**Apache Camel Multicast Examples**

In computer networking, multicast is group communication where information is addressed to a group of destination computers simultaneously. [Camel multicast](http://camel.apache.org/multicast.html) pattern borrows the same terminology, it routes a message to a number of endpoints at the same time. The main difference between the Multicast and [Splitter](http://javarticles.com/2015/05/apache-camel-splitter-example.html) is that Splitter will split the message into several pieces but the Multicast will not modify the request message.  
If you note, multicast is similar to [Recipient List](http://camel.apache.org/recipient-list.html), the main difference is, in case of multicast the recipient list are already known thus is static and fixed whereas in case of recipient list it is dynamic.

Before we start with the example, let’s look into the setup details.

This example uses the following frameworks:

1. [Maven](http://maven.apache.org/) 3.2.3
2. [Apache Camel](http://camel.apache.org/) 2.15.1
3. [Spring](http://projects.spring.io/spring-framework/) 4.1.5.RELEASE
4. [Eclipse](http://www.eclipse.org/) as the IDE, version Luna 4.4.1.

**Dependencies**

We are just relying camel’s core components and the logger component so our pom.xml consists of:

1. camel-core
2. slf4j-api
3. slf4j-log4j12
4. camel-stream
5. spring-context
6. camel-spring

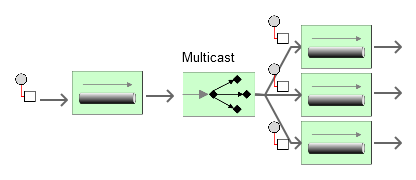
*pom.xml:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | <project xmlns="<http://maven.apache.org/POM/4.0.0>" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"      xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> <http://maven.apache.org/xsd/maven-4.0.0.xsd>">      <modelVersion>4.0.0</modelVersion>      <groupId>com.javarticles.camel</groupId>      <artifactId>camelHelloWorld</artifactId>      <version>0.0.1-SNAPSHOT</version>      <dependencies>          <dependency>              <groupId>org.apache.camel</groupId>              <artifactId>camel-core</artifactId>              <version>2.15.1</version>          </dependency>          <dependency>              <groupId>org.apache.camel</groupId>              <artifactId>camel-stream</artifactId>              <version>2.15.1</version>          </dependency>          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>slf4j-api</artifactId>              <version>1.7.12</version>          </dependency>          <dependency>              <groupId>org.slf4j</groupId>              <artifactId>slf4j-log4j12</artifactId>              <version>1.7.12</version>          </dependency>          <dependency>              <groupId>org.springframework</groupId>              <artifactId>spring-context</artifactId>              <version>4.1.5.RELEASE</version>          </dependency>          <dependency>              <groupId>org.apache.camel</groupId>              <artifactId>camel-spring</artifactId>              <version>2.15.1</version>          </dependency>      </dependencies>  </project> |

**Multicast Example**

Using multicast we can route the same message to many endpoints and have them processed in different ways. This can be done either sequentially in the same thread or using parallel processing.  
Let’s first see an example of the default one, that is, message being sent to multiple destinations in the same thread.

[](http://javarticles.com/wp-content/uploads/2015/05/camel_multicast.png)

*Multicast*

To use multicast call multicast() in the Java DSL and then pass the destinations to the to() method . Below we consume message from direct:start and is multicast to three different destinations direct:a, direct:b anddirect:c.

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3 | from("direct:start")  .multicast()  .to("direct:a", "direct:b", "direct:c"); |

We can achieve the same with multiple calls to to() method in sequence.

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | from("direct:start")  .multicast()  .to("direct:a")  .to("direct:b")  .to("direct:c"); |

The message sent to each destination is further processed using MyBean. At each destination we will modify the body, append the thread name to it and then finally send it to console. The message sent to first destination is modified by MyBean.addFirst() which appends ‘first destination’ to the body. Likewise we modify the messages at the other two destinations using MyBean.addSecond() and MyBean.addThird().

