ENTERPRISE INTEGRATION PATTERNS WITH SPRING INTEGRATION

By

Kiran Hegde

https://www.linkedin.com/in/hegdekiran

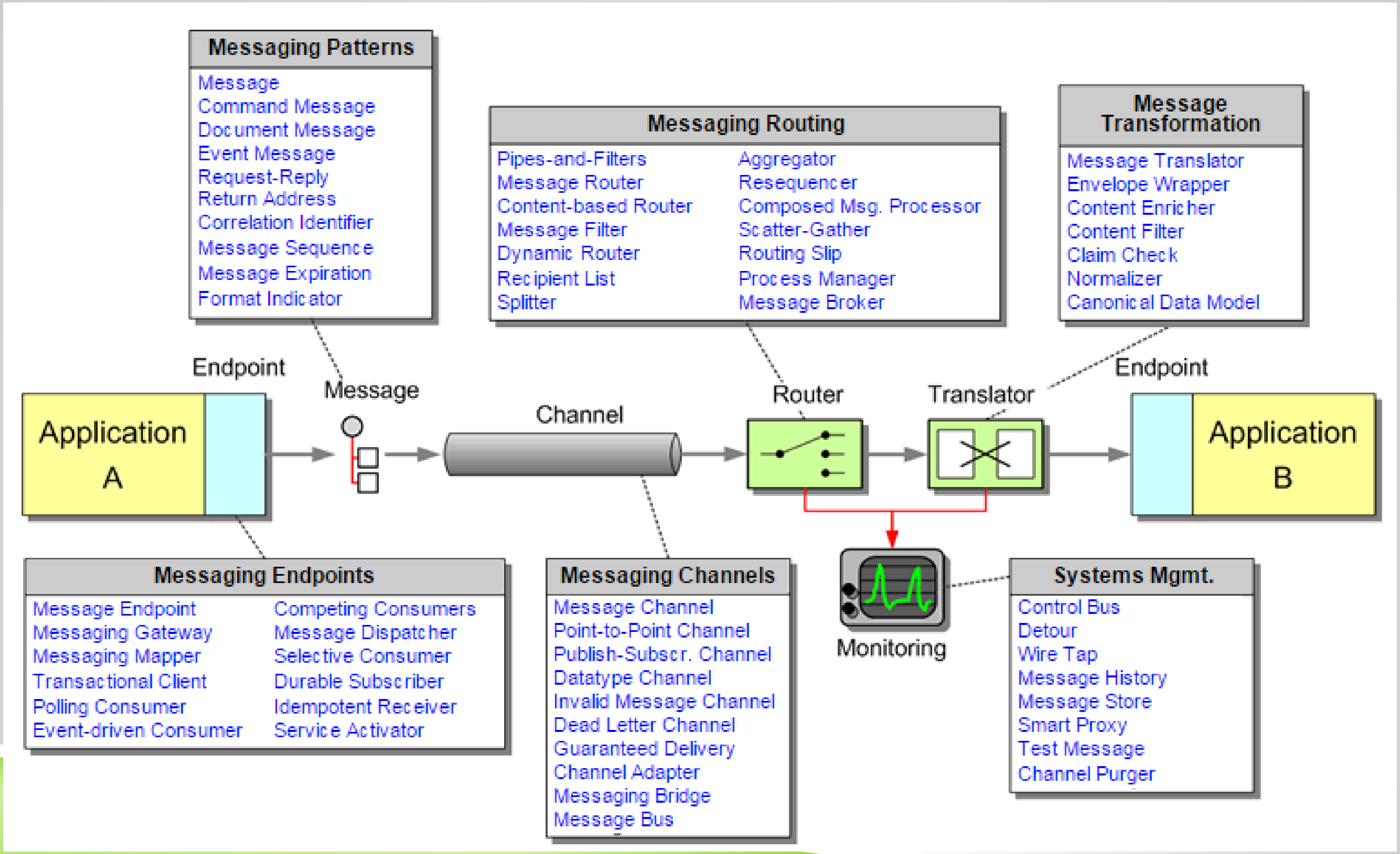
ENTERPRISE INTEGRATION!

