

Engenharia de Softwares Escaláveis

Domain-Driven Design (DDD) e Arquitetura de
Softwares Escaláveis com Java

Agenda

Etapas 6: Design e Implementação de Microsserviços com Comunicação Assíncrona e Message Brokers em Java.

- Spring Cloud Stream.
- Revisão do PubSub.
- Implementação do Spring Cloud Stream.





Spring Cloud Stream

Spring Cloud Stream é um framework para construir aplicações orientadas a eventos e mensagens (como microservices) de forma padronizada e desacoplada.

Ele funciona como um "adaptador universal" para sistemas de mensageria - **Message Brokers**.

A sua aplicação Java escreve a lógica de negócio (o que fazer com uma mensagem) e se comunica com "bindings" (canais) genéricos.

Spring Cloud Stream, através de um "binder" específico (como o spring-cloud-gcp-pubsub-stream-binder), traduz essa comunicação para o sistema de mensagens que você estiver usando, seja Kafka, RabbitMQ, Google Pub/Sub etc.

O principal objetivo do **Spring Cloud Stream** é abstrair a complexidade e a implementação dos Message Brokers.

Ele permite que sua lógica de aplicação seja totalmente independente do broker.

Sua aplicação apenas "envia" ou "recebe" dados de canais, sem precisar saber como se conectar ao Kafka, autenticar no Google Pub/Sub ou configurar uma fila no RabbitMQ.

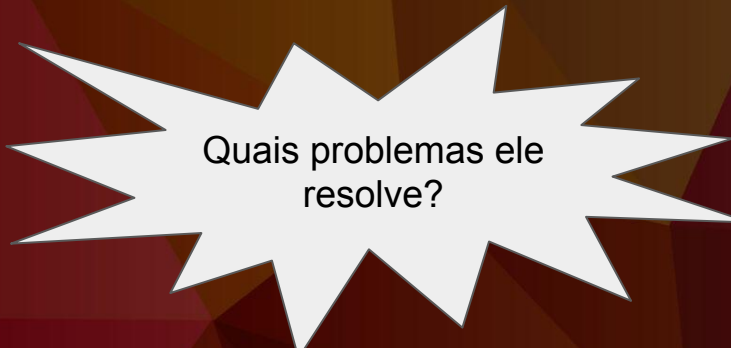
Ele se integra perfeitamente com o modelo de programação funcional do **Spring**, permitindo que você defina consumidores, produtores ou transformadores de mensagens como simples Beans Java.

Acoplamento e "Vendor Lock-in"

Problema: se você escreve seu código usando a biblioteca cliente nativa do Kafka, você está "preso" ao Kafka. Mudar para o Google Pub/Sub ou RabbitMQ exigiria reescrever todo o código de comunicação.

Solução: com o **Stream**, sua lógica de negócio permanece idêntica. Para mudar de broker, você apenas trocava a dependência no pom.xml e atualizaria o `application.properties`.

O seu código Java `.java` não mudaria.

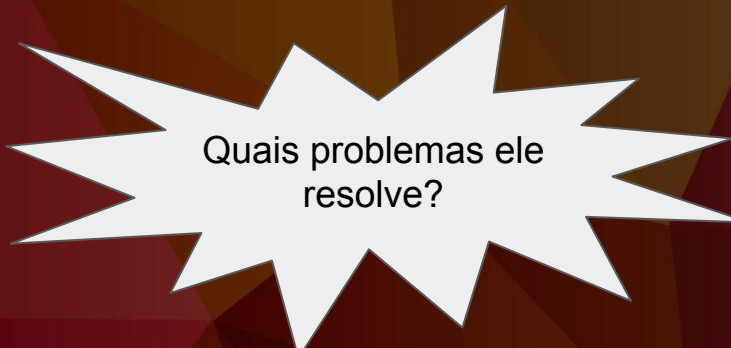


Quais problemas ele resolve?

Complexidade e Código "Boilerplate"

Problema: configurar manualmente produtores, consumidores, listeners, pools de conexão, serialização/desserialização, retentativas (retries) e tratamento de erros para cada broker é complexo, repetitivo e propenso a erros.

Solução: o **Stream**, com a automação do **Spring Boot**, cuida de toda essa configuração (plumbing). Você apenas define o binding (ligação) no `application.properties` e o **Spring** faz todo o resto.

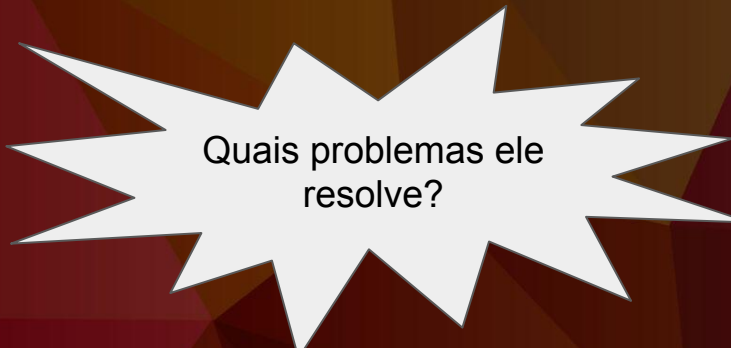


Quais problemas ele resolve?

