[Java教程]Camel in action(第一章译文)

2014-04-05 10:00:06

1

1 First steps
Camel 介绍
Camel路由

First steps

Apache camel 是 一个开源的一体化框架，其目的是使一体化系统更容易。本书的第一章节我们将介绍camel及展示它适合大企事业单位的软件。你将会学习到关于camel的概念及一些专业术语。

第二章将介绍camel其中一个最重要的一部分：消息路由。Camel主要有两种方式的路由规则：基于java领域的指定语言（DSL）和基于spring的

Camel介绍

本章要点
Camel介绍
Camel主要特点
你的第一个camel试驾
Camel的体系结构及内容

打造一个复杂的一体化系统是一个非常昂贵的实验，并且这个实验室几乎不可能成功的。一个有效的并且降低风险的方式是用经过验证的组件以拼图的方式组装一个一体化系统。我们平时就依赖此一体的系统，我们就可以用电话交流发生的一起，例如财务交易、有益于身心健康的旅行计划或者娱乐活动。

你还不能完成一个拼图游戏，直到我们有了这些可以自由组合、无缝连接的集合块。有了这样的系统就可以更多的关系自己的业务，不需要在考虑系统之间的连接及系统内部之间的工作。一个好的一体化的框架提供了简单的，易管理的系统。

Camel就是这样的一个框架，在本书中，我们将帮助你理解camel是什么，如何去使用它及为什么它是一个最好的一体化框架。

这一章就开始介绍camel框架及它核心最精彩的部分。我们接下来将要介绍camel的分布及解释在本书中可以运行起来的例子。我们将会全面的介绍camel的最核心最精彩的部分，是你能够理解camle的体系架构。

介绍**camel**

Camel是一个整合框架，其目的是使你的所有的项目更高效有趣。Camel项目最早开始于2007年。但是它还很年轻，camel已是一个成熟的开源项目了。在apache 2 的许可证是有效的，它有一个强大的交流社区。

Camel 关注在简易集成。我们相信你到目前为止已经完成了前面的阅读。接下来你将会领略到camel的魅力，添加它到你的工具清单中。

Camel项目之所以被命名为简单的"Camel"是 因为这个名字简短易记。传言camel的名字是有一个发现者抽的是骆驼牌香烟。在camel website输入(http://camel.apache.org/why-the-name-camel.html)，camel名字清单。

Camel是什么

Camel框架的核心是一个路由引擎，它允许你定义自己的路由规则，决定接受哪些消息，做出决定如何处理，发送这些消息给其他目标。Camel用这种集成语言允许你定义复杂的路由规则。

Camel的基本原则之一是不会假设任何你需要处理的数据，这是很重要的一点，因为它给你们开发者一个集成任何系统的一个机会，不需要转换你的数据为另外的一种公认格式。

Camel 提供了高水平的抽象，它允许你根据相同的api协议或者系统的数据类型集成各种各样的系统。Camel的组件提供了特殊实现的接口api，其目的是给不同的协议和数据类型服务。Camel打破了传统模式，它支持80多种不同的协议和数据类型，它的扩展性和模块性允许你实现你自己专有协议的无缝插件。这些体系结构的选择淘汰没有必要的转换，从而使camel更加的高效，易学。结果证明，camel是适合嵌入到其他项目中的，因为它提供了充足的处理能力。其他开源的项目，例如Apache ServiceMix和ActiveMQ已经使用camel作为企业集成的一种处理方式。

我们应该提问camel不是什么，camel不是ESB，有些人称camel是个轻量级的ESB，因为它支持路由、事务、监控、编制等。Camel 没有一个容器或者一个可靠的消息总线，但是它可以依赖例如开源的ESB或者前面提到的ServiceMix，由于以上原因，我们更喜欢把camel称作超越一个ESB的集成框架。

理解camel是什么，这样能够更能好的看到它的主要功能点。让我们来看一下它的主要功能点。

为什么使用**Camel**

Camel为整合领域介绍了一些新奇的观点，这就是为什么它的作者们决定在第一领域创建camel，代替正在使用已经存在的框架。我们将通过本书介绍富有camel所有的功能点。这些功能点如下：

路由和斡旋引擎
企业集成模式
Domain-specific language（DSL）
可拓展组件库
有效负载路由
模块化、组件式体系结构
POJO模型
容易配置
数据类型自动转换
轻量核心
测试装备
充满活力的社团

