

- [首页](#)
- [开源项目](#)
  - [国产开源项目](#)
  - [项目分类](#)
  - [最新收录项目](#)
  - [Java 开源软件](#)
  - [C# 开源软件](#)
  - [PHP 开源软件](#)
  - [C/C++ 开源软件](#)
  - [Ruby 开源软件](#)
  - [Python 开源软件](#)
  - [Go开源软件](#)
  - [JS开源软件](#)
- [问答](#)
  - [技术问答 »](#)
  - [技术分享 »](#)
  - [IT大杂烩 »](#)
  - [职业生涯 »](#)
  - [站务/建议 »](#)
  - [支付宝专区 »](#)
  - [MoPaaS专区 »](#)
  - [开源硬件专区 »](#)
- [代码](#)
- [博客](#)
- [翻译](#)
- [资讯](#)
- [专题](#)
  - [源创会 视频](#)
  - [高手问答 访谈](#)
  - [周刊 乱弹](#)
  - [公司开源导航页](#)
  - [Android开发专区](#)
  - [iOS开发专区](#)
  - [iOS代码库](#)
  - [Windows Phone](#)
- [城市圈](#)

当前访客身份: 游客 [ [登录](#) | [加入开源中国](#) ]  
[开源中国](#)

## 技术翻译

已有文章 2343 篇  
当前位置: [译文列表](#) » [服务器端开发](#) , [投递原文](#)

在 2343 篇翻译的文章中搜索

# 针对 Java 开发者的 Apache Camel 入门指南

16  
顶

英文原文: [Getting Started with Apache Camel using Java](#)

标签: [Apache Camel](#) [Java](#)

[oschina](#) 推荐于 3年前 (共 5 段, 翻译完成于 08-29) (22评)

168人收藏此文章, [我要收藏](#)

参与翻译(2 [bigtiger02](#), [jadic](#)  
人):

[仅中文](#) | [中英文对照](#) | [仅英文](#) | [打印此文章](#)

Apache Camel是一个非常实用的规则引擎库，能够用来处理来自于不同源的事件和信息。你可以在使用不同的协议比如 VM, HTTP, FTP, JMS甚至是文件系统中来传递消息，并且让你的操作逻辑和传递逻辑保持分离，这能够让你更专注于消息的内容。

在这篇文章中，我将提供一个Java语言（非[Groovy](#)）的Apache Camel入门演示。

首先创建一个Maven项目的pom.xml。

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="
5     http://maven.apache.org/POM/4.0.0
```



翻译的不

[bigtiger02](#)  
顶 3年前

1人顶

```
6 http://maven.apache.org/maven-v4_0_0.xsd">
7
8 <modelVersion>4.0.0</modelVersion>
9 <groupId>camel-spring-demo</groupId>
10 <artifactId>camel-spring-demo</artifactId>
11 <version>1.0-SNAPSHOT</version>
12 <packaging>jar</packaging>
13
14 <properties>
15 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
16 <camel.version>2.11.1</camel.version>
17 </properties>
18
19 <dependencies>
20 <dependency>
21 <groupId>org.apache.camel</groupId>
22 <artifactId>camel-core</artifactId>
23 <version>${camel.version}</version>
24 </dependency>
25 <dependency>
26 <groupId>org.slf4j</groupId>
27 <artifactId>slf4j-simple</artifactId>
28 <version>1.7.5</version>
29 </dependency>
30 </dependencies>
31
32 </project>
```

在这里我们只用到了 camel-core.jar 包，实际上它提供了许多你可能用到的实用组件。出于日志记录的目的，我使用了 slf4j-simple 来作为 [日志记录的实现](#)，从而我们可以从控制台上看到输出。

接下来我们只需要构造一个路由类。路由就好比是 Camel 中怎样将消息从一端传递到另一端的一个指令定义。我们将会创建 src/main/java/camelcoredemo/TimerRouteBuilder.java 文件，每隔一秒向处理器发送一个消息，简单打印出来。

