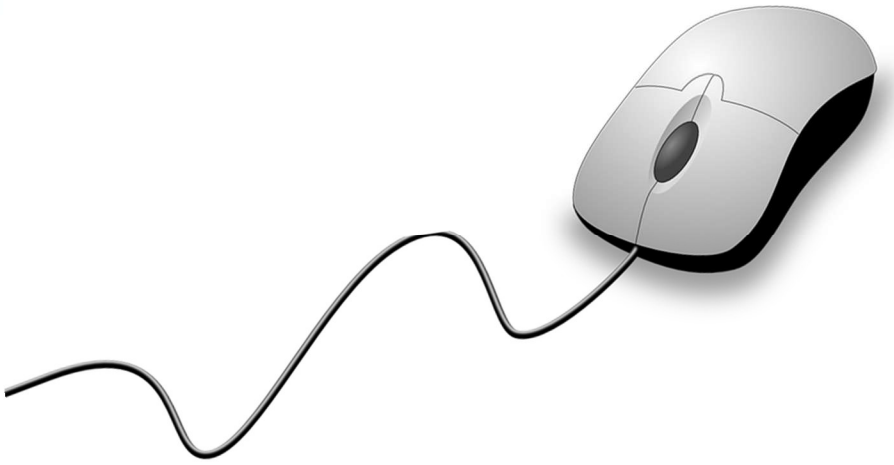
공개SW 솔루션 설치 & 활용 가이드

시스템SW > 분산시스템SW



제대로 배워보자

How to Use Open Source Software

Open Source Software Installation & Application Guide

nipa 공개SW역량프라자

CONTENTS

1. 개요

소개	• EIP(Enterprise Integration Patterns) 기반의 오픈소스 통합 연계 프레임워크 • CAMEL(Concise Application Message Exchange Language)은 복잡한 라우팅 규칙을 정의하는 통합을 위한 언어 • Camel 프로젝트는 2007년에 Apache 2 오픈소스 라이선스로 시작하여 강력한 커뮤니티를 기반으로 Integration 분야에서 유명한 프로젝트		
주요기능	• 통합 연계 라우팅 엔진 (라이브러리) • EIP 연계 패턴을 내부 구현하였으며 다양한 DSL을 통해서 조합 가능 • 라우팅을 위해서 다양한 언어로 구현된 DSL 지원 • 다양한 연계 컴포넌트가 구현되어 있고 사용가능(150+)		
대분류	• 시스템SW	**소분류**	• 분산시스템SW
라이선스형태	• Apache 2	**사전설치 솔루션**	• JAVA 1.8 이상
운영제제	• Cross-platform	**버전**	• 2.22.0
특징	• EIP 연계 패턴 구현 • DSL(Domain specific Lanaguage) 지원 – Java, XML, Scala, Groovy 등 지원 • 특정 컨테이너나 프레임워크의 의존성 없음 (Ligithweight로 마이크로 서비스 형태로 서비스 가능)		
개발회사/커뮤니티	Apache Software Foundation		
공식 홈페이지	http://camel.apache.org		

2. 기능요약

- 통합 연계 라우팅 엔진
  - Camel은 메시지 라우팅 연계 엔진
  - 다양한 프로토콜, 메시지를 통합하여 일관된 인터페이스로 처리 가능

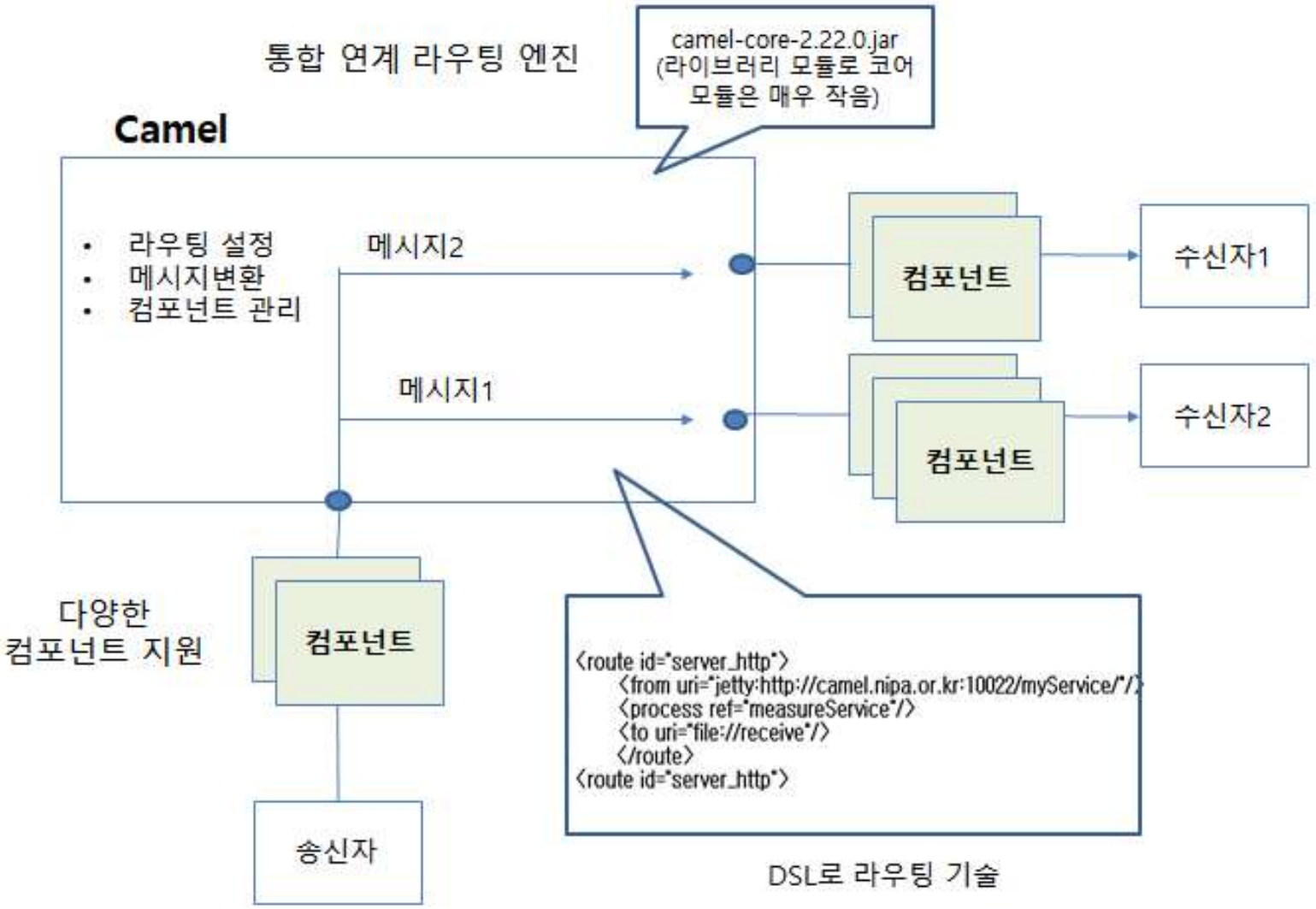
- EIP (Enterprise Integration Patterns) 구현
  - 기업 연계 패턴을 분석하여 패턴화 시킨 것이 EIP 패턴임
  - Camel을 통하여 EIP 패턴에 기반하여 다양한 연계 유형을 조립하여 구성할 수 있음

- DSL (Domain Specific Language) 지원
  - 메시지 라우팅 및 메시지 프로세싱을 다양한 언어로 기술할 수 있도록 함
  - Java, XML, Groovy, Scalar, Kotlin(진행중) 등 다양한 언어 지원

- 다양한 연계 컴포넌트 (150+), 메시지 변환 프로세서 제공
  - 다양한 연계 컴포넌트, 메시지 변환 프로세스가 제작되어 포함되어 있음
  - 필요시 사용자 컴포넌트를 제작하여 함께 구동시킬 수 있음

- Lightwight core 라이브러리 모듈
  - Camel core 모듈은 4M 정도의 라이브러리 형태로 가벼움
  - 다양한 컨테이너에 포함되어 구동될 수 있고 마이크로 서비스 형태의 서비스가 가능함