路由和斡旋引擎

路由和斡旋引擎是camel的核心功能。路由引擎基于路由配置有选择的对消息进行路由。在camel的实例中，路由配置队列是以企业集成模式和DSL。以上两种我们将在下面介绍。

企业集成模式（**EIPS**）

尽管集成的问题多中国多样，Gregor Hohpe 和Bobby Woolf 发现了很多问题。接着你们很容易的把问题解决掉了。他们编写一本《Enterprise Integration Patterns》书，这本书对那些专业集成的人来说是必读的书。接下来，它将帮助你理解又快又容易地理解camel。

企业集成模式或者EIPS对我们是有帮助的，因为它不仅给我们提供了问题的解决方案，并且也帮我们定义了我们所交流问题的本身。有了模式问题交流起来更容易。使用模式语言比描述需要解决的问题容易的多，就好像学习汉语比学习甲骨文要容易的多。

Camel基于EIPs是沉重的。尽管EIPs描述了集成问题的解决方案，同时也提供了通用的词汇列表，但这些词汇列表没有形成书面语言。Camel就避免了这个问题，它提供了一种集成描述性语言。在EIPs和camel的DSL

语言之间几乎有一种一对一的关系。

Domain-specific language（**DSL camel**自己的描述性语言）

Camel的domain-specific language（DSL） 对集成领域来说是一个主要的贡献。有一些类似于DSL（允许你用

DSL的目的是让开发者关注集成问题而不是程序开发工具。尽管Camel大部分是用java开发的。但是它支持多种程序语言。每一种语言都有属于它治的健壮性，你可能想用不同的语言完成的不同的任务。你有这样的自由建立自己的解决方案。

这有一些使用不同程序语言DSL 例子。

Java DSL

```
From("file:data/inbox").to("jms:queue:order");
```

Pring DSL

```
<route>
  <from uri="file:data/inbox"/>
  <to uri="jms:queuq:order"/>
</route>
```

Scala DSL

```
from "file:data/inbox"->"jms:queue:order"
```

这些例子都是真实代码，展示了路由一个或者多个文件从一个文件夹到JMS队列是很容易的。因为这是真正的底层程序语言，你可以用已经存在的工具支持，例如代码实现，代码编译错误的检查。例如：

你刚才看见的是Eclipse IDE的自动完成列表清单，供我们选择。

额外的组件库（大量的组件库）

Camle提供了超过80个组件的拓展库。这些组件使camel用APIs连接传送和理解数据的格式。

有效的负载路由

Camle能够提供任何一种有效的负载路由。你不能限制

模块化和组件式体系结构

Camel 有一个模块化体系结构，它允许任何组件加载到camel中，当组件船上出现不受欢迎的组件时，那就是来自第三方或者你自己定制的组件。

POJO模型

Beans（POJO）在camel是一等公民，它致力于让你在任何地方任何时间使用beans。这也就意味着很多地方你可以继承camel内建功能编写自己风格的代码。在第四章已经讨论了在camel中bean的使用。

配置简单

配置的惯例是尽量减少配置的要求，是为了在路由中配置终点。Camel使用更简单更直接的URL配置。

例如你可能配置一个文件夹下面的a.txt文件，如下：

from("file:data/inbox?recursive-true&include-*.txt")...

类型自动转换

Camel已经嵌入了150多种自动转换的机制。你不在需要配置类型转换规则，例如把byte arrays 转换为Strings。如果你需要一种camel没有的转换类型，你可以自己去创建属于自己的Converter。最关键的部分工作在引擎中，so你可以不用担心it。

Camel的组件有杠杆的特点。他们可以接受各种各样的数据类型，把这些类型转换为其他类型。他们有这个能力。在camel社区里面这是个最受欢迎的特性。你甚至可能开始想问为什么在java本身不提供这样的功能呢？在第三章将介绍多种类型转换。

轻量级的核心

Camel核心被认为是轻量级的，总共自由包加起来大约也就1.6MB，它只依赖Apache Commons Logging，资源的通用管理。这样使camel更容易在你喜欢的任何地方嵌入或者部署。例如在标准的应用、web应用、spring应用、J2EE应用、JBI容器、OSGI bundle、java web start或者Google Appengine。Camel没有被设计成一个server或者ESB，取而代之的是可以嵌入到你选择的任何平台中。

