Using JMS with the Spring Framework

Grant Little

www.grantlittle.me

Basic JMS APIs

- Reasonable amount of boiler plate code

Enterprises use some form of framework

- Reduces boiler plate code

JMS & Spring Framework

- Mature
- Simple utilities
- Reduces boiler plate

Spring Framework Configuration

Prerequisites

Understanding of JMS

Understanding of the Spring Framework

Declaring Connection Factories Using Annotations

Declaring Connection Factories Using XML

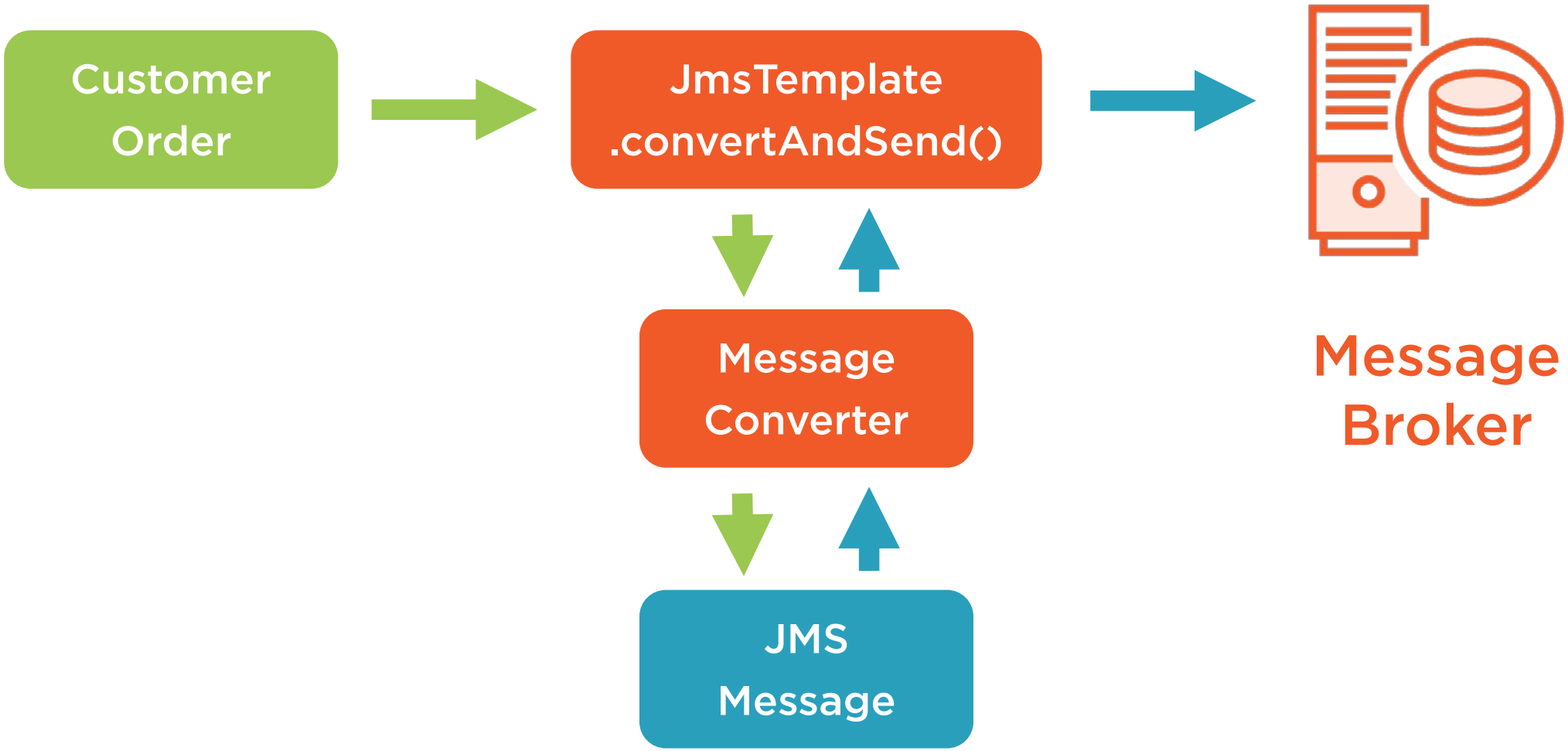
Introduction to the JmsTemplate & MessageConverters