2. 기능요약


통합 연계 라우팅 엔진
camel-core-2.22.0.jar
(라이브러리 모듈로 코어
모듈은 매우 작음)
Camel
• 라우팅 설정
• 메시지변환
• 컴포넌트 관리
메시지2
메시지1
컴포넌트
수신자1
컴포넌트
수신자2
다양한
컴포넌트 지원
컴포넌트
송신자
<route id="server_http">
<from uri="jetty:http://camel.nipa.or.kr:10022/myService/"/>
<process ref="measureService"/>
<to uri="file://receive"/>
</route>
<route id="server_http">
DSL로 라우팅 기술

2. 기능요약

Apache Camel 구조

- **라우트 (Route)**
  - 컴포넌트, 프로세서들의 연속적인 메시지 연결의 정의다.
  - 시스템 간 혹은 시스템 내부에서 정의된 메시지 연결 플로우로 메시지가 어디서/어떻게/ 어디로 흘러갈지 정의한다.
  - Camel 라우트는 메시지 플로우가 1:1 혹은 1:N , N:1 등 다양하게 정의될 수 있다.
- **컴포넌트 (Component)**
  - Endpoint URL을 가지는 Camel이 메시지를 라우팅 할 수 있는 프로그램 단위다.
  - 통신 프로토콜을 구현한 컴포넌트, 파일시스템 연동을 구현한 컴포넌트 등 다양한 컴포넌트가 있다.
  - Camel 내부에 미리 구현된 컴포넌트가 150 여종이 있으며 사용자가 새로 만들어 끼워 넣을 수 있다.
- **프로세서 (Processor)**
  - 프로세서는 Camel 내부에서 메시지를 생성, 변환, 수정, 검증 등의 작업을 하여 다른 컴포넌트로 라우팅하는 모듈이다.
  - Camel은 EIP 패턴에 의한 메시지 프로세싱을 지원한다.

2. 기능요약

Apache Camel 구조

- **CamelContext**
  - Camel의 핵심 런타임 API로 컴포넌트, 프로서서, EndPoint, 데이터 타입, 라우팅 등을 관리한다.
  - CamleContext에 의해서 다양한 모듈이 로딩되고 관리된다.
- **DSL(Domain Specific Language)**
  - 컴포넌트와 프로세서를 통하여 라우트 구성을 정의하기 위해서 사용하는 언어다.
  - Camel은 Java, Spring XML, Scala, Groovy 등 다양한 언어 형태를 지원한다.
  - DSL을 통하여 다양한 동적 라우팅 및 메시지 변환 등 프로그래밍 요소를 삽입하여 사용 가능하다.
- **EndPoint**
  - Camel 컴포넌트의 주소를 나타내며 URI 형태로 기술한다.
  - 라우팅을 기술하기 위해서 컴포넌트의 Endpoint를 기술한다.
- **Producer**
  - Endpoint에 메시지를 생성, 전달할 수 있는 개체다.
- **Consumer**
  - Producer에 의해 생성된 메시지를 수신하는 개체, 수신 후 Exchange를 생성하여 라우터에 던져준다.

3. 실행환경

- **설치 요구사항**

  - Java 응용 프로그램에 쉽게 포함될 수 있도록 최소한의 종속성을 가진 라이브러리 (Camel core Module 4M)
  - Java 1.8 이상 설치 환경
  - 다양한 표준화 기술을 같이 활용하여 환경구축 지원
    - **Apache ServiceMix** - 널리 사용되는 분산형 오픈소스 ESB 및 JBI 컨테이너
    - **Apache ActiveMQ**: 많은 시스템에서 사용하고 안정적이고 신뢰할 수 있는 Message Broker
    - **Apache CXF**: Smart web services 지원 (JAX-WS and JAX-RS)
    - **Apache Karaf:** OSGi Platform으로서 모든 종류의 Application을 운영할 수 있으며 Bundle단위 Life Cycle을 관리(Run-Time환경에서 Bundle단위로 Install, Uninstall, Start, Stop)
    - **Apache MINA:** 높은 성능을 보장하는 NIO-driven networking 서비스 지원

4. 설치 및 실행

세부 목차

4. 설치 및 실행

4.1 다운받기

- Apache Camel WEB site
  - http://camel.apache.org/download.html
- IDE Tool(eclips or STS) install
  - http://www.springsource.org/springsource-tool-suite-download/
- Maven install
  - STS 사용시 maven 이 기본 탑재되어 생략가능한 단계이다.
- JDK install
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html

4. 설치 및 실행

4.2 개발환경 설치

- 해당 압축 파일을 푼 후 이클립스 프로젝트에 넣는다.
- 프로젝트의 Build Path에 필요한 라이브러리를 추가한다.
- 혹은 Maven이 설치 되어 있을 경우 pom.xml에 해당 라이브러리를 명시해준다.


```
<dependencies>
  <dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-bam</artifactId>
  </dependency>
  <dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-jaxb</artifactId>
  </dependency>
  <dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-juel</artifactId>
  </dependency>

  <!-- logging -->
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j-impl</artifactId>
    <scope>runtime</scope>
```

- **Maven**

  - http://camel.apache.org/running-examples.html url에서 Camel에 대한 Maven 실행법을 알 수 있다.
  - 실행하기전에 pom.xml을 확인하여야 한다.
  - Maven을 설치 한 후 다운받은 Camel 예제를 실행하는 방법이다.

```
#예제를 실행할 폴더로 이동 후
Camel#>  mvn exec:java
#spring base의 예제를 실행할 경우
Camel#> camel:run
```

  - 다음은 IDE에서 Maven을 사용하는 방법이다.

```
mvn idea:idea
```

  - 다음은 이클립스에서 사용하는 방법이다.