测试工具箱

Camel提供了测试工具箱，使测试你自己的camel应用更容易。这些测试工具在测试camel本身中广泛应用。例如他们可以帮助你虚拟真正的endpoints。它提供了安装进度条的功能。Camel可以根据它来断定服务是正常启动或者启动失败。在第六章cover了Camel的testing。

活跃的社团

Camel有一个活跃的社团。如果在你的项目中想使用任何开源的项目，加入的该社团中是必须得。不活跃的项目几乎没有社团的支持，出了问题那就是你自己的问题。使用camel，如果你有任何问题，热心的使用者和开发者会迅速处理帮助你。

到目前为止你已经看到了camel的主要组成的功能。我们有不多的人手在观察camel的使用范围和试用的例子。

1.2快速开始

我们将介绍怎么得到Camel的帮助文档，解释camel里面包括什么。怎么用Maven运行例子。After this，你将学会运行从本书源码中的任何例子。手续看一下camel的分布。

1.2.1 Getting camel

在官方Apache Camel上可以看到Camel的下载地址（http://camel.apache.org/download.html）。

打开网址你将会看见你将会看见不同版本的camel下载列表,当然也会有最新版本的下载。

我们将使用camel2.5.0。去下载这个版本，点击camel 2.5.0 版本的链接。几乎在本页的底部你会找到两个二进制的发布包：为windows用户zip发布包，和为Unix/Linux/Cygwin 用户提供的tar.gz发布包。把下载好的包解压开。

```
janstey@mojo:~/apache-camel-2.5.0$ ls
doc  examples  lib  LICENSE.txt  NOTICE.txt  README.txt
```

1.2.2 你的第一个camel例子

本书的中的例子都使用了Apche的Maven。那就意味着Camel仓库将会自动为你下载。本书的源码地址：http://manning.com/ibsen或者http://code.google/p/camelinaction

第一个例子我们会用传统的"hello world"：路由文件。假如你需要读一个目录（data/inbox）下面的文件，然后用一定的方式进行处理，然后把结果写到另外一个文件目录（data/outbox）。很简单，你将会跳过处理过程，你的输出仅仅是一份原来文件的copy。图Figure 1.2 画出了这个过程。

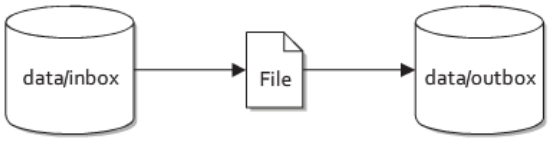

Figure 1.2 Files are routed from the data/inbox directory to the data/outbox directory.

看起来非常简单是不是。以下代码是用纯java来实现的。

Listing 1.1 Routing files from one folder to another in plain Java

```
public class FileCopier {

    public static void main(String args[]) throws Exception {
        File inboxDirectory = new File("data/inbox");
        File outboxDirectory = new File("data/outbox");

        outboxDirectory.mkdir();

        File[] files = inboxDirectory.listFiles();

        for (File source : files) {
            if (source.isFile()) {
                File dest = new File(
                     outboxDirectory.getPath()
                     + File.separator
                     + source.getName());
                copyFIle(source, dest);
            }
        }
    }

    private static void copyFile(File source, File dest)
        throws IOException {
        OutputStream out = new FileOutputStream(dest);
        byte[] buffer = new byte[(int) source.length()];
        FileInputStream in = new FileInputStream(source);
        in.read(buffer);
        try {
            out.write(buffer);
        } finally {
            out.close();
            in.close();
        }
    }
}
```

以上十分简单的用例，但是用java实现仍旧需要34行代码。你不得不使用低级的文件API，确保数据流已经关闭，这样的程序很容易出错。

用camel来实现如下：

Listing 1.2 Routing files from one folder to another with Apache Camel

```
public class FileCopierWithCamel {
    public static void main(String args[]) throws Exception {
        CamelContext context = new DefaultCamelContext();
        context.addRoutes(new RouteBuilder() {
            public void configure() {
                from("file:data/inbox?noop=true")
                    .to("file:data/outbox");
            }
        });
        context.start();
        Thread.sleep(10000);
        context.stop();
    }
}
```

1 Routes files from inbox to outbox

但你使用Camel时大部分类似于java这样的代码都显得有点样板了。每一个Camel应用使用CamelContext中的方法进行启动或者停止。你也可以添加一个sleep method为你copy文件准备时间。见listing 1.2.

