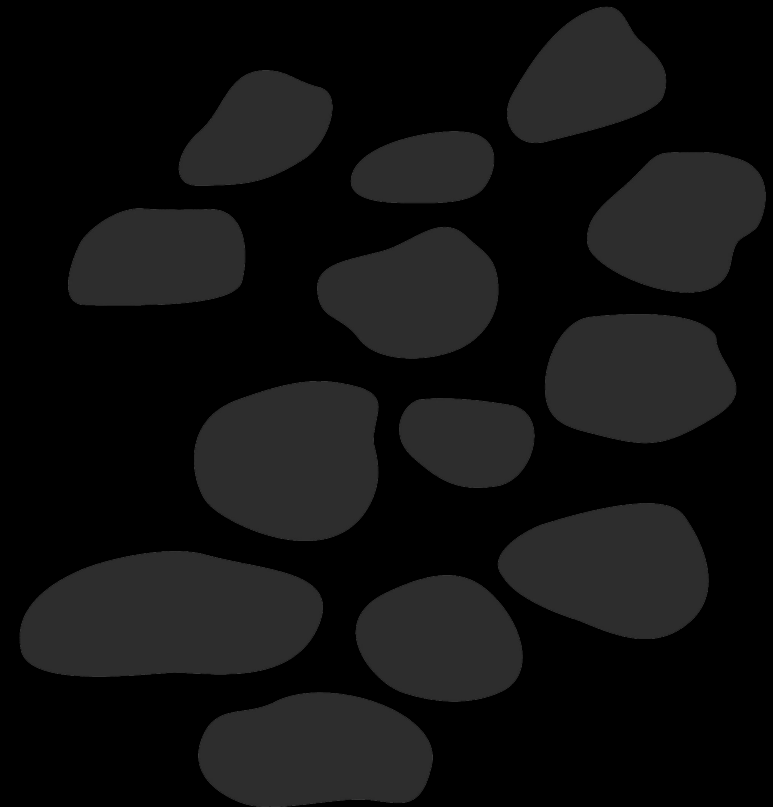


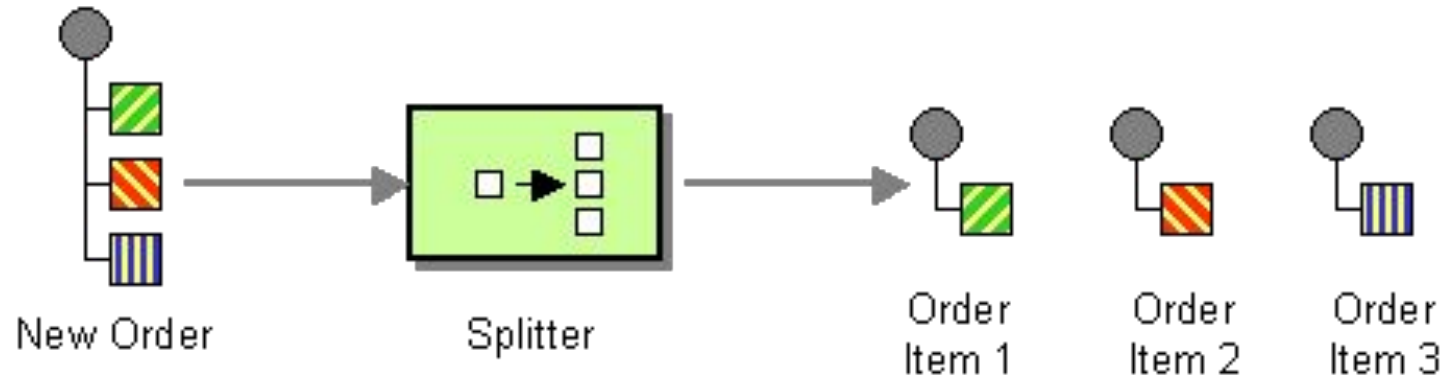
Camel Training

split & aggregate

Datum: 23.10.2020
Autor: Johannes Riegler



Splitter EIP



```
from("direct:splitUsingBody")
  .split(body())
    .to("mock:result")
  .end()
  .to("log:something")
```

returns (per default) the original exchange

```
from("direct:splitWithResult")
  .split(body()).streaming()
    .to(splitEndpoint)
  .end()
  .to(sumEndpoint);
```

sends splitted exchanges

doesn't wait until all submessages are read

Options

- streaming splits the messages on demand
- stopOnException stops further processing on exception
- parallelProcessing processing submessages in parallel
- timeout

```
.split (body())  
  .streaming()  
  .parallelProcessing()  
  .timeout (5000)
```

Exchange properties

- CamelSplitIndex counter that increases for each Exchange being split
- CamelSplitSize total count of submessages
- CamelSplitComplete true if all submessages split

```
.log( "${header.CamelSplitIndex}/${header.CamelSplitSize}/${header.CamelSplitComplete}")
```

What is splittable ?

