POLE STAR
www.polestarglobal.com

Purplefinder Enterprise Platform Design Patterns with Camel

Peter Potts

10th November 2010

Resources

- Reference Book: Enterprise Integration Patterns by Gregor Hohpe & Bobby Woolf

- Manning Book: Camel in Action
- Apache Camel User Guide & downloads: http://camel.apache.org
- 2 Camel example applications in PEP R2: http://repository.enterprise.purplefinder.com
- Available on public Maven repositories.

Design Pattern

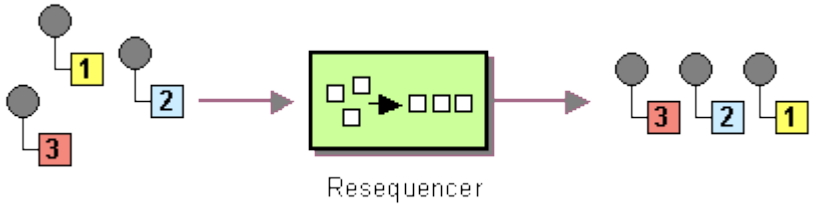
- A general reusable solution to a commonly occurring problem.
- Large specialized problem modularized into collection of small common problems.
- Anti-pattern: Bad practice.
- Example: Builder Pattern.

  Abstracts out the construction steps of an object.

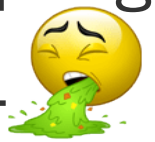
  Use StringBuilder to build a String.

Enterprise Integration Pattern (EIP)

- Message-based technology-independent design patterns focused on integrating services, applications and transport protocols.
- There are 65 well documented patterns that cover a wide range of problems.
- See http://www.eaipatterns.com/toc.html.
- Example: Resequencer

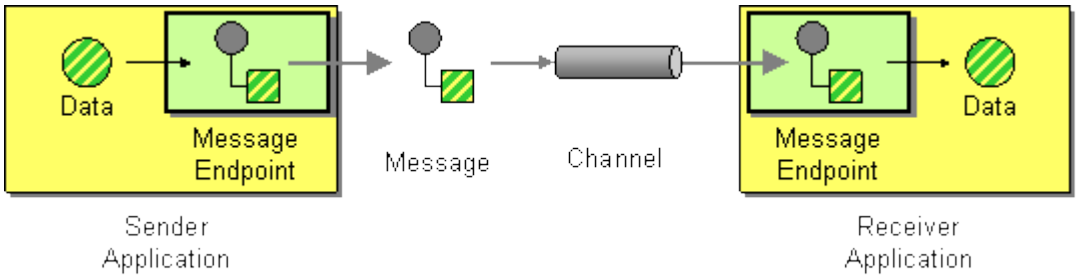


Apache Camel

- Open source integration framework based on Enterprise Integration Patterns.
- Domain Specific Language for routing and transforming messages between endpoints.
- DSL available in Spring, Java and Scala.
- Spring DSL in XML  .
- Scala DSL is incomplete and therefore unfortunately useless until completed.
- However, Java DSL can be happily used in Scala.

Camel Endpoint

- Message Endpoint is an EIP.