Camel路由是以边读边流动的方式来定义的。上面那个路由是以这样的方式来读的：消费者的消息来自位置为data/inbox（设置noop选项）文件夹下面的文件，接着把文件路由到位置data/outbox中。noop 选项告诉Camel离开（保留）资源文件。若有你不使用noop选项，这个文件会被删除掉。从没有见过Camel的人们能够理解camel的路由做了些什么。你可能也想关注这一点。出去掉样板代码，你创建一个路由文件替换上面的java code。

运行这个例子。你需要去下载and实例一个apache的maven(http://maven.apche.org/download.html)。

Note 本书上的代码用的是Maven 2.2.1。最新版本可能出现意想不到的错误。

POM展现如下：

Listing 1.3 The Maven POM required to use Camel's core library

```
<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>
  <parent>
    <groupId>com.camelinaction</groupId>
    <artifactId>chapter1</artifactId>
    <version>1.0</version>
  </parent>

  <artifactId>file-copy</artifactId>

  <name>Camel in Action :: Chapter 1 :: File Copy Example</name>

  <dependencies>
    <dependency>
      <groupId>org.apache.camel</groupId>
      <artifactId>camel-core</artifactId>
      <version>${camel-version}</version>
    </dependency>
  </dependencies>
</project>
```

1 Parent POM

2 Camel's core library

Maven本身是个复杂的话题。我们在这不做更多的细节介绍。我们将给你足够的信息介绍本书中的所有例子。推荐读 Maven by Example 和 The Complete Reference。我们也用Maven部署应用。（没有用过Maven的同学，可以去学习一下，很不错 比ant好用）

To run the example in listing 1.2, use the following command:

```
mvn compile exec:java -Dexec.mainClass=camelinaction.FileCopierWithCamel
```

1.3 camel消息模型

在camel中有两个消息模型的抽象概念。下面的section我们将会涉及到。如下：
Org.apache.camel.Message----这个基础的实体包含需要在camel中路由的数据。
Org.apache.camel.Exchange----消息交换是camel的抽象概念。消息交换有个"in"消息作为回复，还有一个"out"消息。

我们开始通过查看消息来理解数据是怎样在Camel中被塑造和运输的。我们将看到在Camel中"conversation"是怎样被塑造的。

1.3.1 消息

系统之间通讯的时候Messages是作为独立实体来用的。消息从发送者的方向到消息的接受者，插入如下：

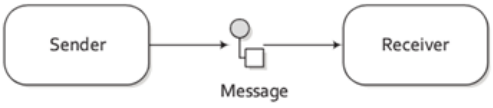

Figure 1.3 Messages are entities used to send data from one system to another.

消息有一个体，头信息及一些附加配置，如下图：

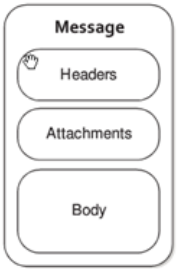

Figure 1.4 A message can contain headers, attachments, and a body.

消息有定义标识，标识的type是java.lang.String。消息创建的时候，其唯一标识是被迫必须被创建的。消息是依赖协议的，它没有强制的格式。这些协议没有定义一种消息的唯一标识库。Camel通常用它自己的UID。

头信息和附加配置

头信息里面放的是消息之间的关系，例如发送者的UID、内容编码的迹象、认证消息等等。

头信信息是name-value对；name是唯一的。强转换为String类型，因为name的类型是java.lang.Object类型的。这就意味着Camel在header类型上强加约束。头是以Map的形式来存储信息。一个消息可能也会有选项附加配置，例如通常被web service和email 组件使用。

消息体（**Body**）

消息体是java.lang.Object类型。这就意味着可以存储任何类型的消息。由应用设计者来决定消息接收者可以理解的消息内容。当消息的接收者和发送者使用不同的body 类型。Camel提供了一些转换可以接收格式的数据机制。

失败标志（**FAULT FLAG**）

消息也有一个失败标志。

1.3.2 交换

Camel的交换是在路由过程中发生在消息容器中。一种交换同时提供了系统之间不同类型的集成，例如众所周知的MEPs（Message exchange patterns）。MEPs between事件消息（one-way） and request-response以不同的消息风格被使用。Camel的exchange有一种属性模式供选择。

InOnly ---事件消息（ A one-way message）。例如，JMS消息经常就以时间消息来使用。

InOut---请求响应消息（request-response message）。例如，HTTP。一个客户端请求一个需要返回的web页面，等待server回复。Canel交换都包括哪些内容如下插图：

