

**Apache Camel:**

Apache Camel is an open-source integration framework based on known Enterprise Integration Patterns. Camel needs you to define routes and rules and headers based on the specification like (aws-ses, pdf-create, file )

Apache Camel uses URIs to work directly with any kind of Transportor messaging model such as HTTP,ActiveMQ,JMS,JBI, SCA,MINA or CXF, as well as pluggable Components and Data Format options . Apache Camel is a small library with minimal dependencies for easy embedding in any Java application. Apache Camel lets you work with the same API regardless which kind of Transport is used.

Camel can be configured either by using Spring or directly in Java - which this example does , We start with creating a CamelContext- which is a container for Components,Routes . Below is the camel context and route configuration example.

CamelContext context=**new** DefaultCamelContext();

There is more than one way of adding a Component to the CamelContext. You can add components implicitly - when we set up the routing - as we do here for the “seda:end”.

context.addRoutes(**new** RouteBuilder() {

@Override

**public void** configure() **throws** Exception {

from(**"direct:start"**)

.to(**"seda:end"**); }

});

In normal use, an external system would be firing messages or events directly into Camel through one if its Components but we are going to use the Producer Template which is a really easy way for testing your configuration.

ProducerTemplate producerTemplate=context.createProducerTemplate();

Next you must start the camel context.though if you are using a pure Java approach then you just need to call the start() method.

context.start();

This will start all of the configured routing rules.So after starting the CamelContext, we can send some objects or body to camel using producertemplate with endpoint uri and object.

producerTemplate.sendBody(**"direct:start"** ,**"Hello World!"** );

Camel supports either a processor or an Expression to populate the new Exchange. Using a processor gives you full power over how the Exchange is populated as you can set properties,headers, et cetera. An Expression can only be used to set the IN body.

Below is the processor variation. This example is to create a new, empty Exchange.

context.addRoutes(**new** RouteBuilder() {

@Override

**public void** configure() **throws** Exception {

from(**"direct:start"**)

.process(**new** Processor() {

**public void** process(Exchange exchange) **throws** Exception {

String message=exchange.getIn().getBody(String.**class**);

message= message + **" Welcome to My world"**;

Message message1= exchange.getIn();

message1.addAttachment(**"filename"** , **new** DataHandler(**new** FileDataSource(file)));

*// message1.setHeader(SesConstants.FROM , "noreply@500aps.com");*

*// message1.setHeader(SesConstants.TO , "chandipriya.gangishetty@500apps.com");*

exchange.getOut().setBody(message1);

exchange.getOut().setBody(message);

}

}).to(**"aws-ses://%s?amazonSESClient=#client&subject=%s&to=%s"**,**"noreply@500aps.com"**,**"Hi"**, **"chandipriya.gangishetty@500apps.com"**);

}

});

Below are some of the pos related to apache camel for creating file and sending mail using apache camel.

**public static void** main(String[] args) {

SimpleRouteBuilder routeBuilder = **new** SimpleRouteBuilder();

CamelContext ctx = **new** DefaultCamelContext();

**try** {

ctx.addRoutes(**new** RouteBuilder() {

@Override

**public void** configure() **throws** Exception {

from(**"direct:start"**)

.to(**"file:/home/agile/Desktop/text.txt"**).to(**"direct:test"**);

from(**"direct:test"**).process(exchange -> {

System.***out***.println(**"Processing 2"**+exchange.getProperties().keySet());

})

.to(**"aws-ses://noreply@unscr.me?region=US\_WEST\_2&accessKey=AKIAJNLQ5LXNMMXCLLMA&secretKey=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&subject=CamelTest&to=apachecamel@yopmail.com"**);

}

});

ctx.start();

ProducerTemplate producerTemplate = ctx.createProducerTemplate();

producerTemplate.sendBody(**"direct:start"**,**"multiple from"**);

ctx.stop();

} **catch** (Exception e) {

e.printStackTrace();

}

}

Below example is for Read data from producer and add file in S3.

from(**"direct:textToS3"**)

.setHeader(S3Constants.***KEY***,simple(**"order.txt"**)) .to(**"aws-s3://appup-bucket?region=US\_EAST\_1&accessKey=AKIAIVJMBGBK4PRJ3SKA&secretKey=RAW(YTLxPha9rAUkPQzU/ywYD+ur**

This one will create file and upload file in S3

from(**"direct:textToS3"**)

.to("file:/home/agile/Desktop/text?fileName=report.txt")

.convertBodyTo(byte[].class ,"iso-8859-1")

.setHeader(S3Constants.*CONTENT\_LENGTH*, simple("14"))

.setHeader(S3Constants.*KEY*,simple("test/report.txt"))

.to("aws-s3://appup-bucket?region=US\_EAST\_1&accessKey=AKIAIVJMBGBK4PRJ3SKA&secretKey=RAW(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)");

This will create pdf file and upload pdf file in aws-s3

context.addRoutes(**new** RouteBuilder() {

@Override

**public void** configure() **throws** Exception {

from(**"direct:start"**)

.to(**"aws-s3://appup-bucket?region=US\_EAST\_1&accessKey=AKIAIVJMBGBK4PRJ3SKA&secretKey=RAW(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)"**);

}

});