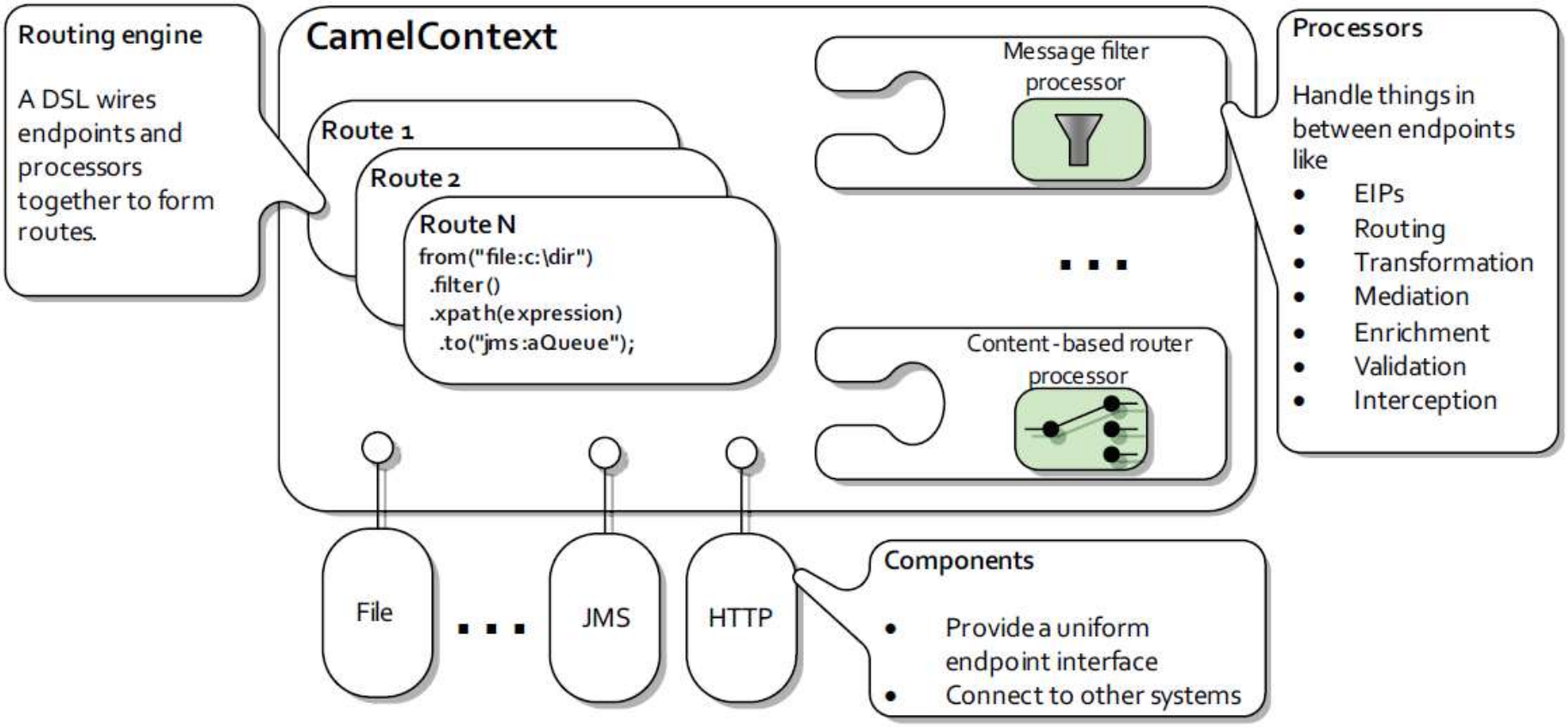
```
mvn eclipse:eclipse
```

5. 기능소개

세부 목차

5. 기능 소개

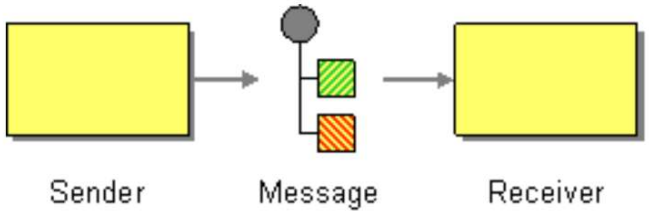
5.1 기본 아키텍처

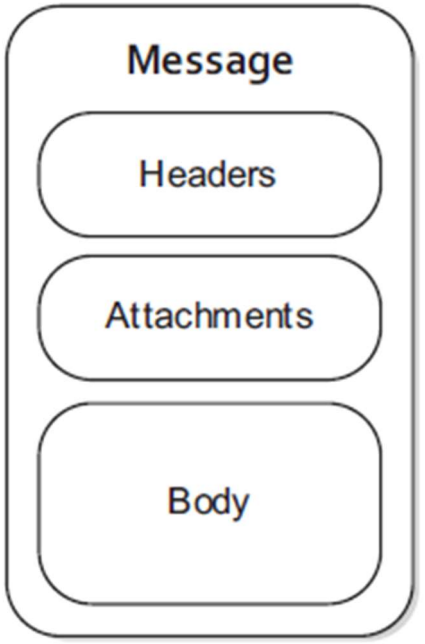


[CAMEL 기본 아키텍처] 출처 : Camel in Action (manning)

- **메시지 개념**

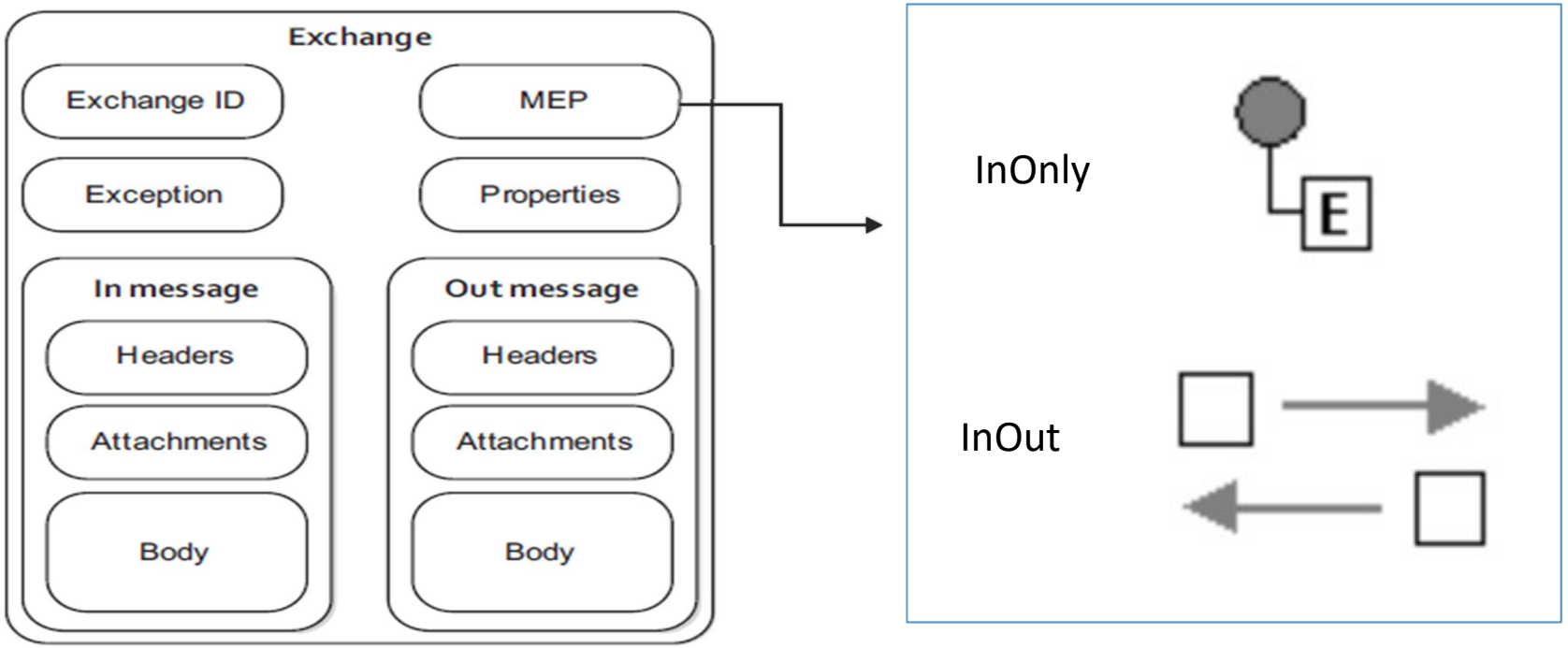
  - EIP에 정의된 메시지는 메시징 채널을 사용할 때 시스템이 서로 통신하기 위해 사용하는 엔티티
  - 메시지는 송신측에서 수신측으로 방향성이 정해짐
  - 메시지는 다양한 전송프로토콜에 의해서 다양한 포맷으로 생성되나 Camel은 헤더(Header), 첨부(Attachments), 본문(body)로 표준화 시켜서 관리함.
  - 메시지는 java.lang.String 유형의 식별자로 고유하게 식별됨