- Enterprise interaction allowed organizations to both share data and make use of functionality provided by other systems.
- Although enterprise application integration can take many different forms
  - **From** extract-transform-load jobs run overnight
  - **To** all-encompassing SOA strategies
- All approaches leverage one of four well-known integration styles
  - File-based integration
  - Shared-database integration
  - Remote Procedure Calls
  - Message-based integration

4 WELL KNOWN ENTERPRISE INTEGRATION STYLES

- **File-based integration –**
  - Simple interoperable but deals with file read/write complexity
- **Shared-database integration**
  - Atomic & Consistent but doesn’t solve invoking functionality in remote application
- **Remote Procedure Calls**
  - Method invocation, return value is serialized via Stubs, Proxies & Marshalling
  - Therefore serializing arguments and return values harms interoperability
- **Message-based integration**
  - Messaging is an integration style based on exchanging encapsulated data packets (messages) between components (endpoints) through connections (channels). As described at www.enterpriseintegrationpatterns.com,
  - The packets should be small, and they should be shared frequently, reliably, immediately, and asynchronously.
- The middleware that can orchestrate reliable communication between these disparate endpoints, are typically called as **Enterprise Service Bus (ESB).**

ENTERPRISE INTEGRATION PATTERNS?



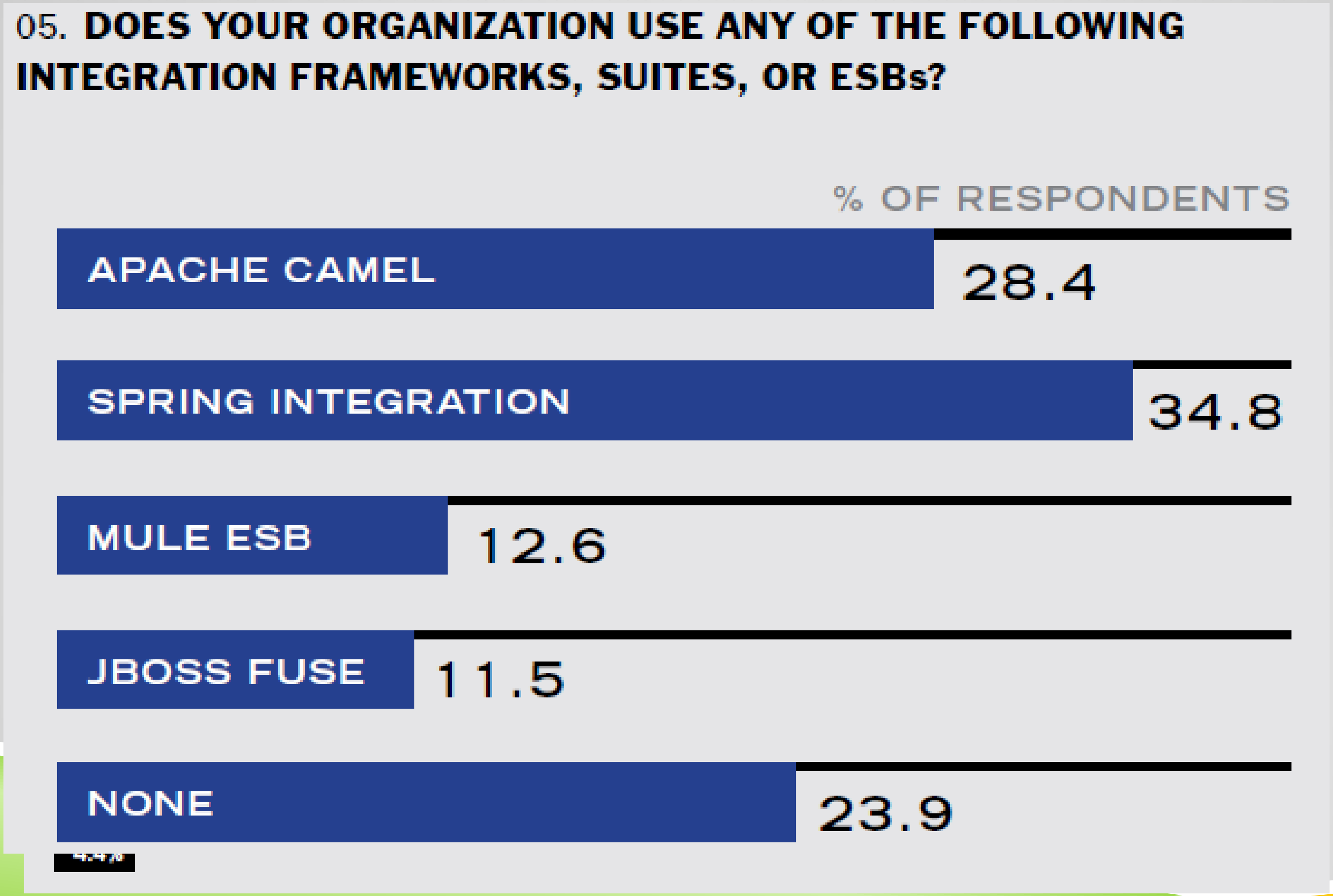
WHAT PRODUCTS USE ENTERPRISE INTEGRATION PATTERNS?