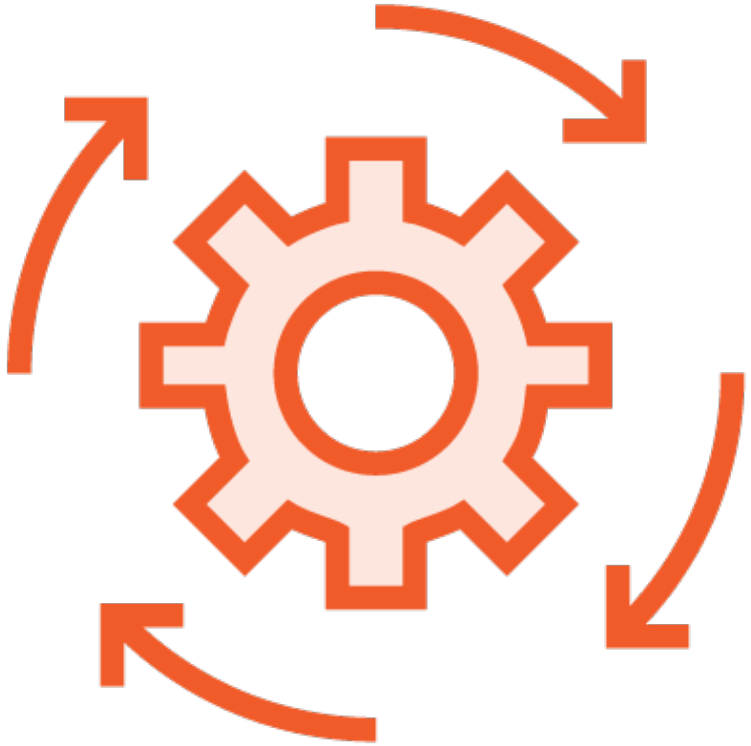
Receive Messages

Send Messages

- Create JMS Message Objects
- Send Business Objects

JmsTemplate & MessageConverters



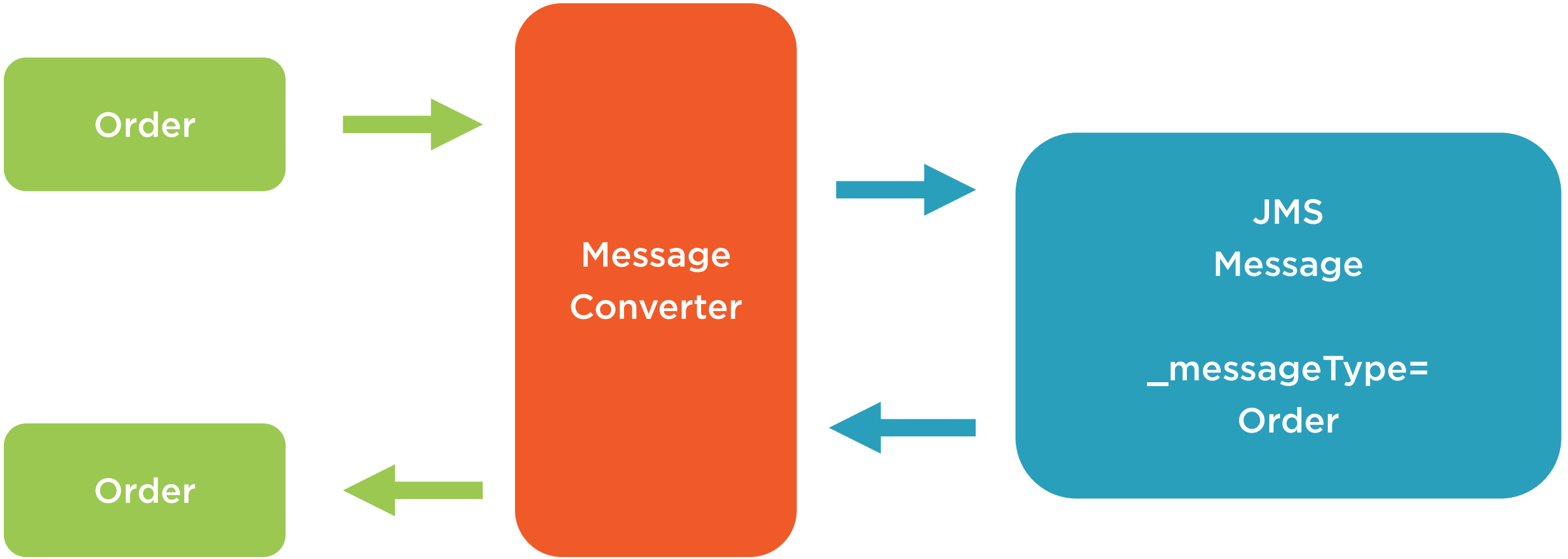


Message Converter

- XML
- Java Serialization
- JSON
- Create Your Own

Don’t use standard Java Serialization for messages – hard to upgrade your system dynamically

JmsTemplate & MessageConverters



JmsTemplate by default assumes destination is a Queue

The ConnectionFactory used with this template should return pooled Connections (or a single shared Connection) as well as pooled Sessions and MessageProducers.

