

Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



O que veremos:

1 – Introdução	página 3
1.1 – Utilização	página 3
2 – Princípios Básicos	página 4
2.1 – Dependências	página 4
2.2 – Estrutura básica de uma rota	página 4
2.3 – diretivas: from, to, process, exchange	página 4
2.4 – convertBodyTo e setHeader	página 5
2.5 – Predicate	página 6
2.6 – doTry(), doCatch(), choice(), when() e otherwise()	página 7
2.7 – Bean	página 8
2.8 – Utilitários/complementos	página 9
2.8.1 – fromJson / toJson / String	página 9
2.8.2 – java.lang.reflect.Type – conversão	página 9

Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



1 - Introdução

- **Apache Camel** é um framework open source focado em integrações;
- teve início em 2007;
- Apache 2 License;
- tem como foco e principal objetivo realizar integrações;
- <https://camel.apache.org/> ;
- Ilustração:



1.1 - Utilização

- roteamento;
- integração;
- transformação;
- ampla biblioteca;
- Enterprise Integration Patterns (EIPs);
- Domain-Specific Language (DSL);
- POJO model;
- fácil configuração;
- vasta comunidade ativa;
- dentre outras características...

Nosso foco

Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



2 – Princípios Básicos

2.1 – Dependências

→ dependências usuais para utilização em rotas:

```
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-http4</artifactId>
  <version>${camel.version}</version>
</dependency>
<dependency>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-jetty</artifactId>
  <version>${camel.version}</version>
</dependency>
```

2.2 – Estrutura básica de uma rota

```
public class RotaTeste extends RouteBuilder {

    // variáveis... etc...

    @Override
    public void configure() throws Exception {

        // implementação

    }

}
```

2.3 – diretivas: from, to, process, exchange

```
public class R02JettyToRest extends RouteBuilder {

    @Override
    public void configure() throws Exception {

        from(<rota1>)
            .process(new Processor() {
                @Override
                public void process(Exchange exchange) throws Exception {
                    log.info("---> Teste");
                    exchange.setProperty(<propriedade>, <valor>);
                }
            })
            .to(<rota2>);
    }

}
```

Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



- `exchange.getOut().setBody(valor de saída);`
- `exchange.getIn().getBody(String.class);`
- `exchange.setProperty(nomePropriedade, valorPropriedade);`
- `exchange.getProperty(nomePropriedade);`
- process – formas de implementação:

a)

```
.process(new PessoaProcessorJsonToXml())
```

```
public class PessoaProcessorJsonToXml implements Processor {  
  
    public void process(Exchange exchange) throws Exception {  
        // ... implementação  
    }  
  
}
```

b)

```
.process(new Processor() {  
    @Override  
    public void process(Exchange exchange) throws Exception {  
        ... // implementação  
    }  
})
```

2.4 – convertBodyTo e setHeader

```
from(directHttp)
```

```
.convertBodyTo(String.class)
```

```
.setHeader(Exchange.HTTP_METHOD, constant(org.apache.camel.component.http4.HttpMethods.GET))  
.to(rotaHttp4)
```

Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



2.5 – Predicate

→ validações e verificações;

→ Declaração/implementação:

```
private CustomPredicate customPredicate = new CustomPredicate();
```

```
public class CustomPredicate {  
    public Predicate getSimulacaoPredicate() {  
        Predicate validacao = new PredicateValidationLista();  
        return PredicateBuilder.and(validacao);  
    }  
}
```

```
public class PredicateValidationLista implements Predicate {  
  
    @Override  
    public boolean matches(Exchange exchange) {  
        //... implementação  
        return <true|false>;  
    }  
}
```

Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



an NTT DATA Company

2.6 – doTry(), doCatch(), choice(), when() e otherwise()

```
public void configure() throws Exception {
    from(<rota1>)
        .doTry()
            .to(<rota2>)
            .process(new Processor() {
                @Override
                public void process(Exchange exchange) throws Exception {
                    // ... implementação
                }
            })
        .choice()
            .when(<predicate>)
                .process(new Processor() {
                    @Override
                    public void process(Exchange exchange) throws Exception {
                        // ... implementação
                    }
                })
            .otherwise()
                .process(new Processor() {
                    @Override
                    public void process(Exchange exchange) throws Exception {
                        // ... implementação
                    }
                })
        .endChoice()
        .endDoTry()
        .doCatch(Exception.class)
            .process(new Processor() {
                @Override
                public void process(Exchange exchange) throws Exception {
                    // ... implementação
                }
            })
        .end();
}
```

Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



an NTT DATA Company

2.7 – Bean

→ implementações de diversas funcionalidades;

@Component

```
public class R05Bean extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from(rotaJetty8089)
            .to(directHttp);

        from(directHttp)
            .bean(GerarObjetoBean.class, gerarObjeto)
            .convertBodyTo(String.class)
            .to(rotaHttp4)
            .process(new Processor() {
                @Override
                public void process(Exchange exchange) throws Exception {
                    //... implementações
                }
            });
    }
}

public class GerarObjetoBean {

    public void gerarObjeto(Exchange exchange) {
        // ... implementação
    }
}
```


Pocket – Treinamento Apache Camel + Spring Boot

Ministrante: Thiago Hernandes de Souza – thiago.hernandes.souza@everis.com



2.8 – Utilitários/complementos

2.8.1 – fromJson / toJson / String

```
public void process(Exchange exchange) throws Exception {
    log.info("---> Isso veio da URL 8083: '{} ' ", exchange.getProperty(msgFrom8083));
    log.info("---> Isso será a saída para outro redirecionamento (String): '{} ' ",
        exchange.getIn().getBody(String.class));
    Gson g = new Gson();
    Funcionario[] p = g.fromJson(exchange.getIn().getBody(String.class),
        Funcionario[].class);
    log.info("---> JSON de objetos de Funcionários '{} ' ", g.toJson(p));
    exchange.getOut().setBody(valorOut8083);
}
```

2.8.2 – java.lang.reflect.Type - conversão

```
.process(new Processor() {
    @Override
    public void process(Exchange exchange) throws Exception {
        Gson gson = new Gson();
        java.lang.reflect.Type listaFuncionariosType = new
            TypeToken<ArrayList<Funcionario>>().getType();
        List<Funcionario> listaFuncionarios =
            gson.fromJson(exchange.getIn().getBody(String.class), listaFuncionariosType);

        listaFuncionarios.add((Funcionario)exchange.getProperty(novoObjeto));
        listaFuncionarios.forEach(i -> System.out.println(i.getNome()));
    }
});
```