

# Spring Integration

Connecting Enterprise Applications

Jonas Partner

SpringSource consultant and Spring Integration contributor

- 
- **Enterprise Application Integration**
  - Enterprise Integration Patterns
  - Spring Integration
  - Comparing Spring Integration to others
  - Summary and questions

# Enterprise Application Integration

---



*“Enterprise Application Integration (EAI) is defined as the uses of software and computer systems architectural principles to integrate a set of enterprise computer applications”*

*Wikipedia*

**Messaging is the most common approach to EAI  
and the basis for Spring Integration**

# What is Messaging?

---

*How can multiple applications work together?...*

**...without being in each others way.**

Waiter helps customer and cook to collaborate.



# Characteristics of Messaging

---



## **Transport**

The waiter takes an order and moves it to the barista

## **Asynchronous**

Different actors do different things in parallel

## **Translation**

menu item => number => recipe

## **Routing**

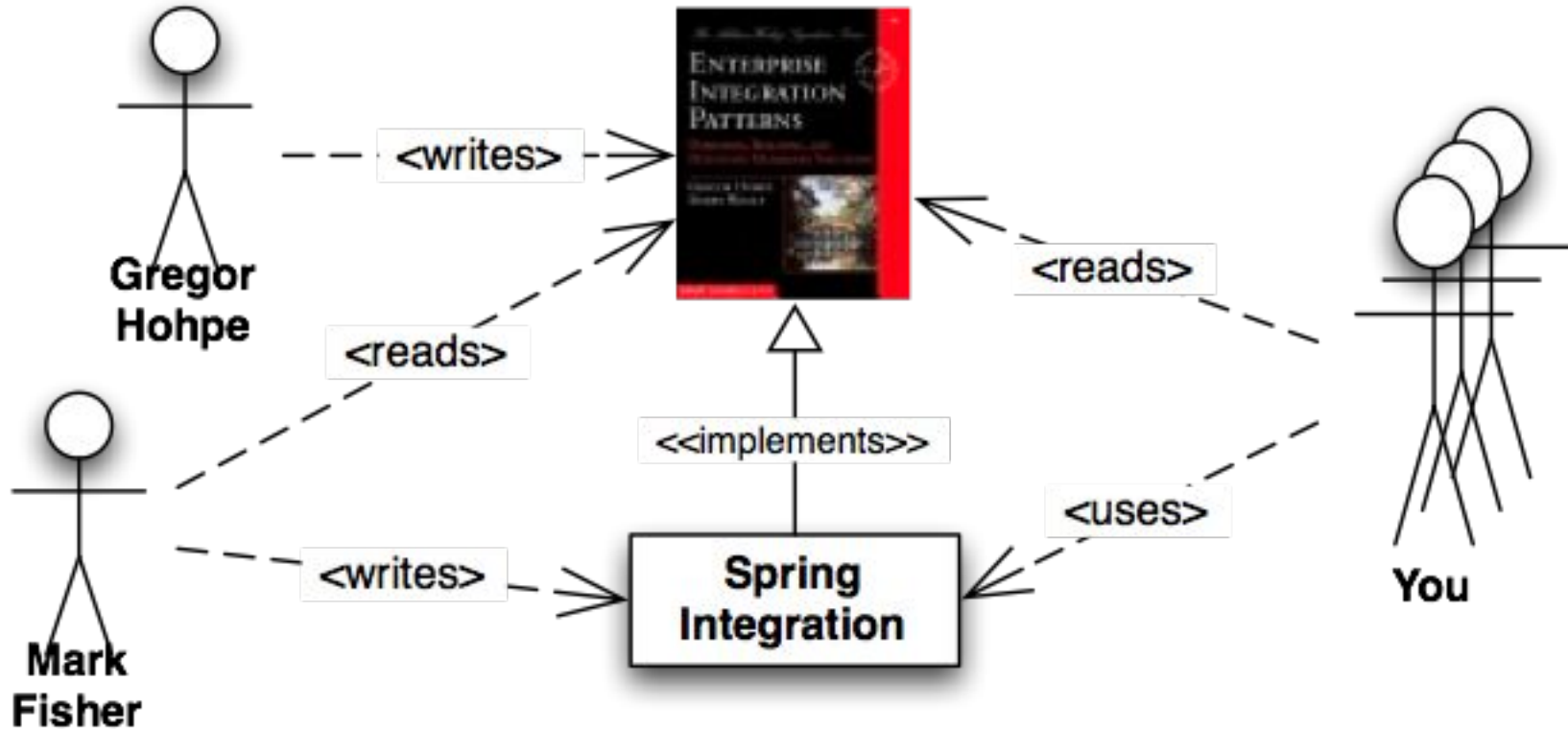
Orders arrive back at the proper table

# Topics in this session

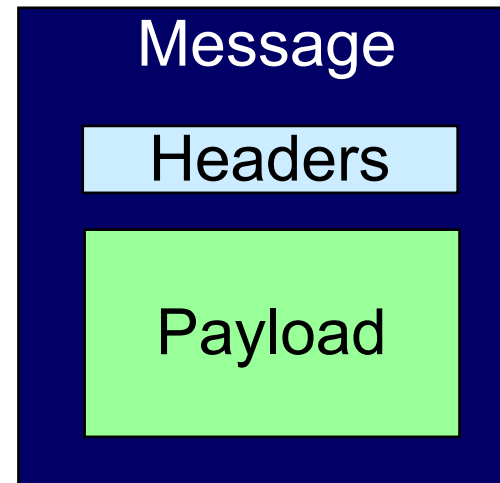


- Enterprise Application Integration
- **Enterprise Integration Patterns**
- Spring Integration
- Comparing Spring Integration to others
- Summary and questions

# Spring Integration and EIP

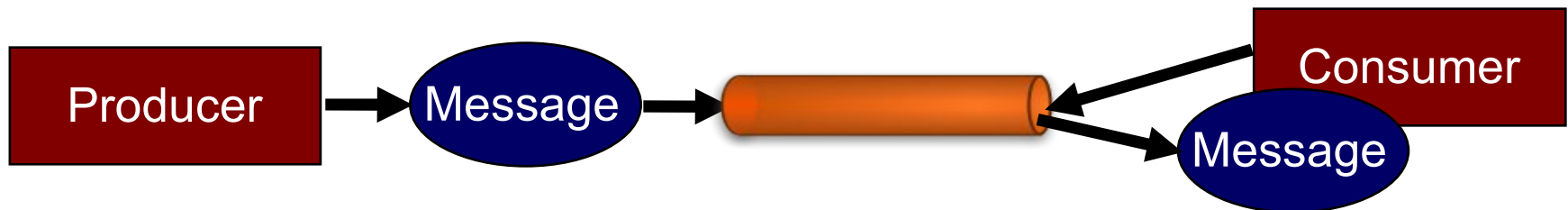


- A generic package for data (the Message payload) that can be transported via channels
- Message Headers provide information to other components that consume from channels
  - Sequence Number
  - Sequence Size
  - Expiration Date
  - Correlation Identifier
  - Return Address
  - Transport Info



# Message Channel

- Decouples producers from consumers
- Supports Point-to-Point or Publish/Subscribe
- Enforces data type consistency



# Channel Adapter

- Connect a source to the messaging system so it can send to a Message Channel

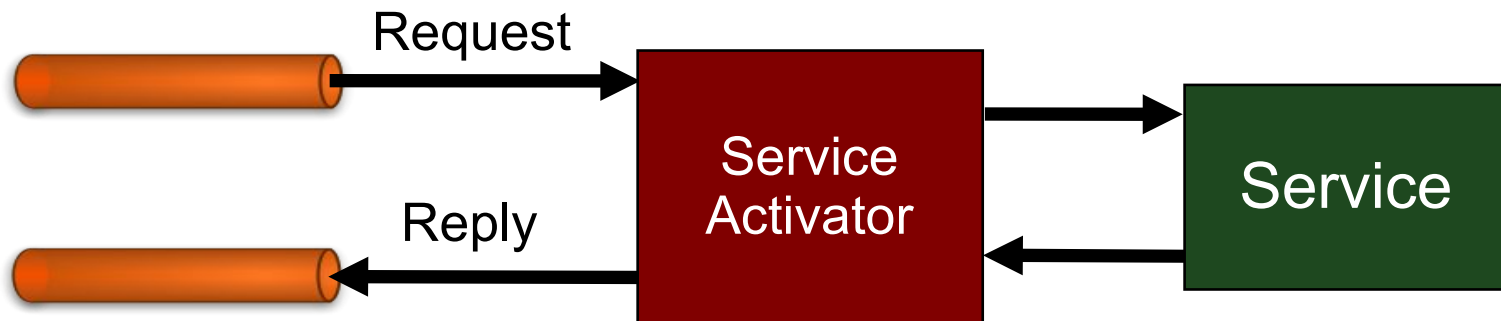


- Connect a target to the messaging system so it can receive from a Message Channel