- Open source **ESB's** like Mule ESB, JBoss Fuse, Open ESB, WSo2, Spring Integration, or Talend ESB
- Message Brokers like ActiveMQ, Apache Kafka, or RabbitMQ
- **EAI and SOA platforms,** such as IBM WebSphere MQ, TIBCO, Vitria, Oracle ServiceBus, WebMethods (now Software AG), Microsoft BizTalk, or Fiorano.

HOW IS AN INTEGRATION FRAMEWORK USEFUL?

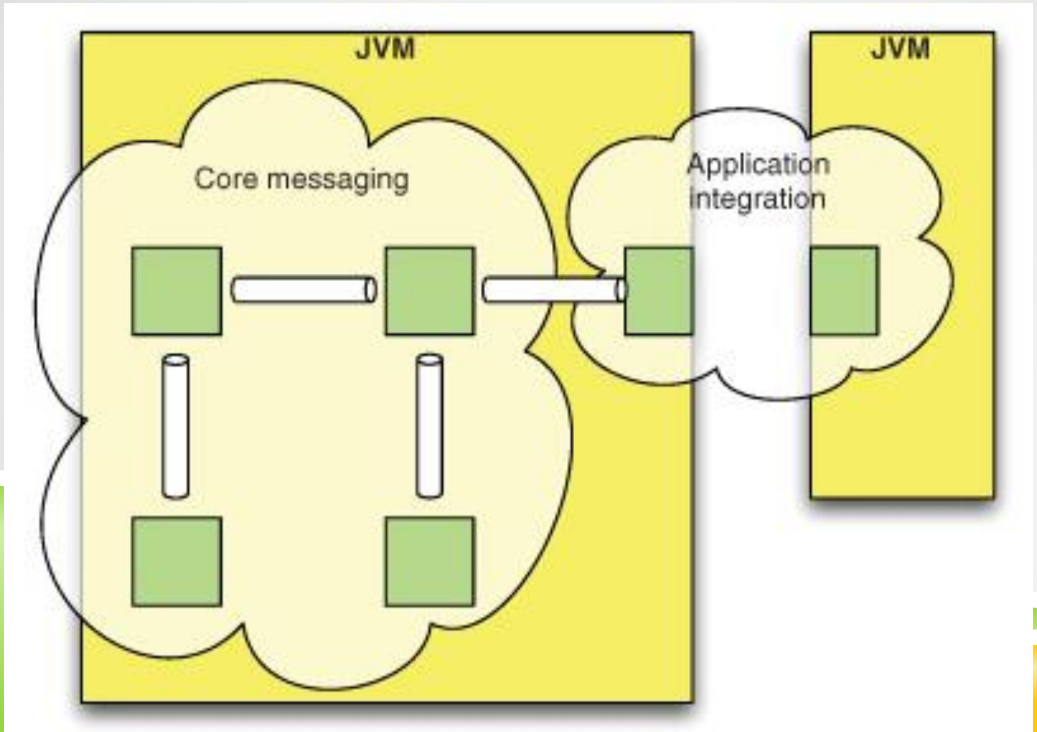
- We live in an event-driven world. Throughout each day, we’re continuously bombarded by phone calls, emails, and instant messages.
- Interestingly I heard this somewhere –
  - This Talk is also Messaging, perhaps we could see as
    - Publish – Subscribe,
    - One producer – Many Consumers
    - May give immediate a.k.a Synchronous feedback
    - May take this message – Transform a bit an give it to teams/friends
    - Or leave an Aynchronous feedback – perhaps that this talk was good – jk!! ☺
- Real World example –
  - Online travel booking application