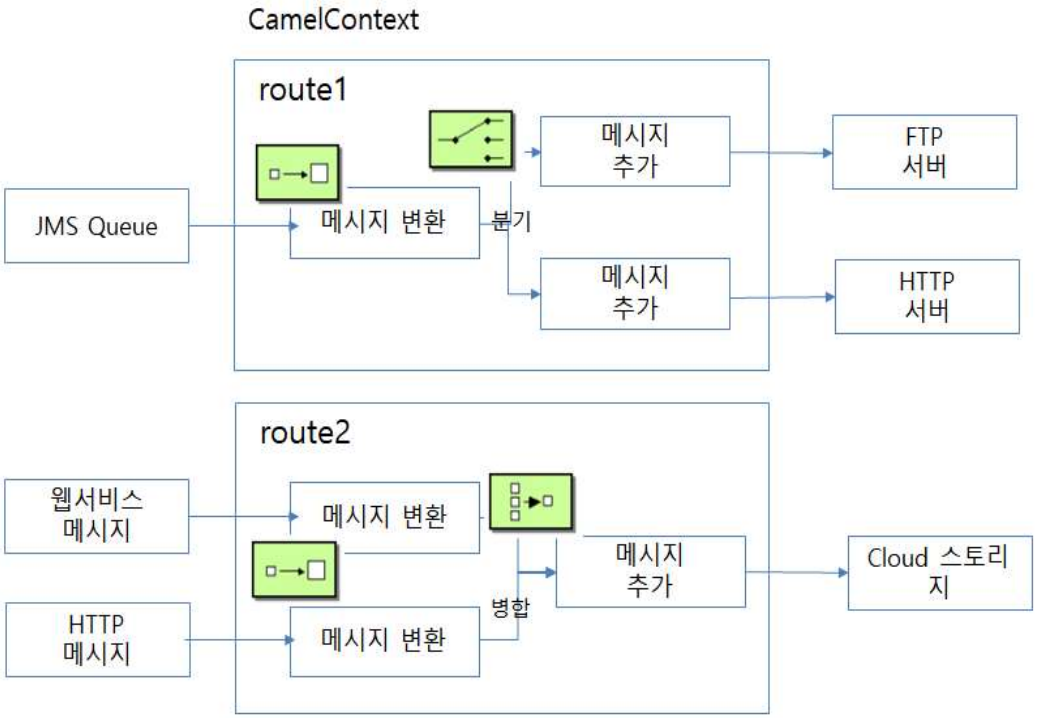
- **Exchange 개념**

  - Exchange는 Camel 내부에서 사용하는 메시지의 컨테이너 모델
  - Message Exchange Patterns (MEPs)에 의해서 정의
  - InOnly 또는 InOut 메시징 스타일을 사용하는지 메시징 교환 패턴 정보 가짐
  - InOnly - 단방향 메시지 (이벤트 메시지). 예를 들어, JMS 메시징은 보통 단방향 메시징
  - InOut - 요청 - 응답 메시지. 예를 들어, HTTP 기반 전송은 클라이언트가 웹 페이지를 검색하고 서버의 응답을 기다리면서 요청하는 경우가 많음
  - Exchange ID : 교환을 식별하는 고유 ID. 명시적으로 제공되지 않으면 기본적으로 자동 생성됨



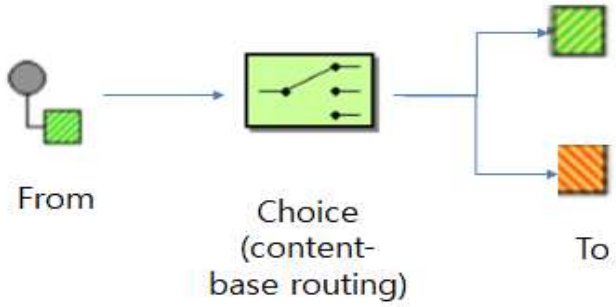
- **라우트 개념**

  - 하나의 시스템 간 연동 인터페이스를 정의
  - 예로 시스템 A에서 B로 웹 서비스를 이용해서 연동을 했다면 이것이 하나의 Route가 됨
  - 1:1 관계, 1:N관계, N:1 등 다양한 관계 지원
  - CamelContext에 의해서 다수의 route가 관리되며 각각 DSL(domain specific Language)에 의해서 기술됨



- **라우트 기술(DSL)**

  - 라우트는 DSL (Domain specific Language)에 의해서 기술됨
  - 같은 내용을 Java, XML, Scalar, Groovy 등의 언어를 통해서 기술 가능




```
public class MyRouteBuilder extends RouteBuilder {

   public void configure() {

      from("file:src/data?noop=true")
            .choice()
            .when(xpath("/person/city=seoul"))
            .to("file:target/messages/uk")
            .otherwise()
            .to("file:target/messages/others");

   }

}
```

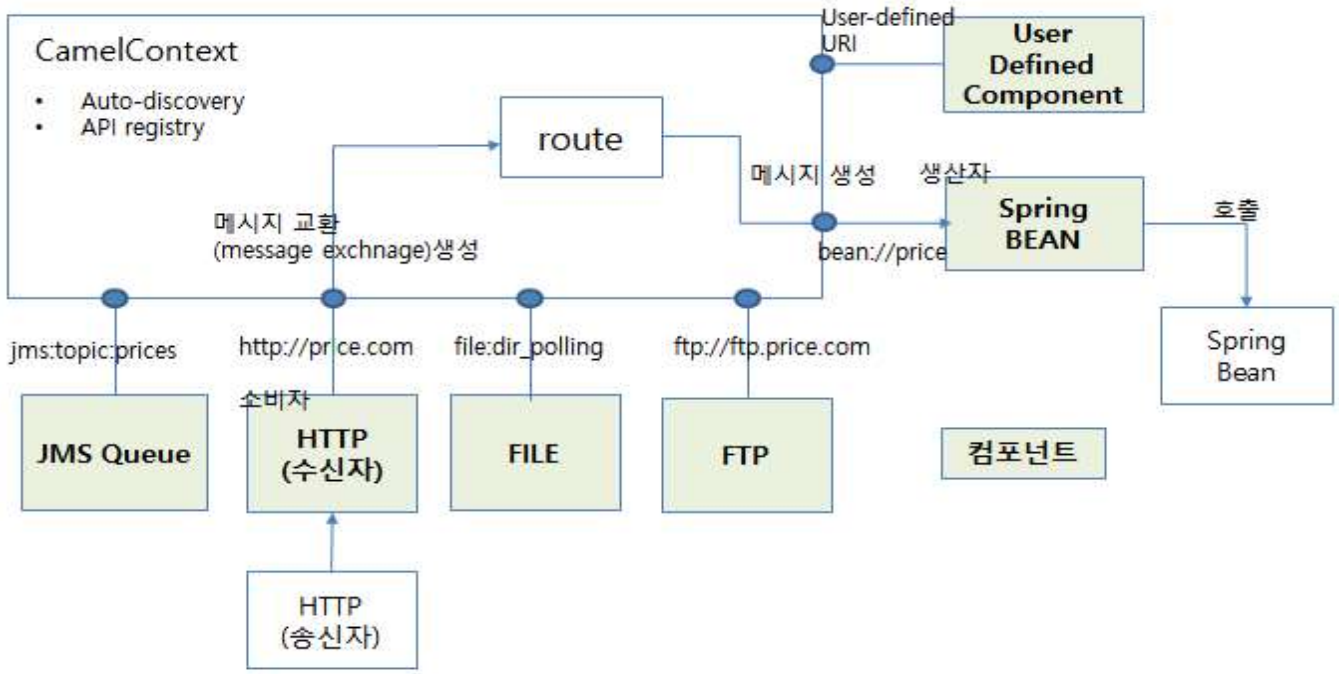
Java DSL 를 통한 라우트 설정

```
<route>
   <from uri="file:src/data?noop=true">
   <choice>
      <when>
         <xpath>"/person/city=seoul"</xpath>
         <to uri="file:target/messages/uk">
      </when>
      <otherwise>
         <to uri="file:target/messages/others">
      </otherwise>
   </choice>
</route>
```

XML DSL 를 통한 라우트 설정

- **컴포넌트 개념**

  - 어댑터의 개념으로 각 컴포넌트는 Endpoint URI를 생성함
  - Camel 내에서는 URI를 통해서 각각의 컴포넌트를 인식함
  - 컴포넌트는 송신 시스템으로부터 메시지를 읽어오며(소비자 역할), 수신 시스템으로 메시지를 전송(생상자 역할) 하는 역할 수행
  - 예를 들어 송신 시스템을 FTP로 연동하고 싶으면 FTP Conponent, JDBC로 연동하고 싶으면 JDBC컴포넌트를 사용
  - 컴포넌트는 클래스패스에 의해서 자동으로 등록 가능하며 API를 통한 등록도 가능함