Padronização de Comunicação

Problema: em uma arquitetura de microservices, pode ser difícil padronizar como os serviços trocam dados.

Solução: o **Stream** formaliza os conceitos de "canais de entrada" (inputs) e "canais de saída" (outputs) através dos bindings. Isso torna a arquitetura de dados mais clara e fácil de entender.

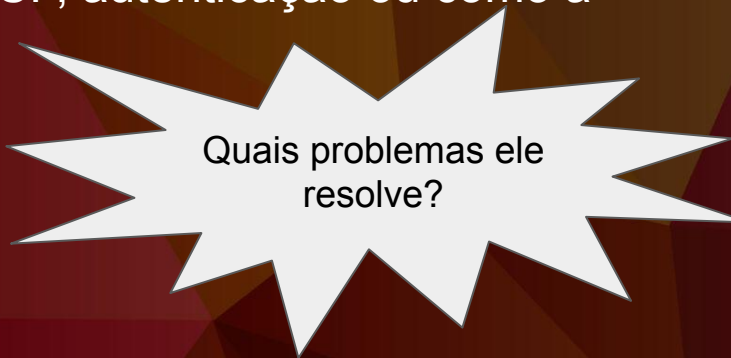


Quais problemas ele resolve?

Foco no Negócio

Problema: desenvolvedores gastam muito tempo escrevendo código de infraestrutura (o "como" enviar/receber) em vez de focar na lógica de negócio (o "o que" fazer com a mensagem).

Solução: o Stream permite que o desenvolvedor foque 100% na lógica de negócio. A classe de configuração apenas se preocupa em registrar o payload da mensagem, sem se preocupar com a conexão ao GCP, autenticação ou como a mensagem chegou lá.