DZONE – 2015 GUIDE TO ENTERPRISE INTEGRATION


05. **DOES YOUR ORGANIZATION USE ANY OF THE FOLLOWING INTEGRATION FRAMEWORKS, SUITES, OR ESBs?**

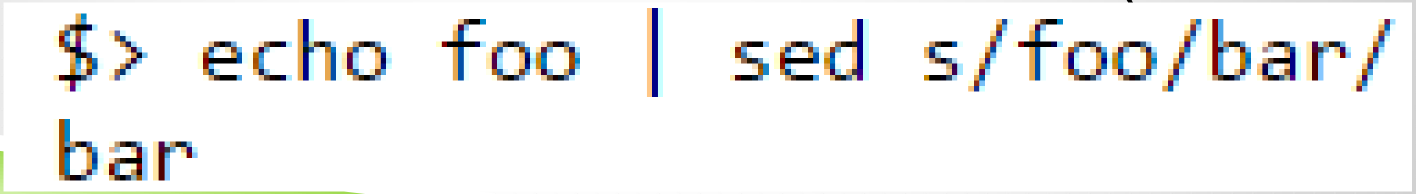
SPRING INTEGRATION OVERVIEW

- It enables lightweight messaging *within* Spring-based applications
- Supports integration with external systems via declarative adapters.
- Those adapters provide a higher-level of abstraction over Spring's support for remoting, messaging, and scheduling.
- So essentially **lightweight intra-application messaging**
- **And flexible inter-application integration**



PIPES & FILTERS

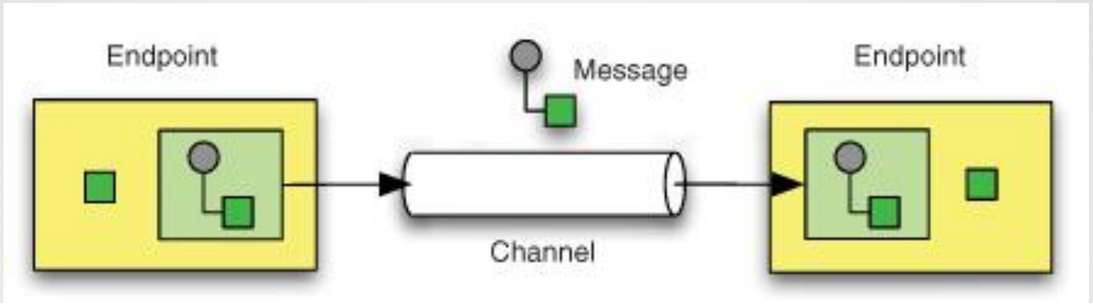
- Anyone familiar with a UNIX-based operating system can appreciate the pipes-and-filters style: it provides the foundation of such operating systems.
- Consider a basic example:
- You can see that it's literally the pipe symbol being used to connect two commands (the filters)

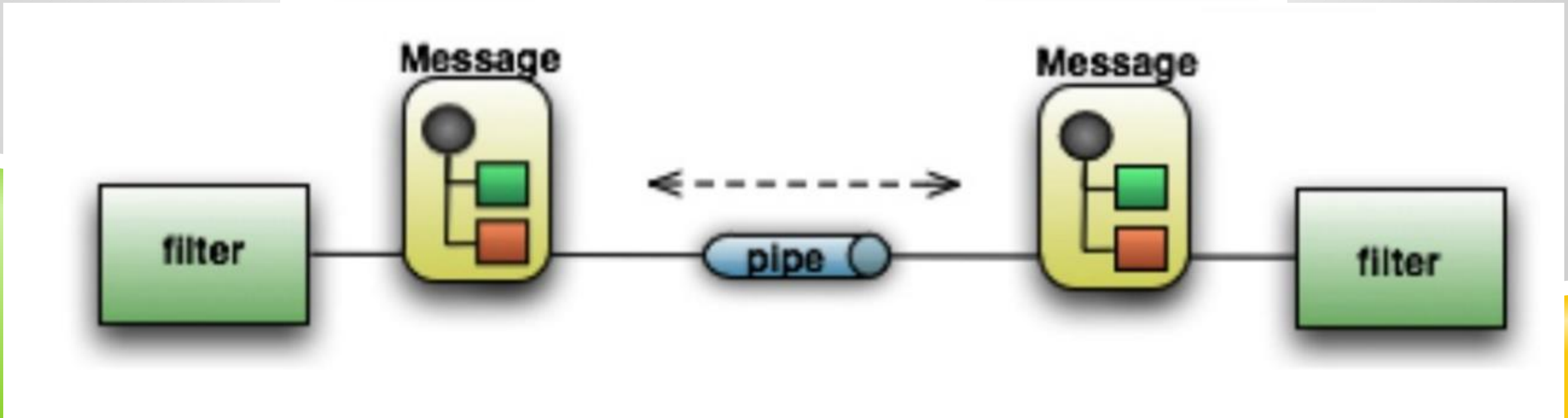


```
$> echo foo | sed s/foo/bar/
bar
```