- Out of the box

- `.split(body(String.class).tokenize("\n"))`
- XPath, XQuery, SQL or one of the Scripting Languages `.split(xpath("//foo/bar"))`
- Collection (e.g body or headers)

- Write custom Splitter

```
from ("direct:message" )
    .split ().method ("mySplitterBean", "spltMessage" )
    .to ("mock:result" );
```

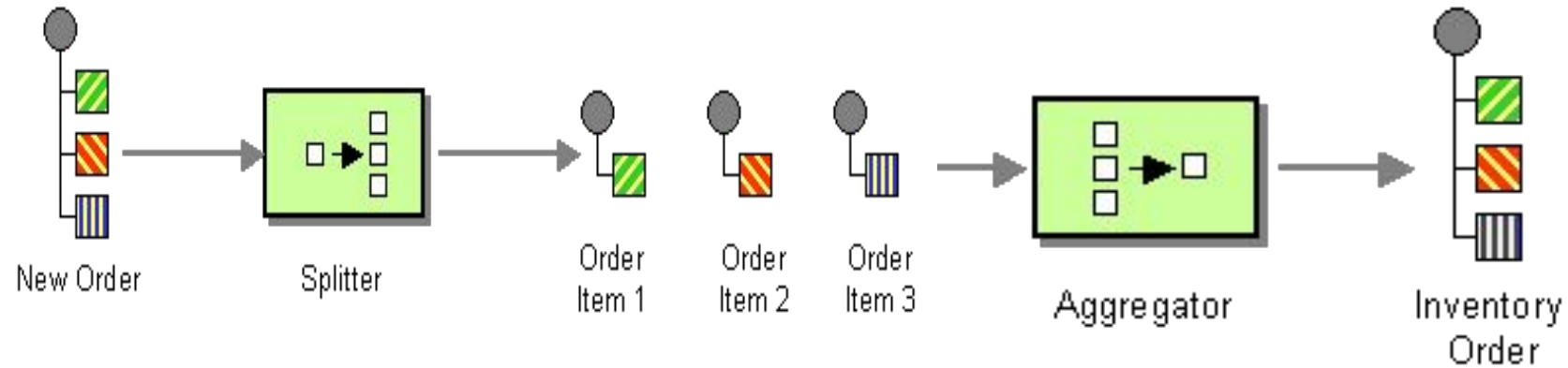
```
from ("direct:message" )
    .split ().method (new OrderSplitter() )
    .to ("mock:result" );
```

```
public class OrderSplitter {

    public static List<OrderToProducer> splitOrder(Order order) {
        List<OrderToProducer> result = new ArrayList<>();
        for (OrderItem item : order.getItems()) {
            result.add(new OrderToProducer(item, order.getPartnerId()));
        }
        return result;
    }
}
```

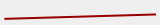
Example: best_08_1_splitter

Aggregate



```
.aggregate(new StringAggregationStrategy())  
    .constant(true)  
    .completionTimeout(500L);
```

my AggregationStrategy

```
class StringAggregationStrategy implements AggregationStrategy {  
  
    public Exchange aggregate(Exchange oldExchange, Exchange newExchange) {  
        if (oldExchange == null) {  
            return newExchange;  first exchange  
        }  
  
        String oldBody = oldExchange.getIn().getBody(String.class);  
        String newBody = newExchange.getIn().getBody(String.class);  
        oldExchange.getIn().setBody(oldBody + "+" + newBody);  
        return oldExchange;  
    }  
}
```

combine the new exchange
with the previously combined
exchanges

AggregationStrategy completion

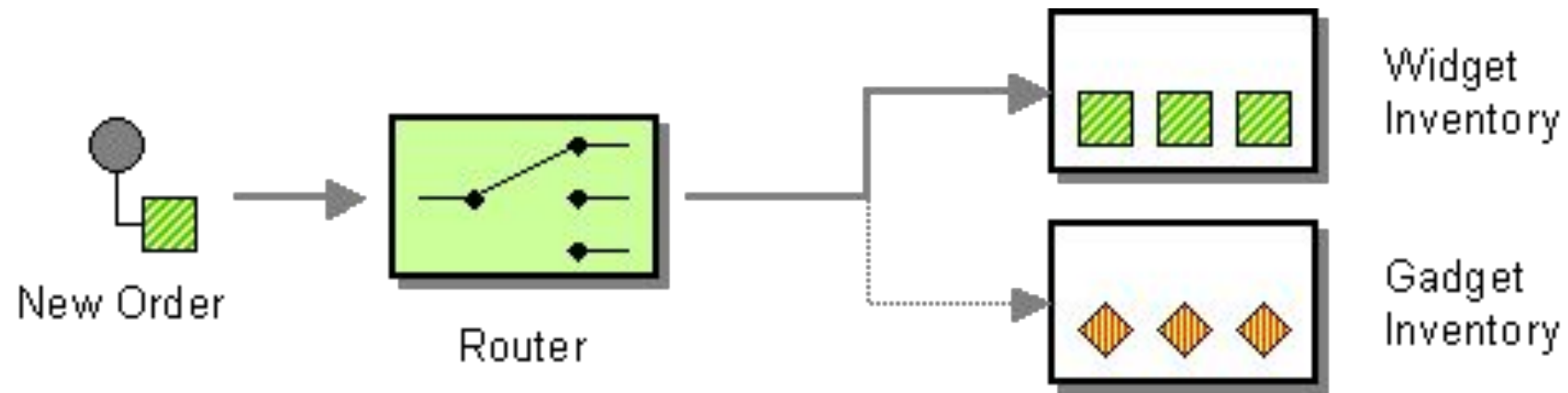
- There are several ways to determine whether the aggregation is complete. E.g:
 - `.completionTimeout(500L)`: Collect for a period of time
 - `.completionSize(3)`: Collect three items and return
 - `.completion(ex -> ex.getIn().getBody(String.class).contains("pear"))`: Predicate to determine completion