Otherwise, performance of ad-hoc JMS operations is going to suffer

Declaring the JmsTemplate & MessageConverter Using Annotations

Declaring the JmsTemplate & MessageConverter Using XML

Sending Messages with the JmsTemplate

Creating JMS Messages Using JmsTemplate

```
jmsTemplate.send("DESTINATION_NAME", (session) -> {
    TextMessage msg = session.createTextMessage();
    msg.setStringProperty("PropertyX", "ValueY");
    msg.setText("Some text");
    return msg;
});
```

Receiving Messages with the JmsTemplate

receive()

receiveSelected()

receiveAndConvert()

receiveSelectedAndConvert()

“Convert”

- receiveAndConvert()
- receiveSelectedAndConvert()

Use a MessageConverter to convert JMS Message into a business object

“Receive”

- receive()
- receiveSelected()

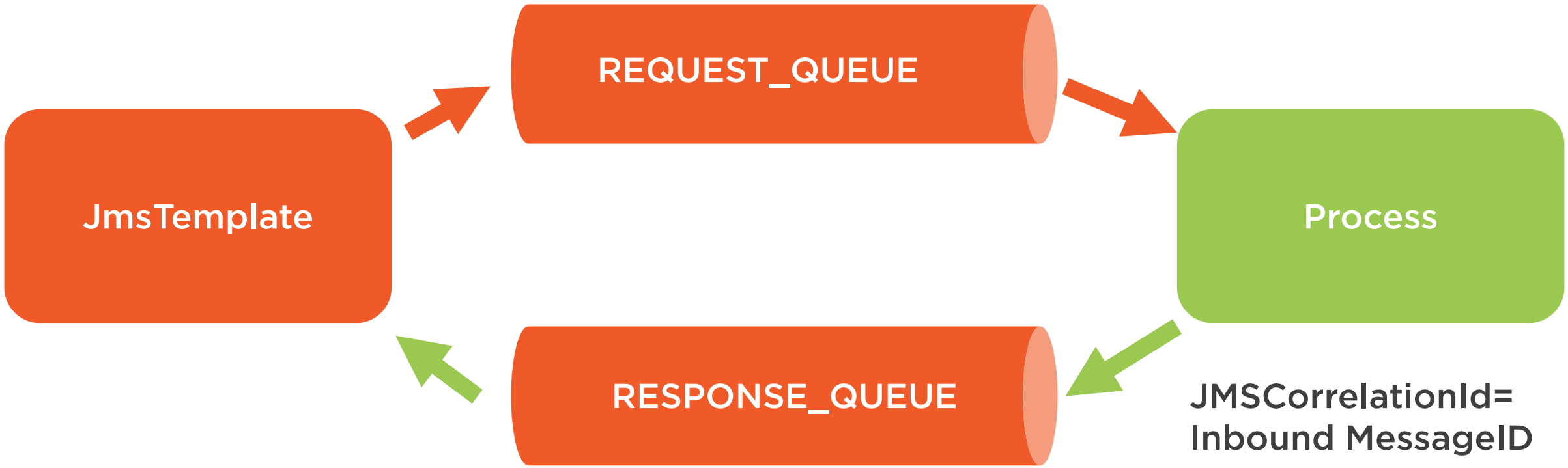
Receive a JMS Message object without any conversion

“Selected”

- receiveSelected()
- receiveSelectedAndConvert()

Use a message selector to only receive messages matching that selector

JmsTemplate Example



Limited Use Case

Message Listeners

= DefaultMessageListenerContainer

Introduction to the DefaultMessageListenerContainer

Role of the DefaultMessageListenerContainer

Message Broker

DefaultMessage ListenerContainer

MessageListener

MessageListener

MessageListener

- Multiple Consumers (Dynamic Scaling)
- Automatic Reconnection
- Caches JMS Resources
- Supports Transactions
- Stopped/Started at Runtime

DefaultMessageListenerContainer

“Don’t use Spring’s CachingConnectionFactory in combination with dynamic scaling”

Consuming Messages Asynchronously Using Annotation Driven Configuration

MessageConverter

ConnectionFactory

@EnableJMS

When using destination names the “DefaultMessageListener Container” assumes a Queue destination

Consuming Messages Asynchronously Using XML Driven Configuration

Summary

- Declaring Connection Factories
  - XML
  - Annotations
- Message Converters
- JmsTemplate
  - Sending
  - Receiving
- DefaultMessageListenerContainer
- Consuming Messages Asynchronously