THE CORE OF SPRING INTEGRATION IS

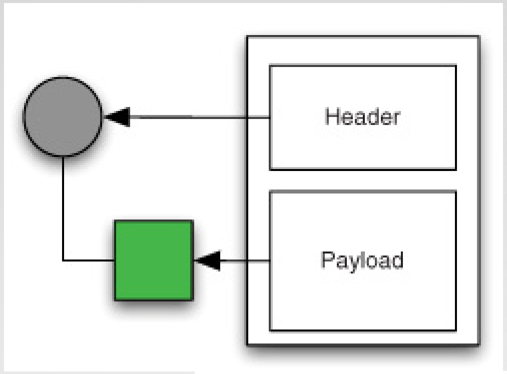
- **End Points** (Filters - similar to a **Processor**)
  - connected through
- **Channels** (Pipes)
  - exchanging
- **Messages**



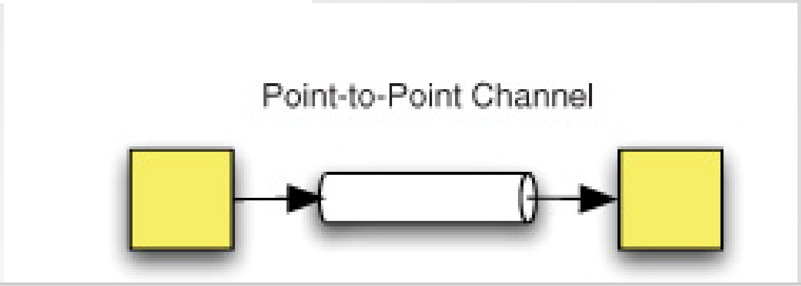


MESSAGES & CHANNELS

- Messages

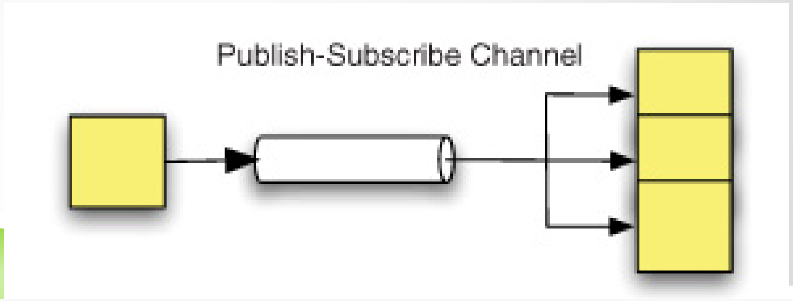


- Channels
  - Point-to-Point Channel



```
<int:channel id="exampleChannel"/>
```

  - Publish-Subscribe Channel



```
<int:publish-subscribe-channel id="exampleChannel"/>
```

DEMO

DIFFERENT TYPES OF CHANNELS

- Datatype Channel Configuration

```
<int:channel id="numberChannel" datatype="java.lang.Number"/>
```

- Priority Channel Configuration

```
<int:channel id="priorityChannel" datatype="example.Widget">
    <int:priority-queue comparator="widgetComparator"
                        capacity="10"/>
</int:channel>
```

- Scoped Channel Configuration

  -
    ```
    <int:channel id="threadLocalChannel" scope="thread"/>
    ```

- Channel Interceptor & Wire Tap Configuration

  -
    ```
    <int:channel id="inputChannel">
      <int:interceptors>
        <int:wire-tap channel="logger"/>
      </int:interceptors>
    </int:channel>
    ```

EVENT DRIVEN ARCHITECTURE & LOOSE COUPLING

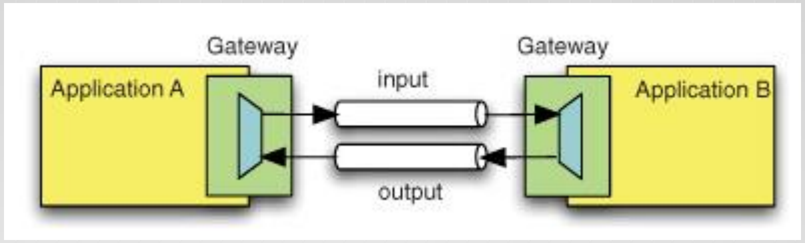
- Achieving an appropriate degree of **loose coupling** allows you to spend more time adding new features and delivering business value.
- Event-driven architecture (EDA) is an architectural pattern in which complex applications are broken down into a set of components or services that interact via events.
- Where events are communicated via channels that can act as buffers in periods of high throughput, such a system can be described as having a staged event-driven architecture (SEDA).
- The question of whether an application built around the Spring Integration framework is inherently an EDA or SEDA application is open to debate. Certainly Spring Integration provides the building blocks to create both EDA and SEDA applications.