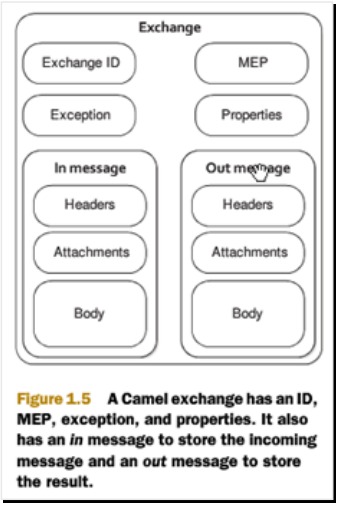

Figure 1.5 A Camel exchange has an ID, MEP, exception, and properties. It also has an *in* message to store the incoming message and an *out* message to store the result.

让我们来看一下Figure1.5每个元素的细节。

Exchage ID---用来交换的唯一标识。Camel会提供一个默认的唯一ID

MEP-A-- 指定的样式。你是否正在使用Inonly或者InOut消息结构样式。当只是InOnly样式时，The change只包含一种in消息。若是InOut模式，out消息也包含一个回复消息给请求者。

Exception（异常）---如果在路由的过程中任何时间发生异常，则这个异常被放到异常空间。

属性---相似的消息头，他们持续在整个交换过程中。属性包含有被使用global-leve信息。反之消息头对哪些特有的消息来说是特别的。Camel在路由期间它自己将添加各种属性。你作为一个开发者可以在交换的生命周期内的任何point保留或者恢复这些属性。

In 消息—这是一种被强制的输入消息。这In 消息包含请求消息。

Out 消息---这是一种宣泄消息。它仅存在MEP中。这Out消息包含恢复消息。

我们想在讲camel体系结构前讨论一下camel的消息模型。我们想你对camel的消息已经有了一定的理解。毕竟camel大部分重要的方面都是路由消息。你现在已经具备了更多的去学习关于camel和他的体系结构。

1.4 camel体系结构

我们现在把注意力转到camel的体系架构上来。我们将首先看一下高水平的体系架构，接着深入的研究一下这些内容。你读过本章段后，你应该懂得了集成的术语，在第二章为你准备了。我们将深入研究camel的路由能力。

1.4．1 camel体系机构

我们制作了高水平的camle体系架构视图，如下插图：

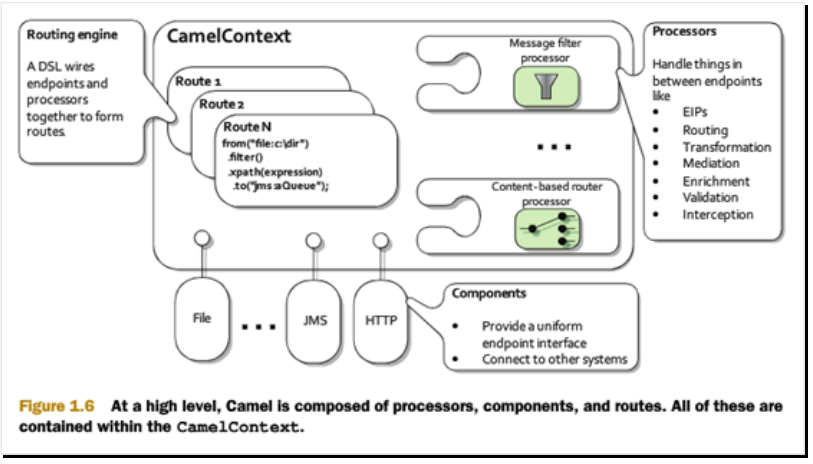

Figure 1.6 At a high level, Camel is composed of processors, components, and routes. All of these are contained within the `CamelContext`.

路由引擎有消息路由的详细说明。路由作为DSLs语言被定义使用。在路由期间消息会被处理成各种类型的消息。在消息在路由期间实现了所以的EIP。在camel中这些组件时作为连接其他系统的拓展点。

1.4.2 camel的概念

Camel有很多新的概念，因此让我们花一些时间一个一个理解它。我们从CamelContext开始，它是camel的运行环境。很重要哟。

CamelContext

你可能已经猜到CamelContext是一个有类别容器，从figure1.6的插图你可能认为它是camel的运行环境系统。Figure1.7展示了大部分值得注意的服务。这些服务都是camel保留下来的。如下插图：

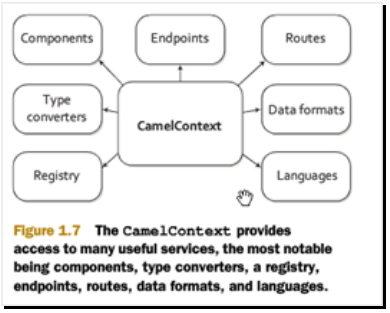

Figure 1.7 The CamelContext provides access to many useful services, the most notable being components, type converters, a registry, endpoints, routes, data formats, and languages.

