Spring Integration: Using Channel Adapters to Integrate with External Systems

INTEGRATING WITH RESTFUL WEB SERVICES

Steven Haines
PRINCIPAL SOFTWARE ARCHITECT

@geekcap www.geekcap.com

Overview

Overview of RESTful Web Service Integrations

HTTP Outbound Channel Adapters

HTTP Outbound Gateways

RESTful Web Services

Representational state transfer (REST) is an architectural pattern, based on HTTP, used for creating Web Services.

HTTP Verbs

GET

Retrieve a resource

POST

Create a new resource

PUT

Update an existing resource

DELETE

Remove a resource

PATCH

Partially update and existing resource

Resource Formats

JSON

Most Popular

XML

Still very much alive

Channel Adapters and Gateways

Inbound Channel Adapter

Receive a Web Service call and handle it using Spring Integration

Outbound Channel Adapter

Invoke a RESTful Web Service

Outbound Gateway

Invoke a RESTful Web Service and receive a response

HTTP Outbound Channel Adapters and Gateway

HTTP Outbound Components

Outbound Channel Adapter

HttpRequestExecuting
MessageHandler
expectRepy = false

Outbound Gateway

HttpRequestExecuting
MessageHandler
expectReply = true

```
@Bean
@ServiceActivator(inputChannel =
"toServiceChannel")
public MessageHandler postToService() {
    HttpRequestExecutingMessageHandler handler
=
                  new
HttpRequestExecutingMessageHandler(

"http://someservice/resource");


messageHandler.setHttpMethod(HttpMethod.POST);

    messageHandler.setExpectReply(false);

    return handler;
}
```

- Define a ServiceActivator
- Create an HttpRequestExecutingMessageHandler, specifying the URL to a RESTful Web Service
- Set the HTTP method
- We do not expect a reply

```
@Bean
@ServiceActivator(inputChannel =
"getReservationChannel")
public MessageHandler httpOutboundChannel() {
    HttpRequestExecutingMessageHandler handler
=
             new
HttpRequestExecutingMessageHandler(

"http://localhost:7080/reservation/{id}");
    handler.setHttpMethod(HttpMethod.GET);

    handler.setExpectReply(true);

handler.setExpectedResponseType(Reservation.cl
ass);

    SpelExpressionParser p = new
SpelExpressionParser();

    handler.setUriVariableExpressions(
         Collections.singletonMap("id",
p.parseRaw("payload")));


handler.setOutputChannelName("replyChannel");
```

- Define a ServiceActivator
- Create an HttpRequestExecutingMessageHandler, specifying the URL to a RESTful Web Service
- Set the HTTP method
- We expect a reply of a Reservation class

- Resolve ID in the URL to the message payload
- Set the name of the channel to which to send the response from the web service

