Spring Integration: Advanced Message Handling Using Routing and Transformations

MESSAGE ROUTERS AND FILTERS

Steven Haines
PRINCIPAL SOFTWARE ARCHITECT

@geekcap www.geekcap.com

Overview

Overview of Routers and Filters

Payload Type Router

Header Value Router

Generic Router

Recipient List Router

Message Filters

Message Router

A messaging component that consumes a message from a message channel and republishes it to a different message channel depending on a set of conditions

Message Router Types

Payload Type Router

Routes based on Message Type

Header-Value Router

Routes based on a header value

Generic Router

Routes based on custom logic

Recipient List Router

Routes to multiple channels

Message Filter

A messaging component that consumes a message from a message channel and only routes it to an output channel if the message meets specific criteria

Payload Type Router

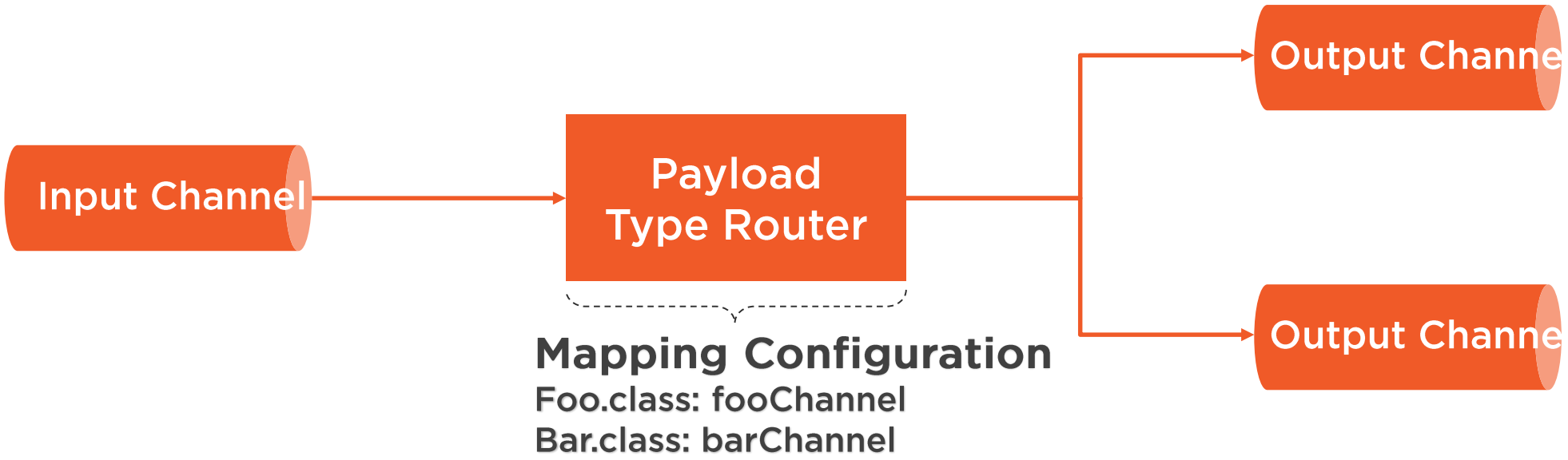
Payload Type Router

A message router that routes messages based on their payload type, as defined by a payload-type mapping configuration