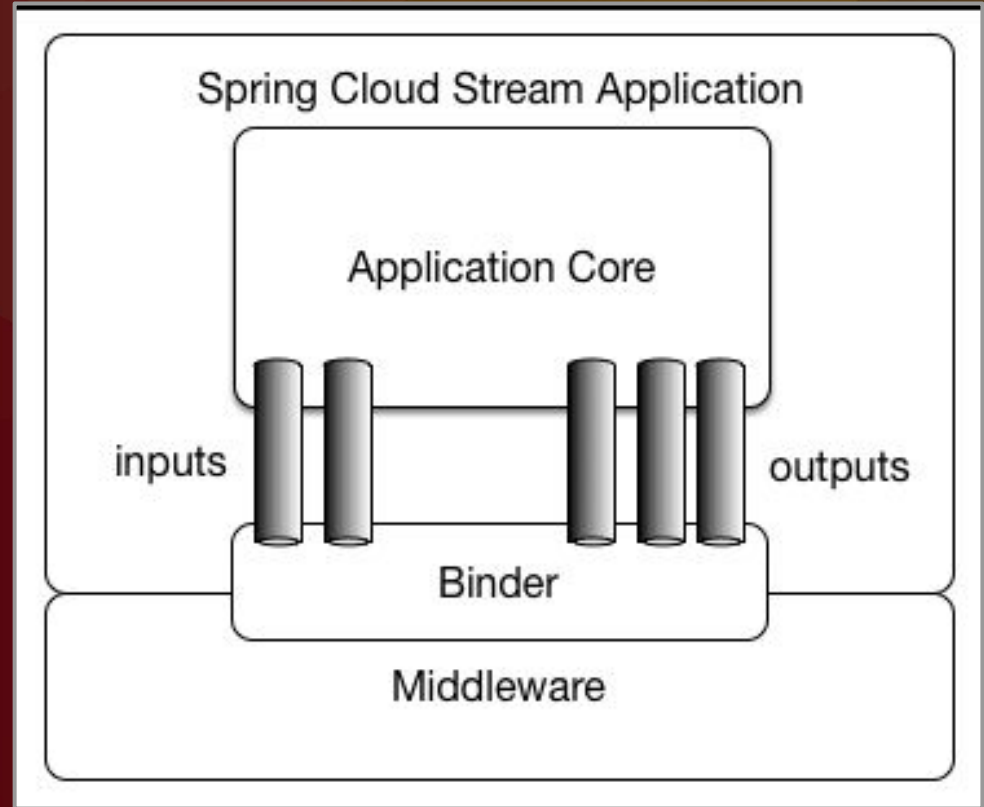


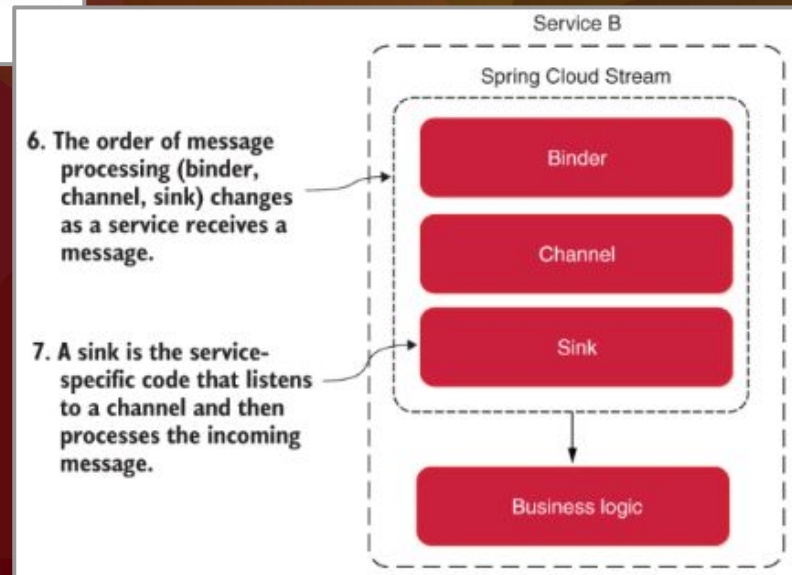
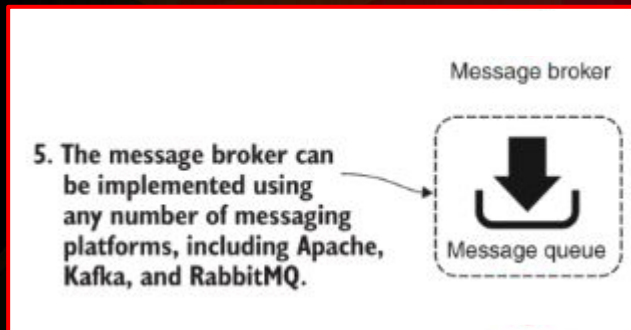
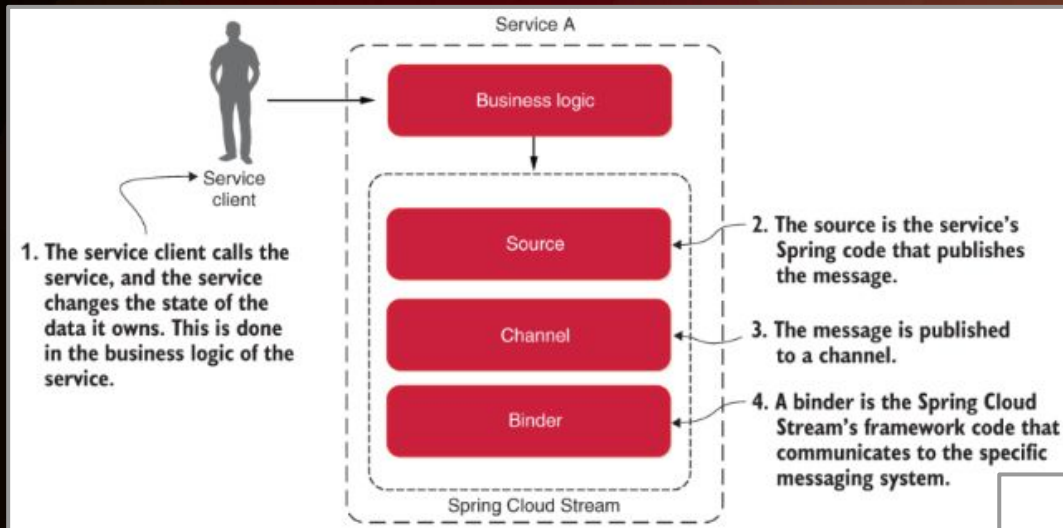
Quais problemas ele resolve?

Um aplicativo **Spring Cloud Stream** consiste em um núcleo neutro de middleware.

O aplicativo se comunica com o mundo externo estabelecendo ligações entre destinos expostos pelos brokers externos e argumentos de entrada/saída no seu código.

Os detalhes específicos do broker necessários para estabelecer ligações são manipulados por implementações específicas de middleware.