*MyBean:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | package com.javarticles.camel;    public class MyBean {        public String addFirst(String body) {          return body + " first destination";      }        public String addSecond(String body) {          return body + " second destination";      }        public String addThird(String body) {          return body + " third destination";      }  } |

*CamelMulticastExample:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | package com.javarticles.camel;    import org.apache.camel.CamelContext;  import org.apache.camel.ProducerTemplate;  import org.apache.camel.builder.RouteBuilder;  import org.apache.camel.impl.DefaultCamelContext;  import org.apache.camel.util.jndi.JndiContext;    public class CamelMulticastExample {      public static final void main(String[] args) throws Exception {          JndiContext jndiContext = new JndiContext();          jndiContext.bind("myBean", new MyBean());          CamelContext camelContext = new DefaultCamelContext(jndiContext);          try {              camelContext.addRoutes(new RouteBuilder() {                  public void configure() {                      from("direct:start")                      .multicast()                      .to("direct:a", "direct:b", "direct:c");                        from("direct:a")                      .to("bean:myBean?method=addFirst")                      .setBody(simple("body: ${body}, thread: ${threadName}"))                      .to("stream:out");                        from("direct:b")                      .to("bean:myBean?method=addSecond")                      .setBody(simple("body: ${body}, thread: ${threadName}"))                      .to("stream:out");                        from("direct:c")                      .to("bean:myBean?method=addThird")                      .setBody(simple("body: ${body}, thread: ${threadName}"))                      .to("stream:out");                  }              });              ProducerTemplate template = camelContext.createProducerTemplate();              camelContext.start();              template.sendBody("direct:start", "Multicast");          } finally {              camelContext.stop();          }      }  } |

If you note messages are sent to all the three destinations are in the same thread.  If you want parallel processing, we will have to further configure it which we see in our next section.

*Output:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3 | body: Multicast first destination, thread: main  body: Multicast second destination, thread: main  body: Multicast third destination, thread: main |

**Multicast with Parallel Processing**

If you want to add parallel processing to the above example, simply add parallelProcessing(). By default thread pool size of 10 is used, you can configure your own size of pool by adding your own ExecutorService usingexecutorService.

*CamelMulticastParallelProcessingExample:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | package com.javarticles.camel;    import java.util.concurrent.ExecutorService;  import java.util.concurrent.Executors;    import org.apache.camel.CamelContext;  import org.apache.camel.ProducerTemplate;  import org.apache.camel.builder.RouteBuilder;  import org.apache.camel.impl.DefaultCamelContext;  import org.apache.camel.util.jndi.JndiContext;    public class CamelMulticastParallelProcessingExample {      public static final void main(String[] args) throws Exception {          JndiContext jndiContext = new JndiContext();          jndiContext.bind("myBean", new MyBean());          CamelContext camelContext = new DefaultCamelContext(jndiContext);          try {              camelContext.addRoutes(new RouteBuilder() {                  public void configure() {                      ExecutorService executor = Executors.newFixedThreadPool(12);                      from("direct:start")                      .multicast()                      .parallelProcessing()                      .executorService(executor)                      .to("direct:a", "direct:b", "direct:c");                        from("direct:a")                      .to("bean:myBean?method=addFirst")                      .setBody(simple("body: ${body}, thread: ${threadName}"))                      .to("stream:out");                        from("direct:b")                      .to("bean:myBean?method=addSecond")                      .setBody(simple("body: ${body}, thread: ${threadName}"))                      .to("stream:out");                        from("direct:c")                      .to("bean:myBean?method=addThird")                      .setBody(simple("body: ${body}, thread: ${threadName}"))                      .to("stream:out");                  }              });              ProducerTemplate template = camelContext.createProducerTemplate();              camelContext.start();              template.sendBody("direct:start", "Multicast");          } finally {              camelContext.stop();          }      }  } |

*Output:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3 | body: Multicast third destination, thread: pool-1-thread-3  body: Multicast first destination, thread: pool-1-thread-1  body: Multicast second destination, thread: pool-1-thread-2 |

**Multicast Example Using Spring**

We can also achieve the same using spring using <multicast>.