# Service Activator

- A Message Endpoint that invokes a service
- Supports multiple communication styles
  - one-way and request-reply
  - synchronous and asynchronous
- The service is unaware of the messaging system



# Message Translator

- Payload Transformer
  - converts the type or format of a Message
- Header Transformer
  - add-to or remove-from the MessageHeaders



# Topics in this session

---



- Enterprise Application Integration
- Enterprise Integration Patterns
- **Spring Integration**
- Comparing Spring Integration to others
- Summary and questions

# Objectives (1/2)



## **Ease of adoption**

Programming model that will be familiar to existing Spring users

Allows easy addition of a messaging to existing applications

## **Lightweight**

Does not require install of heavyweight software

Quick to start and stop as part of a Spring application context

# Objectives (2/2)

---



## **Non intrusive**

Framework decouples components from messaging infrastructure

## **Testable**

Simple unit testing of POJO components

Quick and simple integration testing

## **First class Spring support**

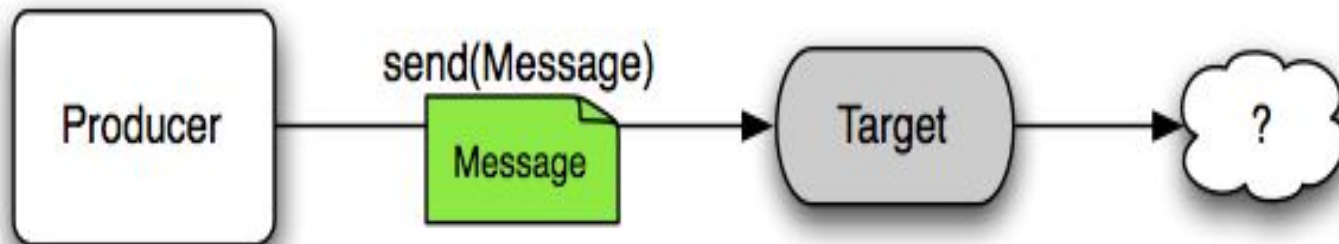
Exposes full power of Spring

Integrates seamlessly with Spring Security and Spring Web Services

# Message Target

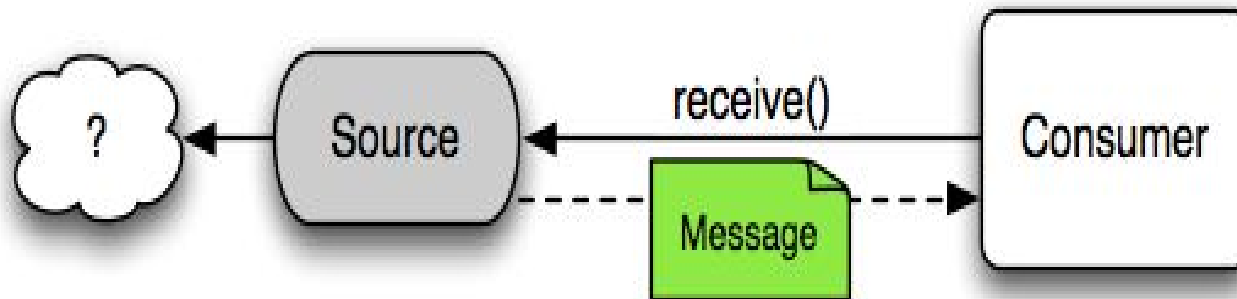
- Interface for any component to which Messages can be sent

```
public interface MessageTarget {  
    boolean send(Message message);  
}
```



- Interface for components from which Polling Consumers can receive Messages

```
public interface PollableSource<T> extends MessageSource {  
    Message<T> receive();  
}
```