```
1 package camelcoredemo;
2
3 import org.slf4j.*;
4 import org.apache.camel.*;
5 import org.apache.camel.builder.*;
6
7 public class TimerRouteBuilder extends RouteBuilder {
8     static Logger LOG = LoggerFactory.getLogger(TimerRouteBuilder.class);
9     public void configure() {
10         from("timer://timer1?period=1000")
11             .process(new Processor() {
12                 public void process(Exchange msg) {
13                     LOG.info("Processing {}", msg);
14                 }
15             });
16     }
17 }
```

以上就是这个示例的全部所需，现在编译运行。

```
bash> mvn compile
```

```
bash> mvn exec:java -Dexec.mainClass=org.apache.camel.main.Main -Dexec.args='-r camelcoredemo.TimerRouteBuilder'
```

注意，这里我们并没有编写 Java 类的 main 入口，我们只是将 RouteBuilder 的类名当作参数简单传递给 org.apache.camel.main.Main，然后它将自动加载路由。

## 控制 CamelContext

当启动 Camel 后，它会创建一个 CamelContext 对象，该对象拥有了很多关于如何运行 Camel 的信息，还包含我们所创建的 Route 的定义。现在如果你想通过 CamelContext 获得更多的控制，那么你需要编写自己的主类代码。我在这举个简单的例子。

```
1 package camelcoredemo;
2
3
4
5
6 import org.slf4j.*;
7 import org.apache.camel.*;
8 import org.apache.camel.impl.*;
9 import org.apache.camel.builder.*;
10
11
12
13
14 public class TimerMain {
15     static Logger LOG = LoggerFactory.getLogger(TimerMain.class);
16     public static void main(String[] args) throws Exception {
17         new TimerMain().run();
18     }
19     void run() throws Exception {
20         final CamelContext camelContext = new DefaultCamelContext();
```



翻译的不

bigtiger02

顶 错哦！

1 人顶



翻译的不

jadic

顶 错哦！

2 人顶

```
21 camelContext.addRoutes(createRouteBuilder());
22 camelContext.setTracing(true);
23 camelContext.start();
24
25
26
27
28 Runtime.getRuntime().addShutdownHook(new Thread() {
29     public void run() {
30         try {
31             camelContext.stop();
32         } catch (Exception e) {
33             throw new RuntimeException(e);
34         }
35     }
36 });
37
38
39
40
41 waitForStop();
42 }
43 RouteBuilder createRouteBuilder() {
44     return new TimerRouteBuilder();
45 }
46 void waitForStop() {
47     while (true) {
48         try {
49             Thread.sleep(Long.MAX_VALUE);
50         } catch (InterruptedException e) {
51             break;
52         }
53     }
54 }
55 }
```

可以看到，我们在createRouteBuilder()方法中重用了已有的TimerRouteBuilder类。现在我们的主类对在什么时候创建、启动、停止CamelContext有了完全的控制。context(camelContext)对象允许你全局性地控制如何配置Camel，而不是在Route级。它的JavaDoc链接给出了所有setter方法，你可以研究下它都可以做些什么。

注意到一点，我们也需要在我们的主类中提供少量设置代码。首先我们需要处理优雅关闭的问题，所以我们增加了一个Java关闭回调函数去调用context的stop()方法。其次在context已经启动后，我们需要增加一个线程阻塞。如果在启动后你不阻塞你的主线程，那么它会在启动后就简单的退出了，那就没啥用了。你会把Camel一直作为一个服务(就像一个服务器)运行，直至你按下CTRL+C键去终止该进程。

## 改善启动CamelContext的主类

如果你不想像上面例子一样过多的处理主类设置代码，那么你可以简单地继承由camel-core提供的org.apache.camel.main.Main类作为代替。通过利用这个类，你不仅可以让你的context自动设置，还可以获得所有附加的命令行特性，比如控制进程运行多久，启用追踪，加载自定义route类等等。

重构了下上一个例子，代码如下：

```
1 package camelcoredemo;
2
3 import org.slf4j.*;
4 import org.apache.camel.builder.*;
5 import org.apache.camel.main.Main;
6
7 public class TimerMain2 extends Main {
8     static Logger LOG = LoggerFactory.getLogger(TimerMain2.class);
9     public static void main(String[] args) throws Exception {
10         TimerMain2 main = new TimerMain2();
11         main.enableHangupSupport();
12         main.addRouteBuilder(createRouteBuilder());
13         main.run(args);
14     }
15     static RouteBuilder createRouteBuilder() {
16         return new TimerRouteBuilder();
17     }
18 }
```

现在TimerMain2类的代码比之前的更少了，你可以试试看，它应该和之前的功能一样。