*applicationContext.xml:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | <?xml version="1.0" encoding="UTF-8"?>    <beans xmlns="<http://www.springframework.org/schema/beans>"      xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"      xsi:schemaLocation="  <http://www.springframework.org/schema/beans> <http://www.springframework.org/schema/beans/spring-beans.xsd>  <http://camel.apache.org/schema/spring> <http://camel.apache.org/schema/spring/camel-spring.xsd>         ">      <camelContext xmlns="<http://camel.apache.org/schema/spring>">          <route>              <from uri="direct:start" />              <multicast>                  <to uri="direct:a" />                  <to uri="direct:b" />                  <to uri="direct:c" />              </multicast>          </route>          <route>              <from uri="direct:a" />              <to uri="bean:myBean?method=addFirst"/>              <to uri="stream:out"/>          </route>          <route>              <from uri="direct:b" />              <to uri="bean:myBean?method=addSecond"/>              <to uri="stream:out"/>          </route>          <route>              <from uri="direct:c" />              <to uri="bean:myBean?method=addThird"/>              <to uri="stream:out"/>          </route>      </camelContext>      <bean id="myBean" class="com.javarticles.camel.MyBean"/>  </beans> |

*Output:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3 | Multicast first destination  Multicast second destination  Multicast third destination |

**Multicast and Pipeline**

By default, all the endpoints in a route are in pipeline so we don’t have to use pipeline() method.  
For example, below route is with pipeline

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | from("direct:start")  .pipeline()  .to("bean:myBean?method=addFirst")  .to("bean:stringUtils?method=upperCase")  .to("stream:out"); |

And the same can be defined without pipeline(), both the routes are same.

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4 | from("direct:start")  .to("bean:myBean?method=addFirst")  .to("bean:stringUtils?method=upperCase")  .to("stream:out"); |

In case of multicast, we may need to use pipeline when one of its endpoint is composed of other endpoints in sequence. For example, the first destination is composed of two bean endpoints. Each endpoint work on the payload and transform. The first one appends a string and the second one converts the string to uppercase. If we don’t use the pipeline, event the bean endpoint will be treated as one of the multicast destination so the bean that converts the payload to uppercase will end up converting the original payload rather than the appended payload.

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | from("direct:start")  .multicast()  .pipeline()  .to("bean:myBean?method=addFirst")  .to("bean:stringUtils?method=upperCase")  .to("stream:out")  .end()  .... |

*CamelMulticastPipelineExample:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | package com.javarticles.camel;    import org.apache.camel.CamelContext;  import org.apache.camel.ProducerTemplate;  import org.apache.camel.builder.RouteBuilder;  import org.apache.camel.impl.DefaultCamelContext;  import org.apache.camel.util.jndi.JndiContext;    public class CamelMulticastPipelineExample {      public static final void main(String[] args) throws Exception {          JndiContext jndiContext = new JndiContext();          jndiContext.bind("myBean", new MyBean());          jndiContext.bind("stringUtils", new StringUtils());          CamelContext camelContext = new DefaultCamelContext(jndiContext);          try {              camelContext.addRoutes(new RouteBuilder() {                  public void configure() {                      from("direct:start")                      .multicast()                      .pipeline()                      .to("bean:myBean?method=addFirst")                      .to("bean:stringUtils?method=upperCase")                      .to("stream:out")                      .end()                        .pipeline()                      .to("bean:myBean?method=addSecond")                      .to("bean:stringUtils?method=upperCase")                      .to("stream:out")                      .end()                        .pipeline()                      .to("bean:myBean?method=addThird")                      .to("bean:stringUtils?method=upperCase")                      .to("stream:out")                      .end()                      .end()                      .setBody(simple("Final Output: ${body}"))                      .to("stream:out");                  }              });              ProducerTemplate template = camelContext.createProducerTemplate();              camelContext.start();              template.sendBody("direct:start", "Multicast");          } finally {              camelContext.stop();          }      }  } |

The final output from the multicast is the the latest reply message and it discards any earlier replies. If you want a different aggregation strategy to include even other reply messages, you need to create your ownAggregationStrategy. We will see in our next section how to set our own custom aggregation strategy.