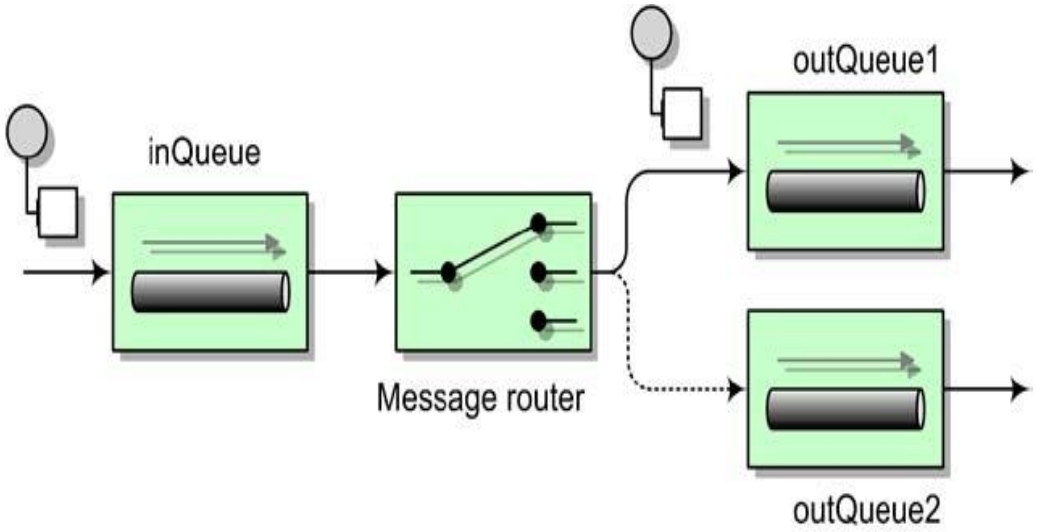
- Defines the interface between modules.
- Specified by a URI.
- A Component is an Endpoint factory.

URI Examples

- “jms:queue:destinationName”
- “file:src/data?noop=true&sortBy=file:name”
- “direct:name”
- “http://hostName:port”
- “log:loggingCategory?showBody=true”
- “mock:name”
- “bean:beanName?method=someMethod”
- “mail://user@host:port”
- “jmx://platform?options”
- “quartz://groupName/timerName”
- “rss:uri”
- “smpp://user@host:port?options”
- “jdbc:dataSourceName?options”
- “ldap:host:port?options”

Camel Route

- The wiring from one endpoint to any number of other endpoints.

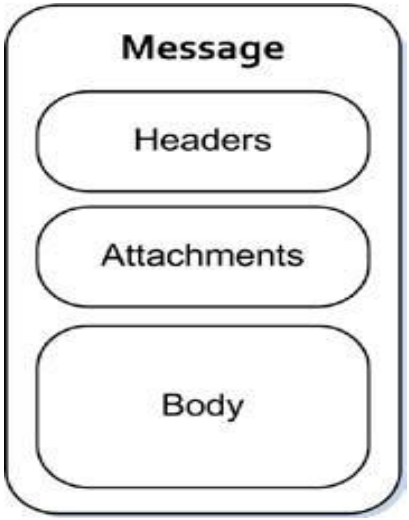


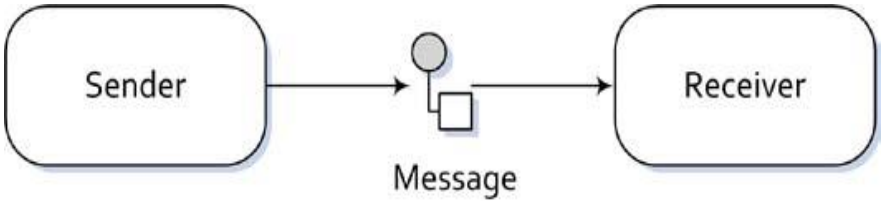
RouteBuilder

- Use a RouteBuilder to construct a Camel route.

