# Chapter 02 : Routing with Camel

1. A close up of a text

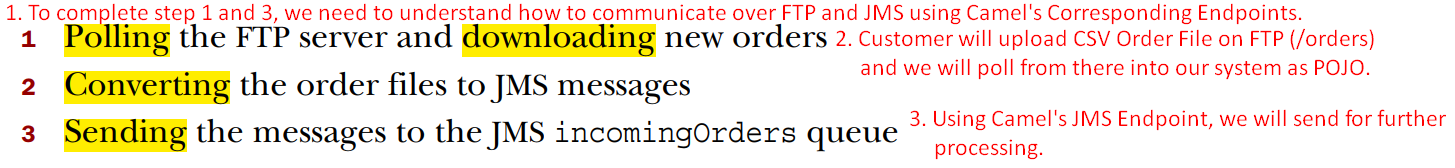
   Description automatically generated
2. When routing **Physical mail**, it may be **routed through several cities**.  
   When **routing email**, it may be routed through **several computer network systems**.  
   In all cases, the router’s function is to selectively move the message forward.
3. In the context of Enterprise Messaging System, routing is the process by which a message is taken from one input queue and based a set of conditions, send to one of several output queues, as shown in figure 2.1.  
   Consumer and producer are unaware of these conditions.  
   A diagram of a diagram

   Description automatically generated
4. **In Apache Camel, Routing** = Origin Endpoint (Consumer) + A Processing Component + Target Endpoint (Producer).
   1. Routing is step by step movement of message, which originates from an endpoint in the role of consumer.
   2. **Consumer could be receiving message from:**
      1. External Service.
      2. Polling for the message on some system.
      3. Or creating the msg itself.
   3. **Then the message flows through a processing component**: With zero or more processing component. If zero, then simple pipeline.
      1. Enterprise Integration Pattern.
      2. A Processor.
      3. An interceptor or
      4. Some other custom creation.
   4. Finally, the message is sent to the target endpoint in the role of producer.
5. In this chapter, we’ll first introduce the financial company that we will use as the running example through the book.
6. A text on a white background

   Description automatically generated

## 2.1 Introducing Rider Auto Parts

1. Our fictional motorcycle parts business, **Rider Auto Parts**, supplies to **motorcycle manufacturers**.
2. Over the years, they’ve changed the way they receive orders several times.
3. **Earlier**: CSV over FTP. A User uploads Part order on FTP Server in CSV Format from where the order (CSV) is transferred over FTP to Rider Order Frontend.
4. **New Way**: XML over HTTP by Rider Auto Parts Web Store. A user uses web store and the order as XML is transferred over HTTP as XML to Rider Order Frontend.
5. No matter what, once order comes inside Rider Order Frontend, it is converted POJO and then it is transferred to Rider Order Backend by JMS.
6. A diagram of a diagram

   Description automatically generated  
   **Figure 2.2**
7. **Problem**: Rider Auto Parts acquires Software Baggage int the form of transports (FTP, HTTP, JMS) and Data formats (CSV, HTTP).  
   **But no problem** with Integration Framework like Camel and we will implement the current requirements and new functionality using CAMEL.
8. As a first assignment, we will need to implement the FTP module in the Rider Order Frontend system.  
   Later on, we will see how backend services are implemented too.
9. Implementing the FTP module will involve the following Steps:
   1. 

## 2.2 Understanding Endpoints

1. Endpoint is the abstraction that models the endpoint of a **message channel** through which a system can send or read messages.
2. Let’s see how to configure the camel to communicate over FTP and JMS.

### 2.2.1 Working with Files over FTP

1. One of the things that make Camel easy to use is the **Endpoint URI**.
2. **URI identifies**:
   1. A component to be used (Like for each kind of transport (FTP, HTTP, Kafka etc)
   2. And how that component is configured.  
      See the following snapshot where URI is defined which will identify Camel Component and its configuration.
3. Once the URI is defined, we can either send or receive a message to the component configured by the URI.
4. To download new orders from the FTP server, we need the following.
   1. A close up of a sign

      Description automatically generated
5. See, in the following snapshot by configuring Camel with URI, we are saying to do all the above.   
   A screenshot of a computer

   Description automatically generated  
   **Figure 2.3**
6. **Scheme**:
   1. **Scheme** tells Camel which component.
   2. Camel will look for the **Scheme** “ftp” in the **Camel Registry** and it will resolve to **FileComponent.**
7. **Context Path + Options**: Based on these values, **FtpComponent** will work as factory to create **FtpEndpoint.**
8. **Context Path = rider.com/orders** : Tells **FileComponent** to log into the **FTP Server** at rider.com on the default port 21 and change the director to “orders”.
9. **Options = username=rider$password=secret** : Used **to log in to** the **FTP Server**.
10. Username and password can be specified in the context path itself without changing its earlier meaning.  
    [ftp://**rider:secret**@rider.com/orders](ftp://rider:secret@rider.com/orders)
11. **Add the following dependency as FTP Component is not part of the camel-core**.  
    