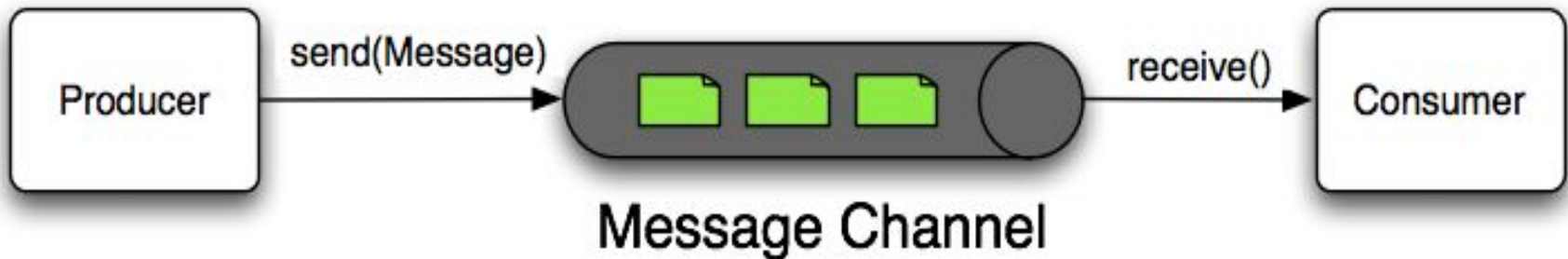
- Interface for components that send Messages to Event-Driven Consumers

```
public interface SubscribableSource extends MessageSource {  
    boolean subscribe(MessageTarget target);  
    boolean unsubscribe(MessageTarget target);  
}
```

# Message Channel

```
public interface MessageChannel extends
    BlockingSource, BlockingTarget {

    boolean send(Message message);
    boolean send(Message message, long timeout);
    Message receive();
    Message receive(long timeout);
}
```



# Message Handler

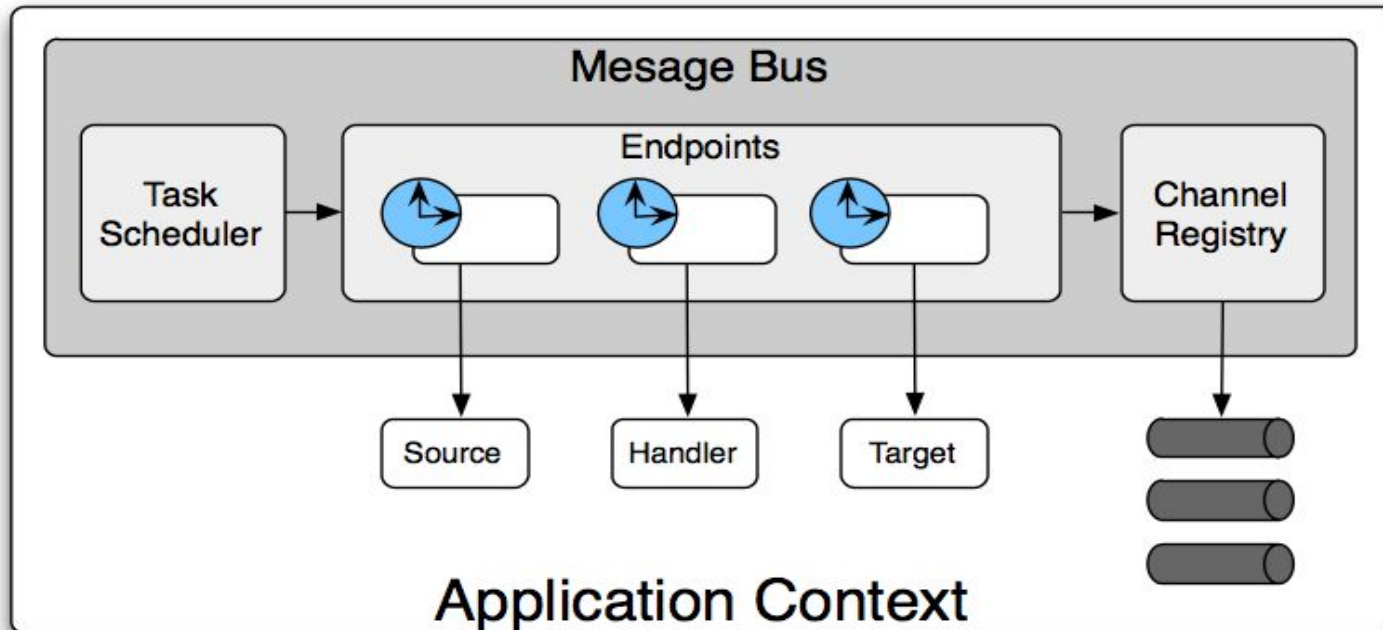
```
public interface MessageHandler {  
    Message handle(Message message);  
}
```