5. 기능소개

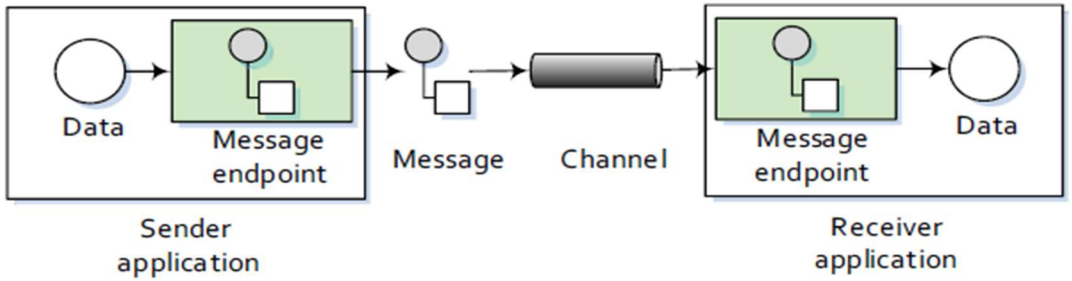
5.5 컴포넌트(Component)(2/2)

- Camel은 미리 구현된 150개 이상의 다양한 컴포넌트 구현체가 있음
- 다양한 상용/공개 소프트웨어와 연동이 가능하며 연계 프로토콜 구현체를 컴포넌트로 사용 가능
- 라우터 설정시 해당 컴포넌트의 URI를 통해서 호출함

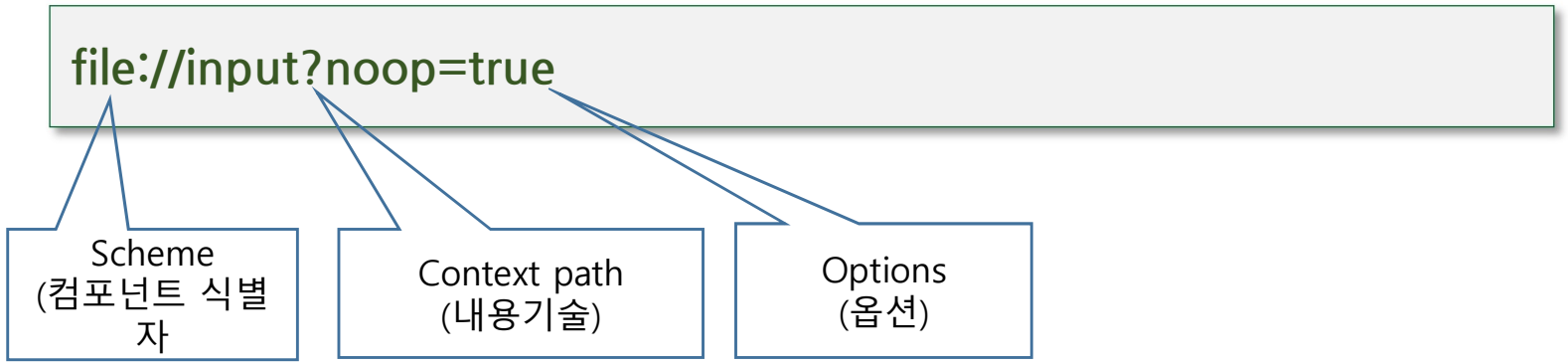
컴포넌트	ArtifactId	URI
AMQP	camel-amqp	amqp:[queue:|topic:]destinationName[?options]
AWS-EC2	camel-aws	aws-ec2://label[?options]
Bean	camel-core	bean:beanName[?options]
CXF	camel-cxf	cxf:<bean:cxfEndpoint|//someAddress>[?options]
FTP	camel-ftp	ftp:contextPath[?options]
Git	camel-git	git:localRepositoryPath[?options]
Google Drive	camel-google-drive	google-drive://endpoint-prefix/endpoint?[options]
HTTP	camel-http	http:hostName[:port][/resourceUri][?options]
JMS	camel-jms	jms:[queue:|topic:]destinationName[?options]

- **Endpoint 개념**

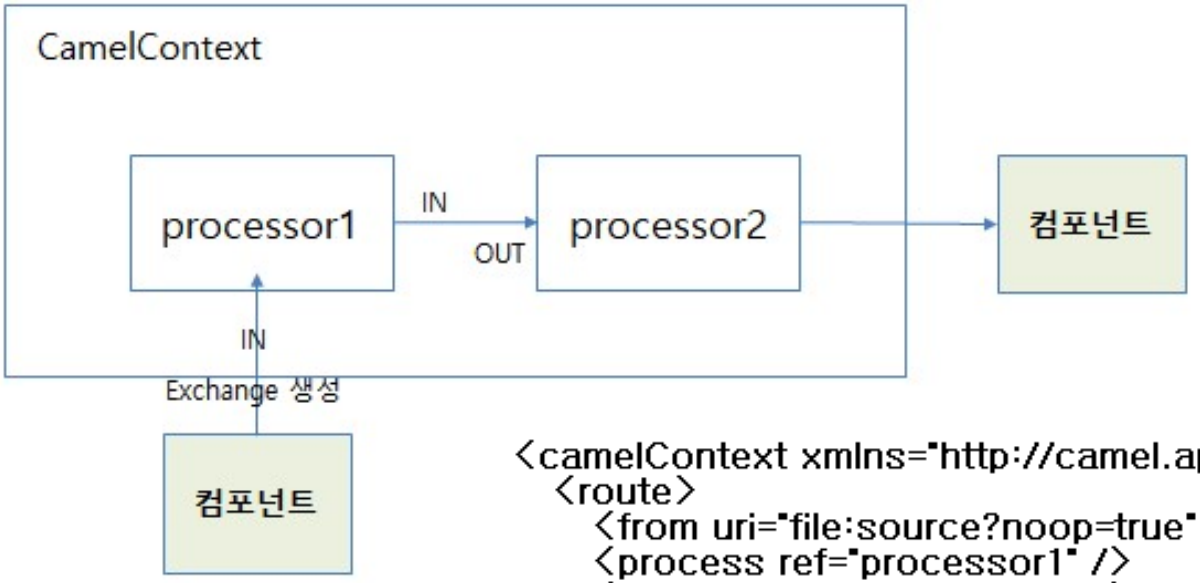
  - 시스템은 채널을 통하여 메시지를 송수신 가능
  - Endpoint는 채널의 끝을 모델링하여 추상화한 것으로 컴포넌트에 의해서 생성됨
  - 시스템들을 통합하게 해주는 중립적인 인터페이스 역할
  - 생산자와 소비자를 생성하는 팩토리 역할을 수행
  - URI는 Scheme, Context Path, Options으로 구성됨.



Endpoint 구성(예)



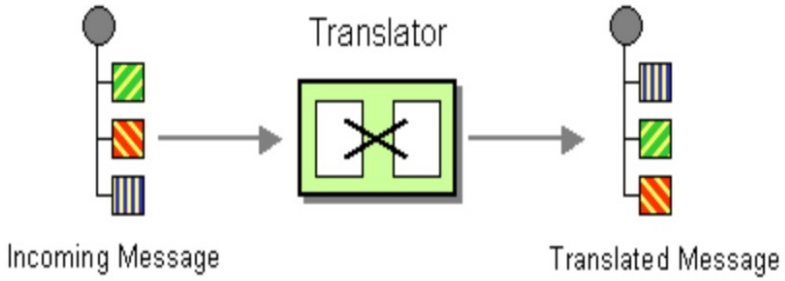
- **Processor의 개념**
  - 시스템간의 연동에는 메시지를 받을 후에 수신 시스템으로 보내기전에 처리 수행
  - 송신 시스템으로부터 받은 메시지를 수신 시스템에 보내기 전에 처리를 수행하는 부분을 Processor라 함
  - EIP에 정의된 라우팅(Routing), 메시지 변환(Transformation), 메시지 중계(Mediation), 검증(Valiadtion)등이 구현되어 있고 사용자 정의 processor를 구현할수 있음




```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="file:source?noop=true" />
    <process ref="processor1" />
    <process ref="processor2" />
    <to uri="file:dest?fileName=output.csv" />
  </route>
</camelContext>
```