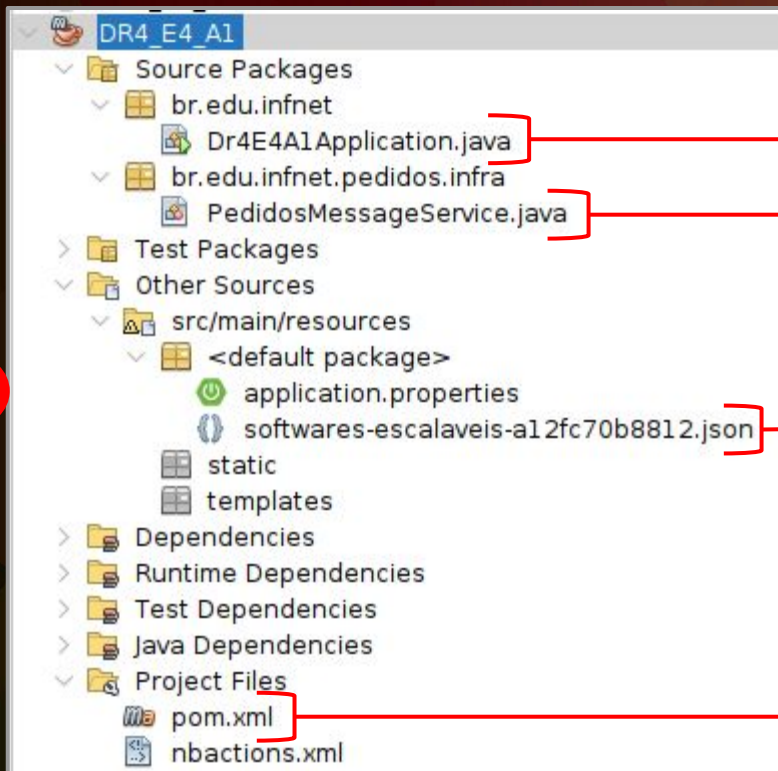




Conceito	O que é?	Exemplo
Source Produtor	<p>É o componente que envia mensagens para um canal.</p> <p>É a origem dos dados no fluxo.</p> <p>No modelo funcional do Spring, isso é geralmente um <code>Supplier<T></code>.</p>	<p>A classe <code>StreamBridge</code> atua como um <code>Source</code> ao enviar a mensagem.</p> <p>O <code>Supplier</code> comentado <code>enviarMensagem()</code> também seria um <code>Source</code>, mas no estilo funcional.</p>
Sink Consumidor	<p>É o componente recebe mensagens de um canal.</p> <p>É o destino final dos dados no fluxo.</p> <p>No modelo funcional do Spring, isso é um <code>Consumer<T></code>.</p>	<p>O bean <code>receberMensagem()</code> em <code>StreamConfiguration.java</code> é um <code>Sink</code>.</p> <p>Ele é um <code>Consumer</code> que "lê" a mensagem que chega.</p>
Channel Canal / Binding	<p>É a abstração pela qual os dados fluem.</p> <p>É o "cano" lógico que conecta o <code>Source</code> ao <code>Sink</code>.</p> <p>A sua aplicação não sabe o que é o Kafka ou o Pub/Sub; ela apenas sabe que está enviando ou recebendo de um "canal".</p>	<p>Os <i>bindings</i> (ligações) no <code>application.properties</code>, como <code>enviarMensagem-out-0</code> e <code>receberMensagem-in-0</code>, são a implementação prática dos canais.</p>
Binder Adaptador	<p>É o "tradutor" de infraestrutura.</p> <p>É a biblioteca que conecta os <i>Channels</i> abstratos da sua aplicação ao sistema de mensageria real.</p>	<p>É o "tradutor" que sabe como pegar uma mensagem do "canal" <code>enviarMensagem</code> e publicá-la em um tópico do Google Pub/Sub.</p>

Revisão do PubSub

1



Spring Boot Application

Classe de Acesso ao Tópico

Chave de Acesso ao Tópico

Bibliotecas do Projeto

application.properties x

Source History

```
1 spring.application.name=DR4_E4_A1
2 spring.datasource.url=jdbc:h2:file:./h2_databases/DR4_1
3 spring.datasource.driverClassName=org.h2.Driver
4 spring.datasource.username=sa
5 spring.datasource.password=password
6 spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
7 spring.h2.console.enabled=true
8 spring.jpa.hibernate.ddl-auto=update
9 spring.cloud.gcp.project-id=softwares-escalaveis
10 spring.cloud.gcp.credentials.location=classpath:softwares-escalaveis-a12fc70b8812.json
```

2

3

pom.xml [DR4_E4_A1] x

Source Graph Effective History

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <groupId>org.springframework.boot</groupId>
7         <artifactId>spring-boot-starter-parent</artifactId>
8         <version>3.3.2</version>
9         <relativePath/> <!-- lookup parent from repository -->
10    </parent>
11    <groupId>br.edu.infnet</groupId>
12    <artifactId>DR4_E4_A1</artifactId>
13    <version>0.0.1-SNAPSHOT</version>
14    <name>DR4_E4_A1</name>
15    <description>Google Cloud Message Test</description>
16    <properties>
17        <java.version>17</java.version>
18        <spring-cloud-gcp.version>5.4.3</spring-cloud-gcp.version>
19        <spring-cloud.version>2023.0.3</spring-cloud.version>
20    </properties>
```

4