- Invoke application services from the messaging system
- adapters enable invocation of any method on a POJO
  - configurable via annotations or XML

# Message Bus

- Auto-detects Spring Integration components within the Application Context and registers Message Endpoints
- Schedules polling and dispatching for Message Endpoints



# Service Activator



```
<message-bus/>
```

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

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

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

```
<beans:bean id="loanBroker" class="example.LoanBroker"/>
```

# Annotation-Based Configuration



```
public class LoanBroker {
```

```
    @Handler
```

```
    public Quote generateQuote(LoanRequest request) {
```

```
        Quote quote = ...
```

```
        return quote;
```

```
    }
```

```
}
```

In the XML:

```
<annotation-driven/>
```

# Polling and Transactions



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

using @Handler  
instead of the  
"method" attribute

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

# Securing Endpoints



```
<service-activator id="endpoint" input-channel="input"
                    ref="testHandler">
  <interceptors>
    <si-security:endpoint-security-policy
      access="ROLE_ADMIN" />
  </interceptors>
</service-activator>
```

# Channel Adapters



```
<channel-adapter id="requests" source="fileIn"/>
```

```
<channel-adapter id="quotes" target="jmsOut"/>
```

```
<file-source id="fileIn" directory="/tmp/samples/input"/>
```

```
<jms-target id="jmsOut" destination="demoQueue"/>
```

```
<service-activator ref="loanBroker"  
    input-channel="requests"  
    output-channel="quotes">
```

```
    <poller period="5000"/>
```

```
</service-activator>
```

# Web Service Activator



```
<ws-service-activator id="temperatureConverter"  
    input-channel="fahrenheitChannel"  
    output-channel="celsiusChannel"  
    uri="http://www.w3schools.com/webservices/tempconvert.asmx">
```

# Other Types of Handlers



@Splitter

```
public List<Drink> split(DrinkOrder order) {  
    return order.getDrinks();  
}
```

@Router

```
public String resolveDrinkChannel(Drink drink) {  
    return (drink.isIced()) ? "coldDrinks" : "hotDrinks";  
}
```

@Aggregator

```
public DrinkOrder createOrder(List<Drink> drinks) {  
    DrinkOrder order = new DrinkOrder();  
    for (Drink drink : drinks) { order.addDrink(drink); }  
    return order;  
}
```

# Router Example



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

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

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

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

# Channels

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



```
<direct-channel id="orderedNotification"
```



```
<message-bus/>
```

# Hello world



```
<message-bus/>
```

```
<service-activator input-channel="inputChannel"  
    default-output-channel="outputChannel"  
    handler-ref="helloService"  
    handler-method="sayHello"/>
```

```
<beans:bean id="helloService" class="...HelloService"/>
```

```
public class HelloService {  
    public String sayHello(String name) {  
        return "Hello " + name;  
    }  
}
```

# Hello world



```
inputChannel =  
    channelRegistry.lookupChannel("inputChannel");  
outputChannel =  
    channelRegistry.lookupChannel("outputChannel");  
  
inputChannel.send(new StringMessage("World"));  
System.out.println(  
    outputChannel.receive().getPayload());
```

```
$ java HelloWorldDemo  
Hello World
```

# Topics in this session



- Enterprise Application Integration
- Enterprise Integration Patterns
- Spring Integration
- **Comparing Spring Integration to others**
- Summary and questions

# What's different about Spring Integration?

---



- Can be used from within an existing application.
- Lightweight (like any Spring application):
  - run from JUnit test
  - run within webapp
- Focussed on integration, not on ESB

- Full blown ESB
- It's an application, not a framework
  - You need to install it
  - You need to run it
- Typically a lot heavier
- Focus on the deployment architecture (SOA) not the actual integration.

- Integrates very well with Spring
- Lots of integration options:
  - REST, SOAP
  - JMS
- Embeddable
- Distribution
  - 32Mb (zipped)
  - 2Mb (zipped) jar only (for embedding)