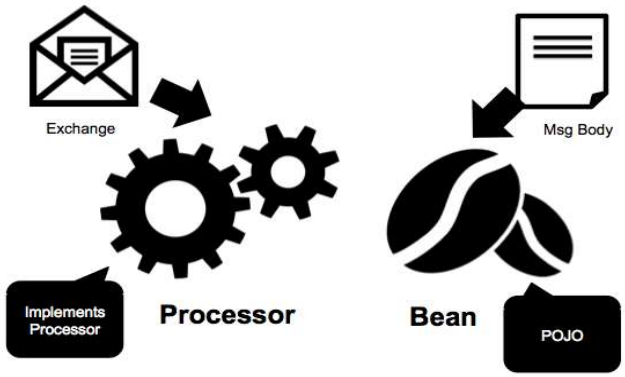
- **메시지 변환**

  - 포맷 변환 (Format Transformation) : 메시지의 포맷을 변경하는 작업 수행



  - JAXB
  - JSON
  - Google ProtoBuf
  - HL7
  - EDI/CSV 등

  - 타입 변환 (TypeConverter) : 데이터의 타입을 변경하는 작업을 수행



  - File
  - String
  - byte[]
  - InputStream/OutputStream
  - Document/Source (XML) 등

- **소비자(Consumer) 개념**

  - 생성자로 부터 메시지를 수신 후 Exchange를 생성하여 Camel 내부로 전달 이때 메시지 객체가 새로 생성

  - 소비자에는 두가지 모델 존재
    - Event-driven consumers
      웹 서비스 와 같이 Consumer는 특별한 메시징 채널(TCP/IP,JMS..)을 통해 클라이언트가 메시지를 보내는 것을 기다림 (웹서비스, HTTP, RMI 등)

    - Polling consumers
      스케줄을 통해 주기적으로 메시지를 가져감, 단 이전 메시지가 처리 완료 되지 않았다면 추가적으로 메시지를 가져가지 않음 (FTP, Email, File 등)

- **생상자 (Producer) 개념**

  - Endpoint에 메시지를 생성, 전달할 수 있는 개체

6. 활용예제

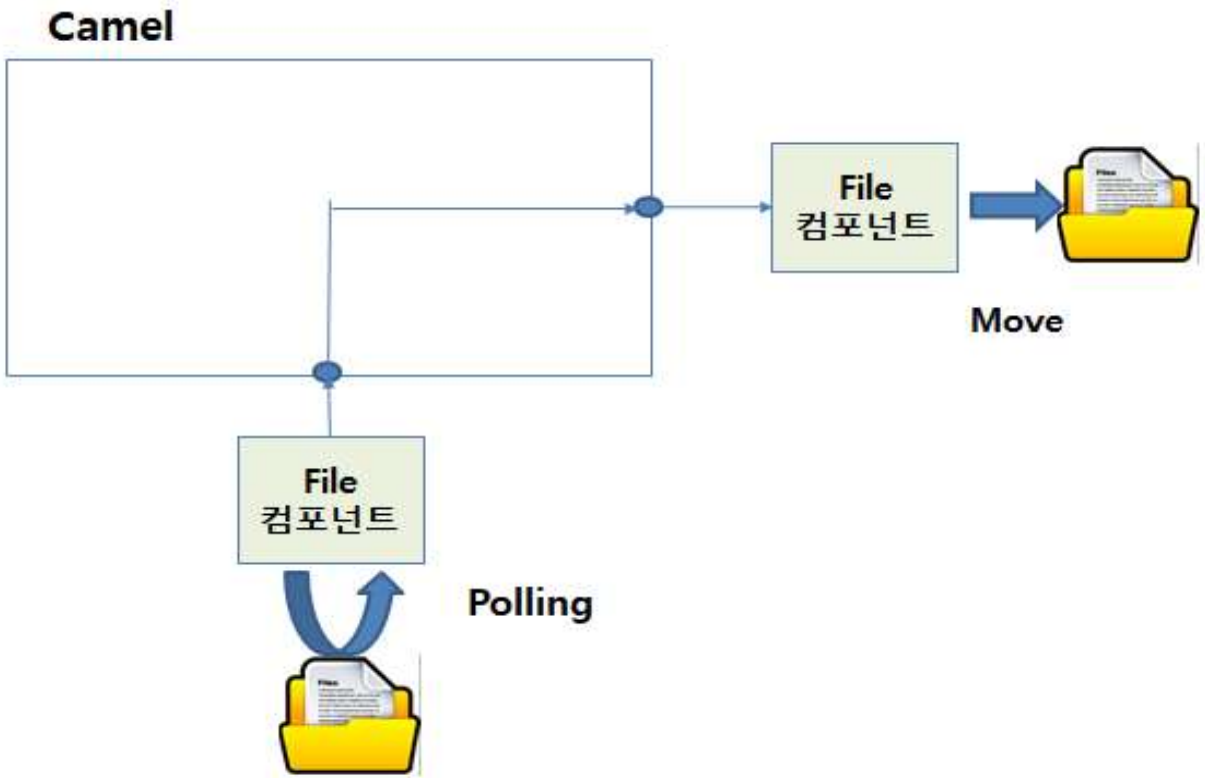
세부 목차

6. 활용예제

6.1 Java DSL을 통한 파일 이동(1/3)

1. 프로그램 개요

- Camel을 이용하여 특정 디렉터리 폴더의 파일을 폴링하여 파일이 존재하면 다른 디렉터리로 복사하여 이동한다.



2. RouteBuilder 클래스 작성

```
public class CamelRouteBuilder extends RouteBuilder{

    @Override
    public void configure() throws Exception {
        from("file://input?noop=true").to("file://output");
    }
}
```

CamelRouteBuilder.java

- CamelRouteBuilder.java 클래스 작성
- JAVA DSL을 통해 라우팅 설정
- RouteBuilder 클래스를 상속하여 configure() 메서드를 구현하여 작성
- 라우팅 설정은 from (“”) - to(“”) 메시드 체인을 통해서 구현
- file:// uri 를 통하여 파일 디렉터리 관련 컴포넌트를 사용
- 라우팅이 from input 폴더를 폴링 후 파일이 있을 시에 to output 폴더로 보내주는 라우팅 설정