```
StringMessageService.java x
Source History
1 package br.edu.infnet.pedidos.infra;
2
3 import com.google.cloud.spring.pubsub.core.PubSubTemplate;
4 import com.google.cloud.spring.pubsub.integration.AckMode;
5 import com.google.cloud.spring.pubsub.integration.inbound.PubSubInboundChannelAdapter;
6 import com.google.cloud.spring.pubsub.support.BasicAcknowledgeablePubsubMessage;
7 import com.google.cloud.spring.pubsub.support.GcpPubSubHeaders;
8 import org.slf4j.Logger;
9 import org.slf4j.LoggerFactory;
10 import org.springframework.beans.factory.annotation.Qualifier;
11 import org.springframework.context.annotation.Bean;
12 import org.springframework.integration.annotation.ServiceActivator;
13 import org.springframework.integration.channel.PublishSubscribeChannel;
14 import org.springframework.messaging.MessageChannel;
15 import org.springframework.messaging.handler.annotation.Header;
16 import org.springframework.stereotype.Service;
17
18 @Service
19 public class StringMessageService {
20
21     private static final Logger LOG = LoggerFactory.getLogger(StringMessageService.class);
```

```
23     @Bean
24     public MessageChannel inputMessageChannel() {
25         return new PublishSubscribeChannel();
26     }
27
28     @Bean
29     public PubSubInboundChannelAdapter inboundChannelAdapter(
30         @Qualifier("inputMessageChannel") MessageChannel messageChannel, PubSubTemplate pubSubTemplate) {
31
32         PubSubInboundChannelAdapter adapter = new PubSubInboundChannelAdapter(pubSubTemplate, "dr4 sub");
33         adapter.setOutputChannel(messageChannel);
34         adapter.setAckMode(AckMode.MANUAL);
35         adapter.setPayloadType(String.class);
36         return adapter;
37     }
38
39     @ServiceActivator(inputChannel = "inputMessageChannel")
40     public void messageReceiver(String payload,
41         @Header(GcpPubSubHeaders.ORIGINAL_MESSAGE) BasicAcknowledgeablePubsubMessage message) {
42
43         LOG.info("***** Mensagem Recebida ---> " + payload);
44         message.ack();
45     }
46 }
```

6

```
Dr4E4A1Application.java x StringMessageService.java x
Source History
1 package br.edu.infnet;
2
3 import com.google.cloud.spring.pubsub.core.PubSubTemplate;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.boot.CommandLineRunner;
8 import org.springframework.boot.SpringApplication;
9 import org.springframework.boot.autoconfigure.SpringBootApplication;
10
11 @SpringBootApplication
12 public class Dr4E4A1Application implements CommandLineRunner {
13
14     private static Logger LOG = LoggerFactory.getLogger(Dr4E4A1Application.class);
15
16     @Autowired
17     private PubSubTemplate pubSubTemplate;
18
19     public static void main(String[] args) {
20         SpringApplication.run(Dr4E4A1Application.class, args);
21     }
22
23     @Override
24     public void run(String... args) throws Exception {
25         String mensagem = "Hello DR4 World!";
26         pubSubTemplate.publish("dr4_topic", mensagem);
27         LOG.info("***** Mensagem Publicada ---> " + mensagem);
28     }
29 }
```



```
PedidoMessageService.java x
Source History
49 @ServiceActivator(inputChannel = "inputMessageChannel")
50 public void messageReceiver(Pedido payload,
51                             @Header(GcpPubSubHeaders.ORIGINAL_MESSAGE) ConvertedBasicAcknowledgeablePubsubMessage<Pedido> message) {
52
53     LOG.info("***** Mensagem Recebida ---> " + payload);
54     message.ack();
55 }
56
57 // @ServiceActivator(inputChannel = "inputMessageChannel")
58 // public void messageReceiver(Pedido payload,
59 //                             @Header(GcpPubSubHeaders.ORIGINAL_MESSAGE) BasicAcknowledgeablePubsubMessage message) {
60 //
61 //     LOG.info("***** Mensagem Recebida ---> " + payload);
62 //     message.ack();
63 // }
64 }
```


Implementação do Spring Cloud Stream

DR4_E6_A2_Stream/

├── pom.xml

├── src/

│ ├── main/java/br/edu/infnet/

│ │ ├── Dr4E6A2StreamApplication.java

│ │ └── StreamConfiguration.java

│ └── main/resources/

│ ├── application.properties

│ └── softwares-escalaveis-*.json

← classe principal (produtor)

← beans funcionais (consumer)

← configuração opinativa

← credenciais GCP

1

O projeto segue o modelo moderno do **Spring Cloud Stream** baseado em funções Java (Supplier, Function, Consumer), que substituem as antigas interfaces `@EnableBinding`.

pom.xml [DR4_E6_A2_Stream] x

```
1  <?xml version="1.0" encoding="UTF-8"?>
19  <dependencies>
20      <dependency>
21          <groupId>org.springframework.boot</groupId>
22          <artifactId>spring-boot-starter-web</artifactId>
23      </dependency>
24      <dependency>
25          <groupId>com.google.cloud</groupId>
26          <artifactId>spring-cloud-gcp-pubsub-stream-binder</artifactId>
27      </dependency>
28      <dependency>
29          <groupId>org.springframework.boot</groupId>
30          <artifactId>spring-boot-starter-test</artifactId>
31          <scope>test</scope>
32      </dependency>
33      <dependency>
34          <groupId>org.springframework.cloud</groupId>
35          <artifactId>spring-cloud-stream-binder-test</artifactId>
36          <version>3.2.10</version>
37          <scope>test</scope>
38      </dependency>
39  </dependencies>
40  <dependencyManagement>
41      <dependencies>
42          <dependency>
43              <groupId>org.springframework.cloud</groupId>
44              <artifactId>spring-cloud-dependencies</artifactId>
45              <version>${spring-cloud.version}</version>
46              <type>pom</type>
47              <scope>import</scope>
48          </dependency>
49          <dependency>
50              <groupId>com.google.cloud</groupId>
51              <artifactId>spring-cloud-gcp-dependencies</artifactId>
```

application.properties x

Source

History