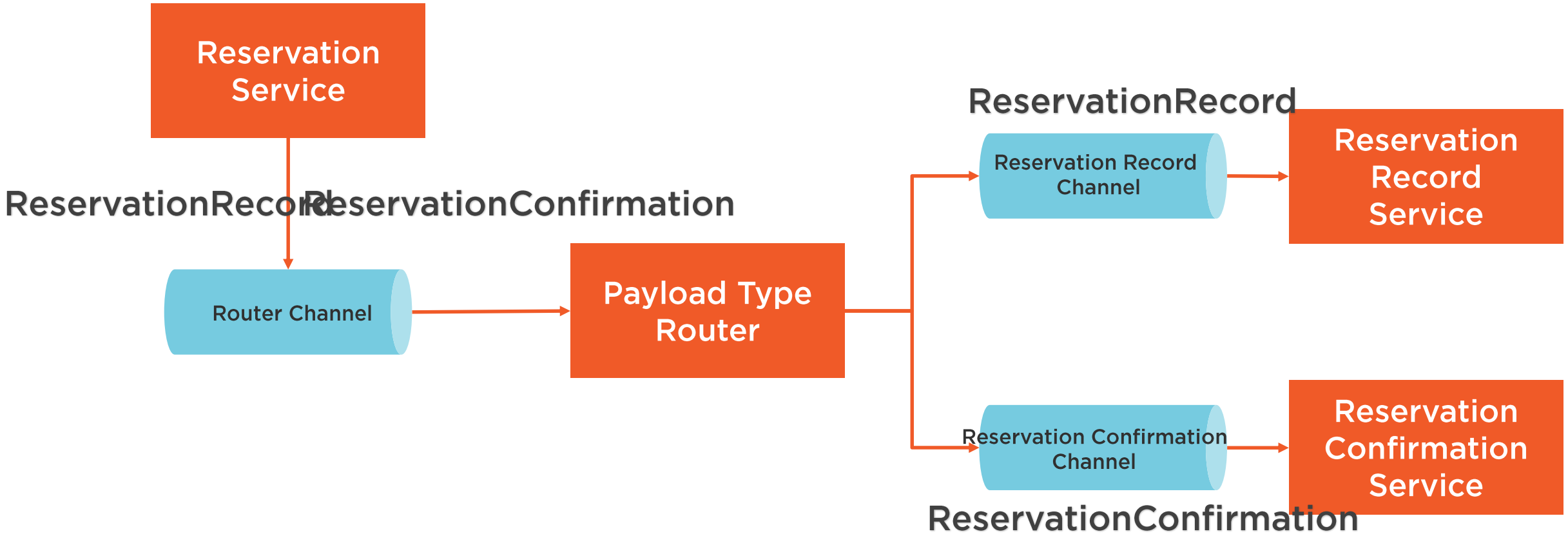
Use Case

You want to publish messages of different types to a single channel and deliver those messages to specific destination topics

How the Payload Type Router Works



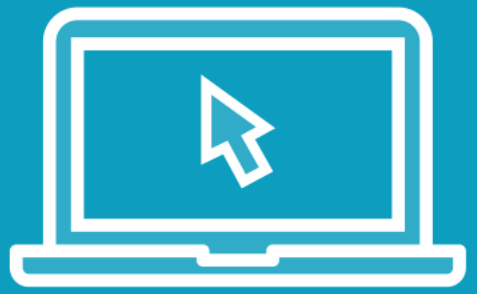
Example: Reservation Service



```
@Configuration
@EnableIntegration
public class PayloadTypeRouterConfig {
   @Bean
    public MessageChannel routerChannel() {
        return new DirectChannel();
    }

   @Bean
   @ServiceActivator(inputChannel =
"routerChannel")
   public PayloadTypeRouter
payloadTypeRouter() {

        PayloadTypeRouter router = new
PayloadTypeRouter();
        router.setChannelMapping(

ReservationRecord.class.getName(),
             "reservationRecordChannel");
        router.setChannelMapping(

ReservationConfirmation.class.getName(),

"reservationConfirmationChannel");
```

- Setup a configuration class and enable Spring Integration
- Create a channel that delivers messages to the router
- Configure a PayloadTypeRouter bean to listen for messages published to the router channel
- Create a PayloadTypeRouter
- Define channel mappings on the router, specifying the message payload class type and the name of the destination channel name

Demo



Define our components

- Three channels
- Payload Type Router
- Reservation Service
- Reservation Record and Confirmation Services

Invoke the Reservation Service to publish the messages

Publish the messages to the router channel

Route the messages

Validate that the correct services received the correct messages

Summary

A payload type router routes messages based on their payload type

Routing is configured in the payload type router's channel mapping

It allows one channel to receive different types of messages that ultimately are processed by different components