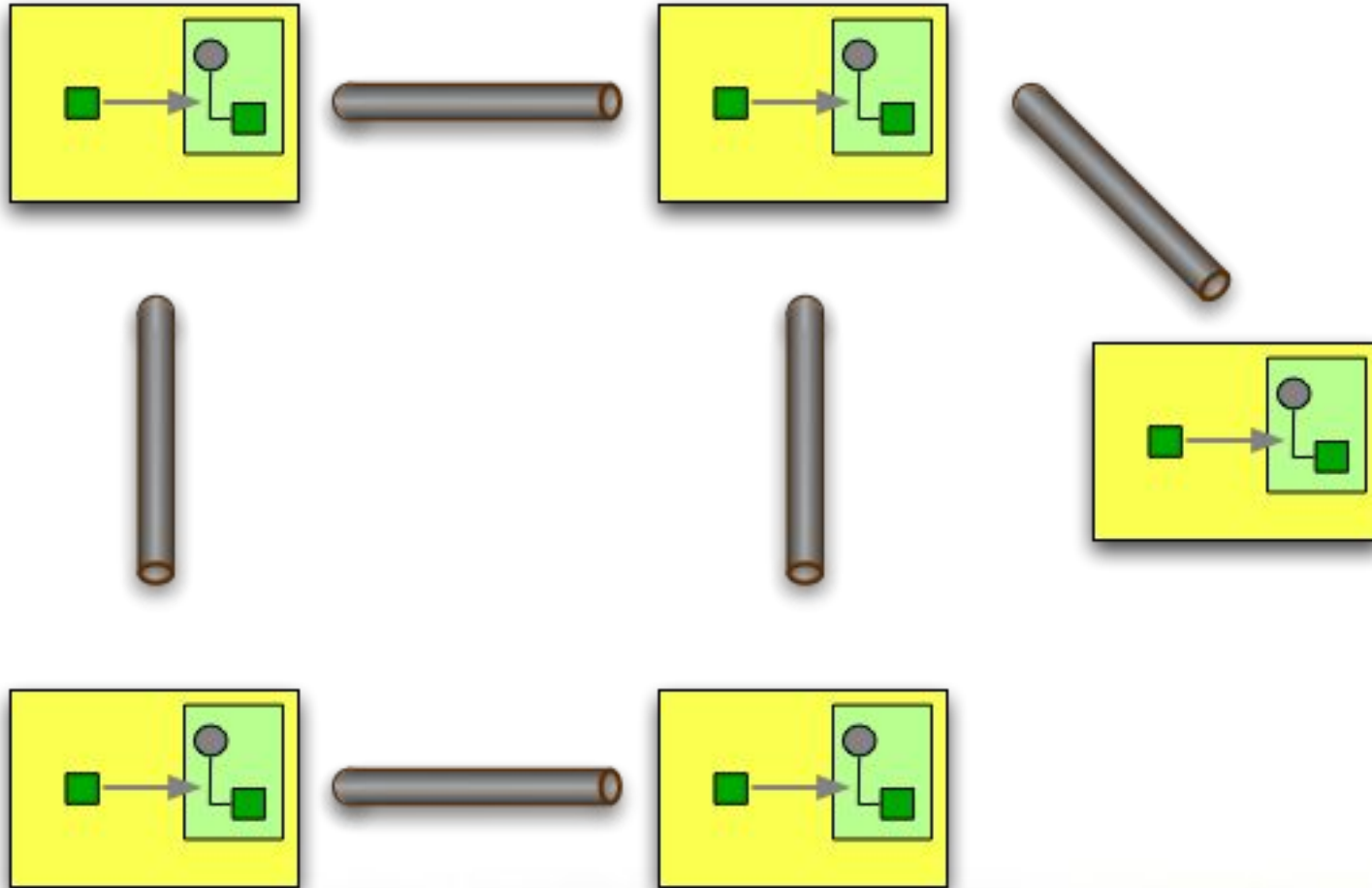
- Based on JBI
  - (Mediated Message Exchange Model)
- Distribution:
  - 100mb
- <http://servicemix.apache.org/how-to-evaluate-an-esb.html>

- 
- Most direct competition for Spring Integration
  - Fits less natural with Spring
  - Better routing
  - Fluent interface