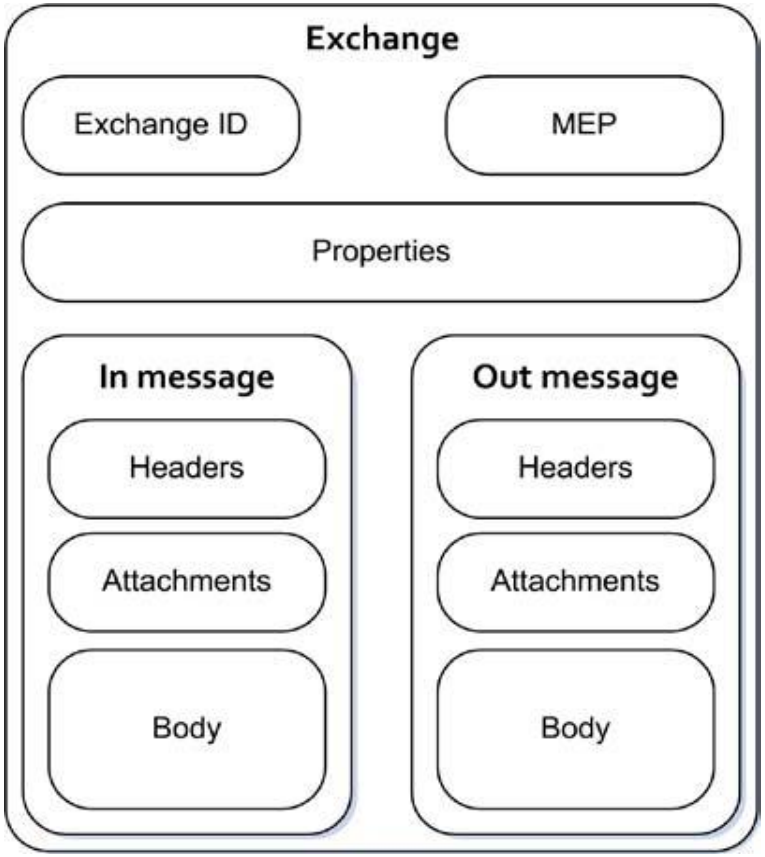
```
new RouteBuilder {
   def configure() {
    from("file:data/inbox")
        .filter()
        .xpath("/report/type = 'position'")
        .bean(classOf[SomeTrait])
        .to("jms:queue:positionReport")
   }
}
```