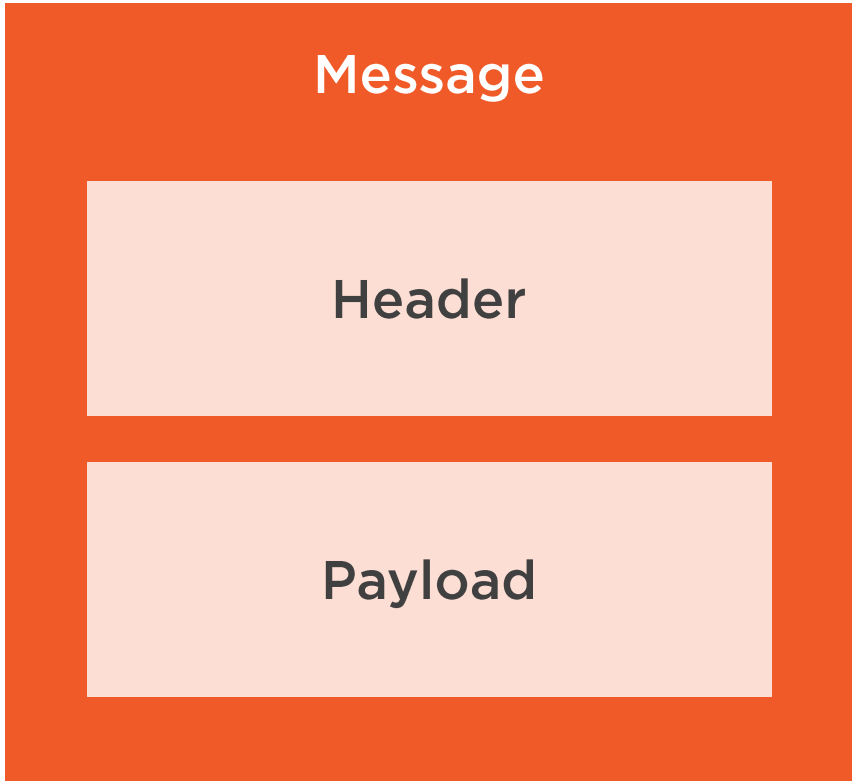
Next up: Header-Value Routers

Header-Value Router

Header-Value Router

A message router that routes messages based on the value of a message header

Messages



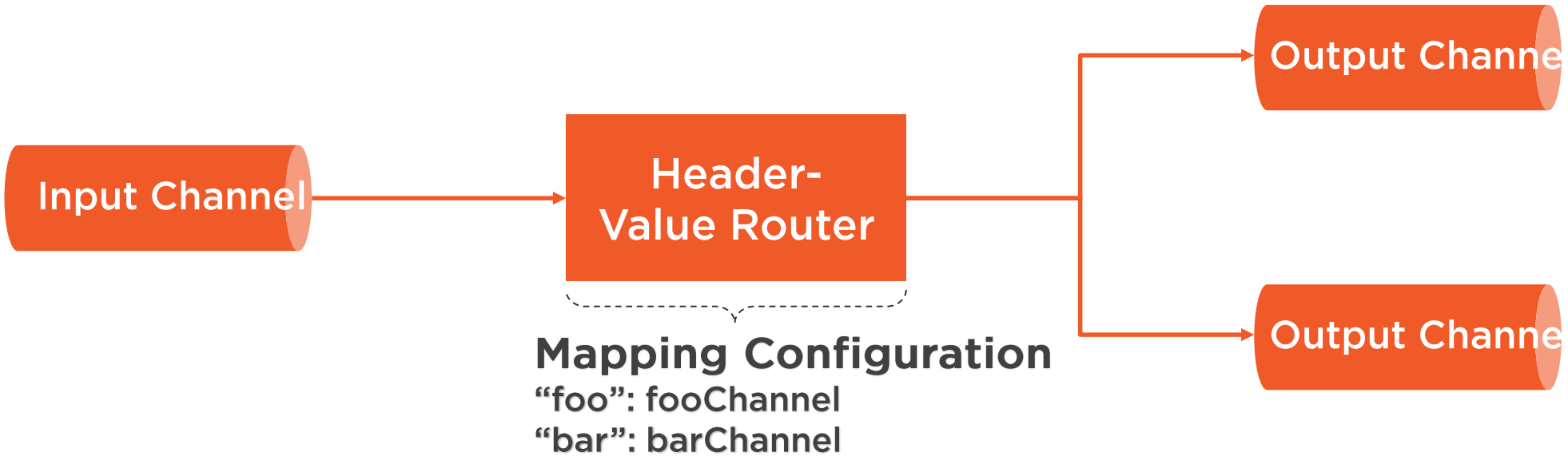
The Header contains additional information about the message, such as a correlation ID, sequence ID, expiration time, and custom headers