MESSAGE ENDPOINTS

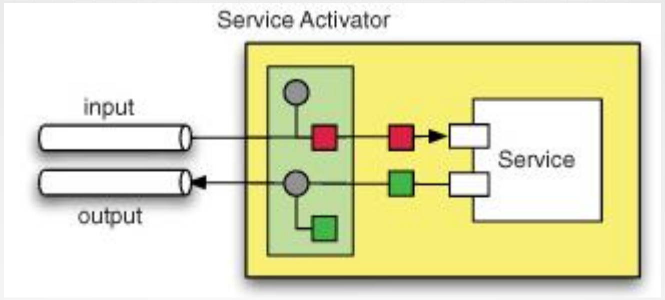
- Receivers or senders
  - Endpoints can either receive messages from the channel or put messages on the channel for further processing.
- Polling endpoints or event-driven endpoints
  - Endpoints can either pull messages from the channel or can subscribe to it. Whenever a message is available, a registered callback method is called.
- Unidirectional or bidirectional endpoints
  - Unidirectional endpoints send off or receive messages, but do not expect or receive any acknowledgement. Spring Integration provides channel adapters for such types of interactions. Bidirectional adapters can send, receive, and acknowledge messages. Spring Integration provides gateways that are synonymous with synchronous two-way communication.
- Inbound or outbound endpoints
  - Outbound endpoints interact with external systems such as social networks, mail servers, enterprise JMS, and others, whereas inbound endpoints listen for events from outside entities such as mail connector, FTP connector, and so on.