```
1 bash> mvn compile
2 bash> mvn exec:java -Dexec.mainClass=camelcoredemo.TimerMain2 -Dexec.args='-t'
```

注意到我们给出-t选项后，会转储Route追踪。使用-h会看到所有可用的选项。

## 用Camel的注册机制添加bean

在之前的TimerRouteBuilder例子中，我们已经在代码中创建了一个匿名Processor。现在如果你想将几个不同的Processor放在一起，那么使用Camel的注册机制添加bean的方式将能更好的减少代码混乱。Camel允许你通过将processing当作bean



翻译的不

jadic  
顶 错哦！

1人顶



翻译的不

错哦！

注入到它的registry space, 然后你只要把它们当作bean组件来进行调用。如下是我的重构代码:

```
1 package camelcoredemo;
2
3 import org.slf4j.*;
4 import org.apache.camel.*;
5 import org.apache.camel.builder.*;
6 import org.apache.camel.main.Main;
7
8 public class TimerBeansMain extends Main {
9     static Logger LOG = LoggerFactory.getLogger(TimerBeansMain.class);
10    public static void main(String[] args) throws Exception {
11        TimerBeansMain main = new TimerBeansMain();
12        main.enableHangupSupport();
13        main.bind("processByBean1", new Bean1());
14        main.bind("processAgainByBean2", new Bean2());
15        main.addRouteBuilder(createRouteBuilder());
16        main.run(args);
17    }
18    static RouteBuilder createRouteBuilder() {
19        return new RouteBuilder() {
20            public void configure() {
21                from("timer://timer1?period=1000")
22                    .to("bean:processByBean1")
23                    .to("bean:processAgainByBean2");
24            }
25        };
26    }
27
28    // Processor beans
29    static class Bean1 implements Processor {
30        public void process(Exchange msg) {
31            LOG.info("First process {}", msg);
32        }
33    }
34    static class Bean2 implements Processor {
35        public void process(Exchange msg) {
36            LOG.info("Second process {}", msg);
37        }
38    }
39 }
```

bigtiger02  
顶 于 3年前

2人顶

现在Route类更简洁明了, 同时处理代码也被重构到了独立的类中。当你需要编写很复杂的Route来实现业务逻辑时, 这种方式能够帮助你更好的组织和测试你的代码。它能够让你构建像”乐高”积木那样可复用的POJO bean。Camel的registry space同样可用于其他很多用途, 比如你可以自定义许多具有附加功能的endpoint组件或者注册一些信息, 更或者替换线程池实现策略之内的事情。

上述Route示例是用所谓的Java DSL来构成的, 它的可读性较高, 你可以用IDE提供的支持查看所有可用于Route的方法。

我希望这篇文章能够帮助你跳过Camel的摸索阶段。除了已经提到的事件组件之外, camel还提供了如下组件:

- [bean component](#)
- [browse component](#)
- [dataset component](#)
- [direct component](#)
- [file component](#)
- [log component](#)
- [mock component](#)
- [properties component](#)
- [seda component](#)
- [test component](#)
- [timer component](#)
- [stub component](#)
- [validator component](#)