One-Way Message

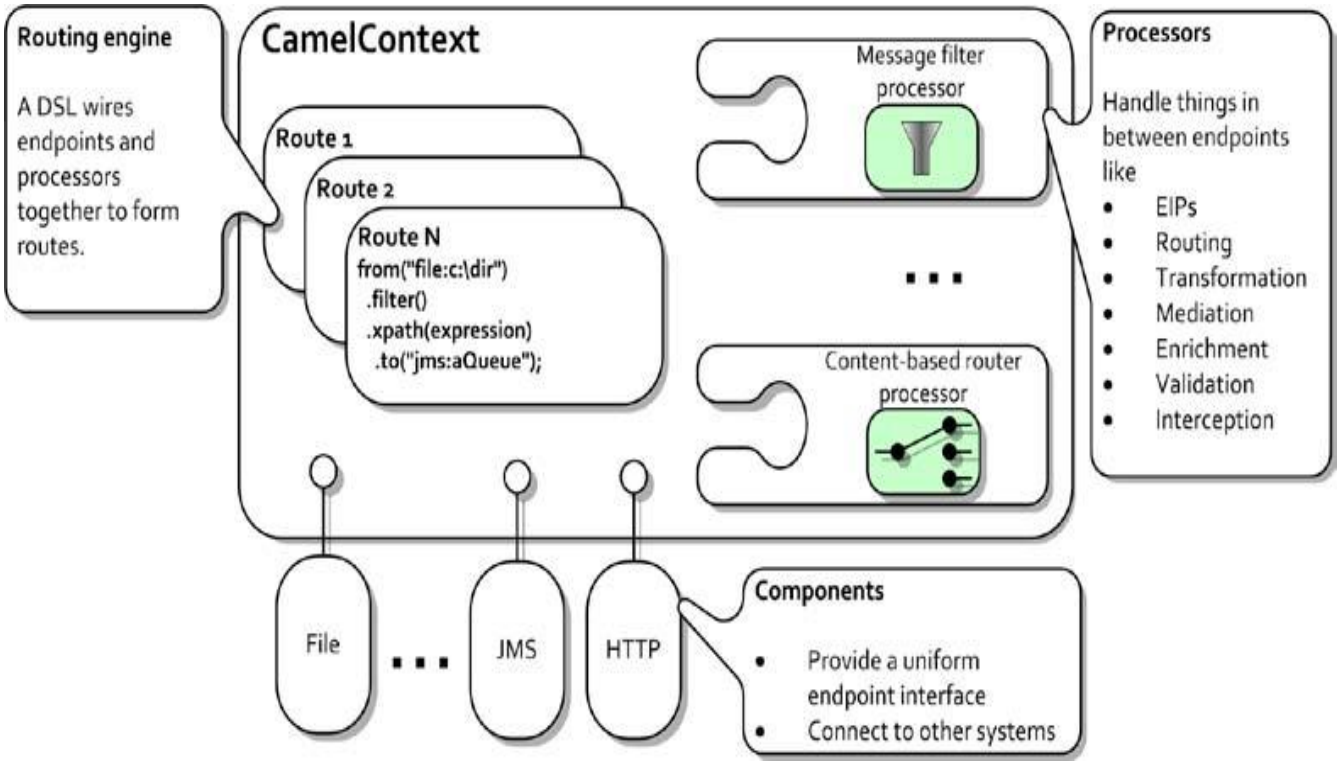




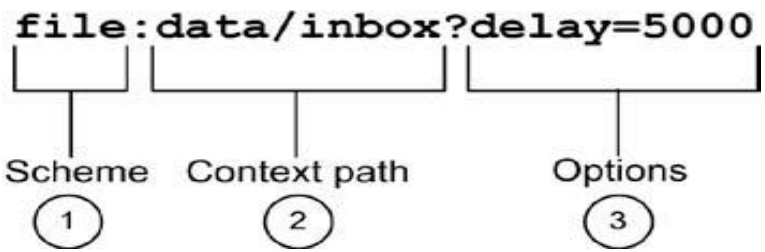
Two-Way Exchange



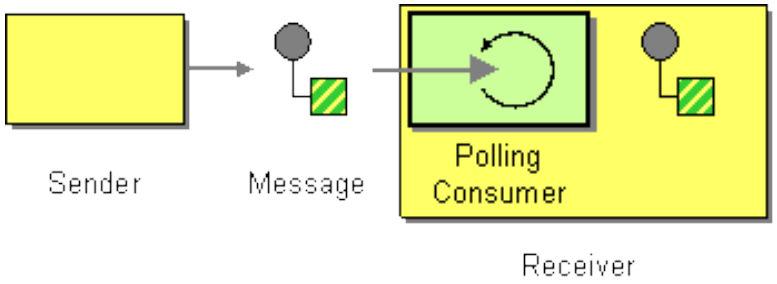
Context


Routing engine
A DSL wires endpoints and processors together to form routes.
CamelContext
Route 1
Route 2
Route N
from("file:c:\dir")
.filter()
.xpath(expression)
.to("jms:aQueue");
Message filter processor
Content-based router processor
Processors
Handle things in between endpoints like
EIPs
Routing
Transformation
Mediation
Enrichment
Validation
Interception
File
JMS
HTTP
Components
Provide a uniform endpoint interface
Connect to other systems