```
1  spring.application.name=DR4_E6_A2_Stream
2  spring.cloud.gcp.project-id=softwares-escalaveis
3  spring.cloud.gcp.credentials.location=classpath:softwares-escalaveis-a12fc70b8812.json
4  #-----
5  spring.cloud.stream.gcp.pubsub.default.consumer.auto-create-resources=false
6  spring.cloud.stream.gcp.pubsub.default.producer.auto-create-resources=false
7  #spring.cloud.function.definition=receberMensagem;enviarMensagem
8  spring.cloud.function.definition=receberMensagem
9  #-----
10 spring.cloud.stream.bindings.enviarMensagem-out-0.destination=dr4_topic
11 spring.cloud.stream.gcp.pubsub.bindings.receberMensagem-in-0.consumer.subscription-name=dr4_sub
12
```

3

A

B

4

```
Dr4E6A2StreamApplication.java x
Source History
1 package br.edu.infnet;
2
3 import org.slf4j.Logger;
4 import org.slf4j.LoggerFactory;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.CommandLineRunner;
7 import org.springframework.boot.SpringApplication;
8 import org.springframework.boot.autoconfigure.SpringBootApplication;
9 import org.springframework.cloud.stream.function.StreamBridge;
10
11 @SpringBootApplication
12 public class Dr4E6A2StreamApplication implements CommandLineRunner {
13
14     private static final Logger LOG = LoggerFactory.getLogger(Dr4E6A2StreamApplication.class);
15
16     @Autowired
17     private StreamBridge streamBridge;
18
19     public static void main(String[] args) {
20         SpringApplication.run(Dr4E6A2StreamApplication.class, args);
21     }
22
23     @Override
24     public void run(String... args) throws Exception {
25         String mensagem = "Somente uma mensagem";
26         LOG.info("Enviando uma mensagem: " + mensagem);
27         streamBridge.send("enviarMensagem-out-0", mensagem);
28     }
29 }
```

StreamBridge é o componente que permite enviar mensagens para um binding

5

```
StreamConfiguration.java x
Source History
1 package br.edu.infnet;
2
3 import java.util.function.Consumer;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8 import org.springframework.messaging.Message;
9
10 @Configuration
11 public class StreamConfiguration {
12
13     private static final Logger LOG = LoggerFactory.getLogger(StreamConfiguration.class);
14
15     @Bean
16     public Consumer<Message<String>> receberMensagem() {
17         return mensagem -> {
18             LOG.info("receberMensagem: " + mensagem.getPayload());
19         };
20     }
21
22     // @Bean
23     // public Supplier<Message<String>> enviarMensagem() {
24     //     return () -> {
25     //         Message<String> mensagem = MessageBuilder.withPayload("mensagem-" + Math.random()).build();
26     //         LOG.info("enviarMensagem: " + mensagem.getPayload());
27     //         return mensagem;
28     //     };
29     // }
30 }
```

Output

Run (DR4_E6_A2_Stream) x

Run (DR4_E6_A2_Stream) x