MESSAGE PROCESSING USING SPRING INTEGRATION ENDPOINTS

- Messaging Gateway



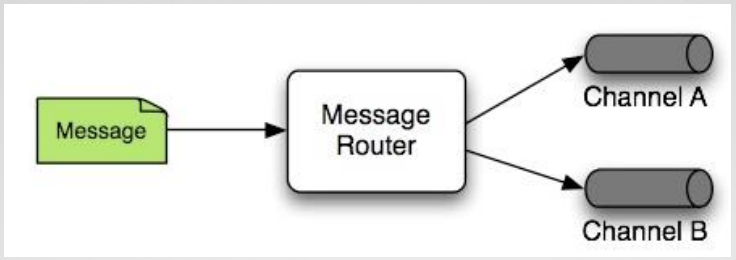
- Service Activator



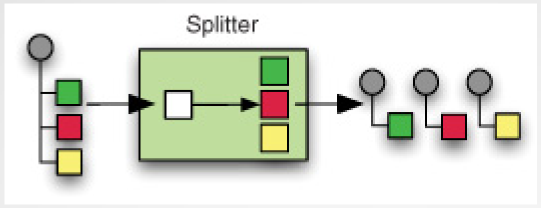
DEMO

FLOW COMPONENTS

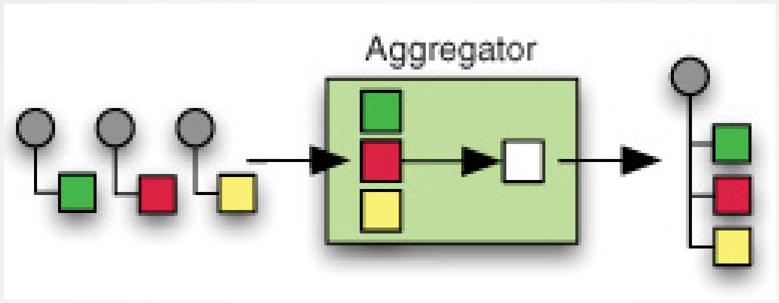
- Router



- Splitter



- Aggregator



ROUTER

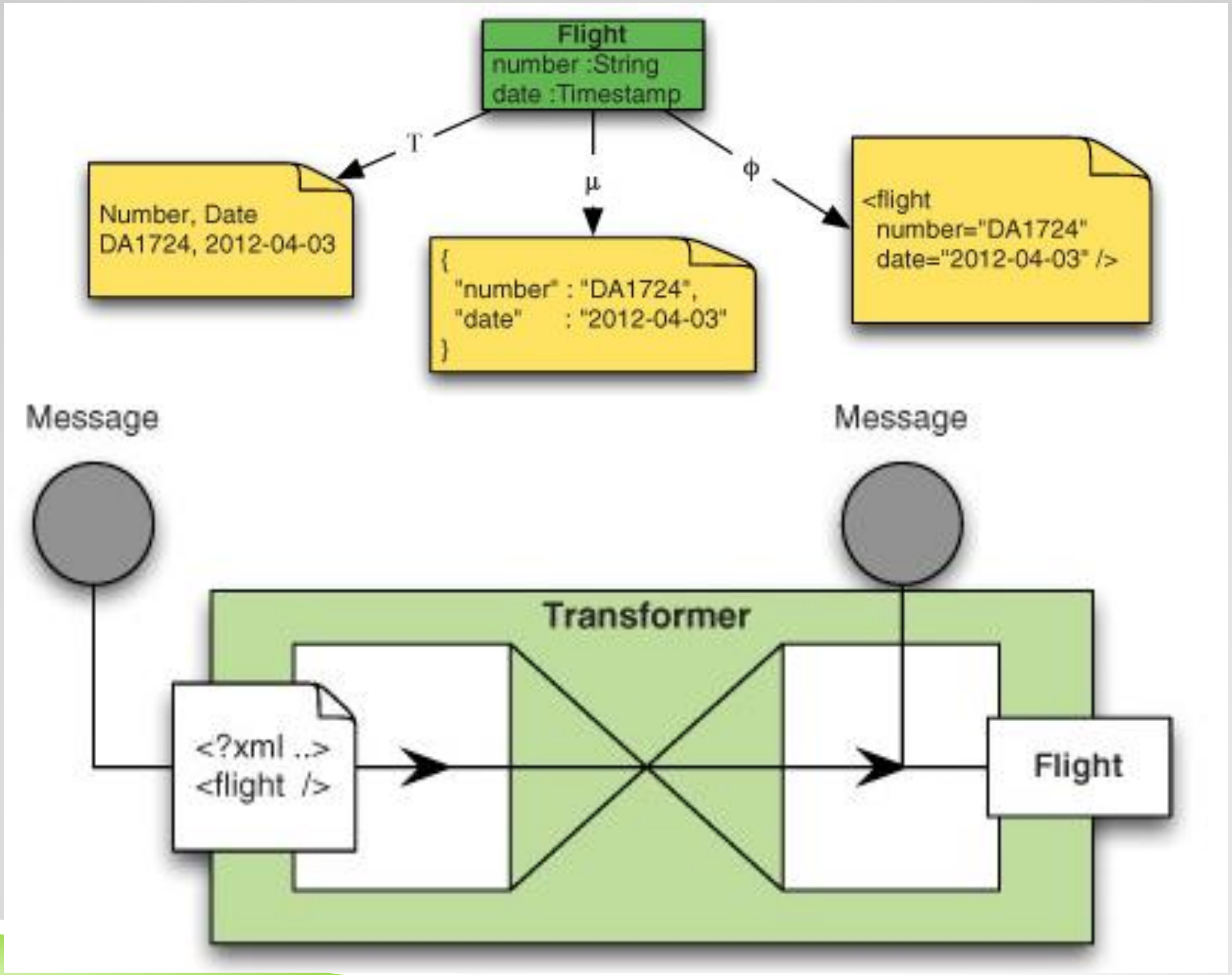
- Payload type Router

```
<int:payload-type-router input-channel="transformedChannel">
  <int:mapping type="com.cpandey.siexample.pojo.SoFeed" channel="jdbcChannel" />
  <int:mapping type="java.lang.String" channel="jmsChannel" />
  <int:mapping type="org.springframework.messaging.Message"
channel="mailChannel" />
</int:payload-type-router>
```

- Header value Router

```
<int:header-value-router
  input-channel="feedsChannel"
  header-name="feedtype">
  <int:mapping value="java" channel="javachannel" />
  <int:mapping value="spring" channel="springchannel" />
</int:header-value-router>
```

DOMAIN DRIVEN TRANSFORMATION



Message transformers are implementations of the **Enterprise Integration Pattern (EIP)** named **Message Translator**,

