Enterprise Integration with Spring Integration

Agim Emruli
SpringSource
8100
Presentation Goal

Learn how **Spring Integration** helps to solve common **Enterprise Integration** challenges
Integration Challenges
System Failures

The Electricity Failed. The Following Data Was Lost:

Fax Journal
Transmission Memory
Reception Memory
Intolerant Systems
Enterprise Integration Patterns
Pattern Catalog Overview

Message Construction

- Message
- Command Message
- Document Message
- Event Message

Message Transformation

- Request
- Reply
- Content Filter
- Content Enricher
- Message Translator
- Envelope Wrapper
- Normalizer
- Claim Check

Message Routing

- Message Router
- Content Based Router
- Composed Message
- Splitter
- Process Manager
- Aggregator
- Message Filter
- Recipient List
How does Spring help?
Inversion of Control
Focus on Business domain
Spring Powered Applications

- Testable
- Robust
- Maintainable
- Flexible

Application
Layered Architecture
Layered Architecture
Layered Architecture

Infrastructure

- Database
- Email
- Messaging
Layered Architecture

- Database
- Email
- Messaging

- Data Access
- Infrastructure
- Domain Objects
Layered Architecture

- Services
- Data Access
- Infrastructure
  - Database
  - Email
  - Messaging
- Domain Objects

Diagram depicting layers of architecture:
1. Services
2. Data Access
3. Infrastructure
4. Domain Objects

Diagram elements include:
- Database
- Email
- Messaging

Diagram also includes logos for Spring Source and Sun Microsystems.
Layered Architecture

AOP

Services

Data Access

Infrastructure

Domain Objects

- Database
- Email
- Messaging

JAZOON09
THE INTERNATIONAL CONFERENCE ON JAVA TECHNOLOGY
JUNE 22 - 25, 2009 ZURICH

spring
source

Sun microsystems
Layered Architecture
Layered Architecture

MVC

Spring Application
Layered Architecture

MVC

RMI

Spring Application
Layered Architecture

Webservice  MVC  RMI

Spring Application
Layered Architecture

Spring Application

Batch

Webservice

MVC

RMI
Layered Architecture

- Batch
- JMS
- Webservice
- MVC
- RMI

Spring Application
Event Driven Architecture

Framework

Event

Application
Spring JMS Support

```xml
<jms:listener-container transaction-manager="txManager">
  <jms:listener ref="orderService"
    method="order"
    destination="queue.orders"
    response-destination="queue.confirmation"/>
</j/ms:listener-container>
```
Spring JMS Support

```xml
<jms:listener-container transaction-manager="txManager">
  <jms:listener ref="orderService"
    method="order"
    destination="queue.orders"
    response-destination="queue.confirmation"/>
</jms:listener-container>
```

```java
public class OrderService {

    public OrderConfirmation order(Order o) { .. }

}
```
Introducing Spring Integration
Goals
Reuse Service Layer
Incremental Extension
Spring Integration Architecture
Spring Integration Architecture

Services
Spring Integration Architecture

Services

Webservice
Spring Integration Architecture

Services

Webservice

File
Spring Integration Architecture

Webservice

File

Services
Spring Integration Architecture

Webservice -> Transformer

Transformer -> File

Services
Spring Integration Architecture

Webservice

Transformer

Router

File

Services
Message Construction
Message Structure

Message Header
- Sequence Number
- Sequence Size
- Expiration Date
- Correlation Identifier

Message Body
- Command Message
- Document Message
- Event Message
public interface Message<T> {
    MessageHeaders getHeaders();
    T getPayload();
}
MessageHeaders headers = message.getHeaders();

String value =
headers.get("key", String.class);

Object id = headers.getId();

Long timestamp = headers.getTimestamp();

MessagePriority priority =
headers.getPriority();
Message<String> message =
MessageBuilder.withPayload("test")
    .setHeader("foo", 123)
    .setPriority(MessagePriority.HIGHEST)
    .build();

Message<String> copy =
MessageBuilder.fromMessage(message)
    .setHeader("foo", 456)
    .setHeaderIfAbsent("bar", 789)
    .build();
Channels And Endpoints
Message Channel

Message Endpoint

Channel

Message Endpoint
Direct Channels

Message Endpoint → Channel → Event-Driven Consumer
Direct Channels

<Message Endpoint> → <Channel> → <Event-Driven Consumer>

<channel id="sync-p2p" />
Queue Channel

Message Endpoint -> Channel -> Polling Consumer
Queue Channel

<channel id="async-p2p">
  <queue capacity="50" />
</channel>
Publish-Subscribe Channel

Message Endpoint → Publish Subscribe Channel → Event-Driven Consumer
Event-Driven Consumer

Event-Driven Consumer
Publish-Subscribe Channel

Message Endpoint

Publish Subscribe Channel

Event-Driven Consumer

Event-Driven Consumer

<publish-subscribe-channel id="pubsub" />
Priority Channel

Message Endpoint → Channel → Resequencer → Polling Consumer
Priority Channel

<channel id="priorityChannel">
  <priority-queue comparator="someComp" />
</channel>
Message Transformation
Message Translator