3. 메인 프로그램 작성

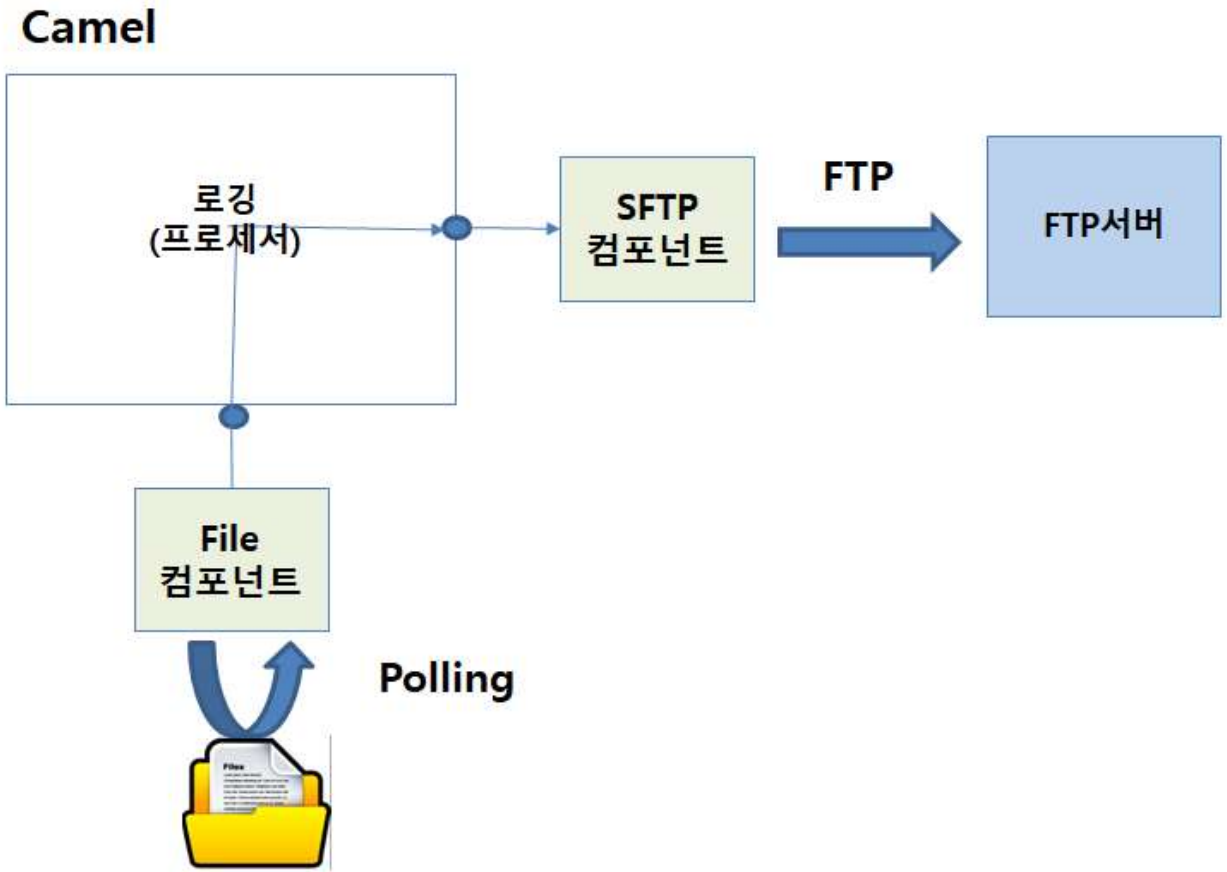
- FileRouter.java 메인 실행 클래스 작성
- Camel Context 구현체 중 default 구현체를 이용하여 Camel 엔진을 구동
- 이전 장에서 구현한 CamelRouteBuilder 클래스의 인스턴스를 addRoutes() 메서드를 통하여 CamelContext에 라우트 추가
- 프로그램을 실행하면 Input 폴더의 파일들이 Output 폴더로 이동한다.

```
public class FileRouter {

    public static void main(String[] args) {
        //구현한 RouteBuilder
        CamelRouteBuilder routeBuilder = new CamelRouteBuilder();

        CamelContext ctx = new DefaultCamelContext();
        try {
            //구현한 routeBuilder를 추가
            ctx.addRoutes(routeBuilder);
            //Camel Context 호출
            ctx.start();
            //Polling
            Thread.sleep(1000*1000L);

        }catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

FileRouter.java

1. 프로그램 개요

- Camel을 이용하여 특정 디렉터를 폴링하여 파일이 존재하면 SFTP 프로토콜을 이용하여 파일 전송한다. (전송도중 전송파일의 크기 로깅)



2. Routing XML 파일 생성

- XML DSL (Spring DSL)을 통하여 라우팅 설정(sftp_client.xml 파일 생성)
- 라우팅 설정은 XML 내의 <route> 엘리먼트 내애서 <from url = ""> <to url="">를 이용하여 설정
- from의 설정은 File 컴포넌트(file://)를 이용하고 to 설정은 SFTP 컴포넌트(sftp://) 이용
- from과 to 사이의 process 엘리먼트를 설정하여 파일 로깅을 위한 프로세서 설정

```
<?xml version="1.0" encoding="UTF-8"?>

 <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

   <!--  중간 단계에서 데이터 처리를 위해 사용할 프로세서 bean 지정-->
   <bean name="measureService" class="kr.co.bizframe.camel.test.processor.TestMeasureProcessor" />

   <camelContext xmlns="http://camel.apache.org/schema/spring">
       <route id="fileTest1">
           <!– 시작부분 -->
           <from uri=file://input?noop=true/>
           <process ref="measureService"/>
           <!– 수신 부분-->
           <to uri="sftp://계정명@주소:포트/camel_test/output?password=비밀번호"/>
       </route>
   </camelContext>
</beans>
```

sftp_client.xml

3. 메인 프로그램 작성

- FtpClient.java라는 클래스 작성
- CamelContext를 Spring에서 이용하기 위해서 래핑한 클래스인 rg.apache.camel.spring.Main 클래스를 통하여 CamelContext 이용
- 앞서 작성한 sftp_client.xml을 Main 인스턴스의 setApplicationContextUri() 메서드를 통하여 라우팅 설정
- 해당 프로그램을 실행하면 input 폴더의 파일을 sftp로 파일이 업로드 되는 것을 볼 수 있음

```
public class FtpClient {

    public void send() throws Exception{
        Main main = new Main();
        try{
            String ctxPath = "sftp_client.xml";
            // Camel 생산자(발신자) 객체 획득
            main.setApplicationContextUri(ctxPath);
            // Camel 컨텍스트 실행
            main.start();
            Thread.sleep(1000000L);

        }catch(Exception e){
            main.stop();
            e.printStackTrace();
        }
    }

    public static void main(String[] args) throws Exception{
        FtpClient client = new FtpClient();
        client.send();
    }
}
```

FtpClient.java

4. 프로세서 프로그램 작성

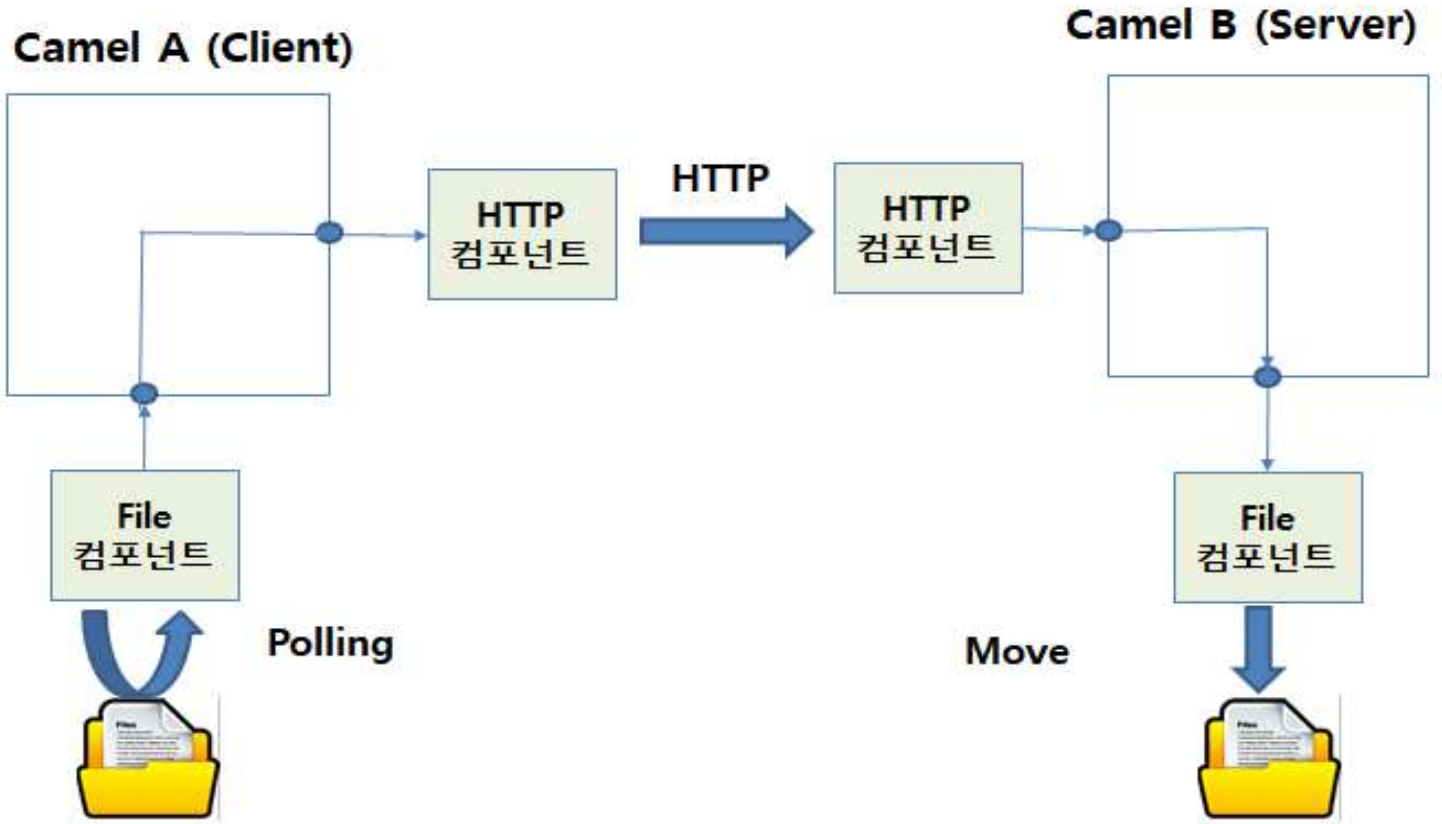
```
import org.apache.camel.Exchange;

public class MeasureProcessor implements Processor  {
    private static final org.apache.log4j.Logger log = org.apache.log4j.Logger.getLogger(MeasureProcessor.class);
    @Override
    public void process(Exchange exchange) throws Exception {
        log.debug("==>process");
        try {
            log.debug("byte[].length : " +exchange.getIn().getBody(byte[].class).length);
        } catch (Exception e) {
            log.error(e.getMessage(), e);
        }
    }