The details of each of these services will be discussed throughout the book. Let's

now take a look at routes and Camel's routing engine.

这些服务将会贯穿本书。现在我们先看一下ROUTES和ROUTING ENGINE 。

Table 1.1 The services that the CamelContext provides

Service	Description
Components	Contains the components used. Camel is capable of loading components on the fly either by autodiscovery on the classpath or when a new bundle is activated in an OSGi container. In chapter 7 we'll discuss components in more detail.
Endpoints	Contains the endpoints that have been created.
Routes	Contains the routes that have been added. We'll cover routes in chapter 2.
Type converters	Contains the loaded type converters. Camel has a mechanism that allows you to manually or automatically convert from one type to another. Type converters are covered in chapter 3.
Data formats	Contains the loaded data formats. Data formats are covered in chapter 3.
Registry	Contains a registry that allows you to look up beans. By default, this will be a JNDI registry. If you're using Camel from Spring, this will be the Spring `ApplicationContext`. It can also be an OSGi registry if you use Camel in an OSGi container. We'll cover registries in chapter 4.
Languages	Contains the loaded languages. Camel allows you to use many different languages to create expressions. You'll get a glimpse of the XPath language in action when we cover the DSL. A complete reference to Camel's own Simple expression language is available in appendix A.

上图部分翻译：

Components：包含用到的组件。Camel 能够通过自动载入类路径或者自动激活OSGI容器等方式来运行。第七章中讨论。

Endpoints：包含已创建的endpoints。

Routes：包含被添加的路由。会在第二章中讲解。

Type converters：包括已经载入的类转换。Camel拥有可以手动或自动转换类的机制。第三章中详细描述。

Registry：包含可以查阅beans的注册表。默认情况下，它是JNDI registry。如果基于Spring，它是Spring ApplicationContext ，如果使用OSGI容器，它是OSGI registry。第四章讲。

Languages：包含载入的语言。Camel允许使用不同的语言创建表达式。你会看到在讲解DSL的时候使用的Xpath 语言。完全参考camel自己的简约的表达式语言可以参考 附录A。

路由引擎

Camel的路由引擎其实就是做消息移动。 引擎不会暴露给开发者。但是你应该意识到它的存在，它做了所以的重活，确保消息能够根据属性准确路由。

路由

路由是camel的一个核心抽象概念。最简单的方式是定义一个路由作为处理链。在消息 应用中使用路由有很多种原因。解耦的客户端和服务器端，生产者和消费者，路由可以
动态的决定一个客户端将会被那个服务调用。
提供了一种处理额外流程的灵活方式
运行服务器端和客户端独立开发。
增强一些系统的功能。

Camel中的每一个路由都有一个唯一的标识。这些标识为logging，debuging，监控，路由的启动停止服务的。路由也有一个精确的输入消息资。

Each route in Camel has a unique identifier that's used for logging, debugging, monitoring, and starting and stopping routes. Routes also have exactly one input source for

messages, so they're effectively tied to an input endpoint.

Camel中每一个路由都有一个唯一的标识，用于日志、调试、监控、和启动停止路由。路由有完整的输入资源，所以他可以有效的和input endpoint 联系。

下面定义一个使用DSL语言的路由。

DSL：

定义一个路由。

DSL（Domain-Specific Language）

在camel中，DSL也就是经常使用的java API（给EIP提供的方法名称）。考虑一下下面的例子：

from("file:data/inbox").filter().xpath("/order[not(@test)]).to("jms:queue:order")

在这一句java语句中，你定义了路由。From 一个文件，这个文件路由到EIP，这个消息是否是测试的目的或者不是。如果一个消息通过了测试，它将会被转发到JMS结束点。失败消息过滤器将会放弃执行。

Canmel提供了多种DSL语言，你可以用spring定义同样的路由。Like this：

```
<route>
  <from uri="file:data/inbox"/>
  <filter>
    <xpath>/order[not(@test)]</xpath>
    <to uri="jms:queue:order"/>
  </filter>
</route>
```