The Payload contains the body of the message

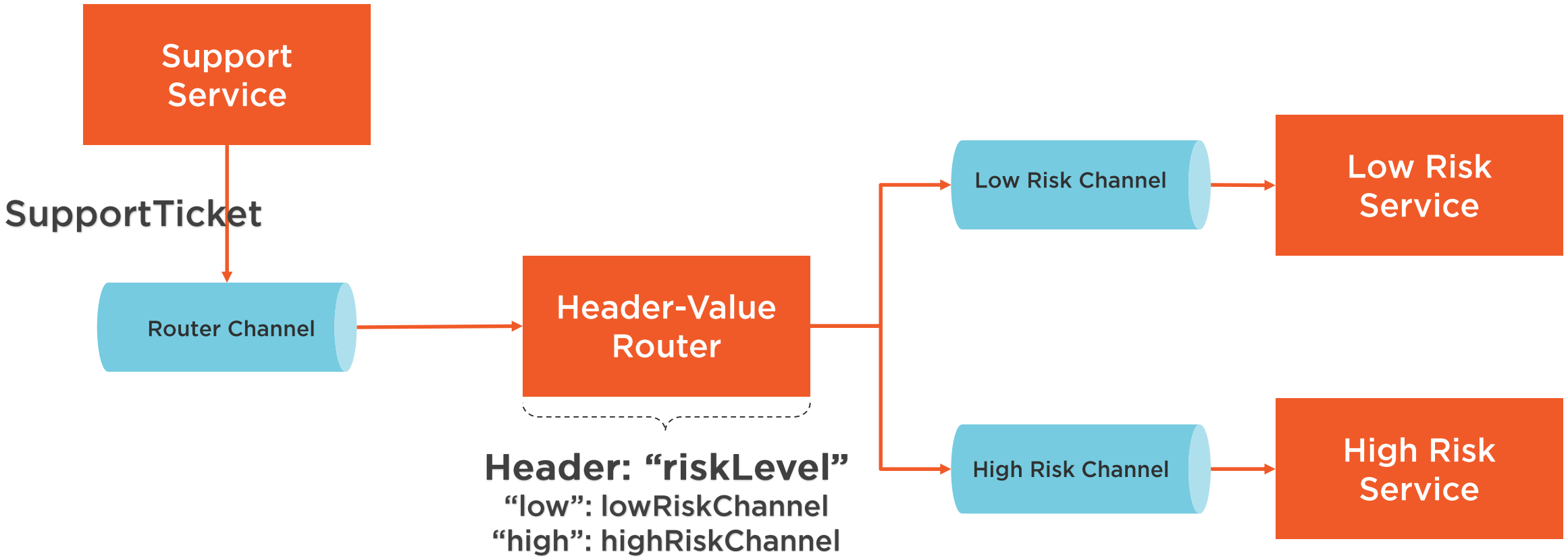
Use Case

You want to handle the same type of messages differently based on criteria that is easily added to a message header

How the Header-Value Router Works



Example: Support Service



```
@Configuration
@EnableIntegration
public class HeaderValueRouterConfig {
    @Bean
    public MessageChannel routerChannel() {
        return new DirectChannel();
    }

    @Bean
    @ServiceActivator(inputChannel =
"routerChannel")
    public HeaderValueRouter
headerValueRouter() {
        HeaderValueRouter router =
                                    new
HeaderValueRouter("riskLevel");

        router.setChannelMapping("low",
"lowRiskChannel");
        router.setChannelMapping("high",
"highRiskChannel");

        return router;
    }
}
```

- Setup a configuration class and enable Spring Integration
- Create a channel that delivers messages to the router
- Configure a HeaderValueRouter bean to listen for messages published to the router channel
- Create a HeaderValueRouter
- Define channel mappings on the router, specifying the header value and the name of the destination channel name

Demo

Define our components

- Three channels
- Header Value Router
- Support Service
- High and Low Risk Services

Invoke the Support Service to publish the messages to the router channel

Route the messages, based on their risk levels

Validate that the correct services received the correct messages

Summary

A header-value router routes messages based on their values of a specific message header