URI to Component to Endpoint

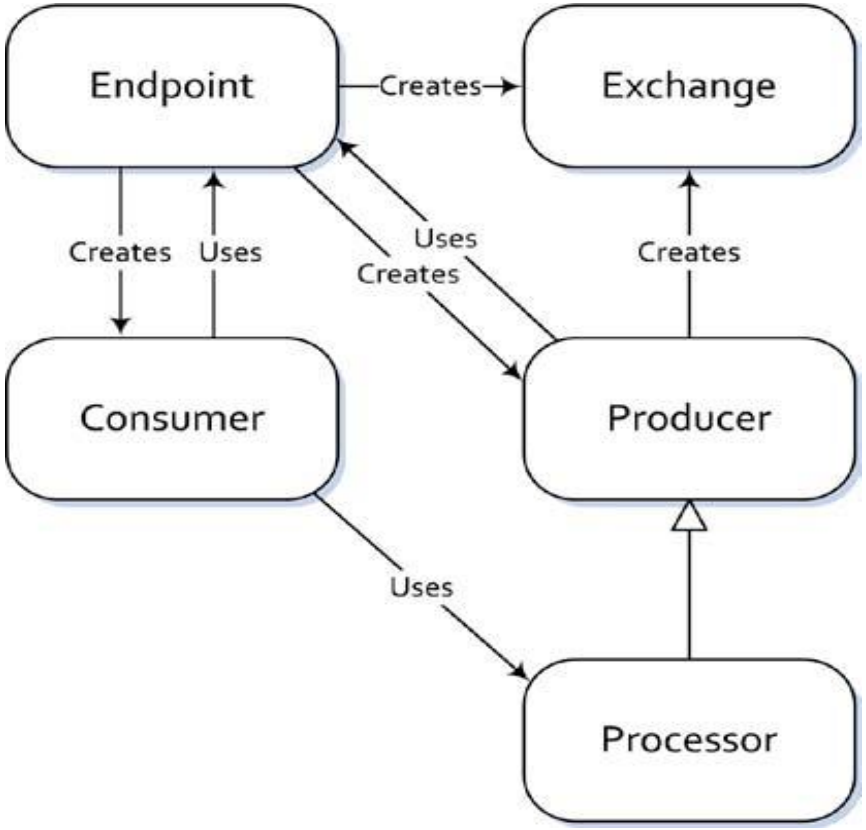


- Scheme “file” chooses FileComponent.
- FileComponent is a factory that creates a FileEndpoint based on context path and options.



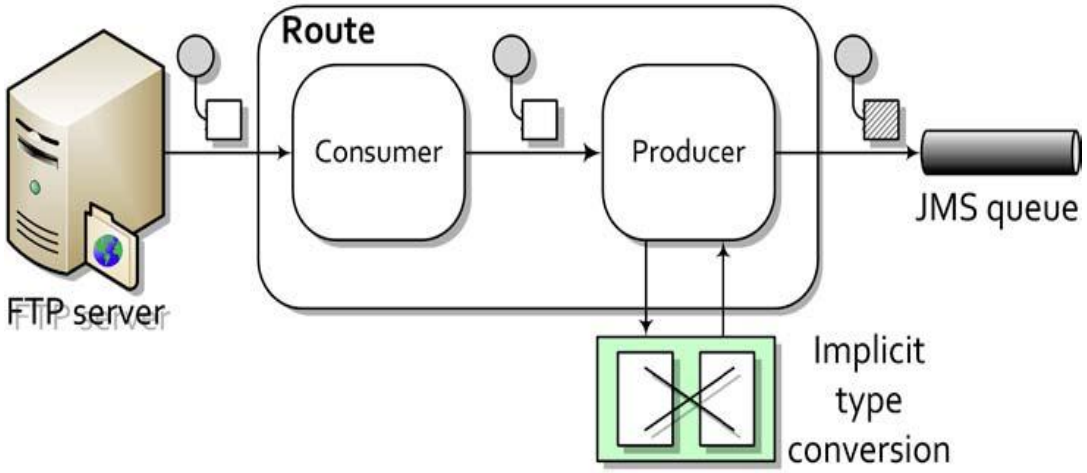
Endpoint to Producers and Consumers

- Messages are consumed and produced.



FTP to JMS Example

```
val context = new DefaultCamelContext {
  addComponent("jms", JmsComponent.jmsComponentAutoAcknowledge(
    new ActiveMQConnectionFactory("vm://localhost")))

  addRoutes(new RouteBuilder() {
    def configure() {
      from("ftp://pep.com/orders?username=demo?password=demo")
        .to("jms:queue:incomingOrders")
  }
}

context.start
Thread.sleep(10000L)
context.stop
```



Adding a Processor

- The processor interface is important for complex routes and has only a single method:

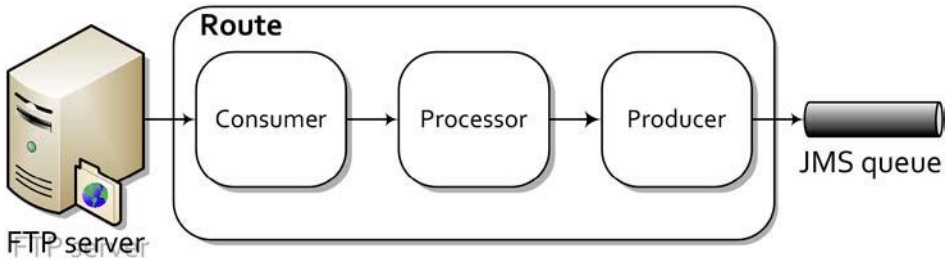
```
trait Processor { def process(exchange: Exchange): Unit }
```