创建应用的时候DSL为Camel用户提供了一个抽象的概念。路由真正构成了处理过程。让我们来看一下处理过程是什么。

PROCESSOR

处理过程是camel的一个核心概念。代表了一个有处理能力节点的使用，创建，或者修改交换输入。在路由期间交换会从一个处理流到下一个处理；例如你可以想象一个画面，每个节点都有指定的处理方式。一个处理的输出到一个处理的输入连成一个队。怎么做交换的输入输出处理画面呢？我们需要看一下components和endpoints

Component

在camel中组件是主要的继承点。时至今日，已经有超过80个组件体系，应用范围有数据转换函数，DSL，数据格式，等等。你甚至可以创建自己的组件。在第11章讨论。

从程序的视角来看，组件是十分简单扼。他们的名字会用在URL中产生关联关系，他们扮演着终点工厂。例如一个 FileComponent 会关联URI中的一个文件，然后创建一个FileEndpoints。EndPoint可能是一个非常基本的事件概念。

EndPoint

Endpoint是camel的抽象概念。信道末端模型。系统可以利用信道收发消息。如下图：

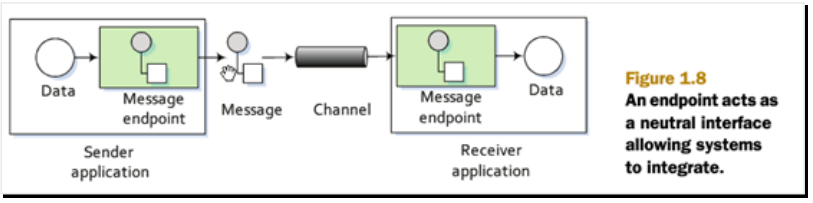

Figure 1.8 An endpoint acts as a neutral interface allowing systems to integrate.

在camel中，你需要在URL中配置endpoint。例如file:data/inbox?delay=5000,你也用这种法师涉及到了endpoints。在运行的时候，camel将会用URL标记的方式查找endpoint。

如下图：展示了它是怎么工作的

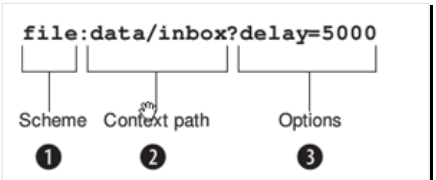

Figure 1.9 Endpoint URIs are divided into three parts: a scheme, a context path, and options.

Scheme 标识endpoint的类型。在这个例子中，文件模式（scheme of file）选择文件组件（FileComponent）。这个FileComponent作为创建FileEndpoint 的工厂。这个Context path 是data/inbox 告送FileComponent文件夹的路径。这个Option 属性 delay=5000表示此文件应该被推迟5秒钟。

Figure 1.10 展示了endpoint和producer、exchange、consumer之间如何工作的。

先看一眼figure1.10可能有点晕，在几分钟后你就会领会的。Endpoint作为创建consumer和producer的工厂。

Producer

Producer是camel的一个抽象概念。它是用来创建消息给endpoint

++

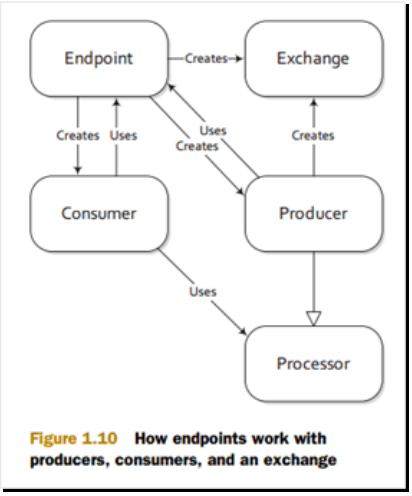

Figure 1.10 How endpoints work with producers, consumers, and an exchange

Producer：

A producer is the Camel abstraction that refers to an entity capable of creating and

sending a message to an endpoint. Figure 1.10 illustrates where the producer fits in with other Camel concepts.

When a message needs to be sent to an endpoint, the producer will create an exchange and populate it with data compatible with that particular endpoint. For example, a FileProducer will write the message body to a file. A JmsProducer, on the other hand, will map the Camel message to a javax.jms.Messagebefore sending it to a JMS destination. This is an important feature in Camel, because it hides the complexity of interacting with particular transports. All you need to do is route a message to an endpoint, and the producer does the heavy lifting.

