address="vm://TappedQueue"/>
</wire-tap-router>

<exception-based-router>
  <outbound-endpoint
      address="axis:http://myserver/myWebService"/>
  <outbound-endpoint address="jms://OrderQueue"/>
  <outbound-endpoint
      address="file://./pendingOrders"/>
</exception-based-router>
Basic Routing Patterns

- Selective Consumer:
  - Do I want to route this message?

- Idempotency:
  - Do I want to route this message again?

- Aggregation and Resequencing:
  - Is this message part of a larger (logical) message?
  - Are these message parts in the right order?

- Recipient List:
  - Which services need to know about this message?
Indirect Patterns in Mule

**Transports:**
- Message Dispatcher
- Message Receiver
- Polling Consumer
- Transactional Clients
- Channel Adapter

**Routers:**
- Return Address
- Request-reply
- Point-to-Point
Using Indirect Patterns

<jms:activemq-connector name="salesServer"
    brokerURL="tcp://localhost:61616"/>

<file:connector name="paxDetails" streaming="false"
    moveToDirectory="./processed"
    outputPattern="${ORIGINALNAME}.XML" outputAppend="true"/>
Grouping patterns together:
- Larger building blocks
- Larger Patterns?

Mule Transport:
- Is a Channel Adapter
- Uses a Message Receiver & a Message Dispatcher
- Has dedicated Message Translators
- May have a Polling Consumer
- May have a Transactional Client
Summary

- Integration patterns are fundamental in Mule:
  - Direct or Indirect

- Use of the patterns allows for greater flexibility

- Use of composite patterns and architecture:
  - New integration patterns.
  - Mule lends itself to use and manipulation of patterns
Use Case – Capturing Errors

- Public Web Service receives message
- Message needs validation:
  - Specific rules in place
- Valid message causes DB lookup
- Result of lookup is response of web service
- If message cannot be validated, raise Exception:
  - Mule can catch this exception and handle it
  - How to route exception to caller using Mule?
Explained

Validator → Processor

Validator ← Processor
Solution – Plan for exception

- Mule’s traditional exception handling is inadequate here:
  - Designed for technical exceptions
  - Exception is not a programming exception
  - Exception is a business message:
    - Should be routed to a business consumer

- Exception should have its own routing:
  - Filter by Type?

- Exception should be part of overall architecture:
  - Output by service component **not** thrown.
  - Handled by Mule as a message **not** an exception
Solution

Diagram:
- Message
- Validator
- Processor
- Dummy Service
<service name="validatorService" >
  <inbound>
    <inbound-endpoint
        address="axis:http://myServer/myService"/>
  </inbound>
  <component class="com.ricston.Validator"/>
  <outbound>
    <filtering-router>
      <outbound-endpoint address="vm://forError"/>
      <payload-type-filter
          expectedType="com.ricston.UserException"/>
    </filtering-router>

    <outbound-pass-through-router>
      <outbound-endpoint address="vm://forLookup"/>
    </outbound-pass-through-router>
  </outbound>
</service>
Transacting with a Bank

- Internal customer generates transactions
- Transactions placed into file or files:
  - File format determined by bank
- File(s) sent to bank for processing: (Batch process)
- Restrictions:
  - Total value of transactions inside a single file; $ 99m
  - Transactions need to be stored until sent to bank
  - Can we store them in the file?
Explained Further
Solution - Asynchronicity

- Transactions placed on asynchronous queue
  - Bonus: Items can be de-queued if necessary
- Multiple patterns just before items sent to bank
  - 01 – Quartz used to de-queue items
  - 02 – Aggregator used to build batches (€ 99m batch)
  - 03 – Batches written out to file (s)
Solution
Aggregator Configuration

```xml
<service name="bankTransfer">
  <inbound>
    <quartz:inbound-endpoint
      cronExpression="0 45 16 ? * *">
      <quartz:endpoint-polling-job
        <quartz:job-endpoint address="jms://pending"
          timeout="4000"/>
      </quartz:endpoint-polling-job>
    </quartz:inbound-endpoint>
    <custom-correlation-aggregator-router
      class="com.ricston.TransactionAggregator"/>
  </inbound>
  <outbound>
    <pass-through-router>
      <file:outbound-endpoint path="./transactions"/>
    </pass-through-router>
  </outbound>
</service>
```
Summary

Integration patterns are fundamental in Mule:
- Direct or Indirect

Use of the patterns allows for greater flexibility

Use of composite patterns and architecture:
- New integration patterns.
- Mule lends itself to use and manipulation of patterns
Concluding statement

Mule lends itself to use and manipulation of patterns
Q&A

Do you have any questions?
Thanks for your attention!

http://www.ricston.com
http://www.mulesource.com
http://blog.ricston.com