HTTP & OBJECT TO JSON TRANSFORMER

- Demo

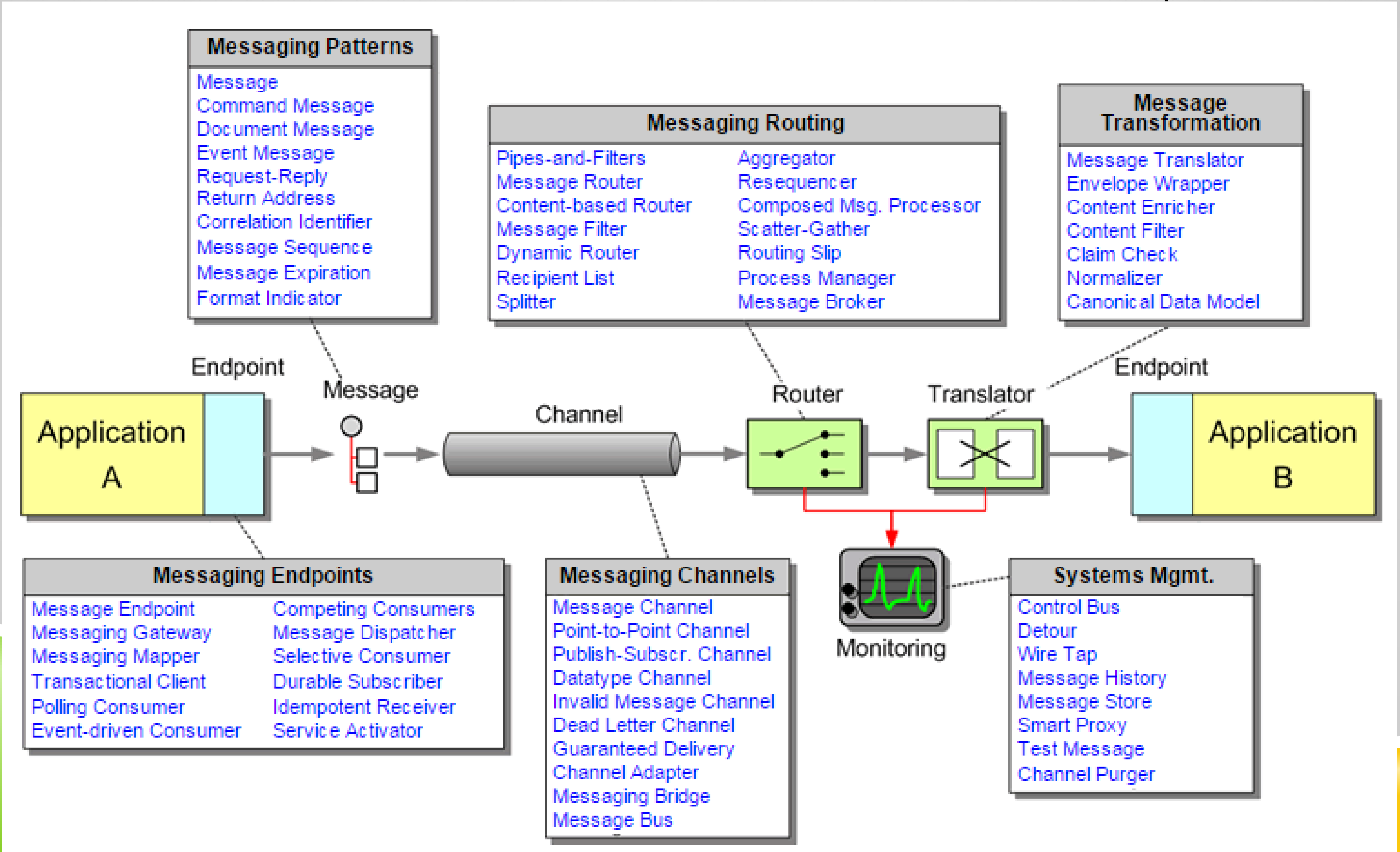
SPRING BATCH + SPRING INTEGRATION WITH RABBIT MQ

- Demo

ENTERPRISE INTEGRATION PATTERNS!!

- Reconciling the EIP slide shared earlier in presentation –

we should not have correlation to one each of the broader level component.



REFERENCES

- Views personal and not of employer's
- Sharing what I learnt, is not an endorsement!
- Content referenced from few books, blogs & online resources
- Few of them below
- http://www.enterpriseintegrationpatterns.com/
- http://projects.spring.io/spring-integration/
- Spring Integration in Action by Mark Fisher, Jonas Partner, Marius Bogoevici, and Iwein Fuld
- Spring Integration Essentials By Chandan Pandey
- https://github.com/spring-projects/spring-integration

THANK YOU!!

- Questions??