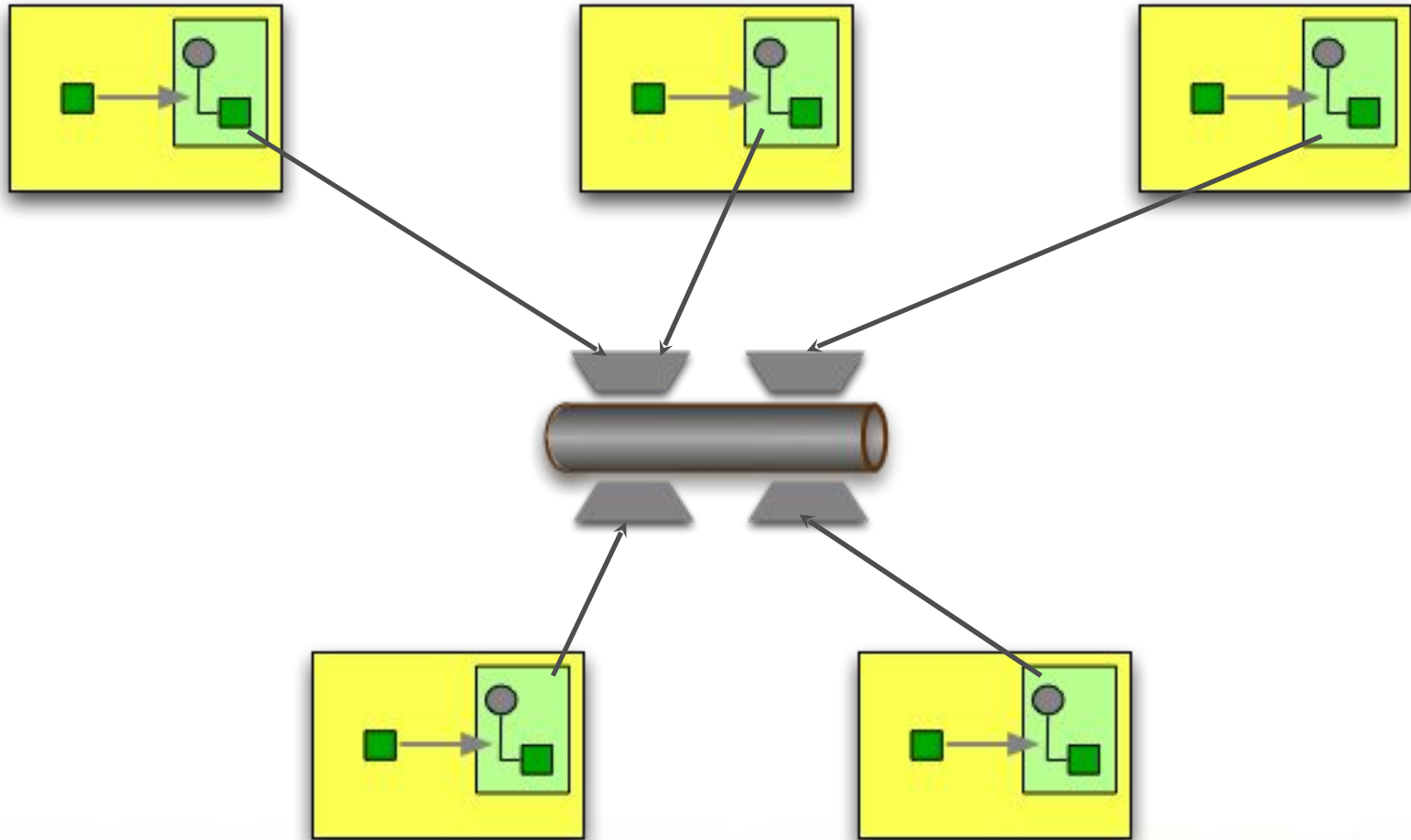
- 
- Heavyweight ESB (requires WebLogic)
  - Highly robust
  - Configuration based
  - Not easy to extend

- Routing complexity
  - a bus is useful for more complex routing problems
- Comfort zone
  - what will the developers feel at home with
- Keep it clean
  - don't scatter the routing rules

# Basic Integration



# With an ESB



# Topics in this session

---



- Introduction to Spring Integration
- Comparing Spring Integration to others
- Enterprise Integration Patterns
- **Summary and questions**

- Spring Integration
  - works from existing Spring applications
  - lightweight
  - decentralized (if you want)
- Enterprise Integration Patterns
  - describe ways of plumbing the enterprise application landscape.

- Roadmap
  - Milestone 6 currently available
    - Spring Security and Spring Web Service support
  - JMS, File, FTP, RMI, HttpInvoker, Mail, ApplicationEvents
  - 1.0 Final end of October

- *Enterprise Integration Patterns* (the book)
  - by Gregor Hohpe and Bobby Woolf (Addison Wesley 2004)
  - website: <http://www.eaipatterns.com>
- Spring Integration Home Page:
  - <http://www.springframework.org/spring-integration>