```

2024-09-04T18:34:07.865-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:07.866-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:07.950-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:07.952-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
in 2785 ms
2024-09-04T18:34:08.851-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
gcp-messageria@softwares-escalaveis.iam.gserviceaccount.com
2024-09-04T18:34:08.851-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
ogleapis.com/auth/pubsub, https://www.googleapis.com/auth/spanner.admin, https://www.googleapis.com/auth/datastore, https://www.google
apis.com/auth/sqlservice.admin, https://www.googleapis.com/auth/devstorage.read_only, https://www.googleapis.com/auth/devstorage.read_write, https://www.googleapis.com/auth/cloudru
ntimeconfig, https://www.googleapis.com/auth/trace.append, https://www.googleapis.com/auth/cloud-platform, https://www.googleapis.com/auth/cloud-vision, https://www.googleapis.com/
auth/bigquery, https://www.googleapis.com/auth/monitoring.write]
2024-09-04T18:34:08.852-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.023-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
subscriber(s).
2024-09-04T18:34:10.123-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
integration.errorLogger} as a subscriber to the 'errorChannel' channel
2024-09-04T18:34:10.123-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
ber(s).
2024-09-04T18:34:10.124-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
ogger'
2024-09-04T18:34:10.147-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
 '/'
2024-09-04T18:34:10.149-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.149-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.232-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.259-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.260-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.260-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.260-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.260-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:10.285-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
bound.PubSubInboundChannelAdapter@ab2009f
2024-09-04T18:34:10.313-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
rocess running for 6.43)
2024-09-04T18:34:10.318-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
2024-09-04T18:34:11.647-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [
subscriber(s).
2024-09-04T18:34:12.642-03:00 INFO 514298 --- [DR4_E6_A2_Stream] [sub-subscriber1]

```

main] o.apache.catalina.core.StandardService : Starting service [Tomcat]

main] o.apache.catalina.core.StandardEngine : Starting Servlet engine:

main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embed

main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext

main] c.g.c.s.core.DefaultCredentialsProvider : Default credentials provi

main] c.g.c.s.core.DefaultCredentialsProvider : Scopes in use by default credentials: [https://www.go

main] c.g.c.s.a.c.GcpContextAutoConfiguration : The default project ID is softwares-escalaveis

main] o.s.c.s.m.DirectWithAttributesChannel : Channel 'DR4_E6_A2_Stream.receberMensagem-in-0' has 1

main] o.s.i.endpoint.EventDrivenConsumer : Adding {logging-channel-adapter:_org.springframework.

main] o.s.i.channel.PublishSubscribeChannel : Channel 'DR4_E6_A2_Stream.errorChannel' has 1 subscri

main] o.s.i.endpoint.EventDrivenConsumer : started bean '_org.springframework.integration.errorL

main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with context path

main] o.s.c.s.binder.DefaultBinderFactory : Creating binder: pubsub

main] o.s.c.s.binder.DefaultBinderFactory : Constructing binder child context for pubsub

main] o.s.c.s.binder.DefaultBinderFactory : Caching the binder: pubsub

main] o.s.c.stream.binder.BinderErrorChannel : Channel 'dr4_sub.errors' has 1 subscriber(s).

main] o.s.c.stream.binder.BinderErrorChannel : Channel 'dr4_sub.errors' has 0 subscriber(s).

main] o.s.c.stream.binder.BinderErrorChannel : Channel 'dr4_sub.errors' has 1 subscriber(s).

main] o.s.c.stream.binder.BinderErrorChannel : Channel 'dr4_sub.errors' has 2 subscriber(s).

main] g.c.s.p.i.i.PubSubInboundChannelAdapter : started com.google.cloud.spring.pubsub.integration.in

main] br.edu.infnet.Dr4E6A2StreamApplication : Started Dr4E6A2StreamApplication in 5.878 seconds (pr

main] br.edu.infnet.Dr4E6A2StreamApplication : Enviando uma mensagem: Somente uma mensagem

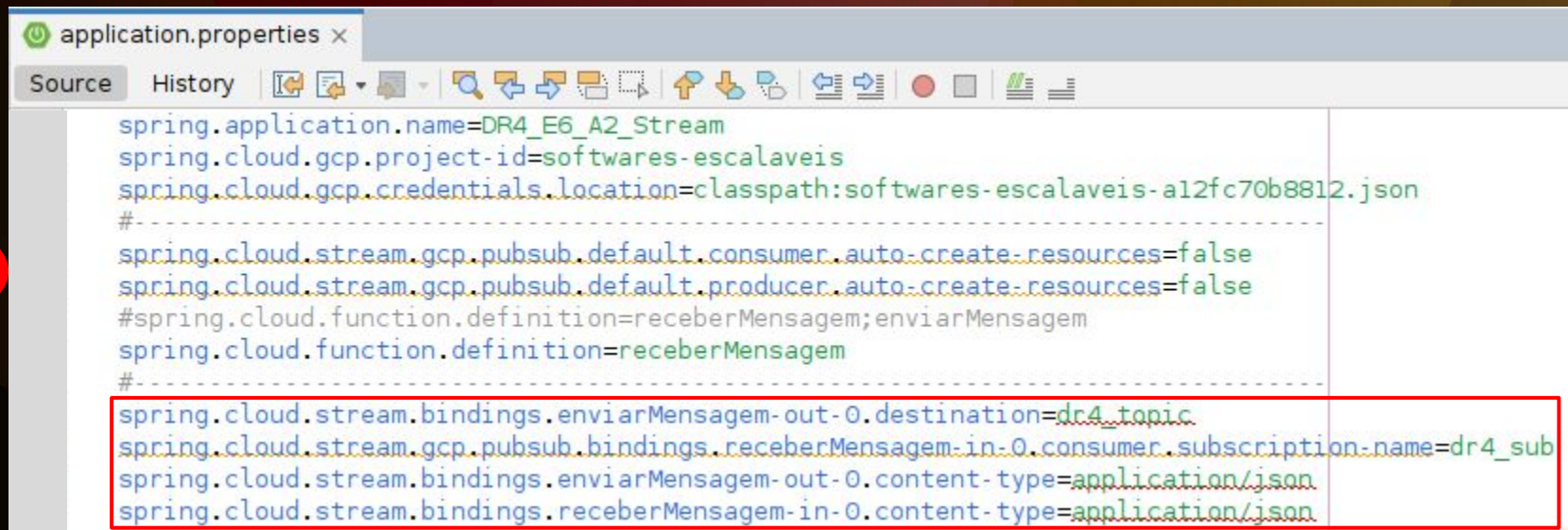
main] o.s.c.s.m.DirectWithAttributesChannel : Channel 'DR4_E6_A2_Stream.enviarMensagem-out-0' has 1

br.edu.infnet.StreamConfiguration : receberMensagem: Somente uma mensagem