12. The above URI will work fine for consumer and producer, but we will use it to download order files CSVs from FTP.  
    So, use it in from() node in the **Camel’s DSL**.  
    
13. Recall from figure 2.2, the downloaded orders from FTP need to be sent to JMS Queue.  
    This process requires a little more setup but easy.

### 2.2.2 Sending to a JMS Queue

1. Camel provides extensive support for connecting to **JMS-Enabled Providers**. (We will cover that in detail in Chapter 7).

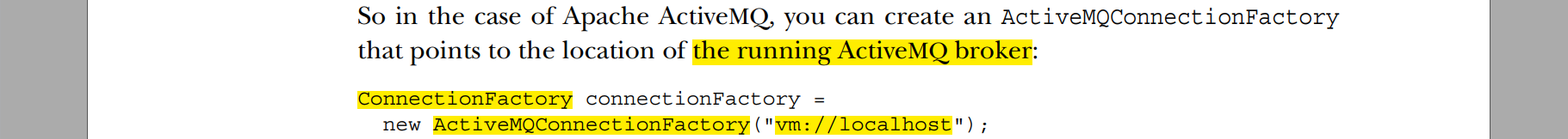
#### What is JSM?

1. JMS is a Java API to create, send, receive, and read messages. (De-Facto Messaging Solution in Java Community)
2. It also mandates that messaging is asynchronous and has specific elements of reliability like guaranteed and once-and-only-once delivery.
3. Consumers and producers talk to each other through an intermediary - **JMS Destination** which can be a queue or topic.  
   A diagram of a flowchart

   Description automatically generated
4. JMS also provides **a ConnectionFactory** which a client (like Camel) can use to create a connection with a JMS Provider.
5. JMS Providers are usually called Brokers as they manage communication b/w a producer and a consumer.

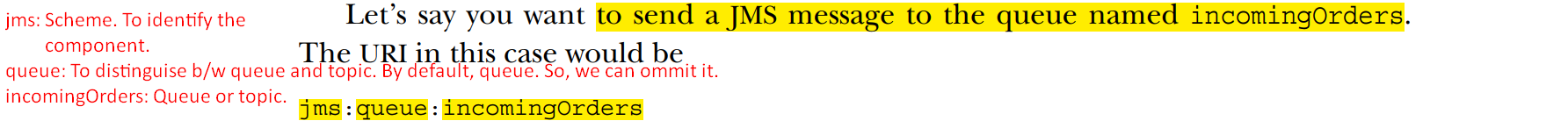
#### How to configure Camel to Use A JMS Provider

1. Configure Camel JSM Component with an appropriate **ConnectionFactory**.
2. We will use ApacheMQ JMS Provider which is the most popular and the same is used by Camel team to test Camel’s JMS Components.  
   Read ActiveMQ in Action by Bruce Snyder.
3. So, we will use **ActiveMQConnectionFactory** that points to the location of the running ActiveMQ Broker.
4. A diagram of a diagram

   Description automatically generated
5. 
6. A screenshot of a computer screen

   Description automatically generated

### Using URIs to specify the Destination.

1. 
2. JMS Component has about 60 options which can be configured in endpoint but camel provides defaults for most of the options and many options are used in very specific cases.
3. A text on a white background

   Description automatically generated

# 2.3 Creating Routes in Java

1. In chapter 1, we saw how **CamelContext** can contain multiple routes and how **RouteBuilder** can be used to create one route.
2. A close up of a computer screen

   Description automatically generated
3. A close up of text

   Description automatically generated
4. A white rectangular object with black text

   Description automatically generated

## 2.3.1 Using the RouteBuilder

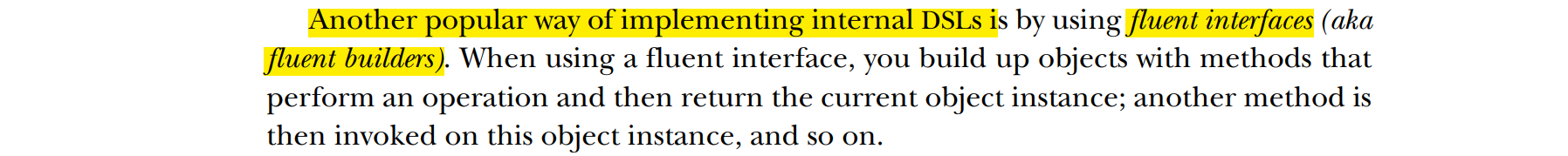
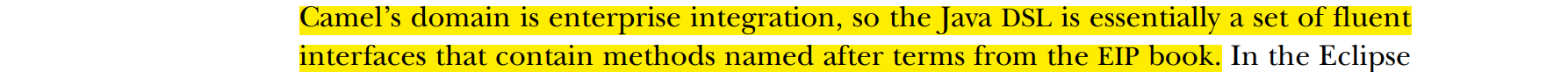
1. A screenshot of a computer program