}
```

MessureProcessor.java

- MeasureProcessor.java는 Camel의 사용자 정의 프로세서에 해당
- 해당 예제는 파일에 대한 크기를 로깅하는 프로세스
- Process() 메서드 내에서 원하는 로직을 추가 가능(로깅, 변환)
- sftp_client.xml 파일 내에서 <process ref="measureService"/> 로 <from> <to> 사이에 정해주면 해당 프로세서가 실행

1. 프로그램 개요

- HTTP 프로토콜을 이용하여 클라이언트에서 서버로 파일을 전송하는 프로그램으로 클라이언트, 서버 각각 camel을 이용하여 구현한다.



6. 활용예제

6.3 Spring DSL을 통한 http파일 전송(송신)(1/2)

1. 송신(클라이언트)측 Routing xml 생성

- XML DSL (Spring DSL)을 이용하여 라우팅 설정(http_client.xml 파일 생성)
- 라우팅 설정은 XML 내의 <route> 엘리먼트 내애서 <from url = “”> <to url=“”>를 이용하여 설정
- from의 설정은 File 컴포넌트(file://)를 이용하고 to 설정은 HTTP 컴포넌트(http://)를 이용
- to 부분에 파일을 보낼 서버의 HTTP endpoint 입력

```
<?xml version="1.0" encoding="UTF-8"?>

 <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

<!--파일 용량 체크 프로세서 선언-->
<bean name="measureService" class=“kr.co.nipa.camel.processor.MeasureProcessor" />

 <camelContext xmlns="http://camel.apache.org/schema/spring">
         <route id="fileTest">
              <from uri="file://input?noop=true" />
              <process ref="measureService"/>
              <to uri="http://camel.nipa.or.kr:10022/myService/"/>
        </route>
 </camelContext>
</beans>
```

http_client.xml

2. 송신(클라이언트)측 메인 프로그램 작성

- HttpClient.java라는 클래스 작성
- CamelContext를 Spring에서 이용하기 위해서 래핑한 클래스인 org.apache.camel.spring.Main 클래스를 이용하여 CamelContext를 이용
- 앞서 작성한 http_client.xml을 Main의 application context 에 설정하여 라우팅을 설정
- 해당 프로그램을 실행하면 input 폴더의 파일을 HTTP 프로토콜을 이용하여 서버로 전송

```
public class HttpClient {

    public void send(){

        try{
            String ctxPath = "http_client.xml";
            Main main = new Main();
            main.setApplicationContextUri(ctxPath);

            // Camel 컨텍스트 실행
            main.start();
            Thread.sleep(1000*1000L);

        }catch(Exception e){
            e.printStackTrace();
        }

    }

    public static void main(String[] argv){
        HttpClient ht = new HttpClient();
        ht.send();
    }

}
```

HttpClient.java

6. 활용예제

6.3 Spring DSL을 통한 http파일 전송(수신)(1/2)

1. 수신(서버) 측 Routing xml 생성

- XML DSL (Spring DSL)을 이용하여 라우팅 설정(http_server.xml 파일 생성)
- 라우팅 설정은 XML 내의 <route> 엘리먼트 내애서 <from url = “”> <to url=“”>를 이용하여 설정
- From 부분에 jetty 컴포넌트(HTTP 서블릿 컴포넌트)를 사용하여 HTTP 서버 설정
- To 부분은 파일 컴포넌트를 사용하여 파일을 받아서 처리할 디렉터리 설정

```
<?xml version="1.0" encoding="UTF-8"?>

 <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd">

<!--파일 용량 체크 프로세서 선언-->
<bean name="measureService" class=“kr.co.nipa.camel.processor.MeasureProcessor" />
 <camelContext id="camel" xmlns="http://camel.apache.org/schema/spring">
               <!--http server-->
               <route id="server_http">
                              <!--jetty Conponent를 사용해  http서버 로드-->
                              <from uri="jetty:http://camel.nipa.or.kr:10022/myService/"/>
                              <!--파일 용량 체크 프로세서-->
                              <process ref="measureService"/>
                              <!--수신 어플리케이션 Endpoint-->
                              <to uri="file://receive"/>
                  </route>
</camelContext>
</beans>
```

http_server.xml

2. 수신(서버) 측 메인 프로그램 작성

- HttpServer.java라는 클래스 작성
- CamelContext를 Spring에서 이용하기 위해서 래핑한 클래스인 org.apache.camel.spring.Main 클래스를 이용하여 CamelContext를 사용
- 앞서 작성한 http_server.xml을 Main의 application context 에 설정하여 라우팅을 설정
- 해당 프로그램을 실행하면 HTTP 클라이언트로부터 HTTP 프로토콜을 통해 전송받은 파일을 receiver 디렉터리로 파일을 이동하여 생성시킴

```
public class HttpServer {

    public void server(){

        try{
            String ctxPath = "http_server.xml";
            Main main = new Main();
            main.setApplicationContextUri(ctxPath);

            // Camel 컨텍스트 실행
            main.start();
            Thread.sleep(1000*1000L);

        }catch(Exception e){
            e.printStackTrace();
        }

    }

    public static void main(String[] argv){
        HttpServer ht = new HttpServer();
        ht.server();
    }

}
```

HttpServer.java

7. FAQ

Q **Camel은 ESB 인가요?**

A Camel 자체는 ESB라고 부르기 어려우며 메시지 라우팅 엔진 라이브러리라고 보시면 됩니다. 물론 ESB의 주요 엔진으로 사용될 수 있고 Camel을 이용하여 간단한 ESB 뿐만 아니라 모니터리어, HA 기능을 추가하여 엔터프라이급 ESB로 사용도 가능합니다.

Q **Camel을 사용시 라이선스는 어떻게 되나요?**

A Camel의 라이선스는 무료입니다. 다만, 기술지원서비스(개발지원서비스, 유지관리서비스)가 필요하신 사용자는 Camel에 대해서 컨설팅을 제공하는 업체를 선택하여 기술지원을 받을 수 있습니다.

8. 용어정리

용어	설 명
ESB (Enterprise Service Bus)	서비스들을 컴포넌트화된 논리적 집합으로 묶는 핵심 미들웨어이며, 비즈니스 프로세스 환경에 맞게 설계 및 전개할 수 있는 아키텍처 패턴
ActiveMQ	JMS (Java Message Service) 클라이언트와 함께 Java로 작성된 오픈 소스 메시지 브로커
Router	둘 혹은 그 이상의 네트워크와 네트워크 간 데이터 전송을 위해 최적 경로를 설정해주며 데이터를 해당 경로를 따라 한 통신망에서 다른 통신망으로 통신할 수 있도록 도와주는 인터넷 접속 장비
JMS (Java Messaging Service)	자바 프로그램이 네트워크를 통해 데이터를 송수신하는 자바 API
Maven	자바 프로젝트를 위한 자동 빌드 툴로 아파치 라이선스로 배포

Open Source Software Installation & Application Guide

nipa 공개SW역량프라자