*Output:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4 | MULTICAST FIRST DESTINATION  MULTICAST SECOND DESTINATION  MULTICAST THIRD DESTINATION  Final Output: MULTICAST THIRD DESTINATION |

**Multicast with a Custom Aggregation Strategy**

In this example, we will come up with our own aggregation strategy. Class JoinReplyAggregationStrategyimplements AggregationStrategy, it concatenates the inbound exchange messages. We configure the aggregation strategy using aggregationStrategy() method.

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | from("direct:start")  .multicast()  .aggregationStrategy(new JoinReplyAggregationStrategy())  .to("direct:a", "direct:b", "direct:c")  ... |

*JoinReplyAggregationStrategy:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package com.javarticles.camel;    import org.apache.camel.Exchange;  import org.apache.camel.processor.aggregate.AggregationStrategy;    public class JoinReplyAggregationStrategy implements AggregationStrategy {        public Exchange aggregate(Exchange exchange1, Exchange exchange2) {          if (exchange1 == null) {              return exchange2;          } else {              String body1 = exchange1.getIn().getBody(String.class);              String body2 = exchange2.getIn().getBody(String.class);              String merged = (body1 == null) ? body2 : body1 + "," + body2;              exchange1.getIn().setBody(merged);              return exchange1;          }      }    } |

Thus, replies from each of the multicast destinations are concatenated into one response.

*CamelMulticastAggregationExample:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | package com.javarticles.camel;    import org.apache.camel.CamelContext;  import org.apache.camel.ProducerTemplate;  import org.apache.camel.builder.RouteBuilder;  import org.apache.camel.impl.DefaultCamelContext;  import org.apache.camel.util.jndi.JndiContext;    public class CamelMulticastAggregationExample {      public static final void main(String[] args) throws Exception {          JndiContext jndiContext = new JndiContext();          jndiContext.bind("myBean", new MyBean());          CamelContext camelContext = new DefaultCamelContext(jndiContext);          try {              camelContext.addRoutes(new RouteBuilder() {                  public void configure() {                      from("direct:start")                      .multicast()                      .aggregationStrategy(new JoinReplyAggregationStrategy())                      .to("direct:a", "direct:b", "direct:c")                      .end()                      .to("stream:out");                        from("direct:a")                      .to("bean:myBean?method=addFirst");                        from("direct:b")                      .to("bean:myBean?method=addSecond");                        from("direct:c")                      .to("bean:myBean?method=addThird");                  }              });              ProducerTemplate template = camelContext.createProducerTemplate();              camelContext.start();              template.sendBody("direct:start", "Multicast");          } finally {              camelContext.stop();          }      }  } |

*Output:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1 | Multicast first destination,Multicast second destination,Multicast third destination |

**Multicast Exception Example**

By default, the multicast will continue sending messages to destinations even if one fails. In this example direct:aendpoint will fail if the message sent is a non-integer. We pass it through a processor that makes sure that the payload is an integer else throws NumberFormatException.

You can configure the routing to handle exceptions.

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | onException(Exception.class)  .handled(true)  .to("log:onException")  .transform(constant("Exception thrown. Stop route"))   .to("stream:out"); |

Our number validator processor.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | package com.javarticles.camel;    import org.apache.camel.Exchange;  import org.apache.camel.Processor;    public class NumberPayloadValidator implements Processor {        public void process(Exchange exchange) throws Exception {          Object body = exchange.getIn().getBody();          if (body instanceof Number) {              return;          }          String bodyAsString = body.toString();          Integer.parseInt(bodyAsString);      }    } |

direct:a endpoint is linked with the above processor.

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4 | from("direct:a")  .process(numberPayloadValidator)  .transform(simple("Received ${body} from direct:a"))  .to("stream:out"); |

We will send two different inputs. The first one would be an integer 1 and the second one would be integer in word form ‘one’.