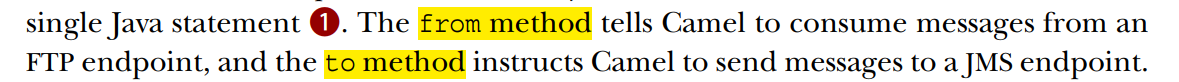
   Description automatically generated
2. **Another way**:  
   A yellow and black text

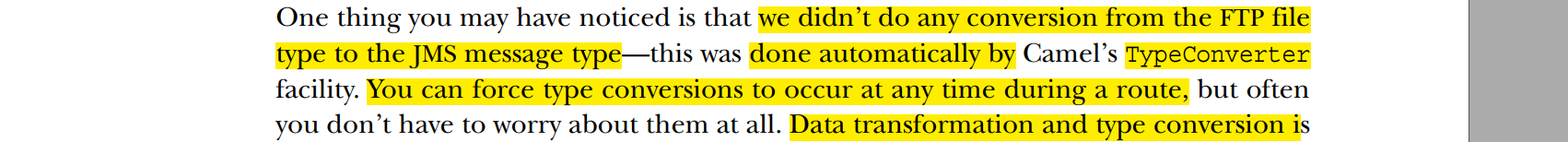
   Description automatically generated
3. A yellow and black text

   Description automatically generated

### 2.3.2 The Java DSL

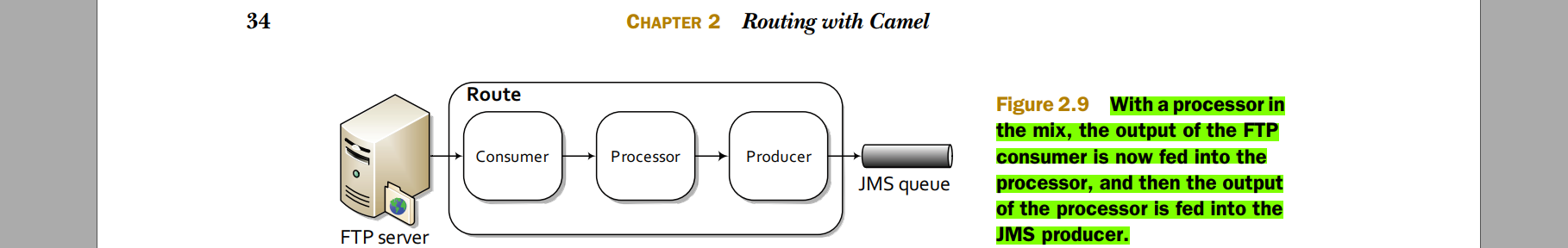
1. DSL (Domain Specific Language) targets a domain problem rather than a general-purpose language.
2. For Example: Regular Expression DSL to match a string text which is clear and concise way of matching strings. Doing the same String-Matching using Java is cumbersome.  
   Regular Expression DSL is **External DSL** as it has its own custom syntax and so requires a separate compiler or interpreter to execute.
3. **Internal DSL:** Uses the existing general-purpose language such as java in such as way that DSL feels like a language from a particular domain.  
   The most obvious way of doing this is by naming the method and argument names to match concepts from the domain in question.
4. 
5. 
6. A screenshot of a computer

   Description automatically generated
7. 
8. A diagram of a machine

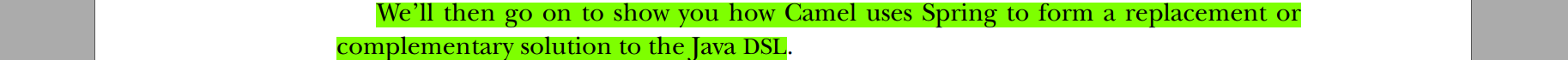
   Description automatically generated
9. 

#### ADDING A PROCESSOR

1. 
2. Interface.
3. Important building block in Camel.
4. Gives us full access to Exchange and we can do anything with Message (Payload) and Headers.
5. All EIPs in **Camel are implemented as Processors**.
6. A screenshot of a computer code

   Description automatically generated
7. 

## 2.4 Creating Routes with Spring

1. Spring is the most popular IOC Java Container.
2. The core framework allows us to wire beans together to form an application.
3. 

### 2.4.1 Bean Injection and Spring

1. Creating an app from Beans using Spring is pretty simple.
2. All we need are beans, ApplicationContext and Spring Container.
3. **So, how does Camel fit into this?**
4. Camel can be configured as if it were another bean.
5. A screenshot of a computer

   Description automatically generated