Routing is configured in the header-value router's channel mapping

It allows messages to be directed to specific channels based on header values

Next up: Generic Routers

Generic Router

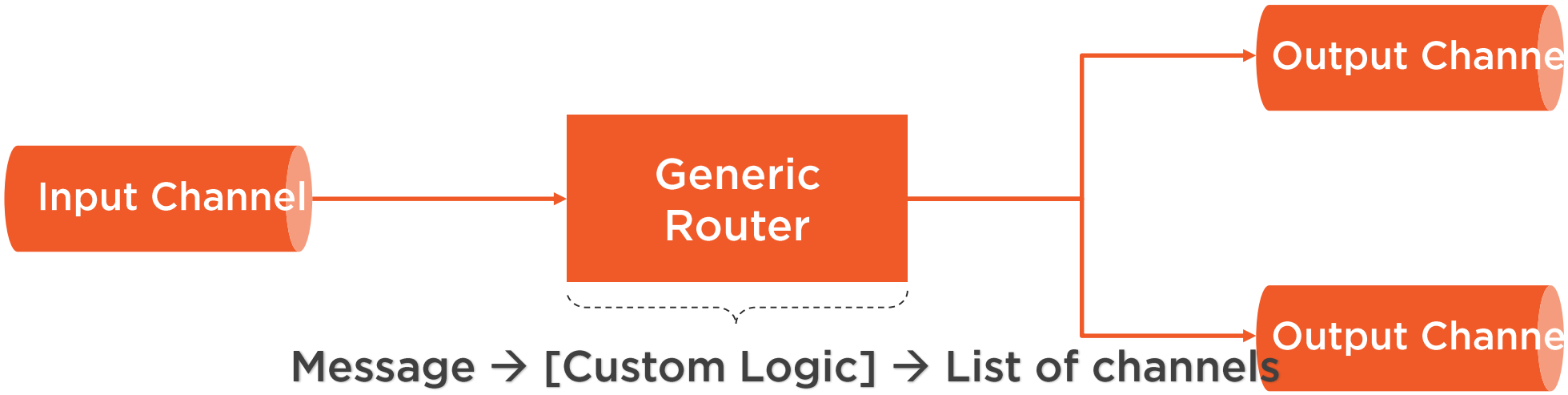
Generic Router

A general-purpose message router that routes messages based on you own custom criteria