*CamelMulticastExceptionExample:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | package com.javarticles.camel;    import org.apache.camel.CamelContext;  import org.apache.camel.ProducerTemplate;  import org.apache.camel.builder.RouteBuilder;  import org.apache.camel.impl.DefaultCamelContext;  import org.apache.camel.util.jndi.JndiContext;    public class CamelMulticastExceptionExample {      public static final void main(String[] args) throws Exception {          JndiContext jndiContext = new JndiContext();          jndiContext.bind("myBean", new MyBean());          CamelContext camelContext = new DefaultCamelContext(jndiContext);          final NumberPayloadValidator numberPayloadValidator = new NumberPayloadValidator();          try {              camelContext.addRoutes(new RouteBuilder() {                  public void configure() {                      onException(Exception.class)                      .handled(true)                      .to("log:onException")                      .transform(constant("Exception thrown. Stop route"))                      .to("stream:out");                        from("direct:start")                      .multicast()                      .to("direct:a", "direct:b")                      .end()                      .transform(simple("Final Output after multicast ${body}"))                      .to("stream:out");                        from("direct:a")                      .process(numberPayloadValidator)                      .transform(simple("Received ${body} from direct:a"))                      .to("stream:out");                        from("direct:b")                      .transform(simple("Received ${body} from direct:b"))                      .to("stream:out");                  }              });              ProducerTemplate template = camelContext.createProducerTemplate();              camelContext.start();              template.sendBody("direct:start", "1");              template.sendBody("direct:start", "one");          } finally {              camelContext.stop();          }      }  } |

As you can see when the input is ‘one’ direct:a routing fails but direct:b routing works fine.

*Output:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Received 1 from direct:a  Received 1 from direct:b  Final Output after multicast Received 1 from direct:b  11:32| INFO | MarkerIgnoringBase.java 95 | Exchange[ExchangePattern: InOnly, BodyType: String, Body: one]  Exception thrown. Stop route  Received one from direct:b |

**Multicast StopOnException Example**

*CamelMulticastOnStopExceptionExample:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | package com.javarticles.camel;    import org.apache.camel.CamelContext;  import org.apache.camel.ProducerTemplate;  import org.apache.camel.builder.RouteBuilder;  import org.apache.camel.impl.DefaultCamelContext;  import org.apache.camel.util.jndi.JndiContext;    public class CamelMulticastOnStopExceptionExample {      public static final void main(String[] args) throws Exception {          JndiContext jndiContext = new JndiContext();          jndiContext.bind("myBean", new MyBean());          CamelContext camelContext = new DefaultCamelContext(jndiContext);          final NumberPayloadValidator numberPayloadValidator = new NumberPayloadValidator();          try {              camelContext.addRoutes(new RouteBuilder() {                  public void configure() {                      onException(Exception.class)                      .handled(true)                      .to("log:onException")                      .transform(constant("Exception thrown. Stop route"))                      .to("stream:out");                        from("direct:start")                      .multicast()                      .stopOnException()                      .to("direct:a", "direct:b")                      .end()                      .transform(simple("Final Output after multicast ${body}"))                      .to("stream:out");                        from("direct:a")                      .process(numberPayloadValidator)                      .transform(simple("Received ${body} from direct:a"))                      .to("stream:out");                        from("direct:b")                      .transform(simple("Received ${body} from direct:b"))                      .to("stream:out");                  }              });              ProducerTemplate template = camelContext.createProducerTemplate();              camelContext.start();              template.sendBody("direct:start", "1");              template.sendBody("direct:start", "one");          } finally {              camelContext.stop();          }      }  } |

As you can see when the input is ‘one’ direct:a routing fails which in turn stops other routes in the multicast.

*Output:*

[?](http://javarticles.com/2015/05/apache-camel-multicast-examples.html)

|  |  |
| --- | --- |
| 1  2  3  4  5 | Received 1 from direct:a  Received 1 from direct:b  Final Output after multicast Received 1 from direct:b  11:33| INFO | MarkerIgnoringBase.java 95 | Exchange[ExchangePattern: InOnly, BodyType: String, Body: one]  Exception thrown. Stop route |