- [vm component](#)
- [xslt component](#)

Have fun!

本文中的所有译文仅用于学习和交流目的, 转载请务必注明文章译者、出处、和本文链接  
我们的翻译工作遵照 [CC 协议](#), 如果我们的工作有侵犯到您的权益, 请及时联系我们

英国一年制硕士c语言入门

apache入门实木床

家具网上商城电脑学习入门

如何学习编程编程入门教程

嵌入式学习路线c语言新手入门

竹中半兵卫六韬刘韬dom4j


网友评论 共22条

[发表评论](#) [回页面顶部](#)

	Tity 发表于 2013-08-30 08:21	Apache Camel ，一般会用到什么样的场景下？	
	寻梦2012 发表于 2013-08-30 08:24	好奥文章	
	mailguest 发表于 2013-08-30 08:34	先收了，不仔细看真心没看明白。	
	王汪欢 发表于 2013-08-30 08:43	同问：一般会用到什么样的场景下？	
	火柴头 发表于 2013-08-30 08:44	谁知道：遇到哪些问题时，可以考虑使用Apache Camel？	
	在云端-看世间变幻 发表于 2013-08-30 08:48	当年刚入职就接触者东西，知识不够无法理解其原理，差点打击的没信心了	
	Rocky-Wood 发表于 2013-08-30 09:19	规则引擎，很好很强大	
	吐槽的达达仔 发表于 2013-08-30 09:20	不明觉厉~~作为规则引擎是否有个完整的案例 + main方法来给我们这些不明白的人看看呢？	
	高跟男爵 发表于 2013-08-30 09:27	说说场景。。	
	艾皮狗 发表于 2013-08-30 09:41	偶工作近5年了，表示从来用不到	
	阿伏流 发表于 2013-08-30 10:06	能给几个用例么。完全不明白用途就设	<a href="#">1</a> <a href="#">2</a> <a href="#">3</a>
	daxiaoming 发表于 2013-08-30 10:19	引用来自“大案要案命案在身”的评论 说说场景。。	

这个是消息路由，基本功能是接收消息、处理消息和分派消息。接收消息使用 component，处理消息使用 processor，分派消息也使用 component。看上面的 <from><to>。使用场景有很多，比如说我

就用 camel 做了一个小型的RPC系统。



daxiaoming 发表于 2013-08-30 10:21


引用来自 “daxiaoming” 的评论

引用来自 “大案要案命案在身” 的评论

说说场景 • •

这个是消息路由，基本功能是接收消息、处理消息和分派消息。接收消息使用 component，处理消息使用 processor，分派消息也使用 component。看上面的 <from><to>。使用场景有很多，比如说我就用 camel 做了一个小型的RPC系统。

三个功能可以一起用，也可以只用接收消息和处理消息的部分，也可以只用分派消息的部分。所以使用场景是很多的。参考 ESB 的使用场景。



苍耳道人 发表于 2013-08-30 10:48

这个东西是按照《企业集成模式:设计、构建及部署消息传递解决方案》还是《企业应用架构模式》那本书弄得，看看书再玩更好！



高跟男爵 发表于 2013-08-30 11:09

引用来自 “daxiaoming” 的评论

引用来自 “daxiaoming” 的评论

引用来自 “大案要案命案在身” 的评论

说说场景 • •

这个是消息路由，基本功能是接收消息、处理消息和分派消息。接收消息使用 component，处理消息使用 processor，分派消息也使用 component。看上面的 <from><to>。使用场景有很多，比如说我就用 camel 做了一个小型的RPC系统。


三个功能可以一起用，也可以只用接收消息和处理消息的部分，也可以只用分派消息的部分。所以使用场景是很多的。参考 ESB 的使用场景。

o 看来不是必须要用到的 • • 懂了



墨竹 发表于 2013-08-30 11:16

mark



猎户座 发表于 2013-08-30 12:46

Android上登录，传输速度最快的FTP的jar包是什么呢。apache的comment果断不行啊。。




人头马没面 发表于 2013-08-30 14:01

gqxie 发表于 2013-08-30 08:21回复

Apache Camel ，一般会用到什么样的场景下 ？

同问



jianglibo 发表于 2013-08-30 14:41


引用来自 “人头马没面” 的评论

gqxie 发表于 2013-08-30 08:21回复

Apache Camel ，一般会用到什么样的场景下 ？

同问

个人觉得它是一个异构系统之间的粘合剂，利用丰富的endpoint，在任何系统之间建立联系。本人撰写的一个建站系统，在两个地方使用camel，一、索引，将新增文章或者修改的文章写入特定目录（xml格式），camel会自动将它post到solr服务器。二、图片和视屏转换。使用camel相当于将任务异步化了，从过网页发出转换命令另，你可以将这个命令保存外一个txt文件后者一个消息，有camel来异步执行。



lxrjszl 发表于 2013-08-30 18:08

真的感觉到门在哪了



发表评论

[回评论顶部](#) | [回页面顶部](#)