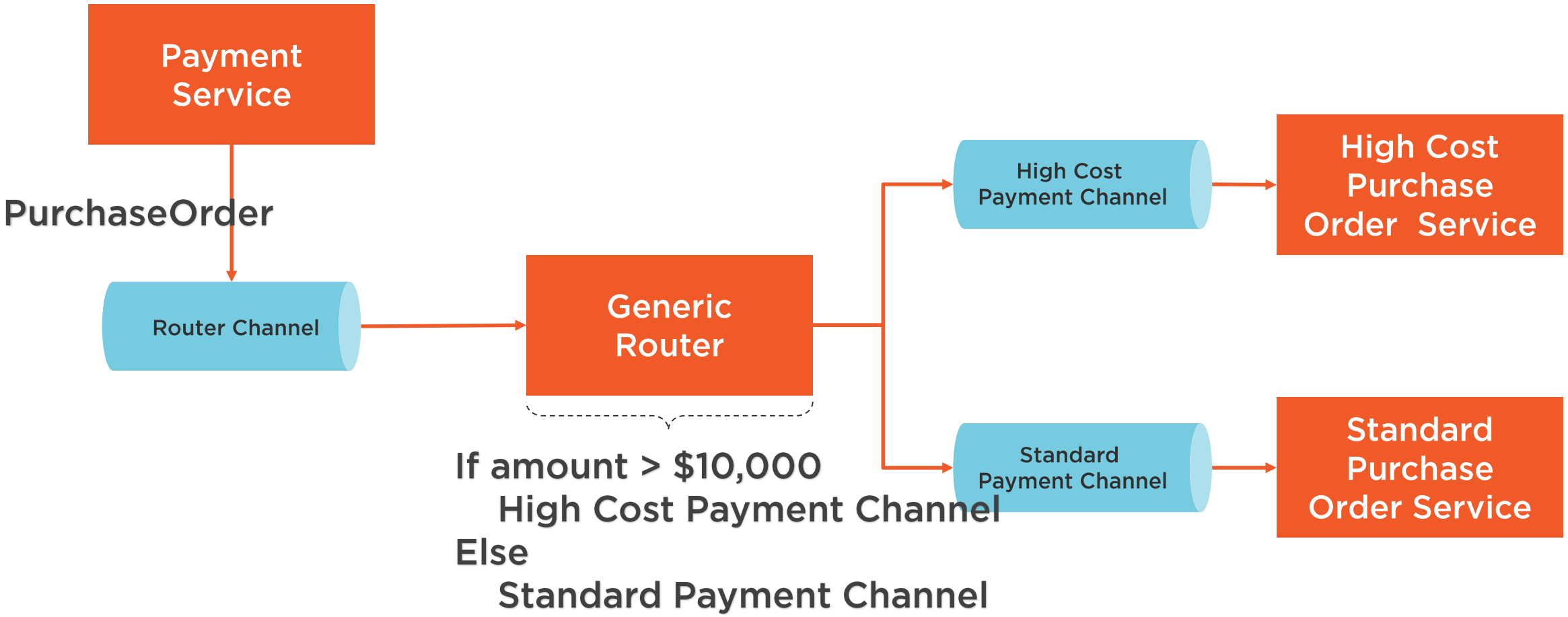
Use Case

You want to route messages using your own custom logic, such as by examining the contents of the message payload

How the Generic Router Works



Example: Payment Service



```
@Configuration
@EnableIntegration
public class GenericRouterConfig {

    @Bean
    @Router(inputChannel = "routerChannel")
    public MessageRouter myCustomRouter() {
       return new AbstractMessageRouter() {
         @Override
         protected Collection<MessageChannel>

determineTargetChannels(Message<?> message) {
                  PurchaseOrder purchaseOrder =

(PurchaseOrder)message.getPayload();
                  if (purchaseOrder.getAmount()
> 10000.0) {
                     return
Arrays.asList(highCostPaymentChannel());
                  }
                  return
Arrays.asList(standardPaymentChannel());
              }
         };
    }
    public MessageChannel
```

- Setup a configuration class and enable Spring Integration
- Configure a custom router bean to listen for messages published to the router channel
- Create an AbstractMessageRouter
- Override determineTargetChannels
- Check the amount and route amounts greater than $10,000 to the high cost payment channel, otherwise route to the standard payment channel
- Bean methods that create channels

Demo

Define our components

- Three channels
- Custom Router
- Payment Service
- Standard and High Cost Purchase Order Services

Invoke the Payment Service to publish purchase order

Route the messages based on PO amount

Validate that the correct services received the correct messages

Summary

A generic router routes messages based on your own custom logic

Create an AbstractMessageRouter and override its determineTargetChannels() method with your own logic