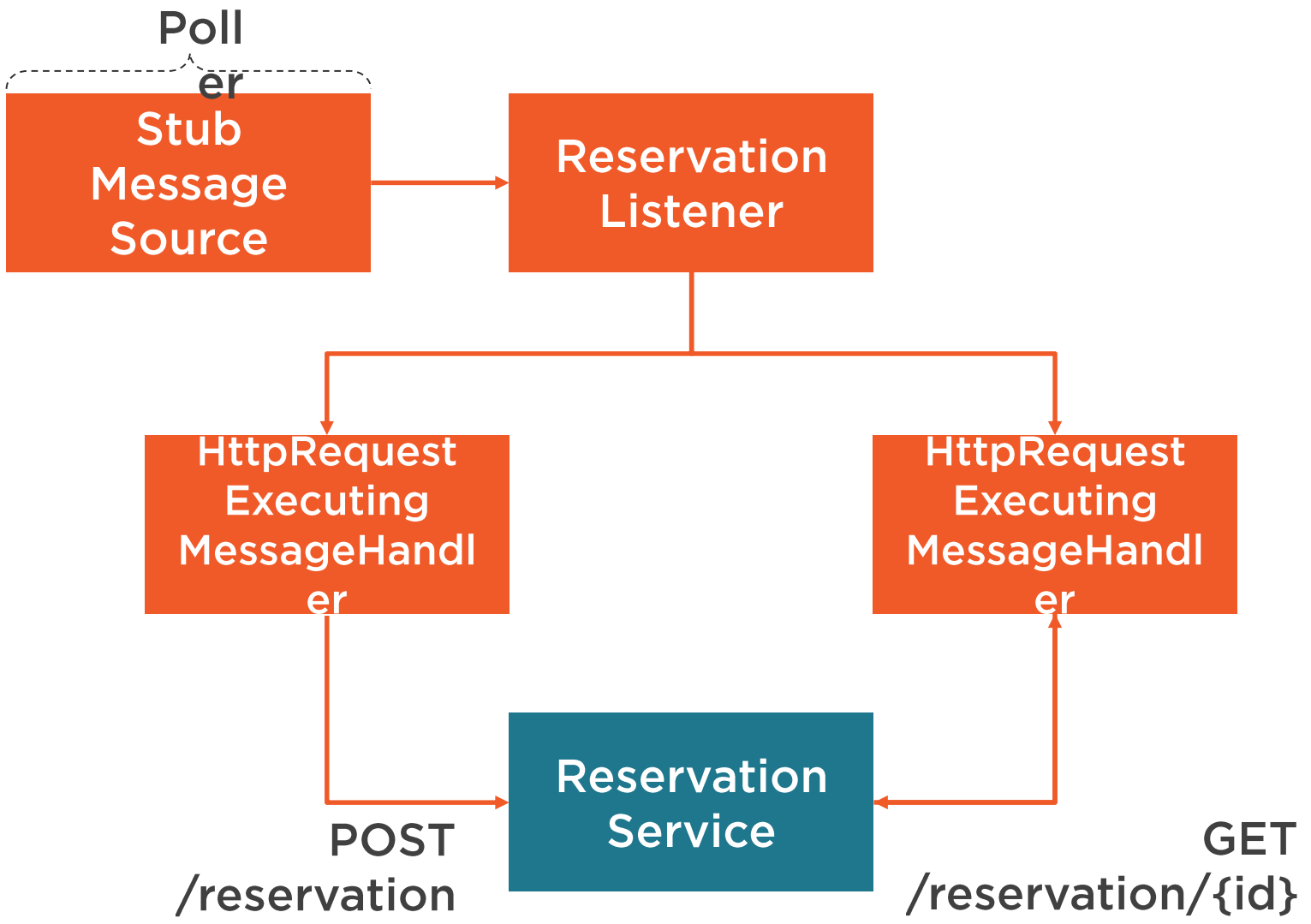
```
@Bean
@InboundChannelAdapter(value =
“fromStubChannel",
                                    poller =
@Poller(fixedDelay = "5000"))
public MessageSource<List<Reservation>>
stubSource() {
    return () -> {
        List<Reservation> reservations = new
ArrayList<>();
        reservations.add(new Reservation(1L,
"Smith", "", 1));
        reservations.add(new Reservation(2L,
"Jones", "", 1));
        return
MessageBuilder.withPayload(reservations).build
();
    };
}
```

- Define an InboundChannelAdapter
- Create a Lambda expression that generates a list containing two reservations
- Return a message with the list as its payload

```
@Splitter(inputChannel = “fromStubChannel",
                outputChannel =
"reservationFromStubChannel")
public List<Reservation> splitter(
```

- Create a splitter that publishes individual Reservations to a channel

Integrating with a RESTful Web Service



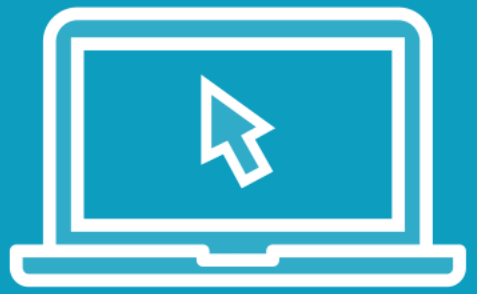
```
POST /reservation

GET /reservation/{id}

GET /reservations

PUT /reservation/{id}

DELETE /reservation/{id}
```

Reservation Service

A simple Spring MVC RESTful Web Service using Spring Data and an embedded H2 database

Course GitHub Page: https://github.com/geekcap-pluralsight/channeladapters

Demo



Build our application

- Stub Message Source
- Outbound Channel Adapter
- Outbound Gateway
- ReservationListener

Run the application

Validate the results

Conclusion

HTTP Outbound Components

Outbound Channel Adapter

HttpRequestExecuting
MessageHandler
expectRepy = false

Outbound Gateway

HttpRequestExecuting
MessageHandler
expectReply = true

Summary

You should understand how to integrate with RESTful Web Services

You should understand how to use different HTTP methods, URL variables, and handle HTTP responses

Next Module: Integrating with Custom External Systems