Split with Aggregation

```
.split (body (), AggregationStrategies.groupedBody ())  
  .process (exchange -> {  
    final ClientEntitySet camelTrainingSet =  
exchange.getProperty ("CamelTraining", ClientEntitySet.class);  
    exchange.getIn ().setBody (  
      new ExpandedCamelTrainingSet (exchange.getIn ().getBody (List.class), camelTrainingSet));  
    })  
  .end ()  
  .to ("direct:camelTrainingAggregated");
```

Example: best_08_2_aggregate

Choice



when also known as "if"

"Simple" language

```
public void configure () {  
    from ("direct:a")  
        .choice ()  
            .when (simple ("${header.foo} == 'bar'"))  
                .to ("direct:b")  
            .when (simple ("${header.foo} == 'cheese'"))  
                .to ("direct:c")  
            .otherwise ()  
                .to ("direct:d")  
        .end ();  
}
```

expects Predicate object

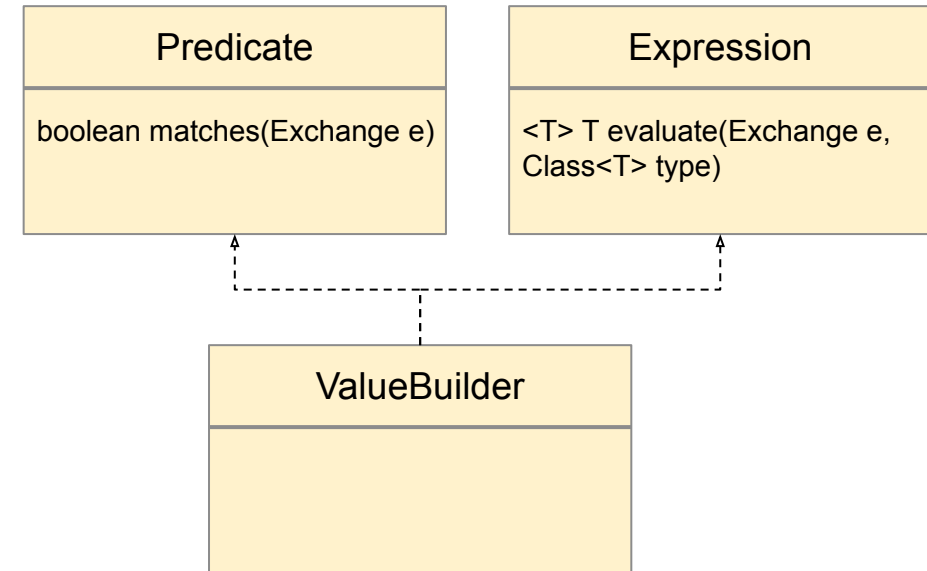
Predicate

boolean matches(Exchange e)

Simple Language

- One of currently 26 [Expression Languages](#)
- Used in other components, like log `.log("Start with body: ${body}")`
- Access it with `simple()` in RouteBuilders

```
public ValueBuilder simple(String value)
```



Simple Language Examples

"Sending \${body} to file"	Writes message body as string
"CamelSplitIndex: \${header.CamelSplitIndex}"	Writes exchange header with name "CamelSplitIndex"
"Body is \${body.class}"	Writes the class of the Body
"\${body.code} == 1"	Invokes getCode() on the body-object of an exchange and returns true if it equals 1.
"Code is \${body.getCode()}"	Invokes getCode() on the body and writes the value
"\${bean:idGenerator.nextId}"	Looks up a registered bean with name "idGenerator" and invokes the method nextId().

See Variables in [Simple-Language](#) and [Apache OGNL](#)

Example: best_09_choice