- Use implicit converter in Scala to allow use of first class function.

```
implicit def wrapper(processor: Exchange => Any) = new Processor {
     def process(exchange: Exchange): Unit = processor(exchange)
}
```

- Example:

```
from("ftp://pep.com/orders?username=demo?password=demo")
     .process((exchange: Exchange) =>
          println("Downloaded: " + exchange.getIn.getHeader("CamelFileName"))
     .to("jms:queue:incomingOrders")
```



Guice Injection and GuicyFruit JNDI

```
import com.google.inject.{Injector, Provides}
import org.apache.camel.guice.CamelModuleWithMatchingRoutes;
import org.guiceyfruit.jndi.JndiBind;

class GuiceModule extends CamelModuleWithMatchingRoutes {
   override protected def configure() {
    super.configure
    bind(classOf[LoadBalanceRouteBuilder])
    bind(classOf[SomeTrait]).to(classOf[SomeBean])
   }

   @Provides
   @JndiBind("jms")
   def jms(injector: Injector) =
    injector.getInstance(classOf[PooledActiveMQComponent])
}
```

JNDI Properties

- Guice JNDI provider.

```
java.naming.factory.initial=org.guiceyfruit.jndi.GuiceInitialContextFactory
```

- List of space separated Guice modules to boot up.

```
org.guiceyfruit.modules=com.purplefinder.enterprise.r2.camelloadbalanceexam
   ple.GuiceModule
```

- Properties injected using the @Named annotation.

```
activemq.brokerURL=tcp://localhost:61616
```

Startup Guice Camel Context

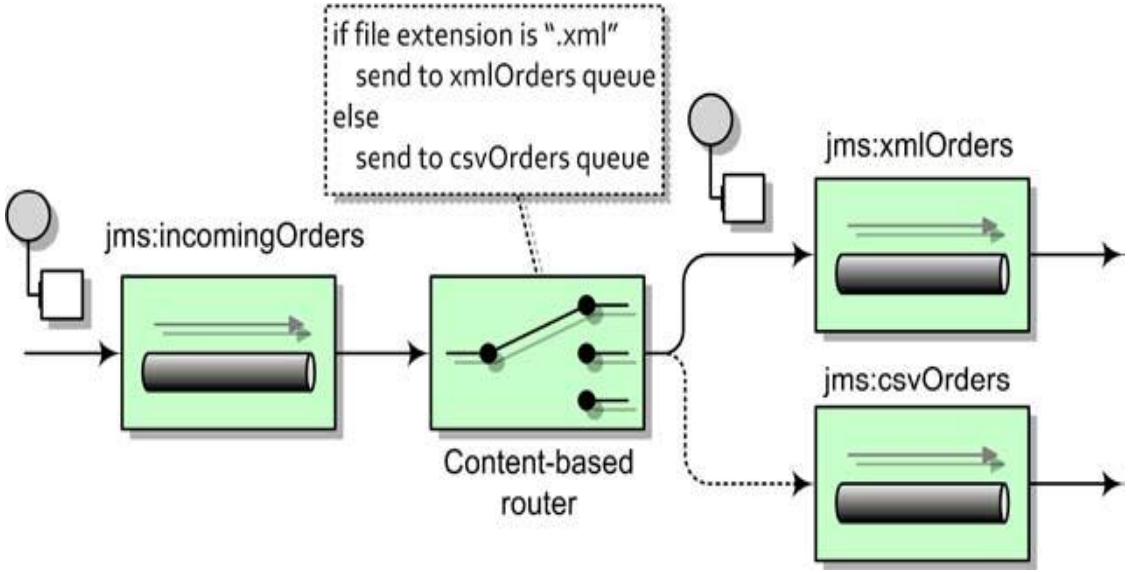
```
import org.apache.camel.guice.Main
import org.apache.camel.Exchange
import org.apache.camel.management.JmxSystemPropertyKeys

object LoadBalanceRouteBuilder extends Application {
  // Disable JMX within Apache Camel.
  System.setProperty(JmxSystemPropertyKeys.DISABLED, “true”)
  val brokerServiceInstance = new BrokerServiceInstance

  try {
    Main.main(“-duration”, “2s”)
  } finally {
    brokerServiceInstance.destroy
  }
}
```