```
PedidoCreatedEvent.java x
Source History
1 package br.edu.infnet;
2
3 import java.time.Instant;
4
5 public class PedidoCreatedEvent {
6
7     private String pedidoId;
8     private String clienteNome;
9     private Instant timestamp;
10
11     // Construtor vazio (obrigatório para o Jackson/JSON)
12     public PedidoCreatedEvent() {
13     }
14
15     // Construtor para facilitar a criação do evento
16     public PedidoCreatedEvent(String pedidoId, String clienteNome) {
17         this.pedidoId = pedidoId;
18         this.clienteNome = clienteNome;
19         this.timestamp = Instant.now();
20     }
21
22     // Getters e Setters
23     public String getPedidoId() {
24         return pedidoId;
25     }
26
27     public void setPedidoId(String pedidoId) {
28         this.pedidoId = pedidoId;
29     }
30
31     public String getClienteNome() {
32         return clienteNome;
33     }
34 }
```





```
application.properties x
Source History
spring.application.name=DR4_E6_A2_Stream
spring.cloud.gcp.project-id=softwares-escalaveis
spring.cloud.gcp.credentials.location=classpath:softwares-escalaveis-a12fc70b8812.json
#-----
spring.cloud.stream.gcp.pubsub.default.consumer.auto-create-resources=false
spring.cloud.stream.gcp.pubsub.default.producer.auto-create-resources=false
#spring.cloud.function.definition=receberMensagem;enviarMensagem
spring.cloud.function.definition=receberMensagem
#-----
spring.cloud.stream.bindings.enviarMensagem-out-0.destination=dr4_topic
spring.cloud.stream.gcp.pubsub.bindings.receberMensagem-in-0.consumer.subscription-name=dr4_sub
spring.cloud.stream.bindings.enviarMensagem-out-0.content-type=application/json
spring.cloud.stream.bindings.receberMensagem-in-0.content-type=application/json
```

6

```
StreamConfiguration.java x
Source History
1 package br.edu.infnet;
2
3 import java.util.function.Consumer;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8
9 @Configuration
10 public class StreamConfiguration {
11
12     private static final Logger LOG = LoggerFactory.getLogger(StreamConfiguration.class);
13
14     @Bean
15     public Consumer<PedidoCreatedEvent> receberMensagem() {
16         return evento -> {
17             LOG.info("RECEBIDO: Evento recebido! Payload: {}", evento.toString());
18         };
19     }
20 }
```

Dr4E6A2StreamApplication.java x

Source

History



```
8  import org.springframework.boot.autoconfigure.SpringBootApplication;
9  import org.springframework.cloud.stream.function.StreamBridge;
10
11  @SpringBootApplication
12  public class Dr4E6A2StreamApplication implements CommandLineRunner {
13
14      private static final Logger LOG = LoggerFactory.getLogger(Dr4E6A2StreamApplication.class);
15
16      @Autowired
17      private StreamBridge streamBridge;
18
19      public static void main(String[] args) {
20          SpringApplication.run(Dr4E6A2StreamApplication.class, args);
21      }
22
23      @Override
24      public void run(String... args) throws Exception {
25          PedidoCreatedEvent evento = new PedidoCreatedEvent("P-77890", "Maria Souza");
26          LOG.info("Enviando um evento: " + evento.getPedidoId());
27          streamBridge.send("enviarMensagem-out-0", evento);
28      }
29  }
```

```
PedidoEventService.java x
Source History
1 package br.edu.infnet;
2
3 import java.util.function.Consumer;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.cloud.stream.function.StreamBridge;
8 import org.springframework.context.annotation.Bean;
9 import org.springframework.stereotype.Service;
10
11 @Service
12 public class PedidoEventService {
13
14     private static final Logger LOG = LoggerFactory.getLogger(PedidoEventService.class);
15
16     @Autowired
17     private StreamBridge streamBridge;
18
19     public void publicarPedidoCriado(PedidoCreatedEvent evento) {
20         LOG.info("ENVIANDO: Publicando evento PedidoCreatedEvent: {}", evento.getPedidoId());
21         streamBridge.send("enviarMensagem-out-0", evento);
22     }
23
24     @Bean
25     public Consumer<PedidoCreatedEvent> receberMensagem() {
26         return evento -> {
27             LOG.info("RECEBIDO: Evento PedidoCreatedEvent! ID: {}", evento.getPedidoId());
28             LOG.info("Payload: {}", evento.toString());
29         };
30     }
31 }
```