Best solution for routing messages based on payload contents

Next up: Recipient List Router

Recipient List Router

Recipient List Router

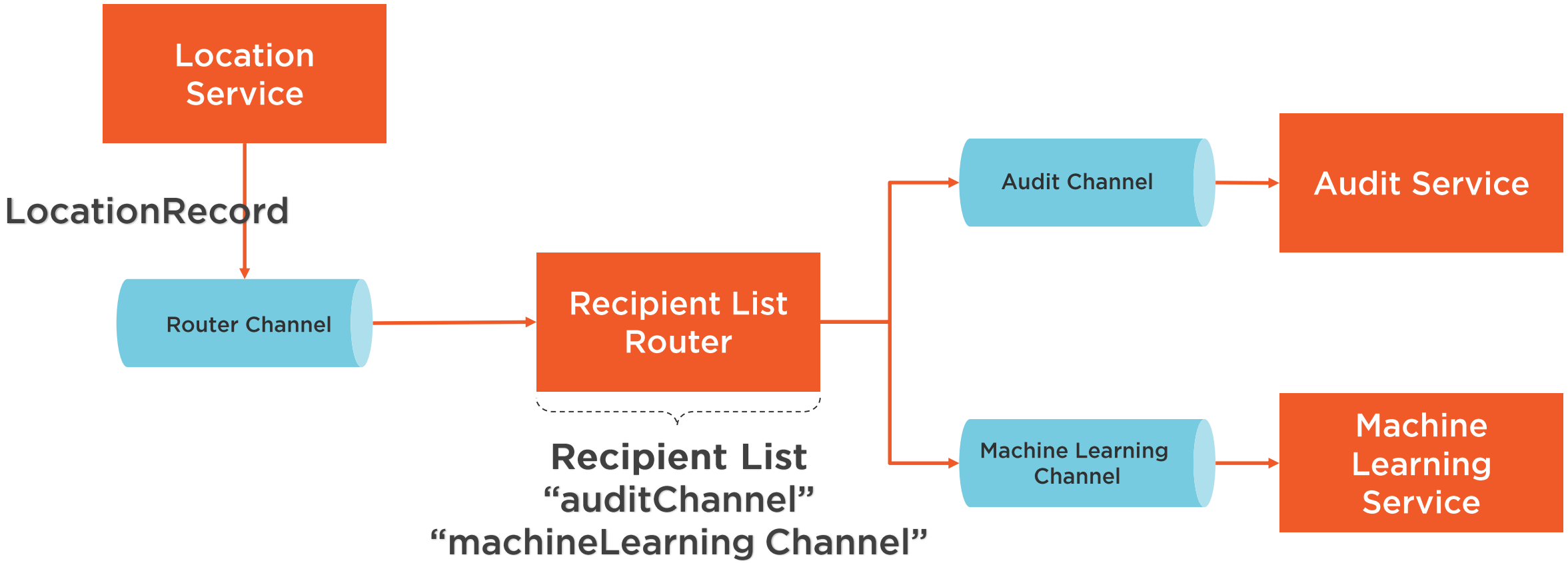
A message router that sends received messages to a statically defined list of message channels

Use Case

You want to publish a message to multiple channels, or you want to publish messages to additional channels

How the Recipient List Router Works

Example: Location Service



```
@Configuration
@EnableIntegration
public class RecipientListRouterConfig {

    @ServiceActivator(inputChannel =
"routerChannel")
    @Bean
    public RecipientListRouter router() {
        RecipientListRouter router = new
RecipientListRouter();
        router.addRecipient("auditChannel");

router.addRecipient("machineLearningChannel");
        return router;
    }

    @Bean
    public MessageChannel routerChannel() {…}

    @Bean
    public MessageChannel auditChannel() {…}

   @Bean
    public MessageChannel
machineLearningChannel() {…}
}
```

- Setup a configuration class and enable Spring Integration
- Configure a Router listening on the routerChannel
- Create a RecipientListRouter
- Add destination channels
- Bean methods that create channels

Demo

Define our components

- Three channels
- Recipient List Router
- Payment Service
- Audit and Machine Learning Services

Invoke the Location Service to publish purchase order

Route the messages

Validate that the correct services received the correct messages

Summary

A recipient list router routes messages to multiple channels

It is used when you want to publish a message to multiple channels, or you want to publish messages to additional channels

Next up: Filters

Filters

Message Filter

A messaging component that consumes a message from a message channel and only routes it to an output channel if the message meets specific criteria