Load Balancer Pattern Example

```
class LoadBalanceRouteBuilder extends RouteBuilder with RouteBuilderExtension {
  def configure() {
    from("file:src/data?noop=true&sortBy=file:name")
            .bean(classOf[SomeTrait])
            .to("jms:queue:input")

    from("jms:queue:input")
            .loadBalance
            .failover(-1, true, true)
            .to("direct:1", "direct:2", "direct:3")

    from("direct:1")
            .process((exchange: Exchange) => throw new Exception)
            .to("log:route.1?showBody=true")

    from("direct:2").to("log:route.2?showBody=true")
    from("direct:3").to("log:route.3?showBody=true")
  }
}
```

Content-Based Router Example

```
from("jms:incomingOrders")
   .choice
    .when(header("CamelFileName").endsWith(".xml"))
        .to("jms:xmlOrders")
    .when(header("CamelFileName").endsWith(".csv"))
        .to("jms:csvOrders")
```



Data Transformation

- Data formats are pluggable transformers that can transform messages from one form to another and vice versa.
- Each data format is represented in Camel as an interface in org.apace.camel.spi.DataFormat containing two methods:
- Marshall – From custom data format to well-known data format.
- Unmarshal – From well-known data format back to the custom data format.

Well-known data formats

- CSV
- XML
- SOAP
- JSON
- JAXB
- Electronic data interchange (EDI)
- …

Error Handling

- Permanent error is an error that will always be an error no matter how many times you try.

```
val fault = exchange.getOut
fault.setFault(true)
fault.setBody("Unknown customer")
```

- Transient error is a temporary error that might not cause a problem on another attempt.

```
exchange.setException(exception)
```

- An exception thrown within a Camel route is considered to be transient by default.

Error Handlers

- DefaultErrorHandler – No redelivery and exceptions propagated to the caller.
- DeadLetterChannel – Implements Dead Letter Queue EIP pattern.
- TransactionErrorHandler – Transaction-aware error handler installed when method transacted is at start of route.
- NoErrorHandler – Disable error handling.
- LoggingErrorHandler – Just log errors.

Redelivery

```
def configure() {
   errorHandler(defaultErrorHandler
    .maximumRedeliveries(5)
    .backOffMultiplier(2.0)
    .redeliveryDelay(1000L)
    .retryAttemptedLogLevel(LoggingLevel.WARN))

   onException(IOException.class)
    .maximumRedeliveries(3)
    .handled(true)
    .to("ftp://gear@ftp.rider.com?password=secret");

   from("file:/riders/files/upload?delay=1h")
    .to("http://riders.com?user=gear&password=secret")
}
```

Testing with mock endpoints

```
val context = new DefaultCamelContext
context.start

context.addRoutes(new RouteBuilder() {
    def configure() {
      from("file:src/data?noop=true").to("mock:x")
  }

val x = context.getEndpoint("mock:x", classOf[MockEndpoint])
x.expectedBodiesReceived(<message>Hello</message>)
MockEndpoint assertIsSatisfied context
```

PEP R2 Example Applications

- pep-r2-camel-load-balance-example

  Demonstrates load balancing and failover with Camel. Application also includes XML, Guice, Guicyfruit, ActiveMQ, Scala, BDD with Specs, mock endpoints, SLF4J.

- pep-r2-camel-jta-transaction-example

  Demonstrates distributed JTA/XA transactions with Camel. Application also includes XML, JDBC endpoint, Guice, Spring JavaConfig, Atomikos, ActiveMQ, DLQ, HSQLDB, SLF4J.