Channel → Message Translator → Channel
Message Translator

<transformer input-channel="input" output-channel="output" ref="transformer" method="transform"/>
Annotation Based Message Translator

```java
@MessageEndpoint
public class MessageTransformer{

    @Transformer(inputChannel="in", outputChannel="out")
    public LoanRequest transform(Loan loan){
        return ...
    }
}
```
Service Activator
Service Activator

Message Endpoint

Channel

Message Endpoint

Channel

Service Activator
Service Activator

```
<channel id="requests"/>
<channel id="quotes"/>

<service-activator input-channel="requests"
ref="loanBroker"
method="processRequest"
output-channel="quotes"/>

<beans:bean id="loanBroker"
class="example.LoanBroker"/>
```
Annotation Based Service Activator

@MessageEndpoint
class LoanBroker {

@ServiceActivator(inputChannel="x", outputChannel="y")
public LoanQuote processRequest(
    LoanRequest request) {
    LoanQuote quote ...
    return quote;
}
}
Polling and Transactions

```
<service-activator ref="loanBroker"
    method="processRequest"
    input-channel="requests"
    output-channel="quotes">
    <poller task-executor="pool1">
        <interval-trigger interval="5000"/>
        <transactional propagation="REQUIRES_NEW"/>
    </poller>
</service-activator>

<pool-executor id="pool1" max-size="25"/>
<beans:bean id="transactionManager" ... />
```
Content Based Router

Channel

Content Based Router

Standard Service Channel

VIP Service Channel
MethodInvokingRouter

<channel id="even"/>

<channel id="odd"/>

<router ref="parityResolver" input-channel="numbers"/>
MethodInvokingRouter

```xml
<channel id="even"/>

<channel id="odd"/>

<router ref="parityResolver" input-channel="numbers"/>
```

```java
@Router
public String getParity(int i) {
    return (i % 2 == 0) ? "even" : "odd";
}
```
PayloadtypeRouter

Channel -> Content Based Router

String Channel

Integer Channel
typeMap.put(String.class, stringChannel);

PayloadTypeRouter router = new PayloadTypeRouter();
router.setPayloadTypeChannelMap(typeMap);

router.handleMessage(new StringMessage("test"));
router.handleMessage(new GenericMessage(123));
RecipientListRouter

Channel

Recipient List

Channel

Channel
channels.add(channel1);
channels.add(channel2);

RecipientListRouter router = new RecipientListRouter();
router.setChannels(channels);
Message<String> message = new StringMessage("test");
router.handleMessage(message);
Splitter And Aggregator

Channel -> Splitter -> Aggregator -> Channel
Splitter And Aggregator

@Splitter
public List<OrderItem> splitOrder(PurchaseOrder order,
   @Header("customerId") String customerId) {
   // split the purchase order into order items...
}
Splitter And Aggregator

@Splitter
public List<OrderItem> splitOrder(PurchaseOrder order,
    @Header("customerId") String customerId) {
    // split the purchase order into order items...
}

@Aggregator
public PurchaseOrder aggregateOrder(List<OrderItem> items) {
    // aggregate the items into a single order object...
}
Channel Adapter
Channel Adapters

Inbound

External System → Channel Adapter → Channel

Message
Channel Adapters

**Inbound**

External System → Channel Adapter → Channel

**Outbound**

Channel → Channel Adapter → External System
File Adapter

```xml
<file:inbound-channel-adapter channel="filesIn"
   directory="${java.io.tmpdir}/test-input">
   <poller max-messages-per-poll="5">
       <cron-trigger expression="*/10 * * * * MON-FRI"/>
   </poller>
</file:inbound-channel-adapter>

<file:outbound-channel-adapter channel="filesOut"
   directory="${java.io.tmpdir}/test-output"/>
```
JMS Adapter

```xml
<jms:inbound-channel-adapter channel="input"
    connection-factory="connectionFactory"
    destination-name="sourceQueueName"/>

<jms:outbound-channel-adapter channel="output"
    destination="targetQueue"/>

<jms:inbound-gateway request-channel="inRequests"
    destination="inboundRequestQueue"/>

<jms:outbound-gateway request-channel="outRequests"
    reply-channel="replies" jms-queue="outQueue"/>
```
Method Invoking Adapter

```xml
<channel id="channel"/>

<inbound-channel-adapter channel="channel"
    ref="reader" method="read">
    <poller max-messages-per-poll="1">
        <interval-trigger interval="1000"/>
    </poller>
</inbound-channel-adapter>

<outbound-channel-adapter channel="channel"
    ref="writer" method="write"/>
```
Webservice Adapter

```xml
<ws:outbound-gateway uri="http://..."
marshaller="someMarshaller"
unmarshaller="someMarshaller"
request-channel="req" reply-channel="rep"/>
```

```xml
<ws:inbound-gateway request-channel="req"
reply-channel="rep"
marshaller="someMarshaller"
unmarshaller="someMarshaller" />
```
Other Adapters

> HTTP
> Mail
> RMI
> springsource.org/extensions
Q & A
Agim Emruli
agim.emruli@springsource.com
THANK YOU!
## Picture Credits

|-----------|-----------|-----------|-----------|

<table>
<thead>
<tr>
<th>Picture Credits</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="http://www.flickr.com/photos/global-jet/2125557004/" alt="Image" /></td>
</tr>
<tr>
<td><a href="http://www.flickr.com/photos/global-jet/2125557004/">Link</a></td>
</tr>
</tbody>
</table>