生产者：

Producer是一个camel的抽象类，实体能够创建并发送消息到endpoint。图1.10一方面也说明了这个概念。当一个消息需要被发送到一个endpoint时，生产者将创建一个exchange并填充兼容个别项目的endpoint数据。例如：FileProducer会将消息body写入到一个文件。还有JmsProducer会把Camel消息映射到正在发送消息到JMS目标上的javax.jms.Messagebefore中。在camel中，这是很重要的功能，因为它隐藏了与特有的transports的相互作用的复杂性。你要做的是将一个消息路由到endpoint，然后producer会做重活。

CONSUMER：

A consumer is the service that receives messages produced by a producer, wraps them

in an exchange, and sends them to be processed. Consumers are the source of the

exchanges being routed in Camel.

Looking back at figure 1.10, we can see where the consumer fits in with other

Camel concepts. To create a new exchange, a consumer will use the endpoint that wraps the payload being consumed. A processor is then used to initiate the routing of the exchange in Camel using the routing engine.

In Camel there are two kinds of consumers: event-driven consumers and polling consumers. The differences between these consumers are important, because they

help solve different problems.

消费者：

消费者是接收生产者生产的消息的服务。把他们封装到exchange中并且发送他们去处理。消费者是camel中被路由的exchanges的源头。

回头看图1.10，我们看到消费者与其他camel概念的交互。创建一个新exchange，消费者将使用endpoint封装被消费的有效载荷。Processor然后使用路由引擎发起exchange的路由。在camel中有两种消费者：事件驱动的消费者 和 轮询的消费者。它们的区别很重要，因为它们有助于解决问题。

事件驱动的消费者（**EVENT-DRIVEN CONSUMER**）：

大多是事件驱动的消费者，如图1.11

这种消费者和客户-服务结构webservice。同时在EIP中也表示为异步的接收者。消费者监听指定的消息通道，通常使用TCP/IP端口或JMS队列，等待客户端发送消息给它。当消息到达的时候，消费者唤醒并且将消息进行处理。

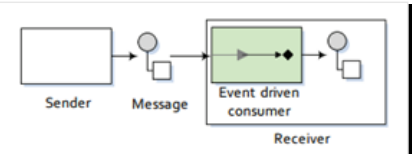

Figure 1.11 An event-driven consumer waits idle until a message arrives, at which point it wakes up and consumes the message.

轮询的消费者（**POLLING CONSUMER**）

另一种是轮询消费者如图1.12 。

相比事件驱动消费者，轮询消费者主动从指定源获取消息，例如FTP服务。轮询消费者在EIP术语中也是 同步的接受者，因为它不会获取更多消息，直到他完成当前消息的处理。通常轮询消费者是定时轮询消费者，通过指定间隔。File，FTP和Email传输通常用定时轮询消费者。

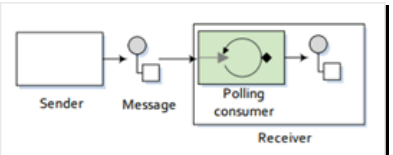

Figure 1.12 A polling consumer actively checks for new messages.

1.5 在回头看第一个camel示例

回头看1.2.2的camel示例，你应该可以更加透彻的理解了。

再来看一个例子。

Listing 1.4 Routing files from one folder to another with Apache Camel

```
public class FileCopierWithCamel {
    public static void main(String args[]) throws Exception {
        CamelContext context = new DefaultCamelContext();
        context.addRoutes(new RouteBuilder() {
            public void configure() {
                from("file:data/inbox?noop=true")
                    .to("file:data/outbox");
            }
        });
        context.start();
        Thread.sleep(10000);
        context.stop();
    }
}
```

1 Java DSL route

示例中，在camel运行时会首先创建一个camelContext，通过RouteBuilder和java DSL来添加路由逻辑。通过DSL，可以一目了然的让camel组件，endpoints，消费者，生产者等实例化。你要关注的是定义路由为你的集成项目。底层中，camel是访问FileComponent并且用它作为一个工厂创建endpoint和它的生产者。同时也创建消费者。

1.6 总结

略

标签：

1

登录 | 注册 还可以输入140字

顺便说点什么吧……

表情 同步到微博

0条评论

还没有人评论过，赶快抢沙发吧！

获得微博评论箱