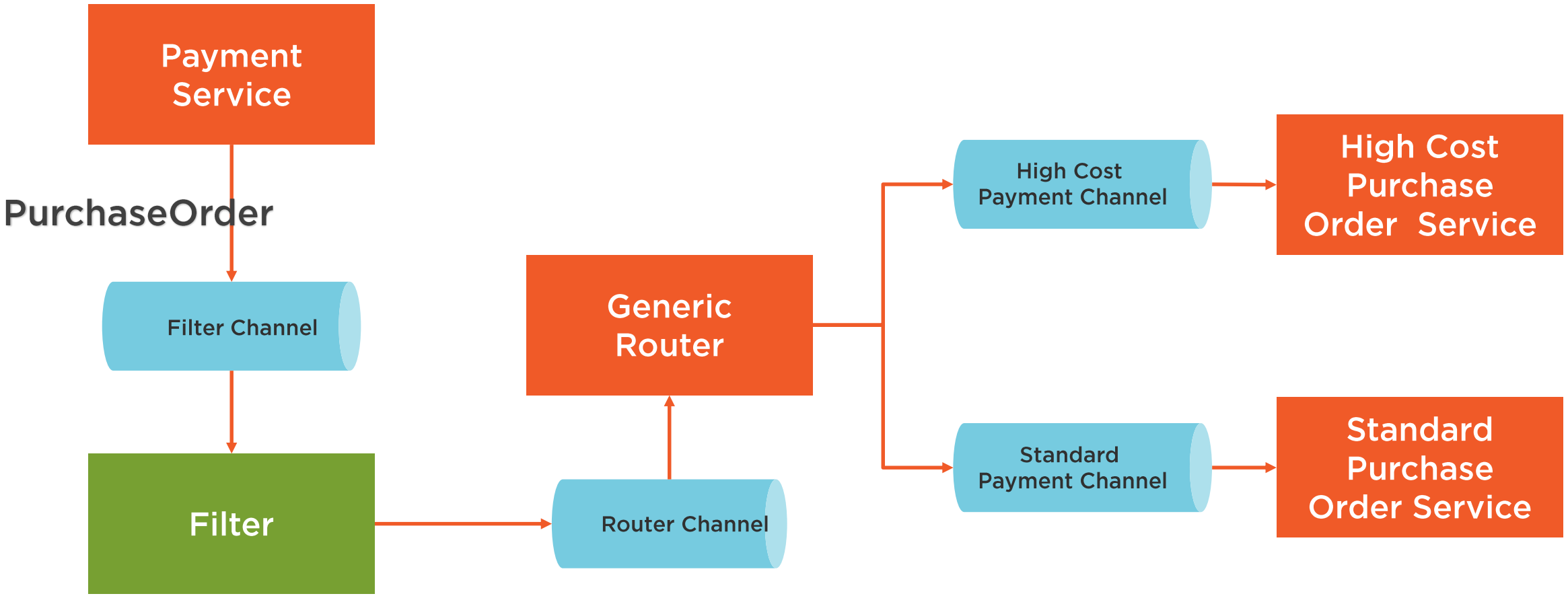
Use Case

You only want to send messages to a channel if it meets a specific criteria

How Filters Works

Criteria:
Return true or false

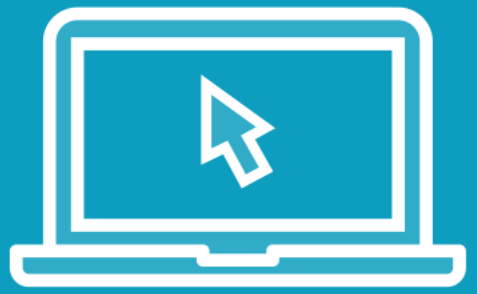
Example: Payment Service With a Filter



```
@Configuration
@EnableIntegration
public class GenericRouterConfig {

    @Bean
    @Filter(inputChannel = "filterChannel",
                 outputChannel =
"routerChannel")
    public MessageSelector noCostFilter() {
       return message ->

((PurchaseOrder)message.getPayload())

.getAmount() > 0.0;
    }

    public MessageChannel
standardPaymentChannel() {…}
    public MessageChannel
highCostPaymentChannel() {…}
    public MessageChannel routerChannel() {…}
    public MessageChannel filterChannel() {…}
}
```

- Setup a configuration class and enable Spring Integration
- Configure a filter listening on the filterChannel and publishing messages to the routerChannel
- Create a MessageSelector (as a lambda function) that filters out any message with an amount less than or equal to $0
- Bean methods that create channels

Demo



Define our components

- Four channels
- Filter and Router
- Update Gateway

Invoke the Payment Service to publish purchase order

Filter $0 messages out and route messages based on purchase order amount

Validate that the $0 message is filtered out and that the correct services received the correct messages

Summary

A filter consumes a message from a message channel and only routes it to an output channel if the message meets specific criteria

It is used when you only want to send messages to a channel if it meets a specific criteria

Next up: Module Wrap-up

Conclusion

Message Router

A messaging component that consumes a message from a message channel and republishes it to a different message channel depending on a set of conditions

Message Router Types

Payload Type Router

Routes based on Message Type

Header-Value Router

Routes based on a header value

Generic Router

Routes based on custom logic

Recipient List Router

Routes to multiple channels

Message Filter

A messaging component that consumes a message from a message channel and only routes it to an output channel if the message meets specific criteria

Summary

You should understand what routers and filters do

You should understand the types of message routers and when to use them

You should feel comfortable implementing them in your applications

Next Module: Splitters and Aggregators